Deploy Your Network

In this lesson, we will launch an environment on which we will run all our network containers(peer, orderer, CA, state-db)

WE'LL COVER THE FOLLOWING

Folder Structure:

First lets take a look at the folder structure. Here is what's inside the `infra-basic-network` folder:

- **Crypto Config**:
  - `crypto-config.yaml` : This is the input file to `cryptogen` tool for generating the crypto material. This tool has already been run for convenience in our environment and resultant keys and certificates are already added in `crypto-config` folder.
- **Configuration Transaction**:
  - `configtx.yaml` : config file defining input to generate the configuration transaction. This will be consumed by `configtx` executable to generate config transaction in files inside infra-basic-network/config folder.
- **Network Components Containers Definition**:
  - `docker-compose.yml` : This file defines each infrastructure component that will run to form the network. If you look inside this file you will see it defines all the containers using official Hyperledger Fabric

images. The crypto-config folder is mounted as volume on these containers to give them access to relevant keys & certs.

Execution Environment for STEP 1:

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgbIRfBJjk/t3HLnEz
32V4sCllmJtnliVv4UmLfrjZ+B6hRANCAASM01iiFoDgTsTd27nU+R1z7YZbqM4I
Tlz13Mg+SQWsWn25IM6/IwtzNq5SSQZtJwpo7+gtS5IggDn7WJMi6Hy6
-----END PRIVATE KEY-----
```

Steps to Run the Environment:

1. Press the RUN button. Wait for environment to be up and the terminal to show up. Ignore the URL being shown for accessing the app since we will stay in the terminal for now.
   In the usercode directory all the code you see in the widget is mounted. Run the following command as we will run all commands in the infra directory:

```
cd /usercode/infra-basic-network
```

Now we will generate the genesis block and configuration transaction as they are needed before we can bring up the network. Genesis block is the first block in the chain, which contains the channel's initial configuration.

2. Generate genesis block for orderer using the `configtxgen` tool which is already on this environment.

```
/workspace/bin/configtxgen -profile OneOrgOrdererGenesis \
                           -outputBlock ./config/genesis.block
```

This command will generate the `genesis.block` file in config folder using the `configtx.yaml` file as input. The genesis.block file is accessed by the orderer node defined in docker-compose.yaml through a mounted volume.

3. Generate channel configuration transaction

```
/workspace/bin/configtxgen -profile OneOrgChannel \
                           -outputCreateChannelTx ./config/channel.tx \
                           -channelID mychannel
```

4. Generate anchor peer transaction

```
/workspace/bin/configtxgen -profile OneOrgChannel \
                           -outputAnchorPeersUpdate ./config/Org1MSPanchor
s.tx \
                           -channelID mychannel \
                           -asOrg Org1MSP
```

5. Now we will bring up the network using `docker-compose up` command. If you do not have knowledge of `docker` and `docker-compose` it is recommended that you take some tutorials so the docker-compose.yaml file starts to make sense for you. Run the following to launch your containers. On first run, it will download fabric official images from docker hub - be patient as this is one time per environment. If you see an error about docker network, please ignore.

```
docker-compose up -d
```

Ensure all your containers are up by running the following. A total of 5 containers should be running. One of these is the container your terminal is running in. All others are the ones you just launched with your docker-compose up command.

```
docker ps
```

Now all the components are up. But we need to create a channel and join our peer to that channel. If we had multiple peers, we would have to join all of those to this channel using the same step.

6. Create the channel

```
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" \
            -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@o
rg1.example.com/msp" \
       peer0.org1.example.com \
            peer channel create -o orderer.example.com:7050 \
                                -c mychannel \
                                -f /etc/hyperledger/configtx/channel.tx
```

This command executes a command in peer0 container. The command it runs is `peer channel create`. The peer command it again provided by official

is `peer channel create`. The peer command it again provided by official

hyperledger setup. It takes the path to channel configuration transaction(channel.tx) we generated earlier.

7. Join peer0 to the channel.

```
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" \
            -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@o
rg1.example.com/msp" \
       peer0.org1.example.com \
            peer channel join -b mychannel.block
```

If you see a success message at the end, CONGRATULATIONS! Your first fabric network is up. It has no chaincode, yet. But all network components are up and connected and a channel has been formed.

To explore further, you can view logs of any container for example try these commands:

```
docker logs ca.example.com
docker logs peer0.org1.example.com
docker logs orderer.example.com
```

Again, having a good grasp on docker is crucial in understanding what is going on here as fabric setup is heavily based on docker.

In the next chapter, we will write and deploy chaincode.