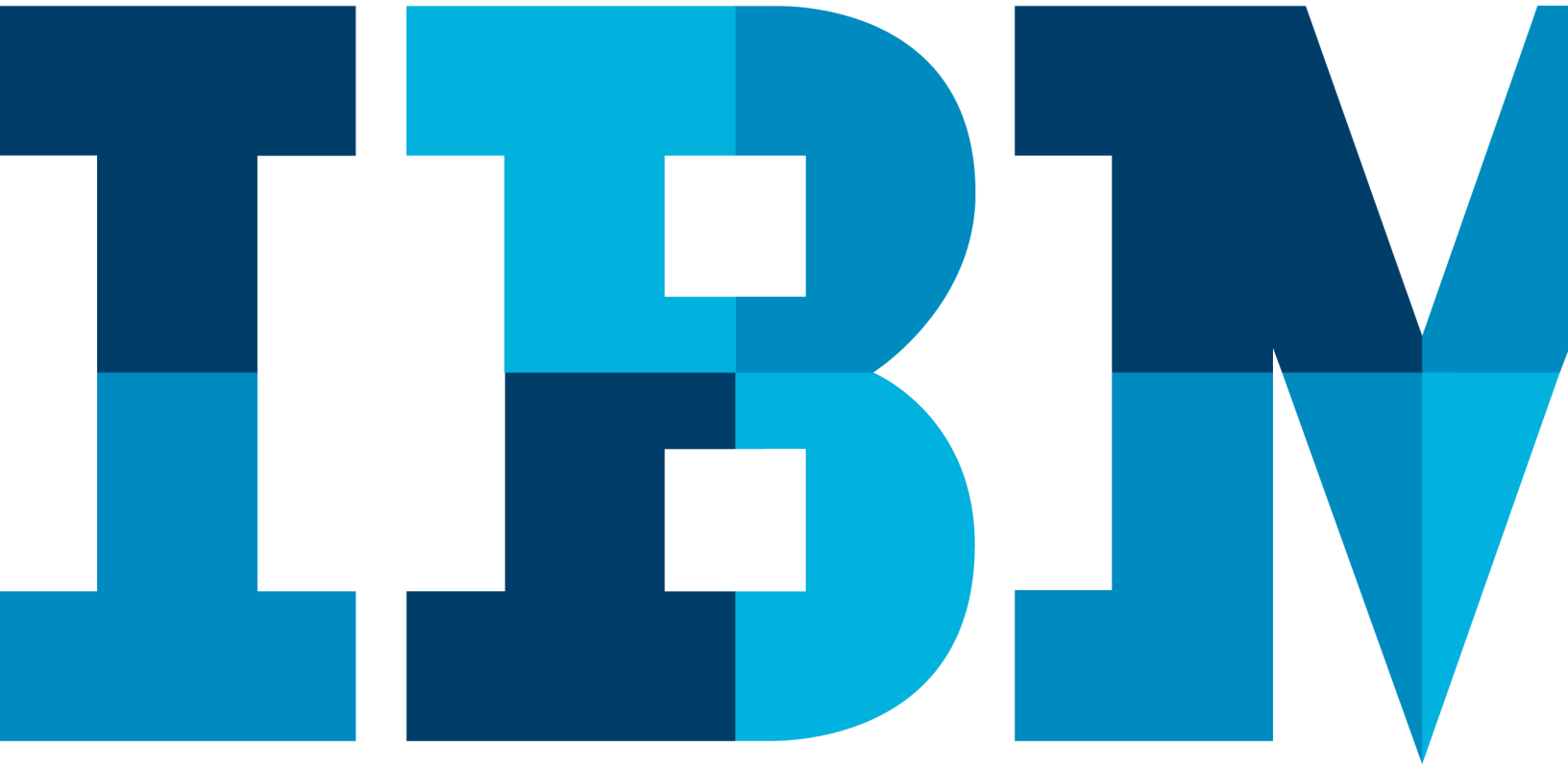
IBM Blockchain Hands-On Composer SDK with IBM Blockchain Platform

Lab Four



IBM

Contents

1 Overview

The aim of this lab is to get you familiar with deploying Hyperledger Composer business networks on IBM Blockchain Platform (Starter Plan).

2 Installation

2.1 Prerequisite

This lab requires to have an IBM Blockchain Starter Plan service subscribed on IBM Cloud.

Install the Nodejs Package Manager npm.
Then initialize your environment :

```
npm init
```

We recommend using Visual Studio Code as text editor to write the code of your Blockchain application (smartcontract as well as client application)

2.2 Install the Composer client on your workstation

You have to install the composer client to deploy the Business Network Archive and interact with the composer server in IBM Cloud.

Open a Linux terminal and issue the following command

```
npm install -g composer-cli@0.20.4
(you may have to install it as administrator. In this case run the following command:
Sudo npm install -g composer-cli@0.20.4)

(on MacOS you have to run the command as root using sudo and adding parameters:
sudo npm install -g composer-cli@0.20.4 --unsafe-perm=true --allow-root)
```

2.3 IBM Blockchain Platform – Starter Membership Plan initialization

The Starter Membership Plan is the IBM Blockchain Platform (Hyperledger Fabric) plan dedicated to development and Proof of Concept of Blockchain solutions.

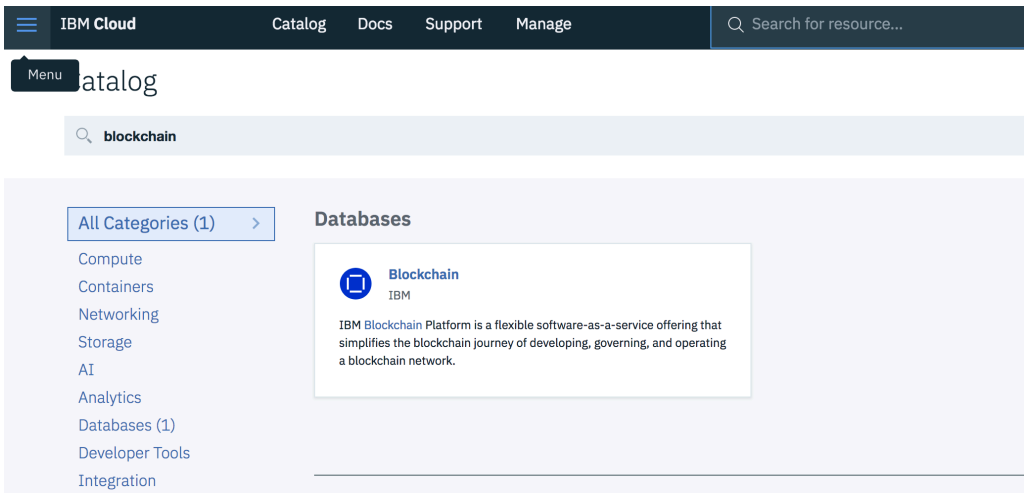
Here, we are going initiate a Blockchain Starter Membership Plan.

PAY ATTENTION: This IBM Cloud service is charged for, but the fees are offered for the first month. After this first trial month, this service is only accessible if you have provided your credit card information in your IBM Cloud profile.

You will activate the service for the duration of the lab. Don't forget to remove , then before the end of the first month, you have to remove it.

Be aware that you can share one Starter Plan between several developers.

Access to your IBM Cloud dashboard (https://console.bluemix.net/catalog) and search for "blockchain" in the search field as shown in the following picture:

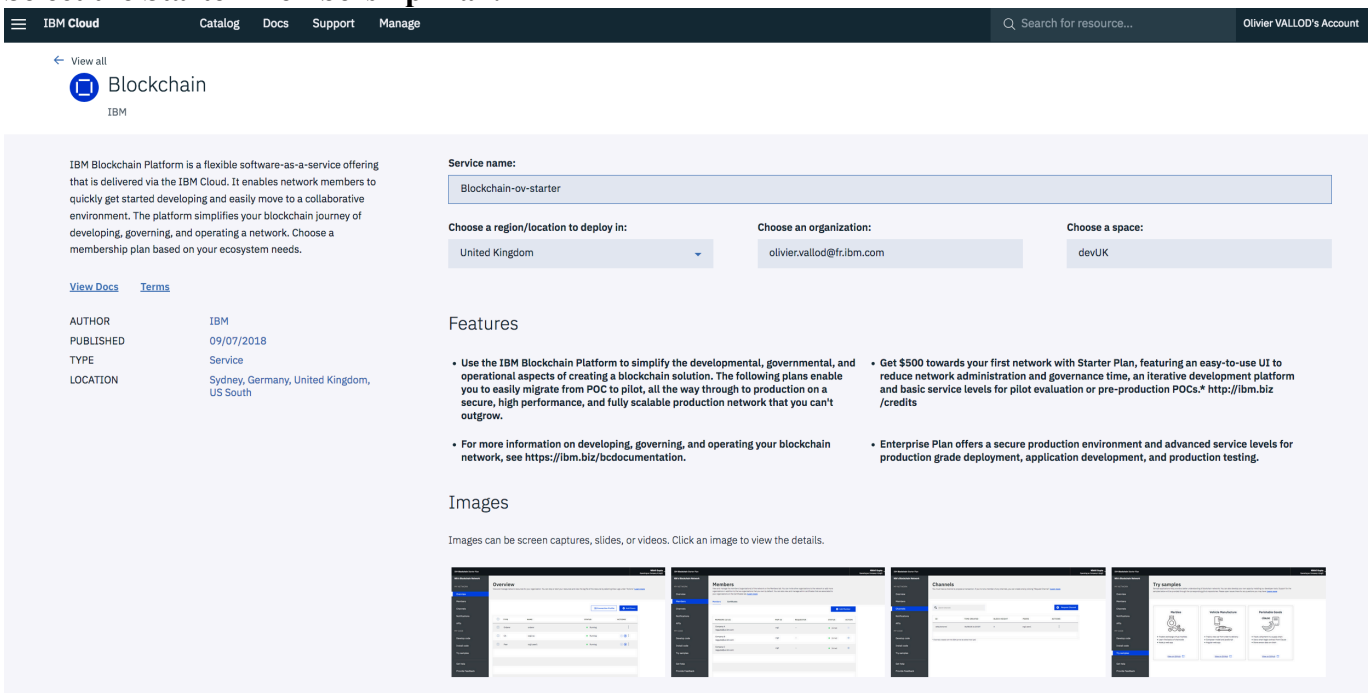


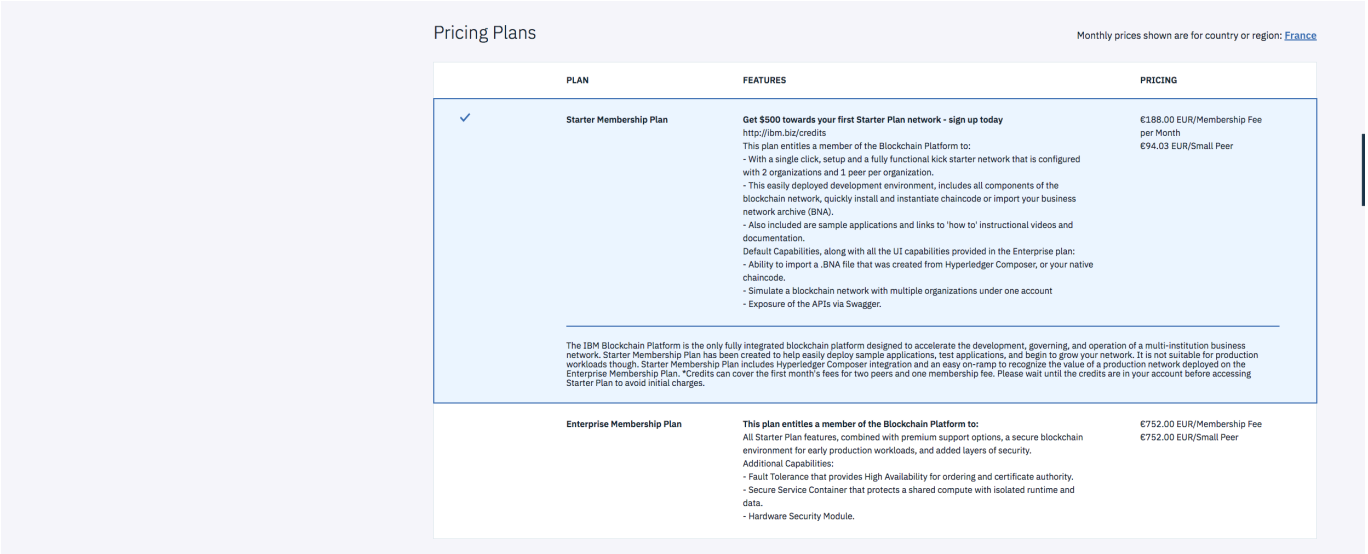
Click on Blockchain frame (under Databases).

In the following panel, you can leave the default information. At the bottom of the page, you will be proposed 2 options:

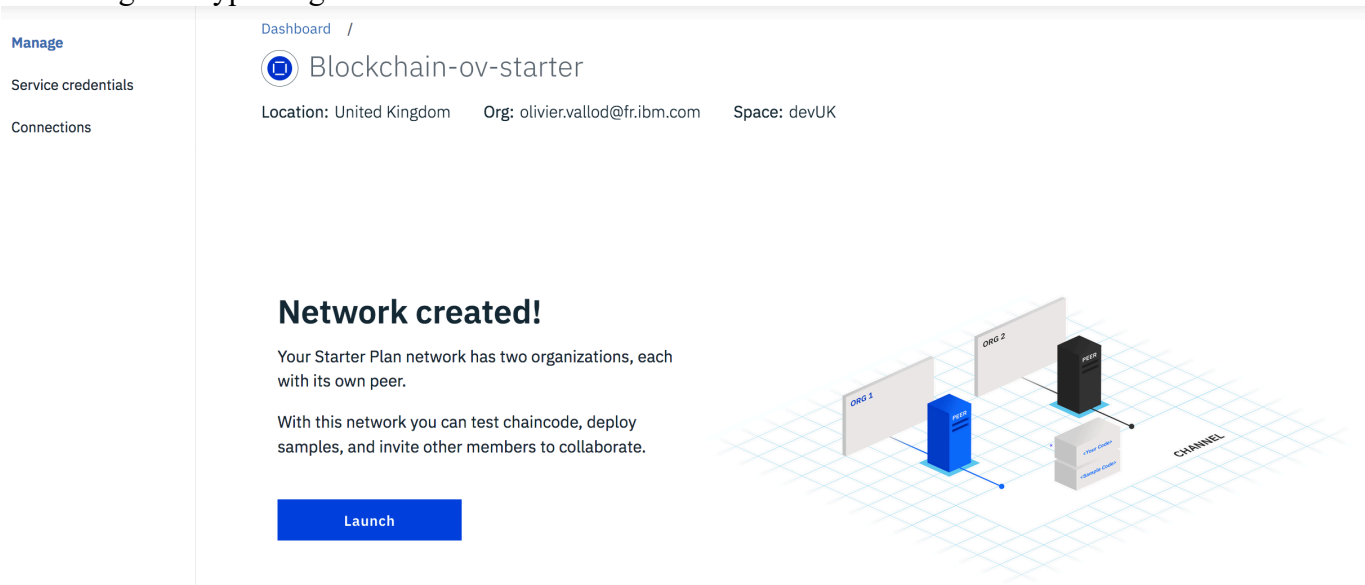
- The Starter Membership Plan.
- The Enterprise Membership Plan.

Select the Starter Membership Plan.





Click on the Create button. Then it displays the following windows (it may take several minutes), describing the Hyperledger fabric environment created. Click on the Launch button.



2.4 Clone GitHub folder for the project

In a Linux terminal window, run the following commands:

```
git clone https://github.com/ajolin/LabBlockchainPlatform.git
```

2.5 Configure the remote access to the IBM Blockchain Starter Plan

2.5.1 Retrieve the connection profile from the starter plan

Go to the Overview windows of the IBSP, then click on connection profile button and choose "download". Then move the file in the marble-project folder, in the file connection-profile.json:
Go to the download folder and run the command:

```
      cd ~/Downloads
      mv <creds_..._org1.json> ~/LabBlockchainPlatform/marble-project/connection-
profile.json

      (on Windows PowerShell, the path to Download should be:
/mnt/c/Users/Administrator/Download)
```

2.5.2 Retrieve the secret in the connection profile file

Display the content of the connection-profile:

```
      cat ~/LabBlockchainPlatform/marble-project/connection-profile.json
```

at the end of the file, under the "certificate authorities" section, (after "registrar") take the "enrolSecret value (the following extract is an example of connection profile):

```
          "certificateAuthorities": {
              "org1-ca": {
                  "url": "https://n846c6d511f81498eb91783a02ceef5dc-org1-
ca.uk02.blockchain.ibm.com:31011",
                  "httpOptions": {
                      "verify": true
                  },
                  "tlsCACerts": {
                      "pem": "-----BEGIN CERTIFICATE…----END CERTIFICATE-----\r\n"
                  },
                  "registrar": [
                      {
                          "enrollId": "admin",
                          "enrollSecret": "464bfa9a26",
                          "x-affiliations": [
                              "org1",
                              "org1.department1",
                              "org1.department2",
                              "org2",
                              "org2.department1"
                          ]
                      }
                  ],
                  "caName": "org1CA",
                  "x-mspid": "org1"
              }
          },
```

2.5.3 Create a certificate authority card and add the certificate to the starter plan

Move to the folder of the connection profile file, then create the certificate authority card using the shell script and the enrollSecret value retrieved in the step before:

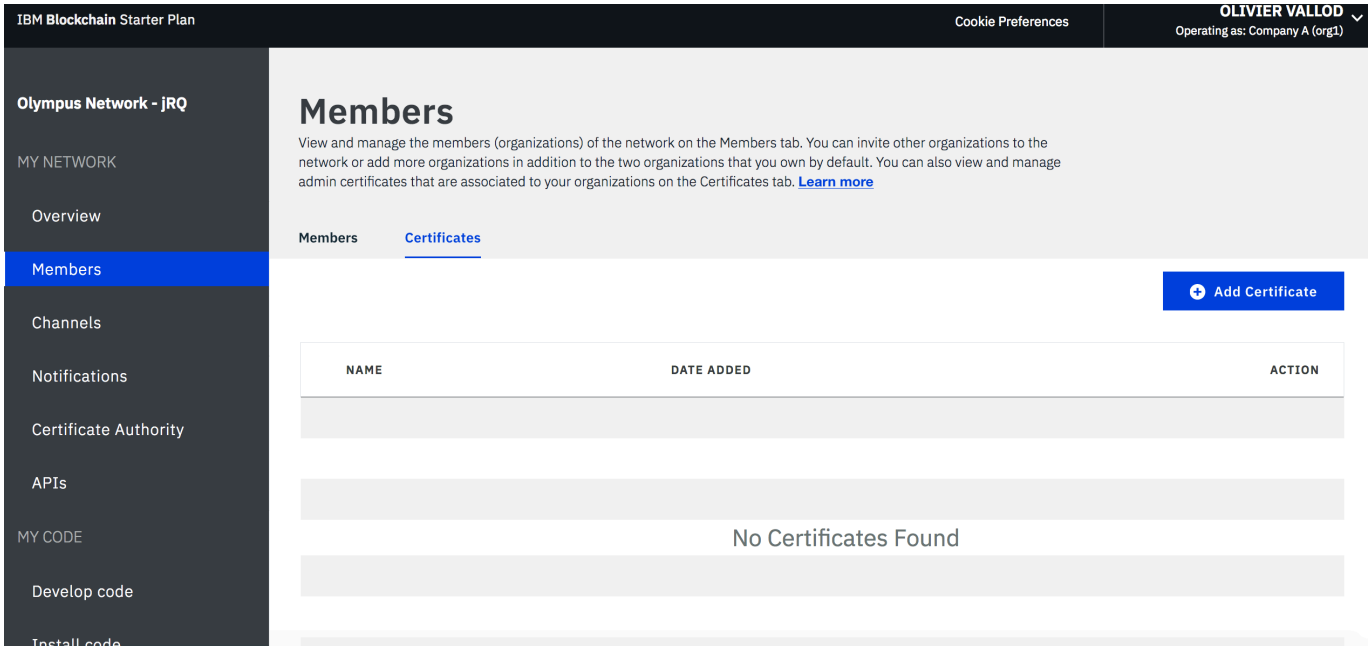
```
cd ~/LabBlockchainPlatform
./create-ca-card.sh <enrollSecret>
```

This script will import the card in your local composer environment.Then it'll retrieve the certificates for the ca (they will be put in the credentials folder of your current folder). For finish, it'll display a certificate.

It should display a certificate similar to:

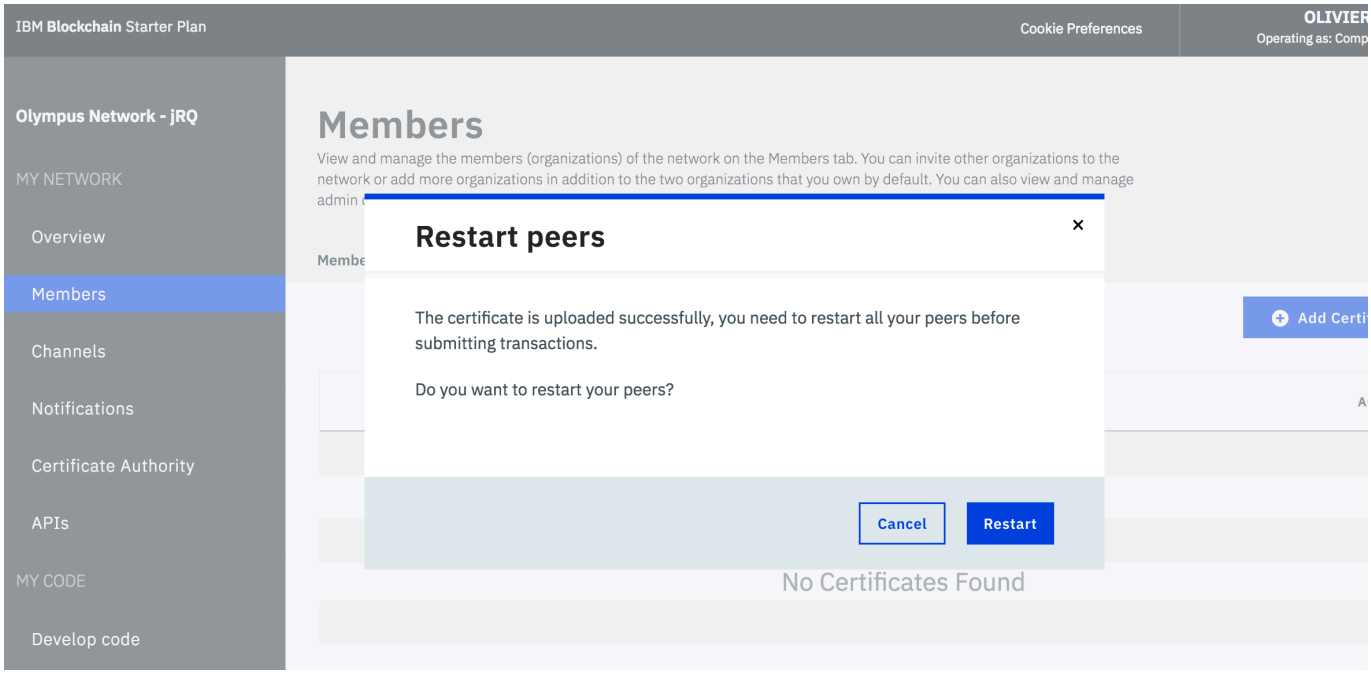
```
-----BEGIN CERTIFICATE-----
MIIB8TCCAZigAwIBAgIUUQyCGAl31oeSW6Hfk1B8gauhGz0wCgYIKoZIzj0EAwIw
….../PMwCgYIKoZIzj0EAwIDRwAw
RAIgBRj2WRb5bejLOAyf18FhdqUv66uezqkvjWTaNv2M6o4CIFrrdSRXtXyr1Dj6
GOxKM7xRnMGtbXKnK8tOmsVdzxTt
-----END CERTIFICATE-----
```

Then copy this certificate content, go to the Starter Plan, click on Member menu, and select the tab Certificates

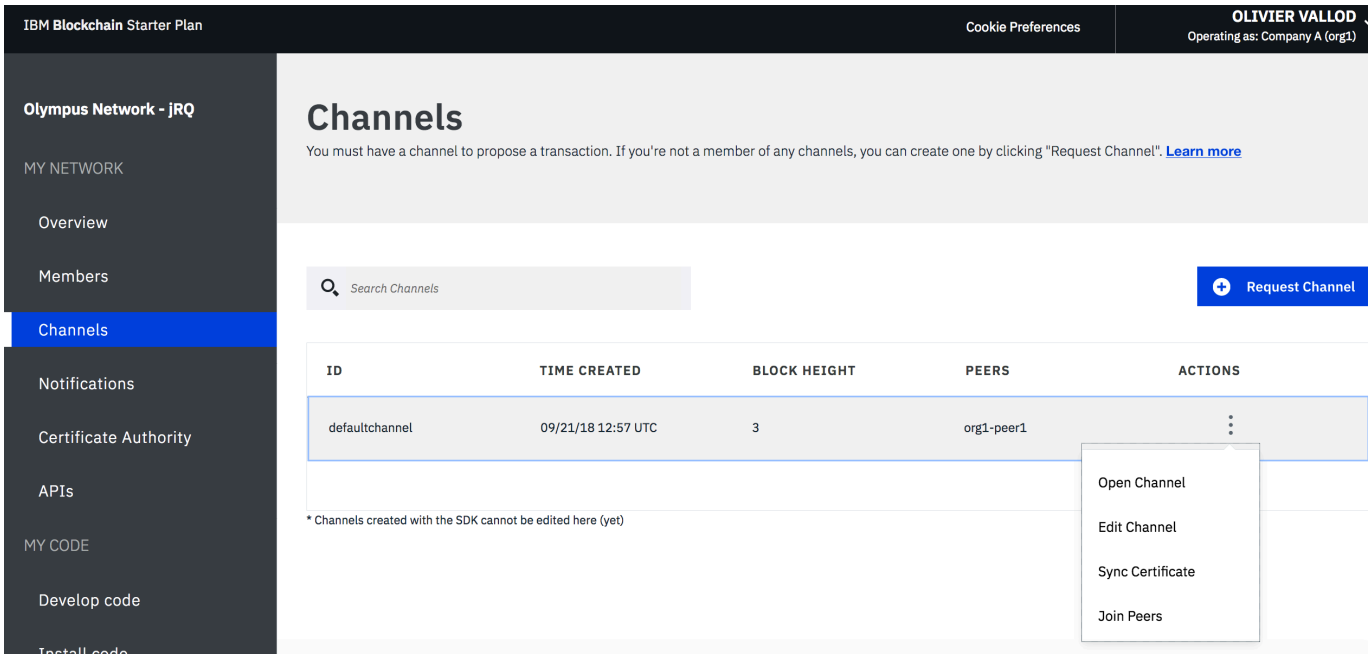


Click on the button Add Certificate, then fill in cert1 as name, and past the certificate content in the field Certificate.

Then it will propose to restart the peers: click on Restart:



Then click on the menu Channels, on the line of the defaultchannel, click on the actions button, then select the menu Sync Certificates, then click on Submit



2.5.4 Creating an admin business network card

Now that the correct certificates have been synced with the peers, business network cards can be created which have the permissions to install the Hyperledger Composer runtime and start chaincode.

Create an admin card with the channel admin and peer admin roles by using the following script:

```
./create-adminCard.sh
```

This card will be used to deploy a business network to Starter Plan.

The script will import the card created and will check if the card is imported

Expected result:

```
The following Business Network Cards are available:

Connection Profile: Olympus Network - jRQ
```

Card Name	UserId	Business Network
ca	admin	
adminCard	admin	

```
Issue composer card list --card <Card Name> to get details a specific card

Command succeeded
```

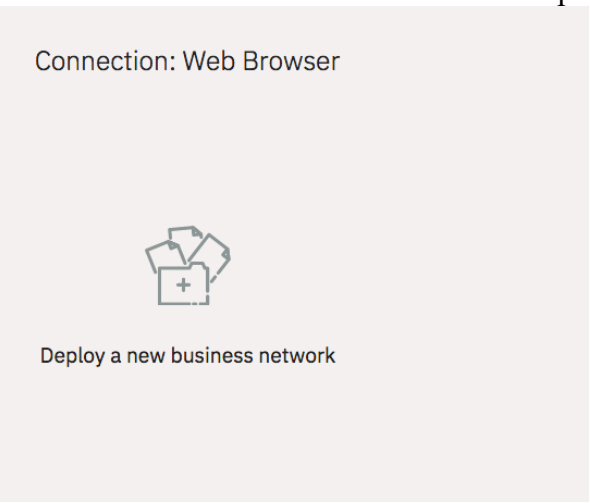
2.6 Create the Business Network (marble-network)

In the Lab 3, we have seen that the Composer Playground allows to create a Business Network and test it.
This business network can be exported as a Business Network Archive (.bna).
This archive file can also be generated from flat file using the composer command.
We are describing both approaches here.

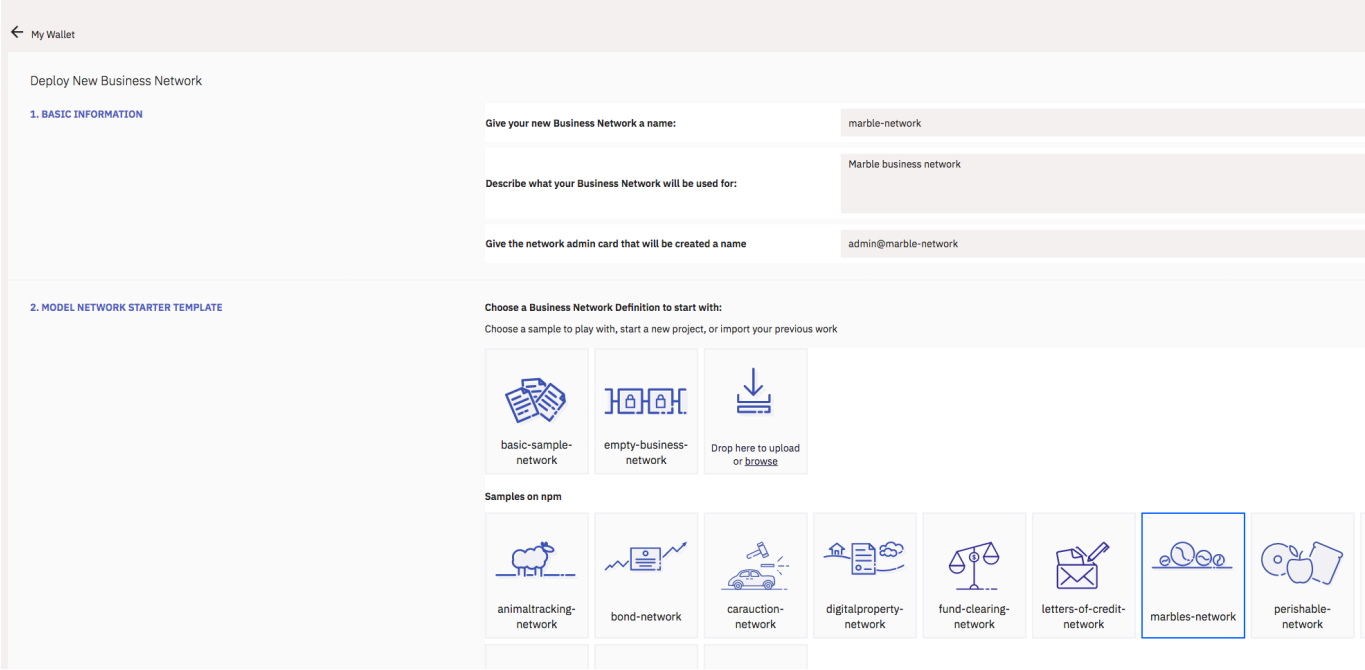
2.6.1 Create the Business Network (marble-network) with composer-playground

Open the Composer Playground (using Firefox browser):
<YOUR PUBLIC IP ADDRESS>:31080
Or **https://composer-playground.mybluemix.net/login**

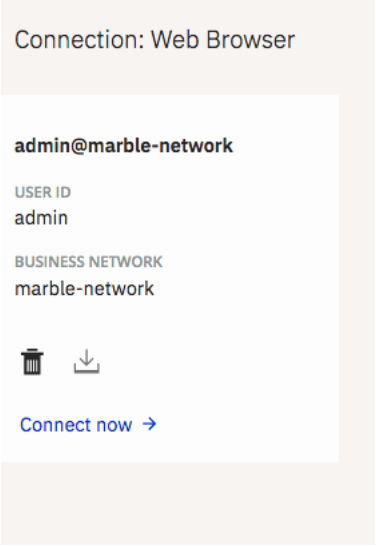
Click on “Connection : Web Browser”> Deploy a new business network



Then fill in the information (name : marble-network, network admin card : admin@ marble-network) and choose marbles-network as Business Network Definition. Then click on DEPLOY.

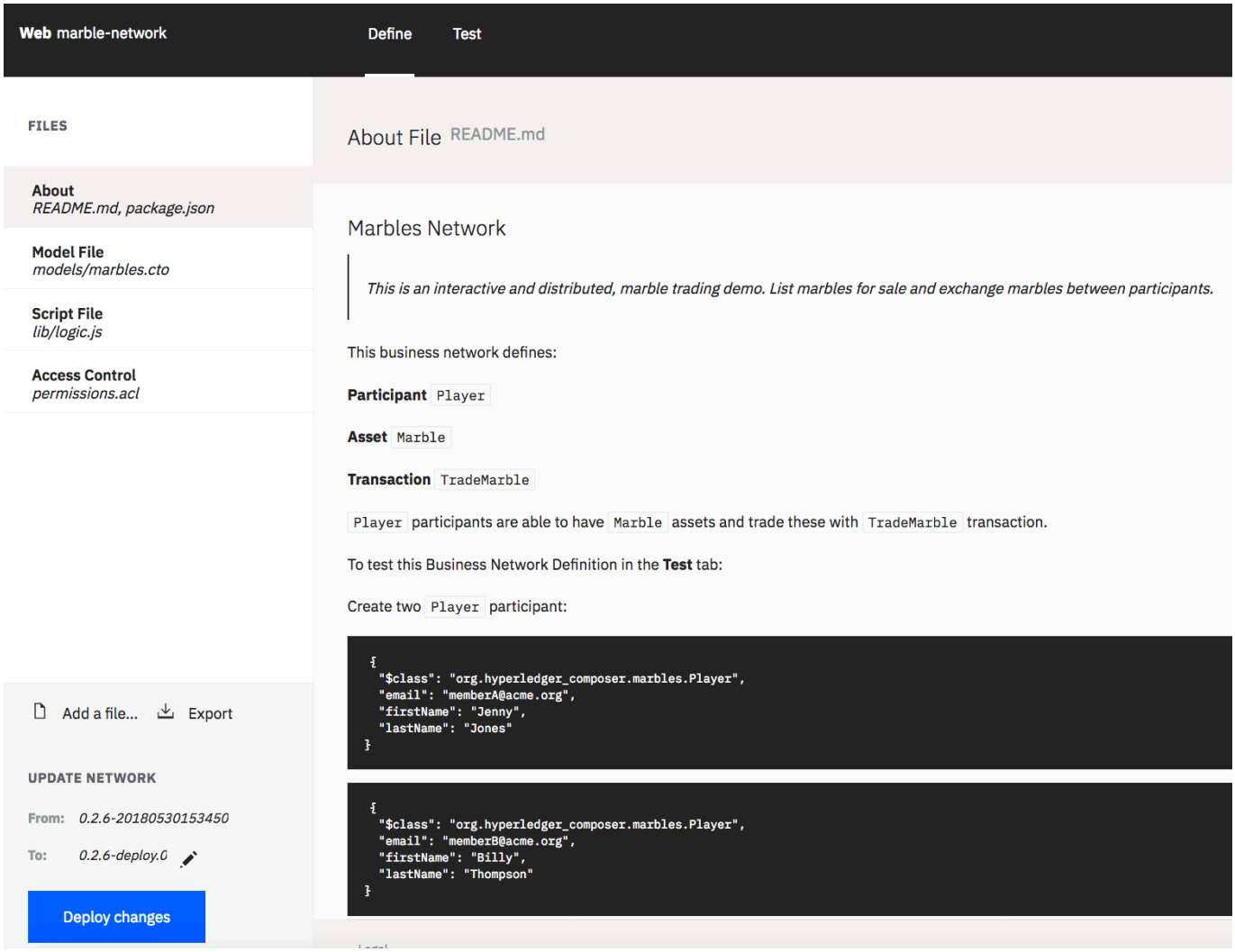


On the My business Networks page, click on Connect now on the marble-network.



Update the Model.cto, and logic.js (retrieve the content given at the chapter 2.4.2).
Then click on Deploy Changes.

Finally, click on export to retrieve the Business Network Archive (.bna) that you will use to deploy on the Hyperledger Fabric in IBM Cloud.



2.6.2 Create the Business Network (marble-network) with composer-cli

Here we are describing the second approach to create a Business Network Archive file : it consists in using the composer-cli (instead of the Composer Playground). We are creating manually the different files that constitutes the BNA - the package description, the ACL, the model, and the logic file. Then we are generating the BNA file.

You can find in marble-network folder all the file to generate the BNA file. To generate the BNA file and deploy on the network, use the following script :

```
./generate-and-deploy-network.sh
```

This script will first create the BNA file. It should return you some information.
Expected result:

```
Creating Business Network Archive


Looking for package.json of Business Network Definition
     Input directory: /Users/ovallod/git/marbles-network

Found:
     Description: Marble Trading Network
```

```
        Name: marbles-network
        Identifier: marbles-network@0.1.1-deploy.0

    Written Business Network Definition Archive file to
        Output file: marbles-network@0.1.1-deploy.0.bna

    Command succeeded
```

In a second time, the script will install the BNA onto your network. You will get those information if everything is ok :

```
    ✔ Installing business network. This may take a minute...
    Successfully installed business network marbles-network, version 0.1.1-deploy.0

    Command succeeded
```

Then the script will start the business network and generate the business network card for the administration of this BN.

Expected result:

```
    Starting business network marbles-network at version 0.1.1-deploy.0

    Processing these Network Admins:
        userName: admin

    ✔ Starting business network definition. This may take a minute...
    Successfully created business network card:
        Filename: admin@marbles-network.card

    Command succeeded
```

Now we create a Business Network card for the operator of the business network by using the following script :

```
./create-business-card.sh
```

This script will import the BN card of the admin user and will ping the BN in order to check it is up running and to do the enrollment of the admin user.

Expected result:

```
    The connection to the network was successfully tested: marbles-network
        Business network version: 0.1.1-deploy.0
        Composer runtime version: 0.20.4
        participant: org.hyperledger.composer.system.NetworkAdmin#admin
        identity:
    org.hyperledger.composer.system.Identity#7c680cf85063a9f8b1082acd5e0a31daf856cf2fb6a8e55f
    0eb288529ba2218d

        Command succeeded
```

At this stage, we have prepared the local environment (composer cli) to access to the remote environment (Hyperledger Fabric and Composer in a Kubernetes clusters in IBM Cloud - ex Bluemix -) and we have deployed the Business Network archive of the Marble Smartcontract.

3 Manipulating and Adding Resources with the SDK

In this section, we will first look at how to connect to a running fabric instance with the composer node.js SDK and secondly follow this by looking at to adding and updating resources.

3.1 Getting Started

First, you'll find all the resource needed for this section into the marble-client folder.

```
cd marble-client
```

3.2 Create the participants and assets

In this part, we are creating a simple NodeJS application which will create participants and asset of the marble-network.

Use a text editor to modify the file add-resources.js

First we reference the NodeJS lib that we are using : composer-client, then we put the squeleton of our application : an asynchronous function 'createResources()' and the call to this function. The result (t) will be displayed ('console.log(t)').

```
const BusinessNetworkConnection = require('composer-client').BusinessNetworkConnection;

async function createResources() {
    try {
       …
    } catch (error) {
        console.log(error);
        process.exit(1);
    }
}
createResources().then((t) => {
    console.log(t);
});
```

Then in the core of the function, those two line start the connection to the Business Network.

```
        // Connect to the Business Network
        let bizNetConnection = new BusinessNetworkConnection();
        let bizNetDef = await bizNetConnection.connect("admin@marbles-network");
```

Then just after, retrieve the description of the Business Network.

```
        // Retrieve the description of the Business Network
        let factory = bizNetDef.getFactory();
```

Then we are creating the first resource which is a participant of type Player.

```
        // Create a resource of type Player
        let player1 = factory.newResource('org.hyperledger_composer.marbles', 'Player',
'email:olivier2@mele');
        player1.lastName = 'Truc';
        player1.firstName = 'Olivier2';
```

Create other Player resources, replicating this code and changing the values: create player2 and player3.

Then retrieve the participant registry of our BN ('org.hyperledger_composer.marbles.Player') and add the created resources - player1, player2 and player3 - in this registry

```
       //retrieve the participant registry and add the Player resources
        let playerRegistry = await
bizNetConnection.getParticipantRegistry('org.hyperledger_composer.marbles.Player');
        await playerRegistry.addAll([player1, player2, player3]);
```

At this stage, you can run this application which will create 3 Participants: in the folder, you can run the following command:

```
       node add-resources.js
```

To check the added resources, use the command:

```
       composer network list -c admin@marbles-network
```

Pay attention to comment the line “await playerRegistry.addAll …” after running the application to avoid an error with duplicate resource. (You can also change the value of the emailId of each Player).

Now we are going to add the assets. Create a new resource of type Marble. Then assign the values to each field (size, color). Then, create a relationship with the selected Player giving his email. For this, uncomment the following line in your file

```
        let marble = factory.newResource('org.hyperledger_composer.marbles', 'Marble',
'marbleId:1');
        marble.size = 'MEDIUM';
        marble.color = 'ORANGE';
        marble.owner = factory.newRelationship('org.hyperledger_composer.marbles', 'Player',
'email:olivier1@mele');
```

Create marble1 and marble2 resources replicating the previous code and changing the values.

Then retrieve the marble registry of our BN ('org.hyperledger_composer.marbles.Marble') and add the created resources - marble, marble1 and marble2 - in this registry. To continue, uncomment the following line in the file.

```
        let marbleRegistry = await
bizNetConnection.getAssetRegistry('org.hyperledger_composer.marbles.Marble');
        await marbleRegistry.add(marble);
        await marbleRegistry.add(marble1);
        await marbleRegistry.add(marble2);
```

3.3 Get and display the list of Marbles

Now we are displaying the list of marbles. Uncomment the following line :

```
        marbleRegistry.getAll()
        let marbles = await marbleRegistry.getAll();
        let tMarbles = new Array({
            head: ['MarbeId', 'Owner', 'Size', 'Color']
        });
        let arrayLength = tMarbles.length;
        marbles.forEach((marble) => {
            let tableLine = [];
            tableLine.push(marble.marbleId);
            tableLine.push(marble.owner);
            tableLine.push(marble.size);
            tableLine.push(marble.color);
            tMarbles.push(tableLine);
        })


        bizNetConnection.disconnect();

        // Put to stdout - as this is really a command line app
        return tMarbles;
```

Run this application with the following command:

```
       node add-resources.js
```

4 Using the composer rest server

In this chapter, we will use the Composer Rest server to publish the transactions as Rest API.

Then we will develop a web application to use these API.

4.1 Install the Composer Rest server on your workstation

You have to install the composer client to deploy the Business Network Archive and interact with the composer server in IBM Cloud.

Open a Linux terminal and issue the following command

```
npm install -g composer-rest-server@0.20.4

(on MacOS you have to run the command as root using sudo and adding parameters:
sudo npm install -g composer-rest-server@0.20.4 --unsafe-perm=true --allow-root)
```

4.2 Start the Composer Rest server

We are starting the composer rest server with the business network card of the operator of the business network. Then it will automatically detect the associated smart contract, and it will create the Rest API to map this smart contract

Issue the following commands in a Linux terminal:

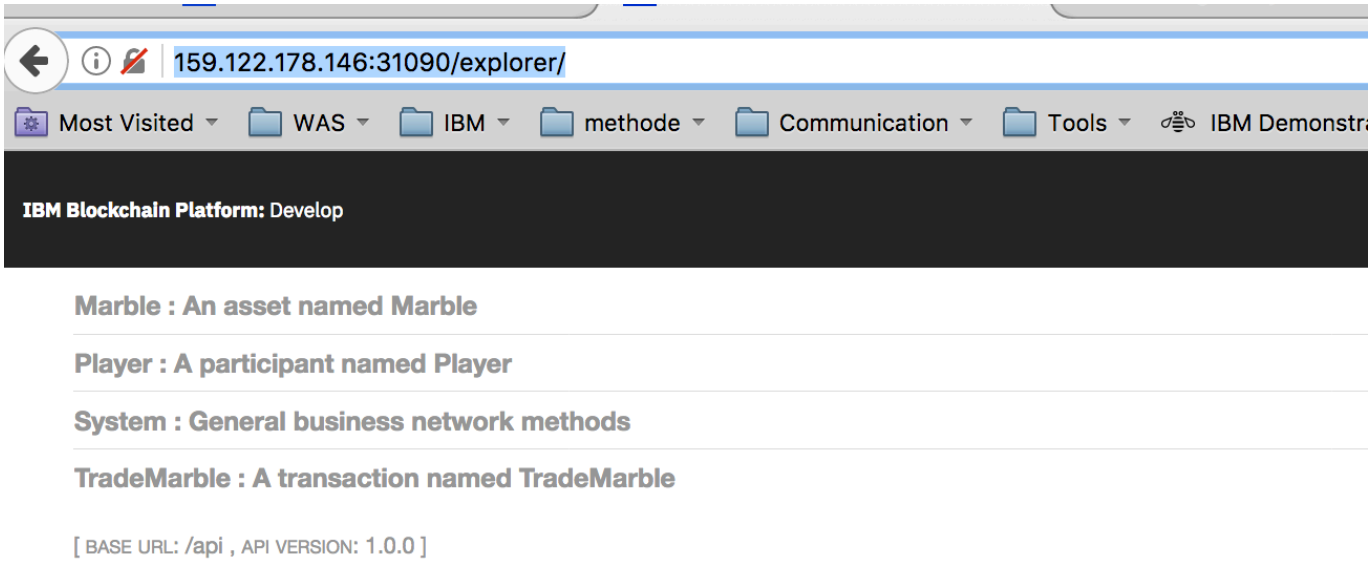
```
composer-rest-server -c admin@marbles-network -n always -u true -w true
```

Expected result:

```
WARNING: NODE_APP_INSTANCE value of '0' did not match any instance config file names.
WARNING: See https://github.com/lorenwest/node-config/wiki/Strict-Mode
Discovering types from business network definition ...
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

4.3 Check the REST API

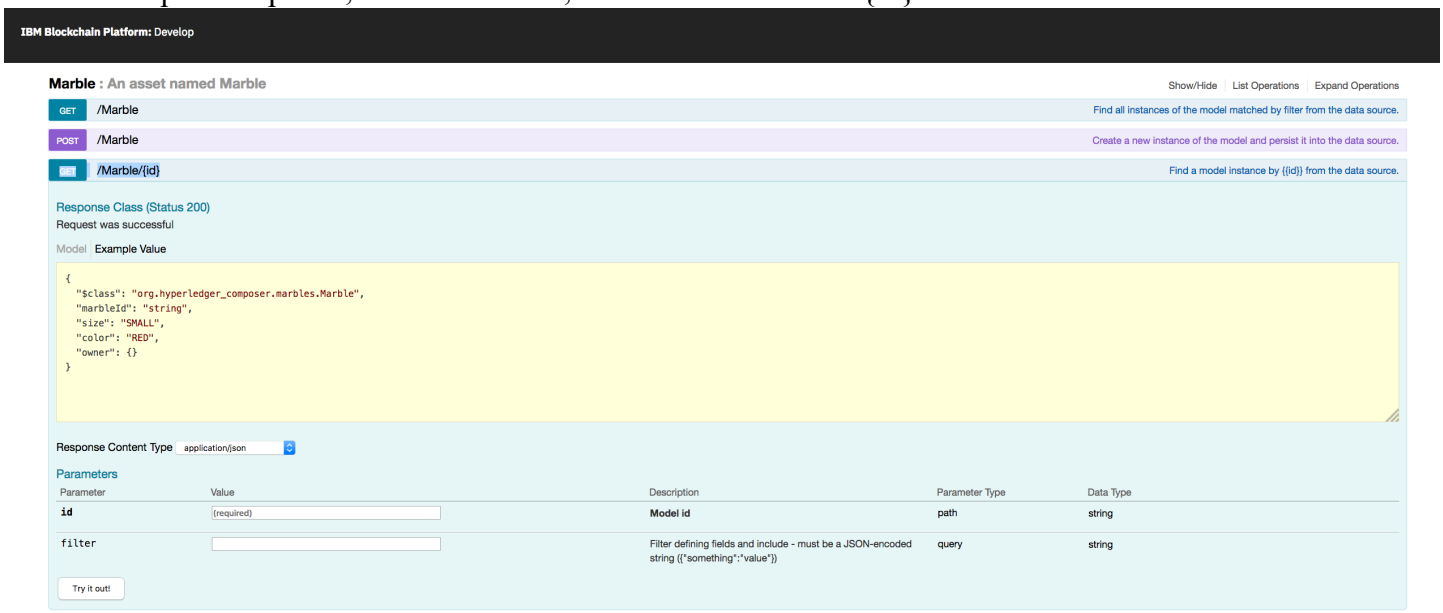
Open your internet browser and access the explorer: http://localhost:3000/explorer



Now you are ready to access to your Business Network through Rest API.
We are going to explore these API.

4.3.1 Display a marble

On the Composer explorer, click on Marble, then on “Get /Marble/{id}”



In the field id, put “marbleId:1” then click on the “Try it out” button. It will display the content of the asset Marble 1 as shown in the following picture.

Curl

```
curl -X GET --header 'Accept: application/json' 'http://159.122.178.146:31090/api/Marble/marbleId%3A1'
```

Request URL

```
http://159.122.178.146:31090/api/Marble/marbleId%3A1
```

Response Body


```
{
  "$class": "org.hyperledger_composer.marbles.Marble",
  "marbleId": "marbleId:1",
  "size": "MEDIUM",
  "color": "ORANGE",
  "owner": "resource:org.hyperledger_composer.marbles.Player#email:olivier1@mele"
}
```


Response Code

```
200
```

Response Headers


```
{
  "vary": "Origin, Accept-Encoding",
  "access-control-allow-credentials": "true",
  "x-xss-protection": "1; mode=block",
  "x-frame-options": "DENY",
  "x-download-options": "noopen",
  "x-content-type-options": "nosniff",
  "content-type": "application/json; charset=utf-8",
  "content-length": "188",
  "etag": "W/\"bc-Gv6r+ZDmEgVddhoiWFnzYsaFfeg\"",
  "date": "Sat, 02 Jun 2018 18:54:53 GMT",
  "connection": "keep-alive"
}
```


4.3.2 Execute a transaction

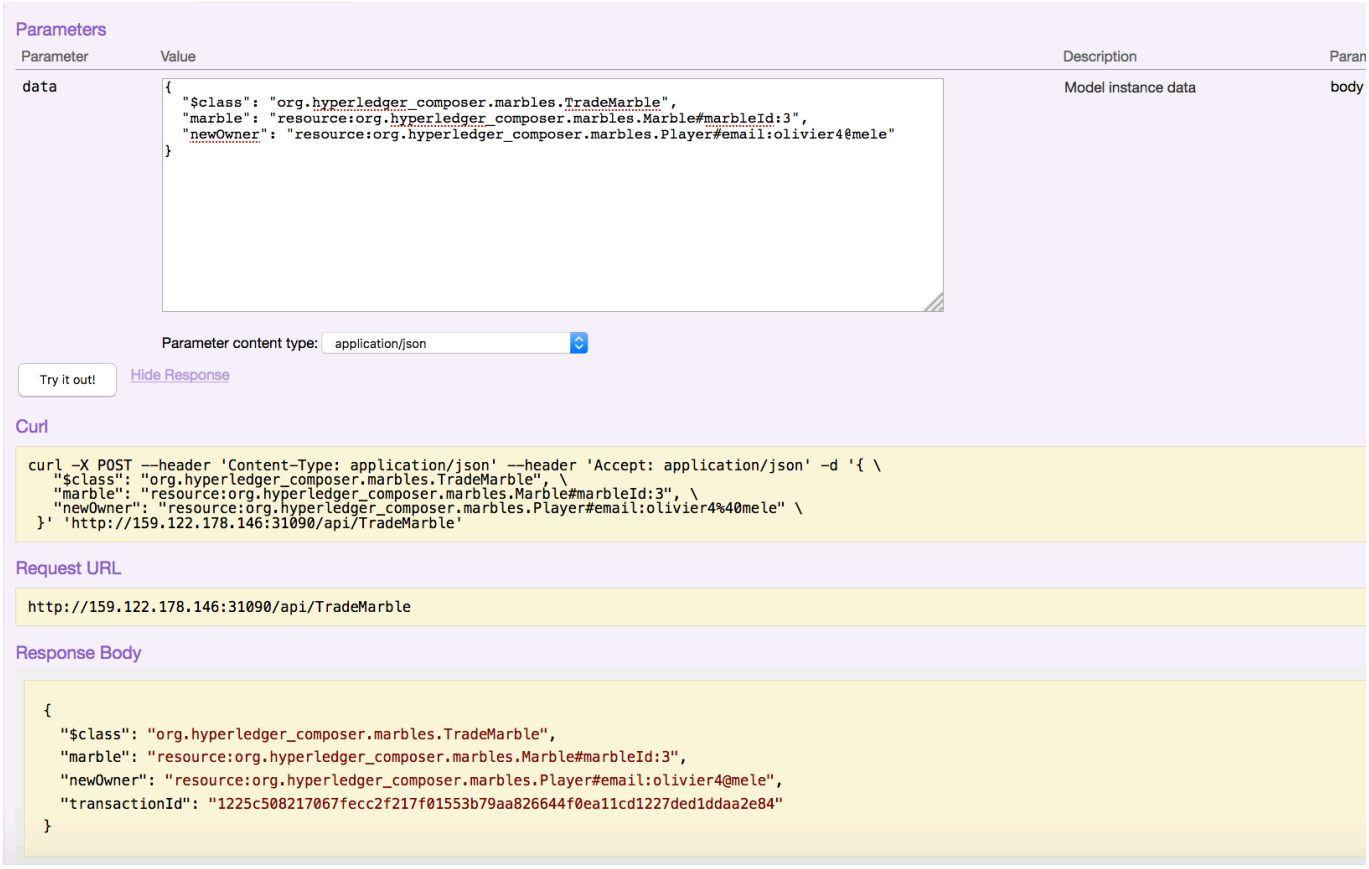
We are going to change the owner of the marbleId:3.

On the Composer explorer, click on TradeMarble, then on “POST /TradeMarble”.

In the field ‘data’, specify the transaction input data:

```
{
  "$class": "org.hyperledger_composer.marbles.TradeMarble",
  "marble": "resource:org.hyperledger_composer.marbles.Marble#marbleId:1",
  "newOwner": "resource:org.hyperledger_composer.marbles.Player#email:olivier4@mele"
}
```

Then click on the button” Try it out” to run the transaction



This concludes the lab on composer SDK development. More information on the SDK can be found here:

https://hyperledger.github.io/composer/jsdoc/index.html

Notices

This information was developed for products and services offered in the U.S.A.
IBM may not offer the products, services, or features discussed in this document in other countries.

Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
IBM may have patents or pending patent applications covering subject MGKer described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.
Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM

products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.
COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix A. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
System z	*Tivoli*	*WebSphere*	*Workplace*	*System p*	

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

NOTES

NOTES