National College of Ireland

Configuration Manual

MSc Research Project
FinTech

Sean Khye Lee
Student ID: x19116888

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sean Khye Lee
Student ID:	X19116888
Programme:	MSc in Fintech **Year:** 2020
Module:	Research Project
Lecturer:	Victor Del Rosal
Submission Due Date:	17 August 2020
Project Title:	Analysing the Evolution of Permissioned Blockchain in Financial Sector
Word Count:	1775 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sean Khye Lee
Student ID: x19116888

1 Introduction

This configuration manual was used to describe the technical requirements as well as the code execution when developing the application and testing the experiments. Steps were arranged accordingly to allow the research can be replicated for the project titled "Analysing the Evolution of Permissioned Blockchain in Financial Sector".

2 Hardware Requirements

- ★ Windows 10 Core Single Language X64bit
- ★ RAM: 12GB
- ★ Processor: Intel(R) Core (TM) i5-4200 CPU @ 1.60GHz
- ★ Hard Disk Drive: 1TB

3 Software Requirements

- ★ Sublime Text 3 - This was used to edit the code
- ★ Oracle Virtual Machine - This was used as the environment to deploy smart contract
- ★ Command-line interface (CLI) - This was used to execute the operating system function
- ★ Hyperledger Composer - This was used as tool to deploy the smart contract
- ★ Angular - This was connected REST API to the front-end
- ★ Apache JMeter - This was used to test the scalability
- ★ GitHub - This was used to store the coding
- ★ Microsoft Word 2016 - This was used to write the report
- ★ Microsoft Excel 2016 - This was used to create and modify the table for evaluation
- ★ Tableau Public - This was used to visualise the .csv file to get data insight

4 Developing Multon Decentralised Application

4.1 Steps to integrate blockchain application with Hyperledger fabric

Step 1: Clone File from GitHub

```
$ git clone https://github.com/Sean5678/Multon-Dapp.git
```

Step 2: Starting fabric

Change directory into fabric-dev-servers by executing the following in the terminal

```
$ cd /home/hyperledger/fabric-dev-servers
```

Start fabric

```
$ ./startFabric.sh
```

Create credentials

```
$ ./createPeerAdminCard.sh
```

Step 3: Setting up our business network

To use the yo tool, execute the following commands in the terminal.

```
$ cd ~/
$ yo hyperledger-composer:businessnetwork
```

Set up the network as follows:

```
Business network name: multon
Description: BankApp
Author name: <Your name>
Author email: <Your email>
License: Apache-2.0
Namespace: multon.bc.bank
Do you want to generate an empty template network?: No
Files will now be available in multon
```

Step 4: Defining the Business Network

Update the model file(.cto), access control file(.acl), and logic file(.js) to define assets, participants and transactions and save the changes.

Defining the .cto file

```
$ cd multon
$ ls
$ cd models
$ more multon.bc.bank.cto
```

Account is an asset which is uniquely identified with accountId. Each account is linked with Customer who is the owner of the account. Account has a property of balance which indicates how much money the account holds at any moment. Customer is a participant which is uniquely identified with customerId. Each Customer has firstName and lastName. AccountTransfer is a transaction that can occur to and from an Account. And how much money is to be transferred is stored in amount.

Defining the .js file

```
$ cd ..
$ cd lib
$ more logic.js
```

Defining the .acl file

```
$ cd ..
$ more permissions.acl
```

Step 5: Generating the Business Network Archive file (.bna)

```
$ cd /home/hyperledger/multon
$ composer archive create -t dir -n .
```

Step 6: Deploying Business Network

Here we will use PeerAdmin business network card which is already created during the installation step of Hyperledger fabric, to deploy the business network. PeerAdmin card is used to install and instantiate chaincodes on peers of fabric network.

When we deploy a business network to the Hyperledger Fabric, we will install the business network Hyperledger Composer. Then, we will start the business network for building the new participant, identity and related card as a network administrator. Eventually, the business network card must be imported into the network administrator for use, and the network can then be pinged to check its response.

To install the business network, from the multon directory, run the following command:

```
$ composer network install --card PeerAdmin@hlfv1 --archiveFile multon@0.0.1.bna
```

To start the business network, run the following command:

```
$ composer network start --networkName multon --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card
```

To import the network administrator identity as a usable business network card, run the following command:

```
$ composer card import --file networkadmin.card
```

To check that the business network has been deployed successfully, run the following command to ping the network:

```
$ composer network ping --card admin@multon
```

After these steps, business network has been deployed and integrated with Hyperledger fabric. Now, we can define the REST API for our business network as well as create an application for the same.

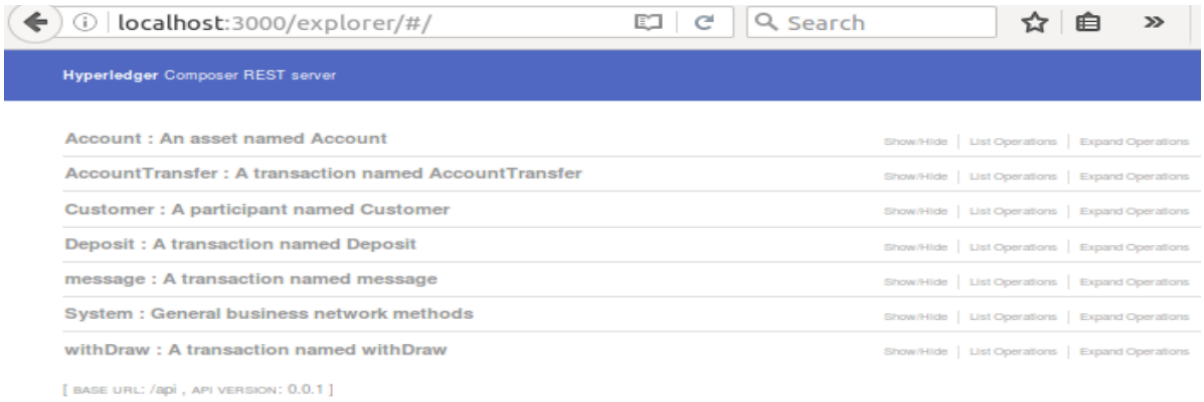
Step 7: Generate REST Server

To generate REST API, run the following command from multon

```
$ composer-rest-server
Choose the following options/Enter the following data when prompted.
Enter the name of the business network card to use: admin@multon
Specify if you want namespaces in the generated REST API: never use namespaces
Specify if you want to use an API key to secure the REST API: N
Specify if you want to enable authentication for the REST API using passport: N
Specify if you want to enable event publication over WebSockets: Y
Specify if you want to enable TLS security for the REST API: N
The REST server will now be installed.
Do not close this terminal window.
```

Now, go to our browser and open up http://localhost/3000

We should see something like this:



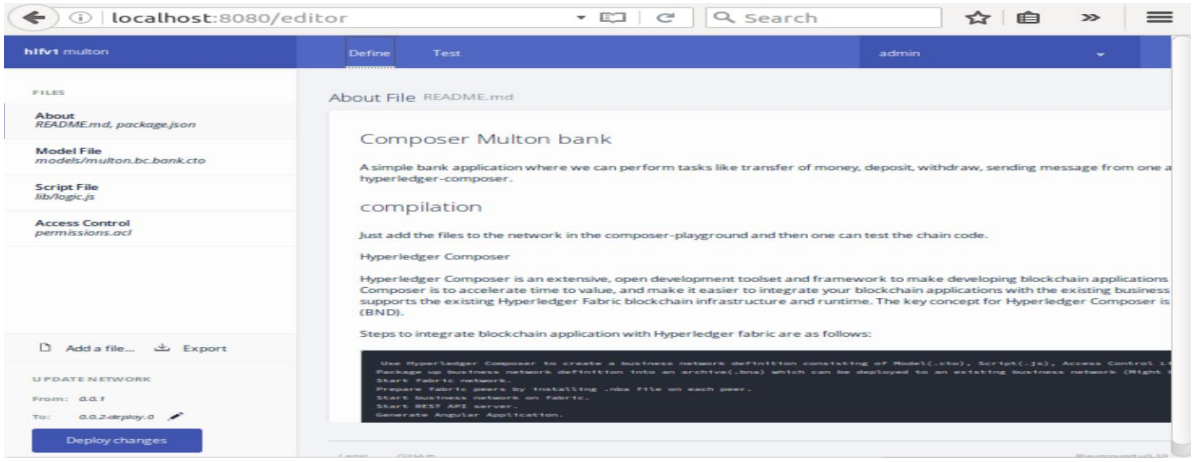
Step 8: Test the application using Composer Playground

Now that we have deployed fabric on top of a business network, and started a REST server, let us take a look at the Composer Playground.

Open another terminal window and execute

```
$ composer-playground
```

A browser window with the address http://localhost/8080 will open as below



We can now go into our project and check our model files(.cto), permission file(.acl), and script(.js). Playground offers a very good web interface so we can edit our code. We can now go into admin and see all our business networks and connect to them.

Step 9: Creating an Angular App to interact with the network

We are going to use the yo tool again.

In a new terminal window, execute

```
$ yo hyperledger-composer:angular
```

Select the following options.

Do you want to connect to a running Business Network? Y

Project name: Angular-Multon-Bankdapp

Description: Hyperledger Composer Angular project

Author name: <Your name>

Author email: <Your email>

License: Apache-2.0

Name of the Business Network card: admin@multon
Do you want to generate a new REST API or connect to an existing REST API?: Connect
to an existing REST API
REST server address: http://localhost
REST server port: 3000
Should namespaces be used in the generated REST API?: Namespaces are not used

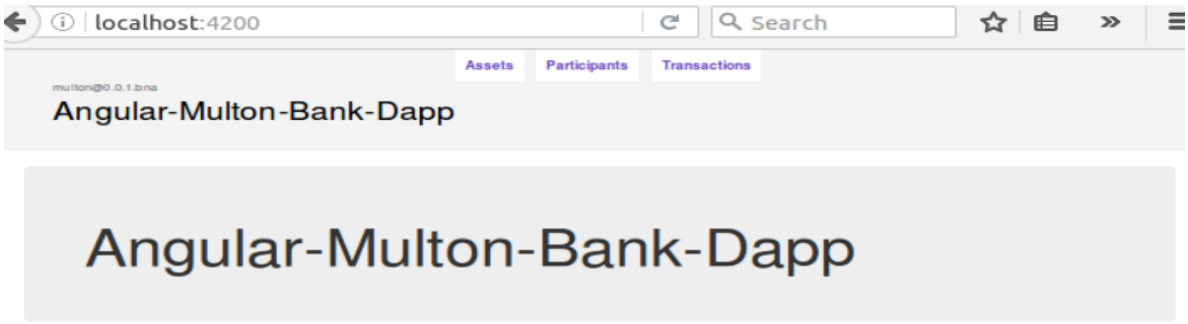
Go to local directory:

$ cd /home/hyperledger/multon/Angular-Multon-Bankdapp

Start the server by executing

$ npm start

Go to localhost:4200. We should see something like this.



Addressing the Transaction Problem in Angular for Hyperledger Blockchain Application

After starting the server, navigate the multon/ Angular-Multon-Bankdapp /src/app and replace the coding in the AccountTransfer, Deposit, withDraw, and message folder for the html file using the Sublime Text editor.

First: Fix the invoke button

Go to the last line of html file. Then, describe the click function and add the data-target as below:

```
        </ul>
      </div>
    </div>
        <button (click)="resetForm()" type="button" class="btn btn-primary
invokeTransactionBtn" data-toggle="modal" data-
target="#addTransactionModal">Invoke</button>
  </div>
 </div>
</div>
```

Second: Remove the additional input like transactionId and timestamp

Find the similar code below in html file and replace the code as below:

```
          <!--<div class="form-group text-left">
            <label for="transactionId">transactionId</label>

              <input formControlName="transactionId" type="text" class="form-control">
```

```
</div>

<div class="form-group text-left">
  <label for="timestamp">timestamp</label>

    <input formControlName="timestamp" type="text" class="form-control">

</div> -->
```

*For repetition from step 1 to step 9
composer card delete --card admin@multon

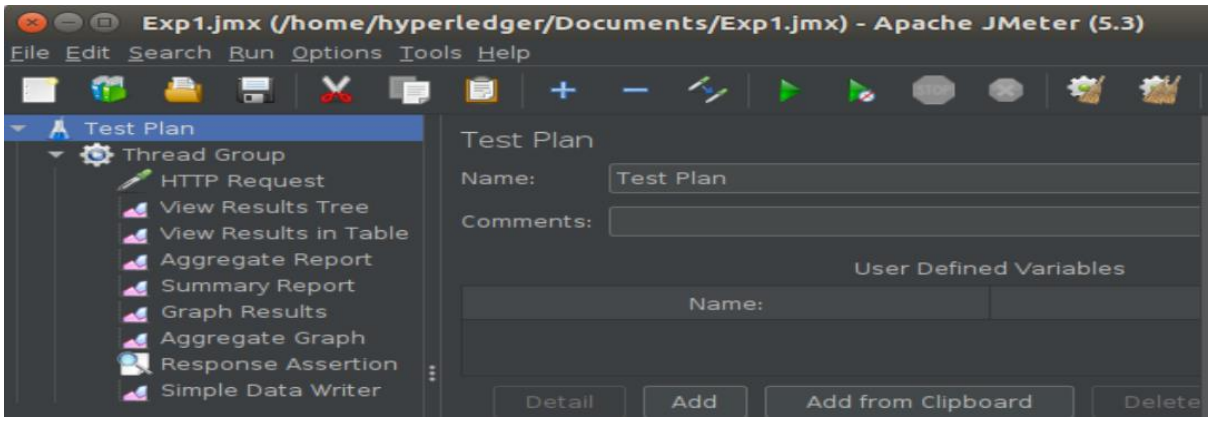
5 Result and Evaluation

5.1 Apache JMeter (Performance for Scalability)

5.1.1 Open the JMeter

```
$ cd apache-jmeter-5.3/bin/
$ ./jmeter
```

Then, the JMeter will open as below:



5.1.2 To run the Experiments in Ubuntu. Use below command:

Move to the directory:

```
$ cd apache-jmeter-5.3
$ cd bin
```

Experiment 1

Before running the code snippet, Exp1.jmx file is saved with thread properties as below:
Number of users: 5
Ramp-up period (seconds): 1
Loop count: 1

```
$ ./jmeter.sh -n -t /home/hyperledger/Documents/Exp1.jmx -l
/home/hyperledger/Documents/Exp1.csv
```

Experiment 2

Before running the code snippet, Exp2.jmx file is saved with thread properties as below:
Number of users: 50
Ramp-up period (seconds): 10
Loop count: 1

```
$ ./jmeter.sh -n -t /home/hyperledger/Documents/Exp2.jmx -l
/home/hyperledger/Documents/Exp2.csv
```

Experiment 3

Before running the code snippet, Exp3.jmx file is saved with thread properties as below:
Number of users: 500
Ramp-up period (seconds): 100
Loop count: 1

```
$ ./jmeter.sh -n -t /home/hyperledger/Documents/Exp3.jmx -l
/home/hyperledger/Documents/Exp3.csv
```

Experiment 4

Before running the code snippet, Exp4.jmx file is saved with thread properties as below:
Number of users: 5000
Ramp-up period (seconds): 1000
Loop count: 1

```
$ ./jmeter.sh -n -t /home/hyperledger/Documents/Exp4.jmx -l
/home/hyperledger/Documents/Exp4.csv
```

Experiment 5

Before running the code snippet, Exp5.jmx file is saved with thread properties as below:
Number of users: 6000
Ramp-up period (seconds): 1200
Loop count: 1

```
$ ./jmeter.sh -n -t /home/hyperledger/Documents/Exp5.jmx -l
/home/hyperledger/Documents/Exp5.csv
```

5.2 Tableau

Finally, Tableau will be used to visualise the .csv file for 5 experiments to get the average response time vs the number of users as shown below.

References

https://github.com/

https://www.hyperledger.org/use/composer

https://www.oracle.com/ie/virtualization/technologies/vm/downloads/virtualbox-downloads.html

https://angular.io/

https://jmeter.apache.org/download_jmeter.cgi

https://public.tableau.com/en-us/s/