

R5 - Installation Documentation of Enterprise Applications on Lightweight 5G Telco Edge (EALTEdge)

- [Introduction](#)
- [How to use this document](#)
- [Deployment Architecture](#)
- [Pre-Installation Requirements](#)
 - [Hardware Requirements](#)
 - [Minimum Hardware Requirements](#)
 - [Recommended Hardware Requirements](#)
 - [Software Prerequisites](#)
 - [Database Prerequisites](#)
 - [Schema scripts](#)
 - [Other Installation Requirements](#)
 - [Jump Host Requirements](#)
 - [Network Requirements](#)
 - [Bare Metal Node Requirements](#)
 - [Execution Requirements \(Bare Metal Only\)](#)
- [Installation High-Level Overview](#)
 - [Bare Metal Deployment Guide](#)
 - [Install Bare Metal Jump Host](#)
 - [Creating a Node Inventory File](#)
 - [Creating the Settings Files](#)
 - [Running](#)
 - [Virtual Deployment Guide](#)
 - [Standard Deployment Overview](#)
 - [Jump Host Software Installations:](#)
 - [Jump Host Pre-Configurations for Center Components Installation](#)
 - [Installing Mode : EALTEdge using Ansible-Playbooks](#)
 - [Snapshot Deployment Overview](#)
 - [Special Requirements for Virtual Deployments](#)
 - [Install Jump Host](#)
 - [Verifying the Setup - VM's](#)
 - [Upstream Deployment Guide](#)
 - [Upstream Deployment Key Features](#)
 - [Special Requirements for Upstream Deployments](#)
 - [Scenarios and Deploy Settings for Upstream Deployments](#)
 - [Including Upstream Patches with Deployment](#)
 - [Running](#)
 - [Interacting with Containerized Overcloud](#)
- [Verifying the Setup](#)
 - [Verifying EALTEdge Deployment](#)
 - [Deploy Application in EALTEdge](#)
- [Developer Guide and Troubleshooting](#)
 - [Uninstall Guide](#)
 - [Using Ansible Playbooks](#)
 - [Vault documentation](#)
- [Troubleshooting](#)
- [Maintenance](#)
 - [Blueprint Package Maintenance](#)
 - [Software maintenance](#)
 - [Hardware maintenance](#)
 - [Blueprint Deployment Maintenance](#)
- [Frequently Asked Questions](#)
- [License](#)
- [References](#)
- [Definitions, acronyms and abbreviations](#)

Introduction

The guide covers the installation details which are related to Enterprise Applications on Lightweight 5G Telco Edge (EALTEdge) Blueprint.

This guide covers detailed information of the various types of deployments, detailed steps and what are the various components it will install. In addition, the guide provides information on hardware requirements, prerequisite software and minimum hardware requirements. On successful deployment, Center and Edge Nodes will be installed. The number of nodes in Center cluster and Edge node in the cluster is configurable.

The CENTER Node is a K8s Cluster and EDGE Node is a K8s Cluster.

How to use this document

The document includes details of prerequisites /pre-installation, installation and uninstalls steps.

The prerequisites and pre-installation software and hardware should be ready before executing the installation steps.

In BP first release Two types of installation mechanisms are provided, as below

1. Ansible-Playbook single command
2. Command Line Interface (CLI)

Deployment Architecture

The Deployment Architecture consists of the following nodes

- One-Click Deployment Node
- Center Node
- Edge Node

Note: For Development environment two nodes is sufficient, where one node plays a dual role of One-Click Deployment Node and MECM Node with other as MEC Host.

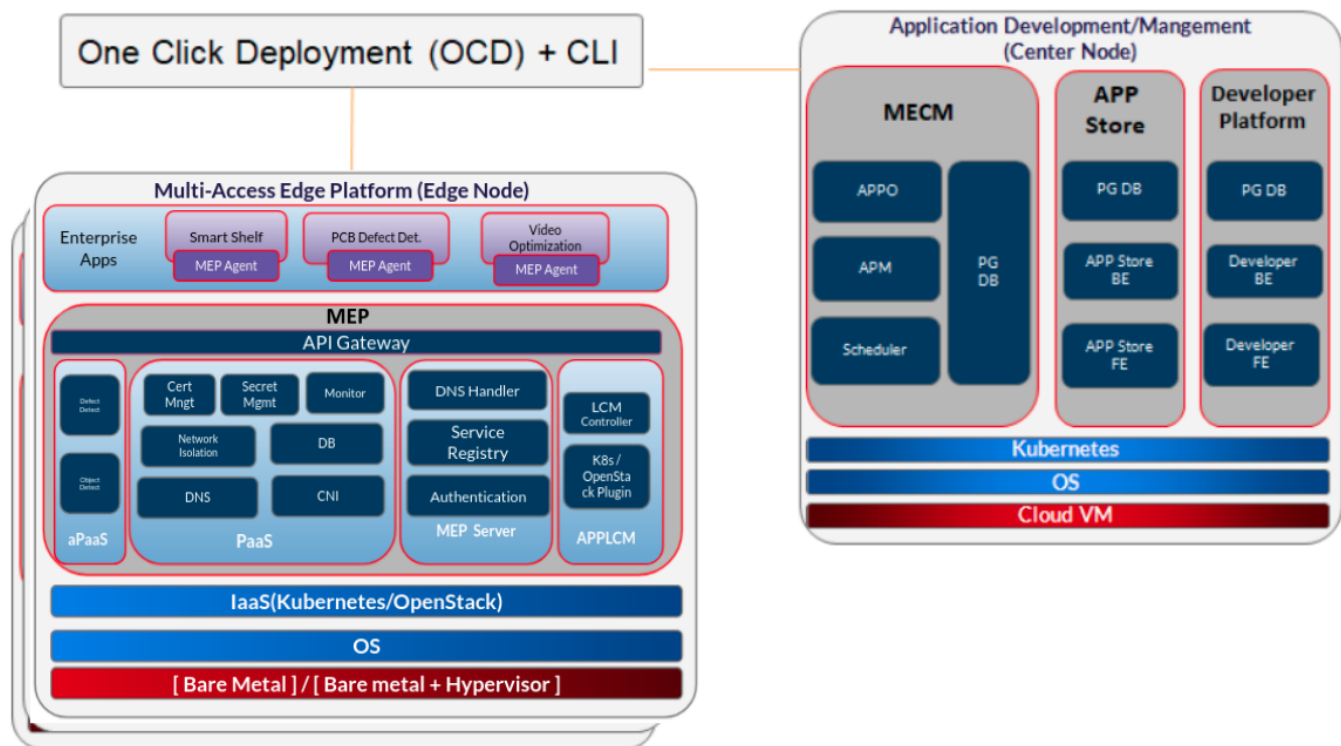


Figure: EALTEdge Deployment Architecture

Note: EALTEdge Blueprint Deployment has been tested on Cloud VM and is not tested on Bare-Metal Environment. Though, theoretically deployment should work in bare metal, provided hardware and software prerequisites are met. Kindly refer [R5 - Test Documentation of Enterprise Applications on Lightweight 5G Telco Edge \(EALTEdge\)](#) to get details on the tested deployment.

Pre-Installation Requirements

Hardware Requirements

The number of Hardware requirements depends mainly on the Use Case Scenario and the enterprise scale. A use case can have one MECM Cluster with one or multiple MEC Host clusters.

The minimum number of nodes required for a complete EALTEdge Topology is three. (Bare-Metal or Virtual Machines)

- 1) Deployment Node
- 2) Center Node

3) Edge Node

Note: The Hardware details provided are of Virtual Machine configurations.

Minimum Hardware Requirements

CENTER Node	
HW Aspect	Requirements
# of Node(s)	A virtual machine hosted in any Cloud Provider having internet connectivity.
# of CPU	8
Architecture	x86_AMD64 or ARM64.
RAM	8 GB
Disk	120 GB ~ 512GB
Networks	1

EDGE Node(s)	
HW Aspect	Requirements
# of Node(s)	1 MEC Host
# of CPU	4
Architecture	x86_AMD64 or ARM64.
RAM	4 GB
Disk	20 GB ~ 256 GB
Network	1

Note: The above specifications are given considering the EALTEdge CI / CD environment. User can try lower configuration considering lightweight components being used.

Recommended Hardware Requirements

CENTER Node	
HW Aspect	Requirements
# of Node(s)	A virtual machine hosted in any Cloud Provider having internet connectivity.
# of CPU	8
Architecture	x86_AMD64 or ARM64.
RAM	8 GB
Disk	120 GB ~ 512GB
Networks	1

EDGE Node(s)	
HW Aspect	Requirements
# of Node(s)	1 MEC Host
# of CPU	4
Architecture	x86_AMD64 or ARM64.
RAM	4 GB
Disk	20 GB ~ 256 GB

Network	1
---------	---

Software Prerequisites

- Virtual Machines preinstalled with Ubuntu 18.04 for MECM Node.
- Virtual Machines preinstalled with Ubuntu 18.04 for MEC Host Nodes
- root user created in the Deployment Node, MEC Node and MEC Host Node.
- SSH Server running in all the Nodes.
- Ansible > 2.10.7 installed in One Click Deployment Node (Jump Host)
- git installed in Jump Host.

Database Prerequisites

Schema scripts

N/A

Other Installation Requirements

Jump Host Requirements

Network Requirements

- Internet connectivity in OCD Host, CENTER and EDGE Nodes.
- The CENTER Node and EDGE Node should be able to ping each other.

Bare Metal Node Requirements

N/A

Execution Requirements (Bare Metal Only)

N/A

Installation High-Level Overview

The blueprint provides one click deployment and command-line interface for installing the EALTEdge blueprint components.

Bare Metal Deployment Guide

Install Bare Metal Jump Host

Note: EALTEdge Blueprint Deployment has been tested on Huawei Cloud Virtual Machines and is not tested on Bare-Metal Environment.

Though theoretically deployment should run successfully in bare metal too provided hardware and software prerequisites are met.

Creating a Node Inventory File

N/A

Creating the Settings Files

N/A

Running

N/A

Virtual Deployment Guide

For Virtual Deployment minimum 2 Virtual machines(OCD and Center node can be deploy on same VM or in different VMs), following are the virtual machines and their usage

No	Usage
1	One Click Deployment Node
2	CENTER Node
3	EDGE Node

All the nodes should have internet connectivity , network interface and network connectivity between the VM's.

Standard Deployment Overview

Jump Host Software Installations:

Login to the Jump Host and perform the below steps:

1. Install Ansible > 2.10.7 [https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html]
2. Install git
3. Install python3 and pip3

Jump Host Pre-Configurations for Center Components Installation

Login to the Jump Host and perform the below configuration steps (Steps : as below-

1. Generate public key : #ssh-keygen -t rsa
2. Setup password-less login from ocd to center and ocd to edge.

```
sshpass -p <password> ssh-copy-id -p <ssh-port> -o StrictHostKeyChecking=no root@<node_ip>
```

3. Review and Change Parameters

For EdgeGallery MUNO Mode

```
ealt-edge/ocd/infra/playbooks/muno-config/controller/hosts-muno-controller
```

```
ealt-edge/ocd/infra/playbooks/muno-config/controller/var.yml
```

```
ealt-edge/ocd/infra/playbooks/muno-config/edge/hosts-muno-edge
```

```
ealt-edge/ocd/infra/playbooks/muno-config/edge/var.yml
```

For EdgeGallery AIO Mode

```
ealt-edge/ocd/infra/playbooks/hosts-aio
```

```
ealt-edge/ocd/infra/playbooks/var.yml
```

For EALT-EDGE stack:

```
ealt-edge/ocd/infra/playbooks/ealt-inventory.ini
```

Installing Mode : EALTEdge using Ansible-Playbooks

1. git clone the ealt-edge repo, to download the software to install the EALTEdge Environment.

```
root@akraino-mec-0001:~# git clone "https://gerrit.akraino.org/r/ealt-edge"
```

2. go to the below directory

```
root@akraino-mec-0001:~# cd ealt-edge/ocd/infra/playbooks
```

3. Modify the Configuration File :

ealt-inventory.ini with the details of CENTER and EDGE Nodes.

For Edge Gallery installation:

MUNO-Mode:

Execute the below command:

```
cd ealt-edge/ocd/infra/playbooks
```

```
ansible-playbook -i muno-config/controller/hosts-muno-controller ealt-eg-muno-controller.yml --extra-vars "operation=install" -e "ansible_user=root"
```

```
ansible-playbook -i muno-config/edge/hosts-muno-edge ealt-eg-muno-edge.yml --extra-vars "operation=install" -e "ansible_user=root"
```

For AIO mode:

Execute the below command

```
cd ealt-edge/ocd/infra/playbooks
```

```
ansible-playbook ealt-eg-aio-latest.yml -i hosts-aio --extra-vars "operation=install" -e ansible_user=root"
```

Installation of ealt-edge stack:

```
ansible-playbook ealt-all.yml -i ealt-inventory.ini --extra-vars "operation=install"
```

Once the execution is completed in console will see prompt "EALTEdge Environment Installed , Components Install CENTER and EDGE Nodes Successfully"

Snapshot Deployment Overview

N/A

Special Requirements for Virtual Deployments

N/A

Install Jump Host

N/A

Verifying the Setup - VM's

N/A

Upstream Deployment Guide

Upstream Deployment Key Features

N/A

Special Requirements for Upstream Deployments

N/A

Scenarios and Deploy Settings for Upstream Deployments

N/A

Including Upstream Patches with Deployment

N/A

Running

N/A

Interacting with Containerized Overcloud

N/A

Verifying the Setup

Verifying EALTEdge Deployment

Currently the verification is manually done.

- 1. Login to the Center Node and check whether K8S cluster is installed.

Components and Services running in CENTER Node

```
root@httpl-virtual-machine-1:~# kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	appstore-be-0	1/1	Running	0	3d17h
default	appstore-be-postgres-0	1/1	Running	0	3d17h
default	appstore-fe-5fc67c4cd7-mw15w	1/1	Running	0	3d17h
default	atp-0	1/1	Running	0	3d17h
default	atp-fe-79d79b9cc4-8z74w	1/1	Running	0	3d17h
default	atp-postgres-0	1/1	Running	0	3d17h
default	developer-be-0	1/1	Running	0	3d17h
default	developer-be-postgres-0	1/1	Running	0	3d17h
default	developer-fe-77695d7b45-t4w4c	1/1	Running	0	3d17h
default	mecm-apm-5d44586555-52xr2	1/1	Running	0	3d17h
default	mecm-appo-995574b48-q88q9	1/1	Running	0	3d17h
default	mecm-fe-769968f86f-7kh2c	1/1	Running	0	3d17h
default	mecm-inventory-78cd95857b-gh5w8	1/1	Running	0	3d17h
default	mecm-postgres-0	1/1	Running	0	3d17h
default	service-center-f45dc6c5b-mh7hp	1/1	Running	0	3d17h
default	tool-chain-0	2/2	Running	0	3d17h
default	user-mgmt-89cfc9d98-jw7nb	1/1	Running	0	3d17h
default	user-mgmt-postgres-0	1/1	Running	0	3d17h
default	user-mgmt-redis-0	1/1	Running	0	3d17h
kube-system	calico-kube-controllers-578894d4cd-nlj4m	1/1	Running	0	3d17h
kube-system	calico-node-72qst	1/1	Running	0	3d17h
kube-system	coredns-66bff467f8-4hs7d	1/1	Running	0	3d17h
kube-system	coredns-66bff467f8-ppmjp	1/1	Running	0	3d17h
kube-system	etcd-httpl-virtual-machine-1	1/1	Running	0	3d17h
kube-system	kube-apiserver-httpl-virtual-machine-1	1/1	Running	0	3d17h
kube-system	kube-controller-manager-httpl-virtual-machine-1	1/1	Running	0	3d17h
kube-system	kube-proxy-w94wb	1/1	Running	0	3d17h
kube-system	kube-scheduler-httpl-virtual-machine-1	1/1	Running	0	3d17h
kube-system	metrics-server-686c5b74fs-ffbtb	1/1	Running	0	3d17h

```
root@httpl-virtual-machine-1:~# kubectl get svc -A
```

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	appstore-be-postgres-svc	ClusterIP	10.105.218.228	<none>	5432/TCP	3d17h
default	appstore-be-svc	NodePort	10.106.65.239	<none>	8099:30099/TCP	3d17h
default	appstore-fe-svc	NodePort	10.107.69.196	<none>	8443:30091/TCP	3d17h
default	atp-fe-svc	NodePort	10.106.253.200	<none>	8443:30094/TCP	3d17h
default	atp-postgres-svc	ClusterIP	10.108.9.225	<none>	5432/TCP	3d17h
default	atp-svc	ClusterIP	10.102.189.63	<none>	8073/TCP	3d17h
default	developer-be-postgres-svc	ClusterIP	10.96.80.173	<none>	5432/TCP	3d17h
default	developer-be-svc	NodePort	10.101.72.131	<none>	9002:30098/TCP	3d17h
default	developer-fe-svc	NodePort	10.99.235.163	<none>	8443:30092/TCP	3d17h
default	kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3d17h
default	mecm-apm	NodePort	10.106.41.128	<none>	8092:30202/TCP	3d17h
default	mecm-appo	NodePort	10.109.24.190	<none>	8091:30201/TCP	3d17h
default	mecm-fe-svc	NodePort	10.103.150.157	<none>	8443:30093/TCP	3d17h
default	mecm-inventory	NodePort	10.96.157.19	<none>	8093:30203/TCP	3d17h
default	mecm-postgres	ClusterIP	10.103.89.131	<none>	5432/TCP	3d17h
default	service-center	ClusterIP	10.105.154.142	<none>	30100/TCP	3d17h
default	tool-chain-svc	ClusterIP	10.97.20.52	<none>	8059/TCP	3d17h
default	user-mgmt-postgres-svc	ClusterIP	10.105.179.251	<none>	5432/TCP	3d17h
default	user-mgmt-redis-svc	ClusterIP	10.111.253.151	<none>	6379/TCP	3d17h
default	user-mgmt-svc	NodePort	10.102.74.34	<none>	8067:30067/TCP	3d17h
kube-system	kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,9153/TCP	3d17h
kube-system	metrics-server	ClusterIP	10.99.188.250	<none>	443/TCP	3d17h

Components and Services running EDGE Node

```

root@httpl-v2:~# kubectl get pods -A
NAMESPACE      NAME                                     READY   STATUS    RESTARTS   AGE
default        cadvisor                               1/1     Running   0           14m
default        influxdb-0                             0/1     Pending   0           14m
default        mecm-mepm-apprulemgr-fc6cf4c76-kwd7g   1/1     Running   0           12d
default        mecm-mepm-k8splugin-587877c5f8-6tlrv   1/1     Running   0           12d
default        mecm-mepm-lcmcontroller-66c5b9955c-q6n2m 1/1     Running   0           12d
default        mecm-mepm-osplugin-76b9b95fc4-zhpzj    1/1     Running   0           12d
default        mep-fe-6bf9b6fbdd-sftjh               1/1     Running   0           12d
default        mepm-fe-9f9bb5d8-cw6wd                1/1     Running   0           12d
default        mepm-postgres-0                       1/1     Running   0           12d
default        rabbitmq-0                             1/1     Running   0           9d
default        rabbitmq-1                             1/1     Running   0           8d
default        rabbitmq-2                             1/1     Running   0           8d
kube-system    calico-kube-controllers-578894d4cd-fgs54 1/1     Running   0           12d
kube-system    calico-node-rv2hs                      1/1     Running   0           12d
kube-system    coredns-66bff467f8-94zzh              1/1     Running   0           12d
kube-system    coredns-66bff467f8-zqcm              1/1     Running   0           12d
kube-system    edgegallery-secondary-ep-controller     1/1     Running   0           12d
kube-system    etcd-httpl-vm-2                        1/1     Running   0           12d
kube-system    kube-apiserver-httpl-vm-2              1/1     Running   0           12d
kube-system    kube-controller-manager-httpl-vm-2     1/1     Running   0           12d
kube-system    kube-multus-ds-amd64-5dfz2            1/1     Running   0           12d
kube-system    kube-proxy-k5cqw                       1/1     Running   0           12d
kube-system    kube-scheduler-httpl-vm-2              1/1     Running   0           12d
kube-system    metrics-server-686c5b74f5-w8q2w       1/1     Running   0           12d
mep            mep-6b9fd55bdc-66xdw                  4/4     Running   1           12d
mep            mep-elasticsearch-8594f69968-6jm4b     1/1     Running   0           12d
mep            mep-pg-84d795854b-864xm               1/1     Running   0           12d
metallb-system controller-b0f656d8c-8khlg              1/1     Running   0           12d
metallb-system speaker-xj6z9                      1/1     Running   0           12d
openebs        openebs-admission-server-78595f744-wvg6b 1/1     Running   0           14m
openebs        openebs-apiserver-649d9b59b4-4fs8q     1/1     Running   0           14m
openebs        openebs-localpv-provisioner-5cc54b5dd4-8ffgp 1/1     Running   0           14m
openebs        openebs-ndm-kkc7g                      1/1     Running   0           14m
openebs        openebs-ndm-operator-6fddc68cff-4f7ks   1/1     Running   0           14m
openebs        openebs-provisioner-7f565c6f7d-26b2l    1/1     Running   0           14m
openebs        openebs-snapshot-operator-769b994d88-sldp8 2/2     Running   0           14m

```

```

root@httpl-v2:~# kubectl get svc
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)                                     AGE
commandnodeport                    NodePort    10.99.209.51     <none>            48082:32452/TCP                           8d
consulnodeport                     NodePort    10.105.196.221   <none>            8400:31177/TCP, 8500:32697/TCP, 8600:31405/TCP 8d
datanodeport                       NodePort    10.104.126.102   <none>            48080:32076/TCP, 5563:38522/TCP           8d
edgex-core-command                 ClusterIP   10.103.57.254    <none>            48082/TCP                                  8d
edgex-core-consul                  ClusterIP   10.110.164.177   <none>            8400/TCP, 8500/TCP, 8600/TCP              8d
edgex-core-data                    ClusterIP   10.109.203.32    <none>            48080/TCP, 5563/TCP                      8d
edgex-core-metadata                ClusterIP   10.104.180.239   <none>            48081/TCP                                  8d
edgex-device-virtual                ClusterIP   10.106.238.45    <none>            49990/TCP                                  8d
edgex-export-client                ClusterIP   10.99.23.221     <none>            48071/TCP                                  8d
edgex-export-distro                ClusterIP   10.109.70.132    <none>            48070/TCP, 5566/TCP                      8d
edgex-mono                          ClusterIP   10.107.245.136   <none>            27817/TCP                                  8d
edgex-support-logging              ClusterIP   10.96.68.148     <none>            48061/TCP                                  8d
edgex-support-notifications        ClusterIP   10.101.242.6     <none>            48060/TCP                                  8d
edgex-support-rulesengine          ClusterIP   10.104.68.172    <none>            48075/TCP                                  8d
edgex-support-scheduler            ClusterIP   10.107.93.250    <none>            48085/TCP                                  8d
influxdb                           ClusterIP   10.100.181.153   <none>            8086/TCP, 8088/TCP                       15m
kubernetes                         ClusterIP   10.96.0.1         <none>            443/TCP                                    12d
loggingnodeport                    NodePort    10.100.126.253   <none>            48061:31235/TCP                           8d
mecm-mepm-apprulemgr               NodePort    10.111.159.200   <none>            8096:30206/TCP                            12d
mecm-mepm-k8splugin                NodePort    10.110.50.109    <none>            8095:30205/TCP                            12d
mecm-mepm-lcmcontroller            NodePort    10.104.223.46    <none>            8094:30204/TCP                            12d
mecm-mepm-osplugin                 NodePort    10.96.105.247    <none>            8234:30207/TCP                            12d
mep-fe                             NodePort    10.100.90.244    <none>            8100:30095/TCP                            12d
mepm-fe                             NodePort    10.107.235.168   <none>            8200:30097/TCP                            12d
mepm-postgres                     ClusterIP   10.97.193.130    <none>            5432/TCP                                    12d
metadatanodeport                  NodePort    10.97.216.102    <none>            48081:32114/TCP                           8d
rabbitmq                           NodePort    10.101.100.94    <none>            15672:31672/TCP, 5672:38672/TCP           8d
rulesenginenodeport                NodePort    10.97.51.220     <none>            48075:38521/TCP                           8d

```

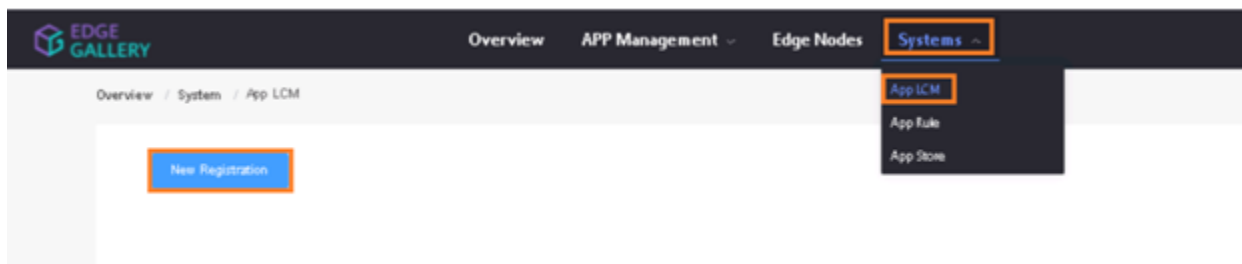
Deploy Application in EALTEdge

1. Login to MECM Portal <https://ip:30093>
 - 1.1 click on **Systems ->App LCM ->New Registration**

Name: Applcm(any general name)

IP: applcm"public ip"

Port: 30204



App LCM Registration

*

Name

*

Ip

*

Port

30204

Cancel

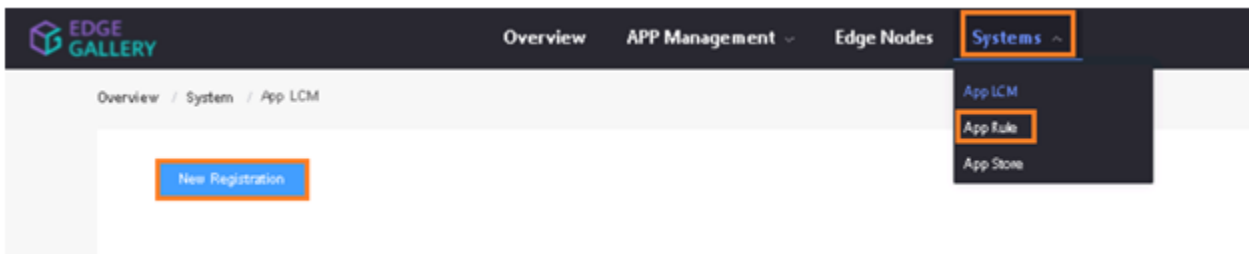
Confirm

1.2. click on **Systems ->App Rule -> New Registration**

Name: Apprule(any general name)

IP: applcm"public ip"

Port: 30206



App Rule MGR Registration

*

Name

*

Ip

*

Port

30206

Cancel

Confirm

1.3. click on **Systems ->App Store ->New Registration**

App Store Name: appstore(any general name)

IP: Appstore public ip

Port: 30099

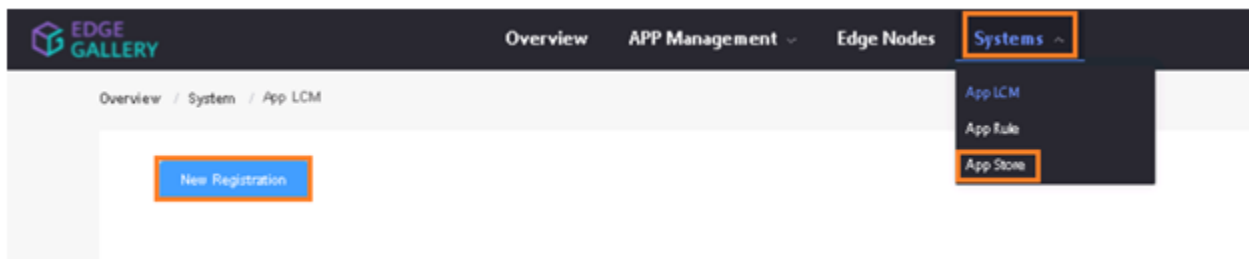
Appstore Repo: {HarborIP:443}(192.168.1.1:443)

Repo Name: appstore(any general name)

Repo Username: admin(harbor user name)

Repo Password: Harbor@edge(harbor password)

Vendor: vendor(any general name)



App Store Registration

* App Store Name

* IP

* Port

* Appstore Repo

* Repo Name

* Repo Username

* Repo Password

* Vendor

Cancel

Confirm

2. log in to MECM Portal <https://ip:30093>

2.1. **Add k8s node:**

Click on **Edge Nodes ->New Rgistration**

System: k8s

Name: edge1(any general name)

IP: edge public IP

Location: Select from the drop-down

Architecture: x86

Capabilities: select none

App LCM: Select edge IP from the drop-down box

App Rule MGR: Select edge IP from the drop-down box

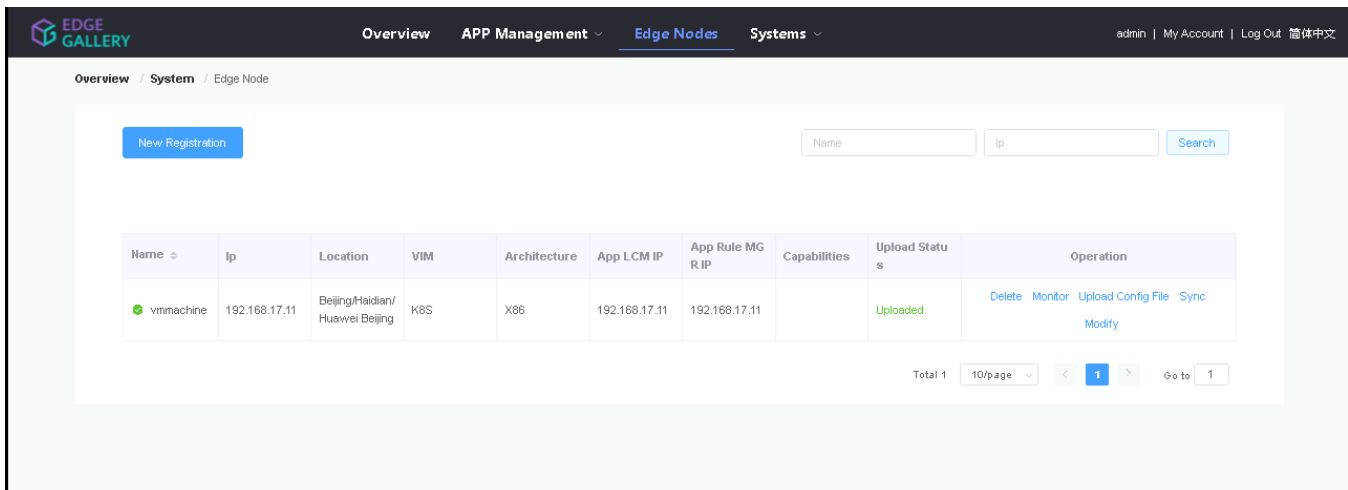


The dialog box titled "Edge Node Registration" contains the following fields and options:

- System: ☒ K8S, ☐ OpenStack
- Name:
- Ip:
- Location:
- Architecture: ☒ X86, ☐ ARM64, ☐ ARM32
- Capabilities: ☐ GPU, ☐ NPU
- App LCM:
- App Rule MGR:
- Buttons: Cancel, Confirm

2.2. Download /root/.kube/config file from edge node

And click on **Upload config file** to upload.



The screenshot shows the "Edge Nodes" page in the Edge Gallery interface. The top navigation bar includes "Overview", "APP Management", "Edge Nodes", and "Systems". The user is logged in as "admin".

Below the navigation bar, there is a "New Registration" button and a search bar with "Name" and "Ip" fields and a "Search" button.

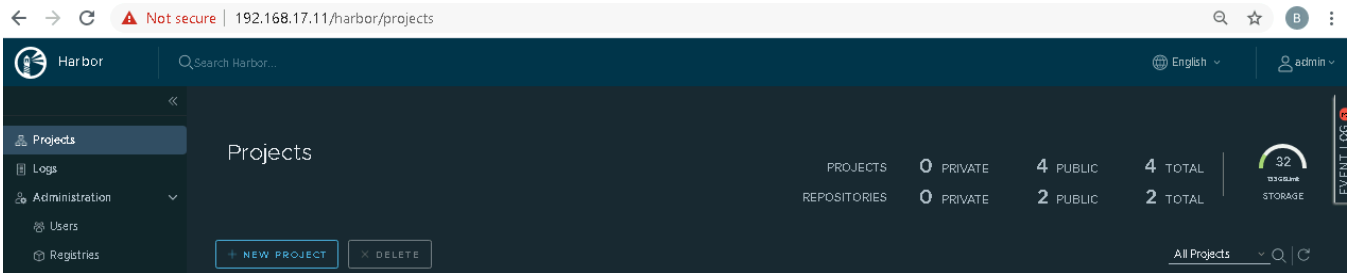
The main content area displays a table of registered edge nodes:

Name	Ip	Location	VIM	Architecture	App LCM IP	App Rule MGR IP	Capabilities	Upload Status	Operation
vmmachine	192.168.17.11	Beijing/Haidian/Huawei Beijing	K8S	X86	192.168.17.11	192.168.17.11		Uploaded	Delete Monitor Upload Config File Sync Modify

At the bottom right, there is a pagination bar showing "Total 1", "10/page", and "Go to 1".

3. log in to harbor Portal <https://ip:443>

3.1. Add three new projects



3.2. Those three **projects' names** are appstore, developer, and mecm. And select **access level** to the public.

3.3. Final page will look like the below screenshot.

☐	Project Name	Access Level	Role	Repositories Count	Creation Time
☐	appstore	Public	Project Admin	1	8/17/21, 1:36 AM
☐	developer	Public	Project Admin	0	8/17/21, 1:35 AM
☐	library	Public	Project Admin	0	8/17/21, 1:11 AM
☐	mecm	Public	Project Admin	1	8/17/21, 1:36 AM

1 - 4 of 4 items

4. log in to Developer Portal <https://ip:30092>

4.1. Add sandbox env to deploy application before publishing

Click System ->Host Management ->Add Host

Name: general name

System: k8s

Lcmip: sandbox ip(for testing purpose can provide edge ip, if no sandbox env)

mecmHost: sandbox ip(for testing purpose can provide edge ip, if no sandbox env)

Port: 30204

Protocol: https

Architecture: X86

Status: Normal

Port Range: leave as it is

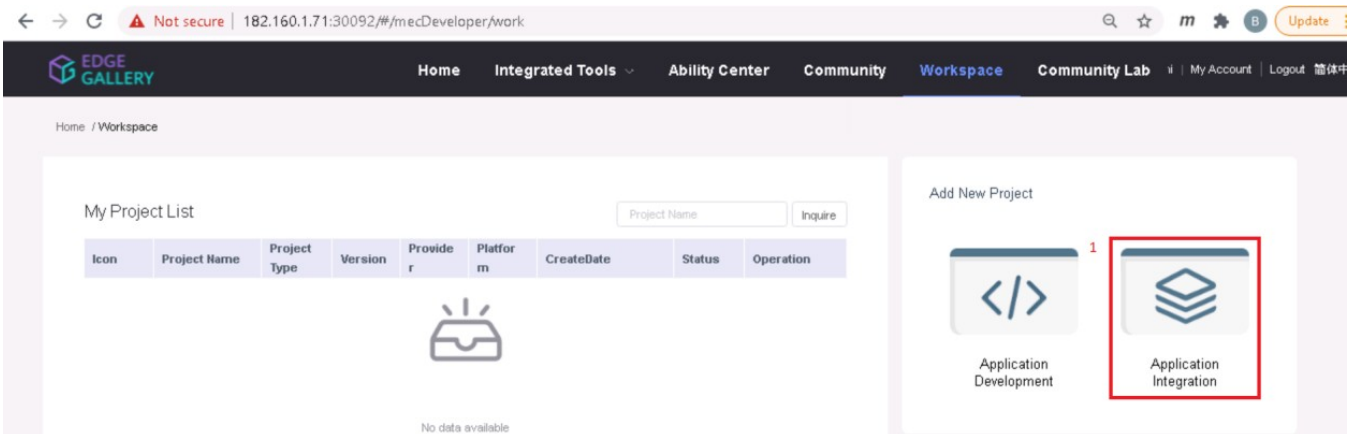
Address: Bangalore

UploadConfig File: upload sandboxenvkubefconfig file

4.2 Click on **Workspace** -> **Create Project** -> **Application Integration** -> **Start**



- Go to **Application Integration**



- Provide **App Name**, **Version**, **Provider**, **Workload Type**, **Architecture**, **Industry**, **Type**.

- Upload **Icon**, provide **Description**. And click on **confirm**.

Add application integration project

* App Name

cherry

* Version

v1.0

* Provider

Huawei

* Workload Type

Container

VM

* Architecture

x86

* Industry

Smart Park

* Type

Video Application

* Icon

* Description

cherry description

18/1024

Confirm

4.3. Now click on **Deployment Test**.

- Upload Docker images directly from the portal by clicking on **Upload App Image**.

1. `docker save -o <path-to-save>/<repo-name>.tar <repo-name>:<tag>`

After that you can take tar file from path and upload image.

Or, directly push Docker images to Harbor repo (takes lesser time, **preferred**). Following command for pushing an image in Harbor:

1. `cat /etc/docker/daemon.json` (execute this command in Edge gallery installed machine, to get an IP)
2. `docker tag <repo-name>:<tag> <IP>/developer/<repo-name>:<tag>`
3. `docker push <IP>/developer/<repo-name>:<tag>`

EDGE GALLERY

Home Integrated Tools Ability Center Community Workspace Community Lab My Account Logout

Project Details

Deployment Test

Application Release

1

Upload App Image

2

Configure Deployment Files

3

Deployment Test

Mode 1:

Upload app image to the public repository (supports dockerhub and SWR)

Mode 2:

Upload app image to the EdgeGallery repository

2

Upload App Image

Mode 3:

Upload app image to the private Edge Node

How to build a private Edge Node, please refer to [Installation Doc](#)

Import Host IP:

IP

Port

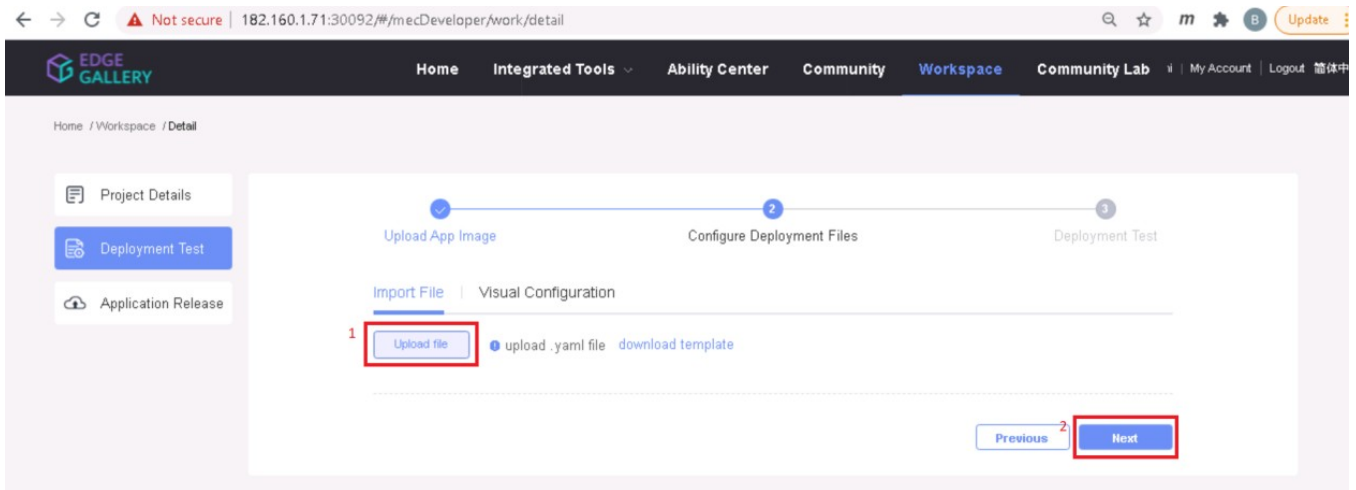
Test

Enable:

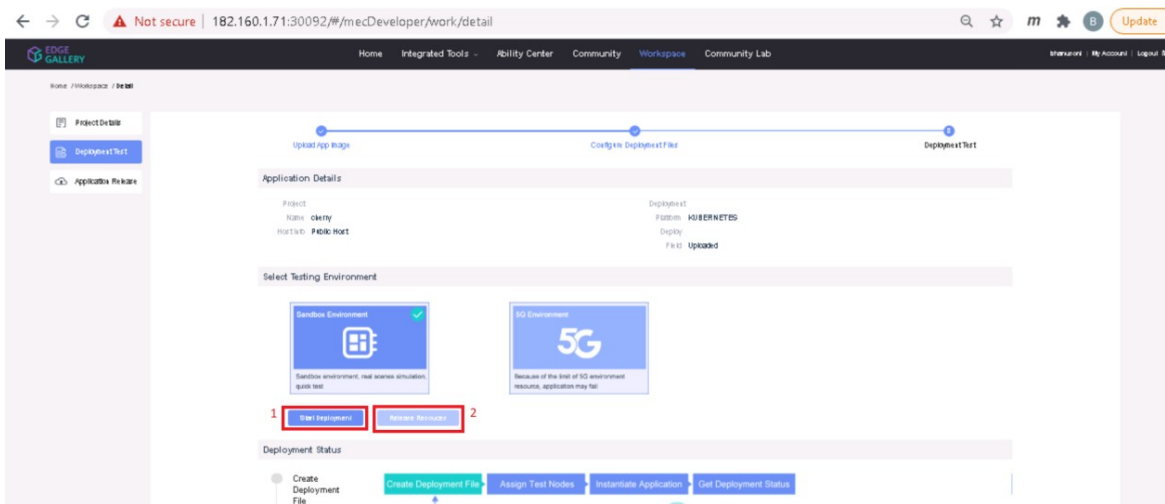
3

Next

- Click **next**, upload deployment yaml file now.



- After config upload, click **next** and click **start deployment**
- After Deployment is successful, click on **Release Resources**

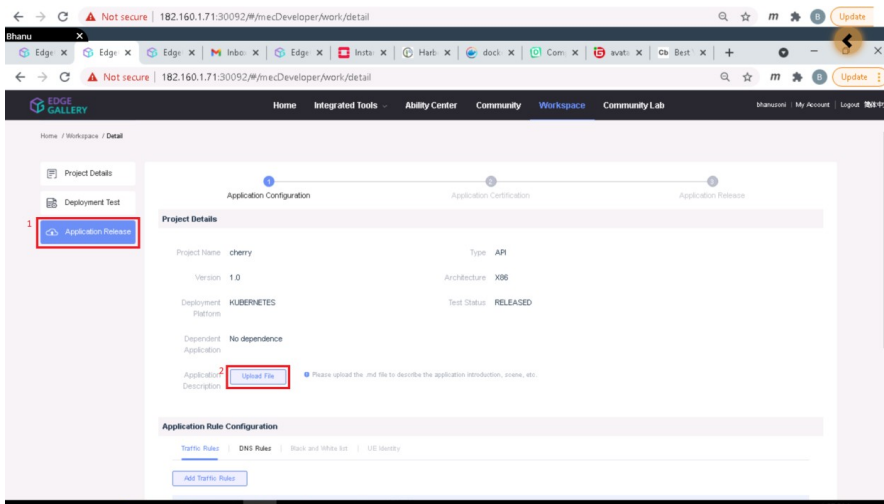


Note:

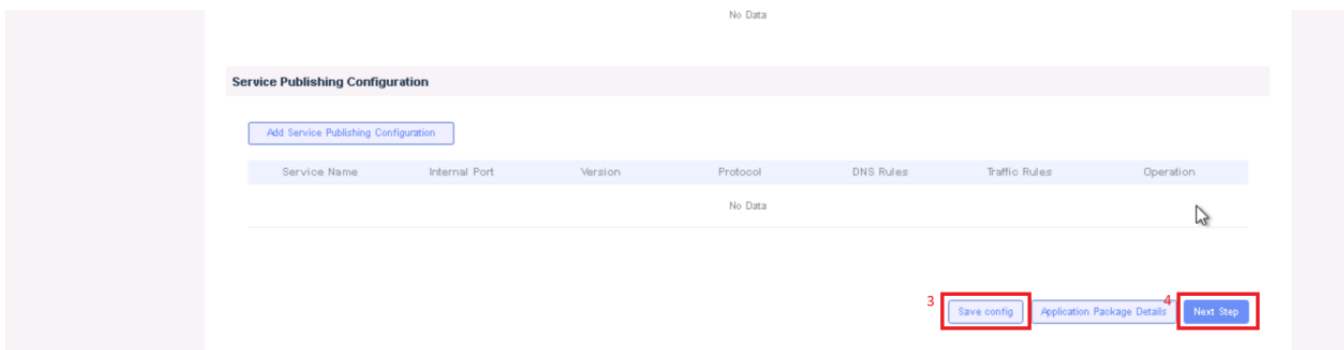
- While Deployment test if any error happens, open ATP portal (<https://ip:30094>) in another tab of the browser, sign in, come back to the developer portal and re run deployment test
- gitee.com/edgegallery/applications repo provides A lot of applications with their logo, deployment YAML & user guides

3.4. Now click on Application Release

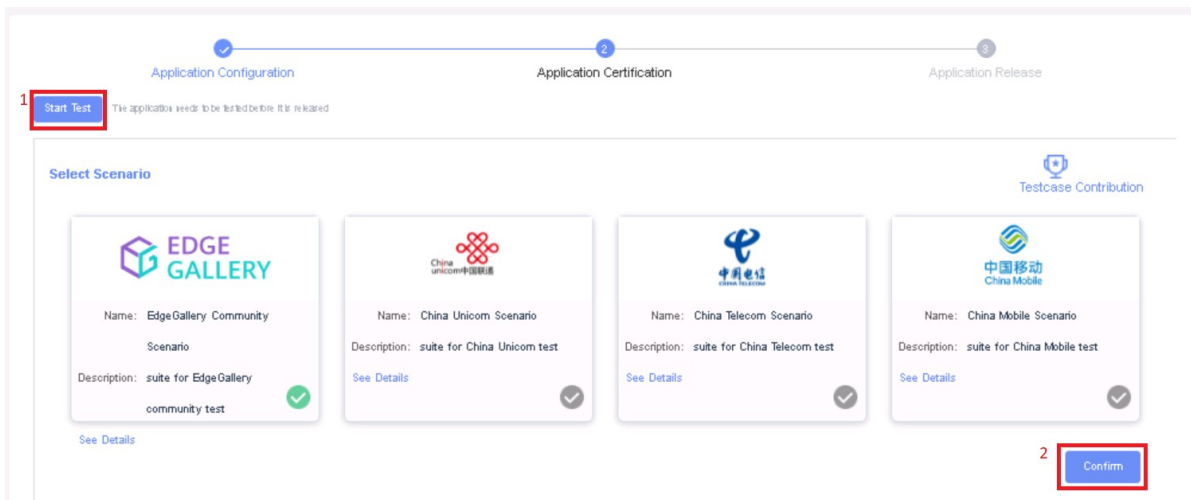
Upload file for **Application Description**

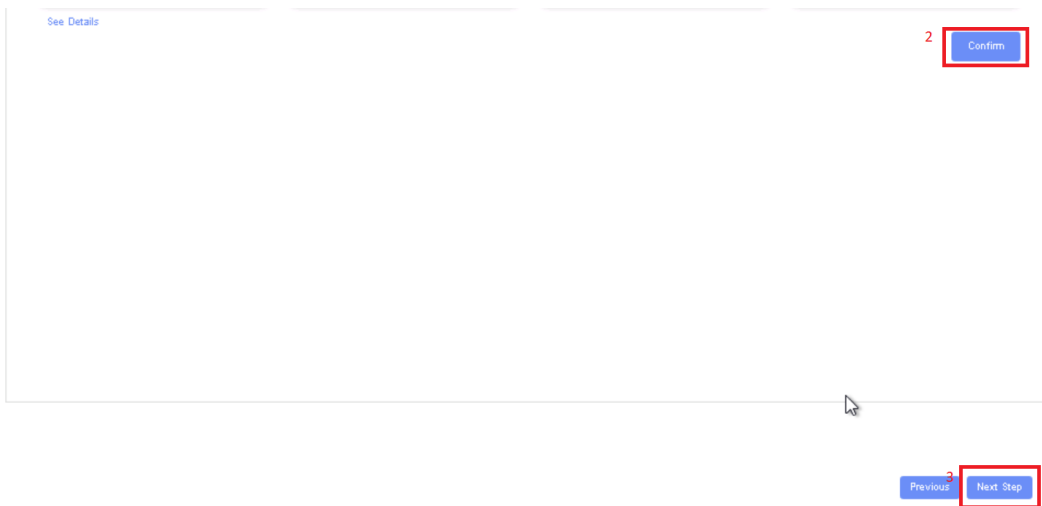


Click **save config**

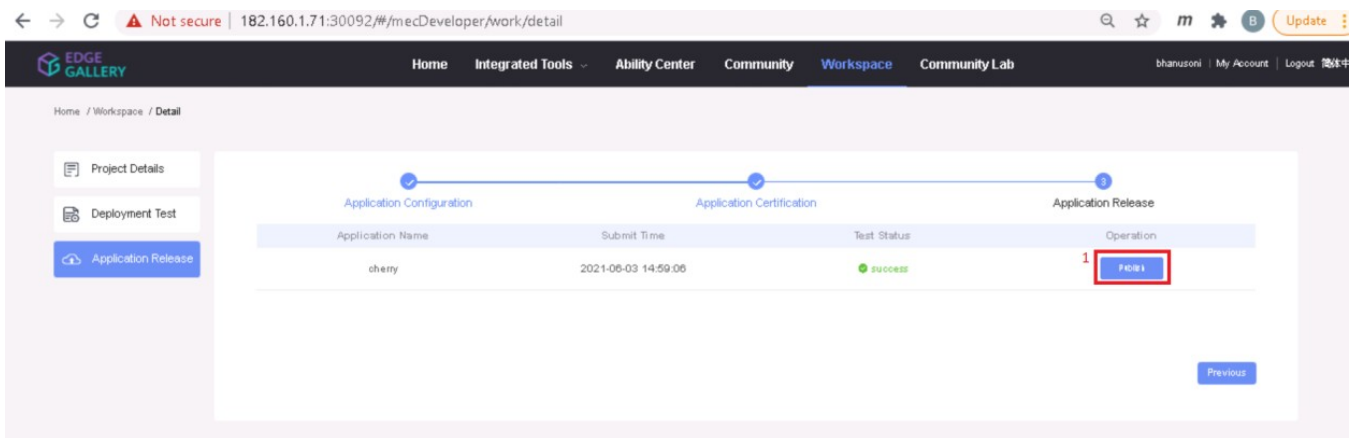


Click **Next Step**, click **Start Test**, scroll down to find & click the **Start Test** button, then confirm. Once the testing is complete click on **Next Step**





click **publish** to publish an application to AppStore. Go to <https://<IP>:30091> and **App Warehouse** to confirm that the application is successfully ported.



Developer Guide and Troubleshooting

Uninstall Guide

Using Ansible Playbooks

For EALT-EDGE stack

```
root@akraino-mec-0001:~#ansible-playbook ealt-all-uninstall.yml -i ealt-inventory.ini --extra-vars "operation=uninstall"
```

For MUNO Mode

```
root@akraino-mec-0001:~#ansible-playbook -i muno-config/controller/hosts-muno-controller ealt-eg-muno-controller.yml --extra-vars "operation=uninstall" -e "ansible_user=root"
```

```
root@akraino-mec-0001:~#ansible-playbook -i muno-config/edge/hosts-muno-edge ealt-eg-muno-edge.yml --extra-vars "operation=uninstall" -e "ansible_user=root"
```

For AIO Mode

```
root@akraino-mec-0001:~#ansible-playbook -i hosts-aio ealt-eg-aio-latest.yml --extra-vars "operation=uninstall" -e "ansible_user=root"
```

Vault documentation

```
**This document explains how to generate certificate by using vault and cert manager**
##Cluster Architecture

##Make a cluster
##The Image try to put with reference to our environment, with reference to EALT Edge. Can make a picture where
Vault will be running in MEC Host (as Root CA) , ##Cert Manager and Applications (App1, App2)
##1. Add helm repo
```
helm repo add hashicorp https://helm.releases.hashicorp.com
helm install vault hashicorp/vault
```
##2. Generate root token and Unseal Key
```
kubectl exec vault-0 -- vault operator init -key-shares=1 -key-threshold=1 -format=""
```
##Note: Root token we will use when we will login vault pod, Unseal Key and Root token will looks like below ex-
##Unseal Key 1: QcTX47IacKidIjFWSrkGLiQG1fwaqoInEz0SqAZ7rMs=
##Initial Root Token: s.A0SXgscZxbCeJRd1AjsVzvUU

##Generated Unseal key need to put in below command then vault will start running as a pod
```
kubectl exec -ti vault-0 -- vault operator unseal <Unseal Key>
```
##Vault is initialised as a pod
##By using below command can login in vault pod
```
kubectl exec -it vault-0 -- /bin/sh
```
##Vault Initialisation and Configuration Steps
####Once we initialize the vault pod we get unseal key and root token, need to put the root token
```
vault login <root token>
```
##Enable the PKI secrets engine
##By default, the secrets engine will mount at the name of the engine. To enable the secrets engine at a ##different path, use the -path argument.
```
vault secrets enable pki
```
##Keep the value in sync with the comment. 30 days, Increase the TTL by tuning the secrets engine. The default value of 30 days may be too short
```
vault secrets tune -default-lease-ttl=2160h -max-lease-ttl=87600h pki
```
##Configure a CA certificate and private key. It can generate ##its own self-signed root
## ealtdge.com is a your common_name or base url
```
vault write pki/root/generate/internal common_name=ealtdge.com ttl=8760h
```
##Update the CRL location and issuing certificates. These values can be updated in the future.
```
vault write pki/config/urls issuing_certificates="http://127.0.0.1:8200/v1/pki/ca" crl_distribution_points="http://127.0.0.1:8200/v1/pki/crl"
```
##It will allow your domain and subdomain
```
vault write pki/roles/my-role allowed_domains=ealtdge.com allow_subdomains=true max_ttl=8760h
```
##Generate a new credential by writing to the /issue endpoint with the name of the role
##The output will include a dynamically generated private key and certificate which corresponds to the ##given role
##The issuing CA and trust chain is also returned for automation simplicity
```
vault write pki/issue/my-role common_name=www.ealtdge.com
```
####Enabling AppRole in Vault
```
vault auth enable approle
```
```

```

##Writing vault policy
...

vault policy write pki-policy -<<EOF
path "pki" { capabilities = ["create", "read", "update", "delete", "list", "sudo"]}
EOF
...

##Write Auth role
...

vault write auth/approle/role/my-role secret_id_ttl=8760h token_num_uses=0 token_ttl=2160h token_max_ttl=8760h secret_id_num_uses=0 policies=pki-policy
...

##Note:-
##my-role - is the role name
##secret_id_ttl - (Optional) The number of seconds after which any SecretID expires
##token_num_uses - (Optional) The period, if any, in number of seconds to set on the token
##token_ttl - (Optional) The incremental lifetime for generated tokens in number of seconds. Its current value will be referenced at renewal time
##token_max_ttl - (Optional) The maximum lifetime for generated tokens in number of seconds. Its current value will be referenced at renewal time
##secret_id_num_uses - (Optional) The number of times any particular SecretID can be used to fetch a token from this AppRole, after which the SecretID will expire. ##A value of zero will allow unlimited uses.

##Read Auth role
##Here it will give you role id which you need to use in vault-approle-issuer.yml
...

vault read auth/approle/role/my-role/role-id
...

##Generate secret id
...

vault write -f auth/approle/role/my-role/secret-id
...

##By using above 2 command role id and secret id you need to pass in below command
...

vault write auth/approle/login role_id=<role-id> secret_id=<secret-id>
...

#####
##If the command successful then vault configuration and authentication via approle is completed
#####

##YAML files to be modified
##First execute below yaml file
...

kubectl apply -f cert-manager.yaml
...

##Need to replace with the latest secret id in base64 format by using below command
##Secret id already generate when we are executing vault command, need to use same secret id here
...

echo secret-id | base64
...

##The output of above command has to be replaced in the vault-apply-secret.yml file data.secretId
...

kubectl apply -f vault-apply-secret.yml
...

##No you will get one ip where your vault is running so that ip you can get by using below command
##Copy vault ip from below command
...

kubectl get svc
...

##Now vault ip and role id need to replace in vault-approle-issuer.yml file
##Role id already generated when we are executing vault commands
...

kubectl apply -f vault-approle-issuer.yml
...

##NOTE: spec.vault.server: IP here you need to change vault ip which you will get when u ren 'kubectl get svc'
##spec.vault.auth.roleId this is you need to replace and need to put latest role id which you get in 'vault read auth/approle/role/my-role/role-id'

##Then final we need to execute below yaml file
...

kubectl apply -f vault-cert-certificate.yml
...

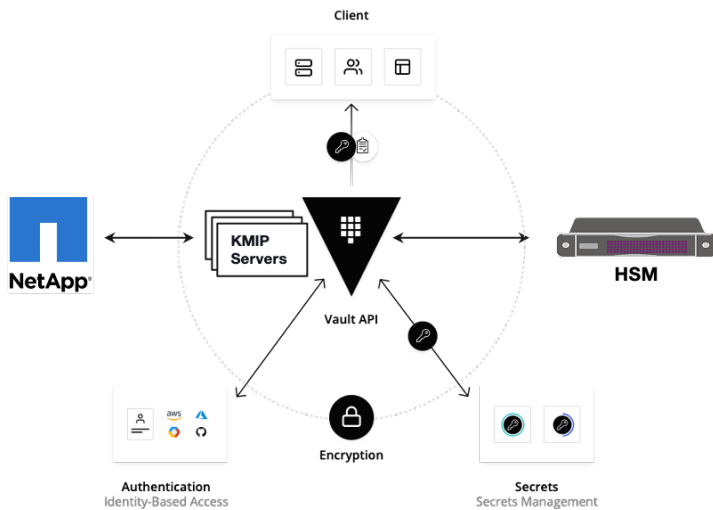
#####
Certificate generate process completed
#####

##Now get ca certificate use below command
...

curl http://10.43.130.35:8200/v1/pki/ca/pem
...

##10.43.130.35 is your vault ip, need to replace with latest vault ip

```



Troubleshooting

N/A

Maintenance

Blueprint Package Maintenance

Software maintenance

N/A

Hardware maintenance

N/A

Blueprint Deployment Maintenance

N/A

Frequently Asked Questions

N/A

License

Any software developed by the "Akraio Enterprise Applications on Lightweight 5G Telco Edge Project" is licensed under the Apache License, Version 2.0 (the "License"); you may not use the content of this software bundle except in compliance with the License. You may obtain a copy of the License at <https://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

License information of EALTEdge Blueprint Components

OCD Host

CENTER Node

Center Node consists of 3 components . MECM , Appstore and Developer Portal.

Refer:

MECM Edge Gallery http://docs.edgegallery.org/zh_CN/latest/Projects/MECM/MECM.html#

S. No	Software	Type	Version	License	Remarks
1.	Docker	CRI	18.09	Apache 2.0 <i>license</i>	No code modifications done
2.	Kubernetes	Orchestration	v1.18.7	Apache 2.0 <i>license</i>	No code modifications done
3.	Edge Gallery	Opensource MEC Platform	1.1.1	Apache 2.0 <i>license</i>	No code modifications done

Edge Node

S. No	Software	Type	Version	License Information	Remarks
1.	Docker	CRI	18.09	Apache 2.0 <i>license</i>	No code modifications done
2.	K8s	Orchestration	1.18.7	Apache 2.0 <i>license</i>	No code modifications done
3.	Edge Gallery	Opensource MEC platform	1.1.1	Apache 2.0 <i>license</i>	Open Source MEC Platform

References

Definitions, acronyms and abbreviations

Abbreviations

- EALTEdge - Enterprise Application on Lightweight 5G Telco Edge (EALTEdge).
- MECM - Multi Access Edge Computing Manager.
- MEC - Multi Access Edge Computing.
- MEP - Multi Access Edge Platform.