AstraKode Blockchain Platform

Astrakode Team

22/12/21

Contents

1 Your Hyperledger Fabric Network

This repository contains an example of the code generated by AKB platform; it is configured to run a simple Fabric network with:

- 1 **consortium** with 1 **channel**
- 1 organization of type **orderer**, 1 ordering service and 1 orderer node
- 1 organization of type **client** containing 1 client node
- a **certificate authority** for each organization
- a default simple **chaincode** to interact with

2 Compatible operative system

The Fabric network is set up through a series of bash scripts, so it can be run on:

- Linux
- Windows (e.g. on WSL, git bash or cgwin)
- 1 MacOs (under testing)

3 Requirements

An introduction to hyperledger fabric, containing a high-level explanation of the different components, can be found `https://deeptiman.medium.com/an-introduction-to-hyperledger-fabric-a58094ac5717` Its software requirements are:

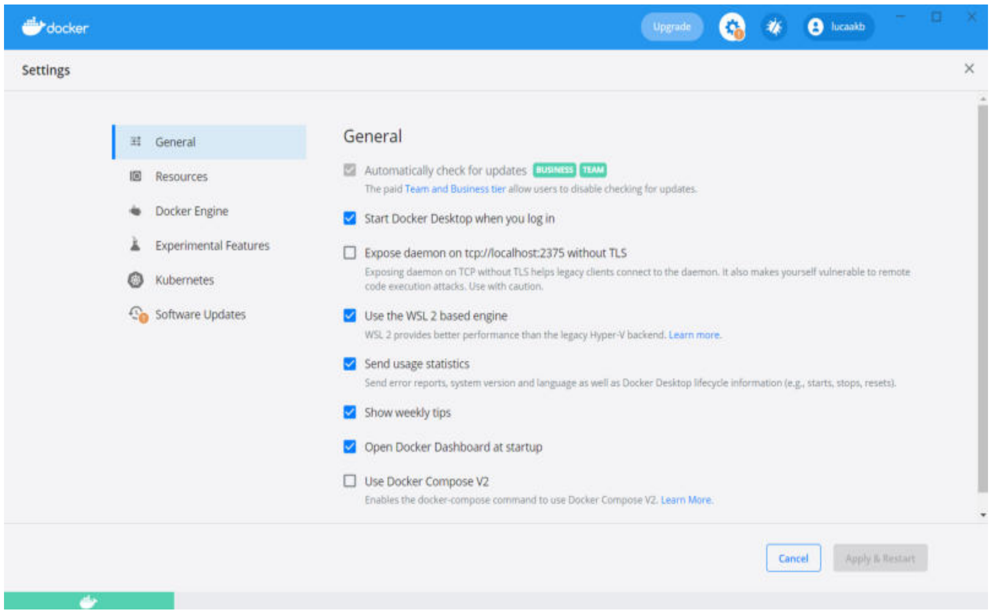
- bash, usually preinstalled on the systems above
- hyperledger fabric prerequisite for WINDOWS `https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html#windows`
- 1 hyperledger fabric prerequisite for LINUX `https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html#linux`

- hyperledger fabric prerequisite for MAC
  `https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html#mac`

Please note that the **jq** requierment is optional in the above links, but it **has to be installed** in order to run the default chaincode provided by AKB.

3.1 Docker Desktop settings

In order to work correctly, the docker desktop must have the following settings



General

- [✓] Automatically check for updates BUSINESS TEAM
  The paid Team and Business tier allow users to disable checking for updates.
- [✓] Start Docker Desktop when you log in
- [ ] Expose daemon on tcp://localhost:2375 without TLS
  Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.
- [✓] Use the WSL 2 based engine
  WSL 2 provides better performance than the legacy Hyper-V backend. Learn more.
- [✓] Send usage statistics
  Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).
- [✓] Show weekly tips
- [✓] Open Docker Dashboard at startup
- [ ] Use Docker Compose V2
  Enables the docker-compose command to use Docker Compose V2. Learn More.

Furthermore, if any Windows Subsystem for Linux is used, the WSL 2 must be enabled on that subsystem. To check which WSL is being currently used, run the following command in the windows cmd :

```
$ wsl -l -v
```

To enable the wsl 2 run:

```
$ wsl --set-version subsystem_name 2
e.g. wsl --set-version Ubuntu-20.04 2
```

4 Run

Before launching the fabric setup, you may need to make bash scripts executable on your machine, for instance with:

```
$ find -name "*.sh" -type f -exec chmod +x {} \;
```

Then, to launch the docker setup, *cd* into the *bc-network* directory and run:

```
./network.sh up
```

1

After a while, if the process executed successfully, you will see

```
NETWORK LAUNCHED SUCCESSFULLY
```

printed on the console. This means you can examine your newly created Fabric network and interact with it. Find out what *network.sh* can do for you by running:

```
$ ./network.sh -h
```

5 Explore the network

Through *docker* commands, you are able to explore the nodes you just launched and look into their behavior; here are a few examples:

```
docker network ls # lists running docker networks
docker ps # lists running docker containers, i.e.: nodes, certificate auth
docker logs [-f] <container> # prints the logs from a given container, allo
docker exec [-it] <container> <command> # allows you to send commands dire
```

6 Structure of the setup

This fabric network is configured through a set of files, in particular:

- *globalParams.sh* lists general information on the network like channels, organizations, ...
- each organization's folder contains a *configParams.sh*, which collects information about nodes belonging to the organization

[1]Note: you can cleanup everything with *./network.sh down*.

- 1 *chaincodeParams.sh* contains information about chaincodes and their deployment

Additionally, a number of scripts are handling different parts of the setup:

- *network.sh* acts as an orchestrator script and it's the only script (alonside with *scripts/chaincode.sh*) meant to be called via command line.
- each organization of type client has a *setupClientOrg.sh* script that sets up the Certification Authority and ensures that every member of the network has the require level of authorization;
- *setupOrdererOrg.sh* is analogous for organizations of type order.
- each organization of type client has a *clientsUp.sh* script that launches a docker container for each of its nodes;
- *orderersUp.sh* is the analogous for organizations of type order.
- *createChannel.sh* is the scipt that manages channels; it also adds peers to a channel and initializes it according to the configuration.

[2]

7 Chaincode

Alongside with your Fabric setup, *network.sh up* also installed and deployed a simple chaincode (*DeployCC* on every channel in order to perform an *end to end test*. Another chaincode is present into the caincode folder, *fabcar*. For more information on how to use it check the following link
`https://hyperledger-fabric.readthedocs.io/en/release-1.2/understand_fabcar_network.html`

7.0.1 Custom Chaincode

It is also possible to deploy and invoke custom chaincodes via the *chaincodeMain.sh* script.

Usage: chaincodeMain.sh [OPTS] MODE

[2]Note: you can edit these configuration files in order to modify the network's topology; emulate the existing structure in order for the resulting setup to be functioning.

```
MODES:
    -deploy   --> deploy the specified chaincode
    -invoke   --> invoke the specified chaincode
    -query     --> query the ledger using the specified chaincode
```

OPTS (default values in *globalParams.sh* or *customChaincodeParams.sh*)

```
    -h    --> print help message
    -c    --> index identifying the chaincode in customChaincodeParams.sh
    -l    --> set verbosity:1->error,2->warning,3->info,4->debug,5->trace
    -v    --> verbose output: same as -l 4
```

A default custom chaincode, Fabcar, is already configured in customChaincodeParams.sh. By creating additional chaincode parameters (or replacing the default ones) in customChaincodeParams.sh one is able to utilize the *chaincodeMain.sh* script to deploy and invoke custom chaincodes.