CAN NETWORK FUNCTION INSTALLATION - MOP (METHOD OF PROCEDURE) FOR AWS WITH EBS

Cognitive Assistant for Networks (CAN) Release 5.5

JULY 6, 2021

AVANSEUS TECHNOLOGY PVT. LTD.

REVISION HISTORY

Version	Date	Change description	Created by	Updated by	Reviewed by
V 1.0	July, 2021	Initial Release	Hemanth/Umesh	Sandeep Singh	Chiranjib

avanseus

Table of Contents

1. Prerequisites

This section contains all the mandatory prerequisites and cluster requirements:

- Any Linux Based System preferably UBUNTU v20.04 or lower or RHEL v8.3 or lower.
- Docker v20.10.2 or higher Installed.
- Kubernetes v1.20.0 or higher Installed.
- Helm should be Installed.
- Domain Name for hosting the application. (Optional)
- SSL Certificate Files for the registered Domain. (Or use Self signed certificate)

Note that Avanseus CAN application uses MongoDB as its persistent storage. Importing the default schema into the mongo database is one of the prerequisites which will be discussed in the later section.

2. Istio Installation and Configuration

Avanseus Cognitive Assistant for Networks (CAN) is integrated with Istio service mesh to provide several features like Authorization Policy, Traffic Management, Peer Authentication Policy, Envoy sidecar Proxy Injection, Monitoring etc., Istio needs to be installed before the installation of CAN related HELM charts.

1. Istio v1.8 or higher.
2. Istio add-ons (Grafana, Kiali, Prometheus) have to be enabled.

2.1. Istio Installation

Note: If you are using ec2 instance provisioned from AMI images then Istio is already available in the Master Node you can skip 1 and 2.

Follow the below steps to install the Istio:

1. You can use the link as a reference to install the Istio:

   https://istio.io/latest/docs/setup/getting-started/

2. Download the Istio installation files using the below command:

```
$curl -L https://istio.io/downloadIstio | sh -   //This will download the latest version.
```

3. Move to the Istio package directory.

```
$cd istio-1.9.3
$export PATH=$PWD/bin:$PATH
```

4. Install Istio:

```
$istioctl install --set profile=demo -y
```

5. The installation of Istio is complete. To check all the pods are up, use the below commands:

```
$kubectl get all -n istio-system
```

```
[ec2-user@ip-172-31-28-192 ~]$ kubectl get all -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
pod/grafana-7bdcf77687-gnsvq            1/1     Running   1          24h
pod/istio-egressgateway-98df895bc-mxs74 1/1     Running   1          24h
pod/istio-ingressgateway-85bc5679c4-4hnfn 1/1   Running   1          24h
pod/istiod-76544c444-lj6rv              1/1     Running   1          24h
pod/jaeger-5c7c5c8d87-gqdtc             1/1     Running   1          24h
pod/kiali-6b66bfc5-nql22                1/1     Running   1          24h
pod/prometheus-f5f544b59-qp2bz          2/2     Running   2          24h

NAME                           TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
service/grafana                ClusterIP      10.96.29.20      <none>        3000/TCP
    24h
service/istio-egressgateway    ClusterIP      10.105.217.145   <none>        80/TCP,443/TCP,15443/TCP
    24h
service/istio-ingressgateway   LoadBalancer   10.106.37.170    <pending>     15021:30192/TCP,80:31403/TCP,443:30604/TCP,31400:32071/TCP,15443:31450/TC
P   24h
service/istiod                 ClusterIP      10.111.233.144   <none>        15010/TCP,15012/TCP,443/TCP,15014/TCP
    24h
service/jaeger-collector       ClusterIP      10.109.58.225    <none>        14268/TCP,14250/TCP
    24h
service/kiali                  ClusterIP      10.104.247.130   <none>        20001/TCP,9090/TCP
    24h
service/prometheus             ClusterIP      10.96.228.163    <none>        9090/TCP
    24h
service/tracing                ClusterIP      10.110.251.75    <none>        80/TCP
    24h
service/zipkin                 ClusterIP      10.96.80.101     <none>        9411/TCP
    24h
```

2.2. Enabling Istio Add-Ons

To install the Istio add-ons, execute the two commands:

```
$kubectl apply -f samples/addons
$kubectl rollout status deployment/kiali -n istio-system
```

Note: If there are errors while trying to install the add-ons, try to run the command again. There may be some timing issues which will be resolved when you run the command again.

3. Pre-Installation Steps

Follow the below steps before you install the HELM charts related to CAN application:

1. Create a namespace.

```
$ kubectl create ns avanseus-workspace
```

2. Enable Istio injection on the desired namespace i.e., avanseus-workspace

```
$ kubectl label namespace avanseus-workspace
istio-injection=enabled
$ kubectl get namespace -L istio-injection
```

(Use this command to check if Istio-injection is enabled or not in the avanseus-workspace).

```
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$ kubectl get namespace -L istio-injection
NAME                 STATUS   AGE    ISTIO-INJECTION
avanseus-workspace   Active   25h    enabled
default              Active   2d3h
istio-system         Active   25h    disabled
kube-node-lease      Active   2d3h
kube-public          Active   2d3h
kube-system          Active   2d3h
[ec2-user@ip-172-31-28-192 ~]$
```

3. Installing Metric Server in the cluster.

   Go to ***kubernetes_resources/Gateway_Metric_HPA/*** folder. Use the file named **metric_server.yaml** and execute the below command:

$ kubectl apply -f metric_server.yaml

Alternatively you can use:
$ kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components. yaml

Use the below command to check, if metric server is running:

$kubectl get all -n kube-system|grep metric

```
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$ kubectl get all -n kube-system|grep metric
pod/metrics-server-c58854fdc-8btq6                                  1/1     Running   1          25h
service/metrics-server   ClusterIP   10.101.43.136   <none>        443/TCP                  25h
deployment.apps/metrics-server            1/1     1            1           25h
replicaset.apps/metrics-server-c58854fdc             1         1         1       25h
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
```

4. Creating Secrets

   This will be used internally by HELM charts to pull Avanseus specific Docker images from the Avanseus public Docker repository.

   Refer **Kubernetes Secret for Docker Registry** for Creating Secrets in the Kubernetes Cluster.

4. Creating Persistent Volume

Four persistence volumes have to be created for the whole application deployment.

SL No	Application name	PVC name
1	Mongodb(database)	mongopvc
2	CAN-logs	canlogpvc
3	CAN-prediction related Files	candatapvc
4	LDAP	ldappvc

In this document we are using Amazon Elastic Block Storage(EBS) for creating persistent volumes. If you are interested in dynamic provisioning, then please refer to the documentation with NFS server.

4.1. Elastic Block Storage (EBS)

Create three different EBS volumes of appropriate size from Amazon and get the volume id of each of the Volumes.

Go to **kubernetes_resources/Helm_Charts/AWS_HELM/Persistent_volume/** folder. You can find three files for creating a persistence volume for mongodb, LDAP and CAN modules named as mongo_pv.yaml, ldap_pv.yaml and can_pv.yaml. Edit these files and modify the new volume id for each of the persistent storage.

Example:

For mongodb persistence storage, In the file named as "**mongo_pv.yaml**" change "volumeID" to new volume id i.e., vol-05ee5fdc09. Repeat the same for the other file.

Execute the below commands to create the persistent volume claim. Note that persistent volume will be automatically created by the storage class:

```
$ kubectl apply -f mongo_pv.yaml -n avanseus-workspace
$ kubectl apply -f can_pv.yaml -n avanseus-workspace
$ kubectl apply -f ldap_pv.yaml -n avanseus-workspace
```

Verify whether all the 4 PVC are created using:

```
$ kubectl get pvc -n avanseus-workspace
```

```
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$ kubectl get pvc -n avanseus-workspace
NAME         STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS          AGE
candatapvc   Bound    pvc-26391959-d7c7-4fbf-960d-41661f094795   4Gi        RWX            managed-nfs-storage   21h
canlogpvc    Bound    pvc-20342d13-d0c9-4ee5-874d-c2993c6850aa   5Gi        RWX            managed-nfs-storage   21h
ldappvc      Bound    pvc-5fcd0821-0038-4d99-93a4-e7f3b8a78429   5Gi        RWX            managed-nfs-storage   21h
mongopvc     Bound    pvc-294856b3-ac07-4e4c-be0c-e7ed7e23164a   50Gi       RWX            managed-nfs-storage   21h
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
```

5. Installing Database and Importing Default Schema

Avanseus CAN application uses Mongo database as its backend database.

5.1. Prerequisites

Below mentioned files are required to set up the Mongo chart along with TLS. Please refer the document **Securing MongoDB with Let's Encrypt SSL certificate** or **Securing MongoDB using a self-signed SSL certificate** to understand the creation of certificates or use client provided certificates to generate the below files.

- mongo.pem
- ca.pem (this file will not be generated if you are using a self-signed certificate so you can ignore the changes to be made in this file).

Execute the below command to generate the mongo.pkcs12 file using the mongo.pem file:

```
$openssl pkcs12 -export -out mongo.pkcs12 -in /etc/ssl/mongodb.pem
```

This mongo.pkcs12 file will be later used by java driver while communicating with the mongo database.

Once you execute the above command, you will be asked to enter a password. Please note the password. In the later stages, the same password (after encrypting with jasypt) will be used in the **config.properties** file of the CAN module.

Replace the generated mongo.pkcs12 file with the old mongo.pkcs12 file present under the folder **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/ConfigMap_Files/can-config/.** In the later stages this file will be used while creating a configuration map of the can module.

5.2. Mongo Database Config map and Secret Creation

Go to **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/MongoTLS_Configs.** You will find two files named **avanseus-mongo-config.yaml** and **avanseus-mongo-secrets.yaml.** Apply the mongo-config file without making any changes but update mongo-secrets file with the certificate contents as described below:

1. Apply **avanseus-mongo-config.yaml** file using the below command:

```
$kubectl apply -f avanseus-mongo-config.yaml -n avanseus-workspace
```

2. **avanseus-mongo-secrets.yaml**

   The content of earlier generated certificates (i.e., ca.pem file and mongo.pem file) has to be used now in order to generate the secret required for the mongo chart. File by the name "avanseus-mongo-secrets.yaml" acts as a template for secret creation as it contains the placeholders for certificate contents.

   So issue the below by specifying correct path for <**PATH_TO_mongo.pem**> and <**PATH_TO_ca.pem**>.

```
$ sed "s/AVANSEUS_KEY_CERT/`cat <PATH_TO_mongo.pem>|base64 -w0`/g" avanseus-mongo-secrets.yaml | \
sed "s/AVANSEUS_CA_CERT/`cat <PATH_TO_ca.pem>|base64 -w0`/g" | \
kubectl apply -f - -n avanseus-workspace
```

5.3. Mongo Chart Installation and Importing Schema

After creating configmap and secret, go to the folder **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/** of the git repository to find all the charts.

Go to the directory where all the charts are present and execute the below command:

```
$ helm install mongo avanseus-mongodb-chart/ -n avanseus-workspace
```

After the above step, mongo database will be successfully installed.

Create the user and load the database with master/mandatory tables by creating an appropriate schema. This will be achieved with the help of mongo client from the local system either from mongo CLI or with the help of softwares like studio3T, NoSQLBooster, MongoDB Compass etc. If you choose to use mongo CLI to create the user and to restore the schema, then proceed with the below steps:

1. Connect to mongo database using root user and ssl certificates

```
$ mongo --tls --tlsCAFile /etc/ssl/ca.pem --tlsCertificateKeyFile /etc/ssl/mongo.pem –host
<ip_address_of_any_nodes> --port 30001
--authenticationDatabase admin -u admin -p 'Avanseus$0'
--tlsAllowInvalidHostnames
```

2. Create schema and add a new user

```
> use <schema_name>
> db.createUser({user: "<user_name>", pwd: "<user_password>", roles: ["userAdmin",
"dbAdmin", "readWrite"]});
```

3. Restore the dump using the below command:

```
$ mongorestore --ssl --sslCAFile /etc/ssl/ca.pem --sslPEMKeyFile /etc/ssl/mongo.pem --host
<ip_address_of_any_nodes> --port 30001 -u <user_name> -p '<user_password>' --
authenticationDatabase
<schema_name> <dump_path>
```

   - <ip_address_of_any_nodes>: ip address of the master of worker node as to be used.
   - <schema_name>: name of your choice.
   - <user_name>: username of your choice.
   - <user_password>: strong password of your choice.

- <dump_path>: path to the dump from where schema should be restored.

After the successful restoration of schema, go to the folder **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/ConfigMap_Files/can-config/** and open **config.properties** file using the vi editor.

Edit the list of the parameters with correct values:

1. **avanseus.mongodb.admin.password**: <<VALUE1>>
2. **avanseus.mongodb.username**: username of the newly created user.
3. **avanseus.mongodb.dbName**: schema name of the restored database.
4. **avanseus.mongodb.password**: <<VALUE2>>
5. **avanseus.ldap.password**: <<VALUE3>>
6. **avanseus.mongodb.keystore.password**: <<VALUE4>>

Before updating any password field in the config values, encrypt the passwords using jasypt encryptor as described in JASYPT for Storing Passwords in Encrypted format in Database (Encryption and Decryption of Passwords).

Let's assume that the key obtained after encrypting our mongo admin password as VALUE1, key obtained after encrypting password of newly created user as VALUE2, key obtained after encrypting ldap password as VALUE3 and the key obtained after encrypting mongodb keystore password as VALUE4 respectively.

6. Creating Configmaps

You can find all the files related to creating config maps in the folder **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/ConfigMap_Files/** and execute the below commands to create the configmap as follows:

```
$ kubectl create configmap controllerconfig --from-file=controller-config/ --namespace=avanseus-workspace
$ kubectl create configmap canconfig --from-file=can-config/ --namespace=avanseus-workspace
$ kubectl get configmap -n avanseus-workspace
```

```
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$ kubectl get configmap -n avanseus-workspace
NAME                    DATA   AGE
avanseus-mongo-config   1      24h
canconfig               3      166m
controllerconfig        1      18h
istio-ca-root-cert      1      25h
kube-root-ca.crt        1      25h
workerconfig            1      18h
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
[ec2-user@ip-172-31-28-192 ~]$
```

Note:

- If you have purchased a domain name (ex: https://can.avanseus.com) then please update domain details in the config map using kubectl edit configmap command.

```
$kubectl edit configmap canconfig -n avanseus-workspace
```

- Keys such as **avanseus.app.cas.domain** and **avanseus.app.can.domain** have to be updated as “can.avanseus.com” or whichever domain name you have procured.

7. Installing Helm Charts

There are 6 helm charts to be installed while deploying the CAN application. All the required helm charts are present under **kubernetes_resources/Helm_Charts/NFS_STORAGE_HELM/NFS_STORAGE_HELM/** folder.

The sequences in which the charts have to be installed are as follows:

1. avanseus-mongo-chart
2. avanseus-ldapchart-chart
3. avanseus-pyvbi-chart
4. avanseus-workerapp-chart
5. avanseus-controllerapp-chart
6. avanseus-canmaster-chart

IMPORTANT: It is the responsibility of the developer to change the docker image names inside **values.yaml** of all the helm charts. By default, these charts will install CAN5.5 docker images.

The Mongo chart is already installed in previous section. Proceed to install the remaining charts by executing the below commands in sequence:

```
helm install ldap avanseus-ldap-chart/ -n avanseus-workspace
helm install vbi avanseus-pyvbi-chart/ -n avanseus-workspace
helm install controller avanseus-controllerapp-chart/ -n avanseus-workspace
helm install worker avanseus-workerapp-chart/ -n avanseus-workspace
helm install can avanseus-canmaster-chart/ -n avanseus-workspace
```

Verify if all the pods have been deployed successfully using the below command.

```
$kubectl get pods -n avanseus-workspace
```

```
[ec2-user@ip-172-31-25-155 AWS_HELM]$
[ec2-user@ip-172-31-25-155 AWS_HELM]$
[ec2-user@ip-172-31-25-155 AWS_HELM]$
[ec2-user@ip-172-31-25-155 AWS_HELM]$ kubectl get pods -n avanseus-workspace
NAME                          READY   STATUS    RESTARTS   AGE
can-7d6f67d5d9-f7l99          3/3     Running   0          6m52s
controller-7b98c6f5b4-5kbzt   2/2     Running   20         9d
ldap-79f55b4665-4rwk5         2/2     Running   26         14d
memcached-6cccdf8bc8-4gvt5    2/2     Running   0          6m52s
mongo-0                       2/2     Running   0          101m
vbi-654f669c5c-nm7tw          2/2     Running   28         15d
worker-7df997f554-jwq4n       2/2     Running   18         9d
worker-7df997f554-qsz4c       2/2     Running   20         9d
[ec2-user@ip-172-31-25-155 AWS_HELM]$
[ec2-user@ip-172-31-25-155 AWS_HELM]$
```

8. Routing Traffic

There are two ways to route the traffic.

- Using Web Server based configuration. (Preferred option)
- Using Load balancer provided by cloud providers.

8.1. Using Web Server based Configuration

Installing and Configuring web server:

1. Install the NGINX web server on the master node (On RHEL server).

```
$yum install -y nginx
$sudo systemctl start nginx
```

nginx.conf file is present in the folder **kubernetes_resources/Nginx_Configuration_files** of the GIT repository for the web server configuration related changes.

2. Go to the /etc/nginx folder.

   Rename the existing nginx.conf file to nginx.conf_bkp and add the nginx.conf present in **kubernetes_resources/Nginx_Configuration_files** folder to **/etc/nginx/ folder**.

3. Few changes have to be made to the nginx.conf file.

   Changes to be made are as follows:

   a. DOMAIN_NAME: this has to be changed in the conf file as applicable. DOMAIN_NAME can be different for different servers.

```
server {
    server_name  <DOMAIN_NAME>;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
```

   For Example: You can use avanseus.telcocloud.com as server name as shown in the below image.

```
server {
    listen       80;
    server_name avanseus.telcocloud.com;
    location / {
        proxy_pass http://allbackend;
        proxy_http_version 1.1;
    }
```

   b. istio_http_port_number (mapping of port no 80 has to be added in the nginx.conf file)

   c. Execute the below command to get the istio_http_port_number

```
$kubectl get svc istio-ingressgateway -n istio-system
```

```
[ec2-user@ip-172-31-25-155 ~]$
[ec2-user@ip-172-31-25-155 ~]$
[ec2-user@ip-172-31-25-155 ~]$ kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S
)                                                                AGE
istio-ingressgateway   LoadBalancer   10.107.249.134   a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com   15021:
30676/TCP,80:31521/TCP,443:31846/TCP,15012:32466/TCP,15443:31628/TCP   174d
[ec2-user@ip-172-31-25-155 ~]$
[ec2-user@ip-172-31-25-155 ~]$
```

   In the above snapshot you can observe that port 80 of ingress gateway is mapped against 31521. Hence port 31521 has to be updated in the **nginx.conf** file as shown below.

```
    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

upstream allbackend {
    server localhost:31521; #istio testing
}

server {
    server_name  avanseuscanvm.com;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
```

   d. path_to_the_certificate_file: path of certificate file has to be added in the nginx.conf file.

e. path_to_the_key_file: path of key file has to be added in the nginx.conf file as seen in the below image.

```
    listen 443 ssl http2; # managed by Certbot
#    ssl_protocols TLSv1.3;
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    include /etc/ssl/ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/ssl/certs/dhparam.pem; # managed by Certbot
```

After making the respective changes, restart the nginx server with the below command.

$sudo systemctl restart nginx

After restarting the nginx server, external traffic from the nginx webserver will be forwarded to the Istio ingress controller which then routes the traffic across different services within the cluster. Routing rules are specified in the **webserver_gateway.yaml** file under **kubernetes_resources/Gateway_Metric_HPA** folder.

$kubectl apply -f webserver_gateway.yaml -n istio-system

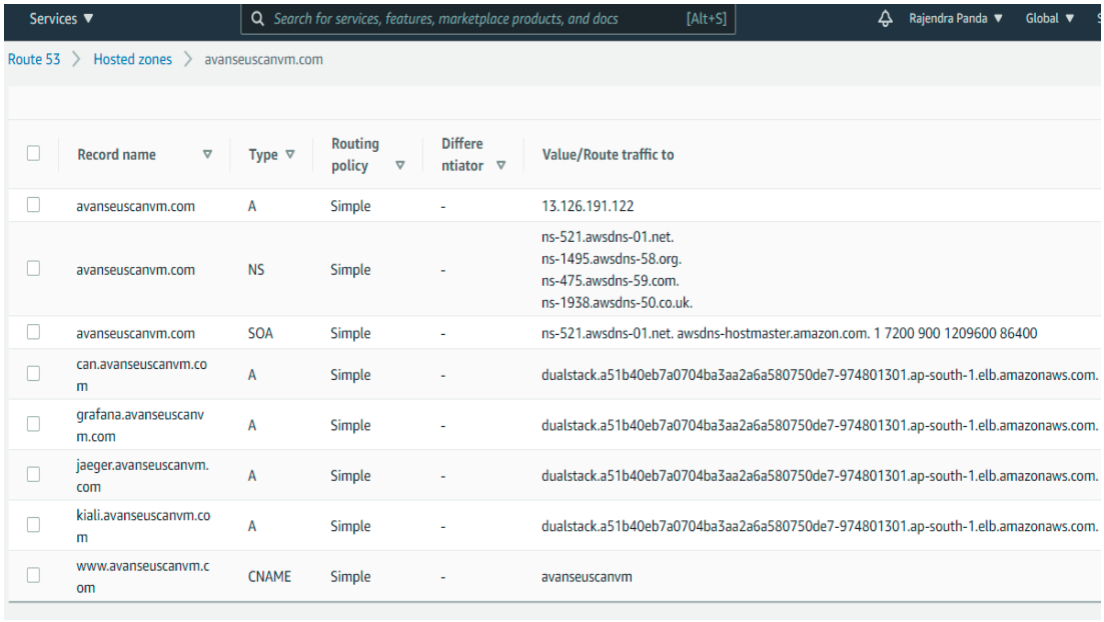
8.2. Using External Load Balancers provided by Cloud Providers

We need an external load balancer provisioned from the AWS console. For provisioning any AWS services, we need to have proper IAM roles assigned against the master and worker ec2 instances as mentioned in section 3 of **Kubernetes Cluster Configuration On AWS** for CAN_5.5.

If IAM rules are configured correctly, then the Istio-ingress controller will provision an external load balancer - ELB under the external-IP section as shown below:

```
[ec2-user@ip-172-31-25-155 ~]$
[ec2-user@ip-172-31-25-155 ~]$ kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP                                                                    PORT(S)
                AGE
istio-ingressgateway   LoadBalancer   10.107.249.134   a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com   15021:30676/TCP,80:31521/TCP,44
443:31628/TCP   152d
[ec2-user@ip-172-31-25-155 ~]$
```

Also, if we have purchased a domain name then we can (optionally) create multiple subdomains using route 53 “A type” record (as shown in the below diagram) and route the external traffic to the intended application within the cluster by configuring **ingress gateway rules** (gateway.yaml).



Record name	Type	Routing policy	Differe ntiator	Value/Route traffic to
avanseuscanvm.com	A	Simple	-	13.126.191.122
avanseuscanvm.com	NS	Simple	-	ns-521.awsdns-01.net. ns-1495.awsdns-58.org. ns-475.awsdns-59.com. ns-1938.awsdns-50.co.uk.
avanseuscanvm.com	SOA	Simple	-	ns-521.awsdns-01.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
can.avanseuscanvm.com	A	Simple	-	dualstack.a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com.
grafana.avanseuscanvm.com	A	Simple	-	dualstack.a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com.
jaeger.avanseuscanvm.com	A	Simple	-	dualstack.a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com.
kiali.avanseuscanvm.com	A	Simple	-	dualstack.a51b40eb7a0704ba3aa2a6a580750de7-974801301.ap-south-1.elb.amazonaws.com.
www.avanseuscanvm.com	CNAME	Simple	-	avanseuscanvm

You can notice one thing that we have provisioned an external load balancer and created aliases for each of subdomains (Ex: kiali.avanseuscan.com, can.avanseuscan.com etc.,) but the traffic is still HTTP not HTTPS. Now, the question arises regarding where to place our TLS certificates so that all the traffic redirections happen in a secured HTTPS layer. This can be achieved by installing Certificate Manager (**Cert-Manager**) in the kubernetes cluster.

Certificate Manager:

Cert-Manager is a tool that runs inside your Kubernetes cluster and is used to request globally valid TLS certificates from Let's Encrypt, HashiCorp Vault, Venafi, or can even issue a self-signed certificate. By default, we have set our cluster issuer to Let's encrypt inside the cluster-issuer.yaml file.

Create namespace:

```
$ kubectl create namespace cert-manager
```

Installing cert manager:

```
$kubectl apply -f
https://github.com/jetstack/cert-manager/releases/download/v0.13.1/cert-manager.yaml
```

Before proceeding to the next step, make necessary changes in the below files with the relevant information. These files are present in our git repository under the folder kubernetes_resources/Cert-Manager:

- cluster-issuer.yaml : Remember to change the email-address in the file.
- certificate.yaml : Remember to changes the DNS names in the file.

Cluster issuer:

```
$kubectl apply -f cluster-issuer.yaml -n cert-manager
```

Requesting Certificate:

```
$kubectl apply -f certificate.yaml -n cert-manager l
```

Now, edit the "**hosts**" name in **gateway.yaml** file under the folder **kubernetes_resources/Gateway_Metric_HPA**/ by providing the valid domain name which you have procured.

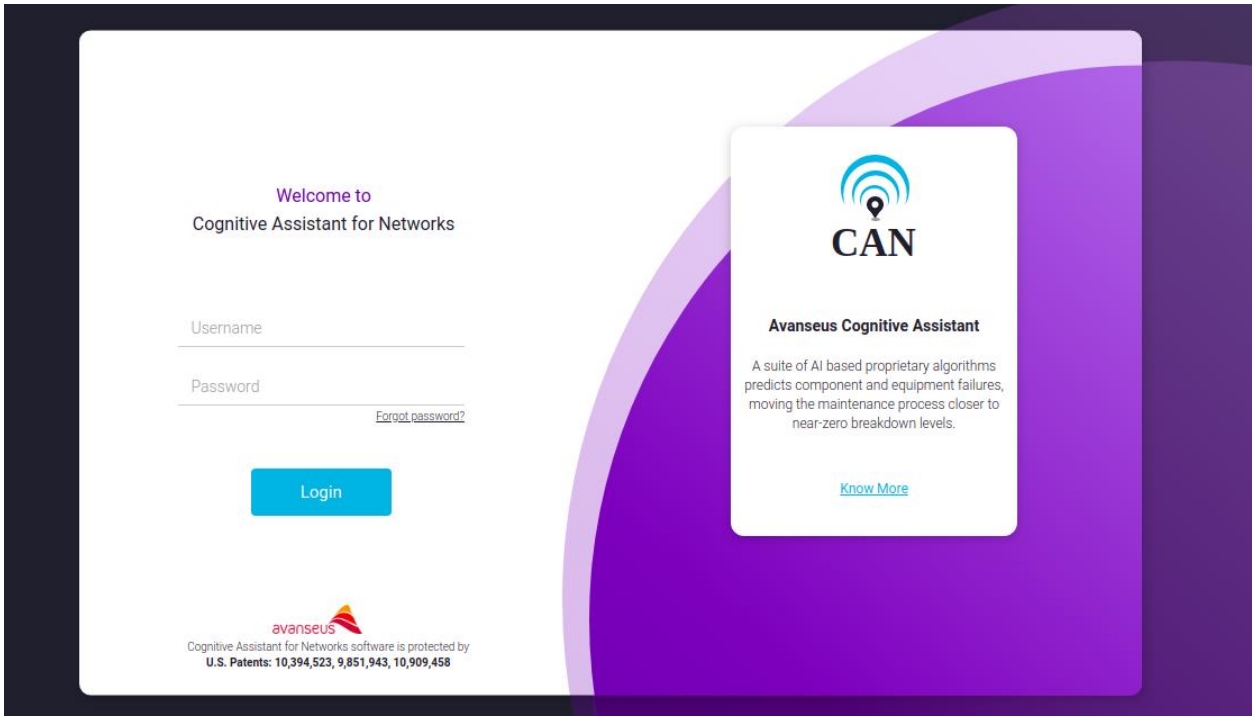
Apply the changes using the below command:

```
Istio VirtualService and Gateway
$kubectl apply -f gateway.yaml -n istio-system
```

9. Accessing the Application and Monitoring Tools from the Browser

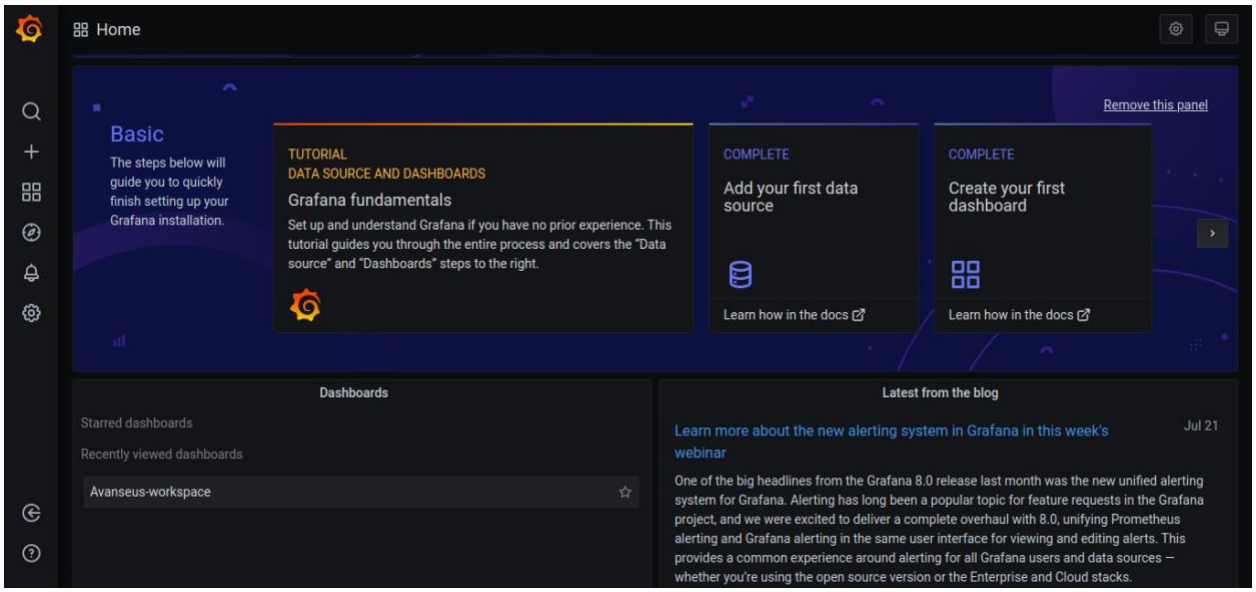
CAN Application: http://<domain_name>/CAN/

Ex: <domain_name> = 192.168.29.100 or can.avanseus.com



Grafana Dashboard: https://<domain_name>/grafana

Ex: <domain_name> = 192.168.29.100 or can.avanseus.com



Kiali Dashboard: https://<domain_name>/kiali

Ex: <domain_name> = 192.168.29.100 or can.avanseus.com