Chaincode

Wrapping up Problem Statement

Participant 

Asset 

Chaincode 

Transaction 

Ledger

World State

- **World State** store the current value of a business object (Asset)
- It stores as **key value pairs**

- For Eg:
  - `{key= Car-01, value= Audi}`
  - `{key= Car-02, value= {type:Sedan,color:Red,owner:Mike}}`

Chaincode

- Fabric uses **Chaincode and Smart Contract** Interchangeably
- A chaincode is a computer program (written in node.Js, Java, or go)
- It defines the **business logic** of your application.
- A chaincode is a **collection of smart contracts**.
- Within a chaincode, you can have multiple Smart contracts that perform related operations.
- A Smart Contract is the code that defines the **agreements or rules** of a transaction.

Chaincode Operations

Ledger Operations

A smart contract accesses two distinct pieces of the ledger

1. **A blockchain**, which immutably records the **history of all transactions**
2. **A world state** that holds a cache of the **current value** of the state

Operations

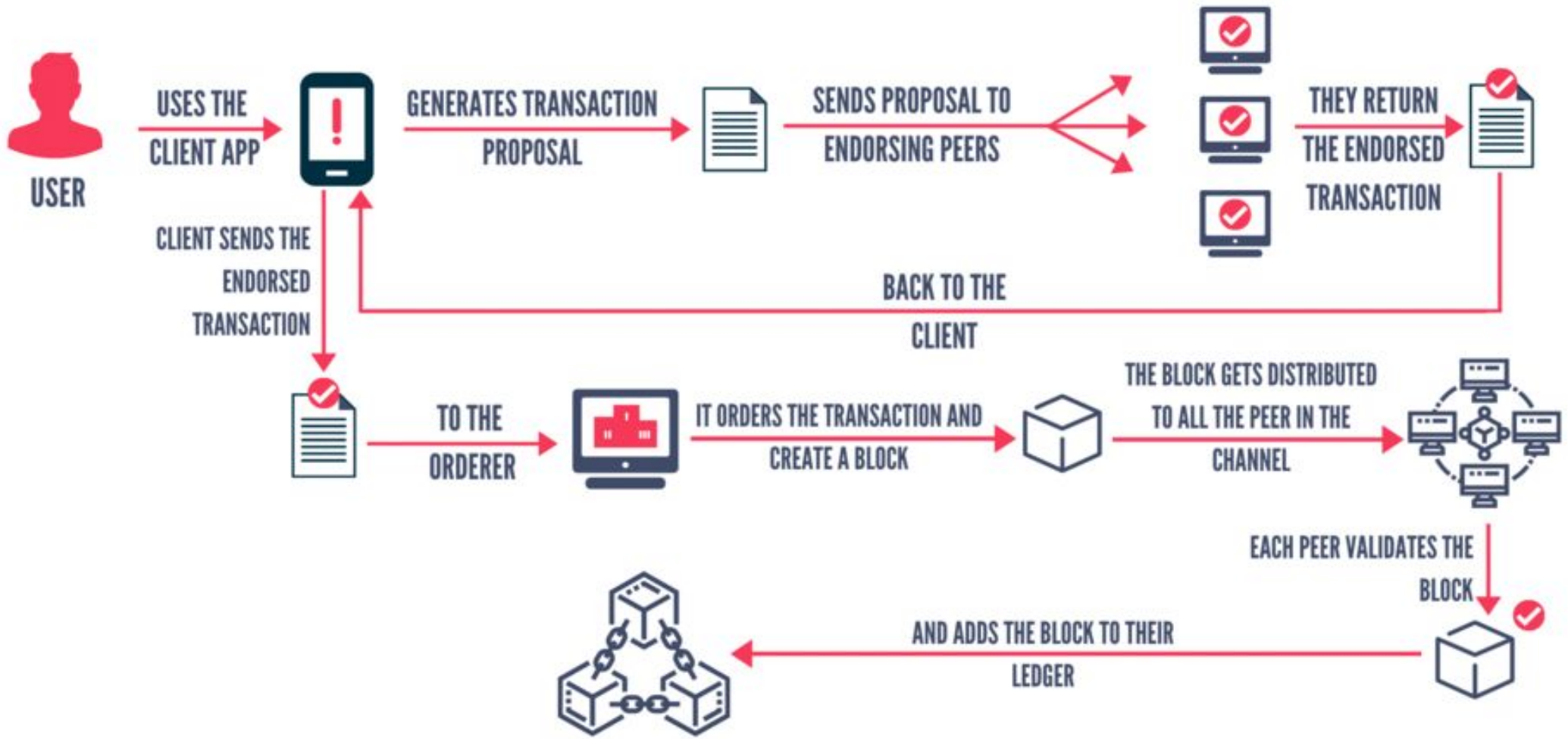
1. **Put, Get** and **Delete** states in the world state,
2. **Query** the immutable **blockchain record** of transactions.

These Smart Contracts operations will done by the way of **Invoking** and **Querying** transactions

Transactions

1. **Invoke**
   a. Update the Ledger
   b. Eg: Creating an asset, Updating an asset, Deleting an asset
2. **Query**
   a. Reads the ledger
   b. Eg: Read state of an asset

Transaction Flow



Invoke Transaction

Query Transactions

Endorsement Policy

- A Set of **Rules** specifies the set of peers on a channel that must **execute chaincode** and **endorse the execution** results for the transaction to be considered valid.
- The developers or administrators can define policies called the endorsement policies that are associated with the chaincode.
- At the time of transaction validation, the peers check for the appropriate number of endorsements from the endorsing peers.

How to build a Chaincode

Chaincode SDK

- Go
- Java
- Node.js

Node modules

- fabric-contract-api
- fabric-shim

fabric-contract-api

fabric-contract-api provides

- **Contract Class**
  - Contract Class is used to create user defined smart contracts by inheriting it
  - Smart contract must extend the Contract class, whose methods are called in response to received transactions.
  - Within each smart contract instance, it is possible to have as many functions as necessary.

Transaction Context

- Every transaction function must take a **transaction context (ctx)** as the first parameter.
- It provides access to **a wide range of Fabric APIs** that allow smart contract developers to perform operations relating to detailed transaction processing.
- **ctx.stub** is used to access APIs that provide a broad range of transaction processing operations
- **ctx.clientIdentity** is used to get information about the identity of the user who submitted the transaction.

Commonly Used Stub Methods

Some of the commonly used methods

- **putState**
  a. **putState** is a commonly used method to register an asset in the Fabric ledger. This method helps you to store a state variable on the ledger as a key-value pair.
  b. **putState** method will overwrite the state variable if it is already stored in the ledger.

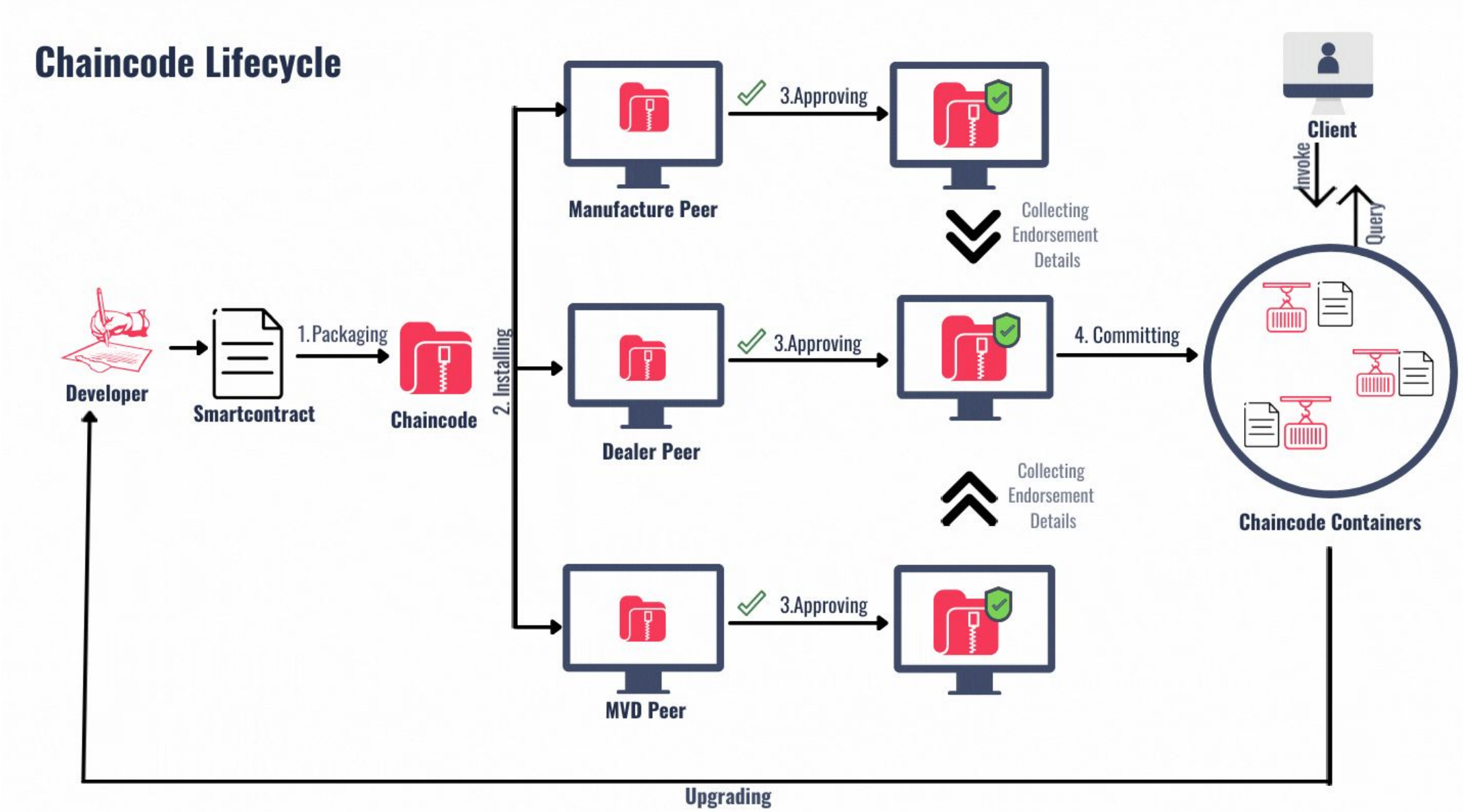
- **getState**
  a. This method helps you to retrieve an already stored state variable from the ledger.

- **deleteState**
  a. deleteState method will remove the state variable key from the world state store. But it will not affect the ledger history.

Chaincode Lifecycle

- Packaging (TAR.gz)
- Installing
- Approving Chaincode Definition
- Committing in a Channel

Chaincode Lifecycle



THANK YOU