Opensource - Minio

Deployment Notes

- Minio is used as storage for vre core and greenroom namespace.
- Encryption at rest is required, so we'll deploy KES and Vault as KMS.
- Encryption in transit is required, so we'll deploy Vault with TLS enable.

Vault Deployment

Opensource -- Vault

Vault Configuration

After Vault deployment and unsealing process:

1. Enable **Vault's K/V** backend

```
vault secrets enable -version=1 kv
```

Note: You may need to access a vault pod and login by using the root token which you can get from cluster-keys.json that you generated during Vault installation.
i.g.

$ cat cluster-keys.json
{
"unseal_keys_b64": [
….
"recovery_keys_threshold": 3,
"root_token": "**s.1RMD2pBWfGyQ3DDqe5ZGMWQn**"
}

$ **kubectl -n vault exec -it vault-0 -- sh**

$ **vault login**
Token (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

2. Enable **AppRole** authentication

```
vault auth enable approle
```

3. Create an **access policy** for the K/V engine

```
cat > kes-policy.hcl <<EOF
path "kv/*" {
     capabilities = [ "create", "read", "delete" ]
}
EOF
```

4. **Upload** the policy to Vault:

```
vault policy write kes-policy ./kes-policy.hcl
```

5. Create a new **AppRole ID** and bind it to a policy

```
# record role_id and secret_id from the last two steps
vault write auth/approle/role/kes-role token_num_uses=0
secret_id_num_uses=0  period=5m
vault write auth/approle/role/kes-role policies=kes-policy
vault read auth/approle/role/kes-role/role-id
vault write -f auth/approle/role/kes-role/secret-id
```

KES Deployment

1. Generate a **TLS private key** and **certificate** for the **KES** server

```
kes tool identity new --server --key server.key --cert server.cert --ip
"127.0.0.1" --dns kes.minio
```

Note: You can download kes tool from https://github.com/minio/kes . Download the binary release and run as executable script directly. Or you can download from https://git.indocresearch.org/platform/platform_maintenance/-/blob/master/kubernetes/minio/kes if the official one doesn't work.

2. Create **private key** and **certificate** for application and record app identity

```
kes tool identity new --key=app.key --cert=app.cert app
kes tool identity of app.cert
```

3. Create **private key** and **certificate** for **minio** and record minio identity

```
kes tool identity new --key=minio.key --cert=minio.cert MinIO
kes tool identity of minio.cert
```

4. Create **KES deployment config** file – kes-config.yaml

```
root:    disabled  # We disable the root identity since we don't need
it in this guide

tls:
  key:  /var/lib/docker/server.key
  cert: /var/lib/docker/server.cert

policy:
  MinIO:
# WARNING: if you are using v0.15.0 or later you should not use "path:"
here and instead
# you shoud use "allow:"
    paths:
    - /v1/key/create/my-minio-key
    - /v1/key/generate/my-minio-key
    - /v1/key/decrypt/my-minio-key
    identities: # from KES deployment step 2&3
    - f73748a843e164be3970a745966010fc1c9496d2bc55adf1f3d485d5163290a5
    - eec51496c11be079b0c6637f2350c4387a6ac263e1fdf1f34be44a80cb755872

keystore:
  vault:
    endpoint: http://vault.vault:8200
    approle:
      id:     "0b965209-9d3e-7dbf-4de1-44d6f007b647" # from vault
configuration step 5
      secret: "948a8391-a57e-6b19-1b8a-464b2a1b1e8b" # from vault
configuration step 5
      retry:  15s
    status:
      ping: 10s
    tls:
      ca: /var/lib/docker/vault.crt # Since we use self-signed
certificates
```

5. Create **config map** or secret for deployment:

```
# create configmap or secret and mount into kes and minio deployment
## this vault.crt is from vault deployment.
kubectl create cm vault.crt --from-file vault.crt -n minio
## these cred files are from minio and kes deployment
kubectl create cm kes-config      --from-file kes-config.yml -n minio
kubectl create cm minio.cert      --from-file minio.cert -n minio
kubectl create cm minio.key       --from-file minio.key -n minio
kubectl create cm server.cert.kes --from-file server.cert -n minio
kubectl create cm server.key.kes  --from-file server.key -n minio
```

6. Deploy **KES** (kes.yaml)

```
kubectl apply -f kes.yaml
```

7. Create a **key** for Minio (Note: The key's name is defined in the minio deployment yaml):

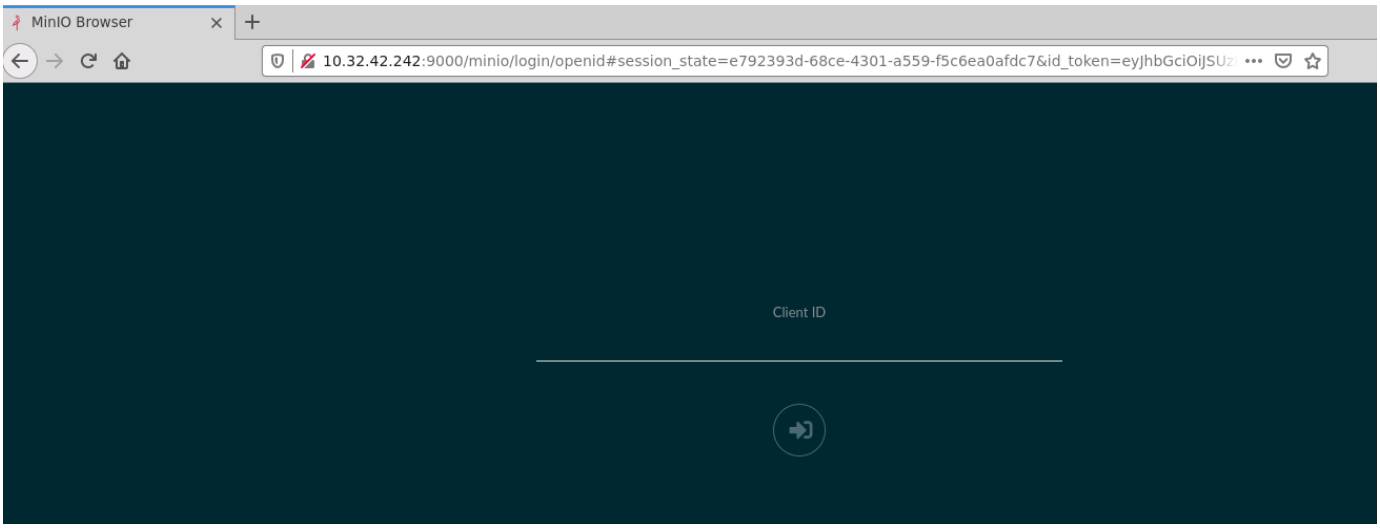
```
# create key in kes, key will store in vault, make sure below cert/key
path is correct:
export KES_SERVER=https://x.x.x.x:7373 # x.x.x.x is external IP for KES
service
export KES_CLIENT_CERT=minio.cert
export KES_CLIENT_KEY=minio.key
kes key create -k my-minio-key

# check key in vault
vault list kv
```

8. **Deploy** Minio (minio-dev.yaml):

```
kubectl apply -f minio-dev.yaml
```

Keycloak Configuration

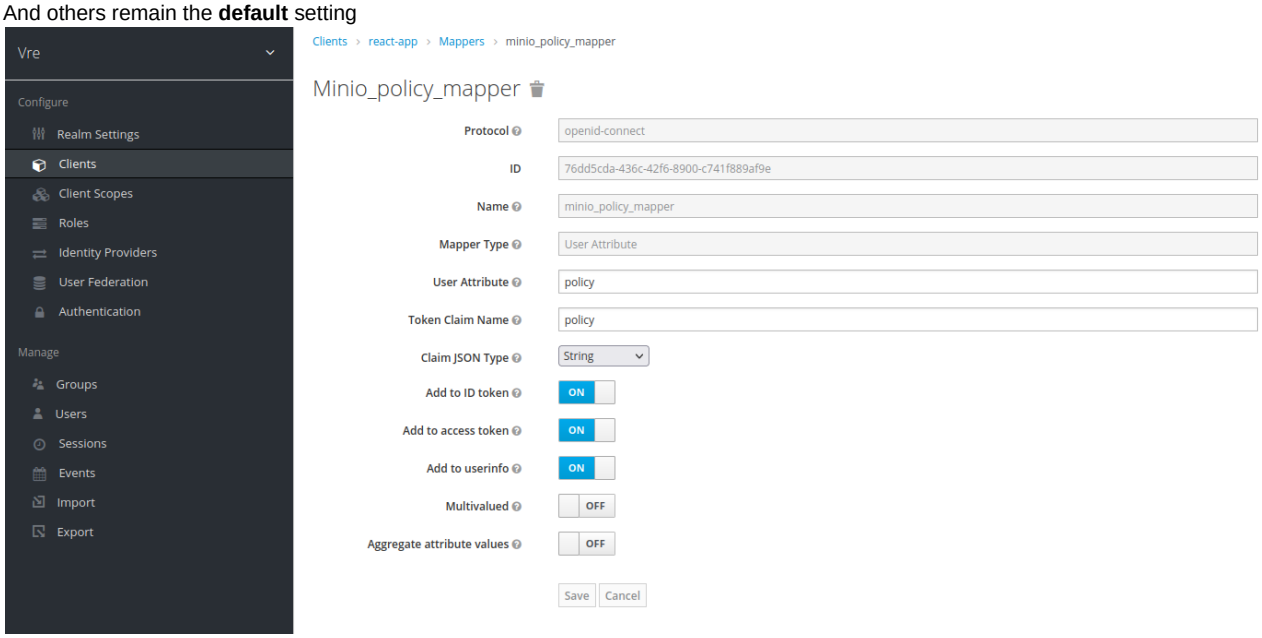
In order to access the Minio via Keycloak, you have to make some configuration in Keycloak end, otherwise, after entering username/password, you will get below screen:



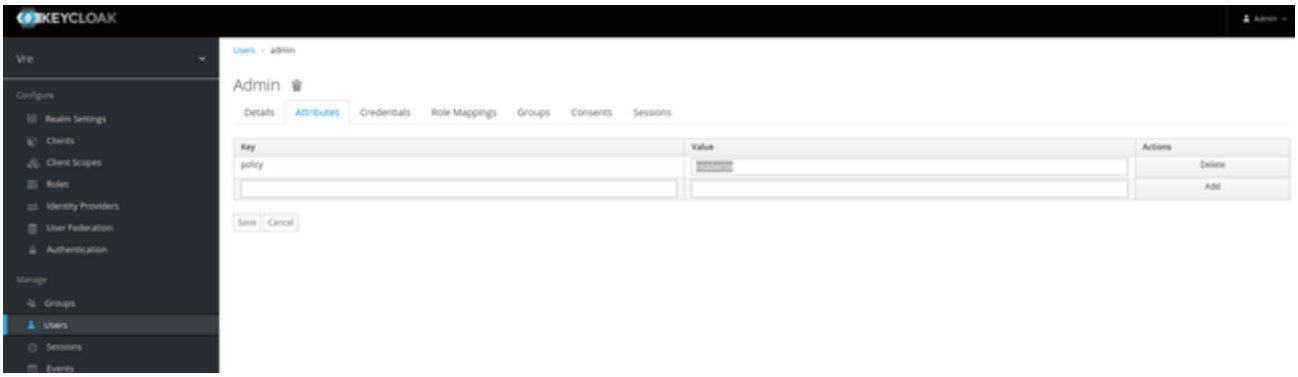
These are steps to configure Keycloak

1. Login to Keycloak and go to “**Client**” “**react-app**” “**Mappers**”
2. Create a **mapper** called “**minio_policy_mapper**” with below configuration
   Name: minio_policy_mapper
   Mapper Type: User Attribute
   User Attribute: policy
   Token Claim Name: policy
   Claim JSON Type: String

   And others remain the **default** setting



3. Go to “**Users**” section and search for “**admin**” user. And then go to its “**Attributes**” tab and add below attribute
   Key: policy
   Value: readwrite



4. **Close** the browser and **re-login** again. You should be able to access the Minio now. If you want to **enable the encryption**. You can run below script in the internal network to enable the specific bucket:

```
$ pip3 install minio
$ Create minio-init.py with below content

from minio import Minio
from minio.commonconfig import Tags, ComposeSource
from minio.versioningconfig import OFF, SUSPENDED,
VersioningConfig, ENABLED
from minio.notificationconfig import (NotificationConfig,
PrefixFilterRule,
                                      QueueConfig)
from minio.sseconfig import Rule, SSEConfig


client = Minio(
    "10.32.42.242",
    "indoc-minio",
    "Trillian42!",
    secure=False)

client.set_bucket_encryption(
    "test-20210617", SSEConfig(Rule.new_sse_s3_rule()),
)

# Run the initial script to enable the encrytion feature
$ python3 minio-init.py
```

Update KES Certificate

By default, the certificate created by kes will **expire in 30 days**. You need to replace a new certificate before it expires, otherwise you will get the below error when uploading or downloading any file from minio:

"x509: certificate has expired or is not yet valid"

Note: **Never delete the master key in vault kv**, otherwise you won't be able to read/download any file updated/encrypted before!

The below are the steps to update the KES certificate

1. **Create** a new key/certificate for KES server

```
# Note: -t specifies the expire days for the certificate. By
default, it is 30 days.
kes tool identity new --server --key server.key --cert server.cert
--ip "127.0.0.1" --dns kes.minio -t 8760h
```

2. **Delete** the key/certificate **configmap** in minio namespace

```
kubectl config use-context <dev|staging|charite>
kubectl -n minio delete cm server.cert.kes server.key.kes
```

3. **Create** a new key/certificate configmap in minio namespace from the new key/certificate generated in the step 1

```
kubectl create cm server.cert.kes --from-file server.cert -n minio
kubectl create cm server.key.kes  --from-file server.key -n minio
```

4. **Restart** both KES server and Minio pod

```
kubectl -n minio rollout restart deployment/kes
kubectl -n minio rollout restart statefulset.apps/minio
kubectl -n minio get all
```

5. **Verify** if the certificate is updated by **downloading** a **old** file and **uploading** a file

Certificate Update Tracker

DEV	STG	PRD
Updated on **20220615** for 8760h (1 year) Should be updated before **20230615**	Updated on **20220624** for 8760h (1 year) Should be updated before **20230623**	Updated on **20210628** for 8760h (1 year) Should be updated before **20220628**

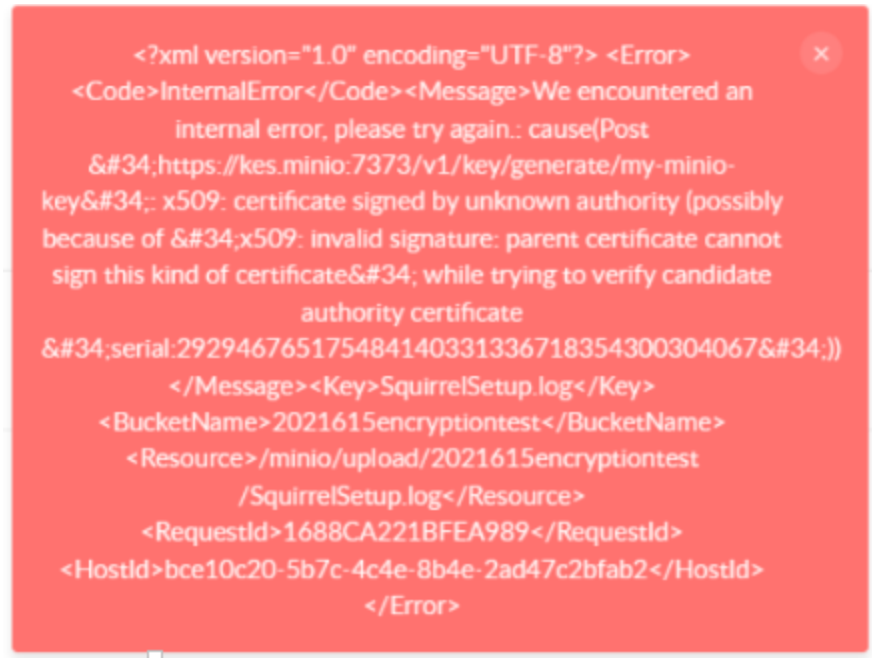
Troubleshooting

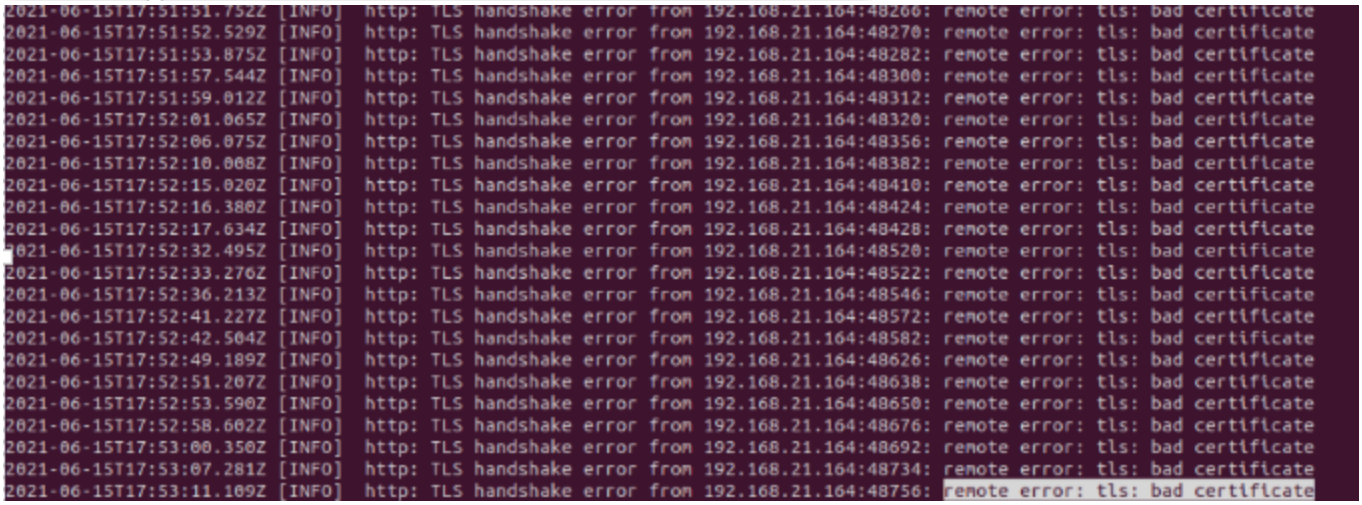
Issue 1: x509: certificate has expired or is not yet valid

Cause: The KES server or vault certificate has expired

Solution: Update the KES server certificate. Please refer to above "**Update KES Certificate**" section to renew the certificate. For vault certificate updates, please refer to https://indocconsortium.atlassian.net/wiki/spaces/PILOT/pages/2504261745/Opensource+--+Vault#Vault-Certificate-Renewal .

Issue 2: X509: certifcate signed by unknown authority / remote error: tls: bad certificate





Cause: The certificate that minio is using doesn't match with the one in KES server

Solution: If it happens after you renew the KES server certificate, please make sure to replace the server.kes.crt configmap and also **restart** the Minio pod once it is done. Please refer to above "**Update KES Certificate**" section to renew the certificate. For vault certificate updates, please refer to https://indocconsortium.atlassian.net/wiki/spaces/PILOT/pages/2504261745/Opensource+--+Vault#Vault-Certificate-Renewal .

Issue 3: key does not exist

Cause: The master key which is defined in Minio deployment yaml doesn't exist in Vault.

Solution: Create a master key in Vault and make sure the name is matching up with the one defined in Minio deployment yaml. Please refer to **"KES Deployment" Section Step 7**.

Issue 4: Cannot upload a file to the subfolder in any bucket in Minio dashboard

Cause: It could be caused by the incorrect routing in ingress

Solution: No solution so far. Since we don't use Minio dashboard to upload file, it won't impact the upload process.

Reference

MinIO

https://github.com/minio/kes/wiki/Hashicorp-Vault-Keystore

https://github.com/minio/kes/wiki/MinIO-Object-Storage

https://github.com/minio/kes/wiki/Getting-Started

https://indocconsortium.atlassian.net/wiki/spaces/VRE/pages/1361838420/MinIO