R3 - Installation Documentation of Enterprise Applications on Lightweight 5G Telco Edge (EALTEdge)

Introduction

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

This is the first release for this blueprint, the 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, MECM and MEC Hosts will be installed. The number of nodes in MECM cluster and MEC Host cluster is configurable.

The MECM is a K8s Cluster and MEC host is a K3s 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
- MECM Node
- MEC Hosts 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.

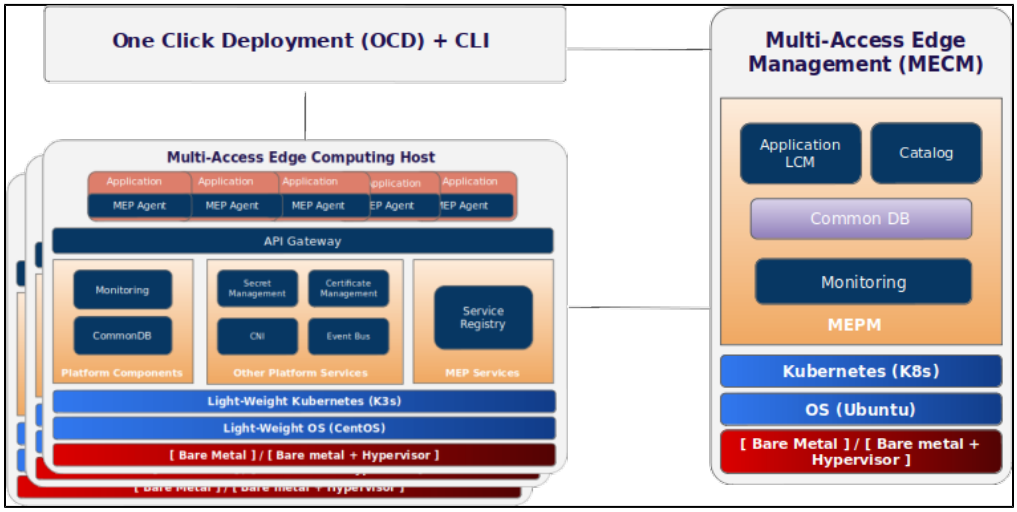


Figure: EALTEdge Deployment Architecture

***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 R3 - 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) MECM

3) MEC Host

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

Minimum Hardware Requirements

MECM	
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

MEC Host(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

MECM	
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

MEC Host(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 16.04 /18.04 for MECM Node.

- Virtual Machines preinstalled with Ubuntu 16.04 / 18.04 or CentOS 7.X 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 > 2.5 installed in One Click Deployment Node (Jump Host)
- git installed in Jump Host.
- Kubespray code is downloaded (https://github.com/kubernetes-sigs/kubespray.git)
- GO Lang (version greater than 1.12) installed in Jump Host, required for CLI.

Database Prerequisites

Schema scripts

N/A

Other Installation Requirements

Jump Host Requirements

Network Requirements

- Internet connectivity in Jump Host, MECM and MEC Hosts.
- The MECM Master Node and MEP Master 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 **three** Virtual machines, following are the virtual machines and their usage

No	Usage
1	Jump Host (One Click Deployment Node)
2	MECM (Controller)

3	MEC Host (Edge Node)

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

In this release providing two ways to install the EALTEdge environment.

i) EALTEdge Deployment using Ansible-Playbook single command.

ii) EALTEdge Deployment using CLI

Standard Deployment Overview

Jump Host Software Installations:

Login to the Jump Host and perform the below steps:

1. Install Ansible > 2.5.[https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html]
2. Install git
3. Install GoLang > 1.12
4. Git clone the Kubespray repo. [https://github.com/kubernetes-sigs/kubespray.git]
5. Install python3 and pip3
6. cd kubespray && pip install -r requirements.txt

Jump Host Pre-Configurations for MECM Components Installation

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

1. Generate public key : #ssh-keygen

2. Copy the ssh key to all the nodes in the MECM Cluster, using ssh-copy-id. (example : #ssh-copy-id root@159.178.17.16)

3. Kubespray configuration

cd kubespray && cp -rfp inventory/sample inventory/mycluster

Update ansible inventory file provided in kubesrpay repo with MECM Cluster node IP's

Example:

declare -a IPS=(158.176.15.55 158.176.15.56 158.176.15.57)

CONFIG_FILE=inventory/mycluster/hosts.yaml python3 contrib/inventory_builder/inventory.py ${IPS[@]}

4. Review and Change Parameters under inventory/mycluster/group_vars

cat inventory/mycluster/group_vars/all/all.yml
cat inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml

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-0002:~# git clone "https://gerrit.akraino.org/r/ealt-edge"

2. go to the below directory

root@akraino-mec-0002:~# cd ealt/infra/playbooks

3. Modify the Configuration File : ealt-inventory.ini with the details of MECM and MEC Hosts.

root@akraino-mec-0002:~# vi ealt-inventory.ini

4. Execute the below command
**Setup develop environment -

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --extra-vars "operation=install mode=dev"

Once the execution is completed in console will see prompt "EALTEdge Environment Installed in unsecured mode , Components Install MECM and MEC Hosts Successfully"

**Setup Production environment

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --extra-vars "operation=install mode=prod"

Once the execution is completed in console will see prompt "EALTEdge Environment Installed in secured mode , Components Install MECM and MEC Hosts Successfully"

Other Options:

To Install only MECM Node

** Install in unsecured mode - Dev mode

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --tags "mecm" --extra-vars "operation=install mode=dev"

** Install in secured mode

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --tags "mecm" --extra-vars "operation=install mode=prod"

To Install only MEC Host
** Install in unsecured mode

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --tags "mep" --extra-vars "operation=install mode=dev"

** Install in secured mode

root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --tags "mep" --extra-vars "operation=install mode=prod"

Installing Mode : EALTEdge using CLI

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

root@akraino-mec-0002:~# git clone "https://gerrit.akraino.org/r/ealt-edge"

2. go to CLI directory

root@akraino-mec-0002:~#cd ealt/infra/cli/

root@akraino-mec-0002:~#chmod +x build _cli.sh

root@akraino-mec-0002:~#source build_cli.sh

root@akraino-mec-0002:~#cd ~/ealt-edge/ocd/infra/playbooks

Note: The CLI Commands to setup the environment should be executed only from above path.

3. Edit the Configuration File : ealt-inventory.ini with the details of MECM and MEC Hosts.

root@akraino-mec-0002:~#vi ealt-inventory.ini

4. Execute the below command to install EALTEdge Environment

In non secure mode

root@akraino-mec-0002:~#ealt init all

or

root@akraino-mec-0002:~#ealt init all --mode dev

In Secure Mode (TLS Enabled)

root@akraino-mec-0002:~#ealt init all --mode prod

Above command will install the various component. Refer the below table for detail information.

Other CLI Commands:

Command to Install only MECM Node.

root@akraino-mec-0002:~#ealt init manager

To Install MECM node in Secure Mode

root@akraino-mec-0002:~#ealt init manager -m prod

Command to Install only MEC Host (MEP Node)

root@akraino-mec-0002:~#ealt init edge

or

root@akraino-mec-0002:~#ealt init edge -m dev

MEC Host in Secure (TLS/HTTPS) Mode

root@akraino-mec-0002:~#ealt init edge -m prod

Following packages will be installed in the cluster in the respective nodes:-

MECM pkgs: Docker, Kubernetes , Helm, Grafana, Calico, PostgresSQL and AppLCM.

MEC Host pkgs: Docker, K3S, Helm, RabbitMQ, Prometheus, CAdvisor, Kong, Vault, Cert-Manager , PostgresSQL, MEP Server and MEP Agent.

Deploying Application Packages : Using CLI

Prerequisite : The EALTEdge Environment should be installed. One MECM and one MEC Host at least.

Pre-configuration : Need to set the below parameters as environment variables on the Virtual Machine / Host where EALTEdge Command Line Interface tool is being installed.

export ONBOARDPACKAGEPATH= <Path where Application CSAR Package is kept in the Node where CLI is running>
export MECMClusterIP=<IP of MECM Node> #IP of Node where MECM is installed.
export MECMClusterPort=30001 #(Node Port of applcm service running on MECM)

To enable HTTPS Mode while using CLI, additional configurations have to be done. The configurations are as follows:

export CertificateKeyFile=<Path of CA Cert>
export EALTSSLMode=1 #If need to disable https mode modify the flag to 0

export MECMClusterIP='edgegallery.org' #Map the DNS Name to the MECM IP in /etc/hosts file.

Command Supported:-

Application Package Management CLI Commands:

1. Package On Boarding : root@akraino-mec-0002:~#ealt app create -f "sample_app.csar "
2. Package Info Query: root@akraino-mec-0002:~#ealt app info -i "<package id number>"
3. Package Delete: root@akraino-mec-0002:~#ealt app delete -i "<package id number>"

Application Life Cycle Management CLI Commands:

1. Application Instance Creation on MEC Host: root@akraino-mec-0002:~#ealt applcm create -i appdid "<id>" -n "<name>" -d "<description>"
2. To start the application instance on MEC Host: root@akraino-mec-0002:~#ealt applcm start -i appid "<id>" -n "<hostname>" -o "<hostip>"
3. To get the application instance details : root@akraino-mec-0002:~#ealt applcm info -i appid "<id>"
4. To terminate the application instance: root@akraino-mec-0002:~#ealt applcm kill -i appid "<id>"
5. To delete the application on MEC Host: root@akraino-mec-0002:~#ealt applcm delete -i appid "<id>"

Snapshot Deployment Overview

Not Applicable

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 MECM Node and check whether K8S cluster is installed.
2. Check grafana and AppLCM services are running as PODS.
3. Login to MEC Host and check K3S is installed.

Developer Guide and Troubleshooting

Uninstall Guide

Using Ansible Playbooks

root@akraino-mec-0002:~#ansible-playbook ealt-all-uninstall.yml -i ealt-inventory.ini --extra-vars "operation=uninstall"
root@akraino-mec-0002:~#ansible-playbook ealt-all-uninstall.yml -i ealt-inventory.ini --tags "infra" --extra-vars "operation=uninstall"
root@akraino-mec-0002:~#ansible-playbook ealt-all-uninstall.yml -i ealt-inventory.ini --tags "mecm" --extra-vars "operation=uninstall"
root@akraino-mec-0002:~#ansible-playbook ealt-all-uninstall.yml -i ealt-inventory.ini --tags "mep" --extra-vars "operation=uninstall"

Using CLI

root@akraino-mec-0002:~#ealt clean all
root@akraino-mec-0002:~#ealt clean mecm
root@akraino-mec-0002:~#ealt clean mep

Vault documentation

```
**This document explains how to generate certificate by using vault and cert manager**
##Cluster Architecture
![](Vault_Cert_Arch)
##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
ealtedge.com is a your common_name or base url
```
vault write pki/root/generate/internal common_name=ealtedge.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=ealtedge.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.ealtedge.com
```

####Enabling AppRole in Vault
```
vault auth enable approle
```

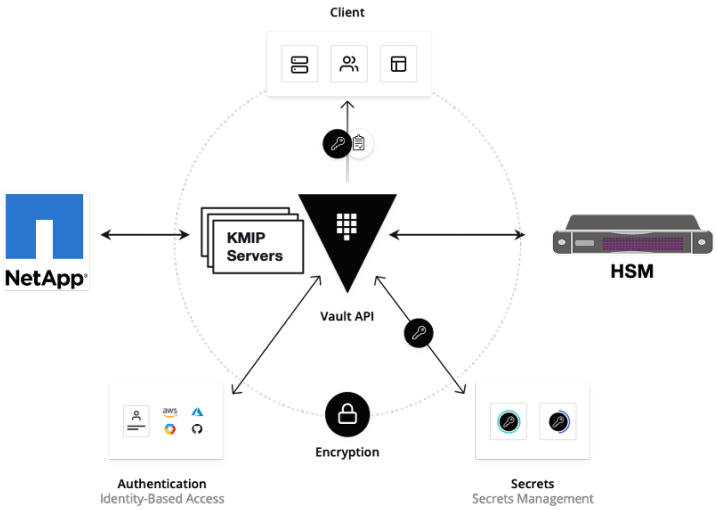
##Writing vault policy
```
vault policy write pki-policy -<<EOF
path "pki*" { capabilities = ["create", "read", "update", "delete", "list", "sudo"]}
EOF
```
##Write Auth role
```
vault write auth/approle/role/my-role secret_id_ttl=8760h token_num_uses=0 token_ttl=2160h token_max_ttl=8760h secret_id_num_uses=0 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



Kong documentation

```
**This document explains how to configure mep and kong**
##set up the EALT Environment. Refer the Installation Guide[Specify the installation guide wiki url]
##Mep will install as a pod
```

##First create setup by using below command server running in https
```
ansible-playbook ealt-all.yml -i ealt-inventory.ini --extra-vars "operation=install mode=prod"
```
##For http use below command
```
ansible-playbook ealt-all.yml -i ealt-inventory.ini --extra-vars "operation=install mode=dev"
```
##Mep will install as a pod
##MEP services will be running in the MEC Host (MEP)
##Check the mep running or not use below command
```
kubectl get po -n mep
```
##Output -

```
kubectl get svc -n mep
```
##Output -

##Configure Kong
##HTTPS Env - Rest API request Adding Services and Adding Route in Kong API Gateway
```
https://www.ealtedge.org:30012/services
https://www.ealtedge.org:30012/routes
```
##DNS mapping should be done correctly

##HTTPS Env - Rest API request deleting route and services
```
https://www.ealtedge.org:30012/services/http-mp1
https://www.ealtedge.org:30012/routes/mp1
```
##HTTP Env - Rest API request Adding Services and Adding Route in Kong API Gateway
```
http://www.ealtedge.org:30011/services
http://www.ealtedge.org:30011/routes
```
##DNS mapping should be done correctly

##HTTP Env - Rest API request deleting route and services
```
http://www.ealtedge.org:30011/services/http-mp1
http://www.ealtedge.org:30011/routes/mp1
```

##To verify the configurations done in Kong.
##Trigger the below Rest API

##In HTTP mode.
```
http://www.ealtedge.org:30010/mp1/mep/mec_service_mgmt/v1/applications/id123456/services
```
##In HTTPS Mode
```
https://www.ealtedge.org:30011/mp1/mep/mec_service_mgmt/v1/applications/id123456/services
```

##Get certificate from MEP server
##Login MEP server
```
cd /tmp/mepserver/deploy/
```
##Given path you will get all required certificates
```
ca.crt
tls.key
tls.crt
```
##ca.crt client certificate
##tls.key is a server key
##tls.crt server certificate

```
[root@huawei-akraino-0001 ~]# kubectl get po -n mep
NAME                              READY   STATUS      RESTARTS   AGE
mep                               1/1     Running     0          22h
postgres-db-5c8949f6b9-6bnsm      1/1     Running     0          22h
prepare-postgre-db-vwll8          0/1     Completed   0          22h
apigw-kong-6fc6db94ff-42d8g       1/1     Running     0          22h
mepagent                          1/1     Running     22         22h
[root@huawei-akraino-0001 ~]#
```

```
[root@huawei-akraino-0001 ~]# kubectl get svc -n mep
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                                  AGE
mep-service    NodePort    10.43.42.165    <none>        8088:30088/TCP,8443:30443/TCP                            22h
kong-service   NodePort    10.43.19.43     <none>        8000:30010/TCP,8443:30011/TCP,8001:30012/TCP,8444:30013/TCP   22h
postgres-db    ClusterIP   10.43.227.195   <none>        5432/TCP                                                 22h
[root@huawei-akraino-0001 ~]#
```

Troubleshooting

Error Message Guide

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 "Akraino 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

S. No	Software	Type	Version	License	Remarks
1.	Kubespray	K8S Tool	2.13	Apache 2.0 *license*	

MECM

S. No	Software	Type	Version	License	Remarks
1.	Docker	CRI	18.06	Apache 2.0 *license*	
2.	Kubernetes	Orchestration	1.16	Apache 2.0 *license*	
3.	Helm	Application Package Manager	3.0.2	Apache 2.0 *license*	
4.	Grafana	Monitoring	8-7.66.0	Apache 2.0 *license*	
5.	Calico	CNI Plugin	3.14.0	Apache 2.0 *license*	
6.	PostgresSQL	DB	9.6	*PostgreSQL License*	
7.	AppLCM	MECM-Service	1.0	Apache 2.0 *license*	Code is part of Akraino Code Repo. Includes Broker, Helm Plugin and Catalog

MEC Host

S. No	Software	Type	Version	License Information	Remarks
1.	Docker	CRI	19.03	Apache 2.0 *license*	
2.	K3S	Orchestration	1.18.2	Apache 2.0 *license*	
3.	Helm	Application Package Manager	3.0.2	Apache 2.0 *license*	
4.	cAdvisor	Container Metrics	v0.36.0	Apache 2.0 *license*	
5	RabbitMQ	Message Broker	3.7	Mozilla Public *License*	No code modifications done. RabbitMQ image is deployed as is.
6	Prometheus	Metrics Collector	9.3.1	Apache 2.0 *license*	Internally its installing following metrics exporter NodeExporter, alertManager, kubeStateMetrics, pushgateway
7	Kong	API Gateway	1.5.1	Apache 2.0 *license*	
8	Vault	Secret Management	0.5.0	Mozilla Public License 2.0	No code modifications done. Vault image is deployed as is.
9	Cert-Manager	Certificate Management	0.15.0	Apache 2.0 *license*	
10.	Postgres SQL	Database	9.6	*PostgreSQL License*	

10	MEP Server	EALTEdge MEP Platform Service	1.0	Apache 2.0 *license*	Code is part of Akraino Code Repository.
11	MEP Agent	EALTEdge MEP Agent Library	1.0	Apache 2.0 *license*	Code is part of Akraino Code Repository.

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.