1. Installation

1.1. Install Hyperledger Fabric

1.1.1. Install Prerequisites

Note: if you are using a windows operating system, please install within a Linux subsystem or Linux Virtual Machine. We recommend using WSL2. This is important for running the shell scripts.

#	Name	Description	Link
1	Git	Required to install fabric samples and images	https://git-scm.com
2	cURL	Required to install fabric samples and images	https://curl.se/download.html
3	Docker (and Docker compose)	Required to run the test network	https://docs.docker.com/get-docker/
4	Node js	Required to run client applications and chaincode	https://nodejs.org/en/
5	OS Build Tools	node-gyp is used to build the fabric SDK. Requires C++ compiler provided by Build Tools.	https://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html#before-you-begin

Note: Hyperledger Fabric provide instructions for installing Prerequisites for each OS:

https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html

1.1.2. Install Fabric Samples and Images

1. Create a parent directory to store Hyperledger Fabric and COMP6452-Project2 (referred to as <PROJ_HOME>)
2. Download Fabric Samples, Docker Images and Binaries to <PROJ_HOME>:

```
curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.3.2 1.5.0
```

After completing this step, you should have the following directory created:

`<PROJ_HOME>/fabric-samples`

1.2. Install Comp6452 Project

Extract COMP6452-Project2.zip to the <PROJ_HOME> directory.

Note: If you are using WSL2 on windows please install into the WSL2 file system, not into the windows file system (e.g. Do not install on a wsl2 path that starts with /mnt/c/…).

You should now have the following directories in <PROJ_HOME>:

- `<PROJ_HOME>/COMP6452-Project2`
- `<PROJ_HOME>/fabric-samples`

2. Operation

2.1. Test the Chaincode (optional)

You can optionally test the three Chaincode classes using the following commands.

You can investigate how the script is implemented to do it manually if you wish.

1. `cd <PROJ_HOME>/COMP6452-Project2`
2. `source deploySC.sh vehicle-chaincode test`
3. `source deploySC.sh registration-chaincode test`
4. `source deploySC.sh violation-chaincode test`

Note: scripts must be run using `source` as above

2.2. Setup the Test Network

Notes:

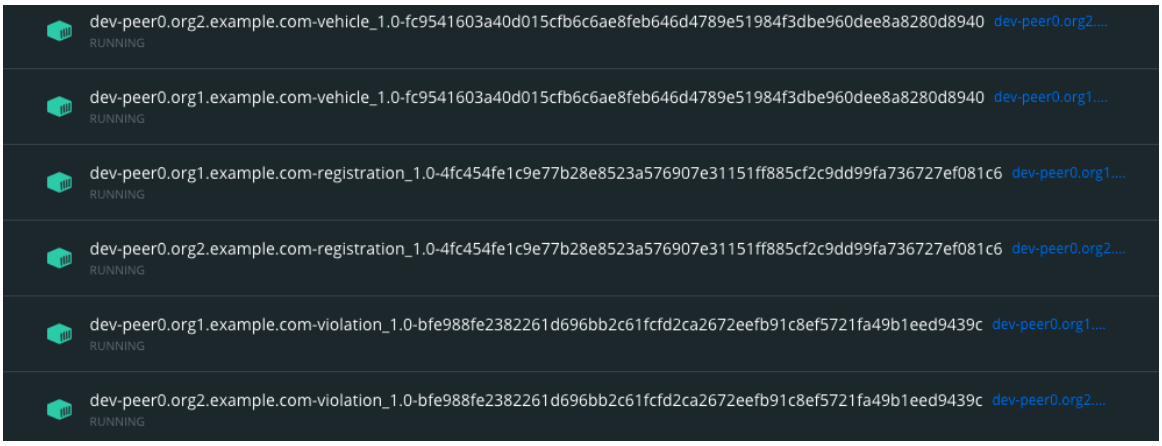
1. A convenience script has been provided which performs the following tasks: start test network, deploy chaincode, run wallet generator and run database generator. If experiencing compatibility issues when running the convenience script, please refer to **Appendix A** for details on performing the individual steps using the scripts provided by Hyperledger.
2. Ensure **Docker** is running before you start the Test Network

To run the convenience setup script:

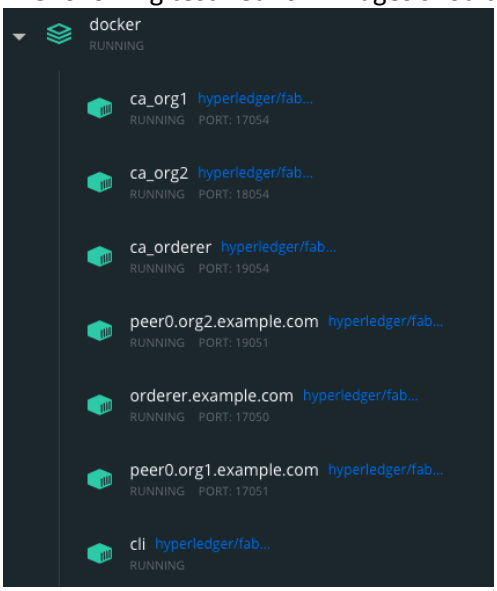
1. `cd <PROJ_HOME>/COMP6452-Project2`
2. `chmod 755 *.sh` (execute permission applied to .sh files)
3. `source setup.sh`

This should take a few minutes.

The following Chaincode Peers should now be running in Docker:



The following test network images should now be running in Docker:



If you do not see something like this, please check the following:

1. Prerequisite steps have been followed
2. Docker is running
3. Execute permission is applied to scripts: "`chmod 755 *.sh`"
4. If issues with Windows carriage return characters run: "`dos2unix *.sh`"
5. Then run "`source setup.sh`" again.

2.3. Start the Client Applications

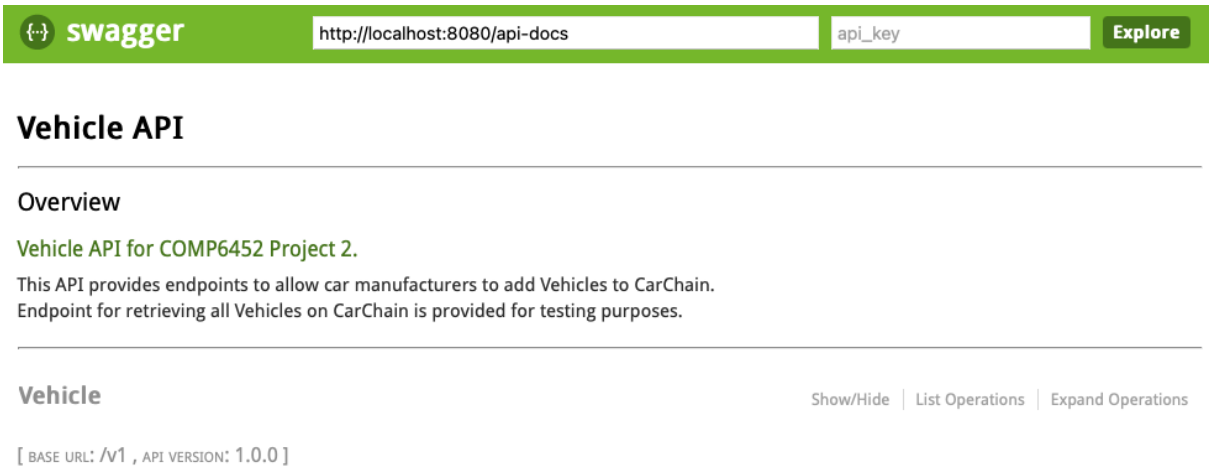
2.3.1. Start Vehicle API

Vehicle API is used by the manufacturer to add Vehicles to the blockchain.

1. Open a new command prompt
2. cd <PROJ_HOME>/COMP6452-Project2/client-applications/vehicle-api-server
3. npm start

If successful, you can Add Vehicles via the Swagger UI by opening the following URL in a Browser:

http://localhost:8080/docs



2.3.2. Start Registration Portal

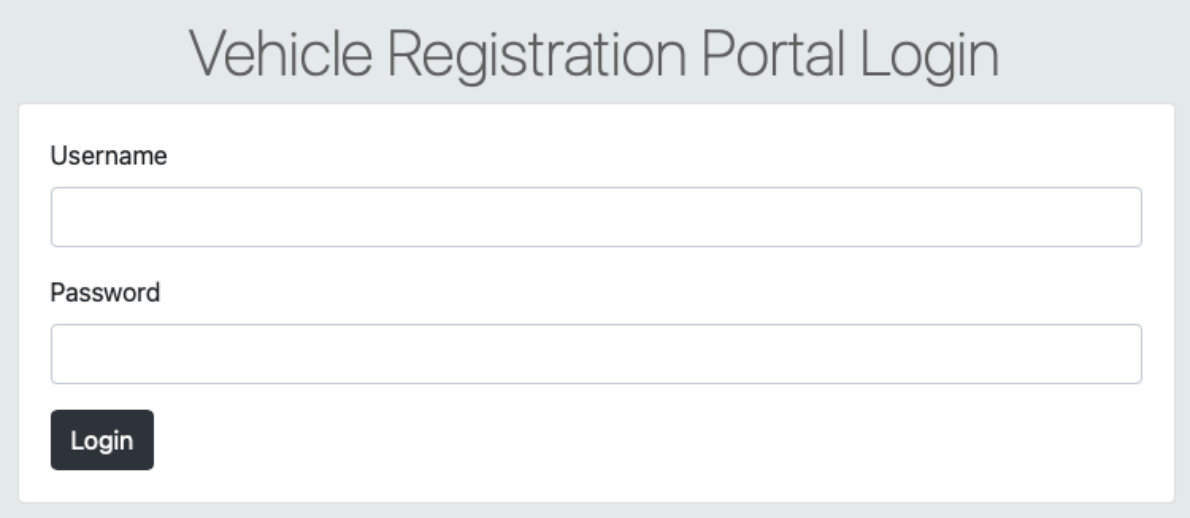
Registration Portal is used by the Customer to create Registrations and pay Fines.

Registration Portal is used by the Employee to approve/reject Registrations.

1. Open a new command prompt
2. cd <PROJ_HOME>/COMP6452-Project2/client-applications/registration-portal
3. npm start

If successful, you can access the Login page by opening the following URL in a Browser:

http://localhost:3001/login



Note: The **Database Generator** is used to insert users into the database.

The following Users are provided by default:

Username	Password	Type
CRN000000001	Test111	Customer
CRN000000002	Test222	Customer
CRN000000003	Test333	Customer
EMP000000001	Test111	Employee

2.3.3. Start Police API

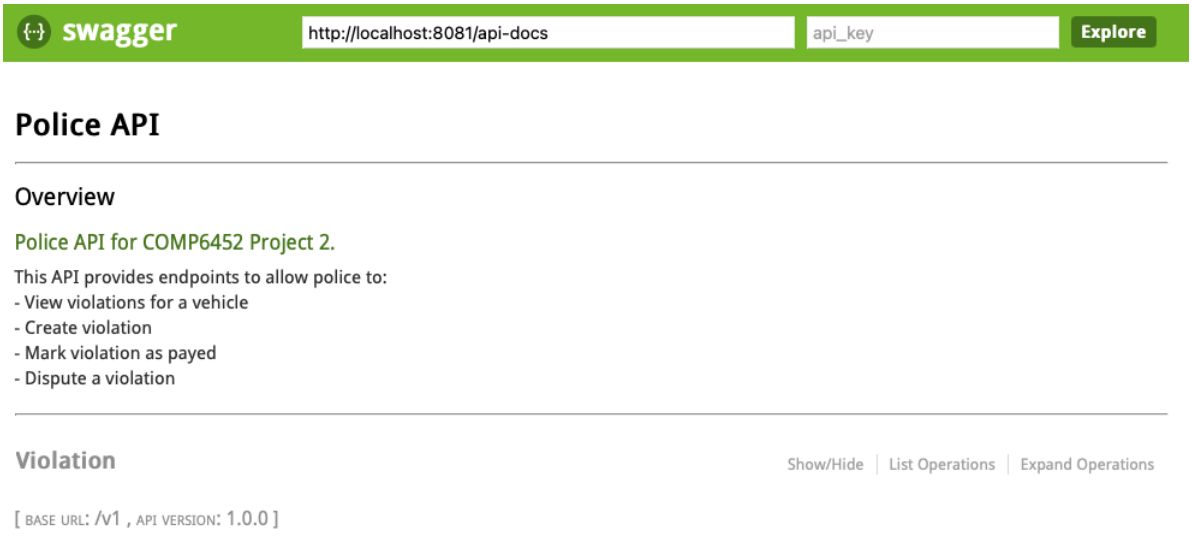
The Police API is used by Police (e.g. speed camera) to add a traffic violation to the blockchain.

The Customer can view and pay fines related to violations in the Registration Portal

1. Open a new command prompt
2. cd <PROJ_HOME>/COMP6452-Project2/client-applications/police-api-server
3. npm start

If successful, you can Add Violations via the Swagger UI by opening the following URL in a Browser:

http://localhost:8081/docs



2.4. Stopping the Test Network

The following command will stop the Docker Images and tear down the Test Network.

1. `cd <PROJ_HOME>/fabric-samples/test-network`
2. `./network.sh down`

3. Appendix A

This section contains instructions for starting the Test Network using the scripts provided in the fabric-samples directory.

Note: These scripts should only be used if experiencing compatibility issues when using the convenience scripts provided in Section 2.1 and 2.2.

These steps are to be repeated each time the Test Network is shutdown.

3.1. Start Test Network

Note: ensure Docker is running before you start the Test Network

1. `cd <PROJ_HOME>/fabric-samples/test-network`
2. `./network.sh down`
3. `./network.sh up createChannel -c carchainchannel -ca`

Note: to shutdown network and all blockchain components simply run "./network.sh down"

3.2. Deploy Chaincode

COMP6452-Project2/smart-contracts contains three Chaincode projects:

1. registration-chaincode
2. vehicle-chaincode
3. violation-chaincode

Deploy the three Chaincode projects to Hyperledger:

1. `cd <PROJ_HOME>/fabric-samples/test-network`
2. `./network.sh deployCC -ccn vehicle -ccp ../../COMP6452-Project2/smart-contracts/vehicle-chaincode -ccl javascript -c carchainchannel`
3. `./network.sh deployCC -ccn registration -ccp ../../COMP6452-Project2/smart-contracts/registration-chaincode -ccl javascript -c carchainchannel`
4. `./network.sh deployCC -ccn violation -ccp ../../COMP6452-Project2/smart-contracts/violation-chaincode -ccl javascript -c carchainchannel`

3.3. Run the Wallet Generator

Each Client Application requires an Identity and Wallet to access the network.

1. `cd <PROJ_HOME>/COMP6452-Project2/utilities/wallet-generator`
2. `npm start`

3.4. Run the Database Generator

The Registration Portal relies on an sqlite3 database to store the offchain data e.g. Customer details.

1. `cd <PROJ_HOME>/COMP6452-Project2/utilities/database-generator`
2. `npm start`