ARCTURUS
CONSTANT INNOVATION

APPLICARE 9.2.X

Applicare 9.2 Complete Manual

Observability, APM and AIOps for cloud, on-premises and air-gapped environments

Version 9.2

Arcturus Technologies Inc. · Published November 3, 2026
Generated from the Applicare master documentation. 210 of 210 articles were migrated from earlier documentation and are pending technical review.
The latest version is always available at https://helpdesk.arcturustech.com/hc/en-us

Contents

GETTING STARTED

Installation & Getting started with Applicare 10
Applicare Architecture and Components 11
Key Concepts 12
Applicare APM Deployment: Infrastructure Pre-requisites for POC 14
Applicare APM Deployment: Infrastructure Pre-requisites for Production monitoring 15
Applicare Configuration 17

CONFIGURATION GUIDES

Monitor .NET Error Transactions with Applicare and OpenTelemetry (Offline Mode) 20
Monitor .NET Error Transactions with Applicare and OpenTelemetry 23
Monitoring Containerized .NET Applications with Applicare and OpenTelemetry 26
Monitor .NET Application 29
Manually Installing the .NET Applicare Agent 32
Set up the .NET agent to run automatically using Task Scheduler 33
Monitor .NET applications running on Windows using Opentelemetry 36
Applicare Java Requirements 38
Install SingleAgent in the Java Application 39
Finding the Java Installation Directory 40
Restarting the SingleAgent 41
Customize Agent heap memory and jvm arguments 52
Java Application Server Configuration Settings 54
Include the server name and JVM arguments in the Gradle Jib build configuration. 57
Install Applicare agent in Java app 58
Installing Applicare SingleAgent using Wrapper 59
Install the Applicare agent using Wrapper on a Linux system 60
Install SingleAgent in a Docker Container 61
Install Applicare SingleAgent in Docker container using script 65
Install ApplicareSingleAgent in a Docker Container 66
Deploy SingleAgent in a Docker Container 68
Agent configuration in docker container 71
Install Applicare SingleAgent in a Java Jar Application Docker Container 73

.NET in Docker Image 76
Applicare deployed in a Docker container with Derby database 79
Install Applicare SingleAgent on Ubuntu 81
Install Applicare SingleAgent on Alpine Linux 83
Install Applicare SingleAgent on CentOS 85
Install Applicare SingleAgent in DebianOS 87
Install Applicare SingleAgent in a RedHat Enterprise Linux (RHEL) Docker Image 89
Configuring an agent within an Nginx container using a (docker-compose.yml) file 91
Agent configuration in Nginx container 97
Deploy the agent in the Nginx-proxy container 102
.NET monitoring in Kubernetes 108
Install OTel agent in Kubernetes cluster 110
Local Kubernetes Monitoring 111
Remote Kubernetes Monitoring 117
Installing Applicare SingleAgent in Kubernetes Containers Using a Script 127
Kubernetes Monitoring 131
Monitoring Nginx in a Kubernetes 139
Deploy Go auto instrumentation in Kubernetes 144
Applicare OTel Collector installation on Linux 146
Applicare Metrics, Trace and Log Collector installation on Windows 148
Monitor Traces and Metrics in Containerized .NET Applications with Applicare and OpenTelemetry 151
Applicare Installation for Linux 158
Run the Applicare Controller as a service on Linux 163
Starting Applicare Controller on a Linux system 164
Applicare Installation for Windows 165
Update the Applicare Controller 171
Applicare Updates 173
Deploying the Applicare Service Pack 174
Starting the Applicare Controller 175
Applicare Sentinel 177
Set up the Applicare Controller in Task Scheduler 178
Start the Controller in Task Scheduler 181
Start the Controller in command-line mode 185
Update Agents 186

Downloading Applicare Agents 189
Installing the SingleAgent on Windows 190
Manual Installation of Applicare SingleAgent on Linux 191
Uninstalling an agent and scheduled task on Windows 192
Remote Deployment for Applicare 193
Deploying the agent on Linux and Windows via the Remote Deployment feature 197
Standalone Agent for BTC Collector 200
Applicare Agent's Impact on Application Performance 202
Starting SingleAgent on Linux 203
SingleAgent Installation Using Command Mode for Linux 205
Starting SingleAgent on Windows 207
Installing SingleAgent via Command Line on Windows 208
Deploying the agent on Windows using an Ansible script 210
SingleAgent Remote deployment in Windows using Powershell 213
Setting Up Automatic Start of Applicare Agent with Windows Task Scheduler 217
Task creation to stop the SingleAgent 218
Installing the Agent for Tomcat, JBoss, WebLogic, WebSphere, and Other Java-Based Monitoring 225
Domain Configuration 226
WebSphere Monitoring 227
Wildfly Monitoring 229
Monitoring JBoss on Linux 230
Installing SingleAgent for AppServer on Linux 231
SingleAgent Installation for Appserver using command Mode for Linux 232
Installing Applicare SingleAgent on WebLogic running on Linux 233
Installing SingleAgent for Appserver on Windows using Command Line 234
Deploying SingleAgent for WebLogic on Windows 235
Apache Monitoring on Linux 236
Apache server 2.4 Monitoring for Linux 237
Apache server 2.4 Monitoring for windows 243
Monitoring Nginx 248
Start Nginx using Task Scheduler 253
Monitoring Applications Hosted on IIS with Applicare 256
Automate Daily Backups of Settings Tables Records to an Archive Database 258
Automate Daily Batch Backups of few Tables Records to an Archive Database 264

Automate Daily Batch Backups of Init profiling hist Table Records to an Archive Database 270
Applicare MySQL Database Backup (Windows) 273
Set up the Applicare schema in MySQL 275
MySQL 8 installer for windows 276
Installing MySQL on Linux 277
Configuring SQL Mode for MySQL in Applicare 279
MySQL Database Monitoring 282
Increasing Memory for Derby 288
Create a new user account and designate the relevant permissions 289
MSSQL manual Trace deletion 290
Monitoring for MSSQL Server Always On Availability Groups 291
Monitoring for MS SQL Server databases 295
Creating a new user for SQL Server authentication and grant the necessary permissions. 302
Oracle Database Monitoring 303
PostgreSQL Database Monitoring 311
MongoDB database monitoring 317
Redis Monitoring 321
Node.JS Monitoring for Linux 324
Manual Instrumentation for Node.js (Docker image) 327
Node.JS Monitoring for Windows 328
PHP Application Monitoring 330
Configure PHP application 332
Manually instrument a Go application using OpenTelemetry and Applicare 334
Automatically instrument a Go Application running in Docker with OpenTelemetry and Applicare (With additional container) 339
COBOL Monitoring with Applicare 341
COBOL Monitoring 345
AWS Monitoring 346
Configuring AWS Monitoring 348
Creating a DB parameter Group for AWS RDS MySQL 350
Configuring Applicare with OKTA using SAML 357
Manage LDAP User Roles 378
Active Directory/LDAP Settings 380
Role of Single Sign-On Users 381
User Role Management 384

Email/SMTP Configuration in Applicare 386
Set Up SMTP for Yahoo Mail 389
Configure SMTP for Gmail 391
Disable Auto Add SNMP on controller 397
SQL injection detection in Applicare 398
Vulnerability Scanner Setup 400
REST API Calls Using Token-Based Authentication 401
REST API Calls Using X.509 Certificate-Based Authentication 409
Applicare Data Security 415
Manage Applicare Data 416
Export/Import Applicare Configuration 417
Remote Commands 418
Applicare Information 425
Compressing log files on Windows 427
LAMA configuration in Applicare 428

USING APPLICARE

App Monitoring Dashboard 436
Operations Center 442
Overview 444
Server Comparison 446
Events Calendar 447
Enterprise 448
Unattached servers 453
Log Analyzer 454
Alerts 465
Custom Search in LogAnalyzer 470
Date and Regex pattern format 475
LAMA 477
Business Transactions User Guide 493
Automatically add KBT configuration 505
Download IntelliTrace report 507
Configuration for KBT with parameter 510
Server Analyzer 514

Applicare Docker Monitoring 529
Operating System Analyzer 532
JVM Analyzer 534
GC Analyzer 535
Network Analyzer 537
Sensor Analyzer 540
Applicare Custom JMX Analysis: Create a Sensor 541
Monitoring MSSQL Metrics with DB Query Sensors 543
Server Availability Monitoring 545
User Guide for Performance Comparison 547
Applicare IntelliSense 551
Arcturus Insights - ArcIn 554
Applicare Genie 556
ArcIn Database Configuration 558
Set Up Predictive Analysis in Applicare on Linux 560
End User Experience Monitoring (EUM)/User Experience Monitoring (UEM) Analyzer 563
EUM Real-Time Transaction Monitoring 573
Applicare RUM/EUM Integration Script 577
UEM Injection through the Applicare Reverse Proxy Server 578
Apache User Experience 585
UEM IP Address Translation 589
iOS Mobile App Performance Monitoring 590
Monitoring performance for Android mobile apps 596
Measuring User Satisfaction - Apdex 601

TROUBLESHOOTING

Error: Unable to Create Task in Task Scheduler 604
Issue with Java during installing wrapper on Linux 606
Java issues encountered during wrapper installation 607
Enable read access to IIS logs 608

RELEASE NOTES

Applicare Release Notes for Version 9.2.9 612
Applicare Release Notes for Version 9.2.8.4 613
Applicare Release Notes for Version 9.2.8.3 614

Applicare Release Notes for Version 9.2.8.2 615
Applicare Release Notes for Version 9.2.8.1 616
Applicare Release Notes for Version 9.2.7.9 617
Applicare Release Notes for Version 9.2.7.8 618
Applicare Release Notes for 9.2.7.7 619
Applicare Release Notes for 9.2.7.6 620
Applicare Release Notes for Version 9.2.7.5 621
Applicare Release Notes for Version 9.2.7.4 622
Applicare Release Notes for Version 9.2.7.3 623
Applicare Release Notes - Version 9.2.7 624
Applicare Release Notes - Version 9.2.6 625
Applicare Release Notes for 9.2.5 626
Applicare Release Notes - Version 9.2.4 627
Applicare Release Notes for 9.2.2 628

SUPPORT & TICKETS

How to get support 630
What to include in a support ticket 631

PART

Getting Started

Architecture, key concepts, prerequisites and first installation.

GETTING STARTED

Installation & Getting started with Applicare

Applicare 9.2 · Updated February 28, 2025 · controller, linux, windows · **Pending technical review**

Setting up Applicare monitoring is straightforward. Follow the steps below to get started:

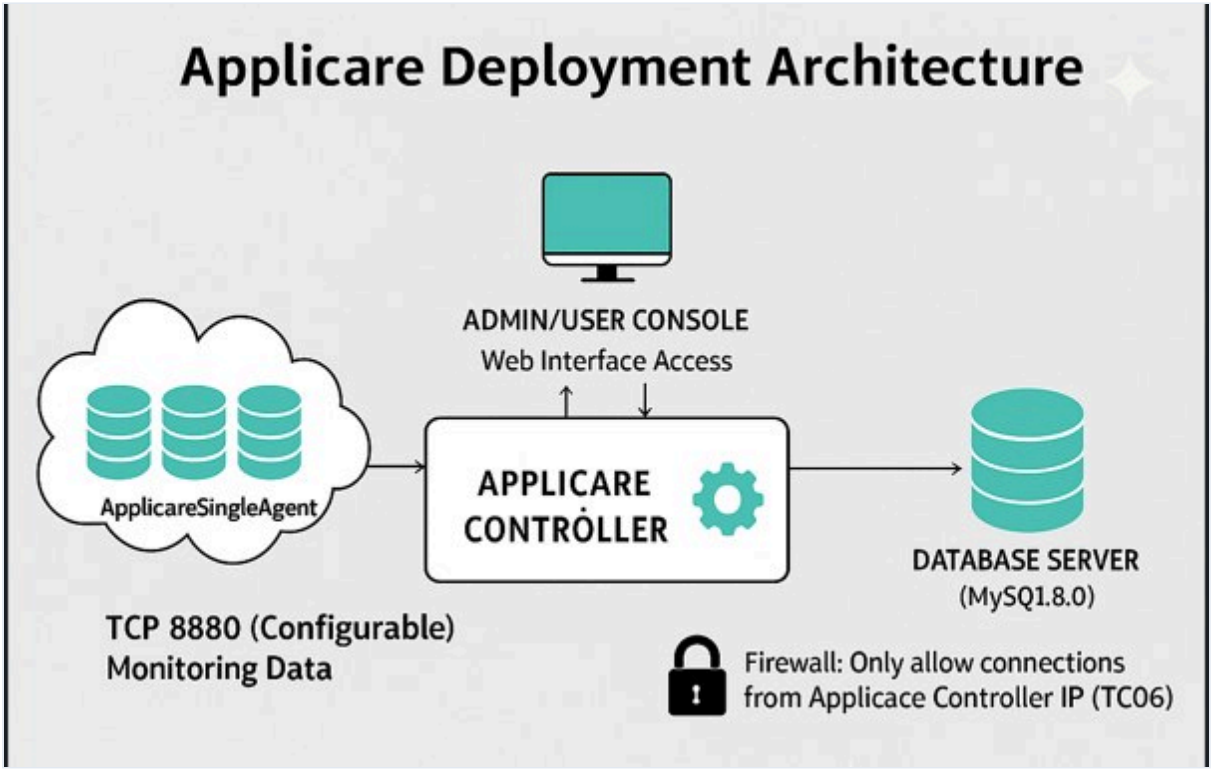
- Applicare Controller Installation
  - Windows
  - Linux
- Installing Applicare Single Agent
- Monitoring Databases
- Monitoring WildFly
- Monitoring JBoss
- Monitoring IIS
- Monitoring Node.js
- Monitoring Nginx
- Monitoring Tomcat
- Monitoring WebSphere
- Monitoring Weblogic
- Monitoring Java Application
- Monitoring Apache (httpd)
- Monitoring AWS RDS MySQL
- Adding collector/s (BTM, UEM) for scalability
- Monitoring Java Application in Docker for App Servers
- Monitoring Java Application in Kubernetes for App Servers
- Monitoring Dotnet Applications hosted on IIS
- Monitoring Kubernetes
- Monitoring AWS

Monitoring with Applicare

GETTING STARTED

Applicare Architecture and Components

Applicare 9.2 · Updated November 4, 2025 · controller · **Pending technical review**



This architectural diagram outlines the roles of the core Applicare components and their communication:

- **Applicare SingleAgent:** Deployed on monitored servers to **collect** performance metrics from diverse technologies.
- **Applicare Controller:** The **central hub** that receives, processes, and manages the data streamed from the SingleAgent.
- **Applicare Database:** The **storage layer** where all collected performance metrics are securely retained.
- **Applicare UI:** The user-facing tool for **visualization, analysis**, and configuration of your monitoring environment.

GETTING STARTED

Key Concepts

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

What is IntelliSense?

IntelliSense proactively monitors, analyzes and notifies you of potential outage conditions.

IntelliSense monitors for patterns that could lead to an outage and provides warnings before they occur.

With IntelliCheck, there's no need to set thresholds manually. IntelliCheck leverage a built-in knowledge base to identify conditions that could lead to outages or performance degradation.

IntelliSense automates root-cause analysis. Applicare automatically detects server failures in production, performs an analyzes of the server state, and generate a root-cause analysis report instantly. IntelliSense will generate an email that not only informs you of what failed but explains why it happened.

This ensure you have all the necessary information, even if the incident occurred happened in the middle of the night while you were asleep. Applicare quickly analyzes thousands of matrices and delves into transaction-level details to determine the root-cause. It identifies the culprit transactions, exact code and/or SQLs.

On top of it, IntelliSense is extremely un-intrusive way to monitor. The IntelliSense technology reduces overhead, because it is designed from ground up with high performance and least impact in mind. The overhead in most cases is so low that it is almost immeasurable.

What is an Analyzer?

Analyzers provide historical views of crucial monitored objects state information.

Applicare automatically stores crucial information about the state of monitored objects for historical analysis. It provides you with intuitive and extremely easy-to-use data-mining capabilities. Starting with a 1-day view of a server's performance, you can drill down to a particular time frame in a matter of seconds

What is an Explorer?

Explorers make JMX, SNMP and other external information easily accessible directly within the Applicare Console.

While servers provide a wealth of useful data through JMX, it is not in an easy-to-use format. Applicare provides an easy-to-use interface for browsing MBeans information. Applicare explorers make JMX information of all monitored servers accessible directly within Applicare Console, without causing you to have to browse through many JSP pages to obtain the same.

What is a Sensor?

Sensors save JMX, SNMP and other external information values for graphical analysis, reporting and alerting over time.

Applicare enables you to create sensors on any property_._ A sensor is a property whose value is recoded at specified intervals. Applicare provides an easy-to-use interface to graphically analyze data accumulated by Applicare sensors.

Monitoring with Applicare

GETTING STARTED

Applicare APM Deployment: Infrastructure Pre-requisites for POC

Applicare 9.2 · Updated December 19, 2025 · controller · **Pending technical review**

Applicare requires one machine deployment setup for performance.

Hardware & Software requirements for Applicare Controller machine:

Applicare installer	**Downloadable from the Applicare support site.**
Disk Space	200GB SSD
RAM	32 GB
CPU	Intel i7 6-core equivalent or higher
Java	Java 25 or any supported version
OS	Windows 10 or later / Windows Server 2012 or newer. (OR) Red Hat Enterprise Linux 6.x or newer (RHEL 8 recommended).
Permission	Login user must have administrative (Windows) or root/sudo (Linux) privileges.
Web Browser	Google Chrome (stable version recommended for Windows)

Port and Network Requirements

- **Agent to Applicare Controller:** Agents must be able to reach the **Applicare Controller's IP and Port**.
  - **Default Port: 8880** (Can be modified).
  - The Controller's VM IP address must be accessible to all agents.

Note:

The above hardware requirements are subject to revision, particularly when moving to a live or production environment, which will necessitate an increase in VM and resource allocation

GETTING STARTED

Applicare APM Deployment: Infrastructure Pre-requisites for Production monitoring

Applicare 9.2 · Updated December 19, 2025 · controller · **Pending technical review**

Applicare requires a two-machine deployment setup for **security** and **performance**:

- **Applicare Controller:** This machine hosts the Applicare controller.
- **Database Server:** This machine hosts the Applicare controller data store.

Security: Only the **Applicare Controller** is permitted to connect to the **Database Server**. All **Agents** communicate exclusively with the **Applicare Controller**, preventing direct agent-to-database connections and enhancing system security.

Hardware & Software requirements for Applicare Controller machine:

Applicare installer	**Downloadable from the Applicare support site.**
Disk Space	100GB SSD
RAM	32 GB
CPU	Intel i7 6-core equivalent or higher
Java	Java 25 or any supported version
OS	Windows 10 or later / Windows Server 2012 or newer. (OR) Red Hat Enterprise Linux 6.x or newer (RHEL 8 recommended).
Permission	Login user must have administrative (Windows) or root/sudo (Linux) privileges.
Web Browser	Google Chrome (stable version recommended for Windows)

Hardware & Software requirements for Applicare database machine:

Mysql	***MySQL Server 8.0***
Disk Space	1 TB SSD or higher (**Critical: Higher capacity recommended for production monitoring**).
RAM	64 GB or higher.
CPU	Intel i7 6-core equivalent or higher.
Supported OS	Windows 10 or later / Windows Server 2012 or newer. (OR) Red Hat Enterprise Linux 6.x or newer (RHEL 8 recommended).
Permission	Login user must have administrative (Windows) or root/sudo (Linux) privileges.

Port and Network Requirements

- **Agent to Applicare Controller:** Agents must be able to reach the **Applicare Controller's IP and Port**.
  - **Default Port: 8880** (Can be modified).
  - The Controller's VM IP address must be accessible to all agents.
- **Applicare Controller to Database Server:** The Applicare Controller must be able to reach the **Database Server's IP and Port**.
  - **Default Port: 3306** (MySQL default).

Note: The existing hardware requirements may undergo changes in the future. The addition of new agents or features could lead to revisions in the hardware requirements.

GETTING STARTED

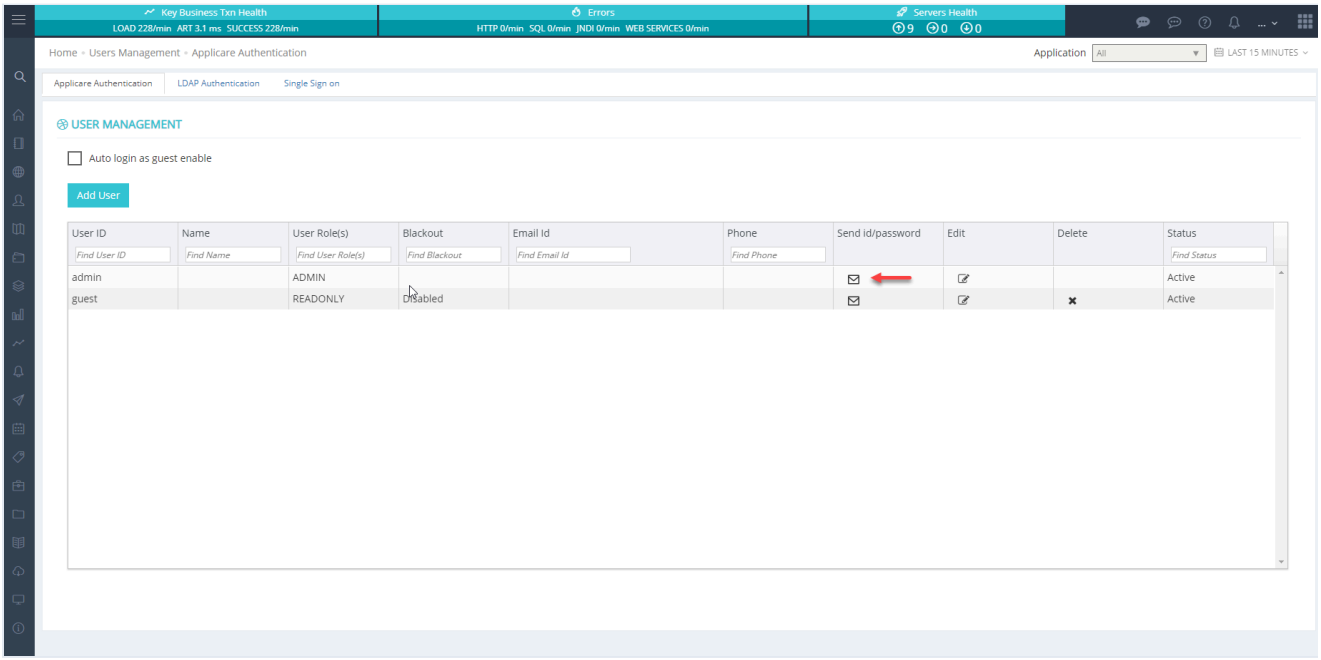
Applicare Configuration

Applicare 9.2 · Updated September 13, 2024 · controller · **Pending technical review**

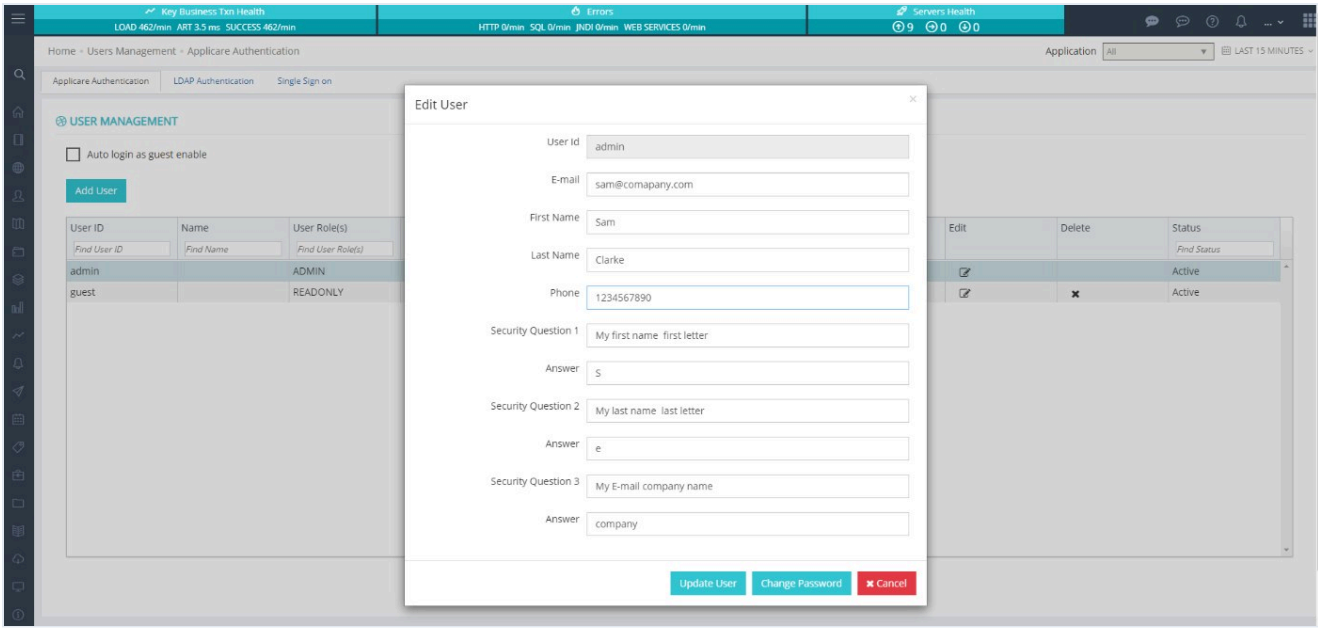
To start with Applicare, You'll need to update the default admin user password and configure the SMTP server settings.

To Change the admin Password

Go to Configuration -> User Management -> Applicare Authentication and choose "edit" for the 'admin' user.

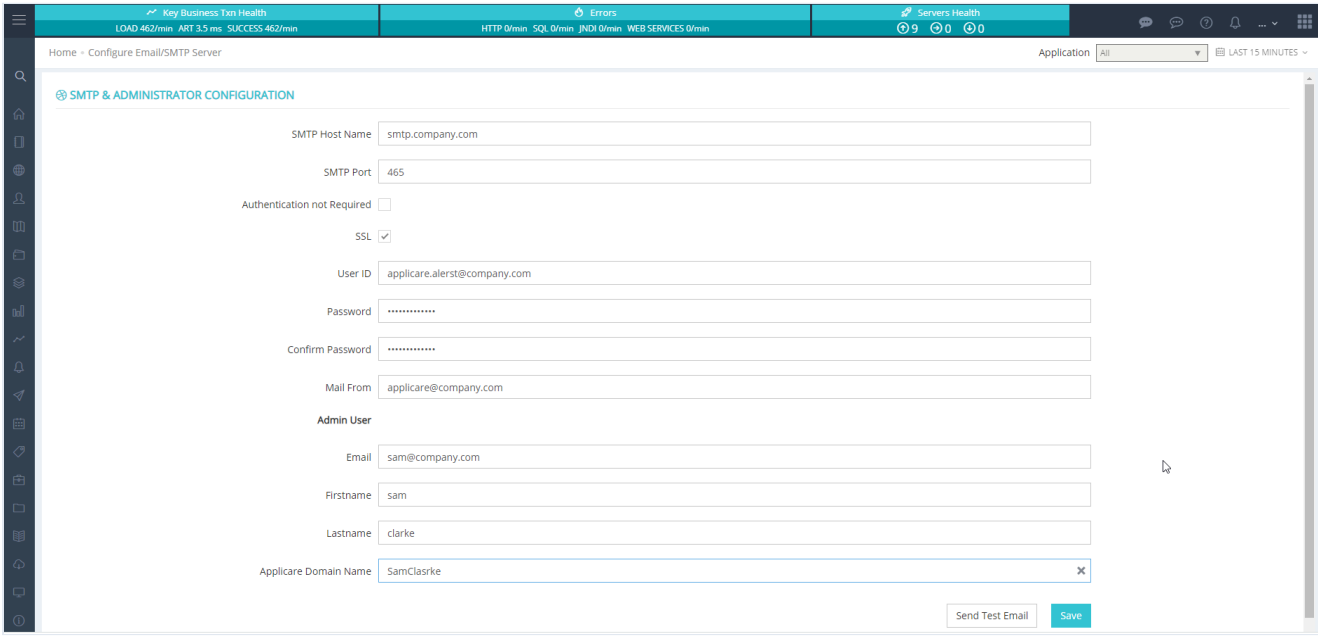


In the edit user details section, you can update the password along with other user information.



SMTP Parameters

This tab lets you set up the SMTP server settings. The Admin User email is where all alerts are sent, so choose this address carefully.



Monitoring with Applicare

PART

Configuration Guides

Install and configure agents, collectors and integrations per technology.

CONFIGURATION GUIDES › .NET

Monitor .NET Error Transactions with Applicare and OpenTelemetry (Offline Mode)

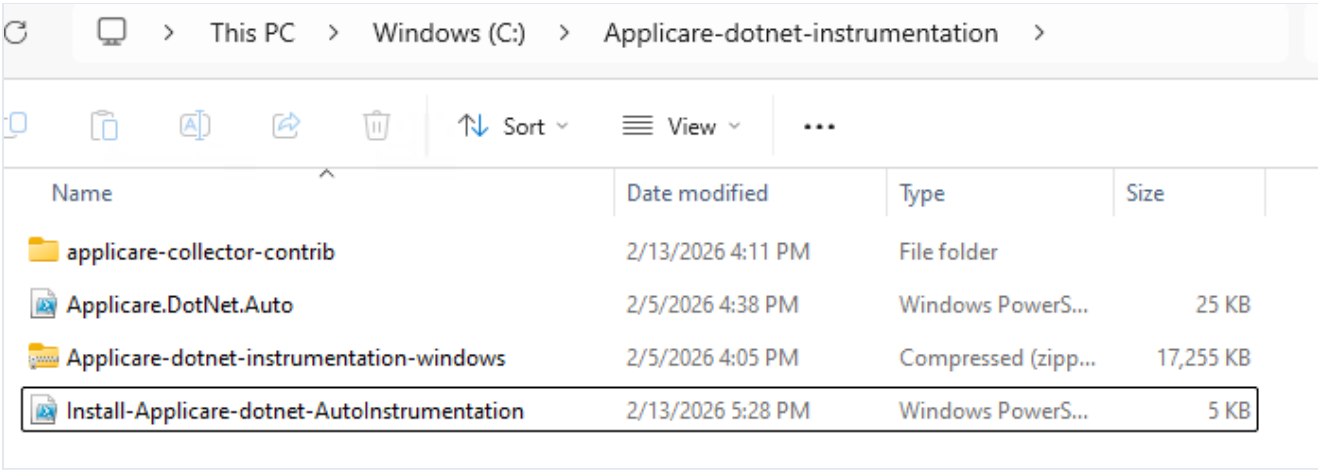
Applicare 9.2 · Updated February 13, 2026 · dotnet, opentelemetry, windows, iis · **Pending technical review**

NOTE: Before starting the configuration, make sure that the system has .NET Framework 4.6.1 or higher. Use the command below to check. Lower versions are not compatible with this Applicare .Net configuration.

```powershell
$PSVersionTable.PSVersion
```

Step 1: On the machine where the .NET application is running, download the **Applicare-dotnet-instrumentation** zip file. Then, create a folder named **Applicare-dotnet-instrumentation** in the **C - drive** and paste the zip file into that folder.

Step 2: Extract the zip file as shown in the following image.



Step 3: Open a **PowerShell window** as **Administrator** and navigate to the directory where extracted the zip file.

Step 4: Execute below command in powershell windows.

```powershell
.\Install-Applicare-dotnet-AutoInstrumentation.ps1
```

```
PS C:\Applicare-dotnet-instrumentation> .\Install-Applicare-dotnet-AutoInstrumentation.ps1
--- Starting Applicare .NET AutoInstrumentation Configuration ---
Step 1: Importing Applicare configuration module...
Successfully imported module.
Step 2: Installing Applicare binaries to C:\Program Files\Applicare .NET AutoInstrumentation...
Applicare binaries extracted successfully.
Step 3: Registering Applicare components for IIS...

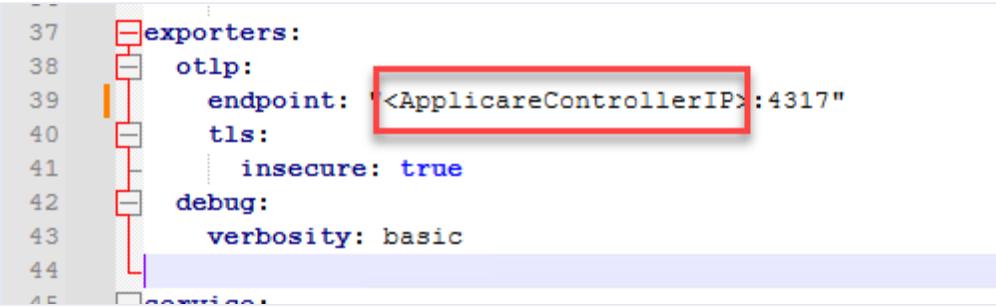
Attempting stop...
Internet services successfully stopped
Attempting start...
Internet services successfully restarted
IIS Registration completed.
Step 4: Configuring System Environment Variables for Applicare...
Environment variables have been updated.

============================================================
Applicare .NET AutoInstrumentation setup is successfully done.
Installation Path: C:\Program Files\Applicare .NET AutoInstrumentation
============================================================
Action Required: Please run 'iisreset' to start instrumentation.
PS C:\Applicare-dotnet-instrumentation>
```

Step 5: After the script is executed, execute the command below to **restart IIS.**

```
TEXT
iisreset
```

Step 6: After the IIS restart is complete, open the **config.yaml** file in the **applicare-collector-contrib** directory using Notepad. Navigate to **line 39** and replace **<ApplicareControllerIP>** with the actual IP address of the machine where the **Applicare Controller** is running, then **save** the file.

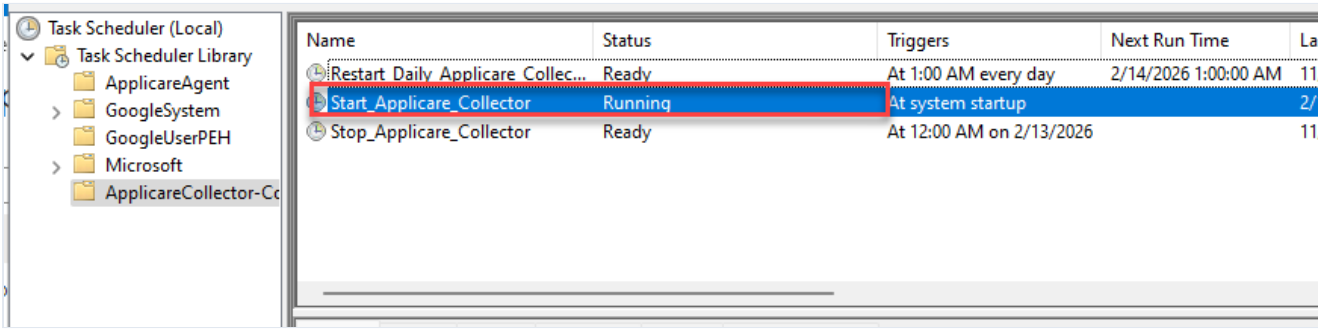


Step 7: Open the **Command Prompt** as **Administrator**, navigate to the **applicare-collector-contrib** directory, and execute the **create-task-applicare-collector-contrib.bat** file.

```
C:\Applicare-dotnet-instrumentation\applicare-collector-contrib>create-task-applicare-collector-contrib.bat
==================================================
  APPLICARE COLLECTOR - TASK SCHEDULER SETUP
==================================================
[1/4] Creating START task (Triggers on Boot)...
SUCCESS: The scheduled task "ApplicareCollector-Contrib\Start_Applicare_Collector" has successfully been created.
[2/4] Creating STOP task (Manual trigger)...
WARNING: Task may not run because /ST is earlier than current time.
SUCCESS: The scheduled task "ApplicareCollector-Contrib\Stop_Applicare_Collector" has successfully been created.
[3/4] Creating DAILY MAINTENANCE task (01:00 AM)...
SUCCESS: The scheduled task "ApplicareCollector-Contrib\Restart_Daily_Applicare_Collector" has successfully been created
.
[4/4] TRIGGERING COLLECTOR START NOW...
SUCCESS: Attempted to run the scheduled task "ApplicareCollector-Contrib\Start_Applicare_Collector".

SUCCESS: All tasks created and Collector is starting.
Folder: Task Scheduler Library -> ApplicareCollector-Contrib
==================================================
```

Note: Open the task scheduler and make sure that the task Start_Applicare_Collector is running.



The following port must be opened on the Applicare controller machine.

4317

CONFIGURATION GUIDES › .NET

Monitor .NET Error Transactions with Applicare and OpenTelemetry

Applicare 9.2 · Updated February 12, 2026 · dotnet, opentelemetry, windows, iis · **Pending technical review**

Note: The following configurations are verified for .NET Framework 4.8.1 Application only

Step 1: Open powershell in administrator mode and execute the following commands one by one.

```powershell
$module_url = "https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases/latest/download/OpenTelemetry.DotNet.Auto.psm1"
$download_path = Join-Path $env:temp "OpenTelemetry.DotNet.Auto.psm1"
Invoke-WebRequest -Uri $module_url -OutFile $download_path -UseBasicParsing

Import-Module $download_path
Install-OpenTelemetryCore
```

Step 2: Run the command below to enable **auto-instrumentation** for all **IIS-hosted** applications using the OpenTelemetry module.

```text
Register-OpenTelemetryForIIS
```

Step 3: Execute the following PowerShell command to configure the system-level environment variables required for .NET Auto-Instrumentation

```
TEXT
$vars = @{
   "COR_ENABLE_PROFILING"="1"
   "COR_PROFILER"="{918728DD-259F-4A6A-AC2B-B85E1B658318}"
   "COR_PROFILER_PATH"="C:\Program Files\OpenTelemetry .NET AutoInstrumentation\win-
x64\OpenTelemetry.AutoInstrumentation.Native.dll"
   "CORECLR_ENABLE_PROFILING"="1"
   "CORECLR_PROFILER"="{918728DD-259F-4A6A-AC2B-B85E1B658318}"
   "CORECLR_PROFILER_PATH"="C:\Program Files\OpenTelemetry .NET AutoInstrumentation\win-
x64\OpenTelemetry.AutoInstrumentation.Native.dll"
   "DOTNET_ADDITIONAL_DEPS"="C:\Program Files\OpenTelemetry .NET
AutoInstrumentation\AdditionalDeps"
   "DOTNET_SHARED_STORE"="C:\Program Files\OpenTelemetry .NET AutoInstrumentation\store"
   "DOTNET_STARTUP_HOOKS"="C:\Program Files\OpenTelemetry .NET
AutoInstrumentation\net\OpenTelemetry.AutoInstrumentation.StartupHook.dll"
   "OTEL_DOTNET_AUTO_HOME"="C:\Program Files\OpenTelemetry .NET AutoInstrumentation"
   "OTEL_DOTNET_AUTO_INSTALL_DIR"="C:\Program Files\OpenTelemetry .NET AutoInstrumentation"
   "OTEL_DOTNET_AUTO_LOGS_CONSOLE_EXPORTER_ENABLED"="none"
   "OTEL_DOTNET_AUTO_METRICS_CONSOLE_EXPORTER_ENABLED"="none"
   "OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED"="true"
   "OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT"="true"
   "OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT"="true"
   "OTEL_EXPORTER_OTLP_ENDPOINT"="http://<APPLICARE_CONTROLLER_IP>:4318"
   "OTEL_SERVICE_NAME"="<APPLICARE_AGENT_NAME>"
}; foreach($k in $vars.Keys) { [System.Environment]::SetEnvironmentVariable($k, $vars[$k],
[System.EnvironmentVariableTarget]::Machine) }
```

Note:

<APPLICARE_CONTROLLER_IP>: Replace this placeholder with the actual IP address or hostname of your Applicare controller instance.

<APPLICARE_AGENT_NAME>: The name of the agent as listed in the Applicare console.

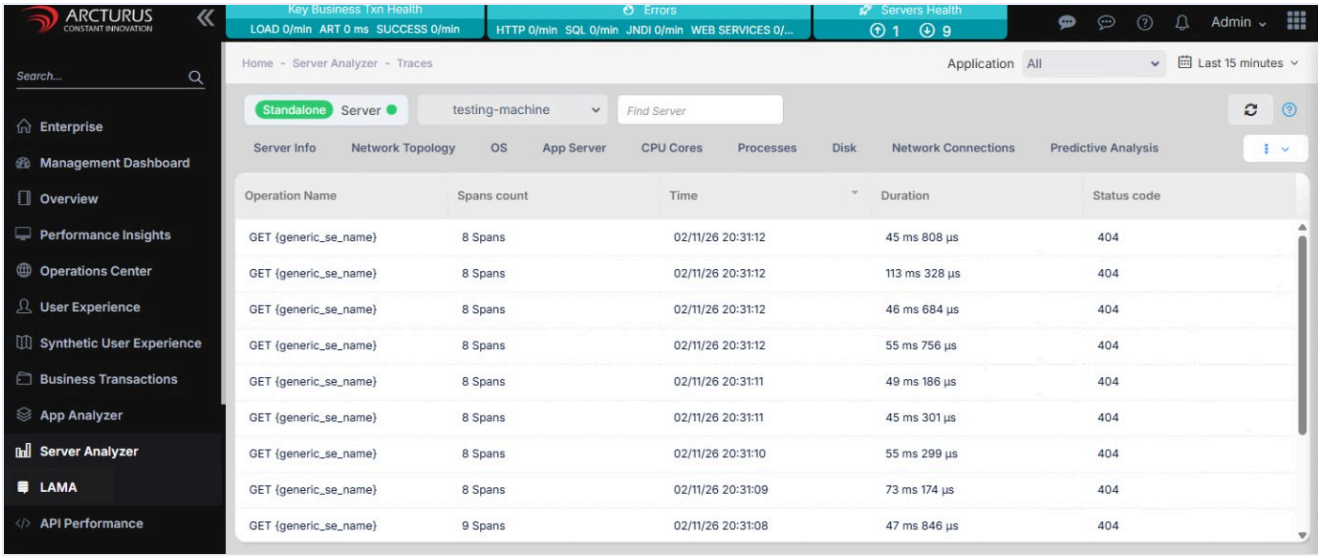
Step 4: Once the configuration is complete, an **IIS service restart is required**. Execute the below command .

```
TEXT
iisreset
```

Step 5: To start the OpenTelemetry Collector on the controller machine using the specified `config.yaml` *(sample file, available on the documentation site)* file, execute the following command.

```
TEXT
otelcol-contrib.exe --config=config.yaml
```

The collected traces will be shown in **Server Analyzer -> Telemetry Traces** screen in **Applicare console**.


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/...
Servers Health
1 9
Admin
Search...
Enterprise
Management Dashboard
Overview
Performance Insights
Operations Center
User Experience
Synthetic User Experience
Business Transactions
App Analyzer
Server Analyzer
LAMA
API Performance
Home - Server Analyzer - Traces
Application All
Last 15 minutes
Standalone Server
testing-machine
Find Server
Server Info
Network Topology
OS
App Server
CPU Cores
Processes
Disk
Network Connections
Predictive Analysis
Operation Name
Spans count
Time
Duration
Status code
GET {generic_se_name} 8 Spans 02/11/26 20:31:12 45 ms 808 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:12 113 ms 328 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:12 46 ms 684 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:12 55 ms 756 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:11 49 ms 186 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:11 45 ms 301 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:10 55 ms 299 µs 404
GET {generic_se_name} 8 Spans 02/11/26 20:31:09 73 ms 174 µs 404
GET {generic_se_name} 9 Spans 02/11/26 20:31:08 47 ms 846 µs 404

CONFIGURATION GUIDES › .NET

Monitoring Containerized .NET Applications with Applicare and OpenTelemetry

Applicare 9.2 · Updated July 14, 2025 · dotnet, docker, opentelemetry · **Pending technical review**

To enable OpenTelemetry auto-instrumentation for your .NET application and send traces to Applicare, you will need to apply configuration changes in both your **Dockerfile** and your application's **YAML deployment script** (e.g., Kubernetes Deployment, Docker Compose file, or other container orchestration tool's configuration).

Step 1: Modify your Dockerfile

Add the following lines to your application's Dockerfile. These commands will install necessary utilities, download the OpenTelemetry .NET auto-instrumentation setup script, execute it.

Note: The `apt-get` commands assume a Debian/Ubuntu-based base image (e.g., `mcr.microsoft.com/dotnet/aspnet:8.0-alpine` would use `apk add --no-cache` instead). Adjust package manager commands as per your base image.

```
TEXT
ENV OTEL_AGENT_VERSION=1.9.0
ENV DOTNET_EnableDiagnostics=1
ENV OTEL_DOTNET_AUTO_HOME=/otel-dotnet-auto
ENV DOTNET_SHARED_STORE=$OTEL_DOTNET_AUTO_HOME/store
ENV
DOTNET_STARTUP_HOOKS=$OTEL_DOTNET_AUTO_HOME/net/OpenTelemetry.AutoInstrumentation.StartupHook
.dll
ENV DOTNET_ADDITIONAL_DEPS=$OTEL_DOTNET_AUTO_HOME/AdditionalDeps
ENV OTEL_TRACES_EXPORTER=otlp
ENV OTEL_DOTNET_AUTO_LOGGING_ENABLED=true
ENV OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT=true
ENV OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT=true
ENV COMPlus_EnableDiagnostics=1
ENV DOTNET_EnableDiagnostics=1
ENV DOTNET_ROLL_FORWARD=LatestMajor

RUN apt-get update && apt-get install -y unzip curl
ADD https://github.com/open-telemetry/opentelemetry-dotnet-
instrumentation/releases/download/v1.9.0/otel-dotnet-auto-install.sh /otel-install.sh

RUN chmod +x /otel-install.sh && \
    /otel-install.sh
```

Step 2: Configure Environment Variables in your YAML Deployment

Add the following environment variables to the container definition section of your application's YAML deployment file (e.g., under `env:` in a Kubernetes `Deployment`'s container specification, or `environment:` in a Docker Compose `service`). These variables instruct the OpenTelemetry auto-instrumentation how to behave and where to send the telemetry data.

```
TEXT
OTEL_DOTNET_AUTO_HOME="/otel-dotnet-auto"
DOTNET_SHARED_STORE="/otel-dotnet-auto/store"
DOTNET_STARTUP_HOOKS="/otel-dotnet-
auto/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll"
DOTNET_ADDITIONAL_DEPS="/otel-dotnet-auto/AdditionalDeps"
OTEL_TRACES_EXPORTER="otlp"
OTEL_EXPORTER_OTLP_ENDPOINT="http://<APPLICARE_CONTROLLER_IP>:4318"
OTEL_SERVICE_NAME="<APPLICARE_AGENT_NAME>"
OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT="true"
OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT="true"
```

Before deploying your application, ensure you replace the following placeholders:

- `<APPLICARE_CONTROLLER_IP>`: Replace this with the actual IP address or hostname of your Applicare controller instance.
- `<APPLICARE_AGENT_NAME>`: Provide a unique, descriptive name that will help you identify this specific container or application within the Applicare monitoring console.

Sample Dockerfile with the above changes.

```
# See https://aka.ms/customizecontainer to learn how to customize your debug container and how Visual Studio uses this Dockerfile to build your imag

# This stage is used when running from VS in fast mode (Default for Debug configuration)
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base

ENV OTEL_AGENT_VERSION=1.9.0
ENV DOTNET_EnableDiagnostics=1
ENV OTEL_DOTNET_AUTO_HOME=/otel-dotnet-auto
ENV DOTNET_SHARED_STORE=$OTEL_DOTNET_AUTO_HOME/store
ENV DOTNET_STARTUP_HOOKS=$OTEL_DOTNET_AUTO_HOME/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll
ENV DOTNET_ADDITIONAL_DEPS=$OTEL_DOTNET_AUTO_HOME/AdditionalDeps
ENV OTEL_TRACES_EXPORTER=otlp
ENV OTEL_DOTNET_AUTO_LOGGING_ENABLED=true
ENV OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT=true
ENV OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT=true
ENV COMPlus_EnableDiagnostics=1
ENV DOTNET_EnableDiagnostics=1
ENV DOTNET_ROLL_FORWARD=LatestMajor

RUN apt-get update && apt-get install -y unzip curl
ADD https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases/download/v1.9.0/otel-dotnet-auto-install.sh /otel-install.sh

RUN chmod +x /otel-install.sh && \
    /otel-install.sh

USER $APP_UID
WORKDIR /app
EXPOSE 8080

# This stage is used to build the service project
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["WebApplication1/WebApplication1.csproj", "WebApplication1/"]
#COPY ["WebApplication1.csproj", "WebApplication1/"]
RUN dotnet restore "./WebApplication1/WebApplication1.csproj"
COPY . .
WORKDIR "/src/WebApplication1"
RUN dotnet build "./WebApplication1.csproj" -c $BUILD_CONFIGURATION -o /app/build

# This stage is used to publish the service project to be copied to the final stage
```

Sample YAML with the above changes.

```
      app: applicareapp
  template:
    metadata:
      labels:
        app: applicareapp
    spec:
      containers:
        - name: webapplication1
          image: applicare910/applicareapp:webapplicationwithenv
          ports:
            - containerPort: 8080
          env:
            - name: OTEL_DOTNET_AUTO_HOME
              value: "/otel-dotnet-auto"
            - name: DOTNET_SHARED_STORE
              value: "/otel-dotnet-auto/store"
            - name: DOTNET_STARTUP_HOOKS
              value: "/otel-dotnet-auto/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll"
            - name: DOTNET_ADDITIONAL_DEPS
              value: "/otel-dotnet-auto/AdditionalDeps"
            - name: OTEL_TRACES_EXPORTER
              value: "otlp"
            - name: OTEL_EXPORTER_OTLP_ENDPOINT
              value: "http://192.168.0.117:4318"
            - name: OTEL_SERVICE_NAME
              value: "Applicare_demo"
            - name: OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT
              value: "true"
            - name: OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT
              value: "true"
---
apiVersion: v1
kind: Service
metadata:
  name: applicareapp
  namespace: applicare-ns
spec:
  selector:
    app: applicareapp
  ports:
    - protocol: TCP
```

CONFIGURATION GUIDES › .NET

Monitor .NET Application

Applicare 9.2 · Updated September 13, 2024 · dotnet · **Pending technical review**

Currently, Applicare supports profiling only .NET applications hosted on IIS in Windows.

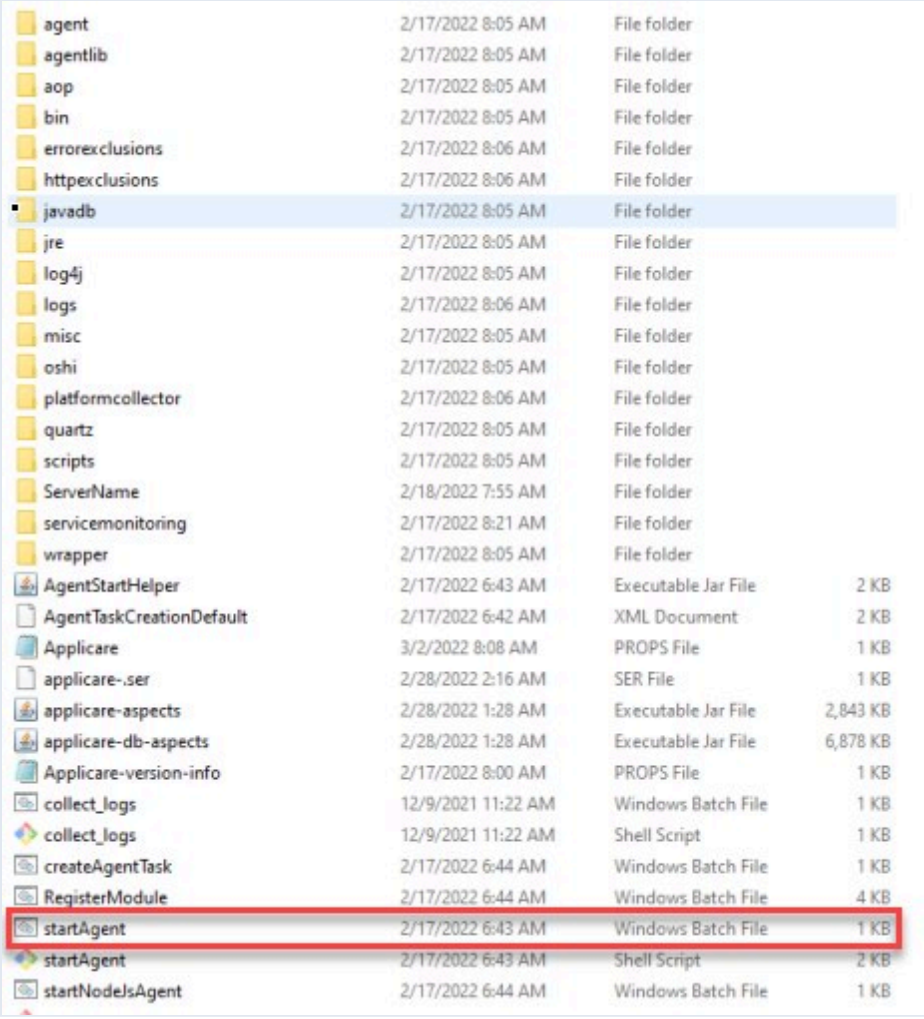
Follow these steps to configure .NET agent with SingleAgent.

Steps 1: Follow the steps for installing Applicare SingleAgent .

Do not start the agent immediately after installation. You will need to start it later.

Steps 2: Navigate to SingleAgent extracted directory.

Steps 3: Edit the **startAgent.bat** file.



Steps 4: In the **startAgent.bat** file, remove the **rem** that precedes the **call RegisterModule.bat** line and save the change.

Before editing the file:

```
@echo off
FOR /F "usebackq" %%i IN (`hostname`) DO SET SERVER_NAME=%%i_sa
rem set SERVER_NAME=Applicare_sa
set ARCTURUS_HOME=%CD%
set JAVA_HOME_PATH=%JAVA_HOME%

set INBUILD_JAVA_PATH=%ARCTURUS_HOME%\jre\windows
if exist "%INBUILD_JAVA_PATH%\bin" (
    set JAVA_HOME_PATH=%INBUILD_JAVA_PATH%
)

rem call RegisterModule.bat
```

After editing the file:

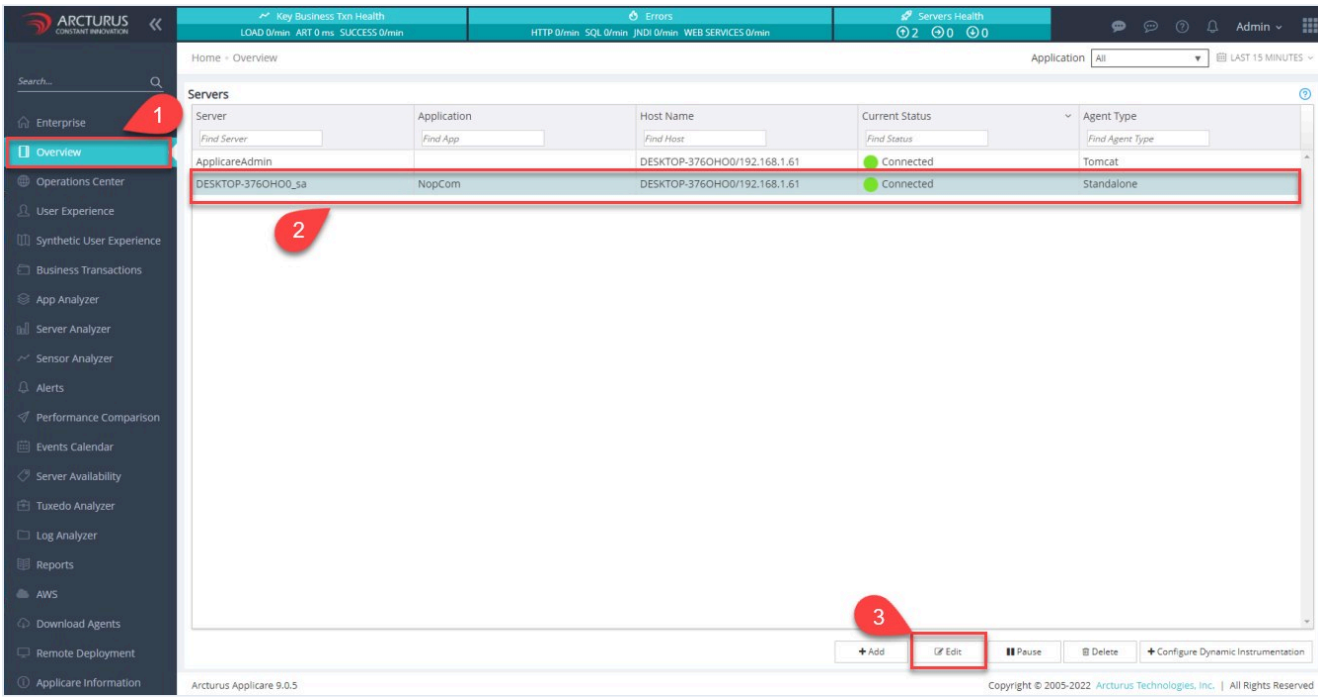
```
@echo off
FOR /F "usebackq" %%i IN (`hostname`) DO SET SERVER_NAME=%%i_sa
rem set SERVER_NAME=Applicare_sa
set ARCTURUS_HOME=%CD%
set JAVA_HOME_PATH=%JAVA_HOME%

set INBUILD_JAVA_PATH=%ARCTURUS_HOME%\jre\windows
if exist "%INBUILD_JAVA_PATH%\bin" (
    set JAVA_HOME_PATH=%INBUILD_JAVA_PATH%
)

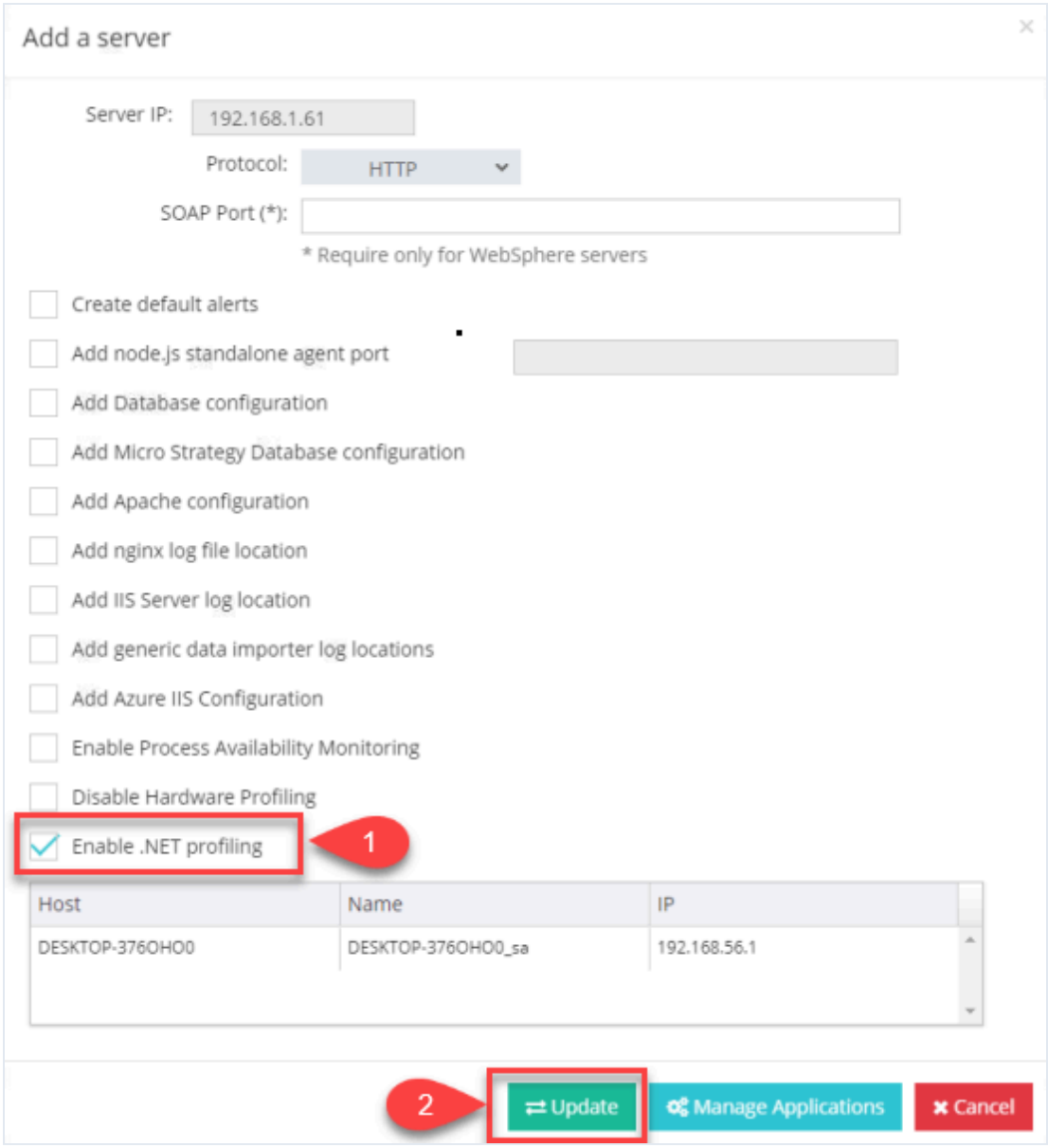
call RegisterModule.bat
```

Steps 5: Start Singleagent.

Steps 6: Log in to the **Applicare console,** navigate to the **Overview** screen, select the agent, click the Edit button.



Steps 7: When the "**Add a server**" popup window appears, check the box for "**Enable .NET profiling**" and click the **Update** button.



Another steps to install Dotnet agent.

Docs Home

Monitoring with Applicare

CONFIGURATION GUIDES › .NET

Manually Installing the .NET Applicare Agent

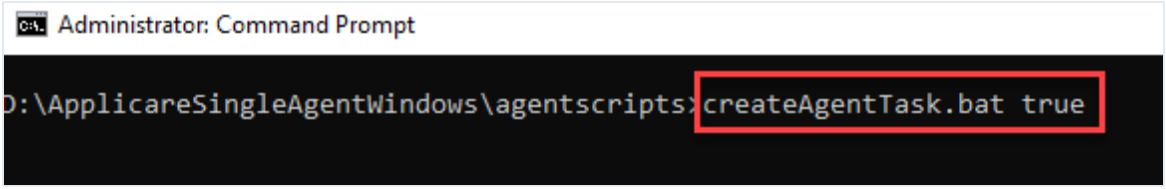
Applicare 9.2 · Updated September 13, 2024 · dotnet, singleagent · **Pending technical review**

1. Log in to **Applicare console,** Navigate to **Download Agents** screen but selecting it on left hand side menu.

2. Click the **SingleAgentWindows** button to download the agent. This will download the **ApplicareSingleAgentWindows.zip** file. Copy the **ApplicareSingleAgentWindows.zip** file to the directory where you intend to deploy the agent e.g. c:\ or d:\.

3. Extract the **ApplicareSingleAgentWindows.zip** file into a directory named **ApplicareSingleAgentWindows**.

4. Open command prompt as an administrator, navigate to the **ApplicareSingleAgentWindows\agentscripts** directory, and run the **createAgentTask.bat** file with **True** parameter.



Monitoring with Applicare

CONFIGURATION GUIDES › .NET

Set up the .NET agent to run automatically using Task Scheduler

Applicare 9.2 · Updated September 13, 2024 · dotnet, singleagent · **Pending technical review**

Step 1: To initiate the .Net Agent with Task Scheduler. Navigate to the **Agent's installation home directory,** open a command window as an **administrator mode** and run the **createDotNetAgentTask.bat** file.

```
C:\ApplicareSingleagent>createAgentTask.bat
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareAgent\SingleAgent" has successfully been created.
Microsoft (R) .NET Global Assembly Cache Utility.  Version 4.0.30319.1
Copyright (c) Microsoft Corporation.  All rights reserved.

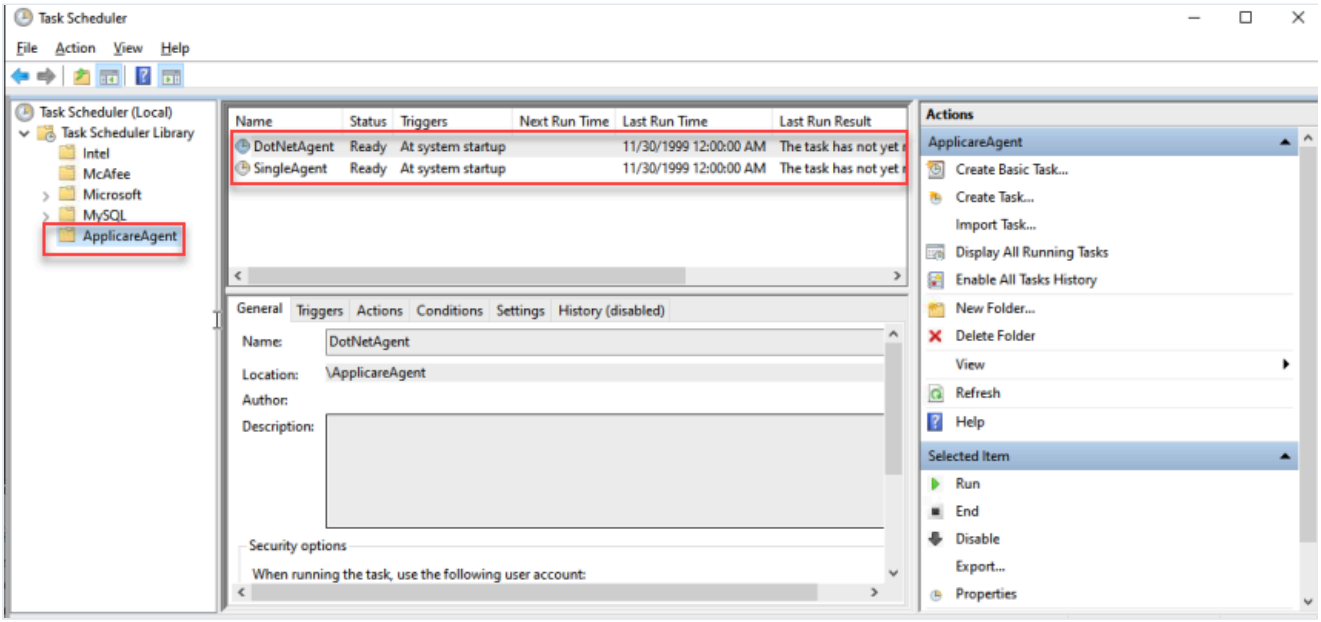
Assembly successfully added to the cache
ERROR ( message:Failed to add duplicate collection element "ApplicareAgentProxy". )

SUCCESS: Specified value was saved.

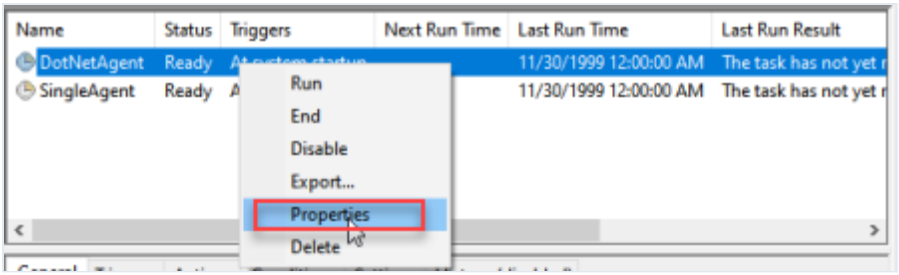
SUCCESS: Specified value was saved.
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareAgent\DotNetAgent" has successfully been created.

C:\ApplicareSingleagent>
```

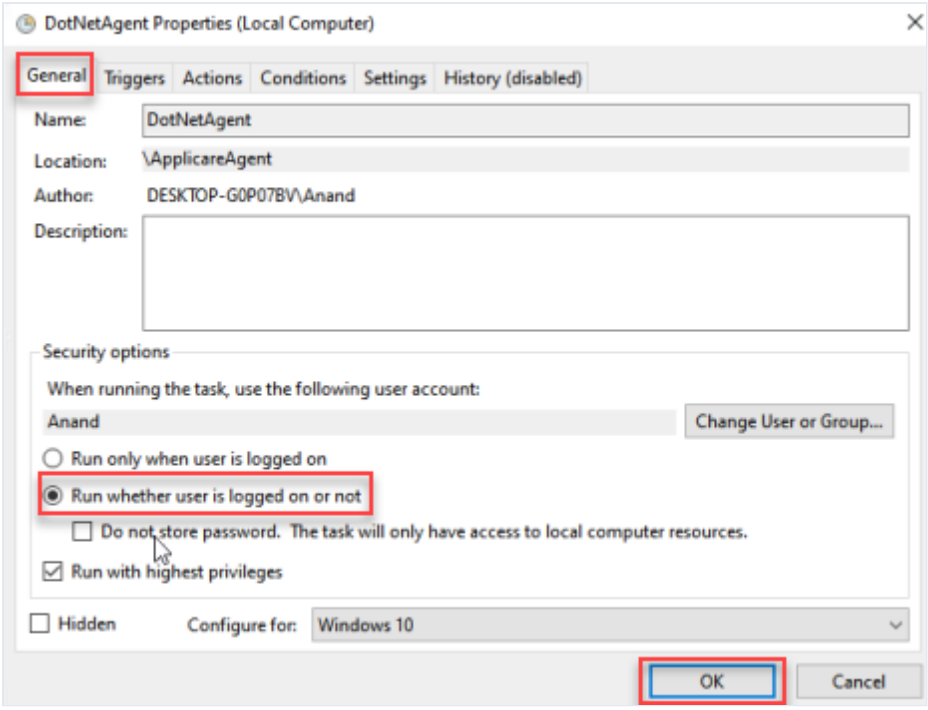
Step 2: Open Task Scheduler, expand the Task Scheduler Library and then select the **ApplicareAgent** Folder.



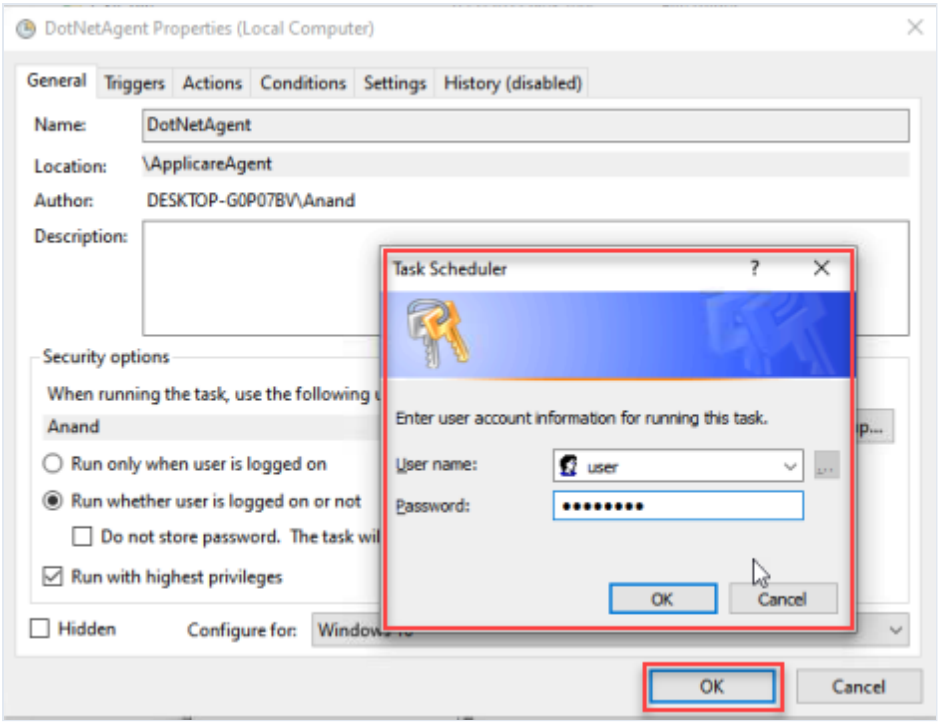
Step 3: Right-Click the **DotnetAgent** task and choose **Properties**.



Step 4: Select the Option to **Run whether user is logged on or not** and then click **OK**.

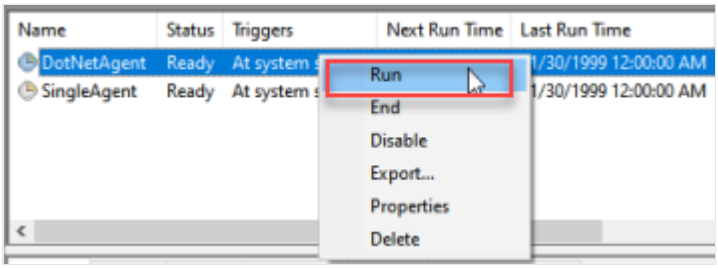


Step 5: Enter the **User name** and **Password** and click **OK**.

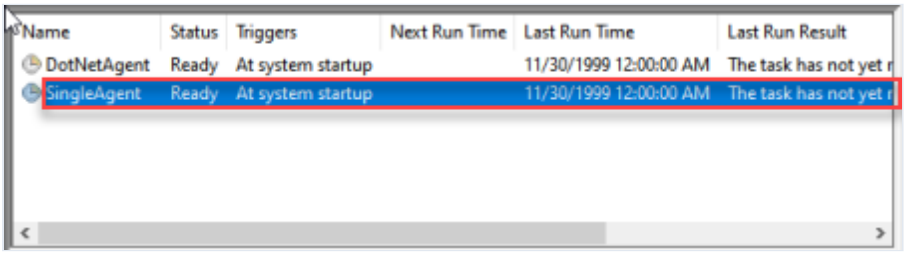


If you encounter error **2147943712** while saving the password. **Refer here**

Step 6: Right-click the **DotnetAgent** task and choose **Run**.



Step 7: Repeat **Steps 3,4,5 and 6** for the SingleAgent task in Task Scheduler.



Monitoring with Applicare

CONFIGURATION GUIDES › .NET

Monitor .NET applications running on Windows using Opentelemetry

Applicare 9.2 · Updated May 5, 2026 · dotnet, opentelemetry, windows · **Pending technical review**

To begin monitoring, Install the SingleAgent.

Once the **SingleAgent** installation is successfully completed, follow the subsequent steps below:

1. Open powershell in adminstrator mode and execute the following commands one by one.

```powershell
$module_url = "https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases/latest/download/OpenTelemetry.DotNet.Auto.psm1"
$download_path = Join-Path $env:temp "OpenTelemetry.DotNet.Auto.psm1"
Invoke-WebRequest -Uri $module_url -OutFile $download_path -UseBasicParsing
Import-Module $download_path
Install-OpenTelemetryCore
```

Additional commands.

```powershell
# List all available commands
Get-Command -Module OpenTelemetry.DotNet.Auto
```

`To monitor Dotnet Applications hosted in IIS, execute the below command`

```text
Register-OpenTelemetryForIIS
```

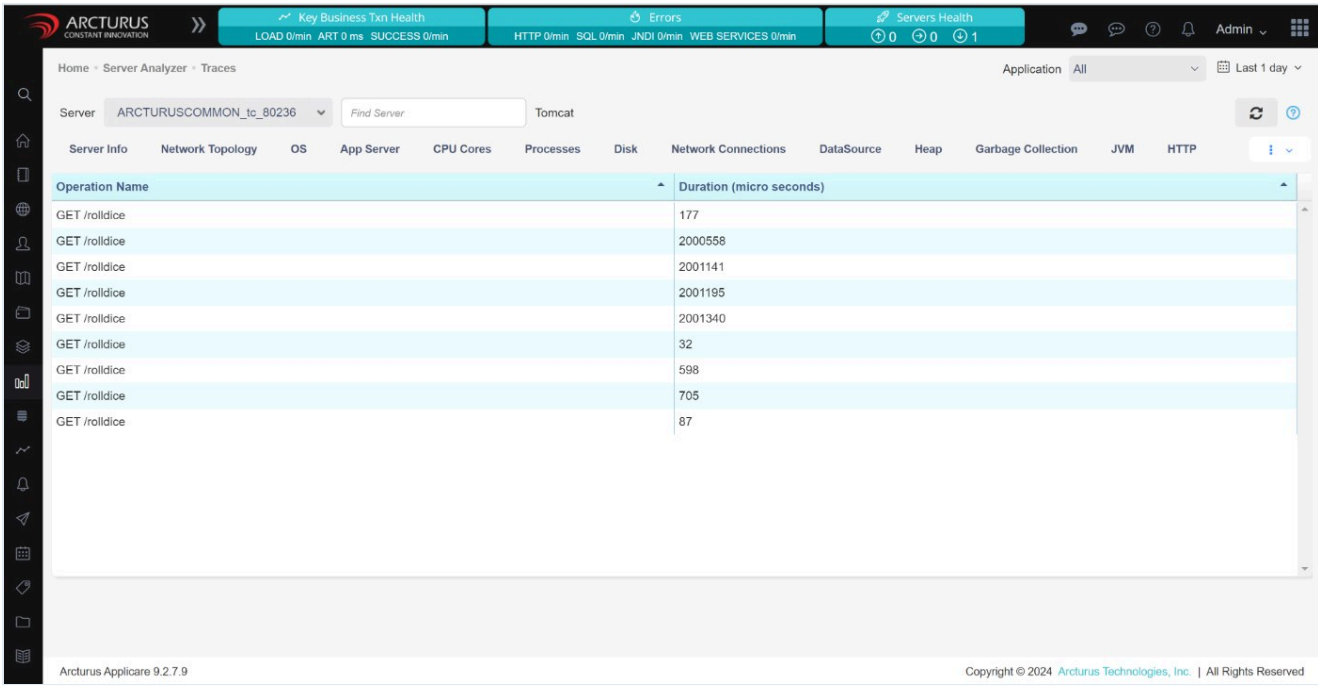
`Add below environment variables to start monitor both .NET Framework and .NET Core applications.`

```
TEXT
COR_ENABLE_PROFILING=1
COR_PROFILER={918728DD-259F-4A6A-AC2B-B85E1B658318}
COR_PROFILER_PATH=C:\Program Files\OpenTelemetry .NET AutoInstrumentation\win-
x64\OpenTelemetry.AutoInstrumentation.Native.dll
CORECLR_ENABLE_PROFILING=1
CORECLR_PROFILER={918728DD-259F-4A6A-AC2B-B85E1B658318}
CORECLR_PROFILER_PATH=C:\Program Files\OpenTelemetry .NET AutoInstrumentation\win-
x64\OpenTelemetry.AutoInstrumentation.Native.dll
DOTNET_ADDITIONAL_DEPS=C:\Program Files\OpenTelemetry .NET AutoInstrumentation\AdditionalDeps
DOTNET_SHARED_STORE=C:\Program Files\OpenTelemetry .NET AutoInstrumentation\store
DOTNET_STARTUP_HOOKS=C:\Program Files\OpenTelemetry .NET
AutoInstrumentation\net\OpenTelemetry.AutoInstrumentation.StartupHook.dll
OTEL_DOTNET_AUTO_HOME=C:\Program Files\OpenTelemetry .NET AutoInstrumentation
OTEL_DOTNET_AUTO_INSTALL_DIR=C:\Program Files\OpenTelemetry .NET AutoInstrumentation
OTEL_DOTNET_AUTO_LOGS_CONSOLE_EXPORTER_ENABLED=true
OTEL_DOTNET_AUTO_METRICS_CONSOLE_EXPORTER_ENABLED=true
OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED=true
OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT=true
OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT=true
OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4418
OTEL_SERVICE_NAME=<APPLICARE_AGENT_NAME>
```

Note: `<APPLICARE_AGENT_NAME>`: Provide a unique, descriptive name that will help you identify this specific container or application within the Applicare monitoring console.

`After the above settings restart the .NET application` or IIS.

The collected traces will be shown in Server Analyzer -> Telemetry Traces screen in Applicare console.



CONFIGURATION GUIDES › JAVA

Applicare Java Requirements

Applicare 9.2 · Updated April 14, 2026 · java · **Pending technical review**

Applicare Controller

Windows

https://download.oracle.com/java/25/archive/jdk-25.0.1_windows-x64_bin.zip

Linux

https://arc-temp.s3.amazonaws.com/java-1.8.0-openjdk-1.8.0.275.b01-1.el8_3.x86_64.zip

Applicare SingleAgent

Windows https://github.com/ojdkbuild/ojdkbuild/releases/download/java-1.8.0-openjdk-1.8.0.252-2.b09/java-1.8.0-openjdk-jre-1.8.0.252-2.b09.ojdkbuild.windows.x86_64.zip

Linux https://github.com/AdoptOpenJDK/openjdk8-binaries/releases/download/jdk8u265-b01/OpenJDK8U-jre_x64_linux_hotspot_8u265b01.tar.gz

CONFIGURATION GUIDES › JAVA

Install SingleAgent in the Java Application

Applicare 9.2 · Updated August 22, 2024 · java, singleagent · **Pending technical review**

Step 1: Install Applicare SingleAgent on the machine where **Java application** is running. Follow the installation instructions for **Installing Applicare SingleAgent**

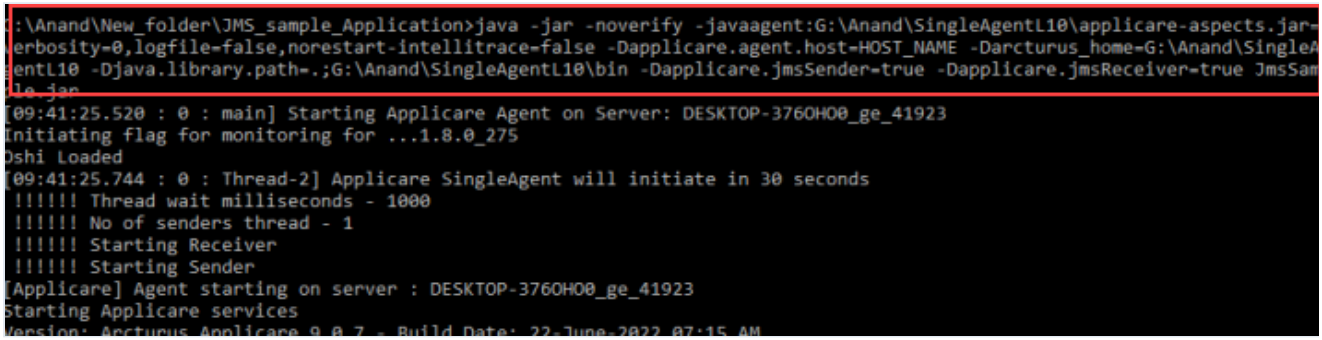
Step 2: Run the JAR file directly with Applicare SingleAgent.

Step 3: Assuming **javaapp.jar** is the executable JAR application that needs to be monitored, Use the following command to run the JAR.

To deploy Applicare SingleAgent, run the **Java Application** with the following arguments:

- -noverify
- -javaagent:**<ApplicareAgentDir>**\applicare-aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false
- -Darcturus_home=**<ApplicareAgentDir>**
- -Dapplicare.agent.host=HOST_NAME
- -Djava.library.path=.;**<ApplicareAgentDir>**\bin
- -Dlogback.configurationFile=**<ApplicareAgentDir>**\logback\logback.xml

e.g. java -jar -noverify -javaagent:**G:\ApplicareSingleAgent**\applicare-aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false -Dapplicare.agent.host=HOST_NAME -Darcturus_home=**G:\ApplicareSingleAgent** -Djava.library.path=.;**G:\ApplicareSingleAgent**\bin -Dlogback.configurationFile=**G:\ApplicareSingleAgent**\logback\logback.xml **javaapp.jar**



Few JVM Arguments

a) To enable IntelliTrace add **-DenableIntellitraceDaemon=true** to startup arguments.

If Java Application is running as a service follow the steps in the below link.

Java Application running as a service

Another steps to install Applicare SingleAgent with Wrapper.

Docs Home

CONFIGURATION GUIDES › JAVA

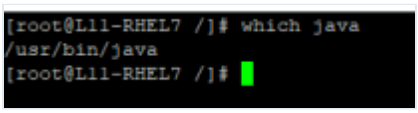
Finding the Java Installation Directory

Applicare 9.2 · Updated September 13, 2024 · java · **Pending technical review**

Follow these steps to find Java Installation path.

Steps 1: Execute below command

```text
which java
```



Steps 2: Run the **ll** command with the directory path obtained from the previous command.

e.g. **ll /usr/bin/java**



Steps 3: Run the **ll** command with the directory path obtained from the previous command. This will reveal the path to the Java installation directory.

e.g. **ll /etc/alternatives/java**



Monitoring with Applicare

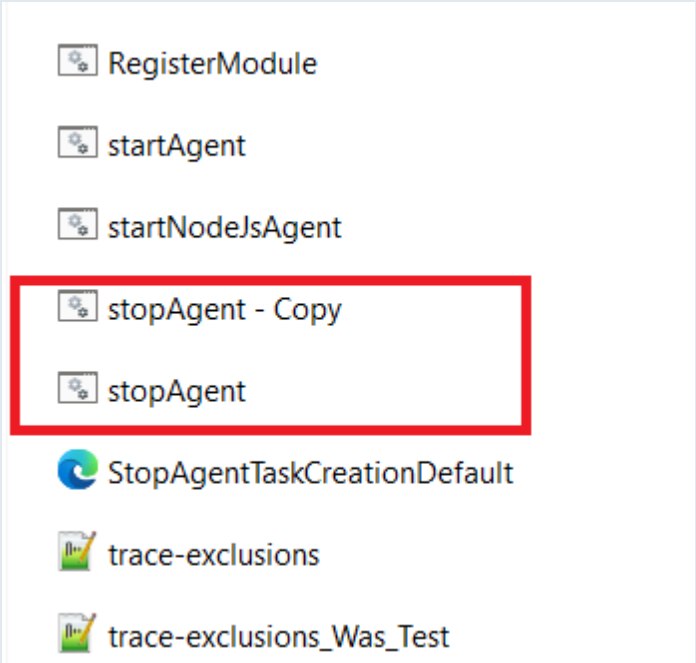
CONFIGURATION GUIDES › JAVA

Restarting the SingleAgent

Applicare 9.2 · Updated September 13, 2024 · singleagent, java · **Pending technical review**

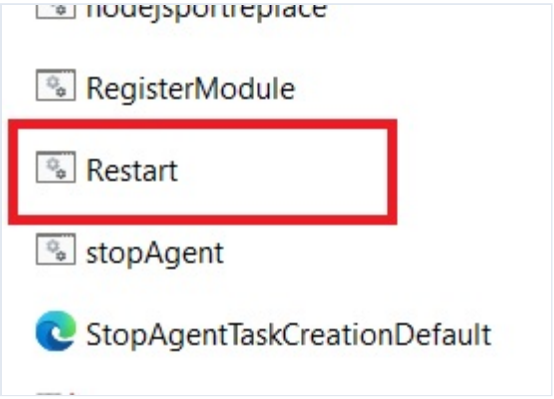
1. Navigate to the ApplicareSingleAgent home directory.

2. Copy and Paste the **StopAgent.bat** file within the same directory. This will create a file **stopAgent - Copy.bat**.



3. Rename the **stopAgent - Copy.bat** file to **Restart.bat** Open the file, replace its existing content with the provided content, **save** the changes.

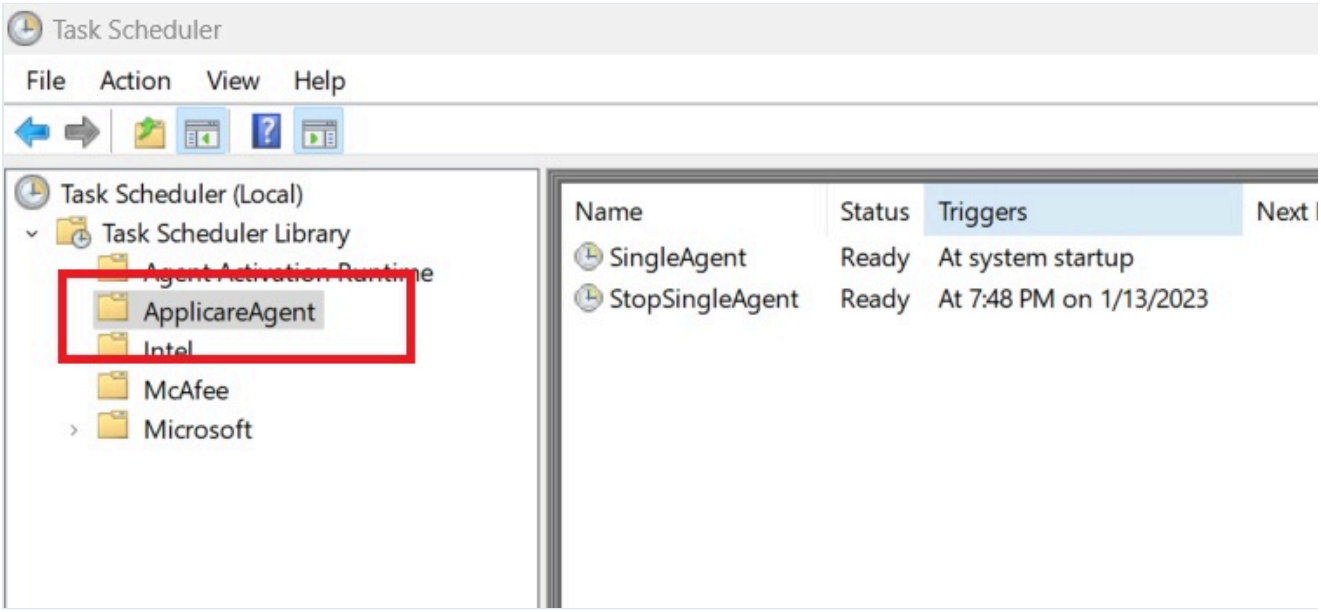
```bat
@echo off
schtasks /run /tn "\ApplicareAgent\StopSingleAgent"
timeout /t 10 /nobreak
schtasks /run /tn "\ApplicareAgent\SingleAgent"
```



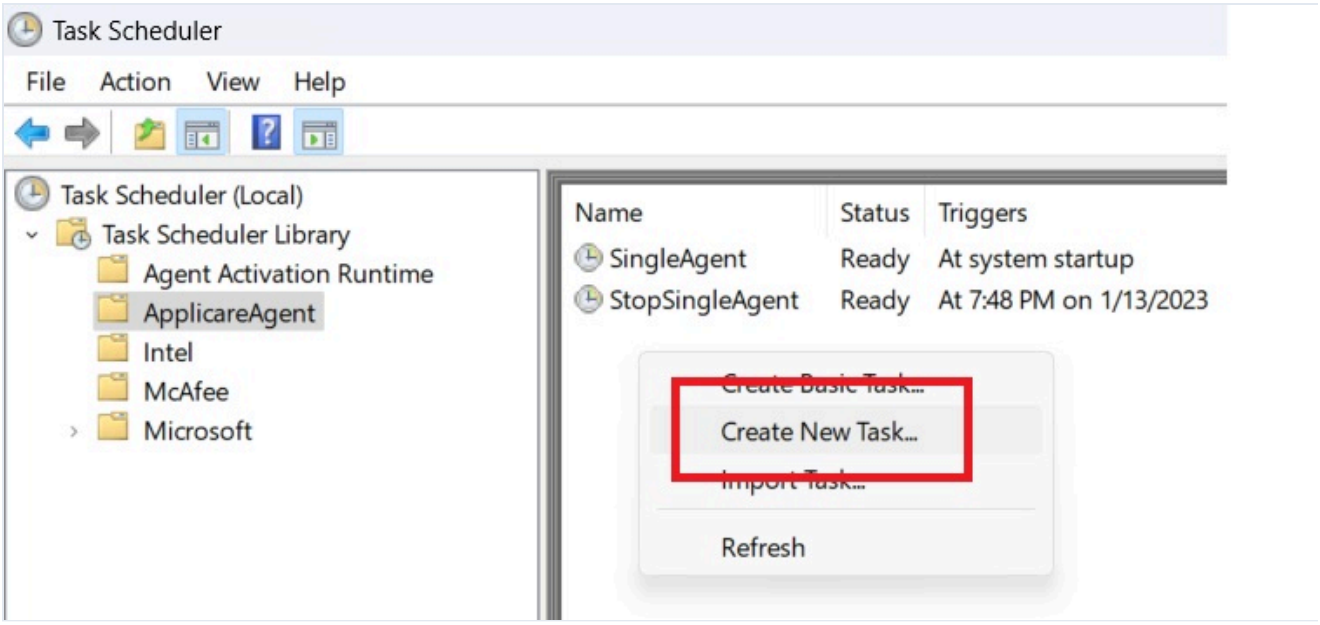
Restart.bat

```
1  @echo off
2  schtasks /run /tn "\ApplicareAgent\StopSingleAgent"
3  timeout /t 10 /nobreak
4  schtasks /run /tn "\ApplicareAgent\SingleAgent"
```

4. Open Task Scheduler in Administrator mode and choose the **ApplicareAgent** directory.



5. Right-click inside the Task Scheduler and select the **Create New Task** option.



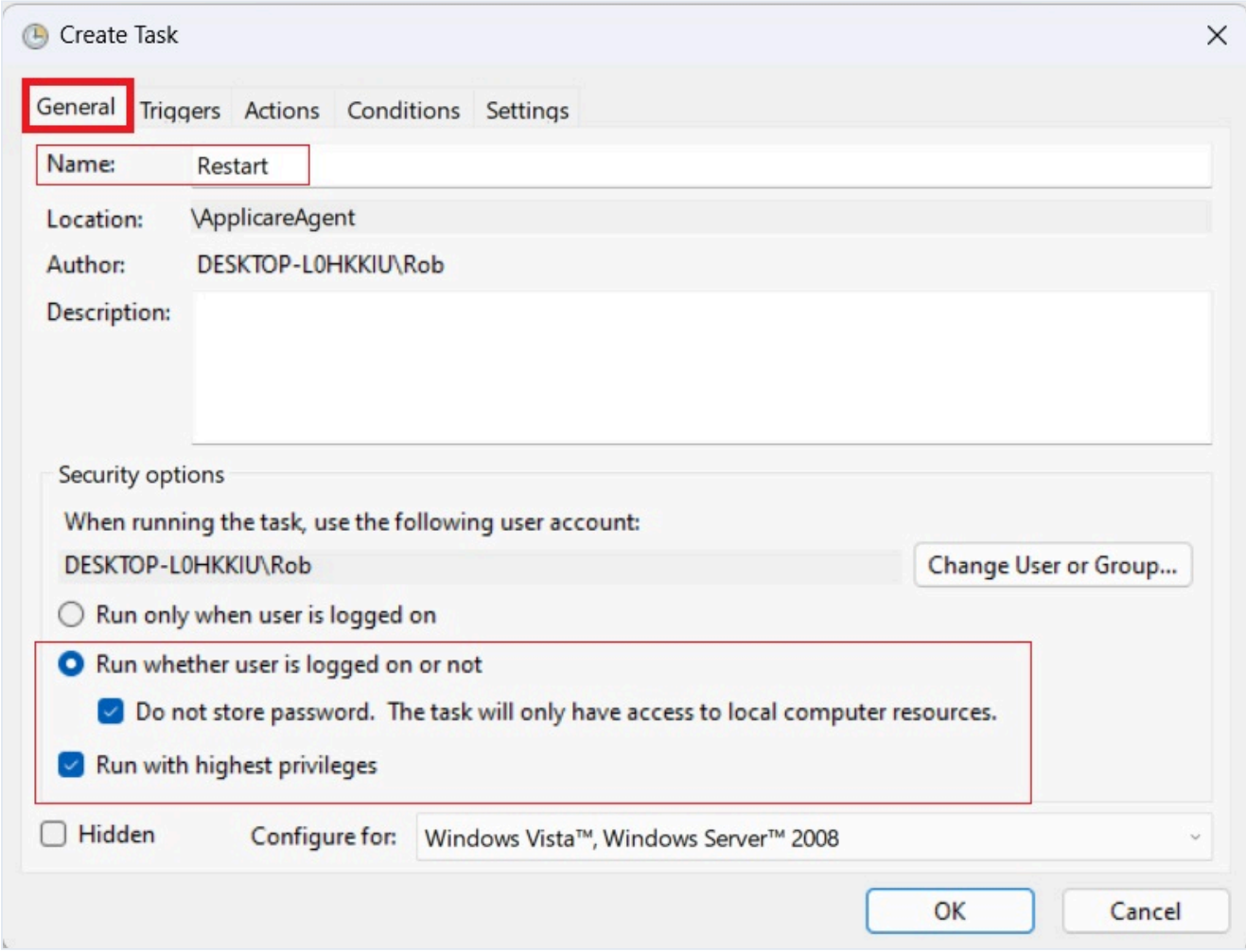
6. In the General tab:

a. Enter the name **Restart**.

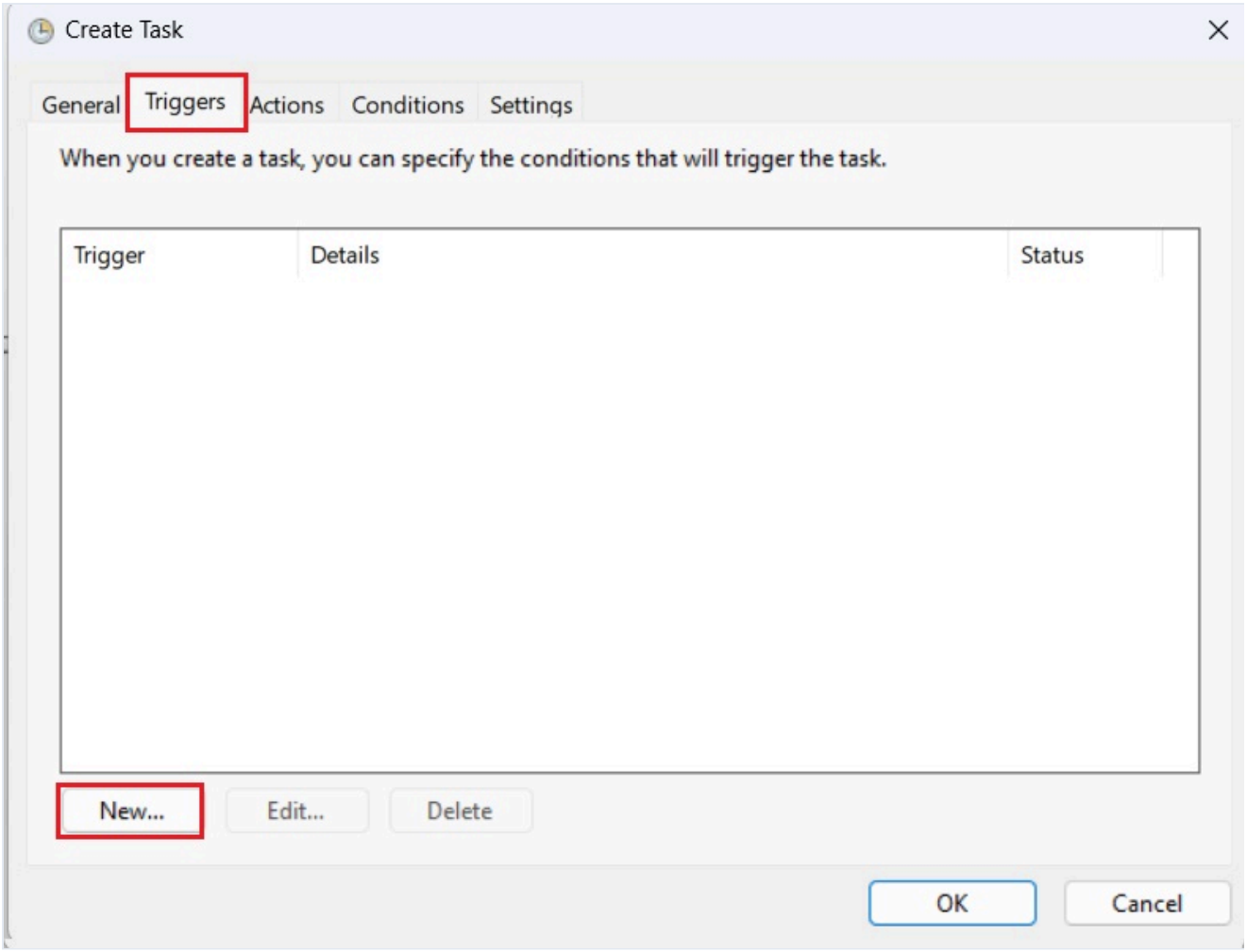
b. Choose **Run whether user is logged on or not**.

c. Select **Do not Store Password**.

d. Select **Run with highest privileges**.



7. Go to the **Triggers Tab** and click the **New** button.



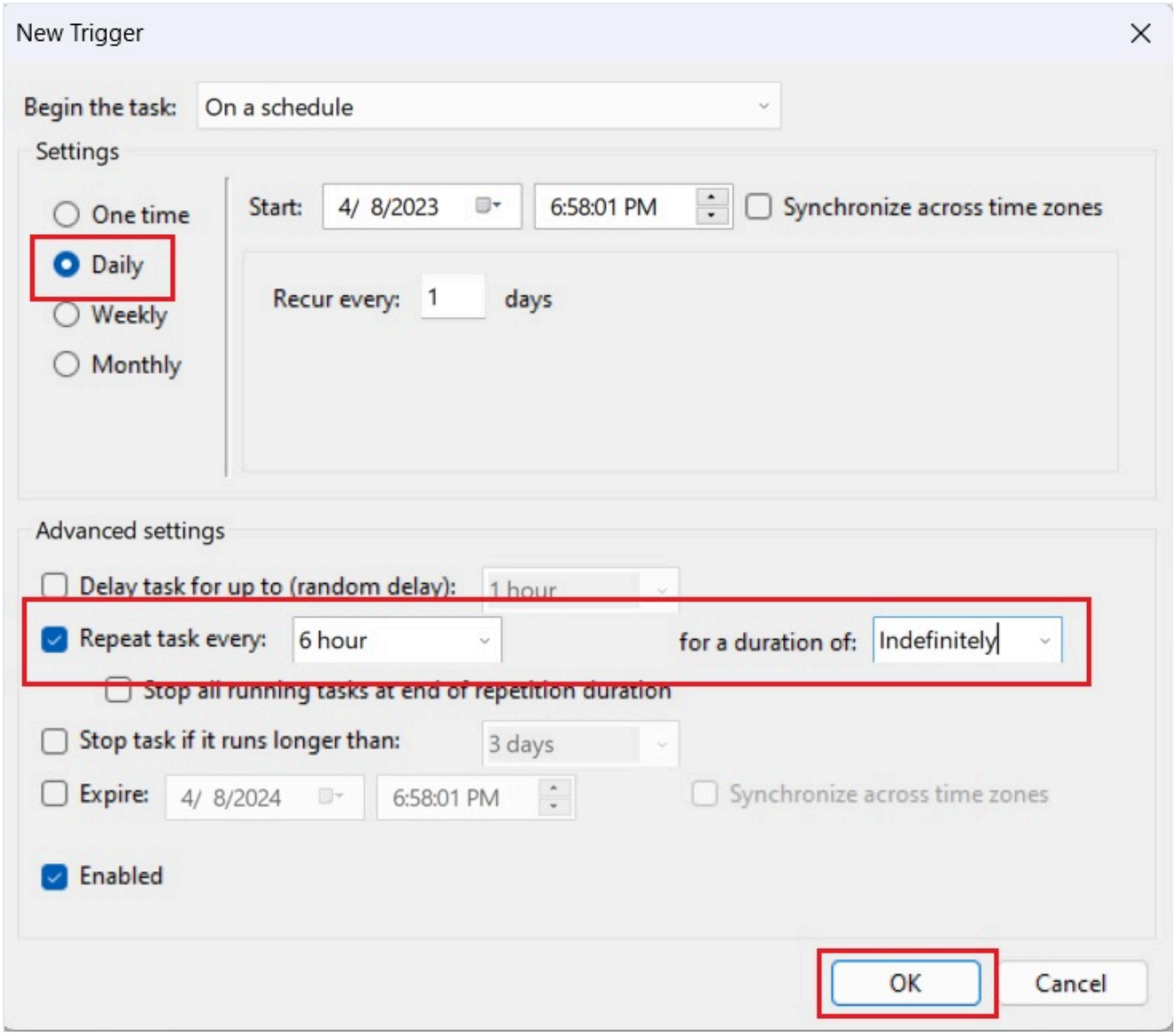
8. Set the below options.

a. Choose **Daily**.

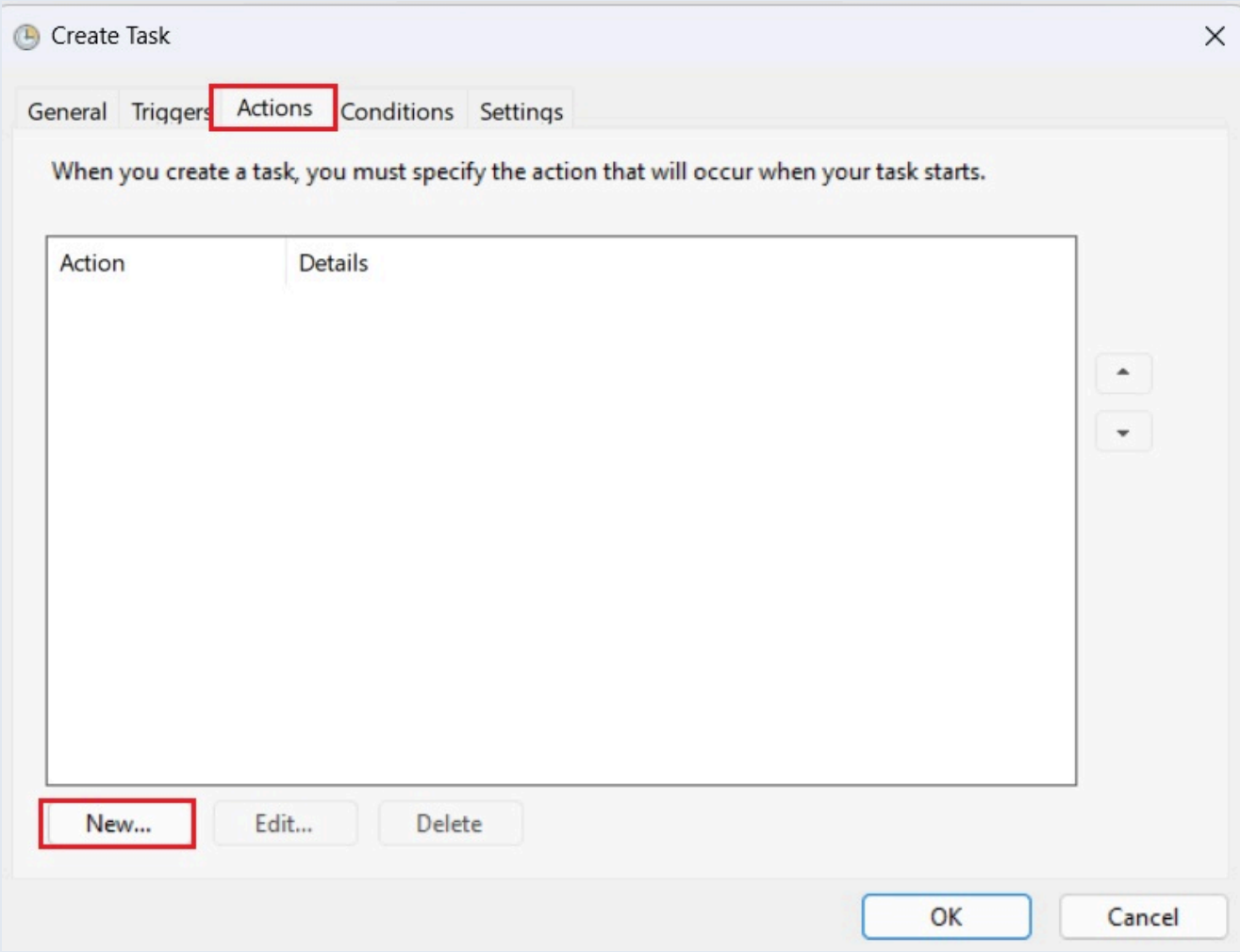
b. Select the **Repeat Task Every** option.

c. Set it to **6 hours** manually (this option may not be available in the dropdown).

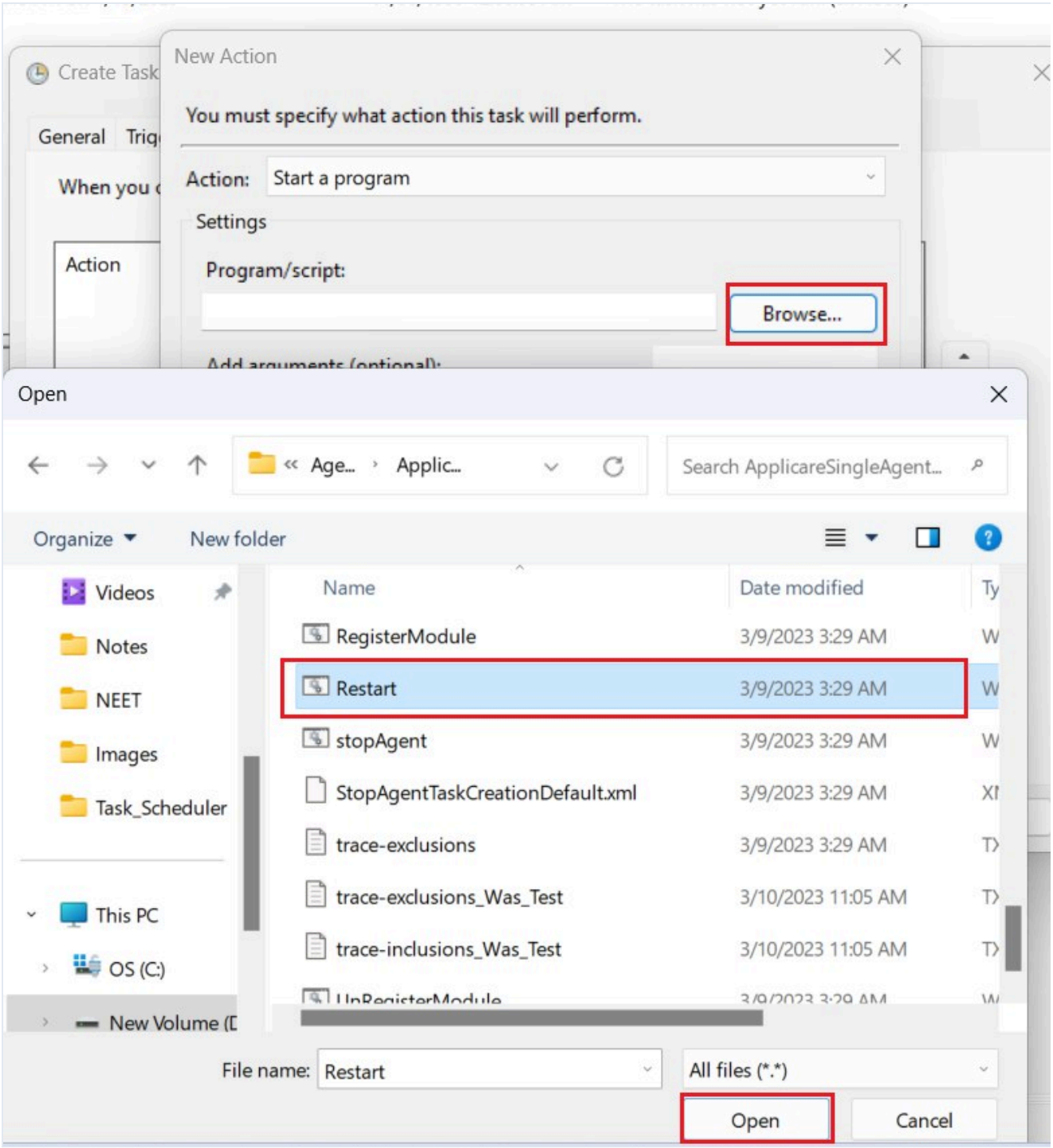
d. Set the duration to **Indefinitely** and click **OK**.



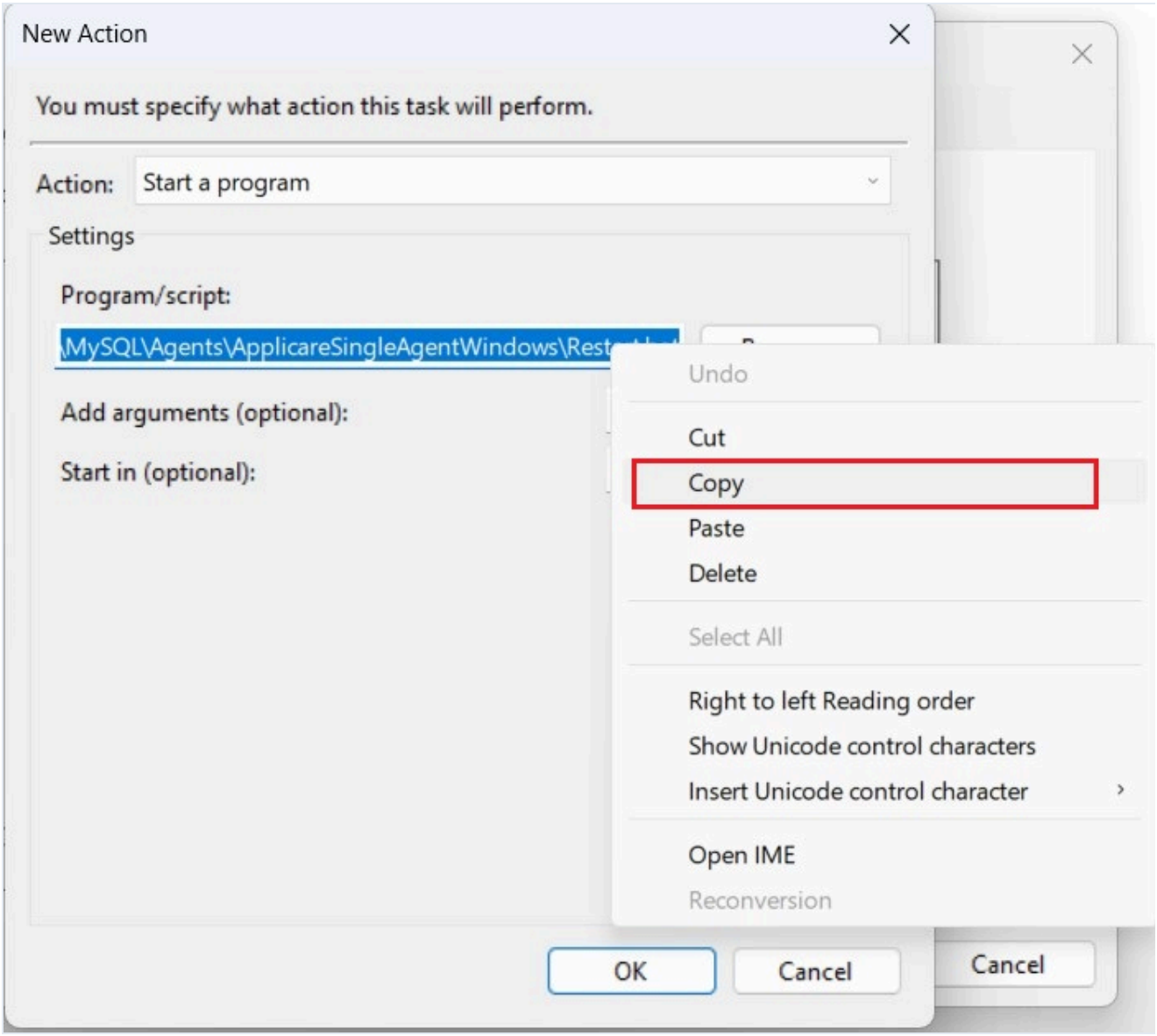
9. Go to the **Actions Tab** and click the **New** button.



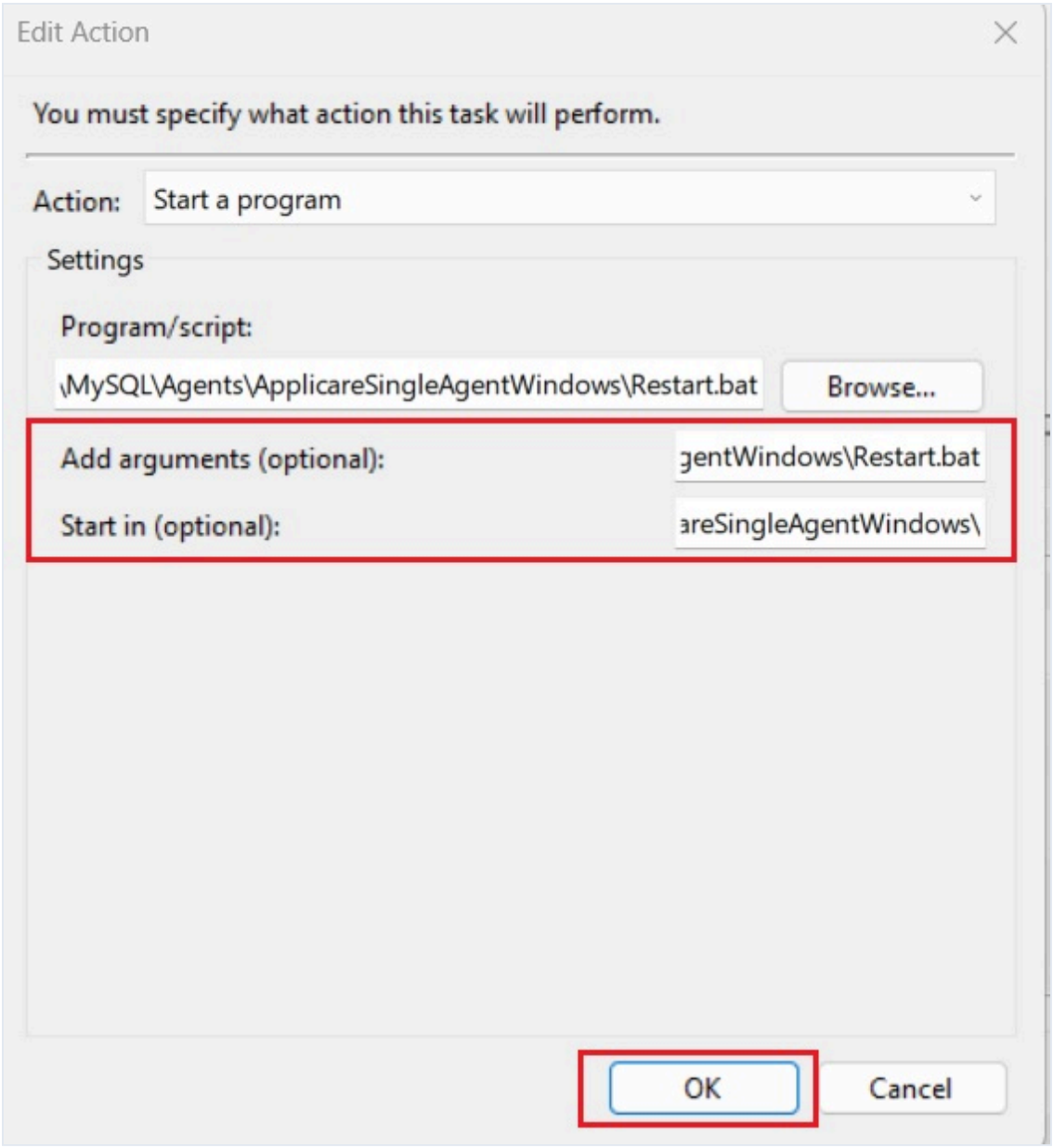
10. Click the **Browse** button, select the **Restart.bat** file, and then click the **Open**.



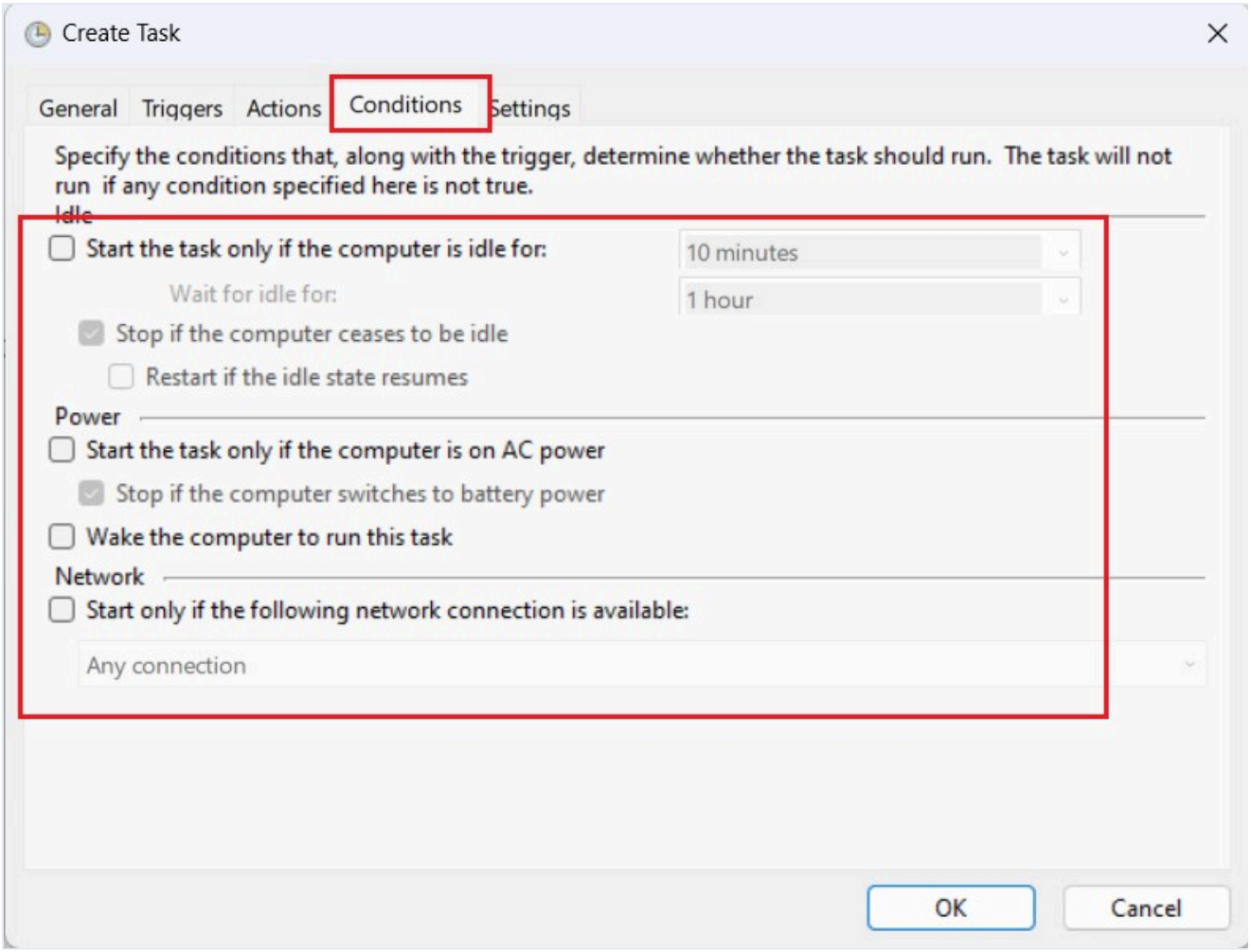
11. Copy the full **Program/script path and paste** it into both the **Add Arguments (optional)** and **start in (optional)** fields.



12. In the **Start in (optional)** field, remove the **Restart.bat** keyword, and then **OK**.



13. Go to the **Conditions Tab** and **uncheck** all the **checkboxes**.

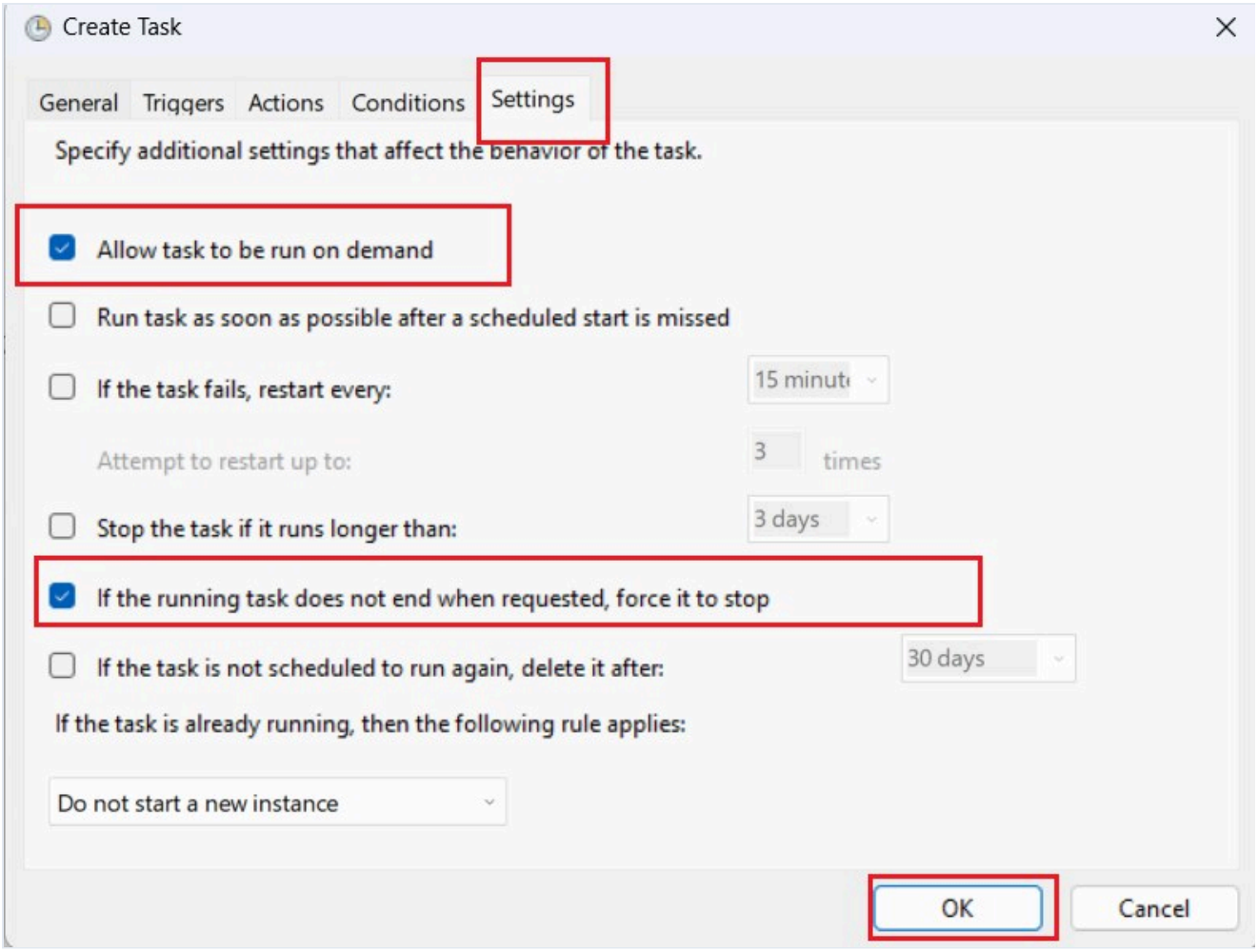


14. Go to the **Settings Tab** and check only following **two options:**

a. Allow task to be run on demand.

b. If the running task does not end when requested, force it to stop.

c. Click OK.



Monitoring with Applicare

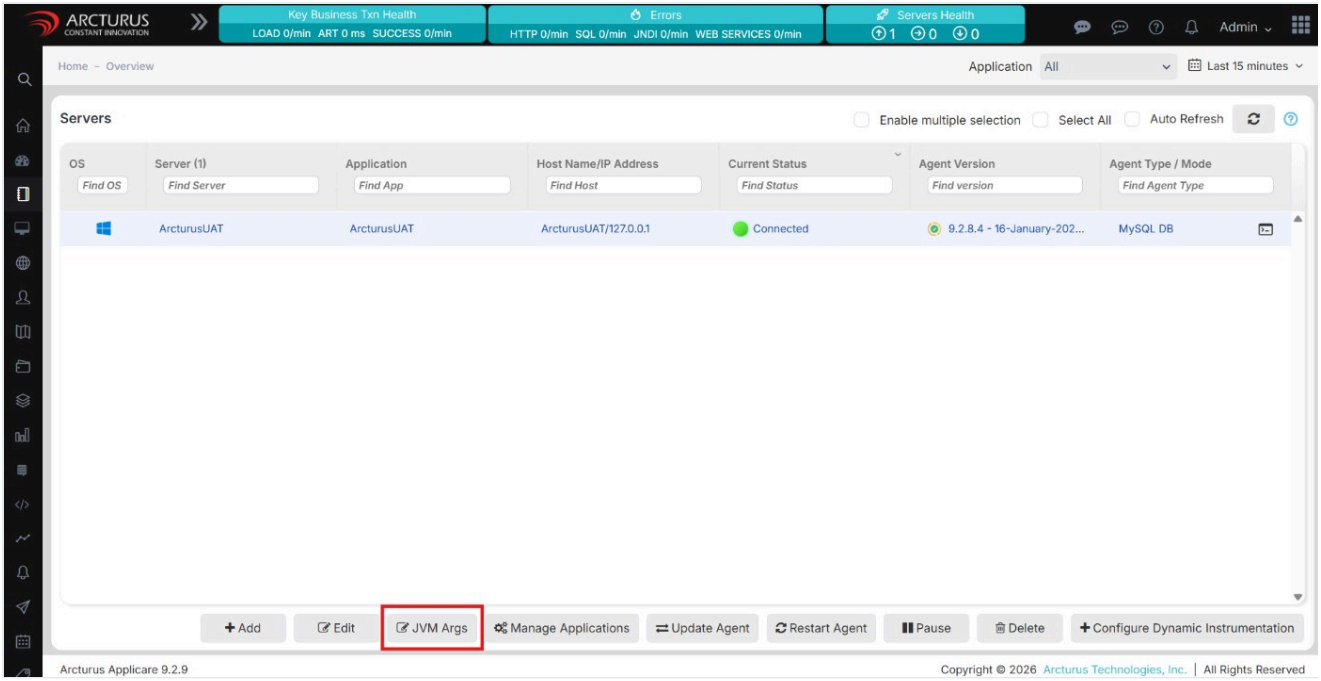
CONFIGURATION GUIDES › JAVA

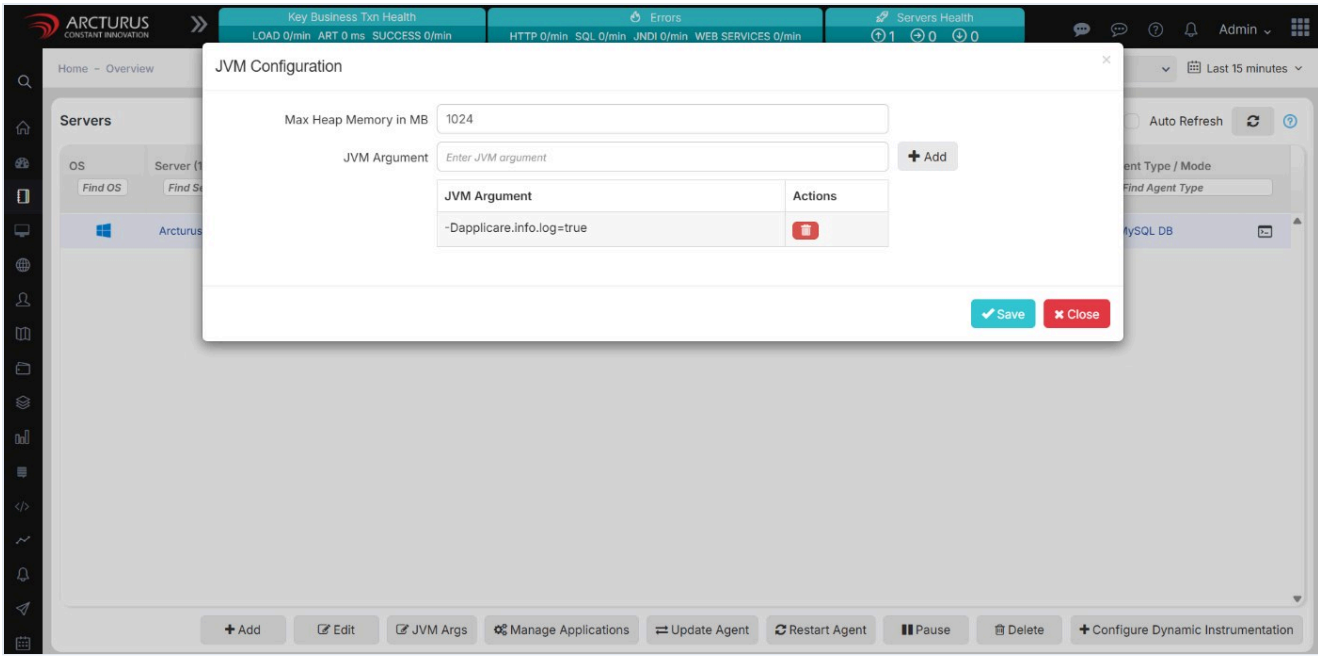
Customize Agent heap memory and jvm arguments

Applicare 9.2 · Updated January 16, 2026 · java, singleagent · **Pending technical review**

Managing JVM configurations manually is time-consuming, requiring remote server access and direct editing of configuration files. Applicare simplifies this by providing a centralized UI to manage these settings. This eliminates the need for manual file intervention and ensures consistency across your environment.

1. Navigate to the **Overview** screen and select your server.
2. Click the **JVM Args** button to open the configuration popup.
3. Add, remove, or edit JVM arguments and heap memory as needed.
4. **Save** your changes.
5. **Wait two minutes** before restarting the Agent to allow the new configuration to sync.




ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
Admin
Home - Overview
Last 15 minutes
Servers
Auto Refresh
OS
Find OS
Arcturus
MySQL DB
JVM Configuration
Max Heap Memory in MB
1024
JVM Argument
Enter JVM argument
Add
JVM Argument
Actions
-Dapplicare.info.log=true
Save
Close
Add
Edit
JVM Args
Manage Applications
Update Agent
Restart Agent
Pause
Delete
Configure Dynamic Instrumentation

CONFIGURATION GUIDES › JAVA

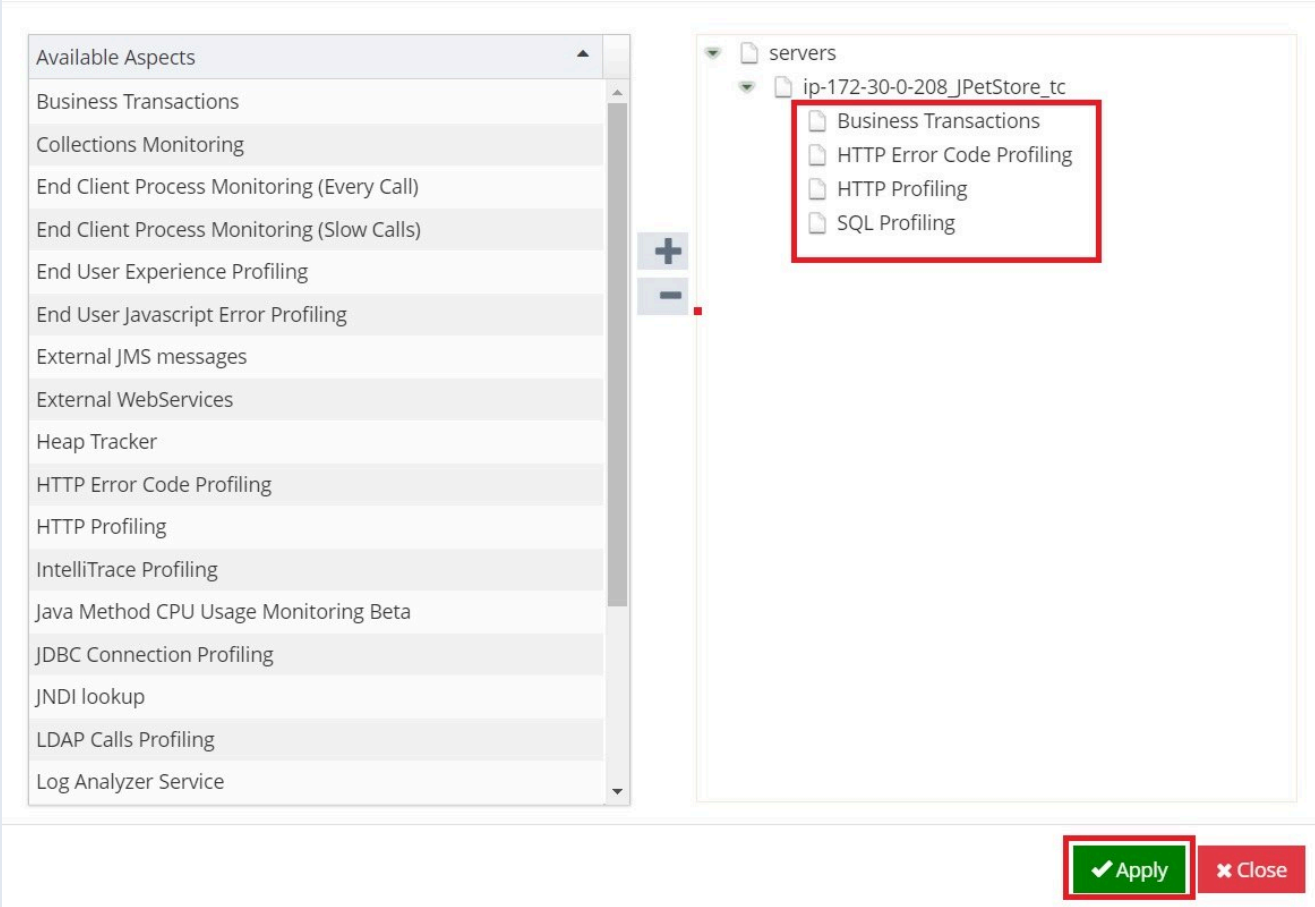
Java Application Server Configuration Settings

Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

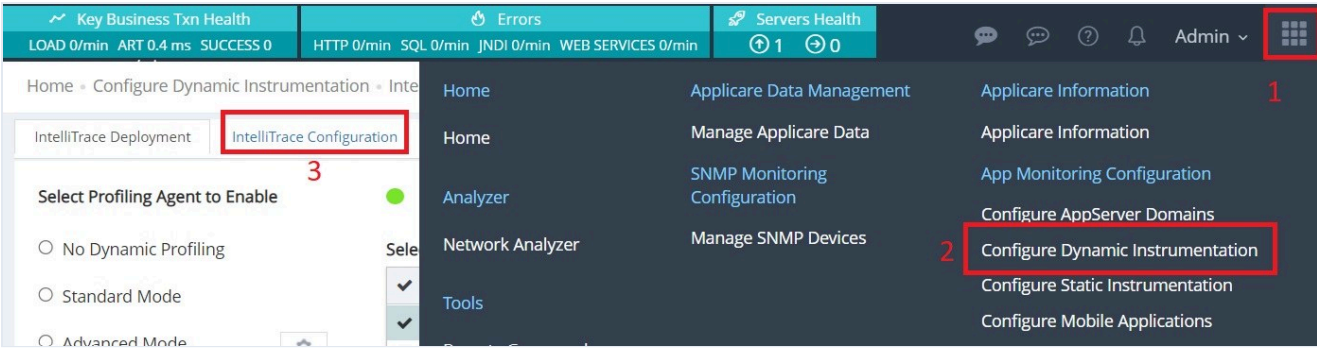
1. Navigate to the **Overview** menu.

2. Choose the **server** from the grid and then click the **Configure Dynamic Instrumentation** button.

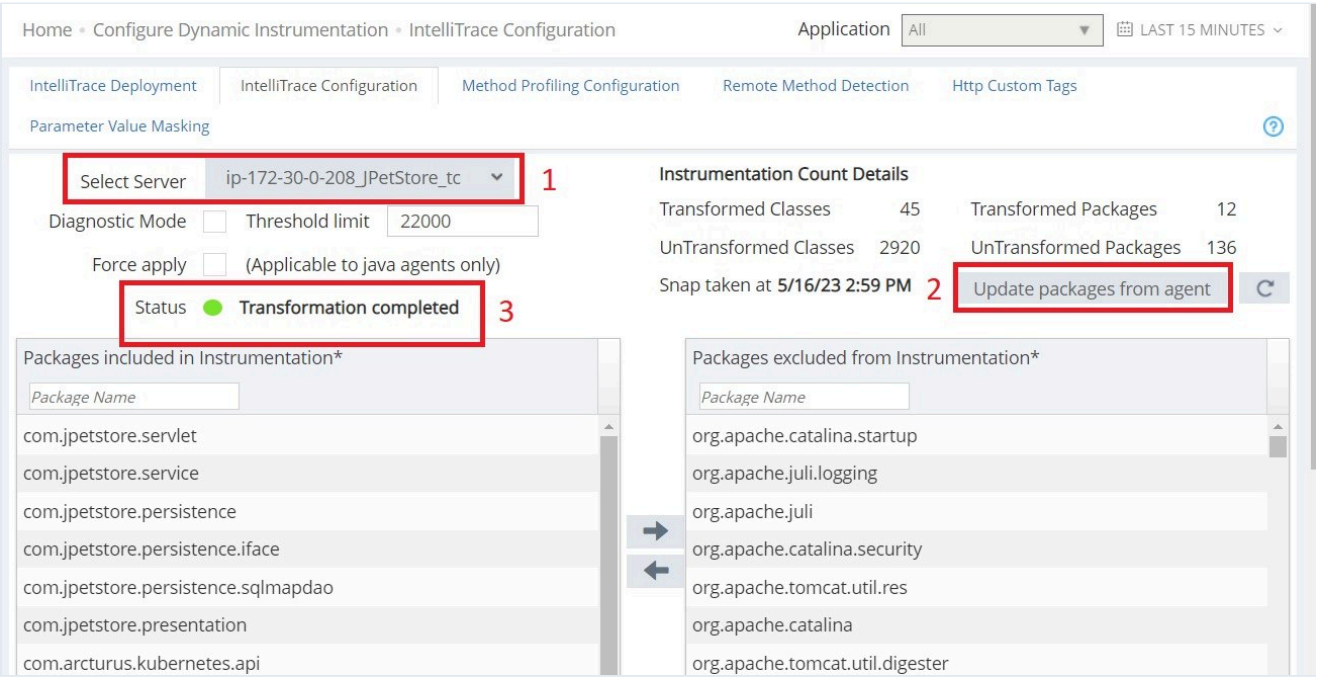
3. On the right hand-side, select only the **Business Transactions, HTTP Error Code Profiling, HTTP Profiling** and **SQL Profiling** aspects and then click the **Apply** button.



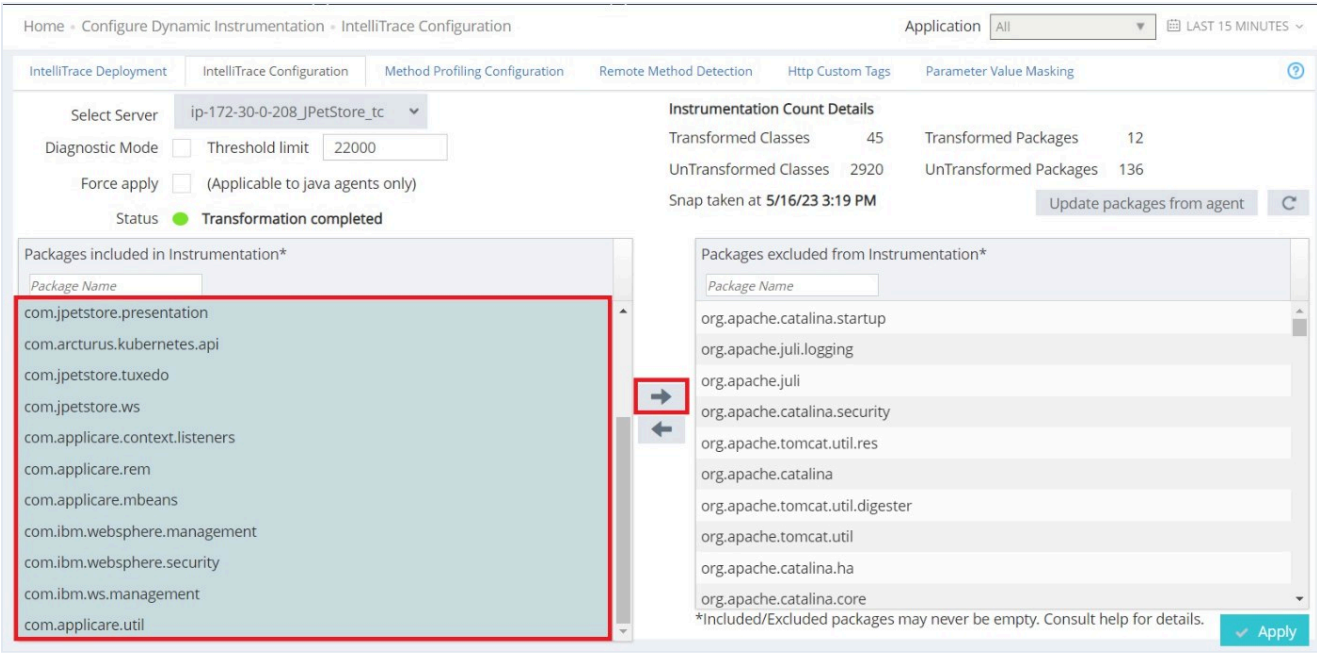
4. In the right-hand side menu, go to **Configure Dynamic Instrumentation** and select the **IntelliTrace Configuration** tab.



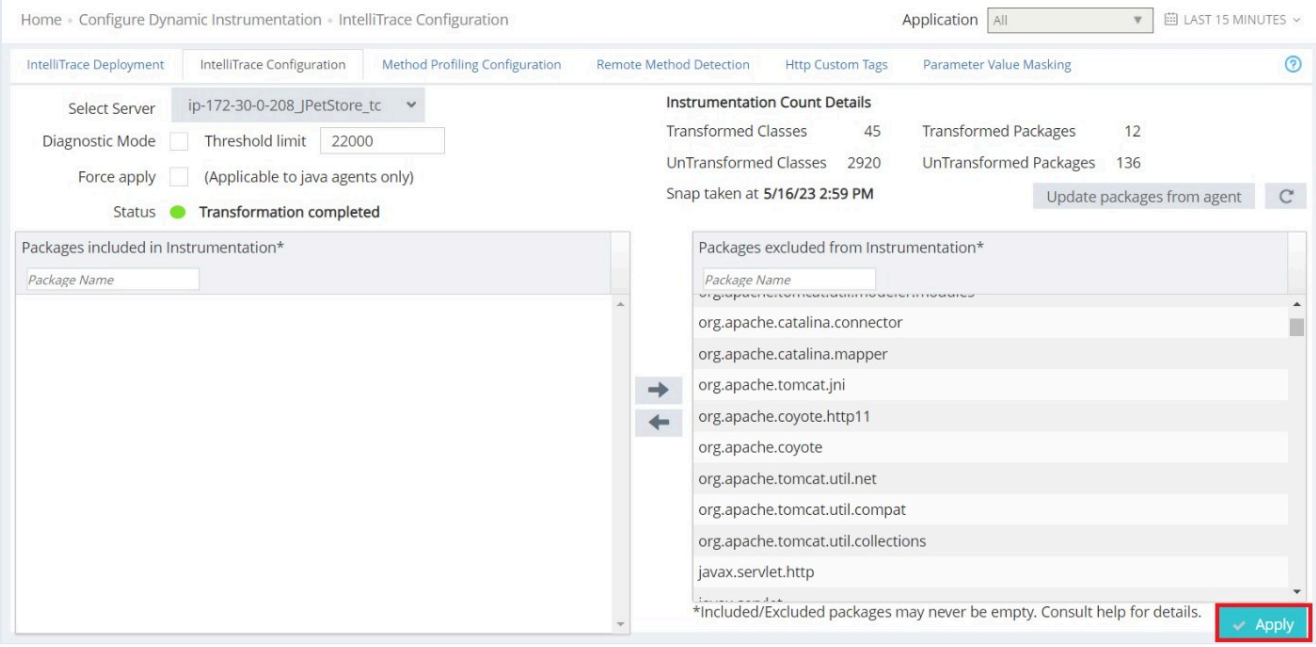
5. Select the **Server** from the server dropdown menu and click the **Update Packages from Agent** button. Then, wait for 2 minutes. Once the transformation is complete, the status field will display **Transformation completed** message. If the message "Transformation in progress" is displayed, please check again after 10 minutes.



6. After the **Transformation completed** message appears, select all the packages on the left hand side and move them to the right hand side.



7. Click the **Apply** button to proceed.



8. **Restart the server** after waiting for two minutes.

Monitoring with Applicare

CONFIGURATION GUIDES › JAVA

Include the server name and JVM arguments in the Gradle Jib build configuration.

Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

1. Open the **Gradle build file** and append the container block at the end of the **Jib** configuration section.

Syntax:

```
TEXT
container {
jvmFlags = ['ARG1=Value','ARG2=Value']
}
```

e.g.

container { jvmFlags = ['-Dserver.name=tomcat10','-Dhost.name=tomcat'] }



Monitoring with Applicare

CONFIGURATION GUIDES › JAVA

Install Applicare agent in Java app

Applicare 9.2 · Updated August 20, 2024 · java, singleagent · **Pending technical review**

a.) Add the following to your service's JVM options and replace the bold values with your specific settings if you're starting your server as a service..

- -javaagent:**<ApplicareAgentHomeDir>**\applicare-aspects.jar=verbosity=0,logfile=false
- -Darcturus_home=**<ApplicareAgentHomeDir>**
- -Djava.library.path=**<ApplicareAgentHomeDir>**\bin
- -Xverify:none

b.) Ensure the JVM is configured to allocate at least 256MB for PermGen by adding **-XX:MaxPermSize=256m** to the startup arguments. The server may fail to start with the Applicare agent if the PermGen size is insufficient.

c.) Start the server.

CONFIGURATION GUIDES › JAVA

Installing Applicare SingleAgent using Wrapper

Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

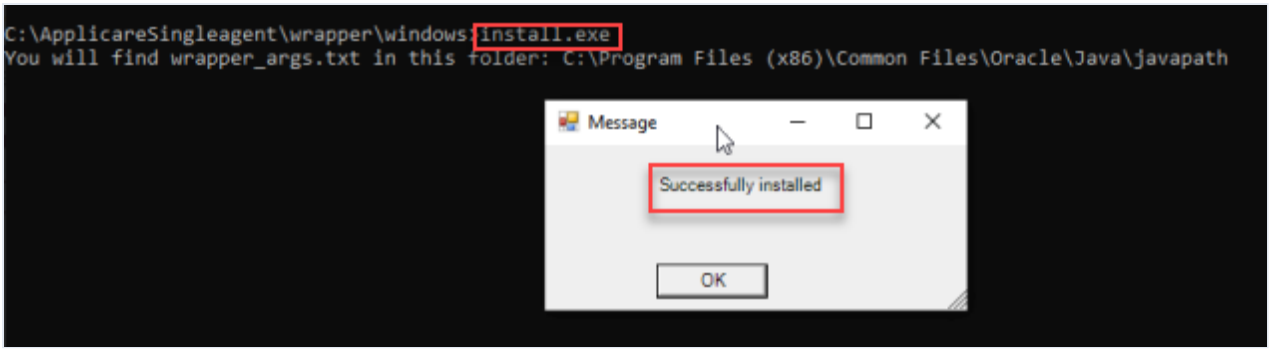
Step 1: Log in to **Applicare console,** navigate to **Download Agents** screen and download **SingleAgentWindows.** Then, Copy the **ApplicareSingleAgentWindows.zip** file to the machine where the agent will be deployed.

Step 2: Extract the **ApplicareSingleAgentWindows.zip** file into the **ApplicareSingleAgentWindows** directory.

Skip steps 1 and 2 if have already downloaded and extracted the SingleAgent.

Step 3: Navigate to the "**ApplicareSingleAgent/wrapper/Windows**" directory and run **install.exe** files as an administrator.

If you need to modify or add any additional arguments, make sure to include wrapper_args_default.txt before installing the wrapper.



If you encounter Java-related issues a problem while installing the wrapper, refer here.

Step 4: Start any Java app server.

Step 5: The agent will be automatically integrated with the Java app server and will begin sending data to Applicare controller. Log in to **Applicare Console** and navigate to the Overview screen to view the agent.

Monitoring with Applicare

CONFIGURATION GUIDES › JAVA

Install the Applicare agent using Wrapper on a Linux system

Applicare 9.2 · Updated August 23, 2024 · java, singleagent, linux · **Pending technical review**

Step 1: Log in to **Applicare console,** navigate to **Download Agents** screen and download **SingleAgentLinux.** Copy **ApplicareSingleAgentLinux.zip** file to the machine where agent will be deployed.

Step 2: Extract the **ApplicareSingleAgentLinux.zip** file to **ApplicareSingleAgentLinux** directory.

Skip steps 1 and 2 if you have already downloaded and extracted the SingleAgent.

Step 3: Navigate to "**ApplicareSingleAgent/wrapper/Linux**" directory and start **install.sh** file.



To change or add any additional arguments, include them in the wrapper_args.txt after installing the wrapper.

Note: Have a problem with Java when installing the wrapper, Refer here.

Step 4: Start any java app server.

Step 5: The agent will be automatically registered with the java app server and will begin reporting data to the Applicare controller. Log in to **Applicare Console** and navigate to Overview screen to view the agent.

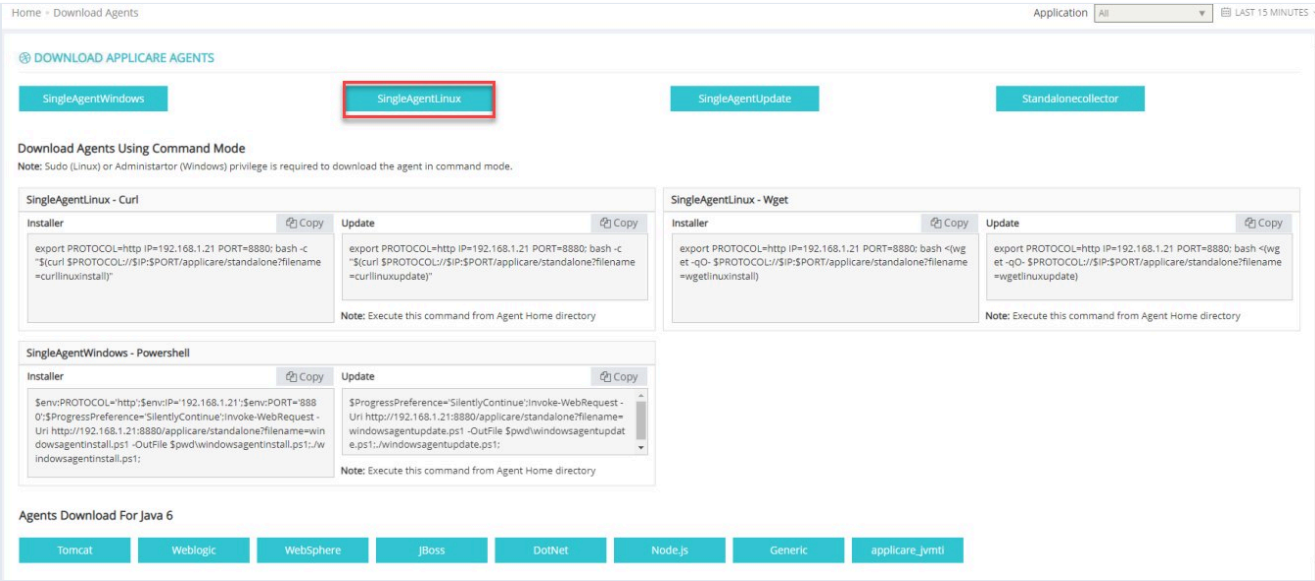
CONFIGURATION GUIDES › DOCKER

Install SingleAgent in a Docker Container

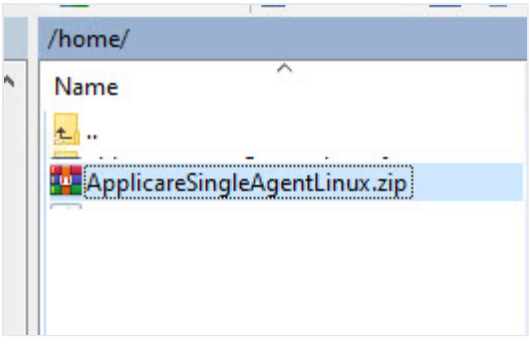
Applicare 9.2 · Updated September 13, 2024 · docker, singleagent · **Pending technical review**

Step 1, 2 and 3 are necessary only if the **ApplicareSingleAgentLinux** directory is not present on the Docker-installed machine.

Step 1: Log in to Applicare console, and navigate to the "**Download Agents**" menu, and download the **SingleAgentLinux** package.



Step 2: Transfer the **ApplicareSingleAgentLinux.zip** file to the Docker-installed machine.



Step 3: Navigate to the location where you copied the zip file, create a directory named **ApplicareSingleAgentLinux** in that location, and extract the zip file into that directory.

```bash
mkdir ApplicareSingleAgentLinux
unzip ApplicareSingleAgentLinux.zip –d ApplicareSingleAgentLinux
```

```
[root@L9-RHEL8 home]# mkdir ApplicareSingleAgentLinux
[root@L9-RHEL8 home]# unzip ApplicareSingleAgentLinux.zip -d ApplicareSingleAgentLinux
Archive:  ApplicareSingleAgentLinux.zip
 extracting: ApplicareSingleAgentLinux/agent/CAgent/agent.lib
 extracting: ApplicareSingleAgentLinux/agent/CAgent/agent.ilk
 extracting: ApplicareSingleAgentLinux/agent/CAgent/agent.exp
 extracting: ApplicareSingleAgentLinux/agent/CAgent/agent.dll
 extracting: ApplicareSingleAgentLinux/agent/CAgent/Assem.asm
```

Step 4: Access inside the Docker container.

```
BASH
docker container exec –it CONTAINER ID bash
```

e.g. docker container exec -it a0eb9355dac0 bash

```
docker container exec -it a0eb9355dac0 bash
root@a0eb9355dac0:/code#
```

Step 5: Note the docker path and **exit** from the container. Here the docker path is **a0eb9355dac0:/code**

```
TEXT
exit
```

```
root@a0eb9355dac0:/code# exit
exit
```

Step 6: Copy the **ApplicareSingleAgentLinux** directory inside Docker container

```
BASH
docker cp sourcePath/ApplicareSingleAgentLinux dockerContainerPath/ApplicareSingleAgentLinux
```

e.g. docker cp **/home**/ApplicareSingleAgentLinux **a0eb9355dac0:/code**/ApplicareSingleAgentLinux

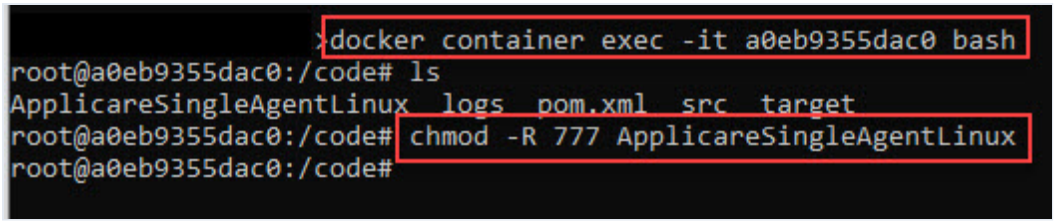
```
>docker cp /home/ApplicareSingleAgentLinux a0eb9355dac0:/code/ApplicareSingleAgentLinux
```

Step 7: Enter the Docker container and set read, write, and executable permissions for the **ApplicareSingleAgentLinux** directory.

```
BASH
docker container exec –it CONTAINER ID bash
```

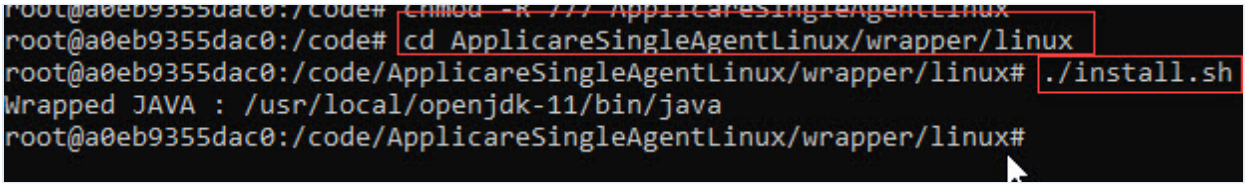
e.g. Docker container exec -it a0eb9355dac0 bash

```
BASH
chmod –R 777 ApplicareSingleAgentLinux
```



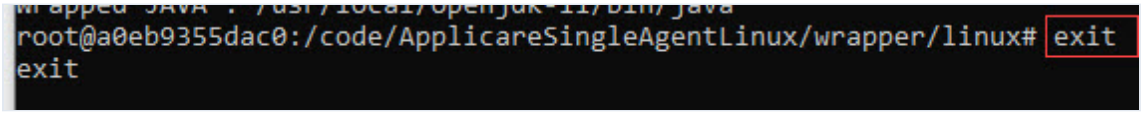
Step 8: Navigate to the **ApplicareSingleAgentLinux/wrapper/linux** directory and run **./install.sh** to install the wrapper.

```bash
cd ApplicareSingleAgentLinux/wrapper/linux
./install.sh
```



Step 9: Exit the container.

```text
exit
```



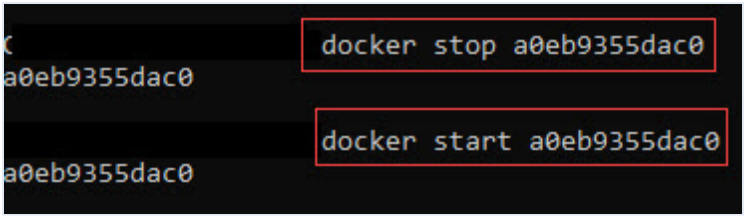
Step 10: Stop and then restart the container.

```bash
docker stop CONTAINER ID
docker start CONTAINER ID
```

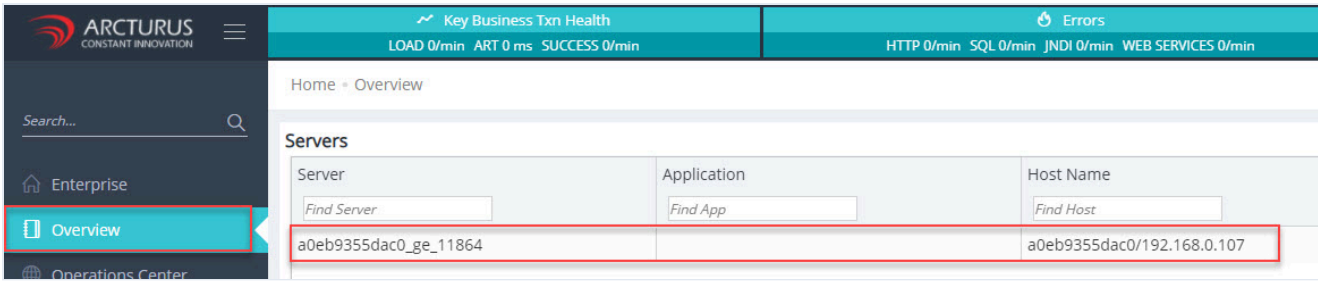
e.g.

docker stop a0eb9355dac0

docker start a0eb9355dac0



Step 11: Log in to Applicare console and navigate to the **Overview** menu to view the newly added agent.



Docs Home

Monitoring with Applicare

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent in Docker container using script

Applicare 9.2 · Updated August 21, 2024 · docker, singleagent, linux · **Pending technical review**

Manual Install:

Each time you build the Docker image, you will need to follow these manual steps.

- **Manual Install Steps**

Auto Installation:

The agent will be pre-built into the Docker image, so it will be automatically installed whenever the Docker container starts.

- **RedHat Rhel**
- **CentOS**
- **DebianOS**
- **Ubuntu**
- **Alpine**

CONFIGURATION GUIDES › DOCKER

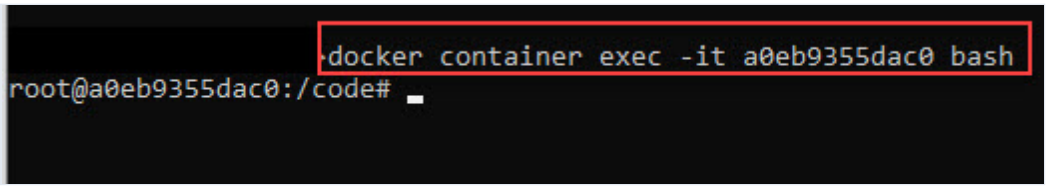
Install ApplicareSingleAgent in a Docker Container

Applicare 9.2 · Updated August 22, 2024 · docker, singleagent · **Pending technical review**

Each time you build the Docker image, follow these steps.

Step 1: Enter the Docker container.

```bash
docker container exec –it CONTAINER ID bash
```



Step 2: Create a directory named ApplicareSingleAgent

Step 3: Navigate to the **ApplicareSingleAgent** directory.

Step 4: Run the following command to download the agent

```bash
curl CONTROLLER_PROTOCOL://CONTROLLER_IP:CONTROLLER_PORT/applicare/standalone?
filename=ApplicareSingleAgentLinux.zip>ApplicareSingleAgentLinux.zip
```

Replace the below values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

e.g.

```bash
curl http://127.0.01:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
```

Note: If the curl command is unavailable, You will need to manually download the ApplicareSingleAgent.

Step 5: Extract the **ApplicareSingleAgent** zip file to that directory.

```text
unzip ApplicareSingleAgent.zip
```

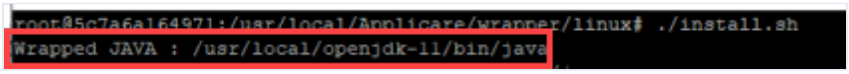
Step 6: Give read, write and executable permission for the **ApplicareSingleAgent** directory.

```bash
chmod –R 777 ApplicareSingleAgent installed directory

chmod +x ApplicareSingleAgent installed directory
```

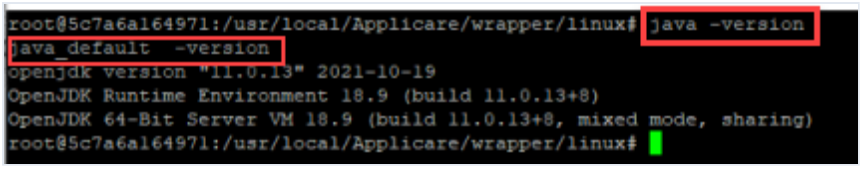
Step 7: Navigate to the **ApplicareSingleAgent installed directory/wrapper/linux** directory and execute the command to install the wrapper.

```bash
./install.sh
```



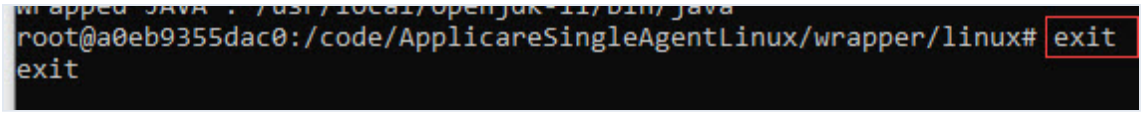
Step 8: Run the following command to verify that the Wrapper is installed.

```bash
java –version
```



Step 9: Exit the container

```text
exit
```



Step 10: Restart the container

```bash
docker restart  container ID
```

CONFIGURATION GUIDES › DOCKER

Deploy SingleAgent in a Docker Container

Applicare 9.2 · Updated September 13, 2024 · docker, singleagent · **Pending technical review**

Step 1: Add the following packages to include the **apk** command if the Docker container is running on **Alpine OS.**

curl, vim, unzip, bash, wget

```bash
Example:

RUN apk add --no-cache \
openjdk8-jre \
wkhtmltopdf \
aws-cli \
busybox-extras \
curl \
vim \
wget \
bash \
unzip \
tzdata && cp /usr/share/zoneinfo/Asia/Kolkata /etc/localtime && echo 'Asia/Kolkata' >
/etc/timezone
```

Step 2: Create a file named startAgent.sh with the following content in the directory containing the **Dockerfile**.

```bash
#!/bin/sh

SERVER_NAME=`hostname`_sa
export SERVER_NAME

ARCTURUS_HOME=/usr/local/Applicare
export ARCTURUS_HOME

JAVA_HOME_PATH=$JAVA_HOME

INBUILD_JAVA_PATH=$ARCTURUS_HOME/jre
export INBUILD_JAVA_PATH

if [[ $(uname -s) == "Darwin" ]]; then
JAVA_HOME_PATH="$ARCTURUS_HOME/jre/mac/zulu-8_jre/Contents/Home"
else
JAVA_HOME_PATH="$ARCTURUS_HOME/jre/linux/jre"
fi

JAVA_HOME_PATH="/usr/lib/jvm/java-1.8-openjdk/jre"
export JAVA_HOME_PATH

JAVA_OPTS="-noverify -Xmx1024m -Dserver.name=$SERVER_NAME -
javaagent:$ARCTURUS_HOME/applicare-db-aspects.jar=verbosity=0,logfile=false,norestart-
intellitrace=false -Darcturus_home=$ARCTURUS_HOME -XX:HeapDumpPath=/dev/null -
Djava.library.path=.:$ARCTURUS_HOME/bin -Dapplicare.standalone=true"
export JAVA_OPTS

EXECUTABLE_NAME=/usr/local/Applicare/AgentStartHelper.jar
export EXECUTABLE_NAME

# Run the JAR file
/usr/lib/jvm/java-1.8-openjdk/jre/bin/java -jar $JAVA_OPTS $EXECUTABLE_NAME
```

Step 3: Add the following line in the Dockerfile immediately after the **package** installation commands.

```
TEXT
# Create working directory for Applicare
RUN mkdir -p /usr/local/Applicare

# Download and unzip Applicare
RUN curl CONTROLLER_PROTOCOL://CONTROLLER_IP:CONTROLLER_PORT/applicare/standalone?
filename=ApplicareSingleAgentLinux.zip -o /usr/local/Applicare/ApplicareSingleAgentLinux.zip
RUN unzip /usr/local/Applicare/ApplicareSingleAgentLinux.zip -d /usr/local/Applicare

# Clean up
RUN rm -fv /usr/local/Applicare/ApplicareSingleAgentLinux.zip
RUN rm -fv /usr/local/Applicare/agentscripts/startAgent.sh

# Copy the start script
COPY startAgent.sh /usr/local/Applicare/agentscripts/startAgent.sh
RUN chmod -R 777 /usr/local/Applicare
```

Replace the below values in the above command

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

Step 4: Comment out the existing **CMD** line in the **Dockerfile** and replace it with the following CMD line.

```
TEXT
CMD ["/bin/sh", "-c", "nohup /usr/local/Applicare/agentscripts/startAgent.sh & java -jar
/usr/local/rest-TestApplication.jar"]
```

After starting the agent using the provided CMD command, it will operate similarly to running a Java application. If you need to run a different application, you can remove the current command and replace it with the appropriate command to start your desired application.

Step 5: Add and execute the following command before starting the container.

--hostname <**Server_Name**>

```
BASH
e.g.
docker run -dit -p 8086:8080 --name restapi --hostname dev c9700fdfc2f5
```

The hostname you specify will be displayed on the overview screen in Applicare.

You can use Applicare to assist with your Monitoring needs.

Monitoring with Applicare

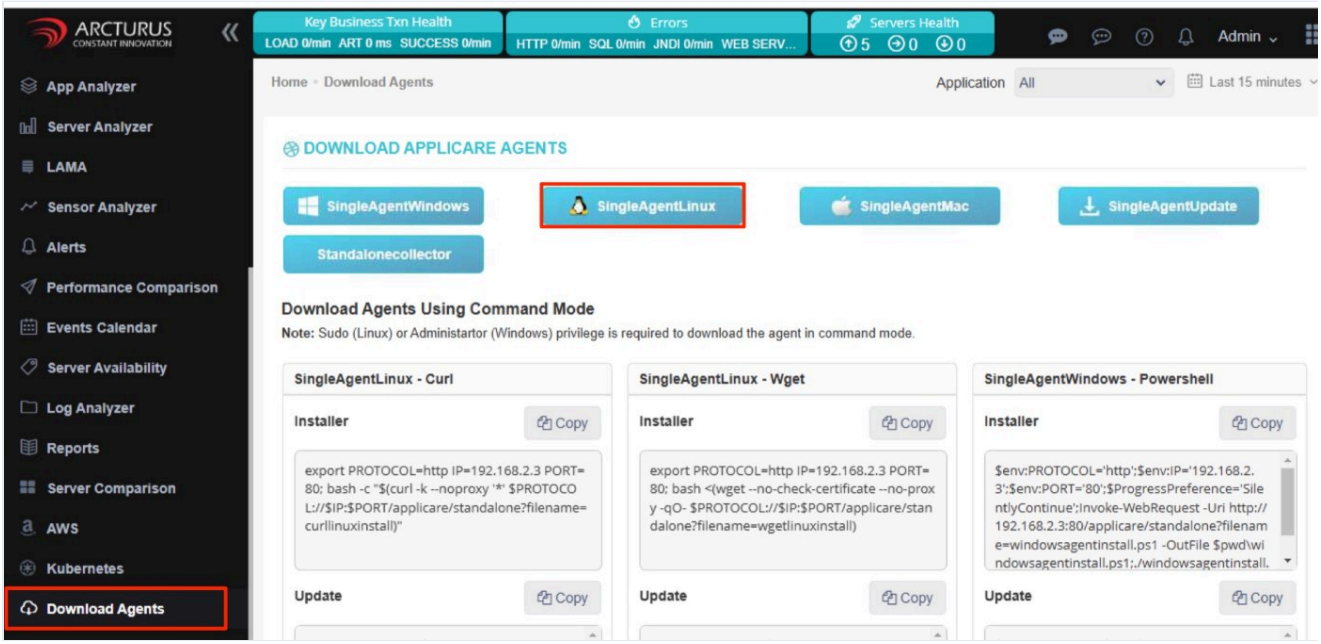
CONFIGURATION GUIDES › DOCKER

Agent configuration in docker container

Applicare 9.2 · Updated February 7, 2025 · docker, singleagent · **Pending technical review**

Note: We can easily download our agent directly into the container, but some containers don't have commands like curl or wget and unzip, so we have to download our agent manually and mount the volume to the container.

Step 1: **Login to** Applicare console**. Go to the** Download Agent **screen in the left-side menu, Download the** SingleAgentLinux **agent and extract it into the designated directory** ApplicareSingleAgentLinux **in local machine.**



Step 2: We need to delete the container in which you want to deploy the agent and then run the following command when you recreate the container.

```text
-v ApplicareSingleAgentLinuxDirectory:/usr/local/ApplicareSingleAgentLinux
--hostname The name which we need to add the agent in Applicare
```

Example:

```bash
docker run -dit -p 8080 --name tomcat-petstore -v
/root/ApplicareSingleAgentLinux:/usr/local/ApplicareSingleAgentLinux --hostname petstore
tomcat
```

Step 3: Once the container starts, execute the following command to start the agent.

```bash
docker exec –d <container_name> nohup
/usr/local/ApplicareSingleAgentLinux/agentscripts/startAgent.sh>/dev/null 2>&1 &
```

Step 4: After a while, confirm that the new agent has been added to the **Overview** screen in the **Applicare console**.

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent in a Java Jar Application Docker Container

Applicare 9.2 · Updated December 22, 2025 · docker, singleagent · **Pending technical review**

To install ApplicareSingleAgent in Java Jar application docker container, add the below script based on the OS after the **"FROM"** command to the dockerfile.

Debian/Ubuntu

Applicare Agent configuration RUN apt-get update \ && apt-get install -y curl unzip \ && rm -rf /var/lib/apt/lists/* RUN mkdir /usr/local/ApplicareSingleAgentLinux WORKDIR /usr/local/ApplicareSingleAgentLinux RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/ApplicareSingleAgentLinux/ApplicareSingleAgentLinux.zip"] RUN chmod -R 755 /usr/local/ApplicareSingleAgentLinux ENV JAVA_OPTS="-noverify -Dserver.name=**SERVER_NAME** -javaagent:/usr/local/ApplicareSingleAgentLinux/applicare-aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false -Darcturus_home=/usr/local/ApplicareSingleAgentLinux -Djava.library.path=.;/usr/local/ApplicareSingleAgentLinux/bin -Dlogback.configurationFile=/usr/local/ApplicareSingleAgentLinux/logback/logback.xml"

Replace the following values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

SERVER_NAME -> Add custom server name

Modify the ENTRYPOINT or CMD line in the Dockerfile

Original Entrypoint: ENTRYPOINT ["java", "-jar", "rest-service.jar"]

Updated Entrypoint: ENTRYPOINT ["sh", "-c", "exec java $JAVA_OPTS -jar rest-service.jar"]

Example:

```
TEXT
# Java 8 lightweight runtime
FROM eclipse-temurin:8-jre

# Applicare Agent configuration
RUN apt-get update \
&& apt-get install -y curl unzip \
&& rm -rf /var/lib/apt/lists/*
RUN mkdir /usr/local/ApplicareSingleAgentLinux
WORKDIR /usr/local/ApplicareSingleAgentLinux
RUN curl http://192.168.0.2:8880/applicare/standalone?filename=ApplicareSingleAgentLinux
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "-frv", "/usr/local/ApplicareSingleAgentLinux/ApplicareSingleAgentLinux.zip"]
RUN chmod -R 755 /usr/local/ApplicareSingleAgentLinux
ENV JAVA_OPTS="-noverify -Dserver.name=rest-api -
javaagent:/usr/local/ApplicareSingleAgentLinux/applicare-
aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false -
Darcturus_home=/usr/local/ApplicareSingleAgentLinux -
Djava.library.path=.;/usr/local/ApplicareSingleAgentLinux/bin -
Dlogback.configurationFile=/usr/local/ApplicareSingleAgentLinux/logback/logback.xml"

# Set working directory
WORKDIR /app
# Copy the JAR
COPY rest-service-0.0.1-SNAPSHOT.jar rest-service-0.0.1-SNAPSHOT.jar
# Expose application port
EXPOSE 8080

# Run the application
ENTRYPOINT ["sh", "-c", "exec java $JAVA_OPTS -jar rest-service-0.0.1-SNAPSHOT.jar"]
```

RHEL/CentOS

Applicare Agent configuration RUN yum install -y curl unzip \ && yum clean all RUN mkdir /usr/local/ApplicareSingleAgentLinux WORKDIR /usr/local/ApplicareSingleAgentLinux RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/ApplicareSingleAgentLinux/ApplicareSingleAgentLinux.zip"] RUN chmod -R 755 /usr/local/ApplicareSingleAgentLinux ENV JAVA_OPTS="-noverify -Dserver.name=**SERVER_NAME** -javaagent:/usr/local/ApplicareSingleAgentLinux/applicare-aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false -Darcturus_home=/usr/local/ApplicareSingleAgentLinux -Djava.library.path=.;/usr/local/ApplicareSingleAgentLinux/bin -Dlogback.configurationFile=/usr/local/ApplicareSingleAgentLinux/logback/logback.xml"

Replace the following values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

SERVER_NAME -> Add custom server name

Modify the ENTRYPOINT or CMD line in the Dockerfile

Original Entrypoint: ENTRYPOINT ["java", "-jar", "rest-service.jar"]

Updated Entrypoint: ENTRYPOINT ["sh", "-c", "exec java $JAVA_OPTS -jar rest-service.jar"]

Example:

```
TEXT

#  Red Hat UBI
FROM registry.access.redhat.com/ubi8/openjdk-8-runtime
# Applicare Agent configuration
RUN yum install -y curl unzip \
&& yum clean all
RUN mkdir -p /usr/local/ApplicareSingleAgentLinux
WORKDIR /usr/local/ApplicareSingleAgentLinux
RUN curl "http://192.168.0.2:8880/applicare/standalone?filename=ApplicareSingleAgentLinux" \
   -o ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip \
&& rm -f ApplicareSingleAgentLinux.zip \
&& chmod -R 755 /usr/local/ApplicareSingleAgentLinux
ENV JAVA_OPTS="-noverify -Dserver.name=rest-api \
-javaagent:/usr/local/ApplicareSingleAgentLinux/applicare-
aspects.jar=verbosity=0,logfile=false,norestart-intellitrace=false \
-Darcturus_home=/usr/local/ApplicareSingleAgentLinux \
-Djava.library.path=.;/usr/local/ApplicareSingleAgentLinux/bin \
-Dlogback.configurationFile=/usr/local/ApplicareSingleAgentLinux/logback/logback.xml"

WORKDIR /app
COPY rest-service-0.0.1-SNAPSHOT.jar rest-service-0.0.1-SNAPSHOT.jar
EXPOSE 8080
ENTRYPOINT ["sh", "-c", "exec java $JAVA_OPTS -jar rest-service-0.0.1-SNAPSHOT.jar"]
```

CONFIGURATION GUIDES › DOCKER

.NET in Docker Image

Applicare 9.2 · Updated August 23, 2024 · docker, singleagent · **Pending technical review**

Use the official Windows Server Core image FROM mcr.microsoft.com/dotnet/aspnet:7.0-windowsservercore-ltsc2019 AS base LABEL author="Cerasela Constantin"

EXPOSE 80 EXPOSE 443

create Applicare singleagent directory RUN mkdir -p C:\app\ApplicareSingleAgentWindows

set environment variable in container ENV CORECLR_ENABLE_PROFILING 1 ENV CORECLR_PROFILER {AD0AB006-9F79-43F1-B5CF-26768A70E789} ENV CORECLR_PROFILER_PATH "C:\app\Arcturus.Applicare.DotNetAgent.Profiler.dll" ENV eum.beacon **<Applicare_controller_protocol>:<controller_IP>:<controller_port>**:<controller_port>/applicare/eum/beacon ENV arceum.js **<Applicare_controller_protocol>:<controller_IP>:<controller_port>**:<controller_port>/applicare/eum/beacon/arc-eum.js ENV applicare.address **<Applicare_controller_protocol>:<controller_IP>:<controller_port>** ENV applicare.agent.name **<Server_Name>_sa** ENV DotnetAgent.httpProfilingState Enabled ENV DotnetAgent.sqlProfilingState Enabled

Download ApplicareSingleAgent and extract the contents of the zip file RUN curl -k --noproxy '*' **<Applicare_controller_protocol>:<controller_IP>:<controller_port>**/standalone?userId=0"&"filename=ApplicareSingleAgentWindows >C:/app/ApplicareSingleAgentWindows/ApplicareSingleAgentWindows.zip RUN powershell -Command Expand-Archive -Path C:/app/ApplicareSingleAgentWindows/ApplicareSingleAgentWindows.zip -DestinationPath C:/app/ApplicareSingleAgentWindows -Force

RUN curl -L https://s3.amazonaws.com/ARC_PUBLIC/DotNet.zip -o C:/app/DotNet.zip RUN powershell -Command Expand-Archive -Path C:/app/DotNet.zip -DestinationPath C:/app -Force RUN powershell -Command Expand-Archive Remove-Item -Path C:/app/DotNet.zip -Force

Enable the .Net 3.5 framework feature in container SHELL ["powershell", "-ExecutionPolicy", "Bypass"] RUN powershell -ExecutionPolicy Bypass -Command { Invoke-WebRequest -Uri https://dotnetbinaries.blob.core.windows.net/dockerassets/microsoft-windows-netfx3-1809.zip -OutFile C:/microsoft-windows-netfx3.zip } RUN powershell -Command Expand-Archive -Path C:/microsoft-windows-netfx3.zip -DestinationPath C:/packages RUN powershell -Command Expand-Archive Remove-Item -Path C:/microsoft-windows-netfx3.zip -Force RUN Install-WindowsFeature -Name NET-Framework-Features -Source C:/packages -Verbose

Copy the Dotnet Application to container. WORKDIR /app COPY /net7.0/* ./

Start the Dotnet Agent, SingleAgent and Dotnet application. ENTRYPOINT ["cmd", "/C", "start", "/B", "Arcturus.Applicare.DotNetAgent.Console.exe", "&&", "cmd", "/C", "start", "/B", "C:/app/ApplicareSingleAgentWindows/startAgent.bat", "&&", ".net", "<**Application_name**>.dll"]

Replace the below values in the above script

Applicare_controller_protocol -> http (or) https

controller_IP -> Applicare Controller IP

controller_port -> Applicare Controller Port

Application_name -> Name the file for the .NET application

Docker

If you are running the .NET image in a Docker container, Add and execute the following command before starting the container.

```
TEXT
--hostname <Server_Name>
```

Server_Name = Give the same name here as already given in dotnet dockerfile.

Kubernetes

If running the .NET image on a Kubernetes pod, Add the following command to the YAML file, placing it next to the container 'spec' line.

```
YAML
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  hostname: <Server_Name>
  containers:
    - name: Dotnet
      image: Nop
```

(OR)

```
YAML
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  env:
   - name: hostname
     value: <Server_Name>
  containers:
    - name: Dotnet
      image: Nop
```

Server_Name = Use the same name here as already given in .NET Dockerfile.

CONFIGURATION GUIDES › DOCKER

Applicare deployed in a Docker container with Derby database

Applicare 9.2 · Updated February 4, 2025 · docker, singleagent · **Pending technical review**

Pre Requisite: Make sure docker is installed before following the below steps.

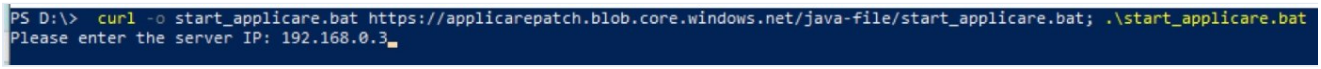
Windows:

Step 1: Launch PowerShell as an Administrator and execute the following command.

```bat
 curl –o start_applicare.bat https://applicarepatch.blob.core.windows.net/java-file/start_applicare.bat; .\start_applicare.bat
```

Step 2: Then enter the **server IP** when prompted.

server IP -> The IP of the machine where docker is running.



Step 3: After few minutes you can access the Applicare console in the browser.

```text
http://<server IP>:8880/applicare
```

server IP ->The IP of the machine where docker is running.

Linux:

Step 1: Download the file "**start_applicare_linux.sh**" using either **curl** or **wget**, depending on your preference.

Curl:

```bash
curl –o start_applicare_linux.sh https://applicarepatch.blob.core.windows.net/java-file/start_applicare_linux.sh
```

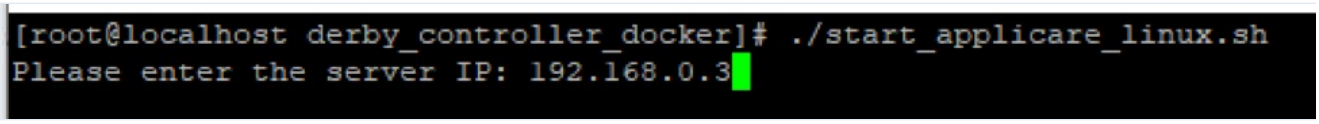
Or

Wget:

```bash
wget https://applicarepatch.blob.core.windows.net/java-file/start_applicare_linux.sh
```

Step 2: After download the file give the execute permission for that file. Then run the file in putty season, then enter the **server IP** when prompted.

server IP -> The IP of the machine where docker is running.



Step 3: After few minutes can access the console in the browser.

```
TEXT
http://<server IP>:8880/applicare
```

server IP ->The IP of the machine where docker is running.

Mac:

Step 1: Download the file "**start_applicare_mac.sh**" using either **curl** or **wget**, depending on your preference.

Curl:

```
BASH
curl -o start_applicare_mac.sh
https://applicare.blob.core.windows.net/docker/start_applicare_mac.sh
```

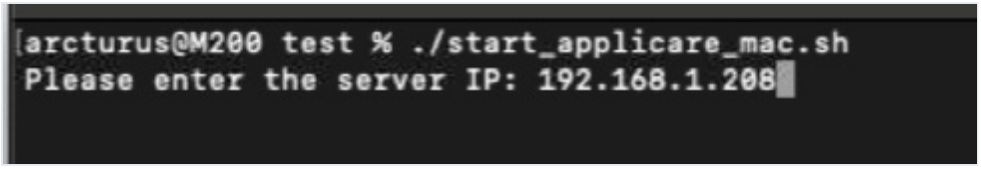
Or

Wget:

```
BASH
wget https://applicare.blob.core.windows.net/docker/start_applicare_mac.sh
```

Step 2: After download the file give the execute permission for that file. Then run the file in terminal window, Then enter the **server IP** when prompted.

server IP -> The IP of the machine where docker is running.



Step 3: After few minutes can access the console in the browser.

```
TEXT
http://<server IP>:8880/applicare
```

server IP ->The IP of the machine where docker is running.

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent on Ubuntu

Applicare 9.2 · Updated October 16, 2024 · docker, singleagent, linux · **Pending technical review**

Step 1: Insert the following scripts after the "FROM" command in the Dockerfile.

RUN apt-get update && \ apt-get install -y unzip net-tools RUN mkdir /usr/local/Applicare WORKDIR /usr/local/Applicare RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux.zip >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"] WORKDIR /usr/local/Applicare WORKDIR /usr/local/Applicare/wrapper/linux RUN chmod +x install.sh RUN chmod -R 777 /usr/local/Applicare RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]

Update the values in the script above as Indicated

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

Example:

```bash
FROM tomcat:8
RUN  apt-get update && \
     apt-get install -y unzip
RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
WORKDIR /usr/local/Applicare
WORKDIR /usr/local/Applicare/wrapper/linux
RUN chmod +x install.sh
RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]
```

Step 2: Specify the server name when creating the Docker container.

When creating the Docker container, set the server name in the environment variable with -esname = server name, then start the container. Replace Servername with the desired name for the Applicare agent, ensuring it is unique across different containers.

e.g.

-e sname=tomcatcontainer

BASH

```
docker container run –it –d –p 8065:8080 ––name container1 –e sname=tomcatcontainer 1a2
```

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent on Alpine Linux

Applicare 9.2 · Updated October 16, 2024 · docker, singleagent, linux · **Pending technical review**

Step 1: Add the following scripts immediately after the FROM command in the Dockerfile.

RUN apk update && apk add --no-cache curl vim wget bash unzip util-linux eudev-libs net-tools RUN mkdir /usr/local/Applicare WORKDIR /usr/local/Applicare RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux.zip >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"] WORKDIR /usr/local/Applicare WORKDIR /usr/local/Applicare/wrapper/linux RUN chmod +x install.sh RUN chmod -R 777 /usr/local/Applicare RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]

Replace the below values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

e.g.

```
TEXT
FROM eclipse-temurin:17-jre-alpine
RUN apk update && apk add --no-cache curl vim wget bash unzip util-linux
RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
WORKDIR /usr/local/Applicare
WORKDIR /usr/local/Applicare/wrapper/linux
RUN chmod +x install.sh
RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]
```

Step 2: Specify the server name when creating the Docker container.

While creating the docker container add the server name to the environment variable using **-e sname=<SERVERNAME>** and start the container. Replace **<SERVERNAME>** with the name which we need to add the agent in Applicare. The **<SERVERNAME>** should be **unique for different containers.**

e.g.

-e sname=tomcatcontainer

BASH

```
docker container run –it –d –p 8065:8080 ––name container1 –e sname=tomcatcontainer
19f36bd86c9c
```

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent on CentOS

Applicare 9.2 · Updated October 16, 2024 · docker, singleagent, linux · **Pending technical review**

Step 1: Insert the following scripts immediately after the FROM command in the Dockerfile.

RUN apt-get update && \ apt-get install -y unzip net-tools RUN mkdir /usr/local/Applicare WORKDIR /usr/local/Applicare RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux.zip >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"] WORKDIR /usr/local/Applicare WORKDIR /usr/local/Applicare/wrapper/linux RUN chmod +x install.sh RUN chmod -R 777 /usr/local/Applicare RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]

Replace the below values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

e.g.

```bash
FROM tomcat:8
RUN apt-get update && \
    apt-get install -y unzip
RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
WORKDIR /usr/local/Applicare
WORKDIR /usr/local/Applicare/wrapper/linux
RUN chmod +x install.sh
RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]
```

Step 2: Specify the server name when creating the Docker container.

While creating the docker container add the server name to the environment variable using **-e sname=<SERVERNAME>** and start the container. Replace **<SERVERNAME>** with the name which we need to add the agent in Applicare. The **<SERVERNAME>** should be **unique for different containers.**

e.g. -e sname=tomcatcontainer

BASH

```
docker container run –it –d –p 8065:8080 ––name container1 –e sname=tomcatcontainer
1a28b359235a
```

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent in DebianOS

Applicare 9.2 · Updated October 16, 2024 · docker, singleagent, linux · **Pending technical review**

Step 1: Insert the following scripts immediately after the FROM command in the Dockerfile.

RUN yum update && \ yum -y install wget unzip net-toolsRUN mkdir /usr/local/Applicare WORKDIR /usr/local/Applicare RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux.zip >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"] WORKDIR /usr/local/Applicare WORKDIR /usr/local/Applicare/wrapper/linux RUN chmod +x install.sh RUN chmod -R 777 /usr/local/Applicare RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]

Replace the below values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

Example:

```bash
FROM tomcat:8
RUN yum update && \
    yum –y install wget unzip
RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "–frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
WORKDIR /usr/local/Applicare
WORKDIR /usr/local/Applicare/wrapper/linux
RUN chmod +x install.sh
RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]
```

Step 2: Add server name while creating a Docker container.

When creating the Docker container, set the server name as an environment variable using **-e sname= <SERVERNAME> and start the container. Replace <SERVERNAME> with the name you want to assign to the agent in Applicare.** The **<SERVERNAME>** should be unique for each container.

e.g.

-e sname=tomcatcontainer

BASH

```
docker container run –it –d –p 8065:8080 ––name container1 –e sname=tomcatcontainer
1a28b359235a
```

CONFIGURATION GUIDES › DOCKER

Install Applicare SingleAgent in a RedHat Enterprise Linux (RHEL) Docker Image

Applicare 9.2 · Updated October 16, 2024 · docker, singleagent, linux · **Pending technical review**

Step 1: Add the following scripts after the command in the Dockerfile.

RUN dnf -y update RUN dnf -y install unzip RUN dnf -y install net-tools RUN mkdir /usr/local/Applicare WORKDIR /usr/local/Applicare RUN curl **CONTROLLER_PROTOCOL**://**CONTROLLER_IP**:**CONTROLLER_PORT**/applicare/standalone?filename=ApplicareSingleAgentLinux.zip >ApplicareSingleAgentLinux.zip RUN unzip ApplicareSingleAgentLinux.zip RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"] WORKDIR /usr/local/Applicare WORKDIR /usr/local/Applicare/wrapper/linux RUN chmod +x install.sh RUN chmod -R 777 /usr/local/Applicare RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]

Replace the below values in the above script

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

Example:

```
TEXT
FROM tomcat:8
RUN dnf –y update
RUN dnf –y install unzip
RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip
>ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "–frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
WORKDIR /usr/local/Applicare
WORKDIR /usr/local/Applicare/wrapper/linux
RUN chmod +x install.sh
RUN ["/usr/local/Applicare/wrapper/linux/install.sh"]
```

Step 2: Specify the server name when creating the Docker container.

When creating the Docker container, set the server name as an environment variable using **-e sname= <SERVERNAME> then start the container. Replace <SERVERNAME> with the desired name to register the agent in Applicare**. Ensure that **<SERVERNAME> is unique for each container**.

e.g.

-e sname=tomcatcontainer

```bash
docker container run –it –d –p 8065:8080 ––name container1 –e sname=tomcatcontainer
1a28b359235a
```

CONFIGURATION GUIDES › DOCKER

Configuring an agent within an Nginx container using a (docker-compose.yml) file

Applicare 9.2 · Updated February 27, 2025 · docker, nginx · **Pending technical review**

Step 1: Create a file called **custom_nginx.conf** with the content below.

```text
user nginx;
worker_processes auto;
pid /var/run/nginx.pid;

events {
     worker_connections 1024;
}

http {
    log_format main '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                    '$status $body_bytes_sent || "$request_body" || "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

   access_log /var/log/access.log main;
   access_log /dev/stdout main;

   include /etc/nginx/conf.d/*.conf;
}
```

Step 2: Add the following **location** block to the **server** block in the **nginx.conf** or **default.conf** file you are already using, which is loaded as a Volume mount into the nginx container.

```text
location /nginx-status {
stub_status on;
allow 127.0.0.1;
allow all;
}
```

```
server {
        listen 80;
        server_name _;
        root /srv;                  # Default root directory for the server
        disable_symlinks off;       # Allow symlinks

        location /nginx-status {
            stub_status on;
            allow 127.0.0.1;
            allow all;
        }

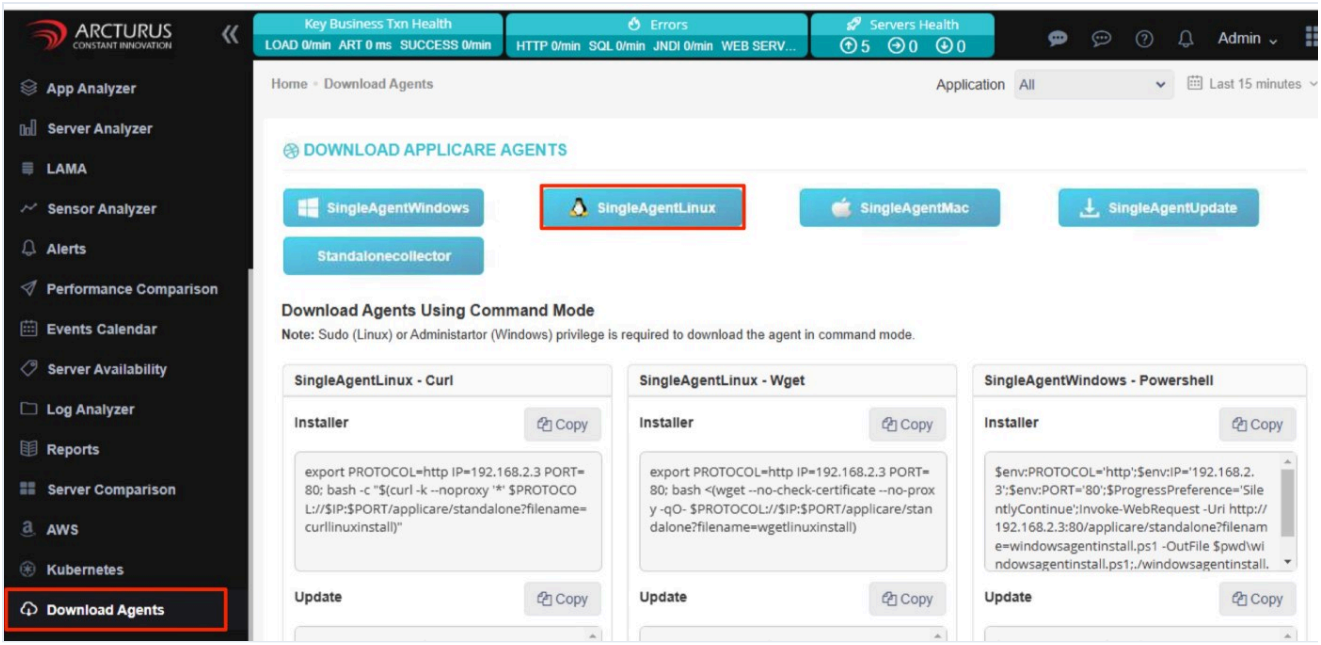
        location /petstore {
            retrun 301 http://192.168.4.48:8880;
        }
}
```

Step 3: If an **accesslog** line exists in the **nginx.conf** or **default.conf** file, comment it out.

Step 4: Download the file **agent_startup.sh** ***(sample file, available on the documentation site)*** from this link.

Note: While our agent can be directly downloaded into the nginx container, some containers may lack essential commands like curl or wget and unzip, In such cases, the agent must be downloaded manually and mounted as a volume in the container.

Step 5: Log in to Applicare console. Go to the Download Agent screen in the left-side menu, Download the SingleAgentLinux agent and extract it into the designated directory ApplicareSingleAgentLinux in host machine.



Step 6: In the '**agentscripts**' directory within the extracted **ApplicareSingleAgentLinux** folder, locate and edit the '**startAgent.sh**' file. Before the line containing '**JAVA_OPTS**', insert the following line and save the changes.

```bash
JAVA_HOME_PATH=/usr/bin/java
export JAVA_HOME_PATH
```

```
if [[ $(uname -s) == "Darwin" ]]; then
  JAVA_HOME_PATH="$ARCTURUS_HOME/jre/mac/zulu-8_jre/Contents/Home"
else
  JAVA_HOME_PATH="$ARCTURUS_HOME/jre/linux/jre"
fi
#export JAVA_HOME

JAVA_HOME_PATH=/usr/bin/java
export JAVA_HOME_PATH

JAVA_OPTS="-noverify -Xmx1024m -Dserver.name=$SERVER_NAME -javaagent:$
#JAVA_OPTS=" $JAVA_OPTS -Darcturus_info_pass=arcturus -Darcturus_debug
export JAVA_OPTS
```

Step 7: Add the following command in your existing **docker-compose.yml** file.

hostname: **The name which we need to add the agent in Applicare** entrypoint: ["sh", "-c"]

command: > 'apk add --no-cache openjdk8-jre && chmod +x /usr/local/agent_startup.sh && /usr/local/agent_startup.sh && nginx -g "daemon off;"'

Add the following line to the Volume mount line in that yml file.

- **where the agent_startup.sh file is located**/agent_deploy.sh:/usr/local/agent_startup.sh - **where the agent directory is located**/ApplicareSingleAgentLinux:/usr/local/ApplicareSingleAgentLinux - **where the custom_nginx.conf file is located**/custom_nginx.conf:/etc/nginx/nginx.conf

Example:

```bash
version: '3.8'

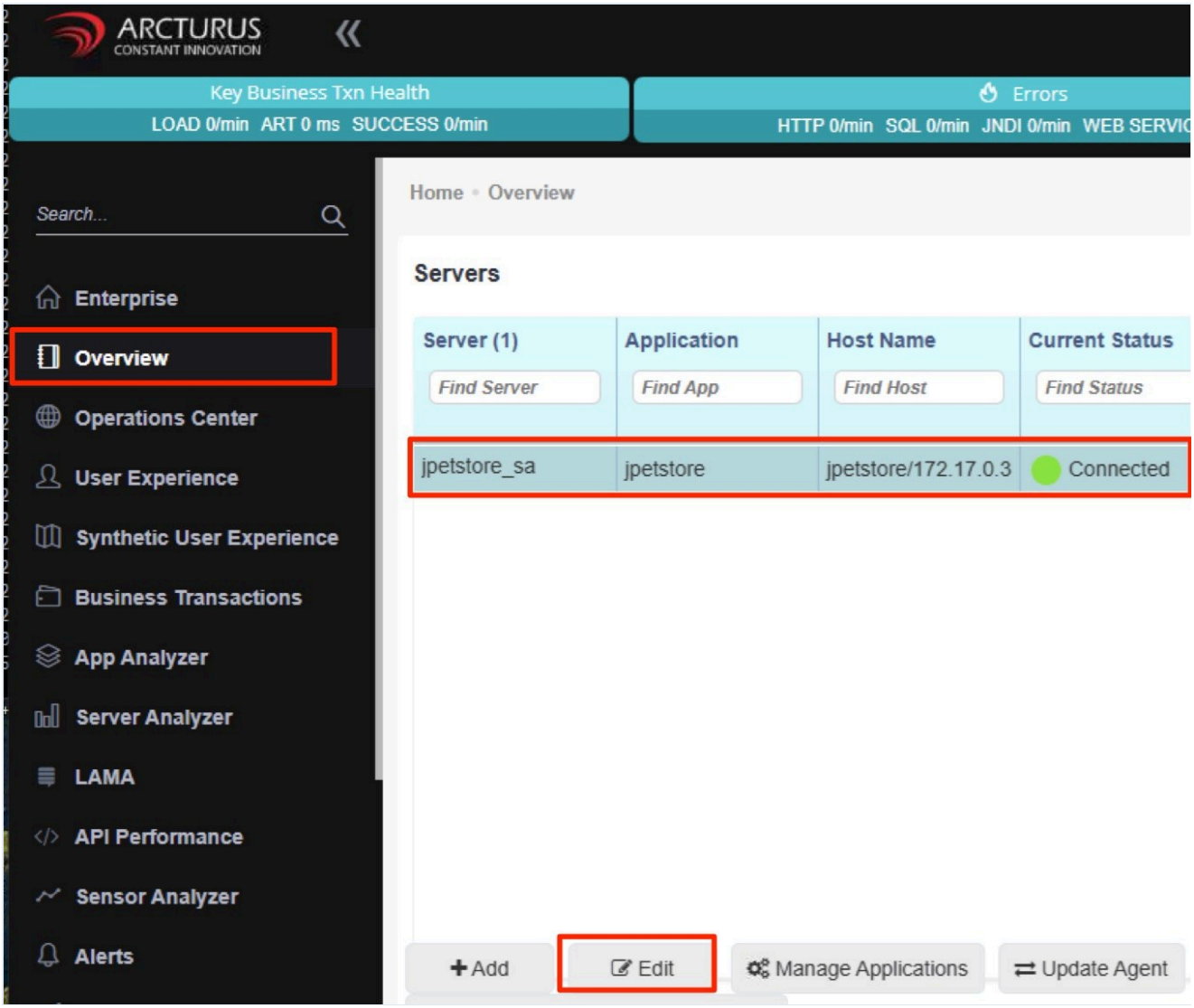
services:
nginx:
image: nginx:1.23-alpine
container_name: nginx-server
restart: always

hostname: nginx_agent
entrypoint: ["sh", "-c"]
command: >
      'apk add --no-cache openjdk8-jre &&
      chmod +x /usr/local/agent_startup.sh &&
      /usr/local/agent_startup.sh &&
      nginx -g "daemon off;"'

ports:
- "85:80"
- "443:443"

volumes:
- /usr/local/agent_startup.sh:/usr/local/agent_startup.sh
- /usr/local/ApplicareSingleAgentLinux:/usr/local/ApplicareSingleAgentLinux
- /usr/local/custom_nginx.conf:/etc/nginx/nginx.conf
```

Step 8: After the container starts, the agent is automatically added with the Applicare controller. To manage it, log in to the Applicare console and go to the Overview menu. Find the newly added agent, select it, and click **Edit** to configure as needed.



Note: If the sub_status module is available, continue with Step 9. Otherwise, skip Step 9 and move directly to Step 10.

Step 9: Enable the "**Add Nginx Configuration**" checkbox and add the below values.

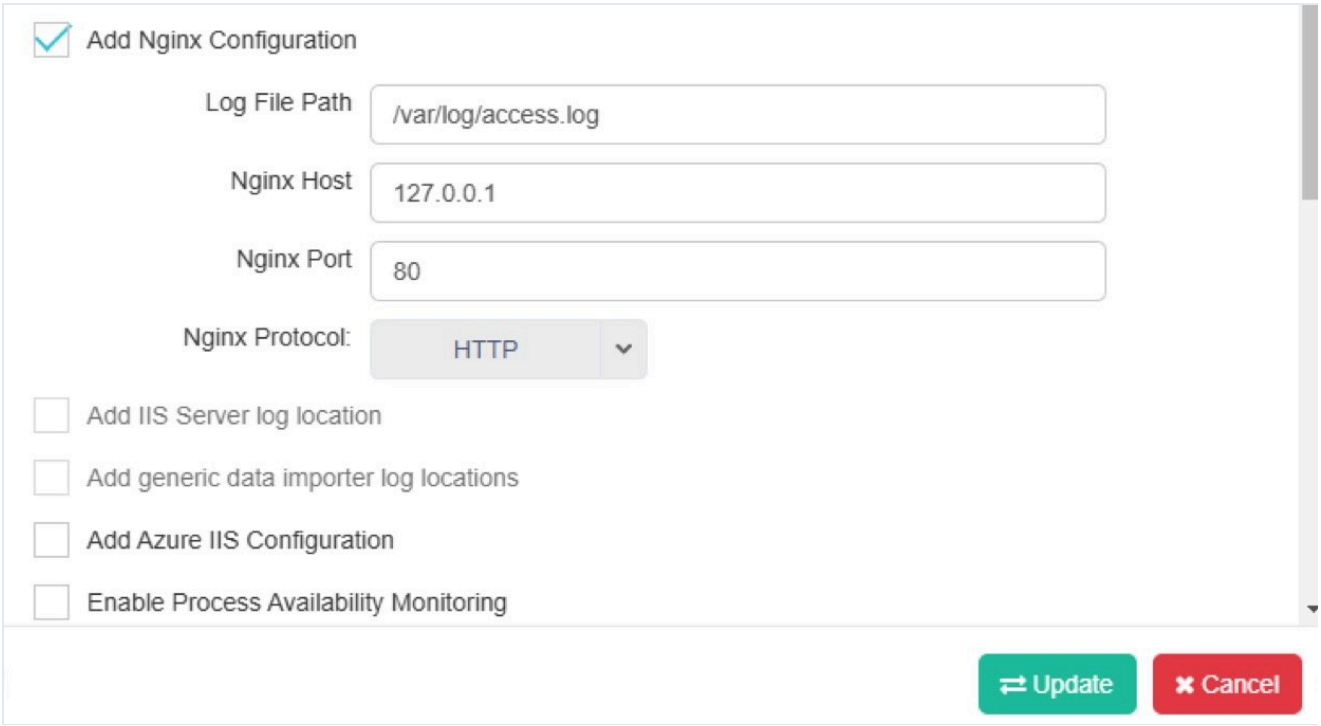
1. Log file path

2. Nginx Host

3. Nginx Port

4. Nginx Protocol

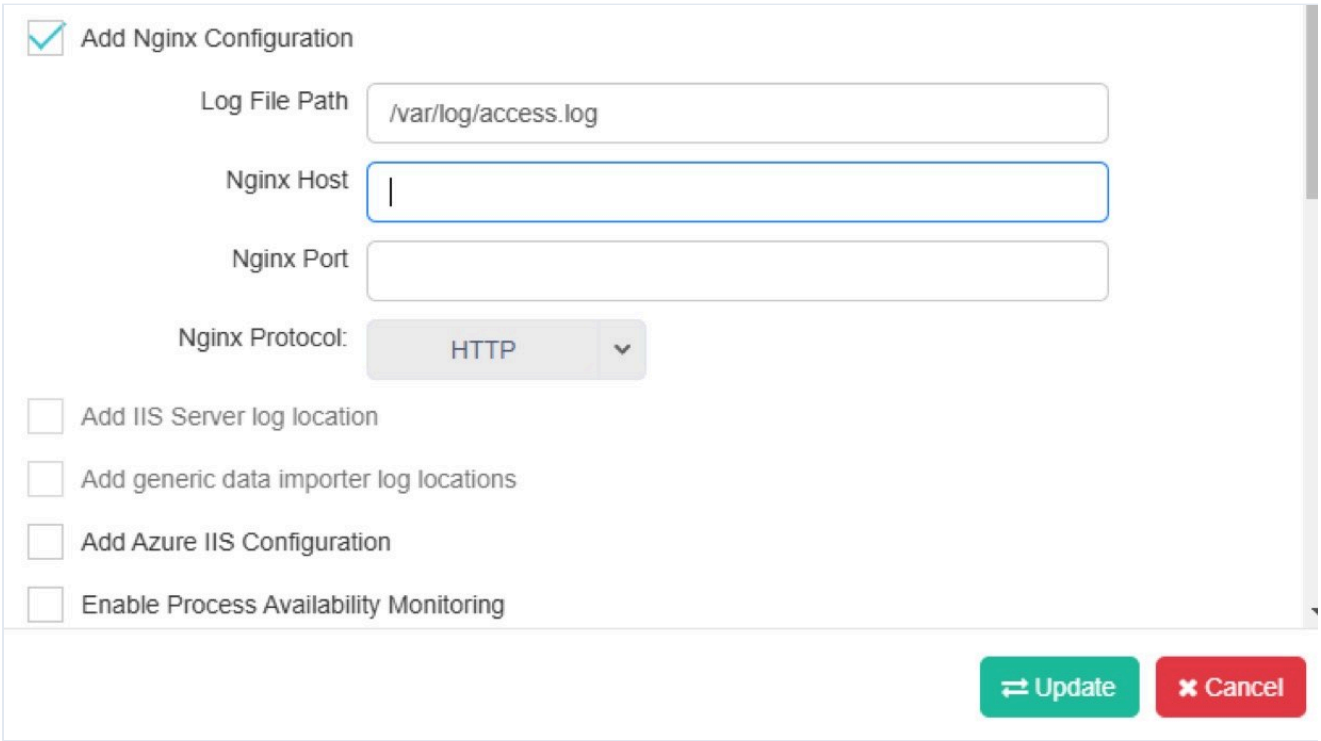
- Click the **update** button to save.



Step 10: Select the "**Add Nginx Configuration**" checkbox and add the below values.

1. Log file path

- Click the **update** button to save.



CONFIGURATION GUIDES › DOCKER

Agent configuration in Nginx container

Applicare 9.2 · Updated September 16, 2025 · docker, nginx · **Pending technical review**

If the nginx.conf file is not on the local machine, copy the file from the nginx container and save it on the local machine.

Run the following command to copy the nginx.conf file from nginx container to local machine, replacing <container_id> with your actual container ID or name

```bash
docker cp <container_id>:/etc/nginx/nginx.conf ./nginx.conf
```

Note: The following changes need to be made to the nginx.conf file that is passed from the local machine to the nginx container.

Step 1: Edit the nginx.conf file and comment out the existing log format and replace it with the following log_format.

```text
log_format main '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                        '$status $body_bytes_sent || "$request_body" || "$http_referer" '
                        '"$http_user_agent" "$http_x_forwarded_for"';
```

```
log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                  '$status $body_bytes_sent ||  "$request_body" || "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

access_log  logs/access.log  main;
```

Step 2: If the access_log line is commented out, Please uncomment it.

```
log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                  '$status $body_bytes_sent ||  "$request_body" || "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

access_log  logs/access.log  main;
```

Note: If the sub_status module is available, proceed with Step 3 as outlined. However, if the module is not available, skip Step 3 and proceed directly to Step 4.

Step 3: Insert the following lines within the **server** module.

location /nginx-status { stub_status on; allow 127.0.0.1; deny all; }

```
server {
    listen       82;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

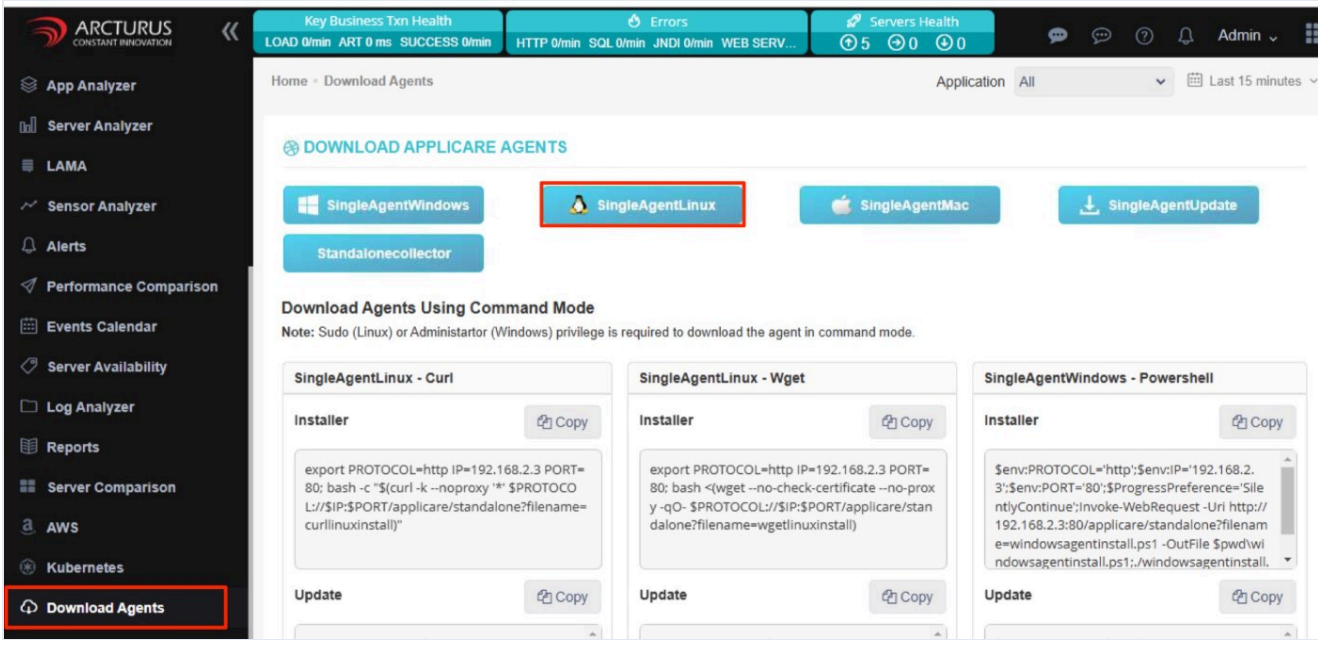
    location / {
        root   html;
        index  index.html index.htm;
    }

    location /nginx-status {
        stub_status on;
        allow 127.0.0.1;
        deny all;
    }
```

Step 4: Download the file **agent_deploy.sh** from this link.

Note: We can easily download our agent directly into the nginx container, but some containers do not have commands like curl or wget and unzip, so we have to download our agent manually and mount the volume to the container.

Step 5: Login to Applicare console. Go to the Download Agent screen in the left-side menu, Download the SingleAgentLinux agent and extract it into the designated directory ApplicareSingleAgentLinux in local machine.



Step 6: Add the following command along with the command that starts the container.

Add the following command before the image name.

```
TEXT
-v where the agent_deploy.sh file is located/agent_deploy.sh:/usr/local/agent_deploy.sh-v
ApplicareSingleAgentLinuxDirectory:/usr/local/ApplicareSingleAgentLinux--entrypoint sh --
hostname The name which we need to add the agent in Applicare
```

Add the following command after the image name.

```
TEXT
-c 'rm /var/log/nginx/access.log && touch /var/log/nginx/access.log && chown nginx:nginx
/var/log/nginx/access.log && chmod +x /usr/local/agent_deploy.sh &&
/usr/local/agent_deploy.sh && nginx -g "daemon off;"'
```

Example:

```
BASH
docker run -d --name nginx -v /home/nginx/nginx.conf:/etc/nginx/nginx.conf:ro -v
/var/run/docker.sock:/tmp/docker.sock:ro  -p 81:80  -v
/home/nginx/agent_deploy.sh:/usr/local/agent_deploy.sh -v
/root/ApplicareSingleAgentLinux:/usr/local/ApplicareSingleAgentLinux  --entrypoint sh --
hostname jpetstore nginx -c 'rm /var/log/nginx/access.log && touch /var/log/nginx/access.log
&& chown nginx:nginx /var/log/nginx/access.log && chmod +x /usr/local/agent_deploy.sh &&
/usr/local/agent_deploy.sh && nginx -g "daemon off;"'
```

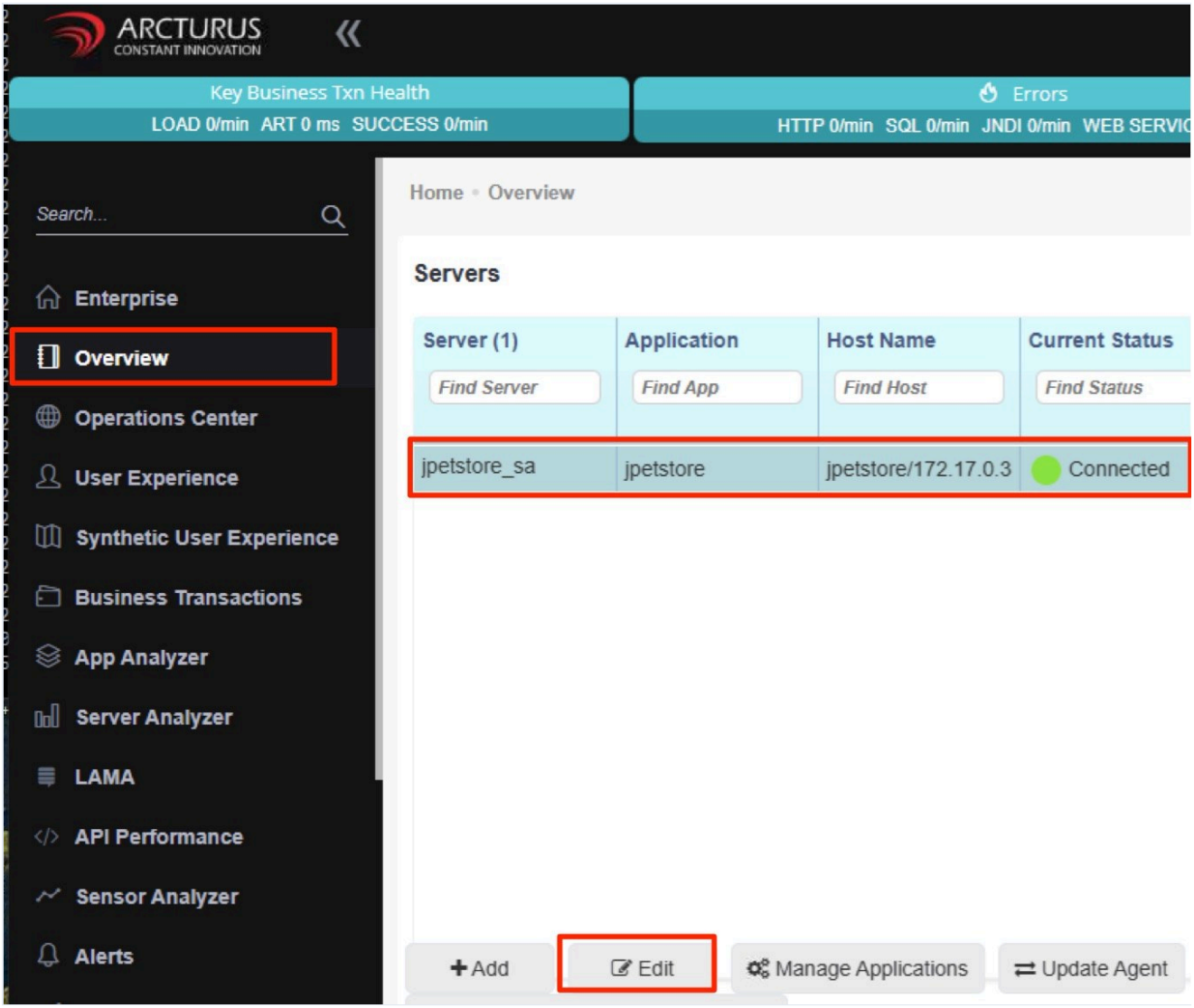
Note:

rm /var/log/nginx/access.log= Deletes the file or symbolic link at /var/log/nginx/access.log

touch /var/log/nginx/access.log= Creates a new, empty file at /var/log/nginx/access.log if it doesn't already exist.

chown nginx:nginx /var/log/nginx/access.log= Changes the ownership of /var/log/nginx/access.log to the nginx user and group.

Step 7: Once the container starts, the agent will be automatically added to the Applicare controller. To manage it, log in to the Applicare console and go to the Overview menu. Find the newly added agent, select it, and click **Edit** to configure as needed.



Note: If the sub_status module is available, proceed with Step 8 as outlined. However, if the module is not available, skip Step 8 and proceed directly to Step 9.

Step 8: Enable the "**Add Nginx Configuration**" checkbox and add the below values.

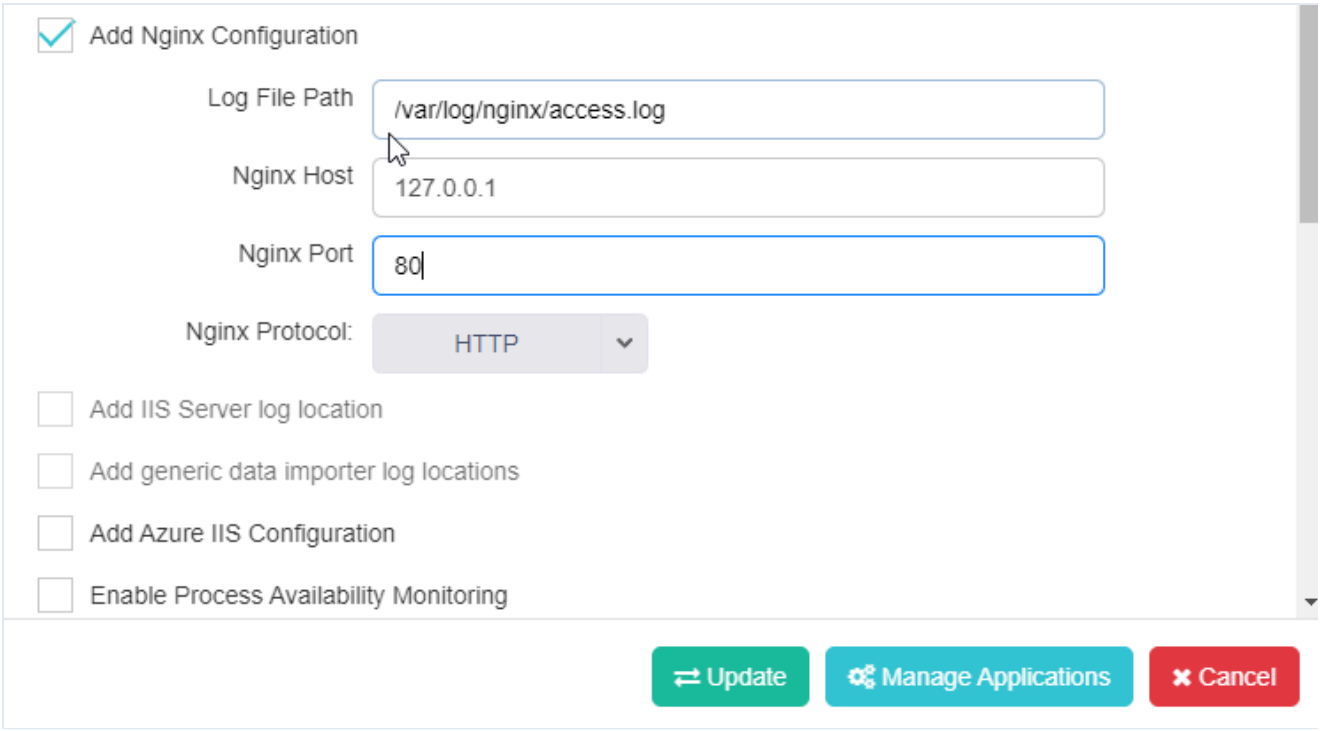
1. Log file path

2. Nginx Host

3. Nginx Port

4. Nginx Protocol

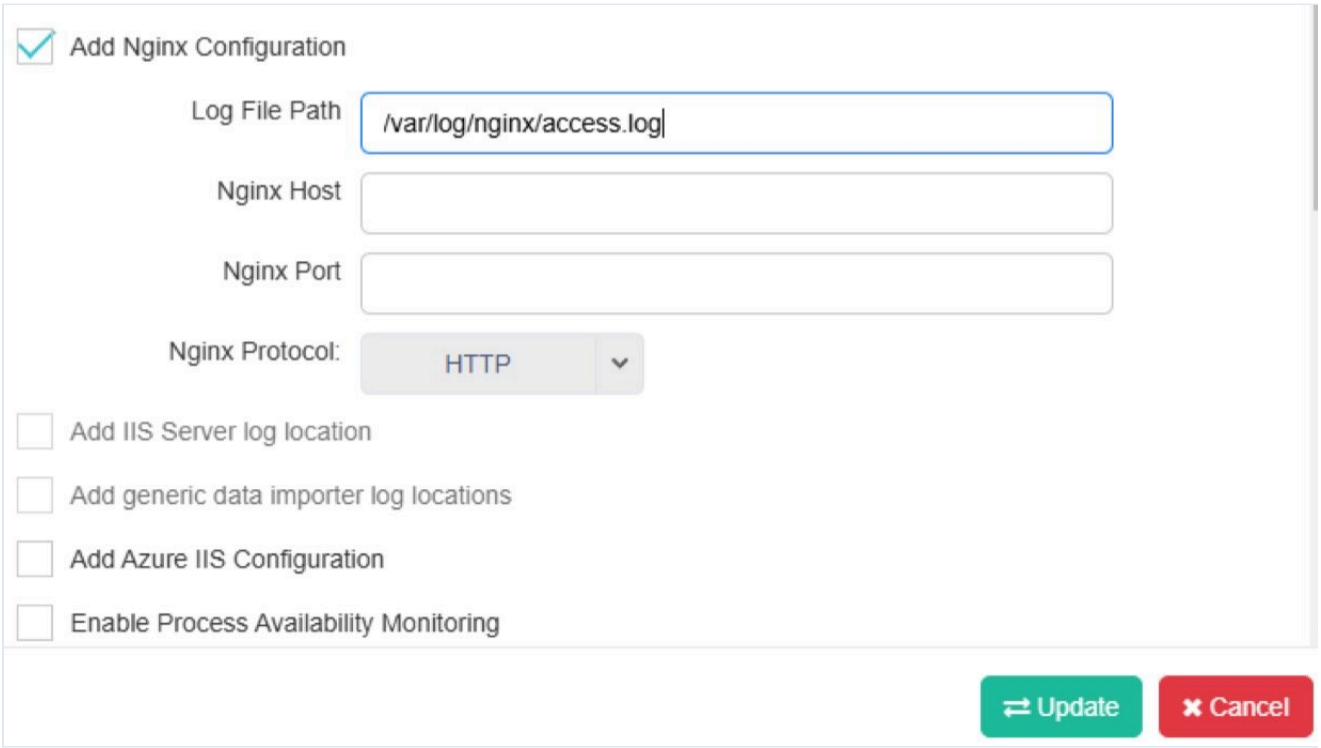
- Click the **update** button to save.



Step 9: Enable the "**Add Nginx Configuration**" checkbox and add the below values.

1. Log file path

- Click the **update** button to save.



CONFIGURATION GUIDES › DOCKER

Deploy the agent in the Nginx-proxy container

Applicare 9.2 · Updated February 27, 2025 · docker, nginx · **Pending technical review**

Note: This step is only necessary when initially creating the container that deploys the agent. For subsequent operations, such as stopping and restarting the container, the steps below are not needed.

Step 1: Edit the default.conf file and comment out the existing log format and replace it with the following log_format.

```
TEXT
log_format main escape=default '$remote_addr – $remote_user $time_iso8601 [$request_time]
"$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
"$http_x_forwarded_for" "$upstream_addr" $host';
```

```
gzip_types text/plain text/css application/javascript application/json application/x-javascript text/xml application/xml application/xml+rss
#log_format vhost escape=default '$host $remote_addr - $remote_user [$time_local] "$request" $status $body_bytes_sent "$http_referer" "$http
log_format vhost escape=default '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" $status $body_bytes_sent "$http_refere
access_log off;
```

Step 2: If there is a line called **access_log off**, comment out that line and add it to the below condition in the next line.

```
TEXT
access_log /var/log/nginx/access.log main;
access_log /dev/stdout main;
```

```
    log_format vhost escape=default '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request"
    #access_log off;
    access_log /var/log/nginx/access.log vhost;
    access_log /dev/stdout vhost;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA
        ssl_prefer_server_ciphers off;
    error_log /dev/stderr;
```

Note: If the sub_status module is available, proceed with Step 3 as outlined. However, if the module is not available, skip Step 3 and proceed directly to Step 4.

Step 3: Add the following lines into the http server module.

```
YAML
Syntax:
location /nginx-status {stub_status on;allow IPADDRESS;
deny all;}

Example:
location /nginx-status {stub_status on;allow 192.168.1.7;
allow  all;}
```

```
server {
    server_name _; # This is just an invalid value which will never trigger on a real hostname.
    server_tokens off;
    access_log /var/log/nginx/access.log vhost;
    access_log /dev/stdout vhost;
    http2 on;
    listen 80;
    listen 443 ssl;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
    # No default certificate found, so reject SSL handshake;
    ssl_reject_handshake on;
    location ^~ / {
        return 503;
    }
    location /nginx-status {
        stub_status on;
        allow 192.168.1.8;
        allow all;
        }
}
```

Step 4: Download the file **agent_deploy.sh** ***(sample file, available on the documentation site)*** from this link. Edit the file and change the **protocol**, **IP**, and **port** as shown below.

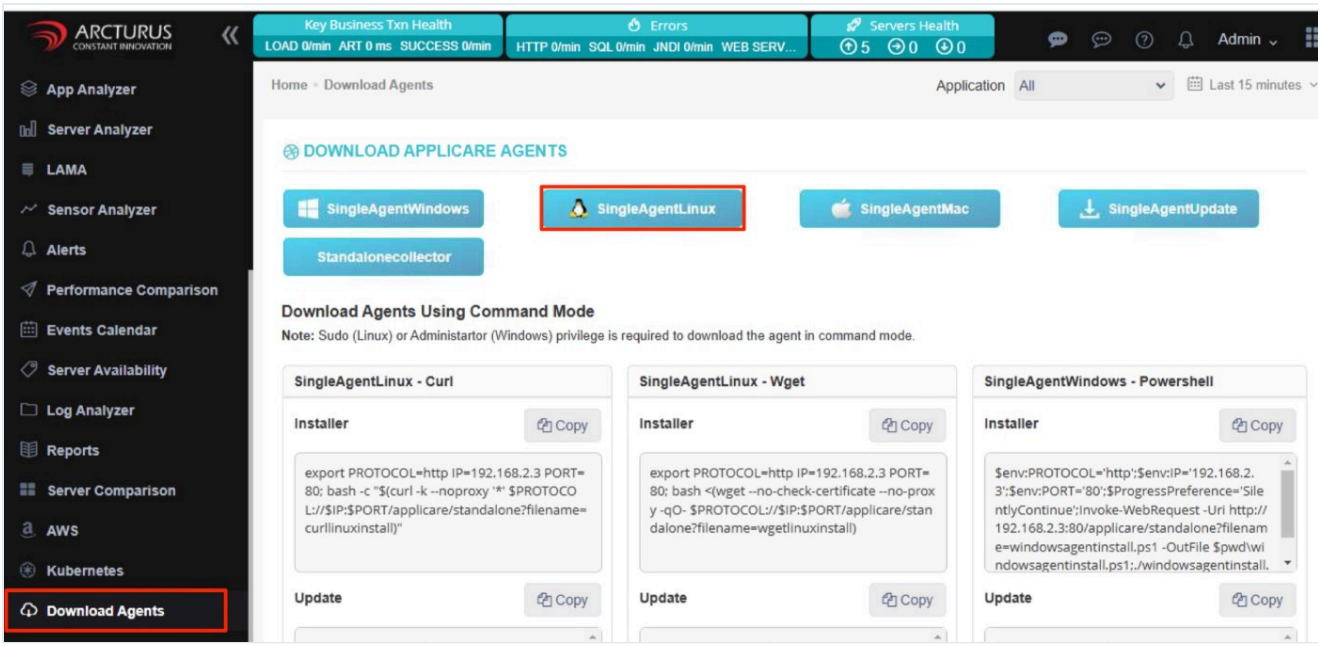
```
TEXT
PROTOCOL="<Applicare_protocol>"
IP="<Applicare_IP>"
PORT="<Applicare_port>"
```

<Applicare_protocol>= The protocol used by the Applicare controller is http or https.

<Applicare_IP>**= IP of the machine running Applicare controller**

<Applicare_port>= Applicare controller running port

Step 5: Access the **Applicare console**. Go to the **Download Agent** screen in the left-side menu, Download the **SingleAgentLinux** agent and extract it into the designated directory. **ApplicareSingleAgentLinux.**



Step 6: Add the following command along with the command that starts the container.

Add the following command before the image name.

```
TEXT
-v where the agent_deploy.sh file is located/agent_deploy.sh:/usr/local/agent_deploy.sh
-v ApplicareSingleAgentLinuxDirectory:/usr/local/ApplicareSingleAgentLinux
--entrypoint sh
--hostname The name which we need to add the agent in Applicare
```

Add the following command after the image name.

```
TEXT
-c 'rm /var/log/nginx/access.log && touch /var/log/nginx/access.log && openssl dhparam -out
/etc/nginx/dhparam/dhparam.pem 2048  && chown nginx:nginx /var/log/nginx/access.log && chmod
+x /usr/local/agent_deploy.sh && /usr/local/agent_deploy.sh && nginx -g "daemon off;"'
```

Example:

```
BASH
docker run -d --name nginx -v /home/nginx/nginx.conf:/etc/nginx/nginx.conf:ro -v
/var/run/docker.sock:/tmp/docker.sock:ro  -p 81:80 -p 443:443 -v
/home/nginx/agent_deploy.sh:/usr/local/agent_deploy.sh -v
/root/ApplicareSingleAgentLinux:/usr/local/ApplicareSingleAgentLinux  --entrypoint sh --
hostname jpetstore jwilder/nginx-proxy -c 'rm /var/log/nginx/access.log && touch
/var/log/nginx/access.log && openssl dhparam -out /etc/nginx/dhparam/dhparam.pem 2048 &&
chown nginx:nginx /var/log/nginx/access.log && chmod +x /usr/local/agent_deploy.sh &&
/usr/local/agent_deploy.sh && nginx -g "daemon off;"'
```

Note:

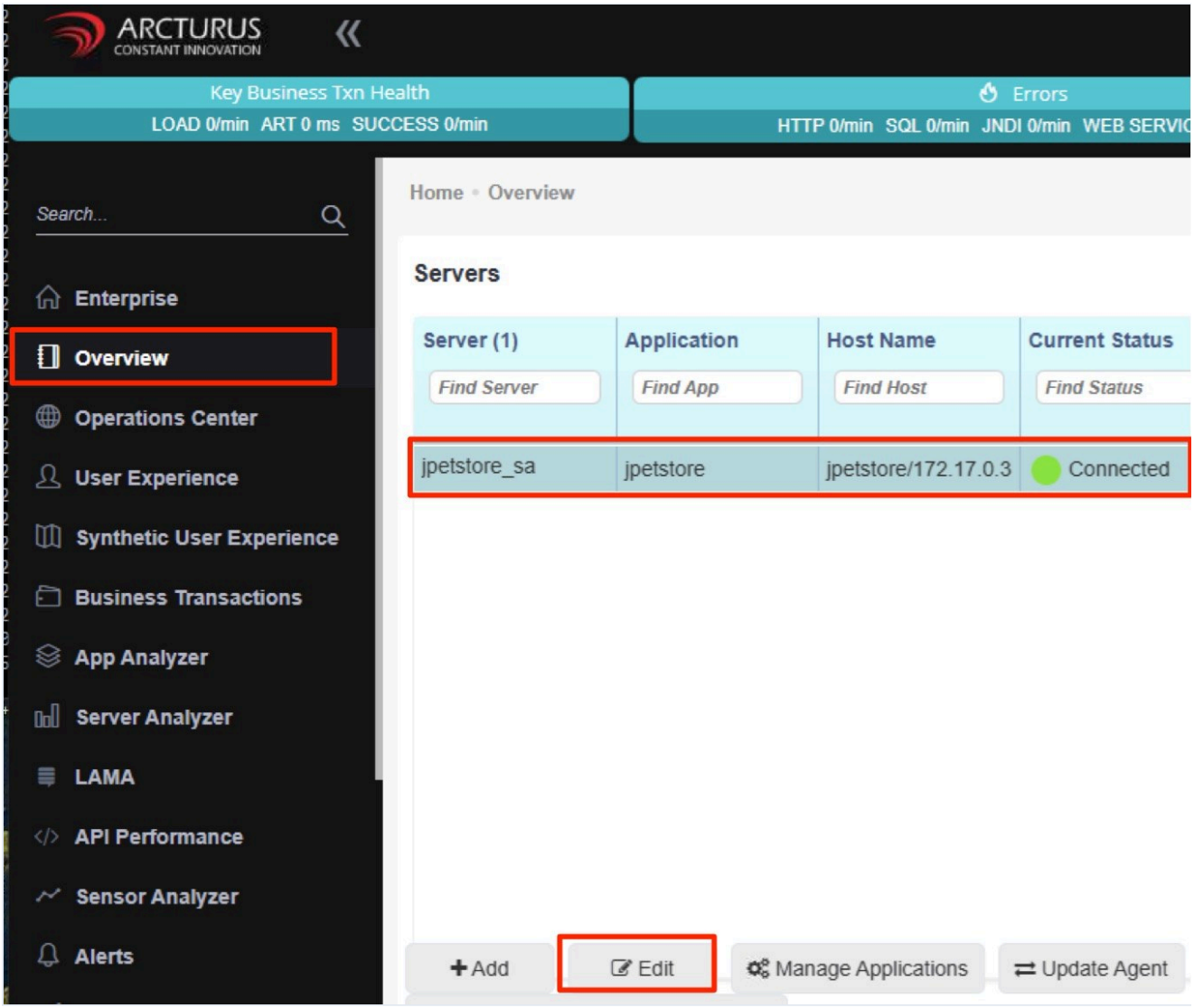
rm /var/log/nginx/access.log= Deletes the file or symbolic link at /var/log/nginx/access.log

touch /var/log/nginx/access.log= Creates a new, empty file at /var/log/nginx/access.log if it doesn't already exist.

openssl dhparam -out /etc/nginx/dhparam/dhparam.pem 2048= This command creates the DH parameters essential for SSL/TLS encryption, to enhancing secure communication and ensuring forward secrecy.

chown nginx:nginx /var/log/nginx/access.log= Changes the ownership of /var/log/nginx/access.log to the nginx user and group.

Step 7: Once the container starts, the agent will be automatically added to the Applicare controller. To manage it, log in to the Applicare console and go to the Overview menu. Find the newly added agent, select it, and click **Edit** to configure as needed.



Note: If the sub_status module is available, proceed with Step 8 as outlined. However, if the module is not available, skip Step 8 and proceed directly to Step 9.

Step 8: Enable the "**Add Nginx Configuration**" checkbox and add the below values.

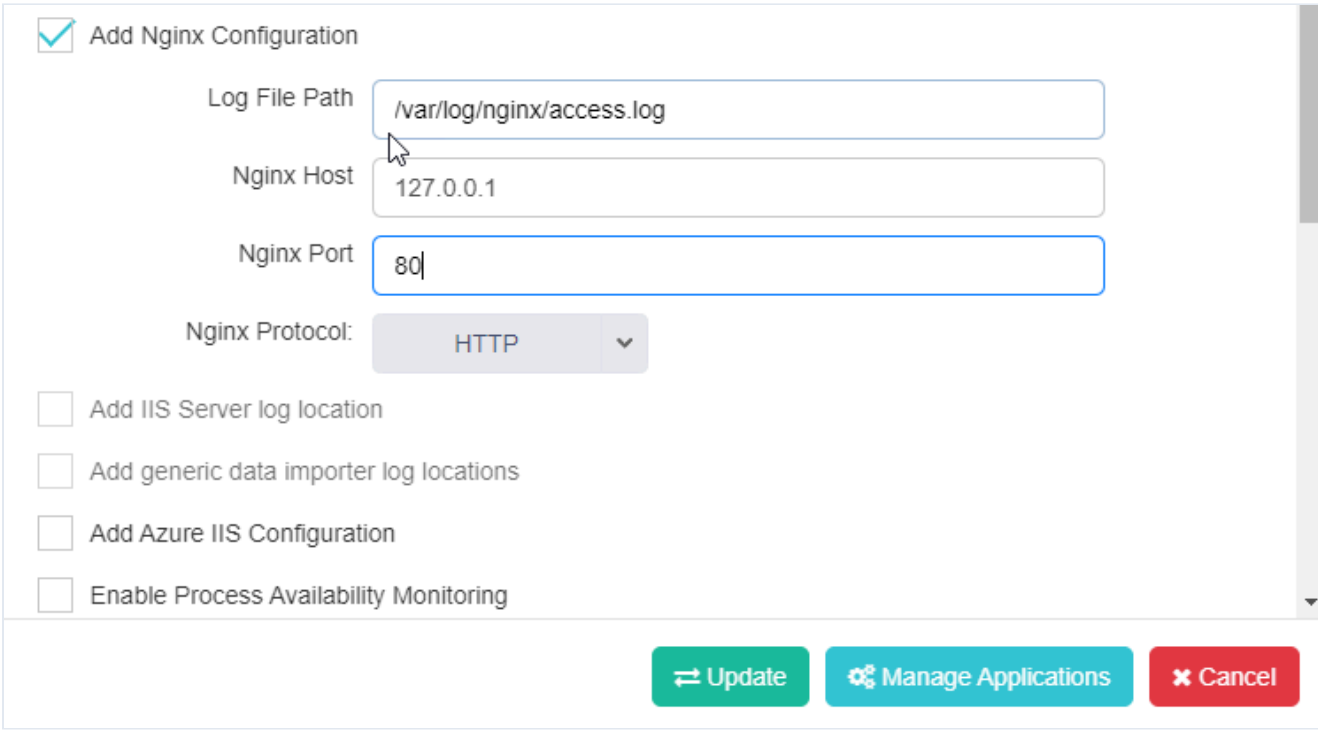
1. Log file path

2. Nginx Host

3. Nginx Port

4. Nginx Protocol

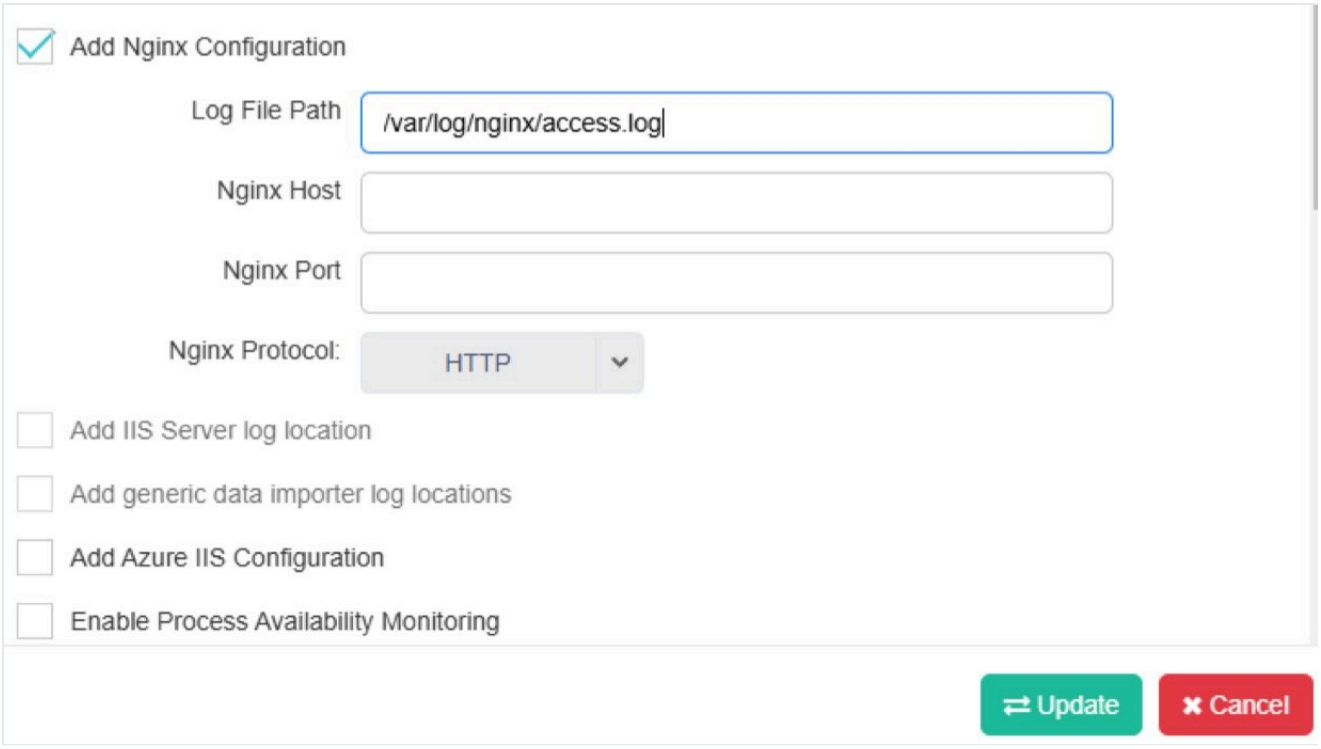
- Click the **update** button to save.



Step 9: Enable the "**Add Nginx Configuration**" checkbox and add the below values.

1. Log file path

- Click the **update** button to save.


Add Nginx Configuration
Log File Path
/var/log/nginx/access.log
Nginx Host
Nginx Port
Nginx Protocol:
HTTP
Add IIS Server log location
Add generic data importer log locations
Add Azure IIS Configuration
Enable Process Availability Monitoring
Update
Cancel

CONFIGURATION GUIDES › KUBERNETES

.NET monitoring in Kubernetes

Applicare 9.2 · Updated July 18, 2025 · dotnet, kubernetes · **Pending technical review**

Before you begin .NET monitoring installation, ensure the **Applicare Collector** is already installed and actively running on your **Applicare Controller** machine. For detailed instructions on setting up the collector, please refer to our guide: Applicare Collector Configuration on Linux/Ubuntu

To install the operator in an existing cluster, install cert-manager, if it is not already installed.

```bash
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.18.2/cert-manager.yaml
```

Install Opentelemetry operator in your Kubernetes cluster.

```bash
kubectl apply -f https://github.com/open-telemetry/opentelemetry-operator/releases/latest/download/opentelemetry-operator.yaml
```

Create a file dotnet-instrumentation.yaml and add the below content and save it.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: applicareotel-ns
  labels:
    name: applicareotel-ns

---

apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: dotnet-instrumentation
  namespace: applicareotel-ns
spec:
  exporter:
    endpoint: http://<APPLICARE_CONTROLLER_IP>:4318
  propagators:
    - tracecontext
    - baggage
  dotnet:
    image: ghcr.io/open-telemetry/opentelemetry-operator/autoinstrumentation-dotnet:1.9.0
```

Before deploying the above Instrumentation, ensure you replace the following placeholders:

`<APPLICARE_CONTROLLER_IP>` : Replace this with the actual IP address or hostname of your Applicare controller instance

After creating the yaml file, execute the command below.

```bash
kubectl apply -f dotnet-instrumentation.yaml
```

Add the below annotation in your application and **redeploy the application**.

```text
instrumentation.opentelemetry.io/inject-dotnet: "applicareotel-ns/applicaredotnet-instrumentation"
```

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: applicareapp
  namespace: applicare-ns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: applicareapp
  template:
    metadata:
      labels:
        app: applicareapp
      annotations:
        instrumentation.opentelemetry.io/inject-dotnet: "applicareotel-ns/applicaredotnet-instrumentation"
    spec:
      containers:
        - name: webapplication1
          image: applicare910/applicareapp:dotnetcore
          ports:
            - containerPort: 8080
```

CONFIGURATION GUIDES › KUBERNETES

Install OTel agent in Kubernetes cluster

Applicare 9.2 · Updated September 10, 2025 · kubernetes, opentelemetry · **Pending technical review**

Step 1: Install the OTel log agent on the Kubernetes cluster. Download the yaml file and replace **<Applicare_Ip>** on line **53** of the yaml file with the IP of the machine running the Applicare controller.

If have Wget, execute the command below.

```bash
sudo wget
https://applicare.blob.core.windows.net/demoapplication/applicare_log_collector.yaml
```

```text
**(OR)**
```

If have Curl, execute the command below.

```bash
sudo curl -O
https://applicare.blob.core.windows.net/demoapplication/applicare_log_collector.yaml
```

Then apply that yaml file to install collector in Kubernetes cluster.

```bash
kubectl apply -f applicare_log_collector.yaml
```

CONFIGURATION GUIDES › KUBERNETES

Local Kubernetes Monitoring

Applicare 9.2 · Updated March 24, 2026 · kubernetes · **Pending technical review**

Step 1: Run **kubectl top nodes** in your Kubernetes environment. If you do not see the details as expected, proceed to **Step 2** if the information is displayed correctly, move on to **Step 3**.

```
C:\Windows\system32>kubectl top nodes
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   181m         9%     1601Mi          41%
```

Step 2: Install the metrics server in your Kubernetes cluster. Execute the following command to deploy the metrics pod.

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Check if the metrics server is running by executing the following command.

kubectl get pods -n kube-system

```
E:\kubernetes>kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS        AGE
coredns-6d4b75cb6d-l7n9m           1/1     Running   0               6d21h
etcd-minikube                      1/1     Running   0               6d21h
kube-apiserver-minikube            1/1     Running   0               6d21h
kube-controller-manager-minikube   1/1     Running   1 (6d21h ago)   6d21h
kube-proxy-kmsnd                   1/1     Running   0               6d21h
kube-scheduler-minikube            1/1     Running   0               6d21h
metrics-server-658867cdb7-m46jm    1/1     Running   0               6d21h
storage-provisioner                1/1     Running   3 (2d4h ago)    6d21h

E:\kubernetes>
```

If the metrics server is running, proceed to step 3. If it is not running, follow the steps below

1: Edit the configuration file by running the following command.

```bash
kubectl edit deployments.apps -n kube-system metrics-server
```

2: Add the following lines under in the **spec: container: args:** in the configuration file and save your changes.

```text
- --kubelet-insecure-tls
```

```
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: k8s.gcr.io/metrics-server/metrics-server:v0.6.1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
```

3: After a few minutes, run the command **kubectl get pods -n kube-system** to check if the metrics server is running. If it is, running skip the following step.

4: Edit the configuration file by running the command **kubectl edit deployments.apps -n kube-system metrics-server** update the lines under **spec: container: args:** and save your changes.

```
TEXT
- --kubelet-preferred-address-types=InternalIP
```

```
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: k8s.gcr.io/metrics-server/metrics-server:v0.6.1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
```

5: To verify that the metrics API server is running, execute the following command and review the output:

kubectl top nodes

```
C:\Windows\system32>kubectl top nodes
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   181m         9%     1601Mi          41%
```

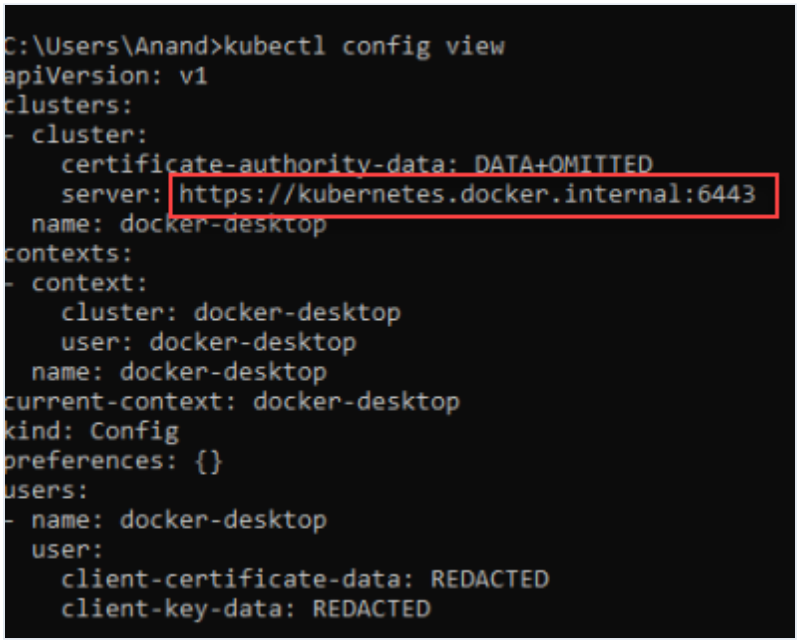
SingleAgent should be deployed in the machine where Kubernetes cluster is running.

Step 3: Refer here to **install SingleAgent and start.**

Once the agent is started, it will automatically be added to the Applicare Console.

Step 4. Run the following command and record the **cluster server** details for configuration.

```bash
kubectl config view
```



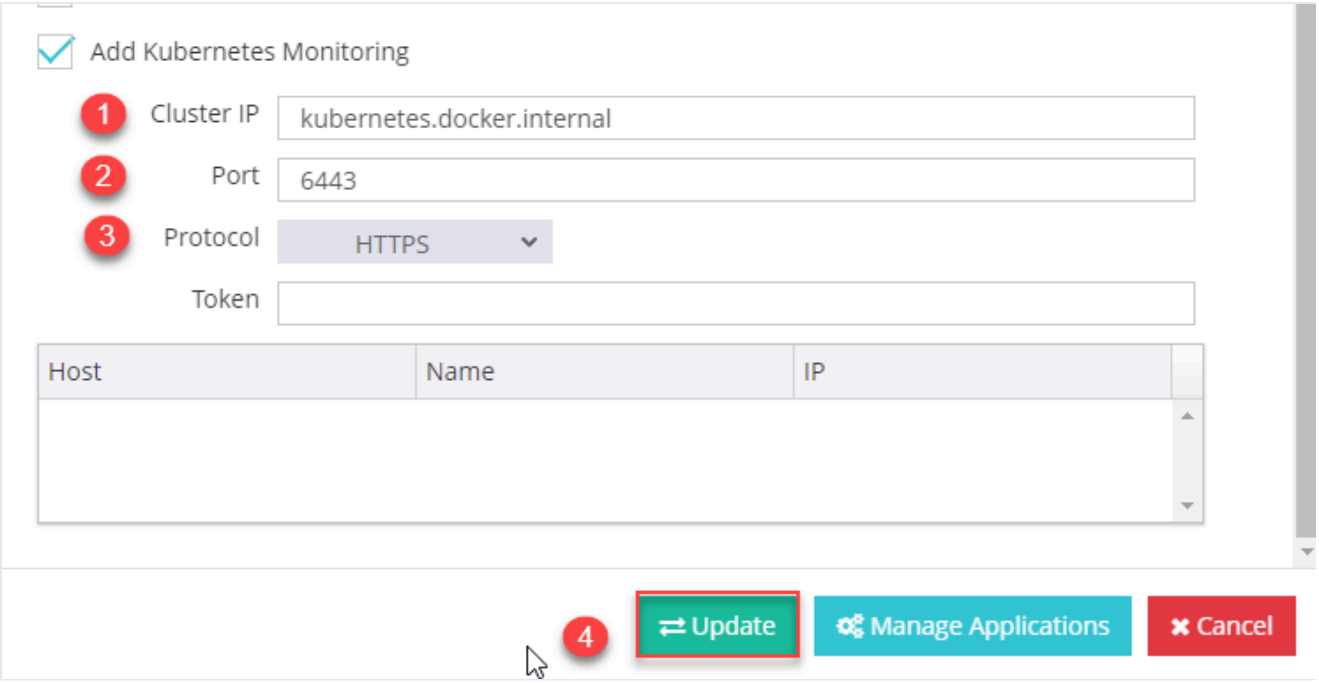
Cluster IP : kubernetes.docker.internal

Port : 6443

Protocol : https

Step 5: Log in to Applicare Console and navigate to the **Overview screen,** select the newly added server and click **Edit** button.

Step 6: Enable the **Add Kubernetes Monitoring checkbox** and enter the configuration details. Leave the **Token** field empty and click the **update** button.

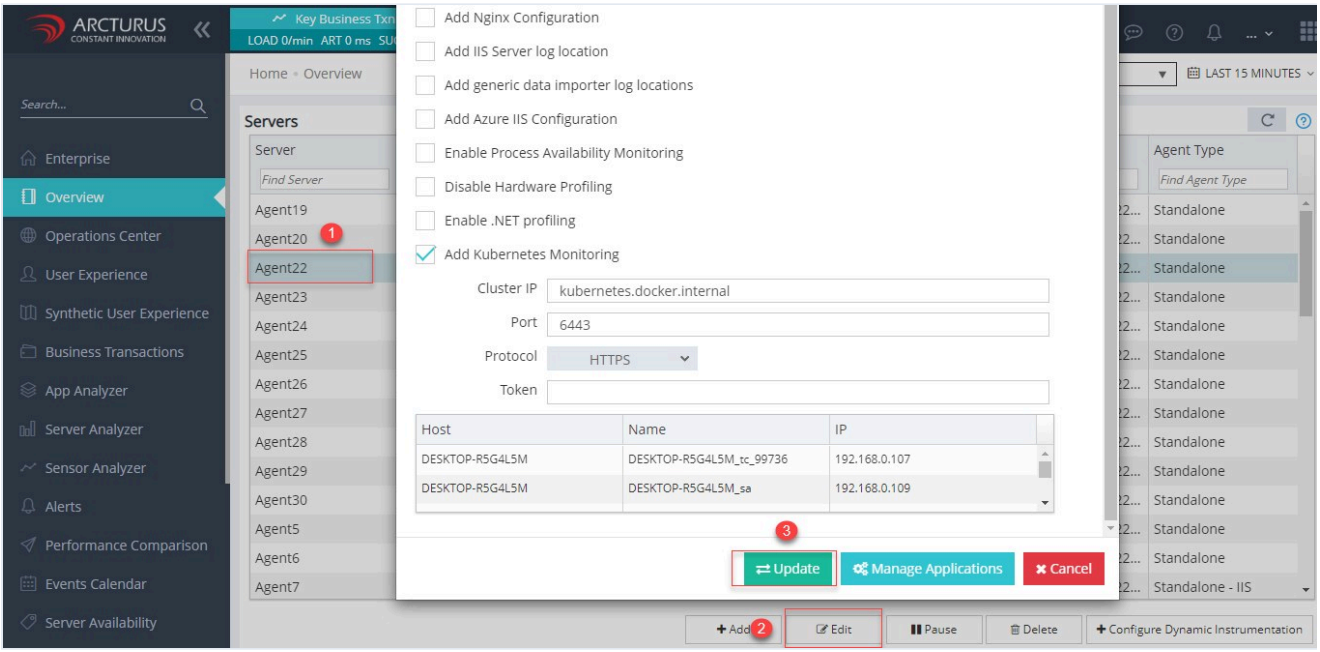


We can edit or delete the kubernetes configuration from both the **Overview** or **Kubernetes** menu.

Overview Menu

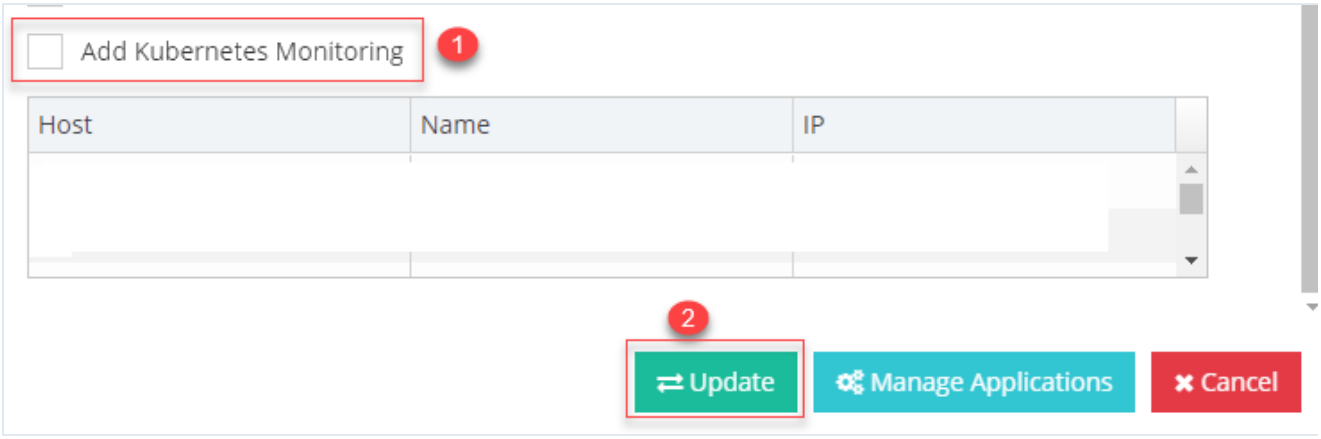
Update

Select the **row** and click **Edit** button and update the details and click **Update** button.



Delete

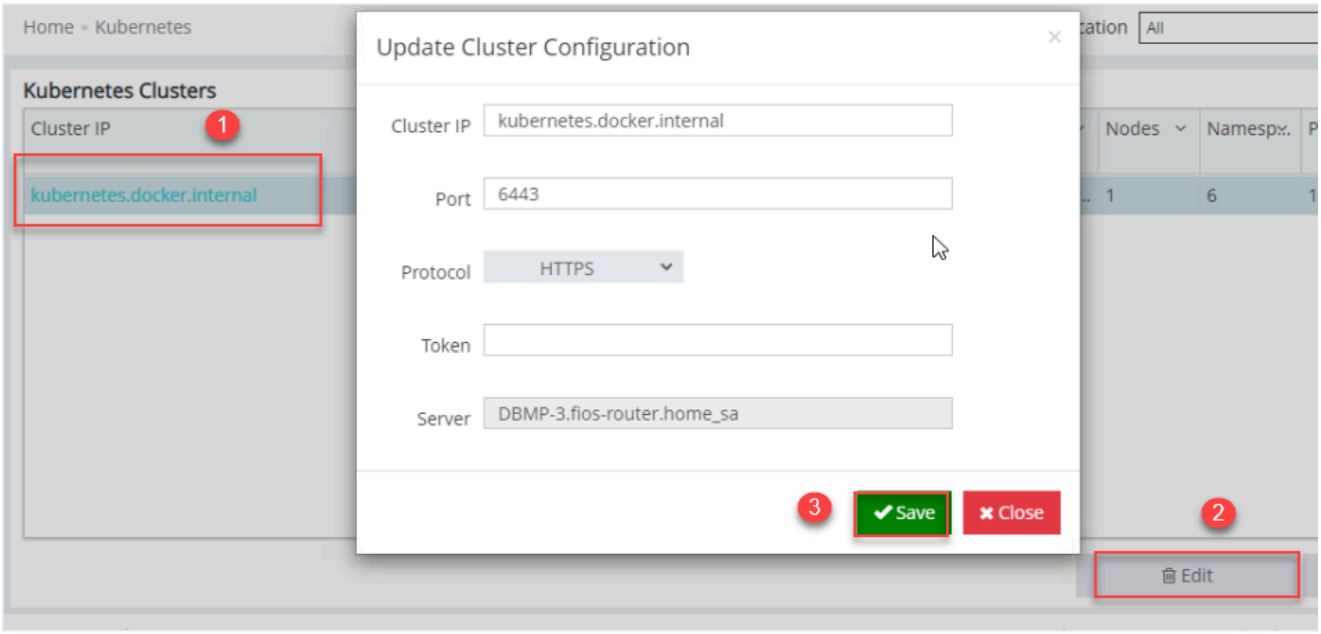
Select the **row** and click **Edit** button and uncheck the **Add the Kubernetes Monitoring** checkbox and click **Update** button.



Kubernetes Menu

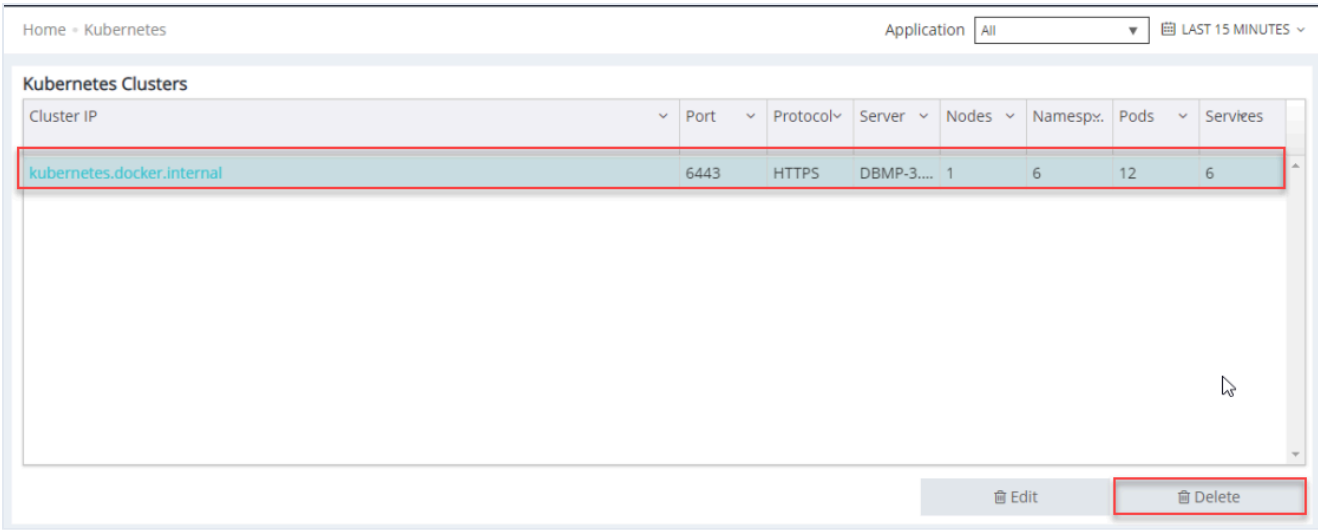
Update

Select the **row** and click **Edit** button and update the details and click **Save** button.



Delete

Select the **row** and click **Delete** button.


Home • Kubernetes
Application
All
LAST 15 MINUTES
Kubernetes Clusters
Cluster IP
Port
Protocol
Server
Nodes
Namespx.
Pods
Services
kubernetes.docker.internal
6443
HTTPS
DBMP-3....
1
6
12
6
Edit
Delete

CONFIGURATION GUIDES › KUBERNETES

Remote Kubernetes Monitoring

Applicare 9.2 · Updated June 20, 2026 · kubernetes · **Pending technical review**

1. Create an **applicare-setup.yml** file with the below content:

YAML

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: applicare
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: applicare
  namespace: applicare
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: applicareapi-cluster-role
rules:
  # Core inventory + storage objects
  - apiGroups: [""]
    resources:
      - nodes
      - namespaces
      - pods
      - services
      - events
      - endpoints
      - configmaps
      - persistentvolumeclaims
      - persistentvolumes
      - replicationcontrollers
    verbs: ["get", "list", "watch"]
  # Container log reads (fixes "Failed to get log details ... 403")
  - apiGroups: [""]
    resources:
      - pods/log
    verbs: ["get"]
  # Kubelet Summary API + cAdvisor via the API-server proxy:
  #   GET /api/v1/nodes/<node>/proxy/stats/summary     -> pod disk (ephemeral + PVC) and
network
  #   GET /api/v1/nodes/<node>/proxy/metrics/cadvisor  -> CPU throttling
  # nodes/proxy is a distinct RBAC subresource; granting "nodes" above does NOT cover it.
  - apiGroups: [""]
    resources:
      - nodes/proxy
      - nodes/stats
```

```
      - nodes/metrics
    verbs: ["get"]
  # Resource quotas per namespace (near-limit alerting)
  - apiGroups: [""]
    resources:
      - resourcequotas
      - limitranges
    verbs: ["get", "list", "watch"]
  # Events via the newer events.k8s.io API group
  - apiGroups: ["events.k8s.io"]
    resources:
      - events
    verbs: ["get", "list", "watch"]
  # CPU / memory metrics (KubernetesNodesMetrics / ContainerMetrics jobs)
  - apiGroups: ["metrics.k8s.io"]
    resources:
      - nodes
      - pods
    verbs: ["get", "list"]
  # Workload objects (desired-vs-ready replica health)
  - apiGroups: ["apps"]
    resources:
      - deployments
      - replicasets
      - statefulsets
      - daemonsets
    verbs: ["get", "list", "watch"]
  # Batch workloads (Job / CronJob failure monitoring)
  - apiGroups: ["batch"]
    resources:
      - jobs
      - cronjobs
    verbs: ["get", "list", "watch"]
  # Ingress resources — topology Ingress→Service→Pod chain
  - apiGroups: ["networking.k8s.io"]
    resources:
      - ingresses
      - ingressclasses
    verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: applicareapi-cluster-role-binding
subjects:
  - kind: ServiceAccount
```

```
    name: applicare
    namespace: applicare
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: applicareapi-cluster-role
---
apiVersion: v1
kind: Secret
metadata:
  name: applicare-token
  namespace: applicare
  annotations:
    kubernetes.io/service-account.name: applicare
type: kubernetes.io/service-account-token
```

2. Execute the below command in the same directory where the file was created:

BASH
```
kubectl apply -f applicare-setup.yml
```

3. Execute **kubectl top nodes** in the kubernetes.

```
C:\Windows\system32>kubectl top nodes
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   181m         9%     1601Mi          41%
```

- ***If you are not able to view the below details, then:*** Install metrics pod in Kubernetes. Execute the below command to install metrics pod:
  - **kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml**
- Check if the metrics server is running by executing the below command.
  - **kubectl get pods -n kube-system**

```
E:\kubernetes>kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS        AGE
coredns-6d4b75cb6d-l7n9m           1/1     Running   0               6d21h
etcd-minikube                      1/1     Running   0               6d21h
kube-apiserver-minikube            1/1     Running   0               6d21h
kube-controller-manager-minikube   1/1     Running   1 (6d21h ago)   6d21h
kube-proxy-kmsnd                   1/1     Running   0               6d21h
kube-scheduler-minikube            1/1     Running   0               6d21h
metrics-server-658867cdb7-m46jm    1/1     Running   0               6d21h
storage-provisioner                1/1     Running   3 (2d4h ago)    6d21h

E:\kubernetes>
```

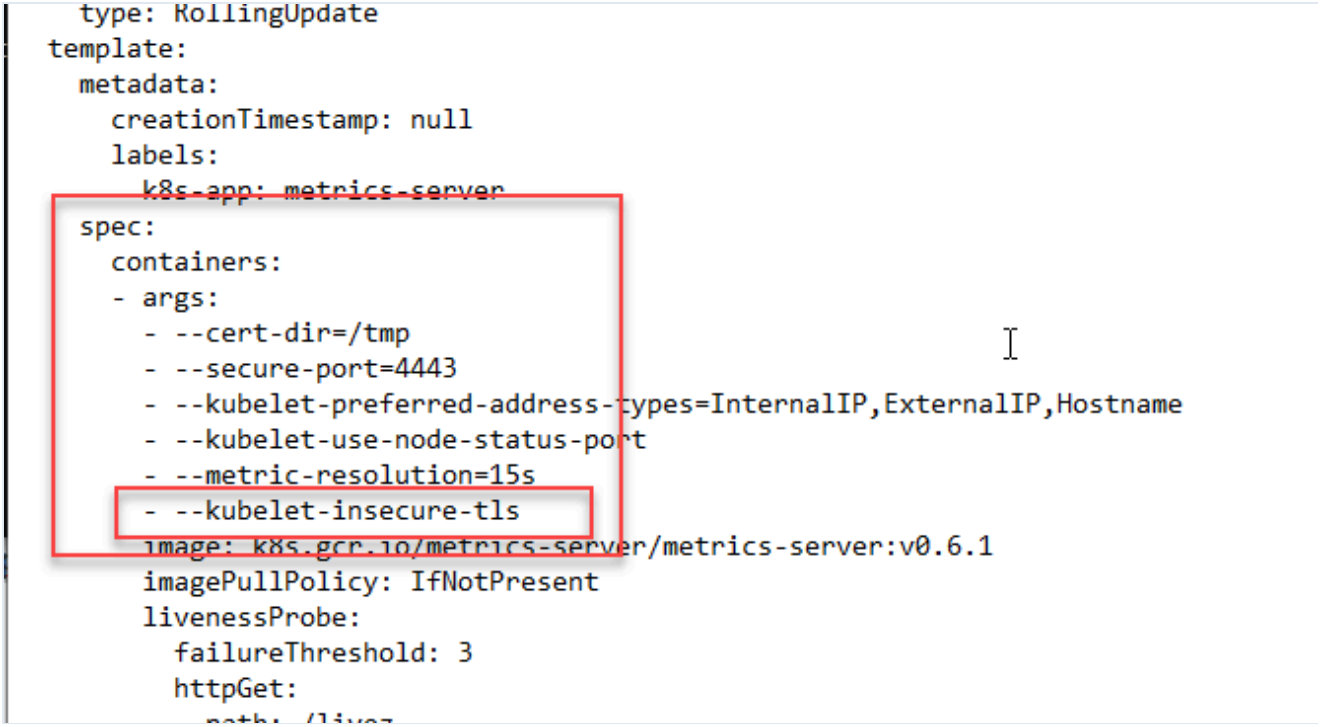
If it is still not running, then follow the below steps:

- **Step 1:** Edit the configuration file by executing the below command.

```bash
kubectl edit deployments.apps -n kube-system metrics-server
```

- **Step 2:** Add the below lines in the **spec: container: args:** and save the file.

```text
- --kubelet-insecure-tls
```



- **Step 3:** After a few minutes, execute the command **kubectl get pods -n kube-system** and check if the metrics server is running.
  - ***If it is not running, then***: Edit the configuration file by executing the command **kubectl edit deployments.apps -n kube-system metrics-server** modify the below lines in the **spec: container: args:** and save it.

```text
- --kubelet-preferred-address-types=InternalIP
```

```
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: k8s.gcr.io/metrics-server/metrics-server:v0.6.1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
```

- **Step 4:** To verify the metrics API server is running, execute the below command and check **kubectl top nodes**.

```
C:\Windows\system32>kubectl top nodes
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   181m         9%     1601Mi          41%
```

4. Refer here to **install SingleAgent and start.**

- Once we start the agent it will be added to Applicare Console.

5. Execute the below command one by one to get the **Cluster IP** and **Token**. Copy the **$APISERVER** and **$TOKEN** value for configuration.

To get APISERVER and TOKEN in PowerShell:

```
TEXT
$APISERVER = kubectl config view --minify -o jsonpath='{.clusters[0].cluster.server}'
Write-Host "APISERVER: $APISERVER"
$TOKEN = kubectl get secret applicare-token -n applicare -o jsonpath='{.data.token}' |
ForEach-Object {
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($_)) }
Write-Host "TOKEN: $TOKEN"
```

To get APISERVER and TOKEN in Bash

```bash
APISERVER=$(kubectl config view --minify -o jsonpath='{.clusters[0].cluster.server}')
echo "APISERVER: $APISERVER"
TOKEN=$(kubectl get secret applicare-token -n applicare -o jsonpath='{.data.token}' | base64 --decode)
echo "TOKEN: $TOKEN"
```

6. Login into the Applicare console and go to the **Overview screen**. Select the newly added server and click the **Edit** button.

7. **Enable** the **Kubernetes Integration checkbox** and add the configuration details.

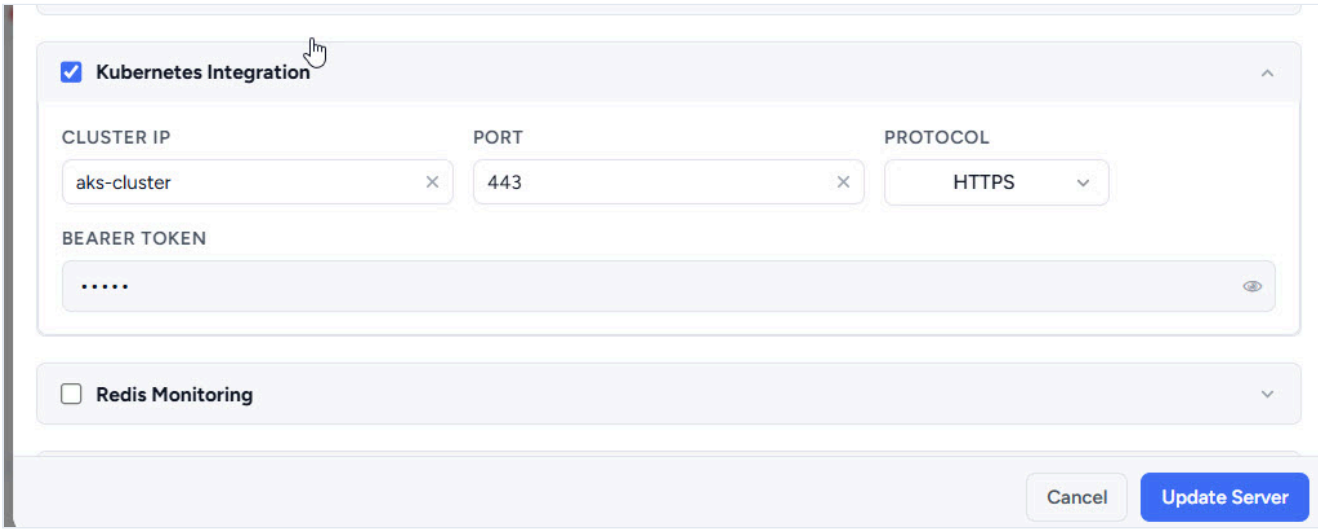
8. Remove the "**https://** (or) **http://"** and **port** in **APISERVER** value and Paste the remaining value in the **CLUSTER IP** field.

9. Copy the **TOKEN** value and paste it in the **BEARER TOKEN** field.

10. Enter the Kubernetes running port in the **PORT** field.

11. Select the Kubernetes protocol from the **PROTOCOL dropdown**. Protocol details will be available in the **APISERVER. "https:// (or) http://"**

12. Click the **Update Server** button.

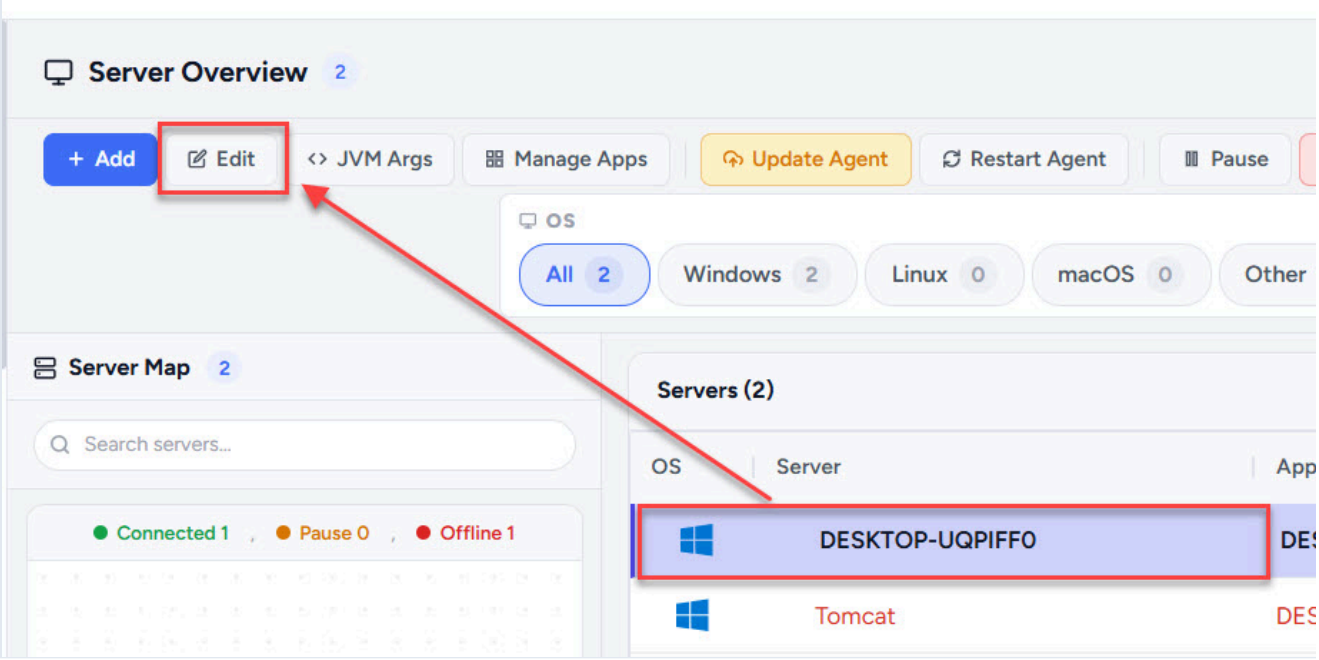


- **Note:** We can edit or delete the Kubernetes configuration from both the **Overview** or **Kubernetes** menu.

Overview Menu

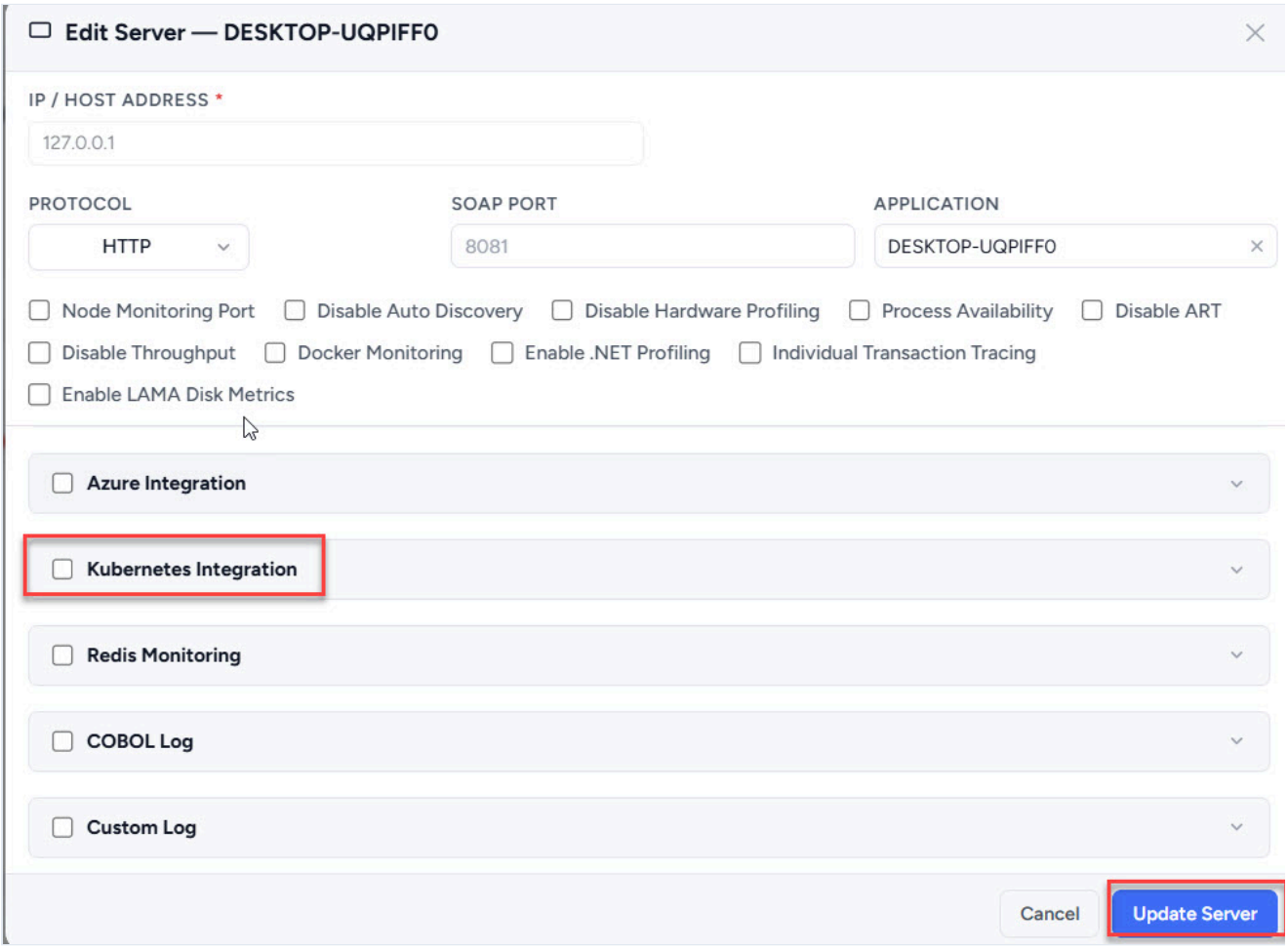
Update

- Select the **row** and click the **Edit** button to update details and then click the **Update** button.



Delete

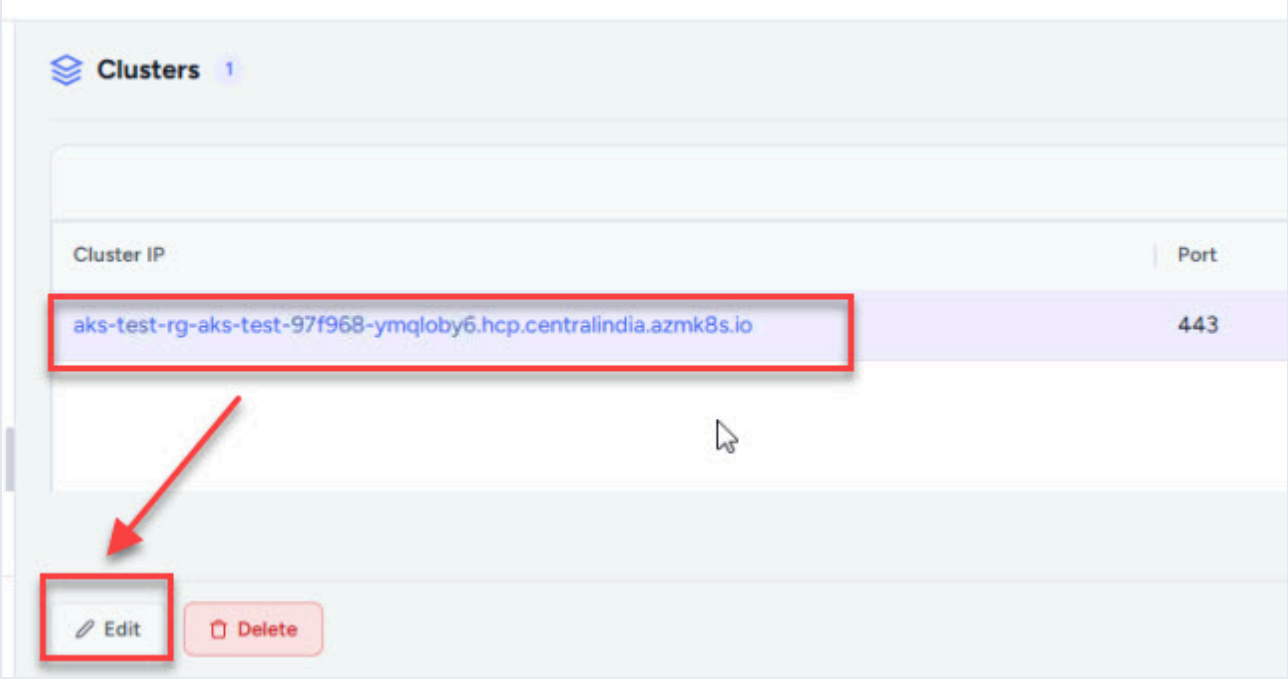
- Select the **row** and click the **Edit** button and **Uncheck** the **Add the Kubernetes Monitoring** checkbox. Then, click the **Update** button.



Kubernetes Menu

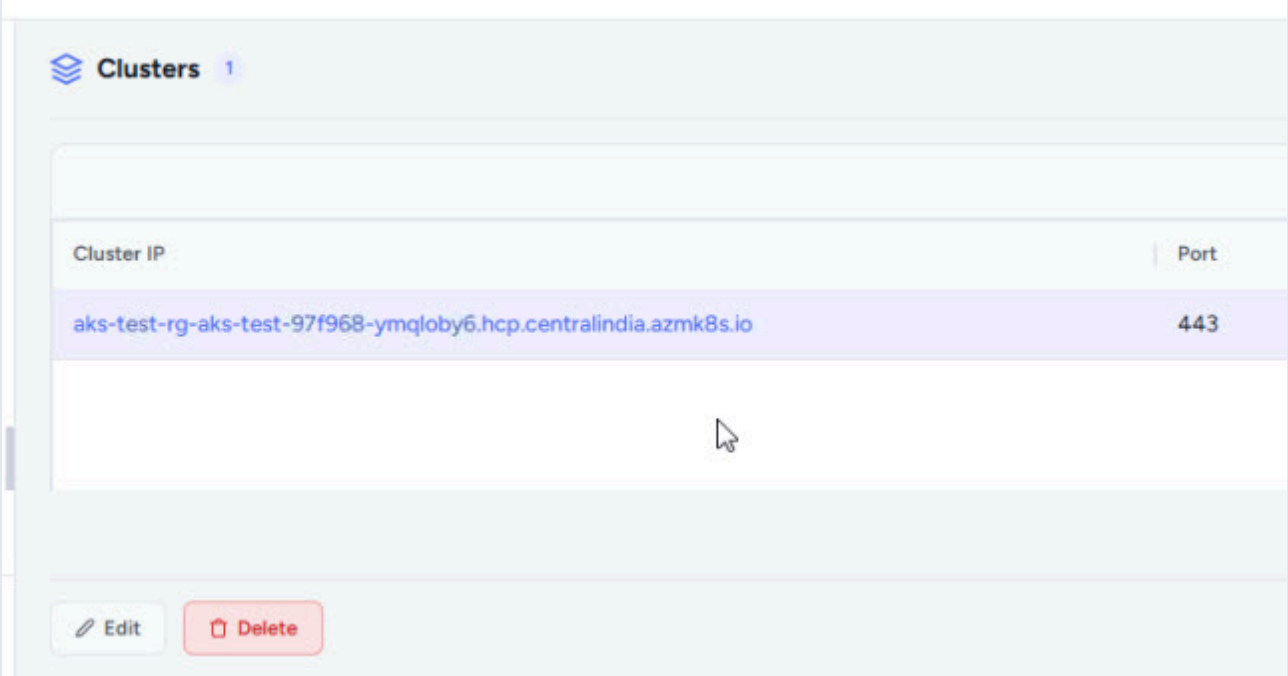
Update

- Select the **row** and click the **Edit** button to update details. Then, click the **Save** button.



Delete

- Select the **row** and click the **Delete** button.



Note: We can edit or delete the Kubernetes configuration from both the **Overview** or **Kubernetes** menu.

Uninstall the Applicare Kubernetes monitor service account and secrets.

```bash
kubectl delete namespace applicare
kubectl delete clusterrole applicareapi-cluster-role
kubectl delete clusterrolebinding applicareapi-cluster-role-binding
```

CONFIGURATION GUIDES › KUBERNETES

Installing Applicare SingleAgent in Kubernetes Containers Using a Script

Applicare 9.2 · Updated August 21, 2024 · kubernetes, singleagent · **Pending technical review**

Note: Follow the steps in this to install the agent on the docker image.

Add the following line to the end of your YAML file during container creation.

```yaml
env:
- name: sname
value: <SERVERNAME>
```

Replace **<SERVERNAME>** with the desired name for container in Applicare. The **<SERVERNAME>** should be **unique for each containers.**

e.g.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
namespace : tomcat-ns1
name: app-deployment-1
spec:
replicas: 1
selector:
matchLabels:
app: tomcat
template:
metadata:
labels:
app: tomcat
spec:
containers:
- name: tomcat-container1
image: petstore:latest
imagePullPolicy: Never
ports:
- containerPort: 8080
env:
- name: sname
value: petstore
```

Assign a unique name to each Kubernetes container.

e.g.

YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
namespace : tomcat-ns1
name: app-deployment-1
spec:
replicas: 1
selector:
matchLabels:
app: tomcat
template:
metadata:
labels:
app: tomcat
spec:
containers:
- name: tomcat-container1
image: petstore:latest
imagePullPolicy: Never
ports:
- containerPort: 8080
env:
- name: sname
value: petstore1
---
apiVersion: apps/v1
kind: Deployment
metadata:
namespace : tomcat-ns1
name: app-deployment-2
spec:
replicas: 1
selector:
matchLabels:
app: tomcat
template:
metadata:
labels:
app: tomcat
spec:
containers:
- name: tomcat-container1
image: petstore:latest
imagePullPolicy: Never
```

```
ports:
- containerPort: 8080
env:
- name: sname
value: petstore2

---
apiVersion: v1
kind: Service
metadata:
namespace : tomcat-ns
name: tomcat-service-1
spec:
selector:
app: tomcat
ports:
- protocol: TCP
port: 81
targetPort: 8080
type: LoadBalancer
```

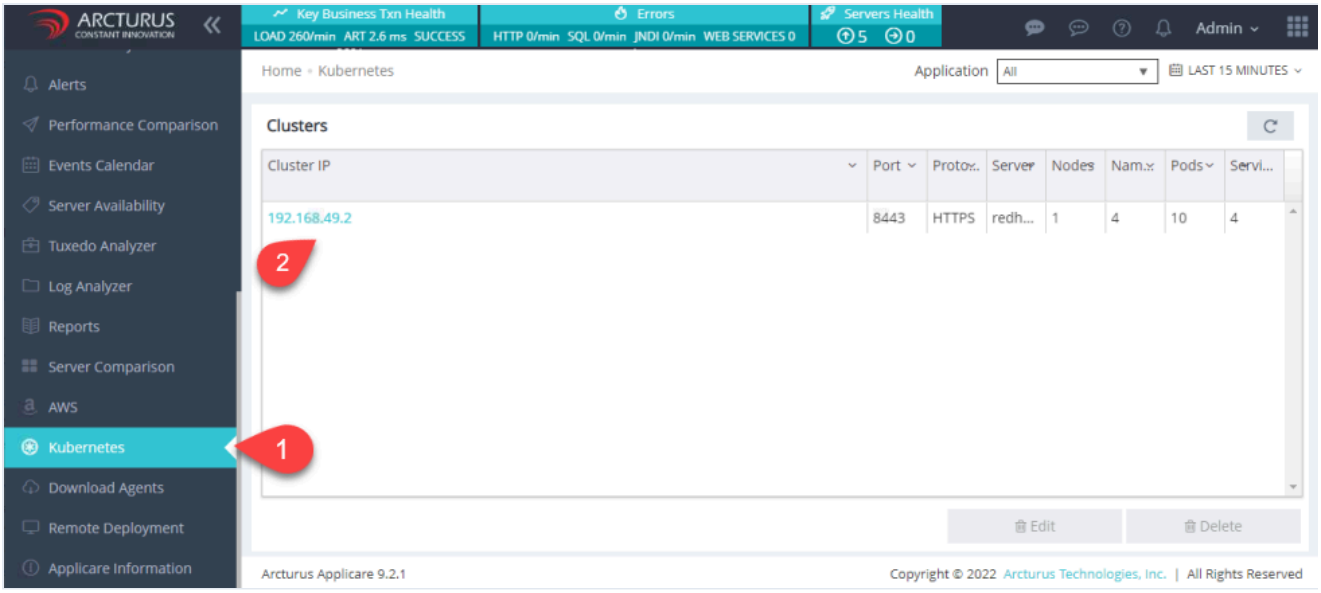
CONFIGURATION GUIDES › KUBERNETES

Kubernetes Monitoring

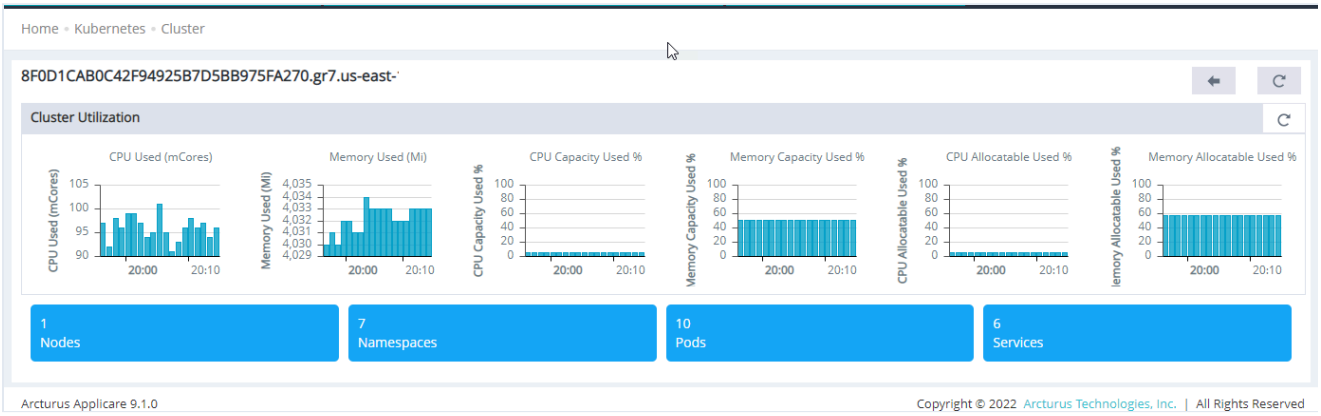
Applicare 9.2 · Updated June 11, 2026 · kubernetes · **Pending technical review**

Agent less K8S monitoring for Kubernetes

1. Go to the **Kubernetes** menu in the left side menu, click the **Cluster IP** to view the details.



2. Here you get the **Kubernetes Cluster Details.**



- **CPU Used** -

  Total CPU utilization across all nodes in the cluster, measured in milicores.

- **Memory Used** -

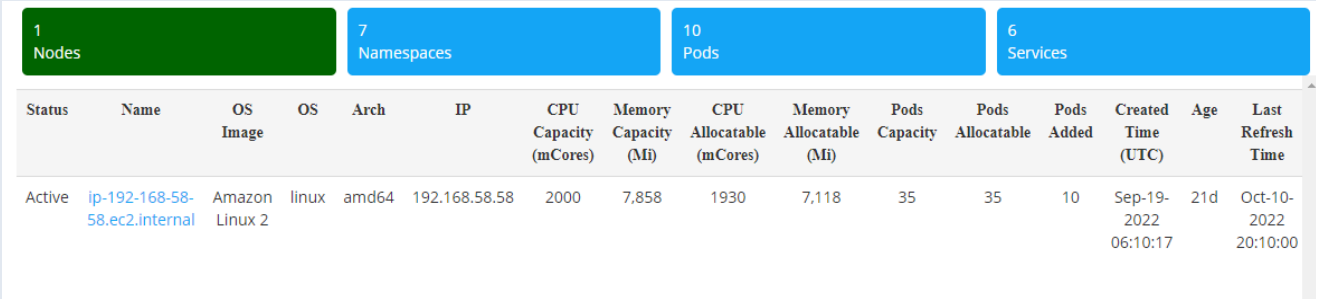
  Total memory utilization across all nodes in the cluster, measured in mebibytes (MiB).

  1. **CPU Capacity Used %** - Percentage of CPU utilized from its capacity value.
  2. **Memory Capacity Used %** - Percentage of memory utilized from its capacity value.

- **Allocatable** - It indicates the amount of resources on a Node that is available to be consumed by normal Pods.
- **Capacity -** Indicates the total amount of resources that a Node has
- **CPU Allocatable Used %** - Percentage of CPU utilized from its allocatable value.
- **Memory Allocatable Used %** - Percentage of memory utilized from its allocatable value.

3. Click the **Nodes, Namespaces, Pods and Services** to view the details.

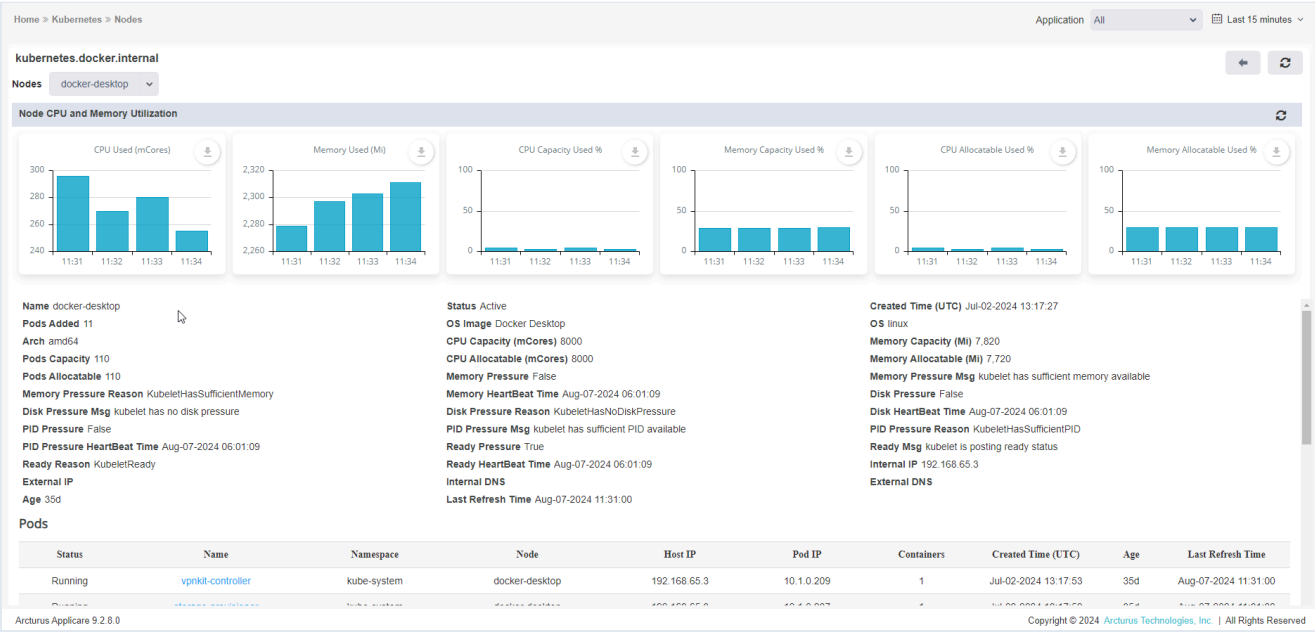
Nodes



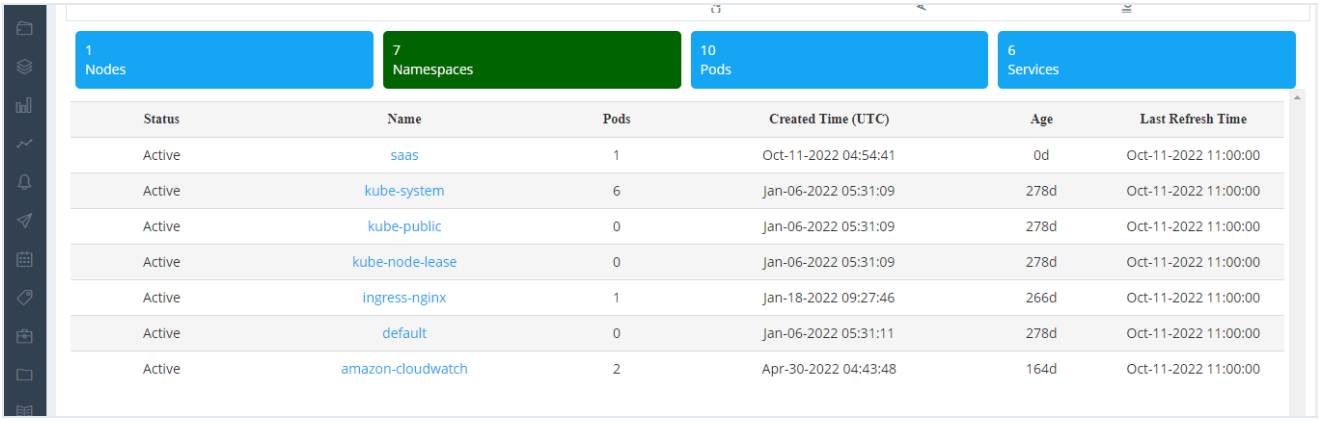
Status	Name	OS Image	OS	Arch	IP	CPU Capacity (mCores)	Memory Capacity (Mi)	CPU Allocatable (mCores)	Memory Allocatable (Mi)	Pods Capacity	Pods Allocatable	Pods Added	Created Time (UTC)	Age	Last Refresh Time
Active	ip-192-168-58-58.ec2.internal	Amazon Linux 2	linux	amd64	192.168.58.58	2000	7,858	1930	7,118	35	35	10	Sep-19-2022 06:10:17	21d	Oct-10-2022 20:10:00

4. Click the **Name** column to view the **node** details.

- **CPU Used** - Total CPU utilization for **individual node** in the cluster.
- **Memory Used** - Total memory utilization for **individual node** in the cluster.

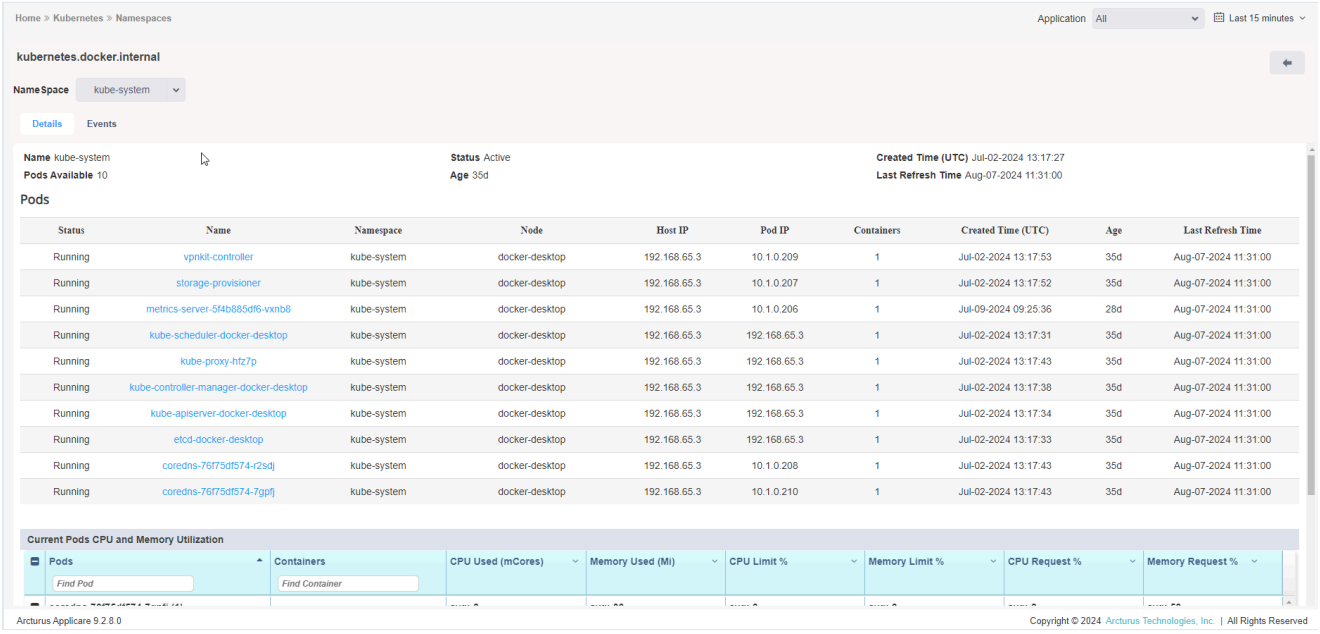


Namespaces

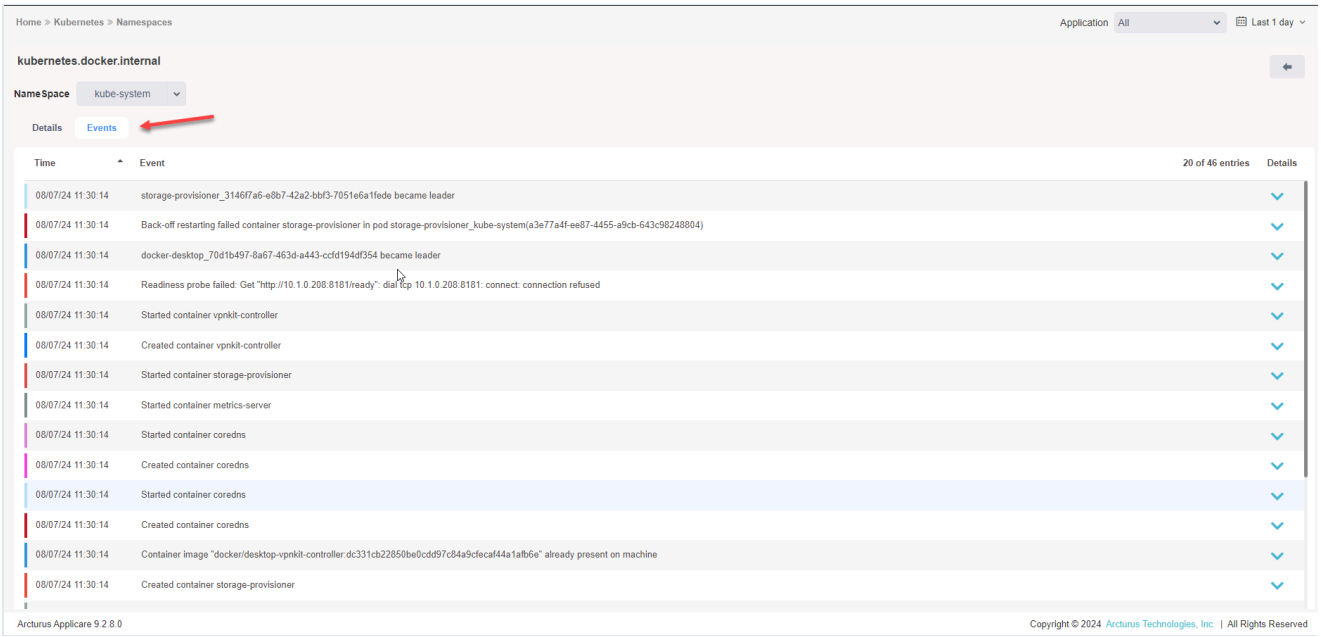


Status	Name	Pods	Created Time (UTC)	Age	Last Refresh Time
Active	saas	1	Oct-11-2022 04:54:41	0d	Oct-11-2022 11:00:00
Active	kube-system	6	Jan-06-2022 05:31:09	278d	Oct-11-2022 11:00:00
Active	kube-public	0	Jan-06-2022 05:31:09	278d	Oct-11-2022 11:00:00
Active	kube-node-lease	0	Jan-06-2022 05:31:09	278d	Oct-11-2022 11:00:00
Active	ingress-nginx	1	Jan-18-2022 09:27:46	266d	Oct-11-2022 11:00:00
Active	default	0	Jan-06-2022 05:31:11	278d	Oct-11-2022 11:00:00
Active	amazon-cloudwatch	2	Apr-30-2022 04:43:48	164d	Oct-11-2022 11:00:00

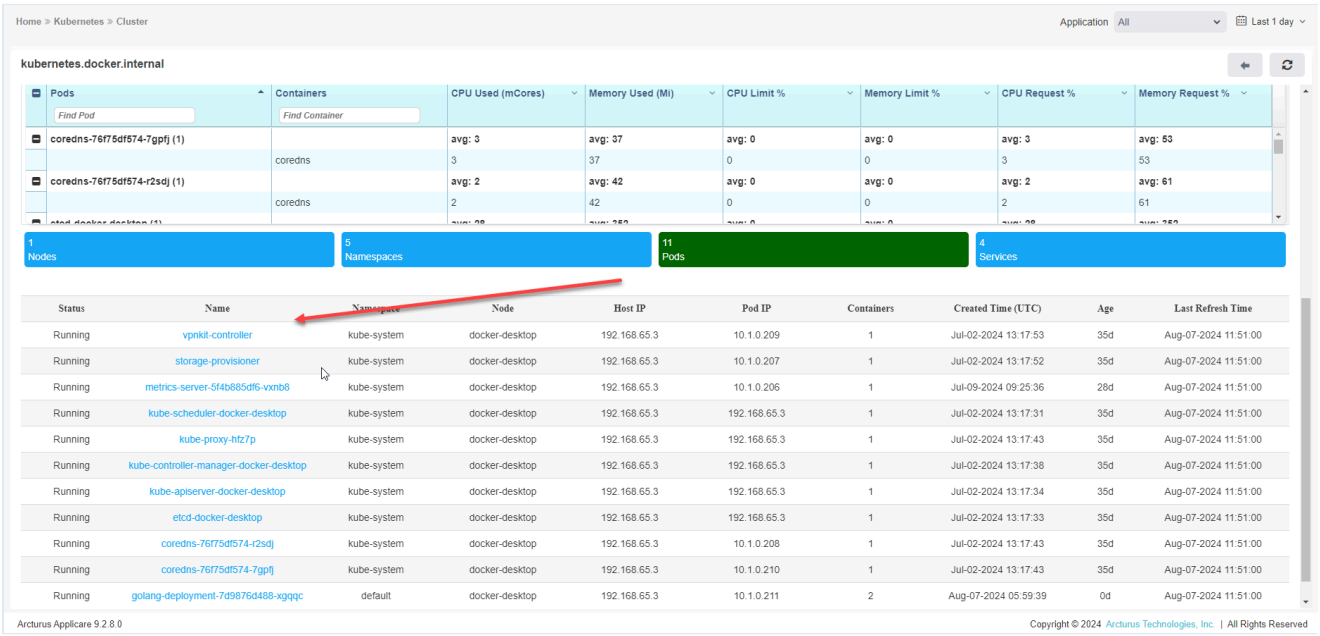
5. **Click** the **Name** column to view the **namespace** details.



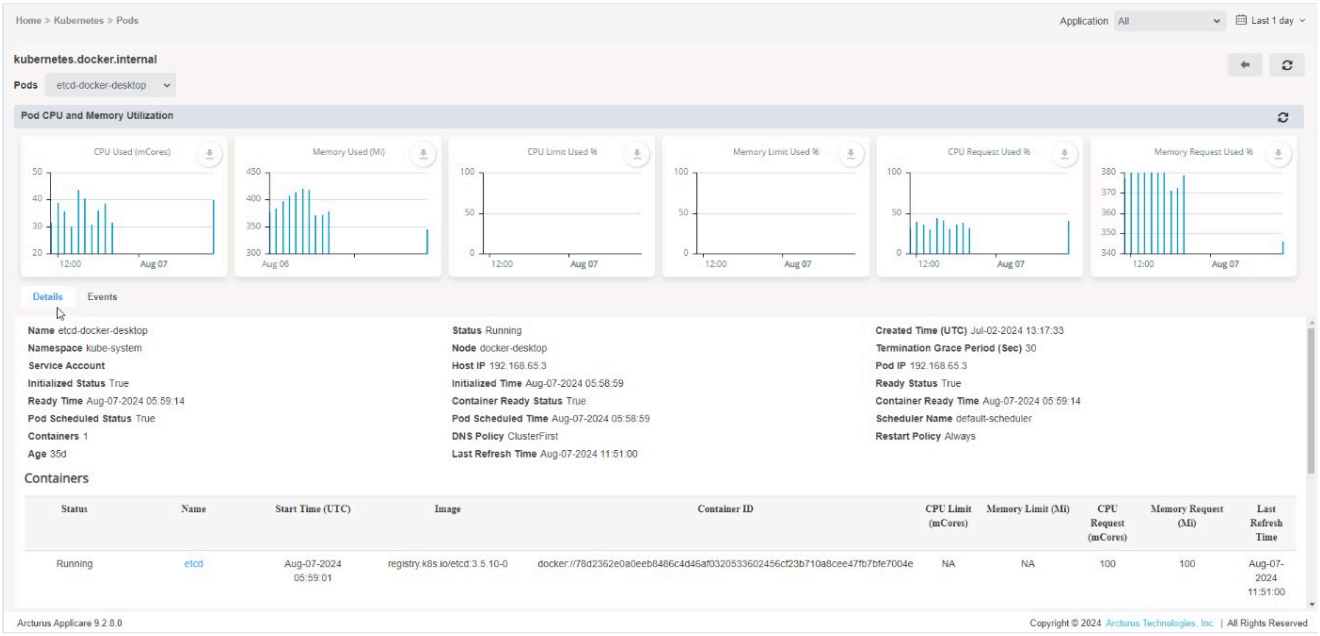
6. Click the **Events** tab to view the event details for the selected **namespace.**



Pods



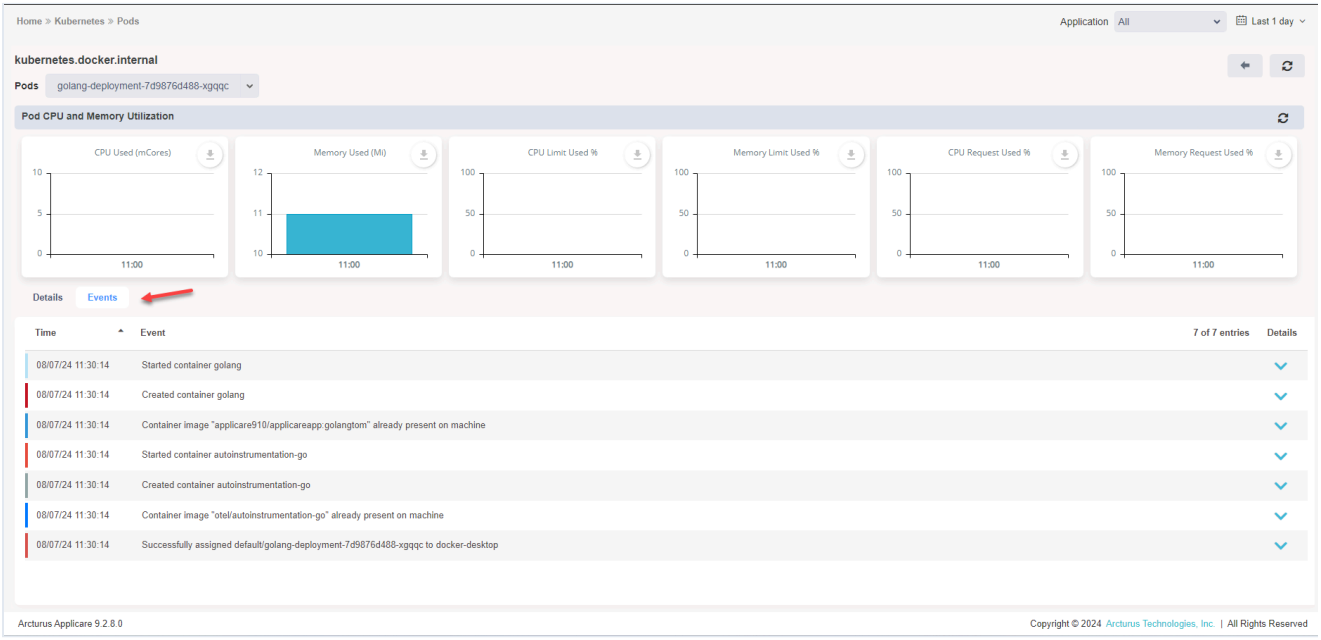
7. Click on the Pod in **Name** column to view the **pod** details.



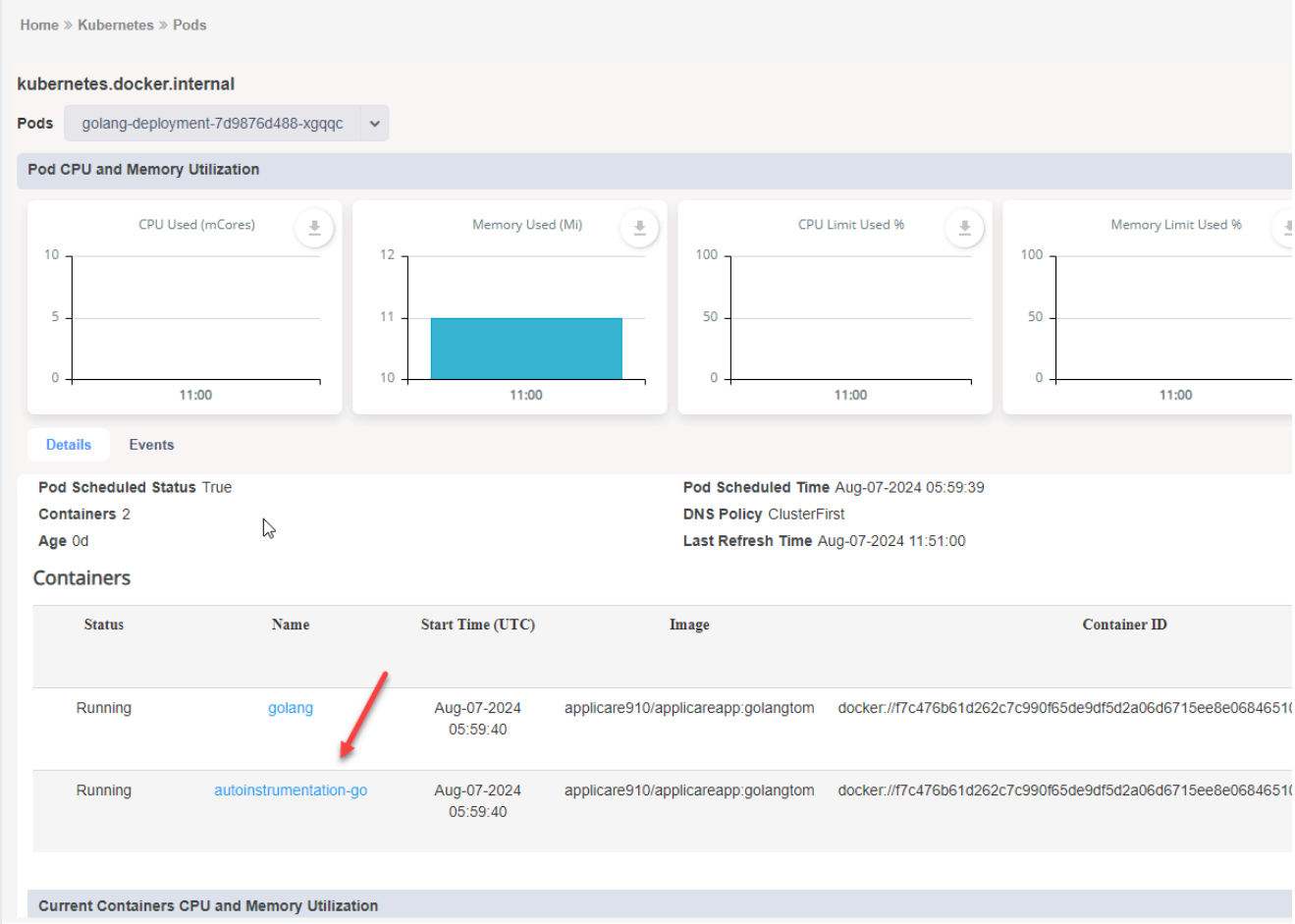
- **CPU Used** - Total CPU utilization of all the containers in the pods (**milicores**).
- **Memory Used** - Total memory utilization of all the containers in the pods (**mebibytes**).
- **CPULimit** - Defines a hard ceiling on how much CPU time that the container can use. During each scheduling interval (time slice), the Linux kernel checks to see if this limit is exceeded; if so, the kernel waits before allowing that cgroup to resume execution.
- **Memory Limit** - Defines a memory limit for that cgroup. If the container tries to allocate more memory than this limit, the Linux kernel out-of-memory subsystem activates and, typically, intervenes by stopping one of the processes in the container that tried to allocate memory. If that process is the container's PID 1, and the container is marked as restartable, Kubernetes restarts the container.

- **CPU request** - Typically defines a weighting. If several different containers (cgroups) want to run on a contended system, workloads with larger CPU requests are allocated more CPU time than workloads with small requests.
- **Memory request** - Mainly used during (Kubernetes) Pod scheduling.
- **CPU Limit Used %** - Percentage of CPU utilized from its cpu limit.
- **Memory Limit Used %** - Percentage of memory utilized from its memory limit.
- **CPU Request Used %** - Percentage of CPU utilized from its cpu request.
- **Memory Request Used %** - Percentage of memory utilized from its memory request.

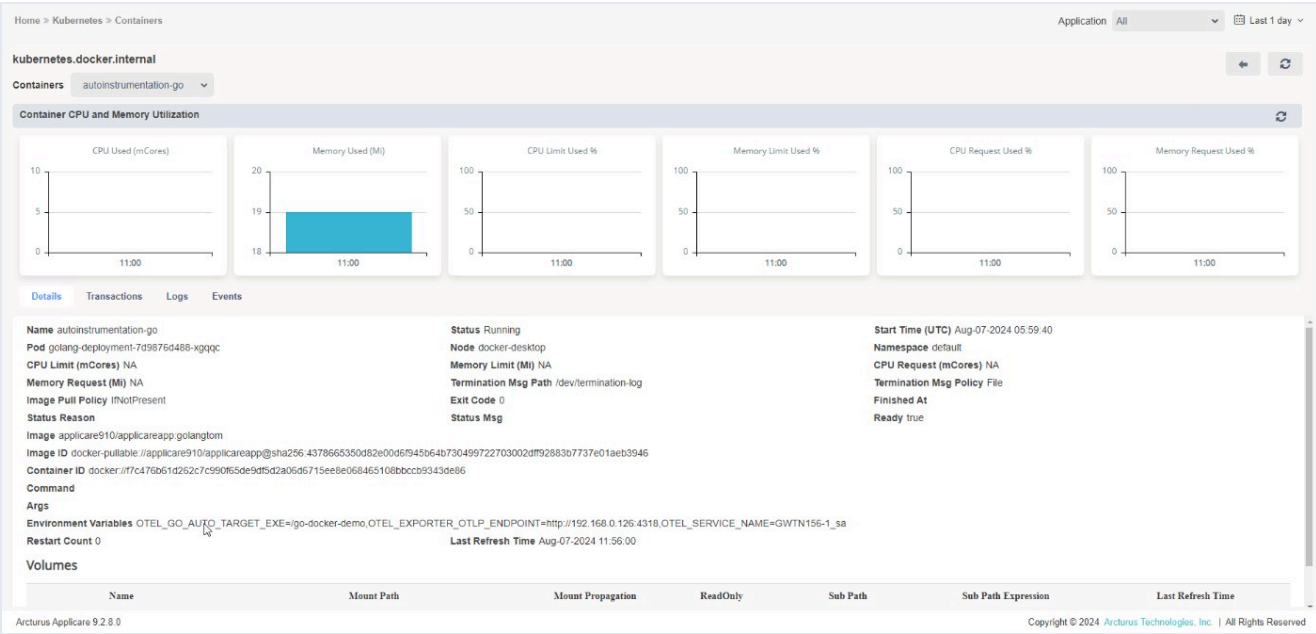
8. Click on the **Events** tab to view the **pod event details**.



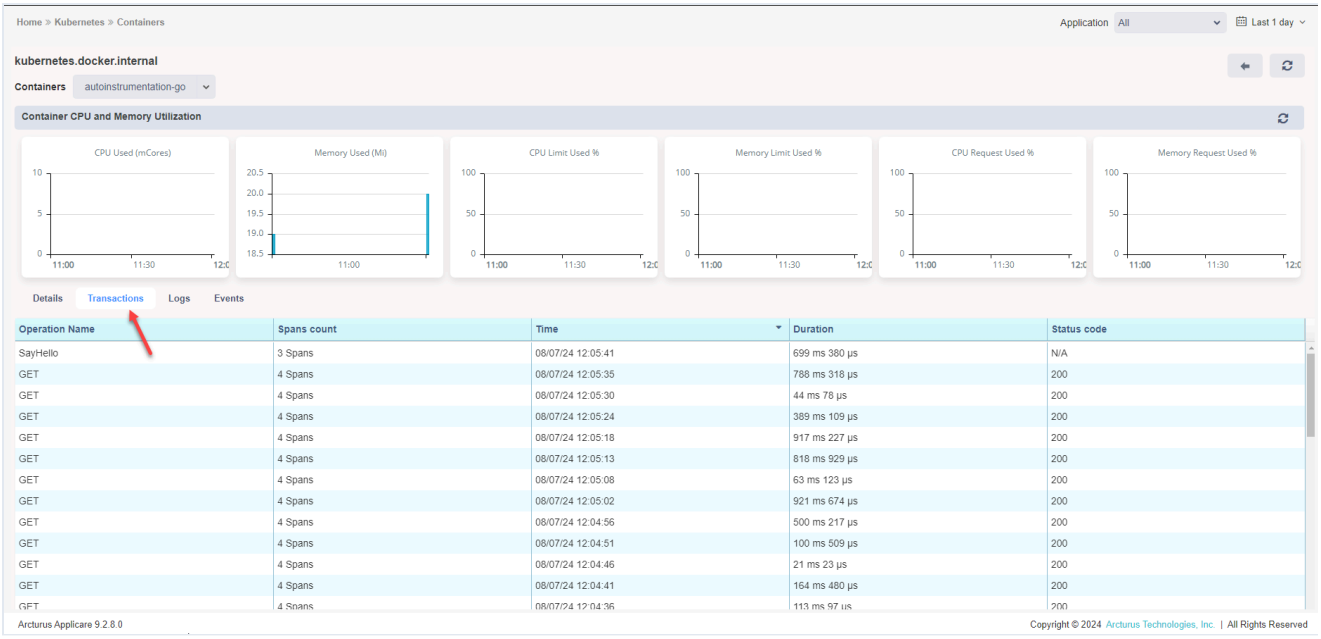
9. Click on the **Container** in the **Name** column to view the container **details**.



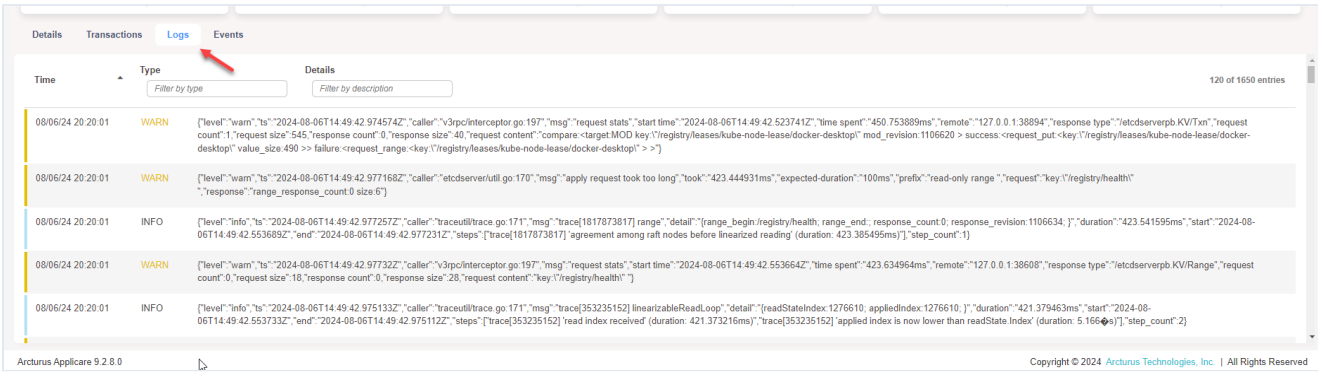
- **CPU Used** - Total CPU utilization for each **individual container** in the pod.
- **Memory Used** - Total memory utilization for each **individual container** in the pod.



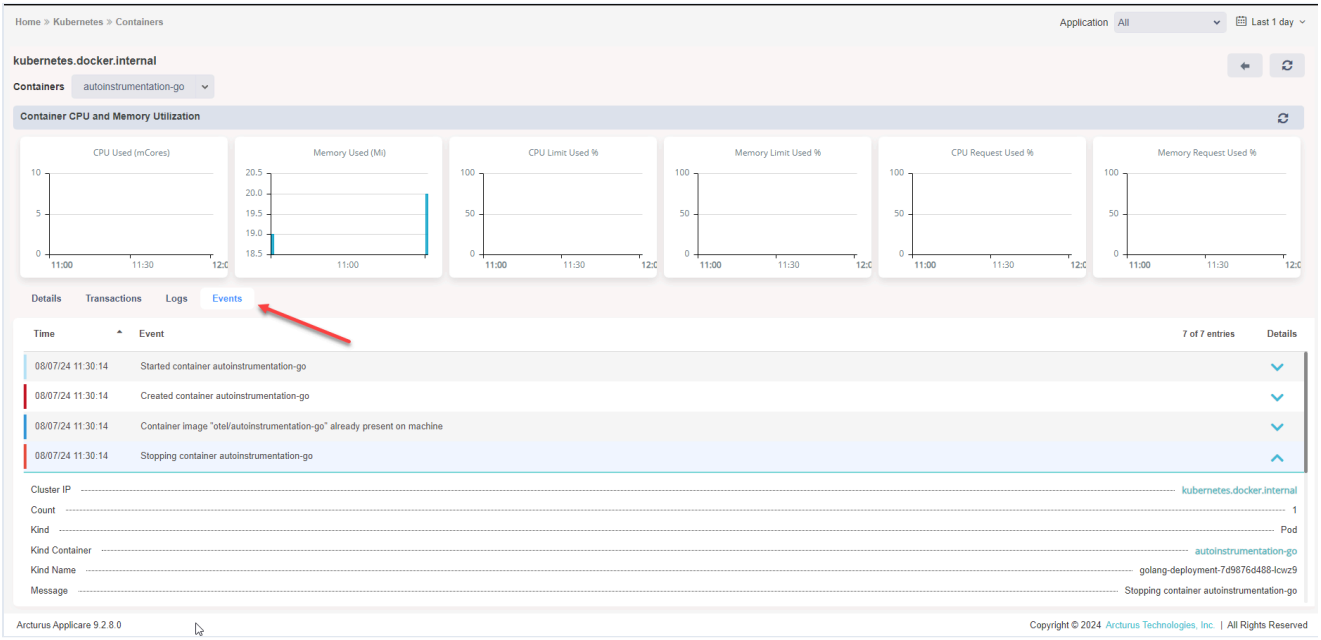
10. Click on the **Transactions** tab to view the Application transaction details.



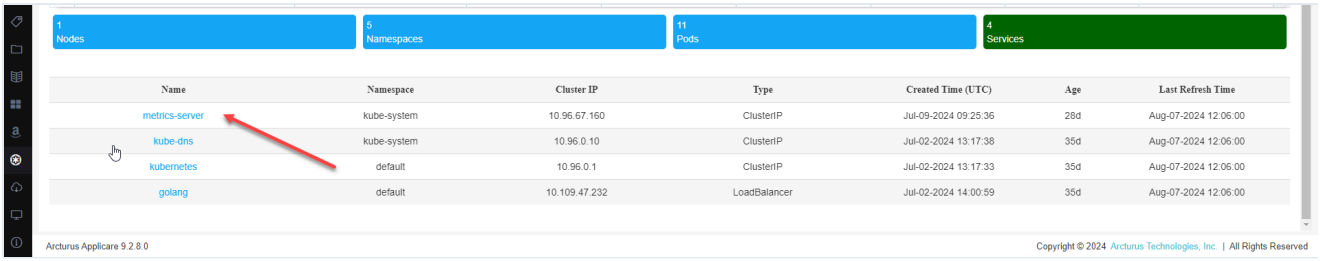
11. Click on the **Logs** tab to view the **Container** Log entries.



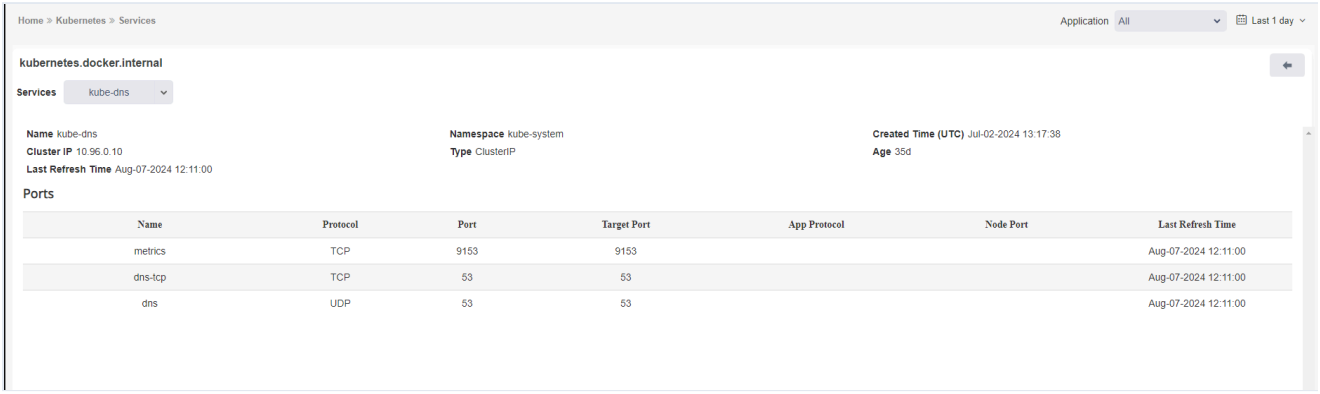
12. Click the **Events** tab to view the container event entries.



Services



13. Click the **Name** column to view the details of the **service**.



Configuration

* Configuring Local Kubernetes Monitoring

* Configuring Remote Kubernetes Monitoring

You can use Applicare to assist with your Monitoring needs.

Monitoring with Applicare

CONFIGURATION GUIDES › KUBERNETES

Monitoring Nginx in a Kubernetes

Applicare 9.2 · Updated May 9, 2026 · kubernetes, nginx · **Pending technical review**

Step 1: Add the following line next to the **From** line in the **Dockerfile** that builds Nginx.

```bash
RUN apt-get update && \
apt-get install unzip -y && \
apt-get install curl -y && \
apt-get clean

RUN mkdir /usr/local/Applicare
WORKDIR /usr/local/Applicare
RUN curl $CONTROLLER_PROTOCOL://$CONTROLLER_IP:$CONTROLLER_PORT/applicare/standalone?
filename=ApplicareSingleAgentLinux >ApplicareSingleAgentLinux.zip
RUN unzip ApplicareSingleAgentLinux.zip
RUN ["rm", "-frv", "/usr/local/Applicare/ApplicareSingleAgentLinux.zip"]
COPY startAgent.sh /usr/local/Applicare/agentscripts/startAgent.sh
COPY start_nginx.sh /usr/local/Applicare/start_nginx.sh
RUN chmod -R 755 /usr/local/Applicare
WORKDIR /usr/local/Applicare
CMD ["./start_nginx.sh"]
```

Replace the below values in the above command

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

Step 2: Create a file named **start_nginx.sh** with the following content:

```bash
#!/bin/bash# Start Nginx in the backgroundnginx -g "daemon off;" &# Start our
Agent/usr/local/Applicare/startAgent.sh
```

Step 3: Create a file named **startAgent.sh** with the content below.

BASH

```
#!/bin/sh
SERVER_NAME=`hostname`_sa

export SERVER_NAME

if [ -z "$ARCTURUS_HOME" ]; then

ARCTURUS_HOME="/usr/local/Applicare"

fi

export ARCTURUS_HOME

JAVA_HOME_PATH=$JAVA_HOME

INBUILD_JAVA_PATH=$ARCTURUS_HOME/jre

export INBUILD_JAVA_PATH

if [[ $(uname -s) == "Darwin" ]]; then

  JAVA_HOME_PATH="$ARCTURUS_HOME/jre/mac/zulu-8_jre/Contents/Home"

else

  JAVA_HOME_PATH="$ARCTURUS_HOME/jre/linux/jre"

fi

#export JAVA_HOME

export JAVA_HOME_PATH

JAVA_OPTS="-noverify -Xmx1024m -Dserver.name=$SERVER_NAME -
javaagent:$ARCTURUS_HOME/applicare-db-aspects.jar=verbosity=0,logfile=false,norestart-
intellitrace=false -Darcturus_home=$ARCTURUS_HOME -XX:HeapDumpPath=/dev/null -
Djava.library.path=.:$ARCTURUS_HOME/bin -Dapplicare.standalone=true"

export JAVA_OPTS

EXECUTABLE_NAME=$ARCTURUS_HOME/AgentStartHelper.jar

export EXECUTABLE_NAME
```

```
if [ -d "$JAVA_HOME_PATH/bin" ] ; then

 "$JAVA_HOME_PATH/bin"/java -jar $JAVA_OPTS $EXECUTABLE_NAME

else

 java -jar $JAVA_OPTS $EXECUTABLE_NAME

fi
```

Step 4: In the `nginx.conf` file, make the following changes:

a.) Comment out the existing `log_format` directive and add the new `log_format` as shown below.

```
TEXT
log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
'$status $body_bytes_sent "$http_referer" '                    '"$http_user_agent"
"$http_x_forwarded_for"';
```

```
http {
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;
```

b.) Uncomment the **access_log** line if it is already commented.

```
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;
```

c.) Add the below lines inside the **server** module

```
TEXT
location /nginx-status {stub_status on;allow 127.0.0.1;}
```

```
server {
    listen       80;
    #listen       [::]:83;
    listen       127.0.0.1:80;

    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }

    location /nginx-status {
    stub_status on;
    allow 127.0.0.1;
    }
}
```

Step 5: Add the following line to the container specification in the YAML file that starts Nginx on Kubernetes.

```
TEXT
hostname: <hostname>
```

Replace **<hostname>** with the name which we need to add the agent in Applicare.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-ubuntug-deployment
  namespace: nginx-ubuntu
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-ubuntu
  template:
    metadata:
      labels:
        app: nginx-ubuntu
    spec:
      hostname: nginx-ubuntu
      containers:
      - name: nginx-ubuntu
        image: nginx-ubuntu:latest
        ports:
        - containerPort: 80
```

Step 6: Log in to the Applicare console, go to the **Overview** menu, select the newly added agent, and click the **Edit** button.

Step 7: Enable the "**Add Nginx Configuration**" checkbox and add the below value.

1. Log file path - <**access log file location**>

2. Nginx Host - 127.0.0.1

3. Nginx Port - <**Nginx_Port**>

4. Nginx Protocol - http (or) https

Click the **update** button to save.

CONFIGURATION GUIDES › KUBERNETES

Deploy Go auto instrumentation in Kubernetes

Applicare 9.2 · Updated April 1, 2025 · kubernetes, go, opentelemetry · **Pending technical review**

Step 1: Create an **autoinstrument.yaml** file with the following content.

```yaml
spec:
template:
spec:
shareProcessNamespace: true
containers:
- name: autoinstrumentation-go
image: otel/autoinstrumentation-go
imagePullPolicy: IfNotPresent
env:
- name: OTEL_GO_AUTO_TARGET_EXE
value: /<File_executable_path>
- name: OTEL_EXPORTER_OTLP_ENDPOINT
value: "http://<Applicare_ip>:<Applicare_collector_port>"
- name: OTEL_SERVICE_NAME
value: "<service_name>"
securityContext:
runAsUser: 0
privileged: true
```

It is important to change the following values in the above file.

<File_executable_path>= Directory for starting the Go application

<Applicare_ip>= IP address of the machine running Applicare

<Applicare_collector_port>= Port of the machine running Applicare

<Service_Name>= The name of the agent as shown in the Applicare console.

Step 2: Go to the directory with the YAML file and run the command below

```bash
kubectl patch deployment <deployment_name> --patch "$(cat autoinstrument.yaml )"
```

It is important to change the below values in the above command.

<deployment_name>= Go Application Deployment Name

Step 3: This will restart your pods in the Kubernetes deployment.

The following document outlines how to access Go application data in Applicare after configuration.

Go application data in Kubernetes Monitoring

CONFIGURATION GUIDES › OPENTELEMETRY

Applicare OTel Collector installation on Linux

Applicare 9.2 · Updated September 9, 2025 · opentelemetry, linux · **Pending technical review**

Applicare OTel Collector is only supported on Redhat, Ubuntu and Debian OS.

Minimum required version: RHEL ≥ 7.9, Ubuntu ≥ 18.04 and Debian ≥ 10

Step 1: Execute the command below inside the **applicare_otel_collector** directory in the **Applicare home directory**. (Create the **applicare_otel_collector\linux** directory if it does not exist)

If have Wget, execute the command below.

```bash
sudo wget
https://applicare.blob.core.windows.net/demoapplication/applicare_otel_collector.zip
```

(OR)

If have Curl, execute the command below.

```bash
sudo curl -O
https://applicare.blob.core.windows.net/demoapplication/applicare_otel_collector.zip
```

Step 2: Install **unzip,** if not already installed.

```bash
sudo apt install unzip -y
```

Step 3: Execute below command to unzip the file.

```bash
sudo unzip applicare_otel_collector.zip
```

Step 4: Execute the below command to give the permission.

```bash
sudo chmod -R 750 *
```

Step 5: Execute the command below to install the collector.

```bash
sudo ./install_applicare_otel_collector.sh
```

Step 6: After executing the command, some packages will be installed, please wait for a few minutes.

After everything is done, a message like the one below will appear, meaning that Applicare collector is running.

```
[applicare@agentmachine applicare-otel-collector]$ sudo ./install_applicare_otel_collector.sh
===> Detected OS: rhel 9.4
===> Configuring Elasticsearch...
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /usr/lib/systemd/syste
m/elasticsearch.service.
===> Waiting for Elasticsearch to start...
âœ… Elasticsearch is running!
===> Setting environment variables...
===> Installing Applicare OTel Collector & Query binaries...
Created symlink /etc/systemd/system/multi-user.target.wants/applicare-otel-collector.service → /etc/system
d/system/applicare-otel-collector.service.
Created symlink /etc/systemd/system/multi-user.target.wants/applicare-query.service → /etc/systemd/system/
applicare-query.service.
âœ… Setup complete!
ðŸ“Š Manage services with: systemctl status elasticsearch|applicare-otel-collector|applicare-query
```

The following port must be opened on the system where the ApplicareCollector is running.

4318

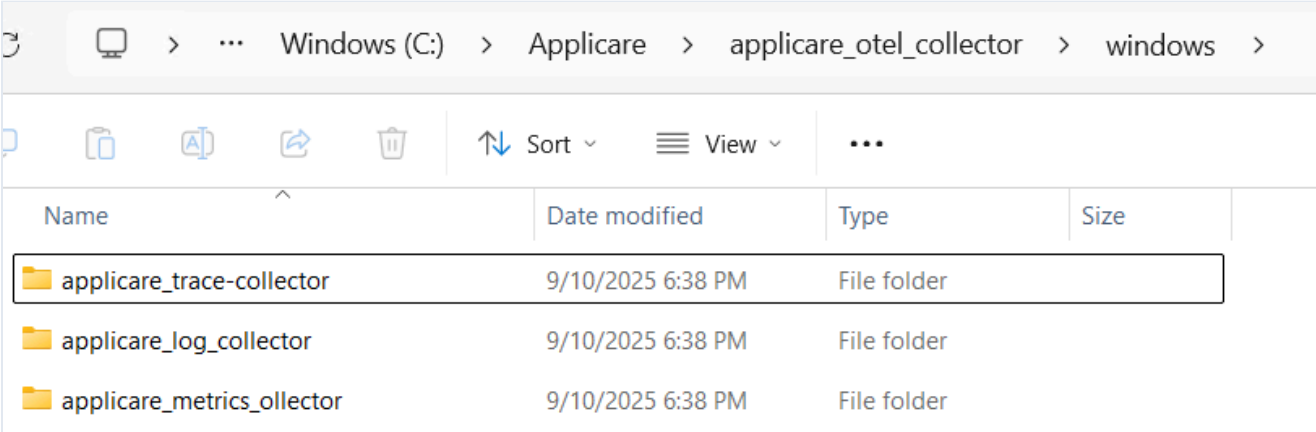
CONFIGURATION GUIDES › OPENTELEMETRY

Applicare Metrics, Trace and Log Collector installation on Windows

Applicare 9.2 · Updated October 29, 2025 · opentelemetry, windows · **Pending technical review**

Step 1: Download **Applicare collector** from the link below and extract it to the directory named **applicare_otel_collector\windows** in the Applicare home directory.

```
TEXT
https://applicare.blob.core.windows.net/demoapplication/windows_collector.zip
```



Step 2: We can see three different directories.

Metric collector installation:

If want to install **Metric Collector**, open the command prompt in **administrator** mode, go to the **applicare_metrics_collector** directory and execute the **create_metrics_collector_task.bat** file.

```
C:\Applicare\applicare_otel_collector\windows\applicare_metrics_collector>create_metrics_collector_task.bat
Creating task "\ApplicareCollector\start_metric_collector"...
SUCCESS: The scheduled task "\ApplicareCollector\start_metric_collector" has successfully been created.
SUCCESS: "\ApplicareCollector\start_metric_collector" created.
Starting metric collector...
Creating task "\ApplicareCollector\stop_metric_collector"...
SUCCESS: The scheduled task "\ApplicareCollector\stop_metric_collector" has successfully been created.
SUCCESS: "\ApplicareCollector\stop_metric_collector" created.

Task created successfully!:
```

The following port must be opened on the system.

9090

Trace collector installation:

If want to install **Trace Collector**, open the command prompt in **administrator** mode, go to the **applicare_trace_collector** directory and execute the **create_collector_task.bat** file.

After executing, it will take a minute for elasticsearch to run, so wait until then and do not close the command window.

```
D:\windows_collector\applicare_trace-collector>create_collector_task.bat

SUCCESS: Specified value was saved.
Successfully set SPAN_STORAGE_TYPE to "elasticsearch"

SUCCESS: Specified value was saved.
Successfully set ES_SERVER_URLS to "http://localhost:9200"
Creating task "\ApplicareCollector\startelasticsearch"...
SUCCESS: The scheduled task "\ApplicareCollector\startelasticsearch" has successfully been created.
SUCCESS: "\ApplicareCollector\startelasticsearch" created.
Starting Elasticsearch task...
SUCCESS: Attempted to run the scheduled task "\ApplicareCollector\startelasticsearch".
Creating task "\ApplicareCollector\stopelasticsearch"...
SUCCESS: The scheduled task "\ApplicareCollector\stopelasticsearch" has successfully been created.
SUCCESS: "\ApplicareCollector\stopelasticsearch" created.
NOTE: This task will not run now. You must run it manually when needed.
Creating task "\ApplicareCollector\applicare-collector"...
SUCCESS: The scheduled task "\ApplicareCollector\applicare-collector" has successfully been created.
Creating task "\ApplicareCollector\applicare-query"...
SUCCESS: The scheduled task "\ApplicareCollector\applicare-query" has successfully been created.
ECHO is off.
ECHO is off.
ECHO is off.
"It will take 120 seconds for Elasticsearch to start completely. Please wait 120 seconds."

Waiting for   0 seconds, press a key to continue ...
SUCCESS: "\ApplicareCollector\applicare-collector" created.
Starting collector...
SUCCESS: Attempted to run the scheduled task "\ApplicareCollector\applicare-collector".
SUCCESS: "\ApplicareCollector\applicare-query" created.
Starting collector...
SUCCESS: Attempted to run the scheduled task "\ApplicareCollector\applicare-query".
Creating task "\ApplicareCollector\stopcollector"...
SUCCESS: The scheduled task "\ApplicareCollector\stopcollector" has successfully been created.
SUCCESS: "\ApplicareCollector\stopcollector" created.
```

The following port must be opened on the system.

4318

Log collector installation:

If want to install **Log Collector**, open the command prompt in **administrator** mode, go to the **applicare_log_collector** directory and execute the **create_logcollector_task.bat** file.

```
C:\Applicare\applicare_otel_collector\windows\applicare_log_collector>create_logcollector_task.bat
Creating task "\ApplicareCollector\start_log_collector"...
SUCCESS: The scheduled task "\ApplicareCollector\start_log_collector" has successfully been created.
SUCCESS: "\ApplicareCollector\start_log_collector" created.
Starting log collector...
SUCCESS: Attempted to run the scheduled task "\ApplicareCollector\start_log_collector".
Creating task "\ApplicareCollector\stop_log_collector"...
SUCCESS: The scheduled task "\ApplicareCollector\stop_log_collector" has successfully been created.
SUCCESS: "\ApplicareCollector\stop_log_collector" created.

Task creation process completed!
```

The following port must be opened on the system.

3100

CONFIGURATION GUIDES › OPENTELEMETRY

Monitor Traces and Metrics in Containerized .NET Applications with Applicare and OpenTelemetry

Applicare 9.2 · Updated August 22, 2025 · opentelemetry, dotnet, docker · **Pending technical review**

To enable OpenTelemetry auto-instrumentation for your .NET application and send traces to Applicare, you will need to apply configuration changes in both your **Dockerfile** and your application's **YAML deployment script** (e.g., Kubernetes Deployment, Docker Compose file, or other container orchestration tool's configuration).

Step 1: Modify your Dockerfile

Add the following lines to your application's Dockerfile. These commands will install necessary utilities, download the OpenTelemetry .NET auto-instrumentation setup script, and Applicare.NetTrace setup, execute it.

Add the Diagnostic tools stage below before the FROM base AS final (or) FROM runtime AS final stage

```text
# Diagnostic tools stage
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS diagnostic-tools
RUN dotnet tool install --tool-path /dotnet-tools dotnet-counters \
&& dotnet tool install --tool-path /dotnet-tools dotnet-trace \
&& dotnet tool install --tool-path /dotnet-tools dotnet-dump
```

```
# See https://aka.ms/customizecontainer ...
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
USER $APP_UID
WORKDIR /app
EXPOSE 8080

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["WebApplication1/WebApplication1.csproj", "WebApplication1/"]
RUN dotnet restore "./WebApplication1/WebApplication1.csproj"
COPY . .
WORKDIR "/src/WebApplication1"
RUN dotnet build "./WebApplication1.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./WebApplication1.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

#Diagnostic tools stage ✅ PLACE IT HERE
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS diagnostic-tools
RUN dotnet tool install --tool-path /dotnet-tools dotnet-counters \
 && dotnet tool install --tool-path /dotnet-tools dotnet-trace \
 && dotnet tool install --tool-path /dotnet-tools dotnet-dump

FROM base AS final
WORKDIR /app
```

Note: The `apt-get` commands assume a Debian/Ubuntu-based base image (e.g., `mcr.microsoft.com/dotnet/aspnet:8.0-alpine` would use `apk add --no-cache` instead). Adjust package manager commands as per your base image.

Below lines should be added inside final runtime stage, and specifically after FROM base AS final (or) FROM runtime AS final

```
TEXT
ENV OTEL_AGENT_VERSION=1.9.0
ENV DOTNET_EnableDiagnostics=1
ENV OTEL_DOTNET_AUTO_HOME=/otel-dotnet-auto
ENV DOTNET_SHARED_STORE=$OTEL_DOTNET_AUTO_HOME/store
ENV
DOTNET_STARTUP_HOOKS=$OTEL_DOTNET_AUTO_HOME/net/OpenTelemetry.AutoInstrumentation.StartupHook
.dll
ENV DOTNET_ADDITIONAL_DEPS=$OTEL_DOTNET_AUTO_HOME/AdditionalDeps
```

```
#Diagnostic tools stage  ✅  PLACE IT HERE
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS diagnostic-tools
RUN dotnet tool install --tool-path /dotnet-tools dotnet-counters \
 && dotnet tool install --tool-path /dotnet-tools dotnet-trace \
 && dotnet tool install --tool-path /dotnet-tools dotnet-dump

FROM base AS final
WORKDIR /app

ENV OTEL_AGENT_VERSION=1.9.0
ENV DOTNET_EnableDiagnostics=1
ENV OTEL_DOTNET_AUTO_HOME=/otel-dotnet-auto
ENV DOTNET_SHARED_STORE=$OTEL_DOTNET_AUTO_HOME/store
ENV DOTNET_STARTUP_HOOKS=$OTEL_DOTNET_AUTO_HOME/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll
ENV DOTNET_ADDITIONAL_DEPS=$OTEL_DOTNET_AUTO_HOME/AdditionalDeps
```

a.) If "USER $APP_UID" already exists in the Dockerfile, add the following lines after the ENV stage.

```
TEXT
USER root
RUN apt-get update && apt-get install -y unzip curl
ADD https://github.com/open-telemetry/opentelemetry-dotnet-
instrumentation/releases/download/v1.9.0/otel-dotnet-auto-install.sh /otel-install.sh
RUN chmod +x /otel-install.sh && \
   /otel-install.sh
RUN curl -L https://applicare.blob.core.windows.net/demoapplication/arc_dotnet.zip -o
arc_dotnet.zip \
&& unzip arc_dotnet.zip -d arc_dotnet \
&& rm arc_dotnet.zip
COPY --from=diagnostic-tools /dotnet-tools /dotnet-tools
ENV PATH="$PATH:/dotnet-tools"
ENV APPLICARE_URL=Protocol://ApplicareIP:ApplicarePort
USER $APP_UID
```

(OR)

b.) If "USER $APP_UID" not exists in the Dockerfile, add the following lines after the ENV stage.

```
TEXT
RUN apt-get update && apt-get install -y unzip curl ADD https://github.com/open-
telemetry/opentelemetry-dotnet-instrumentation/releases/download/v1.9.0/otel-dotnet-auto-
install.sh /otel-install.sh RUN chmod +x /otel-install.sh && \    /otel-install.sh

# Download and unzip arc_dotnet.zip
RUN curl -L https://applicare.blob.core.windows.net/demoapplication/arc_dotnet.zip -o
arc_dotnet.zip \
&& unzip arc_dotnet.zip -d arc_dotnet \
&& rm arc_dotnet.zip

# ✅ Copy diagnostic tools into runtime
COPY --from=diagnostic-tools /dotnet-tools /dotnet-tools

ENV PATH="$PATH:/dotnet-tools"
ENV APPLICARE_URL=Protocol://ApplicareIP:ApplicarePort # e.g. http://192.168.1.100:8880
```

```
ENV DOTNET_ADDITIONAL_DEPS=$OTEL_DOTNET_AUTO_HOME/AdditionalDeps
ENV OTEL_TRACES_EXPORTER=otlp
ENV OTEL_DOTNET_AUTO_LOGGING_ENABLED=true
ENV OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT=true
ENV OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT=true
ENV COMPlus_EnableDiagnostics=1
ENV DOTNET_EnableDiagnostics=1
ENV DOTNET_ROLL_FORWARD=LatestMajor

USER root

RUN apt-get update && \
    apt-get install -y curl unzip && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

ADD https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases/download/v1.9.0/otel-dotnet-auto-install.sh /otel-install.sh
RUN chmod +x /otel-install.sh && \
    /otel-install.sh

# Download and unzip arc_dotnet.zip
RUN curl -L https://applicare.blob.core.windows.net/demoapplication/arc_dotnet.zip -o arc_dotnet.zip \
 && unzip arc_dotnet.zip -d arc_dotnet \
 && rm arc_dotnet.zip

# ✅ Copy diagnostic tools into runtime
COPY --from=diagnostic-tools /dotnet-tools /dotnet-tools
ENV PATH="$PATH:/dotnet-tools"

ENV APPLICARE_URL=https://testenv.applicare.io:443
```

We need to add **ApplicareDotnetTrace.dll** to the Entrypoint and configure it to start.

```
TEXT
Before Update ENTRYPOINT in Dockerfile
ENTRYPOINT ["dotnet", WebApplication1.dll]

After Update ENTRYPOINT in Dockerfile
ENTRYPOINT sh -c "dotnet WebApplication1.dll & OTEL_DOTNET_AUTO_INSTRUMENTATION_ENABLED=false
dotnet ./arc_dotnet/ApplicareDotnetTrace.dll"
```

```
USER $APP_UID
COPY --from=publish /app/publish .

ENTRYPOINT sh -c "dotnet WebApplication1.dll & OTEL_DOTNET_AUTO_INSTRUMENTATION_ENABLED=false  dotnet ./arc_dotnet/ApplicareDotnetTrace.dll"
```

Step 2: Install Opentelemetry collector in your Kubernetes cluster. Download the yaml file and change **<Applicare_Ip>** in that yaml file.

If have Wget, execute the command below.

```
BASH
sudo wget
https://applicare.blob.core.windows.net/demoapplication/opentelemetry_collector.yaml
```

```
TEXT
                                                                **(OR)**
```

If have Curl, execute the command below.

```
BASH
sudo curl -O
https://applicare.blob.core.windows.net/demoapplication/opentelemetry_collector.yaml
```

Then apply that yaml file to install collector in Kubernetes cluster.

```
BASH
kubectl apply -f opentelemetry_collector.yaml
```

Step 3: Configure Environment Variables in your YAML Deployment

Add the following environment variables to the container definition section of your application's YAML deployment file (e.g., under `env:` in a Kubernetes `Deployment`'s container specification, or `environment:` in a Docker Compose `service`). These variables instruct the OpenTelemetry auto-instrumentation how to behave and where to send the telemetry data.

YAML

```
          env:
            - name: OTEL_DOTNET_AUTO_HOME
              value: "/otel-dotnet-auto"
            - name: DOTNET_SHARED_STORE
              value: "/otel-dotnet-auto/store"
            - name: DOTNET_STARTUP_HOOKS
              value: "/otel-dotnet-
auto/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll"
            - name: DOTNET_ADDITIONAL_DEPS
              value: "/otel-dotnet-auto/AdditionalDeps"
            # Kubernetes Metadata (Critical Fix)
            - name: KUBE_POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: KUBE_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            # Resource Attributes (Updated)
            - name: OTEL_RESOURCE_ATTRIBUTES
              value: "k8s.pod.name=$(KUBE_POD_NAME),k8s.namespace.name=$(KUBE_NAMESPACE)"
            # Metrics Configuration
            - name: OTEL_METRICS_EXPORTER
              value: "otlp"
            - name: OTEL_DOTNET_AUTO_METRICS_ENABLED
              value: "true"
            - name: OTEL_EXPORTER_OTLP_ENDPOINT
              value: "http://otel-collector.monitoring.svc.cluster.local:4318"
            - name: OTEL_DOTNET_AUTO_METRICS_INCLUDE_PROCESS_RUNTIME
              value: "true"
            - name: OTEL_DOTNET_AUTO_METRICS_SET_SPECIFIC
              value:
"Microsoft.AspNetCore.Hosting[0],Microsoft.AspNetCore.Http.Connections[0]"
            - name: OTEL_DOTNET_AUTO_METRICS_ADDITIONAL_SOURCES
              value: "*"
            # Tracing Configuration
            - name: OTEL_TRACES_EXPORTER
              value: "otlp"
            # Instrumentation
            - name: OTEL_DOTNET_AUTO_ASPNETCORE_INSTRUMENTATION_ENABLED
              value: "true"
            - name: OTEL_DOTNET_AUTO_PROCESS_ENABLED
              value: "true"
```

```
        - name: OTEL_DOTNET_AUTO_HTTPCLIENT_ENABLED
          value: "true"
        - name: OTEL_DOTNET_AUTO_RUNTIMEMETRICS_ENABLED
          value: "true"
        - name: OTEL_DOTNET_AUTO_ENTITYFRAMEWORKCORE_SET_DBSTATEMENT_FOR_TEXT
          value: "true"
        - name: OTEL_DOTNET_AUTO_SQLCLIENT_SET_DBSTATEMENT_FOR_TEXT
          value: "true"
```

Sample YAML with the above changes.

```
    app: applicareapp
  spec:
    containers:
      - name: webapplication1
        image: applicare910/applicareapp:dotnetcoregateway
        ports:
          - containerPort: 8080
        env:
          # OpenTelemetry Auto-Instrumentation
          - name: OTEL_DOTNET_AUTO_HOME
            value: "/otel-dotnet-auto"
          - name: DOTNET_SHARED_STORE
            value: "/otel-dotnet-auto/store"
          - name: DOTNET_STARTUP_HOOKS
            value: "/otel-dotnet-auto/net/OpenTelemetry.AutoInstrumentation.StartupHook.dll"
          - name: DOTNET_ADDITIONAL_DEPS
            value: "/otel-dotnet-auto/AdditionalDeps"
          # Kubernetes Metadata (Critical Fix)
          - name: KUBE_POD_NAME
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: KUBE_NAMESPACE
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          # Resource Attributes (Updated)
          - name: OTEL_RESOURCE_ATTRIBUTES
            value: "k8s.pod.name=$(KUBE_POD_NAME),k8s.namespace.name=$(KUBE_NAMESPACE),k8s.deployment.name=$(KUBE_DEPLOYMENT_NAME)"
          # Metrics Configuration
          - name: OTEL_METRICS_EXPORTER
            value: "otlp"
          - name: OTEL_DOTNET_AUTO_METRICS_ENABLED
            value: "true"
          - name: OTEL_EXPORTER_OTLP_ENDPOINT
            value: "http://otel-collector.monitoring.svc.cluster.local:4318"
          - name: OTEL_DOTNET_AUTO_METRICS_INCLUDE_PROCESS_RUNTIME
            value: "true"
          - name: OTEL_DOTNET_AUTO_METRICS_SET_SPECIFIC
            value: "Microsoft.AspNetCore.Hosting[0],Microsoft.AspNetCore.Http.Connections[0]"
          - name: OTEL_DOTNET_AUTO_METRICS_ADDITIONAL_SOURCES
            value: "*"
          # Tracing Configuration
          - name: OTEL_TRACES_EXPORTER
            value: "otlp"
          # Instrumentation
          - name: OTEL_DOTNET_AUTO_ASPNETCORE_INSTRUMENTATION_ENABLED
            value: "true"
          - name: OTEL_DOTNET_AUTO_PROCESS_ENABLED
            value: "true"
          - name: OTEL_DOTNET_AUTO_HTTPCLIENT_ENABLED
            value: "true"
          - name: OTEL_DOTNET_AUTO_RUNTIMEMETRICS_ENABLED
            value: "true"
```

Step 4: On the machine running Applicare, download the shell script for below URL and give the executable permission for that shell script and execute the shell script.

If have Wget, execute the command below.

```bash
sudo wget
https://applicare.blob.core.windows.net/demoapplication/applicare_metric_collector_install.sh
```

```text
                                                                **(OR)**
```

If have Curl, execute the command below.

```bash
sudo curl -O
https://applicare.blob.core.windows.net/demoapplication/applicare_metric_collector_install.sh
```

The following port must be opened on the system where the Applicare controller is running.

9090 and 4318

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Applicare Installation for Linux

Applicare 9.2 · Updated July 17, 2025 · controller, linux · **Pending technical review**

Prerequisites

Essential System Requirements for Applicare Controller and Database

- Red Hat Linux 6.x or newer - RHEL 8 recommended

System Requirements

- - **Disk Space:** 10 GB
  - **RAM:** 16 GB
  - **CPU:** Intel i7 4-core or equivalent, or higher

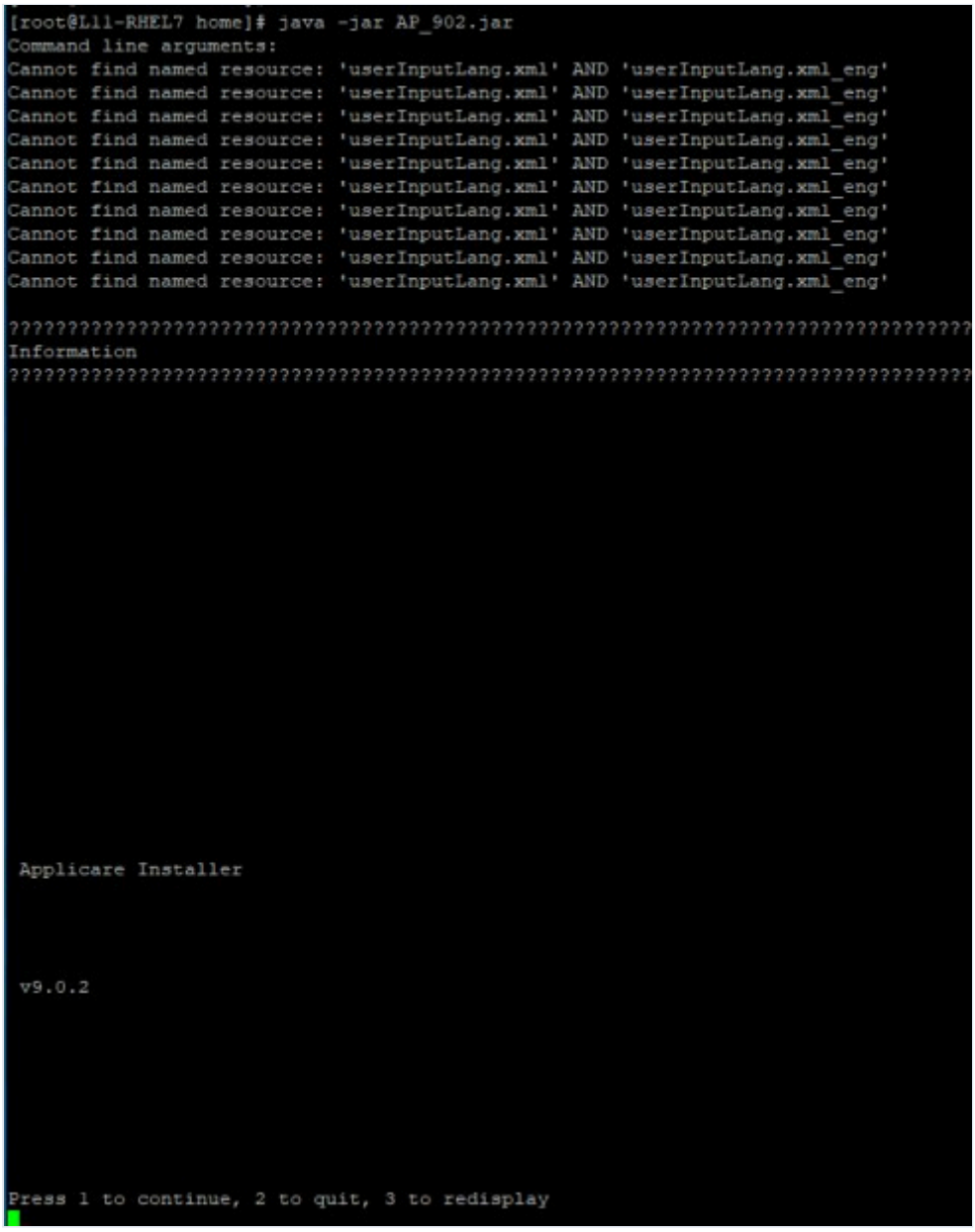
Ports Requirements

- **VM Access:** Ensure the VM is accessible to all agents via its IP address.

Additional products needed

- Java 1.8. java downloads.

Installation



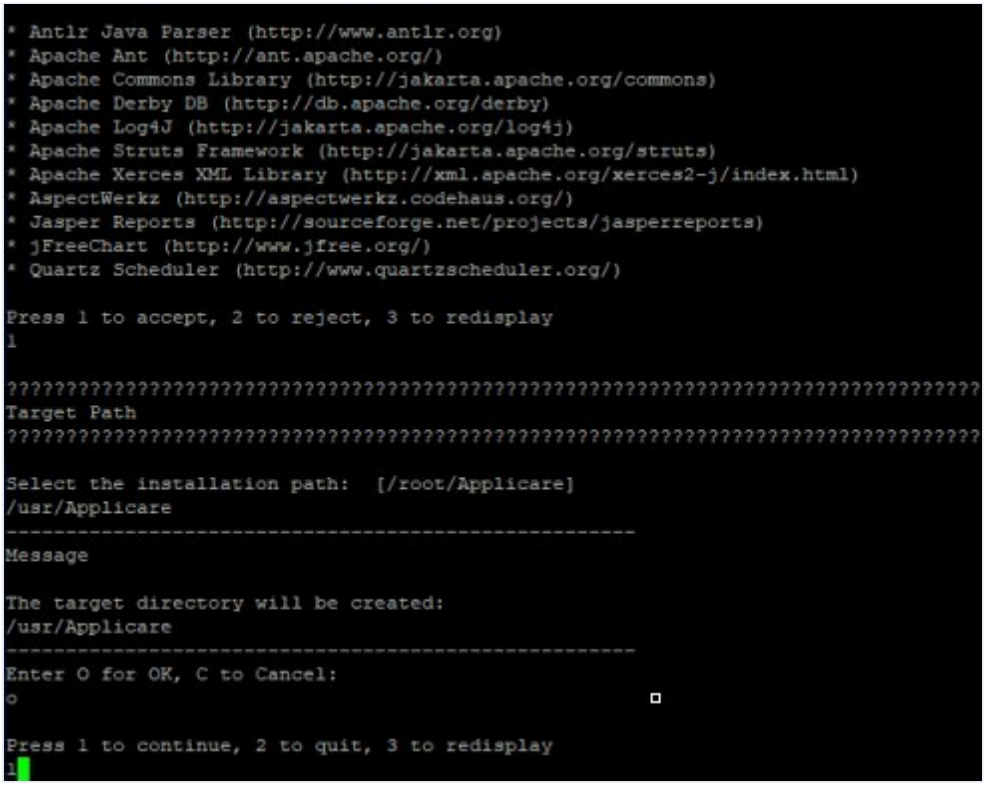
1. Copy the downloaded installer to the machine where Applicare will be installed.

- Open PuTTY and navigate to the directory containing the installer. Execute the command **java -jar AP_XXX.jar** to begin with installation process.
- Press '**1**' and 'Enter' to continue the installation.
- Next, read the license agreement. To accept it press '1' and then press 'Enter'.



2. Enter the directory to install Applicare and (APPLICARE_HOME) and press 'Enter'.

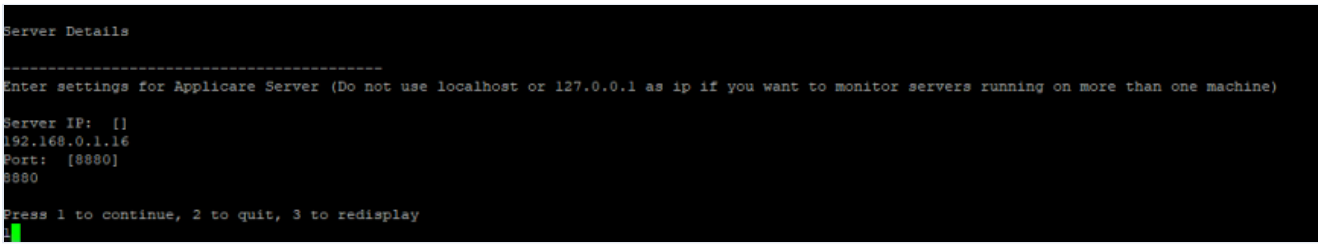
- Press **0** and then 'Enter' to create the target directory.
- Press **1** and 'Enter' to proceed with the installation.



3. Enter the **IP** address of the machine (the same machine that Applicare is being installed on to) and enter.

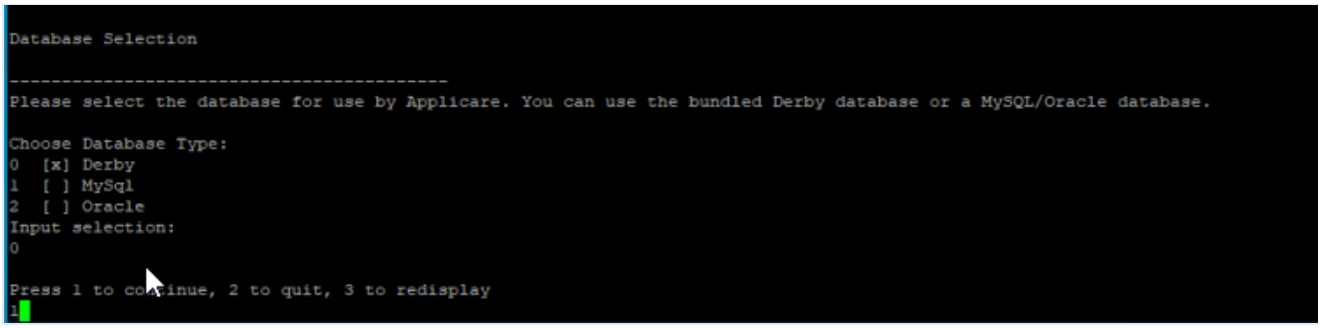
- Choose any **port** as you prefer and enter.

- Press **1** and then 'Enter' to continue installation.

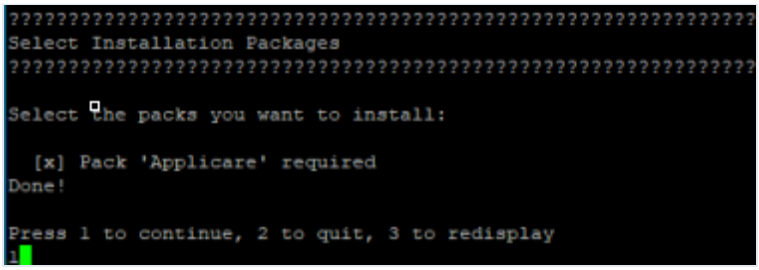


4. Select the database of your choice. Press '**1**' and then 'Enter' to proceed the installation.

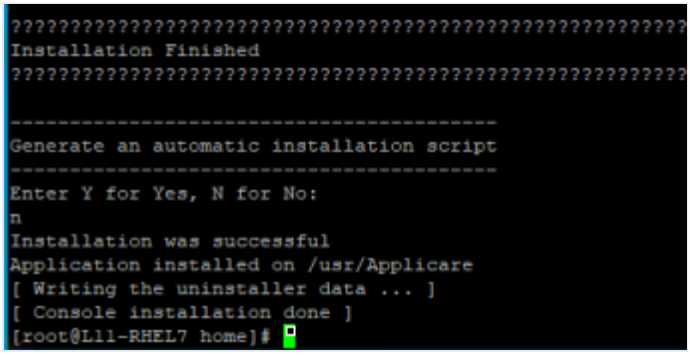
- For proof of concept (POC) purposes, we recommend using the Derby Database. For production environments or system with high workloads, we advise using MySQL or Oracle.



5. Review the installation packages on the screen, Press '**1**' and 'Enter' to proceed with the installation.



7. Installation is now completed. The automatic installation script screen will appear. Type N and press 'Enter'.



8. Grant executive permissions by running the following command from the parent directory of Applicare.

- (In the example above we installed Applicare in /usr/Applicare and hence run this command from /usr directory)

```bash
chmod -R 777 Applicare/*.
```

9. Applicare controller port (default port 8880) and 8983 must be opened to access the Applicare controller in a browser.

1. Installed on RHEL/CentOS/Fedora Os, use the following command to open the port.

```bash
sudo firewall-cmd --permanent --add-port=8880/tcp
sudo firewall-cmd --permanent --add-port=8983/tcp
sudo firewall-cmd --reload
```

2. Installed on Ubuntu/Debian Os, use the following command to open the port.

```bash
sudo ufw allow 8880/tcp
sudo ufw allow 8983/tcp
sudo ufw enable
```

Note:

To configure predictive analysis data, Refer here.

Starting Applicare Controller

For Other Docs - Click here

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Run the Applicare Controller as a service on Linux

Applicare 9.2 · Updated August 22, 2024 · controller, linux · **Pending technical review**

To **install** or **uninstall applicare.service, admin permissions** are required.

1. Navigate to the directory where the **Applicare Controller** is installed.

2. Run **./installControllerService.sh** to install **applicare.service.** This command will both install and start the service.

```
[root@L11-RHEL7 Applicare]# ./installControllerService.sh
Created symlink from /etc/systemd/system/multi-user.target.wants/applicare.servi
ce to /etc/systemd/system/applicare.service.
[root@L11-RHEL7 Applicare]#
```

1. To **view service status** execute the command **systemctl status applicare**

```
[root@L11-RHEL7 Applicare]# systemctl status applicare
● applicare.service - Applicare Controller
   Loaded: loaded (/etc/systemd/system/applicare.service; enabled; vendor preset
: disabled)
   Active: active (running) since Fri 2021-08-20 12:27:25 EDT; 2min 0s ago
 Main PID: 23486 (java)
   CGroup: /system.slice/applicare.service
           ├─23486 /usr/bin/java -Djava.util.logging.config.file=/home/rob/Ap...
           ├─23507 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7_8.x86_...
           └─23592 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7_8.x86_...
```

2. To **Stop service** execute the command **systemctl stop applicare**

```
[root@L11-RHEL7 Applicare]# systemctl stop applicare
```

3. To **Start service** execute the command **systemctl start applicare**

```
[root@L11-RHEL7 Applicare]# systemctl start applicare
[root@L11-RHEL7 Applicare]#
```

4. To **Uninstall service** execute **./uninstallControllerService.sh**

```
[root@L11-RHEL7 Applicare]# ./uninstallControllerService.sh
Removed symlink /etc/systemd/system/multi-user.target.wants/applicare.service.
[root@L11-RHEL7 Applicare]#
```

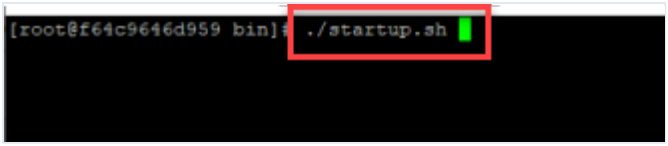
CONFIGURATION GUIDES › INSTALL, START & UPDATE

Starting Applicare Controller on a Linux system

Applicare 9.2 · Updated August 22, 2024 · controller, linux · **Pending technical review**

Method 1: Starting Controller via Command mode

Navigate to the **Applicare Controller** installation directory, then proceed to the **/server/bin** path and run **./startup.sh** to start the Applicare Controller.



Method 2:Starting the Controller in Background Mode

Navigate to the **Applicare Controller** installation directory, then go to the path **/server/bin** path and execute the following command to start the Applicare Controller.

```text
nohup ./startup.sh &
```



Method 3: Start Controller as a Service

Run Applicare controller as a service

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Applicare Installation for Windows

Applicare 9.2 · Updated December 3, 2024 · controller, windows · **Pending technical review**

Prerequisites

System Requirements for Applicare Controller and Database*

- Applicare controller and Agents must have Administrative rights
- Operating System: Windows 10 or later, or Windows Server 2012 or newer

Note that these requirements may vary depending on the number of agents deployed and the volume of data being collected and below should be used as a starting point.

System Requirements

- **Disk Space:** 100 GB (mostly for storing data) or 1 TB or higher for production monitoring.
- **RAM:** 16 GB (small POC environment) or 48 GB or higher for production.
- **CPU:** Intel i7 6-core equivalent, or higher.

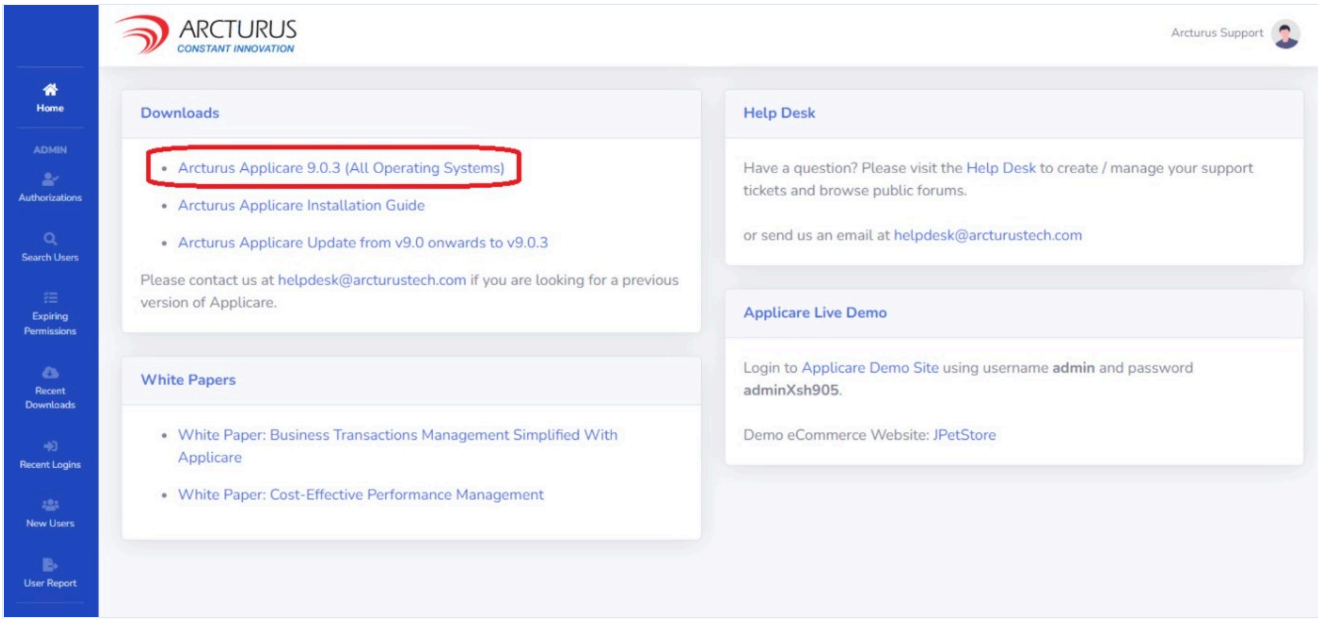
Port Requirements

- The Agent must have access to Applicare Controller IP and port. (Default port 8880)
- The VM should be accessible to all agents via its IP address

Additional Required Products

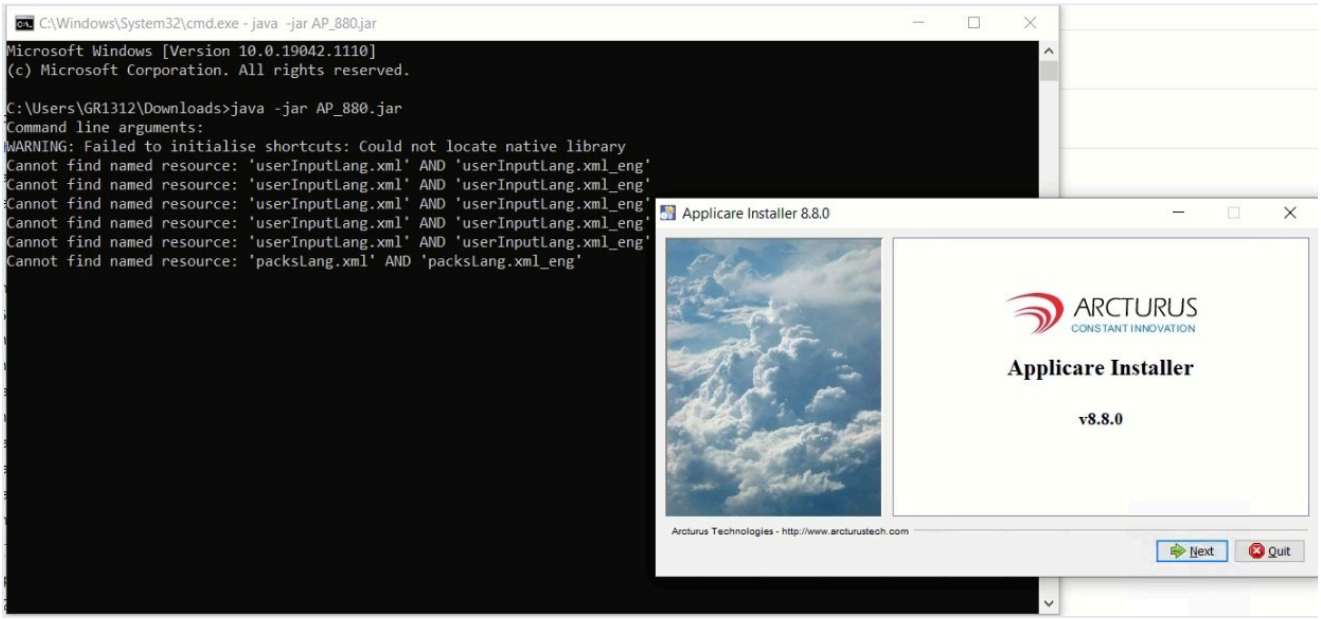
- Java 1.8. Java downloads
- MySQL is optional if you prefer using Derby as your database. If you choose MySQL, you can download and install it from MySQL's official site. To get the latest Applicare installer, please register and log in to our support site and download it from there.

Installation



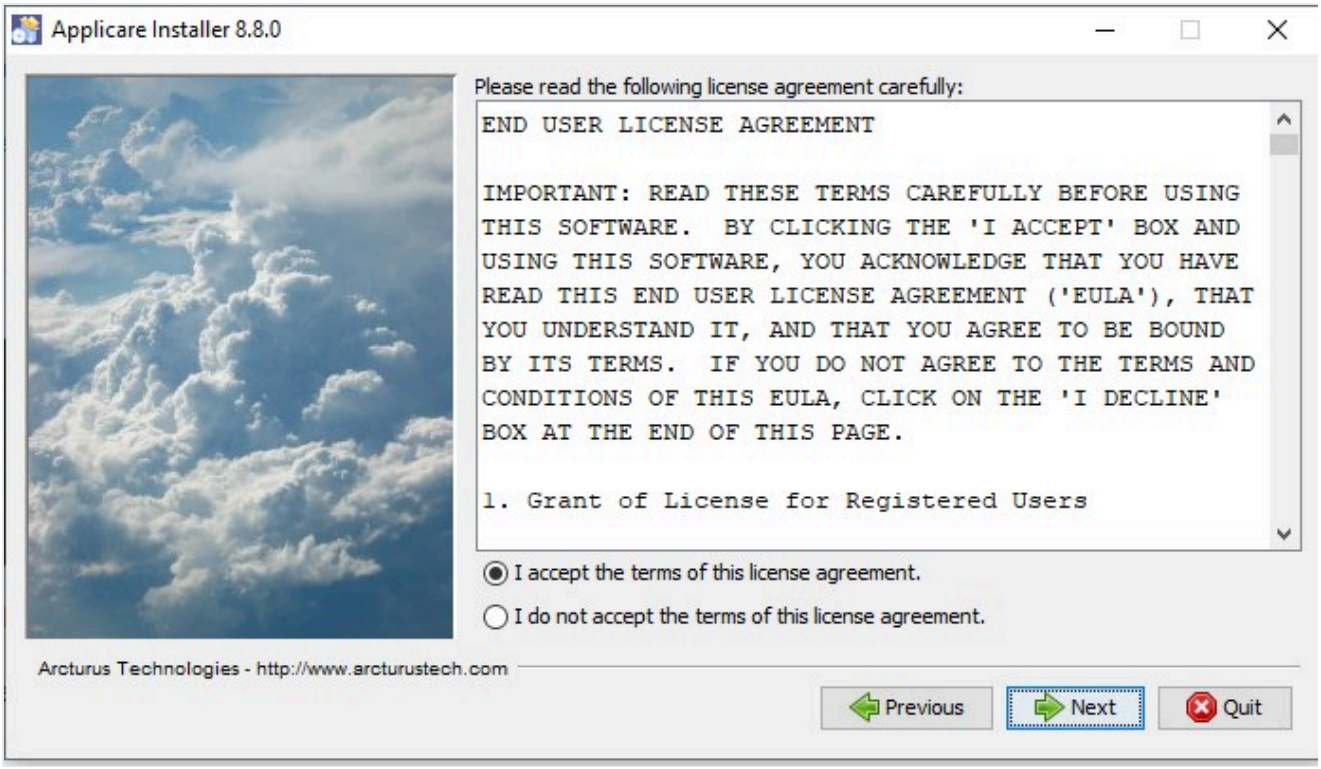
1. Transfer the downloaded installer to the machine where Applicare will be installed.

- Double-click the installer to begin the installation process.
- If double-clicking the installer doesn't work, open a command window, navigate to the directory where the installer is located and run the command. Execute the **java -jar AP_XXX.jar** command to start the installation.

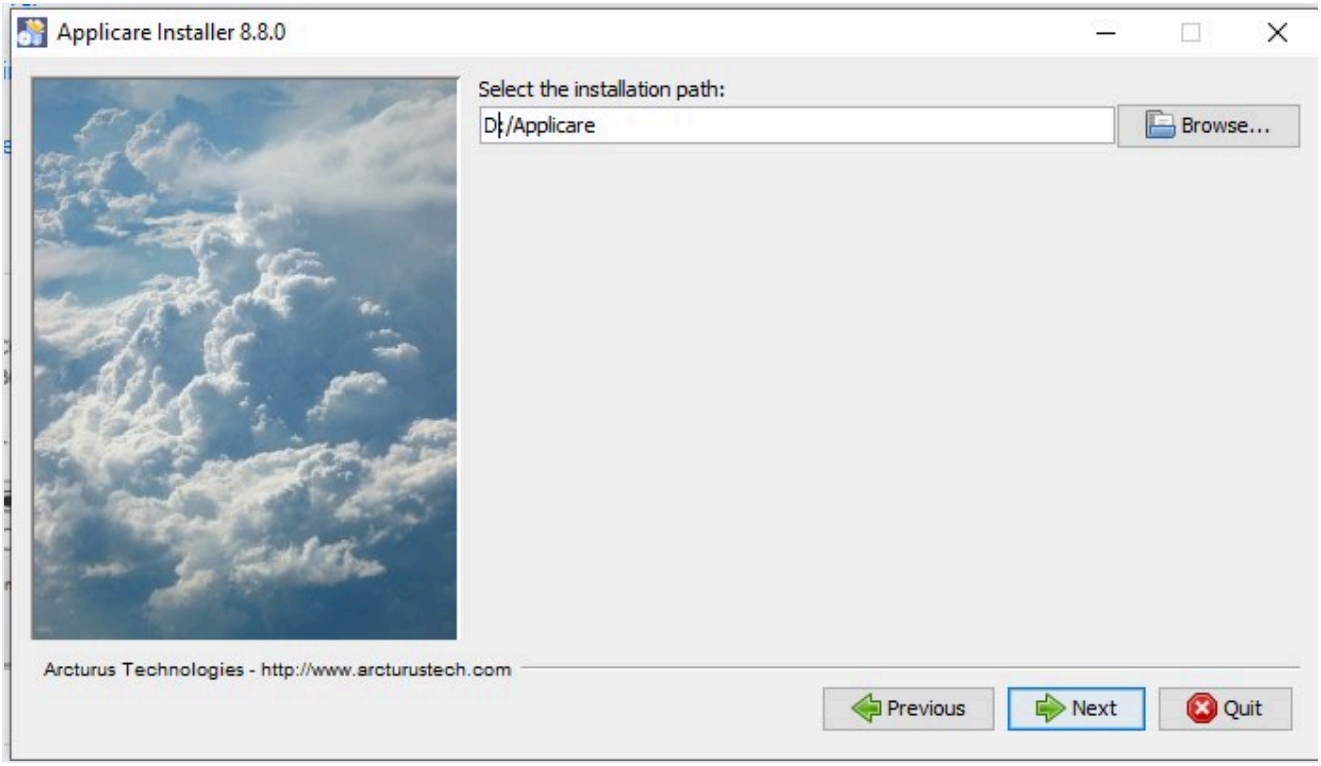


2. Click '**Next**' to proceed with the installation.

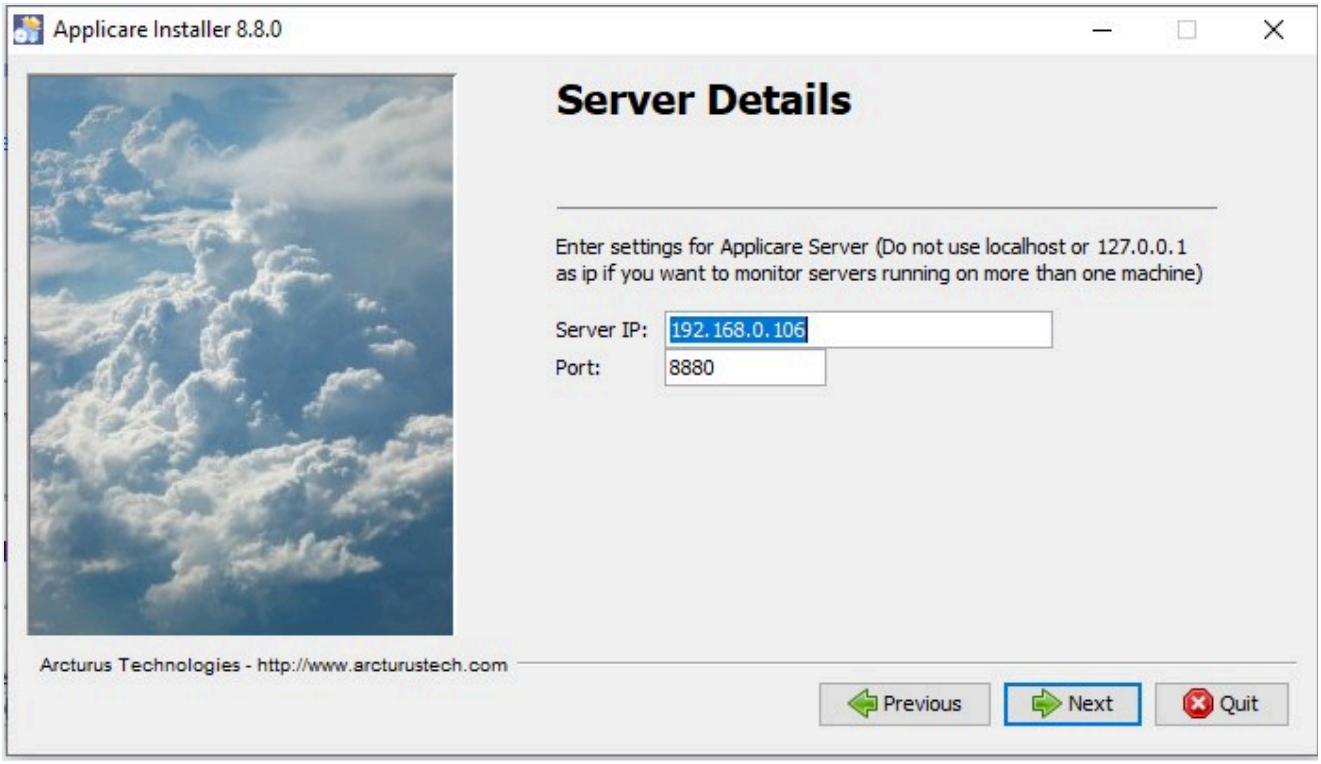
- Then, review and accept the license agreement, and click on **Next**.



3. Select the directory where you want to install Applicare (APPLICARE_HOME) and click **Next**.



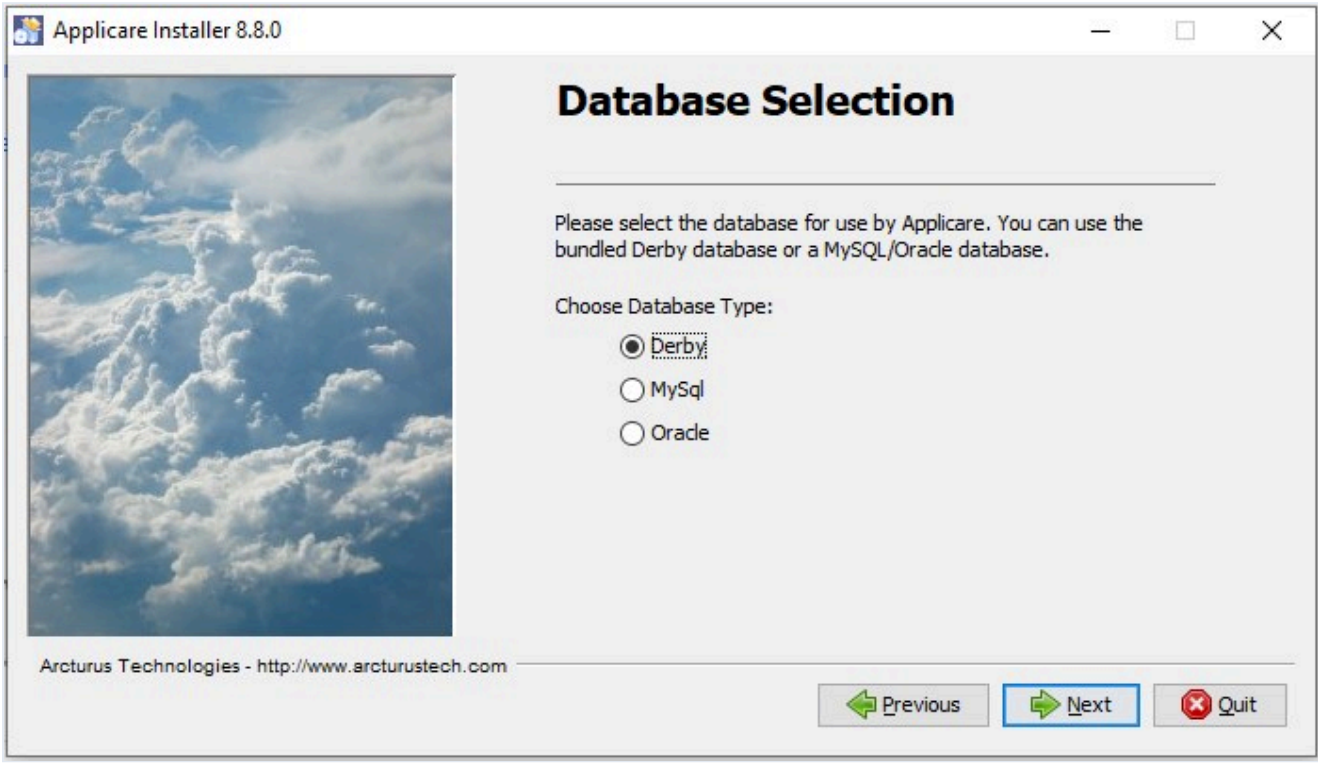
4. In the server details window, enter the IP address of the machine where Applicare is being installed and choose a preferred port.



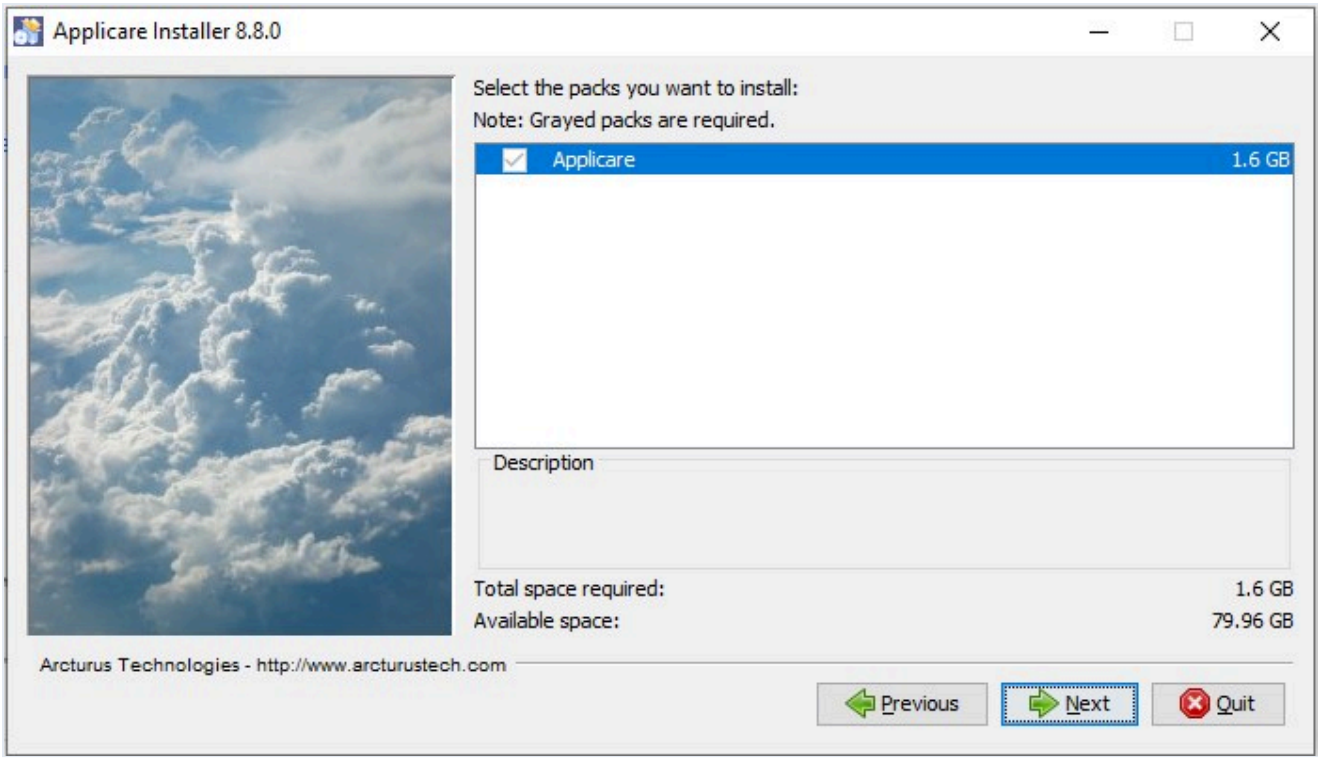
5. On the next screen, select your preferred database and click on Next.

- For proof of concept (POC) purposes, we recommend using the Derby database. For production environments or system with high load, we advise using MySQL or Oracle**.

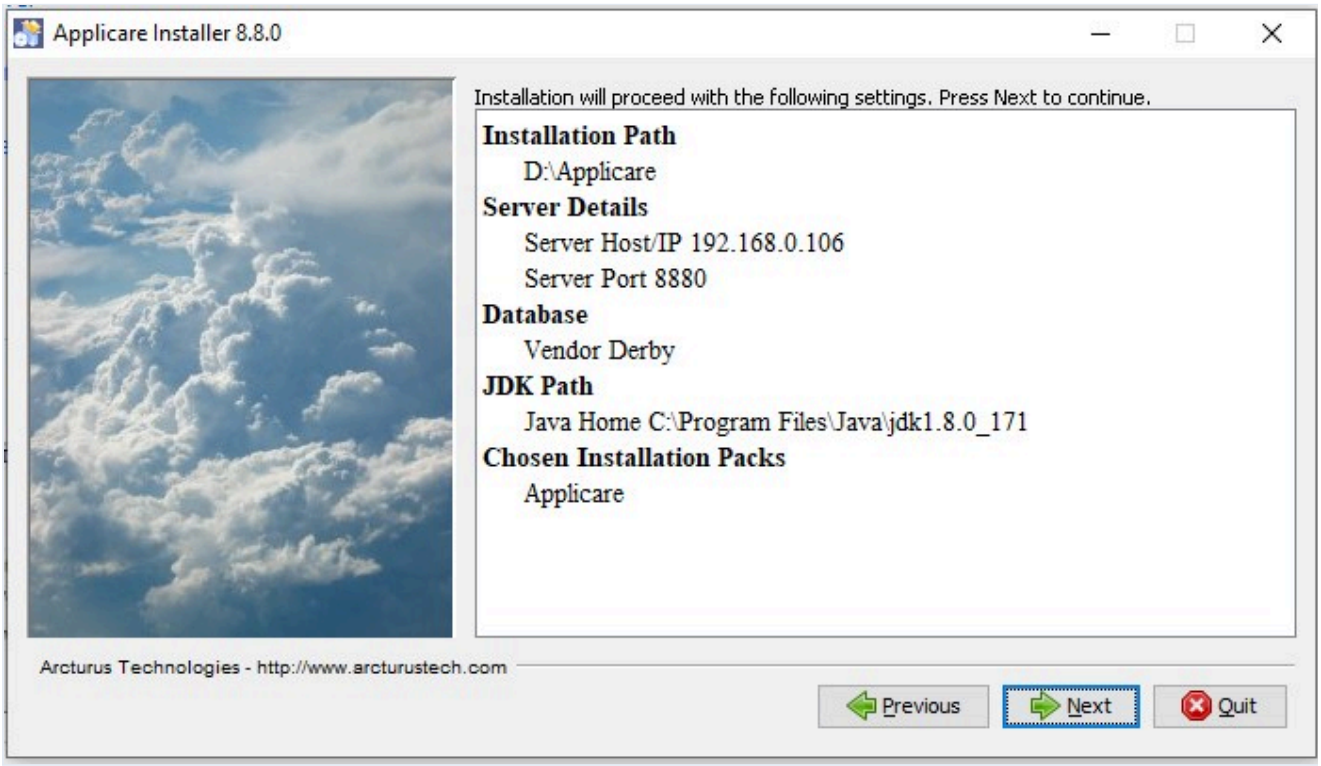
** In case of MySQL or Oracle as the database, you will need to create the schema for Applicare before proceeding past this step.

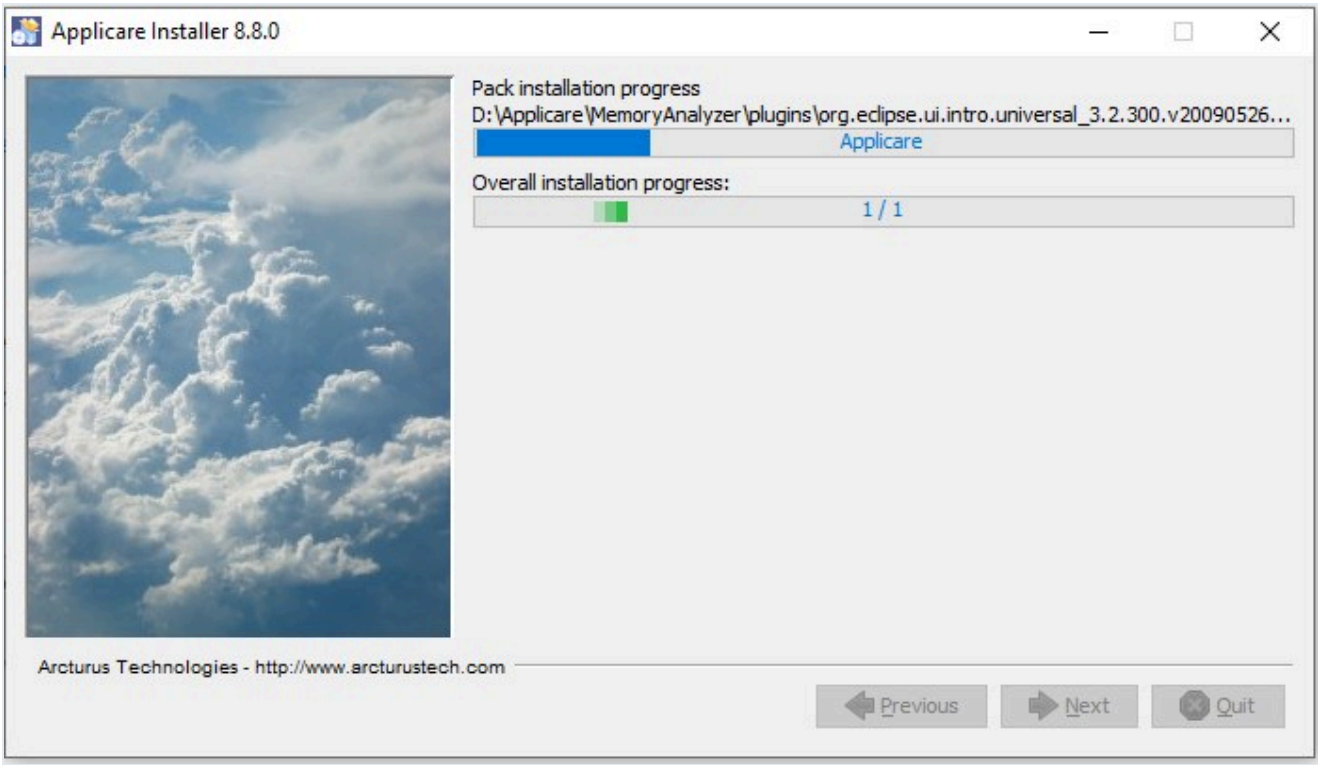


6. Choose the Applicare package and click on **Next**.



7. Review the settings and click on **Next**.





8. Click on the **Next** button once the package installation is completed.

- Then, click on **Done**.



Starting Applicare Controller

For other docs - click here

Configuring SQL Mode for MySQL in Applicare - click here

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Update the Applicare Controller

Applicare 9.2 · Updated April 2, 2025 · controller · **Pending technical review**

Please follow the steps below to update your Applicare installation

Step 1: To download the update.jar, log in with your Arcturus Support Login ID on the Arcturus Support site.

Step 2: For updating **Arcturus Applicare v9.0 onwards to v9.*** Click on provided link.



Step 3: After downloading the update.jar copy **update.jar** in the **<Applicare Installation Dir>/tmp/updates/** folder. If there is **update.jar.processed** file then delete that file.

Step 4: Stop any active Applicare processes if they are running.

Step 5: If the **Applicare controller is running on windows**, open the command prompt and cd into the directory **<Applicare Installation Dir>/scripts.** Then, execute **runupdate.bat**.

```
TEXT
                                                                    OR
```

If the **Applicare controller is running on Linux**, go to the directory **<Applicare Installation Dir>/scripts** and execute **runupdate.sh**.

If Applicare controller is running on Linux, Grant full permissions to Applicare directory after completing the update.

Step 6: On windows, navigate to **<Applicare Installation Dir>/controllerscripts.** Open a **command prompt in administrator mode** at this location and execute the **createControllerTask.bat**.

Step 7: Start the Applicare controller.

Follow the steps below to update the license file

Step 1: Download the license file and replace it in **<Applicare Installation Dir>** directory.

Step 2: Restart Applicare if it is running, replacing the license file.

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Applicare Updates

Applicare 9.2 · Updated September 13, 2024 · controller · **Pending technical review**

To check for available updates and download update files, visit the Applicare Updates page located in the **Configuration** section.

Downloading updates requires an Arcturus support login ID with Applicare download access. You can create an ID or request download access by visiting the Arcturus Support Site.

Downloading Updates

Enter your Arcturus support login ID and password, then click the **Check For Updates** button. The **Download** button will become enabled if a new version Applicare is available. Click the **Download** button to begin downloading the updates files, which will be saved in the Applicare/tmp/updates directory.

Performing the Update

After the download is complete, follow the instructions provided in the Version Notes section to complete the update process.

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Deploying the Applicare Service Pack

Applicare 9.2 · Updated September 13, 2024 · controller · **Pending technical review**

Service packs (patches) are minor incremental updates or bug fixes that provide incremental improvements.

Step 1: Create a **backup** of the **applicare.war** file.

Step 2: Stop Applicare server and delete the directory: **<Applicare-Home>/server/webapps/applicare**

Don't delete **<Applicare-Home>/server/webapps/applicare.war**, as this will prevent the service pack from working correctly.

Step 3: create the directory **<Applicare-Home>/sp** if it doesn't already exist. If there are any files in <Arcturus-Home>/sp, delete them.

Step 4: Extract the files from **patch.zip** and copy them to **<Applicare-Home>/sp**

Step 5: Open command prompt in administrator mode and go to the **<Applicare-Home>** path and run the patch using the following command.

java -Darcturus_home=<PATH_TO_APPLICARE_HOME> -jar applicare-patch.jar

Example: java -Darcturus_home=E:\Applicare -jar applicare-patch.jar

The **modification date** of the file **<Applicare-Home>/server/webapps/applicare.war** should be updated.

Step 6: Start the Applicare server.

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Starting the Applicare Controller

Applicare 9.2 · Updated March 12, 2025 · controller · **Pending technical review**

Method 1 - Start the Controller Using Command mode

Open the command prompt as an administrator. Navigate to **Applicare Controller** installation directory then to the path **/server/bin** and run **startup.bat** to start the Applicare Controller.

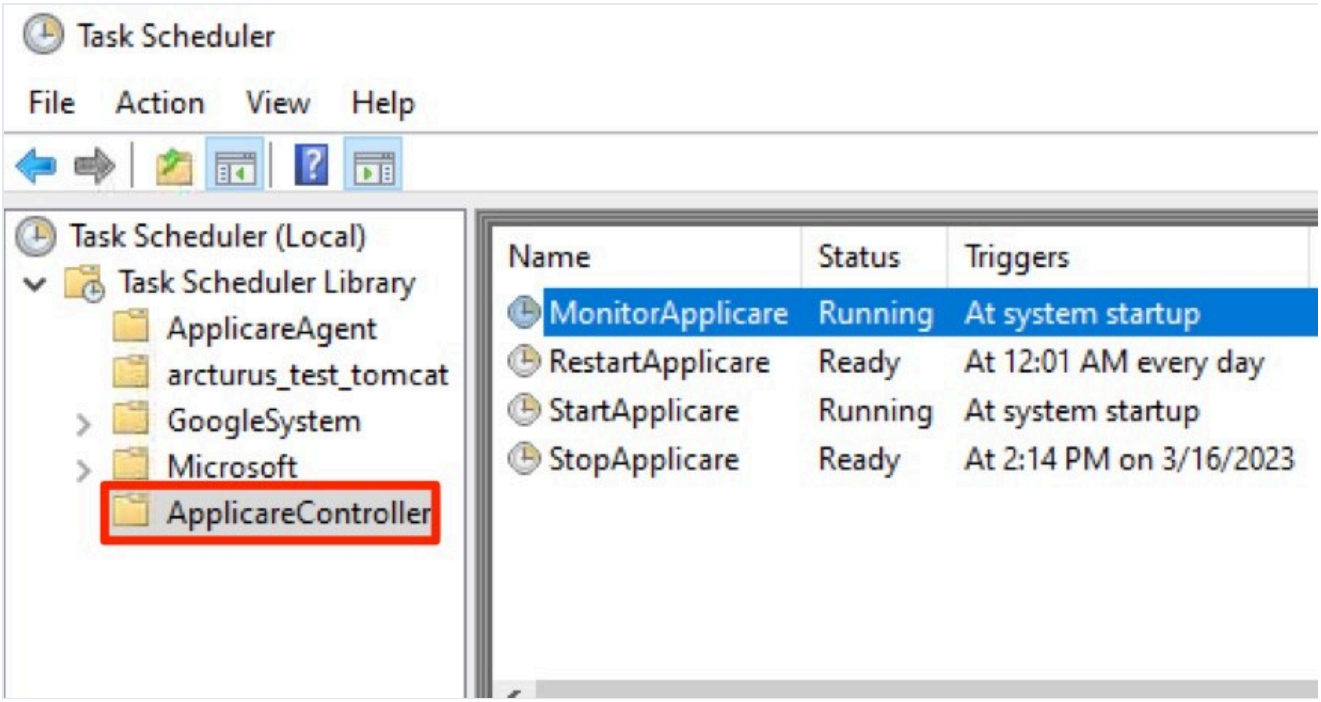


Method 2 - Start the Controller Using Task Scheduler

a. Please open the command prompt as an administrator, navigate to the **Applicare Controller home directory\controllerscripts** directory and run **createControllerTask.bat**

```
D:\Applicare_9281_mysql\controllerscripts>createControllerTask.bat
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareController\StartApplicare" has successfully been created.
SUCCESS: The parameters of scheduled task "\ApplicareController\StartApplicare" have been changed.
SUCCESS: Attempted to run the scheduled task "\ApplicareController\StartApplicare".
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareController\StopApplicare" has successfully been created.
2025-03-12T00:01:01
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareController\RestartApplicare" has successfully been created.
        1 file(s) moved.
SUCCESS: The scheduled task "\ApplicareController\MonitorApplicare" has successfully been created.
SUCCESS: The parameters of scheduled task "\ApplicareController\MonitorApplicare" have been changed.
SUCCESS: Attempted to run the scheduled task "\ApplicareController\MonitorApplicare".
```

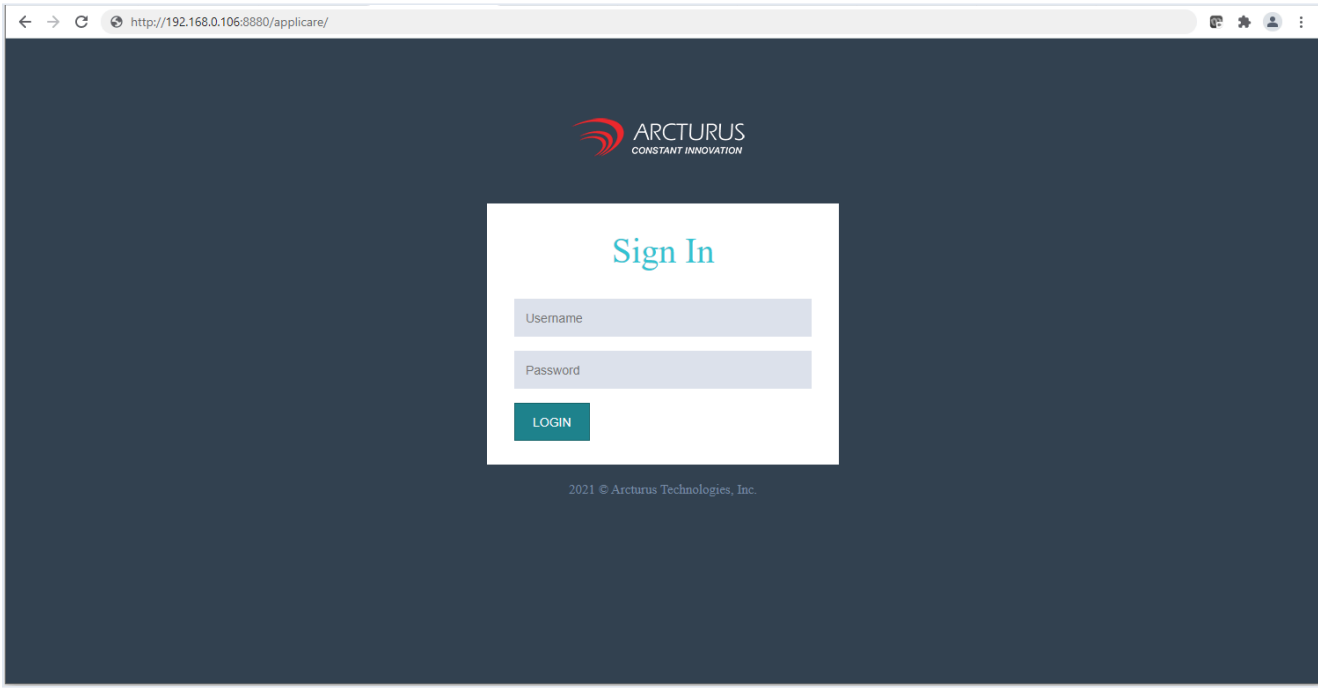
b. The controller will start automatically after executing the script. Open the task scheduler, expand the task scheduler library, and select the ApplicareController folder. Make sure that the task created in it is the same as the one for the controller start, stop, and restart.



Once the controller is running, you can browse the Applicare console using any browser with the IP address and port you chose during installation.

Applicare console URL : **http://<IP>:<PORT>/applicare**

e.g. If the controller IP is 192.168.0.106 and port is 8880 then the Applicare console URL would be: `http://192.168.0.106:8880/applicare`



The default username and password to logging into the Applicare dashboard are both "admin".

Next Steps - Installing SingleAgent

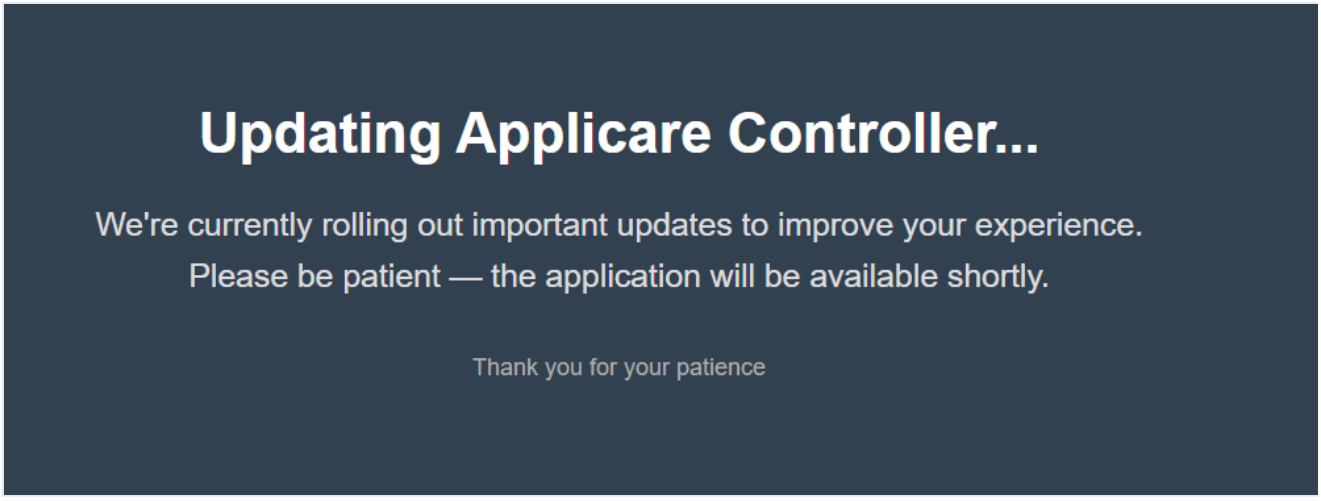
Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Applicare Sentinel

Applicare 9.2 · Updated January 16, 2026 · controller · **Pending technical review**

A new custom error handling mechanism within Sentinel. This feature is designed to replace the standard blank white error page with a branded, user-friendly HTML interface during Controller update cycles, ensuring users are informed of the system's status during scheduled downtime.



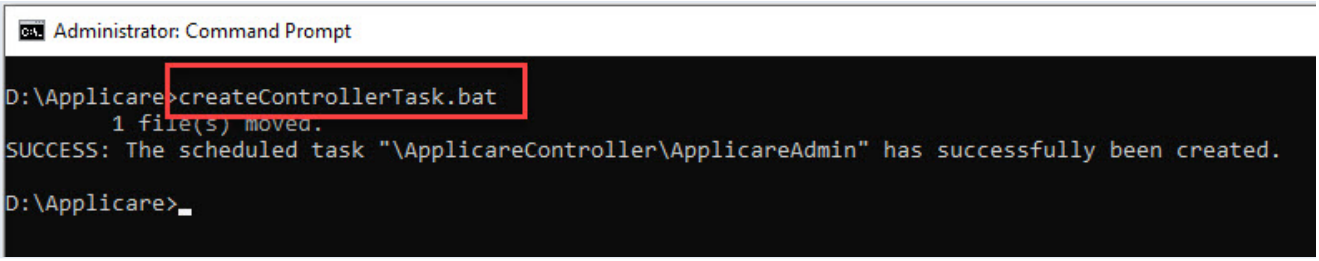
CONFIGURATION GUIDES › INSTALL, START & UPDATE

Set up the Applicare Controller in Task Scheduler

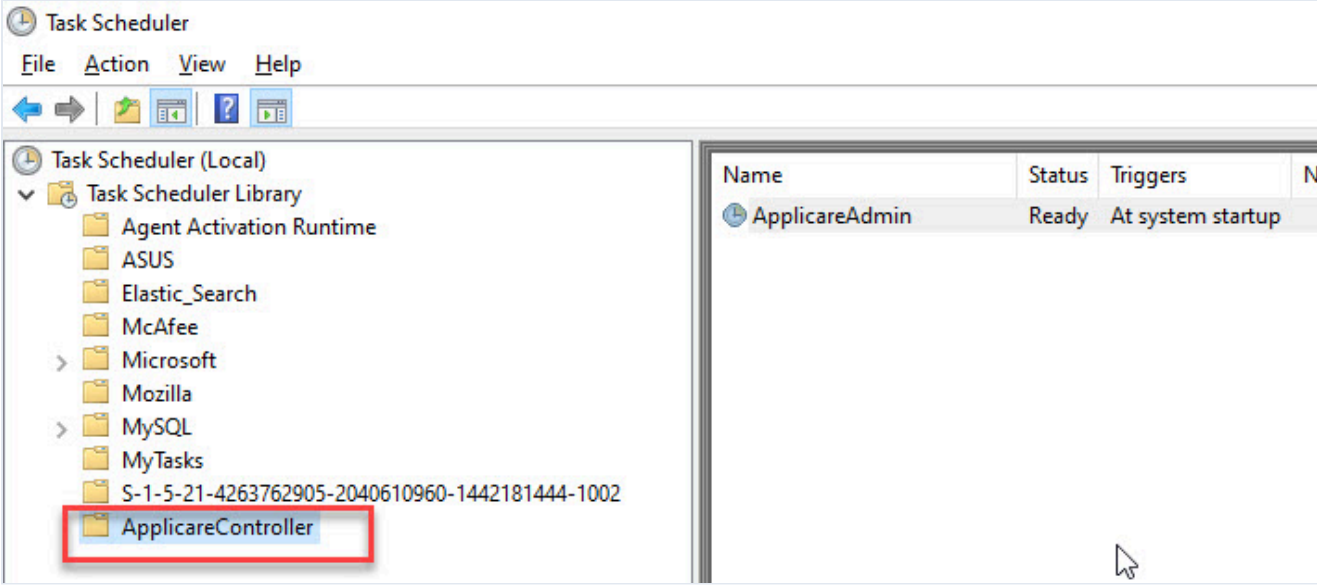
Applicare 9.2 · Updated September 13, 2024 · controller, windows, tasks · **Pending technical review**

Follow the steps for **installing ApplicareController**

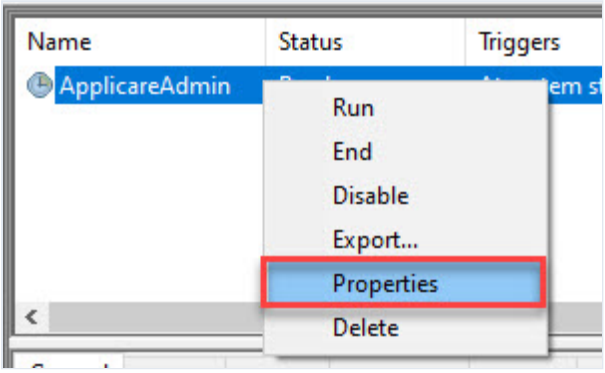
1. Launch the command prompt with administrator rights, go to the **directory** where Applicare controller is installed and run the **createControllerTask.bat** file.



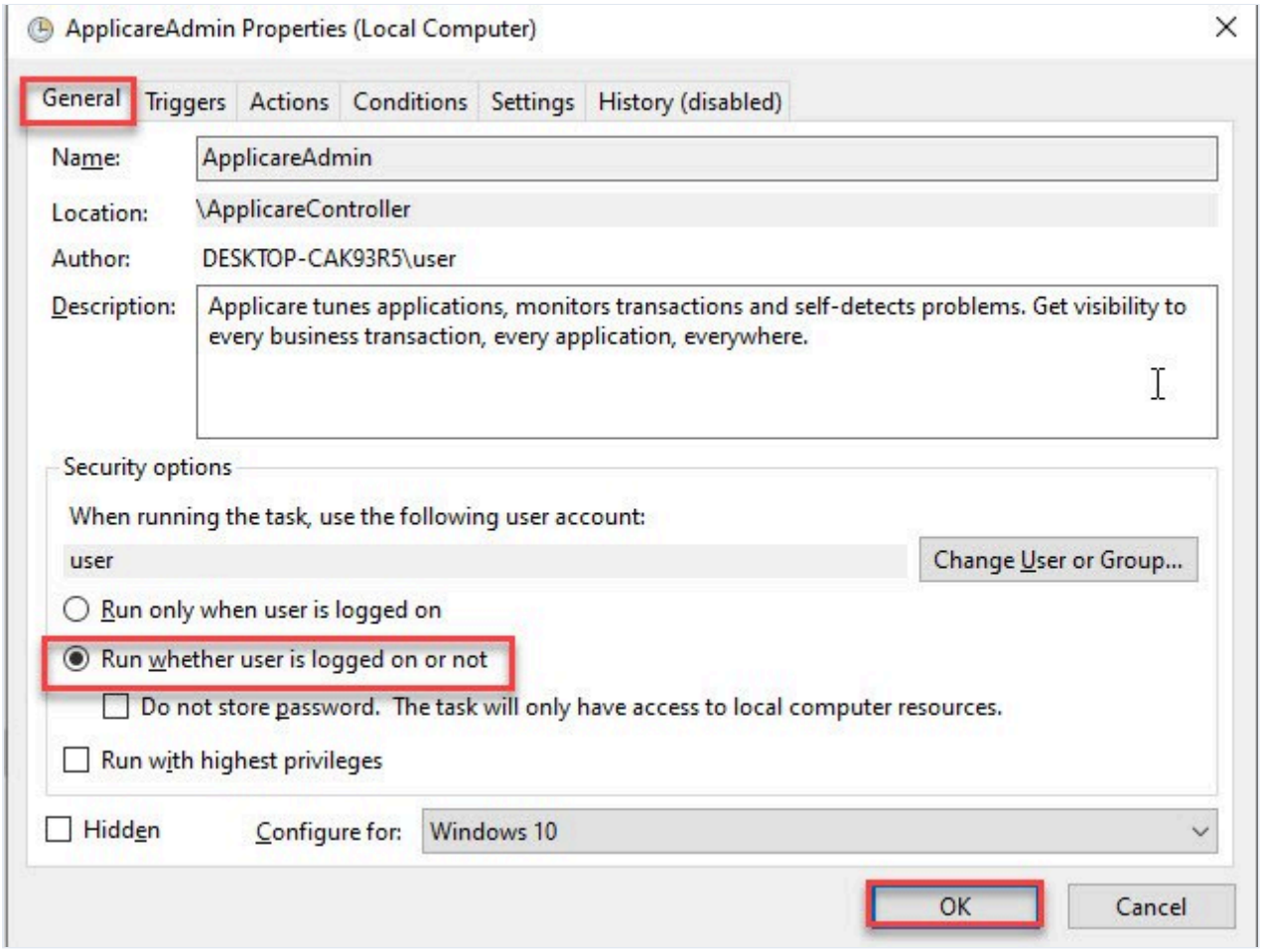
2. Open **Task Scheduler,** expand the **Task Scheduler Library,** and choose the **ApplicareController** folder.



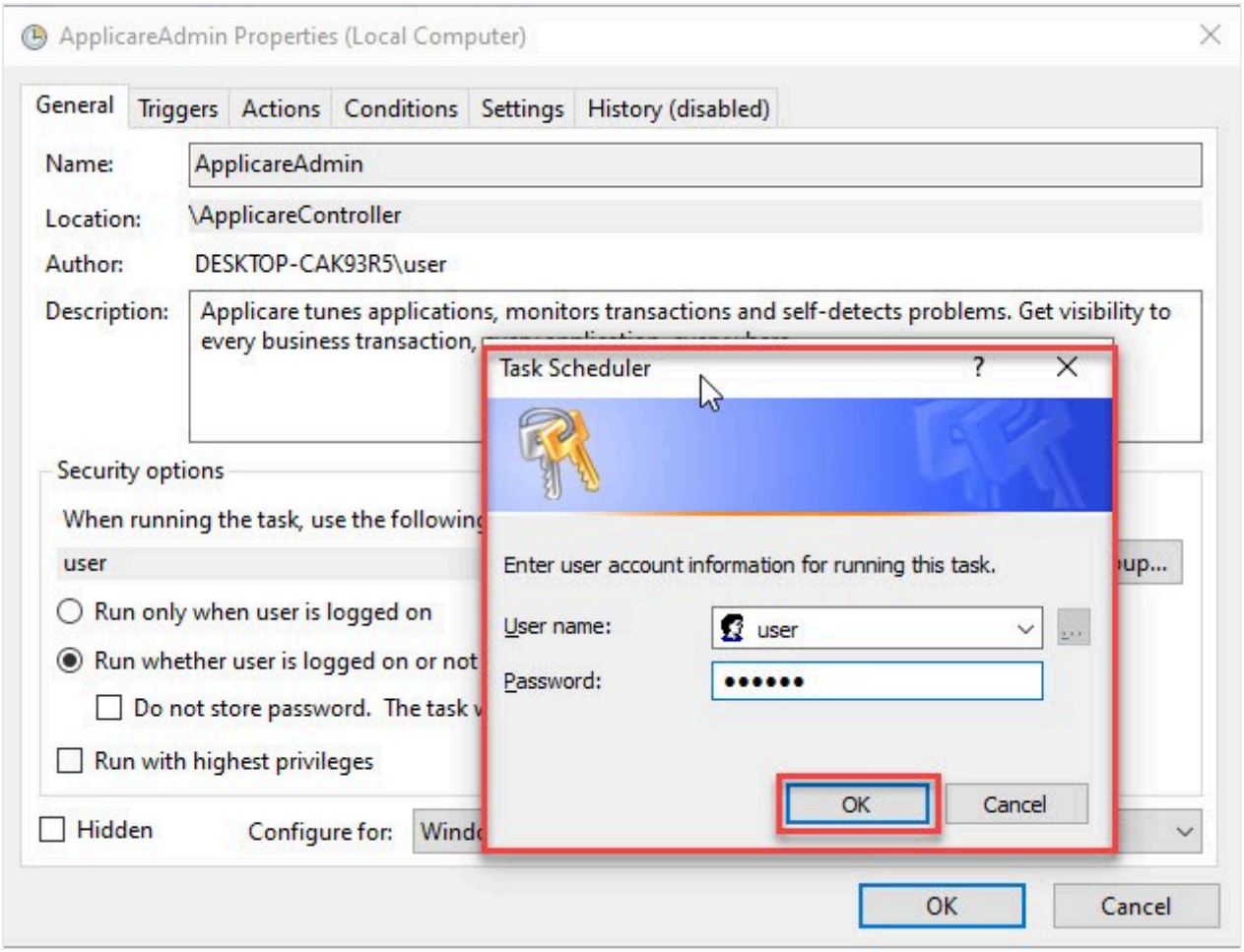
3. Right-click on the **ApplicareAdmin** task and choose **Properties**.



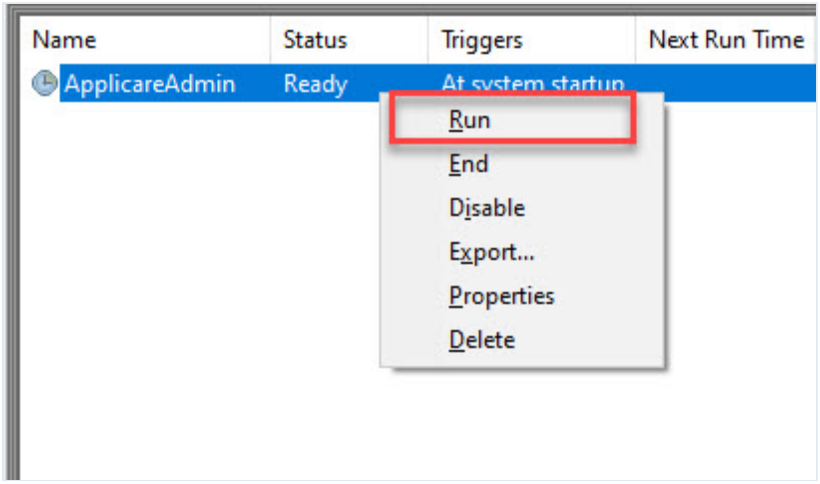
4. Enable **Run whether the user is logged on or not** option and click **OK**.

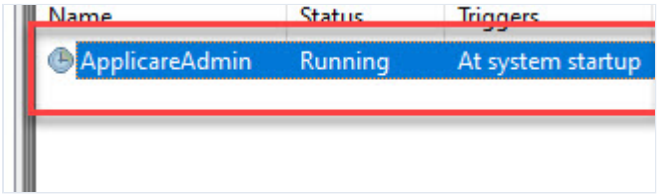


5. Provide the **username** and p**assword,** then click **OK**.



6. Right-click the **ApplicareAdmin** task and choose **Run**.





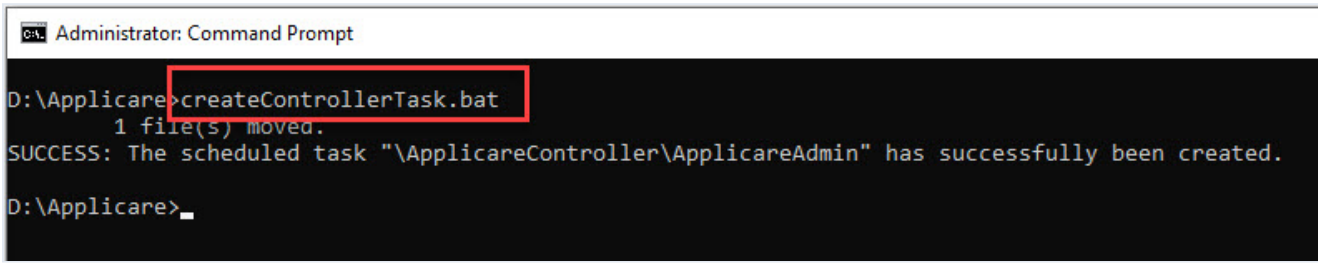
Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

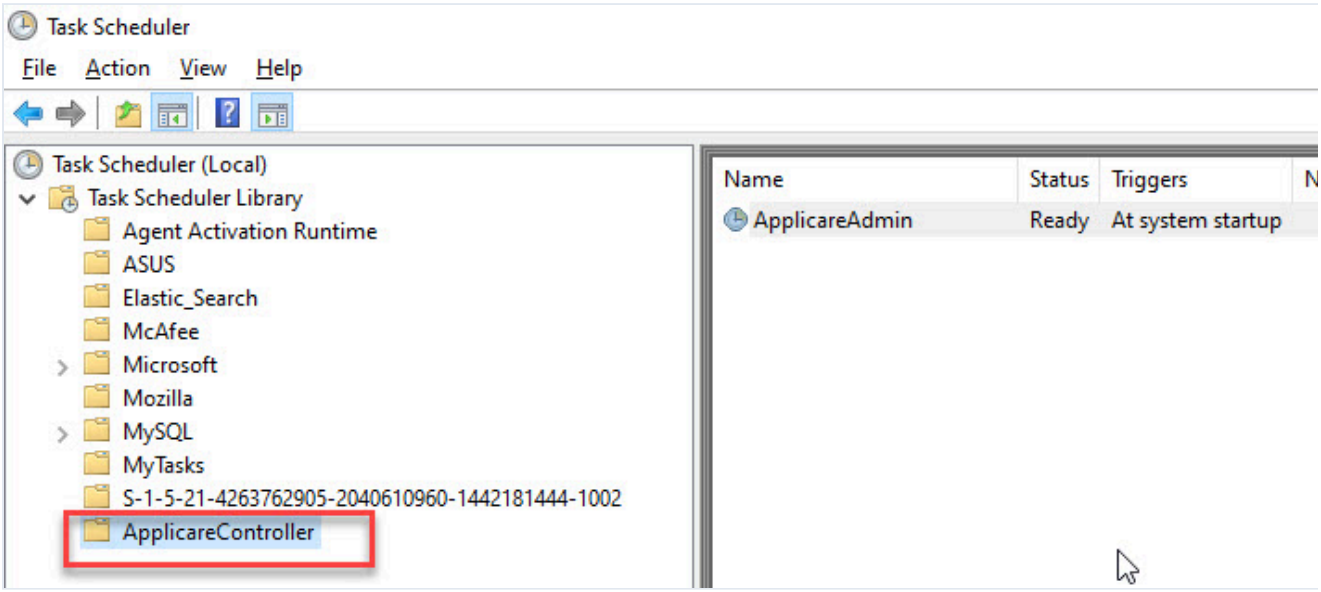
Start the Controller in Task Scheduler

Applicare 9.2 · Updated September 13, 2024 · controller, windows, tasks · **Pending technical review**

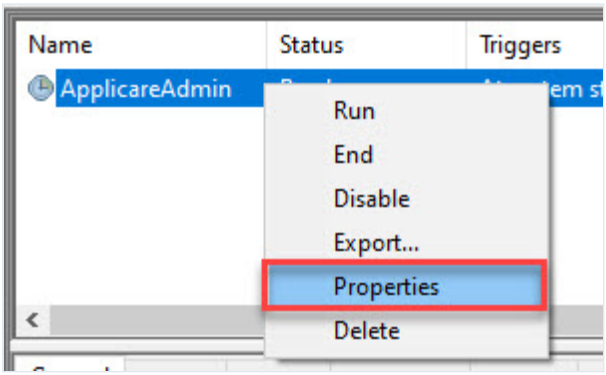
Step 1: Open the command prompt with administrator privileges, and navigate to the **Applicare controller** installation directory, and run the **createControllerTask.bat**. file.



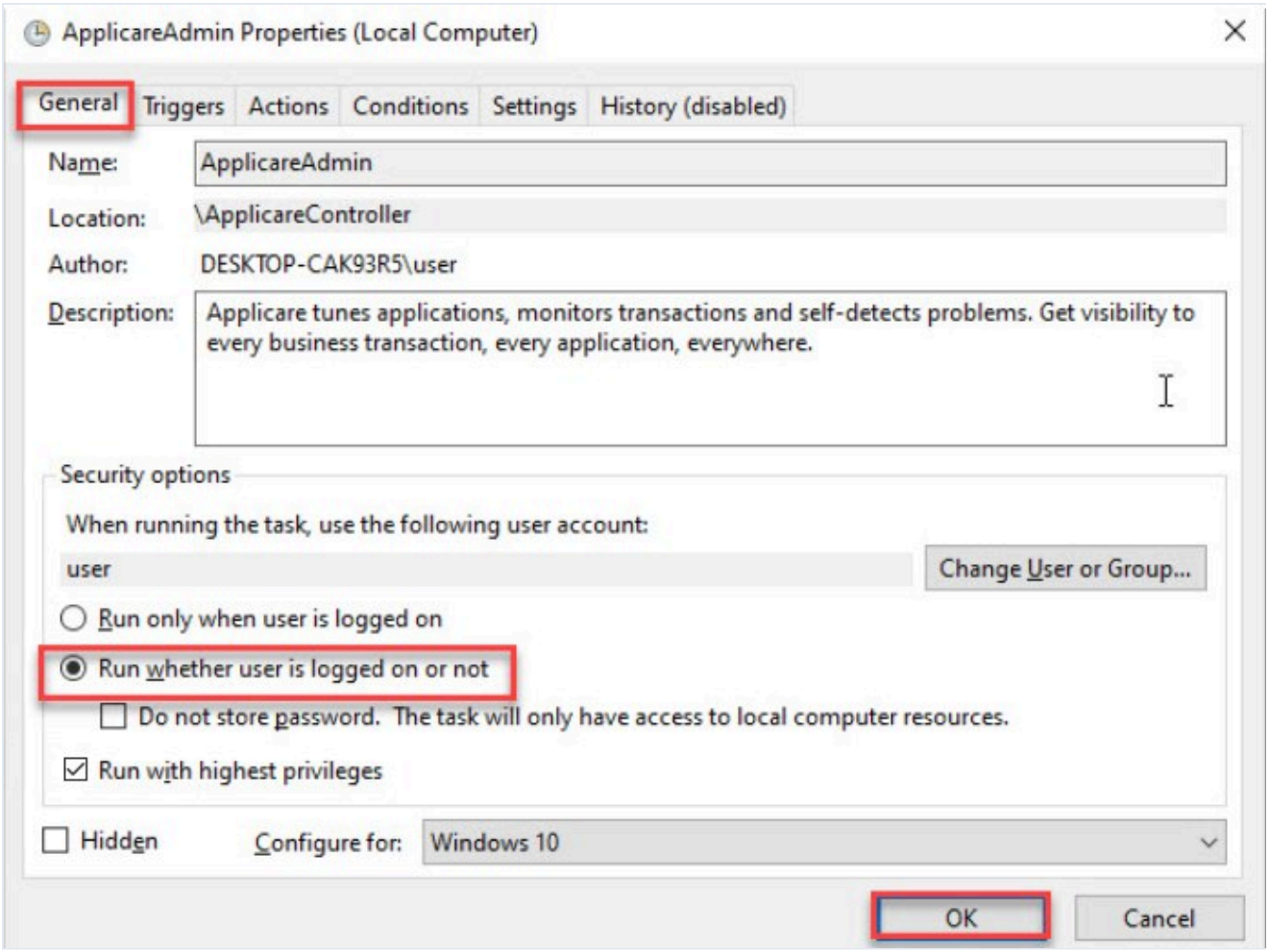
Step 2: Open Task Scheduler, expand the Task Scheduler Library, and select the ApplicareController folder.



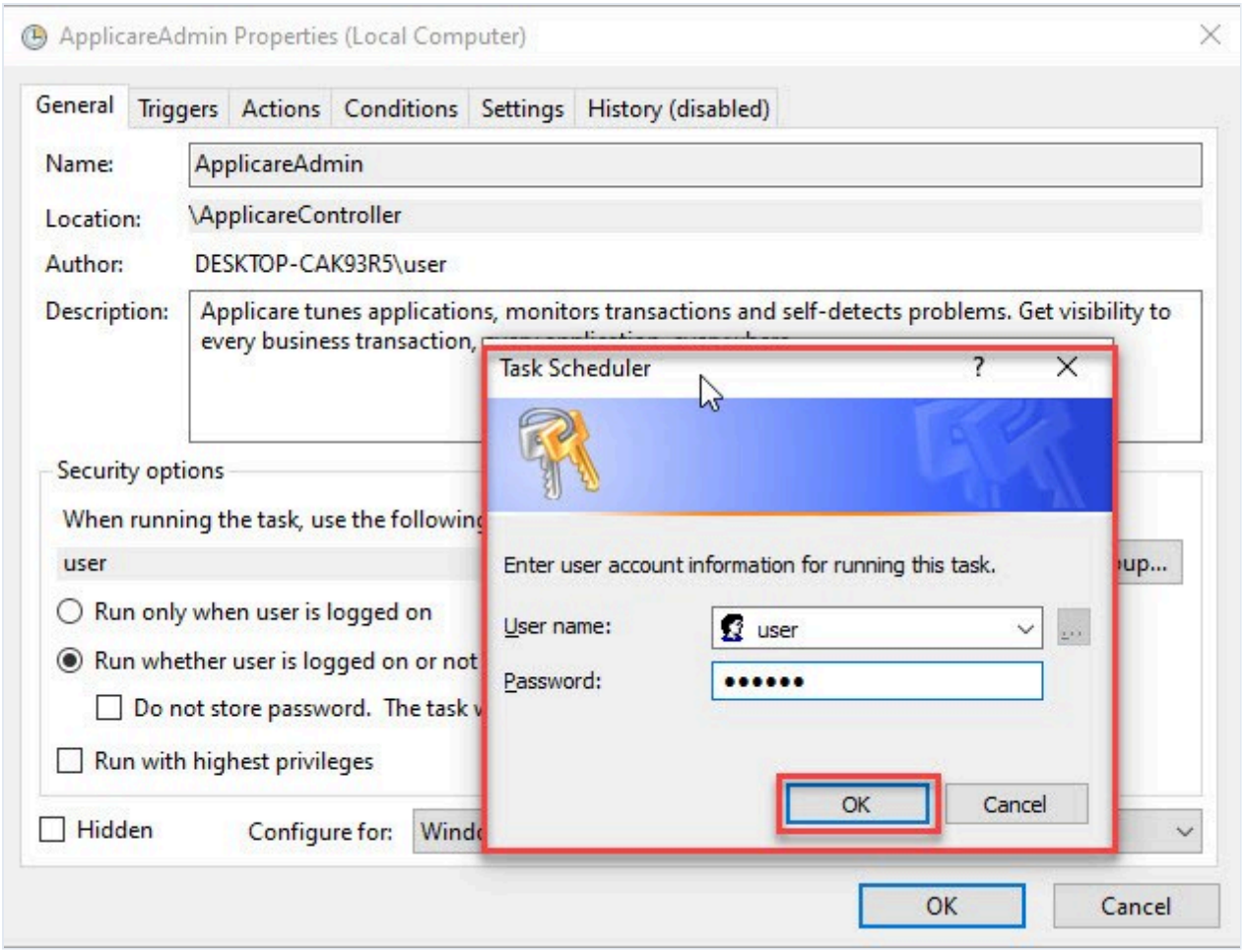
Step 3: Right-click the **ApplicareAdmin** task and choose **Properties**.



Step 4: Select the **Run whether user is logged on or not** option and click **OK**.

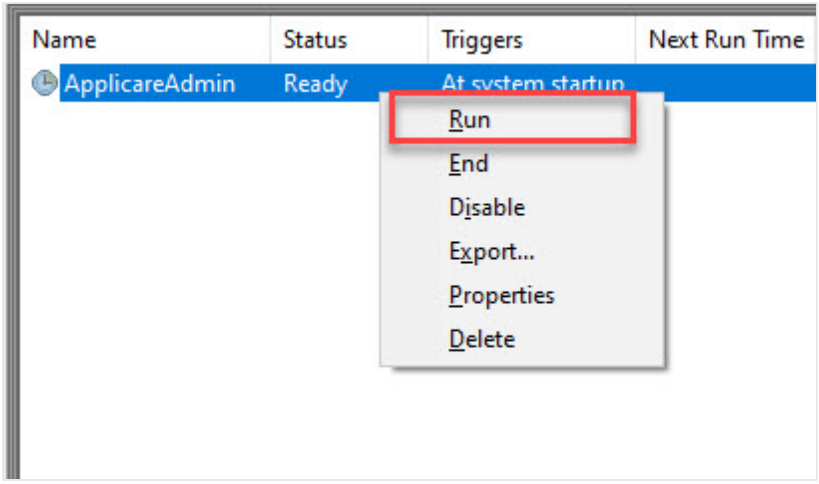


Step 5: Enter the **username** and **password,** then click **OK**.



If the **2147943712** error occurs while saving the password. **Refer here**

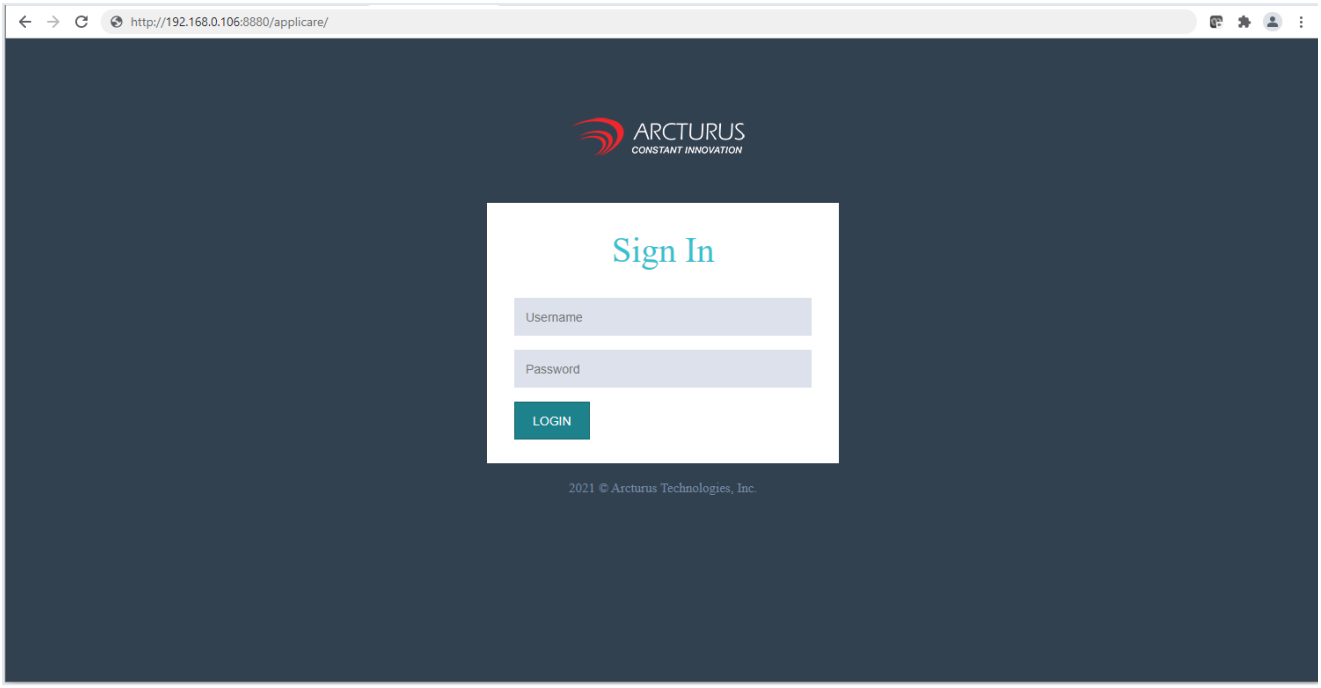
Step 6: Right Click the **ApplicareAdmin** task and select **Run**.



Step 7: After starting the controller, you can access the Applicare console using any browser by entering the IP address and port you specified during the installation.

URL for the Applicare Console : **http://<IP>:<PORT>/applicare**

For the above case Applicare console URL will be: `http://192.168.0.106:8880/applicare`



The default login credentials for the Applicare dashboard are both "admin" for the username and password.

Next Steps - Installing SingleAgent

Monitoring with Applicare

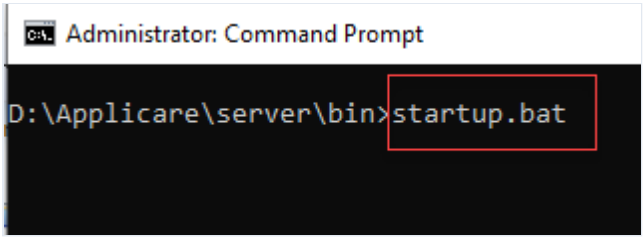
CONFIGURATION GUIDES › INSTALL, START & UPDATE

Start the Controller in command-line mode

Applicare 9.2 · Updated September 13, 2024 · controller, windows, tasks · **Pending technical review**

Step 1: Open a command prompt with **administrator privileges.**

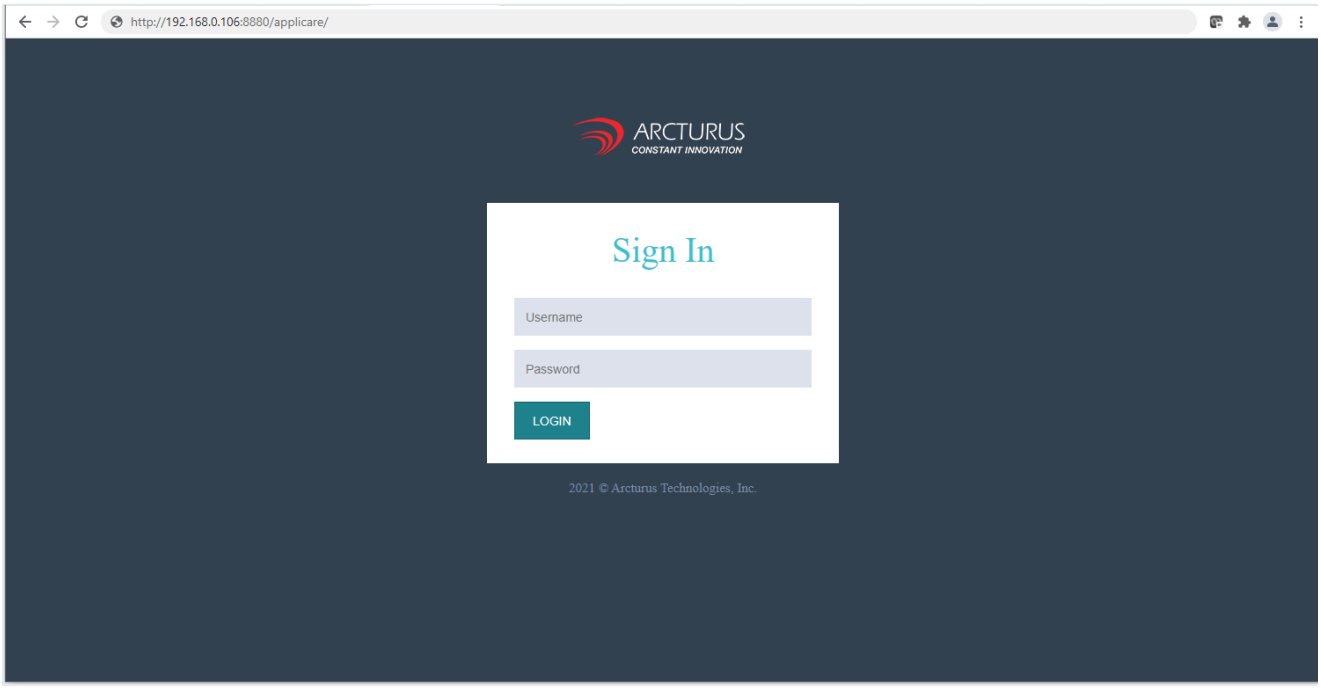
Step 2: Navigate to the directory where the **Applicare controller** is installed, then go to the **/server/bin** path and run **startup.bat** to start the Applicare controller.



Step 3: After the controller is started, you can access the Applicare console in any web browser using the IP address and port specified during installation.

Applicare console URL : **http://<IP>:<PORT>/applicare**

For the above case Applicare console URL would be: `http://192.168.0.106:8880/applicare`



The default username and password for logging into the Applicare dashboard are both "admin".

Next Steps - Installing SingleAgent

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Update Agents

Applicare 9.2 · Updated August 23, 2024 · singleagent · **Pending technical review**

Overview

You can update or restart the active agent in the Applicare controller remotely, without needing physical access to machine running the agent; instead, these tasks can be managed directly from the Applicare console.

- Agent **Auto Update** is only available for **Active** Agents.
- For **Windows,** the agent must be running as a **task scheduler, and the PowerShell version 5 and above.**
- For **Linux,** the agent must be running as a **systemctl service**.
- Agent **Auto Restart** is available only for **Active Standalone Agent.**

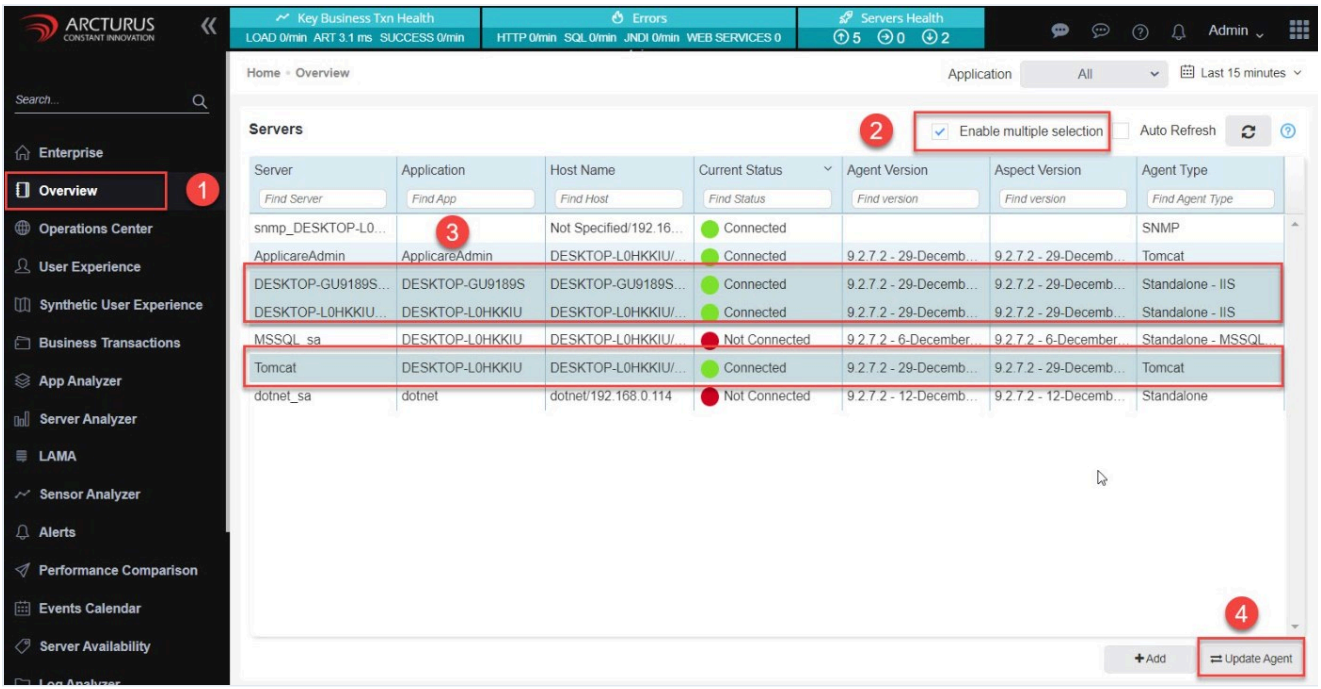
Steps to Update and Restart Agents

Step 1: Navigate to the **Overview** menu.

Step 2: Check the **Enable multiple selection** checkbox to update multiple agents. If updating only one agent you can skip this step.

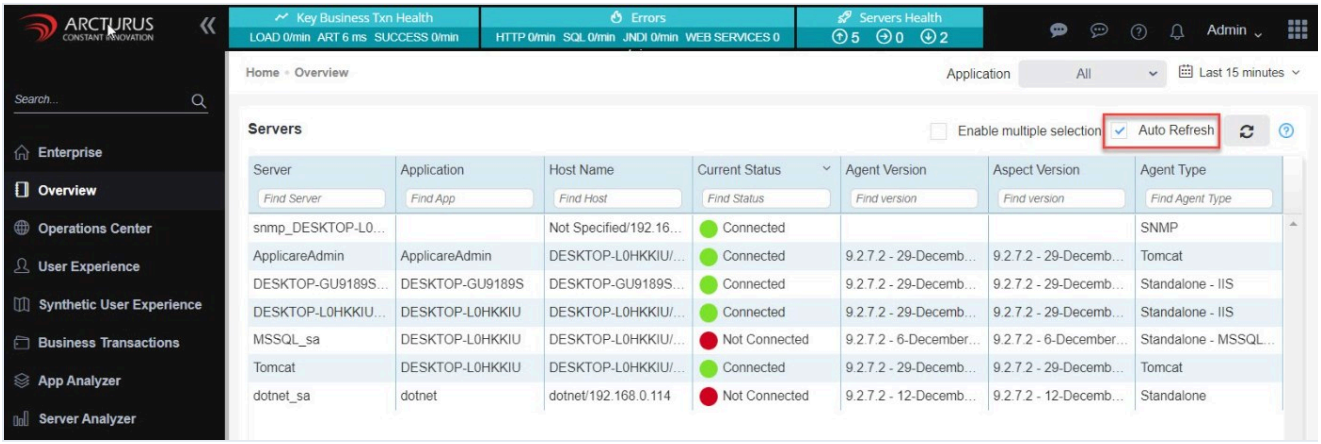
Step 3: Select the agents from the grid.

Step 4: Click the **Update Agent** button to update the agents.

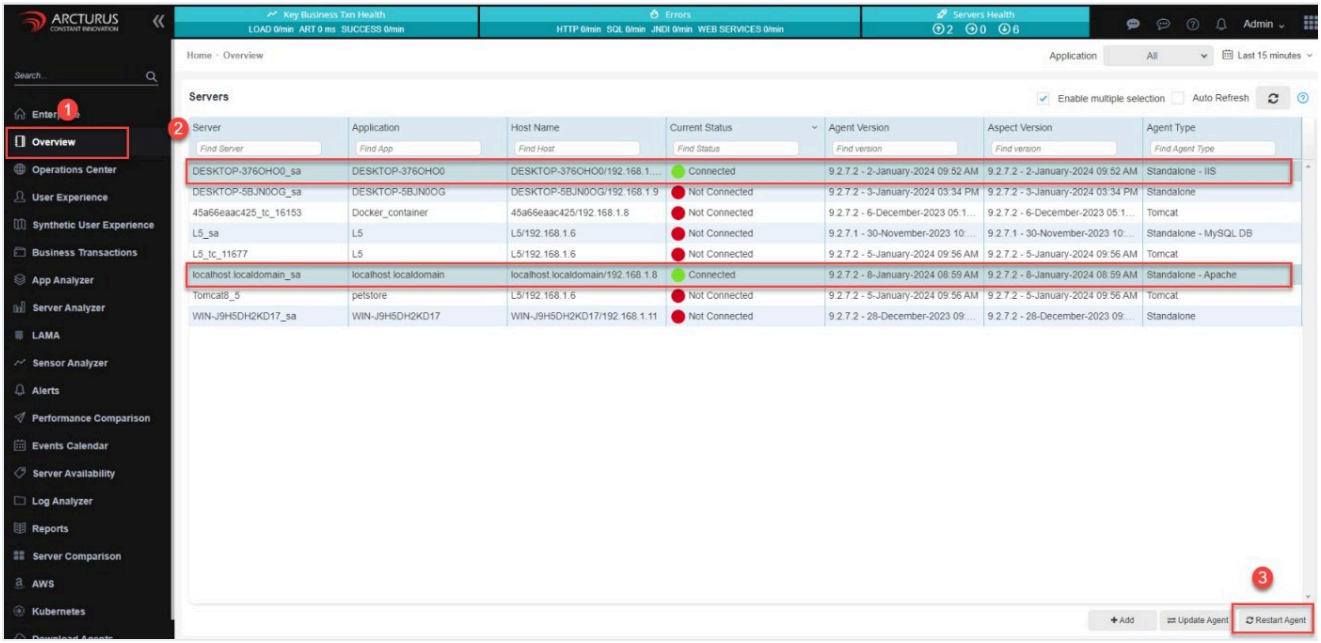


Step 5: The agent update process typically 1 or 2 minutes. After updating, the "Agent" and "Aspect" versions columns will reflect the changes SingleAgent for grid. For other server agents, you may need to manually restart the servers to see the updated versions. Check the **Auto Refresh** checkbox to view the

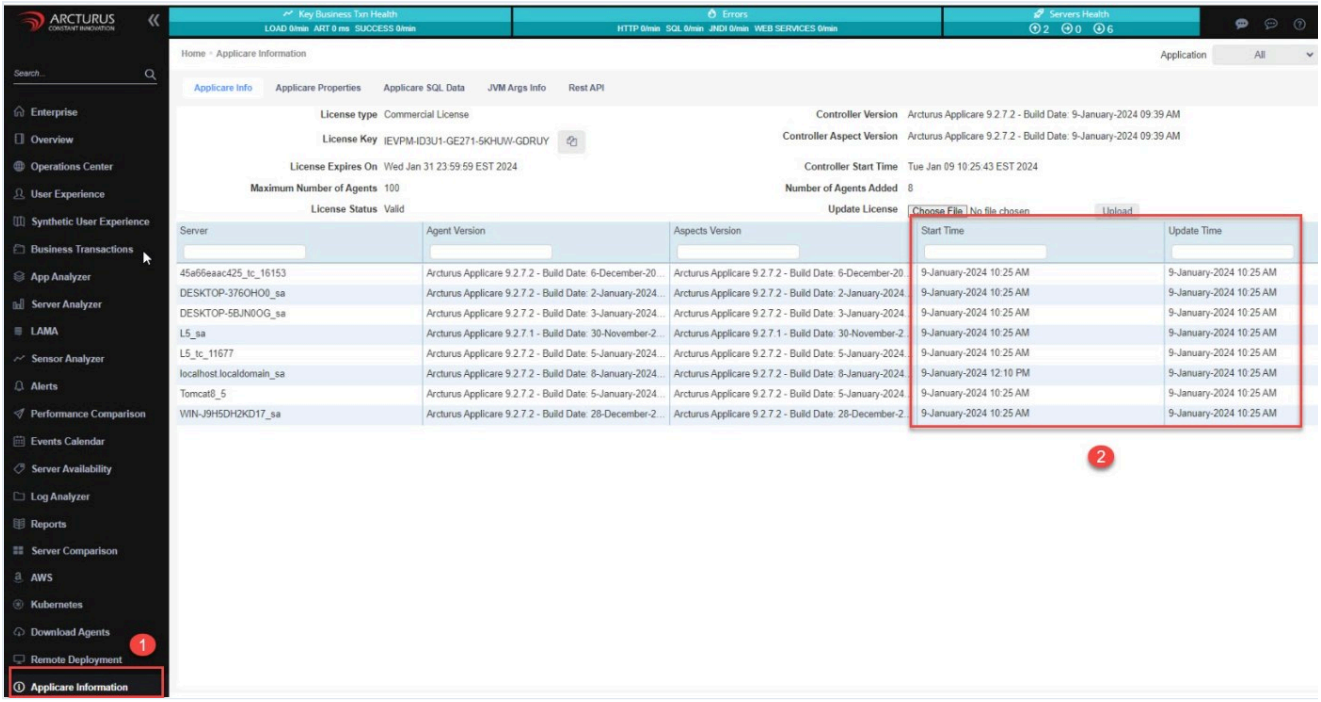
updated Agent and Aspect versions.



Step 6: To **Restart SingleAgent,** select the rows and click the **Restart Agent** button. The agents will restart within few minutes.



Step 7: To verify that the agent is updated and restarted, go to the **Applicare Information menu** and check the **Start Time** and **Update Time columns**.



Configuration for Linux agents:

To enable a specific user to execute `singleagent.service` commands without requiring a password, Adjust sudoers file as follows.

1. open the **sudoers** file with **visudo**

```bash
sudo visudo
```

2. Append the following lines to the end of the sudoers file

<USERNAME> ALL=(ALL:ALL) NOPASSWD: /bin/systemctl start singleagent.service, /bin/systemctl stop singleagent.service, /bin/systemctl restart singleagent.service, /bin/systemctl status singleagent.service

Replace **<USERNAME>** with the actual username.

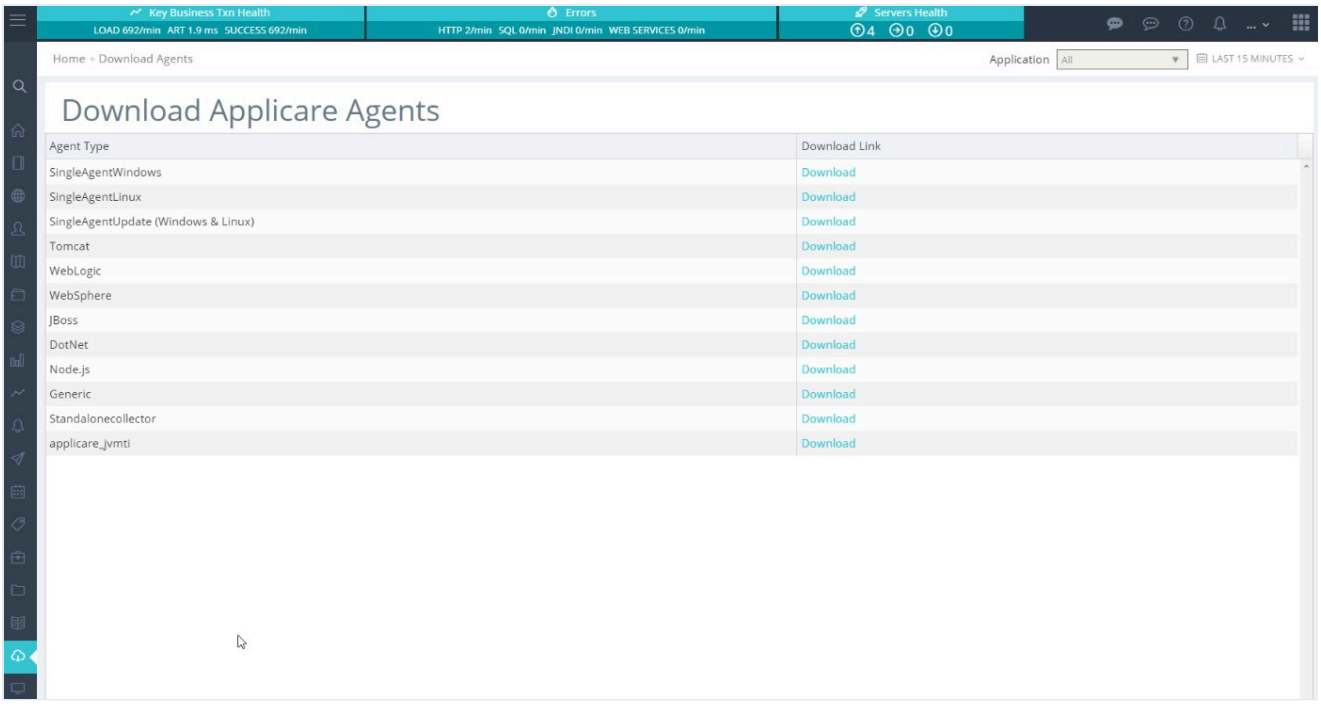
```text
applicare ALL=(ALL:ALL) NOPASSWD: /bin/systemctl start singleagent.service, /bin/systemctl stop singleagent.service, /bin/systemctl restart singleagent.service, /bin/systemctl status singleagent.service
```

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Downloading Applicare Agents

Applicare 9.2 · Updated September 13, 2024 · singleagent · **Pending technical review**

Applicare Console



Using Curl to Download SingleAgent

1. Navigate to the machine where you plan to deploy SingleAgent and run the command provided below. Replace the ControllerProtocol, ControllerIP, ControllerPort and ApplicareSingleAgentName details.

curl **ControllerProtocol**://**ControllerIP**:**ControllerPort**/applicare/standalone?filename=**ApplicareSingleAgentName**.zip >**ApplicareSingleAgentName**.zip

ControllerProtocol -> http (or) https **ControllerIP** -> Applicare Controller running IP **ControllerPort** -> Applicare Controller running port **ApplicareSingleAgentName** -> ApplicareSingleAgentWindows (or) ApplicareSingleAgentLinux (or) ApplicareSingleAgentUpdate

Example:

curl `http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentWindows.zip` >ApplicareSingleAgentWindows.zip curl `http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentLinux.zip` >ApplicareSingleAgentLinux.zip curl `http://127.0.0.1:8880/applicare/standalone?filename=ApplicareSingleAgentUpdate.zip` >Ap

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Installing the SingleAgent on Windows

Applicare 9.2 · Updated February 6, 2025 · singleagent · **Pending technical review**

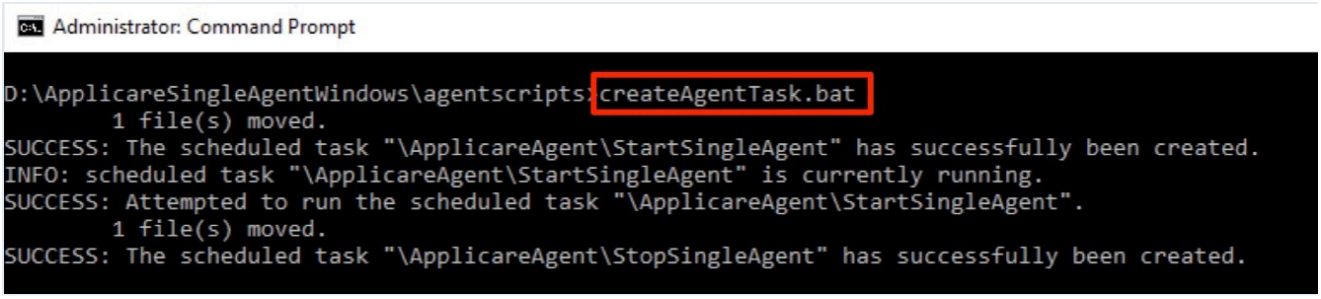
Installation Steps for Windows

1. Login into **Applicare console** and navigate to the **Download Agents** screen.

2. Click the **SingleAgentWindows** button to download the agent. This will download the **ApplicareSingleAgentWindows.zip** file. Copy **ApplicareSingleAgentWindows.zip** file to the machine where the agent will be deployed.

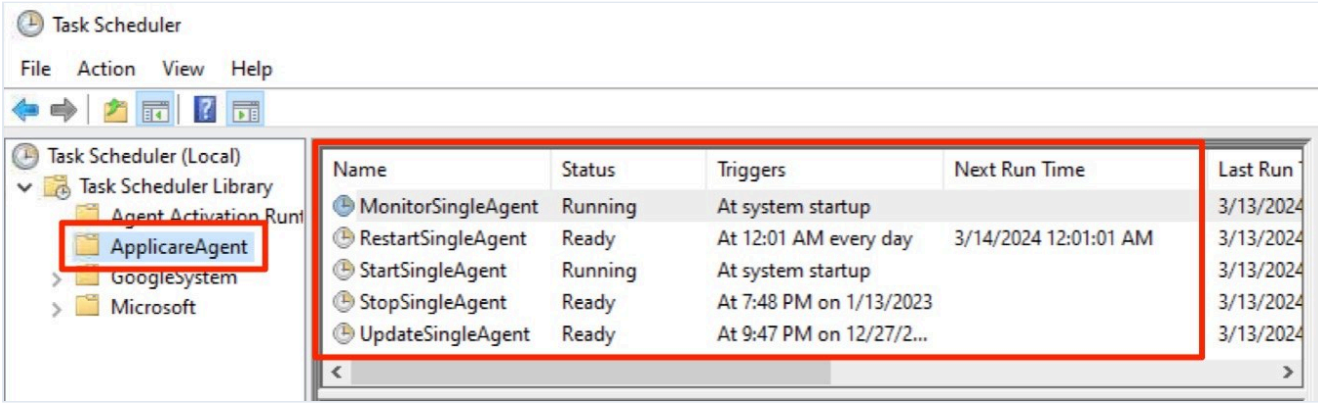
3. Extract the contents of **ApplicareSingleAgentWindows.zip** file into a directory named **ApplicareSingleAgentWindows**.

4. Open a command prompt in administrator mode, navigate to the **ApplicareSingleAgentWindows\agentscripts** directory and run the **createAgentTask.bat** file.



The SingleAgent will be installed as a task and will run by default. To view the task, open **Task Scheduler,** expand **Task Scheduler Library** and select the **ApplicareAgent** folder. You will see two tasks will be listed there.

- **SingleAgent -** Run this task to **start** the SingleAgent. After installation, the agent will run by default.
- **StopSingleAgent -** Run this task to **stop** the SingleAgent.



For Other Docs - Click here

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Manual Installation of Applicare SingleAgent on Linux

Applicare 9.2 · Updated December 17, 2024 · singleagent · **Pending technical review**

For a quick installation of SingleAgent, use the CURL or WGET commands from the Download Agents page. If CURL or WGET is not available, follow the instructions below.

1. Login into **Applicare console,** navigate to **Download Agents** screen and download **SingleAgentLinux.** Copy **ApplicareSingleAgentLinux.zip** file to the machine where the agent will be deployed.

2. Create a directory named "**ApplicareSingleAgent**" on the agent machine and extract the contents of **ApplicareSingleAgentLinux.zip** into that directory.

3. Set **chmod -R 755** permission to that **ApplicareSingleAgent** directory.

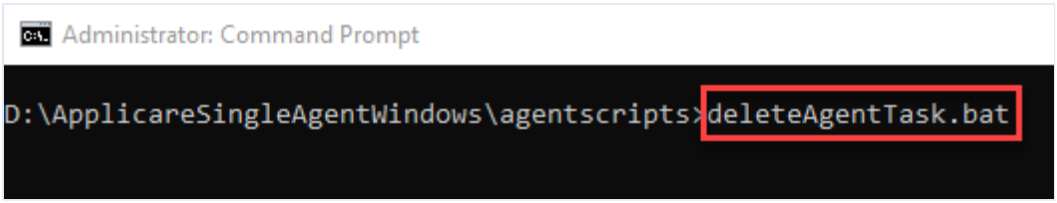
Refer here to start SingleAgent.

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Uninstalling an agent and scheduled task on Windows

Applicare 9.2 · Updated August 22, 2024 · singleagent · **Pending technical review**

Step 1: Open the Command Prompt in administrator mode, navigate to the `ApplicareSingleAgentWindows\agentscripts` directory, and execute the `deleteAgentTask.bat` file.



Step 2: After deleting the task, close the Command Prompt and delete the `ApplicareSingleAgent` directory.

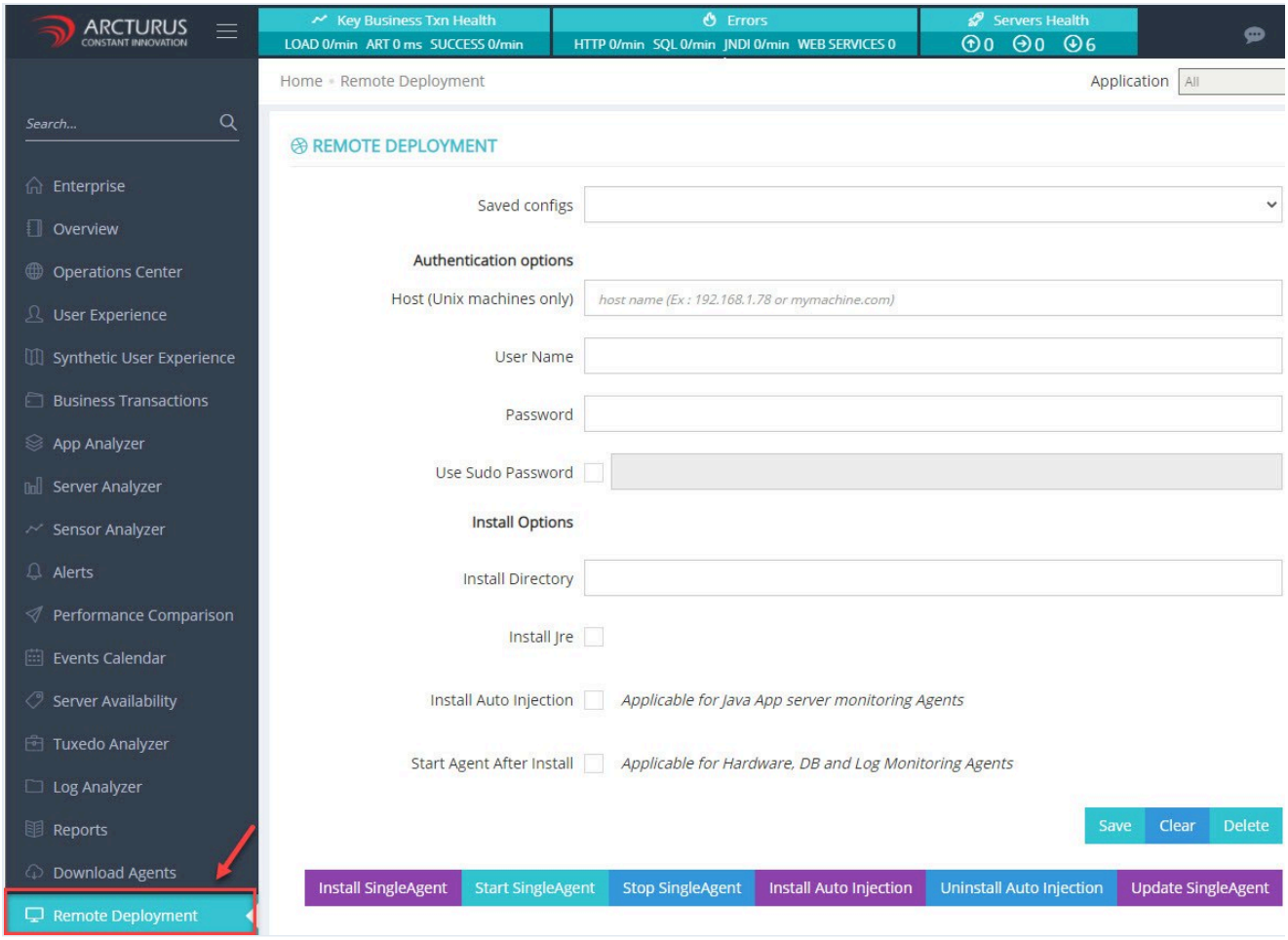
CONFIGURATION GUIDES › INSTALL, START & UPDATE

Remote Deployment for Applicare

Applicare 9.2 · Updated September 13, 2024 · singleagent · **Pending technical review**

Linux

1. Navigate to the **Remote Deployment** menu.



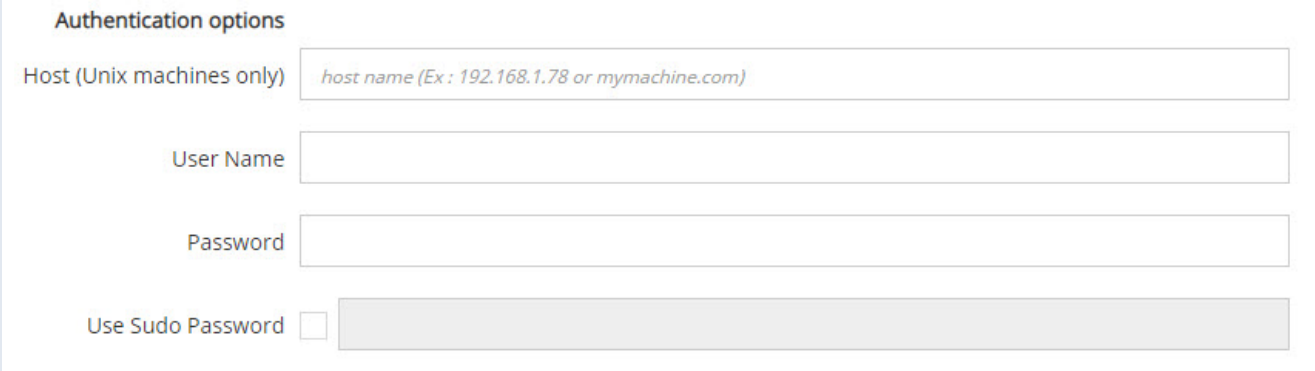
2. Provide the configuration details for the machine where the agent will be deployed:

Host (Unix machines only) - Enter the hostname or IP address of the machine.

User Name - Provide the user name for login.

Password - Enter the login user password.

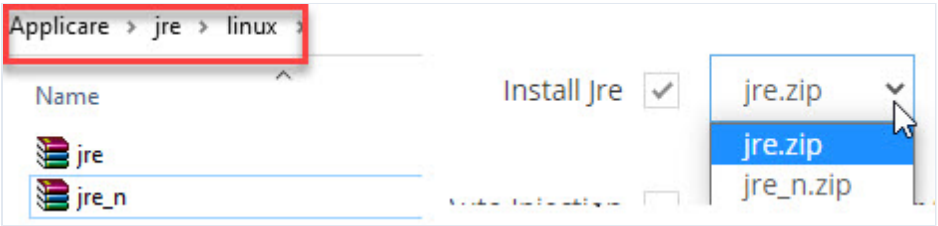
Use Sudo Password - Check this option if the login user requires elevated privileges through sudo to perform actions.



Install directory - Specify the location where the agent will be deployed.

Install JRE - Enable this option if the machine does not have the java installed. If this option is enabled agent will have its inbuilt JRE. Applicable for hardware, database and log monitoring agents.

Applicare Installer will have the JRE inbuilt by default in the location **<ApplicareInstallerDirectory/jre/linux>.** If we are going to use different JRE then zip the needed JRE folder and give a name to that new jre zip file and place it in the location **<ApplicareInstallerDirectory/jre/linux>.** So all the JRE's will be displayed in the JRE dropdown in the remote deployment screen. We can select the needed JRE and install the agent.



To zip custom JRE

Install the JRE based on the operating system and zip JRE directory.

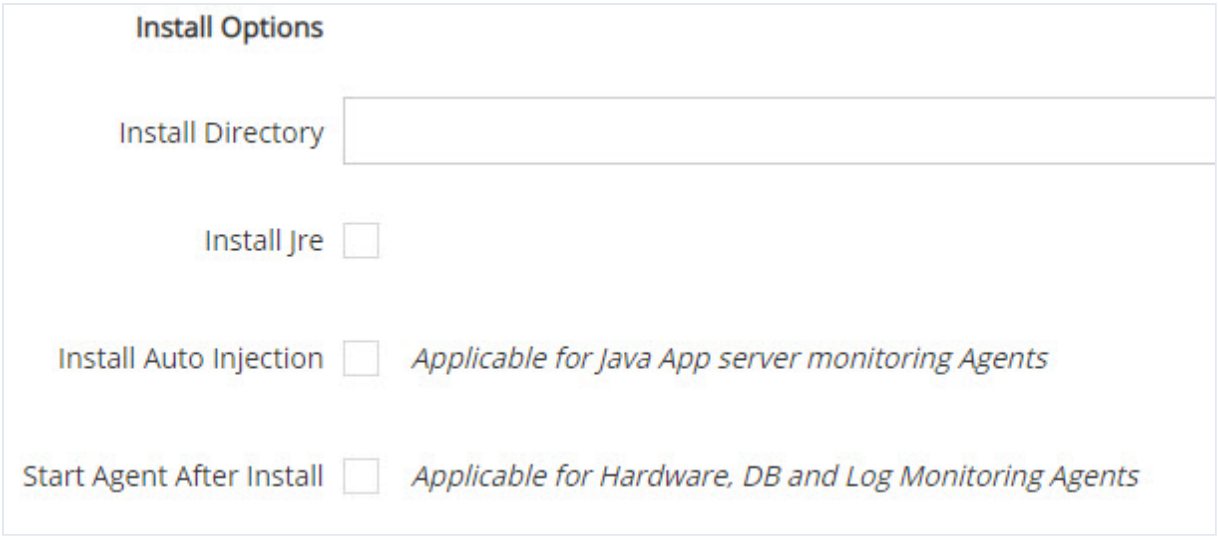
Command to zip JRE directory

```
TEXT
zip –r <zipname>.zip <zip_directory>
```

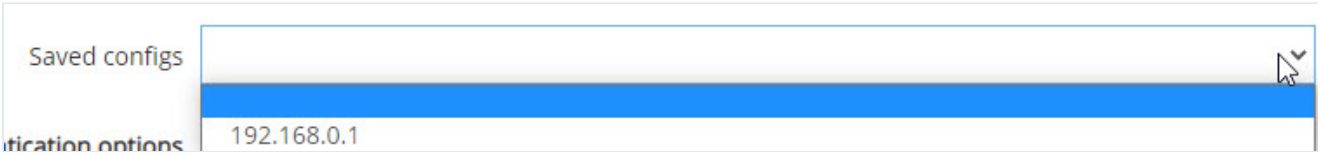
e.g. To zip **jre1.8 directory** which is available in the path **/dev/home** zip -r **jre18**.zip **/dev/home/jre1.8**

Install Auto Injection - Select this option to install the Applicare wrapper on the configured machine, which will automatically inject the SingleAgent startup scripts while starting the java app servers. Enable this if you plan to monitor Java app server agents.

Start Agent After Install - It will start the SingleAgent immediately after installing the agent. Enable this option if we are going to monitor hardware, database and log monitoring agents.



Saved configs - This dropdown will have all the saved configuration details.

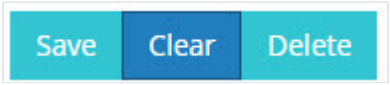


Button Functionalities

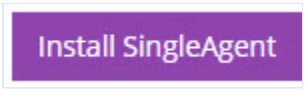
Save - It will save the machine configuration details.

Clear - It will clear the fields on the screen.

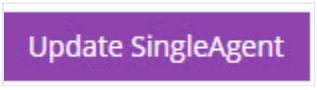
Delete - It will remove the previously saved configuration details.



Install SingleAgent - This will set up Install SingleAgent on the configured machine.



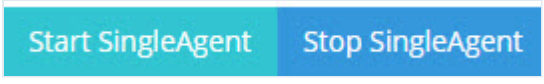
Update SingleAgent - This will update the SingleAgent on the configured machine.



Applicable to hardware, database, and log monitoring agents:

Start SingleAgent - It will Start SingleAgent on the configured machine.

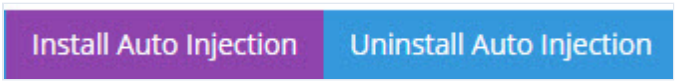
Stop SingleAgent - It will Stop SingleAgent on the configured machine.



Applicable for Java app server agents

Install Auto Injection - It will install the wrapper on the configured machine to auto inject the SingleAgent startup scripts while starting the Java app servers.

Uninstall Auto Injection - It will uninstall the wrapper in the configured machine.



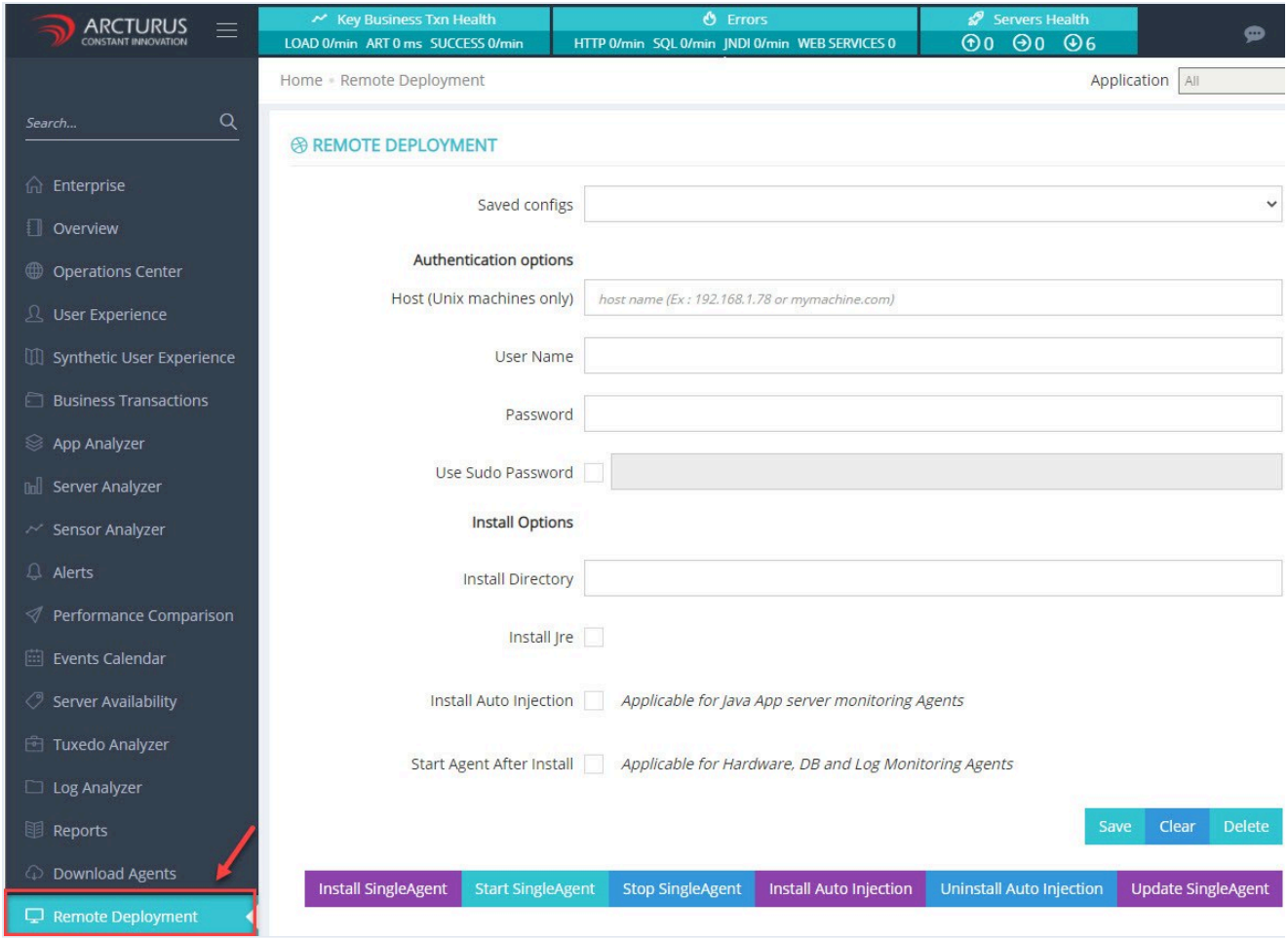
Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Deploying the agent on Linux and Windows via the Remote Deployment feature

Applicare 9.2 · Updated September 13, 2024 · singleagent · **Pending technical review**

Step 1: Log in to Applicare controller and navigate to the **Remote Deployment** screen on left side menu.



Step 2: Enter the configuration details of the machine where the agent will be deployed.

Host (Unix machines only) - Enter the hostname or IP address of the machine.

User Name - Enter the login user name.

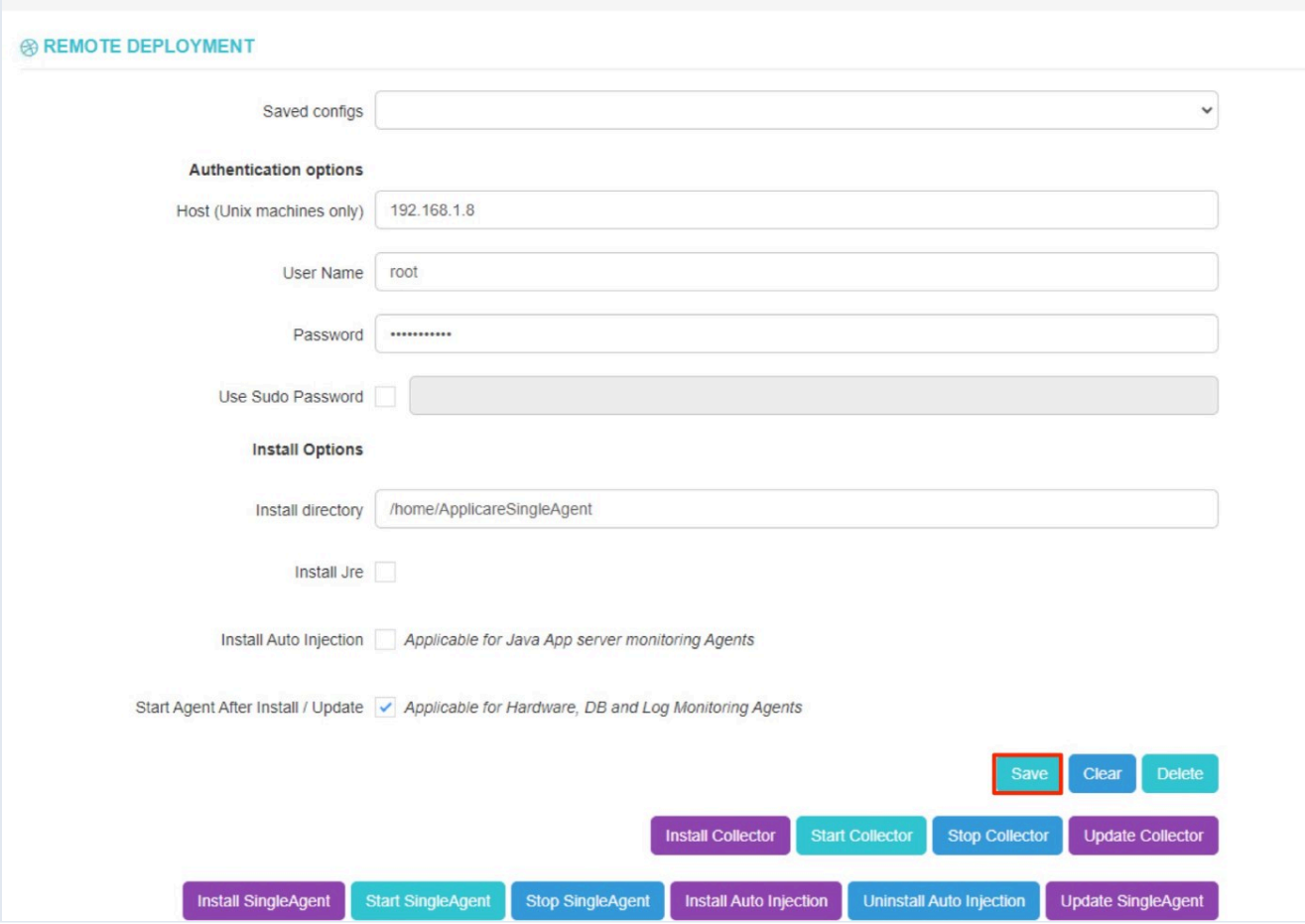
Password - Enter the login user password.

Use Sudo Password - Enable this option if the login user does not have root permission.

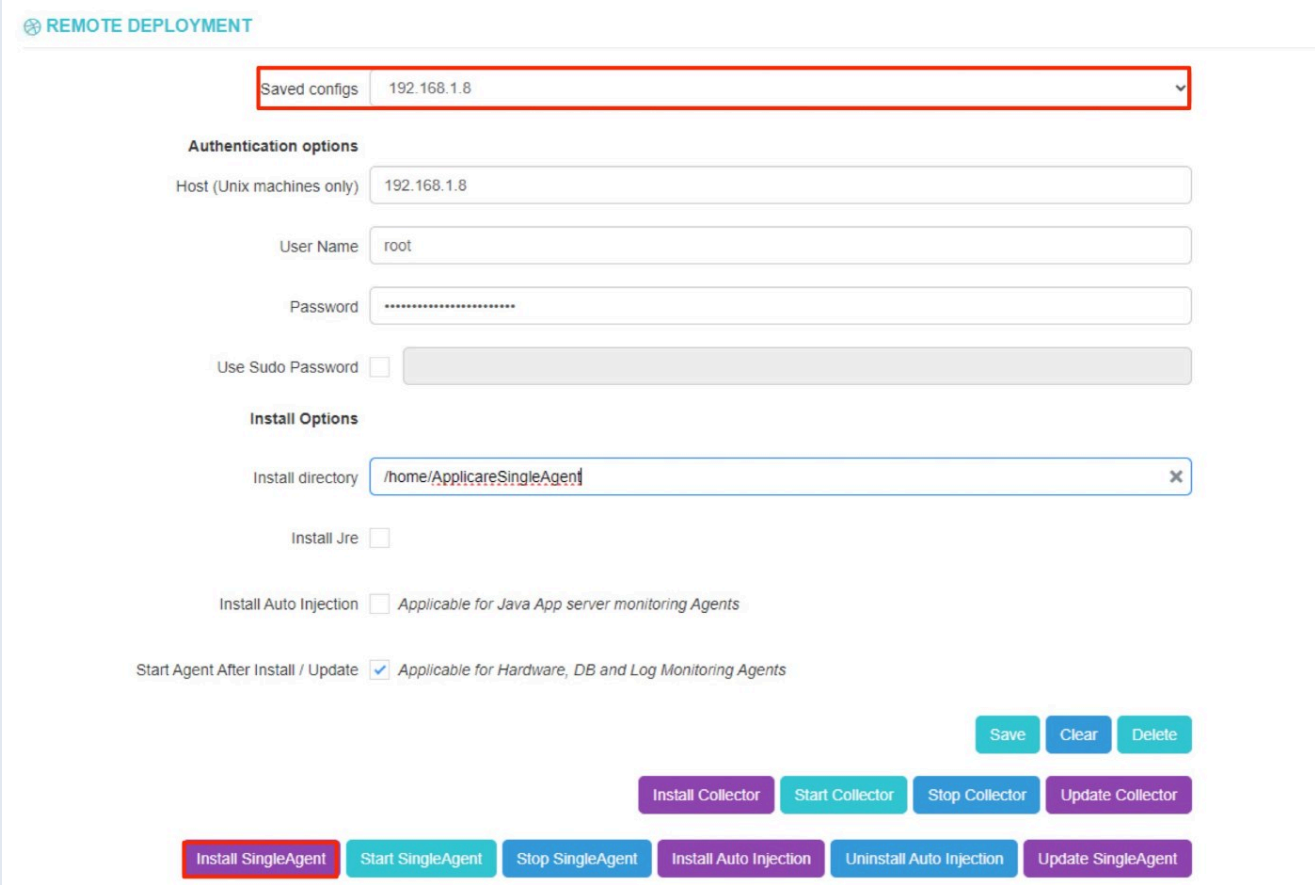
Install directory - Specify the location where the agent will be deployed.

Start Agent After Install - Enable this option to start the agent immediately after installation, particularly if you plan to monitor hardware, database and logs.

Step 3: After entering all the configuration details click the **save** button.



Step 4: Save the configuration, select the machine from the **saved config** dropdown. Then click on **Install SingleAgent** button.



Follow these steps only if you need to update the agent.

Step 1: Go to the Remote Deployment screen from the left-side menu. **Step 2:** Select the machine where the agent is running from the **saved config** dropdown, and then click the **Update SingleAgent** button.

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Standalone Agent for BTC Collector

Applicare 9.2 · Updated September 13, 2024 · singleagent · **Pending technical review**

Steps to Configure the Standalone Agent as a BTC Collector

1. Install the Standalone Agent.

2. Add the JVM argument **-Dapplicare.btx.collector=true** to the Standalone Agent startup configuration. To prevent the automatic addition of the collector agent to the controller add the JVM argument **-Dapplicare.disable.autoadd=true** in the Standalone Agent startup.Then, start the Agent.

3. Log in to Applicare Console and navigate to the **"Applicare Advanced Options"** screen.

4. Enable **Push** option in the **Txn data persistence** setting.

5. Add the Standalone Agent URL in the **Txn Data Persistence URL.**

6. Restart **Applicare Controller and Agents**.

7. Add the **BTC Collector** as one of the **Agent in Applicare Controller.**

Steps to enable the Queue Mechanism for BTC Collector

- By default, the queue mechanism is not be enabled in the BTC Collector.

1. Stop both the **BTC Collector** and **Agent**.

2. Add the following **JVM arguments** to enable the **queue mechanism:**

- Dapplicare.enable.intellitrace.queue=true // To enable intellitrace queue
- Dapplicare.enable.btm.queue=true // To enable BTM queue

3. Start the **BTC Collector** and the **Agent**.

Custom Startup Arguments to Add in Collector

To specify the number of threads, use the following argument:

- Dapplicare.intellitrace.queue.collector.threadcount=No.of.Threads // Intellitrace Data collector Thread count
- Dapplicare.btm.queue.collector.threadcount =No.of.Threads // BTM Data collector Thread count

To set the thread sleep time, Use the following argument

- Dapplicare.intellitrace.queue.collector.threadsleep=time // Intellitrace Data collector Thread sleep time in milliseconds
- Dapplicare.btm.queue.collector.threadsleep=time // BTM Data collector Thread sleep time in milliseconds

To enable logging of messages, use the following argument:

- Dapplicare.enable.queue.log=true

The Default thread-count is 1 for intellitrace and 1 for BTM with 1 second sleep time.

Docs home

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Applicare Agent's Impact on Application Performance

Applicare 9.2 · Updated October 9, 2024 · singleagent · **Pending technical review**

The Applicare agent imposes very minimal overhead on application performance, with the impact being mostly negligible and almost undetectable. By default, Applicare only instruments user application packages and namespaces, ensuring efficient monitoring without compromising performance.

We advise against installing the Applicare agent on app/web servers if CPU or memory utilization is exceeds 95% in the machine.

Following are few load test results executed at our labs.

Without agent

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
petstore_item	956	7	1	128	6.15	0.00%	47.6/min	3.42	4420.0
category	956	10	2	33	3.72	0.00%	47.6/min	3.28	4232.0
product	956	8	2	239	8.82	0.00%	47.6/min	3.42	4420.0
Add-to-Cart	955	11	2	587	20.34	0.00%	47.6/min	4.14	5348.8
TOTAL	3823	9	1	587	11.76	0.00%	3.2/sec	14.26	4605.0

With agent

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
petstore_item	957	8	2	246	8.77	0.00%	47.1/min	3.39	4420.0
category	957	12	3	60	3.84	0.00%	47.2/min	3.25	4232.0
product	956	9	2	58	3.73	0.00%	47.2/min	3.39	4420.0
Add-to-Cart	956	12	2	202	7.79	0.00%	47.2/min	4.11	5348.8
TOTAL	3826	10	2	246	6.69	0.00%	3.1/sec	14.13	4605.1

Without agent

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
petstore_item	52510	0	0	1883	9.50	0.00%	175.4/sec	757.21	4420.0
category	52510	1	1	137	1.14	0.00%	175.4/sec	725.01	4232.0
product	52510	1	0	67	0.85	0.00%	175.4/sec	757.22	4420.0
Add-to-Cart	52510	1	1	66	1.14	0.00%	175.4/sec	916.34	5348.8
TOTAL	210040	1	0	1883	4.85	0.00%	701.7/sec	3155.70	4605.2

With agent

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
petstore_item	30938	1	1	49	0.97	0.00%	113.5/sec	490.12	4420.0
category	30937	2	2	764	4.45	0.00%	113.5/sec	469.28	4232.0
product	30937	1	1	374	3.15	0.00%	113.5/sec	490.13	4420.0
Add-to-Cart	30937	2	1	65	1.23	0.00%	113.5/sec	593.13	5348.8
TOTAL	123749	2	1	764	2.87	0.00%	454.2/sec	2042.57	4605.2

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Starting SingleAgent on Linux

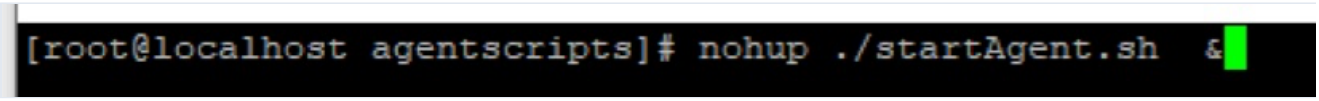
Applicare 9.2 · Updated February 5, 2025 · singleagent, linux · **Pending technical review**

We can start the SingleAgent in two ways

Background mode

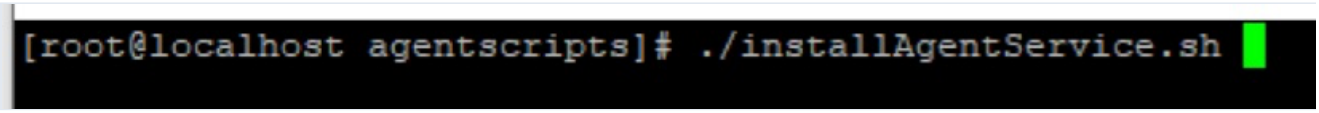
1. To enable **background mode**, Navigate to the directory where **ApplicareSingleAgent/agentscripts** and run following command.

```text
nohup ./startAgent.sh &
```

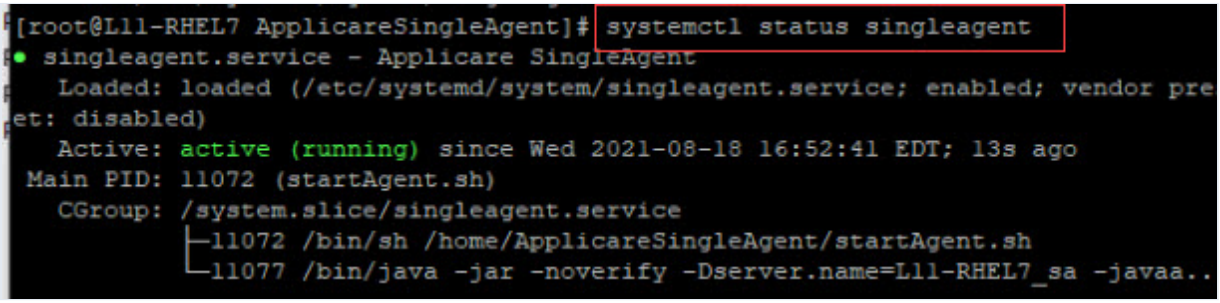


Service Mode

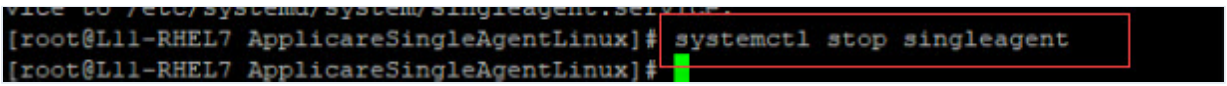
1. Administrative permission required to **install or uninstall singleagent.service**
2. Navigate to the directory where **ApplicareSingleAgent/agentscripts**.
3. Execute **./installAgentService.sh** to install **singleagent.service.** and start **singleagent.service.**



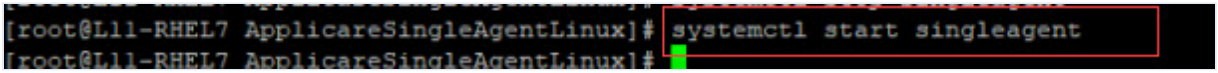
1. To check the status of service, use the command **systemctl status singleagent:**



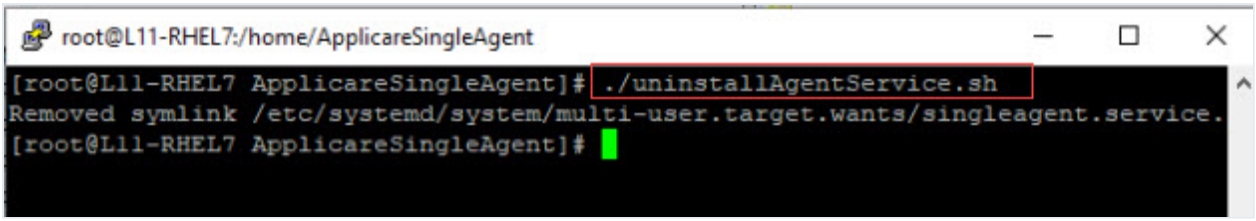
2. To **stop** the service, run the command **systemctl stop singleagent:**



3. To **Start** the service, execute the command **systemctl start singleagent:**



4. To **Uninstall** the service, execute **./uninstallAgentService.sh:**



Monitoring with Applicare

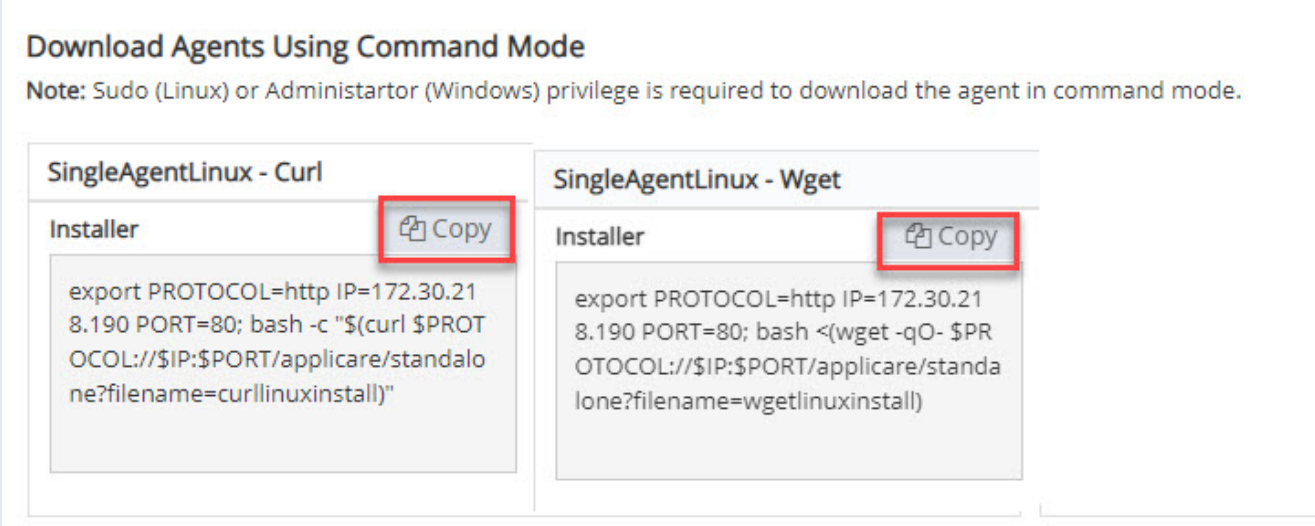
CONFIGURATION GUIDES › INSTALL, START & UPDATE

SingleAgent Installation Using Command Mode for Linux

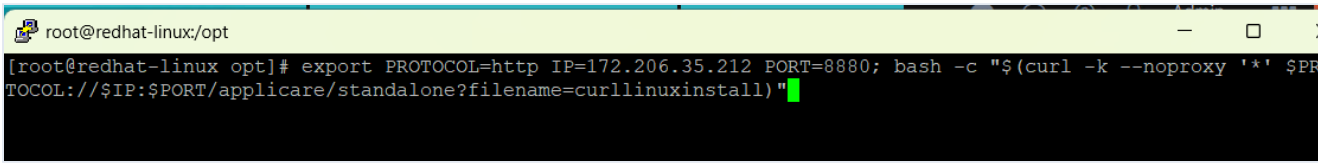
Applicare 9.2 · Updated January 27, 2026 · singleagent, linux · **Pending technical review**

1. Login into Applicare console and navigate to **Download Agents** screen.

2. On the target machine, use either **WGET** or **CURL** installer command, depending on which is available to deploy the agent.

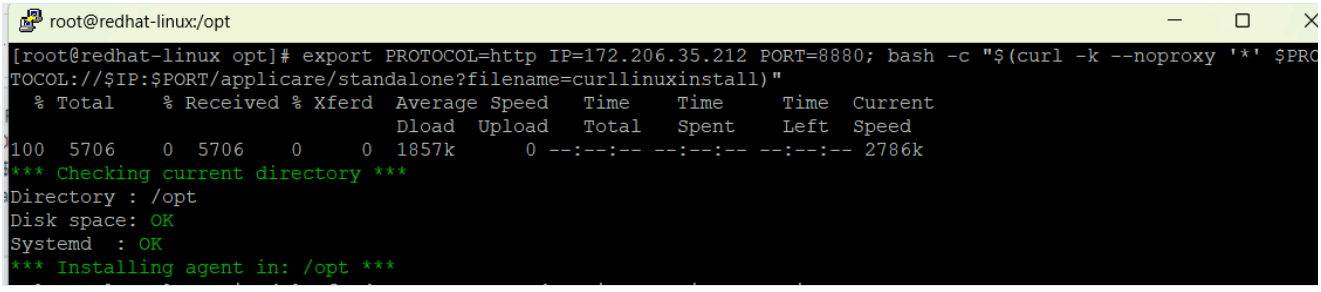


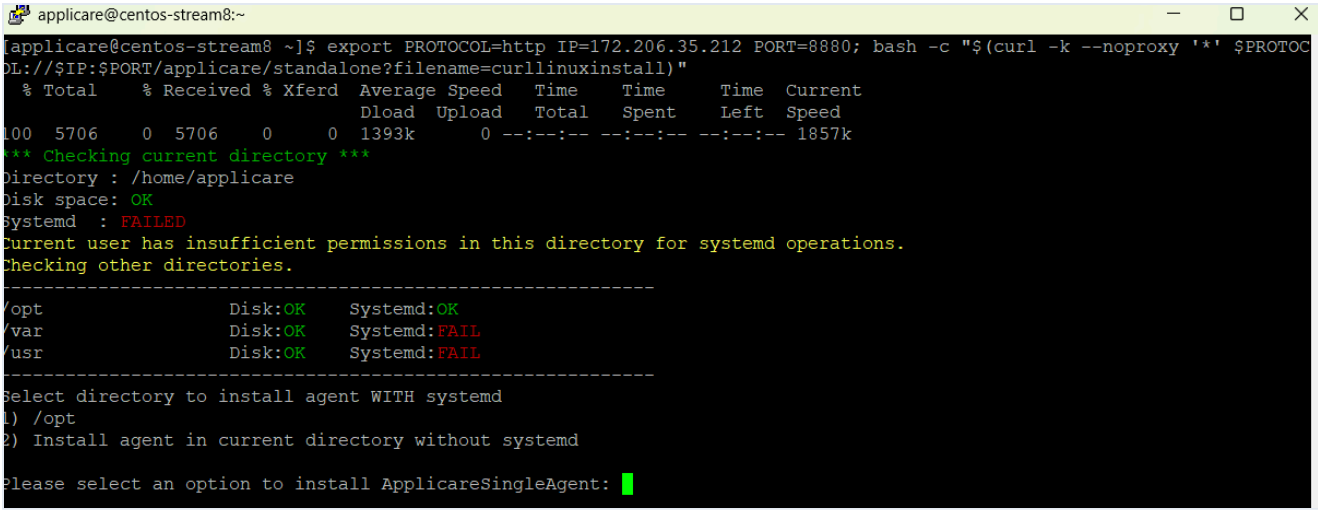
3. Paste the **copied** command into the terminal window and press **Enter** to begin the installation process.



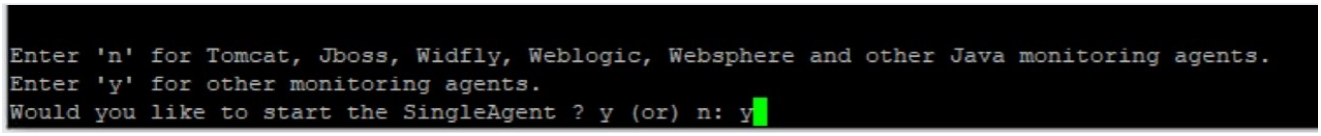
4. The script automatically validates the environment by checking for minimum of **disk space**, verifying directory write permissions, and confirming whether the current user has permission to perform **systemd** operations.

If all requirements are satisfied, the agent is installed in the current directory and configured to run as a **systemd service**. If systemd permissions or disk space are insufficient, the script displays a list of alternative directories (such as **/opt**, **/var**, or **/usr**) that meet the requirements. Select one of these options, or choose to proceed with installation in the current directory without systemd, in **nohup mode**.





5. To complete the setup, enter **“y”** to start the SingleAgent for standard monitoring, or enter **“n”** if you are monitoring Java-based applications such as **Tomcat, JBoss, WildFly, WebLogic, or WebSphere**.



CONFIGURATION GUIDES › INSTALL, START & UPDATE

Starting SingleAgent on Windows

Applicare 9.2 · Updated September 13, 2024 · singleagent, windows · **Pending technical review**

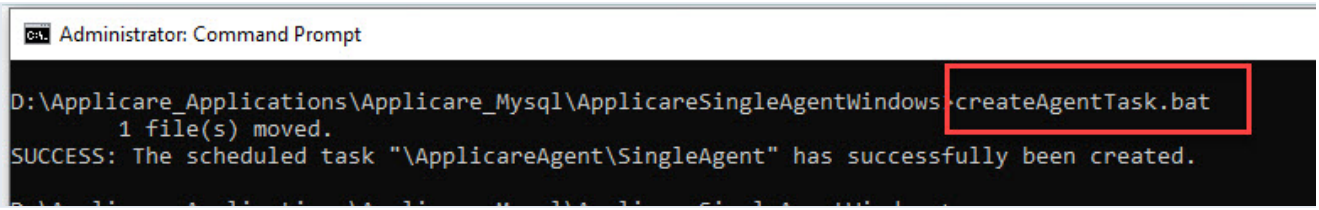
We can start SingleAgent using either of the following methods:

Script Mode

1. Navigate to the **ApplicareSingleAgent** installation directory and execute the **startAgent.bat** file

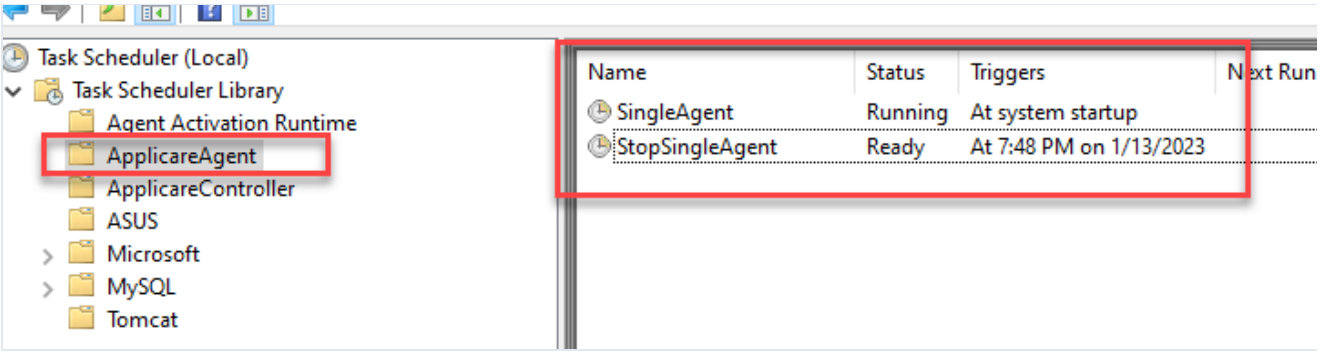
Task Scheduler Mode

1. Open command prompt as an administrator mode, navigate to the **ApplicareSingleAgent** installation directory and run the **createAgentTask.bat** file.



2. SingleAgent will be set up as a scheduled task and will start automatically when the machine boots. To view the task, open **Task Scheduler**, expand the **Task Scheduler Library**, and select the **ApplicareAgent** folder. You will find two tasks listed there.

- **SingleAgent - Use this task to start SingleAgent. By default, it will be running after the agent installation.**
- **StopSingleAgent -** Execute this task to **stop the SingleAgent**.



Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

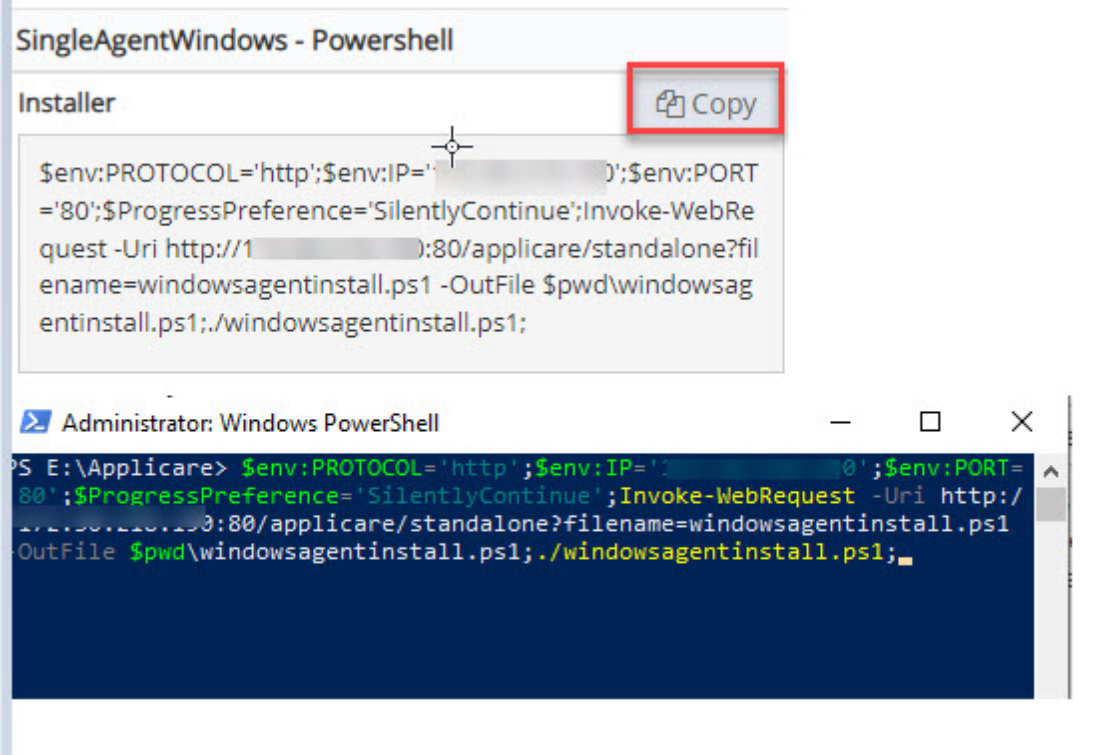
Installing SingleAgent via Command Line on Windows

Applicare 9.2 · Updated February 6, 2025 · singleagent, windows · **Pending technical review**

1. Login in to **Applicare console** and navigate to the **Download Agents** screen.

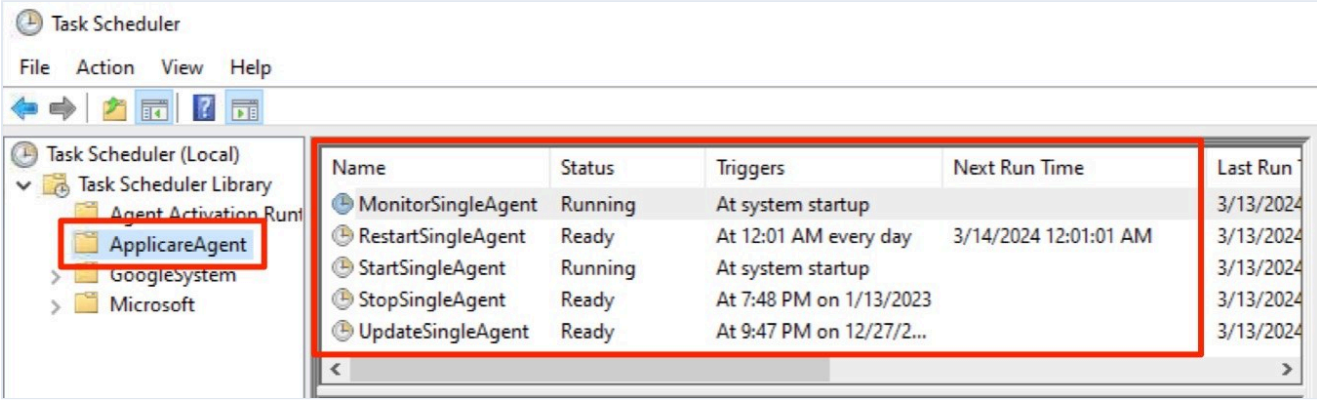
2. Copy the **SingleAgentWindows - Powershell Installer** command.

3. On the target machine where agent will be deployed. Open **PowerShell** in administrator mode, navigate to any one root directory, paste the copied command, press **Enter**.



4. Select option "1".

SingleAgent will be installed as a task and will start automatically. To view the task, open **Task Scheduler** and expand the **Task Scheduler Library,** and select the **ApplicareAgent** folder. You will see five task listed there.



- **SingleAgent -** Run this task to **start the SingleAgent**. After installation, the agent will automatically be in a running state.
- **StopSingleAgent -** Use this task to **stop the SingleAgent**.
- **RestartSingleAgent -** Execute this task to **restart the SingleAgent.**

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Deploying the agent on Windows using an Ansible script

Applicare 9.2 · Updated September 13, 2024 · singleagent, windows · **Pending technical review**

Ensure that Ansible is installed before proceeding with the steps below.

If Ansible is not installed, follow the steps in this link - Installing Ansible on Windows via Ubuntu

Step 1: On the machine where the agent will be deployed, open **PowerShell** in administrator mode and execute the below command one by one.

```text
Enable-PSRemoting -Force
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v
LocalAccountTokenFilterPolicy
```

Open port **5985** on the Machine where the agent will be deployed.

Step 2: On the machine where **Ansible** is running, Create a file named **host.yml** with the content below.

```xml
[win]
<Machine_IP>

[win:vars]
ansible_user=<UserName>
ansible_password=<Password>
ansible_connection=winrm
ansible_winrm_transport=ntlm
ansible_winrm_scheme=http
ansible_winrm_server_cert_validation=ignore
ansible_shell_type=cmd
ansible_port=5985
ansible_winrm_read_timeout_sec=300
ansible_winrm_operation_timeout_sec=150
```

Update the values in the file as specified below.

<Machine_IP> = The machine_IP that wants to deploy the agent

<**UserName>** = Machine Username (Ensure the account has administrative privileges on the Windows machine.)

<Password> = Machine Password

Step 3: On the machine where **Ansible** is running, Create a file named **Agent_deploy.yml** with the following content.

```
POWERSHELL
- name: Create Directory on Windows
  hosts: win
  gather_facts: no
  tasks:
    - name: Change to the specific directory
      win_shell: cd C:\
      args:
        chdir: C:\

    - name: Execute PowerShell command in the specific directory
      win_shell: |
        # Your PowerShell command here

$env:PROTOCOL='CONTROLLER_PROTOCOL';$env:IP='CONTROLLER_IP';$env:PORT='CONTROLLER_PORT';$Prog
ressPreference='SilentlyContinue';Invoke-WebRequest -Uri
CONTROLLER_PROTOCOL://CONTROLLER_IP:CONTROLLER_PORT/applicare/standalone?
filename=windowsagentinstall.ps1 -OutFile
$pwd\windowsagentinstall.ps1;./windowsagentinstall.ps1;
      args:
        chdir: C:\

    - name: Run the BAT file
      win_shell: |
        .\agentscripts\createAgentTask.bat
      args:
        chdir: C:\ApplicareSingleAgentWindows
      become: yes
      become_user: <UserName>
      become_method: runas

    - name: Start the MyScheduledTask task
      community.windows.win_scheduled_task:
        name: StartSingleAgent
        state: present
        folder: "\"ApplicareAgent\""
        actions:
          - path: C:\ApplicareSingleAgentWindows\StartAgent.bat  # Example action
            arguments: test.txt  # Optional arguments for the executable
```

Change the below values in the above file.

CONTROLLER_PROTOCOL -> http (or) https

CONTROLLER_IP -> Applicare Controller IP

CONTROLLER_PORT -> Applicare Controller Port

<**UserName**> = Machine Username (Ensure the account has administrative privileges on the Windows machine.)

Step 3: Navigate to the directory where the file was created and execute the below command.

```
TEXT
 ansible-playbook -i host.yml Agent_deploy.yml
```

Follow these steps only if you are updating the agent.

Step 1: On the machine where **Ansible** is running, Create a file named **update_agent.yml** with the following content.

```
BAT
---

- name: Check Task Status on Task Scheduler
  hosts: win
  gather_facts: false
  tasks:

   - name: Run the BAT file
     win_shell: |
       .\agentupdate.bat
     args:
       chdir: C:\ApplicareSingleAgentWindows\agentscripts
     become: yes
     become_user: <UserName>
     become_method: runas
```

Update the following values in the file.

<**UserName**> = Machine Username (Ensure the account has administrative privileges on the Windows machine.)

Step 2: Run the following command from the same directory where host.yml is located.

```
TEXT
ansible-playbook -i host.yml update_agent.yml
```

Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

SingleAgent Remote deployment in Windows using Powershell

Applicare 9.2 · Updated December 31, 2024 · singleagent, windows · **Pending technical review**

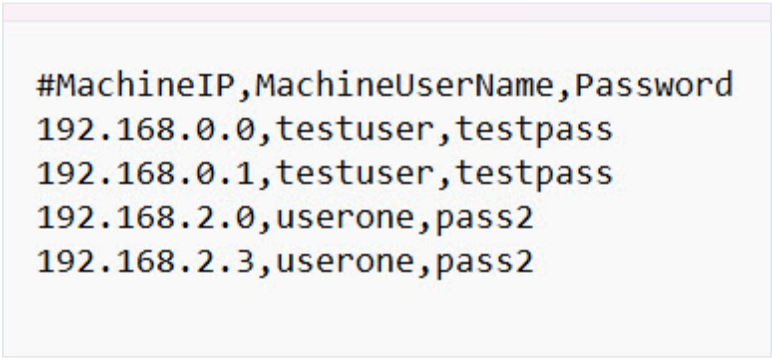
Applicare provides the capability to deploy agents remotely, ensuring a smooth and efficient setup process.

Step 1: Download the machine_details.txt file in the Host machine and **add the Machine IPs, Login Usernames and passwords.**

- Download the **machine_details.txt** from **here** and add the list of target machines with their IP addresses, login usernames and passwords.

Example:

```
TEXT
#MachineIP,MachineUserName,Password
192.168.0.0,testuser,testpass
192.168.0.1,testuser,testpass
192.168.2.0,userone,pass2
192.168.2.3,userone,pass2
```



Step 2: Download the windows_agent_instalation_script.ps1 file in the Host machine and **add Controller Protocol, IP, and Port.**

- Downlaod the **windows_agent_instalation_script.ps1** from **here** and configure the controller protocol, ip and port details in the script using the following variables.

Template:

```powershell
$env:PROTOCOL = 'Controller-protocol'
$env:IP = 'Controller-ip'
$env:PORT = 'Controller-port'
$ProgressPreference = 'SilentlyContinue'

# Download and execute the script
Invoke-WebRequest -Uri Controller-protocol://Controller-ip:Controller-
port/applicare/standalone?filename=windowsagentinstall.ps1 -OutFile
$pwd\windowsagentinstall.ps1
.\windowsagentinstall.ps1
```

Example:

```powershell
$env:PROTOCOL = 'http'
$env:IP = '192.168.0.154'
$env:PORT = '8880'
$ProgressPreference = 'SilentlyContinue'

# Download and execute the script
Invoke-WebRequest -Uri http://192.168.0.154:8880/applicare/standalone?
filename=windowsagentinstall.ps1 -OutFile $pwd\windowsagentinstall.ps1
.\windowsagentinstall.ps1
```



Step 3: Enable the following PowerShell settings on both the host and remote machines to ensure proper communication and functionality

On the Host Machine:

1. **Set Network Profile to Private**

   Powershell

   ```powershell
   Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Private
   ```

Explanation: Setting the network category to private allows trusted communication with other devices on the network and enables protocols like WinRM for remote management.

2. **Enable WinRM**

   Powershell

   ```
   TEXT
   winrm quickconfig -Force
   ```

   Explanation: WinRM (Windows Remote Management) is required for PowerShell remoting. This command ensures WinRM is enabled and ready to communicate with remote machines.

3. **Enable the ExecutionPolicy**

   Powershell

   ```
   POWERSHELL
   Set-ExecutionPolicy Unrestricted -Scope CurrentUser
   ```

   Explanation: PowerShell to execute scripts without any restrictions, but only for the current user.

On the Remote Machine:

1. **Set Network Profile to Private**

   Powershell

   ```
   POWERSHELL
   Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Private
   ```

   Explanation: A private network category allows trusted communication with other devices in the network and enables protocols like WinRM for remote management.

2. **Enable PowerShell Remoting**

   Powershell

   ```
   TEXT
   Enable-PSRemoting -Force
   ```

   Explanation: This command configures the remote machine to accept PowerShell commands from the host.

3. **Allow Traffic on Port 5985** Ensure the firewall allows traffic on the **WinRM default port (5985) TCP.**

   Explanation: This opens the required port to enable communication via WinRM over HTTP.

4. **Machine IP, Login Username and Password** Ensure that the login user has **Administrator** privileges on the machine.

   Explanation: Administrator access is required to enable remote communication protocols, and execute scripts with elevated permissions.

Step 4: Open **PowerShell** in **Administrator mode** in the **Host machine** and navigate to the directory where the script is located and execute the following command

```powershell
.\windows_agent_instalation_script.ps1
```

Note: Ensure that both the **machine_details.txt** and **windows_agent_instalation_script.ps1** are located in the same directory before running the script.

CONFIGURATION GUIDES › INSTALL, START & UPDATE

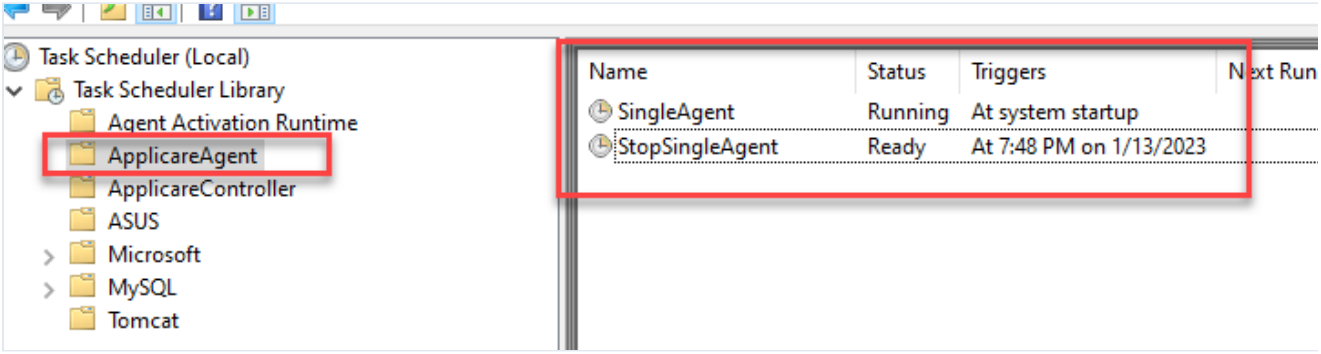
Setting Up Automatic Start of Applicare Agent with Windows Task Scheduler

Applicare 9.2 · Updated September 13, 2024 · singleagent, windows, tasks · **Pending technical review**

To Applicare SingleAgent start task is automatically added to Windows Task Scheduler when downloaded using the **PowerShell command** from the **Agent Download** page. Alternatively, it can be manually added by running **createAgentTask.bat** in Applicare SingleAgent home directory.

1. Open **Task Scheduler,** expand the **Task Scheduler Library** and select the **ApplicareAgent** folder. You will find two task.

- **SingleAgent -** To start the SingleAgent
- **StopSingleAgent -** To stop the SingleAgent



Monitoring with Applicare

CONFIGURATION GUIDES › INSTALL, START & UPDATE

Task creation to stop the SingleAgent

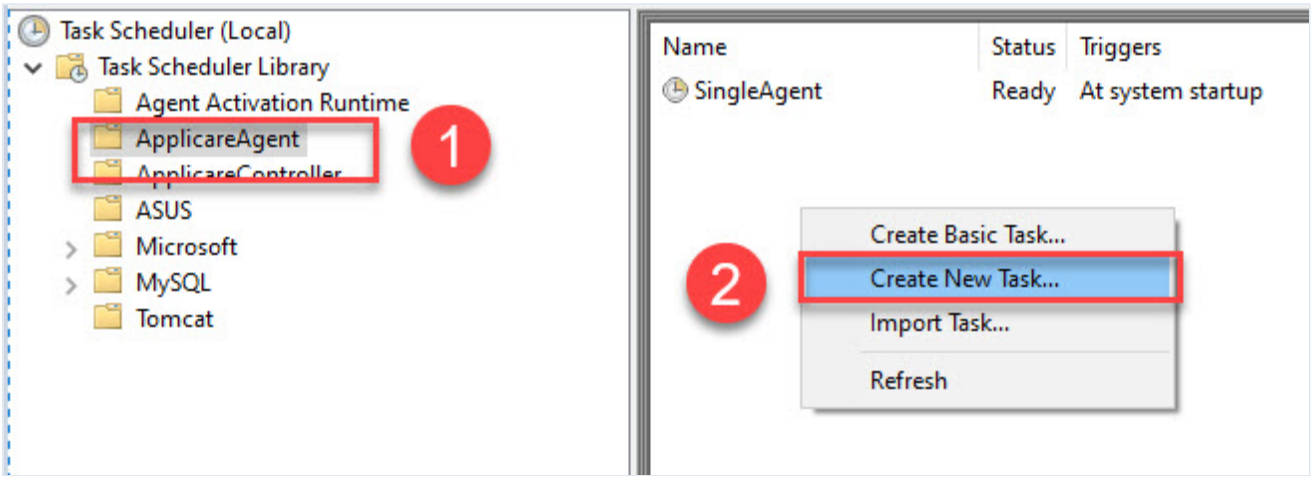
Applicare 9.2 · Updated September 13, 2024 · singleagent, windows, tasks · **Pending technical review**

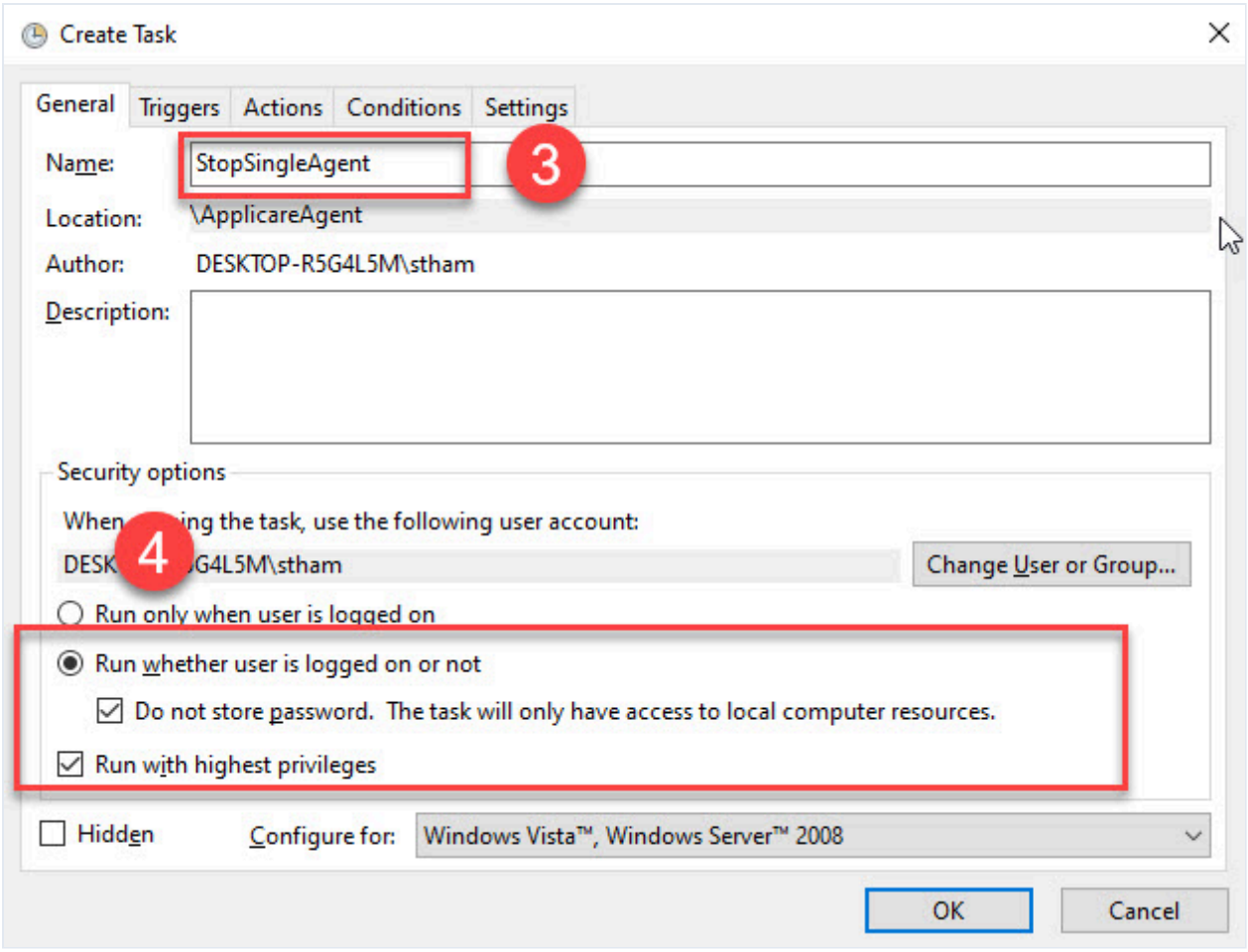
1. Navigate to **ApplicareSingleAgent** home directory and create a **stopAgent.bat** file with the following content. Save the file once created.

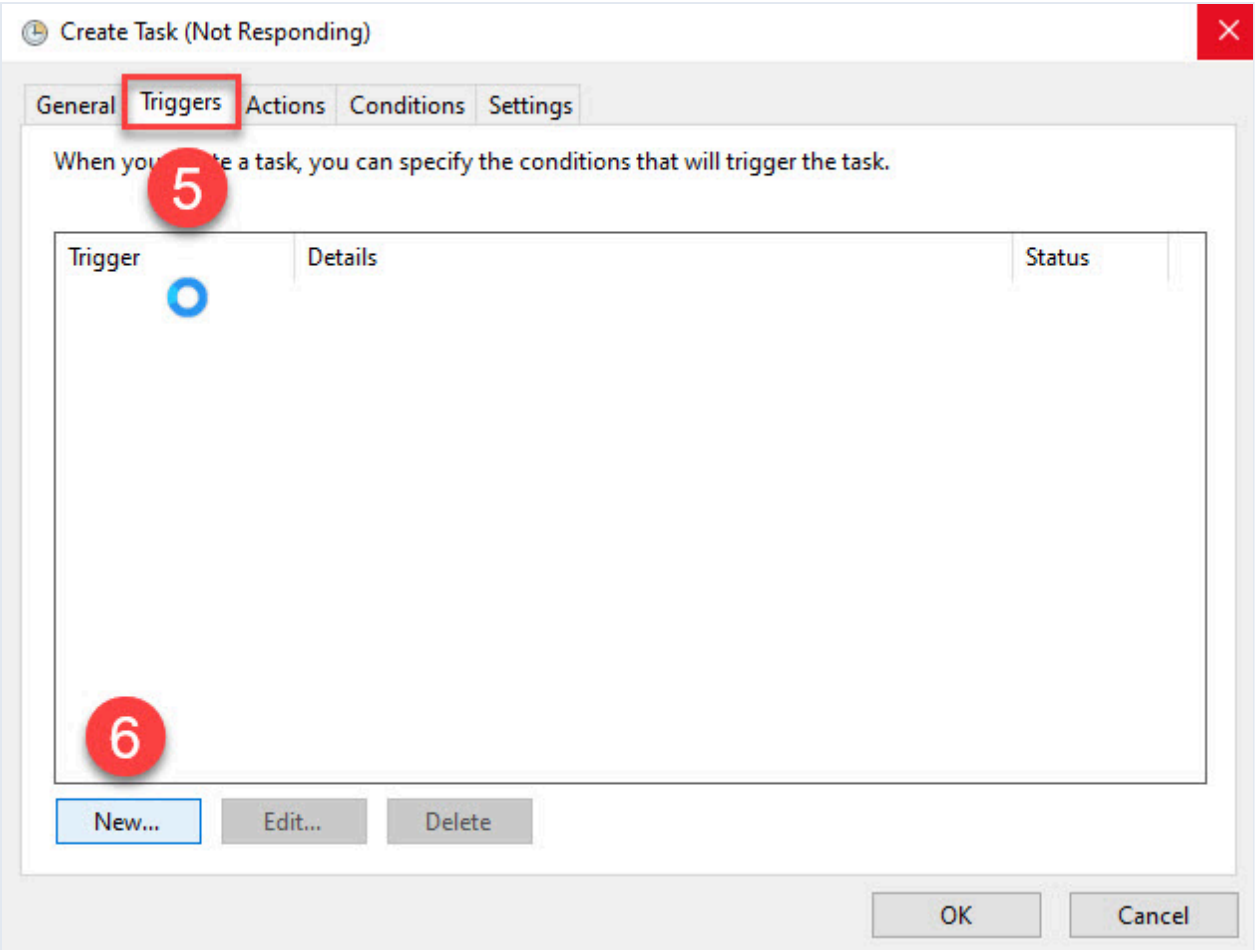
@echo off wmic process where "commandline like '%%applicare.standalone=true%%' and name='java.exe'" delete

```
@echo off
wmic process where "commandline like '%%applicare.standalone=true%%' and name='java.exe'" delete
```

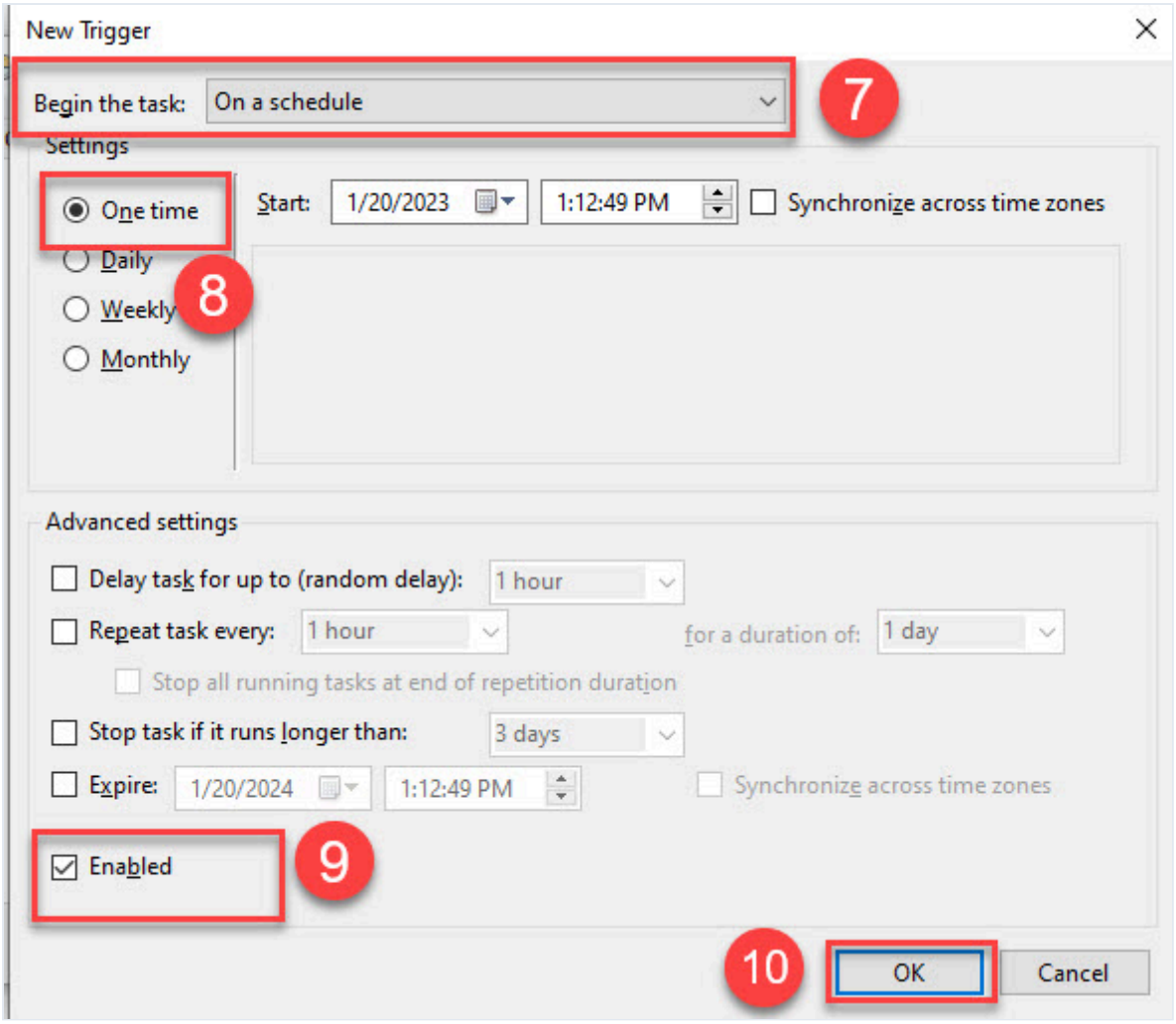
2. Open **Task Scheduler** and create the task as specified below.




Create Task
General
Triggers
Actions
Conditions
Settings
Name:
StopSingleAgent
3
Location:
\ApplicareAgent
Author:
DESKTOP-R5G4L5M\stham
Description:
Security options
When ing the task, use the following user account:
DESK G4L5M\stham
4
Change User or Group...
Run only when user is logged on
Run whether user is logged on or not
Do not store password. The task will only have access to local computer resources.
Run with highest privileges
Hidden
Configure for:
Windows Vista™, Windows Server™ 2008
OK
Cancel



Do not alter the start time.

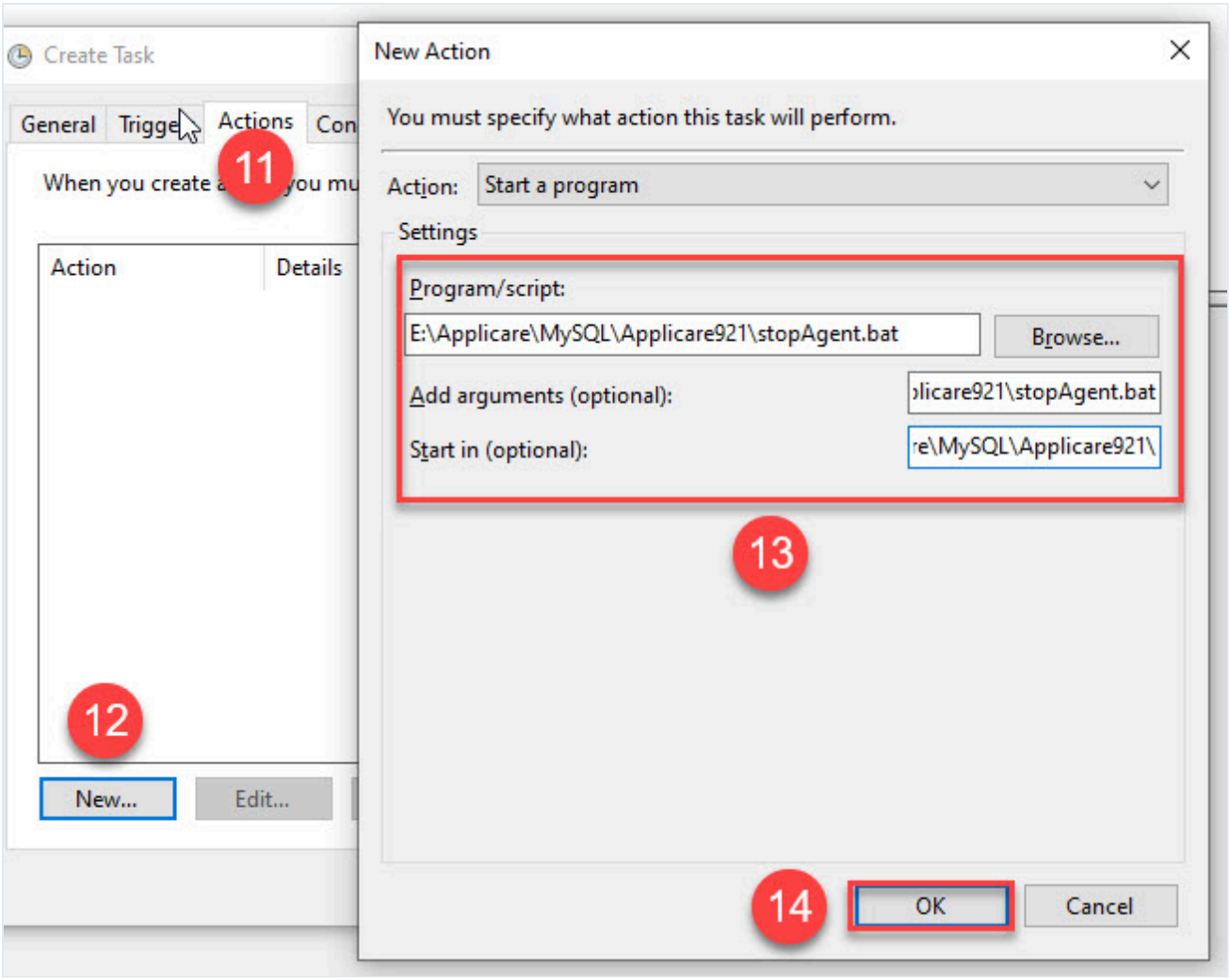


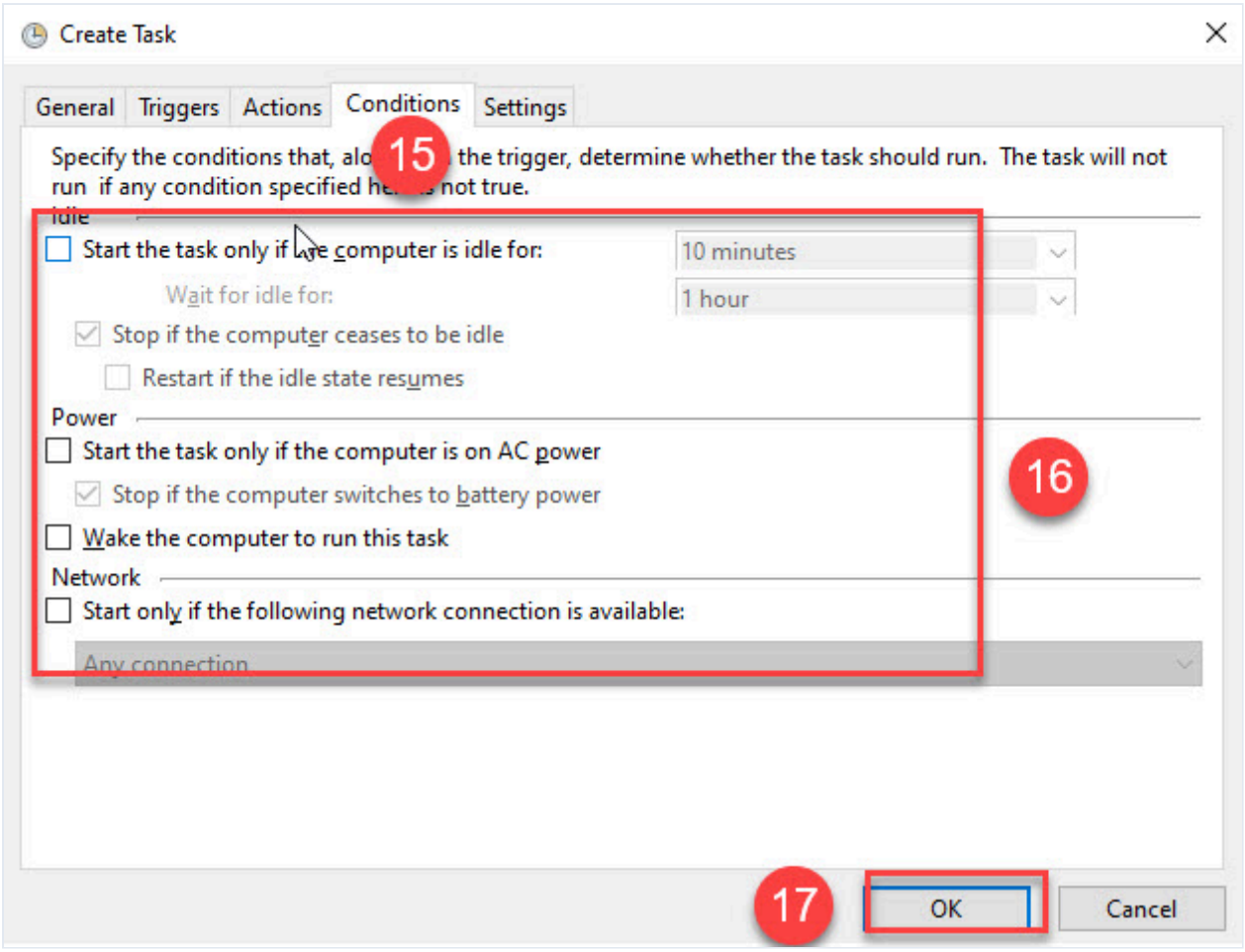
Program/script: AgentHomeDir\stopAgent.bat

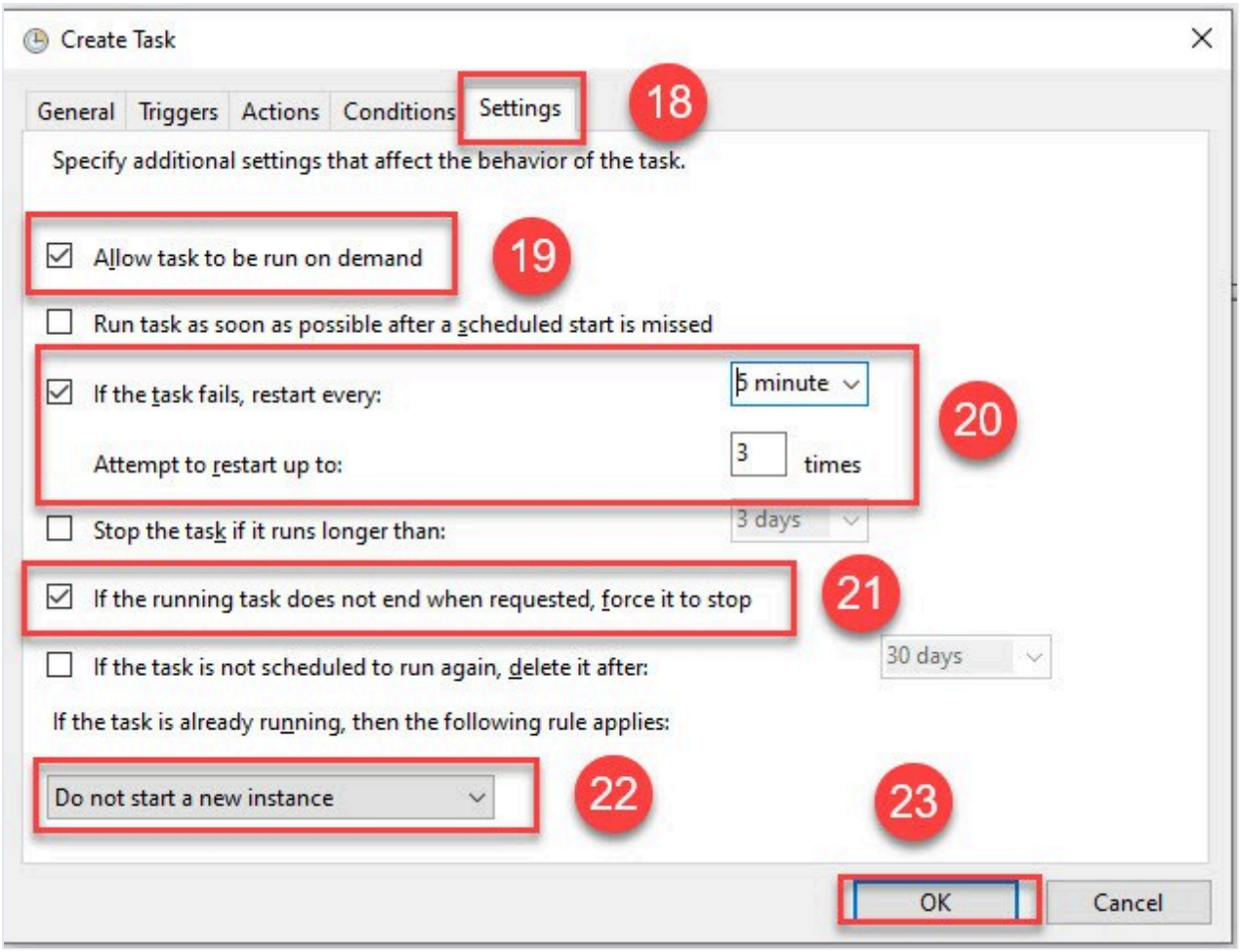
Add arguments(optional): AgentHomeDir\stopAgent.bat

Start in (optional): AgentHomeDir\

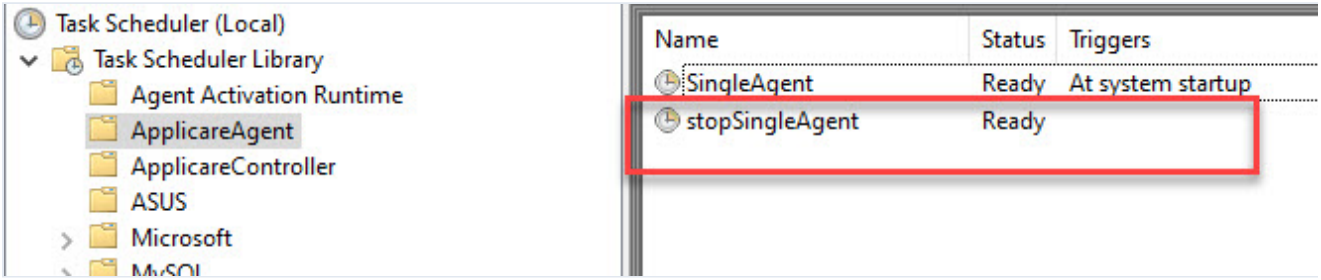
Replace **AgentHomeDir** with the correct path.


Create Task
General
Trigge
Actions
Con
11
When you create a
you mu
Action
Details
12
New...
Edit...
New Action
You must specify what action this task will perform.
Action:
Start a program
Settings
Program/script:
E:\Applicare\MySQL\Applicare921\stopAgent.bat
Browse...
Add arguments (optional):
licare921\stopAgent.bat
Start in (optional):
e\MySQL\Applicare921\
13
14
OK
Cancel


Create Task
General
Triggers
Actions
Conditions
Settings
15
Specify the conditions that, alo the trigger, determine whether the task should run. The task will not run if any condition specified he is not true.
Idle
Start the task only if the computer is idle for:
10 minutes
Wait for idle for:
1 hour
Stop if the computer ceases to be idle
Restart if the idle state resumes
Power
Start the task only if the computer is on AC power
Stop if the computer switches to battery power
Wake the computer to run this task
16
Network
Start only if the following network connection is available:
Any connection
17
OK
Cancel



The task will be generated.



Running this task it will terminate all the Java processes, and the **SingleAgent** task Status will be changed to Ready. Next, right-click the **SingleAgent** task and select **Run** to start the agent.

Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Installing the Agent for Tomcat, JBoss, WebLogic, WebSphere, and Other Java-Based Monitoring

Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

Windows

1. Log in to **Applicare console** and navigate to the **Download Agents** screen.

2. Click the **SingleAgentWindows** button to download the agent. This will download the **ApplicareSingleAgentWindows.zip** file. Next, transfer the **ApplicareSingleAgentWindows.zip** file to the machine where the agent will be deployed.

3. Extract the **ApplicareSingleAgentWindows.zip** file into a directory named **ApplicareSingleAgentWindows**.

Linux

1. Log in to the **Applicare console,** navigate to the **Download Agents** screen and download **SingleAgentLinux.** Then, transfer the **ApplicareSingleAgentLinux.zip** file to the machine where agent will be deployed.

2. Create a directory named "**ApplicareSingleAgent**" on the agent machine, then extract the **ApplicareSingleAgentLinux.zip** file into that directory.

3. Set **chmod -R 777** permission for the **ApplicareSingleAgent** directory.

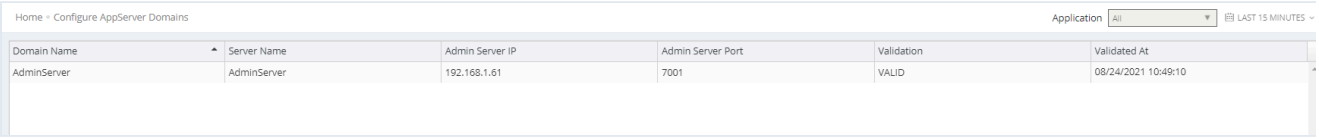
Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Domain Configuration

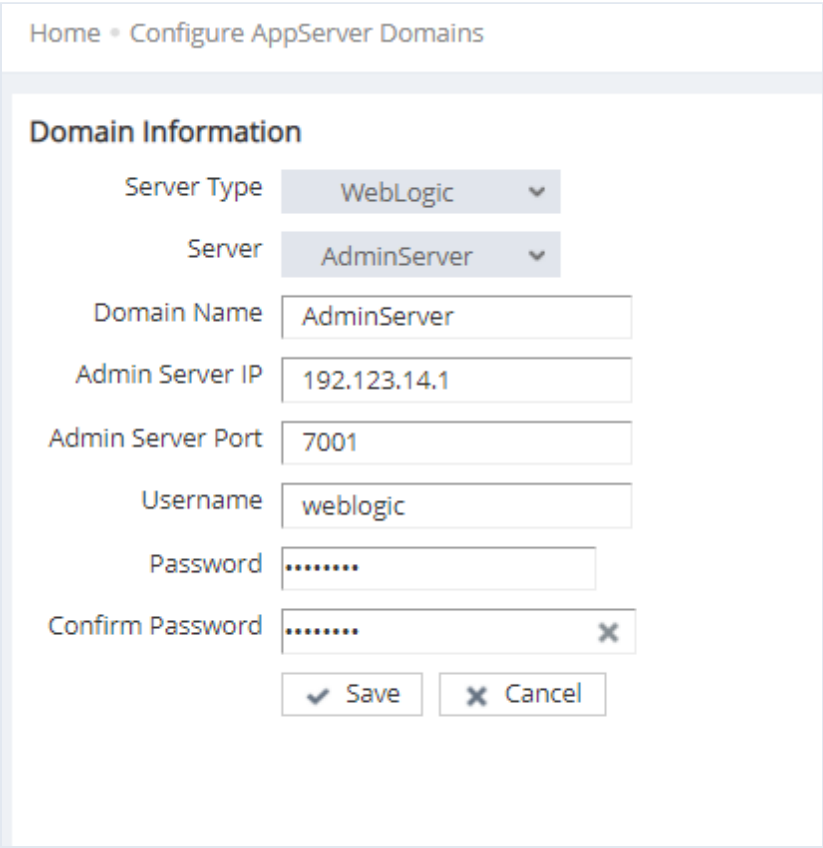
Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

You need to save WebLogic domain information in Applicare before deploying the Applicare agent to WebLogic servers.



Configuration

Click the "New" button to add a new domain. Enter the domain name and the Admin Server connection details for the domain.



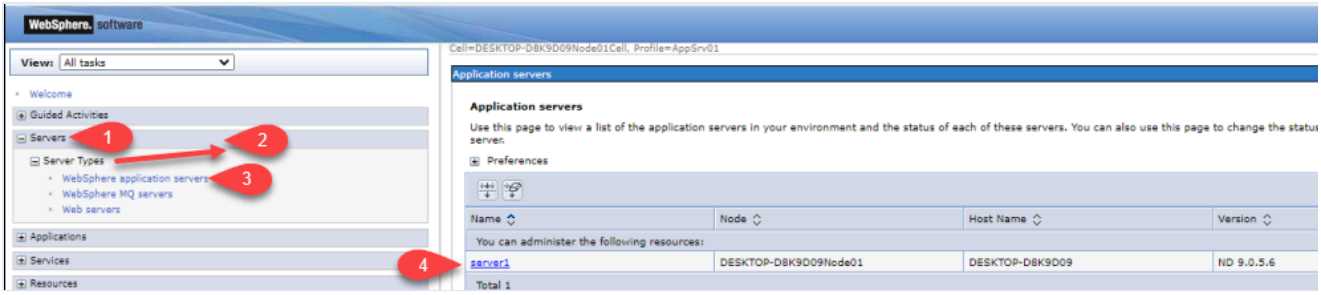
Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

WebSphere Monitoring

Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

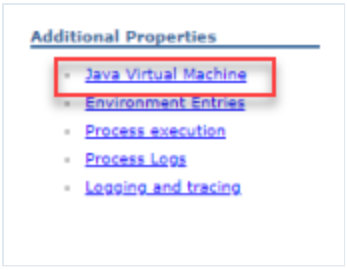
1. Install Applicare SingleAgent in the machine that **WebSphere** is running in.
2. Log in to the Websphere console and navigate **servers**> **server Types** > **Websphere application servers**, and then select the server.



3. Click on **Java and Process Management**> **Process definition** under the Server Infrastructure section.



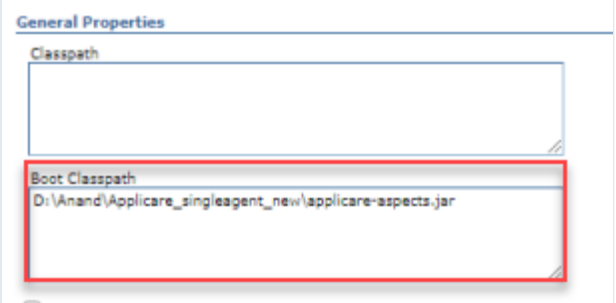
4. In the additional properties section, click on **Java Virtual Machine**.



5. Save the value in the **boot classpath** as shown below

```xml
<ApplicareSingleagent_homepath>\applicare-aspects.jar
```

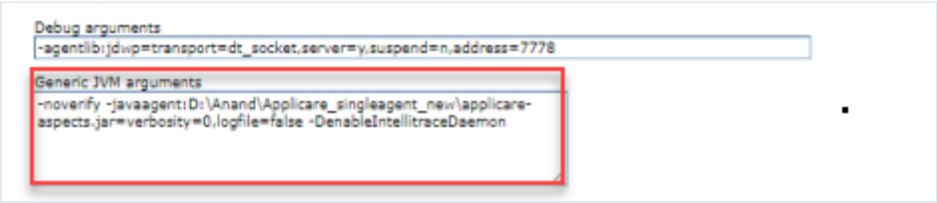
- **<ApplicareSingleagent_homepath>** - refers to the directory where the Applicare SingleAgent zip file extracted.



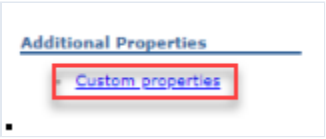
6. Save the value as shown below the **Generic JVM arguments.**

```
TEXT
-noverify -javaagent:<ApplicareSingleagent_homepath>\applicare-
aspects.jar=verbosity=0,logfile=false
```

- **Note: <ApplicareSingleagent_homepath>** - Applicare SingleAgent zip file extracted directory.



7. Next, add the following **custom properties** by clicking the Custom Properties link under Additional Properties heading on the same page.



Name	Value	Description
Arcturus_home	<**ApplicareSingleagent_homepath**>	Absolute path to your **ApplicareSingleagent_homepath** directory
java.awt.headless	True	
java.library.path	<**ApplicareSingleagent_homepath**>**/bin**	The full path to your **ApplicareSingleagent_homepath** directory
profile.name	<**Profile Name**>	The profile name you used when configuring the profile in Applicare console
server.name	<**Server Name**>	Name of your WebSphere server

8. Restart the **WebSphere server** to apply the changes.

For Other Docs - Click here

Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Wildfly Monitoring

Applicare 9.2 · Updated August 22, 2024 · java, singleagent · **Pending technical review**

Step 1: Install Applicare SingleAgent in the machine where **Wildfly** is running. Follow the instructions for **installing Applicare SingleAgent**

If Wildfly is started using the standalone.conf.bat file proceed to Step 2.

Step 2: Navigate to <**Wildfly_home_path**>\bin Add the following line at the beginning of the **standalone.conf.bat** file.

```bat
set SERVER_NAME=ServerName
call <ApplicareSingleAgentHomePath>\scripts\setJBossAgentEnv.bat
```

```
15  rem Uncomment the following line to disable manipulation of JAVA_OPTS (JVM parameters)
16  rem set PRESERVE_JAVA_OPTS=true
17
18  call C:\Anand\ApplicareSingleagent_L8\scripts\setJBossAgentEnv.bat
19
```

Replace ServerName with the desired name for the agent. The name will be used when adding the agent to the controller.

Replace **ApplicareSingleAgentHomePath** with the actual path to your **ApplicareSingleAgent** home directory.

Example:

```
rem set PRESERVE_JAVA_OPTS=true

set SERVER_NAME=Petstore
call I:\Anand\ApplicareSingleAgentWindows\scripts\setJBossAgentEnv.bat

if not "x%JAVA_OPTS%" == "x" (
  echo "JAVA_OPTS already set in environment; overriding default settings with values: %JAVA_OPTS%"
  goto JAVA_OPTS_SET
)
```

Additional steps for install Applicare SingleAgent with Wrapper.

For Other Docs - Click here

CONFIGURATION GUIDES › APPLICATION SERVERS

Monitoring JBoss on Linux

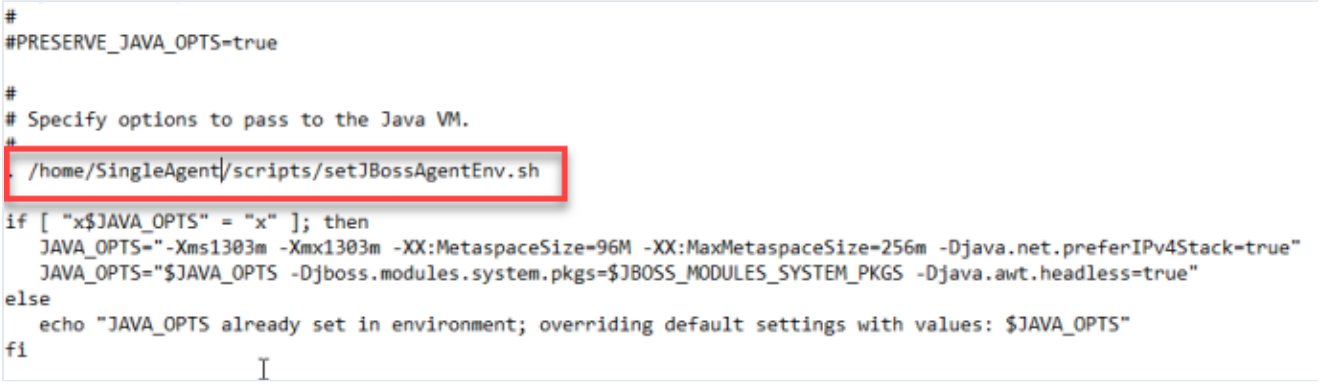
Applicare 9.2 · Updated September 13, 2024 · java, singleagent · **Pending technical review**

Step 1: Install the Applicare SingleAgent on the machine running **JBoss**. Follow these steps: **installing Applicare SingleAgent**

If Jboss is started using the standalone.conf.sh file proceed with Step 4.

Step 2: Navigate to <**JBoss_home_path**>\bin and Add the following line at the beginning of the **standalone.conf.sh** file.

```bash
export SERVER_NAME=ServerName
. /<Applicare_agent_home_directory>/scripts/setJBossAgentEnv.sh
```



```
#
#PRESERVE_JAVA_OPTS=true

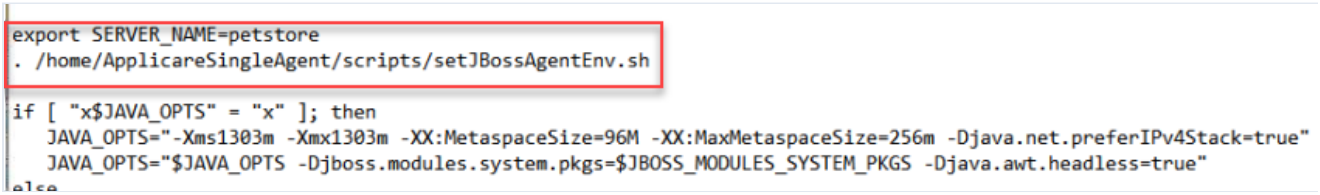
#
# Specify options to pass to the Java VM.
#
. /home/SingleAgent/scripts/setJBossAgentEnv.sh

if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
else
   echo "JAVA_OPTS already set in environment; overriding default settings with values: $JAVA_OPTS"
fi
```

Replace ServerName with the desired name for the agent. The agent will be registered with the controller under this name.

Replace **Applicare_agent_home_directory** with the actual directory path of the **ApplicareSingleAgent**.

e.g.



```
export SERVER_NAME=petstore
. /home/ApplicareSingleAgent/scripts/setJBossAgentEnv.sh

if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
else
```

Additional steps to install Applicare SingleAgent with Wrapper.

Docs Home

Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Installing SingleAgent for AppServer on Linux

Applicare 9.2 · Updated August 12, 2024 · java, singleagent, linux · **Pending technical review**

1. Log in to **Applicare console,** and navigate to the **Download Agents** section.

2. Click the **SingleAgentLinux** button to download the agent. This will download the **ApplicareSingleAgentLinux.zip** file. Copy this file to the machine where the agent will be deployed.

3. Extract the contents of the **ApplicareSingleAgentLinux.zip** file into a directory named **ApplicareSingleAgentLinux**.

4. Grant **chmod -R 777** permission to the **ApplicareSingleAgentLinux** directory.

For Other Docs - Click here

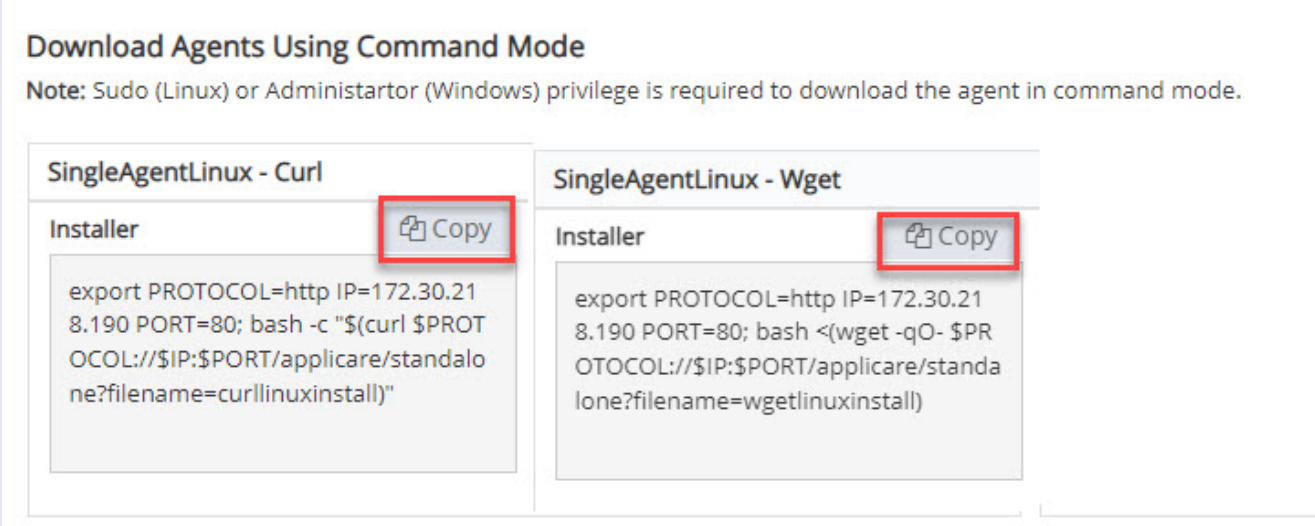
CONFIGURATION GUIDES › APPLICATION SERVERS

SingleAgent Installation for Appserver using command Mode for Linux

Applicare 9.2 · Updated September 13, 2024 · java, singleagent, linux · **Pending technical review**

1. Log in to **Applicare console,** Navigate to **Download Agents** screen.

2. On the machine where the agent will be deployed. If **Wget** is installed, execute the **Wget Installer** command. If **Curl** is installed, Use the **Curl Installer** command instead.



3. Enter **n** to prevent the SingleAgent from starting Automatically.

Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Installing Applicare SingleAgent on WebLogic running on Linux

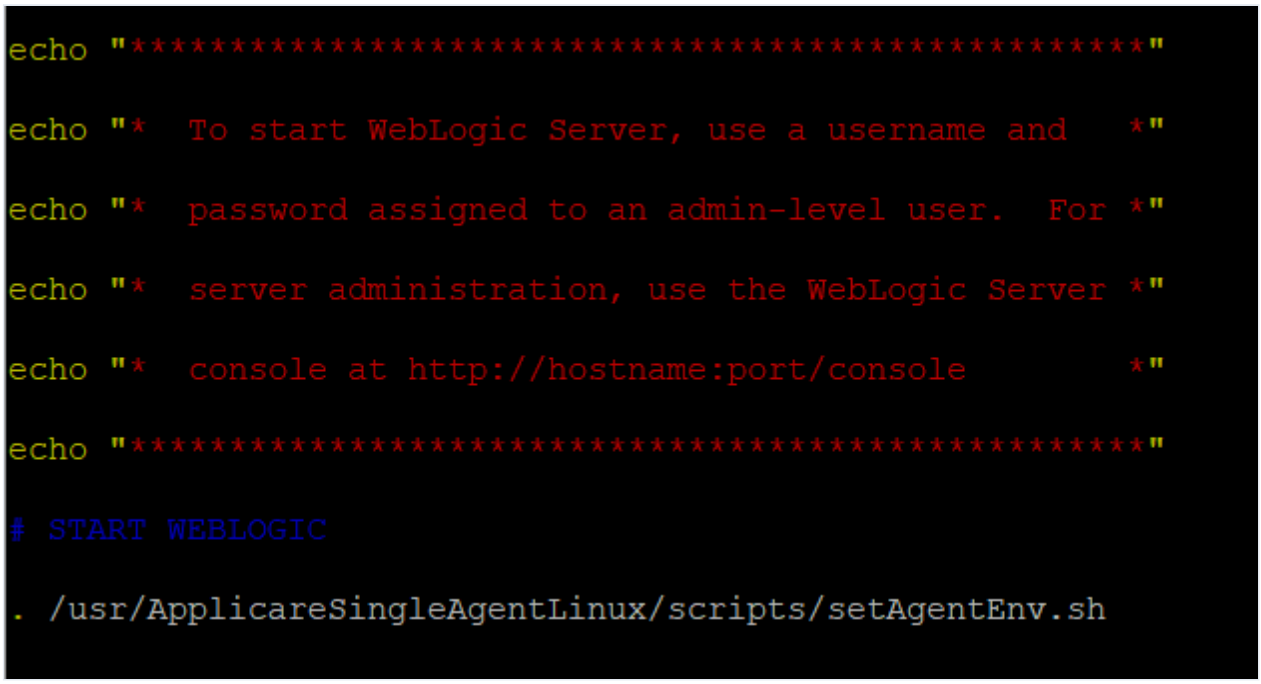
Applicare 9.2 · Updated March 23, 2026 · java, singleagent, linux · **Pending technical review**

1. Install Applicare SingleAgent in the machine where **Weblogic** is running.
2. Navigate to the <**weblogic_domain_path**>\bin directory and add the following line to the **startWebLogic.bat** file:

```text
. /<ApplicareSingleAgent_homepath>/scripts/setAgentEnv.sh
```

- **<ApplicareSingleagent_homepath>**

  - The directory where the Applicare SingleAgent zip file has been extracted.



3. Restart Weblogic

For Other Docs - Click here

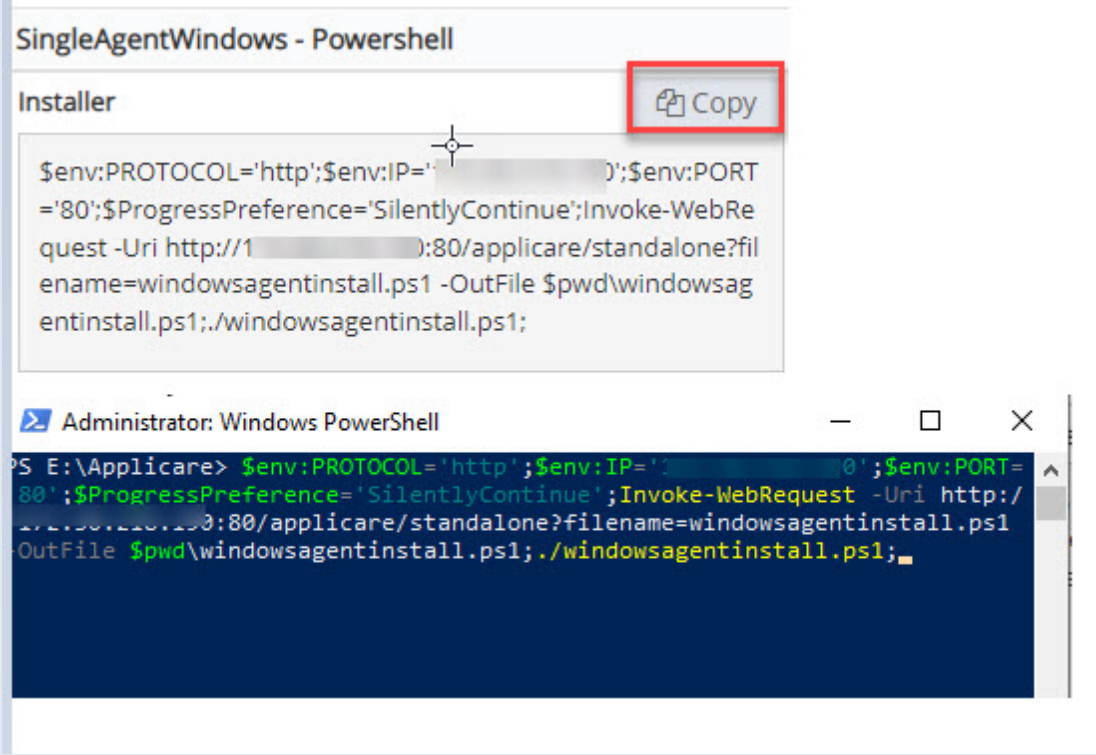
CONFIGURATION GUIDES › APPLICATION SERVERS

Installing SingleAgent for Appserver on Windows using Command Line

Applicare 9.2 · Updated September 13, 2024 · java, singleagent, windows · **Pending technical review**

1. Log in to the **Applicare console** and navigate to the **Download Agents** screen.

2. On to the machine where the agent is to be deployed. Copy the **SingleAgentWindows - PowerShell Installer** command. Open **PowerShell** in administrator mode, navigate to any drive, and paste the copied command and press **Enter** button.



3. Select option **n** if you do not wish the start SingleAgent.

Monitoring with Applicare

CONFIGURATION GUIDES › APPLICATION SERVERS

Deploying SingleAgent for WebLogic on Windows

Applicare 9.2 · Updated March 23, 2026 · java, singleagent, windows · **Pending technical review**

1. **Install Applicare SingleAgent** on the machine where **Weblogic** is running.
   - **If Weblogic is started using startWebLogic.bat file follow these step.**
2. Navigate to the <**weblogic_domain_path**>\bin directory **startWebLogic.bat** file.
   - Go to <**weblogic_domain_path**>\bin directory and add the following line to the **startWebLogic.bat** file.

```bat
call <ApplicareSingleAgent_homepath>\scripts\setAgentEnv.bat
```

- **<ApplicareSingleagent_homepath>** - The directory where the Applicare SingleAgent zip file has been extracted.

```
187 echo ************************************************
188
189 echo *  To start WebLogic Server, use a username and   *
190
191 echo *  password assigned to an admin-level user.  For *
192
193 echo *  server administration, use the WebLogic Server *
194
195 echo *  console at http:\\hostname:port\console        *
196
197 echo ************************************************
198
199 @REM START WEBLOGIC
200
201 call D:\ApplicareSingleAgentWindows\scripts\setAgentEnv.bat
202
203
```

3. Restart Weblogic

Docs Home - Click here

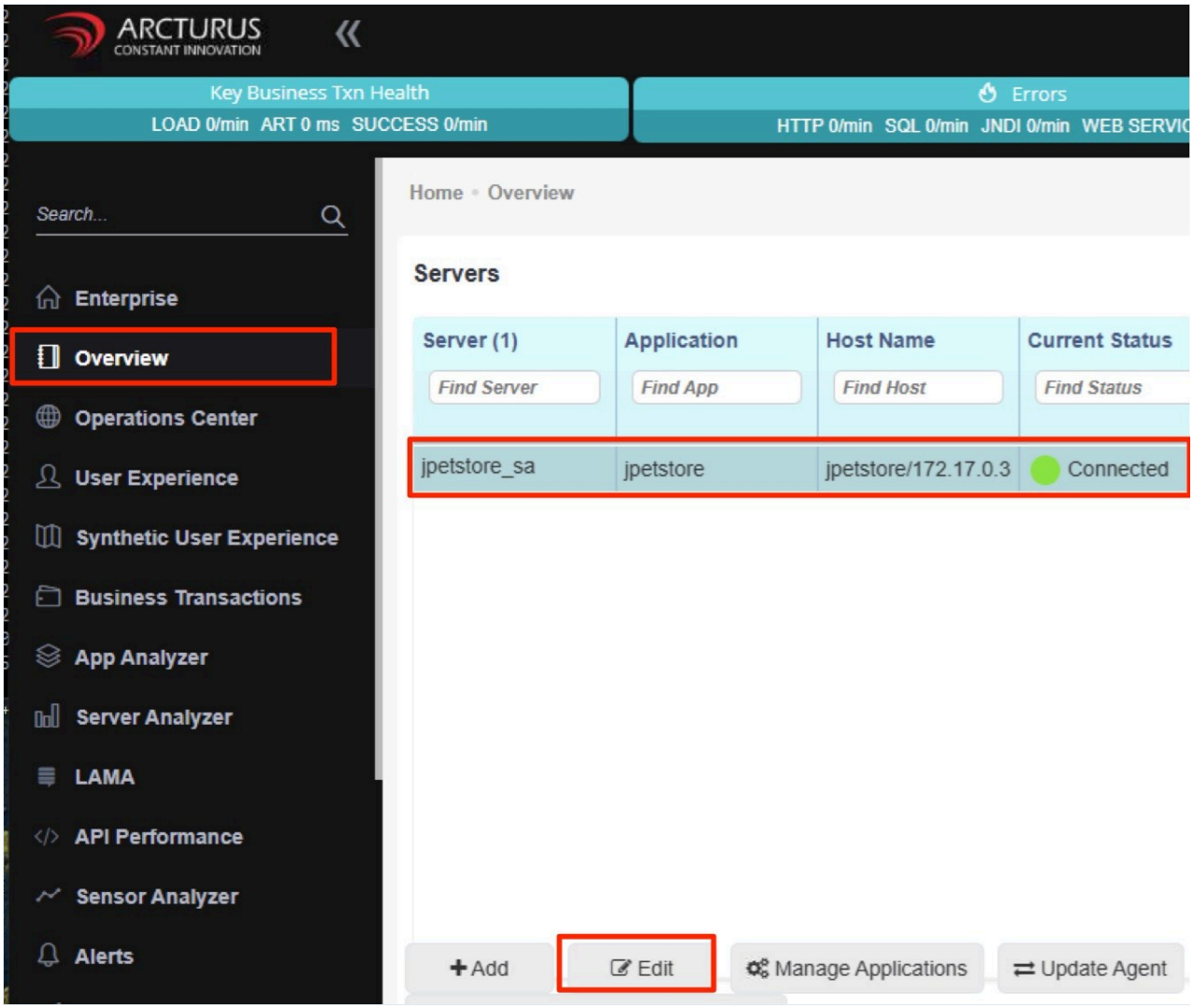
Monitoring with Applicare

CONFIGURATION GUIDES › WEB SERVERS

Apache Monitoring on Linux

Applicare 9.2 · Updated February 28, 2025 · apache · **Pending technical review**

1. Follow the steps for **Installing Applicare SingleAgent.**
2. After starting the agent, it will automatically be registered with the Applicare controller. Log in to the **Applicare console** and navigate to the **Overview** menu.
3. After a few minutes, choose the newly added agent and click the **Edit** button.



4. Verify if the **Apache configuration** appears in the popup screen under the **Add Apache configuration.**

If the Apache configuration is not visible in this popup screen, follow the steps provided in this link

CONFIGURATION GUIDES › WEB SERVERS

Apache server 2.4 Monitoring for Linux

Applicare 9.2 · Updated September 13, 2024 · apache · **Pending technical review**

To installing Applicare SingleAgent, Follow these steps:

Before proceeding with the configuration verify the following:

1. Identify the location of the default log configuration settings for the Apache server.

2. Confirm that all Apache server requests are being logged in the currently configured log file.

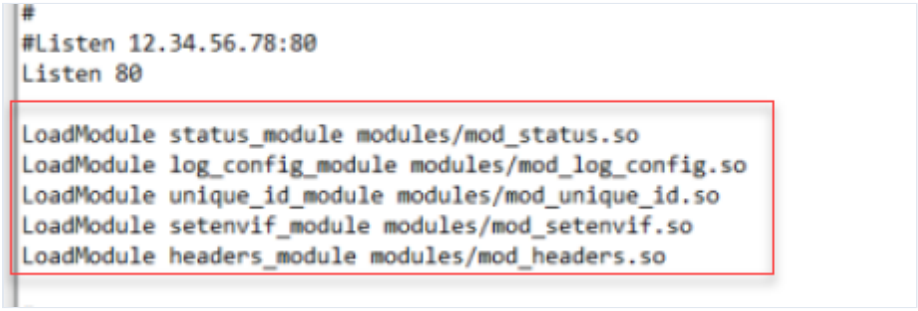
Apache server Monitoring with Applicare

Step 1: Enable the server-status and custom log modules in Apache. Navigate to **Apache installation directory** and edit the **conf file**. (Default location: **/etc/httpd/conf** (or) **/etc/apache/conf**).

Step 2: Ensure the following modules are enabled in the **conf file**. If it is not enable it by removing '#' at the start of the line.

If any of these modules are not present in the Apache conf file by default. add them below manually below the 'Listen' line in the directive conf file.

```text
LoadModule status_module modules/mod_status.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule headers_module modules/mod_headers.so
```



Module Details:

mod_status.so

Provides server performance and status information.

mod_log_config.so

Configures and customizes Apache logging, allowing precise control over what information is logged, its format, and its destination.

mod_unique_id.so

Sets environment variables based on client request characteristics the server.

mod_setenvif.so

It is used to set environment variables based on characteristics of the client request.

mod_headers.so

Adds custom headers to server responses.

Step 3: Ensure that the **LogLevel** is set to **warn**. If the **LogLevel** is different, comment out the existing **LogLevel** line.

```
#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn
```

Step 4: Create a new log folder within the Apache home directory and set its permission to '777' using the command "**chmod -R 777"**.

Step 5: Append the following 5 lines to the end of the **log_config_module** section in the configuration file. Do not modify the existing **CustomLog** path within the **log_config_module**.

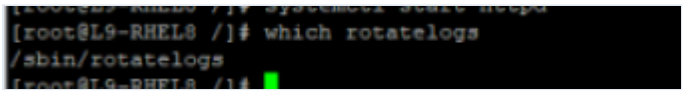
```
TEXT
SetEnvIfNoCase Request_URI "^/server-status/pingFromApplicare" dontlog
SetEnvIfNoCase Request_URI "^/applicare_agent/*" dontlog
SetEnvIf _ap_trace_id ^[a-zA-Z0-9].* HAVE_AP_ID
RequestHeader set _ap_trace_id "%{UNIQUE_ID}e" env=!HAVE_AP_ID
RequestHeader set _ap_apache_id "<Apache_servername>"
CustomLog "|<rotatelogs_path> -n 1440 <Apachelog_directory>/newlogfolder/apl_log 60" "%t %h
%s %B %D %U :%{_ap_trace_id}i" env=!dontlog
```

1. Replace the **<Apache_servername>** with a unique custom name (without space) and including double quotes and without angle brackets. You can choose any name in the **<Apache_servername>.** But it must be unique.

2. The "**_ap_trace_id**" is an environment variable added to each request. Its value will be a random id.

3. Replace "**<rotatelogs_path>** " (removing the angle brackets too) with the path where 'rotatelogs' is installed. On Linux, you can find this path using the following command.

```
TEXT
which rotatelogs (or) sudo which rotatelogs
```



4. Replace **<newlogfolder>**with the name of the folder you created within the Apache log directory (be sure remove the angle brackets).

```
<IfModule log_config_module>
    #
    # The following directives define some format nicknames for use with
    # a CustomLog directive (see below).
    #
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>

    #
    # The location and format of the access logfile (Common Logfile Format).
    # If you do not define any access logfiles within a <VirtualHost>
    # container, they will be logged here.  Contrariwise, if you *do*
    # define per-<VirtualHost> access logfiles, transactions will be
    # logged therein and *not* in this file.
    #
    #CustomLog "logs/access_log" common

    #
    # If you prefer a logfile with access, agent, and referer information
    # (Combined Logfile Format) you can use the following directive.
    #
    CustomLog "logs/access_log" combined

    SetEnvIfNoCase Request_URI "^/server-status/pingFromApplicare" dontlog
    SetEnvIf _ap_trace_id ^[a-zA-Z0-9].* HAVE_AP_ID
    RequestHeader set _ap_trace_id "%{UNIQUE_ID}e" env=!HAVE_AP_ID
    RequestHeader set _ap_apache_id "Apache"
    CustomLog "|/usr/sbin/rotatelogs -n 1440 /var/log/httpd/New_folder/apl_log 60" "%t %h %s %B %D %U :%{_ap_trace_id}i" env=!dontlog
</IfModule>
```

Step 6: Append the following modules to the end of the **conf** file.

```xml
<IfModule mod_status.c>
<Location /server-status>
ProxyPass !
SetHandler server-status
Require local
</Location>
</IfModule>
```

Require local - It will restrict the server status access only from the local machine.

Step 7: Verify the configuration for any Syntax errors. If the output shows **Syntax OK** the configuration is correct.

For httpd

```text
httpd -t
```

For apachectl

```
TEXT
apachectl –t
(or)
apachectl configtest
```

For apache2ctl

```
TEXT
apache2ctl –t
(or)
apache2ctl configtest
```

Step 8: If the configuration is successful, restart the **Apache server.**

After restarting, enter the following request in your browser's URL bar to display information about the Apache server.

To check in browser:

```
XML
<Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto
Example: http://127.0.0.1:80/server-status?auto
```

To check using curl:

```
BASH
curl <Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto
Example: curl http://127.0.0.1:80/server-status?auto
```

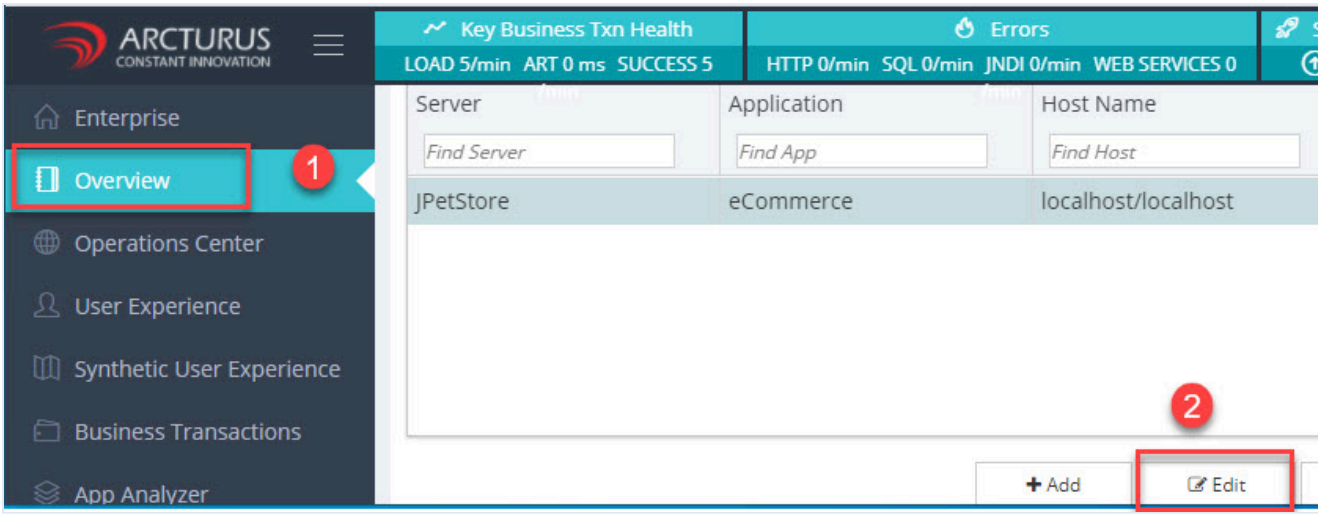
<Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto

<Apache_Protocol> - http (or) https **<Apache_IP> -** Apache server running IP **<Apache_Port> -** Apache server running port

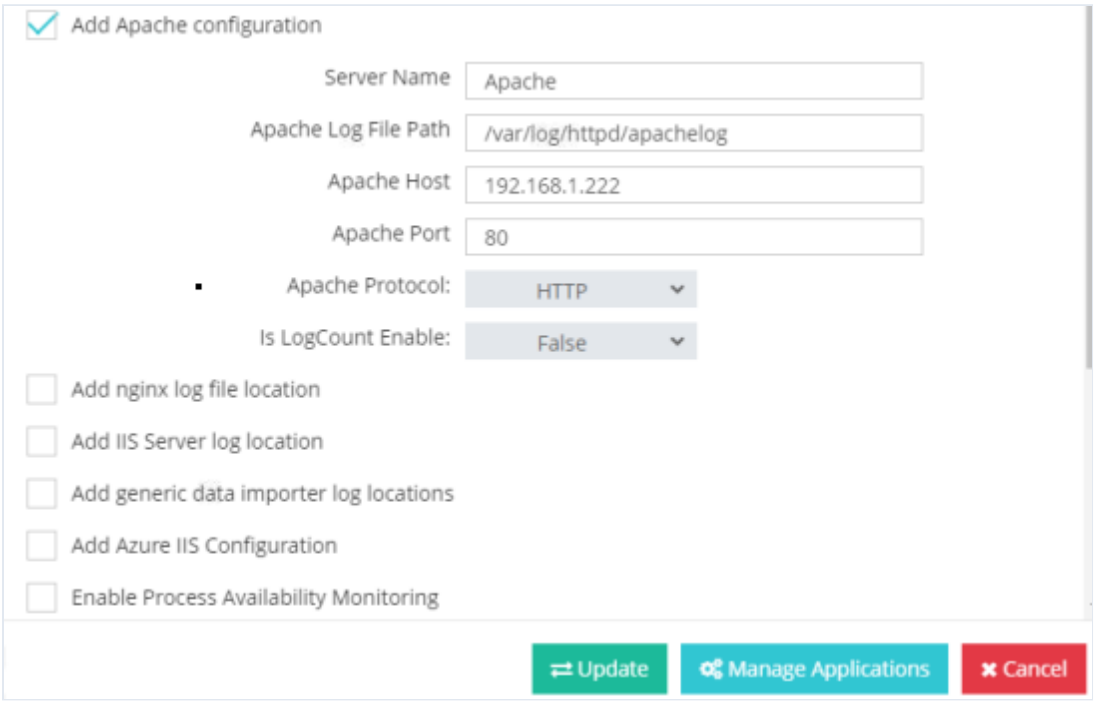
```
YAML
ServerMPM: event
Server Built: Jan 27 2021 07:22:47
CurrentTime: Monday, 28-Jun-2021 18:11:40 EDT
RestartTime: Monday, 28-Jun-2021 18:08:24 EDT
ParentServerConfigGeneration: 1
ParentServerMPMGeneration: 0
ServerUptimeSeconds: 195
ServerUptime: 3 minutes 15 seconds
```

Step 9: After starting the agent, it will be automatically registered with the Applicare controller. Log in to the **Applicare console** and navigate to the **Overview** menu.

Step 10: Locate the newly added agent and click the **Edit** button.



Step 11: Enable the "**Add Apache Configuration**" box and configure the setting as shown below. If the Apache is running in **https** select **HTTPS** in the **Apache Protocol dropdown**. Click the **update** button to save the change.



Note:

1. Other command to check the configuration and virtual hot details of apache server.

httpd -S, apachectl -S, apache2ctl -S and for version **httpd -V**

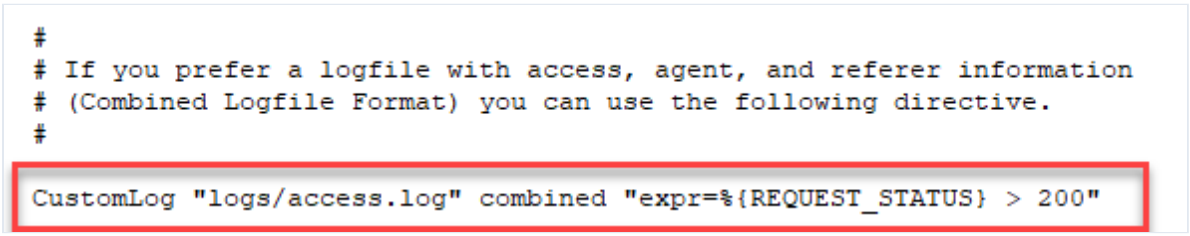
2. If the new log configuration does not take effect, verify if any custom configuration are included in the default configuration. Custom configuration may be included using the "**IncludeOptional**" directive.

3. To monitor only the error logs, append the following syntax to the end of the CustomLog.

```
TEXT
"expr=%{REQUEST_STATUS} > 200"
```

e.g.

CustomLog "logs/access.log" combined **"expr=%{REQUEST_STATUS} > 200"**



```
#
# If you prefer a logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#

CustomLog "logs/access.log" combined "expr=%{REQUEST_STATUS} > 200"
```

4. **To capture only the specific request URL use this.** This will capture only the **/testrequest/order** requests.

SetEnvIfNoCase Request_URI "^/testrequest/order" log_this

CustomLog "logs/access.log" combined **env=log_this**

Monitoring with Applicare

CONFIGURATION GUIDES › WEB SERVERS

Apache server 2.4 Monitoring for windows

Applicare 9.2 · Updated August 22, 2024 · apache · **Pending technical review**

Follow the steps for installing Applicare SingleAgent

Step 1: To enable server-status and custom log modules in Apache by navigating to the Apache installed directory and editing the conf file.

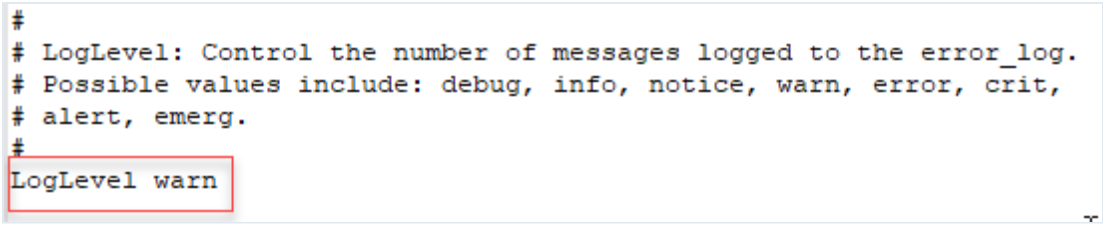
Step 2: Ensure the following modules are enabled in the conf file. If they are not, enable them by removing '#' at the beginnings of the lines.

Some of the modules may not be present in the Apache conf file by default. In that cases, manually add the following modules below the Apache Listen line in the Apache conf file.

```text
#Listen 12.34.56.78:80
Listen 80

LoadModule status_module modules/mod_status.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule headers_module modules/mod_headers.so
```

Step 3: Ensure the **LogLevel** is set to **warn**. If it is set to a **LogLevel** is different value, comment out existing **LogLevel** line.



Step 4: Create a new log folder within the Apache home directory.

Step 5: Append the following five lines to the end of the **log_config_module** and section in the **CustomLog** and leave the existing **log_config_module** path unchanged.

```
TEXT
SetEnvIfNoCase Request_URI "^/server-status/pingFromApplicare" dontlog
SetEnvIfNoCase Request_URI "^/applicare_agent/*" dontlog
SetEnvIf _ap_trace_id ^[a-zA-Z0-9].* HAVE_AP_ID
RequestHeader set _ap_trace_id "%{UNIQUE_ID}e" env=!HAVE_AP_ID
RequestHeader set _ap_apache_id "<Apache_servername>"
CustomLog "|<RotateLogPath> -n 1440 <ApacheInstalledDir>/newlogfolder/apl_log 60" "%t %h %s
%B %D %U :%{_ap_trace_id}i" env=!dontlog
```

1. Replace **<Apache_servername>** with a custom name (without spaces) and include double quotes **(Need to remove the angle brackets too)**. The name can be **<Apache_servername>**. But it must be unique.

2. Replace **<RotateLogPath>** with the path to your **rotatelogs.exe** file. **(and remove the angle brackets).** By default, it is located in the Apache installation directory **<ApacheInstalledDir>/bin/rotatelogs.exe**

3. Replace **"<ApacheInstalledDir>"** with the path to your Apache installation **(and remove the angle brackets).**

4. Replace **<newlogfolder>**with the name of the folder you created inside the Apache installation directory **(add remove the angle brackets too)**.

```
<IfModule log_config_module>
    #
    # The following directives define some format nicknames for use with
    # a CustomLog directive (see below).
    #
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>

    #
    # The location and format of the access logfile (Common Logfile Format).
    # If you do not define any access logfiles within a <VirtualHost>
    # container, they will be logged here.  Contrariwise, if you *do*
    # define per-<VirtualHost> access logfiles, transactions will be
    # logged therein and *not* in this file.
    #
    #CustomLog "logs/access.log" common

    #
    # If you prefer a logfile with access, agent, and referer information
    # (Combined Logfile Format) you can use the following directive.
    #

    CustomLog "logs/access.log" combined "expr=%{REQUEST_STATUS} > 200"
    #CustomLog "logs/access.log" combined

    SetEnvIfNoCase Request_URI "^/server-status/pingFromApplicare" dontlog
    SetEnvIf _ap_trace_id ^[a-zA-Z0-9].* HAVE_AP_ID
    RequestHeader set _ap_trace_id "%{UNIQUE_ID}e" env=!HAVE_AP_ID
    RequestHeader set _ap_apache_id "Apache"
    CustomLog "|E:/Apache/Apache24/bin/rotatelogs.exe -n 1440 E:/Apache/Apache24/newlogfolder/apl_log 86400" "%t %h %s %B %D %U :%{_ap_trace_id}i" env=!dontlog
</IfModule>
```

Step 6: Append the following modules to the end of the **conf** file.

```xml
<IfModule mod_status.c>
<Location /server-status>
ProxyPass !
SetHandler server-status
Require all granted
</Location>
</IfModule>
```

Step 7: Verify the configuration for syntax errors. If the output shows **Syntax OK,** the configuration is correct.

For httpd

```text
httpd -t
```

Step 8: Finally, **Restart Apache server**.

After restarting the Apache server, Enter the following URL in your browser. This will display information about the apache server.

To check in browser:

```xml
<Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto
Example: http://127.0.0.1:80/server-status?auto
```

To check using curl:

```bash
curl <Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto
Example: curl http://127.0.0.1:80/server-status?auto
```

<Apache_Protocol>://<Apache_IP>:<Apache_Port>/server-status?auto

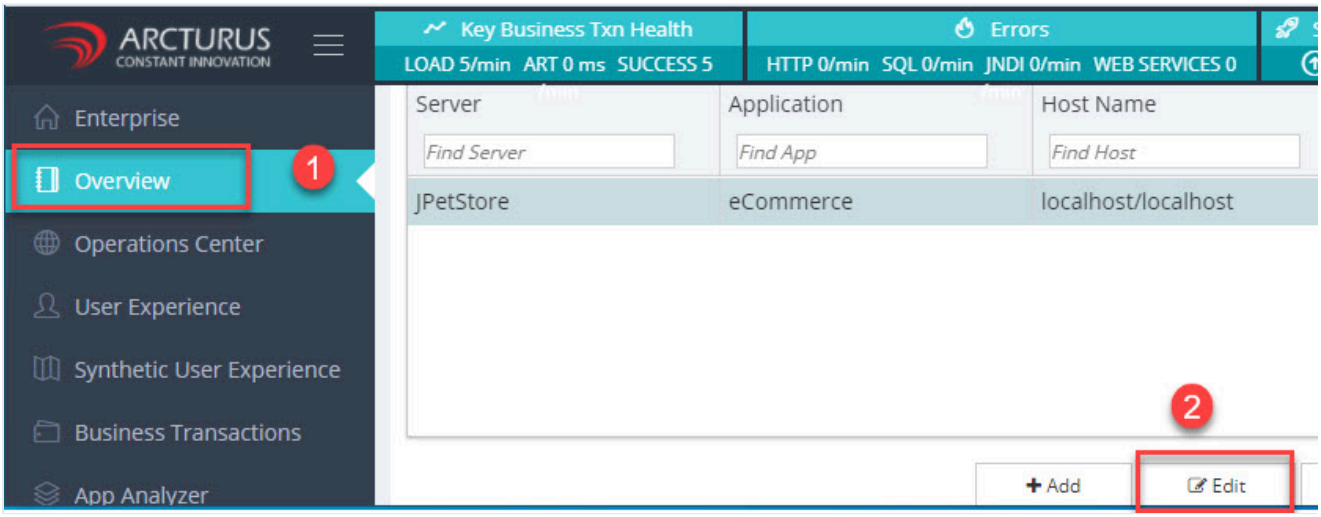
<Apache_Protocol> - http (or) https **<Apache_IP> -** Apache server running IP **<Apache_Port> -** Apache server running port

```yaml
ServerMPM: event
Server Built: Jan 27 2021 07:22:47
CurrentTime: Monday, 28-Jun-2021 18:11:40 EDT
RestartTime: Monday, 28-Jun-2021 18:08:24 EDT
ParentServerConfigGeneration: 1
ParentServerMPMGeneration: 0
ServerUptimeSeconds: 195
ServerUptime: 3 minutes 15 seconds
```

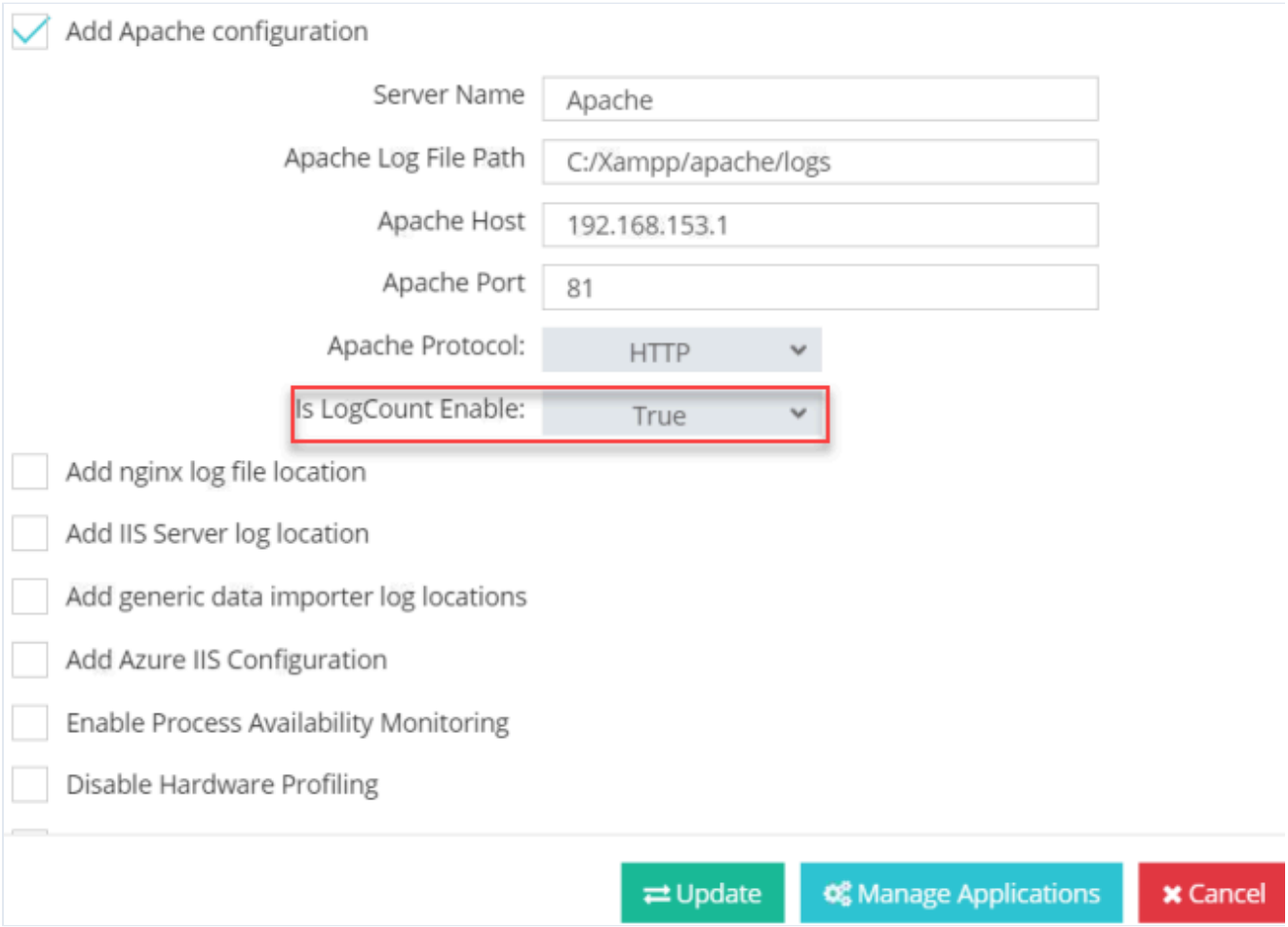
Step 9: Start the agent by executing the **startAgent.bat** in the **<Agent_Home_Directory>**.

Step 10: After the agent starts, It will be automatically added to the Applicare controller. Log in to the **Applicare console** and navigate to the **Overview** menu.

Step 11: Choose the newly added agent and click the **Edit** button.



Step 12: Check the "**Add Apache Configuration**" box and Set the configure the setting as shown as below. If the Apache is running over **HTTPS,** select **HTTPS** from the **Apache Protocol dropdown**. Click the **update** button to save the change.



1. Use the following command to check the configuration and virtual host details of Apache server.

httpd -S and for version **httpd -V**

2. If the new log configuration isn't taking effect, check whether a custom configuration is being referenced in the default configuration. Custom configuration may be included using the "**IncludeOptional**" keyword.

3. If the customer is wants to log only error logs to their existing log files. Add the following syntax to the end of their CustomLog configuration.

```
TEXT
"expr=%{REQUEST_STATUS} > 200"
```

e.g.

CustomLog "logs/access.log" combined **"expr=%{REQUEST_STATUS} > 200"**

```
#
# If you prefer a logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#

CustomLog "logs/access.log" combined "expr=%{REQUEST_STATUS} > 200"
```

Docs home

CONFIGURATION GUIDES › WEB SERVERS

Monitoring Nginx

Applicare 9.2 · Updated May 9, 2026 · nginx · **Pending technical review**

Step 1: Follow the steps for Installing Applicare SingleAgent

Nginx Configuration

Step 2: Stop the Nginx server.

Step 3: Navigate to the directory **<Nginx_Installed_Directory>/conf** and open the **nginx.conf** file for editing. On Linux, the default path is (**/etc/nginx**).

Step 4: Comment out the current format and insert the following **log_format**.

log_format main '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" ' '$status $body_bytes_sent || "$request_body" || "$http_referer" ' '"$http_user_agent" "$http_x_forwarded_for" ' '"$scheme://$host:$server_port"';

```
log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                  '$status $body_bytes_sent ||  "$request_body" || "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

access_log  logs/access.log  main;
```

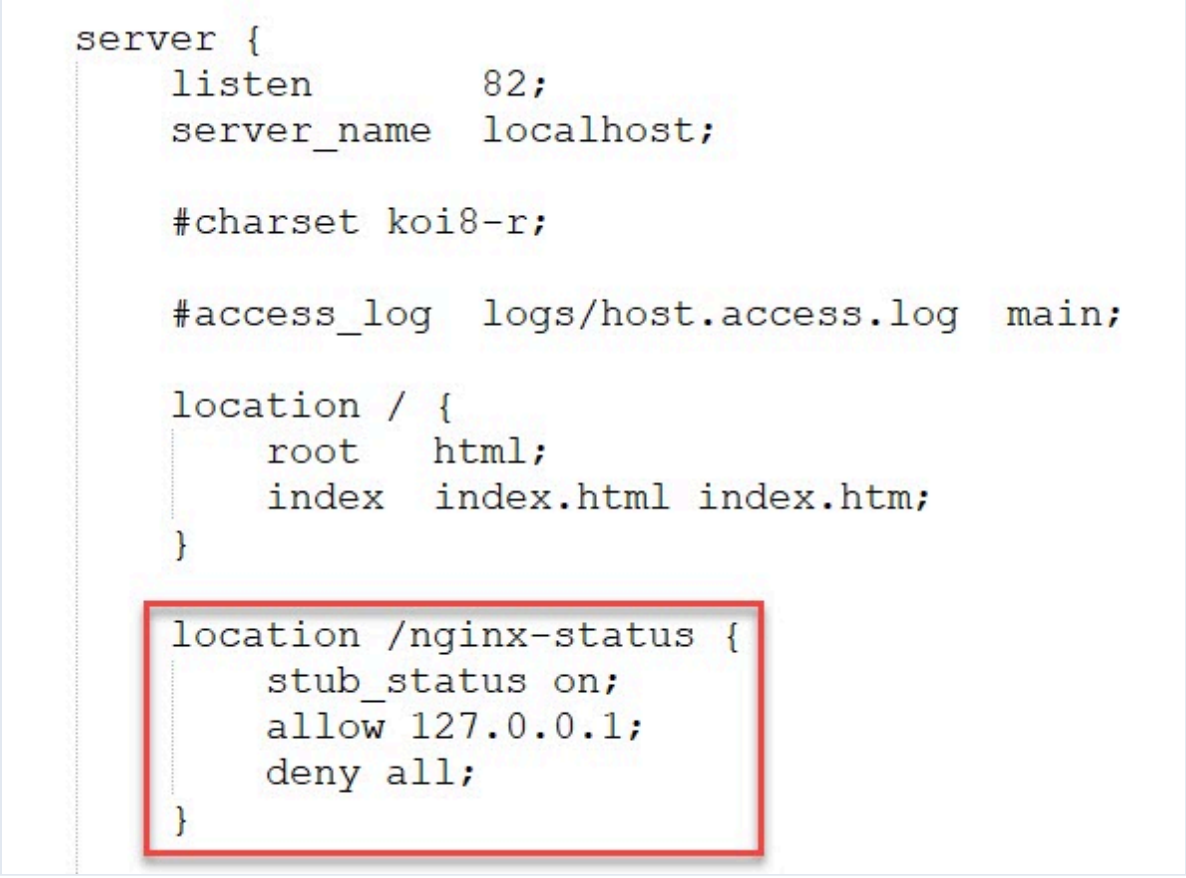
Step 5: If the **access_log** line is commented out, Please uncomment it.

```
log_format  main  '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" '
                  '$status $body_bytes_sent ||  "$request_body" || "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

access_log  logs/access.log  main;
```

Step 7: Insert the following lines within the **server** module

location /nginx-status { stub_status on; allow 127.0.0.1; deny all; }



```
server {
    listen       82;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }

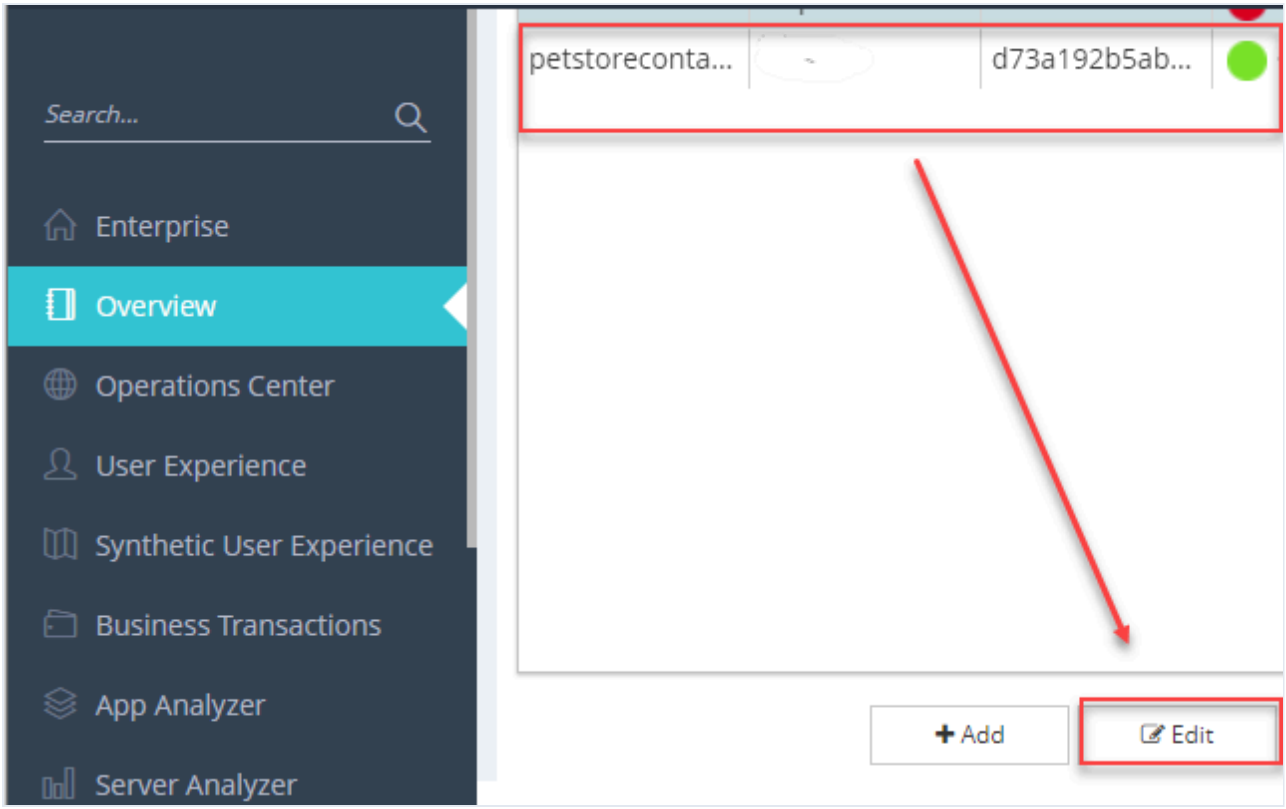
    location /nginx-status {
        stub_status on;
        allow 127.0.0.1;
        deny all;
    }
```

Step 8: Start the Nginx server.

Integrating Nginx to Applicare

Step 9: Once the agent is started, It will be automatically registered with the Applicare controller. Log in to the **Applicare console** and go to the **Overview** menu.

Step 10: Select the newly added agent and click the **Edit** button.



Step 11: Enable the "**Add Nginx Configuration**" checkbox and add the below value.

1. Log file path

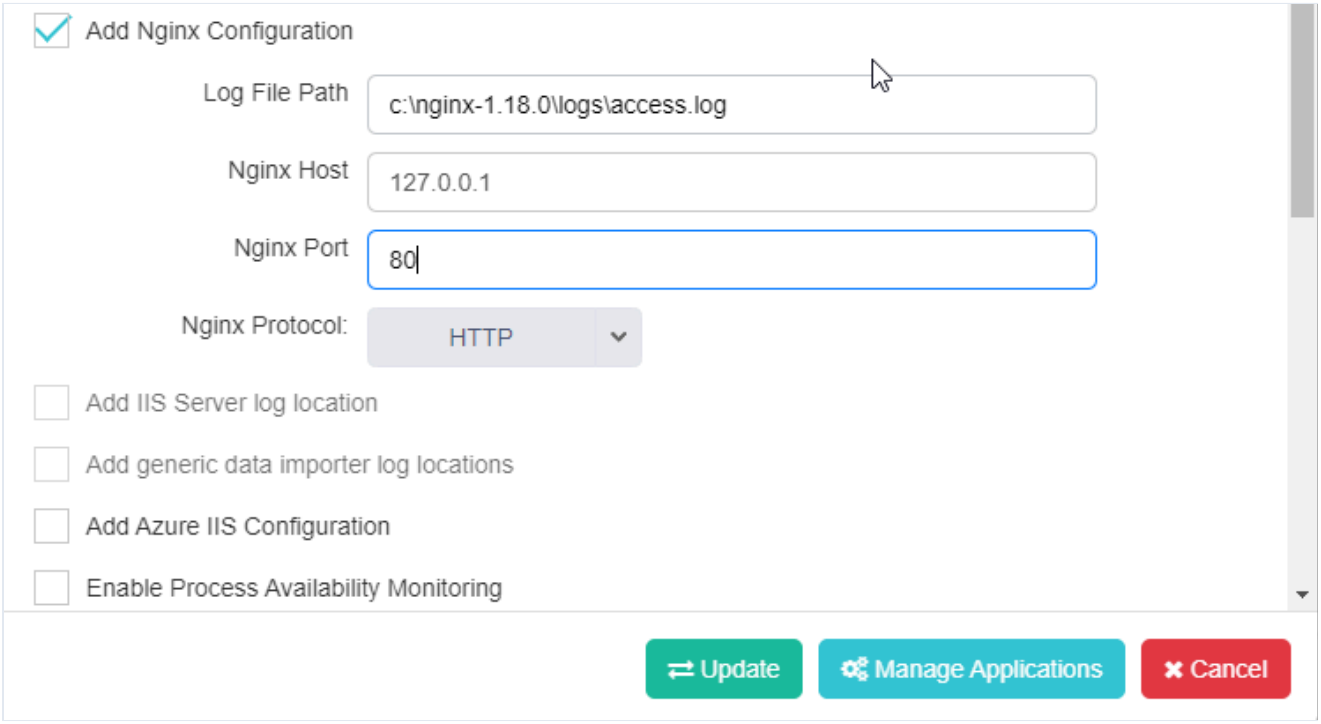
2. Nginx Host

3. Nginx Port

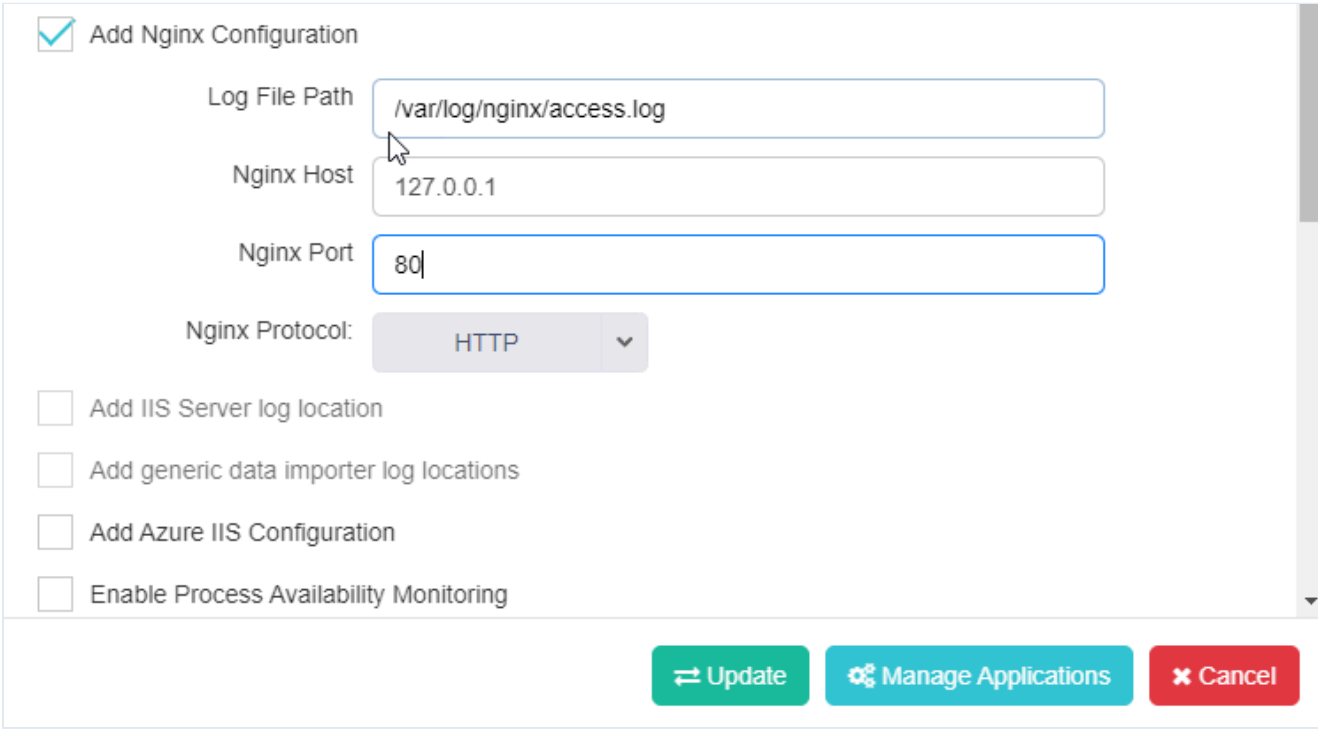
4. Nginx Protocol

Click the **update** button to save.

Windows



Linux



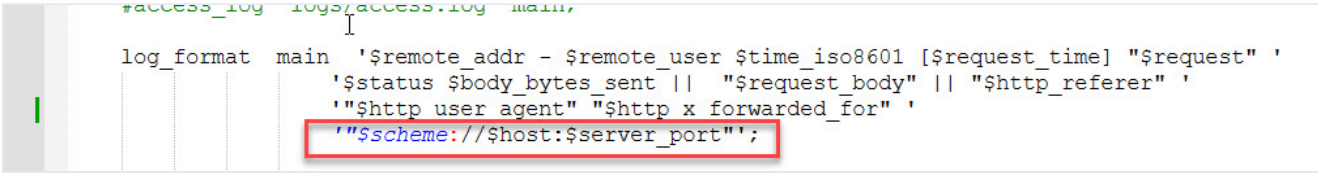
Note:

1. Configuration to capture the base url and save and restart the NGINX.

Remove the **semicolon** after the **"$http_x_forwarded_for"'** keyword and add the **'"$scheme://$host:$server_port"';** in the new line.

log_format main '$remote_addr - $remote_user $time_iso8601 [$request_time] "$request" ' '$status $body_bytes_sent || "$request_body" || "$http_referer" ' '"$http_user_agent" "$http_x_forwarded_for"'

```
TEXT
                '"$scheme://$host:$server\_port"';
```



Docs Home

CONFIGURATION GUIDES › WEB SERVERS

Start Nginx using Task Scheduler

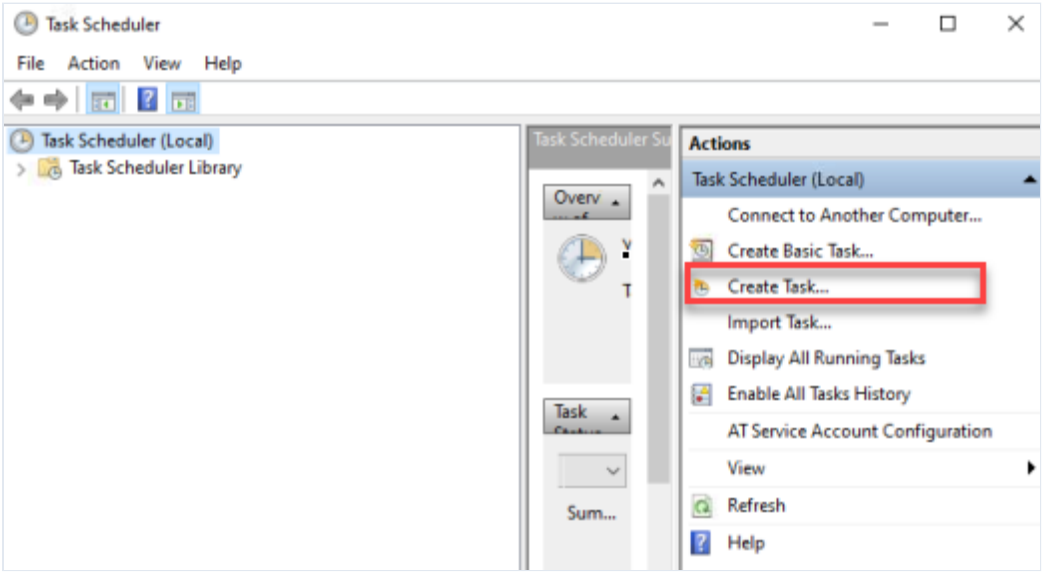
Applicare 9.2 · Updated September 13, 2024 · nginx · **Pending technical review**

Follow the steps below to start Nginx using Task Scheduler.

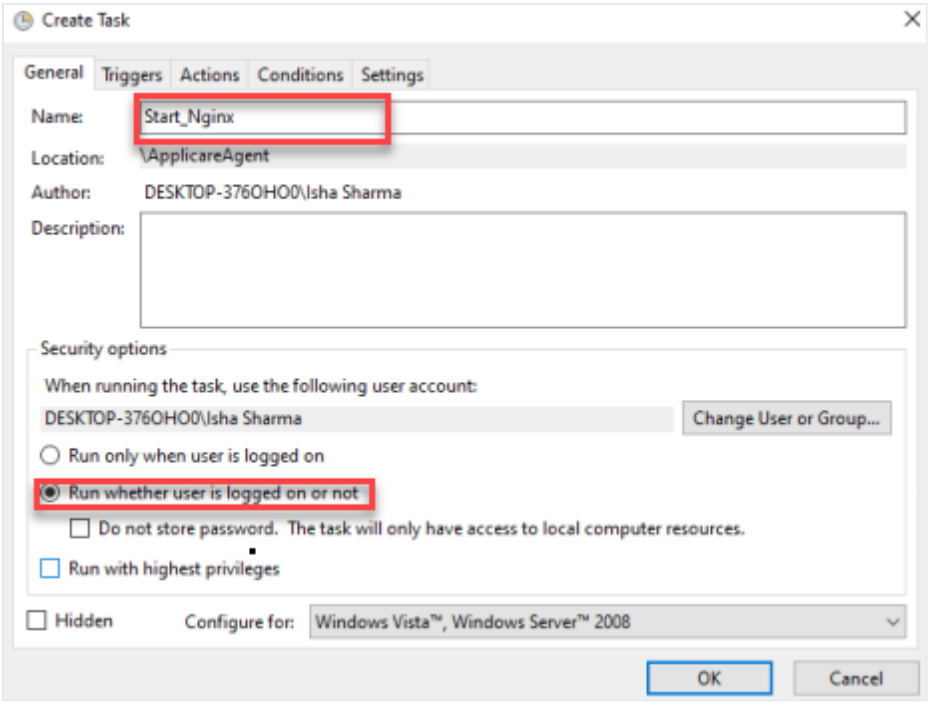
Steps 1: Create a **startn.bat** file with the following content in the folder where Nginx is extracted.

```text
start nginx.exe
```

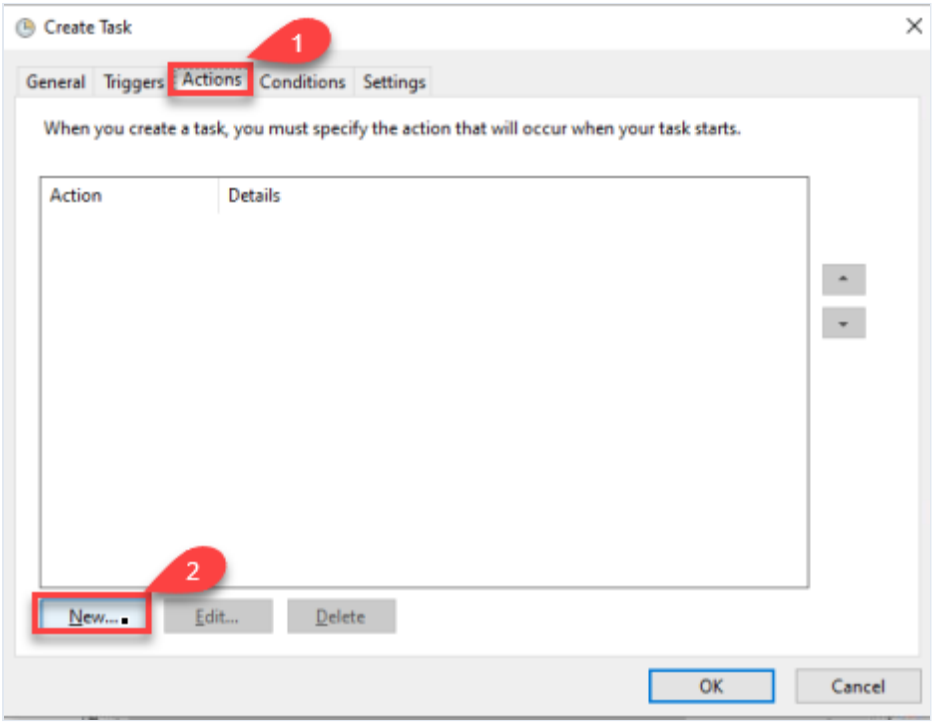
Steps 2: Open Task Scheduler and click **Create Task**.



Steps 3: Enter a name for the task and select the **Run whether the user is logged on or not**.



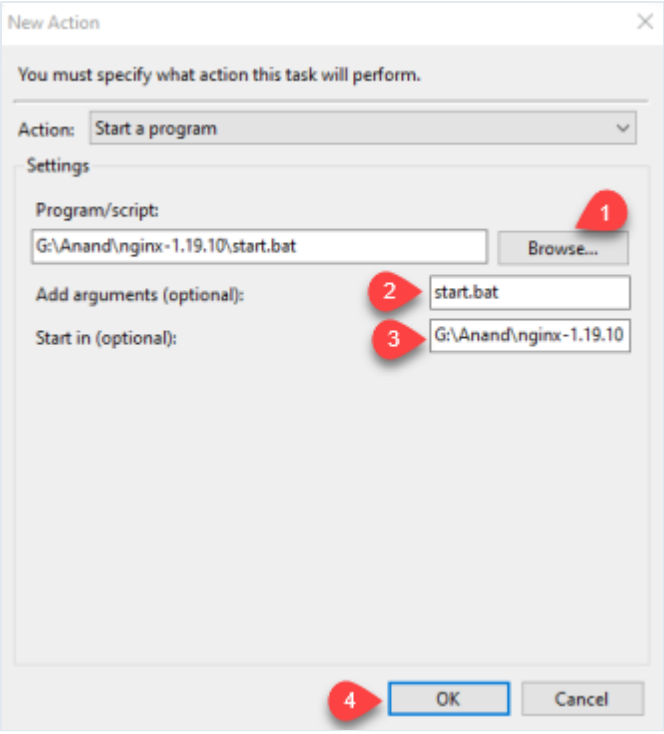
Steps 4: Go to the **Action** tab click the **New** button.



Steps 5: Click the Browse button to select the **startn.bat** file. Enter the values in the **Add arguments** and **start in** fields, then click **OK**.

Add arguments: startn.bat

start in: <Nginx_extracted_directory>



Steps 6: Finally, click **OK,** enter the **user credential** and save the task.

To stop Nginx, navigate to the directory where Nginx is extracted and execute the following command.

```
TEXT
nginx –s stop
```

Monitoring with Applicare

CONFIGURATION GUIDES › WEB SERVERS

Monitoring Applications Hosted on IIS with Applicare

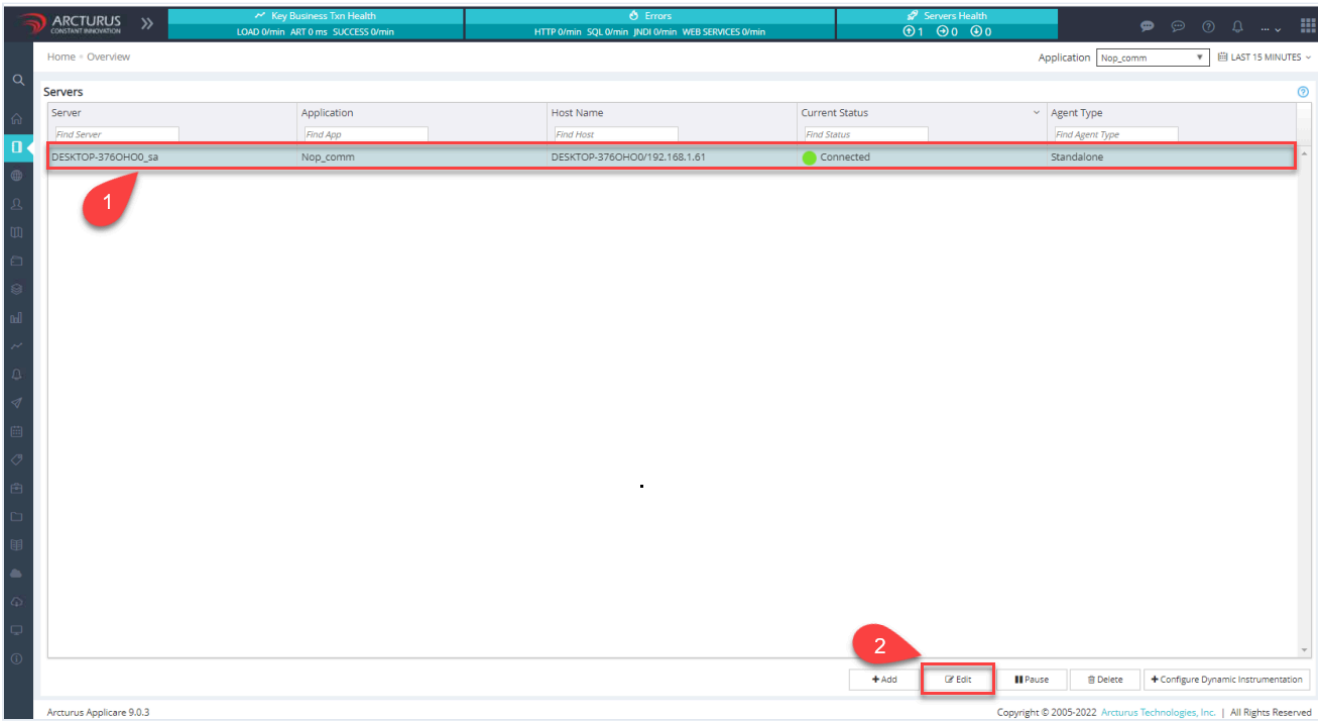
Applicare 9.2 · Updated September 13, 2024 · iis, windows · **Pending technical review**

1. Follow the steps for **Installing Applicare SingleAgent.**

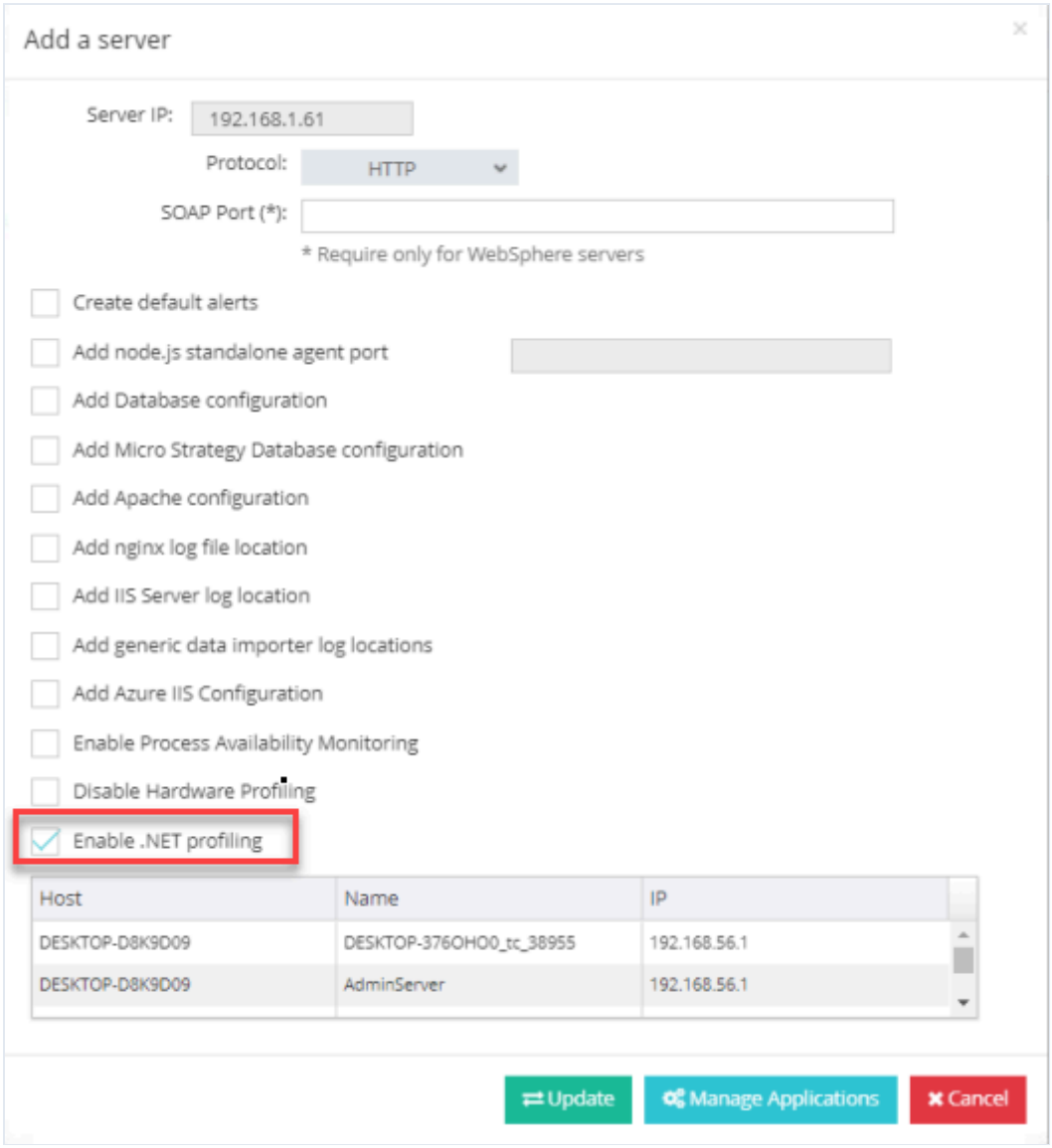
2. Navigate to the folder where the ApplicareSingleAgent.zip was extracted from and then run **startAgent.bat** in administrator mode.

3. Login to the Applicare controller, Navigate to the overview screen, and check if the agents appear. If needed, add the app name to that agent.

4. Click the agent and then the edit button.



5. Click the edit button to open the "Add Server" popup. In the popup, check the "Enable .NET Profiling" checkbox.



Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

Automate Daily Backups of Settings Tables Records to an Archive Database

Applicare 9.2 · Updated February 12, 2026 · mysql · **Pending technical review**

Step 1: Before backing up the database, create the **applicarearchive** schema, execute the applicare-related sql's and ensure all tables are created.

Step 2: Connect to the database via **MySQL Workbench**, open a new SQL query tab, and execute the following command.

NOTE: Replace the <targetDB> placeholder with actual backup database name before running the query.

YAML

```
DELIMITER //
DROP PROCEDURE IF EXISTS applicarearchive.sp_MasterArchive_Settings //
CREATE PROCEDURE applicarearchive.sp_MasterArchive_Settings()
BEGIN
  DECLARE v_done INT DEFAULT FALSE;
  DECLARE v_table_name VARCHAR(255);
  DECLARE v_source_db VARCHAR(100) DEFAULT '<targetDB>';
  DECLARE v_backup_db VARCHAR(100) DEFAULT 'applicarearchive';
  DECLARE v_columns TEXT;

  -- Full Table List
  DECLARE table_cursor CURSOR FOR
      SELECT tname FROM (
          SELECT 'mg_dash_config' AS tname UNION SELECT 'mg_dash_exclude_api' UNION SELECT
'mg_dash_property' UNION
          SELECT 'server' UNION SELECT 'application' UNION SELECT 'settings' UNION SELECT
'user_settings' UNION
          SELECT 'configs_modified_time' UNION SELECT 'aop_config' UNION SELECT
'op_center_config' UNION
          SELECT 'eum_url' UNION SELECT 'monitor_config' UNION SELECT 'btm_txn_settings'
UNION
          SELECT 'business_transaction' UNION SELECT 'web_transaction' UNION SELECT
'java_transaction' UNION
          SELECT 'processes_to_monitor' UNION SELECT 'lamaconfig' UNION SELECT
'lamaconfig_apps' UNION
          SELECT 'lamaconfig_snmpapps' UNION SELECT 'lama_errors' UNION SELECT
'lama_holidays' UNION
          SELECT 'apiconfig' UNION SELECT 'api_threshold' UNION SELECT 'api_exclude_api'
UNION
          SELECT 'api_count_alert_config' UNION SELECT 'mbean_prop_identifier' UNION
          SELECT 'runtime_mbean_properties' UNION SELECT 'runtime_mbean_types' UNION
          SELECT 'mbean_prop_persisted' UNION SELECT 'alert_definitions' UNION SELECT
'reports_schedule' UNION
          SELECT 'reports_scheduler' UNION SELECT 'log_monitor' UNION SELECT 'log_alerts'
UNION
          SELECT 'log_file' UNION SELECT 'log_patterns' UNION SELECT 'login_stat' UNION
          SELECT 'user_actions' UNION SELECT 'user_applications' UNION SELECT 'user_roles'
UNION
          SELECT 'user_subscription' UNION SELECT 'users' UNION SELECT 'users_login_info'
UNION
          SELECT 'ldap_user' UNION SELECT 'ldap_user_applications' UNION SELECT
'ldap_user_role' UNION
          SELECT 'single_sign_user' UNION SELECT 'single_sign_user_applications' UNION
          SELECT 'single_sign_user_role' UNION SELECT 'server_availability_config' UNION
```

```
          SELECT 'server_details' UNION SELECT 'server_domain' UNION SELECT 'session_info'
UNION
          SELECT 'kubernetes_config' UNION SELECT 'aws_user'
      ) mapping;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_done = TRUE;
  -- Ensure Log table exists
  CREATE TABLE IF NOT EXISTS applicarearchive.archive_log (
      id INT AUTO_INCREMENT PRIMARY KEY,
      table_name VARCHAR(255),
      rows_moved INT,
      sync_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
      status VARCHAR(255)
  );
  SET FOREIGN_KEY_CHECKS = 0;
  OPEN table_cursor;
  read_loop: LOOP
      FETCH table_cursor INTO v_table_name;
      IF v_done THEN LEAVE read_loop; END IF;
      -- RESET columns variable for each loop
      SET v_columns = NULL;
      -- FIND MATCHING COLUMNS (Using LOWER for maximum compatibility)
      SELECT GROUP_CONCAT(CONCAT('`', src.COLUMN_NAME, '`')) INTO v_columns
      FROM information_schema.columns src
      WHERE src.table_schema = v_source_db
        AND LOWER(src.table_name) = LOWER(v_table_name)
        AND src.COLUMN_NAME IN (
            SELECT COLUMN_NAME
            FROM information_schema.columns
            WHERE table_schema = v_backup_db
              AND LOWER(table_name) = LOWER(v_table_name)
        );
      IF v_columns IS NOT NULL AND v_columns != '' THEN
          -- 1. TRUNCATE archive
          SET @truncate_sql = CONCAT('TRUNCATE TABLE `', v_backup_db, '`.`', v_table_name,
'`');
          PREPARE t_stmt FROM @truncate_sql;
          EXECUTE t_stmt;
          DEALLOCATE PREPARE t_stmt;
          -- 2. INSERT fresh data
          SET @move_sql = CONCAT(
              'INSERT INTO `', v_backup_db, '`.`', v_table_name, '` (', v_columns, ') ',
              ' SELECT ', v_columns, ' FROM `', v_source_db, '`.`', v_table_name, '`'
          );
          PREPARE m_stmt FROM @move_sql;
          EXECUTE m_stmt;
```

```
            SET @rows_affected = ROW_COUNT();
            DEALLOCATE PREPARE m_stmt;
            INSERT INTO applicarearchive.archive_log (table_name, rows_moved, status)
            VALUES (v_table_name, @rows_affected, 'SUCCESS');
        ELSE
            -- DETAILED ERROR LOGGING
            INSERT INTO applicarearchive.archive_log (table_name, rows_moved, status)
            VALUES (v_table_name, 0, 'FAIL: Could not find matching table/columns in Archive');
        END IF;
    END LOOP;
    CLOSE table_cursor;
    SET FOREIGN_KEY_CHECKS = 1;
END //
DELIMITER ;
```

NOTE: The query above only backs up data from the settings tables, rather than the entire database.

Step 3: Run the query below to enable the MySQL Event Scheduler and schedule the **sp_BatchArchiveYesterday** procedure to run daily at 1:00 AM.

```
TEXT
SET GLOBAL event_scheduler = ON;
DELIMITER //
DROP EVENT IF EXISTS applicarearchive.setting_table_daily_archive_at_1am; //
CREATE EVENT applicarearchive.setting_table_daily_archive_at_1am
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 1 HOUR)
DO
  BEGIN
    TRUNCATE TABLE applicarearchive.archive_log;
    CALL applicarearchive.sp_MasterArchive_Settings();
  END //

DELIMITER ;
```

To modify the scheduled time, first execute the query below to drop the existing event, then re-create it with your desired schedule.

```
TEXT
DROP EVENT IF EXISTS applicarearchivesettings.daily_archive_at_1am;
```

To change the time to 8:00 PM, We need to adjust the `INTERVAL` in the `STARTS` clause. Since MySQL uses a 24-hour format, 8:00 PM is 20 hours.

E.g.

```
TEXT
SET GLOBAL event_scheduler = ON;
DELIMITER //
DROP EVENT IF EXISTS applicarearchive.setting_table_daily_archive_at_8pm //
CREATE EVENT applicarearchive.setting_table_daily_archive_at_8pm
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 20 HOUR)
DO
  BEGIN
    TRUNCATE TABLE applicarearchive.archive_log;
    CALL applicarearchive.sp_MasterArchive_Settings();
  END //

DELIMITER ;
```

NOTE: Execute the below query to confirm the event is saved in the system.

```
TEXT
SHOW EVENTS FROM applicarearchive;
```

Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Starts	Ends	Status	Originator	character_set_client	collation_connection	Database Collation
applicarearchive	setting_table_daily_archive_at_8pm	applicare@%	SYSTEM	RECURRING	NULL	1	DAY	2026-02-13 20:00:00	NULL	ENABLED	1	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

Step 4: We need to create the `archive_log` table in your archive database. This table will record which table was processed, how many rows were moved, and exactly when it finished.

```
TEXT
CREATE TABLE IF NOT EXISTS applicarearchive.archive_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    table_name VARCHAR(255),
    rows_moved INT,
    finish_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(255)
);
```

Explanation of the Columns:

- **id**: A unique number for every entry.
- **table_name**: Records the table name (and the column name used, e.g., `oshistory_detail [dt]`).
- **rows_moved**: The number of records successfully transferred for "Yesterday".
- **finish_time**: The exact date and time the backup for that specific table completed.

- **status:** The `status` column is your best friend. If a table fails because of a schema change (like a missing column in the archive), you will see "FAIL: Could not find matching table/columns" immediately.

Step 5: Execute query to see the results. It will show you the most recent backups at the top.

```
TEXT
SELECT * FROM applicarearchive.archive_log ORDER BY finish_time DESC;
```



id	table_name	rows_moved	finish_time	status
10	lama_db_latency (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
18	lama_networks_avg (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
17	lama_network_metrics (via recorded)	0	2026-02-12 13:24:25	NO NEW DATA
16	lama_holidays (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
15	lama_hardware_metrics (via recorded)	0	2026-02-12 13:24:25	NO NEW DATA
14	lama_errors (via excep_at)	0	2026-02-12 13:24:25	NO NEW DATA
13	lama_disk_details (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
12	lama_db_status (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
11	lama_db_queue (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
1	intellitrace_errors (via excep_at)	0	2026-02-12 13:24:25	NO NEW DATA
9	lama_db_bandwidth (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
8	lama_database_metrics (via recorded)	0	2026-02-12 13:24:25	NO NEW DATA
7	lama_application_metrics (via recorded)	0	2026-02-12 13:24:25	NO NEW DATA
6	int_profiling_hist (via from_time)	0	2026-02-12 13:24:25	NO NEW DATA
5	business_txn_avg (via dt)	0	2026-02-12 13:24:25	NO NEW DATA
4	monitor_data (via dt)	0	2026-02-12 13:24:25	NO NEW DATA

archive_log 21 ×

CONFIGURATION GUIDES › DATABASES

Automate Daily Batch Backups of few Tables Records to an Archive Database

Applicare 9.2 · Updated March 27, 2026 · mysql · **Pending technical review**

Step 1: Before backing up the database, create the **applicarearchive** schema, execute the applicare-related sql's and ensure all tables are created.

Step 2: Connect to the database via **MySQL Workbench**, open a new SQL query tab, and execute the following command.

NOTE: Replace the <targetDB> placeholder with actual backup database name before running the query.

YAML

```
DELIMITER //
DROP PROCEDURE IF EXISTS applicarearchive.sp_MasterArchive_Int_Profiling //
CREATE PROCEDURE applicarearchive.sp_MasterArchive_Int_Profiling()
BEGIN
   DECLARE v_done INT DEFAULT FALSE;
   DECLARE v_table_name VARCHAR(255);
   DECLARE v_id_col VARCHAR(100);
   DECLARE v_time_col VARCHAR(100);
   DECLARE v_source_db VARCHAR(100) DEFAULT '<targetDB>';
   DECLARE v_backup_db VARCHAR(100) DEFAULT 'applicarearchive';

   -- DATA MAP for selected tables only
   DECLARE table_cursor CURSOR FOR
       SELECT * FROM (
           SELECT 'intellitrace_errors' as t, 'id' as i, 'excep_at' as c UNION
           SELECT 'alert_queue', 'id', 'created_time' UNION
           SELECT 'int_profiling_hist', 'rl', 'from_time'
       ) mapping;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_done = TRUE;

   -- Ensure custom log table exists
   CREATE TABLE IF NOT EXISTS applicarearchive.archive_logcustom (
       id INT AUTO_INCREMENT PRIMARY KEY,
       table_name VARCHAR(255),
       rows_moved INT,
       sync_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
       status VARCHAR(255)
   );

   -- Target exactly 31 days ago (30 days before yesterday)
   SET @start_val = CONCAT(CURDATE() - INTERVAL 38 DAY, ' 00:00:00');
   SET @end_val   = CONCAT(CURDATE() - INTERVAL 38 DAY, ' 23:59:59');

   OPEN table_cursor;
   read_loop: LOOP
       FETCH table_cursor INTO v_table_name, v_id_col, v_time_col;
       IF v_done THEN LEAVE read_loop; END IF;

       SET @rows_affected = 1;
       SET @total_moved = 0;

       -- Move data in batches of 10,000 to prevent lag/locking
       WHILE @rows_affected > 0 DO
```

```
            SET @move_sql = CONCAT(
                'INSERT INTO ', v_backup_db, '.', v_table_name,
                ' SELECT * FROM ', v_source_db, '.', v_table_name,
                ' WHERE ', v_time_col, ' BETWEEN ? AND ? ',
                ' AND ', v_id_col, ' NOT IN (SELECT ', v_id_col, ' FROM ', v_backup_db, '.',
v_table_name, ') ',
                ' LIMIT 10000'
            );
            PREPARE m_stmt FROM @move_sql;
            EXECUTE m_stmt USING @start_val, @end_val;
            SET @rows_affected = ROW_COUNT();
            SET @total_moved = @total_moved + GREATEST(0, @rows_affected);
            DEALLOCATE PREPARE m_stmt;
        END WHILE;

        -- Log result into custom log table
        INSERT INTO applicarearchive.archive_logcustom (table_name, rows_moved, status)
        VALUES (
            CONCAT(v_table_name, ' (via ', v_time_col, ')'),
            @total_moved,
            IF(@total_moved > 0, 'SUCCESS', 'NO NEW DATA')
        );
    END LOOP;
    CLOSE table_cursor;
END //
DELIMITER ;
```

NOTE: The query above only backs up data from the few tables, rather than the entire database.

If you want to run the above stored procedure manually, execute the query below.

```
TEXT
CALL applicarearchive.sp_MasterArchive_Int_Profiling();
```

Step 3: Run the query below to enable the MySQL Event Scheduler and schedule the **sp_BatchArchiveYesterday** procedure to run daily at 1:00 AM.

```
TEXT
SET GLOBAL event_scheduler = ON;
DELIMITER //
DROP EVENT IF EXISTS applicarearchive.daily_archive_at_1am //
CREATE EVENT applicarearchive.daily_archive_at_1am
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 1 HOUR)
DO
  BEGIN
    TRUNCATE TABLE applicarearchive.archive_log;
    CALL applicarearchive.sp_MasterArchive_Final();
  END //

DELIMITER ;
```

To modify the scheduled time, first execute the query below to drop the existing event, then re-create it with your desired schedule.

```
TEXT
DROP EVENT IF EXISTS applicarearchive.daily_archive_at_1am;
```

To change the time to 8:00 PM, We need to adjust the `INTERVAL` in the `STARTS` clause. Since MySQL uses a 24-hour format, 8:00 PM is 20 hours.

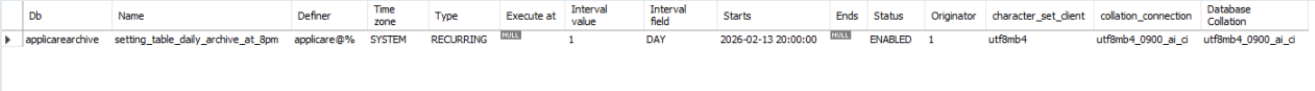
e.g.

```
TEXT
SET GLOBAL event_scheduler = ON;
DELIMITER //
DROP EVENT IF EXISTS applicarearchive.daily_archive_at_8pm //
CREATE EVENT applicarearchive.daily_archive_at_8pm
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 20 HOUR)
DO
  BEGIN
    TRUNCATE TABLE applicarearchive.archive_log;
    CALL applicarearchive.sp_MasterArchive_Final();
  END //

DELIMITER ;
```

NOTE: Execute the below query to confirm the event is saved in the system.

```
TEXT
SHOW EVENTS FROM applicarearchive;
```



Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Starts	Ends	Status	Originator	character_set_client	collation_connection	Database Collation
applicarearchive	setting_table_daily_archive_at_8pm	applicare@%	SYSTEM	RECURRING	NULL	1	DAY	2026-02-13 20:00:00	NULL	ENABLED	1	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

Step 4: We need to create the `archive_log` table in your archive database. This table will record which table was processed, how many rows were moved, and exactly when it finished.

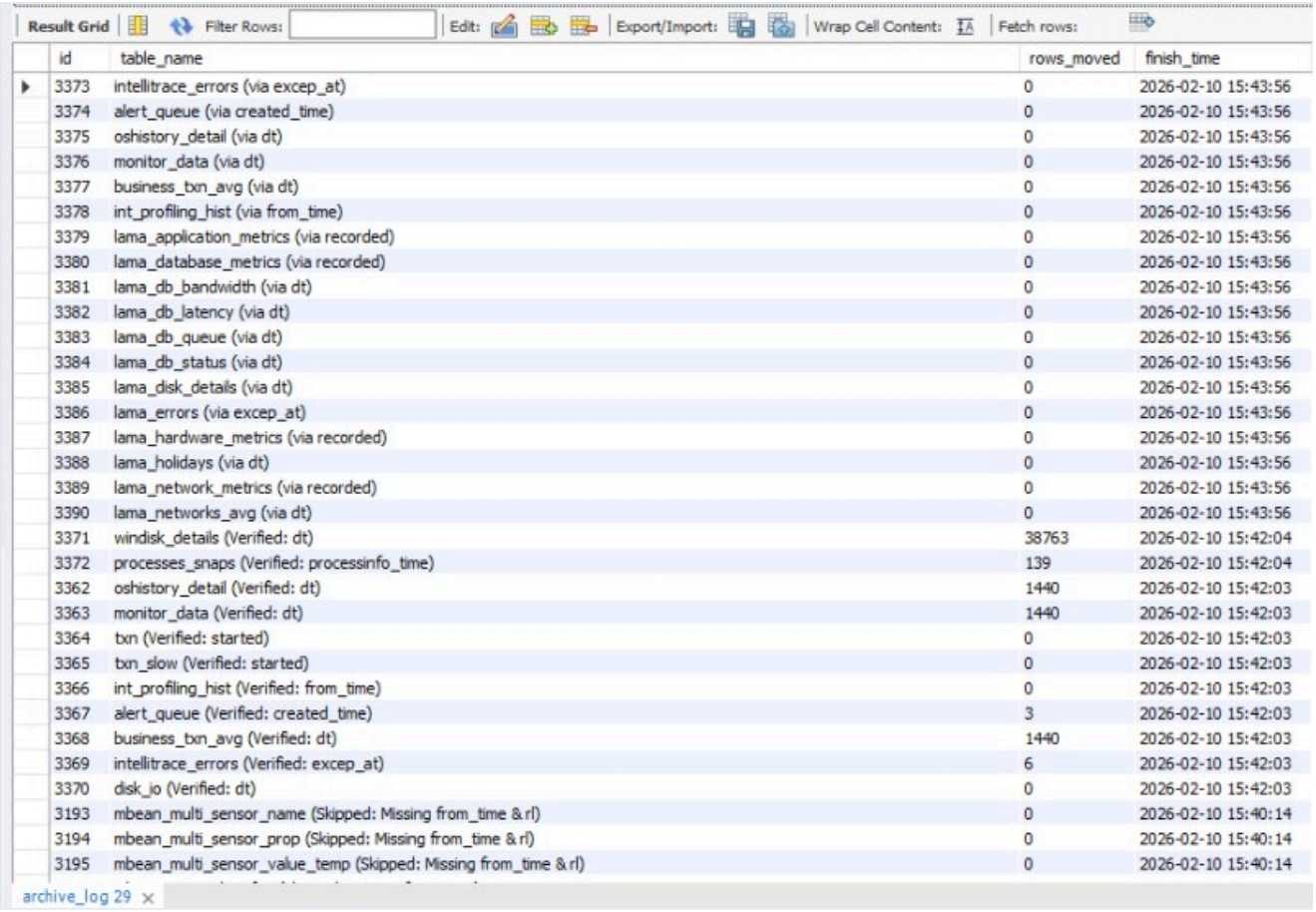
```
TEXT
CREATE TABLE IF NOT EXISTS applicarearchive.archive_logcustom (
    id INT AUTO_INCREMENT PRIMARY KEY,
    table_name VARCHAR(255),
    rows_moved INT,
    finish_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(255)
);
```

Explanation of the Columns:

- **id**: A unique number for every entry.
- **table_name**: Records the table name (and the column name used, e.g., `oshistory_detail [dt]`).
- **rows_moved**: The number of records successfully transferred for "Yesterday".
- **finish_time**: The exact date and time the backup for that specific table completed.

Step 5: Execute query to see the results. It will show you the most recent backups at the top.

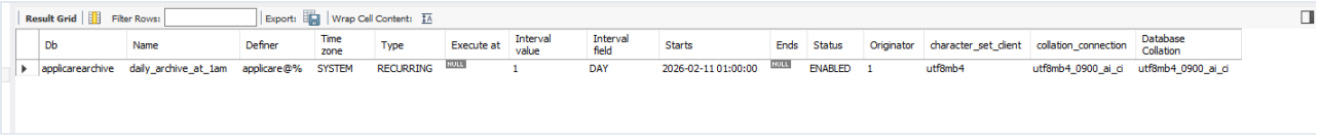
```
TEXT
SELECT * FROM applicarearchive.archive_log ORDER BY finish_time DESC;
```



id	table_name	rows_moved	finish_time
3373	intellitrace_errors (via excep_at)	0	2026-02-10 15:43:56
3374	alert_queue (via created_time)	0	2026-02-10 15:43:56
3375	oshistory_detail (via dt)	0	2026-02-10 15:43:56
3376	monitor_data (via dt)	0	2026-02-10 15:43:56
3377	business_txn_avg (via dt)	0	2026-02-10 15:43:56
3378	int_profiling_hist (via from_time)	0	2026-02-10 15:43:56
3379	lama_application_metrics (via recorded)	0	2026-02-10 15:43:56
3380	lama_database_metrics (via recorded)	0	2026-02-10 15:43:56
3381	lama_db_bandwidth (via dt)	0	2026-02-10 15:43:56
3382	lama_db_latency (via dt)	0	2026-02-10 15:43:56
3383	lama_db_queue (via dt)	0	2026-02-10 15:43:56
3384	lama_db_status (via dt)	0	2026-02-10 15:43:56
3385	lama_disk_details (via dt)	0	2026-02-10 15:43:56
3386	lama_errors (via excep_at)	0	2026-02-10 15:43:56
3387	lama_hardware_metrics (via recorded)	0	2026-02-10 15:43:56
3388	lama_holidays (via dt)	0	2026-02-10 15:43:56
3389	lama_network_metrics (via recorded)	0	2026-02-10 15:43:56
3390	lama_networks_avg (via dt)	0	2026-02-10 15:43:56
3371	windisk_details (Verified: dt)	38763	2026-02-10 15:42:04
3372	processes_snaps (Verified: processinfo_time)	139	2026-02-10 15:42:04
3362	oshistory_detail (Verified: dt)	1440	2026-02-10 15:42:03
3363	monitor_data (Verified: dt)	1440	2026-02-10 15:42:03
3364	txn (Verified: started)	0	2026-02-10 15:42:03
3365	txn_slow (Verified: started)	0	2026-02-10 15:42:03
3366	int_profiling_hist (Verified: from_time)	0	2026-02-10 15:42:03
3367	alert_queue (Verified: created_time)	3	2026-02-10 15:42:03
3368	business_txn_avg (Verified: dt)	1440	2026-02-10 15:42:03
3369	intellitrace_errors (Verified: excep_at)	6	2026-02-10 15:42:03
3370	disk_io (Verified: dt)	0	2026-02-10 15:42:03
3193	mbean_multi_sensor_name (Skipped: Missing from_time &rl)	0	2026-02-10 15:40:14
3194	mbean_multi_sensor_prop (Skipped: Missing from_time &rl)	0	2026-02-10 15:40:14
3195	mbean_multi_sensor_value_temp (Skipped: Missing from_time &rl)	0	2026-02-10 15:40:14

Step 6: Use this query to list all active events within the **applicarearchive** schema.

```text
SHOW EVENTS FROM applicarearchive;
```



Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Starts	Ends	Status	Originator	character_set_client	collation_connection	Database Collation
applicarearchive	daily_archive_at_1am	applicare@%	SYSTEM	RECURRING	NULL	1	DAY	2026-02-11 01:00:00	NULL	ENABLED	1	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

CONFIGURATION GUIDES › DATABASES

Automate Daily Batch Backups of Init profiling hist Table Records to an Archive Database

Applicare 9.2 · Updated February 10, 2026 · mysql · **Pending technical review**

Step 1: Before backing up the database, create the **applicarearchive** schema, execute the applicare-related sql's and ensure all tables are created.

Step 2: Connect to the database via **MySQL Workbench**, open a new SQL query tab, and execute the following command.

NOTE: Replace the <targetDB> placeholder with actual backup database name before running the query.

```text
DELIMITER //

CREATE PROCEDURE applicarearchive.sp_BatchArchiveYesterday()
BEGIN
   DECLARE rows_affected INT DEFAULT 1;

   SET @yesterday_start = CONCAT(CURDATE() - INTERVAL 1 DAY, ' 00:00:00');
   SET @yesterday_end   = CONCAT(CURDATE() - INTERVAL 1 DAY, ' 23:59:59');

   SET @sourcedb = '<targetDB>';
   SET @backupdb  = 'applicarearchive';
   WHILE rows_affected > 0 DO

       SET @sql_query = CONCAT(
           'INSERT INTO ', @backupdb, '.int_profiling_hist ',
           'SELECT * FROM ', @sourcedb, '.int_profiling_hist ',
           'WHERE from_time BETWEEN ? AND ? ',
           'AND rl NOT IN (SELECT rl FROM ', @backupdb, '.int_profiling_hist) ',
           'LIMIT 10000'
       );
       PREPARE stmt FROM @sql_query;
       EXECUTE stmt USING @yesterday_start, @yesterday_end;
       SET rows_affected = ROW_COUNT();
       DEALLOCATE PREPARE stmt;

   END WHILE;
END //
DELIMITER ;
```

NOTE: The query above only backs up data from the int_profiling_hist table, rather than the entire database

Safe Data Duplication: This query performs a "Safe Copy" that mirrors yesterday's data into the archive without removing or altering the original records

Batch Load Management: It processes records in chunks of 10,000 to maintain low CPU usage and prevent the database from locking up during the transfer.

Automated Time Filtering: The script automatically calculates the 24-hour range for "Yesterday," ensuring only the correct historical data is captured each time it runs.

Duplicate Prevention: It uses a unique identifier check (`rl`) to ensure that even if the script runs twice, it won't create double entries in the archive.

Dynamic Table Routing: Through the use of Prepared Statements, the script can dynamically target different databases without needing manual code changes.

Step 3: Run the query below to enable the MySQL Event Scheduler and schedule the **sp_BatchArchiveYesterday** procedure to run daily at 1:00 AM.

```text
SET GLOBAL event_scheduler = ON;
DROP EVENT IF EXISTS applicarearchive.daily_archive_at_1am;
CREATE EVENT applicarearchive.daily_archive_at_1am
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 1 HOUR)
DO
 CALL applicarearchive.sp_BatchArchiveYesterday();
```

To modify the scheduled time, first execute the query below to drop the existing event, then re-create it with your desired schedule.

```text
DROP EVENT IF EXISTS applicarearchive.daily_archive_at_1am;
```

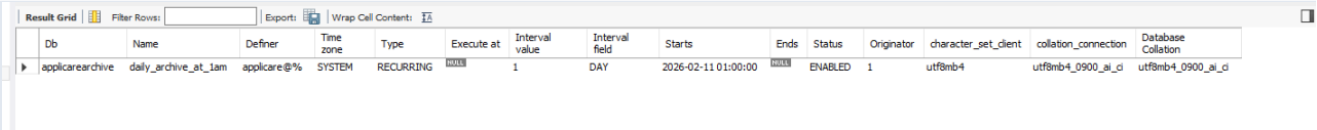
To change the time to 8:00 PM, We need to adjust the `INTERVAL` in the `STARTS` clause. Since MySQL uses a 24-hour format, 8:00 PM is 20 hours.

e.g.

```text
SET GLOBAL event_scheduler = ON;
CREATE EVENT applicarearchive.daily_archive_at_8pm
ON SCHEDULE EVERY 1 DAY
STARTS (CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 20 HOUR)
DO
 CALL applicarearchive.sp_BatchArchiveYesterday();
```

Step 4: Use this query to list all active events within the **applicarearchive** schema.

```
TEXT
SHOW EVENTS FROM applicarearchive;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Starts	Ends	Status	Originator	character_set_client	collation_connection	Database Collation
applicarearchive	daily_archive_at_1am	applicare@%	SYSTEM	RECURRING	NULL	1	DAY	2026-02-11 01:00:00	NULL	ENABLED	1	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

CONFIGURATION GUIDES › DATABASES

Applicare MySQL Database Backup (Windows)

Applicare 9.2 · Updated February 20, 2026 · mysql · **Pending technical review**

Step 1: Open command prompt in **administrator mode** on Windows machine. Navigate to the path bin directory where MySQL was installed.

e.g. cd C:\Program Files\MySQL\MySQL Server 8.0\bin

Step 2: Once in that directory, Please select either option a or b below..

a.) To export a complete applicare database, including both its structure and its data.

```
TEXT
mysqldump.exe –u <Username> –p <database_name> > <Backup_directory>\applicare.sql
```

e.g.

```
TEXT
mysqldump.exe –u root –p applicare > C:\Applicare_DB_Backup\applicare.sql
```

```
TEXT
                                                                **(OR)**
```

b.) To export only the data (the INSERT statements) without the CREATE TABLE structures, We need to use the **--no-create-info** argument with mysql backup command.

```
TEXT
mysqldump.exe  –u <username> –p ––no–create–info <destination_db_name>
<Backup_directory>\applicare.sql
```

e.g.

```
TEXT
mysqldump.exe  –u root –p ––no–create–info applicare > C:\Applicare_DB_Backup\applicare.sql
```

Step 3: After running the command, MySQL will prompt you for a **password**, Enter the password and wait few minutes for the process to complete.

After executing the command, verify that there is SQL file named applicare.sql in backup directory.

To import .sql backup file into another MySQL database, Use the standard mysql command.

Step 4: Before running the command, ensure the destination database already exists. If it doesn't, Create the database.

Step 5: Open command prompt in **administrator mode** on Windows machine. Navigate to the path bin directory where MySQL was installed.

e.g. cd C:\Program Files\MySQL\MySQL Server 8.0\bin

Step 6: Once in that directory, execute the following command.

```
TEXT
"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" -u [username] -p
[destination_database_name] < "C:\backups\applicare.sql"
```

e.g.

```
TEXT
"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" -u root -p applicarebackup <
"C:\applicaredb_backup\test_backup_processes\applicaredb.sql"
```

After running the command, MySQL will prompt you for a **password**, Enter the password and wait few minutes for the process to complete.

CONFIGURATION GUIDES › DATABASES

Set up the Applicare schema in MySQL

Applicare 9.2 · Updated September 13, 2024 · mysql · **Pending technical review**

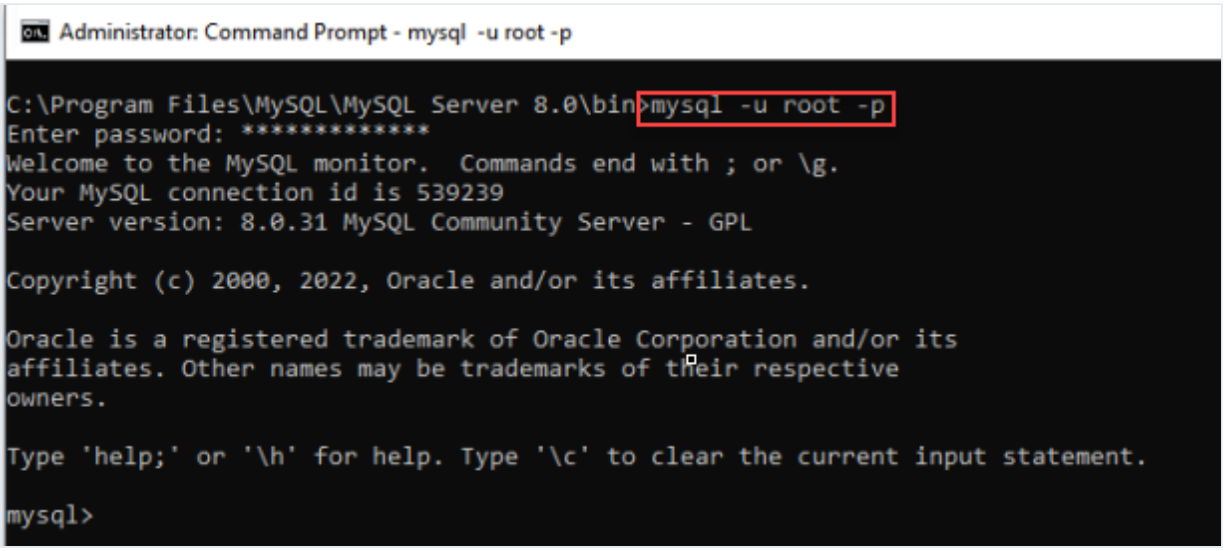
1. Open the command prompt as an administrator then navigate to **MySQL_install_directory/bin** path.

e.g. C:\Program Files\MySQL\MySQL Server 8.0\bin

2. After executing the command below, you will be prompted for a password, Once entered, you will be connected to the MySQL database.

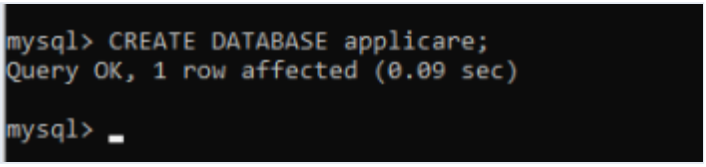
```
TEXT
mysql –u UserName –p
```

UserName - The username you specified during MySQL installation



3. Once connected to the MySQL database, execute the command below to create the schema.

```
TEXT
CREATE DATABASE applicare;
```



4. Executing the command below will display the created database.

```
TEXT
SHOW DATABASES;
```

Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

MySQL 8 installer for windows

Applicare 9.2 · Updated September 13, 2024 · mysql · **Pending technical review**

Step 1: Download the MySQL 8 on below link and install it.

MySQL8

Step 2: After installing the MySQL, follow the steps in the below link to change the SQL mode configuration.

SQL_mode

Step 3: Follow the steps in link below to create Applicare schema.

create Applicare_schema

Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

Installing MySQL on Linux

Applicare 9.2 · Updated August 22, 2024 · mysql · **Pending technical review**

OS: RHEL 9

Step 1: Download and add the following **MySQL Yum repository** to install the latest version of MySQL.

```bash
wget https://repo.mysql.com/mysql80-community-release-el8-1.noarch.rpm
```

Step 2: After downloading the package for Linux platform, now install it using the following command.

```bash
yum localinstall mysql80-community-release-el8-1.noarch.rpm
```

Step 3: You can verify that the MySQL Yum repository has been successfully added by using the following command.

```bash
yum repolist enabled | grep "mysql.*-community.*"
```

Step 4: Install the latest version of **MySQL** (currently **8.0**) using the following command.

```bash
yum install mysql-community-server
```

Step 5: After successfully installing of MySQL, start and enable the MySQL server with the following commands.

```text
systemctl enable mysqld.service
```

```text
systemctl start mysqld.service
systemctl status mysqld.service
```

Step 6: The **MySQL_secure_installation** command helps secure your to MySQL installation by configuring essential settings, such as setting the root password, removing anonymous users and disabling remote root login, and among other tasks.

Use the following command to view the current password before running MySQL secure command.

```
TEXT
grep 'temporary password' /var/log/mysqld.log
```

Once you have the password, you can proceed to run the following command to secure your MySQL installation.

```
TEXT
mysql_secure_installation
```

Enter new Root password, which is your temporary password from a file `/var/log/mysqld.log`.

Step 7: Connect to the newly installed MySQL server by providing a username and password.

```
TEXT
mysql –uroot –p
```

Step 8: Once the connection is working, execute the **exit** command to exit the MySQL query window.

Step 9: To set SQL mode.- **Click here**

Uncomment the following line in my.cnf file to stop bin log file generation in MySQL.

```
TEXT
disable_log_bin
```

Related articles

Applicare database backup from the Windows to Linux machine.

CONFIGURATION GUIDES › DATABASES

Configuring SQL Mode for MySQL in Applicare

Applicare 9.2 · Updated October 9, 2024 · mysql · **Pending technical review**

Windows

1. Open **Services** application and **stop the MySQL service**.

2. Navigate to the directory **C:\ProgramData\MySQL\MySQL Server <version>**.

3. Create a backup of the current **my.ini** file.

4. Launch notepad as an administrator, then go to **File -> Open** and select the **C:\ProgramData\MySQL\MySQL Server <version>\my.ini** file.

5. Comment out the **existing sql-mode line** and add the new **sql-mode line** as specified.

```text
sql-mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"
```

File Edit View

```
# The current server SQL mode, which can be set dynamically.
# Modes affect the SQL syntax MySQL supports and the data validation checks it performs. This
# makes it easier to use MySQL in different environments and to use MySQL together with other
# database servers.
# sql-
mode="ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_ENGINE_SUBSTITUTION"
sql-mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"

# General and Slow logging.
log-output=FILE

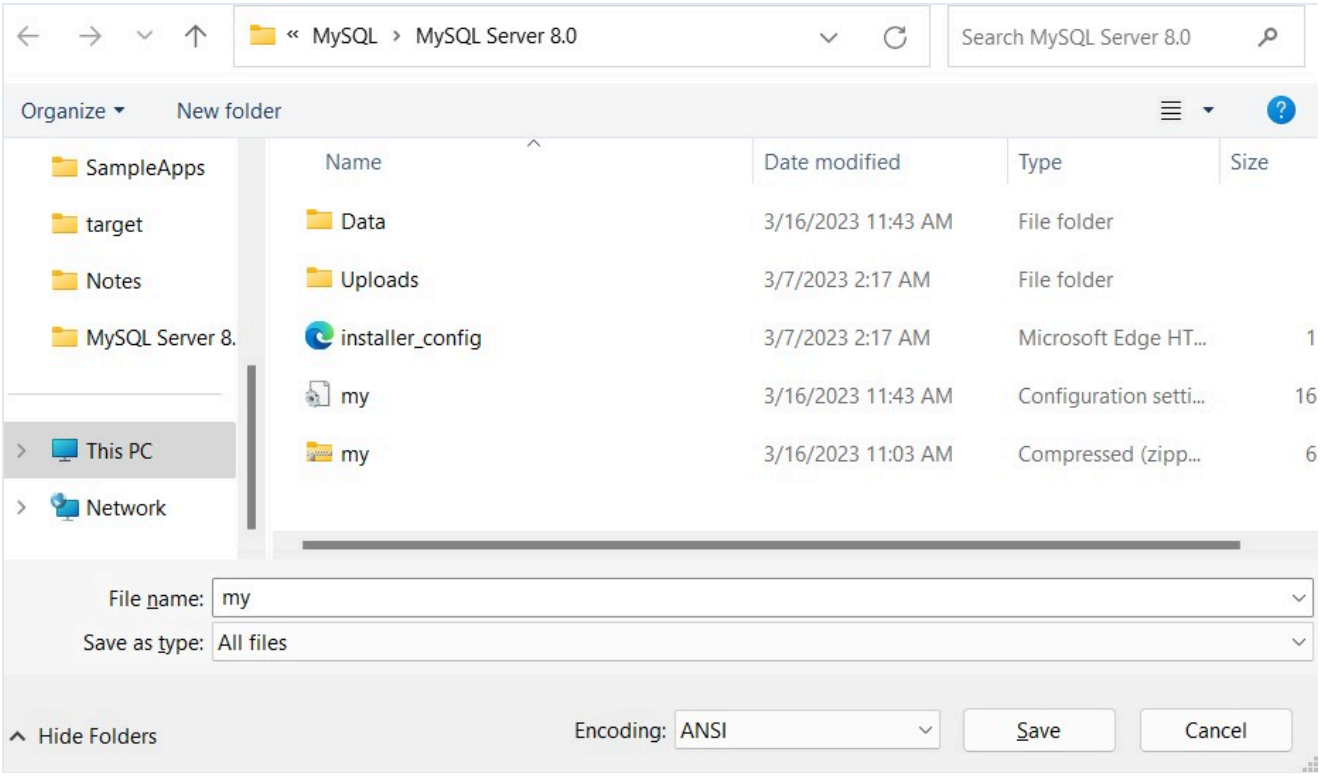
general-log=0

general_log_file="DESKTOP-L0HKKIU.log"

slow-query-log=1
```

6. Go to **File -> Save as**

- Change **Save as type** to **All files.**
- Change **Encoding** to **ANSI**.
- Click **Save**.



7. Open the **Services** application and **start the MySQL service**.

Linux

1. Stop the MySQL service.

2. Backup of the existing **my.cnf or my.ini** file.

3. Locate the MySQL configuration file: The file is typically named **my.cnf** or **my.ini** and can be found in the MySQL installation directory or the etc directory. Use the following command to search for the file.

```bash
sudo find / -name my.cnf
```

4. Open the MySQL configuration file: Once located, open it with a text editor.

5. Comment out the existing sql-mode line and add the following sql-mode line.

```text
sql-mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"
```

```
# The current server SQL mode, which can be set dynamically.
# Modes affect the SQL syntax MySQL supports and the data validation checks it performs. This
# makes it easier to use MySQL in different environments and to use MySQL together with other
# database servers.
# sql-
mode="ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_ENGINE_SUBSTITUTION"
sql-mode="STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"

# General and Slow logging.
log-output=FILE

general-log=0

general_log_file="DESKTOP-L0HKKIU.log"

slow-query-log=1
```

6. Save the changes to the configuration and **start** the MySQL services.

In production environments and when the expected data is huge, it is recommended to increase memory for mysql.

key_buffer_size innodb_buffer_pool_size

When a separate machine is dedicated to the MySql then we can set these values in the range as below.

RAM size	`key_buffer_size`	`innodb_buffer_pool_size`
16GB	2GB	12GB
32GB	4GB	24GB
64GB	8GB	48GB

7. The user which we create should have all the privileges.

Syntax: GRANT ALL PRIVILEGES ON *.* TO '**username**'@'%' IDENTIFIED BY '**password**'; FLUSH PRIVILEGES;

Example: GRANT ALL PRIVILEGES ON *.* TO '**root**'@'%' IDENTIFIED BY '**testuserpass**'; FLUSH PRIVILEGES;

CONFIGURATION GUIDES › DATABASES

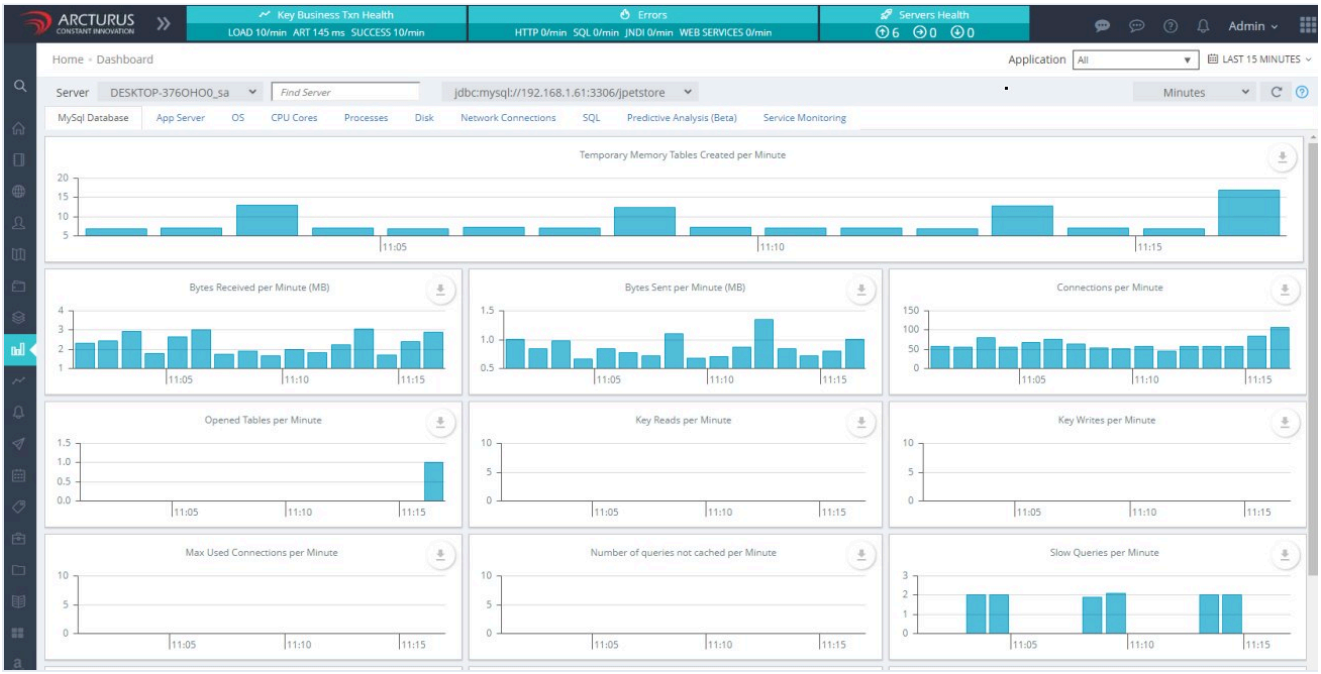
MySQL Database Monitoring

Applicare 9.2 · Updated August 22, 2024 · mysql · **Pending technical review**

Login to Applicare controller and navigate to the **Server Analyzer** menu to view the details.

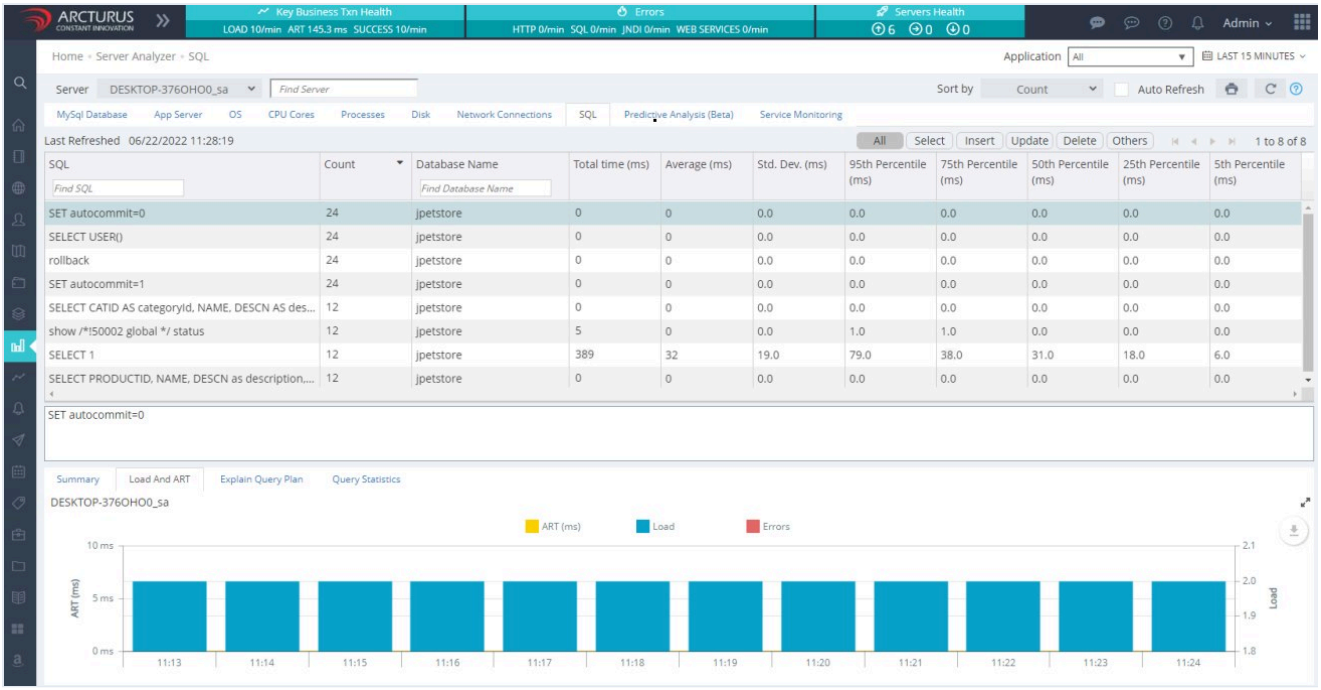
Dashboard

Database performance metrics assist in monitoring and optimizing your database for business needs. These metrics are calculated for the entire server and are not limited to the specific database indicated in the JDBC URL.



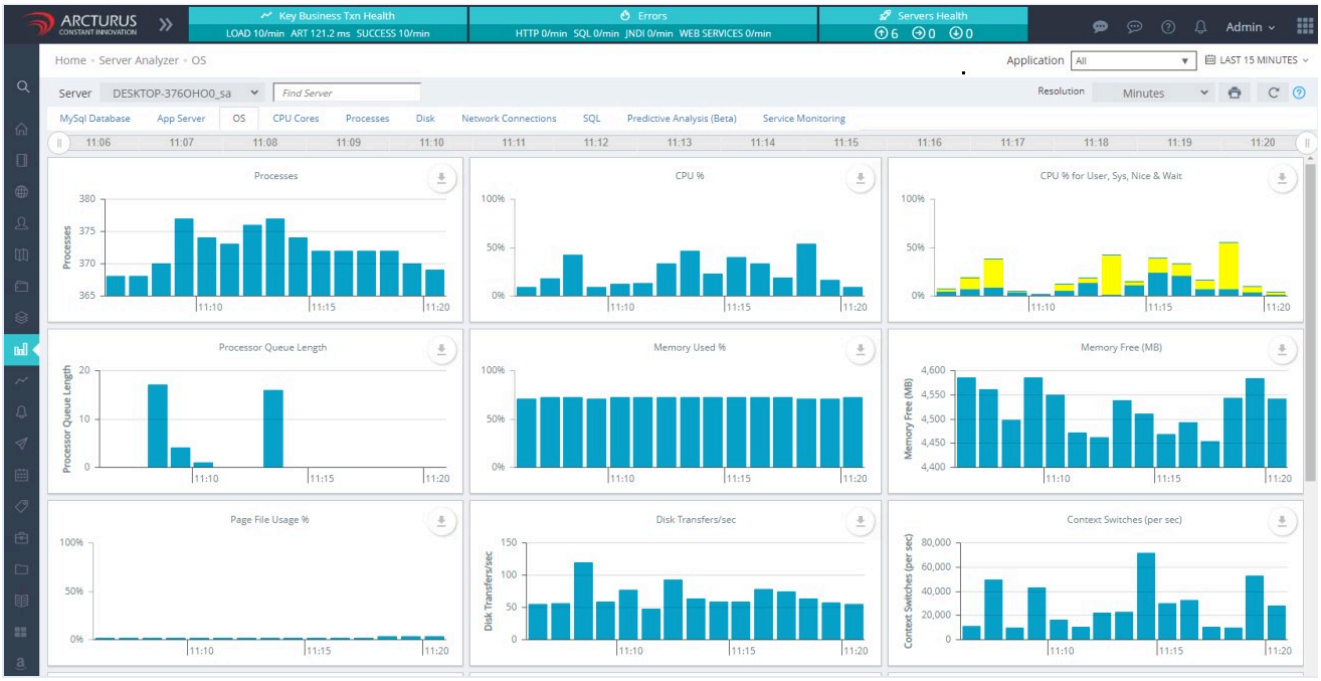
SQL

SQL Statement Profiling shows the performance details of your database queries.It helps you identify which statement are slow, making it easier to diagnose and fix performance issues or SQL/s. You'll get detailed stats, including the database name, query count, average time and various percentile.



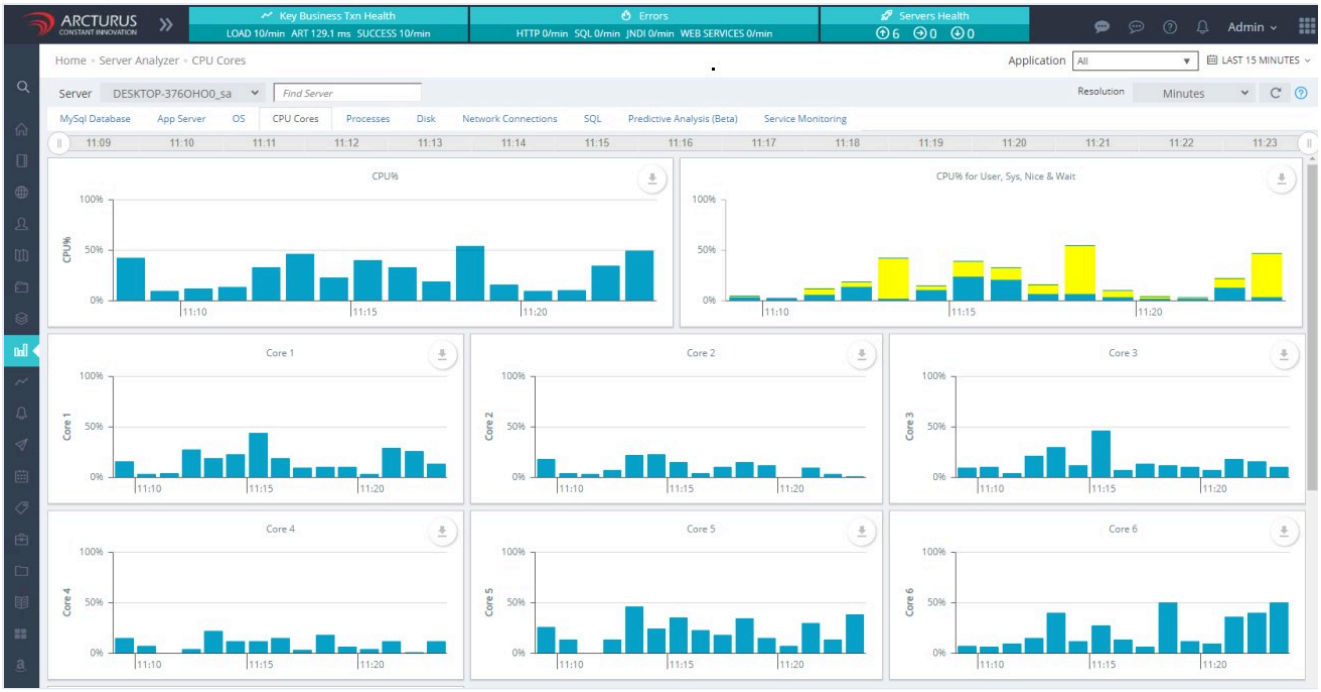
OS

This view offers a quick and easy overview of your physical server's/VM's performance including metrics such as the number of processes, context switches, CPU Usage, memory, storage performance and etc.



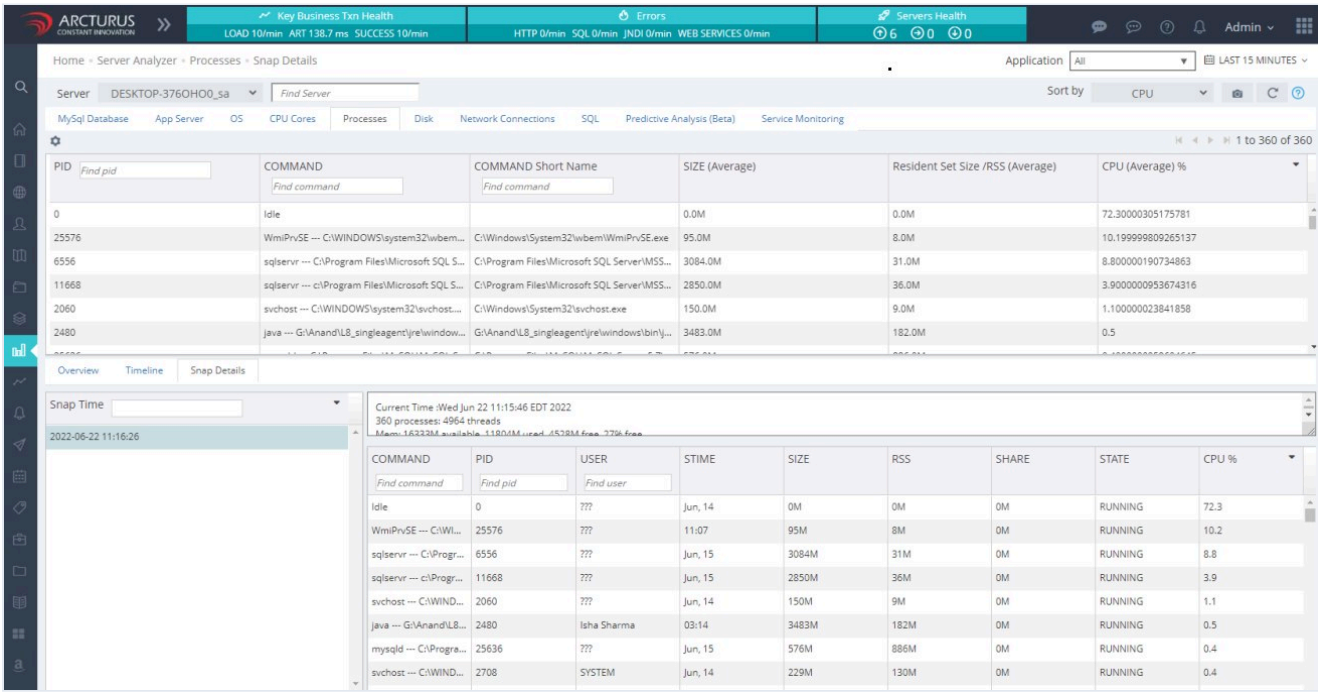
CPU CORE Analyzer

This view offers an overview of the overall CPU performance along with detailed information for each individual CPU core.



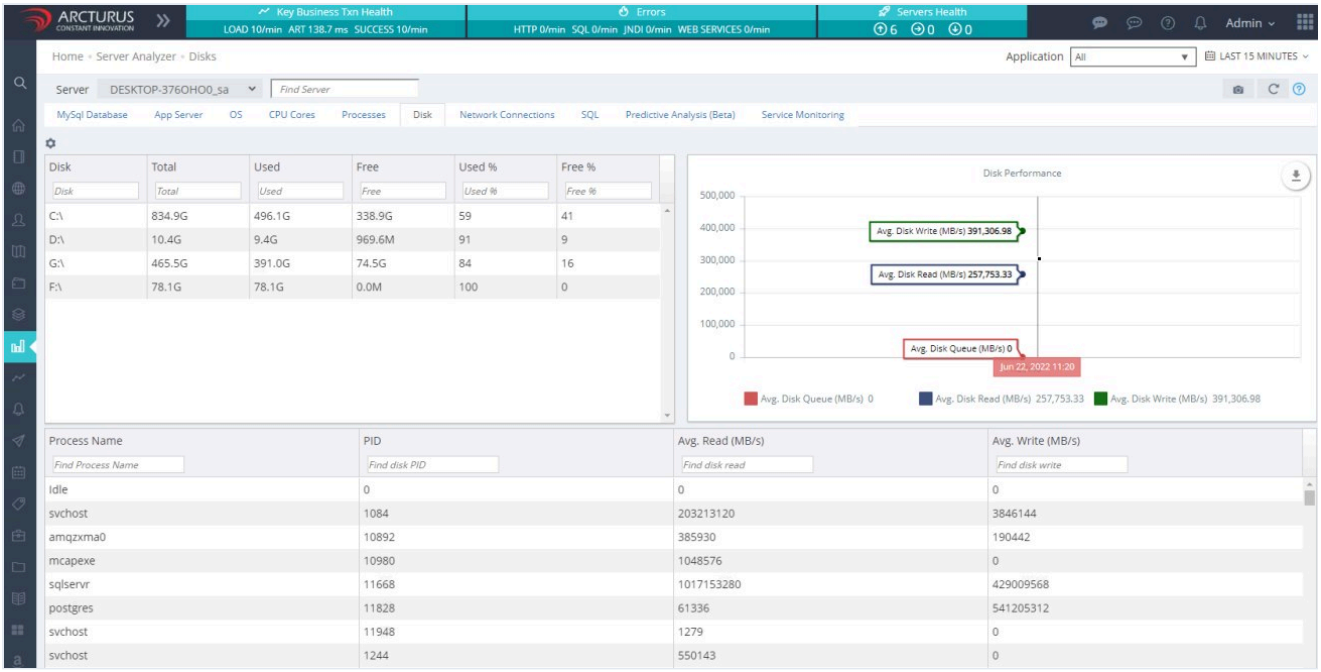
Processes

This view presents organized performance stats for CPU, memory and running processes. It includes historical snapshots of processes at specific times, showing the resources they were consuming. By default, Applicare takes these snapshot in every 10 minutes, but you can adjust the interval as needed.



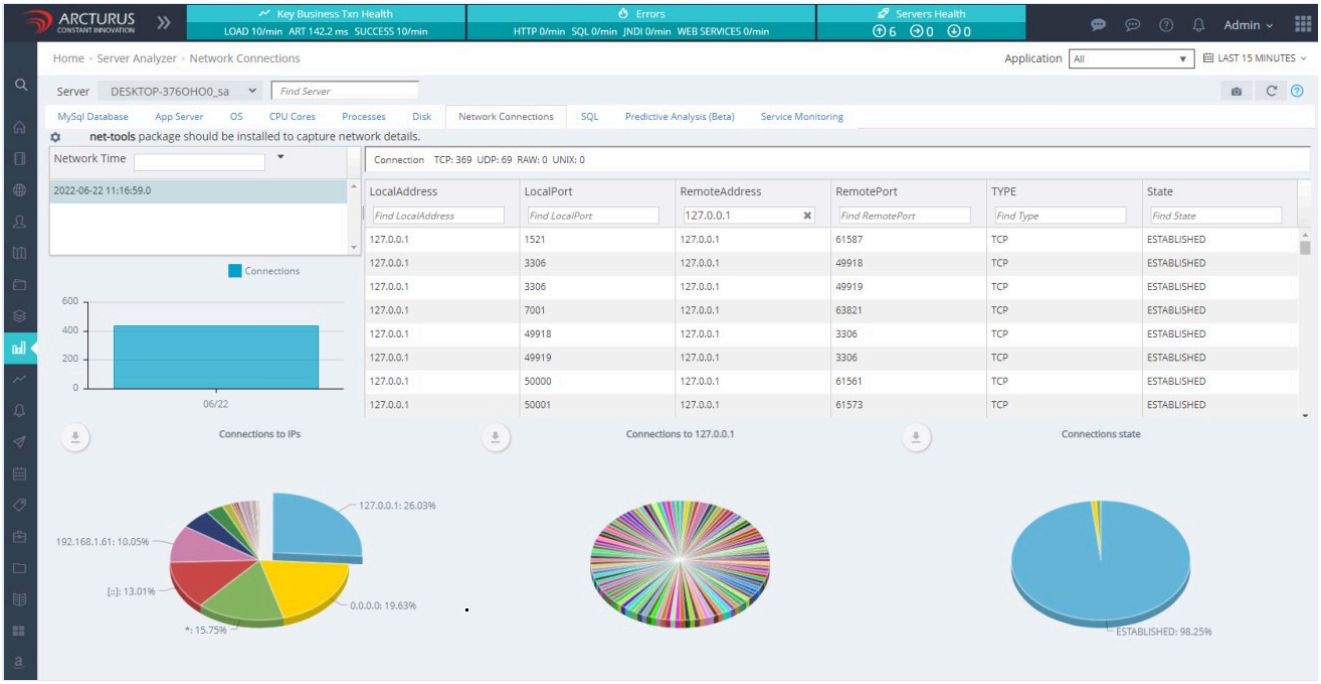
Disk

This view shows the disk usage for each process, detailing how much disk space each one is accessing.



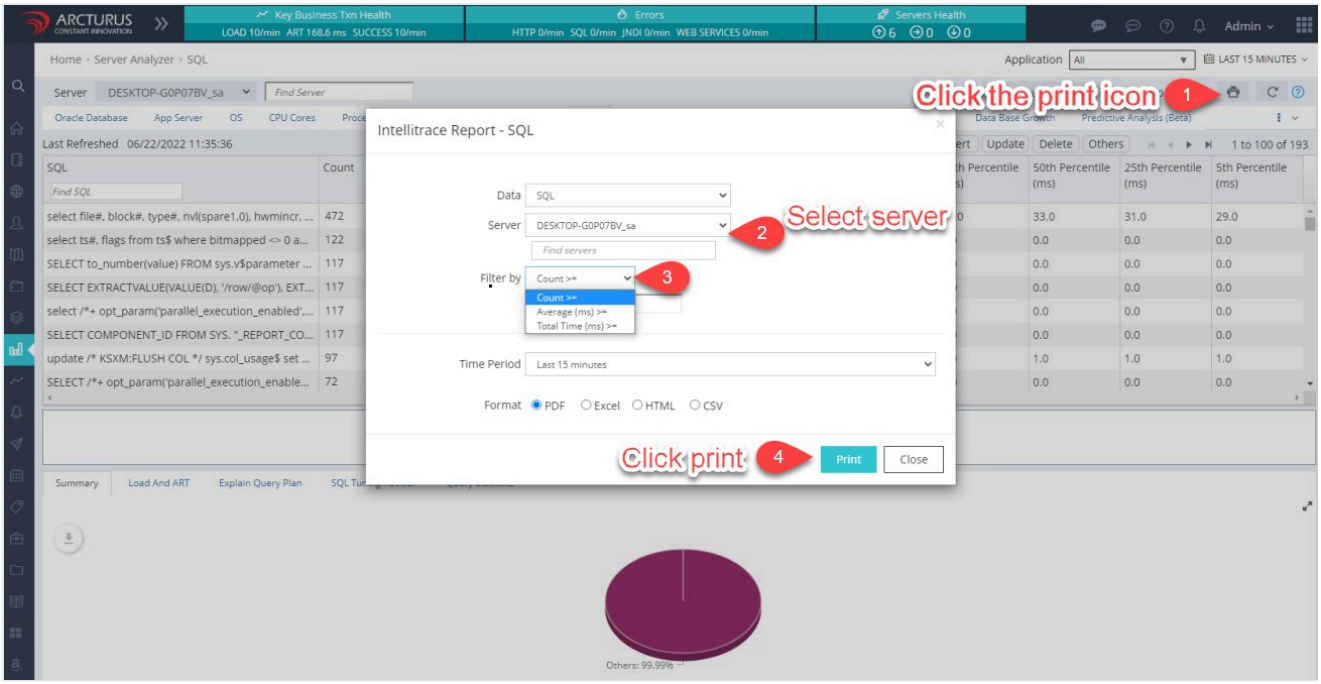
Network Connections

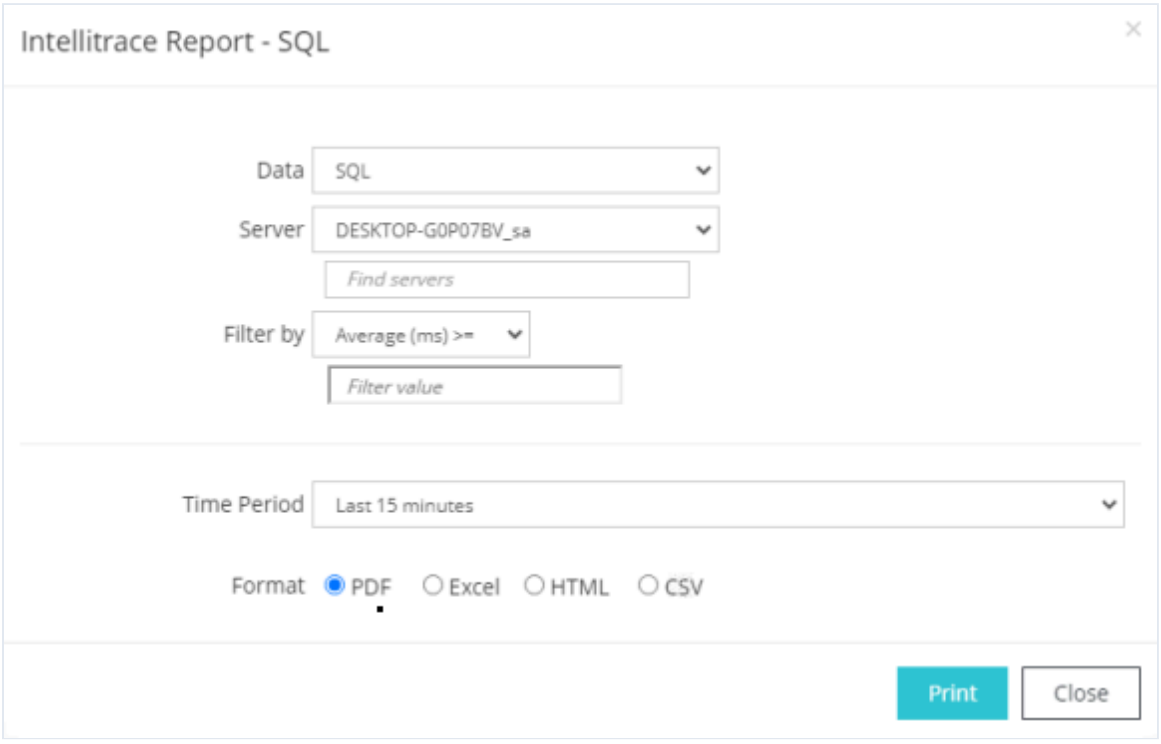
This view provides details on TCP, UDP, RAW, and UNIX connections.

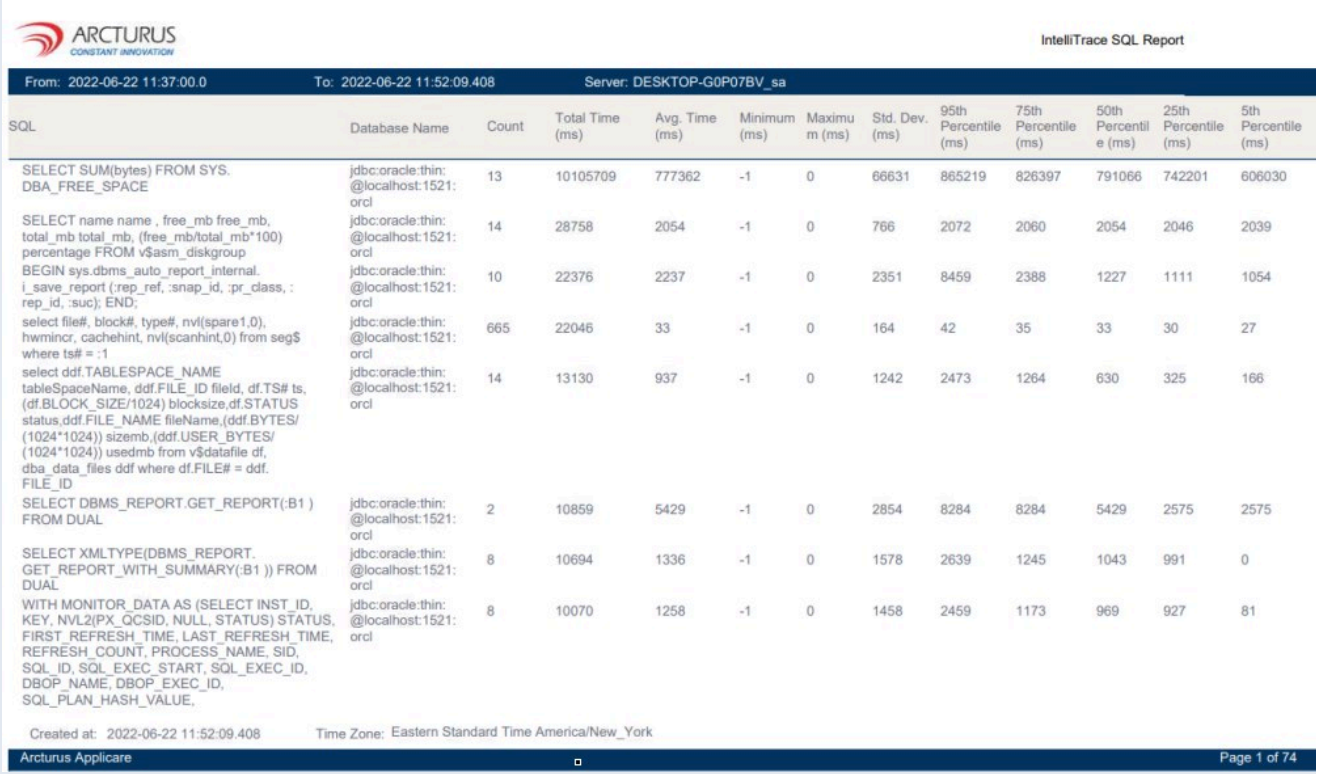


SQL Report

To download a report of SQL statements, click the print icon on the SQL screen within the server Analyzer menu.


Click the print icon
1
Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
2
Select server
Find servers
Filter by
Count >=
3
Count >=
Average (ms) >=
Total Time (ms) >=
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Click print
4
Print
Close


Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
Find servers
Filter by
Average (ms) >=
Filter value
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Print
Close



ARCTURUS
CONSTANT INNOVATION

IntelliTrace SQL Report

From: 2022-06-22 11:37:00.0 To: 2022-06-22 11:52:09.408 Server: DESKTOP-G0P07BV_sa

SQL	Database Name	Count	Total Time (ms)	Avg. Time (ms)	Minimum (ms)	Maximum (ms)	Std. Dev. (ms)	95th Percentile (ms)	75th Percentile (ms)	50th Percentile (ms)	25th Percentile (ms)	5th Percentile (ms)
SELECT SUM(bytes) FROM SYS.DBA_FREE_SPACE	jdbc:oracle:thin:@localhost:1521:orcl	13	10105709	777362	-1	0	66631	865219	826397	791066	742201	606030
SELECT name name , free_mb free_mb, total_mb total_mb, (free_mb/total_mb*100) percentage FROM v$asm_diskgroup	jdbc:oracle:thin:@localhost:1521:orcl	14	28758	2054	-1	0	766	2072	2060	2054	2046	2039
BEGIN sys.dbms_auto_report_internal.i_save_report (:rep_ref, :snap_id, :pr_class, :rep_id, :suc); END;	jdbc:oracle:thin:@localhost:1521:orcl	10	22376	2237	-1	0	2351	8459	2388	1227	1111	1054
select file#, block#, type#, nvl(spare1,0), hwmincr, cachehint, nvl(scanhint,0) from seg$ where ts# = :1	jdbc:oracle:thin:@localhost:1521:orcl	665	22046	33	-1	0	164	42	35	33	30	27
select ddf.TABLESPACE_NAME tableSpaceName, ddf.FILE_ID fileId, df.TS# ts, (df.BLOCK_SIZE/1024) blocksize,df.STATUS status,ddf.FILE_NAME fileName,(ddf.BYTES/(1024*1024)) sizemb,(ddf.USER_BYTES/(1024*1024)) usedmb from v$datafile df, dba_data_files ddf where df.FILE# = ddf.FILE_ID	jdbc:oracle:thin:@localhost:1521:orcl	14	13130	937	-1	0	1242	2473	1264	630	325	166
SELECT DBMS_REPORT.GET_REPORT(:B1) FROM DUAL	jdbc:oracle:thin:@localhost:1521:orcl	2	10859	5429	-1	0	2854	8284	8284	5429	2575	2575
SELECT XMLTYPE(DBMS_REPORT.GET_REPORT_WITH_SUMMARY(:B1)) FROM DUAL	jdbc:oracle:thin:@localhost:1521:orcl	8	10694	1336	-1	0	1578	2639	1245	1043	991	0
WITH MONITOR_DATA AS (SELECT INST_ID, KEY, NVL2(PX_QCSID, NULL, STATUS) STATUS, FIRST_REFRESH_TIME, LAST_REFRESH_TIME, REFRESH_COUNT, PROCESS_NAME, SID, SQL_ID, SQL_EXEC_START, SQL_EXEC_ID, DBOP_NAME, DBOP_EXEC_ID, SQL_PLAN_HASH_VALUE,	jdbc:oracle:thin:@localhost:1521:orcl	8	10070	1258	-1	0	1458	2459	1173	969	927	81

Created at: 2022-06-22 11:52:09.408 Time Zone: Eastern Standard Time America/New_York

Arcturus Applicare Page 1 of 74

Configuration

MySQL database monitoring configuration

CONFIGURATION GUIDES › DATABASES

Increasing Memory for Derby

Applicare 9.2 · Updated September 13, 2024 · mysql · **Pending technical review**

By default, the Applicare Derby database is configured to run on a laptop and is suitable for testing with Applicare with One or two agents. If the Derby database is used for POCs or testing scenarios that involve a higher no. of agents and transactions, you should increase the amount of memory allocated to Derby.

Step 1: Stop the Derby database.

Step 2: Navigate to the directory listed below.

```xml
<Applicare_home_directory>\javadb\bin
```

For Windows

Edit the **startNetworkServer.bat** file. Locate the two **-Xmx**, parameter within the file and update both values as needed.

```
if "%_JAVACMD%"=="" goto end

if "%_USE_CLASSPATH%"=="no" goto runNoClasspath
if not "%CLASSPATH%"=="" goto runWithClasspath

:runNoClasspath
"%_JAVACMD%" -Xmx2048m %DERBY_OPTS% -classpath "%LOCALCLASSPATH%" org.apache.derby.drda.NetworkServerControl start -h 0.0.0.0
goto end

:runWithClasspath
"%_JAVACMD%" -Xmx1000m %DERBY_OPTS% -classpath "%CLASSPATH%;%LOCALCLASSPATH%" org.apache.derby.drda.NetworkServerControl start
goto end
```

For Linux

Edit the **startNetworkServer.sh** file. Find the **-Xmx** parameter within the file and adjust its value accordingly.

```
. $DERBY_HOME/bin/derby_common.sh

derby_exec_command="exec \"$JAVACMD\" -Xmx2048m $DERBY_OPTS -classpath
eval $derby_exec_command
```

Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

Create a new user account and designate the relevant permissions

Applicare 9.2 · Updated April 16, 2026 · mysql · **Pending technical review**

Syntax:

```
TEXT
CREATE USER '<user>'@'%' IDENTIFIED BY '<password>';
GRANT SELECT, SHOW DATABASES, SHOW VIEW ON *.* TO '<user>'@'%';
GRANT EXECUTE ON xp_readerrorlog TO <user>;
```

Substitute **<new_user>** and **<password>** with the newly username and password you have just created.

Example:If the newly created username is **appuser** and the password is **appuserpass**, use these details in place of **<new_user>** and **<passwords>**."

```
TEXT
CREATE USER 'appuser'@'%' IDENTIFIED BY 'appuserpass';
GRANT SELECT, SHOW DATABASES, SHOW VIEW, REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
'appuser'@'%';
GRANT EXECUTE ON xp_readerrorlog TO 'appuser';
```

CONFIGURATION GUIDES › DATABASES

MSSQL manual Trace deletion

Applicare 9.2 · Updated November 5, 2024 · sql-server · **Pending technical review**

Step 1: Login into **MSSQL Server** with '**sa**' or **Applicare** user. The user should have **sysadmin** permission and create and modify permission for SQL traces in "**master**" database.

Step 2: Select "**master**" database and select "**New Query**" and execute the below command.

```
TEXT
Select * from sys.traces;
```

Step 3: Delete Trace

a. Stop the Agent from the task manager. b. Stop the trace if it is already running.

```
TEXT
sp_trace_setstatus @traceid , @status;
```

If it is a Applicare trace then in the path column it will have the Applicare Agent Home path and at last it will end with **SQLTraceProfiling.trc** and **SQLDeadlockTraceProfiling.trc**

Trace details

id - @traceID

status - @status

Status Conditions

0 - Stops the specified trace. 1 - Starts the specified trace. 2 - Closes the specified trace and deletes its definition from the server.

Path - Trace file created path

Modifies the current state of the specified trace

Replace the **@traceid** with correct **id** and **@status** with **status condition** and execute the above query

Let us consider **@traceid - 2**

Stop and delete Trace entry from server

```
TEXT
sp_trace_setstatus 2 , 0;
```

```
TEXT
sp_trace_setstatus 2 , 2
```

Step 4: Delete the trace files from the Agent Home directory.

CONFIGURATION GUIDES › DATABASES

Monitoring for MSSQL Server Always On Availability Groups

Applicare 9.2 · Updated September 13, 2024 · sql-server · **Pending technical review**

Applicare SingleAgent is an effective tool for managing and monitoring your availability groups. It offers a comprehensive view of performance across your MS SQL Server, helping you identify and troubleshoot issues at the Windows Server Failover Cluster (WSFC) level, SQL Server instance level, and Replica level.

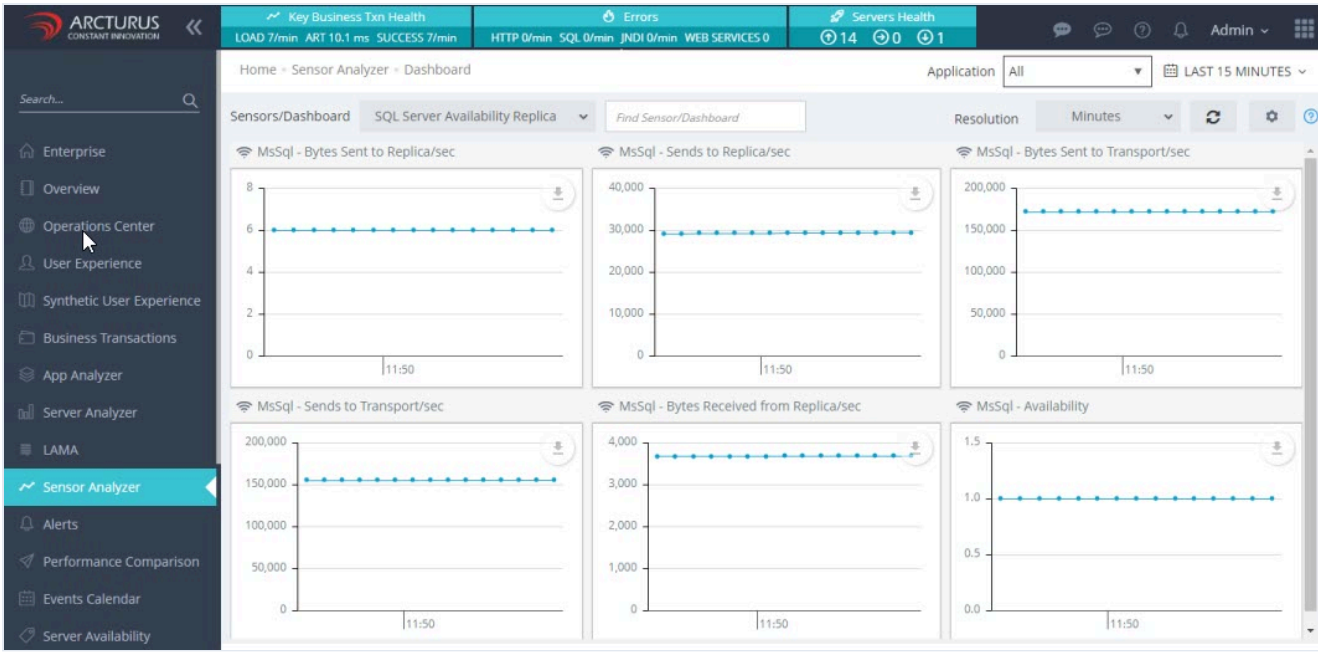
Sensor Analyzer

Sensor Analyzer allows users to create and customize their own dashboards, enabling them to monitor the metrics that matter most to them..

- **Visualize data trends:** Charts enable you to swiftly recognize trends in your data, aiding in the detection of potential issues or opportunities.
- **Identify Anomalies:** Charts can highlight anomalies such as sudden spikes or drops, which is useful for troubleshooting.
- **Correlate Data:** Charts facilitate the correlation of data from various sources. Helping you understand the root cause of issues.

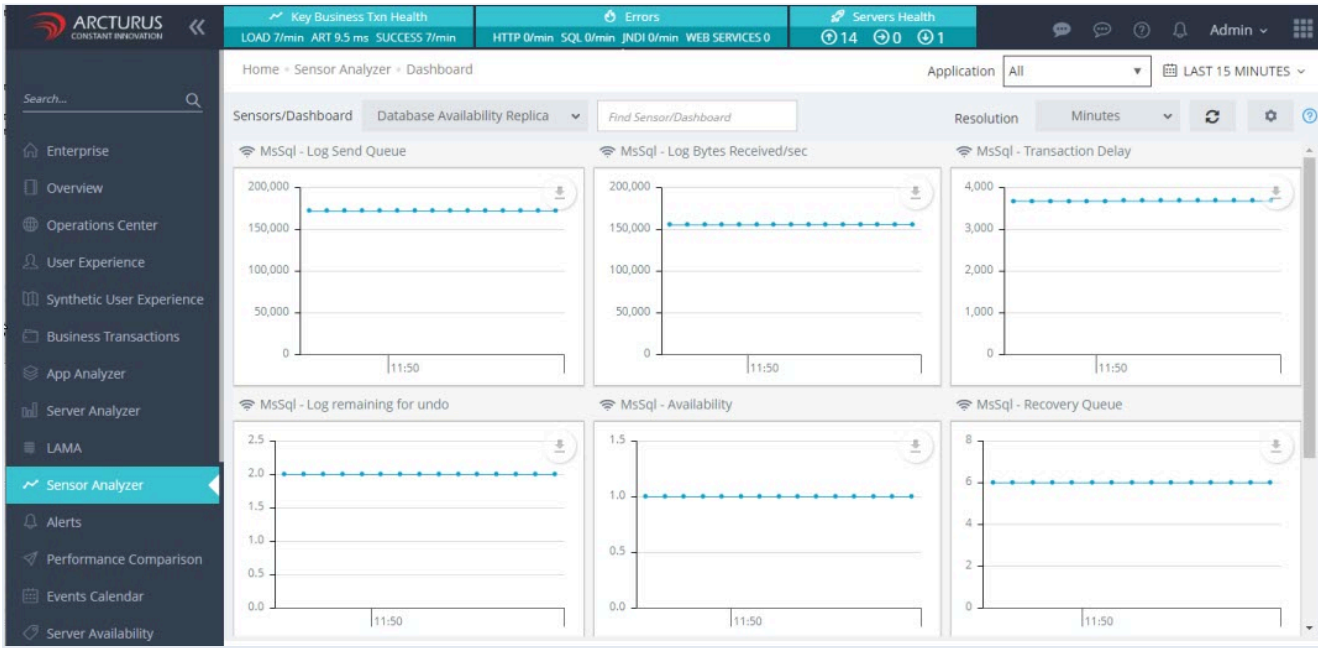
Server Availability Replication

This feature monitors the availability replicas in Always On availability groups in SQL Server. It includes sensors for both the primary and the secondary replicas, with send/receive counters reflecting the status of each local replica.



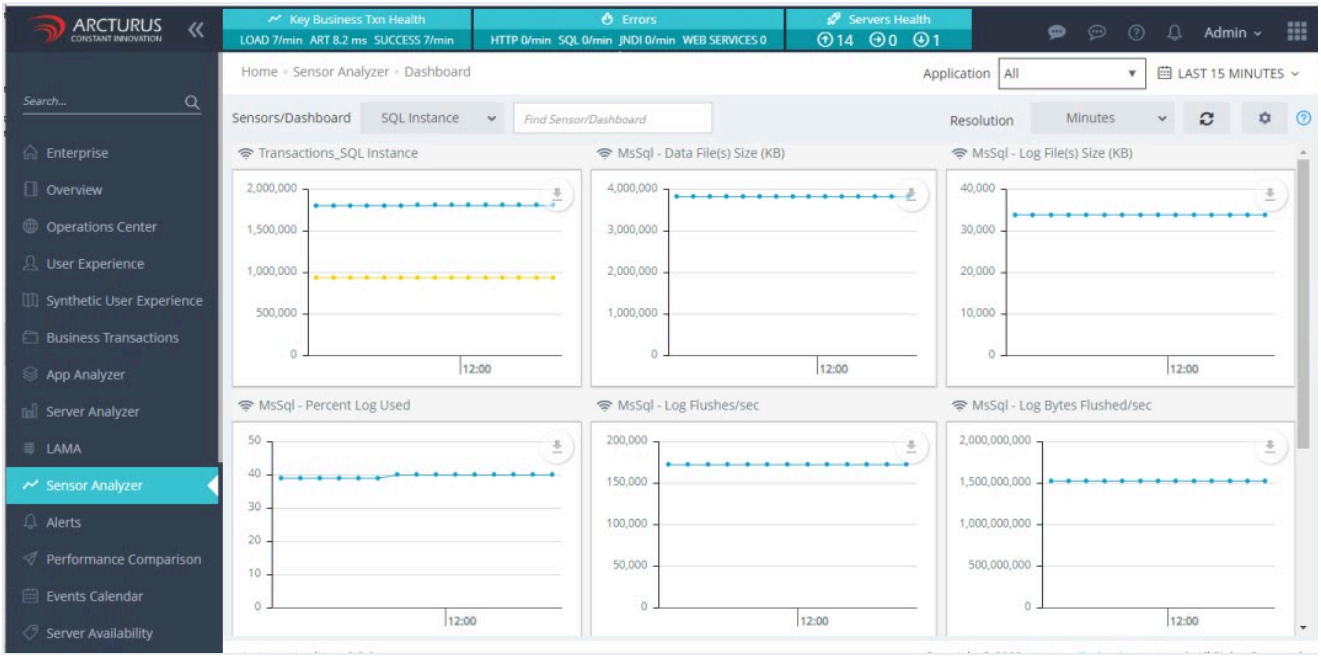
Database Availability Replication

This feature provides details about the secondary databases of an Always On availability group in SQL Server.



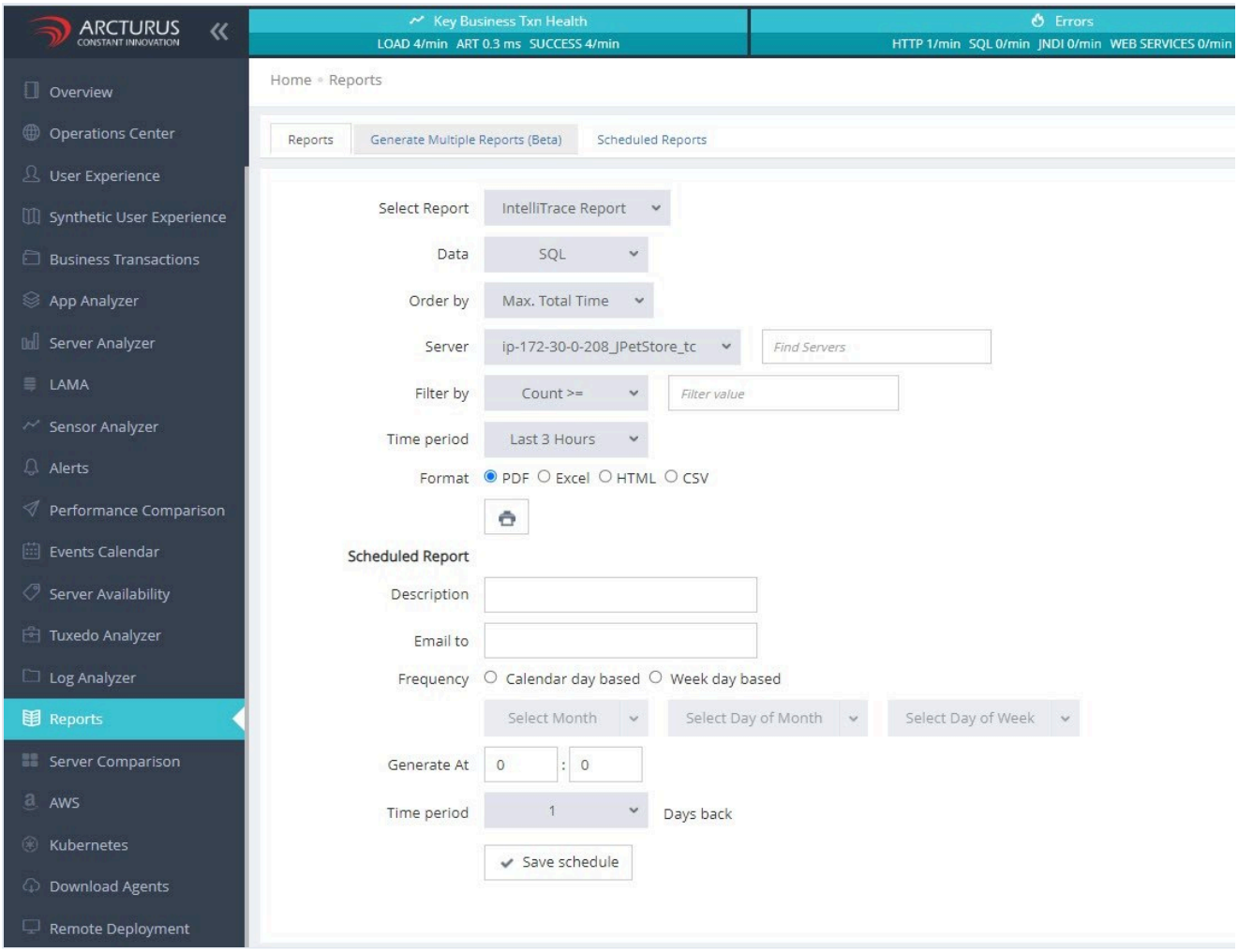
SQL Server Instance

Monitor transactions and the transaction log to assess user activity within the database and the growth of the transaction log. The level of user activity impacts database performance and influence log size, locking, and replication.



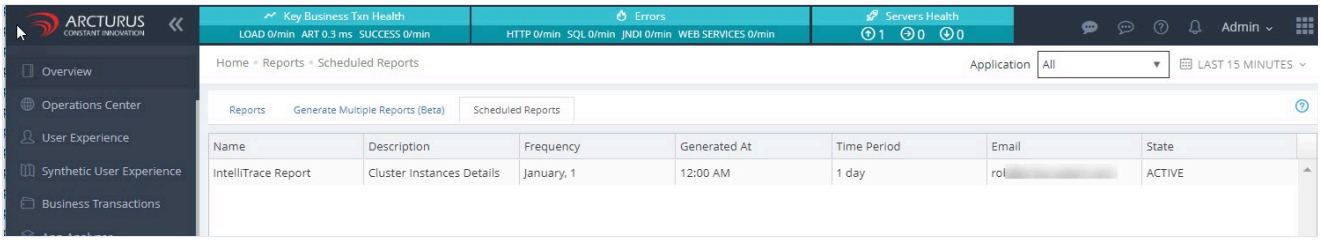
Reports

Reports offer on-demand access to historical analysis data, enabling you to understand and optimize the performance of your MS SQL Server Always On availability groups.



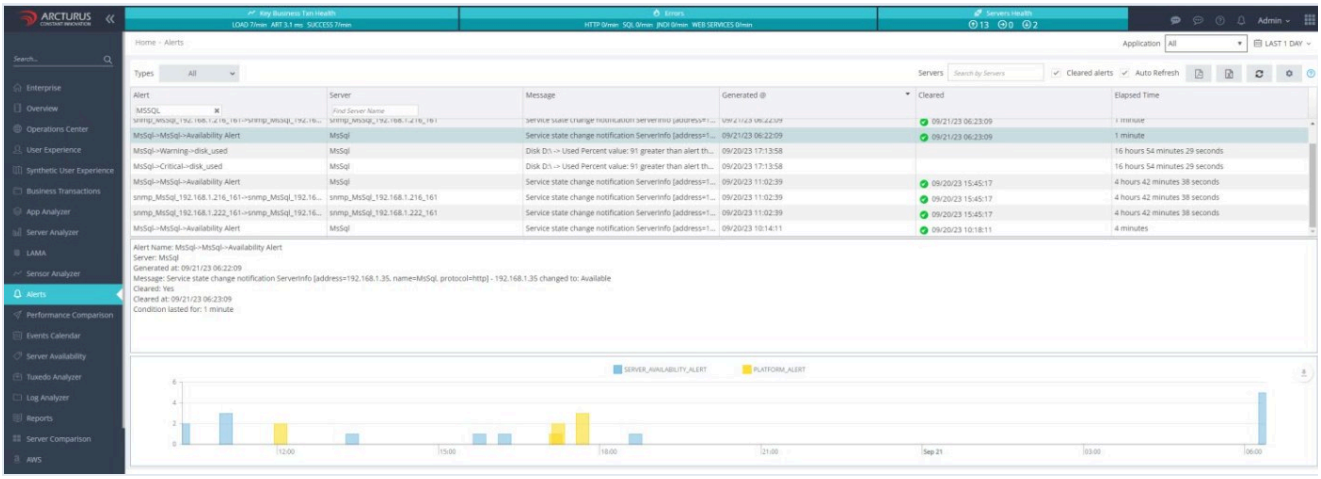
Reports Scheduling

Auto report generation and delivery through scheduling can save time and effort, leading to enhanced performance and efficiency.



Alerts

Alerts notify you of potential issues or opportunities, allowing you to take prompt corrective action reduce impact.



Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

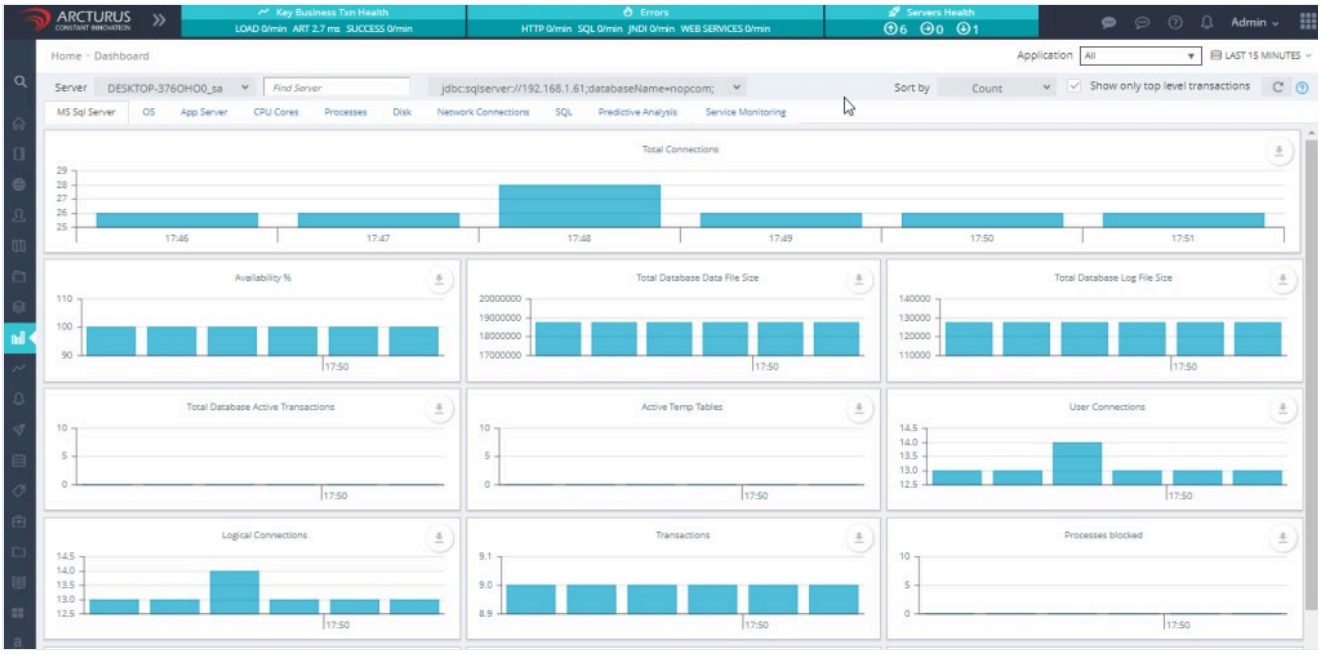
Monitoring for MS SQL Server databases

Applicare 9.2 · Updated August 22, 2024 · sql-server · **Pending technical review**

Access the Applicare controller and navigate to the **Server Analyzer m**enu to view the details.

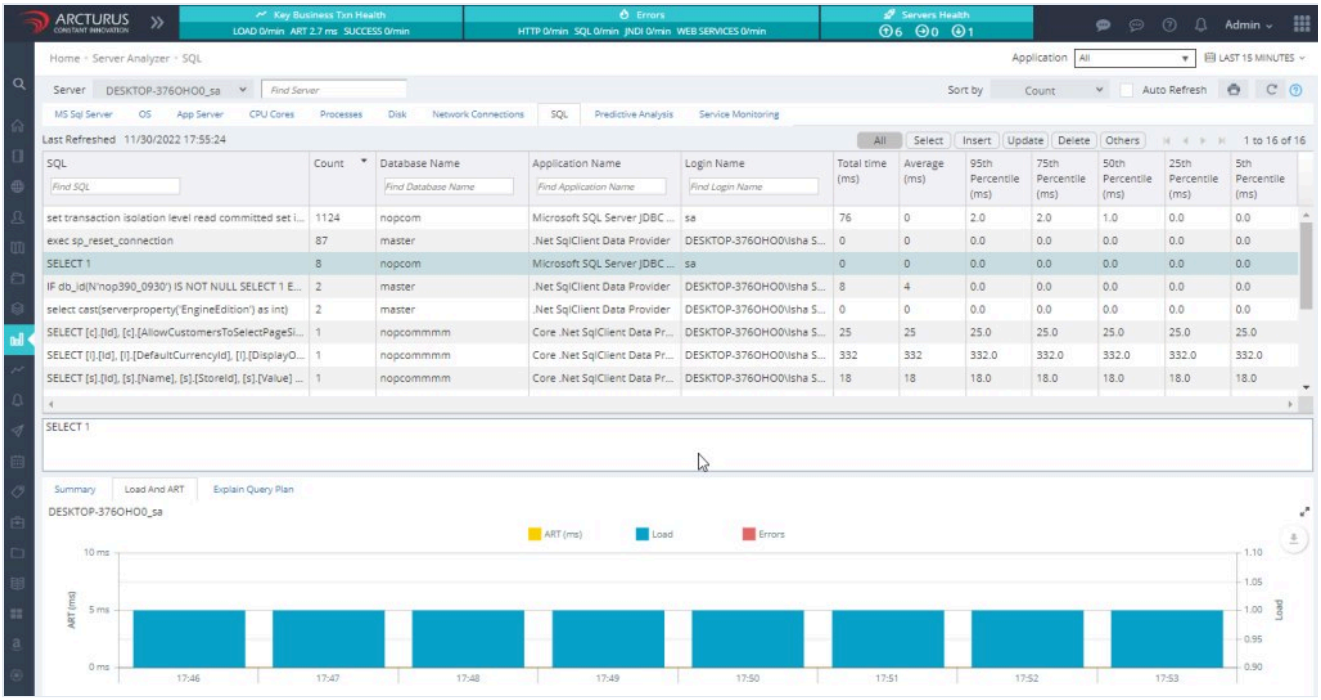
Dashboard

Database performance metrics help you monitor and improve the performance of your database. They cover the whole server, not just the specific database mentioned in the JDBC URL.

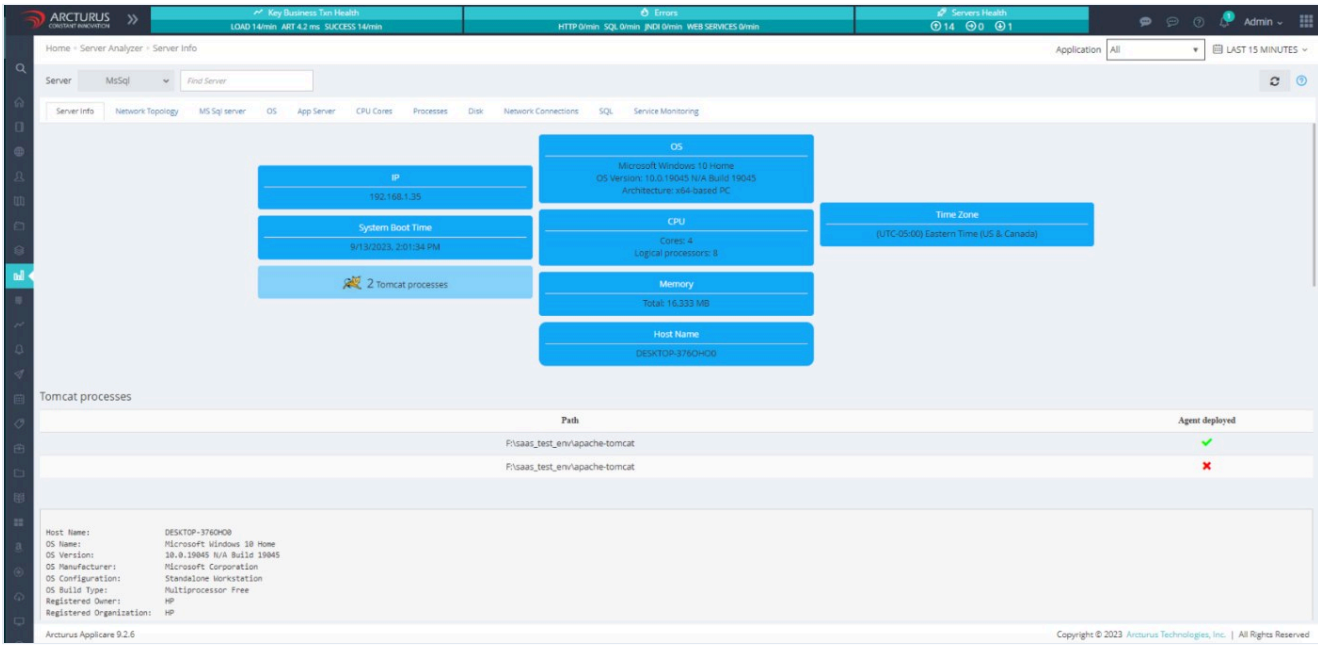


SQL

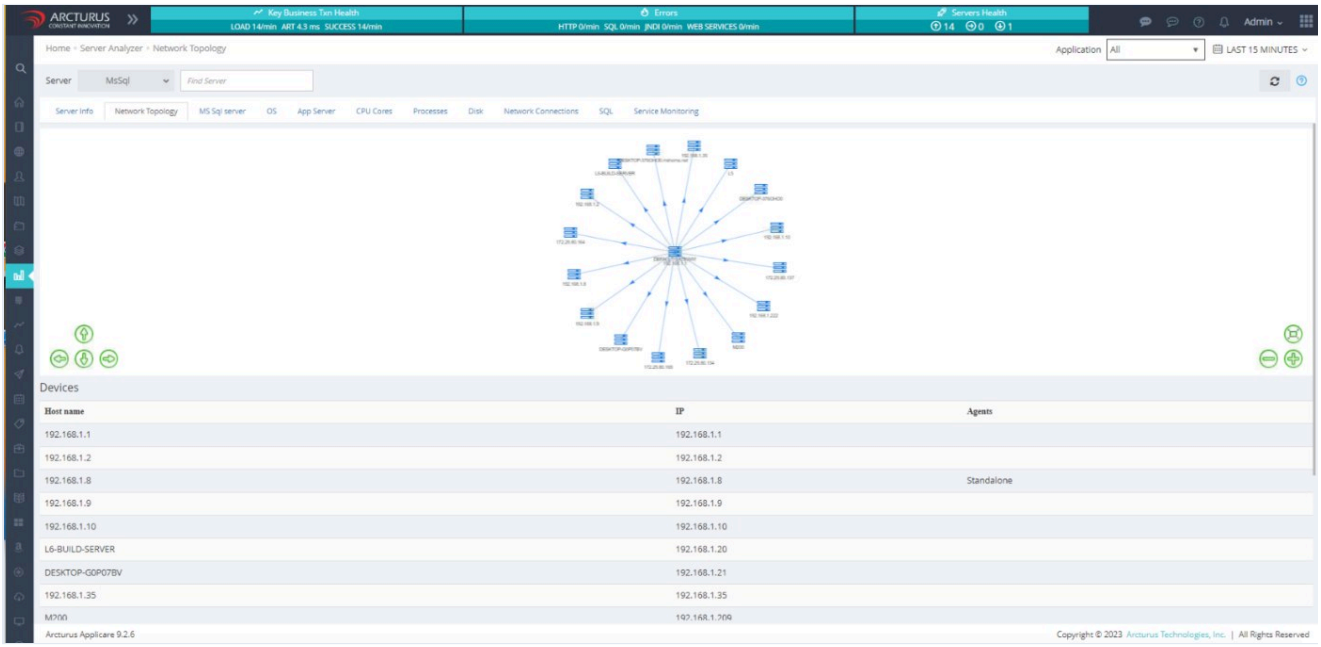
SQL Statement Profiling shows how well your database queries are performing. It helps you see which queries are slow and find out what's causing performance problems. You'll get details like the number of queries, their average time, and other useful stats.


ARCTURUS
Key Business Txn Health
LOAD 0/min ART 2.7 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
6 0 1
Admin
Home · Server Analyzer · SQL
Application All
LAST 15 MINUTES
Server DESKTOP-376OHO0_sa
Find Server
Sort by Count
Auto Refresh
MS Sql Server OS App Server CPU Cores Processes Disk Network Connections SQL Predictive Analysis Service Monitoring
Last Refreshed 11/30/2022 17:55:24
All Select Insert Update Delete Others
1 to 16 of 16
SQL Count Database Name Application Name Login Name Total time (ms) Average (ms) 95th Percentile (ms) 75th Percentile (ms) 50th Percentile (ms) 25th Percentile (ms) 5th Percentile (ms)
set transaction isolation level read committed set i... 1124 nopcom Microsoft SQL Server JDBC ... sa 76 0 2.0 2.0 1.0 0.0 0.0
exec sp_reset_connection 87 master .Net SqlClient Data Provider DESKTOP-376OHO0\Isha S... 0 0 0.0 0.0 0.0 0.0 0.0
SELECT 1 8 nopcom Microsoft SQL Server JDBC ... sa 0 0 0.0 0.0 0.0 0.0 0.0
IF db_id(N'nop390_0930') IS NOT NULL SELECT 1 E... 2 master .Net SqlClient Data Provider DESKTOP-376OHO0\Isha S... 8 4 0.0 0.0 0.0 0.0 0.0
select cast(serverproperty('EngineEdition') as int) 2 master .Net SqlClient Data Provider DESKTOP-376OHO0\Isha S... 0 0 0.0 0.0 0.0 0.0 0.0
SELECT [c].[Id], [c].[AllowCustomersToSelectPageSi... 1 nopcommmm Core .Net SqlClient Data Pr... DESKTOP-376OHO0\Isha S... 25 25 25.0 25.0 25.0 25.0 25.0
SELECT [i].[Id], [i].[DefaultCurrencyId], [i].[DisplayO... 1 nopcommmm Core .Net SqlClient Data Pr... DESKTOP-376OHO0\Isha S... 332 332 332.0 332.0 332.0 332.0 332.0
SELECT [s].[Id], [s].[Name], [s].[StoreId], [s].[Value] ... 1 nopcommmm Core .Net SqlClient Data Pr... DESKTOP-376OHO0\Isha S... 18 18 18.0 18.0 18.0 18.0 18.0
SELECT 1
Summary Load And ART Explain Query Plan
DESKTOP-376OHO0_sa
ART (ms) Load Errors
10 ms
5 ms
0 ms
1.10
1.05
1.00
0.95
0.90
17:46 17:47 17:48 17:49 17:50 17:51 17:52 17:53

Server Info

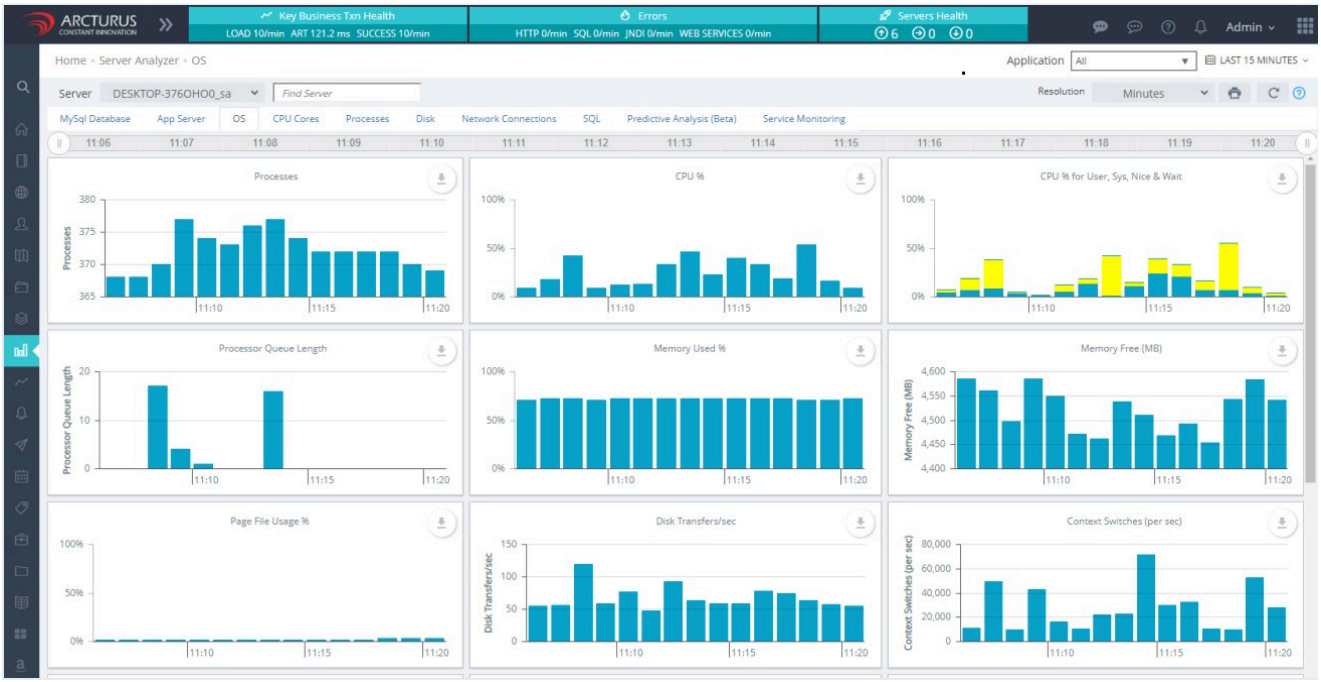

ARCTURUS
Key Business Txn Health
LOAD 14/min ART 4.2 ms SUCCESS 14/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
14 0 1
Admin
Home · Server Analyzer · Server Info
Application All
LAST 15 MINUTES
Server MsSql
Find Server
Server Info Network Topology MS Sql server OS App Server CPU Cores Processes Disk Network Connections SQL Service Monitoring
IP
192.168.1.35
OS
Microsoft Windows 10 Home
OS Version: 10.0.19045 N/A Build 19045
Architecture: x64-based PC
System Boot Time
9/13/2023, 2:01:34 PM
CPU
Cores: 4
Logical processors: 8
Time Zone
(UTC-05:00) Eastern Time (US & Canada)
2 Tomcat processes
Memory
Total: 16,333 MB
Host Name
DESKTOP-376OHO0
Tomcat processes
Path
Agent deployed
F:\saas_test_env\apache-tomcat
F:\saas_test_env\apache-tomcat
Host Name: DESKTOP-376OHO0
OS Name: Microsoft Windows 10 Home
OS Version: 10.0.19045 N/A Build 19045
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: HP
Registered Organization: HP
Arcturus Applicare 9.2.6
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved

Network Topology



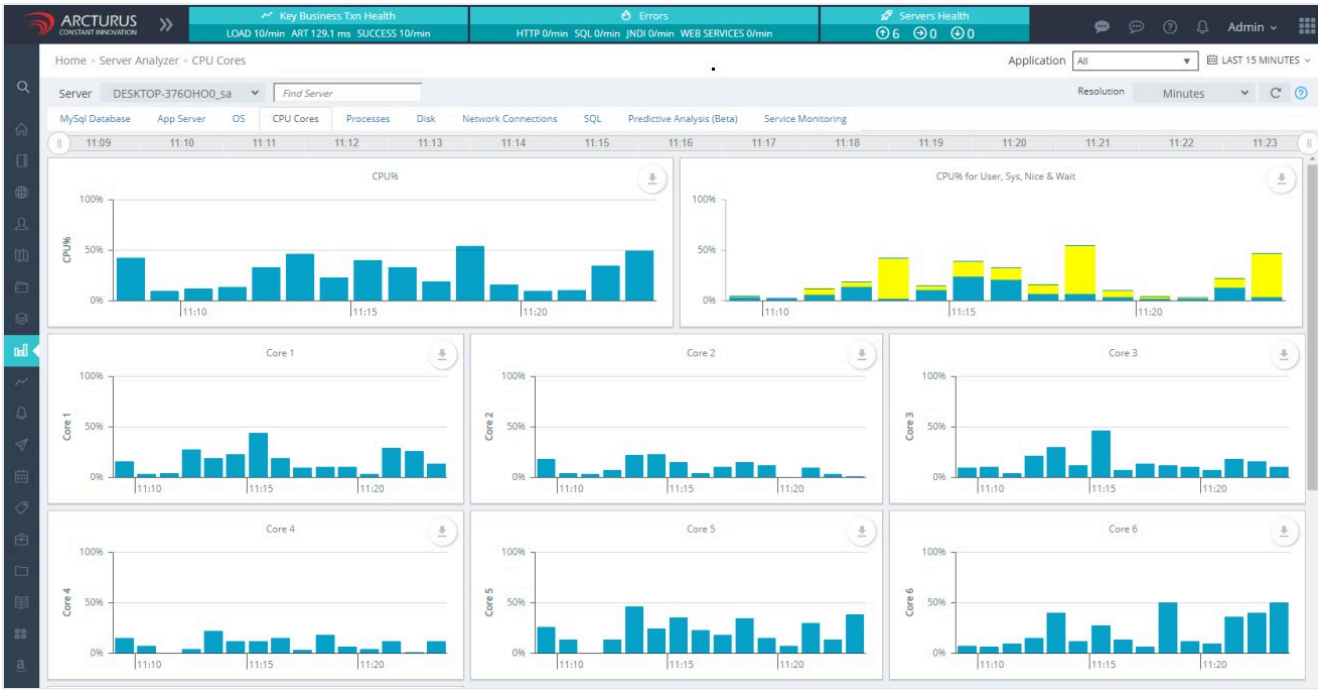
OS

This view offers a quick and easy overview of your physical server's or VM's performance, including details on processes, context switches, CPU usage, memory, and storage performance..



CPU CORE Analyzer

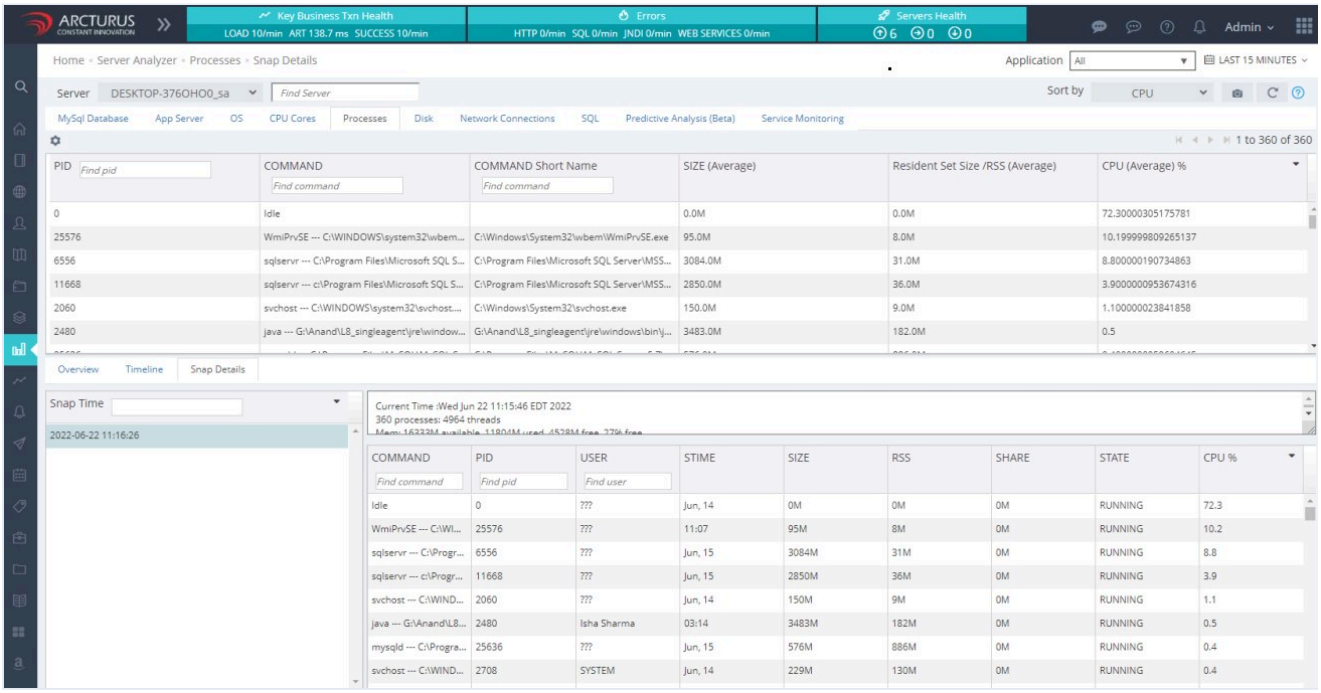
This view offers a comprehensive overview of the CPU including details information about each individual CPU core.



Processes

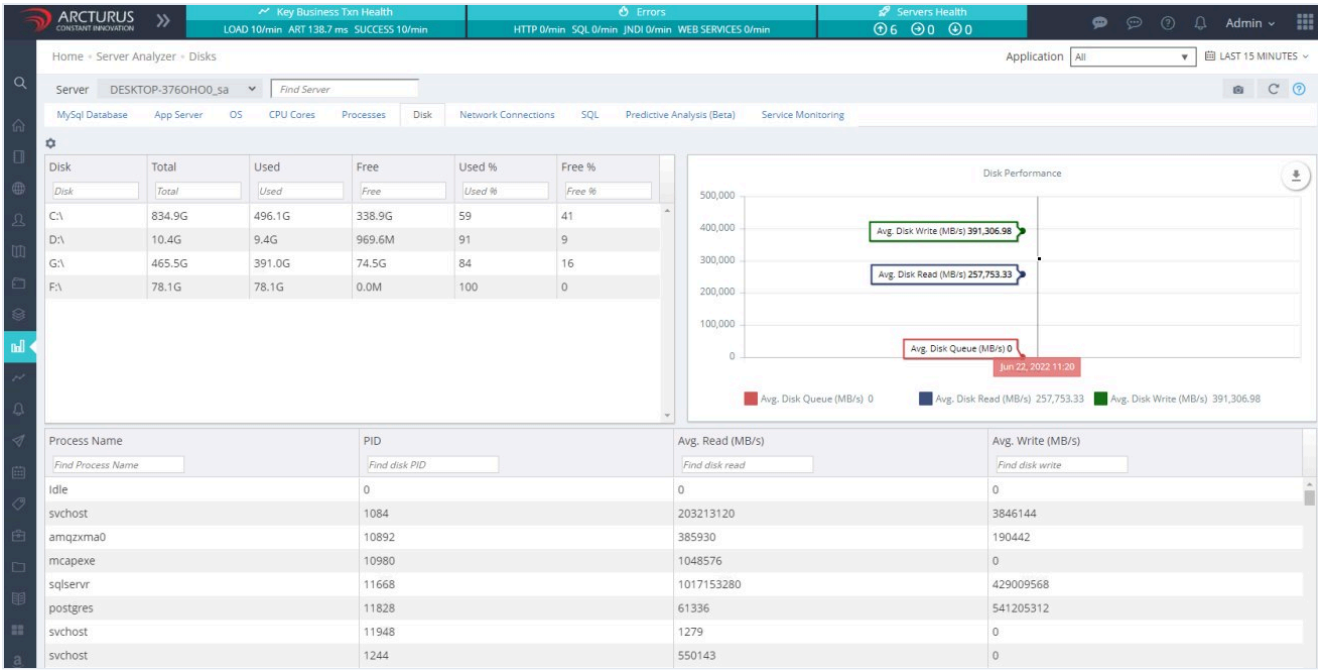
This view offers organized performance stats for the CPU, memory and running processes.

It includes historical snapshots of processes showing resource usage at various points in time. by default, Applicare takes a snapshot every 10 minutes but this interval can be adjusted according to your needs.



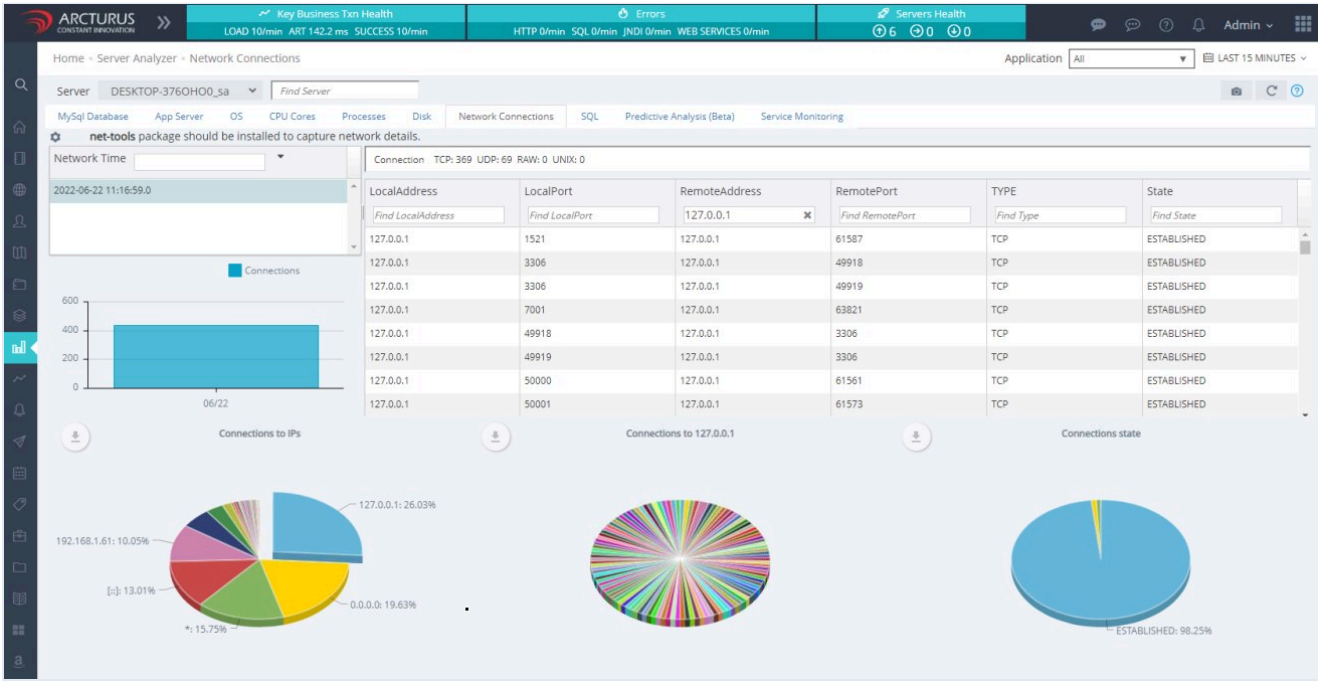
Disk

This view provides the disk usage accessed by each and every process.



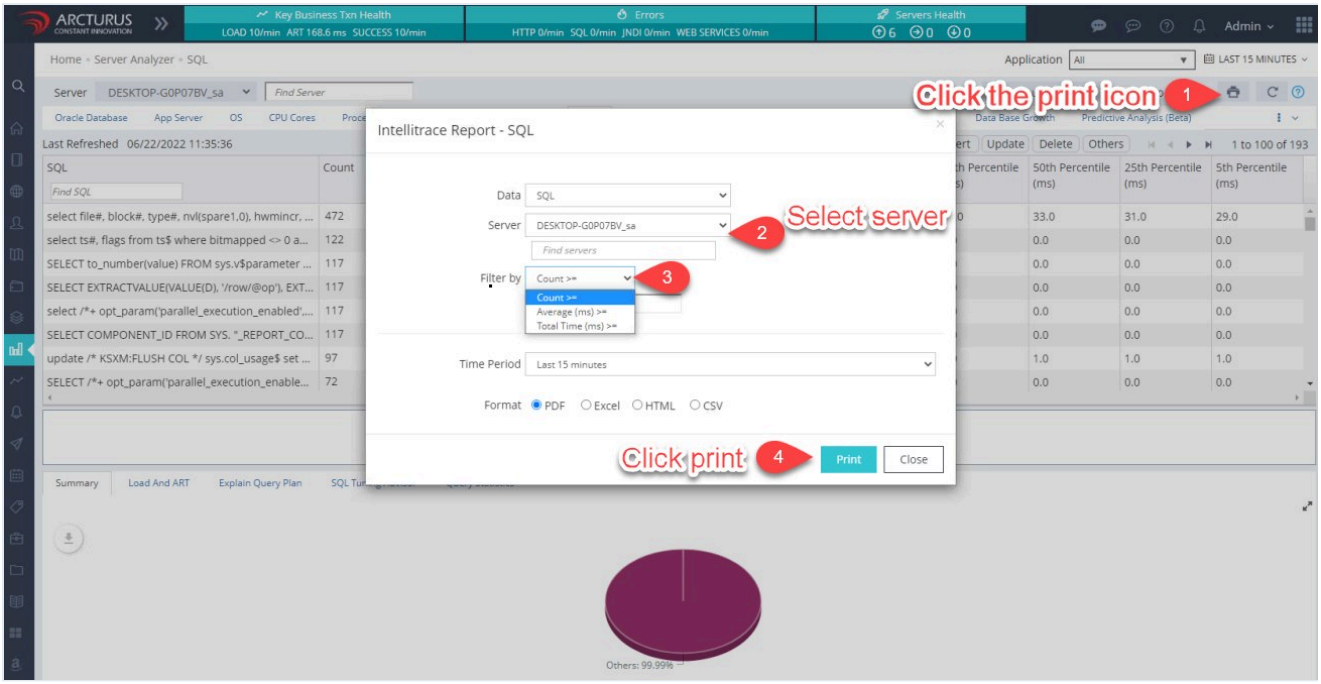
Network Connections

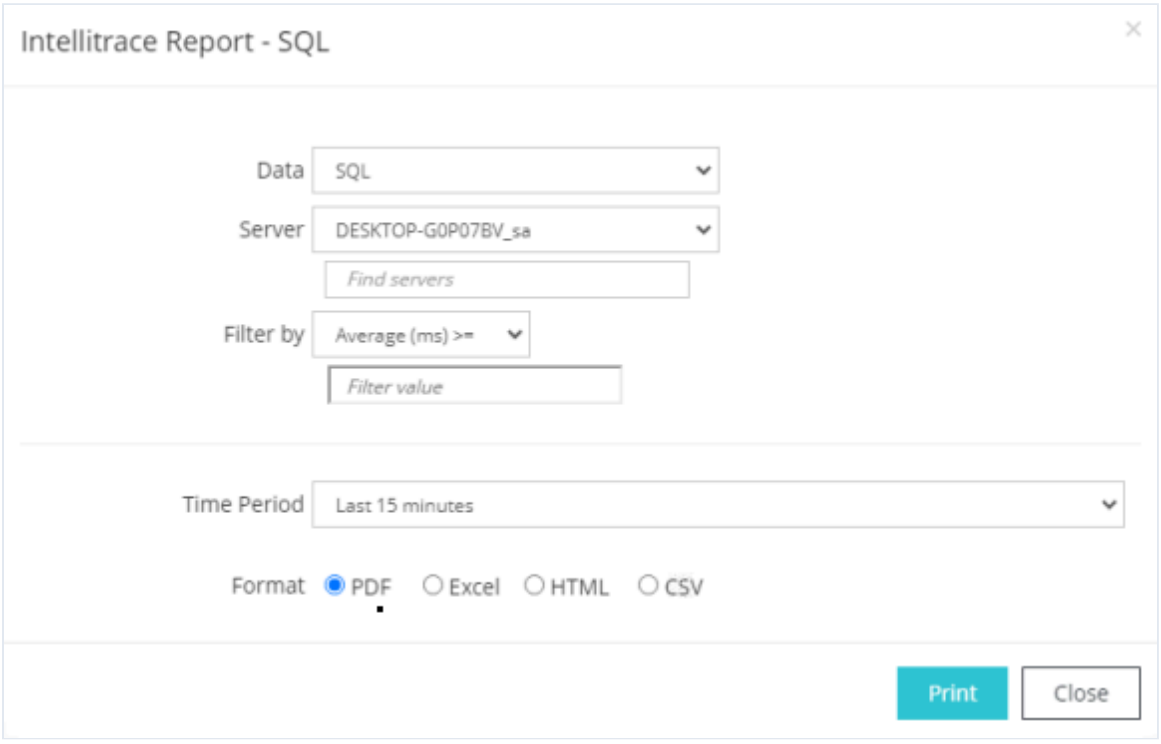
This view provides the TCP, UDP, RAW, and UNIX connection Details.



SQL Report

The SQL statements can be downloaded as a report by clicking the print icon from the Server Analyzer menu SQL Screen.


Click the print icon
1
Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
2
Select server
Find servers
Filter by
Count >=
3
Count >=
Average (ms) >=
Total Time (ms) >=
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Click print
4
Print
Close


Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
Find servers
Filter by
Average (ms) >=
Filter value
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Print
Close

SQL	Database Name	Count	Total Time (ms)	Avg. Time (ms)	Minimum (ms)	Maximum (ms)	Std. Dev. (ms)	95th Percentile (ms)	75th Percentile (ms)	50th Percentile (ms)	25th Percentile (ms)	5th Percentile (ms)
ORDER BY [alt].[Name]												
SELECT [c].[Id], [c].[AllowCustomersToSelectPageSize], [c].[CategoryTemplateId], [c].[CreatedOnUtc], [c].[Deleted], [c].[Description], [c].[DisplayOrder], [c].[IncludeInTopMenu], [c].[LimitedToStores], [c].[MetaDescription], [c].[MetaKeywords], [c].[MetaTitle], [c].[Name], [c].[PageSize], [c].[PageSizeOptions], [c].[ParentCategoryId], [c].[PictureId], [c].[PriceRanges], [c].[Published], [c].[ShowOnHomepage], [c].[SubjectToAcl], [c].[UpdatedOnUtc] FROM [Category] AS [c] WHERE (([c].[Published] = 1) AND ([c].[Deleted] = 0)) AND ([c].[ShowOnHomepage] = 1) ORDER BY [c].[DisplayOrder], [c].[Id]	nopcommmm	1	25	25	25	25	0	25	25	25	25	25
SELECT [p].[Id], [p].[Name] FROM [ProductTag] AS [p]	nopcommmm	1	25	25	25	25	0	25	25	25	25	25
SELECT [s].[Id], [s].[Name], [s].[StoreId], [s].[Value] FROM [Setting] AS [s] ORDER BY [s].[Name], [s].[StoreId]	nopcommmm	1	18	18	18	18	0	18	18	18	18	18
SELECT COUNT(*) FROM [Manufacturer] AS [m] WHERE ([m].[Published] = 1) AND ([m].[Deleted] = 0)	nopcommmm	1	17	17	17	17	0	17	17	17	17	17
SELECT [sao].[Id], [sao].[ColorSquaresRgb], [sao].[DisplayOrder], [sao].[Name], [sao].[SpecificationAttributeId] FROM [SpecificationAttributeOption] AS [sao] WHERE [sao].[Id] IN (14, 15, 16)	nopcommmm	1	17	17	17	17	0	17	17	17	17	17
SELECT [p].[Id], [p].[AdditionalShippingCharge], [p].[AdminComment], [p].[AllowAddingOnlyExistingAttributeCombinations], [p].[AllowBackInStockSubscriptions], [p].[AllowCustomerReviews], [p].[AllowedQuantities], [p].[ApprovedRatingSum], [p].[ApprovedTotalReviews], [p].[AutomaticallyAddRequiredProducts], [p].[AvailableEndDateTimeUtc], [p].[AvailableForPreOrder], [p].	nopcommmm	1	2	2	2	2	0	2	2	2	2	2

Created at: 2022-11-30 18:04:46.818 Time Zone: Eastern Standard Time America/New_York

MS SQL Server Always On Availability Groups Monitoring

MS SQl Server database configuration

CONFIGURATION GUIDES › DATABASES

Creating a new user for SQL Server authentication and grant the necessary permissions.

Applicare 9.2 · Updated September 9, 2024 · sql-server · **Pending technical review**

1. Grant the **db_datareader** role to all the databases, including the **master** database. 2. Grant the following permissions: **alter trace, view server state, view database state** and **alter any event session**.

Syntax:

```
TEXT
use master;
grant alter trace to <new_user>;
grant view server state to <new_user>;
grant view database state to <new_user>;
grant alter any event session to <new_user>;
```

Replace **<new_user>** with the username of the newly created user.

e.g. If the new user name is **appms,** use that in place of **<new user>**

```
TEXT
use master;
grant alter trace to appms;
grant view server state to appms;
grant view database state to appms;
grant alter any event session to appms;
```

CONFIGURATION GUIDES › DATABASES

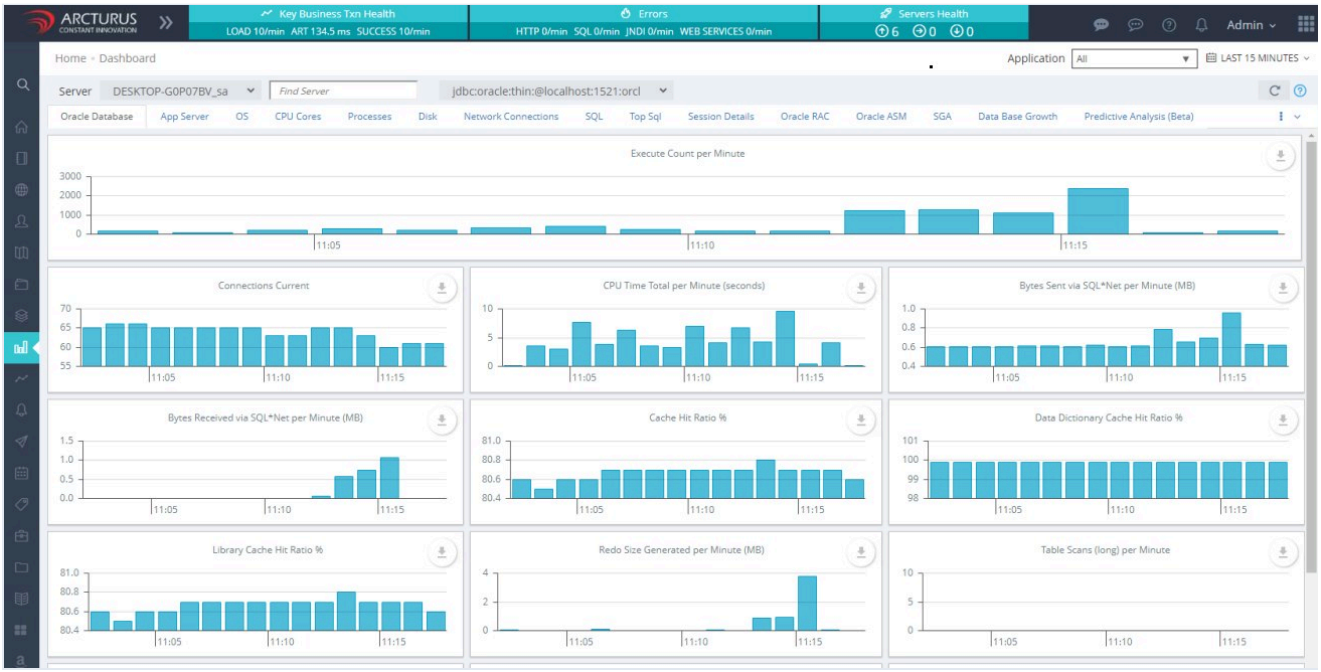
Oracle Database Monitoring

Applicare 9.2 · Updated August 22, 2024 · oracle · **Pending technical review**

Log in to Applicare console and navigate to the **Server Analyzer** menu item on the left to view the details.

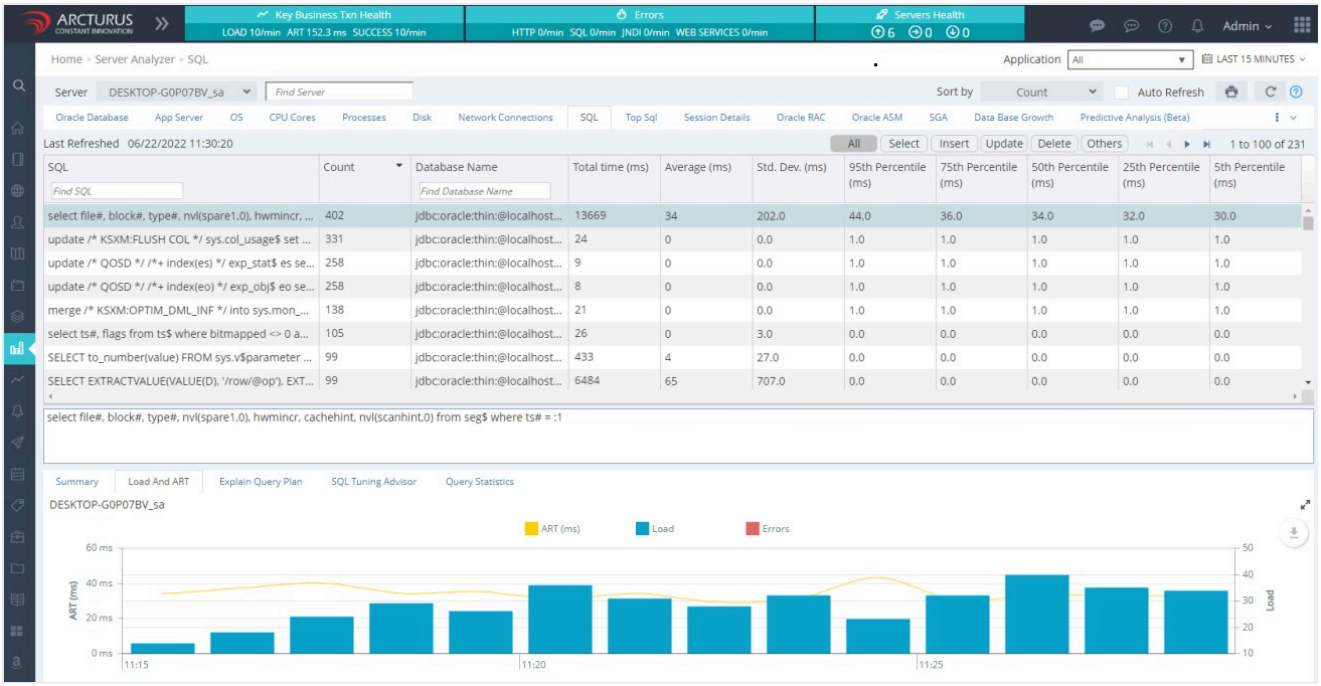
Dashboard

Database performance metrics assist monitoring and optimizing your database's performance business. These metrics are calculated for the entire server, not just the database specified in the JDBC URL.



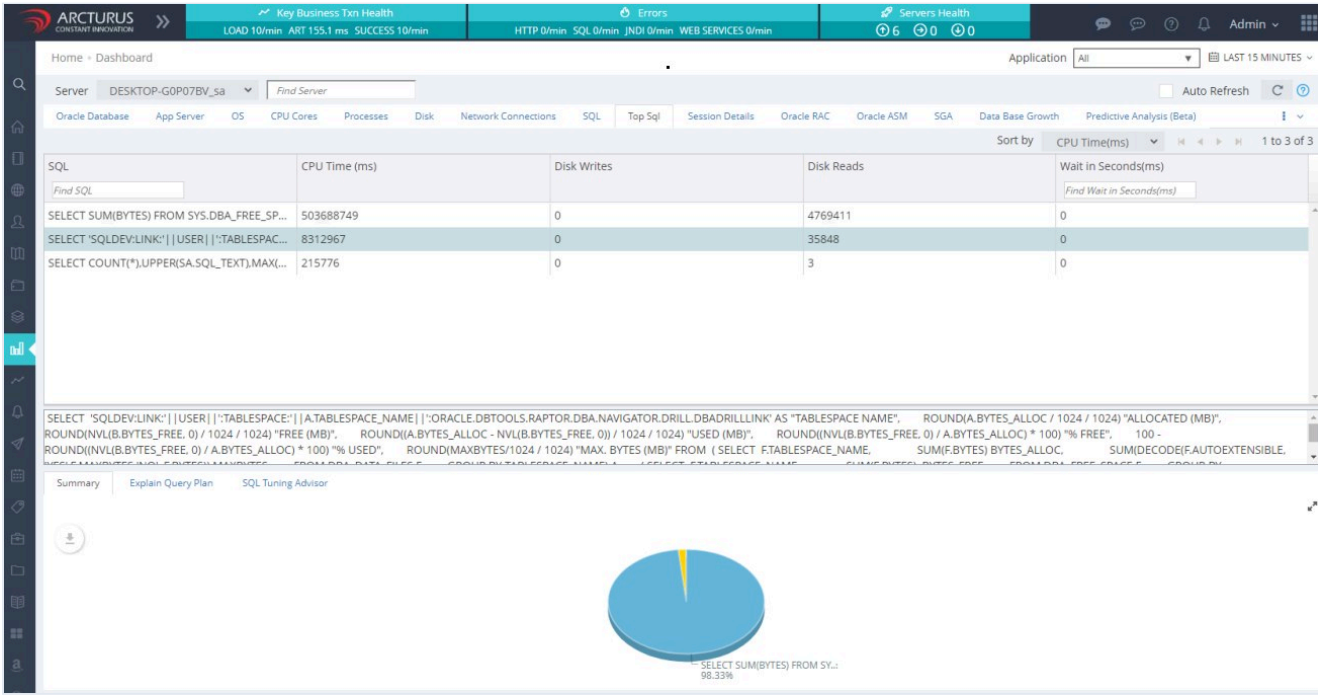
SQL

SQL Statement Profiling offers performance statistics for database queries. Helping you identify which statements most time-consuming and diagnose performance issues swiftly. It also provides detailed stats including information of the database, query count, average execution time and various percentiles.



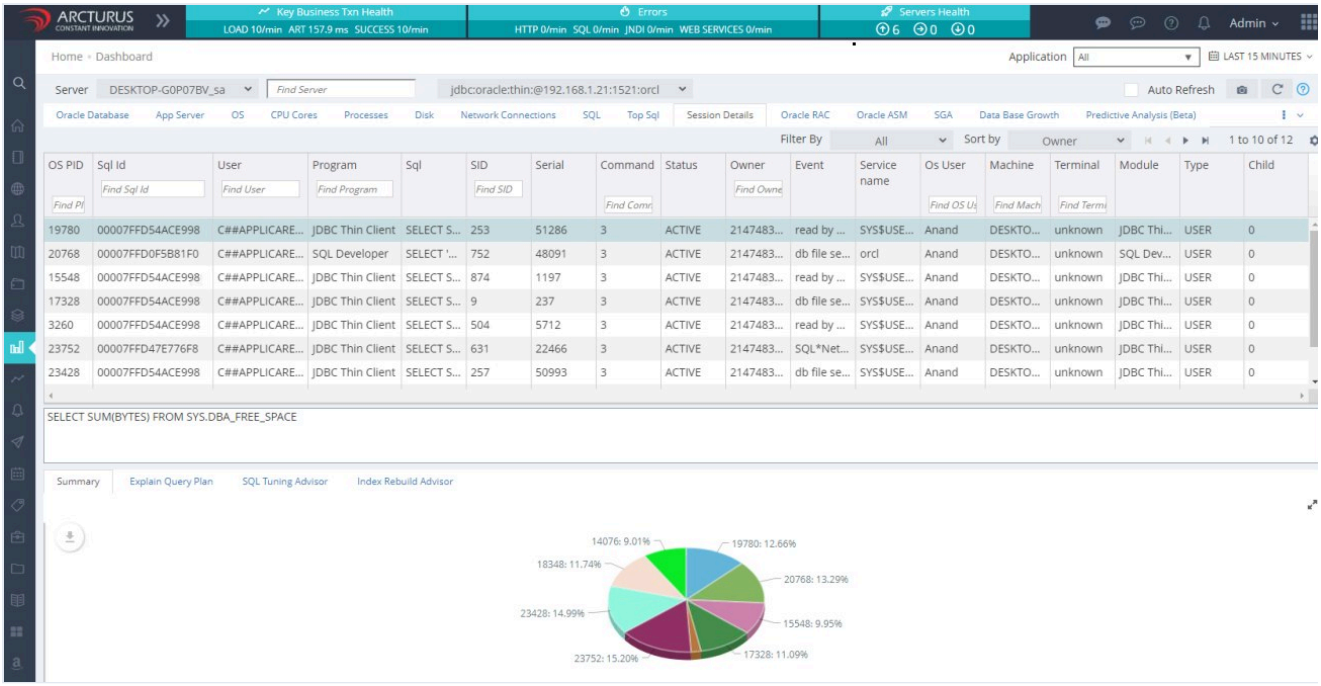
Top SQL Details

This view shows the SQL statements that are executed most frequently and consume more system resources compared to others.



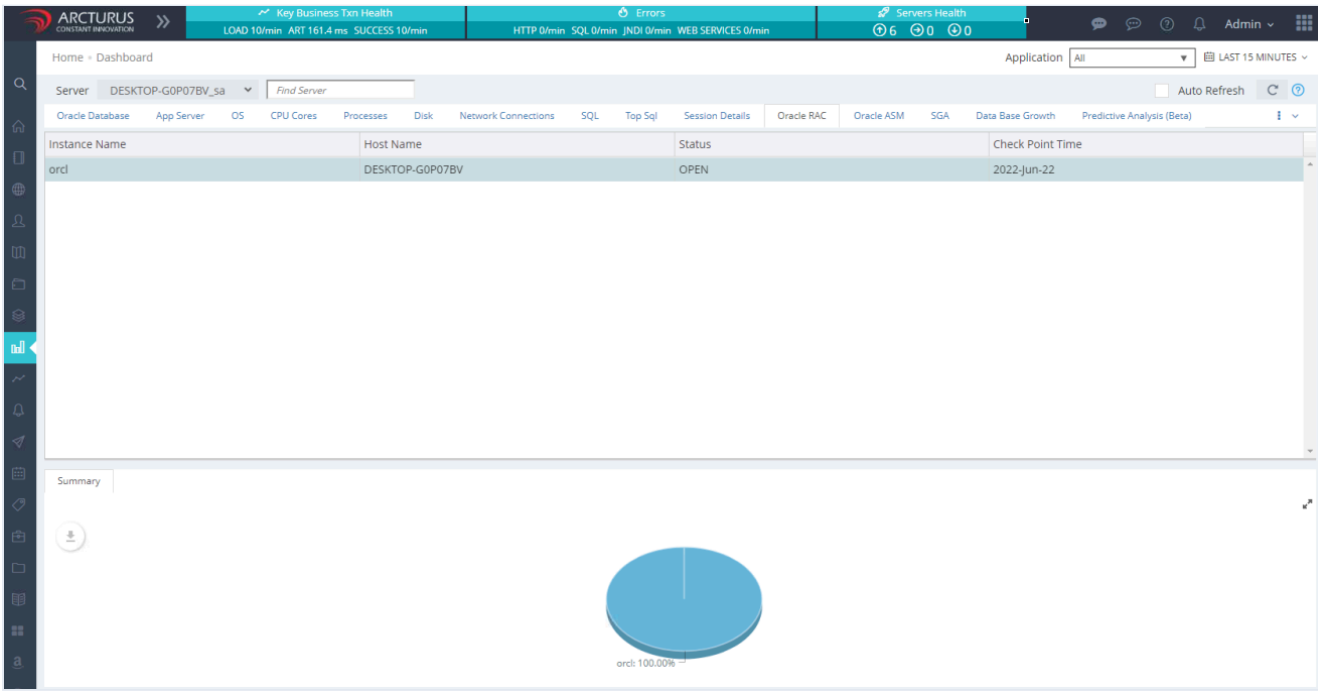
Session Details

It provides details about the current user's session.



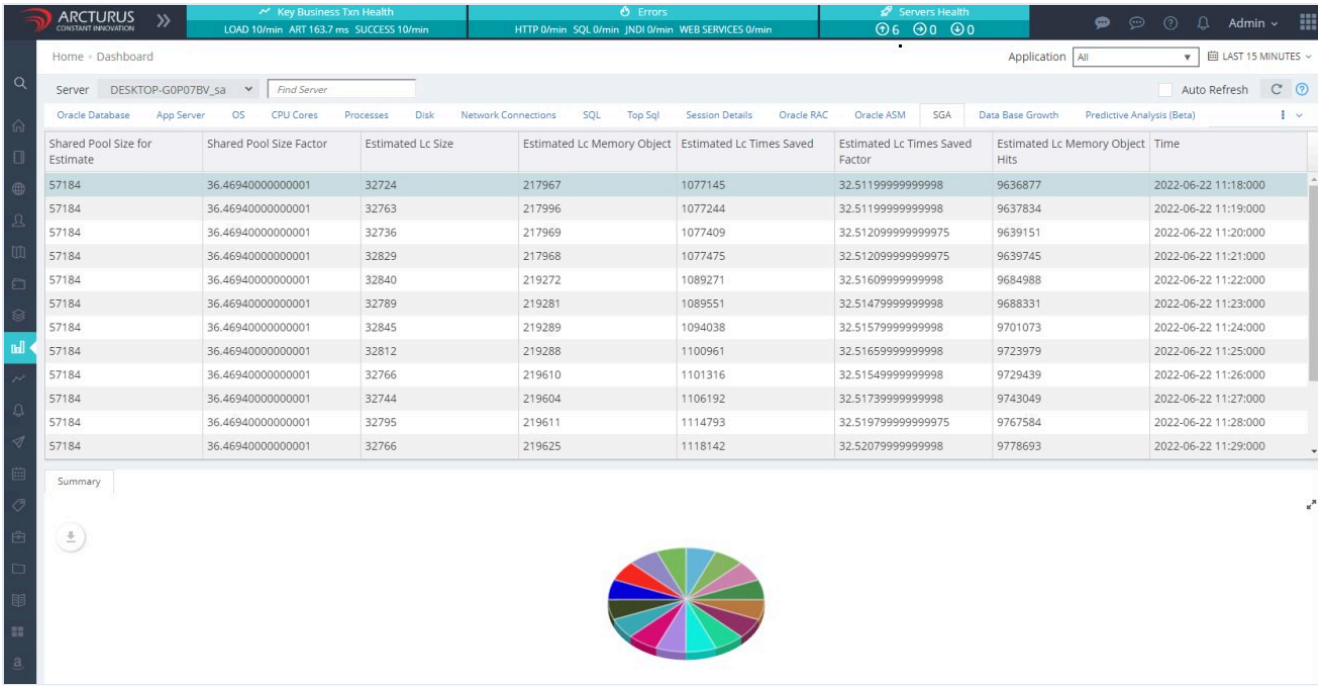
RAC Details

Oracle Real Application Clusters (RAC) allow customers to run a single Oracle Database across multiple servers in order to maximize availability and enable horizontal scalability, while accessing shared storage.



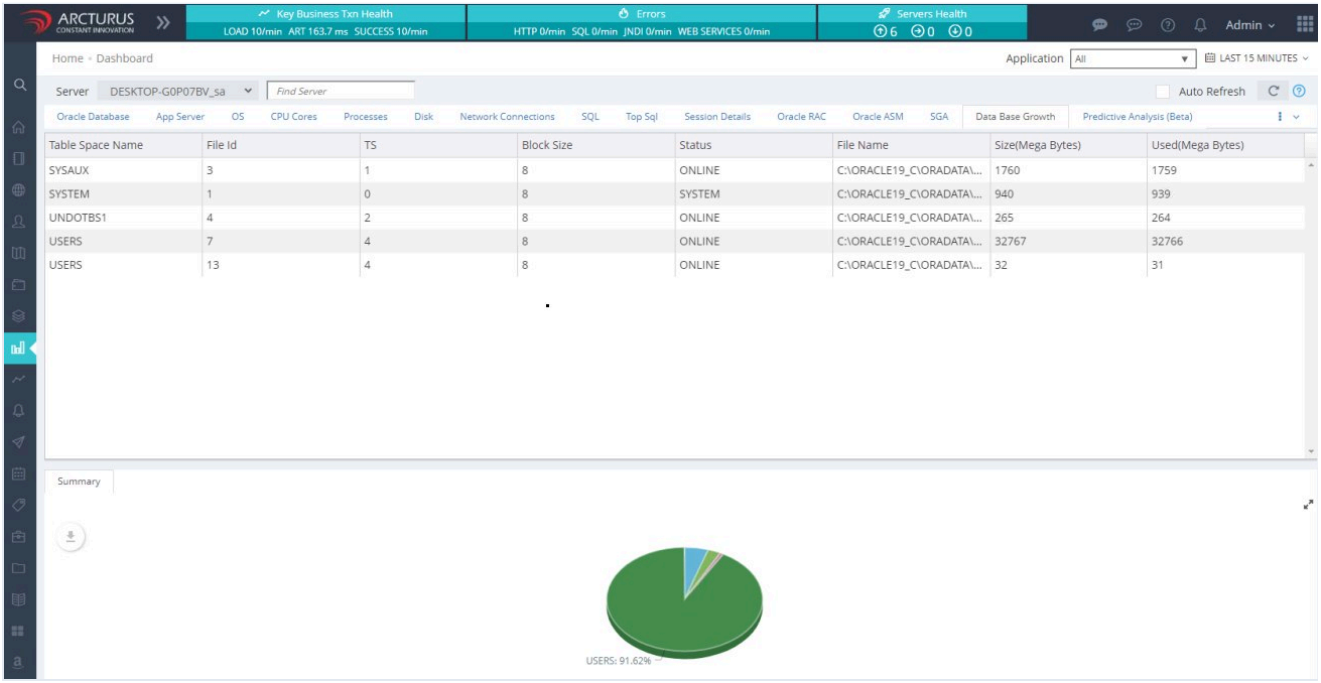
SGA Details

The System Global Area (SGA) is a collection of shared memory structures, referred to as SGA components, that store data and control information.



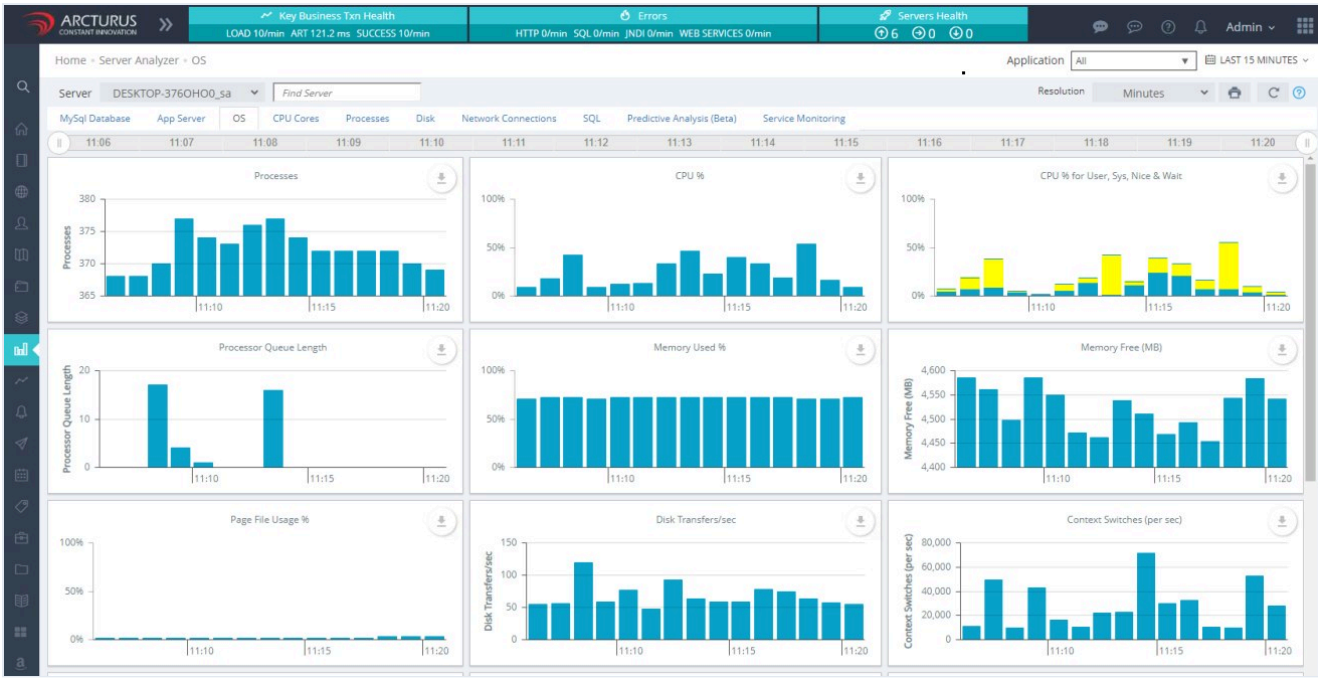
Database Growth

It display the Database Growth details.



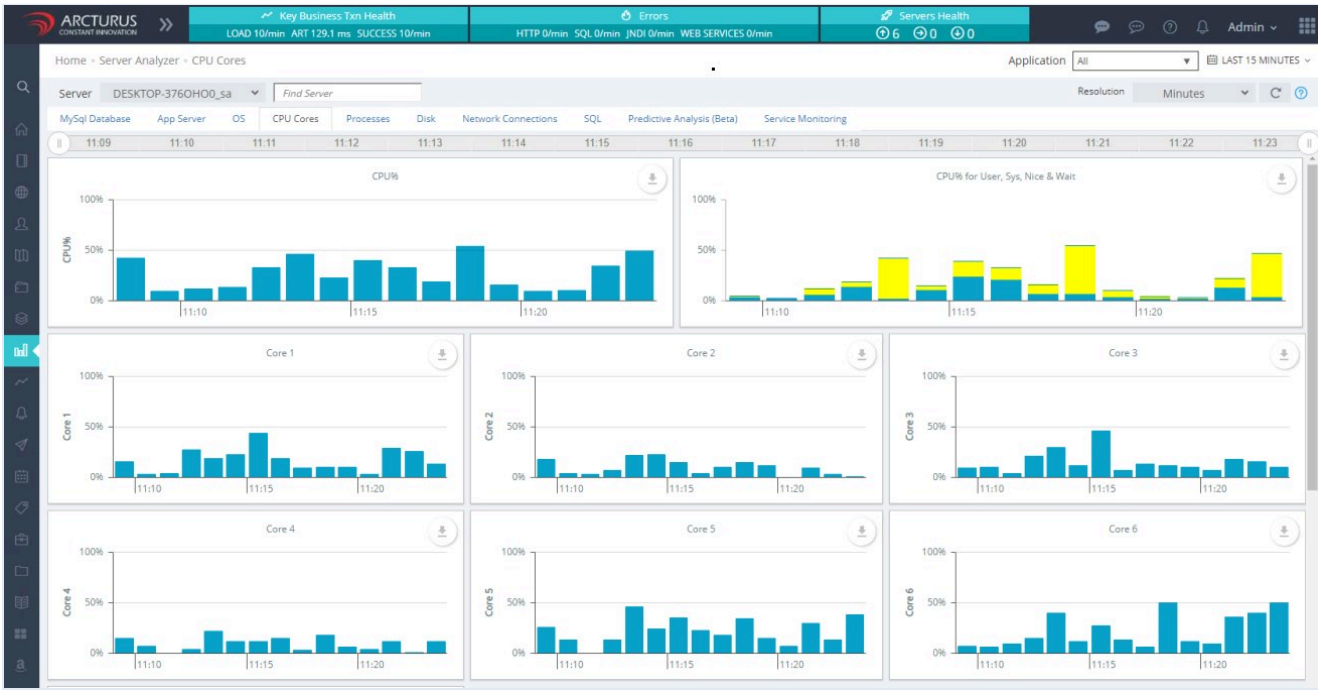
OS

This view offers a quick and easy overview of your physical server's or VM's performance, including metrics such as the number of processes, context switches, CPU Usage, memory and storage performance, among others.



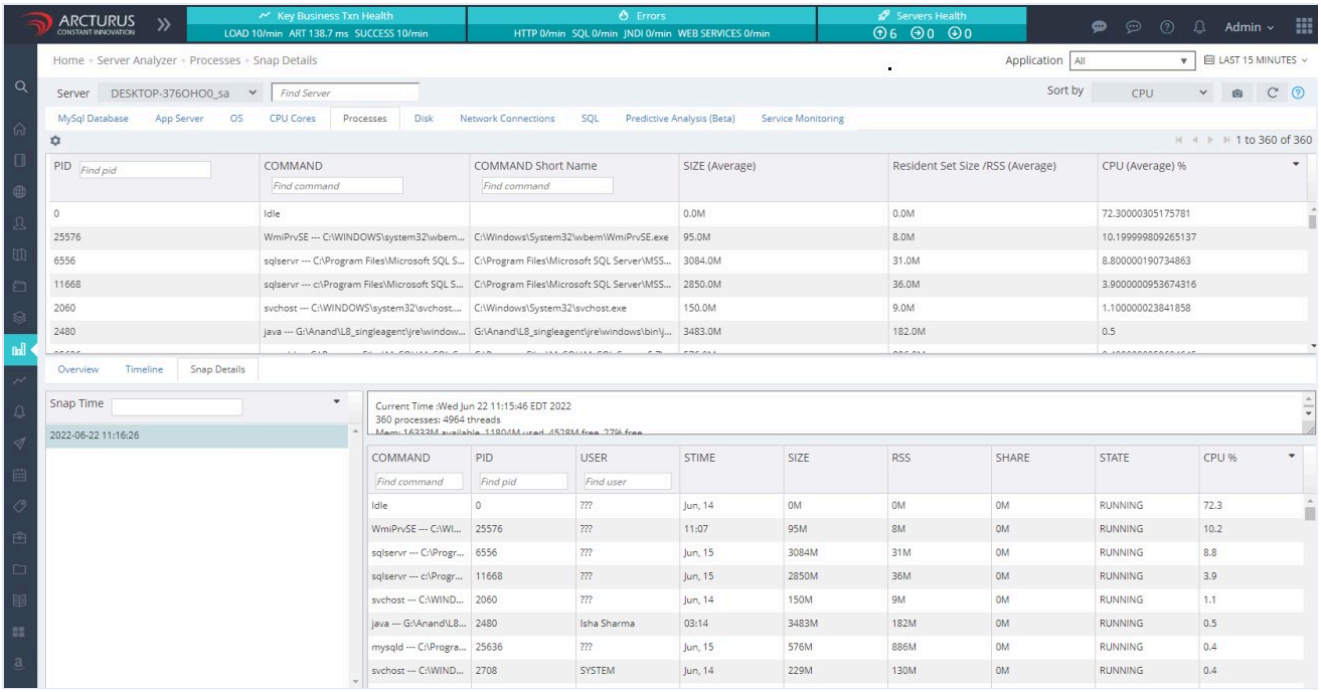
CPU Core Analyzer

This view provides the overall CPU and it's individual CPU core details.



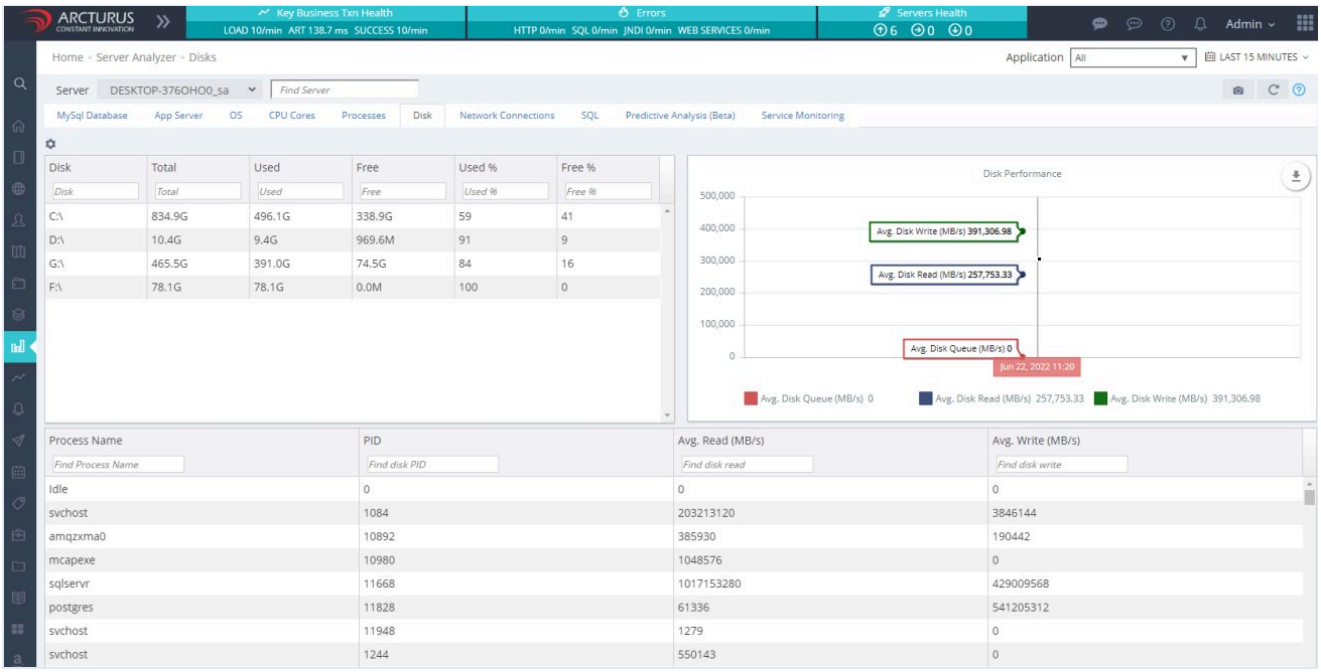
Processes

This view offers performance stats for CPU, memory and running processes. It includes historical snapshots of various processes. showing their resource consumption at specific times. By default, Applicare takes a snapshot every 10 minutes, but this interval can be adjusted through configuration.



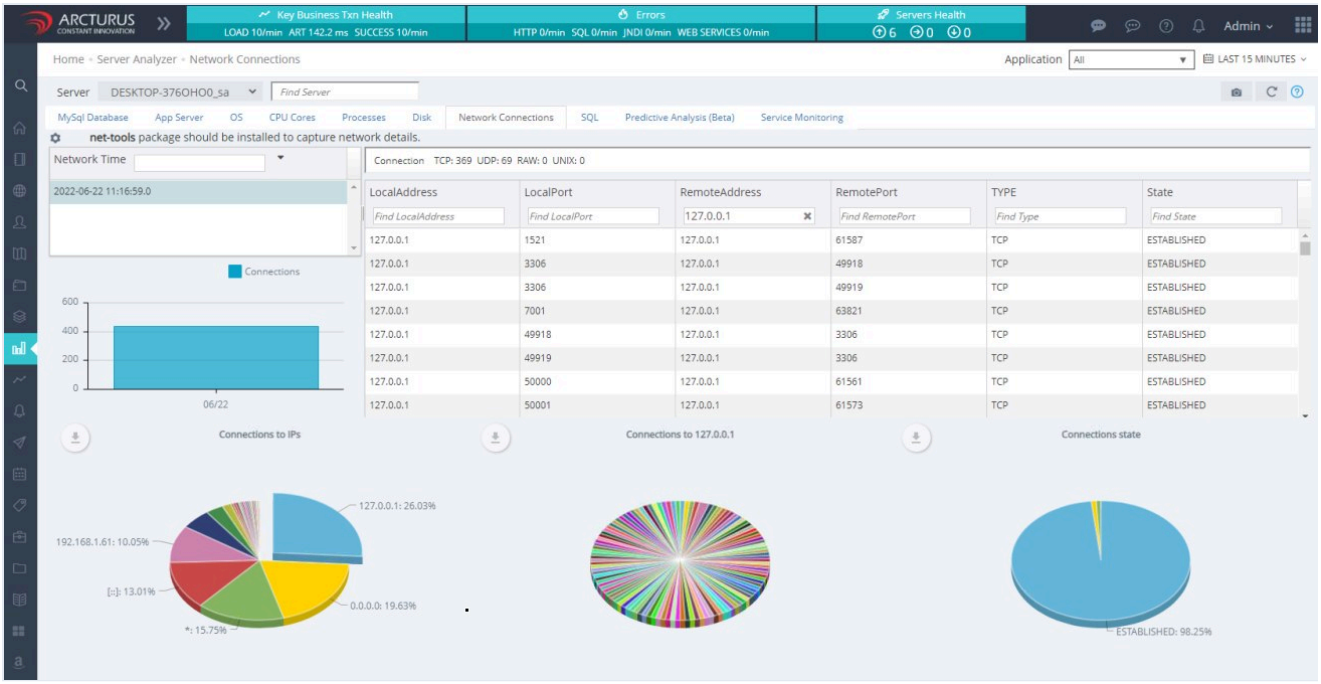
Disk

This view shows the disk usage for each individual process.



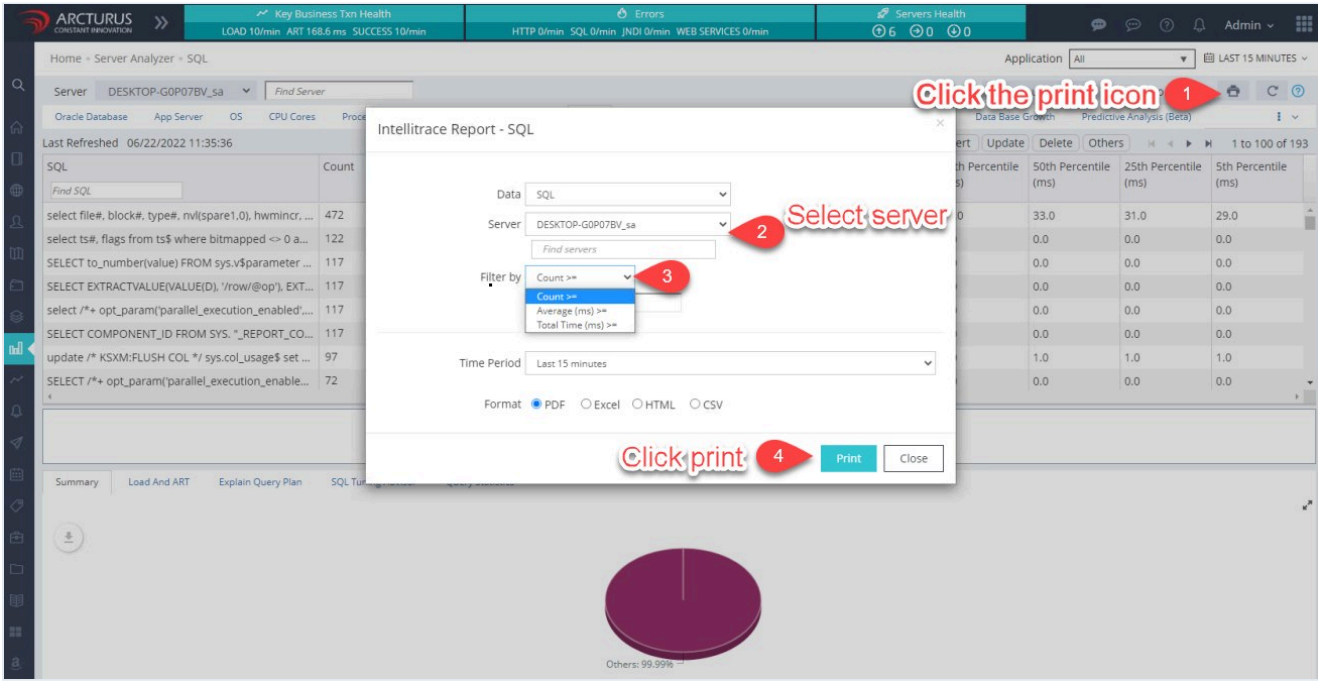
Network Connections

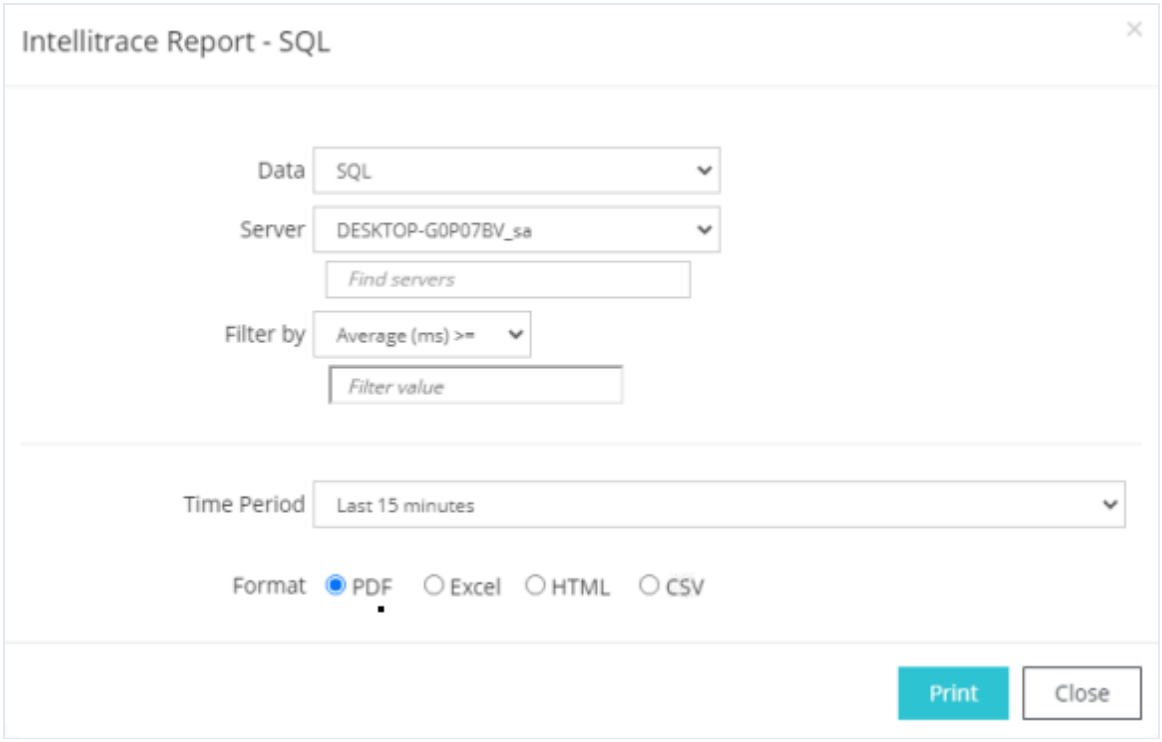
This view provides the TCP, UDP, RAW, and UNIX connection Details.



SQL Report

You can download the SQL Statements as a report by clicking the print icon from the SQL Screen within the Server Analyzer menu.







ARCTURUS
CONSTANT INNOVATION

IntelliTrace SQL Report

From: 2022-06-22 11:37:00.0 To: 2022-06-22 11:52:09.408 Server: DESKTOP-G0P07BV_sa

SQL	Database Name	Count	Total Time (ms)	Avg. Time (ms)	Minimum (ms)	Maximum (ms)	Std. Dev. (ms)	95th Percentile (ms)	75th Percentile (ms)	50th Percentile (ms)	25th Percentile (ms)	5th Percentile (ms)
SELECT SUM(bytes) FROM SYS. DBA_FREE_SPACE	jdbc:oracle:thin: @localhost:1521: orcl	13	10105709	777362	-1	0	66631	865219	826397	791066	742201	606030
SELECT name name , free_mb free_mb, total_mb total_mb, (free_mb/total_mb*100) percentage FROM v$asm_diskgroup	jdbc:oracle:thin: @localhost:1521: orcl	14	28758	2054	-1	0	766	2072	2060	2054	2046	2039
BEGIN sys.dbms_auto_report_internal. i_save_report (:rep_ref, :snap_id, :pr_class, : rep_id, :suc); END;	jdbc:oracle:thin: @localhost:1521: orcl	10	22376	2237	-1	0	2351	8459	2388	1227	1111	1054
select file#, block#, type#, nvl(spare1,0), hwmincr, cachehint, nvl(scanhint,0) from seg$ where ts# = :1	jdbc:oracle:thin: @localhost:1521: orcl	665	22046	33	-1	0	164	42	35	33	30	27
select ddf.TABLESPACE_NAME tableSpaceName, ddf.FILE_ID fileId, df.TS# ts, (df.BLOCK_SIZE/1024) blocksize,df.STATUS status,ddf.FILE_NAME fileName,(ddf.BYTES/ (1024*1024)) sizemb,(ddf.USER_BYTES/ (1024*1024)) usedmb from v$datafile df, dba_data_files ddf where df.FILE# = ddf. FILE_ID	jdbc:oracle:thin: @localhost:1521: orcl	14	13130	937	-1	0	1242	2473	1264	630	325	166
SELECT DBMS_REPORT.GET_REPORT(:B1) FROM DUAL	jdbc:oracle:thin: @localhost:1521: orcl	2	10859	5429	-1	0	2854	8284	8284	5429	2575	2575
SELECT XMLTYPE(DBMS_REPORT. GET_REPORT_WITH_SUMMARY(:B1)) FROM DUAL	jdbc:oracle:thin: @localhost:1521: orcl	8	10694	1336	-1	0	1578	2639	1245	1043	991	0
WITH MONITOR_DATA AS (SELECT INST_ID, KEY, NVL2(PX_QCSID, NULL, STATUS) STATUS, FIRST_REFRESH_TIME, LAST_REFRESH_TIME, REFRESH_COUNT, PROCESS_NAME, SID, SQL_ID, SQL_EXEC_START, SQL_EXEC_ID, DBOP_NAME, DBOP_EXEC_ID, SQL_PLAN_HASH_VALUE,	jdbc:oracle:thin: @localhost:1521: orcl	8	10070	1258	-1	0	1458	2459	1173	969	927	81

Created at: 2022-06-22 11:52:09.408 Time Zone: Eastern Standard Time America/New_York

Arcturus Applicare Page 1 of 74

Configuration

Configuring Monitoring for Oracle Databases

CONFIGURATION GUIDES › DATABASES

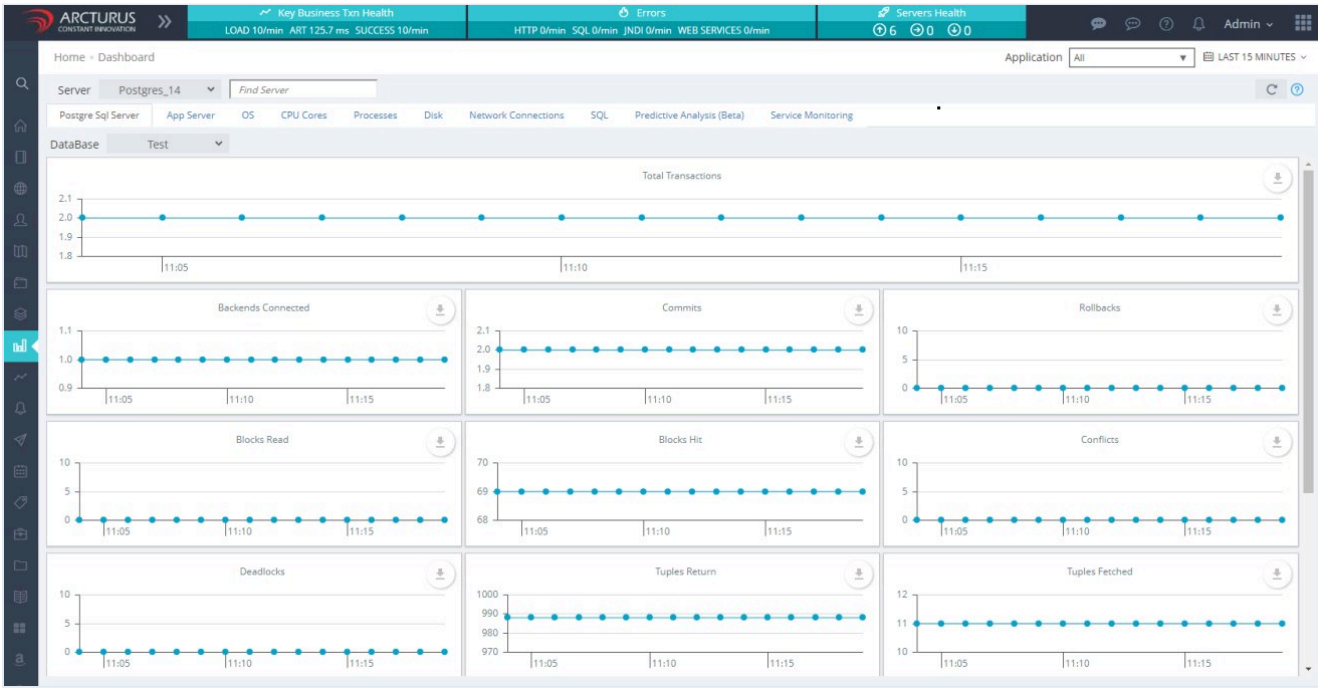
PostgreSQL Database Monitoring

Applicare 9.2 · Updated September 13, 2024 · postgresql · **Pending technical review**

Log in to Applicare controller and navigate to the **Server Analyzer** menu to view the details.

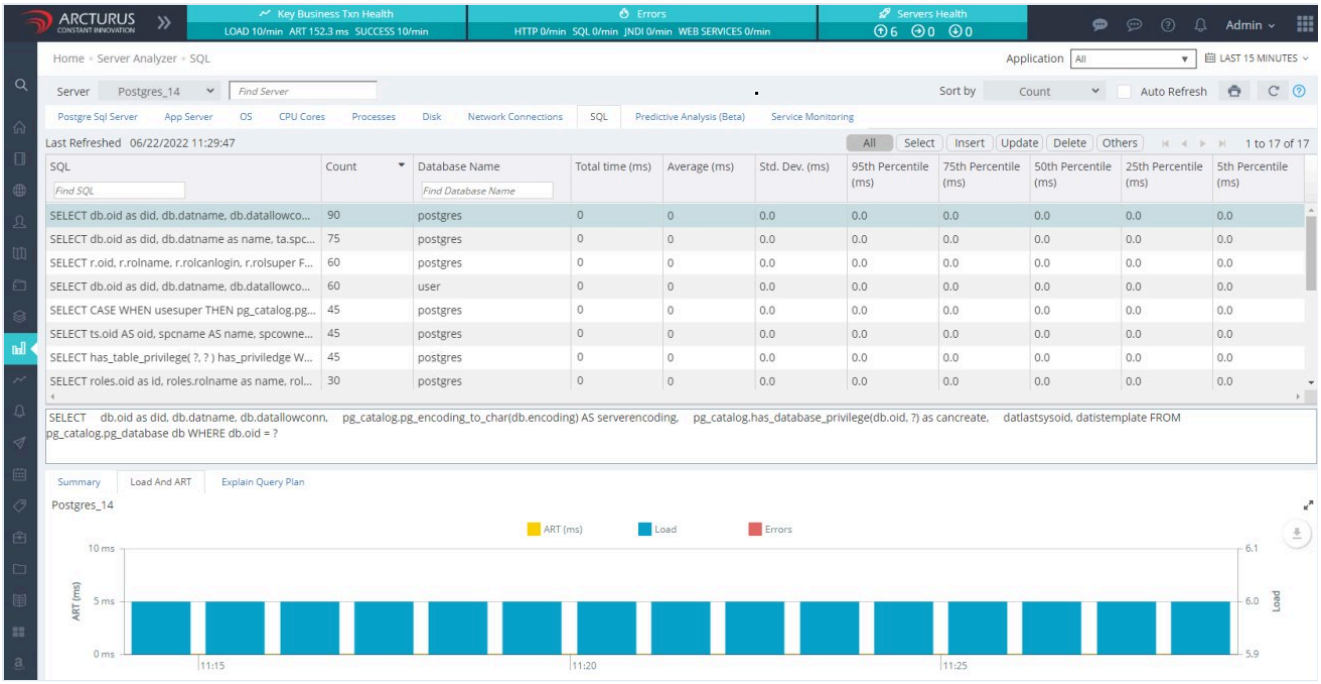
Dashboard

Database performance metrics assist in monitoring and optimize your database's performance in your business. The metrics are calculated for the entire server, not limited to the database specified in the JDBC URL.



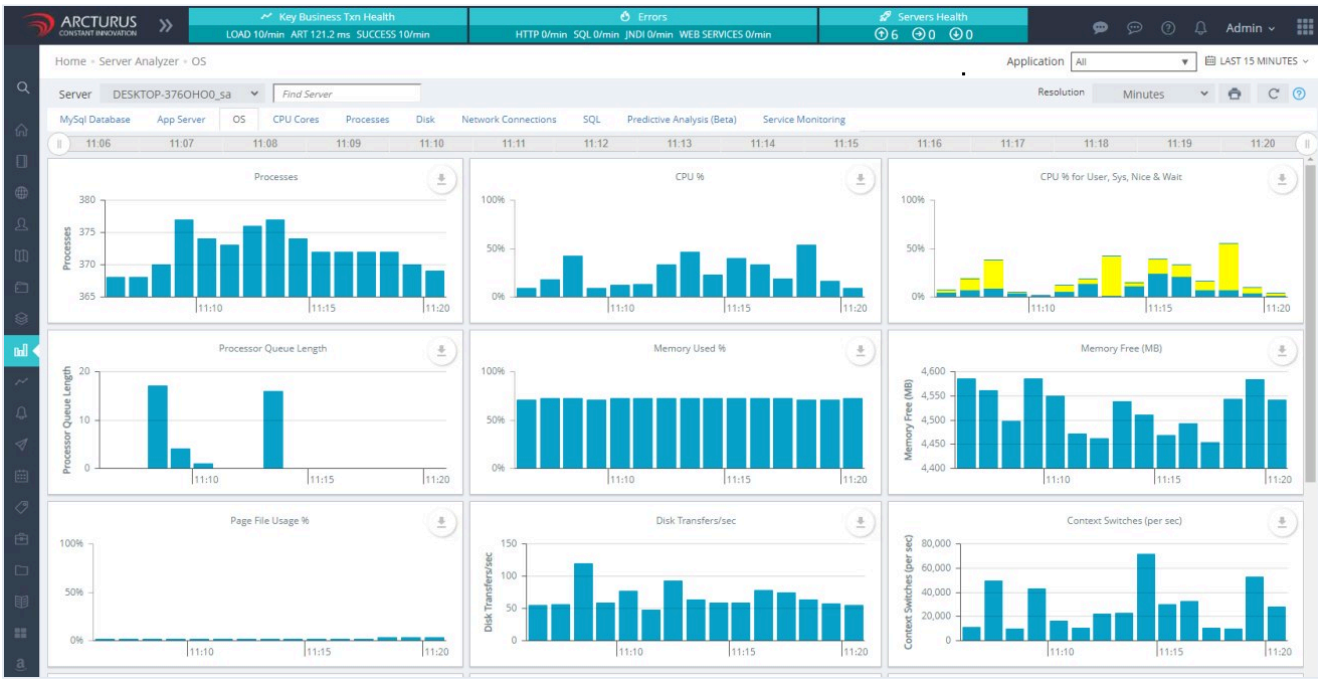
SQL

SQL Statement Profiling provides the performance statistics of database queries. It helps you identify which statements are the most time and quickly diagnose performance issues SQL/s. Additionally, It provides detailed stats, including information on the database, query count, average execution time and various percentiles.



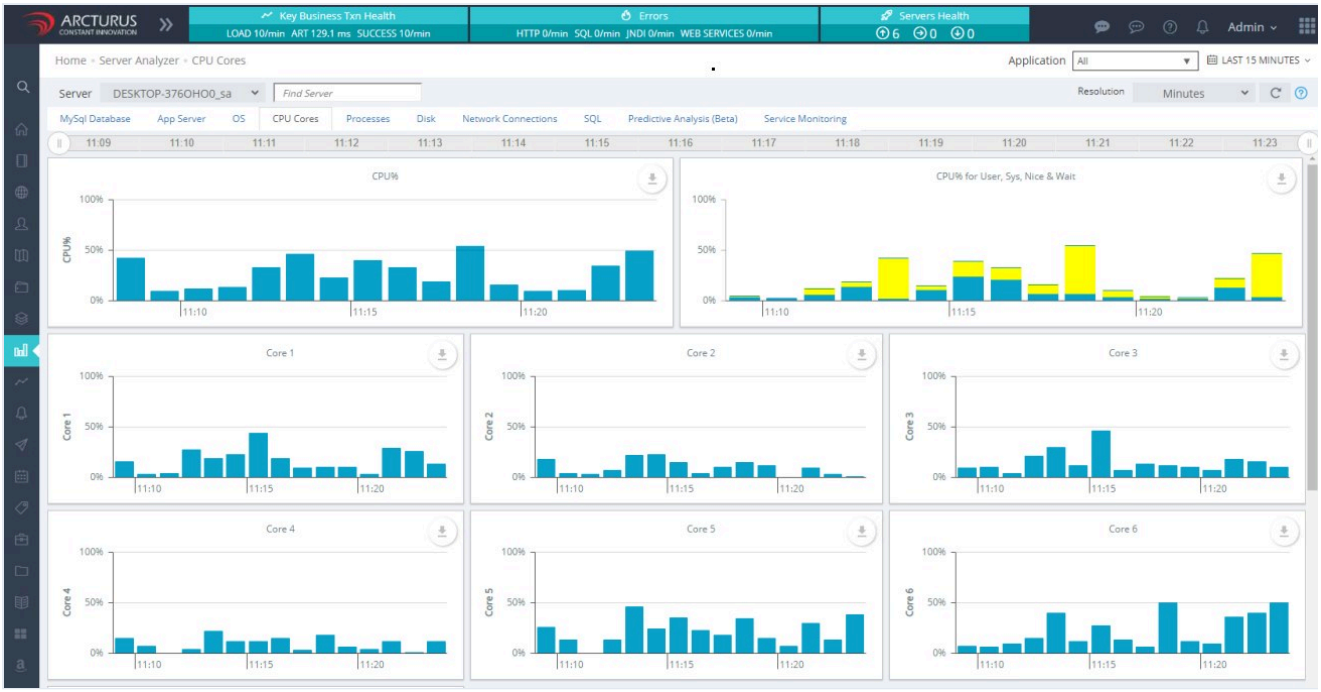
OS

This view offers a quick and convenient overview of your physical server's/VM's performance including details on the number of processes, context switches, CPU Usage, memory, storage performance and more.



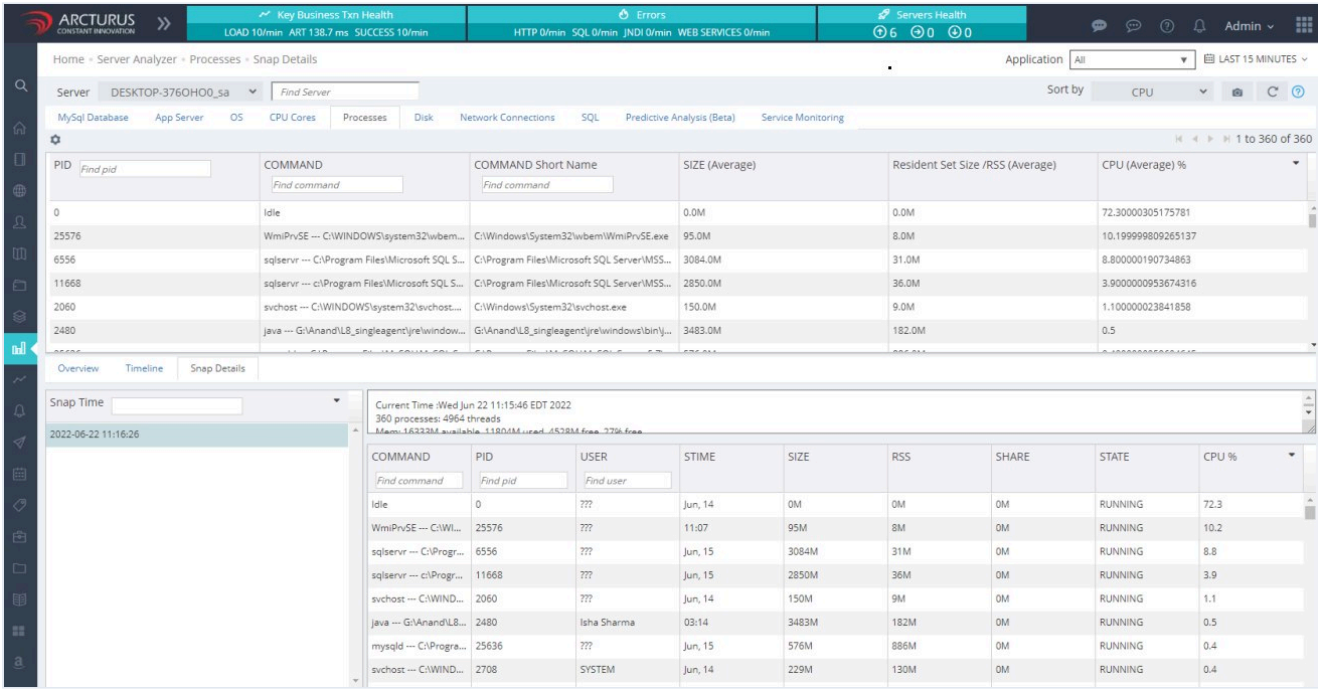
CPU CORE Analyzer

This view presents an overview of the overall CPU as well as detailed information about each individual CPU core.



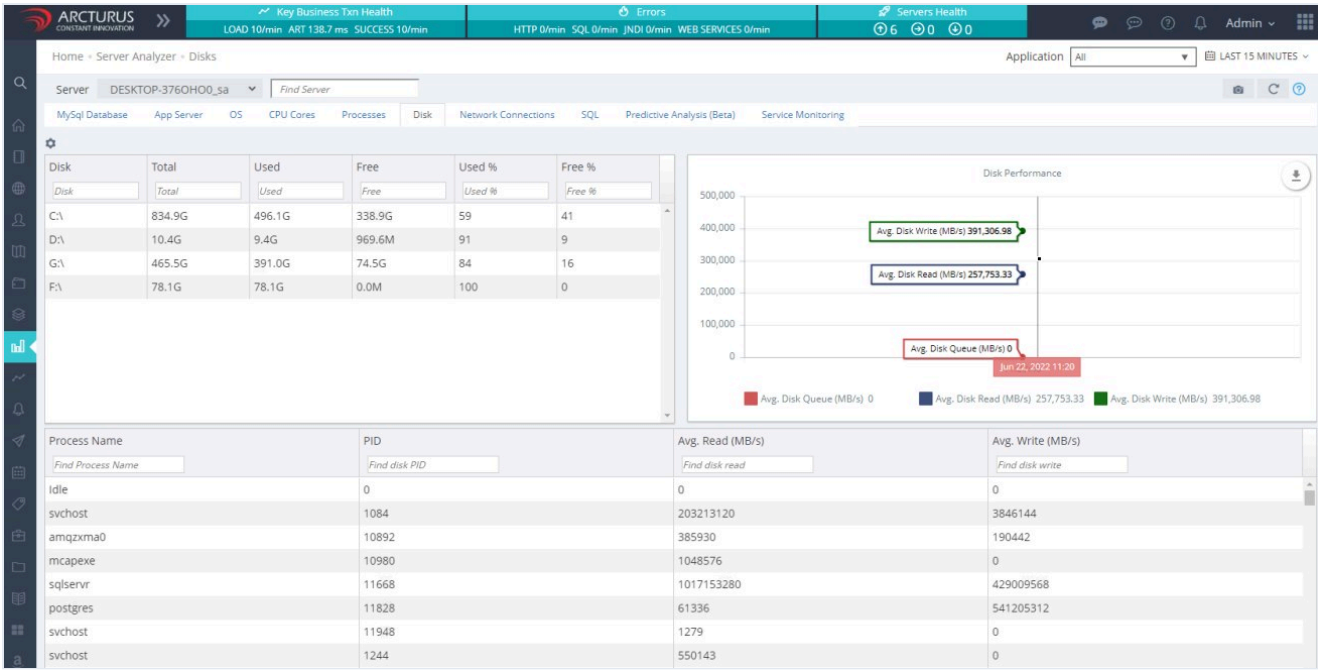
Processes

This view offers organized performance stats for CPU, memory and running processes. It includes historical snapshots of various processes, showing their resources consumption at specific times. By default, Applicare captures these snapshots every 10 minutes but you can adjust by changing the interval as needed.



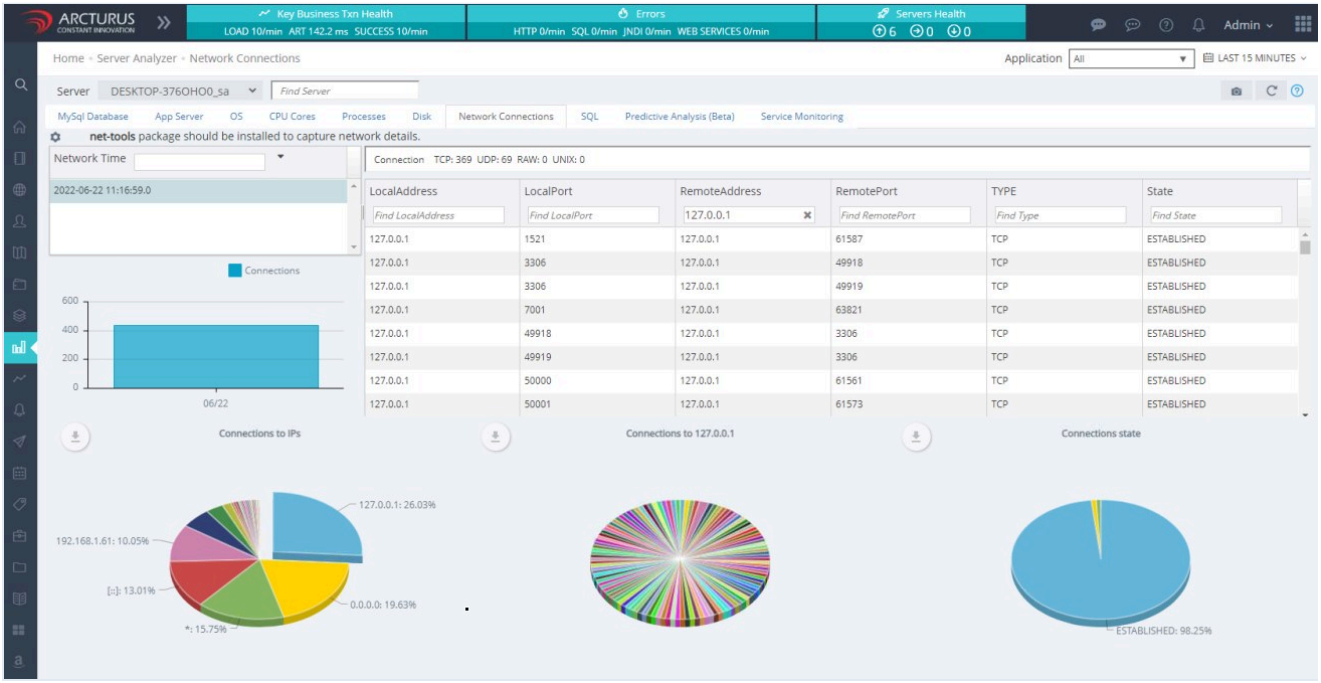
Disk

This view displays the disk usage for each individual process.



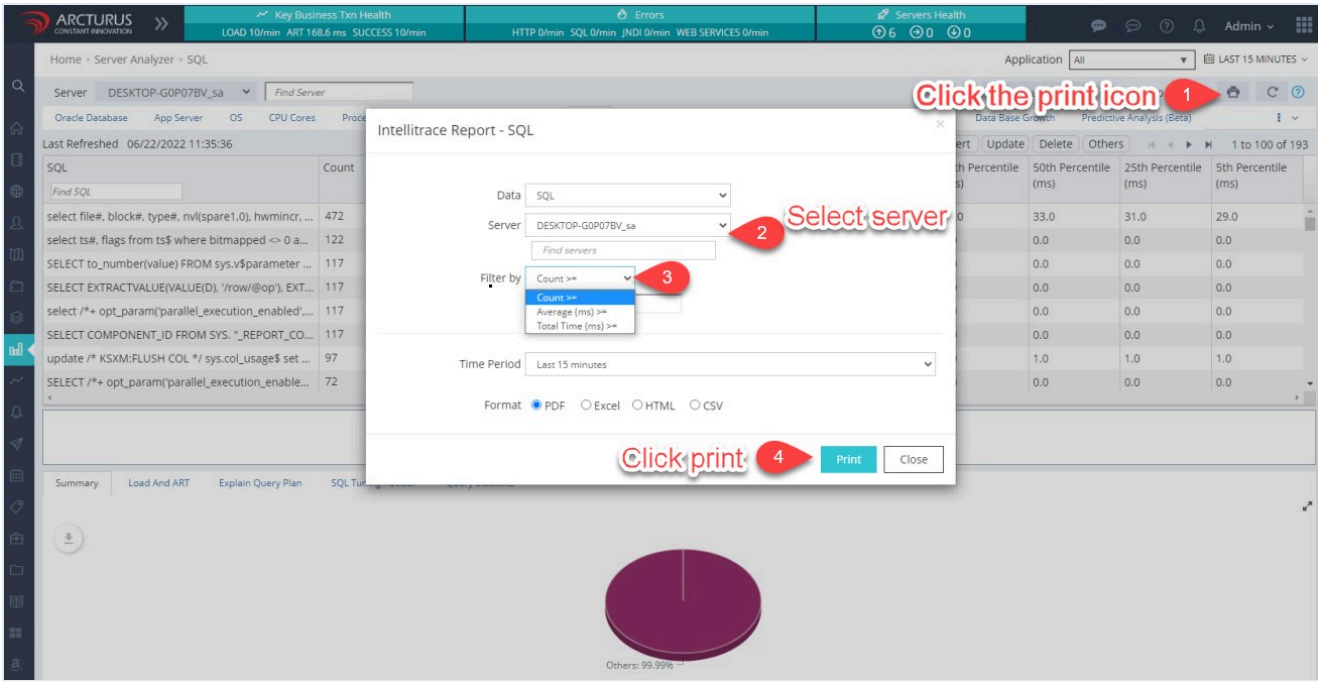
Network Connections

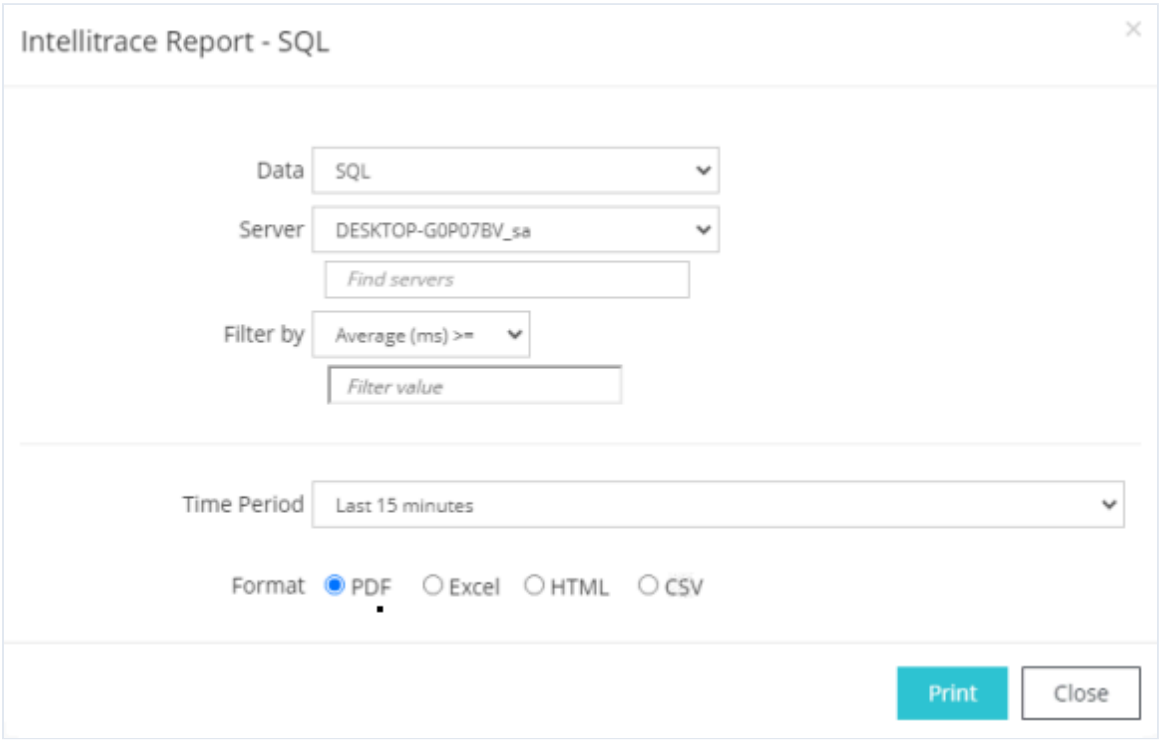
This view offers detailed information on TCP, UDP, RAW, and UNIX connection.

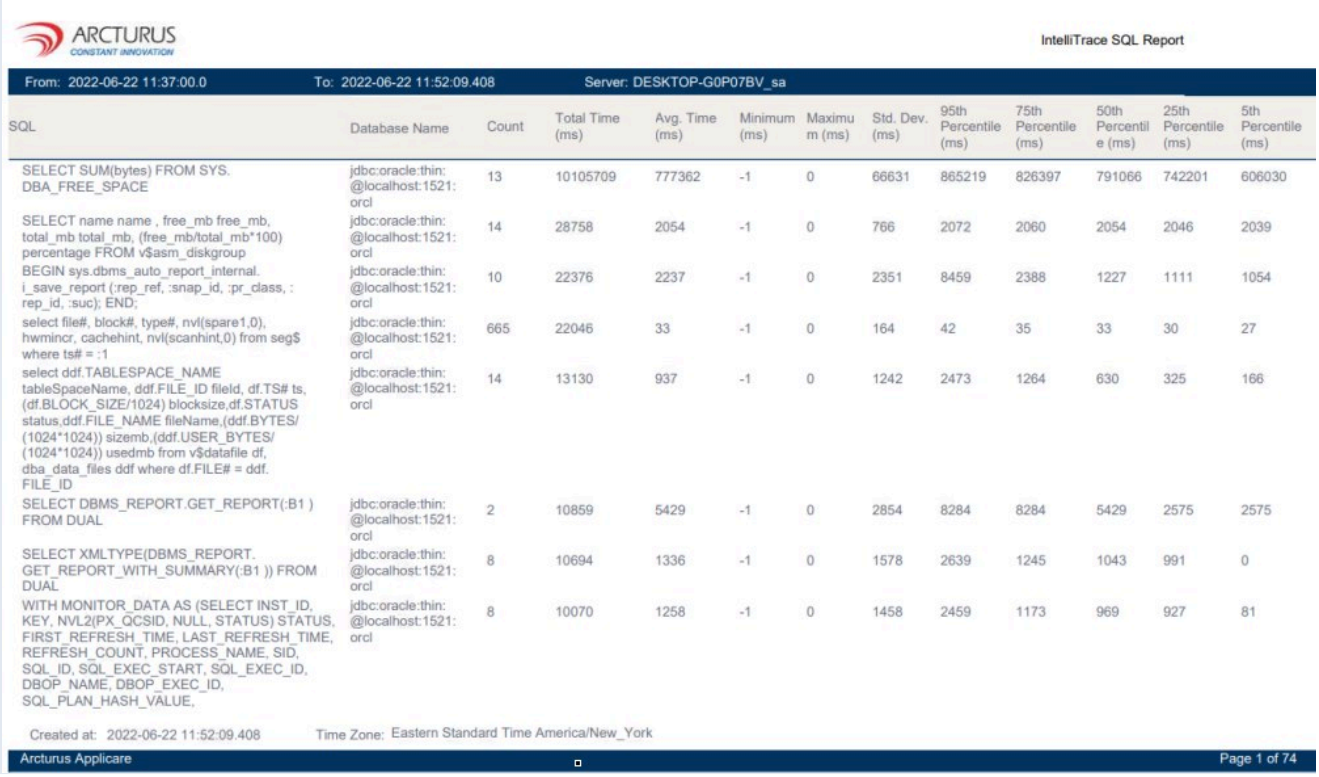


SQL Report

You can download SQL statements as a report by clicking the print icon on the SQL Screen within the Server Analyzer menu.


Click the print icon
1
Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
Select server
2
Find servers
Filter by
Count >=
3
Count >=
Average (ms) >=
Total Time (ms) >=
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Click print
4
Print
Close


Intellitrace Report - SQL
Data
SQL
Server
DESKTOP-G0P07BV_sa
Find servers
Filter by
Average (ms) >=
Filter value
Time Period
Last 15 minutes
Format
PDF
Excel
HTML
CSV
Print
Close



ARCTURUS
CONSTANT INNOVATION

IntelliTrace SQL Report

From: 2022-06-22 11:37:00.0 To: 2022-06-22 11:52:09.408 Server: DESKTOP-G0P07BV_sa

SQL	Database Name	Count	Total Time (ms)	Avg. Time (ms)	Minimum (ms)	Maximum (ms)	Std. Dev. (ms)	95th Percentile (ms)	75th Percentile (ms)	50th Percentile (ms)	25th Percentile (ms)	5th Percentile (ms)
SELECT SUM(bytes) FROM SYS.DBA_FREE_SPACE	jdbc:oracle:thin:@localhost:1521:orcl	13	10105709	777362	-1	0	66631	865219	826397	791066	742201	606030
SELECT name name , free_mb free_mb, total_mb total_mb, (free_mb/total_mb*100) percentage FROM v$asm_diskgroup	jdbc:oracle:thin:@localhost:1521:orcl	14	28758	2054	-1	0	766	2072	2060	2054	2046	2039
BEGIN sys.dbms_auto_report_internal.i_save_report (:rep_ref, :snap_id, :pr_class, :rep_id, :suc); END;	jdbc:oracle:thin:@localhost:1521:orcl	10	22376	2237	-1	0	2351	8459	2388	1227	1111	1054
select file#, block#, type#, nvl(spare1,0), hwmincr, cachehint, nvl(scanhint,0) from seg$ where ts# = :1	jdbc:oracle:thin:@localhost:1521:orcl	665	22046	33	-1	0	164	42	35	33	30	27
select ddf.TABLESPACE_NAME tableSpaceName, ddf.FILE_ID fileId, df.TS# ts, (df.BLOCK_SIZE/1024) blocksize,df.STATUS status,ddf.FILE_NAME fileName,(ddf.BYTES/(1024*1024)) sizemb,(ddf.USER_BYTES/(1024*1024)) usedmb from v$datafile df, dba_data_files ddf where df.FILE# = ddf.FILE_ID	jdbc:oracle:thin:@localhost:1521:orcl	14	13130	937	-1	0	1242	2473	1264	630	325	166
SELECT DBMS_REPORT.GET_REPORT(:B1) FROM DUAL	jdbc:oracle:thin:@localhost:1521:orcl	2	10859	5429	-1	0	2854	8284	8284	5429	2575	2575
SELECT XMLTYPE(DBMS_REPORT.GET_REPORT_WITH_SUMMARY(:B1)) FROM DUAL	jdbc:oracle:thin:@localhost:1521:orcl	8	10694	1336	-1	0	1578	2639	1245	1043	991	0
WITH MONITOR_DATA AS (SELECT INST_ID, KEY, NVL2(PX_QCSID, NULL, STATUS) STATUS, FIRST_REFRESH_TIME, LAST_REFRESH_TIME, REFRESH_COUNT, PROCESS_NAME, SID, SQL_ID, SQL_EXEC_START, SQL_EXEC_ID, DBOP_NAME, DBOP_EXEC_ID, SQL_PLAN_HASH_VALUE,	jdbc:oracle:thin:@localhost:1521:orcl	8	10070	1258	-1	0	1458	2459	1173	969	927	81

Created at: 2022-06-22 11:52:09.408 Time Zone: Eastern Standard Time America/New_York

Arcturus Applicare Page 1 of 74

Configuration

PostgreSQL Database Setup

Monitoring with Applicare

CONFIGURATION GUIDES › DATABASES

MongoDB database monitoring

Applicare 9.2 · Updated September 23, 2025 · mongodb · **Pending technical review**

Create a new user with the **root** role for that new user.

Syntax:

use admin

db.createUser({ user: "**USERNAME**", pwd: "**PASSWORD**", roles: [{ role: "**root**", db: "admin" }] })

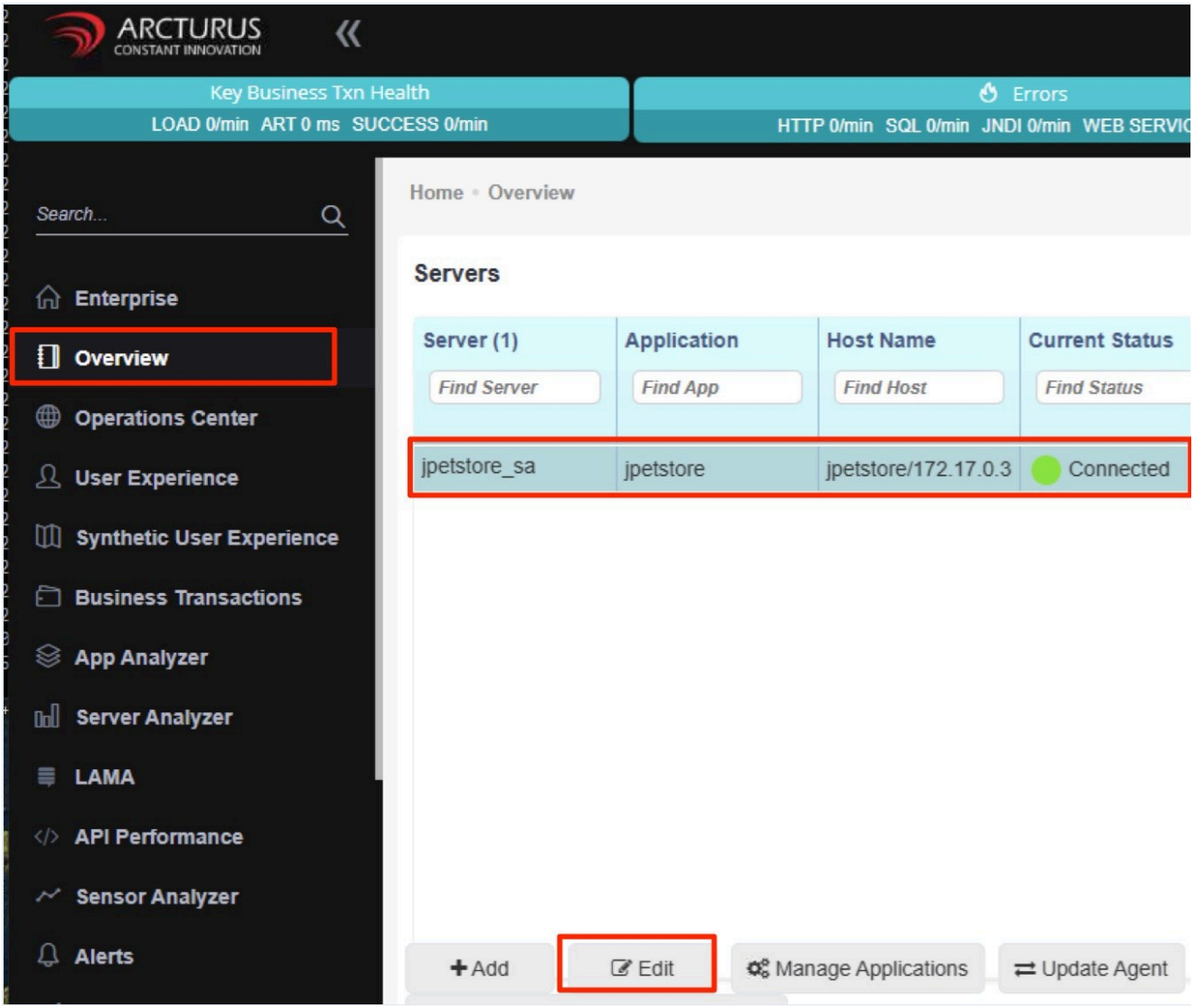
Example:

```yaml
use admin

db.createUser({
user: "applicare",
pwd: "applicare123",
roles: [ { role: "root", db: "admin" } ]
})
```

Step 1: Install Applicare SingleAgent in the machine where **MongoDB** is running by following the provided installation steps.

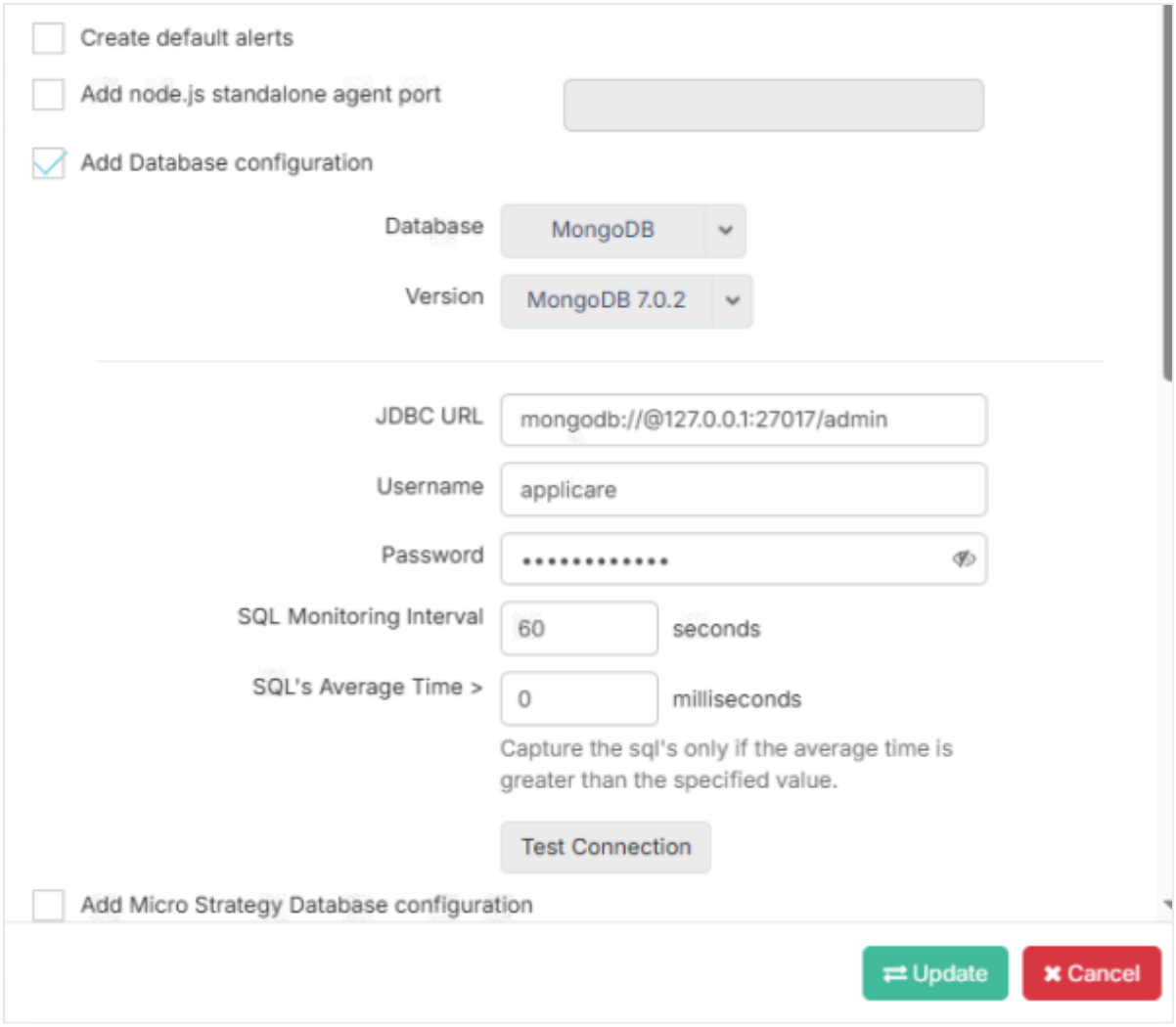
Step 2: Select the newly added agent and click the **Edit** button.



Step 3: Check the "**Add Database Configuration**" checkbox, choose "**MongoDB**" from the **Database** dropdown, and Select your MongoDB server version from the **Version** dropdown. If the version is not listed in the dropdown then select the most recent version available.

Step 4: Enter the **JDBC URL, Username** and **Password** then click the **update** button.

Checking the **Test Connection** option is not mandatory. This test verifies the connection from controller machine to the agent machine.

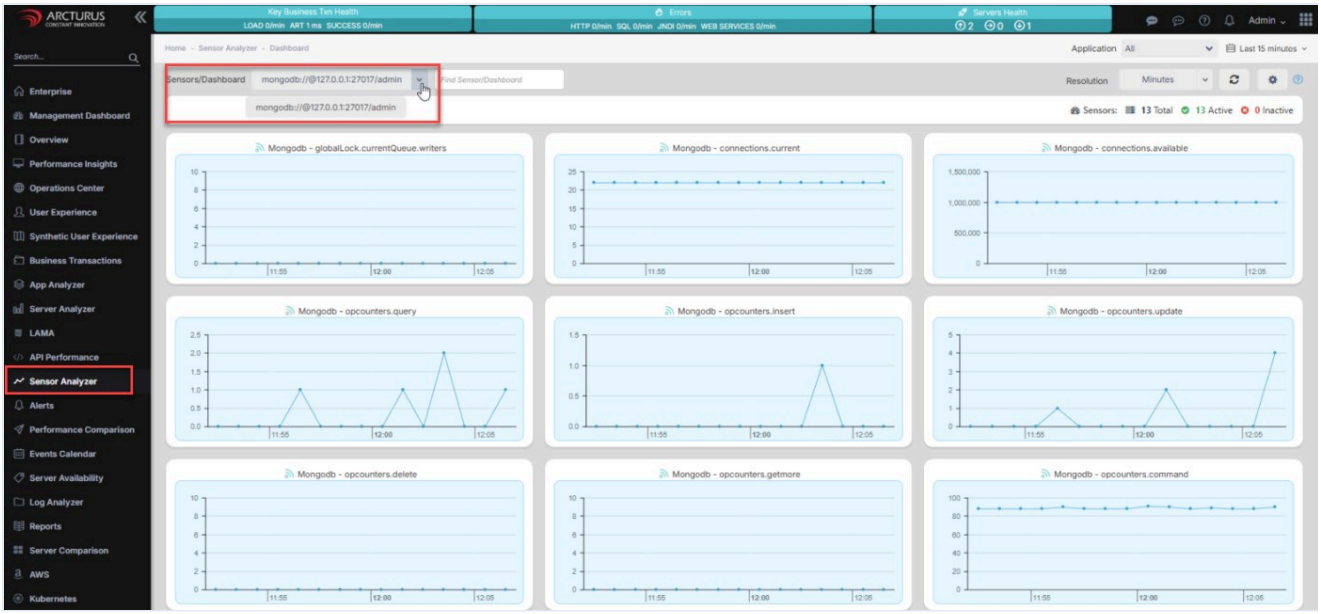


Sample JDBC URL

Syntax: mongodb://@[ip]:[port]/[database]

Eg: mongodb://@127.0.0.1:27017/admin

Step 4: To verify the MongoDB data, go to the **Sensor Analyzer** screen in the left-hand menu and select MongoDB connection string in the **sensor dashboard drop down**.


ARCTURUS
Key Business Txn Health
LOAD 0/min ART 1 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
2 0 1
Admin
Search...
Enterprise
Management Dashboard
Overview
Performance Insights
Operations Center
User Experience
Synthetic User Experience
Business Transactions
App Analyzer
Server Analyzer
LAMA
API Performance
Sensor Analyzer
Alerts
Performance Comparison
Events Calendar
Server Availability
Log Analyzer
Reports
Server Comparison
AWS
Kubernetes
Home - Sensor Analyzer - Dashboard
Application All
Last 15 minutes
Sensors/Dashboard
mongodb://@127.0.0.1:27017/admin
Find Sensor/Dashboard
Resolution
Minutes
Sensors: 13 Total 13 Active 0 Inactive
Mongodb - globalLock.currentQueue.writers
Mongodb - connections.current
Mongodb - connections.available
Mongodb - opcounters.query
Mongodb - opcounters.insert
Mongodb - opcounters.update
Mongodb - opcounters.delete
Mongodb - opcounters.getmore
Mongodb - opcounters.command

CONFIGURATION GUIDES › DATABASES

Redis Monitoring

Applicare 9.2 · Updated December 24, 2025 · redis · **Pending technical review**

Create a new user account and designate the relevant permissions in Redis.

Step 1: Open the **redis.conf** file, Add the following line to the file to create a **read-only** user, and save it.

```
TEXT
user <readonly_user> on ><readonly_password> ~* +@read +info +slowlog
```

Substitute `<readonly_user>` and `<readonly_password>` with the newly username and password you have just created.

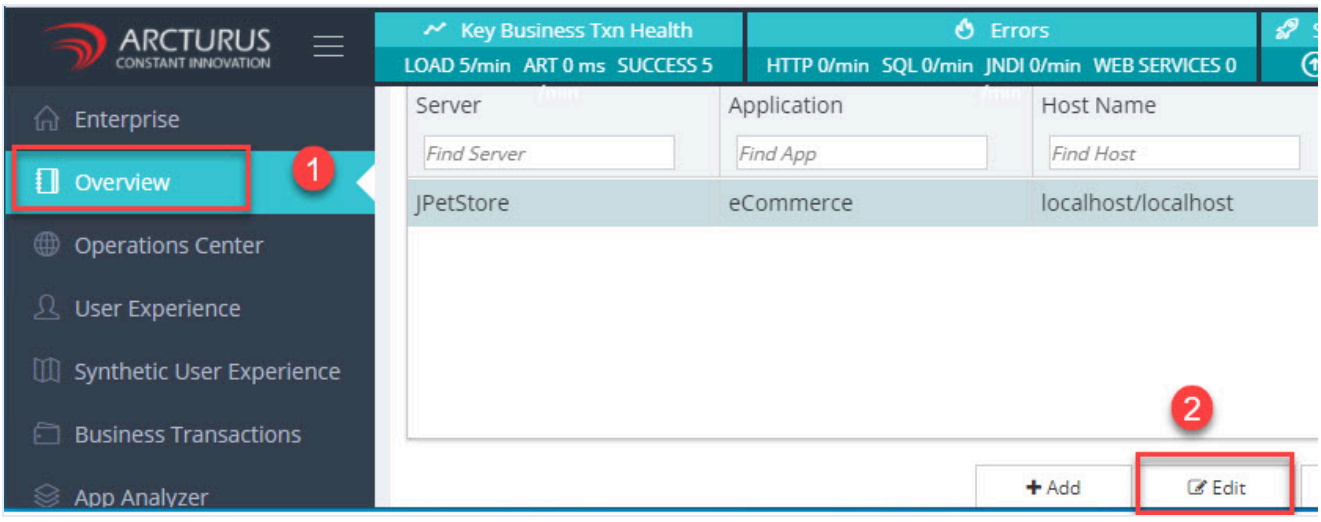
Example: If the newly created username is **appuser** and the password is **appuserpass**, use these details in place of `<readonly_user>` and `<readonly_password>`."

```
TEXT
user appuser on >appuserpass ~* +@read +info +slowlog
```

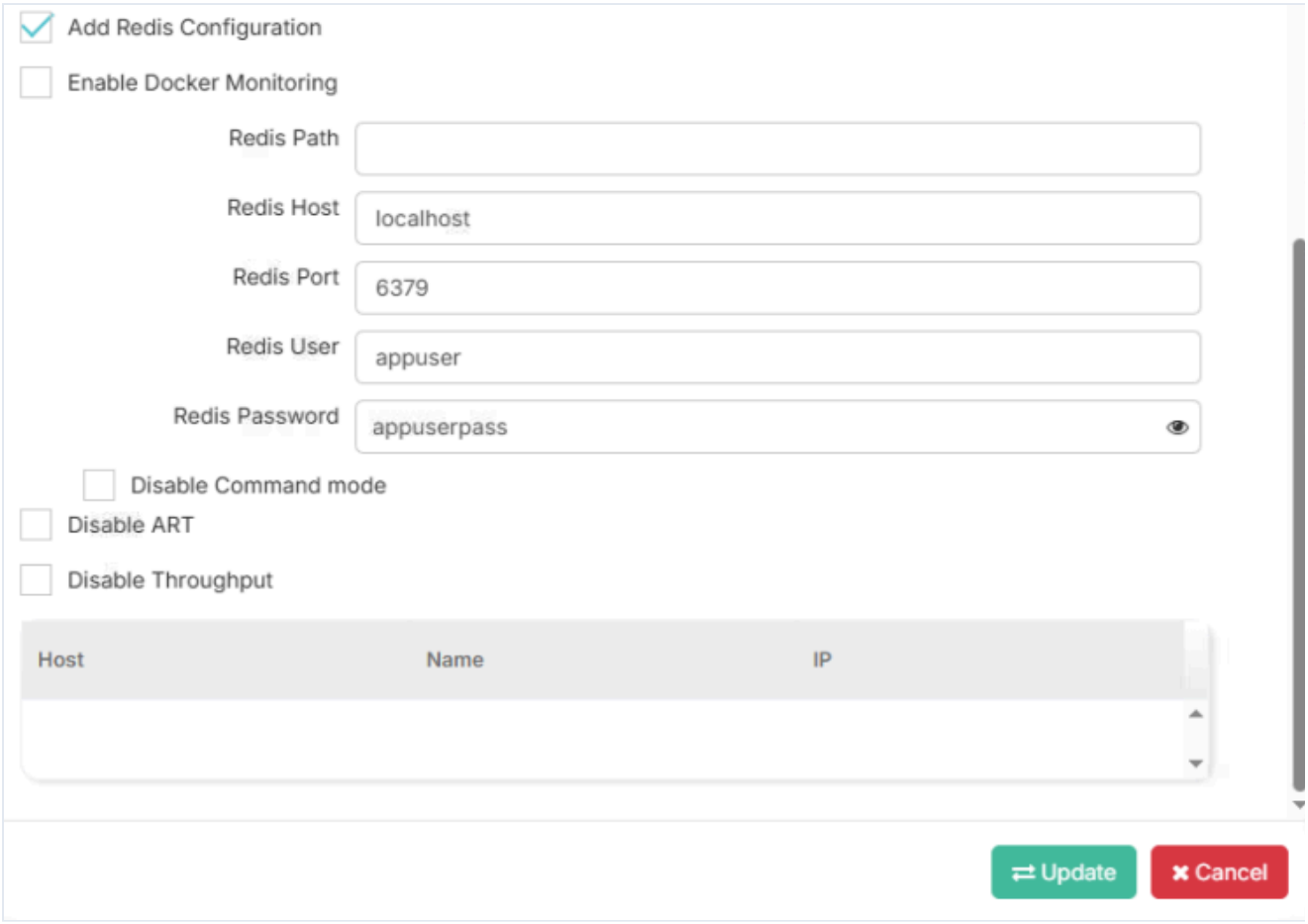
Step 2: Once the file is saved, **Restart Redis**.

Step 3: Install Applicare SingleAgent in the machine where **Redis database** is running by following the provided installation steps. Follow the steps for **Installing Applicare SingleAgent**

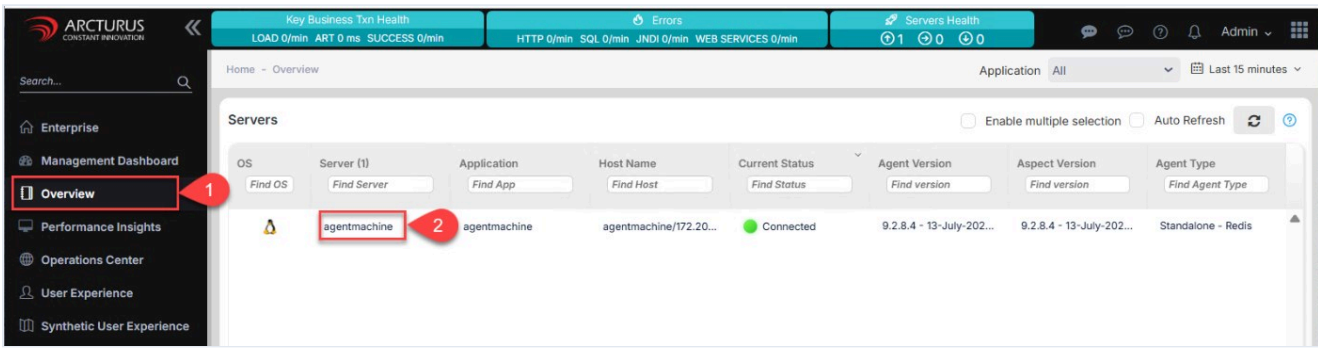
Step 4: Select the newly added agent and click the **Edit** button.

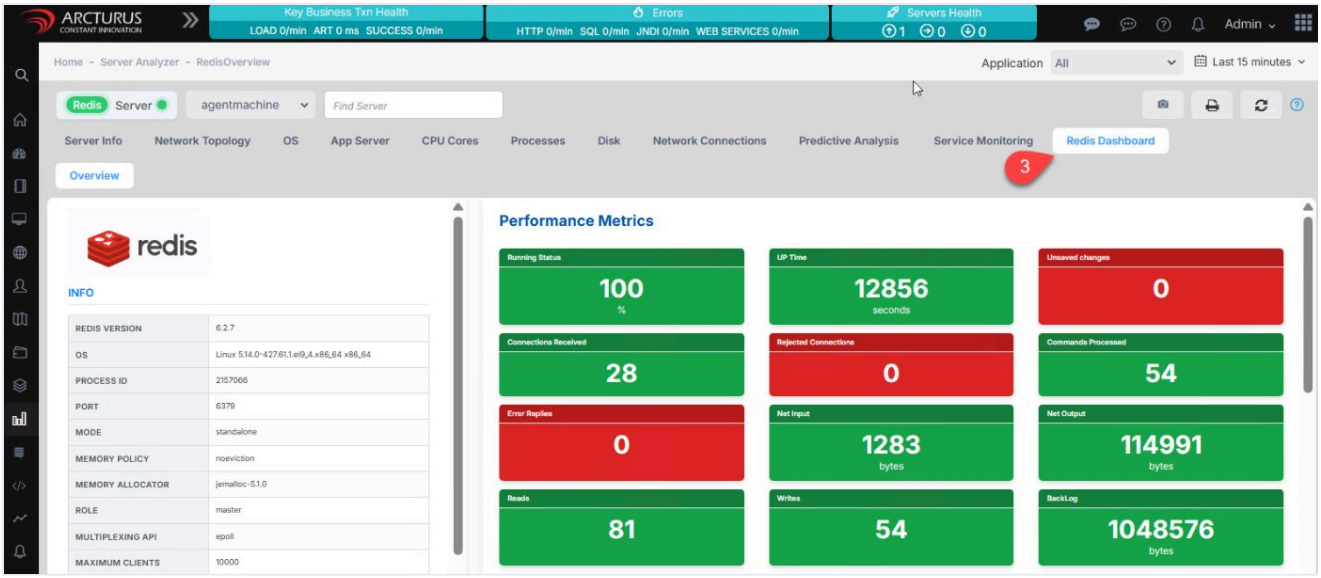


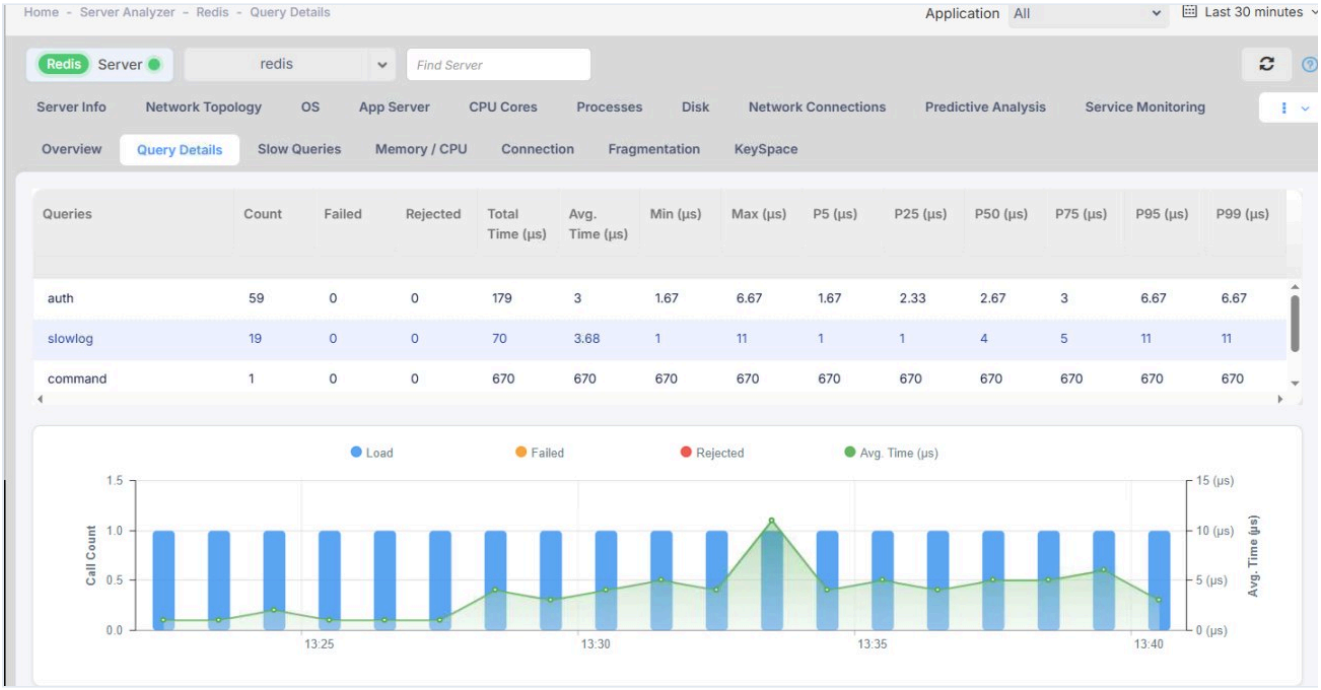
Step 5: Enable the **Add Redis configuration** checkbox. Then enter the **Host**, **Port**, **User** and **Password** then click the **update** button.



Step 5: After updating the configuration, to view the **Redis** data, click on the Redis configured server in the 'Server' column on the '**Overview**' screen. Then, select the '**Redis Dashboard**' tab.




ARCTURUS
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
Admin
Home - Server Analyzer - RedisOverview
Application All
Last 15 minutes
Redis Server
agentmachine
Find Server
Server Info
Network Topology
OS
App Server
CPU Cores
Processes
Disk
Network Connections
Predictive Analysis
Service Monitoring
Redis Dashboard
Overview
3
redis
INFO
REDIS VERSION 6.2.7
OS Linux 5.14.0-427.61.1.el9_4.x86_64 x86_64
PROCESS ID 2157066
PORT 6379
MODE standalone
MEMORY POLICY noeviction
MEMORY ALLOCATOR jemalloc-5.1.0
ROLE master
MULTIPLEXING API epoll
MAXIMUM CLIENTS 10000
Performance Metrics
Running Status 100 %
UP Time 12856 seconds
Unsaved changes 0
Connections Received 28
Rejected Connections 0
Commands Processed 54
Error Replies 0
Net Input 1283 bytes
Net Output 114991 bytes
Reads 81
Writes 54
BackLog 1048576 bytes


Home - Server Analyzer - Redis - Query Details
Application All
Last 30 minutes
Redis Server
redis
Find Server
Server Info
Network Topology
OS
App Server
CPU Cores
Processes
Disk
Network Connections
Predictive Analysis
Service Monitoring
Overview
Query Details
Slow Queries
Memory / CPU
Connection
Fragmentation
KeySpace
Queries Count Failed Rejected Total Time (µs) Avg. Time (µs) Min (µs) Max (µs) P5 (µs) P25 (µs) P50 (µs) P75 (µs) P95 (µs) P99 (µs)
auth 59 0 0 179 3 1.67 6.67 1.67 2.33 2.67 3 6.67 6.67
slowlog 19 0 0 70 3.68 1 11 1 1 4 5 11 11
command 1 0 0 670 670 670 670 670 670 670 670 670 670
Load
Failed
Rejected
Avg. Time (µs)
Call Count
Avg. Time (µs)
1.5
1.0
0.5
0.0
15 (µs)
10 (µs)
5 (µs)
0 (µs)
13:25
13:30
13:35
13:40

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Node.JS Monitoring for Linux

Applicare 9.2 · Updated February 28, 2025 · nodejs · **Pending technical review**

Prerequisites

To monitor Node.js Applications, You need to configure both Node.js and Single agent. Ensure that both are running on the same machine:

- Single Agent
- Node.js Agent

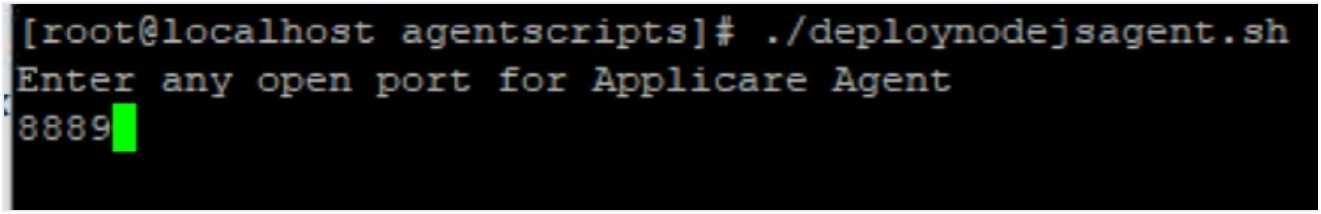
Monitoring

1. Login into the **Applicare console,** go to the **Download Agents** screen, and download the **SingleAgentLinux** for Linux. Then, transfer the download file to the machine where the agent will be deployed.

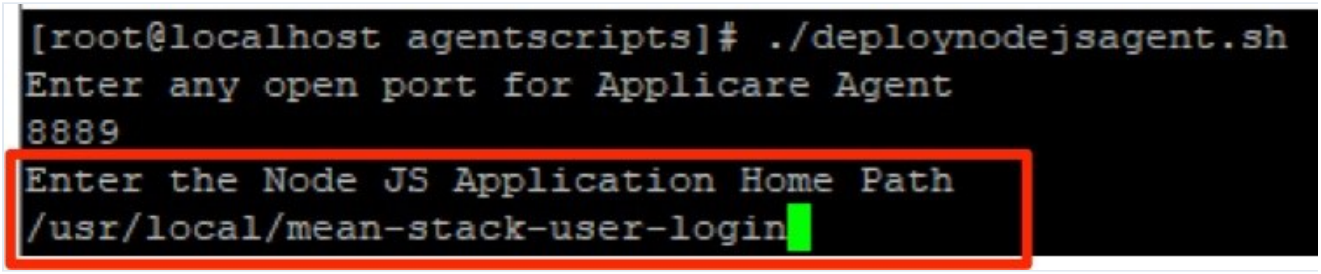
2. Extract the contents of the downloaded file and give the below permission for that extracted directory.

```bash
chmod -R 777 /usr/local/ApplicareSingleAgentLinux
```

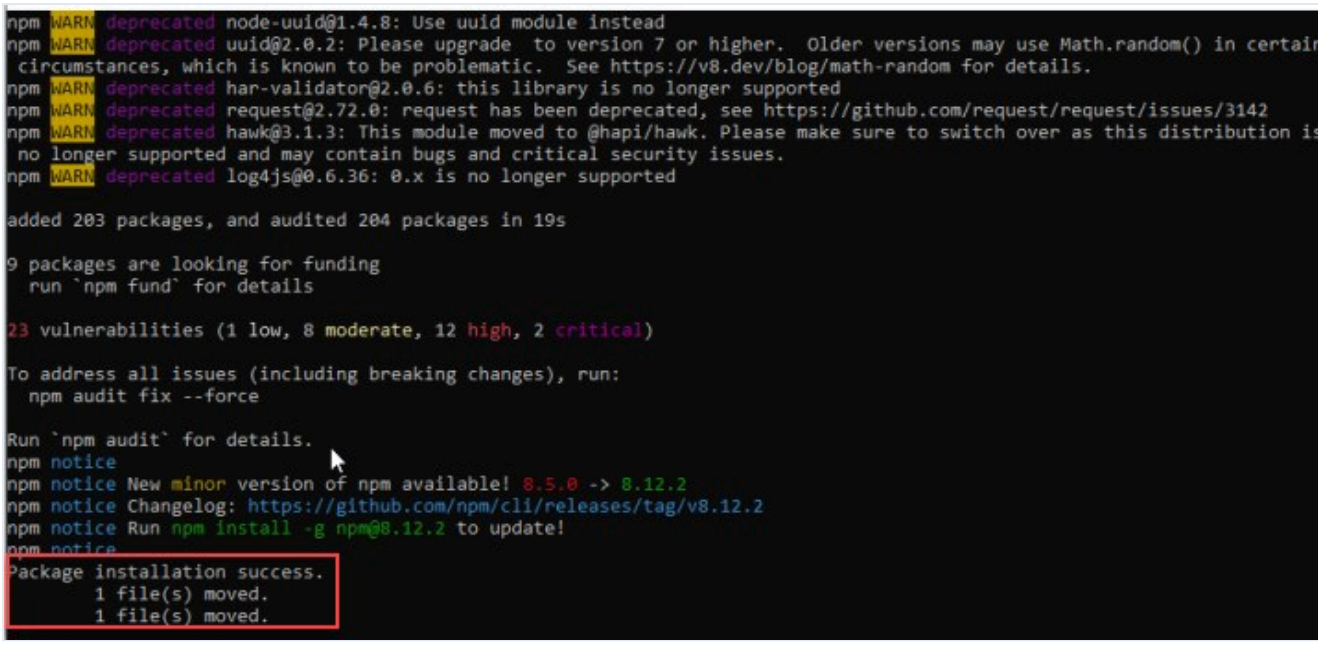
3a. Execute the **deploynodejsagent.sh** inside the **agentscritpts** extracted directory. After executing, the message "**Enter any open port for Applicare Agent**" will appear in the terminal window and enter any **open port** in it.



3b. Enter the **Node.js home path** where the application is running.



3c. Once the agent is **successfully installed,** You will see a message similar to the one shown below



4. Navigate to the Node.js application home directory and insert the following code at the beginning of the main file (usually called **server.js** or **app.js**).

```
TEXT
var applicareConfig = require('./agent.config.js');var applicare = require('applicare-agent')(applicareConfig);
```

```
require('rootpath')();

var applicareConfig = require('./agent.config.js');

var applicare = require('applicare-agent')(applicareConfig);

var express = require('express');
var app = express();
var session = require('express-session');
var bodyParser = require('body-parser');
var expressJwt = require('express-jwt');
```

5. Restart the **Node.js** server.

6. If **systemctl** is available on agent deployed Linux machine, execute **installNodeJsAgentService.sh** file in **agentscripts** directory.

```
TEXT
To start as a service: execute ./installNodeJsAgentService.sh
```

If systemctl is not there on agent deployed machine, To start in background mode execute below command in agentscripts directory on agent home directory.

```text
nohup ./startNodeJsAgent.sh &
```

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Manual Instrumentation for Node.js (Docker image)

Applicare 9.2 · Updated April 1, 2025 · nodejs · **Pending technical review**

Step 1: Download the tracing.js *(sample file, available on the documentation site)* **file to the root directory.**

Step 2: Add the following **dependencies** to the **Dockerfile**.

```
TEXT
# Install dependencies
RUN npm install --save @opentelemetry/api
RUN npm install --save @opentelemetry/instrumentation
RUN npm install --save @opentelemetry/tracing
RUN npm install --save @opentelemetry/exporter-trace-otlp-http
RUN npm install --save @opentelemetry/resources
RUN npm install --save @opentelemetry/semantic-conventions
RUN npm install --save @opentelemetry/auto-instrumentations-node
RUN npm install --save @opentelemetry/sdk-node
RUN npm install --save @opentelemetry/exporter-jaeger
```

Step 3: Add the following **environment variable** line in Dockerfile.

```
TEXT
ENV SERVER_NAME=<Server_name>
ENV APPLICARE_ENDPOINT=http://<Applicare_IP>:14268/api/traces
```

Change the value in that variable as below.

<Server_name>= The name of the agent shown in the Applicare console.

<Applicare_IP>= The IP address of the machine running the Applicare controller.

Step 4: Comment the CMD line already in the **Dockerfile**. Add below **CMD** line.

```
TEXT
CMD ["node", "--require", "./tracing.js", "app.js (or) server.js (or) node.js"]
```

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Node.JS Monitoring for Windows

Applicare 9.2 · Updated December 18, 2025 · nodejs, windows · **Pending technical review**

Prerequisites

To monitor Node.js Applications, You need to configure both Node.js and Single agent. Ensure that both are running on the same machine:

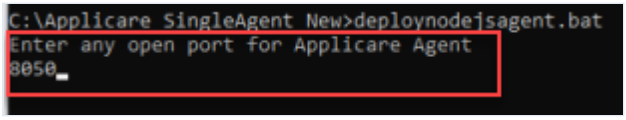
- Single Agent
- Node.js Agent

Monitoring

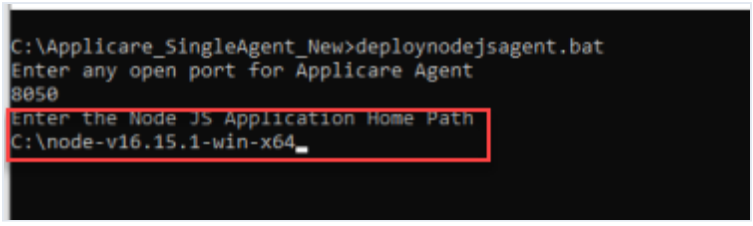
1: Install Applicare SingleAgent on the machine running **Tomcat** by following these steps

installing Applicare SingleAgent

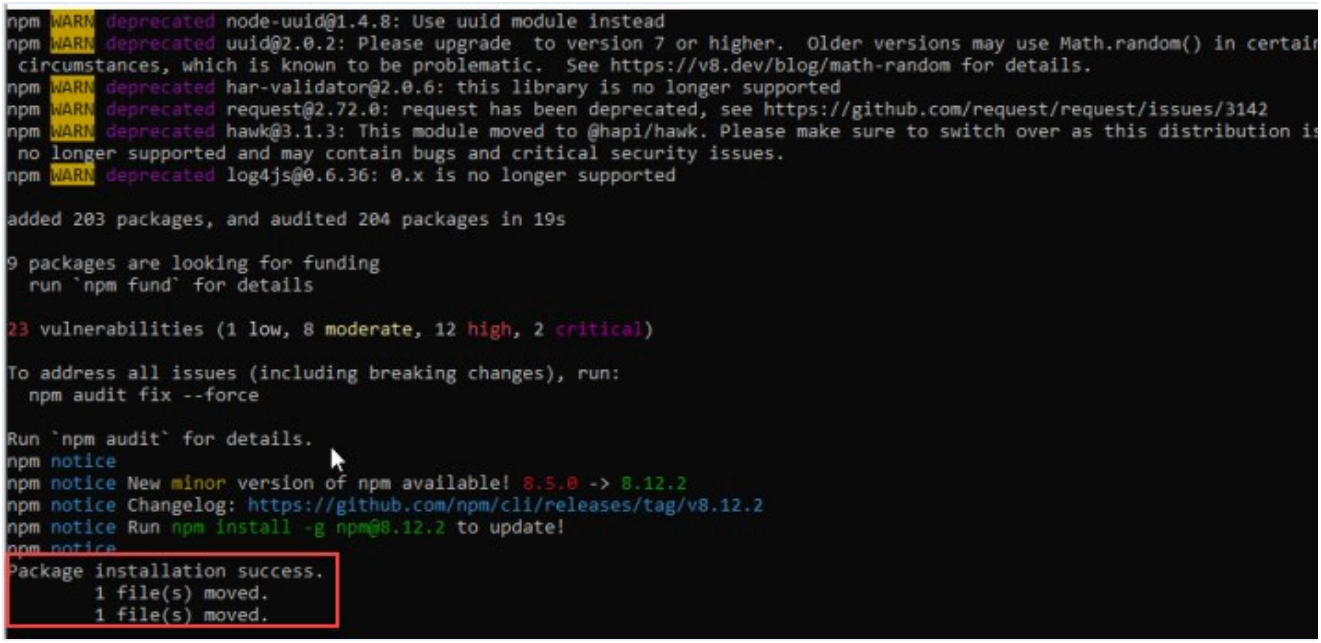
2 a. Run `deploynodejsagent.bat` with elevated (Administrator) privileges from the directory **agentscripts** inside Applicare SingleAgent installed folder. Once executing started , the message "**Enter any open port for Applicare Agent**" will appear in the command window and enter any **open port** in the machine.



b. Enter the **Node.js home path** where your nodejs application is running.



c. Once the agent is **successfully installed,** You will see a message similar to the one shown below.



3. **Zero code instrumentation** : Execute your nodejs application with "**-r ./agent-injector.js**" to inject Applicare SingleAgent in your application. For example if you start your application with a command "node wss_server.js", then you should invoke it as "`node -r ./agent-injector.js wss_server.js`"

   (OR)

Manual instrumentation : Navigate to the Node.js application home directory and insert the following code at the beginning of the main file (usually called **server.js** or **app.js**). Then, restart the **Node.js** server.

```
TEXT
var applicareConfig = require('./agent.config.js');
var applicare = require('applicare-agent')(applicareConfig);
```

4. Go to the **SingleAgent** home directory and execute the command based on the OS.

5.Open a command window in **administrator** mode, go to the Agent_home/agentscripts directory and execute the file below.

```
BAT
createNodejsAgentTask.bat
```

For Other Docs - Click here

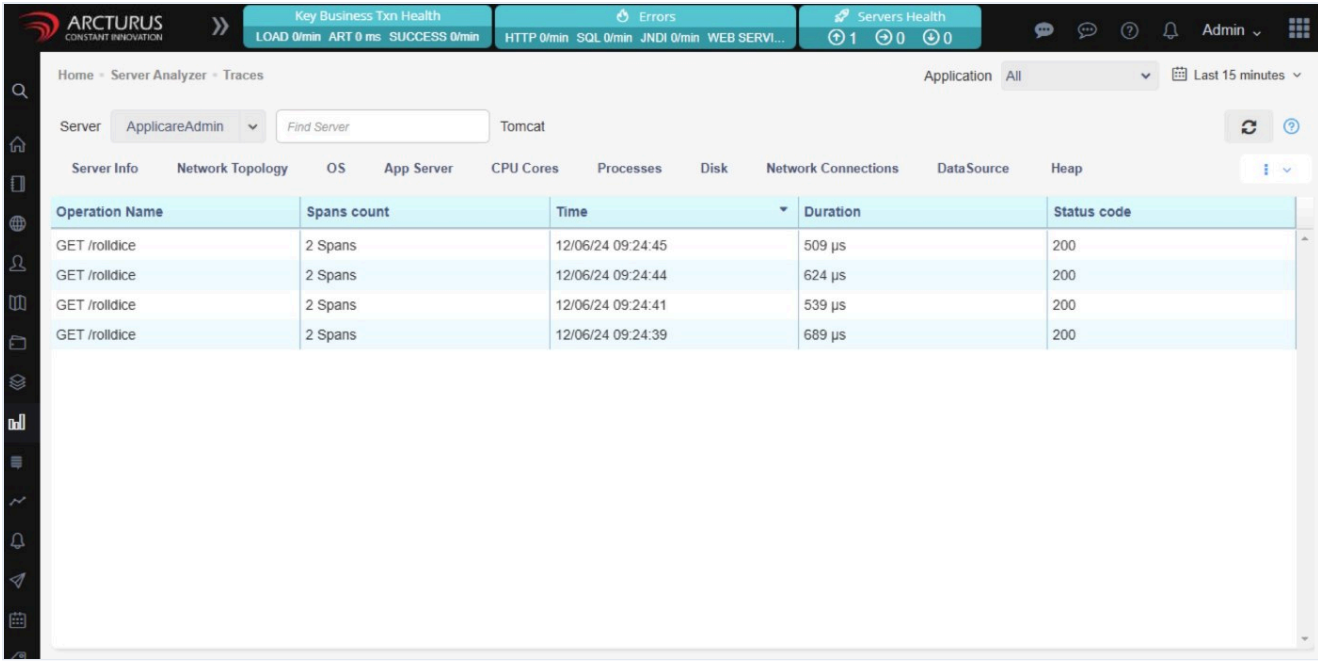
CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

PHP Application Monitoring

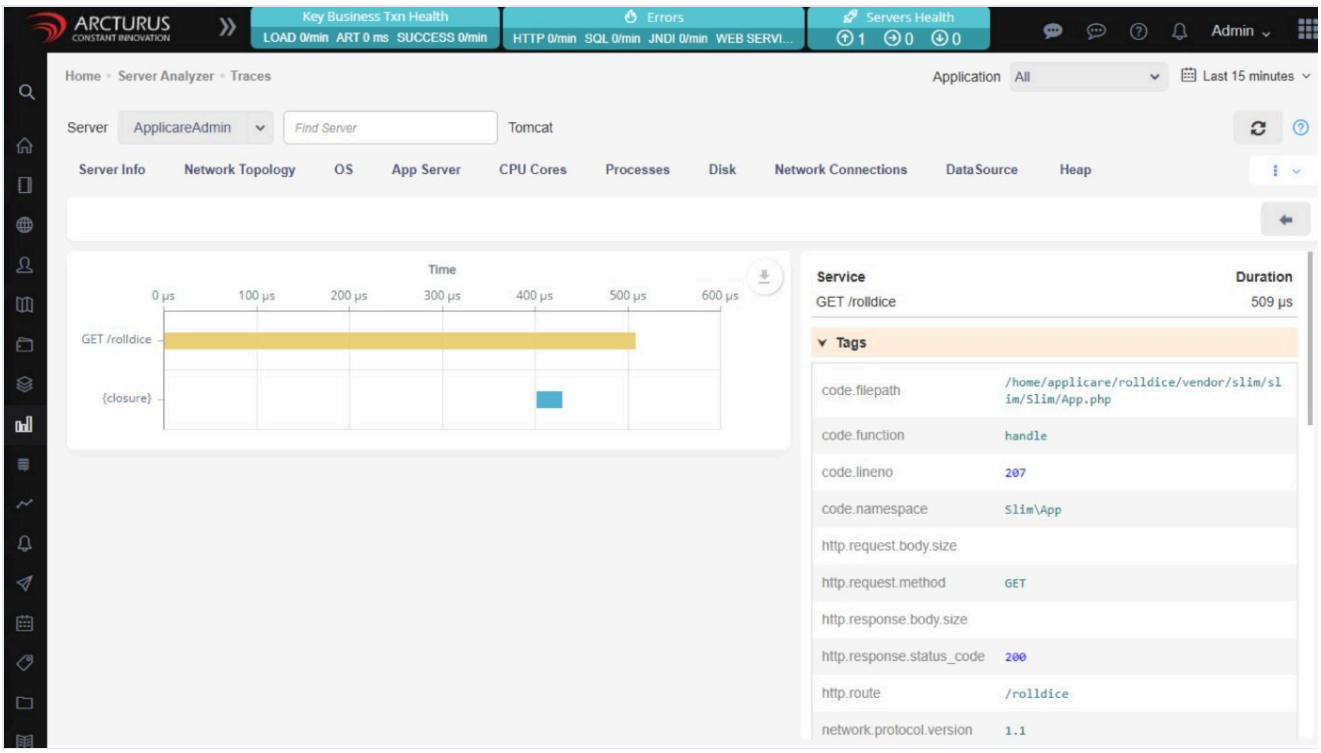
Applicare 9.2 · Updated April 1, 2025 · php · **Pending technical review**

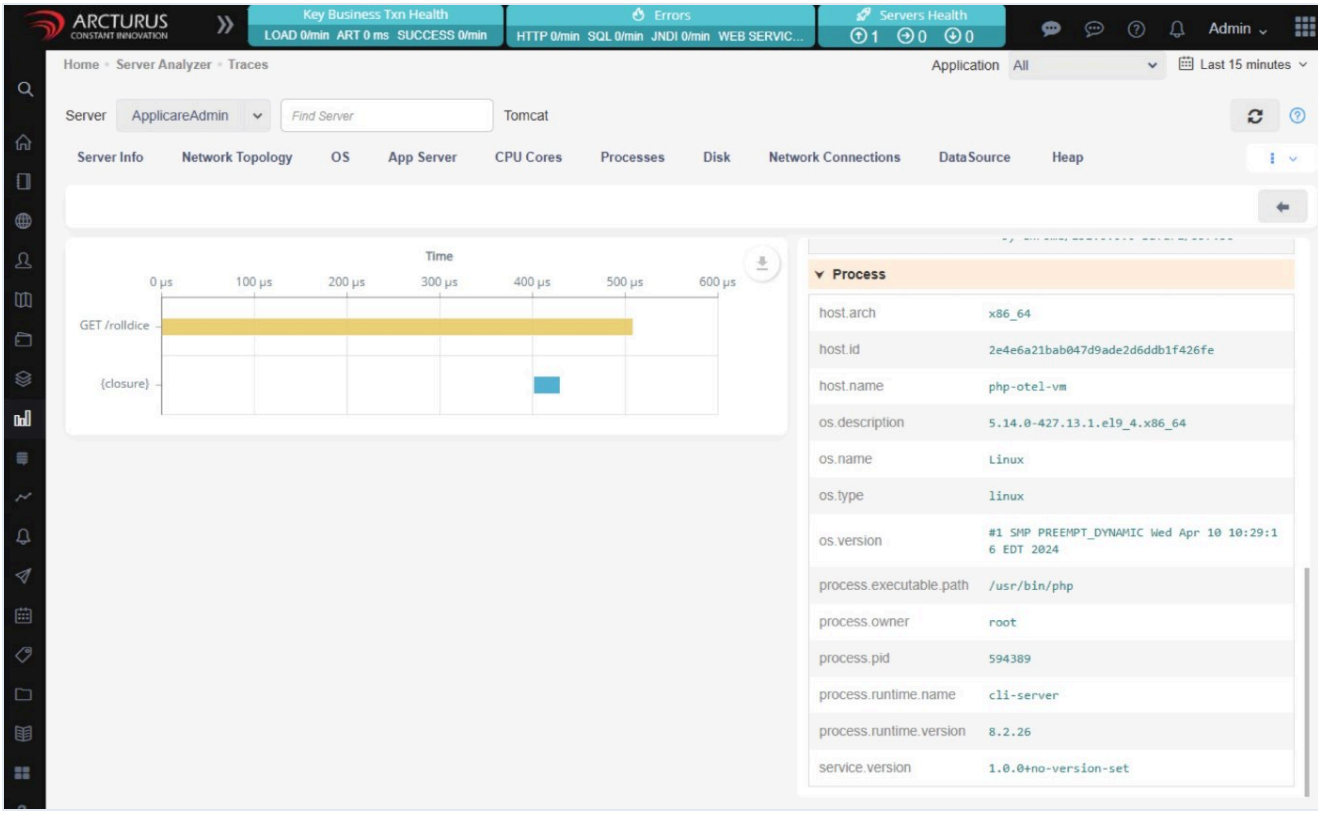
Log in to Applicare controller and navigate to the Server Analyzer->Telemetry Traces menu to view the details of PHP application transaction.

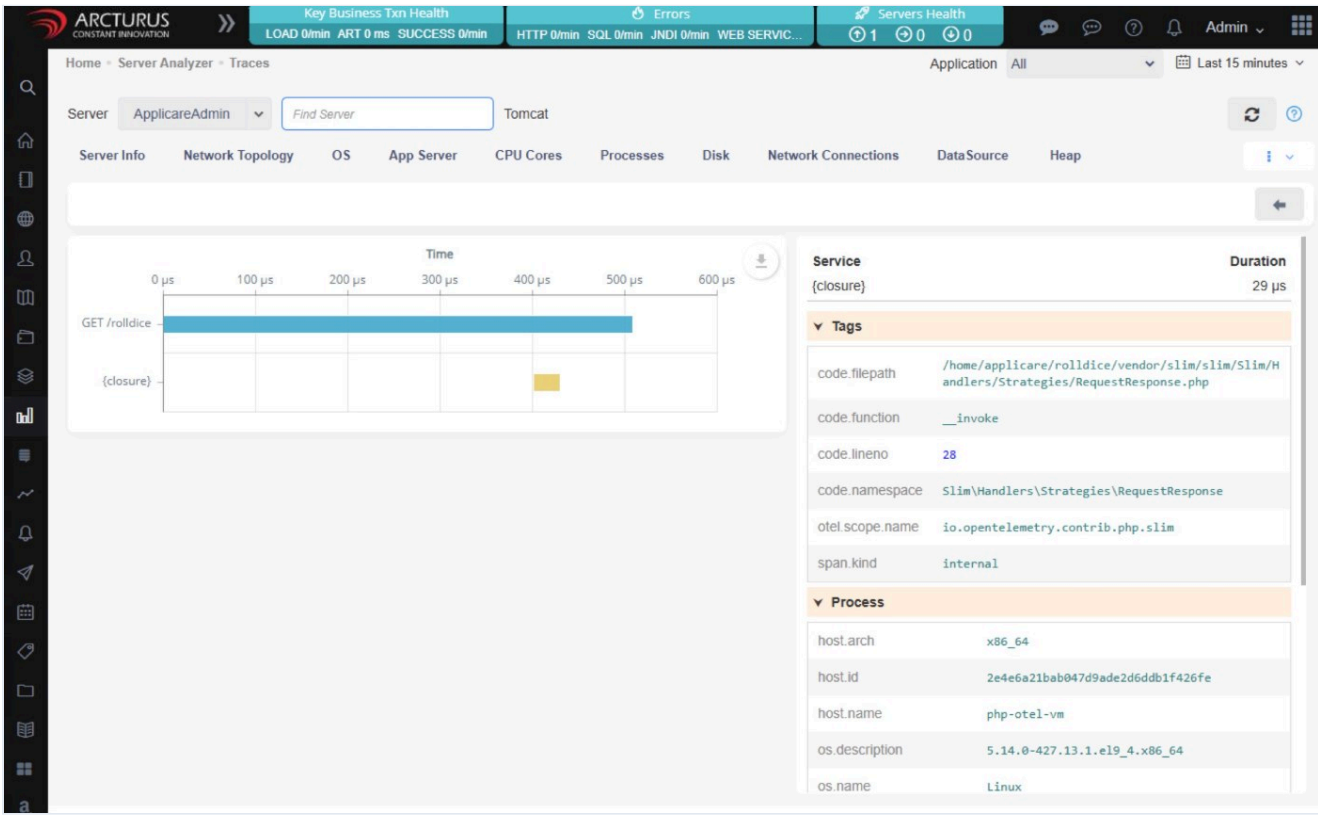
1. The screen displays a list of PHP applications transactions, including their durations and status codes.



2. After clicking on any transaction, it will show the transaction trace details.







Configuration

Follow the configuration instructions provided in the link below to monitor your PHP application.

Configure PHP application

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Configure PHP application

Applicare 9.2 · Updated April 1, 2025 · php · **Pending technical review**

To automatically instrument a PHP application, use PHP latest version 8.2 or later is required.

Prerequisites: PHP 8.2 or later, Composer, and PHP Extension Community Library (PECL) are installed.

Step 1: Execute below command to build the OpenTelemetry PHP extension.

```bash
yum install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -E %rhel).rpm
yum install php-cli php-pear php-devel gcc php-grpc -y
yum install gcc make autoconf
```

Step 2: Use PECL to build the OpenTelemetry PHP extension.

```text
pecl install opentelemetry
```

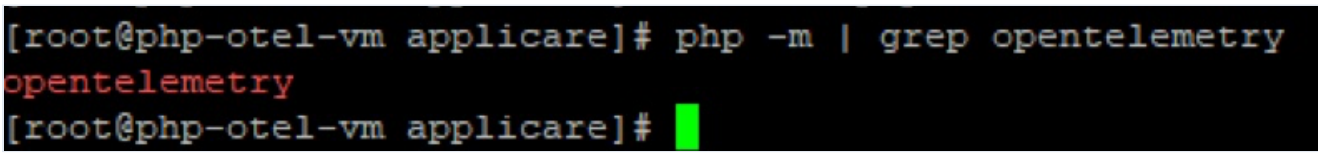
Step 3: Add the following code to the php.ini file.

```text
[opentelemetry]
extension=opentelemetry.so
```

Step 4: Execute below command to verify whether the OpenTelemetry PHP extension is built and enabled.

```text
php -m | grep opentelemetry

Expected output: opentelemetry
```



Step 5: Exceute below command to add additional dependencies required for OpenTelemetry SDK for PHP to perform automatic instrumentation on your application.

```
TEXT
composer config allow-plugins.php-http/discovery false

composer require \
open-telemetry/sdk \
open-telemetry/opentelemetry-auto-slim \
open-telemetry/exporter-otlp \
php-http/guzzle7-adapter \
open-telemetry/transport-grpc
```

Step 6: Run the application with below command.

```
TEXT
env OTEL_PHP_AUTOLOAD_ENABLED=true OTEL_SERVICE_NAME=<Server-Name> OTEL_TRACES_EXPORTER=otlp
OTEL_METRICS_EXPORTER=none OTEL_LOGS_EXPORTER=none OTEL_EXPORTER_OTLP_PROTOCOL=grpc
OTEL_EXPORTER_OTLP_ENDPOINT=http://<Applicare_IP>:4317
```

Change the value in that variable as below.

<Server_name>= The name of the agent shown in the Applicare console.

<Applicare_IP>= The IP address of the machine running the Applicare controller.

Example: env OTEL_PHP_AUTOLOAD_ENABLED=true OTEL_SERVICE_NAME=**rolldice** OTEL_TRACES_EXPORTER=otlp OTEL_METRICS_EXPORTER=none OTEL_LOGS_EXPORTER=none OTEL_EXPORTER_OTLP_PROTOCOL=grpc OTEL_EXPORTER_OTLP_ENDPOINT=http://**192.168.2.6**:4317 php -S 10.0.0.4:8080

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Manually instrument a Go application using OpenTelemetry and Applicare

Applicare 9.2 · Updated August 22, 2024 · go, opentelemetry · **Pending technical review**

Install the following packages:

```text
go get "go.opentelemetry.io/otel" \
  "go.opentelemetry.io/otel/exporters/stdout/stdoutmetric" \
  "go.opentelemetry.io/otel/exporters/stdout/stdouttrace" \
  "go.opentelemetry.io/otel/exporters/stdout/stdoutlog" \
  "go.opentelemetry.io/otel/sdk/log" \
  "go.opentelemetry.io/otel/log/global" \
  "go.opentelemetry.io/otel/propagation" \
  "go.opentelemetry.io/otel/sdk/metric" \
  "go.opentelemetry.io/otel/sdk/resource" \
  "go.opentelemetry.io/otel/sdk/trace" \
  "go.opentelemetry.io/otel/semconv/v1.24.0" \
  "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"\
  "go.opentelemetry.io/contrib/bridges/otelslog"
```

This installs the OpenTelemetry SDK components and `net/http` instrumentation.

Add the following packages to your Go application:

```
TEXT
import (
          "context"
          "fmt"
          "net"
          "os"
          "os/signal"
        "time"
          "net/http"
          "google.golang.org/grpc"
          "google.golang.org/grpc/credentials/insecure"
          "go.opentelemetry.io/otel"

          "go.opentelemetry.io/otel/exporters/otlp/otlptrace/otlptracegrpc"
          "go.opentelemetry.io/otel/propagation"
          "go.opentelemetry.io/otel/sdk/resource"

         semconv "go.opentelemetry.io/otel/semconv/v1.25.0"
sdktrace "go.opentelemetry.io/otel/sdk/trace"
          "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
)
```

You need to wrap the HTTP handlers with instrumented code.

YAML

```
var serviceName = semconv.ServiceNameKey.String(<APPLICARE_AGENT_NAME>)

func initConn() (*grpc.ClientConn, error) {
    conn, err := grpc.NewClient("localhost:4317",
        grpc.WithTransportCredentials(insecure.NewCredentials()),
    )
    if err != nil {
        return nil, fmt.Errorf("failed to create gRPC connection to collector: %w", err)
    }
    return conn, err
}

func initTracerProvider(ctx context.Context, res *resource.Resource, conn *grpc.ClientConn)
(func(context.Context) error, error) {
 traceExporter, err := otlptracegrpc.New(ctx, otlptracegrpc.WithGRPCConn(conn))
    if err != nil {
        return nil, fmt.Errorf("failed to create trace exporter: %w", err)
    }
    bsp := sdktrace.NewBatchSpanProcessor(traceExporter)
    tracerProvider := sdktrace.NewTracerProvider(
        sdktrace.WithSampler(sdktrace.AlwaysSample()),
        sdktrace.WithResource(res),
        sdktrace.WithSpanProcessor(bsp),
    )
    otel.SetTracerProvider(tracerProvider)
    otel.SetTextMapPropagator(propagation.TraceContext{})
    return tracerProvider.Shutdown, nil
}

func main() {
    if err := run(); err != nil {
        log.Fatalln(err)
    }
}

func run() (err error) {
    ctx, stop := signal.NotifyContext(context.Background(), os.Interrupt)
    defer stop()

    conn, err := initConn()

    if err != nil {
        log.Fatal(err)
    }
```

```
    res, err := resource.New(ctx,
        resource.WithAttributes(
            serviceName,
        ),
    )

    if err != nil {
        log.Fatal(err)
    }

    shutdownTracerProvider, err := initTracerProvider(ctx, res, conn)

    if err != nil {
        log.Fatal(err)
    }

    defer func() {
        if err := shutdownTracerProvider(ctx); err != nil {
            log.Fatalf("failed to shutdown TracerProvider: %s", err)
        }
    }()

// Start HTTP server.
    srv := &http.Server{
        Addr:         ":8080",
        BaseContext:  func(_ net.Listener) context.Context { return ctx },
        ReadTimeout:  time.Second,
        WriteTimeout: 10 * time.Second,
        Handler:      newHTTPHandler(),
    }

    srvErr := make(chan error, 1)
        go func() {
        srvErr <- srv.ListenAndServe()
    }()

    select {
        case err = <-srvErr:
        return
        case <-ctx.Done():
        stop()
    }

    err = srv.Shutdown(context.Background())
    return
```

```
}

   func newHTTPHandler() http.Handler {
       mux := http.NewServeMux()
       handleFunc := func(pattern string, handlerFunc func(http.ResponseWriter,
*http.Request)) {
        handler := otelhttp.WithRouteTag(pattern, http.HandlerFunc(handlerFunc))
        mux.Handle(pattern, handler)
    }

   // your handler function here ..........
   // ...........

    handler := otelhttp.NewHandler(mux, "/")
    return handler
}
```

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

Automatically instrument a Go Application running in Docker with OpenTelemetry and Applicare (With additional container)

Applicare 9.2 · Updated August 22, 2024 · go, opentelemetry · **Pending technical review**

A Go executable can be instrumented without modifying the original code of the application by configuring environment variables and running the instrumentation with elevated privileges.

1. To instrument an application automatically, you need the following:

a. Linux with kernel version 4.19 or newer

b. x64 or ARM processor

c. Docker image or compiled binary for OpenTelemetry Go Automatic Instrumentation

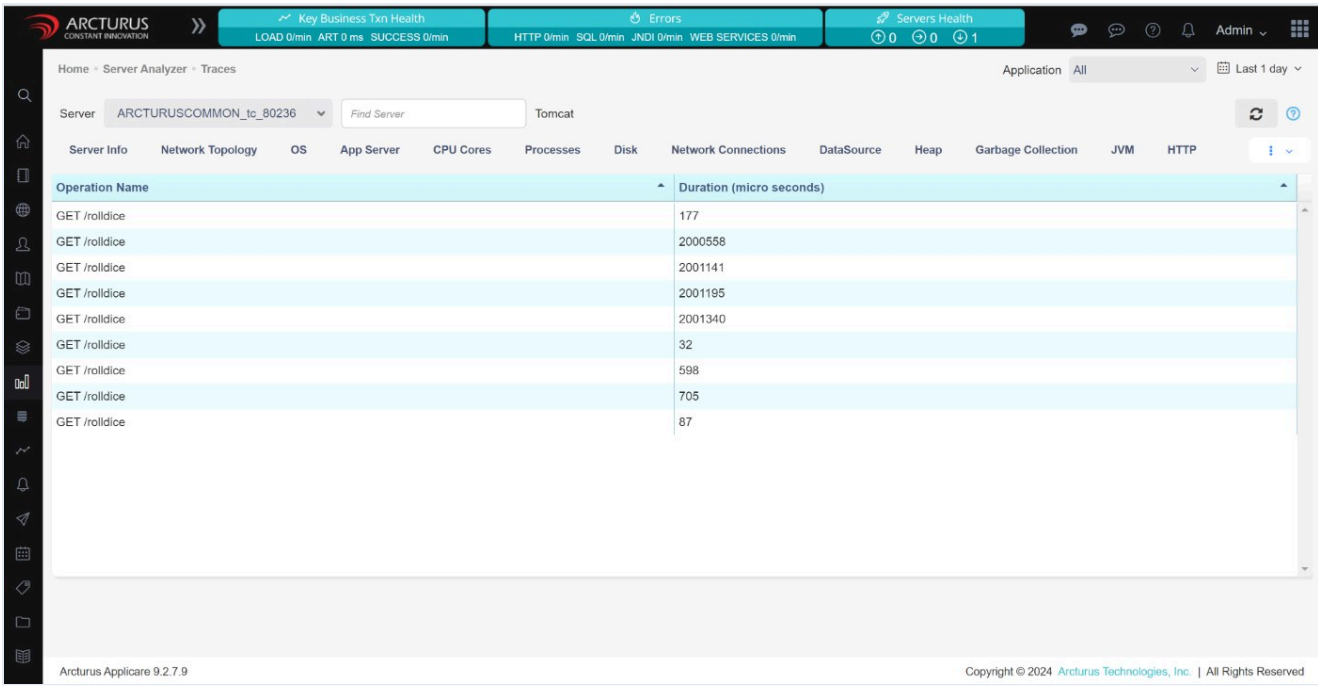
d. Go 1.18 or newer

2. Edit the docker-compose file to add a new service for the instrumentation.

```yaml
  go-auto:
    image: otel/autoinstrumentation-go
    privileged: true
    pid: "host"
    environment:
      - OTEL_EXPORTER_OTLP_ENDPOINT=http://<Applicare_Controller_IP>:4318
      - OTEL_GO_AUTO_TARGET_EXE=<location_of_target_application_binary>
      - OTEL_SERVICE_NAME=<Applicare_Agent_Name>
      - OTEL_PROPAGATORS=tracecontext,baggage
    volumes:
      - <shared_volume_of_application>
      - /proc:/host/proc
```

3. Use the command `docker compose up` to build and start the Docker containers.

4. The collected traces will be shown in Server Analyzer -> Telemetry Traces screen in Applicare console.


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
0 0 1
Admin
Home • Server Analyzer • Traces
Application All
Last 1 day
Server ARCTURUSCOMMON_tc_80236
Find Server
Tomcat
Server Info
Network Topology
OS
App Server
CPU Cores
Processes
Disk
Network Connections
DataSource
Heap
Garbage Collection
JVM
HTTP
Operation Name
Duration (micro seconds)
GET /rolldice 177
GET /rolldice 2000558
GET /rolldice 2001141
GET /rolldice 2001195
GET /rolldice 2001340
GET /rolldice 32
GET /rolldice 598
GET /rolldice 705
GET /rolldice 87
Arcturus Applicare 9.2.7.9
Copyright © 2024 Arcturus Technologies, Inc. | All Rights Reserved

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

COBOL Monitoring with Applicare

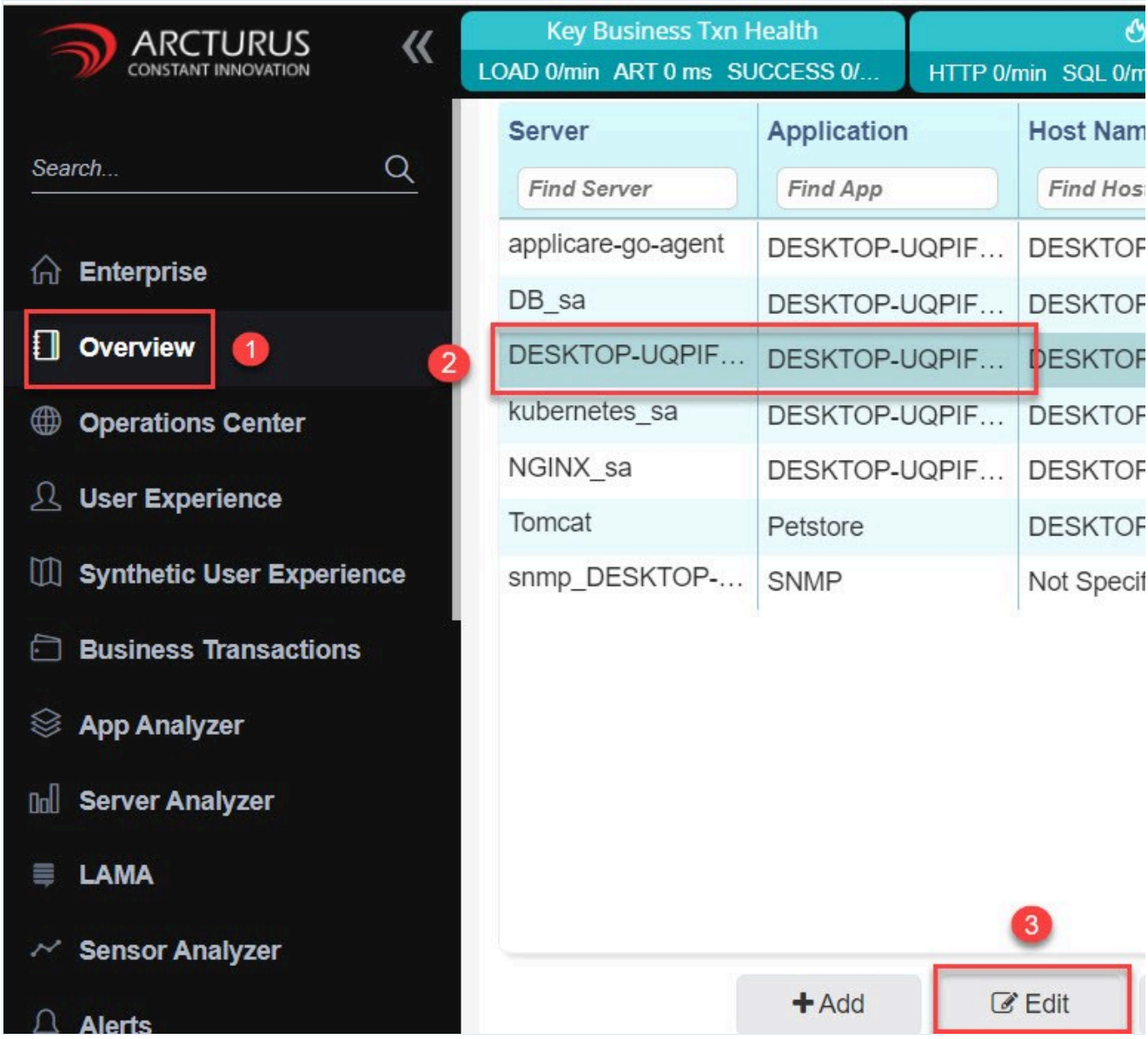
Applicare 9.2 · Updated September 24, 2024 · cobol · **Pending technical review**

COBOL monitoring is especially important for maintaining the health, performance, and security of COBOL-based systems, which often serve as the backbone for mission-critical applications in industries like finance, insurance, and government.

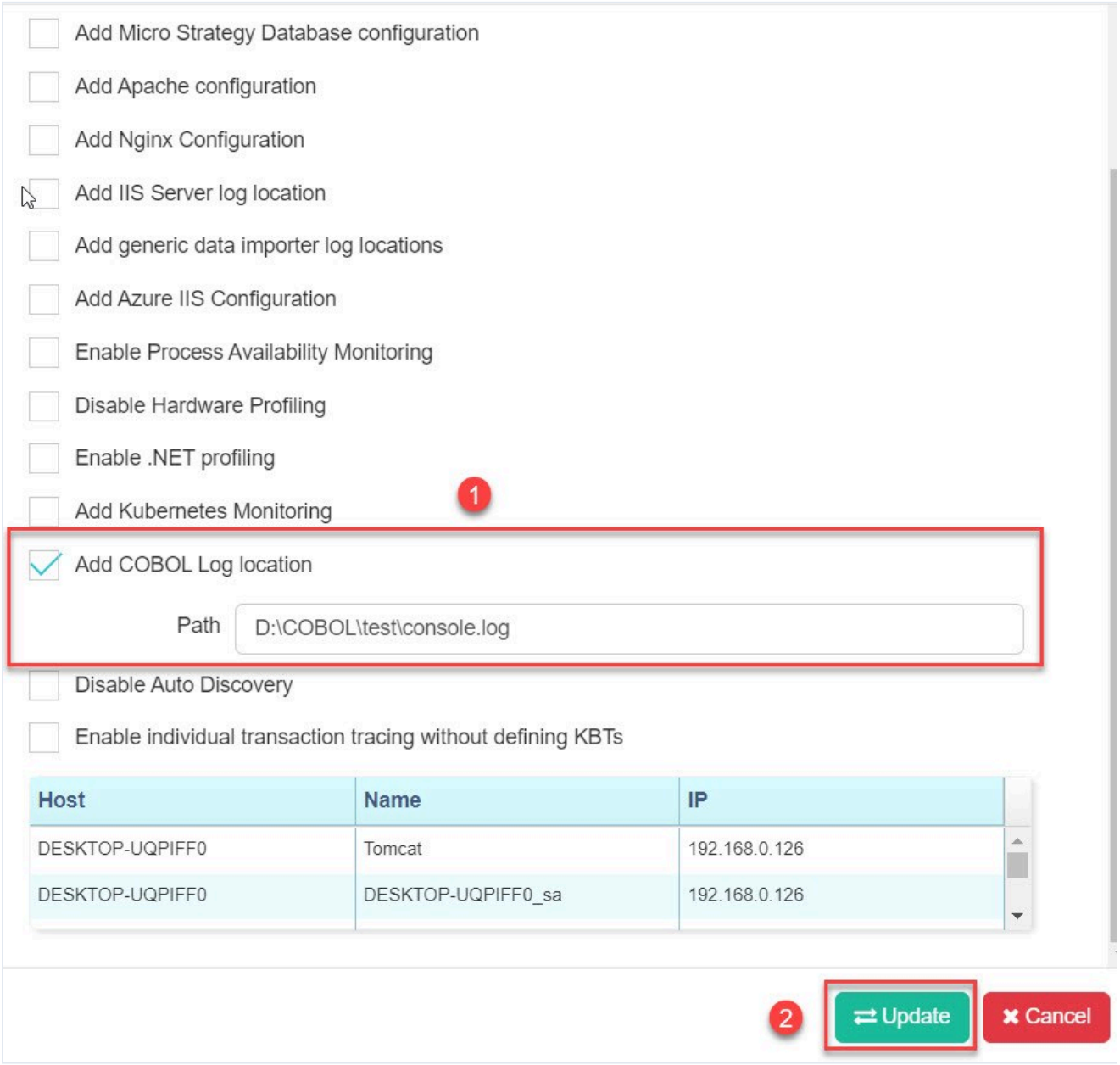
Applicare doesn't require any adjustments to COBOL applications. It pulls in the logs generated by COBOL applications and gathers transactions data from them.

Configuration for the COBOL Log monitoring in Applicare

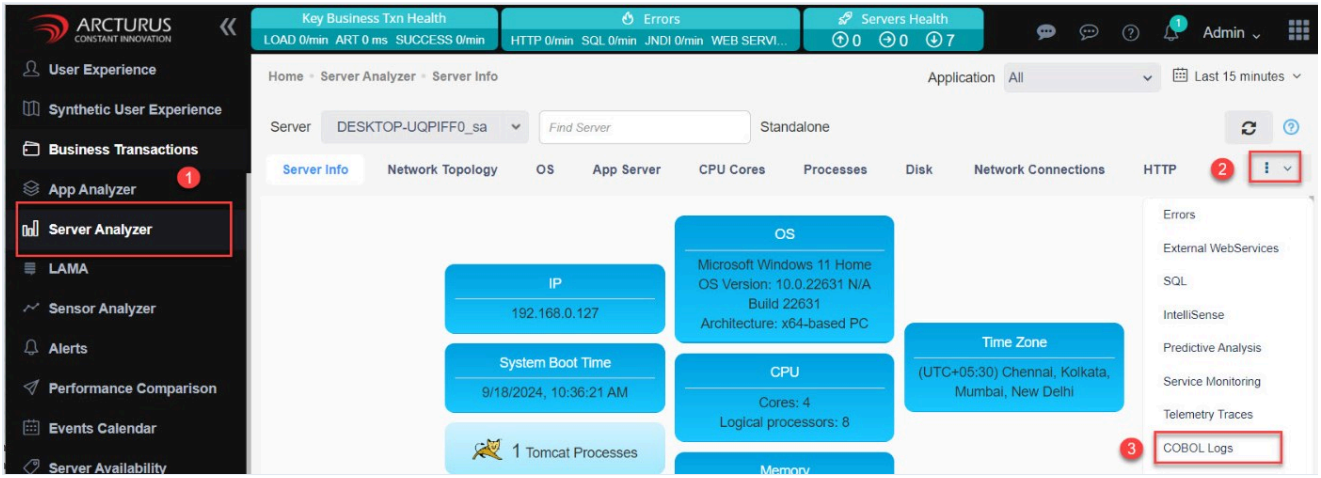
1. Go to the **Overview** menu and select the **server** and click the **Edit** button.

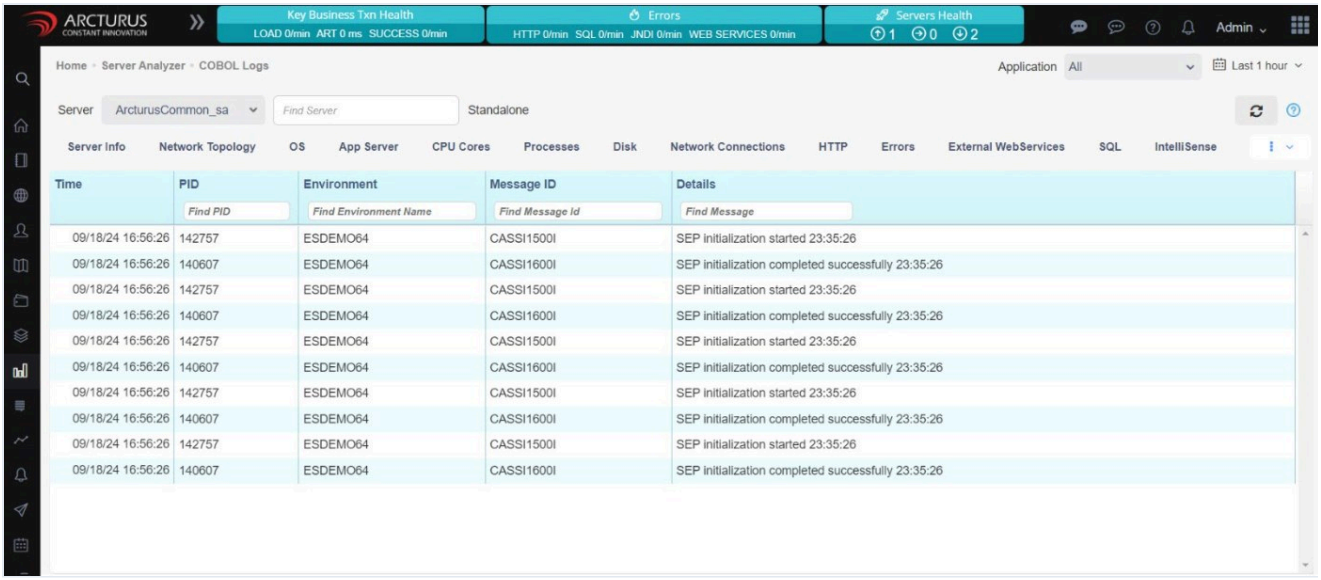


2. Enable the **Add COBOL Log Location** checkbox and enter the log file path and click the **update** button.



3. Go to the **Server Analyzer** menu and select the **COBOL Logs**.





Why COBOL Monitoring with Applicare?

1. Error Detection and Troubleshooting Applicare provide detailed information about errors and exceptions that occur during the execution of COBOL programs. Monitoring the COBOL logs in real time helps detect issues early, allowing teams to quickly troubleshoot and resolve problems before they escalate into larger failures.

2. Performance Analysis Allows teams to identify performance bottlenecks, slowdowns, and inefficiencies, enabling optimizations that ensure smooth operations.

3. Compliance and Security COBOL systems frequently handle sensitive information, especially in sectors like banking and healthcare. Applicare COBOL log monitoring is essential for tracking access, changes to data, and unauthorized activities. This is crucial for ensuring compliance with industry regulations and maintaining system security through audit trails and anomaly detection.

4. Root Cause Analysis When issues arise, historical logs are essential for performing root cause analysis. Applicare COBOL log monitoring provides access to historical data that can help teams trace the cause of failures or inefficiencies, identify which part of the system was impacted, and determine the best course of action for resolution.

5. Business Continuity and Uptime Applicare monitoring ensures that the COBOL systems, which often support critical business operations, continue to run smoothly. Any significant issue that impacts uptime, such as job failures or resource exhaustion, can be quickly identified through logs, ensuring minimal disruption to business processes.

6. Audit and Accountability Applicare COBOL log monitoring helps in keeping a detailed record of activities within COBOL systems, including who accessed what and when. This supports both internal auditing and regulatory compliance, ensuring that all activities are traceable and transparent.

7. Security Incident Detection Applicare monitoring is key to identifying security breaches or attempts at unauthorized access. By continuously tracking login attempts, changes to system configurations, or unusual behavior in the COBOL system, it becomes easier to detect and respond to security threats in real-time.

8. Support for Legacy Systems Transition If an organization is modernizing or migrating away from COBOL-based systems, log monitoring helps ensure a smooth transition by providing insights into current operational patterns. Understanding how the COBOL system behaves through its logs makes it easier to map requirements for new systems or monitor the migration process.

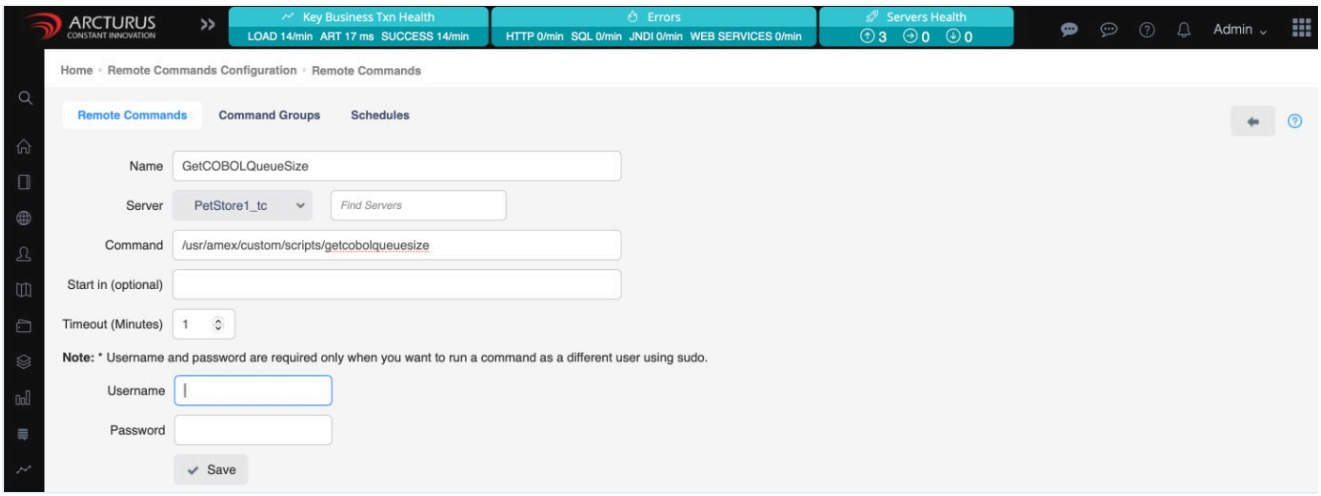
In summary, Applicare plays a crucial role in ensuring system stability, security, and performance. It enables organizations to quickly detect and address issues, maintain compliance, and support ongoing modernization efforts, while minimizing the risk of downtime or data breaches.

CONFIGURATION GUIDES › NODE.JS, PHP, GO & COBOL

COBOL Monitoring

Applicare 9.2 · Updated September 13, 2024 · cobol · **Pending technical review**

Applicare supports COBOL and other legacy technologies without requiring source code changes by using Shell Script Sensors and Remote Commands. Applicare can ingest data from existing COBOL monitoring commands via Shell Script Sensors. For example, if you need to monitor, chart, alert, or report on queue sizes for COBOL requests, and you have a shell script named getQueuesize that utilizes COBOL/OS APIs to return the precise queue size, you can set up a Shell Script Sensor in Applicare to run at your desired frequency. Once the Shell Script Sensor is configured, Applicare will process the queue size data natively, enabling you to display, chart, and report on this data seamlessly.



Monitoring with Applicare

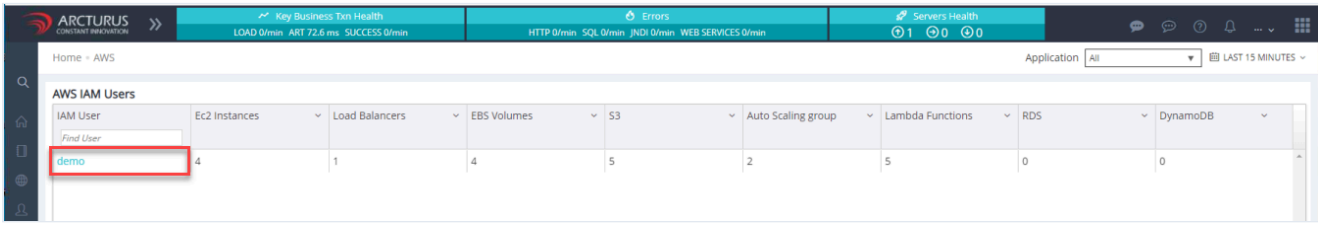
CONFIGURATION GUIDES › AWS

AWS Monitoring

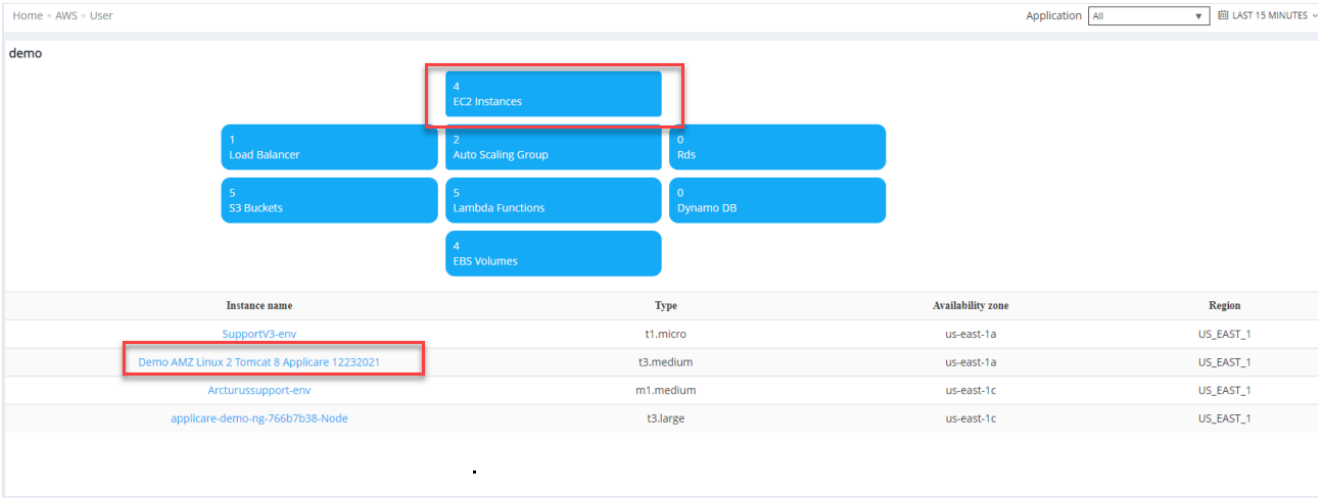
Applicare 9.2 · Updated September 13, 2024 · aws · **Pending technical review**

Agent less Monitoring of the AWS environment

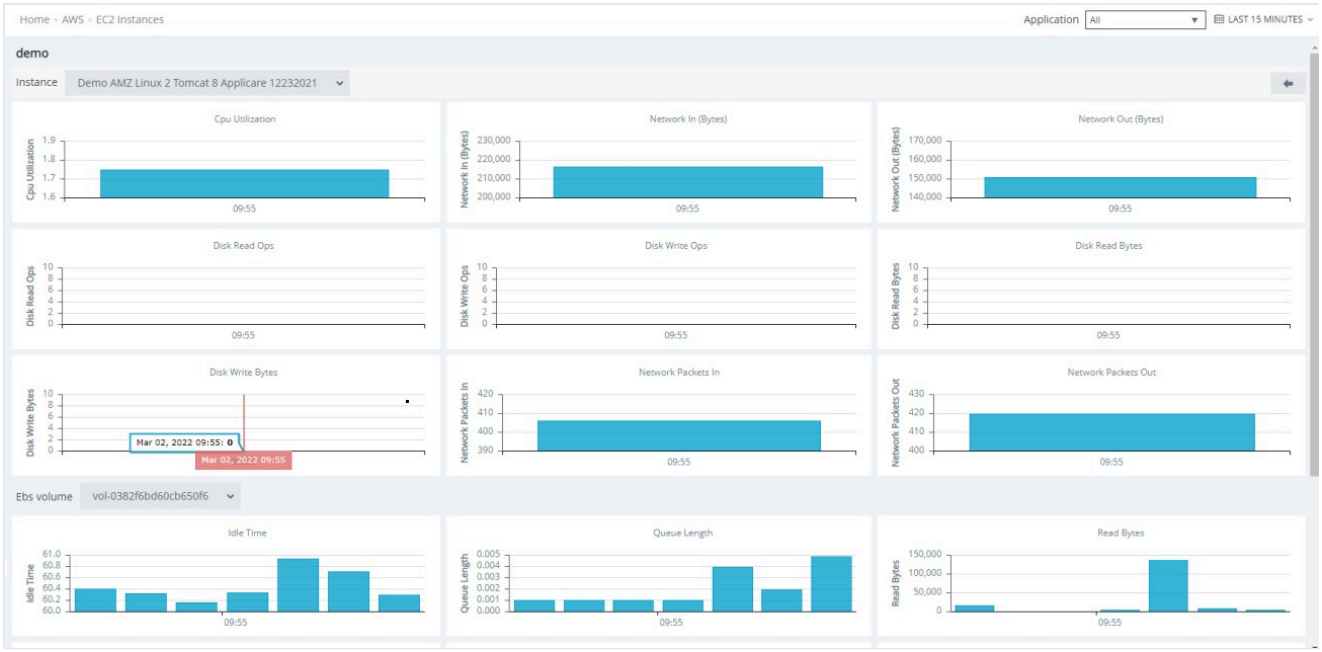
1. Log in to the Applicare controller and navigate to the **AWS** screen in the left side menu. Select the configured IAM user.



2. Applicare automatically maps the entire AWS environment, with information being auto-detected and updated as changes occur. You can view detailed information by selecting a specific AWS resource.



3. You can access a detailed view of the resource without needing to deploy any agents.



Follow this for **AWS monitoring configuration**

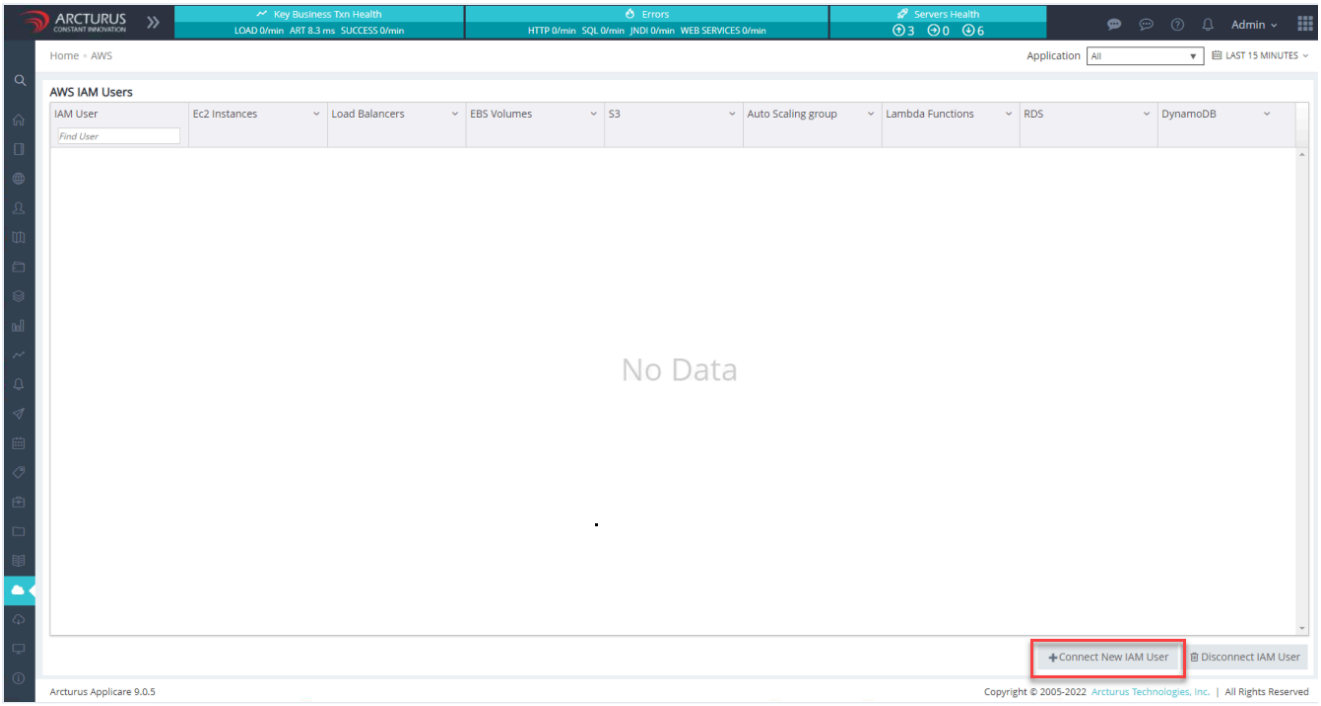
Monitoring with Applicare

CONFIGURATION GUIDES › AWS

Configuring AWS Monitoring

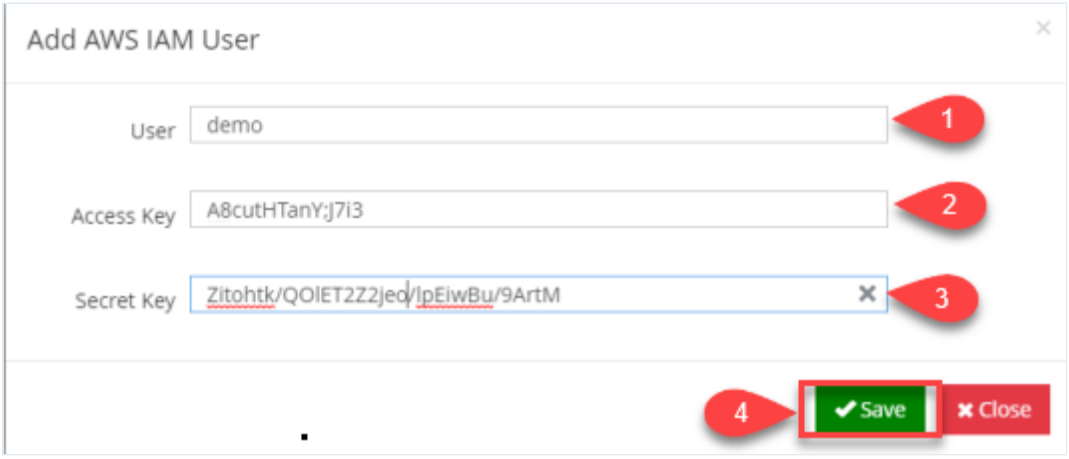
Applicare 9.2 · Updated September 13, 2024 · aws · **Pending technical review**

Steps 1: Log in to Applicare controller. On the left side menu, navigate to the AWS screen and click **Connect New IAM User** button located at the bottom right of AWS screen.



Step 2: Fill in the Add AWS IAM user popup with the required information as shown below. and then click the **Save** button. If needed, follow the steps in the **IAM user creation/configuration for AWS monitoring guide for further instructions.**.

- **User**: Enter the IAM user name.
- **Access Key**: Enter the access key for the IAM user from the monitored AWS account.
- **Secret Key**: Enter the secret key for the IAM user from the monitored AWS account.



Save the configuration and allow up to 15 minutes for the data to become available.

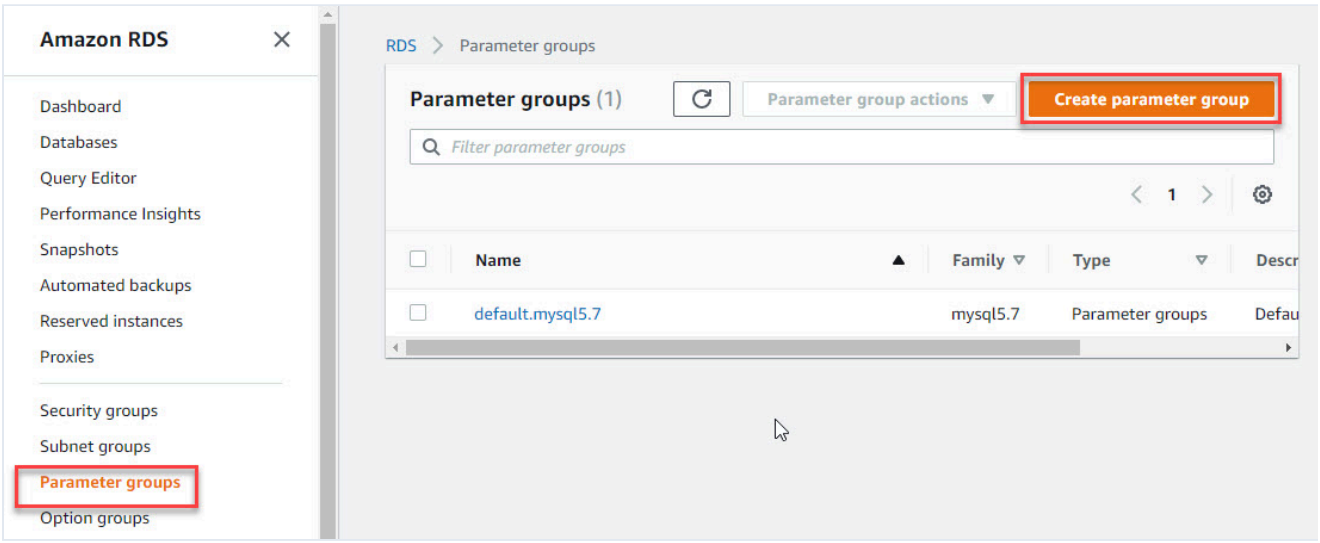
Monitoring with Applicare

CONFIGURATION GUIDES › AWS

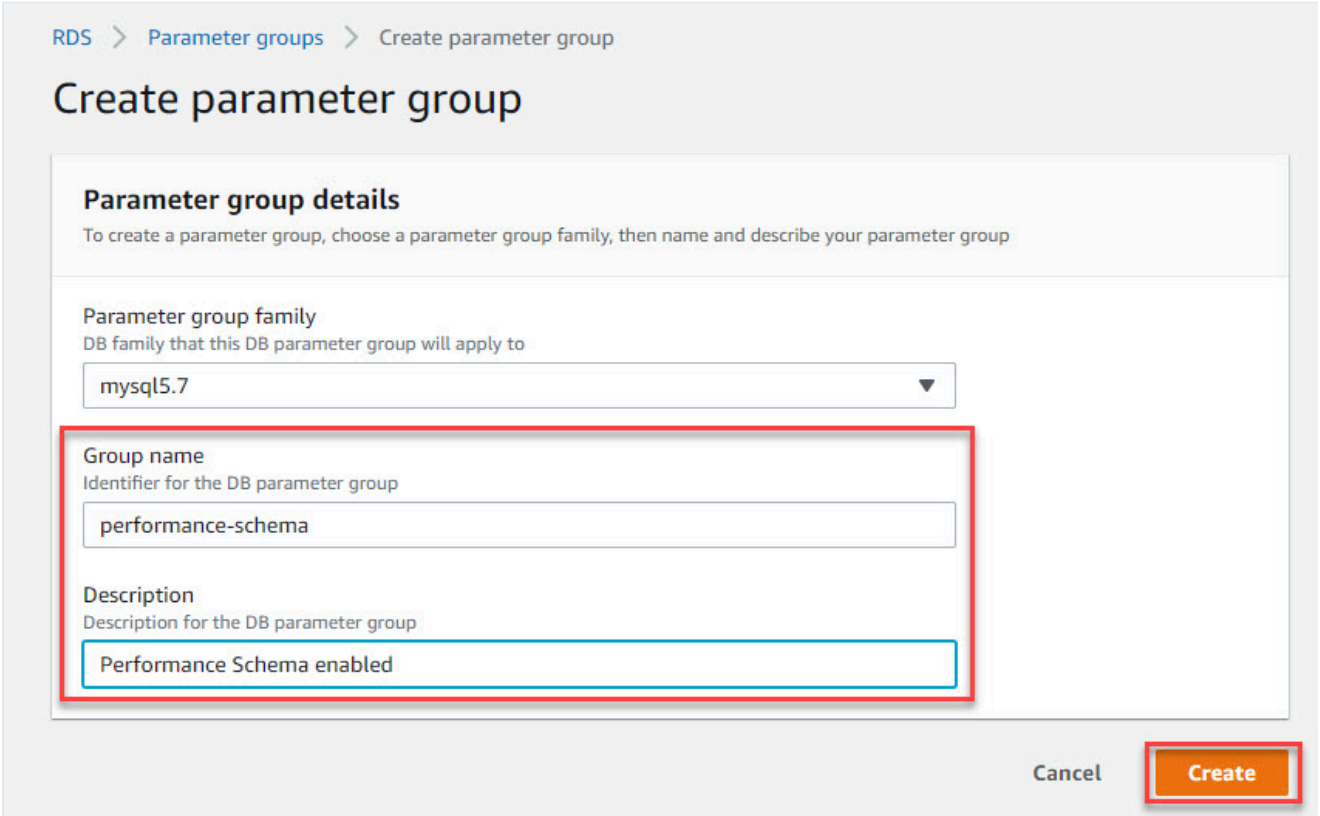
Creating a DB parameter Group for AWS RDS MySQL

Applicare 9.2 · Updated August 22, 2024 · aws, mysql · **Pending technical review**

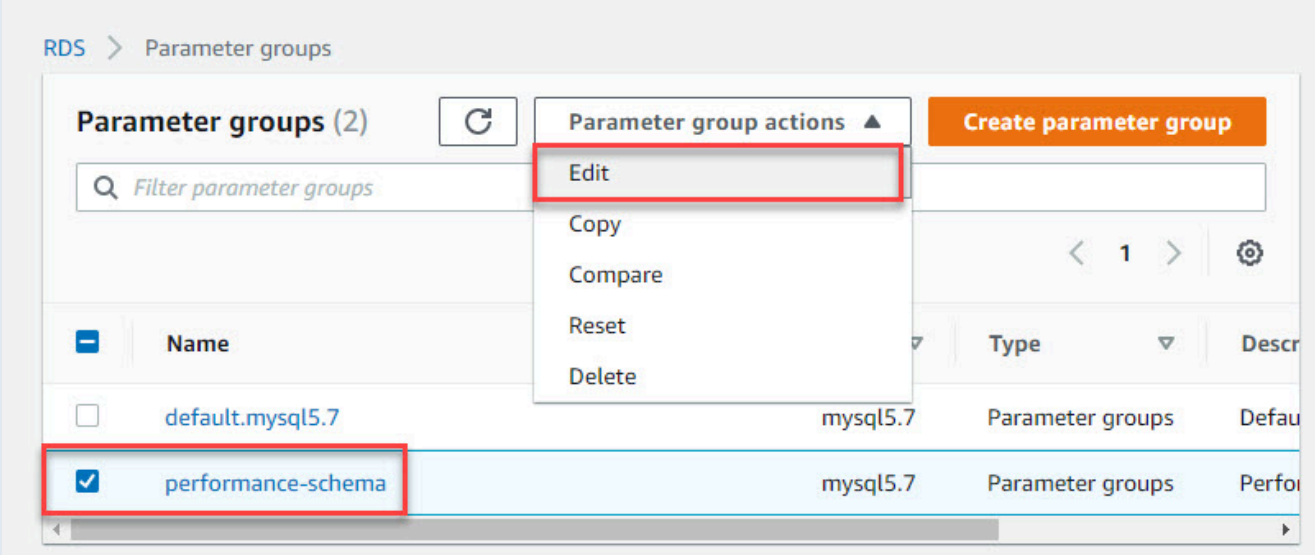
1. Navigate to **RDS -> Parameter Groups** and click the **Create parameter group** button.



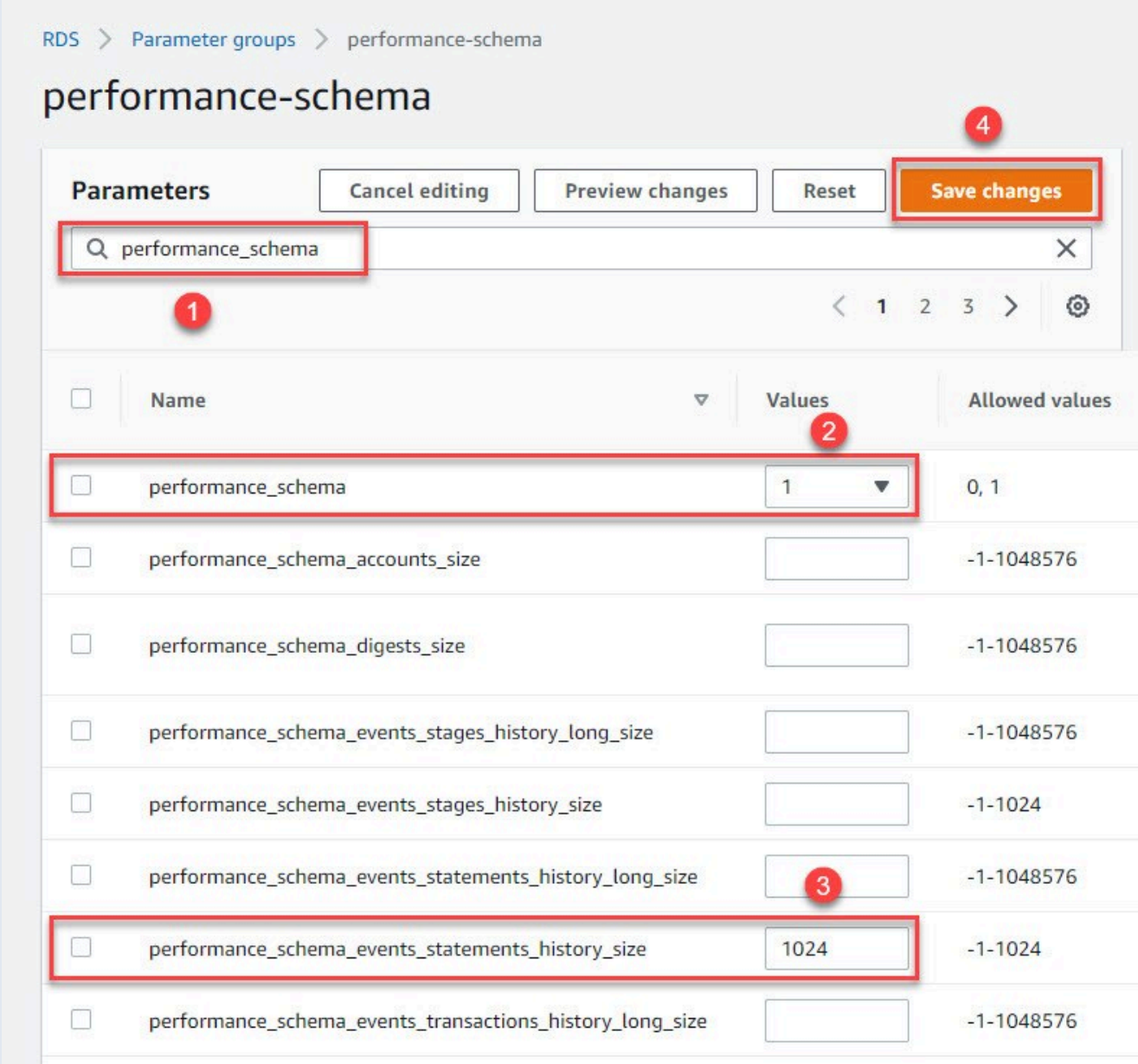
2. Enter the **Group name** and **Description** then click the **Create** button.



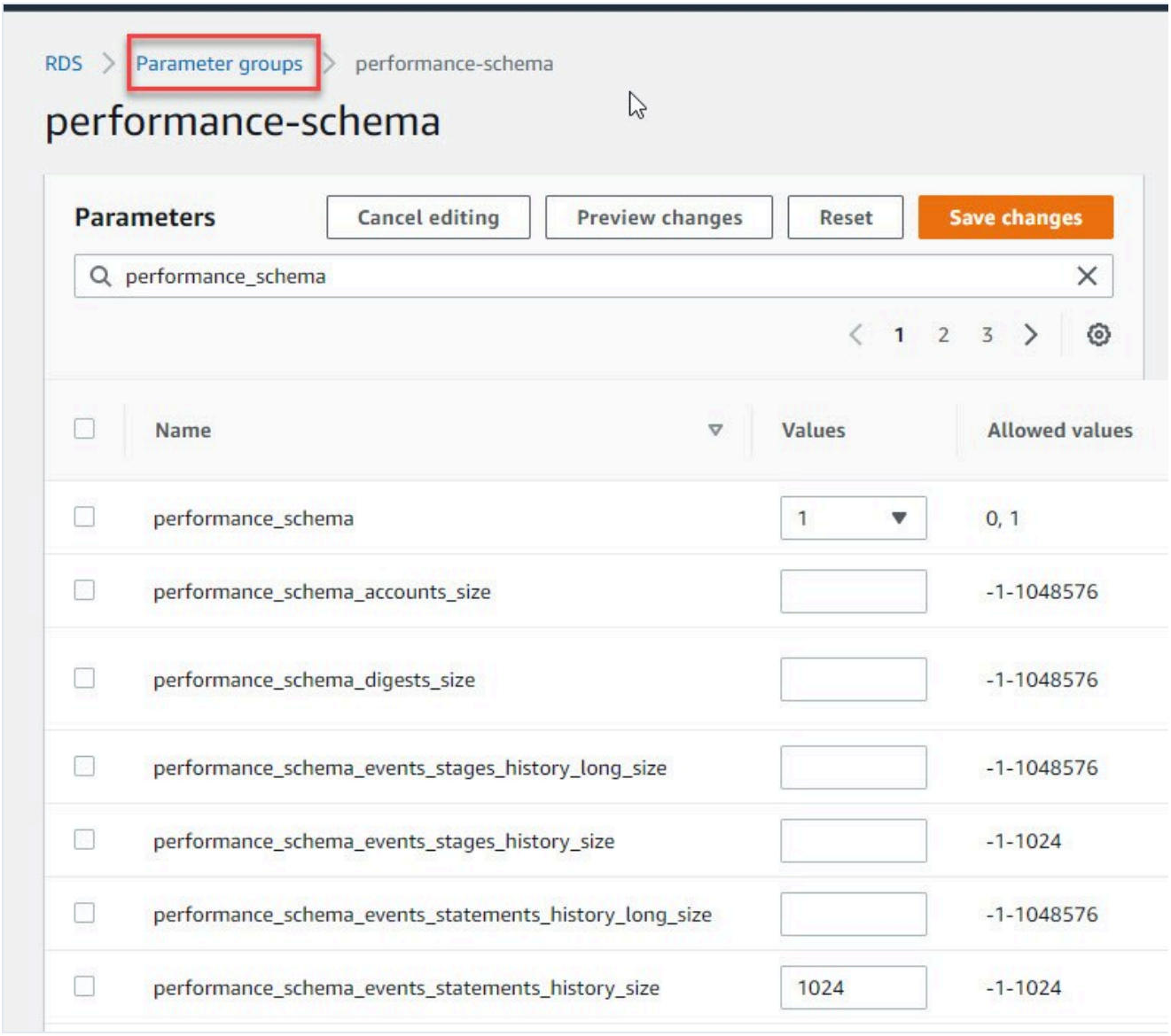
3. Select the **newly created parameter groups** and select the **Edit** option from the **Parameter group actions** dropdown.



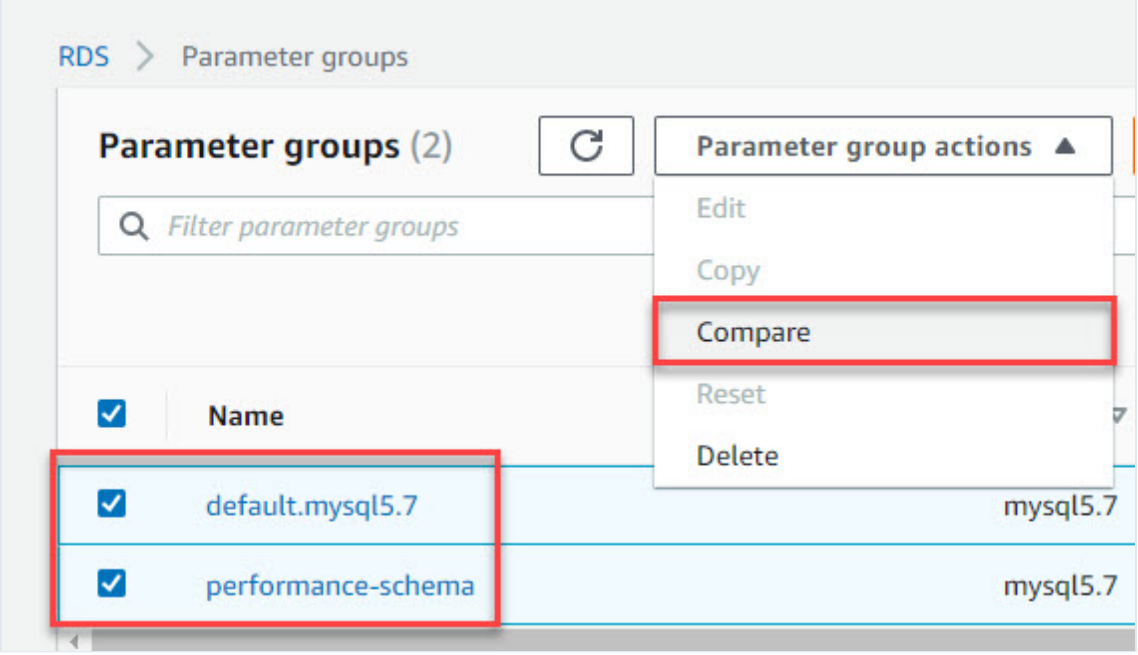
4. Search for **performance_schema** and set its value to **1** change **performance_schema_events_statements_history_size** to **1024** and click the **Save changes** button.



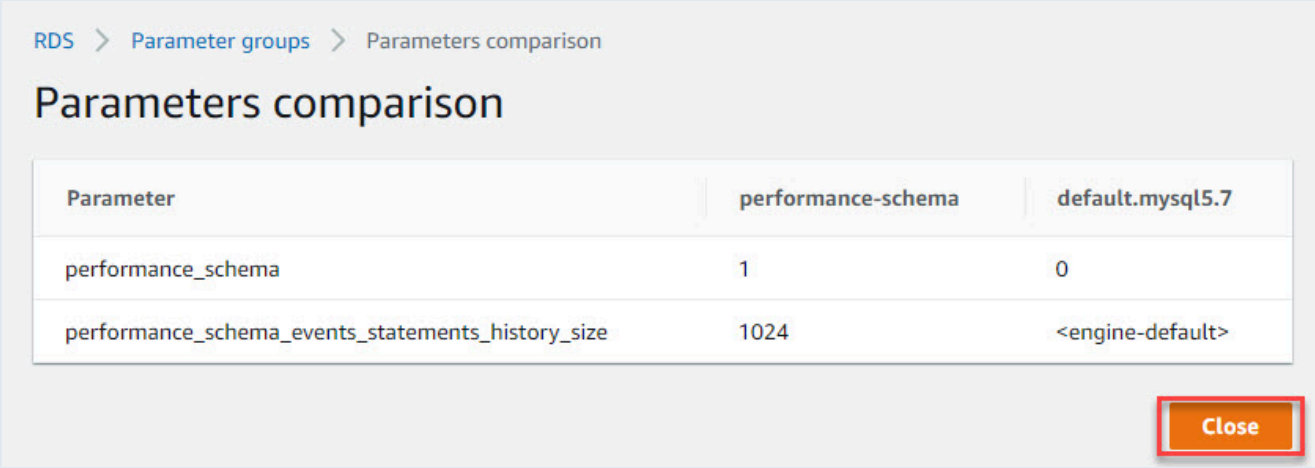
5. Navigate to the **Parameter Groups** screen.



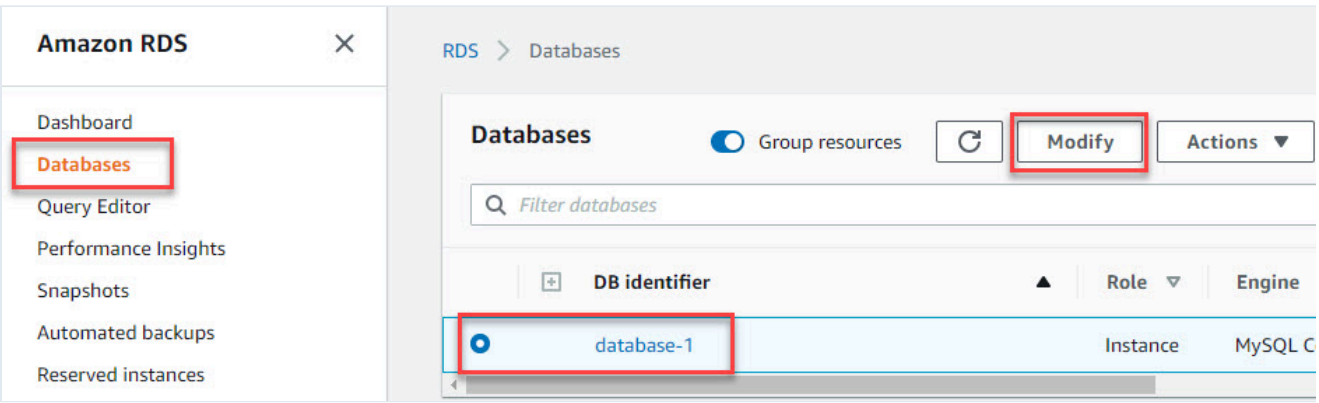
6. Select both the **default** and **newly created parameter** groups, then choose the **Compare** option from the **Parameter group actions** dropdown.



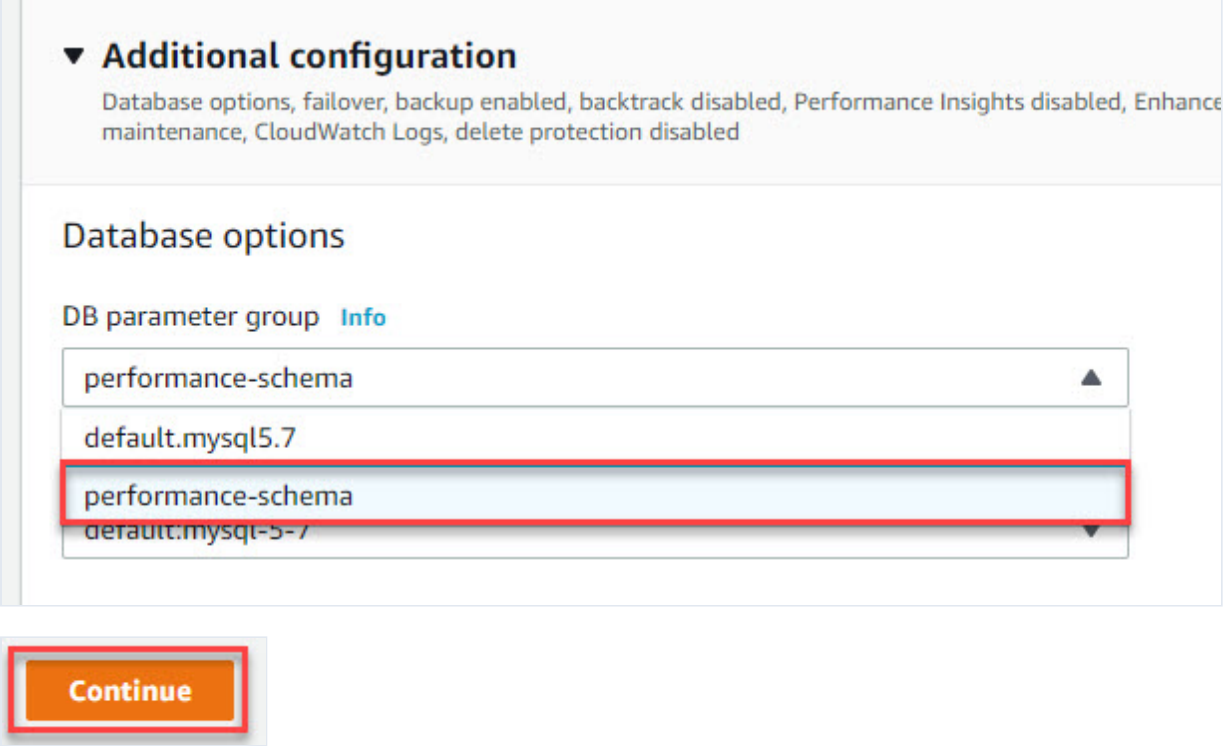
7. This will display a list of **Parameters list** that differ between the from default and the newly created groups. If there are additional changes beyond these parameters, edit the newly created parameter group to include those change and save your updates.



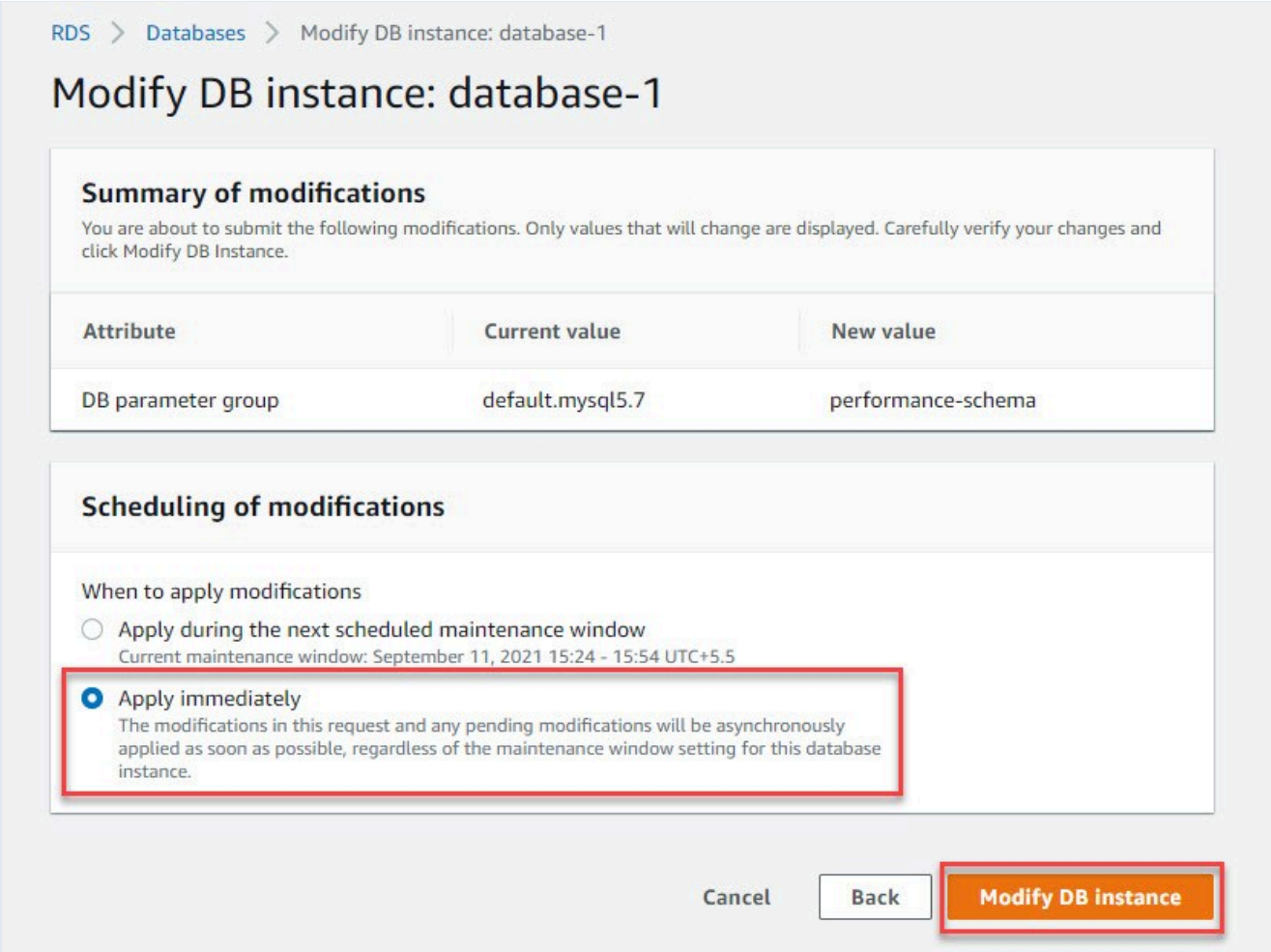
8. Navigate to **RDS > Databases** and click the **Modify** button.



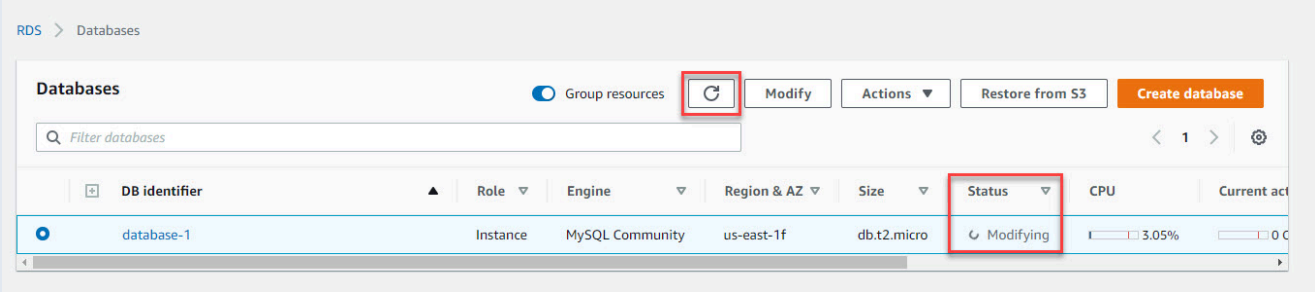
9. Choose the newly created parameter group from the **DB parameter group** dropdown and click the **Continue** button.



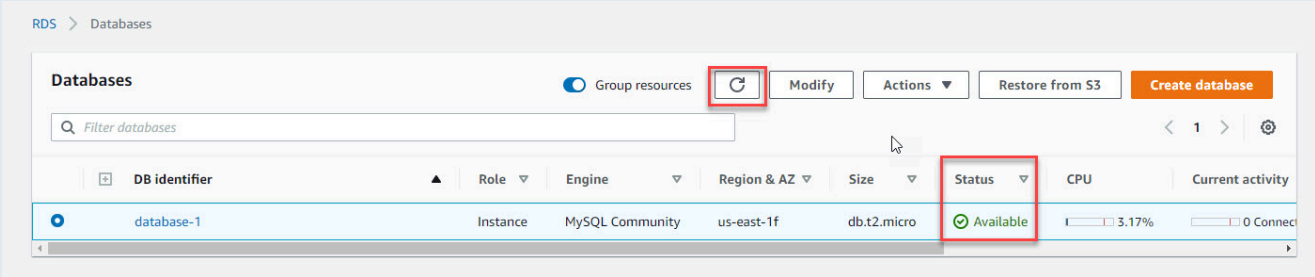
10. Select the **Apply immediately** option and click the **Modify DB instance** button.



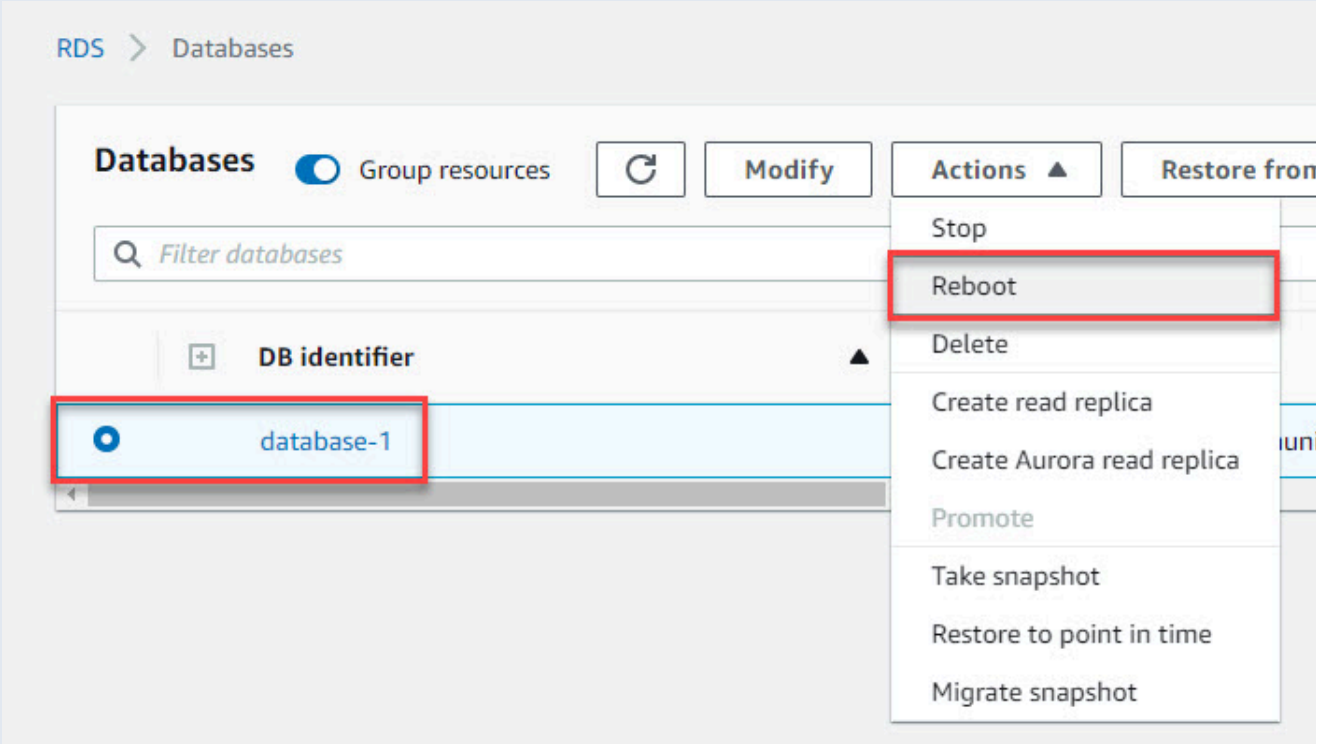
11. Select the DB instance and click the **Reload** button to check the status. It will display the status as **Modifying**. Wait until the Status to became Available.



12. Click the **Reload** button to check the status frequently.



13. Once the status changes Available, select the **Reboot** option from the **Actions** dropdown.



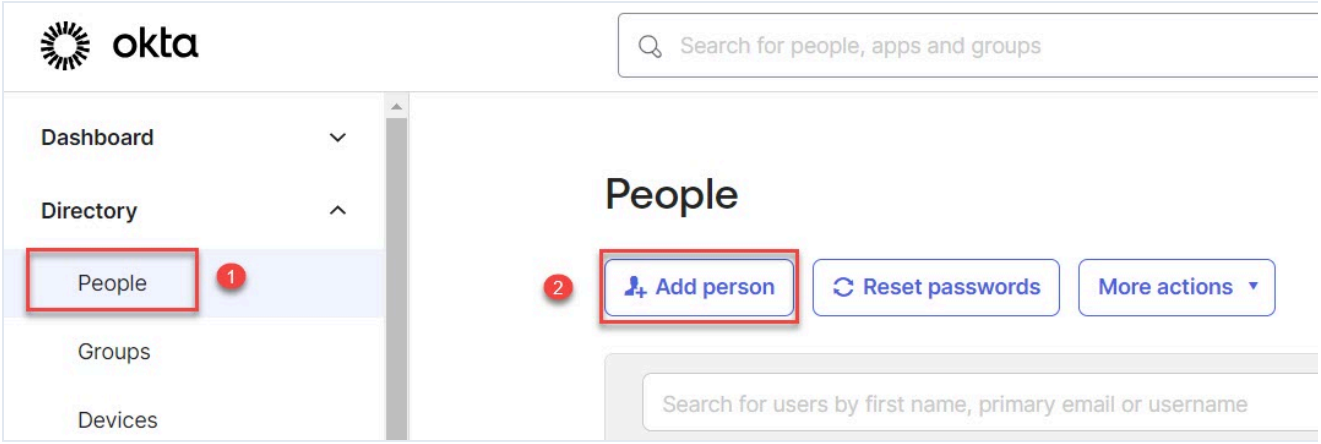
CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Configuring Applicare with OKTA using SAML

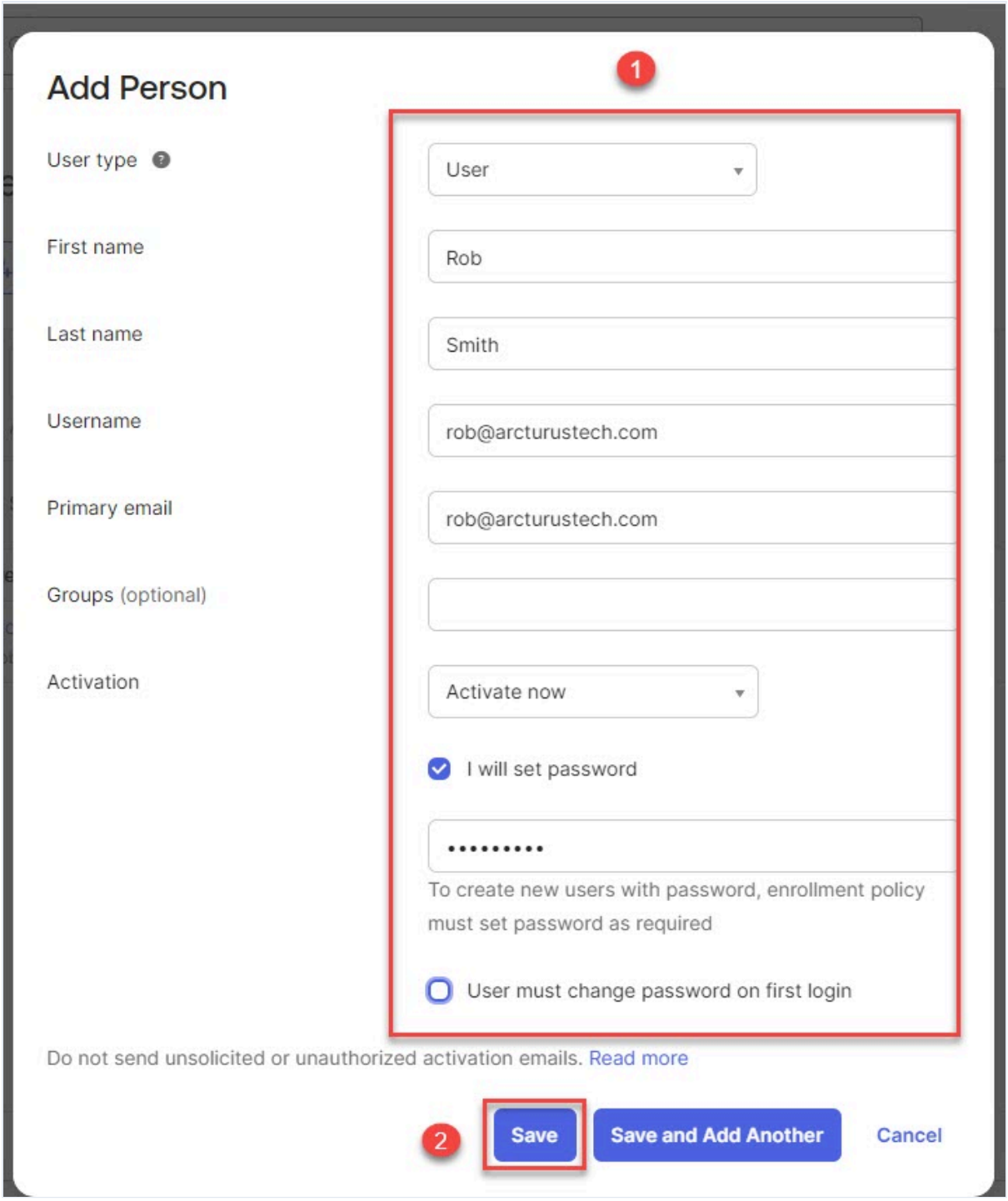
Applicare 9.2 · Updated September 13, 2024 · sso · **Pending technical review**

User Creation

1. Log in to your **OKTA** Admin account and select People from the **Directory** menu.

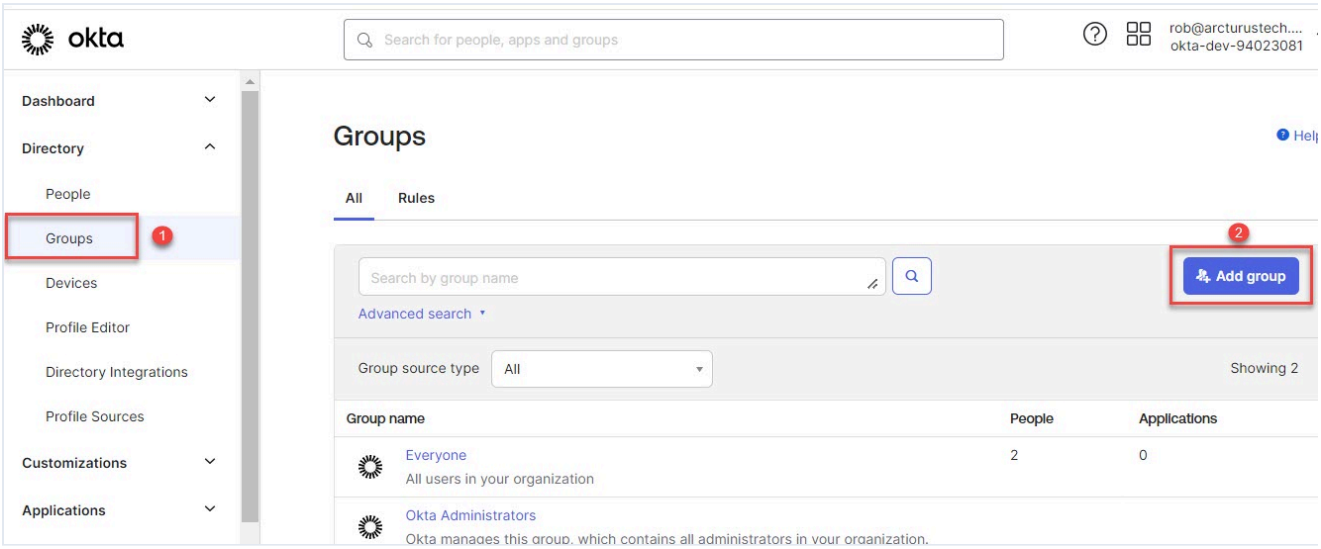


2. Enter the user details and set the password and click the **Save** button.

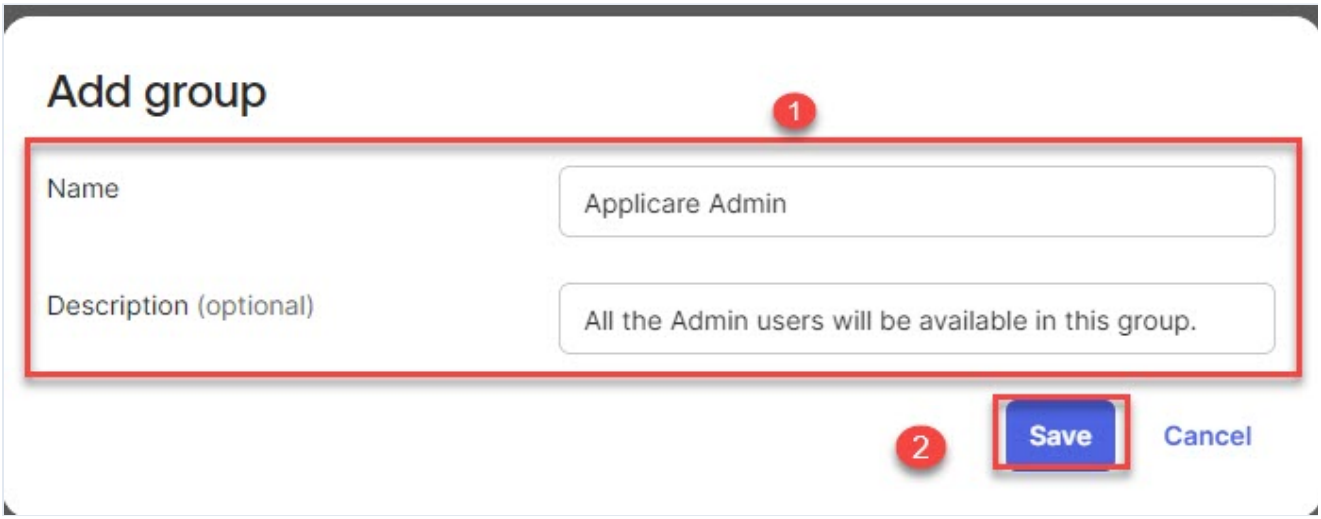


Group Creation

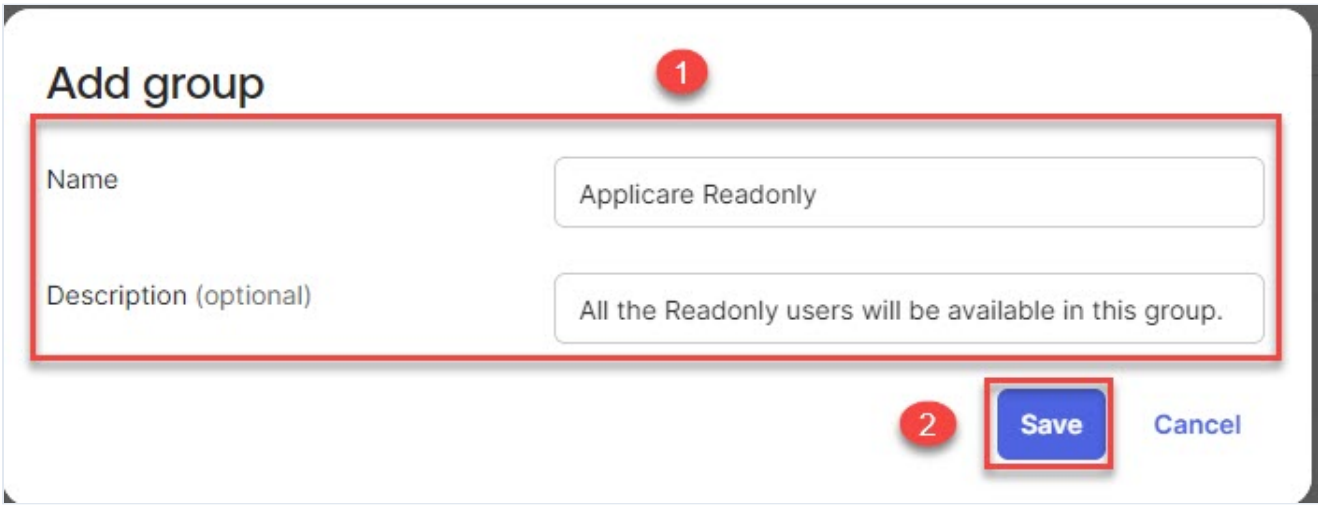
1. Go to **Groups** under the **Directory** menu and click the **Add Group** button.



2. Enter **Applicare Admin** as the name and click the **Save** button. The description is optional.

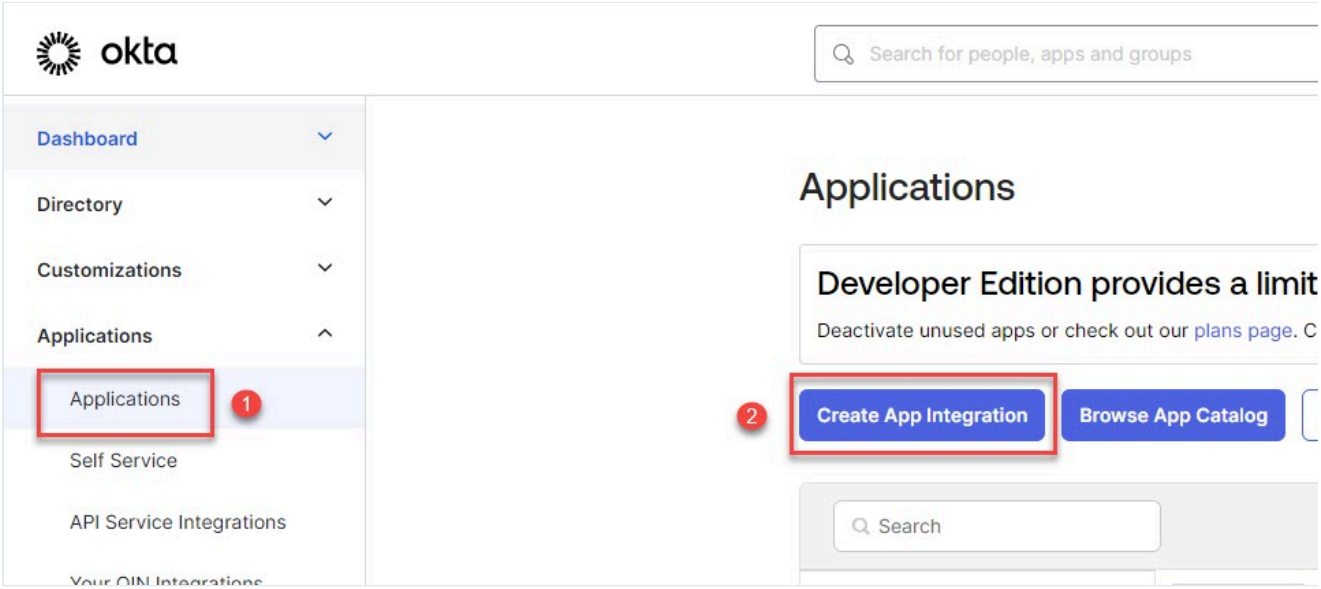


3. Similarly, add another group with the name as **Applicare Readonly** and click the **Save** button. The description is optional.

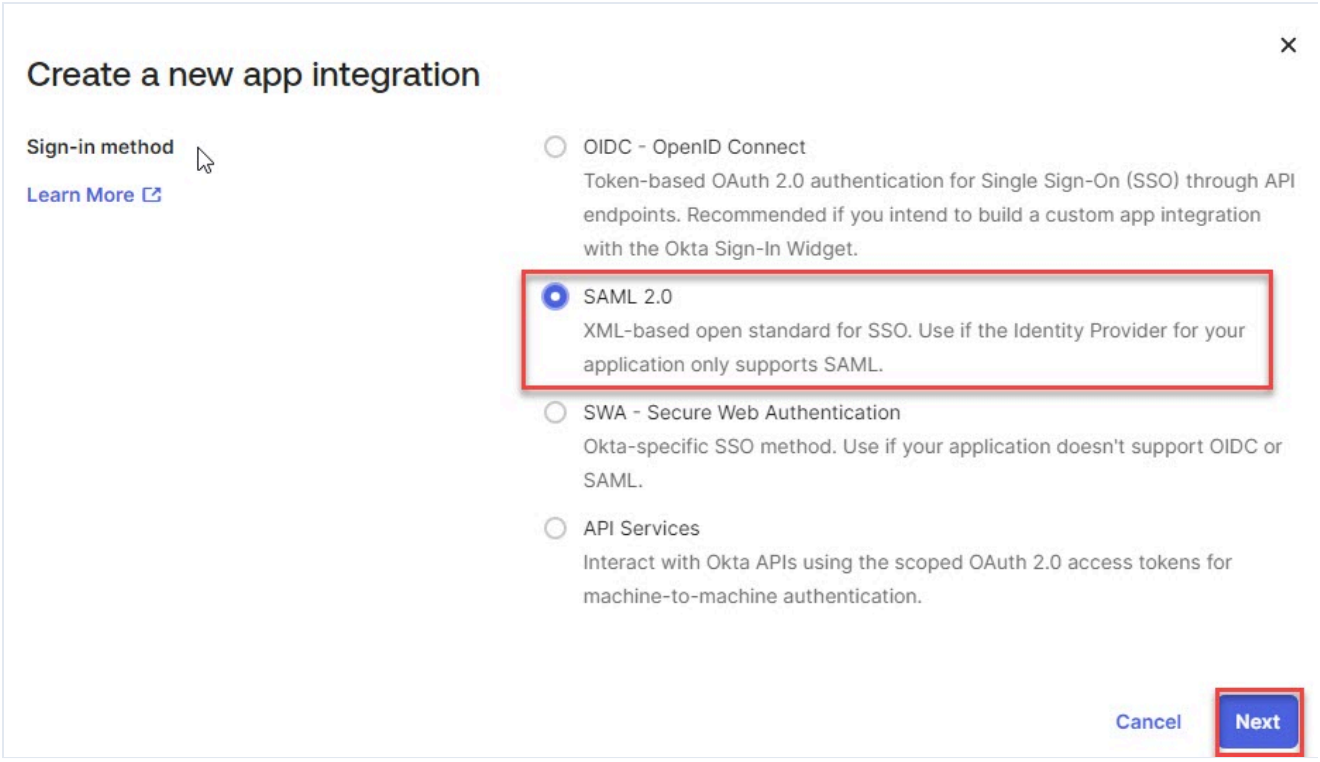


App Integration

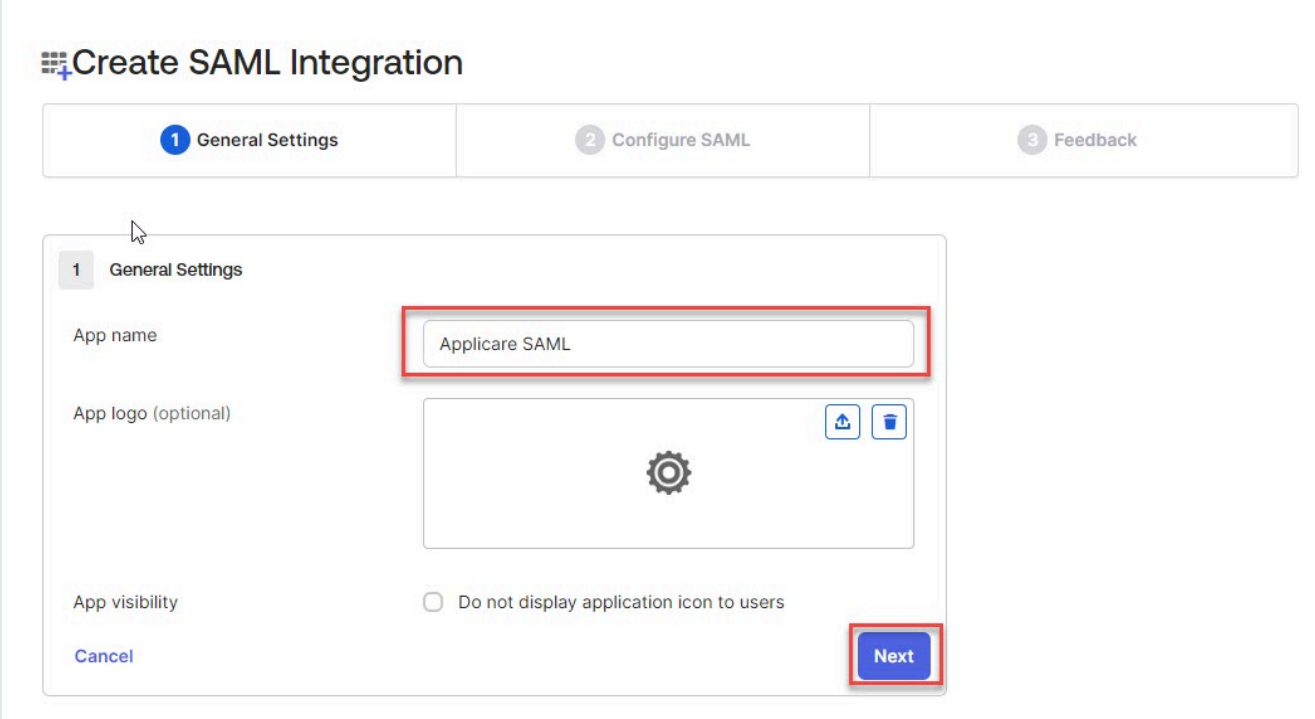
1. Go to **Applications** under the **Applications** menu and click the **Create App Integration** button



2. Select **SAML 2.0** and click the **Next** button.



3. Enter the App name as **Applicare SAML** and click the **Next** button.

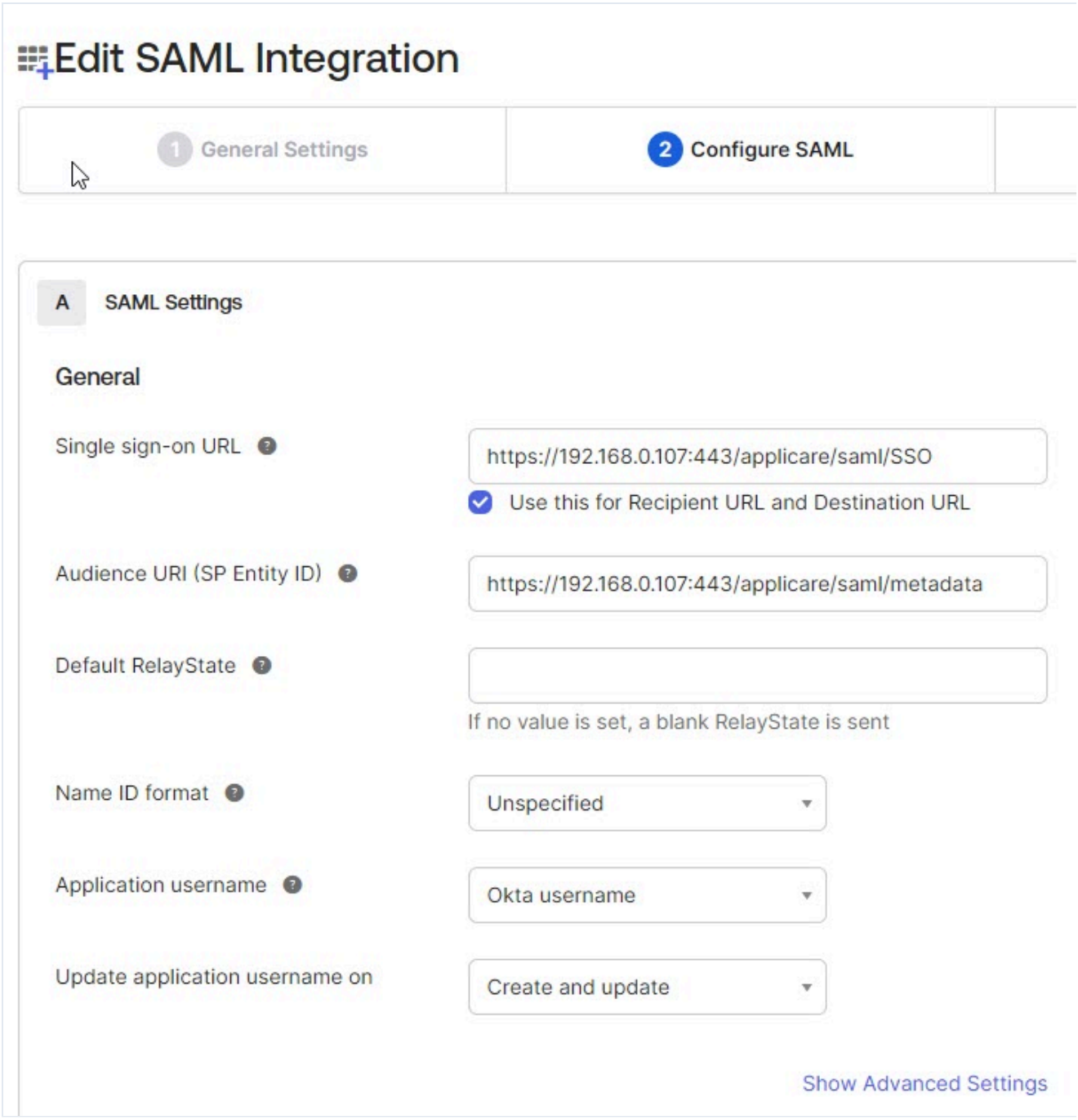


4. Enter the **Single sign-on URL** and **Audience URI** and select other options in the dropdown.

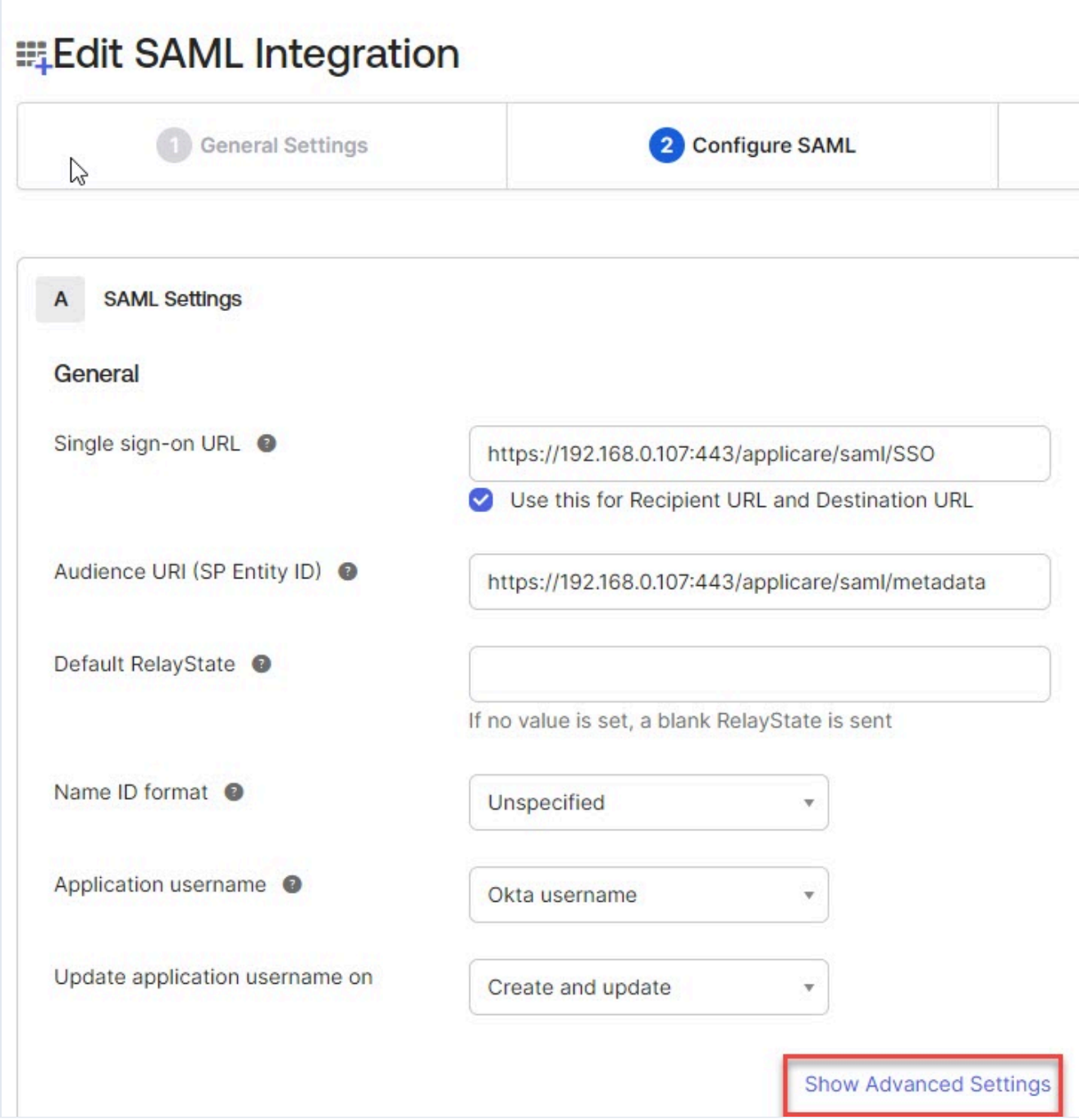
- Single sign-on URL -> **Applicare_Base_URL**/saml/SSO
- Audience URI (SP Entity ID) -> **Applicare_Base_URL**/saml/metadata
- Replace the **Applicare_Base_URL** with the actual URL.

e.g.

- Applicare_Base_URL-> `https://192.168.0.107:443/applicare`
- Single sign-on URL -> `https://192.168.0.107:443/applicare`/saml/SSO
- Audience URI (SP Entity ID) -> `https://192.168.0.107:443/applicare`/saml/metadata



5. Click **Show Advanced Settings** button



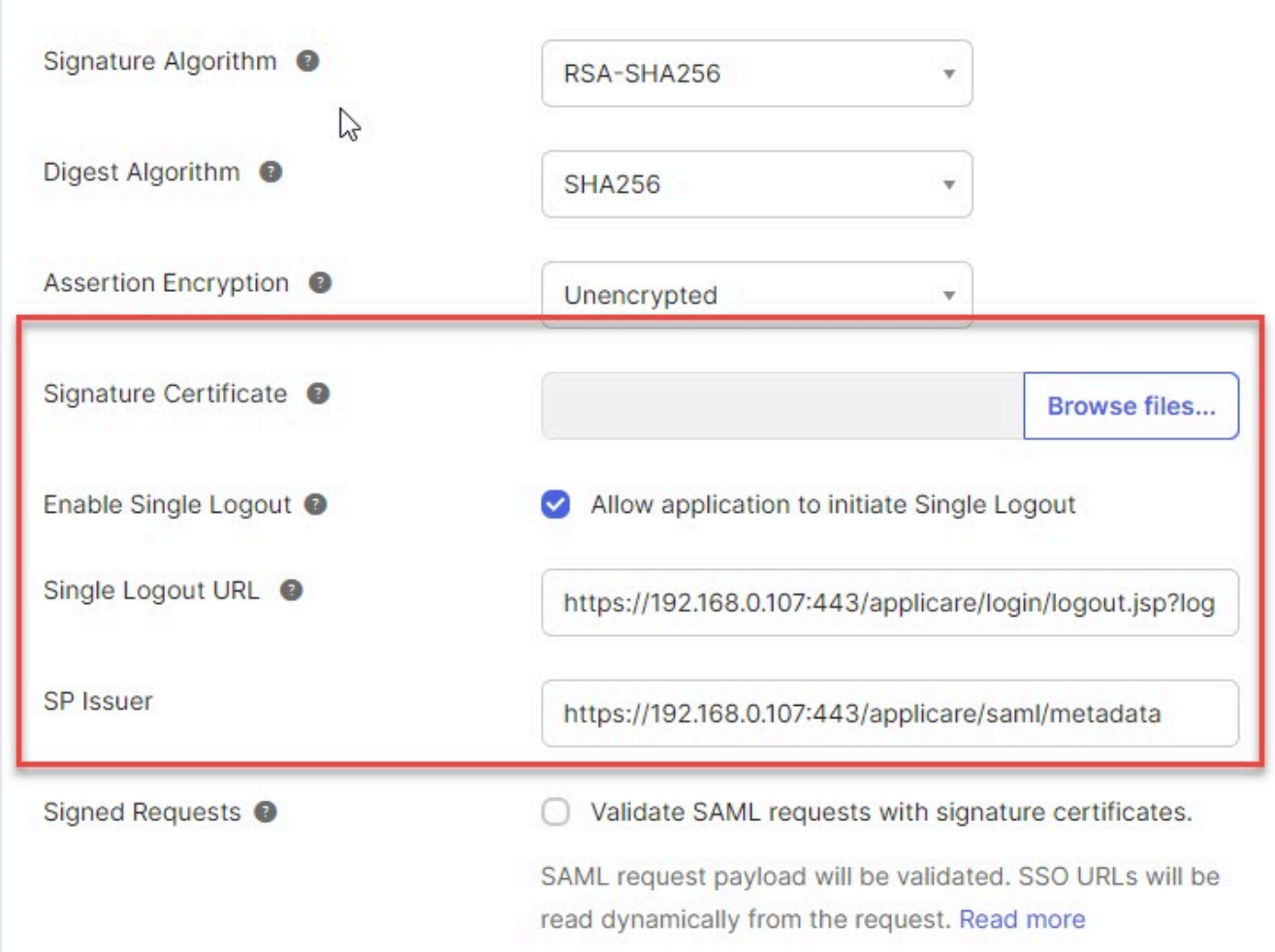
6. Add the below details

- Browse and upload the Applicare Signature Certificate file (.cer). This certificate file is generates during the HTTPS configuration for Applicare. For instruction on self- signed HTTPS configuration **Click here.**
- Check the box for **Allow Application to initiate Single Logout**.
- Enter the **Single Logout URL** as **Applicare_Base_URL**/login/logout.jsp?loginType=console
- Enter the **SP Issuer** as **Applicare_Base_URL**/saml/metadata
- Replace the **Applicare_Base_URL** with actual URL.

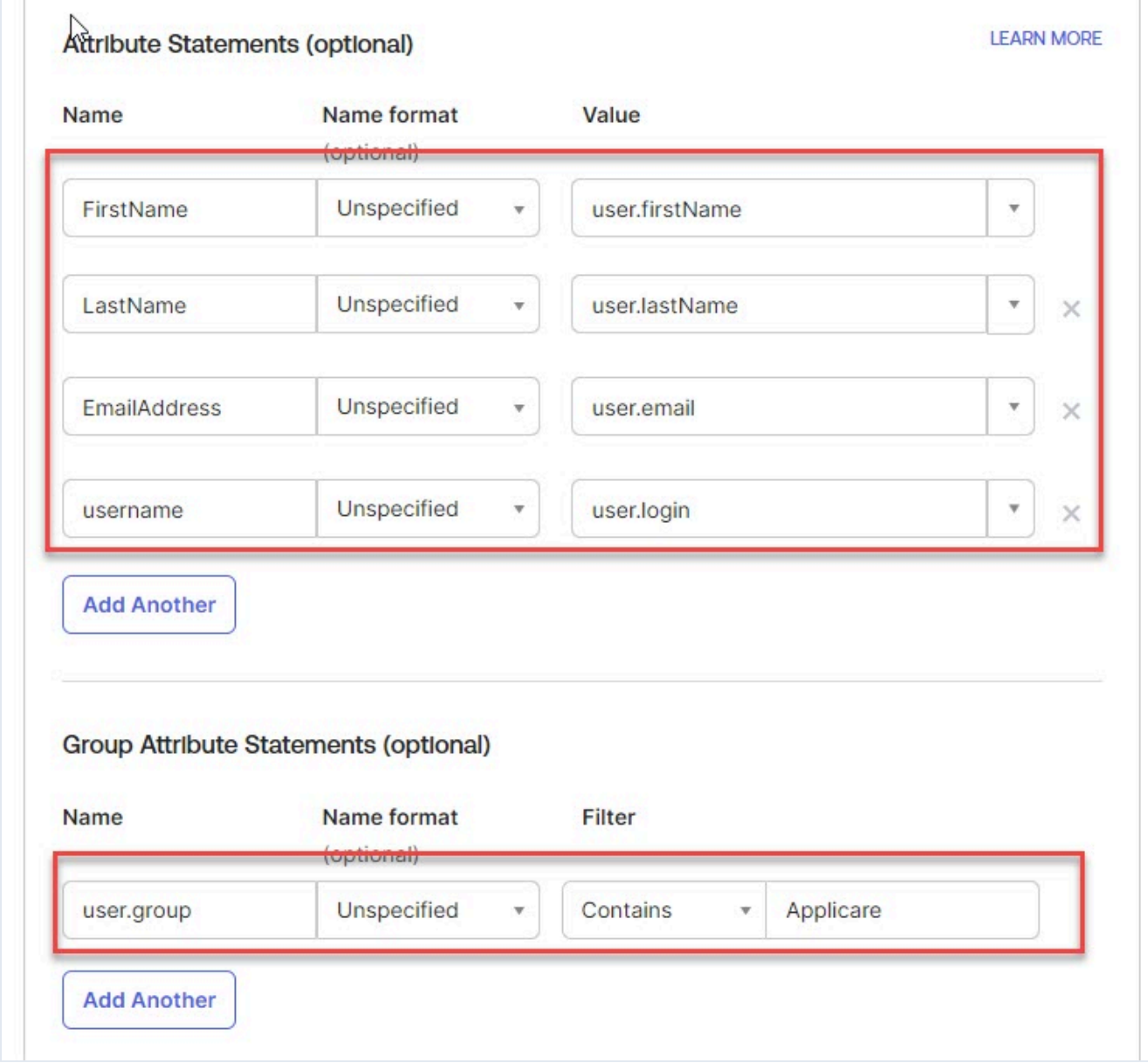
e.g.

- Applicare_Base_URL -> `https://192.168.0.107:443/applicare`

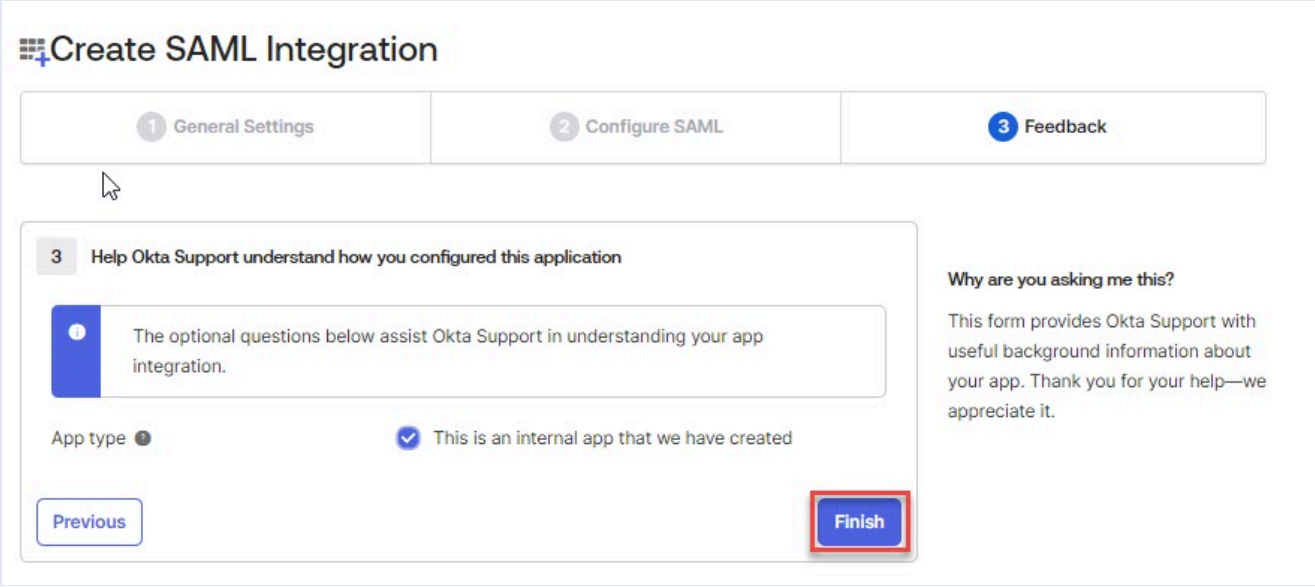
- Single Logout URL -> `https://192.168.0.107:443/applicare`/login/logout.jsp?loginType=console
- SP Issuer -> `https://192.168.0.107:443/applicare`/saml/metadata



7. Add the **FirstName, LastName, EmailAddress, Username** and **user.group** attributes, then click the **Next** button.

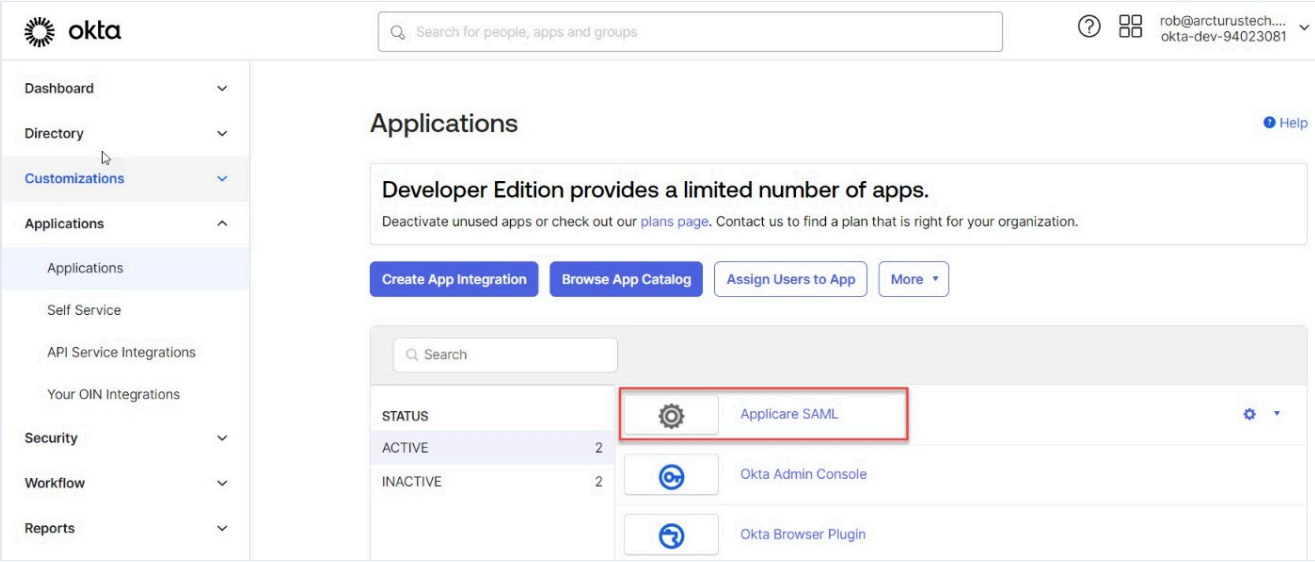


8. Click the **Finish** button.

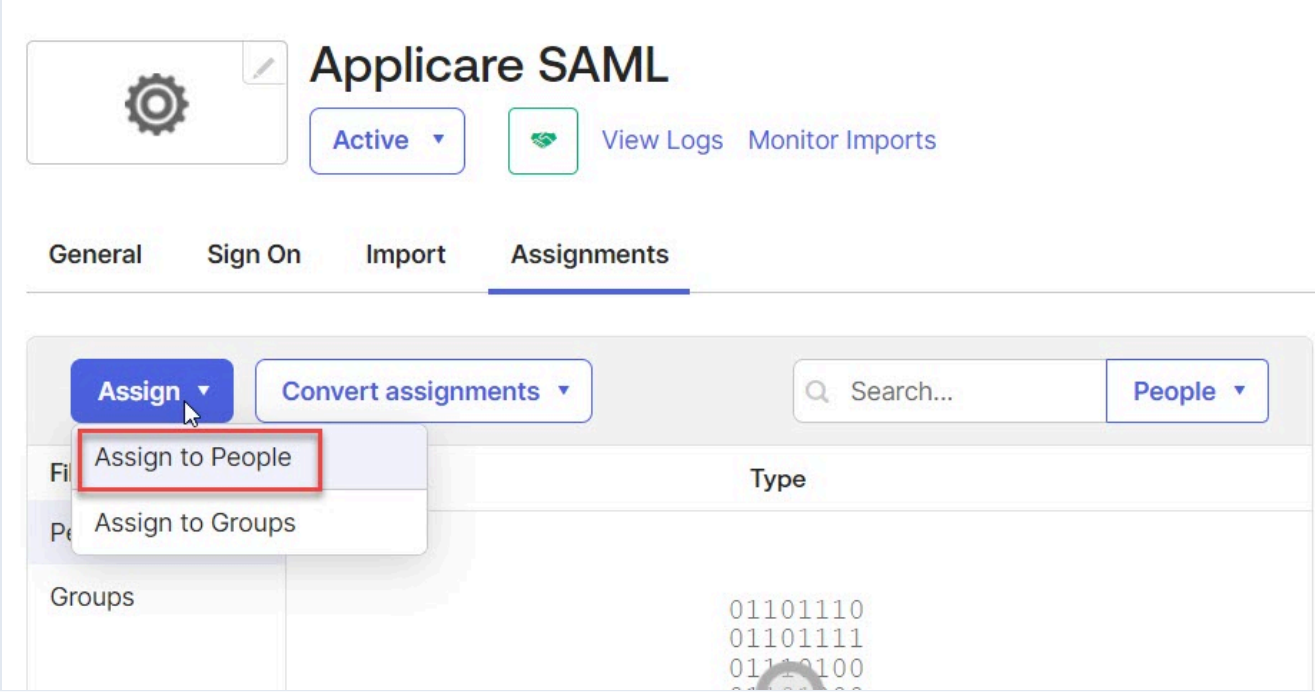


Assign Application to People

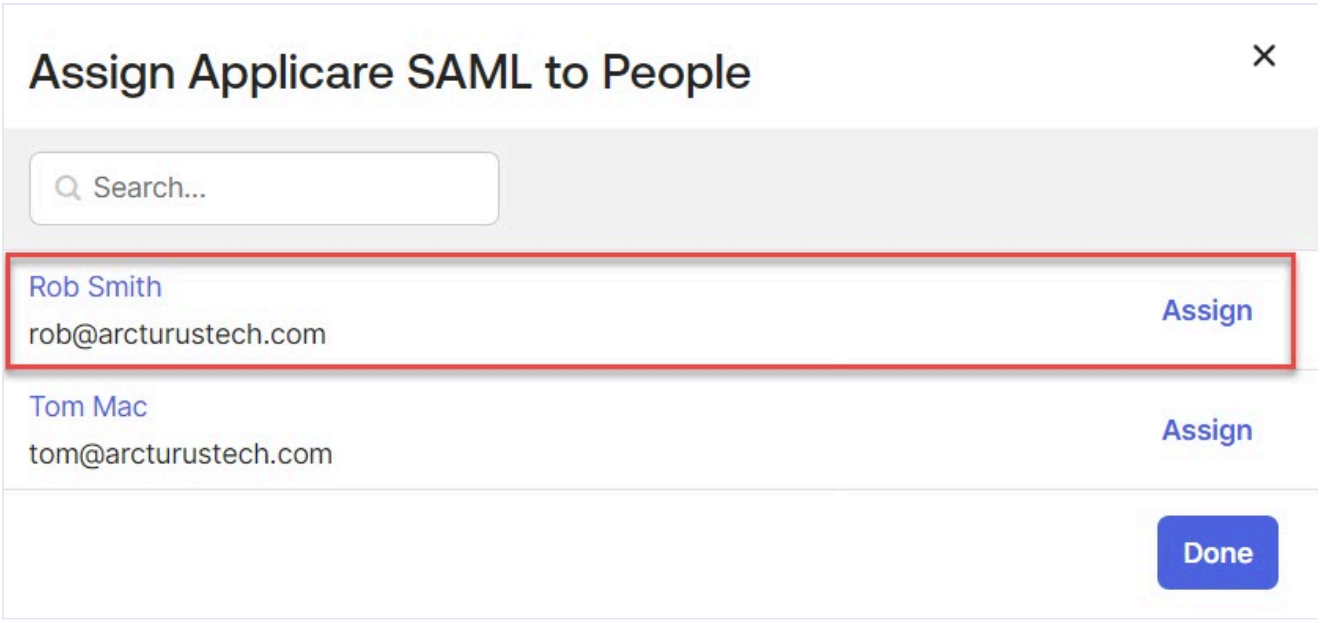
1. Go to **Applications** under the **Applications** menu and select **Applicare SAML**.



2. Select **Assign to People** option under the **Assign** dropdown.



3. Click the **Assign** button for the user which we are going to assign the App.



4. Click the **Save and Go Back** button.

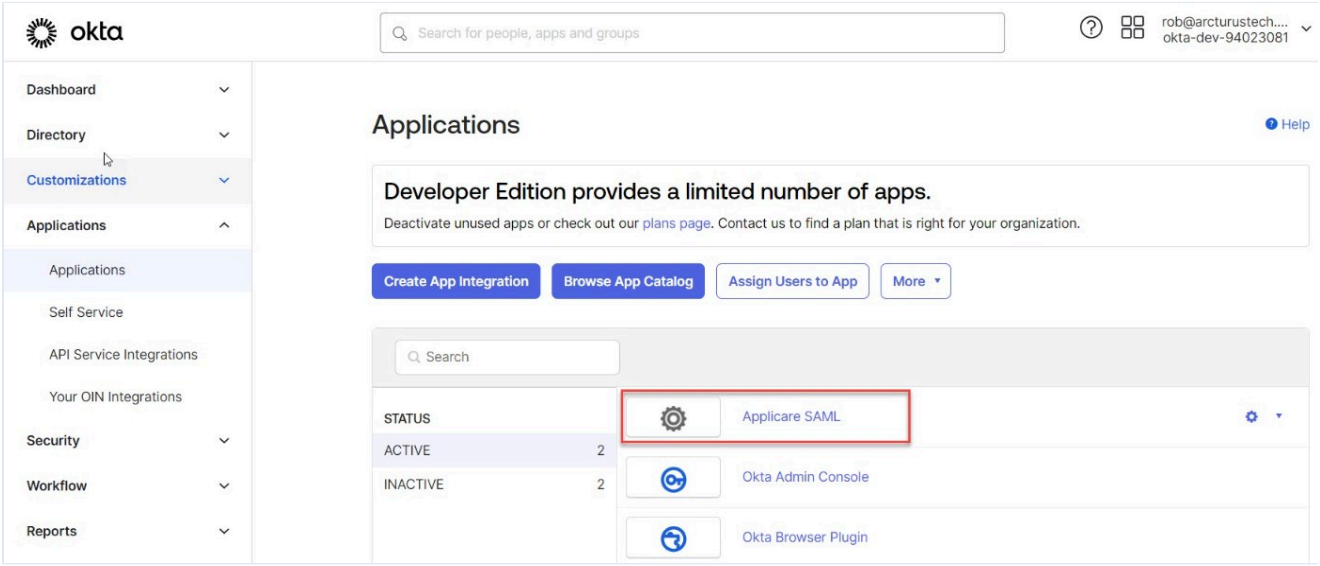


5. After assigning the users click the **Done** button.

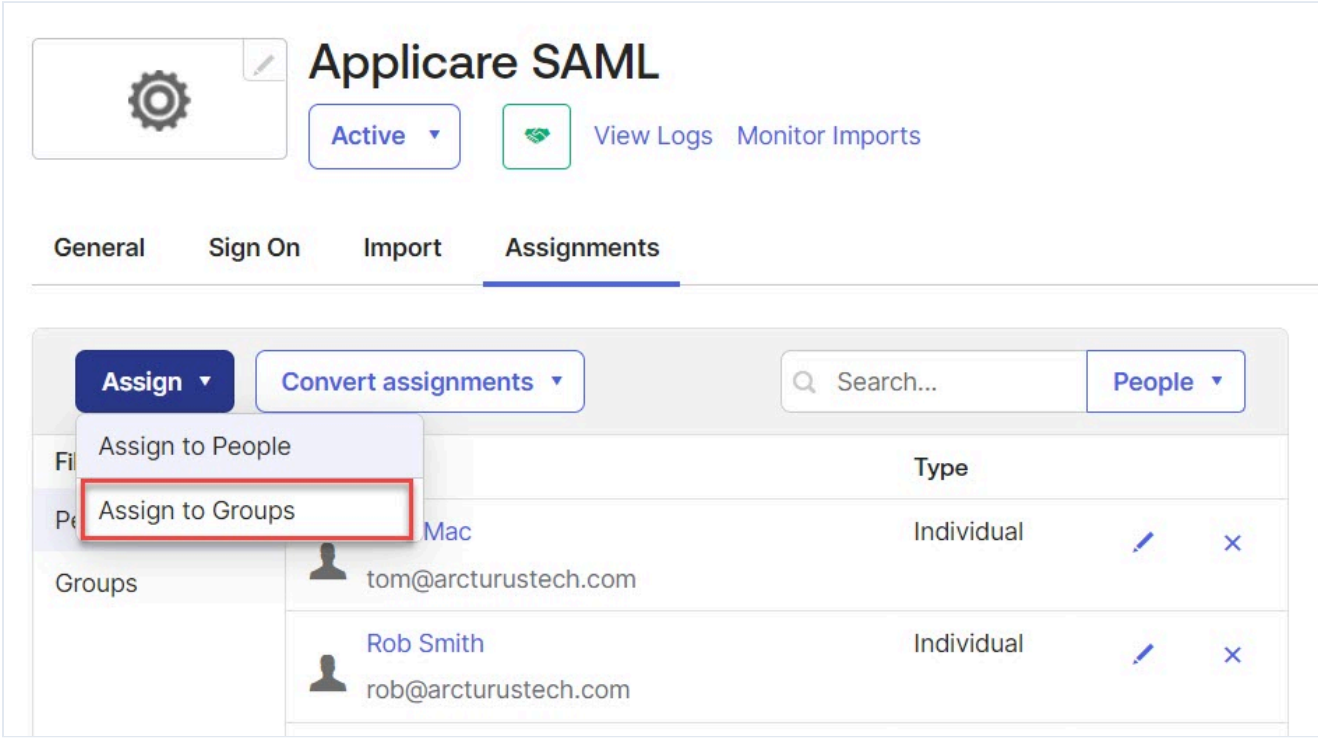


Assign Application to Group

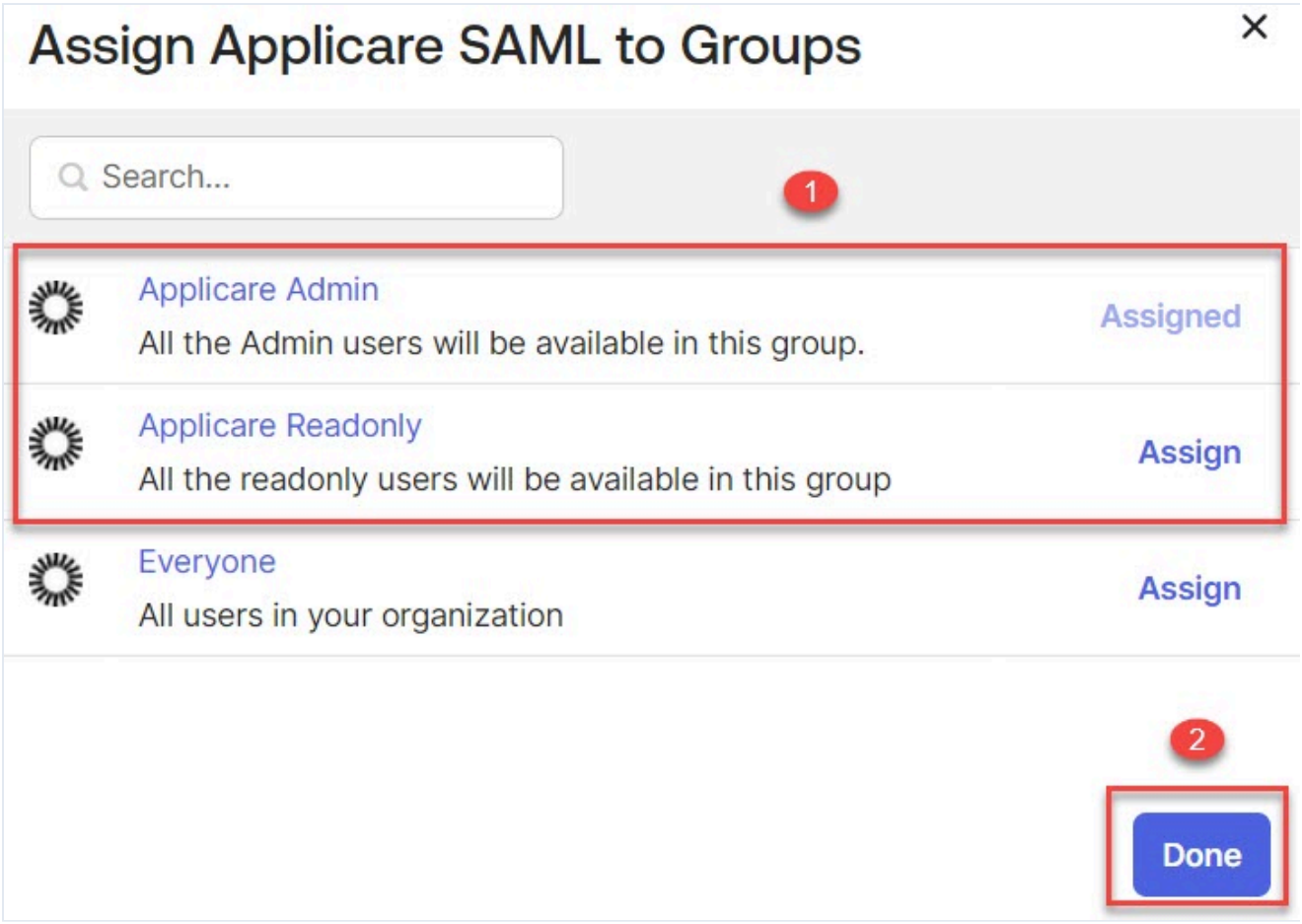
Navigate to **Applications** under the **Applications** menu and select **Applicare SAML**.



2. Choose **Assign to Groups** from the **Assign** dropdown.

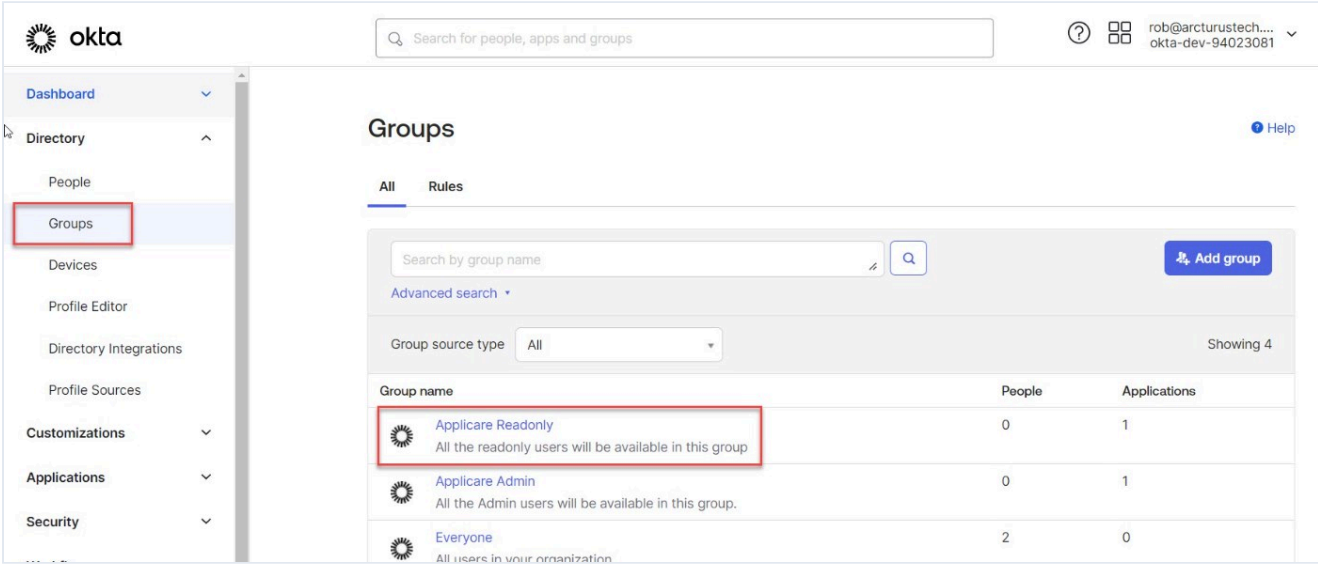


3. Click the **Assign** button for both the **Applicare Admin** and **Applicare Readonly** groups, then click the **Done** button.

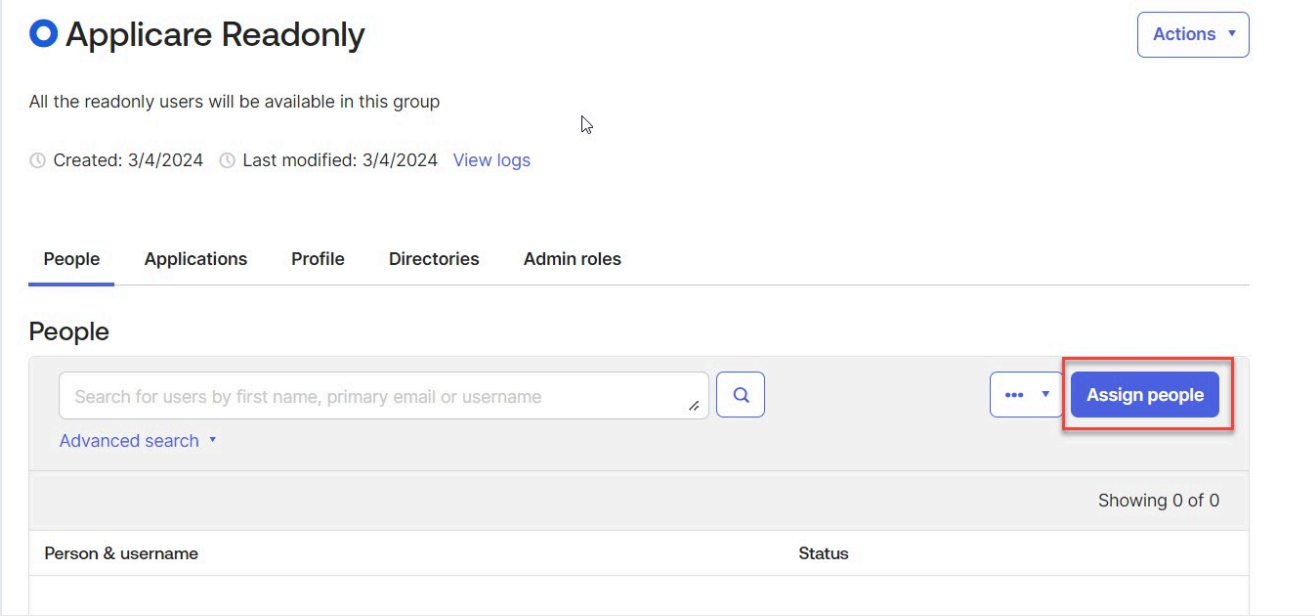


Assign People to Group

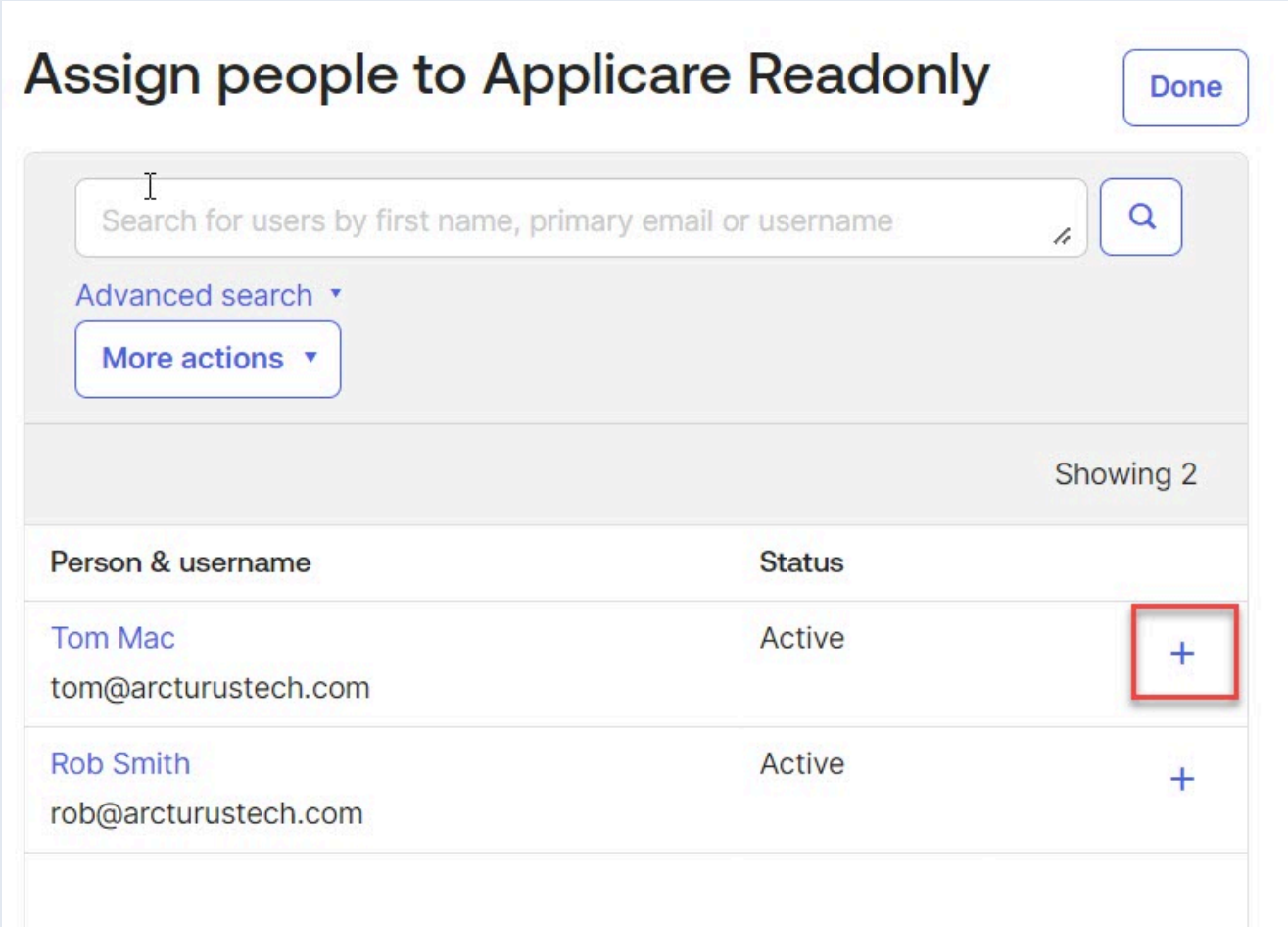
1. Go to **Groups** under the **Directory** menu and select the **Applicare Readonly** group.



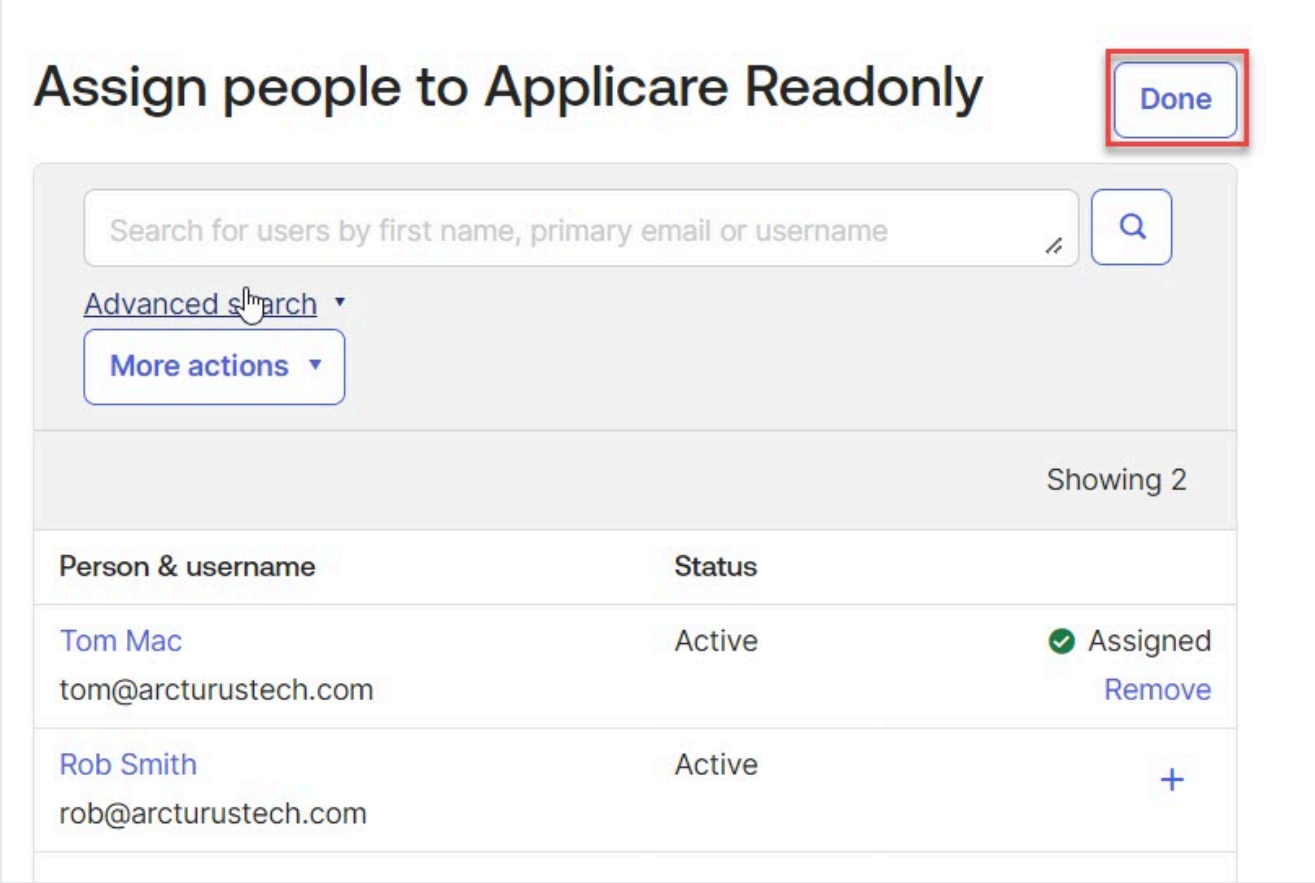
2. Click the **Assign people** button.



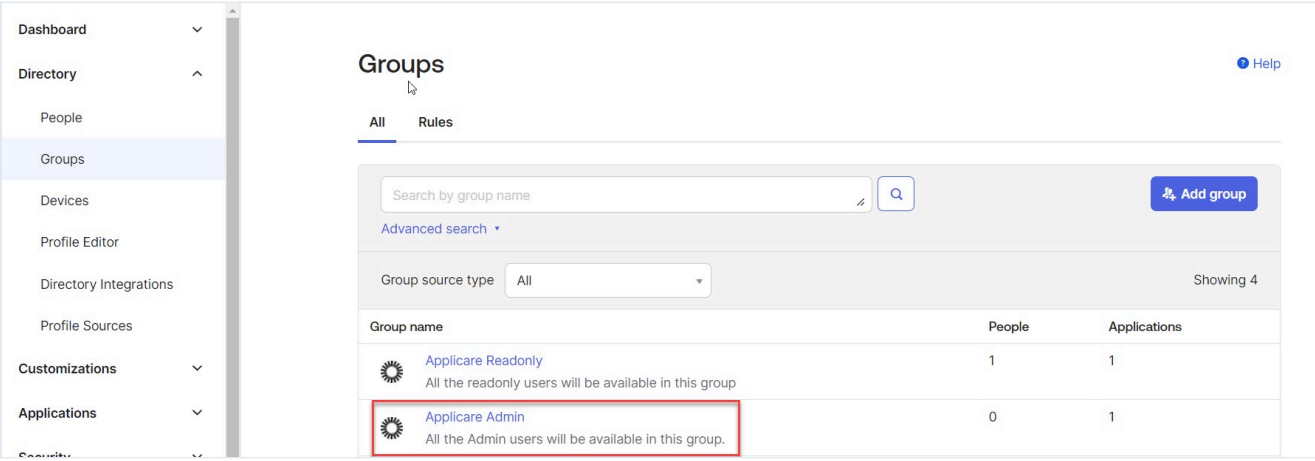
3. Click the **plus button** for the users which we need to assign that group.



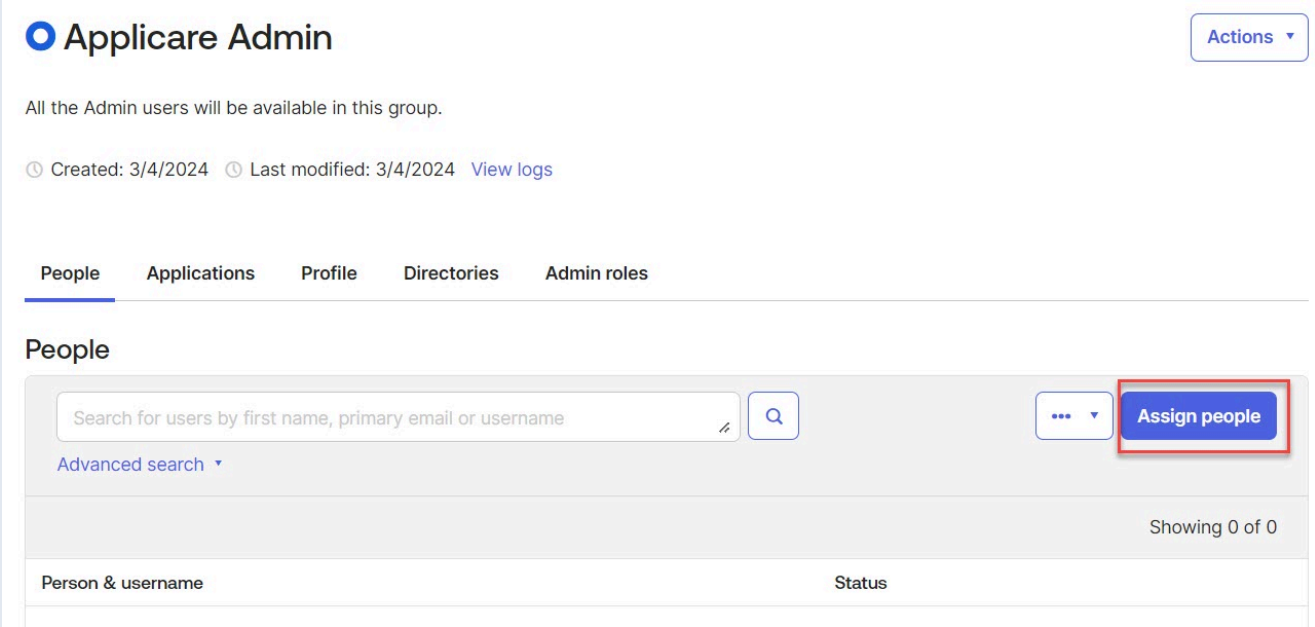
4. After assigning the users, click the **Done** button.



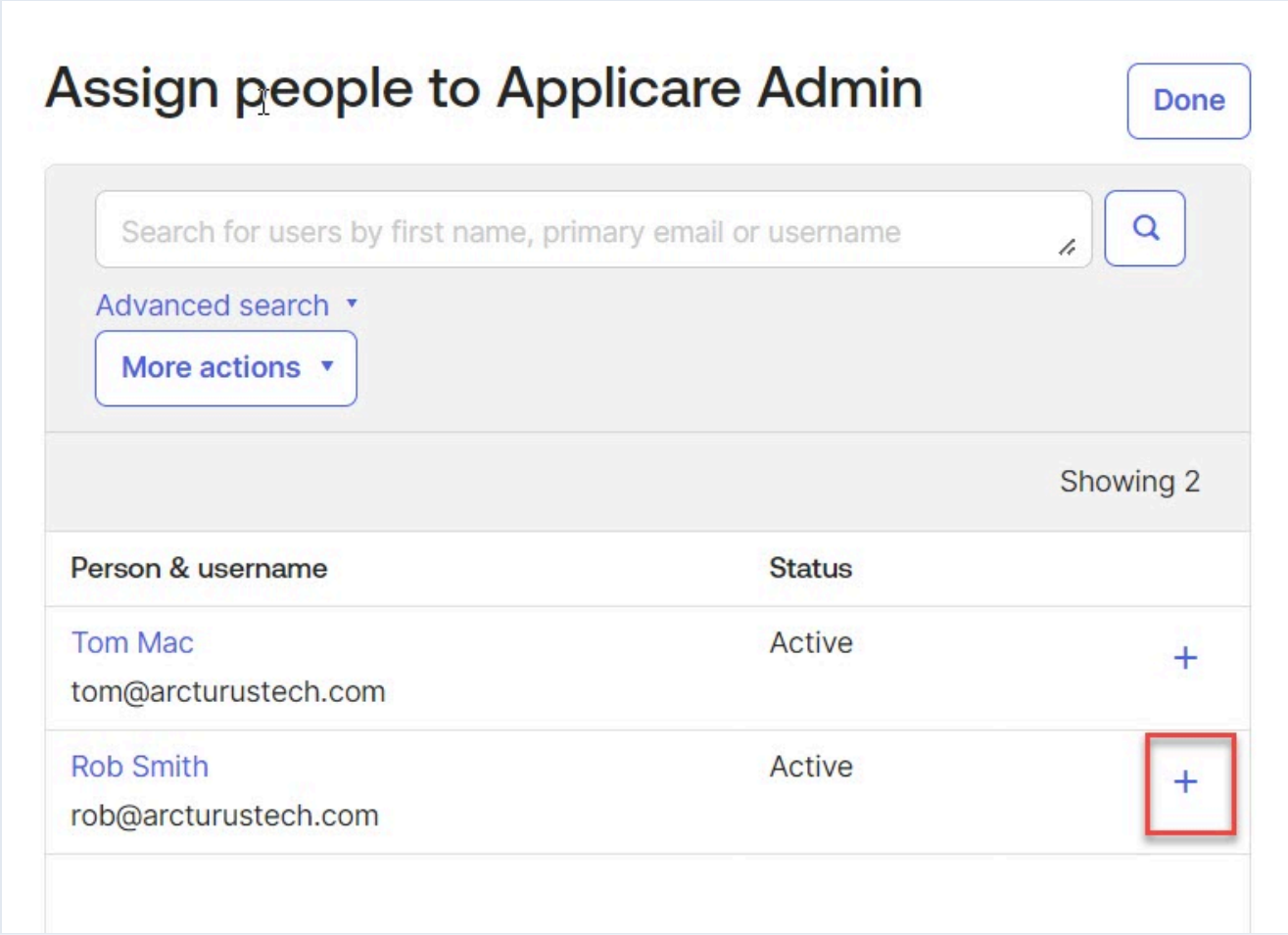
5. Go to **Groups** under the **Directory** menu and select the **Applicare Admin** Group.



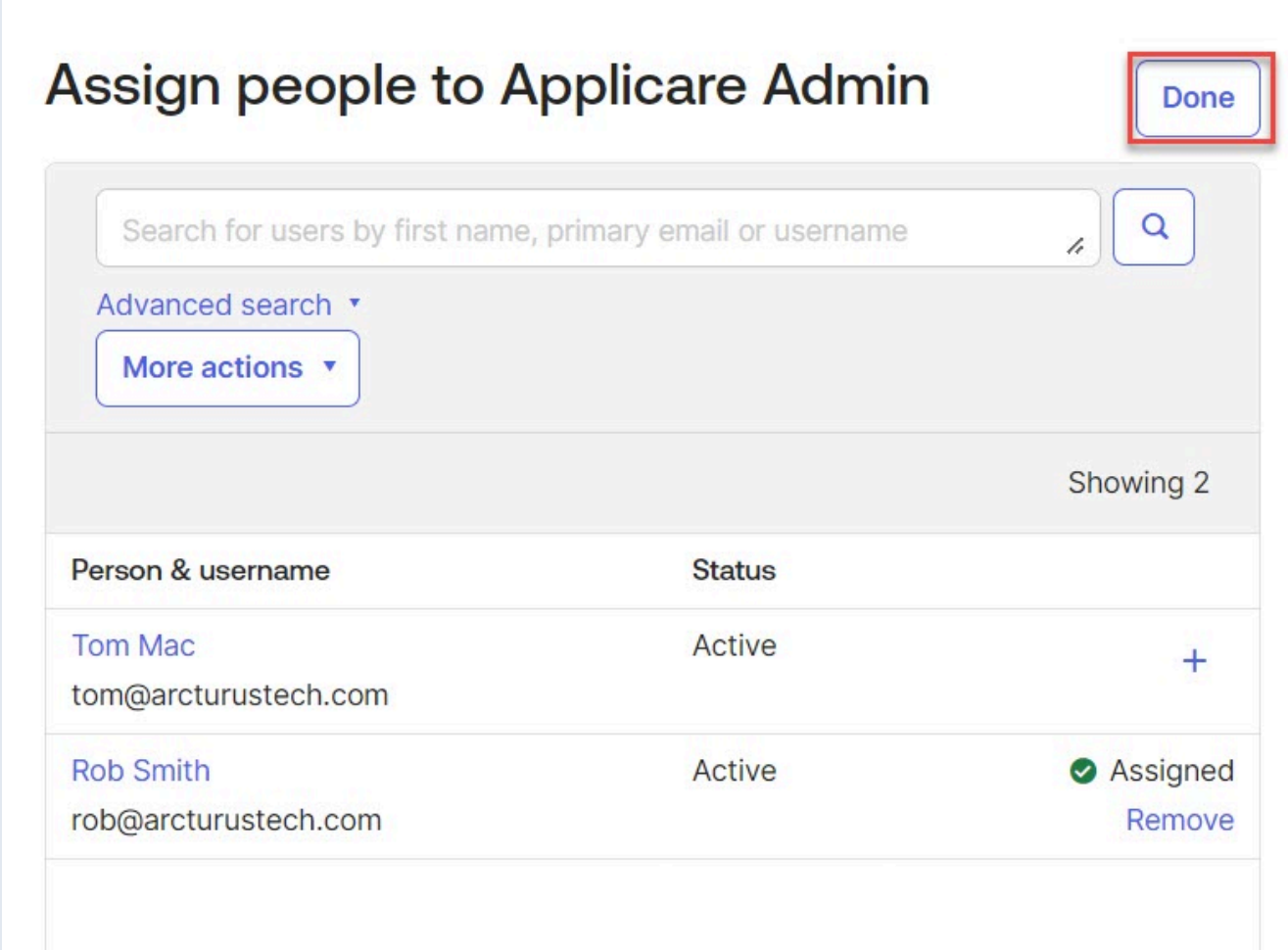
6. Click the **Assign people** button.



7. Click the **plus button** for the users which we need to assign that group.



8. After assigning the users click the **Done** button.



Configuration on Applicare Controller

Step 1: Add the following properties to the **ApplicareController.props** file.

saml.firstname.field=FirstName saml.lastname.field=LastName saml.email.field=EmailAddress saml.login.field=username saml.group.field=user.group applicare.saml.enableReadOnlyAccess=true

The **saml.login.field** is used to authenticate the SAML user for the Application. If the **username** attribute is not available, use any other available provided by the Application.

e.g.

saml.login.field=EmailAddress **(or)** saml.login.field=FirstName **(or)** saml.login.field=LastName

```
saml.firstname.field=FirstName
saml.lastname.field=LastName
saml.email.field=EmailAddress
saml.login.field=username
saml.group.field=user.group
applicare.saml.enableReadOnlyAccess=true
```

Step 2: Modify the Applicare server startup script located in **${arcturus_home/scripts/setAdminEnv.sh}** for Unix/Linux or **${arcturus_home/scripts/setAdminEnv.bat}** for Windows.

1. Change the **-Dspring.profiles.active=nosaml** to **-Dspring.profiles.active=saml** option to JAVA_OPT as shown here:

```
TEXT
-Dspring.profiles.active=saml
```

This is required for the Applicare server to start.

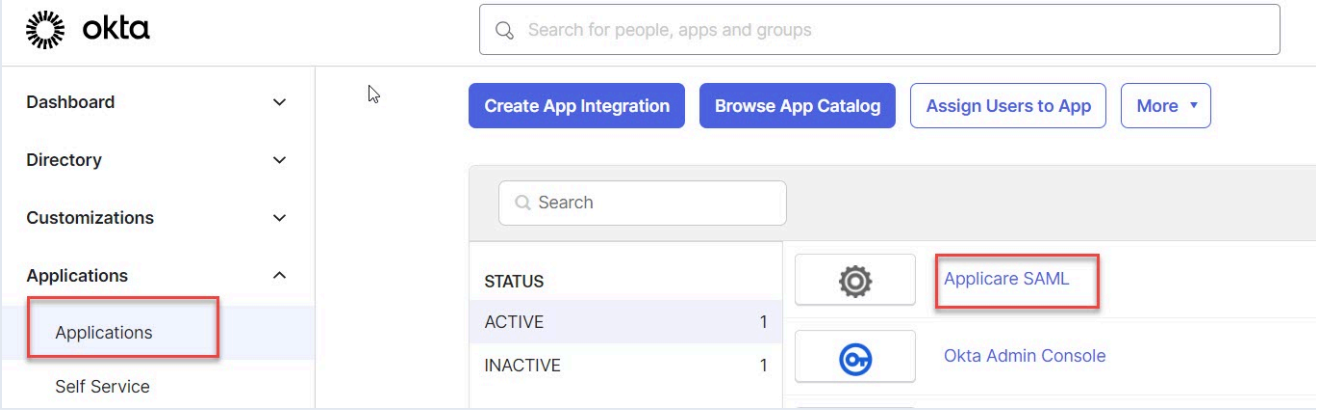
```
set JAVA_OPTS=-noverify -Darcturus_home=%ARCTURUS_HOME% -Dspring.profiles.active=saml
```

2. Configure Identity Provider Metadata XML

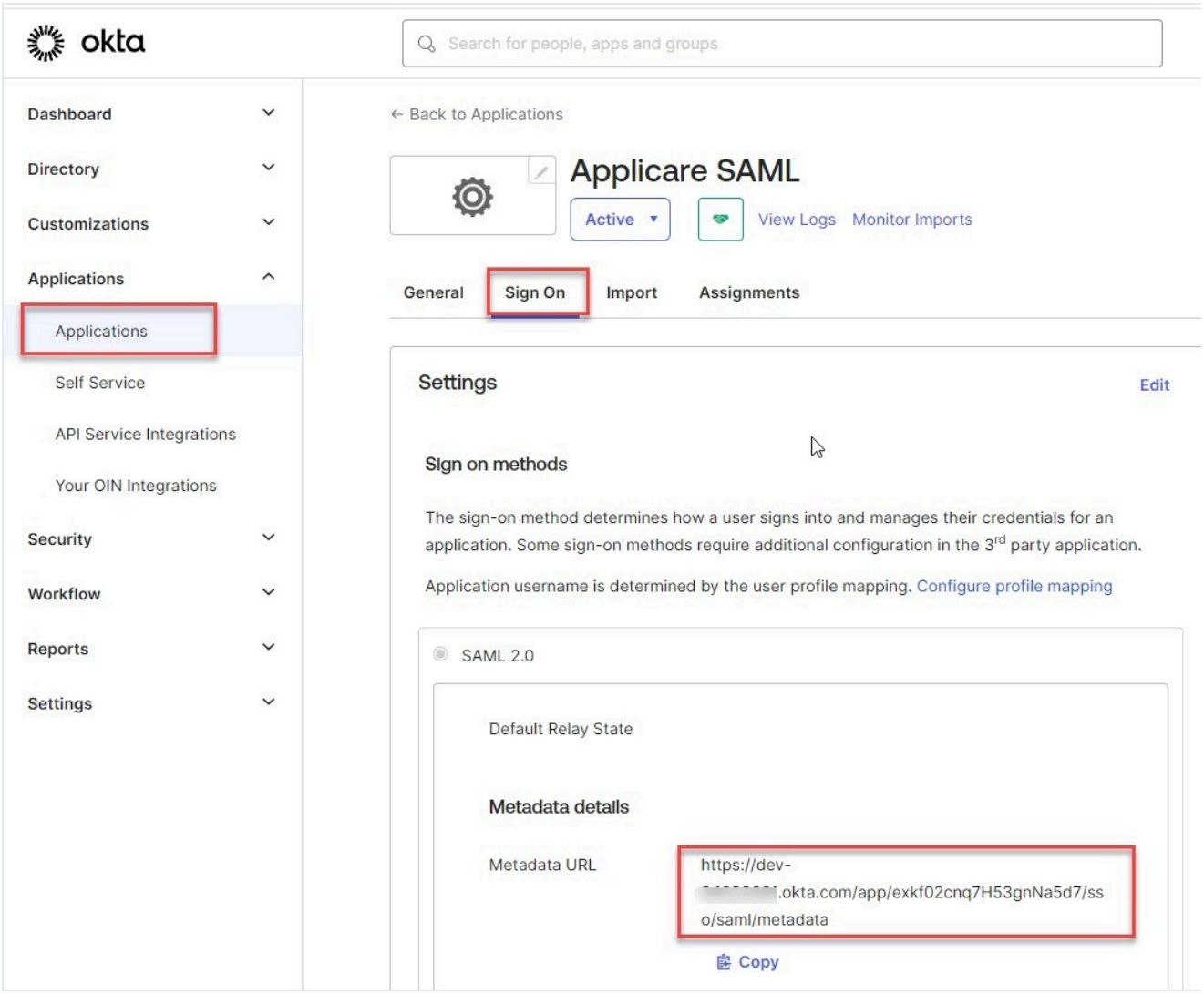
Obtain the URL from the Identity Provider.

To retrieve URL from OKTA:

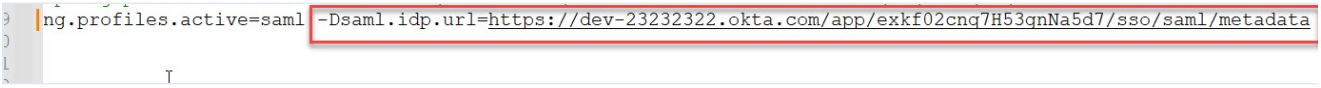
- **Login** in to **OKTA**.
- Select **Applications** from the left hand menu.
- Choose the **Applicare SAML** app.



- Select the **Sign On** tab.
- Copy the **Metadata URL** and replace it in the **<metadataurl>** in the **-Dsaml.idp.url** argument.

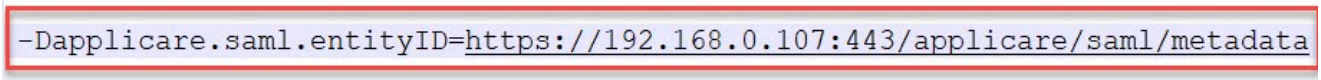



```yaml
-Dsaml.idp.url=<metadataurl>
Example:
-Dsaml.idp.url=https://dev-23232322.okta.com/app/exkf02cnq7H53gnNa5d7/sso/saml/metadata
```



3. Configure entity ID

```yaml
-Dapplicare.saml.entityID=https://{ApplicareIP}:{ApplicarePort}/applicare/saml/metadata
Example:
-Dapplicare.saml.entityID=https://192.168.0.107:443/applicare/saml/metadata
```



4. Set the entity base

```yaml
-Dapplicare.saml.entityBaseUrl=https://{ApplicareIP}:{ApplicarePort}/applicare
Example:
-Dapplicare.saml.entityBaseUrl=https://192.168.0.107:443/applicare
```



5. Set the Certificate file path

```yaml
-Dsaml.jksfile=Applicare controller SSL Certificate (.jks) file path
Example:
-Dsaml.jksfile=C:\sscertificate\certificate.jks
```



6. Set the KeyStore alias name

```yaml
-Dsaml.jksalias=ALIAS_NAME
Example:
-Dsaml.jksalias=applicare
```



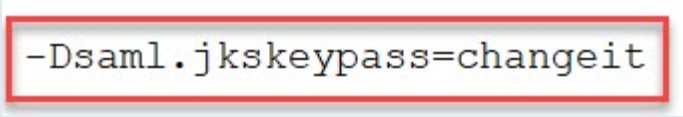
7. Set the KeyStore password

```yaml
-Dsaml.jkskeystorepass=Keystore password
Example:
-Dsaml.jkskeystorepass=changeit
```



8. Set the Key password

```yaml
-Dsaml.jkskeypass=Key password for alias
Example:
-Dsaml.jkskeypass=changeit
```



Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Manage LDAP User Roles

Applicare 9.2 · Updated September 13, 2024 · sso · **Pending technical review**

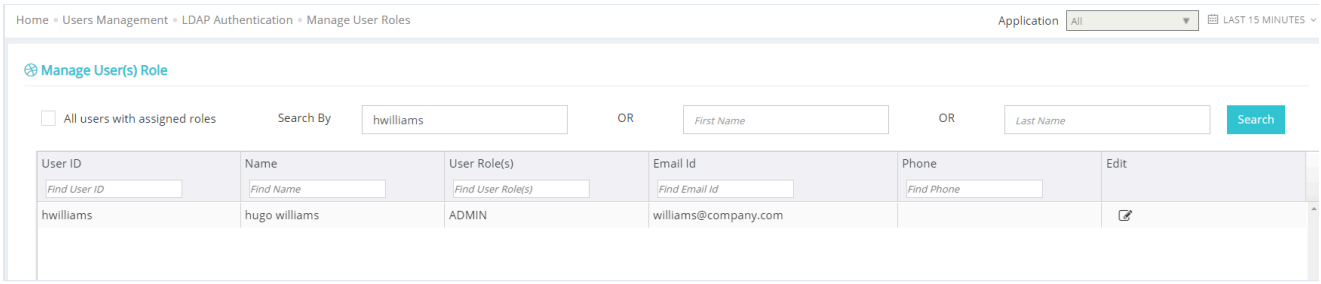
This page guides how to configure LDAP user Roles.

Search for LDAP users by username, first name or last name. The search will retrieve and displays all matching records on this page. You can click the **Edit** button to assign a role to a user. Applicare supports predefined roles such as **Admin** or **Readonly**. You must assign either of these roles to a user in order to provide Applicare access. If readonly access is enabled for all LDAP user, they will have readonly access by default. Only users assigned the admin role in Applicare will have admin access.

Applicare supports LDAP group-based access control. To enable this, create 'admin' and 'readonly' groups in LDAP and assigned them to the appropriate users.

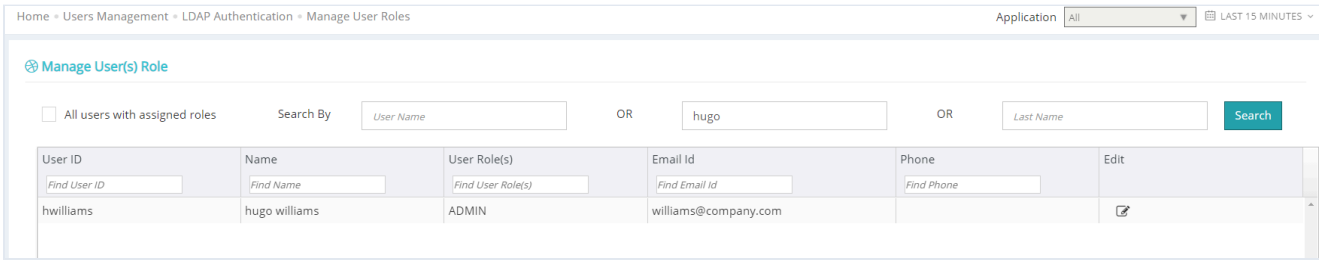
Search By Username

Searching by user name requires the full username and doesn't support wild characters.



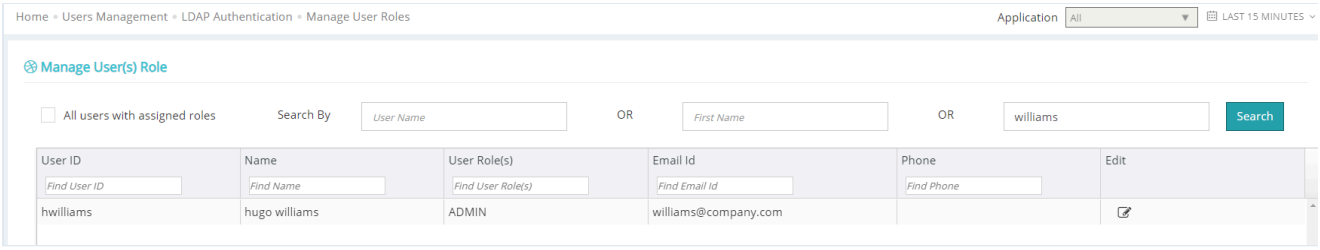
Search By First Name

Search by first name supports wildcard character and will retrieve all matching records. You do not need to include "*" at the end search of them.



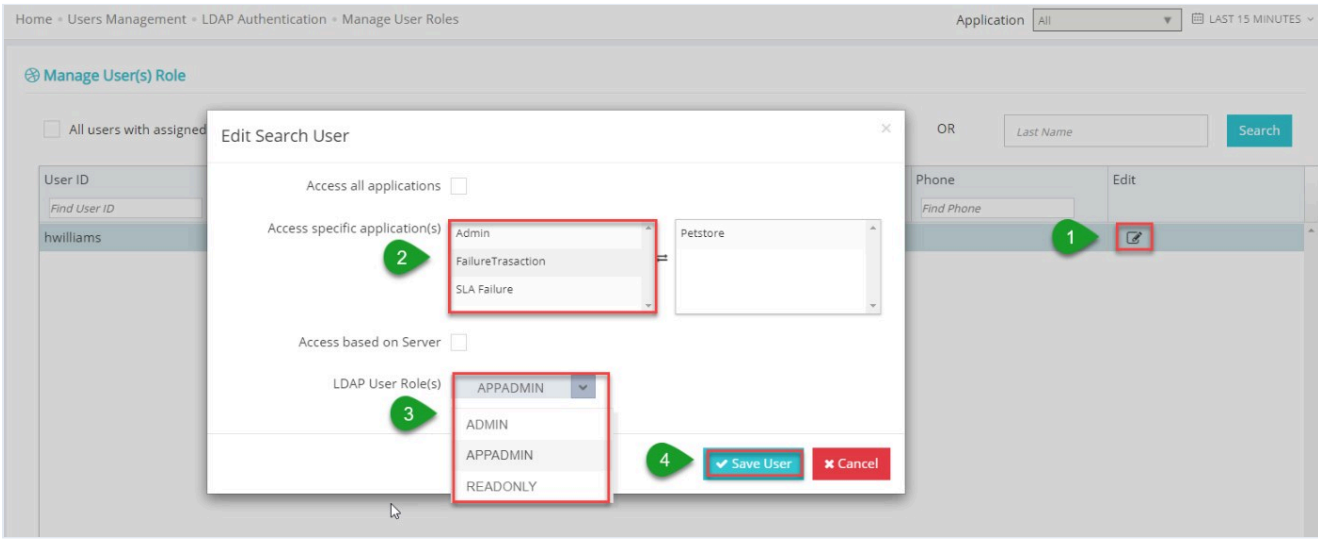
Search By Last Name

Searching by last name supports wildcard characters and retrieves all matching records. There is no need to add '*' at the end of the search term.



Save User Role

After locating the user, click the "Edit" button and assign role to user on the following screen.



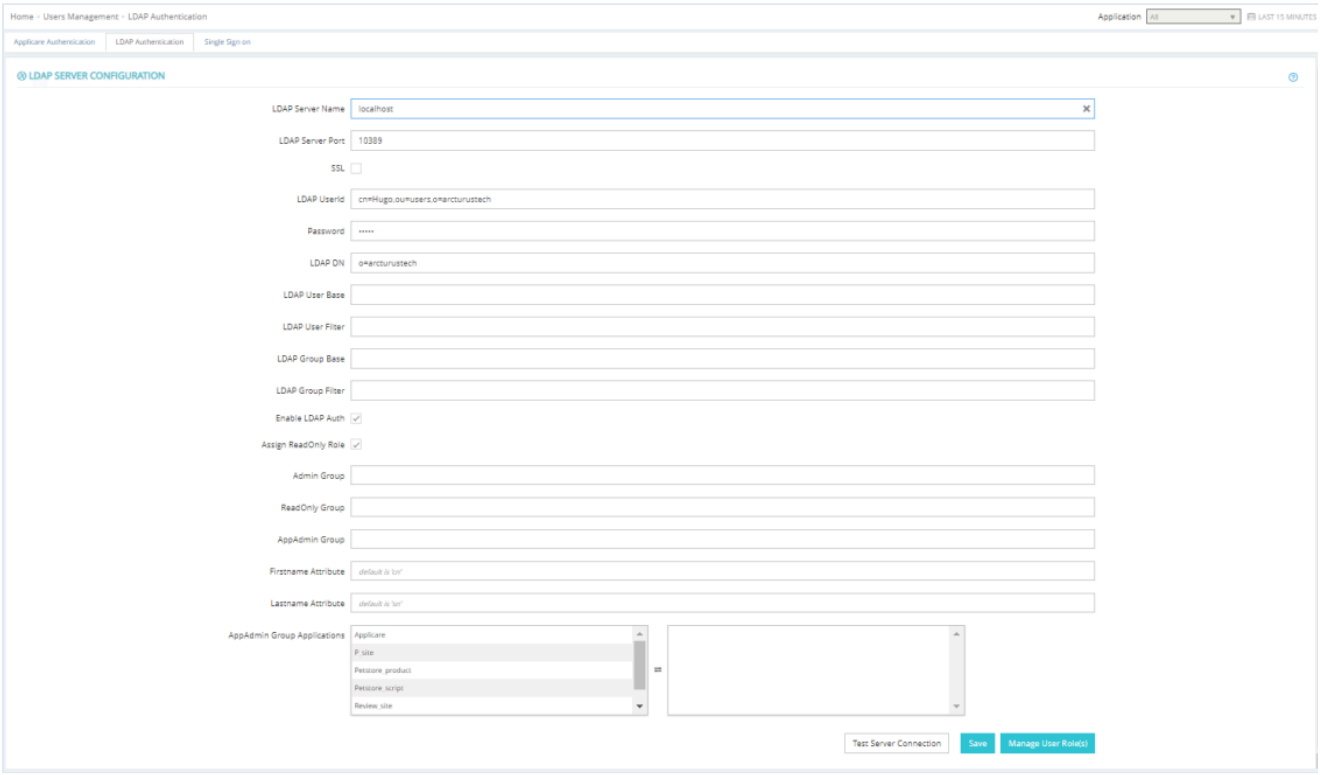
Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Active Directory/LDAP Settings

Applicare 9.2 · Updated September 13, 2024 · sso · **Pending technical review**

Go to Configuration -> LDAP Server Configuration to set up LDAP server settings and enable LDAP authentication.



If SSL is enabled, you need to install LDAP SSL certificate on the Applicare server.

The DN (Distinguished Name) is the base directory folder in Active Directory where organizational users and groups are defined. It is required field.

The User OU (Organization Unit) is the base directory where user details are configured (Optional). If you are familiar about AD query formats, you can specify the filter otherwise, you can leave it blank (Optional).

The Group OU (Organization Unit) is the base folder where user group details are configured (Optional). If you are familiar with AD query formats, you can specify the filter; Otherwise, you may leave it blank (Optional).

Enable LDAP Auth to allow users to log into Applicare using their LDAP credentials. If it is not enabled, user authentication will be handled using Applicare database.

Assign the Readonly role to grant default read-only access to all the LDAP users if read-only access is enabled. You can override this default and assign different roles as needed. Click "Manage Role" to assign the appropriate Applicare role to an LDAP user.

Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Role of Single Sign-On Users

Applicare 9.2 · Updated September 13, 2024 · sso · **Pending technical review**

Applicare provides granular access control to the resources it monitors. A user can be assigned either full control or read-only access to a specific application, or to a single server. This way each Application administrator has access to their application resources and are not distracted by other application alerts.

Applicare Users

Applicare comes with two predefined users in its default configuration.

Super Admin ('admin')

The Super Admin user has a fixed ID of "admin" and possesses full access to all features in Applicare. The default password for this account is 'admin', which should be changed after installation is complete.

Guest ('guest')

In the default configuration, there is a guest user with ID 'guest'. The Default password for this account is 'guest'. This user has a read-only access and will must be granted access to specific applications or servers.

Applicare User Roles

User access in Applicare is managed by assigning specific roles. The following roles can be assigned in Applicare.

Admin

The Admin role grants the user with full of rights to create, edit and remove users, as well as manage their access. However, this role does not permit modifications to or removal of the Super Admin's privileges.

App Admin

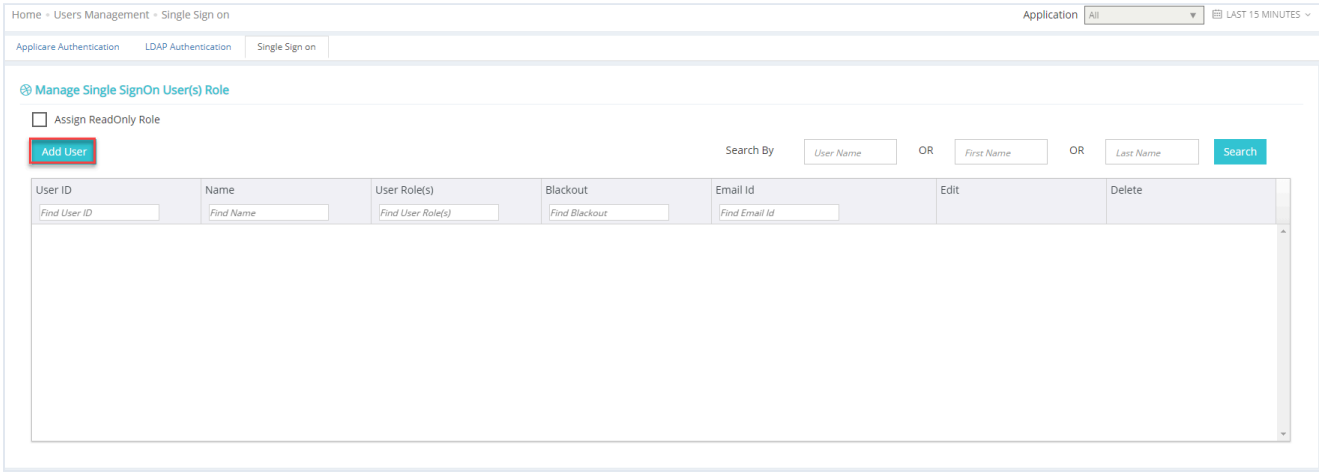
The App Admin role allows users to administer and manage a specific applications. Users assigned this role are designated by the Super Admin or Admin to oversee particular applications.

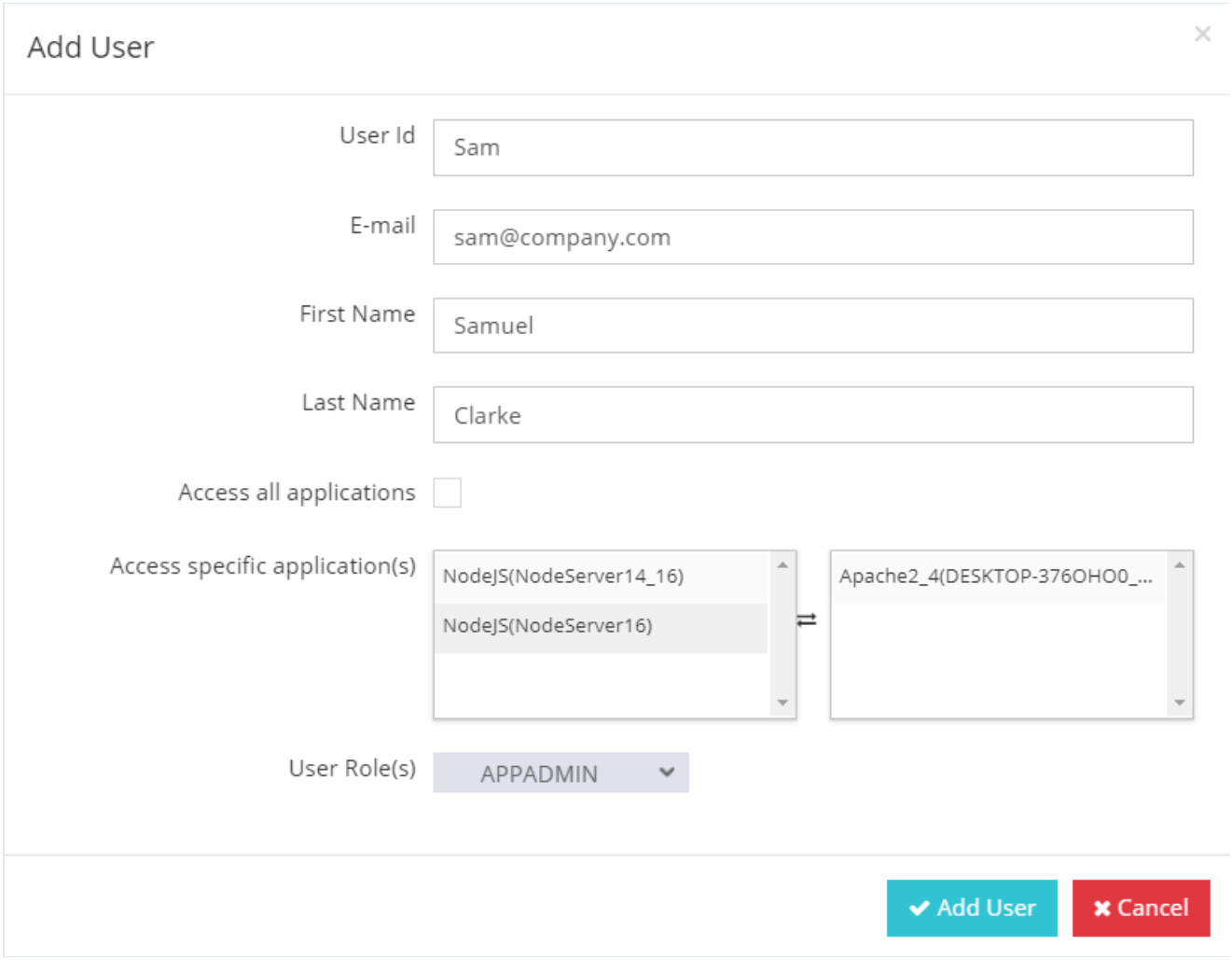
Read only

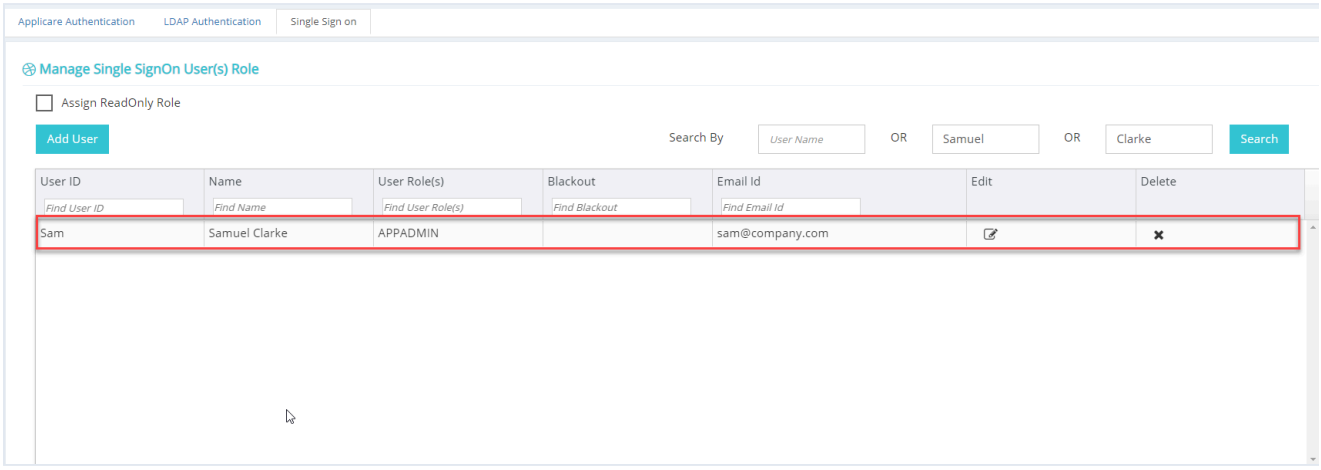
Read only role grants the user read only access and they can't make changes to monitoring.

Adding a user

Creating a new Applicare user is straightforward. Enter the user's details, select the applications they should access and assign the appropriate role. Application for access control are application created in Applicare when adding app servers or server Agents to the system. These are NOT the .ear or .war applications deployed on app servers.


Home • Users Management • Single Sign on
Application
All
LAST 15 MINUTES
Applicare Authentication
LDAP Authentication
Single Sign on
Manage Single SignOn User(s) Role
Assign ReadOnly Role
Add User
Search By
User Name
OR
First Name
OR
Last Name
Search
User ID
Name
User Role(s)
Blackout
Email Id
Edit
Delete
Find User ID
Find Name
Find User Role(s)
Find Blackout
Find Email Id


Add User
User Id
Sam
E-mail
sam@company.com
First Name
Samuel
Last Name
Clarke
Access all applications
Access specific application(s)
NodeJS(NodeServer14_16)
NodeJS(NodeServer16)
Apache2_4(DESKTOP-376OHO0_...
User Role(s)
APPADMIN
Add User
Cancel



Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

User Role Management

Applicare 9.2 · Updated September 13, 2024 · sso · **Pending technical review**

Applicare allows fine grain access control to the resources that are monitored by Applicare. A user can be given either full control or readonly access to a particular application consisting of many many servers in different geographic locations or a particular server. This way each Application administrator has access to their application resources and are not distracted by other application alerts.

Applicare Users

Applicare out of the box configuration has two pre defined users.

Super Admin ('admin')

The Super Admin user's id is fixed - admin. Super Admin user has all permissions to access every feature in Applicare. Default password for this account is 'admin'. You should change this password after installation is complete.

Guest ('guest')

In out of the box configuration, a guest user exists with id - guest. Default password for this account is 'guest'. This user is a readonly user and will need to be assigned access to specific applications/servers.

Applicare User Roles

Applicare user's access is controlled by assigning them a particular role. Following roles can be assigned in Applicare

Admin

The Admin role grants the user all rights to create/edit/remove users and control other user's access. Admin role just doesn't allow modifing or removing Super Admin's privileges.

App Admin

App Admin role grants the user access to administer and manage a particular application. Users with App Admin role are assigned application/s to manage by Super Admin or Admin

Read only

Read only role grants the user read only access and they can't make changes to monitoring.

Adding a user

Adding a new Applicare user is fairly simple. Fill in user's details, select the applications and assign the role. Application for access control are application created in Applicare at the time of adding the app server/server agents to Applicare. These are NOT the .ear or .war applications deployed on app servers.

Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Email/SMTP Configuration in Applicare

Applicare 9.2 · Updated September 13, 2024 · alerts · **Pending technical review**

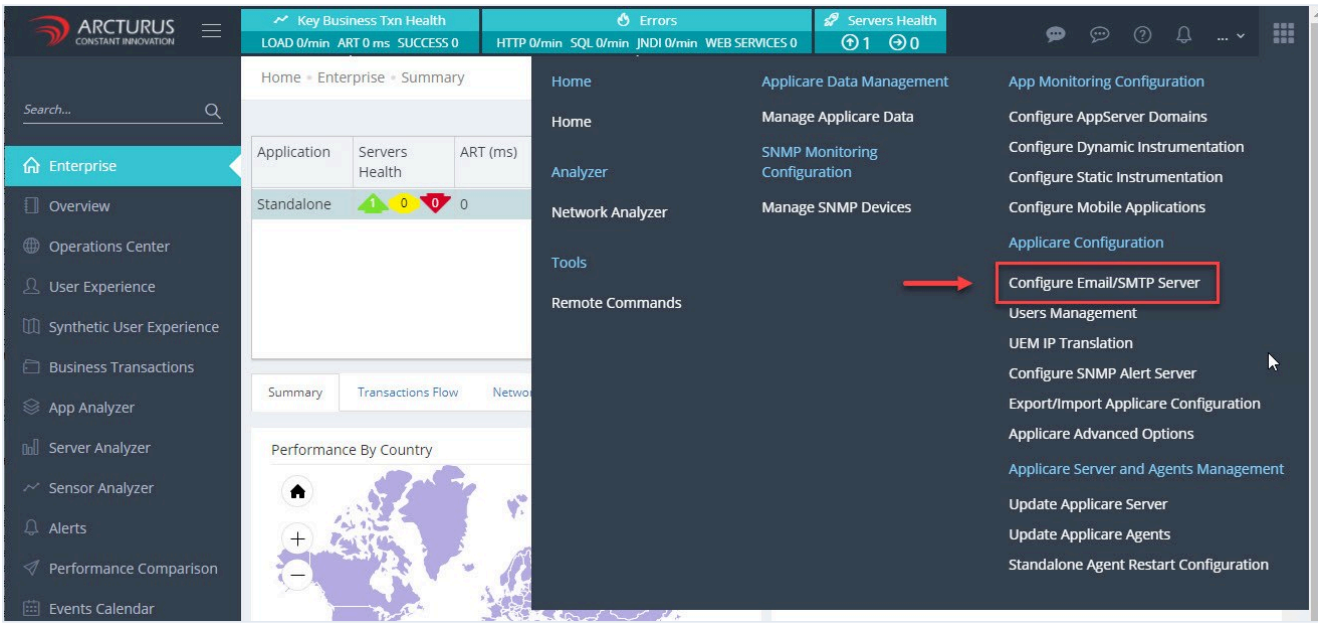
An app password must be created before setting up mail in Applicare.

Steps to Create an App Password for Gmail Here

Steps to Create an App Password for yahoo mail Here

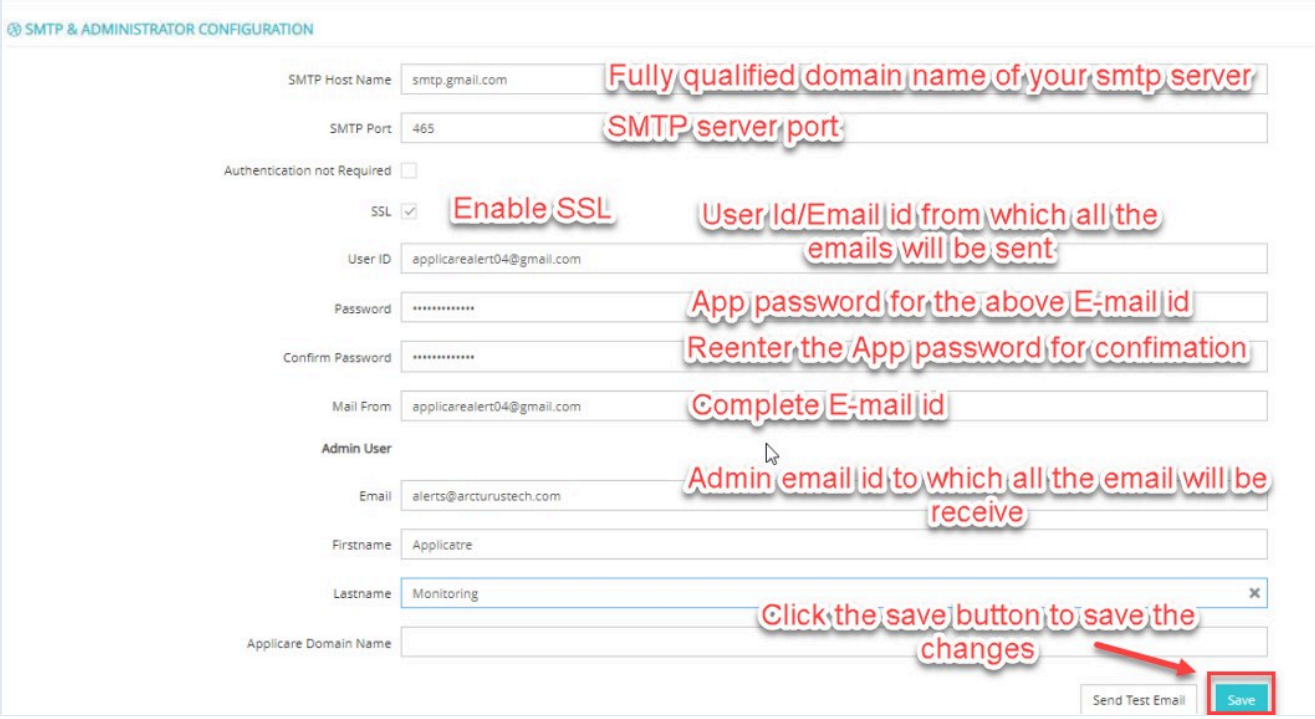
To enable email and alert configuration in Applicare, you need to set up the SMTP server details in the **SMTP & ADMINISTRATOR CONFIGURATION** dashboard.

In Applicare dashboard, navigate to the top menu on the right and click on **Configure Email/SMTP Server** under **Applicare Configuration** section.



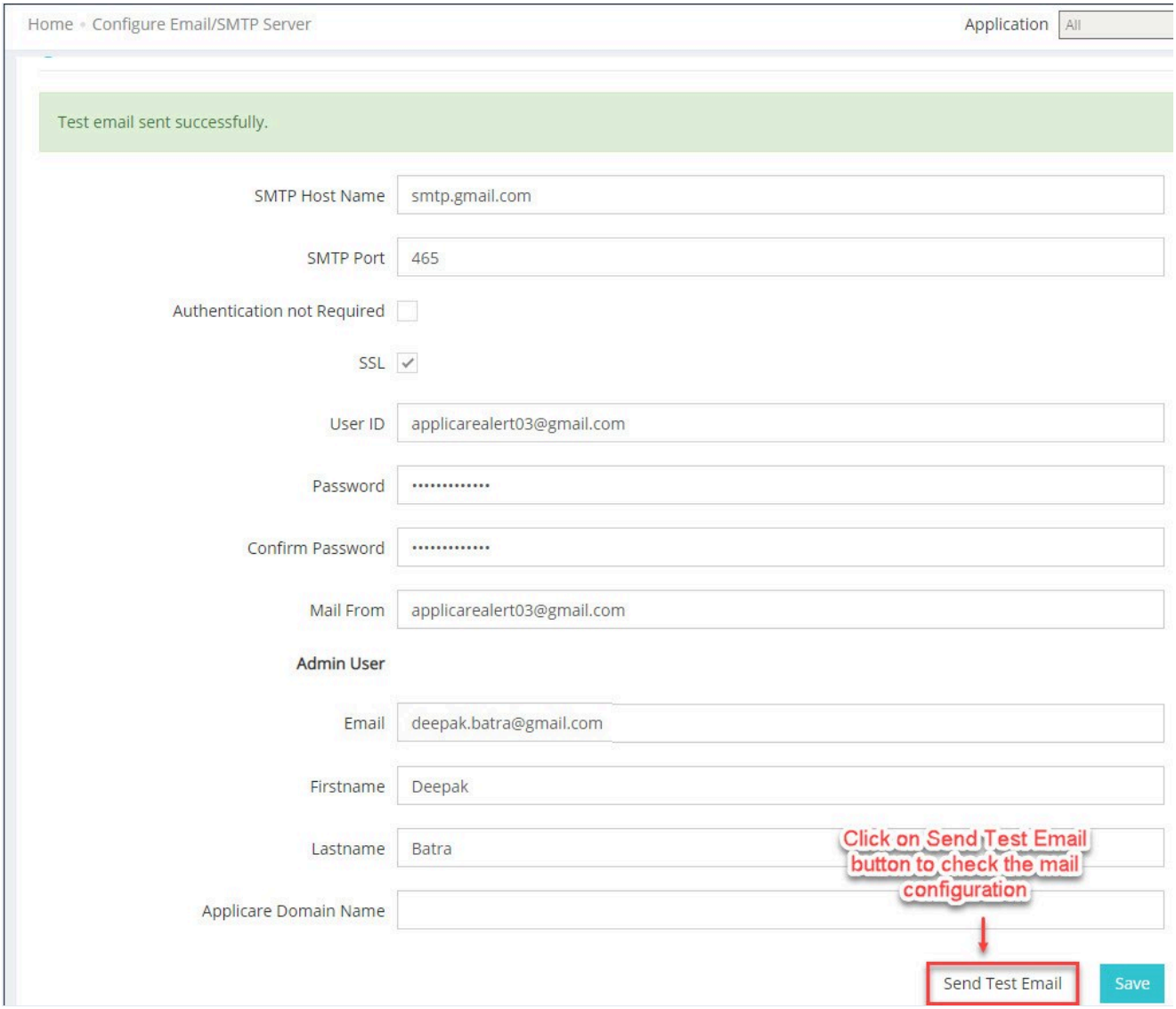
In **SMTP & ADMINISTRATOR CONFIGURATION** dashboard, enter the SMTP server details.

Here is an example of the **SMTP server details for Gmail:**



You can add multiple admin email addresses, separated by commas.

After entered the details, click the **Send Test Email** button to verify the configuration.



Monitoring with Applicare

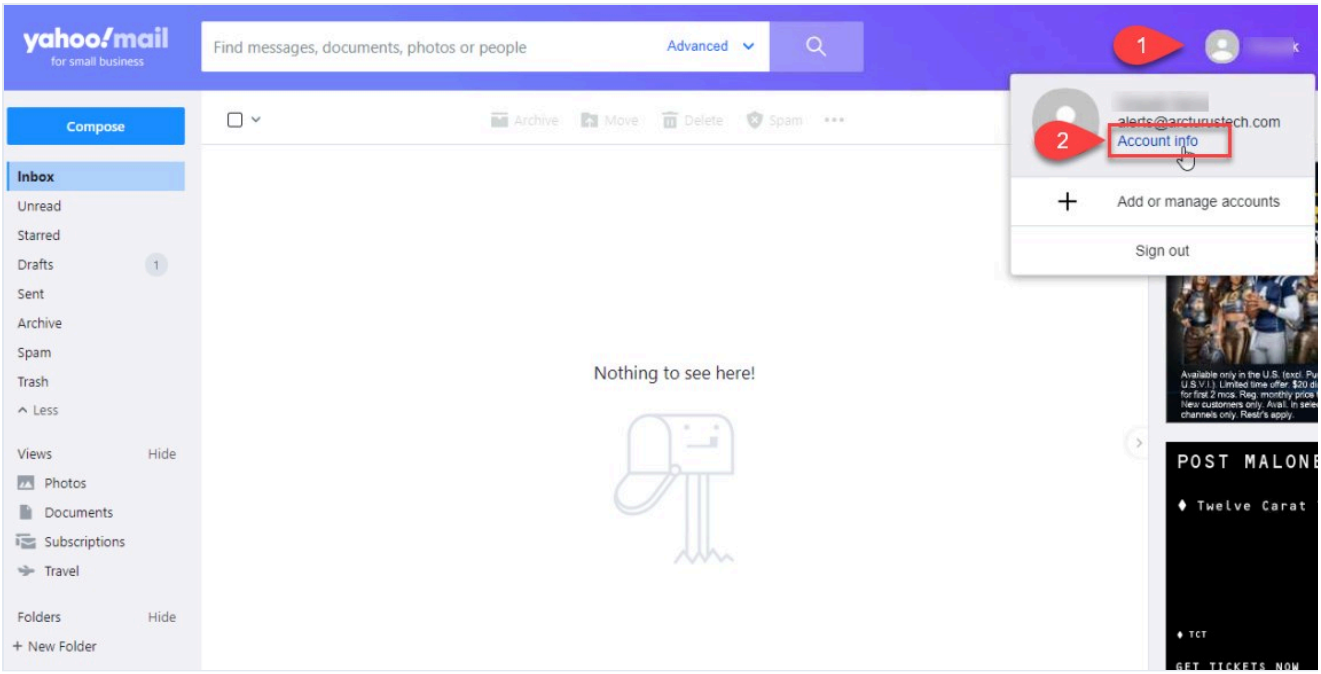
CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Set Up SMTP for Yahoo Mail

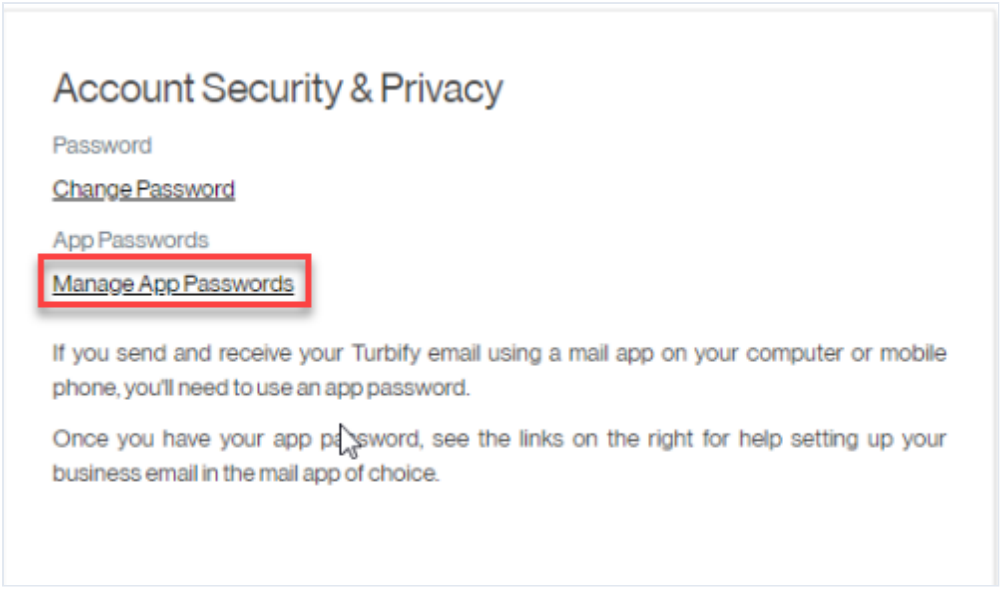
Applicare 9.2 · Updated September 13, 2024 · alerts · **Pending technical review**

Step 1: Log in to your Yahoo mail account.

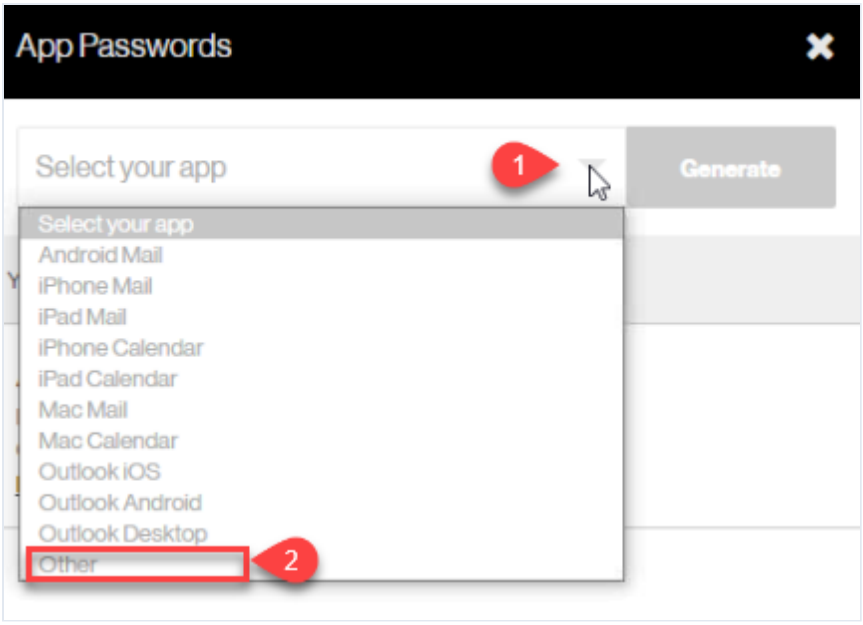
Step 2: Click on your Yahoo account icon and then select **Account info.**

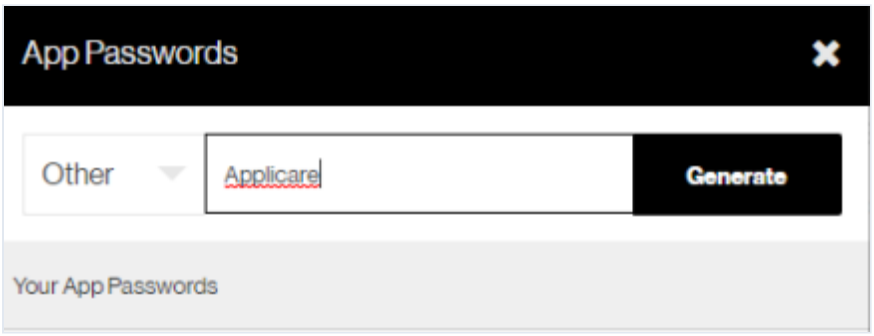


Step 3: Click on **Manage App Passwords** option under the **Account Security & Privacy section.**



Step 4: A popup window for app will appear, Click on **Other** from the **select your app** dropdown menu, enter "Applicare" in the text box and click the **generate** button. Once the app password is generated, copy it and save it securely.





Monitoring with Applicare

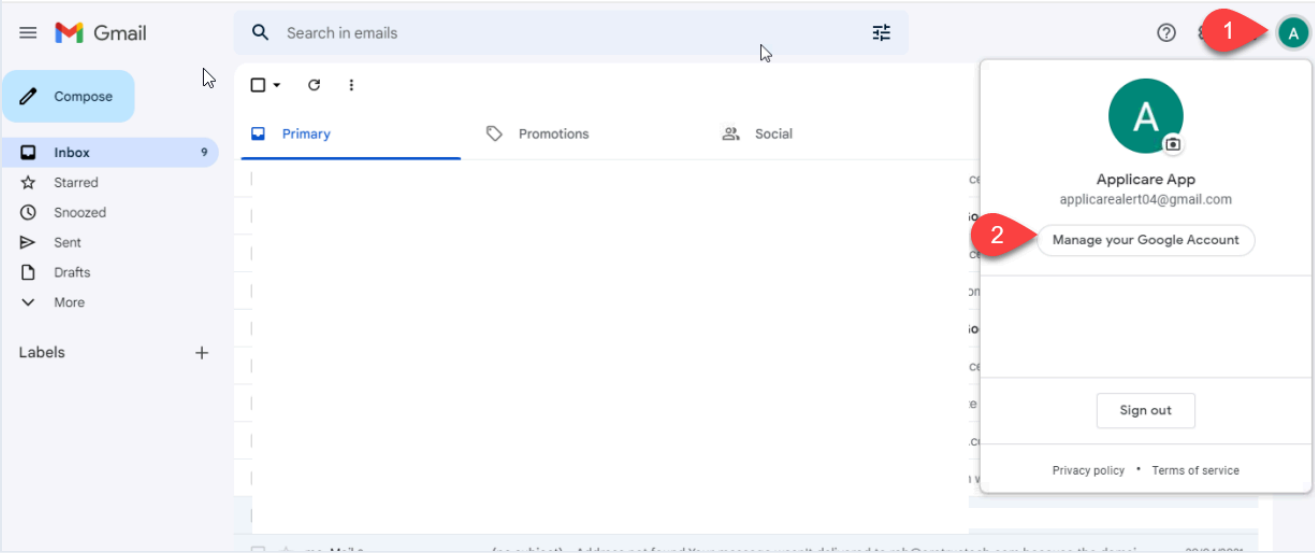
CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Configure SMTP for Gmail

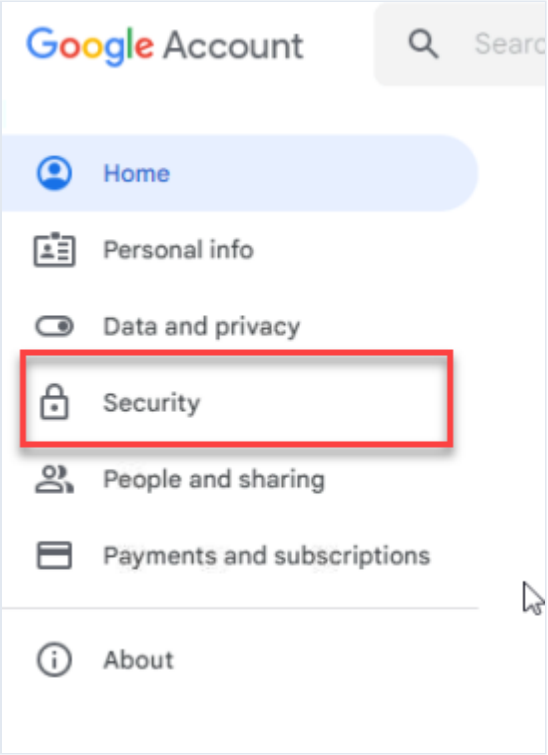
Applicare 9.2 · Updated September 13, 2024 · alerts · **Pending technical review**

Step 1: Log in to your **Gmail** account.

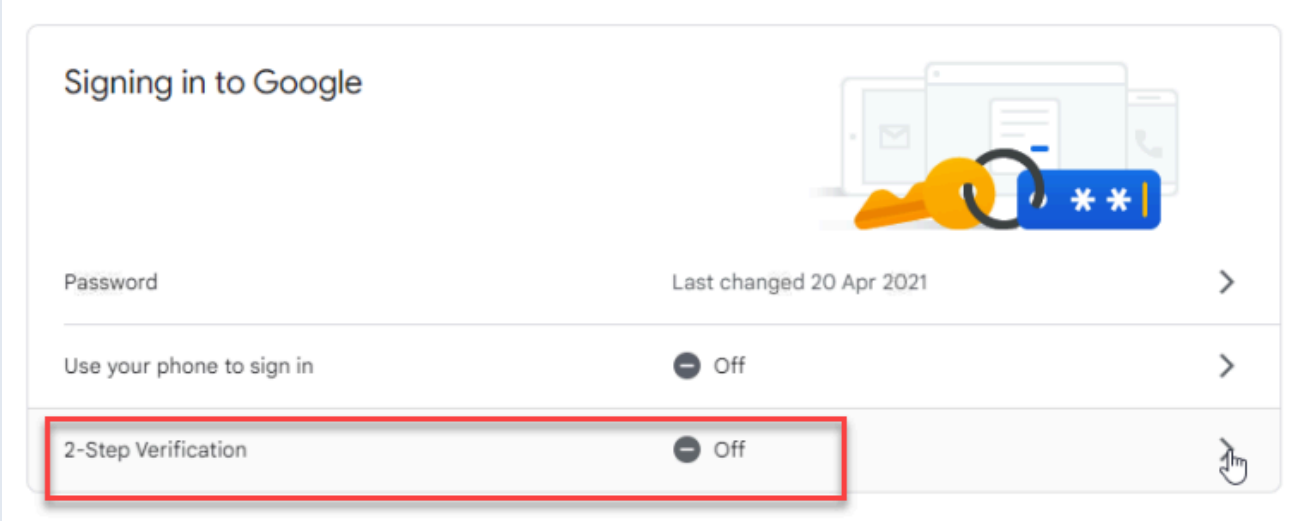
Step 2: Click on your Google account icon and select **Manage your Google Account**.



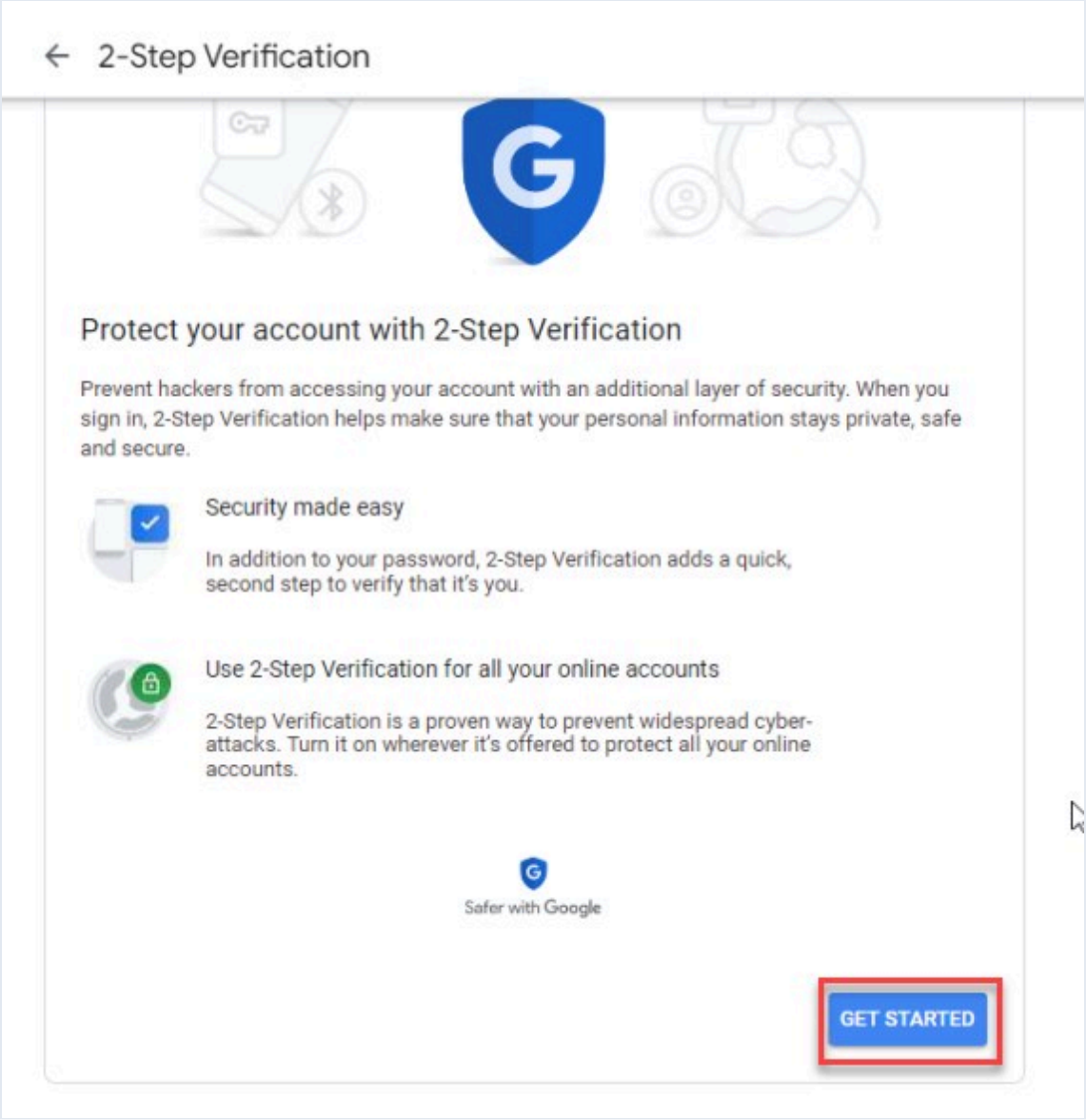
Step 3: Click on the **Security** option on the left side.



Step 4: Click on **2-Step Verification.**

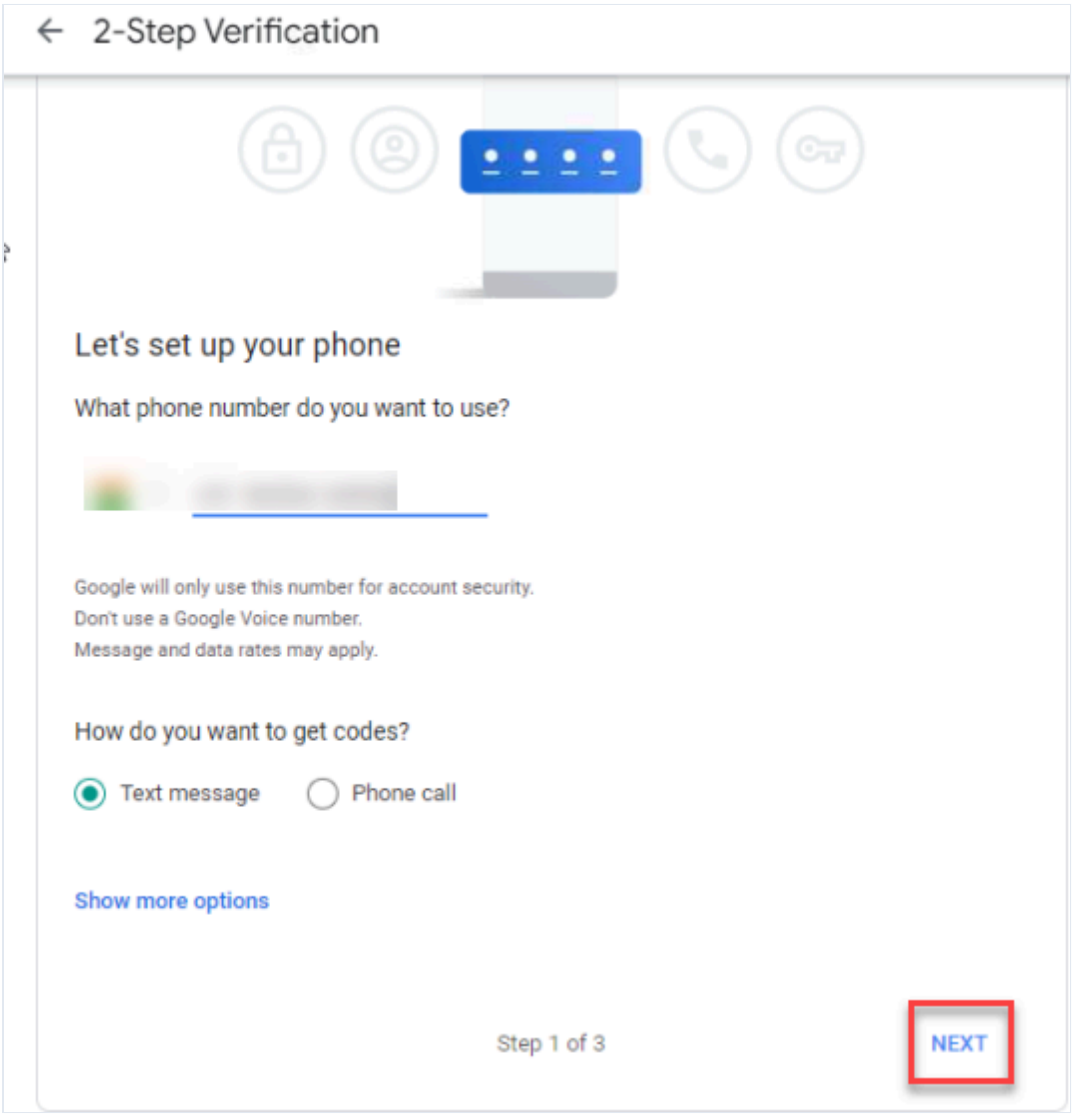


Click the **"Get Started"** button.

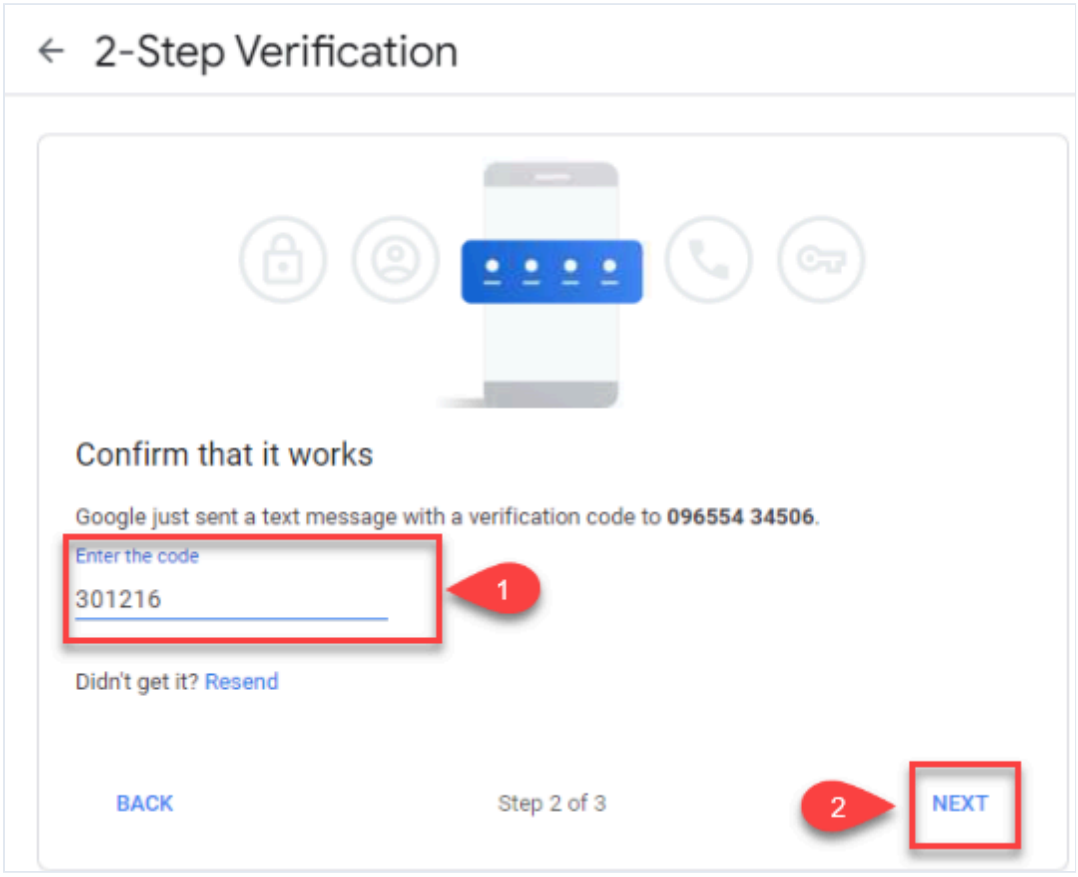


Step 5: Log in again using your **Gmail password** and **verification code**.

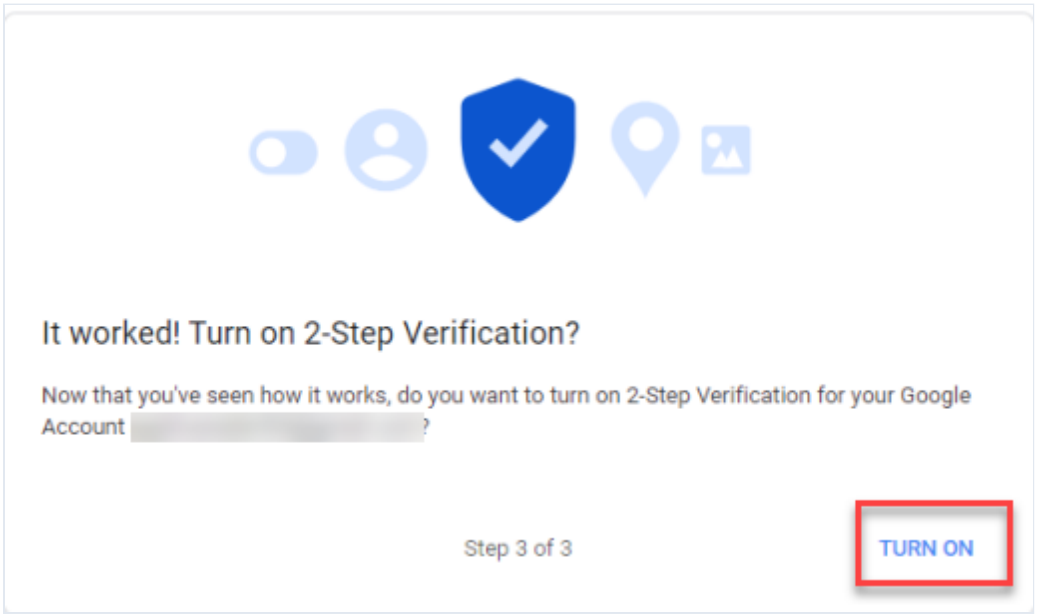
Step 6: Verify your **mobile number** and click on **Next** button.



Step 7: Enter the **verification code** and click on the **Next** button.

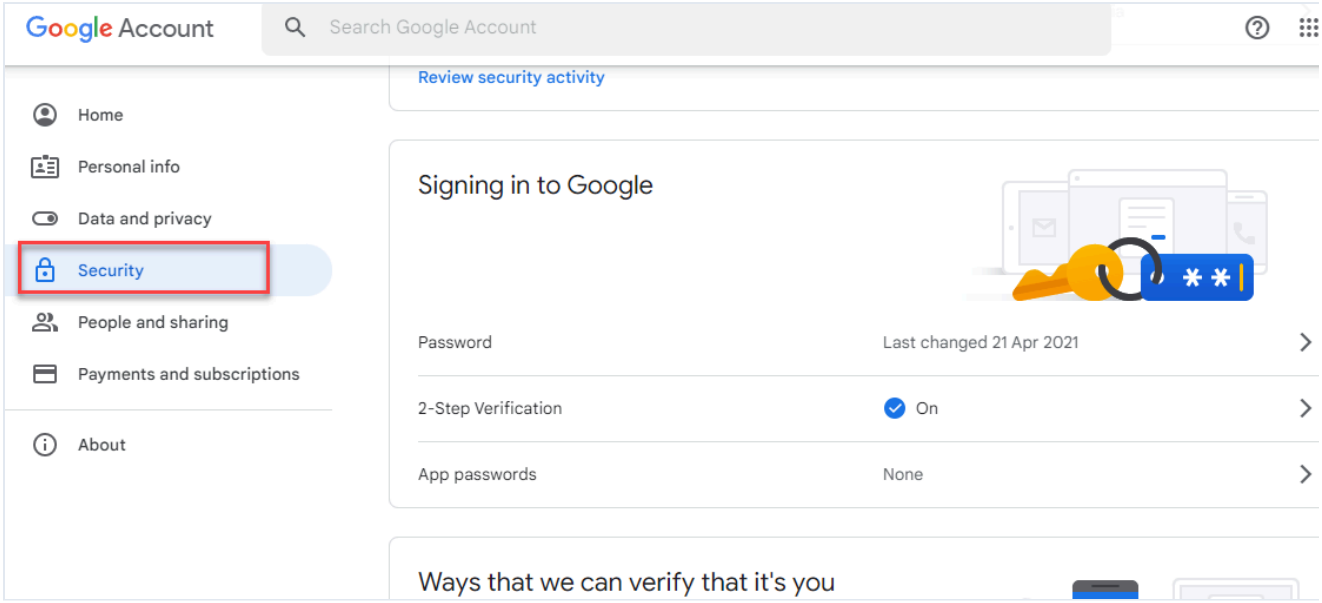


Step 8: Click the **Turn On** button.

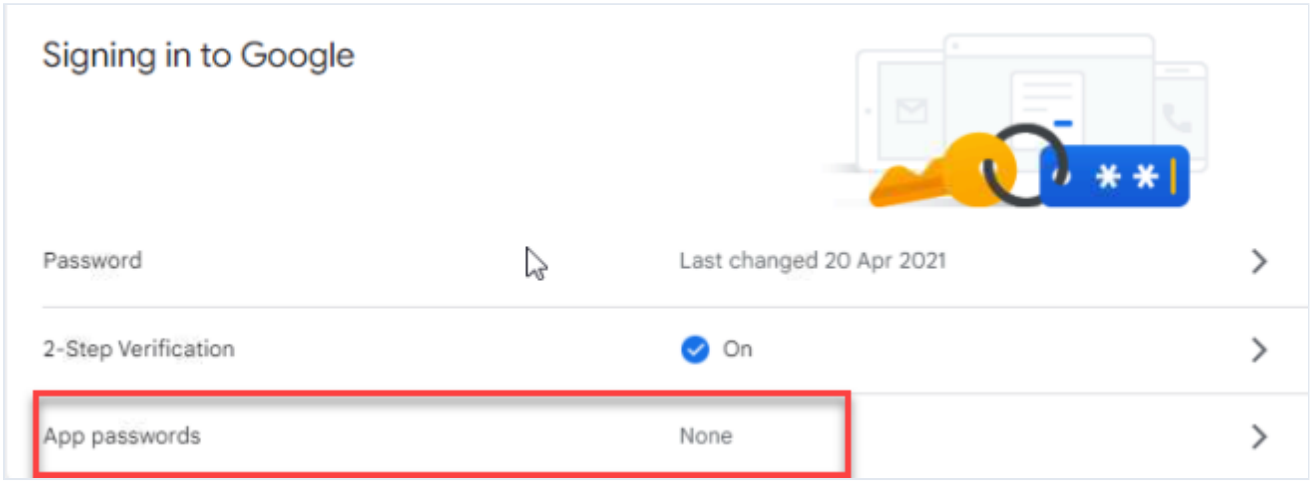


Click on 2-Step Verification to access the Gmail security screen.



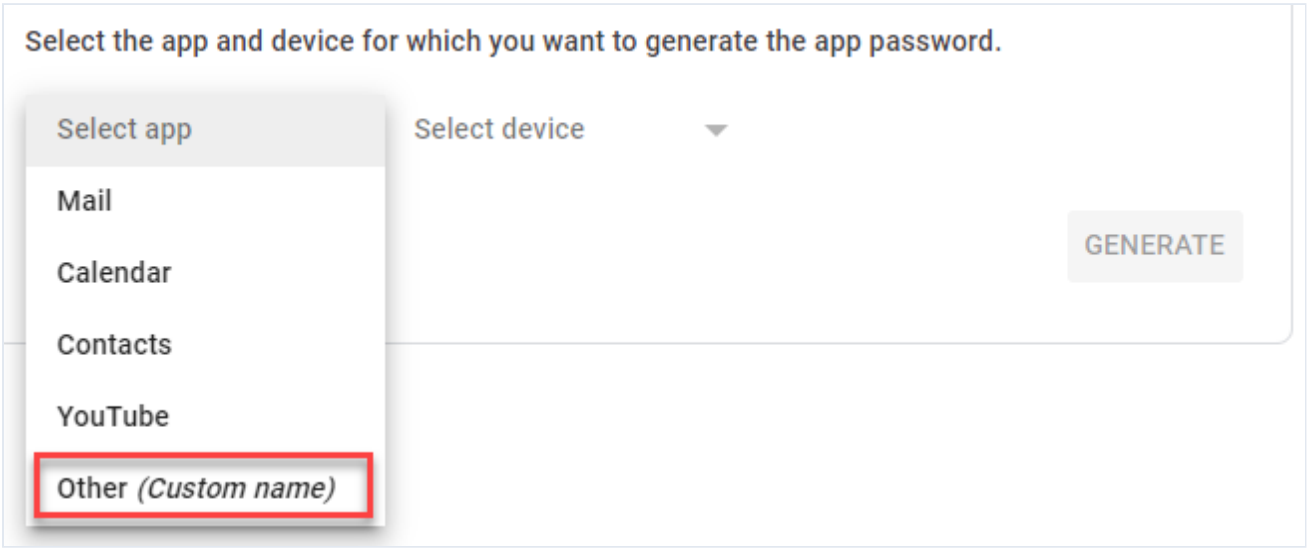


Step 9: Click on **App password**

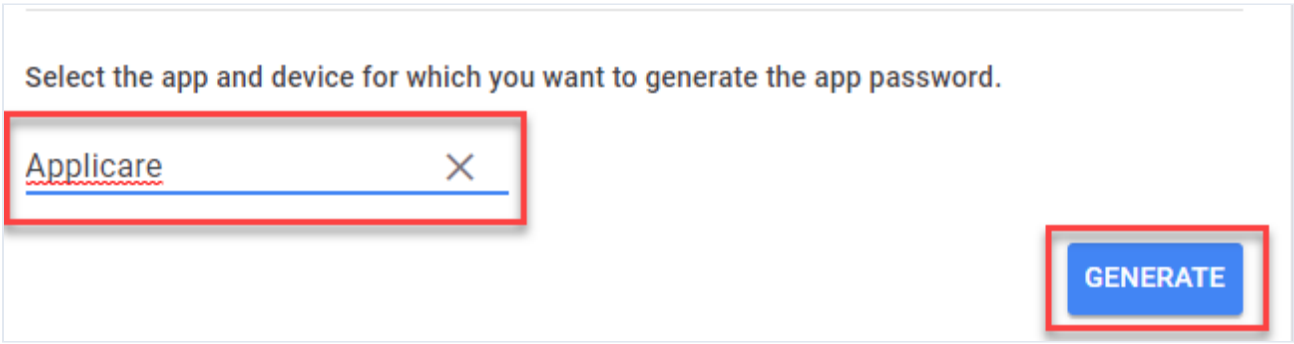


Step 10: Log in again your **Gmail password** and **verification code**.

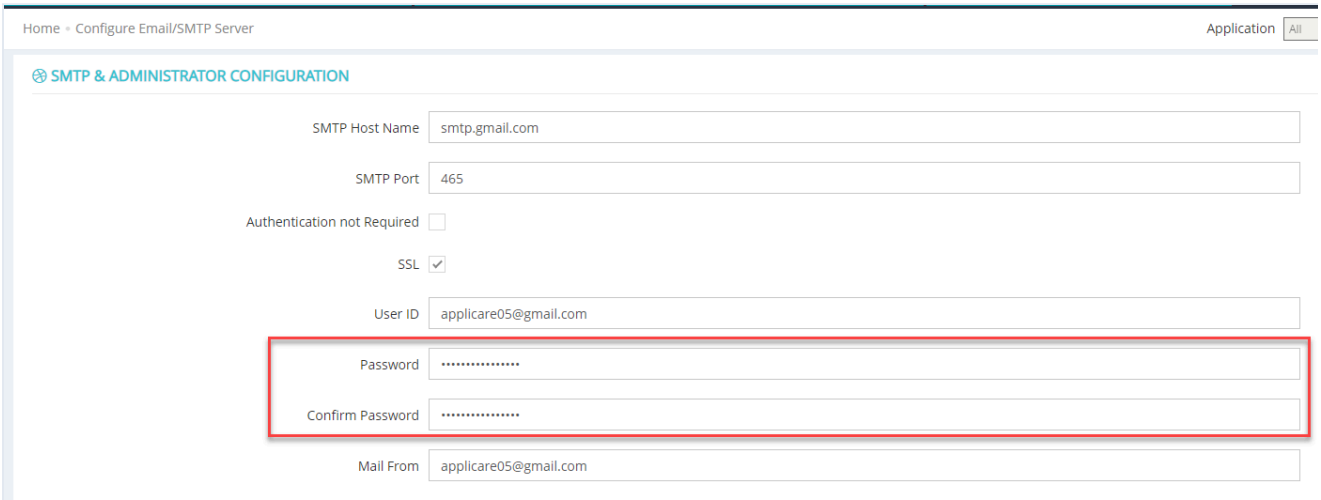
Step 11: From the **Select app** dropdown menu, choose the "other (custom name)" option.



Step 12: Enter "Applicare" name as the **App name** and click the **Generate** button.



Step 13: Copy the generated app password and save it in a file. This app password will be used as the email password in the SMTP configuration.



Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Disable Auto Add SNMP on controller

Applicare 9.2 · Updated January 2, 2025 · alerts · **Pending technical review**

Add the below jvm argument in the controller **catalina.bat/sh** file

```
TEXT
-Dapplicare.disableAutoAddSnmp=true
```

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

SQL injection detection in Applicare

Applicare 9.2 · Updated February 24, 2025 · **Pending technical review**

SQL injection is a type of cyberattack where malicious SQL code is inserted into a web application's input fields, which then interferes with the application's database queries. This can allow attackers to view, modify, or delete data that they are not authorized to access.

How it works:

1. **Vulnerable Application:** A web application is vulnerable to SQL injection if it directly incorporates user-supplied input into SQL queries without proper sanitization or parameterization.
2. **Malicious Input:** An attacker crafts malicious SQL code and inserts it into an input field, such as a search box or login form.
3. **Query Manipulation:** When the application processes the input, the malicious SQL code becomes part of the database query, altering its intended logic.
4. **Unauthorized Access:** The modified query can grant the attacker access to sensitive data, allow them to modify or delete data, or even take control of the database server.

Example:

Imagine a web application with a login form. The application constructs an SQL query like this:

SQL

```
TEXT
SELECT * FROM users WHERE username = '$username' AND password = '$password';
```

If an attacker enters the following username:

```
TEXT
' OR '1'='1
```

The resulting query becomes:

SQL

```
TEXT
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = '$password';
```

Since `'1'='1'` is always true, this query will return all users in the database, effectively bypassing authentication.

Impact:

SQL injection attacks can have severe consequences, including:

- **Data breaches:** Attackers can steal sensitive data, such as customer information, financial records, or intellectual property.
- **Data manipulation:** Attackers can modify or delete data, leading to data corruption or loss.
- **Denial of service:** Attackers can disrupt the availability of the application or the database server.
- **Complete system compromise:** In some cases, attackers can gain complete control of the database server and potentially other systems on the network.

Prevention:

The most effective way to prevent SQL injection is to use parameterized queries or prepared statements. These techniques treat user input as data rather than executable code, which prevents attackers from injecting malicious SQL code. Other prevention measures include input validation, least privilege, and web application firewalls.

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Vulnerability Scanner Setup

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Vulnerability Scanner

1. The scan will run on the configured server. 2. Once the scan is complete, the report will be transferred to the controller and saved in the controller's home directory. 3. Users can view the scan results and download the report from the Vulnerability Scanner screen.

Note

1. An internet connection is required on the agent machine for scanning to work. 2. The first scan will take longer as it needs to download and save data from the internet to the local machine. Subsequent scans will be quicker. 3. The duration of the scan depends on the number and size of the files. 4. CPU usage is also dependent on the number and size of the files.

Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

REST API Calls Using Token-Based Authentication

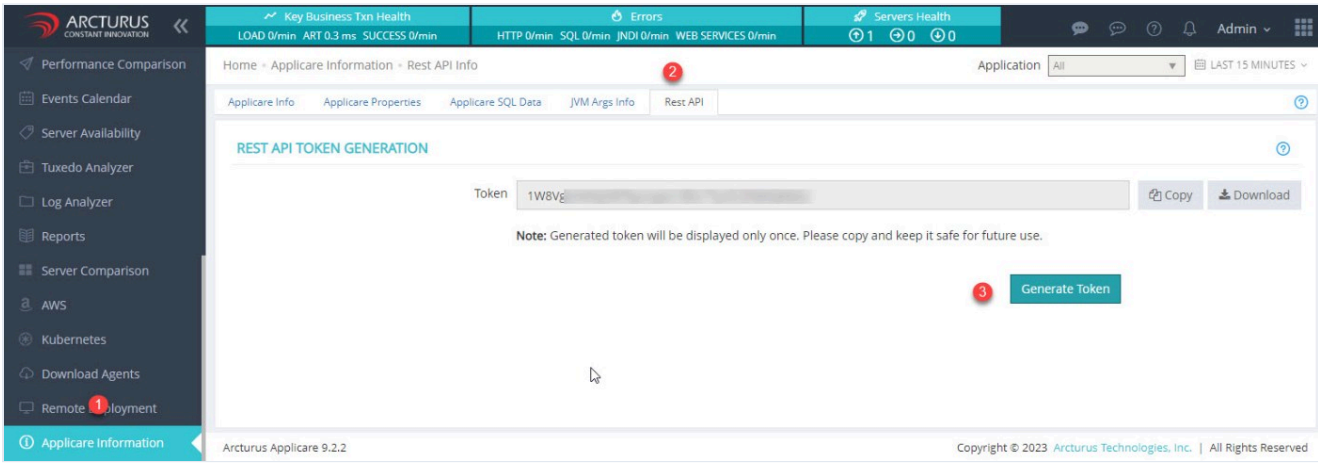
Applicare 9.2 · Updated September 4, 2024 · **Pending technical review**

To access data via the REST API, generate a API token in Applicare and include it in your request.

Token Generation

Navigate to Menu> Applicare Information > Rest API and click the **Generate Token** button. The newly generated token will appear in the Token field. This token is shown only once, So be sure to copy and securely store it for future use. The Controller saves the token in an encrypted format. **Once a new token is generated old token becomes invalid.**

REST APIs retrieve details from the controller



Authentication - API Key Authentication.

Token - One token per controller.

Token Validity - The token expires either when a new token is issued or if the Applicare license expires. After renewing the license, the existing token will remain valid.

Request content details

```yaml
Method Type: POST
Content-Type: application/json
REST_API_URL: Controller_Protocol://Controller_IP:Controller_Port/applicare/restapi
Header: X-API-Key: TOKEN_VALUE
REQUEST_BODY: REST API details to fetch
CA_CERTIFICATE_FILE: For SSL Certificate
```

Response content details

```
TEXT
Content-Type: application/json
```

```
TEXT
Success Response
{"status":"success","details":[{"name": "Google","state": "Failed"}]}

Failed Response
{"status":"error","details":[{"error": "Unauthorized. Token is missing or invalid"}]}

status – It displays the response is success or failure
details – Response message (JSON Array)
```

CURL Syntax

HTTP:

curl -H "X-API-Key: **TOKEN_VALUE**" -H "Content-Type: application/json" -X POST -d **REQUEST_BODY REST_API_URL**

HTTPS:

curl --cacert **CA_CERTIFICATE_FILE** -H "X-API-Key: **TOKEN_VALUE**" -H "Content-Type: application/json" -X POST -d **REQUEST_BODY REST_API_URL**

Parameters:

CA_CERTIFICATE_FILE: .crt (or) **.pem** file location

TOKEN_VALUE: Generated Token value from Applicare Console

REQUEST_BODY: REST API details to fetch

REST_API_URL: Controller_Protocol://Controller_IP:Controller_Port/applicare/restapi

```
TEXT
Controller_Protocol: Applicare Controller running protocol (http or https)
Controller_IP: Applicare Controller running IP or domain name
Controller_Port: Applicare Controller running Port
```

For example, let us consider the below values

CA_CERTIFICATE_FILE: D:\applicare.pem

TOKEN_VALUE: 123456789

REST_API_URL:

HTTP: `http://localhost:8880/applicare/restapi`

HTTPS: `https://localhost:8443/applicare/restapi`

REQUEST_BODY details for Various API

1. To get the status of all active synthetic transactions

```
TEXT
{"\q\":\"syntheticAllTransactionStatus\"}
```

CURL e.g.

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionStatus\"} http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionStatus\"} https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "Google","state": "Failed"},
{"name": "google","state": "Queued"},{"name": "Json","state":"Success"}]}
```

name - Synthetic Transaction Name

state - State of the transaction. (Success, Failed or Queued)

2. To retrieve details of all active synthetic transactions

```
TEXT
{\"q\":\"syntheticAllTransactionDetails\"}
```

CURL e.g.

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionDetails\"} http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionDetails\"} https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"group": "Default","app": "Applicare",
"name": "Google","node": "ApplicareAdmin","state": "Failed","lastChecked": 1657724700000,
"executionTime": 21040,"message": ""},{"group": "Default","app":"","name": "Json",
"node": "ApplicareAdmin","state": "Success","lastChecked":1657724700000,
"executionTime": 316,"message": ""}]}
```

group - Name of the group

app - Name of the application

name - Name of the synthetic transaction

node- Node where the transaction executed

state - State of the transaction. (Success, Failed or Queued)

lastChecked - Last execution time

executionTime - Time taken for the transaction. Value in milliseconds

3. To get the transaction status based on transaction names

```
TEXT
{\"q\":\"syntheticTransactionStatus\",\"txnname\":\"TXN_NAME1,TXN_NAME2\"}
```

TXN_NAME1, TXN_NAME2 - Name of synthetic transactions

cURL e.g.

a. To get the transaction status for single transaction

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "00001","state": "Failed"}]}
```

b. To get the transaction status for multiple transactions

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001,00002**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001,00002**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "00001","state": "Failed"},
{"name": "00002","state":"Success"}]}
```

4. To get the transaction status based on applications

```
TEXT
{\"q\":\"syntheticTransactionStatus\",\"appname\":\"APP_NAME1,APP_NAME2\"}
```

APP_NAME1, APP_NAME2 - Name of the Application which synthetic transactions was assigned.

CURL e.g.

a. To retrieve transaction the status for single application

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"name":"00004","state":"Queued"},
{"name":"10jmeter","state":"Success"}]}
```

b. To get the transaction status for multiple applications

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp,Applicare**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp,Applicare**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"name":"00004","state":"Queued"},
{"name":"10jmeter","state":"Success"},{"name":"Error","state":"Failed"},
{"name":"Google","state":"Failed"}]}
```

5. To get the transaction details based on transaction names

```
TEXT
{\"q\":\"syntheticTransactionDetails\",\"txnname\":\"TXN_NAME1,TXN_NAME2\"}
```

TXN_NAME1, TXN_NAME2 - Name of synthetic transactions

CURL e.g.

a. To get the transaction details for single transaction

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**txnname**\":\"**00001**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**txnname**\":\"**00001**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"group":"Default","app":"","name":"00001","node":"Agent1",
"state":"Queued","lastChecked":0,"executionTime":0,"message":""}]}
```

b. To get the transaction details for multiple transactions

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"syntheticTransactionDetails\",\"**txnname**\":\"**00001,00002**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"syntheticTransactionDetails\",\"**txnname**\":\"**00001,00002**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```yaml
{"status":"success","details":[{"group":"Default","app":"","name":"00001","node":"Agent1",
"state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"","name":"00002","node":"ApplicareAdmin","state":"Success",
"lastChecked":1657878180000,"executionTime":116720,"message":""}]}
```

6. To get the transaction details based on applications

```text
{\"q\":\"syntheticTransactionDetails\",\"appname\":\"APP_NAME1,APP_NAME2\"}
```

APP_NAME1, APP_NAME2 - Name of the Application which synthetic transactions was assigned.

CURL e.g.

a. To get the transaction for single application

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```yaml
{"status":"success","details":[{"group":"Default","app":"TestApp","name":"00004",
"node":"Agent2","state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"TestApp","name":"10jmeter","node":"ApplicareAdmin",
"state":"Success","lastChecked":1657879080000,"executionTime":1469,"message":""}]}
```

b. To get the transaction details for multiple applications

HTTP:

curl -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp,Applicare**\"} **http**://**localhost**:**8880**/applicare/restapi

HTTPS:

curl --cacert **D:\applicare.pem** -H "X-API-Key: **123456789**" -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp,Applicare**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

YAML

{"status":"success","details":[{"group":"Default","app":"TestApp","name":"00004",
"node":"Agent2","state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"TestApp","name":"10jmeter","node":"ApplicareAdmin",
"state":"Success","lastChecked":1657879080000,"executionTime":1469,"message":""},
{"group":"Default","app":"Applicare","name":"Error","node":"ApplicareAdmin",
"state":"Failed","lastChecked":1657879020000,"executionTime":24432,"message":""},
{"group":"Default","app":"Applicare","name":"Google","node":"ApplicareAdmin",
"state":"Failed","lastChecked":1657879080000,"executionTime":21031,"message":""}]}

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

REST API Calls Using X.509 Certificate-Based Authentication

Applicare 9.2 · Updated August 30, 2024 · **Pending technical review**

X.509 Certificate-Based Authentication

X.509 Cert-based authentication provides an easy and secure way to access Applicare REST APIs. Below are the details on enabling it.

1. Bring down the Applicare controller.

2. Navigate to **<Applicare controller home directory>\server\conf** and open the **server.xml** file.

3. Add a new connector as detailed below, specifying the **Keystore** and **Truststore** file paths password and set the **clientAuth** to true.

<Connector protocol="HTTP/1.1" port="8444" maxThreads="200" scheme="https" secure="true" SSLEnabled="true" keystoreFile="**<keystorefile.jks>**" keystorePass="**password**" truststoreFile="**<truststorefile.jks>**" truststorePass="**password**" **clientAuth="true"** sslProtocol="TLS" />

Example:

```
<Connector
protocol="HTTP/1.1"
port="8444" maxThreads="200"
scheme="https" secure="true" SSLEnabled="true"
keystoreFile="E:\Applicare\MySQL\keystore.jks" keystorePass="123456"
truststoreFile="E:\Applicare\MySQLtruststore.jks" truststorePass="123456"
clientAuth="true" sslProtocol="TLS" />
```

4. Restart the Applicare Controller.

Request content details

```yaml
Method Type: POST
Content-Type: application/json
REST_API_URL: Controller_Protocol://Controller_IP:Controller_Port/applicare/restapi
REQUEST_BODY: REST API details to fetch
CA_CERTIFICATE_FILE: CA Certificate file for SSL.
CLIENT_CERTIFICATE_FILE: Client Certificate file for client side authentication. (.p12) file
PassphrasePassword: Password for the .p12 file at the time of file generation.
```

Request Content Information

```
TEXT
Content-Type: application/json
```

```
TEXT
Success Response
{"status":"success","details":[{"name": "Google","state": "Failed"}]}

Failed Response
{"status":"error","details":[{"error": "Unauthorized. Token is missing or invalid"}]}

status – It displays the response is success or failure
details – Response message (JSON Array)
```

cURL Command Syntax

HTTPS:

curl --cacert **CA_CERTIFICATE_FILE** --cert **CLIENT_CERTIFICATE_FILE:PassphrasePassword** -H "Content-Type: application/json" -X POST -d **REQUEST_BODY REST_API_URL**

Fields:

CA_CERTIFICATE_FILE : Location of the **.crt** (or) **.pem** certificate file

CLIENT_CERTIFICATE_FILE: Client certificate file with **.p12** extension

PassphrasePassword: Passphrase for the client certificate file

REQUEST_BODY: Details of the REST API request

REST_API_URL: Controller_Protocol://Controller_IP:Controller_Port/applicare/restapi

```
TEXT
Controller_Protocol: Applicare Controller running protocol (http or https)
Controller_IP: Applicare Controller running IP or domain name
Controller_Port: Applicare Controller running Port
```

For example, Consider the following values:

CA_CERTIFICATE_FILE: D:\applicare.pem

.P12_FILE: D:\client.p12

PassphrasePassword: clientpass

REST_API_URL: `https://localhost:8443/applicare/restapi`

REQUEST_BODY details for Various API

1. To retrieve the status of all active synthetic transactions

```
TEXT
{"\q\":\"syntheticAllTransactionStatus\"}
```

cURL Example

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionStatus\"}** **https**://**localhost**:**8444**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "Google","state": "Failed"},
{"name": "google","state": "Queued"},{"name": "Json","state":"Success"}]}
```

name - Synthetic Transaction Name

state - State of the transaction. (Success, Failed or Queued)

2. To retrieve details of all active synthetic transactions

```
TEXT
{\"q\":\"syntheticAllTransactionDetails\"}
```

cURL Example

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d **{\"q\":\"syntheticAllTransactionDetails\"}** **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"group": "Default","app": "Applicare",
"name": "Google","node": "ApplicareAdmin","state": "Failed","lastChecked": 1657724700000,
"executionTime": 21040,"message": ""},{"group": "Default","app":"","name": "Json",
"node": "ApplicareAdmin","state": "Success","lastChecked":1657724700000,
"executionTime": 316,"message": ""}]}
```

group - Group Name

app - Application Name

name - Synthetic Transaction Name

node- In Which Node it executes

state - State of the transaction. (Success, Failed or Queued)

lastChecked - Last execution time

executionTime - Time taken for the transaction. Value in milliseconds

3. To retrieve transaction status based on transaction names

```
TEXT
{\"q\":\"syntheticTransactionStatus\",\"txnname\":\"TXN_NAME1,TXN_NAME2\"}
```

TXN_NAME1, TXN_NAME2 - Name of synthetic transactions

cURL Example

a. To get the transaction status for single transaction

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "00001","state": "Failed"}]}
```

b. To get the transaction status for multiple transactions

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**txnname**\":\"**00001,00002**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details": [{"name": "00001","state": "Failed"},
{"name": "00002","state":"Success"}]}
```

4. To retrieve the transaction status based on applications

```
TEXT
{\"q\":\"syntheticTransactionStatus\",\"appname\":\"APP_NAME1,APP_NAME2\"}
```

APP_NAME1, APP_NAME2 - Name of the Application which synthetic transactions was assigned.

cURL Example

a. To get the transaction status for single application

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"name":"00004","state":"Queued"},
{"name":"10jmeter","state":"Success"}]}
```

b. To get the transaction status for multiple applications

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionStatus**\",\"**appname**\":\"**TestApp,Applicare**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"name":"00004","state":"Queued"},
{"name":"10jmeter","state":"Success"},{"name":"Error","state":"Failed"},
{"name":"Google","state":"Failed"}]}
```

5. To get the transaction details based on transaction names

```
TEXT
{\"q\":\"syntheticTransactionDetails\",\"txnname\":\"TXN_NAME1,TXN_NAME2\"}
```

TXN_NAME1, TXN_NAME2 - Name of synthetic transactions

cURL Example

a. To get the transaction details for single transaction

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**txnname**\":\"**00001**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
TEXT
{"status":"success","details":[{"group":"Default","app":"","name":"00001","node":"Agent1",
"state":"Queued","lastChecked":0,"executionTime":0,"message":""}]}
```

b. To get the transaction details for multiple transactions

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"syntheticTransactionDetails\",\"**txnname**\":\"**00001,00002**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
YAML
{"status":"success","details":[{"group":"Default","app":"","name":"00001","node":"Agent1",
"state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"","name":"00002","node":"ApplicareAdmin","state":"Success",
"lastChecked":1657878180000,"executionTime":116720,"message":""}]}
```

6. To get the transaction details based on applications

```
TEXT
{\"q\":\"syntheticTransactionDetails\",\"appname\":\"APP_NAME1,APP_NAME2\"}
```

APP_NAME1, APP_NAME2 - Name of the application to which the synthetic transactions are assigned.

cURL Example

a. To get the transaction for single application

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
YAML
{"status":"success","details":[{"group":"Default","app":"TestApp","name":"00004",
"node":"Agent2","state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"TestApp","name":"10jmeter","node":"ApplicareAdmin",
"state":"Success","lastChecked":1657879080000,"executionTime":1469,"message":""}]}
```

b. To retrieve transaction details for multiple applications

curl --cacert **D:\applicare.pem** --cert **D:\client.p12:changeit** -H "Content-Type: application/json" -X POST -d {\"q\":\"**syntheticTransactionDetails**\",\"**appname**\":\"**TestApp,Applicare**\"} **https**://**localhost**:**8443**/applicare/restapi

Response

```
YAML
{"status":"success","details":[{"group":"Default","app":"TestApp","name":"00004",
"node":"Agent2","state":"Queued","lastChecked":0,"executionTime":0,"message":""},
{"group":"Default","app":"TestApp","name":"10jmeter","node":"ApplicareAdmin",
"state":"Success","lastChecked":1657879080000,"executionTime":1469,"message":""},
{"group":"Default","app":"Applicare","name":"Error","node":"ApplicareAdmin",
"state":"Failed","lastChecked":1657879020000,"executionTime":24432,"message":""},
{"group":"Default","app":"Applicare","name":"Google","node":"ApplicareAdmin",
"state":"Failed","lastChecked":1657879080000,"executionTime":21031,"message":""}]}
```

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Applicare Data Security

Applicare 9.2 · Updated September 2, 2024 · **Pending technical review**

Applicare can be deployed either as an On-Prem(ise) hosting or accessed through our SaaS environment.

Applicare On-Prem

In the Applicare On-Prem environment, security is managed by the customer. Applicare supports multiple TLS versions and cipher suites, including TLS 1.2 (SSL Labs Grade A+). All traffic to the Applicare controller is secured with HTTPS/TLS. Customers are responsible for using their own SSL certificate.

Applicare On-Prem Data Storage

Applicare supports industry-standard databases such as Oracle and MySQL for data storage, adhering to the customer's security guidelines. We also offer encryption for JDBC communication with the selected database.

Applicare On-Prem Isolation

In an Applicare On-Prem deployment, connection to Arcturus occur only if explicitly initiated by customer such as for checking updates. The Applicare On-Prem Controller doesn't require any internet access to function.

Applicare SaaS

The Applicare SaaS environment is build on industry leading AWS (or Microsoft Azure or Google Cloud Platform as needed). It is secured foundation-ally by design with the following

- Each customer is operates within dedicated runtime environment
- Each customer's data is stored in a separate database schema
- Each customer assigned a unique domain

All Applicare traffic between SingleAgents deployed at customer's site or cloud and Applicare SaaS is encrypted using latest TLS currently TLS 1.2 (SSL Labs Grade A+) and certs are managed by AWS ACM.

Applicare database is NOT accessible from outside and is only accessible from within Arcturus VPN and only by Applicare SaaS Controllers.

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Manage Applicare Data

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

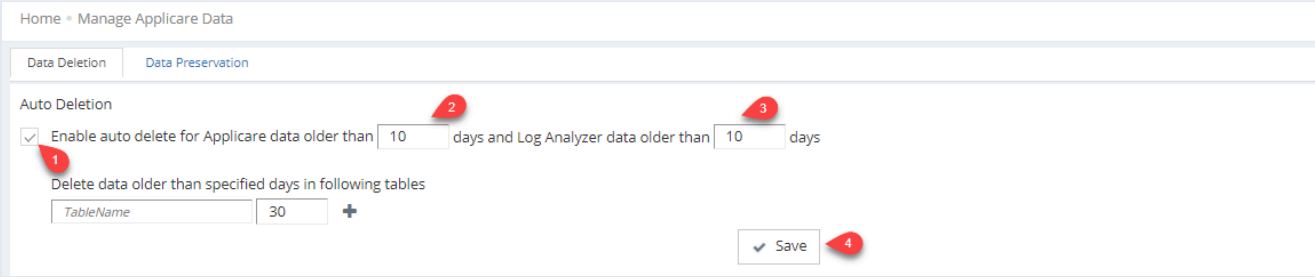
Overview

As the Applicare Controller operates, table storage in the database will grow, potentially reducing disk space on the machine running the controller. Configuring this screen can help mitigate such issues.

Configuration

Auto Deletion

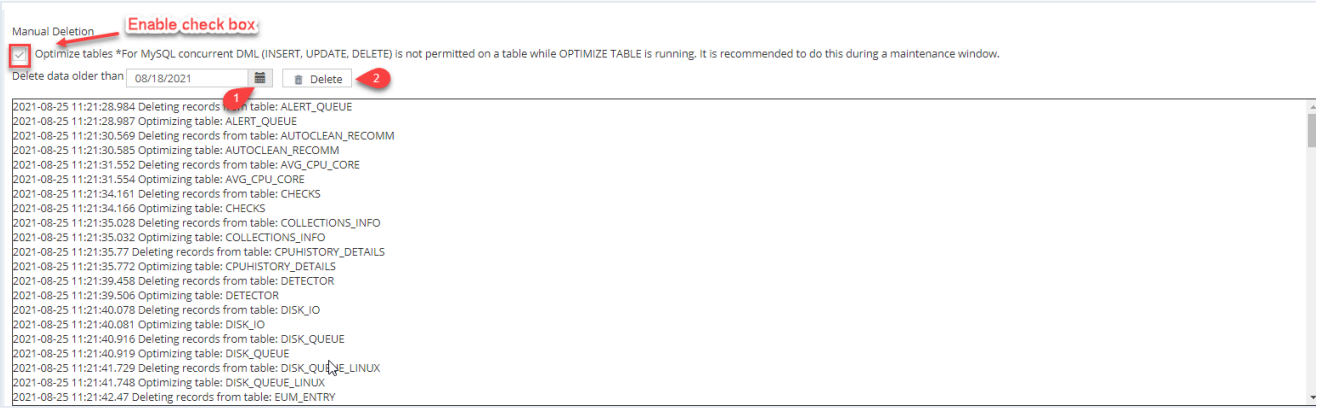
When set up on this screen, Applicare will automatically clear the database storage as per the configuration.



Enable auto deletion by checking the appropriate checkbox. Then, specify the no of days after which data should be deleted and save the setting.

Manual Deletion

Use this option to immediately delete database storage if disk space is running low on the machine running the Applicare Controller.



Enable the Manual deletion check box, select the date for which you want to remove the data, and click the Delete button. The database storage will be cleared according to the specified date within a few seconds.

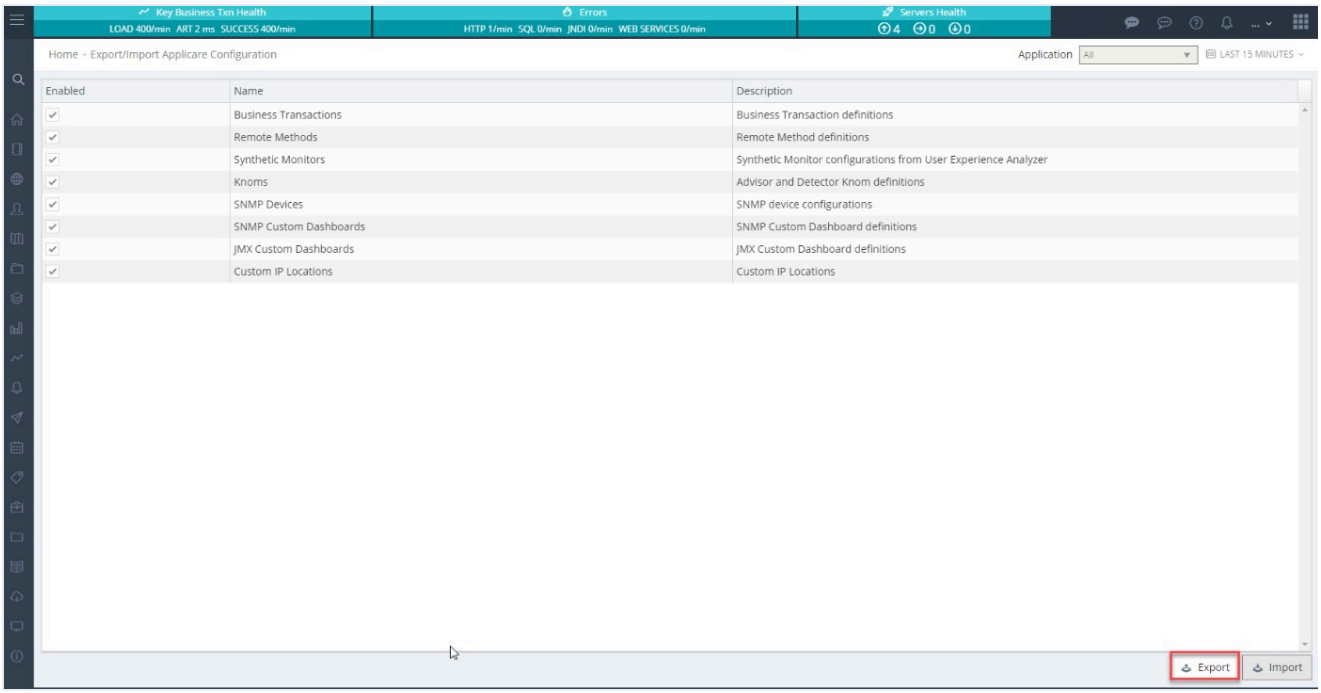
Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

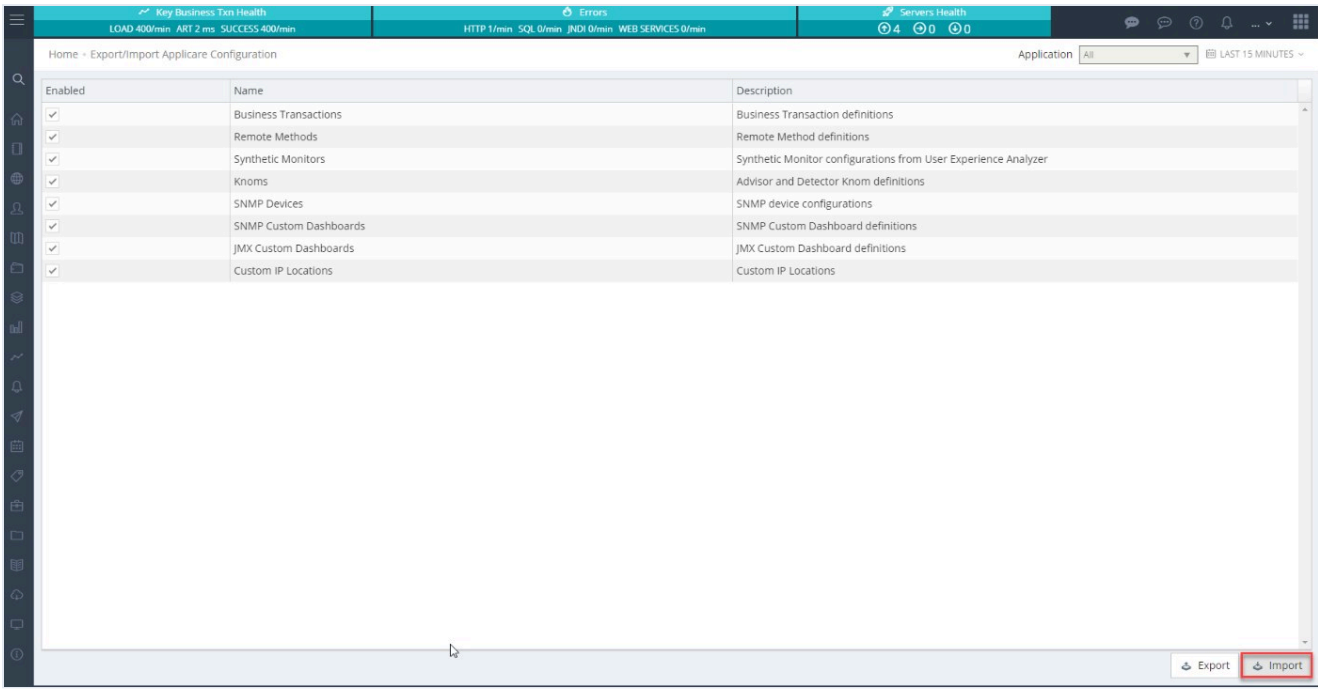
Export/Import Applicare Configuration

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

Export Applicare Configuration



Import Applicare Configuration



CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

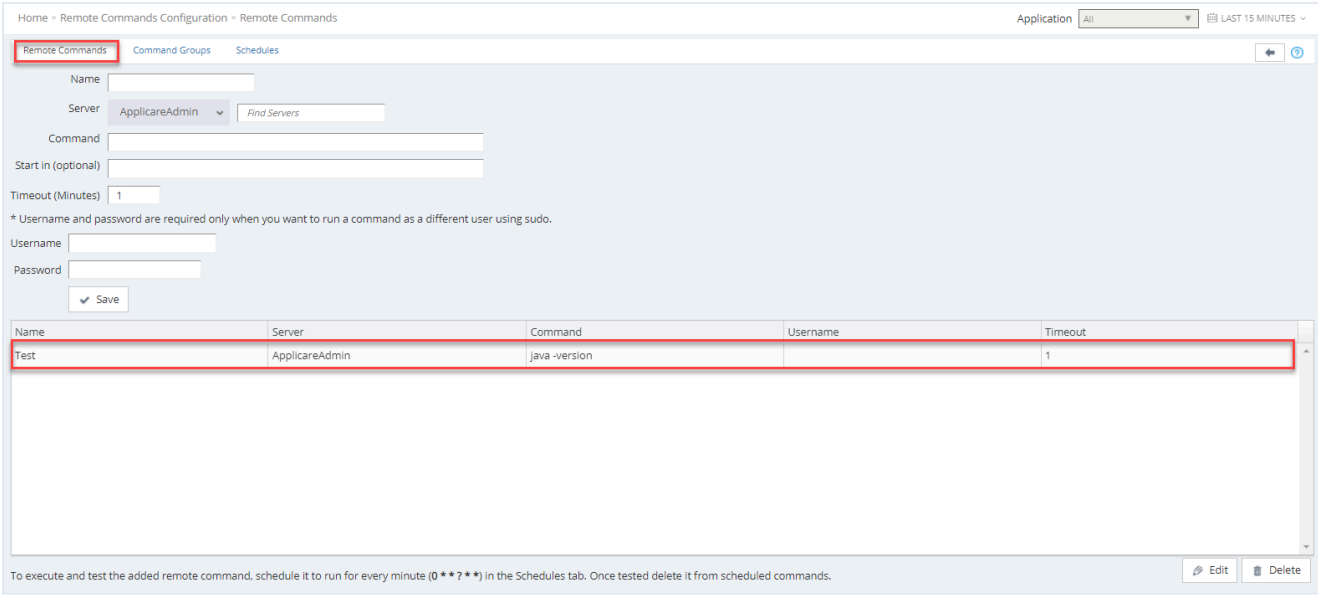
Remote Commands

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Remote Commands allow you to execute scripts or processes on servers where Applicare agents are installed. They can be configured to run as specific users (On Unix-based systems with sudo access) and scheduled to execute periodically using cron-like schedules.

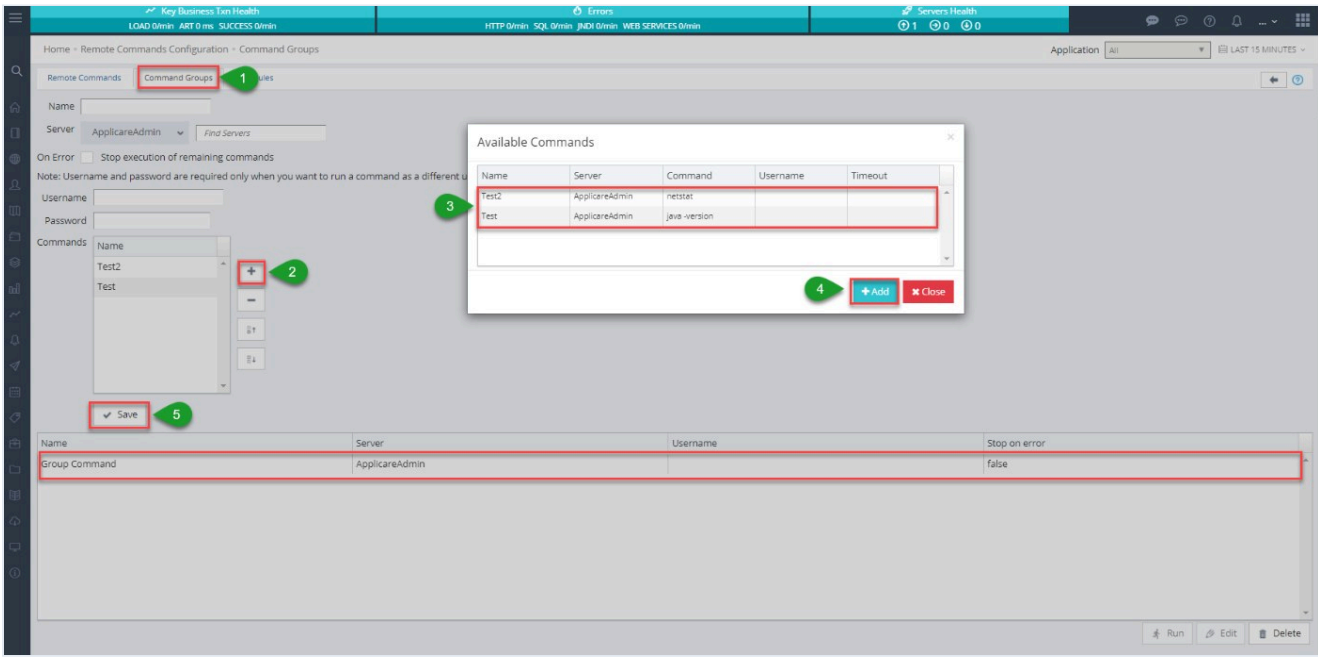
Configuration

Name	**A unique name to identify the command within the Applicare console**
Server	The Applicare agent that will execute this command
Command	The script or command to be executed
Start in (optional)	Specify the directory path from which the command should be executed
Timeout (Minutes)	Applicare will automatically terminate the process if it does not complete within the specified number of minutes.
Username (optional)	You can execute commands as a specific user on the target server. This setting can be overridden when running commands manually and is only supported on operating systems with sudo installed.
Password (optional)	Enter the password for the specified username. This can be overridden when executing commands manually.



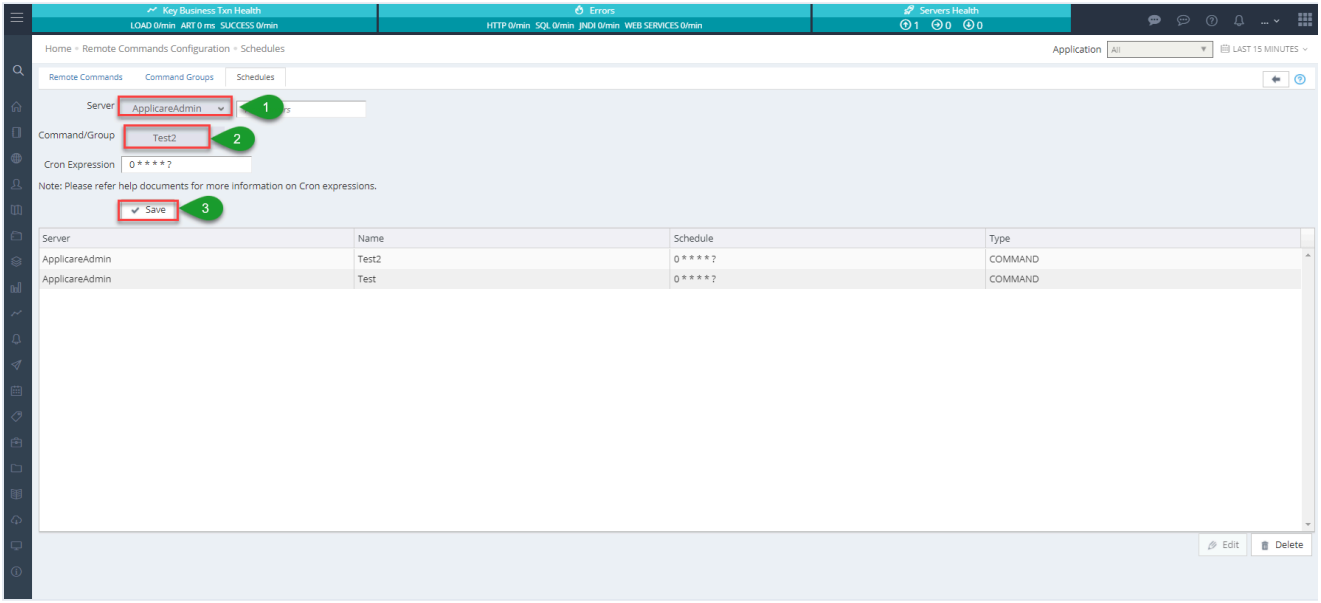
Command Groups

You can organize multiple commands on a server into a single group. Commands within the group will be executed in the specified order. Similarly to Individual commands, you can execute the command group as a specific user on operating systems where sudo is available.



Command Schedules

You can schedule commands and command groups to run periodically using cron-like scheduling expressions.



Format

A cron expression is a string consisting of 6 or 7 fields separated by whitespace. Each field can include any of the permitted values and various combinations of the special characters to that field. The fields are described as follows:

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

Cron expressions can be as simple as: * * * * ? * or more complex, like this: 0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2010

Special characters

- *

(*"all values"*)- used to select all values within a field. For example, "*" in the minute field means *"every minute"*

- ?

(*"no specific value"*) - useful when you need to specify something in one of the two fields in which the character is allowed, but not the other. For example, if I want my trigger to fire on a particular day of the month (say, the 10th), but don't care what day of the week that happens to be, I would put "10" in the day-of-month field, and "?" in the day-of-week field. See the examples below for clarification.

- -

  - used to specify ranges. For example, "10-12" in the hour field means *"the hours 10, 11 and 12"*

- ,

  - used to specify additional values. For example, "MON,WED,FRI" in the day-of-week field means *"the days Monday, Wednesday, and Friday"*

- /

  - used to specify increments. For example, "0/15" in the seconds field means *"the seconds 0, 15, 30, and 45"*. And "5/15" in the seconds field means *"the seconds 5, 20, 35, and 50"*. You can also specify '/' after the '**' character - in this case **'' is equivalent to having '0' before the '/'. '1/3' in the day-of-month field means *"fire every 3 days starting on the first day of the month"*

- **L**

  (*"last"*) - has different meaning in each of the two fields in which it is allowed. For example, the value "L" in the day-of-month field means *"the last day of the month"* - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But

if used in the day-of-week field after another value, it means *"the last xxx day of the month"* - for example "6L" means *"the last friday of the month"*. When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing results.

- **W**

  (*"weekday"*) - used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: *"the nearest weekday to the 15th of the month"*. So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The 'W' character can only be specified when the day-of-month is a single day, not a range or list of days.

  The 'L' and 'W' characters can also be combined in the day-of-month field to yield 'LW', which translates to *"last weekday of the month"*.

- #

  - used to specify "the nth" XXX day of the month. For example, the value of "6#3" in the day-of-week field means *"the third Friday of the month"* (day 6 = Friday and "#3" = the 3rd one in the month). Other examples: "2#1" = the first Monday of the month and "4#5" = the fifth Wednesday of the month. Note that if you specify "#5" and there is not 5 of the given day-of-week in the month, then no firing will occur that month.

  The legal characters and the names of months and days of the week are not case sensitive. MON is the same as mon.

Examples

Here are some full examples:

Expression	Meaning
0 * * * * ?	Fire at every minute
0 */2 * ? * *	Fire at every 2 minutes
0 */3 * ? * *	Fire at every 3 minutes
0 0 * ? * *	Fire at every hour
0 0 */2 ? * *	Fire at every 2 hours
0 0 */3 ? * *	Fire at every 3 hours
0 0 12 * * ?	Fire at 12pm (noon) every day
0 15 10 ? * *	Fire at 10:15am every day
0 15 10 * * ?	Fire at 10:15am every day
0 15 10 * * ? *	Fire at 10:15am every day
0 15 10 * * ? 2005	Fire at 10:15am every day during the year 2005

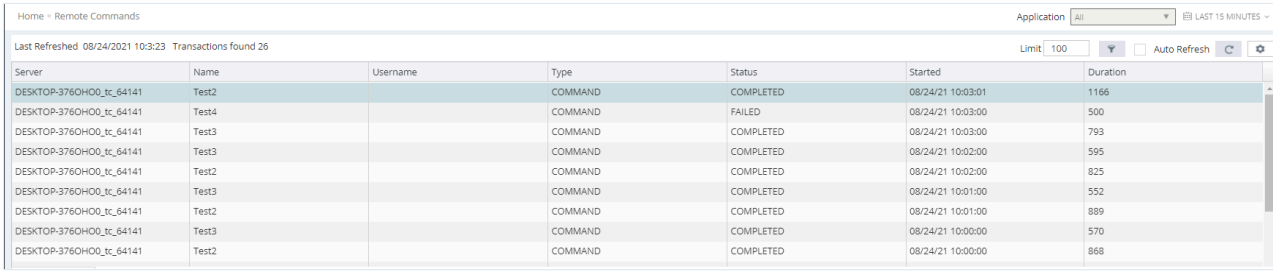
Expression	Meaning
0 * 14 * * ?	Fire every minute starting at 2pm and ending at 2:59pm, every day
0 0/5 14 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day
0 0/5 14,18 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day
0 0-5 14 * * ?	Fire every minute starting at 2pm and ending at 2:05pm, every day
0 10,44 14 ? 3 WED	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
0 15 10 ? * MON-FRI	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Fire at 10:15am on the 15th day of every month
0 15 10 L * ?	Fire at 10:15am on the last day of every month
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Fire at 10:15am on every last friday of every month during the years 2002, 2003, 2004 and 2005
0 15 10 ? * 6#3	Fire at 10:15am on the third Friday of every month
0 0 12 1/5 * ?	Fire at 12pm (noon) every 5 days every month, starting on the first day of the month.
0 11 11 11 11 ?	Fire every November 11th at 11:11am.

Pay attention to the effects of '?' and '*' in the day-of-week and day-of-month fields!

- Support for specifying both a day-of-week and a day-of-month value is not complete (you must currently use the '?' character in one of these fields).
- Be careful when setting fire times between mid-night and 1:00 AM - "daylight savings" can cause a skip or a repeat depending on whether the time moves back or jumps forward.

Execution Results

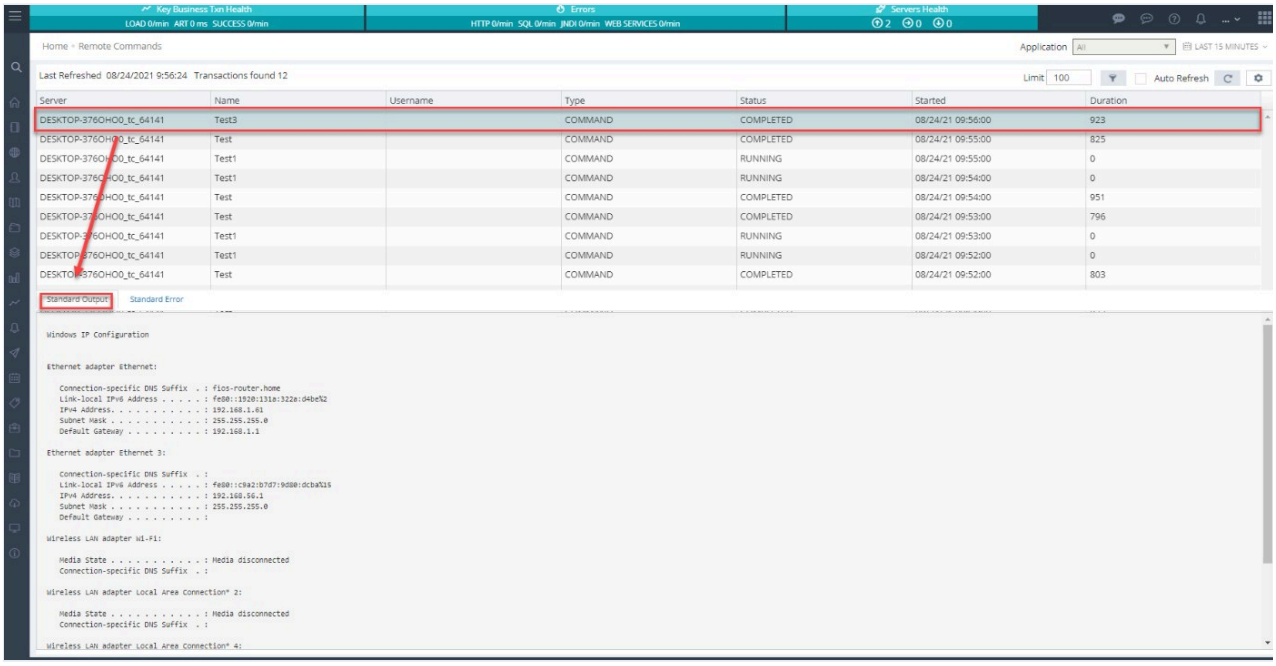
You can view the execution details of commands and command groups run by Applicare agents, whether they are manually triggered by a user or executed automatically according to a user-defined schedule.



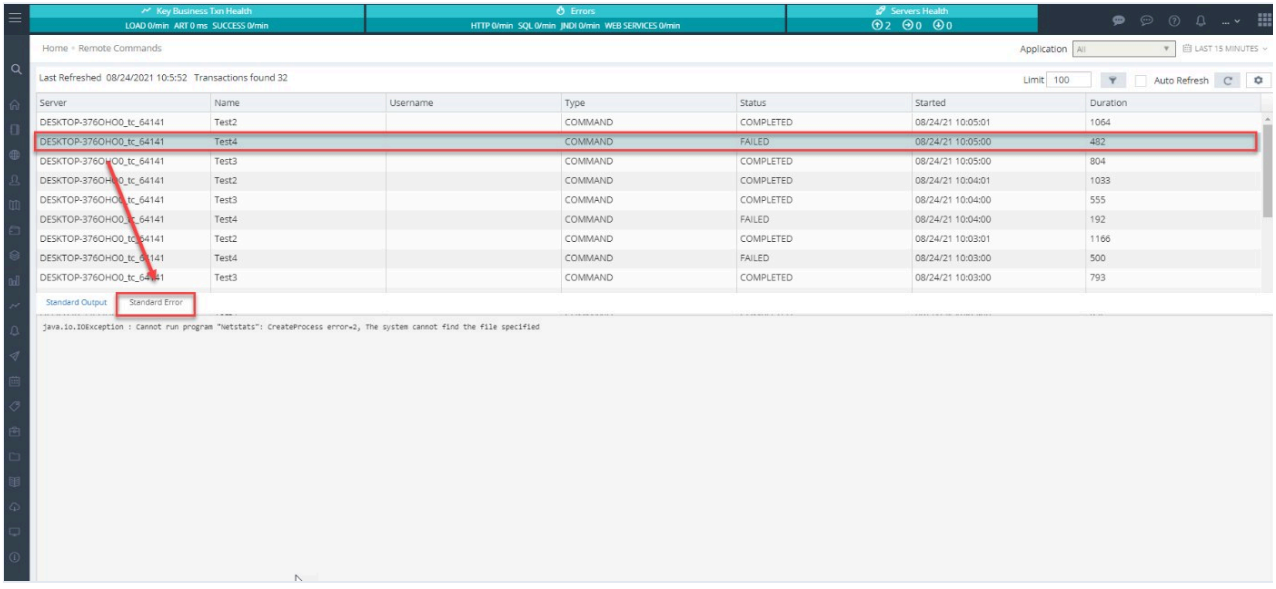
Double-clicking an entry will open a detailed view that displays information about the command or commands (for command groups). By selecting a command in this view, you can see the details written to standard output and standard error during the execution.

Commands

Standard Output



Standard Error



Command Groups

```
TEXT
      ![Screenshot 7 — Remote Commands](asset:images/configuration--admin--remote-commands/fig-07.png)
```

Monitoring with Applicare

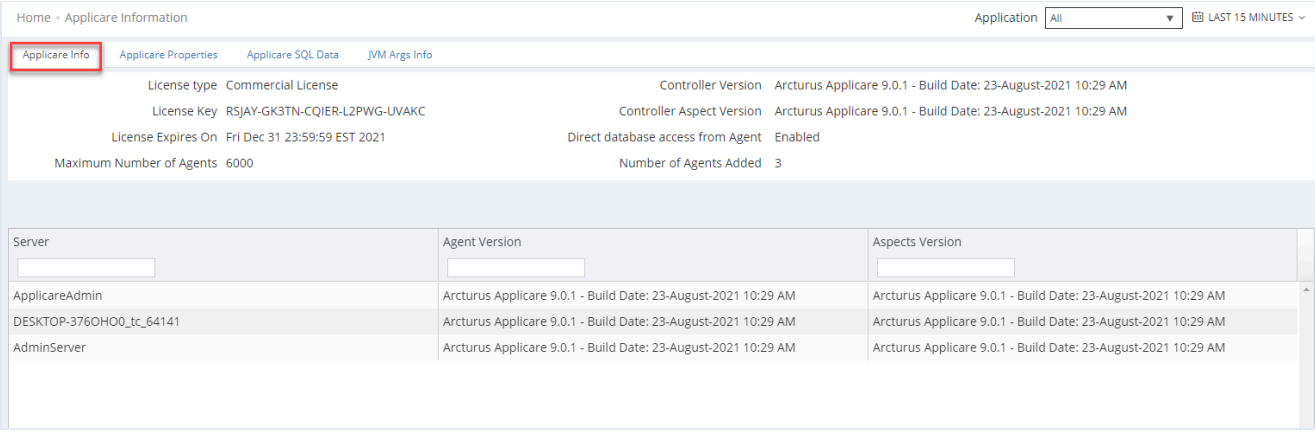
CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Applicare Information

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

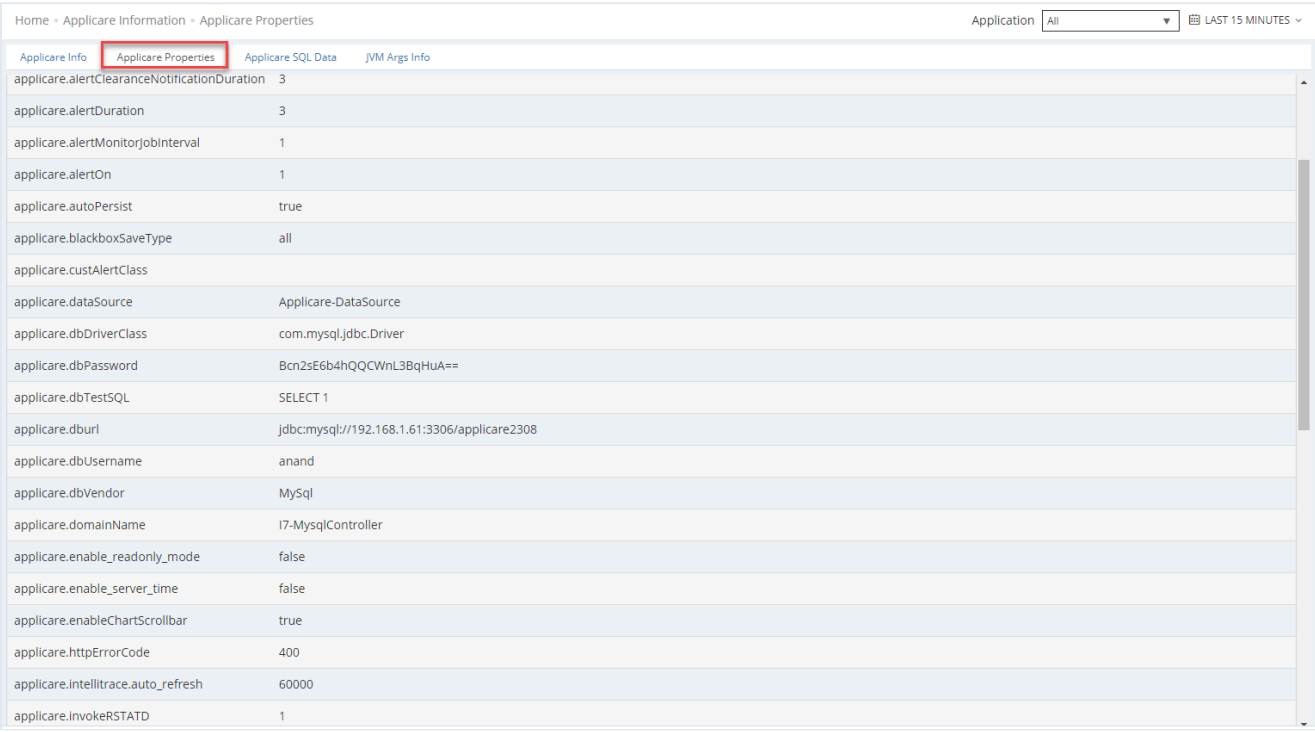
Applicare Info

Display the information of the Applicare Controller and agent that has been added to the Applicare Controller and license details.



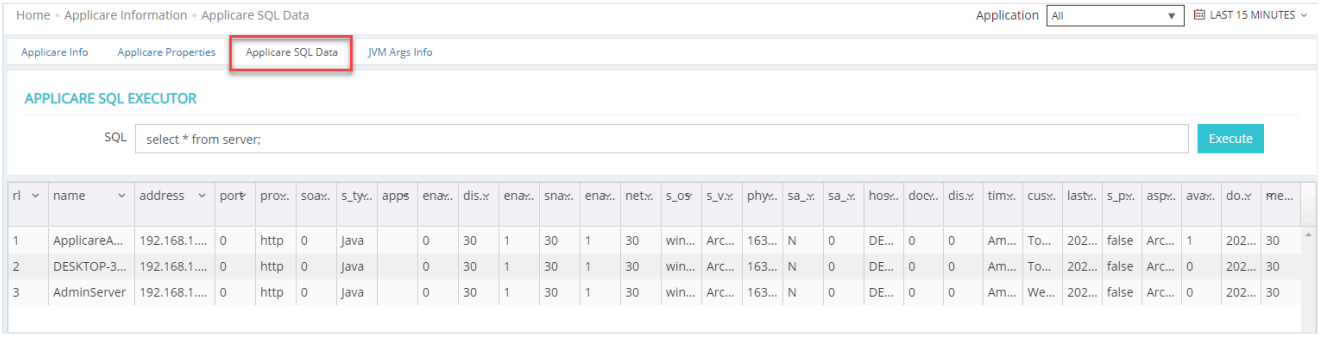
Applicare properties

This screen displays details about the Applicare Controller, including the database name, log file path, DB URL, and DB DriverClass, among other information.



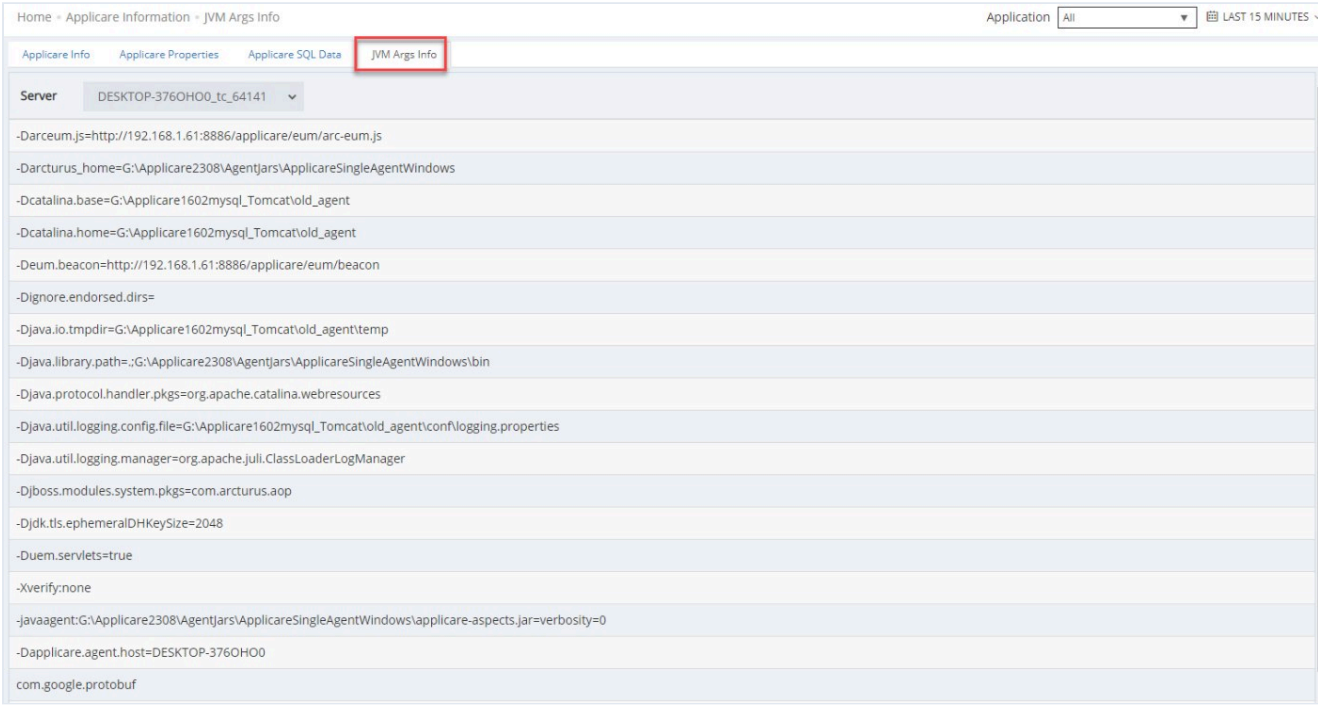
Applicare SQL Data

You can execute database queries directly from this screen. e.g.- you can run queries related to the agent server in the Applicare Controller.



JVM Arguments Info

Shows the Java arguments used by the server within the Applicare controller.



REST API

https://helpdesk.arcturustech.com/hc/en-us/articles/7644666377357

Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

Compressing log files on Windows

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

1. Create a **.bat** file in the directory where the log folder is located.

2. Edit that **.bat** file, paste the following contents into it, and then save the changes.

```bat
@echo
for /f "tokens=2 delims==" %%a in ('wmic OS Get localdatetime /value') do set "dt=%%a"
set "YY=%dt:~2,2%" & set "YYYY=%dt:~0,4%" & set "MM=%dt:~4,2%" & set "DD=%dt:~6,2%"
set "HH=%dt:~8,2%" & set "Min=%dt:~10,2%" & set "Sec=%dt:~12,2%"

set "fullstamp=%MM%-%DD%-%YYYY%_%HH%-%Min%-%Sec%"

echo compressed file to zip that folder
set path=%cd%
:: uncomment the below line enter the path if the bat file is in different location
rem set path=

cd %path%
zip -r compressed_%fullstamp%.zip *.* -x "*.zip" "*.bat"
```

3. Running the **.bat** file will compress all the files within the folder.

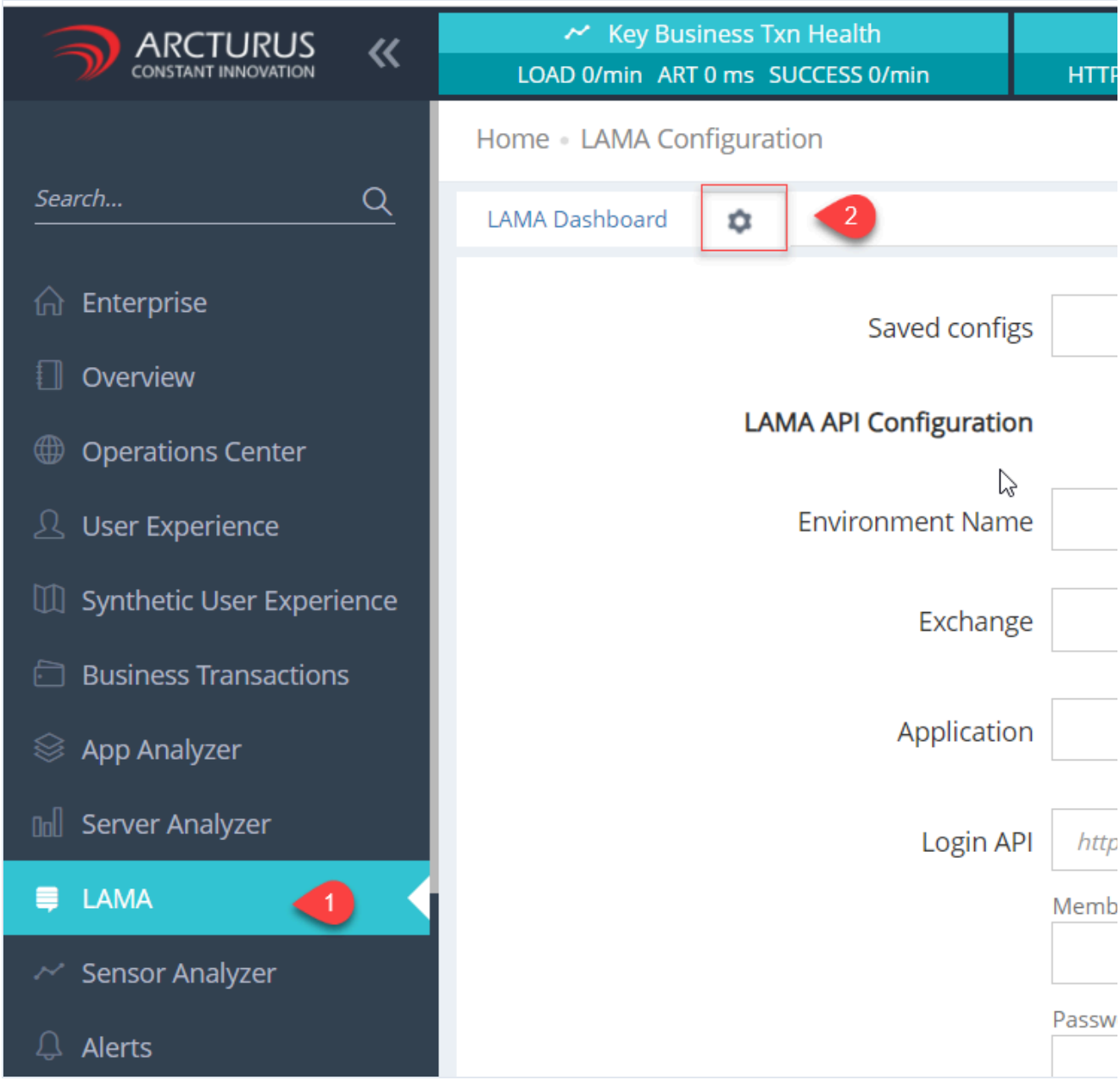
Monitoring with Applicare

CONFIGURATION GUIDES › ADMINISTRATION & SECURITY

LAMA configuration in Applicare

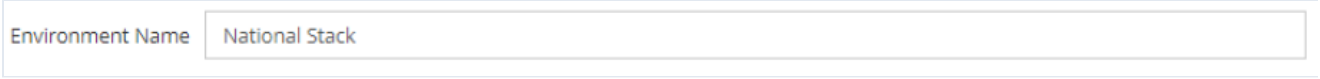
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

1. Log in to **Applicare console.** Go to **LAMA** menu and click the **settings** icon.

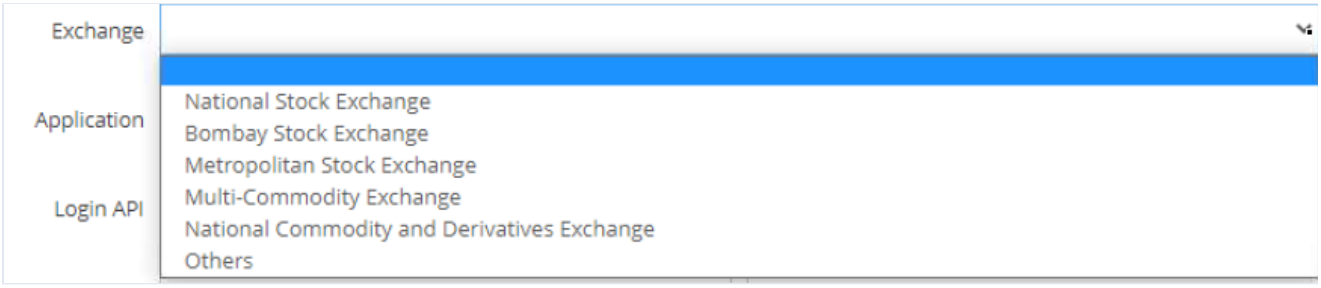


2. Follow the below steps to configure LAMA API.

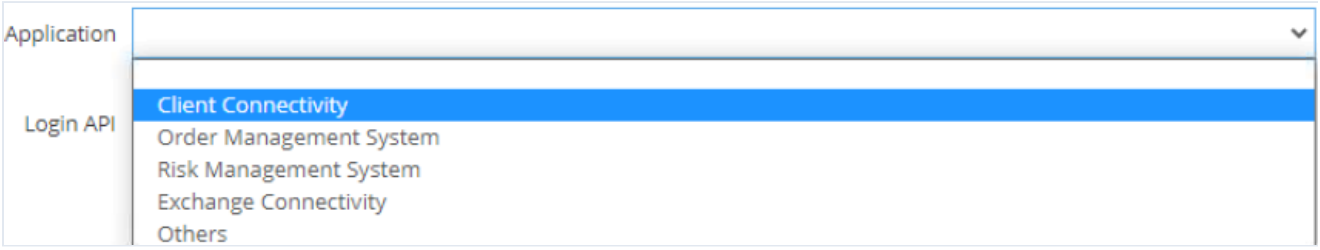
a. Enter the required name in the **Environment Name** text box.



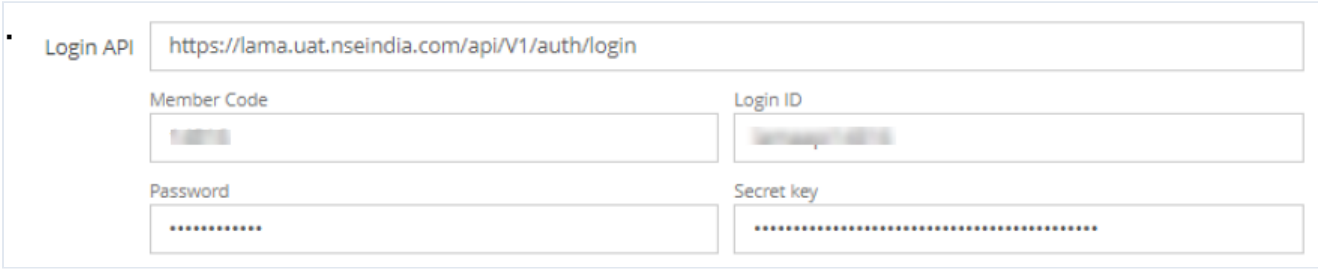
b. Click on **Exchange dropdown** and choose the desired **stock exchange**.



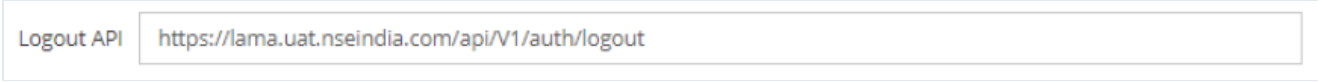
c. Click on **Application dropdown** and select any required application.



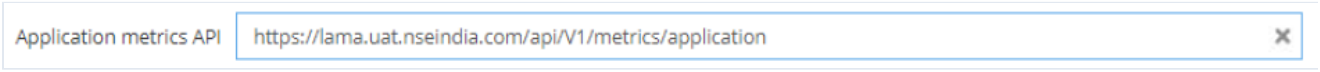
d. Enter the **API login URL** and your login credential.



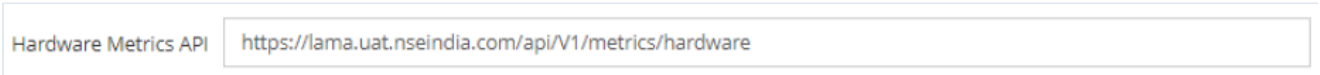
e. Enter the **API logout URL**



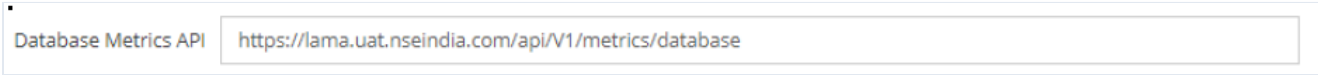
f. Enter the **Application metrics API URL**.



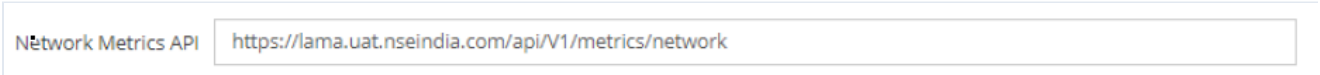
g. Enter the **Hardware Metrics API URL**.



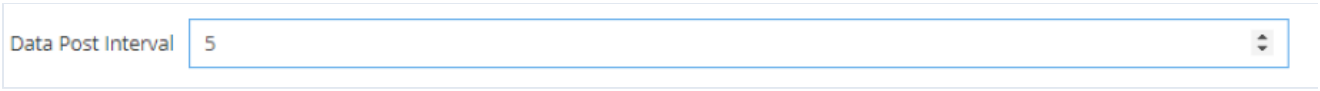
h. Enter the **Database Metrics API URL**.



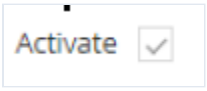
i. Enter the **Network Metrics API URL**.



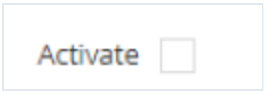
j. In the **Data Post Interval** textbox, Enter the time interval for posting the data.



k. To post data, You need to check the "Activate" checkbox.



To avoid posting data, Uncheck the checkbox.



l. Choose the **application** to which you want to post data from the **Assign Application** box.



m. Select the **application** for posting network data from the **Assign SNMP Application** box.



n. Enter the index details for APPs in the "SNMP Index Details" section and click the "Save" button..

INERR - IfInErr Index

OUTERR - IfOutErr Index

IN - IfInOctets Index

OUT - IfOutOctets Index

SP - IfSpeed Index



e.g.

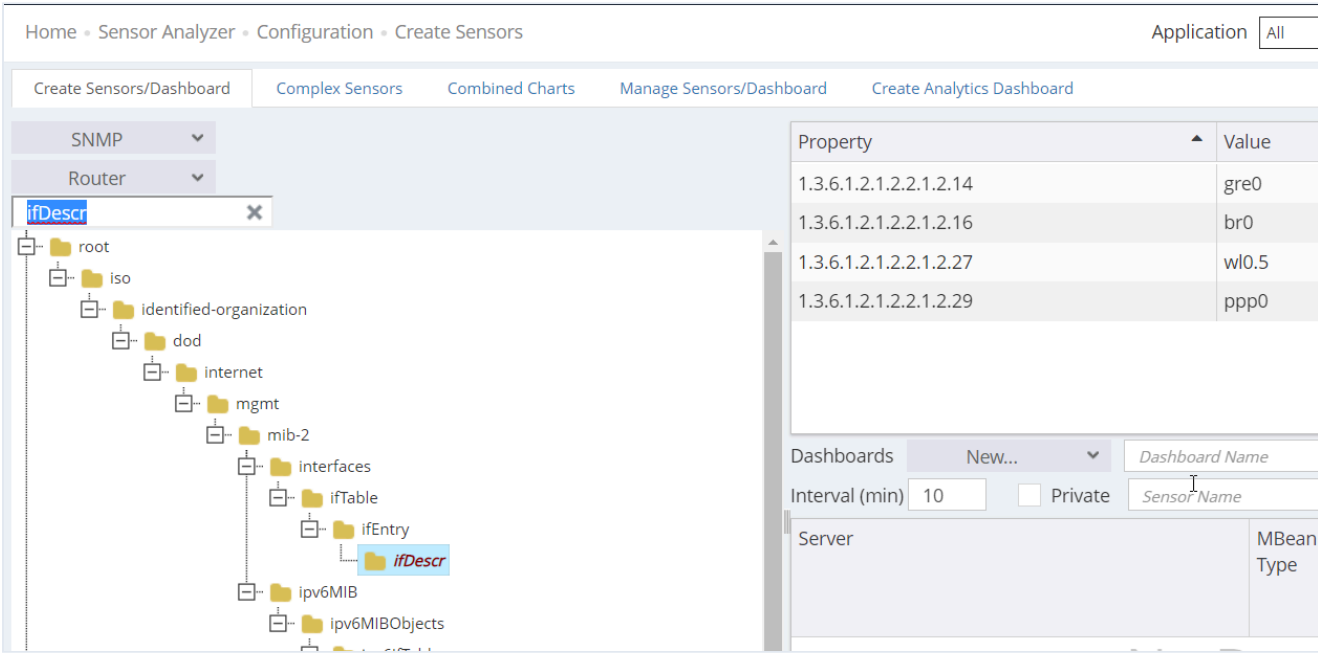
1. For Single App entry : {"Router":"INERR_16~~OUTERR_16~~IN_16~~OUT_16~~SP_16"}

2. For Multiple App entry

{"Router":"INERR_16~~OUTERR_16~~IN_16~~OUT_16~~SP_16",
"Router2":"INERR_29~~OUTERR_29~~IN_29~~OUT_29~~SP_29"}

To find the index, follow these steps:

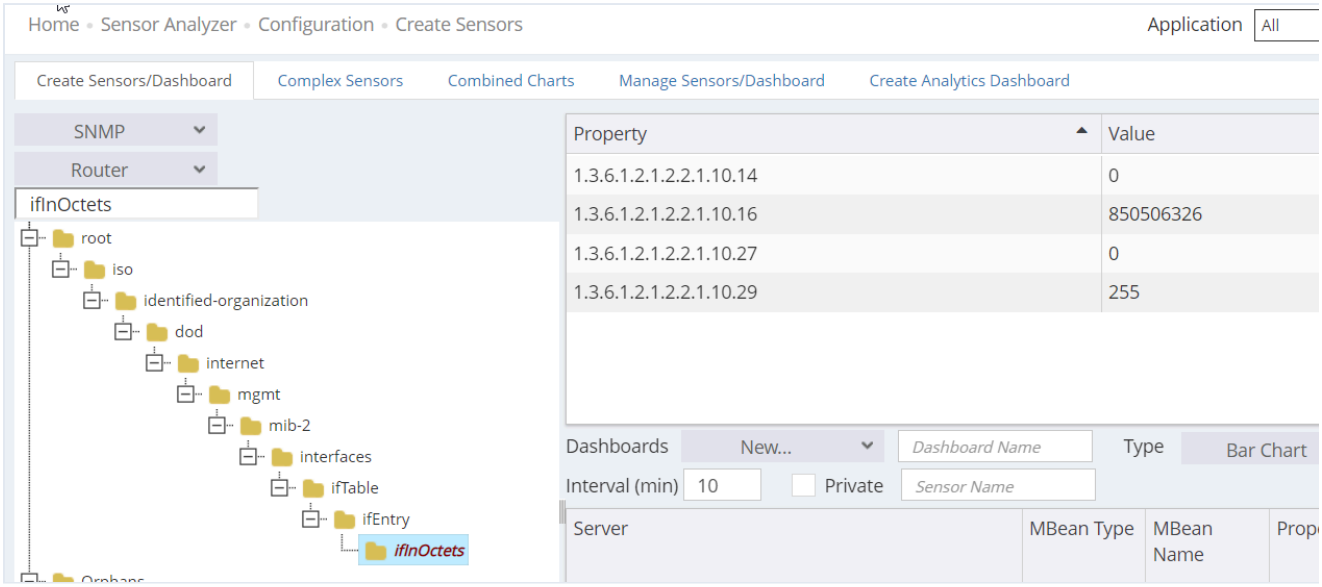
- The index refers to the interface. To locate the index for a specific interface, go to the Sensor Analyzer screen.
- Set the type to SNMP and select the server.
- In the search box, type 'ifDescr' and select that field from the left-hand side.
- On the right-hand side, you will see a list of interfaces and their OIDs. The last value in the OID is the index for that interface. Typically, this index will be the same for the IN, OUT, and SP values.

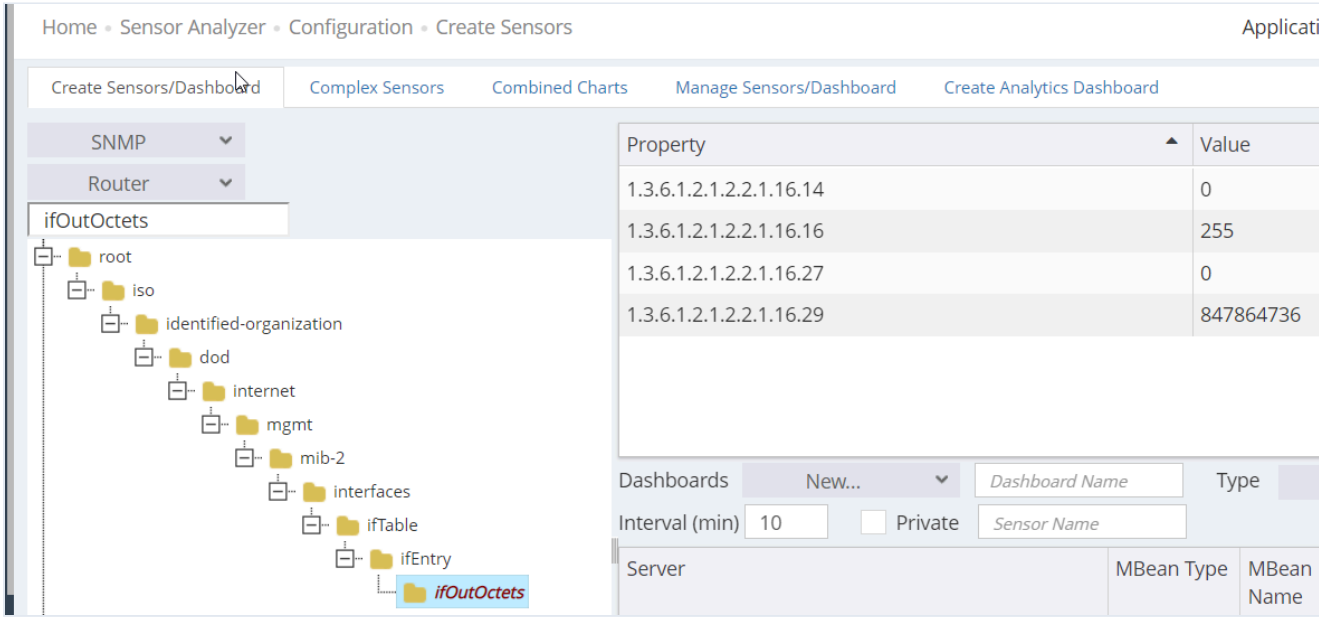


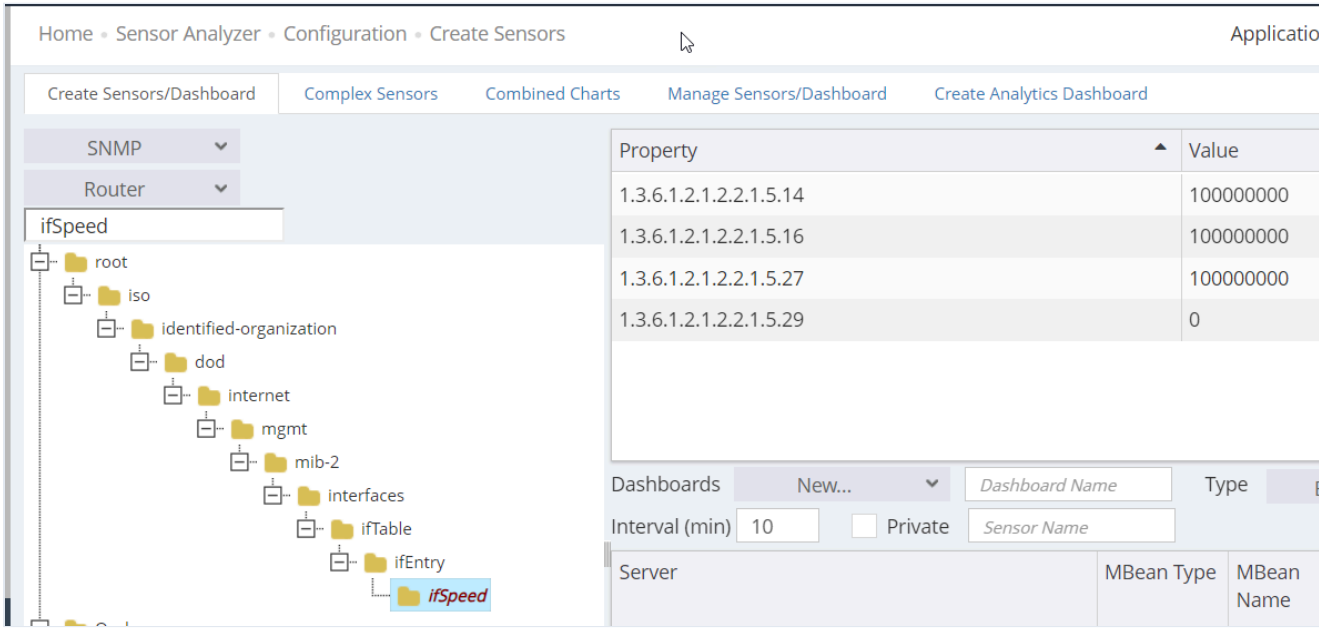
The index for **gre0** interface is **14**

The index for **br0** interface is **16**

Likewise we can find the index for **IN, OUT and SP** values.


Home • Sensor Analyzer • Configuration • Create Sensors
Application All
Create Sensors/Dashboard
Complex Sensors
Combined Charts
Manage Sensors/Dashboard
Create Analytics Dashboard
SNMP
Router
ifInOctets
root
iso
identified-organization
dod
internet
mgmt
mib-2
interfaces
ifTable
ifEntry
ifInOctets
Property
Value
1.3.6.1.2.1.2.2.1.10.14
0
1.3.6.1.2.1.2.2.1.10.16
850506326
1.3.6.1.2.1.2.2.1.10.27
0
1.3.6.1.2.1.2.2.1.10.29
255
Dashboards
New...
Dashboard Name
Type
Bar Chart
Interval (min)
10
Private
Sensor Name
Server
MBean Type
MBean Name


Home • Sensor Analyzer • Configuration • Create Sensors
Create Sensors/Dashboard
Complex Sensors
Combined Charts
Manage Sensors/Dashboard
Create Analytics Dashboard
SNMP
Router
ifOutOctets
root
iso
identified-organization
dod
internet
mgmt
mib-2
interfaces
ifTable
ifEntry
ifOutOctets
Property
Value
1.3.6.1.2.1.2.2.1.16.14
0
1.3.6.1.2.1.2.2.1.16.16
255
1.3.6.1.2.1.2.2.1.16.27
0
1.3.6.1.2.1.2.2.1.16.29
847864736
Dashboards
New...
Dashboard Name
Type
Interval (min)
10
Private
Sensor Name
Server
MBean Type
MBean Name


Home • Sensor Analyzer • Configuration • Create Sensors
Create Sensors/Dashboard
Complex Sensors
Combined Charts
Manage Sensors/Dashboard
Create Analytics Dashboard
SNMP
Router
ifSpeed
root
iso
identified-organization
dod
internet
mgmt
mib-2
interfaces
ifTable
ifEntry
ifSpeed
Property
Value
1.3.6.1.2.1.2.2.1.5.14
100000000
1.3.6.1.2.1.2.2.1.5.16
100000000
1.3.6.1.2.1.2.2.1.5.27
100000000
1.3.6.1.2.1.2.2.1.5.29
0
Dashboards
New...
Dashboard Name
Type
Interval (min)
10
Private
Sensor Name
Server
MBean Type
MBean Name

For Database Metrics

1. For oracle we need to add few custom values in the **LAMA Settings** in the **overview screen.**

PNAME_<VALUE>,TMBPS_<VALUE>

e.g.

PNAME - Process Name indicating Replication Availability

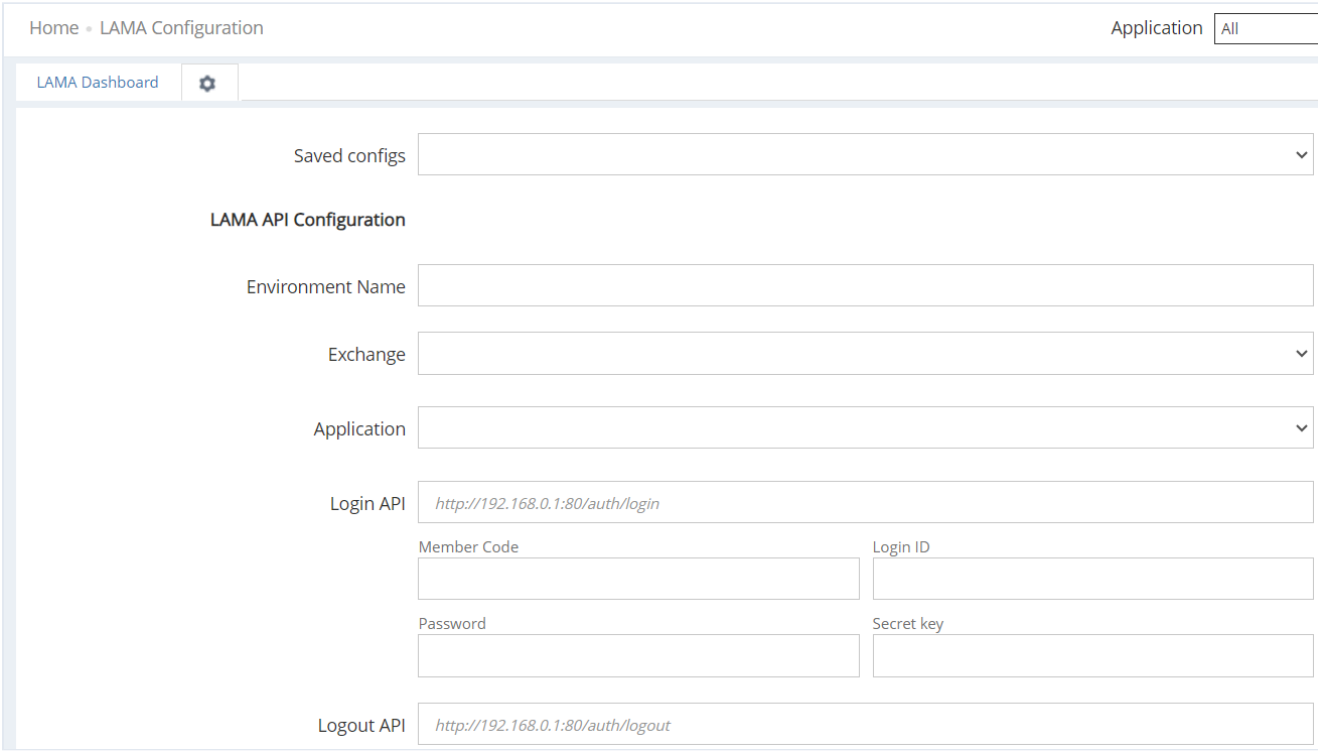
TMBPS - Total Megabits per second of internet speed

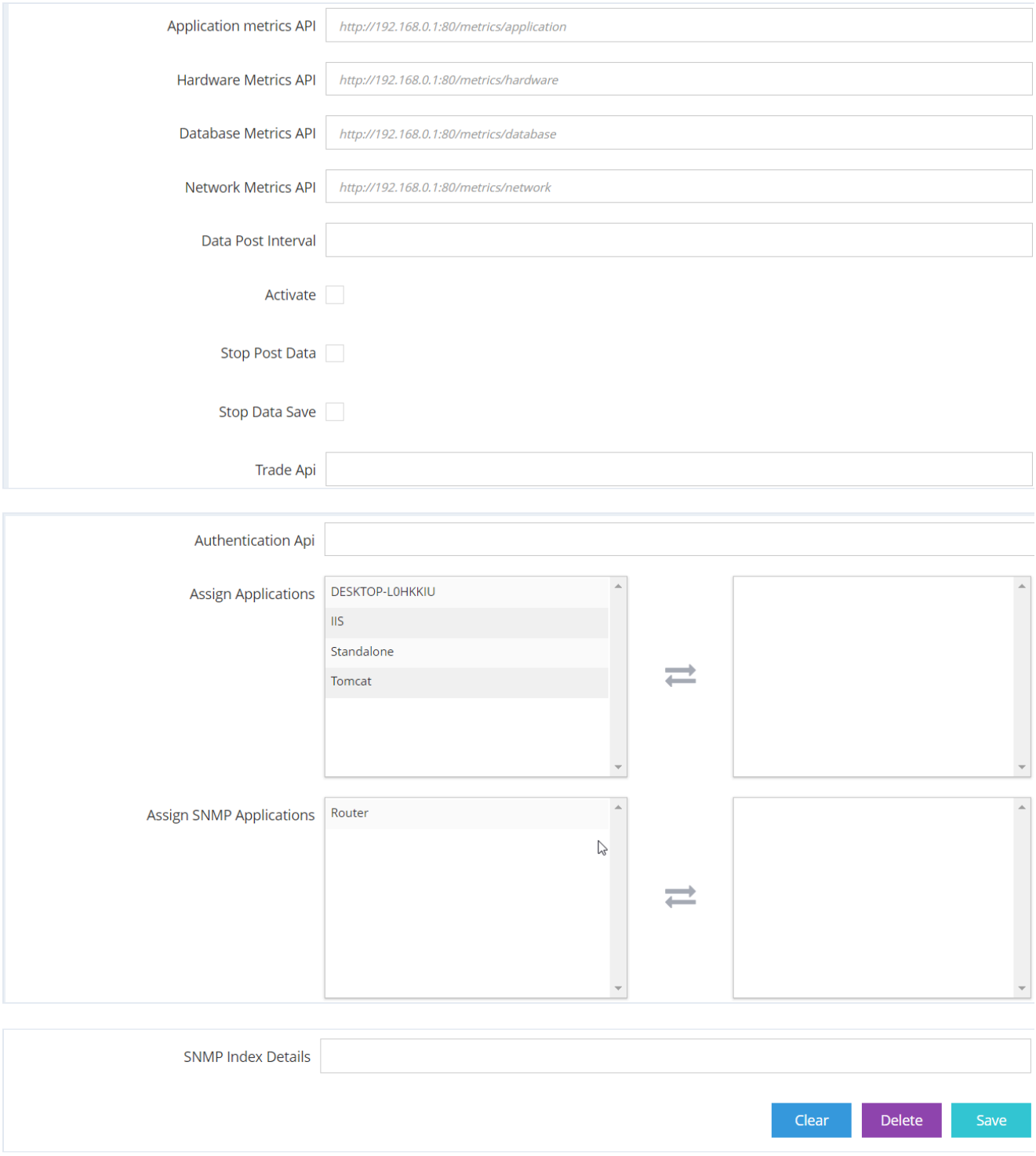
PNAME_MRP0,TMBPS_1000

MRP0 - It is the process name. If this process is not available then the replication is not active.

1000 - 1000 Megabits per second.

LAMA Configuration





Monitoring with Applicare

PART

Using Applicare

Dashboards, applications, transactions, infrastructure and AIOps features.

USING APPLICARE › DASHBOARDS

App Monitoring Dashboard

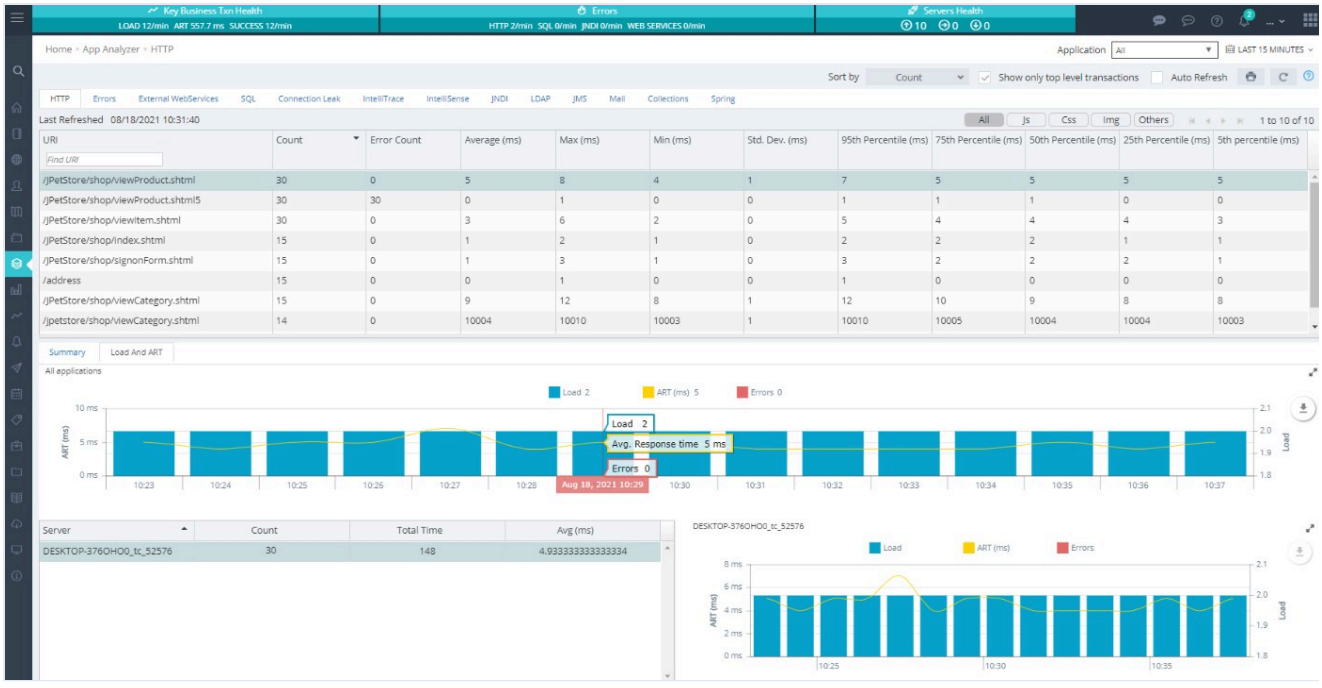
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Overview

The App Analyzer is designed to evaluate the performance of an entire enterprise application, providing comprehensive insights into various transactions, SQL queries and other metrics across all servers that support the application.

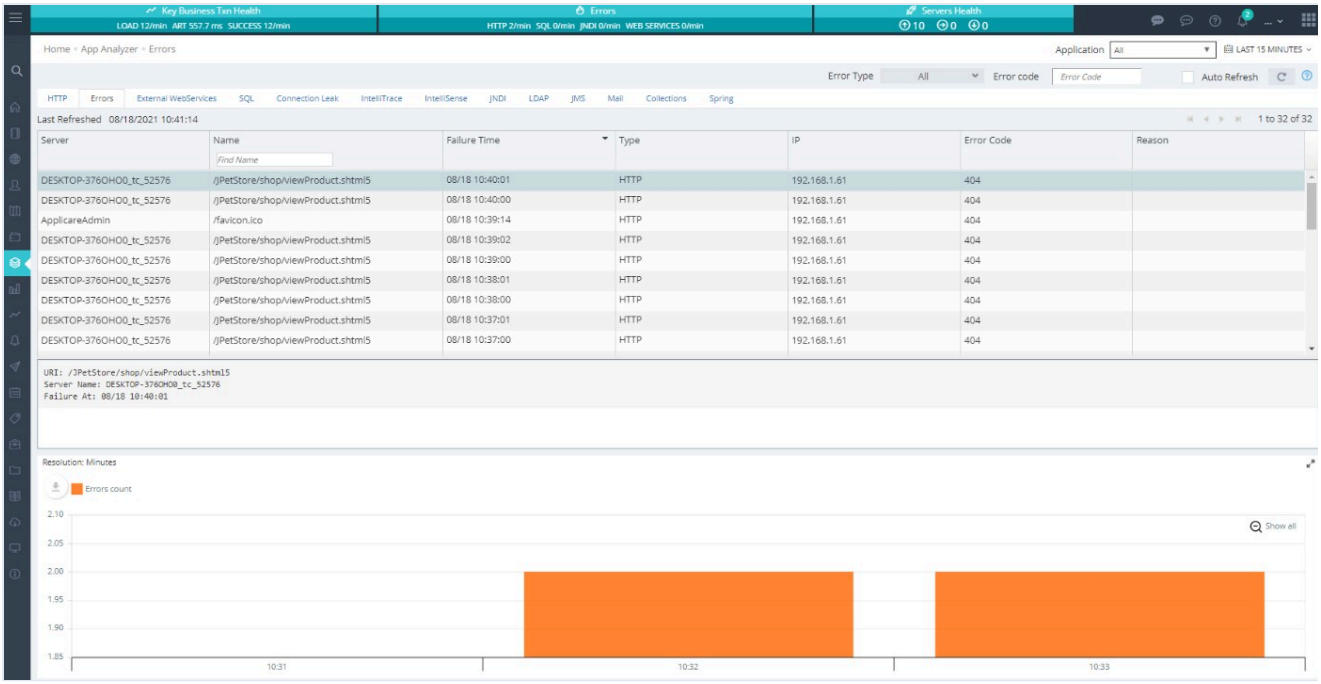
HTTP

The top table displays URI along with its performance metrics, including count, average and performance percentiles. Selecting a row will reveal the performance details for that URI within the chosen time period in the central chart. On bottom left, performance is shown on all the servers in the application and on bottom right is the chart showing comparative performance across all servers.



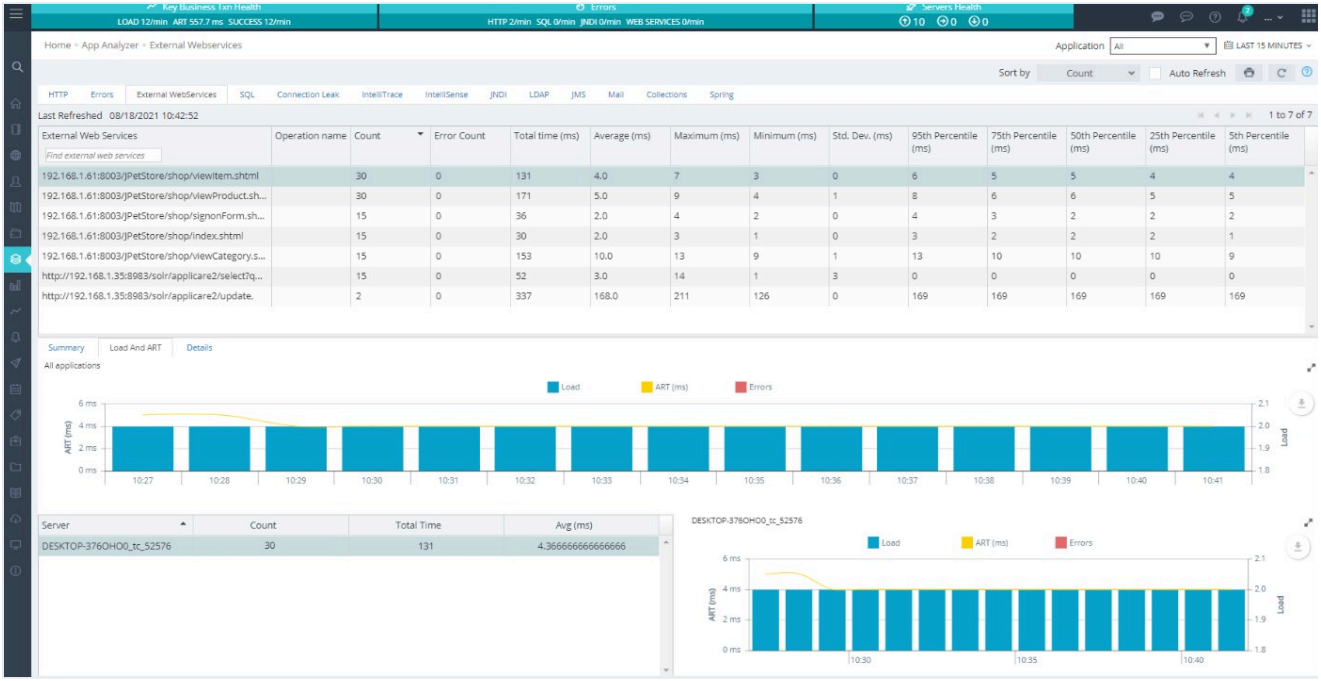
HTTP Errors

The HTTP Errors tab provides details of HTTP errors and allows drilling down into any specific http error. Details available for each http error are server name, URI, failure time, IP of the user who experienced the error, HTTP error code (404, 500 etc.) and HTTP exception details. Exception stack is available in the middle text box on selecting a transaction row in the table on top.



External WebServices

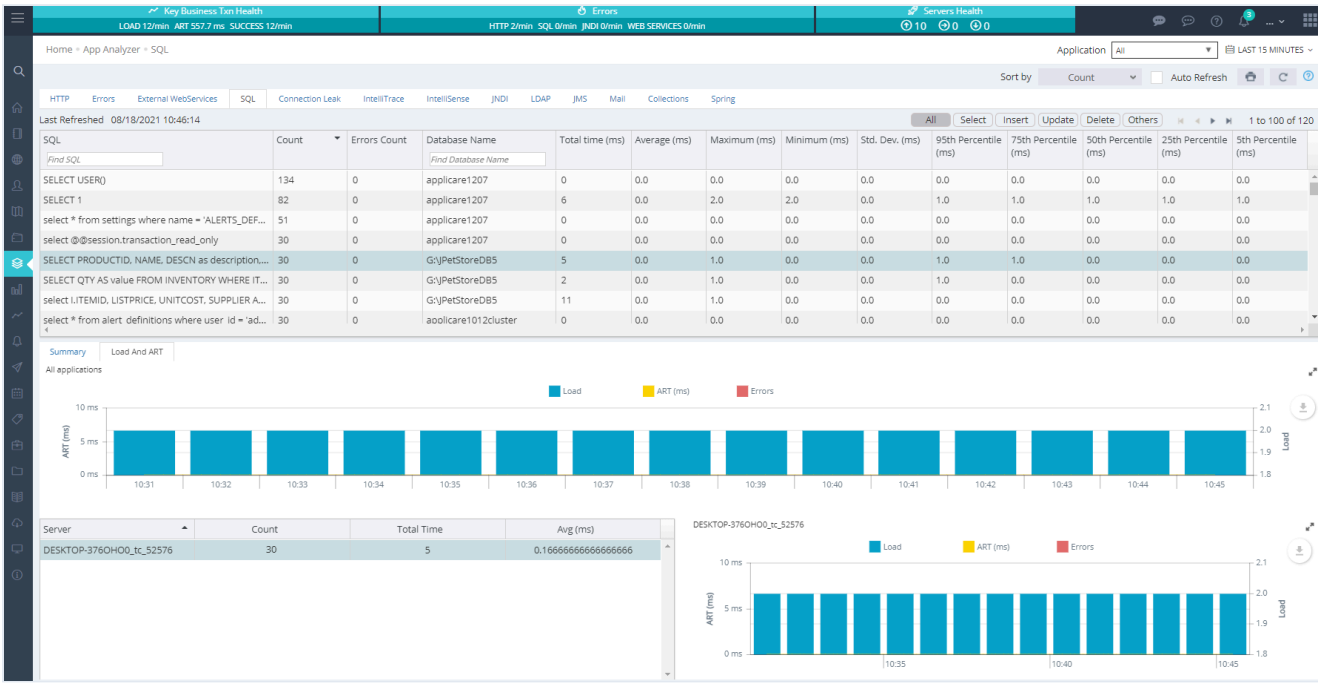
The External Web Services tab works very similar to HTTP tab, and provides performance data for external web service calls made from your application. Top table presents performance data in a tabular format, while the middle charts illustrate the performance of web services across all servers over time. Bottom left table shows performance on individual servers and bottom right chart either shows comparative performance or performance on an individual server if a server or row is selected in the bottom left table.



SQL

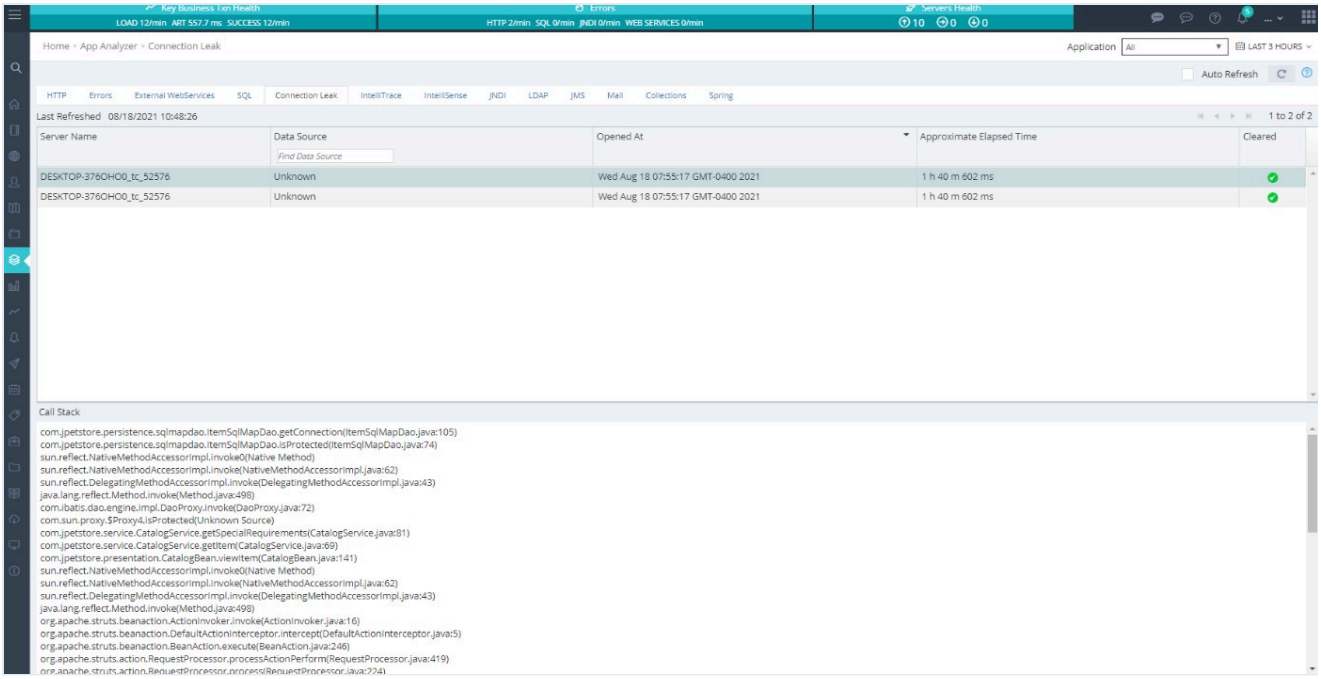
The SQL tab displays cumulative performance data for various SQL queries executed on the server within the app. Initially, a pie chart reveals the percentage of total time consumed by different SQL queries.

When you select a specific SQL query, the middle chart will present its cumulative performance across the application, while the bottom right chart provides a comparative view of performance across all servers. The bottom left section details performance on individual servers, allowing you to a server and view its performance in the bottom right chart.



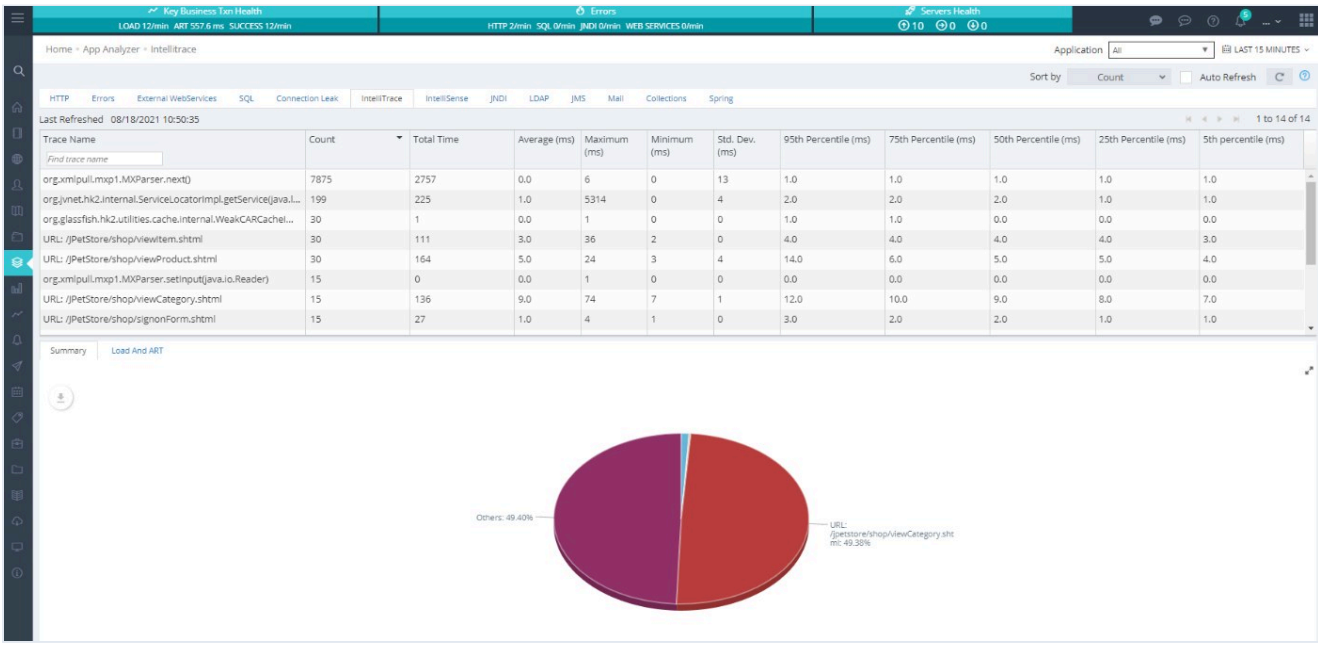
Connection Leak

The Connection Leak tab shows detailed information about each connection leak, including the code responsible for the leak, the associated datasource, the time the connection was opened, and the duration for which it has been held.

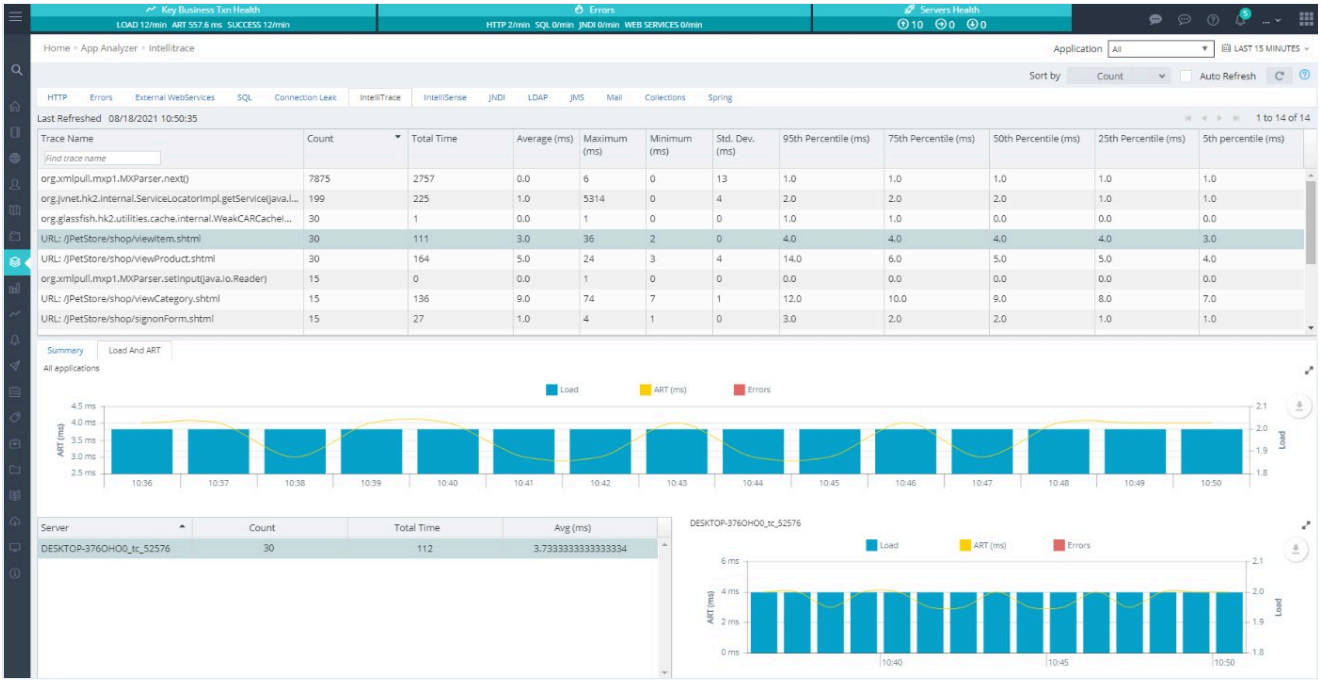


IntelliTrace

IntelliTrace tab shows various transactions including java, web sphere and HTTP calls flowing through the system. Bottom pie chart shows % based on total time consumed by a particular transaction and shows the impact of the transaction in grand schema of things.

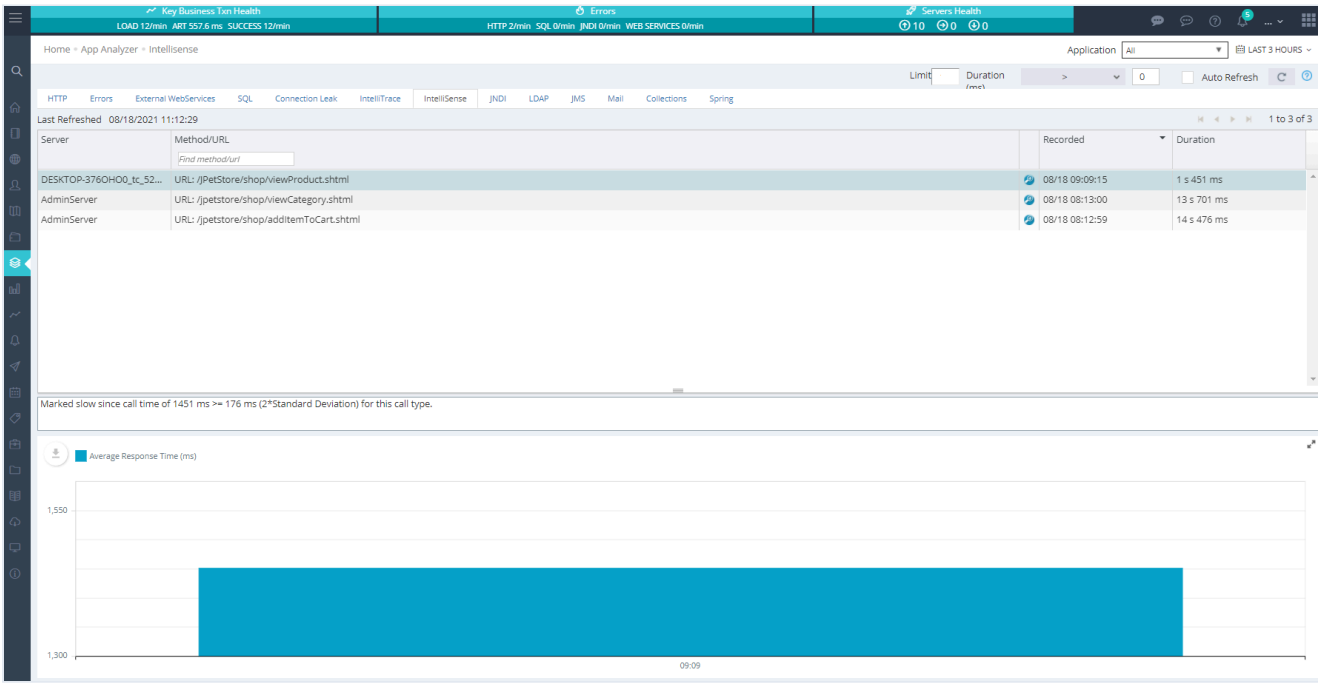


When you select a transaction in top table, the middle chart shows cumulative performance. Bottom left table shows transaction's performance across servers. Bottom right chart shows comparative performance or performance on a particular server if a sever row is selected in the table on the left.



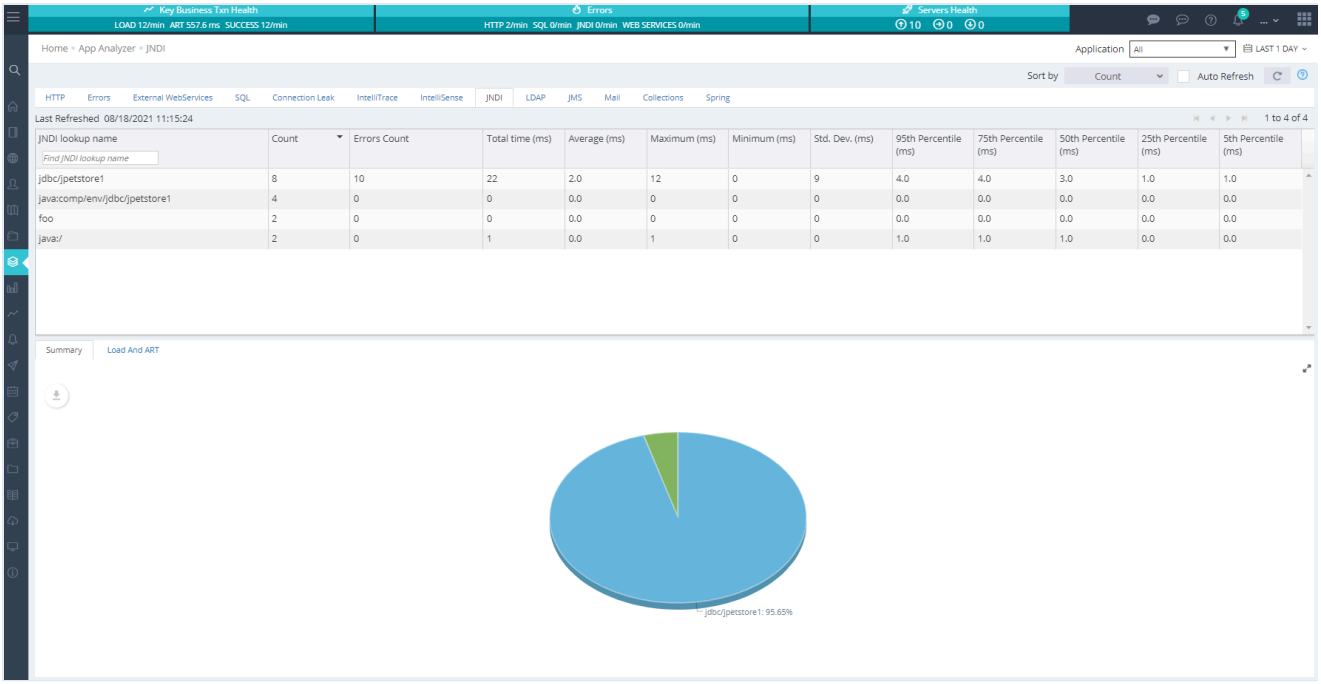
IntelliSense

IntelliSense automatically collects in-depth performance data for transactions that exceeds predefined criteria such as those with duration more than a twice the standard deviation for the specified category call type. For detailed transaction performance stats, simply double-click the corresponding row.

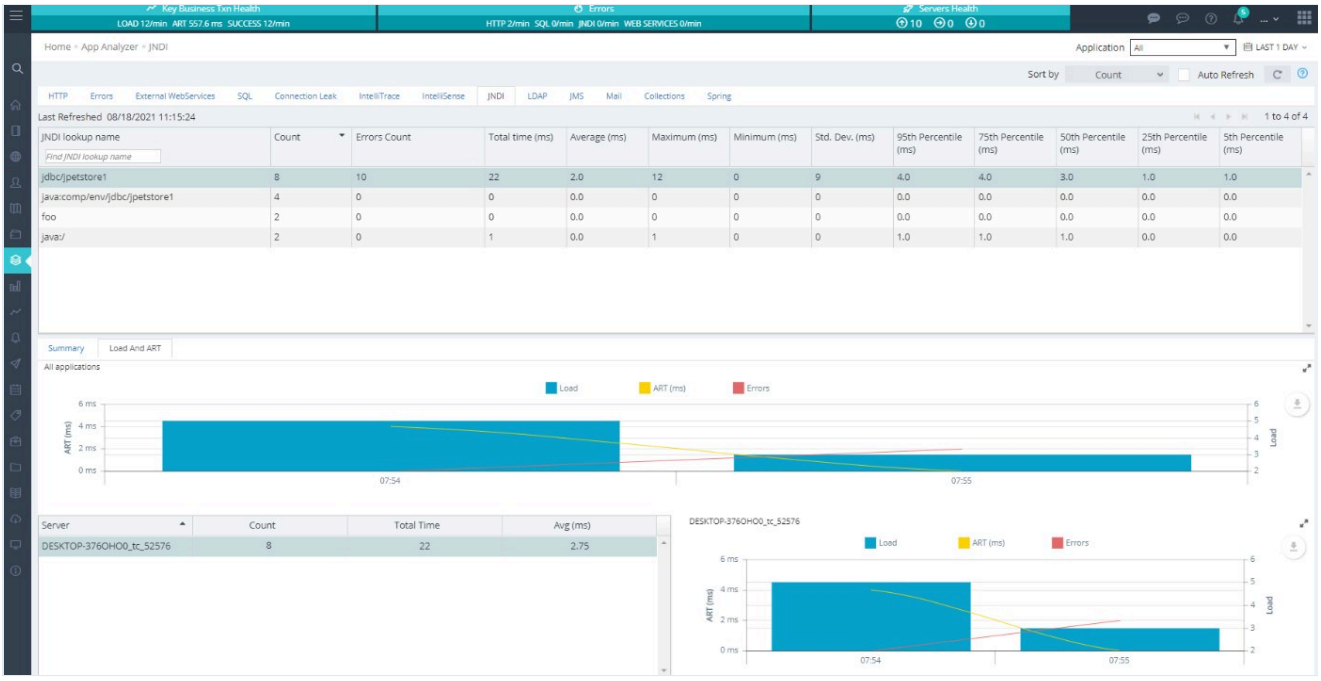


JNDI

The JNDI tab displays performance details for JNDI lookups. The initial view represent a pie chart with illustrating the percentage of time taken for these lookups.



When you select a specific row, detailed information will be displayed in both the middle and bottom tables/charts.



Monitoring with Applicare

USING APPLICARE › DASHBOARDS

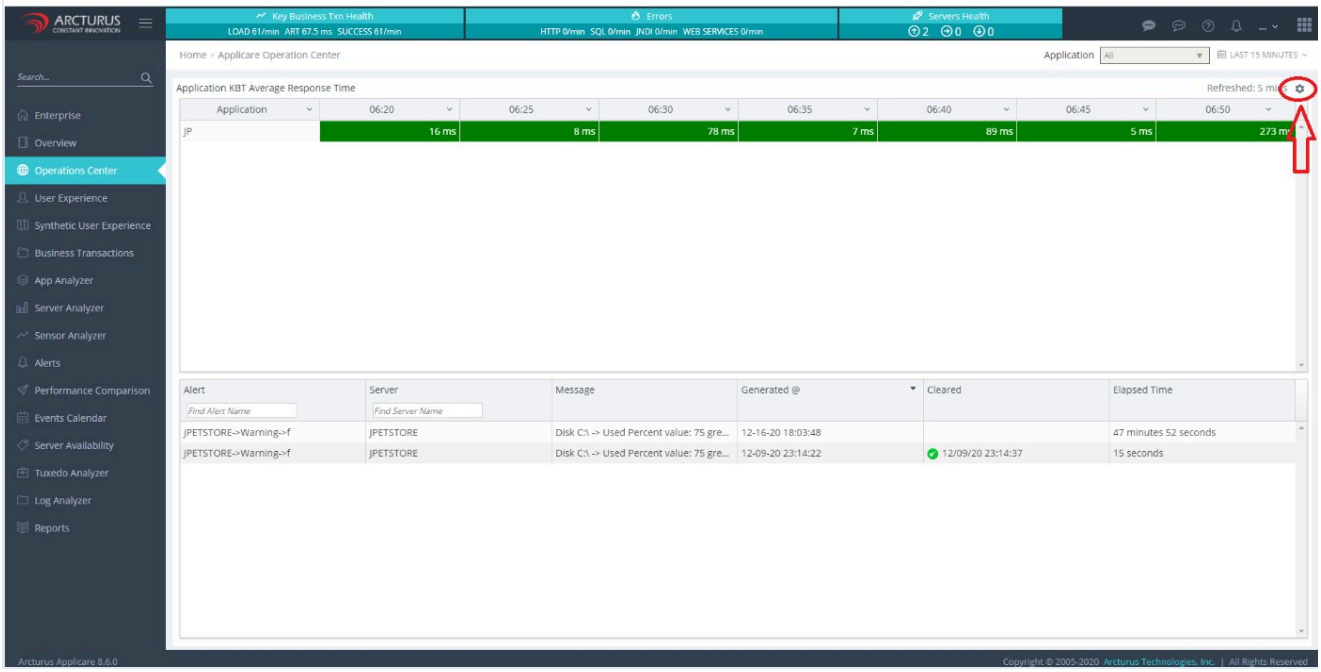
Operations Center

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The Operation center shows Average Performance Time (ART) of KBT performance based on applications. It displays from last 60 mins data and refreshes automatically every 5 mins.

Applications KBT transactions in last 60 minutes will not be displayed. At the bottom of the page grid, the grid shows alerts from the last 60 minutes for the Application listed in the KBT's ART grid at the top.

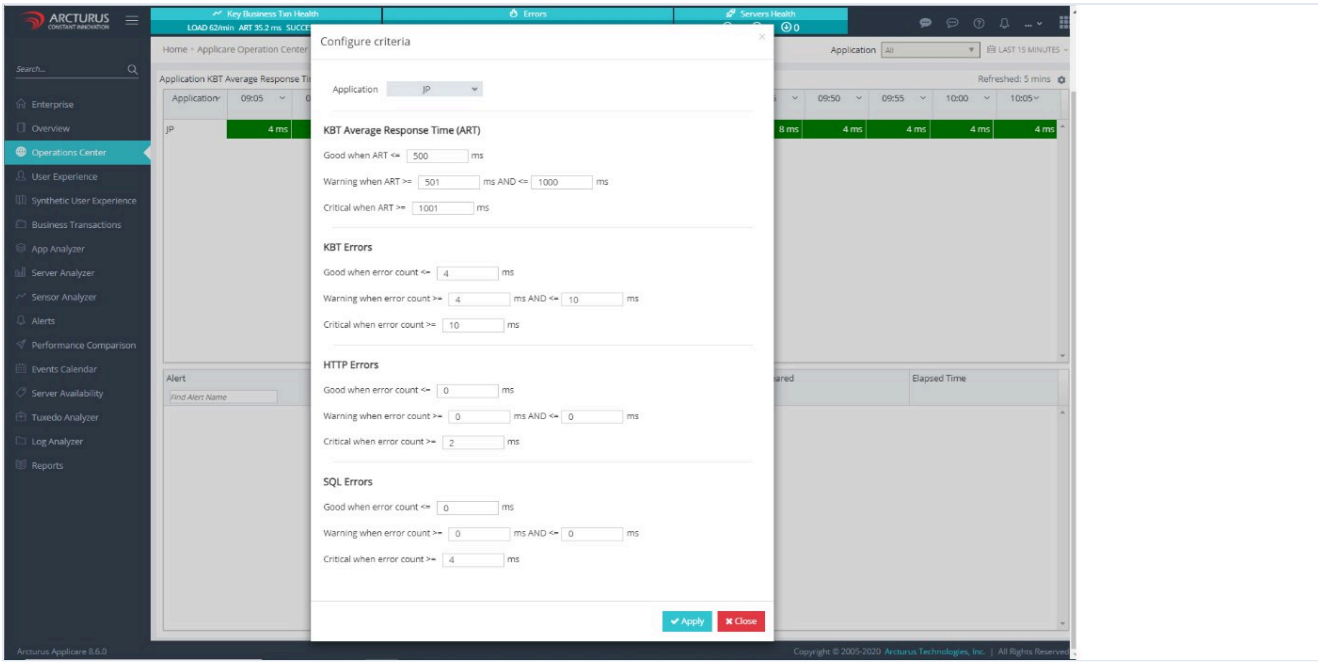
Configure criteria



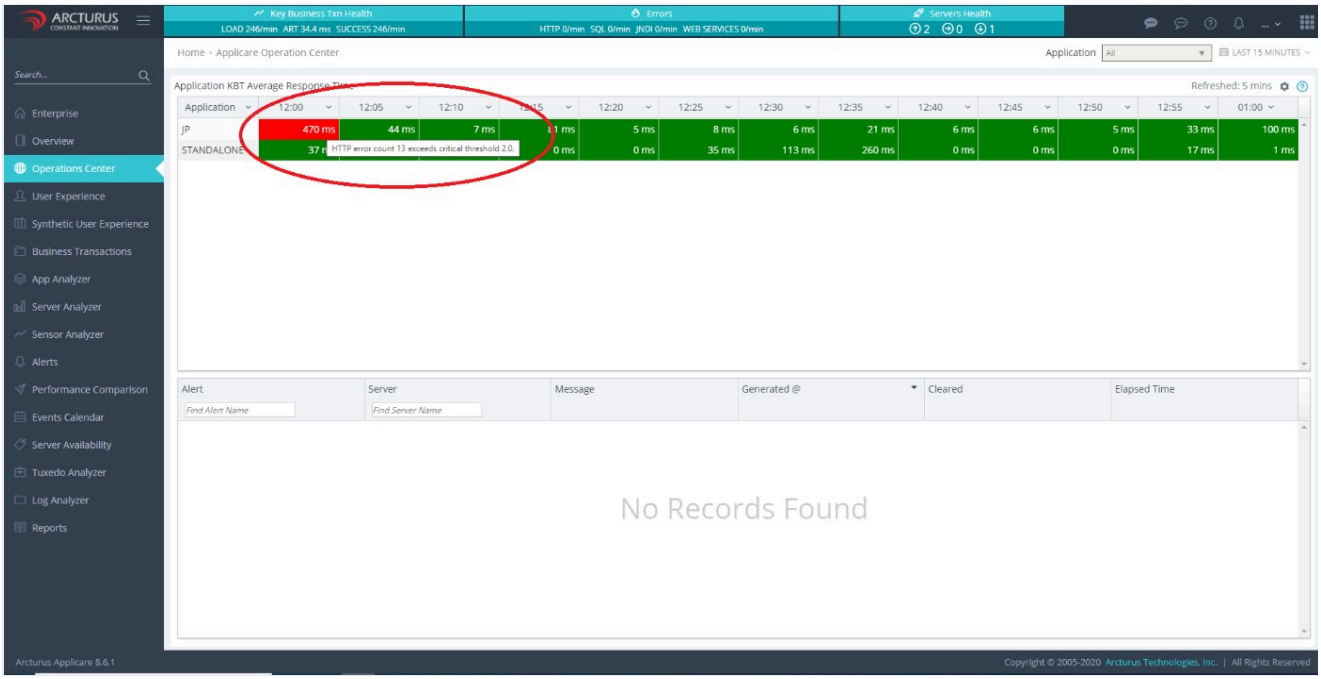
> [Screenshot 2 could not be downloaded during import — see migration report]

The "Configure Criteria" feature helps users easily differentiate between best, average, and worst performance using color coding. You can set color codes for KBT ART, KBT errors, HTTP errors, and SQL error counts. Values within the good range are shown in green, those in the warning range in yellow, and critical values in red.

You can configure KBT criteria for individual applications.



_[Screenshot 4 could not be downloaded during import — see migration report]_To view more details, hover your mouse over any color-coded cell in the KBT's ART grid and a tooltip will display the information.



_[Screenshot 6 could not be downloaded during import — see migration report]_By default, Application with more critical state are displayed on the top and least one are displayed at the bottom (i.e Apps with more Red are displayed on the top and least Red are displayed at the bottom). It helps to easily identity the apps in trouble.

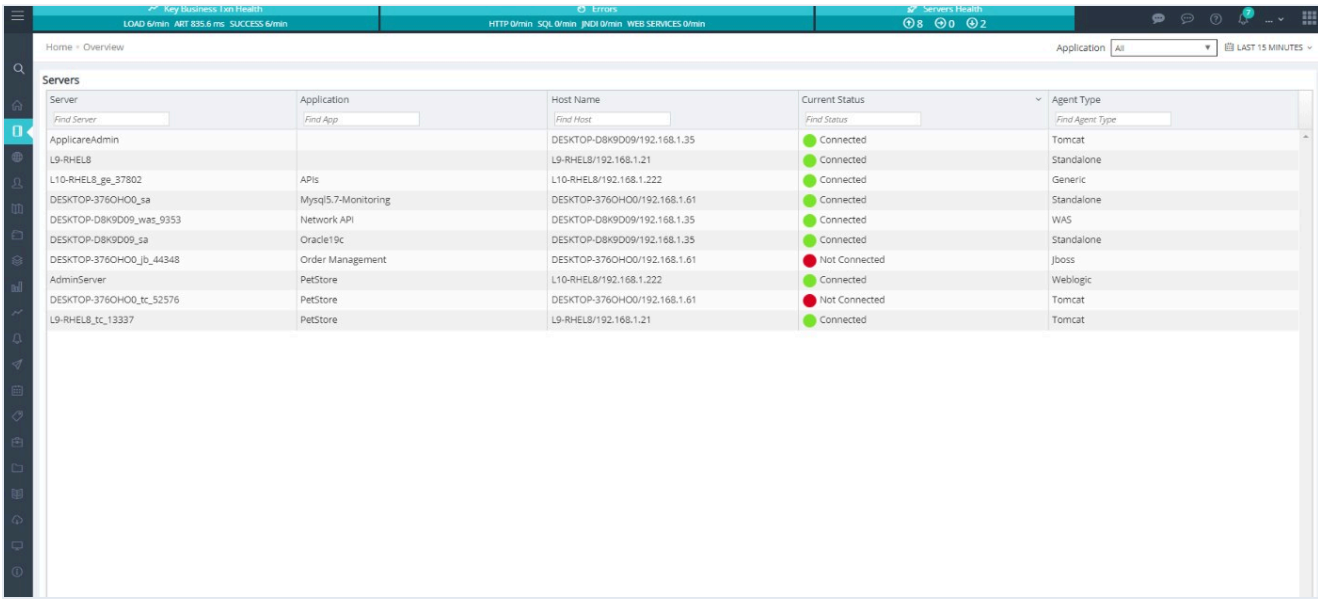
Monitoring with Applicare

USING APPLICARE › DASHBOARDS

Overview

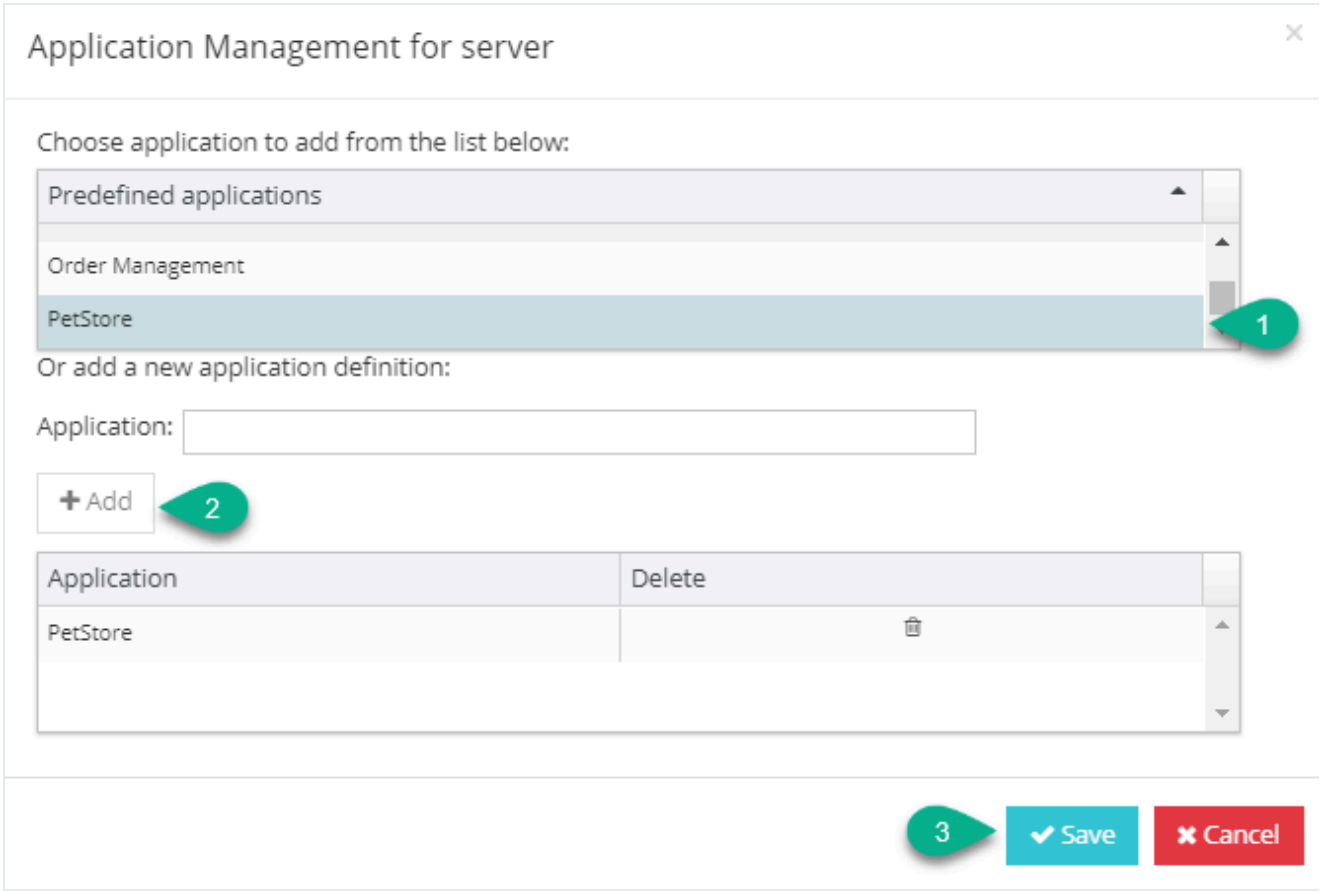
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The Overview tab provides information on the server's health & other details.



Servers

After adding an Applicare agent here, you can assign it to an application. An application serves as an administrative entity, enabling the grouping of data.



For Agent Auto Update and Restart Click here

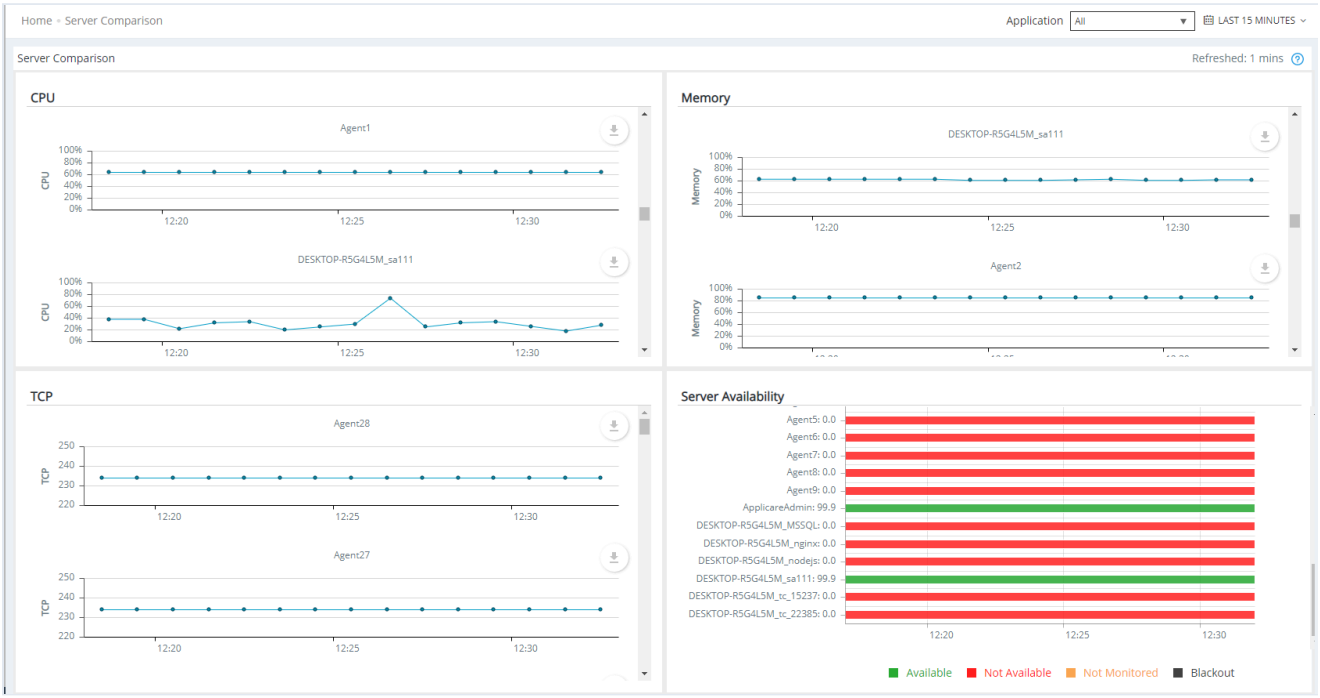
Monitoring with Applicare

USING APPLICARE › DASHBOARDS

Server Comparison

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Server Comparison offers a quick and convenient view of your server's CPU, memory, tcp and health details.



Monitoring with Applicare

USING APPLICARE › DASHBOARDS

Events Calendar

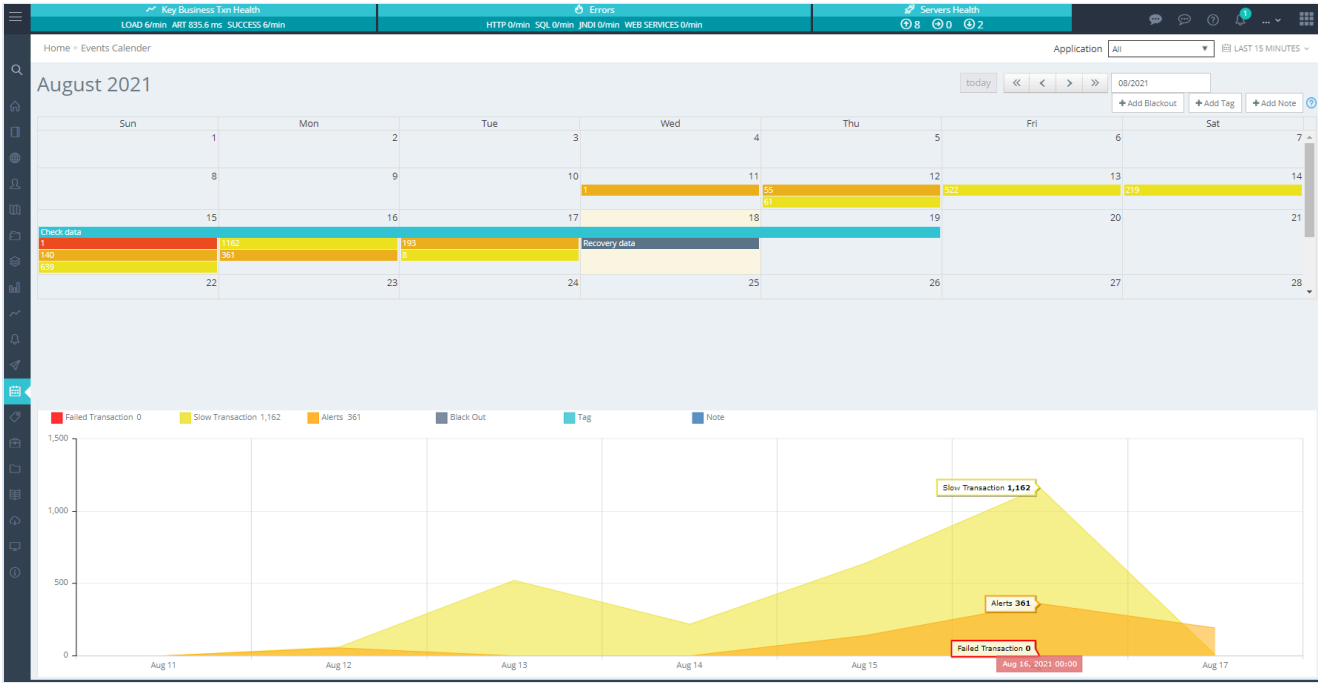
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Overview

Events Calendar provides a monthly overview of entire enterprise's performance or an application's performance in a calendar format. Here you easily see how the app performance was for a particular day or for the entire month. You see number of alerts, failed transactions, slow transactions etc on a particular day and for the entire month. This way you know how things are improving

- Reduction in number of alerts over time
- Transaction's performance improving
- Reduction in number of failed transactions
- Reduction in number of slow transactions

You can add notes on any day to keep track of relevant data such as an outage occured on Aug 18 due to an app update or why that outage occurred. This way you don't have to dig root cause details/discussions/notes in your email box, they can all be available here.



Monitoring with Applicare

USING APPLICARE › DASHBOARDS

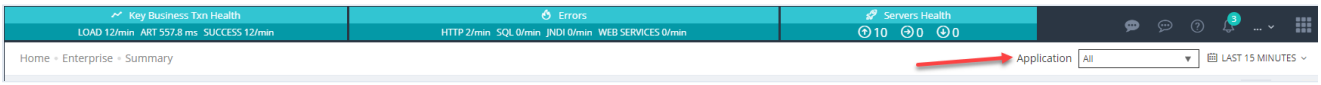
Enterprise

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Applicare Home

The Applicare home page displays the health and performance of the monitored applications. At top is the application selector, time selector, transactions score card and servers health status.

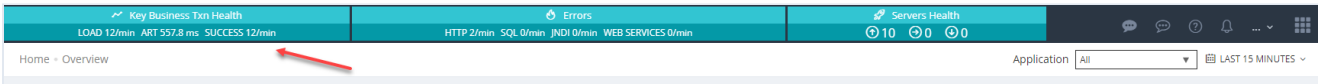
The application selector dropdown in the top right allows selection of an application to focus the scope of the analyzers. A server is assigned to an application at the time of adding it to Applicare. Application assignments can be managed from 'Overview' tab -> servers. A double click on an existing server allows editing the assigned application. Adding a server to an application allows grouping of data.



The time period selection on the top bar lets you narrow the focus to a specific time range.



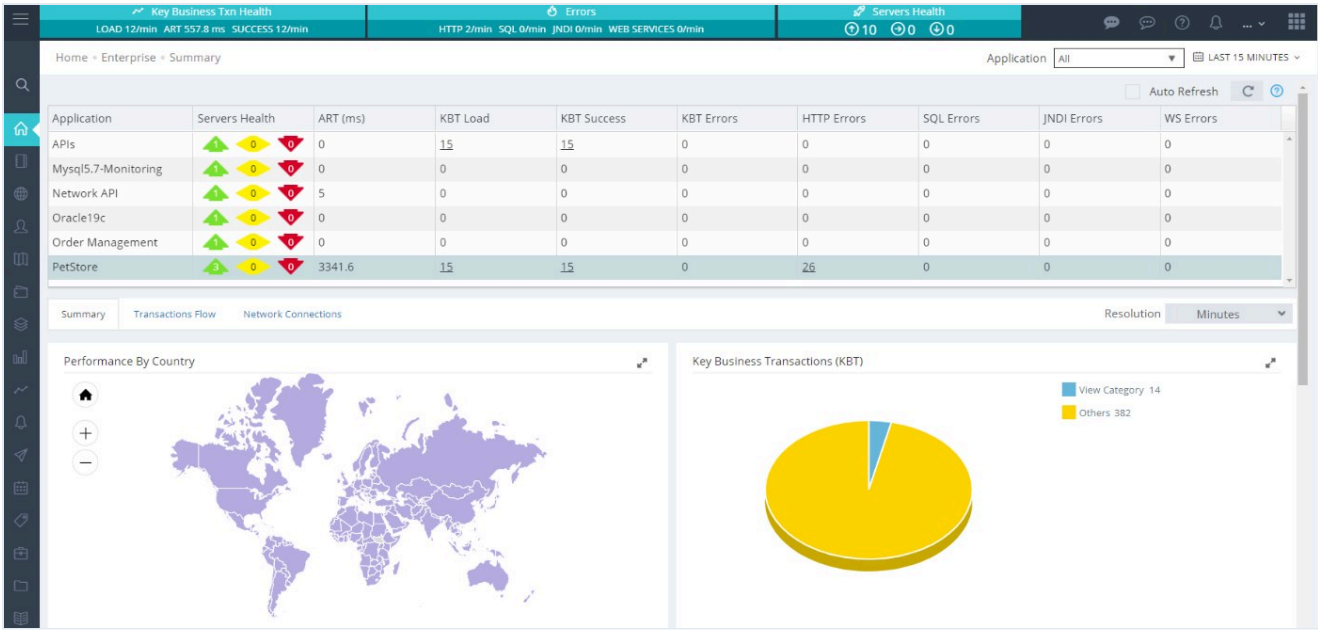
The **Business Transactions** & **Servers Health** shows the data for the selected application or all applications if 'All' is selected in application selection dropdown on left.



Below the top scorecards and selectors are various tabs that provide visibility into different aspects of the enterprise applications.

Enterprise Overview Dashboard

The **Enterprise Dashboard** provides a quick overview of the health of enterprise applications. It displays the average response time, business transactions, business transaction's load, success/errors, page load performance and health of monitored servers.



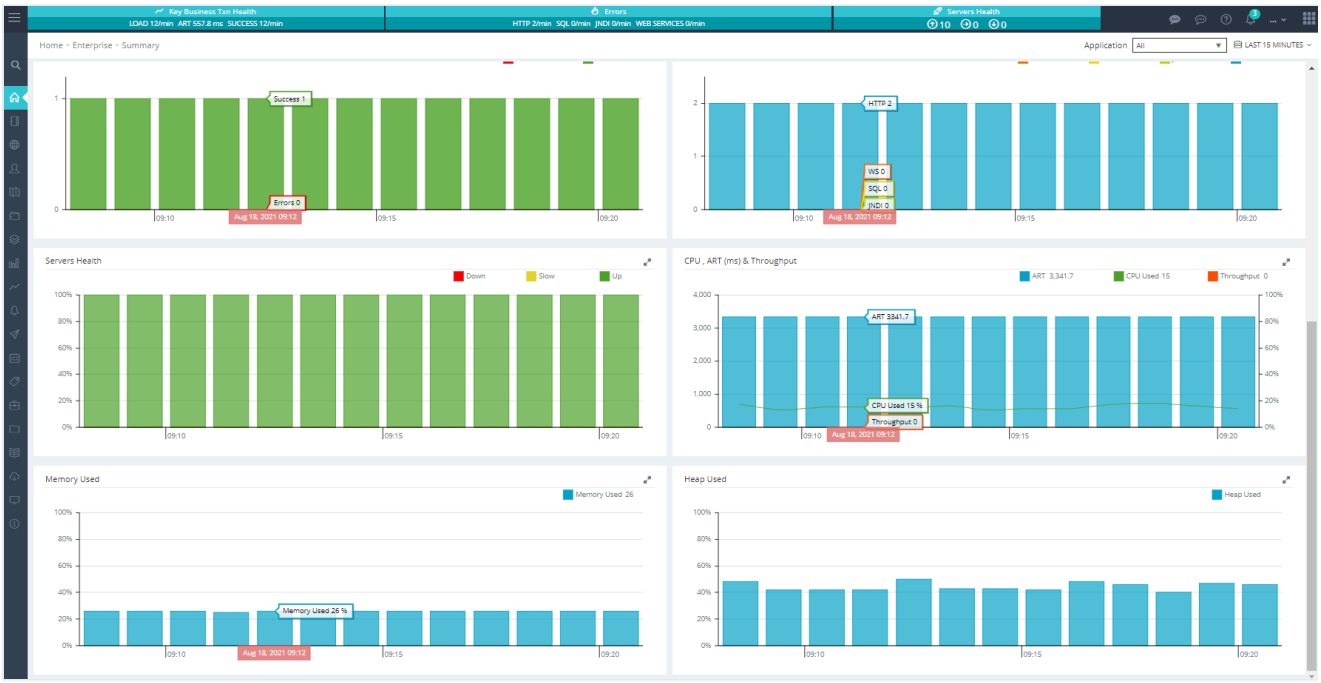
The **Enterprise Dashboard** offers a convenient to way identify performance issues and drill down into the details as needed:

- **Increase in average response time (ART)** for applications

- **Business transactions:** load, success rates, failures

- **Page Load Performance:** to show the user's experience from around the globe

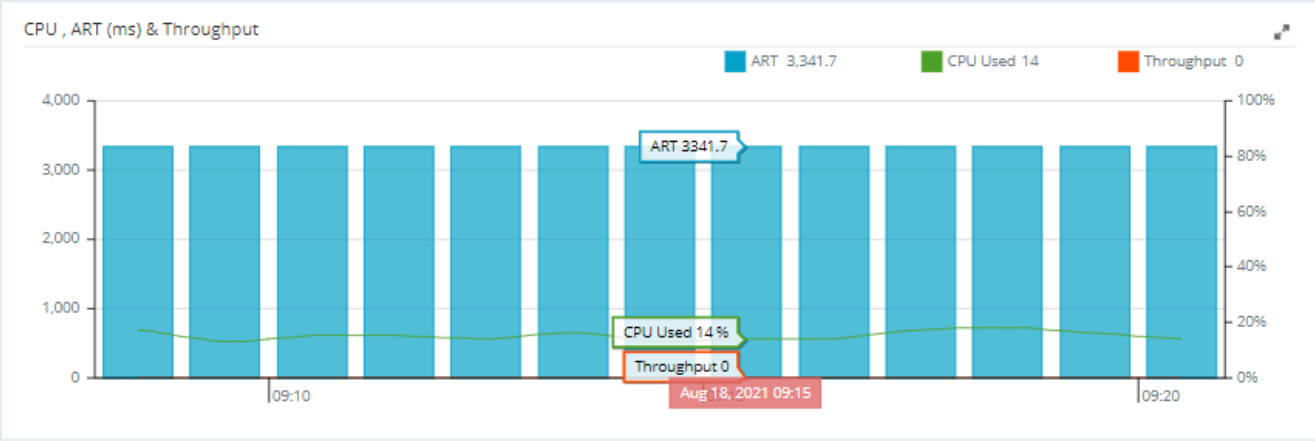
- **Servers Issues:** becoming slow or going down



On the Enterprise Dashboard, the top table allows you to select an application. Double-clicking an application table switches to the 'Transaction Flow' tab, which displays the flow and performance at each leg of the application.

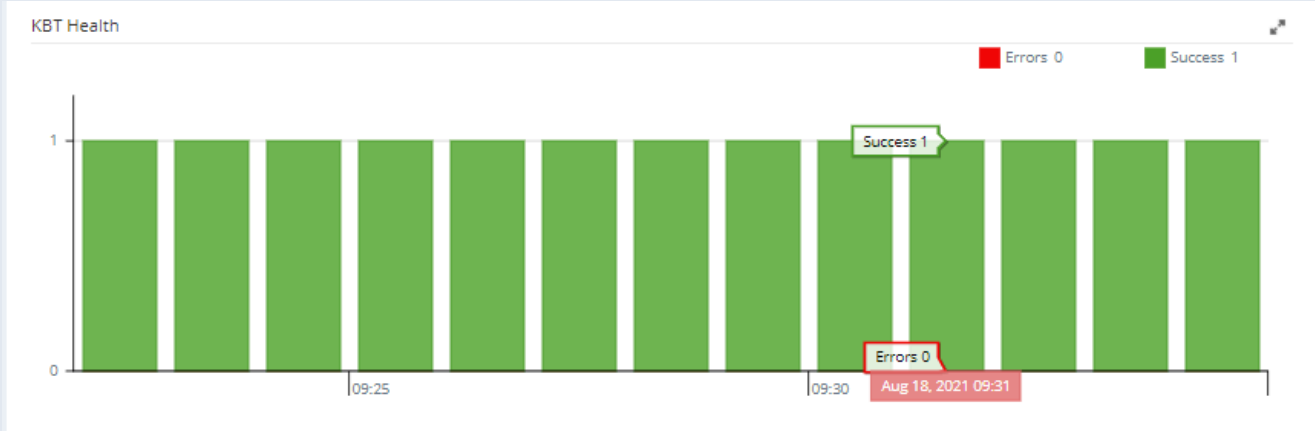
What is the average response time for the app and how does it vary across each participating server?

Select the bar indicating an increase in ART and it will display a list of servers in the selected app along with their respective ARTs, making easy to identify any server/s has/have slowed down.



Which business transactions are being invoked, and what is their performance?

The Business Transactions Health section displays load, success rates and errors and provides the option to drill into detailed information.

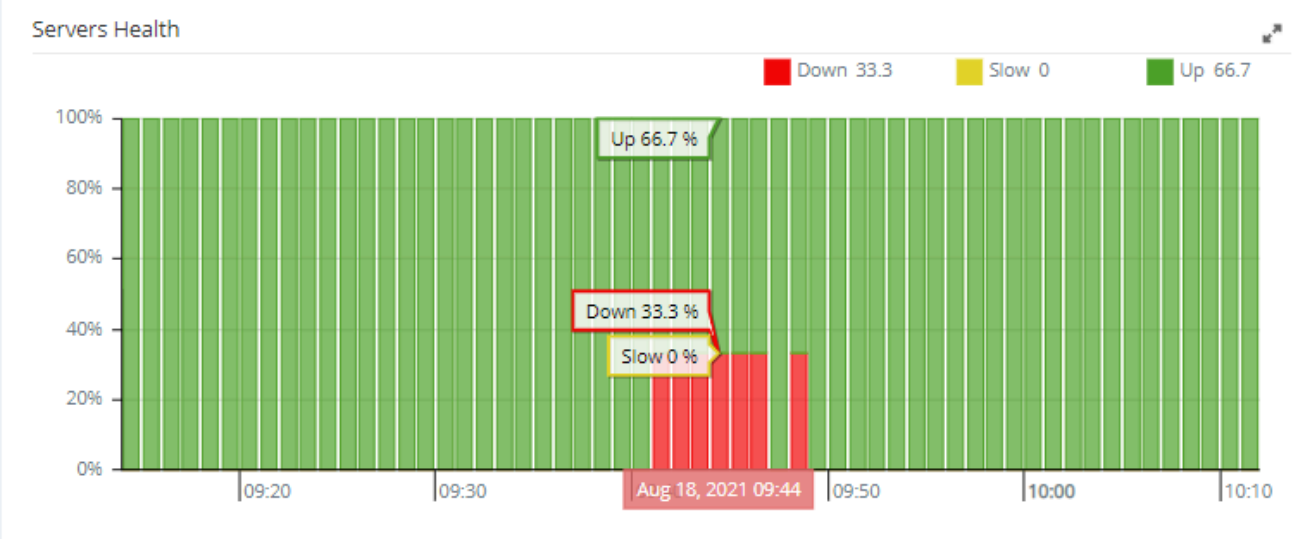


Where are the users located, and how is their experience with app?

Performance by Country/City provides insights into end user's experience with the selected application, including average app server response times, network latency, and browser's page rendering times. It also allows drill down to 'User Experience' tab for detailed information on individual slow transactions.

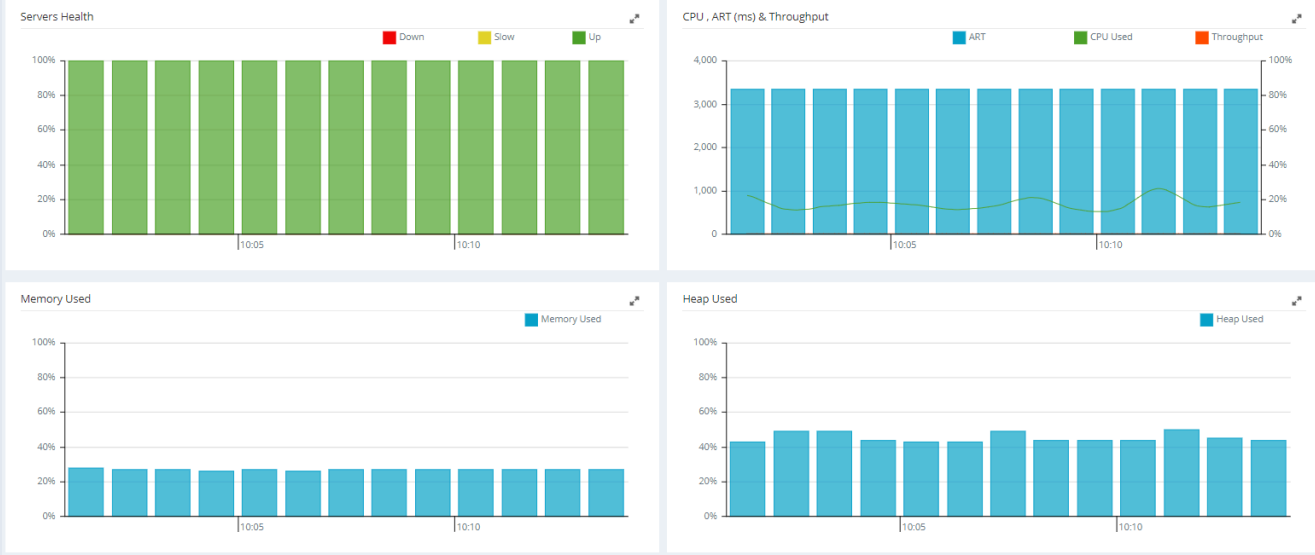


What servers are currently slow or have experienced downtime?



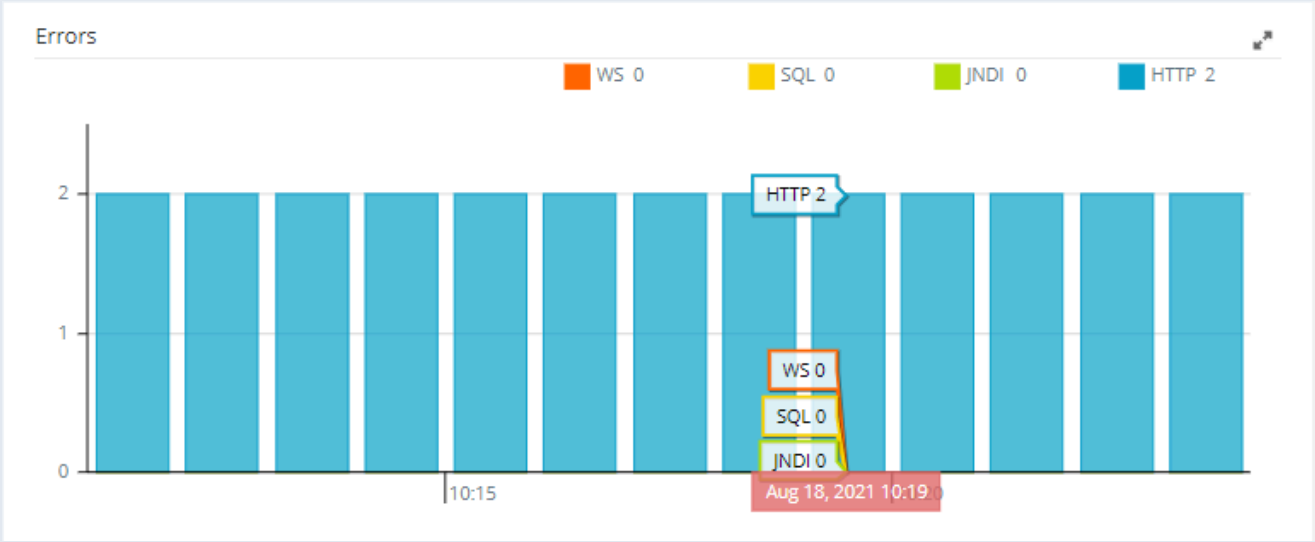
How is my application doing in terms of capacity?

The Enterprise Dashboard offers a clear view of an application's performance with respect to CPU, memory and heap usage. The three charts in the bottom right display cumulative usage by all servers within the selected application. i.e. in the JPetStore application shown below, the three servers each use 6% CPU, 26% memory, and 51% heap.



What is the transaction flow of my app and where are the errors occurring?

In the Enterprise dashboard, the **Network Connections** section displays the connections made by application's member servers. Clicking on server name hyperlink in bottom left table will provide detailed information about these connections, including their endpoints and current state.



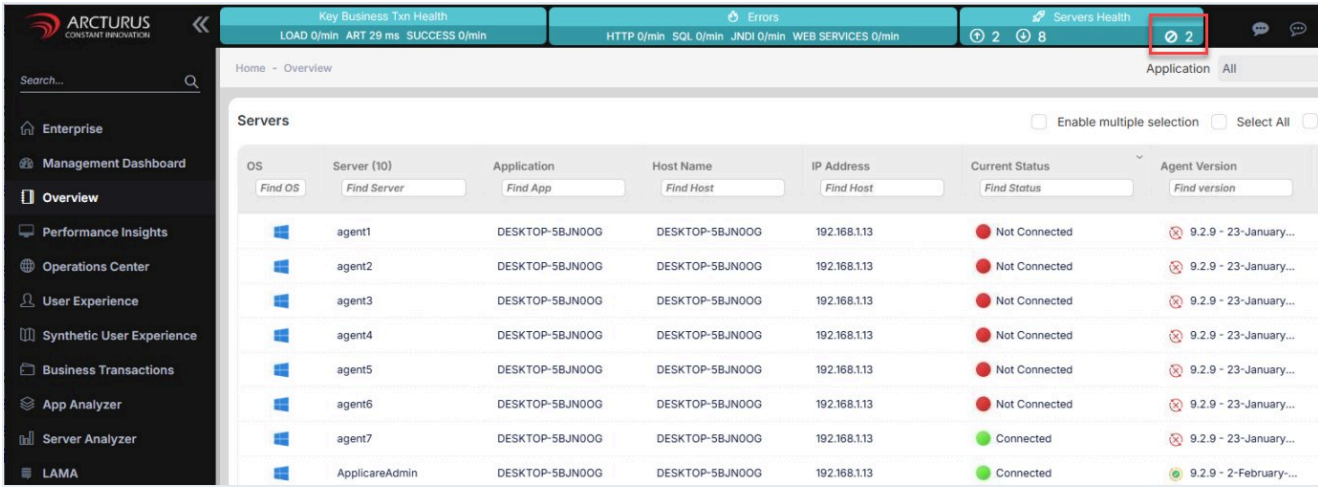
[Monitoring with Applicare]

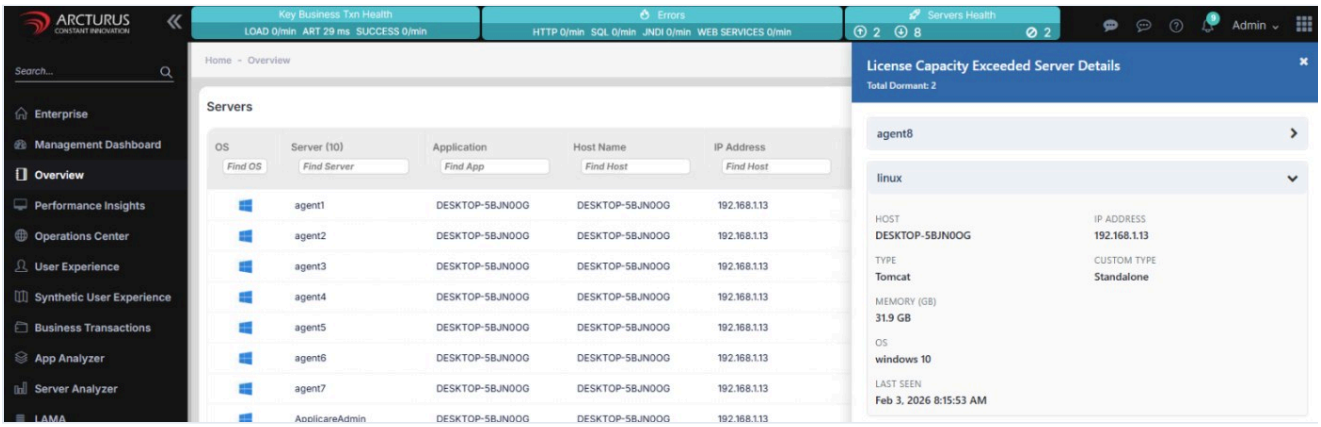
USING APPLICARE › DASHBOARDS

Unattached servers

Applicare 9.2 · Updated February 4, 2026 · **Pending technical review**

This feature displays the details of agents who attempt to add on the Applicare after reaching the allocated agent license limit. This helps administrators identify and monitor additional agent registration attempts when licenses are unavailable.





USING APPLICARE › APPLICATIONS

Log Analyzer

Applicare 9.2 · Updated January 21, 2026 · logs · **Pending technical review**

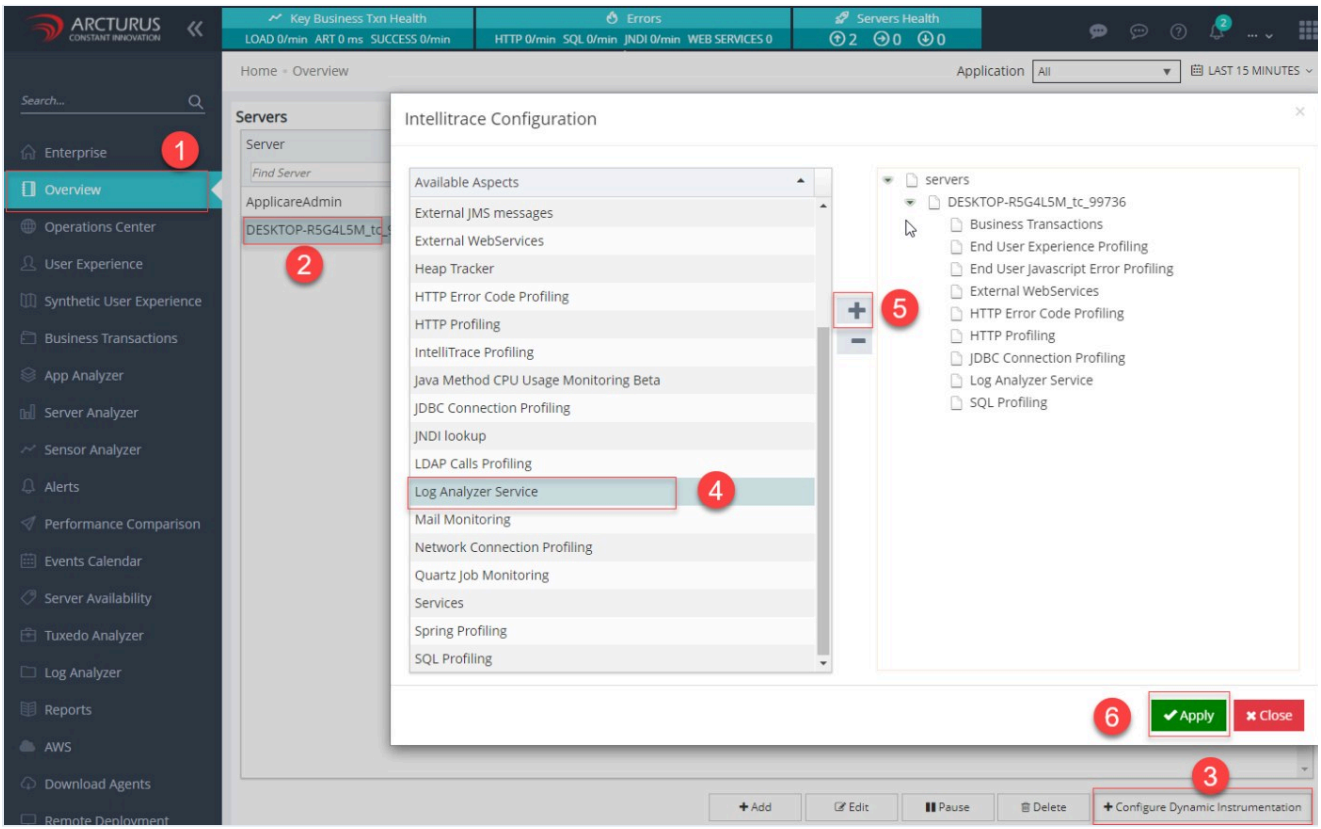
Applicare log monitoring allows you to search and find information spread across log files from applications and servers through a single, intuitive interface. Its seamless integration with IntelliSense and Business Transaction Monitoring allows you to quickly focus on relevant sections of logs files for any transaction with just a few clicks.

Configuration

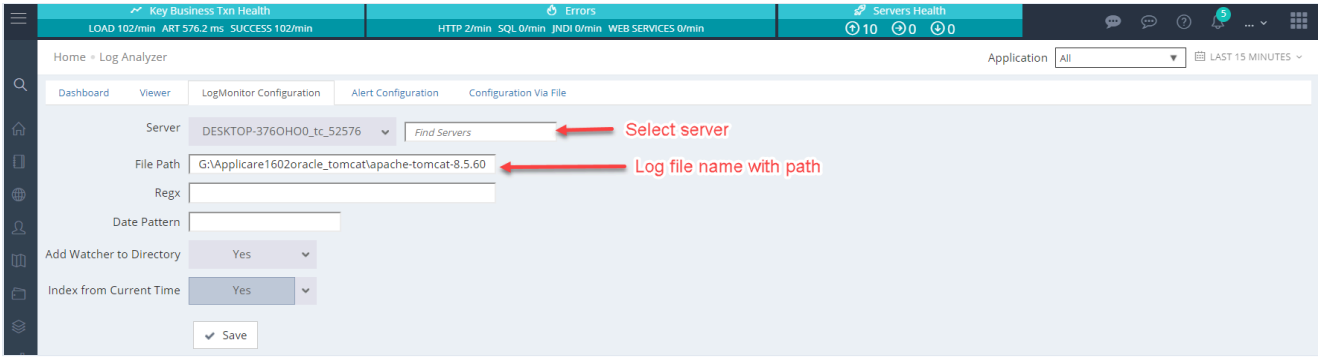
Applicare can monitor any log file accessible by a monitored server / agent. You can provide the absolute path to the log file or a directory containing log files. If the provided path is a directory, all log files found in the directory will be monitored.

By default, the **Log Analyzer Service** aspect is available for all agents. If it is not available, follow these Steps to add that Aspect.

Step 1: Go to **Overview Menu,** select the server, click the **Configure Dynamic Instrumentation** button and add the **Log Analyzer Service** to the server.



Step 2: Navigate to the **LogAnalyzer Menu** -> **LogMonitor Configuration** to add the logs.



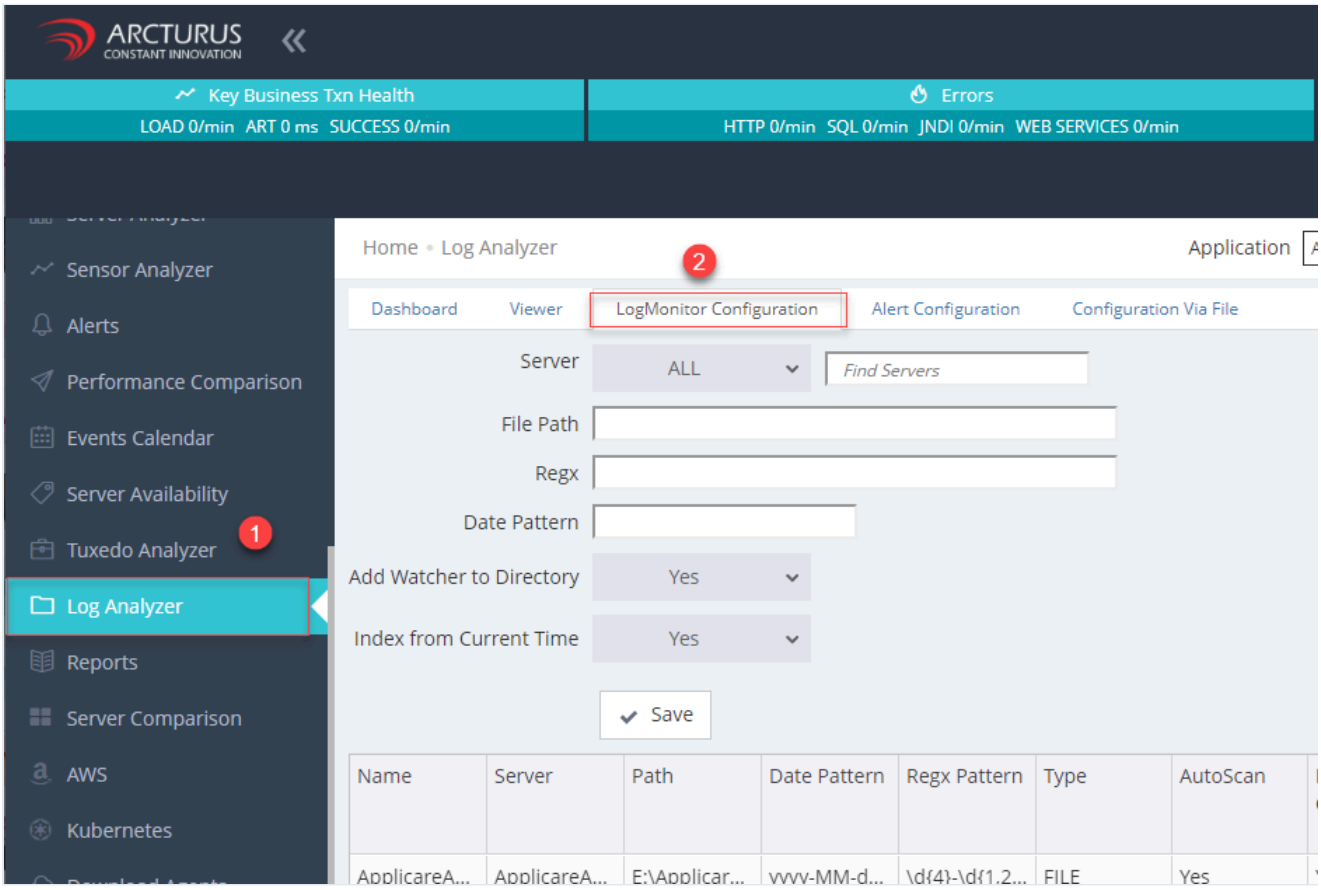
Example: Assume the Applicare Agent is installed in the **D:** drive and you want to add the Applicare Agent log for monitoring.

1. Select the **Applicare Agent name** from the **Server** dropdown.
2. Copy the Applicare Agent log path.
3. Paste the path into the **File Path** field.

Example log path: `D:\ApplicareSingleAgentWindows\logs\IIS.log`

Log Monitor Configuration

1. Go to **Log Analyzer - Log Monitor Configuration** menu.



2. Enter the details and click the **save** button to save the changes.

Server - Select the Server to which we are going to add the logs.

File Path - Either you can provide the directory path or file path.

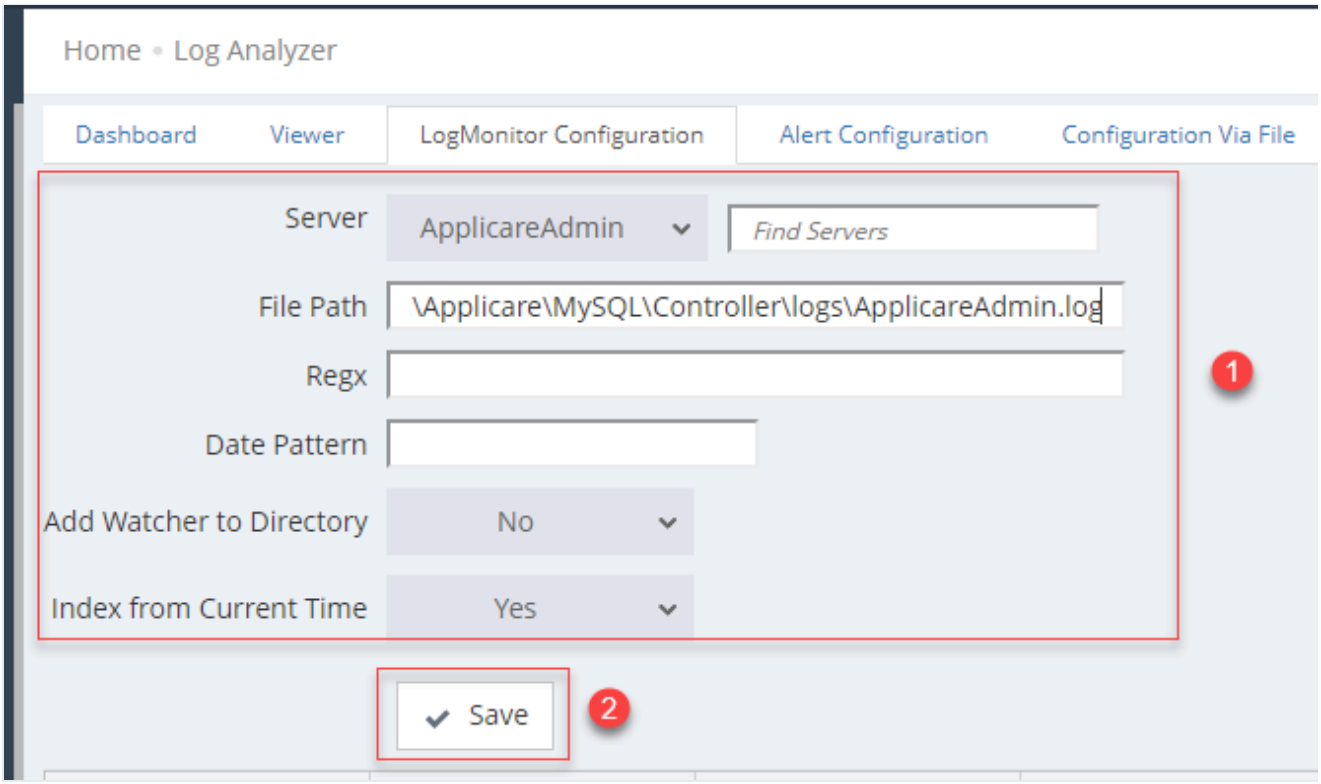
Regex Date Pattern - Leave empty.

Add watcher to Directory - Applicable for directories.

- **Yes** - The system will automatically monitor any new log files created in the directory.
- **No** - The system will not monitor new log files in the directory.

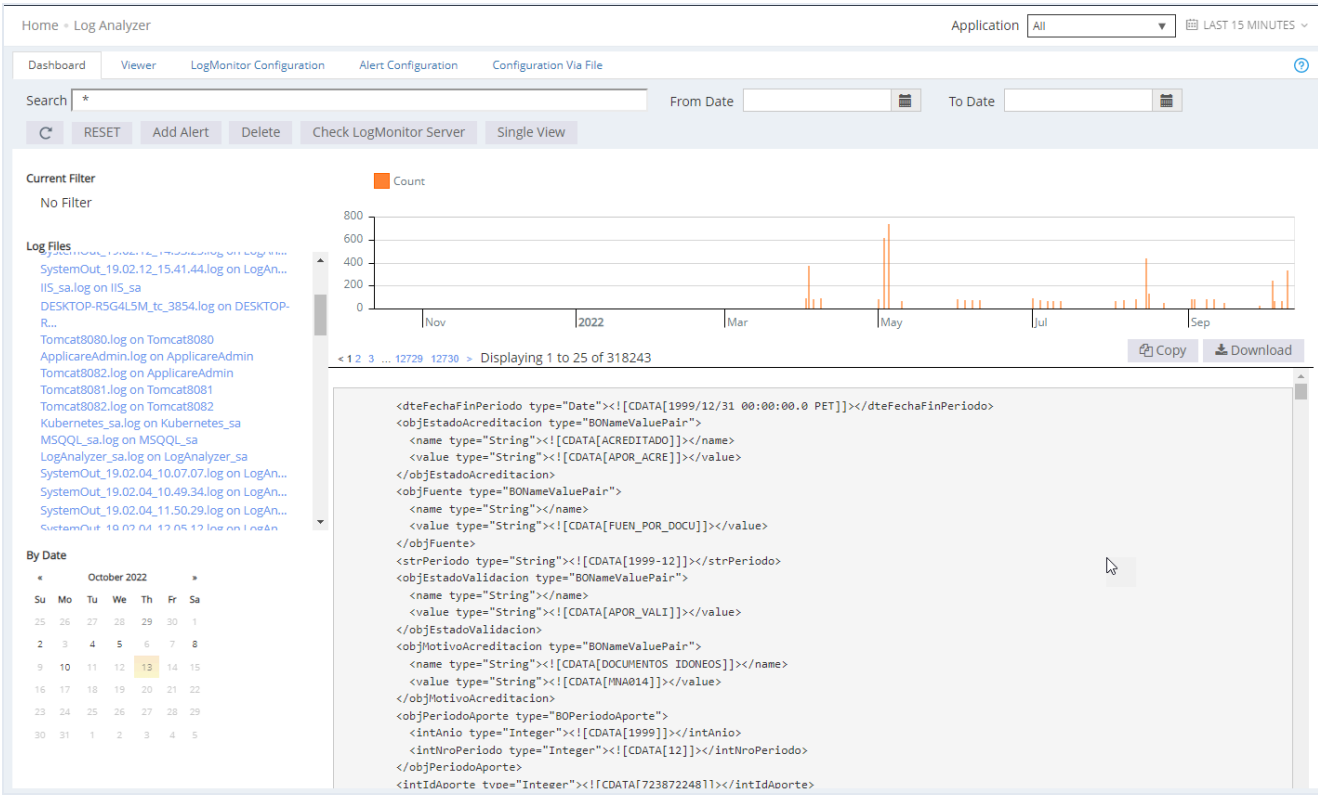
Index From Current Time - Choose whether to start monitoring logs from the current time.

- **Yes** - Monitoring will begin from the current time.
- **No** - Monitoring will include both new and existing records.

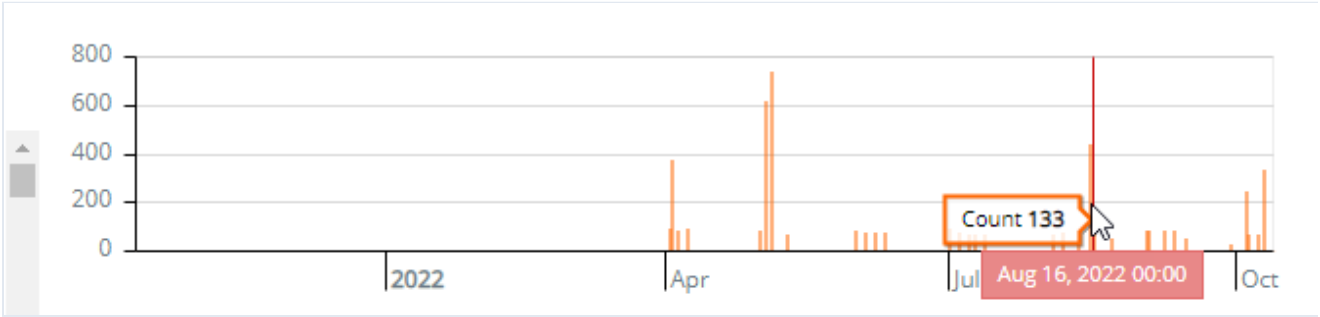


Dashboard

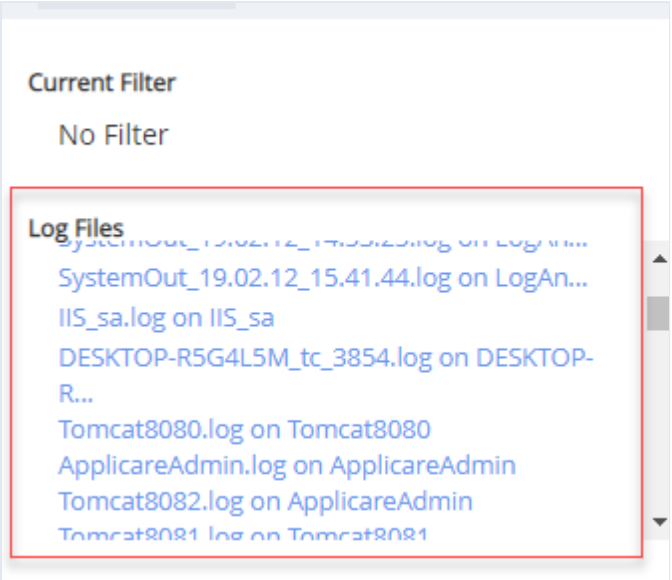
With the Log Analyzer, you can search through all your logs files from a single UI. You can refine search results by specifying a date/time range and further filter by log file or date using the dynamic filters on the left. By default, records are displayed in descending order based on the log date and time.



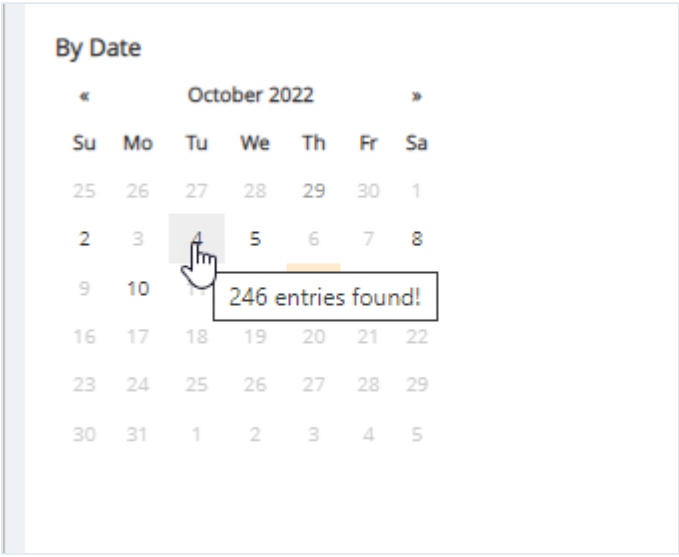
- **Graph** - Displays the count of records per day.



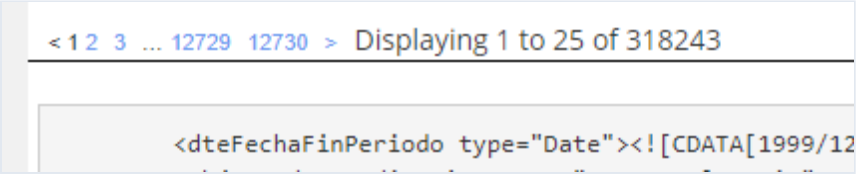
- **Log Files** - It will display the log file path and names based on the search result. By clicking on the Log file name it will filter the records available in that particular log files.



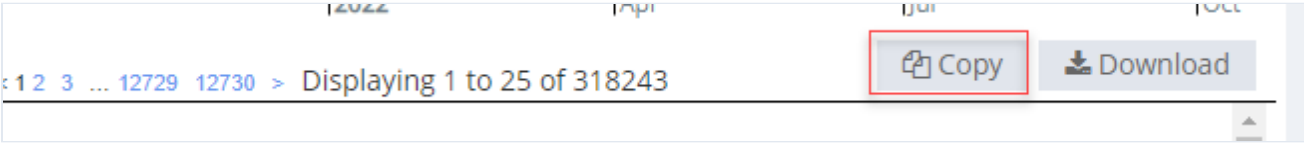
- **Calendar** - The calendar highlights days with log records in bold. Hovering over a highlighted day will display the count of log records for that day.



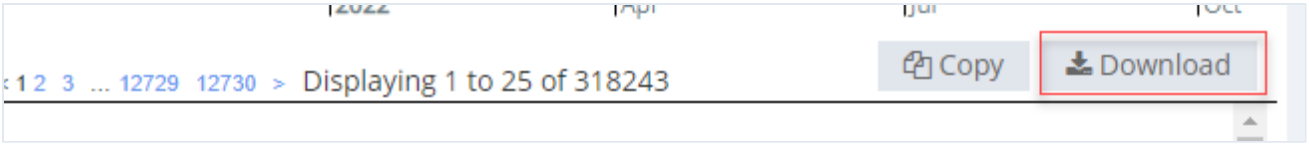
- **Pagination -** By default, 25 records are shown per page. Use the page numbers to navigate to the pages.



- **Copy button** - Copies the records currently displayed on the page.

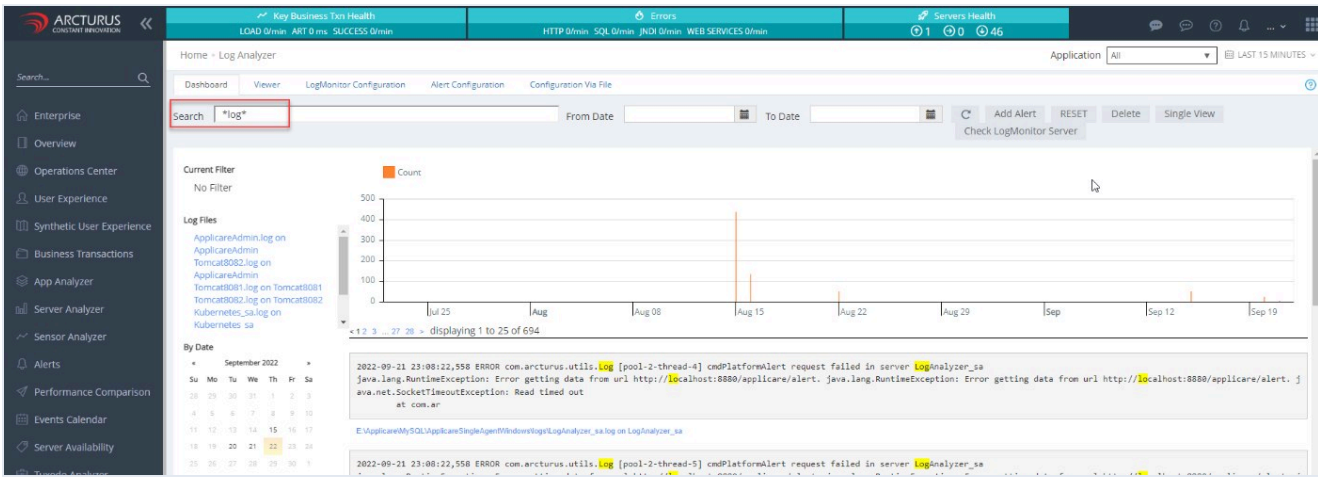


- **Download button** - Downloads the records currently displayed on the page as a text file.

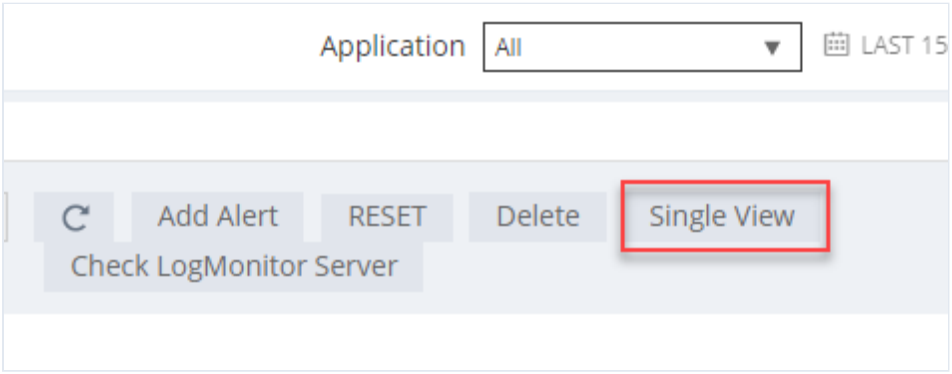


Single View

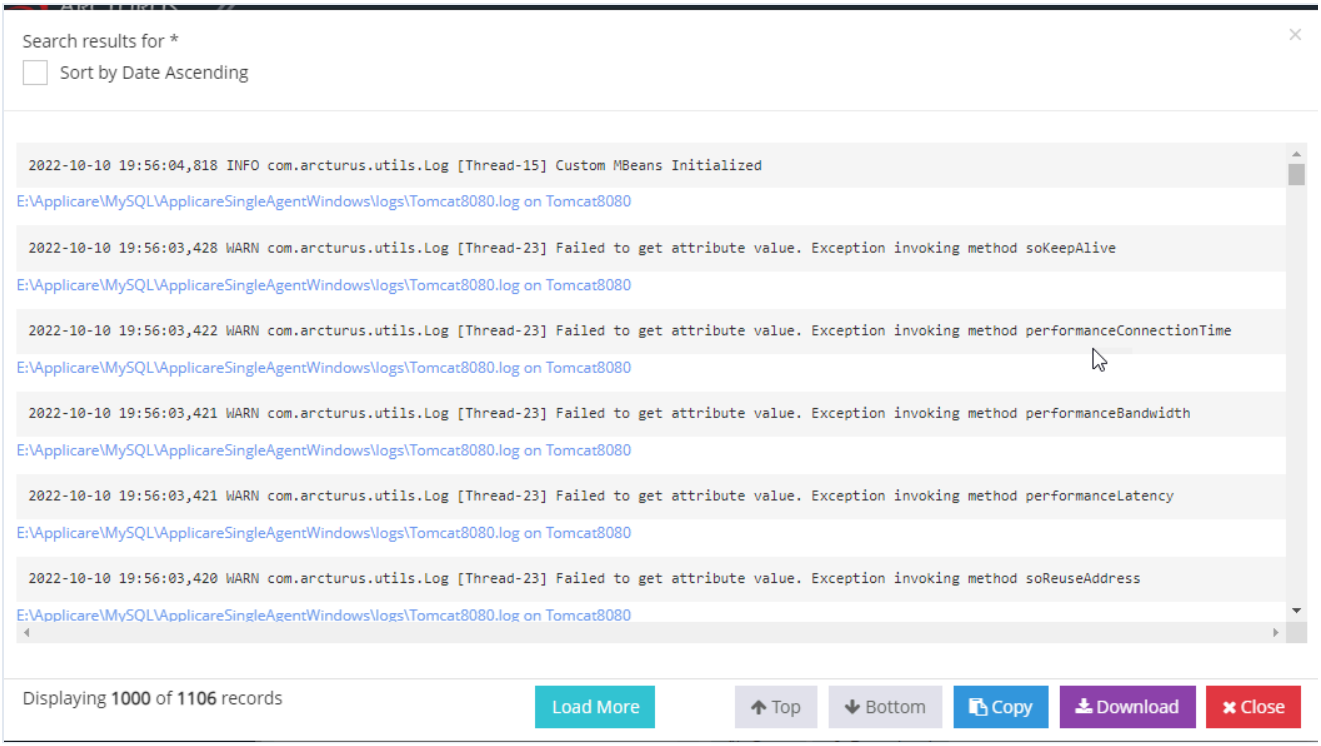
1. Enter your **search query** in the field and press Enter to search the records.



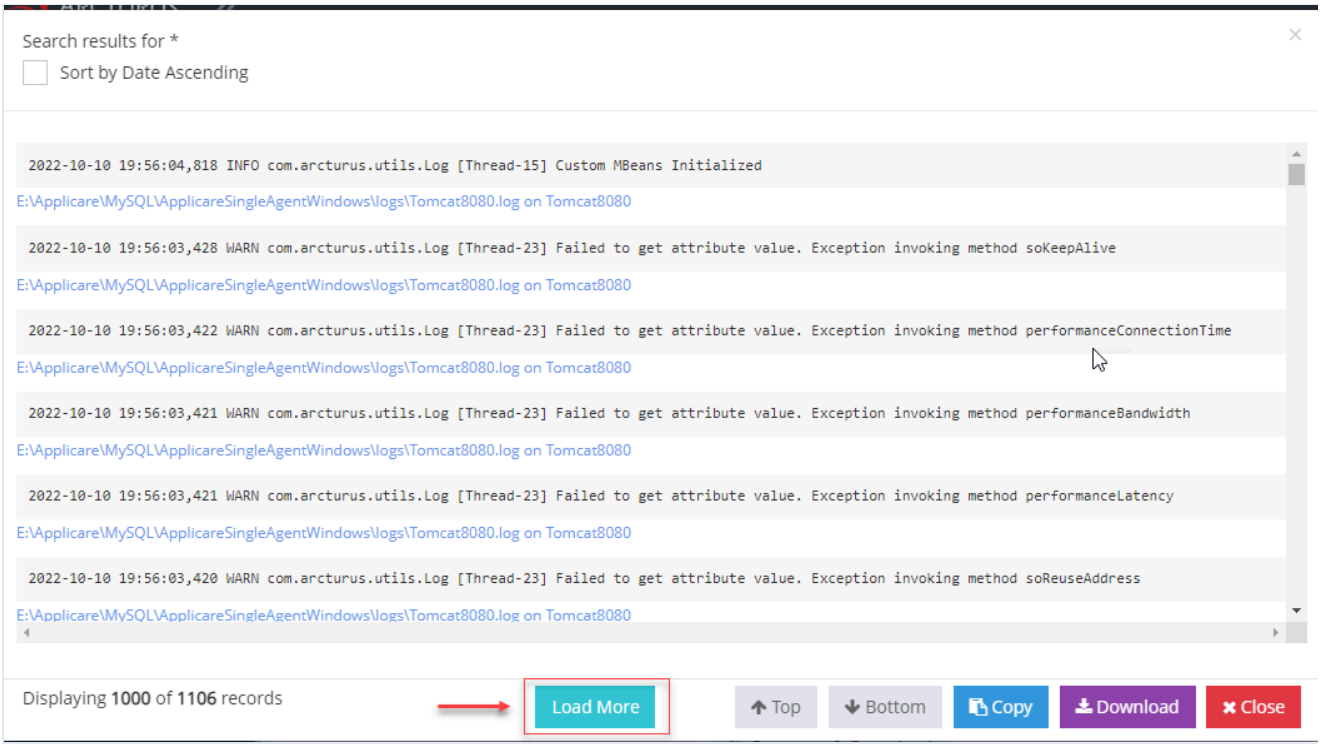
2. Click the **Single View** button.



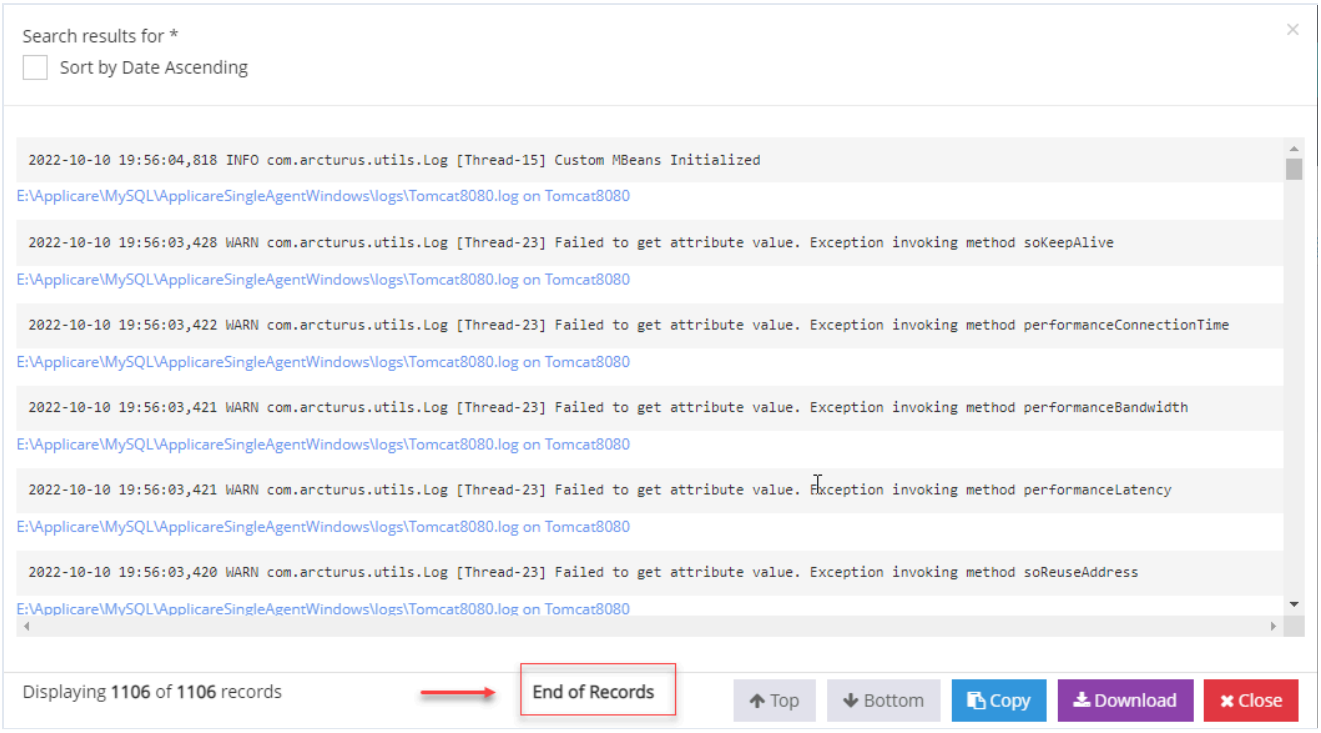
3. A **popup** will appear, Displaying up to **1000** records by default.



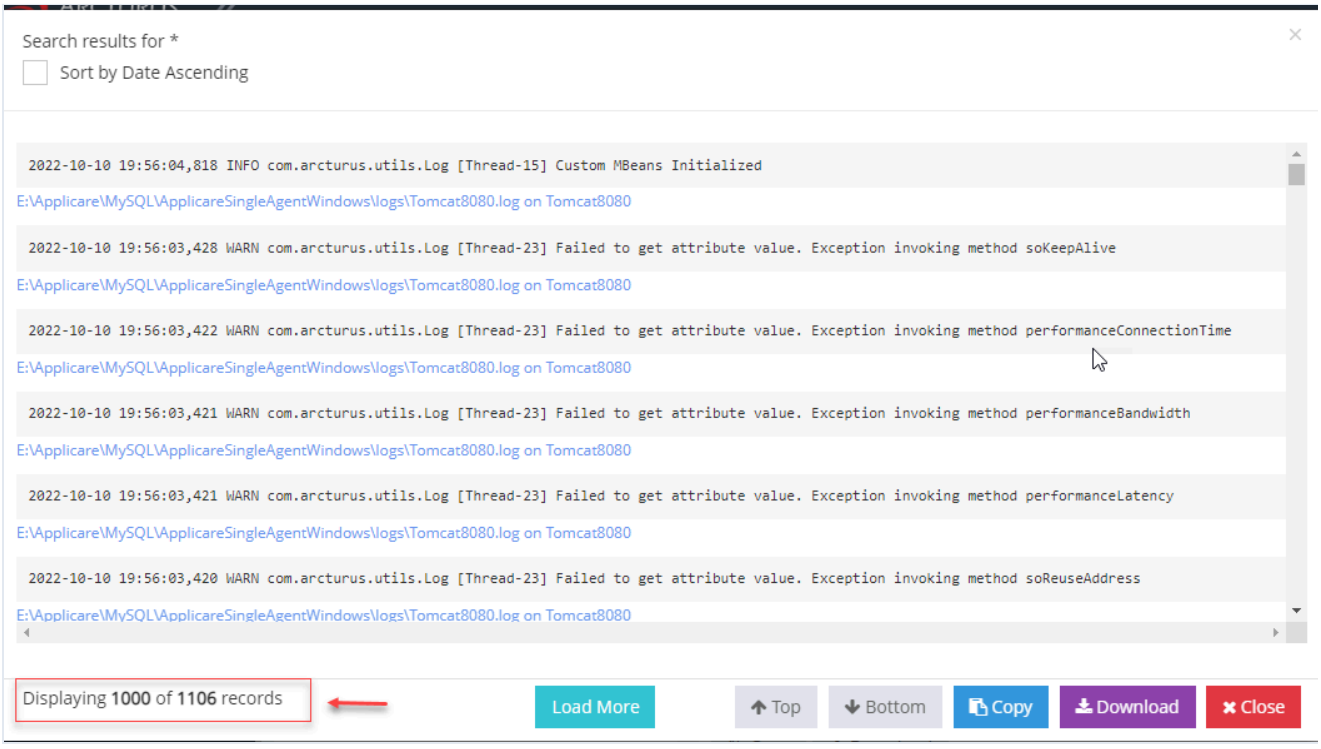
4. To load additional records, click the **Load More** button.



When all records have been loaded, you will see a message indicating **End of records**.

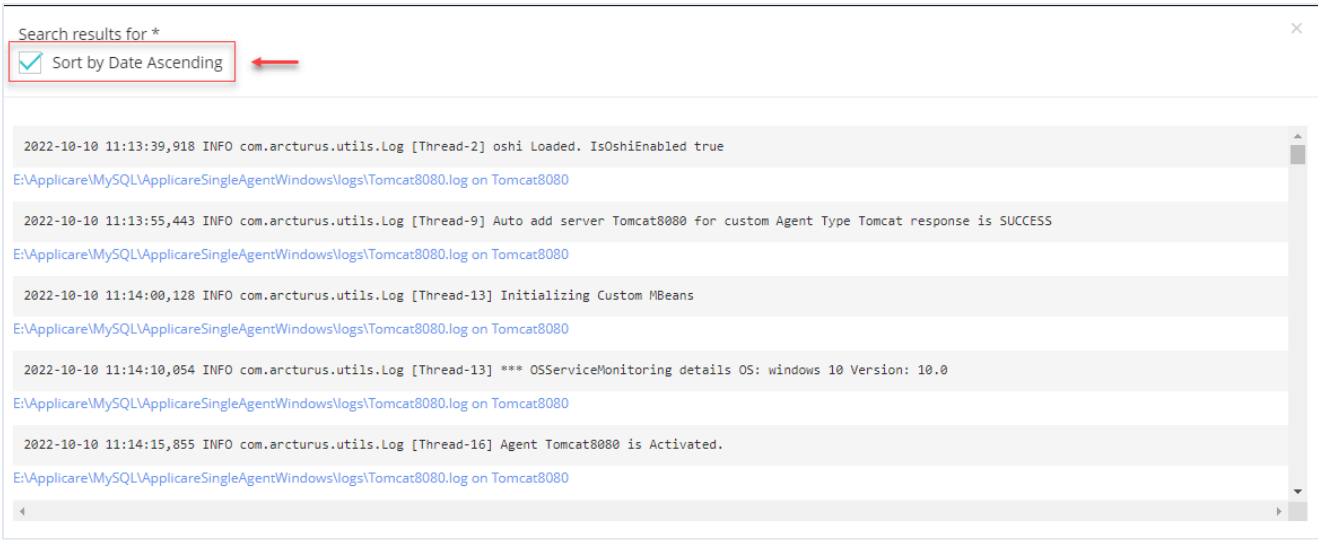


5. The **count** of loaded records will be displayed on the left-hand side.

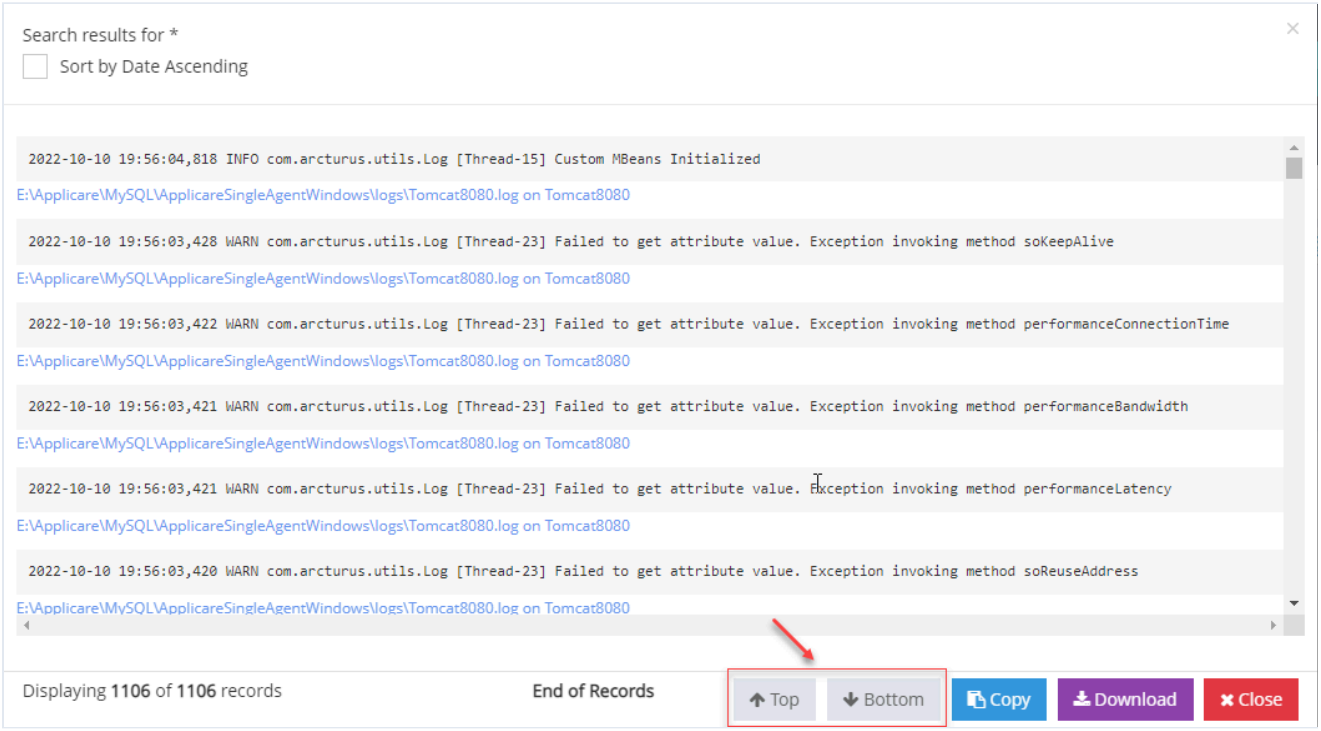


6. By default, records are loaded in descending order based on log date and time.

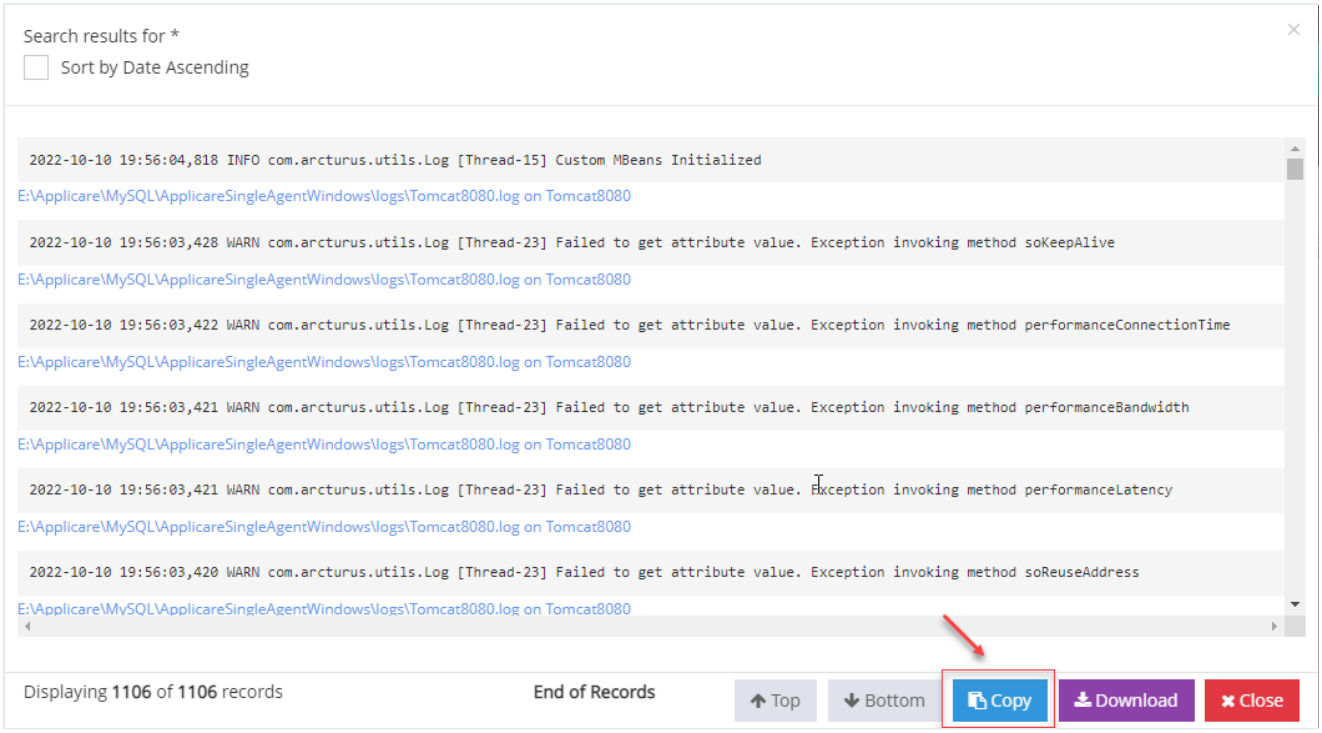
- To sort the records in ascending order, select the **Date ASC** check box.
- To sort the records in descending order, clear the **Date ASC** check box.



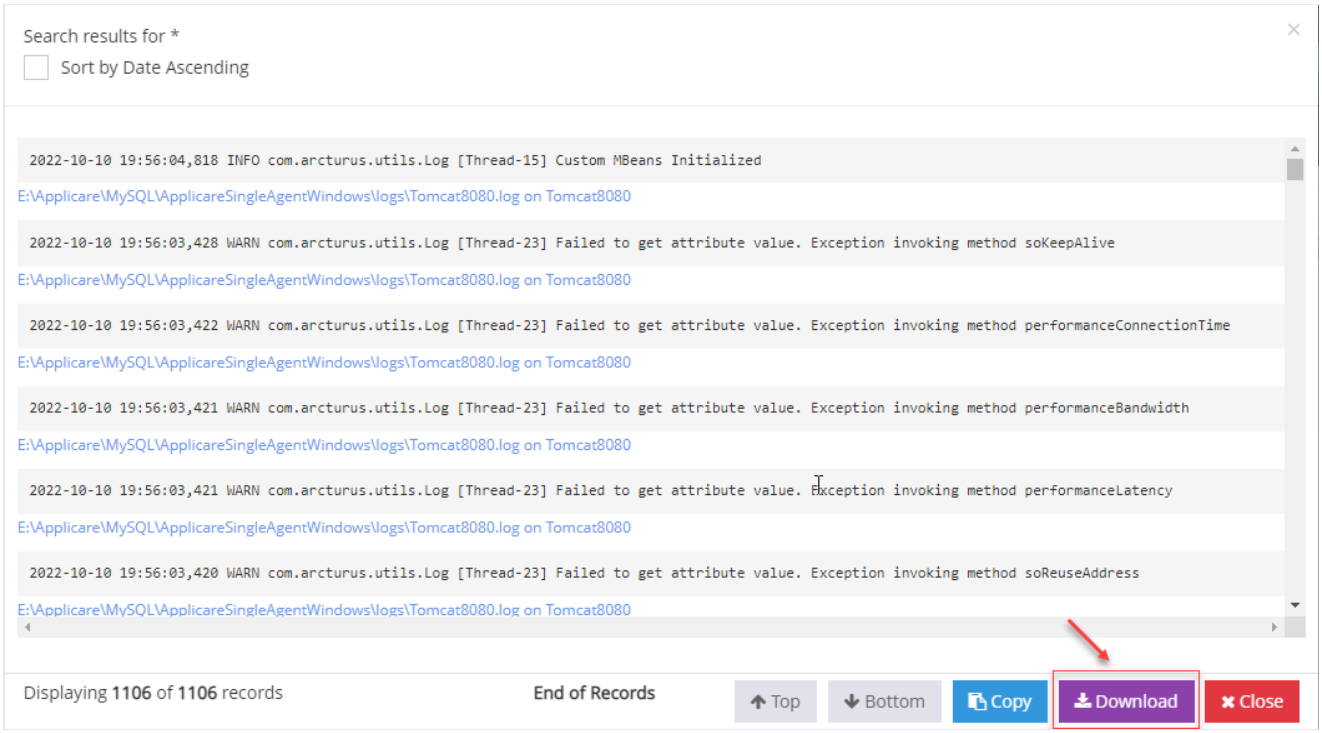
7. Click the **Top** and **Bottom** buttons to scroll to the top and bottom of the records. Clicking **Bottom** will not only scroll to the end but also load the next set of 1000 records.



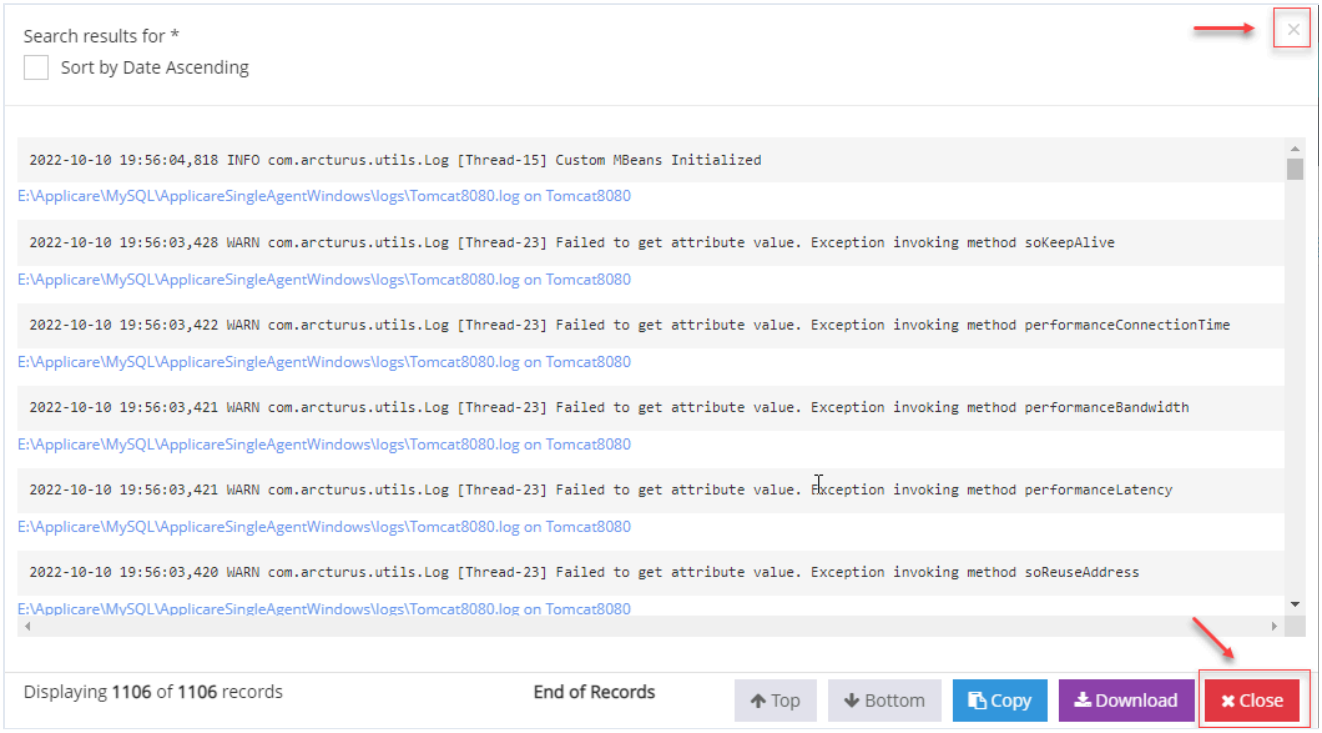
8. Click the **Copy** button to duplicate the currently loaded records.



9. Click the **Download** button to save the loaded records as a text file.



10. **Close Popup** - To close the popup, click either the **close button** at the bottom or the **close icon** at the top of the popup.



Log Analyzer configuration import via csv file

Date and Regx pattern format

Monitoring with Applicare

USING APPLICARE › APPLICATIONS

Alerts

Applicare 9.2 · Updated August 29, 2024 · alerts · **Pending technical review**

User Role and Alerts

Super Admin, Users with Admin rights or App Admin rights for an application can add/view alerts related to that application. User with read only access for an application can only view predefined alerts on the application.

Applicare automatically generate platform alerts for CPU, disk and memory utilization for the agents if it is configured in MySQL or Oracle database.

Configure Alerts

Applicare allows users to set alerts and receive email notifications based on custom conditions. It offers the following types of alerts:

Platform Alerts	**- Operating System and JVM metrics (CPU, Memory, Swap, Heap Usage)**
JMX Alerts	- Set up alerts based on MBean attributes
SNMP Alerts	- Create alerts for values on SNMP capable devices
Tuxedo Queue Alerts	- Set up alerts for Tuxedo Queue
Tuxedo Service Alerts	- Create alerts on Tuxedo Services
UEM Slow Transactions Alerts	- Create User Experience alerts for slow transactions
Slow Transactions	- Create alerts on slow transactions as detected by IntelliSense
Business Transactions - SLA Avg	- Create alerts on Business Transactions exceeding SLA Avg
Business Transactions - SLA Max	- Create alerts on Business Transactions exceeding SLA Max

Tools -> Manage Alerts enable you to create, edit, pause or activate alerts. It also allows you to assign a command to be executed when an alert is fired. Applicare allows any shell/command script execution on any location where Applicare agent is running. This is very valuable for doing diagnosis or taking remediation actions when a certain alert is generated.

For creating high CPU utilization alert select New and platform alert

> [Screenshot 1 could not be downloaded during import — see migration report]

Click on the browser icon next to the property and select CPU -> Usage.

The Alert Trigger Threshold * Alert Monitor Job Interval defines the time interval that condition should hold before generating the alerts. Alert Monitor Job Interval by default is 1 minute but can be adjusted by adding **applicare.alertMonitorJobInterval** (in minutes) property to Applicare.props file. In the example below alert will be generated if the CPU usage is higher than 90 for 3 consecutive polls by Applicare. Similarly in the example below auto clear threshold defines that CPU usage needs to be below 90 for 3 consecutive polls for generating clear alert.

> [Screenshot 2 could not be downloaded during import — see migration report]

Intellicheck Alerts Configuration

IntelliCheck alerts are paused for **intellicheckAlertsPauseTime** (in minutes) after # of alert reaches the **intellicheckAlertPauseThreshold** limit set in Applicare.properties. With the example values below 3 IntelliCheck alerts would come for JDBC resource not available and then alerts on the same JDBC resource would be paused for 30 minutes.

```
intellicheckAlertsPauseTime=30

intellicheckAlertPauseThreshold=3
```

By default, **intellicheckAlertsPauseTime** is set to 15 minutes alert and **intellicheckAlertPauseThreshold** is set to 5 alerts.

Email Notifications

To start the receiving email notifications for new alerts, you first need to configure the SMTP server details.

Navigate to **Applicare** > **Configuration** > **Configure Email/SMTP Configuration** from the top menu.

> [Screenshot 3 could not be downloaded during import — see migration report]

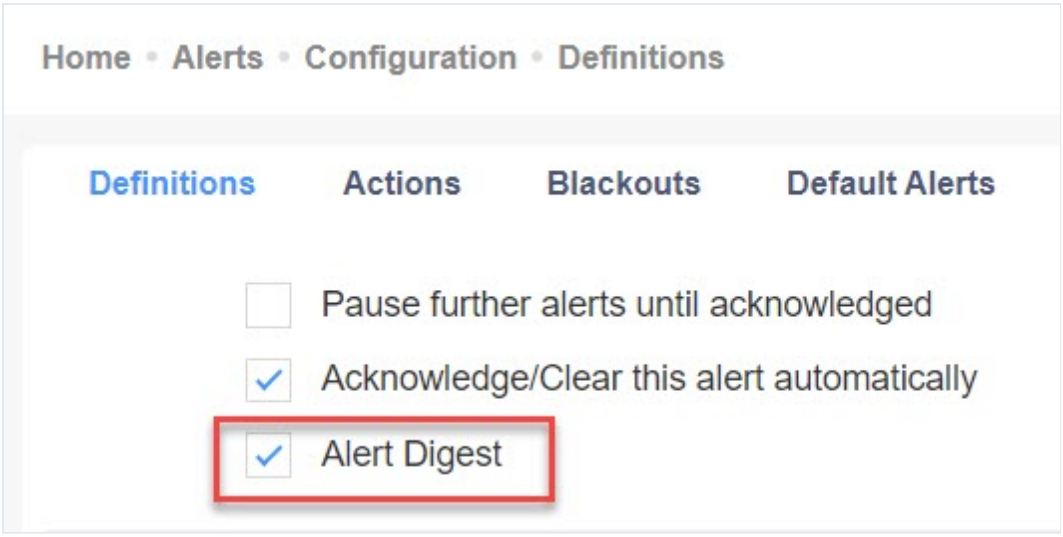
Mail To is the catch all email address. All Applicare alerts are sent to this email address.

To customize who gets what alerts select **Home** > **Alerts** (tab)> **Configuration** (wheel icon). Then select **Actions**.

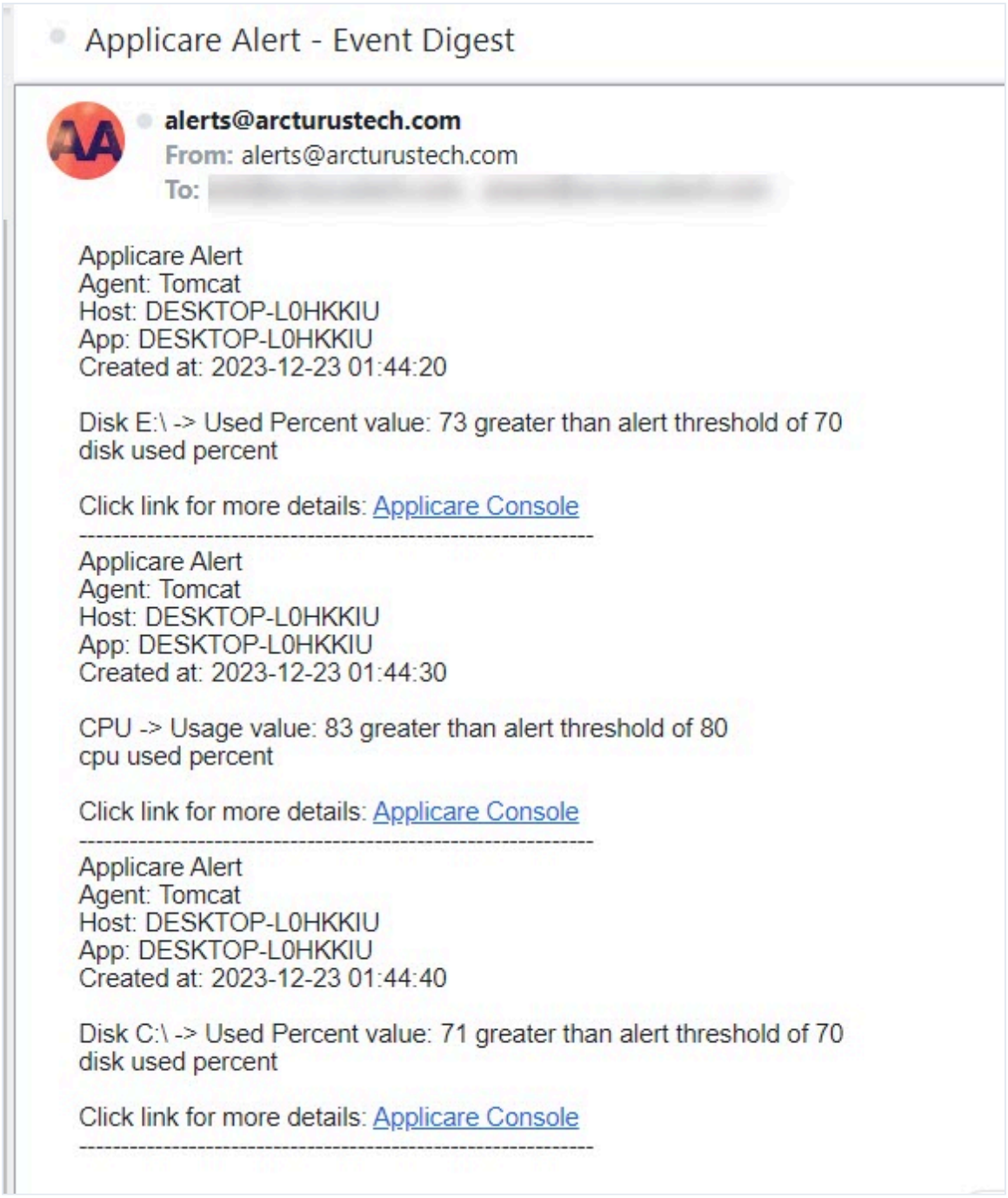
Under actions you can select users, add email addresses or email list addresses for receiving alerts on an application, a server or an individual alert. Applicare would send alert to emails/users defined for individual alert. If there are no specific emails/users defined for individual alerts, it would inherit the same from the server or the application it is part of e.g. If **portalLX** application alerts are sent to Larry and no other specifics are defined for servers that are a part of **portalLX** or individual alerts then all the alert emails for portalLX would be sent to Larry. Once a specific email address/s and users are defined for either server alerts or individual alerts then that overrides the application alert settings.

If you update the emails/users settings are changed for an application after defining setting for individual alerts or server alerts, they would be overwritten by the newly defined settings for the application.

Alert Digest



Applicare includes an alert digest feature that consolidates multiple alerts into a single email. This approach simplifies the management and readability of alerts, providing users with a more organized and user-friendly experience.



Advanced Options

From the configuration screen, you can enable or disable alerting.

Select **Applicare** > **Configuration** > **Advanced Options** from the top menu.

> [Screenshot 6 could not be downloaded during import — see migration report]

Using A Custom Alert Processing Class

You can create a Java class to which will be invoked for each alert created by Applicare agents by implementing the interface **com.arcturus.alerts.notifications.AlertAction** and the only method in the interface public void **processAlert(AlertQueue alert)**. You can get the details of the alert using the getter methods of AlertQueue object as shown in the example class below.

```
TEXT
package com.arcturus.alerts.notifications;

import java.text.SimpleDateFormat;

import com.arcturus.alerts.domain.AlertQueue;

public class SampleAction implements com.arcturus.alerts.notifications.AlertAction{

    public void processAlert(AlertQueue alert) {
        System.out.println("Server: " + alert.getServer()); //the server name where the alert
was generated
        System.out.println("Type: " + alert.getSubsystem()); //type of the alert (JMX_ALERT,
PLATFORM_ALERT, etc..)
        System.out.println("Priority: " + alert.getPriority()); //priority of the alert
        System.out.println("Message: " + alert.getMessage()); //user defined message in alert
definition (description)
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        System.out.println("Created: " + dateFormat.format(alert.getCreated())); //time the
alert was created
    }
}
```

To deploy the class and any dependent classes, you can package it in a jar file and copy to **Applicare/server/ext** folder (create it if it doesn't exist) and restart Applicare server.

When compiling make sure to include the **Applicare/server/webapps/applicare/WEB-INF/classes** folder in your classpath.

After restarting the server, navigate to **Applicare** > **Configuration** > **Advanced Options.** Enter the fully qualified class name of your custom class in the "Custom Alert Processing Class:" field and click **Apply**.

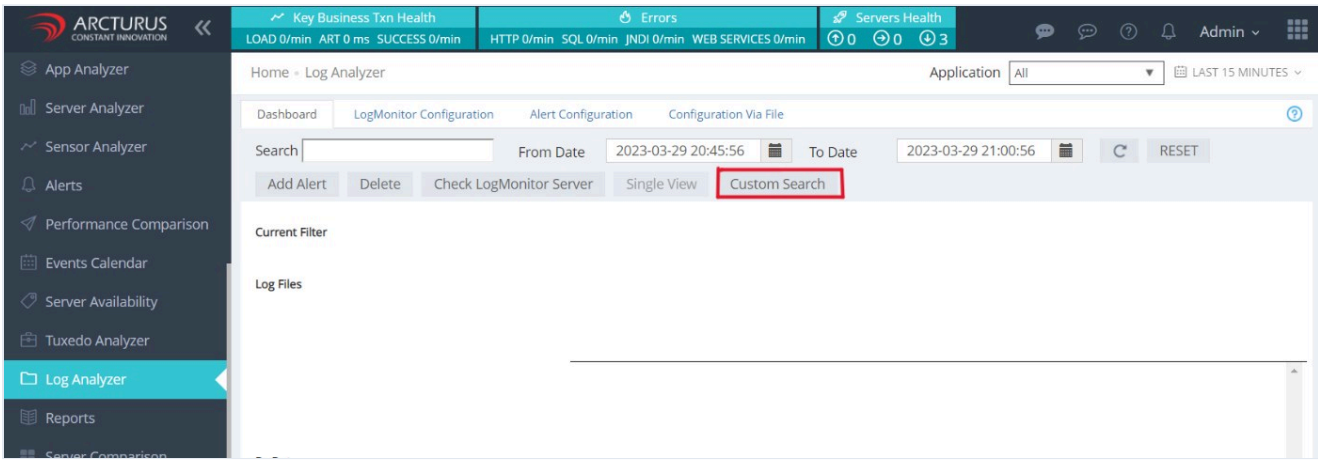
> [Screenshot 7 could not be downloaded during import — see migration report]

USING APPLICARE › APPLICATIONS

Custom Search in LogAnalyzer

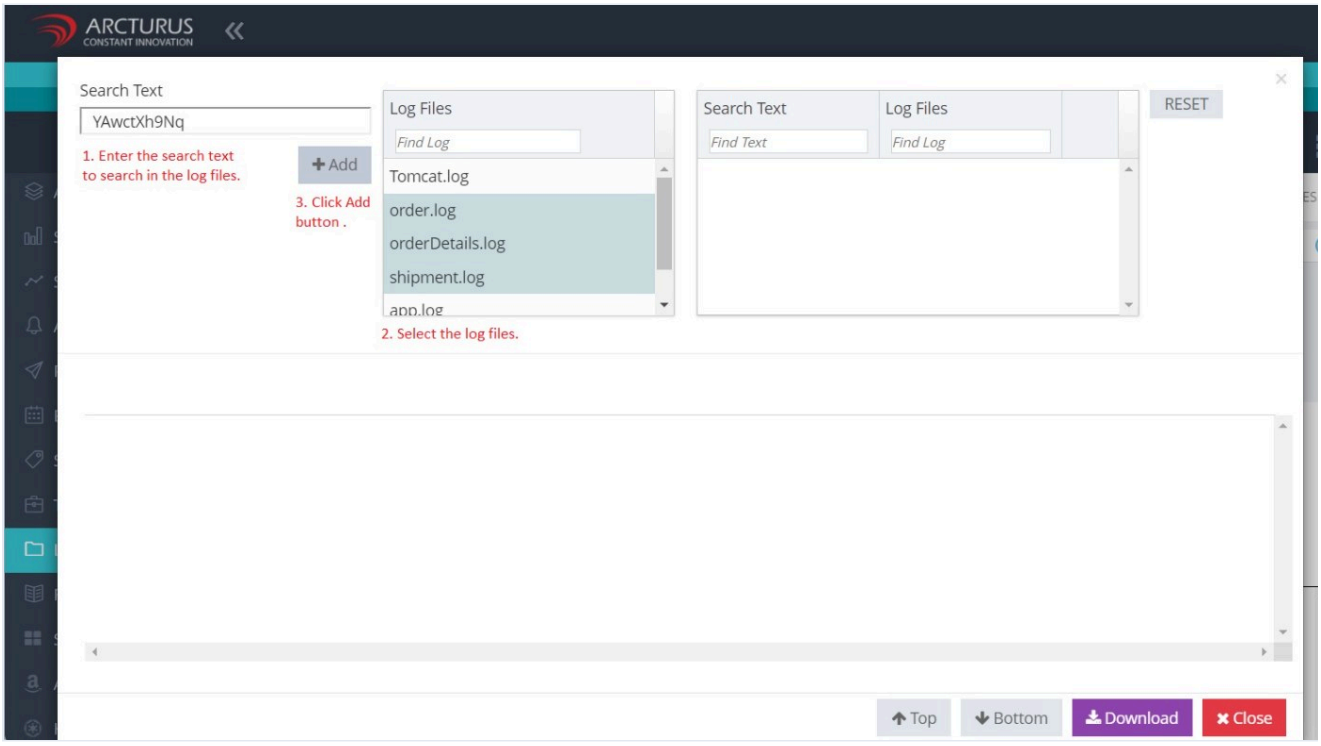
Applicare 9.2 · Updated September 13, 2024 · logs · **Pending technical review**

1. Go to **Log Analyzer** menu and click the **Custom Search** button.

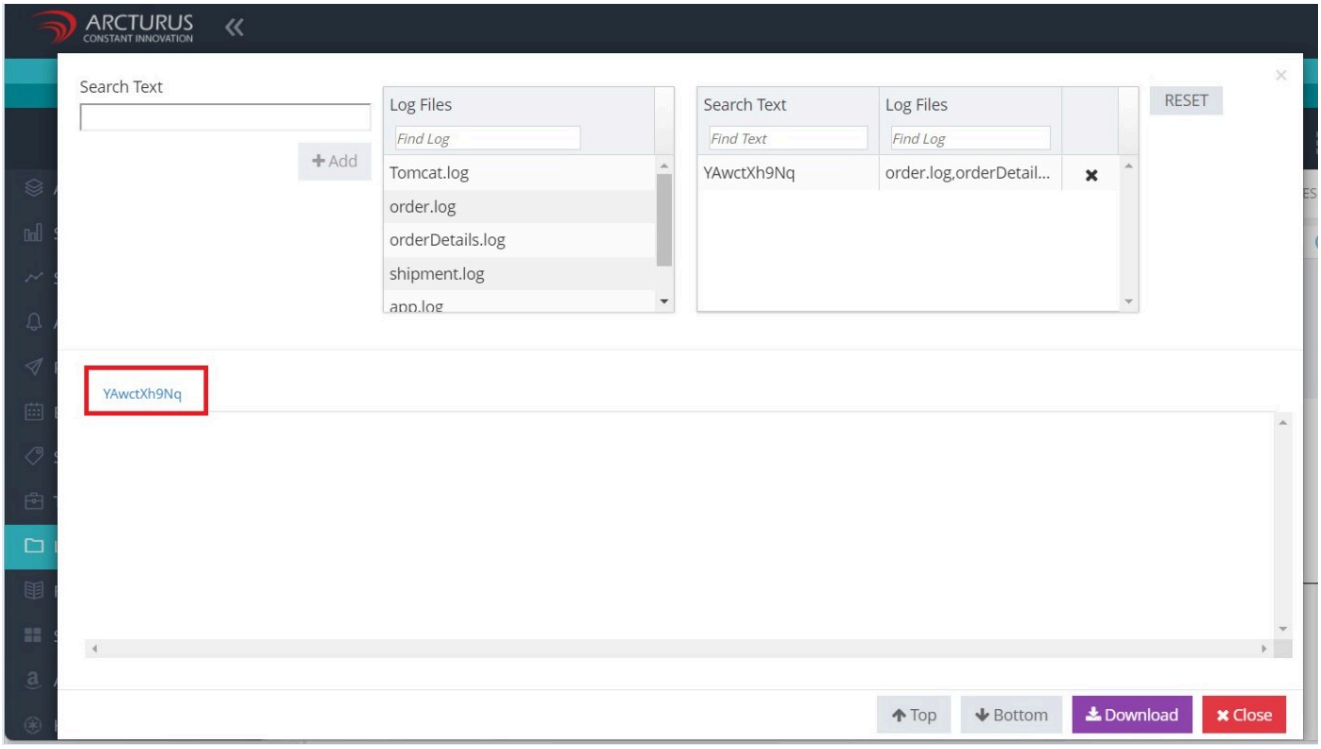


2. In the Custom Search popup

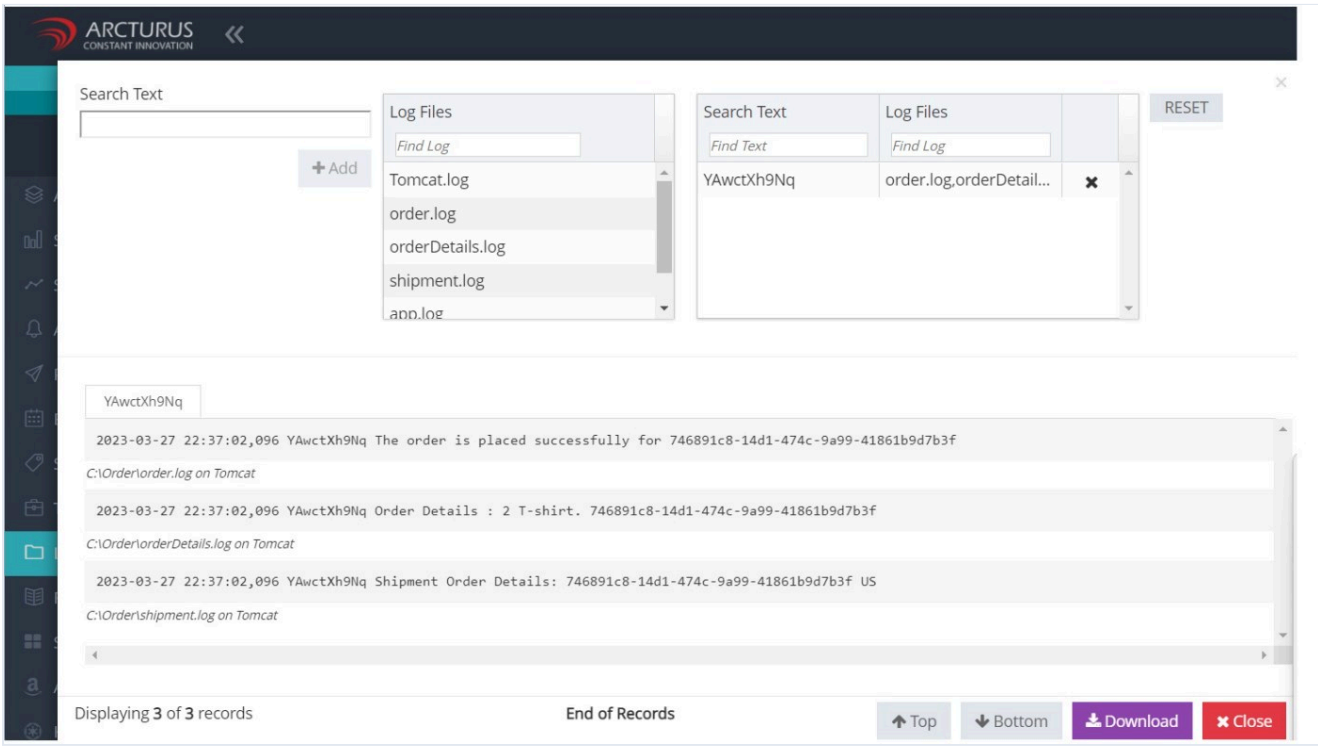
1. Enter the search text.
2. Select the log files.
3. Click the Add button.



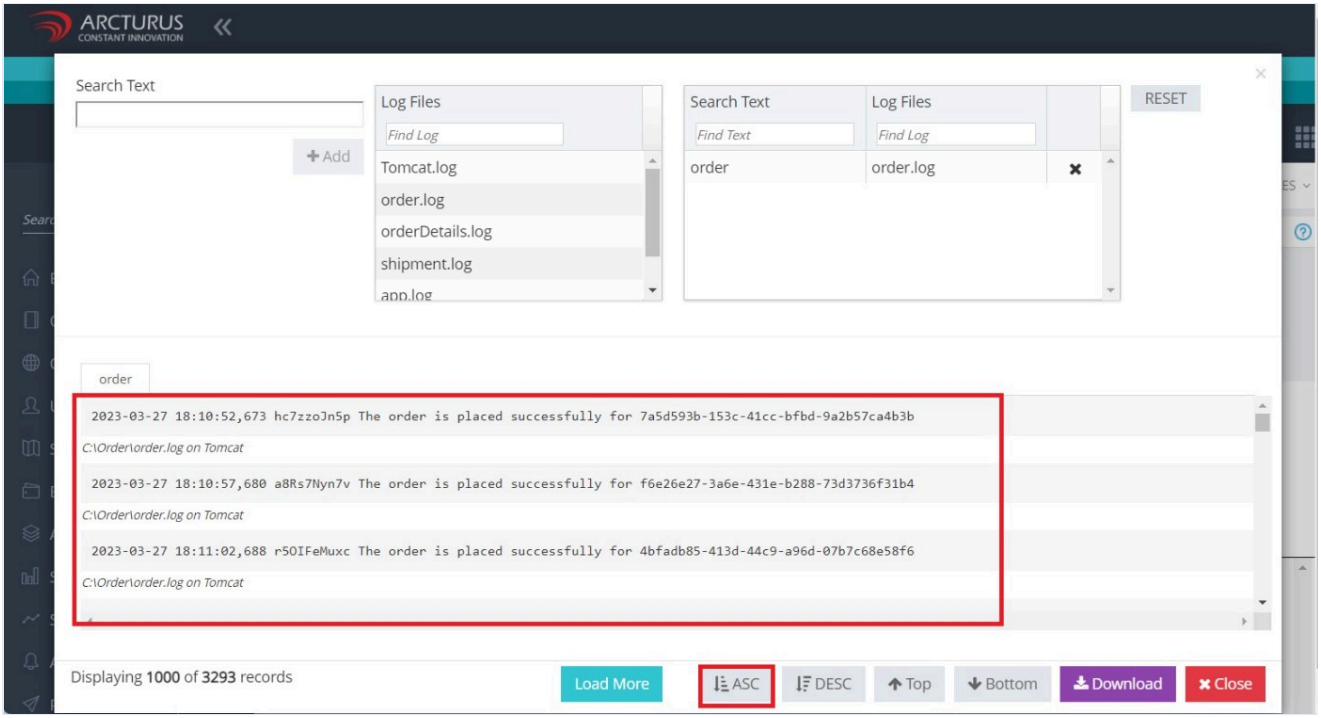
3. A tab will appear with the name of the search text. Click on the tab to view the results.



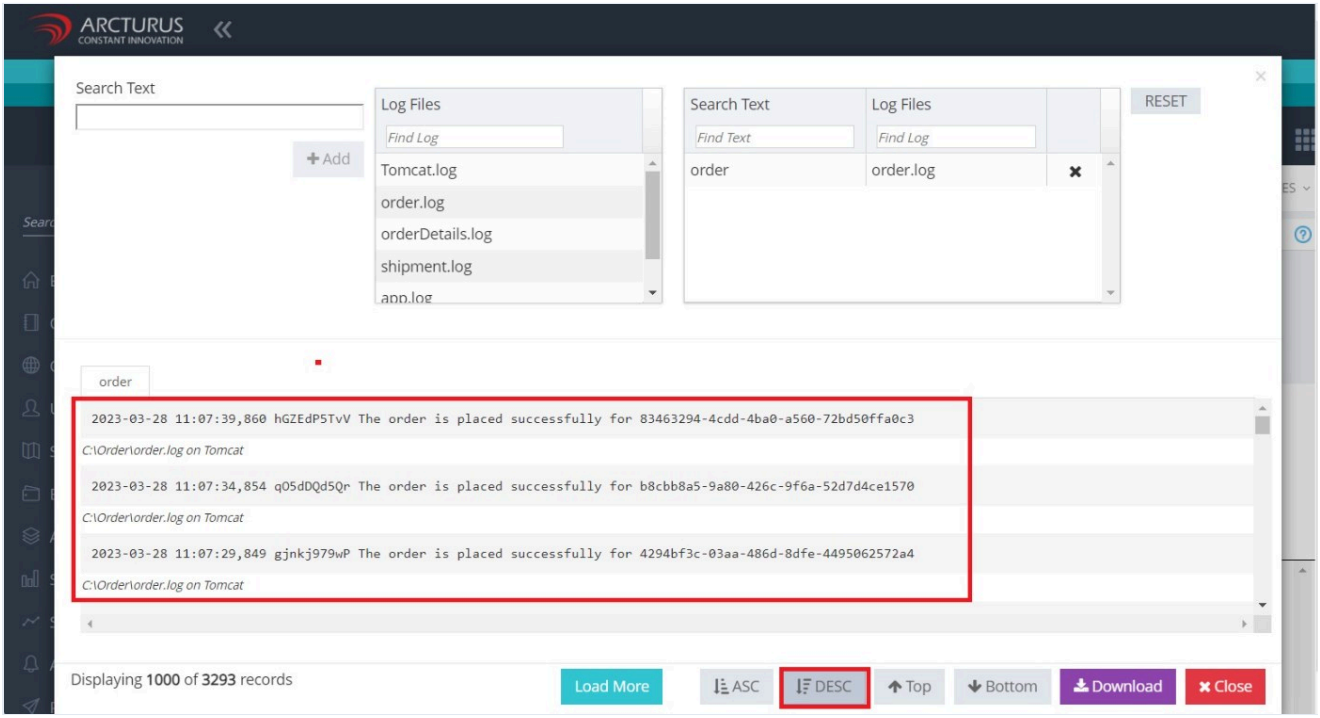
By default, it will display the **top 1000 records** in **descending order** based on the log time.



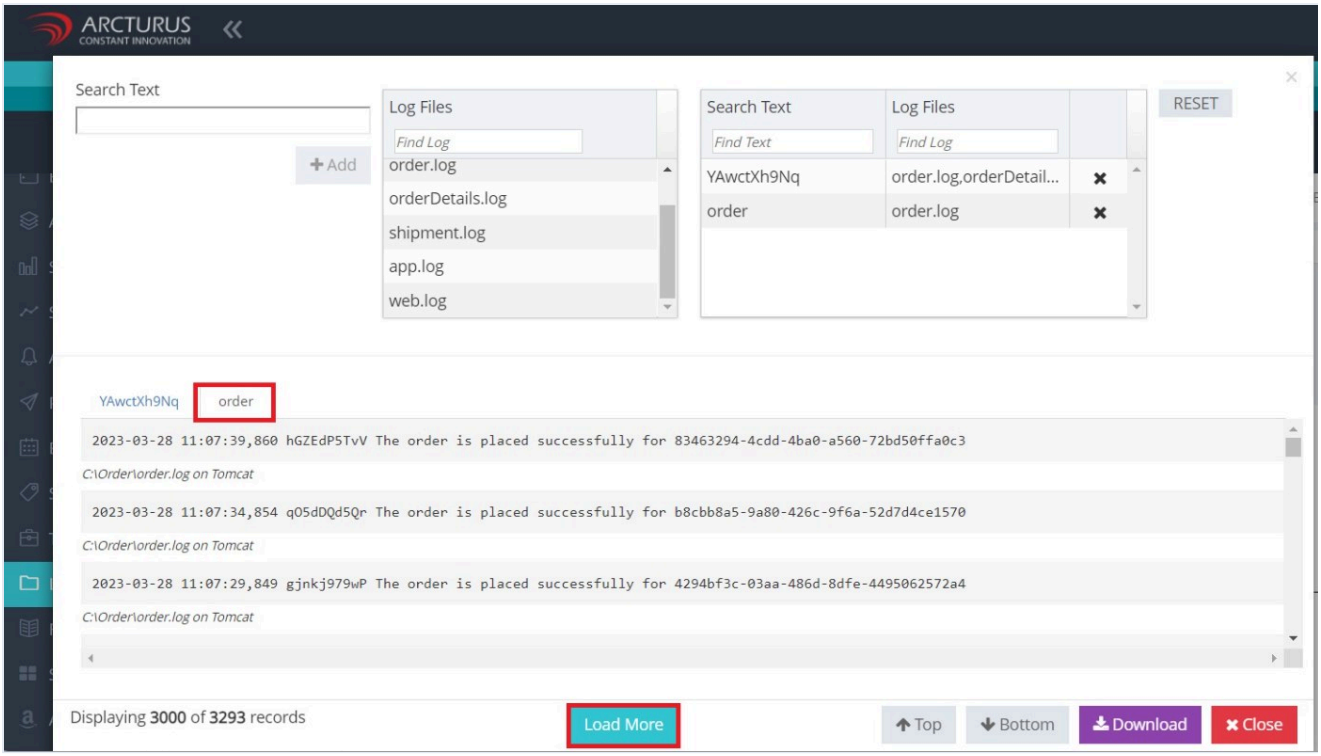
To view the records in the **descending order** based on date-time click the **DESC** button.

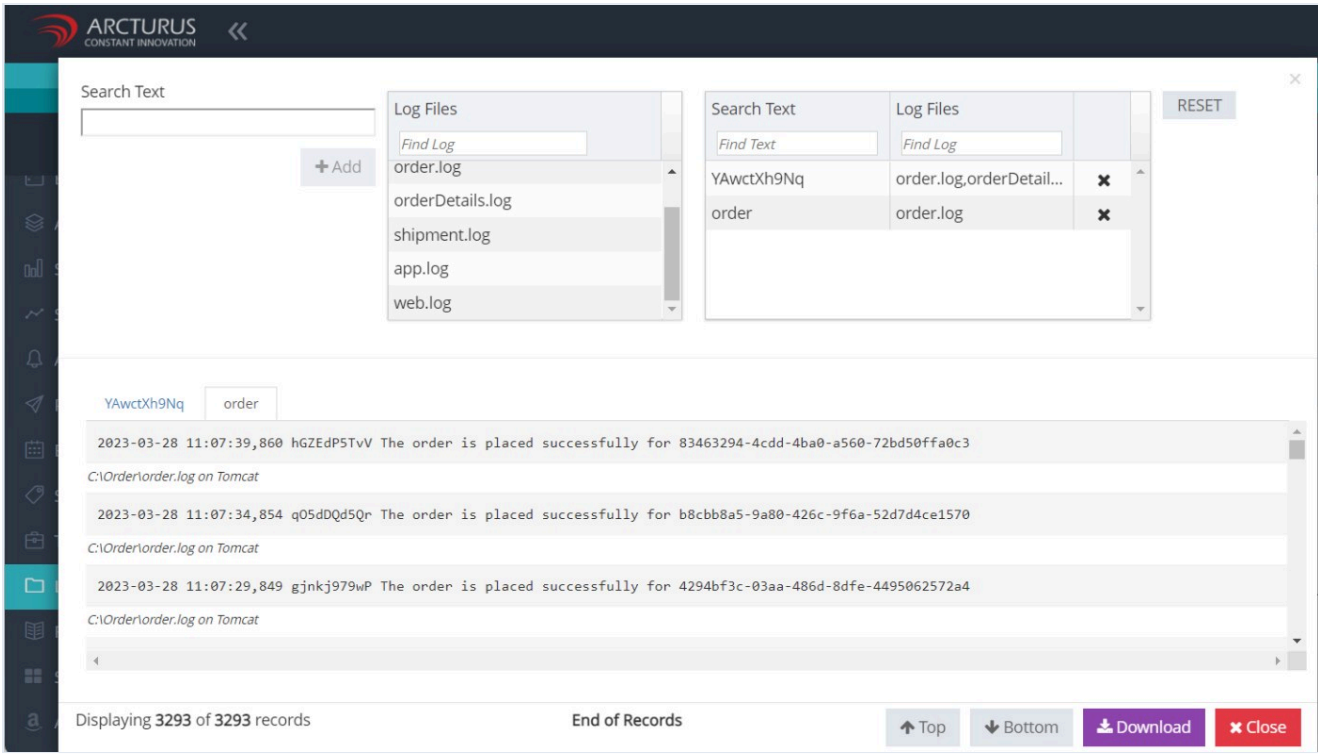


To load the records in the **descending order** based on datetime click the **DESC** button.

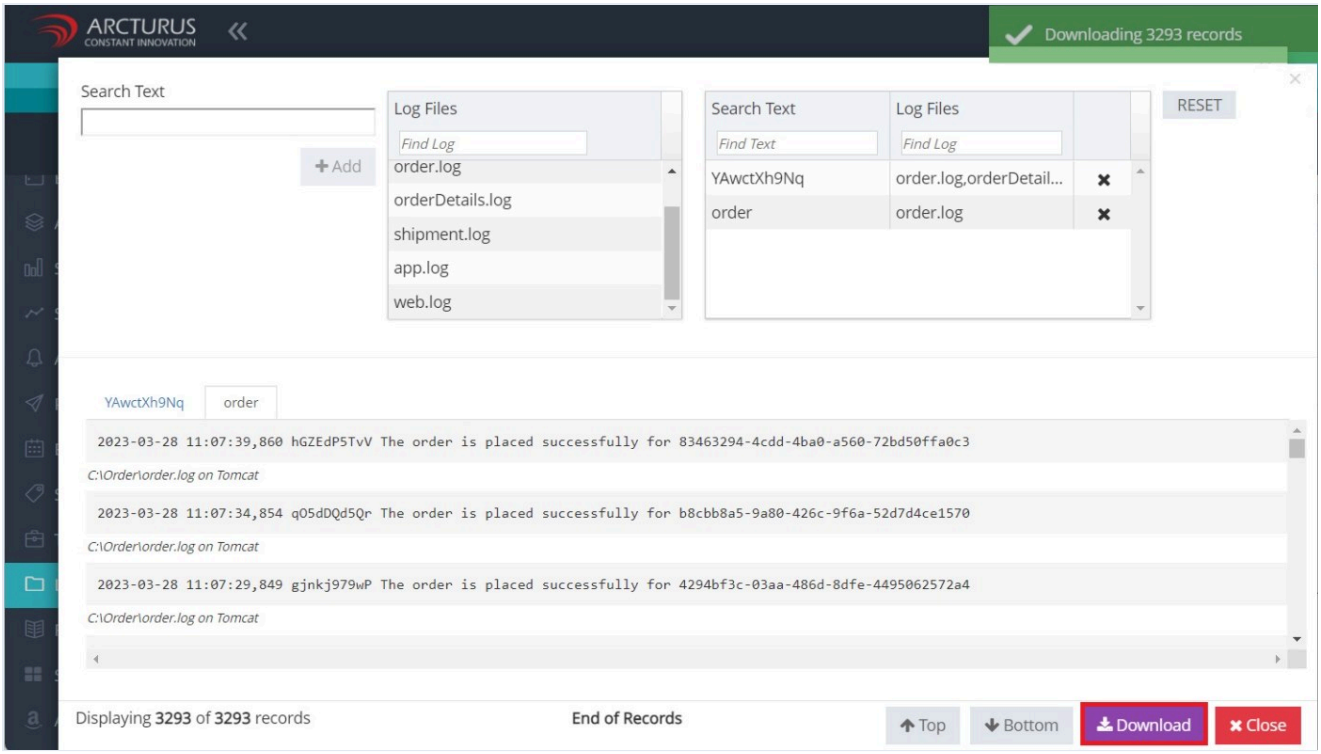


4. If there are more than 1000 records the **Load More** button will be displayed. Click the **Load More** button to load next set of 1000 records.





5. Click the **Download** button to download the records.



Monitoring with Applicare

USING APPLICARE › APPLICATIONS

Date and Regex pattern format

Applicare 9.2 · Updated April 21, 2024 · logs · **Pending technical review**

Sl.No.	Regx	Date Pattern
1	\d{4}-\d{2}-\d{1,2}T\d{1,2}:\d{2}:\d{2}.\d{3}+\d{4}	yyyy-MM-dd'T'HH:mm:ss.SSSZ
2	[A-Za-z]{3,}\s\d{1,2},\s\d{4},\s\d{1,2}:\d{2}:\d{2},\d{1,3}\s[A-Z]{2}\s[A-Z]{3}	MMMMM dd, yyyy, hh:mm:ss,SSS a z
3	[A-Za-z]{3}\s\d{1,2},\s\d{4},\s\d{1,2}:\d{2}:\d{2},\d{1,3}\s[A-Z]{2}\s[A-Z]{3}	MMM dd, yyyy, hh:mm:ss,SSS a z
4	[A-Za-z]{3,}\s[A-Za-z]{3,}\s\d{1,2}\s\d{1,2}:\d{2}:\d{2}\s[A-Za-z]{3,}\s\d{4}	EEE MMM d HH:mm:ss z yyyy
5	[A-Za-z]{3}\s[A-Za-z]{3}\s\d{1,2}\s\d{1,2}:\d{2}:\d{2}\s[A-Za-z]{3,}\s\d{4}	EEE MMM d HH:mm:ss z yyyy
6	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}:\d{1,3}\s[a-zA-Z]{3,}	MM/dd/yy HH:mm:ss:SSS z
7	\d{4}-\d{1,2}-\d{1,2}\s\d{1,2}:\d{2}:\d{2},\d{1,3}	yyyy-MM-dd HH:mm:ss,SSS
8	\d{4}-\d{1,2}-\d{1,2}\s\d{1,2}:\d{2}:\d{2}.\d{1,3}	yyyy-MM-dd HH:mm:ss.SSS
9	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}:\d{1,3}\s[A-Z]{3}	MM/dd/yy HH:mm:ss:SSS z
10	\d{1,2}/[A-Za-z]{3,}/\d{4}:\d{1,2}:\d{2}:\d{2}\s-\d{4}	dd/MMMMM/yyyy:HH:mm:ss Z
11	\d{1,2}/[A-Za-z]{3}/\d{4}:\d{1,2}:\d{2}:\d{2}\s-\d{4}	dd/MMM/yyyy:HH:mm:ss Z
12	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}:\d{1,3}	MM/dd/yy HH:mm:ss:SSS
13	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}	MM/dd/yy HH:mm:ss
14	\d{1,2}/[A-Za-z]{2,}/\d{4}:\d{1,2}:\d{2}:\d{2}	dd/MMMM/yyyy:HH:mm:ss
15	\d{14}	yyyyMMddHHmmss
16	\d{12}	yyyyMMddHHmm
17	\d{8}	yyyyMMdd
18	\d{8}\s\d{6}	yyyyMMdd HHmmss
19	\d{8}\s\d{4}	yyyyMMdd HHmm
20	\d{1,2}-\d{1,2}-\d{4}\s\d{1,2}:\d{2}:\d{2}	dd-MM-yyyy HH:mm:ss
21	\d{4}-\d{1,2}-\d{1,2}\s\d{1,2}:\d{2}:\d{2}	yyyy-MM-dd HH:mm:ss
22	\d{1,2}/\d{1,2}/\d{4}\s\d{1,2}:\d{2}:\d{2}	MM/dd/yyyy HH:mm:ss
23	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}	MM/dd/yy HH:mm:ss
24	\d{4}/\d{1,2}/\d{1,2}\s\d{1,2}:\d{2}:\d{2}	yyyy/MM/dd HH:mm:ss

Sl.No.	Regx	Date Pattern
25	\d{1,2}/\d{1,2}/\d{2}\s\d{1,2}:\d{2}:\d{2}	dd/MM/yy HH:mm:ss
26	\d{1,2}\s[a-z]{3}\s\d{4}\s\d{1,2}:\d{2}:\d{2}	dd MMM yyyy HH:mm:ss
27	\d{1,2}\s[a-z]{4,}\s\d{4}\s\d{1,2}:\d{2}:\d{2}	dd MMMM yyyy HH:mm:ss
28	\d{1,2}-\d{1,2}-\d{4}\s\d{1,2}:\d{2}	dd-MM-yyyy HH:mm
29	\d{4}-\d{1,2}-\d{1,2}\s\d{1,2}:\d{2}	yyyy-MM-dd HH:mm
30	\d{1,2}/\d{1,2}/\d{4}\s\d{1,2}:\d{2}	MM/dd/yyyy HH:mm
31	\d{4}/\d{1,2}/\d{1,2}\s\d{1,2}:\d{2}	yyyy/MM/dd HH:mm
32	\d{1,2}\s[a-z]{3}\s\d{4}\s\d{1,2}:\d{2}	dd MMM yyyy HH:mm
33	\d{1,2}\s[a-z]{4,}\s\d{4}\s\d{1,2}:\d{2}	dd MMMM yyyy HH:mm
34	\d{1,2}-\d{1,2}-\d{4}	dd-MM-yyyy
35	\d{4}-\d{1,2}-\d{1,2}	yyyy-MM-dd
36	\d{1,2}/\d{1,2}/\d{4}	MM/dd/yyyy
37	\d{4}/\d{1,2}/\d{1,2}	yyyy/MM/dd
38	\d{1,2}\s[a-z]{3}\s\d{4}	dd MMM yyyy
39	\d{1,2}\s[a-z]{4,}\s\d{4}	dd MMMM yyyy

USING APPLICARE › APPLICATIONS

LAMA

Applicare 9.2 · Updated April 9, 2026 · logs · **Pending technical review**

LAMA is an API based performance data gathering and early warning system that is designed to help brokerage firms identify and address potential problems before they cause an outage. It collects data from the broker's trading systems and uses machine learning to identify patterns that could indicate a problem. If LAMA detects a potential problem, it will generate an alert so that the brokerage firm and exchange can take preventive corrective action. It is a cost-effective way to prevent outages and reduce the risk of financial losses.

NSE India has adopted LAMA and LAMA reporting is a requirement for brokerage firms.

The data sent from Applicare to LAMA represents a consolidated summary of all servers within the application, rather than details for individual servers.

You can access individual server data directly through the Applicare console. Applicare computes the average of this data and posts the aggregated result to LAMA.

It is important to note that data posting to LAMA occurs only while the controller is actively running.

Median formulae for all the metrics:

Median Calculation Formula

Let:

- T be the sorted list of throughput values.
- n be the number of values in T.
- $T[i]$ represents the i^{th} element in the sorted list.

The median T_{med} is calculated as:

1. **If n is odd**:

$$T_{med} = T\left(\frac{n+1}{2} - 1\right)$$

(Select the middle element from the sorted list.)

2. **If n is even**:

$$T_{med} = \frac{T\left(\frac{n}{2} - 1\right) + T\left(\frac{n}{2}\right)}{2}$$

(Take the average of the two middle elements.)

Hardware Metrics

1. CPU (%) - Machine CPU usage. Available in Enterprise -> Summary.

CPU Average = (Sum of CPU usage across all configured applications) / (Total number of configured applications).

```yaml
Example:
Configured Applications : Nopcom, Petstore
Nopcom CPU values in last 5 minutes -> 10,24,52,34,12
Petstore CPU values in last 5 minutes -> 1,5,20,7,15
CPU Average = (SUM of configured applications CPU) / (Total no. of configured applications CPU)
CPU Average = (10 + 24 + 52 + 34 + 12 + 1 + 5 + 20 + 7 + 15)/ (10)
CPU Average = 18
```

2. Memory (%) - Machine memory usage. Available in Enterprise -> Summary.

Memory Average = (Sum of memory usage across all configured applications) / (Total number of configured applications)

```yaml
Example:
Configured Applications : Nopcom, Petstore
Nopcom Memory values in last 5 minutes -> 45,32,52,60,47
Petstore Memory values in last 5 minutes -> 51,53,35,20,55
Memory Average = (SUM of configured applications Memory) / (Total no. of configured applications Memory)
Memory Average = (45 + 32 + 52 + 60 + 47 + 51 + 53 + 35 + 20 + 55)/ (10)
Memory Average = 45
```

3. Disk (%) - Machine disk usage. Available in Server Analayzer -> OS.

4. Uptime (in minutes) - Availability of the server in the machine for 5 minutes interval. Available in the Enterprise -> Summary.

Exchange DOC

Metric Data

Field Description	Key Name	Data Type	Frequency	Expected Value
CPU Utilization (in %)	cpu	Object	Every 5 min	Float (between 0-100, upto 2 decimals)
Memory Utilization (in %)	memory	Object	Every 5 min	Float (between 0-100, upto 2 decimals)
Disk Utilization (in %)	disk	Object	Every 5 min	Float (between 0-100, upto 2 decimals)
Uptime (in minutes)	uptime	Object	Every 5 min	Long value

Application Metrics

1. Request per second - Indicates the server's throughput. This metric can be found in Server Analyzer -> App Server.

2. Response Time (milliseconds) - Average Response Time - Reflects the server's average response time. You can find this metric in Server Analyzer -> App Server.

3. Trading Failure API count - Failure count of the trading API's. Available in Key Business Transactions Load Distribution.

4. Client Authentication Failure Auth - Failure count of the client authentication API's. Available in Key Business Transactions Load Distribution.

5. History request per second - Historical throughput of the server.

Available in Server Analyzer -> App Server.

6. Historical Response Time (milliseconds) - Represents the average response time of the server.

Available in Server Analyzer -> App Server.

1. Logic to implement the Historical Throughput (HST_THRGHPT) and Historical Latency (HST_LTNCY)

a. Everyday at 11:30 PM the last 24 hours throughput and latency (min, max, med and avg) values will be calculated and save it in a table. So this table will have every day details.

b. On every start of the day the last 21 days average data for the (min, max, med and avg) values are calculated and send the details to LAMA. By Default all the Saturday's and Sunday's data will be ignored in this 21 days data.

Example:

Day	Historical Throughput				Historical Latency				LAMA Data (21 days data)
	Min	Max	Med	Avg	Min	Max	Med	Avg	
1	100	500	125	326	1000	5000	1245	3567	Average of Day1
2	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 2
3	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 3
4	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 4
5	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 5
6	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 6
7	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 7
8	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 8
9	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 9
10	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 10
11	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 11
12	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 12
13	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 13
14	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 14
15	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 15
16	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 16
17	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 17
18	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 18
19	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 19
20	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 20

21	100	500	125	326	1000	5000	1245	3567	Average of Day 1 to 21
22	100	500	125	326	1000	5000	1245	3567	Average of Day 2 to 22
23	100	500	125	326	1000	5000	1245	3567	Average of Day 3 to 23
24	100	500	125	326	1000	5000	1245	3567	Average of Day 4 to 24
25	100	500	125	326	1000	5000	1245	3567	Average of Day 5 to 25
26	100	500	125	326	1000	5000	1245	3567	Average of Day 6 to 26
27	100	500	125	326	1000	5000	1245	3567	Average of Day 7 to 27
28	100	500	125	326	1000	5000	1245	3567	Average of Day 8 to 28
29	100	500	125	326	1000	5000	1245	3567	Average of Day 9 to 29
30	100	500	125	326	1000	5000	1245	3567	Average of Day 10 to 30
31	100	500	125	326	1000	5000	1245	3567	Average of Day 11 to 31
32	100	500	125	326	1000	5000	1245	3567	Average of Day 12 to 32
33	100	500	125	326	1000	5000	1245	3567	Average of Day 13 to 33

7. Log Monitoring - Error codes configured in the LAMA of the configured Applications. Available in Errors screen.

Exchange DOC

Metric Data

Field Description	Key Name	Data Type	Frequency	Expected Value
Log Monitoring (log messages for log levels Error, Fatal and Warning)	Log	String (2000)	Every 5 min	Timestamp, Server Name, Source Name, Log ID, Log Level, Description
Request/Second - throughput of E2E system	throughput	Object	Every 5 min	Min, Max, Avg & Median values
Response Time (in microseconds) – round trip time of E2E system	latency	Object	Every 5 min	Min, Max, Avg & Median values
Trading API Failure Count	failureTradeApi	Numeric	Every 5 min	Long value
Client Authentication Failure Count	failureAuthentication	Numeric	Every 5 min	Long value
Historical Requests/Second	historicalThroughput	Object	Start of Day	Min, Max, Avg & Median values
Historical Response Time (in microseconds)	historicalLatency	Object	Start of Day	Min, Max, Avg & Median values

Network Metrics

1. Bandwidth (%) - It is the average of both the input and output bandwidth utilization of the interface in the router.

```yaml
Configured Mbps = 1000 // 1Gbps  Default value

Input rate in bits / sec – (IfInOctects * 0.8) / 60;
Output rate in bits / sec – (IfOutOctects * 0.8) / 60;

Input Utilization:
Input Utilization (%) = ((Input Rates in bits/sec) / (Configured Mbps * 1000000)) * 100;

Output Utilization:
Output Utilization (%) = ((Output Rates in bits/sec) / (Configured Mbps * 1000000)) * 100;

Bandwidth Utilization = (Input Utilization (%) + Output Utilization (%)) / 2;

Example:
Configured Mbps – 300Mbps
Input Utilization (%) = (47575000 / (300 * 1000000)) * 100;
Input Utilization (%) = 15.86

Output Utilization (%) = (157305000 / (300 * 1000000)) * 100;
Output Utilization (%) = 52.44

Bandwidth Utilization = (15.86 + 52.44) / 2
Bandwidth Utilization (%) = 34.15
```

Important:

The above calculation is the average for a minute. Likewise for LAMA it will be the average for 5 minutes.

2. Packet error count - Sum of current Input and output error count of the interface in the router via SNMP.

ifinErrors

ifOutErrors

```text
Packet error count = (ifinErrors+ ifOutErrors)
```

Important:

The above calculation is the average for a minute. Likewise for LAMA it will be the average for 5 minutes.

Exchange DOC

Metric Data

Field Description	Key Name	Frequency	Data Type	Expected Value
Bandwidth Utilization (in %)	bandwidth	Every 5 min	Object	Float (between 0-100, upto 2 decimals)
Network Latency (in microseconds)	latency	Every 5 min	Object	Long value
Packet Error Count	packetCount	Every 5 min	Numeric	Long value
DNS Lookup Failure Count	lookupCount	Every 5 min	Numeric	Long value

Important:

The Exchanges have decided to deprecate the below-listed parameters. **Network Latency DNS Lookup Count**

Database Metrics

ORACLE DB

1. Replication status - Refers to the availability of the replication process, not the availability of the database itself.

```
TEXT
select process from v$managed_standby where process='<PROCESS_NAME>' fetch first 1 row only
```

Example:

PROCESS_NAME - **MRP0**

Here **MRP0** is the process which is doing the replication. If the process is active, the replication status will be "Up." If the process is not running, the status will be "Down."

2. Replication Queue size - Pending log sync count in the replication.

```
TEXT
SELECT MAX(A.SEQUENCE#) FROM V$ARCHIVED_LOG A,v$archive_dest_status b where (a.dest_id='1' or a.applied ='YES') and a.dest_id=b.dest_id group by b.destination,b.ERROR, a.dest_id, thread# order by a.thread#
```

Here it will give the list of SEQUENCE#. The SEQUENCE# will be sorted in the descending order and the difference of the first two values will be calculated as a queue size.

Example:

```
TEXT
Intital Result
86773
86632
86775

Result after sorting
86775
86773
86632

Queue Size = Value1 – Value2
Queue Size = 86775 – 86773
Queue Size = 2
```

3. Replication Bandwidth (%) - Bandwidth Utilization of the Database.

```
TEXT
select AVG(round((blocks*block_size/1024/1024)/((next_time-first_time)*86400),2)) as
bytesinMbps from v$archived_log where ((next_time-first_time)*86400<>0) and first_time
between to_date('FROM_DATE','YYYY/MM/DD HH24:MI:SS') and to_date('TO_DATE','YYYY/MM/DD
HH24:MI:SS') and dest_id=1
```

FROM_DATE and **TO_DATE** for every 1 minute interval.

Example:

FROM_DATE - 2024/05/02 16:13:00

TO_DATE - 2024/05/02 16:14:00

Important:

The calculation provided reflects a one-minute average. For LAMA, the average is computed over a five-minute period.

4. Replication Latency (milliseconds) - Time delay occurred on the data transfer.

```
TEXT
SELECT VALUE FROM V$DATAGUARD_STATS WHERE UPPER(NAME)='APPLY LAG'
```

MySQL DB

1. Replication status - Refers to the availability of the replication process, not the availability of the database itself.

For MySQL < 8.0.22

```
TEXT
SHOW SLAVE STATUS;
```

For MySQL ≥ 8.0.22

```
TEXT
SHOW REPLICA STATUS;
```

Logic

MySQL < 8.0.22

- **DB Status is UP** when:
  - Slave_IO_Running = `Yes`
  - Slave_SQL_Running = `Yes`
- **DB Status is DOWN** when:
  - Either of the above is not `Yes`

MySQL ≥ 8.0.22

- **DB Status is UP** when:
  - Replica_IO_Running = `Yes`
  - Replica_SQL_Running = `Yes`
- **DB Status is DOWN** when:
  - Either of the above is not `Yes`

2. Replication Queue size - Pending log sync count in the replication.

Logic

```
TEXT
MySQL < 8.0.22
Queue Size = Read_Master_Log_Pos – Exec_Master_Log_Pos

MySQL ≥ 8.0.22
Queue Size = Read_Source_Log_Pos – Exec_Source_Log_Pos
```

3. Replication Bandwidth (%) - Bandwidth Utilization of the Database.

```
TEXT
SHOW GLOBAL STATUS WHERE Variable_name IN ('Bytes_received', 'Bytes_sent');
```

Logic

Step 1: Delta Calculation

```
TEXT
Delta Bytes = Current Value – Previous Value
```

Step 2: Convert to Bits/sec

```
TEXT
Bandwidth (bps) = (Delta Bytes / Time Interval) * 8
```

Step 3: Utilization % for 1 Gbps

```
TEXT
Utilization (%) = (Bandwidth / 1000000000) * 100
```

4. Replication Latency (microseconds) - Time delay occurred on the data transfer.

Logic

```
TEXT
MySQL < 8.0.22
Latency = Seconds_Behind_Master * 1000000

MySQL ≥ 8.0.22
Latency = Seconds_Behind_Source * 1000000
```

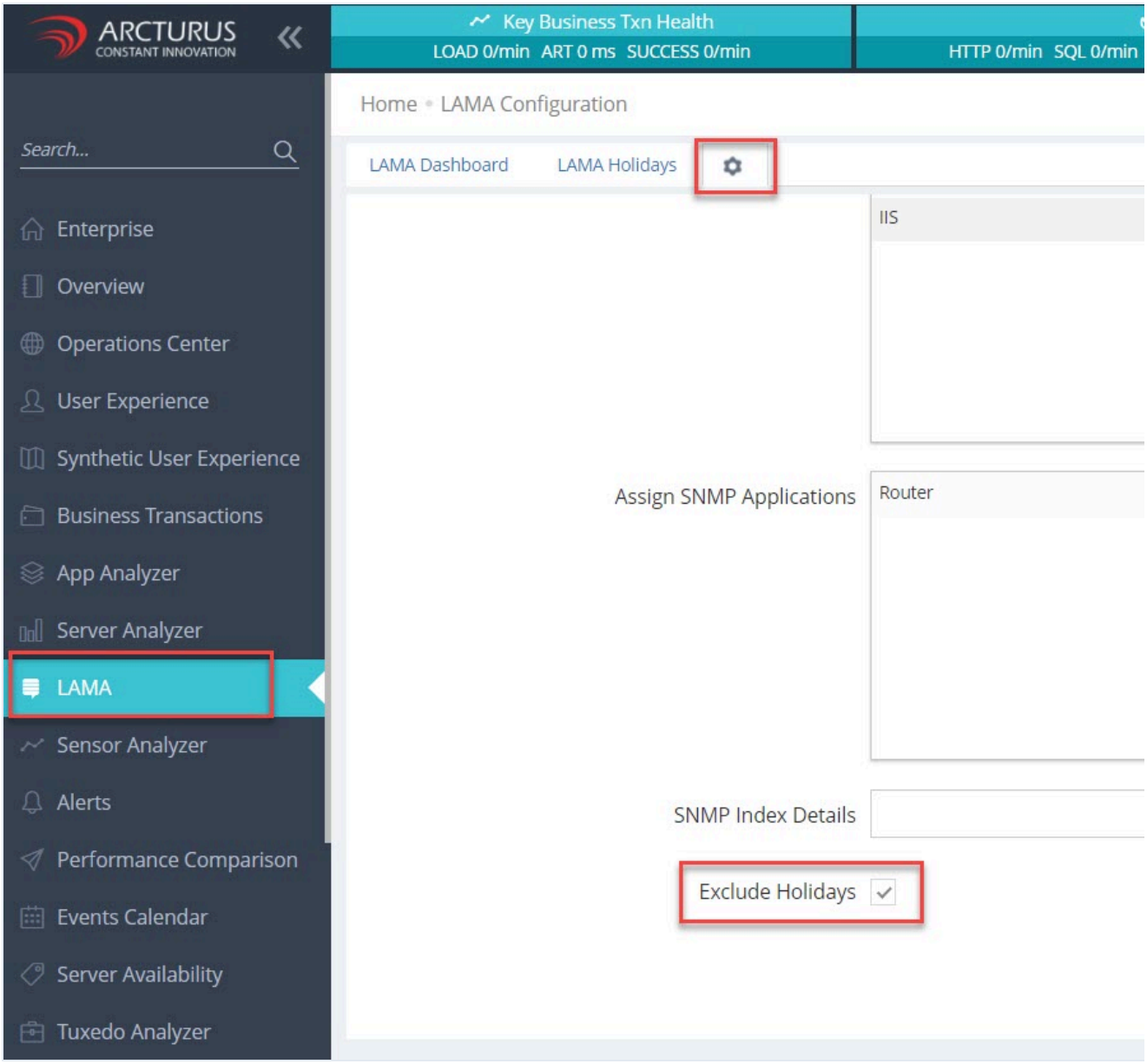
Exchange DOC

Metric Data

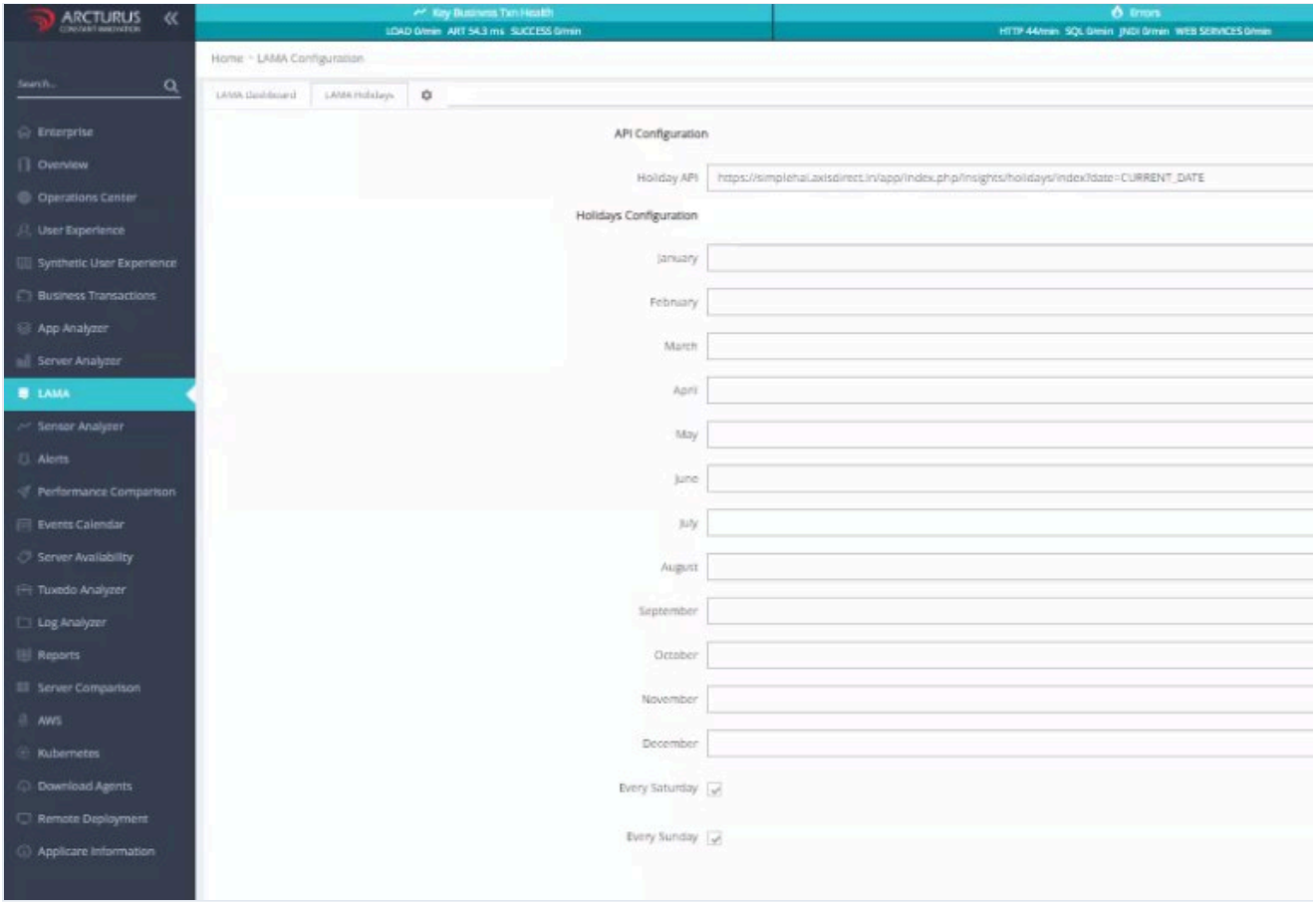
Field Description	Key Name	Frequency	Data Type	Expected Value
Replication Status (1=Up, 0= Down)	status	Every 5 min	Short	1
Replication Queue Size	qSize	Every 5 min	Object	Long value
Replication Bandwidth Utilization (in %)	bandwidth	Every 5 min	Object	Float (between 0-100, upto 2 decimals)
Replication Latency (in microseconds)	latency	Every 5 min	Object	Long value

LAMA Holiday Configuration

If the Exclude Holidays check box is checked in the **LAMA configuration** then both the data collection and data post will be skipped for LAMA based on the LAMA Holiday Configuration.



LAMA Holidays



Holiday API

The holiday will be checked in two conditions.

1. Holiday API is added in the LAMA Holidays screen

Every day after 12:00 AM the api will be triggered once and check whether that day is holiday or not. If yes then both the data collection and data post will be skipped on that day. Otherwise the data collection and data post will happen on that day.

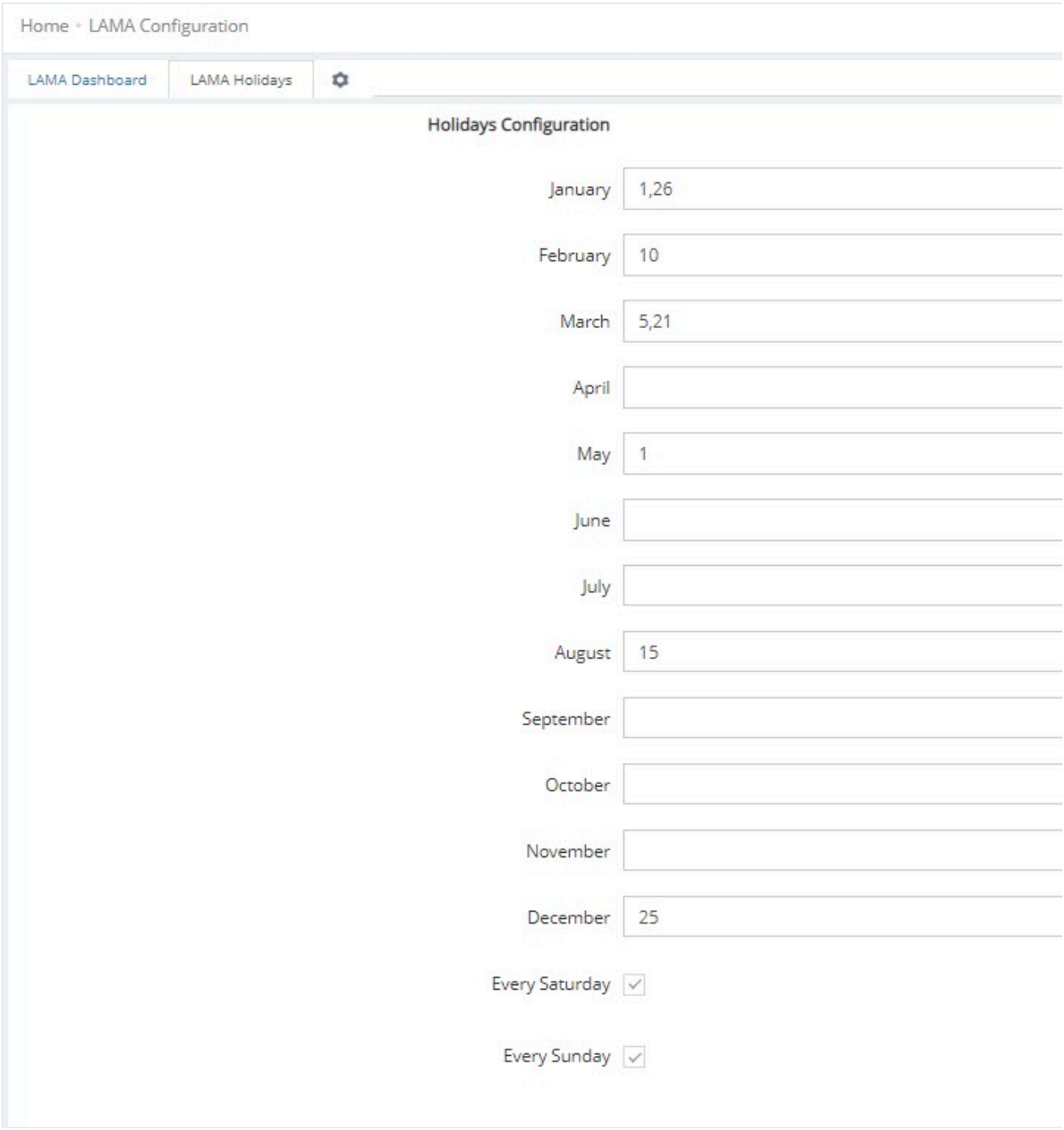


Note:

It is not mandatory to update the holidays in the LAMA Holidays screen. Because controller will automatically update the holiday details from that API. The holiday details will be updated only for that current day.

2. Holiday API is not added in the LAMA Holidays screen

The holiday condition will be fully evaluated based on the days configured in the LAMA Holidays screen.

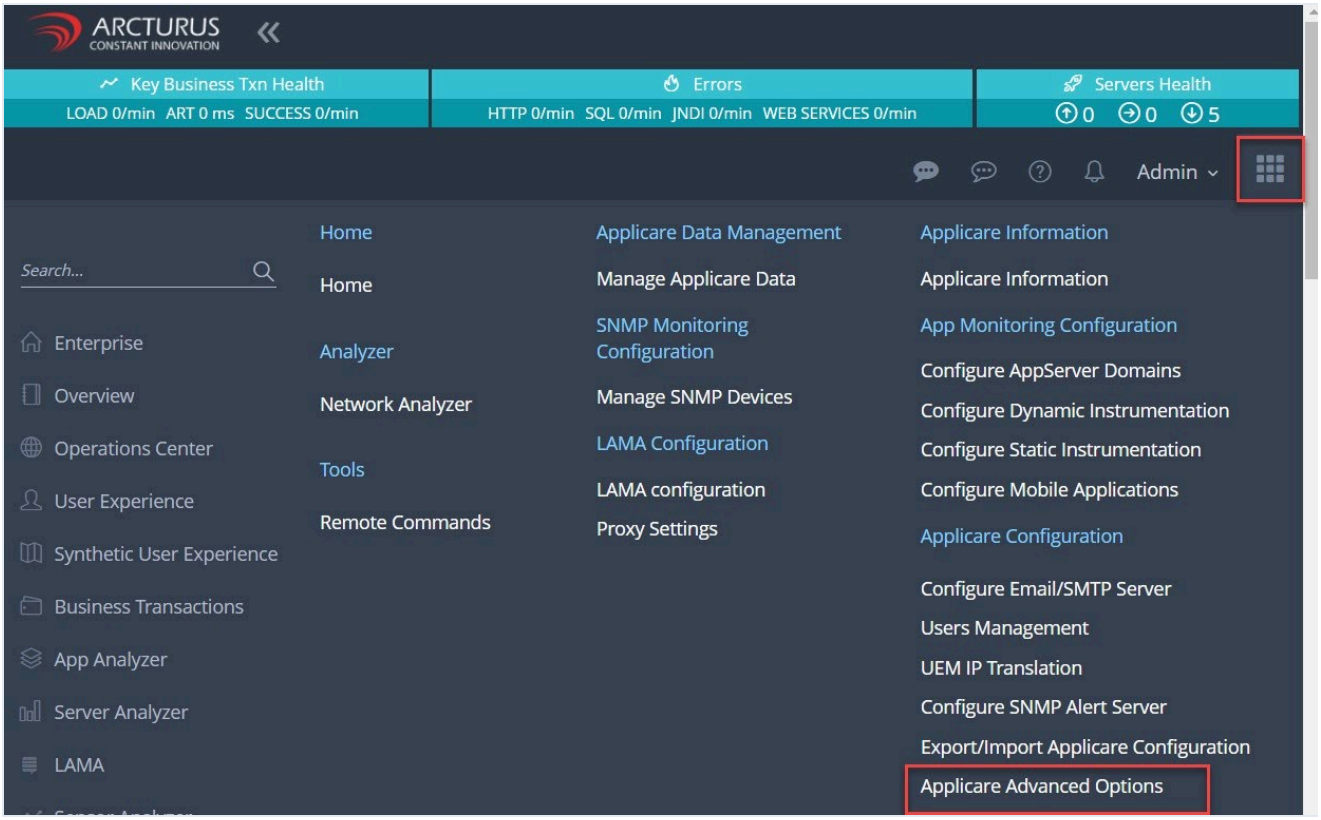


LAMA Logs

By default only **error logs** and the **API check log** for LAMA will be printed in the **ApplicareAdmin.log** file. The log file will be flushed once it reaches 10 MB.

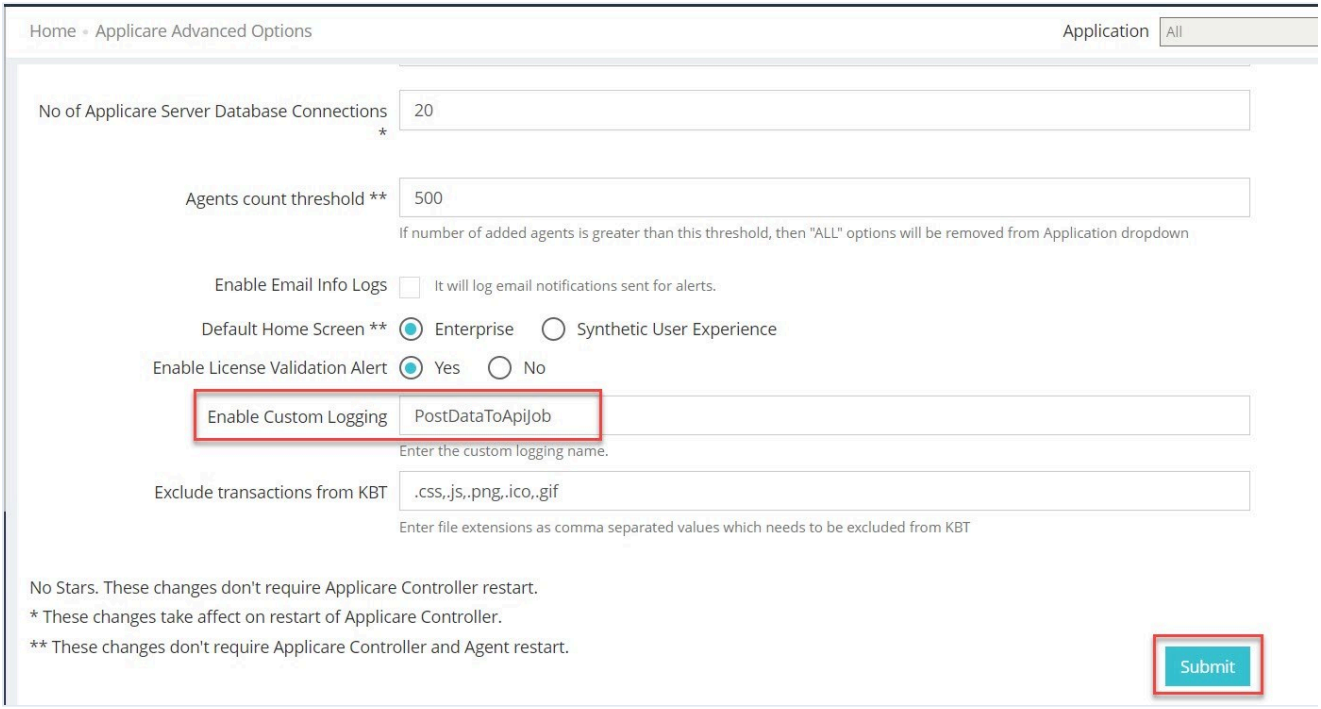
Steps to enable the success log for LAMA.

1. Go to **Applicare Advanced Options** menu.



2. Enter the value **PostDataToApiJob** in the **Enable Custom Logging** field.

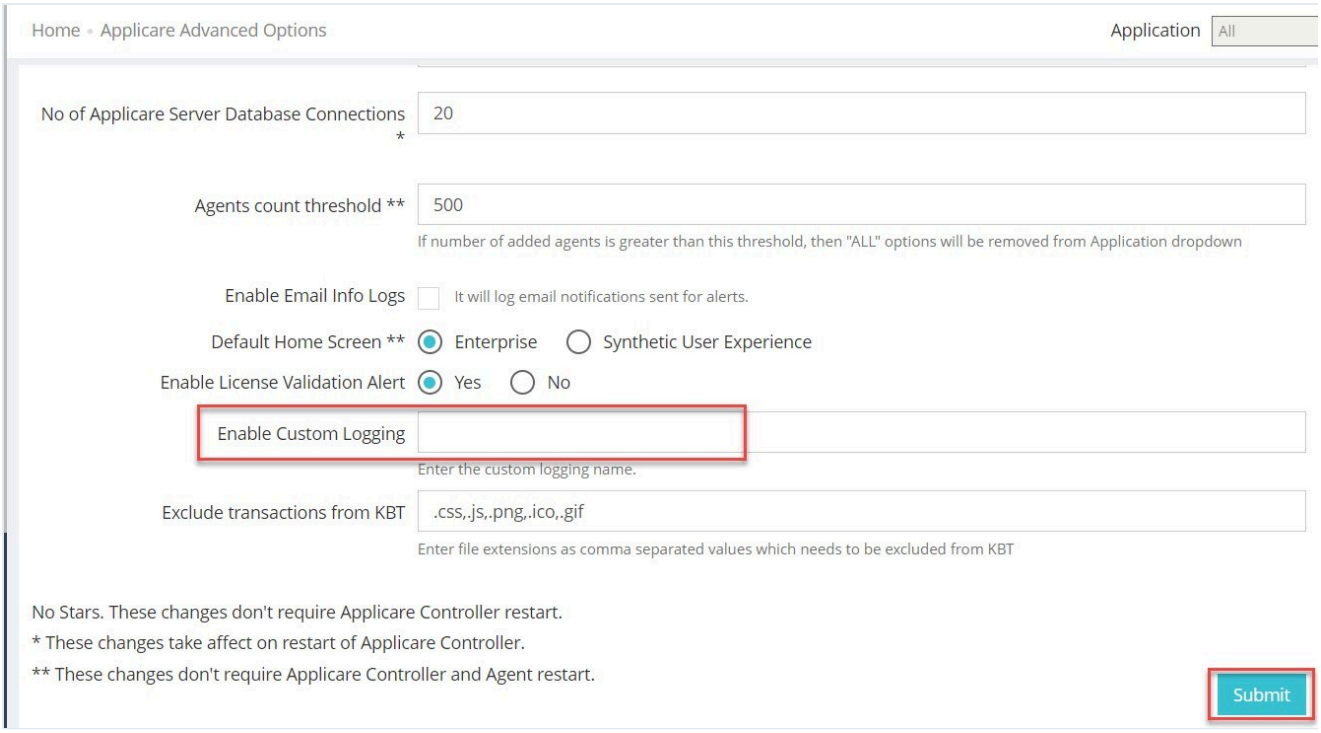
3. Click the **Submit** button to apply and save your changes.



Steps to disable the success log for LAMA.

1. Remove the **PostDataToApiJob** value from the **Enable Custom Logging** field.

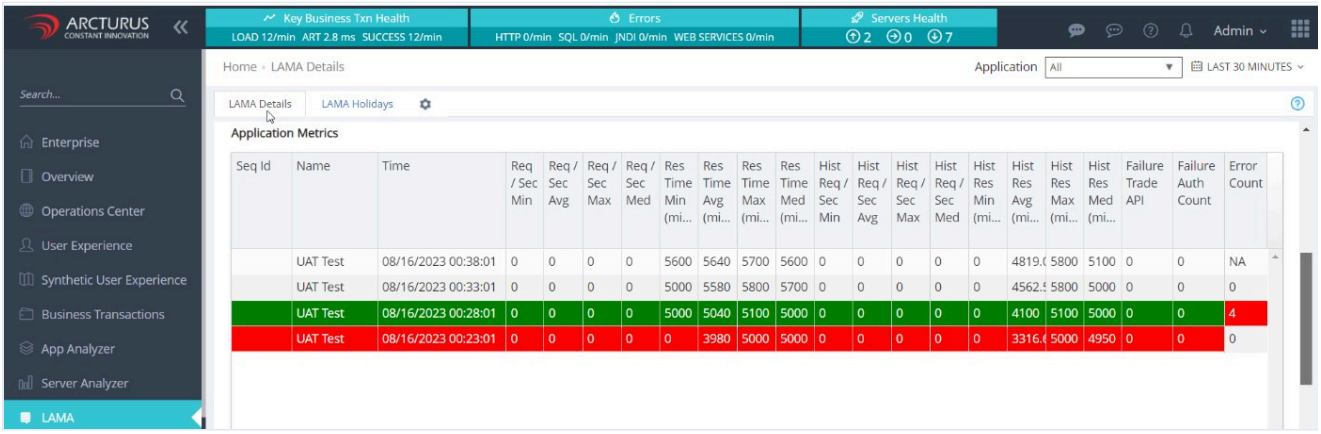
2. Click the **Submit** button to save the changes.



Important:

Controller restart is not required.

LAMA Details



Seq Id	Name	Time	Req / Sec Min	Req / Sec Avg	Req / Sec Max	Req / Sec Med	Res Time Min (mi...	Res Time Avg (mi...	Res Time Max (mi...	Res Time Med (mi...	Hist Req / Sec Min	Hist Req / Sec Avg	Hist Req / Sec Max	Hist Req / Sec Med	Hist Res Min (mi...	Hist Res Avg (mi...	Hist Res Max (mi...	Hist Res Med (mi...	Failure Trade API	Failure Auth Count	Error Count
	UAT Test	08/16/2023 00:38:01	0	0	0	0	5600	5640	5700	5600	0	0	0	0	0	4819.(	5800	5100	0	0	NA
	UAT Test	08/16/2023 00:33:01	0	0	0	0	5000	5580	5800	5700	0	0	0	0	0	4562.5	5800	5000	0	0	0
	UAT Test	08/16/2023 00:28:01	0	0	0	0	5000	5040	5100	5000	0	0	0	0	0	4100	5100	5000	0	0	4
	UAT Test	08/16/2023 00:23:01	0	0	0	0	0	3980	5000	5000	0	0	0	0	0	3316.(	5000	4950	0	0	0

Red color row - Data posting does not occur at regular intervals.

Green color row - Data posting begins occurring at regular intervals.

Error Count Column

1. **Red color** - If there was any errors posted to LAMA this column will be displayed in red color with the total error count. **Click that column to view the error details posted to LAMA**.
2. **No color** with **0** value - No errors have been posted to LAMA.
3. **No color** with **NA** value - No errors have been posted to LAMA.

Updated Metrics Details from the Exchange.

	A	B	C	D	E
1	Metric Type	Metric Key	Value Type	Metric Key Description	Expected Value
2	Application Sequence Metric(ASM)	throughput(THRGHPT)	Request/Second - throughput of E2E system	It defines the number of requests passing through the application/system per second	Count
3		latency(LTNCY)	Response Time (in Milliseconds) – round trip time of E2E system	It defines the time taken for user to submit the request and receive the response of the request (New-Ack Latency)	Milliseconds
4		failureTradeApi(FLD_TRDAPI)	Trading API Failure Count	It refers to the number of failures while executing Trading API.	Count
5		failureAuthentication(FLD_ATH)	Client Authentication Failure Count	It refers to the number of authentication failures.	Count
6		historicalThroughput(HST_THRGHPT)	Historical Requests/Second	21 days summary of Throughput	Count
7		historicalLatency(HST_LTNCY)	Historical Response Time (in Milliseconds)	21 days summary of Response Time	Milliseconds
8	Hardware Sequence Metric(HSM)	cpu(CPU)	CPU Utilization (in %)	the percentage of available time spent is executing the tasks.	Percentage
9		memory(MEM)	Memory Utilization (in %)	the percent of available memory in use at a given moment.	Percentage
10		disk(DSK)	Disk Utilization (in %)	the percentage of available disk space in use at a given moment.	Percentage
11		uptime(UPTM)	Uptime (in minutes)	the time period for which the system has been active and running	Minutes
12	Database Sequence Metric(DSM)	status(status) (Threshold Not required, we only persist data)	Replication Status (1=Up, 0= Down)	Replication status determines the current state of an object being replicated. The replication status of a source object will return either 1=Up or 0=Down	Integer
13		qSize(QSIZE)	Replication Queue Size	It is the number of units/requests/packets waiting in a queue or present in a system.	Count
14		bandwidth(BNDWTH)	Replication Bandwidth Utilization (in %)	It refers to the amount of information that can be sent over a network connection at one time.	Percentage
15		latency(LTNCY)	Replication Latency (in Milliseconds)	Replication latency is the amount of time it takes for a transaction that occurs in the primary database to be applied to the replicate database.	Milliseconds
16	Network Sequence Metric(NSM)	bandwidth(BNDWTH)	Bandwidth Utilization (in %)	Bandwidth utilization refers to a computer network's capacity to transfer data between devices or over the internet in a specific amount of time.	Percentage
17		packetCount(PCKT_CT)	Packet Error Count	It defines the number of packets received with errors.	Count

Monitoring with Applicare

USING APPLICARE › TRANSACTIONS

Business Transactions User Guide

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Overview

Key Business Transactions enables you to capture performance statistics and set SLAs for Web or Java method calls. It also allows you to track individual transactions with detailed performance metrics and call graphs.

Web key business transactions are defined by specifying one or more URI patterns. Any request matching these defined patterns will be intercepted and statistic will be gathered under the given business transaction name.

Java key business transactions operate similarly, but instead of URIs, you define class and method name patterns.

The Profiling Agent must be enabled for business transactions to function on a server. For web based business transactions to work you need to deploy both HTTP Profiling and Business Transactions aspects. You can enable or deploy these on the Configuration -> Configure Dynamic Instrumentation page.

Configuring Business Transactions

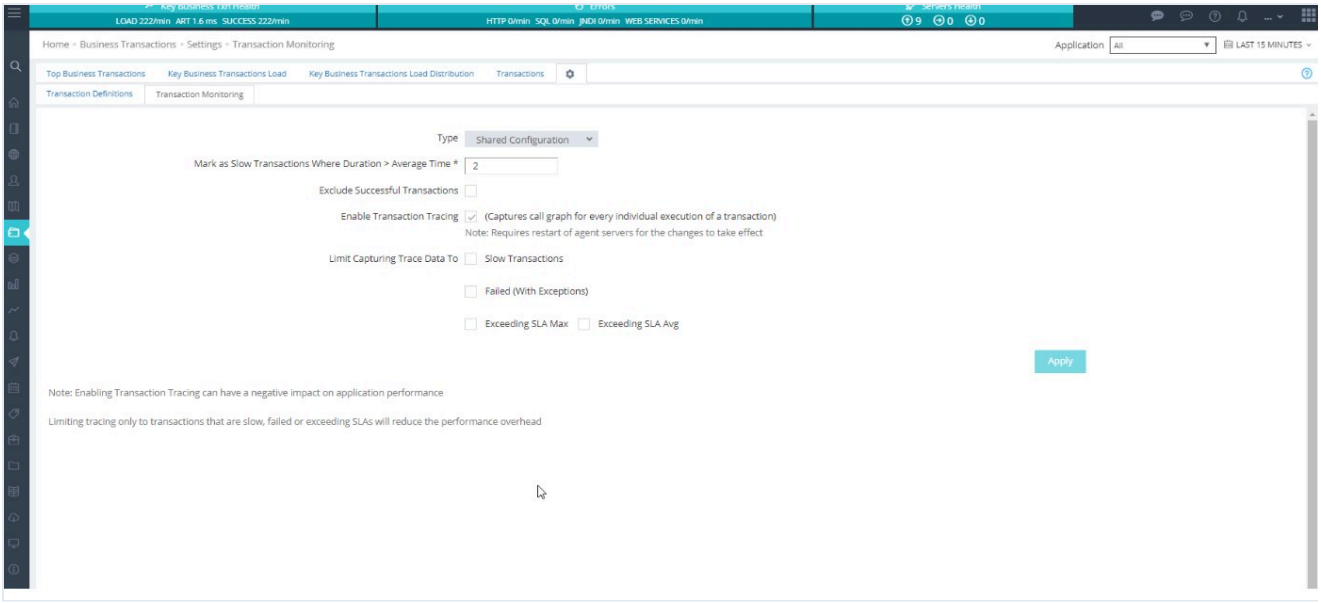
Prerequisites

Before Applicare 7.1.4 you must enable Standard Mode Instrumentation and activate the instrumentation agent on the servers where you want to track business transactions. See Configuring Dynamic Instrumentation on how to enable Standard Mode instrumentation on servers.

For Applicare 7.1.4 onwards At a minimum you must enable HTTP, SQL, JDBC Connection and Business Transactions profiling by deploying those aspects in Configuring Dynamic Instrumentation In Applicare 7.1.4 onwards, 'Enable Per-Transaction Monitoring' in Business Transactions configuration has no impact and BTM wouldn't start until 'Business Transactions' aspect is deployed along with other prerequisites in 'Configure Dynamic Instrumentation'.

Configuration

You can access the business transaction configuration page by navigating to Business Tansactions -> Configure (wheel button on right) Transactions Monitoring. On Transaction Monitoring tab you can configure the amount of details to be captured on configured transactions.



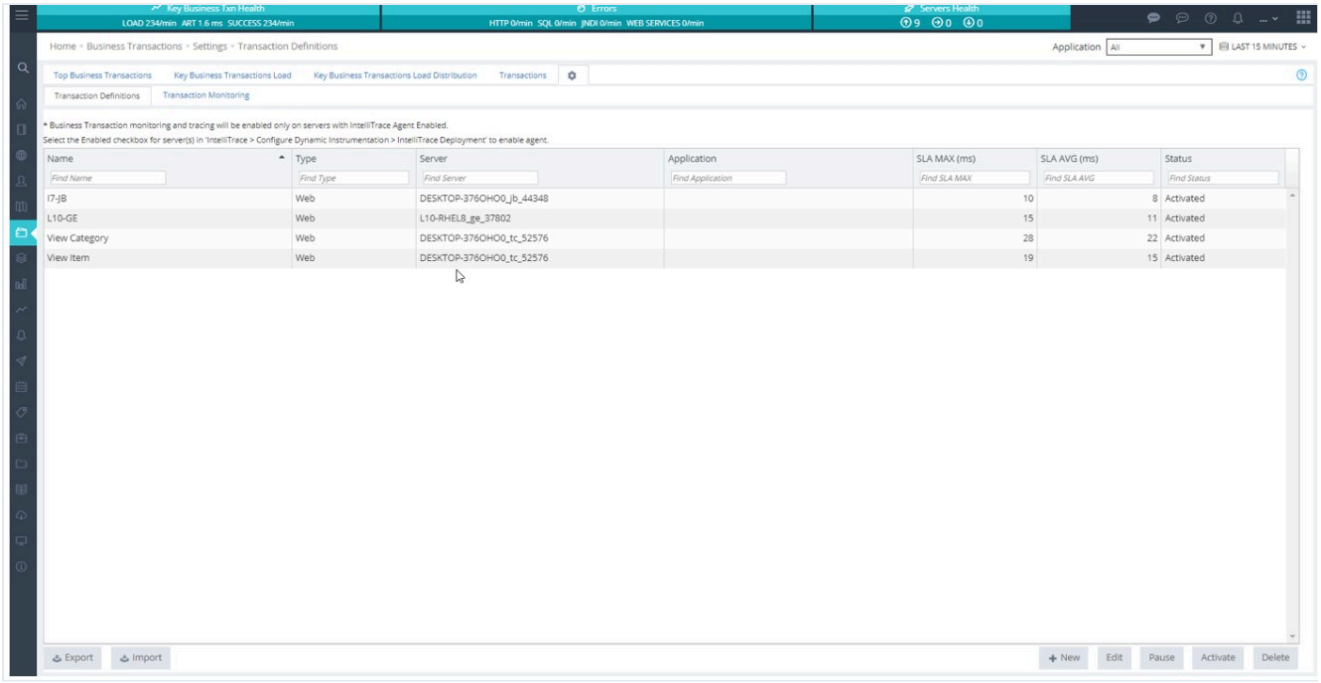
Exclude Successful Transactions allows you to exclude successful/good transactions from Busniss Transaction Monitoring. This helps concentrate on bad transactions which needs a fix or improvement.

Enabling Per-Transaction monitoring gives you the benefit of monitoring your business transactions performance over time as a whole and on a per server basis. Performance overhead of gathering this level of data is minimal and can be enabled even on production systems.

Enabling Transaction Tracing will capture execution graph of each transaction including Java method calls, SQLs and their parameters. Performance overhead of gathering this level of data might be noticable specially in production systems with very high number of business transaction executions per minute. You can reduce the overhead by limiting call graph tracing to transaction executions that are slow, failed or exceeding SLA limits.

Business Transaction Definition

To view the Business Transaction Definition page, navigate to: **Applicare Home -> Business Transactions -> Configure Business Transactions -> Transaction Definitions**.

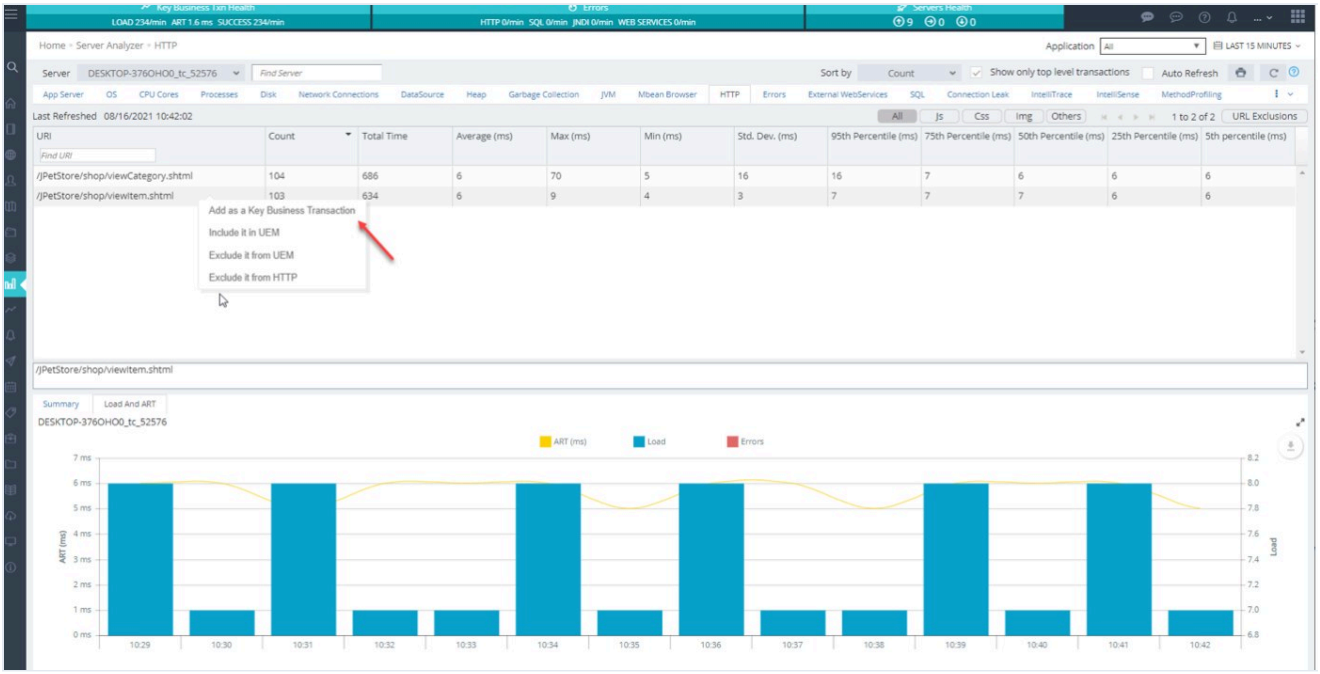


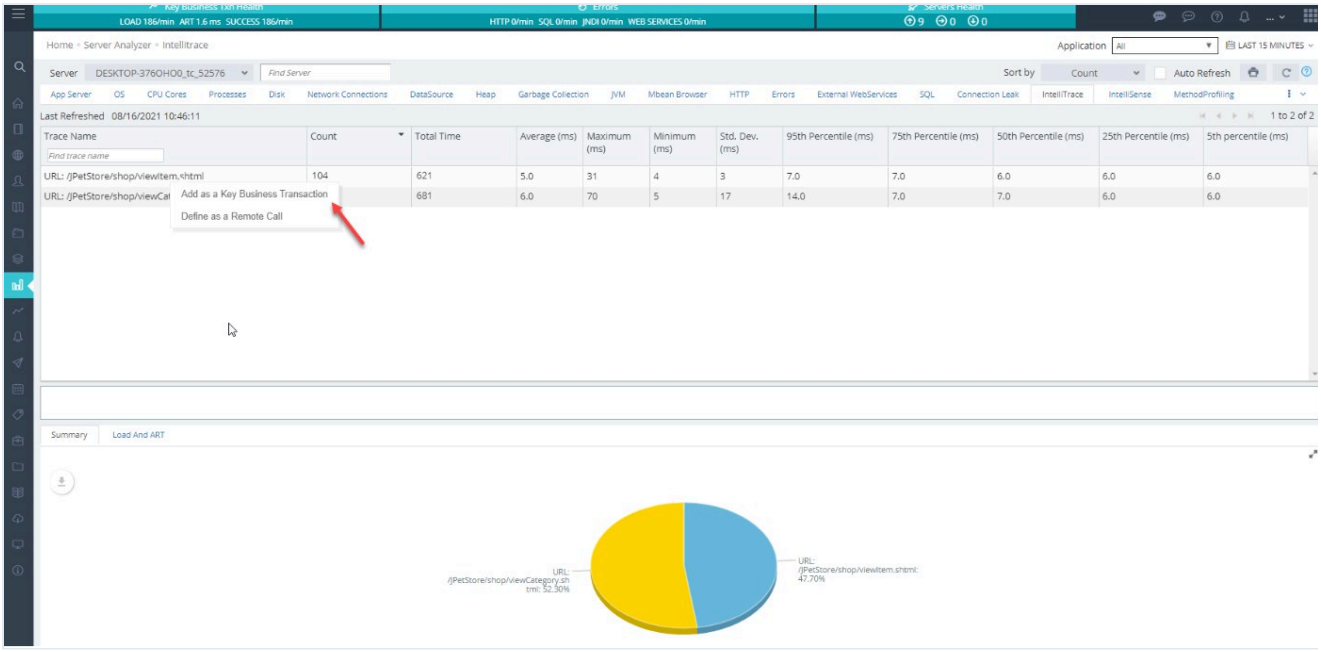
This page shows all defined Key Business Transactions and lets you create, edit, or delete them. Changes made here take effect immediately across the domain without requiring a server restart.

You can export definitions to an XML file or import them into another Applicare environment using the Export/Import buttons.

Defining Business Transactions

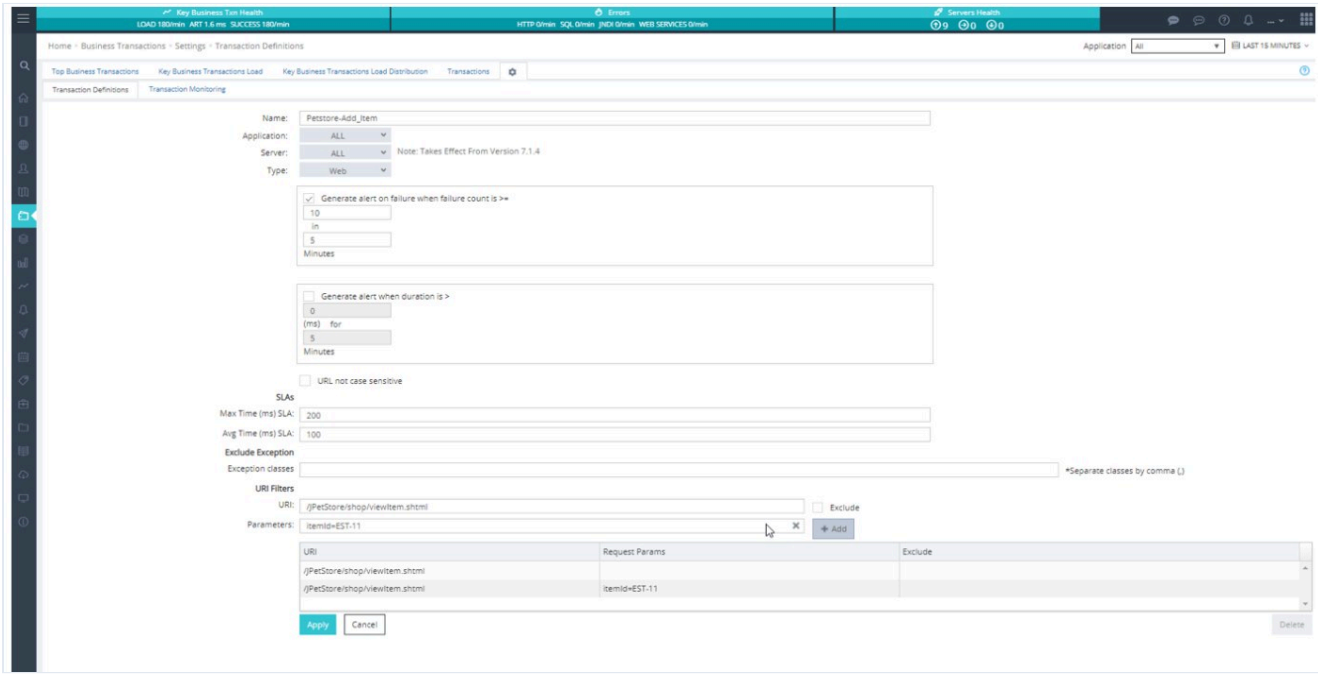
Business transactions can be defined either manually from this page or within the IntelliTrace Analyzer. If IntelliTrace Standard Mode or Advanced Mode with IntelliTrace Profiling is enabled, you can navigate to the HTTP or IntelliTrace tabs in IntelliTrace Analyzer. Right-click on any autodetected URL or Java method to define a Business Transaction for that URL or method.





Add a New Web Business Transaction

To manually add a new Web Business Transaction, click the **New** button on the Transaction Definition page.



1. Provide a **Name** for the Business Transaction

2. Select **Web** from the **Type** drop down box

3. Provide average and maximum SLA values. Leave them blank or set to zero if you don't want to enforce SLA values for this transaction definition

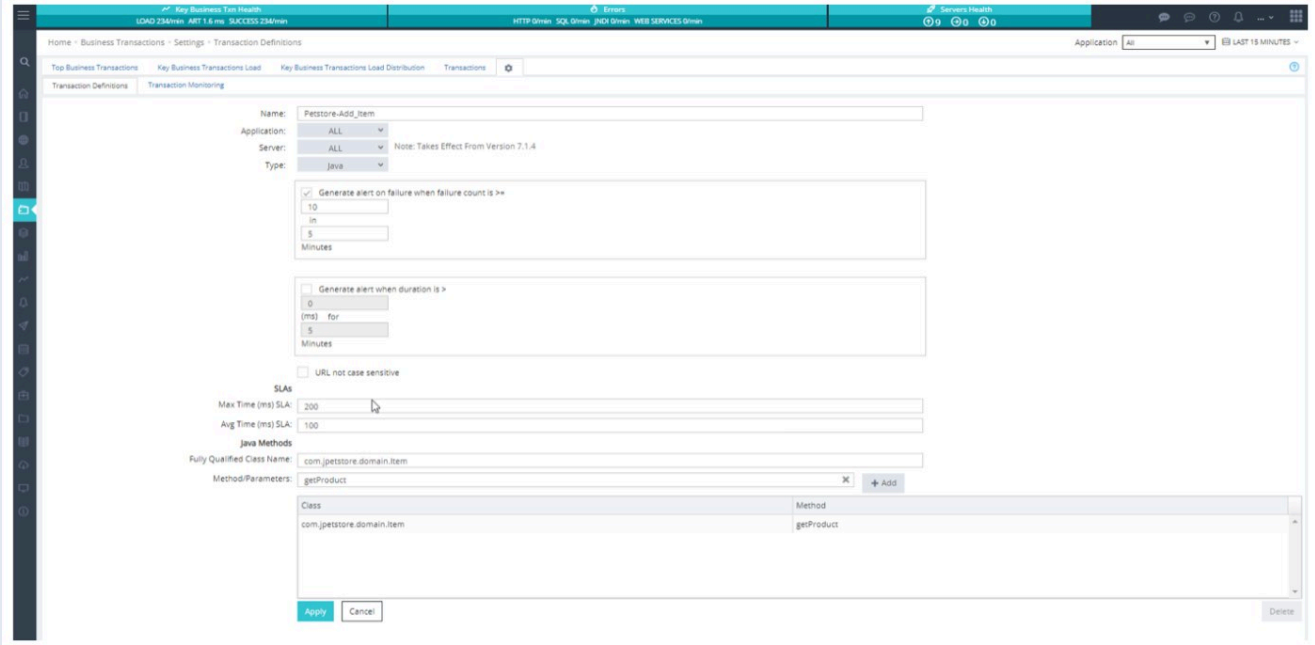
4. Provide a URI pattern (Note: You can use * character as a wild card here)

5. Optionally provide a HTTP parameter pattern if you want to filter further by HTTP parameters. This follows the normal HTTP convention of separating key and value with = sign and separating multiple parameters with & sign. (Note: You can use * character as a wild card for values)

6. Click **Add** button to add the URI pattern to definition. You must add at least one URI pattern. Multiple URI patterns can be added to a definition and HTTP requests matching at least one URI pattern will be included in the Business Transaction. 7. Once you have added all URI Patterns, click **Save** button to save the Business Transaction Definition.

Add New Java Business Transaction

To manually add a new Java Business Transaction, click the **New** button on the Transaction Definition page



1. Provide a **Name** for the Business Transaction

2. Select **Java** from the **Type** drop down box

3. Provide average and maximum SLA values. Leave them blank or set to zero if you don't want to enforce SLA values for this transaction definition

4. Provide fully qualified class name (Note: You can use * character as a wild card for the class name)

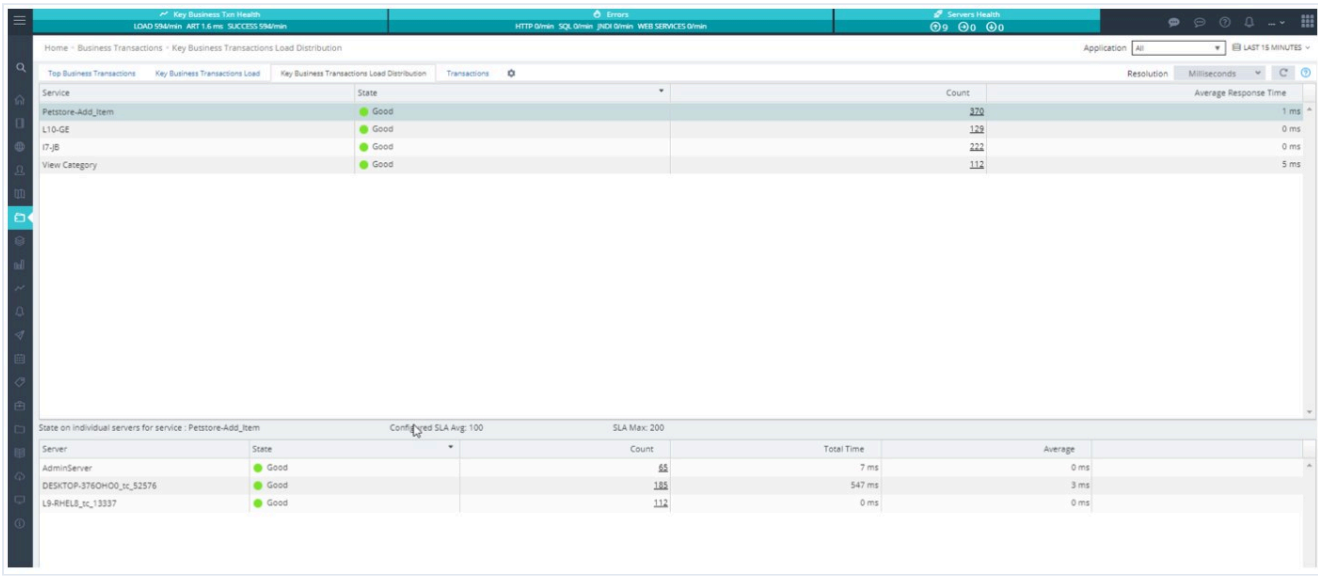
5. Provide a method name (Note: You can use * character as a wild card here)

6. Click **Add** button to add the class/method pattern to definition. You must add at least one class/method pattern. Multiple patterns can be added to a definition and method calls matching at least one pattern will be included in the Business Transaction.

7. Once you have added all patterns, click **Save** button to save the Business Transaction Definition.

Key Business Transactions Load Distribution View

This view displays the state of Key Business Transactions across all servers.



If a Transaction meets SLA values defined for it across all servers, it is shown as in **Good** state. If its not meeting SLA values across all servers, the state will be **Failed**. If its not meeting SLA values on some servers then the state will be **Partial Failure**.

Clicking on a Business Transaction will display the state of the transaction on individual servers in the lower table.

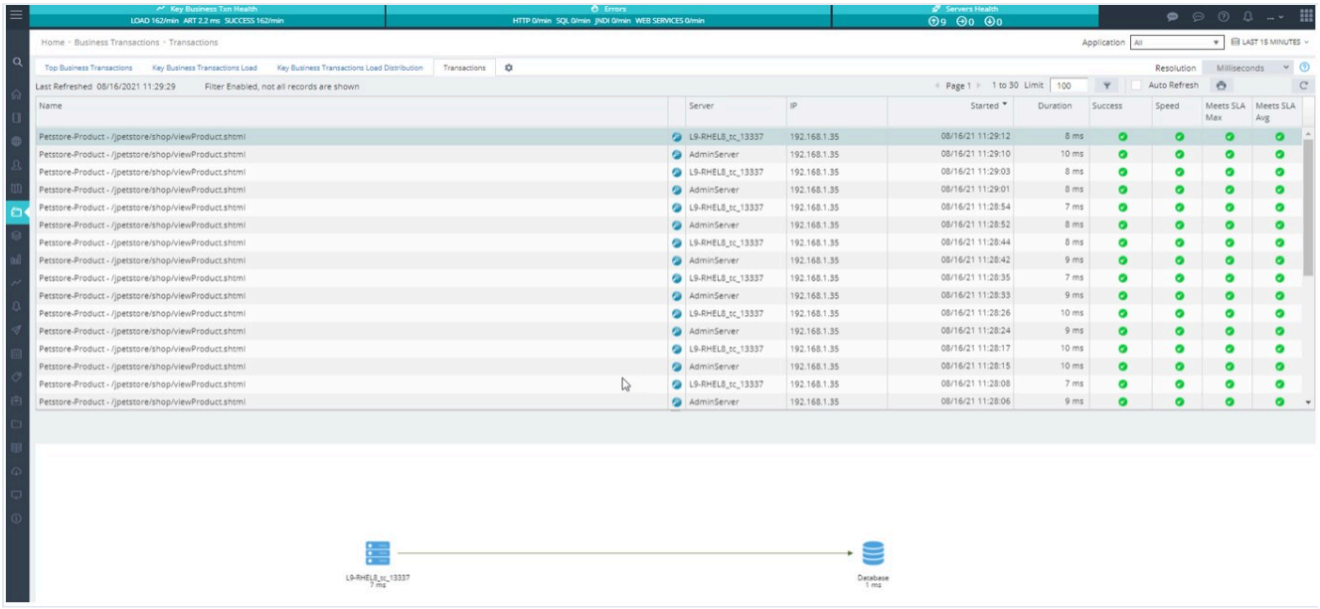
Double clicking on a Transaction in the top table will take you to the Transactions tab and will set the search filter to the selected transaction.

Double clicking on a Transaction in the bottom table will take you to the Transactions tab and will set the search filter to the selected transaction and server.

The states displayed on Summary View are calculated periodically for performance reasons and are not updated real time.

Transactions View

Transaction view displays transaction execution stats if Per-Transaction Monitoring is enabled. Double clicking on any transaction displayed here will take you to the detailed view where you can see the in parameters, exceptions if any and trace details if transaction tracing is enabled.

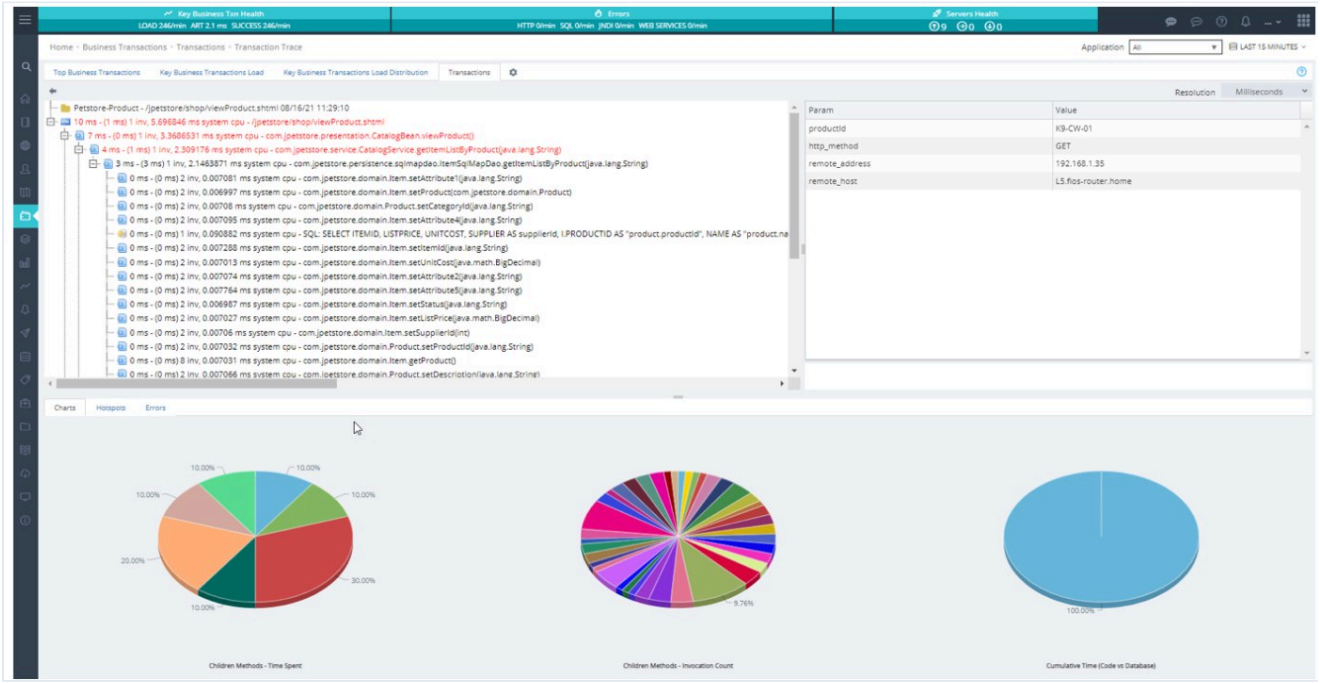


Success - If transaction was completed normally without an exception thrown from the method (applies only to Java transactions).

Speed - If transaction duration was less than the average * (User defined value in Business Transaction Configuration).

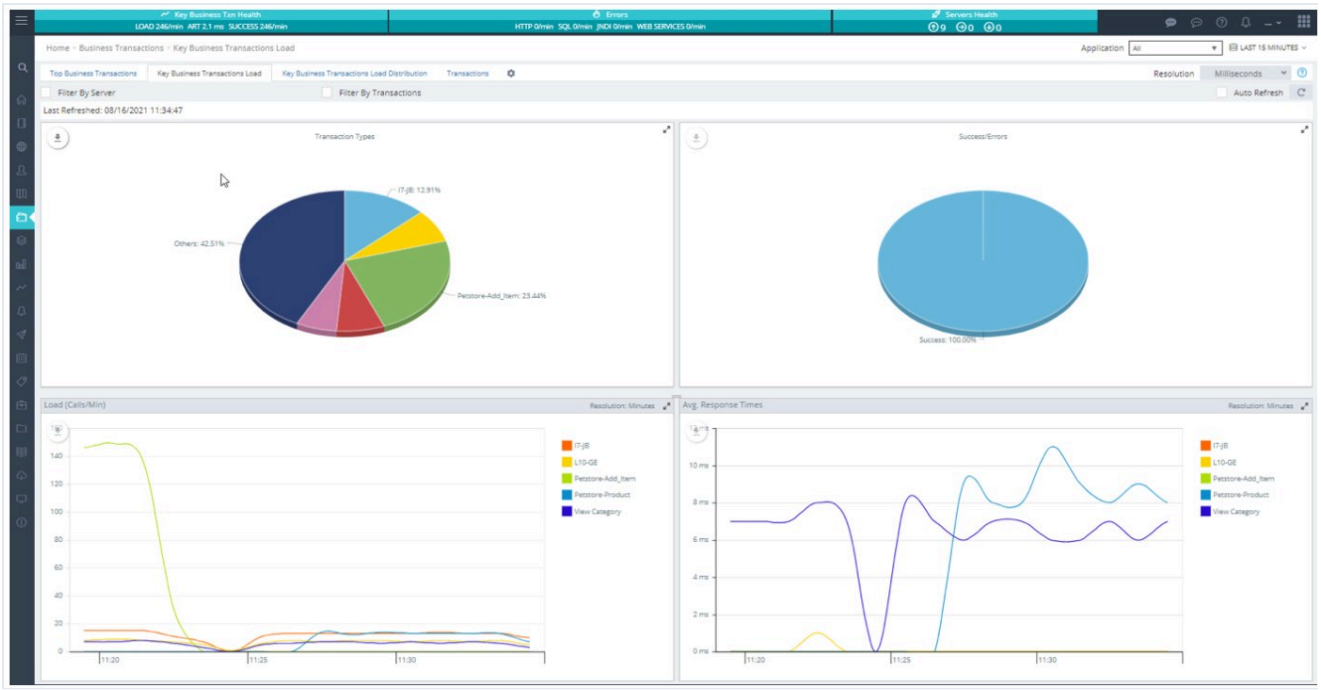
Meets SLA Max - Indicates transaction duration met the configured SLA Max value.

Meets SLA Avg - Indicates transaction duration met the configured SLA Average value.



Key Business Transactions Load View

Using this tab you can view the performance stats of key business transactions across the environment as well as on a per server basis. You must enable Business Transaction Profiling in order to gather data to be displayed on the dashboard.



Configuring Distributed Transaction Tracing

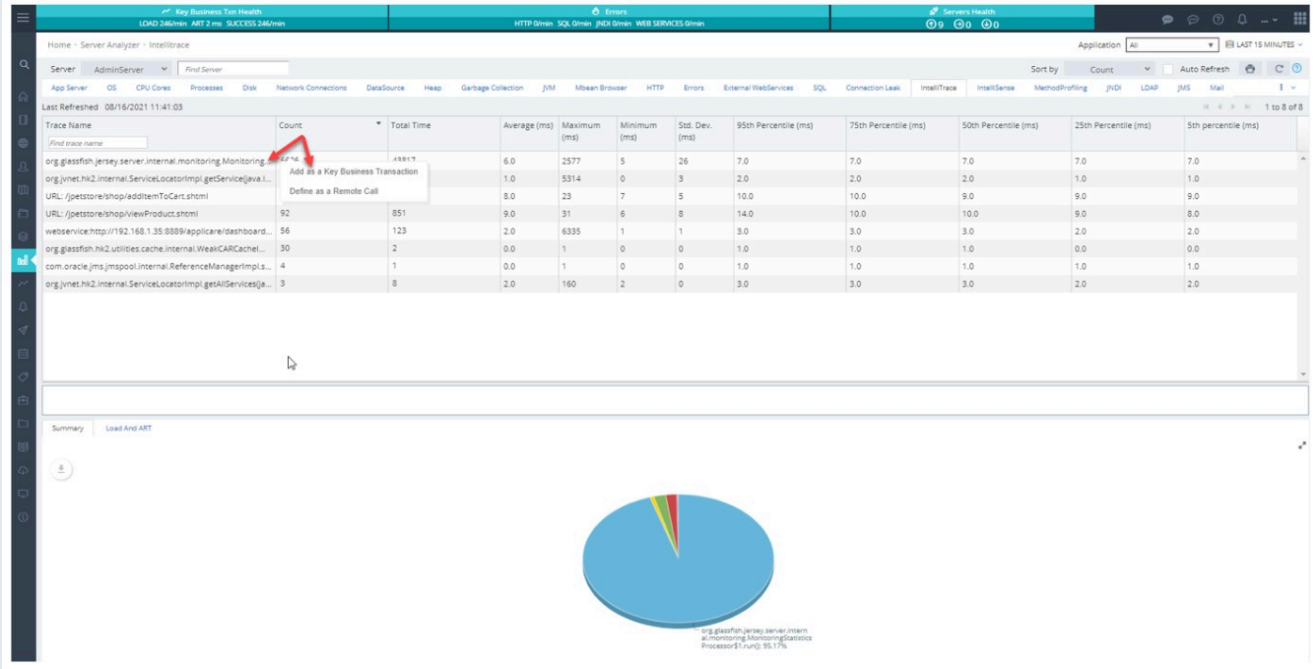
Applicare BTM can track transactions that span multiple servers (JVMs) and display the consolidated call graph consisting of executions on multiple servers in a single UI.

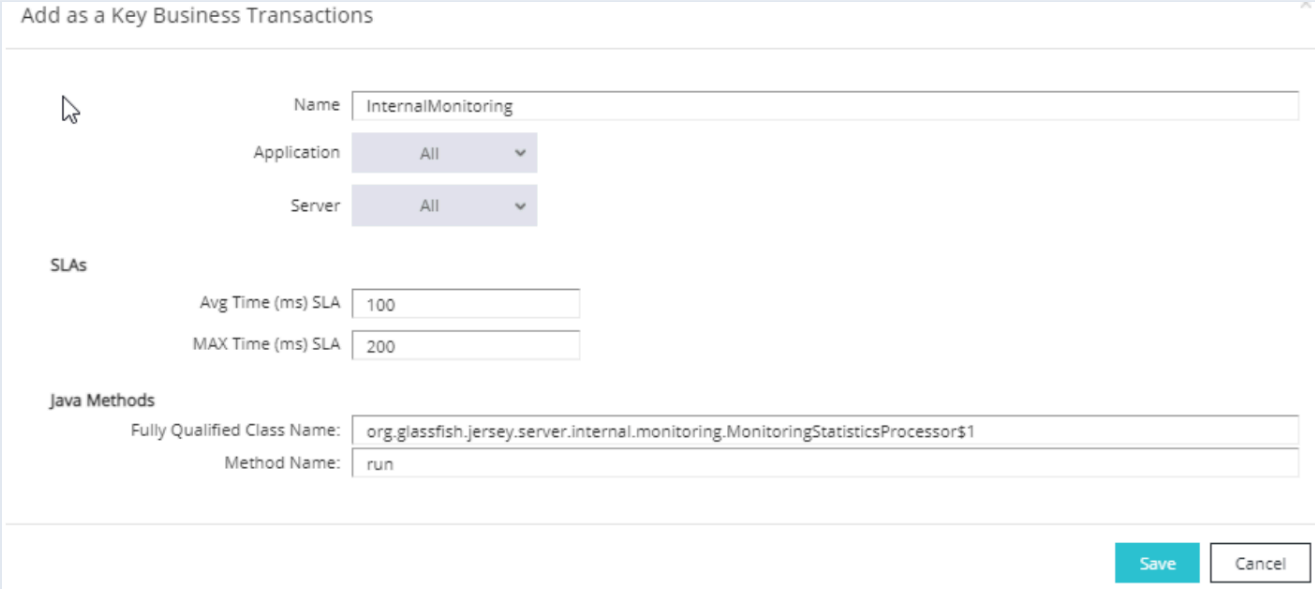
Note: All servers that participates in distributed transactions must have their system clocks synchronized (E.g. Using NTP) for remote transaction tracing to work properly.

There are two ways to enable transaction tracking between two servers.

First Method

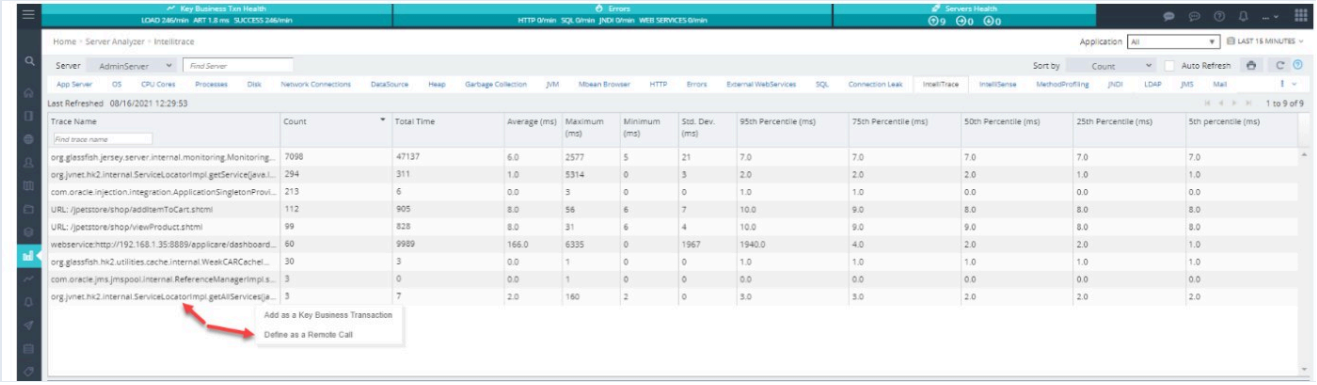
First you must define the remote service as a Java business transaction. You can right click your service method which accepts the remote parameters in IntelliTrace window and select "Define as a Business Transaction" option.

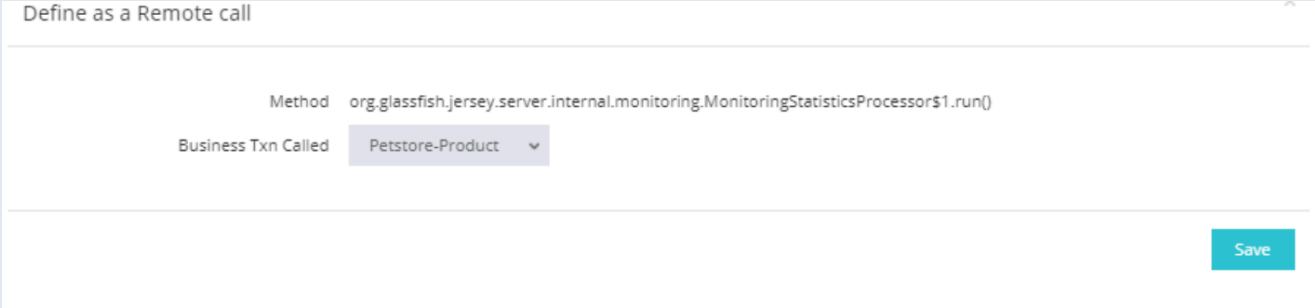




In the above example we are defining the org.glassfish.jersey.server.internal.monitoring.MonitoringStatisticsProcessor$1.run() method which is the service method that parameter as a Business Transaction named "**Petstore-Product**".

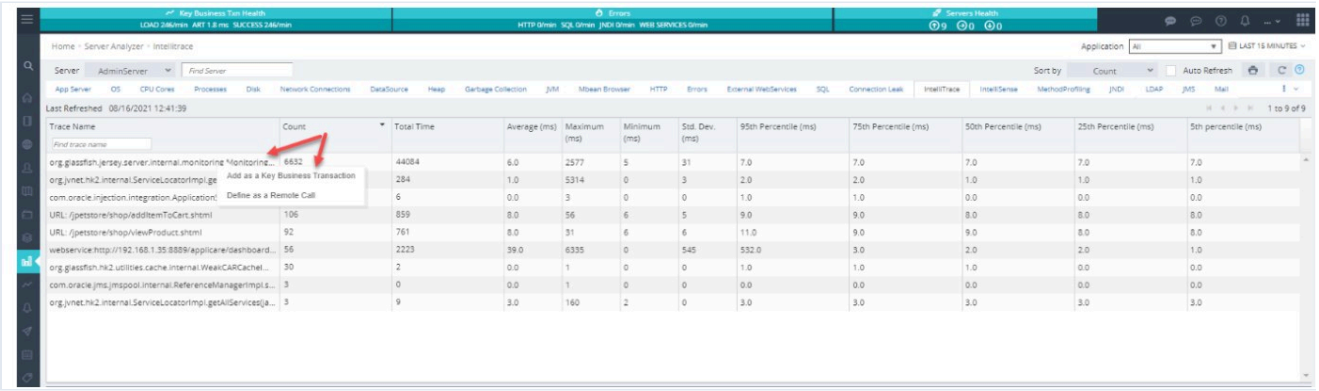
Next we define the method that calls the remote service as a remote method by right clicking the method name in IntelliTrace or in the BTM trace window and selecting "**Define as a Remote Call**" option. Be sure to choose the method that accepts the parameters which are sent to the remote service.

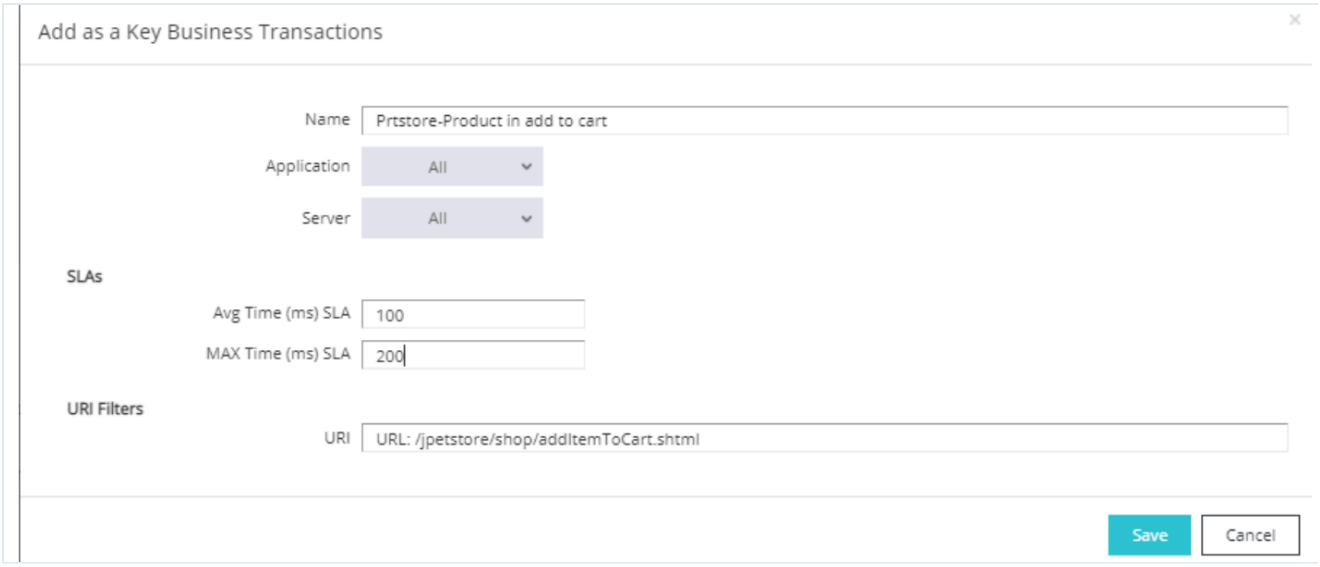




In the example above we define the method jpetstore/shop/addItemToCart.shtml as a remote method call and we select the previously defined **Pet store-Product in add to cart** service as the Business Transaction called by this method.

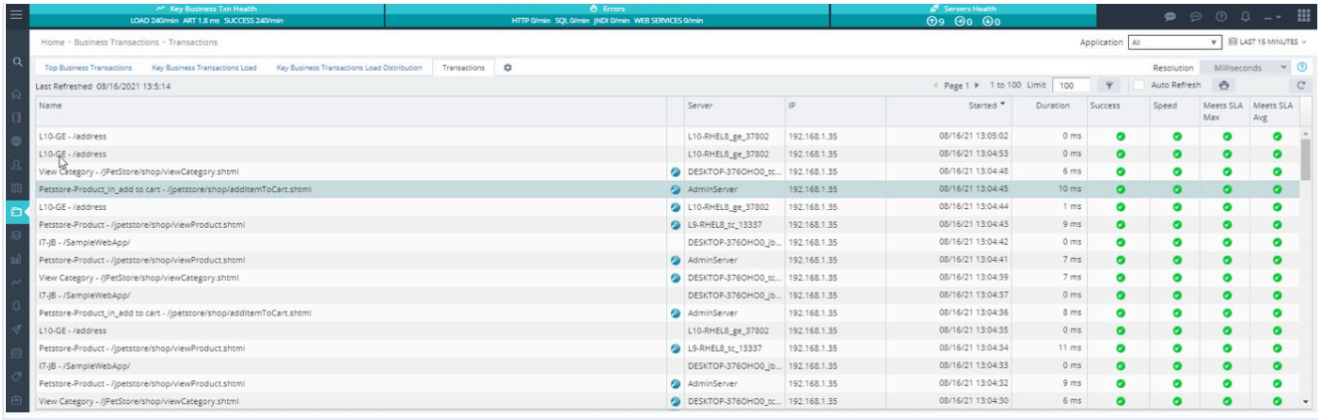
Next we will define the HTTP call which makes the remote method call as a business transaction too

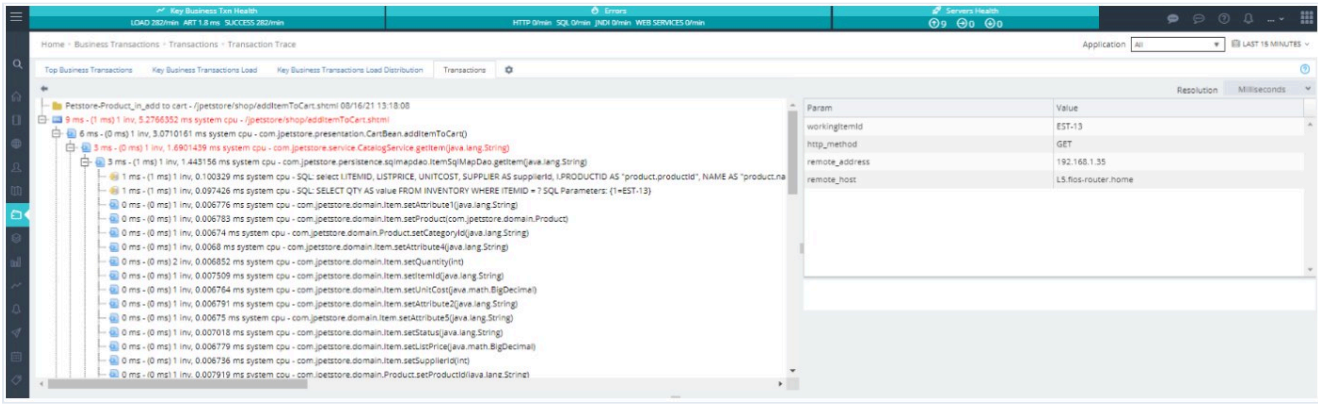




Now that both transactions and remote method calls are defined, you can monitor the transactions as they are invoked on the server and track their flow across different servers.

The following two images illustrate a **Validate CC** transaction invoked on the AdminServer. By double-clicking, you can view the call graph for the AdminServer and the call graph for the remote service on the Tomcat server, which was invoked as part of the transaction on the AdminServer.



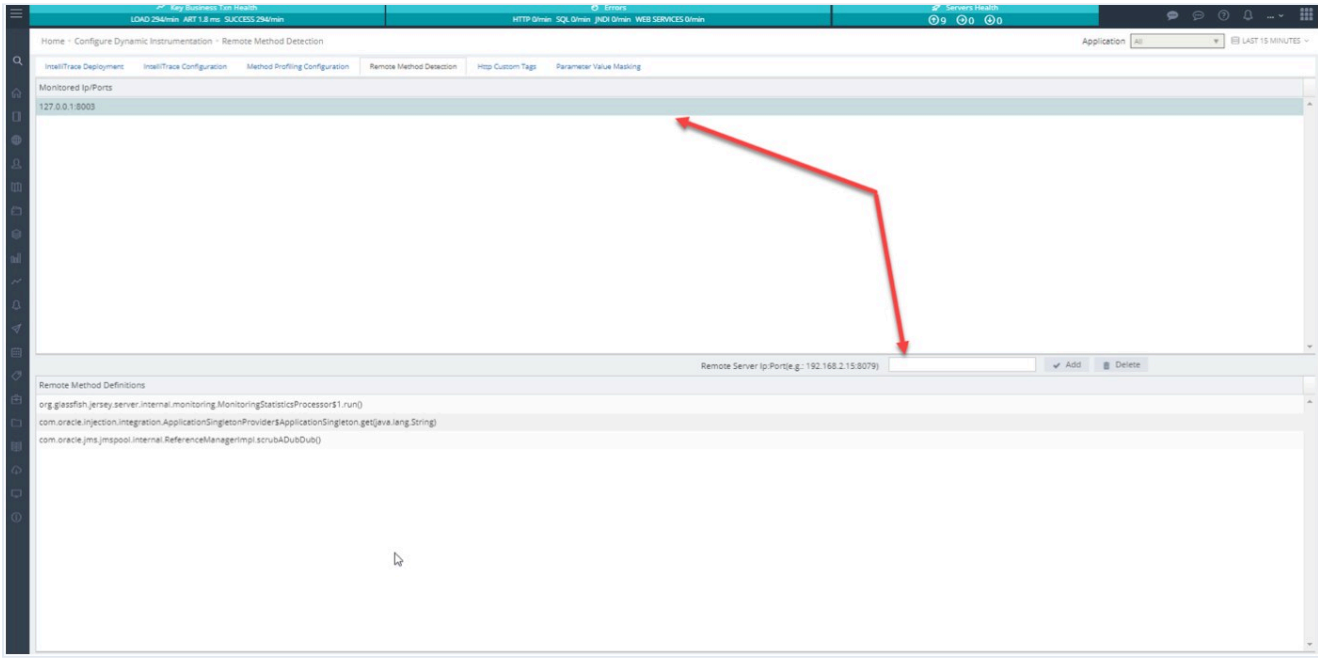


Second Method

Second approach is to use Network Connection Profiling to track transactions and as in the first method, you will have to define the calling and the called business transactions. You will not need to define the method that calls the remote transaction as a Remote Method.

For this approach to work, Network Connection Profiling must be enabled on all participating servers. You can do this by going to **Profiling -> Configure Dynamic Instrumentation** page and deploying Network Connection Profiling in Advanced mode.

Once you enable Network Connection Profiling, go to the Remote Method Definition tab on the same page and add the server IP and port where the remote transactions are running on.



Monitoring with Applicare

USING APPLICARE › TRANSACTIONS

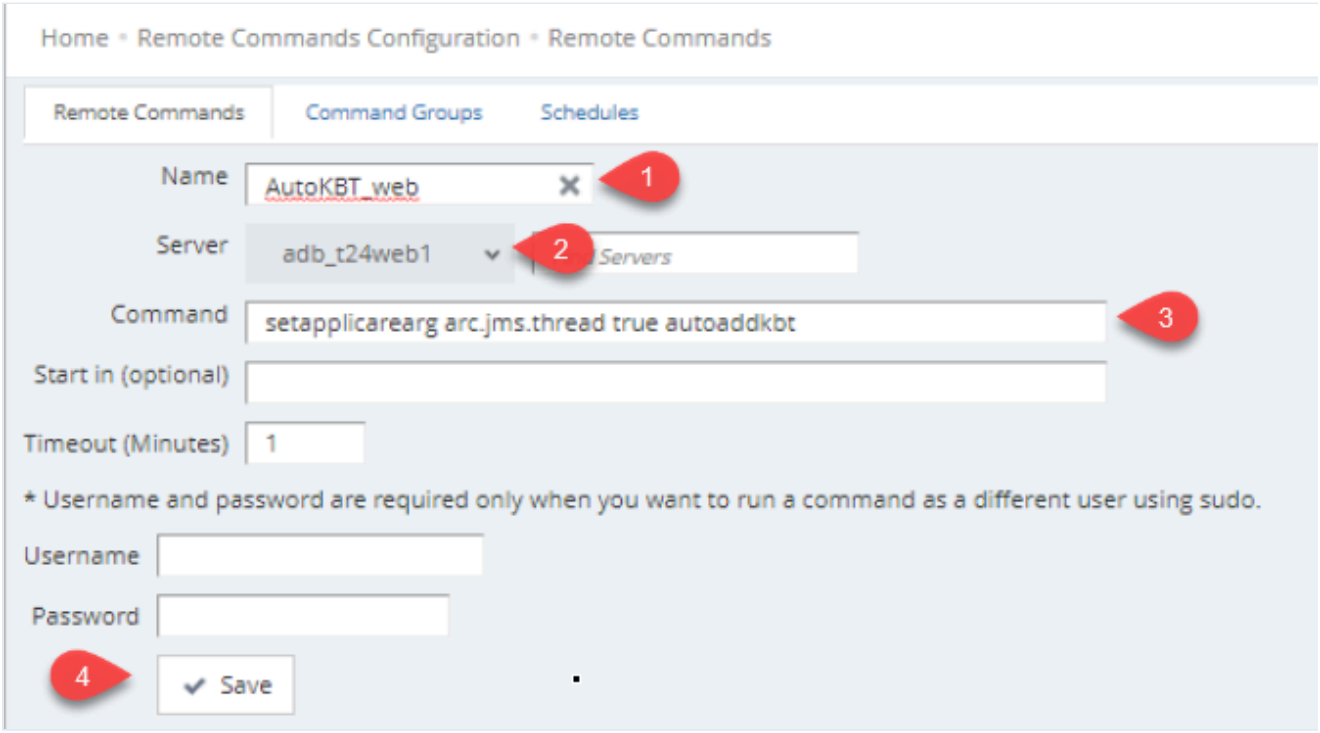
Automatically add KBT configuration

Applicare 9.2 · Updated September 9, 2024 · **Pending technical review**

Step 1: Log in to Applicare console, navigate to the **Remote Commands** screen, and then click the **setting** button.

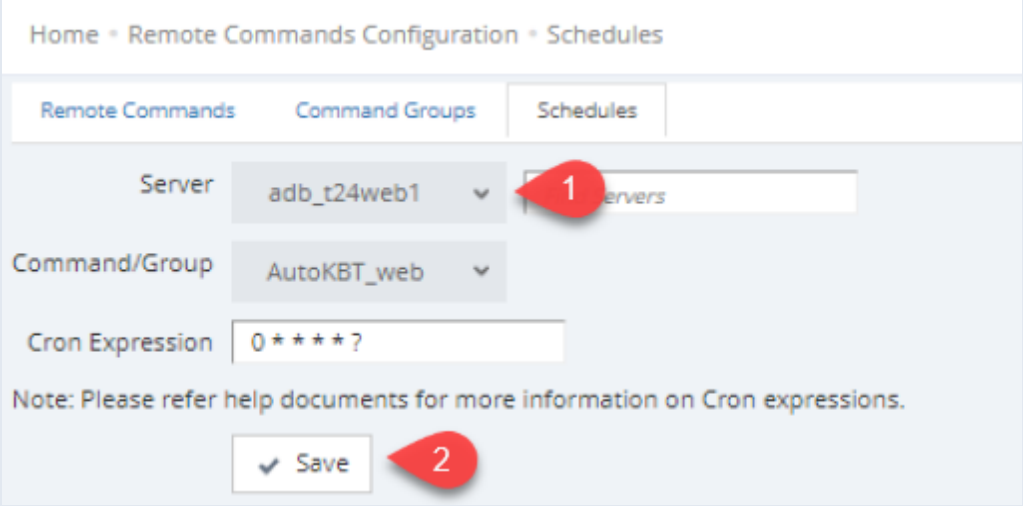
Step 2: Enter a name in the **Name** field, select a server with either **sender or receiver** application from the **Server dropdown**, paste the command into the **command field** and then click **save** button.

```text
setapplicarearg arc.jms.thread true autoaddkbt
```



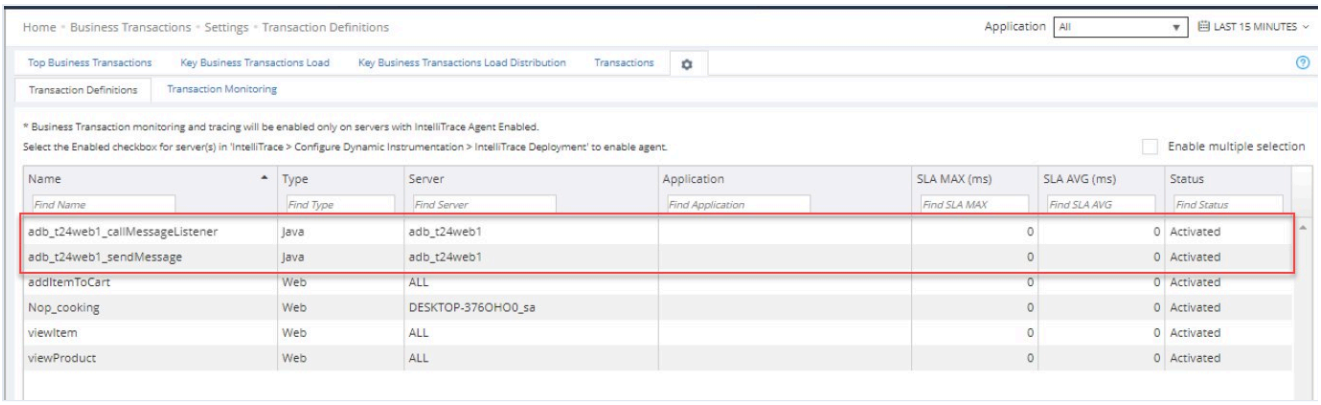
Step 3: On the same screen, click on the third tab **schedules**.

Step 4: Choose the server configured for the remote command from the **server dropdown**, then click the **save** button.



Step 5: After 2-3 minutes, navigate to **Business Transactions-> Settings-> Transaction** Definitions screen to verify that the KBT has been created.

KBT creation only occurs when a JMS transaction is executed.



If you prefer not to auto add KBT and only view KBT trace details in remote commands, remove the "**autoaddkbt**" keyword from the command.

```text
setapplicarearg arc.jms.thread true
```

USING APPLICARE › TRANSACTIONS

Download IntelliTrace report

Applicare 9.2 · Updated January 8, 2025 · **Pending technical review**

IntelliTrace report for Http:

Step 1: Go to the report screen in the left-hand menu.

Step 2: Below is a list of which options to select from which dropdown.

a.) **Select Report:** Select the "**Intellitrace report"** option in this dropdown.

b.) **Data:** Select the "**http"** option in this dropdown.

c.) **Order by:** Select the "**MaxAvg.Time"** option in this dropdown.

d.) **Server:** In this dropdown, select the "**server"** for which you want intellitrace data.

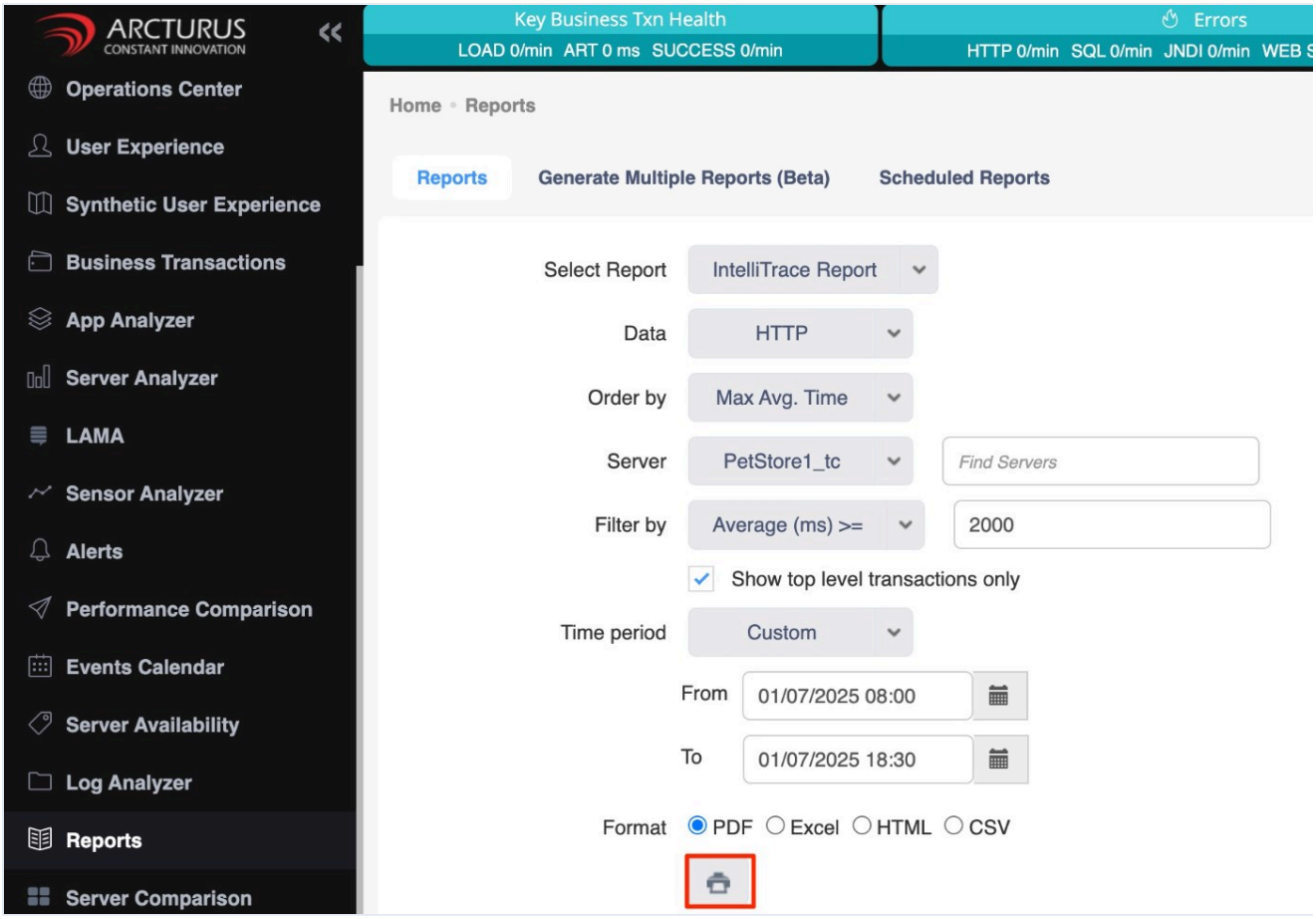
e.) **Filter by:** In this dropdown, select the "**Average>="** option and enter "**2000"** in the filter value text box.

f.) Enable the Show top level transactions only checkbox.

g.) **Time period:** Select "**Custom**" from this dropdown menu, then specify the desired data time period using the "**From**" and "**To**" fields.

h.) **Format:** Choose your preferred output format (**PDF, Excel, HTML,** or **CSV**).

Step 3: Enter the values as shown above, then click the **print icon**.



IntelliTrace report for SQL:

Step 1: Go to the report screen in the left-hand menu.

Step 2: Below is a list of which options to select from which dropdown.

a.) **Select Report:** Select the "**Intellitrace report"** option in this dropdown.

b.) **Data:** Select the "**SQL"** option in this dropdown.

c.) **Order by:** Select the "**MaxAvg.Time"** option in this dropdown.

d.) **Server:** In this dropdown, select the "**server"** for which you want intellitrace data.

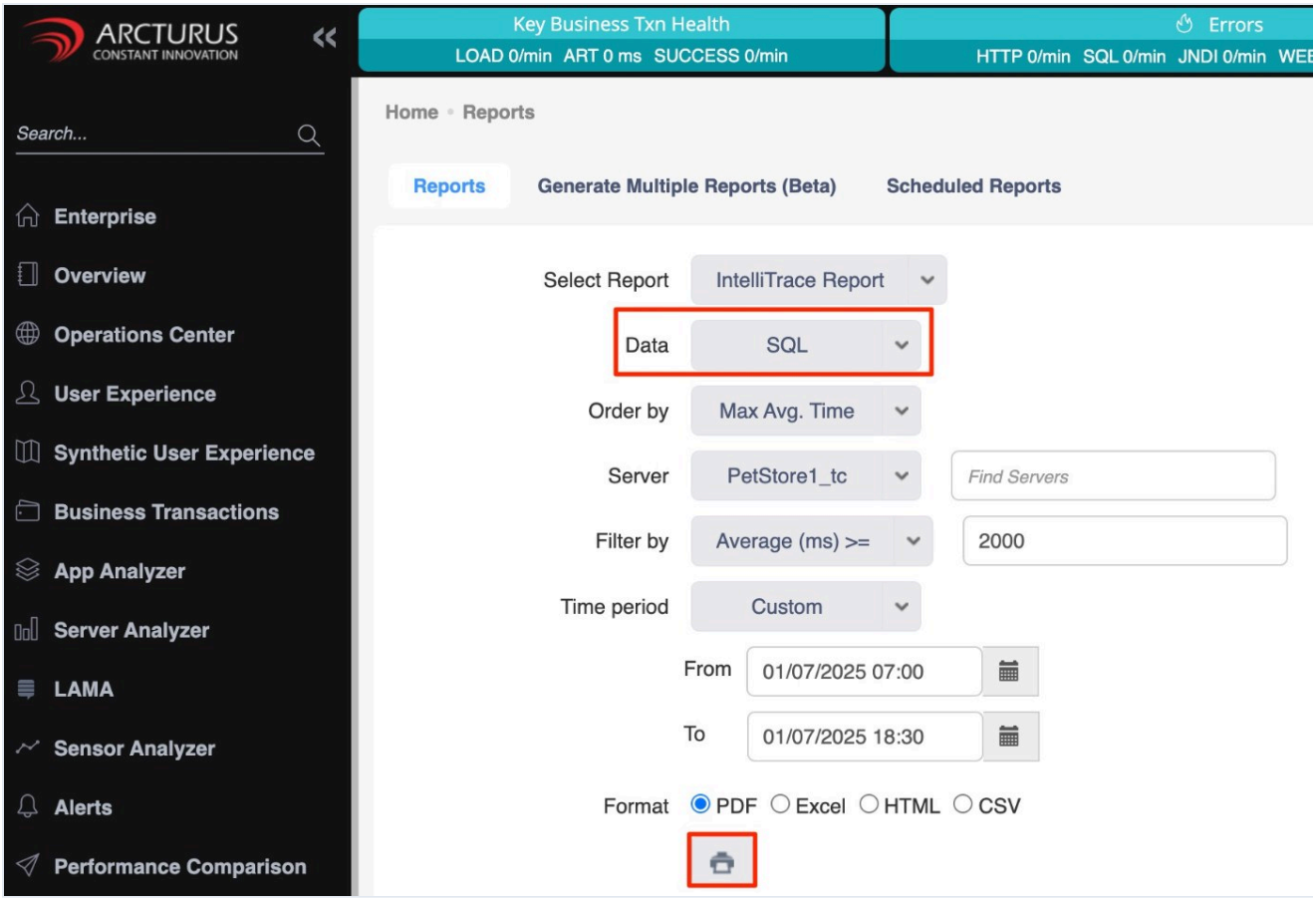
e.) **Filter by:** In this dropdown, select the "**Average>="** option and enter "**2000"** in the filter value text box.

f.) Enable the Show top level transactions only checkbox.

g.) **Time period:** Select "**Custom**" from this dropdown menu, then specify the desired data time period using the "**From**" and "**To**" fields.

h.) **Format:** Choose your preferred output format (**PDF, Excel, HTML,** or **CSV**).

Step 3: Enter the values as shown above, then click the **print icon**.


ARCTURUS
CONSTANT INNOVATION
Search...
Enterprise
Overview
Operations Center
User Experience
Synthetic User Experience
Business Transactions
App Analyzer
Server Analyzer
LAMA
Sensor Analyzer
Alerts
Performance Comparison
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min
Home • Reports
Reports
Generate Multiple Reports (Beta)
Scheduled Reports
Select Report
IntelliTrace Report
Data
SQL
Order by
Max Avg. Time
Server
PetStore1_tc
Find Servers
Filter by
Average (ms) >=
2000
Time period
Custom
From
01/07/2025 07:00
To
01/07/2025 18:30
Format
PDF
Excel
HTML
CSV

USING APPLICARE › TRANSACTIONS

Configuration for KBT with parameter

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

If you need to capture only a specific URL in KBT, then follow these steps:

Step 1: Navigate to **Business Transactions-> Settings-> Transaction Definitions** in Applicare, and then click the **New** button at the bottom right.

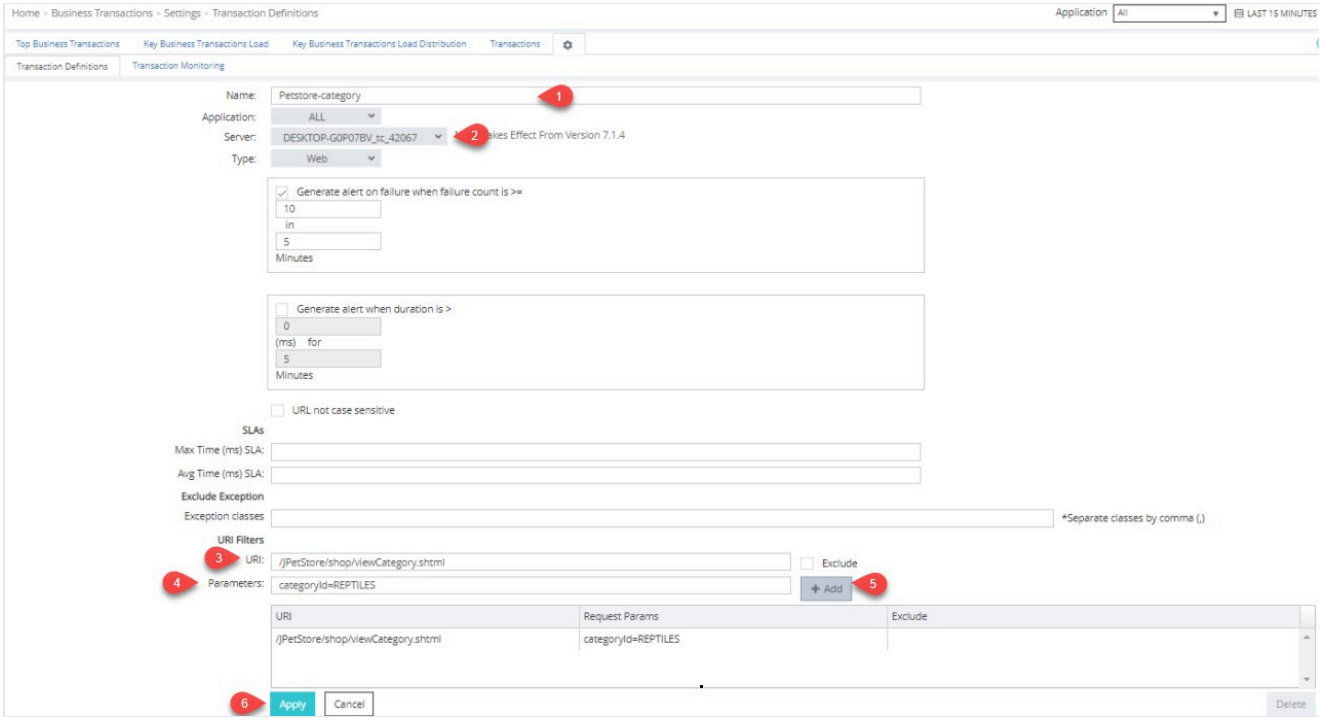
Step 2: Enter the **KBT** name in the Name text box, then click the **server dropdown** and select the appropriate **server**.

Step 3: Enter the application URL in the **URL** text box, then input the parameter you want to add in the **parameter** text box. Click on **Add** button, and then click **Apply**.

Example:

URL = /JPetStore/shop/viewCategory.shtml

Parameter = categoryId=REPTILES



If you need to fetch a single parameter with different values for a specific URL in KBT, follow these steps:

Step 1: Navigate to **Business Transactions-> Settings-> Transaction Definitions** screen in Applicare and click the **New** button at the bottom right.

Step 2: Enter the **KBT** name in the **Name** text box, then click the **server dropdown** and select the desired **server**.

Step 3: Enter the **application URL** in the URL text box, but this time parameter with a different value. After adding the parameter, click the **Apply** button.

Step 4: Repeat the steps in **step 3,** to add another parameter, then click the **Apply** button.

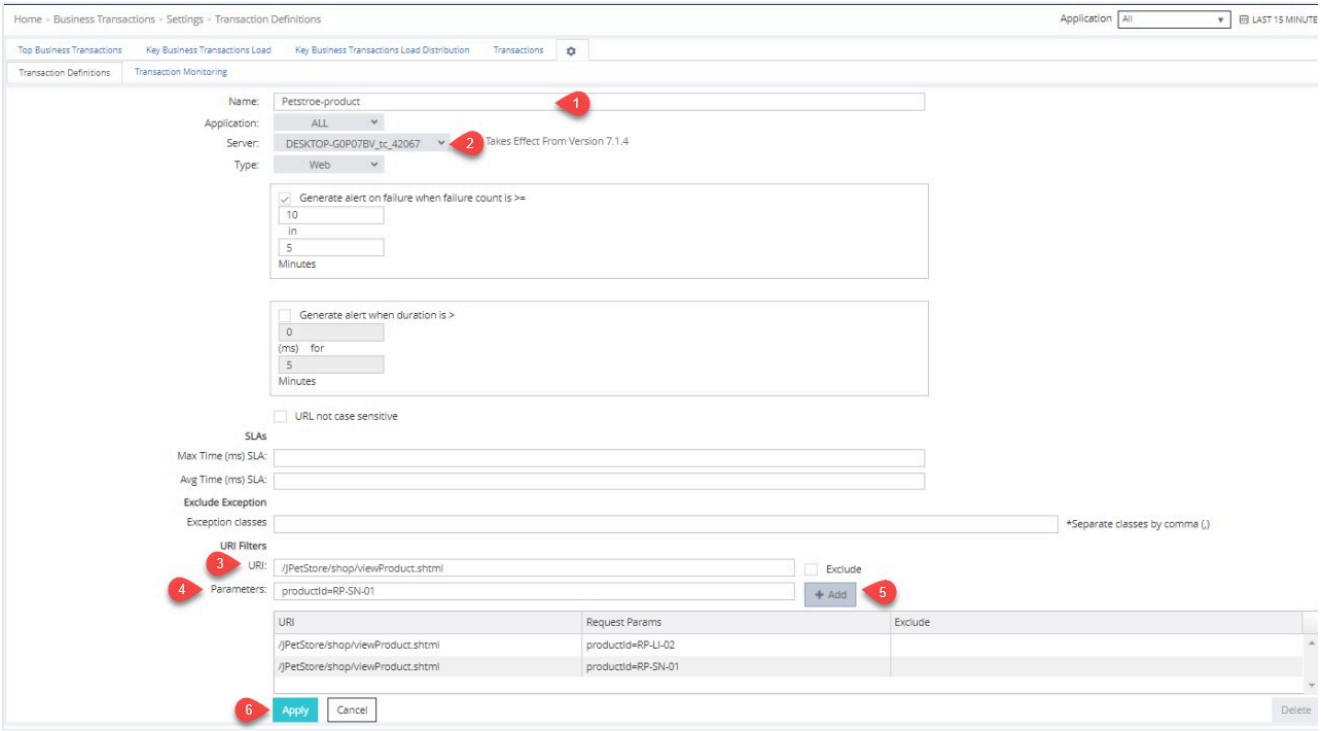
e.g.

URL = /JPetStore/shop/viewProduct.shtml

Parameter = productId=RP-LI-02

URL = /JPetStore/shop/viewProduct.shtml

Parameter = productId=RP-SN-01



If you need to capture only different parameters for a specific URL in KBT. Follow these steps:

Step 1: Navigate to **Business Transactions-> Settings-> Transaction Definitions** in Applicare, and click the **New** button at the bottom right.

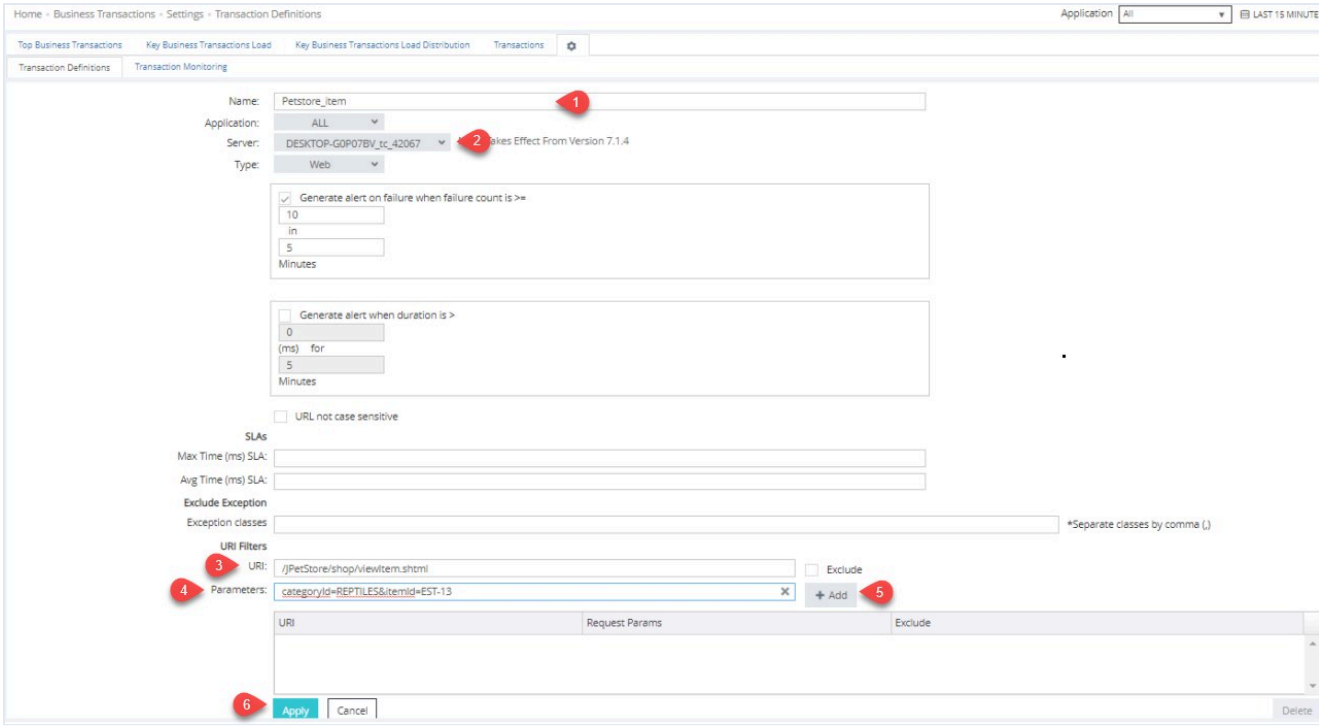
Step 2: Enter the **KBT** name in the **Name** text box, then click the **server dropdown** and select the appropriate **server**.

Step 3: Enter the application URL in the **URL** text box, then input the different parameter you want to add in the **Parameter** text box, click the **Add** button.

e.g.

URL = /JPetStore/shop/viewItem.shtml

Parameter = categoryId=REPTILES&itemId=EST-13



If want to capture all request from a specific parameter in KBT, follow these steps:

Step 1: Navigate to **Business Transactions-> Settings-> Transaction Definitions** in Applicare and click the **New** button at the bottom right.

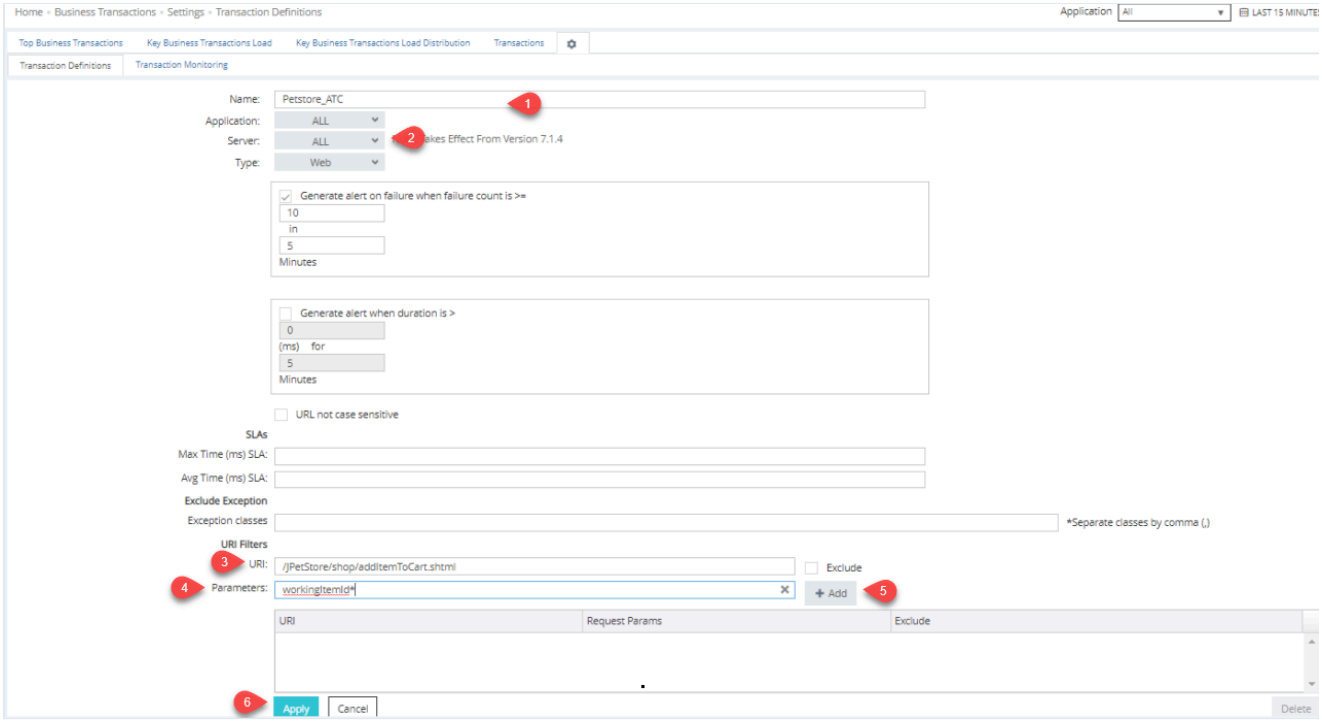
Step 2: Enter the **KBT** name in the **Name** text box, then click the **server dropdown** and select the desired **server**.

Step 3: Enter the application URL in the **URL** text box, then input the **parameter** you want to include in the Parameter **text box,** click the **Add** button.

e.g.

URL = /JPetStore/shop/addItemToCart.shtml

Parameter =workingItemId*



Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

Server Analyzer

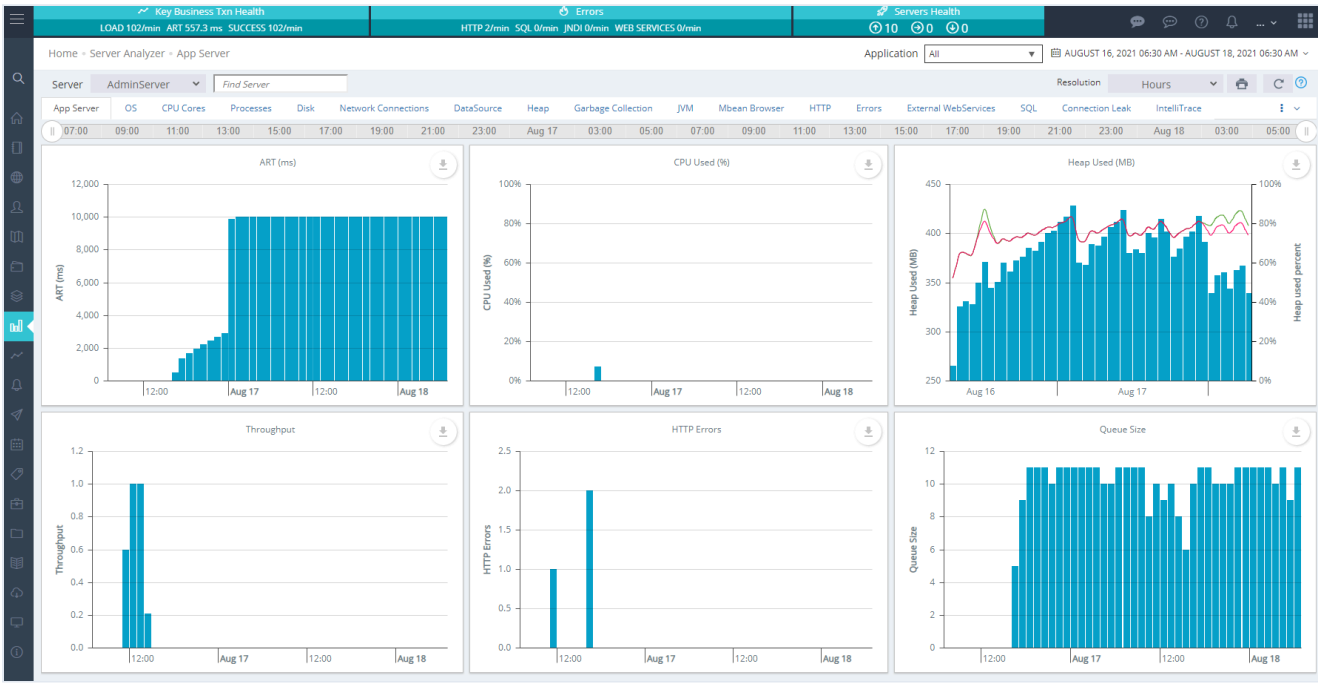
Applicare 9.2 · Updated April 5, 2025 · **Pending technical review**

Overview

The Server Analyzer gives you a quick look at your server's health and activity. Select a time frame at the top to view historical data. The server list will show only those related to chosen application. To zoom in, click and drag over the area you want to explore.

App Server Analyzer

The AppServer tab offers a quick overview of application server performance. It helps identify potential bottleneck, such as queue buildup, which indicates the server is struggling to process incoming request. Issues like garbage collection or high CPU can also be mentioned here. The AppServer Analyzer provides a correlated view to pin point the effects of change in performance metrics.



If a time period is selected on top of the home dash then history view is visible. Average Response Time (ART) is a great indicator of a server's performance. A significant increase in ART means things are getting stuck somewhere in the app server. Similarly heap used, throughput, queue size, http errors and CPU used are available.

Throughput: Server throughput is the number of requests served per second by the server. **Busy Threads**: The number of Server threads thats are busy serving requests at any given time.

OS Analyzer

The OS Internals Analyzer provides a fast and convenient window into your physical server's/VM's performance including # of processes, tcp connections, CPU, memory and storage performance.

Processes: Number of processes running on the machine. Running too many processes may cause memory contention (paging), I/O contention or excessive context switching. This contention can reduce system throughput if parallel execution were not used.

CPU %- Cpu usage in percent. High CPU usage is not a significant cause for concern as long as it doesn't occur over the long-term. But if the cpu usage remains high for long term then they system may freeze or will respond too slow.

Process Queue Length- The Processor Queue Length is the number of threads that are ready but currently unable to run on the processor due to another active thread. This count shows ready threads only, not threads that are running. A bottleneck on the processor may be thought to occur where the number of threads in the queue is more than 2 times the number of processor cores over a continuous period.

Memory Free %- Percentage of free memory available in physical memory. Memory can have a huge impact on a system is performance. As physical free space increases, the machineï¿½s dependency on virtual memory decreases. The less a computer has to depend on virtual memory, the more efficiently that machine will run.

Memory Free- Free memory available (in Mega Bytes) in physical memory.

Page File Usage %- Displays the percentage of the paging file that is currently in use.

Disk Transfers/sec- A high Disk Transfers/sec value may occur due to a burst of disk transfer requests by either an operating system or application.

Context Switches- This attribute reports the combined rate at which all processors on the computer are switched from one thread to another. High context switches, shows that threads have less time to do their work and the system performance might go down.

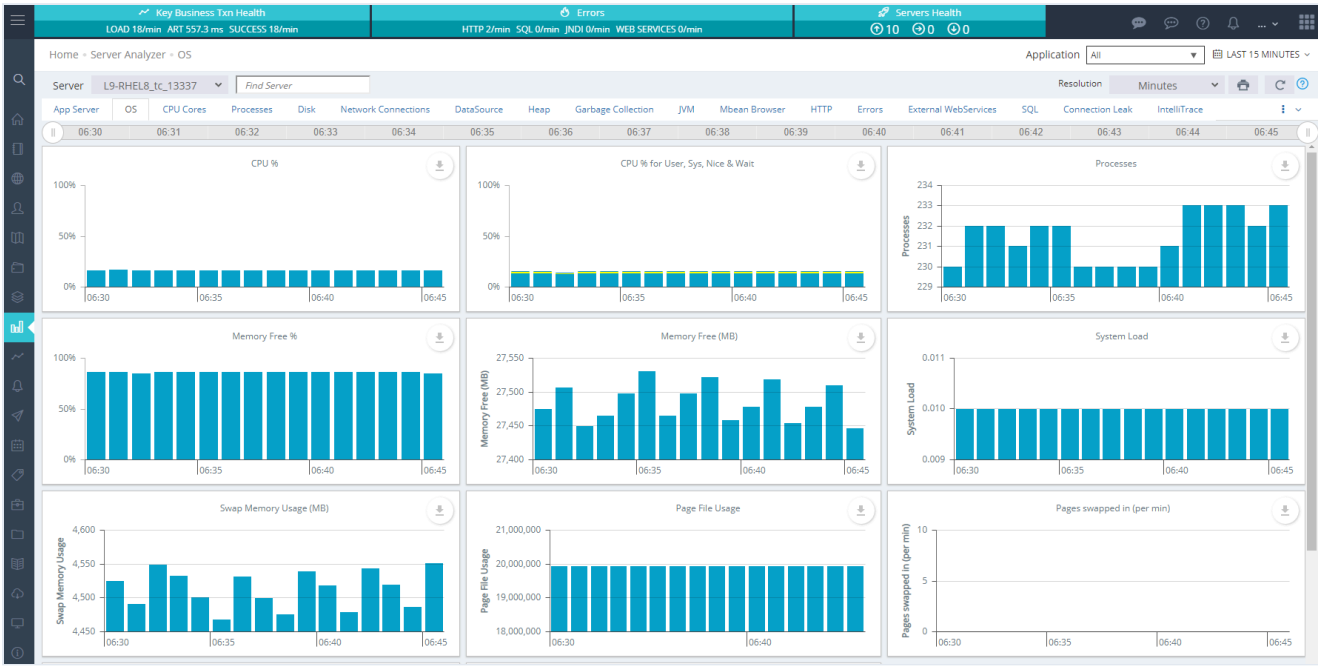
Swap memory usage- Swap usage refers to the amount of virtual memory that is currently being used to temporarily store inactive pages from the main physical memory. Swap space is your safety net for when you run out of RAM. While swapping, processes may get slower, data is still nevertheless processed. However, when you run out of virtual memory, processes are queued and stalled until some memory is freed up.

Pages Swapped In- Number of pages read(in last one minute) from paging/swap file(s) to resolve hard page faults. (Hard page faults occur when a process requires code or data that is not in physical memory, and must be retrieved from disk.)

Pages Swapped Out- Number of pages written(in last one minute) to paging/swap file(s) to free up space in physical memory. Pages are written back to disk only if they are changed in physical memory, so they are likely to hold data, not code. A high rate of pages output might indicate a memory shortage. The operating system writes more pages back to disk to free up space when physical memory is in short supply. Even when swap space is not particularly full, continuously high swap-out rates indicate that the pages that are swapped are actively processed. This means that your system requires more fast physical memory for your workloads, and swap is not helping.

TCP Connections- Total TCP connections count.

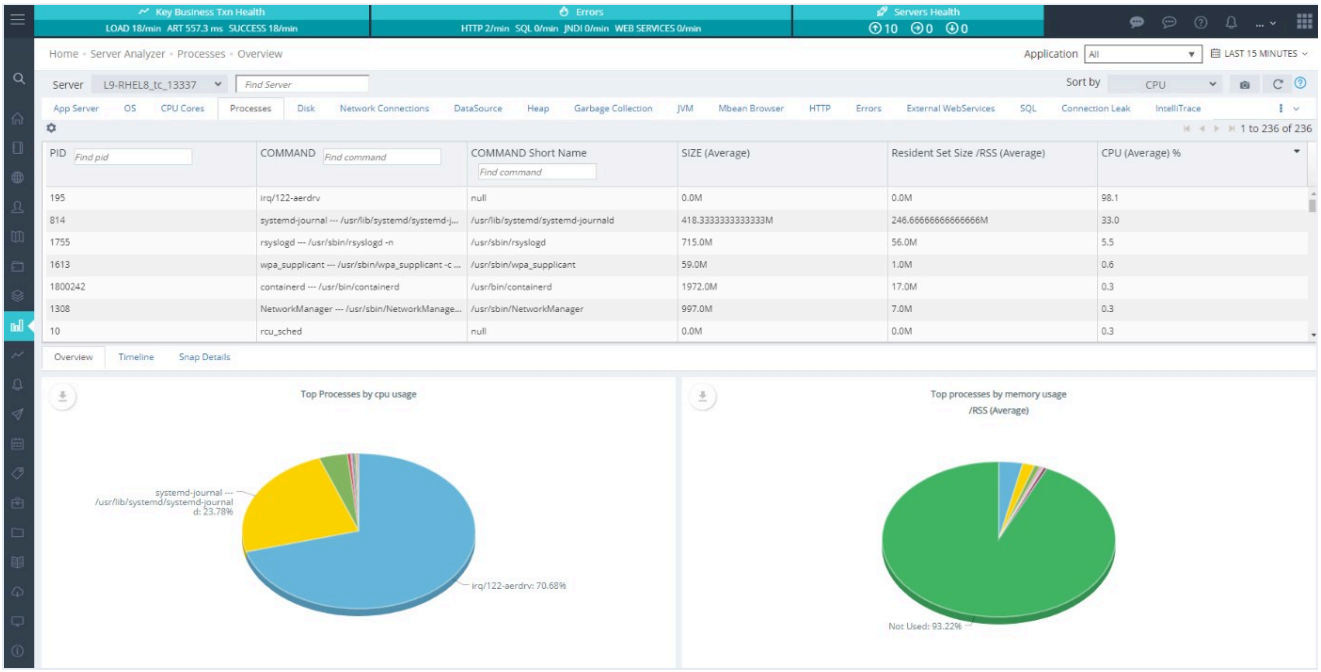
Here, you can access the OS performance history for the selected time period.



Process Analyzer

This view displays the performance stats for CPU, memory and running processes organized in somewhat similar manner to the output of **Top** utility on Unix based systems.

snapshots of various processes running at any point of time and what kind of resources they were consuming at that point of time is available here. Applicare by default would take a snapshot of this info every 30 minutes but this can be configured by changing the snaps interval.



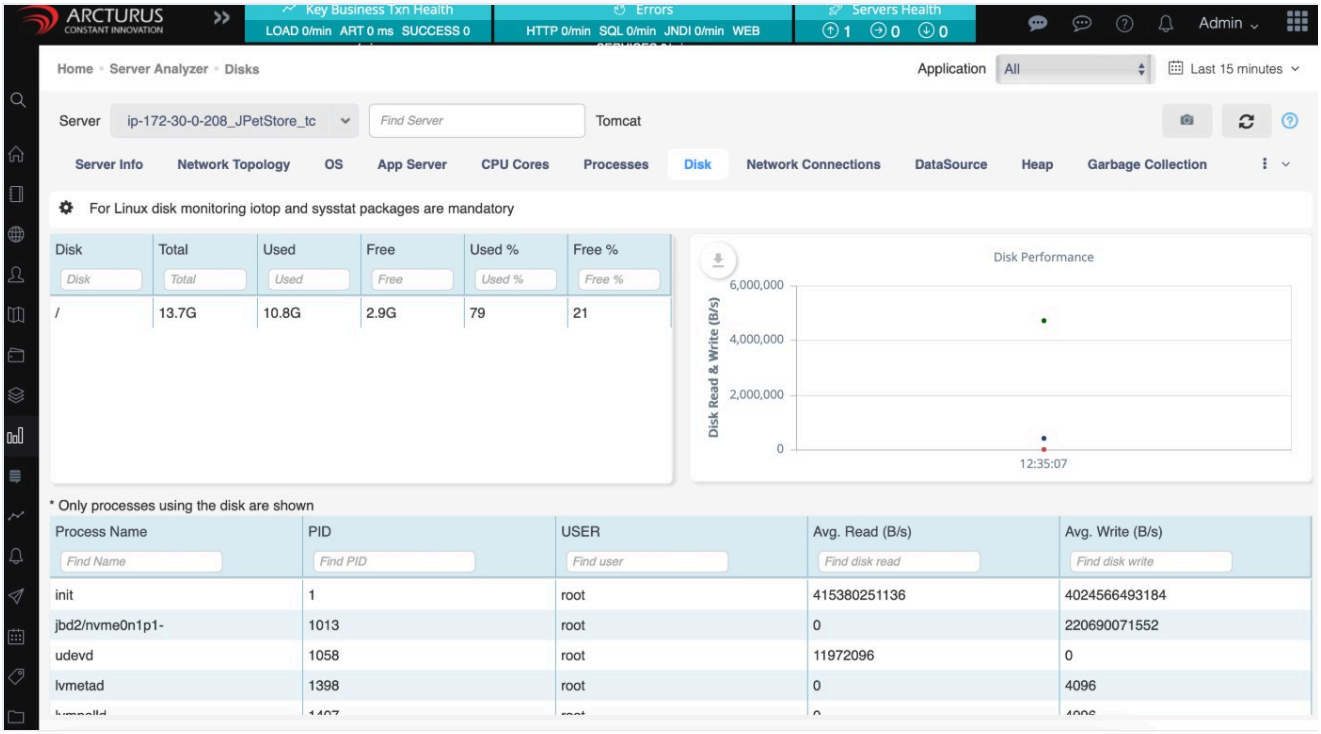
Column Descriptions:

PID :	**The process ID of each task.**
USER :	The user name of the task's owner.

PID :	**The process ID of each task.**
STIME :	Time the process was started.
SIZE :	The size of the task's code plus data plus stack space, in megabytes, is shown here.
RSS :	The total amount of physical memory used by the task, in megabytes, is shown here.
SHARE :	The amount of shared memory used by the task is shown in this column.
STATE :	The state of the task is shown here. The state is either S for sleeping, D for uninterruptible sleep, R for running, Z for zombies, or T for stopped or traced.
TIME :	Total CPU time the task has used since it started in MM:SS format.
CPU :	The task's current usage of CPU time as a percentage.

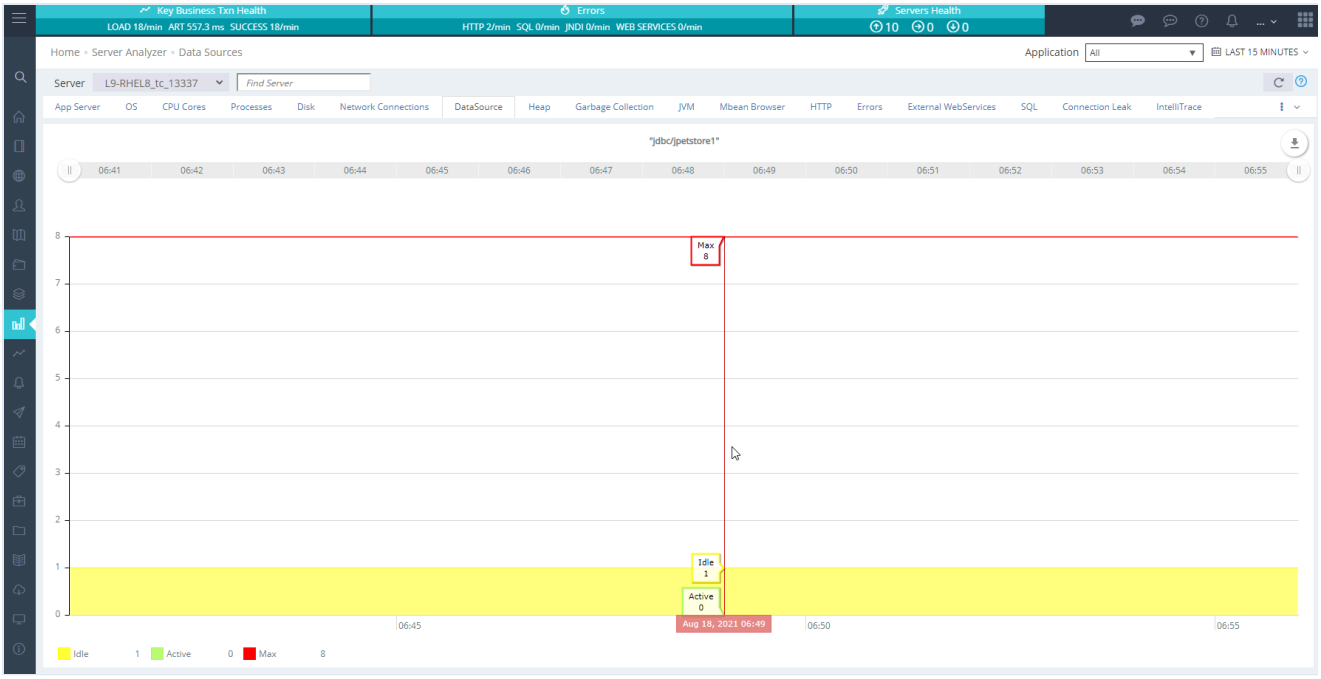
DISK

On the left side, view the used and available disk space. On the right side, monitor disk performance, with only disk-using processes shown.



Data Source Analyzer

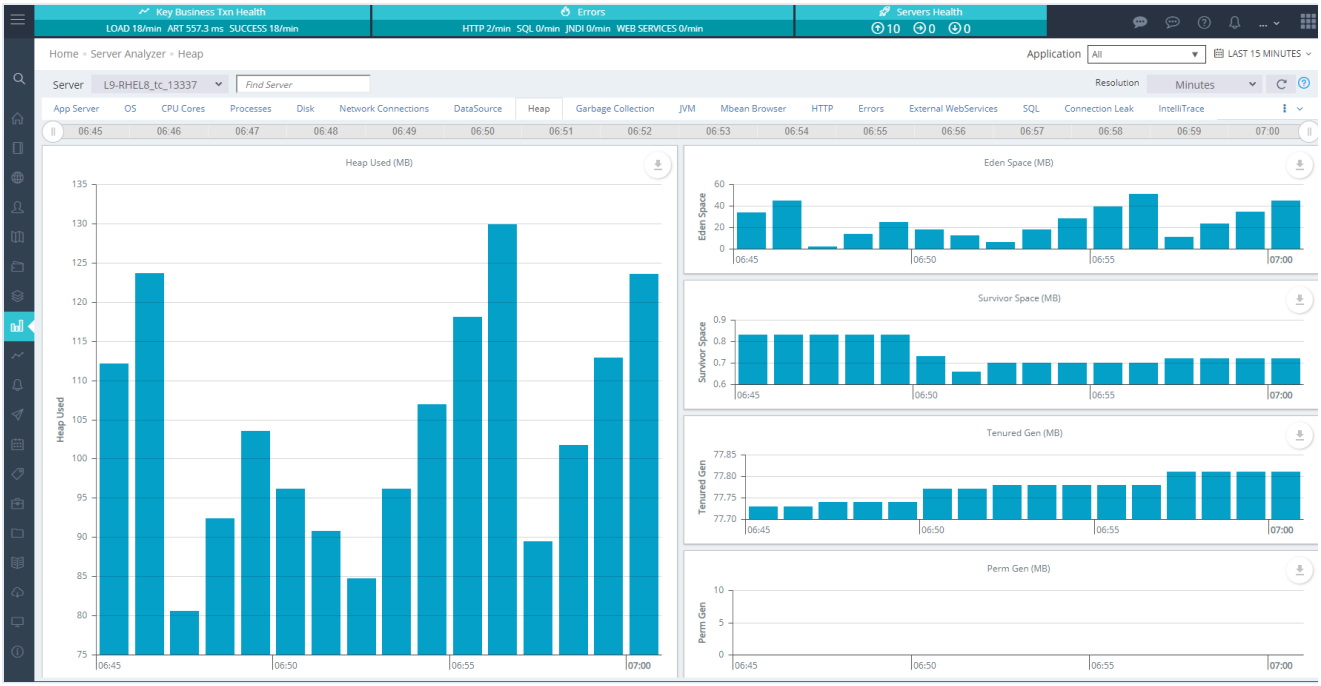
The DataSource Analyzer provides a fast and convenient window into various data source's usage. Datasource utilization is very useful in finding out issues related to connections running out and tuning of connection pools.



Heap Analyzer

The Heap Analyzer provides a convenient window into the history of your server's garbage collection and memory allocation behavior. You can zoom in and out, choose the range of days from the drop down, and select which server to display.

You can easily spot slow memory leaks with the Heap Analyzer. Zoom out to show the data for as long as the server has been up, and you can see how each generation is growing over time. If after major garbage collections the heap doesn't settle to prior levels, but instead keeps increasing, you may have a memory leak.



Heap Dump

You can generate a heap dump of the selected server by clicking on Heap Dump button. Please note this will make the server unresponsive until the heap dump is written to a file on the server. Generating heap dumps on demand is availble on

- Sun JVM 6 and above. - Oracle JRockit R28 and above.

Also you can enable automatic heap dumps on supported JVMs when available free memory reaches a certain threshold by setting 2 startup arguments.

e.g. Using startup arguments *-Dapplicare.dumpheap.perc=10 -Dapplicare.dumpheap.maxcount=2* will generate heap dumps when the server's available free heap size reaches 10% of the max heap size and will create maximum of 2 heap dumps during the lifetime of the JVM.

You can use the generated heap dumps to detect memory leaks and analyze the objects and their relationships by loading the heap dump in Arcturus Memory Analyzer.

Garbage Collection Analyzer

The GC Analyzer provides a fast and convenient window into your server's garbage collector behavior. This gives you the ability to quickly spot garbage collection issues and tweak GC parameters to improve performance and reduce latency of your applications.

GC Analyzer is available on Oracle Sun JVMs, JRockit and IBM JVMs. It requires verbose GC enabled on the JVM.

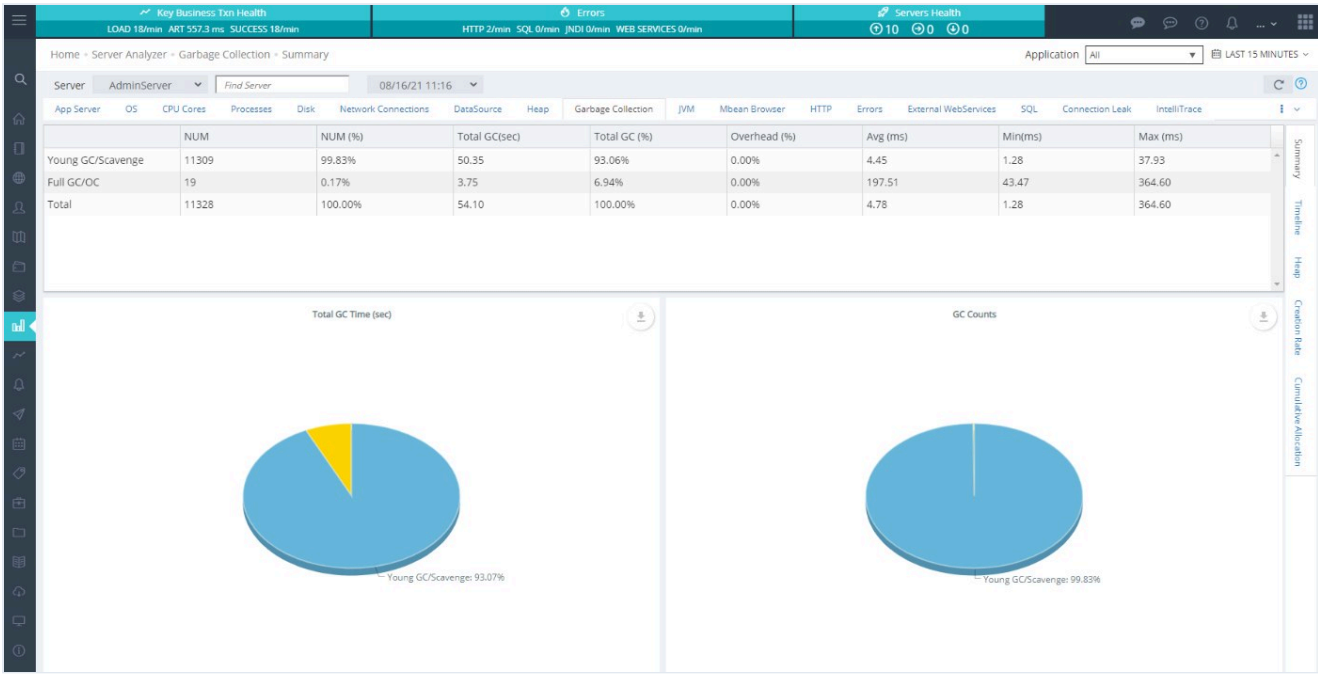
e.g. following arguments are automatically added by Applicare to enable verbose GC on Oracle Sun JVM.

-verbosegc -Xloggc:%SERVER_NAME%.gc

Also the use of **-XX:+PrintTenuringDistribution** flag is not supported at the moment and the use of it will prevent Applicare from reading the GC log file.

GC Summary

The summary tab gives you an overview of Full and Young GC cycles that occured over the lifetime of JVM. An important value you should take note of is the Overhead (%) in the table.

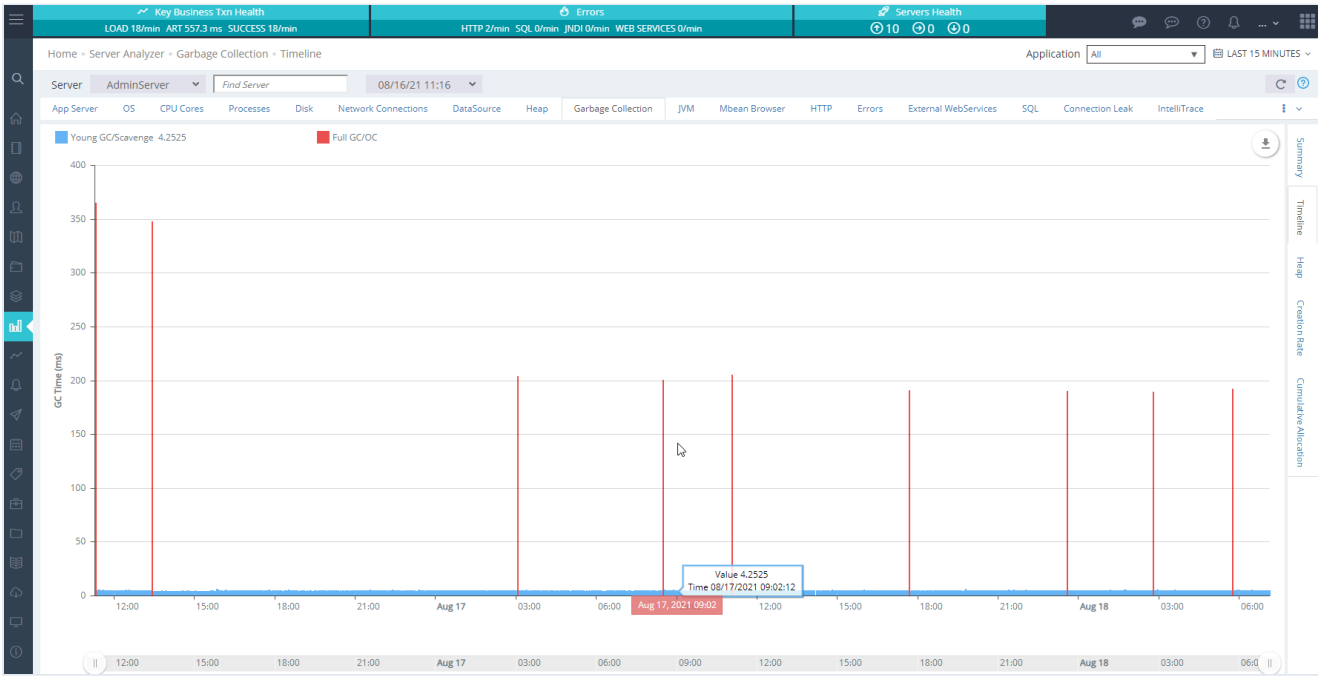


Detailed Charts

The GC Analyzer feature charts that visually represent various aspects of Garbage Collection cycles.

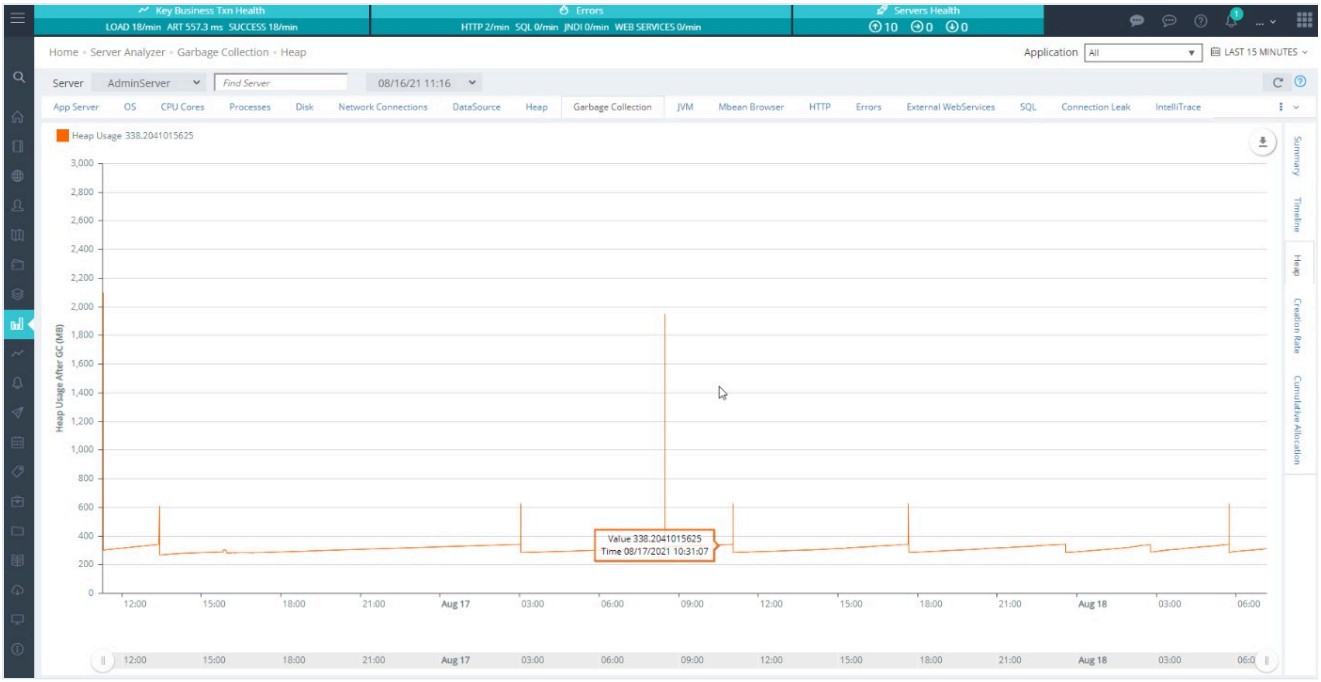
GC Timeline

Garbage Collection shows each GC cycle over time and provides a clear picture of the type of GC cycle and time taken by each.



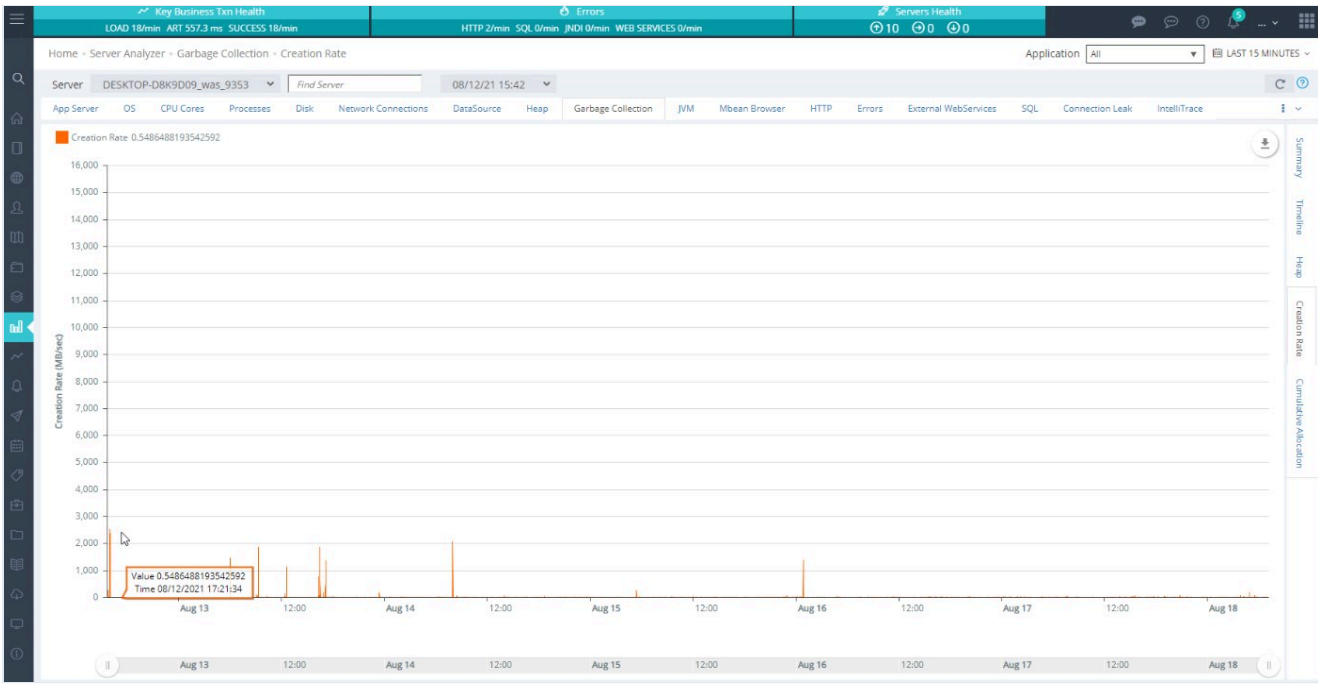
Heap

This chart plots the Heap size before and at the end of each GC cycle over time.



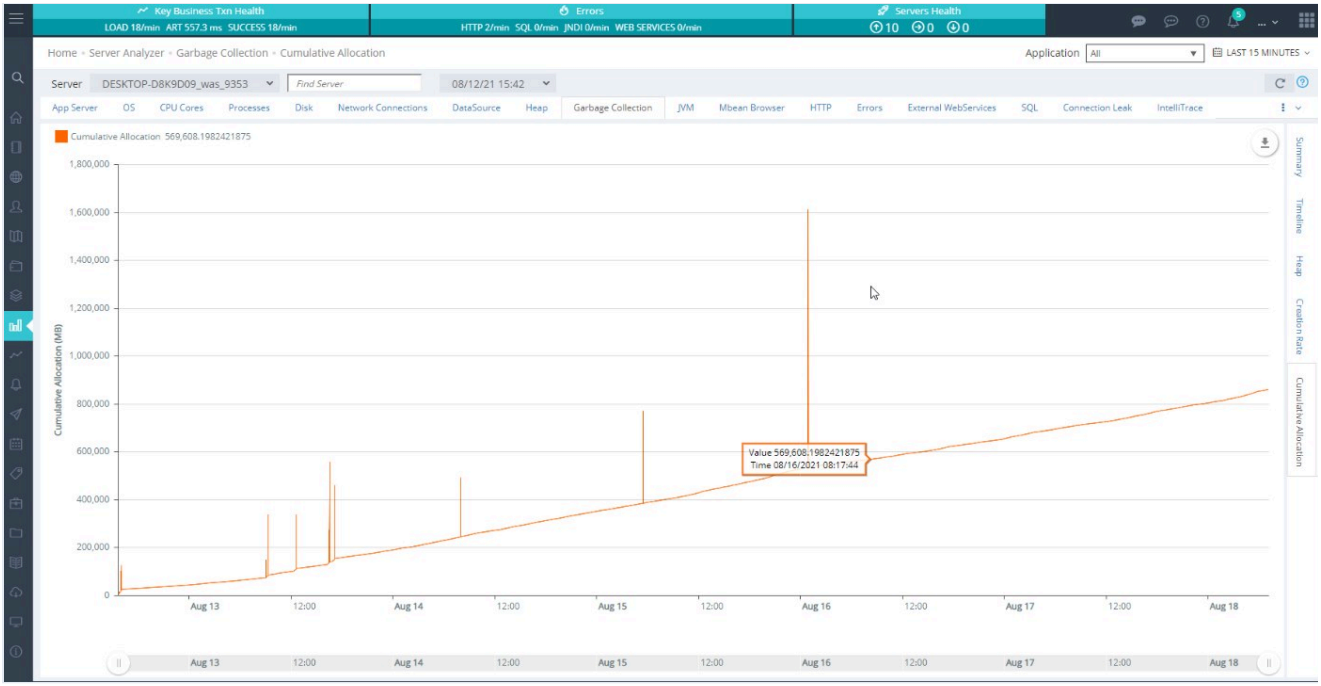
Creation Rate

This analyzer shows the total size of objects created between each GC cycle.



Cumulative Allocation

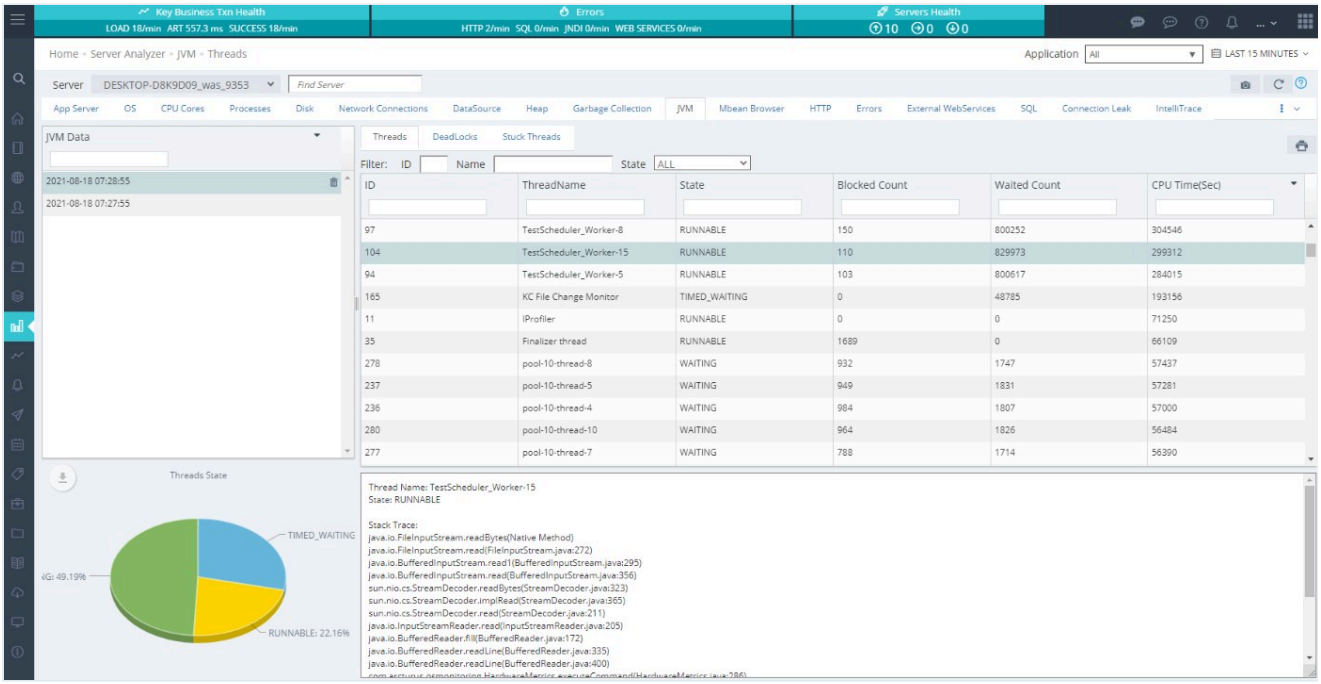
This analyzer shows total size of objects created over time.



JVM Analyzer

The JVM Analyzer provides a very detailed view into a server's thread behavior, detects Deadlocks, detects stuck threads, and provides an interface for viewing stack traces.

This view allows you to save a thread dump of a running server and analyze the saved dumps at a later time. The saved snap shots contain all thread information displayed on the real time view and any deadlock or stuck threads detected at the time of creating the thread dump.



Thread Analysis

Deadlock Detection

A ***Deadlock*** is a situation where two or more threads are blocked forever, waiting on each other to finish. You can find a deadlock by taking a thread dump to display the current threads and look for the BLOCKED state, or you can use the **JVM Analyzer** and click on the **Deadlocks tab.** This tab shows all available information about the deadlock situation. When a deadlock occurs, the tab will turn red to alert you that a deadlock exists. Then you can capture the data and send it to development so they can analyze the code.

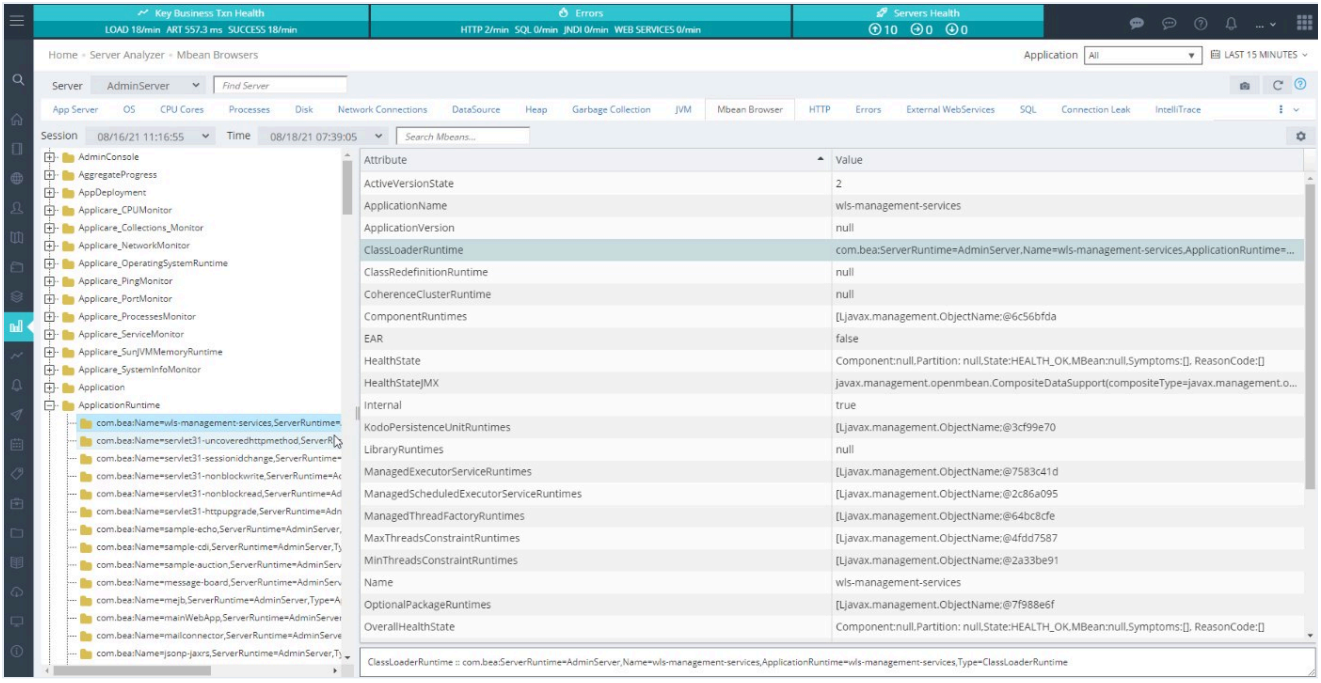
good course of action to follow when you detect a deadlock is to assign a portion of your team to concentrate on finding out how to reproduce the deadlock, and the other portion to look at the stack traces to figure out why the code is deadlocking

Stuck Thread Detection

A ***stuck thread*** is a thread that is blocked and can't return to the thread pool within a defined period of time. When an application thread is blocked, it can't quickly finish its job and be reused. In most situations, the cause of stuck threads is also the cause of poor system performance. This is because the stuck thread interferes with regular task execution. The “Stuck Threads” tab will turn red and display the diagnostic information to help you identify the root cause of the stuck thread. Stuck Threads are often caused by the code opening a network connection without specifying a network connect or read timeout. When a timeout isn’t configured for each method call that involves networking, it will potentially create a block because it will wait infinitely for a response. The fallback timeout in this case comes from the OS which is by default set very long (e.g. 5-10 minutes). Especially in the case of SOA and Web Services where there are many network calls, timeouts need to be defined properly.

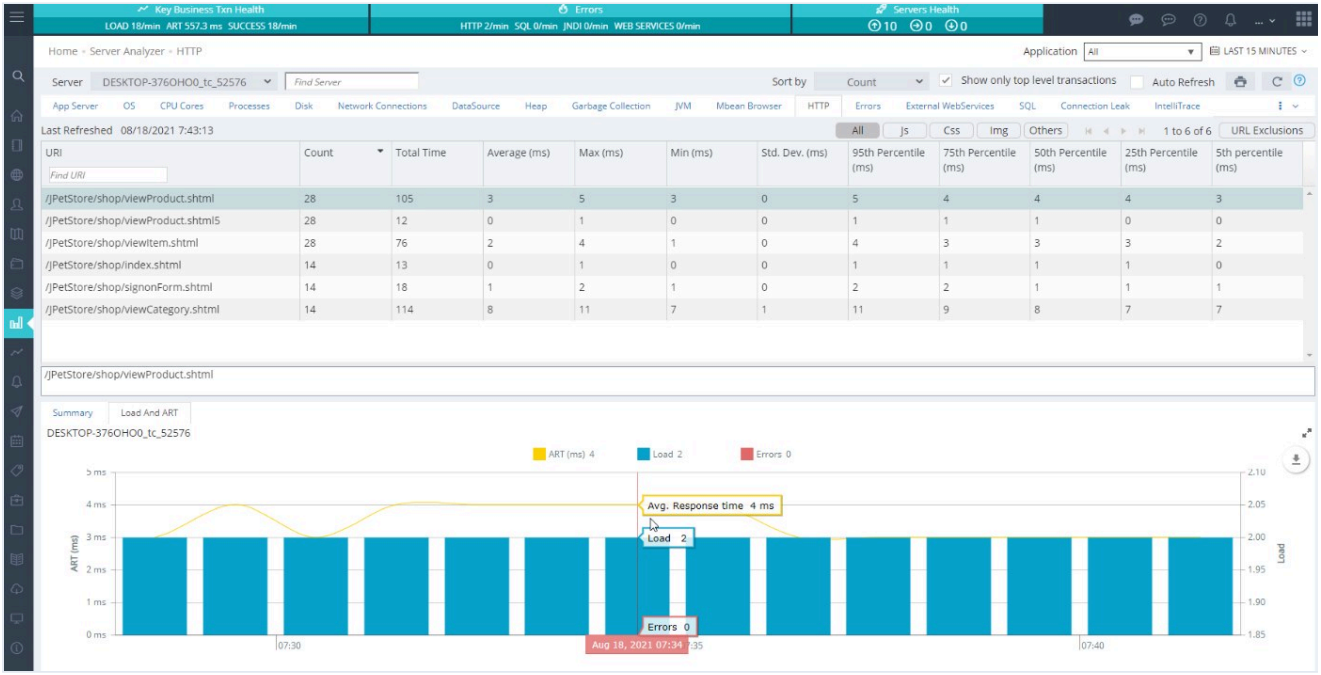
MBean Browser

The MBean Browser enables you to explore MBean properties and their values.



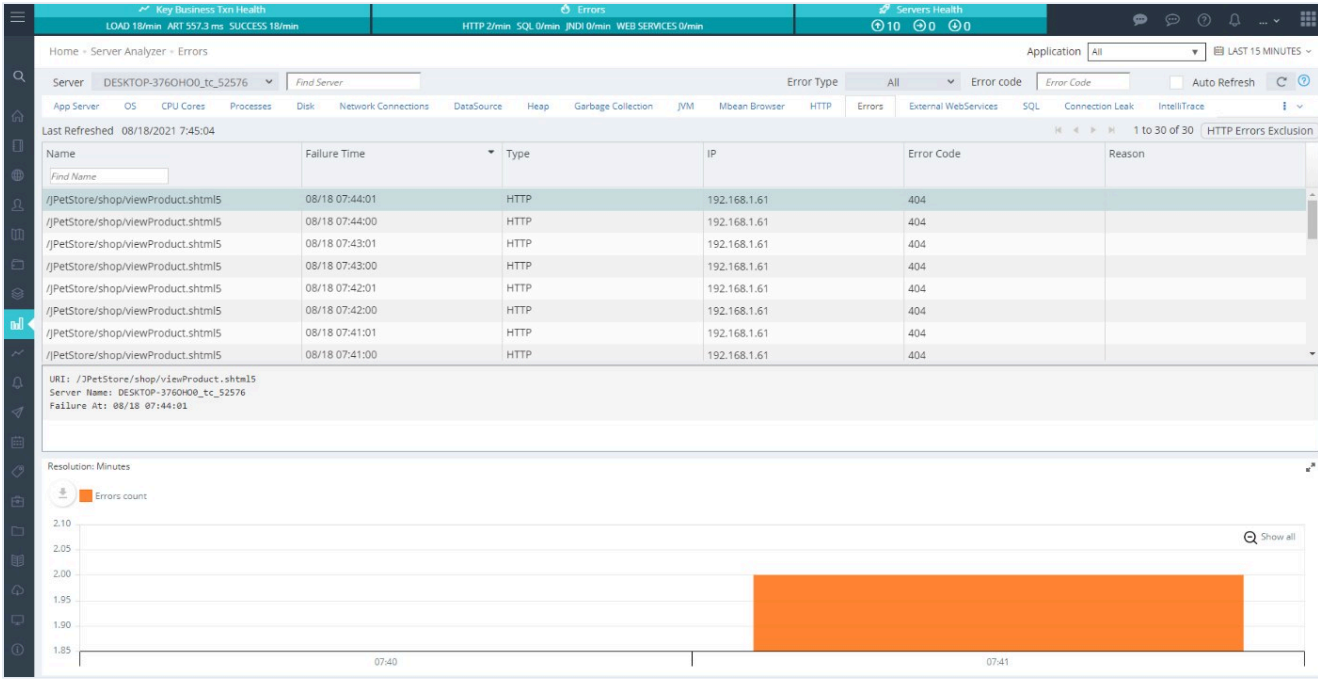
HTTP

The HTTP Analyzer offers performance statistics for HTTP requests on the selected server. The real-time view displays HTTP stats since the JVM started, while the historical view provides data for the selected time period. The real-time view also indicates whether HTTP Profiling is enabled and, if so, how long it has been active.



HTTP Errors

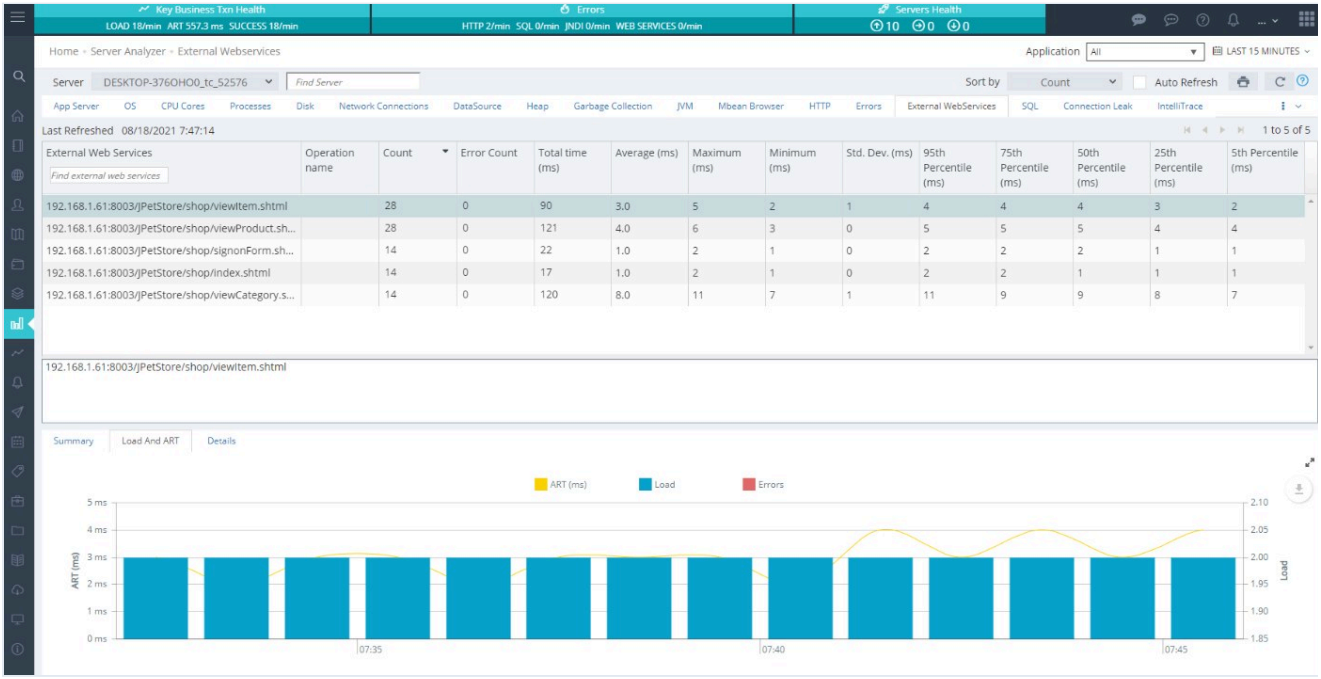
HTTP Errors shows URI, failure time, http error code, exception and IP address where the call came from.



External WebServices

Server Analyzer External WebServices shows all the web services that are called from the monitored JVM/App Server.

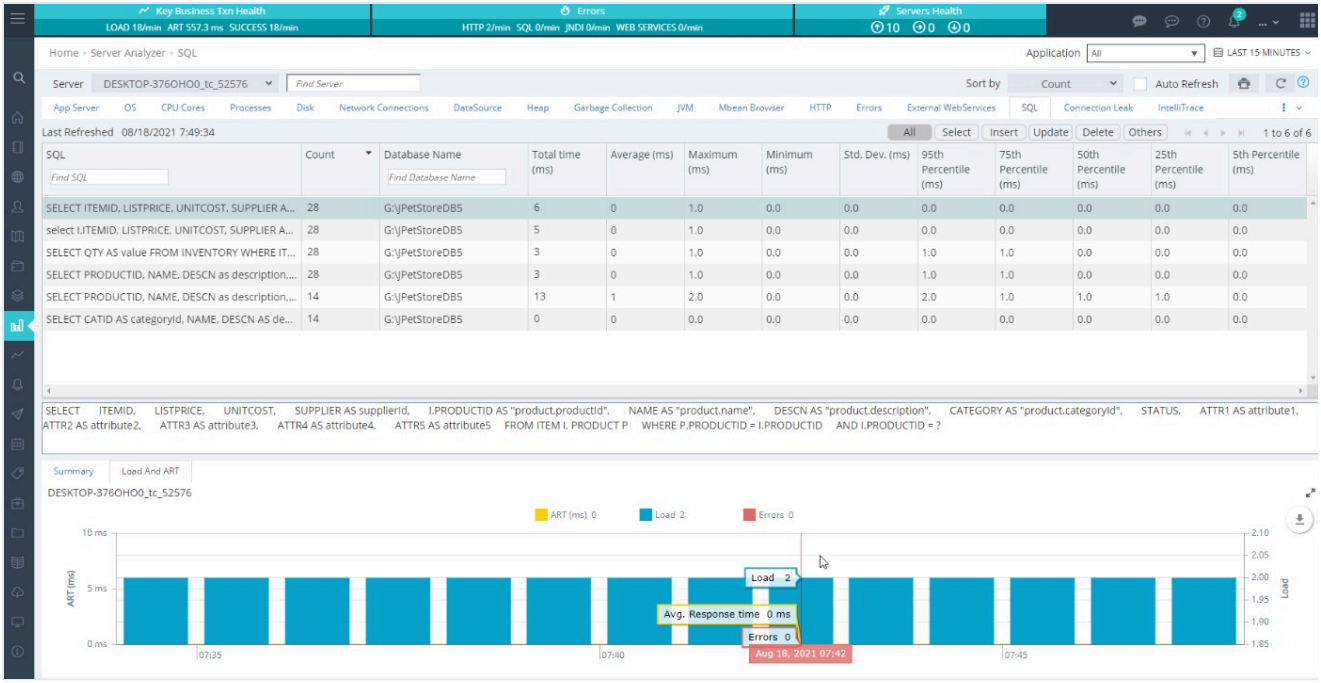
External WebServices monitors all calls to externally hosted webservices and provides performance stats, server ip and port where the calls are going for easily tracking the issues that come with external webservices response time going up.



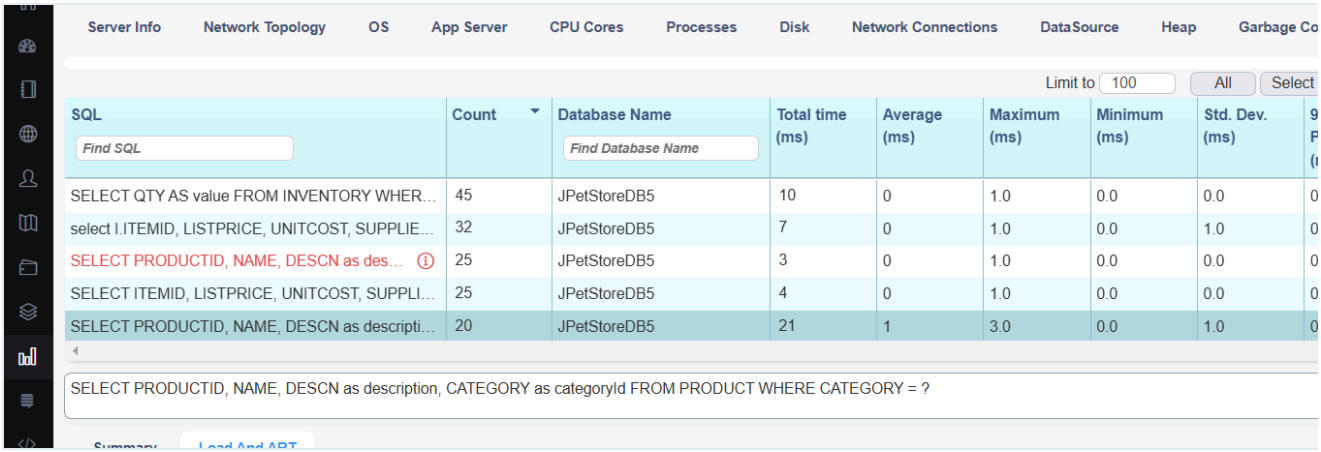
SQL

IntelliTrace Analyzer contains separate views to display data collected by IntelliTrace aspects. SQL Statement Profiling provides real-time performance statistics of database requests. You can view what statements are taking the most time and quickly diagnose performance impacting SQL/s.

Maximum (ms) and Minimum (ms) data will not be available for database monitoring agents.

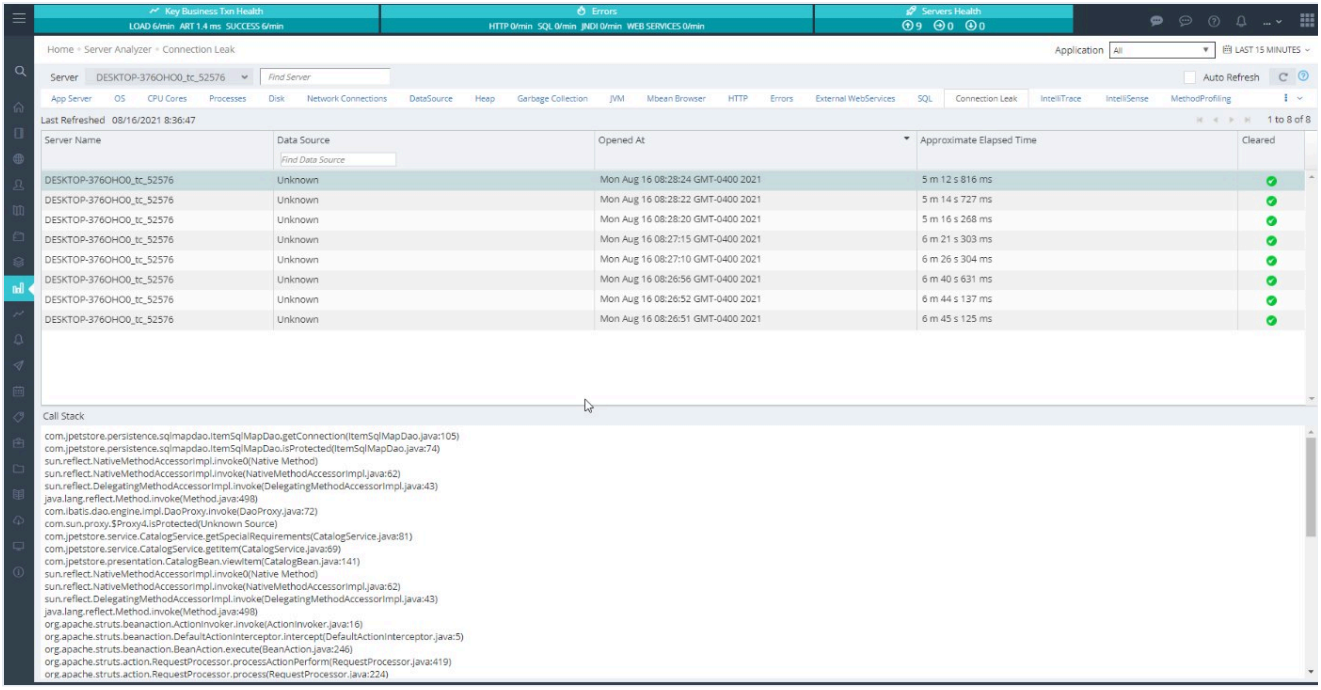


This screen highlights SQL queries vulnerable to injection attacks in red color.



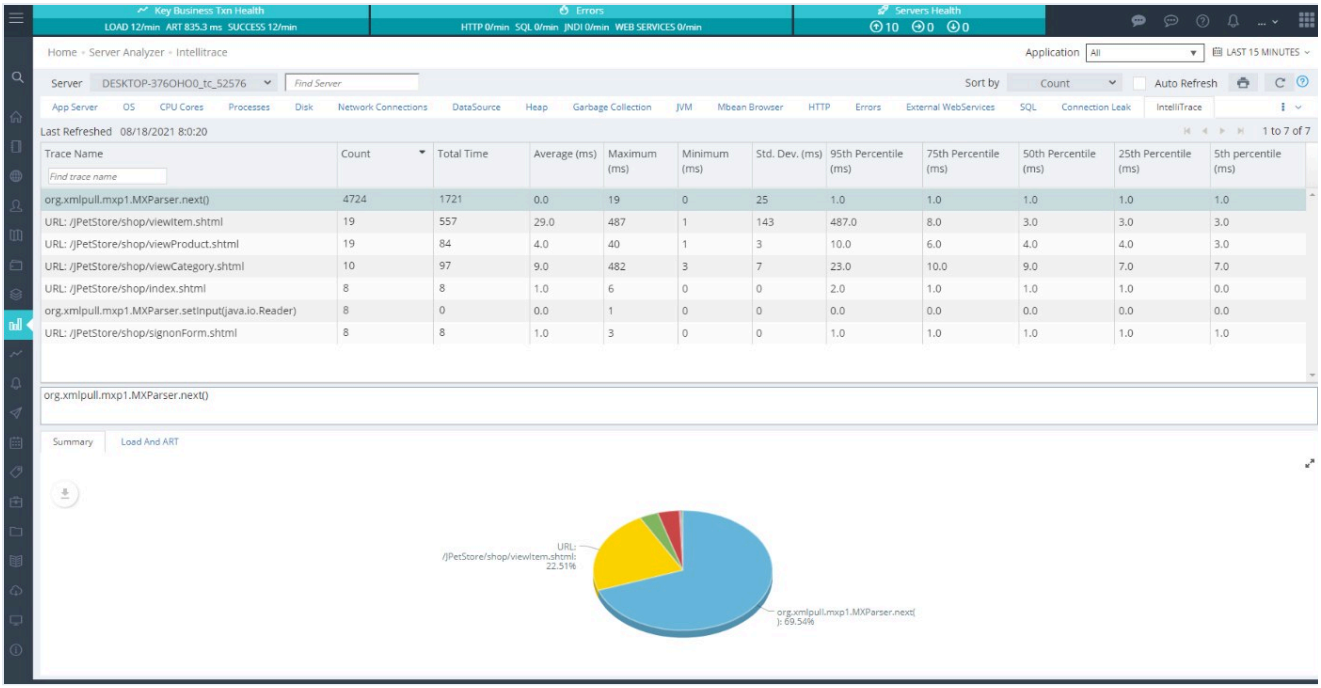
Connection Leak

JDBC Connection Profiling feature allows you to detect leaked JDBC connections, connections held open for long periods of time and code responsible for connection leaks. You can view the connections that are currently in open state and clicking on an open connection will display the call stack which requested the connection. Statistics displayed are for the selected server in the UI.



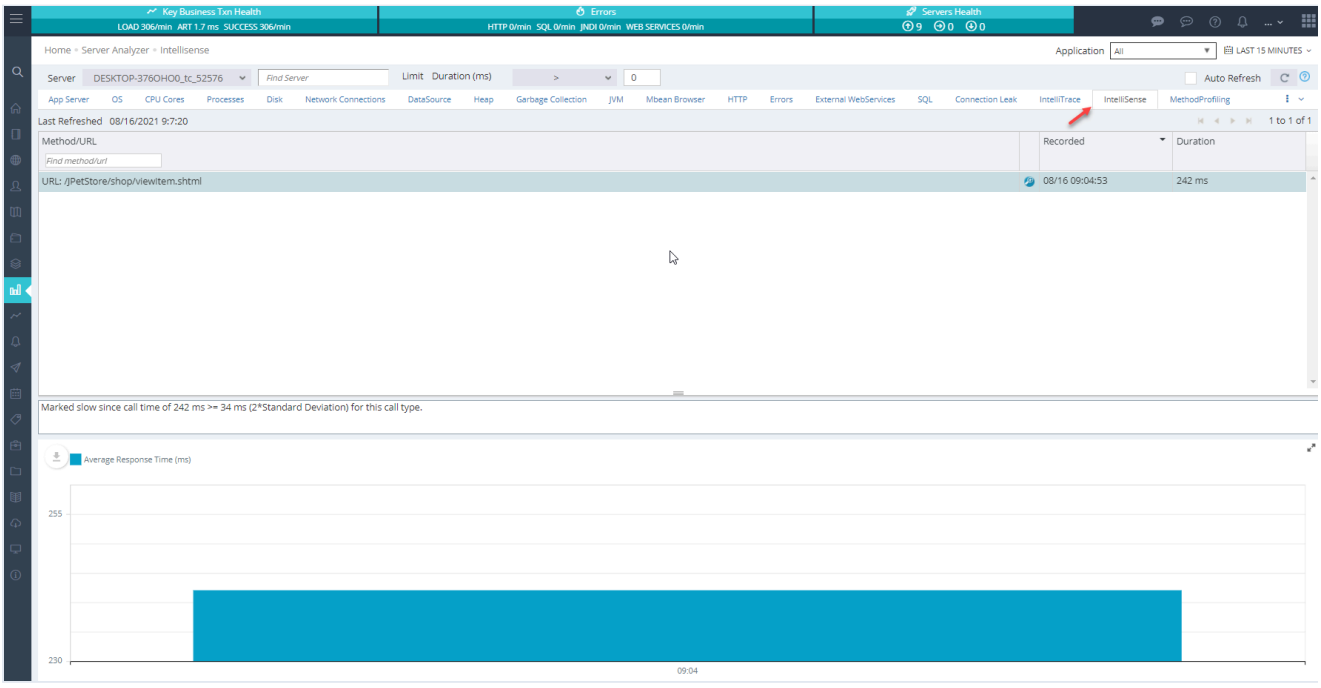
IntelliTrace

IntelliTrace decides what methods to track dynamically and decides the depth of the call stack based on invocation counts and time spent on each individual method to minimize the overhead of tracing. IntelliTrace does what the real experts do with the profiling data - illuminate data that has little impact but it does it in real time. This way no time/resources are spent on gathering data that doesn't have meaningful impact on performance.



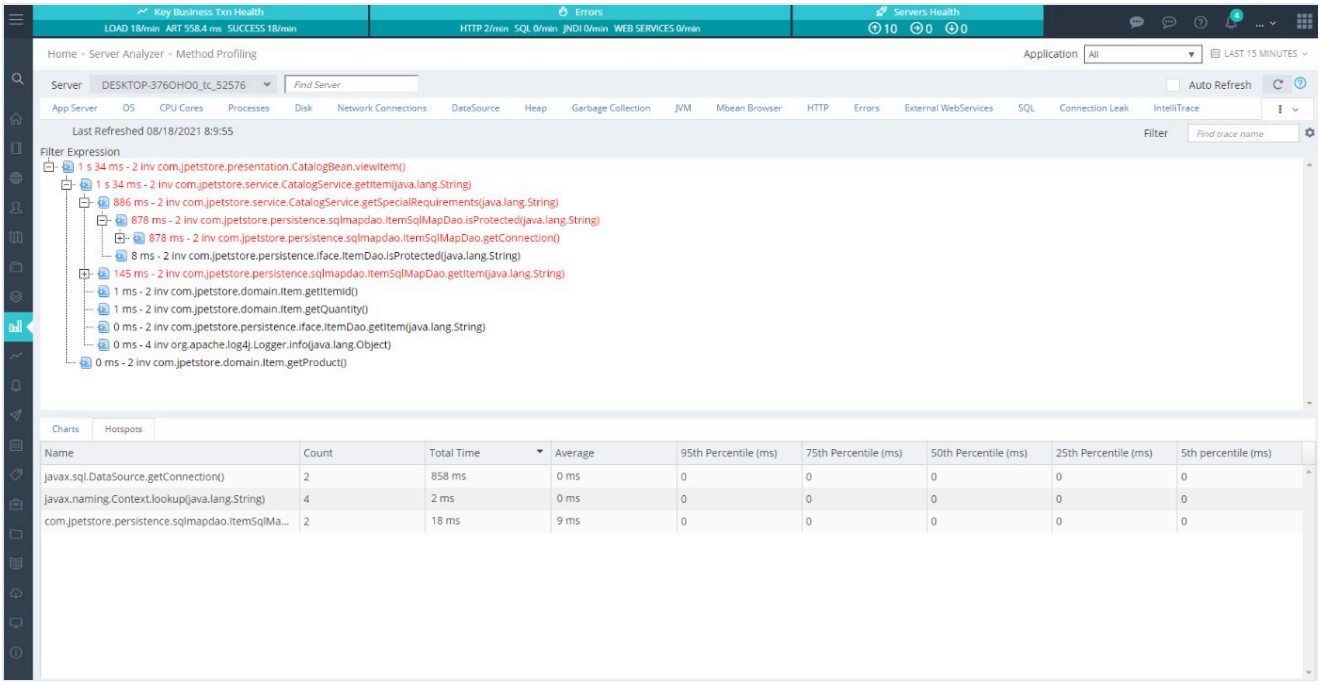
IntelliSense

IntelliSense is a feature that works when IntelliTrace is enabled to automatically detect transaction executions that deviate considerably from the normal execution times and saves a complete call graph of the execution for later analysis.



Method profiling

Method Profiling feature allows you to easily view the performance statistics of real-time method executions on-the-fly. This Profiling Aspect needs to be configured and deployed before you can view performance data. See Method Profiling Configuration for more information on configuration.



JNDI

JNDI calls monitoring tracks JNDI performance and helps identify issues related to lookup delays.

JMS

JMS calls monitoring delivers performance statistics for JMS queues and topics, including metrics such as the number of calls, total time, average duration, and various percentiles.

USING APPLICARE › INFRASTRUCTURE

Applicare Docker Monitoring

Applicare 9.2 · Updated September 13, 2024 · docker · **Pending technical review**

The Docker container must have the following packages installed for monitoring.

procps, net-tools, iotop and sysstat

To install packages inside a Docker container

Enter the Docker containers and install the packages one at a time.

If you encounter an **Unable to locate package** error, run the command **apt-get update** for Debian /Ubuntu or **yum update** for CentOS or RHEL. After updating, install the packages individually.

On Debian, Ubuntu, and Linux Mint

apt-get install procps apt-get install net-tools apt-get install iotop apt-get install sysstat

On RedHat, CentOS and Fedora yum install procps yum install net-tools yum install iotop yum install sysstat

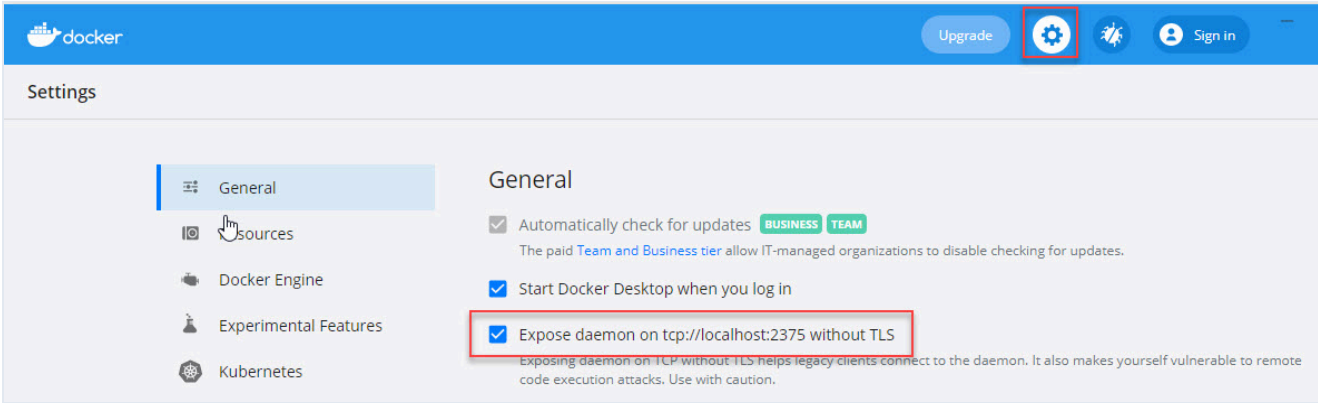
On Fedora 22+ Releases

dnf install procps dnf install net-tools dnf install iotop dnf install sysstat

Windows

For HTTP Monitoring

1. Open the Docker desktop, and go to **settings,** and enable the "**Expose daemon on tcp://localhost:2375 without TLS**" option and restart the docker desktop.



Linux

For HTTP Monitoring

1. Open the "**docker.service**" file and change the "**ExecStart=**" line to the below one

Syntax

```
TEXT
ExecStart=<dockerdpath> –H fd:// –H tcp://<docker_installed_ip>:<docker_running_port>
```

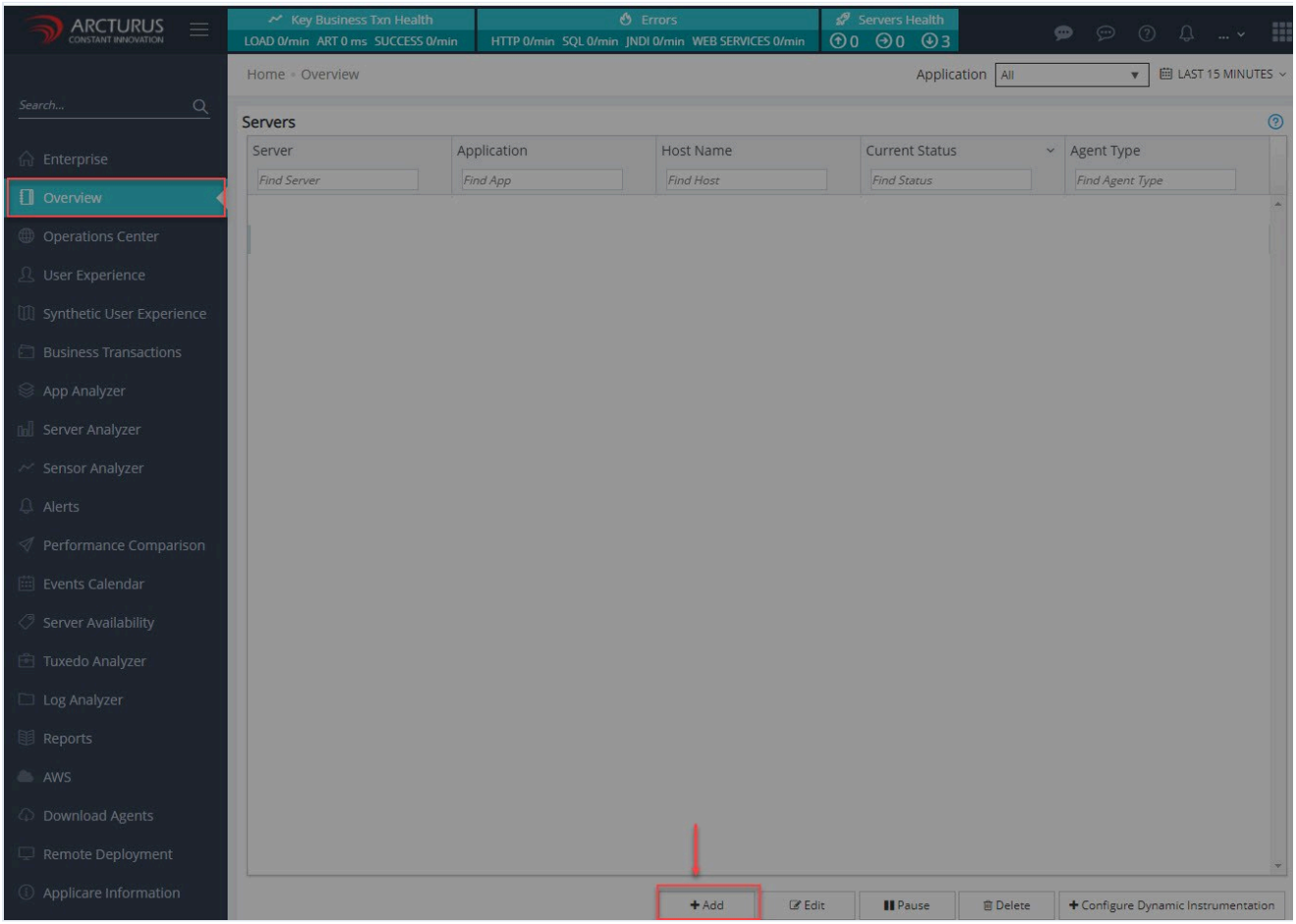
e.g.

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://192.168.0.1:2375

2. Run the command "**systemctl daemon-reload**" and then restart the "**docker.service**".

Applicare Controller

1. Log in to **Applicare Console** and navigate to the **Overview** screen, and click the **Add** button.



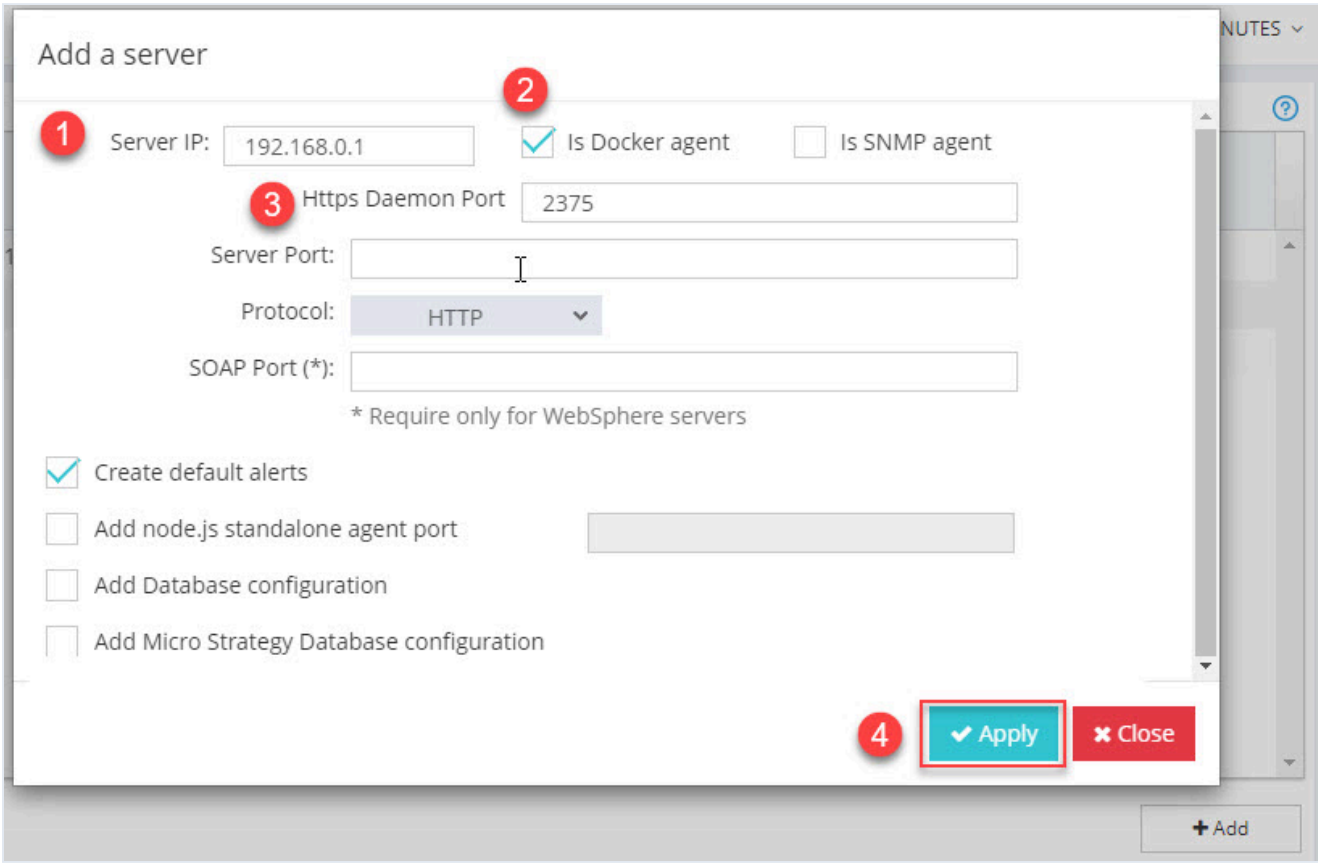
2. Enter the required details and save.

a. Input the **Server IP**

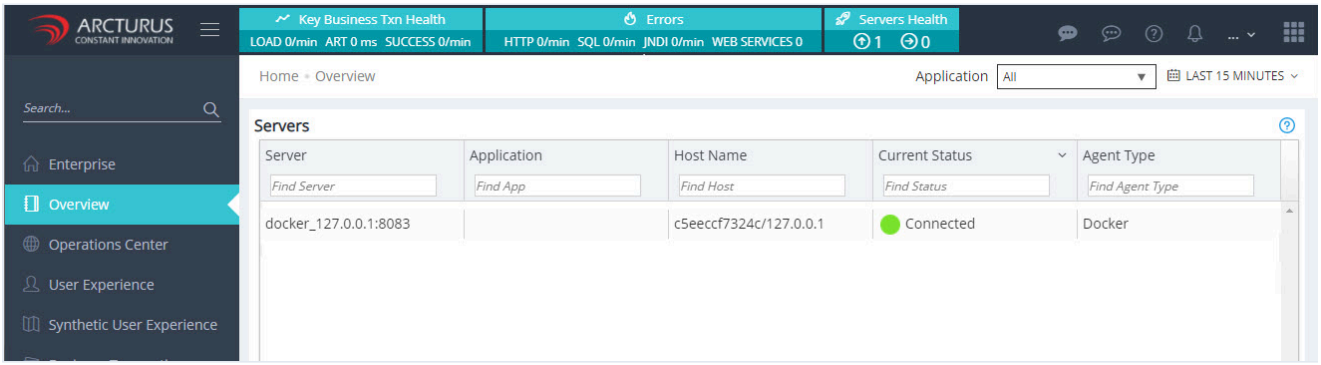
b. Enable the **Is Docker agent** option.

c. Enter the **HTTP Daemon Port**

d. Click the "**Apply**" button.



3. All running containers will be added as agents, any new containers that start will be automatically added as new agent.



4. After adding the containers, navigate to the Server Analyzer screen and check the metrics.

Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

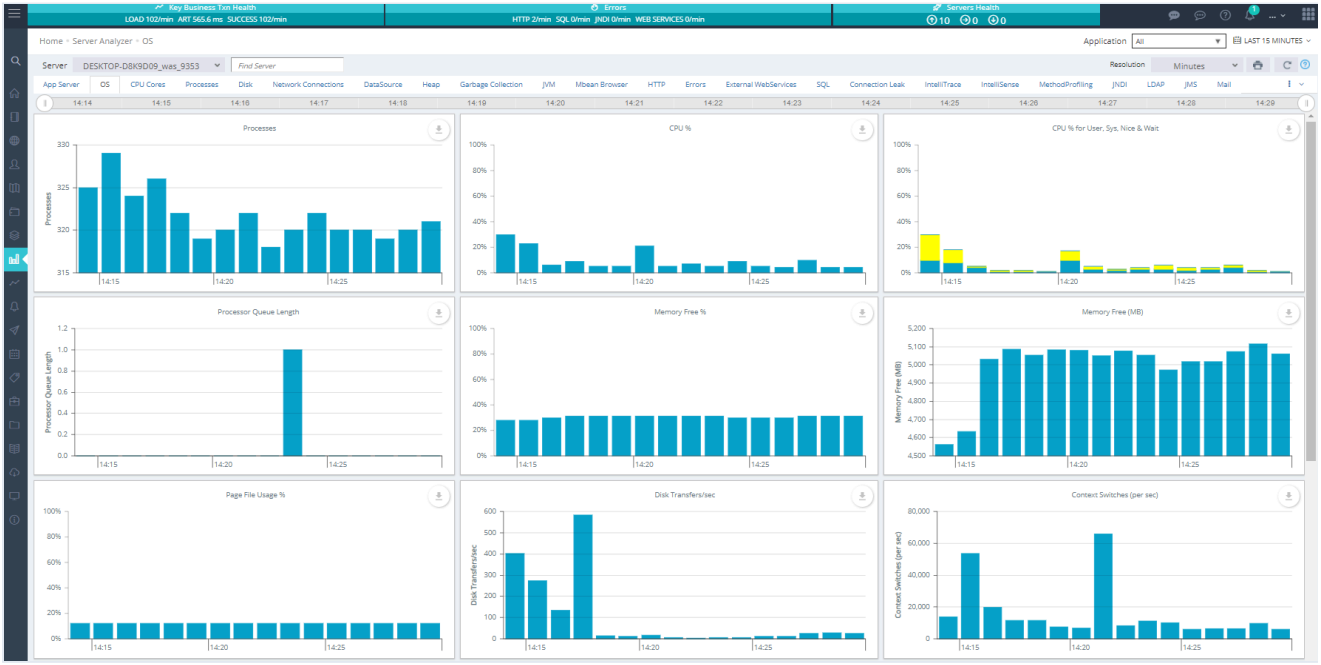
Operating System Analyzer

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The OS Internals Analyzer offers a quick and convenient view into your physical server's processes, network activity, CPU usage, memory and storage performance.

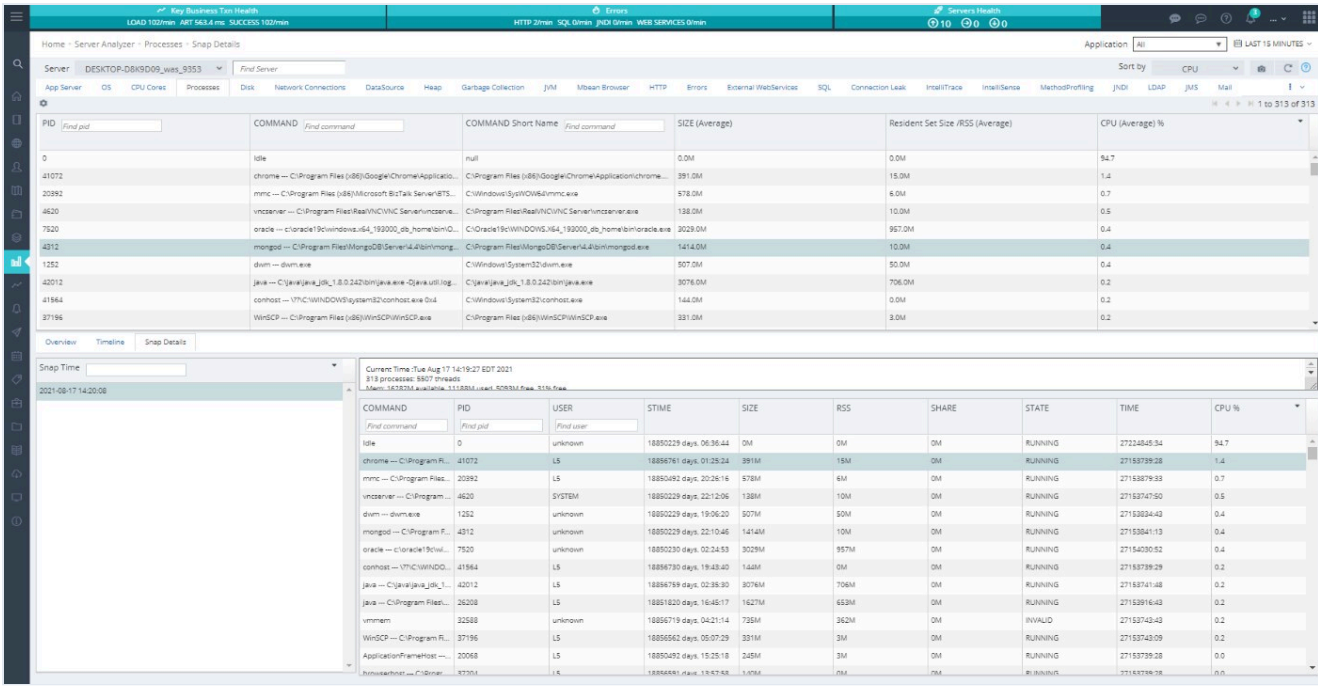
Charts

This tab allows you to monitor overall system performance visually through graphical representations. You can monitor your server's memory usage, system load, CPU utilization, context switching, and network data.



Processes

This view displays performance stats for CPU, memory and running processes, organized similarly to the output of the **Top** utility on Unix-based systems.



Column Descriptions:

PID :	**The process ID assigned to each task.**
USER :	The username of the task's owner.
STIME :	The time when the process was initiated.
SIZE :	The size of the task's code, data and stack space, displayed in megabytes, is shown here.
RSS :	The total amount of physical memory used by the task, measured in megabytes, is shown here.
SHARE :	This column displays the amount of shared memory used by the task.
STATE :	This column indicates the task's state, which can be S for sleeping, D for uninterruptible sleep, R for running, Z for zombies, or T for stopped or traced.
TIME :	The total CPU time utilized by the task since its start, displayed in MM.
CPU :	The task's current CPU time usage, expressed as a percentage.

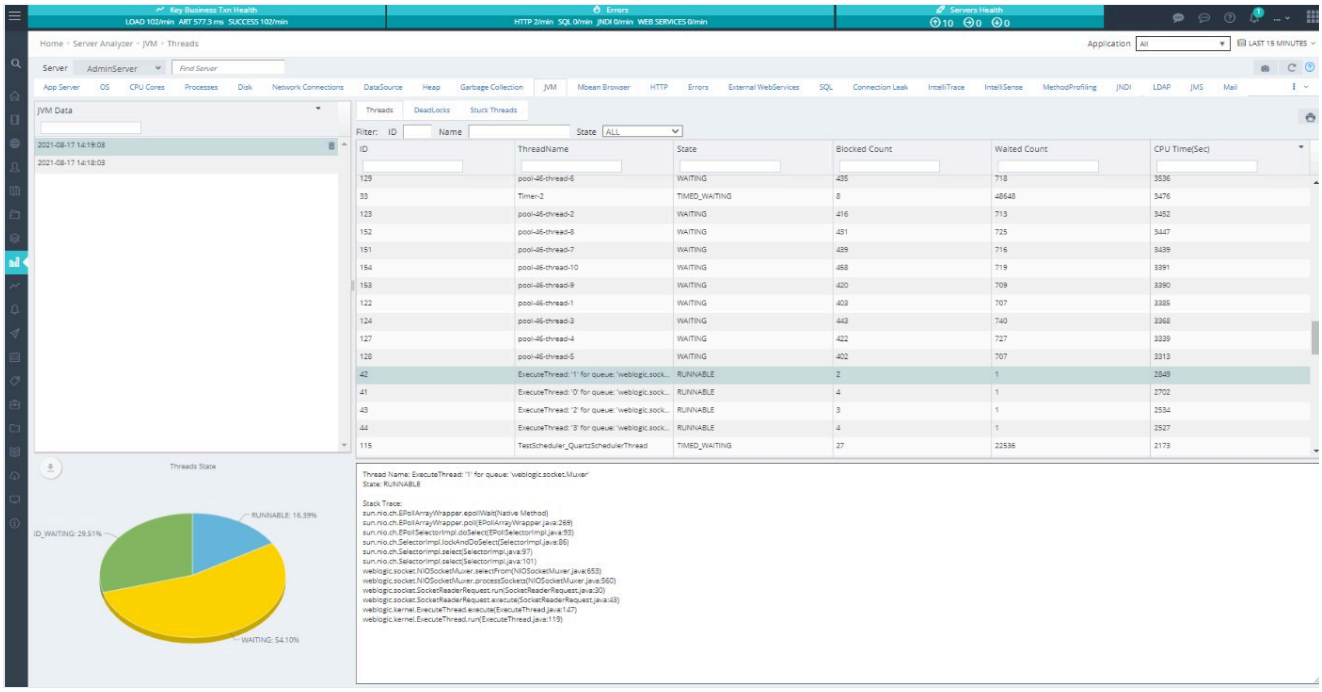
USING APPLICARE › INFRASTRUCTURE

JVM Analyzer

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Thread Dumps

This view enables you to capture a thread dump of a live server, which you can analyze the saved later. The saved snapshots include all the thread details visible in the real-time view and as well as any deadlock or stuck threads detected when the dump was created.



Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

GC Analyzer

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The GC Analyzer offers a quick and easy way to examine your server's garbage collector performance. This tool enables you to swiftly identify garbage collection issues and adjust GC parameters, enhancing performance and reducing application latency.

The GC Analyzer is available exclusively on Sun JVMs and requires that verbose GC logging be enabled on the JVM.

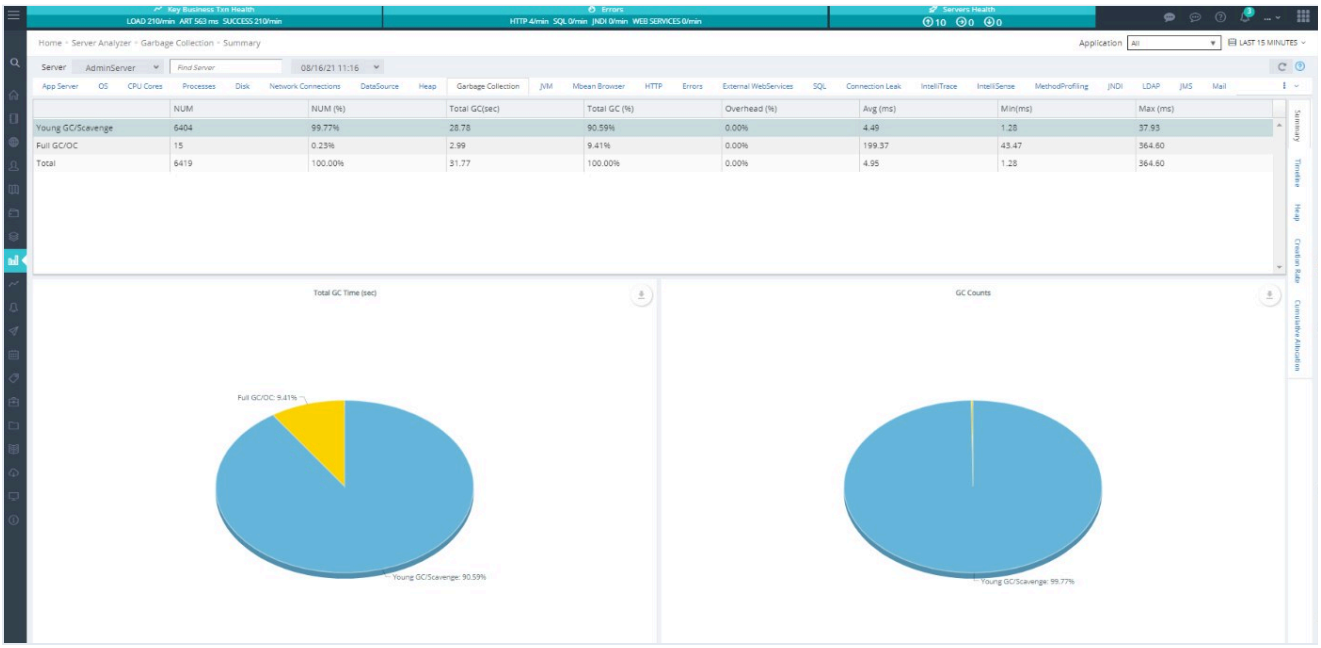
Applicare automatically adds the following argument to enable verbose GC:

-verbosegc -Xloggc:%SERVER_NAME%.gc

Currently, the use of the **-XX:+PrintTenuringDistribution** flag is not supported and will prevent Applicare from accessing the GC log file.

Summary View

The summary tab provides an overview of Full and Young GC cycles that have occurred throughout the JVM's lifetime. A key value to watch is the Overhead (%) displayed in the table.



Detailed Charts

The GC Analyzer features various charts that visually represent different aspects of Garbage collection cycles.

Timeline Tab - Provides a visual representation of each GC cycle over time, offering a clear view of the type of GC cycle and then duration it took.

Heap Tab - Displays a graph of heap size at the end of each GC cycle, tracking changes over time.

Object Creation Rate Tab - This graph shows the total size of objects created between each GC cycle.

Cumulative Allocation Tab - Displays the total cumulative size of objects created over time.

Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

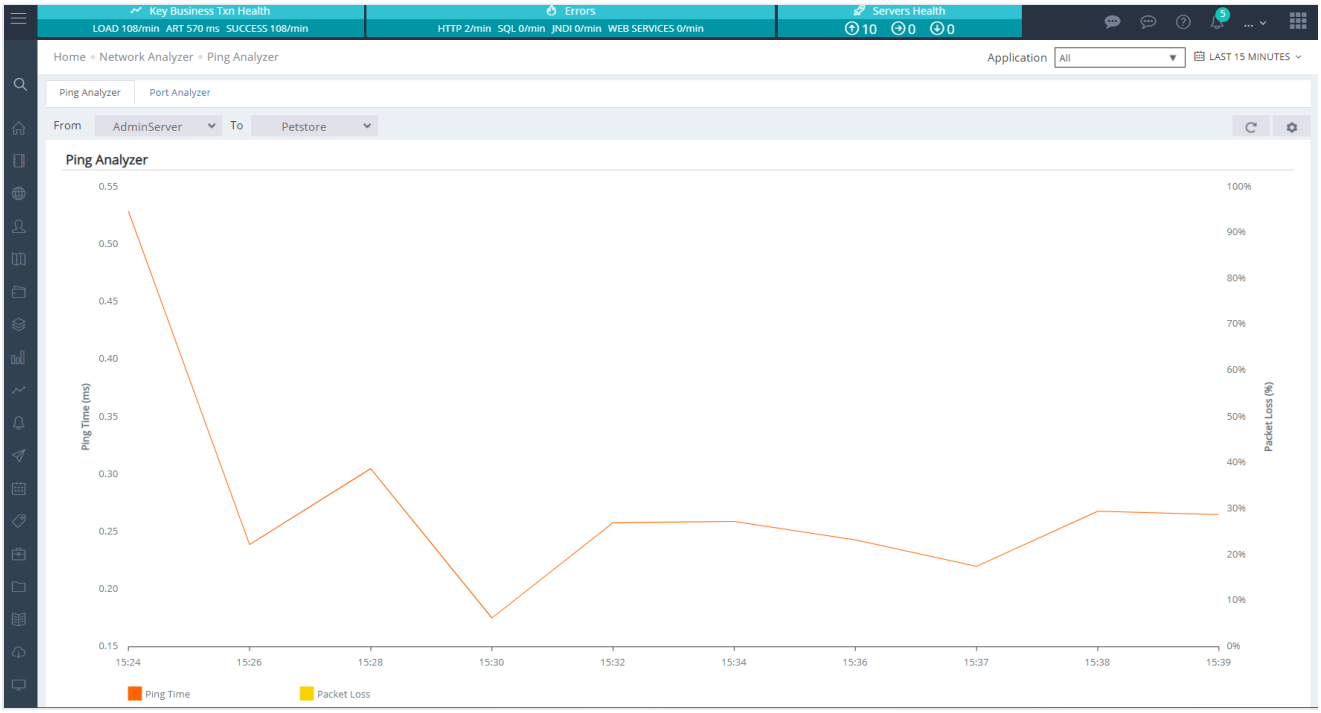
Network Analyzer

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

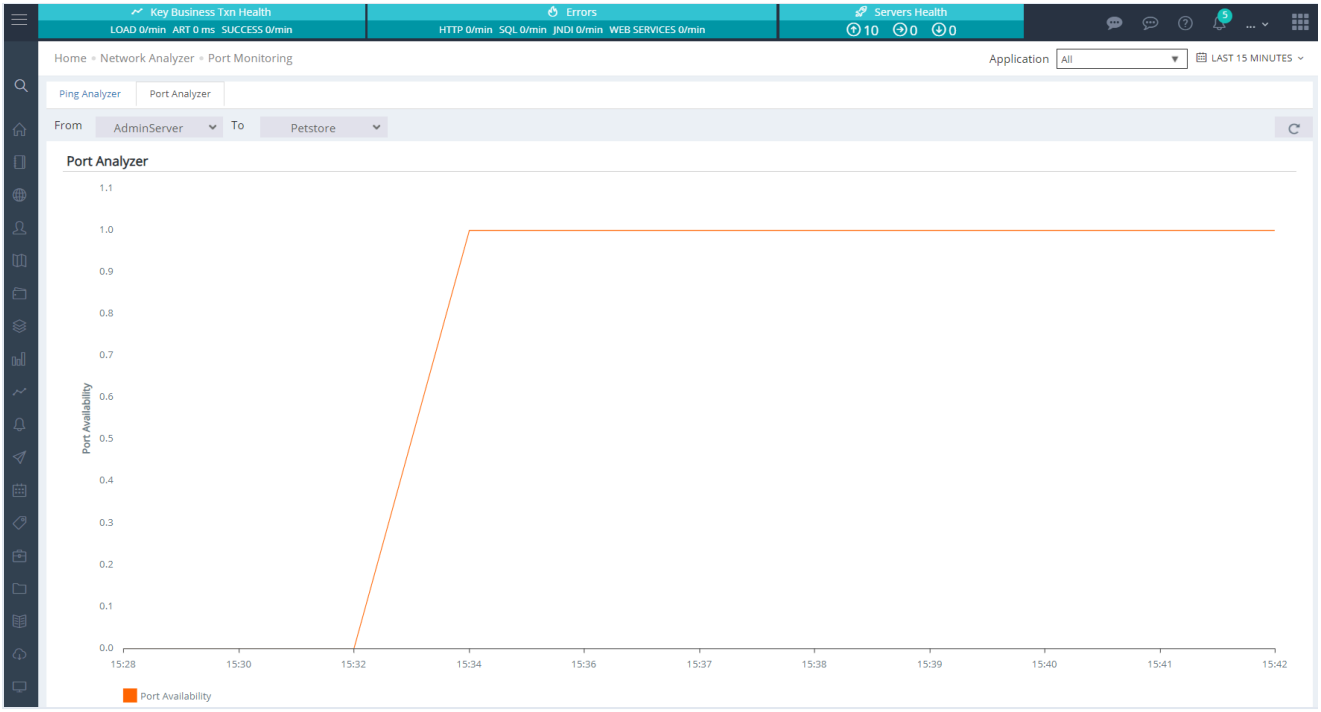
Ping Analyzer and Port Monitor

The Ping Analyzer measure the latency and packet loss when pinging a server from either the Applicare server or Applicare agents. This utility is essential for diagnosing network issues. You can configure alerts based on these metrics through JMX Alerts by navigating to Applicare Ping Monitor MBean and adjusting the Ping Time and Packet Loss properties.

Ping Analyzer

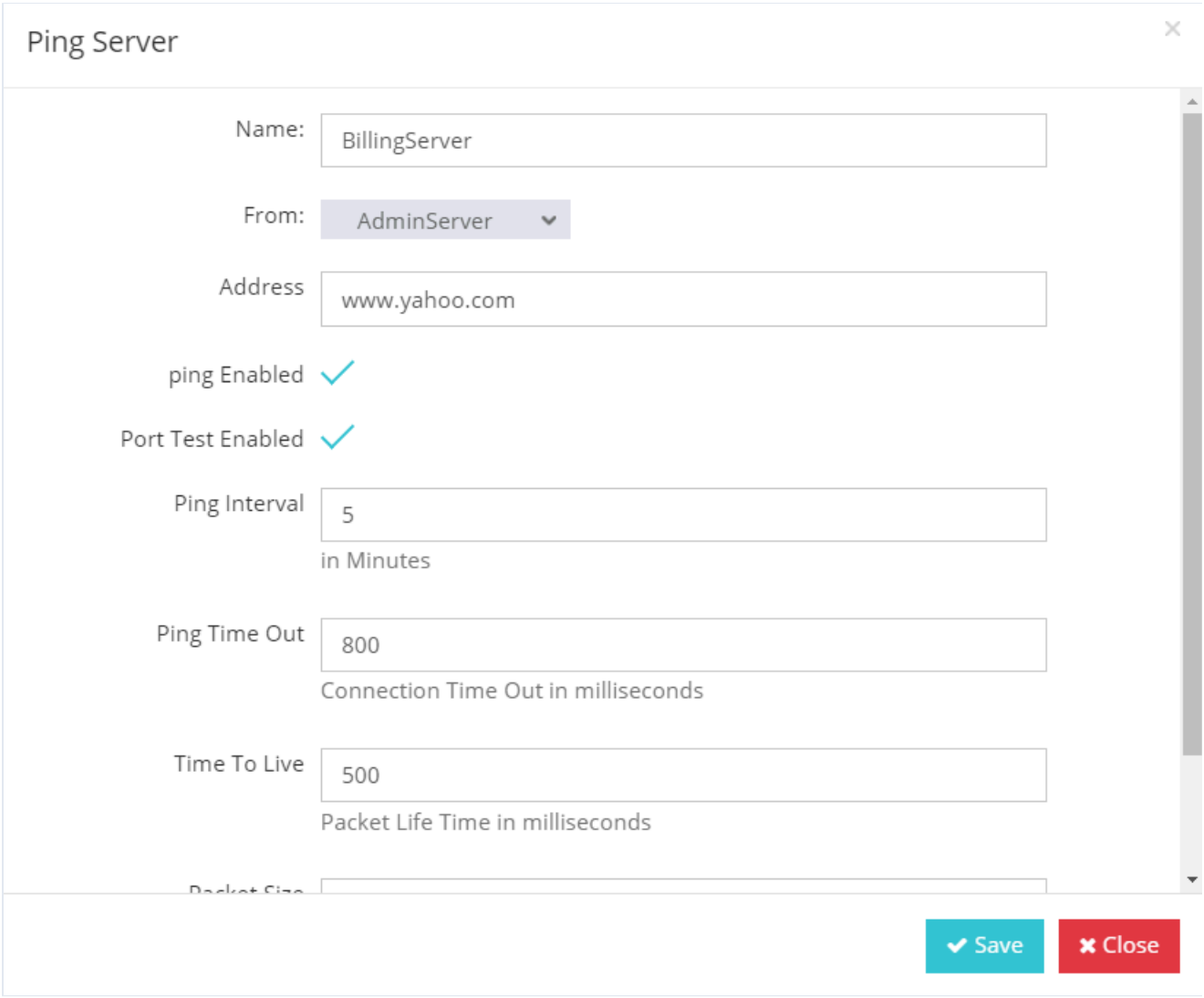


Port Analyzer



Parameters

Time to Live(TTL): Duration for which a sent packet remains active before being discarded. **Packet size:**The Size of the packet being sent. **Time Out:** Duration after which the connection will be terminated if no response is received.



Monitoring with Applicare

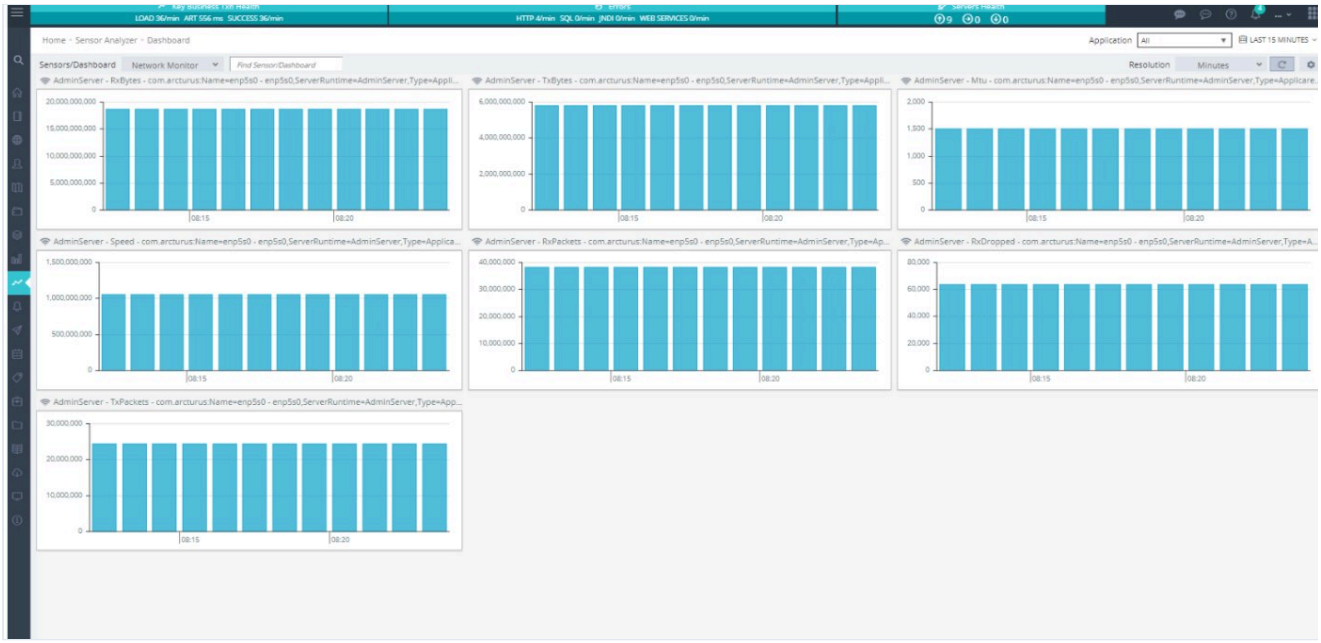
USING APPLICARE › INFRASTRUCTURE

Sensor Analyzer

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The Sensors Analyzer offers an accessible view of data gathered by JMX Sensors over time. For guidance on setting up JMX Sensors, refer to the instructions under Create Sensors.

You can customize your view by adding or removing charts. Simply click the **Sensors** button and choose the charts you wish to include or exclude.



Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

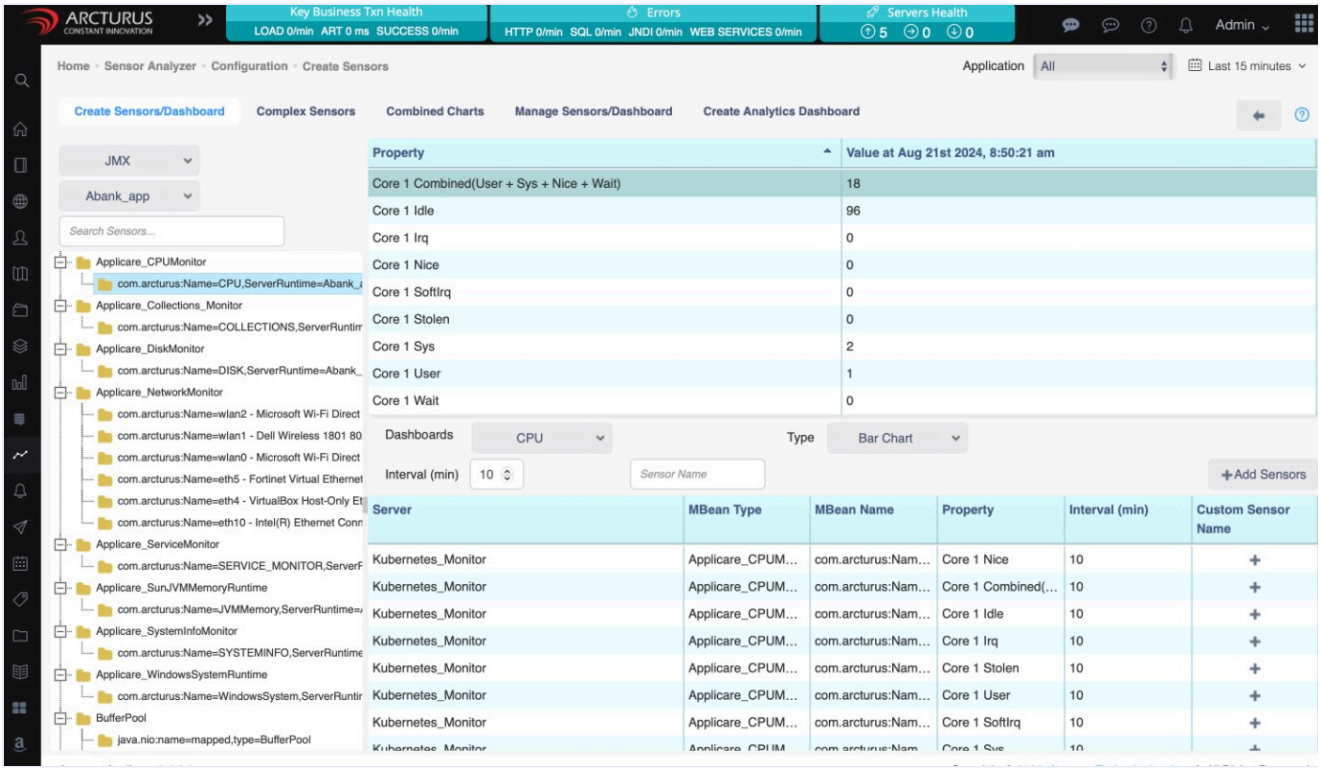
Applicare Custom JMX Analysis: Create a Sensor

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

An Applicare JMX Sensor is a JMX property whose value is stored by Applicare for analysis at any time. The server instance does not need to be active during the analysis. Applicare uses an efficient mechanism to persist sensor data with minimal overhead.

Create Sensors

To access the "Creating JMX Sensors" screen, select **Applicare Home > Sensors Analyzer > Configure Sensors** button on right (wheel). The tree structure on the **Create Sensors** tab can be expanded to display the MBeans and its properties. You can select the Properties to create sensors and click **Save Sensors** button to save. This is illustrated below:

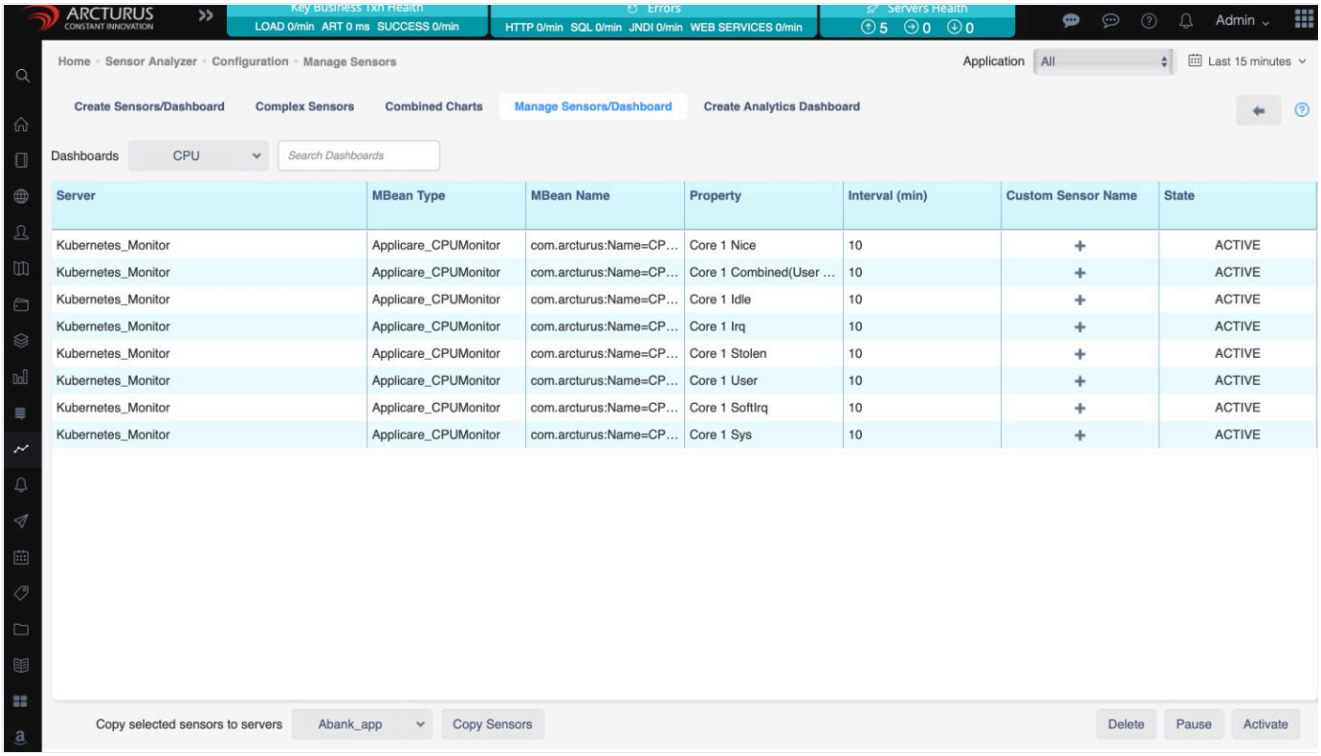


Click here to configure DB Query Sensor

Manage Sensors

Manage Sensors tab can be used to view all created sensors, delete and copy sensors from one server to another.

To copy sensors, select sensors you want to copy and select the target server from server drop down at the bottom. Click **Copy Sensors** button to perform the copy. MBean names should exist on the target server.



Monitoring with Applicare

USING APPLICARE › INFRASTRUCTURE

Monitoring MSSQL Metrics with DB Query Sensors

Applicare 9.2 · Updated August 29, 2024 · **Pending technical review**

The user must have enough permissions to access these metrics.

For details on user Permissions and DB Configuration Click here

Using DB Query Sensors to Retrieve MSSQL Metrics

Private Sensor Name: Total Connection **SQL:** select sum(cntr_value), '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name in ('User Connections','Logical Connections');

Private Sensor Name: User Connections **SQL:** select cntr_value,'~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='User Connections';

Private Sensor Name: Logical Connections **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Logical Connections';

Private Sensor Name: Active Temp Tables **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Active Temp Tables';

Private Sensor Name: Lock Blocks Allocated **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Lock Blocks Allocated';

Private Sensor Name: Lock Owner Blocks Allocated **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Lock Owner Blocks Allocated';

Private Sensor Name: Connection Memory **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Connection Memory (KB)';

Private Sensor Name: Process blocked **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Processes blocked';

Private Sensor Name: Transactions **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Transactions' and object_name like '%Transactions%';

Private Sensor Name: Active Transactions **SQL:** select sum(cntr_value), '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Active Transactions';

Private Sensor Name: Total Database file size (KB) **SQL:** select sum(cntr_value), '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Data File(s) Size (KB)';

Private Sensor Name: Total Database Log file size (KB) **SQL:** select sum(cntr_value), '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Log File(s) Size (KB)';

Private Sensor Name: Buffer cache hit ratio **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Buffer cache hit ratio';

Private Sensor Name: Database pages **SQL:** select cntr_value, '~' AS Applicare_Internal from sys.dm_os_performance_counters where counter_name='Database pages' and object_name like '%Transactions%';

USING APPLICARE › INFRASTRUCTURE

Server Availability Monitoring

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Overview

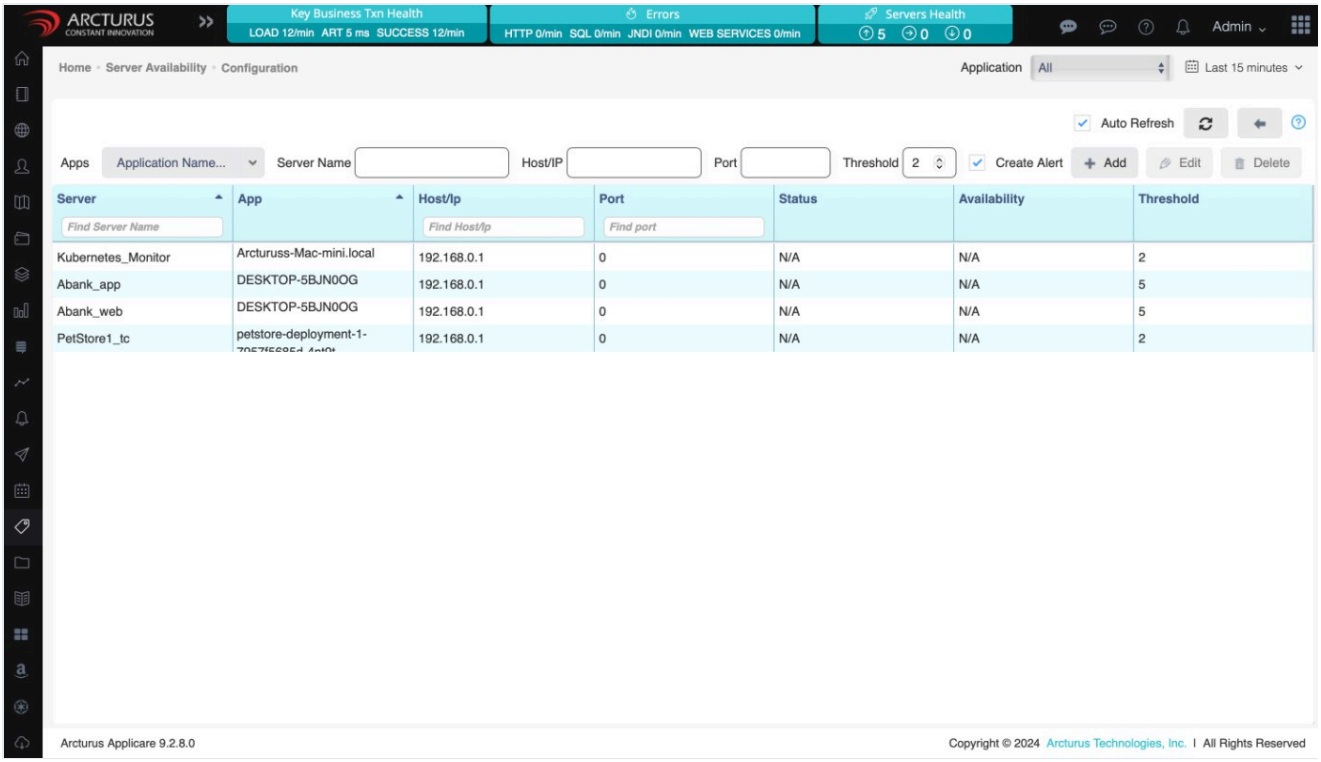
Server availability monitoring tracks changes in server status and generates reports on uptime and performance.

Server Availability

The Server Availability chart displays the status of the servers as available, not available, not monitored, or under blackout. A blackout status indicates that a blackout was or is currently in effect during that time period.



Server Availability Monitoring Configuration



Monitoring with Applicare

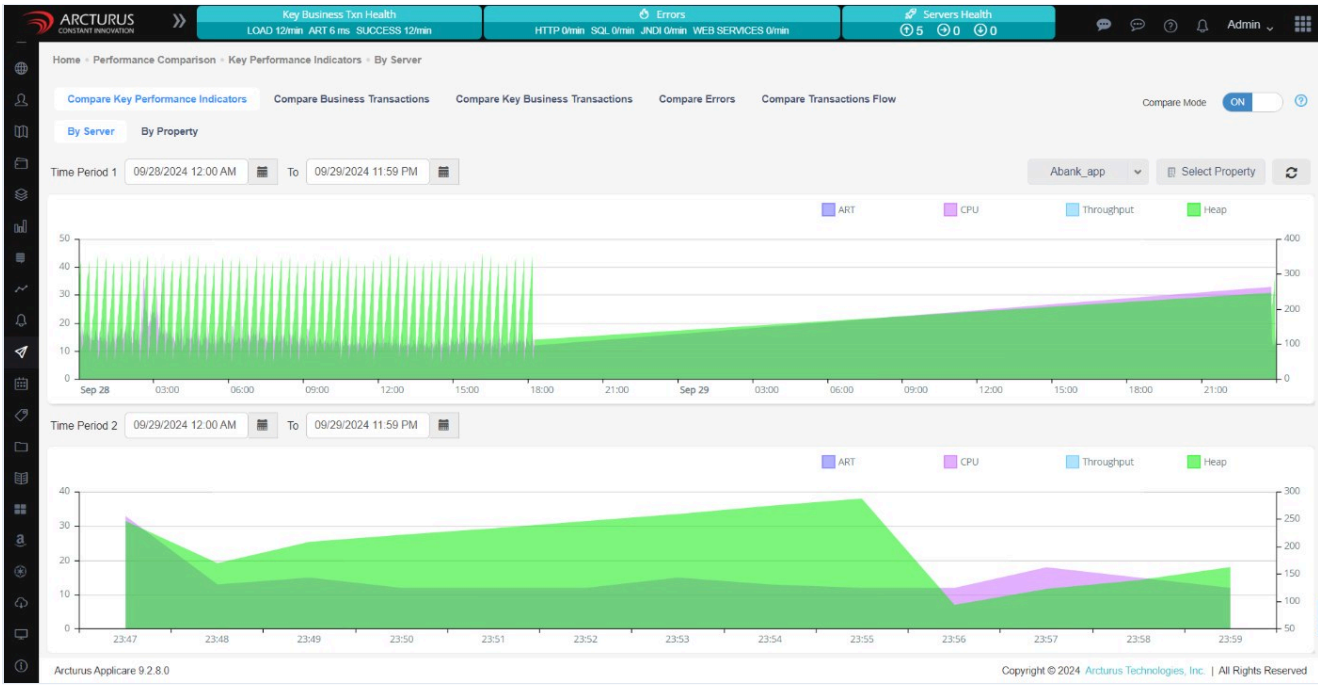
USING APPLICARE › INFRASTRUCTURE

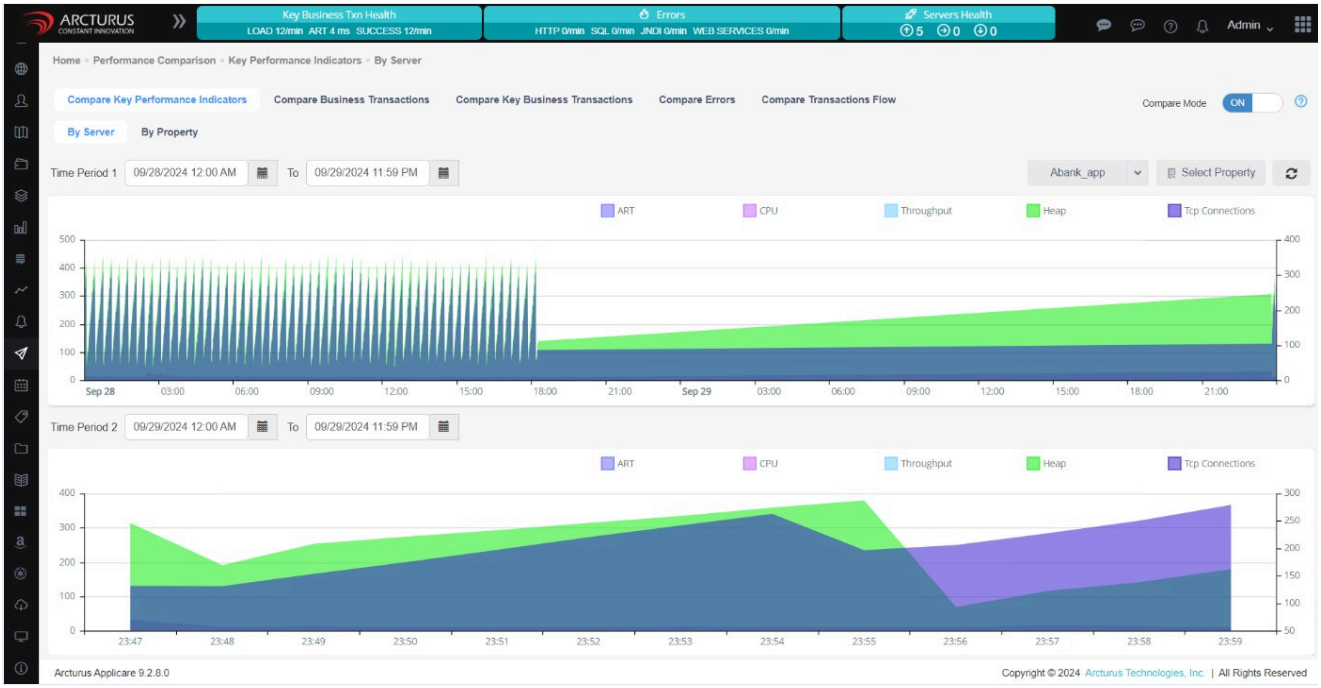
User Guide for Performance Comparison

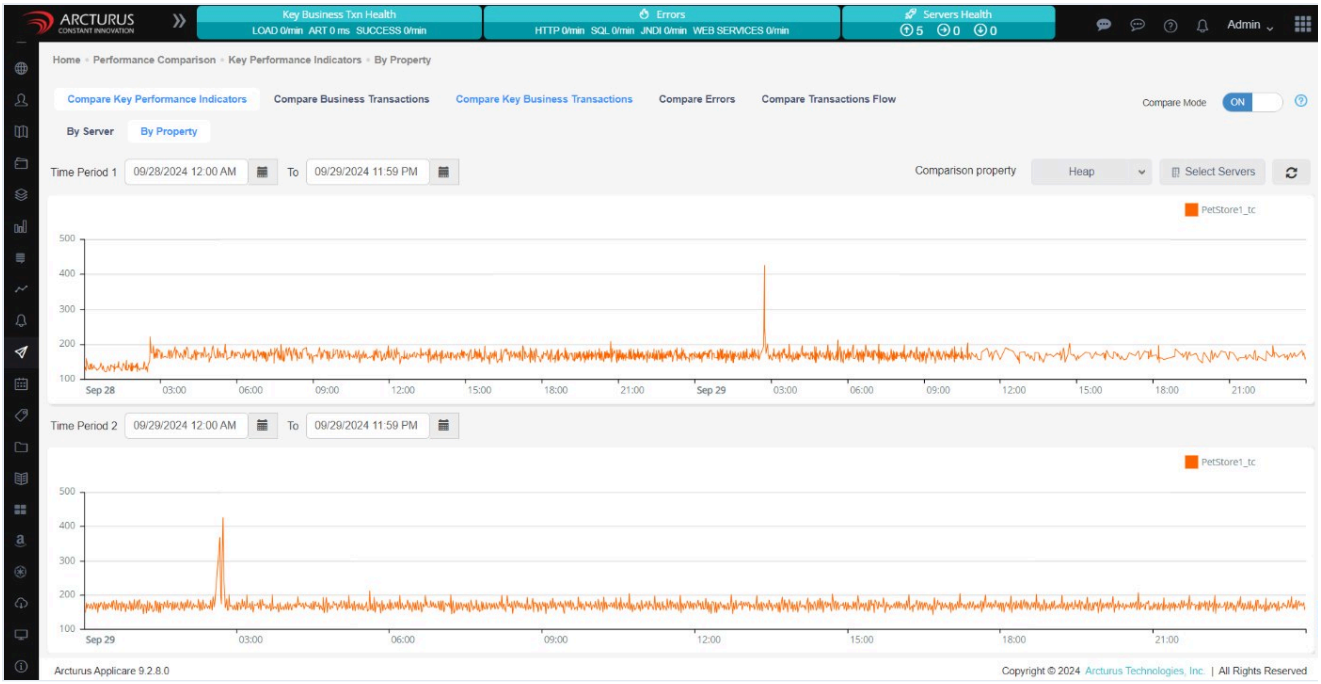
Applicare 9.2 · Updated September 30, 2024 · **Pending technical review**

Key Performance Indicators

Performance Comparison allows of various performance metrics such as (Heap, CPU, Throughput, Average Response Time etc.) between two time periods. It is extremely simple - select the time periods, select servers to compare, choose property to compare e.g. CPU on NodeServer16.

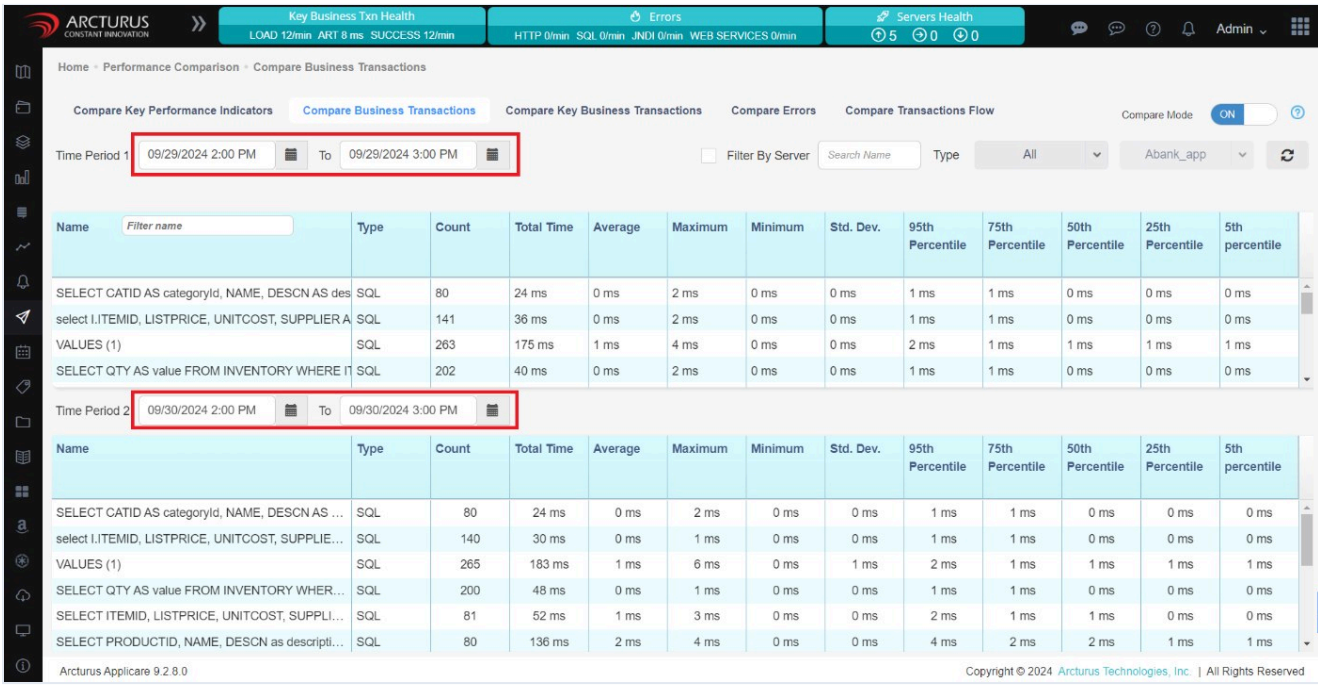






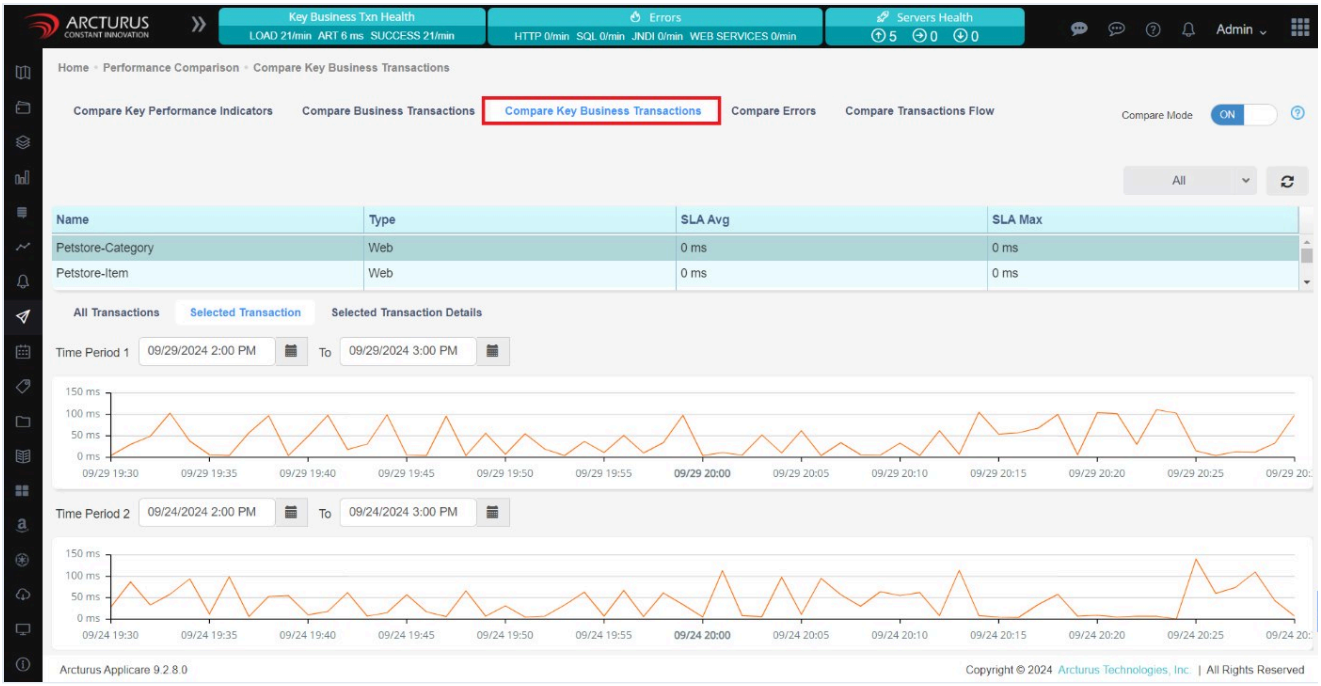
Compare Business Transactions

Displays the HTTP data for all servers in the Applicare Controller based on the specified time period.



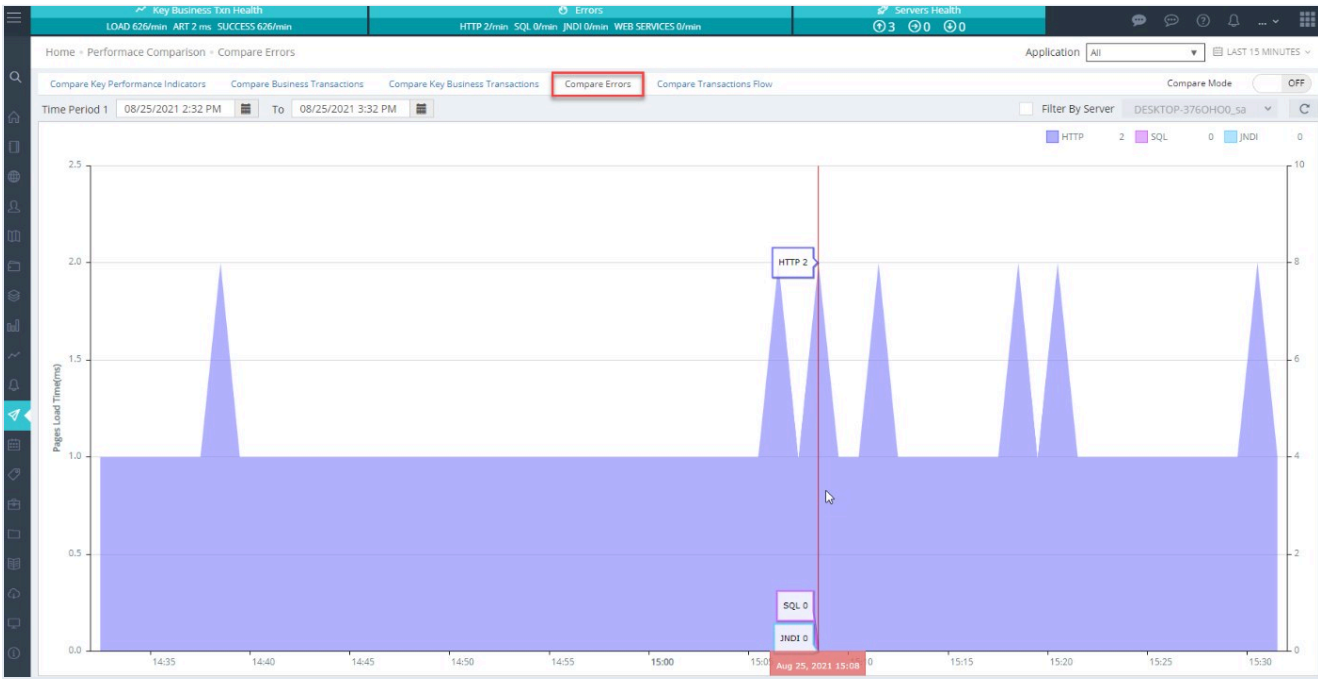
Compare Key Business Transactions

If KBT is added to the server in the Applicare controller then all the KBT will be displayed according to the time period we have given.

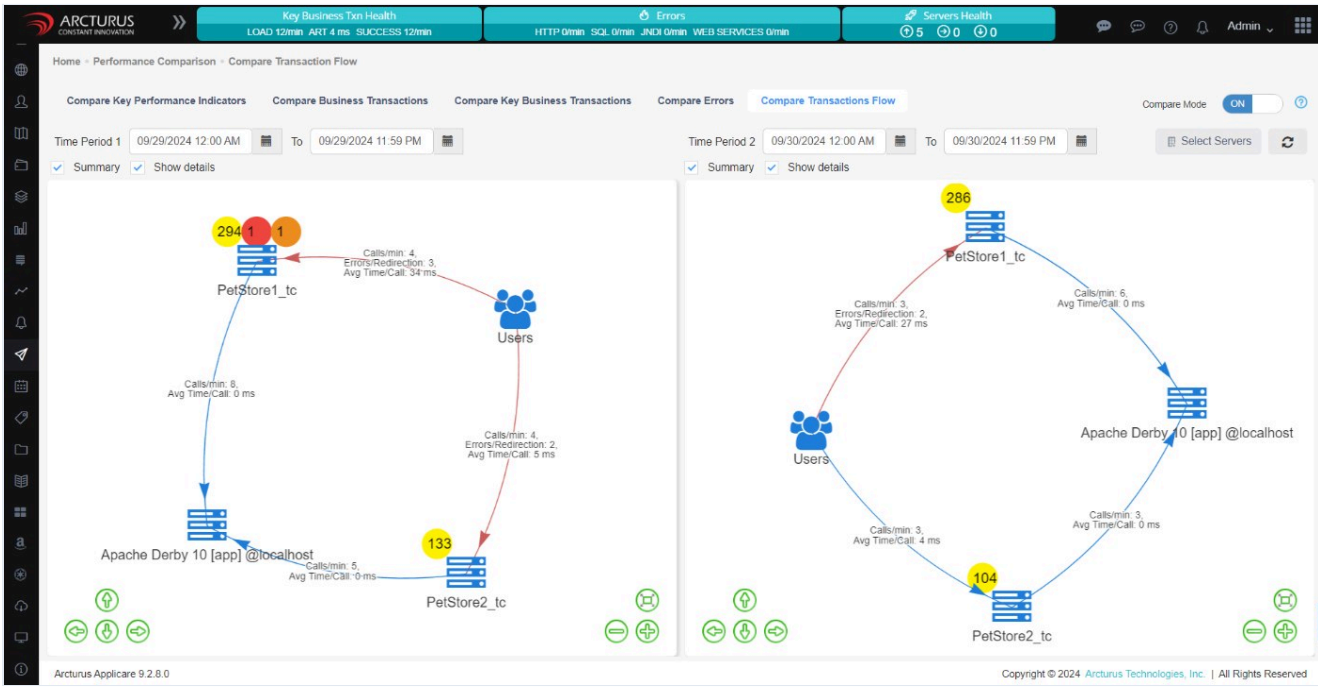


Compare Errors

Display any errors like HTTP errors, SQL errors, or JNDI issues, encountered on the server within the specified time period in the Applicare controller.



Compare Transaction Flow

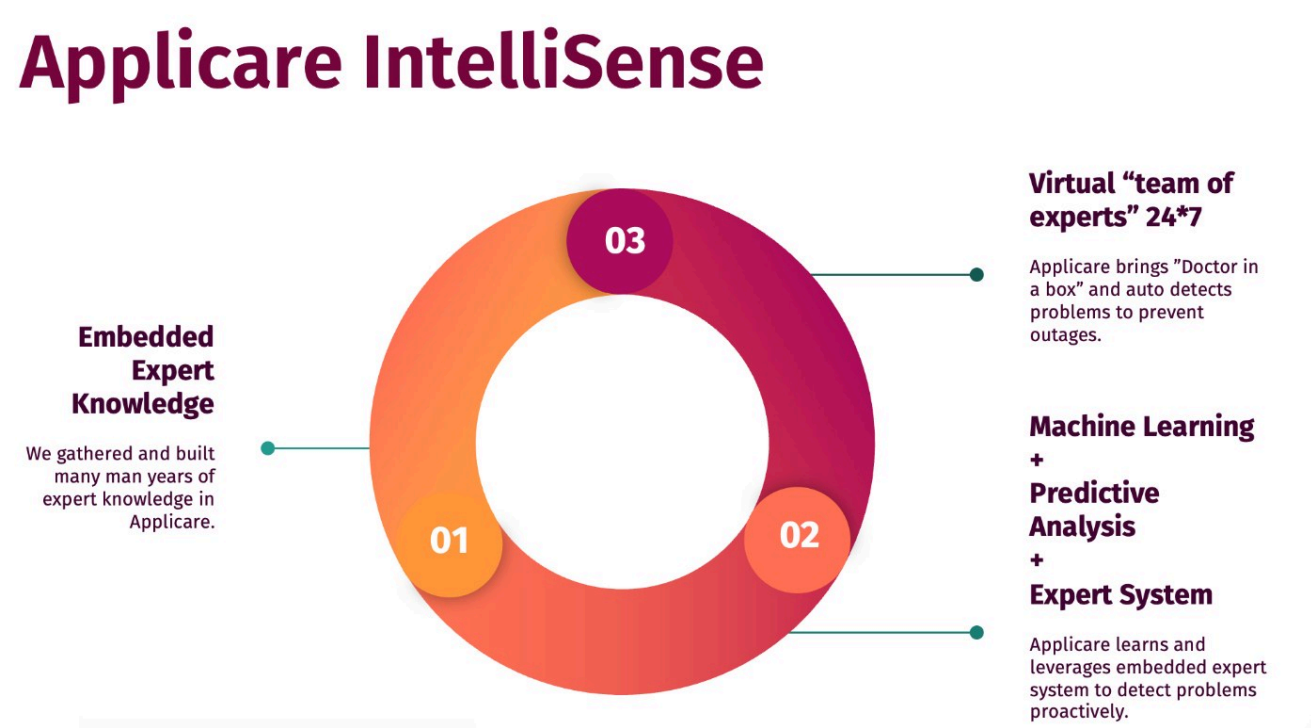


Monitoring with Applicare

USING APPLICARE › AIOPS

Applicare IntelliSense

Applicare 9.2 · Updated September 13, 2024 · aiops · **Pending technical review**



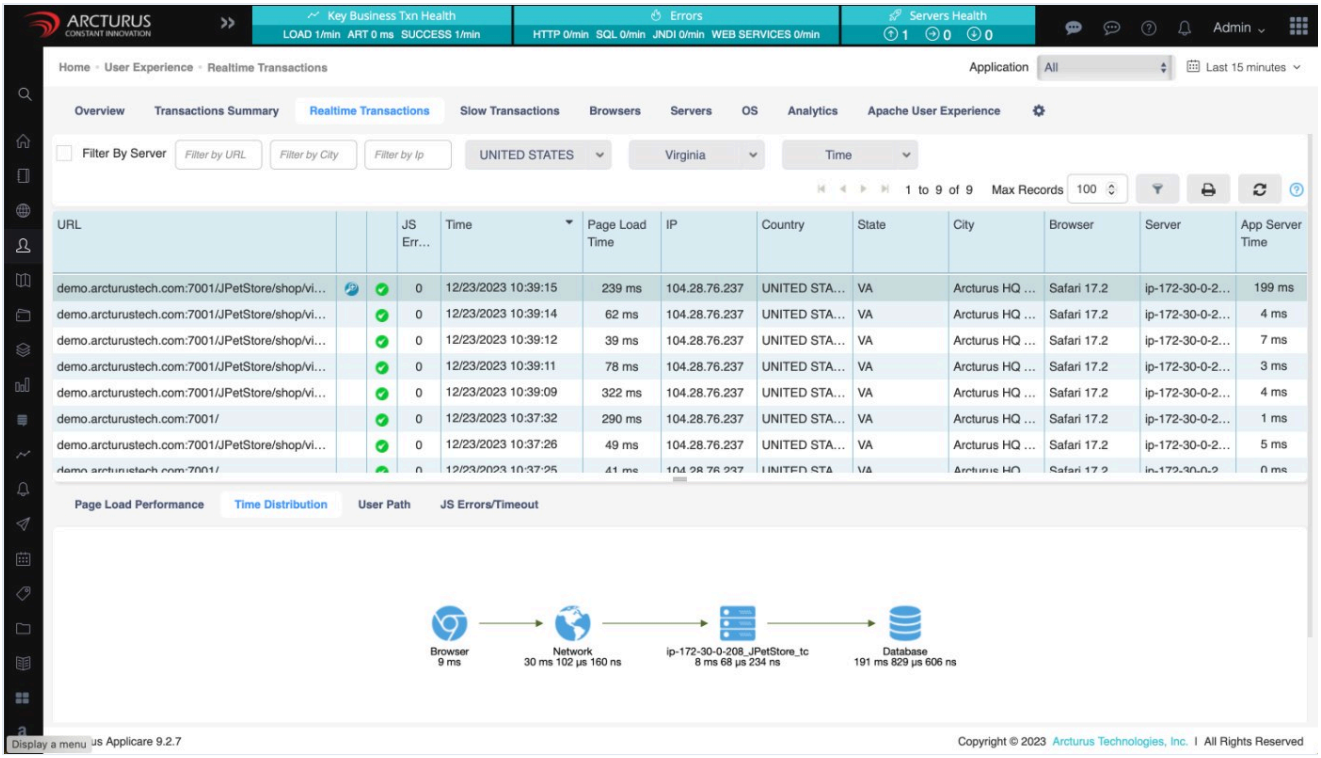
Applicare IntelliSense leverages AI to generate

- Machine Learning for Anomaly Detection
- Predictive Forecasting
- Problematic Pattern Recognition and Alerting
- Generative AI

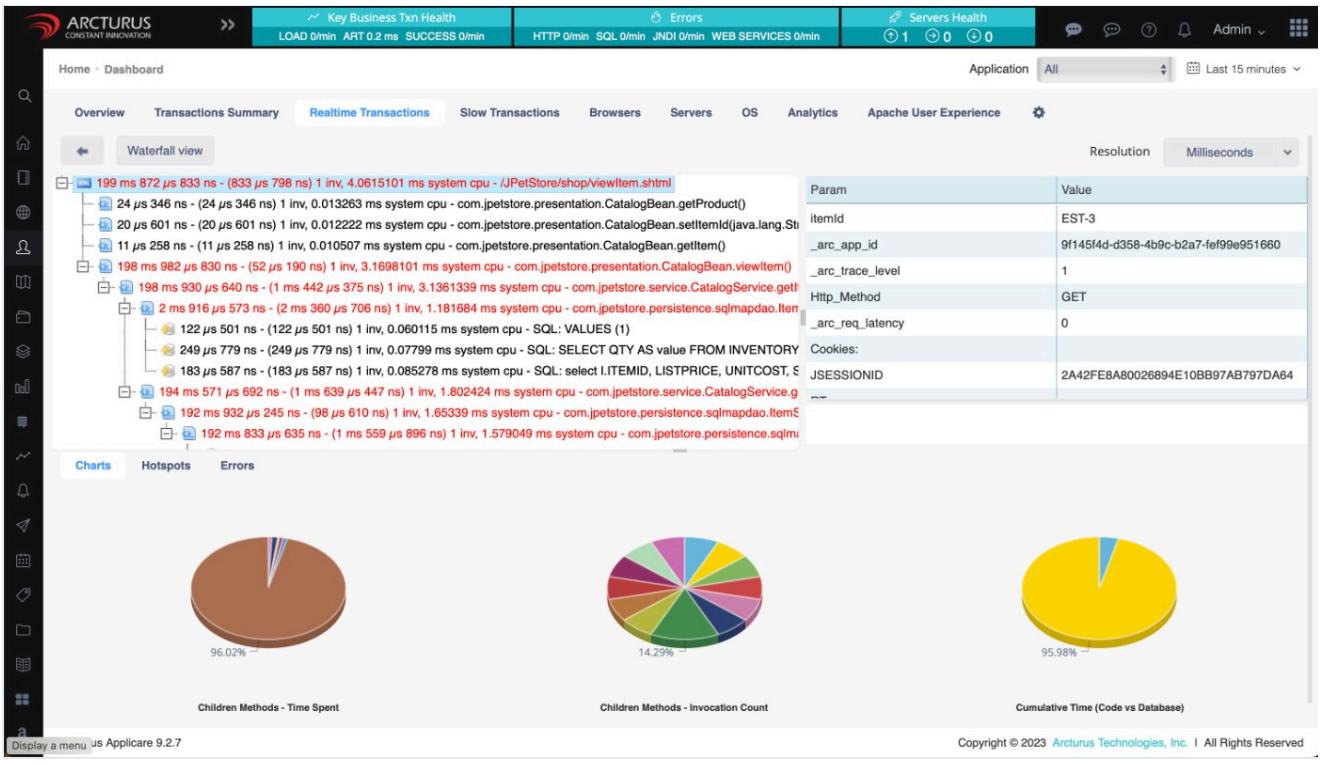
Machine Learning for Anomaly Detection

Applicare IntelliSense learns in real time from each transaction that passes through Applicare. Using machine learning Applicare automatically builds baseline and topology for each transaction type and each method with the transaction. IntelliSense leverages this machine learnt data for anomaly detections and is able to identify transactions that are slow and pin point the cause of the slowness within that transaction. IntelliSense using its machine learning does this root cause analysis down to the line of code that is causing the issue in the very first occurrence of the problem without requiring any human intervention.

Applicare auto detects slow transaction/s, identifies slowness is coming from database tier



and captures the complete stack trace (only for the slow transactions).



Predictive Forecasting

Applicare offers predictive forecasting with auto-alerts for when key metrics like Average Response Time (ART) and CPU usage approach critical thresholds.

Problematic Pattern Recognition and Alerting

Applicare has many man years for expert knowledge that is constantly working without any breaks in customer's environment. Applicare InstelliSense leveraging this knowledge auto identifies complex problematic patterns that are causing issues in the the environment e.g. detecting packets getting dropped from an ethernet card and that causing slowness in transactions due to resubmission.

We are constantly gathering these patterns from around the globe, validating and adding them to Applicare via updates. This way Applicare is getting smarter on a daily basis with a strong team of experts behind it.

Generative AI

Our team is working some of the coolest use cases of generational AI as we speak and we can't wait to show you those in our upcoming releases. Hang tight!

Monitoring with Applicare

USING APPLICARE › AIOPS

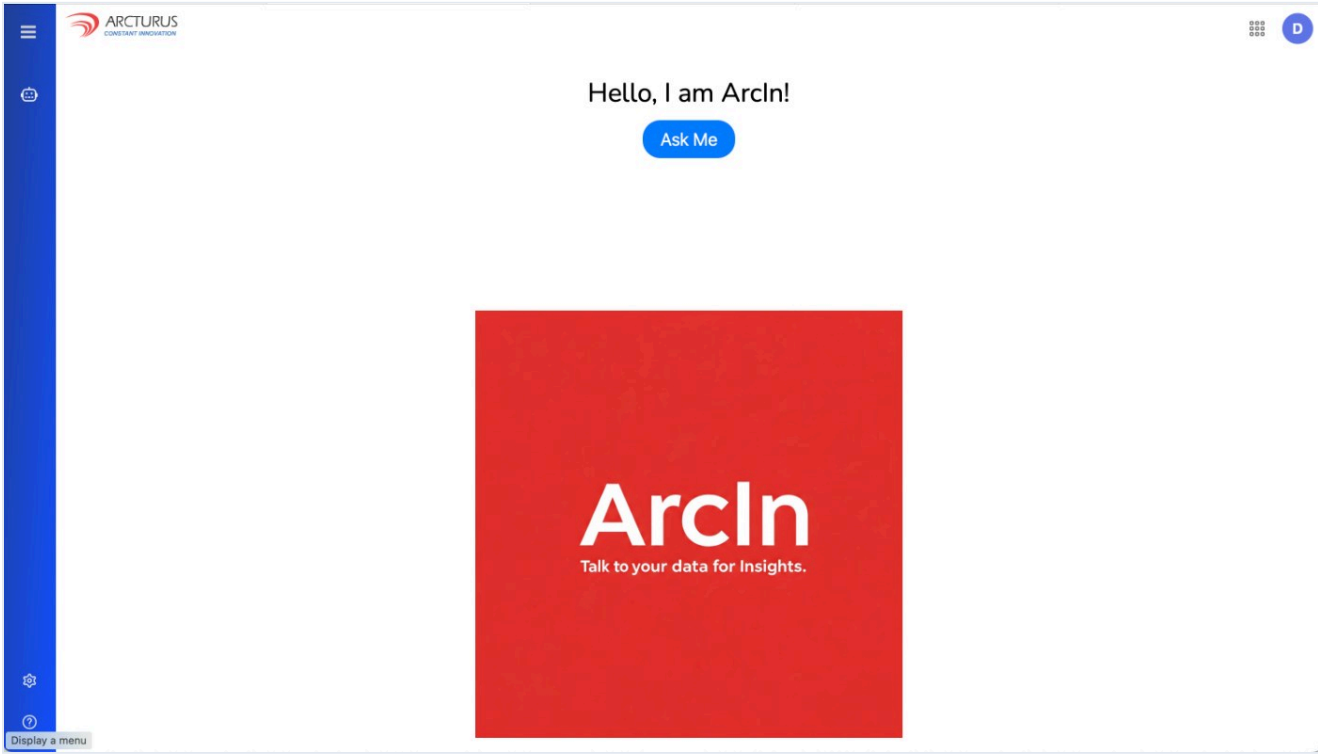
Arcturus Insights - ArcIn

Applicare 9.2 · Updated April 1, 2025 · aiops · **Pending technical review**

Talk to your data for Insights. Simply login to ArcIn, configure your database and start asking questions.

Login at

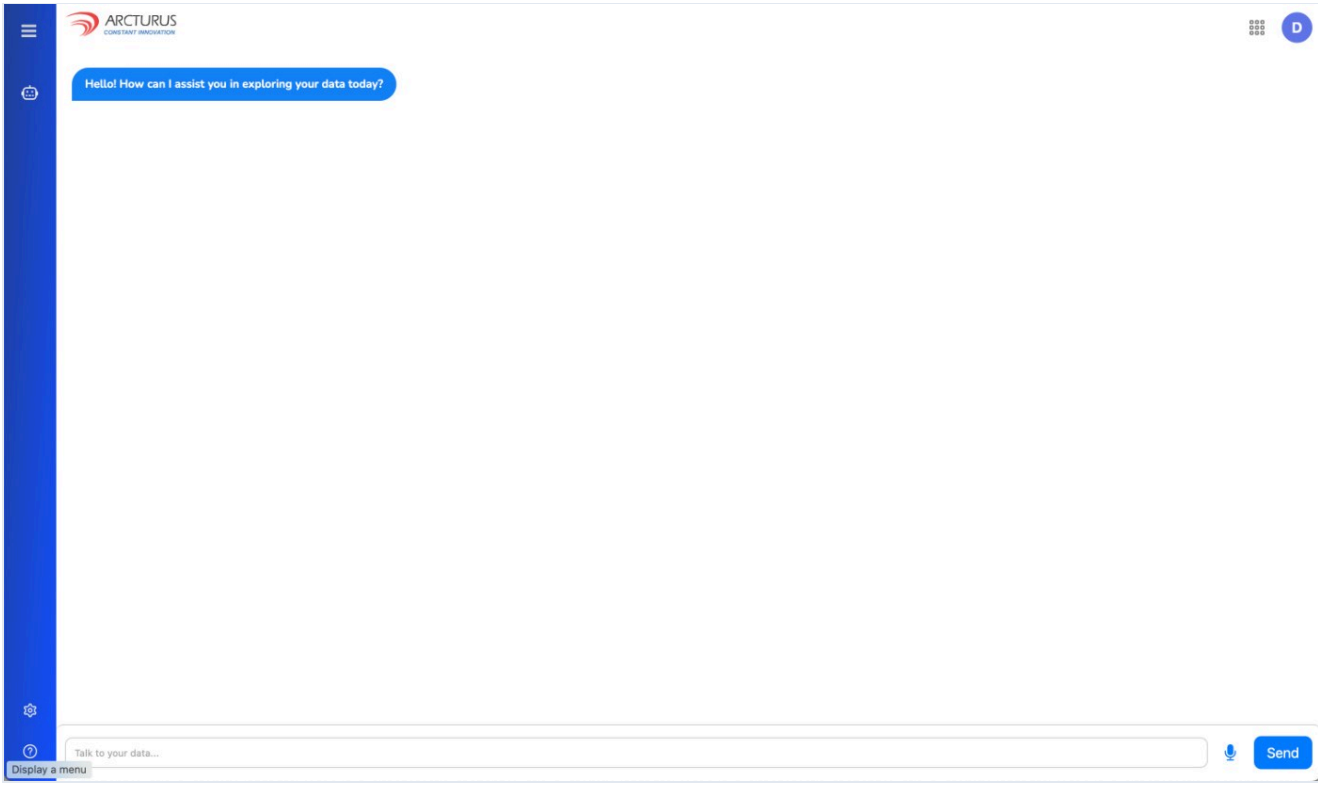
```text
https://arcin.applicare.io/
```



Configure database to talk to and get insights.

https://helpdesk.arcturustech.com/hc/en-us/articles/35357028029581-ArcIn-Database-Configuration

Simply ask questions about your data and get instant answers!


ARCTURUS
Hello! How can I assist you in exploring your data today?
Talk to your data...
Send
Display a menu

USING APPLICARE › AIOPS

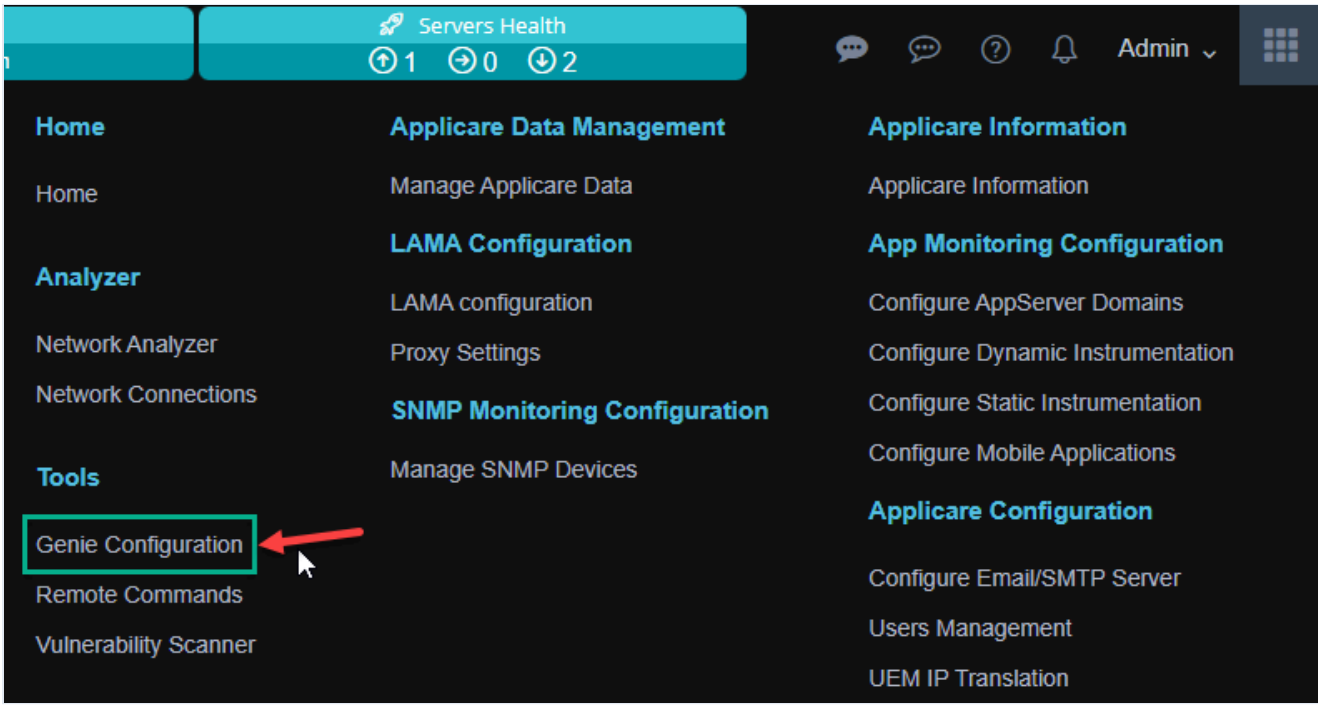
Applicare Genie

Applicare 9.2 · Updated October 23, 2024 · aiops · **Pending technical review**

One of the standout features in our Applicare is the **Genie Decision Support System**, an intelligent AI-powered assistant designed to revolutionize server management and monitoring. With Genie, users can effortlessly inquire about real-time server status, receive detailed alerts, and access critical performance data through a conversational interface. This seamless interaction allows for quick retrieval of essential information, whether tracking server health or investigating system alerts, significantly enhancing efficiency and oversight.

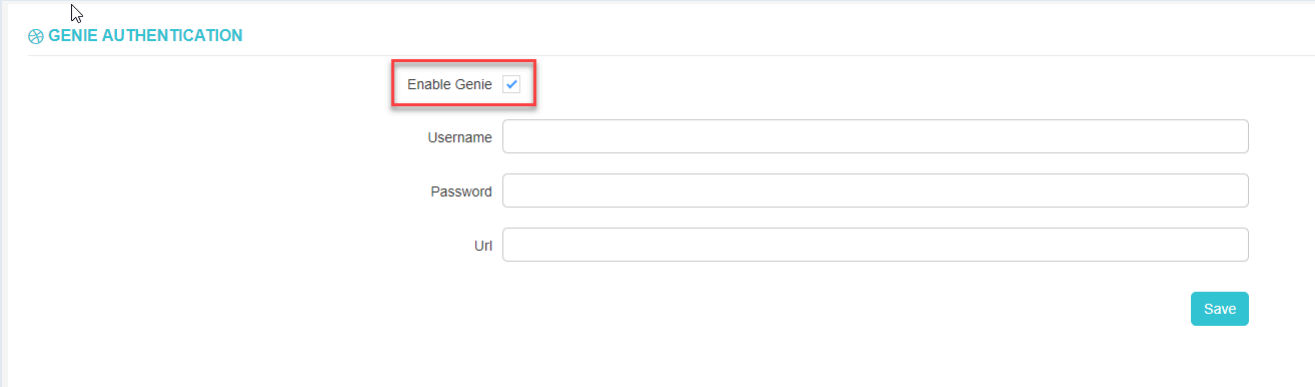
Step 1: Open Genie

1. Navigate to the Right side menu
2. Find the option labeled **Genie Configuration**. Clicking on it will take you to the configuration page where you can set up Genie.



Step 2: Enable Genie

1. Once in the configuration page, you will see a checkbox to **Enable Genie**.
2. Switch it on to activate the chat bot feature.



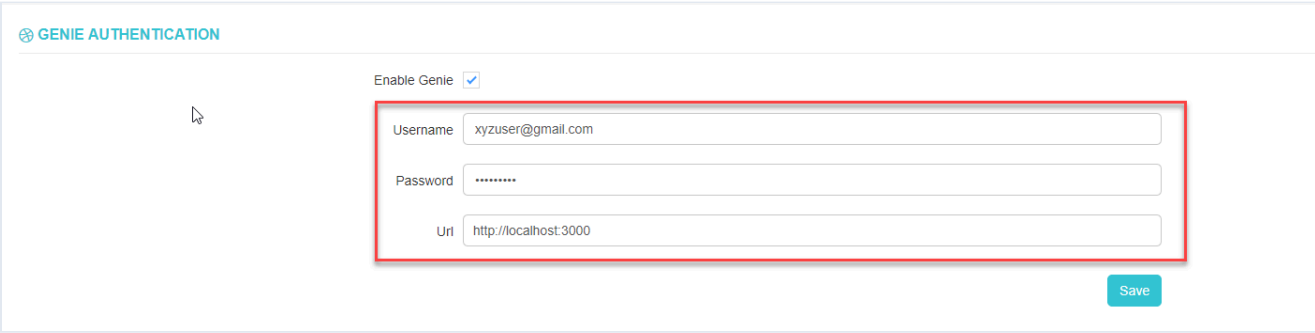
Step 3: Provide the Required Credentials

1. **Enter the URL**: You will need to provide the URL where Genie will connect. Enter the correct Genie server URL (e.g., `http://localhost:3000`) or (e.g., `https://localhost:3000`) in the field labeled **Genie Chat URL**.

   Localhost refers to the **IP address** `127.0.0.1`, which is the loopback address, commonly referred to as "localhost."

   3000 is the **port number** where the server or service is listening for incoming connections.

   2. Username and Password: In the respective fields, enter the **Genie Username** and **Genie Password**. These credentials are necessary for authenticating and establishing a secure connection with the Genie Chat Bot.



Step 4: Save the Details

1. After entering the username, password and URL, click on the **Save** button to proceed.
2. This will save the credentials and URL, ensuring that Genie is properly configured and ready to interact with your system.

Step 5: Start Using Genie

1. Once the setup is complete and saved, Genie will be ready to use. You can now access the chat bot.

USING APPLICARE › AIOPS

ArcIn Database Configuration

Applicare 9.2 · Updated March 27, 2026 · aiops · **Pending technical review**

Prerequisites: Allow the **target database** to accept connections from the IP address 20.57.1.113.

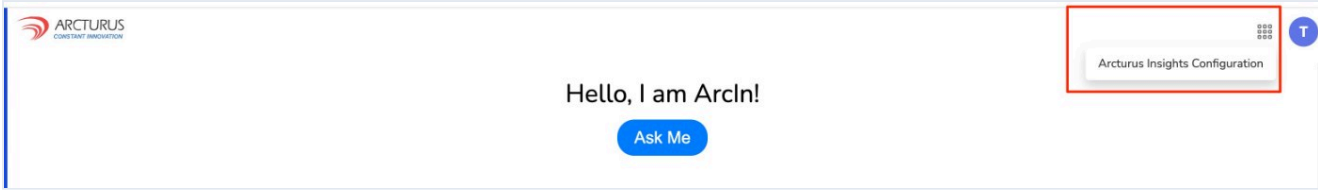
Step 1: If you don't have an Arcturus account, register here

```text
https://support.arcturustech.com/register
```

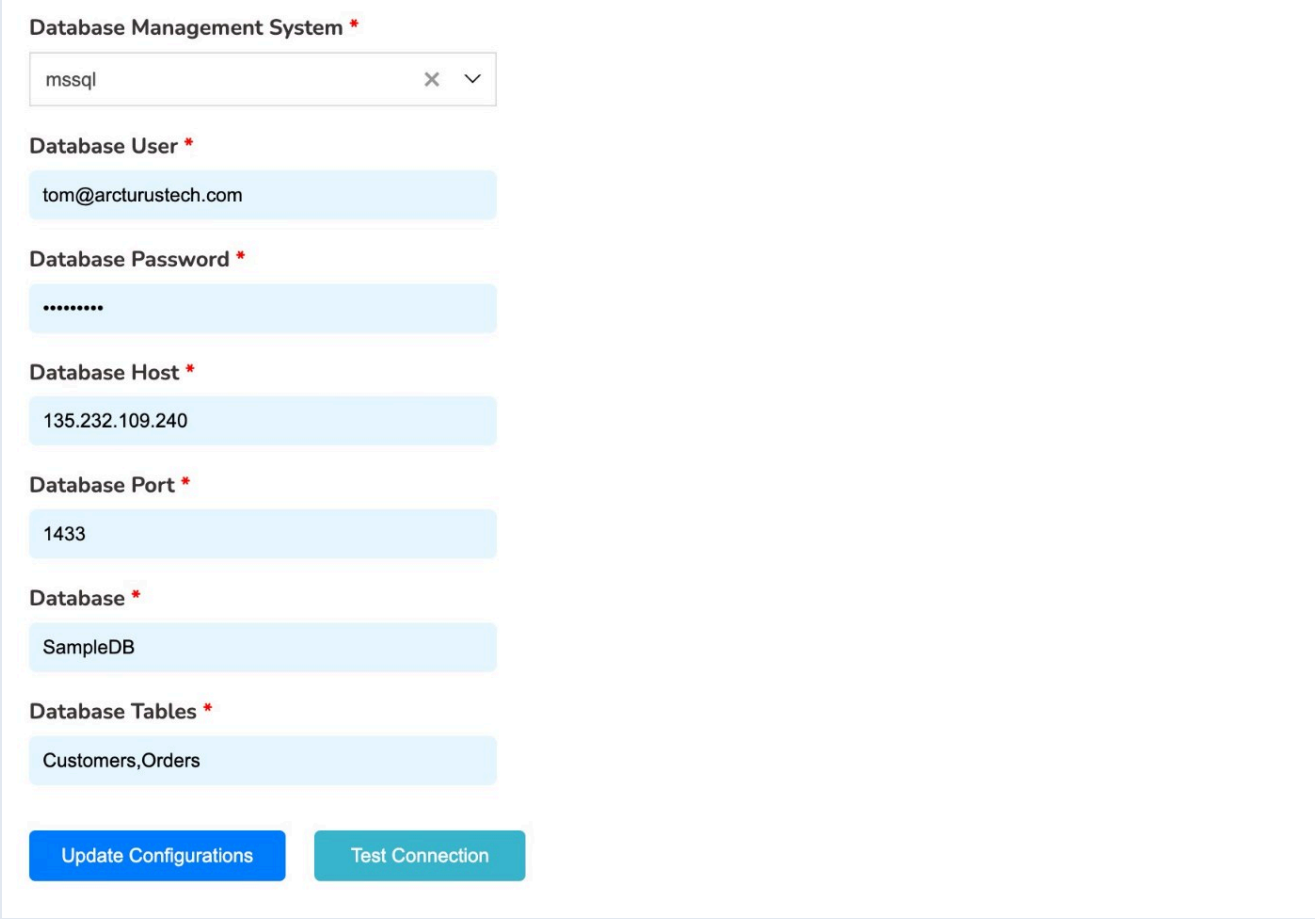
Step 2: Login our ArcIn site using the registered support site credentials.

```text
https://arcin.applicare.io/
```

Step 3: Navigate to Arcturus Insights Configuration page.



Step 4: Select your database type from the **"Database Management System"** dropdown, enter the required database details, and click **"Update configuration"** to save your settings.



Test Connection and ensure test is successful. Once you have successfully tested database connectivity, you can start asking questions. Its that simple.

USING APPLICARE › AIOPS

Set Up Predictive Analysis in Applicare on Linux

Applicare 9.2 · Updated August 22, 2024 · aiops, linux · **Pending technical review**

Step 1: Open up a command prompt or terminal then navigate to the Applicare home directory.

Step 2: Create a directory named **predictive_analysis** in the Applicare home path and set the appropriate permission for this directory.

```bash
chmod -R 777 <predictive_analysis>
```

Step 3: Navigate to the **predictive_analysis** directory and create a file named **rhel8.sh.** Paste the following contents into that file and save it. Then set the appropriate permission for this file.

```bash
#!/bin/bash
export R_VERSION=4.2.0
bash -c "$(curl -L https://rstd.io/r-install)"
ln -s /opt/R/${R_VERSION}/bin/R /usr/local/bin/R
ln -s /opt/R/${R_VERSION}/bin/Rscript /usr/local/bin/Rscript
chmod -R 777 pred.R
```

```bash
chmod +x rhel8.sh
```

Step 4: After executing the **basch script.** It should produce output similar to the example below.

```
[root@localhost predictive_analysis]# ./rhel8.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 13764  100 13764    0     0  43012      0 --:--:-- --:--:-- --:--:-- 43012
Downloading installer from https://cdn.rstudio.com/r/centos-8/pkgs/R-4.2.0-1-1.x86_64.rpm...
Downloading https://cdn.rstudio.com/r/centos-8/pkgs/R-4.2.0-1-1.x86_64.rpm...
--2022-06-23 10:16:08--  https://cdn.rstudio.com/r/centos-8/pkgs/R-4.2.0-1-1.x86_64.rpm
Resolving cdn.rstudio.com (cdn.rstudio.com)... 13.224.214.113, 13.224.214.87, 13.224.214.67, ...
Connecting to cdn.rstudio.com (cdn.rstudio.com)|13.224.214.113|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67579618 (64M) [binary/octet-stream]
Saving to: 'R-4.2.0-1-1.x86_64.rpm'

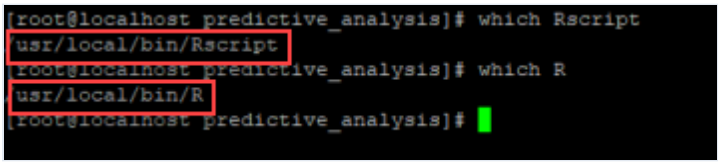
R-4.2.0-1-1.x86_64.rpm        100%[=============================================>]  64.45M  30.7MB/s    in 2.1s

2022-06-23 10:16:10 (30.7 MB/s) - 'R-4.2.0-1-1.x86_64.rpm' saved [67579618/67579618]

User install from RPM installer R-4.2.0-1-1.x86_64.rpm...
Updating package indexes...
Updating Subscription Management repositories.
Last metadata expiration check: 0:45:19 ago on Thu 23 Jun 2022 09:30:53 AM EDT.
```

Step 5: Ensure that **R** is correctly installed and located. Execute the below command it.

```
TEXT
which Rscript
```



The result should be **"usr/local/bin/Rscript"**

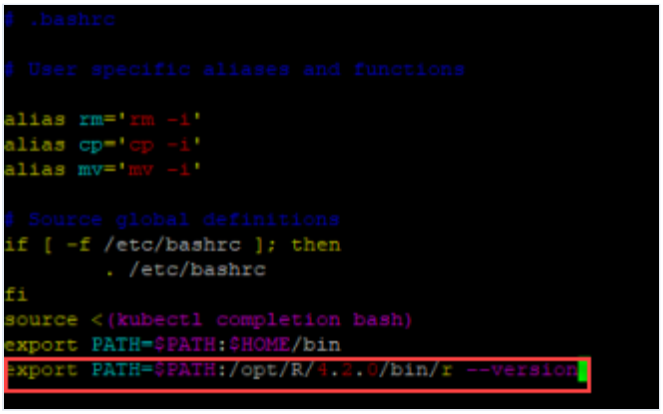
Apply step 6 ,7 and 8 only if you didn't get results on executing which R command.

Step 6: If the results differ from the expected output, modify your .bashrc file to set the correct path for **R**:

```
TEXT
vim ~/.bashrc   //OR THE PATH TO YOUR BASHRC FILE
```

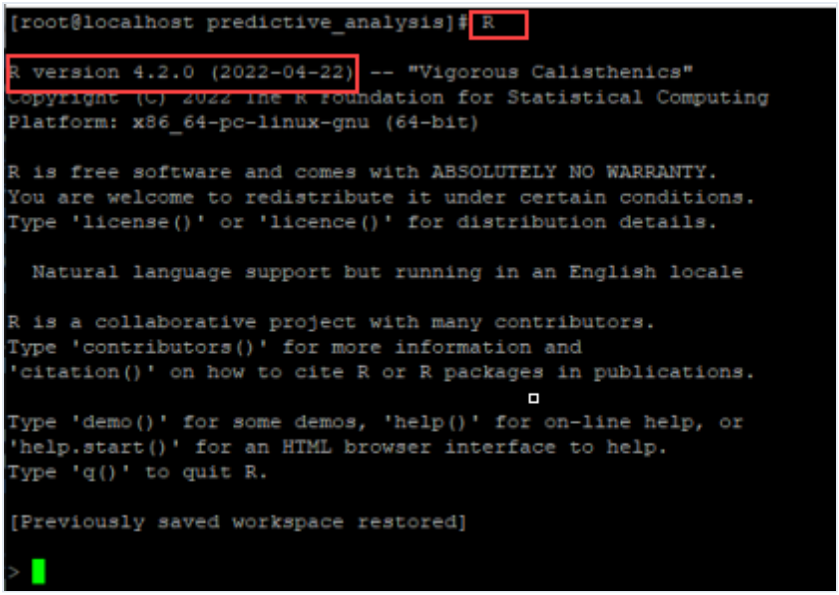
Step 7: Once you are in the bashrc file, add the following line to your file:

```
BASH
export PATH=$PATH:/opt/R/4.2.0/bin/r --version
```



Step 8: Save and exit the file

Step 9: To verify that **R** was properly installed, type **R** in the terminal. You should see the **R** interface appear in the terminal or command prompt.




```text
R should now be successfully installed and the Predictive Analysis program should be running in the background.
```

Step 10: Go to <**Applicare_home_directory**>\server\bin and edit the **catalina.sh** file.

```text
-Dapplicare.rHomePathDir=<R_installed_path>
```

Enter the directory displayed after executing **R** in **step 5,** replacing the **<R_installed_path>** placeholder with this directory path .

Example

```text
-Dapplicare.rHomePathDir=/usr/local/bin/Rscript
```

USING APPLICARE › USER EXPERIENCE MONITORING

End User Experience Monitoring (EUM)/User Experience Monitoring (UEM) Analyzer

Applicare 9.2 · Updated August 28, 2024 · **Pending technical review**

The End User Experience Analyzer delivers detailed performance insights into web applications as experienced by actual end users. Applicare agent automatically injects Java script code to JSP pages and that reports back the total time it took to load the page from the time a user clicked the link, network time, server processing and browser rendering times, browser type and user location.

Configuration

By default, Applicare scripts for collecting EUM data are served by the Applicare Controller. However, they can also be served by any Applicare Collector instance. Following JVM arguments need to be set in case a collector is serving these scripts and persisting returned data

Example URLs below assumes that '192.168.1.10' is the IP address of the Applicare Collector and '8080' is the port it is listens on. Please update the IP address and Port as needed adding these arguments and adding them to agent start.

VM Argument	Example & Description
arceum.js	-Darceum.js=`https://192.168.1.10:8080/applicare/eum/arc-eum.js`
	Utilize this parameter to specify from where the Applicare should load arc-eum.js This should be URL to applicare_agent application.
arceumprocess.js	-Darceumprocess.js=`https://192.168.1.10:8080/applicare/eum/arc-eum-process.js`
	Utilize this parameter to specify from where the Applicare should load arc-eum-procss.js This should be URL to applicare_agent application.
eum.beacon	-Deum.beacon=`https://192.168.1.10:8080/applicare/eum/beacon`
	Utilize this parameter to specify on which URL Applicare is listening for UEM data. This should be URL to applicare_agent application.
arc-nav.js	-Darc-nav.js=https://`192.168.1.10:8080/applicare`/eum/arc-nav.js
	If your web application is using servlets and you want to monitor them under UEM then you can utilize this parameter. By default Applicare does.
uem.servlets	-Duem.servlets=true
	If your web application is using servlets and you want to monitor them under UEM then you can utilize this parameter. By default Applicare does

The most crucial step is ensuring that these scripts and beacon URLs are accessible to end users. You might need to adjust firewall rules to permit external access to these URLs.

By setting the parameters mentioned, Applicare will inject JavaScript like the following into pages generated by your application's JSP/servlet when accessed by end users.

```xml
<script src="https://192.168.1.10:8080/applicare_agent/eum/arc-eum.js"
type="text/javascript"></script>

<script type="text/javascript">

BOOMR.addVar({ "server": "AdminServer","t_backend": "4","b_name":
BrowserDetect.browser,"b_ver": BrowserDetect.version,"b_os": BrowserDetect.OS });

BOOMR.init({user_ip: "192.168.1.16",beacon_url:
"https://192.168.1.10:8080/applicare_agent/eum/beacon"});

</script>
```

Next, enable the End User Experience Profiling aspect on your servers to automatically inject JavaScript into JSP pages. Follow these steps:

- Navigate to **Configuration > Configure Dynamic Instrumentation** page in Applicare console.
- On IntelliTrace Deployment tab, select Advanced Mode and click Configure button next to it.
- Move End User Experience Profiling aspect from left side list to Deployed Aspect list by selecting the aspect and clicking Add button.
- Click Apply button and make sure Agent is enabled for servers in IntelliTrace deployment tab.

Programmatic JavaScript Injection

If auto-instrumentation is not available or feasible, you can still implement end user monitoring by manually injecting the necessary scripts into your pages.

Applicare provides an API that allows you to insert the required JavaScript just before the "</body>" tag of your pages.

For instance, you can adjust your application's template to include the following for manual JavaScript injection:

```
... existing template code ...<%= com.arcturus.aop.shared.RUMManager.injectJS() %></body>
```

Additionally, ensure that "<Applicare Installation Dir>/applicare-aspects.jar" is included in your class path during development.

Manual JavaScript Injection for Static Pages

For end user experience monitoring of static pages, Applicare EUM scripts can be manually injected.

URL Exclusion & Inclusion

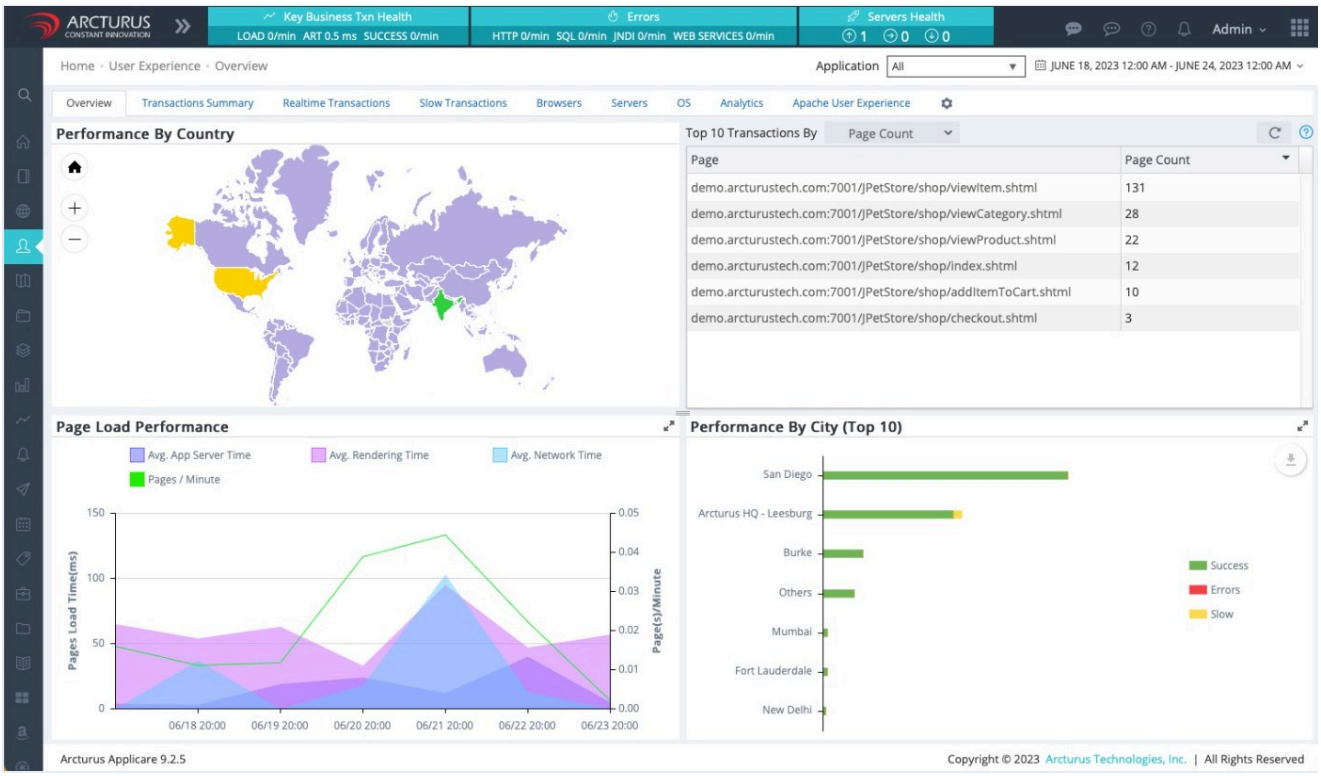
To manage URL monitoring in UEM: Add a URL from the HTTP Analyzer to the UEMURLExclusions.txt file to exclude it from monitoring.

Add a URL from the HTTP Analyzer to the UEMURLInclusions.txt file to include it in monitoring.

Analyzer

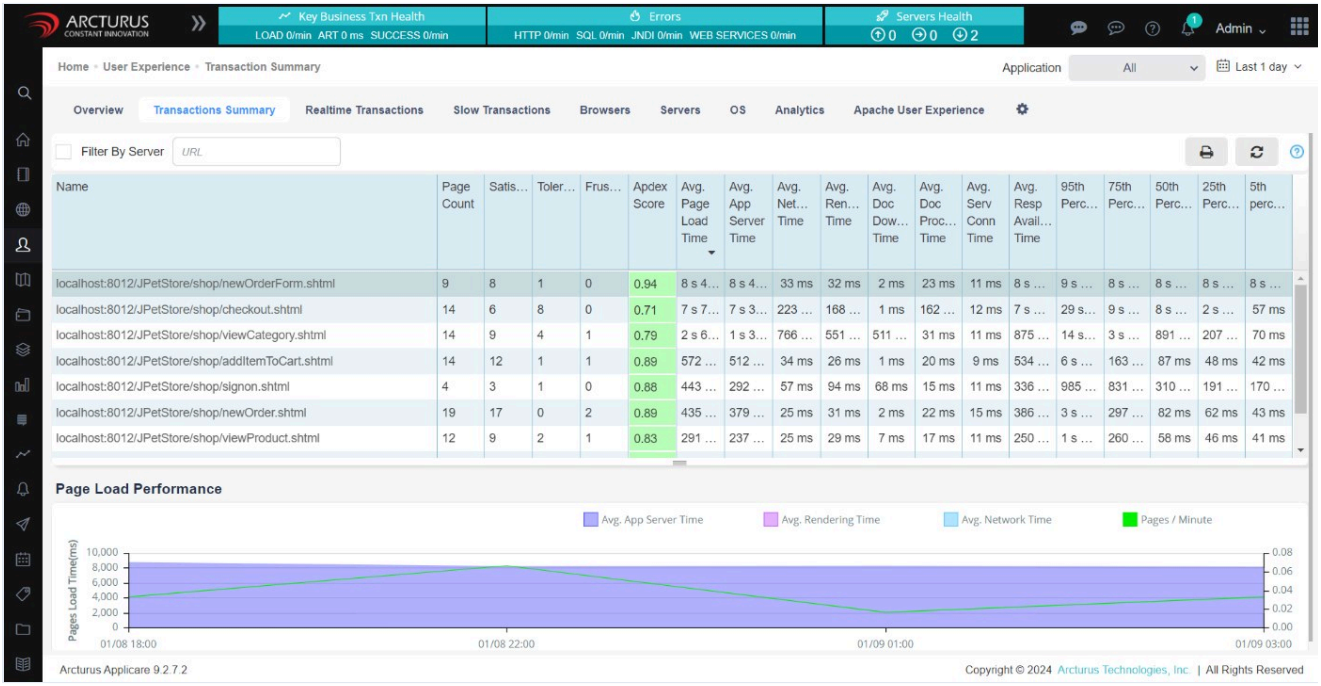
The End User Experience Analyzer presents historical performance data across various aspects in separate tabs. The Real-time Transactions tab updates immediately as users interact with your application, while data on other tabs may take a few minutes to appear.

Overview



The world map on the dashboard shows the performance breakdown for each country. Mouse over a country to see performance summary data for the country. On the right you can view top transactions by different criteria such as page count, average page load time etc... The chart below shows the performance data over the selected time period and if you click on a country in the map, the chart will display the performance data for the country as shown in the image below.

Transaction Summary



The Transaction Summary view shows performance statistics for transactions over a selected time period, with a timeline chart at the bottom. Clicking on a transaction will refresh the chart to display performance data for the selected transaction.

The **Apdex score** quantifies user satisfaction with web application performance on a scale from 0 to 1, where 1 signifies perfect satisfaction and 0 denotes complete dissatisfaction. It condenses various response times into a single metric.

By default, Applicare will consider the average response time of any transactions as its **Desired response time(T).**

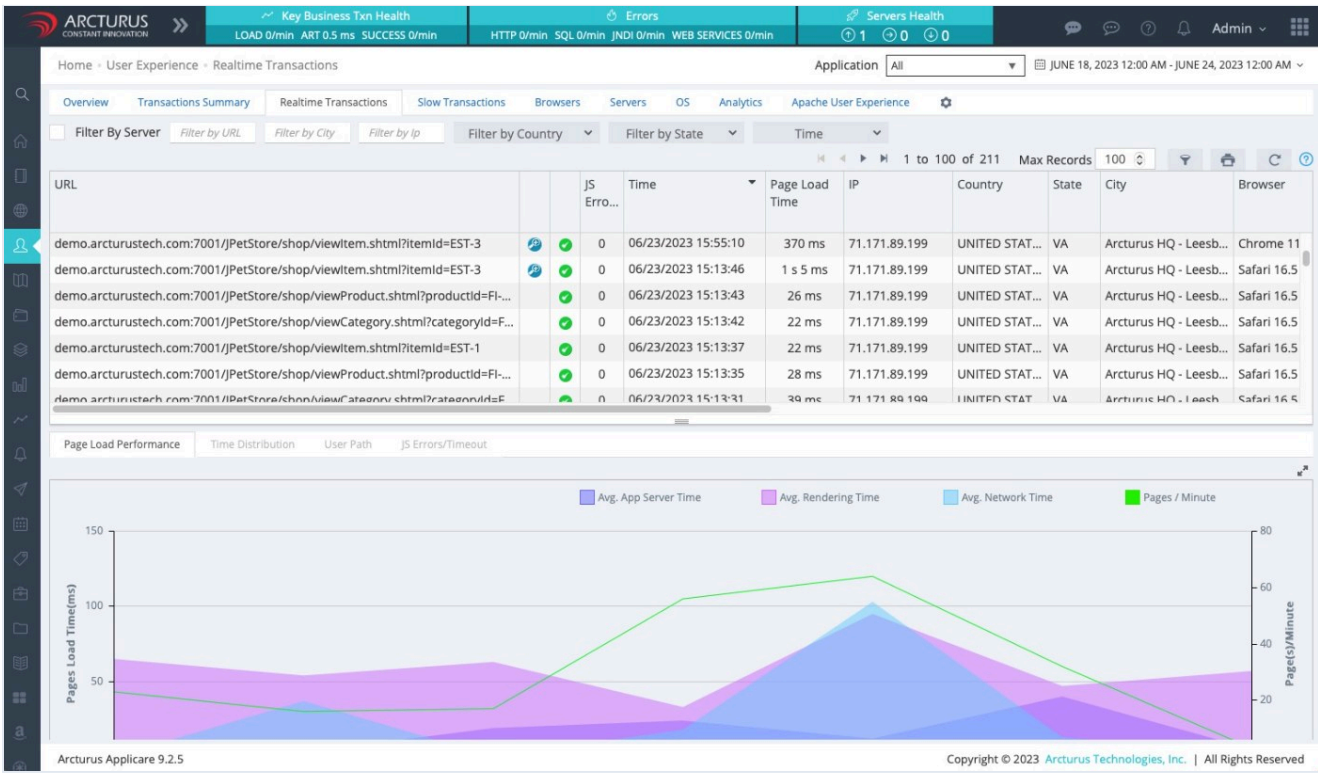
Transactions will be classified as follows:

- **Satisfied**: Response times ≤ T.
- **Tolerating**: Response times between T and 4T.
- **Frustrated**: Response times > 4T.

Apdex Score is calculated with the formula:

Apdex Score = (Satisfied Count + (Tolerating Count / 2)) / Total Count

Realtime Transactions



The Realtime view displays the last n number individual transactions recorded. Along with performance stats you can view location and browser information of users accessing your application. This view doesn't show any transactions older than a day.

In case the Applicare Controller/Collector collecting EUM data is behind a proxy, firewall or load balancer, make sure that the proxy, firewall or load balancing server forwards the client requests with "X-Forwarded-For" header parameter. If this parameter is not included in the HTTP request then Applicare may show the IP address of proxy, firewall or load balancing server in End User Experience Analyzer.

Below are the details on monitored data shown for each transaction

- **Resource Fetch Time:** The time it takes for a web browser to fetch a resource, such as an image, CSS file, or JavaScript file. This includes the time it takes to:
  - **DNS lookup:** The time it takes to resolve the domain name of the resource.
  - **TCP connection establishment:** The time it takes to establish a connection to the server that hosts the resource.
  - **TLS handshake:** The time it takes to establish a secure connection to the server, if the resource is hosted over HTTPS.
  - **Request:** The time it takes to send the request for the resource to the server.
  - **Response:** The time it takes to receive the response from the server, including the time it takes to download the resource.

    Resource Fetch Time is an important metric for measuring the performance of a web page. A short Resource Fetch Time means that the page will load more quickly, which can improve the user experience.

There are a number of factors that can affect Resource Fetch Time, including:

- **The size of the resource:** Larger resources will take longer to fetch.
- **The speed of the user's internet connection:** A slower internet connection will result in a longer Resource Fetch Time.
- **The performance of the browser:** A faster browser will be able to fetch resources more quickly.
- **The location of the server:** The closer the server is to the user, the shorter the Resource Fetch Time will be.

Here are some tips for improving Resource Fetch Time:

- **Use minified and compressed resources:** Minified and compressed resources are smaller, which means they will take less time to fetch.
- **Use a CDN:** A CDN (Content Delivery Network) can deliver resources from servers that are closer to the user, which can improve Resource Fetch Time.
- **Use a caching mechanism:** A caching mechanism can store resources that have already been fetched, which can reduce the need to fetch them again.

- **Document Download Time:** The time it takes to download a document from a web server to a user's device. This includes the time it takes to:
  - **DNS lookup:** The time it takes to resolve the domain name of the web server.
  - **TCP connection establishment:** The time it takes to establish a connection to the web server.
  - **TLS handshake:** The time it takes to establish a secure connection to the web server, if the request is made over HTTPS.
  - **Request:** The time it takes to send the request for the document to the web server.
  - **Response:** The time it takes to receive the response from the web server, including the time it takes to download the document.
- **Document Processing Time:** The time it takes for the browser to finish parsing the HTML code of a web page and construct the Document Object Model (DOM). The DOM is a hierarchical representation of the page's elements, which can be manipulated by JavaScript.

  Factors Affecting Document Processing Time:
  - **HTML Code Size**: Larger HTML files take longer to parse.
  - **HTML Code Complexity**: More complex HTML structures require additional parsing time.
  - **Browser Performance**: Faster browsers can parse HTML more quickly.
  - **JavaScript Presence**: JavaScript files can increase document processing time.

  Tips for Improving Document Processing Time:
  - **Use minified and compressed HTML code-** Minified and compressed HTML code is smaller, which means it will take less time to parse.
  - **Use a CDN-** A CDN (Content Delivery Network) can deliver HTML code from servers that are closer to the user, which can improve document processing time.
  - **Use a caching mechanism-** A caching mechanism can store HTML code that has already been parsed, which can reduce the need to parse it again.

    - **Optimize the JavaScript files-** Optimizing JavaScript files can improve their performance and reduce the document processing time.

- **Time to First Byte (TTFB)-** A metric that measures the time it takes for a web browser to receive the first byte of a response from a web server. This includes the time it takes to:
    - **DNS lookup-** The time it takes to resolve the domain name of the web server.
    - **TCP connection establishment-** The time it takes to establish a connection to the web server.
    - **TLS handshake-** The time it takes to establish a secure connection to the web server, if the request is made over HTTPS.
    - **Request-** The time it takes to send the request for the resource to the web server.
- **DOM Ready Time:**

  The duration it takes for a web browser to parse the HTML code and build the Document Object Model (DOM) of a web page. The DOM represents the page's elements in a hierarchical structure, which can be manipulated by JavaScript. The DOM Ready Time event is triggered when the DOM is ready, though the page may still be loading images, CSS, and other resources. This metric indicates how quickly a page becomes interactive.

  Factors affecting DOM Ready Time include HTML code size and complexity, the user's internet speed, and browser performance.

  Improving DOM Ready Time can enhance user experience by making the page interactive more quickly. Here are some tips:

    - Minify and compress HTML, CSS, and JavaScript files.
    - Reduce the number of large images or resources.
    - Utilize a Content Delivery Network (CDN) for static resources.
    - Implement caching to store frequently accessed resources.
- **Network Time-** The duration for a web browser to retrieve all resources required to load a web page, including:
    - **DNS lookup-** Time to resolve the domain name of the resources.
    - **TCP connection establishment-** Time to set up a connection with the resource servers.
    - **TLS handshake-** Time to establish a secure connection for HTTPS resources.
    - **Request-** Time to send requests to the servers for the resources.
    - **Response-** Time to receive responses and download the resources.
- **Rendering Time-** The time needed for a web browser to display a web page, including:
    - **Parse the HTML code:** Time to parse HTML code and build the DOM.
    - **Load the CSS and JavaScript files:** Time to load CSS and JavaScript files for styling and interaction.
    - **Render the images and other resources:** Time to render images and other resources used on the page.

There are a several reasons why DOM ready time might be high while document processing time can remains low.

- **The HTML code is large and complex.** A large and complex HTML code can take longer to parse, even if it does not contain any JavaScript. This is because the browser has to process more information to create the DOM.
- **The web page uses a lot of third-party resources.** Third-party resources, such as images, CSS, and JavaScript files, can also add to the DOM ready time. This is because the browser has to wait for these resources to load before it can create the DOM.
- **The web page is using a slow web server.** If the web server is slow, it can take longer for the browser to download the HTML code and third-party resources. This can also increase the DOM ready time.

If your DOM ready time is high, but your document processing time is low, then you may need to optimize your HTML code to make it smaller and easier to parse.

If you access Realtime Transaction and double click on any URL you may encounter a message "Trace not available..."

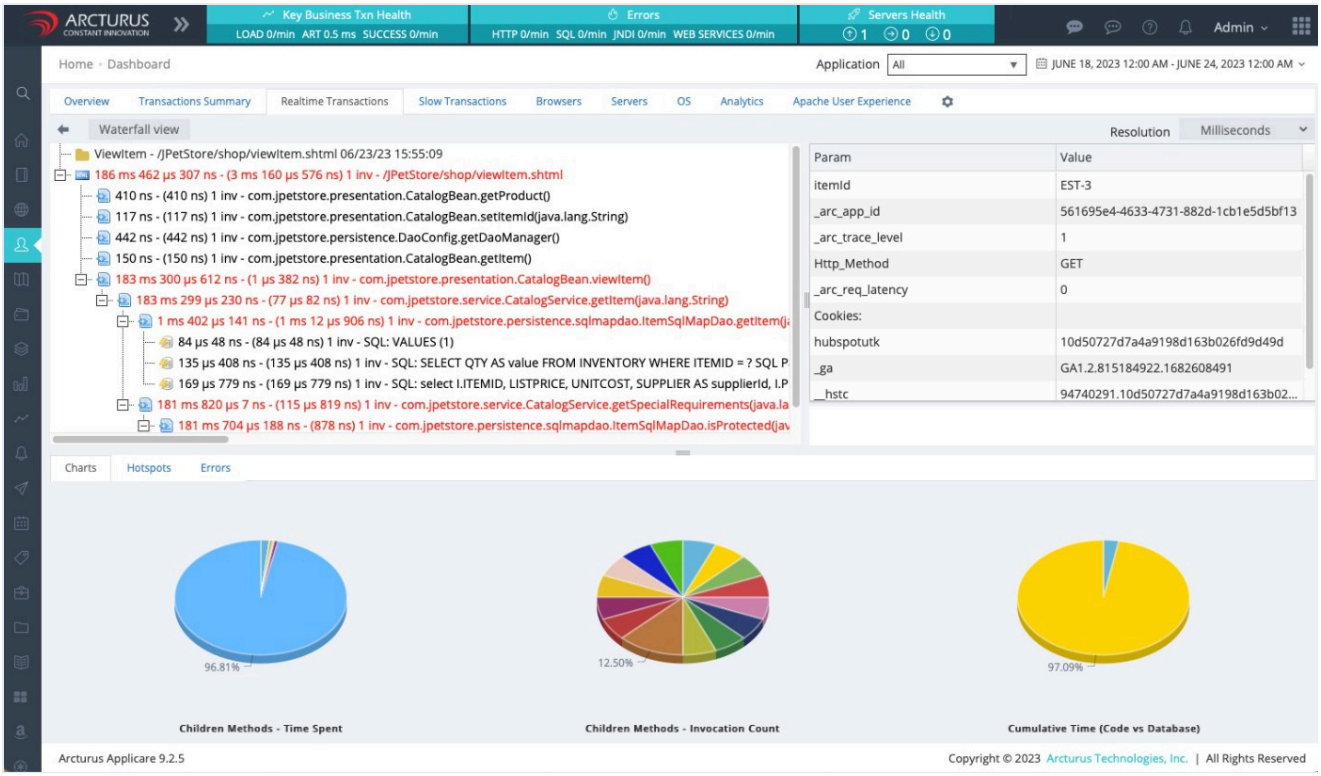
Activating tracing on End User Experience

1. Navigate to **Server Analyzer > IntelliTrace > HTTP**
2. Right-click the URL you want to trace in detail tracing and select "Add as a Key Business Transaction".
3. Add the URL to Key Business Transaction. Alternatively, go to directly visit **Business Transactions > Configure Key Business Transactions** and include the URL to Business Transaction Monitoring.
4. Go to **Profiling > Configure Business Transactions > Transaction Monitoring**.
5. Activate Per-Transaction Monitoring.
6. Activate Transaction Tracing.

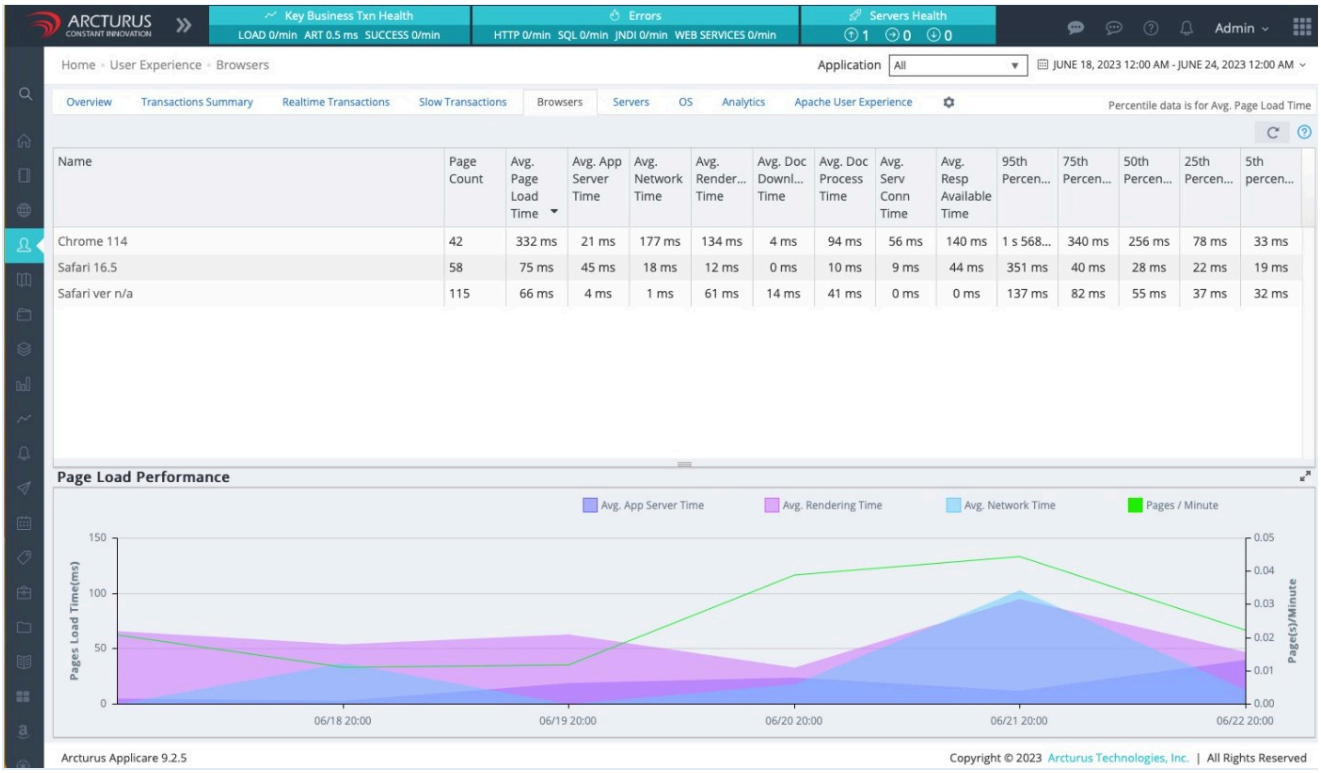
Once you have followed the above steps, restart the agent jvm.

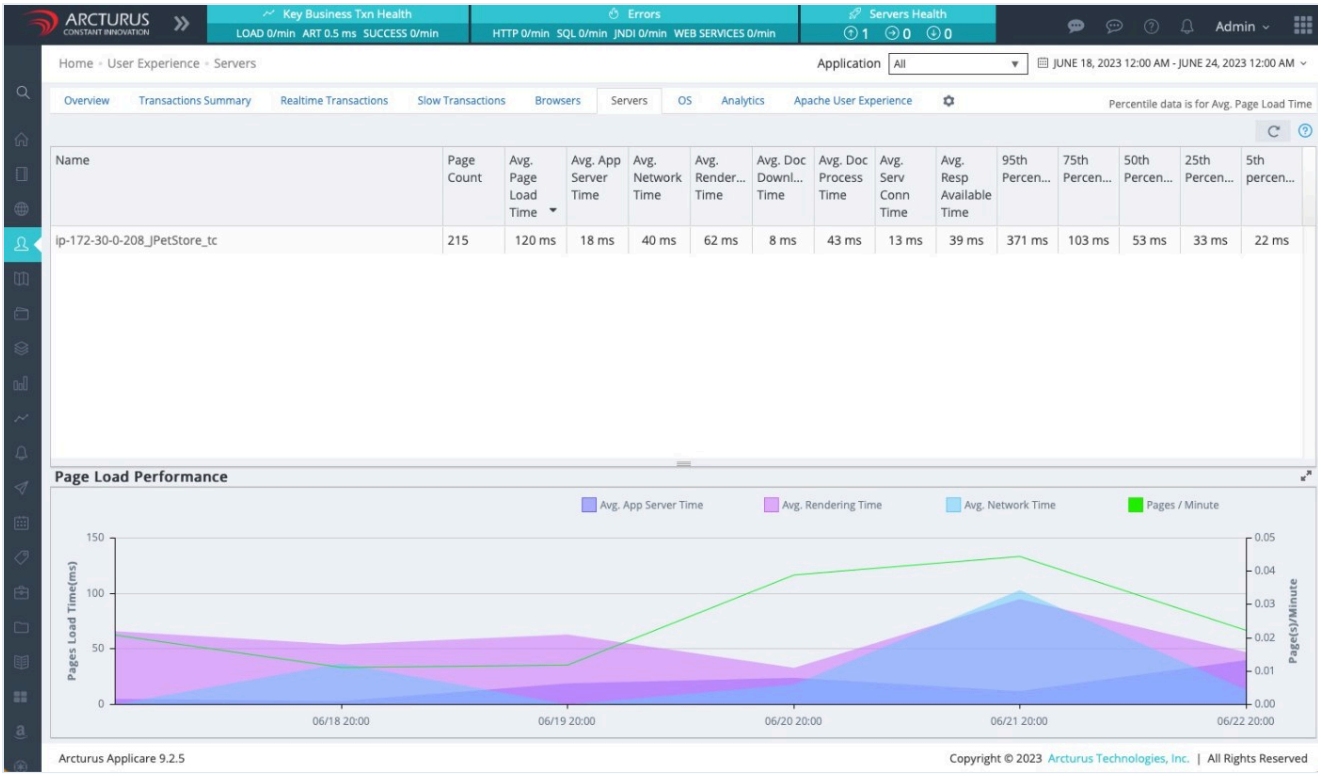
Now you can visit the UEM > End User Experience Analyzer > Realtime Transactions and double click on the URL.

If you have added the URL as KBT, detailed trace will be displayed as shown in the following image.



Browser and Backend Server Views





The Browsers and Servers tabs are similar to the Transaction Summary view but provide performance data segmented by client browser type and backend servers handling transactions.

End Client Processes Monitoring

End Client Process Monitoring tracks the running processes and resources utilized by each process on the end client machine where browser is running. This helps identify if the slowness is caused by some process running on End User's computer. End Client Process Monitoring is only supported on Windows and Internet Explorer (IE) end clients. End Client Process Monitoring requires permissions from the end user (or centrally configured by administrators) to allow running 'Microsoft Script Runtime'. If these permissions are not centrally configured then users are prompted on first visit to the pages with end client process monitoring enabled.

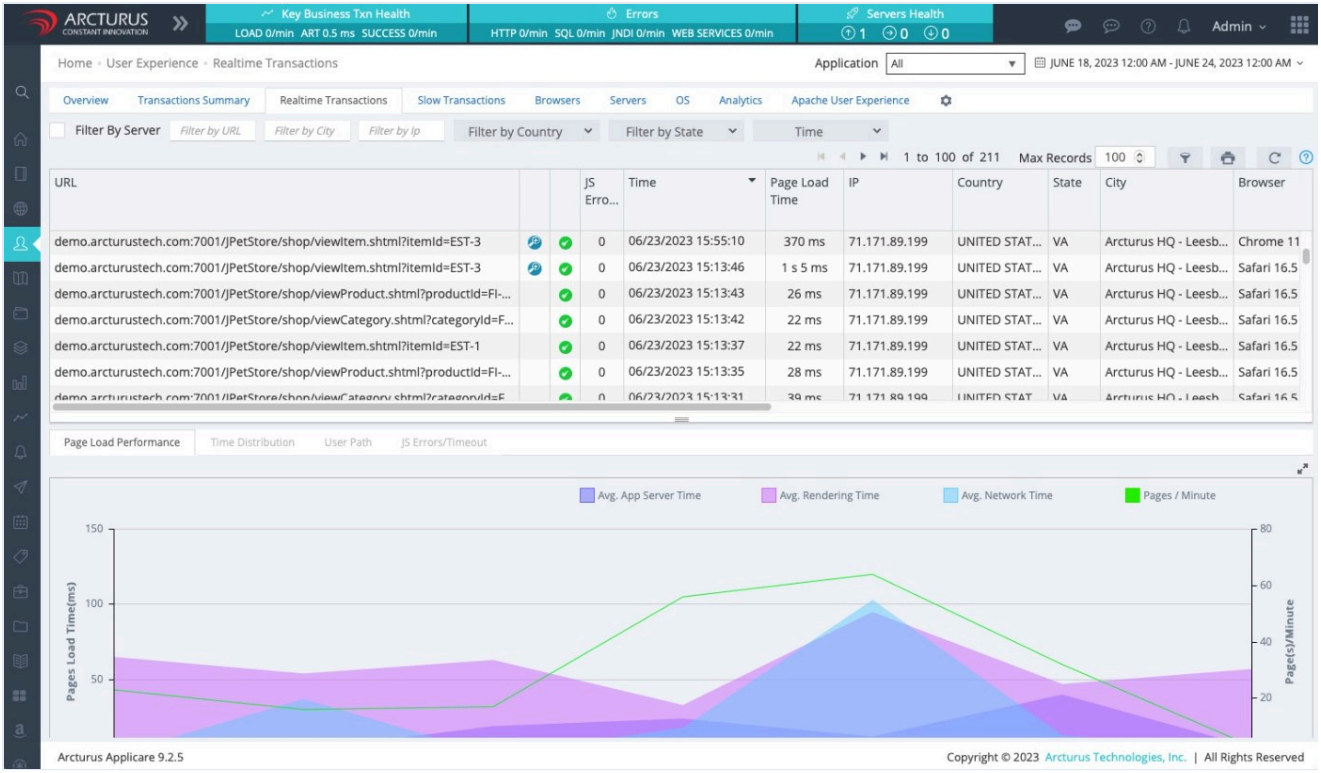
End Client Process Monitoring can be enabled for slow calls or all calls.

USING APPLICARE › USER EXPERIENCE MONITORING

EUM Real-Time Transaction Monitoring

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The real-time view shows the last n number of individual transaction recorded. Along with performance stats you can view location and browser information of users accessing your application. This view doesn't show any transaction older than a day.



If the Applicare Controller/Collector gathering EUM data is situated a proxy, firewall or load balancer, ensure that these intermediaries forward client requests with the "X-Forwarded-For" header parameter. If this parameter is not included in the HTTP request then Applicare may show the IP address of proxy, firewall or load balancing server in End User Experience Analyzer.

Below are the details on monitored data shown for each transaction

- **Resource Fetch Time:** The duration required for a web browser to retrieve a resource, such as an image, CSS file, or JavaScript file. This includes the time it takes to:
  - **DNS lookup:** The time required to resolve the domain name of the resource.
  - **TCP connection establishment:** The time required to establish a connection to the server hosting the resource.
  - **TLS handshake:** The time required to establish a secure connection with the server, if the resource is hosted over HTTPS.
  - **Request:** The time required to send the request for the resource to the server.

- **Response:** The time it takes to receive a response from the server, including duration required to download the resource is known as Resource Fetch Time. This metric is crucial for evaluating web page performance, as a shorter Resource Fetch Time leads to faster page loading and enhance user experience.

  Several factors can influence Resource Fetch Time, including:

  - **Resource Size:** Larger resources will take longer to fetch.
  - **Internet Connection Speed:** A slower internet connection will lead to a longer Resource Fetch Time.
  - **Browser Performance:** A faster browser can retrieve resources more quickly.
  - **Server Location:** The closer the server is to the user, the shorter the Resource Fetch Time will be.

  Here are some tips for improving Resource Fetch Time:

  - **Use Minified and Compressed Resources:** Minified and compressed resources are smaller, resulting in fast fetch time.
  - **Use a CDN:** A CDN (Content Delivery Network) can deliver resources from servers that are closer to the user, which can improve Resource Fetch Time.
  - **Implement Caching:** A caching mechanism can store resources that have already been retrieved, reducing need to fetch them again.

- **Document Download Time:** The duration required to download a document from a web server to a user's device. This includes the time it takes to:
  - **DNS Lookup:** The time needed to resolve the domain name of the web server.
  - **TCP Connection Establishment:** The time required to establish a connection to the web server.
  - **TLS Handshake:** The time required to establish a secure connection with the web server, if the request is made over HTTPS.
  - **Request:** The time required to send the request for the document to the web server.
  - **Response:** The time needed to receive the response from the web server, including the time required to download the document.
- **Document Processing Time:** The time it takes for the browser to finish parsing the HTML code of a web page and construct the Document Object Model (DOM). The DOM is a hierarchical representation of the page's elements, which can be manipulated by JavaScript.

  Document processing time can be affected by a number of factors, including:

  - **Size of the HTML Code:** Larger HTML files will take longer to parse.
  - **Complexity of the HTML Code:** More complex HTML files will take longer to parse.
  - **Browser Performance:** Faster browsers will be able to parse HTML code more efficiently.
  - **The presence of JavaScript:** JavaScript files can increase the document processing time.

  Here are some tips for improving document processing time:

  - **Use Minified and Compressed HTML Code:** Minified and compressed HTML code is smaller, which means it will take less time to parse.

    - **Use a CDN:** A CDN (Content Delivery Network) can deliver HTML code from servers that are closer to the user, which can improve document processing time.
    - **Implement Caching:** A caching mechanism can store HTML code that has already been parsed, which can reduce the need to parse it again.
    - **Optimize JavaScript files:** Optimizing JavaScript files can improve their performance and reduce the document processing time.

- **Time to First Byte (TTFB):** A metric that measures the time it takes for a web browser to receive the first byte of a response from a web server. This includes the time it takes to:
    - **DNS lookup:** The time it takes to resolve the domain name of the web server.
    - **TCP connection establishment:** The time it takes to establish a connection to the web server.
    - **TLS handshake:** The time it takes to establish a secure connection to the web server, if the request is made over HTTPS.
    - **Request:** The time it takes to send the request for the resource to the web server.
- **DOM Ready Time:** The time it takes for a web browser to finish parsing the HTML code of a web page and construct the Document Object Model (DOM). The DOM is a hierarchical representation of the page's elements, which can be manipulated by Javascript.

  The DOM Ready Time event is fired when the DOM is ready, but the page is not necessarily fully loaded yet. This means that images, CSS, and other resources may still be loading. However, the DOM Ready Time event is a good indication of how long it takes for a web page to become interactive.

  There are a number of factors that can affect DOM Ready Time, including the size and complexity of the HTML code, the speed of the user's internet connection, and the performance of the browser.

  DOM Ready Time is an important metric for measuring the performance of a web page. A short DOM Ready Time means that the page will be interactive more quickly, which can improve the user experience.

  Here are some tips for improving DOM Ready Time:

    - Use minified and compressed HTML, CSS, and JavaScript files.
    - Avoid using too many images or other large resources.
    - Use a CDN to deliver static resources.
    - Use a caching mechanism to store frequently accessed resources.
- **Network Time:** The time it takes for a web browser to fetch all of the resources that are needed to load a web page. This includes the time it takes to:
    - **DNS lookup:** The time it takes to resolve the domain name of the resources.
    - **TCP Connection Establishment:** The time it takes to establish a connection to the servers that host the resources.
    - **TLS Handshake:** The time required to establish a secure connection to the servers, if the resources are hosted over HTTPS.
    - **Request:** The time it takes to send the requests for the resources to the servers.
    - **Response:** The time it takes to receive the responses from the servers, including the time it takes to download the resources.

- **Rendering Time:** The time it takes for a web browser to render a web page. This includes the time it takes to:
  - **Parsing the HTML Code:** The time it takes to parse the HTML code of the web page and construct the DOM.
  - **Loading CSS and JavaScript Files:** The time it takes to load the CSS and JavaScript files that are needed to style and interact with the web page.
  - **Rendering Images and Other Resources:** The time required to render the images and other resources that are used on the web page.

Several factors can contribute to high DOM ready time despite a low document processing time.

- **Large and Complex HTML Code:** A large and complex HTML code can take longer to parse, even if it does not contain any JavaScript. This is because the browser has to process more information to create the DOM.
- **Heavy Use of Third-Party Resources:** Third-party resources, such as images, CSS, and JavaScript files, can also add to the DOM ready time. This is because the browser has to wait for these resources to load before it can create the DOM.
- **Slow Web Server:** If the web server is slow, it can take longer for the browser to download the HTML code and third-party resources. This can also increase the DOM ready time.

If you notice a high DOM ready but a low document processing time, consider optimizing your HTML code to reduce its size and complexity, making it easier and quick to parse.

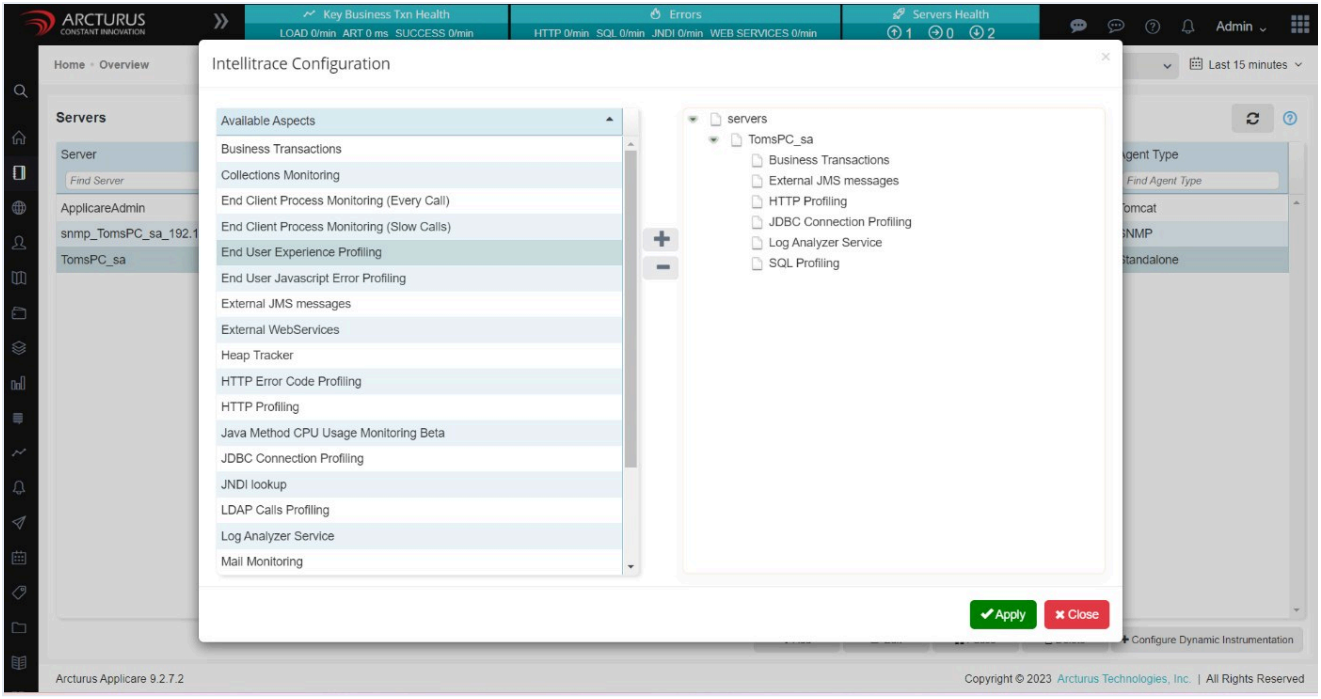
Monitoring with Applicare

USING APPLICARE › USER EXPERIENCE MONITORING

Applicare RUM/EUM Integration Script

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

In Applicare, you can activate end user monitoring by turning on the "End User Experience Profiling" feature from the overview screen. Once this feature is enabled, Applicare will automatically inject its RUM script into the web pages of the monitored applications behind the scenes.



Alternatively, you can manually inject this script into your application's web pages and redeploy the code to collect EUM data. Refer this link for Manual Applicare EUM script injection.

Applicare will inject two JavaScripts files named **arc-eum.js** and **arc-nav.js**.

The **arc-nav.js** script utilizes the Navigation API to measure various parameters including Document download time, Document processing time, Server connection time, Response availability time, Rendering time, DOM readiness time and Resource fetch time.

The **arc-eum.js** script employs BoomerangJS, an open-source middleware, to measure Page load time, Network time and browser details.

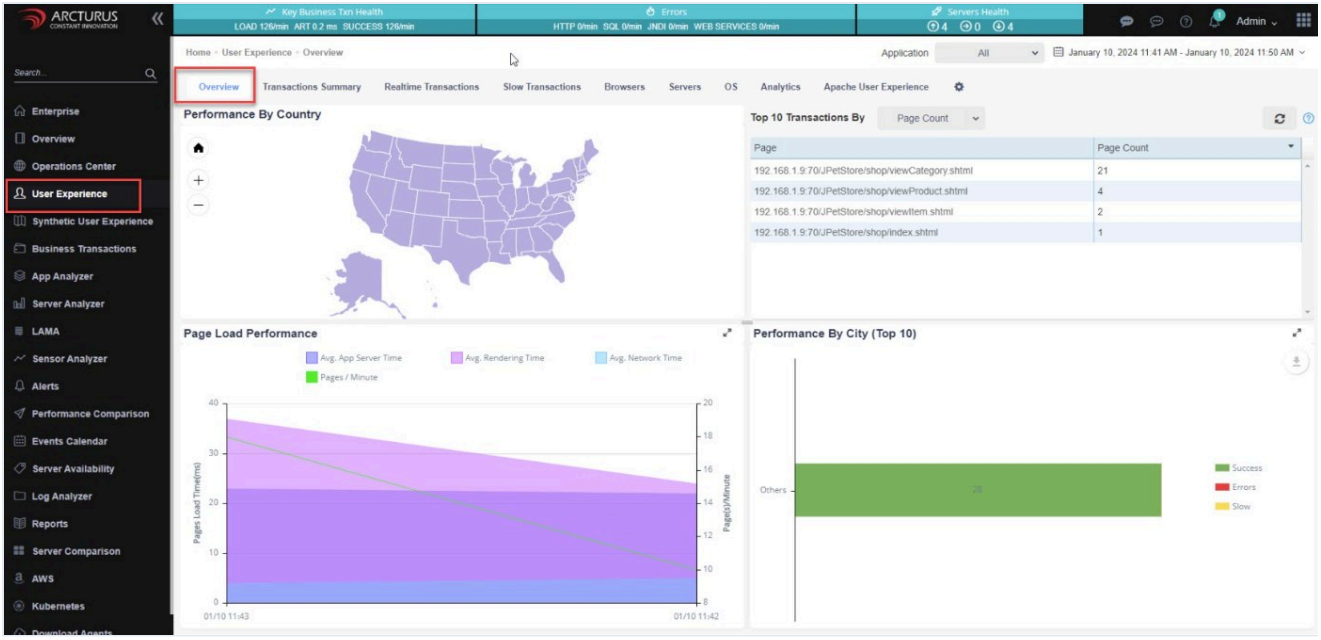
Monitoring with Applicare

USING APPLICARE › USER EXPERIENCE MONITORING

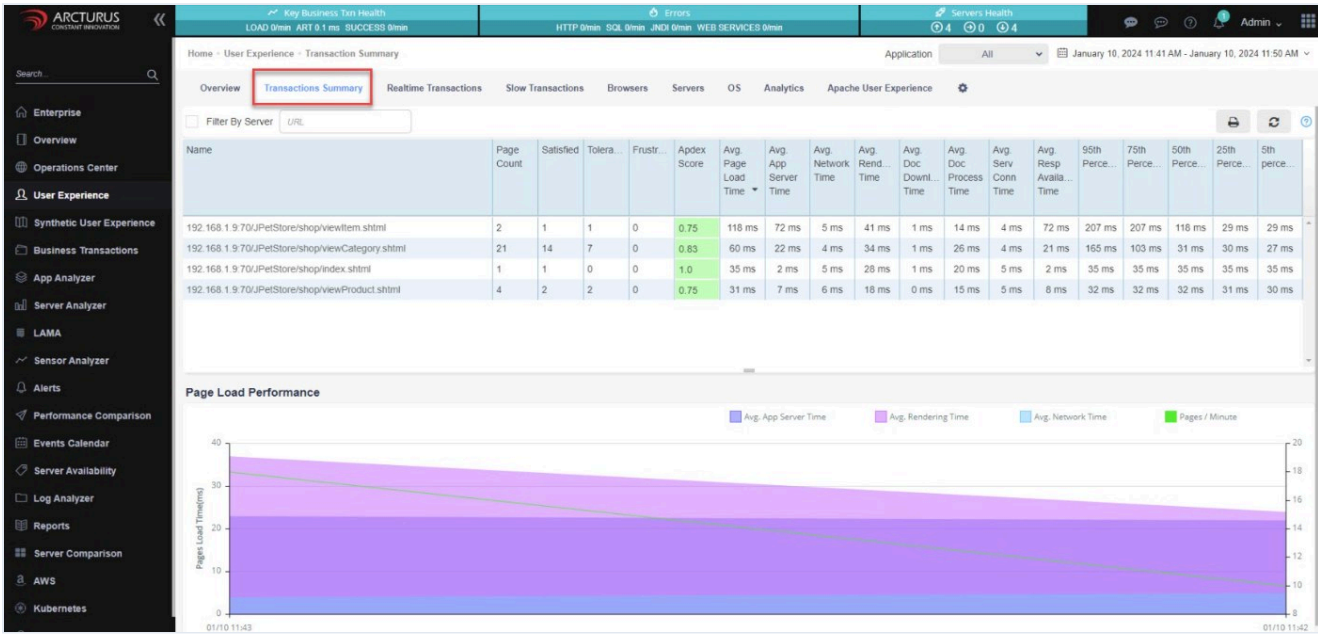
UEM Injection through the Applicare Reverse Proxy Server

Applicare 9.2 · Updated April 1, 2025 · **Pending technical review**

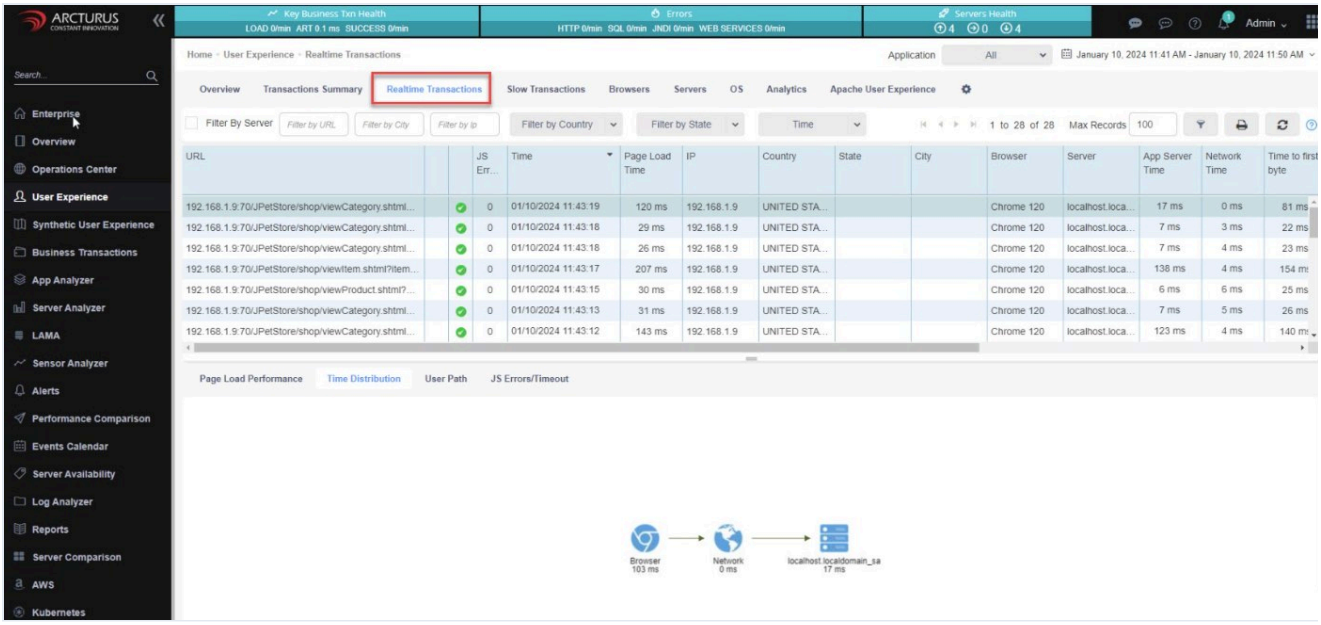
User Experience: An Overview



User Experience: Transaction Overview



User Experience: Real-Time Transactions



Steps to configure the Applicare reverse proxy server

Step 1: Download the Applicare Reverse Proxy Server from this link **download here.**

Step 2: Navigate to the path **<applicare_reverseProxy>/conf** and modify the **reverseproxy.conf** file.

Step 3: Change the **listen port** and **server_name** value inside the **server {** block.

listen -> Applicare reverse proxy server PORT

server_name -> Applicare reverse proxy server IP

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
```

e.g.



Step 4: Add the **upstream** configuration before the **server {** block

Syntax:

```
TEXT
upstream UPSTREAM_REDIRECTOR_NAME {
    ip_hash;
    server REDIRECTOR_APPLICATION_IP:REDIRECTOR_APPLICATION_PORT;
}
```

Example:

```
TEXT
upstream petstoreredirect {
    ip_hash;
    server 192.168.0.165:8881;
}
```

Step 5: Add the new location block after the **location / {** block

Syntax:

```
TEXT
location /REDIRECTOR_APPLICATION_BASE_URL {
    proxy_pass REDIRECTOR_APPLICATION_PROTCOL://UPSTREAM_REDIRECTOR_NAME;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    sub_filter '</body>' '</body>new_string';
    sub_filter_once off;
}
```

Step 6: Replace the **REDIRECTOR_APPLICATION_BASE_URL** with the application base url.

REDIRECTOR_APPLICATION_BASE_URL -> JPetStore

```
TEXT
location /JPetStore {
```

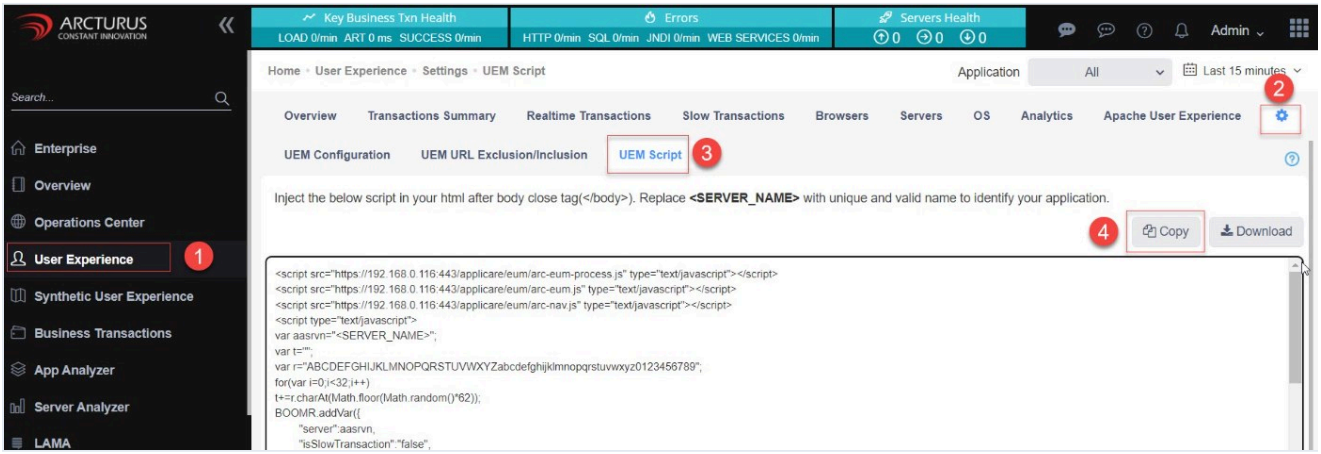
Step 7: Replace the **REDIRECTOR_APPLICATION_PROTCOL** and **UPSTREAM_REDIRECTOR_NAME**.

REDIRECTOR_APPLICATION_PROTCOL -> http

UPSTREAM_REDIRECTOR_NAME -> petstoreredirect

```
TEXT
proxy_pass http://petstoreredirect;
```

Step 8: Login to Applicare console and go to **User Experience -> Settings -> UEM Script** screen and **copy** the script and replace it in the place of **new_string** keyword in the sub_filter in the Nginx sever conf file.



The **conf file** will appear as shown below.

```
        location /JPetStore {
            proxy_pass http://petstoreredirect;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            sub_filter '</body>' '</body><script src="https://192.168.0.116:443/applicare/eum/arc-eum-process.js" type="text/javascript"></script>
<script src="https://192.168.0.116:443/applicare/eum/arc-eum.js" type="text/javascript"></script>
<script src="https://192.168.0.116:443/applicare/eum/arc-nav.js" type="text/javascript"></script>
<script type="text/javascript">
var aasrvn="<SERVER_NAME>";
var t="";
var r="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
for(var i=0;i<32;i++)
t+=r.charAt(Math.floor(Math.random()*62));
BOOMR.addVar({
    "server":aasrvn,
    "isSlowTransaction":"false",
    "avgTime":"0.0",
    "collectProcessListFor":"all",
    "percentile":"0.0",
    "traceId":t,
    "t_backend":"$request_time" * 1000,
    "b_name":BrowserDetect.browser,
    "b_ver":BrowserDetect.version,
    "clientId":0,
    "b_os":BrowserDetect.OS,
    "code":"200",
    "manual":"true",
    "eumId":t});
BOOMR.init({user_ip:"127.0.0.1",beacon_url:"https://192.168.0.116:443/applicare/eum/beacon"});
BOOMR.plugins.RT.startTimer("t_page");
if(!(window.arcPops && window.arcPops.errorInitiated)){
var oldOnError=window.onerror;
window.arcPops={errorInitiated:true};
var xhttp = new XMLHttpRequest();
window.onerror = function(msg, url, lineNo, columnNo, error){
if(oldOnError)oldOnError.apply(this, arguments);
var url="https://192.168.0.116:443/applicare/eum/beacon?serverName="+aasrvn+"&msg="+msg+"&lineNo="+lineNo+"&columnNo="+columnNo+"&eumId="+t+"&jser
navigator.sendBeacon(url);return false;}}
var originalSetRequestHeader = XMLHttpRequest.prototype.setRequestHeader;
XMLHttpRequest.prototype.setRequestHeader = function(header, value) {
originalSetRequestHeader.apply(this, arguments);
};
(function(open){XMLHttpRequest.prototype.open = function(method, url, async, user, password){function handleEvent(e) {
var beaconURL = "https://192.168.0.116:443/applicare/eum/beacon?serverName="+aasrvn+"&msg=Request Timeout&lineNo=&columnNo=&eumId="+t+"&jserror=tr
navigator.sendBeacon(beaconURL);
}function handleHeaderEvent(e) {
if (this.readyState === 1) {
this.setRequestHeader("_arc_req_st_time", Date.now());
}
}
this.addEventListener("abort", handleEvent);this.addEventListener("timeout", handleEvent);this.addEventListener("readystatechange", handleHeaderEve
open.apply(this, arguments);};})(XMLHttpRequest.prototype.open)
</script>';
            sub_filter_once off;
```

Step 9: Substitute the following values.

a. Replace **<SERVER_NAME>** with a unique identifier for your application.

b. Set the value from **"true"** to **"false"**.

c. In the **"t_backend"** section, update the value from **"0"** to **"$request_time" * 1000**



Step 10: Navigate to the <**applicare_reverseProxy>** home directory and run the following command to start.

```bat
startreverseproxy.bat
```

Step 11: Access the application through the Reverse Proxy server.

Syntax:

```xml
<REVERSE_PROXY_PROTOCOL>://<REVERSE_PROXY_IP>:
<REVERSE_PROXY_PORT>/REDIRECTOR_APPLICATION_BASE_URL
```

e.g.

```text
http://192.168.0.113:83/JPetStore
```

Step 12: Navigate to the <**applicare_reverseProxy>** home directory and run the following command to stop the Reverse Proxy Server.

```bat
stopreverseproxy.bat
```

USING APPLICARE › USER EXPERIENCE MONITORING

Apache User Experience

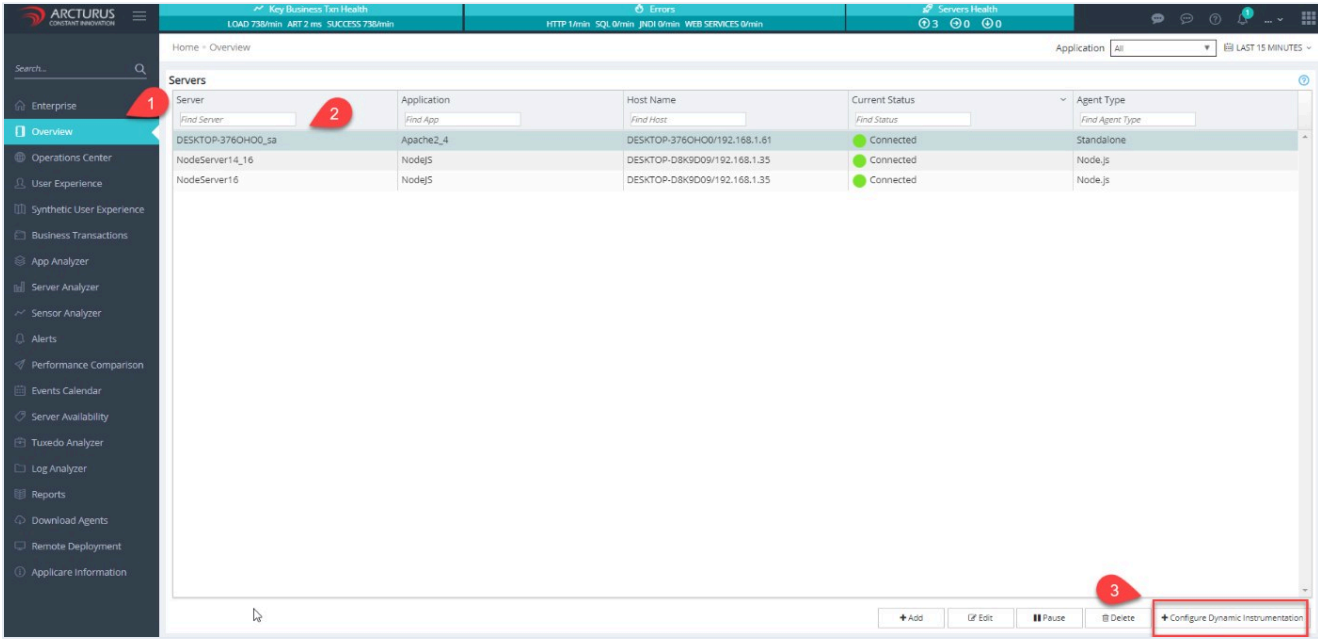
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Enabling Apache User Experience Analyzer

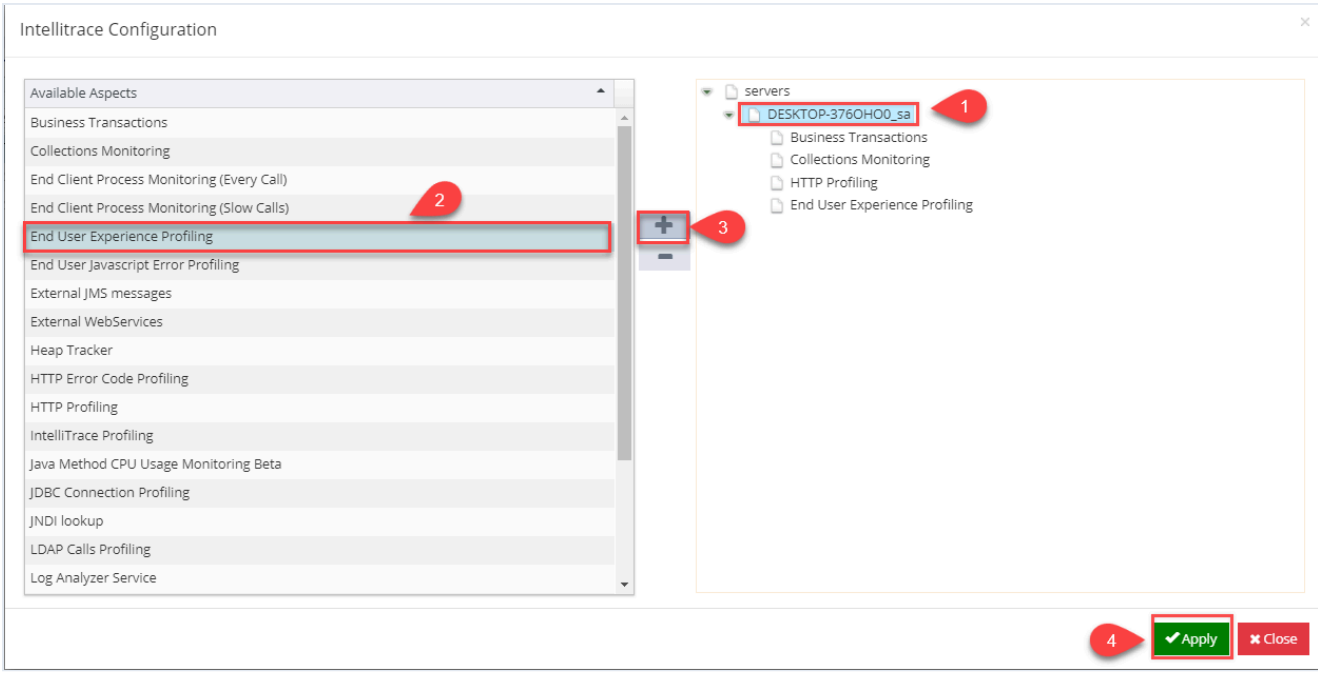
Enabling this feature will log each individual transaction separately into the database, resulting in a large volume of data being recorded.

To set up the Apache User Experience Analyzer, follow these steps:

1. Navigate to the **Overview** screen, select the **server,** and click on the **Configure Dynamic Instrumentation**.

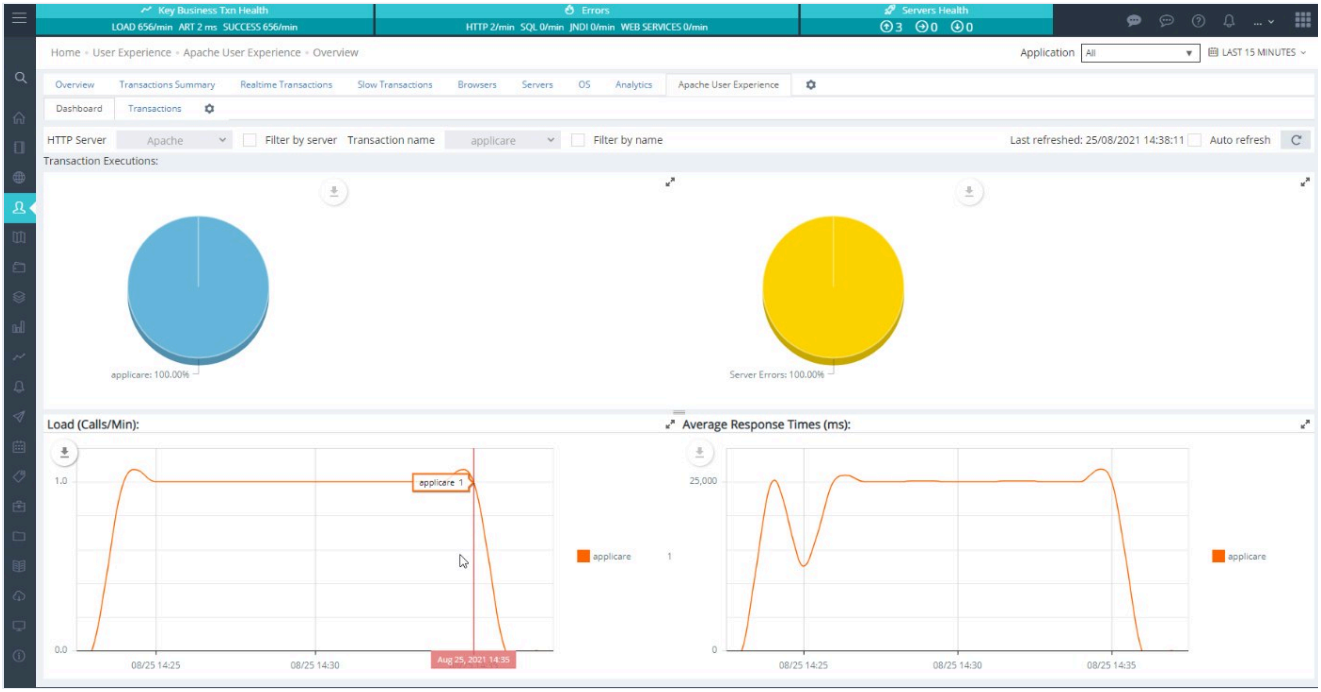


2. Add the "**End User Experience Profiling**" aspect to the server.

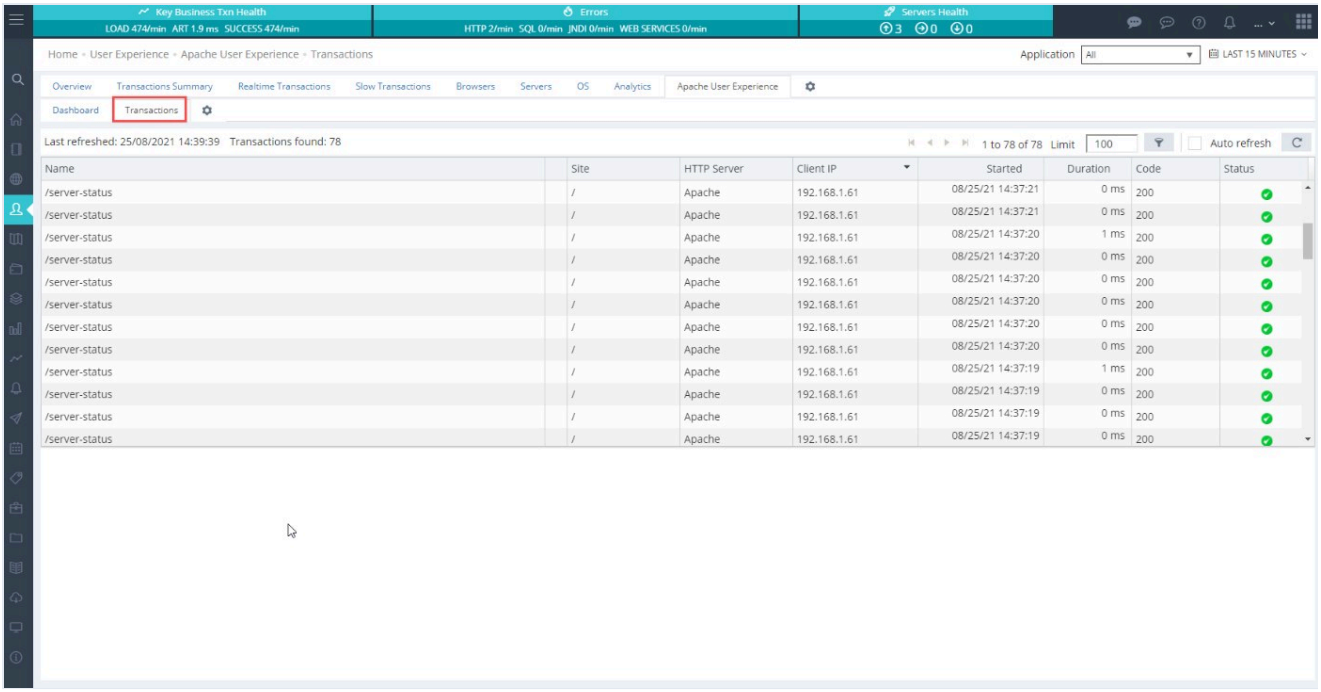


3. Within a few minutes, the data will appear in the "**Apache User Experience Analyzer**".

Dashboard

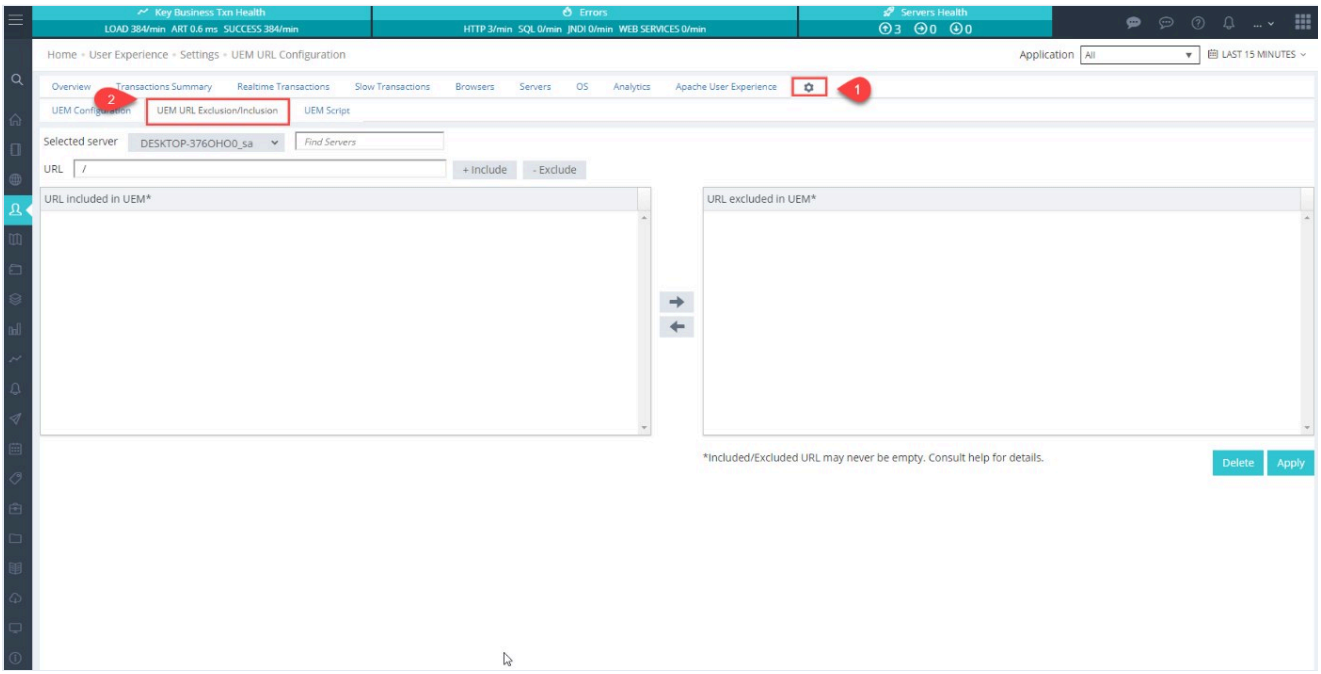


Transactions



To Restrict logging to specific URLs in the Apache User Experience Analyzer

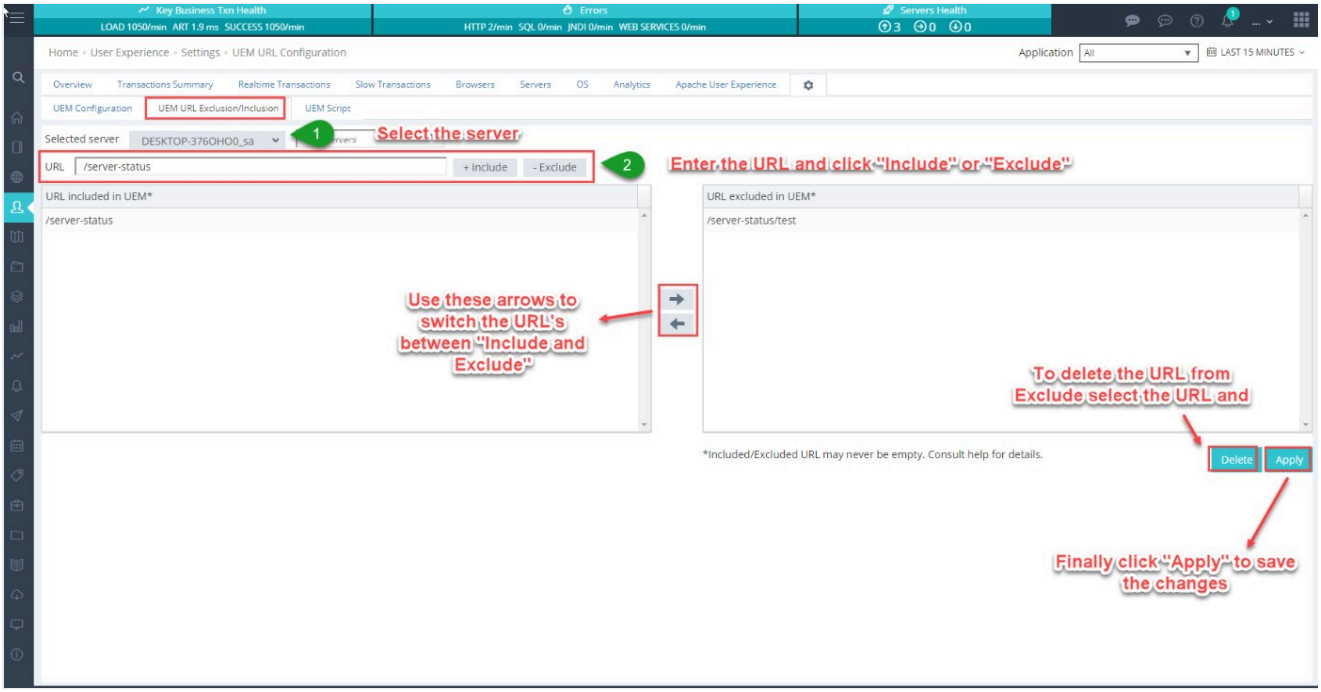
1. Navigate to **User Experience --> Settings (icon) --> UEM URL Exclusion/Inclusion**



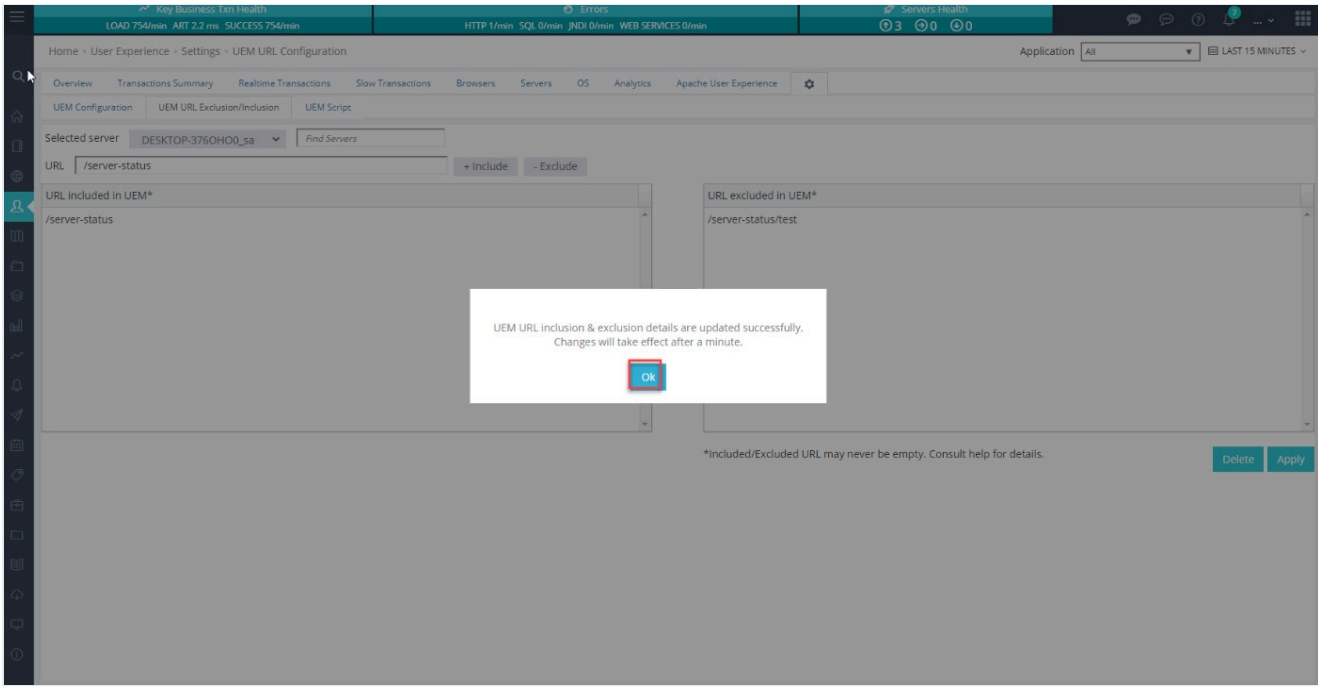
2. Add the URL's to the UEM Include and Exclude lists.

UEM Include & Exclude

If the UEM Include list is populated, Only the URLs specified in that list will be logged.



Click "OK"



After applying the changes, a restart of SingleAgent is required. Ensure you add all necessary URL's to the UEM Include or Exclude list, apply the changes, and then restart the SingleAgent.

Monitoring with Applicare

USING APPLICARE › USER EXPERIENCE MONITORING

UEM IP Address Translation

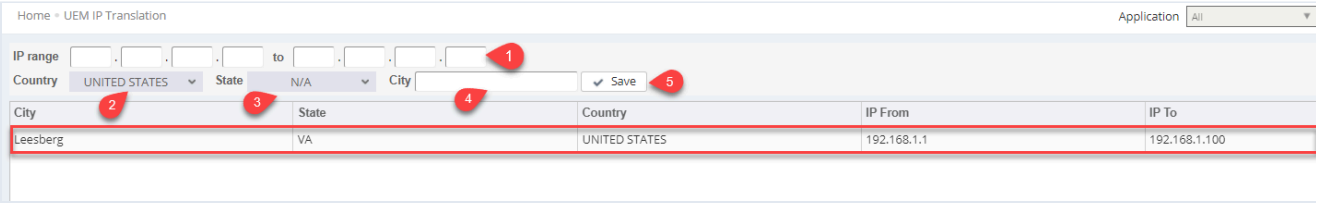
Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Overview

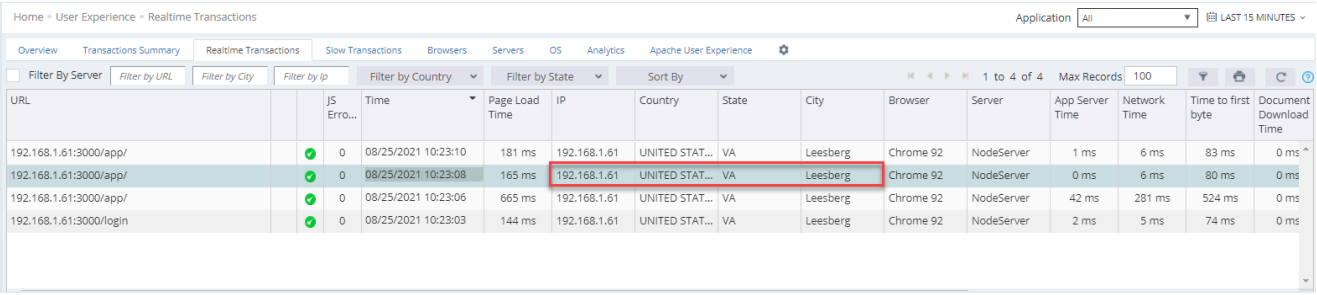
UEM IP translation configures the source of request that reach the application on the app server.

Configuration

Specify on the IP range, select country and state, enter the city name, and then click the save button to apply the configuration.



After configuring and saving, navigate to **User Experience-> Realtime Transactions** to view the transaction data associated country, state, and city.



Monitoring with Applicare

USING APPLICARE › USER EXPERIENCE MONITORING

iOS Mobile App Performance Monitoring

Applicare 9.2 · Updated August 23, 2024 · **Pending technical review**

The Applicare Mobile Application Tracker provides real-time monitoring of your mobile applications, including the ability to track business transactions. This powerful service offers features such as real-time request tracking, activity path tracking, geo-tracking, error reporting, and much more.

One of the advantages of mobile app development is the ability to quickly distribute new versions and ensure that all users are using the latest app. If you haven't been monitoring your code previously, you can easily implement Applicare Tracker, push the update, and start collecting metrics within minutes.

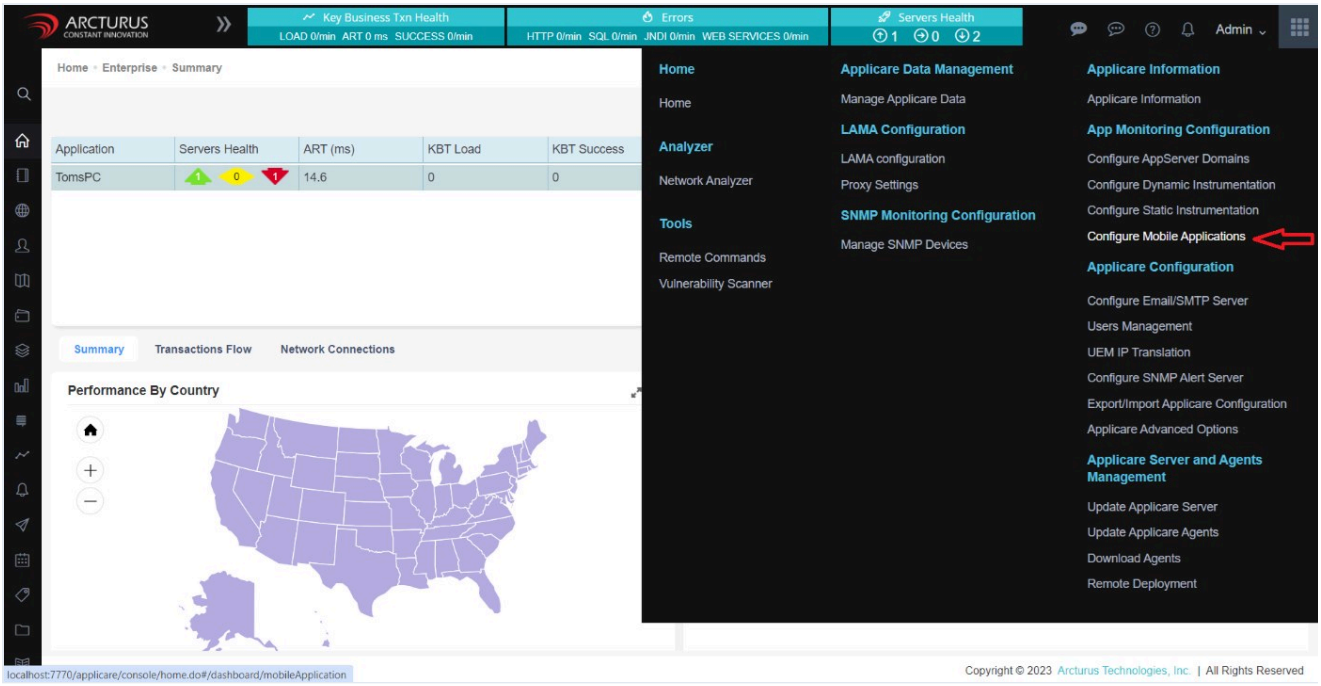
Why Use Applicare Mobile Application Tracker?

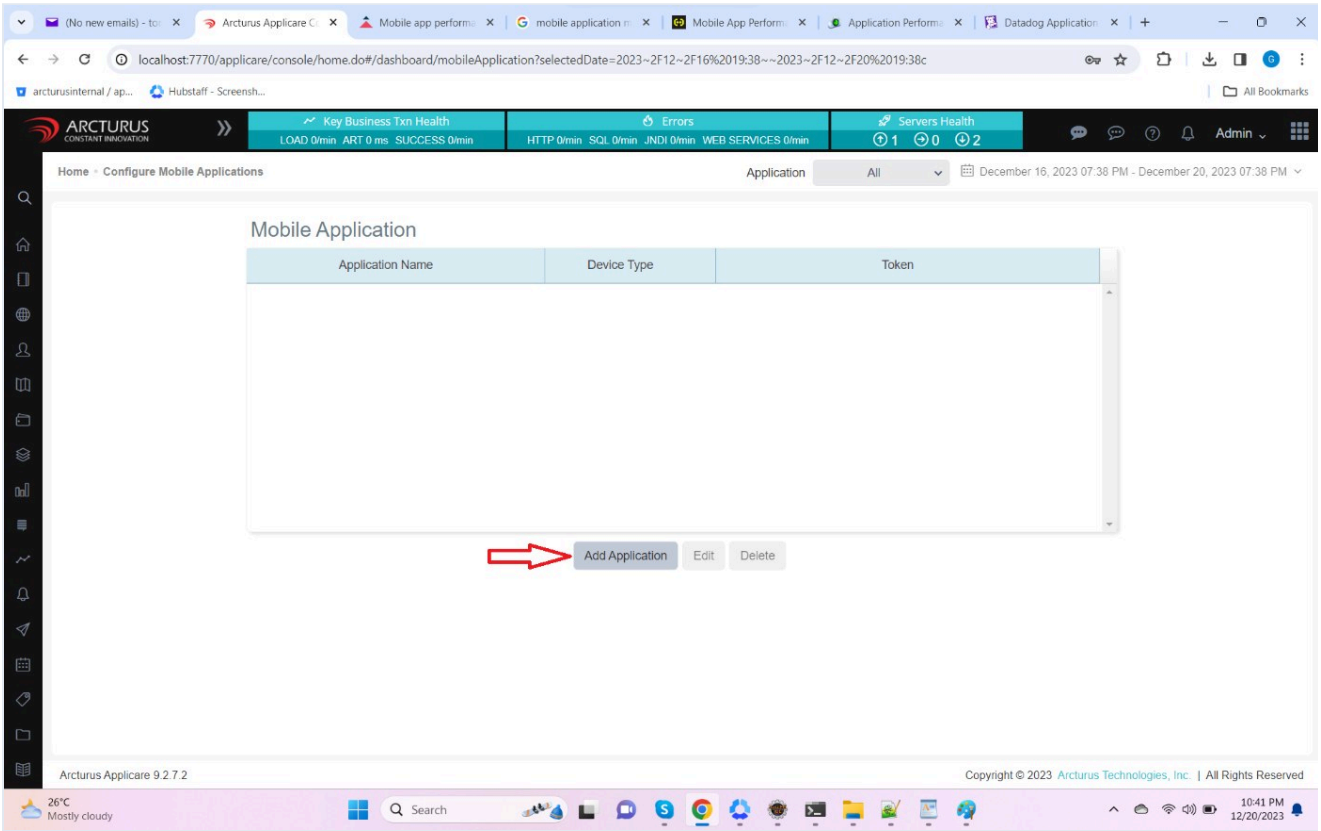
- Quick and straightforward setup process.
- Seamlessly integrates with any Android application.
- Provides insights into real user interactions.
- Capture performance metrics, set SLAs for Android method calls, and track individual transactions with performance statistics and call graphs.
- Gain insights into user characteristics.
- Visualize user locations on a map.
- Detect performance issues and server failures automatically.
- Receive regular reports to stay informed about performance issues.
- Examine detailed historical server data, including conditions before crashes.

Steps to Integrate the Applicare Mobile Tracking Agent

Register your Mobile App and Generate a Token

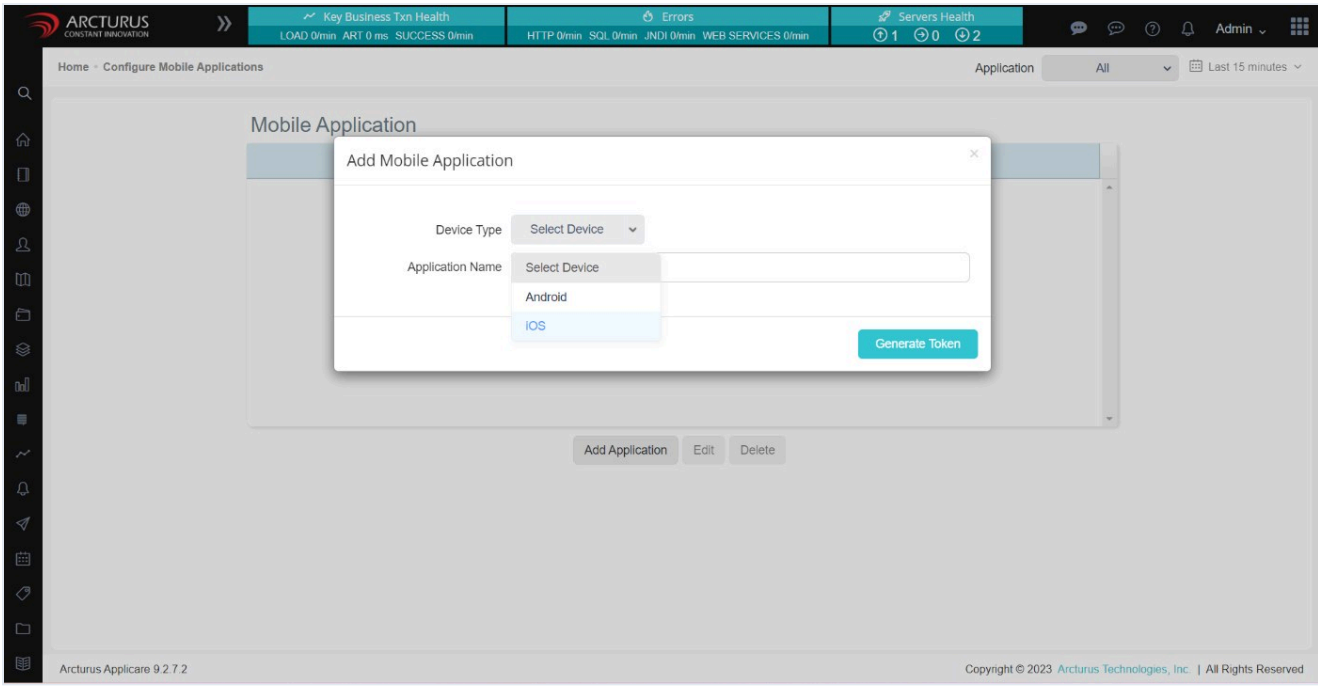
- Sign in to your Applicare Admin Console.
- Access the Menu in top right corner and navigate to the "Configure Mobile Application" screen.

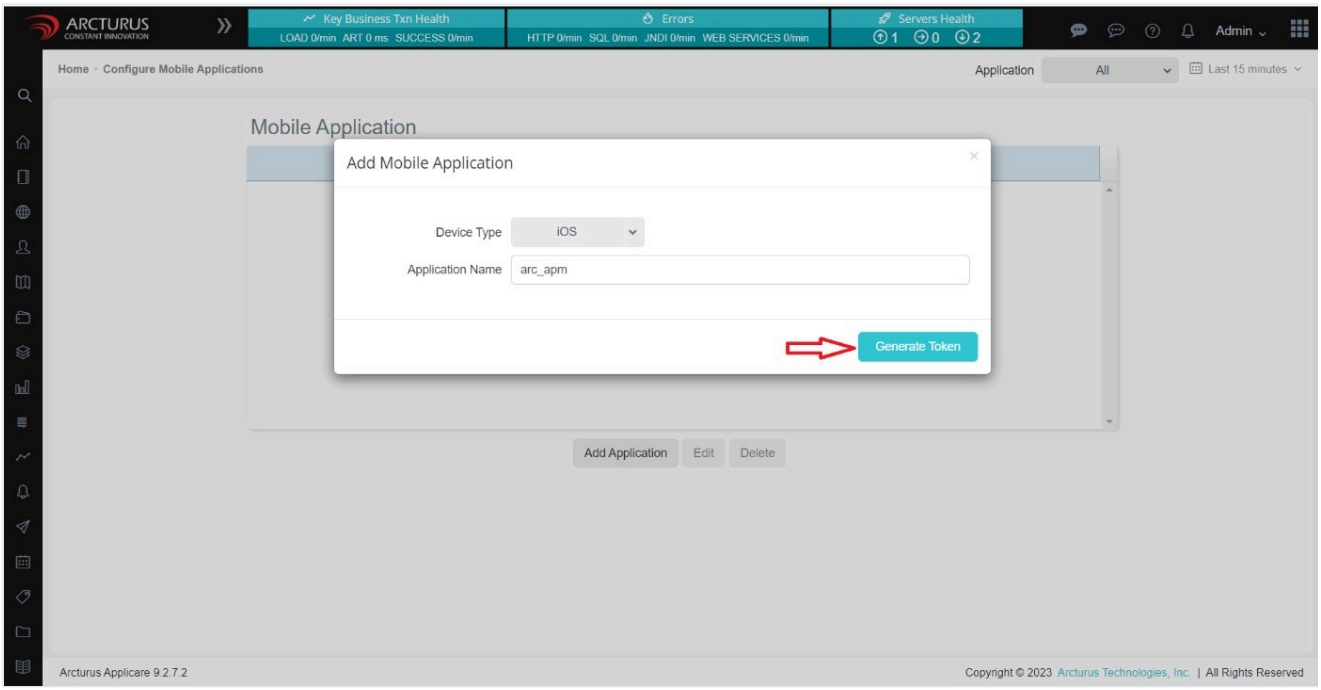


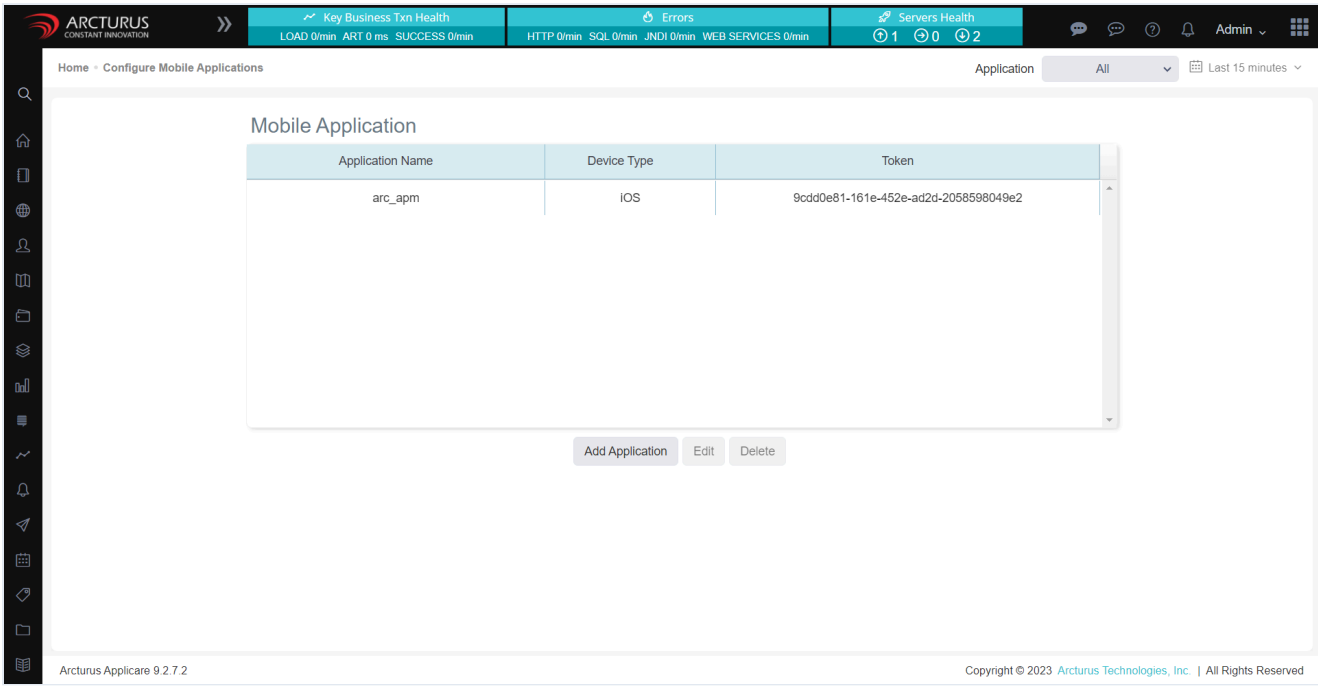


- On the "Configure Mobile Application" screen, select "iOS" from the dropdown menu. Enter your application's name and click the “Generate Token” button. A token will be generated; copy and save this token for future use.

.


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
Last 15 minutes
Mobile Application
Add Mobile Application
Device Type
Select Device
Application Name
Select Device
Android
iOS
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
Last 15 minutes
Mobile Application
Add Mobile Application
Device Type
iOS
Application Name
arc_apm
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved



Adding Applicare Agent to your app

Add the framework applicare_agent.framework file into your Objective-C or Swift project.

For Objective-C

- Inside the AppDelegate.m file import the header by adding the following line.

```
TEXT
#import <applicare_agent/ApplicationTracker.h>
```

- Now inside the didFinishLaunchingWithOptions method, add the following lines

If your app is using server add the below code.

```
TEXT

NSString *serverPort = YourServerPort;
NSString *serverIp = YourServerIp;
NSString *token = YourToken;
NSString *agentType = @"agent";
```

If your app is not using server, add the below code.

```
TEXT
NSString *serverPort = ApplicareServerPort;
NSString *serverIp = ApplicareServerIp;
NSString *token = YourToken;
NSString *agentType = @"none";
```

- Now initialize the framework by adding the following line.

```
TEXT
[ApplicationTracker initialize:serverPort arg2: serverIp arg3: token arg4: agentType];
```

- It should finally looks like below for apps with server.

```
18
19 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
20     // Override point for customization after application launch.
21     NSString *serverPort = @"8080";
22     NSString *serverIp = @"192.168.2.100";
23     NSString *token = @"4db94134-bc29-4eba-b972-038d0145ce87";
24     NSString *agentType = @"agent";
25     [ApplicationTracker initialize:serverPort arg2: serverIp arg3: token arg4: agent
26     return YES;
27 }
28
```

For Swift : Here are steps to add an Objective-C frame to a Swift project.

- **Import the Framework**: Drag and drop the Objective-C framework into your Xcode Swift project.
- **Create a New Objective-C File**: Go to File -> New -> File, and select Objective-C for iOS.
- **Create Bridging Header**: Accept the prompt to create a bridging header file between Objective-C and Swift.
- **Delete the Objective-C File**: You can now delete the newly created Objective-C file, but keep the bridging header file ${YOURPROJ}-Bridging-Header.h.
- **Import the Framework**: In the bridging header file, use the standard Objective-C import syntax to include your framework.

```
TEXT
#import <applicare_agent/ApplicationTracker.h>
```

- This approach eliminates the need to perform an ***import ApplicationTracker.h*** directly in your ***AppDelegate.swift*** file. By using the bridging header, you can access the framework across all Swift files in your project.
- Inside the didFinishLaunchingWithOptions method in the AppDelegate.swift file add the below lines.

If your app is not using server add the below lines.

```
TEXT
let serverPort = ApplicareServerPort
let serverIp = ApplicareServerIp
let token = YourToken
let agentType = "none"
```

If your app is using server then add the below lines.

```
TEXT
let serverPort = YourServerPort
let serverIp = YourServerIp
let token = YourToken
let agentType = "agent"
```

- Now initialize the framework by adding the following lines.

```
TEXT
ApplicationTracker.initialize(serverPort, arg2: serverIp, arg3: token, arg4: agentType)
```

- It should finally looks like below for apps without server.

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject
    // Override point for customization after application launch.
    let serverPort = "8880"
    let serverIp = "192.168.2.100"
    let token = "4db94134-bc29-4eba-b972-038d0145ce87"
    let agentType = "none"
    ApplicationTracker.initialize(serverPort, arg2: serverIp, arg3: token, arg4: agentType)
    return true;
}
```

USING APPLICARE › USER EXPERIENCE MONITORING

Monitoring performance for Android mobile apps

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

Applicare Mobile Application Tracker offers a solution for real-time monitoring of your mobile applications, with a distinctive feature for tracking live business transactions. The Applicare tracker is a robust service equipped with features like real-time request tracking, activity path tracking, geo-tracking and much more.

One of the advantage developing a mobile app is the ability to quickly distribute updates, ensuring that your entire user base is always using the latest version of your app. If you weren't previously monitoring your code, you can effortlessly implement Applicare tracker, deploy it, and begin collecting metrics within minutes.

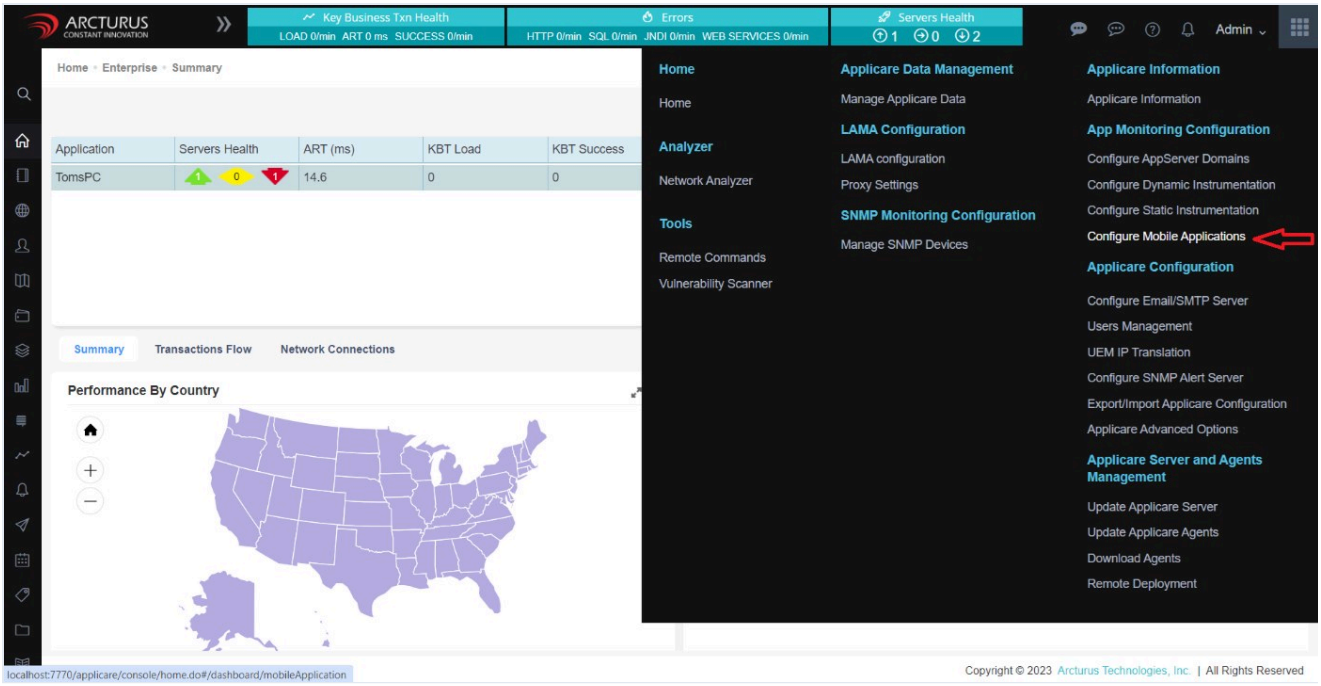
Why choose Applicare Mobile Application Tracker?

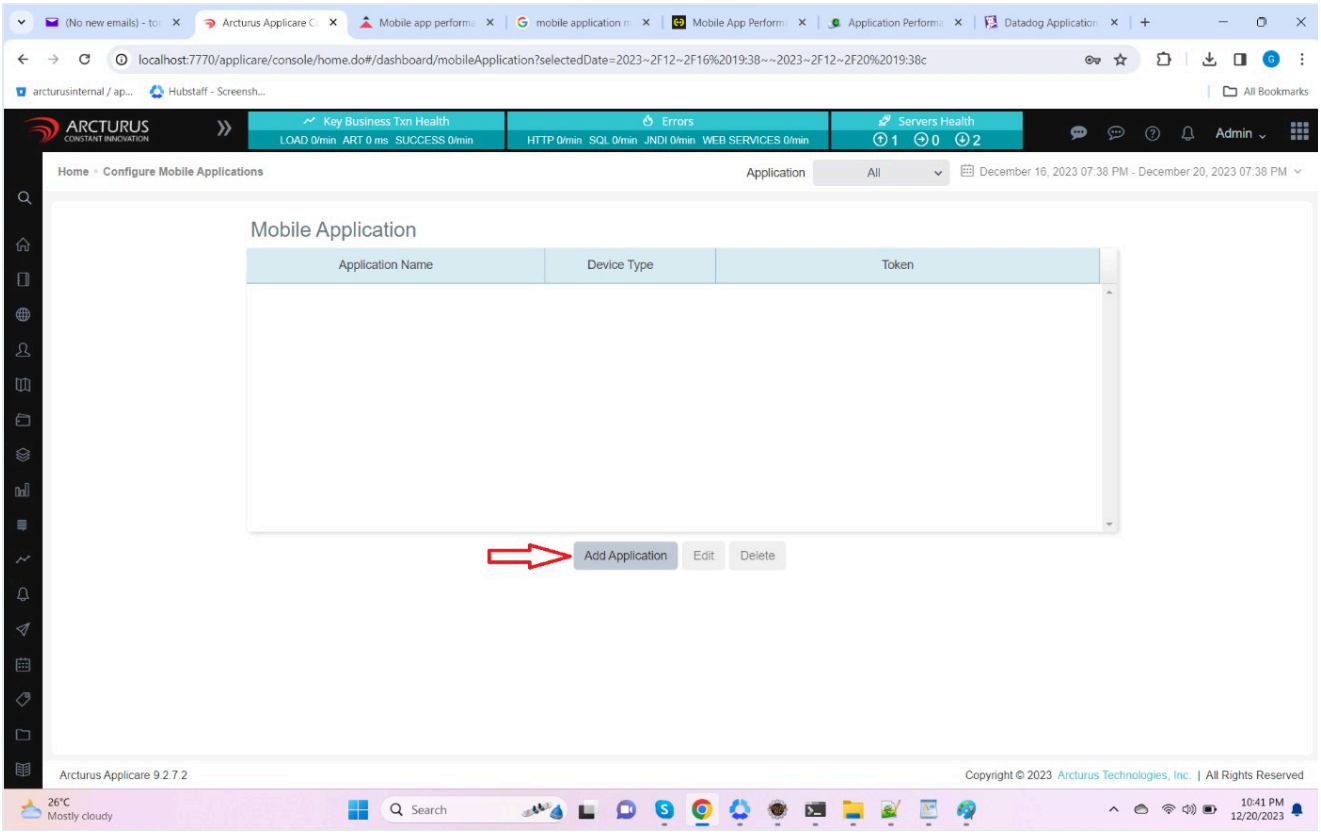
- Ease of installation.
- Easily integrates with any Android application.
- Track real user interactions for comprehensive insights.
- Business Transactions features allows you to capture performance statistics and optionally set SLAs for Android methods calls and track individual execution of transactions with performance stats and call graphs.
- Demographic characteristics of users.
- Visualize user locations.
- Auto detect the root cause of performance degradation and server failure.
- Generate scheduled reports to keep you aware of performance issues.
- Analyze detailed historical server information, even pre-crash conditions.

Steps to Integrate Applicare mobile tracking agent

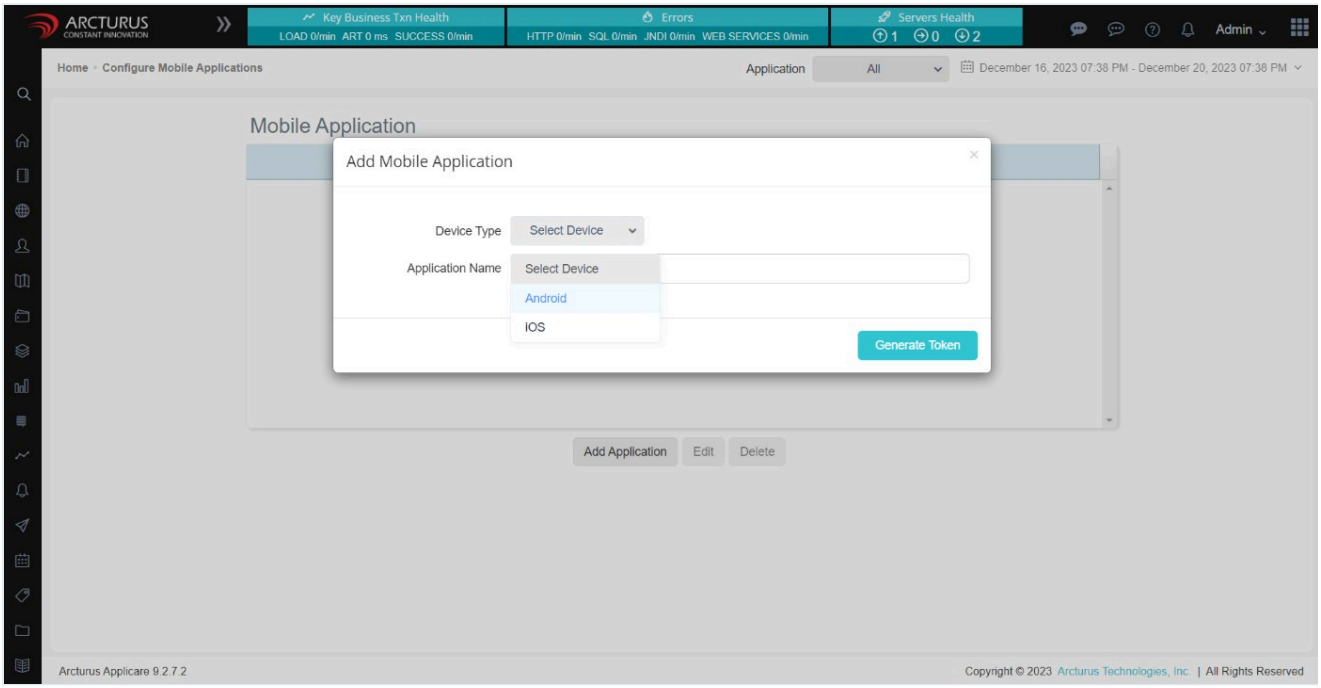
Generate a Token and Download the Applicare Android SDK Agent

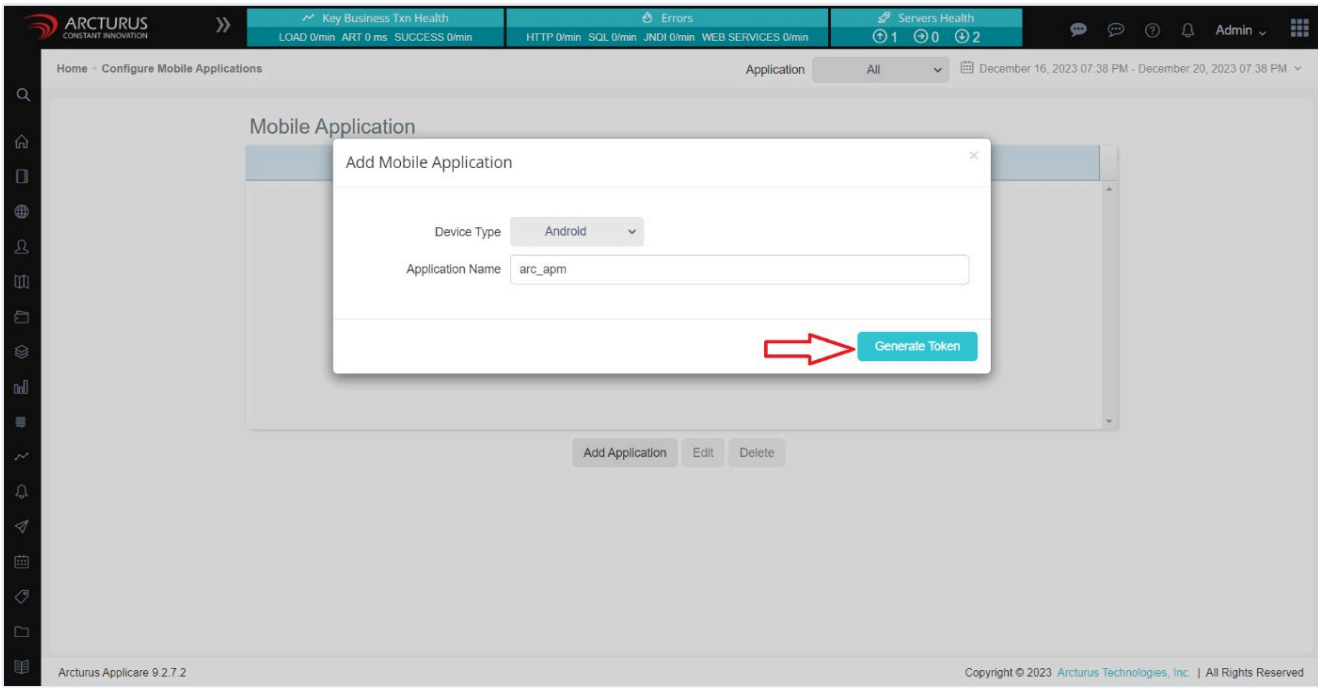
- Log in to your Applicare Admin Console.
- Access the menu in the top right corner and navigate to the "Configure Mobile Application" screen.

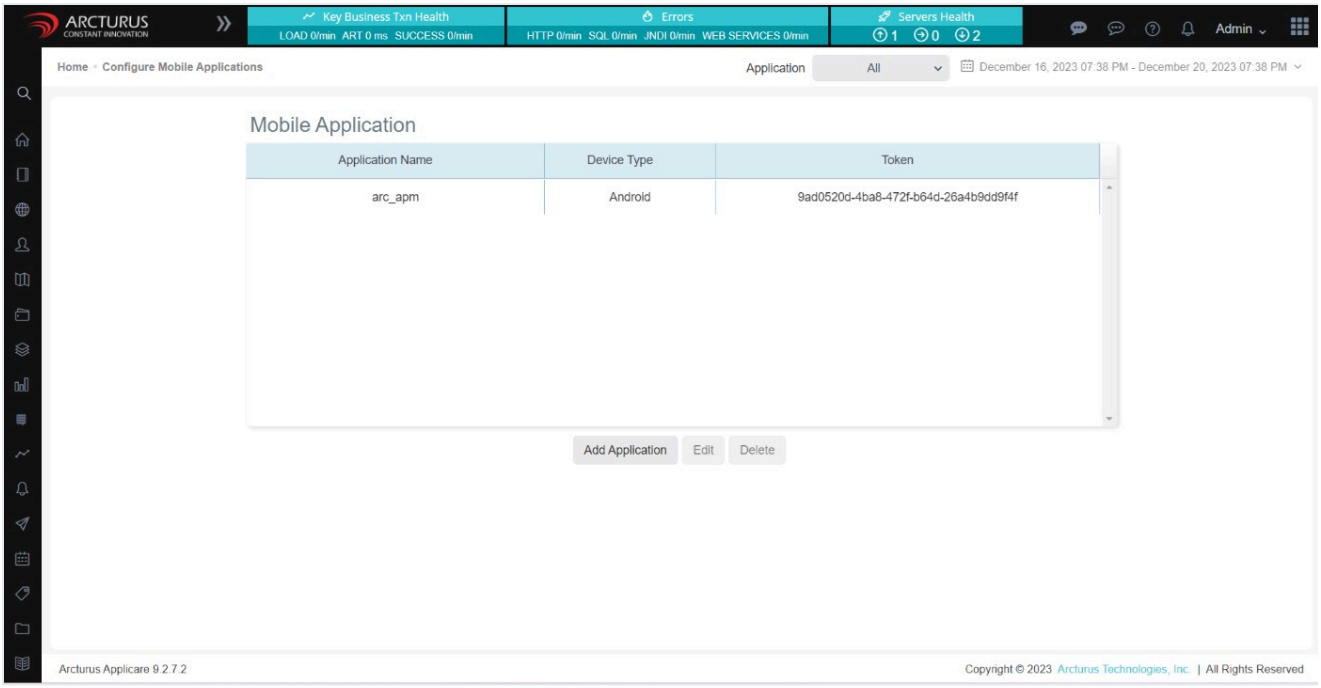




- On the "Configure Mobile Application" screen, select "Android" from the dropdown menu. Enter the application name and click the “Generate Token” button. A token will be generated; be sure to copy and save it for future use.


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
December 16, 2023 07:38 PM - December 20, 2023 07:38 PM
Mobile Application
Add Mobile Application
Device Type
Select Device
Application Name
Select Device
Android
IOS
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
December 16, 2023 07:38 PM - December 20, 2023 07:38 PM
Mobile Application
Add Mobile Application
Device Type
Android
Application Name
arc_apm
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved



- Download the Applicare Android SDK agent.

Integrating the Applicare Agent into your Android Project

- Place the ApplicareMobileTracker.aar file into the **libs** folder under the **library** directory of your Andriod app project.
- Open the build.gradle file located in the **app** folder and add the necessary dependency.

```TEXT
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'ApplicareMobileTracker.aar'
}
```

- Open the activity class where you want to track load time. In the onCreate() method, create an instance of the TesFactoryMethod Class.

```
TEXT
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        TestFactoryMethod.passtoken("XXXXXXXXXXXXXXXXXXX");
        //call applicare method
        TestFactoryMethod.getStartTime();
    }

    @Override
    public void onResume() {
        super.onResume();
        TestFactoryMethod.getEndTime();
        TestFactoryMethod.sendDetails(MyApplication.this);
    }
}
```

- Build and install the updated Application.
- Log into the admin console to monitor your Android app's performance.

Monitoring with Applicare

USING APPLICARE › USER EXPERIENCE MONITORING

Measuring User Satisfaction - Apdex

Applicare 9.2 · Updated September 13, 2024 · **Pending technical review**

The **Application Performance Index (Apdex) score** quantifies user satisfaction with the performance of web applications and services. It converts various response times into a single score ranging from 0 to 1, with 1 indicating perfect satisfaction and 0 representing complete dissatisfaction. By default, Applicare uses the average response time of transactions as its **Desired response time(T).**

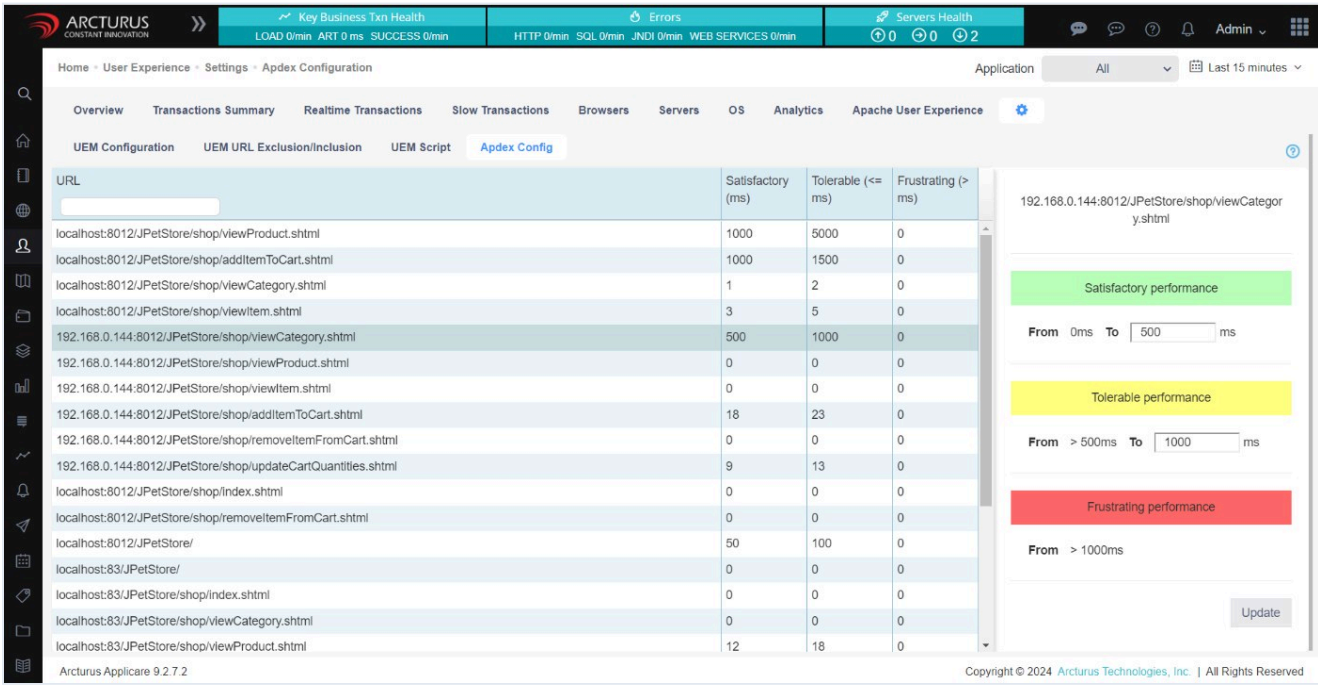
Each transactions is classified as follows:

- **Satisfied:** Response times less than or equal to T.
- **Tolerating:** Response times between T and 4T.
- **Frustrated:** Response times greater than 4T.

The Apdex Score is calculated using the formula:

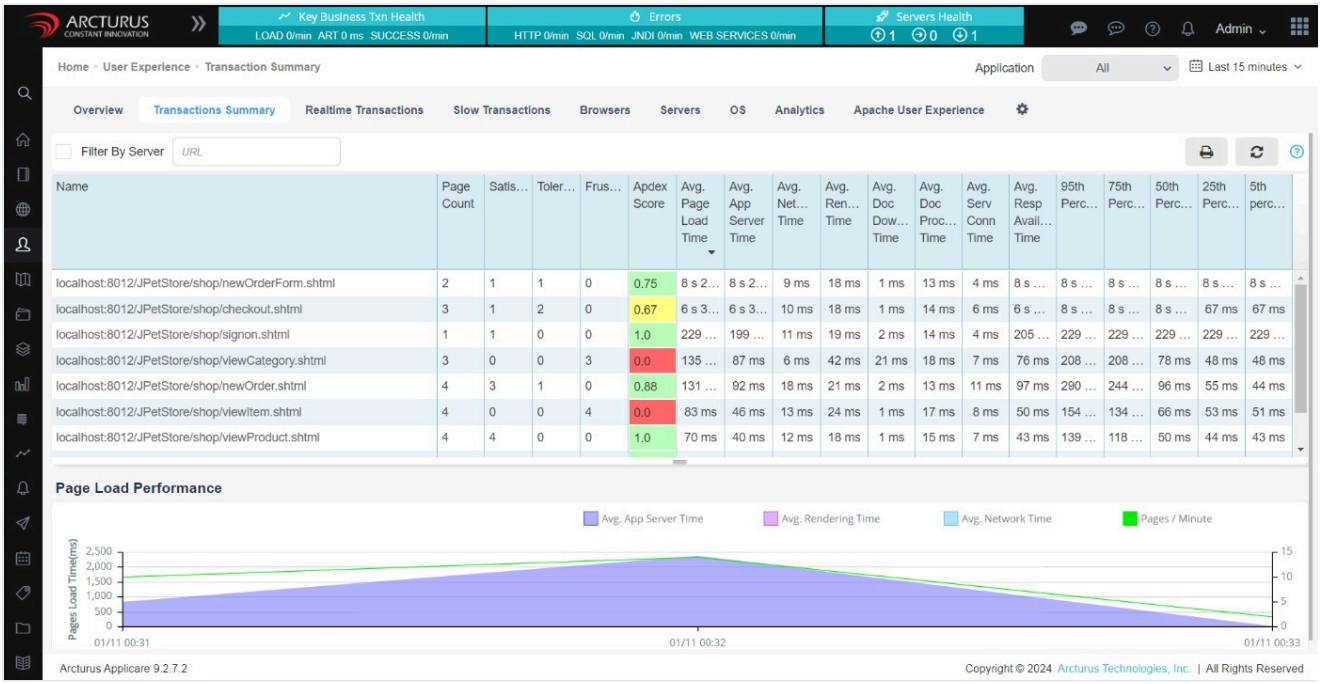
Apdex Score = (Satisfied Count + (Tolerating Count / 2)) / Total Count

You can configure the Apdex threshold for individual transactions in Applicare console by **navigating to Home-> User Experience -> Settings -> Apdex Configuration** screen.



The Apdex score for all the transactions can be viewed under **User Experience -> Transaction summary.**

If no Apdex threshold is set for a transaction, then average response time of that transaction will be used as the default threshold.



Monitoring with Applicare

PART

Troubleshooting

Known errors, causes and fixes.

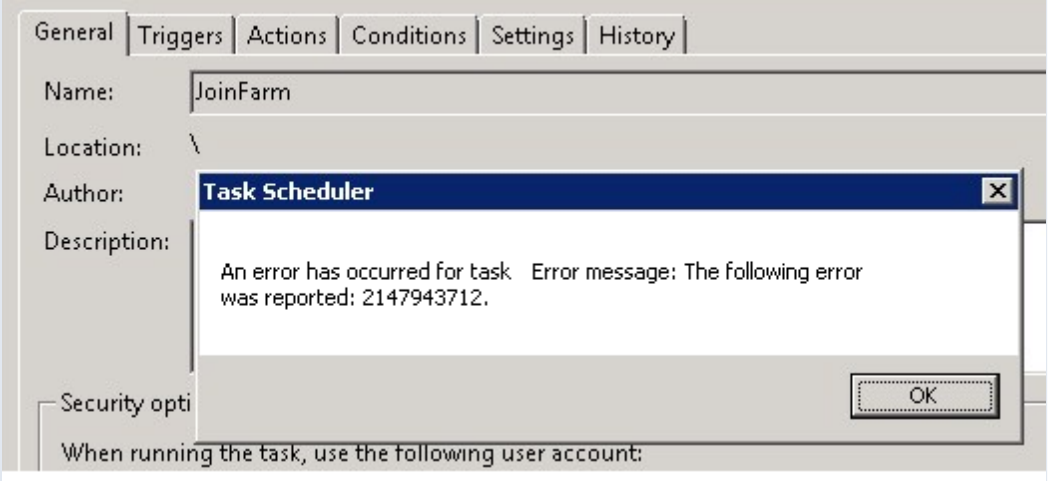
TROUBLESHOOTING

Error: Unable to Create Task in Task Scheduler

Applicare 9.2 · Updated September 13, 2024 · windows, tasks · **Pending technical review**

In Windows, when you confirm the creation of a new task, you might the following error:

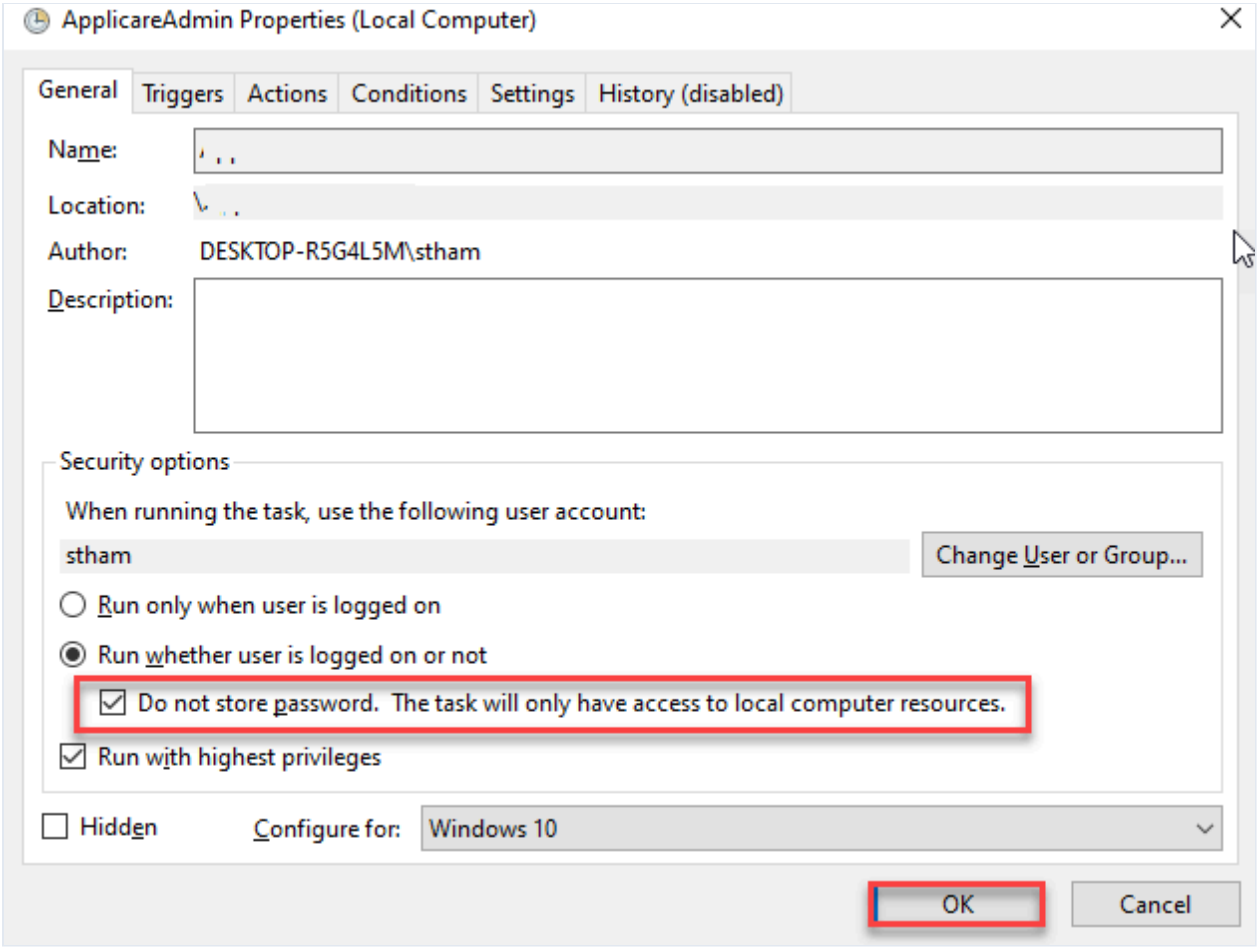
An error with task **<TaskName>**. Error message: **2147943712**.



This error message occurs when Windows was unable to store **credentials** (especially the password) of the user who run the task.

Solution:

1. Enable **Do not store password** option and click **OK**.



If the task is created with the "Do not store password" option, the Java processes may continue to run in the background even after the task ends. You will need to manually terminate the Java processes using Task Manager.

Monitoring with Applicare

TROUBLESHOOTING

Issue with Java during installing wrapper on Linux

Applicare 9.2 · Updated August 23, 2024 · java, linux, singleagent · **Pending technical review**

Step 1: Navigate to "<**Applicare SingleAgent_home_path**>\Wrapper\Linux" directory and edit the **config.sh** file.

Step 2: Copy the line that says **JAVA_BINARY_PATH_DEFAULT=""**, comment it out and paste it on the next line. Then, insert the path to the java directory in double quotes as shown below and save the file.

e.g. /home/apache-tomcat/jre/Linux/bin/java

Before changes config.sh file

```
#!/bin/bash

JAVA_WRAPPER_ARGS="wrapper_args.txt"
JAVA_WRAPPER="wrapper.sh"
JAVA_BACKUP_NAME="java_default"
#JAVA_BINARY_PATH_DEFAULT="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7_8.x86_64/bin/java"
JAVA_BINARY_PATH_DEFAULT=""
```

After changes config.sh file

```
#!/bin/bash

JAVA_WRAPPER_ARGS="wrapper_args.txt"
JAVA_WRAPPER="wrapper.sh"
JAVA_BACKUP_NAME="java_default"
#JAVA_BINARY_PATH_DEFAULT="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7_8.x86_64/bin/java"
# JAVA_BINARY_PATH_DEFAULT=""
JAVA_BINARY_PATH_DEFAULT="/home/apache-tomcat/jre/Linux/bin/java"
```

Step 3: To install Applicare Singleagent execute **install.sh** file.

```
[root@localhost linux]# ./install.sh
Wrapped JAVA : /home/[redacted]/apache-tomcat/jre/Linux/bin/java
[root@localhost linux]#
```

TROUBLESHOOTING

Java issues encountered during wrapper installation

Applicare 9.2 · Updated August 22, 2024 · java, singleagent · **Pending technical review**

If you encounter the following **Java** error during while wrapper installation, follow these steps.

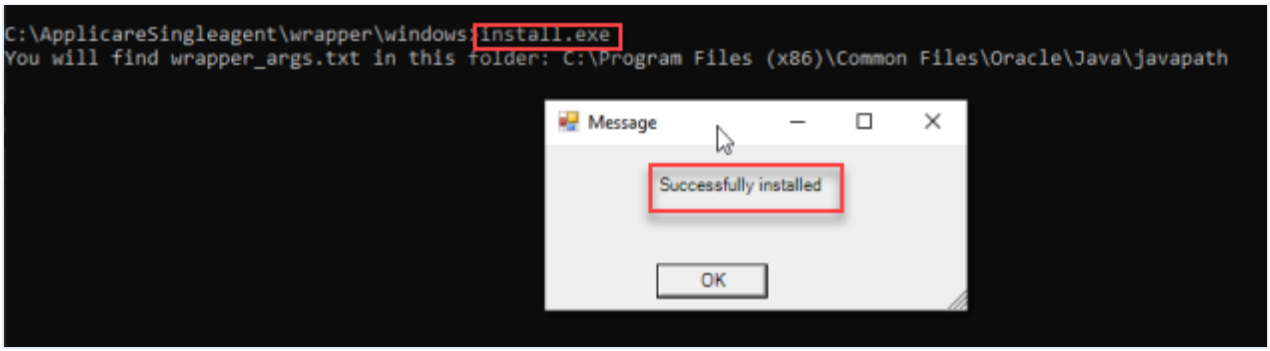
```
E:\ApplicareSingleAgentWindows\wrapper\windows>install.exe
ERROR: The term 'java' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct and try again.
ERROR: Cannot bind argument to parameter 'Path' because it is null.
ERROR: Cannot bind argument to parameter 'Path' because it is null.
ERROR: Cannot bind argument to parameter 'Path' because it is null.
ERROR: Cannot bind argument to parameter 'Path' because it is null.
```

Step 1: Navigate to the "<**Applicare SingleAgent_home_path**>\Wrapper\windows" directory and edit the **java_dir.config.**

Step 2: Save the **path** to the Java installation directory including **bin** subdirectory, where the application is running.

Example: C:\javajdk1_8_0_265\bin

Step 3: Run the **install.exe** file as an administration to install Applicare SingleAgent.

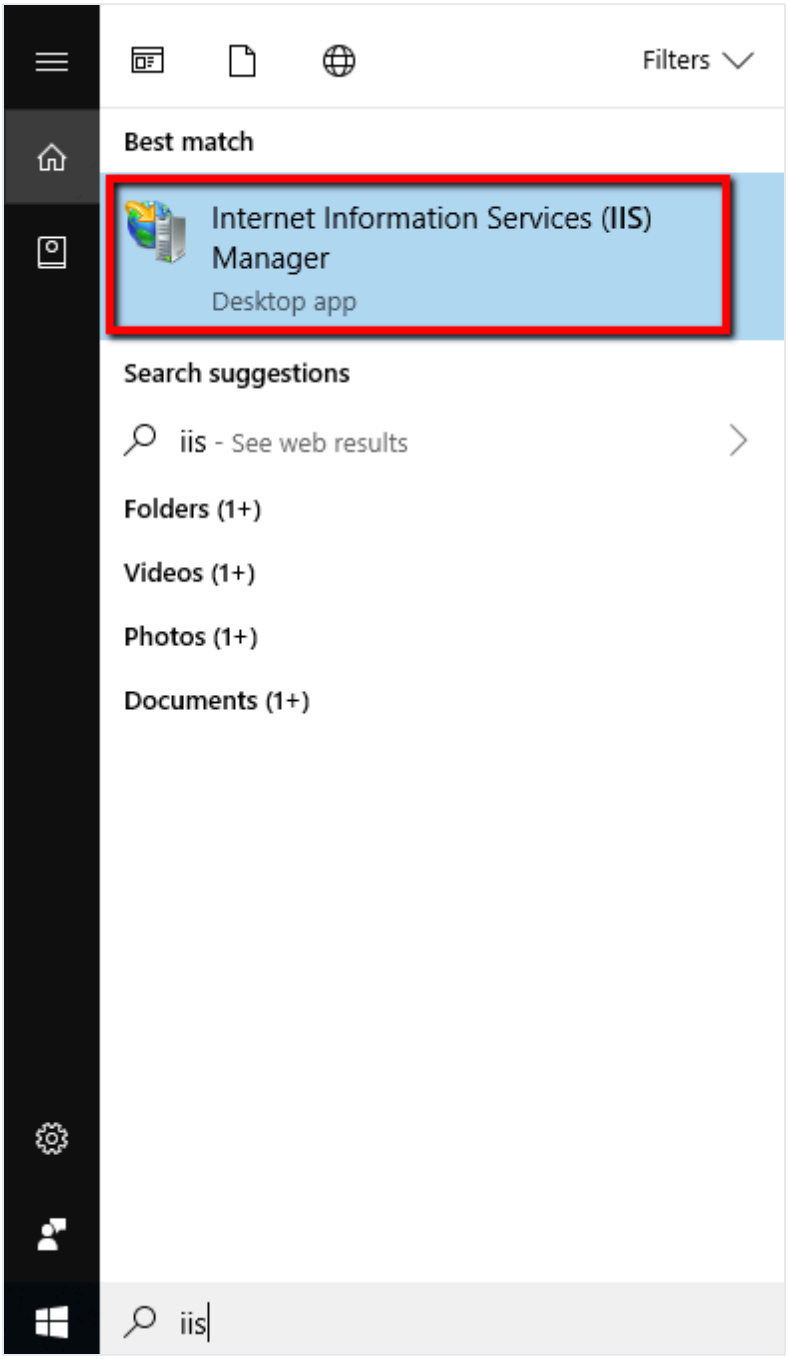


TROUBLESHOOTING

Enable read access to IIS logs

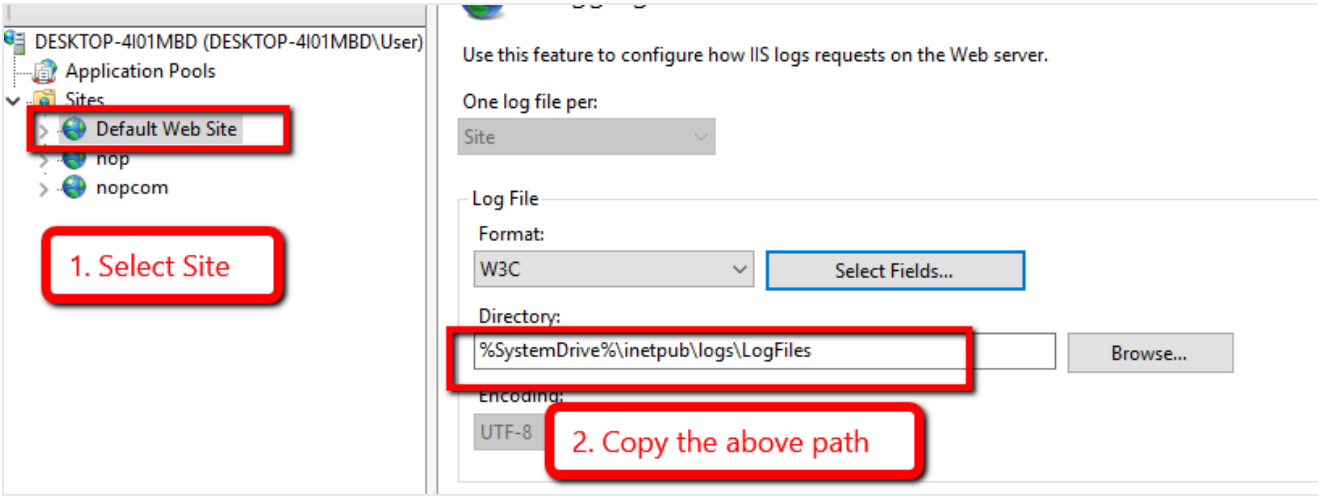
Applicare 9.2 · Updated August 26, 2024 · iis, windows · **Pending technical review**

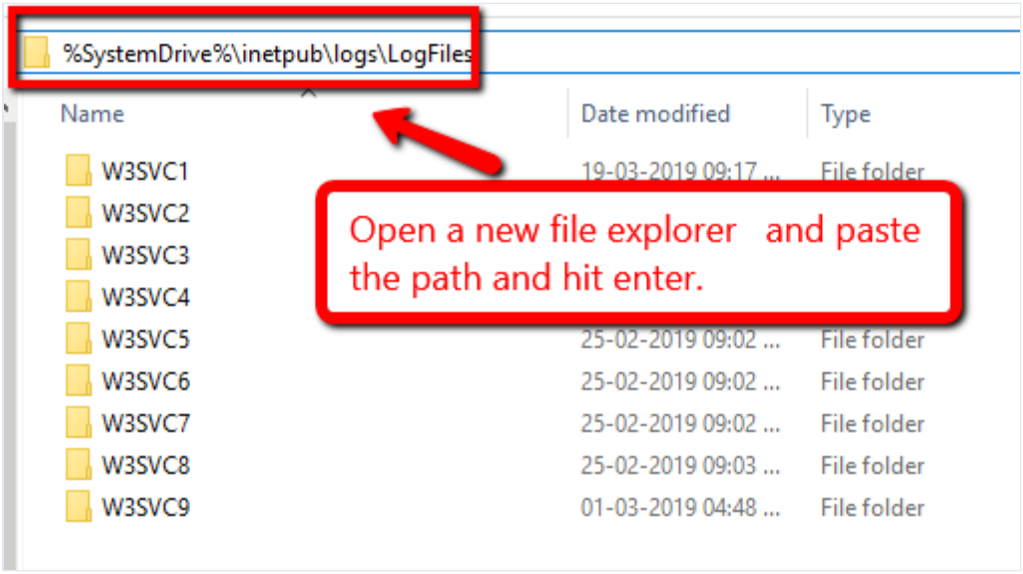
Step 1: On the Agent machine (**i.e** IIS machine), type "**IIS**" in the start menu search and select "**Internet Information Services (IIS) Manager**".



Step 2: Ensure you have access to all the IIS Log folders and log files for all the sites.

To view the IIS log folder and files select the site and double-click the Logging option, copy the path in the **Directory** field.





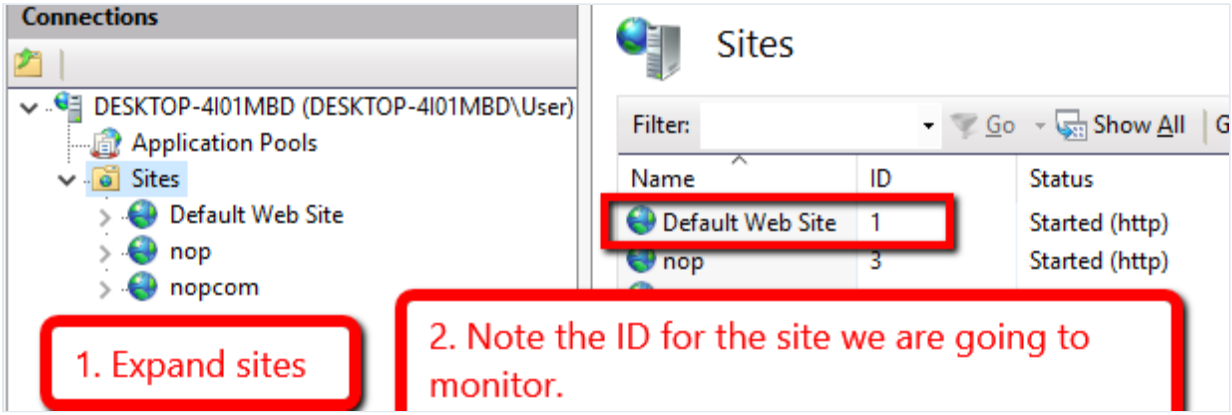
Ensure that the Applicare Agent has access to all the folders and files in the specified path.

e.g.

Let's check the "**Default Web Site**".

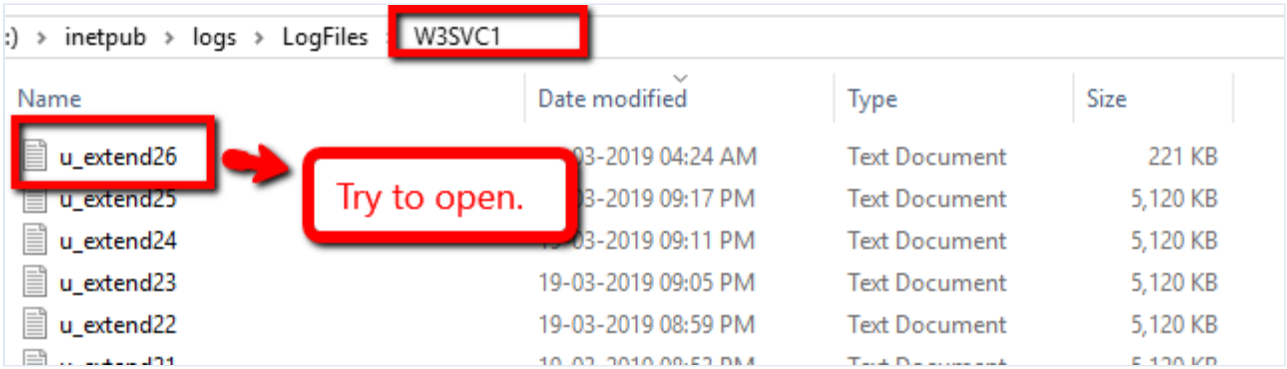
Follow the steps below and verify if you can view the log folder and log files of the "**Default Web Site**".

1. Open IIS, expand Sites, and make a note of the ID for the site you are checking.



2. Navigate to the IIS Log folder path and open the "**W3SVC1**" folder (**i.e**"**W3SVC**" + "**ID of the site**". Here the Default Web Site, ID is "**1**". So you should check the "**W3SVC1**" folder). Ensure you have access to all the files within this folder.

First, attempt to open the folder and files manually. If prompted for permission while accessing them, grant the necessary permission.



PART

Release Notes

What changed in each Applicare release.

RELEASE NOTES

Applicare Release Notes for Version 9.2.9

Applicare 9.2 · Updated February 4, 2026 · **Pending technical review**

- Adding the IIS site name in the Transaction alert
- We have added the capability to configure **JVM parameters and Memory allocation** via the UI console, eliminating the need for manual file edits.
- Integrated a new **Sentinel fallback** UI to provide a branded HTML experience instead of a default browser error when the Controller is updating.
- We've added a new storage management feature: visualize database table usage and reclaim space by deleting old data directly from the UI.
- Enhanced the Linux agent installation to gracefully handle **SELinux** constraints, validate minimum disk space requirements, and manage user permission limitations by automatically selecting suitable installation paths or falling back to non-systemd mode.
- Introduced visibility for **unattached servers** agents attempting to register beyond the allocated license capacity.

RELEASE NOTES

Applicare Release Notes for Version 9.2.8.4

Applicare 9.2 · Updated December 24, 2025 · **Pending technical review**

- Added a chart data that shows its Applicare own database growth.
- Introduced a new Management Dashboard that provides a unified view to monitor hardware metrics and HTTP requests for all application servers in a single screen.
- **Redis Monitoring**: Introduced comprehensive monitoring for Redis to track performance and health.
- Enhanced MS SQL database monitoring with new features, including error log tracking and user connection details
- Added the ability to download error reports filtered by IIS sites.
- Updated legacy libraries within the controller to ensure better performance and security.
- Added support for monitoring both Nginx and MS SQL databases using a single agent
- Introduced a new interface to perform controller updates directly from within the applicare.

RELEASE NOTES

Applicare Release Notes for Version 9.2.8.3

Applicare 9.2 · Updated July 3, 2025 · **Pending technical review**

- Removed unwanted jobs and files from controller.
- Fixed server configuration update.
- NodeJs agent configuration has been overhauled.
- Added the option to download the alert report with the message details in the Alert screen.
- Added the option to download the alerts based on the App or server in the reports screen for the intellitrace report and schedule report feature.
- Added Applicare controller disk details for linux.
- Fixed the JNDI error screen
- Modified the server availability functionality.
- Added the new reports to display the base url for the iis and nginx monitoring.
- Added the base url capture in report for the tomcat server
- Added the Hardware, App analyzer and Api report in custom report screen.
- We have newly implemented the Performance Insights screen to display critical issues related to the following areas: Alerts, Disk Usage, CPU Usage, Memory, TCP Connections, Slow Transactions, and Transaction Counts for HTTP, SQL, Web Services, and KBT data.
- Added the All and multiselect option in the intellitrace report.

RELEASE NOTES

Applicare Release Notes for Version 9.2.8.2

Applicare 9.2 · Updated March 7, 2025 · **Pending technical review**

- Order by functionality added to the App analyzer screen report download..
- Enable both IIS and database monitoring in one SingleAgent.
- We have implemented a two-minute delay for the startAgent, monitorAgent, StartController, and MonitorController tasks after a machine restart.
- Added a mechanism to exclude specific Oracle SQL statements from transaction monitoring using SQL IDs
- To improve monitoring capabilities, the Management Dashboard feature has been added.
- Introduced a feature to distribute email alerts to varying email groups, determined by alert severity.
- Implemented the Applicare System Info screen to provide controller system performance metrics.
- We have added the ability to identify SQL queries susceptible to injection attacks within all Java applications and the MSSQL database.

RELEASE NOTES

Applicare Release Notes for Version 9.2.8.1

Applicare 9.2 · Updated January 13, 2025 · **Pending technical review**

- LAMA Data post skip alert.
- Server Availability alert Suppression.
- Added the server counts on the Overview screen and sensors count on the manage sensors screen.
- Display the IIS and MSSQL server instance details on the Server Info screen.
- Display enable/disable snmp device status in network topology screen.
- Multiple MSSQL servers running on the same machine can be monitored.

RELEASE NOTES

Applicare Release Notes for Version 9.2.7.9

Applicare 9.2 · Updated November 5, 2024 · **Pending technical review**

- **Top Business Transaction Chart**: Added a new chart option on the enterprise screen to visualize top business transactions.
- **Transaction Score Throughput Values**: Introduced an option to display throughput values within the transaction score.
- **IIS Server Information**: Included detailed information for IIS servers on the server information screen.
- **UI Updates**: Implemented changes to the user interface on the download agent and Applicare information screens.
- **Agent Log Download Screen**: Added a dedicated screen on the Applicare information page for downloading agent logs.

Bug Fixes:

- **Error Capture**: Enhanced the capture of 500 errors in KBT (Knowledge Base Technology) and Intellisense.

RELEASE NOTES

Applicare Release Notes for Version 9.2.7.8

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

- Included Java with the Mac installation package.
- Discontinued the CAgent download.
- Updated the UI for the Performance Comparison screen.

RELEASE NOTES

Applicare Release Notes for 9.2.7.7

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

- Removed unnecessary .bat files from the Windows SingleAgent download package..
- Updated the UI for Server Analyzer and UEM Transaction Summary.

RELEASE NOTES

Applicare Release Notes for 9.2.7.6

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

- Added an option to enable the .NET agent during SingleAgent deployment.
- Moved .NET data transfer functionality to be handled via SingleAgent.
- Updated the UI for the Kubernetes monitoring screen.

RELEASE NOTES

Applicare Release Notes for Version 9.2.7.5

Applicare 9.2 · Updated August 23, 2024 · **Pending technical review**

- Configuring Okta Single sign-on with SAML 2.0 authentication.

RELEASE NOTES

Applicare Release Notes for Version 9.2.7.4

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

- Added an option in the LAMA Configuration screen to monitor only specific disks for hardware metrics.
- Enhanced LAMA Application metrics with logic to calculate Historical Throughput (HST_THRGHPT) and Historical Latency (HST_LTNCY) for a 21-day period, posting the data once daily.

RELEASE NOTES

Applicare Release Notes for Version 9.2.7.3

Applicare 9.2 · Updated August 22, 2024 · **Pending technical review**

- Introduced a feature for executing live SQL queries.
- Added the capability to stop the agent while updating the Windows agent using a PowerShell command.

RELEASE NOTES

Applicare Release Notes - Version 9.2.7

Applicare 9.2 · Updated September 2, 2024 · **Pending technical review**

- Added a Dark theme feature.
- Bug Fix: Resolved issue with SQL exclude option in transaction alerts.
- Bug Fix: Corrected issues on the Performance comparison screen.
- Bug Fix: Addressed ART data issues in the server analyzer screen.

RELEASE NOTES

Applicare Release Notes - Version 9.2.6

Applicare 9.2 · Updated September 2, 2024 · **Pending technical review**

- Introduced a Server Information tab in the Server Analyzer.
- Added of Network Topology to the Server Analyzer.
- Enhanced the LAMA data posting function.
- Bug Fix: Resolved an issue with IIS sensor pause functionality.
- Bug Fix: Corrected the custom server availability alert.
- Bug Fix: Addressed the alerts count notification issue.
- Bug fix: Fixed the alerts pause functionality.

RELEASE NOTES

Applicare Release Notes for 9.2.5

Applicare 9.2 · Updated March 8, 2024 · **Pending technical review**

- Improvement for LAMA UI and data posting function.
- Added the Auto KBT feature enable/disable option in JMS screen.
- Enhancements to JMS Monitoring.
- Enabled custom server availability.
- Uem shows App-server Name instead of the text "Application Server.
- Bug Fix: Fixed an issue with custom time selection.
- Bug Fix: Fixed an issue with intellisense trace.

RELEASE NOTES

Applicare Release Notes - Version 9.2.4

Applicare 9.2 · Updated September 2, 2024 · **Pending technical review**

- LAMA Support for National Stock Exchange (India)
- Fix for SLA Max and SLA AVG detection in KBT
- SQL fix for Kubernetes metrics monitoring in Derby and Oracle databases.
- Resolved the application dropdown issue on the Kubernetes screen.
- Set 1GB of default memory for SingleAgent

RELEASE NOTES

Applicare Release Notes for 9.2.2

Applicare 9.2 · Updated March 8, 2024 · **Pending technical review**

Applicare 9.2.2

- Enduser request timeout capture.
- Auto discovery/configuration of Apache and IBM http servers on Windows and Linux.
- Auto discovery/configuration of IIS.
- Auto assign app name to the newly added agents.
- Updated MySQL & MSSQL dashboard.

PART

Support & Tickets

How to get help, what to include, and how tickets work.

SUPPORT & TICKETS

How to get support

Applicare 9.2 · Updated October 20, 2026 · **Pending technical review**

Start with the documentation: use **Search** and try the error text, a technology name (for example `kubernetes`) or a part of the message you see. The **Troubleshooting** section lists known errors with their fixes.

If the documentation does not solve the problem, raise a ticket from the **Support & Tickets** page.

Connected deployments

When this Help Center can reach the support server you can:

- submit a ticket, with attachments and optional basic diagnostics;
- see **My tickets**, including status and every reply from Arcturus Support;
- reply to a ticket and add attachments.

You only ever see tickets that belong to your own organisation. Organisation administrators can see all tickets for their organisation; other users see the tickets they raised.

Offline and air-gapped deployments

An offline copy of the documentation **cannot submit a ticket**. Nothing is sent anywhere, and the page says so with a **NOT SUBMITTED** label.

1. Open **Support & Tickets** and fill in the request as usual.
2. Choose **Export request + diagnostic bundle** and save the file.
3. Move the file to a machine with network access using your organisation's approved transfer process.
4. Send it to Arcturus Support, by email or as an attachment to a new request in the Arcturus Help Center.
5. Quote the **reference code** shown in the file. The ticket exists only after Support confirms they received it.

> **Note**
> The export contains only what you typed plus the documentation version, build id, time, browser and page address. Review it before sending and remove anything sensitive.

SUPPORT & TICKETS

What to include in a support ticket

Applicare 9.2 · Updated October 20, 2026 · **Pending technical review**

A good ticket answers these questions:

- **What were you trying to do?** Name the feature or the guide you were following. The **Contact support** button on an article attaches the article for you.
- **What happened instead?** Copy the exact error text, including any numeric codes.
- **Which versions are involved?** The Applicare version, the operating system, and the application platform being monitored (for example .NET, Java, Docker or Kubernetes).
- **What changed recently?** Upgrades, new hosts, network or firewall changes.
- **How urgent is it?** Choose **Urgent** only when production is affected.
- **Screenshots or logs.** Attach them rather than pasting long output, and check them for passwords, tokens and keys first.

> **Warning**
>
> Never include passwords, API tokens, private keys or personal data in a ticket, an attachment or an exported bundle.