

Aarna Networks Multi-Cluster Orchestration Platform (AMCOP)

Quick Start Guide

Product Version: 3.1.09232022

Copyright © Aarna Networks, Inc. 2022



TABLE OF CONTENTS

INTRODUCTION	3
Server requirements	5
Prepare Deployment host	6
Package installation	7
Network Configuration	9
Installation of AMCOP	9
Ansible-based Installation	9
Operator based installation	13
Installation on RH Openshift Cluster	15
Installation on CRC (Cloud-ready Cluster)	15
Install Openshift CRC	15
Ansible based installation	17
Installation on OpenShift production cluster	18
Installing Monitoring Agent	18
Configure Backup and Restore of Cluster	20
AMCOP Deployment for PNF Management (Optional)	21
Verify AMCOP Deployment	23
AMCOP Deployment On High Availability (HA) Cluster	26
Creating a HA Kubernetes Cluster	26
Testing HA Functionality	29
Cleanup the setup	30
Troubleshooting	30
Production-grade deployment of AMCOP	31
Configuring Desktop/Laptop to access AMCOP portal	33
AMCOP Lifecycle functions	34
Upgrade/Downgrade	34
Cleanup	34

INTRODUCTION

This document explains Aarna Networks' Multi Cluster Orchestration Platform (AMCOP) quick-start installation and administration operations. It does not cover operational aspects of AMCOP, such as design and run-time CNF deployment, which are documented in the AMCOP User Guide.

AMCOP deployment can be done on a single server (all in one), a single VM, on a cloud (GKE, AKS etc.) or on multiple servers/VMs. This quick start guide covers installation on a single server or multiple servers. See the [AMCOP Cloud Quickstart Guide](#) to install it in cloud environments.

The installation can be done by either creating other KVM instances inside these servers (using Ansible scripts) or on existing Kubernetes clusters (using AMCOP Operator). In the case of using a single server/VM for installation, these servers/VMs need to support Nested Virtualization, since the installation creates additional VMs inside the host. It is possible to run these deployments without nested virtualization, but that requires customization of the installation/deployment procedure.

AMCOP uses the Kubernetes Operator (with Aarna's Ansible wrapper) for deployment, using Ubuntu cloud images for creating the Kubernetes cluster. It also supports Operator, which can be used to install AMCOP on any existing Kubernetes cluster.

The configuration in case of a single server (all-in-one) installation looks as follows:

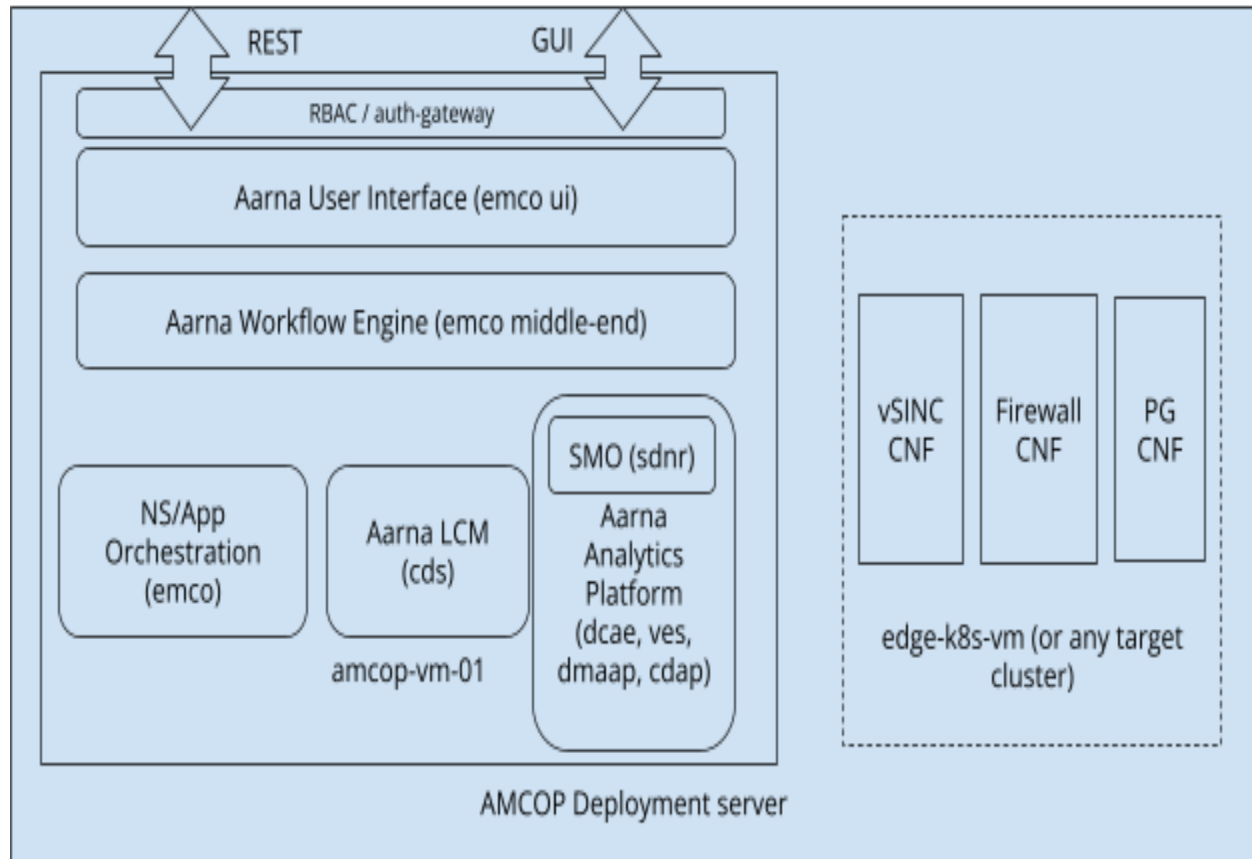


Figure 1: Single Server (or VM) deployment with KuD cluster

The *amcop-vm-01* shown above is used to run AMCOP. The *edge-k8s-vm* is a VM that runs a separate K8s cluster (“*target cluster*”) that can be used to instantiate various CNFs and CNAs. The *edge-k8s-vm* K8s cluster uses a KuD installer from the Akraino ICN project. The target cluster can be any other Kubernetes cluster (open source or commercial distribution), such as RedHat Openshift.

This document uses the following color coding for the commands to be executed by the user.

Install Jump host refers to the server from where the installation of an AMCOP cluster is done.

The VM *amcop-vm-XX* is the virtual machine where AMCOP is deployed.

Commands in blue font are for AMCOP Deployment server where installation is done, or Install Jump host from where installation is initiated. .

Commands in green courier font are for any AMCOP VMs (amcop-vm-XX).

Server requirements

The server(s) where AMCOP is going to be deployed will need the following requirements.

- CentOS 7.x or Ubuntu 18.04/Ubuntu 20.04 on the server
- Install Virtualization packages (KVM/QEMU)
- Login account with sudo permissions
- Optionally if AMCOP is to be deployed as a nested VM, the platform needs to support nested virtualization
 - This is the mode where AMCOP deployment scripts create a VM on the deployment host (and deploy AMCOP inside the VM), so if the deployment host itself is a VM, this will require nested virtualization.
 - Nested VM deployment is not recommended for production deployments. You have to take care of the networking from the nested VM (where AMCOP is deployed) to communicate to the external servers/VMs.
- Note: It is recommended to deploy AMCOP behind NAT for improved security.

Hardware requirements of the server:

Non-HA version:

- 16 vCPUs (logical cores)
- 32 GB RAM
- 150 GB Storage (preferably SSD)

HA version:

- 32 vCPU (logical cores)
- 64 GB RAM
- 500 GB Storage (preferably SSD)

Supported configurations

AMCOP installation is tested on the following configurations.

- Kubernetes version:
 - <= 1.21.0
- Kubernetes distributions:

- Kubeadm
- Kubespray
- KuD
- Openshift 4.6.9
- Cloud distributions (described in a separate guide - Cloud Quick start guide):
 - GKE
 - AKS
 - EKS
- Helm:
 - Helm2 and Helm3 (for CNF orchestration)

Note:

Ubuntu version 18.04 and 20.04 are supported and CentOS 7 is supported.. The Ubuntu/CentOS server should not have any pre-installed tools/software/packages.

Prepare Deployment host

Before starting deployment, one of the servers should be designated as the deployment host. In the case of a single server deployment, the same server can act as the deployment host.

The following steps need to be performed on the Deployment host.

- Install git tools

For Centos:

```
sudo yum install -y git deltarpm
```

For Ubuntu:

```
sudo apt-get install -y git
```

- Create a directory for downloading Aarna's deployment package:

```
mkdir -p ~/amcop_deploy  
cd ~/amcop_deploy
```

- Download the install package using the following link:

[amcop_install_v3.1.zip](#)

- Extract the .zip package.

```
unzip amcop_install_v3.1.zip
```

Package installation

Note:

This is optional and you can skip this if you have already installed virtualization tools and have nested VM capabilities enabled on all your target servers.

Note:

Nested Virtualization is not needed if AMCOP is directly deployed on a bare metal server or a VM, by setting up a k8s cluster.

The following steps on the AMCOP deployment server will install all the required packages for subsequent installation of AMCOP.

In case of multi-server deployment, below steps need to be performed on all the servers that will be part of AMCOP deployment.

Note:

Please do not use *root* user for AMCOP deployment. Instead, create a user with sudo privileges.

- [Optional] You need to enable nested KVM capabilities.

```
# You should have the nested KVM capabilities enabled  
cat /sys/module/kvm_intel/parameters/nested
```

```
Y
```

- Note: You need to make sure there is no ansible pre-installed on the deployment host before executing the next steps.
- Run the master install script as a normal user (to install all the required packages) on CentOS.

```
cd ~/amcop_deploy/aarna-stream/util-scripts
```

```
nohup ./prep_baremetal_centos.sh &
```

Note:

You can monitor the file nohup.out to check if the required package installation is complete. If the script fails to install the ansible tool with the error: "UnicodeEncodeError: 'ascii' codec can't encode character '\xe9' in position 112: ordinal not in range(128)",

Ubuntu:

Please check the value of parameter "LC_CTYPE" using command "locale".

If it is not set to value "en_US.UTF-8", please set it accordingly using the command: "export LC_CTYPE="en_US.UTF-8"" and rerun the script.

CentOS:

Please set the locale if it is not set

```
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
export LC_COLLATE=C
export LC_CTYPE=en_US.UTF-8
source /etc/bashrc
```

- Run the master install script as a normal user (to install all the required packages) on Ubuntu.

```
nohup ./prep_baremetal_ubuntu.sh &
```

Monitor the file nohup.out to check if the required package installation is complete

Network Configuration

Following are the network configuration requirements for the AMCOP installation:

- Make sure to update your firewall settings to allow internet traffic from the allocated AMCOP VM IPs.

Installation of AMCOP

AMCOP supports the following options for installation:

1. Ansible-based installation: This deployment option can be used to install AMCOP on a bare metal server or a Cloud VM, which does not have Kubernetes installed.
2. AMCOP Kubernetes Operator: This deployment option can be used to install AMCOP on an existing Kubernetes deployment, and also for the life-cycle management of AMCOP (eg., upgrades etc.).

Ansible-based Installation

AMCOP supports the Ansible-based installer which allows you to deploy using a single command.

This command can be run either directly on any of the AMCOP servers (where it is to be deployed), or via Install Jump host (eg., Linux laptop or desktop), where the servers are accessible directly or through VPN connection. This single Ansible command performs all the tasks mentioned below using Ansible playbooks.

Note:

You can start Ansible from one of the servers where AMCOP is going to be deployed, instead of a separate Install Jump host. This is referred to as a Deployment Server.

- Download cloud images
- Set up the images to run as VM cluster
- Set up Kubernetes cluster on the node
- Deploy AMCOP

The installer deploys AMCOP as a VM cluster on Deployment servers. This takes around 15-30 minutes to complete, depending on the bandwidth of the internet connection.

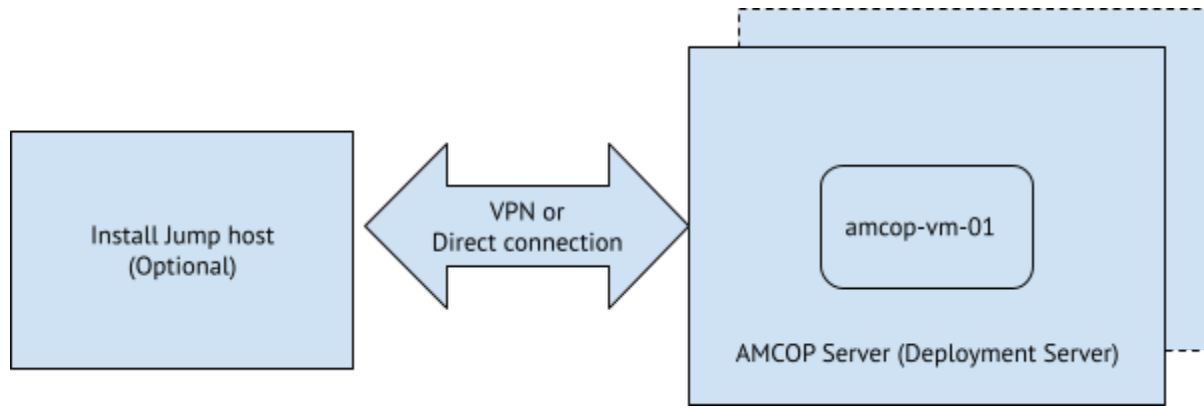


Figure 2: AMCOP Deployment configuration

Following are the steps to start the installation and deployment process. This will install AMCOP on the server.

Note:

In case we use a separate install jump host (different from the deployment server), these commands need to be run on the Install Jump host.

- Make sure you have public and private keys created and available under `~/.ssh/` folder. If not, create them using the command `ssh-keygen` program.

`ssh-keygen`

- Increase the SSH server timeout value to 120 seconds

```
echo "# Increase the server timeout value" >> ~/.ssh/config
echo "ServerAliveInterval 120" >> ~/.ssh/config
chmod 700 ~/.ssh/config
```

- Make sure you can ssh into AMCOP deployment host locally (without supplying password). This requires that `$HOME/.ssh` should contain ssh credentials (`id_rsa` and

id_rsa.pub), and the public key content (id_rsa.pub) is copied in the target server (\$HOME/.ssh/authorized_keys).

```
ssh <user_name>@localhost
```

```
# This should log you into the local server without a password prompt!  
# exit from ssh session  
exit
```

- Verify if you can SSH into other physical servers from AMCOP deployment host

```
ssh <user_name>@<target_host_ip>  
exit
```

- Run the ansible script that installs AMCOP. For non-HA deployment, it creates 1 VM (amcop-vm-01), with the following resources:
 - a. vCPUs = 16
 - b. Memory = 32 GB
 - c. Disk = 80GB

```
cd ~/amcop_deploy/aarna-stream/amcop_deploy/ansible/deployment
```

```
# Update inventory.ini with deployment host ip and user name  
vi inventory.ini
```

 AARNA
NETWORKS

```
deployment_host]  
10.11.16.11 ansible_user=aarna
```

- For default configuration, run the following command. This will create a VM with the name amcop-vm-01 (with user name “ubuntu”), and deploy AMCOP on it.

Note: If you are running the ansible command on the deployment host directly, the *jump_host_user* parameter is the same as *ansible_user* from the previous step.

```
# Run the Ansible command to start deployment  
nohup ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=on-prem -e  
jump_host_user=<user_name_on_jump_host> &
```

- [Optional] For non-default configuration, you can pass the different environment variables to Ansible, as shown in below examples.

```
# Default config with different server (VM) name parameter
```

```
ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=on-prem -e
jump_host_user=<user_name_of_deployment_host> -e server_name=<vm-name>
```

```
# Default config with a different user name parameter
# This will create a new user.
```

```
ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=on-prem -e
jump_host_user=<user_name_of_deployment_host> -e vm_user=<vm-user-name>
```

```
# Default config with a different server (VM) name and user name
```

```
ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=on-prem -e
jump_host_user=<user_name_of_deployment_host> -e server_name=<vm-name> -e
vm_user=<vm-user-name>
```

- [Optional] For non-default configuration, you can also skip some of the playbooks of Ansible. For example, if you want to deploy on an existing VM (running Ubuntu 18.04), you can skip the creation of VMs. When an existing VM is used, the VM name should be passed as an argument to the playbook. If the VM IP cannot be obtained through the virsh command, the IP should be updated in the configuration file. The configuration file is placed in `<dir>/aarna-stream/amcop_deploy/ansible/deployment/config/deployment.json` "ip-address" should be updated to reflect the IP of the VM.

Command to skip VM creation during AMCOP deployment:

```
ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=on-prem -e
jump_host_user=<user_name_of_deployment_host> -e server_name=<server_name> -e
vm_user=<vm-user-name> --skip-tags vm
```

- On success you should get the below success message from the ansible log file

```
tail -f nohup.out
```

```
PLAY RECAP
```

```
*****
192.168.102.85 : ok=28 changed=8 unreachable=0 failed=0 skipped=9
rescued=0 ignored=0
```

- Execute the below commands on the AMCOP master node to verify if ONAP k8s services are fully functional.

```
# You can find out the IP address of the VM where AMCOP is setup
# as follows, if the nested deployment ansible command is executed:
```

```
sudo virsh list --all
```

Id	Name	State
7	amcop-vm-01	running

```
sudo virsh domifaddr <vm-name>
```

For example:

```
sudo virsh domifaddr amcop-vm-01
```

Name	MAC address	Protocol	Address
vnet0	52:54:00:95:7e:54	ipv4	192.168.100.48/24

```
ssh ubuntu@<amcop-VM-ip>
```

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
node1	Ready	control plane,master	7h38m	v1.21.0

Operator based installation

AMCOP can also be installed using a Kubernetes Operator, instead of the above Ansible approach, if you already have an existing Kubernetes cluster deployed on a server (with required configuration), or for future upgrades of an existing AMCOP deployment.

Pre-requisites:

A Kubernetes cluster, which will host AMCOP deployment is a requirement. AMCOP is validated to work with a minimum single node all-in-one cluster with the following hardware configuration:

- 16 vCPUs, 32 GB RAM and 80 GB SSD

Deployment steps:

- AMCOP requires the usage of persistent volumes to manage stateful information. These persistent volumes are required to be provided by a default storage class

configured with a persistent volume provisioner. To create your own storage class with a persistent volume provisioner, execute the following command:

```
kubectl apply -f  
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/storage.yaml
```

- Check for the deployed storage class:

```
kubectl get storageclass
```

- Once availability of storage class is ensured, you need to roll out AMCOP Operator itself using the following command:

```
kubectl apply -f  
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/operator.yaml
```

- Now, create the Custom Resource using the below command. Upon creation of this Custom Resource, AMCOP Operator starts deployment of various AMCOP components in a staged manner.

```
kubectl apply -f  
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/default.yaml
```

Note:

The above commands will also work if you have AMCOP 3.0.0 patch release (#07222022) already deployed. In such a case AMCOP will be upgraded to version 3.1 while retaining the state.

- Progress of deployment can be monitored by watching Kubernetes pods in the amcop-system namespace or by watching the status of Custom Resource itself using the below command:

```
kubectl get installer.amcop
```

Note: When all the components of AMCOP are rolled out successfully, the status of deployment (above command output) will change from "RollingOut " to "Deployed" and you can start using AMCOP.

Installation on RH Openshift Cluster

AMCOP can be deployed on the RH Openshift cluster, either on a CRC (Cloud-ready Cluster) or production deployment of Openshift. The CRC-based deployment is typically used for development/POC purposes.

For deploying on the RedHat Openshift cluster, you can use the Openshift Client CLI command line tool to deploy AMCOP components. This command line is executed using the 'oc' tool. Aarna's Ansible-based installation command downloads and sets up CLI and deploys AMCOP automatically on the Openshift cluster.

Installation on CRC (Cloud-ready Cluster)

Prerequisites for the CentOS 7 server to deploy AMCOP is to have the virtualization capabilities like KVM installed on the bare metal server.

- Install KVM prerequisites on CentOS 7 server.
- Download CRC secret required for deploying Openshift cluster:

You can use the below script to install all the necessary tools on the deployment host.

```
cd <dir>/aarna-stream/util-scripts/  
  
./prep_baremetal_centos.sh
```

Note:

RedHat Openshift was tested with Openshift Code Ready Container (CRC) 4.6.9 version. It is supported on CentOS 7.x platform only

Install Openshift CRC

You can follow the below steps to bring up the Openshift CRC cluster. If you already have a running Openshift CRC cluster, you can skip this step and go to the next subsection on AMCOP installation.

- Download the openshift crc tar package as below:
`wget https://developers.redhat.com/content-gateway/rest/mirror/pub/openshift-v4/clients/crc/latest/crc-linux-amd64.tar.xz`
- Untar the downloaded package:
`tar -xvf crc-linux-amd64.tar.xz`
- Run the crc setup command (input “y” when prompted):
`./crc setup`
- Set/configure the crc pull secret
`./crc config set pull-secret-file <path to the crc pull secret file>`
- Set/configure other crc config parameters:
`./crc config set disable-update-check false`
`./crc config set cpus 32`
`./crc config set memory 134,217,728`
`./crc config set disk-size 500`
- Check the crc configuration:
`./crc config view`
- Start the crc cluster deployment:
`./crc start`
- At the end of the successful CRC cluster deployment, oc login credentials are displayed at the console for both “kubeadmin” and “developer” users”. If required, use them to login to the crc cluster VM brought up by the crc deployment.
- Perform oc login:
 - `eval $(./crc oc-env)`
 - `oc login -u kubeadmin -p <kubeadmin user password from crc start output> api.crc.testing:6443`
- Check the projects list:
`oc projects`

Note:

To overcome the docker hub rate limiting for free accounts, you can use your personal or organization's docker hub credentials for AMCOP deployment on the openshift CRC cluster using the command below.

Note: You need to ensure that the crc cluster is up and running.

```
oc create secret docker-registry docker --docker-server=docker.io
--docker-username=<user> --docker-password=<password> --docker-email=<email> -n
amcop-system
```

And link the secret created in the above command as below:

```
oc secrets link amcop docker --for=pull -n amcop-system
```

- ssh into the crc cluster VM:

```
ssh -i ~/.crc/machines/crc/id_ecdsa core@192.168.130.11
```
- Logout of the crc cluster VM:

```
Logout
```

Ansible based installation

Now you can deploy AMCOP on the Openshift CRC cluster using the following steps from the bare metal CentOS server. This is the simplest way to get started with AMCOP on Openshift.

- Copy the id_ecdsa key generated by the crc cluster to ~/.ssh directory on the CentOS 7 bare metal server.

```
cp ~/.crc/machines/crc/id_ecdsa ~/.ssh/id_ecdsa
```
- Update the inventory.ini file accordingly with deployment host details
Note: You should have downloaded and unzipped the amcop zip file on the bare metal CentOS server.

```
cd ~/amcop_deploy/aarna-stream/amcop_deploy/ansible/deployment
```

```
# Update inventory.ini with deployment host ip and user name
vi inventory.ini
```

```
[deployment_host]
10.11.16.11 ansible_user=aarna
```

- Deploy AMCOP using ansible script:

```
ansible-playbook -v ./main.yml -i inventory.ini -e deployment_env=opshift -e
opshift_oc_user_name=kubeadmin -e opshift_oc_password=<kubeadmin user
password from crc start output> -e opshift_oc_url=https://api.crc.testing:6443 -e
opshift_oc_insecure_flag=true
```

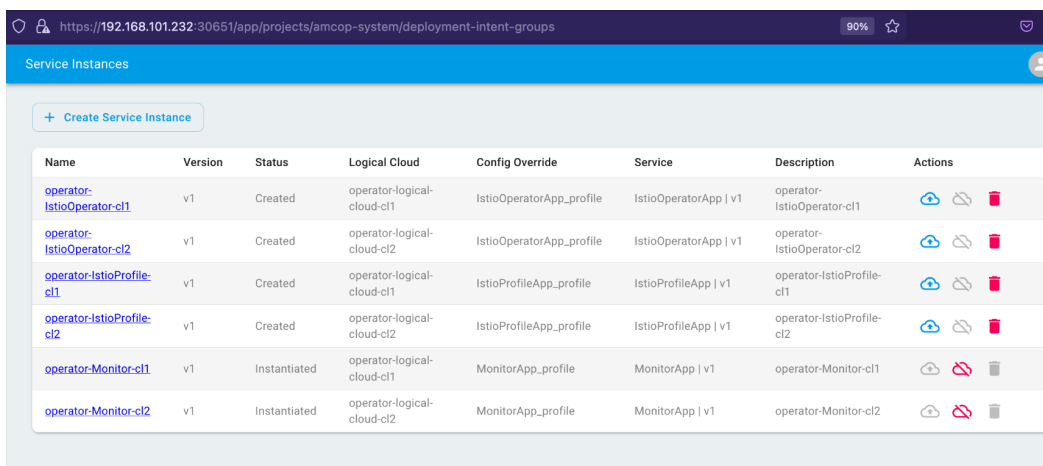
Installation on OpenShift production cluster



















The procedure to install AMCOP on the OpenShift production cluster is to use the AMCOP Operator-based installation.

Please refer to the section on Operator based installation.

Installing Monitoring Agent

AMCOP supports a monitoring agent. As part of AMCOP deployment, a default service instance in the amcop-system tenant for the monitoring agent will be auto-created for every cluster that is onboarded to AMCOP.

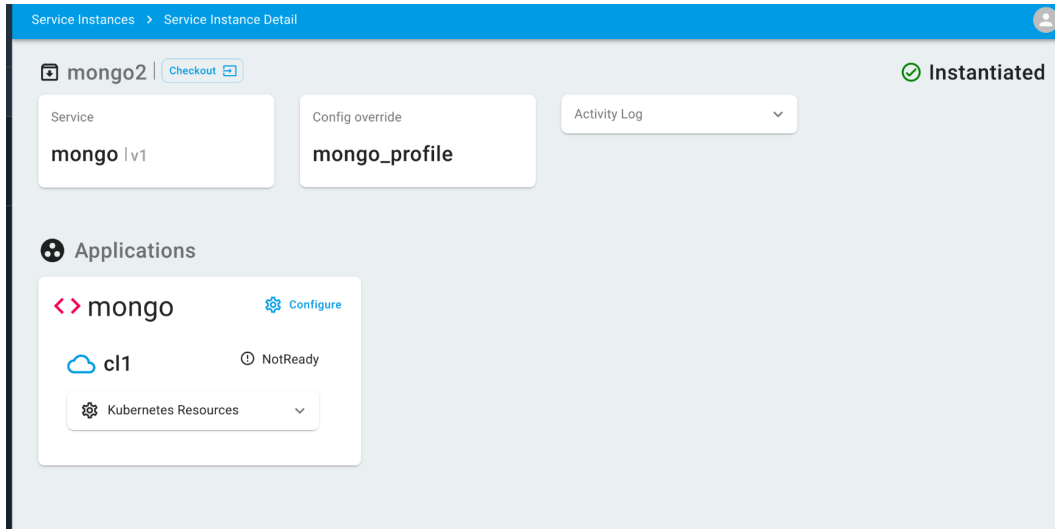


Name	Version	Status	Logical Cloud	Config Override	Service	Description	Actions
operator-IstioOperator-cl1	v1	Created	operator-logical-cloud-cl1	IstioOperatorApp_profile	IstioOperatorApp v1	operator-IstioOperator-cl1	  
operator-IstioOperator-cl2	v1	Created	operator-logical-cloud-cl2	IstioOperatorApp_profile	IstioOperatorApp v1	operator-IstioOperator-cl2	  
operator-IstioProfile-cl1	v1	Created	operator-logical-cloud-cl1	IstioProfileApp_profile	IstioProfileApp v1	operator-IstioProfile-cl1	  
operator-IstioProfile-cl2	v1	Created	operator-logical-cloud-cl2	IstioProfileApp_profile	IstioProfileApp v1	operator-IstioProfile-cl2	  
operator-Monitor-cl1	v1	Instantiated	operator-logical-cloud-cl1	MonitorApp_profile	MonitorApp v1	operator-Monitor-cl1	  
operator-Monitor-cl2	v1	Instantiated	operator-logical-cloud-cl2	MonitorApp_profile	MonitorApp v1	operator-Monitor-cl2	  

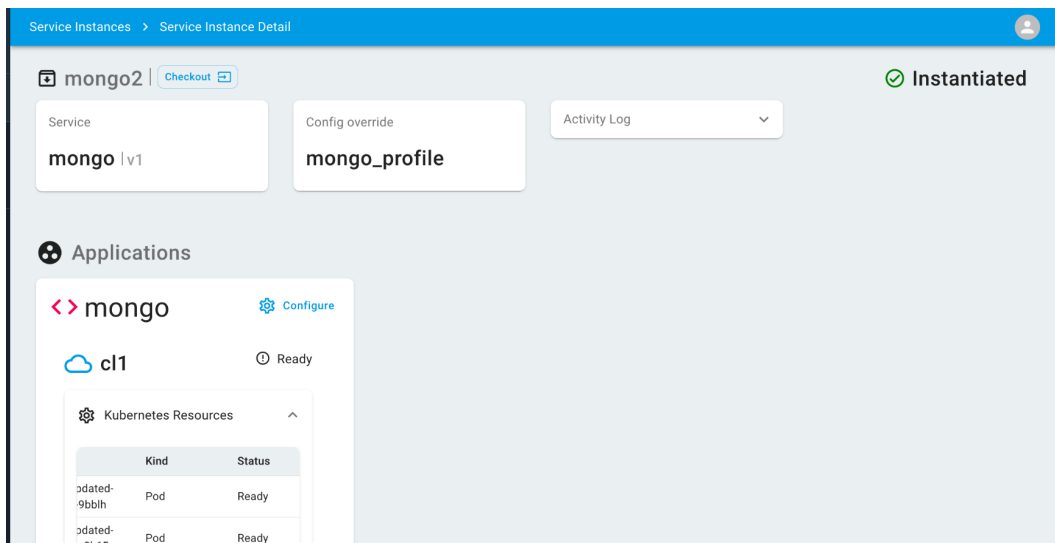
Once you initialize this service instance, the monitoring agent will be deployed in the target cluster.

Example of how the monitoring agent will change the status update behaviour in AMCOP:

1. Deploy a service instance mongo. The status of the K8s components will show as *NotReady* while the pod is in an initialization state (as an example).



2. Refreshing the screen again (after few seconds) will show the resources as ready:



So with the monitoring agents deployed, the service instance page will show the real status of the K8s resources which are orchestrated via AMCOP.

Configure Backup and Restore of Cluster

As a part of the installation package, you will find a script to download and configure the Velero tool. Velero is used for taking a backup of the entire cluster state of your AMCOP deployment onto a AWS S3 bucket. The stored backup can be used to restore the state of the cluster. This is primarily useful in case of Disaster Management.

The script is located at: (aarna-stream/amcop_deploy/scripts/velero-configure-aws.sh)

Before running the backup script, ensure the following.

1. Access to an AWS account capable of creating S3 buckets. These buckets are used for storing the backup.
2. AWS cli installed and configured with access keys.
3. Access to the K8 cluster along with kubeconfig. This is the cluster that needs to be backed up.

Run the above script from the location where the cluster is accessible. This script will download the velero tool and install it on the cluster. In addition to that, it will configure storage in AWS for storing the backup.

At any point, use the below command to create a backup of the current state of the cluster.

```
velero backup create <name of the backup>
```

At a future time, the backup can be restored using the following command.

```
velero restore create --from-backup <name of the backup>
```

Visit the Velero documentation (<https://velero.io/docs/v1.8/>) for more details about and additional commands.

AMCOP Deployment for PNF Management (Optional)

AMCOP can be used for the management of PNF devices along with CNFs through the deployment of the below optional components using the following steps.

Note:

The server resources need to be increased before deploying optional AMCOP PNF management components. Total vCPUs and memory required for AMCOP deployment including these optional components are vCPUs: 24 and Memory: 128 GB.

1. Install helm3:

Install helm version 3 in the VM where AMCOP is deployed.

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh --version v3.5.2
```

2. Deploy below optional components:

Download helm charts for the optional components from [here](#) into the \$HOME in the server where AMCOP is deployed.

Unzip the downloaded folder and copy the helm charts into the \$HOME.

Install the PNF management components one by one as below.

3. Install commons module:

```
helm install -n amcop-system common common-0.1.0.tgz
```

Wait for ~2 min before proceeding with the next steps.

4. Install Cassandra DB module:

```
helm install -n amcop-system cassandara cassandra-8.0.0.tgz
```

Wait for the Cassandra DB pods to be up in the “running” state.

5. Install AAF module:

```
helm install -n amcop-system aaf aaf-8.0.0.tgz
```

Wait for the AAF pods to be up in “running” state.

6. Install A&AI module:

```
helm install -n amcop-system aai aai-8.0.0.tgz
```

Wait for the A&AI pods to be up in “running” state.

Verify AMCOP Deployment

You will be able to start using AMCOP after verifying that the deployment is successful, by running the following commands.

```
kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
amcop-operator	amcop-operator-controller-manager-5985dcfd45-kfzjf	2/2	Running	0	26h
amcop-storage	local-path-provisioner-6696f697ff-2gjrp	1/1	Running	0	26h
amcop-system	camunda-camunda-6474bddd94-5qpfb	1/1	Running	0	26h
amcop-system	cds-blueprints-processor-d4dbdc899-8gx6z	1/1	Running	0	26h
amcop-system	cds-py-executor-5f74484868-jhvw6	1/1	Running	0	26h
amcop-system	cds-ui-7d9f747d7b-wp6zk	1/1	Running	0	26h
amcop-system	clm-5b77b7f669-twcjq	1/1	Running	0	26h
amcop-system	configsvc-54df6bb478-zqtd2	1/1	Running	0	26h
amcop-system	datafile-collector-7b4457f94c-kgh5z	1/1	Running	0	26h
amcop-system	dcm-56df45fc76-gngzp	1/1	Running	0	26h
amcop-system	dmaap-5755b8d44b-fz2z4	1/1	Running	0	26h
amcop-system	dtc-85474c75b9-xjq4r	1/1	Running	0	26h
amcop-system	emcoui-9b468f496-qv8z4	1/1	Running	0	26h
amcop-system	etcd-0	1/1	Running	0	26h
amcop-system	gac-6c4b85c8c8-lvc2l	1/1	Running	0	26h
amcop-system	kafka1-5bb4bc7b9c-9z8gz	1/1	Running	0	26h
amcop-system	mariadb-galera-0	1/1	Running	0	26h
amcop-system	mariadb-galera-1	1/1	Running	0	26h
amcop-system	mariadb-galera-2	1/1	Running	0	26h
amcop-system	middleend-678bd464fd-q4765	1/1	Running	0	26h
amcop-system	mongo-0	1/1	Running	0	26h
amcop-system	ncm-7d9f5c6d56-q2zrk	1/1	Running	0	26h
amcop-system	orchestrator-7cf69df56d-cz758	1/1	Running	0	26h
amcop-system	ovnaction-75868df774-jpzm5	1/1	Running	0	26h
amcop-system	rsync-7d87c464dd-sxdrs	1/1	Running	0	26h
amcop-system	sbackend-556d4b7d4d-tp2rm	1/1	Running	0	26h
amcop-system	sdnr-6f76d567f5-ljgvn	1/1	Running	0	26h
amcop-system	sdnrdb-0	1/1	Running	0	26h
amcop-system	vescollector-9cf8cc877-ssr5k	1/1	Running	0	26h
amcop-system	zookeeper-54f97d94d8-q6lv8	1/1	Running	0	26h
kube-system	calico-kube-controllers-867d8d6bd8-w65ks	1/1	Running	0	26h
kube-system	calico-node-kxrmv	1/1	Running	0	26h

```

kube-system   coredns-558bd4d5db-lpjgg           1/1   Running 0    26h
kube-system   coredns-558bd4d5db-vfgvg           1/1   Running 0    26h
kube-system   etcd-ip-172-31-87-80                1/1   Running 0    26h
kube-system   kube-apiserver-ip-172-31-87-80      1/1   Running 0    26h
kube-system   kube-controller-manager-ip-172-31-87-80 1/1   Running 0    26h
kube-system   kube-proxy-trjzz                    1/1   Running 0    26h
kube-system   kube-scheduler-ip-172-31-87-80      1/1   Running 0    26h

```

```
kubectl get svc -n amcop-system
```

```

NAME              TYPE      CLUSTER-IP    EXTERNAL-IP  PORT(S)
AGE
camunda           NodePort  10.104.64.247 <none>       8443:31661/TCP
26h
cds-blueprints-processor-cluster ClusterIP  10.111.205.95 <none>       5701/TCP
26h
cds-blueprints-processor-grpc ClusterIP  10.105.10.0   <none>       9111/TCP
26h
cds-blueprints-processor-http ClusterIP  10.102.70.114 <none>       8080/TCP
26h
cds-py-executor   ClusterIP  10.108.19.65  <none>       50052/TCP,50053/TCP
26h
cds-sdc-listener  ClusterIP  10.107.150.67 <none>       8080/TCP
26h
cds-ui            NodePort  10.98.98.58   <none>       3000:30497/TCP
26h
clm               NodePort  10.102.135.169 <none>       9061:30461/TCP
26h

configsvc         NodePort  10.103.187.197 <none>       9082:30482/TCP
26h
datafile-collector NodePort  10.98.188.219  <none>
8443:30232/TCP,8100:30756/TCP
26h
dcm               NodePort  10.97.169.238  <none>
9078:30478/TCP,9077:30477/TCP
26h
dmaap             NodePort  10.102.144.141 <none>
3904:32392/TCP,3905:30768/TCP
26h
dtc               NodePort  10.96.153.88   <none>
9048:30483/TCP,9018:30492/TCP
26h
emcoui            ClusterIP  10.106.24.40  <none>       9080/TCP
26h

```


etcd 26h	ClusterIP	10.109.201.179	<none>	2379/TCP,2380/TCP
etcd-headless 26h	ClusterIP	None	<none>	2379/TCP,2380/TCP
gac 9033:30493/TCP,9020:30491/TCP 26h			NodePort	10.100.207.30 <none>
kafka1 26h	ClusterIP	10.108.138.74	<none>	9092/TCP
mariadb-galera 26h	ClusterIP	10.101.9.204	<none>	3306/TCP
mariadb-galera-headless 4567/TCP,4568/TCP,4444/TCP 26h			ClusterIP	None <none>
middleend 26h	ClusterIP	10.110.218.59	<none>	9051/TCP
mongo 26h	ClusterIP	None	<none>	27017/TCP
mongo-read 26h	ClusterIP	10.108.57.1	<none>	27017/TCP
ncm 9082:30489/TCP,9081:30431/TCP 26h			NodePort	10.102.56.144 <none>
orchestrator 9016:30416/TCP,9015:30415/TCP 26h			NodePort	10.109.29.100 <none>
ovnaction 9053:30473/TCP,9051:30471/TCP 26h			NodePort	10.96.209.117 <none>
rsync 26h	NodePort	10.102.120.124	<none>	9031:30441/TCP
sbackend 26h	NodePort	10.101.101.214	<none>	5000:30661/TCP
sdnr 8101:30101/TCP,8181:30181/TCP,4334:30334/TCP 26h			NodePort	10.106.139.113 <none>
sdnrdb 26h	ClusterIP	10.105.217.86	<none>	9200/TCP,9300/TCP
vescollector 26h	NodePort	10.107.64.30	<none>	8080:31080/TCP
zookeeper 26h	ClusterIP	10.105.119.28	<none>	2181/TCP

AMCOP Deployment On High Availability (HA) Cluster

To ensure that AMCOP services are not impacted by failures at the infrastructure level, you can deploy AMCOP in High Availability (HA) mode. Such a setup requires more resources as there will be multiple replicas of each component.

For deployment, there are two options:

1. Use a HA Kubernetes cluster (already installed) and simply deploy a HA version of AMCOP on it using the operator.
2. Build your own HA-enabled Kubernetes cluster and then deploy AMCOP on it.

The following subsection describes the mechanism to build your own HA cluster (Option 2 from above).

Creating a HA Kubernetes Cluster

To build and verify the HA properties of AMCOP we will require a server or VM with the below configurations.

1. 32 logical cores (or more)
2. 64GB of RAM (or more)
3. 500 GB of Disk space (preferably SSD)
4. Internet access from the server (to download and deploy packages)
5. Ubuntu 20.04 is installed on the server.

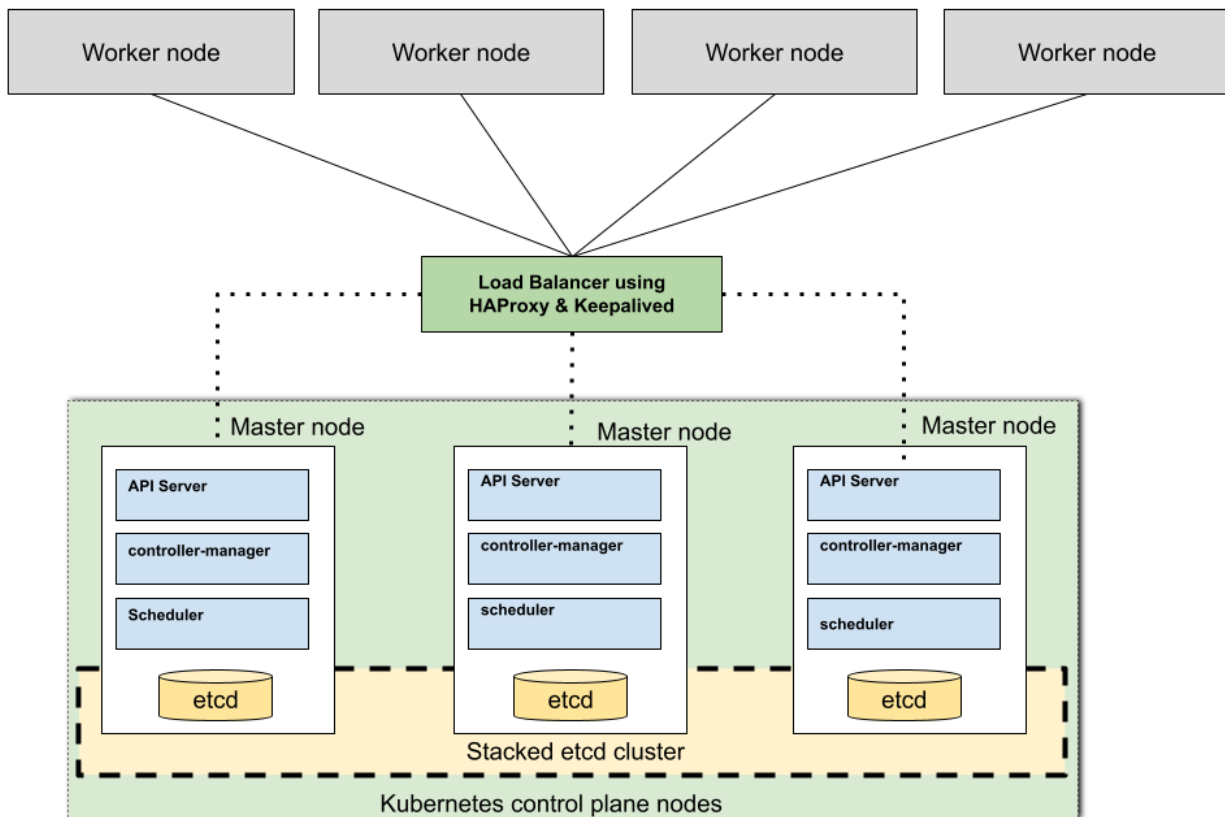


Figure 3: Multi-node Kubernetes Cluster with High-Availability

To build the cluster, you will need to create 7 VMs using QEMU, deploy Kubernetes cluster software on these VMs and use *kubeadm* to build a Kubernetes cluster. Each VM acts as a node of the Kubernetes cluster. The below script installs all the packages, builds the cluster and installs all the tools for High Availability of master nodes.

```
nohup ansible-playbook -vv -i inventory.ini main.yml -e deployment_env=on-prem -e
jump_host_user=<user_name_of_deployment_host> -e server_name=<server_name> -e
vm_user=<vm-user-name> -e enable_ha=true --skip-tags vm &
```

Note: The script will take around 20 mins (or more depending on server speed)

Once the above script completes, the cluster is ready. You can use the following commands to locate the IP address of the master node.

```
sudo virsh domifaddr NODE0
```

Name	MAC address	Protocol	Address
vnet0	52:54:00:d9:b1:bf	ipv4	192.168.122.28/24

Next we log into the master node and deploy the operator.

```
# Login to the Master node
```

```
# ssh ubuntu@192.168.122.28
```

```
ssh ubuntu@<master node ip addr>
```

```
# Run the following kubectl commands to deploy the Operator.
```

```
kubectl apply -f
```

```
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/storage.yaml
```

```
namespace/amcop-storage created
serviceaccount/local-path-provisioner-service-account created
clusterrole.rbac.authorization.k8s.io/local-path-provisioner-role created
clusterrolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind created
deployment.apps/local-path-provisioner created
storageclass.storage.k8s.io/amcop-local-path created
configmap/local-path-config created
```

```
kubectl apply -f
```

```
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/HAoperator.yaml
```

```
namespace/amcop-operator created
customresourcedefinition.apiextensions.k8s.io/installers.amcop.aarnanetworks.com created
role.rbac.authorization.k8s.io/amcop-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/amcop-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/amcop-operator-metrics-reader created
clusterrole.rbac.authorization.k8s.io/amcop-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/amcop-operator-leader-election-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/amcop-operator-manager-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/amcop-operator-proxy-rolebinding created
configmap/amcop-operator-manager-config created
service/amcop-operator-controller-manager-metrics-service created
deployment.apps/amcop-operator-controller-manager created
```

```
kubectl apply -f
```

```
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/defaultHA.yaml
```

```
installer.amcop.aarnanetworks.com/default created
```

Note: The above steps are similar to installation of AMCOP using operator

Once the above deployment is complete, you will see multiple replicas of various pods.

```
kubectl get pod -n amcop-system | grep emcoui
```

```
emcoui-755f98bc7-d2z9s      1/1   Running   0           6h52m
emcoui-755f98bc7-d8ccp      1/1   Running   2 (6h13m ago) 6h52m
```

Testing HA Functionality

To test whether the high availability is working, you can shut down one of the nodes. On a normal setup, this will cause the pods to be down for a significant duration. However, since this is a HA setup, there will be no visible downtime in the availability of service.

Use the below command to bring down one of the nodes.

```
sudo virsh list --all
```

```
Id Name State
-----
1  NODE0 running
2  NODE1 running
3  NODE2 running
4  NODE3 running
6  NODE5 running
7  NODE6 running
8  NODE4 running
```

Manually bring down one of the nodes (eg., NODE6)

```
sudo virsh destroy NODE6
```

```
Domain NODE6 destroyed
```

Now log into the master node and check the state of nodes. You will find that one of the nodes is going into <NoReady> state.

```
kubectl get node
```

```
NAME           STATUS   ROLES    AGE  VERSION
k8s-master-0  Ready   control-plane  18h  v1.25.1
```

```
k8s-master-1 Ready control-plane 18h v1.25.1
k8s-master-2 Ready control-plane 18h v1.25.1
k8s-worker-3 Ready <none> 18h v1.25.1
k8s-worker-5 Ready <none> 18h v1.25.1
k8s-worker-6 NotReady <none> 18h v1.25.1
k8s-worker-4 Ready <none> 54m v1.25.1
```

While the node is not ready, you can try to access the AMCOP GUI, and it should be accessible and fully functional.

<http://<master node ip>:30219/>
e.g.
<http://192.168.122.28:30219/>

Note: AMCOP HA deployment can be used to connect and exercise the SMO functionality of AMCOP using RU/DU/CU simulators only. Also, in the current release AMCOP upgrade in HA deployment is not supported.

Cleanup the setup

You can use the following commands to destroy the setup by deleting all the VM's.

```
sudo virsh destroy <VM NAME>
sudo virsh undefine <VM NAME>
```

Troubleshooting

If one or more VMs fail to boot up, you can look into the following log file to identify the cause of failure.

[./aarna-stream/amcop_deploy/logs/ha_deployment.log](#)

Production-grade deployment of AMCOP

AMCOP can be deployed in a single node (non-HA) configuration, or as a multi-node HA configuration. The non-HA configuration is only suggested for development and POC purposes, whereas the HA configuration is recommended for any production-grade deployments.

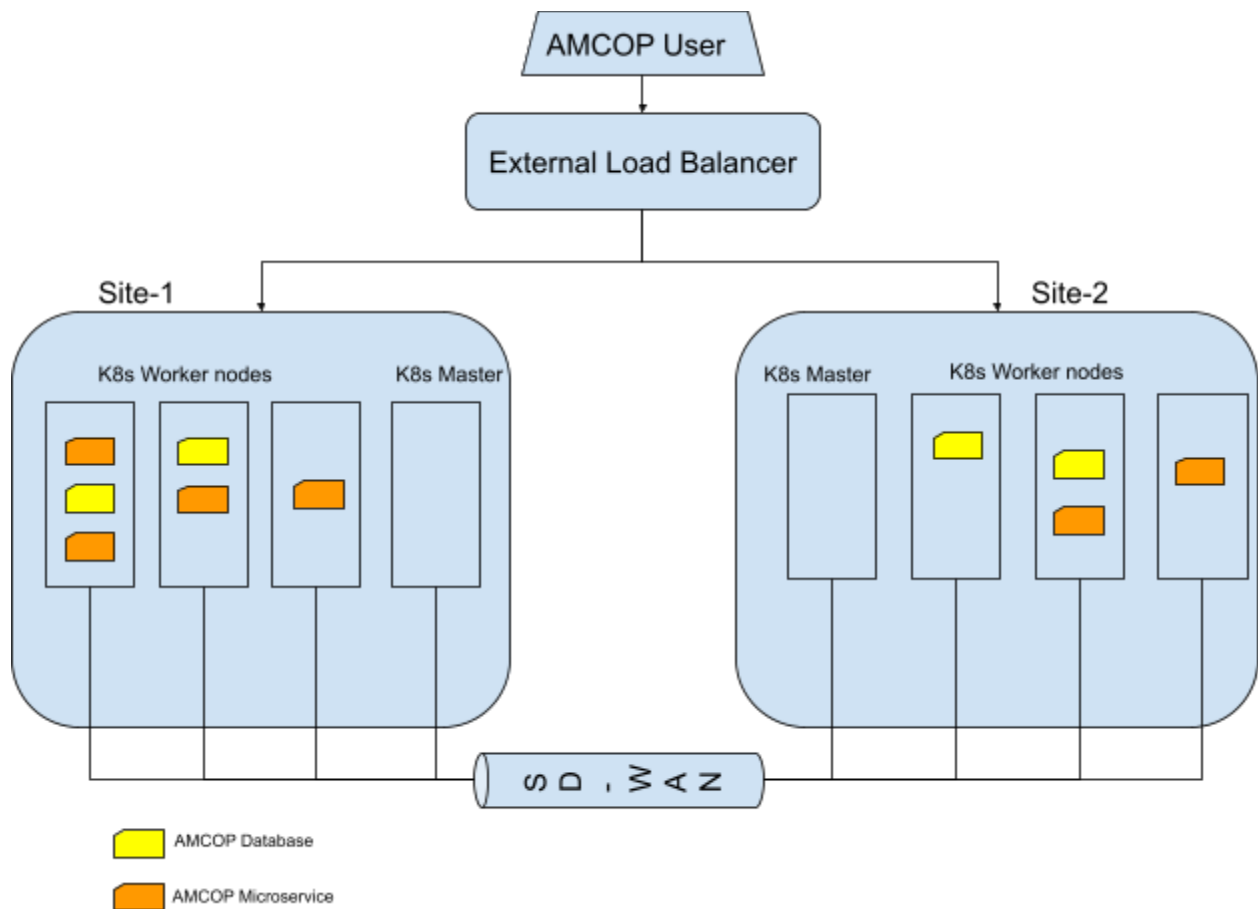
Following are the requirements of a production-grade deployment of AMCOP. These requirements may need to be customized for each deployment. Please contact the Aarna Support team for planning the production deployment.

- Security
 - Work with lowest possible privileges, less chance of affecting other applications
 - Security Certs and passwords need to be different in every deployment
- Data Backup
 - Backup stateful information, which can be reused to restore and recreate the same state of application on redeployment
 - Use single instance of Database where ever possible to avoid managing multiple database backups
- High Availability
 - Planned / unplanned unavailability of instances
 - Database should run as a HA cluster, ensuring seamless working with relevant states
- Performance
 - Fine tune the performance based on the available constructs in production environment
 - Using NFS backed storage should be avoided due to performance reasons
 - Avoid the usage of dockerdata-nfs
 - AMCOP Helm Charts provide the mechanism to override the storage class available
 - AMCOP expects - dynamic persistent volume provisioner
 - PV creation is triggered based on the PVC
 - Support for multiple types of storage class
 - ReadWriteOnce - SSD backed for DataBase and other non-shared persistent storage
 - ReadWriteMany - NFS/Ceph/Cinder backed PV for Sharing content between Pods
- Scalability
 - Scale beyond the regular development process

The HA configuration requirements are as follows:

- Handle Kubernetes node down
 - Microservices and Database needs to be spread across multiple kubernetes worker nodes with more than 1 replicas (minimum recommended 3)
 - Replicas themselves should not cluster up on the same node
- Handle Site down
 - Replicas should not cluster up in single site
- Pod Anti-affinity
 - Ensure that you run only one instance of a microservice on one node
- Node selector and node labels
 - Run Pods only on nodes with specific labels, ensuring control over where these pods can run

The sample deployment of AMCOP for meeting these requirements is as follows:



Configuring Desktop/Laptop to access AMCOP portal

Refer to “Configuring Desktop/Laptop to access AMCOP portal” section in AMCOP User Guide for details on how to access AMCOP portal.

AMCOP Lifecycle functions

Upgrade/Downgrade

Current version of AMCOP (3.1) supports automated upgrades from the 3.0.0 patch release (#07222022) release. This is only supported if you are deploying using AMCOP Operator. To upgrade to 3.1 (on your existing 3.0.0 patch release #07222022 installation), you can run the following commands. This will automatically upgrade all the components of AMCOP to 3.1 release, and restart the pods. All the existing data from your previous deployment will be retained.

```
kubectl apply -f  
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/operator.yaml  
|
```

```
kubectl apply -f  
https://aarna-networks.gitlab.io/amcop-deployment/amcop-k8s-operator/v3.1/default.yaml
```

Note:

Once the upgrade is done, downgrade/rollback to previous versions is not yet supported. This will be added in future versions.

Upgrades from versions prior to one older version (eg., 3.0.0 or below) to the current release (3.1) are also not supported currently. This will be added in future versions. For now, if the upgrade needs to be done from older versions to 2.4.1, it needs to be done as a new installation after uninstalling the previous version.

Please contact the Aarna Support team for any issues encountered during upgrade.

Cleanup

AMCOP deployment cleanup provides commands to clean up AMCOP deployment completely or individual components. The supported tags are vm, cluster and emco (AMCOP components).

- To undeploy AMCOP that is deployed with the default configuration (VM with name amcop-vm-01 and user name "ubuntu"), run the following command. This will delete the VM (including all AMCOP components).

```
# Run the Ansible command to undeploy AMCOP
ansible-playbook -v ./amcop_cleanup.yml -i inventory.ini -e
jump_host_user=<user_name_on_jump_host>
```

- To undeploy configuration components of AMCOP (which includes EMCO components and CDS components from ONAP) with a default configuration (VM with name amcop-vm-01 and user name "ubuntu"), run the following command. This will undeploy AMCOP components only.

```
# Run the Ansible command to undeploy AMCOP components only
```

```
ansible-playbook ./amcop_cleanup.yml -i inventory.ini -e
jump_host_user=<user_name_on_jump_host> --skip-tags vm,cluster
```

- To undeploy orchestration components of AMCOP (which includes EMCO components and CDS components from ONAP) alone with a default configuration (VM with name amcop-vm-01 and user name "ubuntu"), run the following command. This will undeploy AMCOP components and cluster only.

```
# Run the Ansible command to undeploy AMCOP components and cluster
```

```
ansible-playbook ./amcop_cleanup.yml -i inventory.ini -e
jump_host_user=<user_name_on_jump_host> --skip-tags vm
```

- To undeploy AMCOP that is deployed with a user defined VM name or a different user name or both, run the following command as per the deployment configuration. You need to change the --skip-tags parameters according to the cleanup activity that you want to perform.

```
# Run the Ansible command with user-defined VM name and different
# username to undeploy AMCOP components only.
```

```
ansible-playbook ./amcop_cleanup.yml -i inventory.ini -e server_name=<VM_NAME> -e
vm_user=<USER_NAME> -e jump_host_user=<user_name_on_jump_host> --skip-tags
vm,cluster,
```

```
#Run the Ansible command with a user defined VM name to undeploy AMCOP components
along with the cluster.
```

```
ansible-playbook ./amcop_cleanup.yml -i inventory.ini -e server_name=<VM_NAME> -e
jump_host_user=<user_name_on_jump_host> --skip-tags vm
```

AMCOP can be redeployed by running the Ansible scripts from the previous steps.