**First steps:**

**Install aws client**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

aws –version

aws configure

**Install eksctl**

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

**Install kubectl**

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

curl -LO https://dl.k8s.io/release/v1.21.0/bin/linux/amd64/kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

**Install helm**

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

helm version

**Deploy cluster:**

eksctl create cluster --version 1.21 --region eu-west-3 --node-type t3a.large --nodes 2 --nodes-min 1 --nodes-max 3 --vpc-nat-mode Disable --ssh-access --ssh-public-key=5gmeta-cluster-key --region=eu-west-3 --name 5gmeta-cloud

eksctl create cluster -f cluster.yaml

eksctl get cluster

aws eks describe-cluster --name 5gmeta-cloud --output text

helm repo add 5gmeta <https://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@raw.githubusercontent.com/5gmetadmin/helmcharts/main/repository>

kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-username=20211119 --docker-password=74d654c2-d930-4299-bcc3-e24e936cfae9

**Create an IAM OIDC provider:**

aws eks describe-cluster --name 5gmeta-cloud --query "cluster.identity.oidc.issuer" --output text

<https://oidc.eks.eu-west-3.amazonaws.com/id/FA121F5CDBE74953E14413D6E8FB8B2F>

aws iam list-open-id-connect-providers | grep FA121F5CDBE74953E14413D6E8FB8B2F

eksctl utils associate-iam-oidc-provider --cluster 5gmeta-cloud --approve

arn:aws:iam::305612797790:oidc-provider/oidc.eks.eu-west-3.amazonaws.com/id/FA121F5CDBE74953E14413D6E8FB8B2F

**Install AWS Load Balancer Controller add-on:**

The AWS Load Balancer Controller manages AWS Elastic Load Balancers for a Kubernetes cluster. The controller provisions the following resources.

An AWS Application Load Balancer (ALB) when you create a Kubernetes Ingress.

An AWS Network Load Balancer (NLB) when you create a Kubernetes service of type LoadBalancer. In the past, the Kubernetes network load balancer was used for instance targets, but the AWS Load balancer Controller was used for IP targets. With the AWS Load Balancer Controller version 2.3.0 or later, you can create NLBs using either target type. For more information about NLB target types, see Target type in the User Guide for Network Load Balancers.

curl -o iam\_policy.json <https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.1/docs/install/iam_policy.json>

aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document <file://iam_policy.json>

eksctl create iamserviceaccount --cluster=5gmeta-cloud --namespace=kube-system --name=aws-load-balancer-controller --attach-policy-arn=arn:aws:iam::305612797790:policy/AWSLoadBalancerControllerIAMPolicy --override-existing-serviceaccounts --approve

helm repo add eks <https://aws.github.io/eks-charts>

helm repo update

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=5gmeta-cloud --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller

kubectl get deployment -n kube-system aws-load-balancer-controller

**Install AWS Storage Controller**

<https://stackoverflow.com/questions/68359043/whats-the-difference-between-ebs-csi-aws-com-vs-kubernetes-io-aws-ebs-for-provi>

As per the official documentation:

<https://kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-csi-migration-beta/#why-are-we-migrating-in-tree-plugins-to-csi>

*Prior to CSI, Kubernetes provided a powerful volume plugin system. These volume plugins were “in-tree” meaning their code was part of the core Kubernetes code and shipped with the core Kubernetes binaries. However, adding support for new volume plugins to Kubernetes was challenging. Vendors that wanted to add support for their storage system to Kubernetes (or even fix a bug in an existing volume plugin) were forced to align with the Kubernetes release process. In addition, third-party storage code caused reliability and security issues in core Kubernetes binaries and the code was often difficult (and in some cases impossible) for Kubernetes maintainers to test and maintain. Using the Container Storage Interface in Kubernetes resolves these major issues.*

*As more CSI Drivers were created and became production ready, we wanted all Kubernetes users to reap the benefits of the CSI model. However, we did not want to force users into making workload/configuration changes by breaking the existing generally available storage APIs. The way forward was clear - we would have to replace the backend of the “in-tree plugin” APIs with CSI.*

So answering your question - yes, ebs.csi.aws.com is maintained by AWS while the in-tree plugin is maintained by Kubernetes but it seems like they've stopped implementing new features as per this article:

*The idea of this journey started picking up steam when I realized that the in-tree storage plugins were deprecated and no new enhancements were being made to them starting with Kubernetes 1.20. When I discovered that simply switching from gp2 to gp3 volumes meant I had to start using the AWS CSI Driver I realized I was behind the times.*

Answering your last question it's probably better to use ebs.csi.aws.com as per this note:

*The existing in-tree EBS plugin is still supported, but by using a CSI driver, you benefit from the decoupling between the Kubernetes upstream release cycle and the CSI driver release cycle.*

The Amazon Elastic Block Store (Amazon EBS) Container Storage Interface (CSI) driver allows Amazon Elastic Kubernetes Service (Amazon EKS) clusters to manage the lifecycle of Amazon EBS volumes for persistent volumes.

The Amazon EBS CSI driver isn't installed when you first create a cluster. To use the driver, you must add it as an Amazon EKS add-on or as a self-managed add-on.

curl -o example-iam-policy.json <https://raw.githubusercontent.com/kubernetes-sigs/aws-ebs-csi-driver/master/docs/example-iam-policy.json>

aws iam create-policy --policy-name AmazonEKS\_EBS\_CSI\_Driver\_Policy --policy-document <file://example-iam-policy.json>

eksctl create iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system --cluster 5gmeta-cloud --attach-policy-arn arn:aws:iam::305612797790:policy/AmazonEKS\_EBS\_CSI\_Driver\_Policy --approve --override-existing-serviceaccounts

aws cloudformation describe-stacks --stack-name eksctl-5gmeta-cloud-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa --query='Stacks[].Outputs[?OutputKey==`Role1`].OutputValue' --output text

arn:aws:iam::305612797790:role/eksctl-5gmeta-cloud-addon-iamserviceaccount-Role1-1F17M7F2O6W14

helm repo add aws-ebs-csi-driver <https://kubernetes-sigs.github.io/aws-ebs-csi-driver>

helm repo update

helm upgrade -install aws-ebs-csi-driver aws-ebs-csi-driver/aws-ebs-csi-driver --namespace kube-system --set controller.serviceAccount.create=false --set controller.serviceAccount.name=ebs-csi-controller-sa

kubectl get deployment -n kube-system ebs-csi-controller

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-sigs/aws-ebs-csi-driver/master/examples/kubernetes/dynamic-provisioning/manifests/storageclass.yaml>

kubectl get storageclass

kubectl patch storageclass gp2 -p '{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'

kubectl patch storageclass ebs-sc -p '{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

kubectl get storageclass

**Install Cluster Autoscaler**

Create cluster-autoscaler-policy.json

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"autoscaling:DescribeAutoScalingGroups",

"autoscaling:DescribeAutoScalingInstances",

"autoscaling:DescribeLaunchConfigurations",

"autoscaling:DescribeTags",

"ec2:DescribeInstanceTypes",

"ec2:DescribeLaunchTemplateVersions"

],

"Resource": ["\*"]

},

{

"Effect": "Allow",

"Action": [

"autoscaling:SetDesiredCapacity",

"autoscaling:TerminateInstanceInAutoScalingGroup"

],

"Resource": ["\*"]

}

]

}

aws iam create-policy --policy-name AmazonEKSClusterAutoscalerPolicy --policy-document <file://cluster-autoscaler-policy.json>

eksctl create iamserviceaccount --cluster=5gmeta-cloud --namespace=kube-system --name=cluster-autoscaler --attach-policy-arn=arn:aws:iam::305612797790:policy/AmazonEKSClusterAutoscalerPolicy --override-existing-serviceaccounts --approve

curl -o cluster-autoscaler-autodiscover.yaml <https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml>

Edit the cluster-autoscaler manifest to replace <YOUR CLUSTER NAME> (including <>) with the name of your cluster, and add the following options. --balance-similar-node-groups ensures that there is enough available compute across all availability zones. --skip-nodes-with-system-pods=false ensures that there are no problems with scaling to zero.

kubectl apply -f cluster-autoscaler-autodiscover.yaml

kubectl annotate serviceaccount cluster-autoscaler -n kube-system eks.amazonaws.com/role-arn=arn:aws:iam::305612797790:role/AmazonEKSClusterAutoscalerRole --aprove

kubectl patch deployment cluster-autoscaler -n kube-system -p '{"spec":{"template":{"metadata":{"annotations":{"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"}}}}}'

export K8S\_VERSION=$(kubectl version --short | grep 'Server Version:' | sed 's/[^0-9.]\*\([0-9.]\*\).\*/\1/' | cut -d. -f1,2)

export AUTOSCALER\_VERSION=$(curl -s "https://api.github.com/repos/kubernetes/autoscaler/releases" | grep '"tag\_name":' | sed -s 's/.\*-\([0-9][0-9\.]\*\).\*/\1/' | grep -m1 ${K8S\_VERSION})

kubectl set image deployment cluster-autoscaler -n kube-system cluster-autoscaler=k8s.gcr.io/autoscaling/cluster-autoscaler:v${AUTOSCALER\_VERSION}

kubectl -n kube-system logs -f deployment.apps/cluster-autoscaler

**Modify AWS Security Group for accessing NodePorts (30000-32768) from outside (if needed)**

**Monitoring cluster:**

**Installing the Kubernetes Metrics Server**

kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

kubectl get deployment metrics-server -n kube-system

kubectl get --raw /metrics

**Deploy Kubernetes Dashboard**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.1/aio/deploy/recommended.yaml>

Create an eks-admin service account and cluster role binding

Create a file called eks-admin-service-account.yaml with the text below. This manifest defines a service account and cluster role binding called eks-admin.

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: eks-admin

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: eks-admin

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: eks-admin

namespace: kube-system

kubectl apply -f eks-admin-service-account.yaml

To connect to the Kubernetes dashboard

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')

kubectl proxy

<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#!/login>

kubectl port-forward svc/kubernetes-dashboard -n kubernetes-dashboard 6443:443

**Deploy Kube-Prometheus stack**

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update

helm install -n monitoring --create-namespace -f prom-stack-values.yaml prometheus-stack prometheus-community/kube-prometheus-stack

prometheus:

prometheusSpec:

storageSpec:

volumeClaimTemplate:

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 20Gi

alertmanager:

alertmanagerSpec:

storage:

volumeClaimTemplate:

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 5Gi

PROMETHEUS=$(kubectl get pod -n monitoring -l app.kubernetes.io/name=prometheus -o jsonpath='{.items[0].metadata.name}')

GRAFANA=$(kubectl get pod -n monitoring -l app.kubernetes.io/name=grafana -o jsonpath='{.items[0].metadata.name}')

kubectl port-forward -n monitoring $PROMETHEUS 9090 &

kubectl port-forward -n monitoring $GRAFANA 3000 &

**Mysql:**

helm repo add bitnami <https://charts.bitnami.com/bitnami>

helm repo update

helm install mysql-cluster -n mysql --create-namespace bitnami/mysql --set metrics.enabled=true --set metrics.serviceMonitor.enabled=true --set metrics.serviceMonitor.namespace=monitoring --set metrics.serviceMonitor.additionalLabels.release=prometheus-stack

<https://grafana.com/grafana/dashboards/7362>

**Kafka:**

helm repo add confluentinc <https://confluentinc.github.io/cp-helm-charts/>

helm repo update

helm install confluentinc/cp-helm-charts --name kafka --set cp-control-center.enabled=false

OR

git clone <https://github.com/confluentinc/cp-helm-charts.git>

edit values files from different charts (e.g. nodeport.enabled=true) or download from github

helm install kafka-cluster -n kafka --create-namespace ./cp-helm-charts --set cp-control-center.enabled=false --set cp-zookeeper.persistence.dataDirSize=5Gi --set cp-zookeeper.persistence.dataLogDirSize=5Gi --set cp-kafka-connect.imagePullSecrets[0].name=regcred

CONFIGURE KAFKA KSQLDB LISTENERS

Import Grafana dashboard

<https://raw.githubusercontent.com/confluentinc/cp-helm-charts/master/grafana-dashboard/confluent-open-source-grafana-dashboard.json>

Kafka-UI manifest for deploying the UI

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: kafka-ui

labels:

app.kubernetes.io/component: kafka-ui

namespace: kafka

spec:

replicas: 1

selector:

matchLabels:

app.kubernetes.io/component: kafka-ui

template:

metadata:

labels:

app.kubernetes.io/component: kafka-ui

spec:

containers:

- name: kafka-ui

image: provectuslabs/kafka-ui:latest

imagePullPolicy: IfNotPresent

ports:

- name: http

containerPort: 8080

protocol: TCP

env:

- name: KAFKA\_CLUSTERS\_0\_NAME

value: 5gmeta-cloud

- name: KAFKA\_CLUSTERS\_0\_BOOTSTRAPSERVERS

value: kafka-cluster-cp-kafka-headless.kafka.svc.cluster.local:9092

- name: KAFKA\_CLUSTERS\_0\_ZOOKEEPER

value: kafka-cluster-cp-zookeeper-headless.kafka.svc.cluster.local:2181

- name: KAFKA\_CLUSTERS\_0\_SCHEMAREGISTRY

value: http://kafka-cluster-cp-schema-registry.kafka.svc.cluster.local:8081

- name: KAFKA\_CLUSTERS\_0\_KAFKACONNECT\_0\_NAME

value: kafka-connect

- name: KAFKA\_CLUSTERS\_0\_KAFKACONNECT\_0\_ADDRESS

value: http://kafka-cluster-cp-kafka-connect.kafka.svc.cluster.local:8083

- name: KAFKA\_CLUSTERS\_0\_KSQLDBSERVER

value: http://kafka-cluster-cp-ksql-server.kafka.svc.cluster.local:8088

- name: KAFKA\_CLUSTERS\_0\_JMXPORT

value: “5555”

- name: AUTH\_TYPE

value: "LOGIN\_FORM"

- name: SPRING\_SECURITY\_USER\_NAME

value: admin

- name: SPRING\_SECURITY\_USER\_PASSWORD

value: admin

---

apiVersion: v1

kind: Service

metadata:

name: kafka-ui

labels:

app.kubernetes.io/component: kafka-ui

namespace: kafka

spec:

# type: NodePort

ports:

- port: 8080

protocol: TCP

targetPort: http

# nodePort: 8080

selector:

app.kubernetes.io/component: kafka-ui

ServiceMonitor manifest for monitoring kafka metrics through kube-prometheus-stack Prometheus Operator:

---

apiVersion: monitoring.coreos.com/v1

kind: ServiceMonitor

metadata:

labels:

release: prometheus-stack

name: kafka-cluster

namespace: monitoring

spec:

endpoints:

- interval: 30s

port: metrics

namespaceSelector:

matchNames:

- kafka

selector:

matchLabels:

release: kafka-cluster

**Kubernetes Cluster Access:**

Following next tutorial: <https://www.eksworkshop.com/beginner/091_iam-groups/>

aws iam create-group --group-name k8sAdmin

aws iam create-group --group-name k8s5gmeta

ACCOUNT\_ID=305612797790

POLICY=$(echo -n '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::'; echo -n "$ACCOUNT\_ID"; echo -n ':root"},"Action":"sts:AssumeRole","Condition":{}}]}')

echo ACCOUNT\_ID=$ACCOUNT\_ID

echo POLICY=$POLICY

aws iam create-role --role-name k8sAdmin --description "Kubernetes administrator role (for AWS IAM Authenticator for Kubernetes)." --assume-role-policy-document "$POLICY" --output text --query 'Role.Arn'

aws iam create-role --role-name k8s5gmeta --description "5gmeta role for partner access (for AWS IAM Authenticator for Kubernetes)." --assume-role-policy-document "$POLICY" --output text --query 'Role.Arn'

ADMIN\_GROUP\_POLICY=$(echo -n '{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AllowAssumeOrganizationAccountRole",

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::'; echo -n "$ACCOUNT\_ID"; echo -n ':role/k8sAdmin"

},

{

"Sid": "AllowEksDescribeCluster",

"Effect": "Allow",

"Action": "eks:DescribeCluster",

"Resource": "arn:aws:eks:eu-west-3:305612797790:cluster/5gmeta-cloud"

}

]

}')

echo ADMIN\_GROUP\_POLICY=$ADMIN\_GROUP\_POLICY

aws iam put-group-policy \

--group-name k8sAdmin \

--policy-name k8sAdmin-policy \

--policy-document "$ADMIN\_GROUP\_POLICY"

5GMETA\_GROUP\_POLICY=$(echo -n '{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AllowAssumeOrganizationAccountRole",

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::'; echo -n "$ACCOUNT\_ID"; echo -n ':role/k8s5gmeta"

},

{

"Sid": "AllowEksDescribeCluster",

"Effect": "Allow",

"Action": "eks:DescribeCluster",

"Resource": "arn:aws:eks:eu-west-3:305612797790:cluster/5gmeta-cloud"

}

]

}')

echo 5GMETA\_GROUP\_POLICY=$5GMETA\_GROUP\_POLICY

aws iam put-group-policy --group-name k8s5gmeta --policy-name k8s5gmeta-policy --policy-document "$5GMETA\_GROUP\_POLICY"

aws iam list-groups

eksctl create iamidentitymapping --cluster 5gmeta-cloud --arn arn:aws:iam::${ACCOUNT\_ID}:role/k8sAdmin --username admin --group system:masters

eksctl create iamidentitymapping --cluster 5gmeta-cloud --arn arn:aws:iam::${ACCOUNT\_ID}:role/k8s5gmeta --username 5gmeta-user

kubectl get cm -n kube-system aws-auth -o yaml

eksctl get iamidentitymapping --cluster 5gmeta-cloud

Create 5gmeta-cluster-access.yaml manifest file and apply it to namespaces that will access k8s5gmeta group users:

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: 5gmeta-access-role

rules:

- apiGroups:

- ""

- "apps"

- "batch"

- "extensions"

- "autoscaling"

resources: ["\*"]

# resources:

# - "configmaps"

# - "cronjobs"

# - "deployments"

# - "events"

# - "ingresses"

# - "jobs"

# - "pods"

# - "pods/attach"

# - "pods/exec"

# - "pods/log"

# - "pods/portforward"

# - "secrets"

# - "services"

verbs: ["\*"]

# verbs:

# - "create"

# - "delete"

# - "describe"

# - "get"

# - "list"

# - "patch"

# - "update"

---

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: 5gmeta-role-binding

subjects:

- kind: User

name: 5gmeta-user

roleRef:

kind: Role

name: 5gmeta-access-role

apiGroup: rbac.authorization.k8s.io

Create users

Every user:

aws configure

aws eks update-kubeconfig --name 5gmeta-cloud --role-arn arn:aws:iam::305612797790:role/k8s5gmeta

**KSQLDB:**

kubectl run tmp-ksql-cli -n kafka --rm -i --tty --image confluentinc/cp-ksqldb-cli:7.1.0 <http://kafka-cluster-cp-ksql-server:8088>

SHOW STREAMS;

SELECT \* FROM <TOPIC> EMIT CHANGES;

**Kafka Connect:**

Connectors:

{

"connector.class": "com.datamountaineer.streamreactor.connect.jms.source.JMSSourceConnector",

"connect.jms.initial.context.factory": "org.apache.activemq.jndi.ActiveMQInitialContextFactory",

"tasks.max": 1,

"connect.jms.url": "tcp://150.241.250.4:61616",

"connect.jms.username": "5gmeta-user",

"connect.jms.connection.factory": "ConnectionFactory",

"connect.jms.queues": "jms-queue",

"connect.jms.password": "5gmeta-password",

"connect.progress.enabled": "true",

"connect.jms.kcql": "INSERT INTO cits SELECT \* FROM cits WITHTYPE TOPIC ",

"name": "jms-source-cits-vicom"

}

{

"connector.class": "com.datamountaineer.streamreactor.connect.jms.source.JMSSourceConnector",

"connect.jms.initial.context.factory": "org.apache.activemq.jndi.ActiveMQInitialContextFactory",

"tasks.max": 1,

"connect.jms.url": "tcp://150.241.250.4:61616",

"connect.jms.username": "5gmeta-user",

"connect.jms.connection.factory": "ConnectionFactory",

"connect.jms.queues": "jms-queue",

"connect.jms.password": "5gmeta-password",

"connect.progress.enabled": "true",

"connect.jms.kcql": "INSERT INTO image SELECT \* FROM image WITHTYPE TOPIC ",

"name": "jms-source-image-vicom"

}

{

"connector.class": "com.datamountaineer.streamreactor.connect.jms.source.JMSSourceConnector",

"connect.jms.initial.context.factory": "org.apache.activemq.jndi.ActiveMQInitialContextFactory",

"tasks.max": 1,

"connect.jms.url": "tcp://150.241.250.4:61616",

"connect.jms.username": "5gmeta-user",

"connect.jms.connection.factory": "ConnectionFactory",

"connect.jms.queues": "jms-queue",

"connect.jms.password": "5gmeta-password",

"connect.progress.enabled": "true",

"connect.jms.kcql": "INSERT INTO video SELECT \* FROM video WITHTYPE TOPIC ",

"name": "jms-source-video-vicom"

}

**APISIX:**

**Token request BASH**

token=$(curl -s --location --request POST 'https://5gmeta-platform.eu/identity/realms/5gmeta/protocol/openid-connect/token' --header 'Content-Type: application/x-www-form-urlencoded' \

--data-urlencode 'grant\_type=password' --data-urlencode 'username=vicomtech' --data-urlencode 'password=V1c0mt3ch' --data-urlencode 'client\_id=5gmeta\_login' | jq -r '.access\_token')

token=$(echo "Authorization: Bearer $token")

echo $token

curl -vvv --location --request GET 'https://5gmeta-platform.eu/license-api/licenses' --header "$token"

**Token request PYTHON**

import requests

headers = {

'Content-Type': 'application/x-www-form-urlencoded',

}

data = {

'grant\_type': 'password',

'username': 'vicomtech',

'password': 'V1c0mt3ch',

'client\_id': '5gmeta\_login',

}

response = requests.post('https://5gmeta-platform.eu/identity/realms/5gmeta/protocol/openid-connect/token', headers=headers, data=data)

r = response.json()

token = "Bearer " + r['access\_token']

headers = {

'Authorization': token,

}

response = requests.get('https://5gmeta-platform.eu/license-api/licenses', headers=headers)

r = response.json()

print(r)

**Credentials:**

Cloud MySQL Credentials (No admin)

5gmeta-platform

€5gm3t4pl4tf0rm!

APISIX Hardcoded Credentials

5gmeta

5Gm3t4!

**GPU:**

/etc/docker/daemon.json

{

"mtu": 1450,

"default-runtime": "nvidia",

"runtimes": {

"nvidia": {

"path": "nvidia-container-runtime",

"runtimeArgs": []

}

}

}

**INFO:**

Cloud:

- Discovery: https://5gmeta-platform.eu/discovery-api/

- SLA: https://5gmeta-platform.eu/slacloud-api/ (Usuario apisix: vicomtech / Pass apisix: V1c0mt3ch)

- Dataflow: https://5gmeta-platform.eu/dataflow-api/

\*PARA ACCEDER A SWAGGER UI AÑADIR AL PATH /ui

\*CREDENCIALES KEYCLOAK: vicomtec / V1c0mt3ch

Kafka:

- Bootstrap / Broker: 15.236.104.255:31090, 15.236.104.255:31091, 15.236.104.255:31092

- Registry: 15.236.104.255:31081

- KSQLDB: 15.236.104.255:31088

- Kafka UI: http://15.236.104.255:31080

MEC:

- Registration: http://10.0.20.12:12346/ui/

- SLA: http://10.0.20.12:5000/ui/, http://150.241.250.4:5000/

- Message-broker: http://10.0.20.12:8161/, 10.0.20.12:5673, 150.241.250.4:616161

- Video-broker: 10.0.20.12:8443

**MySQL:**

-- kubectl run -it --rm --image=mysql --restart=Never mysql-client -- mysql --host mysql --password=5GM3TA

-- kubectl run -it --rm --image=mysql --restart=Never mysql-client -- mysql --host 10.96.191.239 --password=0CgYjaJ3nC

CREATE DATABASE slaedgedb;

USE slaedgedbdb;

-- CREATE TABLE IF NOT EXISTS sla(sla\_id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT, sla\_level VARCHAR(255) NOT NULL PRIMARY KEY, total\_cpu INT NOT NULL, total\_ram INT NOT NULL, gpu BOOLEAN);

CREATE TABLE IF NOT EXISTS sla(sla\_level VARCHAR(255) NOT NULL PRIMARY KEY, total\_cpu INT NOT NULL, total\_memory INT NOT NULL, gpu BOOLEAN);

CREATE TABLE IF NOT EXISTS reservation(reservation\_id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT, reservation\_date TIMESTAMP, sla\_level VARCHAR(255), data\_type VARCHAR(255), client\_name VARCHAR(255), FOREIGN KEY (sla\_level) REFERENCES sla(sla\_level));

SHOW TABLES;

SELECT \* FROM sla;

SELECT \* FROM reservation;

EXIT;

-- ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';

-- GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'password';

-- GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' WITH GRANT OPTION;

-- FLUSH PRIVILEGES;

-- SELECT COUNT(DISTINCT sla\_level) AS some\_alias FROM reservation;

-- SELECT DISTINCT sla\_level FROM reservation;

-- SELECT COUNT(sla\_level) FROM reservation WHERE sla\_level="small";

-- SELECT total\_ram FROM sla WHERE sla\_level="small";

**MEC Offline Installation:**

sudo apt-get install --download-only apt-transport-https ca-certificates curl gnupg-agent software-properties-common git subversion python3-pip python3-docker libcurl4-openssl-dev libssl-dev

sudo apt-get install --download-only docker-ce docker-ce-cli containerd.io docker-compose-plugin libseccomp2 libseccomp2-dev

sudo apt-get install --download-only kubelet=1.20.11-00 kubeadm=1.20.11-00 kubectl=1.20.11-00

sudo apt-get install --download-only helm

sudo apt-mark hold kubelet kubeadm kubectl

sudo dpkg -i \*

two times

pip3 download kubernetes openshift pycurl pyGeoTile

pip3 install --no-index \*

two times

docker save $name -i $packetname $name

docker load -i $packetname

packetes kubeadm init

sudo kubeadm config images list

5gmeta/slaedge-api:1.0

5gmeta/video-stream-broker:1.0

5gmeta/message-broker:latest

5gmeta/registration-api:test

kubernetesui/dashboard:v2.4.0

kubernetesui/metrics-scraper:v1.0.7

rancher/mirrored-flannelcni-flannel-cni-plugin:v1.0.1

rancher/mirrored-flannelcni-flannel:v0.17.0

k8s.gcr.io/metrics-server/metrics-server:v0.6.1

quay.io/google-cloud-tools/kube-eagle:1.1.4

quay.io/prometheus/prometheus:v2.35.0

quay.io/prometheus/alertmanager:v0.24.0

quay.io/prometheus-operator/prometheus-operator:v0.56.2

quay.io/prometheus-operator/prometheus-config-reloader:v0.56.2

k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1

quay.io/thanos/thanos:v0.25.2

falta el exporter

grafana/grafana:8.5.0

curlimages/curl:7.73.0

busybox:1.31.1

quay.io/kiwigrid/k8s-sidecar:1.15.6

k8s.gcr.io/kube-state-metrics/kube-state-metrics:v2.4.1

quay.io/prometheus/node-exporter:1.3.1

bitnami/mysql:8.0.29-debian-10-r23

bitnami/bitnami-shell:10-debian-10-r431

bitnami/mysqld-exporter:0.14.0-debian-10-r75

quay.io/metallb/speaker:v0.11.0

quay.io/metallb/controller:v0.11.0

openebs/m-apiserver:1.12.0

openebs/openebs-k8s-provisioner:1.12.0

openebs/provisioner-localpv:1.12.0

openebs/snapshot-controller:1.12.0

openebs/snapshot-provisioner:1.12.0

openebs/node-disk-manager:0.7.0

openebs/node-disk-operator:0.7.0

openebs/admission-server:1.12.0

openebs/jiva:1.12.0

openebs/cstor-pool:1.12.0

openebs/cstor-pool-mgmt:1.12.0

openebs/cstor-istgt:1.12.0

openebs/cstor-volume-mgmt:1.12.0

openebs/linux-utils:1.12.0

openebs/m-exporter:1.12.0

helm instal $name $release --set image.pullPolicy=IfNotPresent

-------------------------------------------

OSM

metallb install

sudo apt-get install --download-only osm-devops git wget curl tar snapd python3-osm-im python3-osmclient libcurl4-openssl-dev libssl-dev

sudo dpkg -i \*

two times

sudo apt-get remove --purge -y liblxc1 lxc-common lxcfs lxd lxd-client

systemctl status snapd.service

snap download lxd --channel 4.0/stable

sudo snap ack lxd\_22753.assert

sudo snap install lxd\_22753.snap --channel 4.0/stable

sudo snap install lxd\_22753.snap

core20

snap download jq

sudo snap ack jq\_6.assert

sudo snap install jq\_6.snap

snap ack juju\_17665.assert

snap install juju\_17665.snap --classic --channel=2.8/stable

snap install juju\_17665.snap --classic

snap install juju\_17665.snap

pip3 download

pip3 install --no-index \*

two times

docker save $name -i $packetname $name

docker load -i $packetname

docker.io/wurstmeister/zookeeper:latest

docker.io/library/mongo:latest

docker.io/library/mysql:5

docker.io/wurstmeister/kafka:2.11-1.0.2

docker.io/prom/prometheus:v2.4.3

docker.io/google/cadvisor:latest

docker.io/grafana/grafana:latest

docker.io/library/mariadb:10

docker.io/opensourcemano/mon:10

docker.io/opensourcemano/pol:10

docker.io/opensourcemano/nbi:10

docker.io/opensourcemano/keystone:10

docker.io/opensourcemano/ro:10

docker.io/opensourcemano/lcm:10

docker.io/opensourcemano/osmclient:10