IBM Blockchain Hands-On

IBM Blockchain Platform Visual Studio Code Extension:

Connecting to an Existing Network

Lab Four

Table of Contents

Disclaimer

1 Overview of the lab 4 environment and scenario

This lab is a technical introduction to blockchain, specifically smart contract development using the latest developer enhancements in the Linux Foundation's Hyperledger Fabric v1.4 and shows you how IBM's Blockchain Platform's developer experience can accelerate your pace of development.

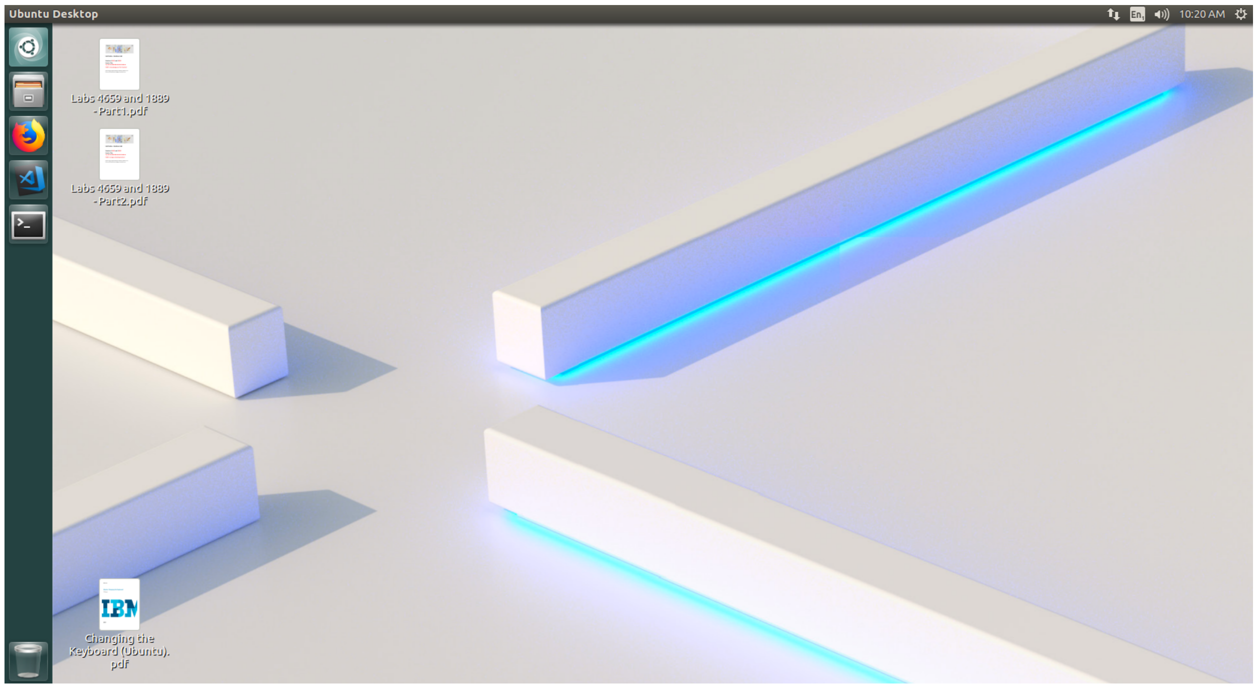
***Note**: The screenshots in this lab guide were taken using version **1.31.1** of **VSCode**, and version **0.3.0** of the **IBM Blockchain Platform** plugin. If you use different versions, you may see differences those shown in this guide.*

Start here. Instructions are always shown on numbered lines like this one:

__ 1. If it is not already running, start the virtual machine for the lab. The instructor will tell you how to do this if you are unsure.

__ 2. Wait for the image to boot and for the associated services to start. This happens automatically but might take several minutes. The image is ready to use when the desktop is visible as per the screenshot below.

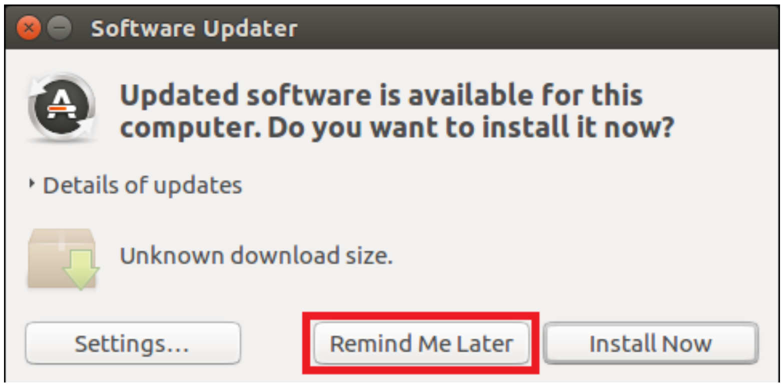
If it asks you to login, the userid and password are both "blockchain".


Ubuntu Desktop
10:20 AM
Labs 4659 and 1889 - Part1.pdf
Labs 4659 and 1889 - Part2.pdf
Changing the Keyboard (Ubuntu).pdf

1.1 Lab 4 Scenario

In this lab, we will take you through connecting to an existing network, one that is running outside of VSCode. The network will be using is the 'basic-network' used by the "Commercial Paper" Hyperledger Fabric tutorial, and we will stand this network up, run through a simple version of the tutorial.

***Note** that if you get an "Software Updater" pop-up at any point during the lab, please click "**Remind Me Later**":*


Software Updater

Updated software is available for this computer. Do you want to install it now?

Unknown download size.

2 Lab 4: Connecting to an Existing Network

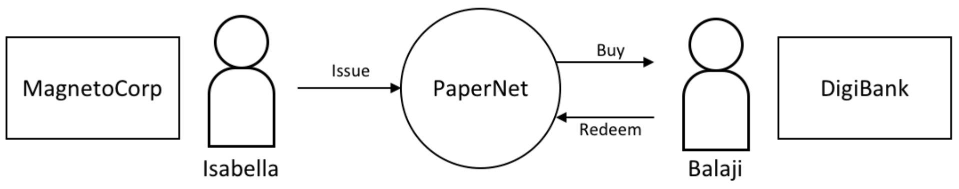
As mentioned above, this lab will be using the Hyperledger Fabric “Commercial Paper” tutorial. The full version of this tutorial is available online and we will be using a simplified version of it.

The scenario the tutorial follows is one of a commercial paper trading network called **PaperNet**. Commercial paper itself is a type of unsecured lending in the form of a “promissory note”. The papers are normally issued by large corporations to raise funds to meet short-term financial obligations at a fixed rate of interest. Once issued at a fixed price, for a fixed term, another company or bank will purchase them at a discount to the face value and when the term is up, they will be redeemed for their face value.

As an example, if a paper was **issued** at a face value of 10M USD for a 6-month term at 2% interest then it could be **bought** for 9.8M USD (10M – 2%) by another company or bank who are happy to bear the risk that the issuer will not default. Once the term is up, then the paper could be **redeemed** or sold back to the issuer for their full face value of 10M USD. Between buying and redemption, the paper can be bought or sold between different parties on a commercial paper market.

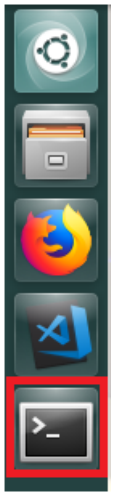
These three key steps of, **issue**, **buy** and **redeem** are the main transactions in a simplified commercial paper marketplace, which we will mirror in our lab. We will see a commercial paper **issued** by a company called MagnetoCorp and once issued on the PaperNet blockchain network, another company called DigiBank will first **buy** the paper and then **redeem** it.

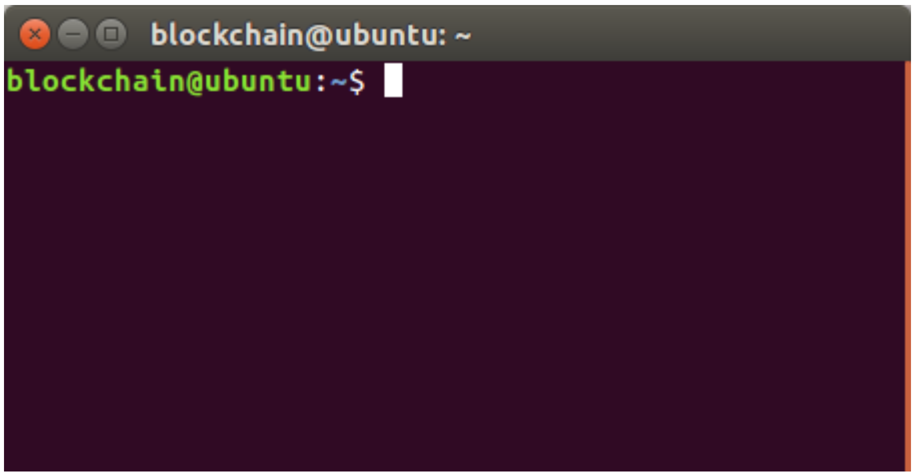
In diagram form it looks like this:



So, let’s begin!

__ 3. Use the launcher to open a terminal:



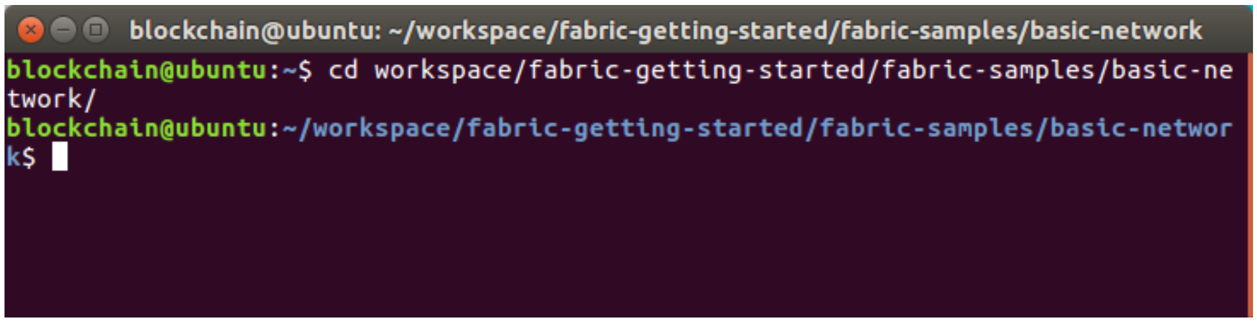


```
blockchain@ubuntu: ~
blockchain@ubuntu:~$ 
```

__ 4. Enter the **cd** command to change directory to the basic network folder that we will use for this lab:

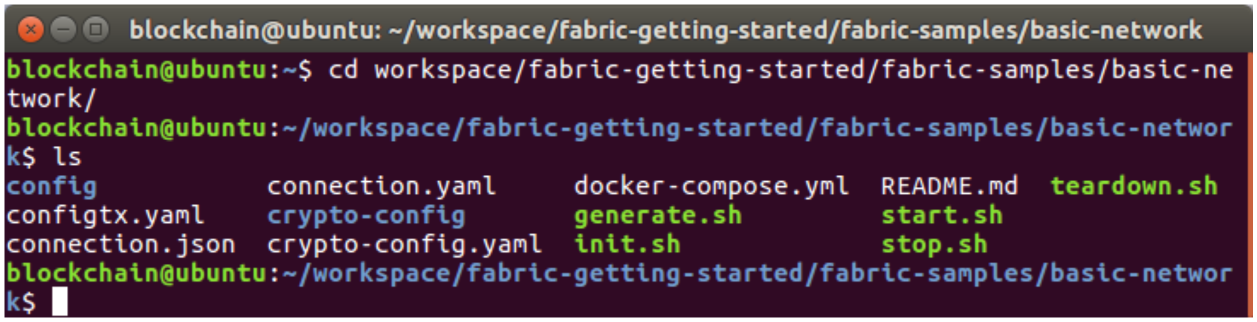
```
cd workspace/fabric-getting-started/fabric-samples/basic-network/
```

If you are not familiar with the command line, you can just copy and paste the above or you can type it yourself if you prefer. Remember you can press the **tab** key after typing the first few characters of the folder name to auto-complete the full name. When you have run the command, your console should look like this:



```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/basic-network
blockchain@ubuntu:~$ cd workspace/fabric-getting-started/fabric-samples/basic-ne
twork/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ 
```

__ 5. Type the **ls** command and press **enter** to see the files that make up the basic-network.



```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/basic-network
blockchain@ubuntu:~$ cd workspace/fabric-getting-started/fabric-samples/basic-ne
twork/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ ls
config           connection.yaml     docker-compose.yml  README.md  teardown.sh
configtx.yaml    crypto-config       generate.sh         start.sh
connection.json  crypto-config.yaml  init.sh             stop.sh
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ 
```

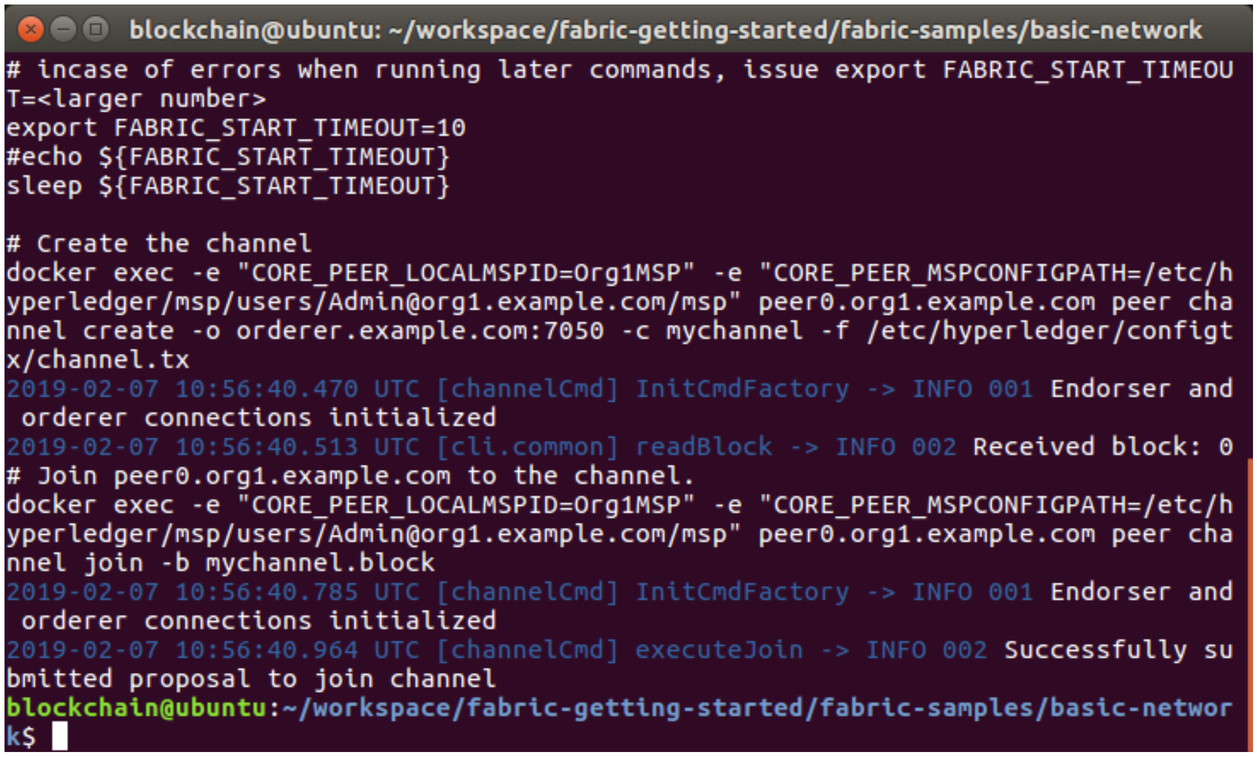
These files contain the configuration for the basic-network along with a script to set it up. Feel free to have a look at their contents if you are curious, the main files of interest are **start.sh** and **docker-compose.yml**

__ 6. To start the network running, tear down any existing blockchain networks and then start the network for this lab by running the following two commands in your terminal (press enter after each command and wait for its completion):

```
./teardown.sh
./start.sh
```

Note: make sure you enter the period (.) at the beginning of each command or the command will not be found.

The start command is a script that starts the docker containers that make up the basic network and may take a minute or so to run. When it has finished, your terminal should look something like this:

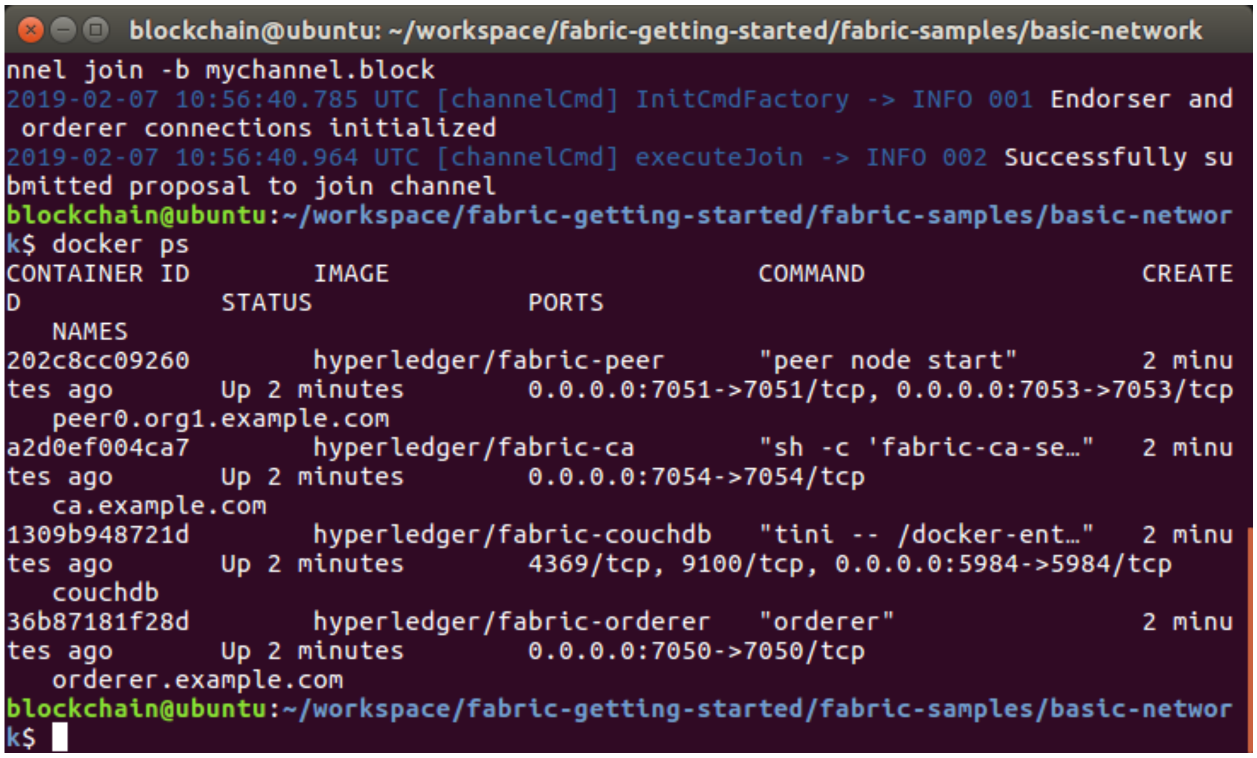


```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/basic-network
# incase of errors when running later commands, issue export FABRIC_START_TIMEOU
T=<larger number>
export FABRIC_START_TIMEOUT=10
#echo ${FABRIC_START_TIMEOUT}
sleep ${FABRIC_START_TIMEOUT}

# Create the channel
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/h
yperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer cha
nnel create -o orderer.example.com:7050 -c mychannel -f /etc/hyperledger/configt
x/channel.tx
2019-02-07 10:56:40.470 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and
 orderer connections initialized
2019-02-07 10:56:40.513 UTC [cli.common] readBlock -> INFO 002 Received block: 0
# Join peer0.org1.example.com to the channel.
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/h
yperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer cha
nnel join -b mychannel.block
2019-02-07 10:56:40.785 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and
 orderer connections initialized
2019-02-07 10:56:40.964 UTC [channelCmd] executeJoin -> INFO 002 Successfully su
bmitted proposal to join channel
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$
```

Next we will take a look at what containers were started.

__ 7. Run the command below in your terminal:

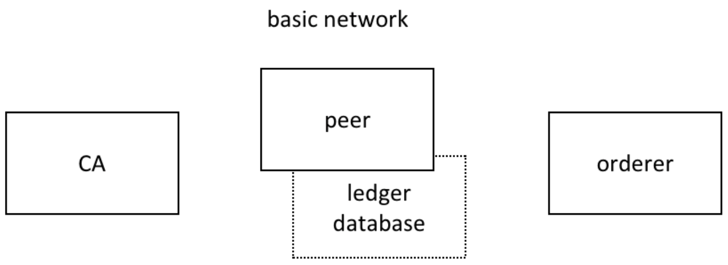
```
docker ps
```



blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/basic-network

```
nnel join -b mychannel.block
2019-02-07 10:56:40.785 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and
 orderer connections initialized
2019-02-07 10:56:40.964 UTC [channelCmd] executeJoin -> INFO 002 Successfully su
bmitted proposal to join channel
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ docker ps
CONTAINER ID        IMAGE                        COMMAND                  CREATE
D              STATUS              PORTS
   NAMES
202c8cc09260        hyperledger/fabric-peer      "peer node start"        2 minu
tes ago        Up 2 minutes        0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp
   peer0.org1.example.com
a2d0ef004ca7        hyperledger/fabric-ca        "sh -c 'fabric-ca-se…"   2 minu
tes ago        Up 2 minutes        0.0.0.0:7054->7054/tcp
   ca.example.com
1309b948721d        hyperledger/fabric-couchdb   "tini -- /docker-ent…"   2 minu
tes ago        Up 2 minutes        4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
   couchdb
36b87181f28d        hyperledger/fabric-orderer   "orderer"                2 minu
tes ago        Up 2 minutes        0.0.0.0:7050->7050/tcp
   orderer.example.com
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$
```

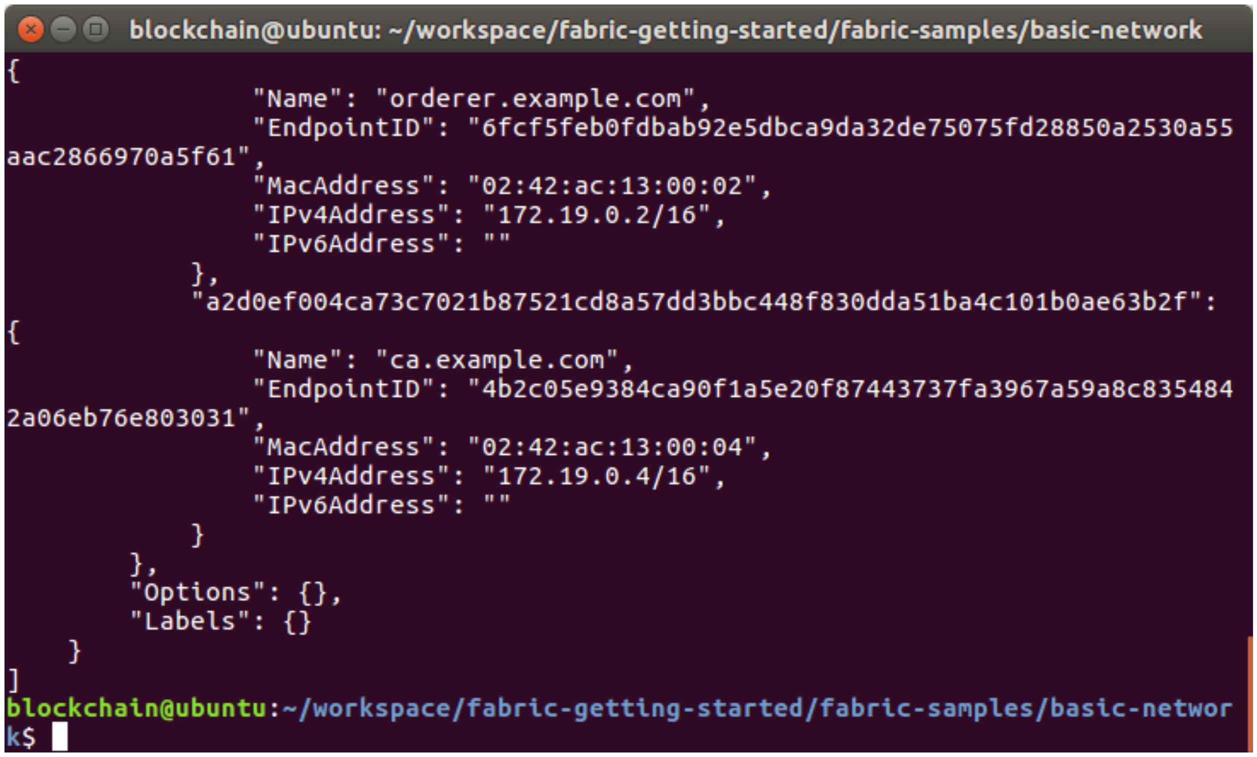
This command lists the docker containers that are running. Although this output is a little hard to read, you can make your terminal window wider to see the output better if you wish. This command shows that we have started four containers, one for each of the Hyperledger **fabric-peer**, **fabric-ca**, **fabric-couchdb** and **fabric-orderer**. Together these make up the simple basic-network that we will be using. A more realistic setup would have multiple copies of the components to better reflect the multiple parties in the network, but for a lab this simple network will suffice. In a diagram form, the network looks like this:



__ 8. These containers are joined together into the same docker network called net_basic. A docker network lets containers communicate with each other. Take a look at the network by running this command to inspect it:

```
docker network inspect net_basic
```

You should see output similar to this:



```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/basic-network
{
                "Name": "orderer.example.com",
                "EndpointID": "6fcf5feb0fdbab92e5dbca9da32de75075fd28850a2530a55
aac2866970a5f61",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            },
            "a2d0ef004ca73c7021b87521cd8a57dd3bbc448f830dda51ba4c101b0ae63b2f":
{
                "Name": "ca.example.com",
                "EndpointID": "4b2c05e9384ca90f1a5e20f87443737fa3967a59a8c835484
2a06eb76e803031",
                "MacAddress": "02:42:ac:13:00:04",
                "IPv4Address": "172.19.0.4/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ 
```

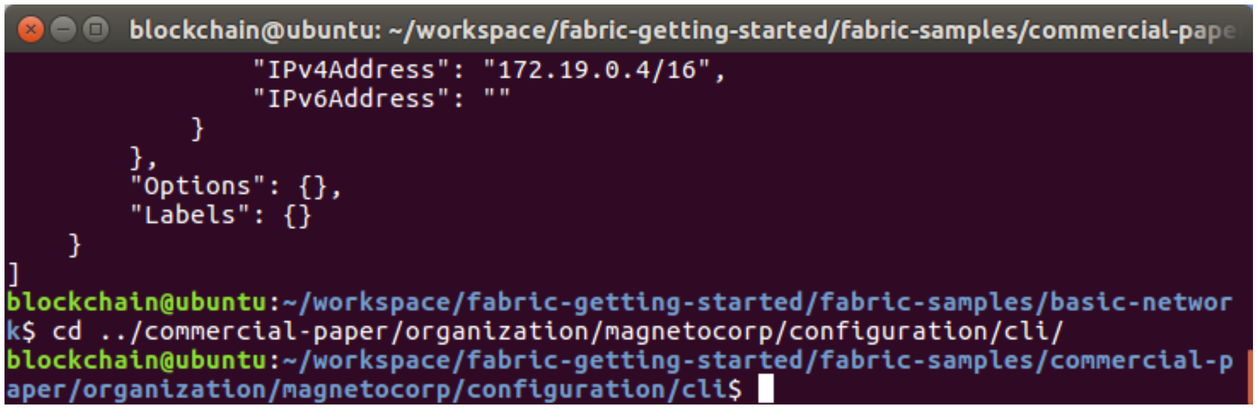
Scroll back up and look at the output. You can see that each of the containers have their own IP address inside the same network so they can communicate with each other.

Next, we will begin working as if we were an administrator for MagnetoCorp who would want to see combined logs from all of these components. Although proper dash-boards could be created, we will use a simple log viewing tool in this lab.

__ 9. From the terminal, change to the following folder:

```
cd ../commercial-paper/organization/magnetocorp/configuration/cli/
```

The full path should now be showing as: **~/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/magnetocorp/configuration/cli**

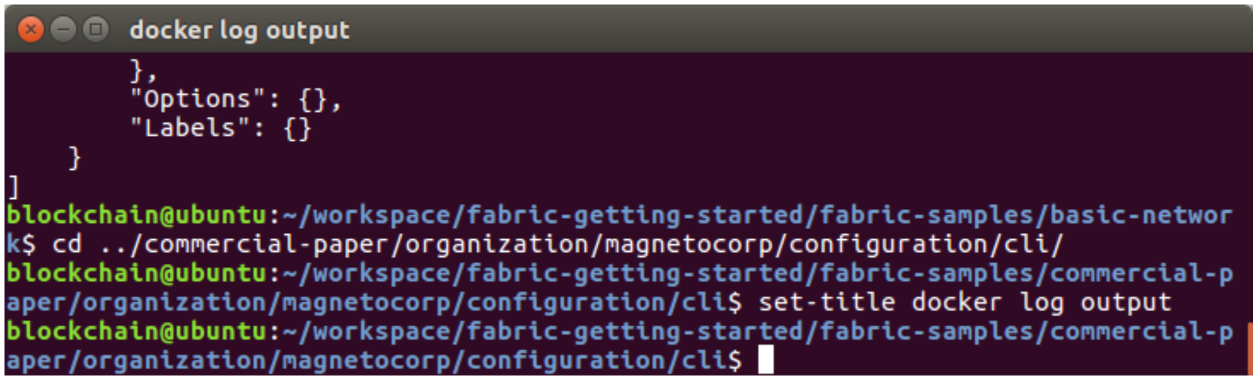


```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/commercial-pape
                "IPv4Address": "172.19.0.4/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ cd ../commercial-paper/organization/magnetocorp/configuration/cli/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ 
```

__ 10. As this terminal is going to spend the rest of this lab showing the logs, let's give it a title so we can easily see what it is doing:

```
set-title docker log output
```

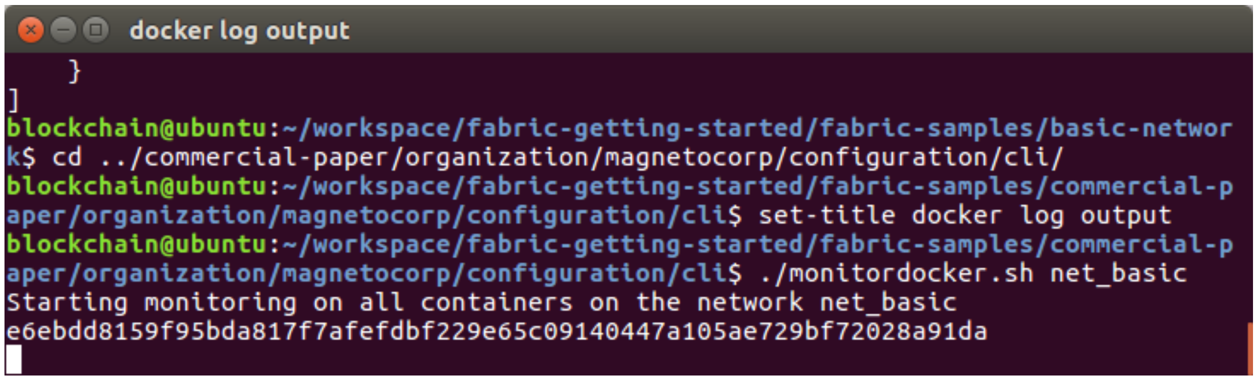
As you can see, this changes the title of the terminal window to make it easily identifiable:



```
docker log output
        },
        "Options": {},
        "Labels": {}
    }
]
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ cd ../commercial-paper/organization/magnetocorp/configuration/cli/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ set-title docker log output
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ 
```

__ 11. Now we will start showing the docker logs in this terminal:

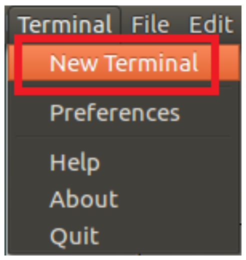
```
./monitordocker.sh net_basic
```



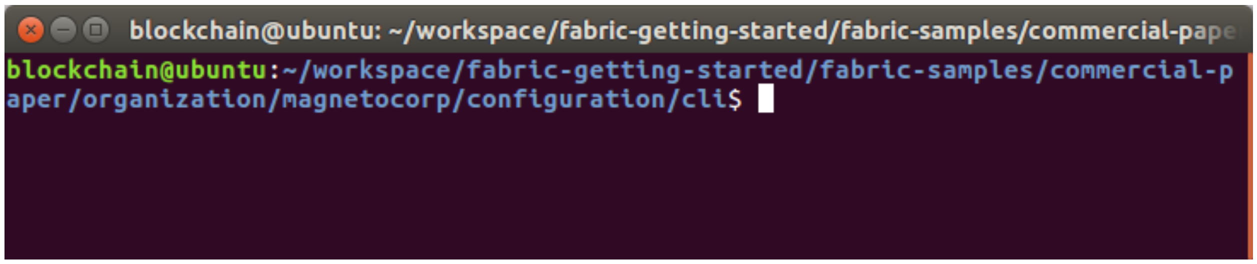
```
    }
]
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/basic-networ
k$ cd ../commercial-paper/organization/magnetocorp/configuration/cli/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ set-title docker log output
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ ./monitordocker.sh net_basic
Starting monitoring on all containers on the network net_basic
e6ebdd8159f95bda817f7afefdbf229e65c09140447a105ae729bf72028a91da
```

This terminal is now waiting to display any logging output from the **net_basic** docker network. Keep an eye on this terminal throughout the rest of this lab and you will see logging messages appear as we run the different commands.

__ 12. Now we need a new terminal window as the current one is dedicated to logging output. Make sure the current terminal window is selected by clicking in it, and from the "**Terminal**" menu at the top of the screen, choose "**New Terminal**"



This should open a new terminal window, with the path set to the same location as the logging terminal where we can enter new commands:



```
blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/commercial-pape
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$
```

Next, we are going to act as a MagnetoCorp administrator again and interact with the network. To do this we need to issue commands to the peer to install and instantiate chaincode. Unlike in the previous labs, we are going to do this manually without VSCode for now so we can see the different experience.

__ 13. First, let's give our terminal a name so we know who we are running as by issuing the command:

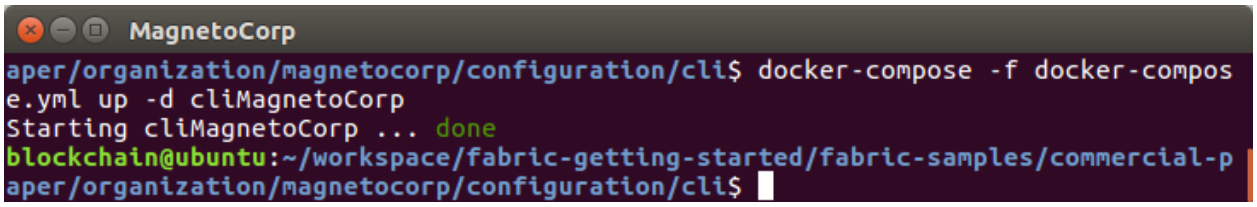
```
set-title MagnetoCorp
```

As before when we ran this for the logging window, you should see the terminal title change to **MagnetoCorp**.

__ 14. The Fabric commands we need to use are in the **fabric-tools** docker image, so let's start it running by issuing this docker-compose command to start the container:

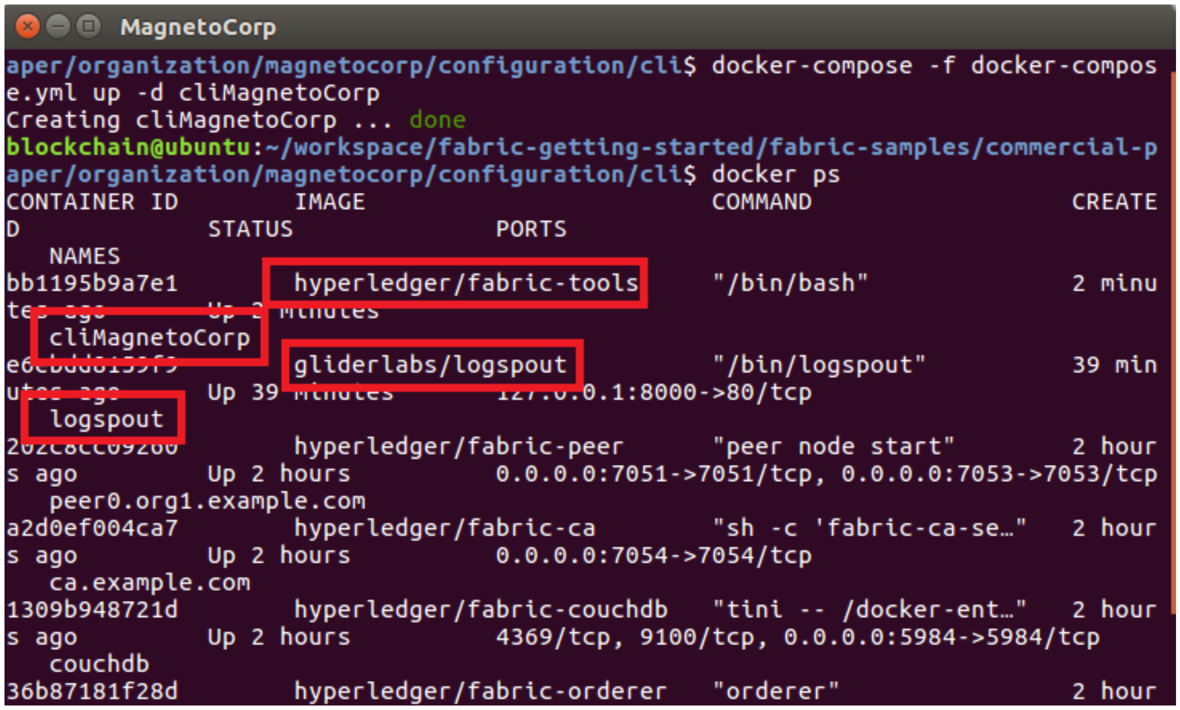
```
docker-compose -f docker-compose.yml up -d cliMagnetoCorp
```

Docker compose is a tool that allows you to define and run multi-container applications. In this case it will start the **fabric-tools** container and join it to the existing **net_basic** network. You should see output telling you that the container has been created, with the **cliMagnetoCorp** name we gave it:



```
aper/organization/magnetocorp/configuration/cli$ docker-compose -f docker-compos
e.yml up -d cliMagnetoCorp
Starting cliMagnetoCorp ... done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$
```

If we run the **docker ps** command again we should see that the new container is running. You can also see the container we created above for logging is running as well:



```
aper/organization/magnetocorp/configuration/cli$ docker-compose -f docker-compos
e.yml up -d cliMagnetoCorp
Creating cliMagnetoCorp ... done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/configuration/cli$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATE
D                   STATUS              PORTS
   NAMES
bb1195b9a7e1        hyperledger/fabric-tools       "/bin/bash"              2 minu
tes ago             Up 2 minutes
   cliMagnetoCorp
e6cbdd8159f9        gliderlabs/logspout            "/bin/logspout"          39 min
utes ago            Up 39 minutes       127.0.0.1:8000->80/tcp
   logspout
202c8cc09260        hyperledger/fabric-peer        "peer node start"        2 hour
s ago               Up 2 hours          0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp
   peer0.org1.example.com
a2d0ef004ca7        hyperledger/fabric-ca          "sh -c 'fabric-ca-se…"   2 hour
s ago               Up 2 hours          0.0.0.0:7054->7054/tcp
   ca.example.com
1309b948721d        hyperledger/fabric-couchdb     "tini -- /docker-ent…"   2 hour
s ago               Up 2 hours          4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
   couchdb
36b87181f28d        hyperledger/fabric-orderer     "orderer"                2 hour
```

Both of these containers have also been added to the **net_basic** network as well and you can run the **docker network inspect net_basic** command again if you want to see this for yourself.

Next we will begin to deploy the **PaperNet** smart contract.

__ 15. Change directory to the **magnetocorp/contract** folder:

```
cd ../../contract/
```

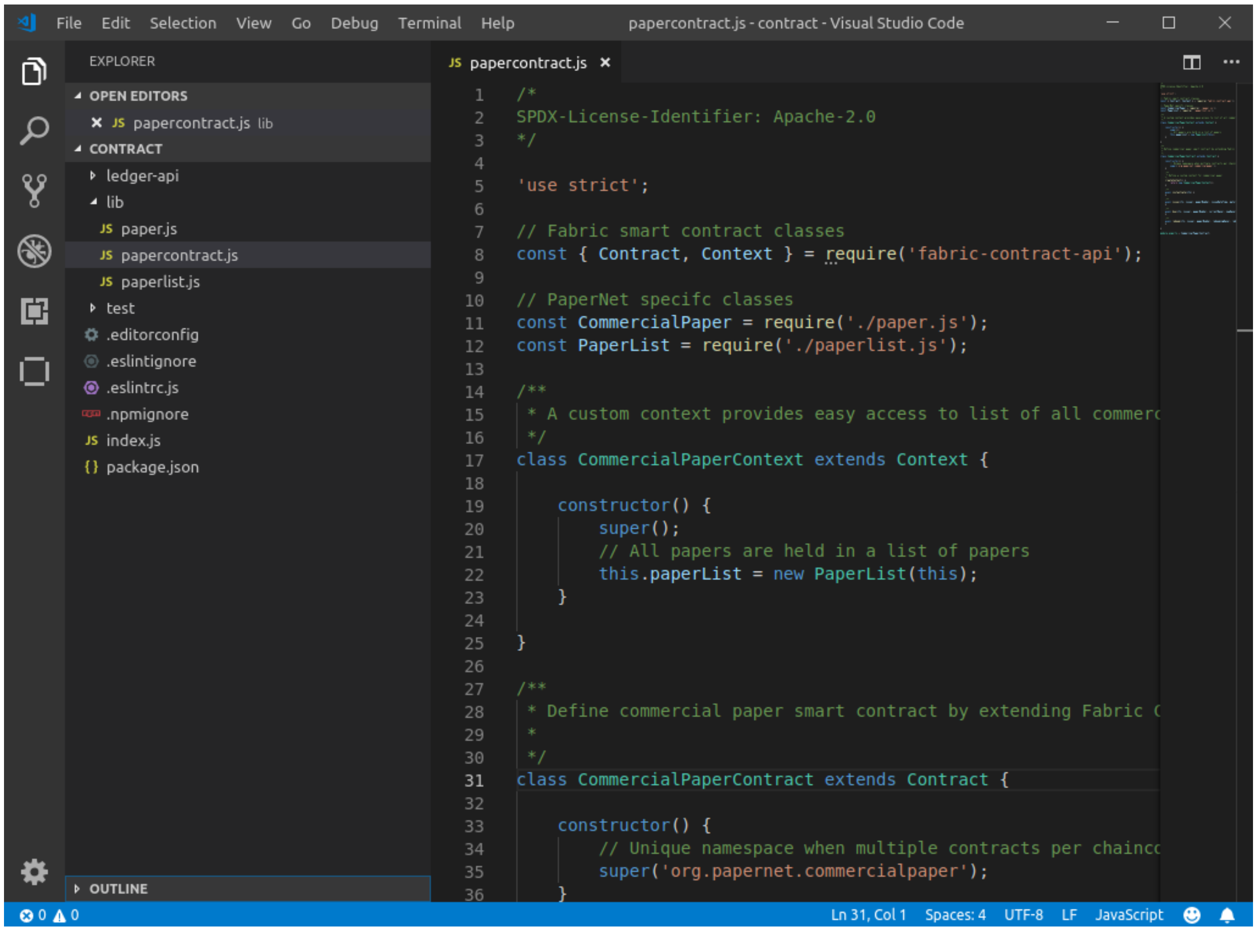
The full path should now be showing as: **~/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/magnetocorp/contract**

__ 16. Let's take a brief look at the contract in VSCode. Open the contract folder in VSCode by running:

```
code .
```

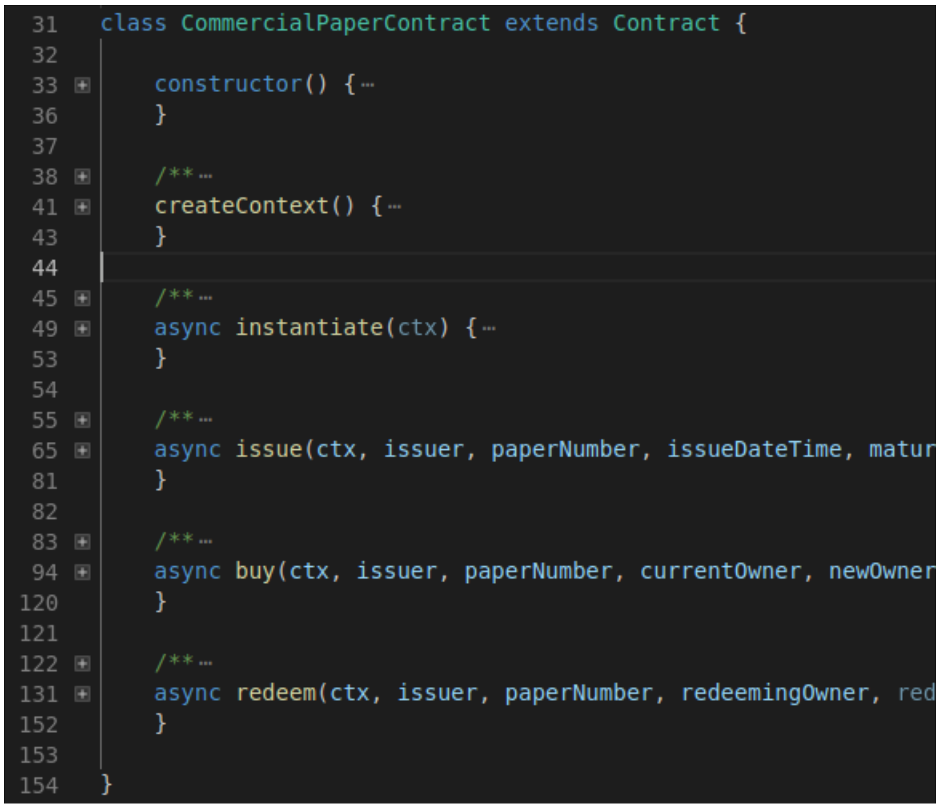
Note: do not miss the period (**.**) after the word **code**.

__ 17. When VSCode opens, expand the "lib" folder and double click on the papercontract.js file to open it in the edit pane:



```
/*
SPDX-License-Identifier: Apache-2.0
*/

'use strict';

// Fabric smart contract classes
const { Contract, Context } = require('fabric-contract-api');

// PaperNet specifc classes
const CommercialPaper = require('./paper.js');
const PaperList = require('./paperlist.js');

/**
 * A custom context provides easy access to list of all commerc
 */
class CommercialPaperContext extends Context {

    constructor() {
        super();
        // All papers are held in a list of papers
        this.paperList = new PaperList(this);
    }

}

/**
 * Define commercial paper smart contract by extending Fabric C
 *
 */
class CommercialPaperContract extends Contract {

    constructor() {
        // Unique namespace when multiple contracts per chainco
        super('org.papernet.commercialpaper');
    }
```

__ 18. The Commercial Paper contract is more sophisticated than the contracts we saw in parts one and two of this lab, but it mostly works the same way. The main **CommercialPaperContract** class starts on **line 31**, and if we use the **"-"** buttons in the VSCode editor to fold methods of this class we can see that the main transactions are **instantiate**, **issue**, **buy** and **redeem**.



```
 31  class CommercialPaperContract extends Contract {
 32
 33      constructor() {⋯
 36      }
 37
 38      /**⋯
 41      createContext() {⋯
 43      }
 44
 45      /**⋯
 49      async instantiate(ctx) {⋯
 53      }
 54
 55      /**⋯
 65      async issue(ctx, issuer, paperNumber, issueDateTime, matur
 81      }
 82
 83      /**⋯
 94      async buy(ctx, issuer, paperNumber, currentOwner, newOwner
120      }
121
122      /**⋯
131      async redeem(ctx, issuer, paperNumber, redeemingOwner, red
152      }
153
154  }
```

__ 19. Let's expand the **issue** transaction and take a look so we can see what it will do.

Line 68 creates a new **CommercialPaper** object from the parameters passed in using the static **createInstance** method on the **CommercialPaper** class. This class is defined in the separate "**paper.js**" file which is also if the **lib** folder alongside **papercontract.js** if you want to take a look at this method.

Line 72 Then moved the newly created paper into the **ISSUED** state and on **line 74** it has its owner set from the parameters passed in.

Line 77 adds the paper to a "**paperList**" which is responsible for storing the state of the paper in the world state. This is defined in the **paperlist.js** file if you want to take a deeper look.

Line 80 then returns a serialized form of the paper to the client who called this transaction.

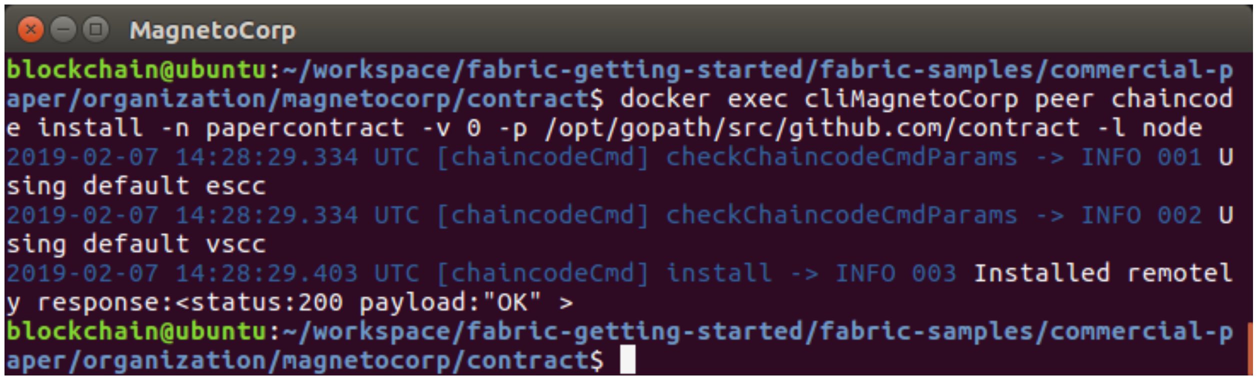
Feel free to have a look at the other files that make up the commercial paper smart contract. If you want to delve even deeper into the design of the CommercialPaper contract, there is much more information online if you have time to take a look.

Now we are going to install the **papercontract** onto a peer in the network.

__ 20. Switch back to the **MagnetoCorp** terminal and issue the following command:

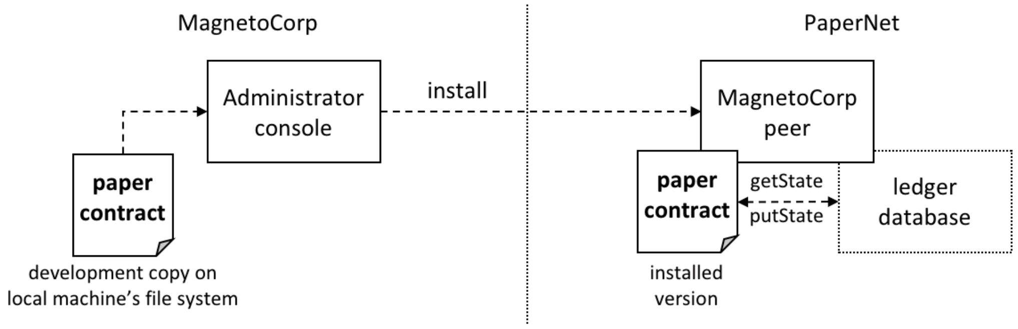
```
docker exec cliMagnetoCorp peer chaincode install -n
papercontract -v 0 -p /opt/gopath/src/github.com/contract -l node
```

Note: The above command must be entered as a single line. If you copy and paste it from the pdf, be sure to enter it as a single line.



```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/contract$ docker exec cliMagnetoCorp peer chaincod
e install -n papercontract -v 0 -p /opt/gopath/src/github.com/contract -l node
2019-02-07 14:28:29.334 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 U
sing default escc
2019-02-07 14:28:29.334 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 U
sing default vscc
2019-02-07 14:28:29.403 UTC [chaincodeCmd] install -> INFO 003 Installed remotel
y response:<status:200 payload:"OK" >
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/contract$
```

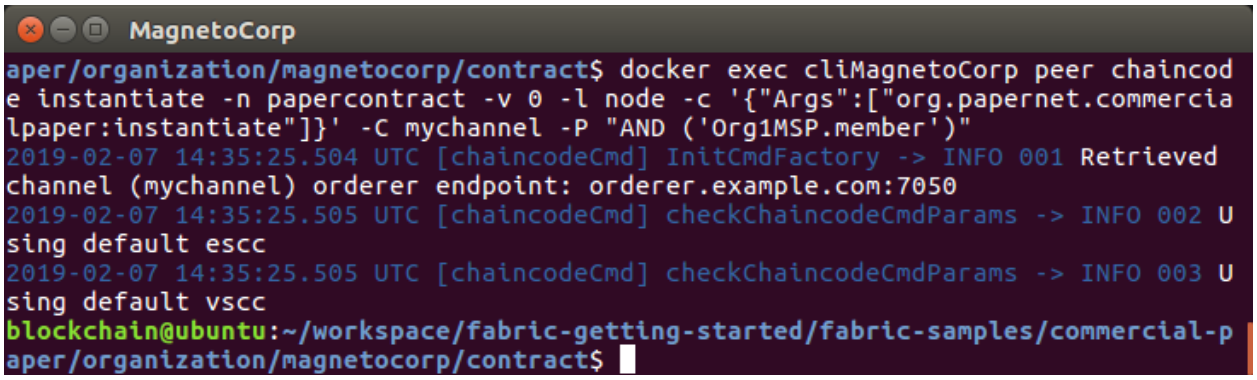
This command uses the **cliMagnetoCorp** container which was configured to send commands to **peer0.org1.example.com** in our basic network. Essentially it copies the **papercontract** source code and sends it to the remote peer, ready for it to be instantiated.



__ 21. To instantiate the contract on **peer0**, issue the following command at the **MagnetoCorp** terminal:

```
docker exec cliMagnetoCorp peer chaincode instantiate -n
papercontract -v 0 -l node -c
'{"Args":["org.papernet.commercialpaper:instantiate"]}' -C
mychannel -P "AND ('Org1MSP.member')"
```

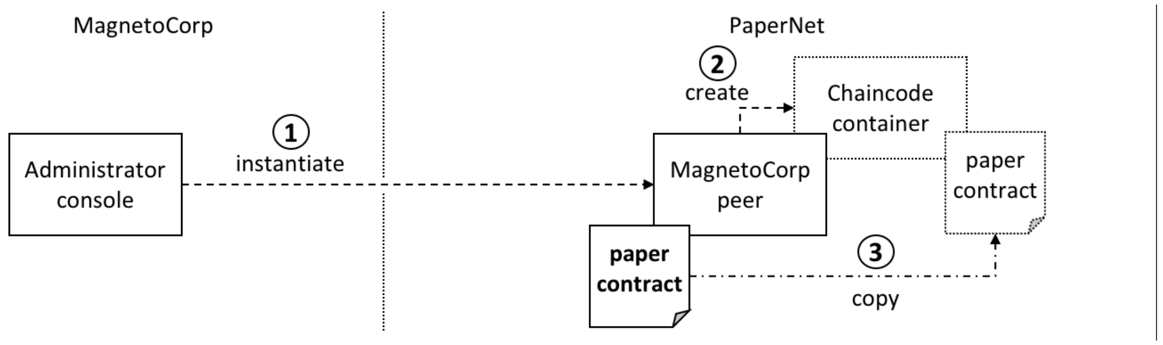
Note: As before, the above command must be entered as a single line. If you copy and paste it from the pdf, be sure to enter it as a single line.



```
aper/organization/magnetocorp/contract$ docker exec cliMagnetoCorp peer chaincod
e instantiate -n papercontract -v 0 -l node -c '{"Args":["org.papernet.commercia
lpaper:instantiate"]}' -C mychannel -P "AND ('Org1MSP.member')"
2019-02-07 14:35:25.504 UTC [chaincodeCmd] InitCmdFactory -> INFO 001 Retrieved
channel (mychannel) orderer endpoint: orderer.example.com:7050
2019-02-07 14:35:25.505 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 U
sing default escc
2019-02-07 14:35:25.505 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 U
sing default vscc
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/contract$
```

This command also uses the **cliMagnetoCorp** container to cause the contract to become instantiated on the **mychannel** channel. In addition, it also invokes the **instantiate** transaction as part of the command which allows any required initialisation to take place. Finally, note the last option – the **-P** flag which specifies which organisations in the network need to endorse the transactions issued by this contract before they will be considered valid.

Note that this command may take a little time to run as it will cause the peer to create a new docker container to be created to run the contract in.



If you wish to see this new container, run the **docker ps** command again – it should have the name: **dev-peer0.org1.example.com-papercontract-0**.

Note 2: Log messages are written to the `docker log output` window as a result of the **instantiate** transaction if you want to take a quick look.

Now the contract is up and running, it is time to start running transactions, and to start things moving, **MagnetoCorp** is going to run an application to **issue** the first commercial paper on the **PaperNet** network. To do this we are going to act as **Isabella**, an employee of **MagnetoCorp.**

__ 22. Change to the folder that contains the **issue** application.

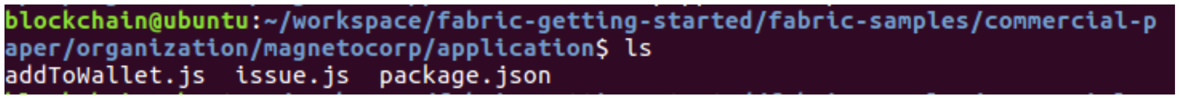
```
cd ../application/
```

The full path at this point should be: **~/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/magnetocorp/application**

__ 23. Run the **ls** command to see the files in this folder:

```
ls
```

We see that there are three files:



```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls
addToWallet.js  issue.js  package.json
```

Next let's take a look at the **issue** application.

__ 24. Run this command to open the issue application in the VSCode editor so we can take a look:

```
code issue.js
```

Take a look at the code for the issue application. The main points are:

Lines 18 - 21: import various dependencies

Line 25: Load the identity from the wallet on the file system

Line 31: Create a new Gateway

Line 41: Load the connection profile from file system

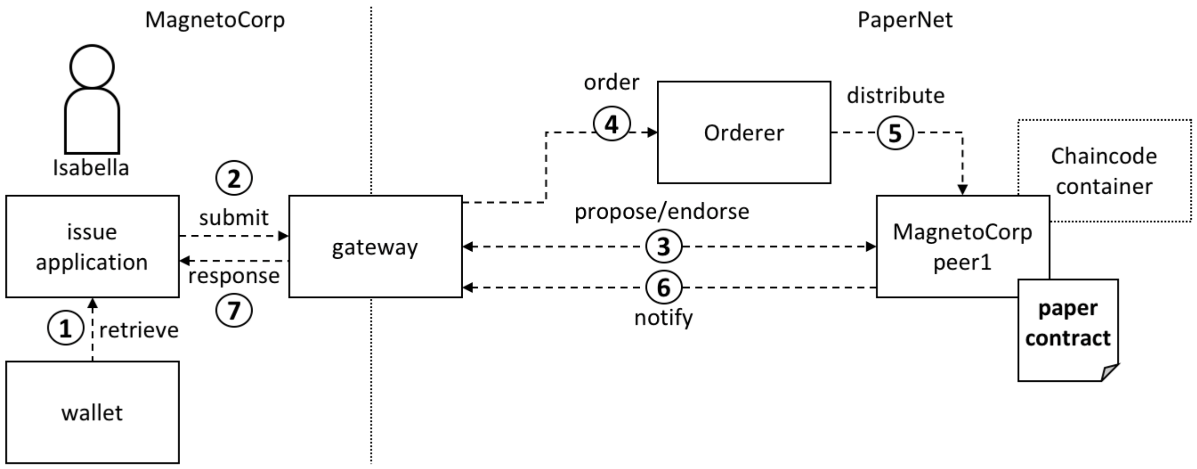
Line 53: Connect to the gateway
Line 58: Get the **mychannel** channel from the gateway

Line 63: Get the **papercontract** contract from the gateway

Line 68: Use the contract to submit the **issue** transaction, passing in the details of the paper to be issued.

Line 73: Log the response

As we can see above on line 25, the **issue** application will need to load Isabella's identity before it can create the transaction, so we need to make sure her identity is in the wallet that the **issue** application will use as shown in this diagram:

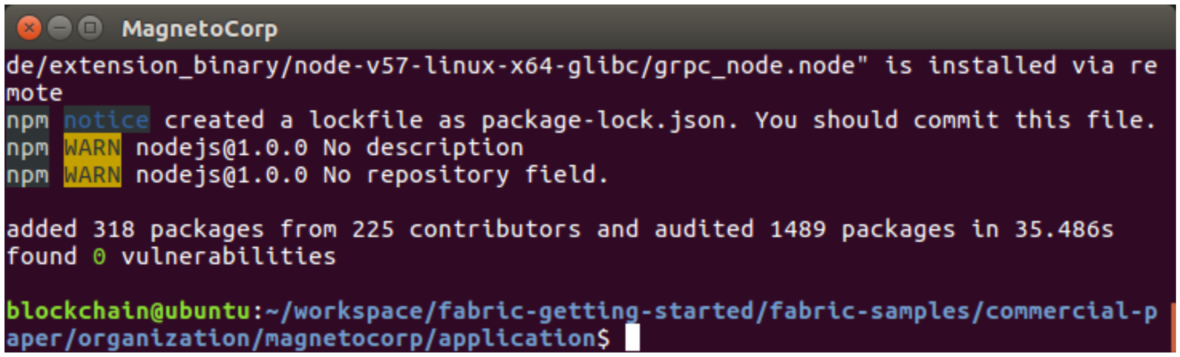


However, before we can run the application, we need to download the dependencies listed in the **package.json** file from **npm**

__ 25.	Switch back to the **MagnetoCorp** terminal and issue this command:

```
npm install
```

Note: This will take a while to download the dependencies.



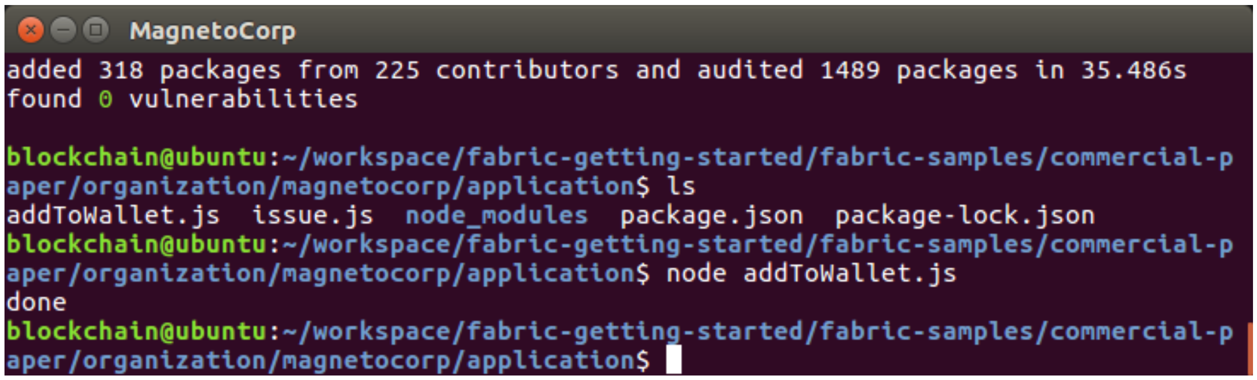
MagnetoCorp

```
de/extension_binary/node-v57-linux-x64-glibc/grpc_node.node" is installed via re
mote
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs@1.0.0 No description
npm WARN nodejs@1.0.0 No repository field.

added 318 packages from 225 contributors and audited 1489 packages in 35.486s
found 0 vulnerabilities

blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$
```

When the download has finished, if you run **ls** again, you will see a new **node_modules** folder has been created along with a **package-lock.json** file. It is the **node_modules** folder that contains the dependencies.

__ 26.	Now we are almost ready to issue a new commercial paper, we just need to load Isabella's digital certificate into the wallet before we can use it. To do this we run the following application:

```
node addToWallet.js
```



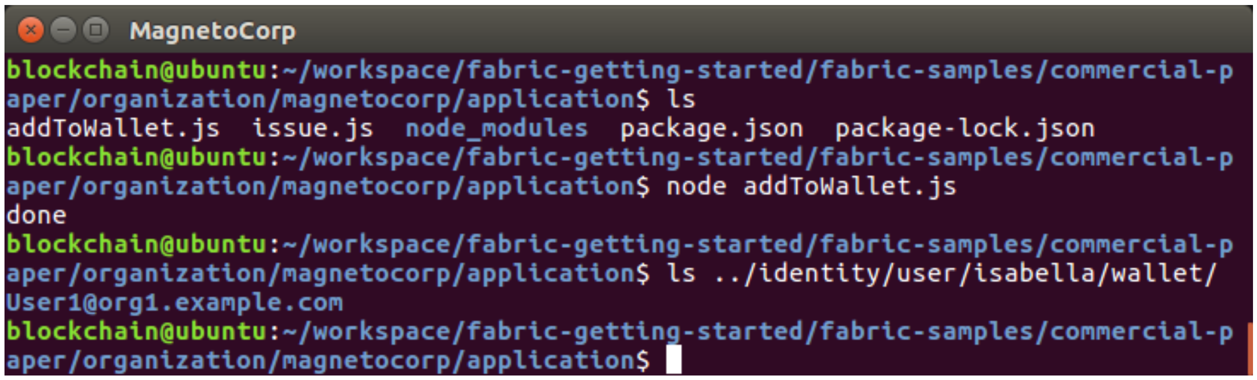
MagnetoCorp

```
added 318 packages from 225 contributors and audited 1489 packages in 35.486s
found 0 vulnerabilities

blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls
addToWallet.js  issue.js  node_modules  package.json  package-lock.json
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ node addToWallet.js
done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$
```

addToWallet simply copies an identity from the **basic_network** to our wallet location for use by other applications.

__ 27. Run this command to see the contents of the newly created wallet:

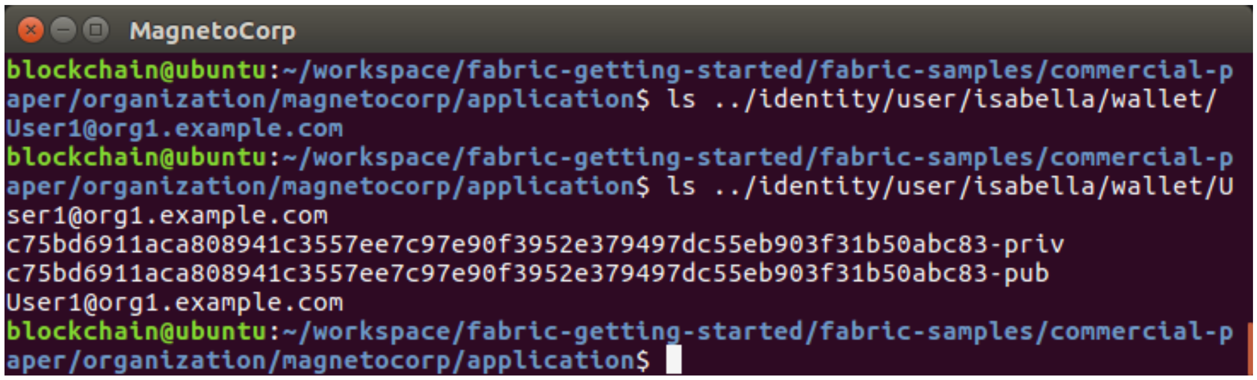
```
ls ../identity/user/isabella/wallet/
```



```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls
addToWallet.js  issue.js  node_modules  package.json  package-lock.json
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ node addToWallet.js
done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls ../identity/user/isabella/wallet/
User1@org1.example.com
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ 
```

Now you can see the **User1@org1.example.com** folder which is used by the issue application. If we run another command, we can see the three files that make up the identity itself:

```
ls ../identity/user/isabella/wallet/User1@org1.example.com
```

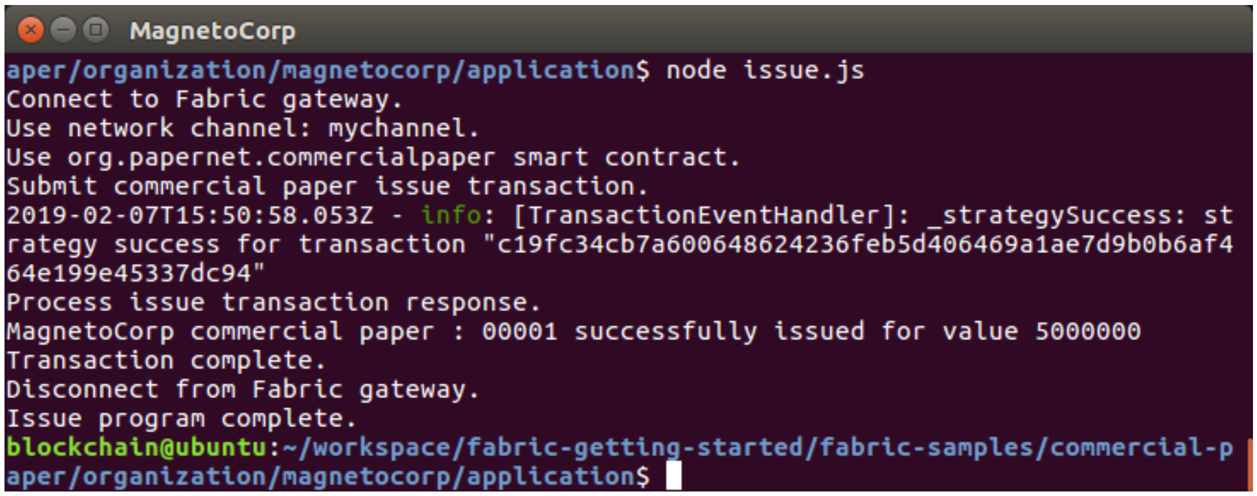


```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls ../identity/user/isabella/wallet/
User1@org1.example.com
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ ls ../identity/user/isabella/wallet/U
ser1@org1.example.com
c75bd6911aca808941c3557ee7c97e90f3952e379497dc55eb903f31b50abc83-priv
c75bd6911aca808941c3557ee7c97e90f3952e379497dc55eb903f31b50abc83-pub
User1@org1.example.com
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ 
```

These files consist of a private key for signing transactions, a public key linked to the private key and a file that contains both metadata and a certificate for our user.

__ 28. Now we can finally issue the commercial paper by running:

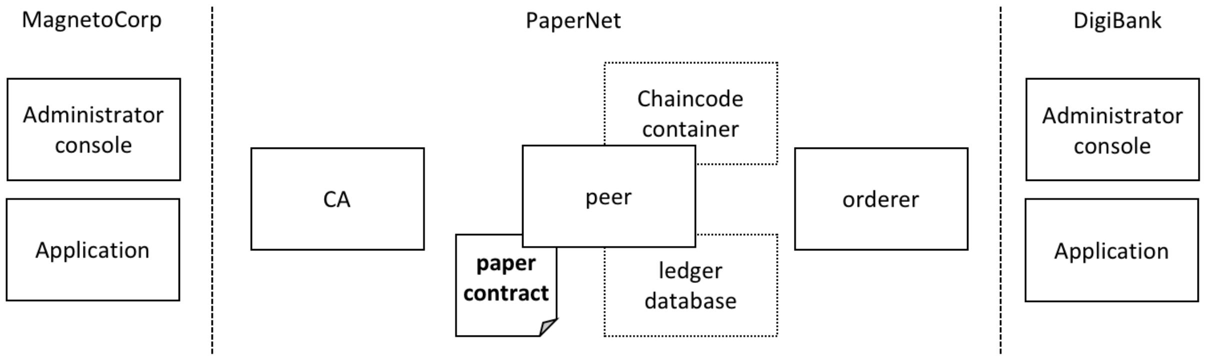
```
node issue.js
```



```
aper/organization/magnetocorp/application$ node issue.js
Connect to Fabric gateway.
Use network channel: mychannel.
Use org.papernet.commercialpaper smart contract.
Submit commercial paper issue transaction.
2019-02-07T15:50:58.053Z - info: [TransactionEventHandler]: _strategySuccess: st
rategy success for transaction "c19fc34cb7a600648624236feb5d406469a1ae7d9b0b6af4
64e199e45337dc94"
Process issue transaction response.
MagnetoCorp commercial paper : 00001 successfully issued for value 5000000
Transaction complete.
Disconnect from Fabric gateway.
Issue program complete.
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$
```

From the output you can see that we followed the steps outlined above to successfully issue commercial paper **00001** for **5,000,000** USD. This has submitted the transaction to the network and the contract has written these details to the world state and the ledger. The transaction was also endorsed and validated before it was committed.

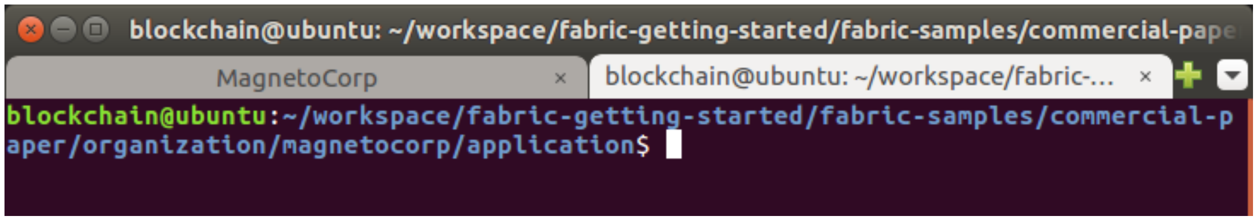
Now that we have issued paper **00001**, we want to take on the persona of an employee of **DigiBank** who is going to **buy** and **redeem** this paper. If we look at a diagram, we can see how they will interact with the same network:



__ 29. Let’s open up a new tabbed terminal window so we can act as **Balaji**, an employee of **DigiBank**. Make sure the **MagnetoCorp** terminal window has focus, then press:

```
ctrl + shift + t
```

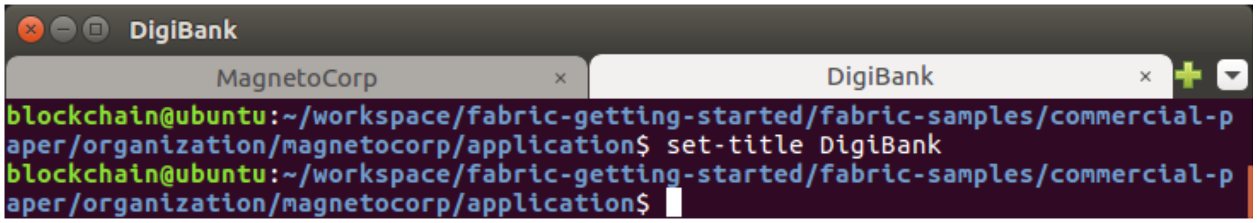
This will open up a new tab alongside the MagnetoCorp one:


blockchain@ubuntu: ~/workspace/fabric-getting-started/fabric-samples/commercial-pape
MagnetoCorp
blockchain@ubuntu: ~/workspace/fabric-...

```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ 
```

__ 30. For ease of use, let’s give this new terminal tab a name:

```
set-title DigiBank
```



```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ set-title DigiBank
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ 
```

Now we can easily switch between the different users as required.

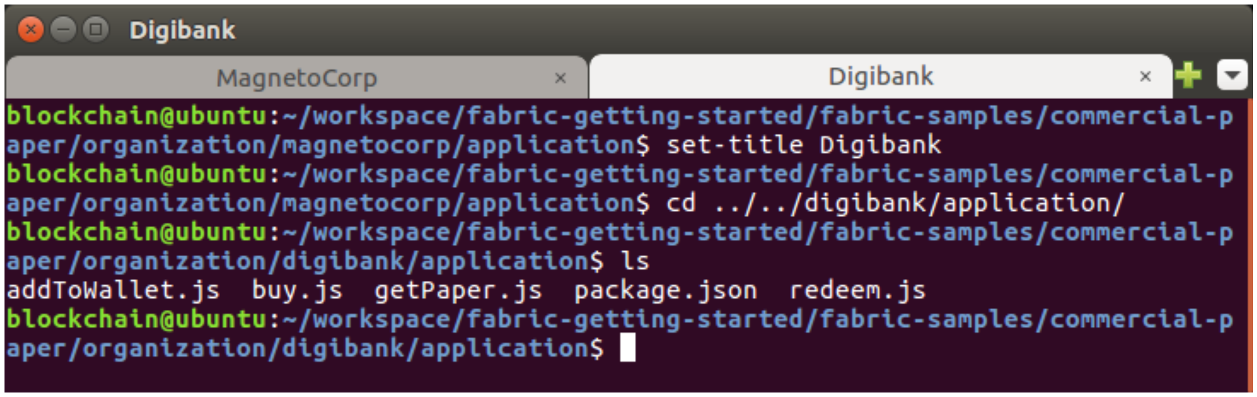
__ 31. Let’s change to **DigiBank**’s application folder:

```
cd ../../digibank/application/
```

The full path should be **~/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/digibank/application**

__ 32. Let's have a look at what files we have in this folder by running:

```
ls
```



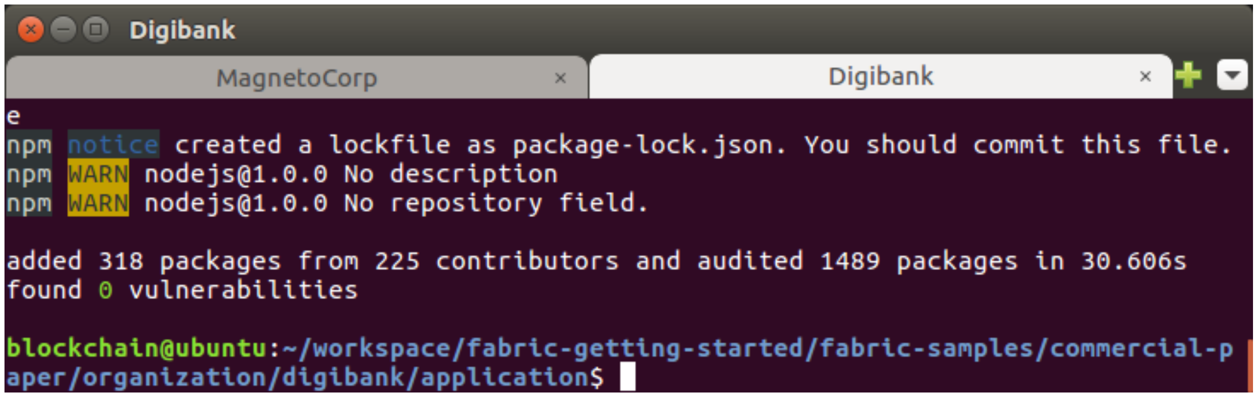
```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ set-title Digibank
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$ cd ../../digibank/application/
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ ls
addToWallet.js  buy.js  getPaper.js  package.json  redeem.js
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$
```

We can see this is similar to **MagnetoCorp's** application folder, but this time we have **buy.js**, **redeem.js** and **getPaper.js** rather than **issue.js**.

Note: **getPaper.js** is added as part of another lab, and it is not part of the original **Commercial Paper** tutorial.

__ 33. As before, before we can run anything, we need to download the dependencies from **npm**:

```
npm install
```

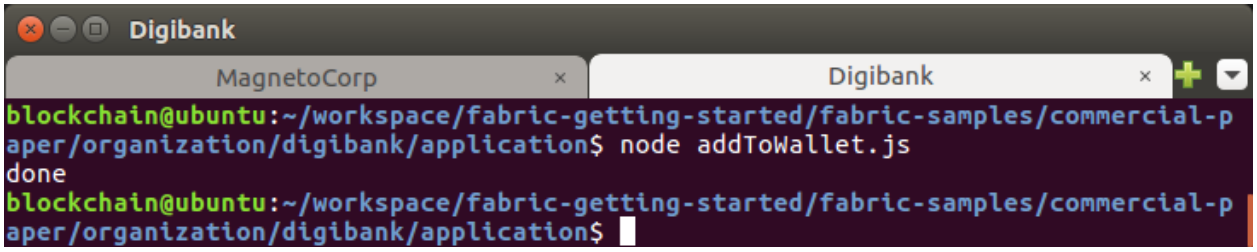


```
e
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs@1.0.0 No description
npm WARN nodejs@1.0.0 No repository field.

added 318 packages from 225 contributors and audited 1489 packages in 30.606s
found 0 vulnerabilities

blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$
```

Note: remember this can take a while to complete.

__ 34. As we are going to be running as **Balaji**, we need to load the identity they are going to use into **DigiBank**’s wallet, just like we did for Isabella earlier:

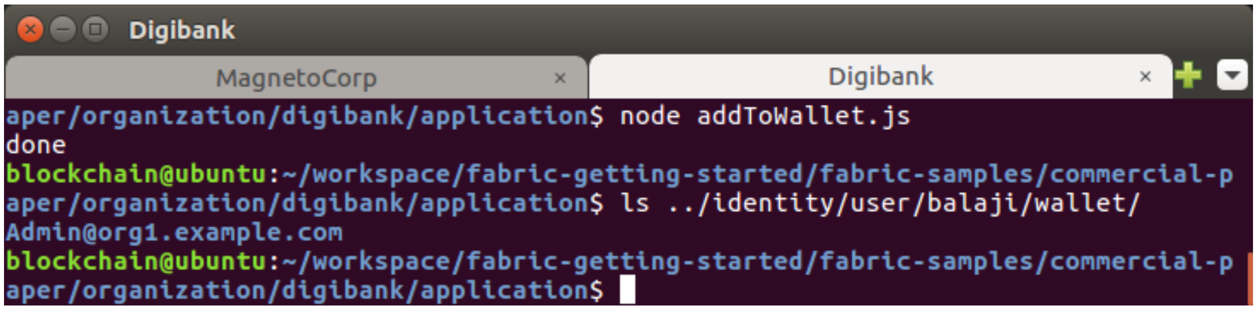
```
node addToWallet.js
```



```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ node addToWallet.js
done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ 
```

__ 35. Let’s take a quick look again at the identity the application copied:

```
ls ../identity/user/balaji/wallet/
```

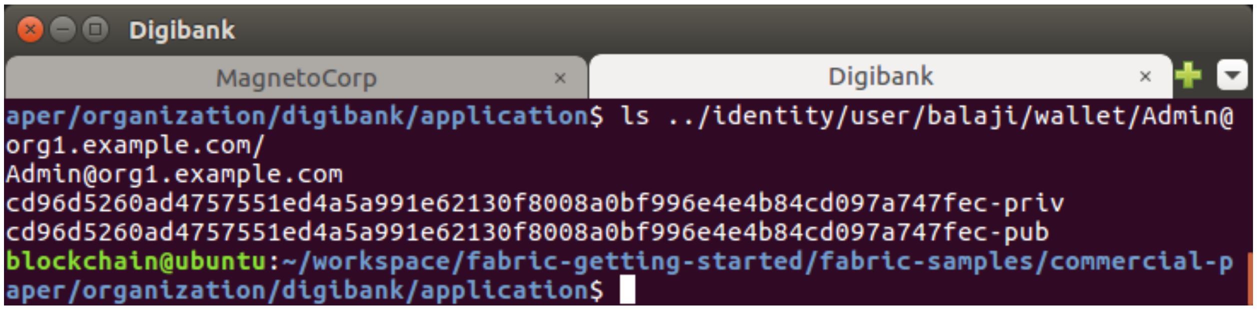


```
aper/organization/digibank/application$ node addToWallet.js
done
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ ls ../identity/user/balaji/wallet/
Admin@org1.example.com
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ 
```

This time you can see the identity is Admin@org1.example.com, as Balaji is acting as admin for DigiBank.

__ 36. Just like last time, we can also run another command, we can see the three files that make up the identity itself:

```
ls ../identity/user/balaji/wallet/Admin@org1.example.com
```

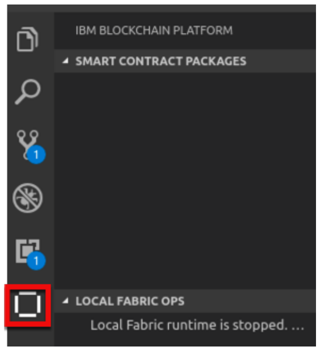


```
aper/organization/digibank/application$ ls ../identity/user/balaji/wallet/Admin@
org1.example.com/
Admin@org1.example.com
cd96d5260ad4757551ed4a5a991e62130f8008a0bf996e4e4b84cd097a747fec-priv
cd96d5260ad4757551ed4a5a991e62130f8008a0bf996e4e4b84cd097a747fec-pub
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$
```

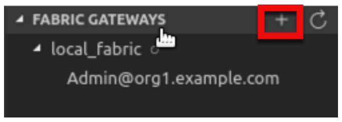
Again, these files consist of a private key for signing transactions, a public key linked to the private key and a file that contains both metadata and a certificate for our user.

Now **Balaji** from **DigiBank** would like to **buy** the commercial paper **00001**.

__ 37. Now we need to connect VSCode to the running **PaperNet** network we set up earlier in this lab. Return to VSCode and navigate to the IBM Blockchain Platform view.

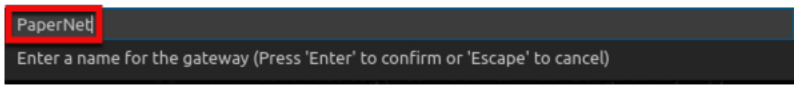


__ 38. Click on the **+** in the **FABRIC GATEWAYS** bar:

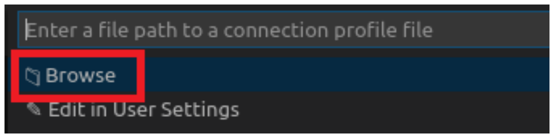


Note: the **+** only appears when you move your mouse over the **FABRIC GATEWAYS** bar.

__ 39. In the "**Enter a name for the gateway…**" pop up at the top of the screen, enter **PaperNet** and press **Enter**

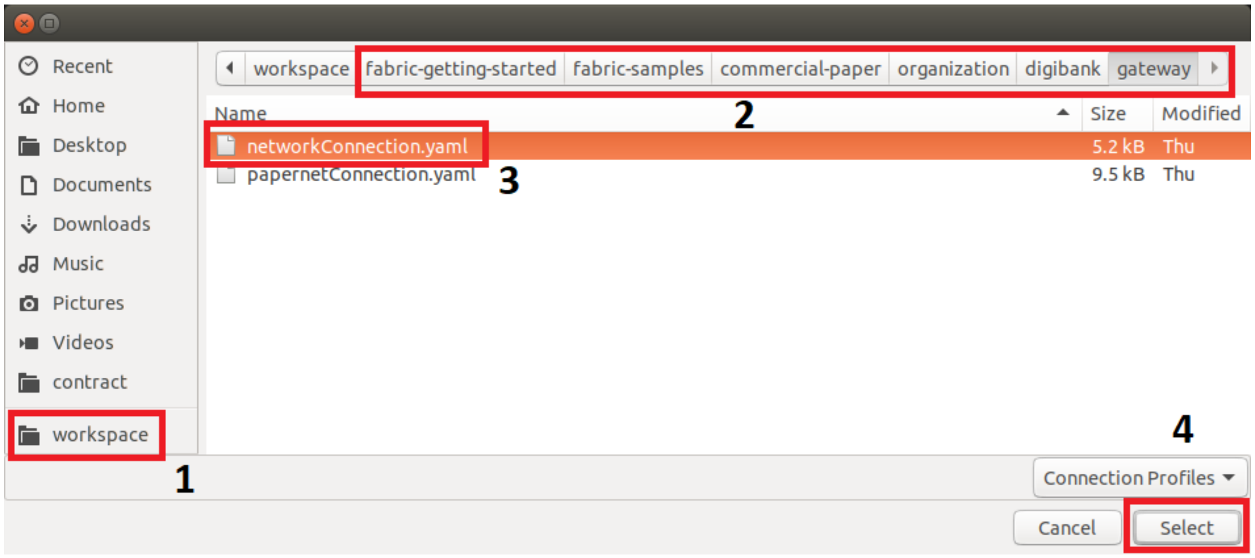

PaperNet
Enter a name for the gateway (Press 'Enter' to confirm or 'Escape' to cancel)

__ 40. In the "**Enter a file path to a connection profile file**" pop up at the top of the screen, click **Browse**:


Enter a file path to a connection profile file

__ 41. Using the screen shot below as a guide, navigate to open a connection profile as follows:

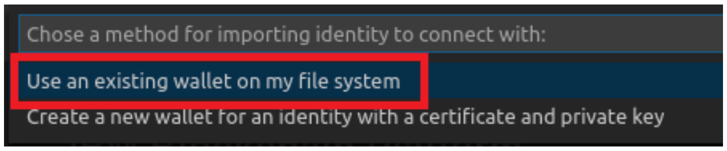
Step 1: Click in the **workspace** folder on the bottom left of the dialogue.
Step 2: Navigate to the folder: **`fabric-getting-started/fabric-samples/commercial-paper/organization/digibank/gateway/`**
Step 3: Select the file **`networkConnection.yaml`**
Step 4: Click the **Select** button on the bottom right.



Note: the full path to the **`networkConnection.yaml`** file you are importing for reference is: **/home/blockchain/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/digibank/gateway/`networkConnection.yaml`**

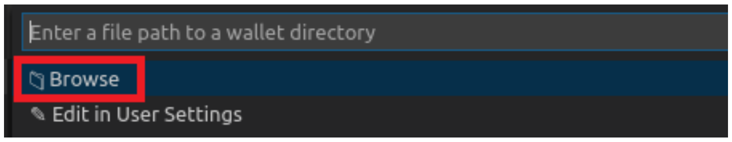
Make sure you choose the **digibank** folder and not the **magnetocorp** one. And make sure you choose the **networkConnection.yaml** file.

__ 42. In the "**Choose a method for importing identity to connect with**" pop up at the top of the screen click "**Use an existing wallet on my file system**"



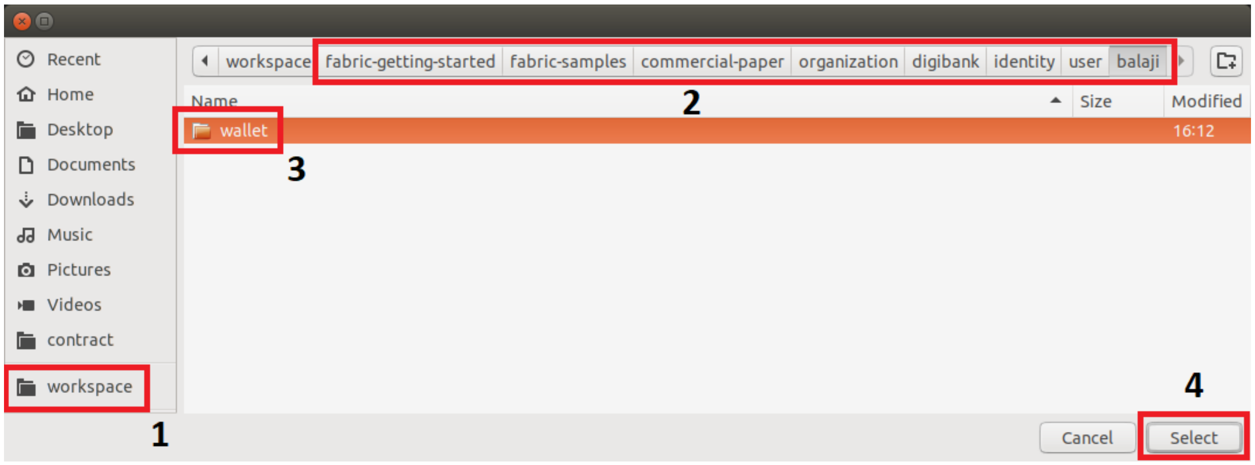
Chose a method for importing identity to connect with:
Use an existing wallet on my file system
Create a new wallet for an identity with a certificate and private key

__ 43. In the “**Enter a file path to a wallet directory**” pop up at the top of the screen click **Browse**:



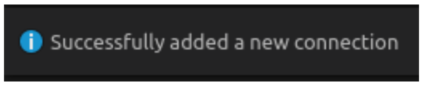
__ 44. Using the screen shot below as a guide, navigate to open a wallet filer as follows:

Step 1: Click in the **workspace** folder on the bottom left of the dialogue.
Step 2: Navigate to the folder: **`fabric-getting-started/fabric-samples/commercial-paper/organization/digibank/identity/user/balaji/`**
Step 3: Select the folder **`wallet`**
Step 4: Click the **Select** button on the bottom right.

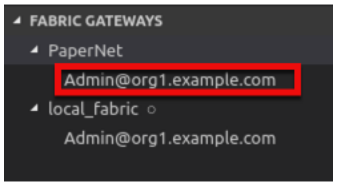

Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
contract
workspace
workspace | fabric-getting-started | fabric-samples | commercial-paper | organization | digibank | identity | user | balaji
Name
Size
Modified
wallet
16:12
Cancel
Select
1
2
3
4

Note: the full path to the **`wallet`** folder you are importing for reference is: **/home/blockchain/workspace/fabric-getting-started/fabric-samples/commercial-paper/organization/digibank/identity/user/balaji/**

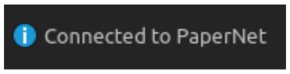
Make sure you choose the **digibank** folder and not the **magnetocorp** one. And make sure you choose the **wallet** folder itself and **not** the **Admin@org1.example.com** folder inside it. You should then see an information message saying that the connection was added successfully.


Successfully added a new connection

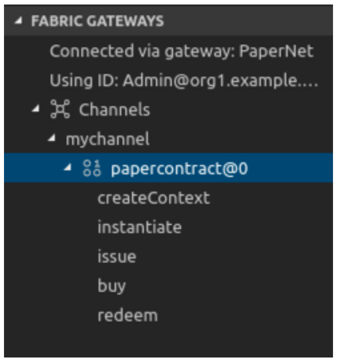
__ 45. From the FABRIC GATEWAYS view, click on Admin user under the newly create PaperNet connection to initiate a connection to PaperNet.


FABRIC GATEWAYS
PaperNet
Admin@org1.example.com
local_fabric ○
Admin@org1.example.com

Once you are connection, you will see an informational message telling you the connection was successful:

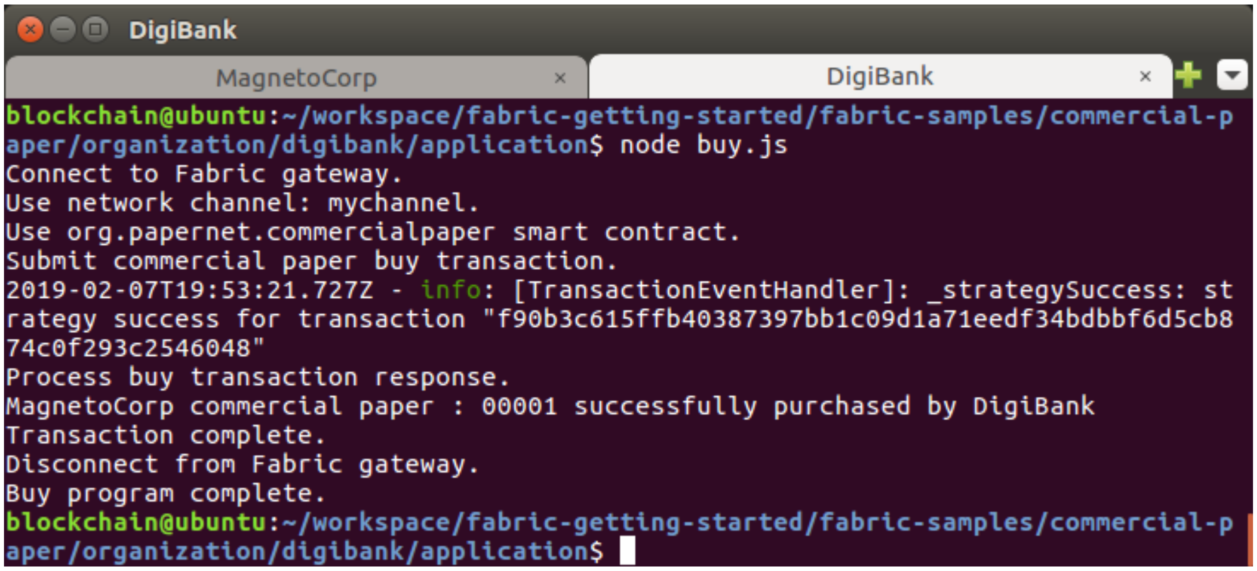

Connected to PaperNet

__ 46. Expand the channel **mychannel** and the previously instantiated **papercontract@0** contract to see the transactions available:


FABRIC GATEWAYS
Connected via gateway: PaperNet
Using ID: Admin@org1.example....
Channels
mychannel
papercontract@0
createContext
instantiate
issue
buy
redeem

__ 47. Run the **buy** application in the DigiBank terminal window:

```
node buy.js
```

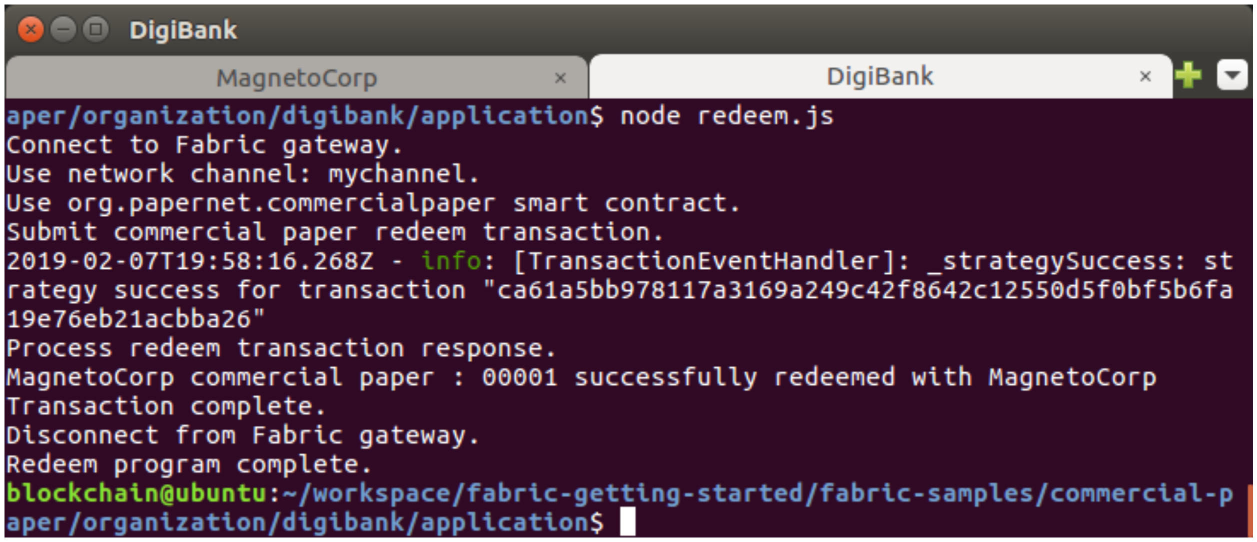


```
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$ node buy.js
Connect to Fabric gateway.
Use network channel: mychannel.
Use org.papernet.commercialpaper smart contract.
Submit commercial paper buy transaction.
2019-02-07T19:53:21.727Z - info: [TransactionEventHandler]: _strategySuccess: st
rategy success for transaction "f90b3c615ffb40387397bb1c09d1a71eedf34bdbbf6d5cb8
74c0f293c2546048"
Process buy transaction response.
MagnetoCorp commercial paper : 00001 successfully purchased by DigiBank
Transaction complete.
Disconnect from Fabric gateway.
Buy program complete.
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$
```

As you can see the transaction went through successfully, and paper **0001** is now owned by DigiBank.

__ 48. Next, let's assume that the maturity date has been reached and **DigiBank** wants to redeem the paper. Let's run the redeem application as Balaji:

`node redeem.js`



```
aper/organization/digibank/application$ node redeem.js
Connect to Fabric gateway.
Use network channel: mychannel.
Use org.papernet.commercialpaper smart contract.
Submit commercial paper redeem transaction.
2019-02-07T19:58:16.268Z - info: [TransactionEventHandler]: _strategySuccess: st
rategy success for transaction "ca61a5bb978117a3169a249c42f8642c12550d5f0bf5b6fa
19e76eb21acbba26"
Process redeem transaction response.
MagnetoCorp commercial paper : 00001 successfully redeemed with MagnetoCorp
Transaction complete.
Disconnect from Fabric gateway.
Redeem program complete.
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/digibank/application$
```

Again the transaction has run smoothly, and the paper was redeemed with MagnetoCorp.

Now we have most things working, let's issue a new commercial paper, but this time we will go through the steps a little faster.

__ 49. Switch to the **MagnetoCorp** terminal tab and open the issue application in VSCode so we can edit it for a new paper:

`code issue.js`

Edit the **submitTransaction** call on **line 68** so we can see differences from the previous one. Enter **00002** for the paper number to match the rest of this lab. A valid set of options would be:

```
'issue', 'MagnetoCorp','00002','2019-06-30','2019-12-30','6000000'
```

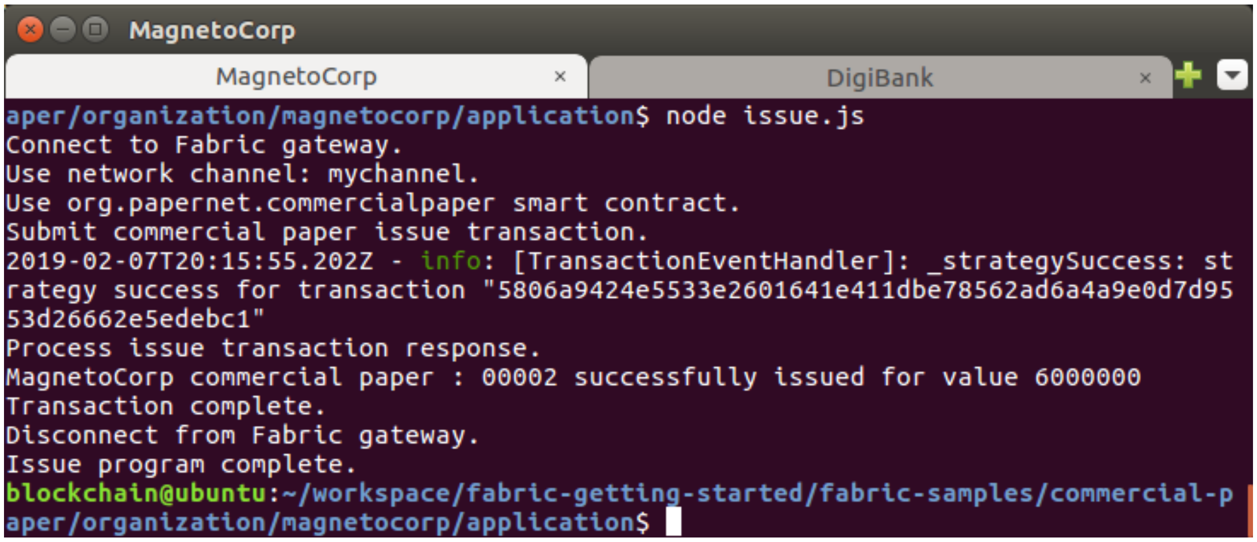
These will issue paper **00002** at a face value of **6,000,000** USD.

__ 50. Make sure you save the changes, using the **File / Save** option or press **ctrl + s**

__ 51. Run the issue command again:

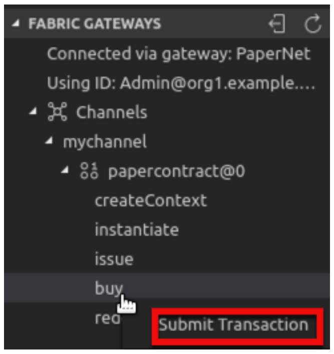
```
node issue.js
```

If you edited the file correctly, the transaction will go through and you can see the new paper was issued matching the new details:

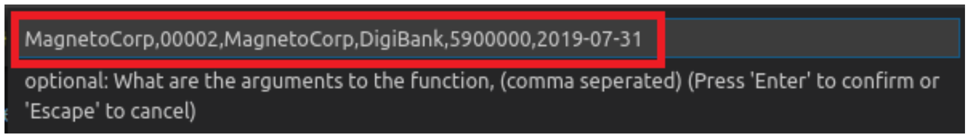


```
aper/organization/magnetocorp/application$ node issue.js
Connect to Fabric gateway.
Use network channel: mychannel.
Use org.papernet.commercialpaper smart contract.
Submit commercial paper issue transaction.
2019-02-07T20:15:55.202Z - info: [TransactionEventHandler]: _strategySuccess: st
rategy success for transaction "5806a9424e5533e2601641e411dbe78562ad6a4a9e0d7d95
53d26662e5edebc1"
Process issue transaction response.
MagnetoCorp commercial paper : 00002 successfully issued for value 6000000
Transaction complete.
Disconnect from Fabric gateway.
Issue program complete.
blockchain@ubuntu:~/workspace/fabric-getting-started/fabric-samples/commercial-p
aper/organization/magnetocorp/application$
```

__ 52. This time we will issue the buy transaction from VSCode. Right click on the **buy** transaction and choose **Submit Transaction**:



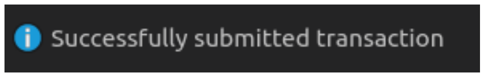
__ 53. In the dialogue at the top of the screen enter the text **MagnetoCorp,00002,MagnetoCorp,DigiBank,5900000,2019-07-31** and press "Enter". Note that you should not enter any quotes or spaces around this string as otherwise they will be taken as part of the string itself which will result in an error.



MagnetoCorp,00002,MagnetoCorp,DigiBank,5900000,2019-07-31

optional: What are the arguments to the function, (comma seperated) (Press 'Enter' to confirm or 'Escape' to cancel)

When the transaction has finished, you should see the information message:

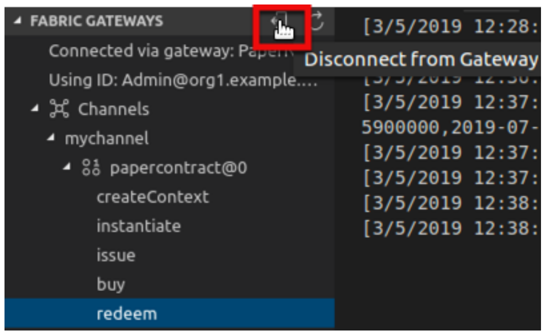


Successfully submitted transaction

Note: If instead you get an error, you may have entered the string incorrectly. You can check the **docker log output window** for an error and try this step again.

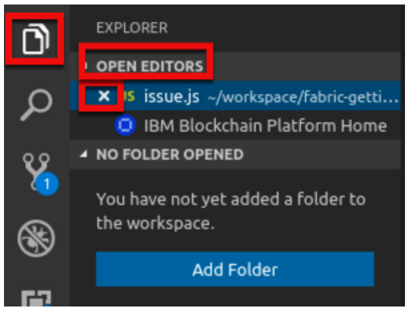
__ 54. Finally, repeat the above two steps, but this time call the **redeem** transaction, passing the parameters **MagnetoCorp,00002,DigiBank,2019-12-30** to it instead

__ 55. From the IBP **FABRIC GATEWAY** view, select the **Disconnect from Gateway** icon as shown below:



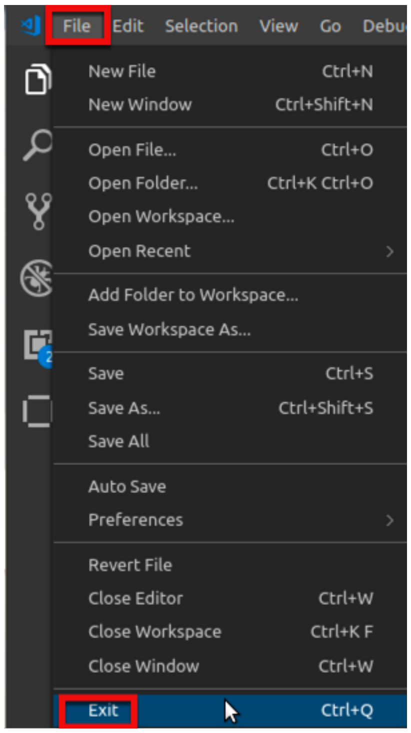
```
[3/5/2019 12:28:
[3/5/2019 12:37:
5900000,2019-07-
[3/5/2019 12:37:
[3/5/2019 12:37:
[3/5/2019 12:38:
[3/5/2019 12:38:
```

__ 56. Switch back to the **Explorer** view and close all open editors in the **Open Editors** view, including the "**IBM Blockchain Platform Home**" by clicking on the "**x**" button on each one in turn:


EXPLORER
OPEN EDITORS
issue.js ~/workspace/fabric-getti…
IBM Blockchain Platform Home
NO FOLDER OPENED
Add Folder

You have not yet added a folder to the workspace.

__ 57. Exit VSCode by selecting **File->Exit** from the File menu.



__ 58. We have now completed this lab **– Connecting to an Existing Network** and we hope you enjoyed it. If you have any time remaining, please feel free to experiment further on your own, maybe changing one of the command line applications.