Twig Instructions

Task Overview

Welcome to Twig! To assist you in your journey to become a Corda developer, you will set off on a practical journey to deploying your very first Cordapp! On the Twig platform, we will have you spin up a node on the server we have provisioned for you and then develop your own asset issuance Cordapp. The first task involves spinning a node on the server. The second task will require you to write a simple Cordapp in which you will (self) issue asset states to yourself. Finally, you will upgrade your Cordapp by adding the functionality to trade your assets on the network.

We will follow the proper procedures as required by Corda (e.g. when upgrading contracts, we must upgrade our states and ensure backwards compatibility and so on). We will leave no stone unturned. By the end, you will be a more competent Corda developer.

Housekeeping

- There's a cheatsheet with useful functions that we have provided for you.
- Keep in mind that when looking up Corda documentation, you need to look at the documentation for V3.4.
- Cloud9 doesn't allow multi-line commenting in kotlin files, so therefore use `alt+click` and drag to select multiple lines.

Getting Started

- Log into AWS using the link provided.
- Check that your region is set to Oregon (us-west-2).
- Go to the Cloud9 Service.
- Click on the hamburger on the left and go to "Shared with you"
- Start up your environment.

Task 1: Joining The Corda Network

To access your Cloud9 IDE, click here. Your username is the 6 letter alphanumeric ID located in the top right of the TWiG dashboard, underneath your name. Your instructor will provide you with the password.

In task 1, you will spin your own node and join our network.

```
// before running your node, you must first prepare your node
configuration file
// you will need your node's ip for this. get that with the following
command.
curl http://169.254.169.254/latest/meta-data/public-ipv4
```

Edit the node configuration file (located in the node folder) named "node.conf". See the node configuration documentation for more details on the below options.

- Update "myLegalName" with the correct name. It must abide by Corda's X500Name format. The name of the organisation of your node corresponds to your unique 6 letter alphanumeric key (e.g. ab1234).
- Update the "p2pAddress" with the address of the ip address of your server (the above curl command) along with the port **8080**.
- Ensure the "compatibilityZoneURL" is correct. It should be "35.162.51.152:10100"
- If you are unfamiliar with RPC, you can read more about it here. It is the main method of connecting to and communicating with the node.

```
// once you have created your configuration file and checked that it is
correct, you will need to
// get a copy of the truststore from the network map service. The
network map service maintains
// the network in Corda.

// this command retrieves the truststore and saves it as a file named
"network-truststore.jks"
curl -o ./network-truststore.jks -X GET
35.162.51.152:10100/network-map/truststore
```

Now that you have the truststore, you are ready to join the network. Go through the following checklist before joining the network. If you make any errors in your configuration file, it could lead to your node not being recognised on the network.

- Ensure your myLegalName is in the correct format. If your username is "ab1234" then it should be "O=ab1234,L=Sydney,C=PH".
- Ensure your IP and port are correct. If either value is incorrect, your node will not be contactable.
- Ensure the compatibilityZoneURL is correct. Or else you may end up joining some other network and feel left out.

```
// now that you have checked your node configuration file, you are ready
to join the network
// the following command registers your node onto our network. the
network map service will become
// aware that you are a part of our network. the password for the
truststore is "trustpass".

// node logs are stored in node/logs
// if errors occur, check here for clues

cd ~/environment/node/
java -Xmx1024m -jar corda.jar --initial-registration
--network-root-truststore <location-of-truststore>
--network-root-truststore-password <password>

// now that you have registered, you can run your node
java -Xmx1024m -jar corda.jar

// pay attention to any Cordapps that your node loads.
// if you make any mistakes in your node configuration, edit it and run
the registration command again
```

Task 2

Task 2 will have you write a Cordapp whereby you will be able to self issue assets to yourself via the node RPC. This task has been structured as a series of failing tests that you have to correct one at a time. Your main working folders for this task will be asset`-contracts/src/main/kotlin/com/asset/issue` and asset`-workflows/src/main/kotlin/com/asset/flow`. The failing tests that you will have to fix (by implementing the features in the respective folders) are located in asset`-contracts/src/test/kotlin/com/asset/issue` and asset`-work`

`flows/src/test/kotlin/com/asset/flows`. In particular, you have to fix the tests in *AssetStateTests, AssetContractTests* and *AssetIssue FlowTests* for task 2.

The recommended order in which you fix the tests is states, followed by contract, followed by flow. The instructions on how to fix the tests will be within the test files as well as the files you have to implement. Comment out each test and then follow the instructions (you may use the cheat sheet, documentation on the official site (V3.4!) to help you). If you have additional questions, ask your supervisor.

```
// test logs are stored in <project-name>/build/test-results/test/
// the following command will build all projects
./gradlew build

// the following command build's a single project
./gradlew <project-name>:build

// test logs are located in cordapp/build/test-results
// if errors occur, check here for clues

// the following command runs all tests
./gradlew test

// the following command runs a single project's test
./gradlew <project-name>:test

// the following command runs a single classes' test
./gradlew test --tests <class-name>

// the following command runs a single test in a single class
./gradlew test --tests <class-name>.<test-name>

// the following command runs the integration tests
./gradlew integrationTest
```

Once you have all your *AssetStateTests, AssetContractTests* and *AssetIssueFlowTests* are passing, you can move onto issuing your asset.

```
// the location of the jars are in the
asset-[contracts|contracts-new|workflows]/build/libs/ directory

// before we can issue the asset, we must first upload our Cordapp (so
that all nodes will have access to it)
// this will upload your Cordapp.
aws s3 cp <location-of-jar> s3://twig-game-cordapps

// now that you have submitted your Cordapp, you must also upload it to
your own Corda node. you can do this by
// shutting down your node, copying the cordapp into the "node/cordapps"
folder and restarting your node.
// notice that the node now loads your Cordapp.

// note that if you submit your cordapp and then rebuild it, you must
resubmit it or else Corda will not accept it
// see contract constraints for more information.

// afterwards, we will issue our Cordapp by using a CRaSH shell. this
shell allows us to run flows directly from the node
// ssh into your node's CRaSH shell. the user should be "user1" (unless
you changed it in your node configuration)
ssh -p 2222 localhost -l user1

// type yes to continue connecting. the password is "test" (unless you
changed it in your node configuration)
// within the shell, you can do a lot of things (e.g. query the vault)
but we want to run your flow
// let's start by listing all flows that are installed on your node (via
Cordapps)
flow list

// you should notice that there is a flow called
"com.<id>.flow.AssetIssueFlow$Initiator". Copy up to the "$" symbol.
// we will run our flow using the [flow start] command.
// it should look something like this: "flow start
com.<id>.flow.AssetIssueFlow amount: <amount>"
// you should receive a SignedTransaction id as output
```

Task 3

Task 3 involves upgrading the Cordapp to include functionality that would allow you to trade it. It must also include support for commissions. Task 3 has been structured similarly to task 2, except that now you will have to port your old contract to the new contract (and upgrade it).

The new working folders are asset-`contracts-new/src/main/kotlin/com/asset/trade` and asset-`workflows/src/main/kotlin/com/asset/flow`.

Once you have finished all your tests, make sure you also complete the integrationTests. They are in asset-`workflows/src/integrationTest/kotlin/com/asset/flow`.

```
Once all your tests pass, you must:

* Submit your Cordapp
* Copy the cordapp into your node's cordapp folder
* Restart your node
* Upgrade your AssetStates to AssetStateNew (we've done the heavy
lifting for you here) by running the following command in your cordapp
folder
./gradlew runUpgradeContractClient -Paddress='localhost:10003'

Now you can await the purchase of your assets by Smurfies!
```