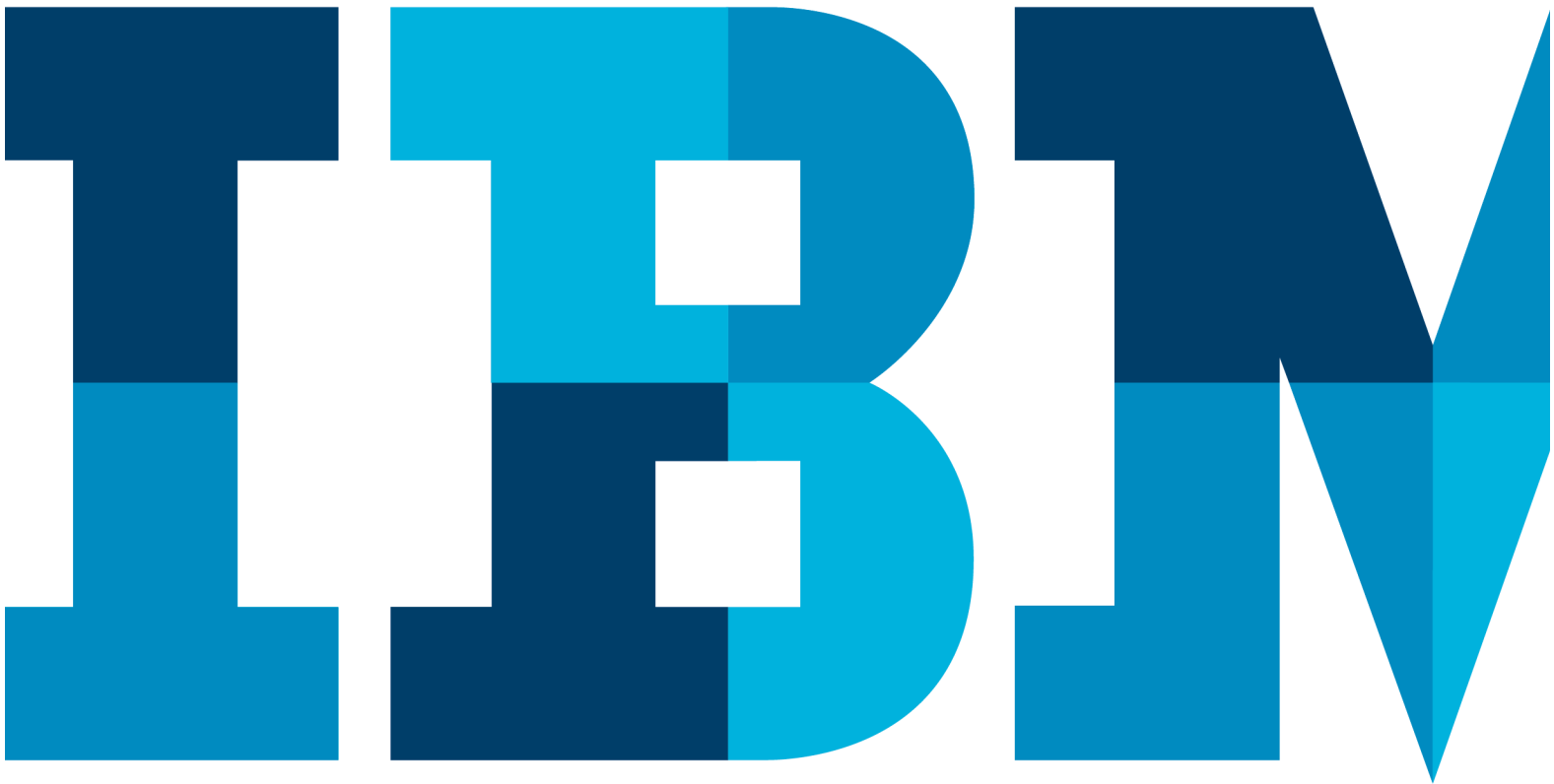


# IBM Blockchain Hands-On Blockchain Explored

*Lab Two – Bluemix – Exercises*



---

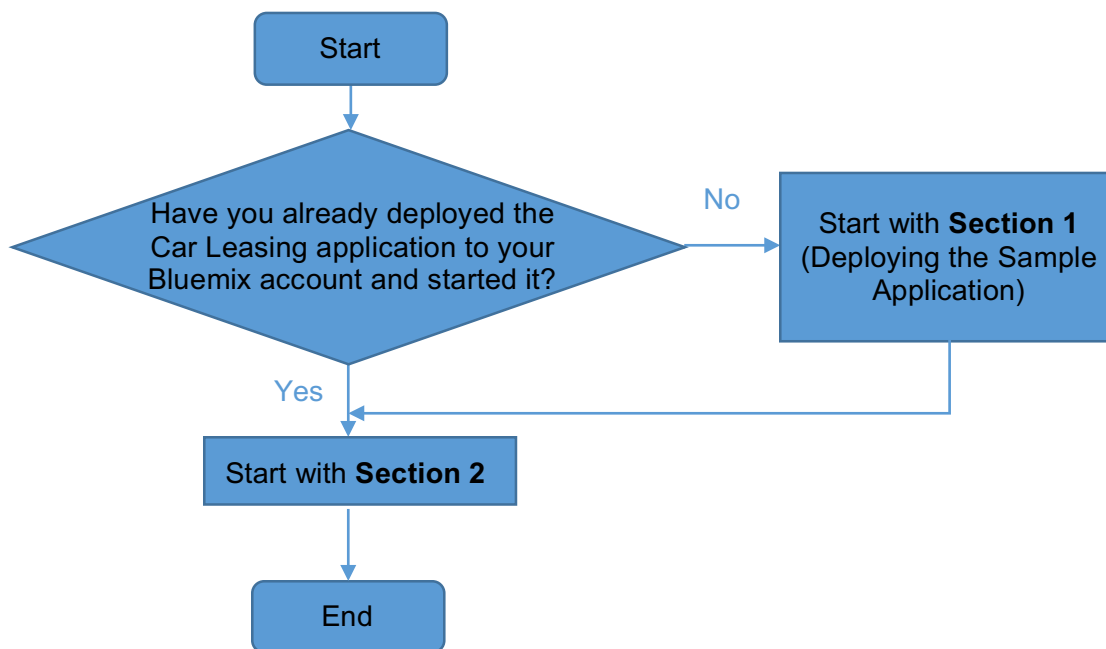
## Contents

<b>OVERVIEW</b>	<b>INTRODUCTION TO THE LAB.....</b>	<b>3</b>
<b>SECTION 1.</b>	<b>DEPLOYING THE SAMPLE APPLICATION.....</b>	<b>4</b>
	1.1. CREATING A BLOCKCHAIN SERVICE.....	4
	1.2. INITIALIZING THE CAR LEASING DEMO .....	6
<b>SECTION 2.</b>	<b>MANAGING THE SAMPLE APPLICATION.....</b>	<b>8</b>
	2.1. VIEWING THE COMPONENTS OF THE BLOCKCHAIN SERVICE.....	8
	2.2. VIEWING THE BLOCKCHAIN.....	9
	2.3. UNDERSTANDING THE BLOCKCHAIN PEERS.....	12
	2.4. INTERACTING WITH THE PEERS.....	13
	2.5. VIEWING THE SERVICE STATUS, SUPPORT CONTACTS AND SAMPLES .....	19
<b>SECTION 3.</b>	<b>SECURITY FUNDAMENTALS.....</b>	<b>20</b>
	3.1. TRANSACTION SIGNING.....	20
	3.2. MEMBERSHIP SERVICES CONCEPTS .....	23
	3.3. USING MEMBERSHIP SERVICES .....	24
<b>SECTION 4.</b>	<b>REMOVING THE SAMPLE APPLICATION.....</b>	<b>27</b>
<b>APPENDIX A.</b>	<b>NOTICES.....</b>	<b>28</b>
<b>APPENDIX B.</b>	<b>TRADEMARKS AND COPYRIGHTS.....</b>	<b>30</b>

## Overview Introduction to the Lab

The purpose of this lab is to introduce you to the Blockchain service on Bluemix. We will build on the car leasing scenario that was introduced in the “Blockchain Explained” lab.

If you are using your own Bluemix account and have already completed the previous lab, you will have already deployed the car leasing application to your account. You can skip section 1 and re-use your existing application:



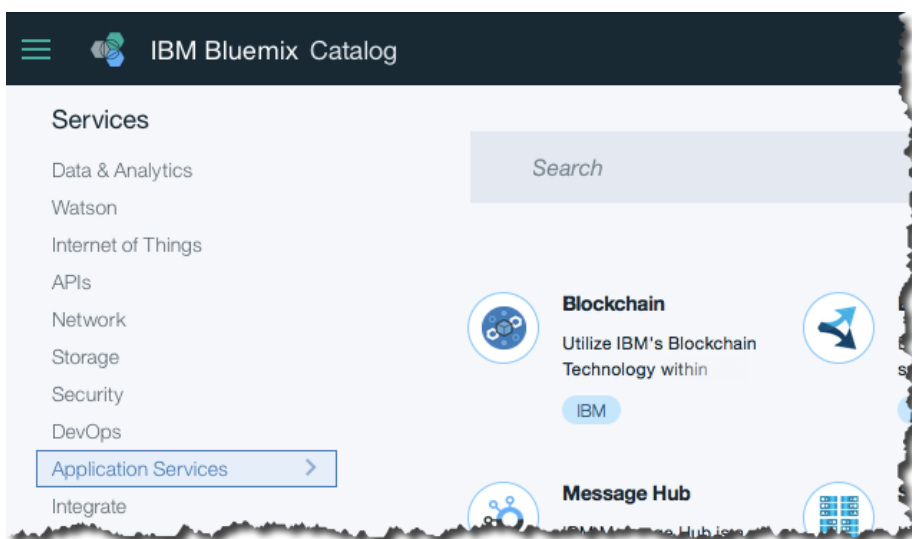
For V0.61 of Hyperledger Fabric in Bluemix

## Section 1. Deploying the sample application

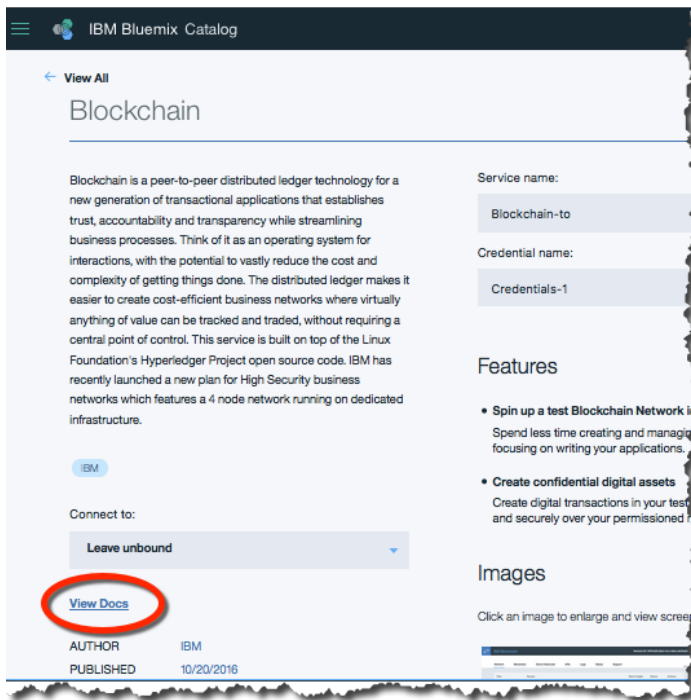
In this section we will use Bluemix to deploy a copy of the car leasing demo application.


### 1.1. Creating a Blockchain Service

- \_\_1. Open a web browser and go to [www.bluemix.net](http://www.bluemix.net).
- \_\_2. Click '**Sign Up**' or '**Log In**' to create a new Bluemix account or log into your existing account.
- \_\_3. Once you have successfully signed up and logged into Bluemix, select **Catalog** from the top bar.
- \_\_4. In the 'Services' section of the sidebar, click '**Application Services**' and select **Blockchain**.



- \_\_5. Review the service description and information about the service.
- \_\_6. Click '**View Docs**' and learn about the process of creating a blockchain environment.



- \_\_7. Click **'Samples and Tutorials'** on the right of the page.
- \_\_8. Click  against the Car Lease demo. Log in to Bluemix again if necessary.
- \_\_9. If you are asked to pick an alias (usually the first time a given Bluemix ID deploys a sample), come up with a unique username and review and accept the terms of use. Click Create and then Continue on the subsequent page.


**IBM Bluemix™ DevOps Services**

**Pick an alias**

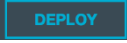
To set up your Git repository, we need to associate your IBM id with an alias.

An alias is a unique, publicly visible short name used in Git repository paths, Track & Plan, and desktop and command line clients.

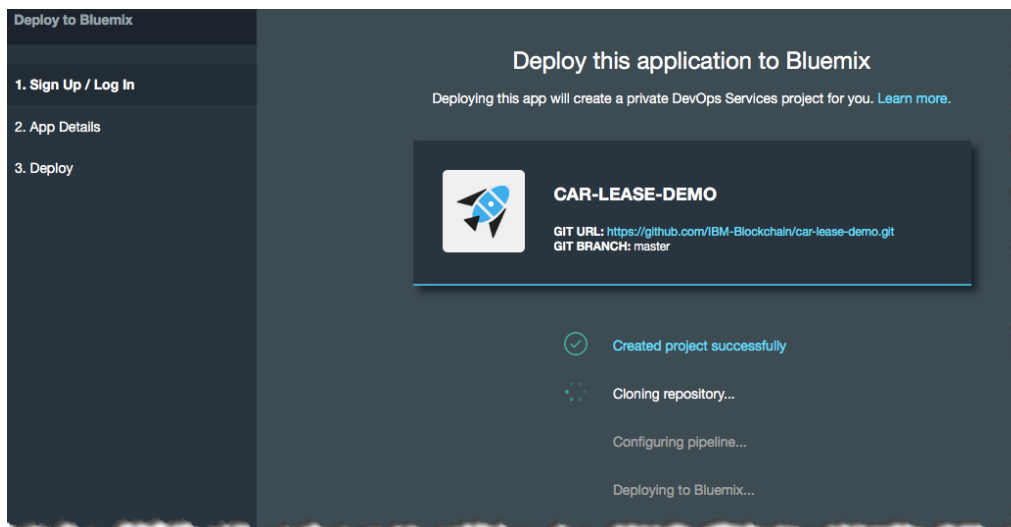


☐ I accept the DevOps Services [Terms of Use](#).

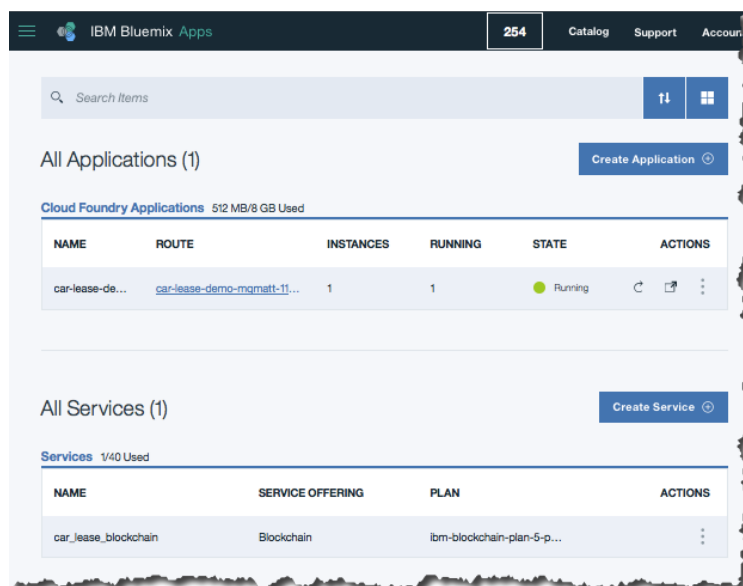
**Create**

- \_\_10. Wait a few seconds for the default field values to be populated. Then leave the App Name, Region, Organization and Space default (unless told otherwise) and click .

This will cause the car leasing demo to be deployed into your Bluemix environment, and will probably take a few minutes to complete.



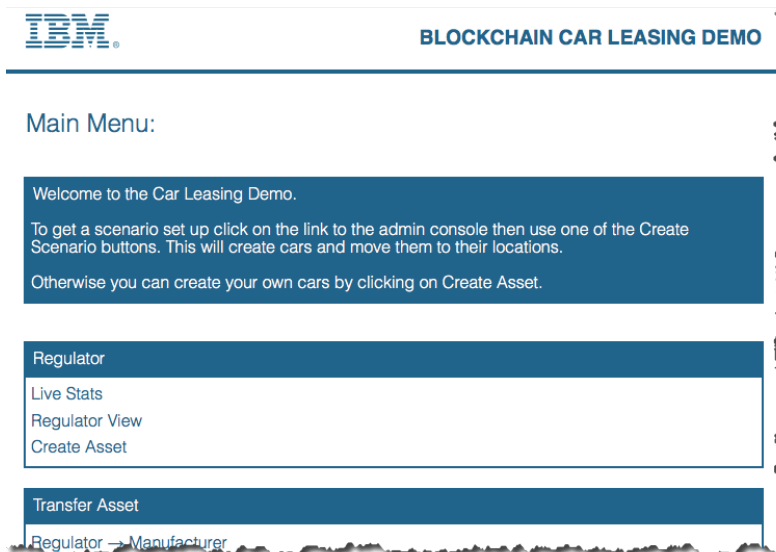
- \_\_\_11. Once you see the 'Success!' message click **DASHBOARD** to see a description of the new car leasing application (and associated Blockchain service) you created.



## 1.2. Initializing the Car Leasing Demo

- \_\_\_12. Click the blue hyperlink under the 'Route' column of your application, and this will load the Car Leasing demo webpage. (Do not click elsewhere on this line, as this will load the administration interface for the application, which we will look at later).

You will now see the front page of the car leasing demo.



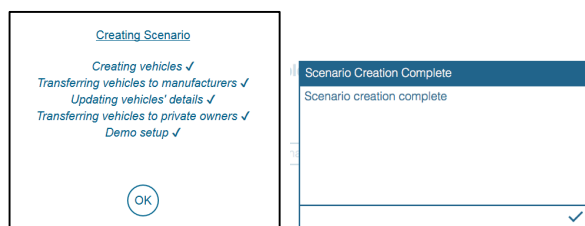
- \_\_\_13. From the Car Leasing demo front page, click '**Admin Console**' and '**Create Simple Scenario**'.



This will preload the blockchain with a set of transactions. (The Full Scenario works fine too; the difference between the Simple Scenario and the Full Scenario is that in the Full Scenario more assets are initially loaded onto the blockchain; this takes longer to initialize, however.)

Wait a few moments for the initialization to complete.


- \_\_\_14. Click '**OK**' to close the Creating Scenario log, and then dismiss the 'Scenario Creation complete' by clicking the check mark.

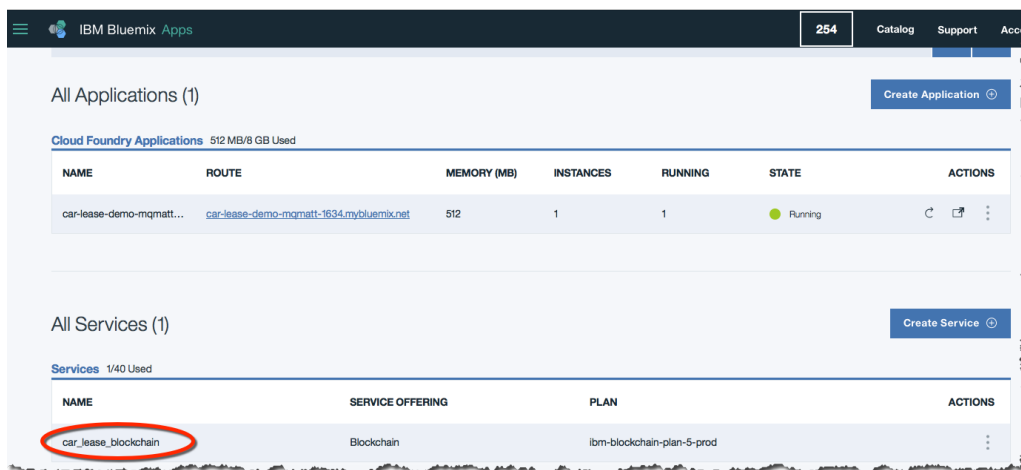


## Section 2. Managing the sample application

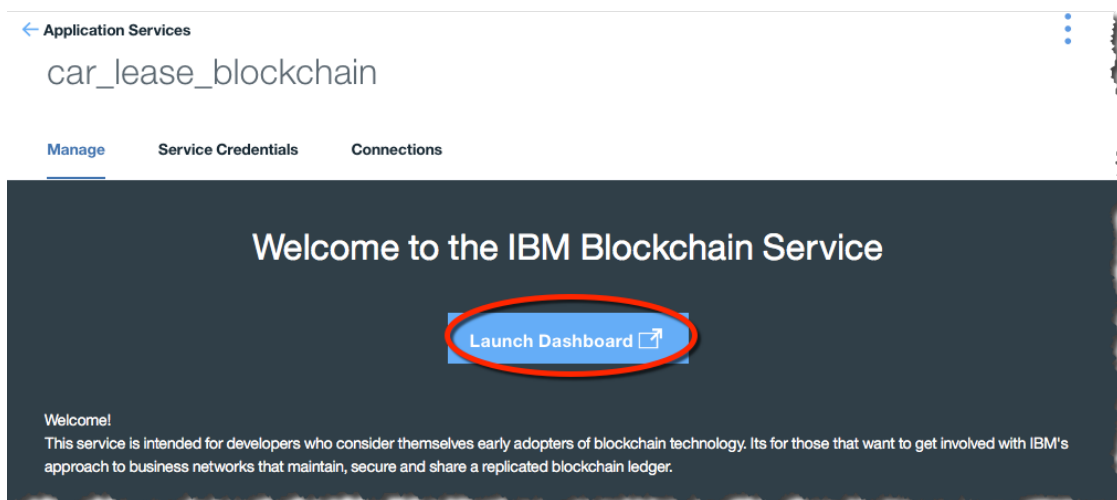
In this section we will use the monitoring tools available inside the Bluemix environment to view and manage the blockchain.

### 2.1. Viewing the components of the Blockchain service

- \_\_\_15. Return to the Bluemix dashboard, either by selecting the  icon in the top left of Bluemix and selecting '**Dashboard**' or by going directly to <https://console.ng.bluemix.net/dashboard/applications>.
- \_\_\_16. Click on the **car\_lease\_blockchain** service in the Services section of the dashboard.



- \_\_\_17. Review the details and select 'Launch Dashboard' to launch the blockchain administration console.





You should now see the administration page with seven tabs down the left-hand side. The **'Network'** tab will be selected by default.

The screenshot shows the IBM Blockchain administration interface. On the left is a sidebar with tabs: Network (selected), Blockchain, Demo Chaincode, APIs, Logs, Service Status, and Support. The main area displays the 'Network' tab content. At the top, it shows the 'Starter Network ID' as '6cdef08ce05c43579895f0c27cf5147b' with a 'Copy' button. Below this is a progress bar and a 'Refresh In 01:21 / 3:00' indicator. The central part of the page is a table with the following columns: Peer, Routes, Block Height, Status, and Actions.

Peer	Routes	Block Height	Status	Actions
Membership Services	gRPC grpcs... <a href="#">Copy</a>	-	Running	<a href="#">Stop</a> <a href="#">Refresh</a>
Validating Peer 0	HTTP https... <a href="#">Copy</a>	83	Running	<a href="#">Stop</a> <a href="#">Refresh</a>
Validating Peer 1	HTTP https... <a href="#">Copy</a>	83	Running	<a href="#">Stop</a> <a href="#">Refresh</a>
Validating Peer 2	HTTP https... <a href="#">Copy</a>	83	Running	<a href="#">Stop</a> <a href="#">Refresh</a>
Validating Peer 3	HTTP https... <a href="#">Copy</a>	83	Running	<a href="#">Stop</a> <a href="#">Refresh</a>

At the bottom of the table, there are tabs for 'Chaincode ID', 'Peers', 'Logs', and 'Status'.

The blockchain is a replicated, shared ledger. This blockchain is shared among all the participants of the network. Each participant still has their own copy of the ledger, and replication ensures that the copies are kept synchronised.

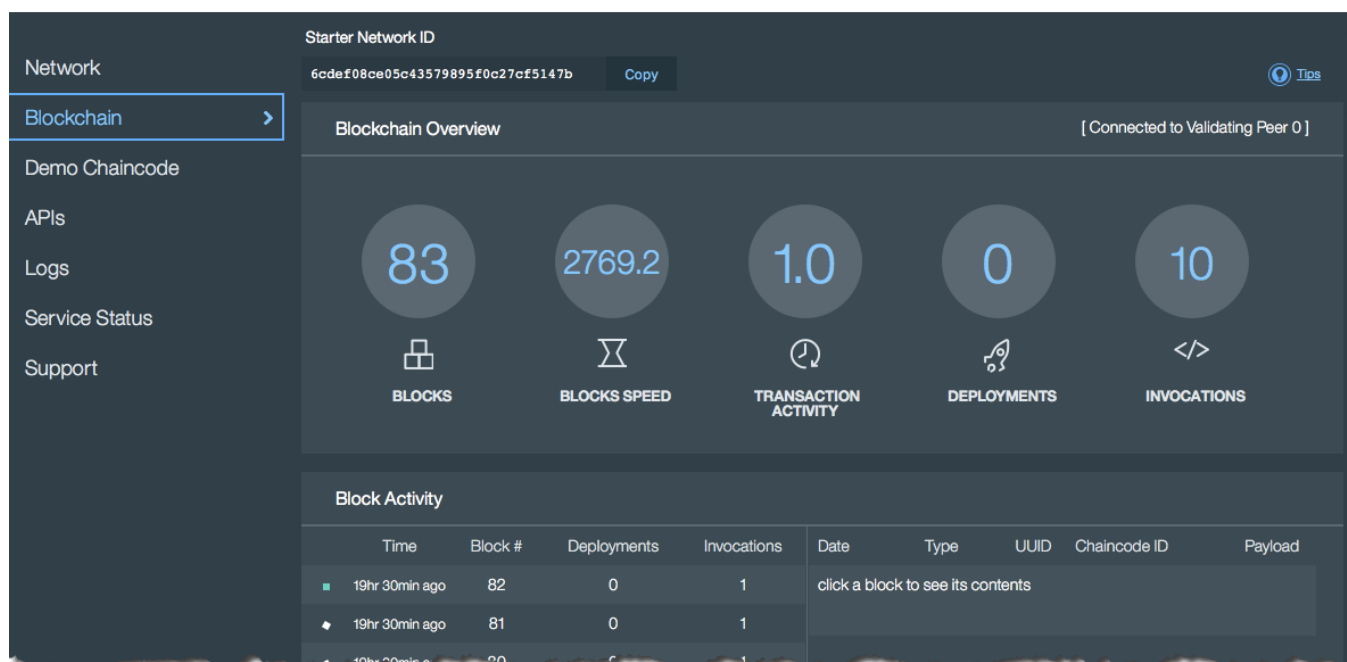
The blockchain network that has been set up for us in this demo contains four participants ("Validating Peers") as well as a Membership Services component that we will look at later. Applications submit transactions into just one validating peer, and peer-to-peer technology is used to replicate the transaction elsewhere.

- \_\_18. Verify that the four validating peers each have the same block height.
- \_\_19. Return to the car leasing demo and invoke another transaction. Verify that the block height increases by one for all four validating peers.

## 2.2. Viewing the Blockchain

The Blockchain tab shows a visual representation of the state of the Blockchain.

- \_\_20. Click the **'Blockchain'** tab at the left of the page.



The icons show:

	Total number of blocks in the chain
	Average number of blocks per hour
	Average 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). The two most important transaction types are:

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

Other request types exist (e.g. query). Not all request types are recorded on the blockchain.

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

\_\_21. Click on a block that contains at least one invocation request.

Block Activity					Date	Type	UUID	Chaincode ID	Payload
Time	Block #	Deployments	Invocations						
19hr 30min ago	82	0	1		11/22 03:33p m UTC	INVOKE	195b22ab-e9b3-408b-89aa-d526b43ae480	@6b3cb42ed2d8..	private_to_lease_co mpany LeaseCan G M1075403
19hr 30min ago	81	0	1						
19hr 30min ago	80	0	1						
19hr 30min ago	79	0	1						
19hr 30min ago	78	0	1						
19hr 30min ago	77	0	1						


\_\_22. Look through the list of transactions that are contained within the block.

Date	Type	UUID	Chaincode ID	Payload
11/22 03:33p m UTC	INVOKE	195b22ab-e9b3-408b-89aa-d526b43ae480	@6b3cb42ed2d8..	private_to_lease_co mpany LeaseCan G M1075403

Each line of information is a transaction stored within the block. A block may contain multiple transactions but in this demo there will often only be one transaction per block due to the low frequency of transactions being made. The information displayed is:

<b>Date</b>	The date the transaction was submitted.
<b>Type</b>	The type of transaction taking place (e.g. INVOKE or DEPLOY).
<b>UUID</b>	The unique identifier for each transaction.
<b>Chaincode ID</b>	Refers to the chaincode that is being invoked or deployed.
<b>Payload</b>	The input parameters to the chaincode.

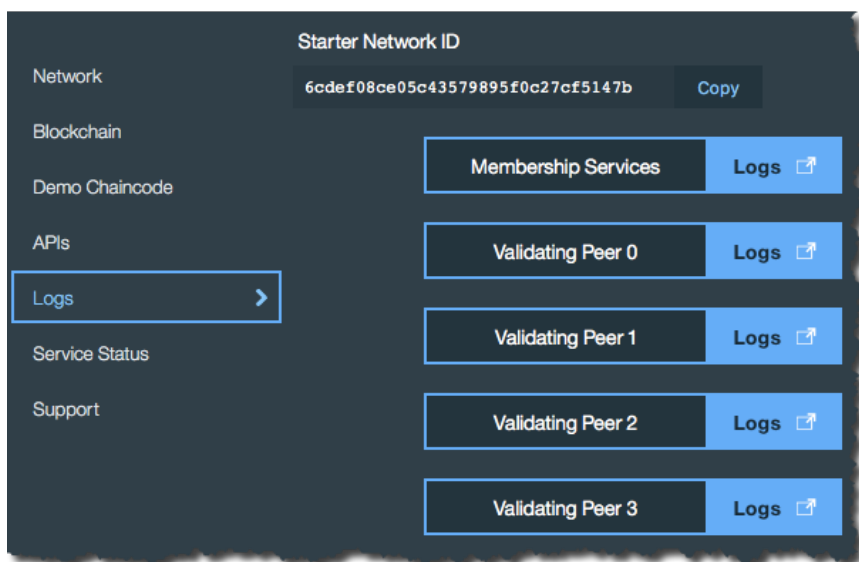
\_\_23. Repeat this for other blocks to understand how the transactions are stored.

	<p>When the Blockchain service is initialised for the car leasing application, the first block in the chain should contain a 'DEPLOY' transaction, where the chaincode is deployed to the validating peers.</p> <p>View these blocks If you're willing to scroll down the Blockchain explorer that far!</p>
---	---

## 2.3. Understanding the Blockchain Peers

We are now going to review the logs associated with the peers. This is useful for understanding how the blockchain works, and for diagnosing problems.

\_\_24. Click on the '**Logs**' tab.



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

\_\_25. Click the Logs button against one of the validating peers.



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

```

OUT - /scripts/start.sh -network_id 6cdef08ce05c43579895f0c27cf5147b -peer_id vp0 -chaincode_port 3380 -network_name us.blockchain.ibm.com -port_discovery 30001 -port_rest
-peer_tls true -num_peers 4
OUT - Enrollment secret is not passed calculating the default
OUT -
CORE_PEER_ID="vp0",CORE_PEER_NETWORKID="6cdef08ce05c43579895f0c27cf5147b",CORE_PEER_ADDRESS="0.0.0.0:5001",CORE_CLI_ADDRESS="0.0.0.0:30404",CORE_PEER_VALIDATOR_EVENTS_ADDRESS="vp0.us.blockchain.ibm.com:30001",CORE_LOGGING_PEER="warning",CORE_LOGGING_CRYPTOP="warning",CORE_LOGGING_LOGIN="warning",CORE_LOGGING_VM="debug",CORE_LOGGING_CHAINCODE="debug",CORE_PEER_LOGGING_ADDRESS="vp0.us.blockchain.ibm.com:3380",CORE_VM_DOCKER_TLS_ENABLED="true",CORE_VM_DOCKER_TLS_CERT_FILE="/certs/chaincode_host/key.pem",CORE_VM_DOCKER_TLS_CA_FILE="/certs/chaincode_host/ca.pem",CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE="us.blockchain.ibm.com",CORE_PEER_TLS_ENABLED="true",CORE_PEER_TLS_KEY_FILE="/certs/peer/key.pem",CORE_PEER_TLS_SERVERHOSTOVERRIDE="vp0.us.blockchain.ibm.com",CORE_PEER_PKI_TLS_ENABLED="true",CORE_PEER_PKI_TLS_ROOTCERT_FILE="/peer/cert.pem",CORE_PEER_PKI_TLS_SERVERHOSTOVERRIDE="6cdef08ce05c43579895f0c27cf5147b-vp0.us.blockchain.ibm.com",CORE_PEER_DISCOVERY_PERIOD="60s",CORE_PEER_DISCOVERY_TOUCHPERIOD="30000",CORE_PEER_VALIDATOR_CONSENSUS_PLUGIN="pbft",CORE_PBFT_GENERAL_MODE="batch",CORE_PBFT_GENERAL_TIMEOUT_REQUEST="30s",CORE_PBFT_GENERAL_TIMEOUT_VIEWCHANGE="30s",CORE_PBFT_GENERAL_TIMEOUT_UEST="0s",CORE_STATETRANSFER_TIMEOUT_SINGLEBLOCK="600s",CORE_STATETRANSFER_TIMEOUT_SINGLEBLOCK_ROOTNODE="6cdef08ce05c43579895f0c27cf5147b-vp1.us.blockchain.ibm.com:30001,6cdef08ce05c43579895f0c27cf5147b-vp2.us.blockchain.ibm.com:30001,6cdef08ce05c43579895f0c27cf5147b-vp3.us.blockchain.ibm.com:30001",CORE_SECURITY_ENABLED="true",CORE_SECURITY_ENROLLID="6cdef08ce05c43579895f0c27cf5147b-ca.us.blockchain.ibm.com:30001",CORE_PEER_PKI_TCA_PADDR="6cdef08ce05c43579895f0c27cf5147b-ca.us.blockchain.ibm.com:30001",CORE_PEER_PKI_TLSCA_PADDR="6cdef08ce05c43579895f0c27cf5147b-ca.us.blockchain.ibm.com:30001"
OUT - 2016-11-22 14:53:46,443 CRIT Supervisor running as root (no user in config file)
OUT - 2016-11-22 14:53:46,445 INFO supervisor started with pid 13
OUT - 2016-11-22 14:53:47,447 INFO spawned: 'start_peer' with pid 16
OUT - 14:53:47.489 [nodeCmd] serve -> INFO 001[0m Security enabled status: true
OUT - 14:53:47.490 [nodeCmd] serve -> INFO 003[0m Privacy enabled status: false
OUT - 14:53:47.490 [eventhub producer] start - INFO 002[0m Eventhub producer started

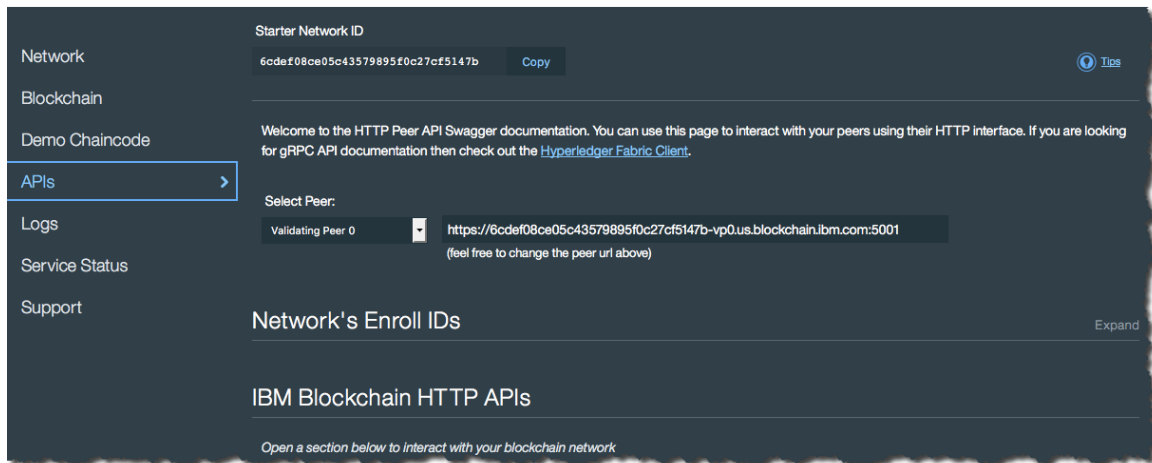
```

## 2.4. Interacting with the peers

It is possible to invoke the management APIs that interact directly with the peers. In this section we will be trying out these APIs directly from the Bluemix environment.

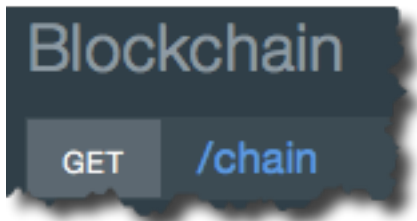
Note that the APIs concern *operationally managing* the Blockchain service – this is not the same as adding and invoking transactions through chaincode!

\_\_\_26. Click on the **'APIs'** tab on the dashboard.



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

\_\_27. Click the '**Blockchain**' section.



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

\_\_28. Select the operation to view information about it.

This reveals the input and output data formats.

Blockchain List Operations | Expand Operations

GET [/chain](#)

### Implementation Notes

The Chain endpoint returns information about the current state of the blockchain such as the height, the current block hash, and the previous block hash.

**Response Class (Status 200)**

```
{
  "height": 0,
  "currentBlockHash": "string",
  "previousBlockHash": "string"
}
```

**Response Messages**

HTTP Status Code	Reason	Response Model
default	Unexpected error	<pre>{   "Error": "string" }</pre>

[Try it out!](#)

\_\_29. Click **Try It Out** to invoke the API.

Curl

```
curl -X GET --header "Accept: application/json" "https://6cdef08ce05c43579895f0c27cf5147b-vp0.us.blockchain.ibm.com:5001/chain"
```

**Request URL**

https://6cdef08ce05c43579895f0c27cf5147b-vp0.us.blockchain.ibm.com:5001/chain

**Response Body**

```
{
  "height": 85,
  "currentBlockHash": "53PweWlNmVKinR0cSh6+GiZJgCn2EAGlF1Z4uaWj5tVjeMWIGYuOmGOMgzE4DvAjuvOGtKz6aAfV5xrGc0Afhg==",
  "previousBlockHash": "7k6bUDgu+0HEeO1Papp/qU3nWAAW/8xglgzfC9nzT7XoelKpsvw4JNRUYLPcn4UVQPDq/8gOHMto54ovRidww=="
}
```

**Response Code**

200

**Response Headers**

```
{
  "content-type": "application/json"
}
```

Review the displayed fields:



- The *Curl* field shows how to perform the same request from a command-line or script.
- The *Request URL* shows the URL that was invoked, including the endpoint information of the peer (hostname:port) and the method call (/chain).
- The *Response Body* shows the information that was returned including, importantly, the height of the blockchain.
- The *Response Code* 200 shows that the request was successful.
- The *Response Headers* confirms that the response body has been returned in a JSON data structure.

The blockchain is immutable: it is append-only and transactions cannot be modified or deleted once committed to the blockchain. Hash functions are used to link the blocks in the chain together; each block is linked to the previous block by a hash of the previous block's contents. If transactions are tampered with, the hash function returns a different value which renders the blockchain un-navigable.

A hash function is simply a function that is applied to a data set that produces a consistent output. It is usually used to map data of an arbitrary size to data of a fixed size. Importantly for blockchain, any change to the input data set will produce a different hash output, which can be used to easily detect any modifications to a block.

In the JSON response body, the '*height*' field shows the number of blocks in the blockchain; the '*currentBlockHash*' is a hash function that has run over the most recent block and '*previousBlockHash*' is the same for the block before it.

- \_\_\_30. Note the first few characters of the value of the *currentBlockHash* ("53Pwe..." in the previous screenshot).
- \_\_\_31. Invoke another transaction in the car leasing demo to force another block to be created.
- \_\_\_32. Re-run the GET /chain operation; verify that the height has increased by 1 and that the new *previousBlockHash* is the same as the previous block's *currentBlockHash* ("53Pwe...").

```
{
  "height": 86,
  "currentBlockHash": "YTrmXTeQG8WAbwds14e+x9Hejj6qx5mgOWPANnM4RBxGK0/E4bSPf8AE7K7CC-
  "previousBlockHash": "53PweWINmVKinR0cSh6+GiZJgCn2EAGIF1Z4uaWj5tVjeMWIGYuOmG0mgzE4Dv
}
```

- \_\_\_33. Click the 'Block' section and click on the 'GET /chain/blocks/{Block}' operation. In the 'Block' text field, enter the number *one less* than the current height of the chain (for example if the height was 86, enter 85).



```
{
  "transactions": [
    {
      "type": 2,
      "chaincodeID": "EkA2YjNjYjQyZWQyZDhiZDdmNzcwMmUxMmFmYzViMDViZmQzNWY4YmM0MzMzU5NTk0NTdlNDY5Yy",
      "payload": "CncIARJCEkA2YjNjYjQyZWQyZDhiZDdmNzcwMmUxMmFmYzViMDViZmQzNWY4YmM0MzMzU5NTk0NTdlNDY5Yy",
      "txid": "0442f7d-652a-491d-90b0-7bc9f29b2cd85",
      "timestamp": {
        "seconds": 1479902045
      },
      "nonce": "mHCyiTy4okJd5mx+9g0/BpdmEGENLr+W",
      "cert": "MIICmTOCAj6gAwIBAgIRAPb3b/snTkG7s8frgfFt9R8wCgYIKoZIzj0EAwMwKTElMAkGA1UEBhMCVVMxDDAI",
      "signature": "MEYCIQDPivGqmNLEP1JdGAk49RGsZzwAT7rc6YNld5SelPPcQlhAJYvkEaYY'IXn8JRGMC6j+exo4wd7Y"
    }
  ],
  "stateHash": "RLQv8Ypzs79ugqkixhXQS9y+gR4Z5Thq2shVWeHLR8Pb4f946y1+LRp6FArll0bqeQAOnsN/wxrYrid5niybQ:",
  "previousBlockHash": "53PweWINmVKinR0cSh6+GiZ.JgCn2EAGIFIZ4uaWj5tVjeMWIGUYOmGOmgzE4DvAjuvOGitKz6aAt",
  "consensusMetadata": "CFU=",
```

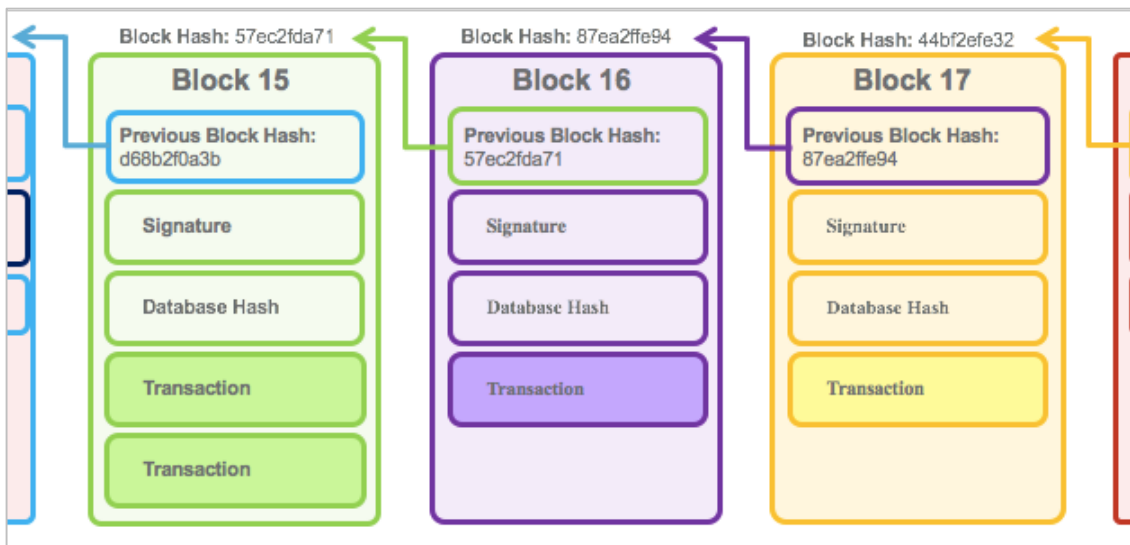
The returned JSON structure contains several elements, including:

- (a) a *transactions* array, which describes the set of transactions in the block. The description of each transaction includes its type (1/invoke or 2/deploy), the unique identifier of the associated chaincode and the encoded input parameters to it (payload).
- (b) a *stateHash*, which is the result of running a hash function over the transaction output,
- (c) a *previousBlockHash*, which is the result of running a hash function of the previous block in the chain, and
- (d) a *nonHashData* element, which contains data that will *not* be used in the computation of the next block's *previousBlockHash*.

Note that the *previousBlockHash* field matches the *previousBlockHash* returned by the GET /chain operation.

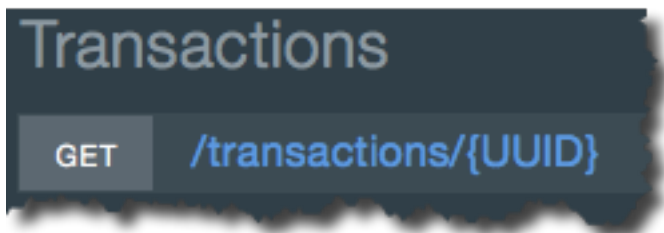
When a new block is created, a hash function is run over the entire previous block (except the *nonHashData* element) and the result stored in the *previousBlockHash* element. This way, if any earlier block in the chain is tampered with, subsequent blocks will be invalid.

Particularly, note that the *previousBlockHash* element is *itself* used to calculate the hash of subsequent blocks. This means that even a small change to one of the first blocks in the chain can be detected in any future block.



We will look at some other fields in this data structure in the next section.

- \_\_34. Copy the *txid* field of a transaction from a block; this will be a unique identifier of the form "04421f7d-652a-491d-90b0-7bc9f29b2d85".
- \_\_35. Click the '**Transactions**' section.



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

- \_\_36. Paste the transaction UUID and click '**Try it out!**'.

The 'payload' field is base64 encoded (use a web tool such as <http://www.base64decode.org> for decoding this information); when decoded you'll see that the payload includes the chaincode ID of the smart contract being called together with its input parameters. For example:



Note that this application does not encrypt the transactions, so the payloads are visible (albeit base64 encoded) to all.

## 2.5. Viewing the Service Status, Support Contacts and Samples

- \_\_37. Click on the '**Status**' tab at the top of the service page.

This page 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.

- \_\_38. Click on the '**Support**' tab at the top of the service page.

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

- \_\_39. Click on the '**Demo Chaincode**' tab at the top of the service page.

This page gives the opportunity to deploy more samples to the Blockchain service, and also some 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".

## Section 3. Security Fundamentals

Despite not using or requiring cryptographic mining, Hyperledger Fabric uses cryptography and other security principals to great effect. In this section of the lab we will look at how the underlying technology uses concepts such as signing and certificates.

### 3.1. Transaction Signing

\_\_40. In the blockchain dashboard, click 'APIs'.

\_\_41. If necessary, re-run the GET /chain/blocks/{Block} operation to return the details of a single block.

```
{
  "transactions": [
    {
      "type": 2,
      "chaincodeID": "EoABMTZlNjU1YzBmY2U2YTtk4ODI4OTZkM2Q2ZDExZjZkY2Q0ZjQ1MDI3ZmQ0NzY0MDA0NDQwZmYxZTYxMzQwOTEwYTkNjc2ODVjNGJiNzIzMjcYQTQ5N2YzY2Y0MjhlNmNmNmIwMDk2MTg2MTIyMjB1MTQ3M3WUwM2I2YzBhYTc2Y2I=",
      "payload": "CqQBCAESgWESgAExNmU2NTVjMGZjZTZhOTg4Mjg5NmQzZDZkMTFmN2RjZDRmNDUwMjdmZDQ3NjQwMDQ0NDBmZjF1NjEzNDA5MTBhOWQ2NzY4NWM0YmI3MjMyNzJhNDk3ZjNjZjQyOGU2Y2Y2YjAwOTYxODYxMjIyMGUxNDcxZTAzYjZjMGFhNzZjYhoaCghzZXRfdXNlcoHcnZ5OTIwZQoFbGVyb3k=",
      "txid": "256cb7a9-1c50-47a8-a078-0bb5bf96fc79",
      "timestamp": {
        "seconds": 1479732624,
        "nanos": 860673029
      },
      "nonce": "Hil0TKfQb80X6WuAzF4L2f+O1WtDZ81r",
      "cert": "MIICQDCCAeegAwIBAgIQDBdV+Atis7SQwj+Uha36rTAKBggqhkJOPQQDAZApMQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMQwwCgYDVQQDEwN0Y2EwHhcNMjYxMTIzNjQyWhcNMjYxMTIzNjQyWjA9MQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMSAwHgYDVQQDEXdUcmFuc2FjdG1vbWV3ZjZkY2Q0ZjQ1MDI3ZmQ0NzY0MDA0NDQwZmYxZTYxMzQwOTEwYTkNjc2ODVjNGJiNzIzMjcYQTQ5N2YzY2Y0MjhlNmNmNmIwMDk2MTg2MTIyMjB1MTQ3M3WUwM2I2YzBhYTc2Y2I=BAQDAgeAMAwGA1UdEwEB/wQCMAAwDQYDVVR0OBAYEBAECAwQwDwYDVR0jBAGwBoAEAAQIDBDBNBGyQAwQFBgcBAf8EQPR58QFNvEdqgHctksWMJ++AKG5rsWINnJVLn1VPyocrTXehS7HHMSDl+stvl+GKsVasbFmQIY1PhStJLQ31a4wSgYGKgMEBQYIBECZj3vpacz9rlXCfQy6nTLKEhc1HQjDJGIR5jrHaK7jpoYItcQA2Ae6nFlJRqkePunKg5c7wV4RkdI0pz+rqkvfMAoGCCqGSM49BAMDA0cAMEQCIFFipIruyutTlbKXLYNzzfC8N4hxxzOQvcABzDU926i8wYAiAJudsovUZketRmEzR/CwpAZqTJ6f1pjZ/FG9Qa4V8Geg==",
      "signature": "MEQCIIHkIzR2C8phQB2/sla/MVg9bZtHYZ/rUJJKFehJEZ7DWAiAQAni1Nz7QtGhOU7hnNdcpkPPSWWnqgBzQPQklrdzu1g=="
    }
  ],
  "stateHash": "HpBslIqtVTXTi4tVOKp7gKD6WVrLs9mjM7Z3O7p0mYHIiXkcHrseQNdCRuEqkFjYBfBcOytlykSXXR0s5o5sHA==",
  "previousBlockHash": "LSK+4Im1LZB0HimLyzTeQnsK3FudB6dWf9PjyvQtJFst5t6MbkuZn1fVDFvUQLU3qnIW1ggSzbrnlfXulqaeg==",
  "consensusMetadata": "CAQ=",
  "nonHashData": {
    "localLedgerCommitTimestamp": {
      "seconds": 1479732625,
      "nanos": 865115723
    },
    "chaincodeEvents": [
      {}
    ]
  }
}
```

The block also reveals information related to privacy and traceability.

Unlike the pseudonymous Bitcoin network, this blockchain can also be private and permissioned. Public key cryptography can ensure that the membership of the network is restricted to authorised participants, and that individual transactions are traceable (i.e. can be signed) and have appropriate privacy (i.e. can be encrypted).

This demo does not encrypt its transaction payloads but the transactions are signed.

42. Review the 'cert' and 'signature' elements of an individual transaction.

```
"cert":
"MITICQDCCAeegAwIBAgIQDBdV+Atis7S9Wj+UhA36rTAKBggqhkJOPQDazApMQswCQYDVQQGEwJVZUeEMMAoGA1UEChMDSUJNMWwCgYDVQQDEwN0Y2EhbcNMNTYxMTIzNjYyYWhhcNMTCwJ+E5MTIzNjYyWjA9MQswCQYDVQQGEwJVZUeEMMAoGA1UEChMDSUJNM5AWhgYDVQQDEwUdcmFuc2FjZGlvbIDZDZjX0aWZpY2F0ZTBZMBMBGByqGSM49AgEGCCqGSM49AwEHA0IABLrxmY9TA2KWhSe0G7jwvLt7hEF55sTqYQZB0s5ifLcMmSlpqzqrzNgwaLeMgfrp3iOmMCMcAVc9y1fWU30eILeJgdwgdwkdGgYDVRPAQH/BAQDAgWCAAGa1UDeWEB/wQCMAAwDQYDVRO0BAYEBAECAwDwYDVRO7jBAGwBoAEAQIDBBNBBGyYqAwECBzBAf8EQr5QFNvEdqgHctksWMJ+u+AKG5rsWINAwJLnlVpYocrTxEhs7HHMSDnl+stv1+GKsVasBfmQY1PhStJLQ31a4wSgYGKMBEYqA1EFQZ373vpacz9r1XCfQy6nTLKEhclHqJDUGIR5jrhAk7jpoYItcQA2A6nF1RqKePunKgc57w4Vrkd10pz+rkqvFYMAAGCCqGSM49BAMD0cAMEQCIFIpIruyutTlbKXLYNzzfC8N4hxxOQvcABZDU926i8wYAiAJudsovUZketRmEzR/CwpAZqTj6f1pjZ/FG9Qa4V8Geg=="
"signature":
"MEQCIHkIzR2C8phQB2/sla/MVg9bZtHYZ/rUJJKFehJEZ7DWAiAQAnilNZ7QtGhOU7hnnNdcpkPPSWWnqgBzQPQklrdzulg=="
```

The “*signature*” element is the output of running a function over the transaction input data using the private key of the initiator of the transaction. As the private key is a secret known only to the transaction initiator, it has the effect of proving who initiated the transaction; it is a *digital signature*.

The “*cert*” element is the public transaction certificate of the transaction initiator.

It is possible to base64 decode the *cert* element (e.g. using [www.base64decode.org](http://www.base64decode.org)) to see the human-readable fields of the certificate. It is also possible to format the data in a form that can be read by the *OpenSSL* tools ([www.openssl.org](http://www.openssl.org)), as shown in the next step.

\_\_43. [OPTIONAL] To view the certificate details, create a file called test.pem that contains the certificate information bounded by -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----, and separate into 65-column lines as per the following screenshot:

```

-----BEGIN CERTIFICATE-----
MIICQDCCAAeegAwIBAgIQD8bDvAtiS75JmJw+UhhA36rTAKBgqqhkJOPQ0DAZApMQSw
CQYDVQQCAGeJVUZeEMMAoGA1UEChMDSUJNMWwwCgYDVQQGEwN0Y2EWHhcnMTYxMTIiX
Tl2INjYwYWhcnMTMTcwIjESMTZ1ZjYwYjASMQSwCgYDVQQDEwN0Y2EWHhcnMTYxMTIiX
SUJNMjAwHgYDVQ0EXdUcmFuc2FjdGJvb1B0ZXJ0aWZpY2F0ZFB0Z0Z0Z0Z0Z0Z0Z0Z0Z0
AGGCGCGSM49AaEHA0IABLrxmY9TA2KwhSe0G7jwvLT7HfE55sTqYQZB0S5ifLcm
15lpqzqrzNgwaleMgFrP3i0MmMcAwA9C5ylfWU30eIeJgdhwgdkgdW0DVR0PAQH/
BAQDAgeAMGAwA1UdEWB/wQCAAawDQYDVR0D0BAYEBAECAwQDwYDVR0jBAGwBoAE
AQIDBDBoBgYqAwFgBgBAf8EQP5R8QFmVedvGdHctksMMJ++AKG5rswJNnJVLnLv
PyocrTXehS7HHMSDL+stv1+GK5VasbFMQIY1PhStJLQ31a4wSgYKGMBG0YIBECZ
j3pavcz9rLXCfY6bnLTKEh1chQJGIR5rhaK7jpy0YtQaA62Een1LRJqkePunK
g5c7w4W4rkD0pz+rqkvfMAoGCCCGSM49BAMD00cAMEQCIFIpIruyutLTlbXLLyWzz
fC8N4hxz0QvcABzDU92618wYA1AjduosovUzketRmEzR/CwpAQTJ6f1pjZ/FG9Qa
fV8G6e=
-----END CERTIFICATE-----

```

Then in a shell window run the command:

```
openssl x509 -in test.pem -text
```

The output will show the certificate information:

## IBM Blockchain

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    0c:17:55:f8:0b:62:4b:b4:90:c2:3f:94:84:0d:fa:ad
  Signature Algorithm: ecdsa-with-SHA384
  Issuer: C=US, O=IBM, CN=tca
  Validity
    Not Before: Nov 21 12:36:42 2016 GMT
    Not After : Feb 19 12:36:42 2017 GMT
  Subject: C=US, O=IBM, CN=Transaction Certificate
  Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    EC Public Key:
      pub:
        04:ba:f1:99:8f:53:03:62:96:85:27:b4:1b:b8:f0:
        bc:b4:fb:84:41:79:e6:c4:d0:c9:06:41:d2:ce:62:
        7c:b7:0c:99:29:69:ab:3a:ab:cc:d8:30:68:b7:8c:
        81:fa:e9:de:23:8c:98:c0:b0:01:57:3d:ca:57:d6:
        53:7d:1e:20:b7
      ASN1 OID: prime256v1
  X509v3 extensions:
    X509v3 Key Usage: critical
      Digital Signature
    X509v3 Basic Constraints: critical
      CA:FALSE
    X509v3 Subject Key Identifier:
      01:02:03:04
    X509v3 Authority Key Identifier:
      keyid:01:02:03:04

    1.2.3.4.5.6.7: critical
      .y..M.....K.0....9...6rU.yU?*.5.....-._.*.Z..f@.5>..$.7..
    1.2.3.4.5.6.8:
      ..{.i...U.)...2...5...$b...h.....YIF..>....;.^.4.7..K.
  Signature Algorithm: ecdsa-with-SHA384
    30:44:02:20:52:29:22:bb:b2:ba:d4:e5:6c:a5:cb:c8:dc:f3:
    7c:2f:0d:e2:1c:73:39:0b:dc:00:1c:c3:53:dd:ba:8b:cc:18:
    02:20:09:b9:db:28:bd:46:64:7a:d4:66:13:34:7f:0b:0a:40:
    66:a4:c9:e9:fd:69:8d:9f:c5:1b:d4:1a:e1:5f:06:7a
```

More information on the signing and encryption methods used can be found here:

<https://github.com/hyperledger/fabric/blob/master/docs/protocol-spec.md>

## 3.2. Membership Services Concepts

Now we will use the Membership Services component of the blockchain. This is a built-in Certificate Authority that provides us with the public/private key pairs that we will use on the blockchain. These were the certificates that we saw in the previous section.

Encryption and signing on the blockchain is done using standard public/private key pairs. It is assumed that everyone on the network can view public keys, and that only a single authorised user holds the private key. A public/private key pair is distributed in the form of a certificate.

Data that is signed with a private key can be verified by anyone with the public key, and is used on blockchain like a real-world signature - to prove the origin of data or code, and/or to state agreement to something.

Data that is encrypted with a public key can only be decrypted by the holder of the private key. This is used on blockchain to ensure privacy. Data can be encrypted multiple times in such a way that any one of a set of users can decrypt it (for example, both beneficiaries of a transaction).

The certificate authority on the blockchain, also known as the membership services component, is able to provide the user with two types of certificate:

- Enrollment certificate: A public/private key pair that is used to provide the user with an identity on the network. It is typically long lasting, like a user-name.
- Transaction certificate: A public/private key pair that is used to sign a transaction. It is typically used only once, like a one-time-use throwaway credit card number. It is not possible for all users to infer the owner of a transaction certificate.

A given user might have one enrollment certificate but many hundreds of transaction certificates, depending on how many transactions they wish to commit to the blockchain.

It is important to understand why the blockchain does this. Why does the blockchain have these two types of certificate? Why does the user not simply use the enrollment certificate for the signing of all transactions?

The reason is that the transaction certificate gives us something called *unlinkable identity*.

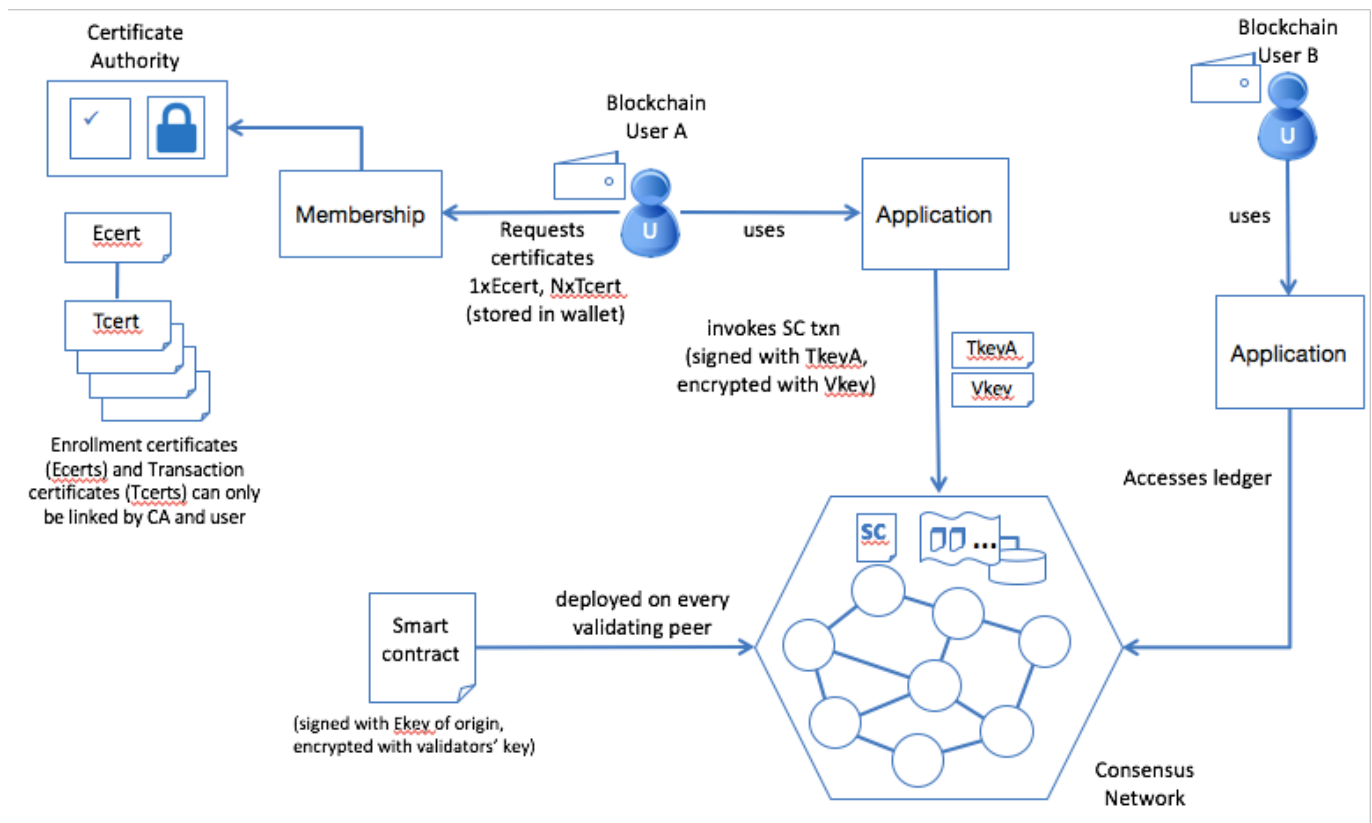
Let's assume Alice, Bob and Charlie form part of a business network and share a blockchain for transactions. Alice is dealing with Bob, and she is also dealing with Charlie. Crucially, Alice should never know that Bob and Charlie have done business together, as this knowledge could violate confidentiality between Bob and Charlie, and thereby give Alice a competitive advantage. For example, if every time Alice transacted with Bob, Alice knew that a transaction subsequently occurred between Bob and Charlie, she could assume that Bob is a middle-man and could avoid working with Bob in the future.

If we only ever used enrollment certificates for signing transactions on the blockchain, Alice could infer all the transactions involving either Bob or Charlie, because we know she has access to their public enrollment keys. By using transaction certificates on the other hand, no identity can be inferred because the certificates are used only once.

There is nothing stopping users from using transaction certificates multiple times, or indeed from using their enrollment certificates to sign transactions, but doing so increases the risk that their identity can be inferred.

Blockchains for business are generally regulated in some way. A regulator might have privileged access to the certificate authority, and thus be able to access the private keys of participants, and also be able to see which transaction certificates are owned by which users. For this reason, the certificate authority is a trusted environment, in much the same way that a passport office or drivers license authority is trusted. A compromised certificate authority will compromise all transactions that used certificates from it.

The security concepts of the blockchain can be summarised in the following diagram. A smart contract is simply a piece of user code deployed to the blockchain, and a transaction is defined as a set of input parameters to that code.



We will now use the membership services component to enroll on the blockchain network, provide us our enrollment certificate and issue some new sample transaction certificates.

### 3.3. Using Membership Services

\_\_44. In the blockchain dashboard 'APIs' tab, click "Network's Enroll IDs".

This reveals the available user IDs and passwords (secrets). This is a convenience for the purposes of this demo, and clearly would not be available in a real-world blockchain.

\_\_45. Click the secret next to the 'admin' user to copy it to the clipboard.



### Network's Enroll IDs

(click ID to copy to clipboard) note that an ID can only be registered against 1 peer

ID	Secret
admin	e9f03d792a
WebAppAdmin	22a3a250i4
user_type1_0	5500e0b74c
user_type1_1	edbcac4ee0

- \_\_46. In the Registrar section, click the POST /registrar operation.
- \_\_47. Fill in the JSON structure for the 'Secret' parameter, as follows. The enrollSecret parameter should match the secret you just copied to the clipboard.

#### Parameters

Parameter	Value
Secret	<pre>{   "enrollId": "admin",   "enrollSecret": "e9f03d792a" }</pre>

Parameter content type:

- \_\_48. Click 'Try it out!'.

You should get an OK response back from the certificate authority.

#### Response Body

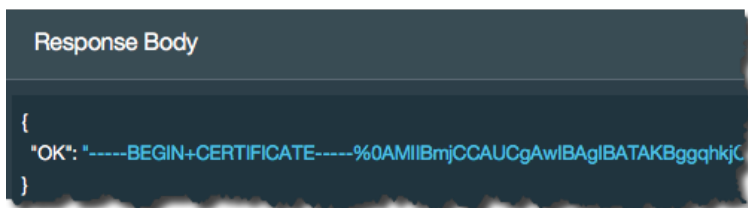
```
{
  "OK": "Login successful for user 'admin'."
}
```

- \_\_49. Select the 'GET /registrar/{enrollmentID}/ecert' operation.
- \_\_50. Enter 'admin' as the enrollmentID.



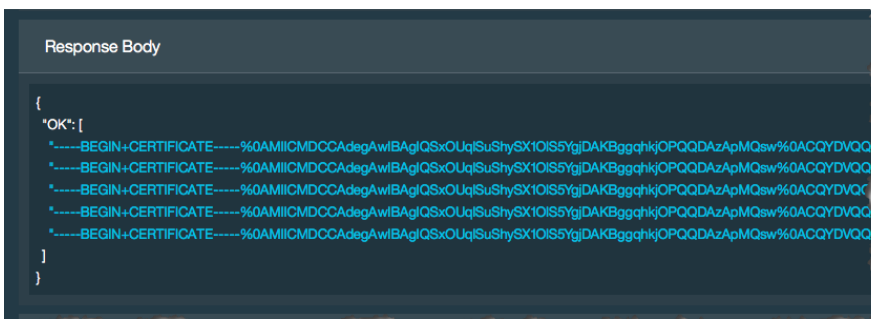
\_\_51. Click 'Try it out!'.

The enrollment certificate for the user will be returned.



\_\_52. Do the same for the 'GET /registrar/{enrollmentID}/tcert' operation. Use 'admin' as the enrollmentID and '5' as the count.

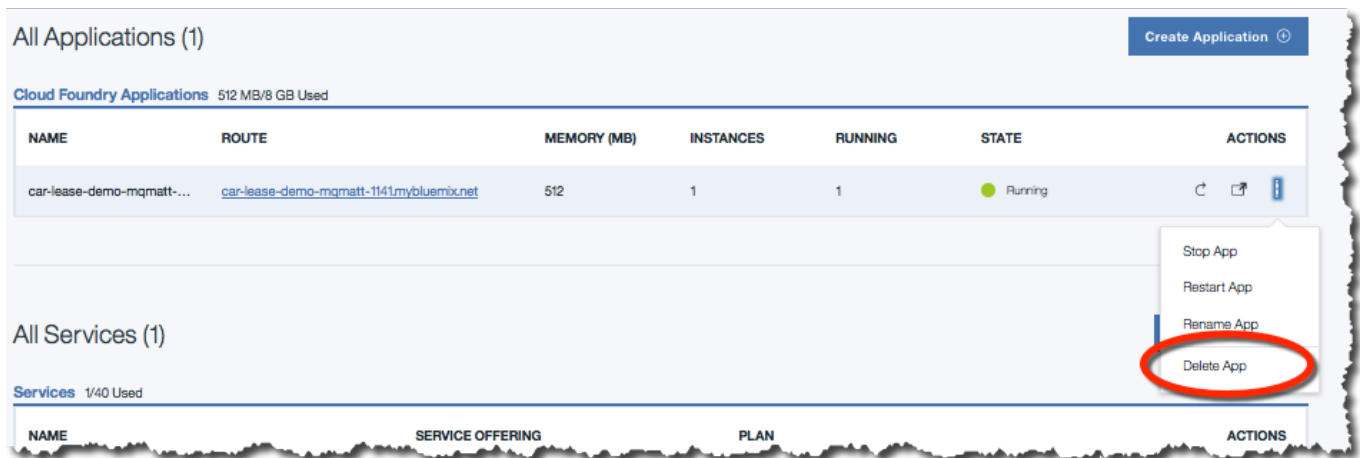
You will receive a set of five transaction certificates. As the developer of a blockchain application, you could then use these certificates for signing new transactions.



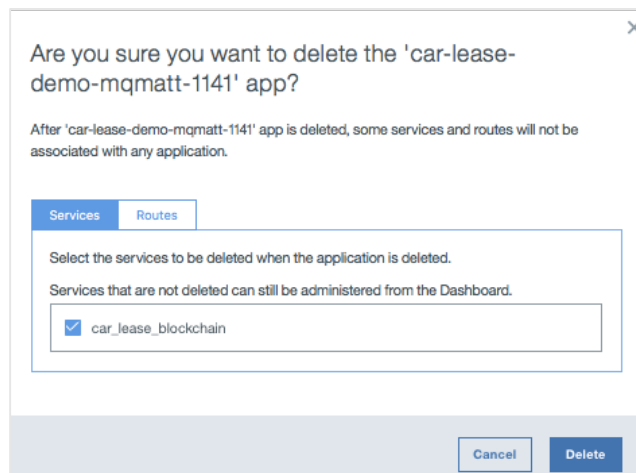
## Section 4. Removing the sample application

The final section of this lab aims to stop and remove the Blockchain service you created.

- \_\_53. Return to the Bluemix Dashboard (<https://console.ng.bluemix.net/dashboard/applications>).
- \_\_54. Click the three vertical dots at the right of the Car Leasing application Settings icon in the car lease demo application and select '**Delete App**' from the menu.



- \_\_55. Ensure that the 'car\_lease\_blockchain' service is selected for deletion and click '**Delete**'.



- \_\_56. Wait for the items to be stopped and deleted. Once this is done, both the application and the associated service will no longer be visible in the Bluemix dashboard.

**Congratulations on completing Lab two – “Blockchain Explored”!**

---

## Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries.

Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation

Licensing

2-31 Roppongi 3-chome, Minato-ku

Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 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.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

## Appendix B. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

## NOTES

## NOTES

[illegible]







---

© Copyright IBM Corporation 2016.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and [ibm.com](http://ibm.com) are trademarks of International Business Machines Corp., 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](http://www.ibm.com/legal/copytrade.shtml).



Please Recycle

---