Blockchain on cloud

workshop guide

Table of Contents

[Section 1 - Building a Development Environment 3](#_Toc1137662)

[1.1 install Prerequisites for Hyperledger fabric and composer 3](#_Toc1137663)

[1.2 Install visual studio code 6](#_Toc1137664)

[1.3 Install Hyperledger Composer tool set 7](#_Toc1137665)

[Section 2 – Hands on with Hyperledger 8](#_Toc1137666)

[2.1 Hyperledger Composer Playground 8](#_Toc1137667)

[2.2 Solution packaging and deployment on Hyperledger Fabric 24](#_Toc1137668)

[2.2.1 Export a BNA file and copy that to a solution folder 24](#_Toc1137669)

[2.2.2 Deploying the bna file to fabric/Composer environment 26](#_Toc1137670)

[2.2.3 Starting the business network and testing deployment 27](#_Toc1137671)

[2.2.4 Generate RESTful APIs from the deployed business network 28](#_Toc1137672)

[2.2.5 Generate an angular application 30](#_Toc1137673)

[Section 3 – Referances and links 33](#_Toc1137674)

# Section 1 - Building a Development Environment

## 1.1 install Prerequisites for Hyperledger fabric and composer

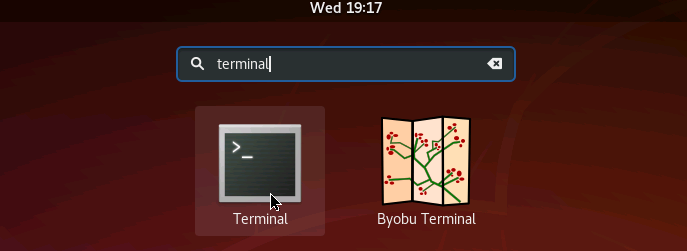
In this section, you will be installing required prerequisites for Hyperledger Fabric v1.2 and Hyperledger composer/playground.

* + 1. Click on “Activities” located at the top left hand corner.



Pic 1.1.1 – Activities menu

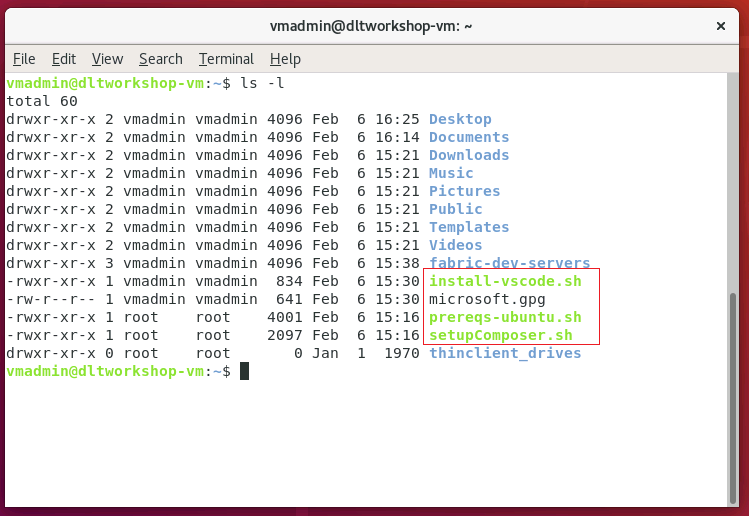
* + 1. Type “terminal” in the search bar and you will see two applications as shown below. Click on “Terminal”.



Pic 1.1.2 – Opening Terminal

* + 1. When you open the terminal app, please type the following command. A list of folders and files located should be displayed in the home folder.

ls -l

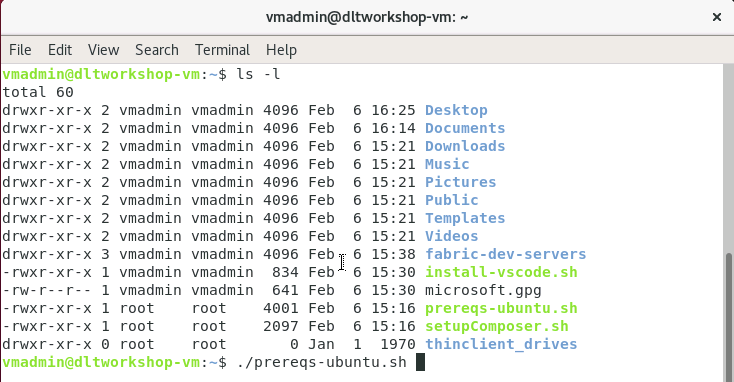


Pic 1.1.3 – Terminal window that list all the prerequisite install scripts

* + 1. Using the “prereqs-ubuntu.sh” script to install the following dependencies need for Hyperledger Fabric and Composer tool set:
* Git
* npm
* NodeJS
* Docker-ce
* Docker-Compose
* Python
* Unzip

Run the following command to execute “prereqs-ubuntu.sh” script.

./prereqs-ubuntu.sh



Pic 1.1.4 – Terminal window with command to run ./prereqs-ubuntu.sh script

Let the script “prereqs-ubuntu.sh run” and wait for the following message: “Please logout then login before continuing“

When you see this message you can execute the following command or manually reboot your VM.

sudo reboot now

## INstall visual studio code

Here we will using a script to install Visual Studio Code editor. You can find more information about this edit at <https://code.visualstudio.com/>

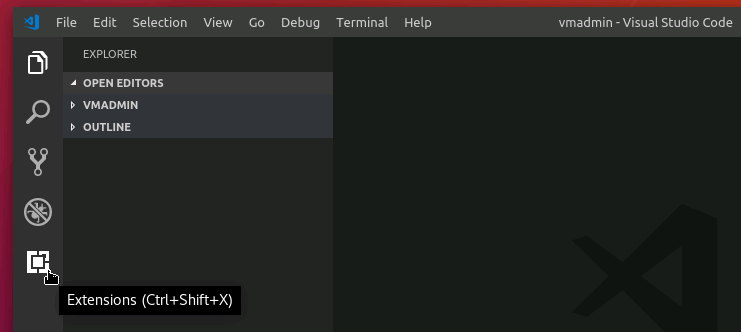
Run the following command and execute “install-vscode.sh” script.

./install-vscode.sh

Once the installation is complete, launch the code editor. Type “code” from the terminal window to open the app.

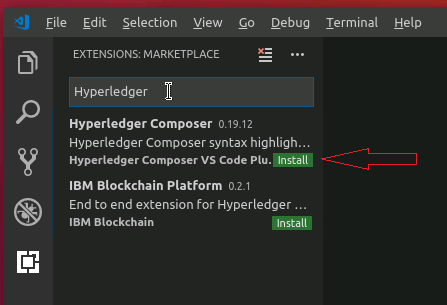
code

Once the app is running, click on the Extensions button in the top left hand corner.



Picture 1.2.1 – Code editor

Type “Hyperledger” in the market place search box. You should see Hyperledger composer as one of the results. Click on “install” to install the composer extension.



Picture 1.2.2 – Market place – search for Hyperledger composer extension

## 1.3 Install Hyperledger Composer tool set

Now that all of the prerequisites are installed, install the Composer tool sets and Fabric.

Run “setupComposer.sh” script found under the home directory.

./setupComposer.sh

This script will install the following components:

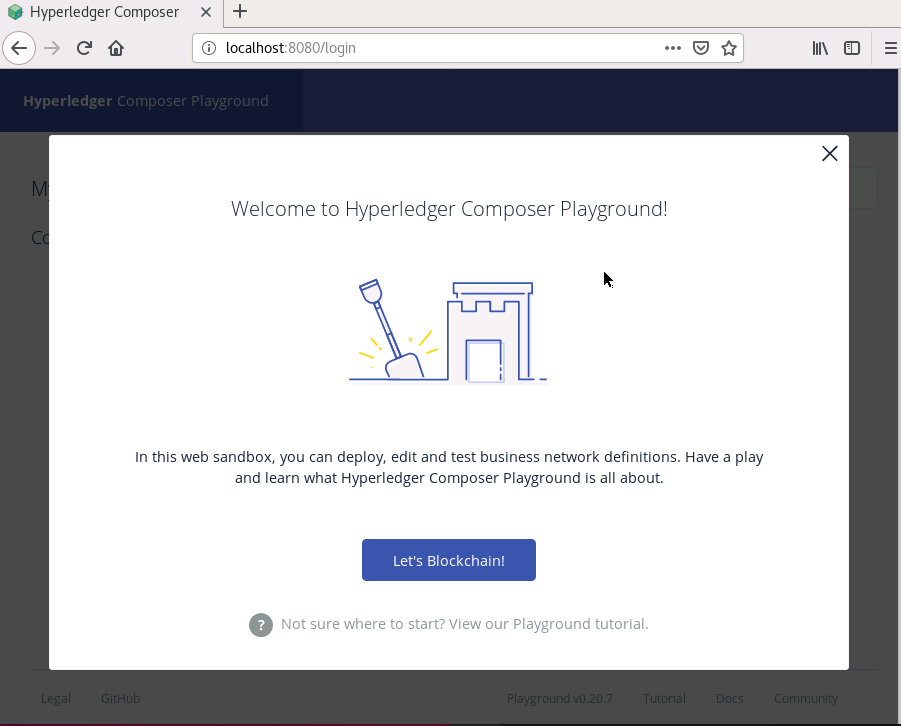
* Note: there will be WARN messages which you can safely ignore.
* Composer-cli – command line interface to Composer.
* Composer-rest-server - Utility for running a REST Server on your machine to expose your business networks as RESTful APIs.
* Generator-hyperledger-composer - Useful utility for generating application assets.
* Yo - Yeoman is a tool for generating applications, which utilises generator-hyperledger-composer.
* Composer-playground – web application that enables testing of Blockchain models.
* Deploy Fabric framework and starts a sample single organization based network.
* Creates a Peer admin card.
* Runs composer-playground web application.

# Section 2 – Hands on with Hyperledger

In this section, we will be exploring components of Hyperledger Composer using a simple Blockchain business model.

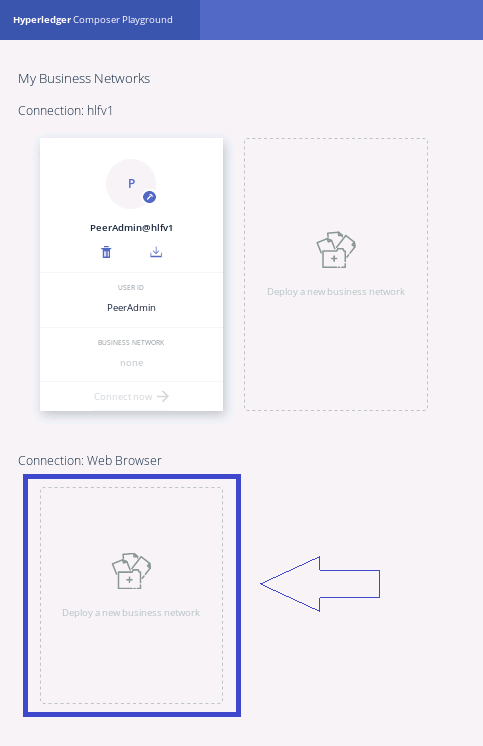
## 2.1 Hyperledger COmposer Playground

After step 1.3, a web based Playground IDE will be launched as shown below. Click on the “Let’s Blockchain!” button to view the main page.



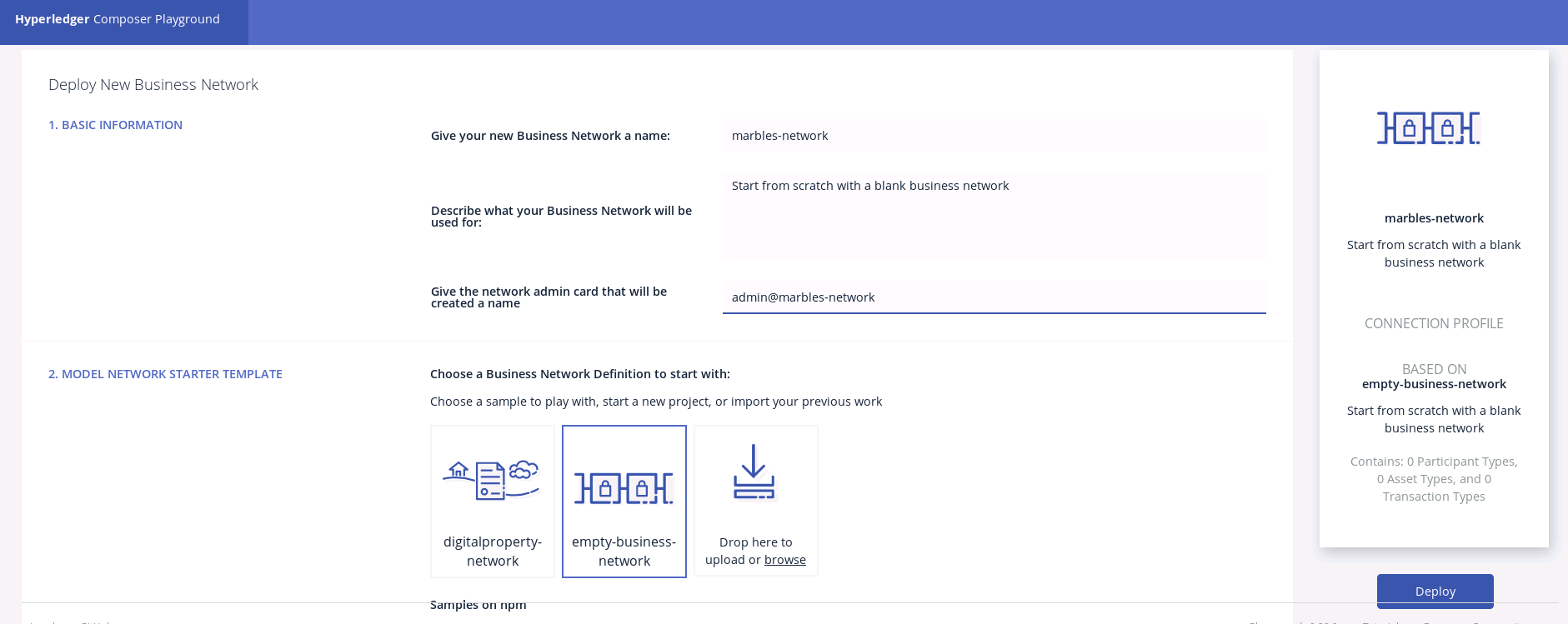
Picture 2.1.1 – Hyperledger composer playground welcome page.

The Business Network page is the default playground loading page. Here you will see two connection options as shown in picture 2.1.2. Connection “hlfv1” points to the actual Hyperledger Fabric network and “web browser“ is a simulation environment that is hosted on the browser cache.



Picture 2.1.2 – Hyperledger composer playground Business Network page.

At this point, we will be using the “Web Browser” connection to explore how a Blockchain application can be developed and tested. Click on the “Deploy a new business network” button which will take you to a page where you can select a sample network and deploy a simulation environment. (See Picture 2.1.3)

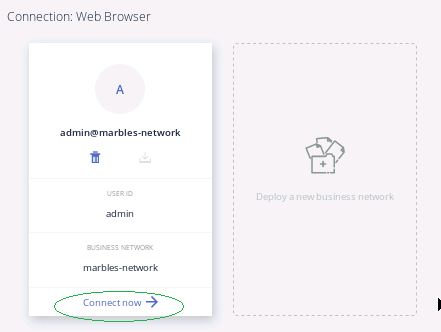


Picture 2.1.3 – Playground select and deploy a sample application

Select “empty-business network” from section 2 on the “Deploy New Business Network” page.

In Section 1 give the Business Network the name “marbles-network”, and give the admin card that will be created the name admin@marbles-network.

Once this is done click on “Deploy button”. Once the deployment is done, you will see a connection option under Web Browser section (see picture 2.1.4).



Picture 2.1.4 – marbles-network ready for deployment

Highlight the admin name by double clicking and then use the TAB key to go to “Connect now”. Once Connect Now is highlighted press enter and it will deploy the sample network in a simulation environment. Once the deployment is done, it will change the view to a web playground view (see picture 2.1.5) where you can edit/test the application.

We will now define the network.

Click on the Activities link in the task bar and go to Files (filing cabinet icon). Open the home directory .

Double Click on README.md and when the file opens copy it to the clipboard.

In your browser click on About, click on the <> in the top right and paste the content into the file.

Click on the eye to save the file.

In the Home directory open marbles.cto and copy the contents to the clipboard.

Click on Model file in the browser and paste the content into the Model file.

Click on the pencil beside the name of the file and change it to “marbles”.

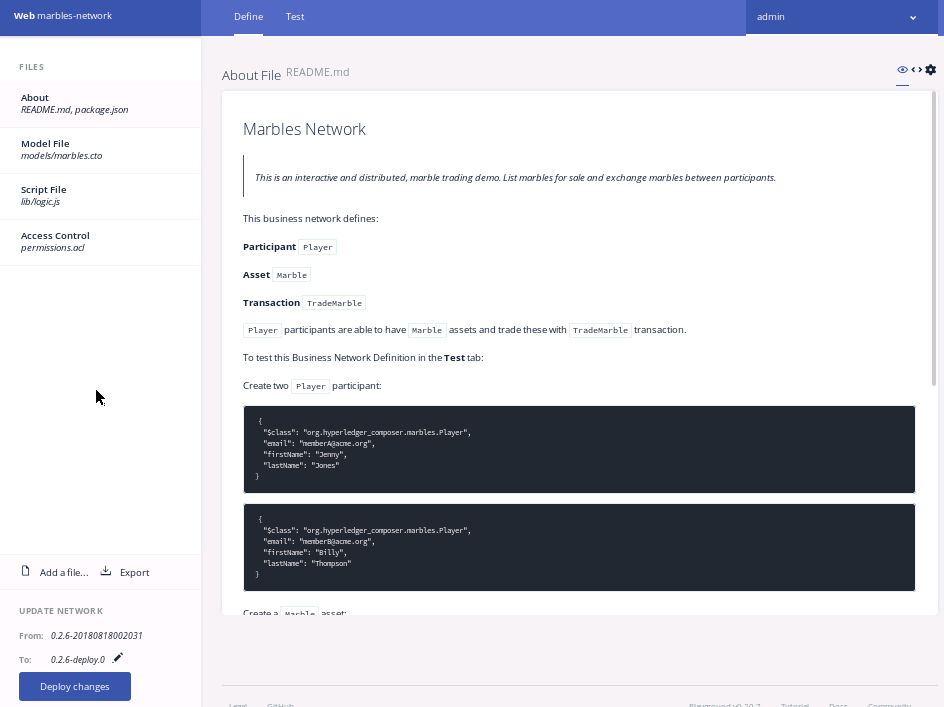
In the Home directory open permissions.acl and copy the contents to the clipboard.

Click on access control in the browser, remove any contents and paste the access control file in.

In the Home directory right click on the logic.js file and copy its contents to the clipboard.

Click on Add a file and select script file. Delete any contents and paste in the contents of the logic.js file. Rename the file from script to logic, by clicking on the pencil beside the name and making it ‘logic’.

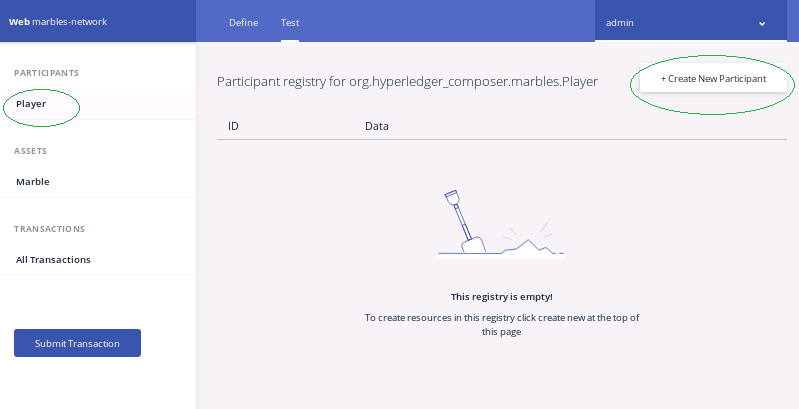
Once this is done click on deploy changes.



Picture 2.1.5 – Playground editor view

Take a few moment to read the “About File” (Readme.md) description file to understand the solution and how it works.

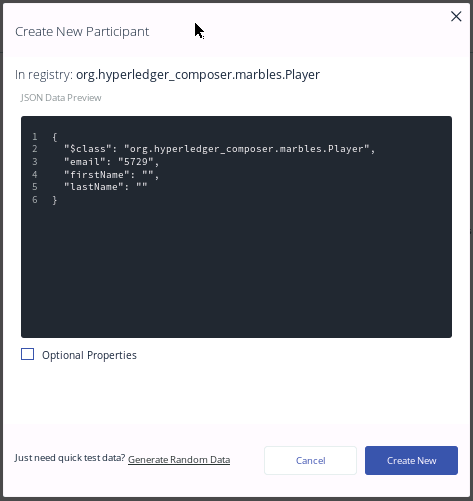
Let’s get hands-on and see how this sample works. Start by clicking on the “Test” tab at the top of the screen.



Picture 2.1.7 – Sample application testing page

Currently the application doesn’t have any participants or Assets defined. Let’s start by creating two participants and an Asset.

Click on “+Create New Participants” and you will see a dialog as shown below (Picture 2.1.8). Enter some sample values for first name and last name and make a note of the email ID. Once you are done, click on “Create new”.



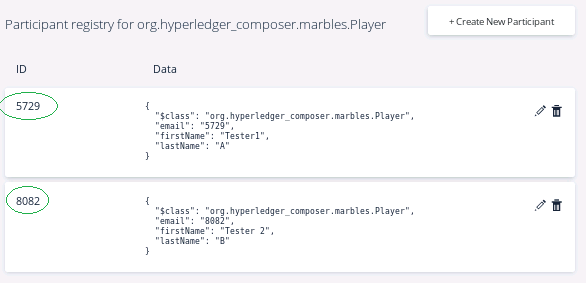
Picture 2.1.8 – Create new Participant dialog

When you are done creating a participant, an entry as shown below will be added to the Participant registry.

Picture 2.1.9 - Participant registry showing the new participant

Now repeat the same steps again to create another participant for testing purposes.

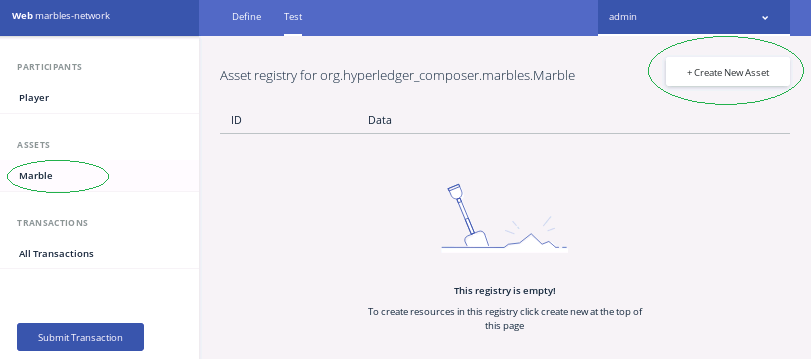
Take a note of the email IDs as we will be using these IDs in the next steps.



Picture 2.1.10 - Participant registry showing two new participants

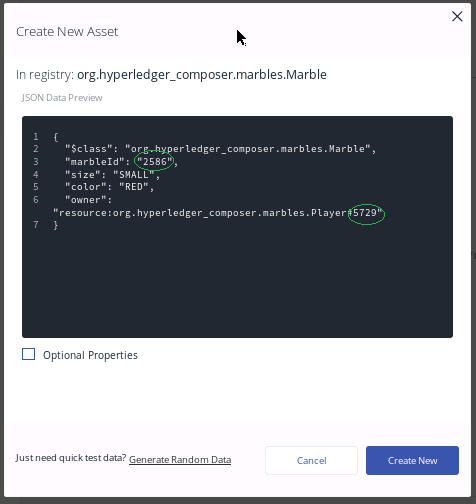
Next, we will be creating an Asset (Marble) that will be traded among the players we created before.

Click on Marble assets on the left menu and click on “+Create New Asset”.



Picture 2.1.11 - Participant registry showing two new participants

When you get “Create new Asset” dialog, please fill in the fields as needed. You can leave most of the fields as default, but change the owner field with proper player ID. You can use one of the email IDs that we used to create players in the previous steps (See picture 2.1.12).

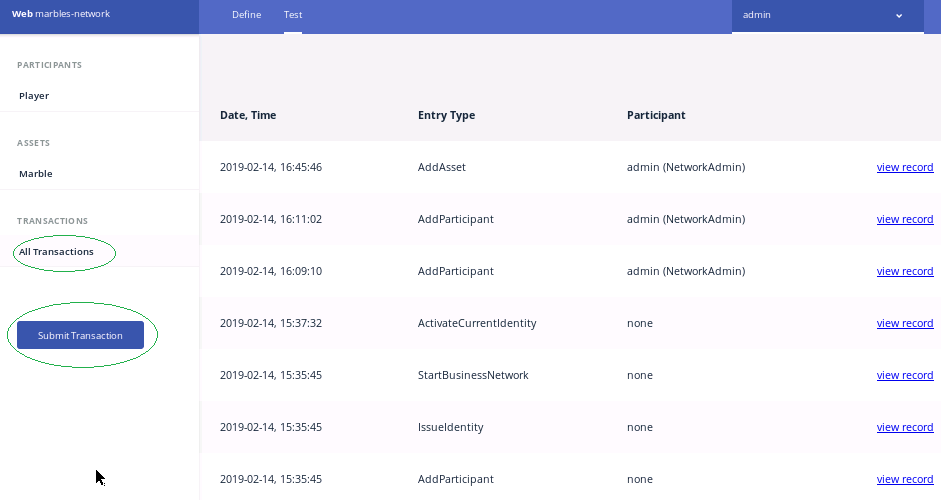


Picture 2.1.12 – Create new asset dialog with sample field values

Now that we have two players and a Marble created, we can try running a sample transaction that transfers marbles from one player to another.

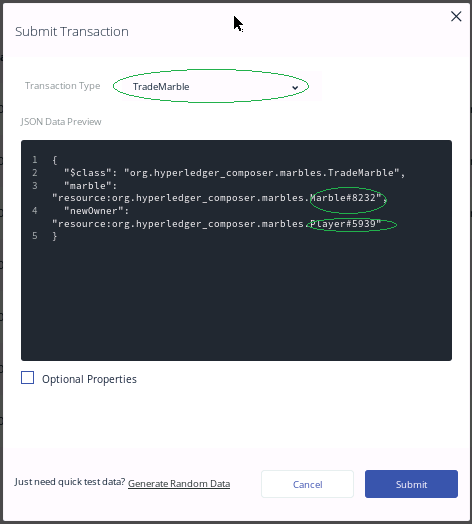
Picture 2.1.13 - Asset registry showing new asset

Click on “All Transactions” from the left menu whereyou willsee some system transactions listed. We will now run a transaction of our own that transfers marbles from one player to another.



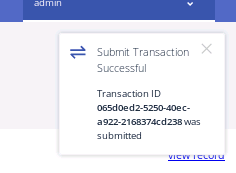
Picture 2.1.14 – All Transaction page

Click on “Submit Transaction” and you will see a dialog as shown below.



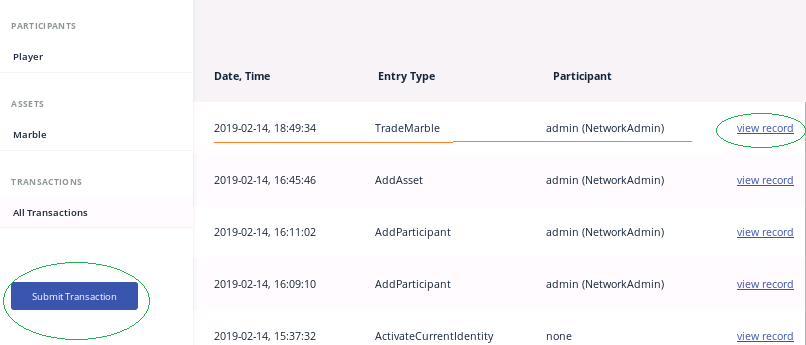
Picture 2.1.15 – Submit Transaction dialog

On the Submit Transaction dialog, make sure to change the marble ID and player ID to match previous steps. Once you are done making changes, click on Submit. This initiates a transaction on Blockchain and displays a confirmation dialog as shown below.

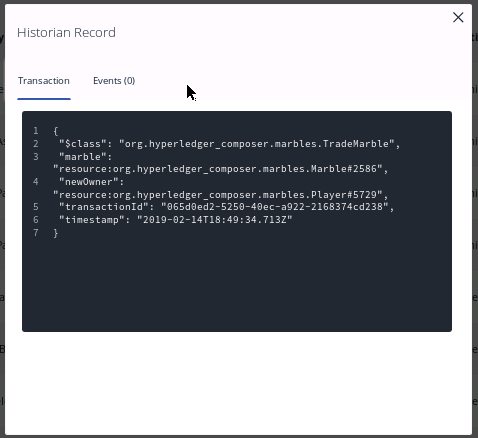


Picture 2.1.15 – Submit transaction dialog with auto generated Transaction ID.

Once the transaction is submitted successfully, you will see an entry created under “All Transaction” page (see picture 2.1.16). You can see details of that transaction by clicking on the “view record” link.



Picture 2.1.16 – New transaction listed on the “All transaction” page.



Picture 2.1.17 – Transaction details dialog

You can also verify the marble asset to see if the owner details are properly updated (see picture 2.1.18).

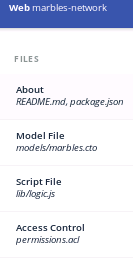


Picture 2.1.18 – Transaction details dialog

Now that we have a bit of understanding of how this application functions, let’s have a quick look at how this application is programmed.

We will start by clicking on the “Define” option in Composer playground.

In this view, you should see four different files (See picture 2.1.19).

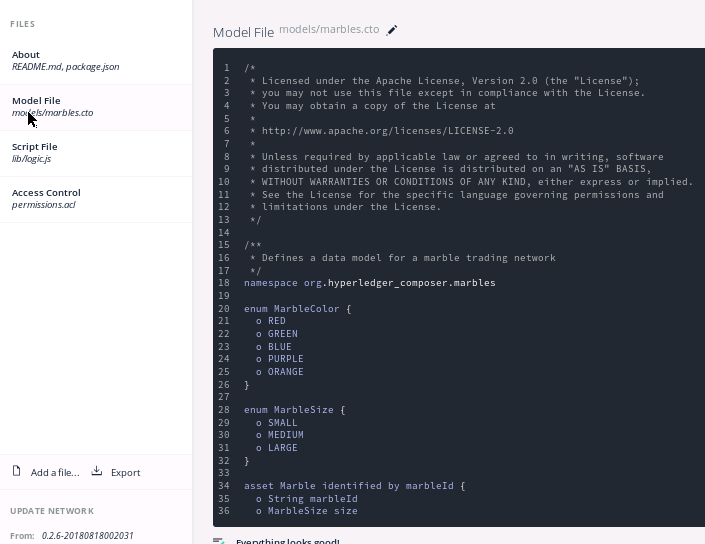


Picture 2.1.19 – Solution definition view

* The "About" file contains information about the solution and how it works. This is the file you reviewed after deploying the Marbles network solution template.
* The "Model" file (.cto file) defines the properties of the Assets, Participants, and Transactions that make up this solution.
* The "Script" file (.js file) contains the code that gets executed for each transaction.
* The "Access Control" file (.acl file) contains the permissions settings for this solution.

**Let’s look at the model file for review:**

Click on “Model file” menu item and we will take a look at how assets, participants, and transactions are defined.



Picture 2.1.20 – Model file

|  |
| --- |
| */\*\**  *\* Defines a data model for a marble trading network*  *\*/*  *namespace org.hyperledger\_composer.marbles*  *enum MarbleColor {*  *o RED*  *o GREEN*  *o BLUE*  *o PURPLE*  *o ORANGE*  *}*  *enum MarbleSize {*  *o SMALL*  *o MEDIUM*  *o LARGE*  *}*  *asset Marble identified by marbleId {*  *o String marbleId*  *o MarbleSize size*  *o MarbleColor color*  *--> Player owner*  *}*  *participant Player identified by email {*  *o String email*  *o String firstName*  *o String lastName*  *}*  *transaction TradeMarble {*  *--> Marble marble*  *--> Player newOwner*  *}* |

Picture 2.1.21 – content of the model file (marbles.cto)

Here we have two enum types defining colors and sizes of Marble asset types. Participant is also a simple object definition that uses email as the main identifier.

If you look at the Marble object, you will see a reference to owner object that is marked with “🡪”. This is a unidirectional reference and it points to tuple that consists of the following elements: (in our example we had: *org.hyperledger\_composer.marbles.Player#5729*)

* The namespace of the type that is being referenced - *“org.hyperledger\_composer.marbles*”
* The type name of the type being referenced - *Player*
* The identifier of the instance being referenced – *5729*

Transaction definition also has references to Marble and Player.Note that the ID for the transaction object is auto-generated by the system. You can get more details about this modeling language from the official site: <https://hyperledger.github.io/composer/latest/reference/cto_language.html>

**Let’s look at the Script file for review:**

Click on the “Script file” menu item.

|  |
| --- |
| /\* global getAssetRegistry \*/  /\*\*  \* Trade a marble to a new player  \* @param {org.hyperledger\_composer.marbles.TradeMarble} tradeMarble - the trade marble transaction  \* @transaction  \*/  async function tradeMarble(tradeMarble) {  tradeMarble.marble.owner = tradeMarble.newOwner;  const assetRegistry = await getAssetRegistry('org.hyperledger\_composer.marbles.Marble');  await assetRegistry.update(tradeMarble.marble);  } |

Picture 2.1.22 – content of the logic.js script file

Logic.js file contains code that implements each of the transactions defined in the model file. Each transaction function takes one input parameter which is passed in by Hyperledger. This parameter represents the transaction object as defined in the model file. In our example, TradeMarble is the parameter getting passed into tradeMarble function.

TradeMarble function makes changes to the marble asset and updates the assets registry and asset changes. You can get more details about transaction processing from the official composer reference guide: <https://hyperledger.github.io/composer/latest/reference/js_scripts>

**Let’s look at the Permissions file for review:**

Click on the “Access Control” (permissions.acl) from the left menu.

|  |
| --- |
| /\*\*  \* Sample access control list.  \*/  rule Default {  description: "Allow all participants access to all resources"  participant: "ANY"  operation: ALL  resource: "org.hyperledger\_composer.marbles.\*"  action: ALLOW  }  rule SystemACL {  description: "System ACL to permit all access"  participant: "org.hyperledger.composer.system.Participant"  operation: ALL  resource: "org.hyperledger.composer.system.\*\*"  action: ALLOW  }  rule NetworkAdminUser {  description: "Grant business network administrators full access to user resources"  participant: "org.hyperledger.composer.system.NetworkAdmin"  operation: ALL  resource: "\*\*"  action: ALLOW  }  rule NetworkAdminSystem {  description: "Grant business network administrators full access to system resources"  participant: "org.hyperledger.composer.system.NetworkAdmin"  operation: ALL  resource: "org.hyperledger.composer.system.\*\*"  action: ALLOW  } |

Picture 2.1.23 – content of the permissions.acl file

* The default rule allows any participant in the solution the ability to perform all operations on anything in the namespace. “org.hyperledger\_composer.marbles.\*”
* SystemACL allows composer system itself to access all resources with the “org.hyperledger.composer.system.\*”
* NetworkAdminUser rule allows all network admins to have full access to all of the solution components.
* NetworkAdminSystem rule allows network admins to have full access to system level resources.

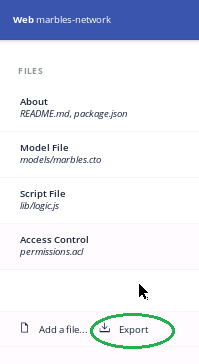
You can find more details about permission files at the following official documentation: <https://hyperledger.github.io/composer/v0.19/reference/acl_language>

## 2.2 Solution packaging and deployment on Hyperledger Fabric

In this section, we will be going through the steps that packages a solution from Composer playground and deploys it into a Hyperledger Fabric / Composer environment.

### 2.2.1 Export a BNA file and copy that to a solution folder

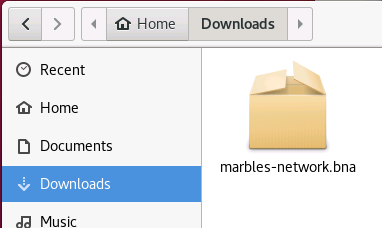
From Composer playground view, select “define tab” at the top of the window. Click on “Export” (See picture 2.2.1.1)



Picture 2.2.1.1 – Define tab with Export button highlighted.

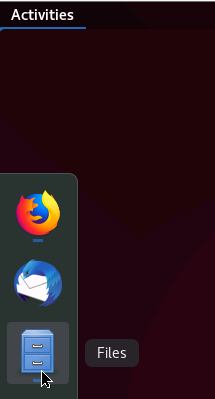
When you click on the export button, you will see a file with “.bna” extension getting downloaded to your local hard drive. This is the versioned archive file of the business solution.

We have to create a solution folder on your home folder and copy this .bna file to that directory.

  
Picture 2.2.1.2 – BNA file location after downloading.

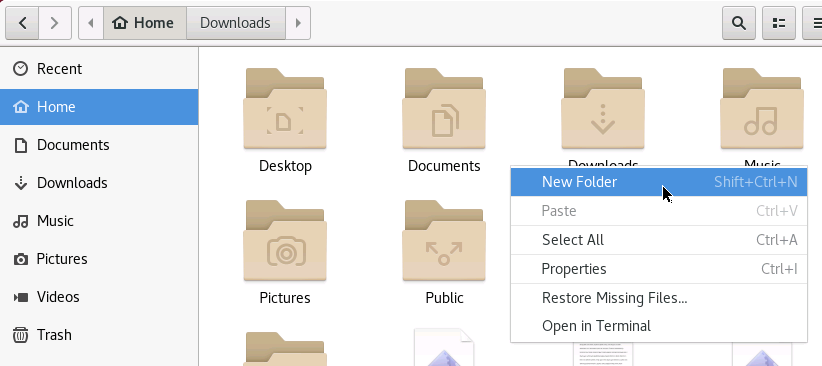
Create a folder under home folder called “marbleNetwork” and copy the .bna file there.

* From the Ubuntu desktop, select the Files tool.



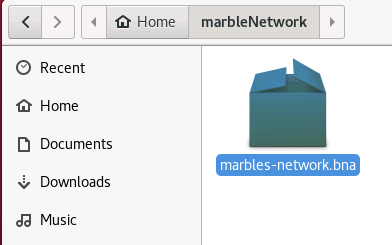
Picture 2.2.1.3 – Files tool

* Select Home folder and right click at the right side panel. When you get the menu, select “New Folder”.



Picture 2.2.1.4 – Creating a new folder under home folder

* Name the new folder “marbleNetwork” and copy the .bna file to that folder.

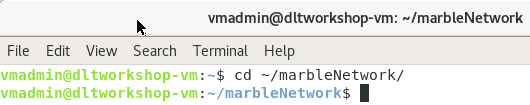


Picture 2.2.1.5 – bna file copied to the solution folder

### 2.2.2 Deploying the bna file to fabric/Composer environment

From the terminal, go to the marbleNetwork folder by running the following command:

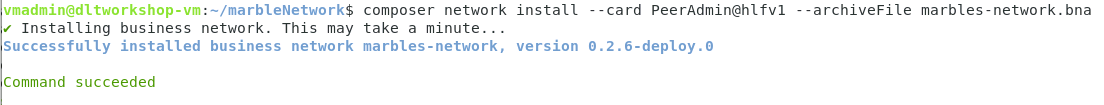
|  |
| --- |
| cd ~/marbleNetwork/ |



Picture 2.2.2.1 – Terminal - cd into marbleNetwork folder

Once you get there, run the following command to deploy the .bna file to the Fabric network.

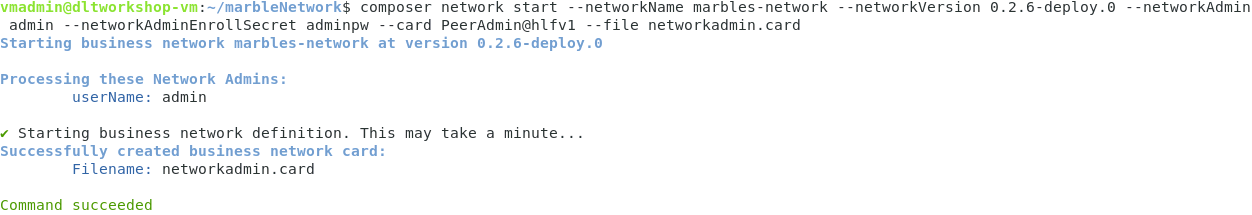
|  |
| --- |
| composer network install --card PeerAdmin@hlfv1 --archiveFile marbles-network.bna |

Picture 2.2.2.2 – Terminal – running composer network install command

### 2.2.3 Starting the business network and testing deployment

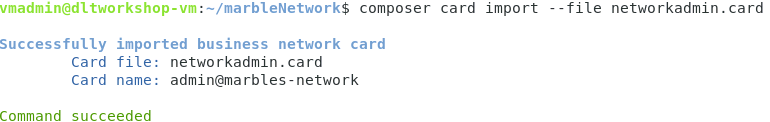
To start the solution, run the following command. It may take few minutes to deploy. Once it is done, it will generate a network admin card (networkadmin.card). We will be using this card for upcoming steps.

|  |
| --- |
| composer network start --networkName marbles-network --networkVersion 0.0.2-deploy.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card |

Picture 2.2.3.1 – Terminal – running composer network start command

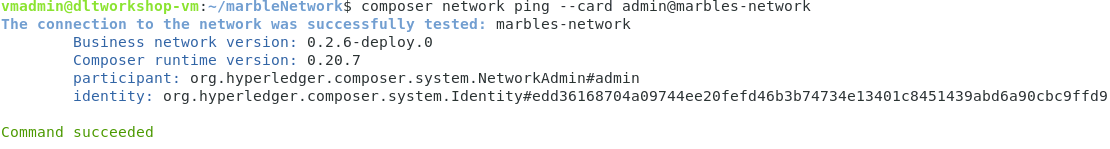
Import the networkadmin.card to Fabric/Composer environment using the following command:

|  |
| --- |
| composer card import --file networkadmin.card |

Picture 2.2.3.2 – Terminal – importing networkadmin.card to Fabric/composer environment.

As a final step, run the following command to test to see if the Business network is running:

|  |
| --- |
| composer network ping --card admin@marbles-network |

Picture 2.2.3.3 – Terminal – testing to see if the deployed business network is running

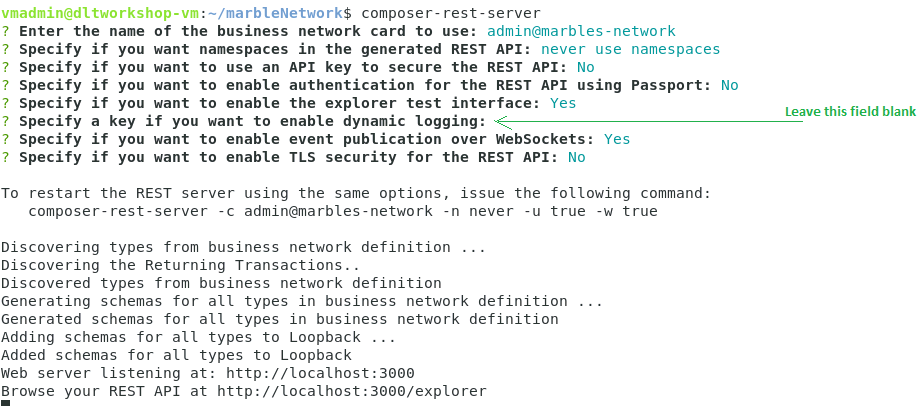
## 2.3 Generate RESTful APIs from the deployed business network

Now that we have a running business network, let’s generate RESTful APIs using composer. Theseexposed APIs can be used by client side application to interact with our business network deployed on Fabric Blockchain.

From the marble-network folder, run the following command:

|  |
| --- |
| composer-rest-server |

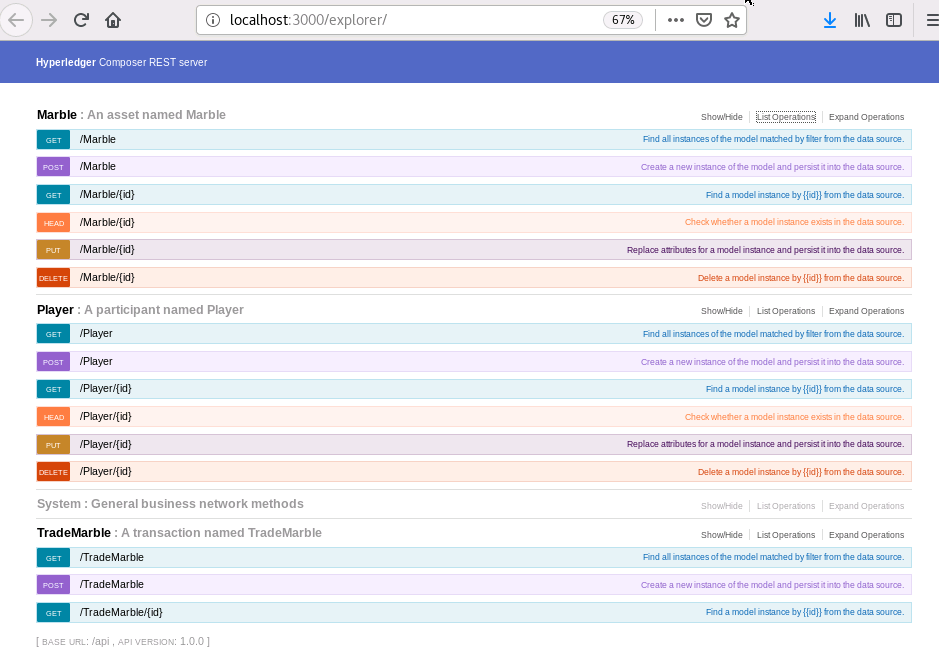
Here, you will be asked to answer series of questions. Please use the following screenshot to see the answers that are needed for the current workshop.

Picture 2.3.1 – Terminal – running composer rest server command

Once you are done inputting all the answers, you will see a confirmation message as shown in picture 2.3.1. At this point, Composer generated RESTful APIs for the business network we just deployed is running a simple test web portal at localhost:3000.

Start a browser and type in the following web address: <http://localhost:3000/explorer>

From this web application (see picture 2.2.3.4), you can test your application and see how it could be integrated with external client applications.

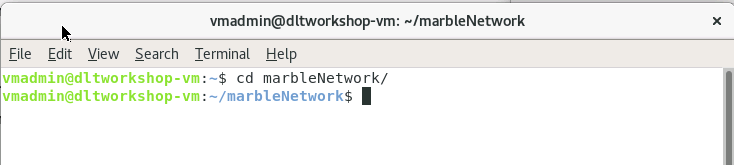


Picture 2.2.4.2 – explorer that is exposing RESTful APIs for the marbles-network.

## 2.4 Generate an angular APPLICATION

Here we will be using Yeoman to build an Angular application that uses the RESTful APIs that we just generated in the previous step.

Start a new terminal window and cd to marbles-network folder under the home folder.

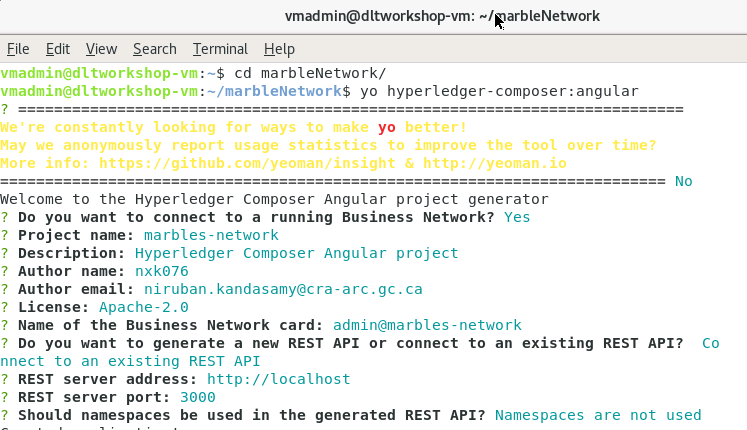


Picture 2.4.1 – terminal window – cd to marbles-network folder

From the marbles-network folder, type in the following command:

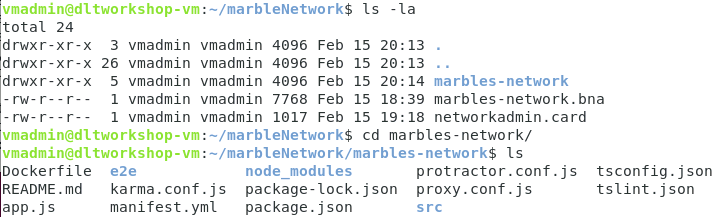
|  |
| --- |
| yo hyperledger-composer:angular |

You will have to answer few question to generate the Angular application. Use the following picture 2.4.2 as a reference.



Picture 2.4.2 – terminal window – yo command

Once you are done, press the enter key and Yeoman will start generating the application and deploying all the dependencies. When everything is done, you can see the generated application under /marbleNetwork folder. (See picture 2.4.3)

Picture 2.4.3 – terminal window – generated angular application folder.

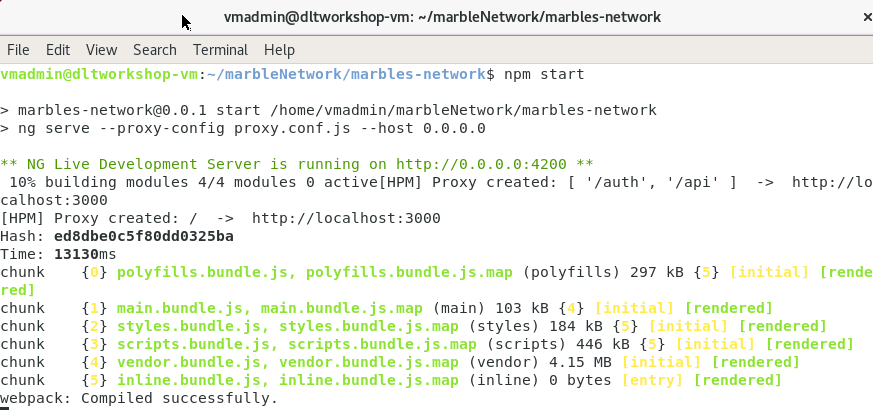
Let’s go to the generated angular application folder:

|  |
| --- |
| cd ~/marbleNetwork/marbles-network/ |

Run the Angular application usingthe following command:

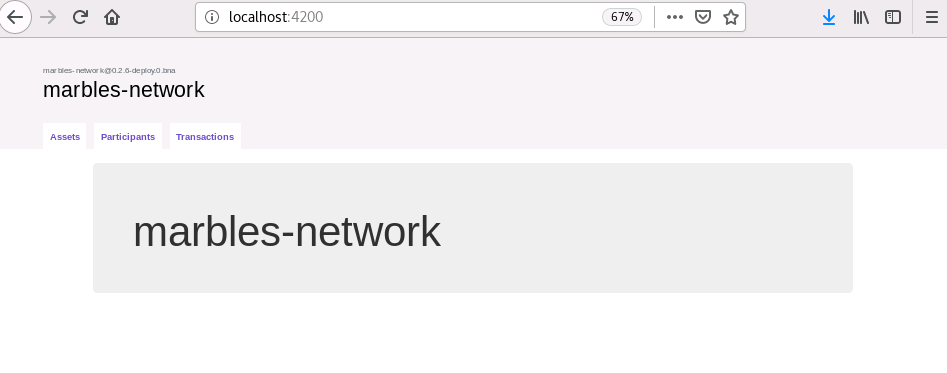
|  |
| --- |
| npm start |

Wait for the message “webpack: compiled successfully”.



Picture 2.4.4 – terminal window – generated angular application folder.

Once you get the successful message, go to <http://localhost:4200>. This is where a simple angular application is running. You can interact Fabric Blockchain using this application.

Picture 2.4.5 – Simple Angular application

# Section 3 – Referances and links

1. Hyperledger project:

<https://www.hyperledger.org/>

1. Hyperledger composer project: <https://hyperledger.github.io/composer/latest/>
2. Hyperledger Fabric:

<https://hyperledger-fabric.readthedocs.io/en/release-1.4/>

1. Composer playground – online:

<https://composer-playground.mybluemix.net/>