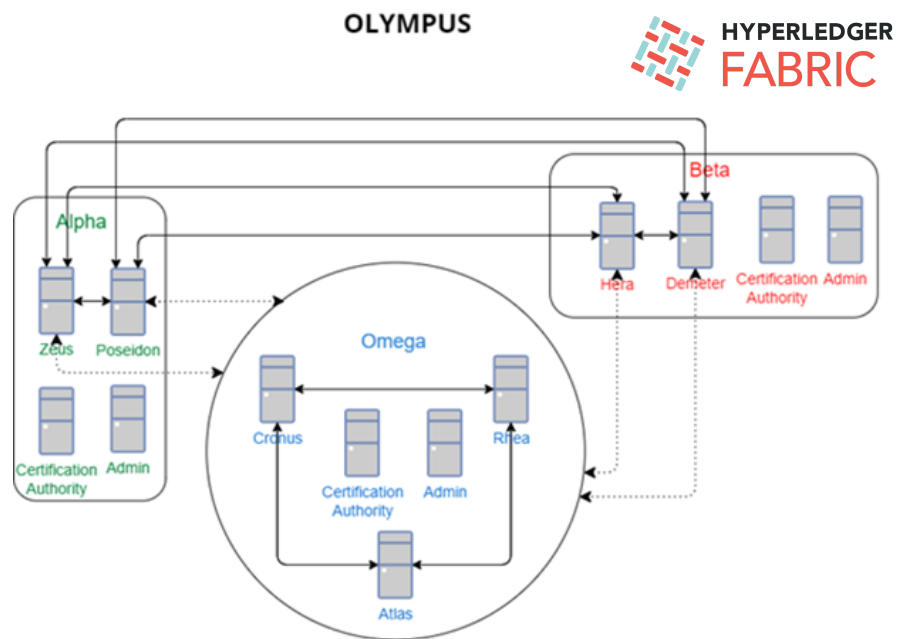


# Guide to Deploy Production Network Hyperledger Fabric



**Ricardo Martins Gonçalves**

May 2022

# Table of Contents

<b>1. Introduction</b>	<b>3</b>
<b>2. Network Configuration</b>	<b>4</b>
<b>3. Set Up a Cluster</b>	<b>5</b>
<b>4. Global Set Up</b>	<b>6</b>
4.1. Set up the Network	6
4.2. Set up the Virtual Machines	6
4.3. Install the Prerequisites	10
4.4. Configure DNS names	11
<b>5. Set Up Your CA's</b>	<b>13</b>
5.1. TLS CA Server	13
5.2. Organisation CA Server	14
<b>6. Register and Enrolling Identities with a CA</b>	<b>16</b>
6.1. Register	16
6.2. Enroll	17
<b>7. Folder Structure</b>	<b>18</b>
7.1. Organisations MSP	18
7.2. Local MSP	20
<b>8. Deploy the Peer Nodes</b>	<b>21</b>
<b>9. Deploy the Ordering Service</b>	<b>23</b>
9.1. System Genesis Block	23
9.2. Deploy the Ordering Nodes	25
9.3. Start the Ordering Service	26
<b>10. Join Peers to the Channel</b>	<b>27</b>
<b>11. Write the Chaincode</b>	<b>29</b>
<b>12. Deploy the Chaincode</b>	<b>30</b>
12.1. Package	30
12.2. Install	30
12.3. Approve for My Organisation	31
12.4. Commit	32
<b>13. Test Network</b>	<b>33</b>

# 1. Introduction

Hyperledger Fabric (HLF) is one of the leading software programs to create private and permissioned Blockchain networks. The HLF documentation was made so that each company would be free to choose various factors, including communication methods, connections' security, and consensus algorithm. Although this gives the possibility for each organisation or consortium to have a "tailor-made" Blockchain, it makes the documentation difficult to follow for someone who is not comfortable with this technology.

This tutorial was made to share a relatively simple and straightforward way to deploy a production network. There are many possibilities and choices that you can make to create a Hyperledger Fabric (HLF) Network, this is an example of a functional HLF Blockchain. As an alternative, if you don't want to deploy a network, you could use the test network provided by HLF.

At the end of this tutorial, you have a private and permissioned Blockchain with three different organisations (2 peer organisations and 1 orderer organisation), one ordering service with 3 orderers, and 4 peers (two of each organisation). You will be able to deploy the first chaincode (or SmartContract), to edit and "query" the data on the Blockchain. This tutorial uses CouchDB as a database so that queries can be made to data.

This tutorial was created through the HLF [documentation](#) and tried the most similar approach to the documentation. However, there were several commands and decisions influenced by forums and other non-official tutorials.

It is possible to make some changes to the network such as using a different number of organisations or an ordering service with a different number of nodes, however, if you do not have experience with HLF, we advise you to follow this tutorial strictly.

## 2. Network Configuration

Network: **Olympus**

Peer Organisations: **Alpha** and **Beta**.

Ordering Organisations: **Omega**

Every organisation needs 2 Certification Authorities (CA), one for TLS and other for the Organisation itself. The admin identity of an organisation does not require a Virtual Machine (VM), however, to simplify, the admin will have a dedicated VM.

Virtual Machines

### Alpha:

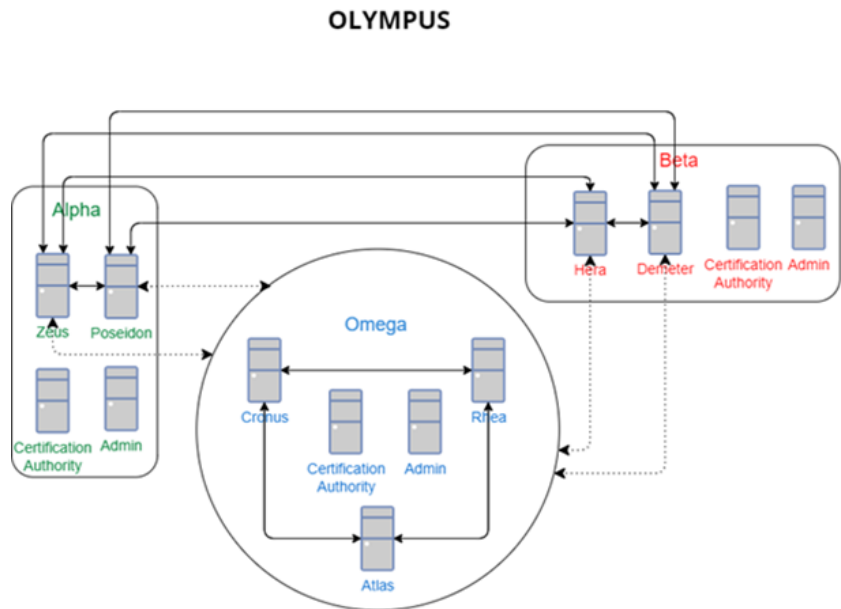
Certification Authority  
Admin  
Zeus (Peer)  
Poseidon (Peer)

### Beta:

Certification Authority  
Admin  
Hera (Peer)  
Demeter (Peer)

### Omega:

Certification Authority  
Admin  
Cronus (Orderer)  
Rhea (Orderer)  
Atlas (Orderer)



Note: All peers are Greek gods and all orderers are Greek titans.

The diagram shows the three organisations and the connections between the Machines. All peers are connected to the other peers and to the ordering service. All orderers are connected to the other orderers.

Network diagram made in <https://online.visual-paradigm.com>

### 3. Set Up a Cluster

Every VM of this network runs Ubuntu 20.04 with 4GB of Ram and a Dynamic disk that can use up to 30 GB.

13 VM's \* 4GB RAM = 52GB

13VM's \* 30GB = 390 GB

CPU: Intel Core i7 2.10 with 12 cores of equivalent

The used Host has 128 GB of RAM, 1 TB of storage and a 12th Gen Intel(R) Core(TM) i7-12700 2.10 GHz with 12 cores. The operating system of the host is windows 11.

The minimum recommended is 64 GB of RAM, 512 GB of storage and an Intel i7 with 12 cores.

Regarding operating systems, we tested Windows 10 and 11, but some equivalent Linux distributions or macOS will work if they can install Virtual Box or other virtualization software.

## 4. Global Set Up

We recommend using Virtual Box for the VMs, however, other virtualization software will do the same. This tutorial gives a detailed explanation of how to set up each VM separately. If you want to be more efficient and have some experience, you can automate the process using Vagrant or other equivalent software. If you want to be fast, but don't have any technology equivalent to vagrant, you can set up just one VM and then clone. When cloning, be careful, after you make the copy, change the MAC address so that each VM has a different IP address.

Download and install Virtual Box (<https://www.virtualbox.org/wiki/Downloads>)

Download a Desktop image of Ubuntu (<https://releases.ubuntu.com/focal/>)

### 4.1. Set up the Network



Go to **Preferences** -> **Network** -> **Add Network** -> Double click on the Network that you created -> Change Network Name to **Olympus** and the Network CIDR to **192.168.100.0/24**, the rest can be left unchanged -> Click Ok.

Create a Shared Folder in the host machine (E.g.: C:\Users\Admin\VirtualBox VM's\shared folder)

### 4.2. Set up the Virtual Machines

Open VirtualBox application.

Click New 

Name: **AlphaCA** | **AlphaAdmin** | **Zeus** | **Poseidon** | **BetaCA** | **BetaAdmin** | **Hera** | **Demeter** | **OmegaCA** | **OmegaAdmin** | **Cronus** | **Rhea** | **Atlas**

Machine Folder can be left unchanged

Type: **Linux**

Version: **Ubuntu (64-bit)**

(Click Next)

Memory size: **4096** (Next)


**Create a virtual disk now** (Create)

**VDI** (Next)

**Dynamically allocated** (Next)

File location can be left unchanged, size: **30.00 GB** (Create)

Before starting the VM:

Go to settings 

## General -> Advanced

Shared Clipboard:	Bidirectional
Drag'n'Drop:	Bidirectional

## Network -> Adapter 1

Adapter 1	Adapter 2	Adapter 3	Adapter 4
<input checked="" type="checkbox"/> Enable Network Adapter			
Attached to: NAT Network			
Name: Olympus			
Advanced			
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)			
Promiscuous Mode: Allow All			
MAC Address: 080027BD563E			
<input checked="" type="checkbox"/> Cable Connected			
Port Forwarding			

## Shared Folders

Add new shared folder

Folder path: The path to the folder you created previously

Folder name: **shared\_folder**

Select **Auto-mount**

Add Share		?	×
Folder Path:	C:\Use...ox VMs\Shared Folder		
Folder Name:	shared_folder		
	<input type="checkbox"/> Read-only		
	<input checked="" type="checkbox"/> Auto-mount		
Mount point:			
OK		Cancel	

Click **Ok** and exit Settings.

Start the machine

In the Optical Disk Selector screen -> Click add -> Select the Ubuntu image previously downloaded -> Click Open -> Click Choose -> Click Start.

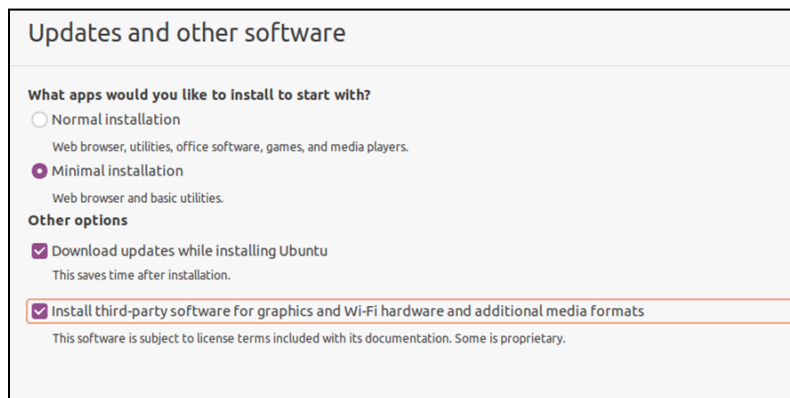
You can do a standard Ubuntu installation, if you are not familiar with the installation process of Ubuntu, you can use the following instructions.

After Ubuntu boots up, a screen with the option of installing Ubuntu will appear.

Click Install Ubuntu

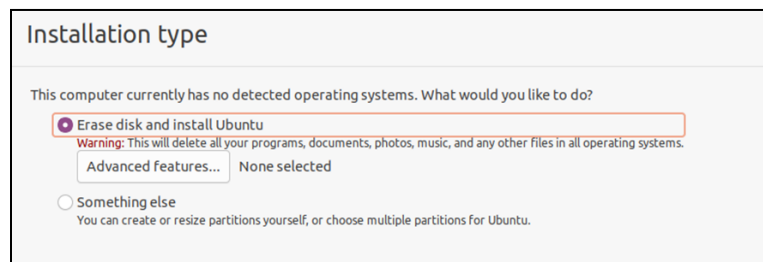
Choose the keyboard layout

Choose Minimal installation and check the two checkboxes



The screenshot shows the 'Updates and other software' window. It has a title bar with the text 'Updates and other software'. Below the title bar, there is a section titled 'What apps would you like to install to start with?'. Under this section, there are two radio button options: 'Normal installation' (with a description: 'Web browser, utilities, office software, games, and media players.') and 'Minimal installation' (which is selected, with a description: 'Web browser and basic utilities.'). Below this is a section titled 'Other options' with two checked checkboxes: 'Download updates while installing Ubuntu' (with a sub-note: 'This saves time after installation.') and 'Install third-party software for graphics and Wi-Fi hardware and additional media formats' (with a sub-note: 'This software is subject to license terms included with its documentation. Some is proprietary.').

Choose to Erase disk and install Ubuntu and click Install Now



The screenshot shows the 'Installation type' window. It has a title bar with the text 'Installation type'. Below the title bar, there is a message: 'This computer currently has no detected operating systems. What would you like to do?'. There are two radio button options: 'Erase disk and install Ubuntu' (which is selected, with a warning: 'Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.') and 'Something else' (with a sub-note: 'You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.'). Below the 'Erase disk and install Ubuntu' option, there is a button labeled 'Advanced features...' and the text 'None selected'.

Click Continue

Choose your location



Fill the Who Are You screen accordingly with the machine that you are creating.

E.g: Alpha CA

The screenshot shows the 'Who are you?' screen in the Ubuntu installer. The fields are filled with the following information:

- Your name: AlphaCA
- Your computer's name: alphaca
- Pick a username: alphaca
- Choose a password: alphaca (Weak password)
- Confirm your password: alphaca

Below the password fields, there are three radio button options:

- Log in automatically
- Require my password to log in
- Use Active Directory

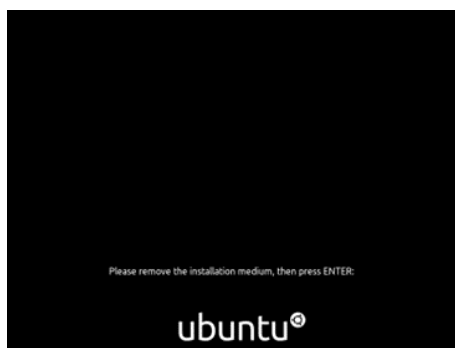
A note at the bottom states: "You'll enter domain and other details in the next step."

The other VMs can be filled with the information in the table.

Virtual Machine	Your name	Your computer's name	Pick a username	Choose a password
AlphaCA	AlphaCA	alphaca	alphaca	alphaca
AlphaAdmin	AlphaAdmin	alphaadmin	alphaadmin	alphaadmin
Zeus	Zeus	zeus	zeus	zeus
Poseidon	Poseidon	poseidon	poseidon	poseidon
BetaCA	BetaCA	betaca	betaca	betaca
BetaAdmin	BetaAdmin	betaadmin	betaadmin	betaadmin
Hera	Hera	hera	hera	hera
Demeter	Demeter	demeter	demeter	demeter
OmegaCA	OmegaCA	omegaca	omegaca	omegaca
OmegaAdmin	OmegaAdmin	omegaadmin	omegaadmin	omegaadmin
Cronus	Cronus	cronus	cronus	cronus
Rhea	Rhea	rhea	rhea	rhea
Atlas	Atlas	atlas	atlas	atlas

Click Restart Now

Wait and click enter when the following screen appears.



## 4.3. Install the Prerequisites

After the VM is started, test your Network connection by trying to open a web page in firefox. Open a terminal (Ctrl + Alt + T) and run the following commands to be able to use clipboard between host and the VM, have a shared folder and scale your VM's display.

Clipboard

```
$ sudo apt install -y virtualbox-guest-x11  
$ sudo VBoxClient --clipboard
```

Display

```
$ sudo apt install -y virtualbox-guest-additions-iso
```

Shared Folder

```
$ sudo adduser $USER vboxsf
```

Restart the VM

View -> Auto-resize Guest Display

Install the prerequisites of HyperLedger Fabric

Git

```
$ sudo apt install -y git
```

Net-tools

```
$ sudo apt install -y net-tools
```

Tree

```
$ sudo apt install -y tree
```

cURL

```
$ sudo apt install -y curl
```

Docker

```
$ sudo apt install -y docker
```

Docker Compose

```
$ sudo apt install -y docker-compose
```

Go

```
$ sudo apt install -y golang-go  
$ export PATH=$PATH:/usr/local/go/bin
```

Atom

```
$ wget -qO - https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -  
$ sudo sh -c 'echo "deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any/ any main" >  
/etc/apt/sources.list.d/atom.list'  
$ sudo apt update  
$ sudo apt install -y atom
```

CouchDB (Peers)

```
$ curl https://couchdb.apache.org/repo/keys.asc | gpg --dearmor > couchdb-repo-keyring.gnp &&  
sudo mv couchdb-repo-keyring.gnp /usr/share/keyrings/  
$ echo "deb [signed-by=/usr/share/keyrings/couchdb-repo-keyring.gnp]  
https://apache.jfrog.io/artifactory/couchdb-deb focal main" > couchdb.list && sudo mv couchdb.list  
/etc/apt/sources.list.d/  
$ sudo apt update  
$ sudo apt install -y couchdb  
Standalone  
Cookie: hlf  
CouchDB interface bind address: 127.0.0.1  
Password for the CouchDB "admin" user: adminpw
```

## 4.4. Configure DNS names

In each VM open a terminal

```
$ ifconfig
```

Find the IP address of the VM and write it in the next table.

Organisation	Virtual Machine	IP Address
Alpha	AlphaCA	192.168.100.___4
	AlphaAdmin	192.168.100.___5
	Zeus	192.168.100.___6
	Poseidon	192.168.100.___7
Beta	BetaCA	192.168.100.___8
	BetaAdmin	192.168.100.___9
	Hera	192.168.100.___10
	Demeter	192.168.100.___11
Omega	OmegaCA	192.168.100.___12
	OmegaAdmin	192.168.100.___13
	Cronus	192.168.100.___14
	Rhea	192.168.100.___15
	Atlas	192.168.100.___16

Create the DNS table in all the machines. This can be done using a dedicated VM as a DNS server, however, to simplify, you just edit the /etc/hosts files directly.

In each VM open a text editor

```
$sudo gedit /etc/hosts
```

Copy this block to the file (after 127.0.1.1)

Change the grey numbers by the numbers you used to fill the table

```
192.168.100.4      ca.alpha.olympus.pt
192.168.100.5      admin.alpha.olympus.pt
192.168.100.6      zeus.alpha.olympus.pt
192.168.100.7      poseidon.alpha.olympus.pt
192.168.100.8      ca.beta.olympus.pt
192.168.100.9      admin.beta.olympus.pt
192.168.100.10     hera.beta.olympus.pt
192.168.100.11     demeter.beta.olympus.pt
192.168.100.12     ca.omega.olympus.pt
192.168.100.13     admin.omega.olympus.pt
192.168.100.14     cronus.omega.olympus.pt
192.168.100.15     rhea.omega.olympus.pt
192.168.100.16     atlas.omega.olympus.pt
```

**Important:** You must change the IP address of the current machine to 127.0.0.1.  
E.g.: Zeus /etc/hosts will have the following list where the line corresponding to Zeus is  
"127.0.0.1 zeus.alpha.olympus.pt"

192.168.100.4	ca.alpha.olympus.pt
192.168.100.5	admin.alpha.olympus.pt
127.0.0.1	zeus.alpha.olympus.pt
192.168.100.7	poseidon.alpha.olympus.pt
192.168.100.8	ca.beta.olympus.pt
192.168.100.9	admin.beta.olympus.pt
192.168.100.10	hera.beta.olympus.pt
192.168.100.11	demeter.olympus.pt
192.168.100.12	ca.omega.olympus.pt
192.168.100.13	admin.omega.olympus.pt
192.168.100.14	cronus.omega.olympus.pt
192.168.100.15	rhea.omega.olympus.pt
192.168.100.16	atlas.omega.olympus.pt

Test your network

```
$ ping -c 2 atlas.omega.olympus.pt
$ ping -c 2 zeus.omega.olympus.pt
$ ping -c 2 cronus.omega.olympus.pt
$ ping -c 2 hera.omega.olympus.pt
```

...

You can turn off all machines (Click the X of VirtualBox and then Save the machine state).

## 5. Set Up Your CA's

Download the CA [binaries](#) in the host and copy them to the shared folder.

Open Certification Authority of each organisation (Alpha-CA, Beta-CA, Omega-CA).

Open a terminal

```
$ mkdir ~/HLF
```

```
$ cd ~/HLF
```

Extract the zip file in the shared folder to the current folder.

```
$ tar -xvf /media/sf_shared_folder/hyperledger-fabric-ca-linux-amd64-1.5.3.tar.gz -C .
```

Create a new directory for fabric-ca-client

```
$ mkdir fabric-ca-client
```

```
$ cd fabric-ca-client
```

```
$ mkdir tls-root-cert tls-ca alpha-ca (substitute the alpha-ca in the other CA)
```

Copy the client binary to the current location (~/.HLF/fabric-ca-client)

```
$ cp ../bin/fabric-ca-client .
```

The following processes for configuring the TLS and the Organisation Certification Authorities must be done in each of the CA's VMs.

### 5.1. TLS CA Server

Port: 7054

Change directory

```
$ cd ~/HLF
```

```
$ mkdir fabric-ca-server-tls
```

```
$ cd fabric-ca-server-tls
```

Copy the server binary to this location

```
$ cp ../bin/fabric-ca-server .
```

Init the server with an admin identity (in this case the admin "tls-admin" with password "tls-adminpw")

```
$ ./fabric-ca-server init -b tls-admin:tls-adminpw
```

Configure the TLS CA.

The previous command creates several files and folders, we are going to edit the fabric-ca-server-config.yaml present in the folder fabric-ca-server-tls.

```
$ atom ~/HLF/fabric-ca-server-tls/fabric-ca-server-config.yaml
```

Edit the following values:

```
tls.enabled: true
```

```
ca.name: alpha-tls-ca | beta-tls-ca | omega-tls-ca
```

```
csr.hosts: add ca.alpha.olympus.pt | ca.beta.olympus.pt | ca.omega.olympus.pt
```

```
signing.profiles.ca: comment the ca profile section
```

```
#ca:
```

```
#usage:
```

```
#- cert sign
```

```
#- crl sign
```

```
#expiry: 43800h
```

```
#caconstraint:
```

```
#isca: true
```

```
#maxpathlen: 0
```

Save and exit

The final result of each file can be seen in [ca/tls](#).

Delete ca-cert.pem and msp

```
$ rm ca-cert.pem && rm -r msp
```

Start the server

```
$ ./fabric-ca-server start
```

Open a new terminal

```
$ cd ~/HLF
```

Copy the TLS CA root certificate file to fabric-ca-client/tls-root-cert and change the name of the file to tls-ca-cert.pem

```
$ cp fabric-ca-server-tls/ca-cert.pem fabric-ca-client/tls-root-cert/tls-ca-cert.pem
```

Store the directory of fabric client binary in a variable.

```
$ cd fabric-ca-client
```

```
$ export FABRIC_CA_CLIENT_HOME=$PWD
```

Enroll the TLS CA admin in each CA using fabric-client (in machine beta-ca use beta and in omega-ca use omega)

```
$ ./fabric-ca-client enroll -d -u https://tls-admin:tls-adminpw@ca.alpha.olympus.pt:7054  
--tls.certfiles tls-root-cert/tls-ca-cert.pem --enrollment.profile tls --csr.hosts '*.alpha.olympus.pt'  
--mspdir tls-ca/tlsadmin/msp
```

Register and enroll the organisation CA bootstrap identity with the TLS CA in each CA (change the red to beta or omega)

```
$ ./fabric-ca-client register -d --id.name rcaadmin --id.secret rcaadminpw -u  
https://ca.alpha.olympus.pt:7054 --tls.certfiles tls-root-cert/tls-ca-cert.pem --mspdir  
tls-ca/tlsadmin/msp
```

```
$ ./fabric-ca-client enroll -d -u https://rcaadmin:rcaadminpw@ca.alpha.olympus.pt:7054  
--tls.certfiles tls-root-cert/tls-ca-cert.pem --enrollment.profile tls --csr.hosts '*.alpha.olympus.pt'  
--mspdir tls-ca/rcaadmin/msp
```

(Modify an identity if necessary)

```
./fabric-ca-client identity modify hera --secret herapw -u https://ca.beta.olympus.pt:7055 --mspdir  
beta-ca/rcaadmin/msp/ --tls.certfiles tls-root-cert/tls-ca-cert.pem
```

## 5.2. Organisation CA Server

Port: 7055

Open a terminal and change directory.

```
$ cd ~/HLF/fabric-ca-client/tls-ca/rcaadmin/msp/keystore
```

Rename the key in the current directory to key.pem

(change 3ce9be768baadce35ec36c59d73e0d6ecdb51efcfd9354d47fc0f1405286e85d\_sk to the name of the key that you have on your folder in each VM)

```
$ mv 3ce9be768baadce35ec36c59d73e0d6ecdb51efcfd9354d47fc0f1405286e85d_sk key.pem
```

Change directory to HLF

```
$ cd ~/HLF
```

The red parts of the commands must be changed to the organisation of the VM (alpha must be substituted by beta or omega).

```
$ mkdir fabric-ca-server-alpha
$ cp bin/fabric-ca-server fabric-ca-server-alpha/
$ cd fabric-ca-server-alpha
```

Copy the tls key and certificate to a separate folder

```
$ mkdir tls
$ cp ../fabric-ca-client/tls-ca/rcaadmin/msp/signcerts/cert.pem tls && cp
../fabric-ca-client/tls-ca/rcaadmin/msp/keystore/key.pem tls
```

Initialise the server

```
./fabric-ca-server init -b rcaadmin:rcaadminpw
```

Edit fabric-ca-server-config.yaml

```
$ atom fabric-ca-server-config.yaml
```

Edit the following values:

```
port: 7055
tls.enabled: true
Tls.certfile: tls/cert.pem
tls.keyfile: tls/key.pem
ca.name: alpha-ca
csr.hosts: add ca.alpha.olympus.pt
operations.listenAddress: 127.0.0.1:9444
```

Save and exit

The final result of each file can be seen in [ca/org](#).

Delete ca-cert.pem and msp.

```
$ rm ca-cert.pem && rm -r msp
```

Start the server

```
$ ./fabric-ca-server start
```

Enroll the CA admin

Open a new terminal in each CA VM

```
$ cd ~/HLF/fabric-ca-client
$ export FABRIC_CA_CLIENT_HOME=$PWD
$ ./fabric-ca-client enroll -d -u https://rcaadmin:rcaadminpw@ca.alpha.olympus.pt:7055
--tls.certfiles tls-root-cert/tls-ca-cert.pem --csr.hosts '*.alpha.olympus.pt' --mspdir
alpha-ca/rcaadmin/msp
```

Copy the folder tls-root-cert that is in ~/HLF/fabric-ca-client to the shared folder and change the name of the folder.

```
$ cp -r ~/HLF/fabric-ca-client/tls-root-cert /media/sf_shared_folder/
$ mv /media/sf_shared_folder/tls-root-cert /media/sf_shared_folder/(alpha)tls-root-cert
```

# 6. Register and Enrol Identities with a CA

## 6.1. Register

The register is made by the admin of each CA in the correspondent CA VM.

The register of all Alpha VM's is made on the AlphaCA VM, on BetaCA you will register all Beta VM's and on OmegaCA you will register all Omega VM's. Every identity must be registered and enrolled in both TLS and Organisation CA.

Open a terminal in the CA of Organisation

```
$ cd ~/HLF/fabric-ca-client
$ export FABRIC_CA_CLIENT_HOME=$PWD
```

The process of registering an identity is similar for each one of the VM, the following example is the register of Alpha Admin, you can use the same commands changing the **red** words according to the organisation and the **green** words according to the VM that you are registering. The words in **blue** have the alternative after the command.

TLS

```
$. /fabric-ca-client register -d --id.name alphaadmin --id.secret alphaadminpw -u
https://ca.alpha.olympus.pt:7054 --mspdir tls-ca/tlsadmin/msp --id.type admin --tls.certfiles
tls-root-cert/tls-ca-cert.pem --csr.hosts '*.alpha.olympus.pt'
```

[admin](#) | [peer](#) | [orderer](#)

To register the same identity in the organisation CA, the command is similar to the previous, there are just to changes **7054** -> **7055** and **tls-ca/tlsadmin** -> **alpha-ca/rcaadmin**.

Org

```
$. /fabric-ca-client register -d --id.name alphaadmin --id.secret alphaadminpw -u
https://ca.alpha.olympus.pt:7055 --mspdir alpha-ca/rcaadmin/msp --id.type admin --tls.certfiles
tls-root-cert/tls-ca-cert.pem --csr.hosts '*.alpha.olympus.pt'
```

The process to register the other identities is similar. Open a terminal in the CA of the organisation and register all the identities of that organisation using the previous commands, changing the red words to the name of the organisation (alpha | beta | omega), the green words to the name of the identity that you are currently registering and the blue word to the role that the current identity will play. (E.g.: Rhea: Omega CA, **alpha** -> **omega**, **alphaadmin** -> **rhea** and **admin** -> **orderer**).

In Alpha-CA VM:

```
zeus:zeuspw
poseidon:poseidonpw
```

In Beta-CA VM:

```
demeter:demeterpw
hera:herapw
```

In Omega-CA VM:

```
cronus:cronuspw
rhea:rheapw
atlas:atlaspw
```



## 6.2. Enroll

After the CA admin registers the identities, he will contact the user to give him the enrollID and his secret (in this case, this step is not necessary). All VMs must enroll in TLS CA and Organisation CA.

In each VM that is not a CA:

Open a terminal

Create a folder in your home directory

```
$ mkdir ~/HLF
```

```
$ cd ~/HLF
```

Extract the tar.gz file in the shared folder to ~/HLF

```
$ tar -xf /media/sf_shared_folder/hyperledger-fabric-ca-linux-amd64-1.5.3.tar.gz -C .
```

```
$ mkdir fabric-ca-client
```

```
$ cd fabric-ca-client
```

Copy the client binary to the current location (~/.HLF/fabric-ca-client)

```
$ cp ../bin/fabric-ca-client .
```

Copy the correspondent tls-root-cert directory that is in the shared folder to ~/.HLF/fabric-ca-client and change the name of the folder.

```
$ cp -r /media/sf_shared_folder/(alpha)tls-root-cert ~/.HLF/fabric-ca-client
```

```
$ mv (alpha)tls-root-cert tls-root-cert
```

```
$ export FABRIC_CA_CLIENT_HOME=$PWD
```

The colour code is the same as the previous section.

TLS

```
$ ./fabric-ca-client enroll -u https://alphaadmin:alphaadminpw@ca.alpha.olympus.pt:7054
```

```
--mspdir ./tls-alpha.olympus.pt/msp --csr.hosts '*.alpha.olympus.pt' --tls.certfiles
```

```
tls-root-cert/tls-ca-cert.pem
```

To register the same identity in the organisation CA, the command is similar to the previous, there are just to changes **7054 -> 7055** and **tls-alpha.olympus.pt -> alpha.olympus.pt**

Org

```
$ ./fabric-ca-client enroll -u https://alphaadmin:alphaadminpw@ca.alpha.olympus.pt:7055
```

```
--mspdir ./alpha.olympus.pt/msp --csr.hosts '*.alpha.olympus.pt' --tls.certfiles
```

```
tls-root-cert/tls-ca-cert.pem
```

The process to enroll the other identities is similar, just change the red words to the name of the organisation (alpha | beta | omega) and the green words to the name of the identity that you are currently registering (E.g.: Rhea: Omega CA, **alpha -> omega**, **alphaadmin -> rhea**).

Rename certificates and keys in each VM

E.g.: Alpha admin

```
(~/HLF/fabric-ca-client tls-alpha.olympus.pt/msp/signcerts)
```

```
cert.pem -> tls-alpha-admin-cert.pem
```

```
(~/HLF/fabric-ca-client/tls-alpha.olympus.pt/msp/keystore/)
```

```
*_sk -> tls-alpha-admin-key.pem
```

```
(~/HLF/fabric-ca-client alpha.olympus.pt/msp/signcerts)
```

```
cert.pem -> alpha-admin-cert.pem
```

```
(~/HLF/fabric-ca-client/tls-alpha.olympus.pt/msp/keystore/)
```

```
key.pem -> alpha-admin-key.pem
```

Do the same in the other VM, changing the red words to the name of the organisation and the green words to the name of the identity .

## 7. Folder Structure

In this topic we will organise the certificates and keys in a strong and useful structure advised by Hyperledger Fabric documentation.

Open the Admin VM's (Alpha Admin, Beta Admin and Omega Admin). In each of the 3 VM's, copy the certificates in `~/HLF/fabric-ca-client/alpha.olympus.pt/msp/cacerts` and `~/HLF/fabric-ca-client/tls-alpha.olympus.pt/msp/cacerts` to the shared folder, in the beta admin VM change the red words to beta and in omega admin change to omega.

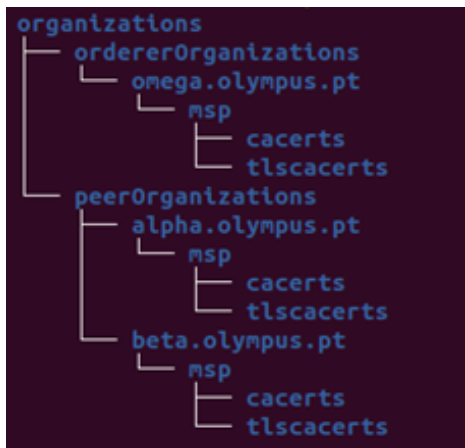
You can now minimise the admin VMs.

Open on of the VM (E.g.: Cronus)

```
$ cd ~/HLF
```

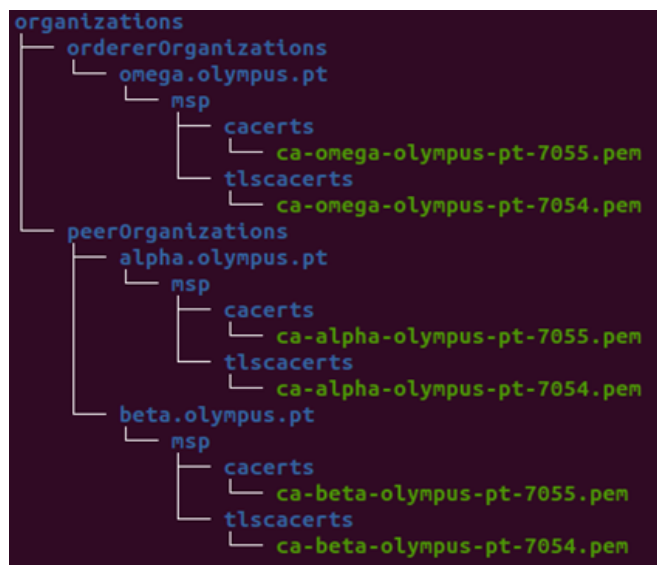
### 7.1. Organisations MSP

Create the folder structure in the image



Copy the certificates previously moved to the shared folder to the respective folder (the certificates ended in 7054.pem are in the tlscacerts folders and the certificates ended in 7055.pem are in cacerts folders.)

The result is in the following image.



In the msp folder of each organisation create a file called config.yaml. The file must have the following information. Copy the following text to the config.yaml file create in the msp of the respective organisation.

alpha

```
"NodeOUs:  
  Enable: true  
  ClientOUIdentifier:  
    Certificate: cacerts/ca-alpha-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: client  
  PeerOUIdentifier:  
    Certificate: cacerts/ca-alpha-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: peer  
  AdminOUIdentifier:  
    Certificate: cacerts/ca-alpha-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: admin"
```

beta

```
"NodeOUs:  
  Enable: true  
  ClientOUIdentifier:  
    Certificate: cacerts/ca-beta-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: client  
  PeerOUIdentifier:  
    Certificate: cacerts/ca-beta-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: peer  
  AdminOUIdentifier:  
    Certificate: cacerts/ca-beta-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: admin"
```

omega

```
"NodeOUs:  
  Enable: true  
  ClientOUIdentifier:  
    Certificate: cacerts/ca-omega-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: client  
  AdminOUIdentifier:  
    Certificate: cacerts/ca-omega-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: admin  
  OrdererOUIdentifier:  
    Certificate: cacerts/ca-omega-olympus-pt-7055.pem  
    OrganizationalUnitIdentifier: orderer"
```

Copy this structure to the shared folder and copy it again to each one of the VM's that is not a CA (and not the VM where you did this). The final folder structure can be seen in [organisations](#).

## 7.2. Local MSP

In every machine that is not a CA.

Go to the file that corresponds to the organisation of the VM

E.g.: `cronus -> ~/organizations/ordererOrganizations/omega.olympus.pt`

Create a folder name `admins` | `orderers` | `peers` depending on whether you are an admin, an orderer or a peer.

Create a folder with the name of your current VM.

E.g.:

```
admins
├── admin.beta.olympus.pt
│   ├── msp
│   │   ├── cacerts
│   │   ├── keystore
│   │   ├── signcerts
│   │   └── tlscacerts
│   └── tls
```

To the folder `tls` previously created, copy the certificate in `signcerts` and the key in `key` in `keystore` under the `~/HLF/fabric-ca-client/tls-beta.olympus.pt/msp`.

From the `beta.olympus.pt/msp`, copy the file under **`cacerts`** to **`msp/cacerts`**, the file in the **`keystore`** to the **`msp/keystore`**, the file in **`signcerts`** to **`msp/signcerts`** and the file under **`tls-beta.olympus.pt/msp/tlscacerts`** to **`msp/tlscacerts`**.

In other VM, substitute `beta` to the name of the VM's organisation.

Result:

```
admins
├── admin.beta.olympus.pt
│   ├── msp
│   │   ├── cacerts
│   │   │   └── ca-beta-olympus-pt-7055.pem
│   │   ├── keystore
│   │   │   ├── beta-admin-key.pem
│   │   ├── signcerts
│   │   │   └── beta-admin-cert.pem
│   │   ├── tlscacerts
│   │   │   └── ca-beta-olympus-pt-7054.pem
│   └── tls
│       ├── tls-beta-admin-cert.pem
│       └── tls-beta-admin-key.pem
```

The `config.yaml` file has two uses, in `ChannelMSP` is used to verify admin and in a `localMSP` (the `msp` of the node) is used to verify the admin and if the node itself has the role of peer or orderer.

Open the folder corresponding to the organisation of your VM and copy the `config.yaml` file under the organisation `msp` to the local `msp`. E.g.:

```
alpha.olympus.pt
├── msp
│   ├── cacerts
│   │   ├── ca-alpha-olympus-pt-7055.pem
│   │   └── config.yaml
│   ├── tlscacerts
│   │   └── ca-alpha-olympus-pt-7054.pem
├── peers
│   └── zeus.alpha.olympus.pt
│       ├── msp
│       │   ├── cacerts
│       │   │   ├── ca-alpha-olympus-pt-7055.pem
│       │   │   └── config.yaml
│       │   ├── keystore
│       │   │   └── zeus-key.pem
│       │   ├── signcerts
│       │   │   └── zeus-cert.pem
│       │   ├── tlscacerts
│       │   │   └── ca-alpha-olympus-pt-7054.pem
│       └── tls
│           ├── tls-zeus-cert.pem
│           └── tls-zeus-key.pem
```

## 8. Deploy the Peer Nodes

In the host machine download the peer [binaries](#).

Change the name to “nodes-hyperledger-fabric-linux-amd64-2.4.3.tar.gz” and copy it to the shared folder.

In each one of the 4 peers (you can close the rest of VM's).

Open a terminal

```
$ cd ~/HLF
$ mkdir fabric
$ cd fabric
```

Extract the folder from the shared folder to the current directory.

```
$ tar -xf /media/sf_shared_folder/nodes-hyperledger-fabric-linux-amd64-2.4.3.tar.gz -C .
```

Add this path to the system variables.

```
$ export PATH=~/HLF/fabric/bin:$PATH
```

Edit the core.yaml present in ~/HLF/fabric/config

```
$ atom config/core.yaml
```

Change the following values

```
peer.id: poseidon.consortium.pt
peer.networkId: production
peer.chaincodeListenAddress: 0.0.0.0:7052
peer.chaincodeAddress: poseidon.alpha.olympus.pt:7052 (uncomment)
peer.address: poseidon.alpha.olympus.pt:7051 (uncomment)
```

```
gossip.bootstrap (all peers of these org): zeus.alpha.olympus.pt:7051
poseidon.alpha.olympus.pt:7051 | hera.beta.olympus.pt:7051 demeter.beta.olympus.pt:7051
gossip.endpoint: poseidon.alpha.olympus.pt:7051
gossip.externaEndpoint: poseidon.alpha.olympus.pt:7051
gossip.requiredPeerCount: 1
gossip.maxPeerCount: 1
```

```
tls.enabled: true
tls.cert.file: ../tls/poseidon-tls-cert.pem (path relative to the location of core.yaml)
tls.key.file: ../tls/poseidon-tls-key.pem (relative to core.yaml)
tls.root.file: ../msp/tlscacerts/ca-alpha-olympus-pt-7054.pem (relative to core.yaml)
tls.client.RootCAs:
  files:
    - ../msp/tlscacerts/ca-alpha-olympus-pt-7054.pem(relative to core.yaml)
tls.mspConfigPath: ../poseidon.alpha.consortium.pt/msp (relative to core.yaml)
```

```
localMspId: Alpha
```

```
state.stateDatabase: CouchDB
state.username: admin
state.password: adminpw
```

The final files of each peer can be found [here](#).

The colour code is similar to the previous, red for the organisations and green for the peers.

Create directory for Hyperledger Fabric

```
$ sudo mkdir /var/hyperledger  
$ sudo chown -R $USER:$USER /var/hyperledger
```

Execute the following commands to start the current peer node.

```
$ cd ~/HLF/fabric/bin  
$ export FABRIC_CFG_PATH=./config  
$ ./peer node start
```

# 9. Deploy the Ordering Service

## 9.1. System Genesis Block

We are going to create an ordering service with **3 nodes** that uses **etcdraft** consensus protocol. Create a folder named `tls` in the shared folder.

On each one of the orderer VMs, copy the certificate under `~/HLF/organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/tls/tls-cronus-cert.pem` to the created directory in the shared folder (`tls`).

Open the Omega Admin VM

Open a terminal and change directory

```
$ cd ~/HLF
$ mkdir fabric
$ cd fabric
```

Copy the `tls` folder in shared folder to this directory

```
$ cp -r /media/sf_shared_folder/tls .
```

Extract the nodes binaries

```
$ tar -xf /media/sf_shared_folder/nodes-hyperledger-fabric-linux64-2.4.3.tar.gz -C .
```

Edit the `configtx.yaml` file present in `~/HLF/fabric/config`

```
$ atom config/configtx.yaml
```

Edit the following values

Attention: The script that reads the file is SENSITIVE TO INDENTATION.

Organizations:

```
- &Omega
  Name: Omega
  SkipAsForeign: false
  ID: Omega
  MSPDir: ../../organizations/ordererOrganizations/omega.olympus.pt/msp
```

Policies:

```
  Readers:
    Type: Signature
    Rule: "OR('Omega.member')"
```

```
  Writers:
    Type: Signature
    Rule: "OR('Omega.member')"
```

```
  Admins:
    Type: Signature
    Rule: "OR('Omega.admin')"
```

OrdererEndpoints:

```
- cronus.omega.olympus.pt:7050
- atlas.omega.olympus.pt:7050
- rhea.omega.olympus.pt:7050
```

```
- &Alpha
```

Name: Alpha  
SkipAsForeign: false  
ID: Alpha  
MSPDir: .././organizations/peerOrganizations/alpha.olympus.pt/msp  
Policies:

Readers:  
Type: Signature  
Rule: "OR('Alpha.member')"  
Writers:  
Type: Signature  
Rule: "OR('Alpha.member')"  
Admins:  
Type: Signature  
Rule: "OR('Alpha.admin')"  
Endorsement:  
Type: Signature  
Rule: "OR('Alpha.member')"

- &Beta

Name: Beta  
SkipAsForeign: false  
ID: Alpha  
MSPDir: .././organizations/peerOrganizations/beta.olympus.pt/msp  
Policies:

Readers:  
Type: Signature  
Rule: "OR('Beta.member')"  
Writers:  
Type: Signature  
Rule: "OR('Beta.member')"  
Admins:  
Type: Signature  
Rule: "OR('Beta.admin')"  
Endorsement:  
Type: Signature  
Rule: "OR('Beta.member')"

Orderer.OrdererType: etcdraft

Remove Orderer.kafka

EtcDRaft.Consenters: change values to this:

Consenters:

- Host: conus.omega.olympus.pt  
Port: 7050  
ClientTLSCert: tls/tls-conus-cert.pem  
ServerTLSCert: tls/tls-conus-cert.pem  
- Host: rhea.omega.olympus.pt  
Port: 7050  
ClientTLSCert: tls/tls-rhea-cert.pem



```
ServerTLSCert: tls/tls-rhea-cert.pem
- Host: atlas.omega.olympus.pt
Port: 7050
ClientTLSCert: tls/tls-atlas-cert.pem
ServerTLSCert: tls/tls-atlas-cert.pem
```

Add this profile to the rest of the profiles

```
OlympusGenesis:
  <<: *ChannelDefaults
  Orderer:
    <<: *OrdererDefaults
    Organizations:
      - *Omega
    Capabilities:
      <<: *OrdererCapabilities
  Consortiums:
    SampleConsortium:
      Organizations:
        - *Alpha
        - *Beta
  Application:
    <<: *ApplicationDefaults
    Organizations:
      - *Alpha
      - *Beta
    Capabilities: *ApplicationCapabilities
```

The final result can be seen [here](#).

After edit the configtx.yaml

Open a terminal in ~/HLF/fabric/bin

(If you need to create a new genesis block because you made some mistake or want to change some information, you will need to remove the folder /var/hyperledger/production)

Create the Genesis Block

```
$ PROFILE=OlympusGenesis
$ export FABRIC_CFG_PATH=../config
$ ./configtxgen -profile $PROFILE -outputBlock ../config/system-genesis-block/genesis.block
--channelID main-channel
```

Copy the folder ~/HLF/fabric/config/system-genesis-block to the shared folder.

```
$ cp -r ~/HLF/fabric/config/system-genesis-block /media/sf_shared_folder
```

## 9.2. Deploy the Ordering Nodes

In each of the orders

Open a terminal in ~/HLF

```
$ mkdir fabric
```

```
$ cd fabric
```

Extract the zip in shared folder

```
$ tar -xf /media/sf_shared_folder/nodes-hyperledger-fabric-linux-amd64-2.4.3.tar.gz -C .
```

Copy the system-genesis-block from the shared folder to ~/HLF/fabric/config.

```
$ cd config
$ cp -r /media/sf_shared_folder/system-genesis-block .
```

Edit the orderer config file

```
$ atom orderer.yaml
```

Edit the following values

```
General.ListenAddress:0.0.0.0
```

```
TLS.Enabled: true
```

```
TLS.PrivateKey:../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/tls/tls-cronus-key.pem
```

```
TLS.Certificate:../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/tls/tls-cronus-cert.pem
```

```
TLS.RootCAs:
```

```
-../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem
```

```
Cluster.BootstrapFile: system-genesis-block/genesis.block
```

```
Cluster.LocalMSPDir:../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/msp/
```

```
Cluster.LocalMSPID: Omega
```

```
Operations.ListenAddress: 0.0.0.0:8443
```

```
Operations.TLS.Enabled: true
```

```
Operations.TLS.Certificate:../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/tls/tls-cronus-cert.pem
```

```
Operations.TLS.PrivateKey:../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/tls/tls-cronus-key.pem
```

```
Operations.TLS.ClientAuthRequired: true
```

```
Operations.ClientRootCAs:[../../organizations/ordererOrganizations/omega.olympus.pt/orderers/cronus.omega.olympus.pt/msp/cacerts/ca-omega-olympus-pt-7055.pem]
```

The orderer.yaml file of Rhea and Atlas are similar, just change the green words to Rhea and Atlas respectively.

The orderer.yaml files of each orderer can be seen [here](#).

### 9.3. Start the Ordering Service

You must complete the previous step on all three orderers before proceeding.

Using the terminal previous in ~/HLF/fabric/bin in each orderer execute the following commands

```
$ export PATH=~/.HLF/fabric/bin:$PATH
$ sudo mkdir /var/hyperledger
$ sudo chown -R $USER:$USER /var/hyperledger
$ export FABRIC_CFG_PATH=../config
```

The next command must be done at the same time in all the 3 orderers

```
$ ./orderer start
```

(If you need to start the system again remove the folder /var/hyperledger/production in every orderer)

# 10. Join Peers to the Channel

The ordering service must be running (i.e. cronus, atlas and rhea virtual machines running and executing the ./orderer script)

In each peer

Confirm if the peer is running ./peer node start

Open a new terminal and fetch the config block from the order service (you can use any one of the orderers instead of cronus)

```
$ cd ~/HLF/fabric/bin/  
$ export FABRIC_CFG_PATH=../config/  
$ ./peer channel fetch config ../config/genesis_block.pb -o cronus.omega.olympus.pt:7050 -c  
main-channel --tls --cafile  
../..../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-  
7054.pem
```

The peer channel join must be done by the admin (although the admin has a dedicated VM, in fact he is just an identity). You will need to copy the admin msp to each one of the peer VMs (in real life this could be done using a pen or a secure channel)

Create a directory in the shared folder for both alpha admin and beta admin (name the folders alpha\_admin and beta\_admin).

Open Alpha-admin VM and Beta-admin VM

In alpha admin VM copy the admins folder admins under

```
~/HLF/organizations/peerOrganizations/alpha.olympus.pt to the alpha admin directory in the shared folder  
$cp ~/HLF/organizations/peerOrganizations/alpha.olympus.pt/admins  
/media/sf_shared_folder/alpha_admin
```

In beta admin VM copy the admin folder admins under

```
~/HLF/organizations/peerOrganizations/beta.olympus.pt to the beta admin directory in the shared folder  
$cp ~/HLF/organizations/peerOrganizations/beta.olympus.pt/admins  
/media/sf_shared_folder/beta_admin
```

In each peer copy the admins folder of his organization to the folder

```
~/HLF/organizations/peerOrganizations/alpha.olympu.pt/
```

E.g.: zeus VM



In each peer

Open a terminal

```
$ cd ~/HLF/fabric/bin
```

```
$ export FABRIC_CFG_PATH=../config
```

```
$ export
```

```
CORE_PEER_MSPCONFIGPATH=~/.HLF/organizations/peerOrganizations/alpha.olympus.pt/ad  
mins/admin.alpha.olympus.pt/msp
```

(Red: Name of the organisation to which the peer belongs to)

Join the peer to the channel

```
$ ./peer channel join -b ../config/genesis_block.pb
```

Confirm if the previous command was successful

```
$ ./peer channel list
```

# 11. Write the Chaincode

“Chaincode is a program, written in Go, Node.js, or Java that implements a prescribed interface. Chaincode runs in a separate process from the peer and initialises and manages the ledger state through transactions submitted by applications.

A chaincode typically handles business logic agreed to by members of the network, so it is similar to a “smart contract”. A chaincode can be invoked to update or query the ledger in a proposal transaction.”

(<https://hyperledger-fabric.readthedocs.io/en/release-2.0/chaincode4ade.html>)

In this tutorial, we are going to use the Go programming language. To deploy the chaincode, all the organisations in the channel must approve it, that’s why the chaincode can be called a “Smart Contract”

You can choose to write your own code or use one of the given examples, my master thesis chaincode [master thesis chaincode](#) that is an adaptation of the example given by HyperLedger Tutorial or the [HLF example](#) itself.

The following steps refer to my chaincode, but with a few adaptations you can use them to any code.

Open one of the peers (E.g.: Demeter)

Open a terminal

```
$ mkdir chaincode
```

```
$ cd chaincode
```

Write the code in a file called chaincode.go

```
$ touch chaincode.go
```

```
$ atom chaincode.go
```

Save and exit the file

# 12. Deploy the Chaincode

## 12.1. Package

Using the same VM used to write the code

Open a terminal

```
$ cd ~/HLF/chaincode
```

Define the Label that will be used across organizations (ex: occv1)

Create the file metadata.json

```
$ touch metadata.json
```

Add {"Path":"chaincode","Type":"golang","Label":"occv1"} to metadata.json

```
$ atom metadata.json
```

Init the code, fetch dependencies and test syntax

```
$ go mod init chaincode.go
```

```
$ go test
```

Package the chaincode:

```
$ cd ~/HLF/fabric/bin/
```

```
$ export FABRIC_CFG_PATH=./config/
```

```
$ ./peer lifecycle chaincode package ../chaincode.tar.gz --path ../chaincode --lang golang --label occv1
```

Copy ~/HLF/chaincode.tar.gz to the shared folder

```
$ cp ~/HLF/chaincode.tar.gz /media/sf_shared_folder
```

## 12.2. Install

In every peer:

Open a terminal and modify permissions of /var/run/docker.sock

```
$ sudo setfacl --modify user:$USER:rw /var/run/docker.sock
```

Using just one peer VM of each organization (ex:zeus and hera)

Open a terminal

```
$ cd ~/HLF
```

Copy the chaincode.tar.gz from the shared folder to the HLF directory

```
$ cp /media/sf_shared_folder .
```

The admin of the organisation need to install the code in each peer

Alpha admin will install the code in zeus and poseidon (using zeus VM)

Beta admin will install the code in hera and demeter (using hera VM)

In zeus and hera VMs save the admins msp information in system variables. Substitute the red word by beta in hera VM.

```
$ cd ~/HLF/fabric/bin
```

```
$ export FABRIC_CFG_PATH=./config/
```

```
$ export CORE_PEER_TLS_ENABLED=true
```

```
$ export CORE_PEER_LOCALMSPID="Alpha"
```

```
$ export
```

```
CORE_PEER_MSPCONFIGPATH=~/.HLF/organizations/peerOrganizations/alpha.olympus.pt/admins/admin.alpha.olympus.pt/msp/
```

```
$ export  
CORE_PEER_TLS_ROOTCERT_FILE=~/.HLF/organizations/peerOrganizations/alpha.olympus.p  
t/msp/tlscacerts/ca-alpha-olympus-pt-7054.pem
```

The package id will be the same in both organizations, so you just need to save the id returned by the first of the following commands.

In Zeus VM:

```
$ ./peer lifecycle chaincode install ../chaincode.tar.gz --peerAddresses  
"zeus.alpha.olympus.pt:7051" --tlsRootCertFiles  
../organizations/peerOrganizations/alpha.olympus.pt/msp/tlscacerts/ca-alpha-olympus-pt-7054.  
pem
```

Write the package id: \_\_\_\_\_

package ID e.g.: 'occv1:0f464f7c936675b365f456e89e607693c4916b46b16d69c5614fbf4ca55ae479'

```
$ ./peer lifecycle chaincode install ../chaincode.tar.gz --peerAddresses  
"poseidon.alpha.olympus.pt:7051" --tlsRootCertFiles  
../organizations/peerOrganizations/alpha.olympus.pt/msp/tlscacerts/ca-alpha-olympus-pt-7054.  
pem
```

In Hera VM:

```
$ ./peer lifecycle chaincode install ../chaincode.tar.gz --peerAddresses  
"hera.beta.olympus.pt:7051" --tlsRootCertFiles  
../organizations/peerOrganizations/beta.olympus.pt/msp/tlscacerts/ca-beta-olympus-pt-7054.pe  
m
```

```
$ ./peer lifecycle chaincode install ../chaincode.tar.gz --peerAddresses  
"demeter.alpha.olympus.pt:7051" --tlsRootCertFiles  
../organizations/peerOrganizations/beta.olympus.pt/msp/tlscacerts/ca-beta-olympus-pt-7054.pe  
m
```

## 12.3. Approve for My Organisation

Using the same 2 VMs used to install the code, approve the code for both organisations. It's very important that every value is exactly the same, otherwise, the package will not be approved.

Using the same terminal that you used for installation

It's possible to use any orderer, this tutorial used Cronus

Execute this command in one peer of each org (zeus and hera)

The parameters **channelID** (the one you saved in the previous step), **name**, **version**, **sequence** and **signature policy** must be exactly the same.

In both peer (zeus and hera) VM execute the following command once:

```
$ ./peer lifecycle chaincode approveformyorg -o cronus.omega.olympus.pt:7050 --tls --cafile  
../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-  
7054.pem --channelID main-channel --name occv1 --version 1.0 --init-required --package-id  
occv1:0f464f7c936675b365f456e89e607693c4916b46b16d69c5614fbf4ca55ae479 --sequence 1  
--signature-policy "OR ('Alpha.member','Beta.member')"
```

## 12.4. Commit

Before commit the code, check if everything is in order to commit the code

In one of the two machines that you used for installing run the following command

```
$ ./peer lifecycle chaincode checkcommitreadiness -o rhea.omega.olympus.pt:7050 --channelID
main-channel --tls --cafile
.../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-
7054.pem --name occv1 --version 1.0 --init-required --sequence 1 --signature-policy "OR
('Alpha.member','Beta.member')"
```

If it's all good you should read the following message:

```
Chaincode definition for chaincode 'occ', version '1.0', sequence '1' on channel 'main-channel'
approval status by org:
Alpha: true
Beta: true
```

Commit the code

After all the peer organisations had approved the code (in this case alpha and beta), **one** peer commits the chaincode.

Choose one of the peers that you used for installation of the code (E.g.:zeus)

Execute the following command:

```
$ ./peer lifecycle chaincode commit -o ctronus.omega.olympus.pt:7050 --channelID main-channel
--name occv1 --version 1.0 --sequence 1 --init-required --signature-policy "OR
('Alpha.member','Beta.member')" --tls --cafile
.../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-
7054.pem --peerAddresses zeus.alpha.olympus.pt:7051 --tlsRootCertFiles
.../organizations/peerOrganizations/alpha.olympus.pt/msp/tlscacerts/ca-alpha-olympus-pt-7054.
pem --peerAddresses poseidon.alpha.olympus.pt:7051 --tlsRootCertFiles
.../organizations/peerOrganizations/alpha.olympus.pt/msp/tlscacerts/ca-alpha-olympus-pt-7054.
pem --peerAddresses hera.beta.olympus.pt:7051 --tlsRootCertFiles
.../organizations/peerOrganizations/beta.olympus.pt/msp/tlscacerts/ca-beta-olympus-pt-7054.pe
m --peerAddresses demeter.beta.olympus.pt:7051 --tlsRootCertFiles
.../organizations/peerOrganizations/beta.olympus.pt/msp/tlscacerts/ca-beta-olympus-pt-7054.pe
m
```



## 13. Test Network

Before testing the network, is required to init by creating the first transaction.

In one of the peers (ex: poseidon)

Using the terminal opened in ~/HLF/fabric/bin

You can use any of the orderers (in this case we used atlas)

```
$ ./peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile  
../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-  
7054.pem -C main-channel -n occv1 --isinit -c '{"Args":["InitLedger","{}"]}'
```

To confirm the process, open a web browser in any other peer (E.g.: demeter) use the url

[http://localhost:5984/\\_utils/#login](http://localhost:5984/_utils/#login)

Credentials admin:adminpw

Refresh the database page, main-channel\_occ1 will appear with 2 Docs

Open and see the first two transactions.

Congratulations!! The Blockchain is done!!

Examples of invocation of functions

To verify open [http://localhost:5984/\\_utils/#login](http://localhost:5984/_utils/#login) , (credentials admin:adminpw) in any peer

Invoke functions using the following commands in any of the peers (E.g.:hera)

```
$ cd ~/HLF/fabric/bin  
$ export FABRIC_CFG_PATH=../config/
```

Function #1:

Add info to the BC:

```
$ ./peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile  
../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-  
7054.pem -C main-channel -n occv1 -c  
'{"Args":["CreateAsset","00001","00001","Commercial+Health","d04b98f48e8f8bcc15c6ae5ac050  
801cd6dcfd428fb5f9e65c4e16e7807340fa"]}'
```

Function #2:

Read one asset:

```
$ ./peer chaincode invoke -o rhea.omega.olympus.pt:7050 --tls --cafile  
../organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-  
7054.pem -C main-channel -n occv1 -c '{"Args":["ReadAsset","00001"]}'
```