- **Modular** - Hyperledger is developing modular, extensible frameworks with common building blocks that can be reused.
- **Highly secure** - Security is a key consideration for distributed ledgers, especially since many use cases involve high-value transactions or sensitive data.
- **Cryptocurrency Agnostic** - Hyperledger projects are independent and agnostic of all altcoins, cryptocurrencies.
- **Complete with APIs** - All Hyperledger projects provide rich and easy-to-use APIs that support interoperability with other systems.
- **Interoperable**- In the future, many different Blockchain networks will need to communicate and exchange data to form more complex and powerful networks.

- Hyperledger Fabric is a enterprise Blockchain framework for developing Blockchain with a modular architecture.
- It is private and permissioned, not a permission less where any participant can join the network .
- Allows participant access to fabric network using Membership Service Provider (MSP).
- Ability to create a separate chains(Ledger) within fabric network using channels.
- Execute ->Order(Sequencing) -> validate-commit rather that validate -> order (Sequencing)-> Execute transaction flow whch brings the scalability and

The Hyperledger Fabric architecture delivers the following advantages:

- **Modularity**: Modular and pluggable consensus mechanism, Membership services & data Storage
- **Scalability**: Fabric architecture provide the scalable Blockchain via separating roles & responsibilities of node.
- **Confidentiality**: Privacy and confidentiality in transactions using channels & side DB
- **Trust & Flexibility**: Smart contract runs outside the control of network node(peer & orderer).
- **Performance** : Transaction processing follows : Execution-> ordering -> validation & commitment which brings the high transaction throughput/performance via separating responsivities of nodes.
- **Network Governance** : There is no centralized governance, consortium based shared governance model

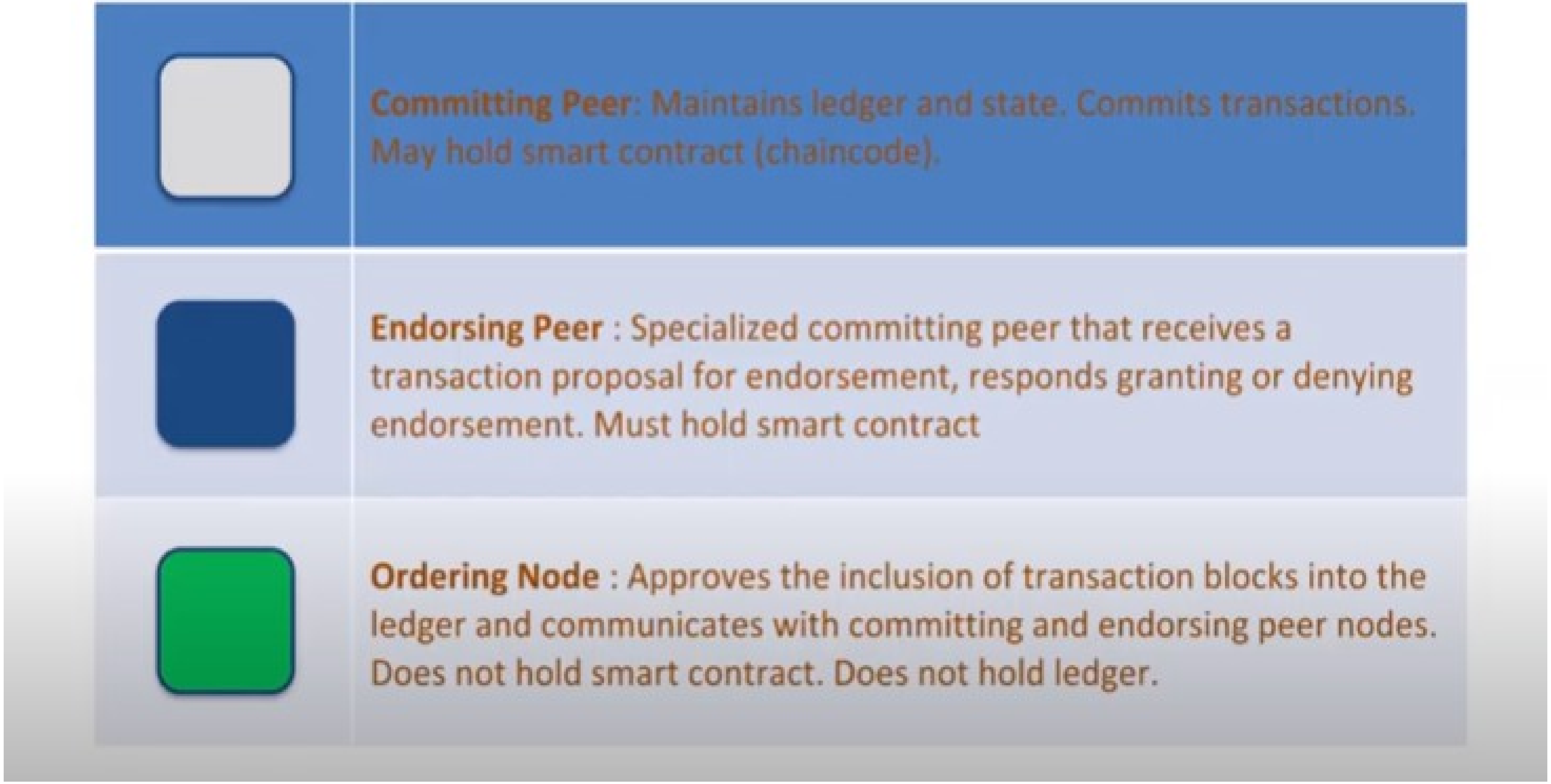
- **Peer**: Peer is participant in fabric network which hold the ledger and world state. They are different kind of peer like endorsing and committing peer.

- **Organization** : An organization can be as large as a multinational corporation or as small as an individual. The transaction endpoint of an organization is a Peer.

- **Ordering Node**: Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

- **MSP** : The Membership Service Provider (MSP) provides credentials to clients, and peers for them to participate in a Hyperledger Fabric network.

- **Fabric-Certificate Authority** : In general, Certificate Authorities manage enrollment certificates for a permissioned Blockchain. Fabric-CA is the default certificate authority for Hyperledger Fabric

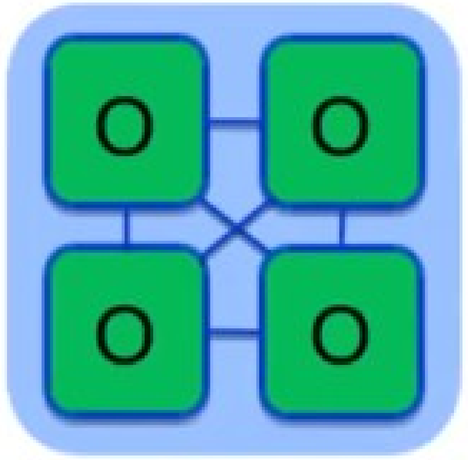


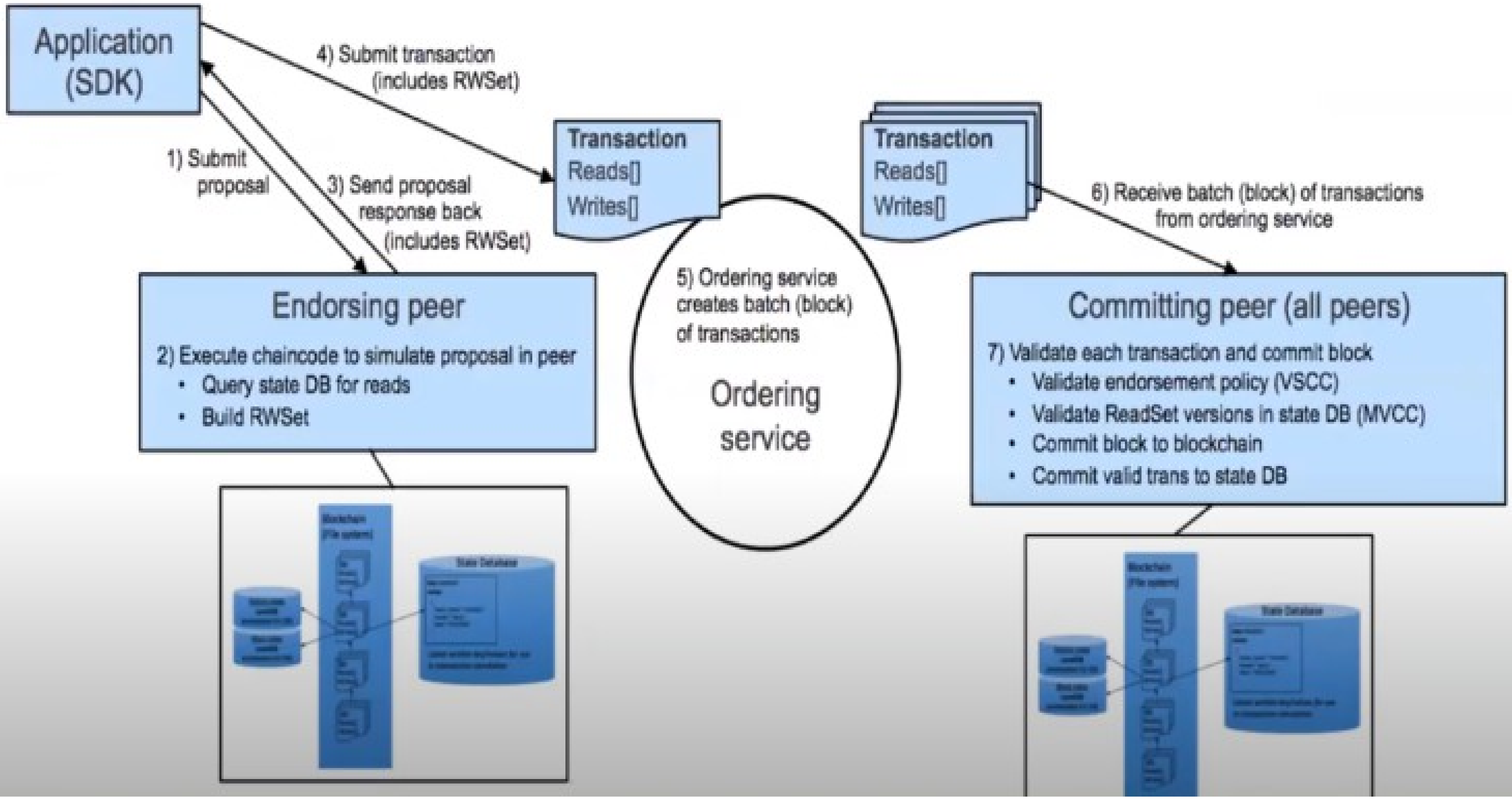
⬜	**Committing Peer**: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
🟦	**Endorsing Peer** : Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
🟩	**Ordering Node** : Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

The ordering service packages transactions into blocks to be delivered to peers.

Different configuration options for the ordering service includes:

1. SOLO - Single node for development
2. Kafka - Crash fault tolerant consensus
3. Raft - Crash fault tolerant consensus




Application (SDK)
4) Submit transaction (includes RWSet)
1) Submit proposal
3) Send proposal response back (includes RWSet)
Transaction
Reads[]
Writes[]
Transaction
Reads[]
Writes[]
6) Receive batch (block) of transactions from ordering service
Endorsing peer
2) Execute chaincode to simulate proposal in peer
• Query state DB for reads
• Build RWSet
5) Ordering service creates batch (block) of transactions
Ordering service
Committing peer (all peers)
7) Validate each transaction and commit block
• Validate endorsement policy (VSCC)
• Validate ReadSet versions in state DB (MVCC)
• Commit block to blockchain
• Commit valid trans to state DB
Blockchain (File system)
State Database
Blockchain (File system)
State Database

All you Need to Know as of..

Anchor Peer

This gets used for communications between organisations. It makes peers in different organisations aware of each other.

Blocks

Consist of a header, block data (transactions) and block metadata (information about nodes involved with creating the block).

Certificate Authorities

Everyone who wants to interact with the networks needs an identity. The CA provides the means for each actor to have a verifiable digital identity. Hyperledger Fabric has a built in CA component for use in the blockchain network.

Chaincode

The Hyperledger Fabric term for a smart contract. Note that chaincode does not have to be installed on every peer in a channel.

Channel

A channel allows a group of participants to create a separate ledger of transactions. The transactions are only visible to the members of the channel.

Channel Configuration

Rules that govern the channel, the channel is governed by the channel members. The channel configuration is separate from the network configuration.

Consortium

A group of organisations that share a need to transact.

Committing Peer

Every peer in the channel.

Endorsing Peer

Every peer that has the smart contract installed can be an endorsing peer.

Endorsement Policy

The rules for which organisations much approve a transaction before the other organisations will accept a copy. This is specific to the chaincode.

Leader Peer

An organization can have multiple peers in a channel. Only one peer from the organization needs to receive the transactions. The leader distributes transactions from the orderer.

Membership Service Provider

Is a trusted authority.

The MSP identifies which Root Certificate Authorities (CA) and Intermediate CA’s are trusted by the network. The MSP identifies what roles different actors in an organization can play in the network.

Nodes join the network through a Membership Service Provider.

Ledger

This is an append only file while can be used to recreate the world state.

Ordering Nodes

Is like a network administration point. The ordering nodes support the application channels for ordering transactions into blocks.

Peer Nodes

Each peer maintains a copy of the ledger for each channel it is a member of.

Policies

These determine who has control over the network configuration.

Private Data Collection

This is used for keeping the data in a transaction confidential. The data is stored in a private database that is separated from the channel ledger.

Public Key Infrastructure

This provides secure communication in a network. CAs issue digital certificates that get used to authenticate messages in the network. The PKI provides a list of identities and the MSP says which of them are part of an organization.

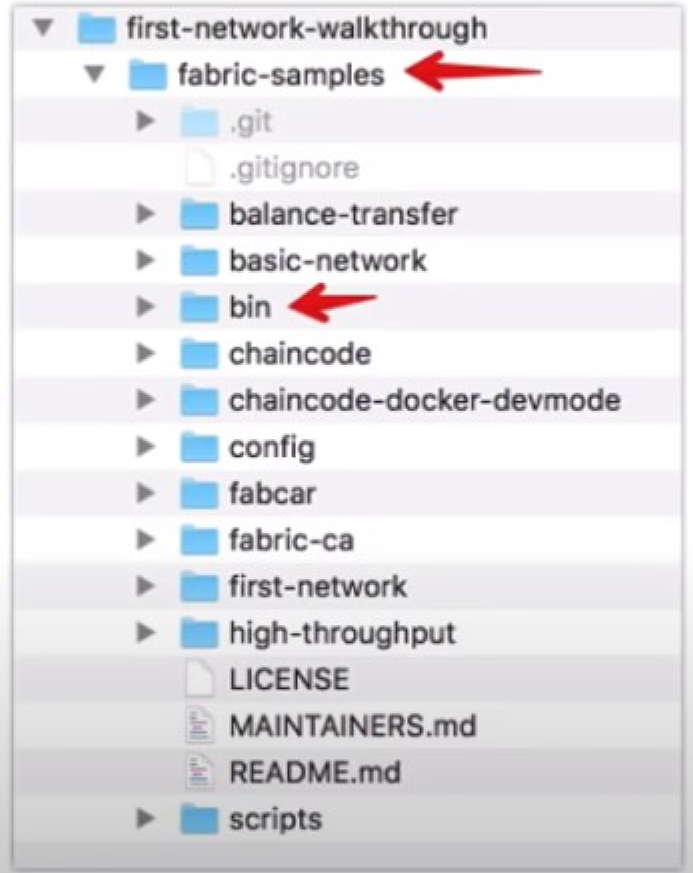
System Chaincode

Is code that defines operating parameters for the entire channel.

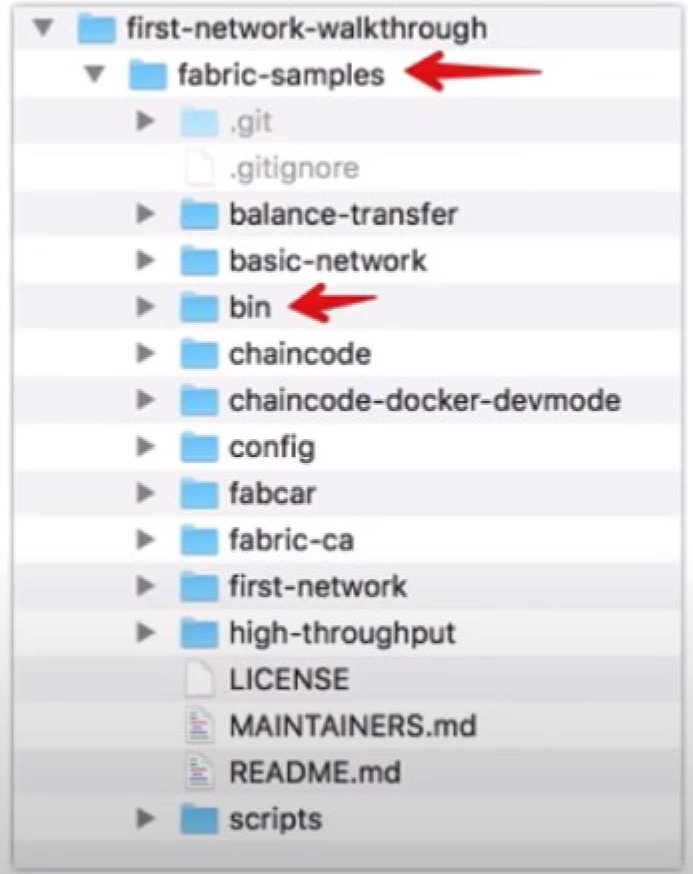
Lifecycle and configuration system chaincode defines the rules for the channel. Endorsement and validation system chaincode defines the requirements for endorsing and validating transactions.

World State

Is a snapshot of the current state of the objects in the network, this is usually a graph database in practise.



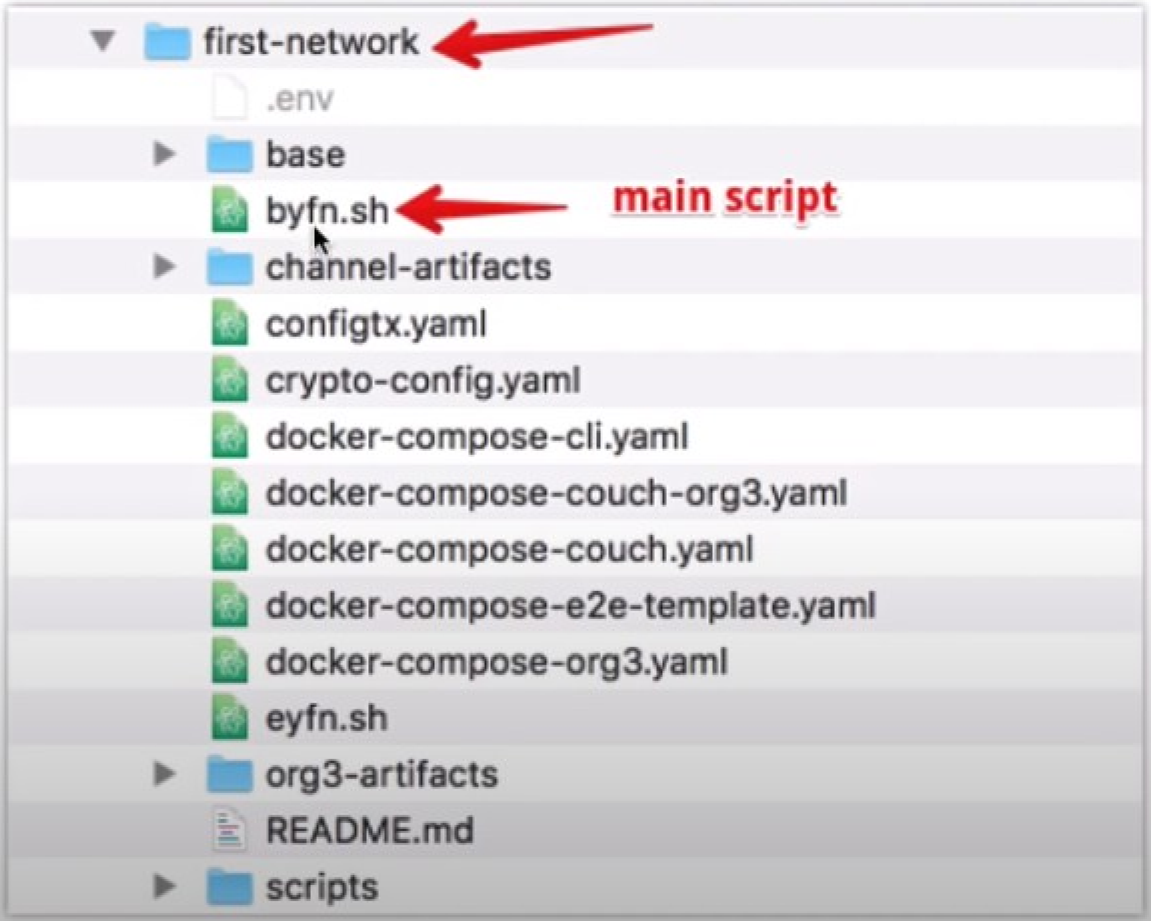
- fabric-samples downloaded
- binary tools installed in /bin
- Docker Images downloaded



```
▼ first-network-walkthrough
    ▼ fabric-samples
        ▶ .git
          .gitignore
        ▶ balance-transfer
        ▶ basic-network
        ▶ bin
        ▶ chaincode
        ▶ chaincode-docker-devmode
        ▶ config
        ▶ fabcar
        ▶ fabric-ca
        ▶ first-network
        ▶ high-throughput
          LICENSE
          MAINTAINERS.md
          README.md
        ▶ scripts
```

- fabric-samples downloaded
- binary tools installed in /bin
- Docker Images downloaded

First network tutorial

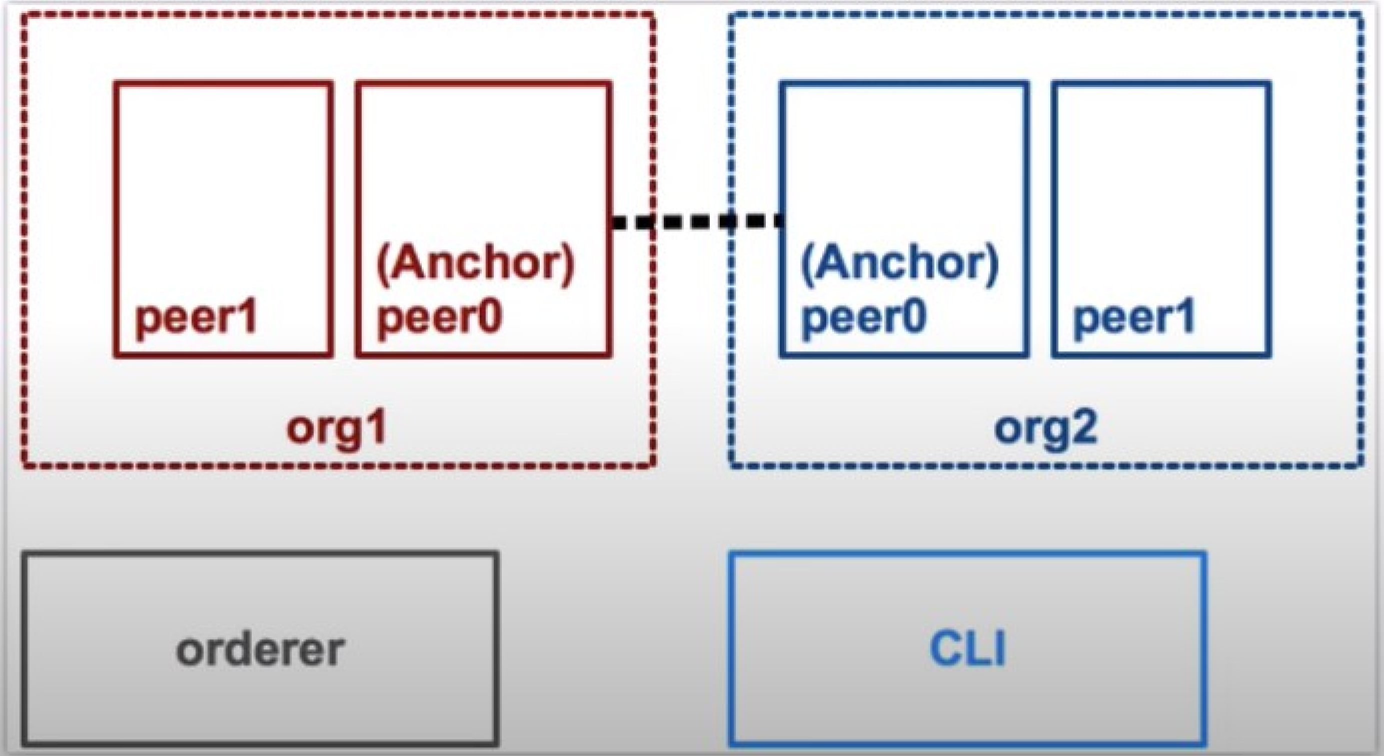


```
▼ first-network
    .env
  ▶ base
    byfn.sh
  ▶ channel-artifacts
    configtx.yaml
    crypto-config.yaml
    docker-compose-cli.yaml
    docker-compose-couch-org3.yaml
    docker-compose-couch.yaml
    docker-compose-e2e-template.yaml
    docker-compose-org3.yaml
    eyfn.sh
  ▶ org3-artifacts
    README.md
  ▶ scripts
```

- main script **`byfn.sh`**
- well documented and worth reading through
- use this to generate cryptographic and network artefacts, bring up the network & run sample scenario

Network structure

- defined in crypto-config.yml, configtx.yml, docker-compose.yml
- high-level view (ignore the details for now)



First network tutorial

- To run the tutorial, use the script with following commands:
- **`./byfn.sh generate`** to create the cryptographic and channel artefacts
- **`./byfn.sh up`** to bring up the network and run a scenario using chaincode
- **`./byfn.sh down`** to stop the network and clean up the system
- before starting, we need to set a path to the binary tools:

```
export PATH=../bin:$PATH
```

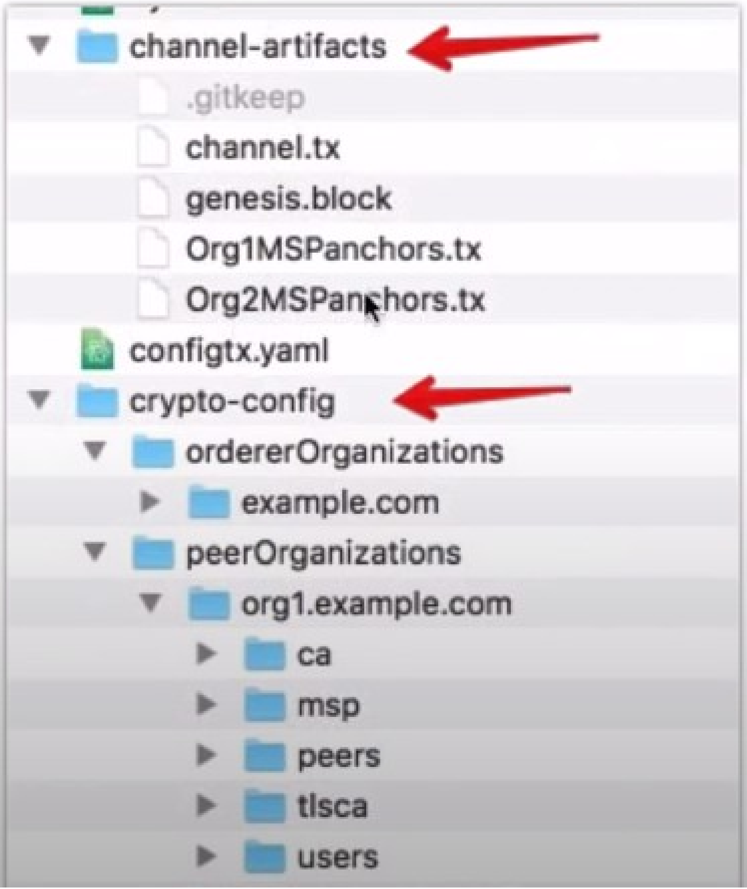
- ./byfn.sh generate

- ./byfn.sh generate

Generate

Tools to create cryptographic and channel artefacts:

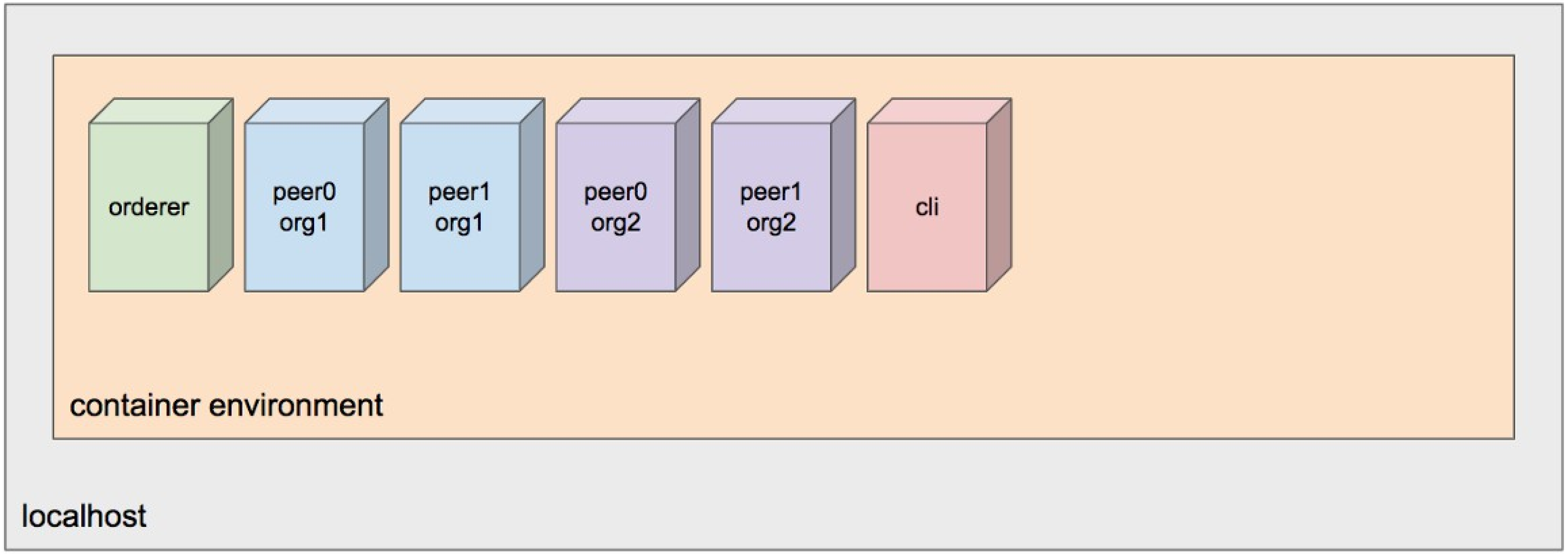
- **cryptogen** `generate —config=./crypto-config.yaml`
- **configtxgen** …
  - reads `configtx.yaml`
  - create genesis block
  - channel configuration
  - anchor peers



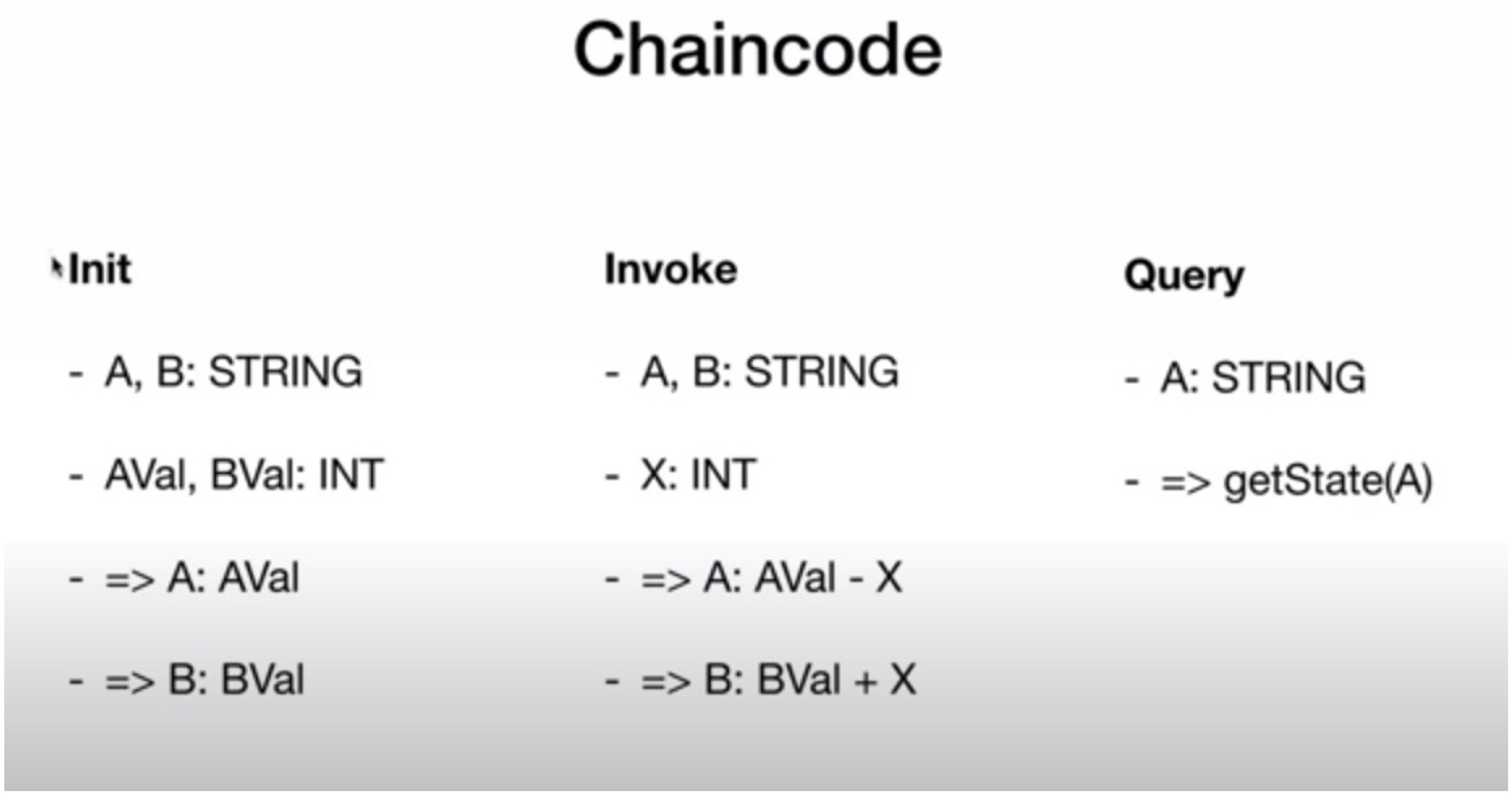
up

- start docker nodes
- create channel
- join 4 peers to channel
- update anchor peers
- run scenario: **`peer chaincode`** commands

```
./byfn.sh up
```


orderer
peer0 org1
peer1 org1
peer0 org2
peer1 org2
cli
container environment
localhost

Reach at Chain code Example to Review it



Chaincode

Init

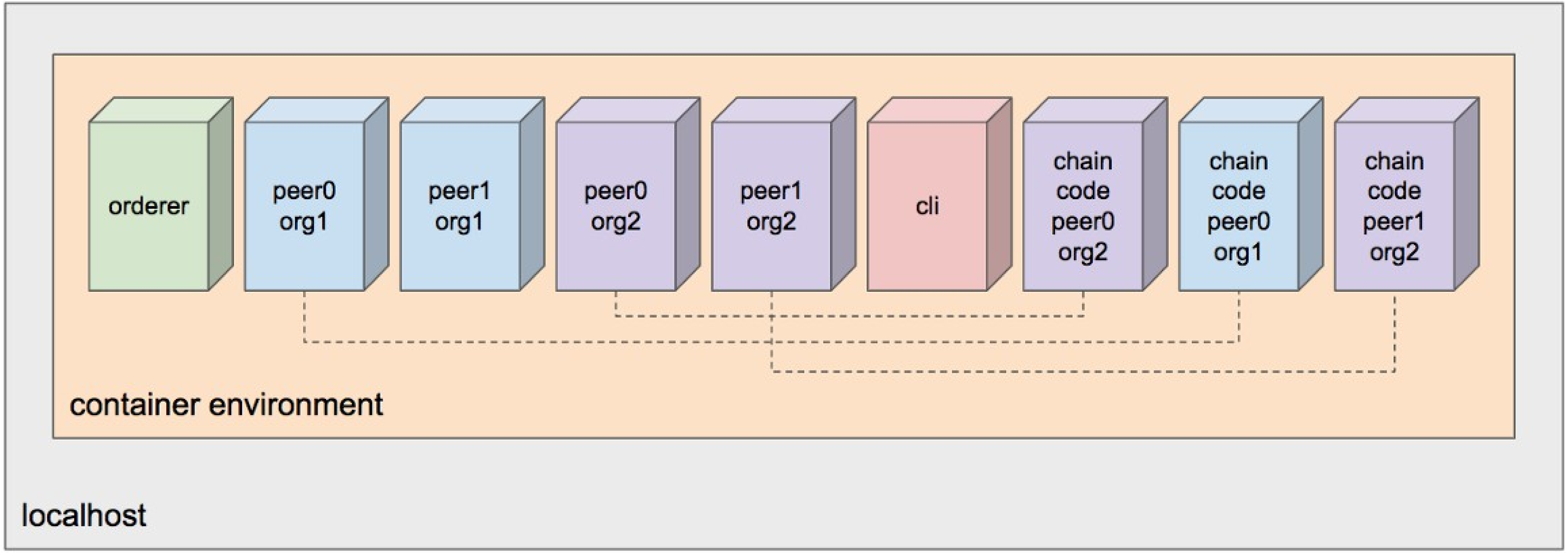
- A, B: STRING
- AVal, BVal: INT
- => A: AVal
- => B: BVal

Invoke

- A, B: STRING
- X: INT
- => A: AVal - X
- => B: BVal + X

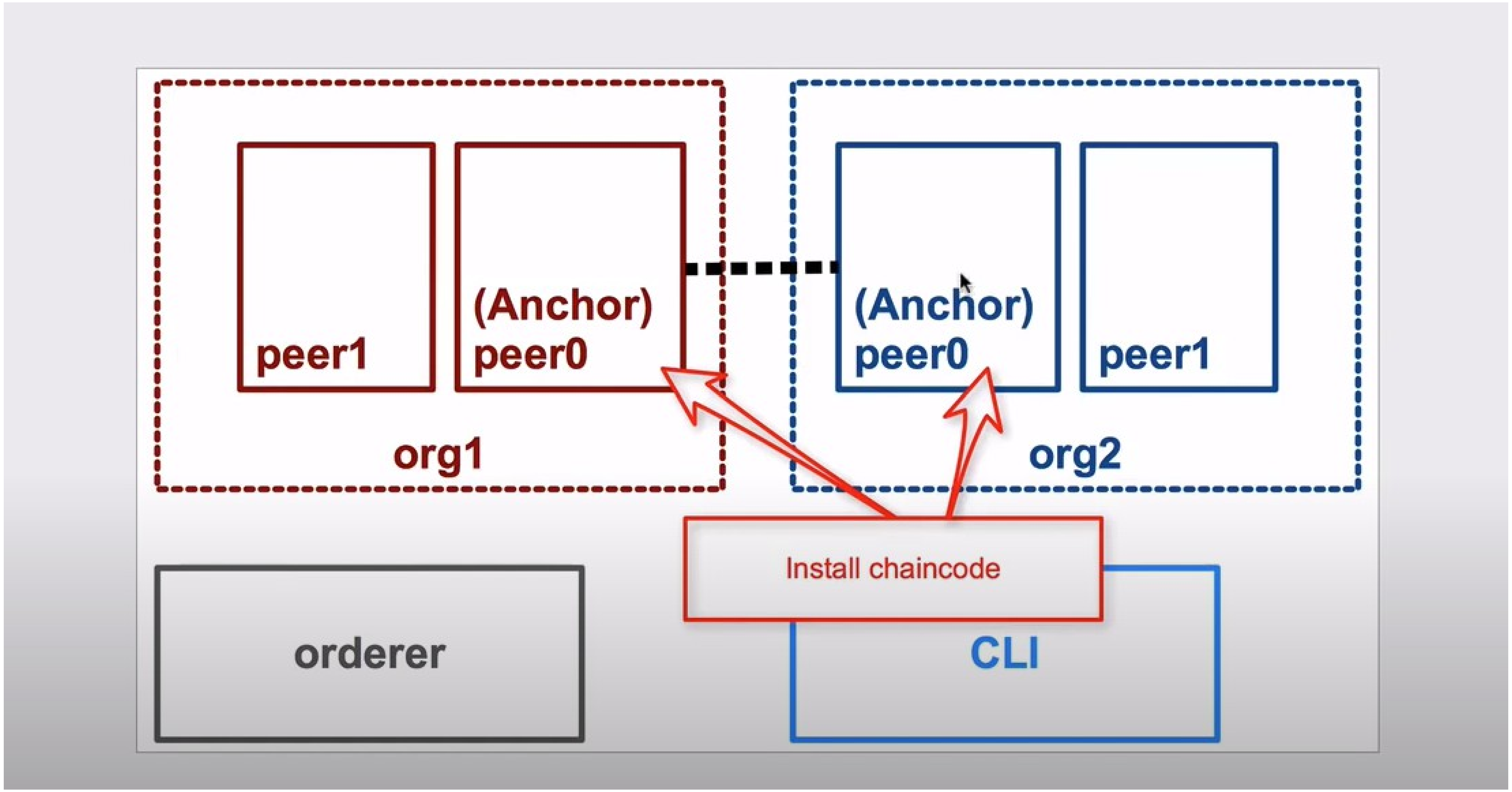
Query

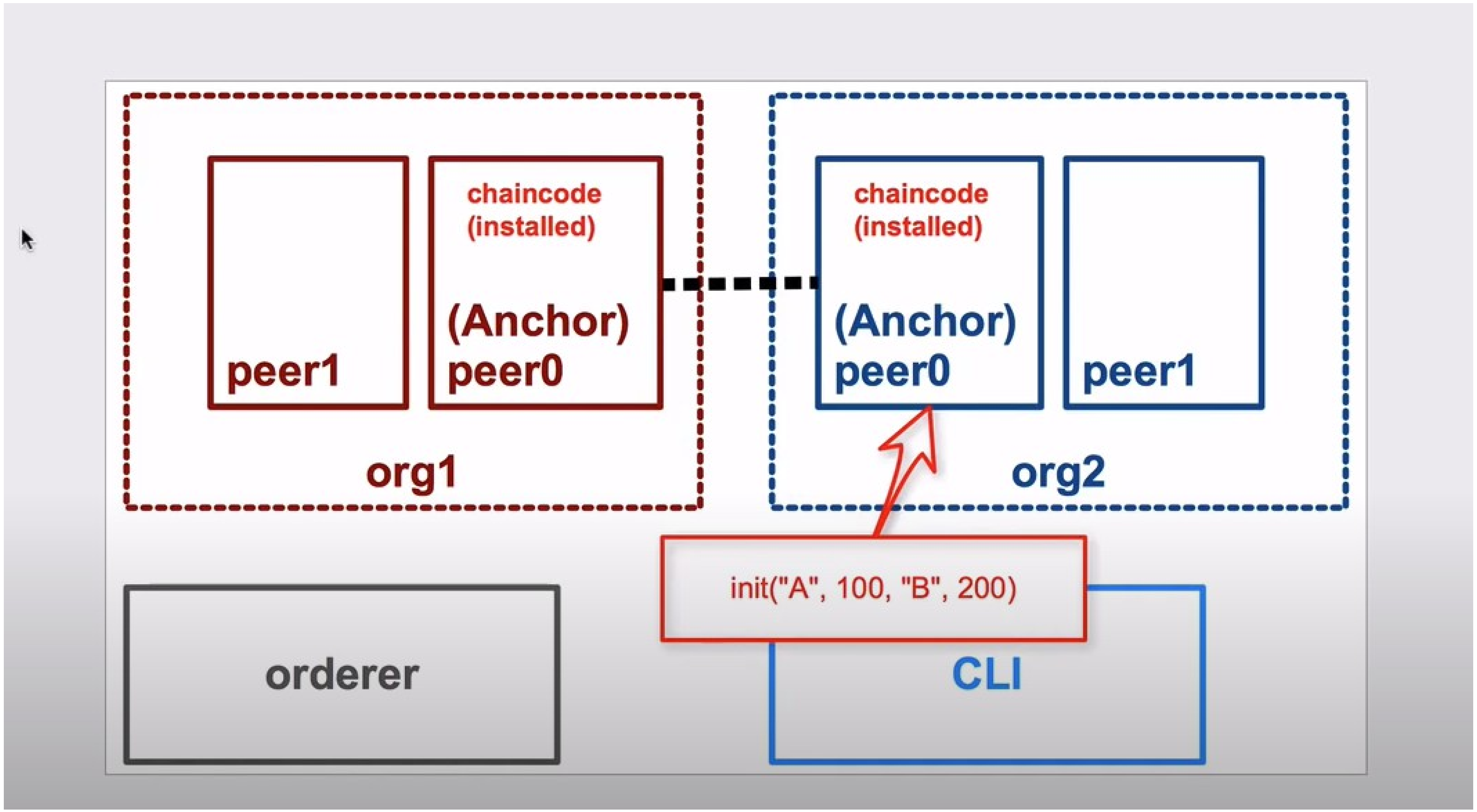
- A: STRING
- => getState(A)

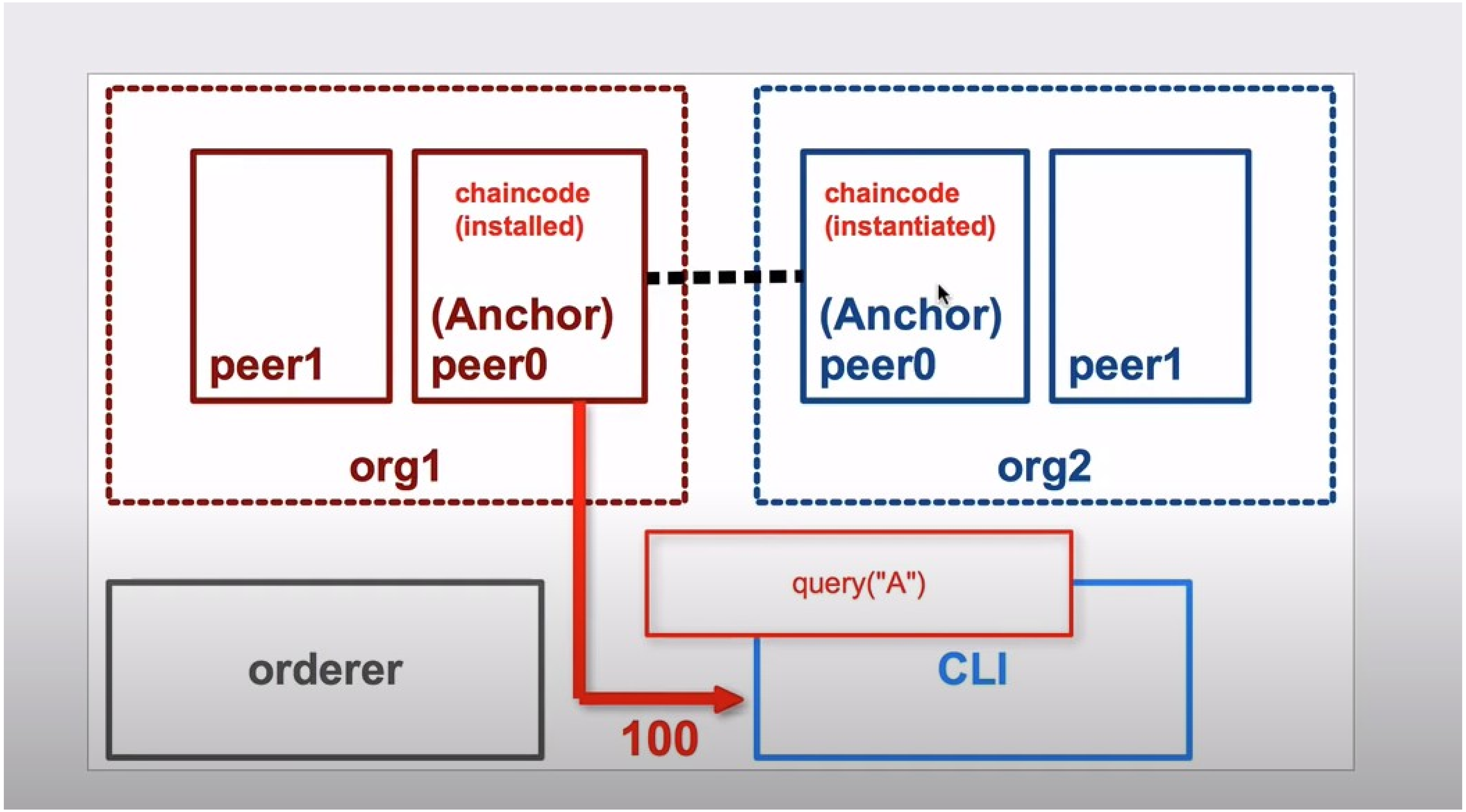

orderer
peer0 org1
peer1 org1
peer0 org2
peer1 org2
cli
chain code peer0 org2
chain code peer0 org1
chain code peer1 org2
container environment
localhost

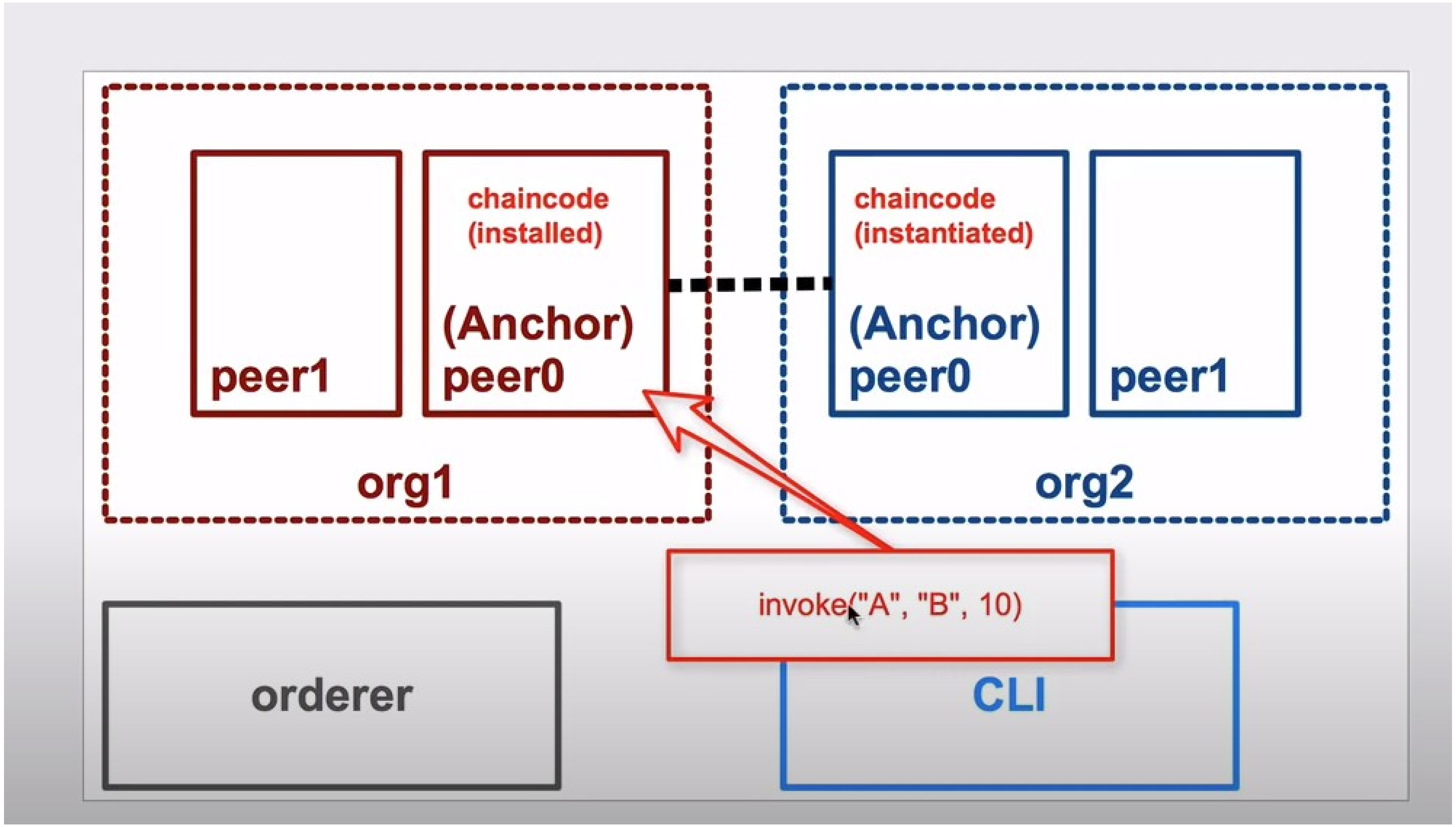
Execution Understanding

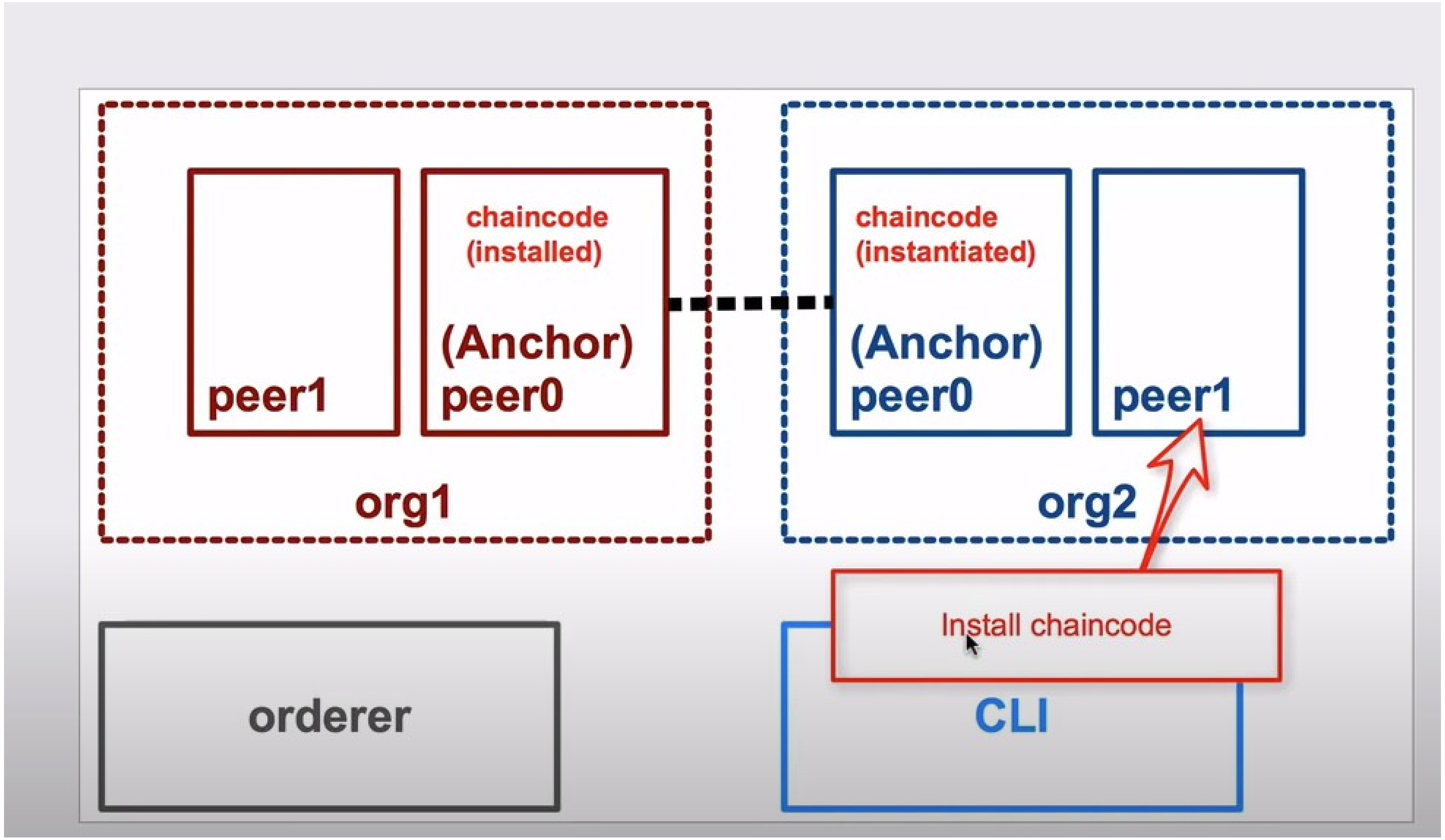
How All Happened

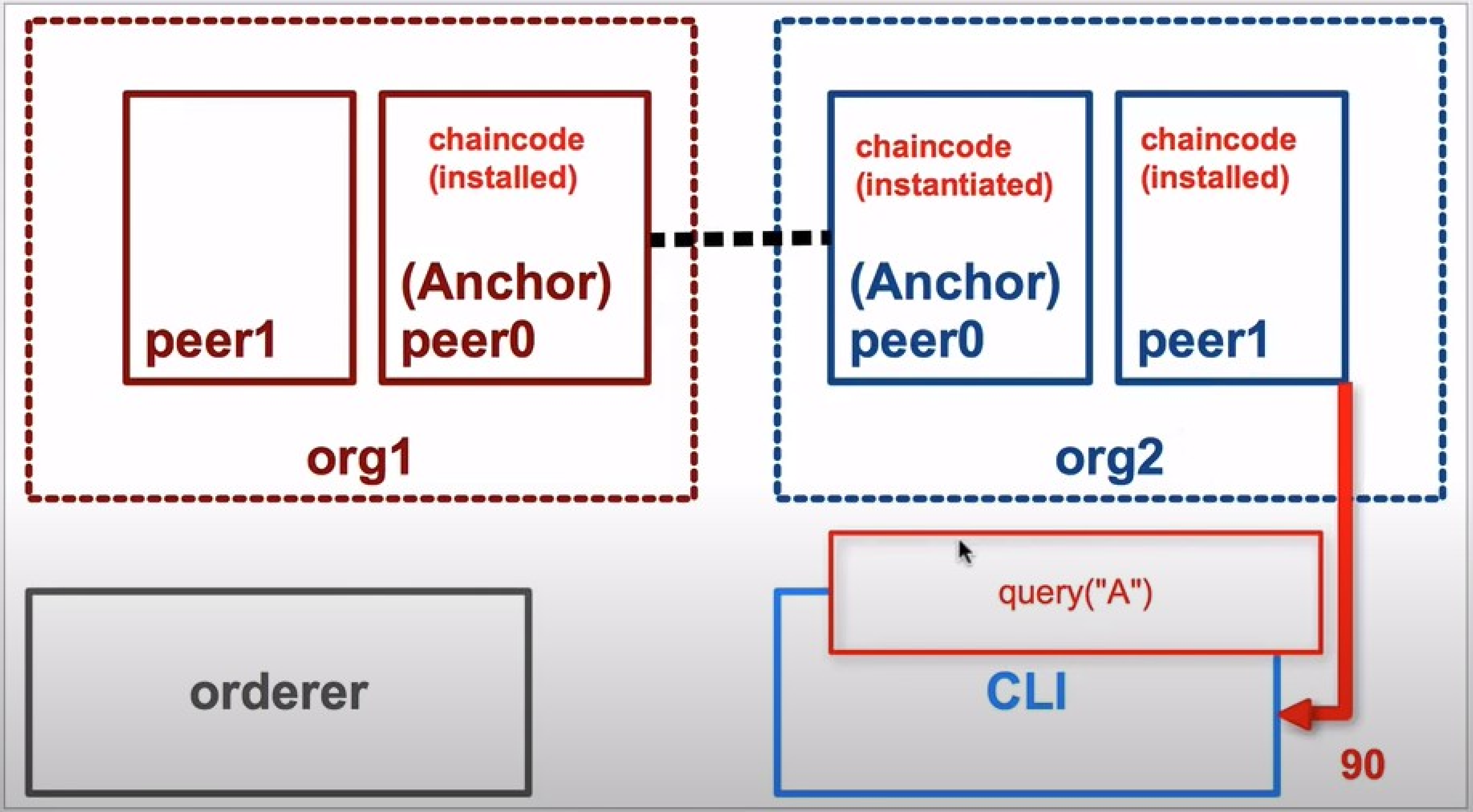

peer1
(Anchor)
peer0
(Anchor)
peer0
peer1
org1
org2
Install chaincode
orderer
CLI


chaincode
(installed)
(Anchor)
peer0
peer1
org1
chaincode
(installed)
(Anchor)
peer0
peer1
org2
init("A", 100, "B", 200)
orderer
CLI


chaincode
(installed)
(Anchor)
peer0
peer1
org1
chaincode
(instantiated)
(Anchor)
peer0
peer1
org2
orderer
query("A")
CLI
100


chaincode
(installed)
(Anchor)
peer0
peer1
org1
chaincode
(instantiated)
(Anchor)
peer0
peer1
org2
invoke("A", "B", 10)
orderer
CLI


chaincode
(installed)
(Anchor)
peer0
peer1
org1
chaincode
(instantiated)
(Anchor)
peer0
peer1
org2
orderer
Install chaincode
CLI


chaincode
(installed)
(Anchor)
peer0
peer1
org1
chaincode
(instantiated)
(Anchor)
peer0
chaincode
(installed)
peer1
org2
orderer
query("A")
CLI
90

Adding Organization