How to articulate a Blockchain Framework?

* Divide it into two portions: **infrastructure** and **application**.

**Infrastructure:**

* Also called ‘*network*’
* Composed of **nodes** and **software** running on them
* Nodes: Physical Machines, Virtual Machines and Containers
* Software: The soul, provides features, capabilities (like *user id/accounts, transaction formation, processing*), consensus protocol, framework specific capabilities (like mining for permissionless blockchains, channel and id management for permissioned blockchains)

**Application:**

* Composed of
  + code running inside the *infrastructure*, known as **Smart Contract** in general
  + client app interacting with infrastructure - access point of external world (Good blockchain framework should allow app dev outside the actual deployment of the network). No need to deploy network before app dev. If app can run on a small network, it should run well in bigger infrastructure.

**Two most popular Blockchain Frameworks:**

**Ethereum & Hyperledger Fabric**

1. **Ethereum**

* **Public** and **Permissionless** blockchain
* People keep developing app on this public blockchain

**Infrastructure**

* Network of nodes running Ethereum client. Have all the required capability to form functioning Ethereum network
* **Ethereum Mainnet** is up and running for years
  + Thousands of nodes
  + Real ethers used
  + Many miners competing against one another to maintain robustness of network with economic incentive
  + Node can join and leave the Mainnet at any time
* Therefore no need to worry too much about infrastructure, take it for granted that ‘it is working’
* For real app dev, don’t begin with Mainnet (due to cost, speed of testing). Use other Ethereum Infrastructure to speed up app dev. Use live public Testnets (eg. Ropsten, Kovan, Rinkeby) for testing before deploying to Mainnet

**Application**

* Known as Decentralized Application (DApp)
* Contract Code executed across network in decentralized way
* Application divided into 2 parts:
  + Contract Code
    - Coding language - usually **Solidity**, also **Vyper** (claimed to be better than Solidity)
    - Code after compilation runs inside the Ethereum nodes (EVM)
    - Result of code execution = update the world state &/or any contract state
  + Client Application interacting with Ethereum network and deployed contract code
    - Network comes with **JSON RPC** - a light-weight RPC protocol, acting as interface to ext. World
    - Besides direct calling, libraries available for different prog languages and frameworks (**web3.js** for JS)
* Application developed on one Ethereum Network should work on another
* Tools like **Remix** and **Truffle** allow you to select which Ethereum network to deploy your contract code

1. **Hyperledger Fabric**

* **Enterprise** Blockchain Platform
* Enables consortium app across business entities

**Infrastructure**

* Deployed in business world
* Consortium model instead of public network
* Example, Hyperledger Fabric network deployed in the datacenters of participating banks for a trade finance network
* No public Hyperledger Fabric that we are deploying application on top of it, like we do on Ethereum Mainnet
* Composed of nodes of different roles:
  + **Peers:** Nodes keeping the ledger (blockchain of transactions and a world state database) (eg. **Endorsing Peers** also execute Contract Code, called chaincode)
  + **Orderers:** Nodes providing consensus and generating blocks for Peer nodes to maintain the ledger
  + **Membership Service Provider (MSP)** and **Certificate Authority (CA):** handle identity and permission in network
* Network can be simple (Single peer and single orderer) or complicated
* No public network, so need a network before we test our app
* Network deployment involves:
  + proper generation of signing key and certification for identity
  + initial configuration transaction for channel setup
  + docker containers as nodes of different roles
  + correct matching of certificates to the nodes and users, etc.
* Fabric-Samples from Hyperledger Fabric comes with tools and container images for bringing up networks.

**Application**

* Composed of:
  + **Chaincode**
    - Code being executed by **Endorsing Nodes**
    - After processing by **Orderer** and verification by all other Peers, result to update the ledger
    - Related Activities:
      * Installation
      * Instantiation
      * Invoke/query
    - Earlier only chaincode coding language - **Golang**, but now we also have **Node** and **Java**
  + **Client Application**
    - Comes with SDK which facilitate the client app development (e.g. Node SDK for JS App)
    - Currently Node and Java SDK officially released
    - Some Platforms like Python, Go and REST are available as well
* As good blockchain framework should allow app development independent from the infrastructure, we can develop and test our chaincode on a minimum setup of Hyperledger Fabric network - to focus only on chaincode. **Basic-Network** inside Fabric Samples is good (comes with one CA, 1 Orderer, 1 Peer)

*Source:* [*https://medium.com/coinmonks/understanding-blockchain-frameworks-ethereum-and-hyperledger-fabric-48a57082903e*](https://medium.com/coinmonks/understanding-blockchain-frameworks-ethereum-and-hyperledger-fabric-48a57082903e)