



# Test drive Hyperledger™ Composer Playground and blockchain demo app with Hyperledger™ Fabric on IBM LinuxONE™ server

Yongkook(Alex) Kim @likepunk

System z Solutions Architect - Vicom Infinity

<http://blockchain.infinite-blue.com>

2017 June 23 VMWorkshop at Ohio State University



The goal of this lab is:

- 1) to let you experience bringing up the Hyperledger™ network/fabrics on LinuxONE™ server
- 2) and then deploy sample applications using either Fabric-composer playground and/or Marbles App on LinuxONE™ server

Estimated run time: 2 hours

This lab utilizes Hyperledger™ Fabric v0.6 and composer-playground v0.7.5 at the time of writing this material (June 22, 2017). It may be updated later when Fabric version 1.0 and composer-playground gets stabilized in the future.

Special thanks to following people who helped make this lab possible:

Barry Silliman @ IBM Washington Systems Center

John Harrison @ IBM Washington Systems Center

Steve LaFalce @ IBM z Systems Technical Enablement

Chee Ye @ IBM z Systems Technical Enablement

Matt Lucas @ IBM Global Blockchain Enablement

# [0] Prepare LinuxONE™ Community Cloud Server

## Sign Up for the LinuxONE™ community server

Visit LinuxONE™ cloud service web site (<https://developer.ibm.com/linuxone/request-a-trial> ) and request for the free service and **create SLES12 SP2 - Medium sized server**. It may take up to a day or two to get a login credentials. Once you get id/passwd, sign in as below.



IBM LinuxONE™  
Community Cloud

[Request a trial](#)

[Resources](#)

[News & Alerts](#)

[Terms of Use](#)

### THANK YOU

Your registration is under review. Your request may take up to one (1) business day to process. You will receive emails containing log on credential once your request is approved. Thank you for your patience.

[Sign in](#)

Click on left side 'Virtual Servers/ Manage Instances' where you can create a Linux server.

## Test drive your apps and more in the LinuxONE Community Cloud

Unleash your open source apps and services with the Linux platform without limits.



### Virtual Servers

Quickly and easily deploy a Linux instance with the Linux distribution of your choice – Red-Hat, SUSE, and coming soon, Ubuntu. Your 120 day trial includes up to 2 virtual CPU, 4 GB memory and 40 GB of storage! Refer to this [Virtual Servers Quick-Start Guide](#).

Manage Instances



### Blockchain Service

Click the button below and follow a quick-start guide to help you spin up a test Blockchain Network quickly in your deployed instance. You can then run an example smart contract application or your own custom chain code.

Try Blockchain Local



### Apache Spark Service

Click the button below and follow a quick-start guide to help you install Apache Spark quickly in your deployed instance. You can then run two examples in your Apache Spark environment.

Try Apache Spark Local

Enter instance details(step1), select SLES12 SP2(step2) and LinuxONE-Medium as a flavor(step3).

You are in: [Instances](#) / [Create Instance](#)

Project:

### Step 1: Enter Instance Details

Instance name \*

Instance Description

### Step 2: Select an Image

RHEL 6.7

RHEL7.2

SLES11 SP4

SLES12 SP2

### Step 3: Select a Flavor

LinuxONE-Small

CPU: 1 core(s)  
Memory: 2048MB  
Disk: 40GB

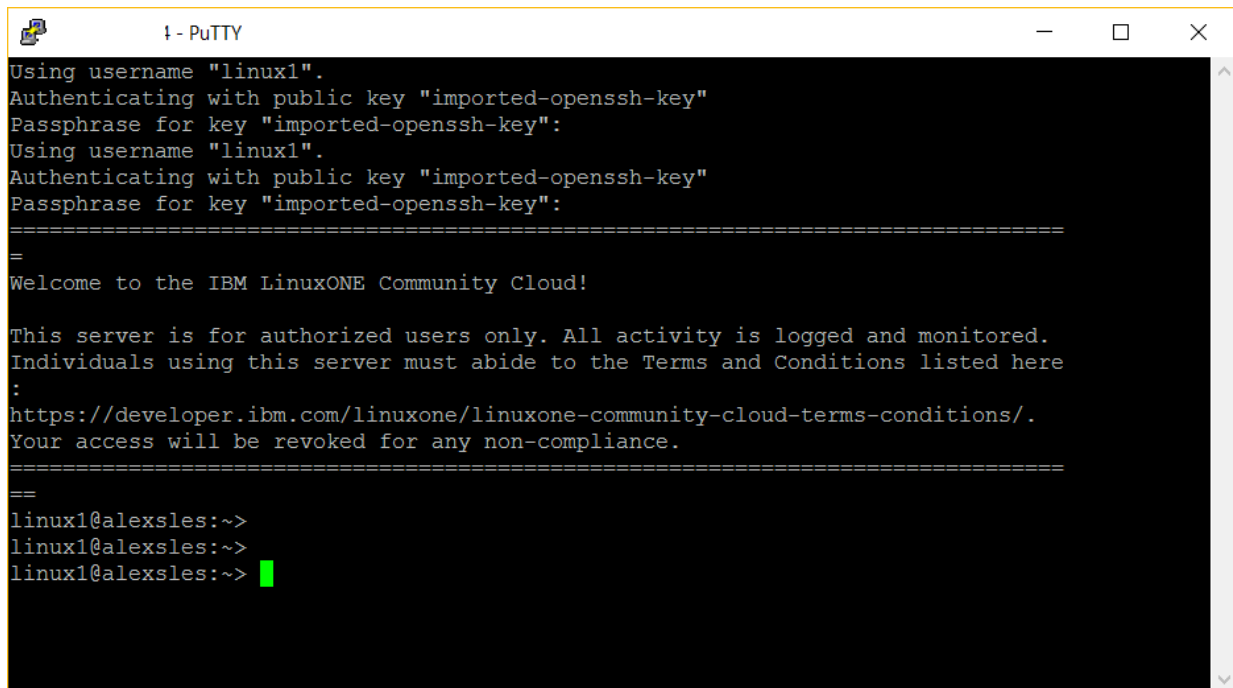
LinuxONE-Medium

CPU: 2 core(s)  
Memory: 4096MB  
Disk: 40GB

**CURRENT SELECTION:**

**Name:**AlexSLES12

Using SSH key that was provided, login to the terminal. If you don't have SSH terminal nor don't know how to connect please refer to this document → <https://developer.ibm.com/linuxone/wp-content/uploads/sites/57/virtual-servers-quick-start.pdf>



```
+- PuTTY
Using username "linux1".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Using username "linux1".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
=====
=
Welcome to the IBM LinuxONE Community Cloud!

This server is for authorized users only. All activity is logged and monitored.
Individuals using this server must abide to the Terms and Conditions listed here
:
https://developer.ibm.com/linuxone/linuxone-community-cloud-terms-conditions/.
Your access will be revoked for any non-compliance.
=====
==
linux1@alexsles:~>
linux1@alexsles:~>
linux1@alexsles:~> █
```

## Install docker-compose tool

Since most of settings are already pre-built in LinuxONE community server, only thing you need to do is installing 'docker-compose' tool. Follow directions from below or website for other distros and make sure you have docker-compose running.

[https://github.com/linux-on-ibm-z/docs/wiki/Building-Docker-Compose?cm\\_mc\\_uid=81288449110413994410116&cm\\_mc\\_sid\\_50200000=1497929327](https://github.com/linux-on-ibm-z/docs/wiki/Building-Docker-Compose?cm_mc_uid=81288449110413994410116&cm_mc_sid_50200000=1497929327)

- 1) Install dependencies

```
sudo zypper install -y python-pyOpenSSL python-setuptools
```

- 2) Install pip with easy\_install

```
sudo easy_install pip
```

3) Install docker-compose itself

```
sudo pip install docker-compose==1.13.0
```

4) verify the Docker Compose installation

```
docker-compose version
```

# [1] Install/setup Hyperledger Fabric v0.6

Following steps will guide you to install/run Hyperledger Fabric on your server. You can either run it with {a single peer + membership services} OR {four peers + membership services}

Reference: <https://hub.docker.com/r/ibmblockchain/fabric-peer/>

## 1.1 Switch user to root:

During the lab, all the commands will be entered as 'root' user id unless specified otherwise. If you are not currently switched to root yet, please enter: ' **sudo su -** ' at the shell command line.

```
linux1@bctest01:~> sudo su -  
bctest01:~ #
```

## 1.2 Pre-requisites

git version 2.12 or higher (*source management tool, already installed for the lab*)

Docker version 1.12 or higher (*already installed for the lab*)

Docker-compose version 1.11.2 or higher

(*already installed for the lab, so you don't need to do anything here. If not, or if you want to try in your own LinuxONE image, follow steps in below site: <https://github.com/linux-on-ibm-z/docs/wiki/Building-Docker-Compose> )*

### About Docker:

Docker is a container/virtualization technology and a key component of Hyperledger. It is not required to know during the lab but it is important to understand basic operations if you want to continue working with Hyperledger.

If you want to learn more about Docker, please refer to <https://docs.docker.com/>

There is a very nice all-inclusive playground website you can learn and try on Docker technology in here: <http://training.play-with-docker.com/>

## 1.3 Clone docker-compose files and bring up the Hyperledger fabric

Change directory to where you want to copy Hyperledger fabric configuration files. For this lab go to /data:

Please enter ' **cd /data** ',

and then enter ' **git clone <https://github.com/IBM-Blockchain/fabric-images.git>** ' ,



```
bctest01:~ #  
bctest01:~ # cd /data  
bctest01:/data # git clone https://github.com/IBM-Blockchain/fabric-images.git
```

Now you will see 'git' is cloning files from github site down to your local directory. Once you get prompt back, change directory to /data/fabric-images/docker-compose/

Please enter ' **cd fabric-images/docker-compose** '

Once changed the directory,

Please enter ' **. setenv.sh** '

→ make sure you enter dot(.) and space. This will check what HW platform the Hyperledger fabric will be running on. We are running on z Systems(s390x). Now bring up the Hyperledger fabric by entering following.

Please enter ' **docker-compose -f single-peer-ca.yaml up** '

```
[linux1@alexrhel fabric-images]$ cd docker-compose/  
[linux1@alexrhel docker-compose]$ ls  
base baseimage four-peer-ca.yaml README.md setenv.sh single-peer-ca.yaml  
[linux1@alexrhel docker-compose]$ . setenv.sh  
[linux1@alexrhel docker-compose]$ docker-compose -f single-peer-ca.yaml up  
Creating dockercompose_membersrv_1 ...  
Creating dockercompose_baseimage_1 ...  
Creating dockercompose_membersrv_1  
Creating dockercompose_membersrv_1 ... done
```

Docker-compose is a command to invoke multiple Docker images to bring up, and connect them together to make a 'service' or 'services'. Note that you are bringing up a Hyperledger network with single peers and one membership services. You can also have four peers + membership services if you specify 'four-peer-ca.yaml' instead.

```
atc  
vp_1 | 03:35:00.473 [buckettree] ClearWorkingSet -> DEBU 071 Enter - ClearWorkingSet()  
vp_1 | 03:35:00.473 [nodeCmd] serve -> DEBU 072 Running as validating peer - installing consensus  
vp_1 | 03:35:00.473 [peer] initDiscovery -> DEBU 073 Retrieved discovery list from disk: []  
vp_1 | 03:35:00.474 [consensus/controller] NewConsenter -> INFO 074 Creating default consensus plugin (noops)  
vp_1 | 03:35:00.474 [consensus/noops] newNoops -> DEBU 075 Creating a NOOPS object  
vp_1 | 03:35:00.478 [consensus/noops] newNoops -> INFO 076 NOOPS consensus type = *noops.Noops  
vp_1 | 03:35:00.478 [consensus/noops] newNoops -> INFO 077 NOOPS block size = 500  
vp_1 | 03:35:00.478 [consensus/noops] newNoops -> INFO 078 NOOPS block wait = 1s  
vp_1 | 03:35:00.478 [peer] chatWithSomePeers -> DEBU 079 Starting up the first peer of a new network  
vp_1 | 03:35:00.478 [consensus/statetransfer] verifyAndRecoverBlockchain -> DEBU 07a Validating existing blockchain, highest validated block is 0, valid through 0  
vp_1 | 03:35:00.478 [consensus/statetransfer] blockThread -> INFO 07b Validated blockchain to the genesis block  
vp_1 | 03:35:00.478 [consensus/handler] 1 -> DEBU 07c Starting up message thread for consenter  
vp_1 | 03:35:00.479 [peer] ensureConnected -> DEBU 07d Starting Peer reconnect service (touch service), with period = 5mls  
vp_1 | 03:35:00.479 [nodeCmd] serve -> INFO 07e Starting peer with ID=name:"vp0" , network ID=dev, address=172.19.0.3:7051, rootnodes=, validator=true  
vp_1 | 03:35:00.479 [rest] StartOpenchainRESTServer -> INFO 07f Initializing the REST service on 0.0.0.0:7050, TLS is disabled.
```

Now you will see a lot of messages scrolling and most of time it is hard to see – not to worry about them for now. Once you see `dockercompose_baseimage_1` exited with code 0(1<sup>st</sup> screenshot), and about a minute or so later you will see a message ‘Discovery knows about:’ for each peer.

Now Hyperledger fabric is ready to take an action!

All of the messages are coming from two(2) different Docker images(Membership Services + 1 peer). If this was four-peer model(if you used four-peer-ca.yaml for docker-compose) you can see there are messages keep checking the connections to each peer to see if there is any new transaction is happening. Let this screen runs itself and move to the next step

**\*\* To move to next section, open another terminal and login to your LinuxONE server**

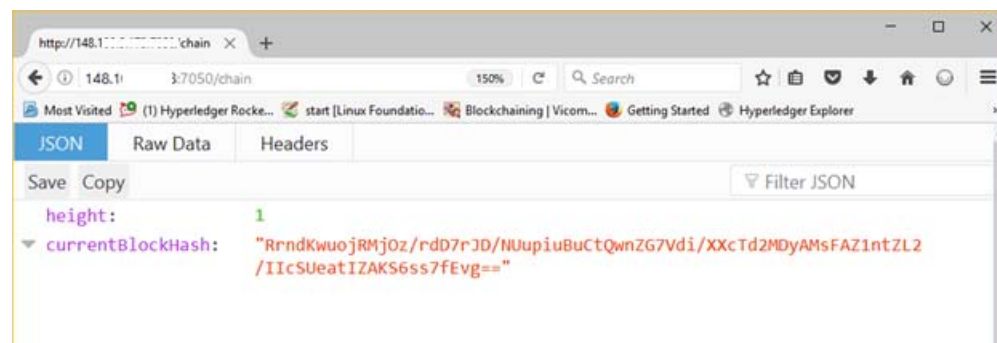
There are many ways to verify if the Hyperledger fabric is running okay. For this lab, you can either use a web browser or command shell to verify it.

#### 1.4 Testing and verifying your local network using Web Browser

Let’s verify if the Hyperledger fabric is working with a web browser. Hyperledger Fabric’s peer has a RestAPI port, which is normally mapped to 7050. Open a web browser and type following in the address section.

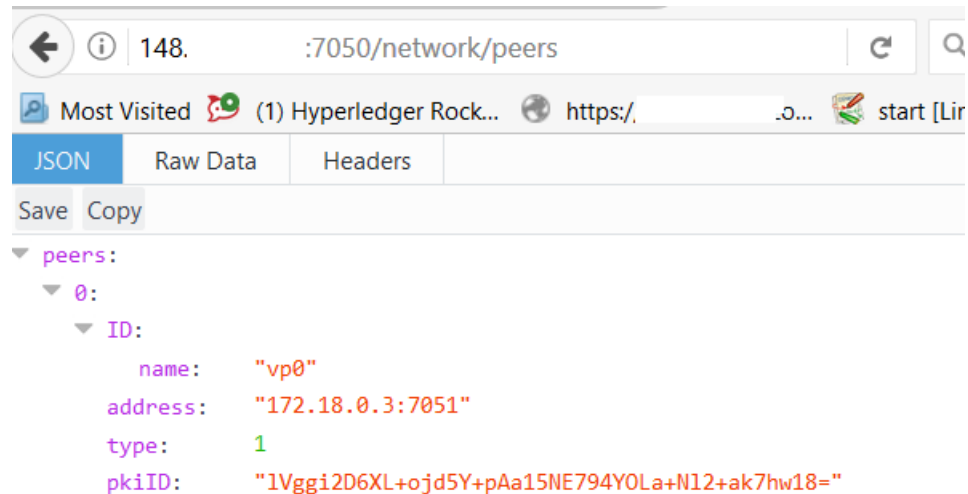
**<your LinuxONE server IP>:7050/chain**

You will see something like following screen:



What this means is currently Hyperledger fabric created the genesis block (initial block of the blockchain) with initial hash value.

Now below you will see how peers are registered in the Hyperledger Fabric, and their address/port, peer type and cryptographic ID of the peer by typing `<your LinuxONE server IP>:7050/network/peers`



```
peers:
  0:
    ID:
      name: "vp0"
      address: "172.18.0.3:7051"
      type: 1
      pkiID: "1Vggi2D6XL+ojd5Y+pAa15NE794YOLa+N12+ak7hw18="
```

To get more information about how to use pre-defined REST APIs for Hyperledger peers, please visit

<http://hyperledger-fabric.readthedocs.io/en/stable/API/CoreAPI/#rest-api>

\*\* To go to section #2, please open another terminal window and connect to LinuxONE shell and go to next page.

# [2] Install/run Hyperledger Fabric Composer Playground

Following steps will guide you to install/run Hyperledger Fabric Composer Playground(composer-playground) app for on your server that fabric is running. Composer-playground app is node.js based web tool that enables a developer to create Hyperledger applications(chaincode), related access controls, and combine them with business logics relatively simple way. For more information, visit <https://hyperledger.github.io/composer/>.

## 2.1 Map port 8080 to open and listen

Since we are using Docker container for the app and it connects to a local Hyperledger fabric also on docker containers, we will make composer-playground app to use host network and interface external ip address/port. To enable it we will need to enter following iptables command. If you like to know more about why please refer to this web site ( <http://www.dasblinkenlichten.com/docker-networking-101-host-mode/> )

Make sure you are still in 'root' id (sudo su), and execute following command:

```
sudo iptables -I INPUT -p tcp -m tcp --dport 8080 -j ACCEPT
```

```
[linux1@alexrhel docker-compose]$  
[linux1@alexrhel docker-compose]$ sudo iptables -I INPUT -p tcp -m tcp --dport 8080 -j ACCEPT  
[linux1@alexrhel docker-compose]$  
[linux1@alexrhel docker-compose]$
```

## 2.2 Run composer-playground app using docker run

2.2.1) Now enter following command to download already installed composer-playground app from the docker hub, and invoke webservice.

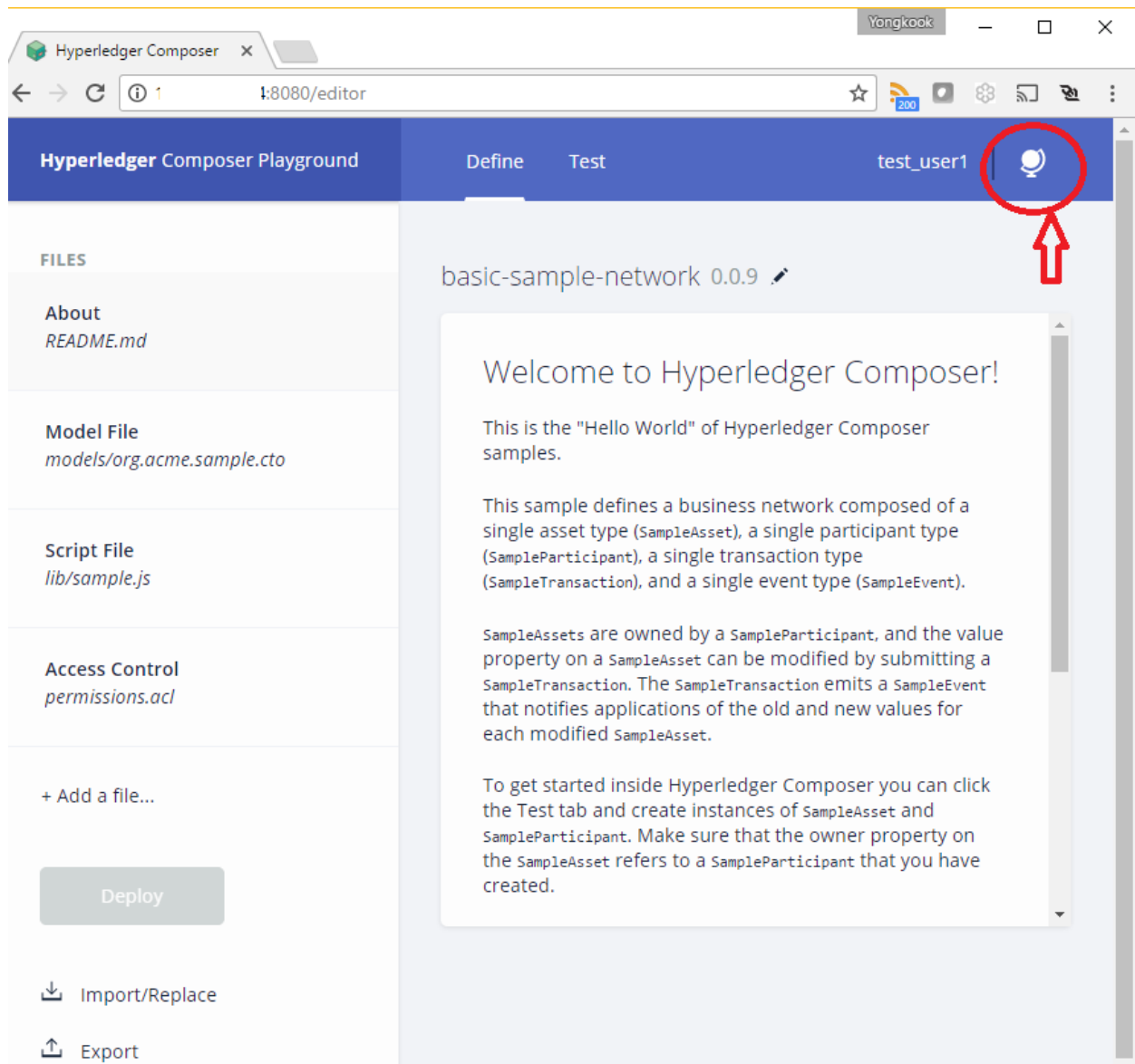
From any directory enter:

```
sudo docker run --net=host -it likepunk/composer_playground:v075_s390 node cli
```

(note for two dashes for --net)

```
[linux1@alexrhel docker-compose]$  
[linux1@alexrhel docker-compose]$  
[linux1@alexrhel docker-compose]$ docker run --net=host -it likepunk/composer playground:v075_s390 node cli  
PlaygroundAPI      :createServer()      > 8080  
ConnectionProfileManager :constructor()      Created a new ConnectionProfileManager {"fs":{"constants":{"O_RDONLY":0,"O_WRONLY":1,"O_RDWR":2,"S_IFMT":61440,"S_IFREG":32768,"S_IFDIR":16384,"S_IFCHR":8192,"S_IFBLK":24576,"S_IFIFO":4096,"S_IFLNK":40960,"S_IFSOCK":49152,"O_CREAT":64,"O_EXCL":128,"O_NOCTTY":256,"O_TRUNC":512,"O_APPEND":1024,"O_DIRECTORY":65536,"O_NOATIME":262144,"O_NOFOLLOW":131072,"O_SYNC":1052672,"O_DIRECT":16384,"O_NONBLOCK":2048,"S_IRWXU":448,"S_IRUSR":256,"S_IWUSR":128,"S_IXUSR":64,"S_IRWXG":56,"S_IRGRP":32,"S_IWGRP":16,"S_IXGRP":8,"S_IRWXO":7,"S_IROTH":4,"S_IWOTH":2,"S_IXOTH":1,"F_OK":0,"R_OK":4,"W_OK":2,"X_OK":1},"F_OK":0,"R_OK":4,"W_OK":2,"X_OK":1}}  
PlaygroundAPI      :createServer()      Playground API started on port 8080  
PlaygroundAPI      :createServer()      <  
Composer           :main()              >
```

2.2.2) Bring up a web-browser and enter <your LinuxONE IPAddress:8080> in the address. You should be able to see following screen. Click a globe shaped icon at upper right corner. It will pop up a window for connection profiles settings. Choose Hyperledger Fabric v0.6 at this time(v1.0 is not working with this version of composer-playground).



Hyperledger Composer x

3080/profile

Hyperledger Composer Playground Define Test test\_user1

### Import/Create a Connection Profile

Upload a Connection Profile from your computer

Drop here to upload or [browse](#)

Or create a connection profile, select your Blockchain platform:

- Hyperledger Fabric v0.6**  
v0.6 Connection Profile.
- Hyperledger Fabric v1.0**  
v1.0 Connection Profile.

Cancel Add

Advanced

Deploy Wait Time (seconds)	500
Invoke Wait Time (seconds)	100

Use this profile Export Connection Profile

2.2.3) Enter each section as shown in the screenshot below. Pick a connection profile name. Leave Peer URL, Membership Services URL, Event Hub URL as they are. Change 'Key Value Store Directory Path' into './'(dot slash).

Change Deploy Wait Time and Invoke Wait Time into 500, 100 each.

The screenshot shows the Hyperledger Composer Playground interface. The top navigation bar includes 'Hyperledger Composer Playground', 'Define', 'Test', and a user profile 'admin'. The left sidebar is titled 'CONNECTION PROFILES' and contains 'Web browser' and '+ Import or Create a Profile'. The main content area is titled 'Basic Configuration' and contains the following fields:

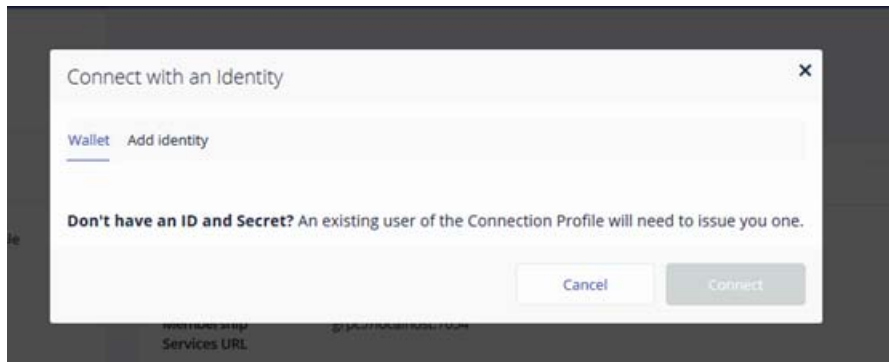
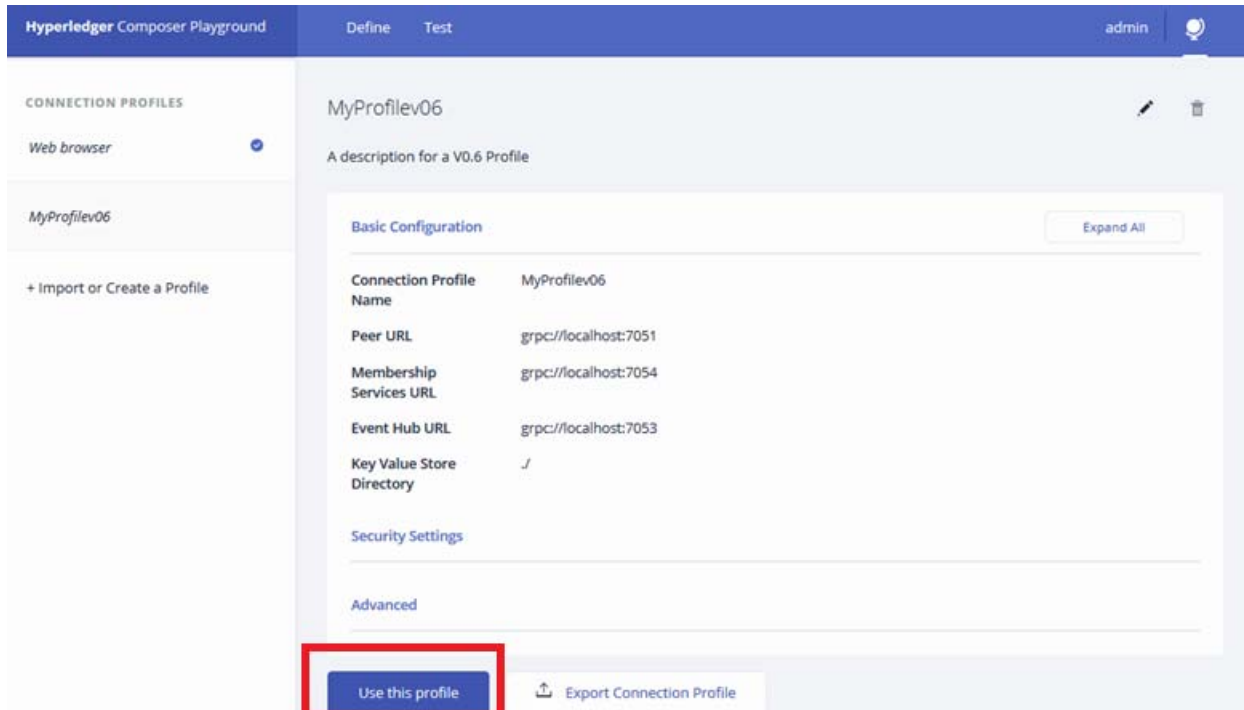
Field	Value
Connection Profile Name*	MyProfilev06
Profile Description	A description for a V0.6 Profile
Peer URL*	grpc://localhost:7051
Membership Services URL*	grpc://localhost:7054
Event Hub URL*	grpc://localhost:7053
Key Value Store Directory Path*	./

Below the 'Basic Configuration' section is the 'Security Settings' section, which includes 'Certificate' and 'Certificate Path' fields. At the bottom is the 'Advanced' section, which includes 'Deploy Wait Time (seconds)' set to 500 and 'Invoke Wait Time (seconds)' set to 100.

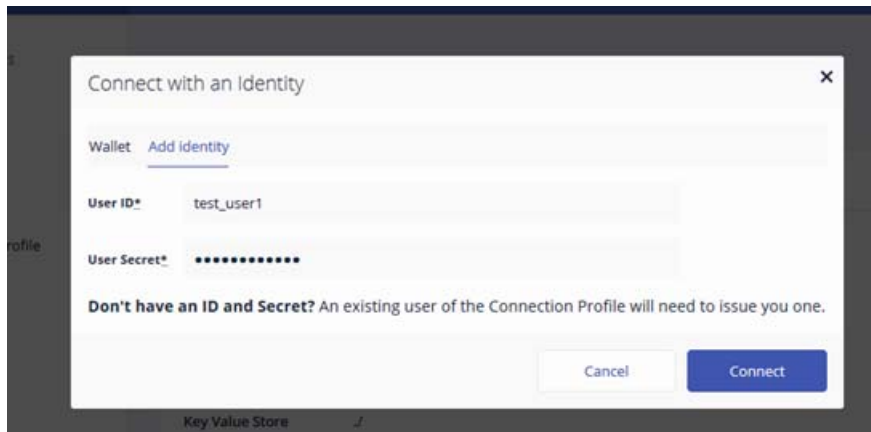
2.2.4) Once Saved, you can click 'Use this Profile' button. It will ask you to connect with an identity. Click 'Add identity' for now and enter following value ID/SECRET, and then click 'Connect'.

ID: test\_user1

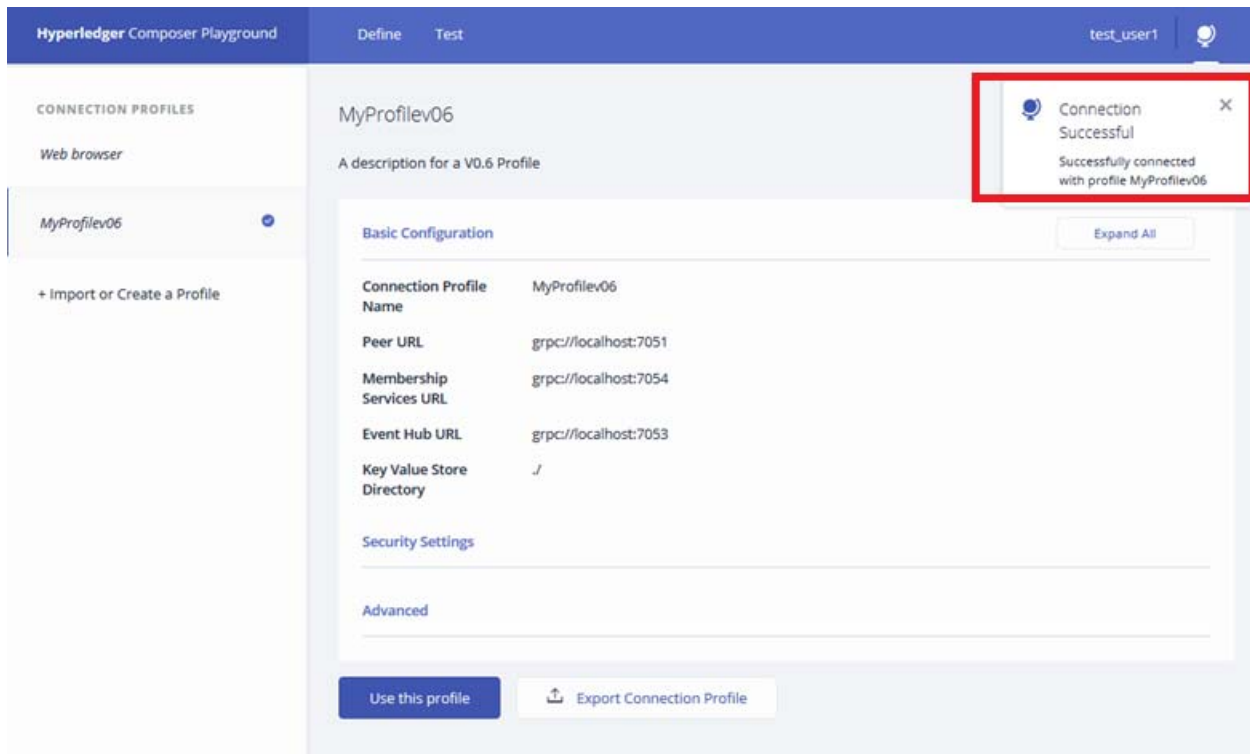
SECRET: jGINI6ImkuDo







You should see a pop-up messages saying it's trying to deploy business network files and once it's done it will say 'Connection Successful' in upper-right corner.



**Now you're ready to move with the Hyperledger Composer Lab. Bring up Hyperledger composer playground lab PDF( <http://bit.ly/2t1h5jE> ) and jump to section 1.2. NOTE: While you're doing this lab, try to examine blockchain contents using browser that was explained in section 1.4. and another example shown below to check actual contents of blockchain data.**



# This is a bonus lab – in case you have extra time left, feel free to try it

To go to section #3, please open another terminal window and connect to LinuxONE shell and begin the lab.

## [3] Running Marbles App

Following steps will guide you to install/run Marbles app demo for Hyperledger on your server. You can install Marbles app either locally or at the IBM Bluemix easily with instruction described in github below, but to minimize errors we will provide you an docker image already integrated with all components needed. <https://github.com/ibm-blockchain/marbles/tree/v2.0>

(if you have completed section [2] and removed all the docker containers in cache, go to section [1] and run fabric network then come back to this step)

Assuming you completed the first lab successfully, we will deploy a sample app called 'Marbles'. It's a very intuitive GUI based blockchain app to create/trade marbles. Marbles represent assets, and whenever an asset gets created and traded, new blocks will be created and shown in the console. You can ask your peers to connect to your website and simulate multi-user based blockchain network communication.

FYI, our implementation of Marbles app is a bit different from the original IBM Marbles. We(Vicom Infinity) modified GUI so that we can simulate four-peer blockchain network.

Again, this is the lab based on Linux for IBM z Systems™ or IBM LinuxONE™, so the binary file for Docker image is for s390x architecture. It was tested with SLES12 SP2 as well as RHEL7.2. If you like to get a Docker image for x86, please contact us.

### 3.1 Map port 3000 to open and listen

Since we are using Docker container for the app and it connects to a local Hyperledger fabric also on Docker containers, we will make marbles app to use host network and interface external ip address/port. To enable it we will need to enter following iptables command. If you like to know more about why please refer to this web site ( <http://www.dasblinkenlichten.com/docker-networking-101-host-mode/> )

)

Make sure you are still in 'root' id (sudo su), and execute following command:

```
iptables -I INPUT -p tcp -m tcp --dport 3000 -j ACCEPT
```

```
[root@openmf ~]# iptables -I INPUT -p tcp -m tcp --dport 3000 -j ACCEPT
```

### 3.2. Run marbles app using Docker run

Now enter following command to download already installed marbles app from the docker hub, and invoke web services.

From any directory enter: `docker run --net=host likepunk/marbles` (note for two dashes for -net)

```
# docker run --net=host likepunk/marbles
```

You will see following start-up logs. Wait until you see a message like `---Websocket Up---`

```

----- docker run --net=host likepunk/marbles
cache busting hash js e95f0ccb21d36cb304146ff3811ee802 css 47323c12c084bce58826a7bbc437c584
----- Server Up - localhost:3000 -----
-----
Running using Developer settings
- Tracking Deployment
loading hardcoded peers
loading hardcoded users
[ibc-js] Peer: vp0-vp0...:7050
[ibc-js] Registering vp0-vp0...:7050 w/enrollId - test_user0
[ibc-js] Registration success x1 : test_user0
[ibc-js] removing temp dir
[ibc-js] Downloading zip
redirect... https://codeload.github.com/IBM-Blockchain/marbles/zip/v2.0
[ibc-js] Downloading zip
[ibc-js] Unzipping zip
[ibc-js] Unzip done
[ibc-js] Scanning files [ 'marbles_chaincode.go' ]
[ibc-js] Parsing file for shim version
[ibc-js] Found shim version: github.com/hyperledger/fabric/core/chaincode/shim
[ibc-js] Parsing file for invoke functions - marbles_chaincode.go
[ibc-js] Found cc invoke function: init
[ibc-js] Found cc invoke function: delete
[ibc-js] Found cc invoke function: write
[ibc-js] Found cc invoke function: init_marble
[ibc-js] Found cc invoke function: set_user
[ibc-js] Found cc invoke function: open_trade
[ibc-js] Found cc invoke function: perform_trade
[ibc-js] Found cc invoke function: remove_trade
[ibc-js] Parsing file for query functions - marbles_chaincode.go
[ibc-js] Found cc query function: read
[ibc-js] load_chaincode() finished
[ibc-js] Deploy Chaincode - Starting
[ibc-js]         function: init , arg: [ '99' ]

Waiting...

deploy success [waiting another 50 seconds]
16e655c0fce6a9882896d3d6d11f7dcd4f45027fd4764004440ff1e61340910a9d67685c4bb723272a497f3cf
428e6cf6b009618612220e1471e03b6c0aa76cb

```

```

deploy success [waiting another 50 seconds]
16e655c0fce6a9882896d3d6d11f7dcd4f45027fd4764004440ff1e61340910a9d67685c4bb723272a497f3cf
428e6cf6b009618612220e1471e03b6c0aa76cb

[ibc-js] Deploy Chaincode - Complete
[preflight check] 1 : testing if chaincode is ready
[ibc-js] read - success: { jsonrpc: '2.0',
  result: { status: 'OK', message: '["rb2264f"]' },
  id: 1493354476022 }
[preflight check] 1 : success
----- Websocket Up -----
[ibc-js] Chain Stats - success

```

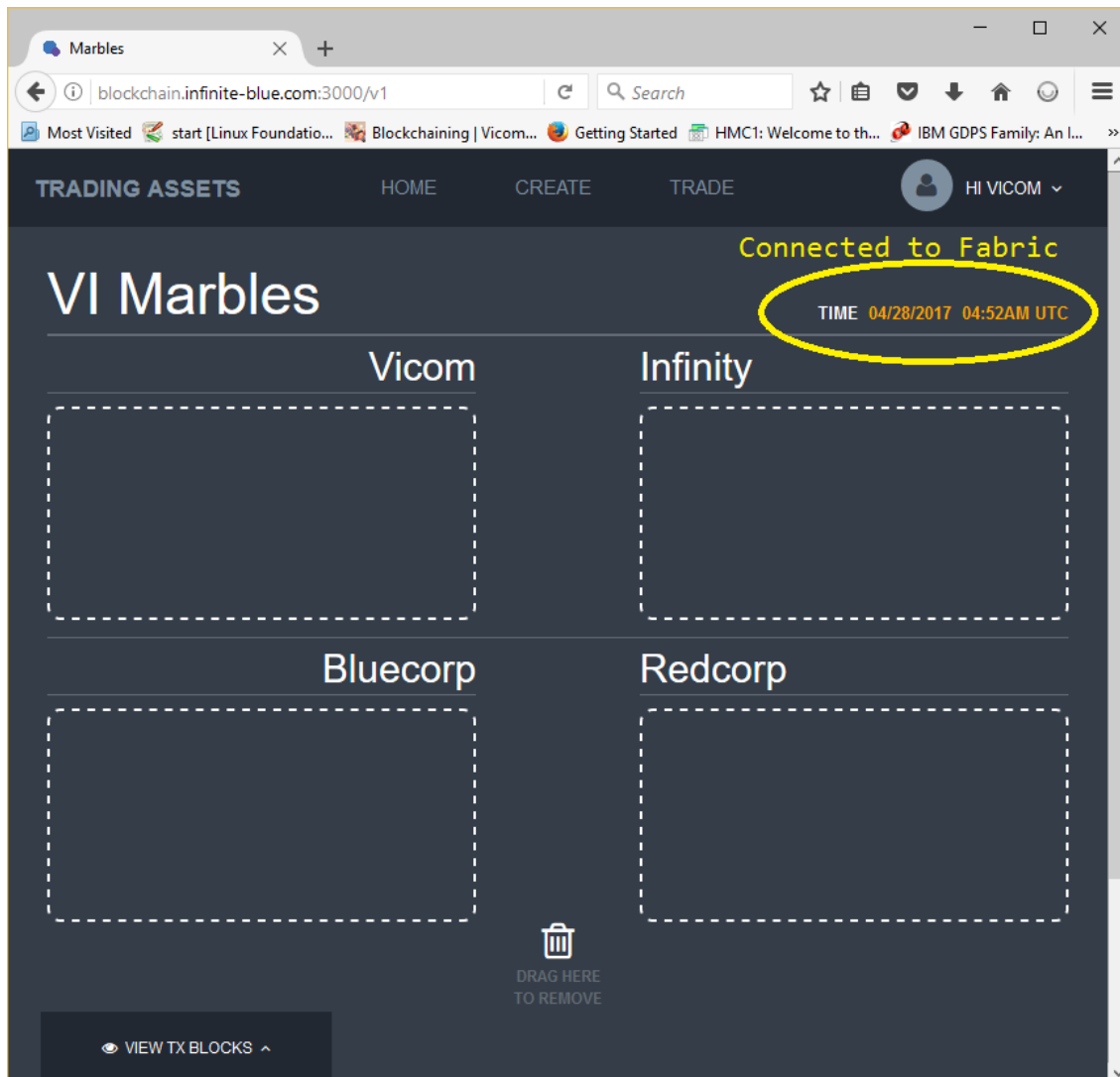
### 3.3. Play Marbles!

Now bring any web browser and type IP address of your server with port number 3000. If you don't know your host server ip, then type following: `ip addr show | grep eth0`

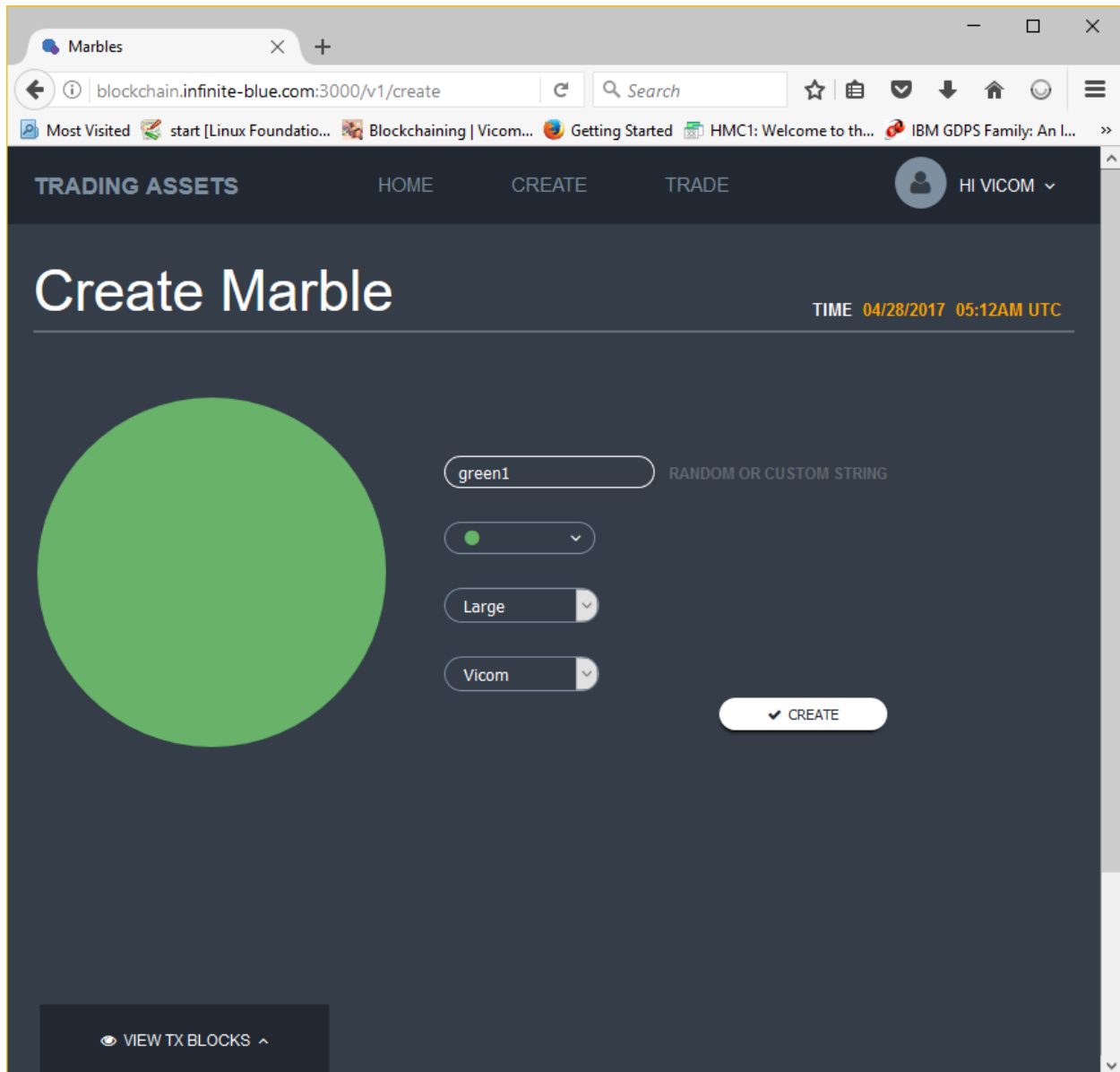
```
bctest04:~ # ip addr show|grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 148.100.5.173/23 brd 148.100.5.255 scope global eth0
55: veth0361901: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-c706eaf4883e state UP group default
bctest04:~ #
```

Now connect to your marbles app using the ip shown above with port 3000 at the browser:

<http://<your LinuxONE's IP address>:3000> and you will see Marbles web page shows as following:



Click on 'CREATE' on the center then create a marble asset of your choice



You can see a large green marble gets created, at the same time a new block is added to the blockchain. You can see it in action by clicking 'VIEW TX BLOCKS'. It will show you details of the block(chaincode ID, payload etc).

Marbles

blockchain.infinite-blue.com:3000/v1/home

Search


Most Visited start [Linux Foundatio... Blockchaining | Vicom... Getting Started HMC1: Welcome to th... IBM GDPS Family: An I... >>

TRADING ASSETS HOME CREATE TRADE HI VICOM

# VI Marbles

TIME 04/28/2017 05:15AM UTC

## Vicom



## Infinity

## Bluecorp

## Redcorp

VIEW TX BLOCKS ^

002 003 004 005 006 007 008 009 010 011

**Block Height: 10**

---

Created: 04-28-2017 05:15am UTC  
 UUID: undefined  
 Type: 2  
 CC ID: 16e655c0fce6a9882896d3d6d11f7dcd4f45027fd4764004440  
 ff1e61340910a9d67685c4bb723272a497f3cf428e6cf6b009618612220  
 e1471e03b6c0aa76cb  
 Payload: init:\_marble green1 green 35 vicom

009 010 011

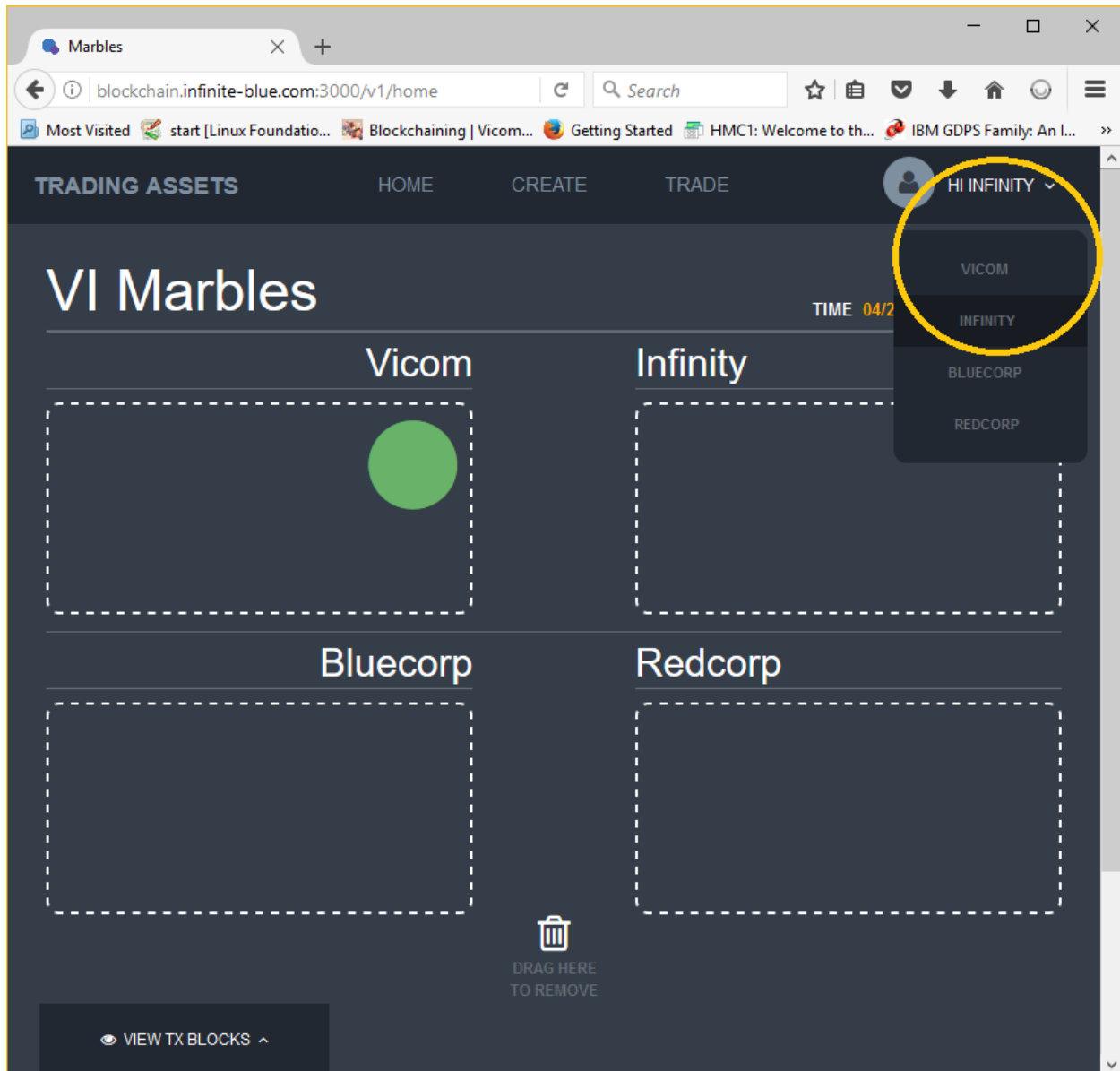
**Block Height: 10**

---

Created: 04-28-2017 05:15am UTC  
 UUID: undefined  
 Type: 2  
 CC ID: 16e655c0fce6a9882896d3d6d11f7dcd4f45027fd4764004440  
 ff1e61340910a9d67685c4bb723272a497f3cf428e6cf6b009618612220  
 e1471e03b6c0aa76cb  
 Payload: init:\_marble green1 green 35 vicom



Now try to switch ID to INFINITY(or ask other people to connect using another browser/PC or mobile device), and create marbles. You can move marbles by drag&drop.



**Now you're done with the Lab! You can bring down all the docker images before next Lab by typing following:**

```
# docker stop $(docker ps -a -q)
```

```
# docker rm -f $(docker ps -a -q)
```

Explorer further more with trading menu. To get more information about it please see below github site. There is also a newer version of Marbles app works with Hyperledger Fabric v1.0.

<https://github.com/ibm-blockchain/marbles/tree/v2.0>

## [x] Extra Information/Debugging

If you experience an error that complains about Chaincode not being deployed in the Hyperledger fabric logs or Marbles app showing it can't find the Hyperledger fabric, try insert following into 'four-peer-ca.yaml' file under **fabric-images/docker-compose** :

- CORE\_CHAINCODE\_DEPLOYTIMEOUT=500000

```
version: '2'
services:
  baseimage:
    build: ./baseimage
    image: hyperledger/fabric-baseimage:latest

  membersrv:
    image: ibmblockchain/fabric-membersrv:${ARCH_TAG}
    extends:
      file: base/membersrv.yaml
      service: membersrv

  vp0:
    image: ibmblockchain/fabric-peer:${ARCH_TAG}
    extends:
      file: base/peer-secure-pbft-base.yaml
      service: peer-secure-pbft-base
    ports:
      - "7050:7050"
      - "7051:7051"
      - "7053:7053"
    environment:
      - CORE_PEER_ID=vp0
      - CORE_SECURITY_ENROLLID=test_vp0
      - CORE_SECURITY_ENROLLSECRET=MwYpmSRjupbT
      - CORE_CHAINCODE_DEPLOYTIMEOUT=500000
    links:
      - membersrv

  vp1:
    image: ibmblockchain/fabric-peer:${ARCH_TAG}
    extends:
      file: base/peer-secure-pbft-base.yaml
      service: peer-secure-pbft-base
    ports:
```

Once edited and saved, stop prior docker processes and remove all of the containers from cache by entering:

```
# docker stop $(docker ps -a -q)
```

```
# docker rm -f $(docker ps -a -q)
```

This is very useful command whenever you want to start from scratch. You want to make sure there's no residues from prior runs.

After you get the prompt back, type following commands to bring up the Hyperledger fabric as well as run marbles demo(you will need two shell screens).

```
# docker-compose -f four-peer-ca.yaml up
```

```
# docker run --net=host likepunk/marbles
```