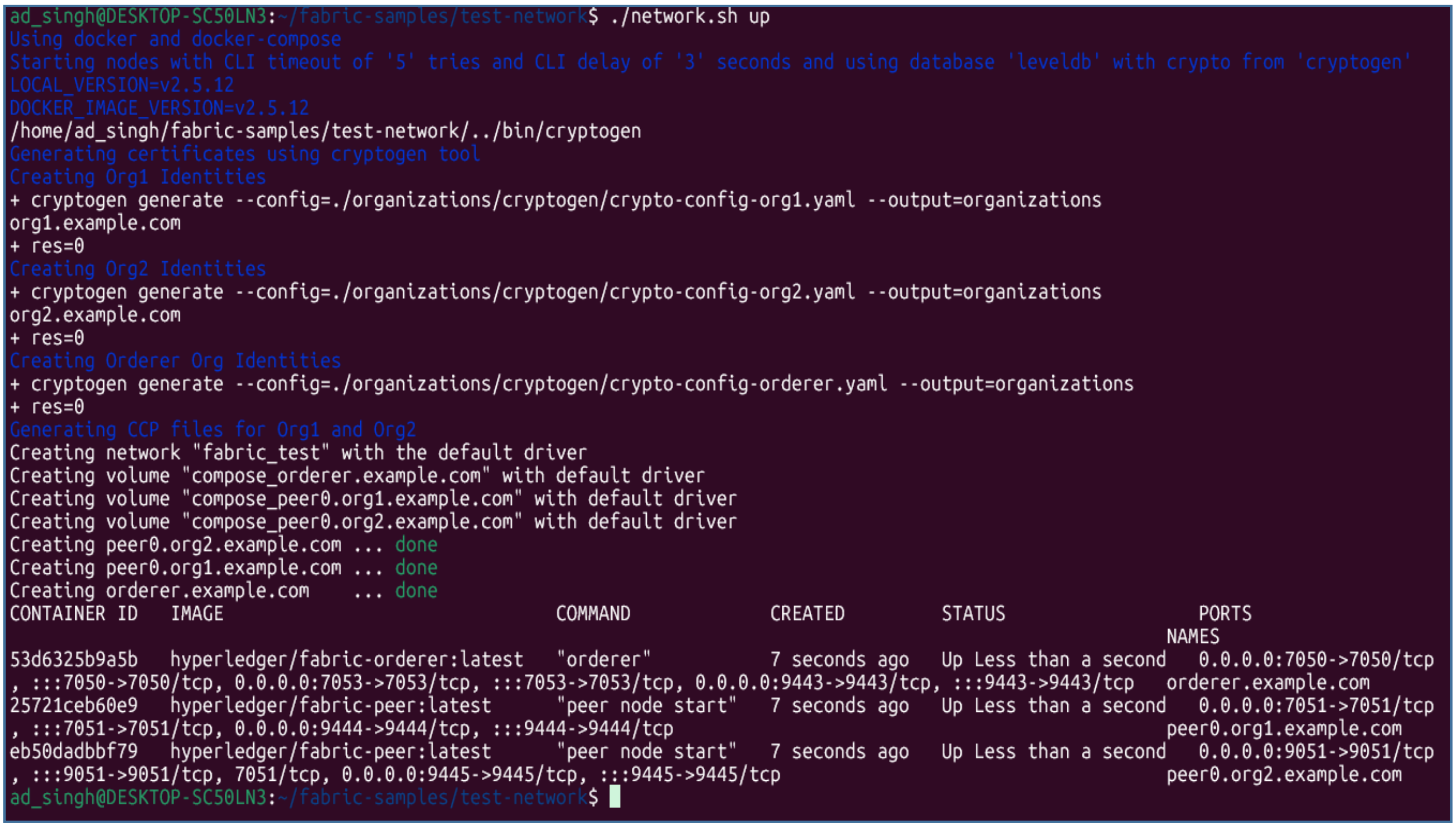
Fabric Network Setup



```
ad_singh@DESKTOP-SC50LN3:~/fabric-samples/test-network$ ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=v2.5.12
DOCKER_IMAGE_VERSION=v2.5.12
/home/ad_singh/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating orderer.example.com    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED         STATUS                  PORTS
                                                                                                              NAMES
53d6325b9a5b   hyperledger/fabric-orderer:latest   "orderer"           7 seconds ago   Up Less than a second   0.0.0.0:7050->7050/tcp
, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
25721ceb60e9   hyperledger/fabric-peer:latest      "peer node start"   7 seconds ago   Up Less than a second   0.0.0.0:7051->7051/tcp
, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                                peer0.org1.example.com
eb50dadbbf79   hyperledger/fabric-peer:latest      "peer node start"   7 seconds ago   Up Less than a second   0.0.0.0:9051->9051/tcp
, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                      peer0.org2.example.com
ad_singh@DESKTOP-SC50LN3:~/fabric-samples/test-network$
```

1. Setting Environment Variables and Checking Versions:

- **`Using docker and docker-compose`**: This confirms that the script is utilizing Docker and Docker Compose to manage the Fabric network components.
- **`Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'`**: This line indicates the configuration parameters that the script intends to use for CLI interactions, the database for the peers' ledgers (LevelDB), and the tool for generating cryptographic material (`cryptogen`). Keep in mind that while these parameters are mentioned, the actual implementation of the timeout and delay might be within the `network.sh` script itself.
- **`LOCAL_VERSION=v2.5.12`**: This shows the version of the Hyperledger Fabric binaries that are present in your `fabric-samples/bin` directory.
- **`DOCKER_IMAGE_VERSION=v2.5.12`**: This indicates the version of the Hyperledger Fabric Docker images that the script will attempt to use. It's good that this matches your local binary version, as it helps prevent compatibility issues.

2. Generating Cryptographic Material:

- **`/home/ad_singh/fabric-samples/test-network/../bin/cryptogen`**: This shows the path to the `cryptogen` tool, which is used to generate the necessary certificates and keys for the organizations and nodes in the network.
- **`Generating certificates using cryptogen tool`**: This confirms that the script is now using `cryptogen`.
- **`Creating Org1 Identities`**: The script is generating the cryptographic identities (certificates and private keys) for the first organization (`Org1`).
- **`+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations`**: This is the actual `cryptogen` command being executed. It uses the configuration file `crypto-config-org1.yaml` to define the structure and number of nodes for Org1 and outputs the generated material to the `organizations` directory.
- **`org1.example.com`**: This likely indicates the domain name associated with Org1.
- **`+ res=0`**: This signifies that the previous command (`cryptogen generate ...`) executed successfully (return code 0).

`Generating CCP files for Org1 and Org2`: The script is now creating Connection Profile (CCP) files. These files contain the necessary information for client applications and SDKs to connect to the peers and orderers within each organization.

3. Setting Up the Docker Network and Volumes:

- **`Creating network "fabric_test" with the default driver`**: Docker is creating a network named `fabric_test`. This network allows the different Docker containers in your Fabric network to communicate with each other.
- **`Creating volume "compose_orderer.example.com" with default driver`**: Docker is creating a named volume for the orderer node. Volumes are used to persist data even when containers are stopped or removed. This volume will likely store the orderer's ledger and configuration.
- **`Creating volume "compose_peer0.org1.example.com" with default driver`**: A named volume for the first peer (`peer0`) of the first organization (`Org1`). This will store its ledger and other data.
- **`Creating volume "compose_peer0.org2.example.com" with default driver`**: A named volume for the first peer (`peer0`) of the second organization (`Org2`), for its data persistence.

4. Creating and Starting Docker Containers:

- **`Creating peer0.org1.example.com ... done`**: Docker Compose is creating the container for the first peer of Org1. The configuration for this container is likely defined in a `docker-compose.yaml` file within the `test-network` directory.
- **`Creating orderer.example.com ... done`**: Docker Compose is creating the container for the orderer node.
- **`Creating peer0.org2.example.com ... done`**: Docker Compose is creating the container for the first peer of Org2.

5. Listing Running Docker Containers:

The final section shows the currently running Docker containers:

- **`CONTAINER ID`**: A unique identifier for each running container.
- **`IMAGE`**: The Docker image used to create the container (e.g., `hyperledger/fabric-peer:latest`, `hyperledger/fabric-orderer:latest`).
- **`COMMAND`**: The command that is being executed inside the container when it starts (e.g., `"peer node start"` for peers, `"orderer"` for the orderer).
- **`CREATED`**: How long ago the container was created.
- **`STATUS`**: The current status of the container (e.g., "Up Less than a second").
- **`PORTS`**: The port mappings between the host machine and the container. For example:
  - `0.0.0.0:7051->7051/tcp, :::7051->7051/tcp`: Maps port 7051 on your host (both IPv4 and IPv6) to port 7051 inside the `peer0.org1.example.com` container. This is the default gRPC port for the peer.
  - `0.0.0.0:9444->9444/tcp, :::9444->9444/tcp`: Maps port 9444 on your host to port 9444 inside the `peer0.org1.example.com` container (used for event listening).

  - Similar port mappings exist for `peer0.org2.example.com` (using ports 9051 and 9445) and the `orderer.example.com` (using ports 7050, 7053, and 9443).
- **`NAMES`**: The assigned names to the Docker containers (`peer0.org1.example.com`, `orderer.example.com`, `peer0.org2.example.com`).