# Module 7: Hyperledger

Aron Kondoro & Anthony Kigombola

# Contents

- Overview
- Projects
- Frameworks
- Tools
- Hyperledger Fabric
- Hands-on: Review open source blockchain frameworks, technologies, and related tools

# Overview

- Hyperledger: A Linux Foundation Project
- A collaborative effort created to advance cross-industry blockchain technologies for business
- Announced in December 2015
- Now over 250 members – Finance, Banking, Internet-of-Things, Supply Chains, and Manufacturing
- Modular approach to hosting projects

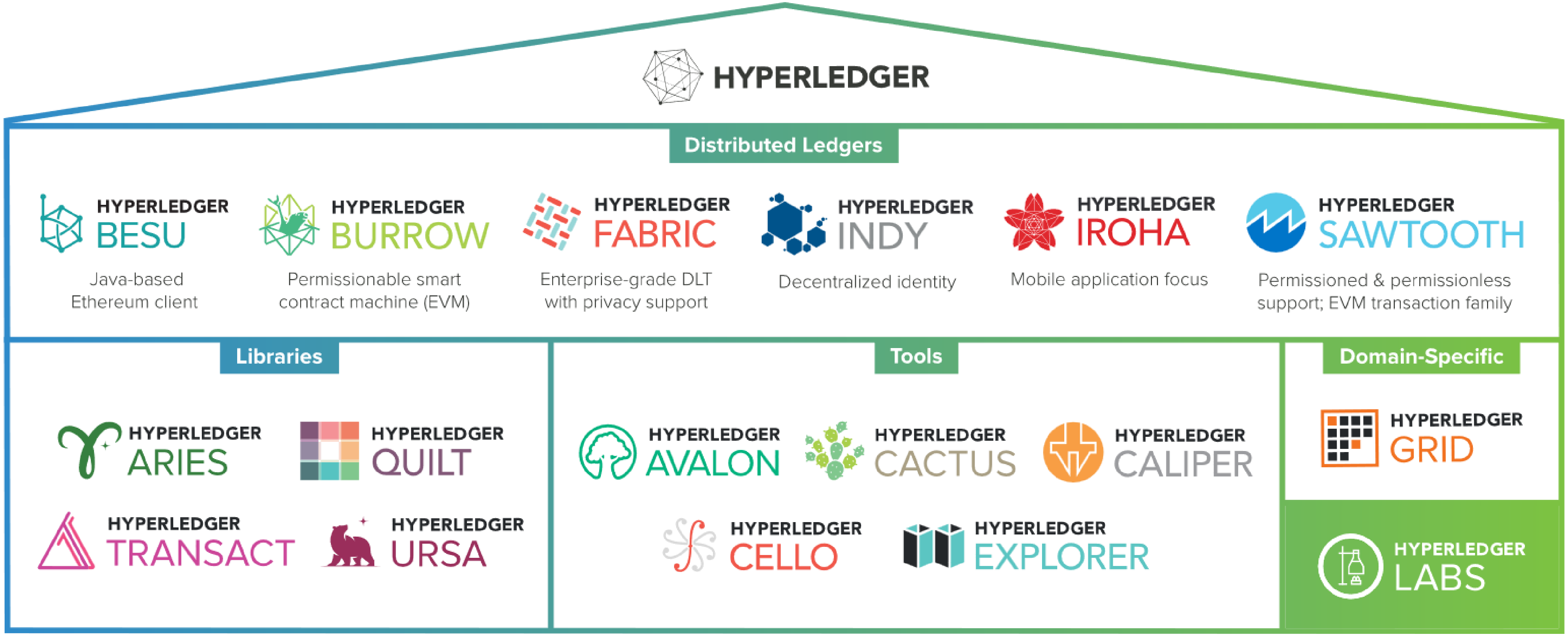
# Why?

- For those interested in non-currency blockchain applications
- Permissioned blockchains
  - Participants in business need to know the identity of one another
- Privacy
  - Participants want to share data with a greater degree of privacy
- Efficiency
  - No need to run proof of work mechanism. Can resolve more immediate problems than on a public cryptocurrency blockchain

# Enterprise Needs

1. Ledger should not be shared with everyone
   - *I don’t want to share all of my data with every participant.*
2. Users should not be anonymous
   - *I want to know who my users are. Anonymity does not benefit me.*
3. Users should not have full transparency
   - *I want to control which users can see which parts of my data.*
4. Group Consensus should be replaced by Participant Consensus
   - *I don’t need the entire network to validate transactions, I can simply have the participants validate their own transactions.*

# Projects



HYPERLEDGER

Distributed Ledgers

HYPERLEDGER BESU
Java-based Ethereum client

HYPERLEDGER BURROW
Permissionable smart contract machine (EVM)

HYPERLEDGER FABRIC
Enterprise-grade DLT with privacy support

HYPERLEDGER INDY
Decentralized identity

HYPERLEDGER IROHA
Mobile application focus

HYPERLEDGER SAWTOOTH
Permissioned & permissionless support; EVM transaction family

Libraries

HYPERLEDGER ARIES

HYPERLEDGER QUILT

HYPERLEDGER TRANSACT

HYPERLEDGER URSA

Tools

HYPERLEDGER AVALON

HYPERLEDGER CACTUS

HYPERLEDGER CALIPER

HYPERLEDGER CELLO

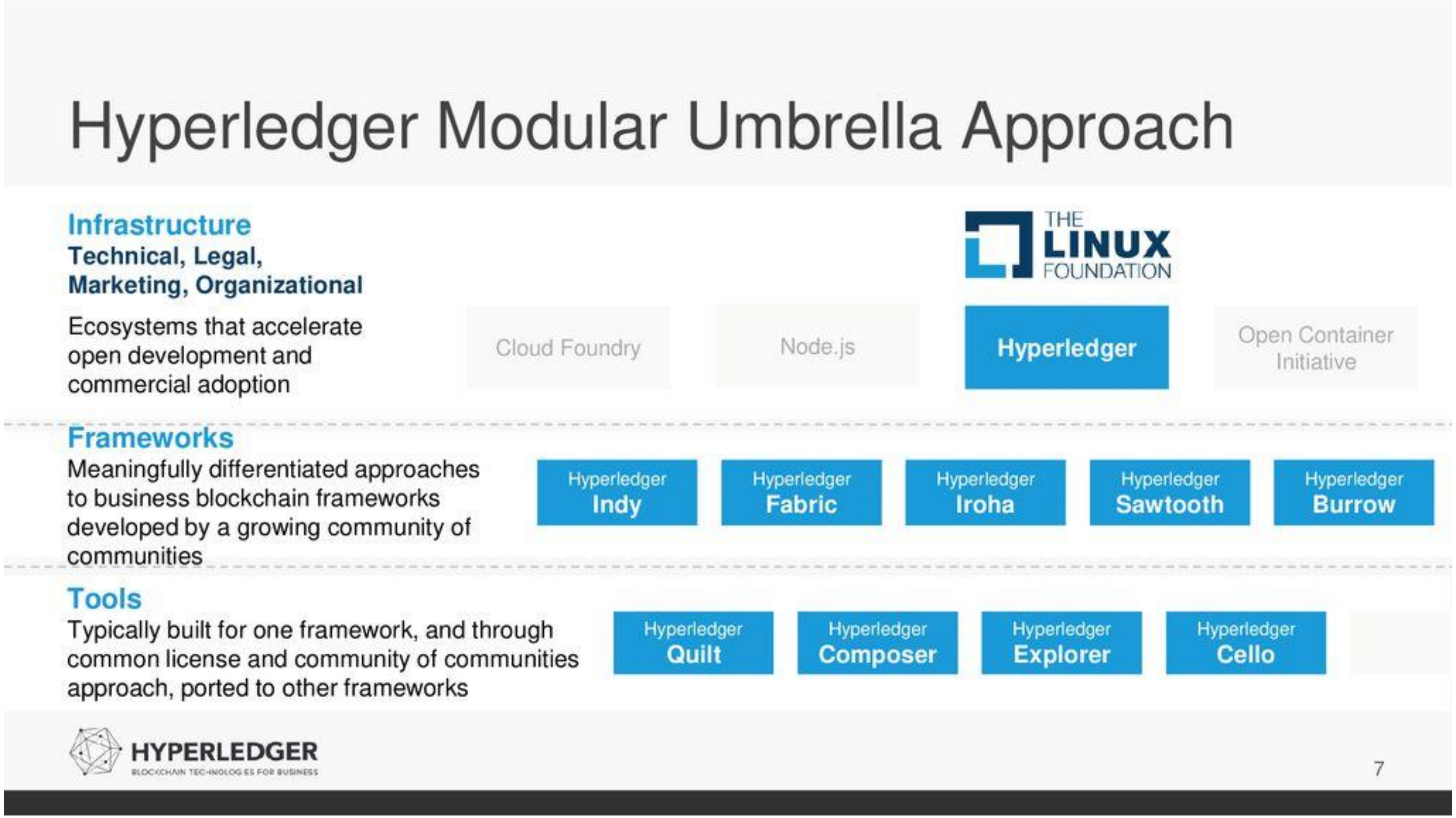
HYPERLEDGER EXPLORER

Domain-Specific

HYPERLEDGER GRID

HYPERLEDGER LABS

# Modular Architecture



## Hyperledger Modular Umbrella Approach

**Infrastructure**
**Technical, Legal, Marketing, Organizational**

Ecosystems that accelerate open development and commercial adoption

**Frameworks**

Meaningfully differentiated approaches to business blockchain frameworks developed by a growing community of communities

**Tools**

Typically built for one framework, and through common license and community of communities approach, ported to other frameworks

# Frameworks and Tools

- 28,000 participants
- 3.6 million lines of code
- Source code: https://github.com/hyperledger
- 10 Projects: 5 frameworks & 5 tools
- Hyperledger Fabric is the most popular framework

# Frameworks

- Hyperledger Burrow
  - Modular blockchain client with permissioned smart contract engine
  - Written in Go
  - Focused on being a deterministic smart contract engine
  - Uses Tendermint Proof-of-Stake BFT
- Hyperledger Iroha
  - Designed for incorporating infrastructure projects & facilitating the creation of applications tailored for end users
  - Written in C++
  - Features: Support for mobile application development, chain-based BFT consensus algorithm

# Frameworks

- Hyperledger Indy
  - Distributed ledger built explicitly for decentralized identity management
  - Server portion Indy-node built in Python while Indy-SDK written in Rust
  - Provides tools and reusable components for managing digital identity
  - Features: self-sovereignty, privacy, identity attestations
- Hyperledger Sawtooth
  - Modular platform for running distributed ledgers
  - Written in Python (Sawtooth-core) and Raft (Sawtooth-raft & Sawtooth-sabre)
  - Features: dynamic consensus, Proof of Elapsed Time (PoET), parallel transaction execution, and private transactions

# Tools

- Hyperledger Caliper
  - Written in Javascript
  - Benchmark tool for measuring blockchain performance i.e. transactions per second (TPS), transaction latency, resource utilization
- Hyperledger Cello
  - Written in Go
  - Automated application for deploying and managing blockchains
  - Provides real-time dashboard for blockchain statutes, system utilization, chain code performance, and configurations of blockchains
  - Works with Hyperledger Fabric

# Tools

- Hyperledger Composer
  - Written in Javascript
  - Tool for building blockchain business networks
  - Designed to accelerate the integration of blockchain applications and smart contracts with existing business models i.e. define how tx interact with specific assets
  - Supports Fabric
- Hyperledger Explorer
  - Written in Javascript
  - Generic, web-based block explorer for Fabric
  - Provides a dashboard for peering into details about blocks

HYPERLEDGER COMPOSER

HYPERLEDGER EXPLORER

# Tools

- Hyperledger Quilt
  - Written in Java
  - Interoperability tool between ledger systems
  - Implements the Interledger Protocol (ILP) and provides libraries and reference implementations of the core InterLedger components used for payment networks

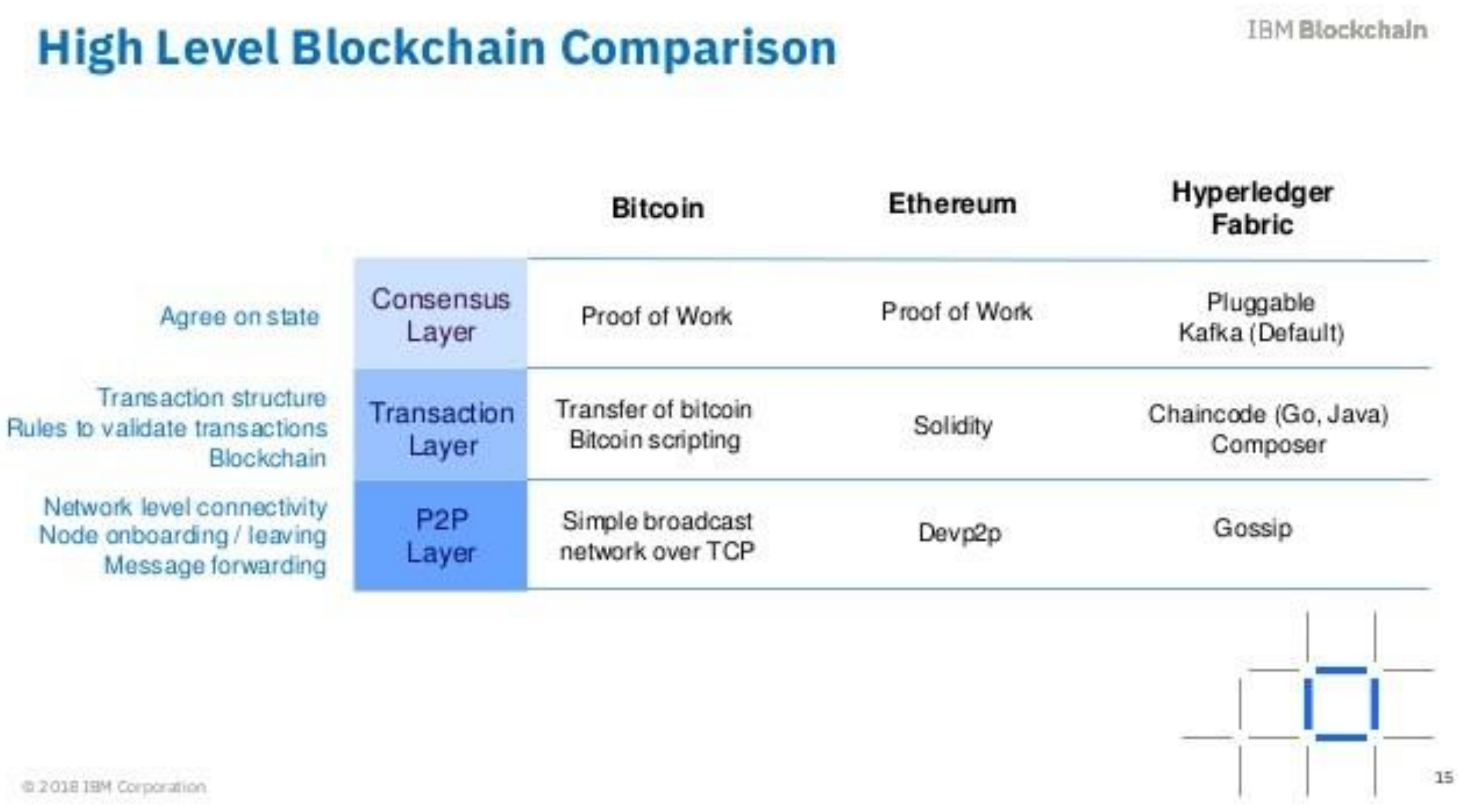
HYPERLEDGER
QUILT

# Hyperledger Fabric

- Hyperledger Fabric
  - Most popular
  - Written in Go
  - Modular platform for building distributed ledger solutions
  - Extensible with pluggable components

HYPERLEDGER
FABRIC

# Comparison with other Blockchain Technology



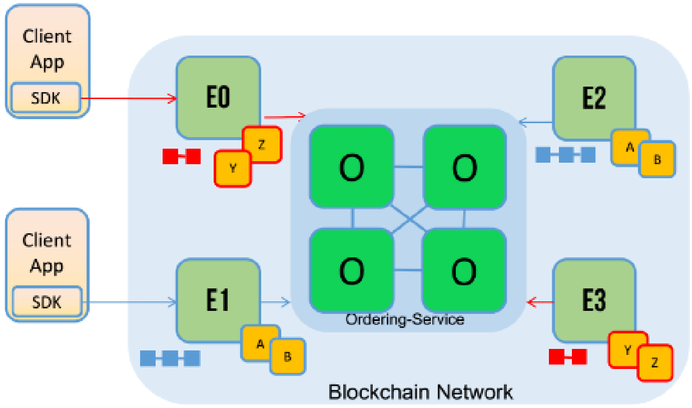
## High Level Blockchain Comparison

IBM Blockchain

| | | Bitcoin | Ethereum | Hyperledger Fabric |
|---|---|---|---|---|
| Agree on state | Consensus Layer | Proof of Work | Proof of Work | Pluggable<br>Kafka (Default) |
| Transaction structure<br>Rules to validate transactions<br>Blockchain | Transaction Layer | Transfer of bitcoin<br>Bitcoin scripting | Solidity | Chaincode (Go, Java)<br>Composer |
| Network level connectivity<br>Node onboarding / leaving<br>Message forwarding | P2P Layer | Simple broadcast network over TCP | Devp2p | Gossip |

# Hyperledger Fabric Key Concepts

- **Problem: Ledger should not be shared with everyone**
  - *I don’t want to share all of my data with every participant.*
- **Solution: Channels**
  - Channels offer a way to create multiple ledgers, each of which can have a unique set of participants and permissions.



# Hyperledger Fabric Key Concepts

- **Problem: Users should not be anonymous**
  - *I want to know who my users are. Anonymity does not benefit me.*
- **Solution: Membership Service Provider**
  - A pluggable component to a Hyperledger Fabric Network
  - One per organization
  - Contains list of all known human and system identities
  - Gives all participants on a Fabric network an identity
  - LDAP, Active Directory, oAuth are common

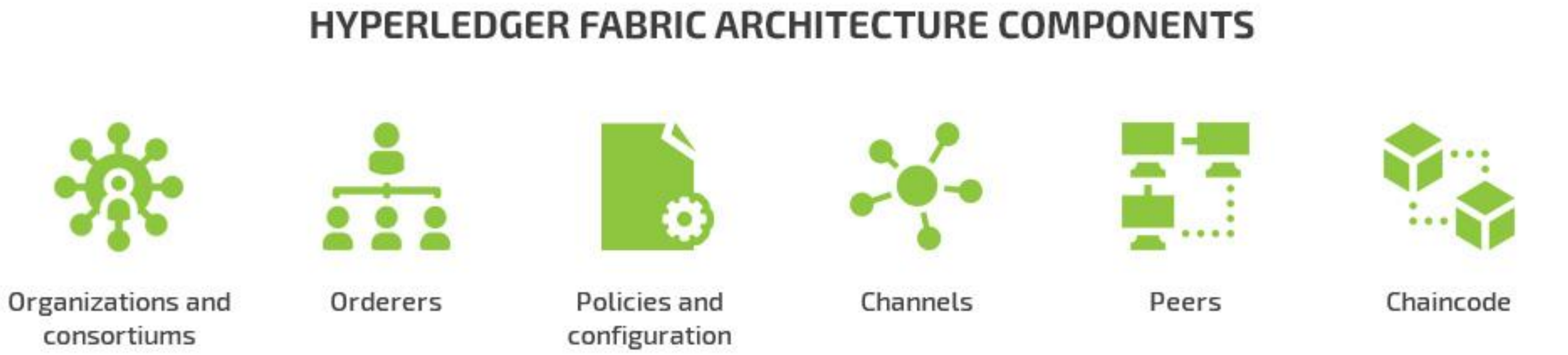
# Hyperledger Fabric Key Concepts

- **Problem: Users should not have full transparency**
  - *I want to control which users can see which parts of my data.*
- **Solution: Channel Permissions and ACLs**
  - ACL == Access Control Listing
  - Permissions made possible via identity (MSP)
  - Permissions can be applied at the channel level

# Hyperledger Fabric Key Concepts

- **Problem: Group Consensus should be replaced by Participant Consensus**
  - *I don’t need the entire network to validate transactions, I can simply have the participants validate their own transactions.*
- **Solution: Endorsement Policies**
  - Once identity is known, group consensus can be replaced by participant consensus.
  - If you and I have a transactions, and we both agree on the outcome of that transaction why do we need anybody to help validate it?

# Fabric Network Architecture Components



HYPERLEDGER FABRIC ARCHITECTURE COMPONENTS

https://www.apriorit.com

# Components

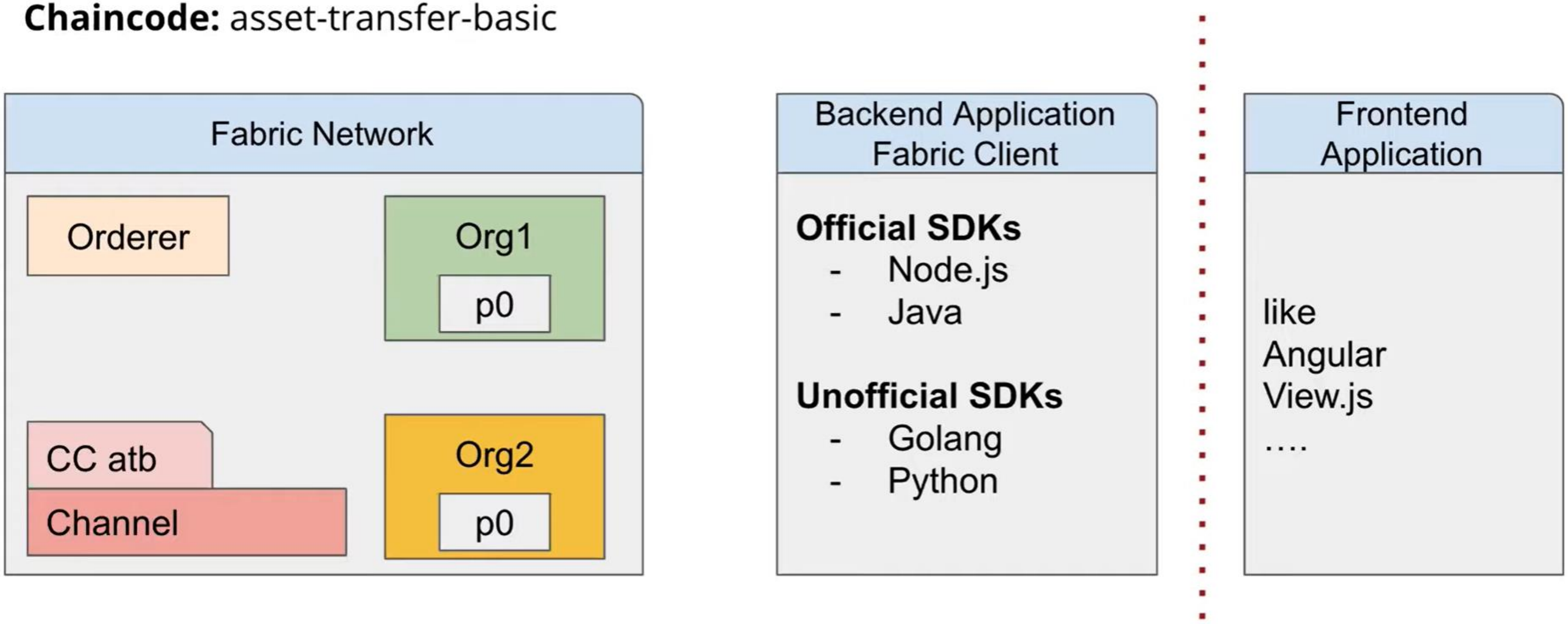
- **Organizations and consortiums**
  - Multiple orgs come together as a consortium to form a network
  - Permissions determined by policies during initial setup
- **Chaincode**
  - Aka smart contract
  - Run on different peers/channels
  - Work as separate programs in peers
- **Orderers**
  - Backbone of the network
  - Node(s) form an ordering service
  - Offer a broadcast service for messages
  - Ensure atomic delivery of msgs and consensus of tx within each channel

# Components

- **Peers**
  - Participants in the network
  - Keep a local copy of the ledger for each channel
  - Provide client applications with access to the ledger and chaincode
- **Channels**
  - Dedicated line of communication between peers
  - Support atomic delivery of messages
  - Have independent blockchain state and ledger
- **Policies and configuration**
  - Rules agreed by the consortium during initial network configuration
  - Include access to admin functions, channel creation, chaincode instantiation, endorsement (validation) policies etc
  - May change over time

# The Whole Picture

**Chaincode:** asset-transfer-basic



Fabric Network

Orderer

Org1

p0

CC atb

Channel

Org2

p0

Backend Application
Fabric Client

**Official SDKs**

- Node.js
- Java

**Unofficial SDKs**

- Golang
- Python

Frontend
Application

like
Angular
View.js
....

# Hyperledger Fabric Transaction Flow

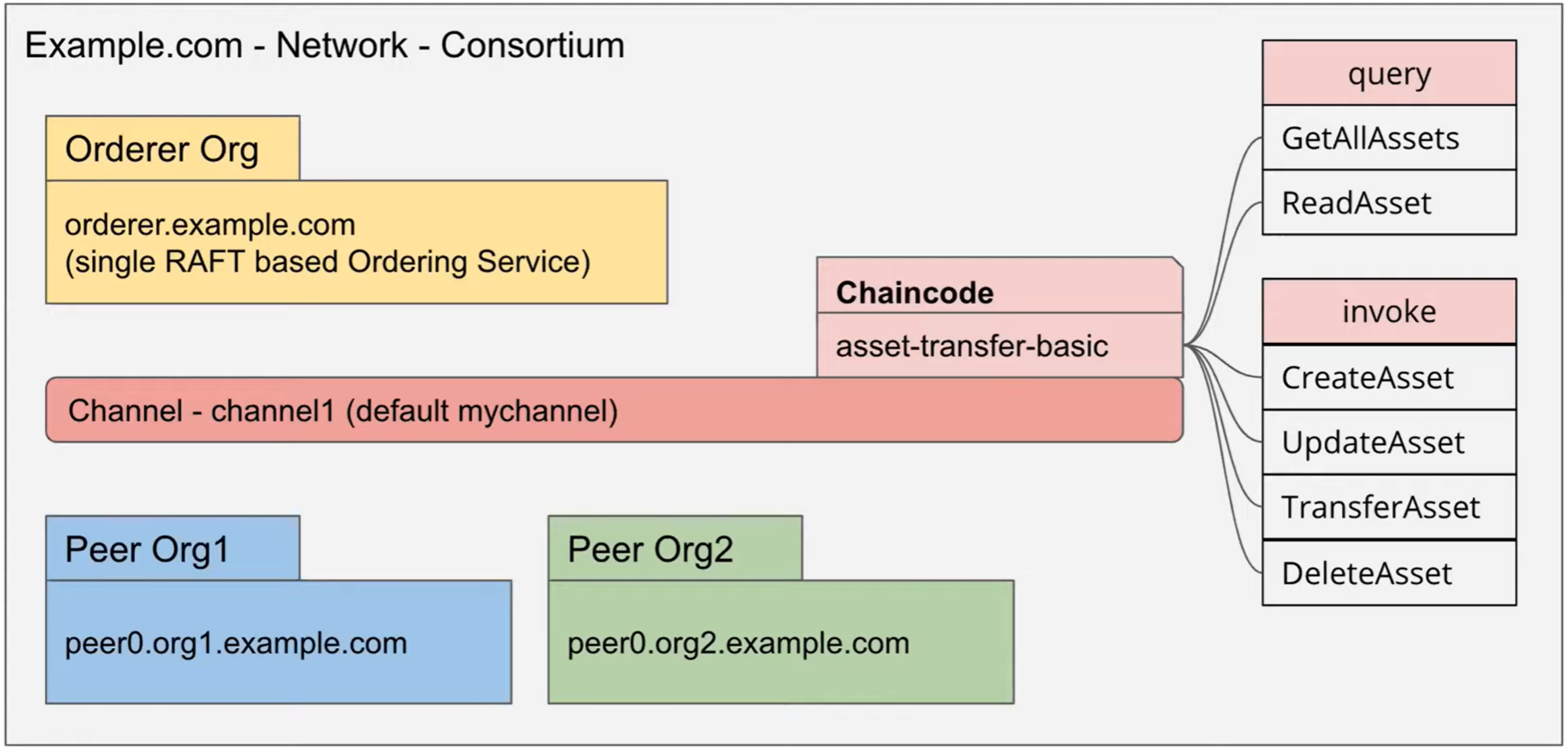
1. An end user initiates a transaction.
2. The network verifies the identity of the initiator using the appropriate Membership Service Provider.
   - Are you a valid user?
3. The network verifies the identity of the initiator using public/private key cryptography.
   - Are you who you claim to be?
4. The network verifies the user has permissions to perform the transaction.

# Hyperledger Fabric Transaction Flow

5. The transaction is broadcast to all Endorsing nodes on the channel.
6. Each Endorsing node executes Smart Contract code and returns their result to the client application.
7. The client application checks to if consensus was reached by examining the returned results.
8. The client application informs the Ordering Nodes that a new Transaction is to be recorded on the ledger.
9. The Committing nodes (and the Endorsing nodes) record the transaction on their copy of the ledger.

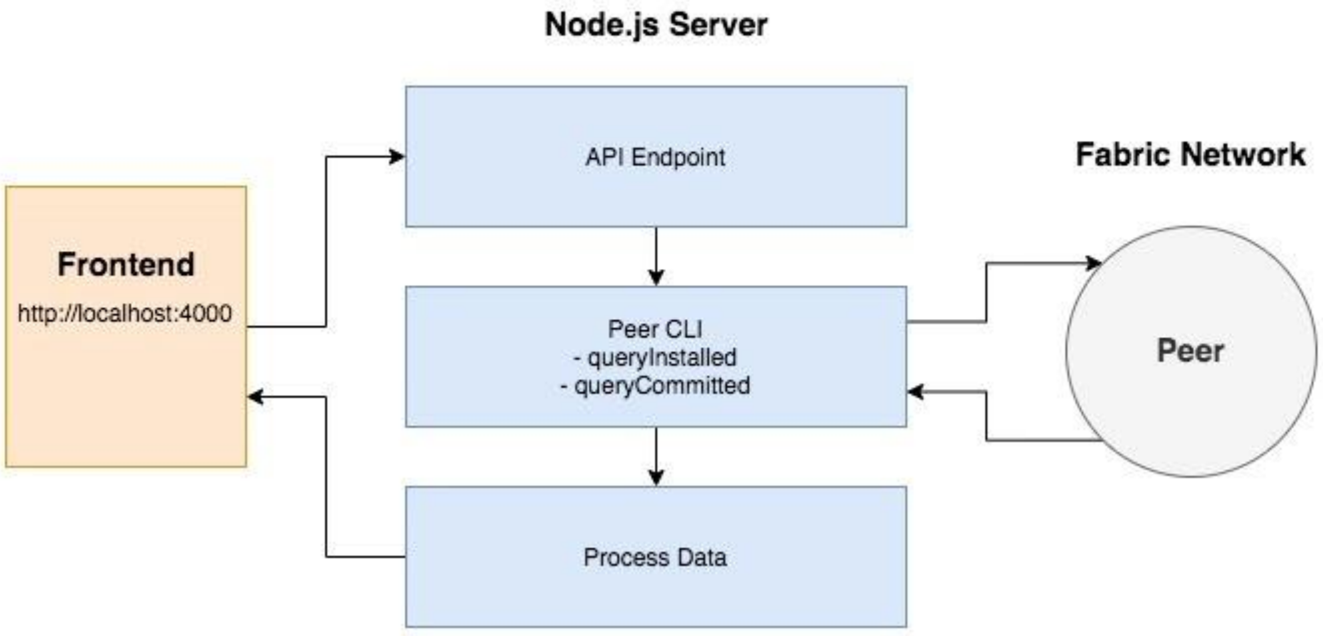
# Activity 1

- Set up a test Hyperledger Fabric network



# Activity 2

- Deploy a Hyperledger Fabric v2 web app using the Node.js SDK + Chainstack


Node.js Server
Frontend
http://localhost:4000
API Endpoint
Peer CLI
- queryInstalled
- queryCommitted
Process Data
Fabric Network
Peer