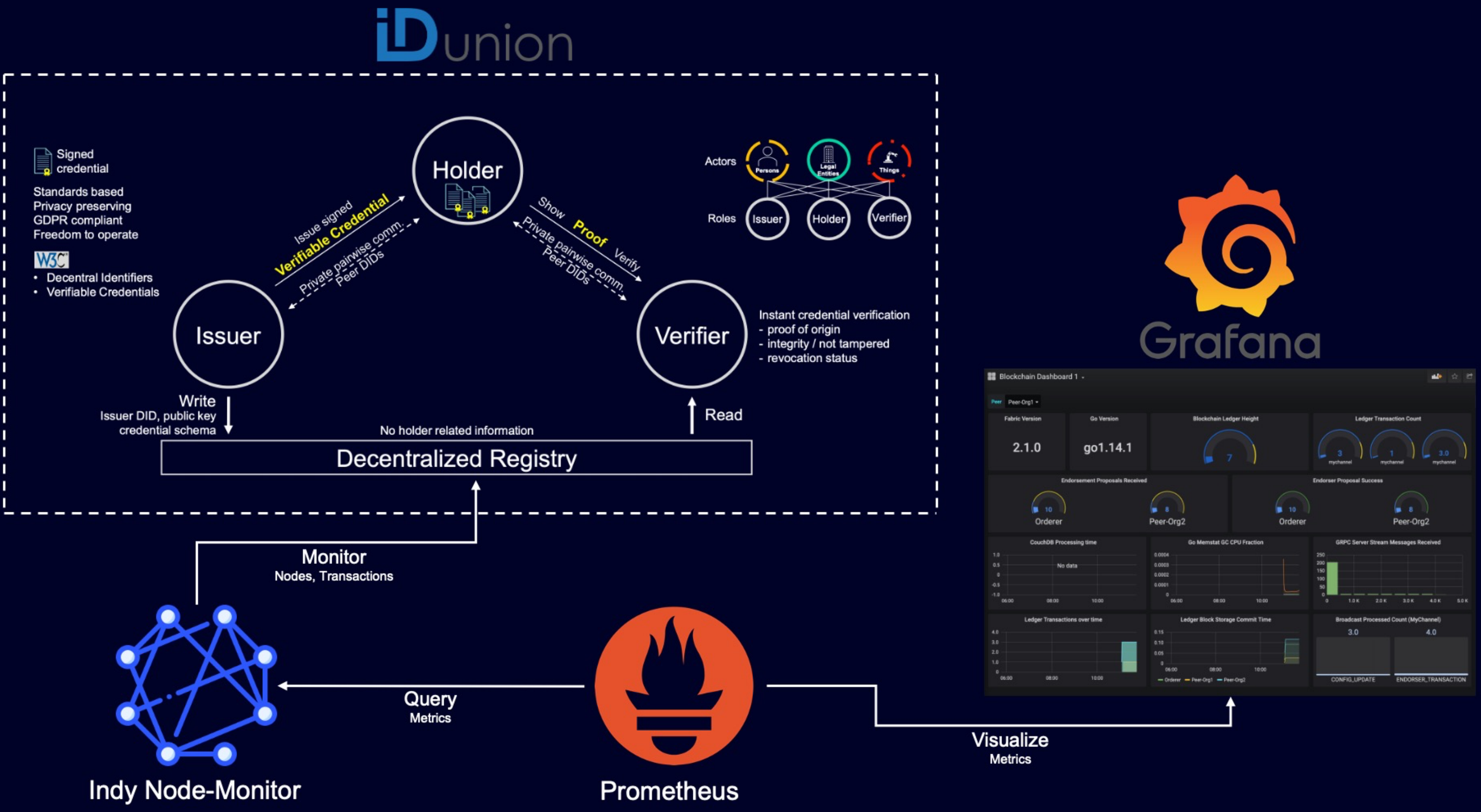


iDunion

Signed credential

Standards based
Privacy preserving
GDPR compliant
Freedom to operate

W3C
- Decentral Identifiers
- Verifiable Credentials

Holder

Issuer

Verifier

Issue signed Verifiable Credential

Private pairwise comm. Peer DIDs

Show Proof Verify

Private pairwise comm. Peer DIDs

Actors: Persons, Legal Entities, Things

Roles: Issuer, Holder, Verifier

Instant credential verification
- proof of origin
- integrity / not tampered
- revocation status

Write
Issuer DID, public key
credential schema

Read

No holder related information

Decentralized Registry

Monitor
Nodes, Transactions

Query
Metrics

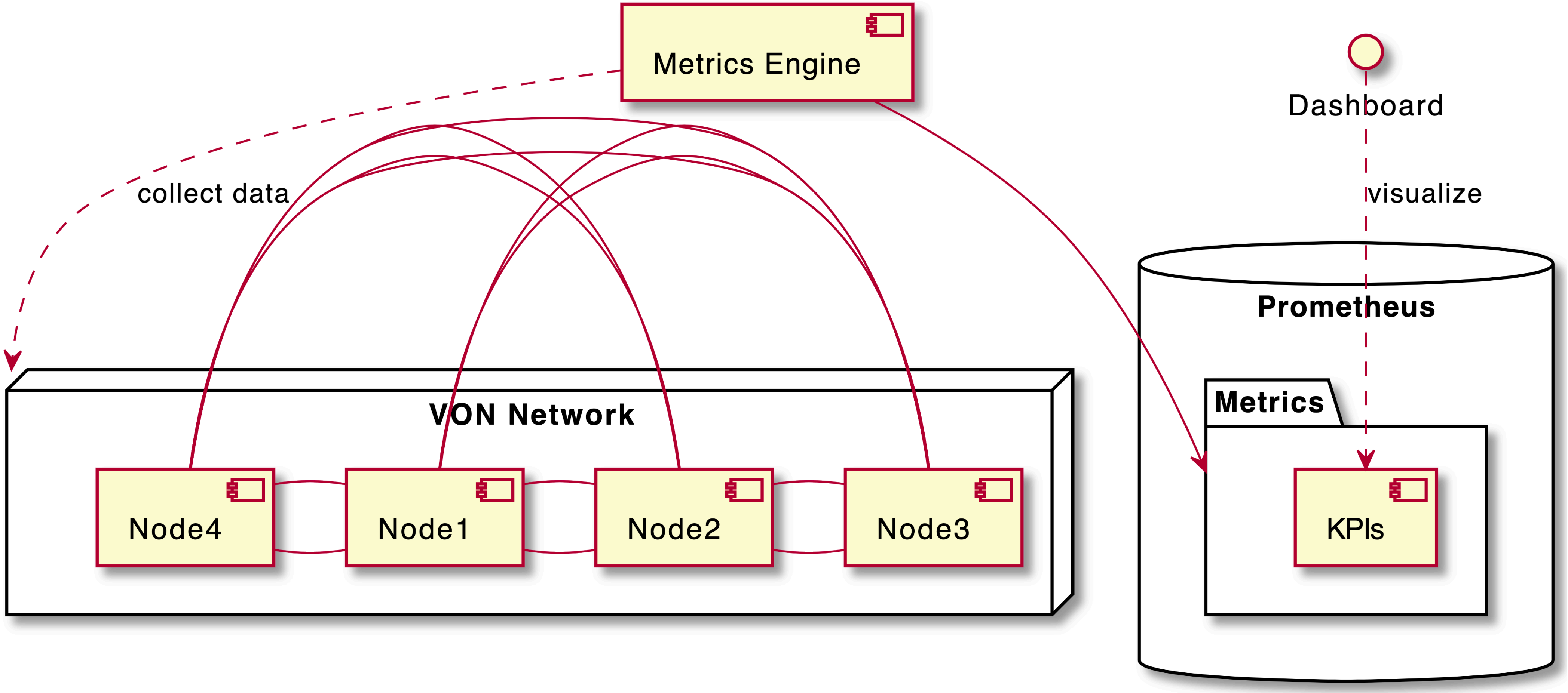
Visualize
Metrics

Indy Node-Monitor

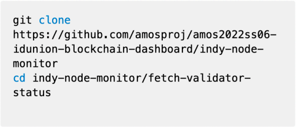
Prometheus

Grafana

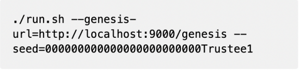
# Runtime Components



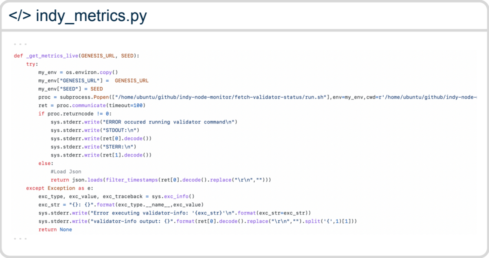
# Code Components



```
git clone
https://github.com/amosproj/amos2022ss06-
idunion-blockchain-dashboard/indy-node-
monitor
cd indy-node-monitor/fetch-validator-
status
```

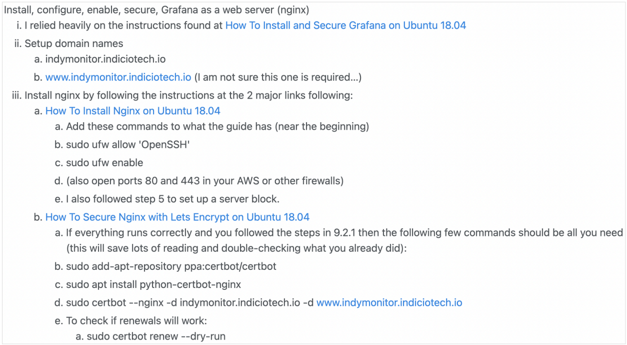


```
./run.sh --genesis-
url=http://localhost:9000/genesis --
seed=000000000000000000000000Trustee1
```



</> indy_metrics.py

```
...
def _get_metrics_live(GENESIS_URL, SEED):
    try:
        my_env = os.environ.copy()
        my_env["GENESIS_URL"] =  GENESIS_URL
        my_env["SEED"] = SEED
        proc = subprocess.Popen(["/home/ubuntu/github/indy-node-monitor/fetch-validator-status/run.sh"],env=my_env,cwd=r'/home/ubuntu/github/indy-node-
        ret = proc.communicate(timeout=100)
        if proc.returncode != 0:
            sys.stderr.write("ERROR occured running validator command\n")
            sys.stderr.write("STDOUT:\n")
            sys.stderr.write(ret[0].decode())
            sys.stderr.write("STERR:\n")
            sys.stderr.write(ret[1].decode())
        else:
            #Load Json
            return json.loads(filter_timestamps(ret[0].decode().replace("\r\n","")))
    except Exception as e:
        exc_type, exc_value, exc_traceback = sys.exc_info()
        exc_str = "{}: {}".format(exc_type.__name__,exc_value)
        sys.stderr.write("Error executing validator-info: '{exc_str}'\n".format(exc_str=exc_str))
        sys.stderr.write("validator-info output: {}".format(ret[0].decode().replace("\r\n","").split('{',1)[1]))
        return None
...
```



Install, configure, enable, secure, Grafana as a web server (nginx)

i. I relied heavily on the instructions found at How To Install and Secure Grafana on Ubuntu 18.04

ii. Setup domain names

a. indymonitor.indiciotech.io

b. www.indymonitor.indiciotech.io (I am not sure this one is required...)

iii. Install nginx by following the instructions at the 2 major links following:

a. How To Install Nginx on Ubuntu 18.04

a. Add these commands to what the guide has (near the beginning)

b. sudo ufw allow 'OpenSSH'

c. sudo ufw enable

d. (also open ports 80 and 443 in your AWS or other firewalls)

e. I also followed step 5 to set up a server block.

b. How To Secure Nginx with Lets Encrypt on Ubuntu 18.04

a. If everything runs correctly and you followed the steps in 9.2.1 then the following few commands should be all you need (this will save lots of reading and double-checking what you already did):

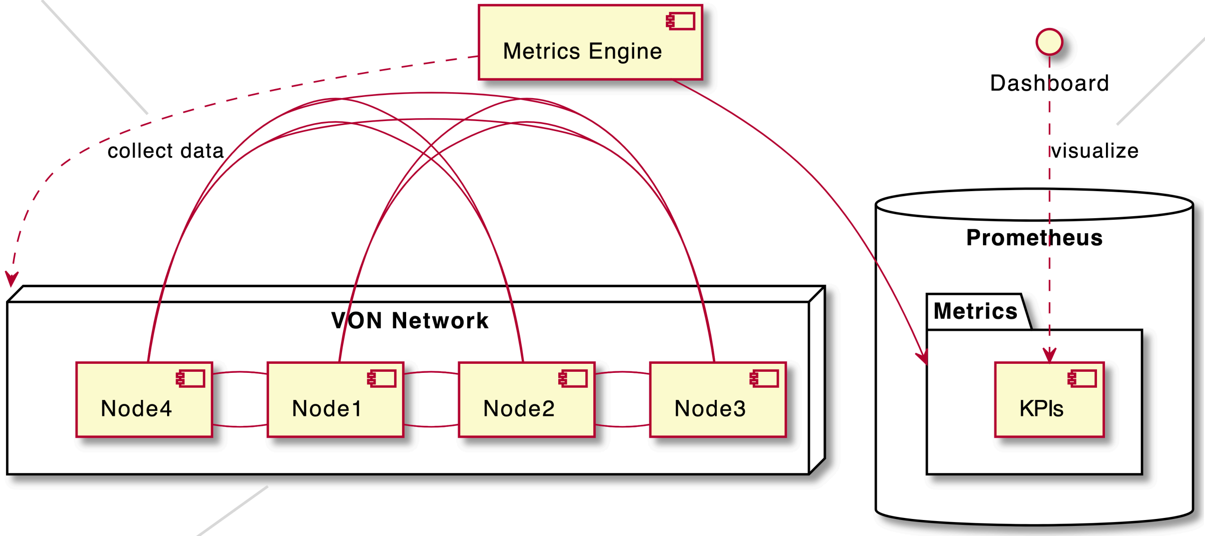
b. sudo add-apt-repository ppa:certbot/certbot

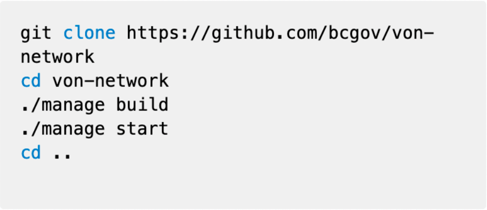
c. sudo apt install python-certbot-nginx

d. sudo certbot --nginx -d indymonitor.indiciotech.io -d www.indymonitor.indiciotech.io

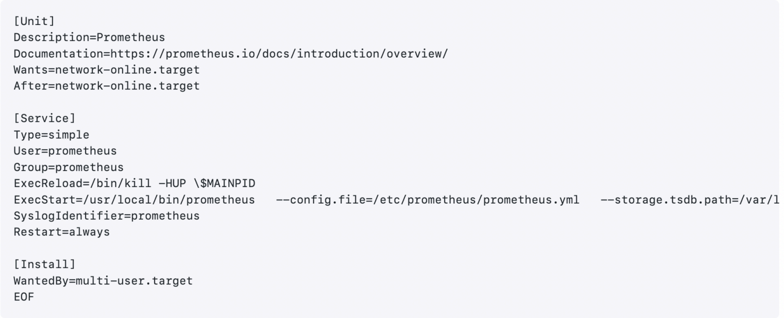
e. To check if renewals will work:

a. sudo certbot renew --dry-run





```
git clone https://github.com/bcgov/von-
network
cd von-network
./manage build
./manage start
cd ..
```

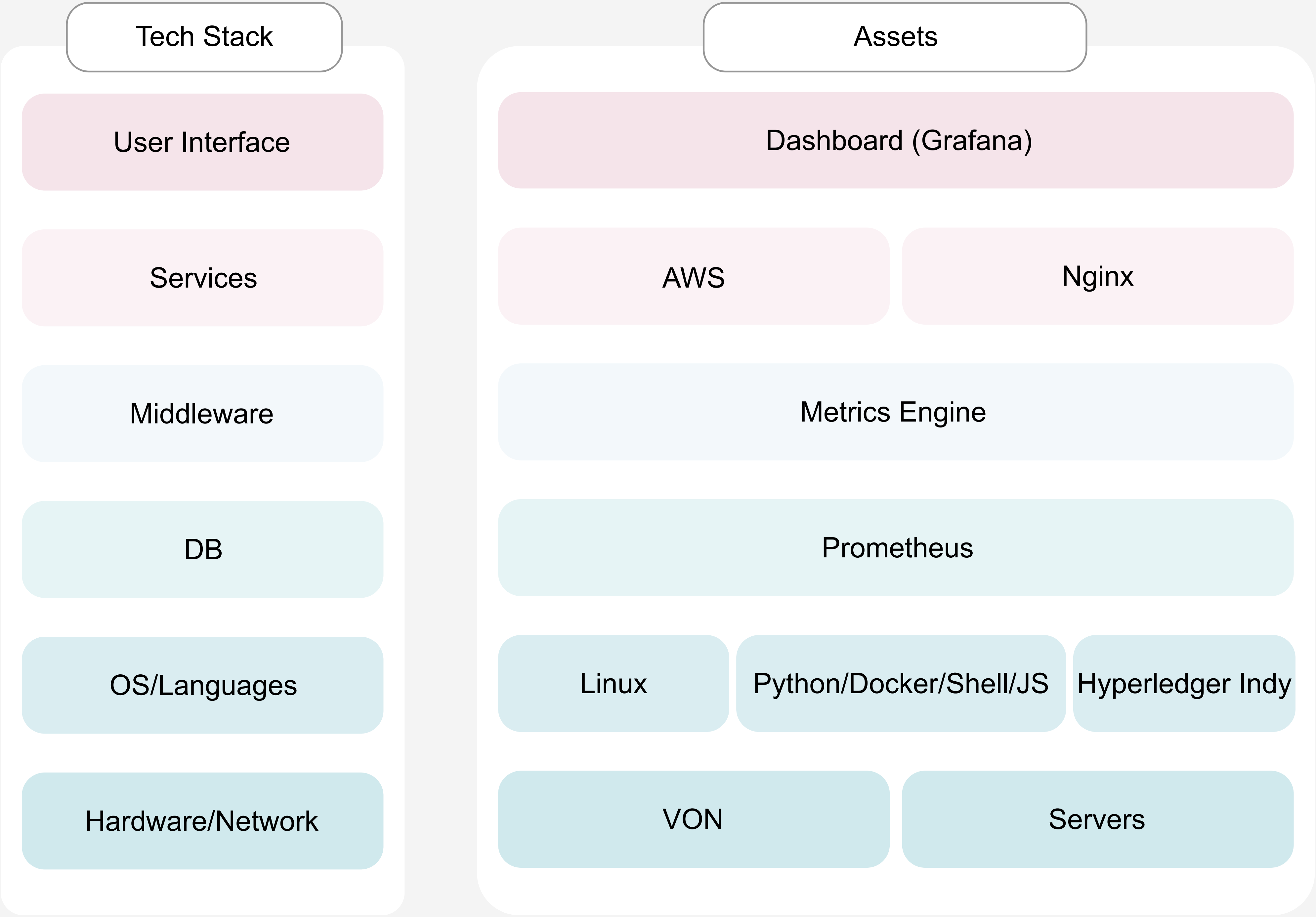


```
[Unit]
Description=Prometheus
Documentation=https://prometheus.io/docs/introduction/overview/
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=prometheus
Group=prometheus
ExecReload=/bin/kill -HUP \$MAINPID
ExecStart=/usr/local/bin/prometheus   --config.file=/etc/prometheus/prometheus.yml   --storage.tsdb.path=/var/l
SyslogIdentifier=prometheus
Restart=always

[Install]
WantedBy=multi-user.target
EOF
```

# Technology Stack



| Tech Stack | Assets | | |
|---|---|---|---|
| User Interface | Dashboard (Grafana) | | |
| Services | AWS | Nginx | |
| Middleware | Metrics Engine | | |
| DB | Prometheus | | |
| OS/Languages | Linux | Python/Docker/Shell/JS | Hyperledger Indy |
| Hardware/Network | VON | Servers | |

# Textual Explanation of Diagrams and Choices

## 1. Software Architecture - High-level view

- Starting point of the software architecture description was a high-level view on the planned architecture, mainly based on the presentation and meeting with Industry partner
- We started with the **IDunion** network, which lies at the core of our architecture
- Our Metrics Engine will be based on the **Indy Node-Monitor**, provided by the Hyperledger foundation
- The data, delivered by the Metrics Engine will then be stored in the Prometheus database
- Grafana and **Prometheus** work perfectly together, so we will use **Grafana** for the visualization part
- Every asset of the figure is open-source

## 2. Runtime Components

- Based on the preceding high-level view, we drew a structural UML Component diagram by using the open-source software http://www.plantuml.com/
- We chose a component diagram as it breaks down our system into different functionality levels
- Further, the component allows us to model the interface, which is our case the IDunion Dashboard
- The component diagram allows others to get a quick overview of the planned architecture of our system
- The **components** we used are the following:
  - Node
  - Component
  - Folder
  - Database
  - Interface
- **Limitations:** The component diagram is still a model of the reality, thus it doesn't yet mirror the reality - it needs to be revised in an iterative manner.

## 3. Code Components

- The code components were added to the existing component diagram
- Here, we focused on the most important steps to set-up our system, which can be found in different documentations, mainly based on the Hyperledger Indy documentation
- We structure the **code components** as follows:
  1. Start VON Network
  2. Fetch Validator Status
  3. Python Script for Metrics Engine
  4. Set-up Prometheus
  5. Set-up Grafana
  6. Connect Prometheus & Grafana

## 4. Technology Stack

- At the core of our Technology Stack lies the **Verifiable Organization Network (VON)** which allows us to set up the decentralized network, consisting out of different nodes
- Every node runs on a different **server**
- These servers run on **Linux** and the used programming languages are **Python, Docker, Shell and JavaScript**
- **Hyperledger Indy** is used as a library to connect the different nodes and to perform transactions, verifications within the VON
- Publicly available information on nodes and transactions can be queried by using the Indy Node Monitor
- This data is then stored in a **Prometheus** Database
- To get this data, a **metrics engine** will be developed
- Ultimately, the **Grafana Dashboard** should be available as a web server, by using **AWS** and **Nginx**