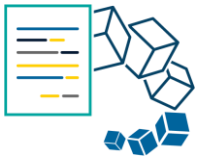


Lab: Explore the IBM Blockchain service on IBM Bluemix

Overview

In this lab, you use the IBM® Blockchain service on IBM® Bluemix to build on the car leasing demo that was introduced in the previous lab, “Transfer assets in a business network.”

If you completed the previous lab, you have already deployed the car leasing application to your account, which means you can skip Step 1 and reuse your existing application.

Tip: This lab shows some screen captures in the IBM Bluemix interface in the classic view. If you log in to Bluemix and want to work in the classic view, click the avatar in the upper right and select **Switch to Classic** at the bottom of the avatar window.

Important: Because the IBM Blockchain service is in beta, it might be temporarily unavailable or at capacity. If you experience problems in the lab when accessing the IBM Blockchain service dashboard, try accessing the dashboard later.

Prerequisites

It’s recommended that you use Firefox or Chrome web browsers.

You will need a Bluemix account to create the sample application.

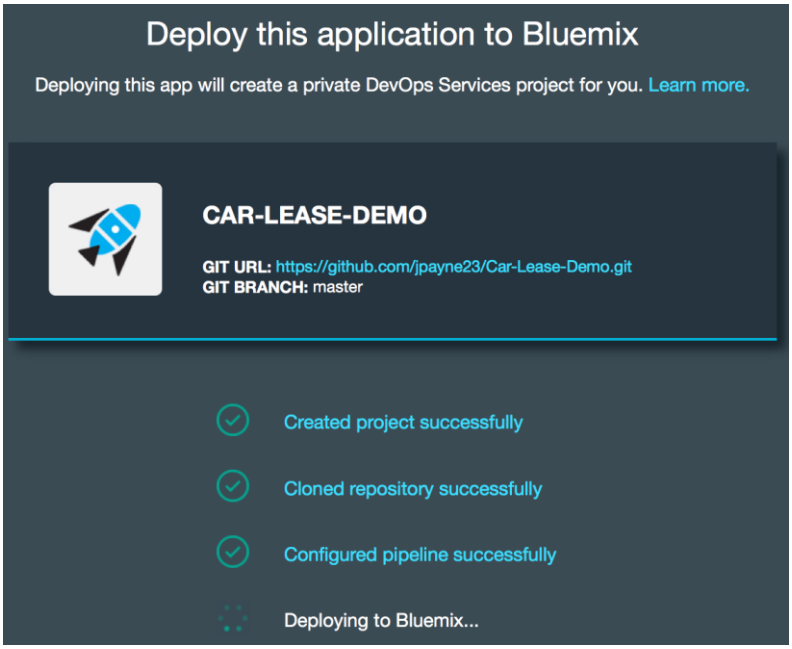
Step 1. Deploy and configure the sample application

If you completed the previous “Transfer assets in a business network” lab, you may skip ahead to Step 2 in this lab. To deploy the sample application:

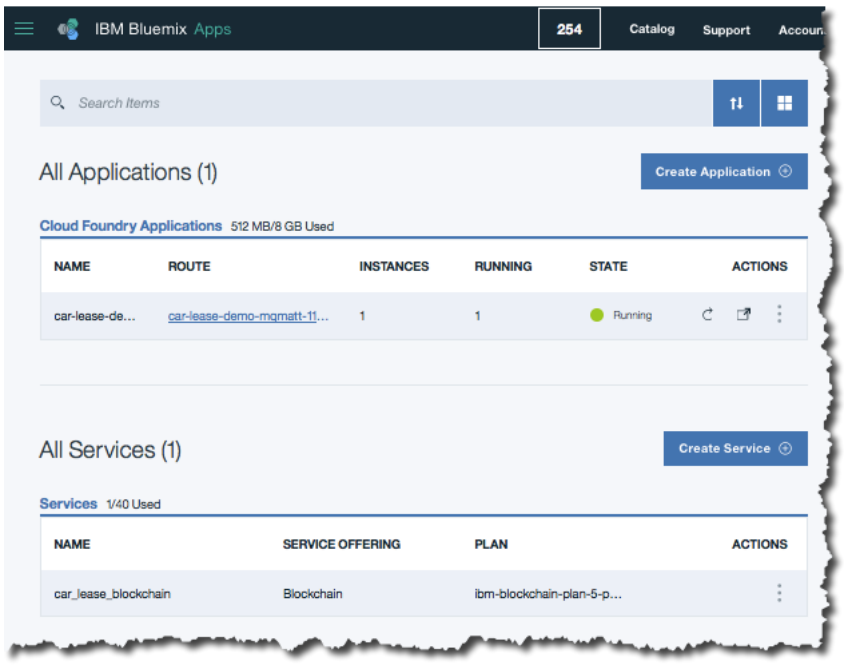
1. Open your browser and go to http://www.bluemix.net.
2. Click **Sign up** or **Log in** to create a new Bluemix account or log into your existing account.
3. After you have successfully signed up and logged into Bluemix, click **CATALOG** from the top bar.
4. Enter `Blockchain` in the search bar. When the service icon is displayed, click it to open the service information page.
5. Click **View Docs** to learn about the process of creating a blockchain environment.
6. Expand **Sample Apps and Tutorials** on the right side of the page to view the available apps.
7. Select the **Using Car Lease Demo** item from the list of apps
8. Click **Deploy to Bluemix** after the Car-Lease-Demo overview paragraph. You might need to log into Bluemix again.

   If this is your first time using Bluemix DevOps services, you will be prompted to create an alias for the DevOps Services Git repository that will to link to your IBM ID. This could be the first part of your email address (add a number afterward if needed to make it unique). Click **Create** after providing the alias.

   You can leave the `App Name, Region, Organization`, and `Space` attributes in the default state and click **Deploy**. It will take a few seconds for the default field values to be populated. This action will cause the Car-Lease-Demo to be deployed to your Bluemix environment and might take a couple of minutes to complete.



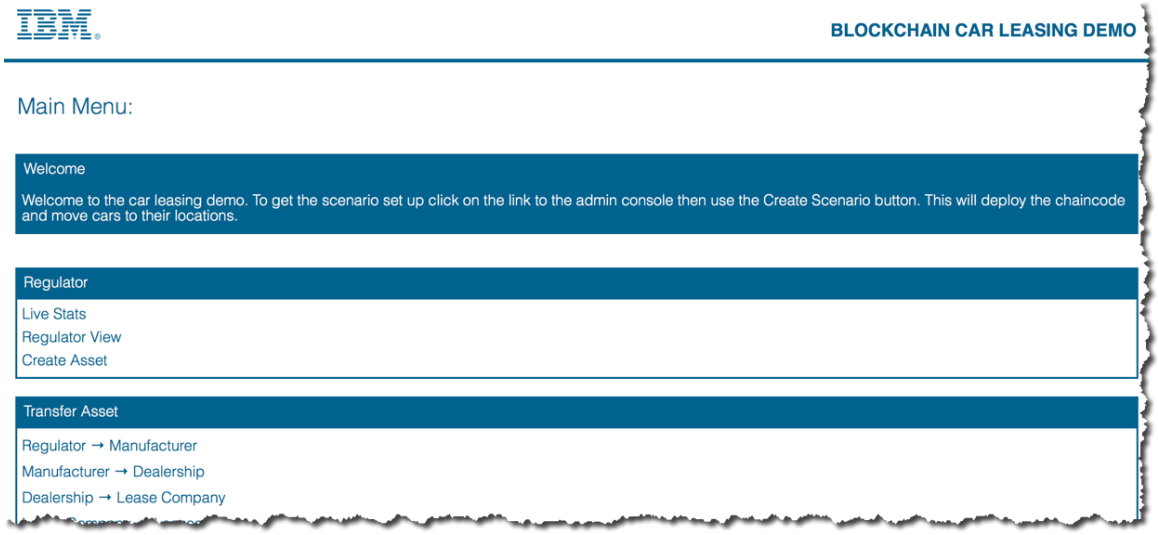
When you see the “Success!” message, click **Dashboard** to see the car leasing application that you created. If you don’t see the application listed, confirm that you are in the same organization and space used for the deploy to Bluemix.



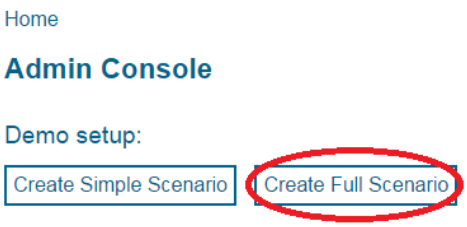
9. Click the application name in the row listing of the dashboard to show the application details page and open the Overview panel.

10. To access the application, click **View App.**

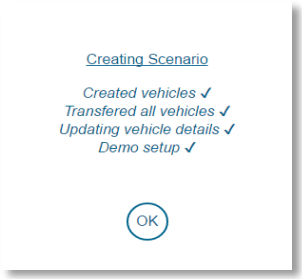
This will load the demo application home page:



11. Scroll down and click **Admin Console > Create Full Scenario** to load the initial set of assets into the blockchain. This will take several minutes to complete.



The scenario setup is complete when “Demo setup” is displayed.



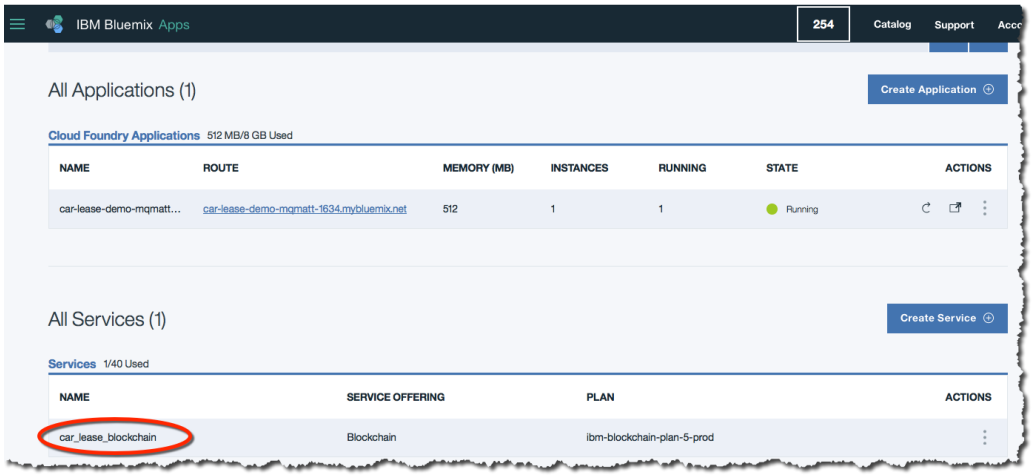
If an error occurs when creating the scenario, read “Remove the sample application” at the end of this document for instructions about how to delete the service.

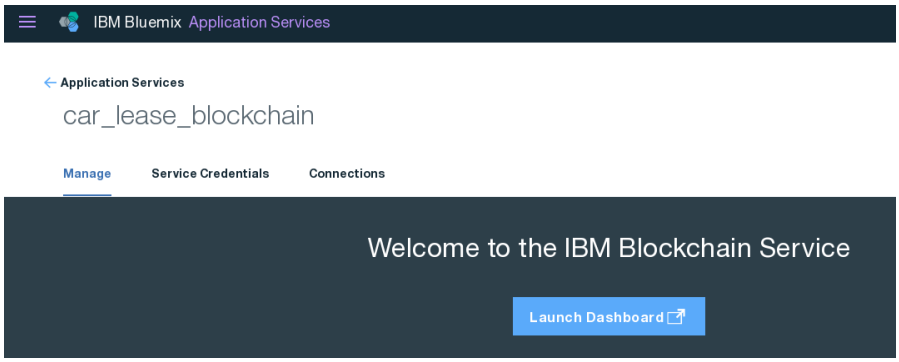
Step 2. Manage the sample application

In this section, use the tools available in the IBM Bluemix environment to view and manage the blockchain.

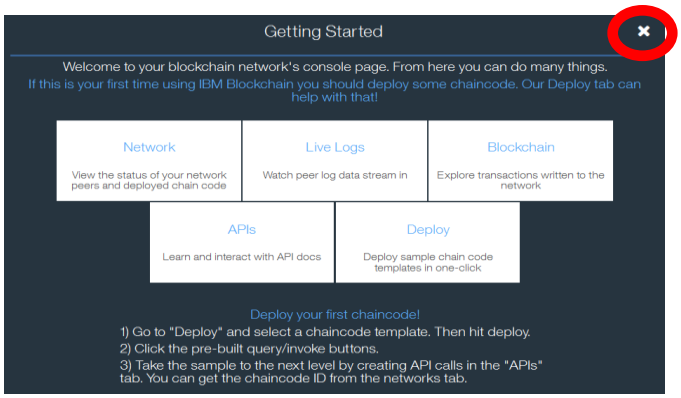
2.1 View the components of the IBM Blockchain service

1. In Bluemix, click **Dashboard** to return to the listing of Applications and Services.
2. Click the name of the Blockchain service for the car leasing application to open the service welcome page:

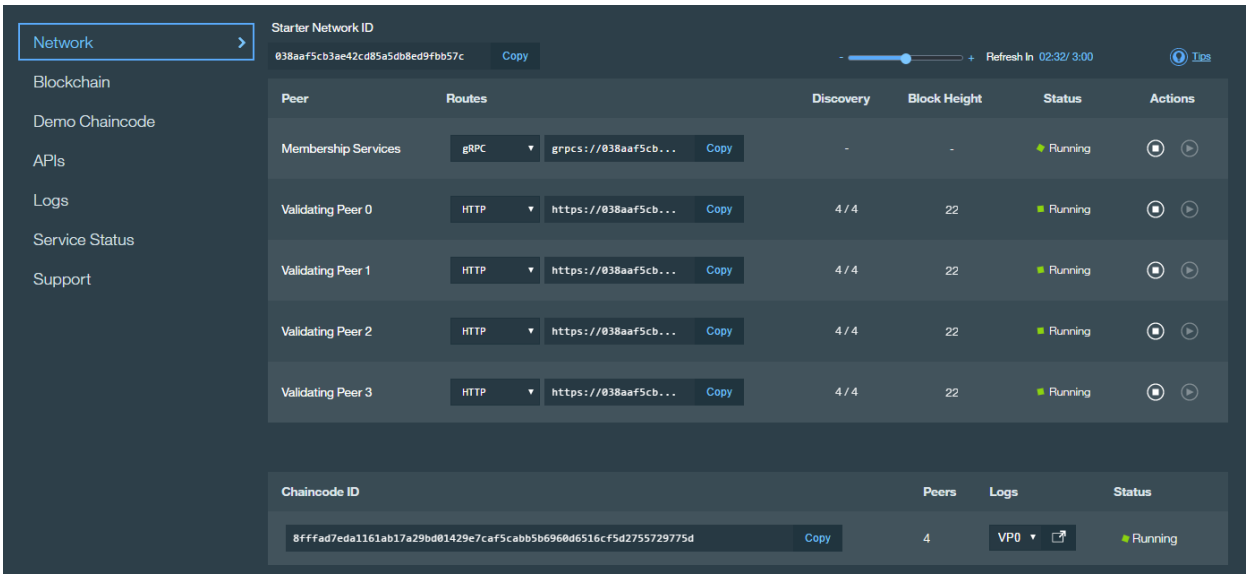




3. Review the details and click **Launch Dashboard** to start the service console.
4. Close the window that shows information about the sections. You will be able to look at these in more detail throughout this lab.



After closing the window, you will see the dashboard page with the Network item selected.

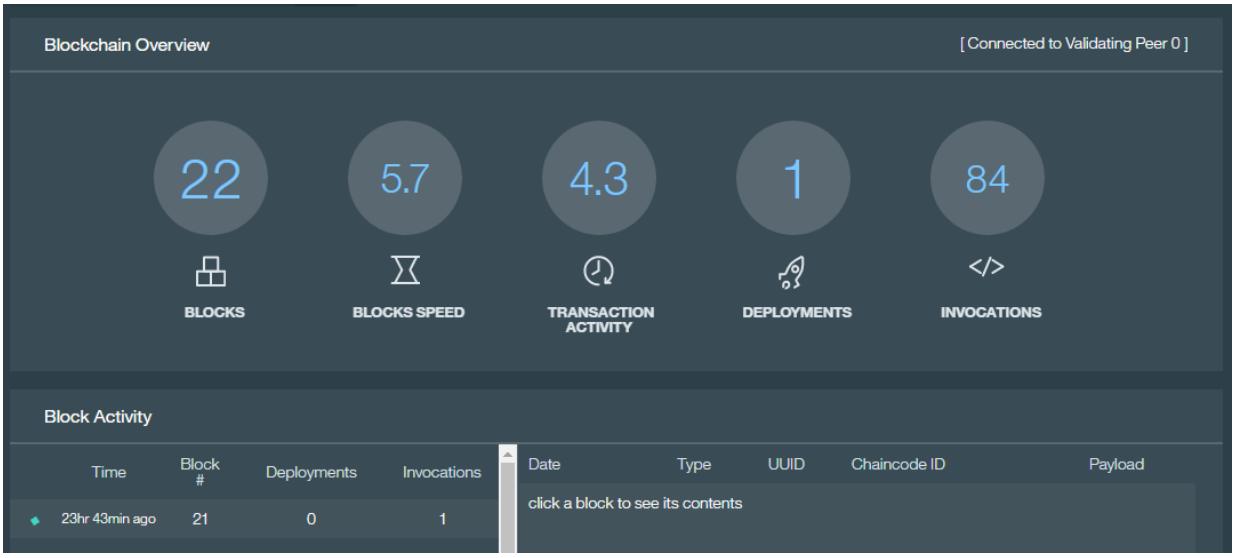


This view confirms that four validating peers and a certificate authority (Membership Services) are running under your Blockchain service. The block height should be the same for each of the validating peers.

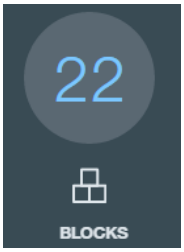
2.2 View the Blockchain Explorer

The **Blockchain** tab provides a visual representation of the state of the blockchain.

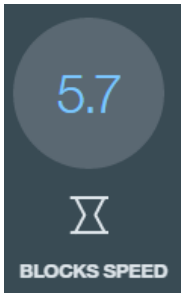
1. Click the **Blockchain** item on the left of the page.



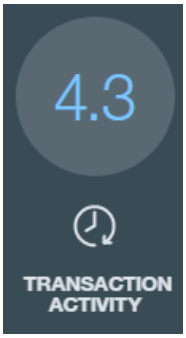
The icons show the following information:



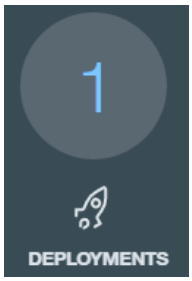
Total number of blocks in the chain.



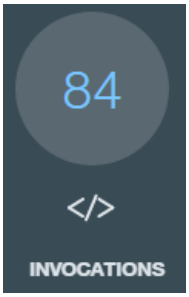
Average number of blocks per hour.



Number of transactions per block.



Number of deployment calls made to deploy chaincode.



Number of invoke requests made within this blockchain.

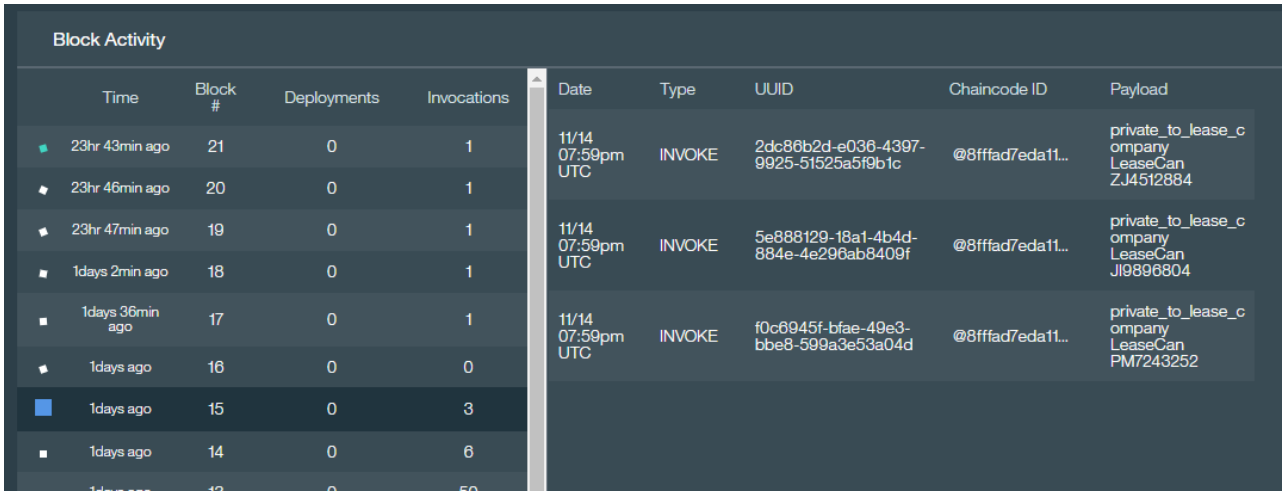
Each block contains a set of transactions. In Hyperledger Fabric, a transaction is the record of the request to interact with chaincode, a smart contract. Two important transaction types are:

- **INVOKE**: The request to invoke a piece of chaincode, such as invoking the chaincode to transfer the ownership of a car.
- **DEPLOY**: The request to deploy a piece of chaincode across all validating peers so that it can be executed at a later date.

Other request types exist, such as QUERY, UPDATE, and TERMINATE. Not all request types are recorded on the blockchain.

The blocks also include when that block was committed to the blockchain.

2. Click a block that contains at least one invocation request.
3. Look through the list of transactions that are contained in the block.



Each line of information is a transaction stored in the block. A block can contain multiple transactions, but in this demo, there will often be only one transaction per block because of the low frequency of transactions being made. The information being provided is:

- **Date**: The date the transaction was submitted.
- **Type**: The type of transaction taking place, such as INVOKE or DEPLOY.
- **UUID**: The unique identifier for each transaction.
- **Chaincode ID**: Refers to the chaincode that is being invoked or deployed.
- **Payload**: The input parameters of the chaincode.

4. Repeat this for other blocks to understand how the transactions are stored.

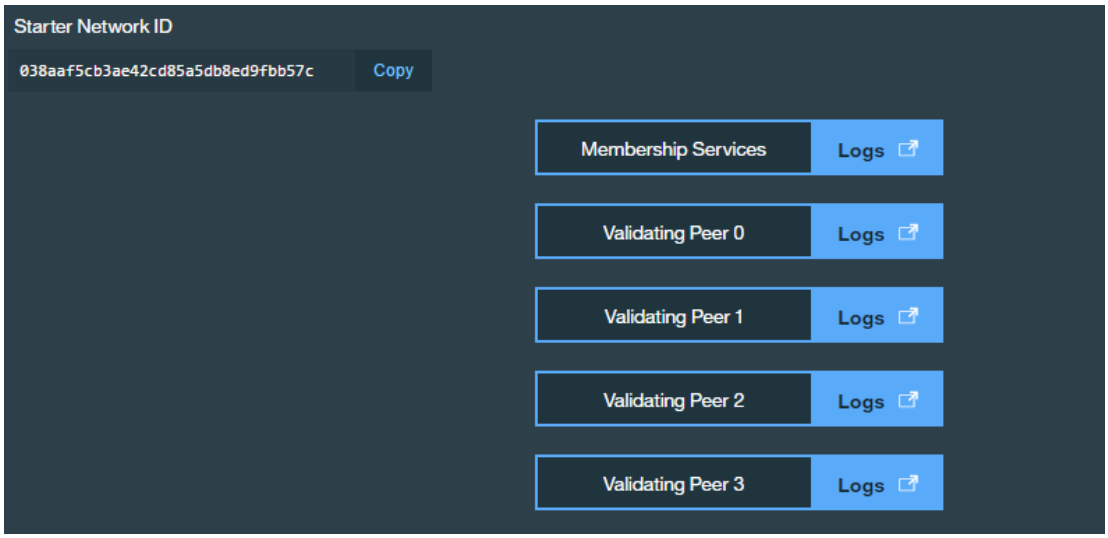
   When the blockchain is initialized for the car leasing application, the first block in the chain will usually contain a DEPLOY transaction by which the chaincode is deployed to the validating peers.

   You can view this block by scrolling down the Block Activity listing.

2.3 Understanding the blockchain peers

Now, you will review the logs associated with the peers. This is useful for understanding how the blockchain works and for diagnosing problems.

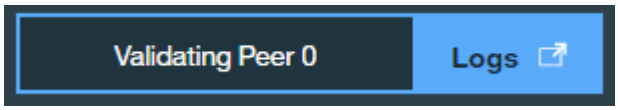
1. Click the **Logs** item on the service page.



Here we can see that this Blockchain service contains four validating peers and a Certificate Authority.

By looking at the logs for each peer you can verify that every node has executed every transaction.

2. Click the **Logs** button against one of the validating peers



This will show the logs for a selected peer in a new window.

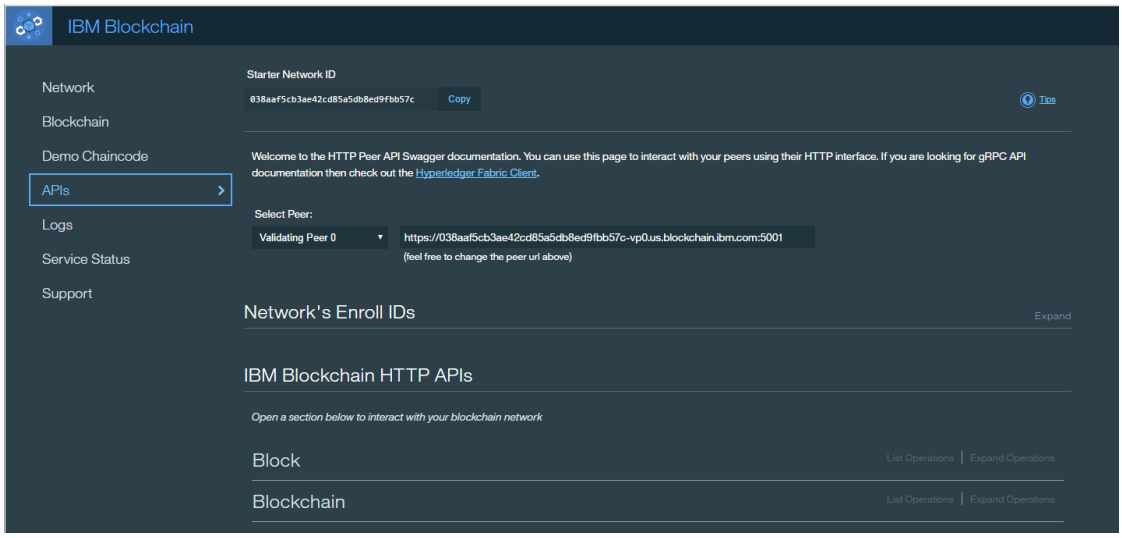
```
OUT - /scripts/start.sh -network_id 038aaf5cb3ae42cd85a5db8ed9fbb57c -peer_id vp0 -chaincode_host prod-us-01-chaincode-swarm-vp0.us.bloc
us.blockchain.ibm.com -port_discovery 30001 -port_rest 5001 -port_event 31001 -peer_enrollid peer0 -chaincode_tls true -peer_tls true -n
OUT - Enrollment secret is not passed calculating the default
OUT -
CORE_PEER_ID="vp0",CORE_PEER_NETWORKID="038aaf5cb3ae42cd85a5db8ed9fbb57c",CORE_PEER_ADDRESSAUTODETECT="false",CORE_PEER_LISTENADDRESS="0
0.0.0.0:30404",CORE_PEER_VALIDATOR_EVENTS_ADDRESS="0.0.0.0:31001",CORE_PEER_ADDRESS="038aaf5cb3ae42cd85a5db8ed9fbb57c-
vp0.us.blockchain.ibm.com:30001",CORE_LOGGING_PEER="warning",CORE_LOGGING_CRYPTO="warning",CORE_LOGGING_STATUS="warning",CORE_LOGGING_ST
,CORE_LOGGING_CHAINCODE="debug",CORE_PEER_LOGGING_LEVEL="warning",CORE_VM_ENDPOINT="tcp://prod-us-01-chaincode-swarm-
vp0.us.blockchain.ibm.com:3380",CORE_VM_DOCKER_TLS_ENABLED="true",CORE_VM_DOCKER_TLS_CERT_FILE="/certs/chaincode_host/cert.pem",CORE_VM_
R_TLS_CA_FILE="/certs/chaincode_host/ca.pem",CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE="us.blockchain.ibm.com",CORE_PEER_TLS_ENABLED="true",
FILE="/certs/peer/key.pem",CORE_PEER_TLS_SERVERHOSTOVERRIDE="038aaf5cb3ae42cd85a5db8ed9fbb57c-
vp0.us.blockchain.ibm.com",CORE_PEER_PKI_TLS_ENABLED="true",CORE_PEER_PKI_TLS_ROOTCERT_FILE="/certs/peer/cert.pem",CORE_PEER_PKI_TLS_SER
vp0.us.blockchain.ibm.com",CORE_PEER_DISCOVERY_PERIOD="60s",CORE_PEER_DISCOVERY_TOUCHPERIOD="60s",CORE_CHAINCODE_DEPLOYTIMEOUT="180000",
US_PLUGIN="pbft",CORE_PBFT_GENERAL_MODE="batch",CORE_PBFT_GENERAL_BATCHSIZE="1000",CORE_PBFT_GENERAL_TIMEOUT_BATCH="1s",CORE_PBFT_GENERA
",CORE_PBFT_GENERAL_TIMEOUT_RESENDVIEWCHANGE="30s",CORE_PBFT_GENERAL_TIMEOUT_NULLREQUEST="0s",CORE_STATETRANSFER_TIMEOUT_SINGLEBLOCK="60
TRANSFER_TIMEOUT_FULLSTATE="600s",CORE_PEER_DISCOVERY_ROOTNODE="038aaf5cb3ae42cd85a5db8ed9fbb57c-vp1.us.blockchain.ibm.com:30001,038aaf5
vp2.us.blockchain.ibm.com:30001,038aaf5cb3ae42cd85a5db8ed9fbb57c-
vp3.us.blockchain.ibm.com:30001",CORE_SECURITY_ENABLED="true",CORE_SECURITY_ENROLLID="peer0",CORE_SECURITY_ENROLLSECRET="08b7cf79a1",COR
ca.us.blockchain.ibm.com:30001",CORE_PEER_PKI_TCA_PADDR="038aaf5cb3ae42cd85a5db8ed9fbb57c-ca.us.blockchain.ibm.com:30001",CORE_PEER_PKI_
ca.us.blockchain.ibm.com:30001"
OUT - 2016-11-14 19:09:52,967 CRIT Supervisor running as root (no user in config file)
OUT - 2016-11-14 19:09:52,968 INFO supervisord started with pid 14
OUT - 2016-11-14 19:09:53,975 INFO spawned: 'start_peer' with pid 17
OUT - 19:09:54.019 [nodeCmd] serve -> INFO 001[0m Security enabled status: true
OUT - 19:09:54.019 [nodeCmd] serve -> INFO 003[0m Privacy enabled status: false
OUT - 19:09:54.019 [eventhub_producer] start -> INFO 002[0m event processor started
OUT - 19:09:54.020 [db] open -> INFO 004[0m Setting rocksdb maxLogFileSize to 10485760
OUT - 19:09:54.021 [db] open -> INFO 005[0m Setting rocksdb keepLogFileNum to 10
OUT - 19:09:54.027 [crypto] RegisterValidator -> INFO 006[0m Registering validator [peer0] with name [peer0]...
OUT - 19:09:54.271 [crypto] RegisterValidator -> INFO 007[0m Registering validator [peer0] with name [peer0]...done!
OUT - 19:09:54.271 [crypto] InitValidator -> INFO 008[0m Initializing validator [peer0]...
OUT - 19:09:54.277 [crypto] InitValidator -> INFO 009[0m Initializing validator [peer0]...done!
OUT - 19:09:54.277 [chaincode] NewChaincodeSupport -> INFO 00a[0m Chaincode support using peerAddress: 038aaf5cb3ae42cd85a5db8ed9fbb57c-
OUT - [33m19:09:54.277 [sysccapi] RegisterSysCC -> WARN 00b[0m Currently system chaincode does support security(noop,github.com/hyperled
OUT - 19:09:54.277 [state] loadConfig -> INFO 00c[0m Loading configurations...
OUT - 19:09:54.278 [state] loadConfig -> INFO 00d[0m Configurations loaded. stateImplName=[buckettree], stateImplConfigs=map[numBuckets:
bucketCacheSize:%!s(int=100)], deltaHistorySize=[500]
OUT - 19:09:54.278 [state] NewState -> INFO 00e[0m Initializing state implementation [buckettree]
OUT - 19:09:54.278 [buckettree] initConfig -> INFO 00f[0m configs passed during initialization = map[string]interface {}{"numBuckets":10
OUT - 19:09:54.278 [buckettree] initConfig -> INFO 010[0m Initializing bucket tree state implemetation with configurations &{maxGrouping
9:1000003 8:200001 7:40001 1:3 5:1601 0:1 6:8001 3:65] hashFunc:0xab4dc0}
OUT - 19:09:54.278 [buckettree] newBucketCache -> INFO 011[0m Constructing bucket-cache with max bucket cache size = [100] MBs
OUT - 19:09:54.278 [buckettree] loadAllBucketNodesFromDB -> INFO 012[0m Loaded buckets data in cache. Total buckets in DB = [0]. Total c
OUT - 19:09:54.278 [genesis] func1 -> INFO 013[0m Creating genesis block.
OUT - 19:09:54.279 [consensus/controller] NewConsenter -> INFO 014[0m Creating consensus plugin pbft
OUT - 19:09:54.280 [consensus/pbft] newPbftCore -> INFO 015[0m PBFT type = *pbft.obcBatch
```

2.4 Interacting with the peers

You can invoke the management APIs that interact directly with the peers. In this lab, you'll be trying out these APIs directly from the Bluemix environment.

Note that the APIs are used to *operationally manage* the blockchain. This is not the same as adding and invoking transactions through chaincode.

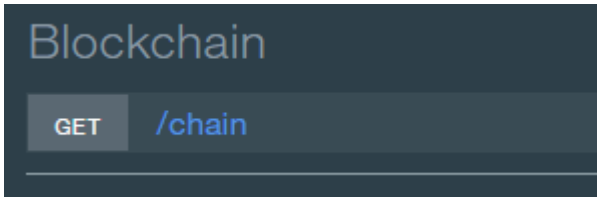
1. Click the **APIs** item on the Service page.



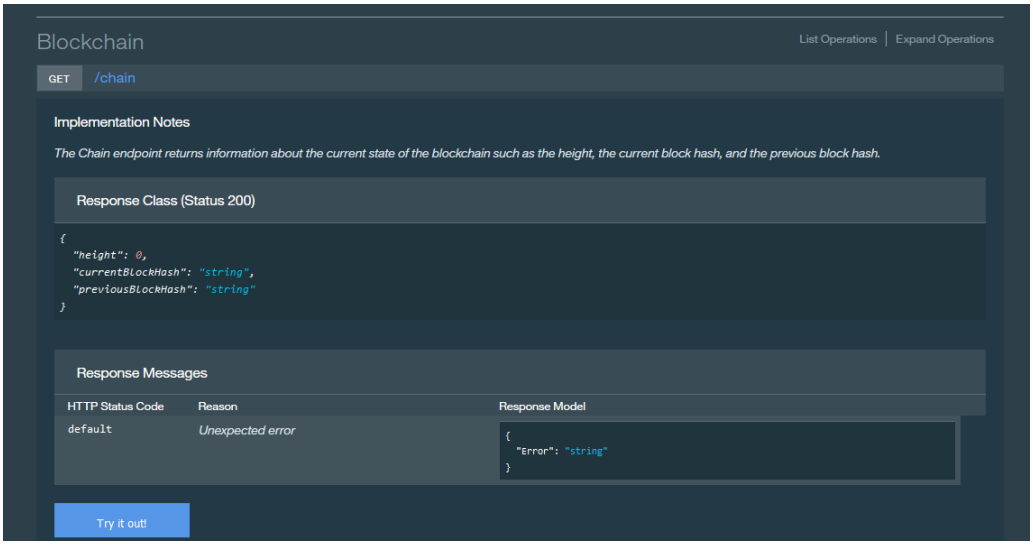
This page allows you to invoke APIs that will directly interrogate and manage the blockchain. First, you will use the API endpoint to query the height of the blockchain (the number of blocks).

2. Click the **Blockchain** section.

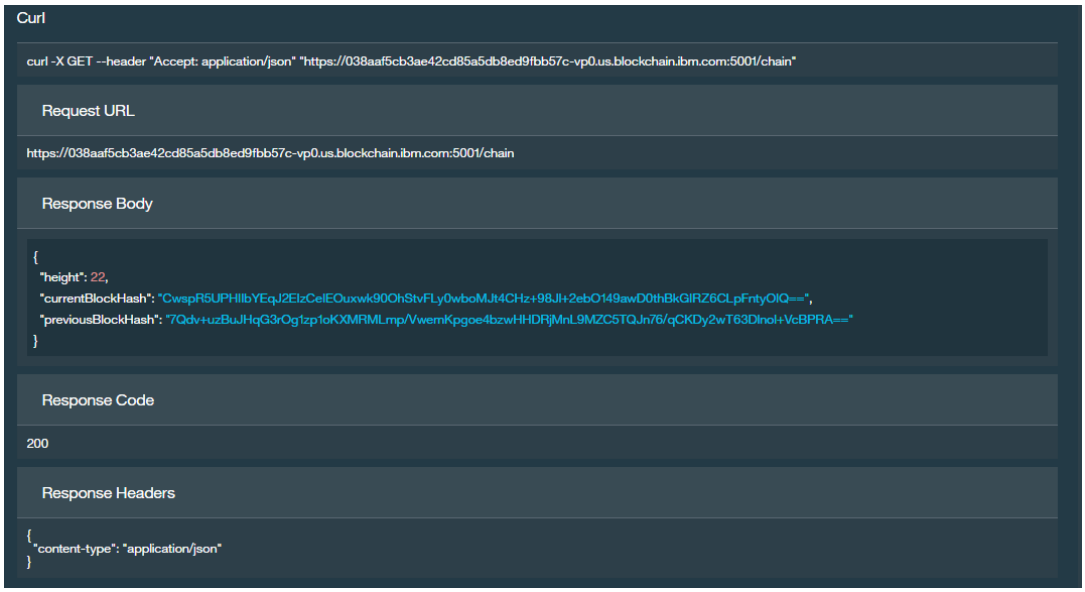
   This reveals the `GET /chain` operation, which is a valid method to call on the peer.



3. Click **Expand Operations** to view information about this API. This displays the input and output data formats.



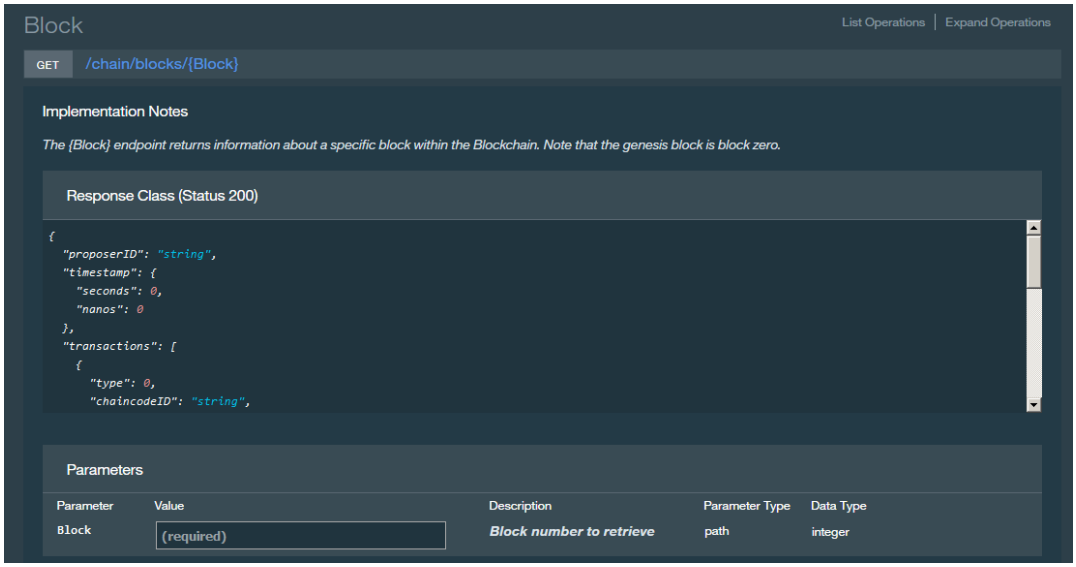
4. Click **Try It Out!** to invoke the API.



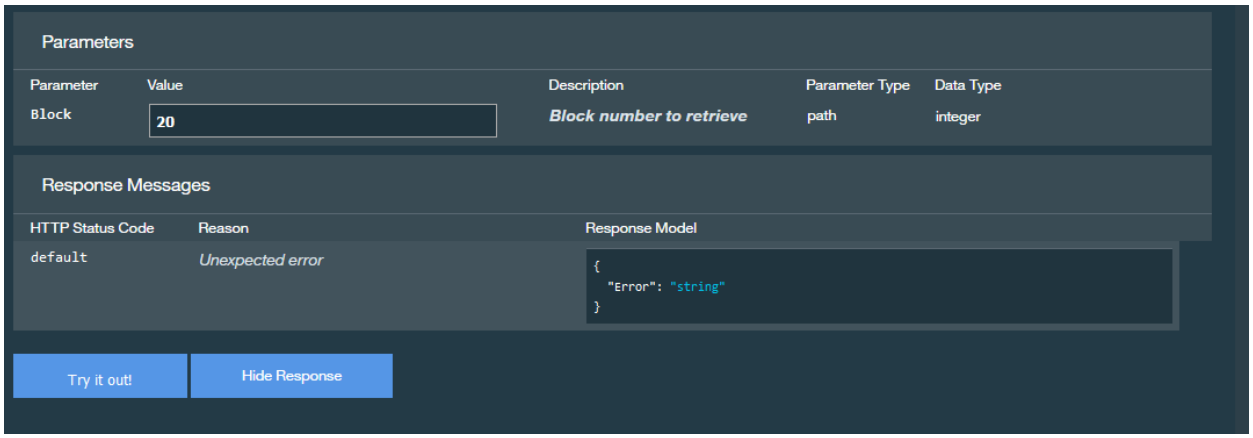
Review the displayed fields:

- **Request URL:** shows the URL that was invoked, including the endpoint information of the peer (hostname:port) and the method call `(/chain)`.
- **Response Body:** shows the information that was returned including, importantly, the height of the blockchain.
- **Response Code:** 200 shows that the request was successful.
- **Response Headers:** a field that confirms that the response body data was returned in a JSON data structure.

5. Expand the **Block** section and review the information about how to interrogate an individual block in the blockchain.



6. Enter the Block parameter to be a number less than the height of the chain and click **Try it out!**

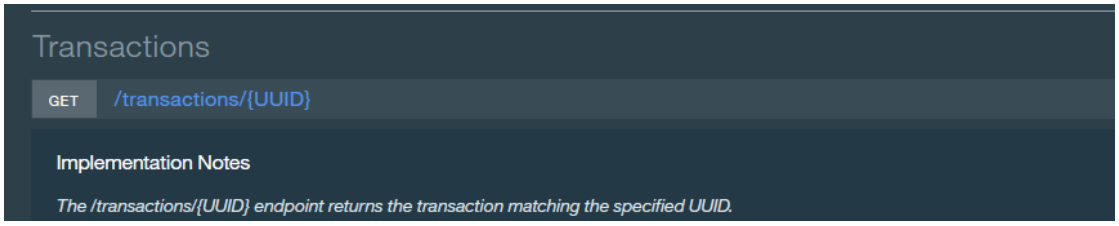


7. Review the information returned in the **Response Body**.

transactions	An array of transactions stored in the block.
type	This shows the possible values: 0. Undefined 1. Deployment 2. Invoke 3. Query

chaincodeID	ID of the chaincode that was invoked or deployed.
payload	Input parameters to the chaincode.
txid	Unique identifier of this transaction.
timestamp	Time at which the block or transaction order was proposed.
cert	Certificate of the participant submitting the transaction.
signature	Signature of the participant submitting the transaction.
stateHash	Hash of the world state changes.
previousBlockHash	Hash of the previous block in the chain.
nonHashData	Data stored with the block, but not included in the block's hash. This allows data to be different per peer or discarded without affecting the blockchain.
localLedgerCommitTimestamp	Time the block was added to the ledger on the local peer.

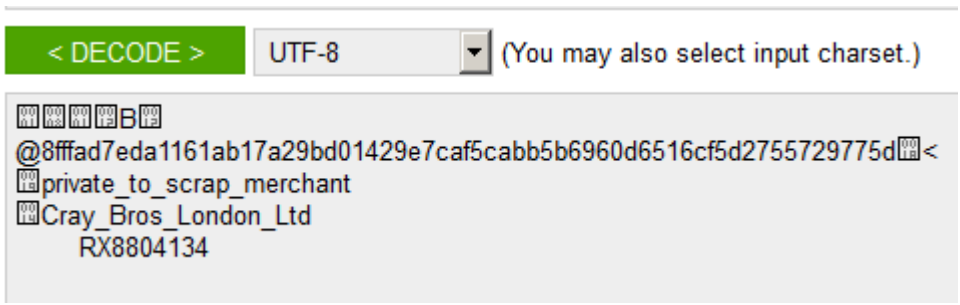
8. Copy the TXID field of a transaction from a block; this will be of the form `cfa79bf5-c5c4-454d-be9f-280c11bf8ad3`.
9. Click the **Transactions** section.



This reveals the `GET /transactions/{UUID}` operation, which is a valid method to call on the peer.

10. Paste the transaction id (TXID from step 8) and click **Try it out!**

The **payload** field is base64 encoded. You can use a web tool such as http://www.base64decode.org to decode this information. When the information is decoded, you'll see that the payload includes the chaincode ID of the smart contract being called together with its input parameters as shown in the image.



This application does not encrypt the transactions, so the payloads are visible to all although they are encoded in base64.

11. You may interact with the other APIs that are available as desired or continue to the final activity for this lab.

2.5 Viewing the Service Status, Support Contacts, and Samples

1. Click the **Service Status** item on the left of the service page.

This panel shows you the recent availability of the Blockchain service on Bluemix, and also the version of Hyperledger Fabric that is being used by your network.

2. Click the **Support** item on the left of the service page.

This page shows you how to get more help with IBM Bluemix and the Blockchain service.

3. Click the **Demo Chaincode** item on the left of the service page.

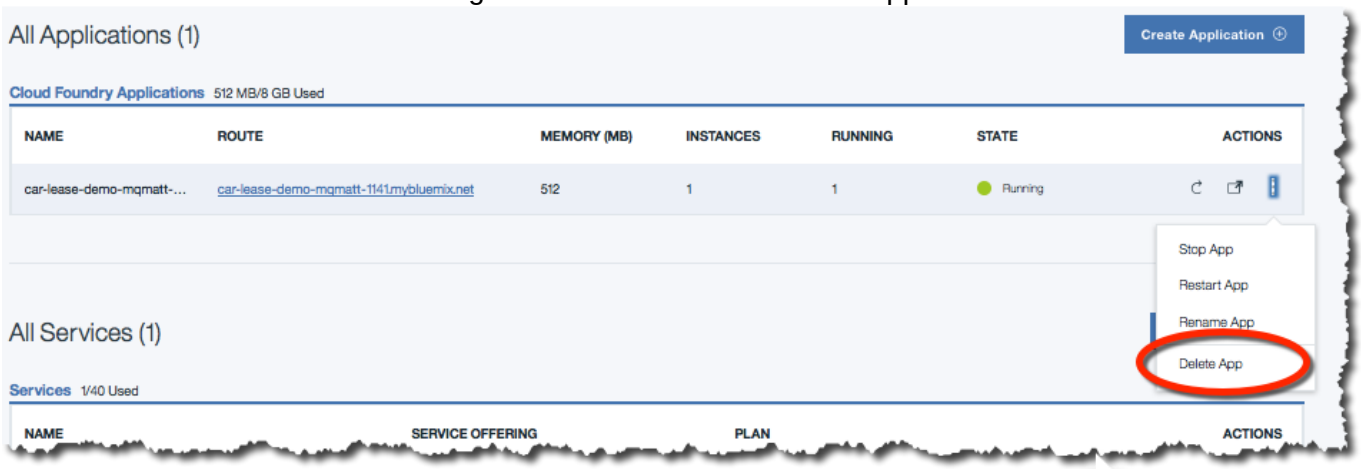
This page gives the opportunity to deploy more samples to the Blockchain service, and also how to get started with writing your own blockchain applications and chaincode.

We will look at chaincode development in more detail in the follow-on lab “Blockchain Unchained.”

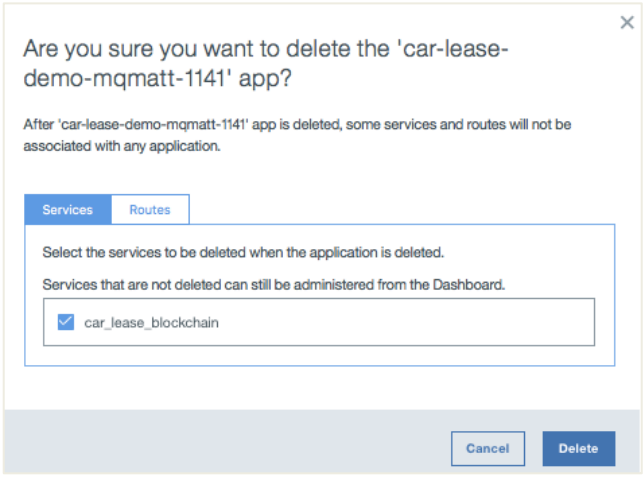
Step 3. Remove the sample application

You will need the car leasing demo for the rest of the course. When you are ready to delete it, follow these steps.

1. Click **Dashboard** to return to the Bluemix dashboard.
2. Click the three vertical dots on the right side of the car lease demo application row.



3. Select **Delete App** from the menu.
4. Ensure that the car_lease_blockchain service is also selected for deletion and click **Delete**.



Wait for the items to stop and be deleted. After this is done, both the application and the associated service will no longer be visible in the Bluemix dashboard.