R4 - 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 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 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
- 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.

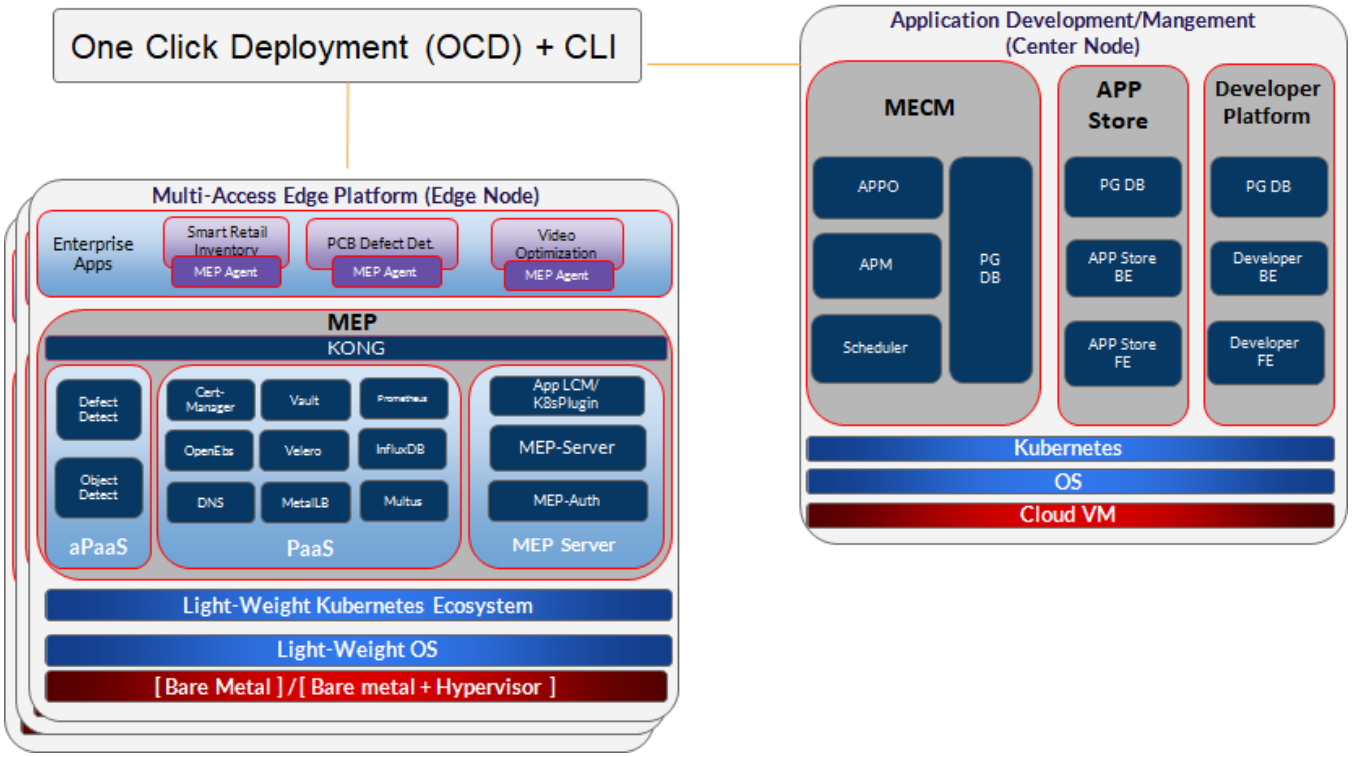


Figure: EALTEdge Deployment Architecture

***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 R4 - 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 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 1.14.2) installed in Jump Host, required for CLI.

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 **three** Virtual machines, 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.

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 > 2.9.6 [https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html]
2. Install git
3. Install GoLang > 1.14+
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 Center 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

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

5. Alter the config.yml and change parameters under path

```
######### Common parameters #########
######### Mandatory #########
private_repo_ip:
  name:

######### Optional #########
eg_image_tag:
  name: 0.9
# User can either give common pwd or can opt to provide individual pwds
common_pwd:
  name:

########## Edge config #############
######### Mandatory #########
mep_kong_pg_pwd:
  name:
######### Optional #########
edge_management_interface:
  name: eth0
edge_dataplane_interface:
  name: eth1
eg-management-address:
  name: 100.1.1.2/24
eg-dataplane-address:
  name: 200.1.1.2/24
```

```
# All Edge related password which needs to be specified if user doesn't
# need common password for security purpose
mep_pg_admin_pwd:
  name:
mep_cert_pwd:
  name:
generate_cert_pass:
  name:
mecm_mepm_postgresPassword:
  name:
mecm_mepm_postgresLcmCntlrPassword:
  name:
mecm_mepm_postgresk8sPluginPassword:
  name:

########### Center config ###########
######### Mandatory #########
mecm_meo_edgeRepoUserName:
  name:
mecm_meo_edgeRepoPassword:
  name:
```

```
######### Optional #########
usermgmt_port:
  name: 30067
appstore_port:
  name: 30091
developer_port:
  name: 30092
mecm_port:
  name: 30093
docker_registry_port:
  name: 5000
prometheus_node_port:
  name: 30009

# All Center related password which needs to be specified if user
# doesn't need common password for security purpose
user_mgmt_encryptPassword:
  name:
mecm_meo_keystorePassword:
  name:
mecm_meo_truststorePassword:
  name:
mecm_meo_postgresPassword:
  name:
mecm_meo_postgresApmPassword:
  name:
mecm_meo_postgresAppoPassword:
  name:
mecm_meo_postgresInventoryPassword:
  name:
```

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 CENTER and EDGE Nodes.

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

4. Execute the below command
**Setup environment -

root@akraino-mec-0002:~# 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"

Other Options:

To Install only Edge node
root@akraino-mec-0002:~# ansible-playbook ealt-all.yml -i ealt-inventory.ini --tags "edge" --extra-vars "operation=install"

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

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

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

Other CLI Commands:

Command to Install only Edge Node

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

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 Center Node and check whether K8S cluster is installed.
2. Check the below mentioned components and services are running as Pods / Services in Kubernetes cluster
    a. PostgresSQL
    b. AppLCM
    c. Appo
    d. Inventory
    e. Apm
    f. MECM - FrontEnd
    g. Appstore
    h. Developer Portal
    i. Service Center
    j. User Management
3. Login to Edge Host and check K3S is installed.

Components and Services running in CENTER Node

```
root@[redacted]:~# kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
appstore-be-0                           1/1     Running   0          20m
appstore-be-postgres-0                  1/1     Running   0          20m
appstore-fe-5854976dcf-zc6d4            1/1     Running   0          20m
developer-be-0                          1/1     Running   0          20m
developer-be-postgres-0                 1/1     Running   0          20m
developer-fe-7bb8b865bf-gtmgk           1/1     Running   0          20m
kubernets-deployment-77698bff7d-5qcch   1/1     Running   0          17m
mec-grafana-5fc44fc96d-qt4gz            1/1     Running   0          20m
mecm-apm-6fddb675b-fjr45                1/1     Running   0          20m
mecm-appo-7cf4d8745f-sjqk6              1/1     Running   1          20m
mecm-fe-64f4786f4f-qj69z                1/1     Running   1          20m
mecm-inventory-75876d8f45-tb4sf         1/1     Running   1          20m
mecm-postgres-0                         1/1     Running   0          20m
service-center-5cc4dcb6f7-7xhhb         1/1     Running   0          20m
user-mgmt-6df4db56b8-9zstk              1/1     Running   0          20m
user-mgmt-postgres-0                    1/1     Running   0          20m
user-mgmt-redis-0                       1/1     Running   0          20m
root@[redacted]:~# kubectl get svc
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
appstore-be-postgres-svc      ClusterIP   10.233.30.3     <none>        5432/TCP         20m
appstore-be-svc               ClusterIP   10.233.6.69     <none>        8099/TCP         20m
appstore-fe-svc               NodePort    10.233.0.5      <none>        8443:30091/TCP   20m
developer-be-postgres-svc     ClusterIP   10.233.25.16    <none>        5432/TCP         20m
developer-fe-svc              NodePort    10.233.39.242   <none>        8443:30092/TCP   20m
kubernetes                    ClusterIP   10.233.0.1      <none>        443/TCP          28m
mec-grafana                   NodePort    10.233.38.81    <none>        80:30000/TCP     20m
mecm-apm                      NodePort    10.233.47.191   <none>        8092:30202/TCP   20m
mecm-appo                     NodePort    10.233.56.131   <none>        8091:30201/TCP   20m
mecm-fe-svc                   NodePort    10.233.61.62    <none>        8443:30093/TCP   20m
mecm-inventory                NodePort    10.233.4.18     <none>        8093:30203/TCP   20m
mecm-postgres                 ClusterIP   10.233.19.83    <none>        5432/TCP         20m
service-center                ClusterIP   10.233.49.162   <none>        30100/TCP        20m
user-mgmt-postgres-svc        ClusterIP   10.233.22.178   <none>        5432/TCP         20m
user-mgmt-redis-svc           ClusterIP   10.233.54.183   <none>        6379/TCP         20m
user-mgmt-svc                 NodePort    10.233.16.0     <none>        8067:30067/TCP   20m
```

Components and Services running EDGE Node

```
root@[redacted]:~# kubectl get pods
NAME                                                READY   STATUS    RESTARTS   AGE
mep-prometheus-kube-state-metrics-664f799f67-nlxp8  1/1     Running   0          5h17m
mep-prometheus-node-exporter-j7gcm                  1/1     Running   0          5h17m
cadvisor                                            1/1     Running   0          5h17m
mep-prometheus-pushgateway-974bb5cb7-fbnxs          1/1     Running   0          5h17m
mepm-postgres-0                                     1/1     Running   0          5h16m
rabbitmq-0                                          1/1     Running   0          5h17m
mecm-mepm-lcmcontroller-767986fc6f-flll9            1/1     Running   0          5h16m
mecm-mepm-k8splugin-6cd6994685-lqddz                1/1     Running   0          5h16m
mep-prometheus-server-9996ddd7f-c29vd               2/2     Running   0          5h17m
mep-prometheus-alertmanager-b49844896-hqhgq         2/2     Running   0          5h17m
rabbitmq-1                                          1/1     Running   0          5h16m
rabbitmq-2                                          1/1     Running   0          5h15m
kubernets-deployment-7494ddbf6c-tfh4v               1/1     Running   0          5h13m
root@[redacted]:~# kubectl get svc
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                          AGE
kubernetes                          ClusterIP   10.43.0.1       <none>        443/TCP                          5h17m
rabbitmq                            NodePort    10.43.62.195    <none>        15672:31672/TCP,5672:30672/TCP   5h17m
mep-prometheus-node-exporter        ClusterIP   None            <none>        9100/TCP                         5h17m
mep-prometheus-kube-state-metrics   ClusterIP   None            <none>        80/TCP                           5h17m
mep-prometheus-server               ClusterIP   10.43.36.63     <none>        80/TCP                           5h17m
mep-prometheus-alertmanager         ClusterIP   10.43.57.199    <none>        80/TCP                           5h17m
mep-prometheus-pushgateway          ClusterIP   10.43.184.131   <none>        9091/TCP                         5h17m
mecm-mepm-lcmcontroller             NodePort    10.43.103.162   <none>        8094:30204/TCP                   5h16m
mepm-postgres                       ClusterIP   10.43.154.253   <none>        5432/TCP                         5h16m
mecm-mepm-k8splugin                 NodePort    10.43.14.158    <none>        8095:30205/TCP                   5h16m
```

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 "edge" --extra-vars "operation=uninstall"

Using CLI

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

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
```

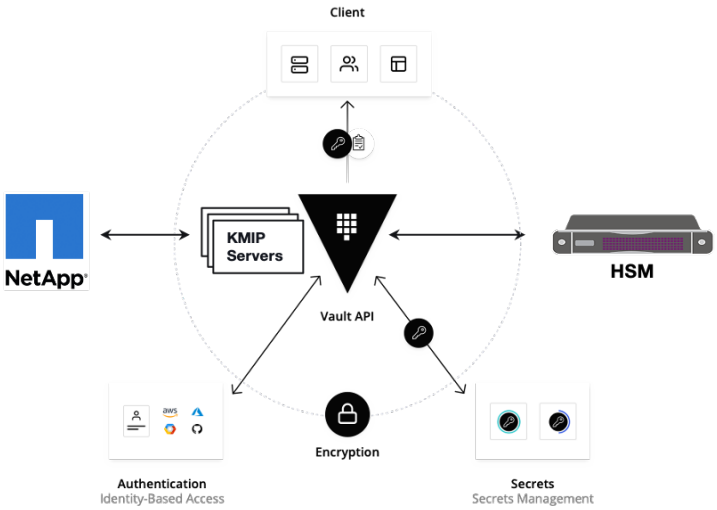
##Writing vault policy
```
vault policy write pki-policy -<<EOF
path "pki*" { capabilities = ["create", "read", "update", "delete", "list", "sudo"]}
EOF
```
##Write Auth role
```
vault write auth/approle/role/my-role secret_id_ttl=8760h token_num_uses=0 token_ttl=2160h token_max_ttl=8760h secret_id_num_uses=0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



Troubleshooting

Error Message Guide

Error: Most of mecm-pods down state including service center pod,

In service center pod logs , "decryption password incorrect" issue

Solution: Please use common_pwd in config.yaml as **te9Fmv%qaq**

Maintenance

Blueprint Package Maintenance

Software maintenance

N/A

Hardware maintenance

N/A

Blueprint Deployment Maintenance

N/A

Frequently Asked Questions

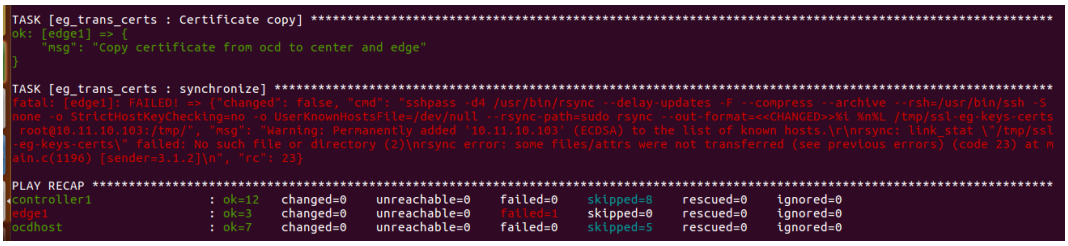
1) k3s not installed on mec host (edge node) Port 6443

This issue might be due to Port on EDGE node 6443 occupied by some other process. In such case, it will not install k3s and ansible command will fail.

Solution:

We need to release port 6443 and completely uninstall the platform and re-install platform

2) eg_trans_certs failure:



The above issue during installation process due to certificate generation again and again continuously without uninstallation process.

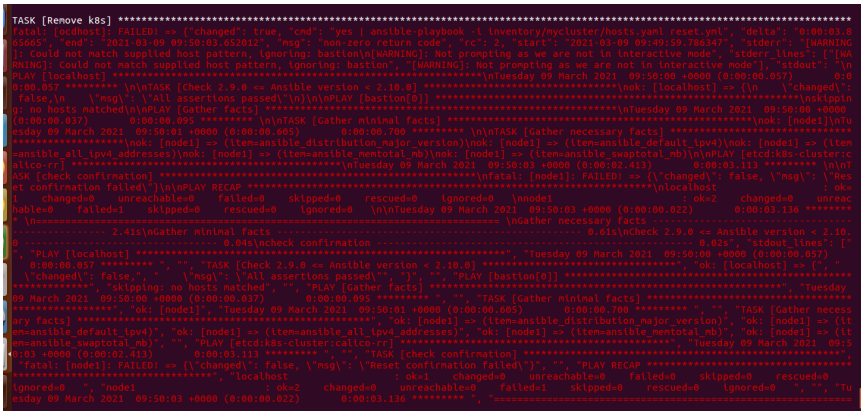
Every time our mec edge node installation requires new sets of certs to download and install the Edge gallery related things i.e pods services and so on.

Solution:

So, if once the installation process failed. We ought to completely uninstall and re-install the ealt-edge platform using ansible commands.

Please refer the installation and uninstallation steps provdied above using ansible commands

3) K8s remove / uninstallation issue through ansible



If k8s not getting removed or not installed properly or if version changed than our recommended version 1.17.2,

Solution:

In such cases, there might be problem in configuration of kubespray folder and its relevant configurations.

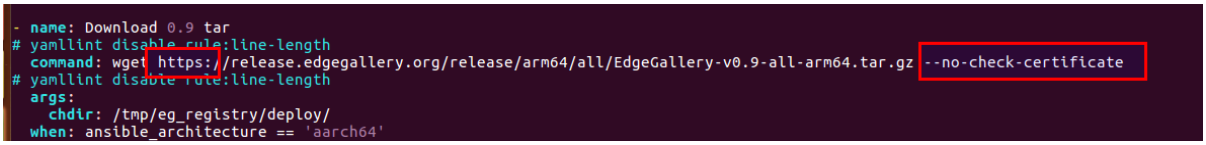
We recommend to configure pre-requisite kubepspray prior to installation process

4) eg_registry Download 0.9 tar : issue

If downloading edge gallery 0.9 version tar failed for no reason,

Solution: We can add –no-check-certificate in the wget of Download 0.9 tar in install.yml of eg_registry role. This issue occurs in extremely rare secure environments.

We will be fixing this issue ASAP.



License

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	2.14.2	Apache 2.0 *license*	No code modifications done
2.	Helm	Application Package Manager	3.0.2	Apache 2.0 license	No code modifications done

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	19.03+	Apache 2.0 *license*	No code modifications done
2.	Kubernetes	Orchestration	v1.17.2	Apache 2.0 *license*	No code modifications done
3.	Helm	Application Package Manager	3.0.2	Apache 2.0 *license*	No code modifications done
4.	Grafana	Monitoring MEC-Graphana	7.1.1	Apache 2.0 *license*	Code part of Edge Gallery
5.	Calico	CNI Plugin	3.16.5	Apache 2.0 *license*	No code modifications done
6.	PostgresSQL	MECM-Service	12.3	*PostgreSQL License*	Code part of Edge Gallery
7.	AppLCM	MECM-Service	0.9	Apache 2.0 *license*	Code part of Edge Gallery
8.	Appo	MECM-Service (MECM-Appo)	0.9	Apache 2.0 *license*	Code part of Edge Gallery
9	Inventory	MECM-Service	0.9	Apache 2.0 *license*	Code part of Edge Gallery
10	Apm	MECM-Service	0.9	Apache 2.0 *license*	Code part of Edge Gallery
11	User Management	Part of Center Node	0.9	Apache 2.0 *license*	Code part of Edge Gallery
12	MECM - FrontEnd	MECM-Service	0.9	Apache 2.0 *license*	Code part of Edge Gallery
13.	Appstore	Service (Part of Center Node)	0.9	Apache 2.0 *license*	Code part of Edge Gallery
14.	Developer Portal	Service (Part of Center Node)	0.9	Apache 2.0 *license*	Code part of Edge Gallery
15	Service Center	Service (Part of Center Node)	0.9	Apache 2.0 *license*	Code part of Edge Gallery

Edge Node

S. No	Software	Type	Version	License Information	Remarks
1.	Docker	CRI	19.03+	Apache 2.0 *license*	No code modifications done
2.	K3S	Orchestration	1.19.4+	Apache 2.0 *license*	No code modifications done
3.	Helm	Application Package Manager	3.0.2	Apache 2.0 *license*	No code modifications done
4.	cAdvisor	Container Metrics	v0.36.0	Apache 2.0 *license*	No code modifications done
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*	Code part of Edge Gallery
7	mepm-postgres	Service Database	12.3	*PostgreSQL License*	Code part of Edge Gallery
8	MEP	Pod	0.9	Apache 2.0 *license*	Code part of Edge Gallery
9	MECM-MEPM	MEPM-Service	0.9	Apache 2.0 *license*	Code part of Edge Gallery

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.