Application Deployment on AKS – Steps

cavproxy Setup

0. Build and push cavProxy Docker image to Azure Container Registry (ACR) (Optional)

```
az acr build --registry gmfpoc --image cavisson-proxy:latest --file
k8s-configs/Dockerfile.cavproxy .
gmfpoc.azurecr.io/cavisson-proxy:latest
```

1. Download Proxy Deployment YAML

```
wget https://nde.cav-test.com/HUB/GMF/cavproxy-deployment.yaml
```

Update Proxy Environment Variables

Edit the **cavproxy-deployment.yaml** file and update the following environment variables to match the target ND Controller details:

```
- CAV_APP_AGENT_NDCHOST: ND Controller Host IP
- CAV_APP_AGENT_NDCPORT: ND Controller Port

Sample configuration:
NDHOME=/opt/cavisson/netdiagnostics
LD_LIBRARY_PATH=/usr/lib/x86_64-linux-gnu
CAV_APP_AGENT_NDCHOST=66.220.31.146
CAV_APP_AGENT_NDCPORT=443
CAV_APP_AGENT_NDC_COMM_PROTOCOL=WSS
CAV_APP_AGENT_PROXYPORT=1111
```

2. Apply the Proxy Deployment

```
kubectl apply -f cavproxy-deployment.yaml
```

Verify Deployment, Pods, and Services

```
kubectl get pods -n cavisson
kubectl get deployment -n cavisson
kubectl get svc -n cavisson
```

Note The cavproxy service is exposed using a LoadBalancer service type. Please note the External IP address and port assigned to this LoadBalancer, as these details are required for configuring the Webhook Injector.

In case of new image and re-applying deployment without deleting
Triggering a Rollout Restart for a Kubernetes Deployment

```
kubectl rollout restart deployment cavproxy -n cavisson
```

ND webInjector Setup

Prerequisite: Ensure that helm is installed

```
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 |
bash
helm version
```

1. Download and Extract Webhook Injector Package

```
wget https://nde.cav-test.com/HUB/GMF/webhook-injector.tar.gz
tar -xvzf webhook-injector.tar.gz
```

2. Update cav-values.yaml

Navigate to the webhook-injector directory and edit the **cav-values.yaml** file. Update the proxy details under the NDController section as shown below:

```
NDController:
  proxyip: <Proxy External IP>
  proxyport: 1111
  proxymode: 1
```

3. Create a separate namespace for webhook injector

```
kubectl create namespace <namespace for webhook injector>
Eg: kubectl create namespace cavwebhook
```

4. Install Webhook Injector Using Helm

```
helm install cavwh . -n <namespace of injector> --values=cav-values.yaml
```

```
Eg: helm install cavwh . -n cavwebhook --values=cav-values.yaml

NAME: cavwh
LAST DEPLOYED: Wed Dec 24 01:26:25 2025
NAMESPACE: cavwebhook
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

5. Verify Webhook Injector Deployment

Confirm that the webhook injector components are running successfully:

```
kubectl get pods -n <namespace of injector>

kubectl get pods -n cavwebhook
kubectl get deployment -n cavwebhook
kubectl get svc -n cavwebhook
```

Note: To uninstall the Helm Release

Confirm that the webhook injector components are running successfully:

```
helm uninstall cavwh --namespace cavwebhook
```

Application-Level Deployment Configuration Changes

1. Application Side changes in Deployment.yaml

Example: eShop-deployment-alpine-injector.yaml

In the application deployment file, add the language and matchLabel under both the **selector** and **template** sections.

Sample:

```
  selector:
    matchLabels:
      app: eshopwebmvc-alpine
      language: dotnet
      matchLabel: dotnetApp
  template:
```

```
    metadata:
      labels:
        app: eshopwebmvc-alpine
        language: dotnet
        matchLabel: dotnetApp
```

These labels must be consistent in both sections to ensure proper pod selection and deployment behavior.

2. Apply Kubernetes deployment on AKS

```
kubectl apply -f k8s-configs/eShop-deployment-alpine-injector.yaml

kubectl get deployment -n cavisson
```

3.Deployment Rollout and Application Restart

```
kubectl rollout restart deployment <app-deployment-name> -n cavisson
```

4. Post-Deployment Validation

Based on the configured matching rules, the agent is automatically injected into the application pods. This can be verified by logging into the application pod and checking the environment variables using the `env` command.

```
env | grep CAV_
env | grep CORECLR
env | grep DOTNET
```

Successful validation confirms that the proxy and webhook injector are correctly configured and the application is ready for monitoring.

```
Sample o/p

/app # env| grep CAV_
CAV_APP_AGENT_PROXYIP=172.168.58.168
CAV_APP_AGENT_TIER=eshopwebmvc-alpine
CAV_APP_AGENT_PROXYPORT=1111
CAV_APP_AGENT_PROXYMODE=1
/app # env| grep CORECLR
```

```
CORECLR_ENABLE_PROFILING=1
CORECLR_PROFILER_PATH=/opt/cavisson/netdiagnostics/DotNetC/dll/ILRewritePro
filer.so
CORECLR_PROFILER={918728DD-259F-4A6A-AC2B-B85E1B658318}
/app # env| grep DOTNET
DOTNET_STARTUP_HOOKS=/opt/cavisson/netdiagnostics/DotNetC/dll/ProfilerHelpe
r.dll
DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false
DOTNET_RUNNING_IN_CONTAINER=true
DOTNET_CULTURE=en-US
DOTNET_VERSION=8.0.22
```

DotNetCore Application

1. Download the application package

```
wget https://nde.cav-test.com/HUB/dotnetCoreApp_latest.tar.gz
```

2. Extract the application

```
tar -xvzf dotnetCoreApp_latest.tar.gz
```

3. Build and push Docker image of Application to Azure Container Registry (ACR)

```
az acr build --registry mydotnetacr --image dotnetcoreapp:latest --file
Dockerfile .
```

OR

Build and push Docker image (only agent) to Azure Container Registry (ACR)

```
az acr build --registry mydotnetacr --image agent-init:latest --file
Dockerfile.cavagent .
```

Note:
List all Azure Container Registry (ACR) in the subscription

```
az acr list -o table
```

List **the repositories** (Docker image collections) stored inside your Azure Container Registry (ACR)

```
az acr repository list -n mydotnetacr -o table
```

4. Apply Kubernetes deployment on AKS

```
kubectl apply -f deployment.yaml -n gmf-dotnet
kubectl get deployment -n gmf-dotnet
```

In case of new image

Triggering a Rollout Restart for a Kubernetes Deployment

```
kubectl rollout restart deployment dotnetcoreapp-deployment -n gmf-dotnet
```

5. Verify the Pods

```
kubectl get pods -n gmf-dotnet
```

6. Expose the Deployment using LoadBalancer

```
kubectl expose deployment dotnetcoreapp-deployment -n gmf-dotnet
--type=LoadBalancer --port=80 --target-port=8080
```

7. Get the External IP

```
kubectl get svc dotnetcoreapp-deployment -n gmf-dotnet

Example:
NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)        AGE
dotnetcoreapp-deployment   LoadBalancer   10.0.22.243    135.237.25.166
80:30724/TCP   19s
```

8. Access the Application

Open in browser: http://135.237.25.166

eShop Application

1. Download the application package

```
wget https://nde.cav-test.com/HUB/eShopOnWeb-dotnet8.tar.gz
```

2. Extract the application

```
tar -xvzf eShopOnWeb-dotnet8.tar.gz
```

3. Build and push Docker image to Azure Container Registry (ACR)

```
For Ubuntu
----------
For Agent Only:
az acr build --registry gmfpoc --image agent-init:latest --file
k8s-configs/Dockerfile.dotnet8.Agent .
gmfpoc.azurecr.io/agent-init:latest

Note:
1. Above command to be executed when Agent image needs to be built/upgrade
2. Below commands to be executed at first time only.

For Application:
az acr build --registry gmfpoc --image eshopwebmvc:latest --file
src/Web/Dockerfile .
gmfpoc.azurecr.io/eshopwebmvc:latest

az acr build --registry gmfpoc --image eshoppublicapi:latest --file
src/PublicApi/Dockerfile .
```

```
gmfpoc.azurecr.io/eshoppublicapi:latest
```

```
For Alpine
—----------
For Agent Only:
az acr build --registry gmfpoc --image agent-init-alpine:latest --file
k8s-configs/Dockerfile.dotnet8.alpine-Agent .
gmfpoc.azurecr.io/agent-init-alpine:latest

Note:
1. Above command to be executed when Agent image needs to be built/upgrade
2. Below commands to be executed at first time only.

For Application:
az acr build --registry gmfpoc --image eshopwebmvc-alpine:latest --file
src/Web/Dockerfile.alpine .
gmfpoc.azurecr.io/eshopwebmvc-alpine:latest

az acr build --registry gmfpoc --image eshoppublicapi-alpine:latest --file
src/PublicApi/Dockerfile.alpine .
gmfpoc.azurecr.io/eshoppublicapi-alpine:latest
```

```
For AzureLinux
—-------------
For Agent Only:
az acr build --registry gmfpoc --image agent-init-azurelinux3:latest --file
k8s-configs/Dockerfile.dotnet8.azureLinux3-Agent .
gmfpoc.azurecr.io/agent-init-azurelinux3:latest

Note:
1. Above command to be executed when Agent image needs to be built/upgrade
2. Below commands to be executed at first time only.

For Application:
az acr build --registry gmfpoc --image eshopwebmvc-azurelinux:latest --file
src/Web/Dockerfile.azureLinux .
gmfpoc.azurecr.io/eshopwebmvc-azurelinux:latest

az acr build --registry gmfpoc --image eshoppublicapi-azurelinux:latest
--file src/PublicApi/Dockerfile.azureLinux .
```

```
gmfpoc.azurecr.io/eshoppublicapi-azurelinux:latest
```

Note:
To change the Agent version, Open **k8s-configs/Dockerfile.dotnet8.Agent** file.

4. Apply Kubernetes deployment on AKS

```
For Ubuntu
—---------
kubectl apply -f k8s-configs/eShop-deployment-init-container-agent.yaml

For Alpine
—---------
kubectl apply -f
k8s-configs/eShop-deployment-init-container-agent-alpine.yaml

For AzureLinux
—--------------
kubectl apply -f
k8s-configs/eShop-deployment-init-container-agent-azurelinux.yaml


kubectl get deployment -n cavisson
```

In case of new image and re-applying deployment without deleting.

Triggering a Rollout Restart for a Kubernetes Deployment

```
kubectl rollout restart deployment eshopwebmvc -n cavisson
kubectl rollout restart deployment eshoppublicapi -n cavisson
```

5. Verify the Pods

```
kubectl get pods -n cavisson
```

6. Expose the Deployment using LoadBalancer (Not required as service exposed in deployment.yaml)

```
kubectl expose deployment eshopwebmvc -n cavisson --type=LoadBalancer
--port=5106 --target-port=8080
```

7. Get the External IP

```
kubectl get svc -n cavisson
OR
kubectl get svc eshopwebmvc -n cavisson

Example:
NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)        AGE
eshopwebmvc   LoadBalancer   10.0.140.183   172.212.118.76   5106:31445/TCP
44h
```

8. Access the Application

Open in browser: http://172.212.118.76:5106/

Important commands for AKS

0. Authorization between your AKS cluster and your Azure Container Registry (ACR)

```
az aks update --name azure_linux --resource-group GMF-POC --attach-acr
gmfpoc

Where
Resource Type    Argument Flag     Correct Value     Source from Your
AKS Cluster Name      -n            azure_linux     az aks list -o table
AKS Resource Group    -g            GMF-POC         az aks list -o table
ACR Name           --attach-acr     gmfpoc          az acr list -o table
```

1. Delete the Deployment

```
kubectl delete deployment eshopwebmvc -n cavisson
kubectl delete deployment eshoppublicapi -n cavisson
```

2. Delete the Service (Optional)

```
kubectl delete service dotnetcoreapp-deployment -n cavisson
```

3. Verify Everything is Removed

```
kubectl get pods -n cavisson
kubectl get svc -n cavisson
kubectl get deployments -n cavisson
```

4.List all Azure Container Registry (ACR) in the subscription

```
az acr list -o table
```

5. List the repositories (Docker image collections) stored inside your Azure Container Registry (ACR)

```
az acr repository list -n gmfpoc -o table
```

6.To delete a specific repository

```
az acr repository delete -n gmfpoc --repository <repository-name> --yes
az acr repository delete -n gmfpoc --repository eshopwebmvc --yes
```

6.To delete a specific service

```
kubectl delete svc <service-name> -n <namespace>
kubectl delete svc eshopwebmvc -n cavisson
```

7. To list cluster in a table

```
az aks list -o table
az aks list --resource-group GMF-POC -o table
```

8. Filter by Resource Group

```
az aks list --resource-group GMF-POC -o table
```

.Net AgentUpgradation- eShop

1. Create a directory named **.NETAgent**.
2. Inside the **.NETAgent** directory, create the following subdirectories:
 a. **CavAgent-Dependencies** – for Ubuntu
 b. **alpine-CavAgent-Dependencies** – for Alpine
3. Download the required agent build version from CloudHub/BuildHub into the respective dependency directory:

```
wget https://nde.cav-test.com/HUB/dotnetAgent_4.14.1.18.tar.gz
```

4. Extract the downloaded build package.
5. Download the Ubuntu or Alpine-based agent Docker files from the .Net Agent Docker Files directory into the **.NETAgent** directory.
6. Build the **.NETAgent** Docker image using the downloaded Dockerfile (based on the OS type).
7. Update the application's **deployment.yaml** and add the `init.container` configuration.

For Ubuntu

```
initContainers:
        - name: agent-init
          image: gmfpoc.azurecr.io/agent-init:latest
          command: ["/bin/sh", "-c"]
          args:
            - |
              echo "Installing Cavisson .NET Agent..."

              mkdir -p /shared-agent/netdiagnostics/DotNetC/dll
              mkdir -p /shared-agent/netdiagnostics/DotNetC/lib
              mkdir -p /shared-agent/netdiagnostics/DotNetC/config

              cp /opt/cavisson/netdiagnostics/DotNetC/dll/*
/shared-agent/netdiagnostics/DotNetC/dll/
              cp /opt/cavisson/netdiagnostics/DotNetC/lib/*
/shared-agent/netdiagnostics/DotNetC/lib/
              cp /opt/cavisson/netdiagnostics/DotNetC/config/*
/shared-agent/netdiagnostics/DotNetC/config/
              cp /opt/cavisson/netdiagnostics/DotNetC/version
/shared-agent/netdiagnostics/DotNetC/

              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libwebsockets.so.19
```

```
/shared-agent/netdiagnostics/DotNetC/lib/libwebsockets.so
              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libapr-1.so.0.6.5
/shared-agent/netdiagnostics/DotNetC/lib/libcavapr-1.so.0

              chmod -R 755 /shared-agent/netdiagnostics
              echo "DONE"

          volumeMounts:
            - name: agent-volume
              mountPath: /shared-agent
```

For Alipine

```
initContainers:
        - name: agent-init-alpine
          image: gmfpoc.azurecr.io/agent-init-alpine:latest
          command: ["/bin/sh", "-c"]
          args:
            - |
              echo "Installing Cavisson .NET Agent..."

              mkdir -p /shared-agent/netdiagnostics/DotNetC/dll
              mkdir -p /shared-agent/netdiagnostics/DotNetC/lib
              mkdir -p /shared-agent/netdiagnostics/DotNetC/config

              cp /opt/cavisson/netdiagnostics/DotNetC/dll/*
/shared-agent/netdiagnostics/DotNetC/dll/
              cp /opt/cavisson/netdiagnostics/DotNetC/lib/*
/shared-agent/netdiagnostics/DotNetC/lib/
              cp /opt/cavisson/netdiagnostics/DotNetC/config/*
/shared-agent/netdiagnostics/DotNetC/config/
              cp /opt/cavisson/netdiagnostics/DotNetC/version
/shared-agent/netdiagnostics/DotNetC/

              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libwebsockets.so.19
/shared-agent/netdiagnostics/DotNetC/lib/libwebsockets.so
              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libcavapr-1.so.0.7.4
```

```
/shared-agent/netdiagnostics/DotNetC/lib/libcavapr-1.so.0
              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libcavapr-1.so.0.7.4
/shared-agent/netdiagnostics/DotNetC/lib/libcavapr-1.so
              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libuuid.so.1.3.0
/shared-agent/netdiagnostics/DotNetC/lib/libuuid.so.1
              ln -sf
/shared-agent/netdiagnostics/DotNetC/lib/libuuid.so.1.3.0
/shared-agent/netdiagnostics/DotNetC/lib/libuuid.so

              chmod -R 755 /shared-agent/netdiagnostics
              echo "DONE"

          volumeMounts:
            - name: agent-volume
              mountPath: /shared-agent
```

8. Apply the updated deployment file.
9. Roll out the deployment and verify the pod status.