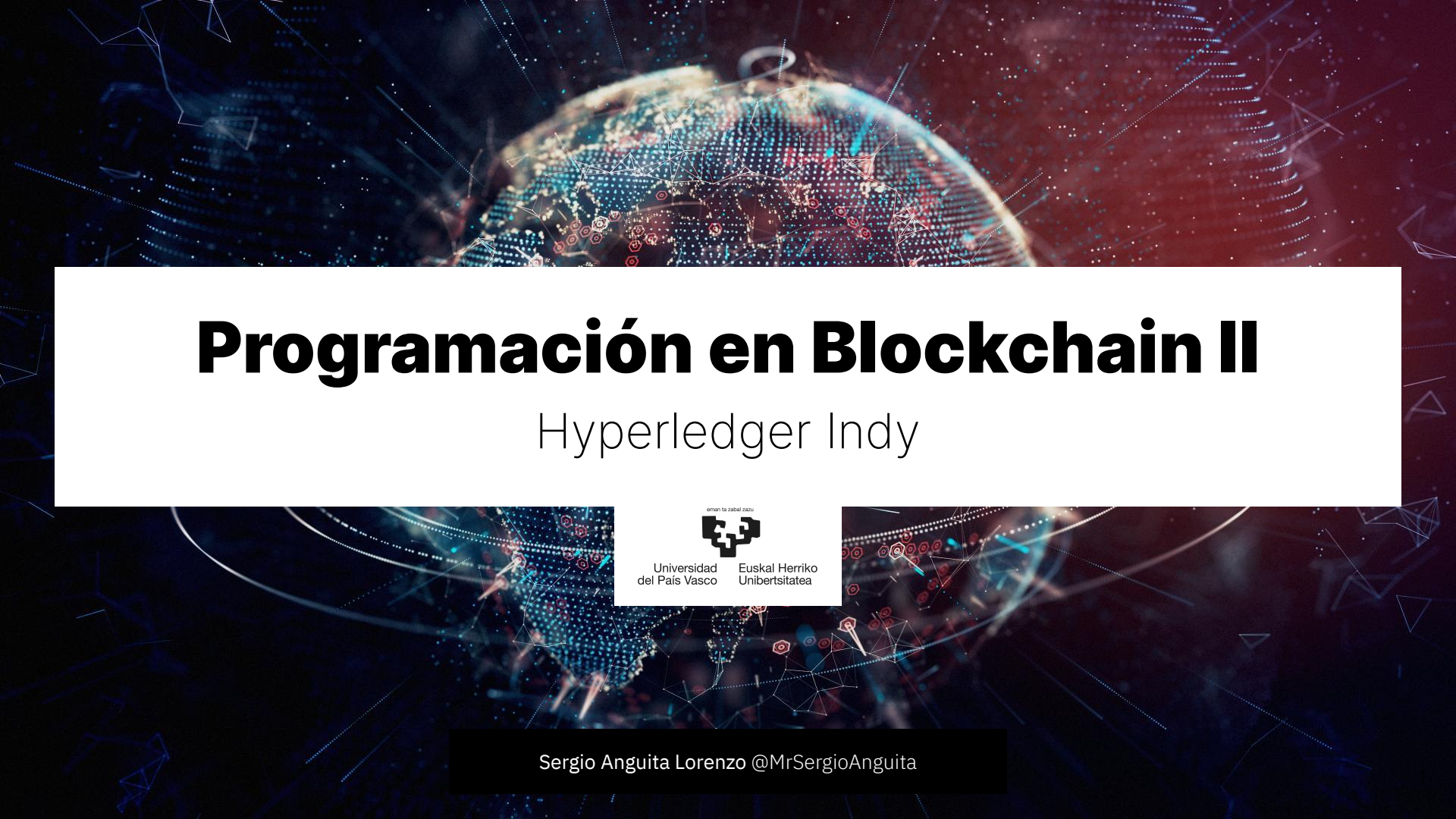


Programación en Blockchain II

Hyperledger Indy

oman ta zabal zazu

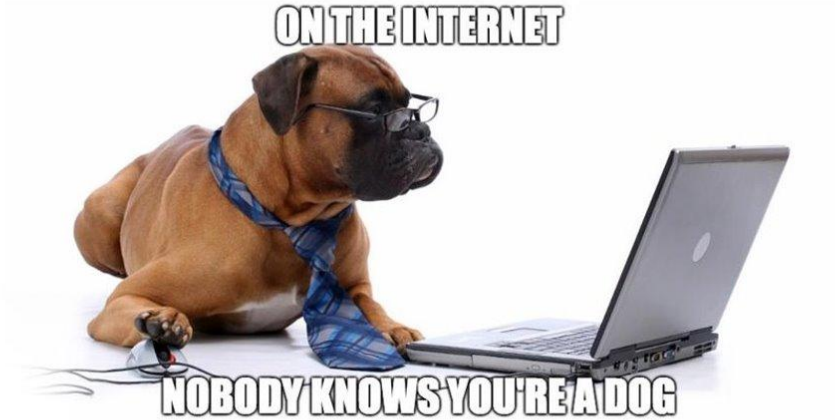
Universidad del País Vasco

Euskal Herriko Unibertsitatea

Sergio Anguita Lorenzo @MrSergioAnguita

¿Que resuelve SSI?

Que resuelve SSI

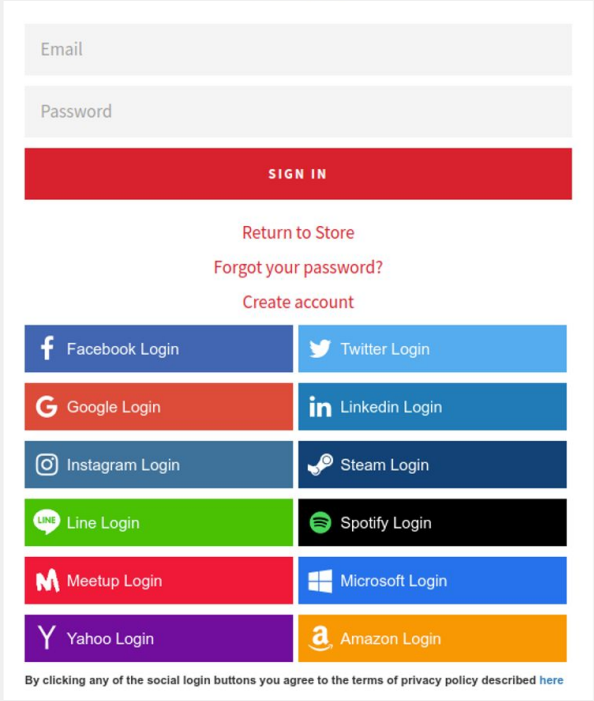


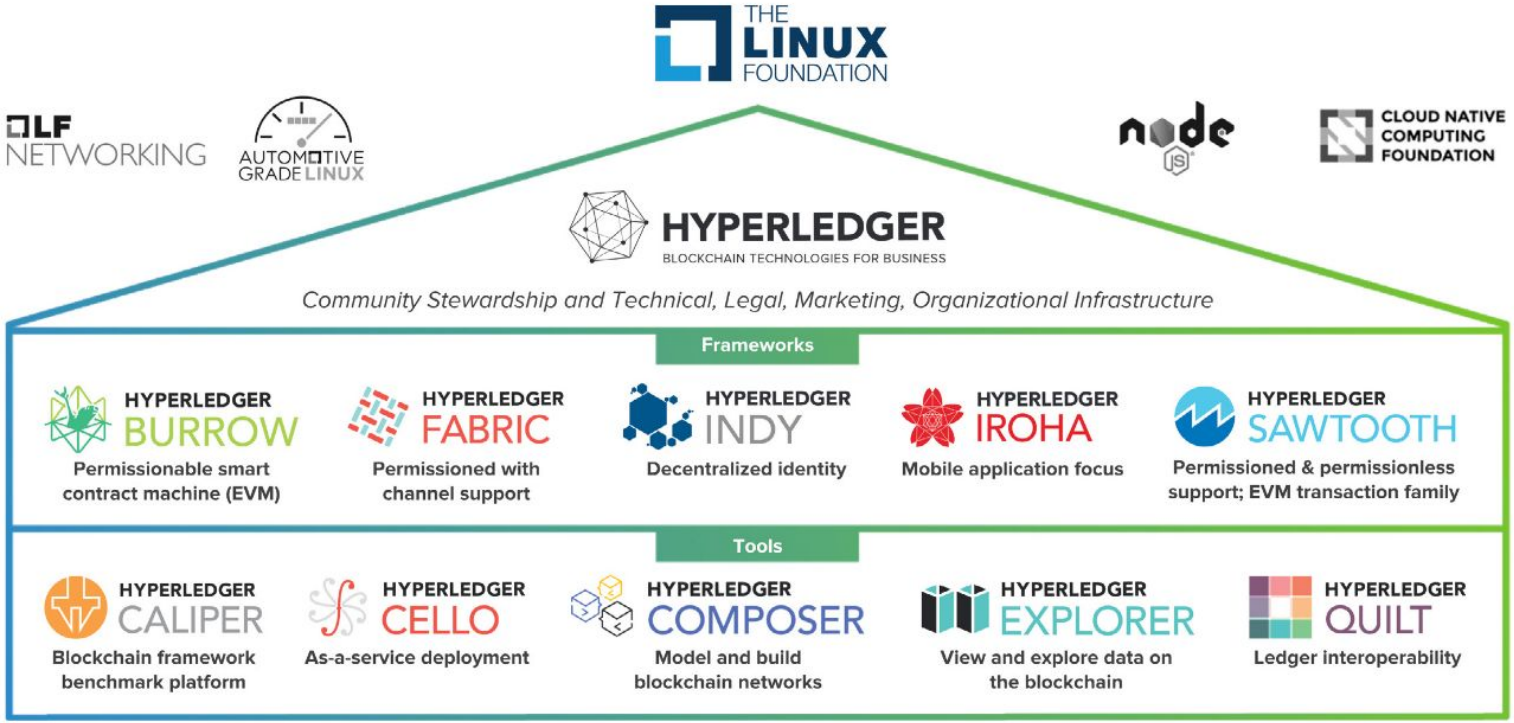
Como se ha resuelto hasta ahora

Como se ha resuelto hasta ahora

- Identidad Centralizada: Ejemplo de ello son los sistemas de identificación privados. Por ejemplo: Tu cuenta de Amazon o Netflix.
- Identidad Federada: misma identidad en múltiples sitios con el consentimiento del usuario. Ejemplos de ello son Bak, Bakq, o Giltza para interactuar con las administraciones vascas. En internet encontramos Google, Github, Facebook, Twitter, etc.
- Identidad Centrada en el Usuario: es el usuario quien tiene control sobre su identidad digital repartida entre diversas autoridades
-

Como se ha resuelto hasta ahora




THE LINUX FOUNDATION
LF NETWORKING
AUTOMOTIVE GRADE LINUX
node JS
CLOUD NATIVE COMPUTING FOUNDATION
HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS
Community Stewardship and Technical, Legal, Marketing, Organizational Infrastructure
Frameworks
HYPERLEDGER BURROW
Permissionable smart contract machine (EVM)
HYPERLEDGER FABRIC
Permissioned with channel support
HYPERLEDGER INDY
Decentralized identity
HYPERLEDGER IROHA
Mobile application focus
HYPERLEDGER SAWTOOTH
Permissioned & permissionless support; EVM transaction family
Tools
HYPERLEDGER CALIPER
Blockchain framework benchmark platform
HYPERLEDGER CELLO
As-a-service deployment
HYPERLEDGER COMPOSER
Model and build blockchain networks
HYPERLEDGER EXPLORER
View and explore data on the blockchain
HYPERLEDGER QUILT
Ledger interoperability

Hyperledger Indy

HYPERLEDGER
INDY

Indy is an Abbreviation for Independent Identity

Hyperledger Indy

Contributed by the Sovrin Foundation, Hyperledger Indy enables individuals to manage and control their digital identities. Instead of companies storing vast amounts of personal data, what they store are identity indicators.

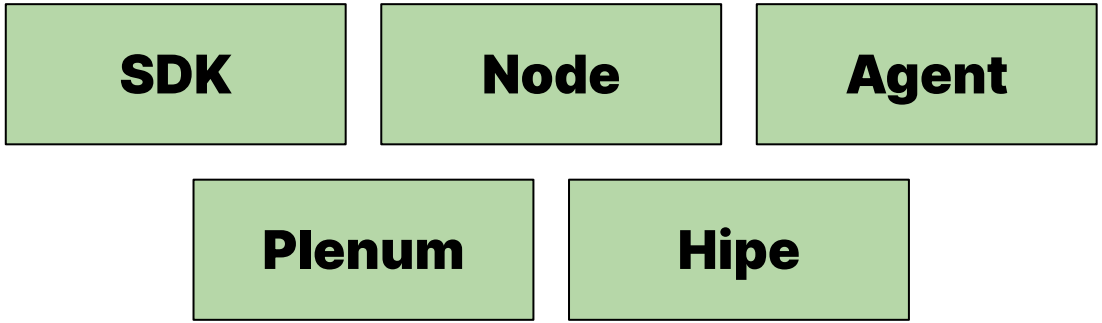
One of the key principles of Hyperledger Indy is its approach to privacy by design: it's not about protecting data, but designing so that data doesn't need protection.

Key Features

- Distributed ledger purpose-built for decentralized identity
- Correlation-resistant by design
- DIDs (Decentralized Identifiers) that are globally unique and resolvable (via a ledger) without requiring any centralized resolution authority
- Pairwise Identifiers create secure, 1:1 relationships between any two entities
- Verifiable Claims are interoperable format for exchange of digital identity attributes and relationships currently in the standardization pipeline at the W3C
- Zero Knowledge Proofs which prove that some or all of the data in a set of Claims is true without revealing any additional information, including the identity of the Prover

Components

Hyperledger Indy Global Component View



Architecture components

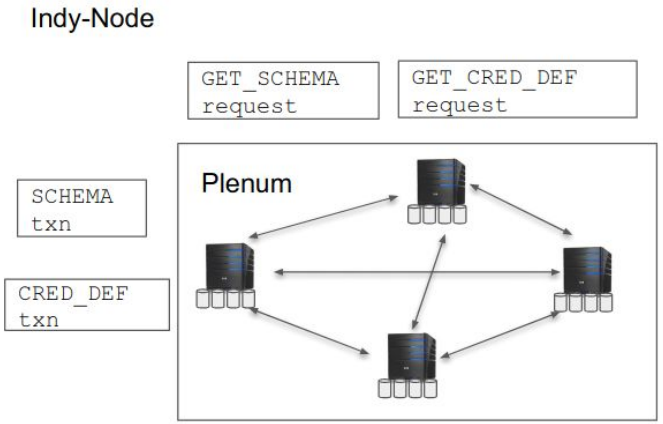
Hyperledger Indy Node - Indy Plenum

Indy-Plenum:

- https://github.com/hyperledger/indy-plenum
- Consensus Protocol
- Ledger

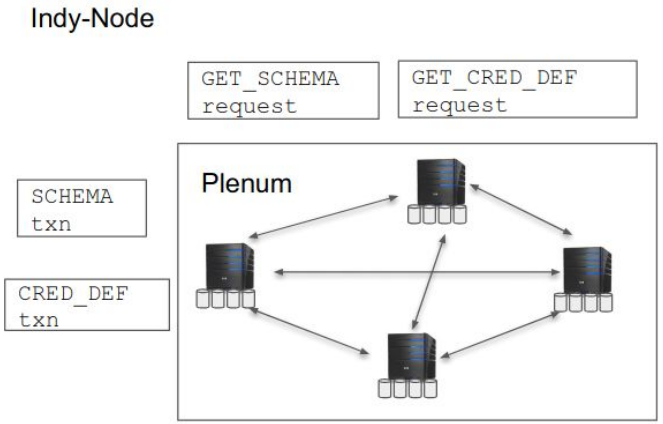
Indy-Node:

- https://github.com/hyperledger/indy-node
- Depends on indy-plenum
- Identity-specific transactions



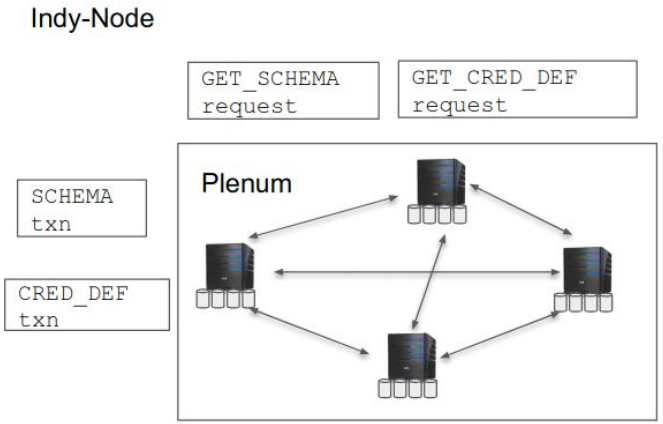
Hyperledger Indy Node - Indy Plenum

- Indy is a Ledger purpose-build for Identity
  - Can be used as a general-purpose Ledger
  - Extend Plenum
  - Custom transactions (pluggable request handlers)
  - Plugins



Hyperledger Indy Node & Indy Plenum

- Written in Python
- Depends on
  - ZMQ
  - Indy-crypto (Ursa)
  - Libsodium
- Message-driven and modular architecture
- Extensive test coverage
  - TDD
  - Unit tests
  - Integration tests
  - Property-based and simulation tests
  - System tests
  - Load tests (usually 25 Nodes)



Hyperledger Indy Peers Roles

- **Validator**
  - Handles writes and reads
  - Nodes involved in the consensus
- **Observer**
  - Handles read operations
  - Keep the state synced with **validator** nodes

Indy transactions & Consensus

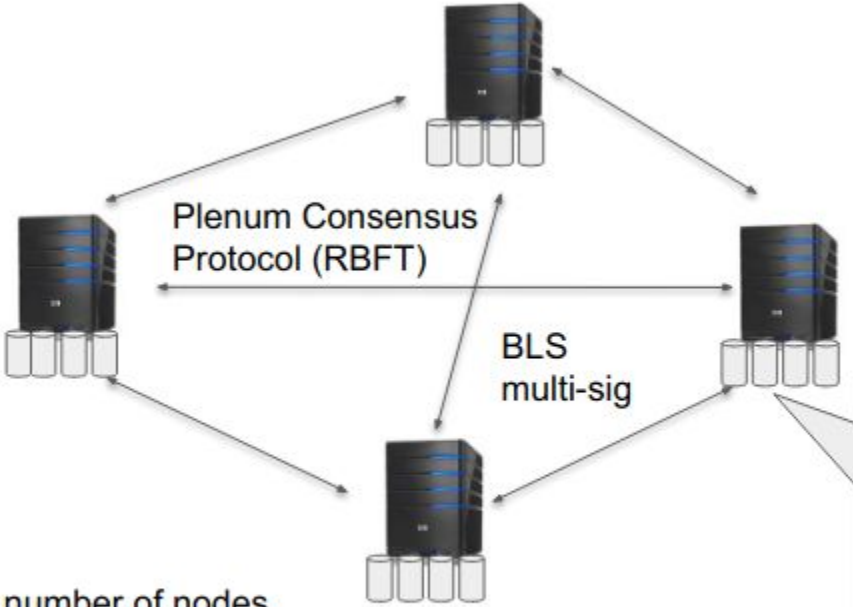
Indy Consensus

CONSENSUS N = 3F + 1

minimum network size 4 validators

N = total number of nodes
F = number of malicious nodes

Indy Consensus



ZMQ as secure transport

- TCP-based
- CurveCP, libsodium
- Authenticated encryption, no digital signatures
  - Authentication: Poly1305 MAC
  - Symmetric key crypto: XSalsa20
  - Public Key crypto: Curve25519

- Each Node replicates all ledgers
- Each Ledger has a Merkle Tree
- Most of the Ledgers have State based on Patricia Merkle Trie

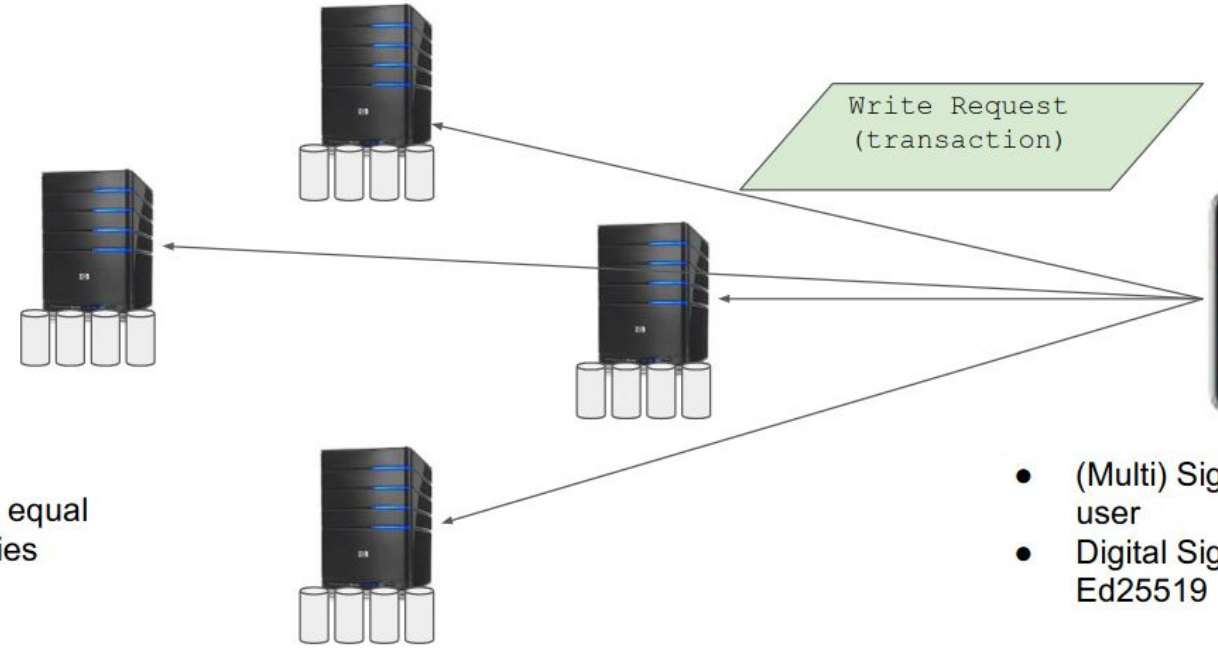
N=3F+1

- N - number of nodes
- F - max number of malicious nodes

Indy Write vs Read transactions

Indy WRITE transaction

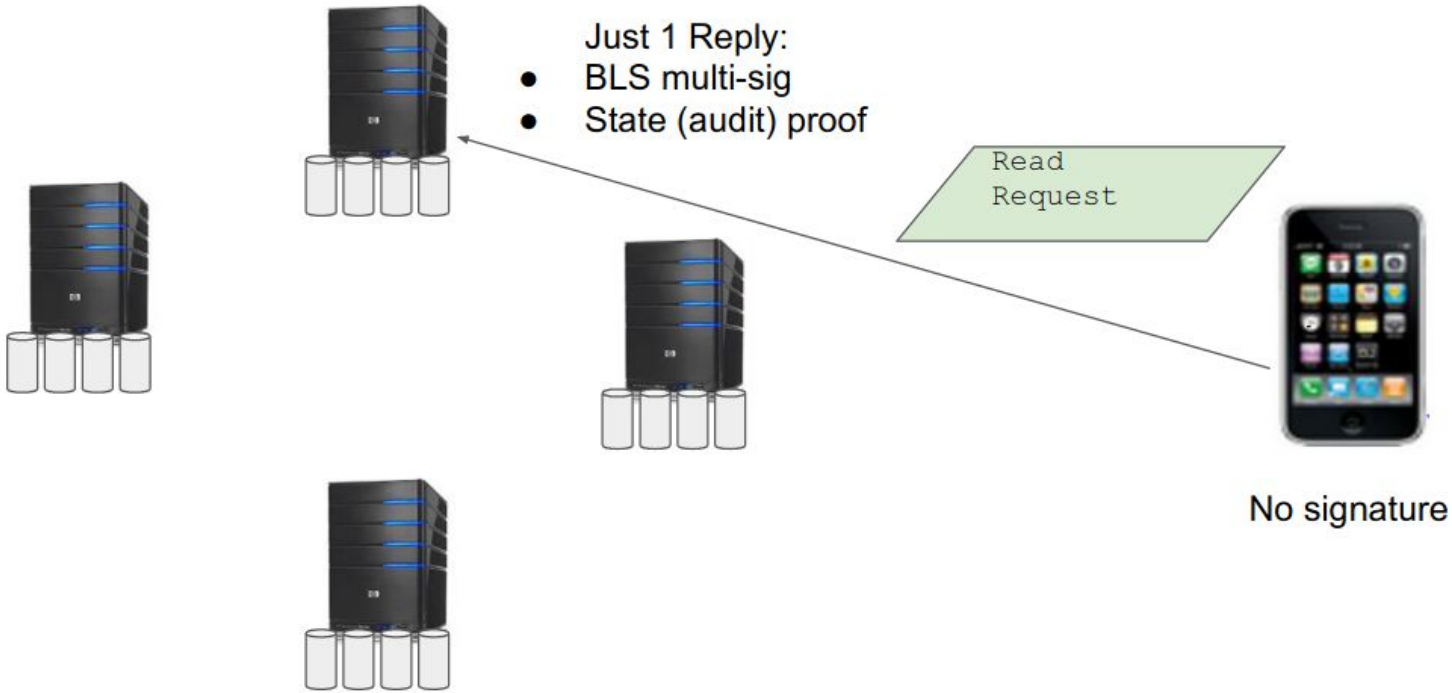
Indy Write vs Read transactions



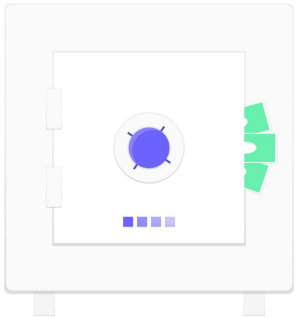
Indy Write vs Read transactions

Indy READ transactions

Indy Write vs Read transactions



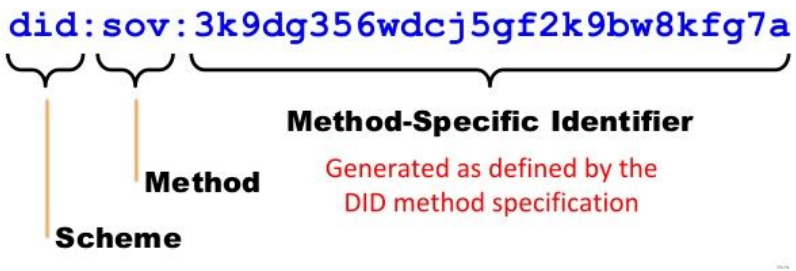
Indy Write vs Read transactions



Indy READ tx: any

Indy WRITE tx: auth required

Authentication is based on Ledger stored data and DIDs



Authenticating with a DID

Authentication is based on the information present in the Ledger.

- Write Requests:
  - Must be signed (Ed25519 digital signature)
  - Signature is verified against a Public Key stored on the Ledger (DID txn)
  - Every transaction author must have a DID transaction on the Domain Ledger.
- Read Requests:
  - Anyone can read, no authentication is required.

What data is stored in Blockchain

- No private data is written to the Blockchain
- Only Public data (such as Issuer's Public Key) is there

What data is stored in Blockchain

- **Identity Records** that describe a **Ledger Entity**
- Identity Records are public data and may include:
  - DIDs ↔ Public Keys
  - Service Endpoints
  - Credential Schemas
  - Credential Definitions
- **Identity Record** is associated with exactly one **DID.**
- To maintain privacy each **Identity Owner** can own multiple DIDs.

Indy ledgers

Indy ledgers

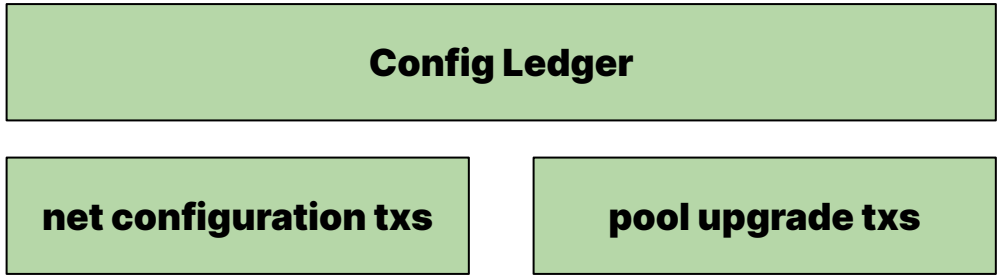
Config Ledger

Pool Ledger

Domain Ledger

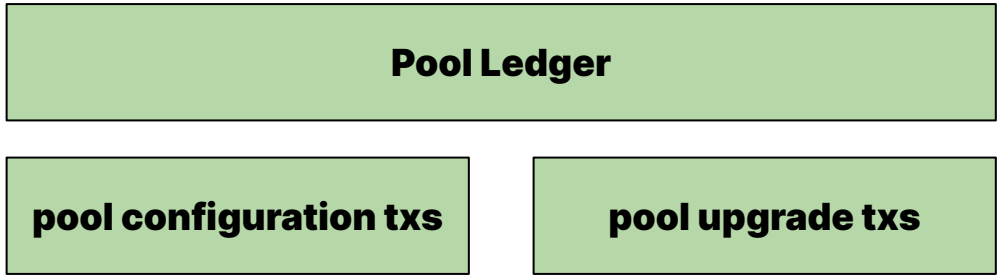
We can understand a ledger as a collection of data or as a partition.

Indy ledgers



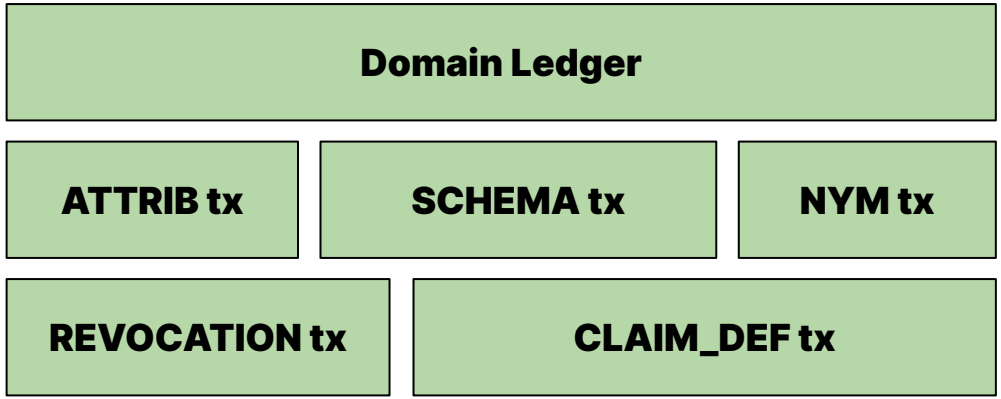
Store transactions related to network configuration or pool upgrade configuration.

Indy ledgers



Store transactions related to network pool configuration. For example: a new node to the pool, update an existing node in the pool.

Indy ledgers



Store transactions related to applications data plus NYM and DID related transactions.

In-depth view of DIDs

2

Purposes of DID

Verinym & Pseudonym

Purposes of DID

- **Verinym**:
  - associated with the **Legal Identity**
  - **For example, all parties should be able to verify that some DID is used by a Government to publish schemas for some document type.**
- **Pseudonym**:
  - a **Blinded Identifier** used to maintain privacy in the context of an ongoing digital relationship
  - If the Pseudonym is used to maintain only one digital relationship we will call it a **Pairwise-Unique Identifier**

Register a DID

- The creation of a **DID** known to the Ledger is an **Identity Record** itself (NYM transaction)
- The NYM transaction can be used for:
  - the creation of new DIDs
  - the setting
  - the rotation of a verification key
  - the setting and changing of roles

The most important fields of **NYM** this transaction are **dest** (target DID), **role** (role of a user NYM record being created for) and the **verkey** (target verification key).

Why register a DID?

Publishing with a DID verification key allows a person, organization or thing, to verify that someone owns this DID as that person, organization or thing is the only one who knows the corresponding signing key and any DID-related operations requiring signing with this key.

TL;DR: if you can sign a message with a verification key, your identity is verified.

Who can register DID?

Anyone who wants to publish DIDs needs to get the role of **Trust Anchor** on the ledger.

A **Trust Anchor** is a person or organization that the ledger already knows about, that is able to help bootstrap others.

(It is *not* the same as what cybersecurity experts call a "trusted third party"; think of it more like a facilitator).

How to become a Trust Anchor?

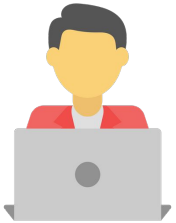
Becoming a **Trust Anchor** requires contacting a person or organization who already has the **Trust Anchor** role on the ledger.

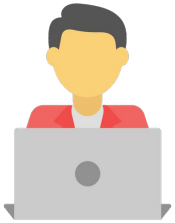
If no Trust Anchors exists registered in the ledger, the caller needs to contact one existing Steward. **Stewards** are automatically **Trust Anchors**.

How to become a Trust Anchor?

Once become a Trust Anchor, you can start registering DIDs

How to become a Trust Anchor? Onboarding first





Each connection is actually a pair of Pairwise-Unique Identifiers (DIDs). The one DID is owned by one party to the connection and the second by another.

Both parties know both DIDs and understand what connection this pair describes.

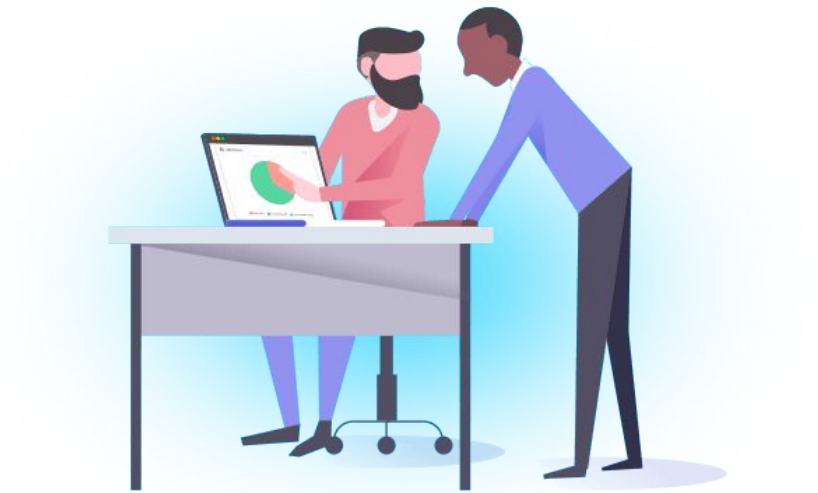
The relationship between them is not shareable with others; it is unique to those two parties in that each pairwise relationship uses different DIDs.

How to register a DID not being a Trust Anchor?

We have seen only Trust Anchors (and Stewards) can register DIDs. However if you:

- Don't want to become a Trust Anchor
- You are not authorized to become a Trust Anchor

You won't be able to register DID's. In this scenario, you have to delegate DID registration on existing Trust Anchor.



How to register a DID not being a Trust Anchor?

create DID
send DID to Anchor

read DID
register DID

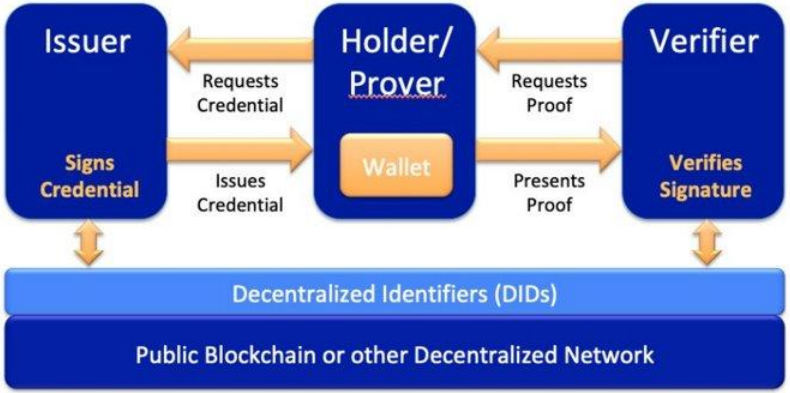
Create your first DID

Now, it's time to create your first DID. To do so, you are required to:

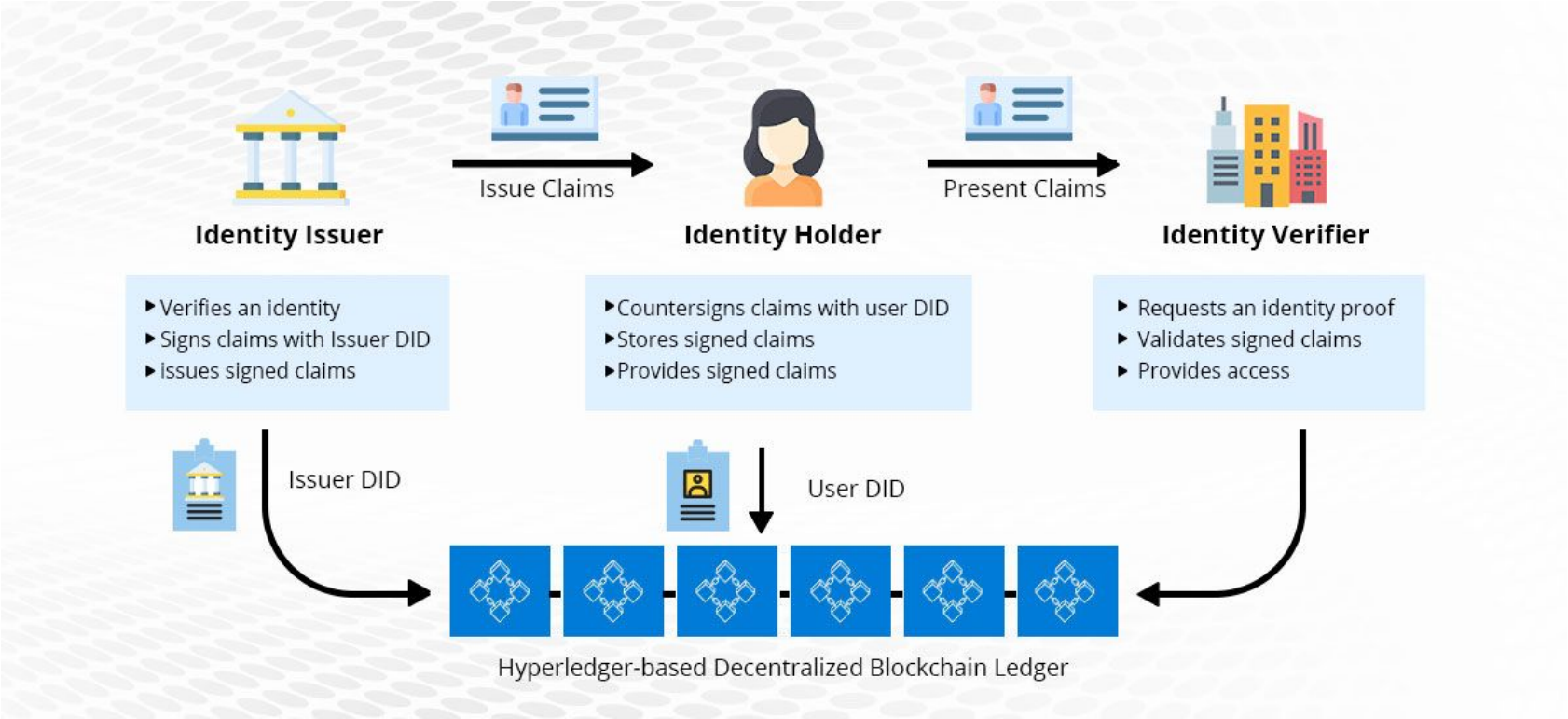
1. Choose your prefered language for indy programming.
2. Install Indy SDK.
3. Create a wallet.
4. Create your first DID.
5. Print on the screen DID and VERKEY

Identity Lifecycle

Typical Verifiable Credential Lifecycle



Typical Verifiable Credential Lifecycle



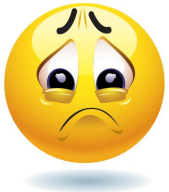
Sovrin Network

What is Sovrin Network

A production ready implementation of Hyperledger Indy working across the Globe with multiple companies

sovrin

Is Sovrin Network FREE?



Nope

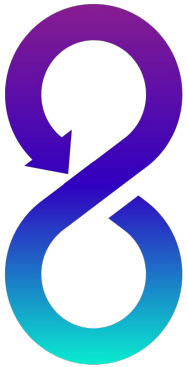
Sovrin Network Pricing

Item	Price
DID Write (for Credential Issuers)	$10
Schema	$50
Credential Definition	$25
Revocation Registry	$20
Revocation Update	$0.10
DIDs for Individuals (Peer DIDs)	Free
Credential Issuance	Free

https://sovrin.org/wp-content/uploads/2019/05/Public_Ledger_Fees_Writes_Definition_010519.pdf

How can I test and learn without paying?

Easy. You need to download all source code to your local machine and setup a local development/testing environment

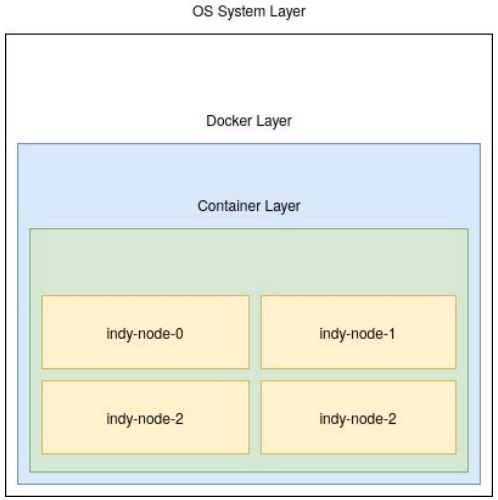


1. Download required Hyperledger Indy code from public sources such as Github, Dockerhub, etc.
2. Compile It
3. Run it
4. Learn it
5. Make some testing
6. Go to Step 2

Minimal

Development Environment

Development Environment Installation



1. Install Linux or VM: https://itsfoss.com/install-linux-in-virtualbox/
2. Install Docker. https://docs.docker.com/engine/install/ubuntu/
3. Follow docker post installation guide: https://docs.docker.com/engine/install/linux-postinstall/
4. To install indy-node testnet:
   1. open a terminal window
   2. mkdir indy-node
   3. cd indy-node
   4. wget https://raw.githubusercontent.com/hyperledger/indy-sdk/master/ci/indy-pool.dockerfile
   5. docker build -t indy/node -f indy-pool.dockerfile .
   6. docker run -it --rm -d -p 9701-9708:9701-9708 indy/node:latest

Development Environment Installation

```
wkm0108a@helios: ~
File  Edit  View  Search  Terminal  Help
wkm0108a@helios ~ docker images indy/node:latest
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
indy/node     latest    5412160c828c   4 days ago    700MB
wkm0108a@helios ~
```

At time of writing this guide, docker *indy-node* image size is **700 Mb.**

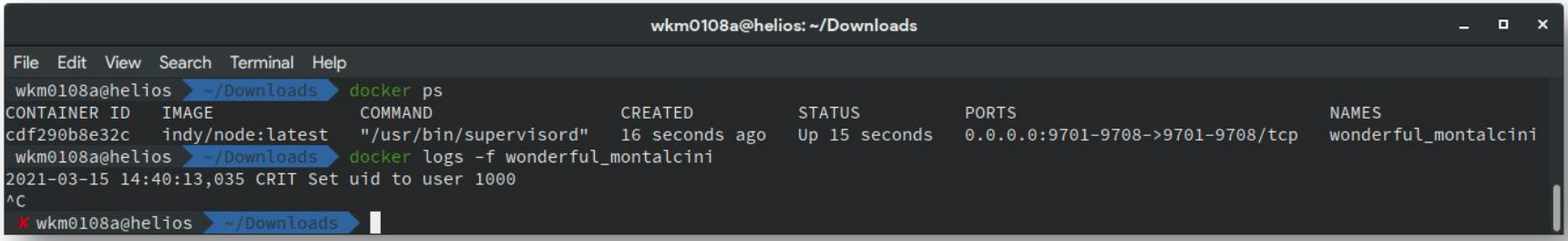
At Tecnalia, we shrink it to around 390 Mb.
Oh yeah 😎

```
wkm0108a@helios: ~/Downloads
File  Edit  View  Search  Terminal  Help
wkm0108a@helios ~/Downloads docker history indy/node:latest
IMAGE          CREATED       CREATED BY                                      SIZE      COMMENT
5412160c828c   4 days ago    /bin/sh -c #(nop)  CMD ["/usr/bin/supervisor…  0B
2f2261684728   4 days ago    /bin/sh -c #(nop)  EXPOSE 9701 9702 9703 970…  0B
ba9c887683cc   4 days ago    |11 indy_crypto_ver=0.4.5 indy_node_ver=1.12…  17.7kB
467ae94ce1b5   4 days ago    /bin/sh -c #(nop)  ARG pool_ip=127.0.0.1        0B
1364699ef463   4 days ago    |10 indy_crypto_ver=0.4.5 indy_node_ver=1.12…  546B
c34d4189759e   4 days ago    |10 indy_crypto_ver=0.4.5 indy_node_ver=1.12…  546B
e0f047fa4a5e   4 days ago    /bin/sh -c #(nop)  USER indy                    0B
727c21d55d6b   4 days ago    |10 indy_crypto_ver=0.4.5 indy_node_ver=1.12…  918B
efe5f8f4029d   4 days ago    |10 indy_crypto_ver=0.4.5 indy_node_ver=1.12…  169MB
21892bcd7852   4 days ago    /bin/sh -c #(nop)  ARG python3_pympler_ver=0…  0B
3332e5120aed   4 days ago    /bin/sh -c #(nop)  ARG python3_psutil_ver=5.…  0B
5e6dffaa146b   4 days ago    /bin/sh -c #(nop)  ARG python3_orderedset_ve…  0B
deb1df8e9541   4 days ago    /bin/sh -c #(nop)  ARG python3_pyzmq_ver=18.…  0B
cc99149ecb6b   4 days ago    /bin/sh -c #(nop)  ARG indy_crypto_ver=0.4.5    0B
76835174e3f8   4 days ago    /bin/sh -c #(nop)  ARG python3_indy_crypto_v…  0B
e89f115aa9a5   4 days ago    /bin/sh -c #(nop)  ARG indy_node_ver=1.12.1~…  0B
f6dd776d5b3f   4 days ago    /bin/sh -c #(nop)  ARG indy_plenum_ver=1.12.…  0B
b8bdbc1860c8   4 days ago    |2 indy_stream=master uid=1000 /bin/sh -c us…  335kB
718ef6936430   4 days ago    |2 indy_stream=master uid=1000 /bin/sh -c ec…  2.81kB
24cfedc6b92e   4 days ago    /bin/sh -c #(nop)  ARG indy_stream=master       0B
f5fa68ca211c   4 days ago    |1 uid=1000 /bin/sh -c apt-key adv --keyserv…  33.6kB
e40b59aac374   4 days ago    |1 uid=1000 /bin/sh -c pip3 install -U  pip=…  13.6MB
aec038f19807   4 days ago    |1 uid=1000 /bin/sh -c apt-get update -y && …  385MB
59ed6db472f0   4 weeks ago   /bin/sh -c #(nop)  ARG uid=1000                 0B
8185511cd5ad   7 weeks ago   /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      7 weeks ago   /bin/sh -c mkdir -p /run/systemd && echo 'do…  7B
<missing>      7 weeks ago   /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>      7 weeks ago   /bin/sh -c set -xe   && echo '#!/bin/sh' > /…  745B
<missing>      7 weeks ago   /bin/sh -c #(nop) ADD file:925571658dd8453e5…  132MB
wkm0108a@helios ~/Downloads
```

Development Environment Installation

Now, you can start your 'testnet' on local computer with:

```
docker run -it --rm -d -p 9701-9708:9701-9708 indy/node:latest
```



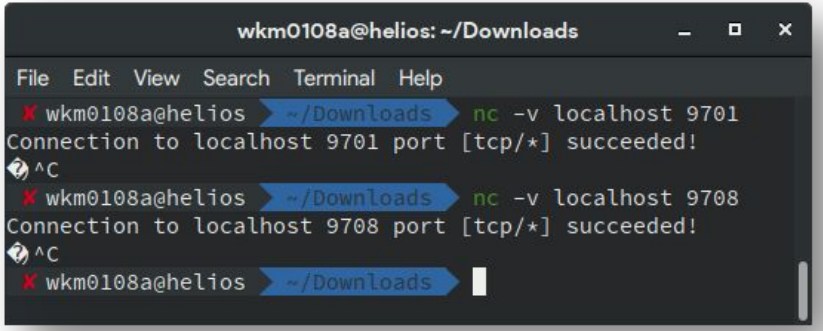
Development Environment Installation

Indy node services will be available at loopback address on specified ports.

localhost 9701 to localhost 9708

To verify indy pool ports are open and available

nc -v localhost 9701
nc -v localhost 9702
…
nc -v localhost 9708



Development Environment Possible Contributions

Create a Docker image for Ubuntu 18

Create a Docker image for Ubuntu 20

Create a Docker image for Alpine

Minimize docker image sizes

Create a Docker image for Ubuntu 19

Development Environment Review

Remember that this development environment is just that. It cannot be used as production environment because:

1. All nodes are running on same local machine.
2. All nodes know each other **crypto keys** (including privates).
3. Indy network bootstraping (genesis setup) is done using default testnet configuration.
4. It is not distributed at all. If machine goes down, network goes down.

Development Environment Review

If previous requirements, are fixed, then we could consider it as a stable and

more production-ready environment.

What's next?

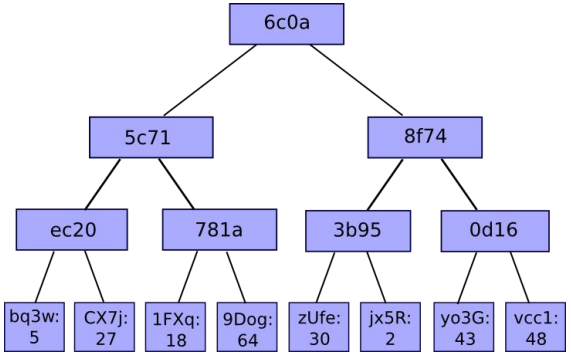
Now, having a working network setup we could:

- claim a Steward DID.
- become a Trust Anchor.
- Create a DID-Verkey (a pairwise identifier).
- Register the ownership of a DID in the ledger (NYM tx).
- Onboard a new user.
- Create and register an schema.
- Create and register a credential definition (or update it).
- Issue a non-revocable credential.
- Request a credential proof.
- Verify credential proof.
- Issue a revocable credential.

Indy Node

Indy Node (ledger peer)

- The software running on each network peer.
- All transactions are stored in a distributed ledger (replicated on all nodes)
- The ledger is based on a Merkle Tree
- The ledger consists of two things:
  - transactions log as a sequence of key-value pairs where key is a sequence number of the transaction and value is the serialized transaction
  - merkle tree (where hashes for leaves and nodes are persisted)
- Each transaction has a sequence number (no gaps) - keys in transactions log



Indy Node (ledger peer)

- So, this can be considered a blockchain where each block's size is equal to 1
- There are multiple ledgers by default:
  - *pool ledger*: transactions related to pool/network configuration (listing all nodes, their keys and addresses)
  - *config ledger*: transactions for pool configuration plus transactions related to pool upgrade
  - *domain ledger*: all main domain and application specific transactions (including NYM transactions for DID)
- All transactions are serialized to MsgPack format
- All transactions (both transaction log and merkle tree hash stores) are stored in a LevelDB
- One can use the read_ledger script to get transactions for a specified ledger in a readable format (JSON)

Genesis Block

- Indy is a public permissioned network, and thus, it may contain some initial data
- Each ledger may have a number of pre-defined transactions defining the initial pool and network.
  - pool genesis transactions define initial trusted nodes in the pool
  - domain genesis transactions define initial trusted trustees and stewards

Exercise: Create Genesis Content and Connect to Pool

```
cd mi-proyecto
wget https://raw.githubusercontent.com/hyperledger/indy-sdk/master/samples/nodejs/src/util.js
touch open-pool.js
```

- En este ejercicio conectaremos con la red de Indy que tenemos en local.
- Para ello crearemos un nuevo fichero llamado `open-pool.js` y descargamos el fichero `util.js` con código de ayuda
- Partiendo de los ejemplos, veremos cómo podemos abrir una conexión y conectarnos.
- Una vez creado el ejemplo, lo ejecutamos con el siguiente comando:

```
TEST_POOL_IP=127.0.0.1 node open-pool.js
```

Transaction Data Model

Field	Type	Description
ver	string	Transaction version to be able to evolve content. The content of all sub-fields may depend on this version.
txn	Object	Transaction-specific payload (data)
txnMetadata	Object	Metadata attached to the transaction.
reqSignature	Object	Submitter's signature over request with transaction (txn field).

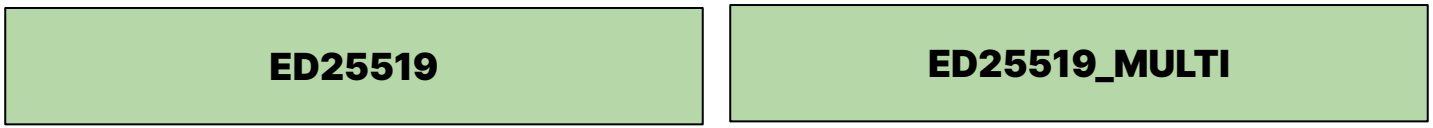
```
{
"ver": <...>,
"txn": {
    "type": <...>,
    "protocolVersion": <...>,
    "data": {
        "ver": <...>,
        <txn-specific fields>
    },
    "metadata": {
        "reqId": <...>,
        "from": <...>,
        "endorser": <...>,
        "digest": <...>,
        "payloadDigest": <...>,
        "taaAcceptance": {
            "taaDigest": <...>,
            "mechanism": <...>,
            "time": <...>
         }
    },
},
"txnMetadata": {
    "txnTime": <...>,
    "seqNo": <...>,
    "txnId": <...>
},
"reqSignature": {
    "type": <...>,
    "values": [{
        "from": <...>,
        "value": <...>
    }]
}}
```

Transaction Data Model (txnMetadata)

- **txnMetadata.txnTime**: The time when transaction was written to the Ledger as POSIX timestamp.
- **txnMetadata.seqNo**: A unique sequence number of the transaction on Ledger
- **txnMetadata.txnId (optional)**: Txn ID as State Trie key (address or descriptive data). It must be unique within the ledger. Usually present for Domain transactions only.

Transaction Data Model (reqSignature)

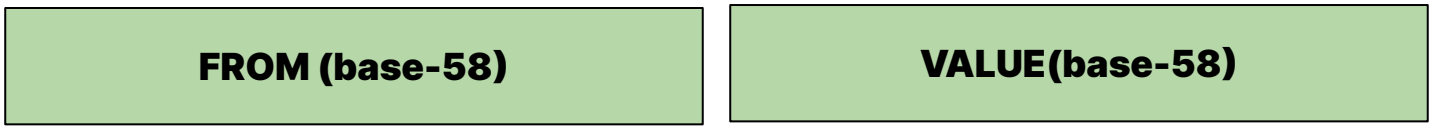
reqSignature.type: ed25519 signature | ed25519 signature in multisig case.



Ed25519 is an elliptic curve signing algorithm using EdDSA and Curve25519. The **Curve25519** is an elliptic curve offering 128 bits of security (256 bits key size) and designed for use with the elliptic curve Diffie–Hellman (ECDH) key agreement scheme. It is one of the fastest ECC curves and is not covered by any known patents.

Transaction Data Model (reqSignature)

reqSignature.values: list of items containing: {from, value}



- **From**: Identifier (DID) of signer as base58-encoded string for 16 or 32 byte DID value.
- **Value**: signature value encoded as base-58

```
"reqSignature": {
        "type": "ED25519",
        "values": [{
            "from": "L5AD5g65TDQr1PPHHRoiGf",
            "value": "4X3skpoEK2DRgZxQ9PwuEvCJpL8JHdQ8X4HDDFyztgqE15DM2ZnkvrAh9bQY16egVinZTzwHqznmnkaFM4jjyDgd"
        }]
    }
```

Transaction Types

14

transaction types

- NODE = 0
- NYM = 1
- TXN_AUTHOR_AGREEMENT = 4
- TXN_AUTHOR_AGREEMENT_AML = 5
- ATTRIB = 100
- SCHEMA = 101
- CLAIM_DEF = 102
- POOL_UPGRADE = 109
- NODE_UPGRADE = 110
- POOL_CONFIG = 111
- REVOC_REG_DEF = 113
- REVOC_REG_DEF = 114
- AUTH_RULE = 120
- AUTH_RULES = 122

What the hell is a NYM transaction

NYM transaction

- Creation of a DID that is known to the ledger is know as a **Verinym**, and the transaction used for creating a Verinym is known as a **NYM** transaction.
- NYM record is created by a for a specific user, Trust Anchor, Sovrin Stewards or trustee. Note that only trustees and Sovrin Stewards can create new Trust Anchors and a trustee can be created only by other trustees.
- The transaction can be used for creation of new DIDs, setting and Key Rotation of verification key, setting and changing of roles.

NYM transaction data

DEST	ROLE	ALIAS	VERKEY

- **Dest**: Target DID as base58-encoded string.
- **Role**: Role of a user that the NYM record is being created for. None (common USER), 0 (TRUSTEE), 2 (STEWARD), 101 (TRUST_ANCHOR)
- **Verkey**: Target verification key as base58-encoded string
- **Alias**: NYM's alias.

- If there is no NYM transaction for the specified DID yet, then this can be considered as the creation of a new DID.
- If there is already a NYM transaction with the specified DID, then this is is considered an update of that DID.
- If Key Rotation needs to be performed, the owner of the DID needs to send a NYM request with did and verkey only. role and alias will stay the same.

Indy actors

Steward	nodes that have permissions to participate in the transaction validation process and consensus
Trustee	a person or organization that the ledger already knows about, that is able to help bootstrap others.
Trust Anchor (Endorser)	a person or organization that writes transactions in the ledger on behalf other users.
Network monitor	a person or organization that the ledger already knows about, that is able to help bootstrap others.
Client	Applications that interact with indy-sdk or with the ledger.

Indy node monitor

Indy node monitor

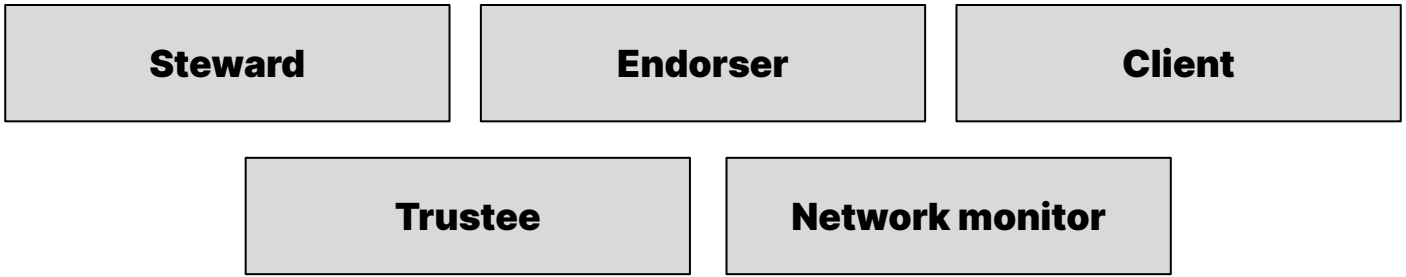
Indy Node Monitor is a set of tools for monitoring the status of an Indy Ledger by querying the validator information of the nodes of the ledger. Based on that, data can be generated data suitable for:

- visualization on a dashboard
- tracking trends about the status of nodes and the overall ledger
- tracking read and write uptimes
- tracking ledger usage such as number of transactions on the ledger
- driving notifications of node outages

Indy ACLs, roles and ownership

Indy Node ACL, Roles and authentication

Indy Node has support for following roles



And each them has a set of authorized actions

Indy Node Ownership

Operation type	Ownership pre-requirements
NYM transaction add	NA
NYM transaction edit	The DID defined by the NYM txn (`dest` field) if `verkey` is set; otherwise the submitter of the NYM txn (`identifier` field)

Indy Node Ownership (cont.)

Operation type	Ownership pre-requirements
Attribute add	The owner of the DID (`dest` field) the ATTRIB is created for (see NYM's owner description)
Attribute edit	The owner of the DID (`dest` field) the ATTRIB is created for (see NYM's owner description)

Indy Node Ownership (cont.)

Operation type	Ownership pre-requirements
SCHEMA add	NA
SCHEMA edit	The DID used to create the SCHEMA

Operation type	Ownership pre-requirements
CLAIM_DEF add	NA
CLAIM_DEF edit	The DID used to create the CLAIM_DEF

Indy Node Ownership (cont.)

Operation type	Ownership pre-requirements
REVOC_REG_ENTRY add	NA
REVOC_REG_ENTRY edit	The DID used to create the corresponding REVOC_REG_DEF

Operation type	Ownership pre-requirements
REVOC_REG_DEF add	NA
REVOC_REG_DEF edit	The DID used to create the REVOC_REG_DEF

Indy Node Ownership (cont.)

Indy also has some actions with no ownership, meaning that all users can execute them.

- POOL_UPGRADE
- POOL_RESTART
- POOL_CONFIG
- GET_VALIDATOR_INFO
- AUTH_RULE
- TRANSACTION_AUTHOR_AGREEMENT
- TRANSACTION_AUTHOR_AGREEMENT_AML

Exercise: Create and register an Schema

```
cd mi-proyecto
wget https://raw.githubusercontent.com/hyperledger/indy-sdk/master/samples/nodejs/src/util.js
touch open-pool.js
```

- En este ejercicio conectaremos con la red de Indy que tenemos en local.
- Para ello crearemos un nuevo fichero llamado `open-pool.js` y descargamos el fichero `util.js` con código de ayuda
- Partiendo de los ejemplos, veremos cómo podemos abrir una conexión y conectarnos.
- Una vez creado el ejemplo, lo ejecutamos con el siguiente comando:

```
TEST_POOL_IP=127.0.0.1 node open-pool.js
```

Indy actions

Indy Node actions

- An action is a command that is executed in the peer.
- A command is not registered as a transaction on the network
- Current version of indy, supports following command.
  - **`pool_restart`**: restart all nodes at the time specified in field “datetime”. **Trustee’s** are the only one who can launch this type of commands.

```
{
    "reqId": 2345474,
    "type": "118",
    "identifier": "M9BJDuS24bqbJNvBRsoGg3",
    "datetime": "2021-03-29T15:38:34.0+00:00",
    "action": "start"
}
```

Indy node recommendations

Indy Data Persistency Recommendations

Indy node requires FS integration!

1. Use system-specific files and folder for indy-node service, indy-config files, other config files, ledger data, genesis files, node keys and logs. Avoid using $HOME for them.
2. Organize folders to be compatible with multiple networks. (dev, test, local, prod, etc). The current Network to work is specified explicitly in the main config file with NETWORK_NAME parameter.
   a. Separate config files for each Network: `/var/lib/indy/{network_name}`
   b. Separate log files for each Network: `/var/log/indy/{network_name}`
3. Set proper permissions for files and folders using **chmod, chgrp and chown.**

Indy Data Persistency Recommendations

1. Clients should use $HOME folder or any userland location. By default indy uses $HOME/.indy_client (its a hidden folder)
2. Clients and nodes should be separated having independent files/folders
3. One indy client should provide service for several end users. To do so, each user must have separate data and key files with proper permissions.

Security Recommendations

It is strongly recommended to add iptables (or some other firewall) rule that limits the number of simultaneous clients connections for client port. There are at least two important reasons for this:

1. Preventing the indy-node process from reaching of open file descriptors limit caused by clients connections.
2. Preventing the indy-node process from large memory usage as ZeroMQ creates the separate queue for each TCP connection.

```
iptables -I INPUT -p tcp --syn --dport 9702 -m connlimit --connlimit-above
500 --connlimit-mask 0 -j REJECT --reject-with tcp-reset
```

Indy Genesis

Indy network genesis

The file containing network validator (stewards) node list. Required to connect.

{"reqSignature":{},"txn":{"data":{"data":{"alias":"ev1","client_ip":"54.207.36.81","client_port":"9702","node_ip":"18.231.96.215","node_port":"9701","services":["VALIDATOR"]},"dest":"GWgp6huggos5HrzHVDy5xeBkYHxPvrRZzjPNAyJAqpjA"},"metadata":{"from":"J4N1K1SEB8uY2muwmecY5q"},"type":"0"},"txnMetadata":{"seqNo":1,"txnId":"b0c82a3ade3497964cb8034be915da179459287823d92b5717e6d642784c50e6"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"zaValidator","client_ip":"154.0.164.39","client_port":"9702","node_ip":"154.0.164.39","node_port":"9701","services":["VALIDATOR"]},"dest":"BnubzSjE3dDVakR77yuJAuDdNajBdsh71ZtWePKhZTWe"},"metadata":{"from":"UoFyxT8BAqotbkhiehxHCn"},"type":"0"},"txnMetadata":{"seqNo":2,"txnId":"d5f775f65e44af60ff69cfbcf4f081cd31a218bf16a941d949339dadd55024d0"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"danube","client_ip":"128.130.204.35","client_port":"9722","node_ip":"128.130.204.35","node_port":"9721","services":["VALIDATOR"]},"dest":"476kwEjDj5rxH5ZcmTtgnWqDbAnYJAGGMgX7Sq183VED"},"metadata":{"from":"BrYDA5NubejDVHkCYBbpY5"},"type":"0"},"txnMetadata":{"seqNo":3,"txnId":"ebf340b317c044d970fcd0ca018d8903726fa70c8d8854752cd65e29d443686c"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"royal_sovrin","client_ip":"35.167.133.255","client_port":"9702","node_ip":"35.167.133.255","node_port":"9701","services":["VALIDATOR"]},"dest":"Et6M1U7zXQksf7QM6Y61TtmXF1JU23nsHCwcp1M9S8Ly"},"metadata":{"from":"4ohadAwtb2kfqvXynfmfbq"},"type":"0"},"txnMetadata":{"seqNo":4,"txnId":"24d391604c62e0e142ea51c6527481ae114722102e27f7878144d405d40df88d"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"digitalbazaar","client_ip":"34.226.105.29","client_port":"9701","node_ip":"34.226.105.29","node_port":"9700","services":["VALIDATOR"]},"dest":"D9oXgXC3b6ms3bXxrUu6KqR65TGhmC1eu7SUUanPoF71"},"metadata":{"from":"rckdVhnC5R5WvdtC83NQp"},"type":"0"},"txnMetadata":{"seqNo":5,"txnId":"56e1af48ef806615659304b1e5cf3ebf87050ad48e6310c5e8a8d9332ac5c0d8"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"OASFCU","client_ip":"38.70.17.248","client_port":"9702","node_ip":"38.70.17.248","node_port":"9701","services":["VALIDATOR"]},"dest":"8gM8NHpq2cE13rJYF33iDroEGiyU6wWLiU1jd2J4jSBz"},"metadata":{"from":"BFAeui85mkcuNeQQhZfqQY"},"type":"0"},"txnMetadata":{"seqNo":6,"txnId":"825aeaa33bc238449ec9bd58374b2b747a0b4859c5418da0ad201e928c3049ad"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"BIGAWSUSEAST1-001","client_ip":"34.224.255.108","client_port":"9796","node_ip":"34.224.255.108","node_port":"9769","services":["VALIDATOR"]},"dest":"HMJedzRbFkkuijvijASW2HZvQ93ooEVprxvNhqhCJUti"},"metadata":{"from":"L851TgZcjr6xqh4w6vYa34"},"type":"0"},"txnMetadata":{"seqNo":7,"txnId":"40fceb5fea4dbcadbd270be6d5752980e89692151baf77a6bb64c8ade42ac148"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"DustStorm","client_ip":"207.224.246.57","client_port":"9712","node_ip":"207.224.246.57","node_port":"9711","services":["VALIDATOR"]},"dest":"8gGDjbrn6wdq6CEjwoVStjQCEj3r7FCxKrA5d3qqXxjm"},"metadata":{"from":"FjuHvTjq76Pr9kdZiDadqq"},"type":"0"},"txnMetadata":{"seqNo":8,"txnId":"6d1ee3eb2057b8435333b23f271ab5c255a59819309045 2e9767f1edf1b4c72b"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"prosovitor","client_ip":"138.68.240.143","client_port":"9711","node_ip":"138.68.240.143","node_port":"9710","services":["VALIDATOR"]},"dest":"C8W35r9D2eubcrnAjyb4F3PC3vWQS1BHDg7UvDkvdV6Q"},"metadata":{"from":"Y1ENo59jsXYvTeP378hKWG"},"type":"0"},"txnMetadata":{"seqNo":9,"txnId":"15f22de8c95ef194f6448cfc03e93aeef199b9b1b7075c5ea13cfef71985bd83"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":{"alias":"iRespond","client_ip":"52.187.10.28","client_port":"9702","node_ip":"52.187.10.28","node_port":"9701","services":["VALIDATOR"]},"dest":"3SD8yyJsK7iKYdesQjwuYbBGCPSs1Y9kYJizdwp2Q1zp"},"metadata":{"from":"JdJi97RRDH7Bx7khr1znAq"},"type":"0"},"txnMetadata":{"seqNo":10,"txnId":"b65ce086b631ed75722a4e1f28fc9cf6119b8bc695bbb77b7bdff53cfe0fc2e2"},"ver":"1"}

https://sovrin-mainnet-browser.vonx.io/genesis

Indy network genesis

The content of a genesis node entry


```
{
    "reqSignature": {},
    "txn": {
        "data": {
                "data": {
                        "alias": "ev1",
                        "client_ip": "54.207.36.81",
                        "client_port": "9702",
                        "node_ip": "18.231.96.215",
                        "node_port": "9701",
                        "services": ["VALIDATOR"]
                },
                "dest": "GWgp6huggos5HrzHVDy5xeBkYHxPvrRZzjPNAyJAqpjA"
        },
        "metadata": {
                "from": "J4N1K1SEB8uY2muwmecY5q"
        },
        "type": "0"
    },
    "txnMetadata": {
        "seqNo": 1,
        "txnId": "b0c82a3ade3497964cb8034be915da179459287823d92b5717e6d642784c50e6"
    },
    "ver": "1"
}
```

How-to create a Tesnet

Creating a indy network

1. Install docker
2. Choose a fancy name for your INDY network
3. Start a local pool following the configuration of the Dockerfile
4. Open a connection against this pool as we created in `open-pool.js`
5. Now, you are ready to make transactions and interact with the ledger.

How-to add node to existing pool

Adding a Node to existing pool

1. You are required to have following files before joining. They should bi provided by one existing peer member.

```
/var/lib/indy/network_name/pool_transactions_genesis
/var/lib/indy/network_name/domain_transactions_genesis
```

2. Build New node initial data: alias, keys and define working IPs and ports.

```
init_indy_node $NODE_NAME 0.0.0.0 9701 0.0.0.0 9702 $NODE_SEED
```

   a. **`NODE_SEED:`** is an alpha-numeric 32 char seed that can be randomly generated or manually defined.

Adding a Node to existing pool

If you are running your new node without Docker, you will need to launch next commands to start the indy-node service

```
sudo systemctl start indy-node
sudo systemctl status indy-node
sudo systemctl enable indy-node
```

If you are using a docker image, you just need to start the container with

```
docker run -it -d $docker_image
```

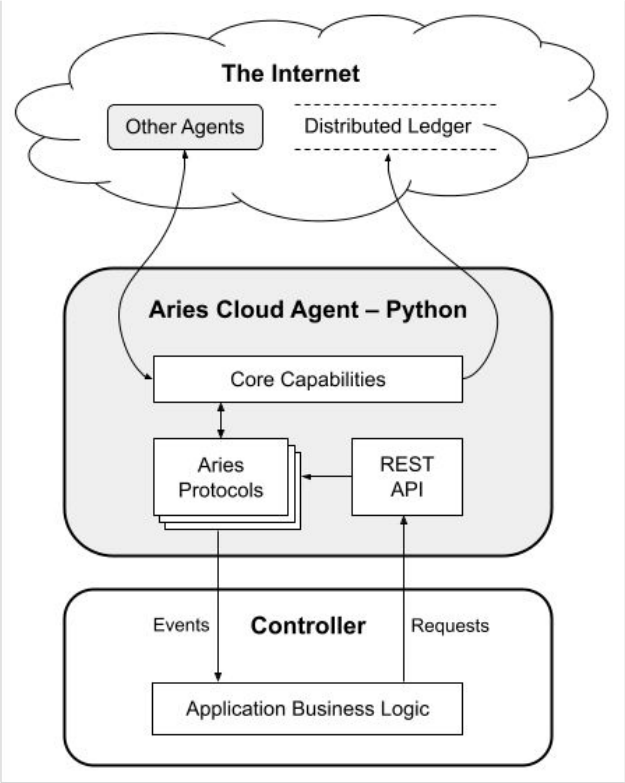
Adding a Node to existing pool

Once indy-node boot up, As Trustee add another Steward if needed (only Steward can add a new Validator Node.

A Steward can add one and only one Validator Node.

Indy Agents

Indy Agents


The Internet
Other Agents
Distributed Ledger
Aries Cloud Agent – Python
Core Capabilities
Aries Protocols
REST API
Events
Controller
Requests
Application Business Logic

Indy Wallet

Indy Wallet

The indy-sdk default wallet implementation uses hardened version of SQLCipher.

- HMAC-SHA256
- PBKDF2 100K rounds for passphrase key data.
- PBKDF2 10 rounds for HMAC key derivation.
- Page size 2K.

The wallet allows an optional passphrase to used for encrypting the data. If no passphrase is provided or is blank, it will not be encrypted but stored in SQLite3 format.

The passphrase to open the wallet is stored outside of indy-sdk and is left to the consumer's security preference such as HSMs, TEEs, or offline methods.

Indy Wallet creation

To create a wallet on the first time, the caller needs to execute the command of creation a new wallet on the SDK. If the wallet wants to be encrypted, a `key` needs to be specified.

```
{
    "key": "Th1sIsArEALLY$3cuR3PassW0RD"
}
```

Indy Wallet (change password)

To change a wallet key, a rekey parameter needs to be provided.

The wallet will be opened using key and change the passphrase to the rekey value for future open calls. rekey is only required for an existing wallet and throws an error when attempting to create a new wallet.

```
{
    "key": "Th1sIsArEALLY$3cuR3PassW0RD",
    "rekey": "s8c0R31tYi$hARd"
}
```

Indy Wallet encryption

To encrypt an non-encrypted wallet, a wallet open command needs to be executed indicating an the encryption key.

```
{
    "key": null,
    "rekey": Th1sIsArEALLY$3cuR3PassW0RD
}
```

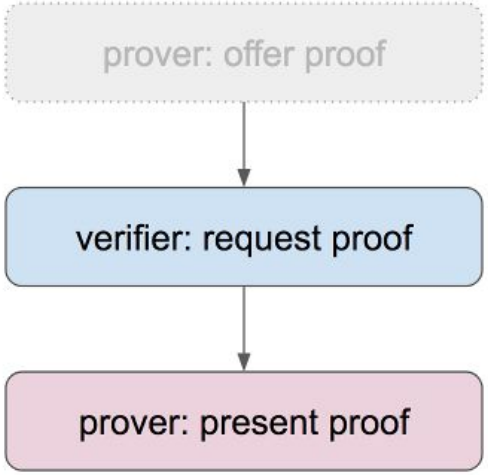
Indy Wallet decryption

To decrypt an encrypted wallet, a wallet open command needs to be executed indicating an empty rekey value.

```
{
    "key": "Th1sIsArEALLY$3cuR3PassW0RD",
    "rekey": null
}
```

Proof Negotiation

Proof negotiation

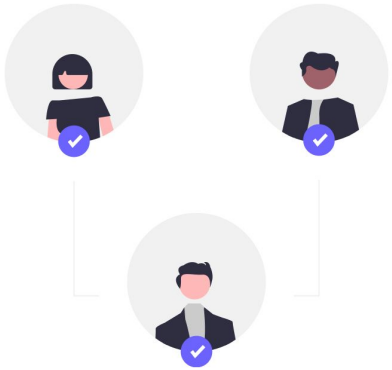


- Proof negotiation typically begins when a *verifier* (also called a *relying party*) requests proof. As with credential issuance, the process has three logical phases, but it is rare to begin with a proof offer.
- A proof request is a JSON file that describes what sort of proof would satisfy the relying party.
- Once the proof request is received, a holder of credentials must scan their *identity wallet* to find out which credentials could be used to satisfy the request.
- The holder becomes a *prover* by generating and presenting a proof. This is done by building some JSON that selects the credentials (out of those identified as valid candidates in the previous step), that the prover wishes to use to satisfy the request. The prover calls prover_create_proof function with appropriate parameters, and the proof is created.

additional info

Agent-to-agent communication

Agent to agent communication



- Agent to agent communication uses Elliptic-Curve Integrated Encryption Scheme (ECIES) to protect messages.
- A2A does not provide *forward-secrecy* and *key-compromise impersonation resistance*.
- Forward Secrecy HIPE is accepted: https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0024-a2a-forward-secrecy/README.html

Hyperledger URSA

Hyperledger Ursa

- Hyperledger Ursa is a shared cryptographic library
- It enables implementations to avoid duplicating other cryptographic work and hopefully increase security in the process.
- The library is an opt-in repository (for Hyperledger and non Hyperledger projects) to place and use crypto.
- Examples: cryptographic accumulators, shamir SS, BLS signatures, CL signatures, ZKP(bulletproof, rangeproof, membership), etc

hands-on

Hyperledger Indy interactions

Register a new Schema

```
async function sendSchema(poolHandle, walletHandle, Did, schema) {
     const schemaRequest = await indy.buildSchemaRequest(Did, schema);
     await indy.signAndSubmitRequest(poolHandle, walletHandle, Did, schemaRequest)
}

const [schemaId, schema] = await indy.issuerCreateSchema(
    issuerDID,
    'Job-Certificate',
    '0.2',
    ['first_name', 'last_name', 'salary', 'employee_status',
    'experience']
);

await sendSchema(poolHandle, issuerWallet, issuerDID, schema);
```

Register a new Schema Credential Definition

```
async function getSchema(poolHandle, did, schemaId) {
      let getSchemaRequest = await indy.buildGetSchemaRequest(did, schemaId);
      let getSchemaResponse = await indy.submitRequest(poolHandle, getSchemaRequest);
      return await indy.parseGetSchemaResponse(getSchemaResponse);
}

[, schema] = await getSchema(poolHandle, issuerDID, schemaId);

const [transcriptCredDefId, transcriptCredDefJson] = await indy.issuerCreateAndStoreCredentialDef(
    issuerWallet,
    issuerDID,
    schema,
    'TAG1',
    'CL',
    '{"support_revocation": false}'
);
```

Credential offer issuing

```
console.log("CREANDO LA OFERTA DEL CREDENCIAL PARA EL USUARIO")
const credOffer = await indy.issuerCreateCredentialOffer(
    issuerWallet,
    credDefId
);
```

Create master secret

```
console.log("HOLDER: CREANDO MASTER SECRET")
const masterSecretId = await indy.proverCreateMasterSecret(
    holder.wh,
    null
);
```

Create a response for an offer

```
console.log("HOLDER: ACEPTANDO LA OFERTA DEL CREDENTIAL Y GENERANDO RESPUESTA");
const [credReq, credReqMetadata] = await indy.proverCreateCredentialReq(
    holder.wh,
    holder.did,
    credOffer,
    defJson,
    masterSecretId);
console.log("HOLDER: RESPUESTA A LA OFERTA")
console.log(credReq);
console.log("HOLDER: RESPUESTA METADATA")
console.log(credReqMetadata);
```

Issuing final credential (non-revocable)

```
let credentialValues = {
    "first_name": { "raw": "Jhon", "encoded": encoder.encodeCredValue("Jhon") },
    "last_name": { "raw": "Doe", "encoded": encoder.encodeCredValue("Doe") },
    "salary": { "raw": "20000", "encoded": encoder.encodeCredValue(20000) },
    "employee_status": { "raw": "hired", "encoded": encoder.encodeCredValue("hired") },
    "experience": { "raw": "2", "encoded": encoder.encodeCredValue(2) },
};
console.log("ISSUER: CREANDO CREDENCIAL CON LOS DATOS")
console.log(JSON.stringify(credentialValues, null, 4));
let [credentialData] = await indy.issuerCreateCredential(
    issuerWallet,
    credOffer,
    credReq,
    credentialValues,
    null,
    -1
);
console.log("ISSUER: CREDENCIAL CREADO")
console.log(JSON.stringify(credentialData));
```

Store credential in wallet

```
console.log("HOLDER: GUARDANDO CREDENCIAL EN WALLET")
const id = await indy.proverStoreCredential(
    holder.wh,
    null,
    credReqMetadata,
    credentialData,
    defJson,
    null
);
```

Create a proof request

```
nonce = await indy.generateNonce();
let proofRequest = {
    'nonce': nonce,
    'name': 'Proof-Request',
    'version': '0.2',
    'requested_attributes': {
        'attr1_referent': {
            'name': 'first_name',
            //'restrictions': [{ 'cred_def_id': credDefId }]
        },
        'attr2_referent': {
            'name': 'last_name',
            //'restrictions': [{ 'cred_def_id': credDefId }]
        }
    },
    'requested_predicates': {
        /*'predicate1_referent': {
            'name': 'salary',
            'p_type': '>=',
            'p_value': 5000,
            restrictions': [{ 'cred_def_id': credDefId }]
        }*/
    }
};
```

Search for credentials

```
let searcher = await indy.proverSearchCredentialsForProofReq(
    wh,
    proofRequest,
    null
);
console.log("USER A: BUSCANDO CREDENCIALES EN LA WALLET: ", wh)
let credentials = await indy.proverFetchCredentialsForProofReq(searcher, 'attr1_referent', 10)
credForAttr1 = credentials[0]['cred_info'];

credentials = await indy.proverFetchCredentialsForProofReq(searcher, 'attr2_referent', 10)
credForAttr2 = credentials[0]['cred_info'];

credentials = await indy.proverFetchCredentialsForProofReq(searcher, 'predicate1_referent', 10)
let credForPredicate1 = credentials[0]['cred_info'];
// se cierra el acceso a la busqueda
await indy.proverCloseCredentialsSearchForProofReq(searcher);

console.log("USER A: CREDENCIALES COINCIDENTES ENCONTRADOS")
console.log(JSON.stringify(credentials, null, 4))

let credsForProof = {};
credsForProof[`${credForAttr1['referent']}`] = credForAttr1;
credsForProof[`${credForAttr2['referent']}`] = credForAttr2;
credsForProof[`${credForPredicate1['referent']}`] = credForPredicate1;
```

Build a proof

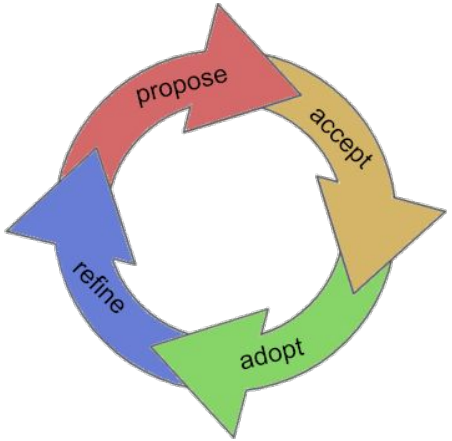
```
const finalProofJson = await indy.proverCreateProof(
    wh,
    proofRequest,
    requestedCredentials,
    masterSecretId,
    schemasJson,
    credDefsJson,
    revocStatesJson
);
console.log("USER A: PRUEBA GENERADA")
console.log(JSON.stringify(finalProofJson, null, 4))
```

Verify a proof

```
const isValid = await indy.verifierVerifyProof(
    proofRequest,
    proofData,
    schemasJson,
    credDefsJson,
    revocRefDefsJson,
    revocRegsJson
);
console.log("VERIFIER: PRUEBA VERIFICADA?");
console.log(isValid);
```

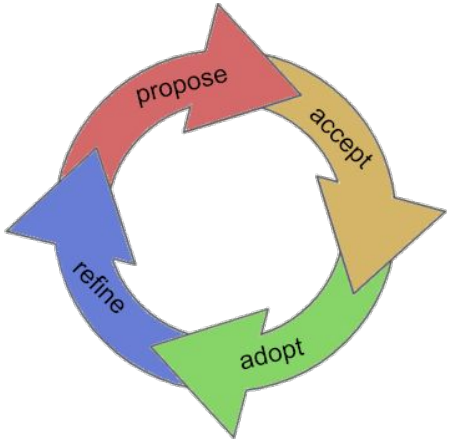
How-to contribute to Indy

Contributions



- https://github.com/hyperledger/indy-hipe
- PROPOSED: Proposed HIPEs are considered a "work in progress", even after they are merged. In other words, they haven't been endorsed by the community yet, but they seem like reasonable ideas worth exploring.
- ACCEPTED: To get a HIPE accepted, build consensus for your HIPE on chat and in community meetings. An accepted HIPE is incubating on a standards track; the community has decided to polish it and is exploring or pursuing implementation.

Contributions



- ADOPTED: To get a HIPE adopted, socialize and implement. A HIPE gets this status once it has significant momentum--when implementations accumulate, or when the mental model it advocates has begun to permeate our discourse. In other words, adoption is acknowledgment of a de facto standard.
- SUPERSEDED: Significant refinements require a superseding document; the original HIPE is superseded with a forwarding hyperlink, not replaced.

Exercises

Exercises

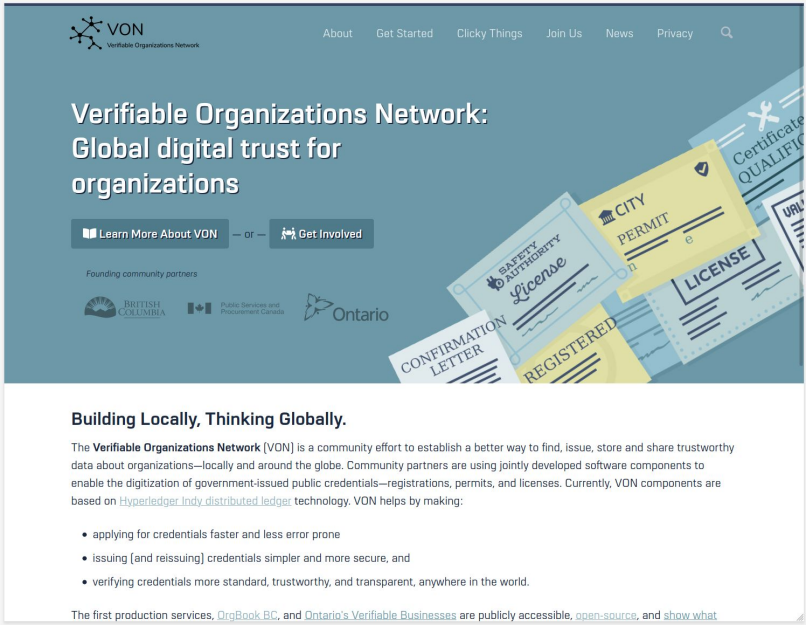
- Deploy a local pool.
- Connect to local pool.
- Create a DID.
- Create a schema.
- Register the schema.
- Create a credential definition.
- Register credential definition.
- Create a Credential Offer.
- As Issuer, create a non-revocable credential.
- As verifier, request a proof.
- As holder, create a proof.
- As verifier, verify a proof.
- Repeat, last 3 but with revocable credential.

Public & opensourced

Use Cases

Use cases

https://vonx.io



Public

References

References

- https://docs.google.com/document/d/1sXZoN18lpFoAF075QoptofwDV_10tUylPGFKRQnA56E/edit
- https://github.com/sovrin-foundation/sovrin/blob/master/TAA/TAA.md
- https://sovrin.org/wp-content/uploads/2019/05/Public_Ledger_Fees_Writes_Definition_010519.pdf
- https://selfserve.sovrin.org/
- https://sovrin.org/library/sovrin-governance-framework/
- https://sovrin.org/wp-content/uploads/Guardianship-Whitepaper2.pdf
- https://docs.google.com/document/d/1SswHBZ1pwuIUcePeFe8czOoAOaHE78ij40kXuQq5OW0/edit
- https://sovrin.org/test-sovrin-tokens/
- https://vonx.io/

this page intentionally left blank