Gospel Deployment Guide - all clouds

Installation Steps

1.Prerequisites Checklist

- [x] Cloud account + CLI
- [x] Kubectl CLI - Install command if you don't have it.
- [x] Kubernetes deployment .yaml files for your deployment (pvcs, secrets, services, pods)
- [x] A valid genesis block file
- [x] One intermediate certificate and private key for each namespace
- [x] One machine intermediate and private key for the network
- [x] Certificate and key for each peer, orderer, backend signed by machine intermediate
- [x] Certificate and key for each keystore signed by machine intermediate (one keystore component is deployed for each namespace)

Before you begin

- Check that you are in the correct Kubernetes context before making any changes

```
kubectl config current-context
```

- Gospel Certificates Guide#GospelCertificatesGuide

2. Deploy Kubernetes Cluster

- Deploying Kubernetes clusters on different cloud providers

Platform	Link
Azure	Creating A Kubernetes Cluster in Azure on a custom subnet
AWS EKS	Configure an EKS cluster in AWS

GCP	Creating a Kubernetes Cluster in Google Cloud Platform Google Marketplace Installer

AWS with Kops -> main deployment replaced with AWS EKS from the link above. For a Kops deploy use Guide 1 or Guide 2 or this list of commands.

3. Deploy additional components: Kubernetes Dashboard and Monitoring with Grafana and Prometheus

- **Deploy a Kubernetes dashboard**

```
#deploy kubernetes dashboard
kubectl apply \
  -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy
/recommended/kubernetes-dashboard.yaml
```

- **Set up access permissions for the service account.**

KUBERNETES OFFICIAL GUIDE

```
#set up permissions for service account
kubectl apply -f kubernetes-dashboard-rbac.yaml
```

- **Deploy Monitoring with Grafana and Prometheus**

Deploying Prometheus and Grafana in a Gospel EKS cluster

4. Install Gospel components using the provided .yaml files.

Deploy in this order: Services, Secrets (inc. configmaps), Storage (persistent volume claims), Pods Deployments.

```
kubectl apply -f <Folder_Path_Containing_Configuration_Files>
```

5. Database authentication

The following section provides a method to set up a simple database for authentication.

Log into gospel-ca pod, mysql container and set up the authentication database

```
#MySQL setup script
#TODO add nonce to password field, change password

kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "CREATE DATABASE IF NOT EXISTS gospel;"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "CREATE TABLE IF NOT EXISTS gospel.ca_users
(username VARCHAR(100) NOT NULL, password VARCHAR(256) NOT NULL,
user_groups VARCHAR(50) NOT NULL, PRIMARY KEY(username));"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "CREATE TABLE IF NOT EXISTS gospel.otp
(username VARCHAR(100) NOT NULL, otp VARCHAR(100) NOT NULL, PRIMARY
KEY(username));"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "CREATE USER 'gospel_ca'@'%' IDENTIFIED BY
'password'"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "GRANT SELECT ON gospel.* TO 'gospel_ca'@'%';"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "FLUSH PRIVILEGES;"
kubectl exec gospel-ca-<pod-identifier> -c mysql-ca  -- mysql -uroot
-pPickYourOwnPassword -e "INSERT INTO gospel.ca_users
(username,password, user_groups) VALUES('trent.ksmith@gospel.tech',
PASSWORD('g0sp3l'), 'ADMIN');"
```

6. Set up the Peers and Orderer

- Copy your genesis block across to the peer container on both the peer pods and the orderer pod

```
kubectl cp Documents/your.genesis.block
peer0-32kejek:/etc/hyperledger/fabric/your.genesis.block -c peer0

#The destination on the orderer is defined in
ORDERER_GENERAL_GENESISFILE in its pod deployment
kubectl cp Documents/your.genesis.block
orderer0-79ygbj:/var/hyperledger/production/your.genesis.block -c
orderer0
```

- Have the peer join the channel - perform this step on all peers

Sign into the peer container on the peer pod. Join a peer to the channel defined in the genesis block. Refer to the pod configuration file for the location.

Reference

```
peer channel join \
  -o orderer0:7050 \
  -b /etc/hyperledger/fabric/your.genesis.block
```

- Confirm that it successfully joined using this command:

```
peer channel list

Example response:
...
Channels peers has joined:
testchainid
```

7. Install and instantiate the chaincode

- On the `peer0` container install the chaincode using this command. Repeat this step on all peers.

```
peer chaincode install -n GospelChaincode -p
"gospel-docker-local.jfrog.io/chaincode:develop-build-27" -v devb27 -l
image
```

The build id must change each time you perform an update. If you supplied Artifactory credentials when deploying the peer, this should now pull the docker image automatically-- you shouldn't need to log into `dind-daemon` to do so.

- Confirm that the chaincode has been successfully installed

```
peer chaincode list -C testchainid --installed

Example response:
...
Get installed chaincodes on peer:
name:"GospelChaincode" version:"b4527"
path:"gospel-docker-local.jfrog.io/chaincode:v4.5-build-27"
```

- Instantiate the chaincode

This is only required when setting up the first peer with a new chaincode. When running this command, ensure there are newline (\n) characters in the certificate.

```
peer chaincode instantiate \
  -n GospelChaincode \
  -v b4527 \
  -c '{"Function":"init",
"Args":["MACHINE_INTERMEDIARY_CERT","NAMESPACE_1_INTERMEDIARY","NAMESPAC
E_2_INTERMEDIARY"]}' \
  -o orderer0:7050 \
  -C testchainid

#Success
...
2019-04-10 03:41:38.502 UTC [chaincodeCmd] checkChaincodeCmdParams ->
INFO 008 Using default escc
2019-04-10 03:41:38.502 UTC [chaincodeCmd] checkChaincodeCmdParams ->
INFO 009 Using default vscc
2019-04-10 03:43:49.694 UTC [main] main -> INFO 00a Exiting.....
```

For all other peers, you can check the logs after joining the channel in order to make sure it's behaving correctly. If the chaincode fails to instantiate check the certificate for this peer is included in the genesis block.

Debugging and testing

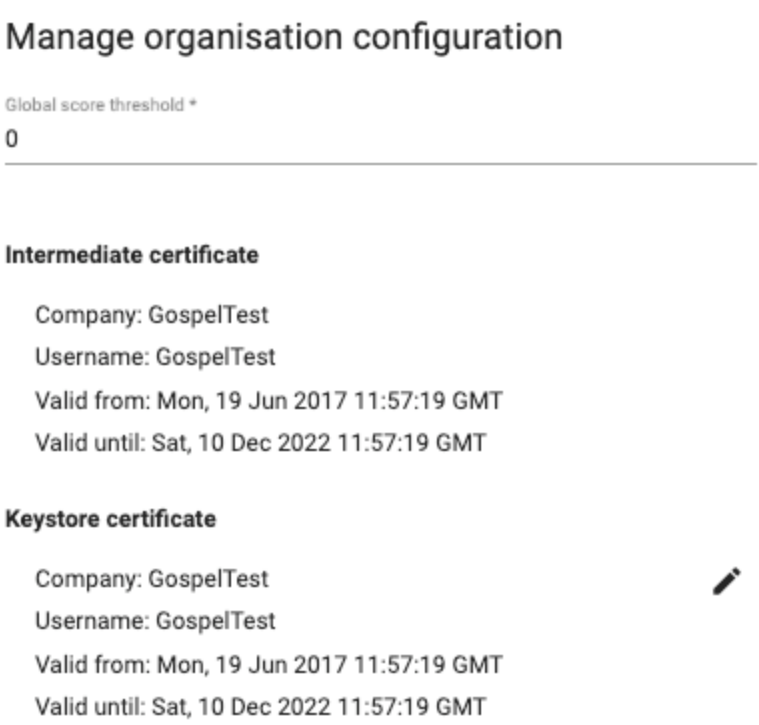
- Check for errors in the logs on kubernetes
- Confirm each peer has joined channel

```
peer channel list
```

8. Set up the Keystore

- For each namespace, log in as an administrator and set up the Keystore. Go to **Settings** - **Gospel config.**

- Click the add icon and add the keystore certificate.



- Define superusers (default ADMIN group)

Production Deployment Additional Steps

https://docs.microsoft.com/en-us/azure/aks/developer-best-practices-resource-management#regularly-check-for-application-issues-with-kube-advisor

https://docs.microsoft.com/en-us/azure/aks/node-updates-kured

Troubleshooting resources

- **Chaincode doesn't download**

THIS STEP IS NO LONGER NECESSARY
Pull the chaincode image to the `dind-daemon` container on peer pod.

- **This container does not have bash. Log in with SH**

```
docker login gospel-docker-local.jfrog.io -u "username" -p "password"
```

You must have a valid Gospel Artifactory account with permissions over the docker repository.

After logging in to the Gospel docker repo, you should be able to pull the chaincode image you want to install:

```
docker pull gospel-docker-local.jfrog.io/chaincode:build-158
```

Useful resources

- **Install kubectl on a Mac**

```
#install kubectl
#https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl
brew install kubernetes-cli
```

- **Kops deployment steps**

Note: These steps include a bastion pod for accessing the cluster

```
#deploy kubernetes
kops create cluster \
  --cloud aws \
  --node-count=3 \
  --node-size=t3.large \
  --zones=eu-west-1a,eu-west-1b,eu-west-1c \
  --master-count=3 \
  --master-size=t3.small \
  --master-zones=eu-west-1a,eu-west-1b,eu-west-1c \
  --dns-zone=Z1OSSOZ2HFXSVX \
  --dns=public \
  --name euwest1.prod3.cloud.gospel.tech \
  --networking canal \
  --network-cidr=172.20.0.0/16 \
  --associate-public-ip=false \
  --topology private \
  --bastion
```

- **Set up the secrets for JFrog authentication in Kubernetes**

Request an API account in JFrog through Jira Service Desk if you don't have credentials. The secret name needs to match the one in your config, login to JFrog with LDAP credentials. See `imagePullSecrets` in the deployments for the secret name.

```
apiVersion: v1
data:
  .dockerconfigjson: <HIDDEN>
kind: Secret
metadata:
  name: gospel-artifactory
  namespace: gospel
type: kubernetes.io/dockerconfigjson
```

- **Load the intermediate certificate into softhsm on the Signer pod**

This step is now automated in the configmap: *signer-setupConfigMap.yaml.* If you need to make changes to this see Adding a New Gospel Client Identifier

- **Configure and load CA's config.yaml**

Obtain the external IP of the nginx service

```
$ kubectl get services
NAME TYPE   CLUSTER-IP  EXTERNAL-IP  PORTS
nginx LoadBalancer 10.27.240.123 34.76.164.99 80:30080/TCP,443:30081/TCP
```

If you will be connecting to Gospel by a URL, put this IP into a record on your DNS server.

In the CA's pod deployment, paste the URL (or the IP, if you're not using DNS) next to `SIGNER_PATH`, `DEFAULT_REDIRECT`, and `INTERNAL_HOST`.

If using the Google authentication plugin, you need to configure the callback URL for Google authentication. Paste the same URL/IP in here. The changes for this are made in the caconfig.yaml kubernetes configmap.

- **Chaincode error**

```
Error: Error endorsing chaincode: rpc error: code = Unknown desc =
chaincode error
(status: 500, message: chaincode instantiation policy violated(Failed to
authenticate policy))


Error: Error getting broadcast client: error connecting to orderer0:7050
due to: context deadline exceeded
```

- **Upgrading the chaincode**

```
peer chaincode upgrade \
  -n GospelChaincode \
  -v b4527 \
  -c '{"Function":"init",
"Args":["MACHINE_INTERMEDIARY_CERT","NAMESPACE_1_INTERMEDIARY","NAMESPAC
E_2_INTERMEDIARY"]}' \
  -o orderer0:7050 \
  -C testchainid
```

- **Adding a new namespace to an existing deployment - currently replaced with Adding a New Gospel Client Identifier**

1. Obtain a new namespace intermediate certificate using GospelTest Root Cert or through a SupportCenter request for production
2. Add config to the Gospel CA and Signer for authenticating into the new namespace

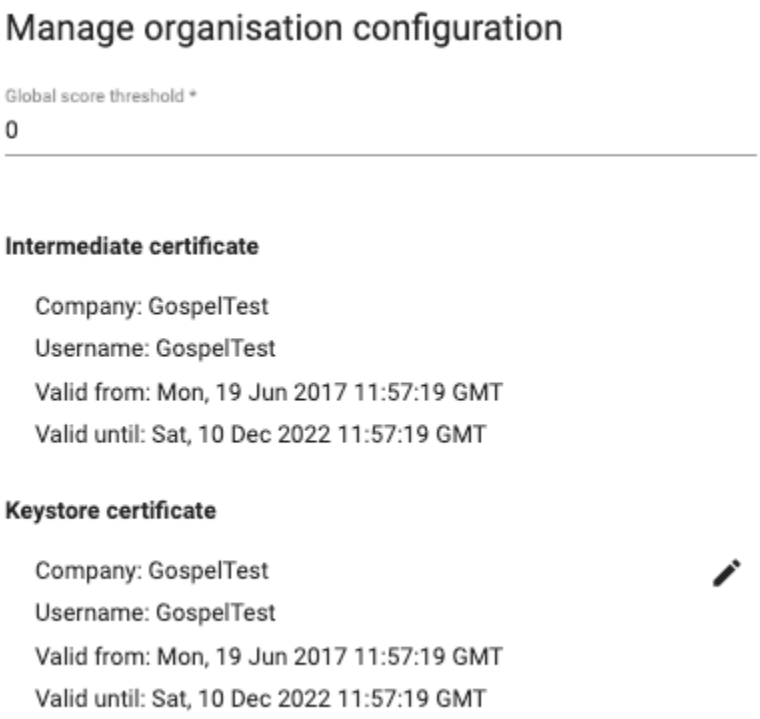
3. Set up a new keystore component with machine certs, and the new namespace cert under `KEYSTORE_INTERMEDIATE`.
4. Set up a service for the new keystore-- you will need it for re-configuring the backend.
5. Set up backend with the namespace certificate (INTERMEDIATE_CERTS env var, certificates are comma separated)
6. Set up backend so that it is pointing to the new keystore (KEYSTORES env var). The URL is the name of the new service you just created.

```
- name: KEYSTORES
              value:
"homeowner|http://keystore-service1:7060,travelagent|http://keystor
e-service2:7060,GospelTest|http://keystore-service3:7060"
```

7. Ensure Chaincode has been updated with the namespace certificate

```
peer chaincode upgrade -n GospelChaincode -v bmaster13 -c
'{"Function":"init", "Args":["-----BEGIN CERTIFICATE-----\n ...
\n-----END CERTIFICATE-----","-----BEGIN CERTIFICATE-----\n ...
\n-----END CERTIFICATE-----"]}' -o orderer0:7050 -C testchainid
```

8. Log into the Gospel UI and go to **Settings** > **Gospel config.** Under "**Manage organisation configuration**", paste in the certificates you used (minus the `\n` new line markers) in the keystore's deployment. "Keystore certificate" should match `KEYSTORE_CERTIFICATE` and "**Intermediate certificate**" should match `KEYSTORE_INTERMEDIATE`



- For generating certificates use Gospel Certificates Guide#Setuptheenduserintermediatecertificate