Assignment on Hyperledger Fabric

Blockchain Network Setup and Application

1. **Generate Crypto Material for 2 Organizations with 2 peers each**

The steps to generate cryptographic material for 2 organizations, each with 2 peers:

1. Install the required tools:
   * Hyperledger Fabric CLI: To generate crypto material, you will need to use the **cryptogen** tool that is included with the Hyperledger Fabric CLI.
   * Docker: You will also need to install Docker to create the required containers.
2. Define the network configuration:
   * Create a YAML file that defines the network topology, including the organizations, peers, and network components.
3. Generate the crypto material:
   * Run the **cryptogen** tool to generate the cryptographic material, such as private keys, public keys, and certificates, for each organization and peer.
   * The generated material will be stored in the **crypto-config** directory.
4. Create the network components:
   * Use Docker to create the required network components, such as orderers and peers, based on the network configuration defined in step 2.
   * Start the containers and initialize the network components.
5. Install the chaincode:
   * Install the chaincode on each peer using the **peer** CLI tool.
6. Start the network:
   * Start the network components, such as the orderers and peers, using the **docker-compose** tool.
7. Test the network:
   * Use the Fabric CLI to interact with the network and test the functionality of the peers.
8. **Generate Genesis file for the generated cryptomaterial.**

To generate the Genesis file for the generated cryptographic material:

1. Define the network configuration:
   * Create a YAML file that defines the network topology, including the organizations, peers, and network components.
2. Generate the crypto material:
   * Run the **cryptogen** tool to generate the cryptographic material, such as private keys, public keys, and certificates, for each organization and peer.
   * The generated material will be stored in the **crypto-config** directory.
3. Create the Genesis block:
   * Use the **configtxgen** tool to create the Genesis block from the network configuration defined in step 1.
   * The **configtxgen** tool generates a Genesis block based on the network configuration and the crypto material generated in step 2.
4. Verify the Genesis block:
   * Verify the Genesis block to make sure that it contains the expected network configuration and the list of initial members.
5. Use the Genesis block to start the network:
   * Use the Genesis block to start the network components, such as the orderers and peers, using the **docker-compose** tool.
6. **Generate channel.tx file with channel name as “mychannel”**

To generate the channel.tx file with channel name "mychannel" for the generated cryptographic material:

1. Define the network configuration:
   * Create a YAML file that defines the network topology, including the organizations, peers, and network components.
2. Generate the crypto material:
   * Run the **cryptogen** tool to generate the cryptographic material, such as private keys, public keys, and certificates, for each organization and peer.
   * The generated material will be stored in the **crypto-config** directory.
3. Create the channel configuration file:
   * Use the **configtxgen** tool to create the channel configuration file, which is used to define the properties of the channel, such as the name, policies, and members.
   * The channel configuration file will be used to create a new channel in the network, or to update the configuration of an existing channel.
4. Generate the channel.tx file:
   * Use the **configtxgen** tool to generate the channel.tx file from the channel configuration file created in step 3.
   * The channel.tx file will contain the channel configuration and the initial list of members (organizations and peers) that are allowed to join the channel.
5. Verify the channel.tx file:
   * Verify the channel.tx file to make sure that it contains the expected channel configuration and the list of members.
6. Use the channel.tx file to create or update the channel:
   * Use the channel.tx file to create a new channel in the network or to update an existing channel using the **peer** CLI tool.
7. **Generate anchor peer tx files.**

To generate anchor peer tx files for each organization in a blockchain network:

1. Define the network configuration:
   * Create a YAML file that defines the network topology, including the organizations, peers, and network components.
2. Generate the crypto material:
   * Run the **cryptogen** tool to generate the cryptographic material, such as private keys, public keys, and certificates, for each organization and peer.
   * The generated material will be stored in the **crypto-config** directory.
3. Create the channel configuration file:
   * Use the **configtxgen** tool to create the channel configuration file, which is used to define the properties of the channel, such as the name, policies, and members.
   * The channel configuration file will be used to create a new channel in the network, or to update the configuration of an existing channel.
4. Generate the channel.tx file:
   * Use the **configtxgen** tool to generate the channel.tx file from the channel configuration file created in step 3.
   * The channel.tx file will contain the channel configuration and the initial list of members (organizations and peers) that are allowed to join the channel.
5. Create the anchor peer tx files:
   * For each organization in the network, use the **configtxgen** tool to create an anchor peer tx file.
   * The anchor peer tx file will contain the configuration for the anchor peer of the organization, which is used to anchor the organization's transactions to the channel.
6. Verify the anchor peer tx files:
   * Verify the anchor peer tx files to make sure that they contain the expected configurations for the anchor peers of each organization.
7. Use the anchor peer tx files to update the channel:
   * Use the anchor peer tx files to update the channel with the configurations for the anchor peers of each organization using the **peer** CLI tool.
8. **Start the Blockchain network with the previously generated artifacts**

to start a blockchain network using the previously generated artifacts:

1. Start the orderer node:
   * Go to the **orderer** directory and run the **start.sh** script to start the orderer node.
   * The orderer node is responsible for maintaining the consensus and ordering the transactions in the network.
2. Start the peer nodes:
   * Go to the **peer** directory and run the **start.sh** script to start each peer node.
   * Each peer node represents an organization in the network and stores a copy of the ledger.
3. Create the channel:
   * Use the **peer** CLI tool to create a new channel using the **channel.tx** file generated in step 4.
   * The channel will define the properties of the channel, such as the name, policies, and members.
4. Join the peer nodes to the channel:
   * Use the **peer** CLI tool to join each peer node to the channel.
   * Each peer node will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
5. Install and instantiate the chaincode:
   * Use the **peer** CLI tool to install the chaincode (smart contract) on each peer node.
   * Use the **peer** CLI tool to instantiate the chaincode on the channel, which will deploy the chaincode to the network and make it executable by the peers.
6. Verify the network:
   * Use the **peer** CLI tool to verify the status of the network, including the peer nodes, orderer nodes, channels, and chaincode.
7. **Create a new channel with the name “mychannel”.**

to create a new channel with the name "mychannel" in a Hyperledger Fabric network:

1. Start the CLI container:
   * Start the CLI container using the **docker-compose up -d cli** command.
   * This will start a Docker container with the CLI tools and a shell environment, allowing you to interact with the network.
2. Create the channel configuration:
   * Create a new channel configuration file with the name "mychannel.tx" using the **configtxgen** tool.
   * This file will define the properties of the channel, such as the name, policies, and members.
3. Generate the channel configuration:
   * Generate the channel configuration using the **configtxgen** tool and the channel configuration file.
   * This will produce a channel configuration block that can be used to create the channel.
4. Create the channel:
   * Use the **peer** CLI tool to create a new channel using the channel configuration block.
   * The channel will be created and broadcast to the network.
5. Join the peer nodes to the channel:
   * Use the **peer** CLI tool to join each peer node to the channel.
   * Each peer node will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
6. **Join the Peers of both organizations into the newly created channel: mychannel**

To join the peers of two organizations into the newly created channel "mychannel" in a Hyperledger Fabric network:

1. Start the CLI container:
   * Start the CLI container using the **docker-compose up -d cli** command.
   * This will start a Docker container with the CLI tools and a shell environment, allowing you to interact with the network.
2. Join the first organization's peers to the channel:
   * Use the **peer** CLI tool to join the first organization's peers to the channel "mychannel".
   * The peers will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
3. Join the second organization's peers to the channel:
   * Use the **peer** CLI tool to join the second organization's peers to the channel "mychannel".
   * The peers will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
4. **Join the anchor peers of both organizations into the newly created channel:mychannel**

To join the anchor peers of two organizations into the newly created channel "mychannel" in a Hyperledger Fabric network:

1. Start the CLI container:
   * Start the CLI container using the **docker-compose up -d cli** command.
   * This will start a Docker container with the CLI tools and a shell environment, allowing you to interact with the network.
2. Join the first organization's anchor peer to the channel:
   * Use the **peer** CLI tool to join the first organization's anchor peer to the channel "mychannel".
   * The anchor peer will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
3. Join the second organization's anchor peer to the channel:
   * Use the **peer** CLI tool to join the second organization's anchor peer to the channel "mychannel".
   * The anchor peer will receive a copy of the channel's ledger and be able to participate in transactions on the channel.
4. **Install the chaincode on all organization peers**

To install chaincode on all peers of two organizations in a Hyperledger Fabric network:

1. Start the CLI container:
   * Start the CLI container using the **docker-compose up -d cli** command.
   * This will start a Docker container with the CLI tools and a shell environment, allowing you to interact with the network.
2. Install the chaincode on the first organization's peers:
   * Use the **peer** CLI tool to install the chaincode on each of the first organization's peers.
   * The chaincode will be stored on each peer's local file system and be available for instantiation and execution on the channel.
3. Install the chaincode on the second organization's peers:
   * Use the **peer** CLI tool to install the chaincode on each of the second organization's peers.
   * The chaincode will be stored on each peer's local file system and be available for instantiation and execution on the channel.
4. **Instantiate the chaincode on peer0 of 1st organization**

****

set the environment variables for the channel and the chaincode:

export CHANNEL\_NAME=<mychannel>

export CHAINCODE\_NAME=<chaincode>

export CHAINCODE\_VERSION=<chaincode\_version>

peer chaincode install -n $CHAINCODE\_NAME -v $CHAINCODE\_VERSION -p <path\_to\_chaincode>

peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n $CHAINCODE\_NAME -v $CHAINCODE\_VERSION -c '{"Args":["<your\_init\_arg1>","<your\_init\_arg2>", ...]}' -P "OR ('Org1MSP.peer')"

1. **Perform a Query operation on the Blockchain network to retrieve the balance of A&B accounts**
2. First, navigate to the CLI container:



1. Then, set the environment variables for the channel and the chaincode:

export CHANNEL\_NAME=<mychannel>

export CHAINCODE\_NAME=<chaincode\_name>

1. Next, use the following command to retrieve the balance of A's account:



1. Use the following command to retrieve the balance of B's account:



1. **Invoke a transaction in Blockchain network for transferring the balance from one account to other.**

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n $CHAINCODE\_NAME -c '{"Args":["transfer","A","B","<amount>"]}'

1. **Perform a Query operation on the Blockchain network to retrieve the updated balance of A&B accounts**





**14. Stop the Blockchain network**

1. **​** Exit the CLI container:

****

1. Use the following command to stop all the running containers:

****

1. Use the following command to remove all the stopped containers:

****