IBM Blockchain Platform Hands-On

Lab 5:

IBM Blockchain Platform for Multicloud Operations Lab

Table of Contents

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and

discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

1 Overview of the lab environment and scenario

This lab is a guide to using the IBM Blockchain Platform operations console. In this lab, you will work with a partner to **Build** a new two organization network using the console and get your partner to **Join** it. You will then deploy an existing smart contract, issue a transaction between the two organizations and see the results.

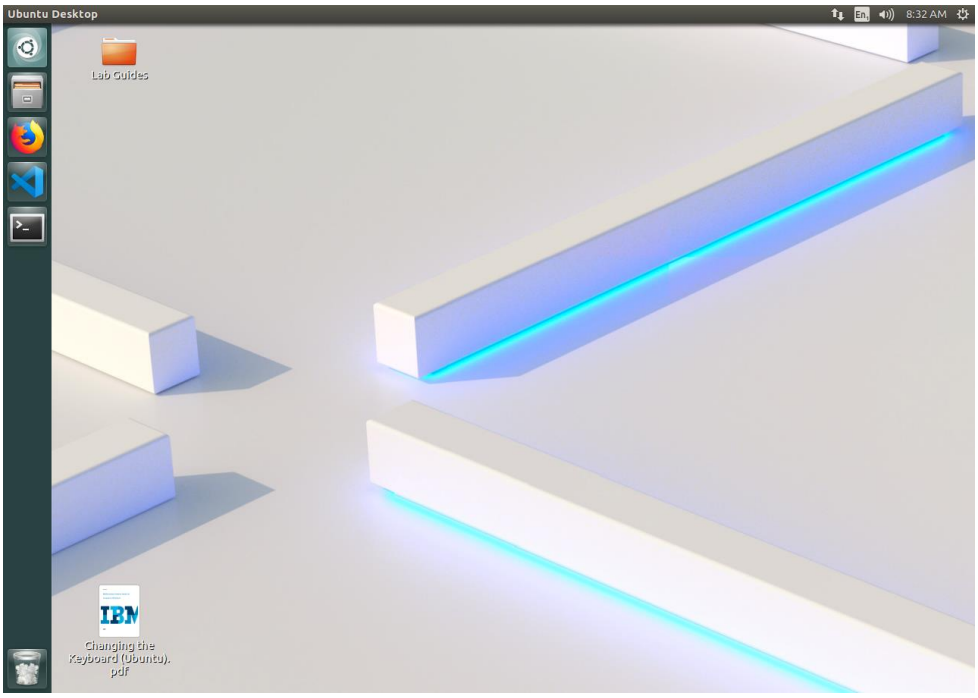
***Note**: The screenshots in this lab guide were taken using version **1.39.2** of **VS Code**, version **1.0.12** of the **IBM Blockchain Platform** plugin and version **0.3.50** of the **IBM Blockchain Platform** console. If you use different versions, you may see differences to those shown in this 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.

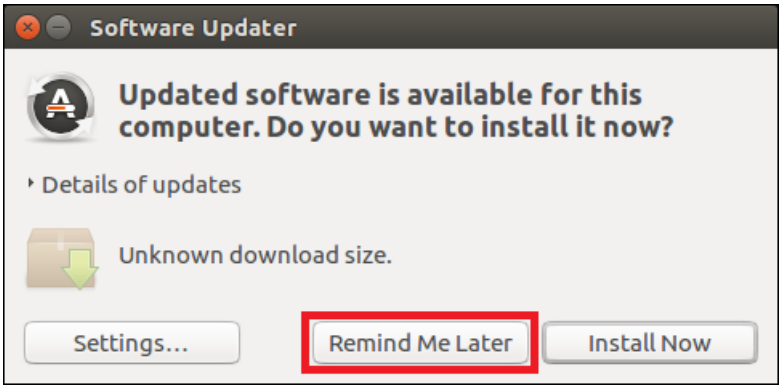
Note: If it asks you to login, the userid and password are both "**blockchain**".



1.1 Lab Scenario

In this lab, we will be creating a new network on the **IBM Blockchain Platform for Multicloud** running on **OpenShift Origin**, also known as **OKD**, which is the open source version of **Red Hat OpenShift**.

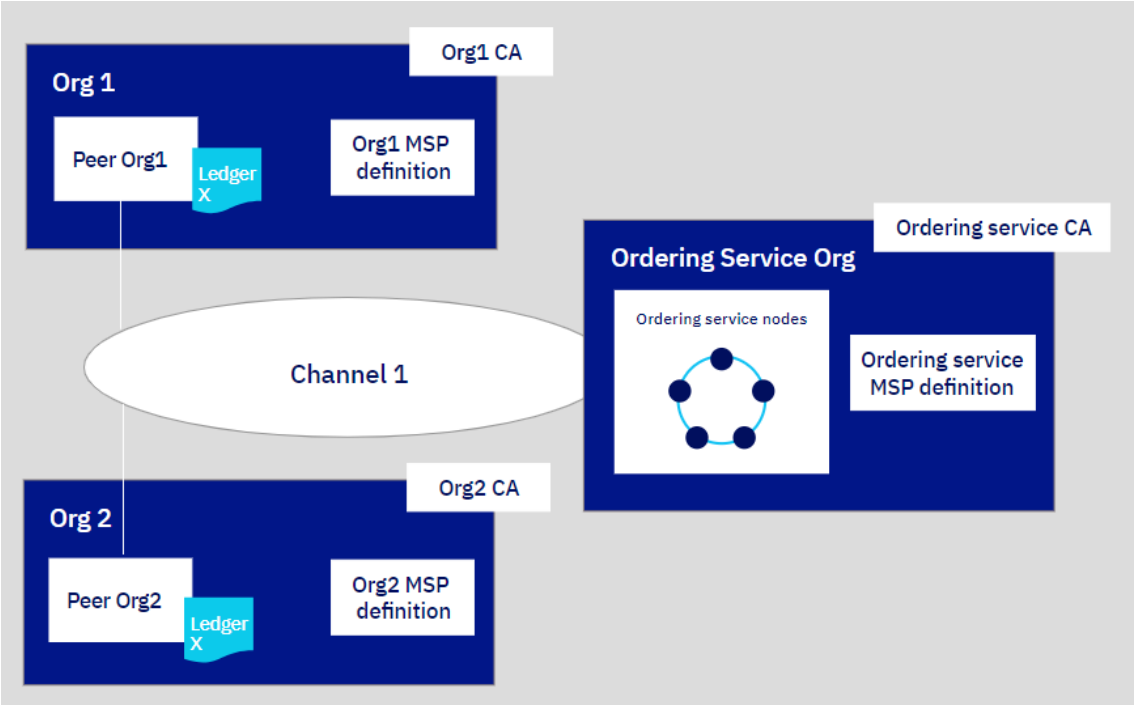
Note *that if you get a "**Software Updater**" pop-up at any point during the lab, please click "**Remind Me Later**":*



2 IBM Blockchain Platform Operations

As mentioned above, in this lab we will be building a new network containing two new organizations using the IBM Blockchain platform **Build** and **Join** tutorials which we will look at shortly.

The network we will be building looks like this:



Although you can build this network on your own, we recommend you build it in pairs with one person taking on the role of Org1 and the Ordering Service Org and the second person taking on the role of Org2.

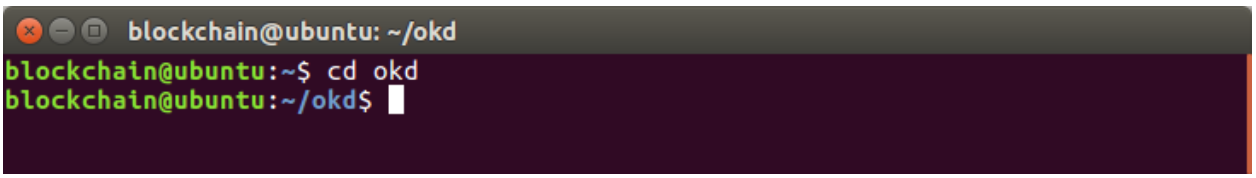
So in pairs, decide who is going to represent **Org1** and who is going to represent **Org2**. Both orgs will have a lot to do, but we will start out with representatives from both orgs provisioning their IBM Blockchain Platform service on OKD which is already installed on the VM.

2.1 Setting up the IBM Blockchain Platform.

This section must be completed by BOTH Org1 and Org2 working independently.

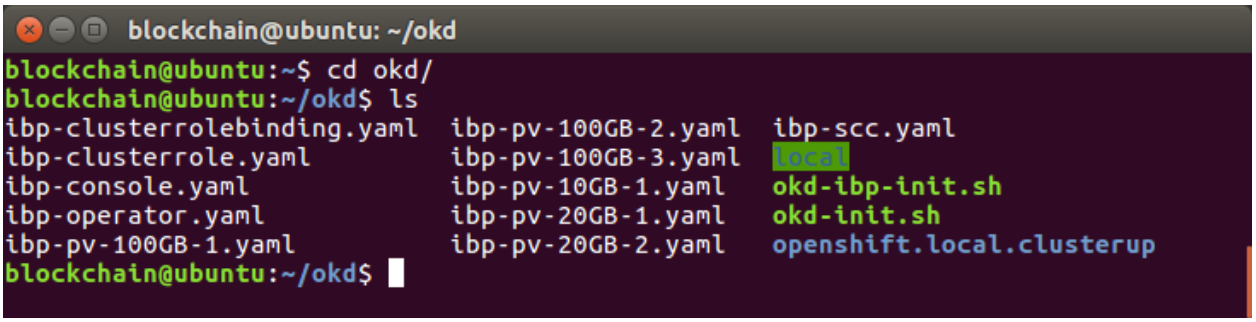
__ **3.** Open a new terminal prompt and change to the **okd** folder in your home directory:

```
cd okd
```



__ **4.** Run the ls command to look at the files available

```
ls
```



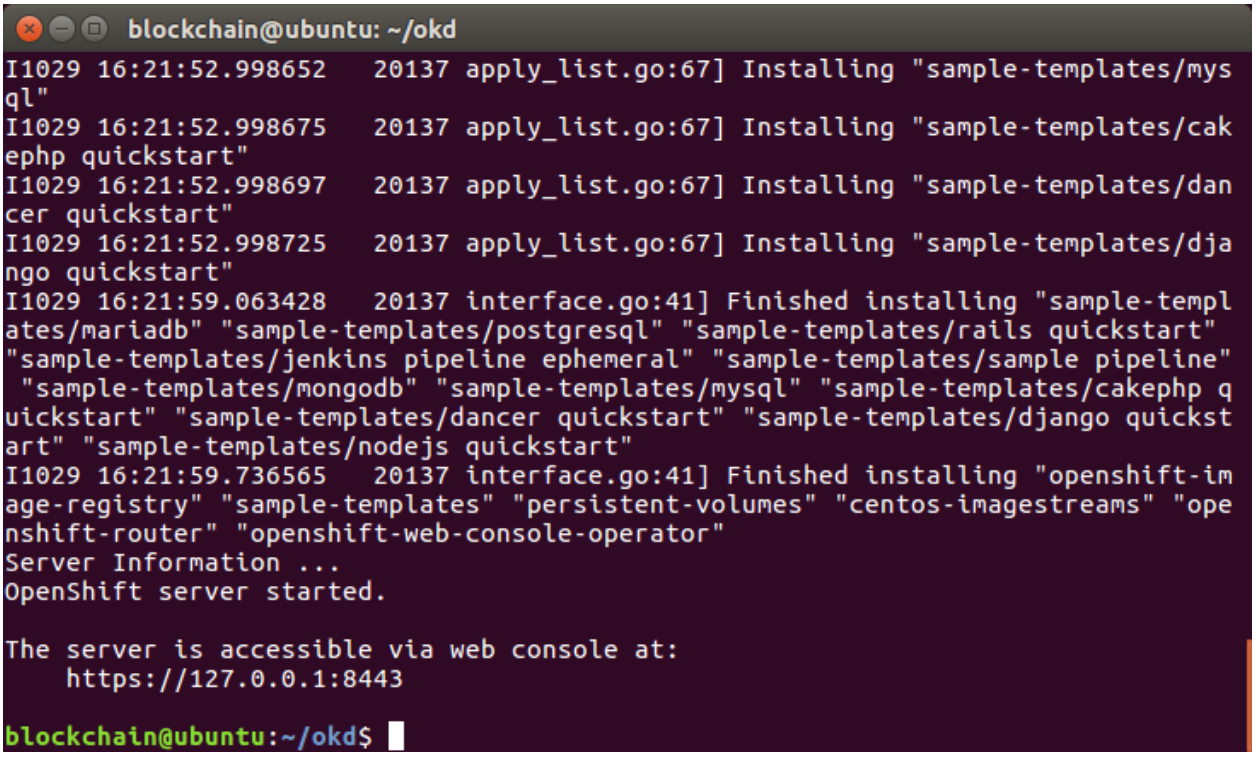
Here you can see several files that make up the configuration of IBP and OKD along with a couple of scripts to set things up.

OKD provides management for groups of containers by grouping them into pods and clusters. In this lab we will use a single node cluster to manage IBM Blockchain Platform. The cluster we will use is pre-configured in the lab with the containers that make up IBM Blockchain Platform, so now we need to bring the cluster up.

__ **5.** Let's start the OKD cluster by running the main OKD administration command called "**oc**". In the terminal enter:

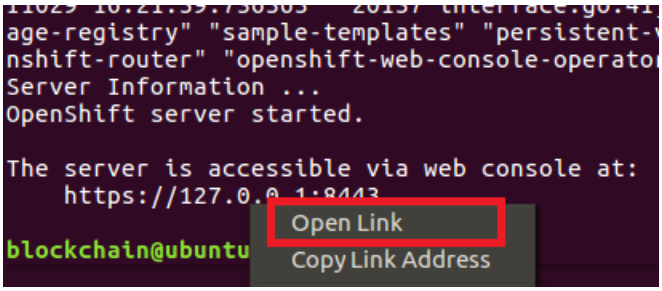
```
oc cluster up
```

This command will take a few minutes to run and will produce several screens worth of output. When it has finished your terminal should look like this:

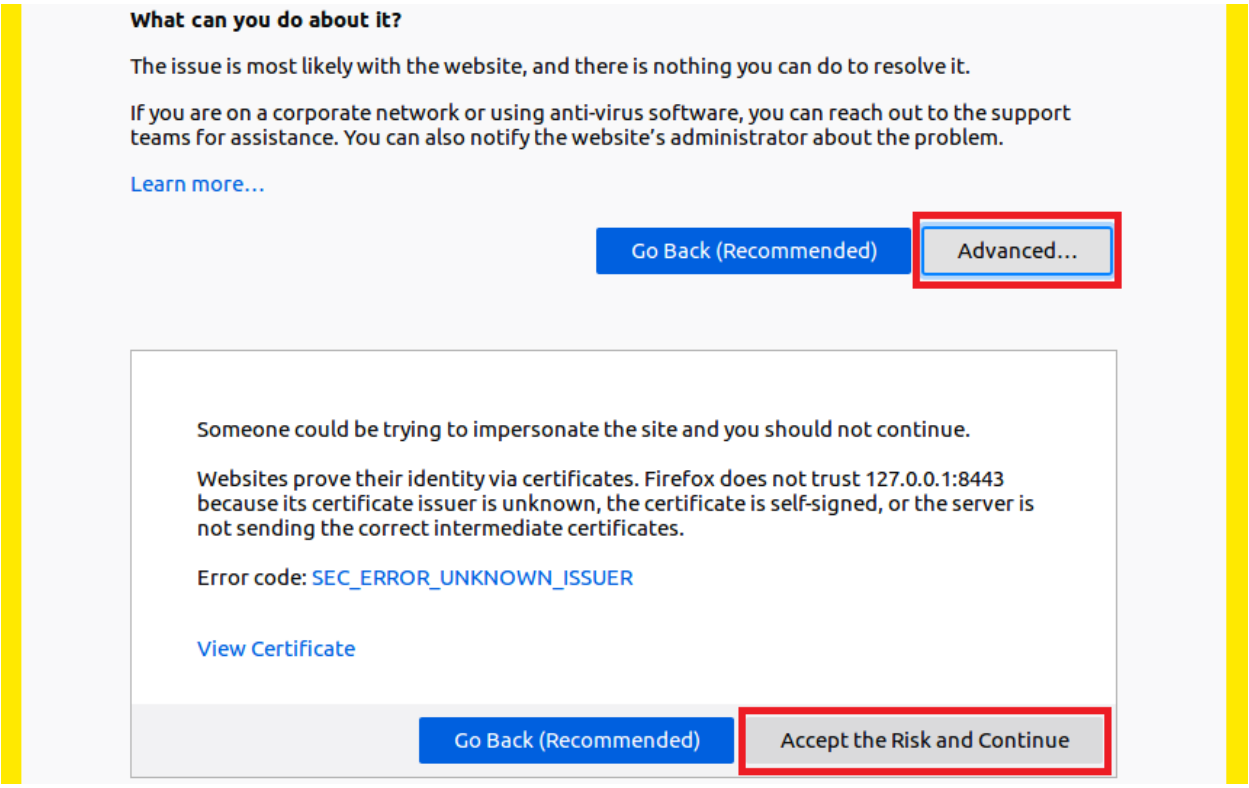


Note: If you have used kubernetes before, the **oc** command is like an enhanced **kubectl** command.

__ **6.** At the end of the output, there is a link to the OKD admin console. Right-click on the link and choose "**Open Link**":



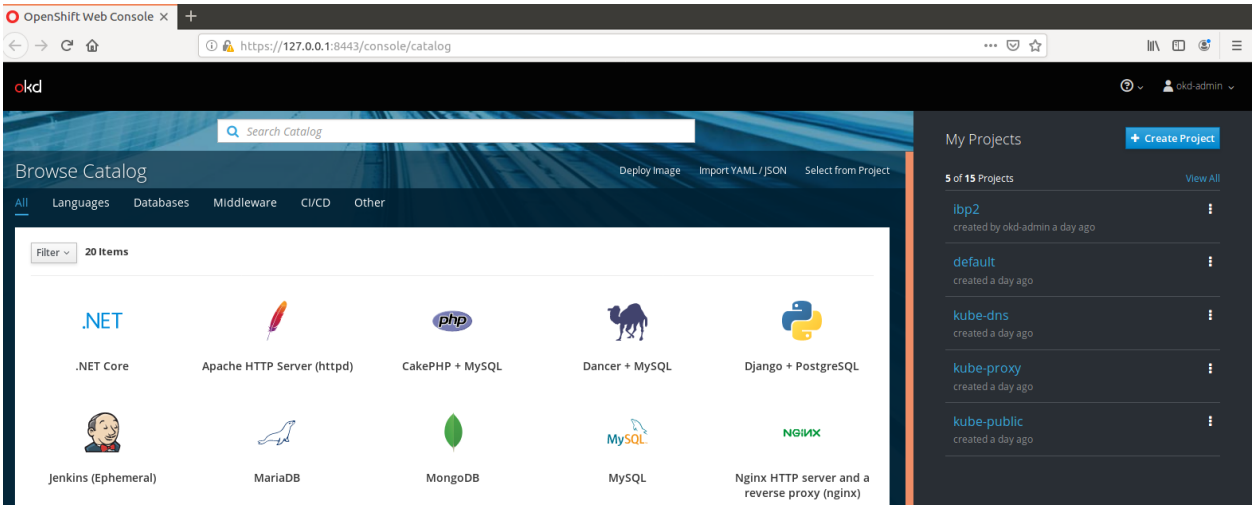
__ **7.** At this point Firefox will open and you should see a security warning in Firefox. This is because the default setup generates self-signed certificates which Firefox does not recognise. We need to accept the certificate to continue, so click the "**Advanced...**" button to expand the information then click the "**Accept the Risk and Continue**" button.



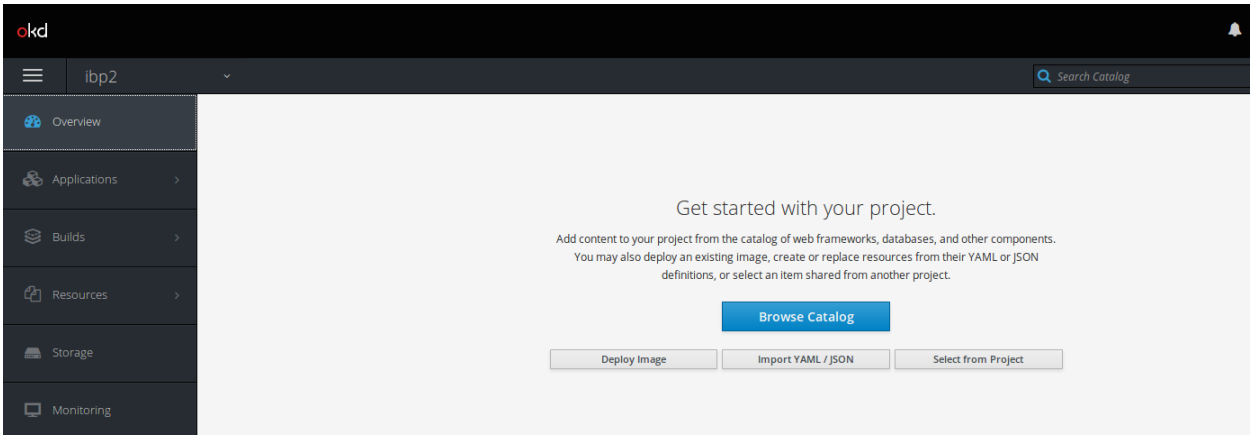
__ **8.** Firefox will load the OKD console login screen. Enter the Username **okd-admin** and a Password of **12345678** and click "**Log In**":

__ **9.** Firefox will ask you if you wish to save the password, click **Save** to continue.

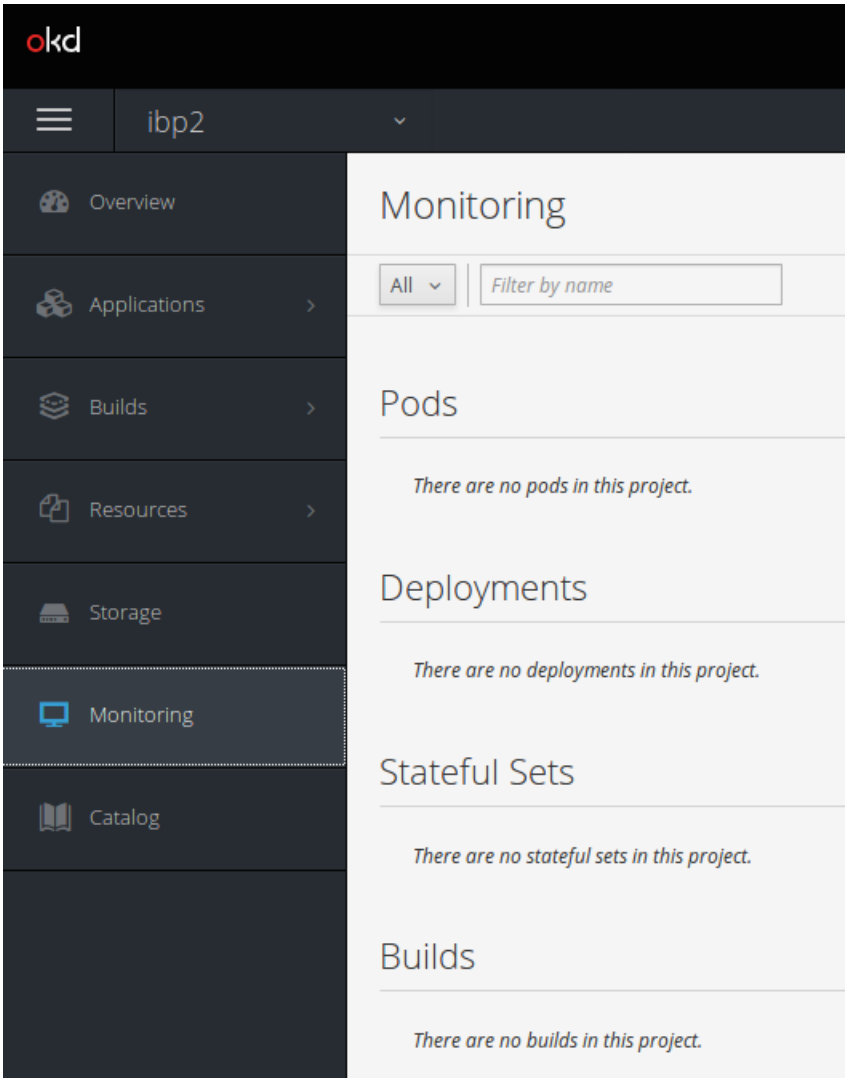
__ **10.** When you are logged in, the main console will appear. On the right-hand side of the console you will see a list of existing projects. Click on the existing "**ibp2**" project:



__ **11.** As the project is currently empty, you will see the "Getting started…" page. From this page click on the "**Monitoring**" tab towards the bottom on the left-hand side:



The "**Monitoring**" tab confirms that the project is currently empty, but we are going to deploy some new artefacts that will start to appear on this page.



IBP Blockchain Platform for Multicloud has two main components, an "**operator**" and a "**console**". The operator's job is to monitor the console and help it perform its duties. We now need to start deploying these components.

__ **12.** Go back to your terminal and run the "**okd-ibp-init.sh**" script to perform the initial set-up and deploy the operator:

```
./okd-ibp-init.sh
```

This command will run quickly but will produce a screen or two of output. When it has finished you terminal will look like this:

```
blockchain@ubuntu: ~/okd
    openshift-infra
    openshift-node
    openshift-service-cert-signer
    openshift-web-console

Using project "ibp2".
==> set project to ibp2
Already on project "ibp2" on server "https://127.0.0.1:8443".
==> apply ibp policy
securitycontextconstraints.security.openshift.io/ibp2 configured
scc "ibp2" added to: ["system:serviceaccounts:ibp2"]
clusterrole.rbac.authorization.k8s.io/ibp2 configured
scc "ibp2" added to groups: ["system:serviceaccounts:ibp2"]
clusterrolebinding.rbac.authorization.k8s.io/ibp2 configured
cluster role "ibp2" added: "system:serviceaccounts:ibp2"
==> delete and recreate the docker-key-secret
secret "docker-key-secret" deleted
secret/docker-key-secret created
==> deploy the ibp-operator pod
deployment.apps/ibp-operator created
==> get the deployments for the ibp2 project
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
ibp-operator   1         1         1            0           0s
blockchain@ubuntu:~/okd$
```

Note: Make sure you have the right script name (which must have **ibp** in its name) and that you type the leading period and forward slash "**./** "

You can see from the last two lines of output that there are 6 columns. The first gives the name of the deployment we just made which is "**ibp-operator**". The next 3 columns show we need **1** up-to-date copy of it running, but currently there are **0** available. This is because the operator takes a little time to start in the background.

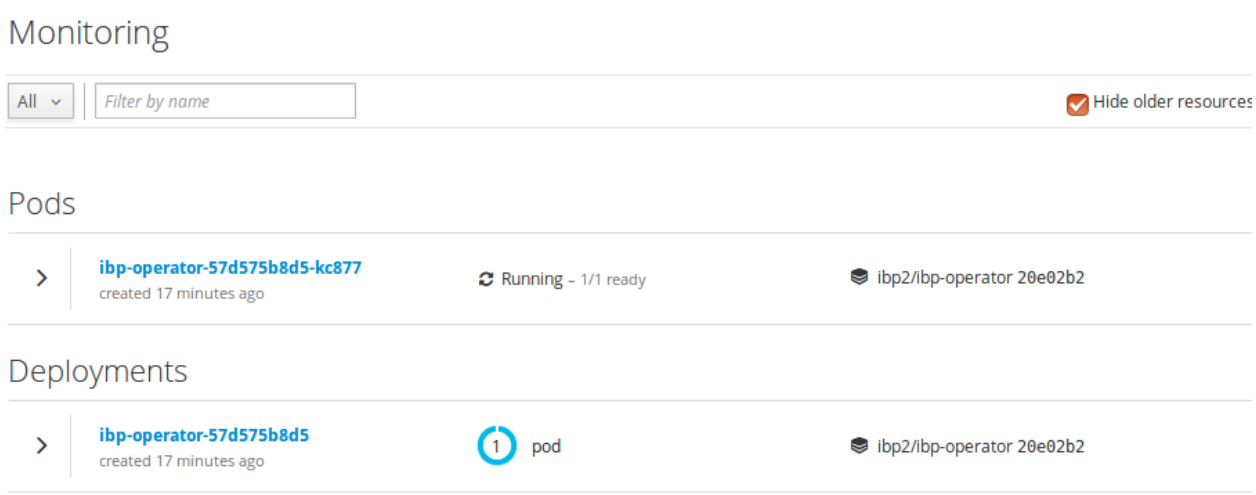
We will now keep checking the operator until it becomes available.

__ **13.** Run the command below in the terminal every 20 seconds or so until the "**available**" column changes from a **0** to a **1** to show the operator has started. It should only take one or two attempts as shown below:

```
oc get deployment -n ibp2
```

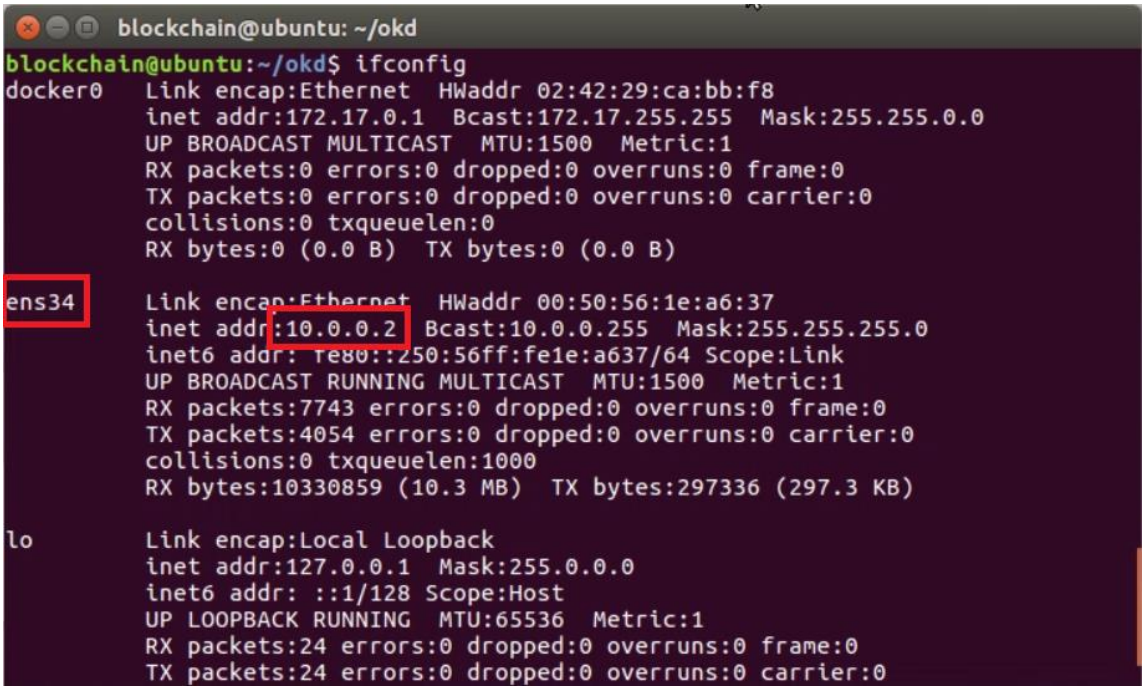
```
blockchain@ubuntu: ~/okd
    openshift-web-console

Using project "ibp2".
==> set project to ibp2
Already on project "ibp2" on server "https://127.0.0.1:8443".
==> apply ibp policy
securitycontextconstraints.security.openshift.io/ibp2 configured
scc "ibp2" added to: ["system:serviceaccounts:ibp2"]
clusterrole.rbac.authorization.k8s.io/ibp2 configured
scc "ibp2" added to groups: ["system:serviceaccounts:ibp2"]
clusterrolebinding.rbac.authorization.k8s.io/ibp2 configured
cluster role "ibp2" added: "system:serviceaccounts:ibp2"
==> delete and recreate the docker-key-secret
secret "docker-key-secret" deleted
secret/docker-key-secret created
==> deploy the ibp-operator pod
deployment.apps/ibp-operator created
==> get the deployments for the ibp2 project
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
ibp-operator   1         1         1            0           0s
blockchain@ubuntu:~/okd$ oc get deployment -n ibp2
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
ibp-operator   1         1         1            1           54s
blockchain@ubuntu:~/okd$
```

__ **14.** If you now look at the monitoring output in Firefox again, you should see the **Monitoring** page is populated with information about the deployment we just made, and the **ibp-operator** pod we just created:



__ **15.** Before we can deploy the IBP Console, we need to update one of the configuration files to make sure it reflects the VM's current IP address. To get the current IP address, run this command in the terminal:

```
ifconfig
```



Several addresses are shown, and you will need to scroll back through the output to find the right one. The IP address you need is the one belonging to the interface with the name starting with "**en**". In the example above, it's "**ens34**". You may need to scroll back up to see it. If you are running in a **SkyTap** environment, your address will normally be either **10.0.0.1** or **10.0.0.2**. In the example above, the address is **10.0.0.2**

Note: You need to remember or write down your IP address as you will need it again later on in this lab.

Now we have the IP address, we need to update the config file with this information in the next step.

__ **16.** From the terminal enter the following command to open the file in VS Code:

code ibp-console.yaml

```
okd > ! ibp-console.yaml
50          1.4.3-0:
51            default: true
52            version: 1.4.3-0
53            image:
54              ordererInitImage: 172.30.1.1:5000/ibp2/ibp-init
55              ordererInitTag: 2.1.0-20190924-amd64
56              ordererImage: 172.30.1.1:5000/ibp2/ibp-orderer
57              ordererTag: 1.4.3-20190924-amd64
58              grpcwebImage: 172.30.1.1:5000/ibp2/ibp-grpcweb
59              grpcwebTag: 2.1.0-20190924-amd64
60      networkinfo:
61        domain: 10.0.0.1.nip.io
62      storage:
63        console:
64          class: default
65          size: 10Gi
```

__ **17.** Scroll to the bottom of the file to find the **networkinfo** section around **line 60**. Then edit the "**domain**" entry on the next line to be your IP address, leaving the **.nip.io** suffix in place. For example, if the domain in the file is **10.0.0.1.nip.io**, and your IP address is **10.0.0.2**, then change the domain to be **10.0.0.2.nip.io** as shown below:

```
! ibp-console.yaml ●
okd > ! ibp-console.yaml
50          1.4.3-0:
51            default: true
52            version: 1.4.3-0
53            image:
54              ordererInitImage: 172.30.1.1:5000/ibp2/ibp-init
55              ordererInitTag: 2.1.0-20190924-amd64
56              ordererImage: 172.30.1.1:5000/ibp2/ibp-orderer
57              ordererTag: 1.4.3-20190924-amd64
58              grpcwebImage: 172.30.1.1:5000/ibp2/ibp-grpcweb
59              grpcwebTag: 2.1.0-20190924-amd64
60      networkinfo:
61        domain: 10.0.0.2.nip.io
62      storage:
63        console:
64          class: default
65          size: 10Gi
```

__ **18.** Save the file with **File** -> **Save** or press **Ctrl**+**s**. Saving the file will change its tab to remove the circle as shown:

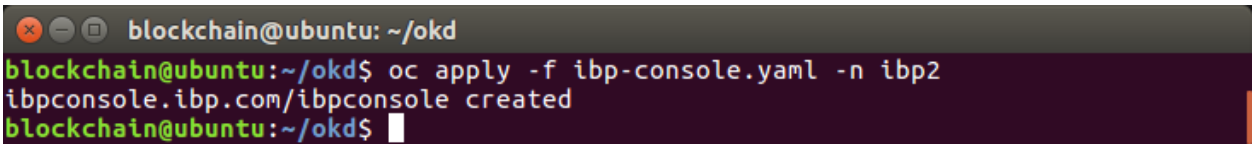
Not saved

Saved

Now we are ready to deploy the IBP console.

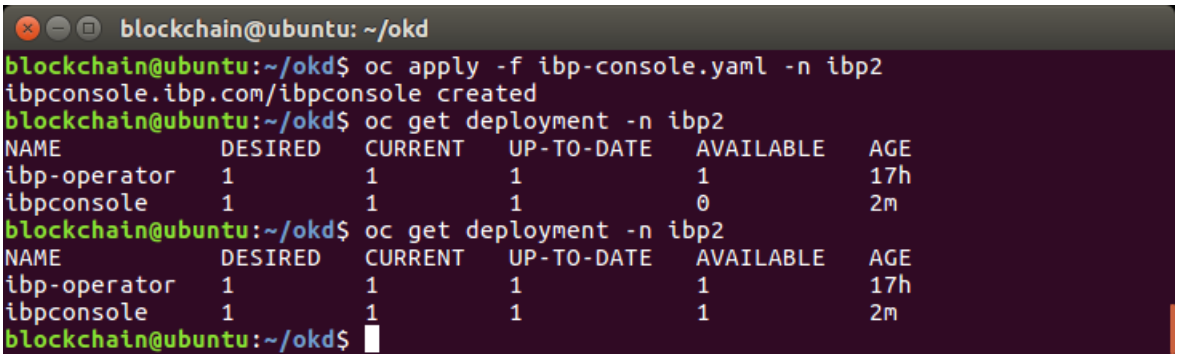
__ **19.** Switch back to the terminal window you already have open and enter the command to deploy the IBP console using the file you just updated:

```
oc apply -f ibp-console.yaml -n ibp2
```



__ **20.** Now the console has been created, we need to wait for it to finish deploying like we did earlier with the **operator**. Run the command below in the terminal every 30 seconds or so until the "**available**" column changes from a **0** to a **1** to show the **console** has started. The console can take longer than the operator to deploy and start, but it should only take a few attempts as shown below:

```
oc get deployment -n ibp2
```



__ **21.** Enter the following command:

```
oc get pods
```

This is a useful command that gets the available pods. It will also tell you when all the containers in the pod are up and running. You should expect to see **1/1** pods running for the **ibp-operator** and **4/4** running for the **ibp-console** as shown below:

```
blockchain@ubuntu: ~/okd
blockchain@ubuntu:~/okd$ oc get pods
NAME                              READY     STATUS    RESTARTS   AGE
ibp-operator-57d575b8d5-kc877     1/1       Running   0          20h
ibpconsole-7bbbc778c9-fx7tb       4/4       Running   0          2h
blockchain@ubuntu:~/okd$
```

Note: You may also see restarts shown as well, which is OK. The numbers refer to the number of main containers **expected** to be running and the number **actually** running.

__ **22.** To look at the logs for any of the containers, you can get the logs with this command which uses the output from the **oc get pods** command we ran above:

```
oc logs ibpconsole-7bbbc778c9-fx7tb
```

In this example, the number after "**ibpconsole-xxx**" must match the output from the **oc get pods** command above. When running this command as is, you will get a helpful error message saying you need to specify the container for which you want to get the logs:

```
blockchain@ubuntu: ~/okd
blockchain@ubuntu:~/okd$ oc get pods
NAME                              READY     STATUS    RESTARTS   AGE
ibp-operator-57d575b8d5-kc877     1/1       Running   0          20h
ibpconsole-7bbbc778c9-fx7tb       4/4       Running   0          2h
blockchain@ubuntu:~/okd$ oc logs -f ibpconsole-7bbbc778c9-fx7tb
Error from server (BadRequest): a container name must be specified for pod ibpco
nsole-7bbbc778c9-fx7tb, choose one of: [optools deployer configtxlator couchdb]
or one of the init containers: [init initcerts]
blockchain@ubuntu:~/okd$
```

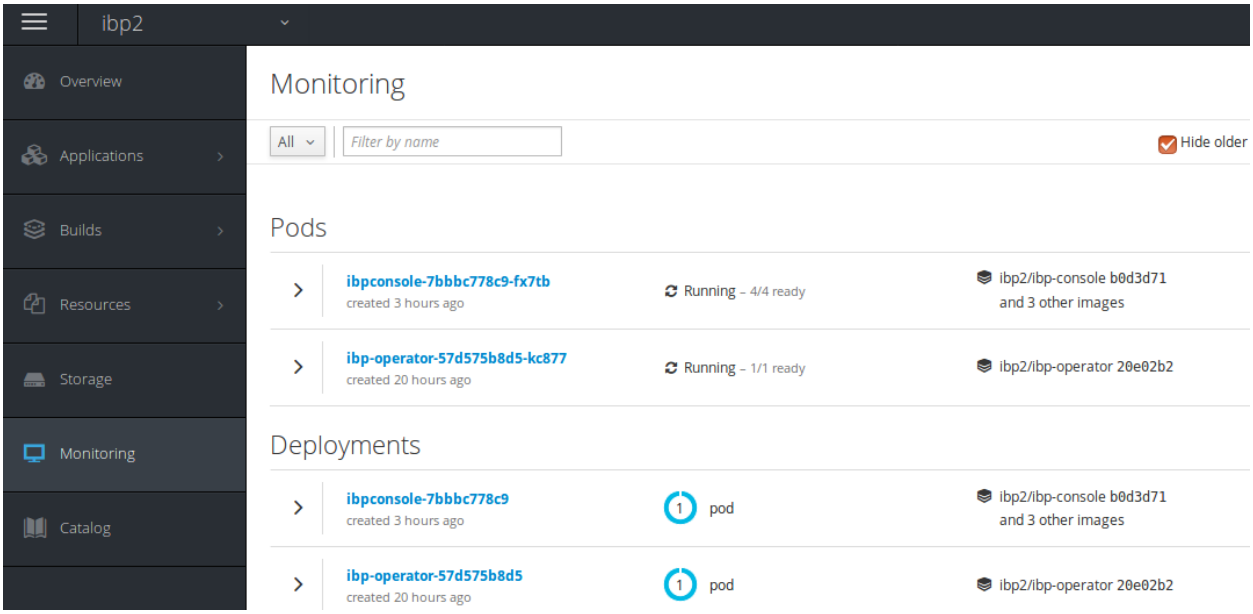
Therefore, you need to run the command again, specifying the actual container you wish to see, from the list shown in the output: **optools**, **deployer**, **configtxlator**, **couchdb** or one of the initialization containers **init** or **initcerts** like this:

```
oc logs ibpconsole-7bbbc778c9-fx7tb -c optools
```

The logs output will look similar to that shown below:

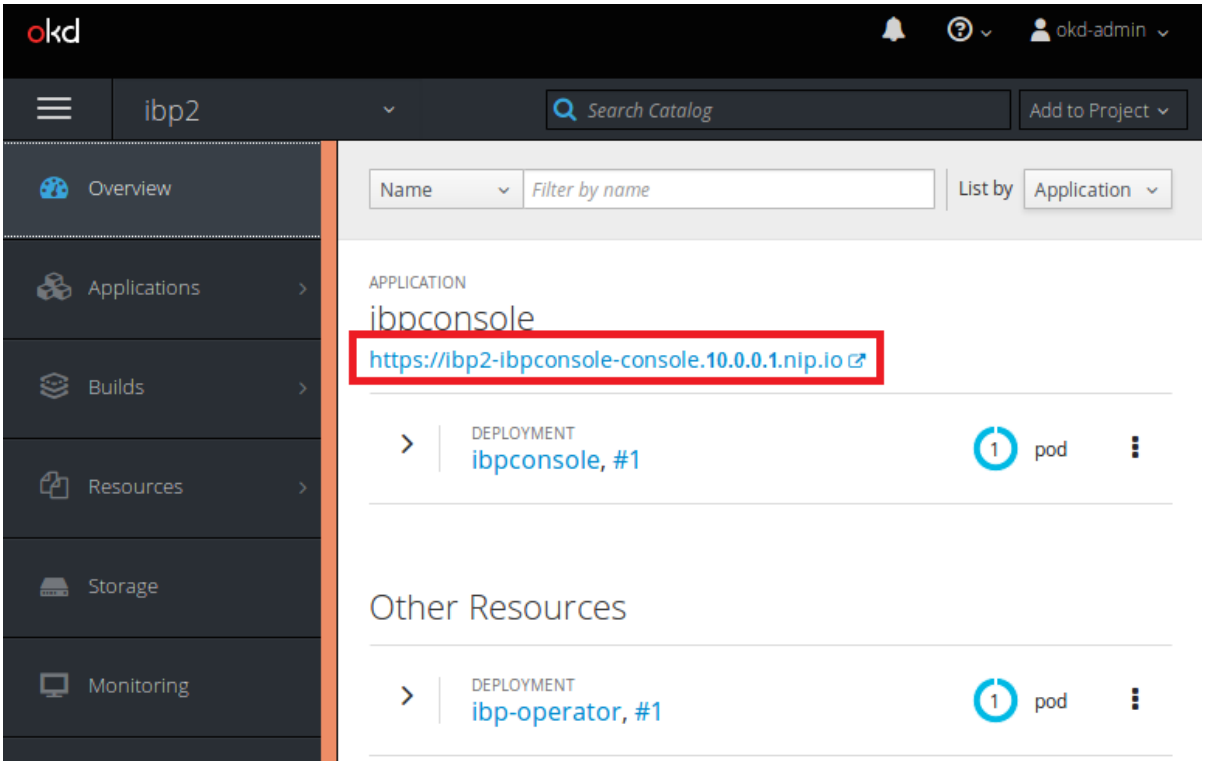
```
blockchain@ubuntu: ~/okd
14:46:21 - debug: Version file:
tag:
athena:c80e9d3
apollo:67519ea
stitch:5e4fcf7

14:46:21 - debug: [session store] session database found: athena-sessions
14:46:21 - debug: [couchdb rfo] sending GET req. timeout: 30.0 secs url: http://
localhost:5984/athena-sessions/_design/session_athena?null 1
14:46:21 - debug: [pillow] 0 old messages found
14:46:21 - warn: [couchdb rfo] status code: 404
14:46:21 - debug: [couchdb icn] sending POST req. timeout: 30.0 secs url: http:/
/localhost:5984/athena-sessions 1
14:46:21 - info: [pillow] starting couch connection. db: athena-system, "since":
 now
14:46:21 - debug: [session store] session design doc created/edited
14:46:23 - debug: [couchdb tse] sending GET req. timeout: 2.0 mins url: http://l
ocalhost:5984/athena-sessions/_design/session_athena/_view/by_expiration?endkey=
1572619583055&include_docs=true 1
14:46:23 - debug: [session store] deleting expired sessions via bulk delete
14:46:23 - debug: [session store] nothing to delete 0
14:46:26 - debug: [tls] the setting "dynamic_tls" is false. will not reload serv
er on tls cert changes
blockchain@ubuntu:~/okd$
```

__ **23.** Now we have the console up and running, we can also check with the **Monitoring** tab on in the OKD console in Firefox again. We would expect to see both the pods are running and there are the correct number of containers in each pod:



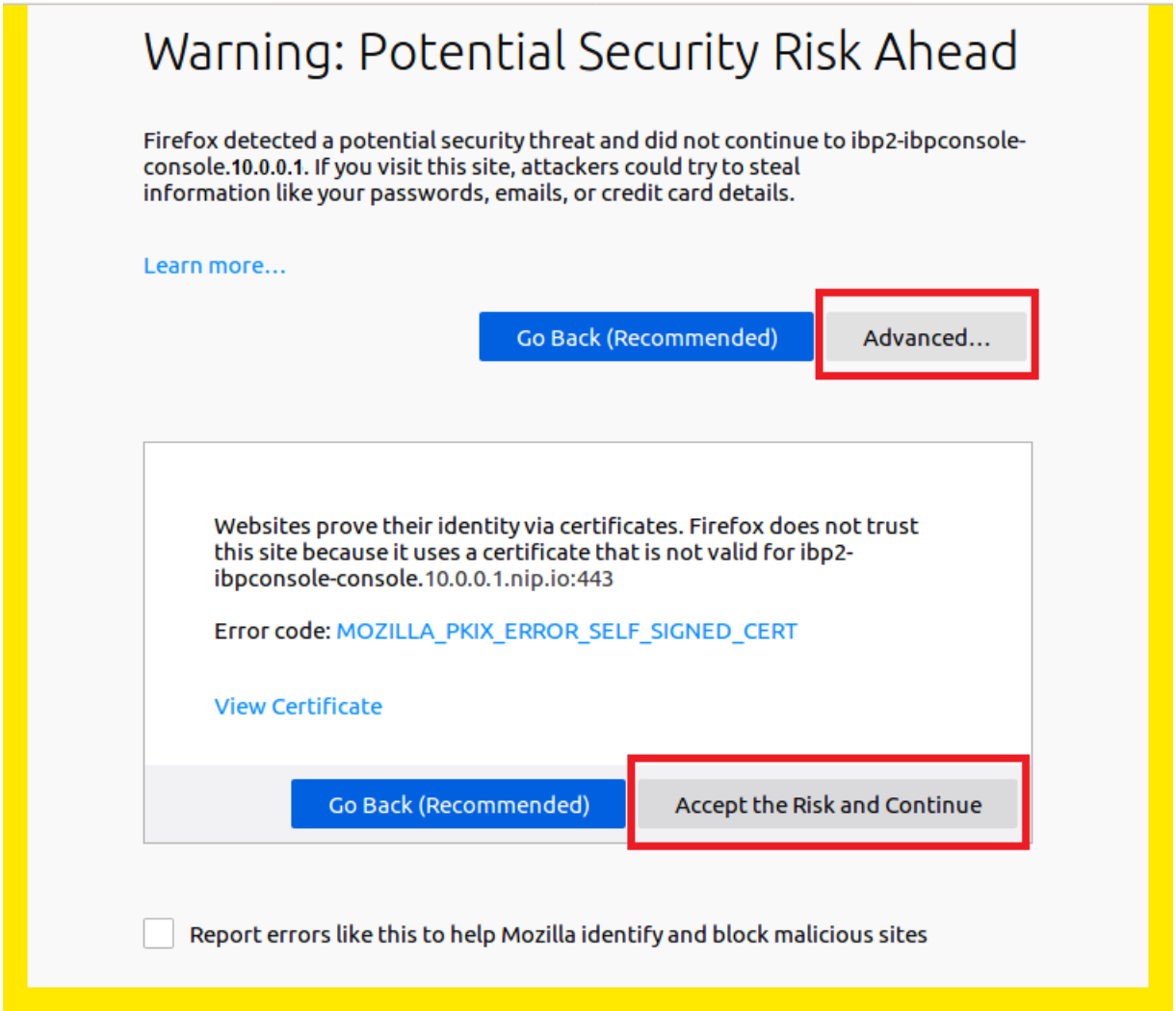
At this point there are just a couple of steps to go before we can start using the console.

__ **24.** In the OKD console, click on the "**Overview**" tab at the top on the left to see what is currently deployed into the "**ibp2**" project. You will see two deployments, one for the **ibp-operator** and one for the **ibpconsole**. On the far right of the **ibpconsole** you will see a URL which contains the IP address you entered earlier. Click on this link to open the IBP console page:



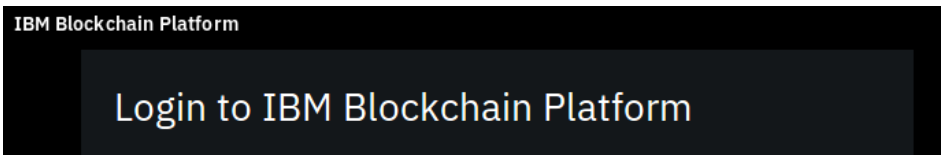
Note: The page may take a little time to load – refresh it if it seems to take too long.

__ **25.** At this point you should see a security warning in Firefox. This is because the default setup generates self-signed certificates which Firefox does not recognise. We need to accept the certificate to continue, so click the "**Advanced...**" button to expand the information then click the "**Accept the Risk and Continue**" button.



As you can see, the error code mentions the self-signed certificate as explained above.

You will then be presented with the "**Login to IBM Blockchain Platform**" page:



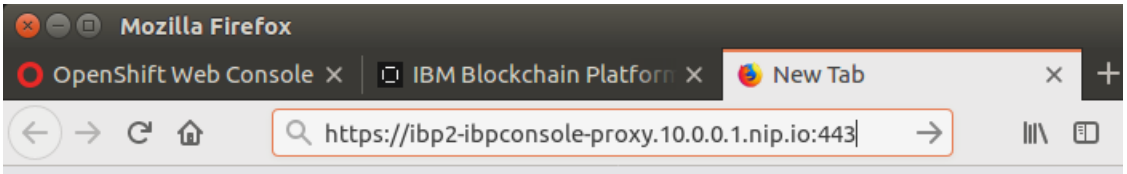
Before we can login though, there is one more step we must follow. There is a second self-signed certificate that is used by the "**grpc-web-proxy**" component of IBM Blockchain Platform that we must accept as well.

__ **26.** In Firefox, open a third tab and copy and paste in the URL below. Make sure you change the **<YOUR_IP_ADDRESS>** part to the IP address you placed into the **ibp-console.yaml** file earlier in this lab:

```
https://ibp2-ibpconsole-proxy.<YOUR_IP_ADDRESS>.nip.io:443
```

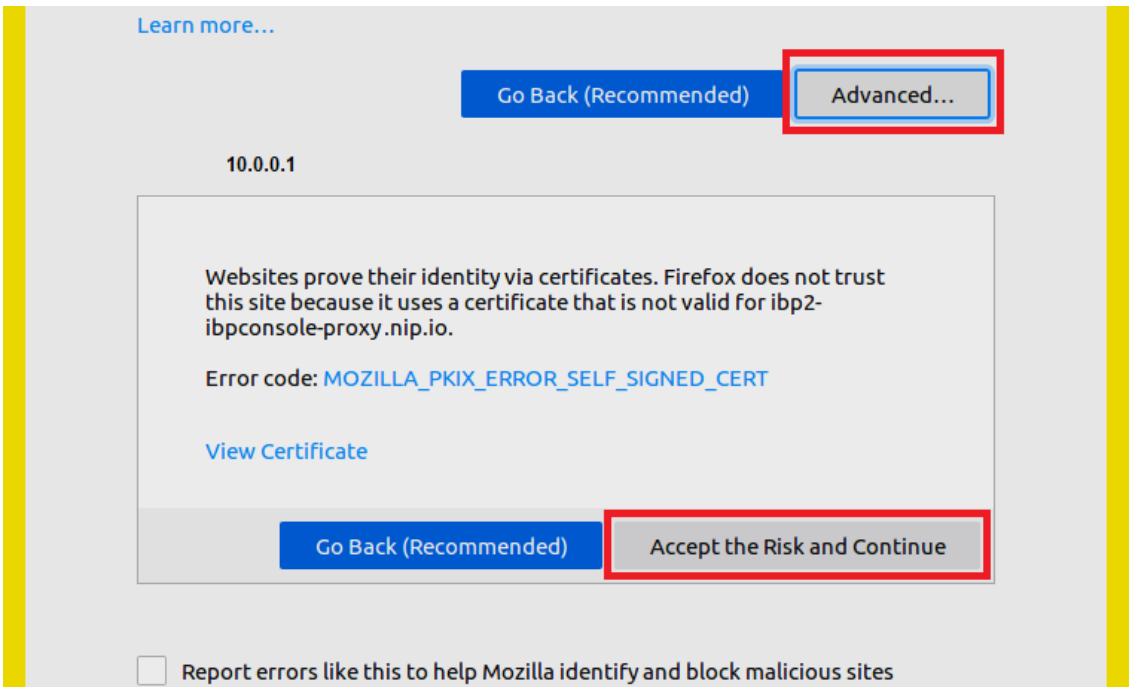
For example, if the IP address you used earlier was **10.0.0.1** then your URL would be:

```
https://ibp2-ibpconsole-proxy.10.0.0.1.nip.io:443
```

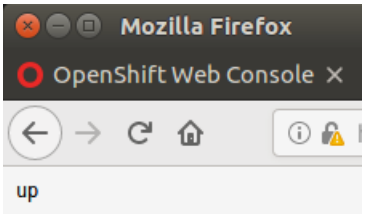


Note: The page may take a little time to load – refresh it if it seems to take too long.

__ **27.** Again, you should now see another security warning in Firefox. As before, click on "**Advanced...**" and then "**Accept the Risk and Continue**":

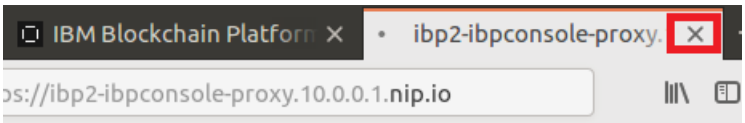


__ **28.** You should now see the single word "**up**" appear in your browser when the page is loaded:

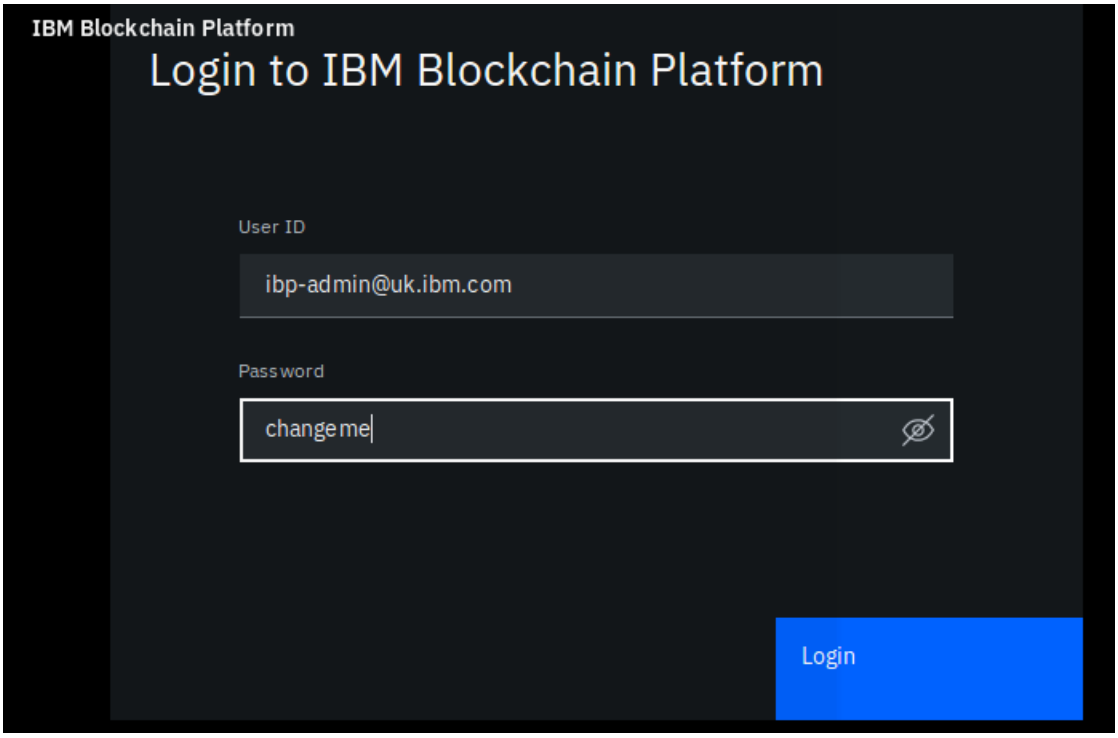


This shows the certificate has been accepted and the proxy is up and running.

__ **29.** Close the current proxy tab by clicking on the "**X**" to go back to the "**Login**" page on the previous tab we already have open:

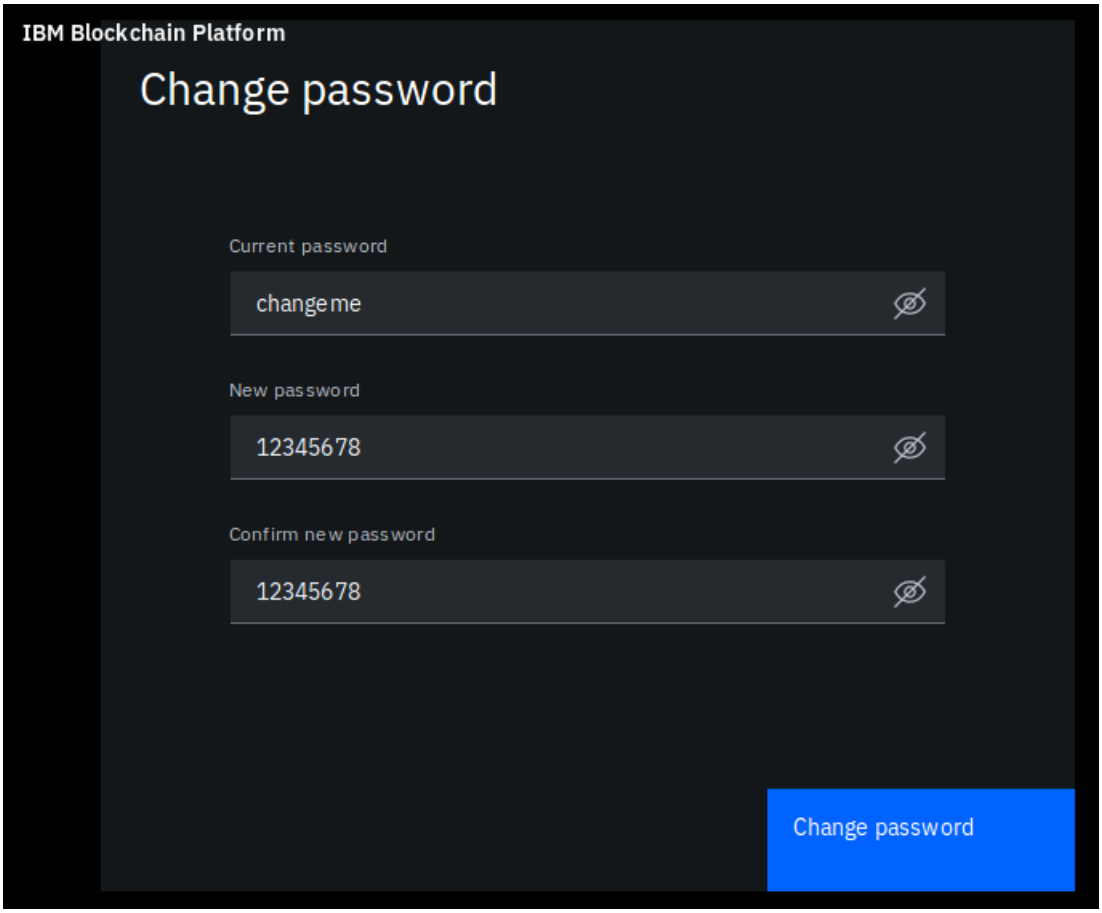


__ **30.** On the Login page, enter the User ID **ibp-admin@uk.ibm.com** and the Password **changeme** then click the "**Login**" button:



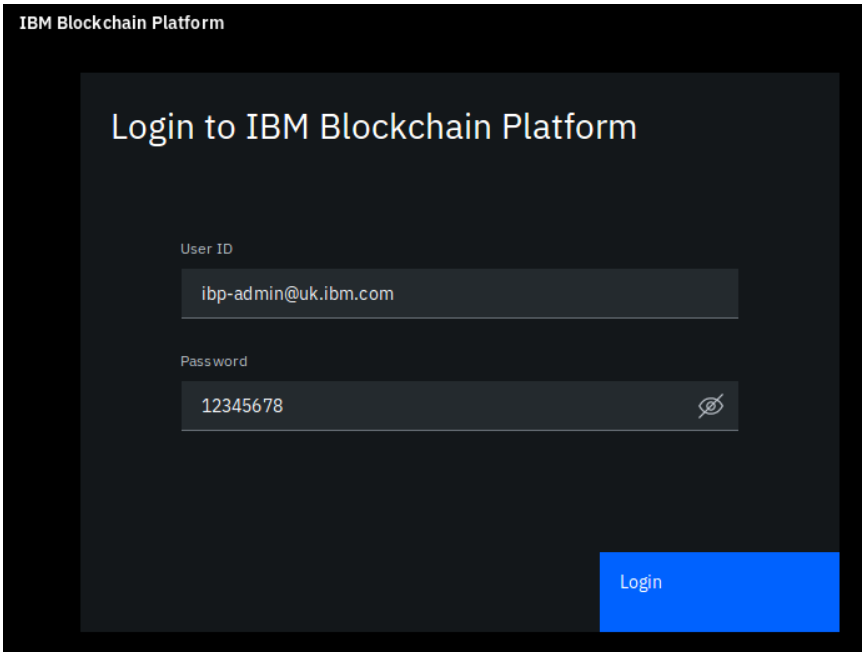
__ **31.** You will then be asked to "**Change password**" because it is the first usage of the console. In the "**Current password**" field enter "**changeme**" again. In the "**New password**" and the "**Confirm new password**" fields enter "**12345678**" as a simple password for the lab.

Make sure you click on the "**eye**" symbol after entering the password to make sure it has been entered correctly as shown below, then click the "**Change password**" button:

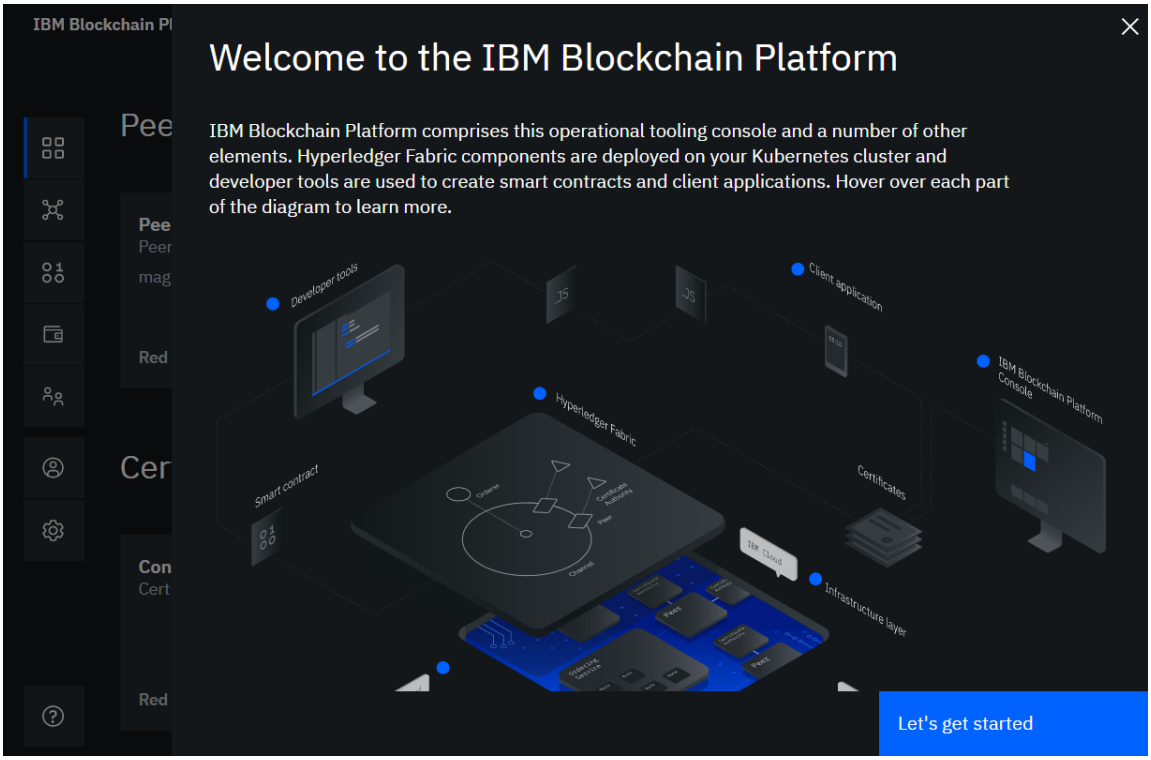


Note: Make sure you remember the password as you will need it again in the next step.

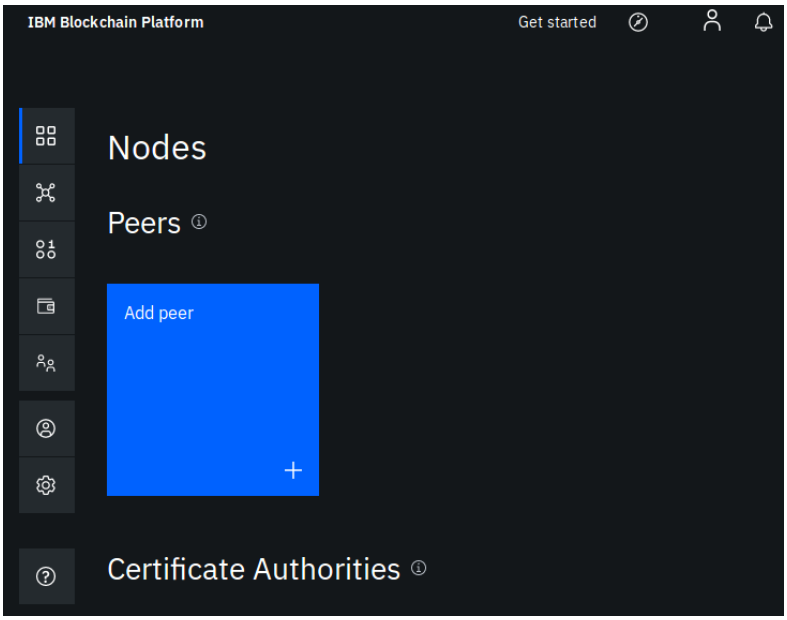
__ **32.** You will now be presented with the Login to IBM Blockchain Platform" page again, but this time you must login with the newly changed password details. In the "User ID" field enter **ibp-admin@uk.ibm.com** and in the "**Password**" field enter **12345678** as you did in the previous step:



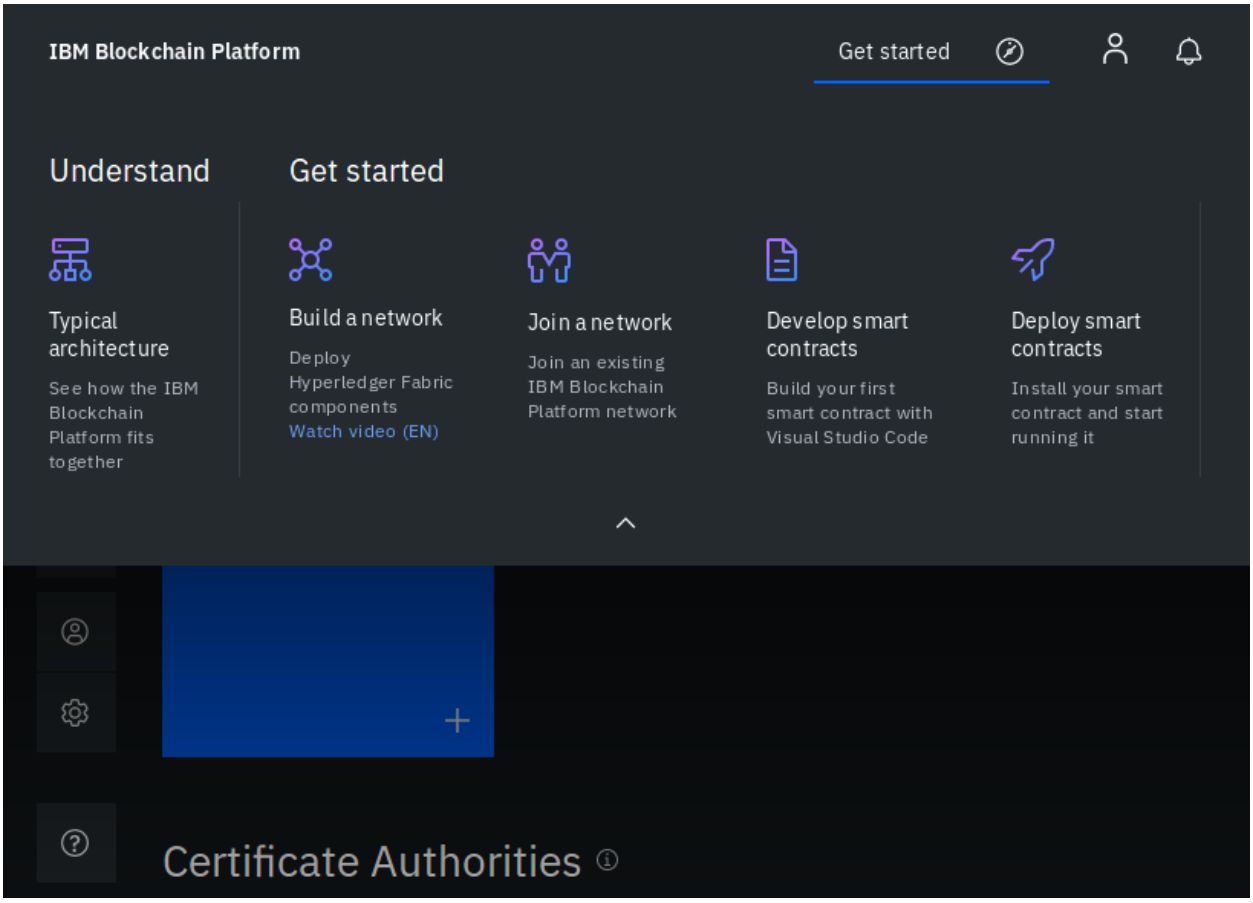
__ **33.** You will then see the **Welcome** page for the platform. Take a little time to move your mouse around the interactive diagram to see the different parts in a simple platform deployment:



__ **34.** When you are done, click the "**Let's get started**" button to close the diagram and move to the main console page:



__ **35.** From the main console page click the "**Get Started**" button at the top of the screen to bring up a list of links to the built-in tutorials:



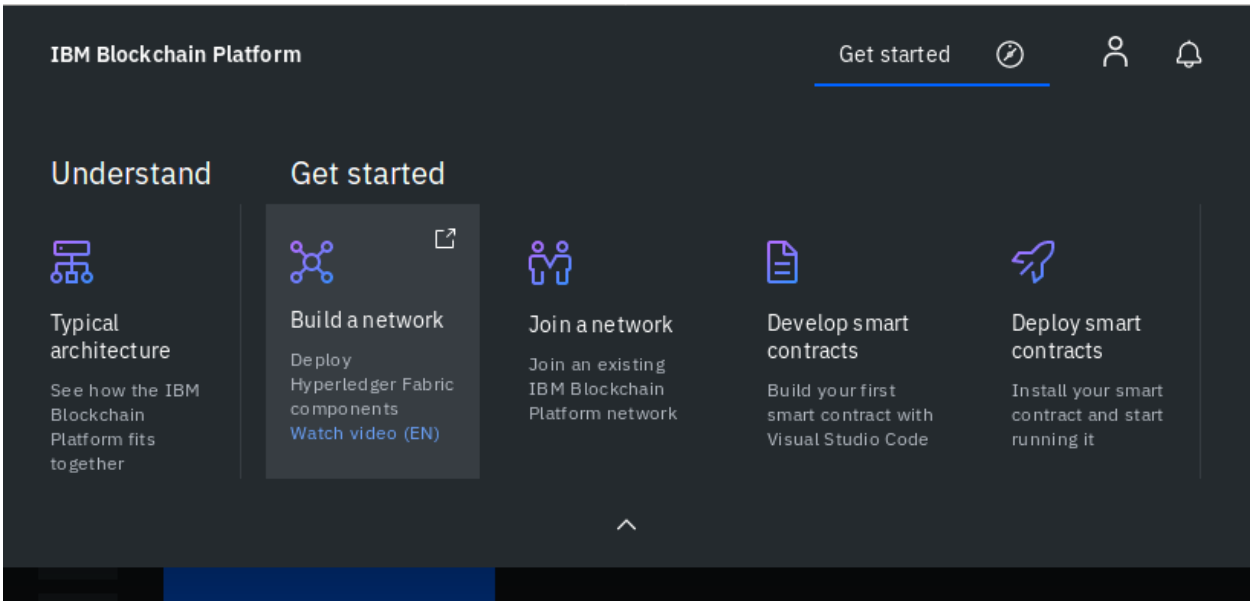
This concludes the setup part of this lab, where representatives from both organizations have deployed a new blockchain service into their respective OKD clusters.

In the rest of this lab, we will work to create a network that spans both of these services and joins them together in a blockchain network.

2.2 Building the network

This section is for Org1 ONLY – Org2 should watch and help along.

__ **36.** As **Org1**, click on the "**Build a network**" link:



The **Build** tutorial will open in a separate tab. For reference the URL to this page is:

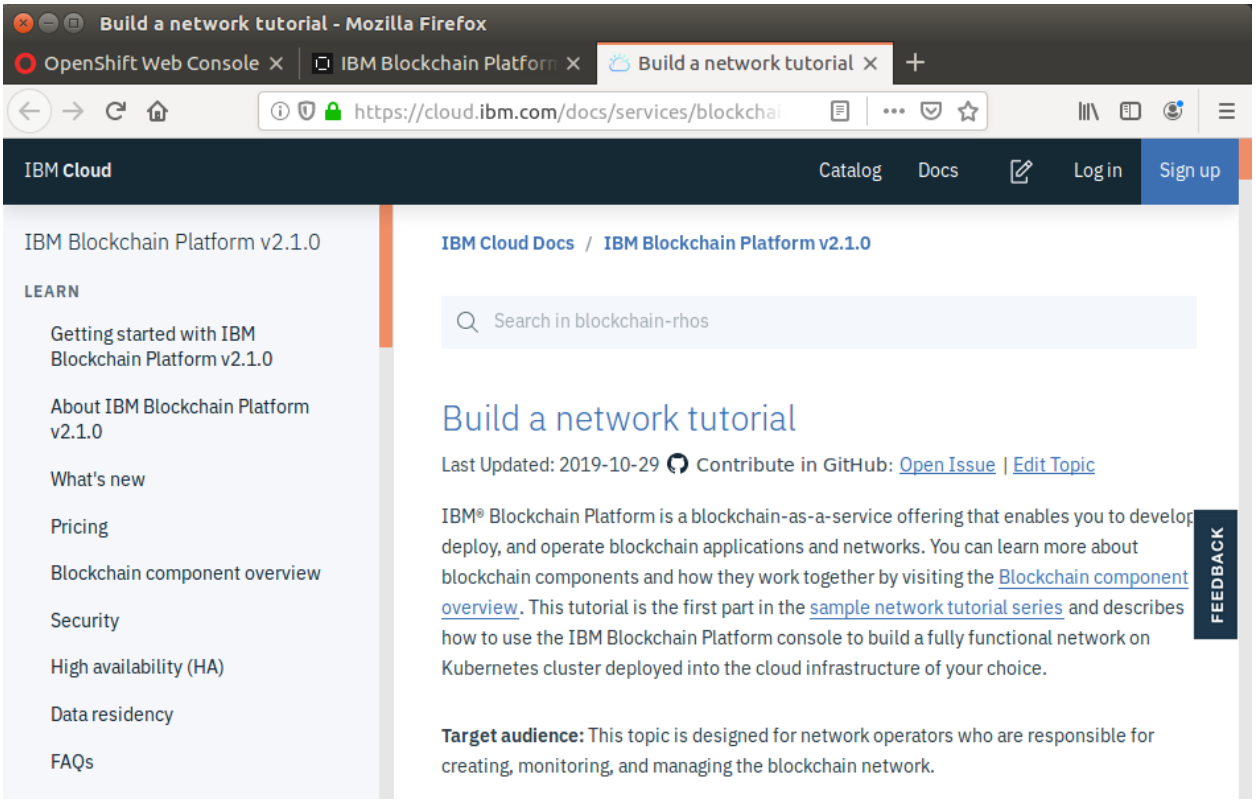
https://cloud.ibm.com/docs/services/blockchain-rhos?topic=blockchain-rhos-ibp-console-build-network

The **Build** tutorial takes you through five steps:

Step one: Create a peer organization and a peer
Step two: Create the ordering service
Step three: Join the consortium hosted by the ordering service
Step four: Create a channel
Step five: Join your peer to the channel

__ **37.** Working as **Org1**, work through the **Build** tutorial, following the numbered steps in circles. Please use the recommended names shown in the tutorial.

When you get to the **Next Steps** part, after **Step five**, stop following the tutorial and move on to the next section in this lab guide.

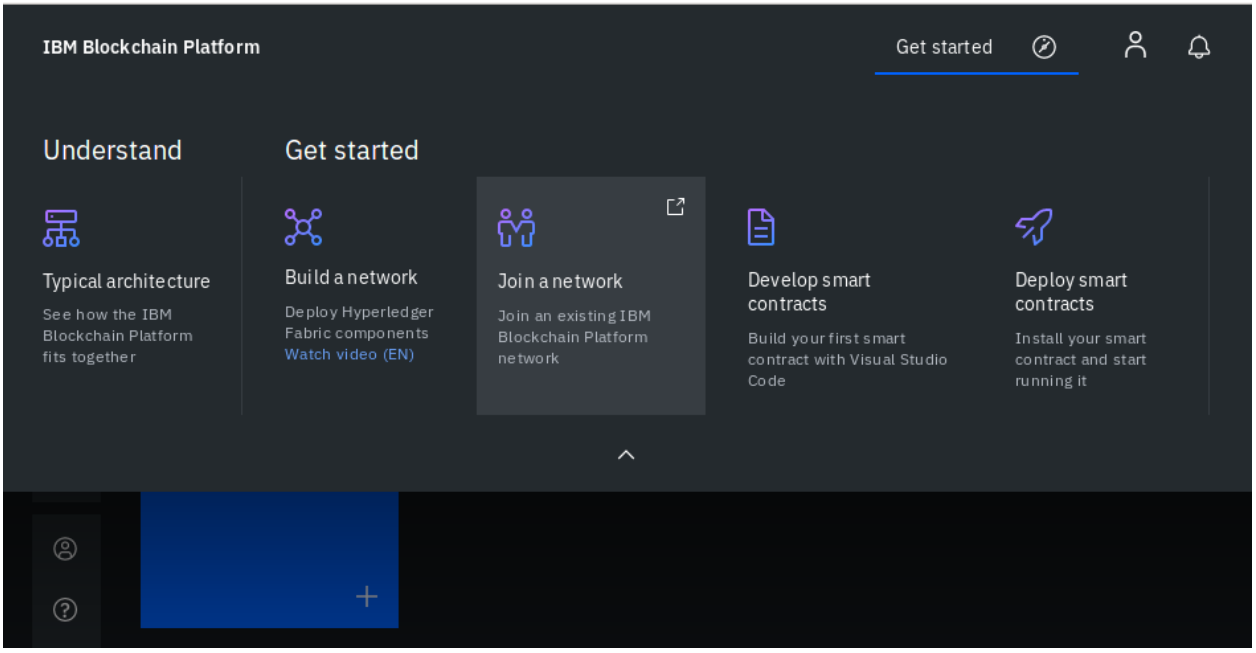


When working through this section, whilst you are waiting for components such as CAs, Peers an Ordering nodes to deploy and start you can check on their progress with the OKD Web Console using the Monitoring tab you used earlier in this lab. You can also use commands like "**`oc get pods`**" in the terminal window to check on progress from the command line as well.

2.3 Joining the Network

This section is for Org2 ONLY except where indicated – Org1 should watch and help.

__ **38.** As **Org2**, click on the "**Join a network**" link from the **Get started** link at the top of the page:



The **Join** tutorial will open in a separate tab. For reference the URL to this page is:

https://cloud.ibm.com/docs/services/blockchain-rhos?topic=blockchain-rhos-ibp-console-join-network

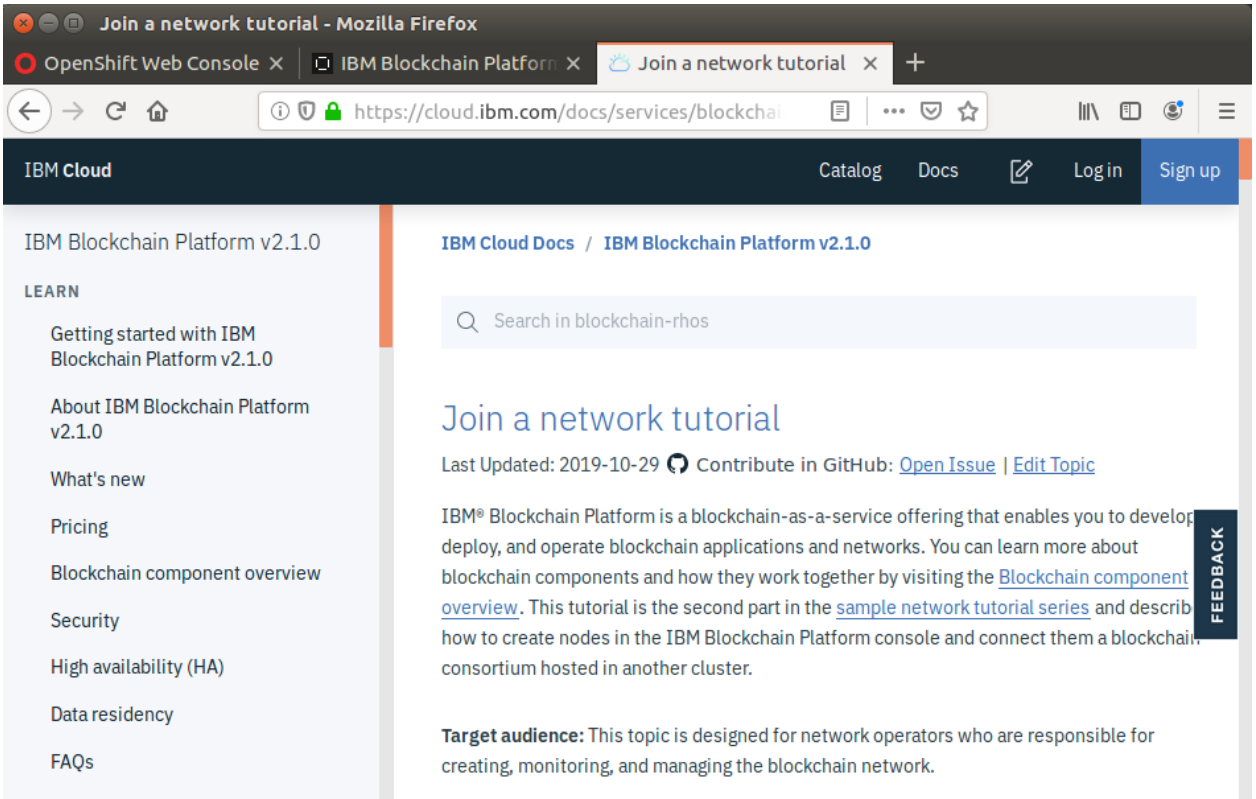
The **Join** tutorial takes you through four steps:

Step one: Create a peer organization and a peer
Step two: Join the consortium hosted by the ordering service
Step three: Add the peer's organization to an existing channel
Step Four: Join your peer to the channel

During a couple of the steps, you will need to exchange JSON information with **Org1** "**out-of-band**" and vice versa. To do this you will need to copy the information into an email or use a similar mechanism to transfer this information to your partner in the other org.

__ **39.** Working as **Org2**, work through the **Join** tutorial, following the numbered steps in circles. However, there are some steps that need to be done by **Org1** so look out for these. Please use the recommended names shown in the tutorial.

When you get to the **Next Steps** part, after **Step four**, stop following the tutorial and move on to the next section in this lab guide. However, if you are running short of time you can stop and move to the next section when you get to the start of the optional "**Creating a Channel**" section.



When working through this section, whilst you are waiting for components such as CAs and Peers to deploy and start you can check on their progress with the OKD Web Console using the Monitoring tab you used earlier in this lab. You can also use commands like "`oc get pods`" in the terminal window to check on progress from the command line as well.

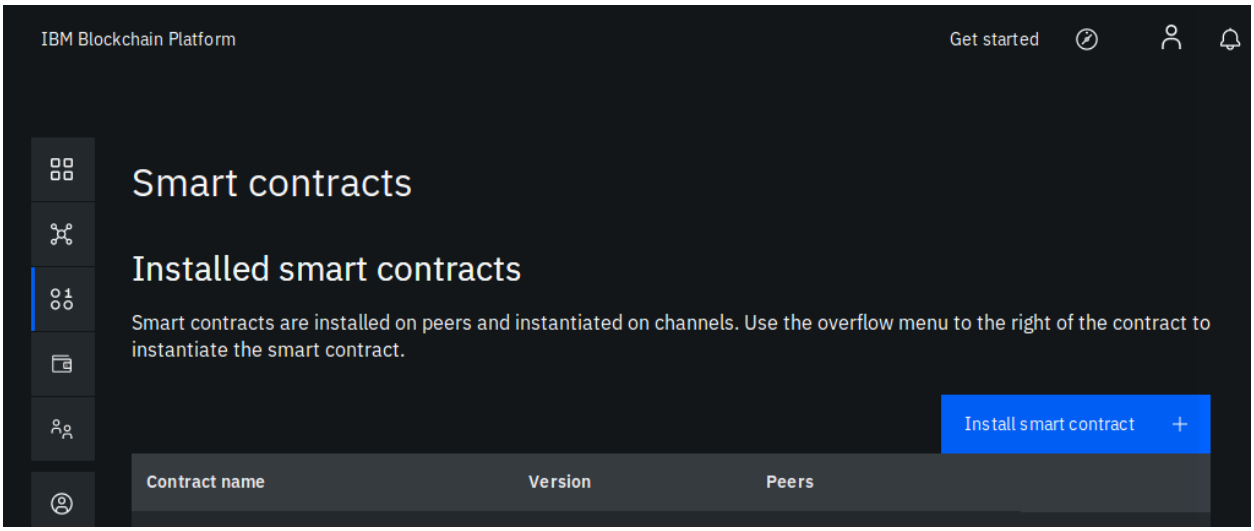
2.4 Deploying into the network

This section must be completed by both organizations as indicated in each step

To test the network out we need to deploy a simple smart contract and issue some transactions against it. To do this we will use the **fabcar** smart contract again.

Because an identical contract needs to be deployed into both organizations, we are going to use an existing packaged version of the contract to save time.

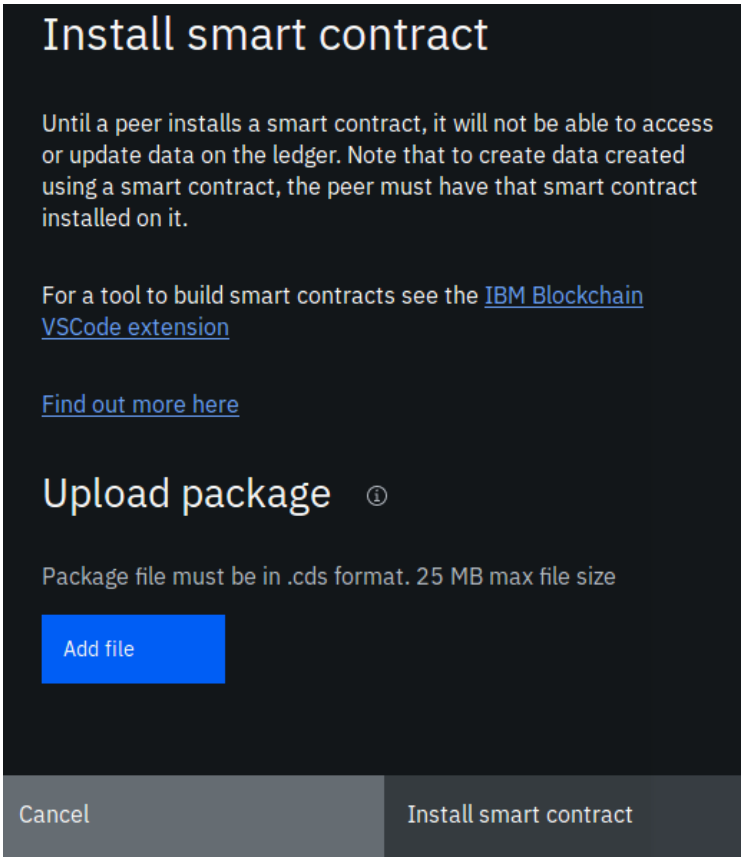
__ **40.** [**Org1 & Org2**] In the console UI, go to the **smart contracts** tab and click on the **Install smart contracts** button:



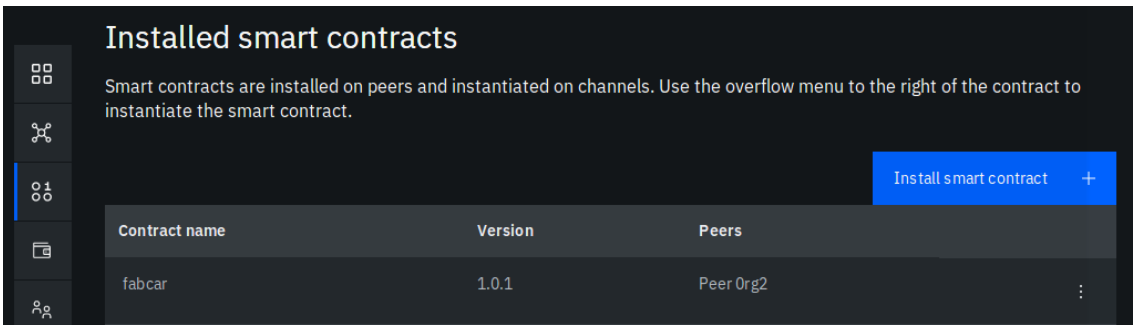
__ **41.** [**Org1 & Org2**] From the side bar, choose **add file** and navigate to the pre-prepared **fabcar@1.0.1.cds** contract in the following folder:

```
~/workspace/fabric-getting-started/fabric-samples/fabcar-complete/packages/
```

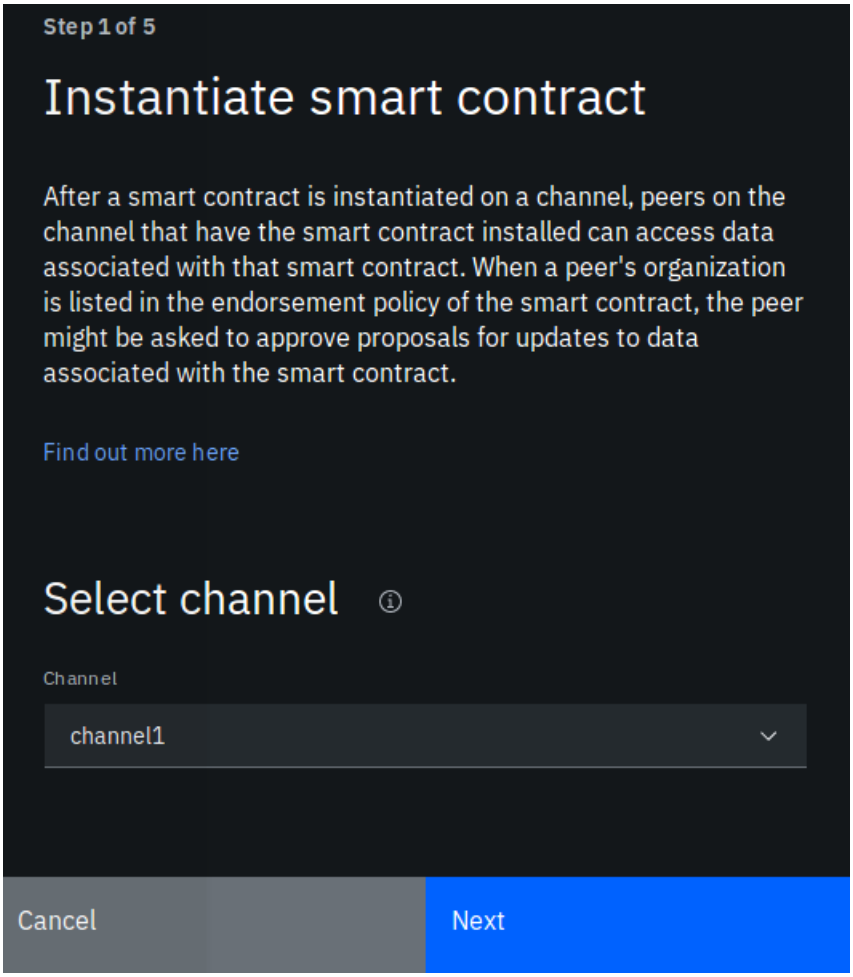
Once you have chosen the file, click "Open" and then click on the "**Install smart contract**" button:



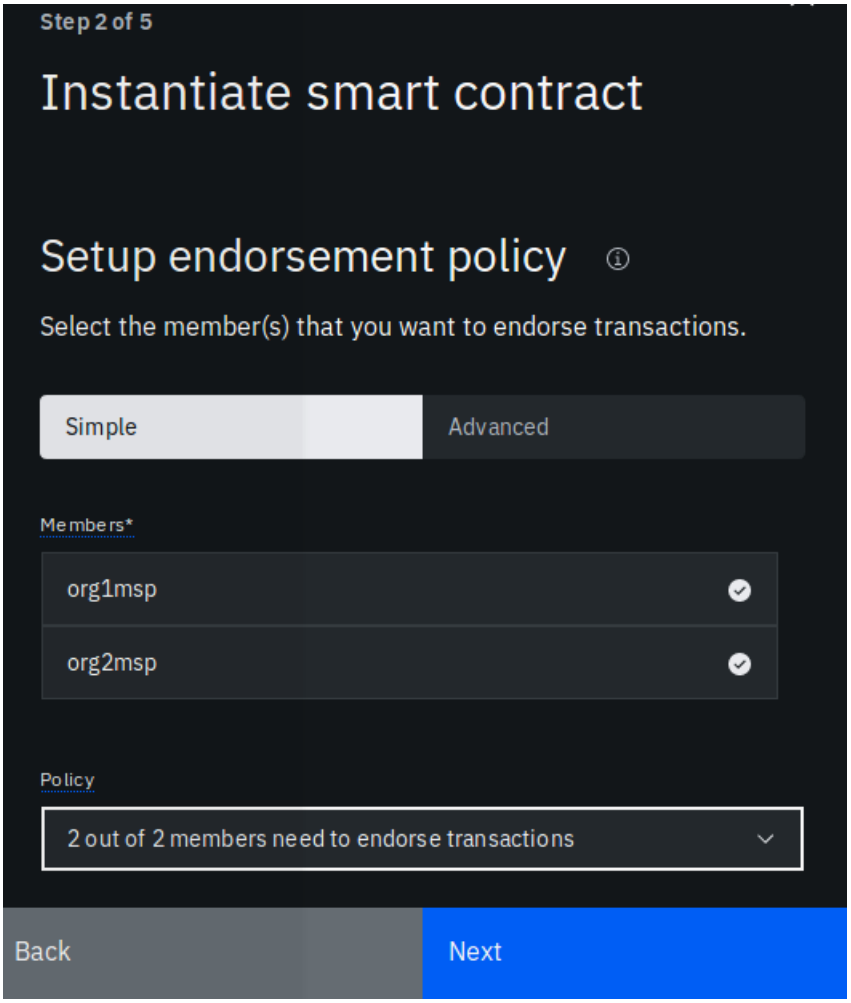
__ **42.** [**Org2 only**] On the same smart contracts page, find the **fabcar** contract we just installed and click on the vertical "**...**" button to the right and choose the "**Instantiate**" option:



__ **43.** [**Org2 only**] In the side panel, on **Step 1**, make sure you select the channel you joined earlier in the lab, **channel1**, and click **Next**:

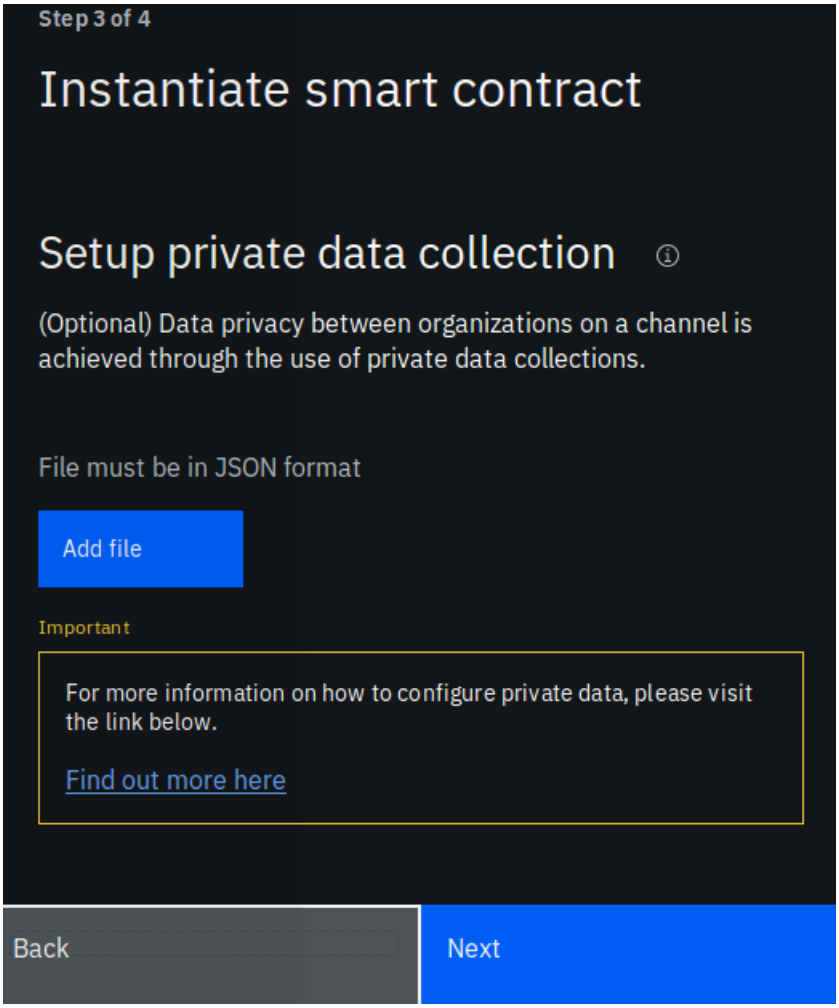


__ **44.** [**Org2 only**] In the side panel on **Step 2**, make sure both Members are selected (ticked) and change the policy to be "**2 out of 2 members need to endorse transactions**", the click **Next**:

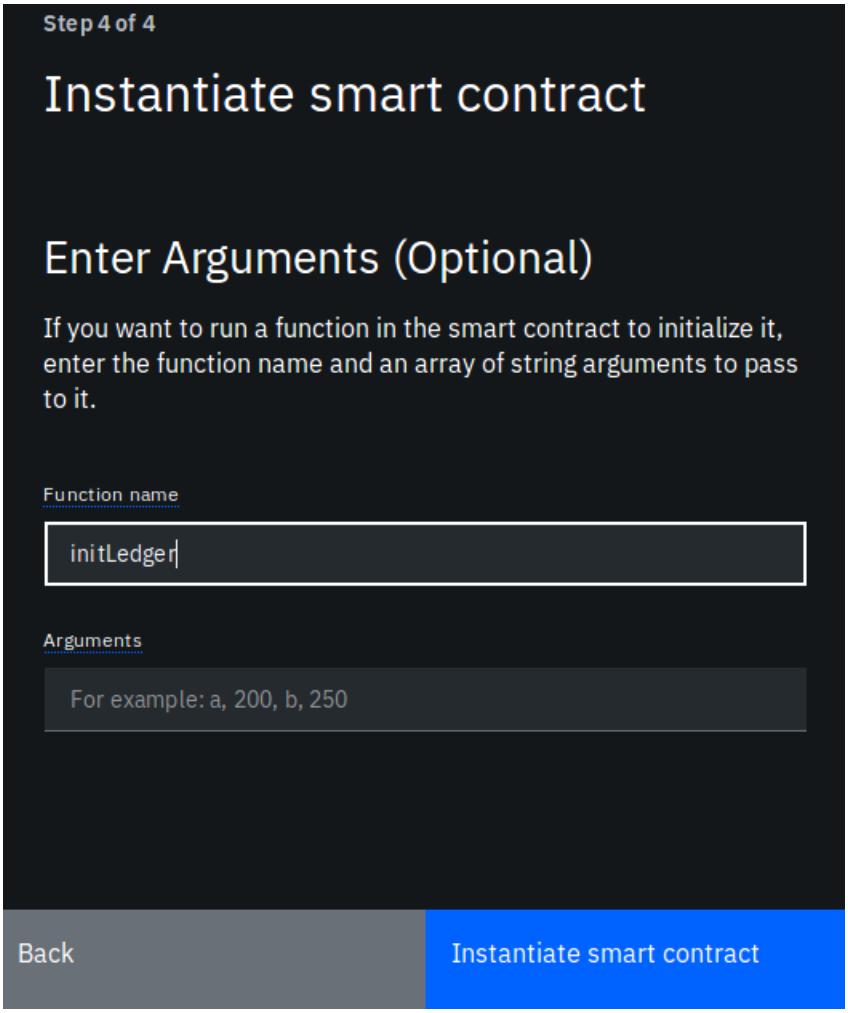


We set the endorsement policy to be "**2 of 2**" so we can ensure that both peers are required to execute and sign update transactions. However, this does mean we will need to configure the peer for **service discovery,** later in this lab.

__ **45.** [**Org2 only**] In the side panel on **Step 3**, as we are not using private data in this lab, you do not need to make any changes and can just click "**Next**":

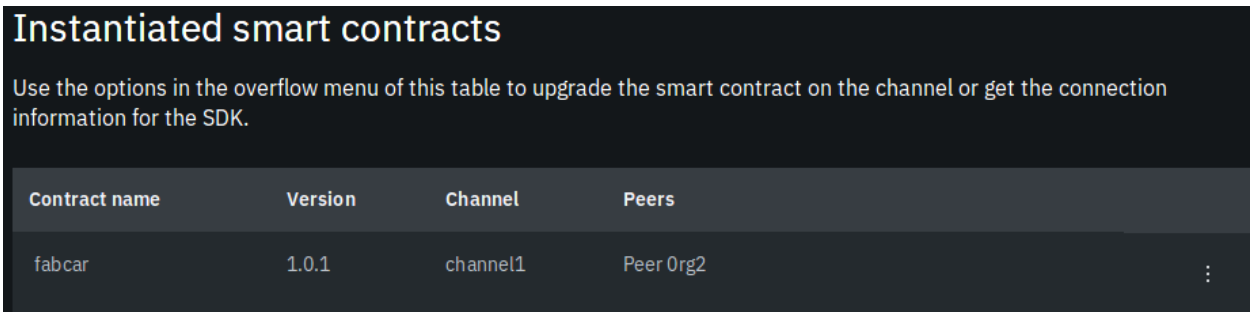


__ **46.** [**Org2 only**] In the side panel on **Step 4**, enter "**initLedger**" as the **Function name** to call which populates the ledger with a selection of sample cars. Leave the **Arguments** field empty and click "**Instantiate smart contract**":



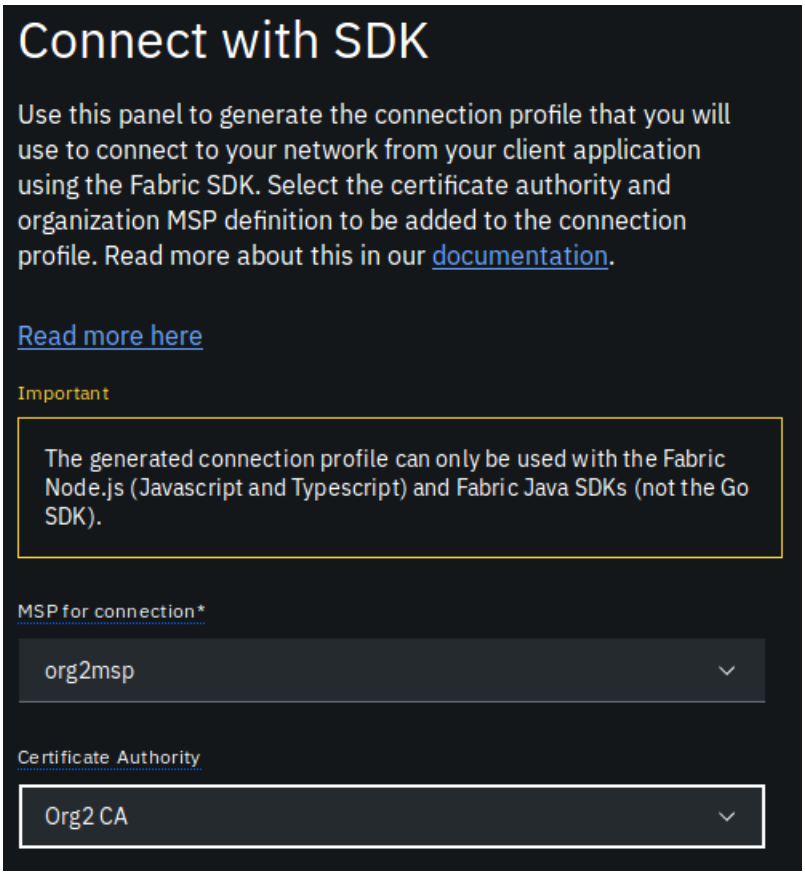
This step can take a few minutes to complete, please be patient.

__ **47.** [**Org1 & Org2**] On the same smart contracts page, scroll down to the "**Instantiated smart contracts**" section find the **fabcar** contract we just instantiated and click on the vertical "**...**" button to the right and choose the "**Connect with SDK**" option:

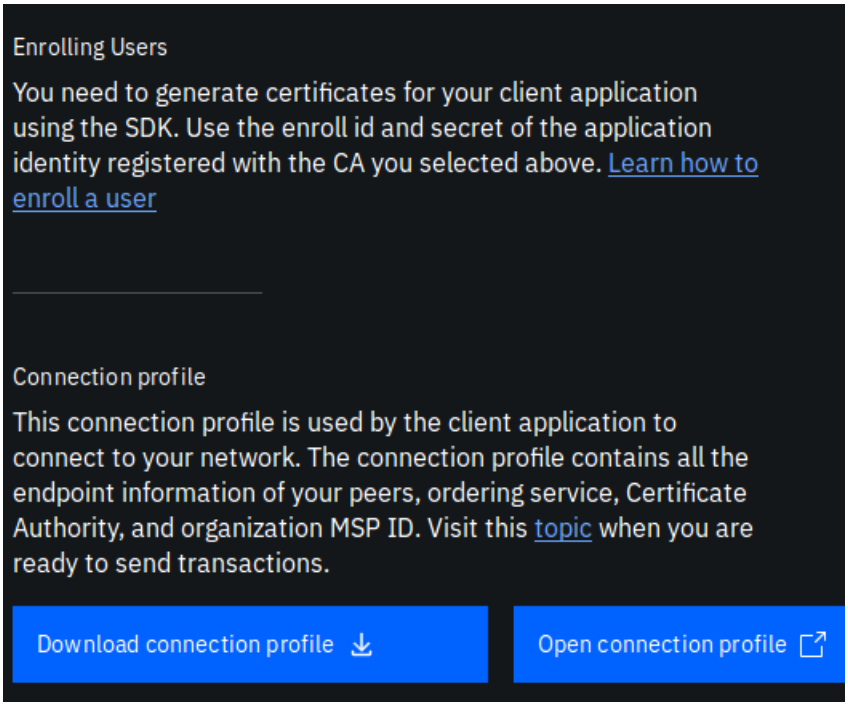


Note: **Org1** may need to refresh their page to see the instantiated contract.

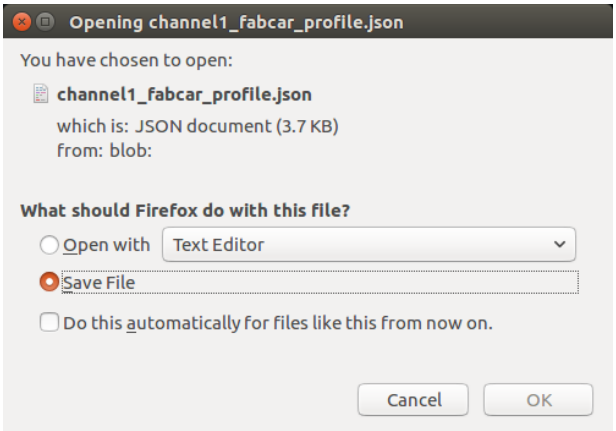
__ **48.** [**Org1 & Org2**] In the side panel select your MSP for connection and Certificate Authority. For **Org1** this will be "**org1msp**" and "**Org1 CA**", whilst for **Org2** this will be "**org2msp**" and "**Org2 CA**":



__ **49.** [**Org1 & Org2**] In the side panel scroll down and choose the "**Download connection profile**" button:



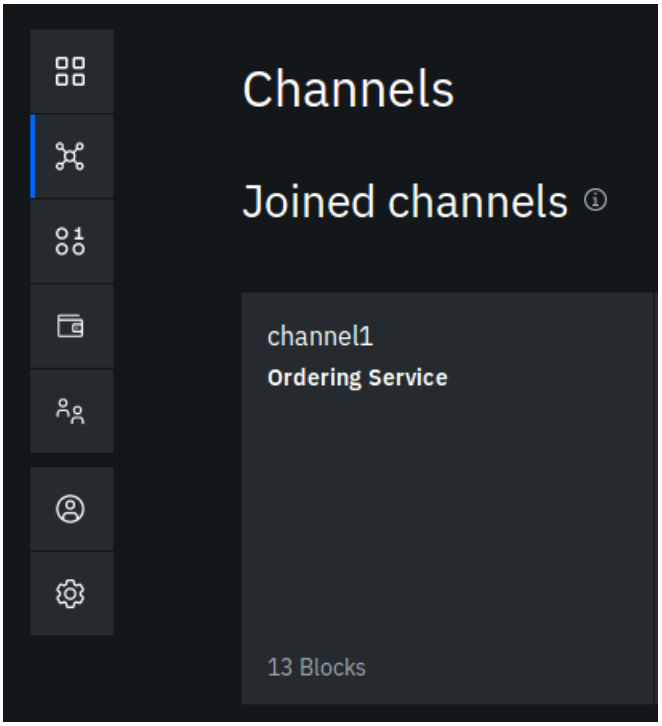
__ **50.** [**Org1 & Org2**] In the dialog, choose the "**Save File**" option and click "**OK**":



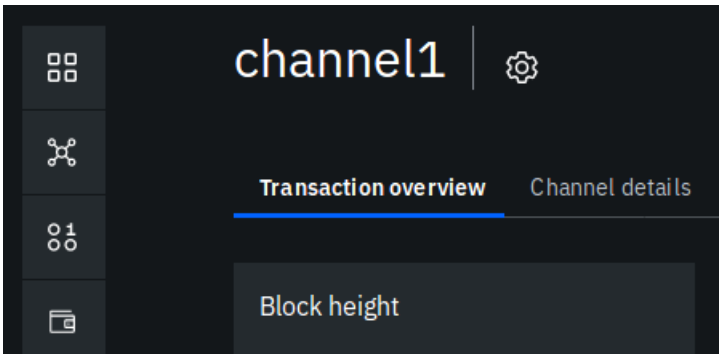
This will place the file "**channel1_fabcar_profile,json**" into the your **Downloads** folder. The path to this file is "**~/Downloads/channel1_fabcar_profile.json**".

__ **51.** [**Org1 & Org2**] Once you have downloaded the connection profile, click the "**Close**" button.

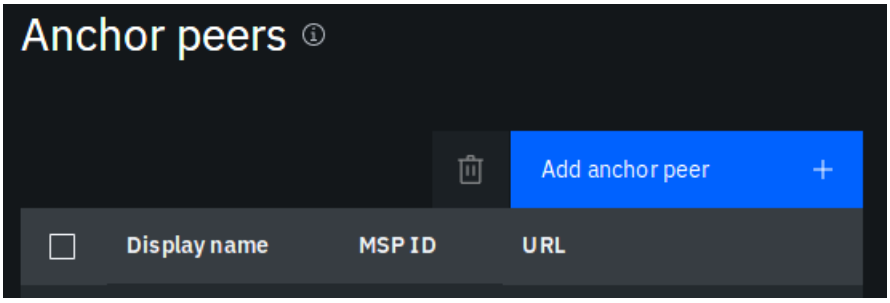
__ **52.** [**Org2 Only**] From the Channels icon, select **channel1**:



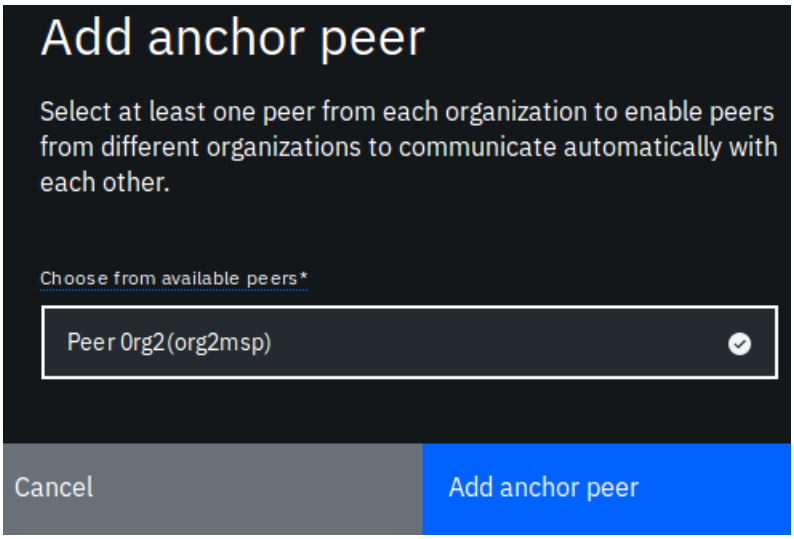
__ **53.** [**Org2 Only**] From the channels pane, choose the "**Channel details**" tab:



__ **54.** [**Org2 Only**] Scroll down the **channel1** details page to the **Anchor peers** section and click the "**Add anchor peer**" button:



__ **55.** [**Org2 Only**] From the side panel, choose your peer "**Peer Org2(org2msp)**" and click "**Add anchor peer**":

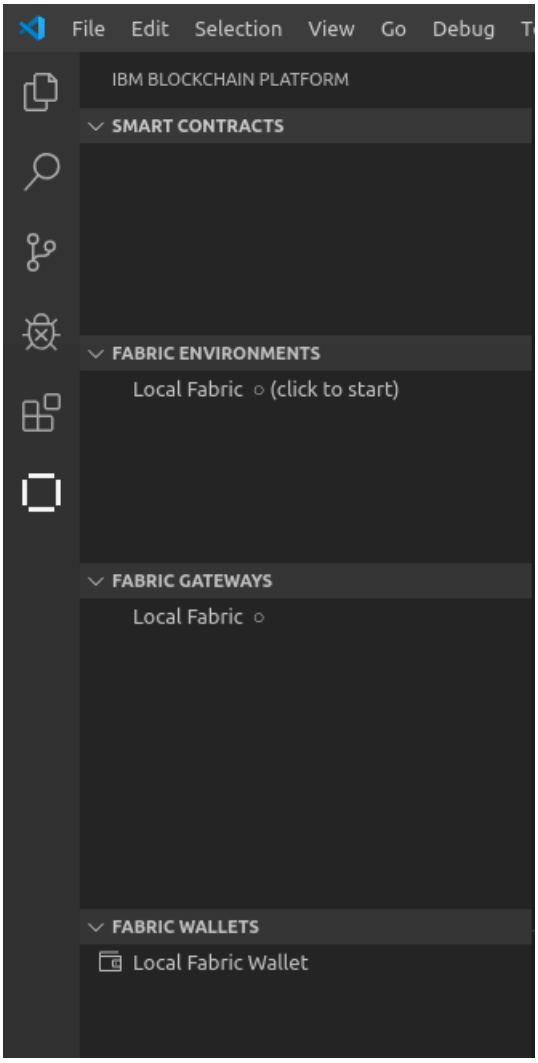


This will cause a configuration block to be added to the channel defining **Peer Org2** as an anchor peer for the channel which will allow Org1's **Peer Org1** to be able to discover Org2's peer and vice versa.

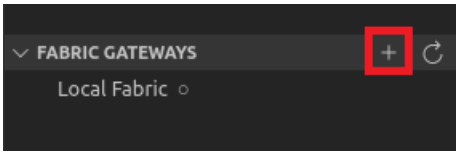
2.5 Connecting to the Network

This section must be completed by both organizations as indicated in each step

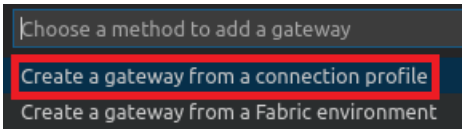
__ **56.** [**Org1 & Org2**] Open a new empty VS Code window and click on the **IBM Blockchain Platform** icon:



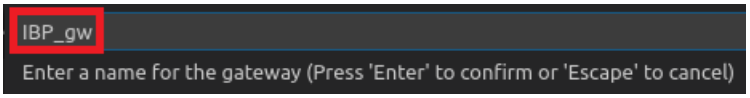
__ **57.** [**Org1 & Org2**] Move your mouse over the "**Fabric Gateways**" pane to make the "**+**" appear and click the "**+**" to start creating a new gateway:



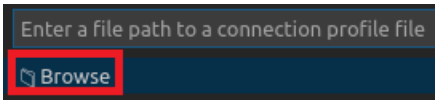
__ **58.** [**Org1 & Org2**] In the pop up, choose the "**Create a gateway from a connection profile**" option:



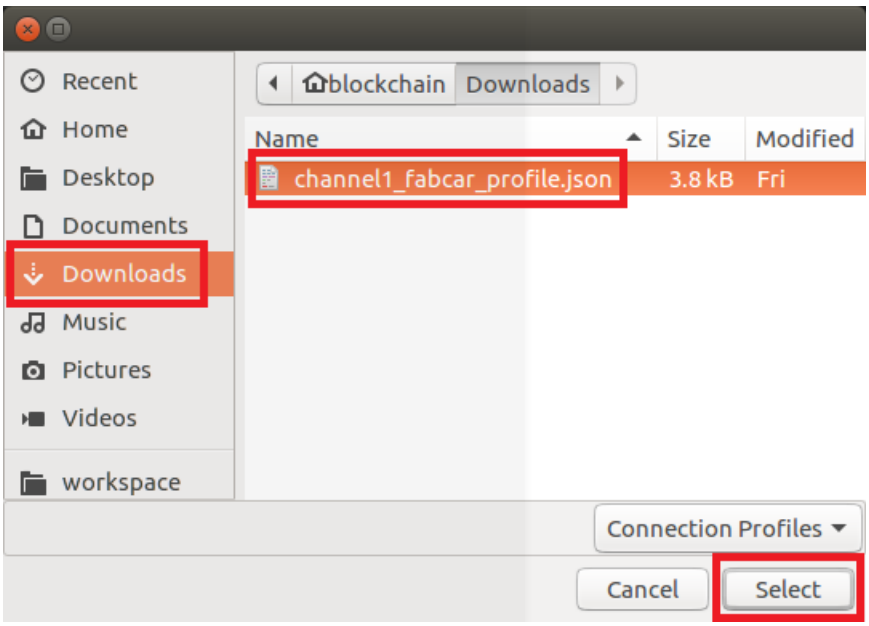
__ **59.** [**Org1 & Org2**] In the pop up enter "**IBP_gw**" as the name:



__ **60.** [**Org1 & Org2**] In the pop up click "**Browse**":

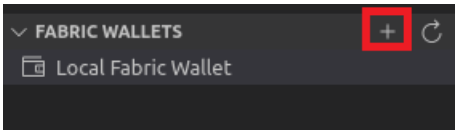


__ **61.** [**Org1 & Org2**] In the dialog click "**Downloads**" on the left, choose the "**channel1_fabcar_profile.json**" file then click "**Select**":

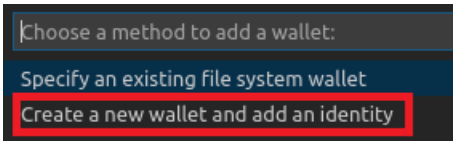


The Fabric Gateways will update with the new gateway, but before we can use it to connect we first need to create a new wallet with a new user id in it.

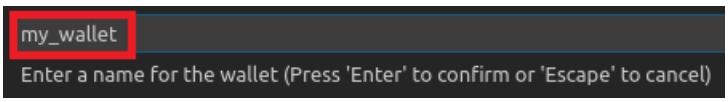
__ **62.** [**Org1 & Org2**] Move your mouse over the "**Fabric Wallets**" pane to make the "**+**" appear and click the "**+**" to start creating a new wallet:



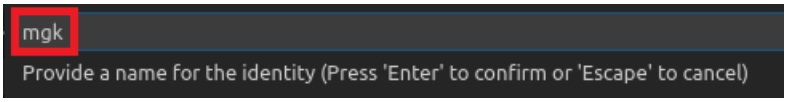
__ **63.** [**Org1 & Org2**] In the pop up choose "**Create a new wallet and add an identity**":



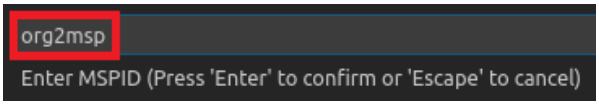
__ **64.** [**Org1 & Org2**] In the pop up enter the name "**my_wallet**" and press enter:



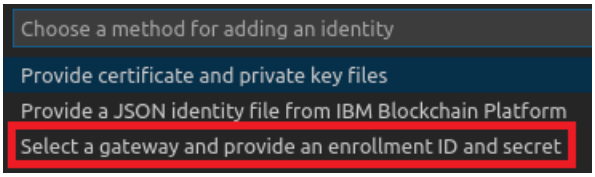
__ **65.** [**Org1 & Org2**] In the pop up enter your own name or initials (with no spaces or special characters) and press enter:



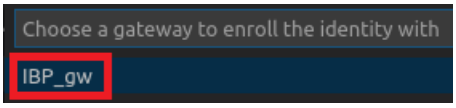
__ **66.** [**Org1 & Org2**] Enter your MSPID. For Org1 this is "**org1msp**" while for Org2 this is "**org2msp**", then press enter:



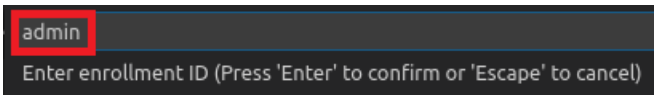
__ **67.** [**Org1 & Org2**] In the pop up choose "**Select a gateway and provide an enrollment ID and secret**":



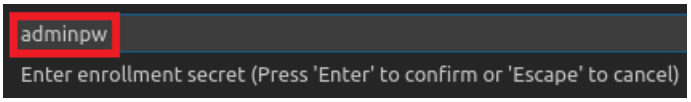
__ **68.** [**Org1 & Org2**] In the pop up choose the "**IBP_gw**" gateway we created earlier:



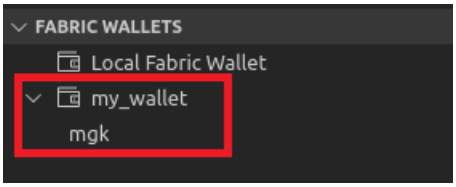
__ **69.** [**Org1 & Org2**] In the pop up enter the CA enrollment ID we created earlier "**admin**":



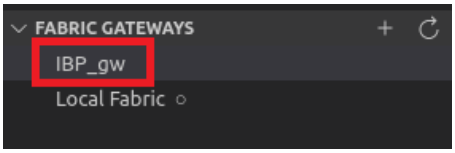
__ **70.** [**Org1 & Org2**] In the pop up enter the CA enrollment secret we created earlier "**adminpw**":



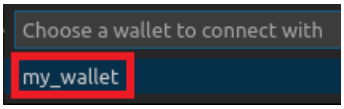
There will be a couple of Information Messages and then the wallets pane will update to show the new wallet with the new ID inside it:



__ **71.** [**Org1 & Org2**] From the "**Fabric Gateways**" view select the "**IBP_gw**" gateway we created earlier:

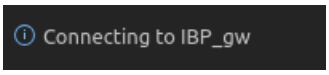


__ **72.** [**Org1 & Org2**] In the pop up choose the "**my_wallet**" wallet we just created:



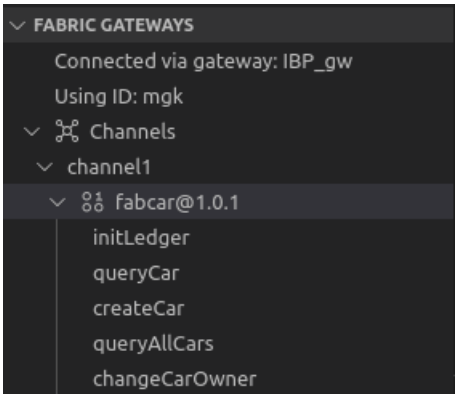
Note: It will not ask you to select the ID inside the wallet as we only have one ID. If we had more than one, there would be an extra step to choose the ID to use as well.

There will be an Information Message when the connection is complete:



Also the Fabric Gateway view will update to show the details of the network we are connected to.

__ **73.** [**Org1 & Org2**] If you expand the **channel1** channel and the **fabcar@1.0.1** smart contract you should be able to see the transactions available in the contract:

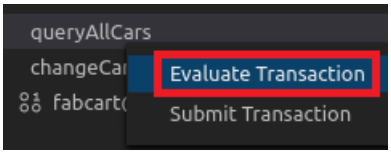


2.6 Issuing Transactions

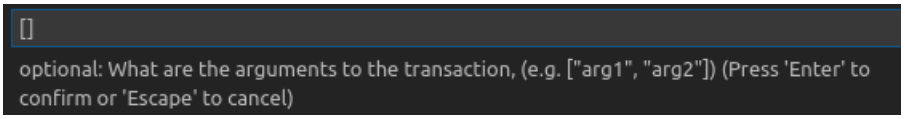
This section must be completed by both organizations as indicated in each step

We are now going to issue transactions to test that both networks are set up correctly.

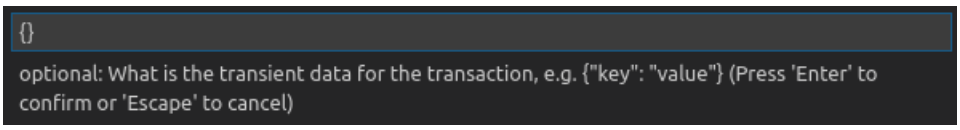
__ **74.** [**Org1 & Org2**] Right click on the "**queryAllCars**" transaction and choose "**Evaluate Transaction**":



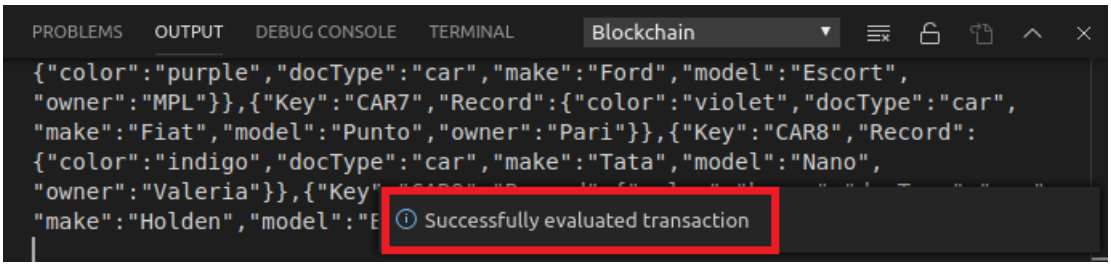
__ **75.** [**Org1 & Org2**] In the pop up just press enter as **queryAllCars** does not need any parameters:



__ **76.** [**Org1 & Org2**] In the next pop up just press enter as **queryAllCars** does not use any transient data:

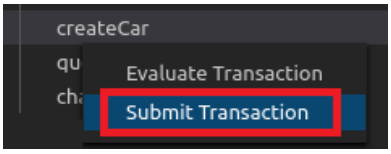


As well as an **Information Message**, you should see the output window update with the details of all the cars that the "initLedger" transaction created when we instantiated the contract:



Now we are going to create a new car. As there are now two peers in the network and both peers are required to endorse the transaction, it will automatically be sent to both peers by the VS Code extension.

__ **77.** [**Org1 & Org2**] Right click on the "**createCar**" transaction and choose "**Submit Transaction**":

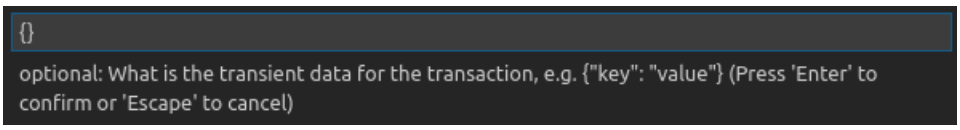


__ **78.** [**Org1 & Org2**] In the pop up enter the parameters for your new car, using a different ID for each org. For Org1 use **"CAR100"** and for Org2 use "**CAR200**" to differentiate between them. Enter some values of your choice like **"CAR200","Tesla","Model S","Red","MGK"** inside the square brackets and press "Enter". The order of the parameters for reference is:

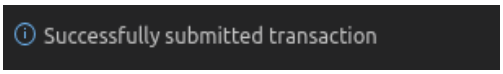
ID, Make, Model, Color, Owner

Note: Remember, you should not enter any quotes or extra spaces around this string as otherwise they may be taken as part of the string itself which will result in an error.

__ **79.** [**Org1 & Org2**] In the next pop up just press Enter as **createCar** does not use any transient data:



You should see a successful **Information Message**:



__ **80.** [**Org1 & Org2**] Use the **queryCar** transaction twice to query for your own car and your partner Organization's car. For example, each org should query for **CAR100** and **CAR200** and verify they can see the expected results.

__ **81.** [**Org1 & Org2**] **Congratulations**, you have now finished this Lab. Please use any remaining time to experiment with the environment you have created.

3 We Value Your Feedback!

- Please ask your instructor for an evaluation form. Your feedback is very important to us as we use it to continually improve the lab material.

- If no forms are available, or you want to give us extra information after the lab has finished, please send your comments and feedback to "**blockchain@uk.ibm.com**"