edda INSPIRING excellence

# Permissioned Blockchain Framework
# Hyperledger Fabric

16/12 – 17/12

WORKSHOP

# AGENDA

01 Private Blockchain Network

02 Hyperledger Fabric

03 NSPA Demonstrators

04 *Practice I*: Test Environment

05 *Practice II*: Build a Channel

06 *Practice III*: Deploy chaincode



edda INSPIRING EXCELLENCE

# PRIVATE BLOCKCHAIN NETWORK

01

edda INSPIRING excellence

# Blockchain Networks

| Private | | Public |
|---|---|---|
| | Access Management | |
|  Permission based | | Open  |
| | Motivation | |
|  Common interest | | Costs / Incentives  |
| | Security | |
|  Proof of Authority | | Proof of Work, Stake...  |



edda INSPIRING excellence

# HYPERLEDGER FABRIC

02

edda INSPIRING excellence

# HF - Agenda

1. Hyperledger Project
2. Components
3. Policies
4. Consensus Algorithm
5. Channel
6. Chaincode
7. Ledger
8. Technologies
9. Architecture
10. Private Data Collection
11. Enforced Validation
12. Summary



edda INSPIRING excellence

# HYPERLEDGER PROJECT

2.01



edda INSPIRING excellence

# Hyperleger Project

- Started in 2015
- Collaborative development of blockchain-based distributed ledgers and tools
- Hosted by Linux Foundation
- Collaborators: *IBM, Intel, Cisco, SAP, Oracle, VMware, Red Hat...*



edda INSPIRING excellence

# Hyperleger Frameworks

besu

*Java-Ethereum client*

Hyperledger INDY

*DLT Platform for digital identities*

Hyperledger IROHA

*C++ DLT platform for digital/financial assets*

Hyperledger FABRIC

*Permissioned DLT platform with custom chaincodes and policies*



edda INSPIRING excellence

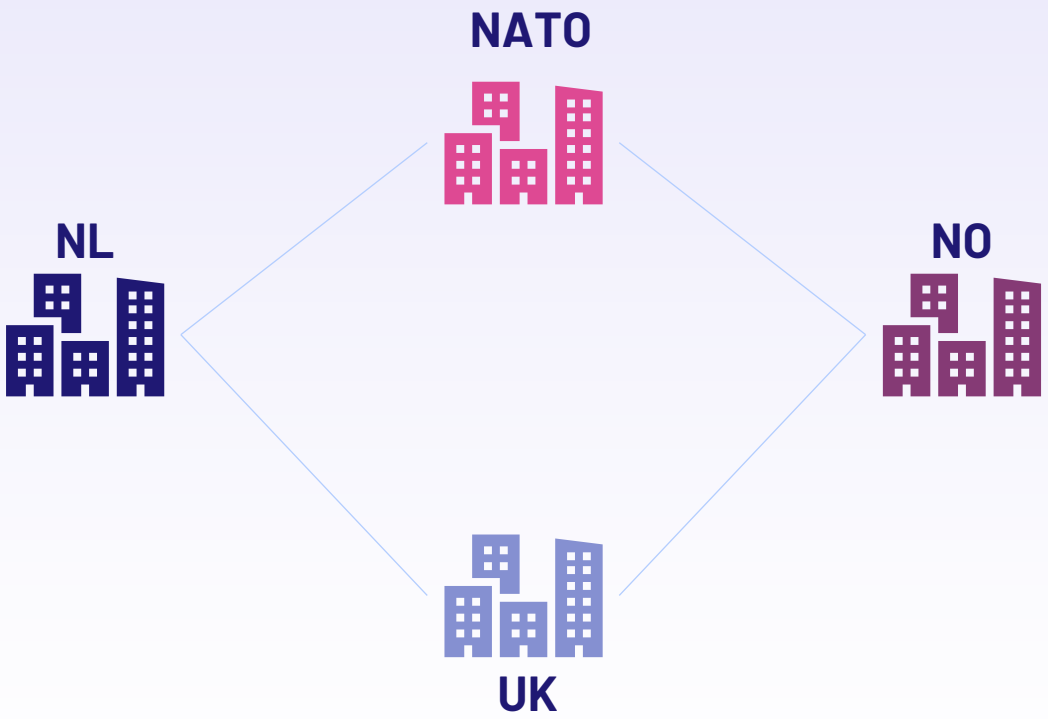
# COMPONENTS

2.02



edda INSPIRING excellence

# Organization




## Organization criteria

- Decentralized
- Autonomous



edda INSPIRING excellence

# Organization’s Components

- Ledger
- Chaincode
- Orderer
- Peer
- Membership Provider (MSP)



edda INSPIRING EXCELLENCE

# Components: Ledger

- State database
  - **LevelDB**:  Key-value (string) storage library. No indexes, no concurrent access
  - **CouchDB**: document-oriented database (key-json). Advanced synchronization and replication features.
  - **Berkeley**: document-oriented database (key-json).
- Blockchain
  - Indexed Files
  - 1 file = 1 block = n transaction(s)



edda INSPIRING excellence

# Components: Chaincode

- Set of Object-Oriented Smart Contract
  - JS/TS, Go, Java supported
  - Deployed on Peers or As-a-Service
- Submitted or evaluated transaction mode
- Evaluate Read/Write operation on State Database
- Context aware
- Shall be deterministic!
- Execution Output: transaction proposal



edda INSPIRING excellence

# Components: Orderer

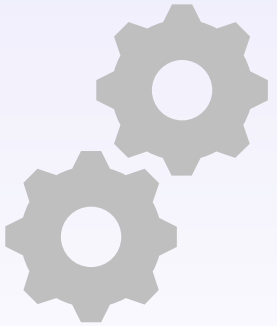
- Transaction Orchestrator
- Collaborates with other Orderer from the network forming the *Ordering Service*.
- Queue and execute transactions based on Channel policies
- Trigger evaluation and commitment of transaction



edda INSPIRING excellence

# Components: Peer

- Computation power of the system
- Anchored to Orderer(s)
- Host or access Chaincode
- Evaluate and commit transaction
- Apply change on Ledger
- Synchronize Ledger with other Peers (internal/external) in case of issues

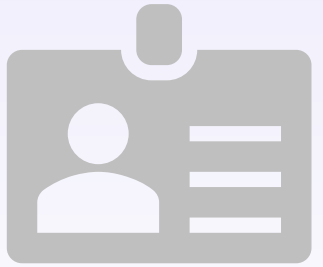




edda INSPIRING excellence

# Components: Membership Provider (MSP)

- Certificate Authority (CA)
- Generate and revoke certificate for
  - **Nodes**: orderer, peer
  - **Roles**: user, admin
- Authorize at Organization and Channel level
- Can enhance certificate with custom attributes
- Trusted by other Organization

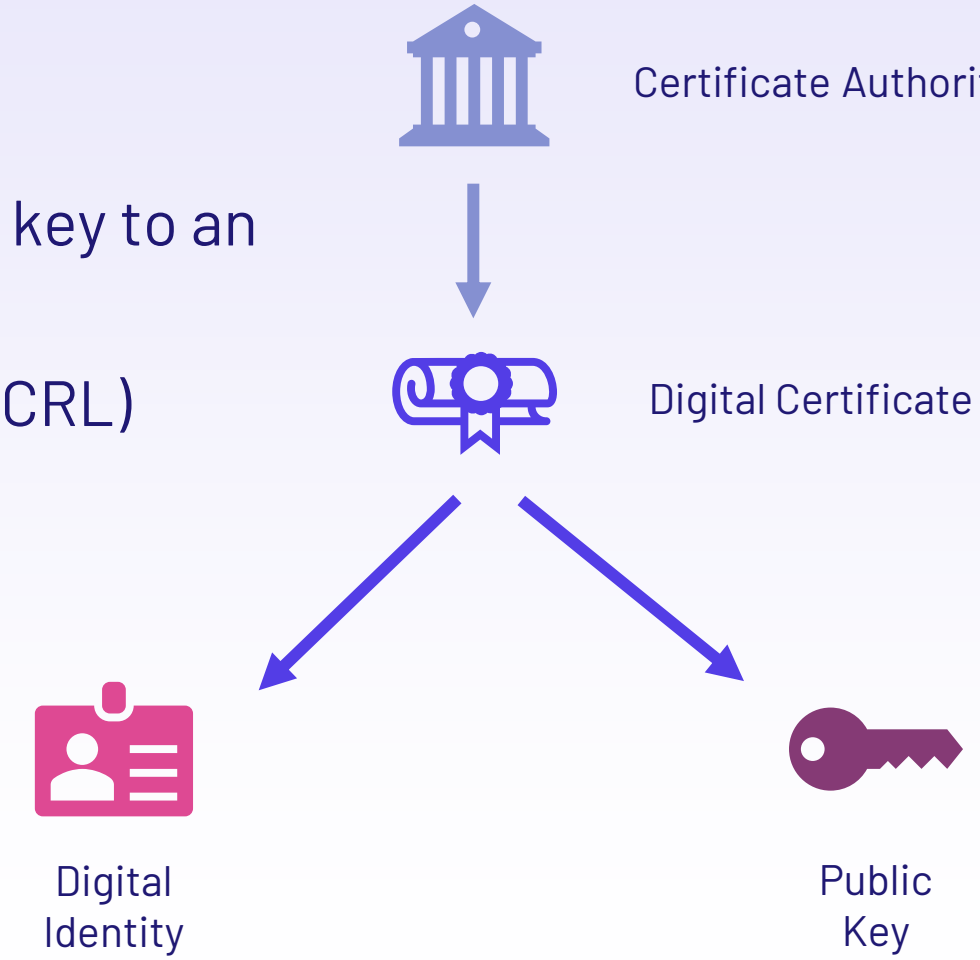




edda INSPIRING excellence

# Certificate Authority (CA)

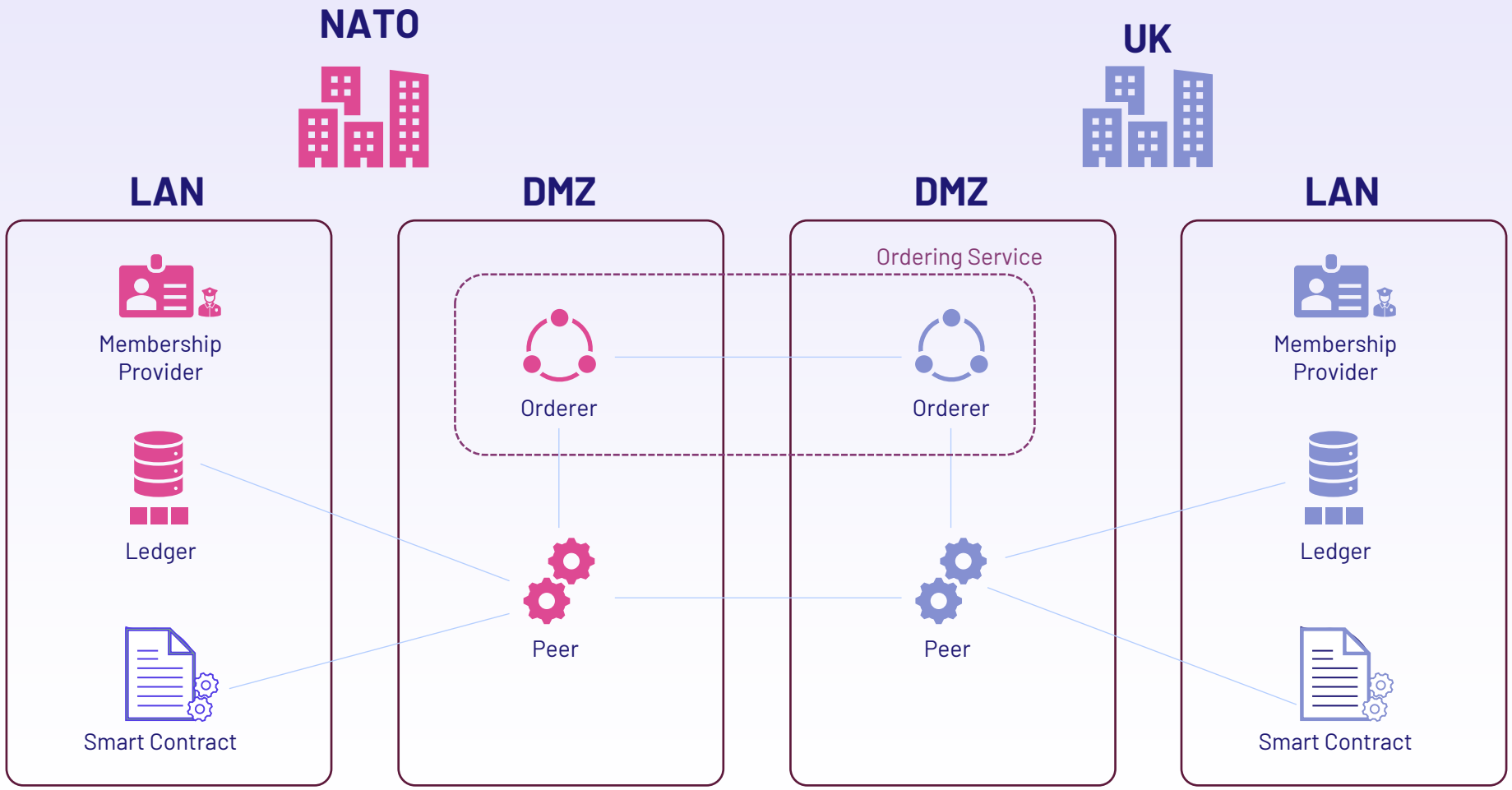
- Third party entity
- Issue certificates that map a public key to an identity
- Maintain a list of valid and revoked (CRL) certificates






edda INSPIRING EXCELLENCE

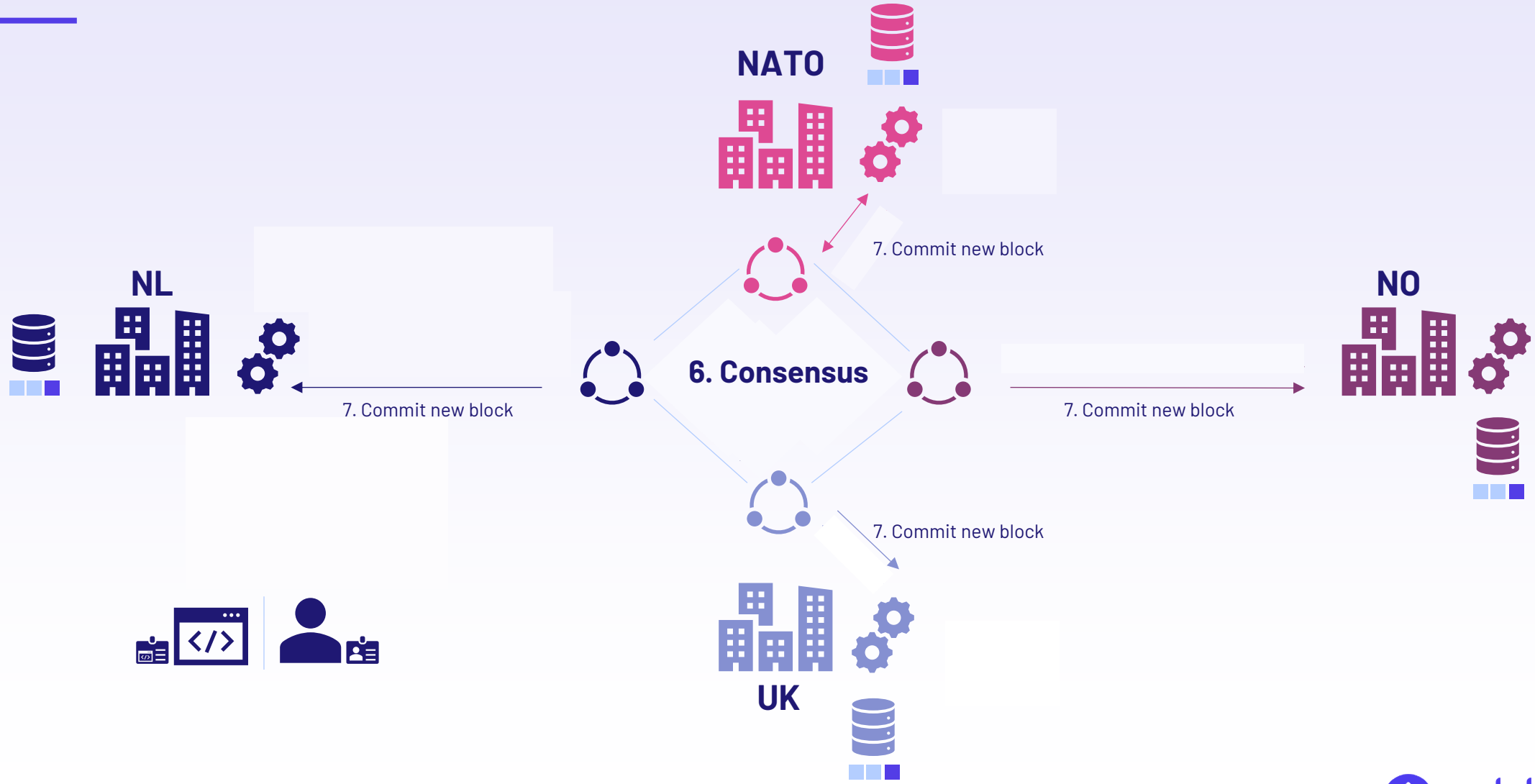
# Components - Overview






edda INSPIRING EXCELLENCE

# Components – How it works?



# POLICIES

2.03



edda INSPIRING excellence

# Policies - Definition

- Mechanism for infrastructure management
- Set of rules that define
  - How decision are made
  - How specific outcomes are reached
  - Who and What



edda INSPIRING excellence

# Policies - Types

- ACLs
  - Allow to configure access to resources (functions)
- Signature Policies
  - Define specific types of users/nodes who must sign
  - Logical syntax with operators (AND, OR, NOutOf)
- ImplicitMeta Policies
  - Only for Channels configuration
  - Specific rules applied on generic users/nodes groups (defined in channel configuration)
  - ANY | ALL | MAJORITY



edda INSPIRING excellence

# Policies - Examples

- ACL

```
# ACL policy for invoking chaincodes on peer
 peer/Propose: /Channel/Application/Writers
```

- Signature

```
# Defining who's a READER for NATO Organization
 Policies: &NATOPolicies
           Readers:
               Type: Signature
               Rule: "OR('NATO.client', 'NATO.peer')"
```

- ImplicitMeta

```
# Require a majority of ADMIN for channel configuration change
 Channel: &ChannelDefaults
     Policies:
          Admins:
                Type: ImplicitMeta
                Rule: "MAJORITY Admins"
```



edda INSPIRING excellence

# Policies - Scope

- Channel Policies
- Channel Modification Policies
- Access Control Lists
- Chaincode Lifecycle Policies
- Chaincode Endorsement Policies.



edda INSPIRING excellence

# CONSENSUS ALGORITHMS 2.04



edda INSPIRING excellence

# Consensus Algorithm

- Used in
  - ImplicitMeta Policies MAJORITY
  - Ordering Service
- CFT (Crash Fault Tolerance)
  - Network resiliency against unavailability of participants.
  - Based on top of RAFT protocol.
- BFT (Byzantine Fault Tolerance)
  - Network resiliency in case of failure or malicious participants

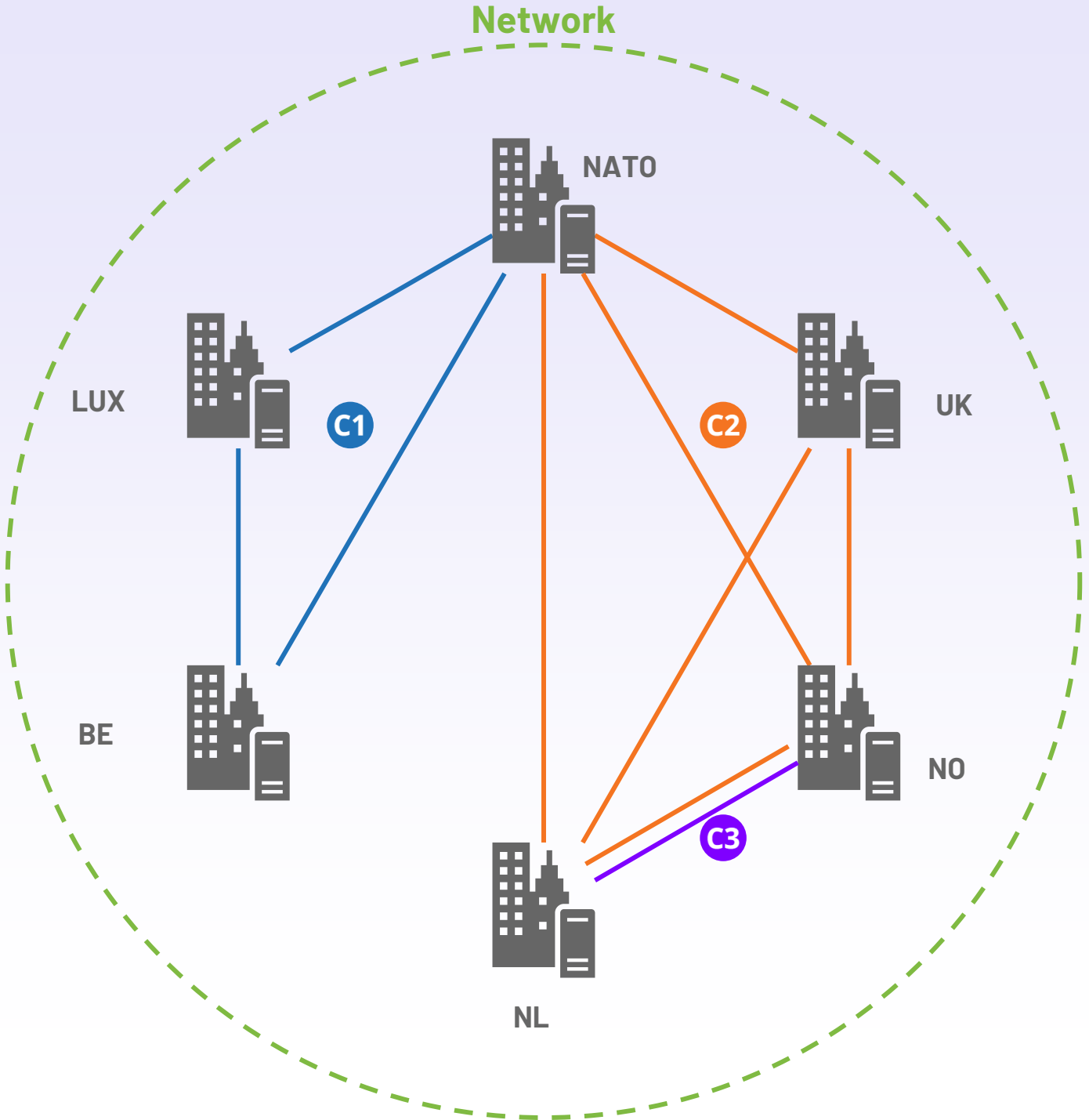
# CHANNEL

2.05



edda INSPIRING excellence

# Channel



# Channel - Concept

- Private sub-network with dedicated Policies
- Each Channel has its own Ledger
- Peer/Orderer can be part of different Channels at the same time



edda INSPIRING excellence

# Channel – Genesis Block

- Contains rulesets to launch the Channel
  - Defining the Organizations' MSP (how a given identity can be associated to an Organization and a role).
  - Defining authority of each Organization (who can endorse and/or consent).
  - Defining Channel configuration (Ordering Service and Consensus Algorithm).
  - Default Smart Contract rules for deployment and execution.
  - Defining rules to manage the Channel (i.e. new member) or change the authoring rules
- Initialized simultaneously on every participants



edda INSPIRING excellence

# Channel Creation - Steps

1. Participant Organizations agree on the Channel Configuration
2. Genesis Block containing the Channel Configuration is generated by an Organization
3. Genesis Block is shared between Organization
4. Each Organization initiate the Channel
   - Orderer join Ordering Service
   - Peer join Channel
   - Peer is anchored to Orderer



edda INSPIRING excellence

# CHAINCODE

2.06



edda INSPIRING excellence

# Chaincode

- Define the rules between different organization in executable code
- Used by Applications/Users to record transaction in the Ledger
- Everything is run on chaincode!
- Chaincode can call another chaincode
- Two types:
  - Application chaincode
  - System chaincode
- Deployed on Peer or As-a-Service



edda INSPIRING excellence

# Chaincode Definition

- **Name**: The name that applications will use when invoking the chaincode.
- **Version**: A version number or value associated with a given chaincodes package. Peers only check that version is unique during installation.
- **Sequence**: The number of times the chaincode has been defined on a channel. This value is an integer and is used to keep track of chaincode upgrades. The sequence number is used by the peer to ensure that all organizations stay in sync regarding the chaincode definitions that they approve and commit.
- **Endorsement Policy**: Custom policies for execution. If not provided default policies is retrieved from the Channel configuration
- **Collection Configuration**: The path to a private data collection definition file associated with your chaincode. For more information about private data collections, see the Private Data architecture reference.
- **ESCC/VSCC Plugins**: The name of a custom endorsement or validation plugin to be used by this chaincode
- **Initialization**: Chaincode methods name that is executed at the initialization



edda INSPIRING excellence

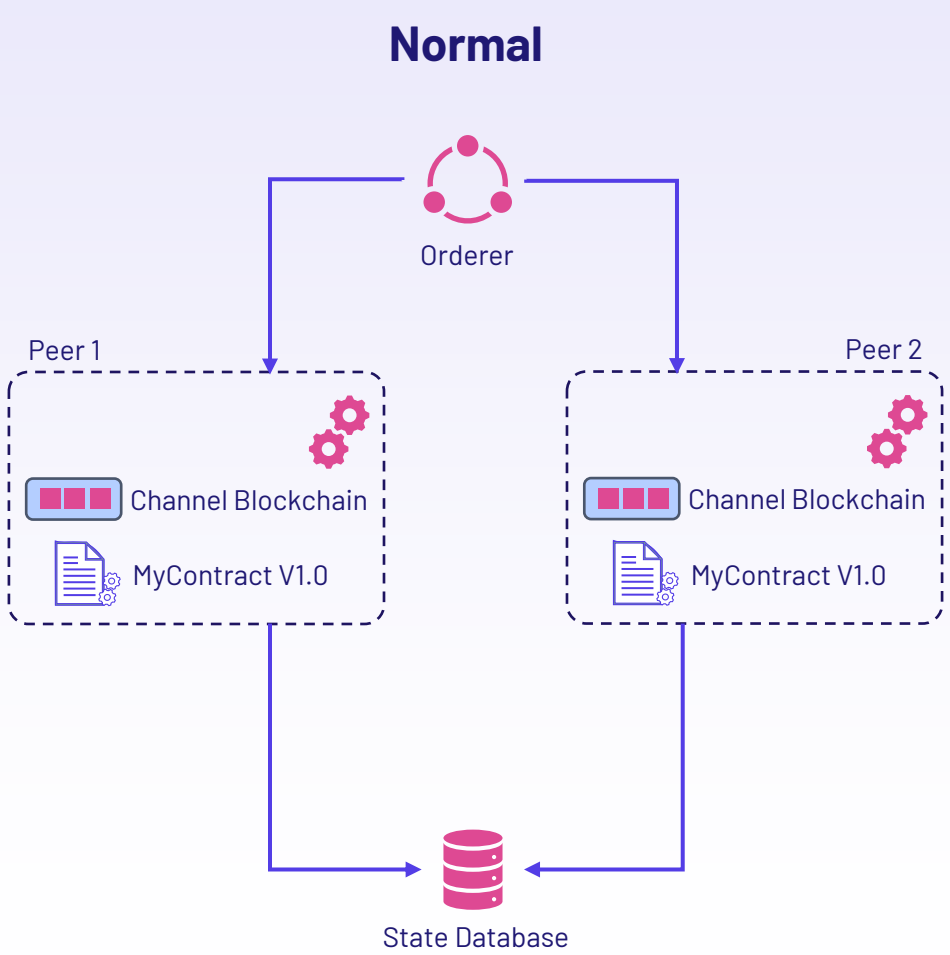
# Chaincode - Lifecycle

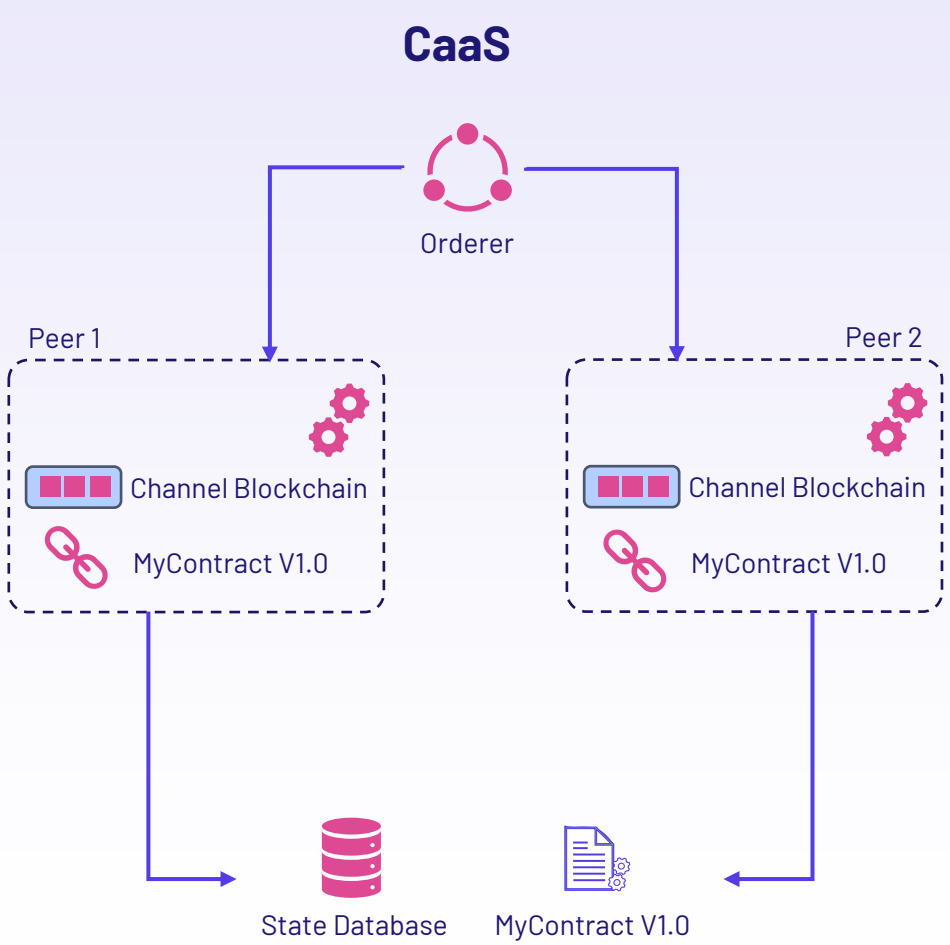
1. **Package the chaincode**: This step can be completed by one organization or by each organization.
2. **Install the chaincode** on your peers: Every organization that will use the chaincode to endorse a transaction or query the ledger needs to complete this step.
3. **Approve a chaincode definition for your organization**: Every organization that will use the chaincode needs to complete this step. The chaincode definition needs to be approved by a sufficient number of organizations to satisfy the channel's LifecycleEndorsement policy (a majority, by default) before the chaincode can be started on the channel.
4. **Commit the chaincode definition to the channel**: The commit transaction needs to be submitted by one organization once the required number of organizations on the channel have approved. The submitter first collects endorsements from enough peers of the organizations that have approved and then submits the transaction to commit the chaincode definition.



edda INSPIRING excellence

# Chaincode – Deployment type






edda INSPIRING excellence

# System Chaincode

- **_lifecycle_ chaincode** runs in all peers and manages the installation of chaincode on your peers, the approval of chaincode definitions for your organization, and the committing of chaincode definitions to channels.
- **Configuration system chaincode** (CSCC) runs in all peers to handle changes to a channel configuration, such as a policy update.
- **Query system chaincode** (QSCC) runs in all peers to provide ledger APIs which include block query, transaction query etc.
- **Endorsement system chaincode** (ESCC) runs in endorsing peers to cryptographically sign a transaction response.
- **Validation system chaincode** (VSCC) validates a transaction, including checking endorsement policy and read-write set versioning.



edda INSPIRING excellence

# Application Chaincode – Example

```
# JS Chaincode example
 async CreateAsset(ctx, id, color, size, owner, appraisedValue) {
        const asset = {
            ID: id,
            Color: color,
            Size: size,
            Owner: owner,
            AppraisedValue: appraisedValue,
        };
        return ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
    }
```

- Implements Fabric Chaincode Interface
- Context allows to:
  - PUT, GET, QUERY from the Ledger State Database
  - Retrieve information about the transaction initiator (certificate)
  - Get transaction ID, Channel ID and inputs



edda INSPIRING excellence

# LEDGER

# 2.07



edda INSPIRING excellence

# Ledger

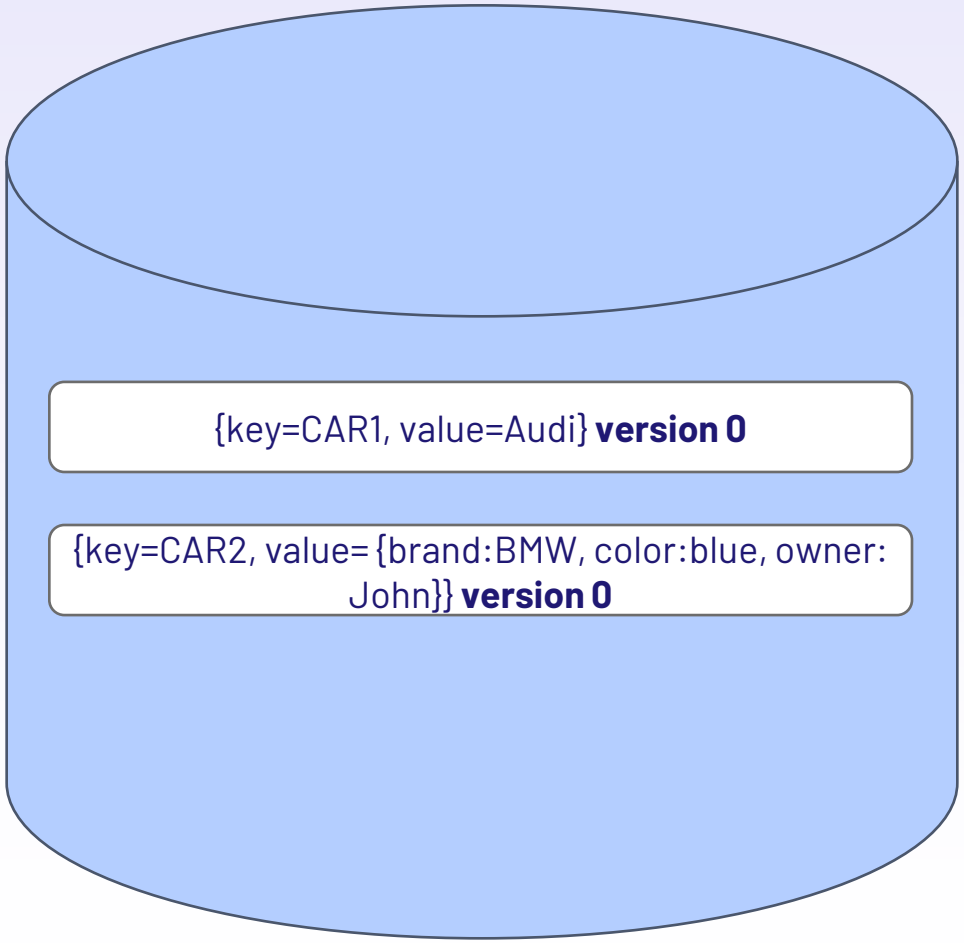
- State Database
- Blockchain
- Blocks
- Transactions



edda INSPIRING excellence

# Ledger – State Database

- NoSQL
- Key-value
- Versioning
- GET, PUT, DELETE
- Key can be multi-attributes
- Standalone component

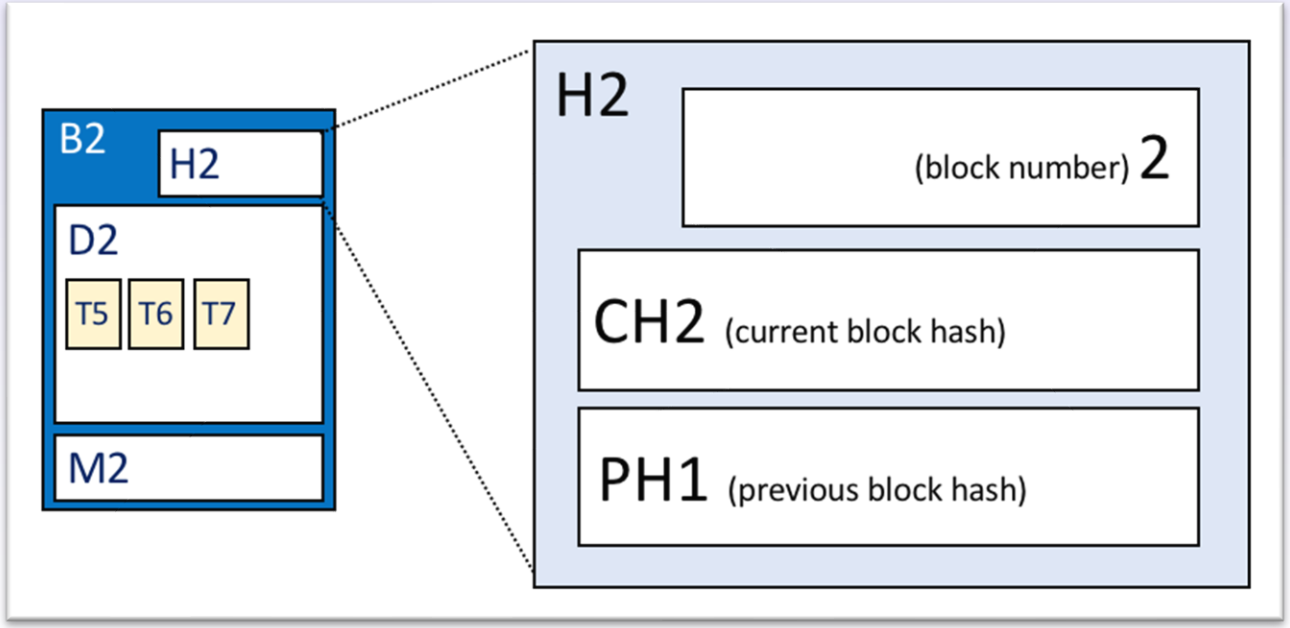





edda INSPIRING excellence

# Ledger – Blockchain - Block

- Indexed files
- Hosted on Peers



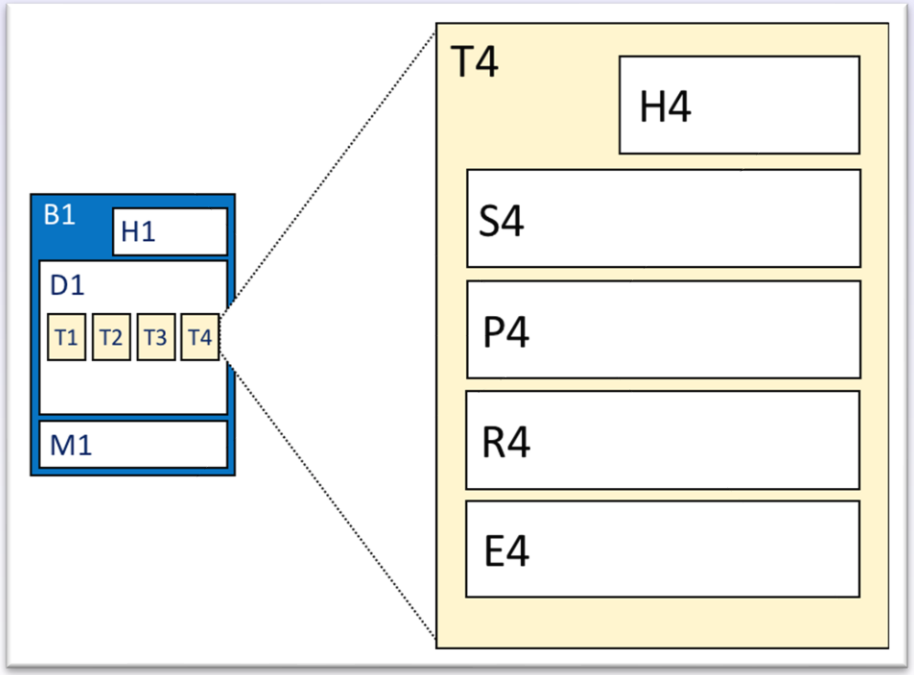

- **Block Header**
  - Block Number
  - Current Block Hash
  - Previous Block Hash
- **Block Data** = list of transactions
- **Block Metadata** = Certification and signature of Block Creator



edda INSPIRING excellence

# Ledger – Blockchain - Transaction

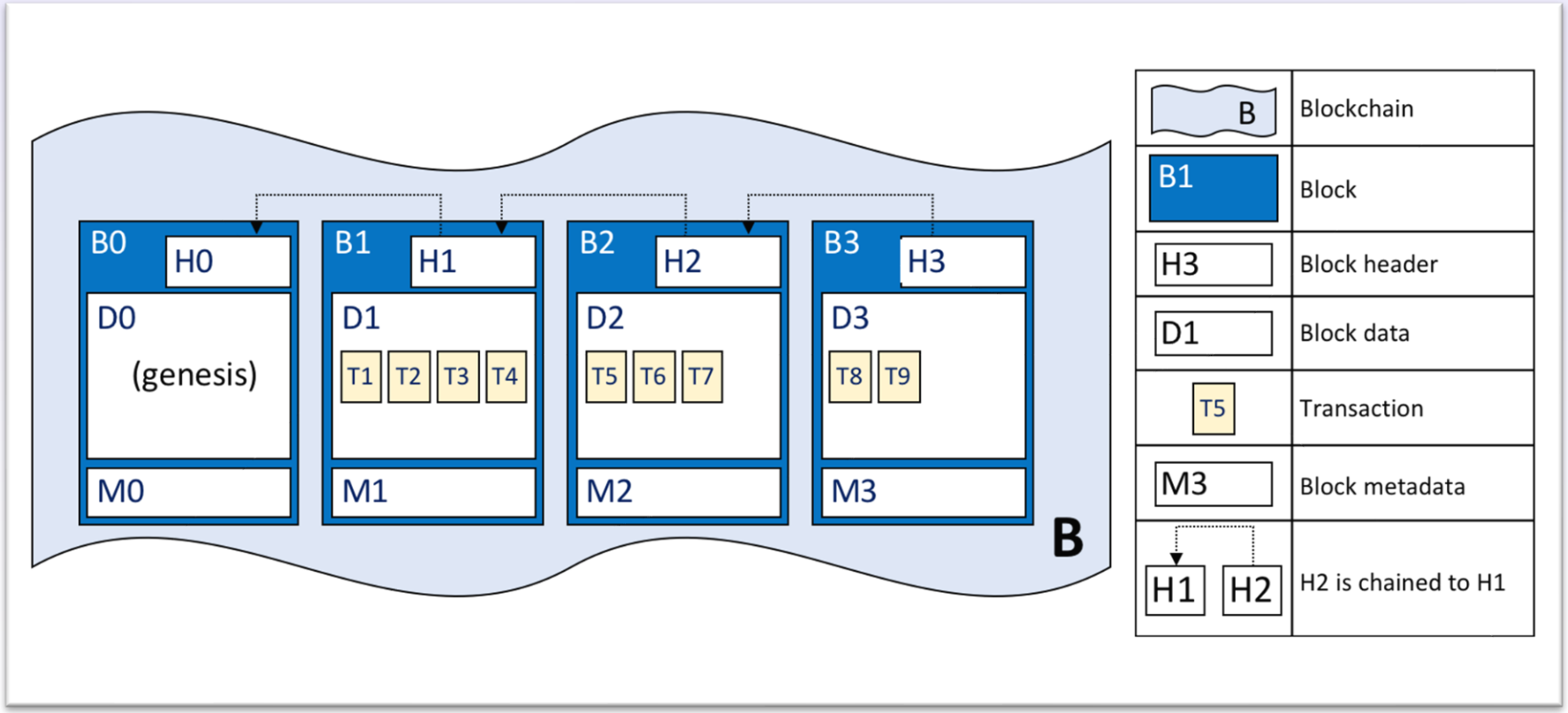
- **Header** = chaincode name and version
- **Signature** of Transaction initiator
- **Proposal** = inputs parameters
- **Response** = computed outputs
- **Endorsement** = list of signed response from other organization






edda INSPIRING excellence

# Ledger – Blockchain - Overview






edda INSPIRING excellence

# TECHNOLOGIES

2.08



edda INSPIRING excellence

# Technologies

| Role | Technology | Vendor | Protocols |
|---|---|---|---|
| **Ledger** | LevelDB<br>CouchDB<br>Berkeley | Apache Foundation<br>Apache Foundation<br>Oracle | HTTPS |
| **Membership Provider** | Hyperledger Fabric CA<br>HSM | The Linux Foundation | HTTPS |
| **Orderer** | Hyperledger Fabric<br>Orderer | The Linux Foundation | HTTPS |
| **Peer** | Hyperledger Fabric Peer | The Linux Foundation | GRPCS/HTTPS |
| **Smart Contract** | Java<br>NodesJS<br>Go | | HTTPS |

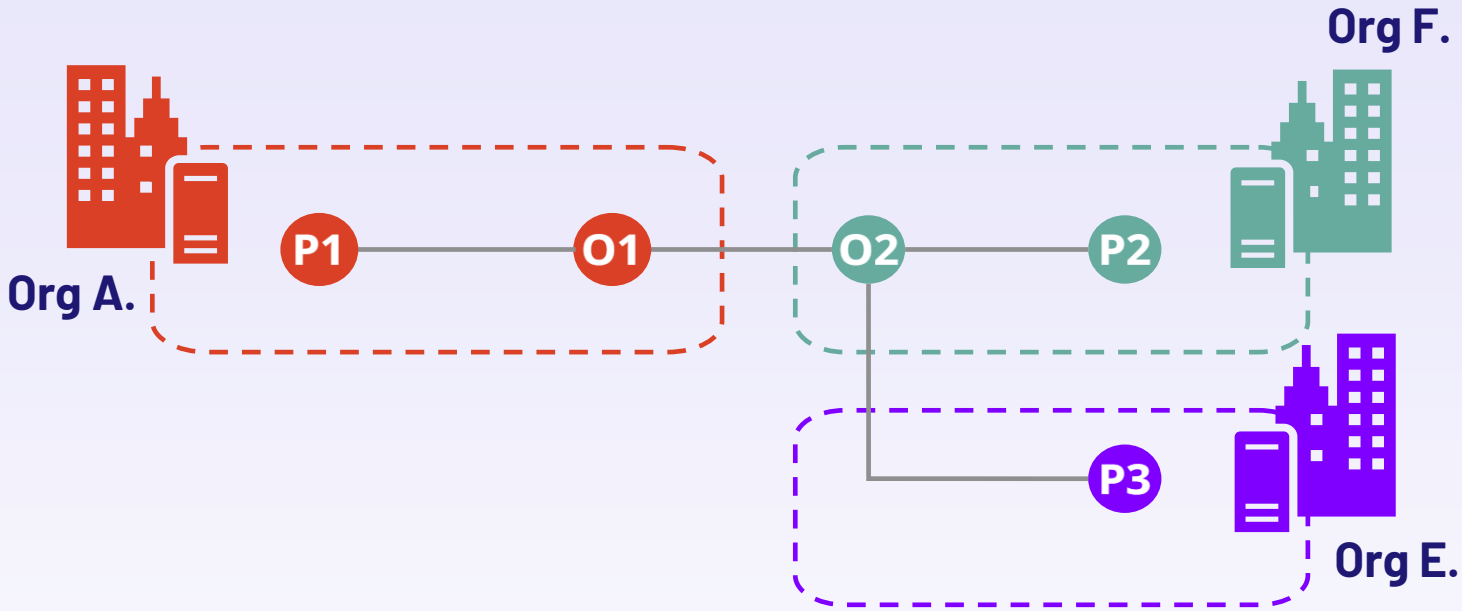
# ARCHITECTURE

2.09



edda INSPIRING excellence

# Architecture




- **Majority/unanimity**: all change on the Channel/Ledger shall be endorsed by a majority or by all Peers (P1, P2, P3).
- **Fail-over/Load-balancing**: Organization F can rely on Organization E in case of failure.
- **Audit**: Organization E acts as an observer/auditor and can only participate to endorsement of transaction as a third-party (can't submit transaction).



edda INSPIRING excellence

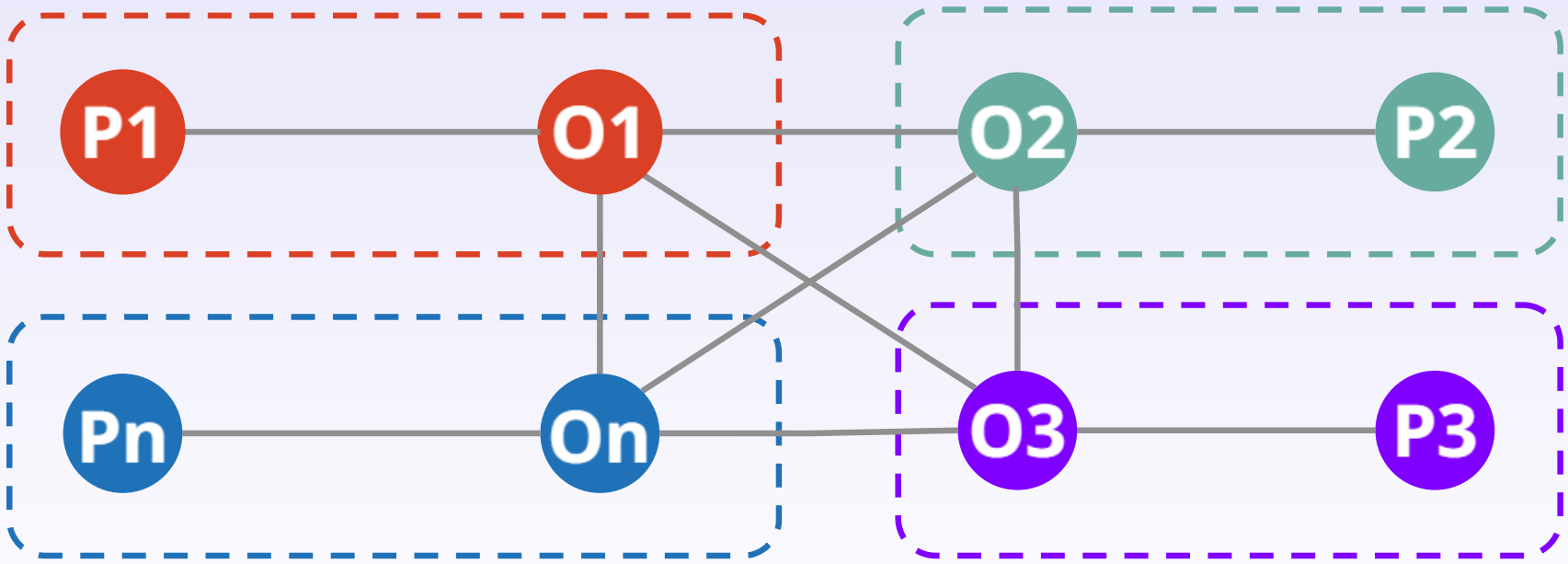
# Architecture

- Targets
  - Decentralized Autonomous Organizations (DAO)
  - Specialized Organizations
  - Blockchain-as-a-Service (BaaS)
- Topology



edda INSPIRING EXCELLENCE

# Target - DAO

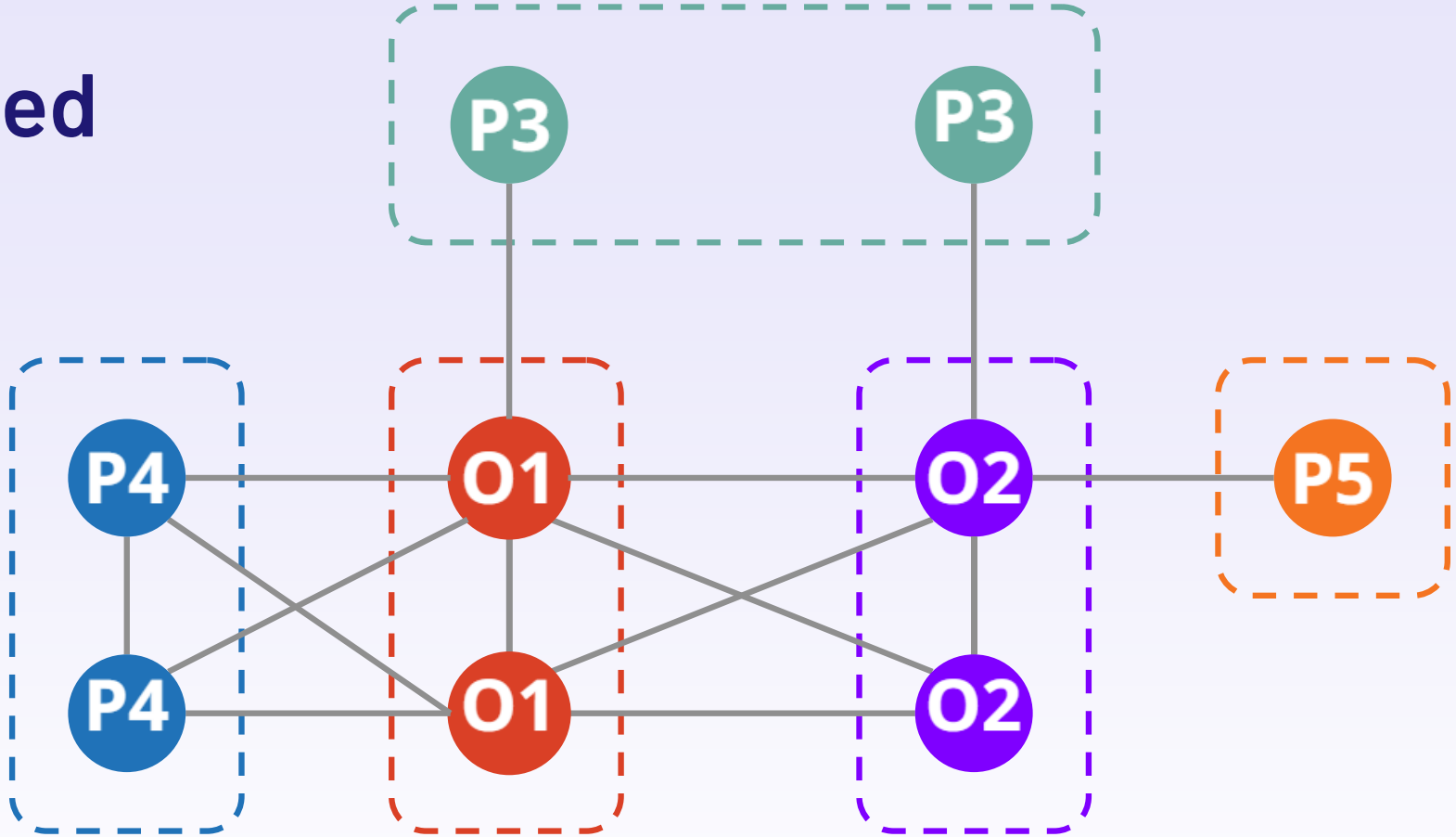



- PRO:
  - Anarchic Governance by design
  - High censorship and tamper resistance
- CONS:
  - Alteration shall be endorsed by a majority
  - Each Organization shall provided services for Ordering and Peering



edda INSPIRING excellence

# Target - Specialized

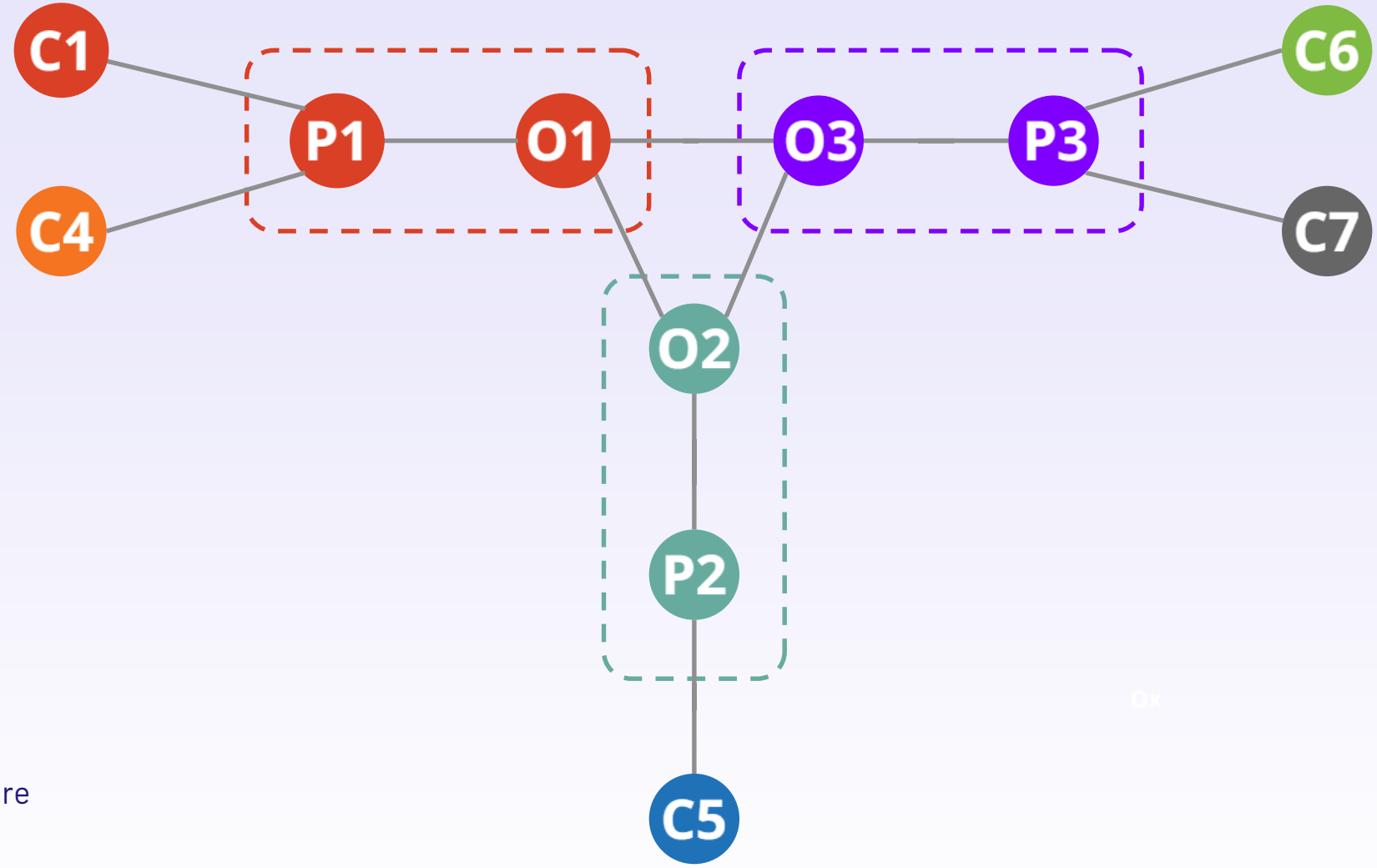



- PRO:
  - Reduce maintenance cost
  - Simplify requirements needed to an Organization to join the network
  - Service continuity
- CONS:
  - Ordering Service can know the content of transaction
  - Ordering Service can put pressure on Peering



edda INSPIRING excellence

# Target - BaaS




- PRO:
  - Cheap for Clients
  - Mutualized infrastructure
- CONS:
  - Organization can put pressure on Clients (revocation, limitation)
  - Ordering Service can put pressure on Peering
  - Capacity Management



edda INSPIRING excellence

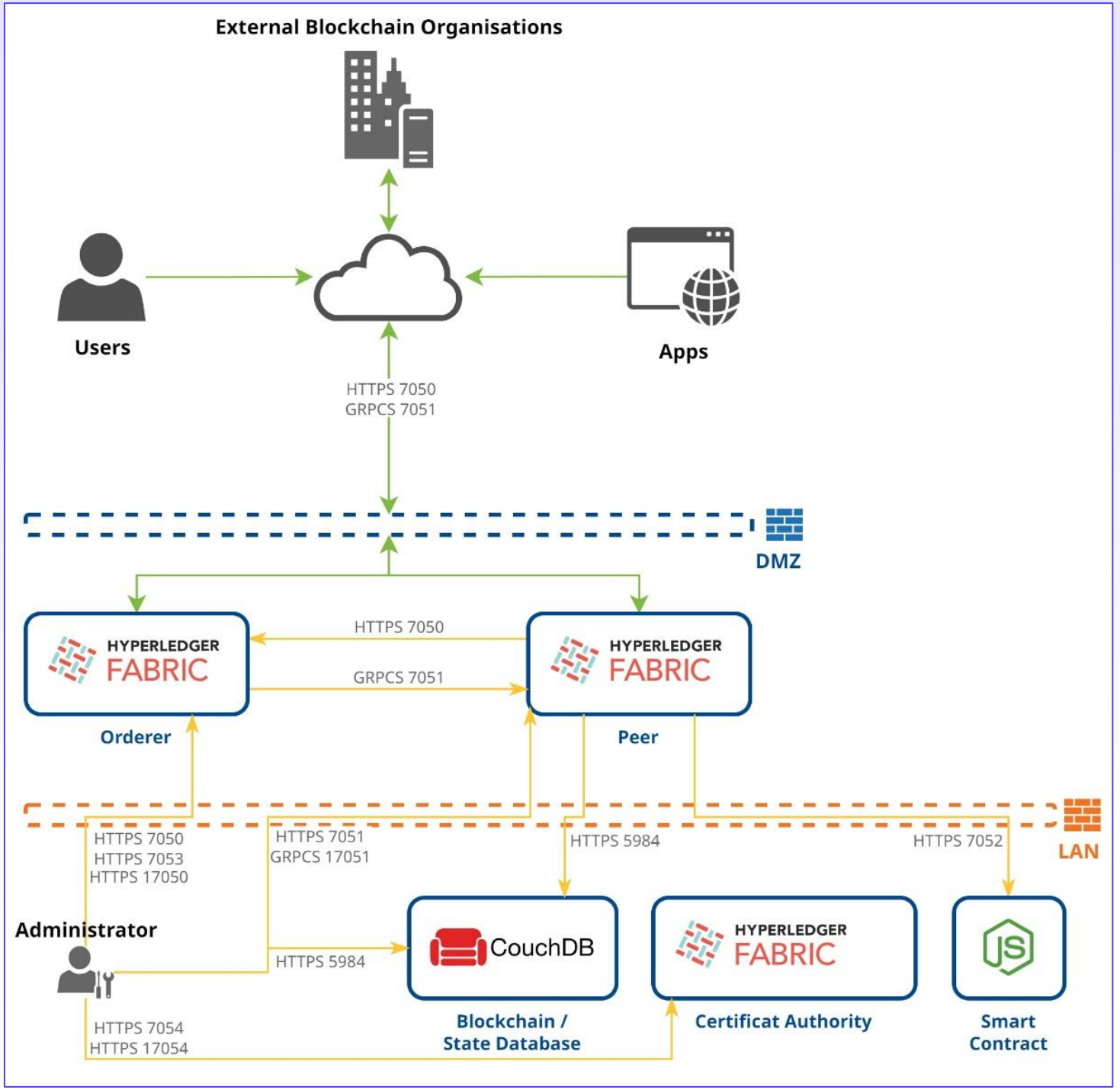
# Architecture - Conclusion

- High modularity of Fabric is its biggest **strength:**
  - Can match any functional and non-functional requirements
  - Any Business case can be covered with Chaincode and Channel Policies
- and its biggest **weakness**:
  - Can be too complex
  - Can deadlock
- Blockchain network impacts Organization Governance:
  - Change Management
  - Incident Management
  - Capacity Management
  - and more...



edda INSPIRING excellence

# Topology






edda INSPIRING EXCELLENCE

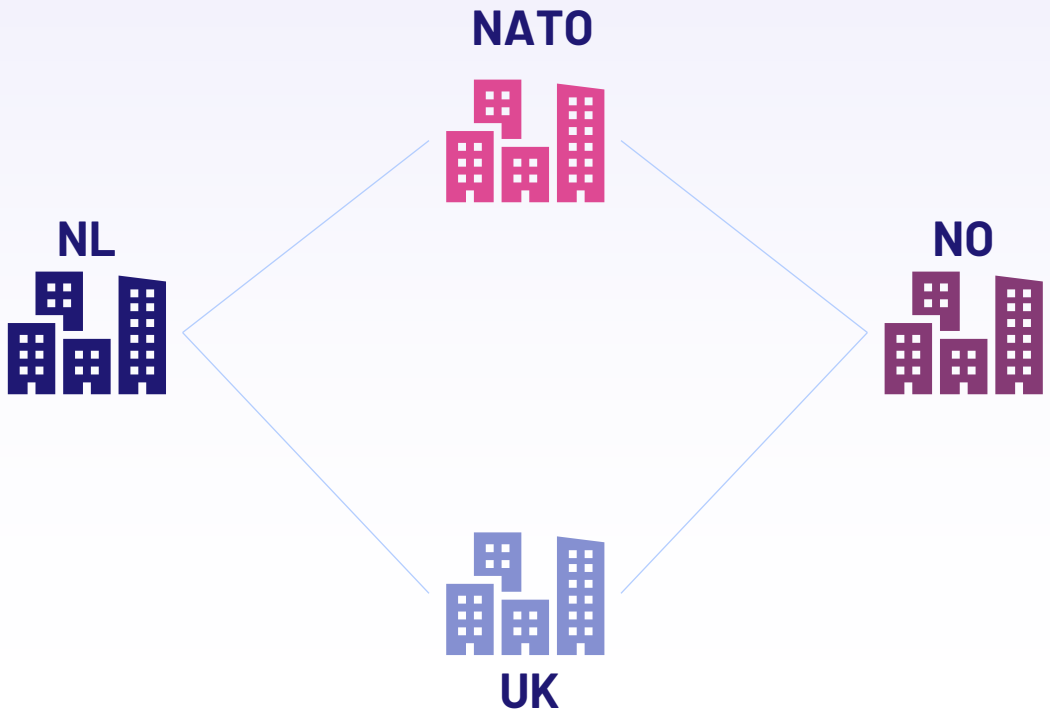
# PRIVATE DATA COLLECTION 2.10



edda INSPIRING excellence

# Scenario

- Four Organization Channel running a Chaincode (CatalogItem)
- Organization can publish Catalog Item's on the Channel Blockchain
- **Problem**: Some Items attributes are confidential

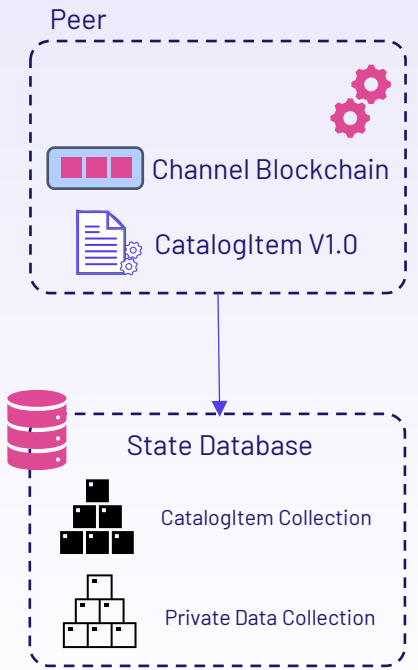





edda INSPIRING excellence

# Private Data Collection (PDC)

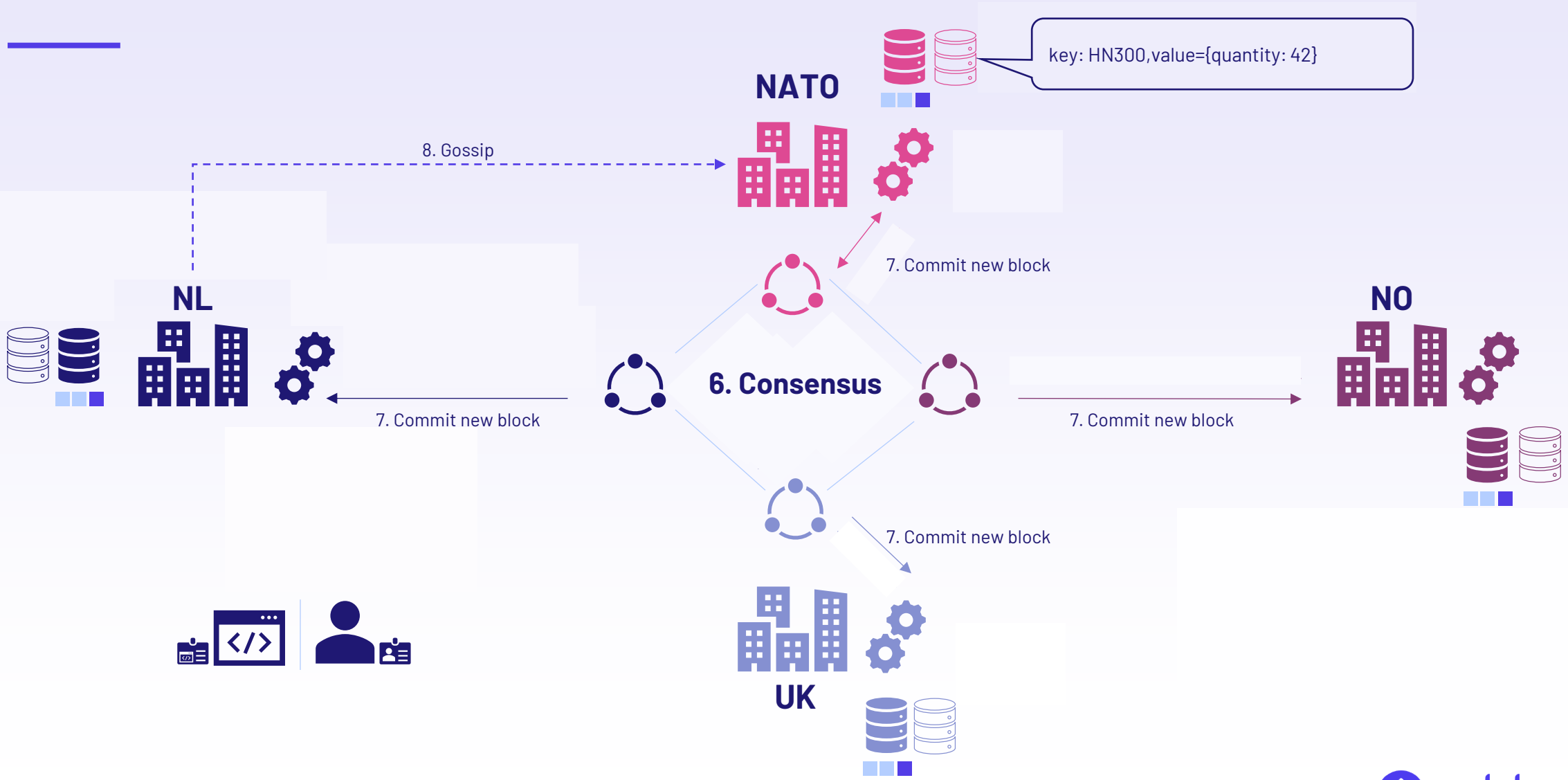
- Chaincode function that allow sharing Private Data between members of a same Channel
- PDC are dedicated schema in State Database
- Organization can write Data in other PDC through Chaincode execution
- Only PDC Owner can read data (by default)
- PDC can be:
  - Implicit per Channel
  - Defined (at creation) in Channel Policies, i.e:
    - PDC1: NATO,NL,NO
    - PDC2: NL,NO,UK






edda INSPIRING EXCELLENCE

# PDC – How it works?






edda INSPIRING excellence

# PDC: Limitations

- Ledger operation on public and private data can’t be mixed in the same transaction
- Implicit PDC can be used by any member of the Channel (if validated by Chaincode execution)



edda INSPIRING excellence

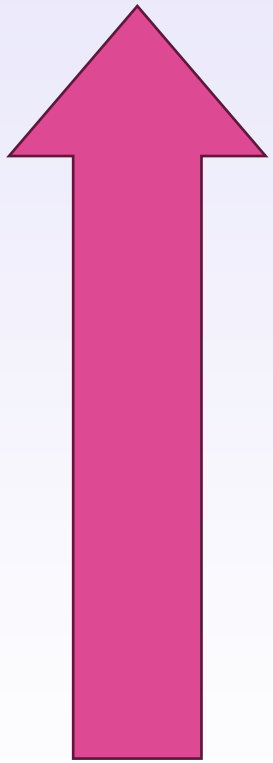
# ENFORCED ENDORSEMENT 2.11



edda INSPIRING excellence

# Endorsement Policies Level

1. Channel-level
   - At Channel creation
2. Chaincode-level
   - At chaincode deployment (override Channel Policies)
3. Collection-level
   - Apply only on defined Private Data Collection (Channel Policies)
4. Key-level
   - During runtime



**Override**



edda INSPIRING excellence

# Key Level

- Allow organization to change endorsement policy for a given key during runtime
- Key-level policy can be less or more restrictive than chaincode/collection/channel policies
- Active if the key's value is modified
- Can be removed

Ex: An Organization wants to be part of the endorsement process for a given key during certain steps of a Business Process.



edda INSPIRING excellence

# HF CAPABILITIES SUMMARY

# 2.12



edda INSPIRING excellence

# Fabric Capabilities – Summary 1/3

## Identity Management

- **Model:** Decentralized identity management
- **Granularity:** Nodes/users level
- **Identity:** X.509 Certificate standard
- **Identity Issuance:** Membership Provider (MSP)
- **Authorization:** Based on role and custom attributes

## Consensus Algorithms

✓ CFT

✗ BFT (planned for 3.x)

## Ledger Storage

- **Replication:** Complete
- **Technologies:** LevelDB / CouchDB / Berkeley (Oracle Blockchain)



edda INSPIRING excellence

# Fabric Capabilities – Summary 2/3

**Data Privacy**
Channel + Private Data Collection

**Network Governance**

- **Topology:** Channel based policy
- **State propagation:** Gossip protocol (HTTPS)
- **Protocols:** HTTPS, GRCPS
- **Transaction Ordering:** Orderer nodes

**Development**

- **Contract supported:** Java / Go / JavaScript/ TypeScript
- **Models:** Transactional (Execute-Order-Validate)
- **SDK libraries:** Java / Go / JS / TS / Python



edda INSPIRING excellence

# Fabric Capabilities – Summary 3/3

**No SDK / Require custom implementation:**

- Tokenization
- Attachments
- Oracle Services



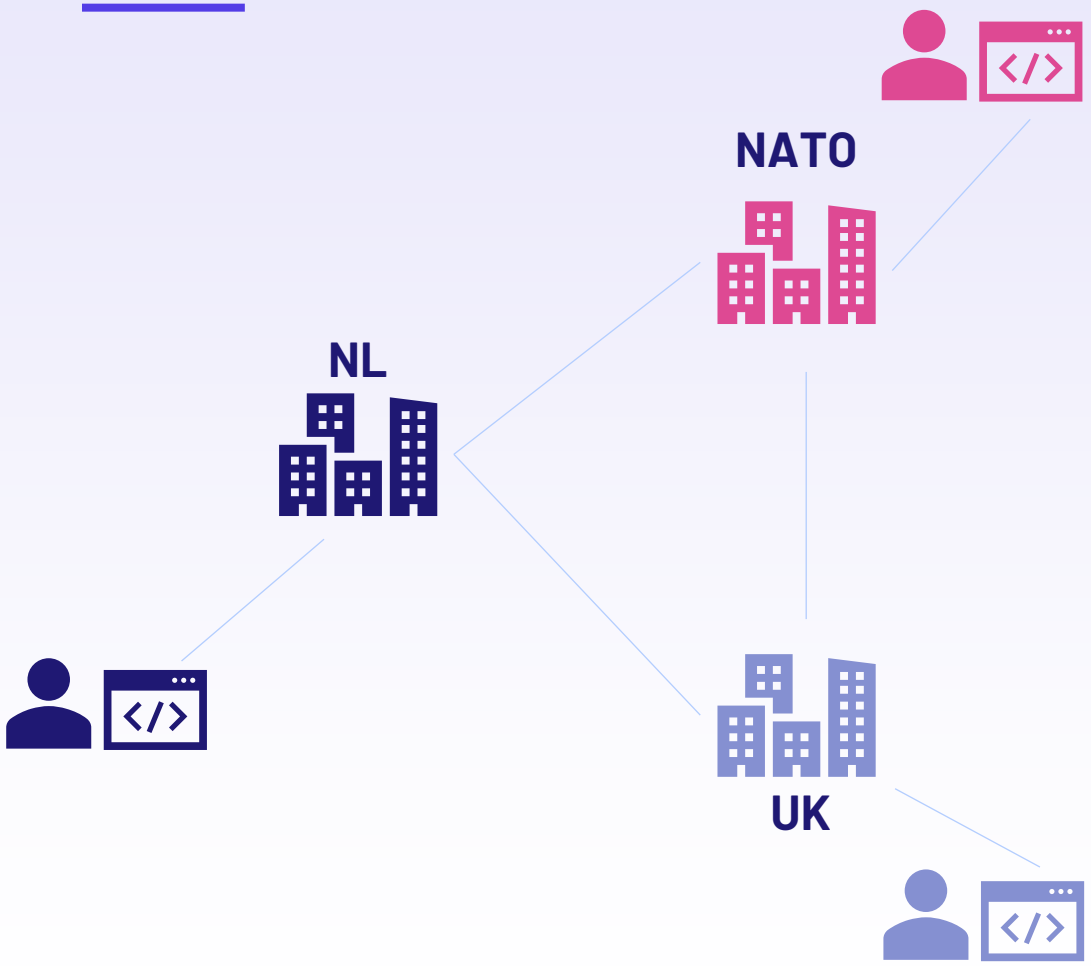
edda INSPIRING excellence

# NSPA DEMONSTRATORS

03

edda INSPIRING excellence

# Architecture




- LAB Network: permissioned blockchain network with 3 organizations (NATO, UK, NL)
- Dedicated channel per Smart Contract
- Dedicated web application deployed on each organization to interact with the network



edda INSPIRING excellence

# Projects

- Asset Tracking
  - Track history of assets activities.
  - Data Access Control and Workflow based on chaincode.
- Custom Forms F302
  - Digitalization of the F302 paper process.
  - Data Access Control and Workflow based on chaincode.
- Private Catalogue (under development)
  - Virtual asset catalogue with confidential data that could be shared between nation on request.
  - Confidential data handle using chaincode controlled private data collection.
- User-friendly Blockchain Explorer



edda INSPIRING excellence

# SET UP TEST ENVIRONMENT

04

edda INSPIRING excellence

# Agenda

1. Install prerequisites
2. Download sources and binaries
3. Deploy your organization
4. Play with components
5. Find some partners!
6. Configure network
7. Collaborate to create Genesis Block
8. Share Genesis Block
9. Build Ordering Service
10. Anchor Peers



edda INSPIRING excellence

# Prerequisites

- Docker
- Windows with WSL2 installed OR Linux Operating system
- IDE (VSCode, NotePad++,....)



edda INSPIRING excellence

# Sources

- git clone https://github.com/jbledda/bc-training
- ./install-fabric.sh



edda INSPIRING excellence

# Files/Directories

- install-fabric.sh
  - Script that installs binaries, docker images and download fabric samples
- fabric-samples/
  - Directory with all the educative samples provided by Fabric
  - Allow you to run a test network with two Organization on your workstation
- network-builder/
  - Allow you  to run your own Organization and connect with other
  - Used in the NSPA Blockchain POC
  - Provided by Edda
- tools/explorer:
  - Blockchain Explorer tool



edda INSPIRING excellence

# fabric-samples

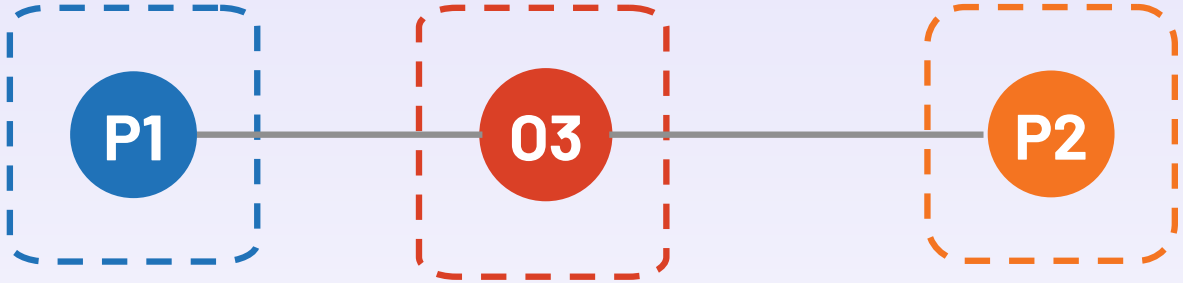
- asset-*/, auction*/
  - Lot of chaincode examples, check README
- token-*/
  - Example of Token implementation
  - ERC-20: Token (ex: create your own cryptocurrency)
  - ERC-721: NFT
  - ERC-1155: Multi-token
  - UTXO
- bin/:
  - Downloaded Fabric binaries (include administration tools)
- test-network/
  - Allow you to run a development network on your workstation



edda INSPIRING excellence

# test-network - Architecture

- **Org 1**:
  - Peer
  - Ledger
- **Org 2**:
  - Peer
  - Ledger
- **OrdererOrg**:
  - Orderer






edda INSPIRING excellence

# Practice I – Let’s run a test network! 1/2

```
cd fabric-samples/test-network
```

```
# Start a test network with two Organization (Org1 and Org2) both using a Certifcate
Authority and CouchDB as State Database.
./network.sh up -ca -s couchdb
```

```
# Create a channel named mychannel and endorse each organization inside
./network.sh createChannel -c mychannel
```

```
# Deploy a sample chaincode (asset-transfer-basic)
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -
ccl javascript
```



edda INSPIRING excellence

# Practice I – Let's run a test network! 2/2

```
# Set environment variable for Fabric Administration Tools
export PATH=${PWD}/../bin:${PWD}:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
```

```
# Set environment variables to authenticate as Org1 Admin
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

```
# Query Peer to obtain list of joined Channel
peer channel list
```

```
# Query Peer to obtain list of installed chaincode on Channel mychannel
peer lifecycle chaincode querycommitted -C mychannel
```

```
# Detroy everything!
./network.sh down
```



edda INSPIRING excellence

# BUILDING A CHANNEL

05

edda INSPIRING excellence

# Practice II – Building a Channel

1. Create your own Organization with the Network Builder
2. Find some partners!
3. Collaborate together to create Genesis Block
4. Share Genesis Block
5. Build Ordering Service
6. Anchor Peers



edda INSPIRING excellence

# Network Builder – Folders/Files

- configtx/configtx.yaml
  - Example of Genesis Block configuration for a CFT Channel of three Organizations
- organization/
  - Contains all configuration and certificates of your organization nodes and users



edda INSPIRING EXCELLENCE

# Network Builder - Commands

```
# CONFIG: duplicate bin/ and config/ from fabric-samples into the root
directory
cp -r fabric-samples/bin .
cp -r fabric-samples/config .
```

```
# Create your own Organization (peer, orderer, couchdb, CA + administrator accounts)
./network.sh up -org <yourOrgName> -orgdomain <yourorg.domain.com>
```

```
# Load all environment variables to administrate your Organization.
. ./setOrgEnv.sh <org-name> <org-domain>
```

```
# Destroy your Organization
./network.sh down
```

```
# Export config for Genesis Block Creation into export folder
./network.sh export -org <yourOrgName> -orgdomain <yourorg.domain.com>
```



edda INSPIRING excellence

# Building Channel – Network configuration

```
# Obtain your network
Windows: ipconfig
Linux: ifconfig
```

```
# Edit your Host file to add your and partner custom domains
# Windows/WSL
C:\Windows\System32\drivers\etc\host

# Linux
sudo vi /etc/hosts
```

```
# Minimal Config
192.168.0.3 peer0.bc.army.mod.uk
192.168.0.3 orderer.bc.army.mod.uk
192.168.0.3 ca.bc.army.mod.uk
192.168.0.3 couchdb0.bc.army.mod.uk
```



edda INSPIRING EXCELLENCE

# DEPLOY CHAINCODE

06

edda INSPIRING excellence

# Practice III – Deploy chaincode on Channel

1. Dev/select a chaincode with your partners
2. Test it and deploy it on your dev network
3. Deploy it on your POC network
4. Bonus: Deploy the Blockchain Explorer



edda INSPIRING excellence