ecosteer.

DOOF DEPENDENCIES AND THIRD-PARTY SOFTWARE INSTALLATION

This document contains the installation, configuration and deployment of DOOF components and of their dependencies.

The scripts snippets and links contained herein refer to a deployment on Ubuntu Server v22.04.3 LTS 64-bit. The installation of this system is out of the scope of the current manual. Please make sure that your environment conforms to these minimum system requirements:

- RAM: 4 GB
- Disk space: 30 GB
- CPU: Double-Core vCPU

1 General dependencies

Add ecosteer user with admin privileges. This step is necessary for the deployment of the DOOF components, which come with configuration files that refer the ecosteer user and to its home directory.

```
sudo adduser ecosteer
sudo usermod -aG sudo ecosteer
```

SSH into your system with new user, and update all the packages:

```
sudo apt-get update && sudo apt dist-upgrade -y
```

2 Code deployment

Install git and set up all the keys and information that you need in order to have ssh access to the repository, or download the repository via HTTP or as tar ball and untar it in the NGI-TRUSTCHAIN/DOOF directory. Please respect the path constraints as the components have custom configuration files that refer to these paths. If you use other paths, you will need to update the configuration files and the environmental variables as well.

Create a root folder for the components code, and close the repository:

```
cd /home/ecosteer
mkdir NGI-TRUSTCHAIN && cd ~/NGI-TRUSTCHAIN
git clone git@github.com:NGI-TRUSTCHAIN/DOOF.git DOOF
```

Execute the general_dependencies.sh script provided in the repo under installation directory

```
cd ~/NGI-TRUSTCHAIN/DOOF/installation
./general_dependencies.sh
```

General_dependencies installs the following packages:

vim net-tools curl git openssh-server software-properties-common supervisor dos2unix apt-transport-https ca-certificates gnupg2 wget build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev python3-pip python3-venv uuid-dev libpq-dev

3 Database and related packages (for worker and processors)

The worker was tested with PostgreSQL RDBM, version 16. If you do not have a running instance of Postgres, this section gives you an indication of how to install it.

Install PostgreSQL version 16.

```
cd
sudo apt-cache search postgresql | grep postgresql
```

Start the installation by enabling Postgres official package repository by running the following command:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

Import the GPG signing key for the repository:

```
wget -qO- https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo tee /etc/apt/trusted.gpg.d/pgdg.asc &>/dev/null
```

Update the system packages so that the changes above can take effect:

```
sudo apt update -y
```

Now install the PostgreSQL 16 Database server and Client:

```
sudo apt install postgresql postgresql-client -y
```

Once the installation is complete, enable and start the PostgreSQL services:

```
sudo systemctl enable postgresql
sudo systemctl start postgresql
```

Test that PostgreSQL was installed successfully, and the service is active:

```
sudo systemctl status postgresql
```

3.1 unixODBC and psqlODBC

The following two packages need to be available on the machine where the worker is running.
Install unixODBC for the usage of the Database by the worker.

```
wget https://www.unixodbc.org/unixODBC-2.3.11.tar.gz
```

Unzip, untar:

```
gunzip unixODBC-2.3.11.tar.gz
tar xvf unixODBC-2.3.11.tar
```

Then go inside the folder execute:

```
cd unixODBC-2.3.11
./configure
make
sudo make install
```

```
sudo apt install -y unixodbc-dev
```

Eliminate the tar files and the installation folder.

Install psqlODBC for the usage of the Database by the worker.

```
cd
wget https://ftp.postgresql.org/pub/odbc/versions.old/src/psqlodbc-16.00.0000.tar.gz
```

Unzip, untar:

```
gunzip psqlodbc-16.00.0000.tar.gz
tar xvf psqlodbc-16.00.0000.tar
```

Then go inside the folder and execute:

```
cd psqlodbc-16.00.0000/
./configure
make
sudo make install
```

Copy the odbcinst.ini file provided in the installation directory under /usr/local/etc:

```
cd
sudo cp NGI-TRUSTCHAIN/DOOF/installation/database/odbcinst.ini /usr/local/etc
```

4 Python

Install the specific version of Python used by the DOOF components which is v.3.9.10 by typing these commands one-by-one:

```
cd
wget https://www.python.org/ftp/python/3.9.10/Python-3.9.10.tar.xz
tar -xf Python-3.9.10.tar.xz
cd Python-3.9.10
./configure -prefix=/usr/local --enable-shared LDFLAGS="-Wl,-rpath /usr/local/lib"
sudo make altinstall
```

Test that Python was installed successfully:

```
python3.9 --version
```

Install the Python requirements that you find in py_requirements.txt file under installation directory:

```
cd
mkdir -p virtualenv && cd virtualenv
python3.9 -m venv dop
source dop/bin/activate
pip install wheel
pip install -r ~/NGI-TRUSTCHAIN/DOOF/installation/py_requirements.txt
```

5 uwsgi

Install uwsgi, needed by the DOOF Gateway

```
sudo apt-get install uwsgi -y
```

6 Message Broker (mosquitto)

Install a message broker locally, or otherwise use a message broker that you may already have. Please take note of IP address / host name of the broker and port on which it is reachable, as it will be used for the configuration of different components.

use the given command to add the PPA repo provided by the official developers of this tool:

```
sudo add-apt-repository ppa:mosquitto-dev/mosquitto-ppa
```

Use the system's default apt package manager to install Mosquitto message broker on a given operating system:

```
sudo apt install -y mosquitto mosquitto-clients
```

Configure mosquitto broker listeners by adding configuration files to the conf.d folder:

```
cd /etc/mosquitto/conf.d
sudo nano mqtt.conf
```

Insert this content to the configuration file and save it:

```
# MQTT Configuration
listener 1883 0.0.0.0
```

Now create a configuration file for websockets listener:

```
cd /etc/mosquitto/conf.d
sudo nano ws.conf
```

Insert this content to the configuration file and save it:

```
# WS Configuration
listener 8083
protocol websockets
```

Finally, configure the broker itself to accept anonymous connections:

```
cd /etc/mosquitto
sudo nano mosquitto.conf
```

Insert these lines at the end of the configuration file:

```
allow_anonymous true
```

Save the file and restart the broker:

```
sudo systemctl restart mosquitto.service
```

Check the status of the service:

```
sudo systemctl status mosquitto
```

6.1 WSS

For the deployment of the front-end used for the pilot, WSS needs to be configured.

Copy the certificates to a directory where `mosquitto` user can access them:

```
sudo cp /path/to/your/certs/chain.pem /etc/mosquitto/certs
sudo cp /path/to/your/certs/privkey.pem /etc/mosquitto/certs
sudo cp /path/to/your/certs/fullchain.pem /etc/mosquitto/certs
```

Change the owner of these files to mosquitto user:

```
sudo chown -R mosquitto:mosquitto /etc/mosquitto/certs
```

Set the appropriate permissions:

```
sudo chmod 400 /etc/mosquitto/certs/chain.pem
sudo chmod 400 /etc/mosquitto/certs/fullchain.pem
sudo chmod 400 /etc/mosquitto/certs/privkey.pem
```

Now create a configuration file for websockets secure:

```
cd /etc/mosquitto/conf.d
sudo nano wss.conf
```

Insert this content to the configuration file and save it:

```
# WSS Configuration
listener 8084
protocol websockets

cafile /etc/mosquitto/certs/chain.pem
keyfile /etc/mosquitto/certs/privkey.pem
certfile /etc/mosquitto/certs/fullchain.pem
```

Save the file and restart the broker:

```
sudo systemctl restart mosquitto.service
```

Important note: you will need to either configure post-renewal hooks or crontab to automatically copy and renew the certificates also in the mosquito directory.

7 Message Queueing (RabbitMQ)

Enable the RabbitMQ PPA repository on your system and import rabbitmq signing key:

```
echo 'deb http://www.rabbitmq.com/debian/ testing main' | sudo tee /etc/apt/sources.list.d/rabbitmq.list
wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add -
```

Update the apt cache and install the RabbitMQ server on your system:

```
sudo apt-get update
sudo apt-get install rabbitmq-server -y
```

After completing installations, enable and start the RabbitMQ service on your system:

```
sudo systemctl enable rabbitmq-server
sudo systemctl start rabbitmq-server
```

Change the default user and give it all the permissions:

```
sudo rabbitmqctl add_user ecosteer ecosteer
sudo rabbitmqctl set_user_tags ecosteer administrator
sudo rabbitmqctl set_permissions -p / ecosteer ".*" ".*" ".*"
```

Disable guest user:

```
sudo rabbitmqctl clear_password guest
```

Restart RabbitMQ service:

```
sudo systemctl restart rabbitmq-server
```

8 NGINX

Install NGINX:

```
sudo apt install -y nginx
```