CONFIGURATION GUIDES › DOCKER

.NET in Docker Image

Applicare 10.0 · 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.