Hyperledger Indy Documentation

Prerequisites

1. System Requirements:

1. **OS**: Ubuntu 18.04 or 20.04 (recommended).
2. **CPU**: 4 cores or higher.
3. **Memory**: Minimum 8GB RAM.
4. **Storage**: 100GB free disk space.

2. Dependencies:

1. Docker and Docker Compose.
2. Git for cloning repositories.
3. Python 3.6+.
4. Libsodium for cryptographic operations.

3. Networking:

1. Open required ports (e.g., 9701-9708) for validator node communication.
2. Stable internet connection for pulling dependencies.

4. Development Tools (optional for building apps):

1. Node.js and npm for front-end development.
2. Indy-SDK and Python bindings for agent development.

Setup Steps

1. Install Docker and Docker Compose

```
sudo apt update

sudo apt install -y docker.io docker-compose

sudo systemctl start docker

sudo systemctl enable docker
```

2. Clone the von-network Repository

Download the von-network project from GitHub, which is used to set up a local Indy ledger.

```
git clone https://github.com/bcgov/von-network.git
```

3. Navigate to the Project Directory

Move into the von-network directory.

```
cd von-network
```

4. Build the von-network

Build the Docker images for the network.

```
./manage build
```

5. Start the von-network

Start the Indy ledger network locally using Docker.

```
./manage start
```

6. Get the Internal Docker Host IP

Retrieve the internal Docker host IP address for accessing the local Indy network.

```
./manage dockerhost
```

7. Initialize a Pool Configuration in Indy-cli

Create a pool configuration in the Indy CLI and connect it to the genesis file served by von-network.

```
./manage cli init-pool My_Pool http://<DockerHostIP>:9000/genesis
```

8. Open Indy-Cli

```
./manage indy-cli
```

9. Connect to the Pool

Open the pool to start interacting with the Indy ledger.

```
pool connect My_Pool
```

10. Create a Wallet

```
wallet create pool_wallet key=poolkey
```

11. Open a Wallet

```
wallet open pool_wallet key=poolkey
```

12. Generate a DID

```
did new
```

Hyperledger Aries Cloud Agent (ACA-Py)

Prerequisites

Ensure you have the following tools installed:

1. Python 3.12 or later
2. Pip (Python package manager)
3. Docker and Docker Compose
4. aries-askar, aries-cloudagent, indy-vdr, indy-credx

Setup Steps

1. Clone the Aries Cloud Agent Python (ACA-Py) repository

```
git clone https://github.com/openwallet-foundation/acapy
```

2. Navigate into the cloned directory

```
cd acapy
```

3. Install Python 3.12

```
sudo add-apt-repository ppa:deadsnakes/ppa

sudo apt update

sudo apt install python3.12
```

4. Create a Virtual Environment

```
sudo apt install python3.12-venv

python3.12 -m venv venv
```

5. Activate the Virtual Environment

```
source venv/bin/activate
```

6. Install Required Packages

```
pip install .

pip install \

aries-cloudagent \

aries-askar \

indy-vdr \

indy-credx \

anoncreds

pip install python3_indy # Use if you encounter a missing module indy
```

7. Start ACA-Py Agent

```
aca-py start \

--inbound-transport http 0.0.0.0 8000 \

--outbound-transport http \

--no-ledger \

--admin-insecure-mode
```

Using Indy with Aries Agent

1. Start Indy Ledger Network Locally

```
cd von-network
```

```
./manage build
```

```
./manage start
```

2. Get the Internal Docker Host IP

```
./manage dockerhost
```

3. Register Aries Agent on Indy Ledger

Visit the following URL:

http://localhost:9000/

Register using a seed:

1. **Seed Name**: Enter your desired agent name.
2. **Seed**: `issuer00000000000000000000000000` *(Copy this seed; it's needed to connect the Aries agent with the ledger.)*

4. Start PostgreSQL Database to Store Wallet

Navigate to the ACA-Py directory:

```
cd scripts
```

```
./run_postgres
```

5. Access PostgreSQL Container

View real-time log entries when agents connect.

```
docker ps
```

Identify the PostgreSQL container and execute the following:

```
docker exec -it <container_id> bash

cd /var/lib/postgresql/data/log

tail -f <postgres-x-y-z>.log
```

Use the following credentials for PostgreSQL:

1. **Username**: postgres
2. **Password**: mysecretpassword

6. Start Aries Agent

Navigate to the ACA-Py scripts directory and run the following:

```
PORTS="8020 8021" \

./run_docker start \

-l Holder \

-it http 0.0.0.0 8020 \

-ot http \

--admin 0.0.0.0 8021 \

--admin-insecure-mode \
-e http://<DockerIP>:8020 \

--genesis-url http://<DockerIP>:9000/genesis \

--wallet-type askar \

--wallet-name verifier_wallet \

--wallet-key verifier_wallet \

--log-level info \

--auto-provision \

--auto-ping-connection \

--seed Verifier000000000000000000000000 \

--auto-accept-invites \

--auto-accept-requests \

--auto-respond-credential-proposal \

--auto-respond-credential-offer \

--auto-respond-credential-request \

--auto-store-credential \

--wallet-storage-type postgres_storage \

--wallet-storage-config "{\"url\":\"172.18.7.14:5432\",\"wallet_scheme\":\"DatabasePerWallet\"}" \

--wallet-storage-creds
"{\"account\":\"postgres\",\"password\":\"mysecretpassword\",\"admin_account\":\"postgres\",\"admin_password\":\"mysecretpassword\"}"
```

Change Ports and Seeds for Other Agents: Repeat the above process with different port labels and seeds for additional agents.

Parameter Details:

- **PORTS="8020 8021":** Ports used for HTTP communication (8020 for agent, 8021 for admin).
- **./run_docker start:** Command to start the agent using Docker.
- **-l Holder:** Role of the agent (e.g., Holder, Issuer, or Verifier).
- **-it http 0.0.0.0 8020:** Inbound transport type (http), bind address (0.0.0.0), and port (8020).
- **-ot http:** Outbound transport type (http).
- **--admin 0.0.0.0 8021:** Admin API bind address (0.0.0.0) and port (8021).
- **--admin-insecure-mode:** Enables admin API access without authentication.
- **-e http://172.18.7.14:8020:** Public endpoint for agent communication.
- **--genesis-url http://172.18.7.14:9000/genesis:** URL for the ledger genesis transaction file.
- **--wallet-type askar:** Specifies the wallet type (askar).
- **--wallet-name verifier_wallet:** Name of the wallet being used.
- **--wallet-key verifier_wallet:** Encryption key/password for the wallet.
- **--log-level info:** Log verbosity level (e.g., info, debug, etc.).
- **--auto-provision:** Automatically provisions the wallet and configuration on startup.
- **--auto-ping-connection:** Enables automatic connection pings.
- **--seed Verifier000000000000000000000000:** Seed used to generate a deterministic DID.
- **--auto-accept-invites:** Automatically accepts connection invitations.
- **--auto-accept-requests:** Automatically accepts connection requests.
- **--auto-respond-credential-proposal:** Automatically responds to credential proposals.
- **--auto-respond-credential-offer:** Automatically responds to credential offers.
- **--auto-respond-credential-request:** Automatically responds to credential requests.

IMPLEMENTATION OF HYPERLEDGER INDY

The Virat/IIT Bombay/Accenture Demo (Verifiable Credentials)

This demo shows how Virat, a student from IIT Bombay, proves his qualification to Accenture using a special digital proof called verifiable credentials. Here's how it works:

Step-by-Step Flow:

- Virat connects with IIT Bombay
  Virat, a former student of IIT Bombay, gets a special digital certificate (credential) from the college. This certificate proves that he graduated from IIT Bombay and has earned a degree.

- Virat shares his credential
  Virat is applying for a job at Accenture. To prove his qualification, he needs to show that he graduated from IIT Bombay. Instead of sending a physical document or a regular PDF, he shares the digital certificate directly from his phone or computer.

- Accenture asks for proof
  Accenture, the company Virat is applying to, asks Virat for proof of his degree. The company doesn't need to verify the document manually. Instead, they ask Virat to show them a verifiable credential.

- Virat shares the digital proof
  Virat provides the digital certificate to Accenture using a special technology. Accenture can instantly verify that the certificate is real and comes from IIT Bombay. This happens without the need for physical documents, making the process much faster and safer.

- Accenture confirms the credential
  Accenture checks the certificate, and since it's a verifiable credential, they can trust that Virat's degree is valid. The company doesn't need to contact IIT Bombay to confirm the degree; the credential itself provides that assurance.

Flow from Accenture Inviting Virat to Verification:

Step 1: Virat Receives Invitation from Accenture

- **Accenture** sends an **invitation** to **Virat** requesting proof of his **degree from IIT Bombay** as part of the hiring process. The request may specify that he needs to provide a **verifiable credential** (digital degree certificate).
- This invitation is typically sent through **email** or a **secure platform** and includes details about how Virat can share his credential securely.

Step 2: Virat Requests Credential from IIT Bombay

- **Virat** logs into his **digital wallet** (or the platform used to manage verifiable credentials).
- He initiates a **Verifiable Credential Request** to **IIT Bombay** by submitting his personal information, like his **name**, **student ID**, and the specific credential (degree) he needs.
- The **request is digitally signed** by Virat's wallet to authenticate the source.
- The request is securely transmitted to **IIT Bombay's Credential Issuance System** via a protocol like **HTTPS** or **DIDComm**.

Step 3: IIT Bombay Validates and Issues the Credential

- **IIT Bombay's Credential Issuance System** receives the request.
- The system validates Virat's identity and checks his records against the university's database to confirm that he is eligible to receive the credential.
- If everything is valid, **IIT Bombay generates a Verifiable Credential** that contains:
    - **Degree details** (e.g., degree type, student ID, graduation date).
    - **Digital signature** of IIT Bombay.
    - **Decentralized Identifier (DID)** of IIT Bombay, confirming the authenticity of the credential.
- The **verifiable credential** is formatted as a **signed JSON-LD object**.
- IIT Bombay securely sends the **Verifiable Credential** to Virat's **digital wallet** via the same secure communication protocol (e.g., **HTTPS** or **DIDComm**).

Step 4: Virat Receives the Credential

- **Virat's digital wallet** receives the issued **Verifiable Credential** (degree certificate).
- Virat can now review and manage this credential in his digital wallet. The credential is stored securely and is ready to be shared.

Step 5: Virat Shares the Credential with Accenture

- Virat logs into his **digital wallet**, selects the **degree credential** issued by **IIT Bombay**, and securely shares it with **Accenture**.
- The credential is sent via a **secure platform** or **API** that ensures the integrity and privacy of the information. This could involve using a **QR code** or a **DIDComm** protocol to share the credential securely.

Step 6: Accenture Receives the Credential

- **Accenture** receives the **Verifiable Credential** from Virat, which contains:
    - The degree details (e.g., degree type, student ID, graduation date).
    - A **digital signature** from IIT Bombay to verify its authenticity.
    - The **DID** of IIT Bombay, confirming the credential's source.
- Accenture stores or processes the credential securely.

Step 7: Accenture Verifies the Credential

- **Accenture uses a credential verification system** to check the authenticity of the received credential.
    - **Verification System**: This system verifies the credential by performing the following checks:
        1. **Validating the Digital Signature**: Accenture ensures that the **digital signature** on the credential matches the public key associated with **IIT Bombay's DID** (Decentralized Identifier).
        2. **Checking the DID**: The DID confirms that the credential was issued by **IIT Bombay**. The system ensures that the DID is valid and corresponds to IIT Bombay's verified identity.
        3. **Authenticating the Credential**: The system confirms that the **credential has not been tampered with**, ensuring its integrity.
- If the digital signature and DID are valid, the credential is considered **authentic**.

Step 8: Accenture Confirms the Credential

- After successful verification, **Accenture confirms the authenticity** of Virat's **degree** from **IIT Bombay**.
- Accenture can now use this verified information to make informed decisions about Virat's eligibility for the job role.

Step 9: Accenture Makes a Decision

- **Accenture** proceeds with the next steps in the hiring process based on the verified credential, such as scheduling interviews or making an offer to Virat.

Technical Flow

Step 1→ Invitation and Receiving Phase

VIRAT		IIT BOMBAY
Create-Invitation	→	Receive Invitation
Accept-connection	←	Accept Invitation

Connection Established Successfully now both the VIRAT and IIT BOMBAY will get their connection id to contact each other.

Step 2→ Requesting Credentials from College

VIRAT		IIT BOMBAY
Send Proposal(for Credential)	→	Proposal Recieved
Offer-Recieved	←	Offer Send(with credential Info)
Request Sent (for Credential Confirmation)	→	Request Recieved
Credential Received (In Virat Wallet)	←	Credential Issued

Step 3→ Sharing Credential with the Accenture

$$\begin{tikzcd}
\text{ACCENTURE} & \text{VIRAT} \\
\text{Request Sent (Req for Required Cred)} \arrow[r] & \text{Request Received} \\
\text{Presentation Received (Credetial Rec)} & \text{Presentation Sent(Sent Vc in res)} \arrow[l]
\end{tikzcd}$$

Done/Abandoned:

- Both parties mark the process as completed (**done**) or terminate it if no further action is needed (**abandoned**).