

Oracle Content & Experience Cloud Service

Workshop

Contents:

[Demo Attributes 2](#_Toc524422156)

[Oracle Blockchain Cloud Service 2](#_Toc524422157)

[Overview 3](#_Toc524422158)

[Chapter 1: Launch BCS Provisioning Console 4](#_Toc524422159)

[Chapter 2: Creating a Blockchain Network – Founder 8](#_Toc524422160)

[Chapter 3: Joining Founder and Particpants 10](#_Toc524422161)

[Chapter 4: Creating Channels 14](#_Toc524422162)

[Chapter 5: Inform the Founder about the participant Nodes 16](#_Toc524422163)

[Chapter 6: Review the network 19](#_Toc524422164)

[Chapter 7: Installing Chaincode on your network 20](#_Toc524422165)

[Chapter 8: Re-Export Peer Information 27](#_Toc524422166)

[Chapter 9: Configure the REST Gateway 28](#_Toc524422167)

[Chapter 9: Initialize the ledger 29](#_Toc524422168)

## Demo Attributes

|  |  |
| --- | --- |
| **Product(s)** | Oracle BlockChain Cloud Service |
| **Date last updated** | July 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 the 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”  **NOTE: If you are using this username / password combination for the 1st time you will see the following dialog.**  **Click ‘OK’ to create a new account.** |  |
| 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**. This will be the founding organization that controls the Blockchain network. | | | |
| 2.01 | 1. Select the “Create Blockchain Network Radio” button. 2. 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 screenshot Is just showing sample data. Don’t use the IP Address displayed in the screenshot.**  DESELECT ‘EnableTLS’ as we are not going to use this in the VM.   1. 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:**  **Enable TLS**  REST Proxy Port:  Console Port: | **Dealer A**  SamDealer  sam.com  2  **x.x.x.x**  **UNCHECK**  3110  3210 | **Dealer B**  JudeDealer  jude.com  2  **x.x.x.x**  **UNCHECK**  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’.  You also can clean and remove a participant if e.g. you have accidentally entered a wrong value. |  | | |
| **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/participants admin 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. 2. Export orderer setting by clicking the hamburger menu on the “DetroitAuto” organization. 3. Save the exported file. 4. 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. 5. Click on “Upload Orderer Settings” and navigate to where you saved the exported file in step 3. Select this as the import file. 6. **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.   1. Open the participant organization’s console. 2. From the Network tab, you can export the certificates by clicking the hamburger menu and selecting ‘Export Certificates’ on the dealer’s organization. 3. Save the exported file. 4. **Repeat this process for the other participant**. |  |
| 3.03 | Now we have enough information to inform the founder of the other organisations which intend to use the network. Normally this process wouldn’t be performed by a single user, and would instead involve the transfer of this metadata through some out of band mechanism in order onboard a new organization to the Blockchain network.  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. | |
| 3.04 | 1. To add the two dealers to our network, return to the founder organization’s console, then from the “Network” tab, 2. Click on “Add Organizations”. 3. You can import the certificates in bulk, by clicking the ‘plus’ button and uploading both sets of certificates before 4. clicking “Add”, then 5. “Finish” once the organisations have been successfully added. |  |

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

|  |  |  |
| --- | --- | --- |
| Chapter 4: Creating Channels | | |
| 4.00 | 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.** | |
| 4.01 | 1. Navigate to the “Channels” tab in the **founder organization’s console**. 2. Then click “Create Channel”. 3. Create a channel for each dealer named ‘<dealer>channel’, i.e. ‘samchannel’ and ‘judechannel’ if you have been following these notes. 4. 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 | 1. In order to join the created channel, **access the participant organization’s console**. 2. Navigate to the “Nodes” tab and 3. Click on the hamburger menu for each peer node, then select “Join New Channels”. 4. A dialog box is displayed, asking for the name of the channel. 5. For each of the peers, enter the channel name (‘samchannel’ or ‘judechannel’) as appropriate, then 6. click ‘Join’   **Repeat for each of the peers, and each of the dealers.** | 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. |
| **Congratulation**  **You completed Chapter 4** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 5: Inform the Founder about the participant Nodes | | |
| 5.00 | In order to allow the founder to include the other organizations as endorsers of transactions, they need to know about the other peer nodes. In order to do this, we export the node information, and import it on the founder. | |
| 5.01 | 1. In the participant organizations, navigate to the “Nodes” tab, 2. Click the “Export/Import” button to export the node information. 3. Select both of the participant’s peers, 4. Click “Export” and save the resulting file.   **Repeat this process for the other participant.**   1. 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. 2. Upload both sets of remote node configuration that you exported earlier. i.e. | 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 5** |

|  |  |  |
| --- | --- | --- |
| Chapter 6: Review the network | | |
| 6.00 | 1. Navigate to the “Nodes” tab on the founder. 2. Ensure that the dauto nodes are in the list, as well as the supplier's peer nodes. 3. Then click the topology view button, which is outlined in red in the below image. | At this point we have built out a pretty good network, and it is pretty good looking too. This can be seen by reviewing the structure of the network    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. |

|  |  |  |
| --- | --- | --- |
| Chapter 7: Installing Chaincode on your network | | |
| 7.00 | 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 | |
| 7.01 | 1. Navigate to the “Chaincodes” tab in the founder organisation’s console, 2. Click the “Deploy a New 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 3. Enter the chaincode name ‘carTrace’ 4. Version 1 ‘v1’ 5. Select peers to deploy to 6. Upload the carTrace,zip file. 7. Click ‘Next’. You should receive a success message.   **Note: Take care with the Chaincode Name, as this needs to be consistent across all of the Organisations on which the chaincode is installed.**  In the next step you will be prompted to ‘instantiate’ the chaincode, which involves pushing it to the channels in which it will run.  **At this step, you will need to**  •  Select the channel you are instantiating the chaincode for;  •  Select the local peers that will participate in the channel;  •  While the default endorsement policy is probably adequate, we can specify here that we want to ensure transactions include both of our organizations by adding their identities to the policy then selecting that the transactions need to be signed by 2 of 2 Organizations;  •  The initial parameters should be left as default, as this particular chaincode doesn’t require or use them, and there is no need for a Transient Map  Click ’Next’.  This might take some time.  The last step is to expose the chaincode through the REST proxy. | There are two options for installing chaincode, either ‘Quick Deploy’ which is designed for one- click setup of chaincode and includes a number of defaults, or Advanced, which allows for more fine-grained control over the chaincode setup process.  For this exercise, we will use the **Advanced option**, which allows us to step through the process and explain the meaning of each setting, however for most development scenarios, Quick Deploy is perfect.      The completed configuration will look like the following: |
| 7.02 | In order to expose the chaincode, select a REST proxy (one was created by default at network setup, gateway0.dauto, and select the peers where the chaincode was deployed (i.e. both of them): |  |
| 7.03 | The chaincode is now setup and running on samchannel on the founder, Detroit Auto. In order to also have this chaincode running on our second channel, judechannel, we will need to manually instantiate it. | |
| 7.04 | 1. Expand the installed **carTrace** chaincode 2. Click the Hamburger menu for the installed version (v1) 3. And select ‘Instantiate’ 4. Select the 2nd channel 5. Add both peers 6. And configure the endorsement policy 7. Click ‘Instantiate’. This might again take some time. |  |
| 7.05 | Now we need to install the chaincode on the participants. We only need to install the chaincode on the participant’s peers, since the chaincode was already instantiated on the channels by the founder. To do this, we will once more use the ‘Advanced’ mode of chaincode deployment. | |
| 7.06 | 1. In each of the participant’s consoles, navigate to the “Chaincodes” tab. 2. Click on ‘Deploy Chaincode’ 3. Select ‘Advanced’ 4. In the “Install a new Chaincode on peers’ step, upload the same chaincode zip file. Ensure the name and version 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. 5. Click ‘Next’.   Once the chaincode has been installed, there is no need to instantiate it, as instantiation is channel-specific, not organization or peer specific.   1. Cancel out of the Instantiate Chaincode step, by pressing the “Close” button.   You can check that this has occurred 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 7** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 8: Re-Export Peer Information | | |
| 8.00 | 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. | |
| 8.01 | 1. Go to the ‚Node‘ section of the participants Screen 2. Click the ‚Export/Import‘ button 3. Export the participants peers 4. Save the file   Repeat this for the other participant. |  |
| 8.02 | 1. Go to the founder‘s ‚Node‘ tab 2. Select ‚Import‘ and select the 2 participants files 3. Click ‚Import‘. |  |
| 8.03 | 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. | |
| **Congratulation**  **You completed Chapter 8** | | |
| Chapter 9: Configure the REST Gateway | | |
| 9.00 | While the chaincode deployment process made the chaincode available externally via the REST proxy, we specified that transactions require endorsement by multiple organisations, which are not configured by default in the REST proxy. In order to transactions to be appropriately endorsed, we need to update the proxy configuration to include additional peer nodes. | |
| 9.01 | 1. In the founder’s dashboard, navigate to the ‘Nodes’ tab. 2. Locate the ‘Gateway’ node. 3. Using the hamburger menu click on “Edit Configuration”. 4. Add SamDealer’s peers to the samchannel configuration, and then click the plus button to expose the chaincode running on judechannel, such that the final configuration looks like the below, then click Submit.   **You will need to repeat these steps for each organization**. |  |
|  |  |  |
| **Congratulation**  **You completed Chapter 8** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 9: 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. | |
| 9.01 | 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** | | |