
ELEMENTAL CONCEPT

Hyperledger Fabric Workshop

Operations

Paul Sitoh

10-11 November 2018

Disclaimer

- Applies to version 1.1
- This is one way of implementing
- For development only
- Use for production at your own risk!

Agenda

- Prerequisite
- Cryptographic
- Channel configuration
- Network configuration using Docker compose
- Fabric CA
- Adding a new organization to an existing channel
- Kafka/Zookeeper (Clustering orderers)

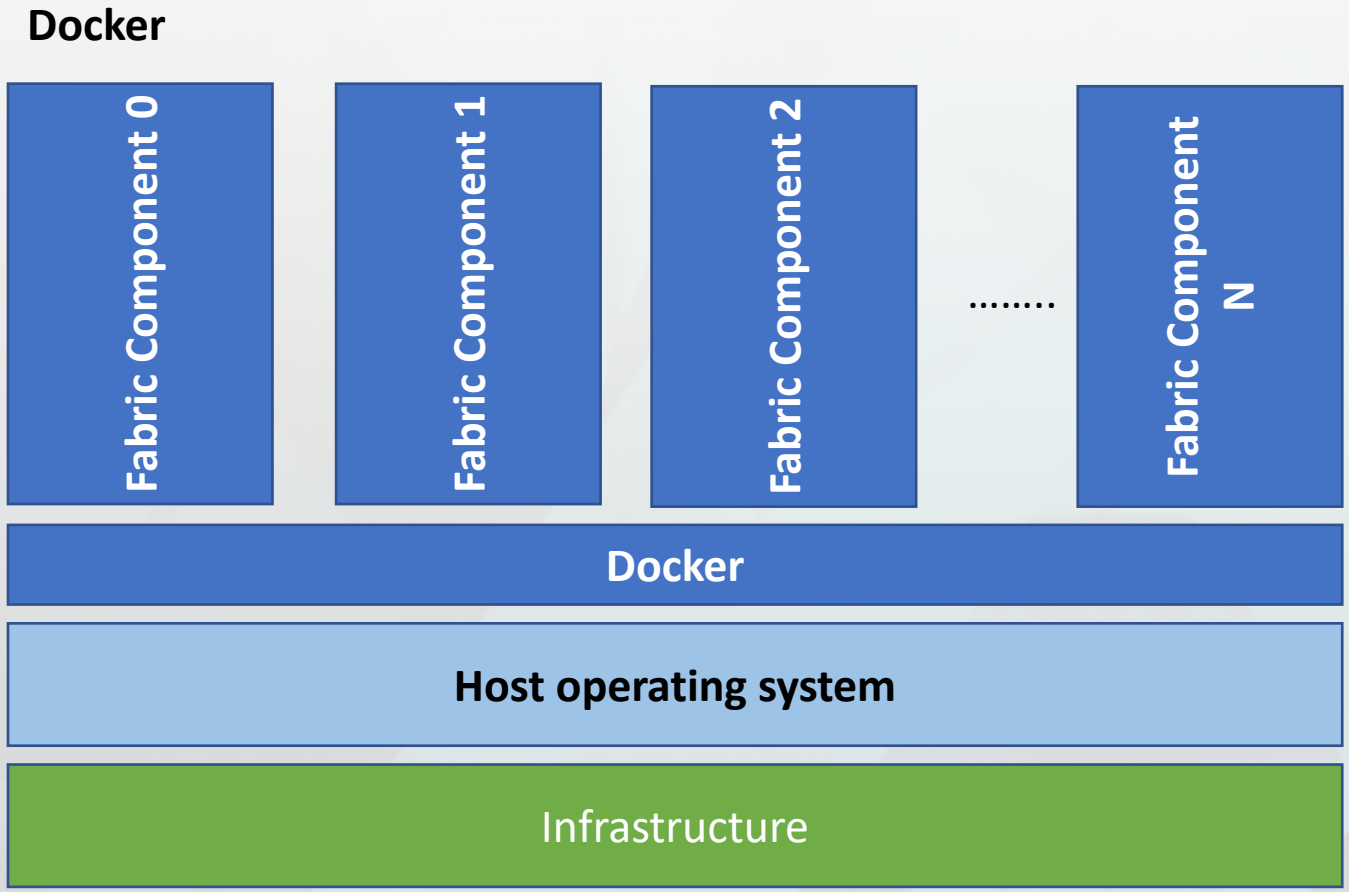
Prerequisite

Fabric operations

Prerequisite

- Install Git
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- Install Docker
  - https://tuleap-documentation.readthedocs.io/en/latest/developer-guide/quick-start/install-docker.html
- Install Go
  - https://golang.org/doc/install#install
- Install Fabric Developer Kit
  - https://github.com/aladdinid/fabric-devkit

Prerequisite



- Image
  - reusable code with executable configuration
- Container
  - runnable app
- Docker-compose
  - script for orchestrating containers

Prerequisite

Let’s see docker in action!

Prerequisite

Setup for Go development

- Assuming you have install Go and on macOS/Linux
- Set the environmental variable `GOPATH` to a reference a directory to host your Go source codes and binaries (i.e. Go workspace). For example,

```
export GOPATH=$HOME/go-projects
```

Prerequisite

Fabric Development Kit

- Create a folder in $GOPATH/src and navigate to it. Under the folder clone this repository.
- Navigate (cd) into $GOPATH run the command

  go get github.com/aladdinid/fabric-devkit

Prerequisite

Let’s check out the devkit!

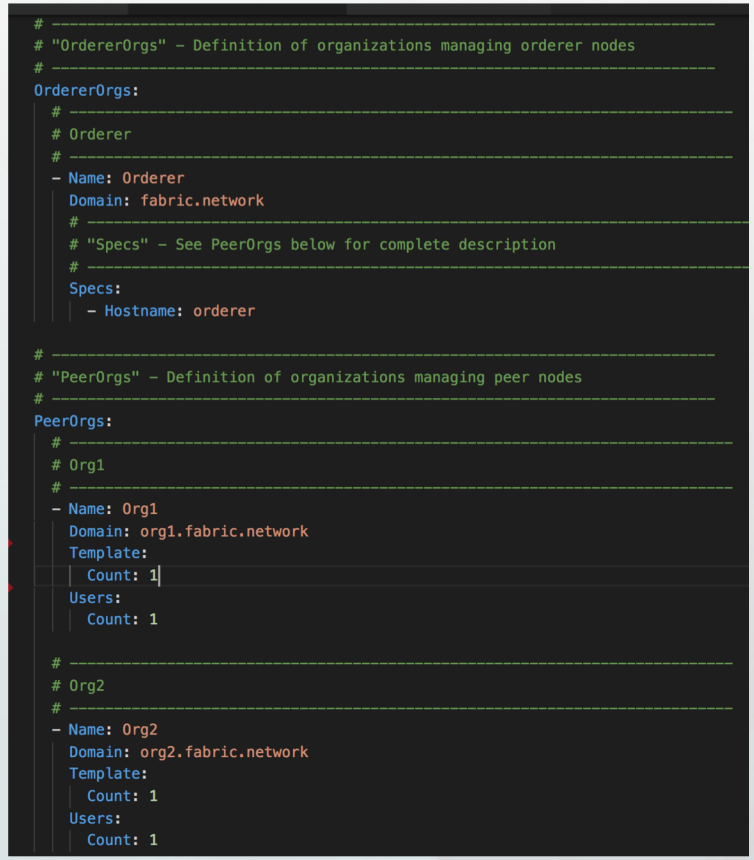
Cryptographic assets

Fabric Operations

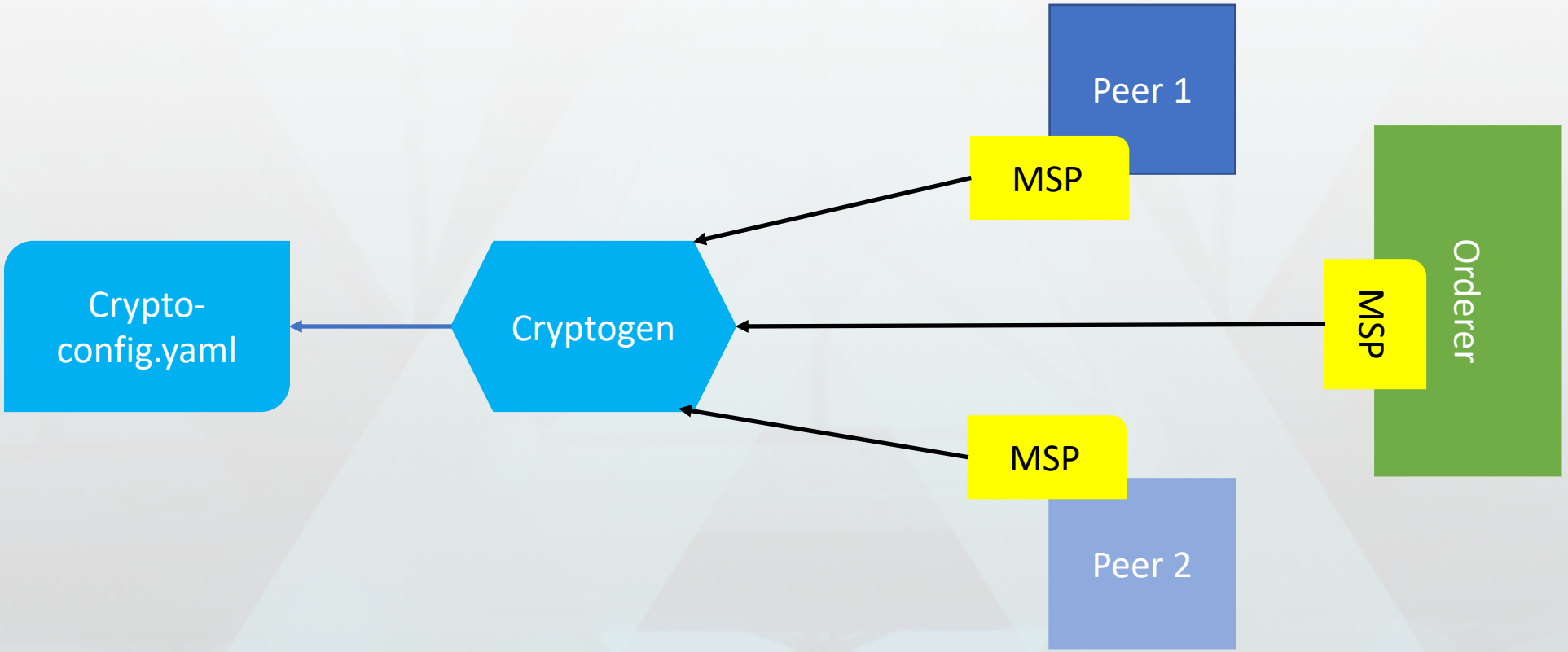
Cryptogen

- Assume that we are going to build a network with two organisations as members (Org1 and Org2)
- Cli cryptogen
- Crypto-config.yaml
- Run the command:

  cryptogen generate --config=./crypto-config.yaml --output="./assets/crypto-config"



Cryptogen


Peer 1
MSP
Orderer
MSP
Crypto-
config.yaml
Cryptogen
MSP
Peer 2

Channel configuration

Fabric Operations

Channel configuration

- Assume that we are going to build a network with two organisations as members (Org1 and Org2)
- Cli configtxgen
- Configtxgen.yaml
  - Generates genesis block
  - Generates channel

```
Profiles:

    TwoOrgsOrdererGenesis:
        Capabilities:
            <<: *ChannelCapabilities
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
    TwoOrgsChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
            Capabilities:
                <<: *ApplicationCapabilities
```

Channel configuration

```
peer channel create -o $ORDERER -c $CHANNELNAME -f
./channel-artefacts/channel.tx --tls --cafile
$ORDERER_CA

peer channel join -o $ORDERER -b ./$CHANNELNAME.block
--tls --cafile $ORDERER_CA
```

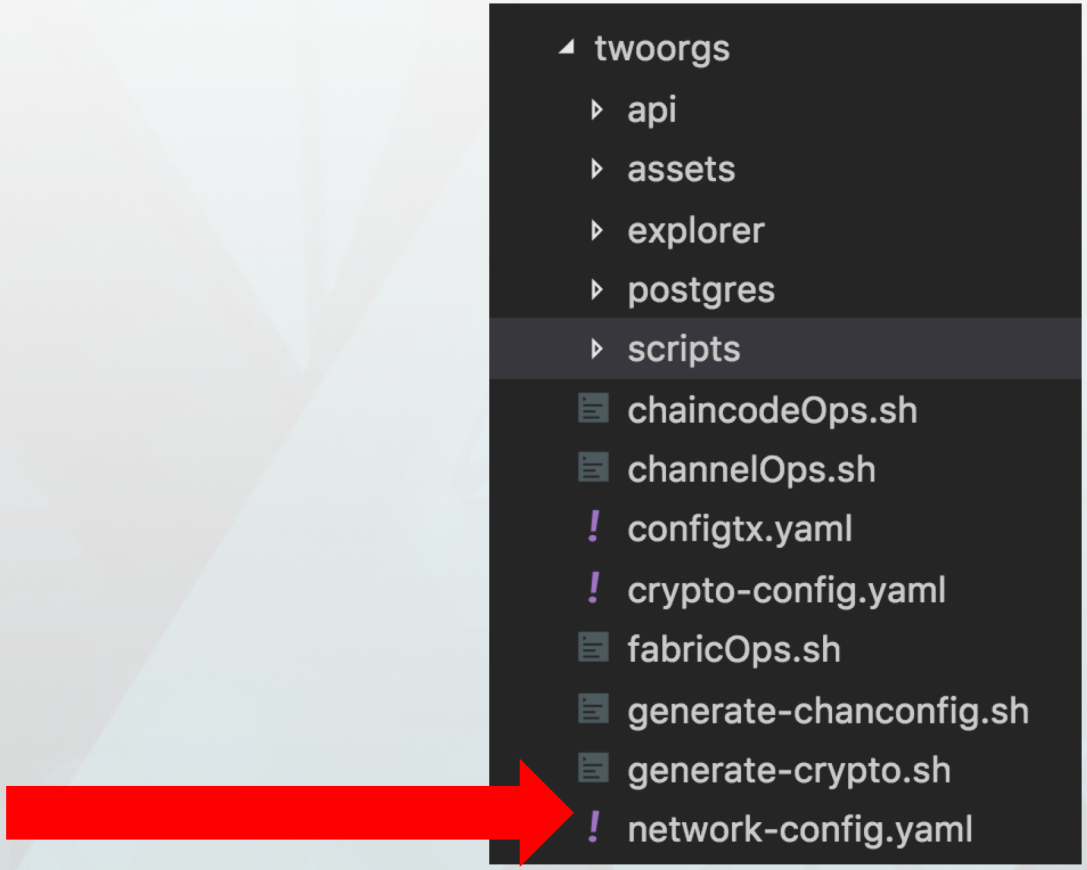
Network configuration using Docker compose

Fabric Operations

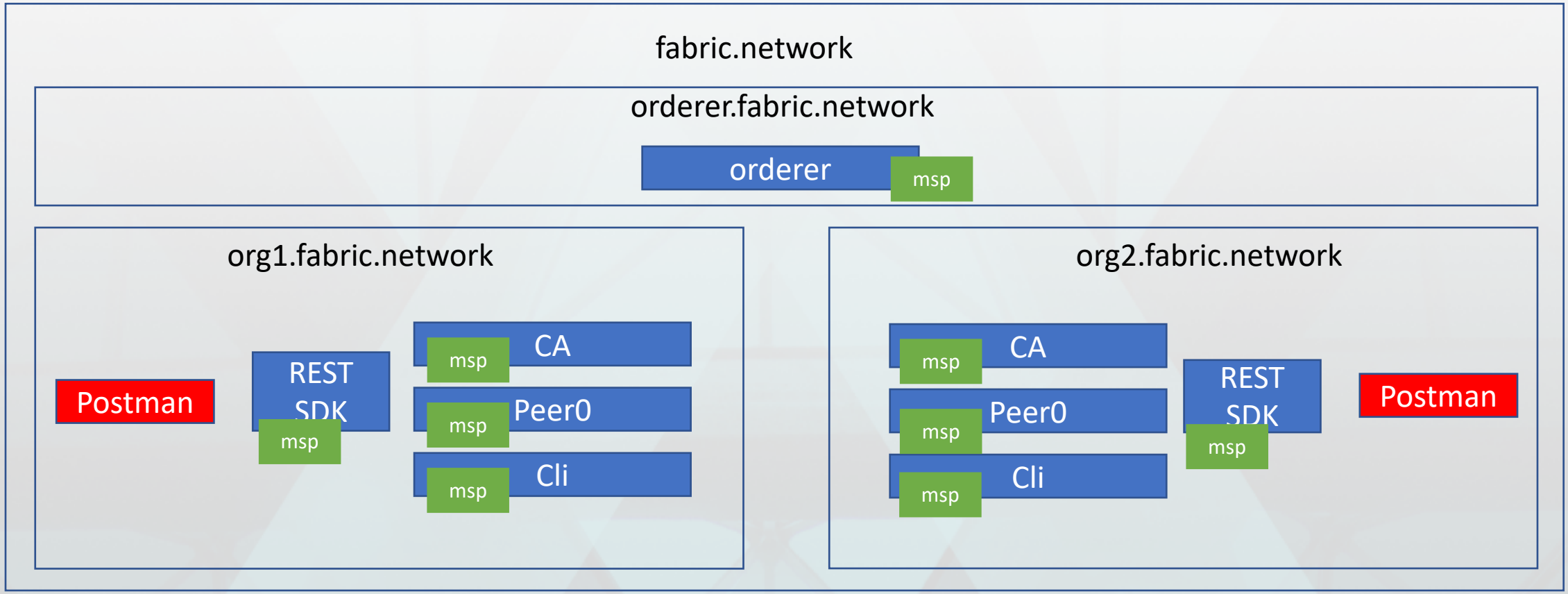
Steps

- Spin up a network via Docker compose
- Configure channel
- Install and instantiate chaincode

Spin up a network via Docker compose



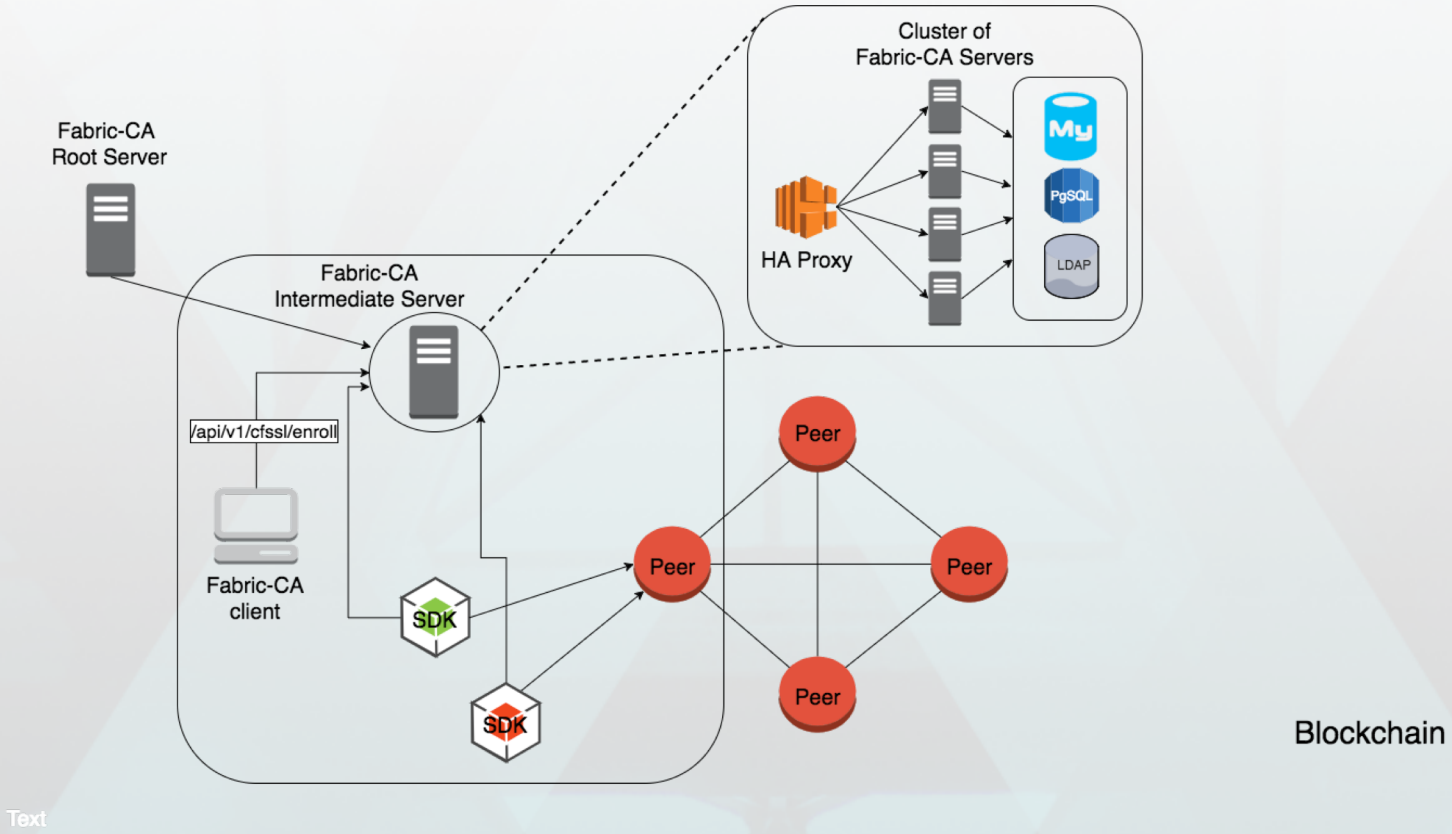
Network configuration



Fabric CA

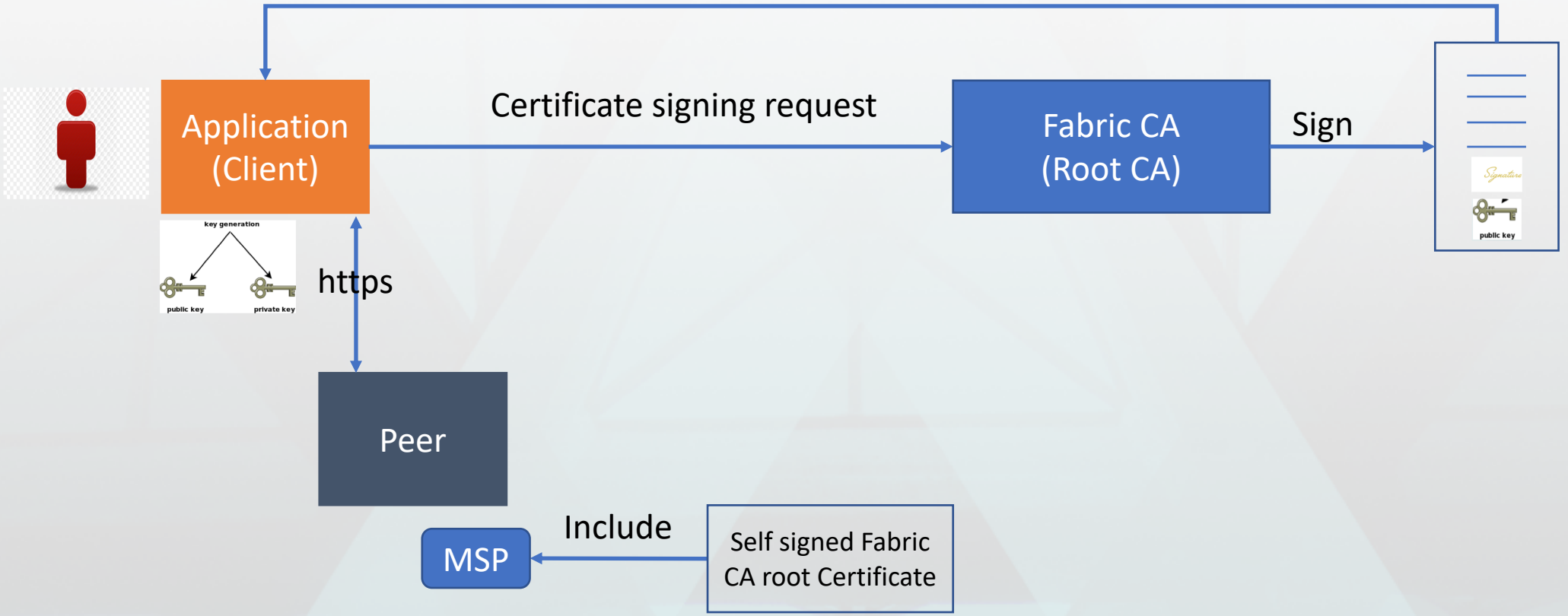
Fabric Operations

Fabric CA(1)



(1) https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html

Public Key Infrastructure (Self-signed)



Adding a new organization to an existing channel

Fabric operations

Adding a new organization to an existing channel

- Ensure the new organizational node has all the necessary cryptographic materials installed and able to see the orderer of the existing network
- Attach a cli to organization and fetch the block configuration from the orderer using the command:

```
peer channel fetch config config_block.pb -o orderer.example.com:7050 -c
$CHANNEL_NAME --tls --cafile $ORDERER_CA
```

- Extract the JSON from config_block.pb using configtxlator and jq by issuing this command:

```
configtxlator proto_decode --input config_block.pb --type common.Block | jq
.data.data[0].payload.data.config > config.json
```

Adding a new organization to an existing channel

- Add new organization to the configuration:

```
jq -s '.[0] * {"channel_group":{"groups":{"Application":{"groups": {"Org3MSP":.[1]}}}}}' config.json ./channel-artifacts/org3.json > modified_config.json
```

- Convert the JSON configuration back to a special format known as protobuf

```
configtxlator proto_encode --input config.json --type common.Config --output config.pb
```

- Encode the modified JSON configuration back to protobuf

```
configtxlator proto_encode --input modified_config.json --type common.Config --output modified_config.pb
```

Adding a new organization to an existing channel

- Compute the difference between the original configuration and the modified one.

```
configtxlator compute_update --channel_id $CHANNEL_NAME --original config.pb --updated modified_config.pb --output org3_update.pb
```

- Convert the new channel configuration into org3_update.pb JSON and add the header field associated back to the JSON

```
echo '{"payload":{"header":{"channel_header":{"channel_id":"mychannel", "type":2}},"data":{"config_update":'$(cat org3_update.json)'}}}' | jq . > org3_update_in_envelope.json
```

- Convert the updated json to PB

```
configtxlator proto_encode --input org3_update_in_envelope.json --type common.Envelope --output org3_update_in_envelope.pb
```

Adding a new organization to an existing channel

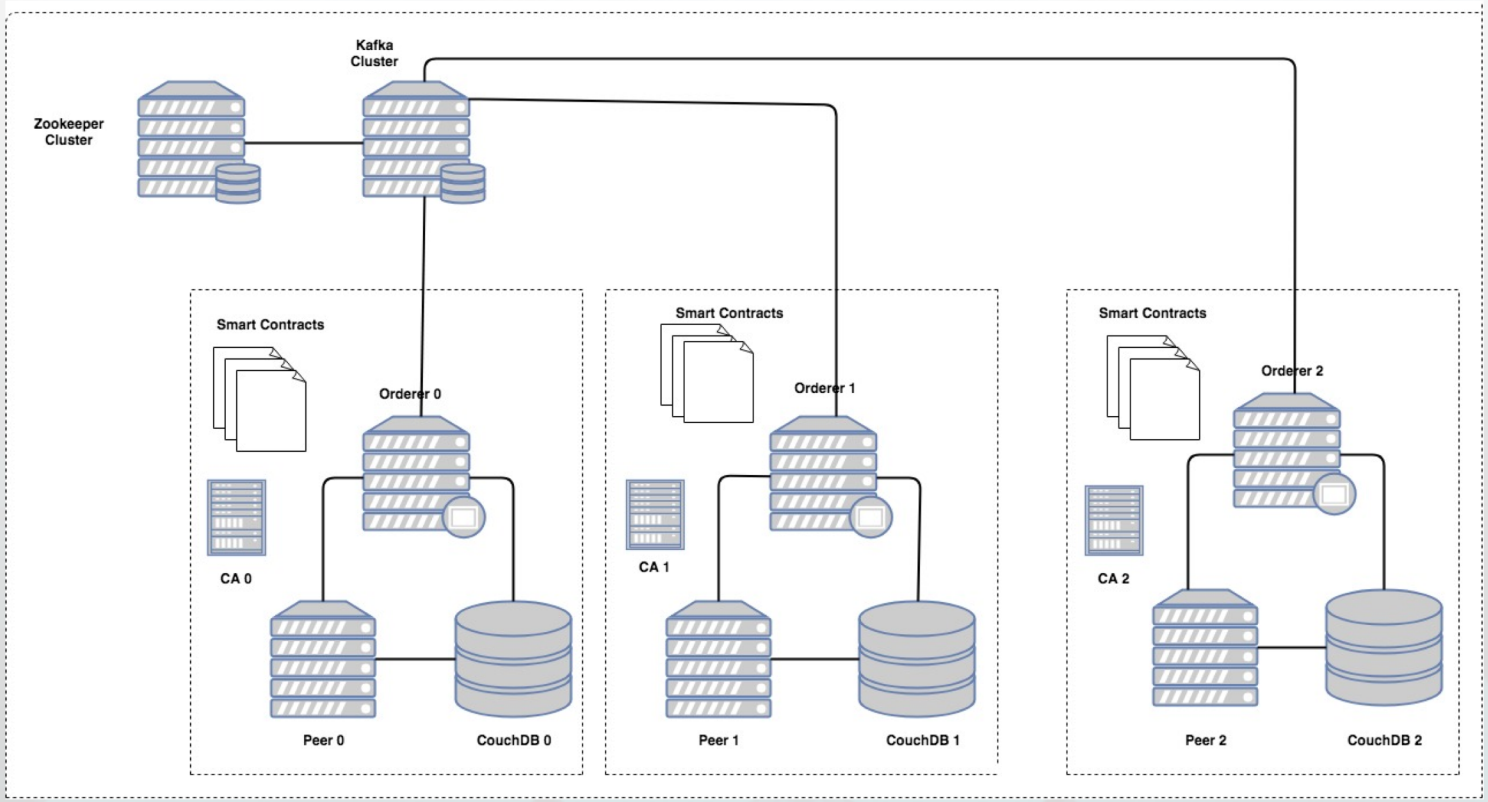
- Submit the new updated config.

```
peer channel signconfigtx -f org3_update_in_envelope.pb
```

Kafka/Zookeeper

Fabric operations

Kafka/Zookeeper(1)



(1): https://medium.com/@thotanarendranathreddy/hyperledger-fabric-multi-orgs-multi-nodes-with-kafka-zookeeper-cluster-with-swarm-cluster-5d38be8b1fbc

Q & A

Operations