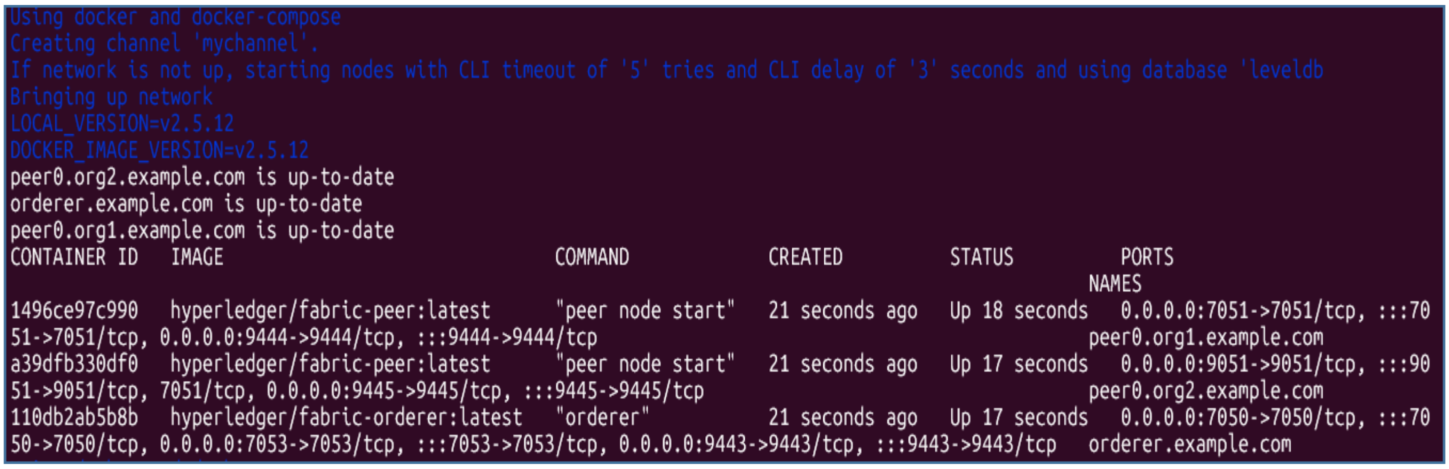
Fabric createChannels

1. Initial Setup and Network Check:

- `Using docker and docker-compose`: Confirms the use of Docker and Docker Compose.
- `Creating channel 'mychannel'.`: Indicates the goal of this script execution.
- `If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'`: This is a conditional statement. Since the network was already running from your previous execution, it skipped the network startup phase.
- `Bringing up network`: Even though the network was running, this line might still be printed as part of the script's flow.
- `LOCAL_VERSION=v2.5.12` and `DOCKER_IMAGE_VERSION=v2.5.12`: Re-verification of the versions, confirming they are consistent.
- `peer0.org2.example.com is up-to-date`, `orderer.example.com is up-to-date`, `peer0.org1.example.com is up-to-date`: Docker Compose checked the status of the existing containers and found them already running with the desired images.
- **The `docker ps` output**: This confirms that the three essential containers (two peers and one orderer) are running.

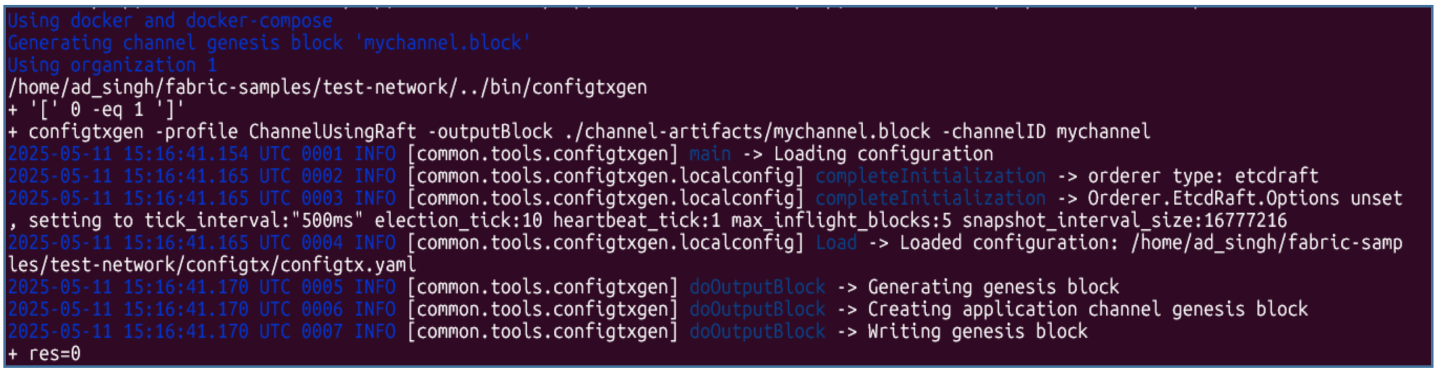


```
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb
Bringing up network
LOCAL_VERSION=v2.5.12
DOCKER_IMAGE_VERSION=v2.5.12
peer0.org2.example.com is up-to-date
orderer.example.com is up-to-date
peer0.org1.example.com is up-to-date
CONTAINER ID   IMAGE                               COMMAND             CREATED          STATUS          PORTS
                                                                                                      NAMES
1496ce97c990   hyperledger/fabric-peer:latest      "peer node start"   21 seconds ago   Up 18 seconds   0.0.0.0:7051->7051/tcp, :::70
51->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                               peer0.org1.example.com
a39dfb330df0   hyperledger/fabric-peer:latest      "peer node start"   21 seconds ago   Up 17 seconds   0.0.0.0:9051->9051/tcp, :::90
51->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                     peer0.org2.example.com
110db2ab5b8b   hyperledger/fabric-orderer:latest   "orderer"           21 seconds ago   Up 17 seconds   0.0.0.0:7050->7050/tcp, :::70
50->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
```

2. Generating the Channel Genesis Block:

- `Using docker and docker-compose`: Another confirmation.
- `Generating channel genesis block 'mychannel.block'`: This is a crucial step in creating a new channel. The genesis block is the first block of the channel's ledger and contains the initial configuration of the channel.
- `Using organization 1`: This likely indicates which organization's MSP (Membership Service Provider) information is being used at this point in the configuration generation.
- `/home/ad_singh/fabric-samples/test-network/../bin/configtxgen`: The path to the `configtxgen` tool, which is used to generate configuration artifacts like the genesis block.
- `+ '[' 0 -eq 1 ']'`: A shell conditional check (in this case, it's true, so the subsequent block is skipped).
- `+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel`: This is the core command that generates the genesis block.

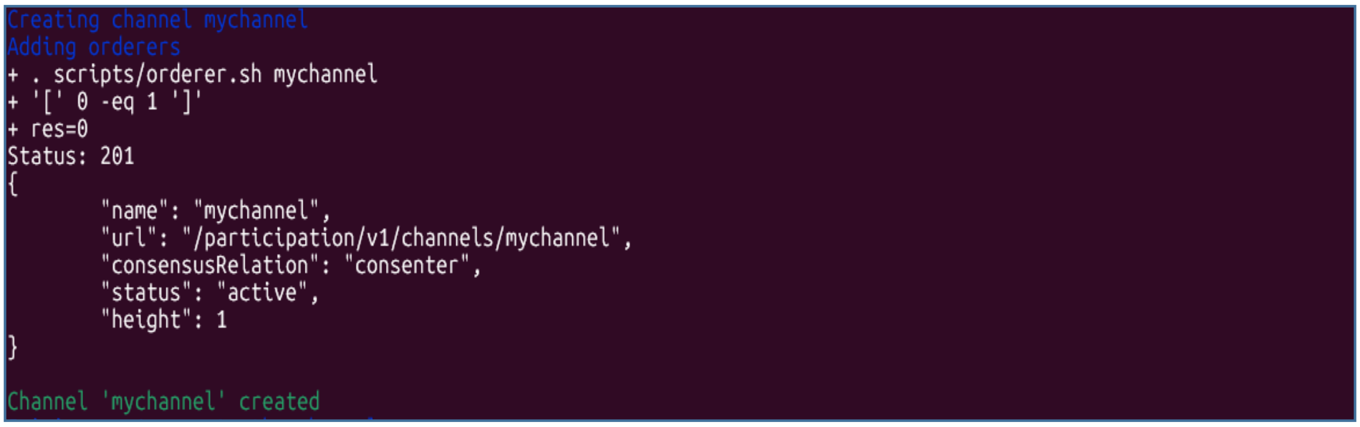
- `-profile ChannelUsingRaft`: Specifies the configuration profile to use from the `configtx/configtx.yaml` file. This profile defines the initial settings for a channel using the Raft consensus mechanism for the ordering service.
- `-outputBlock ./channel-artifacts/mychannel.block`: Specifies the output file path for the generated genesis block
- `-channelID mychannel`: Sets the ID of the channel being created.
- **The subsequent `configtxgen` logs**: These logs from the `configtxgen` tool show the process of loading the configuration, initializing the orderer type (etcdraft), setting default Raft options, loading the `configtx.yaml` file, and finally generating and writing the genesis block.
- **`+ res=0`**: Indicates successful execution of the `configtxgen` command.



```
Using docker and docker-compose
Generating channel genesis block 'mychannel.block'
Using organization 1
/home/ad_singh/fabric-samples/test-network/../bin/configtxgen
+ '[' 0 -eq 1 ']'
+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2025-05-11 15:16:41.154 UTC 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2025-05-11 15:16:41.165 UTC 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2025-05-11 15:16:41.165 UTC 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset
, setting to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2025-05-11 15:16:41.165 UTC 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /home/ad_singh/fabric-samp
les/test-network/configtx/configtx.yaml
2025-05-11 15:16:41.170 UTC 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2025-05-11 15:16:41.170 UTC 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2025-05-11 15:16:41.170 UTC 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
```

3. Creating the Channel on the Orderer:

- **`Creating channel mychannel`**: Indicates the next step.
- **`Adding orderers`**: This step involves informing the ordering service about the new channel.
- **`+ . scripts/orderer.sh mychannel`**: This likely executes a script (`orderer.sh`) to interact with the orderer and create the channel.
- **`+ '[' 0 -eq 1 ']'`**: Another shell conditional check.
- **`+ res=0`**: Successful execution of the `orderer.sh` script.
- **The JSON response**: This is the orderer's response confirming the successful creation of the channel:
  - `"name": "mychannel"`: The name of the created channel.
  - `"url": "/participation/v1/channels/mychannel"`: An API endpoint related to the channel (this might be specific to newer Fabric versions and the orderer's participation API).
  - `"consensusRelation": "consenter"`: Indicates this channel uses the consenter role within the Raft ordering service.
  - `"status": "active"`: The channel is now active.
  - `"height": 1`: The current height of the channel's ledger (only the genesis block exists initially).
- **`Channel 'mychannel' created`**: A user-friendly confirmation message.

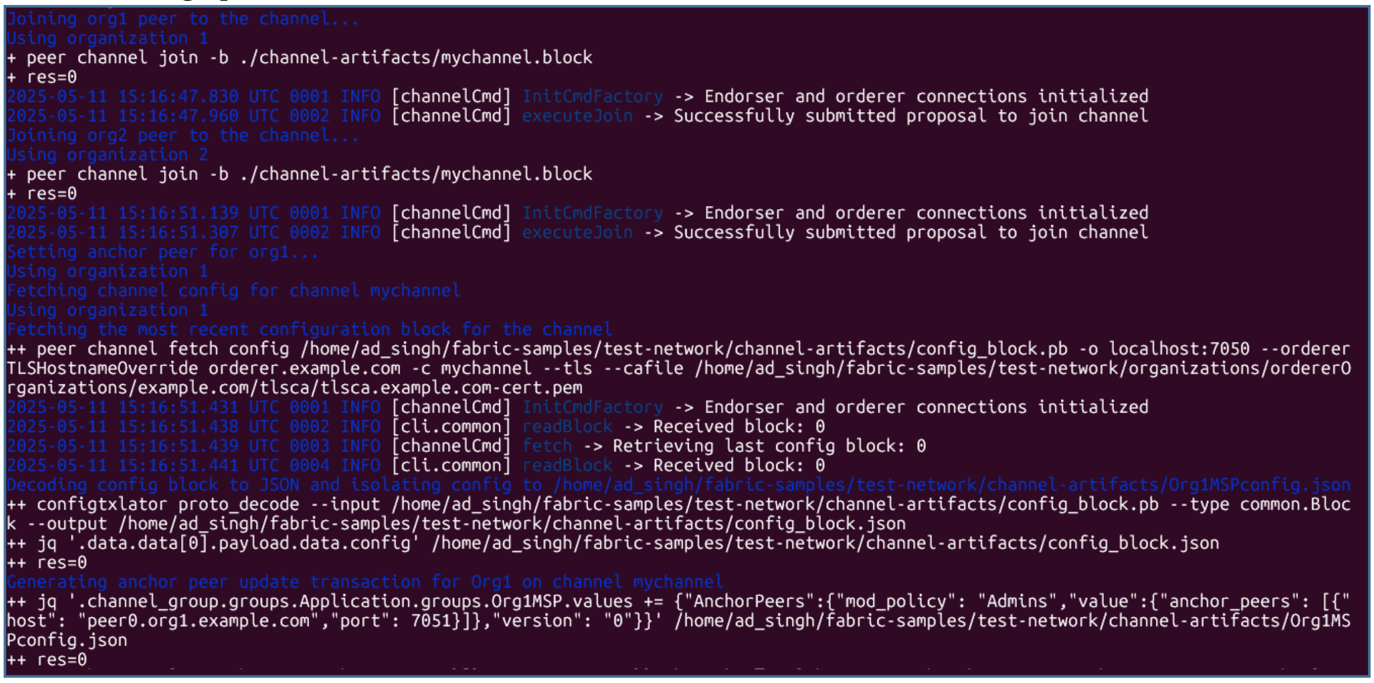


```
Creating channel mychannel
Adding orderers
+ . scripts/orderer.sh mychannel
+ '[' 0 -eq 1 ']'
+ res=0
Status: 201
{
        "name": "mychannel",
        "url": "/participation/v1/channels/mychannel",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'mychannel' created
```

4. Joining Peers to the Channel:

- **`Joining org1 peer to the channel...`**: The script now instructs the peer of the first organization (`peer0.org1.example.com`) to join the newly created channel.
- **`Using organization 1`**: Indicates the context for the following peer command.
- **`+ peer channel join -b ./channel-artifacts/mychannel.block`**: This is the `peer` CLI command executed by the script.
    - `peer channel join`: The command to join a peer to a channel.
    - `-b ./channel-artifacts/mychannel.block`: Specifies the genesis block of the channel to join. The peer needs this initial configuration to understand the channel.
- **`+ res=0`**: Successful execution of the `peer channel join` command for the Org1 peer.
- **The subsequent `peer` CLI logs**: These logs confirm that the peer successfully submitted a proposal to the orderer to join the channel.
- **`Joining org2 peer to the channel...`**: The same process is now repeated for the peer of the second organization (`peer0.org2.example.com`).
- **`Using organization 2`**: Indicates the context for the following peer command.
- **`+ peer channel join -b ./channel-artifacts/mychannel.block`**: The `peer channel join` command for the Org2 peer.
- **`+ res=0`**: Successful execution of the `peer channel join` command for the Org2 peer.
- **The subsequent `peer` CLI logs**: Confirmation of the successful join proposal for the Org2 peer.

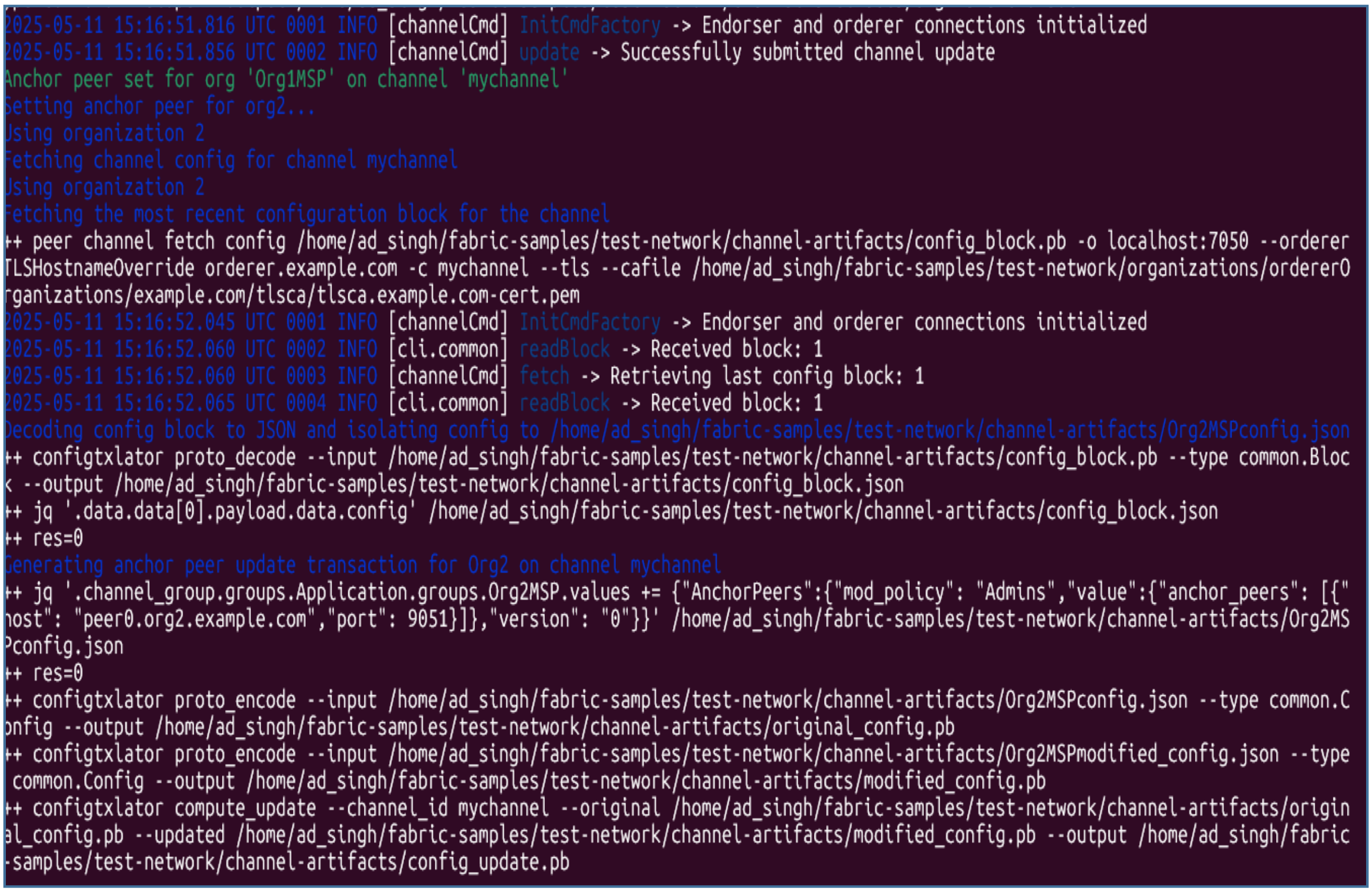


```
Joining org1 peer to the channel...
Using organization 1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2025-05-11 15:16:47.830 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:47.960 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org2 peer to the channel...
Using organization 2
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2025-05-11 15:16:51.139 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:51.307 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Setting anchor peer for org1...
Using organization 1
Fetching channel config for channel mychannel
Using organization 1
Fetching the most recent configuration block for the channel
++ peer channel fetch config /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.pb -o localhost:7050 --orderer
TLSHostnameOverride orderer.example.com -c mychannel --tls --cafile /home/ad_singh/fabric-samples/test-network/organizations/ordererO
rganizations/example.com/tlsca/tlsca.example.com-cert.pem
2025-05-11 15:16:51.431 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:51.438 UTC 0002 INFO [cli.common] readBlock -> Received block: 0
2025-05-11 15:16:51.439 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config block: 0
2025-05-11 15:16:51.441 UTC 0004 INFO [cli.common] readBlock -> Received block: 0
Decoding config block to JSON and isolating config to /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org1MSPconfig.json
++ configtxlator proto_decode --input /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.pb --type common.Bloc
k --output /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.json
++ jq '.data.data[0].payload.data.config' /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.json
++ res=0
Generating anchor peer update transaction for Org1 on channel mychannel
++ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"
host": "peer0.org1.example.com","port": 7051}]},"version": "0"}}' /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org1MS
Pconfig.json
++ res=0
```

5. Setting Anchor Peers:

- **`Setting anchor peer for org1...`**: Anchor peers are crucial for enabling gossip communication between peers of different organizations on a channel. They serve as entry points for inter-organization communication.
- **`Using organization 1`**: Context for the following operations.

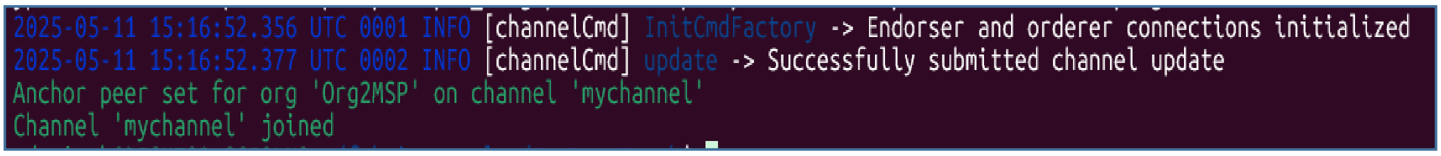
- **`Fetching channel config for channel mychannel`**: To update the channel configuration with the anchor peer information, the current configuration needs to be retrieved.
- **`Using organization 1`**: Context.
- **`Fetching the most recent configuration block for the channel`**: The script fetches the latest configuration block (which is currently just the genesis block).
- **The `peer channel fetch config` command and logs**: This command retrieves the configuration block.
- **Decoding and Isolating Configuration**: The script uses `configtxlator` to decode the configuration block from its protocol buffer format to JSON and then uses `jq` to extract the configuration relevant to Org1's MSP.
- **Generating Anchor Peer Update Transaction**: `jq` is used to modify the Org1 MSP configuration to include the anchor peer information (in this case, `peer0.org1.example.com:7051`).
- **Encoding and Computing Configuration Update**: `configtxlator` is used to encode the original and modified configurations back into protocol buffer format and then compute the difference (the update).
- **Creating Configuration Update Envelope**: The configuration update is wrapped in an envelope that can be submitted as a transaction to the orderer.
- **Submitting Channel Update**: The `peer channel update` command sends the configuration update transaction to the orderer.
- **`Anchor peer set for org 'Org1MSP' on channel 'mychannel'`**: Confirmation that the anchor peer for Org1 has been successfully set.
- **The same process is then repeated for `org2`**, setting its anchor peer to `peer0.org2.example.com:9051`.
- **`Anchor peer set for org 'Org2MSP' on channel 'mychannel'`**: Confirmation for Org2.



```
2025-05-11 15:16:51.816 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:51.856 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org1MSP' on channel 'mychannel'
Setting anchor peer for org2...
Jsing organization 2
Fetching channel config for channel mychannel
Jsing organization 2
Fetching the most recent configuration block for the channel
++ peer channel fetch config /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.pb -o localhost:7050 --orderer
TLSHostnameOverride orderer.example.com -c mychannel --tls --cafile /home/ad_singh/fabric-samples/test-network/organizations/orderer0
rganizations/example.com/tlsca/tlsca.example.com-cert.pem
2025-05-11 15:16:52.045 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:52.060 UTC 0002 INFO [cli.common] readBlock -> Received block: 1
2025-05-11 15:16:52.060 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config block: 1
2025-05-11 15:16:52.065 UTC 0004 INFO [cli.common] readBlock -> Received block: 1
Decoding config block to JSON and isolating config to /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org2MSPconfig.json
++ configtxlator proto_decode --input /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.pb --type common.Bloc
k --output /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.json
++ jq '.data.data[0].payload.data.config' /home/ad_singh/fabric-samples/test-network/channel-artifacts/config_block.json
++ res=0
Generating anchor peer update transaction for Org2 on channel mychannel
++ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"
host": "peer0.org2.example.com","port": 9051}]},"version": "0"}}' /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org2MS
Pconfig.json
++ res=0
++ configtxlator proto_encode --input /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org2MSPconfig.json --type common.C
onfig --output /home/ad_singh/fabric-samples/test-network/channel-artifacts/original_config.pb
++ configtxlator proto_encode --input /home/ad_singh/fabric-samples/test-network/channel-artifacts/Org2MSPmodified_config.json --type
 common.Config --output /home/ad_singh/fabric-samples/test-network/channel-artifacts/modified_config.pb
++ configtxlator compute_update --channel_id mychannel --original /home/ad_singh/fabric-samples/test-network/channel-artifacts/origin
al_config.pb --updated /home/ad_singh/fabric-samples/test-network/channel-artifacts/modified_config.pb --output /home/ad_singh/fabric
-samples/test-network/channel-artifacts/config_update.pb
```

6. Final Confirmation:

- **`Channel 'mychannel' joined`**: A final message indicating that the channel creation and joining process is complete.



```
2025-05-11 15:16:52.356 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-11 15:16:52.377 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```