

Oracle Content & Experience Cloud Service

Workshop

Contents:

[Demo Attributes 2](#_Toc506979765)

[Oracle Blockchain Cloud Service 2](#_Toc506979766)

[Overview 3](#_Toc506979767)

[Chapter 1: Launch BCS Provisioning Console 4](#_Toc506979768)

[Chapter 2: Creating a Blockchain Network – Founder 8](#_Toc506979769)

[Chapter 3: Joining Founder and Particpants 10](#_Toc506979770)

[Chapter 4: Creating Channels 17](#_Toc506979771)

[Chapter 5: Review your Blockchain Network 20](#_Toc506979772)

[Chapter 6: Installing Chaincode on your network 21](#_Toc506979773)

[Chapter 7: Configure the REST Gateway 26](#_Toc506979774)

[Chapter 8: Initialize the ledger 28](#_Toc506979775)

## Demo Attributes

|  |  |
| --- | --- |
| **Product(s)** | Oracle BlockChain Cloud Service |
| **Date last updated** | March 2018 |
| **Author(s)** | Jens Lusebrink |
| **Demo Title(s)** | PaaS – BlockChain Cloud Service (BCS) |

## Oracle Blockchain Cloud Service

Oracle Blockchain Cloud Service is a new offering that is part of Oracle’s comprehensive platform-as-a service (PaaS) portfolio. Delivered by the world’s most scalable, distributed transaction processing platform provider, Oracle Blockchain Cloud Service is the most comprehensive distributed ledger cloud platform to provision Blockchain networks, join other organizations, and deploy & run smart contracts to update and query the ledger. Reliably share data and conduct trusted transactions with suppliers, banks, and other trade partners through integration with existing or new cloud-based or on-premises applications.

## Overview

**End-to-End Application Flow**

This HandsOn Lab (HOL) showcases the end to end flow of setting up a Blockchain network and simulate the orchestration between 3 parties that participate in a Blockchain network based on Hyperledger Fabric. The HOL consists of the following chapter:

* Setup the Blockchain network
* Adding participants
* Creating channels between participants
* Establishing and instantiating the chaincode
* Execute transactions on the Blockchain

| **S.No.** | **Action** | **Description** |
| --- | --- | --- |
| Chapter 1: Launch BCS Provisioning Console | | |
| 1.00 |  | To get started, we are launching the Oracle Blockchain Cloud Service.  ***NOTE****: this process uses a utility which has been supplied with the VM for simplicity, provisioning is different in Blockchain Cloud Service.* |
| 1.02 | This is Desktop for the BCS VM.  **NOTE:** Depending on your class setup you can skip this step  **Right-mouse** click on the Desktop and open a **Terminal** window | FOLLOW THE INFORMATION IN YOUR **BCS\_VM\_Setup** DOCUMENT |

|  |  |  |
| --- | --- | --- |
| 1.03 | In the Terminal window, enter the following commands and hit **Enter**:  *bash ~/oracle\_fabric/startProvosion.sh*  **NOTE**: all in one line |  |
| 1.04 | The Oracle Blockchain Cloud Service will start.  **NOTE:** You might get prompted twice for the **root** password. Enter the password provided by your instructor. |  |
| 1.05 | Finally, the Oracle Blockchain Cloud Service has started and we can proceed to the Administration Console |  |
| 1.06 | Open **Chrome** from the Desktop icon.  Enter “**localhost:3000**” in the URL bar.  Enter the **Admin credentials** for this workshop:  **Username:** *username*  **Password:** *password*  And click “Sign In” |  |
| 1.07 | The Oracle Blockchain Cloud Service Administration Console has started and we can create our Blockchain network. |  |
|  |  |  |
| **Congratulation**  **You completed Chapter 1** | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chapter 2: Creating a Blockchain Network – Founder | | | | |
| 2.00 |  | To get started, we are going to provision the **Blockchain network Founder**. | | |
| 2.01 | This will be the founding organization that controls the Blockchain network.  Select the “Create Blockchain Network Radio” button.  Then configure the founder network as per the below screenshot.  ***NOTE:*** *Please have a look at your course instruction with the correct values for your Blockchain organization.*  *The screenshots Is just showing sample data*  Click ‘Apply’ to begin creating the network and wait for the process to complete. Ignore the warning about using an IP address. You may need to refresh the page for the new Organization to show up in the Organization list on the left hand side of the page. |  | | |
| 2.02 | Creating the Blockchain network participants.  For participants, you select the “Create Block Chain Organization” radio button.  Create two dealer organizations configured as in the table on the right, clicking **Apply** after each to create the underlying nodes. | Name:  Domain:  Number of Peers:  Host Address:  REST Proxy Port:  Console Port: | **Dealer A**  SamDealer  sam.com  2  x.x.x.x  3110  3210 | **Dealer B**  JudeDealer  jude.com  2  x.x.x.x  3120  3220 |
| 2.03 | The post setup display will look like this:  ***Note:*** *All networks are activated* ***(ON)*** *after creation.*  You can start and stop networks from this screen by right clicking the row in the table and pressing ‘Start’ and ‘Stop’. |  | | |
| **Congratulation**  **You completed Chapter 2** | | | | |

|  |  |  |
| --- | --- | --- |
| Chapter 3: Joining Founder and Particpants | | |
| 3.00 |  | Once the networks have been created, we need to connect them to each other so that they can participate on the Blockchain network. This will be done on each founder/particpants management console.  You can access the admin console for each of them on the ports specified during provisioning. If you used the above configuration, this means the console for the founder, Detroit Auto, is available at <http://localhost:3200>, for SamDealer at <http://localhost:3210> and for JudeDealer at <http://localhost:3220>.  Alternatively, you can click on the founder/participants name. |
| 3.01 | Import Orderer settings.   1. In the console for “**DetroitAuto**” founder organization, select the “Network” tab. Then export orderer setting by clicking the hamburger menu on the “DetroitAuto” organization. Save the exported file. 2. In the console of each of the participant organizations, select the “Network” tab. Then import the founder’s orderer settings by selecting the “Import Orderer Settings” button. 3. Click on “Upload Orderer Settings” and navigate to where you saved the exported file in step 2. Select this as the import file. **Repeat this on the other participant** |  |
| 3.02 | The founder needs two pieces of information from the participant organizations, **certificates** to be able to validate communication, and **information about the peer nodes** maintained by the organization.  Open the participant organization’s console.  Click the organization name in the header, and then click ‘Export Certificates’  Save the certificates locally. Since these files have the same name for each organization, it may be helpful to create separate directories for each of your participant organizations to keep things separate. |  |
| 3.03 | Export the node information of the peers by navigating the Nodes tab, then click the “Export/Import” button to export the node information. Select **both of the participant’s peers**, then click “Export” and save the resulting file.  Repeat this process for the other participant. |  |
| 3.04 | Oracle Blockchain Cloud Service is a ‘permissioned’ network, in that each member organization is known and has been granted access, and the process which we are currently undertaking is what enables that.  To add the two dealers to our network, **return to the founder organization’s console**, then from the “Network” tab, click on “Add Organizations”.  You can import the certificates in bulk by clicking the ‘plus’ button and uploading both sets of certificates before clicking “Add All”.  The founder also needs to know about the peer nodes maintained by the participant organization. The node information which was previously imported can also be imported in bulk, this time from the “Nodes” tab, via the “Export/Import” button and “Import” option. Upload both sets of remote node configuration that you exported earlier.  If this process has been completed correctly, the peer nodes for the other organizations should appear in the Nodes list, associated with the correct organization in the MSP ID column (which stands for Membership Service Provider, which is responsible for keeping track of the organizations who are involved in the network). |  |

|  |
| --- |
| **Congratulation**  **You completed Chapter 3** |

|  |  |  |
| --- | --- | --- |
| Chapter 4: Creating Channels | | |
| 4.01 | The final step in setting up our multi-party blockchain network is to create channels for each of the dealers, in order to segregate transactions from each dealer from the transactions of other dealers.  **In our scenario, we want a separate channel for each of our two dealers.**  To create these, navigate to the “Channels” tab in the **founder organization’s console**. Then click “Create Channel”.  Create a channel for each dealer named ‘<dealer>.channel’, i.e. ‘sam.channel’ and ‘jude.channel’ if you have been following these notes. For each channel, ensure that the relevant dealer is included as a ReaderWriter and both of Detroit Auto’s peers are joined to the channel.  Create the above for each of the dealers. |  |
| 4.02 | These channels, at present, only contain Detroit Auto’s peer nodes. The dealers cannot create transactions on these channels, nor view transactions on these channels, as they do not have nodes which participate in them, though as their Organizations were added at channel creation, they are allowed to add their own peers to the channel.  In order to join the created channel, **access the participant organization’s console**. From there navigate to the “Nodes” tab and click on the **name of the peer to be added to the channel**. From the side bar, you are able to review logs and diagnostic information for the peer, chaincode which has been installed on the peer, and for our current usecase, the channels that the peer has joined.  Navigate to the “Joined Channels” tab, then click “Join Channel”. A dialog box is displayed, asking for the name of the channel. For each of the peers, enter the channel name (‘sam.channel’ or ‘jude.channel’) as appropriate, then click ‘Join’.  Repeat for each of the peers, and each of the dealers. |  |
| **Congratulation**  **You completed Chapter 4** | | |
| Chapter 5: Review your Blockchain Network | | |
| 5.00 | At this point we have built out a pretty good network. This can be seen by reviewing the structure of the network, which can be done by navigating to the “Nodes” tab on the founder. Ensure that the dauto nodes are in the list, as well as the supplier's peer nodes. Then click the topology view button, which is outlined in red in the below image.  As you can see, this resembles the original plan we had for the network when discussing the original scenario (though with the addition of detroitautoorderer as an internal channel). If your topology doesn't look like the one shown here, or peers are not present in the list, check what is missing, then import/export those components, or check the configuration of your channels. |  |

|  |
| --- |
| **Congratulation**  **You completed Chapter 5** |

|  |  |  |
| --- | --- | --- |
| Chapter 6: Installing Chaincode on your network | | |
| 6.00 |  | While we have built a pretty good-looking network, it doesn't actually do anything yet. This is where chaincode comes in.  Chaincode is the code that runs your Blockchain network, defining what types of data are written during transactions, providing logic around how transactions are executed and validating the conditions under which they should be run.  Chaincode initializes and manages ledger state through transactions submitted by applications. A chaincode typically handles business logic agreed to by members of the network, so it is also sometimes referred to as a “smart contract”.  Chaincode is written in golang, also known as GO, but don't worry too much if you are not familiar with golang, as the chaincode for this supplier network has been provided for you in ~/Documents/carTrace.zip |
| 6.01 | To install the chaincode, navigate to the “Chaincodes” tab in the founder organisation’s console, then click the “Install Chaincode” button. This allows you to upload chaincode from your local machine, or reference a remote URL, such as the output of an automated build process. For now, we will use the pre-prepared chaincode located in ~/Documents/carTrace.zip  Upload the carTrace,zip file and fill in the chaincode details as above, ensuring all of the available peers are included in the “Target Peers” selection.  ***Note:*** *Take care with the Chaincode Name, as this needs to be consistent across all of the Organisations on which the chaincode is installed.* |  |
| 6.02 | Once the chaincode has been installed on the peers, it needs to be ‘instantiated’ to the channels on which it will run. To do this, **click on the name** of the newly installed carTrace chaincode to view its details. Then **from the hamburger menu** on the latest version (probably v1), click “Instantiate”  Engage the context menu as shown to bring up the chaincode instantiation screen.  You will need to   * Select the channel you are instantiating the chaincode for * Select the local peers that will participate in the channel * The initial parameters should be left as default, as this particular chaincode doesn’t require or use them, the default endorsement policy should be adequate, and there is no need for a Transient Map.   Then click ‘Instantiate’. This process usually takes some time.  After this is done, **repeat the same process for your second channel.** |  |
| 6.03 | In **each of the participant’s consoles**, navigate to the “Chaincodes” tab. Then repeat the process of installing the chaincode that was used on the founder.  Click on “Install Chaincode” and then upload the same chaincode zip file.  Ensure the name of the chaincode is identical to the name provided on the founder organization, and that each of this organization’s peers are included in the “Target Peers” selection.  Once the chaincode has been installed, there is no need to instantiate it, as instantiation is channel-specific, not organization or peer specific. You can check this by expanding the chaincode entry via clicking the small arrow next to the chaincode name, and observing that it already has an instantiated channel. |  |
| **Congratulation**  **You completed Chapter 6** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 7: Updating Peer information | | |
| 7.00 | In order to expose our chaincode for consumption, we need to inform the founder organization that this chaincode is now available on our participant’s peers. To do this, we will export the node information once more, by clicking the Export/Import button from the Nodes tab, and exporting our participant’s peers. Once the node information has been exported, it can be imported into the founder from its “Nodes” tab, via the “Export/Import” button and “Import” option.  ***Note****: In these instructions, we perform this export twice, initially after joining channels, then again here. The first export/import is not required, and was included just to allow for validation of the network topology.*  After importing the participant node information into the founder, you will need to export the founder’s node information into each of the participants, following the instructions above. | Proceed as you did in chapter 3.03 |
| **Congratulation**  **You completed Chapter 7** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 8: Configure the REST Gateway | | |
| 8.00 | To make our chaincode available externally, we need to enable access to it via the REST Proxy. This enables simple access to chaincode functions via any HTTP client.  ***Note****: For simplicity, the gateway that is set up by the virtual machine doesn’t have any authentication or authorization enforcement, as it has no identity provider. In a cloud instance of BCS, this is managed by Identity Cloud Service.*  In order to expose our chaincode through the REST Proxy, navigate to the Nodes tab and locate the gateway node (gateway0.dauto.com or similar). Using the context menu click on “Edit”. Configure each channel, chaincode and peers as below, then click Submit. **You will need to repeat these steps for each organization.**  Once done, the gateway will be marked as configured, but the changes will only be reflected after restarting the Gateway. You can do this now using the ../../../Desktop/Screen%20Shot%202018-01-06%20at%2012.46.25%20AM. button. |  |
| **Congratulation**  **You completed Chapter 8** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 8: Initialize the ledger | | |
| 9.00 | Now the chaincode has been uploaded and instantiated, and can be invoked remotely via the REST gateway, we are able to invoke chaincode functions and query the ledger. The ledger is completely empty at the moment, so in order to have some meaningful data to inspect, a script has been provided which adds some sample data to the ledger.  The script creates a number of vehicle parts and vehicles, then transfers them from Detroit Auto to each of the dealers, to establish some initial data to query. | You can find the script in the /Documents folder of your OBCS instance. Open a terminal window and execute the following command: bash ~/Documents/initLedger.sh      ***NOTE: If you made any changes to the naming of channels, chaincode, or port numbers please edit the script before running it***. |
|  | | |
| **Congratulation**  **You completed Chapter 9** | | |