Agent Deployment Guide

AKS

Overview **3**
Cavisson Proxy Setup 3
1. Download Proxy Deployment YAML 3
2. Apply the Proxy Deployment 3
ND webInjector Setup 4
1. Download and Extract Webhook Injector Package 4
2. Update cav-values.yaml 4
3. Create a separate namespace for webhook injector 4
4. Install Webhook Injector Using Helm 5
Application-Level Deployment Configuration Changes 5
2. Apply Kubernetes deployment on AKS 6
3.Deployment Rollout and Application Restart 6
Post-Deployment Validation 6

Overview

This document describes the end-to-end steps for deploying and managing applications on Azure Kubernetes Service (AKS) with Cavisson .NET agent auto-instrumentation. It covers setting up the Cavisson proxy, configuring the webhook injector using Helm, building and pushing application and agent Docker images to Azure Container Registry (ACR). The guide also includes rollout restart procedures, verification steps, service exposure, important AKS and ACR management commands, and detailed steps for .NET agent upgrade using init containers to ensure seamless monitoring and instrumentation of application pods.

Cavisson Proxy Setup

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

```
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
```

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