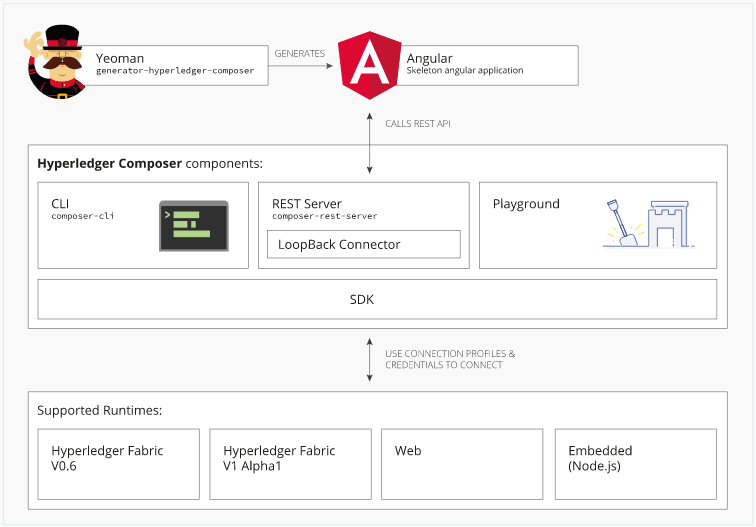
Hyperledger Composer development setup

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier.

These are the components we are going to setup.



Composer Solution Architecture with 8 steps.

**Step 1**: Installing Hyperledger Composer development tools.

**Step 2**: Starting Hyperledger Fabric.

**Step 3**: Create a Business Network Definition.

**Step 4:** Building .bna (banana file) or Generate the Business Network Archive.

**Step 5:** Deploy banana file to the running Hyperledger Fabric.

**Step 6:** Generate REST API.

**Step 7:** Test REST API.

**Step 8**: Generate angular application and test.

Follow the instructions below to get the required Hyperledger Composer development tools and stand up a Hyperledger Fabric.

There are two version of Hyperledger Fabric: v0.6 and v1.0. The default is for **v1.0** and we suggest this is the one you use.

Before you begin, following are prerequisites for installing the required development tools:

Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12

Docker Engine: Version 17.03 or higher

Docker-Compose: Version 1.13 or higher

Node: 6.x (note version 7 is not supported)

npm: v3.x or v5.v

git: 2.9.x

Python: 2.7.x

**NOTE 1:** Please do not install Composer as a ***superuser*** - or use **'*sudo*'** or the **'*root*'** user, if on Linux (doing will cause issues with the install). Composer should be **installed as non-privileged ($) user**.

**NOTE 2:** Make sure you have open internet access in order to execute the below commands. If no open internet access you will end up with SSL Cert issues while executing some of the https curl URLs within the scripts.

**NOTE 3**: Assuming CURL is already installed in your Linux machine.

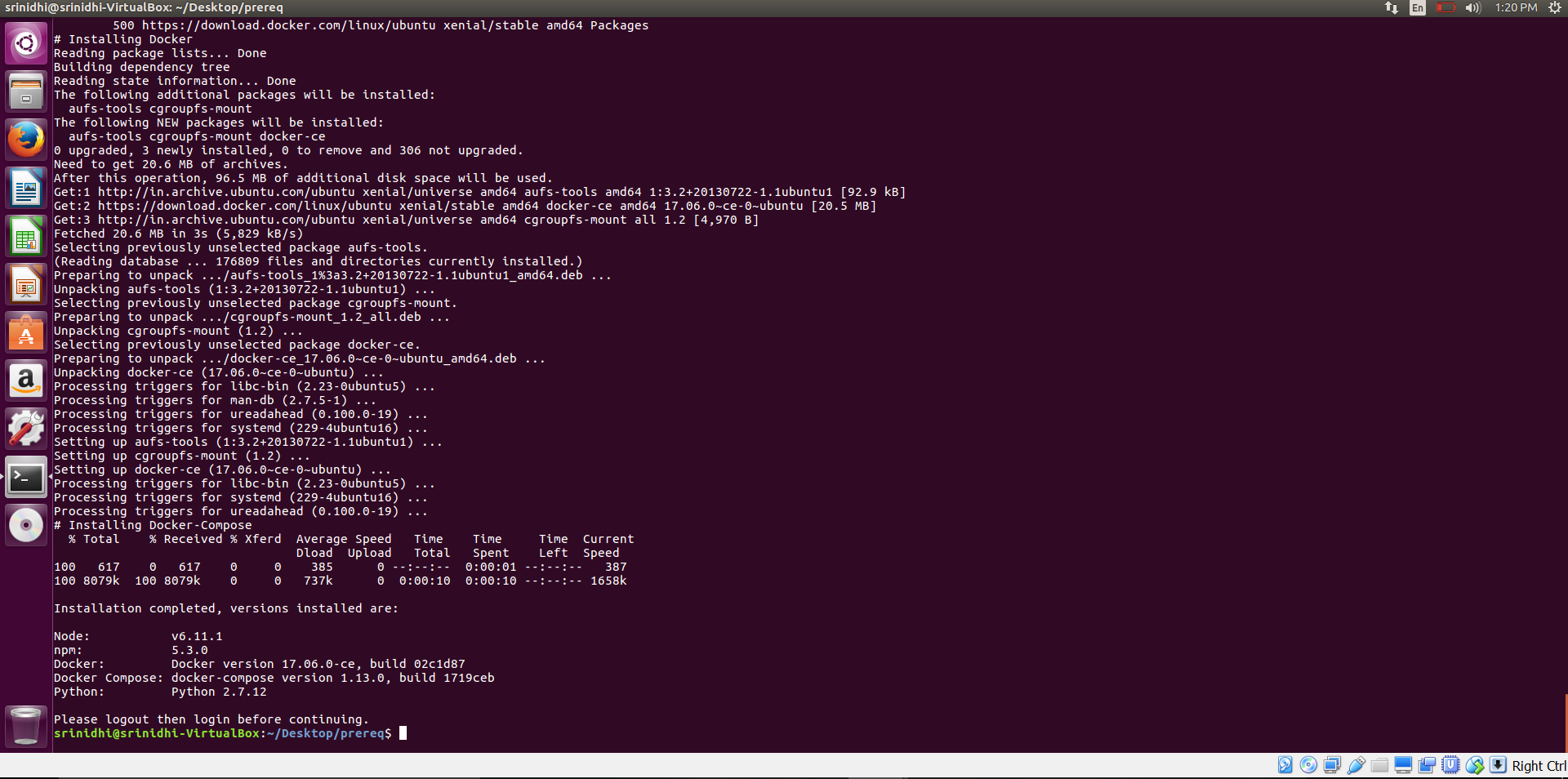
If you're running on Ubuntu, you can download the prerequisites using the following commands:

curl -O https://hyperledger.github.io/composer/prereqs-ubuntu.sh

chmod u+x prereqs-ubuntu.sh

Next run the script - as this briefly uses sudo during its execution, you will be prompted for your password.

./prereqs-ubuntu.sh



**Step 1**: Installing Hyperledger Composer development tools

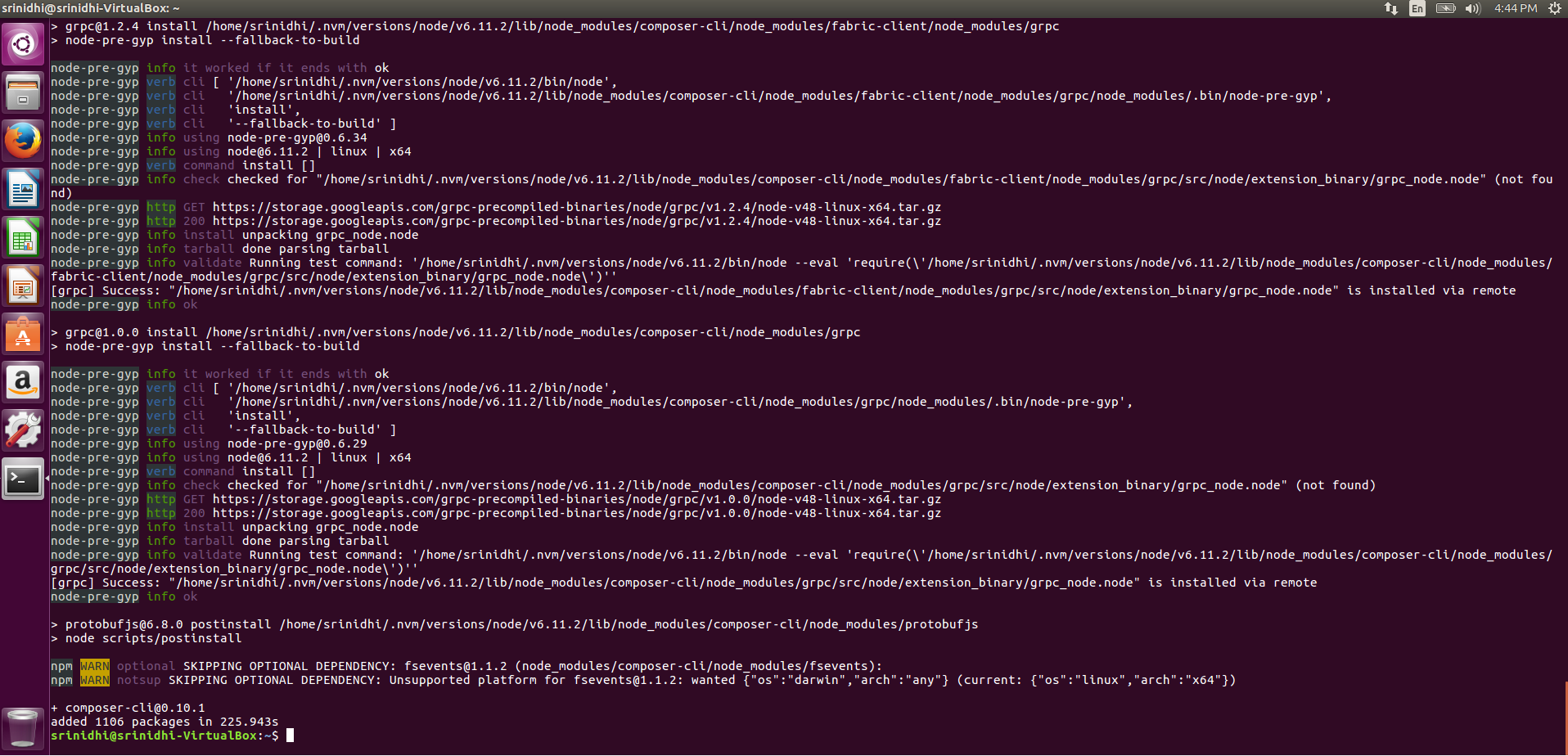
The development tools you'll need can all be installed (as a non-privileged user e.g. non-root) as mention in **NOTE 1**

1. **composer-cli** – contains all the command line operations for developing business networks.

It interprets the composer commands in building and deploying the networks.

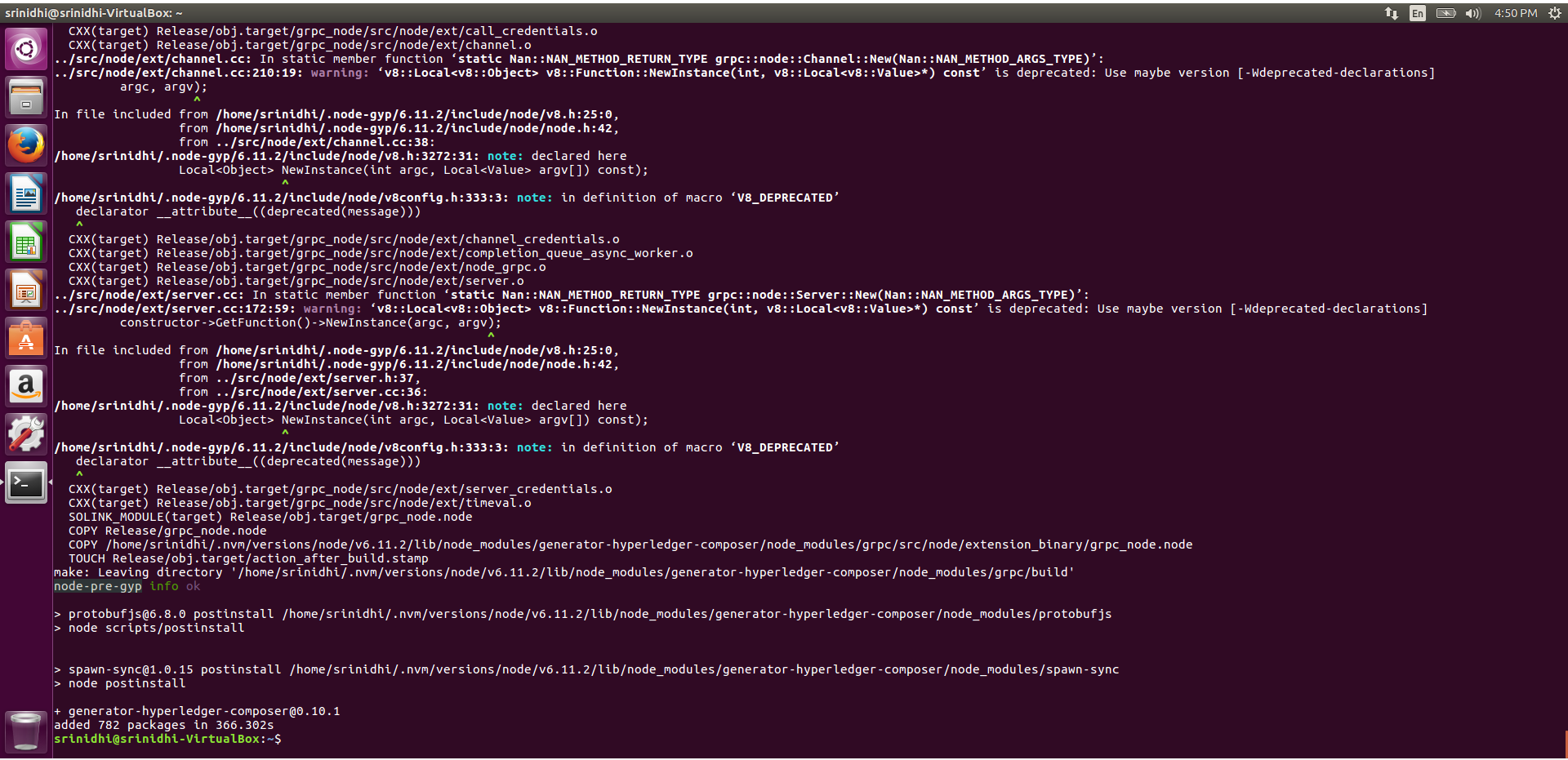
To install composer-cli run the following command (check above architecture for components overview).

***npm install –g composer-cli***



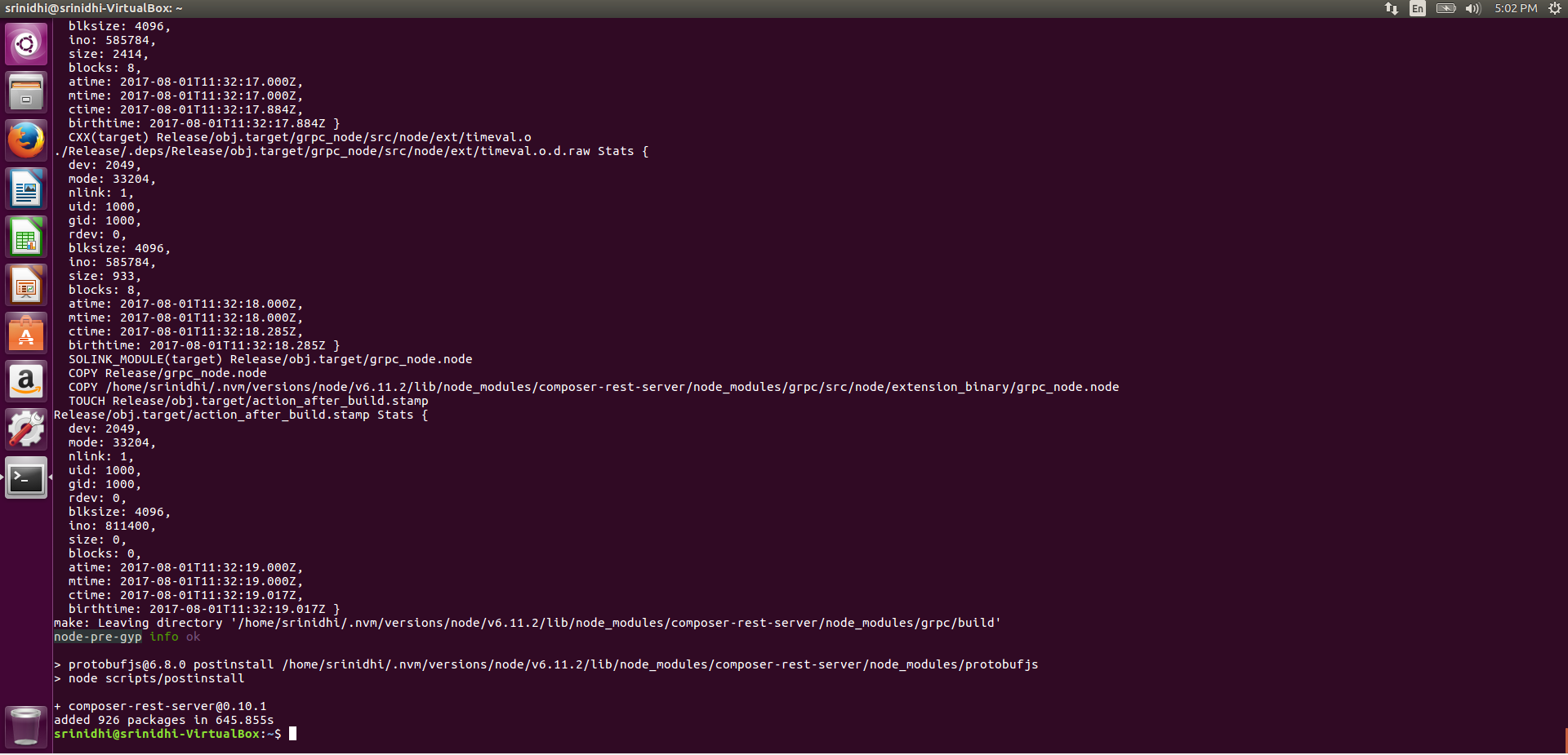
1. **generator-hyperledger-composer-** It is a Yeoman plugin that creates skeleton applications for your business network.

***npm install -g generator-hyperledger-composer***



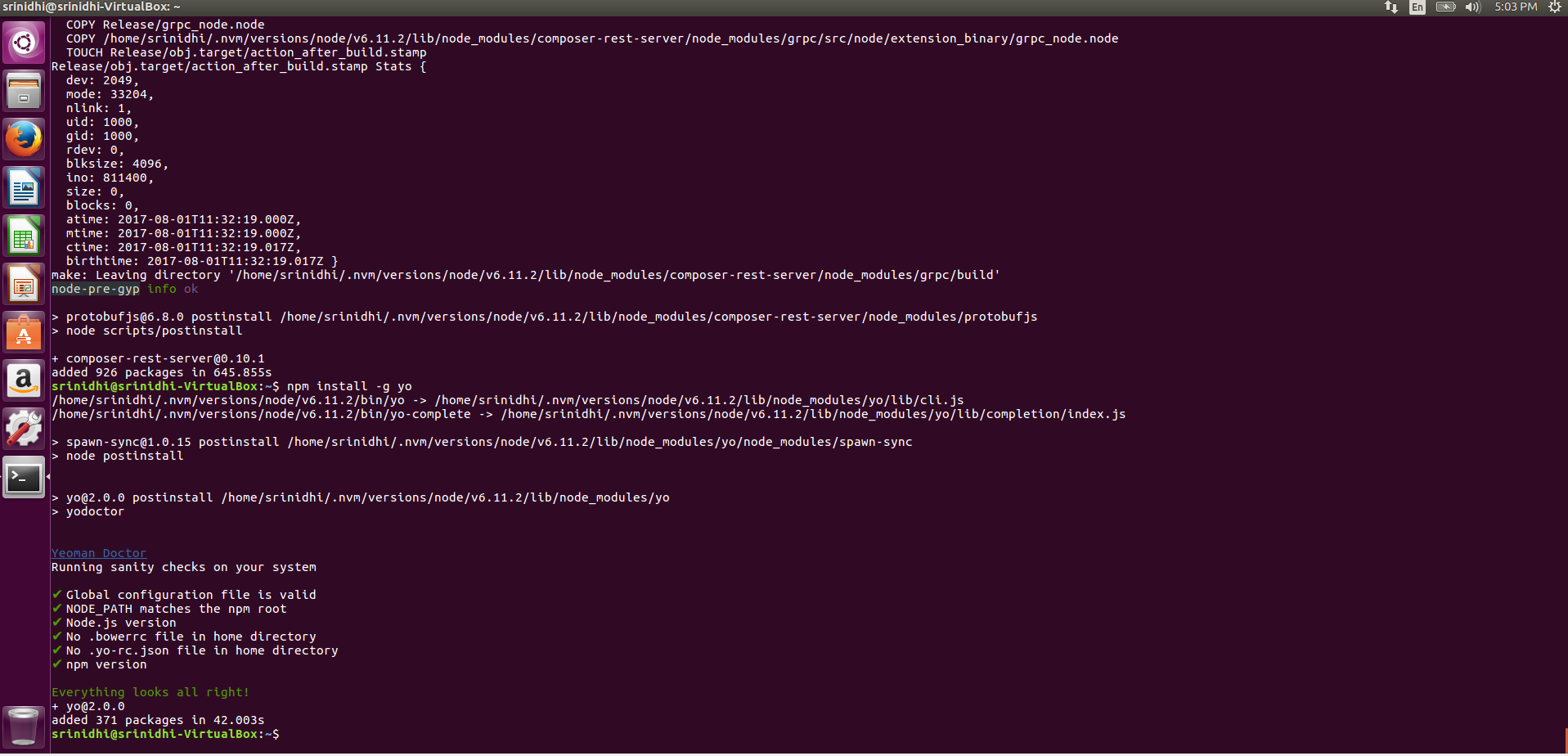
1. **composer-rest-server -** uses the Hyperledger Composer loopback connector to connect to a business network, extract the models and then present a page containing the REST APIs that have been generated for the model.

***npm install -g composer-rest-server***



1. ***yo*** - Yeoman is a tool for generating applications. When combined with the generator-hyperledger-composer component, it can interpret business networks and generate applications based on them.

***npm install -g yo***



***This completes the hyperledger composer components installation for development***

**Step 2**: Starting Hyperledger Fabric.

**Note:** You have to run these as sudo user and have open internet access. If not you will end up with certificates issue.

1. In a directory of your choice (will assume ~/fabric-tools) get the zip file that contains the tools.

This will set up fabric dev servers in your box.

mkdir ~/fabric-tools && cd ~/fabric-tools

curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.zip

unzip fabric-dev-servers.zip

1. Choose which version of Fabric to use. Hyperledger Fabric v1.0 is highly recommended and the default. If for some reason v0.6 needs to be installed, you can set it explicitly as follows: **export FABRIC\_VERSION=hlfv0.6. Don’t change anything for 1.0 (Just ignore this step b)**
2. Start fabric - All the scripts will be in the directory **~/fabric-tools** A typical sequence for Hyperledger Composer use would be

cd ~/fabric-tools

./downloadFabric.sh

./startFabric.sh

./createComposerProfile.sh

Then at the end of your development session

**cd ~/fabric-tools**

**./stopFabric.sh**

**./teardownFabric.sh**

**Step 3**: Create a Business Network Definition.

1. Download sample network using below git command.

git clone https://github.com/hyperledger/composer-sample-networks.gitnetworks.git

1. Rename this folder to “my-network” using the below command.

***cp -r ./composer-sample-networks/packages/basic-sample-network/ ./my-network***

1. You should now have a folder called my-network (as the basis for our project) that we can start to modify.
2. Update your **package.json** file - The metadata (name, version, description) for the business network definition is stored in the package.json file. Edit the file to change the name to my-network and modify the prepublish script to change the name of the business network archive.

**NOTE :** Update only the bold tags below. Keep rest of the things same.

"**name**": "my-network",

"**version**": "0.0.1",

"**description**": "My very first Hyperledger Composer Network",

"**scripts**": {

"prepublish": "mkdirp ./dist && composer archive create --sourceType dir --sourceName . -a ./dist/my-network.bna",

"pretest": "npm run lint",

"lint": "eslint .",

"postlint": "npm run licchk",

"licchk": "license-check",

"postlicchk": "npm run doc",

"doc": "jsdoc --pedantic --recurse -c jsdoc.conf",

"test-inner": "mocha --recursive && cucumber-js",

"test-cover": "nyc npm run test-inner",

"test": "mocha --recursive -t 4000"

}

1. Update your README.md file to - **# My very first Hyperledger Composer Network.**

Save your changes.

1. Define your Domain Model - overwrite below code and save your changes to **models/sample.cto**

/\*\*

\* My commodity trading network

\*/

namespace org.acme.mynetwork

asset **Commodity** identified by tradingSymbol {

o String tradingSymbol

o String description

o String mainExchange

o Double quantity

--> Trader owner

}

participant **Trader** identified by tradeId {

o String tradeId

o String firstName

o String lastName

}

transaction **Trade** {

--> Commodity commodity

--> Trader newOwner

}

1. Write Transaction Processor Functions – overwrite below code and save your changes to **lib/sample.js ()**

function **tradeCommodity**(trade) {

trade.commodity.owner = trade.newOwner;

return getAssetRegistry('org.acme.mynetwork.Commodity')

.then(function (assetRegistry) {

return assetRegistry.update(trade.commodity);

});

}

1. Update your Access Control Rules - The file permissions.acl defines the access control rules for the business network definition. Replace the entire contents of permissions.acl with the rule below and save your changes to permissions.acl

/\*\*

\* Access control rules for mynetwork

\*/

rule Default {

description: "Allow all participants access to all resources"

participant: "ANY"

operation: ALL

resource: "org.acme.mynetwork.\*"

action: ALLOW

}

**Step 4**: Building .bna or Generate the Business Network Archive.

To check that the structure of the files is valid, you can now generate a Business Network Archive (BNA) file for your business network definition. The BNA file is the deployable unit -- a file that can be deployed to the Composer runtime for execution.

1. Switch back to the terminal and type:

cd my-network

npm install

You should see the following output:

my-network@0.0.1 prepublish /home/user/my-network

mkdirp ./dist && composer archive create --sourceType dir --sourceName . -a ./dist/my-network.bna

Creating Business Network Archive

Looking for package.json of Business Network Definition in /home/user/my-network

Found:

Description:My very first Hyperledger Composer Network

Name:my-network

Identifier:my-network@0.0.1

Written Business Network Definition Archive file to ./dist/my-network.bna

Command completed successfully.

Command succeeded

**Step 5:** Deploy to the running Hyperledger Fabric.

We are going to deploy the BNA (suffix .bna) file to Hyperledger Fabric v1.0; this blockchain environment was set up earlier in this guide in **Step 2**.

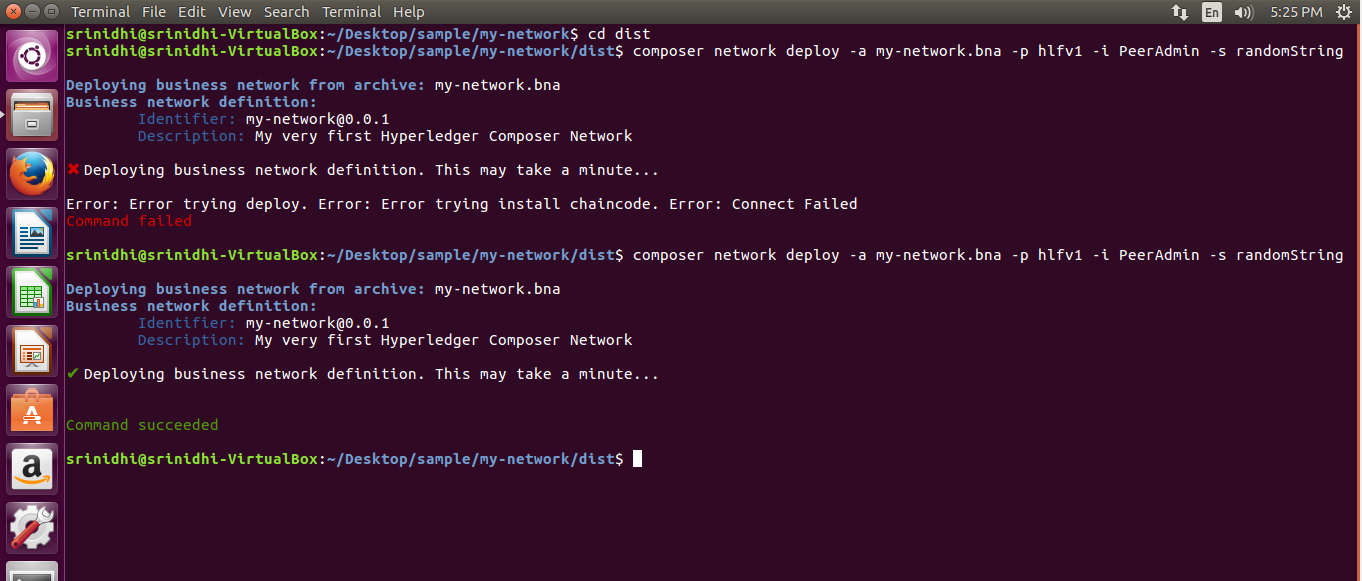
Switch to the terminal, change directory to the dist folder containing the my-network.bna file and type:

cd dist

composer network deploy -a my-network.bna -p hlfv1 -i PeerAdmin -s randomString

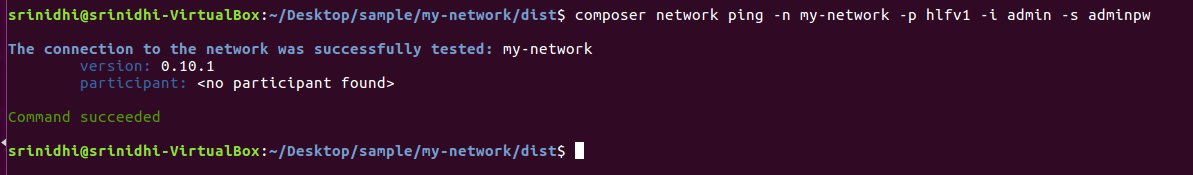
**NOTE** : ***PeerAdmin*** is purely an identity label referencing cryptographic (eg. signer certificate) material you have imported into the keyValStore identified in your connection profile which has meaning to a ***single Peer*** (in our dev setup) or set of Peers (if configured in another way) within the same organization.

After approximately 30 seconds or so, the business network should have been deployed to your local Hyperledger Fabric. You should see output as follows:



You can verify that the network has been deployed by typing:

composer network ping -n my-network -p hlfv1 -i admin -s adminpw



**Step 6**: Generate REST API

To integrate with the deployed business network (creating assets/participants and submitting transactions) we can either use the Composer Node SDK or we can generate a REST API.

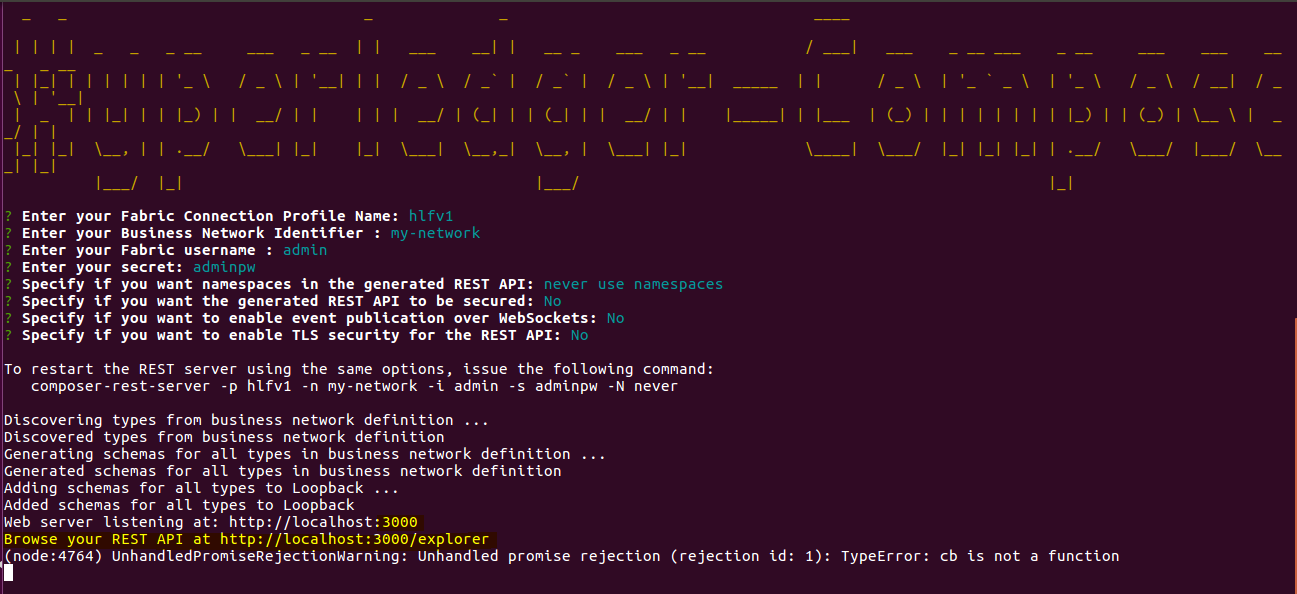
To create the REST API we need to launch the composer-rest-server and tell it how to connect to our deployed business network. The composer-rest-server was installed when you installed the development environment.

Now launch the server by changing directory to the my-network folder and type:

cd ..

composer-rest-server

These allow the composer-rest-server to connect to Hyperledger Fabric and configure how the REST API is generated.

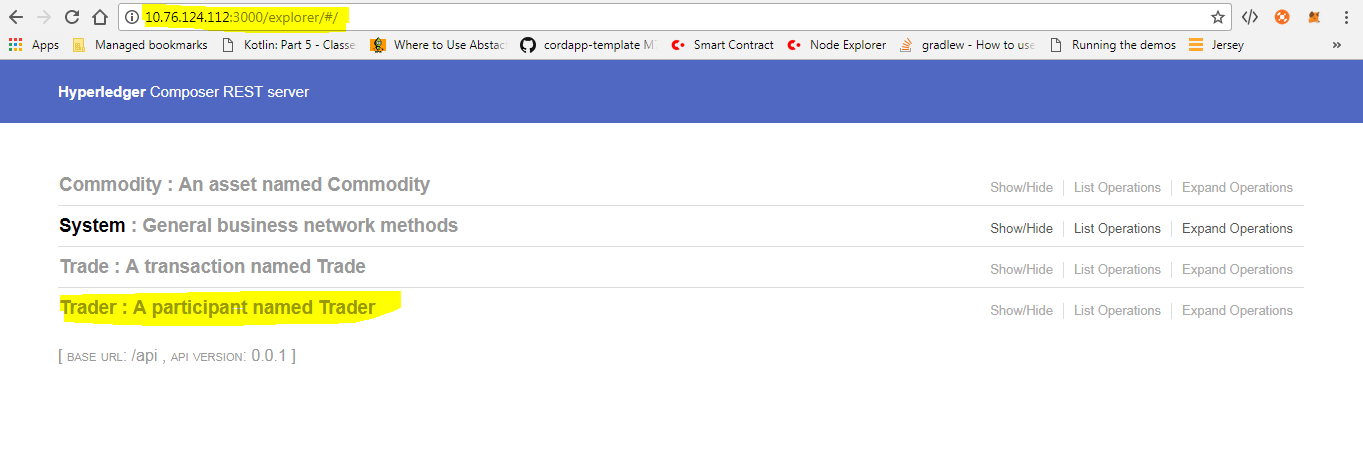


**Step 7**: Test REST API

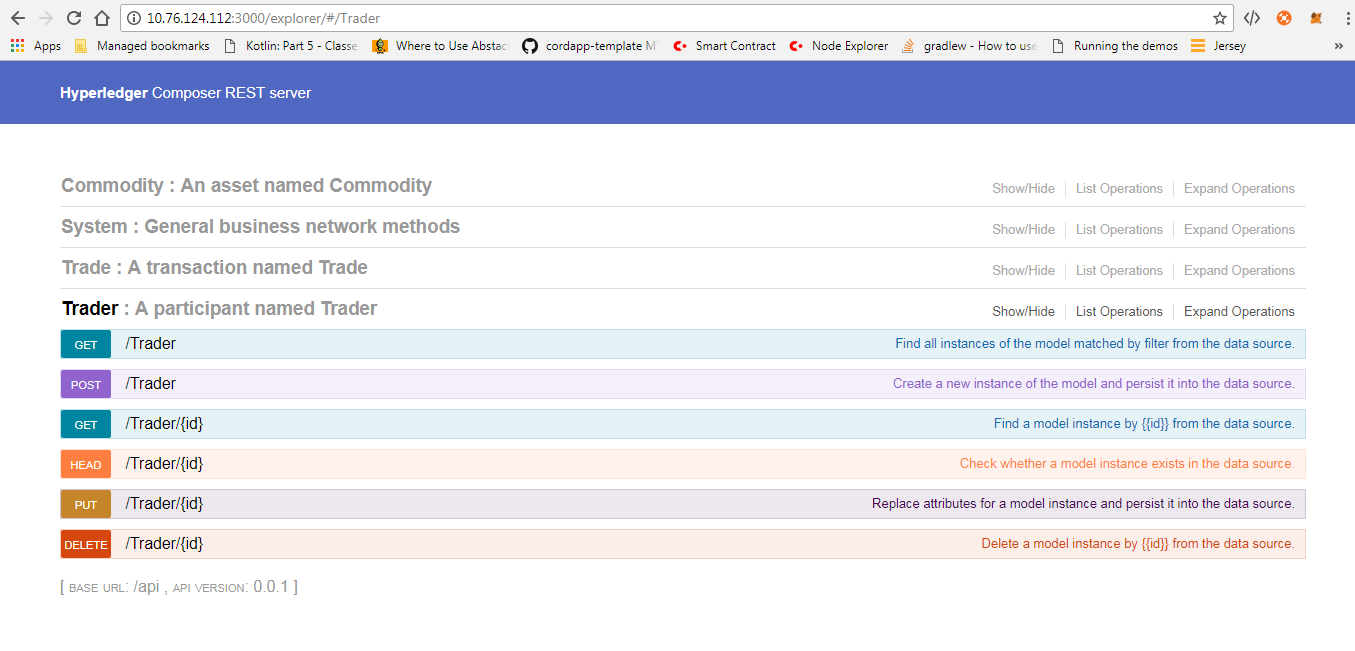
Open a web browser and navigate to <http://localhost:3000/explorer>

You should see the LoopBack API Explorer, allowing you to inspect and test the generated REST API.

Let’s POST a Trader and GET him by his id.



Select Trader.



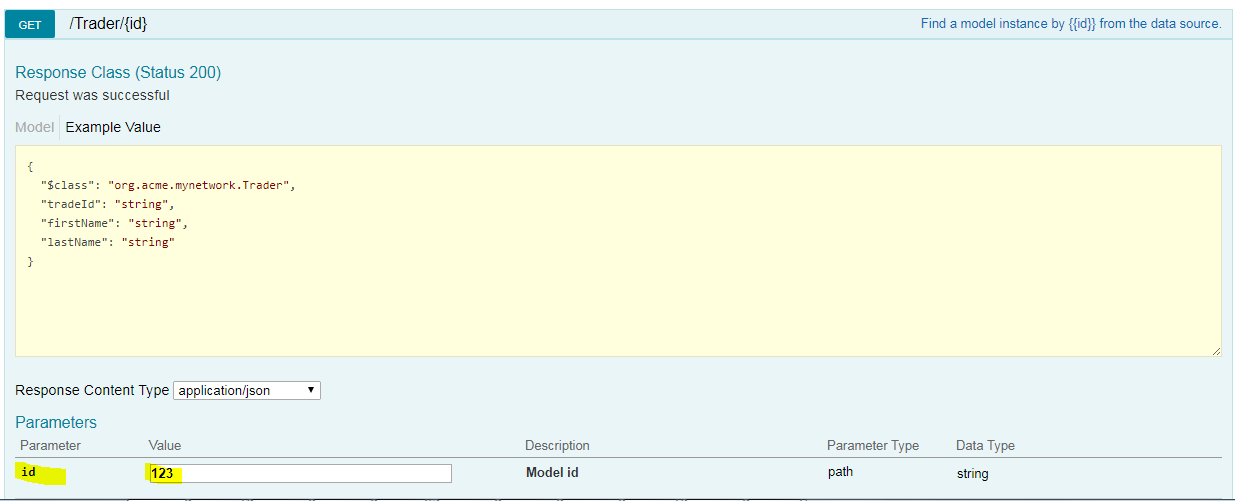
Select POST operation to post a trader in blockchain.



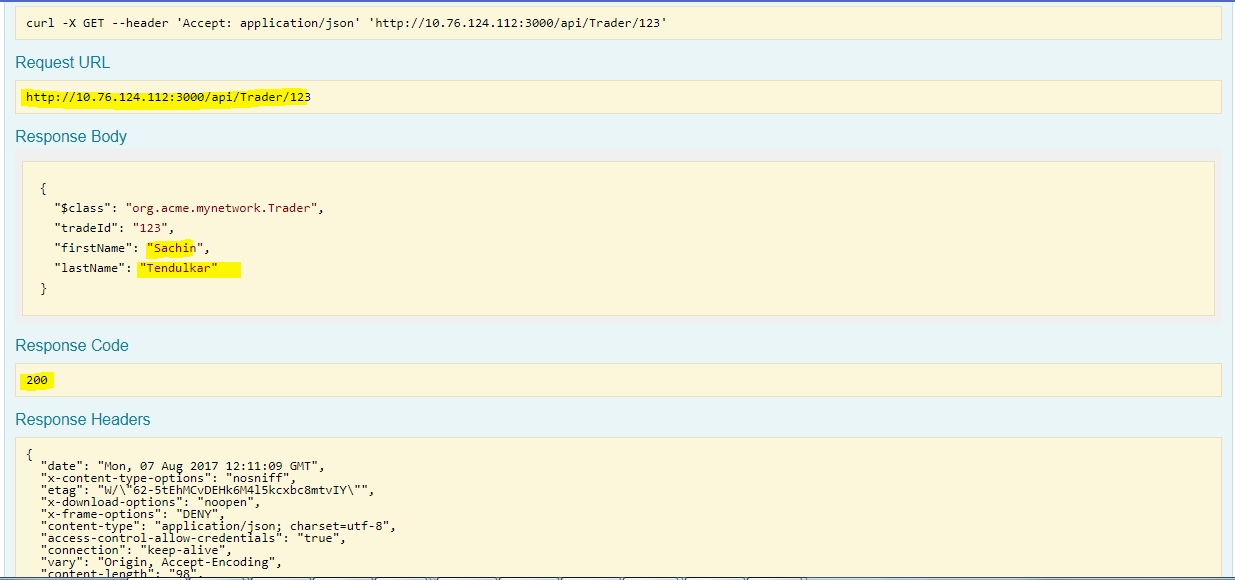
Click on ***“Try it out”*** button



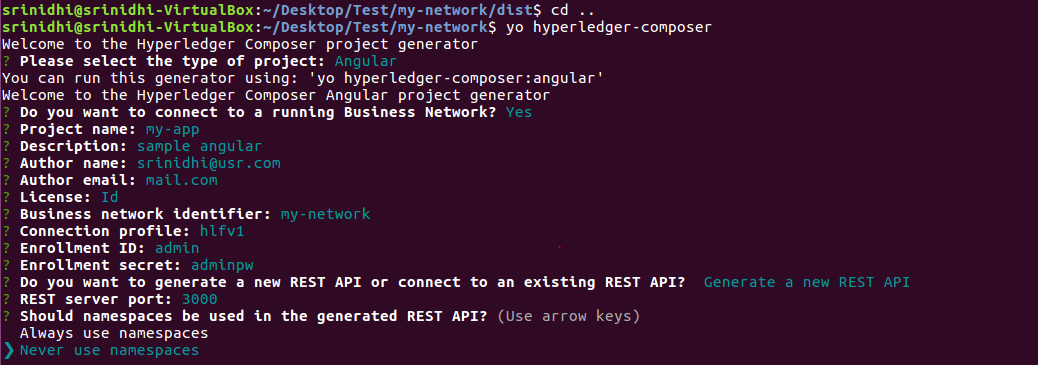
Verify the response code – 200.



Response of GET



**Step 8:** From Yo template.

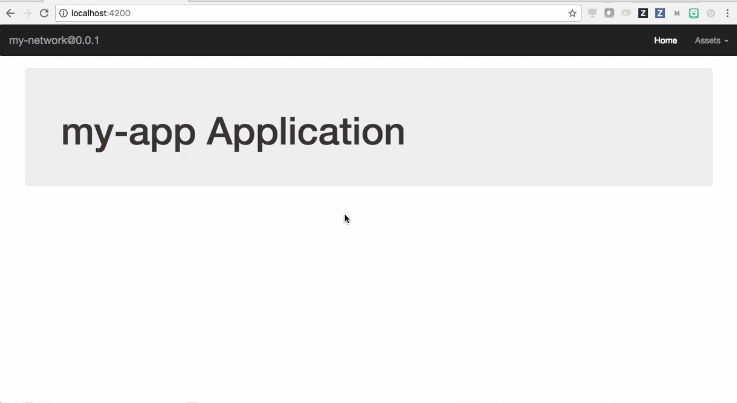


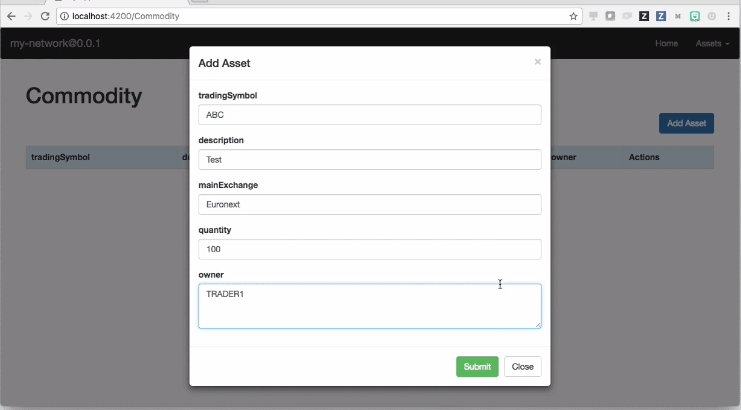
cd my-app

npm start

Your application is running. You should see the composer-rest-server start, and then Angular web packs the web application - it serves the content at URL: [http://localhost:4200](http://localhost:4200/)

If you navigate to this URL and press the "Assets" drop down (at the top-right of the page) you can see any existing instances of Commodity stored on the Hyperledger Fabric in the table (we have not stored them on blockchain yet!). You can create new instances using the "Add Asset" button. Note that the Angular skeleton does not yet allow you to create Participants, so you would need to create a test Trader instance to act as the owner of the Commodity (being added here), using the Loopback Swagger UI ( (ie using Playground) before you can create a Commodity.





**Chain code issue** - <https://stackoverflow.com/questions/43666890/fabric-composer-test-code-not-working>

**Document source** - <https://hyperledger.github.io/composer/tutorials/developer-guide.html>