IDBB EKS Sand-Box Deployment Procedure

Prerequisites

- Make sure the AWS user should have all the permission and roles added to create the EKS cluster.
  Amazon EKS cluster IAM role - Amazon EKS
- AWS credentials in `~/.aws/` folder as given here.
- Copy of `~/.kube/config` file with another name. *(IMPORTANT. As in this process,* your existing `~/.kube/config` file will be overridden).
- eksctl utility.
- kubectl utility installed.
- Key `.pem` file from AWS console in `~/.ssh/` folder. (Generate a new one if you do not have this key file).
- aws-iam-authenticator installed.
- istioctl utility.
- Ansible : version > 2.12.4
- rke : version: 1.3.10
- helm client version 3.8.2 and add the below repo's as well to local

  Note: Ignore if you have already configured prerequisites.

Deployment Repo's to install IDBB using EKS cluster

1. git clone -b tf-develop-B3 https://github.com/tf-govstack/k8s-infra/tree/tf-develop-B3/mosip/aws/istio - Connect your Github account : contains scripts to install and configure Kubernetes cluster with istio installation.
2. git clone -b tf-develop-B3 GitHub - tf-govstack/mosip-infra at tf-develop-B3 : contains deployment scripts to run charts in a defined sequence.
3. GitHub - tf-govstack/mosip-config: This repository contains MOSIP configuration templates : contains all the configuration files required by the Mosip modules.
4. Use below mosip-helm - GitHub - tf-govstack/mosip-helm to install idbb-mosip services, in case anything you need to edit helm-charts for govstack deployment then use this helm-charts - GitHub - tf-govstack/tf-govstack-helm at 1.2.0.1-GB3
5. Helm repos you need to add before proceeding with deployment.
   - ```
     helm repo add bitnami https://charts.bitnami.com/bitnami
     helm repo add mosip https://mosip.github.io/mosip-helm
     helm repo add tf-govstack https://tf-govstack.github.io/tf-gov
     helm repo add kafka-ui https://provectus.github.io/kafka-ui-charts
     ```

AWS EKS Cluster Setup for IDBB

IDBB K8's Cluster Global configmap, Ingress, and Storage Class setup

- **Global configmap**: Global configmap contains the list of necessary details to be used throughout the namespaces of the cluster for common details.
  - Use this branch https://github.com/tf-govstack/k8s-infra/blob/tf-develop-B3/mosip/global_configmap.yaml.sample
  - Copy `global_configmap.yaml.sample` to `global_configmap.yaml.`
  - Update the domain names in `global_configmap.yaml` and run.
    - `kubectl apply -f global_configmap.yaml`
     ```

Storage Class setup

GP2

- Use GP2 as the default storage for the EKS cluster and do the required configurations to setup GP2 is the default storage class for your EKS cluster.
- You can go with the default gp2 EBS volume attached to all the nodes . And to mount gp2 SC PVCs you need to install `aws-ebs-csi-driver-on-aws-eks-for-persistent-storage` follow this document here .
- After installing make sure the storage class is the default one. To check run this command `kubectl get sc` it will show all the storage classes and if your storage class is not set as the default one go with this document to make it the default one.

Ingress and load balancer (LB)

Ingress is not installed by default on EKS. We use Istio ingress gateway controller to allow traffic in the cluster. Two channels are created - public and internal. See architecture.

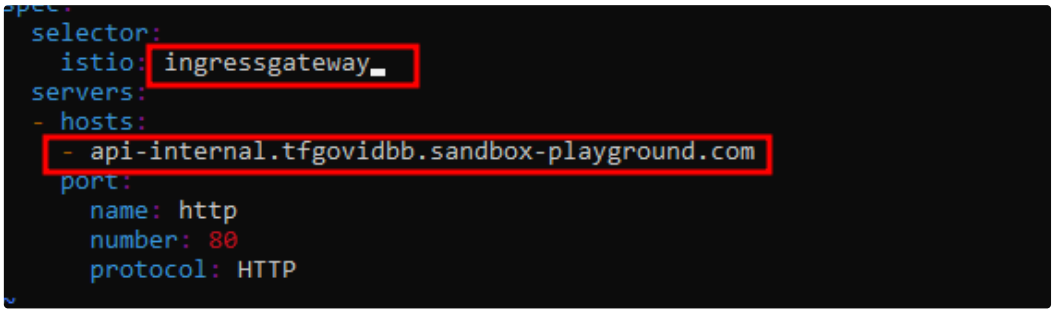
- Install Istioctl as given here Version : 1.15.0
- Install ingress https://github.com/tf-govstack/k8s-infra/tree/tf-develop-B3/mosip/aws/istio - Connect your Github account

```
cd istio
./install.sh

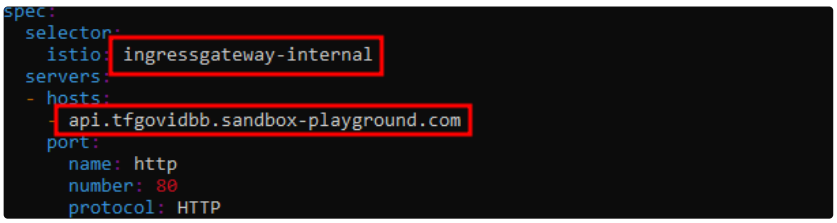
```

Note: In case you need all services/api's to be accessible without wireguard then please edit istio gateways once istio deployed. You can refer below screenshots.

- public gateway (redirecting to the internal host to access all services)



- Internal gateway (redirecting to external host it may use or may not)



- Make sure **you want to access all services publicly** at the istio-level. Then, once this above step is done you need to configure all idbb/mosip internal and external services `gateway` and `VirtualServices` accordingly. Please find the screenshot below for the same.
- **At gateway level** (for each services need to check and edit this)

```
  uid: 6ed54efb-81de-4390-a2d1-7363b08888a8
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - admin.tfgovidbb.sandbox-playground.com
    port:
      name: http
      number: 80
      protocol: HTTP
```

pointing to public gateway of istio

- **At VirtualService level** (for each services need to check and edit this)

```
  resourceVersion: "29453457"
  uid: 56f3d329-fccc-4801-9355-44fd7a95ab3e
spec:
  gateways:
  - istio-system/public
  - istio-system/internal
  hosts:
  - '*'
```

- If you don't want all services publicly available then go with `port forwarding` for the respective services you want access. *But not sure this method may affect to access all the api's.*

Load Balancers

The above istio install will spin off a load balancer on AWS. You may view them on the AWS console. These may be also seen with this command and make sure `CLUSTER-IP` and `EXTERNAL-IP` should be present. Then only you are able to map domains to load balancer.

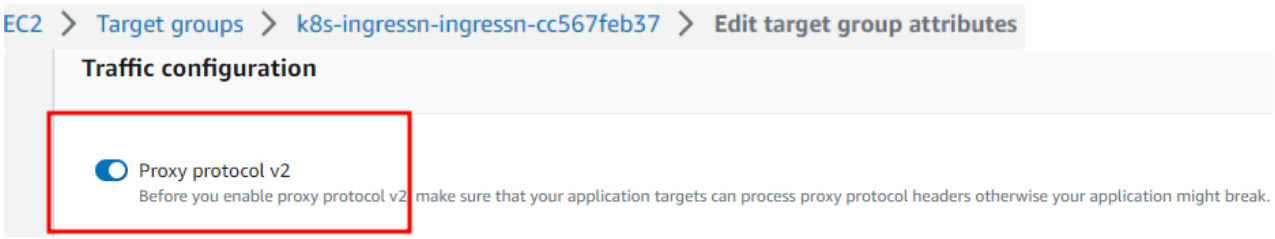
```
kubectl  get svc -n istio-system
```

```
NAME                           TYPE           CLUSTER-IP      EXTERNAL-IP                                                                    PORT(S)
                                                                                                                                              AGE
istio-ingressgateway           LoadBalancer   172.20.57.230   a0c6d2c42992e4b0397266618c11e01a-f88c2565c0833ce6.elb.eu-central-1.amazonaws.com   15021:32381/TCP,80:305
07/TCP,443:32003/TCP,61616:31126/TCP,5432:32068/TCP,9000:32246/TCP   33d
istio-ingressgateway-internal  LoadBalancer   172.20.119.56   a9f6f3b37bdfd404c98b403060a1ca49-de4e5552f94d4ce2.elb.eu-central-1.amazonaws.com   15021:32409/TCP,80:300
91/TCP,443:31499/TCP,61616:30063/TCP,5432:31985/TCP,9000:31349/TCP   33d
istiod                         ClusterIP      172.20.0.42     <none>                                                                         15010/TCP,15012/TCP,44
3/TCP,15014/TCP                                                                                                                              33d
pramod@TECHNO-244:~$
```

- Make sure in EKS-IDBB deployment use only one `ingress-gateway-public` LB for both public and private routing and keep `ingress-gateway-internal` as optional.
- TLS termination is supposed to be on LB. So all our traffic coming to the ingress controller shall be HTTP.
- Obtain AWS TLS certificate as given here. Ignore if you have already created one.
- Add the certificates and 443 access to the LB listener.
  - Update listener TCP->443 to **TLS->443** and point to the certificate of domain name that belongs to your cluster. Same as you have done for rancher LB.
  - Forward TLS->443 listner traffic to target group that corresponds to listner on port 80 of respective Loadbalancers. This is because after TLS termination the protocol is HTTP so we must point LB to HTTP port of ingress controller.
- Update health check ports of LB target groups to node port corresponding to port 15021. You can see the node ports with

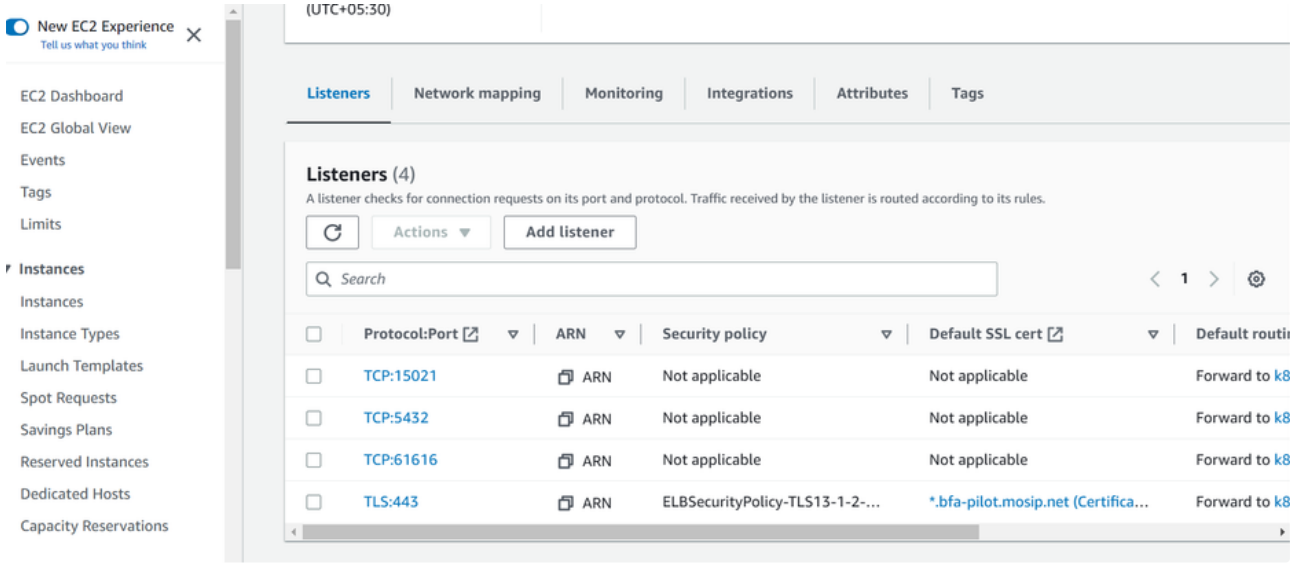
```
kubectl -n istio-system get svc
```

- Enable Proxy Protocol v2 on all target groups.



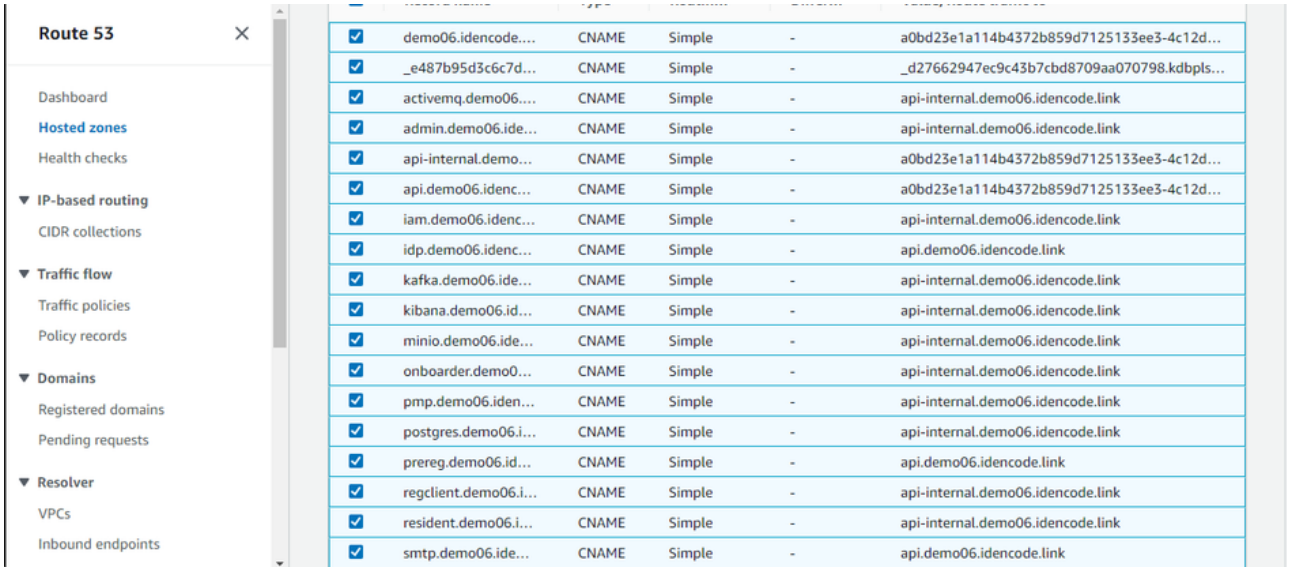
- Make sure all subnets are included in Availability Zones for the LB. Description --> Availability Zones --> Edit Subnets

- Make sure to delete the listenrs for port 80 and 15021 from each of the loadbalancers as we restrict unsecured port 80 access over HTTP. Find the below image for the same changes.



DNS Requirements for IDBB-Govstack cluster:

- Point all your domain names to EXTERNAL LoadBalancers DNS/IP.
- On AWS this may be done on Route 53 console.
- After the Go-live decision enables https://github.com/tf-govstack/k8s-infra/blob/tf-develop-B3/docs/public-access.md The same steps you have done on istio level above. You can refer this too.
- Find the below image for routing on Route53.
- The below image shows an example placeholder for hostnames, the actual name itself varies from organization to organization.
- Only proceed to DNS mapping after the ingressgateways are installed and the nginx reverse proxy is setup. You can refer sand-box route53 for now.



Check Overall if nginx and istio wiring is set correctly

- Install `httpbin` for testing the wiring as per https://github.com/tf-govstack/k8s-infra/tree/tf-develop-B3/utils/httpbin - Connect your Git hub account

IDBB Govstack External Dependencies setup

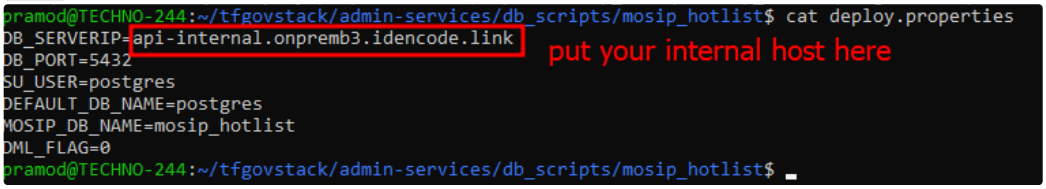
- External Dependencies are a set of external requirements needed for the functioning of MOSIP's core services like DB, object store, hsm etc. Please follow this sequence one by one https://github.com/tf-govstack/mosip-infra/blob/tf-develop-B3/deployment/v3/external/all/install-all.sh.

  Note: Ignore `docker-secrets` not required and `conf-secrets` you will find while installing mosip-internal-modules.
- And while installing **Postgres** make sure that once done with `./postgres-init` installation you need to deploy 3 DB's again by just running DB_scripts from the respective repos to take updated changes for idbb. (no need to delete existing db)
- Here are repos to clone to your local and run db_scripts. Just go inside the db_script and edit `deploy.properties` and run this `./deploy.sh deploy.properties`. It will ask for Postgres password.

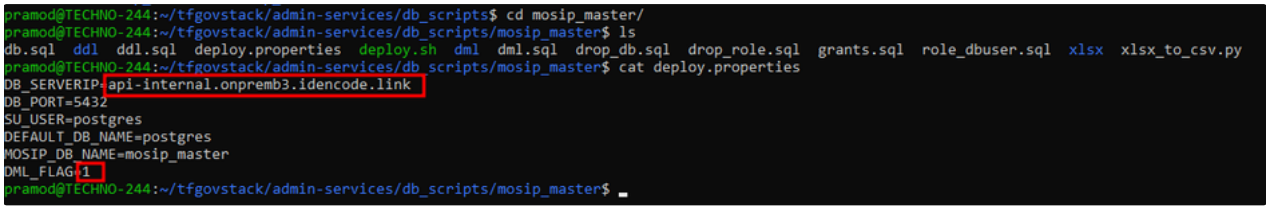
  Note: Take **DB PASSWORD** from running this command in local / or get it from postgres NS secrets.

  `kubectl -n postgres get secrets postgres-postgresql -o json | jq '.data | map_values(@base64d)'`

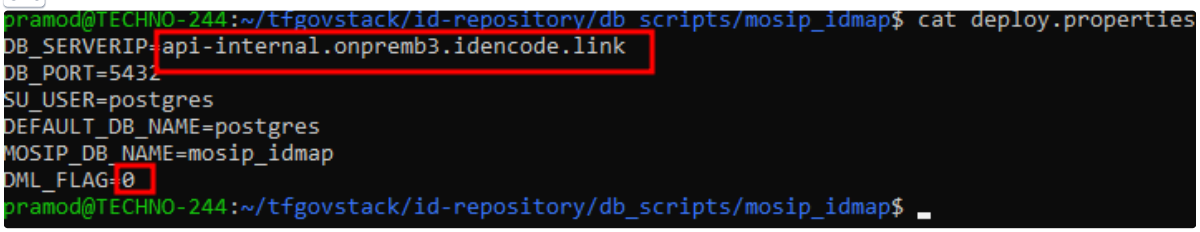
  - mosip_hotlist db - https://github.com/tf-govstack/admin-services/tree/tf-develop/db_scripts/mosip_hotlist - Connect your Github account

  - mosip_master db - https://github.com/tf-govstack/admin-services/tree/tf-develop/db_scripts/mosip_master - Connect your Github account

  - mosip_idmap db - https://github.com/tf-govstack/id-repository/tree/tf-develop/db_scripts/mosip_idmap - Connect your Github account

- Once DB deployment is done GO TO EKS CLUSTER-> POSTGRES NS -> STORAGE -> SECRETS -> COPY `db-common-secrets` PASSWORD and Copy in →

  OPEN PG ADMIN (You can connect with any database app)

  click on Login/Group Role --> right click on masteruser --> properties --> definition (paste `db-common-secrets` as a password)

  click on privilages

  Superuser? ENABLE ---> this will Enable multiple options automatically

  click on save

  Note: DO this for all other 2 DB's also or else you may get issues while installing respective DB services.
- And once DB deployment is done you can follow from keycloak installation as it is.
- Create a Google Recaptcha v2 ("I am not a Robot") from the Google Recaptcha admin while installing the Captcha service. Refer: re CAPTCHA

IDBB Govstack Modules Deployment

- Now external dependencies are installed, will continue with IDBB/MOSIP services deployment.
- Please follow this **sequence** to install **mosip-internal services**. https://github.com/tf-govstack/mosip-infra/blob/tf-develop-B3/deployment/v3/mosip/all/install-all.sh (No need to install **prereg** as part of govstack don't need that module)
- And while installing `config-server` please make sure you need to **create** a branch with respect to your env/cluster name in the **mosip-config repository** from this branch (GitHub - tf-govstack/mosip-config at govstack-v1.2.0.1-B3) Repository you can use to create your branch → GitHub - tf-govstack/mosip-config: This repository contains MOSIP configuration templates and then edit values.yaml file in config-server to take your config branch.

```
ramod@TECHNO-244:~/on-prem/mosip-infra/deployment/v3/mosip/config-server$ ls
README.md  copy_cm.sh  copy_secrets.sh  delete.sh  get_encrypt_key.sh  get_keycloak_secrets.sh  install.sh  restart.sh  values.yaml
ramod@TECHNO-244:~/on-prem/mosip-infra/deployment/v3/mosip/config-server$ cat values.yaml
gitRepo:
  uri: https://github.com/tf-govstack/mosip-config
  version: govstack-v1.2.0.1-B3
  ## Folders within the base repo where properties may be found.
  searchFolders: ""
  private: false
  ## User name of user who has access to the private repo. Ignore for public repo
  username: ""
  token: ""
ramod@TECHNO-244:~/on-prem/mosip-infra/deployment/v3/mosip/config-server$
```

- While installing partner onboarding **do it last after regclient module** and make sure all services are up and running or not, then install and provide the minio URL (http://minio.minio:9000) and Bucket name (onboarder). But make sure you need to onboard partners only for mosip-services and **make sure esignet onboarding will take care of once esignet deployment is done** because this is considered as an external module so.
- Make these false in `values.yaml` while running **partner onboarder**.

```
onboarding:
  modules:
    - name: ida
      enabled: true
    - name: print
      enabled: true
    - name: abis
      enabled: true
    - name: resident
      enabled: true
    - name: mobileid
      enabled: true
    - name: digitalcard
      enabled: false
    - name: esignet
      enabled: false
    - name: demo-oidc
      enabled: false
    - name: resident-oidc
      enabled: false
    - name: mimoto-keybinding
      enabled: true
```

Esignet Deployment Procedure

- Clone the repository to your local to deploy Esignet
  - git clone -b tf-develop-infra GitHub - tf-govstack/esignet at tf-develop-infra
  - git clone -b tf-develop-infra GitHub - tf-govstack/esignet-mock-services at tf-develop-infra

CREATING MOSIP_ESIGNET DATABASE:

cd `esignet/db_scripts/mosip_esignet`

chmod +x deploy.sh

nano `deploy.properties` → edit deploy.properties as per your database hostname and then deploy DB.

DB_SERVERIP=api-internal.sandbox.idencode.link ---> edit internal api hostname EX. api-internal.tf1.idencode.link

DML_FLAG=1 ---> make DML_FLAG=1

COPY POSTGRES PASSWORD. EKS CLUSTER -> POSTGRES NS -> STORAGE -> SECRETS -> COPY postgres-postgresql PASSWORD

Run below **command** to deploy DB

`./deploy.sh deploy.properties` OR `bash deploy.sh deploy.properties`

enter DB password multiple times it will be asked

NOTE: VERIFY IN PGADMIN WEATHER MOSIP_ESIGNET DB IS GENERATED OR NOT

GO TO RANCHER -> POSTGRES -> STORAGE -> SECRETS -> COPY db-common-secrets PASSWORD and Copy

OPEN PG ADMIN

click on Login/Group Role --> right click on esignetuser --> properties --> defination (paste db-common-secrets as a password)

click on privilages

Superuser? ENABLE ---> this will Enable multiple options automatically

click on save

DEPLOY REDIS, ESIGNET AND OIDC-UI :

cd esignet/helm/esignet

helm dependency build

helm repo update

cd esignet/helm

./install-all.sh

> NOTE : PARALLELY CHECK WHILE INSTALLING - CONFIG-SERVER WILL CREATE NEW POD, - REDIS NAMESPACE WILL GENERATE VERIFY THE PODS ARE UP AND RUNNING, - ESIGNET NAMESPACE WILL GENERATE VERIFY THE PODS ARE UP AND RUNNING # esignet-keycloak_init # esignet # oidc-ui

CREATING MOSIP_MOCKIDENTITYSYSTEM DATABASE:

cd esignet-mock-services/db_scripts/mosip_mockidentitysystem

chmod +x deploy.sh

nano deploy.properties

DB_SERVERIP=api-internal.tf1.idencode.link ---> edit internal api hostname EX. api-internal.tf1.idencode.link DML_FLAG=1 ---> make DML_FLAG=1

COPY POSTGRES PASSWORD. GO TO EKS CLUSTER -> POSTGRES NS -> STORAGE -> SECRETS -> COPY postgres-postgresql PASSWORD

RUN BELOW COMMAND TO GENERATE IDP DATABASE

`./deploy.sh deploy.properties` OR `bash deploy.sh deploy.properties`

enter DB password multiple times it will be asked

> ⚠ NOTE: VERIFY IN PGADMIN WEATHER MOSIP_MOCKIDENTITYSYSTEM DB IS GENERATED OR NOT

GO TO EKS CLUSTER -> POSTGRES NS -> STORAGE -> SECRETS -> COPY db-common-secrets PASSWORD and Copy

OPEN PG ADMIN

click on Login/Group Role --> right click on mockidentitysystemuser --> properties --> defination (paste db-common-secrets as a password)

click on privilages

Superuser? ENABLE ---> this will Enable multiple options automatically

click on save

DEPLOY ESIGNET-MOCK-SERVICES :

cd esignet-mock-services/helm/mock-relying-party-service
helm dependency build
helm repo update

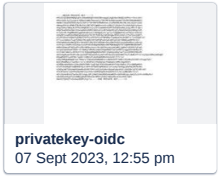
cd esignet-mock-services/helm/mock-identity-system
helm dependency build
helm repo update

cd esignet-mock-services/helm/mock-relying-party-ui
helm dependency build
helm repo update

cd esignet-mock-services/helm
./install-all.sh

PROVIDE THE REQUIRED INFORMATION:

Privatekey-oidc :-



privatekey-oidc
07 Sept 2023, 12:55 pm

Please provide client private key file : /home/shiv/B3-E2E/privatekey-oidc (***path of above file in your local***)
Please provide jwe userinfo private key file : /home/shiv/B3-E2E/privatekey-oidc (***path of above file in your local***)

Please provide Esignet service url : (default: http://esignet.esignet/v1/esignet) (do enter)

Please provide mock relying party ui domain (eg: healthservices.sandbox.xyz.net) : healthservices.tf1.idencode.link (provide you host)

DEPLOYMENT OF PARTNER-ONBOARDER FOR ESIGNET :

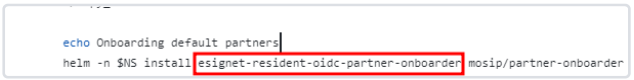
To onboard esignet partners (use this branch https://github.com/tf-govstack/mosip-infra/tree/tf-develop-B3/deployment/v3/mosip/partner-onboarder - Connect your Github account)
Make sure edit values.yaml as per below instructions

nano `values.yml`

VERIFY AND BE CAREFULL ONLY BELOW GIVEN MODULES MUST BE "TRUE":

– name: esignet; enabled: true – name: resident-oidc; enabled: true (Make sure others should be false apart from these two)

Make sure edit **install.sh** with this name while uploading certs or else it will give error.



RUN:

`./install.sh`

⚠ **FOLLOW THE BELOW SETPS CAREFULLY :-**

ONBOARDING THE DEFAULT ESIGNET PARTNER

a). After successfull partner onboarder run for esignet , download html reports from onboarder bucket of object store minio.

b). Get licensekey from response body of request create-the-MISP-license-key-for-partner from the report e-signet.html

c). Update & commit value of mosip.esignet.misp.license.key parameter with licensekey value from last step in esignet-default.properties

d). Restart esignet pod.

ONBOARDING THE DEFAULT RESIDENT-OIDC PARTNER

a). After successfull partner onboarder run for resident-oidc , download html reports from onboarder bucket of object store minio .

b). Get clientId from response body of request create-oidc-client from the report resident-oidc.html .

c). Update & commit value of mosip.iam.module.clientID parameter with clientId value from last step in resident-default.properties

d). edit the mimoto-default.properties config for the below mentioned keys :

wallet.binding.partner.id=mpartner-default-resident-oidc

wallet.binding.partner.api.key=cSjlqz2P1LddutpOaEQf2HChr1116UwRKBUX9OJLikU ---> {replace with generated oidc client id}

e). **Restart resident pods.**

DEPLOYMENT OF PARTNER-ONBOARDER FOR ESIGNET MOCK SERVICES :

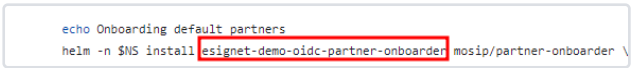
cd esignet-mock-services/partner-onboarder

nano values.yml

⚠ VERIFY AND BE CAREFULL ONLY BELOW GIVEN MODULE MUST BE "TRUE":

- name: demo-oidc enabled: true (make sure apart from this others should be false)

Make sure edit **install.sh** with this name while uploading certs or else it will give error.



RUN:

`./install.sh`

FOLLOW THE BELOW SETPS CAREFULLY :-

1. **ONBOARDING THE DEFAULT DEMO-OIDC PARTNER**

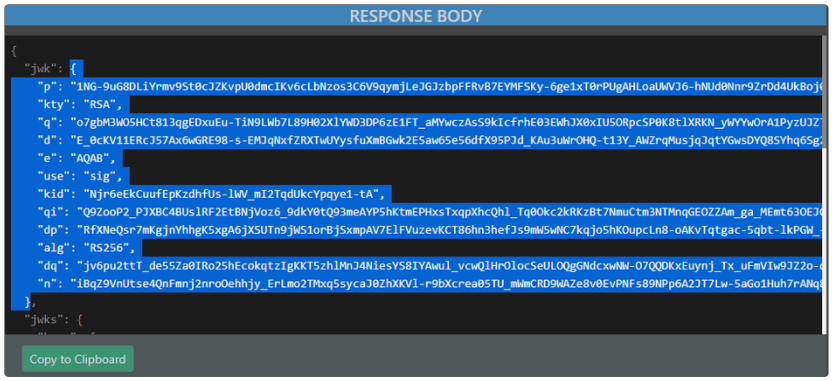
   a). After successfull partner onboarder run for demo-oidc partner , download html reports from onboarder bucket of object store minio.

   b). Get CLIENT_ID from response body of request create-oidc-client from the report demo-oidc.html

   c). Update deployment of mock-relying-party-ui in esignet namespace with CLIENT_ID value from last step .

d). As per screenshot get the private and public key pair (shown as selected in the screenshot) from the response of the get-jwks request from the report demo-oidc.html

---



NOTE: Then format the above selected private-public-keypair in json format and then encode json format with base64 format

e). Update client-private-key in esignet namespace with base64 encoded value of the keypair from previous step.

NOTE : DO ABOVE STEP CAREFULLY AS SHOWN BELOW: EKS cluster--> esignet (namespace) --> storage --> secrets --> mock-relying-party-service-secrets --> edit config --> replace existing client-private-key with base64 encoded value.

f). **Restart mock-relying-party-service pod**

ESIGNET DEPLOYMENT COMPLETED now access Url: healthservices.tf1.idencode.link {change as per your ENV} --> LOGIN WITH ESIGNET

Note: If facing intermittent connectivity issues while login esignet then please disable istio layer from softhsm namespace.

kubectl label ns softhsm istio-injection=disabled --overwrite

Troubleshooting

- If you are facing any issues while accessing the domain names that could be because `proxy-protocol` is not enabled in the target groups. or routing is not done properly and LB listners configurations are not done properly so check everything once again.
- When accessing istio-system from terminal it should show DNS name of load balancer in `EXTERNAL-IP` section or else not able to access endpoints. It causes because of multiple `security-groups` attached to your nodes. Make sure only one security-group attached to each node.

```
ramod@TECHNO-244:~/tfgovstack/mosip-infra-B3/deployment/v3/external/landing-page$ kubectl get svc -n istio-system
AME                          TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
stio-ingressgateway-internal   LoadBalancer   172.20.210.60   <pending>     15021:31741/TCP,80:31342/TCP,443:32060/TCP,61616:31245/TCP,5432:31536/TC
   3m53s
stiod                          ClusterIP      172.20.80.44    <none>        15010/TCP,15012/TCP,443/TCP,15014/TCP
   110m
```

- Sometimes while creating clusters the cluster will create successfully but it will not generate a cluster-config file in `.kube` folder then run this command to generate the config file after cluster creation.
  `eksctl utils write-kubeconfig -n <clustername>`
- Sometimes you will face PVC mount volumes issue for all the stateful sets, and completely env will go down because of restarting nodes or scaling down the nodes. So please avoid this in case of EKS or else you can use EFS as a default storage class to avoid such issues..
- If you want to decrease/increase the no of nodes, you can do that on EKS section NodeGroup group in AWS.
- If you want to delete the whole EKS setup follow this link here