Contents

Demo Attributes

Date last updated	October 18, 2018
Author(s)	Axel Bronder, Pape-Lamine Cisse
Demo Title(s)	HOL4973 - DApps, Chaincode, Smart Contracts: Get Decentralized, or We'll Go Without You!

Abstract

Get your skills going on decentralized application development before everybody else! This session will jump-start your career in the decentralized field—making you ready for the future and an even more requested resource for companies all over the world. Blockchain is on everybody's lips, and in this hands-on lab, you will have the opportunity to get your hands dirty developing Chaincode directly in the cloud, using a fun and engaging forest-saving use case. Later on, *your* code might be implemented to help save the environment!

ORACLE

Overview

End-to-End Application Flow

This Hands-On Lab showcases the end-to-end flow of working as a developer with an existing Hyperledger Fabric Network, hosted in Oracle Blockchain Cloud Service (OBCS).

- Install & instantiate existing Chaincode on the Blockchain
- Invoke a method on the Chaincode using REST API (with PostMan)
- Query the Ledger using REST API (with PostMan)
- Modify & upgrade the Chaincode

Use Case

Current Situation - A Hyperledger Fabric Network of 3 Parties

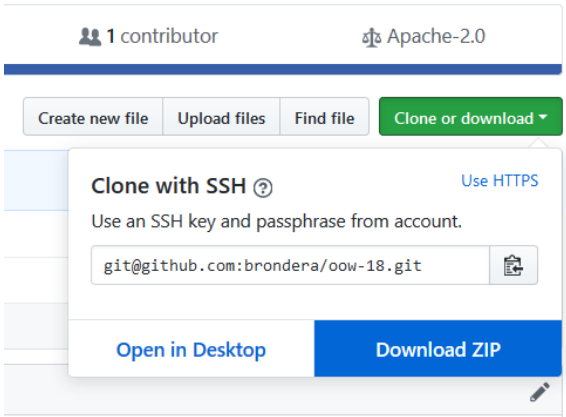
1. **Claudio's Controlling (Founder) - Owner of the Trusted Trees brand**
   The Trusted Tree branding guarantees the lumber origin as well as the worker terms and it also guarantees that more trees are planted than cut. This brand attracts a lot of investors.

2. **Franco's Forestry (Participant)**
   Plant & grow trees on site in Senegal under the Trusted Tree umbrella.

3. **Leopold's Lumber (Participant)**
   Cut trees into lumber under the Trusted Trees brand.

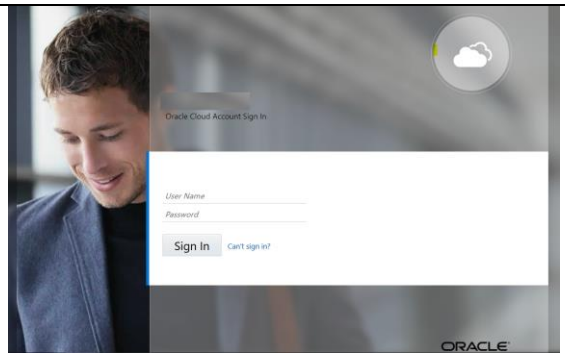
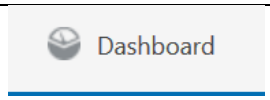
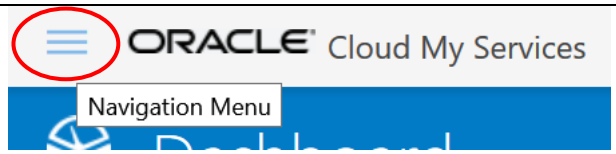
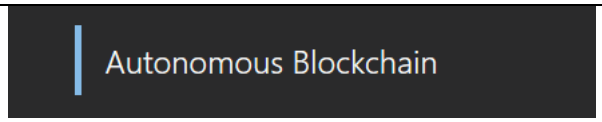
Recent Development

A second tree planting participant, **Pape's Plantation**, wants to join the network. It is our job to spin up their chaincode and adapt the code to this participant.

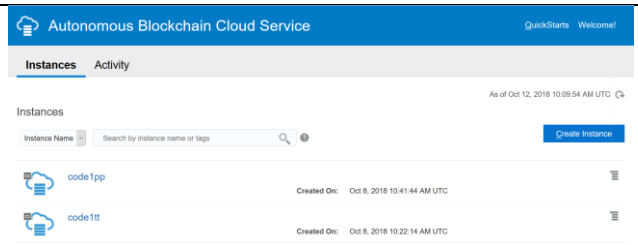
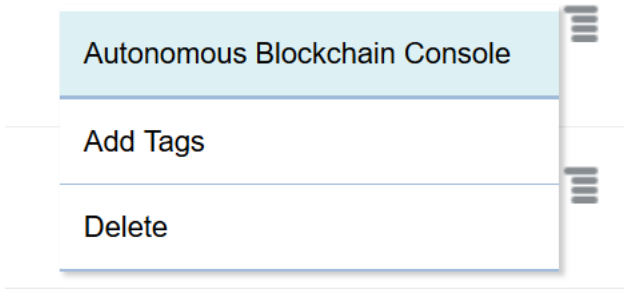
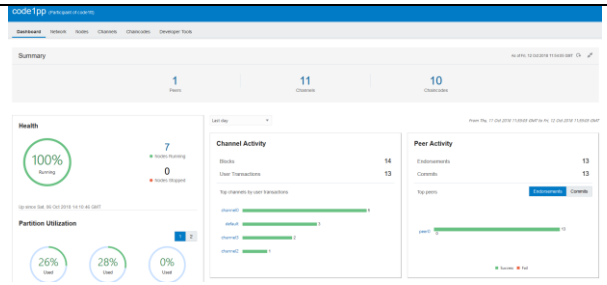
S.No	Action	Description
Chapter 0: Resources		
Welcome to the resources section, this part will make sure you have everything you need to start the lab.		
0.1	Oracle Blockchain Cloud Service	You will be running this lab using OBCS. **Details regarding cloud environment address, username and password will be handed out separately.** **Please consult your lab coach.** Lab login credentials are usually valid for one day only.
0.2	GitHub repository:	https://github.com/brondera/oow-18
0.3	In GitHub: - Select “Clone or download” - Select “Download ZIP” - Save the .zip file for later use The .zip file contains Chaincode in .go format	
0.4	If you haven’t done so already, download & install the latest stable version of Postman suitable for your environment. https://www.getpostman.com/	Postman (https://www.getpostman.com/) is a free API Development environment, for individuals and small projects. to query the Blockchain using the REST API. Postman is available for Mac, Windows and Linux.

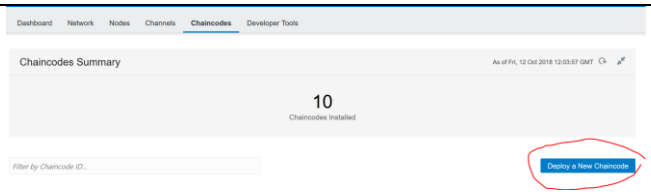
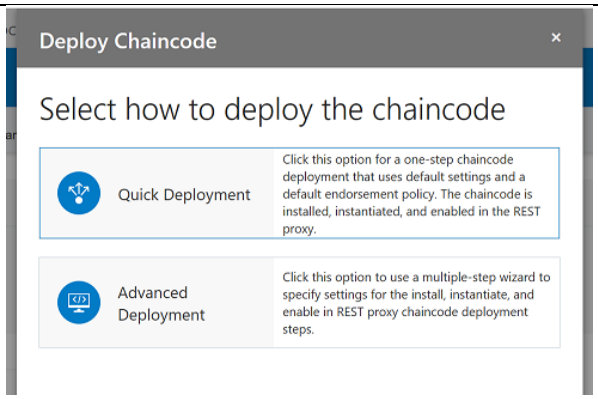
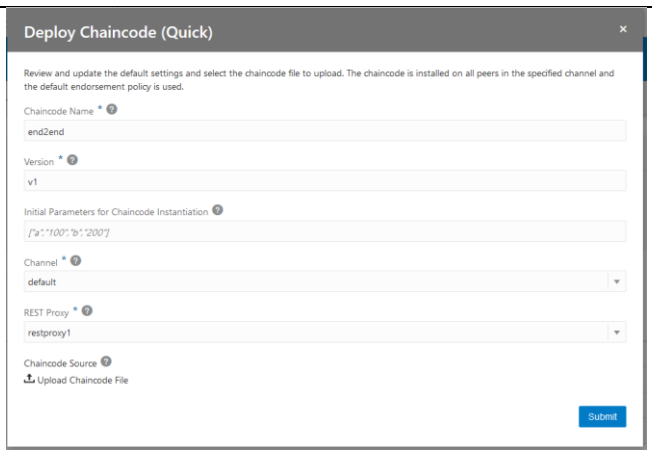
ORACLE

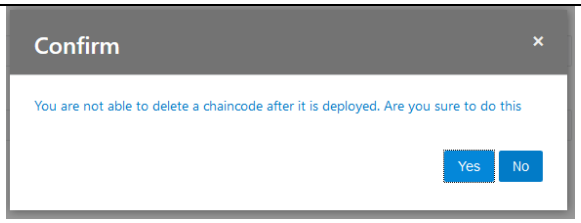
0.5	Start Postman desktop app.	If you haven't done so already, you will be asked to create a Postman account upon your first login. *(An alternative to Postman is Open Source tool cUrl. Although cUrl documentation for this specific lab is not yet available – one can make use of Oracles official, general, documentation if necessary: https://bit.ly/2IZ958e)*
Great Work! You are now ready to get started! **PS. Don't forget to discuss your work with your lab partner!**		

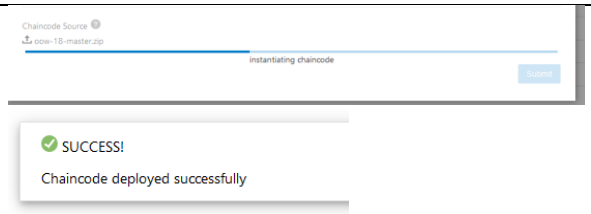
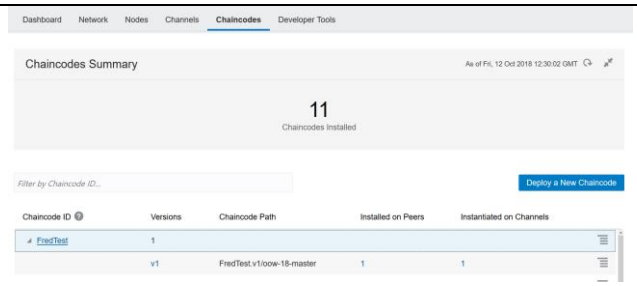
S.No	Action	Description
Chapter 1: Installing & Instantiating Chaincode		
Time to get started! First, let's try spinning up the Chaincode we found on GitHub in our cloud environment!		
1.00	Go to the login page of our cloud environment. **You should find the link as well as your login credentials to your temporary lab account in the handout of the lab** (separate handout).	
1.01	After a successful login, you are in the cloud environment of your lab account. Navigate to your Dashboard	  Locate the buttons above, anywhere in the cloud environment interface, and click on them to navigate to the Dashboard for your lab account.
1.02	From the dashboard page, click on the hamburger menu in the upper left corner of the screen to open a list of services available for your lab user.	
1.03	In the left panel that opens, click on "Services" and scroll down to Autonomous	

ORACLE

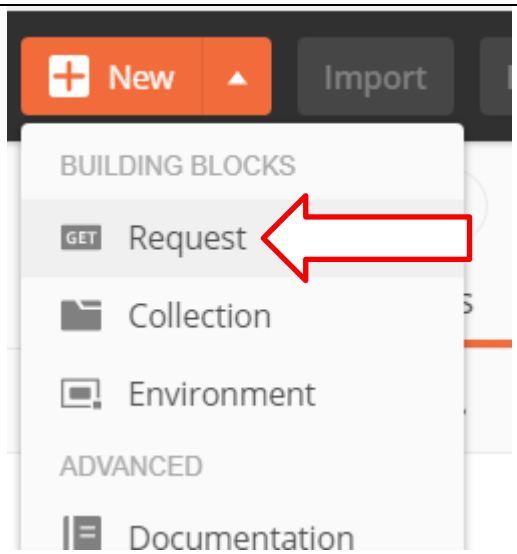
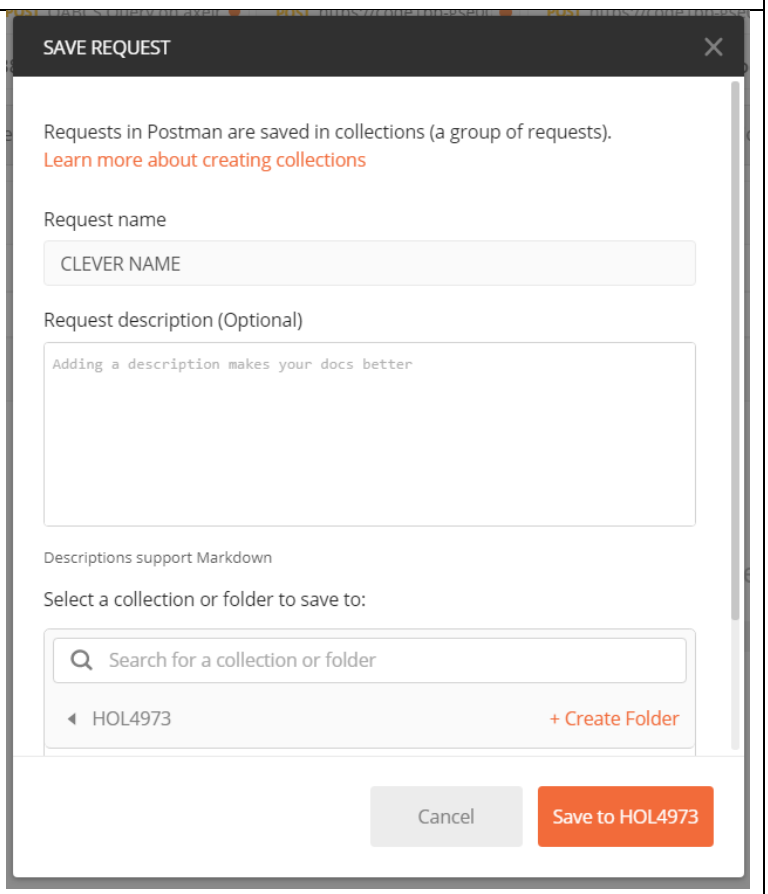
	Blockchain" or "Blockchain" – click on the service to open the Blockchain Cloud Service Instance	
1.04	You should now be in the pre-created Blockchain instance space. Locate the Instance named **papesPlantation**	 *Above is a screenshot from a random environment, please find the name* **papesPlantation**
1.05	Click on the hamburger menu to the right of the instance name and choose Blockchain Console to open up an overview of Pape's Plantation Blockchain network.	
1.06	Welcome to the blockchain console! This is a cloud GUI from Oracle, on top of a Hyperledger Fabric Network. On the Dashboard we can see the overall health of the system.	

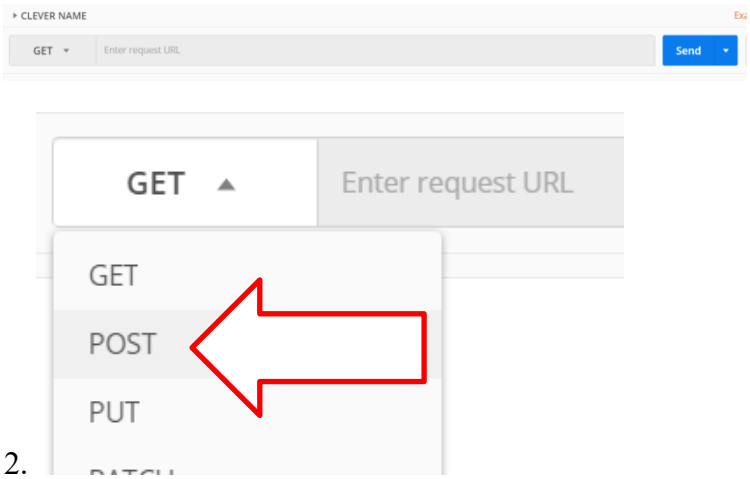
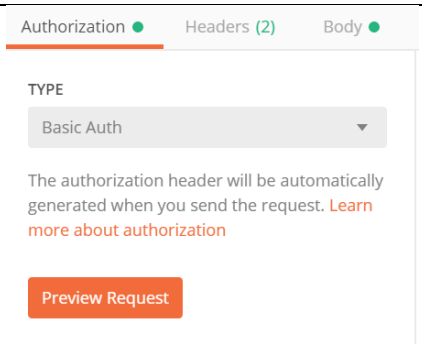
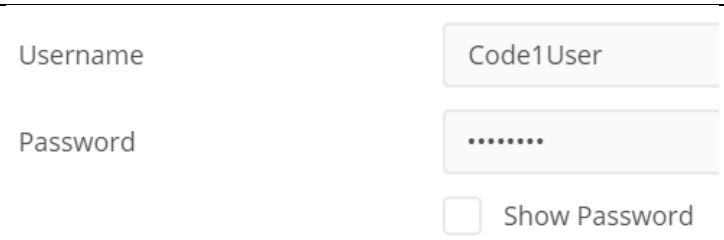
The dashboard and other tabs are part of an overview GUI controlling the Blockchain, something Oracle added on top of Hyperledger Fabric, in OBCS.		
1.07	Navigate to the Chaincodes tab. This is where we will deploy and later upgrade our Chaincode (or Smart Contract) for Pape's Plantation. Click on "Deploy a New Chaincode"	
1.08	A wild "Deploy Chaincode" - dialogue appears. Select "Quick Deployment"	
1.09	Fill in the following parameters in the Quick Deployment Dialogue: Chaincode Name: [any – but remember your choice!] Version: v1 Initial Parameters for chaincode instantiation: [leave empty]	

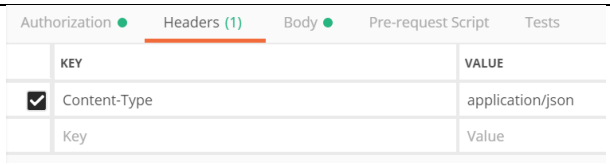
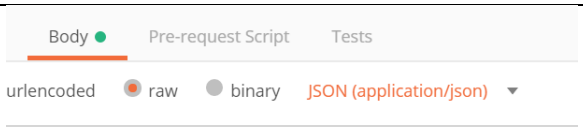
	Channel: [Select one of the precreated channels ch0-ch9. Remember your choice!] REST Proxy: **restproxy1** Chaincode Source: Upload the .zip file from GitHub. **REF: 0.3** When you are ready, click "Submit"	
The Chaincode included in the .zip-file was downloaded from GitHub and in our use-case this code was an existing chancode handed to us from another participant on the network (Franco's Forestry). We will modify the code in a later step.		
1.10	Click "Yes" to confirm that you are a code wizard who knows what you are doing. *The reason we cannot delete a Chaincode once it has been deployed, is because it is now a part of our immutable Blockchain.*	

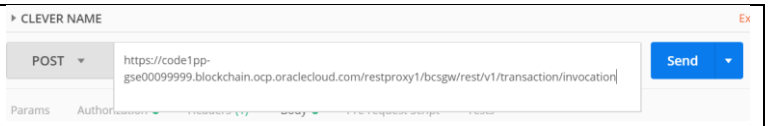
1.11	The code will now install and instantiate on the selected channel. **Sometimes this may take up to a few minutes.** Stay calm and carry on. After a while we should see the "SUCCESS!" message in the Deploy Chaincode dialogue. Well done! You can now close the dialogue.	
1.12	The Chaincode with your name (the name you chose in step 1.09) should now appear in the list of chaincodes under the "Chaincodes" tab in the cloud interface.	
We now have our first Chaincode deployed. This should mean that Pape's Plantation is now ready for business. Let's go ahead and verify it using our REST API (or our application) in the next section.		
Great Work! You have deployed your first chaincode on the Blockchain… in the cloud! You are now ready to move on to the next section. Time to call your code using REST protocol and PostMan application. **PS. Don't forget to discuss your work with your lab partner!**		

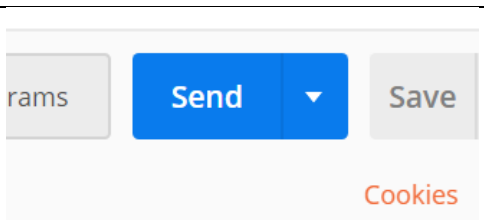
S.No	Action	Description
Chapter 2: Working with the Blockchain REST API using Postman		
Documentation written using Postman v6.3.0 on Windows.		
2.00	If you haven't done so already, download & install the latest stable version of Postman suitable for your environment. https://www.getpostman.com/	Postman (https://www.getpostman.com/) is a free API Development environment, for individuals and small projects. We will use it to query the Blockchain using the REST API. Postman is available for Mac, Windows and Linux. *PostMan is not Oracle software.*
2.01	Start Postman desktop app. If you haven't done so already, you will be asked to create a Postman account.	Now, we are going to work on this network and create transactions that will be recorded as blocks in to our Blockchain network. The OBCS can be accessed via the REST API (something that Oracle added on top of HLF) or, in a real-world scenario through business apps interacting via REST with OBCS. We will begin with the REST API and some sample queries.
PLEASE NOTE: You must complete ALL the steps in this section (2.XX) to be able to make a successful request in PostMan!		

2.02	In Postman: Create a new request	
2.03	In the "Save Request"-dialogue, give the request a *clever name* and choose or create a Collection (like a folder of your saved requests). We suggest you name your collection *HOL4973* Press the orange Save button. *You need to select a collection in order to make the Save button orange and clickable.*	

2.04	In your request: Start by choosing POST as method for your request.	1.  2.
2.05	In the **Authorization Tab**: Choose TYPE Basic Auth	
2.06	In the **Authorization Tab**: Put in your authorization details. This is the same **temporary lab account credentials that you used to login to the cloud environment** (separate handout).	

2.07	In the **Headers Tab**: - Add KEY = Content-Type - VALUE= application/json - Leave DESCRIPTION empty	
2.08	In the **Body tab**: - Choose “raw” radio button - Choose JSON in dropdown list	
2.09	In the Body tab: Put in your request parameters Parameters in red must be synced with your chaincode in a previous step. Remember to remove the brackets. When you create a tree-order, you send three parameters: - "createTreeOrder" is the method we use inside the invoke function - "YOURNAME" is a string parameter of your choice, the name of your order that will be put on the ledger - "YOURMONEY" is a int parameter of your choice, the fictive amount of money that you wish to plant trees for. Set it to anything between 50 & 500 money.	

2.10	Add the REST API URL for your code. The URL are basically made up of two parts:
	
	1. Your Cloud Instance URL, found in your web browers address field:
	
	`https://<CHANGE>.blockchain.ocp.oraclecloud.com/`
	
	2. The REST endpoint for Invocation:
	
	`<CHANGE_REST_Proxy>/bcsgw/v1/transaction/invocation`
	
	For more information, see the OBCS REST Documentation: https://docs.oracle.com/en/cloud/paas/blockchain-cloud/rest-api/rest-endpoints.html
		
		
		Please make sure the proxy (highlighted) is the same proxy you chose in a previous step (1.09)
		
		`https://<CHANGE>.blockchain.ocp.oraclecloud.com/<CHANGE_REST_Proxy>/bcsgw/rest/v1/transaction/invocation`
		
		`Example:`
		
		`https://code1pp-gse00099999.blockchain.ocp.oraclecloud.com/restproxy1/bcsgw/rest/v1/transaction/invocation`
		
		We are now leveraging the oracle REST API, built on top of Hyperledger Fabric. We could use this API to build applications bur please note that Hyper Ledger Fabric also has a SDK available that we can use, regardless of where HLF is deployed.

2.11	Press SEND to invoke the Chaincode via Postman	

2.12	If Your request was successful, you should now find something like this in the response **(scroll down in PostMan to find it).**	`{`
		`  "returnCode": "Success",`
		`  "txid": "48874c4a4a88e80d09f38e0b4576fd4ec010a25819e096ce58234632a522fe42"`
		`}`

We have now invoked the chaincode and written to our ledger, on the channel that Pape's Plantation share with the founder organization: Claudio's Controlling (responsible for the Trusted Trees brand).		

Congratulations - You completed this chapter!

PS. You can now verify your transaction ID against the ledger. In the Oracle Blockchain Cloud Service Console, go to "Channels" and select your channel from the list. This takes you to the ledger for that channel, where you will be able to see your transaction (with same TxID) See screenshot below

Cloud Console

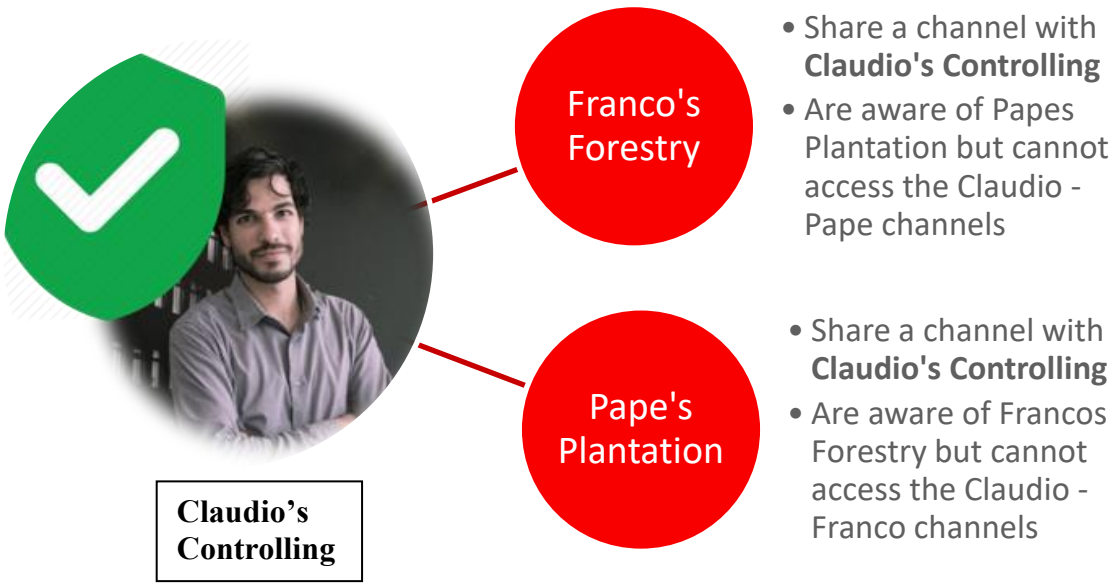
PostMan

S.No	Action	Description

Chapter 3: Raising the Bar

Great work so far. You have installed and tested the pre-created Chaincode. Time to modify and upgrade the Chaincode to a new version.

In this part we will make use of the Hyperledger Fabric technology CHANNELS. Channels allows us to have separate ledgers in the same network. In this case we will modify the price of planting trees in the channel that Pape's plantation shares with Claudio's Controlling - meaning that Pape's Plantation will now (potentially) plant trees at a different price than the other participants such as Franco's Forestry. Since Pape's Plantation and Franco's Forestry are not on the same channel, they will not be aware of each other's pricing. This is our situation:



ORACLE

3.00	Take a closer look at the code example. Inside the .go file you can find more information on what has been going on in the previous steps.	Locate the .go code in the downloaded .zip-file and open it in your preferred text editor to view and modify it.
3.01	To the right is a summary of some interesting methods inside the Invoke function. When we called the **createTreeOrder** (2.09) we sent two in parameters: YOURNAME (str) and YOURMONEY (int) What happened on the Chaincode side was that the method transformed these two parameters into an actual tree-order.	Func Invoke (**method**) **createTreeOrder**(name, cash) cash/price = number of trees to plant put name + ordered on ledger **query**(type, name) return state

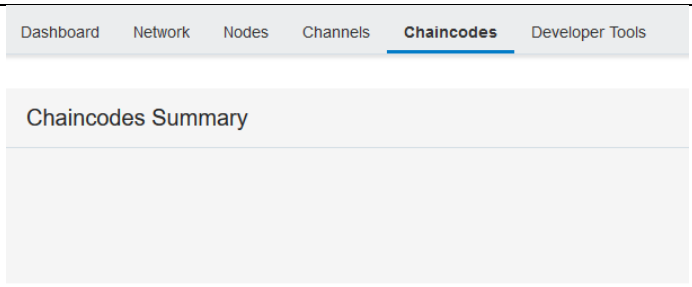
3.02	You can verify your tree-order exists on the ledger by sending a query, instead of using the createTreeOrder method in your POST call from PostMan.
	
	See example of the call on the right. **[YOURNAME]** represent the same name you put in in step 2.09
	
	Please note the highlighted argument changes.
		{
		"channel": "**[YOURCHANNEL]**",
		"chaincode": "**[YOURCHAINCODE]**",
		"method": "invoke",
		"chaincodeVer": "**v1**",
		"args": ["query", "name", "**[YOURNAME]**"]
		}
		
		Example:
		{
		"channel": "ch1",
		"chaincode": "**live4thecode**",
		"method": "invoke",
		"chaincodeVer": "**v1**",
		"args": ["query", "name", "**Larry**"]
		}

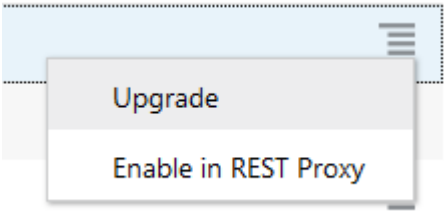
ORACLE

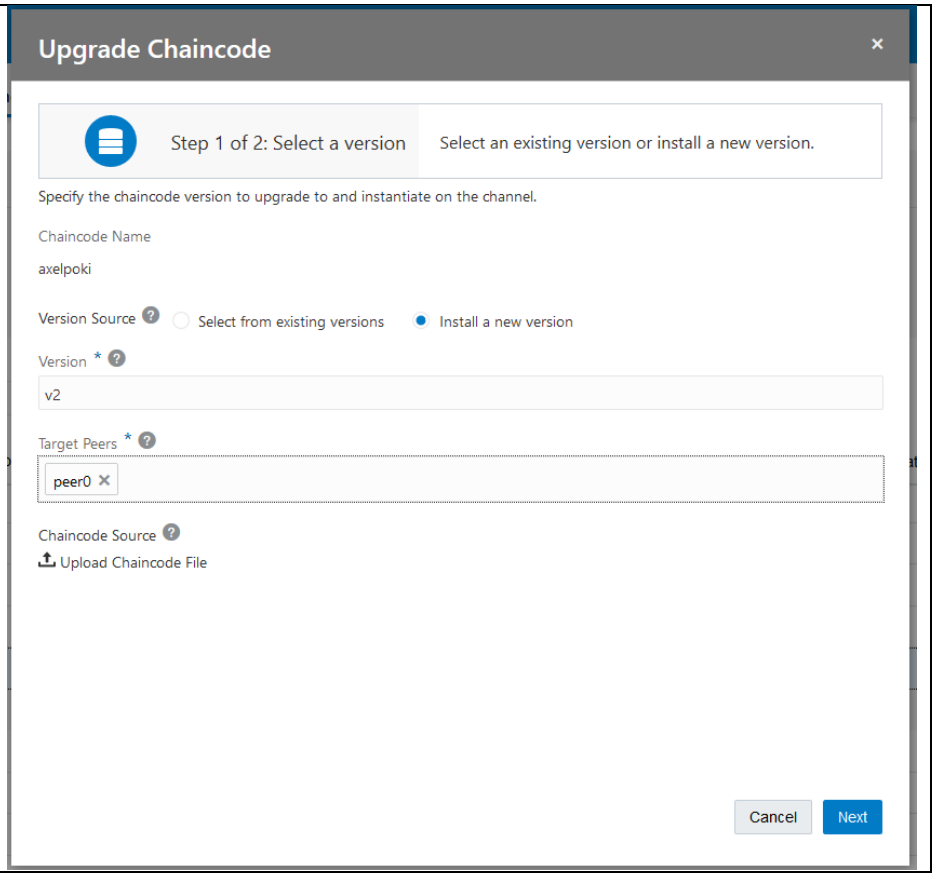
3.03	The response in PostMan should look something like the example on the right. **[YOURNAME+int]** represents your automatically generated tree order ID. We will use the ID in the next step. *Ex: Larry123*
	
	PS: the fact that we got a txid back, means that this query was registered on the ledger. By design, you could choose if a query like this should be registered on the leger or not.
		{
		"returnCode": "Success",
		"result": {
		"payload": "**[YOURNAME+int]**",
		"encode": "UTF-8"
		},
		"txid": "99521901268c25967457ce7a4"
		}
		
		Example:
		{
		"returnCode": "Success",
		"result": {
		*"payload": "**Larry123**",*
		"encode": "UTF-8"
		},
		"txid": "99521901268c25967457ce7a4"
		}

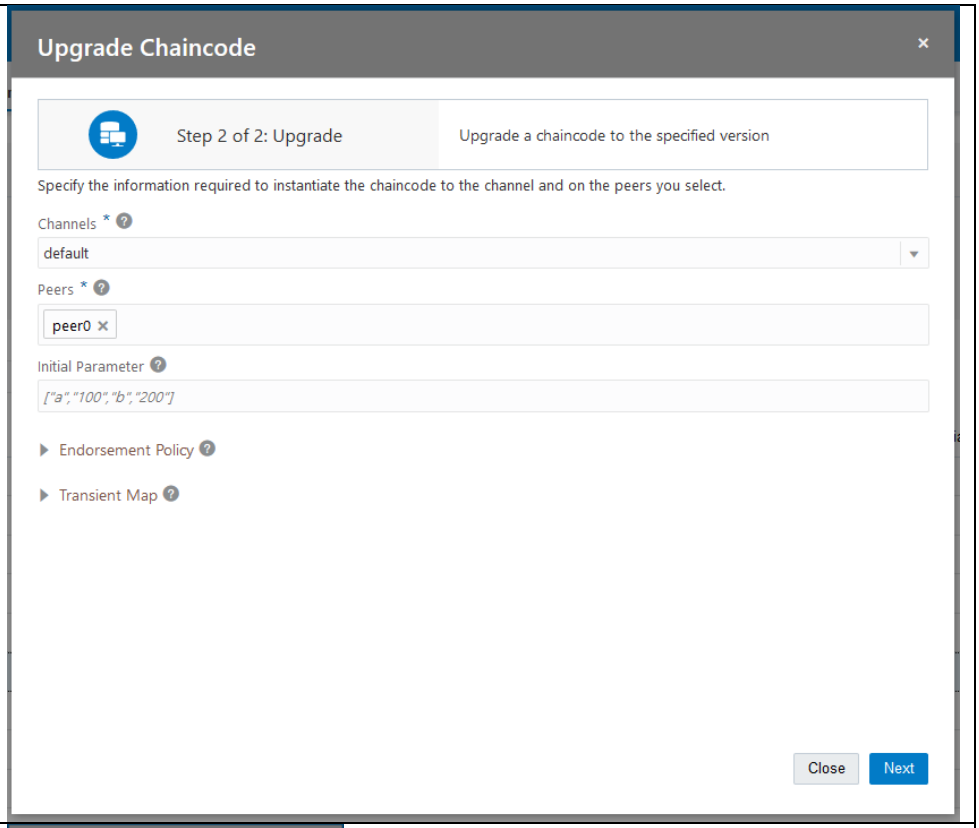
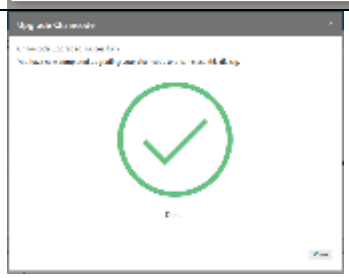
3.04	Again in PostMan, as a last verification step, send the id (**[YOURNAME+int]**) from the previous step as a new query. **Please make sure you also change the second argument to "id".**
	
	Please note the highlighted argument changes.
		{
		"channel": "**[YOURCHANNEL]**",
		"chaincode": "**[YOURCHAINCODE]**",
		"method": "invoke",
		"chaincodeVer": "**v1**",
		"args": ["query", "id", "**[YOURNAME+int]**"]
		}
		
		Example:
		{
		*"channel": "**ch1**",*
		*"chaincode": "**live4thecode**",*
		"method": "invoke",
		*"chaincodeVer": "**v1**",*
		*"args": ["query", "id", "**Larry123**"]*
		}

3.05	The response in PostMan should look something like the example on the right.
	
	This time "payload represents the number of trees sent to Francos Forestry. The number depends on how much YOURMONEY you put in as the argument in step 2.09
		{
		"returnCode": "Success",
		"result": {
		"payload": "**10**",
		"encode": "UTF-8"
		},
		"txid": "416f9794c14d82b171daea594e0"
		}

Thanks to the above steps, we have now been able to find our tree order that was created, and we found out how many trees that was actually in that order, depending on how much money we put in but also depending on the price of each tree. *NOTE: This may seem like a lot of work, but typically these steps would be handled by an application on the user side – handling one or more requests at the time and also presenting the information in an nice way for the end user.*		
3.06	Now, Papes Plantation has agreed to plant trees at the price of 20 money* per tree. Please update the code accordingly. **(EUR/USD/BTC… we don't know)*	Hint: locate where the transaction from money to tree takes place and update the code.
3.07	When you are happy with the code change – **save the .go file and zip it**. We will now upload the .zip-file to upgrade the Chaincode.	
3.08	In the Blockchain Cloud Service Console – navigate to the tab Chaincodes.	

3.09	Locate your chaincode in the list of deployed chaincodes (the chaincode you deployed in Chapter 1, with the name you chose). Press the hamburger menu to the right, linked to your Chaincode, and choose “Upgrade”. This will allow us to deploy a new version of the Chaincode: The code that you just updated (3.07)	

3.10	In the “Upgrade Chaincode” dialogue: - Make sure you have the right chaincode name - Choose the radio button “Install a new version” - Name your new version of the Chaincode **(we suggest v2)** - Choose peer0 as target - Upload your zipped .go file Press Next	

3.11	In the second step of the dialogue, you should see "SUCCESS". - Choose the same channel as before - Choose peer0 - Leave he rest of the parameters unchanged. Press Next.	
3.12	After a short while (stay calm…) you should see your updated chaincode be deployed. You can close this dialogue.	

Great work! Your have upgraded the Chaincode for Pape's Plantation and fulfilled your mission for this customer.

If you wish, you can now test your code by using PostMan and recreating the steps in Chapter 2, targeting your upgraded version of the Chaincode.

PS. Don't forget to discuss your work with your lab partner!

ORACLE

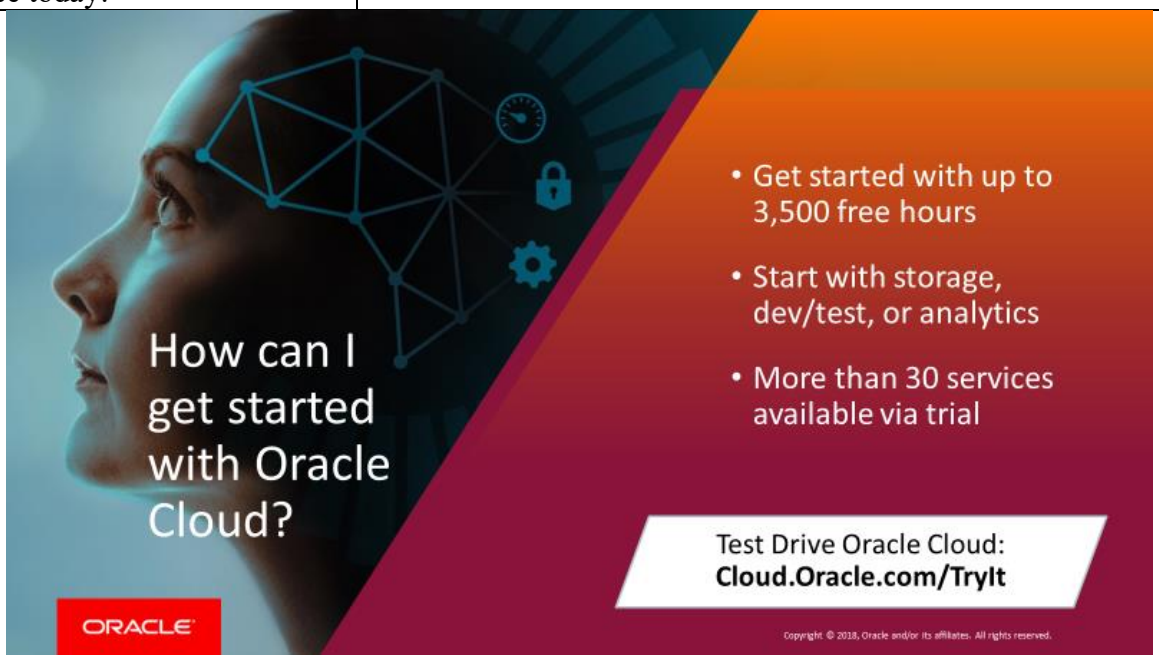
S.No	Action	Description
Chapter 4: Challenge!		
(Depending on the time you have available for the lab – you might want to stop here OR you take the lab one step further.) The Founder, Claudio's Controlling (responsible for the Trusted Trees brand), have heard about your skills as a developer. They have realised they could really benefit from a "query all"- method. Currently they are only able to use Invoke and query one ID or NAME at a time		
4.00	Write a "query all" method inside the Invoke function of your current Chaincode.	The method should be able to query all NAME:s and/or ID:s written to the specific ledger.
4.01	You can test your chaincode directly in the cloud interface.	
Please feel free to upload an updated example of the chaincode to GitHub and make a Pull Request to brondera/oow-18 *PS. It is true that everything in the code can be made better. Please feel free to send us your suggestions on GitHub. Thanks!*		
Your code will make a difference for the future. Thanks!		

ORACLE

References / Further Reading

Thank you for your participation in this lab! Hopefully this has triggered your interest in Blockchain, Hyperledger Fabric, and how you can use it with Oracle Cloud. If so, the Internet is full of more information for you. Find a few of our favourite sources below.

Hyperledger	https://www.hyperledger.org/
Hyperledger Fabric	https://www.hyperledger.org/projects/fabric
The Linux Foundation – Blockchain for Business Certification Program	https://www.edx.org/professional-certificate/linuxfoundationx-blockchain-for-business
Oracle Blockchain Cloud Service	https://cloud.oracle.com/en_US/blockchain
Oracle Blockchain Cloud Service Documentation	https://docs.oracle.com/en/cloud/paas/blockchain-cloud/
Get started with Oracle Cloud for free today!	https://cloud.oracle.com/TryIt



ORACLE

Appendix

This appendix contains a number of the most frequently asked questions as well as a glossary covering some of the most commonly used terminology for the technologies used in this lab. This is in no way a complete guide, but more guidelines in preparation for your own exploration.

Appendix - Glossary

Network

Oracle Blockchain Cloud Service (OBCS) use the concept of network to enable real-time transaction across participants. A blockchain network has a founder that creates and maintains the network, and participants that join the network. All organizations included in the network are called members.

Chaincode

Chaincode is a program, written in Go, Node.js, or Java that implements a prescribed interface. Chaincode runs in a secured Docker container isolated from the endorsing peer process. A Chaincode typically handles business logic agreed to by members of the network, so it may be considered as a "Smart contract".

Ledger

A ledger contains the current state of a business as a journal of transactions. A blockchain is a system for maintaining distributed ledgers of facts and the history of the ledgers' updates. A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography

Peers

A blockchain network is comprised primarily of a set of *peer nodes* (or, simply, *peers*). Peers are a fundamental element of the network because they host ledgers and smart contracts. Each peer that joins a channel has its own identity that authenticates it to the channel peers and services. Although peers can belong to multiple channels, the information on transactions, ledger state, and channel membership is restricted to peers within each channel.

REST Proxy

Applications use a software development kit (SDK) to access the application programming interfaces (APIs) that permit queries and updates to the ledger. The REST APIs provided by OBCS have been created with flexibility in mind; you can invoke a transaction, invoke a query, or view the status of a transaction. However this means that you will likely want to wrap the existing API endpoints in an application to provide object-level control. Applications can contain much more fine-grained operations.

Channel

Channels partition and isolate peers and ledger data to provide private and confidential transactions on the blockchain network. Members define and structure channels to allows specific peers to conduct private and confidential transactions that other members on the same blockchain network cannot see or access. Each channel includes peers, the shared ledger, chaincodes instantiated on the channel, and one or more ordering service nodes.

Consensus

Consensus is required before blocks or transactions are written to the ledger. Therefore, the existence and validity of a data record can't be denied. After endorsement policies are satisfied and consensus is reached, data is grouped into blocks and blocks are appended to the ledger with cryptographically secured hashes that provide immutability. Permissioned blockchains use an access control layer to enforce which organizations have access to the network. All nodes in the network are known and use consensus protocol to ensure that the next block is the only version of truth.

There are three steps to consensus protocol:

- Endorsement — This step determines whether to accept or reject a transaction.
- Ordering — This step sorts all transactions within a time period into a sequence or block.
- Validation — This step verifies that the required endorsement are gotten in compliance with the endorsement policy and organization permissions

Appendix - FAQ

What is Oracle Blockchain Cloud Service?

OBCS is a platform, which customers and partners can use to create permissioned blockchain networks for private or consortia models. OBCS provides the enabling distributed ledger technology and related capabilities for building blockchain applications and business networks.

What is included in an OBCS service instance?

Each service instance has all the components necessary to create a member organization with one or more validating (peer) nodes and an ordering service, plus a REST proxy and an operations console. Instances can also be created that do not include their ordering service in order to join a network that already includes an instance with an ordering service. A blockchain network would include multiple instances, one of which has an ordering service (and is identified as a Founder).

Is Oracle creating a business network on its blockchain?

No, Oracle is providing a blockchain cloud platform for customers and partners to build business networks.

Are there tokens/cryptocurrency involved in running transactions?

No, as an enterprise-focused permissioned blockchain platform, OBCS doesn't require any tokens or cryptocurrency.

What is Hyperledger and how is it related to Blockchain?

Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration focused on business blockchain frameworks, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chain, manufacturing and technology. See https://www.hyperledger.org.

How is OBCS different from Hyperledger Fabric?

OBCS is based on Hyperledger Fabric with a number of enhancements for greater resilience, performance, scalability, security, manageability, enterprise integration, etc. It is deeply integrated with foundational services in Oracle Cloud and provides additional capabilities, e.g., REST proxy for synchronous transactions, operations console with a number of configuration, administration, and monitoring capabilities. It maintains compatibility with Hyperledger Fabric at the protocol and API level.

ORACLE

More information/Sources:

https://cloud.oracle.com/en_US/blockchain/faq

https://docs.oracle.com/en/cloud/paas/blockchain-cloud/user/what-is-oracle-autonomous-blockchain-cloud-service.html

https://hyperledger-fabric.readthedocs.io/en/release-1.3/key_concepts.html

REST API for Oracle Blockchain Cloud Service

https://docs.oracle.com/en/cloud/paas/blockchain-cloud/user/using-oracle-autonomous-blockchain-cloud-service.pdf

ORACLE