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} Version: 0`
  - `{key= Car-02, value= {type:Sedan,color:Red,owner:Mike}} Version:0`

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** holds the **current value** of the attributes of a business object

Operations

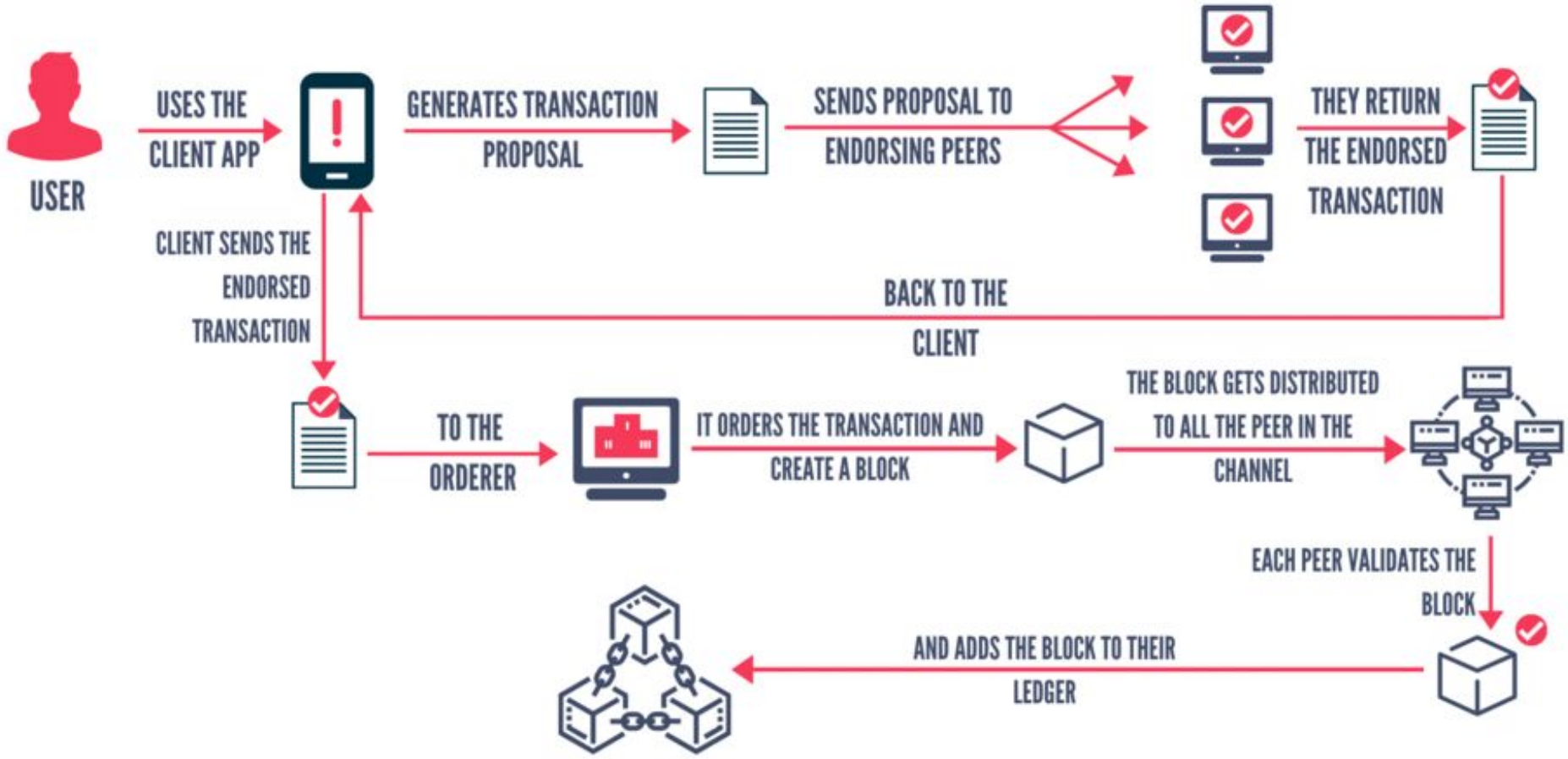
1. **Put, Get** and **Delete** states in the world state.
2. **Query** the immutable **blockchain record** of transactions.

These Smart Contracts operations are done by **Invoking** and **Querying** transactions

Transactions

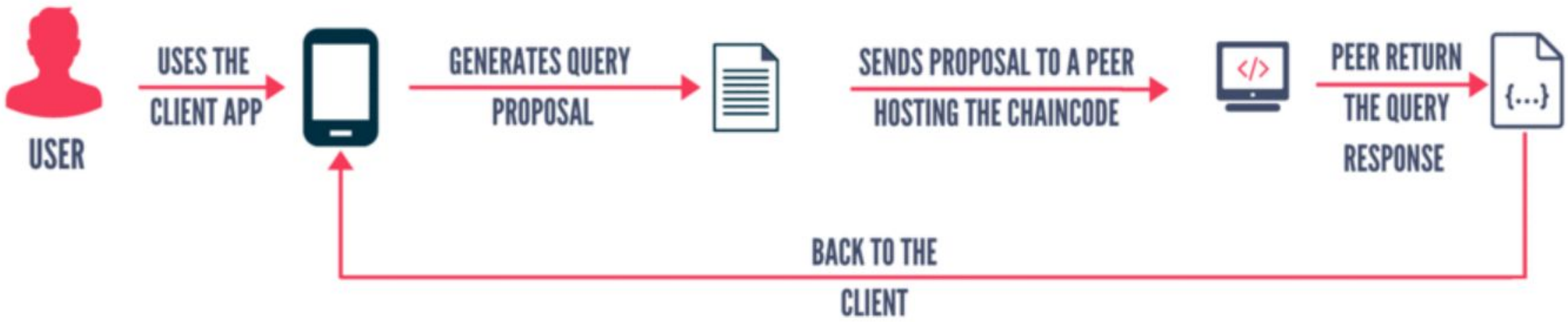
1. **Invoke**
   a. Update the Ledger
   b. Eg: Creating an asset, Updating an asset, Deleting an asset
2. **Query**
   a. Reads the ledger
   b. Eg: Read state of an asset

Transaction Flow



Invoke Transaction

Query 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.

Endorsement Policy

- The syntax of the language is: **EXPR(E[, E...])**, where EXPR is either AND, OR, or OutOf, and E is either a principal or another nested call to EXPR.

  For example:

- **AND('Org1.member', 'Org2.member', 'Org3.member')** requests one signature from each of the three.
- **OR('Org1.member', 'Org2.member')** requests one signature from either one of the two.
- **OR('Org1.member', AND('Org2.member', 'Org3.member'))**
- **OutOf(1, 'Org1.member', 'Org2.member')**, which resolves to the same thing as OR('Org1.member', 'Org2.member')
- **OutOf(2, 'Org1.member', 'Org2.member')** is equivalent to AND('Org1.member', 'Org2.member')

How to build a Chaincode

Chaincode SDK

- Go
- Java
- Node.js

Go package

- contractapi
  - Available in fabric-contract-api-go module
  - Define a struct with contractapi.Contract as a base struct for creating smart contracts.
  - This struct implement methods that represent the transactions or functions supported by the chaincode

Transaction Context

- Every transaction function must take a **transaction context (ctx)** as the first parameter.
- It's used to interact with the ledger, access the current transaction's information, and perform various operations related to the smart contract.
- **ctx.GetStub** is used to access APIs that provide a broad range of transaction processing operations
- **ctx.GetClientIdentity** 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 delete it from the ledger history (transaction log).

Functions

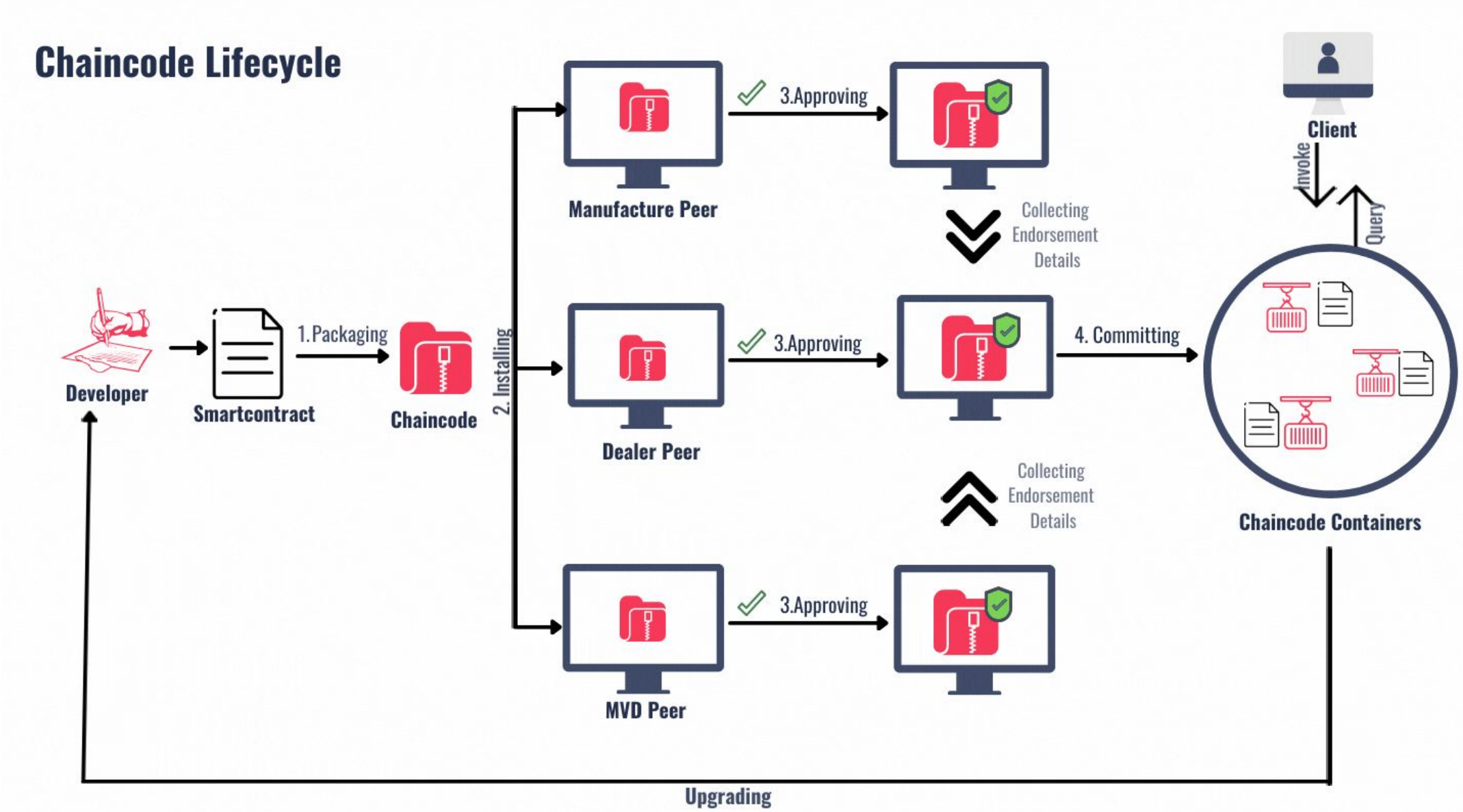
Some of the commonly used functions for chaincode development are:

- **ctx.GetStub().PutState(carID, bytes)**
- **ctx.GetStub().GetState(carID)**
- **ctx.GetStub().DelState(carID)**
- **ctx.GetClientIdentity().GetMSPID()**
- **contractapi.NewChaincode(carContract)**
- **chaincode.Start()**

Chaincode Lifecycle

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

Chaincode Lifecycle



THANK YOU