

1 2 9 0



UNIVERSIDADE D
COIMBRA

João Nuno Videira Rodrigues

**NOVOS MODELOS DE PRIVACIDADE PARA A
INTERNET DAS COISAS**

Dissertação no âmbito do Mestrado Integrado de Engenharia
Eletrotécnica e de Computadores orientado pelo Professor Doutor Jorge
Miguel Sá Silva e apresentada ao departamento de Engenharia
Eletrotécnica.

Junho de 2022

Agradecimentos

A presente dissertação só foi possível graças à cooperação e apoio de várias pessoas, às quais este espaço se dedica.

Primeiramente, gostaria de agradecer aos meus Pais e irmão pelo apoio, paciência e compreensão que demonstraram no decorrer deste longo trajeto. Acima de tudo quero agradecer o esforço feito para que tivesse a oportunidade de continuar o meu percurso.

Seguidamente, agradeço ao meu orientador Professor Jorge Miguel Sá Silva, pela oportunidade de poder participar neste projeto e pelo apoio demonstrado ao longo deste trajeto.

Posteriormente, agradeço ao meu orientador oficioso Mestre Jorge Eduardo Rivadeneira Muñoz, pelo constante apoio e acompanhamento durante toda a dissertação.

Finalmente, agradeço ao Professor José Ernesto Jiménez Merino pela sua ajuda com *Ethereum*.

Abstract

Nowadays with the increasing use of *Internet of Things* (IoT) devices, there are more vulnerabilities, many of them are unknown. All this amplifies the importance and need for security systems. Taking that into account, the topic of security and privacy in IoT systems is very prevailing and interesting.

This dissertation goes through the implementation and integration of innovative privacy and security mechanisms to be tested in the ISABELA and BATINA systems.

For this, it was necessary to acquire knowledge related to the underlying technologies used in the systems. Additionally, it was important to understand the technologies used to implement the security mechanisms, the state of the art, and the ISABELA and BATINA systems.

In conclusion, with this knowledge, it was possible to start the implementation of privacy mechanisms using blockchain technology

Resumo

Atualmente com o aumento do uso de dispositivos IoT, há cada vez mais vulnerabilidades, sendo muitas destas desconhecidas. Tudo isto amplifica a importância e a necessidade de sistemas de segurança, vindo daí a escolha do tema da segurança e privacidade em sistemas IoT.

Esta dissertação passa pela implementação e integração de mecanismos de privacidade e segurança inovadores que serão posteriormente testados nos sistemas ISABELA e BATINA.

Para isso foi necessário adquirir conhecimentos relativos a tecnologias subjacentes aos sistemas. Adicionalmente, foi imperativo o conhecimento das tecnologias usadas para implementar os mecanismos de segurança, do estado de arte e dos sistemas ISABELA e BATINA.

Concluindo, com a obtenção destes conhecimentos foi possível iniciar a implementação de mecanismos de privacidade fazendo uso da tecnologia de *blockchain*.

Índice

1. Introdução	1
1.1 Contexto	1
1.2 Objetivos	1
1.3 Estrutura do Documento	1
2. Metodologia	2
2.1 Grupo de Trabalho	2
2.2 Tecnologias Usadas	2
2.2.1 Android	2
2.2.2 Dash	2
2.2.3 Xamarin	3
2.2.4 Hyperledger Fabric	4
2.2.5 Multipass	5
2.3 ISABELA/BATINA	5
2.4 Planejamento	7
3 Estado da Arte	8
3.1 Plataformas Semelhantes	8
3.2 Grupos de Trabalho	8
3.3 Análise Comparativa	8
4 Segurança e Privacidade	9
4.1 Segurança da ISABELA	9
4.2 Arquitetura do Projeto	9
5 Detalhes de Implementação	11
5.1 Xamarin	11
5.2 Hyperledger Fabric	11
5.2.1 Sistemas operativos usados	11
5.2.1.1 Linux Mint 20.3 Una	11
5.2.1.2 Multipass – Ubuntu	12
5.2.2 Pré-requisitos	14
5.2.2.1 Git	15
5.2.2.2 cURL	15
5.2.2.3 Docker e Docker-Compose	16
5.2.2.4 Documentos Fabric-Samples	19
5.2.2.5 Node.JS e NPM	22
5.2.2.6 GO	23
5.2.2.7 jq	23
5.2.2.8 Github Desktop	23
5.2.2.9 Subversion	25
5.2.2.10 yq	26
5.2.3 Criação, implementação e execução de diferentes configurações de rede	26
5.2.3.1 Implementação inicial de rede com três organizações	26
5.2.3.2 Implementação fazendo uso de scripts de fabric-samples com chaincode modificado	30
5.2.3.3 Implementação de uma rede com quatro organizações sem scripts	40

5.2.3.4	Criação de <i>scripts</i> para conceber uma rede com quatro organizações	63
5.2.3.5	Adicionar uma organização a uma rede em funcionamento	68
5.2.3.6	Implementação de uma rede a correr em várias máquinas	77
5.2.3.7	Criação de um script para gerar redes consoante o número de Peers e Organizações são definidos pelo utilizador	96
5.3	Ethereum-Geth	103
5.3.1	Sistema operativo	103
5.3.1.1	Multipass – Ubuntu	103
5.3.2	Pré-requisito	103
5.3.2.1	Geth	103
5.3.3	Implementação de uma rede privada com dois peers em máquinas diferentes	103
5.4	Avaliações	105
6	Trabalho	106
6.1	Trabalho Realizado	106
6.2	Trabalho Futuro	107
	Referências	108

Índice de Figuras

Figura 1 - Esquema do Xamarin [4]	3
Figura 2 - Exemplo de Rede Hyperledger Fabric [6]	4
Figura 3 - Arquitetura da ISABELA [8]	6
Figura 4 - Tabela do trabalho a realizar	7
Figura 5 - Arquitetura da ISABELA na vertente de segurança	10
Figura 6 - Snapshot da webpage onde se realizou o download de Linux Mint [18]	12
Figura 7 - Snapshot da webpage onde foi feito download de Multipass [19]	13
Figura 8 - Listagem de comandos Multipass	14
Figura 9 - Instalação de Git	15
Figura 10 - Instalação de cURL	15
Figura 11 - Verificação da versão de cURL	16
Figura 12 - Desinstalação de possíveis artefactos de Docker	16
Figura 13 - Instalação de Docker	17
Figura 14 - Configuração da inicialização de Docker no arranque da máquina	17
Figura 15 - Atribuição de permissões ao utilizador	17
Figura 16 - Verificação da versão de Docker	18
Figura 17 - Instalação de Docker-Compose	18
Figura 18 - Atribuição de permissões a Docker-Compose	18
Figura 19 - Verificação da versão de Docker-Compose	18
Figura 20 - Instalação de Fabric-Samples	20
Figura 21 – Encerramento da rede de exemplo	21
Figura 22 - Inicialização da rede de exemplo	22
Figura 23 - Instalação de NPM e Node.JS	23
Figura 24 - Verificação da versão de Go	23
Figura 25 - Snapshot da webpage de instalação de Github Desktop [23]	24
Figura 26 - Snapshot da página de Github Desktop	25
Figura 27 - Instalação de Subversion	26
Figura 28 - Ficheiro crypto-config.yaml	27
Figura 29 - Comando cryptogen utilizado	27
Figura 30 - Excerto de configtx.yaml relativo aos perfis	28
Figura 31 - Comandos configtxgen utilizados	28
Figura 32 - Excerto de docker-compose-base.yaml relativo à terceira organização	29
Figura 33 - Excerto de ficheiro .go usado na criação da função query	29
Figura 34 - Exemplo de valor introduzido na BD utilizando a função set	30
Figura 35 - Excerto do ficheiro fabcar.go relativo à estrutura dos dados	31
Figura 36 - Excerto do ficheiro fabcar.go relativo à inicialização do ledger	31
Figura 37 - Excerto do ficheiro fabcar.go com diversas funções	32
Figura 38 - Excerto do ficheiro fabcar.go relativo à função que realiza a mudança de owner	33
Figura 39 - Excerto do final do ficheiro fabcar.go	33
Figura 40 - Excerto do ficheiro startFabric.sh relativo à inicialização da network e do chaincode	34
Figura 41 - Excerto do ficheiro startFabric.sh relativo às instruções de uso da aplicação em JavaScript	35
Figura 42 - Conjunto de excertos relativos à execução do script startFabric.sh	36
Figura 43 - Excerto do ficheiro query.js	37
Figura 44 - Excerto do ficheiro invoke.js	38
Figura 45 - Instruções apresentadas ao executar startFabric.sh	38

Figura 46 - Execução de npm install, registo de admin e criação de novo utilizador	39
Figura 47 - Execução de query.js e invoke.js	39
Figura 48 - Estrutura de diretorias criada	40
Figura 49 - Estrutura da diretoria jnvr-network	40
Figura 50 - Ficheiro crypto-config.yaml	41
Figura 51 - Execução do comando cryptogen generate	42
Figura 52 - Excerto do ficheiro configtx.yaml relativo à identificação da organização 1	42
Figura 53 - Excerto do ficheiro configtx.yaml relativo à definição das capacidades do canal	43
Figura 54 - Excerto do ficheiro configtx.yaml relativo à definição da secção da aplicação	43
Figura 55 - Excerto do ficheiro configtx.yaml relativo à definição dos orderers	44
Figura 56 - Excerto do ficheiro configtx.yaml relativo à definição do canal	45
Figura 57 - Excerto do ficheiro configtx.yaml relativo à definição das políticas do canal	45
Figura 58 - Criação do bloco de génese	46
Figura 59 - Criação do canal channel1	46
Figura 60 - Criação dos anchor peers em todas as organizações	47
Figura 61 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir um peer de uma organização	47
Figura 62 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir a configuração do CA da organização 1	48
Figura 63 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir uma das bases de dados	48
Figura 64 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir configuração de cli	49
Figura 65 - Excerto do ficheiro docker-compose-base.yaml a definir um peer de uma organização	49
Figura 66 - Excerto do ficheiro peer-base.yaml a definir o comportamento de um peer	50
Figura 67 - Inicialização dos containers da rede	51
Figura 68 - Excertos da criação do canal	51
Figura 69 - Excertos da junção de uma organização ao canal	53
Figura 70 - Excertos da criação de um anchor peer numa organização	54
Figura 71 - Documento foodcontrol.go	55
Figura 72 - Documento go.mod	56
Figura 73 - Execução dos exports dentro de cli	56
Figura 74 - Criação do package com o chaincode a instalar nos peers	56
Figura 75 - Verificação da criação do package do chaincode	57
Figura 76 - Excertos da instalação do chaincode em um dos peers	58
Figura 77 - Excertos da verificação da aprovação das organizações	60
Figura 78 - Excertos do commit do chaincode	61
Figura 79 - Utilização da função set do chaincode	61
Figura 80 - Página inicial de couchdb	61
Figura 81 - Chaincode em couchdb	62
Figura 82 - Valor previamente criado, guardado em couchdb	62
Figura 83 - Detalhes do valor criado, guardado em couchdb	63
Figura 84 - Documento script-jnvr-network.sh	64
Figura 85 - Documento script_jnvr_network_docker.sh	65
Figura 86 - Excertos da execução de script_jnvr_network.sh	66

Figura 87 - Verificação da presença de script_jnvr_network_docker.sh	67
Figura 88 - Ficheiro org5-crypto-config.yaml	68
Figura 89 - Execução do comando cryptogen generate para a organização 5	68
Figura 90 - Criação de org5.json	69
Figura 91 - Excertos da criação de config_block.pb	69
Figura 92 - Representação dos blocos presentes na rede	70
Figura 93 - Excerto de modified_config.json relativo à organização 5	71
Figura 94 - Excertos da assinatura da modificação aplicada à rede	72
Figura 95 - Excerto da configuração da rede que dita ser necessário existir uma maioria para verificar alterações na rede	73
Figura 96 - Assinatura da modificação da rede por parte da organização 2	73
Figura 97 - Excertos da execução de update por parte da organização 3	74
Figura 98 - Definição das configurações do novo peer, novo couchdb e org5cli	75
Figura 99 - Inicialização dos componentes da organização 5	75
Figura 100 - Excertos do uso de peer channel fetch 0	76
Figura 101 - Erro verificado ao tentar juntar o novo peer da organização 5 ao canal	77
Figura 102 - Ficheiros presentes na directoria MultipassFolder	78
Figura 103 - Ficheiro initSetup.sh	78
Figura 104 - Inicialização das VM, criação de directorias partilhadas e listagem de máquinas existentes	79
Figura 105 - Demonstração da paragem de uma máquina multipass	80
Figura 106 - Inicialização de uma máquina multipass	80
Figura 107 - Obtenção do comando para adicionar um manager à rede	81
Figura 108 - Junção das VM 2,3,4 ao swarm criado pela máquina 1	83
Figura 109 - Listagem de máquinas ligadas a nodo	83
Figura 110 - Listagem de redes presentes a partir de cada máquina	84
Figura 111 - Listagem dos ficheiros presentes na directoria 4host-swarm-clis	85
Figura 112 - Caracterização da configuração da rede	85
Figura 113 - Inicialização dos componentes em cada peer	87
Figura 114 - Representação do funcionamento da rede	89
Figura 115 - Inicialização do canal	90
Figura 116 - Verificação das informações do blockchain a correr em cada máquina	91
Figura 117 - Geração de packages necessários para a execução do chaincode	91
Figura 118 - Instalação do chaincode em cada peer	92
Figura 119 - Aprovação do chaincode em ambas as organizações	94
Figura 120 - Verificação da aprovação do chaincode	94
Figura 121 - Aprovação do chaincode	94
Figura 122 - Inicialização de dados no ledger e invocação da função queryCar	95
Figura 123 - Testagem da funcionalidade da rede em todos os hosts	95
Figura 124 - Script makeConfigFiles.sh, inicialização dos valores de peer e organizações	96
Figura 125 - Script makeConfigFiles.sh, configuração de configtx.yaml	97
Figura 126 - Script makeConfigFiles.sh, configuração de crypto-config.yaml	98
Figura 127 - Script makeConfigFiles.sh, configuração de docker-compose-base.yaml	99
Figura 128 - Script makeConfigFiles.sh, criação dos ficheiros host<#>.yaml	100
Figura 129 - Script makeConfigFiles.sh, criação de mychannel.sh	100
Figura 130 - Script makeConfigFiles.sh, criação dos ficheiros host<#>up.sh e host<#>down.sh	101
Figura 131 - Script makeConfigFiles.sh, envio de documentos de configuração modificados para configDocs	101

Figura 132 - Execução do script makeConfigFiles.sh	102
Figura 133 - Configuração de hosts obtida utilizando makeConfigFiles.sh	102
Figura 134 - Criação de máquinas virtuais	103
Figura 135 - Inicialização da consola de geth na VM machine1	104
Figura 136 - Execução do comando admin.nodeInfo.enode	104
Figura 137 - Inicialização da consola geth na VM machine2	104
Figura 138 - Execução do comando multipass ls	104
Figura 139 - Execução do comando admin.addPeer	105
Figura 140 - Execução do comando net.peerCount	105
Figura 141 - Tabela do trabalho realizado	106

Abreviaturas

AI	<i>Artificial Intelligence</i>
BD	<i>Base de Dados</i>
CA	<i>Certificate Authority</i>
DC	<i>Data Controllers</i>
GE	<i>Generic Enabler</i>
HF	<i>Hyperledger Fabric</i>
IoT	<i>Internet of Things</i>
OS	<i>Operating Systems</i>
QRRH	<i>Query, Requests and Responses Handler</i>
SC	<i>Smart Contract</i>
VM	<i>Virtual Machine</i>

1 Introdução

1.1 Contexto

Como foi mencionado anteriormente, a IoT é cada vez mais usada, normalmente na aquisição de dados, sendo a transmissão desses dados um dos focos de interesse relativamente à segurança e privacidade.

Muito se fala na perda da privacidade, mas a cedência de dados não se faz, normalmente, sem a autorização do utilizador. Para isso usam-se formulários para apresentar os consentimentos do utilizador, formulários estes que são potenciais vulnerabilidades nos sistemas, sendo necessária a proteção da informação dos consentimentos para evitar fugas de informação e a perda de dados para entidades desconhecidas.

1.2 Objetivos

No contexto desta temática, este trabalho tem como objetivo a implementação de mecanismos de privacidade na plataforma apresentada em [1] denominada ISABELA (plataforma esta que será descrita com mais detalhe à frente), de modo a fazer a proteção dos dados dos utilizadores, através do controlo de quais entidades têm permissão de acesso aos dados dos utilizadores. Para isso ir-se-á utilizar *Hyperledger Fabric* (HF) tal como foi definido no projeto proposto. Adicionalmente pretende-se fazer uma breve introdução e implementação da possibilidade de fazer o mesmo usando *Ethereum*.

1.3 Estrutura do Documento

Este documento está dividido em seis partes, incluindo este primeiro capítulo de introdução ao trabalho. Partes estas que serão descritas de seguida:

- No capítulo 2 será feita uma descrição da metodologia, onde se explicita o grupo de trabalho, as tecnologias a serem utilizadas, o planeamento e uma análise comparativa das diferentes tecnologias existentes atualmente;
- No capítulo 3 analisar-se o estado da arte existente para dar a conhecer como outros grupos e plataformas utilizam as tecnologias no ensino;
- No capítulo 4 fala-se de segurança e privacidade, onde o foco estará na descrição da segurança da plataforma ISABELA e arquitetura do projeto;
- No capítulo 5 abordam-se todas as implementações realizadas no decorrer da dissertação. Onde se utilizaram as plataformas *Xamarin*, HF e *Ethereum*. O foco irá recair maioritariamente em HF e *Ethereum*, onde se vão dar a conhecer todas as informações necessárias para replicar as implementações. Adicionalmente estão registados alguns dos muitos problemas encontrados ao criar as implementações e como estes foram ultrapassados. No final faz-se uma avaliação comparativa das plataformas HF e *Ethereum*.
- No capítulo 6 refere-se o trabalho realizado (utilizando um gráfico de Gant) e possíveis desenvolvimentos a realizar no futuro.

2 Metodologia

2.1 Grupo de Trabalho

O grupo de trabalho em que me insiro é constituído por diversos alunos, investigadores e professores do Departamento de Engenharia Eletrotécnica e Computadores e do Departamento de Engenharia Informática. Todos trabalham nas áreas das Redes de Comunicação e da Internet das Coisas.

Este grupo de investigação tem desenvolvido o sistema ISABELA e o sistema BATINA com o objetivo de suportar métodos inovadores de aprendizagem, tirando partido das novas tecnologias nas áreas do IoT, da programação de telemóveis e da Inteligência Artificial.

2.2 Tecnologias usadas

Na implementação da dissertação existem diferentes plataformas usadas com as quais foi necessária a familiarização, incluindo as que foram utilizadas nos projetos ISABELA e BATINA, que irão ser descritas de seguida.

2.2.1 Android

Desde o lançamento do primeiro *smartphone* em 1992 até hoje, os telemóveis evoluíram bastante, tendo o seu mercado passado por diversas fases até chegar aos dias de hoje, onde existem essencialmente dois grandes Sistemas Operativos (*Operating Systems – OS*): Android e iOS.

Para esta dissertação, devido a existirem componentes já implementadas em Android nas plataformas ISABELA e BATINA, será essa a plataforma onde irá estar o foco. Apesar disto, dispositivos iOS também serão compatíveis.

2.2.2 Dash

Devido à necessidade de passar os dados recolhidos para um *dashboard* fez-se uso da plataforma *Dash* [2], que possibilitou colocar os dados reunidos, de maneira simples, à disposição do utilizador.

2.2.3 Xamarin

Devido à existência de dois OS dominantes, programadores de aplicações móveis tinham o problema de ser necessário implementar código para ambos. Para resolver essa dificuldade foram criadas diversas plataformas que produzem implementações para ambos OS de uma só vez, como por exemplo [3]:

- *Xamarin* - Criado pela *Microsoft*, utiliza linguagem *C#*, apresentando algumas limitações na migração para outras *frameworks* e com bibliotecas desatualizadas;
- *React Native* - Criado pelo *Facebook*, é necessário utilizar *JavaScript + React Native SDK*, *Kotlin + Android SDK* e *Swift + iOS SDK*. Adicionalmente é de difícil migração para outros *frameworks*;
- *Flutter* – Criado pelo *Google*, utiliza a linguagem *Dart*, criada pelo próprio *Google*, sendo necessário aprendê-la de raiz. Adicionalmente é importante ter alguns conhecimentos de *iOS / Android*, devido à possibilidade de uso de bibliotecas nativas. Em termos de migração para outras *frameworks* é necessário implementar tudo de raiz;
- *Kotlin Multiplatform* – Criado pela *JetBrains*, é uma ferramenta de partilha opcional de código entre plataformas, podendo-se escolher com grande flexibilidade o que partilhar. Esta ferramenta destina-se a *developers* de aplicações moveis que tenham uma grande compreensão de *Kotlin* (*Android*) ou *Swift* (*iOS*).

Para os projetos ISABELA e BATINA foi escolhido *Xamarin* devido à utilização de *C#*, que desta maneira oferece a possibilidade de aceder através *dessa* linguagem aos *Software Development Kits* e serviços dos OS, tanto *Android* como *iOS* através de *Xamarin.Android* e *Xamarin.iOS*, respetivamente. Para além disso ambas as *frameworks* permitem interagir com as *Base Class Libraries* de *.NET*. Usando a *framework Xamarin.Forms* é também possível criar *User Interfaces*, de forma bastante simples, apenas uma vez para ambos os OS.

Foi então necessária a familiarização com a plataforma *Xamarin*.

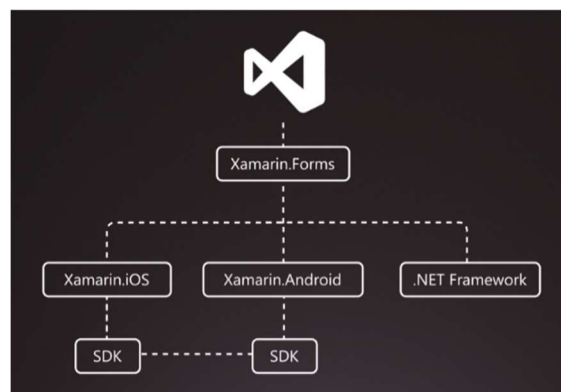


Figura 1 - Esquema do Xamarin [4]

2.2.4 Hyperledger Fabric

Devido à necessidade de fazer transações impossíveis de modificar, em 2008 foi criado o *Bitcoin* [5]. Com esta moeda digital passou a ser possível fazer transações *Peer to Peer* sem qualquer intervenção de bancos. Isto foi possível devido ao uso de criptografia nos nós da rede, de modo a verificar as transações e o consequente registo num *distributed ledger* público, ao qual se chamou *blockchain*.

Com o decorrer do tempo, apesar do *Bitcoin* e outras cripto-moedas serem bastante úteis, o facto de qualquer pessoa poder entrar e ter acesso anónimo era um grande obstáculo para a sua introdução no mundo empresarial, onde era fulcral criar ambientes com entrada controlada de utilizadores, sendo estes identificáveis. Para resolver esse problema foi criado, entre outros, o HF e passou a ser possível identificar e escolher os participantes de uma rede de *blockchain* privada, em que organizações participam e podem criar canais usando configurações de canais. Passando, deste modo, a ser impossível organizações na rede que não pertençam ao canal acederem à informação partilhada no canal.

A estrutura do HF é composta por organizações (onde é necessário usar *Certificate Authorities* [CAs] na sua génese, que emitem certificados digitais, confirmam a identidade do *owner* e fornece provas que o certificado é válido), que têm *peers* que, quando pertencem a um ou mais canais, têm um ou mais *ledgers* (regista todas as transações do canal, sendo *tamper-resistant*) para cada canal. Para além disso, alguns dos *peers* têm instalado *chaincode* (define como as organizações *peer* interagem com o *ledger*), consoante seja necessário o *peer* interagir com o *ledger*.

Adicionalmente pode-se comunicar com canais fazendo uso de aplicações que se associam a organizações e comunicam através do envio de propostas de transação para os *peers* das organizações que participam no canal.

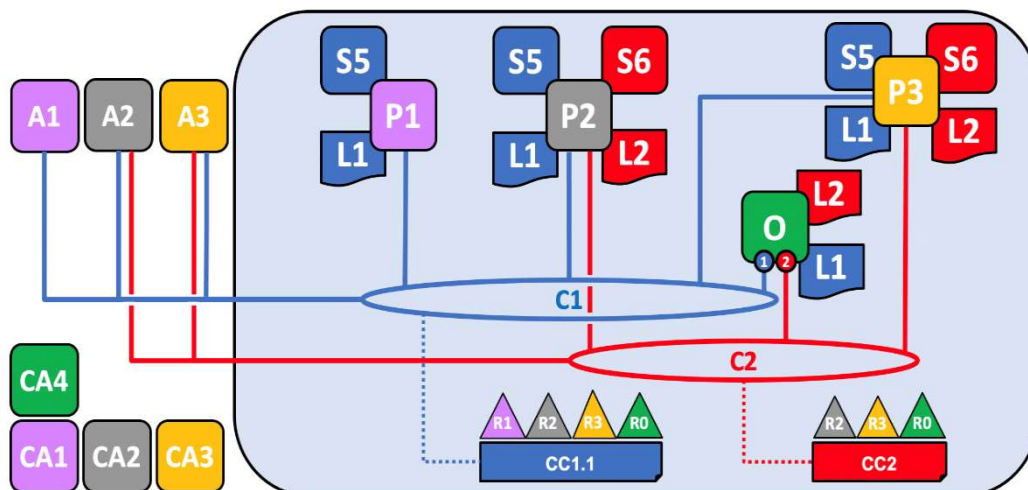


Figura 2 - Exemplo de Rede Hyperledger Fabric [6]

A Figura 2 exemplifica uma rede com quatro organizações (R0, R1, R2 e R3) criadas usando CAs (CA1, CA2, CA3, CA4), em que uma delas tem função de *orderer* ([O] onde se encontram *ledgers* dos dois canais [L1 e L2]) que cria dois canais (C1 e C2)). Ao canal um (C1) pertencem três organizações (R1, R2 e R3) (excluindo o *orderer* [R0]) que têm o *Smart Contract* (SC) e *ledger* de C1, podendo então comunicar todas no canal (assumindo que o SC assim o dita). No caso do *orderer*, apesar de ter a informação do canal no *ledger*, não pode comunicar porque não tem SC. Ao canal dois (C2) pertencem três organizações (R2, R3 e R0), ficando então R2 e R3 com dois pares de SC. Para as organizações R1, R2, R3 há as respetivas aplicações A1, A2 e A3.

2.2.5 Multipass

Tendo em conta que existe a necessidade de criação de Máquinas Virtuais (*Virtual Machines* - VM) com endereços de IP diferentes, para simular e testar as implementações criadas, utilizou-se o *Multipass* [7], uma plataforma focada na criação e gestão de VM em *Ubuntu* de maneira bastante simples.

2.2.6 Ethereum-Geth

Ethereum foi criado com o intuito de aumentar a usabilidade da tecnologia *blockchain*. Assim deixou-se de ter apenas um foco monetário e passou a ser possível criar *smart-contracts* (programa que determina o comportamento de cada *peer*) e *dApps* (aplicações descentralizadas presentes em redes de *blockchain*).

Uma das implementações do protocolo *Ethereum* fez-se usando a linguagem *Go*. Esta implementação encontra-se disponível usando o *standalone client* denominado *Geth*.

2.3 ISABELA/BATINA

Tal como foi mencionado anteriormente, a dissertação irá basear-se na implementação de mecanismos de privacidade para a plataforma ISABELA (IoT Student Advisor and Best Lifestyle Analyser), plataforma esta que recolhe informação de estudantes de modo a poder monitorizar o seu estilo de vida, com o intuito de maximizar o seu desempenho académico.

A arquitetura da ISABELA [8] faz uso de FIWARE para recolher, produzir, publicar e consumir informação, que pode ser proveniente de redes de sensores e telemóveis, utilizadores e sistemas pré-existentes. Para a recolha de dados são usadas:

- Caixas de IoT (desenvolvidas fazendo uso de *Arduino* e *Raspberry*) de modo a obter dados ambientais;
- Redes sociais com o propósito de entender melhor as preferências, relacionamentos e eventos frequentados, etc;
- Aplicações para *smartphones/smartwatches* (desenvolvidas em *Android*) tendo o intuito de averiguar a localização, horas de sono, batimento cardíaco, entre outros.

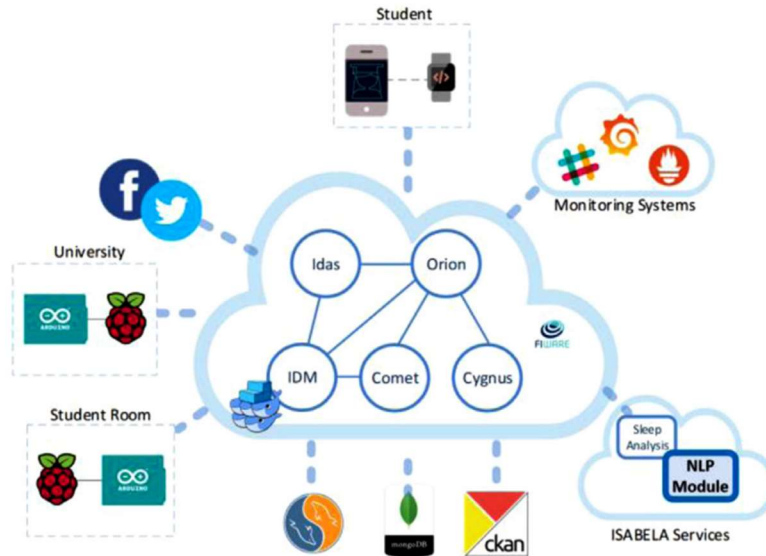


Figura 3 - Arquitetura da ISABELA [8]

No projeto foram usados diferentes *Generic Enablers* (GE) do *FIWARE project*, como se observa na Figura 3, que são descritos de seguida:

- IDAS – Usado na gestão de dispositivos IoT, através do qual se tem a capacidade de interagir entre os dispositivos e FIWARE.
- ORION – Usado na gestão das entidades na aplicação Android da ISABELA. Com esta API pode-se criar/apagar/obter/atualizar entidades. Adicionalmente, para melhorar a distribuição do sistema e a sua escalabilidade, o módulo atua como *broker*, permitindo os consumidores fazerem subscrições de dados com as entidades com regras e atributos específicos. O ORION envia notificações consoante as regras de subscrição são ativadas. Os dados provenientes de dispositivos IoT são associados com as entidades guardadas no ORION.
- Cygnus – Devido ao ORION não ser o GE indicado para gravar *historic data*, usa-se Cygnus para fazer persistir os dados. Este subscreve as entidades ORION e envia a informação para BDs (exemplo: MySQL, MongoDB e CKAN).
- STH-Comet – Tendo em conta que a maioria das BD (que recebem informação através de Cygnus) não têm APIs para aceder à sua informação usa-se este GE para poder ter acesso à informação de modo mais simples.
- IDM – Usado para proteger comunicações entre GEs, dispositivos IoT e a aplicação. Isto é feito com o adicionar de medidas de segurança e autenticação em cada um dos três casos.

Ainda na Figura 3 pode-se observar os dois serviços incluídos na ISABELA, nos quais se inclui o módulo da análise do sono, que faz o cálculo das horas dormidas através do uso dos sensores do smartphone e o módulo NLP faz processamento da linguagem das redes sociais para detetar sentimentos e emoções exprimidos.

Relativamente à plataforma BATINA, trata-se de um projeto com o intuito de fornecer um ensino de maior proximidade entre aluno e professor, tendo em conta a necessidade de privacidade individual.

Comparando ambas as plataformas, a diferença encontra-se na restrição imposta de apenas poder ser usada na sala de aula, no caso da plataforma BATINA. Além disso a plataforma BATINA é uma versão simplificada da ISABELA.

2.4 Planeamento

Devido à necessidade do projeto ISABELA ter acesso a informação pessoal dos utilizadores, quer seja através dos telemóveis ou das caixas IoT, é de extrema importância a procura de mecanismos de segurança que garantam a proteção dos dados segundo o Regulamento Geral de Proteção de Dados. Tendo isso em conta, nesta dissertação pretende-se fazer uso dos canais de HF para privatizar a troca de informação e encontrar outros mecanismos de segurança para conseguir criar um módulo de privacidade que será posteriormente integrado no projeto ISABELA. Adicionalmente pretende-se explorar a possibilidade de criar estes canais seguros fazendo uso de *Ethereum*.

	Março			Abril				Maio				Junho				Julho		
Semanas:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Tarefas de Implementação em Hyperledger Fabric	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█		
Familiarização e implementação com Ethereum																	█	█
Escrita da Dissertação			█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█

Figura 4 - Tabela do trabalho a realizar

3 Estado da arte

3.1 Plataformas Semelhantes

Dentro do mundo do ensino inovador, existem diferentes projetos que desenvolveram plataformas bastante interessantes, alguns dos quais irão ser descritos:

- *OpenCast* [9],[10] – Ferramenta *open-source* que permite partilhar, editar, gravar e fazer *upload* de vídeos, algo que é bastante útil tanto para alunos como professores. Adicionalmente, a vertente mais interessante passa por se poder integrar esta ferramenta em diversas ferramentas distintas, entre as quais se destaca a possível integração com *Matomo*, uma plataforma de análise da web que fornece informação dos utilizadores, como por exemplo a linguagem do utilizador, documentos/vídeos vistos ou que fizeram download, páginas que mais gostaram, etc.
- *BigBlueButton* [11],[12] – Plataforma de ensino *open-source* onde, por exemplo, é possível integrar o uso de múltiplos *whiteboards* para professor e aluno poderem interagir diretamente.
- *Aleks* [13] - Sistema de Inteligência Artificial (*Artificial Intelligence* – AI) usado para determinar o conhecimento dos alunos e ajudar os alunos a aprenderem os tópicos para os quais o sistema determina estarem prontos.

3.2 Grupos de Trabalho

Existem vários grupos de trabalho focados no uso das novas tecnologias para o ensino, nomeadamente o grupo coordenado por Kelly Parke [14] que numa das disciplinas que leciona propôs aos seus alunos encontrarem maneiras de melhorar o ensino fazendo uso de AI. Para além desse exemplo existe o *SthemBrasil* [15], onde um conjunto de Instituições do Ensino Superior desenvolvem trabalho conjunto para implementar inovações relativamente à qualidade do ensino. Um projeto a ser introduzido é o *Aleks* [16], mencionado anteriormente.

3.3 Análise Comparativa

Comparando os projetos ISABELA e BATINA aos pesquisados, o mais semelhante é o *Aleks* que usa os *inputs* dos alunos para maximizar a sua aprendizagem fazendo uso de AI. A diferença nesse caso é a utilização de AI enquanto na ISABELA é usado o processamento de linguagem natural proveniente das redes sociais. Adicionalmente *Aleks*, tal como o BATINA, apenas recolhe os seus dados na sala de aula.

4 Segurança e Privacidade

4.1 Segurança da ISABELA

Na versão corrente da ISABELA descrita em [1],[8], a segurança prende-se na utilização de técnicas de anonimização e agregação de dados, como por exemplo o uso de *hashes* para anonimizar dados relativos às chamadas feitas e mensagens enviadas. Para além disso alguma da informação trocada entre o utilizador e a entidade de atuação é guardada numa Base de Dados (BD) persistente nos smartphones, que é eliminada quando se desinstala a aplicação.

Adicionalmente emprega-se outro mecanismo de anonimização para o armazenamento de dados sensíveis, onde se utiliza o mecanismo de autenticação do Facebook para obter *hashed access tokens* dos seus servidores, que será novamente *hashed* utilizando o algoritmo SHA256.

Informação como a localização do GPS e o ID do WI-Fi estão personalizados para anonimizar os seus valores. No caso do GPS utiliza-se a distância percorrida e a velocidade média. Enquanto o ID do Wi-Fi é substituído pelo número de estudante.

Apesar de tudo, a plataforma ainda precisa de ter um sistema para notificar o utilizador dos processos de privacidade utilizados pelos serviços IoT, tal como dar autorização para *Data Controllers* (DC) poderem aceder às BD, sistema este que será o objetivo da dissertação.

4.2 Arquitetura do Projeto

Durante a realização da dissertação foram estudadas diversas formas de implementar um canal que permita o envio, de forma segura, da informação dos consentimentos necessários para o acesso aos dados do utilizador, para a aplicação desenvolvida.

Uma das possibilidades estudadas (que acabou por não ser implementada, devido à falta de tempo) introduzida pelo Prof. Ernesto Jiménez, foi o uso de *blockchain* com *Etherium*. Apesar de ser uma tecnologia de *blockchain permissionless* (em que qualquer utilizador pode entrar e participar), ir-se-ia restringir o acesso ao canal com a definição de todos os utilizadores no bloco inicial do *blockchain* (bloco de génese) e com a encriptação do conteúdo, tendo apenas os utilizadores presentes no bloco de génese acesso a uma ou várias chaves privadas que iriam encriptar/desencriptar as mensagens. Caso fosse necessário adicionar utilizadores depois da inicialização do *blockchain* pensou-se em criar um subconjunto de nodos, que utilizaria Proof of Stake (normalmente funciona em redor de *cryptocurrencies*, mas neste caso poder-se-ia atribuir artificialmente um determinado valor de *cryptocurrency*, para a decisão de entrada ser realizada consoante a decisão da maioria dos participantes), de modo a permitirem a adição de novos utilizadores e atribuíssem uma chave privada para poder aceder ao conteúdo e outra chave privada para encriptar as mensagens a enviar. Adicionalmente, o subconjunto de nodos poderia conceder/retirar permissões de escrita no canal.

A outra possibilidade prendeu-se no uso de HF, já descrito em 2.2.4, tendo sido esta a alternativa usada para implementar o canal seguro.

Realizou-se então a integração dos módulos de privacidade na arquitetura já existente da ISABELA de maneira semelhante aos módulos já existentes em Docker containers usando-se HF.

Ou seja, do *Consent Manager* da Figura 5, utiliza-se o *Query, Requests and Responses Handler* (QRRH) para passar quem precisa de dar consentimento, que será enviado para a aplicação usando HF no *Data Aquisition Module* que, depois de receber o consentimento de volta da aplicação, vai passar para o *Data Dispatcher Module*, onde vai ser armazenado o novo utilizador que deu consentimento. Finalmente os DC utilizam o QRRH para ir buscar ao *Data Dispatcher Module*, sendo este processo controlado pela *Privacy Enforcement Bridge*.

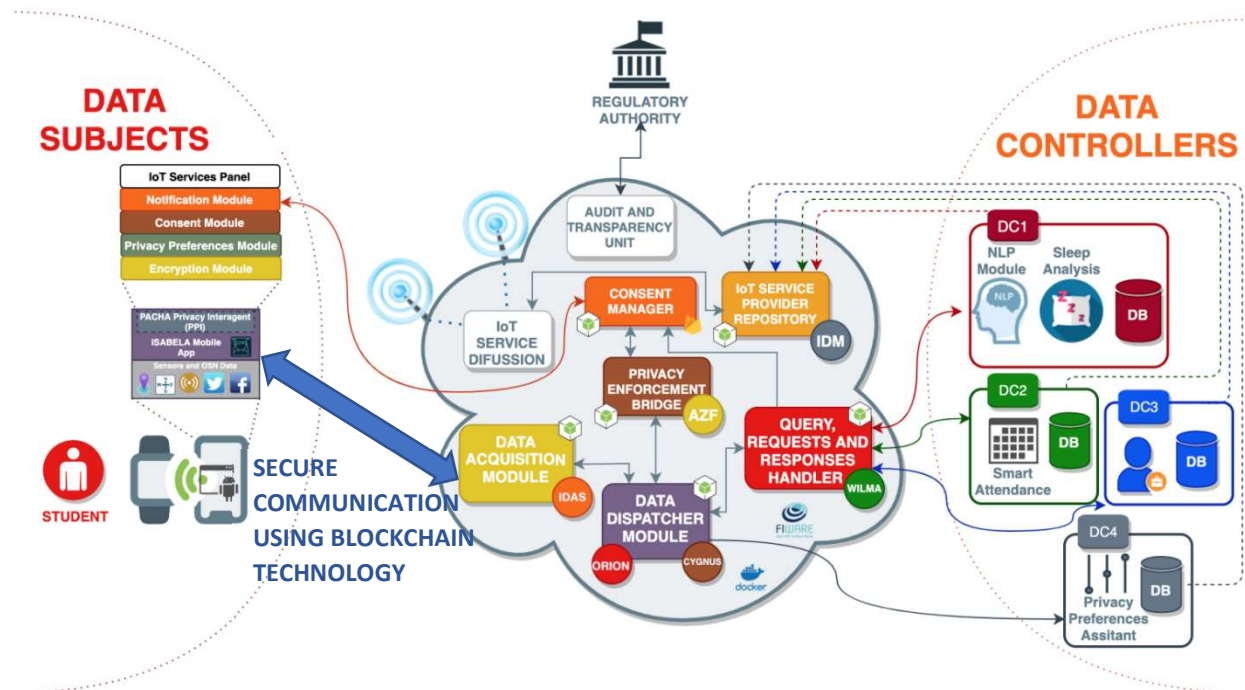


Figura 5 - Arquitetura da ISABELA na vertente de segurança

5 Implementação

Durante a dissertação foram realizadas diversas implementações que irão ser descritas detalhadamente neste capítulo.

5.1 Xamarin

No caso desta plataforma, foi realizado o tutorial de como criar aplicações móveis com o Xamarin.Forms, como se pode ver em [17]. Na implementação destes exemplos foi necessário rever C# e aprender um pouco de .NET, de modo a modificar algumas partes do tutorial que se encontravam desatualizadas.

5.2 Hyperledger Fabric

As implementações realizadas, os sistemas operativos utilizados e os pré-requisitos necessários para implementar em HF serão descritos passo a passo de seguida. Adicionalmente, é revelante ter em conta que foi bastante difícil colocar qualquer uma das implementações apresentadas a funcionar, isto deveu-se a esta tecnologia estar em continuo desenvolvimento, o que levou à desatualização da documentação. Adicionalmente, a relativa obscuridade de HF levou à escassez de recursos para ajudar a ultrapassar os problemas encontrados.

5.2.1 Sistemas operativos

5.2.1.1 Linux Mint 20.3 Una

Nas implementações que serão descritas mais à frente, com apenas um *host* e na criação dos ficheiros de configuração usa-se uma VM de *Linux Mint 20.3 Una*, versão *Cinnamon*, com o software de virtualização *VMware Workstation 16 Player*.

Foi feito download da imagem de *Linux Mint* em [18] (na data de escrita a referência encontra-se indisponível).

Linux Mint

Linux Mint

Linux Mint distribution is based on Ubuntu, it offers more complete out-of-the-box experience by including useful stuff like browser media codecs, DVD playback, plugins for browser, java and other components. The purpose of Linux Mint is to produce a modern, elegant and comfortable operating system which is both powerful and easy to use. It's both free of cost and open source, it provides about 30,000 packages and one of the best software managers.

It's safe and reliable. Thanks to a conservative approach to software updates, a unique Update Manager and the robustness of its Linux architecture, Linux Mint requires very little maintenance (no regressions, no antivirus, no anti-spyware....etc).



Support This Project



How to Setup VMDK for VMware

Credentials for images:

username: osboxes

password: osboxes.org

Root account password: osboxes.org

Linux Mint 20.3 Una

VirtualBox VMware Info

Cinnamon Version

- VMware (VMDK) 64bit [Download](#) Size: 1.55GB
SHA256: F28A19FD3AD064D081EAD55E950EAA2A0CD1D068C59C4FA78C394D68741C711

Mate Version

- VMware (VMDK) 64bit [Download](#) Size: 1.50GB
SHA256: CC90A7B71038F31CC010EA303C7C2E4A9DFD386F61BF68402E9004AEABF60

Xfce Version

- VMware (VMDK) 64bit [Download](#) Size: 1.43GB
SHA256: B65608F9E8227D4EBBA191B9C88737FC7B6306BEA26BA5CAE21E2882922D41

RECENT POSTS

[Peppermint 2022-02 VM Images Released for VirtualBox and VMware](#)

09-02-2022

[TrueNAS 12.0 U7 Virtual Machine Images Available for VirtualBox and VMware](#)

08-02-2022

[Linux Lite 5.8 Virtual Machine Images Available for VirtualBox and VMware](#)

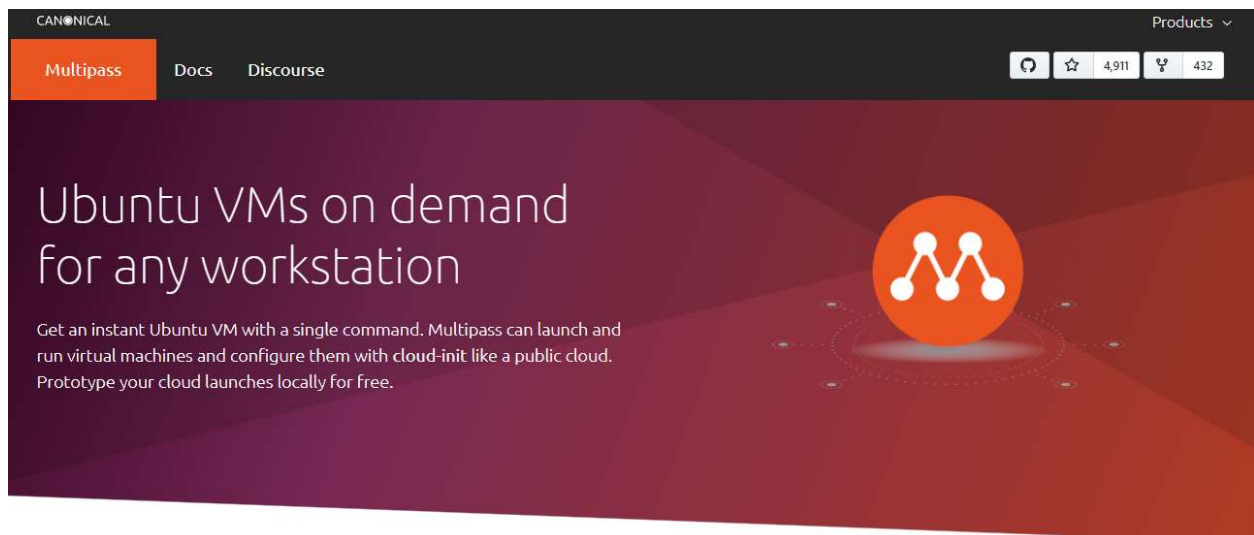
08-02-2022

Figura 6 - Snapshot da webpage onde se realizou o download de Linux Mint [18]

5.2.1.2 Multipass - Ubuntu

Como foi mencionado anteriormente, usa-se a plataforma *Multipass*, de modo a poder criar VM de *Ubuntu* para as implementações *multihost*.

Instalou-se a plataforma seguindo as instruções presentes em [19] para o respetivo sistema operativo a ser utilizado. É fulcral fazer a instalação numa rede privada de modo a evitar problemas de comunicação.



Select OS to get started



Figura 7 - Snapshot da webpage onde foi feito download de Multipass [19]

Após a instalação é possível o uso de comandos *Multipass* no terminal que permitem a criação de VM com diversas configurações, entre as quais: diferentes nomes, versões de *Ubuntu*, espaço de disco a utilizar, memória de *RAM* e *CPUs* alocados, tipo de rede, entre outros.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\2015232235> multipass
Usage: Program Files\Multipass\bin\multipass.exe [options] <command>
Create, control and connect to Ubuntu instances.

This is a command line utility for multipass, a
service that manages Ubuntu instances.

Options:
  -?, -h, --help      Displays help on commandline options.
  --help-all         Displays help including Qt specific options.
  -v, --verbose       Increase logging verbosity. Repeat the 'v' in the short
                    option for more detail. Maximum verbosity is obtained with 4
                    (or more) v's, i.e. -vvvv.

Available commands:
  alias      Create an alias
  aliases    List available aliases
  delete     Delete instances
  exec       Run a command on an instance
  find       Display available images to create instances from
  get        Get a configuration setting
  help       Display help about a command
  info       Display information about instances
  launch     Create and start an Ubuntu instance
  list       List all available instances
  mount      Mount a local directory in the instance
  networks   List available network interfaces
  purge      Purge all deleted instances permanently
  recover    Recover deleted instances
  restart    Restart instances
  set        Set a configuration setting
  shell      Open a shell on a running instance
  start      Start instances
  stop       Stop running instances
  suspend    Suspend running instances
  transfer   Transfer files between the host and instances
  umount    Unmount a directory from an instance
  unalias    Remove an alias
  version    Show version details

PS C:\Users\2015232235>
```

Figura 8 - Listagem de comandos Multipass

5.2.2 Pré-requisitos

Para poder correr as implementações foi necessária a instalação de diversos softwares, que irão ser enumerados e explicados de seguida. Nesta explicação é adquirido que se está a utilizar um sistema atualizado. Para isso corre-se o comando `sudo apt update && sudo apt upgrade`.

Adicionalmente, reitera-se que nestas implementações usou-se a rede em modo privado, de modo a ter mais permissões de acesso à comunicação, de modo a resolver erros de acesso que ocorreram durante a criação dos exemplos.

5.2.2.1 Git

Para instalar este conjunto de *utility programs* contidos em Git introduziu-se o comando `sudo apt-get install git`.

```
osboxes@osboxes:~$ sudo apt-get install git
[sudo] password for osboxes:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,465 kB of archives.
After this operation, 38.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all 1:2.25.1-lubuntu3.2 [884 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-lubuntu3.2 [4,554 kB]
Fetched 5,465 kB in 8s (710 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 312545 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.25.1-lubuntu3.2_all.deb ...
Unpacking git-man (1:2.25.1-lubuntu3.2) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.25.1-lubuntu3.2_amd64.deb ...
Unpacking git (1:2.25.1-lubuntu3.2) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.25.1-lubuntu3.2) ...
Setting up git (1:2.25.1-lubuntu3.2) ...
Processing triggers for man-db (2.9.1-1) ...
osboxes@osboxes:~$
```

Figura 9 - Instalação de Git

5.2.2.2 cURL

Para fazer transferência de dados usou-se *cURL*, que foi instalado através do comando `sudo apt install curl`.

```
osboxes@osboxes:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.68.0-lubuntu2.7).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
osboxes@osboxes:~$
```

Figura 10 - Instalação de cURL

Para verificar a versão instalada utilizar `curl --version`.

```
osboxes@osboxes:~$ curl --version
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.0 libpsl/0.21.0 (+libidn2/2.2.0) l
ibssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPV6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEGO SSL TLS-SRP UnixSockets
osboxes@osboxes:~$
```

Figura 11 - Verificação da versão de cURL

5.2.2.3 Docker e Docker-Compose

Os *containers* onde a *rede* vai ser criada e onde se vai interagir com a mesma, vão ser criados usando *Docker* e *Docker-Compose*.

Começa-se por instalar o *Docker*. Com o comando `sudo apt-get remove docker docker-engine docker.io` para desinstalar versões anteriores de *Docker* que poderiam estar instaladas anteriormente.

```
osboxes@osboxes:~$ sudo apt-get remove docker docker-engine docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package docker-engine
```

Figura 12 - Desinstalação de possíveis artefactos de Docker

Seguidamente usa-se `sudo apt install docker.io`, para instalar *Docker*.

```

osboxes@osboxes:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 74.2 MB of archives.
After this operation, 360 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.0.1-0ubuntu2~20.04.1 [4,155 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.5.5-0ubuntu3~20.04.2 [33.0 MB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 docker.io amd64 20.10.7-0ubuntu5~20.04.2 [36.9 MB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 ubuntu-fan all 0.12.13 [34.5 kB]
Fetched 74.2 MB in 1min 21s (919 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 313479 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.6-2ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.6-2ubuntu1) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.0.1-0ubuntu2~20.04.1_amd64.deb ...
Unpacking runc (1.0.1-0ubuntu2~20.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.5.5-0ubuntu3~20.04.2_amd64.deb ...
Unpacking containerd (1.5.5-0ubuntu3~20.04.2) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../4-docker.io_20.10.7-0ubuntu5~20.04.2_amd64.deb ...
Unpacking docker.io (20.10.7-0ubuntu5~20.04.2) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../5-ubuntu-fan_0.12.13_all.deb ...
Unpacking ubuntu-fan (0.12.13) ...
Setting up runc (1.0.1-0ubuntu2~20.04.1) ...
Setting up bridge-utils (1.6-2ubuntu1) ...
Setting up pigz (2.4-1) ...
Setting up containerd (1.5.5-0ubuntu3~20.04.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.13) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (20.10.7-0ubuntu5~20.04.2) ...
Adding group `docker' (GID 135) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for systemd (245.4-4ubuntu3.15) ...
Processing triggers for man-db (2.9.1-1) ...
osboxes@osboxes:~$

```

Figura 13 - Instalação de Docker

Depois usa-se `sudo systemctl start docker` e `sudo systemctl enable docker` para o Docker inicializar quando se liga o computador.

```

osboxes@osboxes:~$ sudo systemctl start docker
osboxes@osboxes:~$ sudo systemctl enable docker
osboxes@osboxes:~$

```

Figura 14 - Configuração da inicialização de Docker no arranque da máquina

Finalmente, usa-se `sudo usermod -a -G docker <nome de utilizador>` (no caso apresentado o nome de utilizador é `osboxes`) para dar permissões de acesso ao utilizador.

```

osboxes@osboxes:~/Desktop$ sudo usermod -a -G docker osboxes
osboxes@osboxes:~/Desktop$

```

Figura 15 - Atribuição de permissões ao utilizador

Para verificar que versão do *Docker* foi instalada usar `docker --version`

```
osboxes@osboxes:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5-20.04.2
osboxes@osboxes:~$
```

Figura 16 - Verificação da versão de Docker

Passando para a instalação do *Docker-Compose*, usa-se o seguinte comando para fazer *download* e salvar o ficheiro em `/usr/local/bin` com o nome *docker-compose*.

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose.
```

```
osboxes@osboxes:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 664 100 664 0 0 1312 0 --:--:-- --:--:-- --:--:-- 1312
100 12.1M 100 12.1M 0 0 2095k 0 0:00:05 0:00:05 --:--:-- 3123k
osboxes@osboxes:~$
```

Figura 17 - Instalação de Docker-Compose

Seguidamente usa-se o comando apresentado para mudar as permissões do *Docker-Compose*.

```
sudo chmod +x /usr/local/bin/docker-compose
```

```
osboxes@osboxes:~$ sudo chmod +x /usr/local/bin/docker-compose
osboxes@osboxes:~$
```

Figura 18 - Atribuição de permissões a Docker-Compose

Para verificar a versão de *Docker-Compose* utiliza-se `sudo docker-compose --version`.

```
osboxes@osboxes:~$ sudo docker-compose --version
docker-compose version 1.29.2, build 5becea4c
osboxes@osboxes:~$
```

Figura 19 - Verificação da versão de Docker-Compose

5.2.2.4 Documentos Fabric-Samples

Nos documentos Fabric-Samples estão presentes os requisitos que são utilizados na criação das redes em HF. Adicionalmente estão presentes alguns exemplos de redes HF.

Antes de tentar fazer download dos *Fabric-Samples* é necessário reiniciar a VM para o que foi instalado anteriormente, relativamente às permissões do *Docker*, entrarem em vigor.

O *Fabric-Samples* utilizado foi instalado seguindo as instruções fornecidas em [20].

Como o *Fabric-Samples* foi instalado na diretoria *Desktop*, foi utilizado `cd Desktop/`, seguido por `curl -sSL https://bit.ly/2ysb0FE | bash -s`, para fazer download e instalar a última versão de *Fabric-Samples*, que neste caso foram as versões 2.4.2 do *Fabric* e 1.5.2 do *Fabric-CA*. Para instalar uma versão específica utilizar o comando `curl -sSL https://bit.ly/2ysb0FE | bash -s -- <fabric_version> <fabric-ca_version>`.

```

osboxes@osboxes:~$ cd Desktop/
osboxes@osboxes:~/Desktop$ curl -sSL https://bit.ly/2ysb0FE | bash -s

Clone hyperledger/fabric-samples repo

====> Changing directory to fabric-samples
fabric-samples v2.4.2 does not exist, defaulting to main. fabric-samples main branch is intended to work with recent versions of fabric.

Pull Hyperledger Fabric binaries

====> Downloading version 2.4.2 platform specific fabric binaries
====> Downloading: https://github.com/hyperledger/fabric/releases/download/v2.4.2/hyperledger-fabric-linux-amd64-2.4.2.tar.gz
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 680 100 680 0 0 1894 0 --:--:-- --:--:-- --:--:-- 1888
100 76.7M 100 76.7M 0 0 1189k 0 0:01:06 0:01:06 --:--:-- 1510k
==> Done.
====> Downloading version 1.5.2 platform specific fabric-ca-client binary
====> Downloading: https://github.com/hyperledger/fabric/releases/download/v1.5.2/hyperledger-fabric-ca-linux-amd64-1.5.2.tar.gz
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 683 100 683 0 0 1821 0 --:--:-- --:--:-- --:--:-- 1816
100 25.4M 100 25.4M 0 0 1743k 0 0:00:14 0:00:14 --:--:-- 1847k
==> Done.

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer cconv tools baseos
====> Pulling fabric Images
====> hyperledger/fabric-peer:2.4.2
2.4.2: Pulling from hyperledger/fabric-peer
97518928ae5f: Pull complete
42bd03df3e1a: Pull complete
3182c3c96871: Pull complete
44a896d37e25: Pull complete
2b2934b5a05d: Pull complete
e7f95e52b961: Pull complete
650bde1fc3f7: Pull complete
Digest: sha256:5eaad9fd093fdrb449310ae851912ab2cf6cd5b634380497404b9cb8bf91dcd2
Status: Downloaded newer image for hyperledger/fabric-peer:2.4.2
docker.io/hyperledger/fabric-peer:2.4.2
====> hyperledger/fabric-orderer:2.4.2
2.4.2: Pulling from hyperledger/fabric-orderer
97518928ae5f: Already exists
42bd03df3e1a: Already exists
a565d10714cf: Pull complete
c701ceab4d02: Pull complete
641b37a6c630: Pull complete
94f70ee0bbbc: Pull complete
3a78beab9d5d: Pull complete
Digest: sha256:fc9fb6d8c88ef7cff09835c5bf978388897a191ed9c9cf1ba33bc131a50799b
Status: Downloaded newer image for hyperledger/fabric-orderer:2.4.2
docker.io/hyperledger/fabric-orderer:2.4.2
====> hyperledger/fabric-cconv:2.4.2
2.4.2: Pulling from hyperledger/fabric-cconv
97518928ae5f: Already exists
b78c28b3bbf7: Pull complete
248309d37e25: Pull complete
8f893ed93684: Pull complete
60b34f272e36: Pull complete
bde89820d2b: Pull complete
759d9edbc0f: Pull complete
81a0619aeb06: Pull complete
fc7ae8c0e065: Pull complete
Digest: sha256:bd2f8a047b74f422c34b03cfe62750f7499f78410c6f4dd7ce4ef5f594
Status: Downloaded newer image for hyperledger/fabric-cconv:2.4.2
docker.io/hyperledger/fabric-cconv:2.4.2
====> hyperledger/fabric-tools:2.4.2
2.4.2: Pulling from hyperledger/fabric-tools
97518928ae5f: Already exists
b78c28b3bbf7: Already exists
248309d37e25: Already exists
8f893ed93684: Already exists
60b34f272e36: Already exists
fb1c258a462f: Pull complete
cdd115ade33: Pull complete
24ea658952d: Pull complete
Digest: sha256:c3c4cdfc73877e9d3db1fcbdb59c152ecd23876a1ccb9f9bc9c4bed69824e7
Status: Downloaded newer image for hyperledger/fabric-tools:2.4.2
docker.io/hyperledger/fabric-tools:2.4.2
====> hyperledger/fabric-baseos:2.4.2
2.4.2: Pulling from hyperledger/fabric-baseos
97518928ae5f: Already exists
42bd03df3e1a: Already exists
368f3bf0ffdc: Pull complete
Digest: sha256:bd4f46cc0e98ab4cfab4a8cb109ebba5424ae5c84c799d5ec0f5eb7ae2ae2ca
Status: Downloaded newer image for hyperledger/fabric-baseos:2.4.2
docker.io/hyperledger/fabric-baseos:2.4.2
====> Pulling fabric ca Image
====> hyperledger/fabric-ca:1.5.2
1.5.2: Pulling from hyperledger/fabric-ca
a080a04678b: Pull complete
a02550aeb1: Pull complete
6c802cf1a97: Pull complete
Digest: sha256:faa3b743d9ed391c30f518a7cc1168160bf335f3bf60ba6aaaf1aa49c1ed023e
Status: Downloaded newer image for hyperledger/fabric-ca:1.5.2
docker.io/hyperledger/fabric-ca:1.5.2
====> List out hyperledger docker images
hyperledger/fabric-tools 2.4 eb40f70b1174 5 weeks ago 473MB
hyperledger/fabric-tools 2.4.2 eb40f70b1174 5 weeks ago 473MB
hyperledger/fabric-tools latest eb40f70b1174 5 weeks ago 473MB
hyperledger/fabric-peer 2.4 43b970f84604 5 weeks ago 62.3MB
hyperledger/fabric-peer 2.4.2 43b970f84604 5 weeks ago 62.3MB
hyperledger/fabric-peer latest 43b970f84604 5 weeks ago 62.3MB
hyperledger/fabric-orderer 2.4 5edf6bd4489 5 weeks ago 37.3MB
hyperledger/fabric-orderer 2.4.2 5edf6bd4489 5 weeks ago 37.3MB
hyperledger/fabric-orderer latest 5edf6bd4489 5 weeks ago 37.3MB
hyperledger/fabric-cconv 2.4 e377a02242aa 5 weeks ago 517MB
hyperledger/fabric-cconv 2.4.2 e377a02242aa 5 weeks ago 517MB
hyperledger/fabric-cconv latest e377a02242aa 5 weeks ago 517MB
hyperledger/fabric-baseos 2.4 4cfe0148d657 5 weeks ago 6.94MB
hyperledger/fabric-baseos 2.4.2 4cfe0148d657 5 weeks ago 6.94MB
hyperledger/fabric-baseos latest 4cfe0148d657 5 weeks ago 6.94MB
hyperledger/fabric-ca 1.5 4ea287b75c63 5 months ago 69.8MB
hyperledger/fabric-ca 1.5.2 4ea287b75c63 5 months ago 69.8MB
hyperledger/fabric-ca latest 4ea287b75c63 5 months ago 69.8MB
osboxes@osboxes:~/Desktop$
osboxes@osboxes:~/Desktop$

```

Figura 20 - Instalação de Fabric-Samples

Para verificar que tudo foi bem instalado, deverá mudar-se para a diretoria *test-network* dentro da pasta criada *fabric-samples* com `cd fabric-samples/test-network/` e executar `./network.sh down` para desligar a rede e remover ficheiros criados (num primeiro uso é esperado que nada seja removido, devido à rede nunca ter sido inicializada), seguido de `./network.sh up` para inicializar a rede. Se a inicialização correr sem problemas confirma-se que ficou instalado corretamente.

```

osboxes@osboxes:~/Desktop$ cd fabric-samples/test-network
osboxes@osboxes:~/Desktop/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
Removing network fabric_test
WARNING: Network fabric_test not found.
Removing network compose_default
WARNING: Network compose_default not found.
Removing volume compose_orderer.example.com
WARNING: Volume compose_orderer.example.com not found.
Removing volume compose_peer0.org1.example.com
WARNING: Volume compose_peer0.org1.example.com not found.
Removing volume compose_peer0.org2.example.com
WARNING: Volume compose_peer0.org2.example.com not found.
Removing volume compose_peer0.org3.example.com
WARNING: Volume compose_peer0.org3.example.com not found.
Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com
Removing remaining containers
Removing generated chaincode docker images
"docker kill" requires at least 1 argument.
See 'docker kill --help'.

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
7e5209d2300f: Pull complete

Digest: sha256:34c3559bbdedefd67195e766e38cfbb0fcabff4241dbee3f390fd6e3310f5ebc
Status: Downloaded newer image for busybox:latest

```

Figura 21 – Encerramento da rede de exemplo

```

osboxes@osboxes:~/Desktop/fabric-samples/test-network$ ./network.sh up
Using docker and docker compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL VERSION=2.4.2
DOCKER IMAGE VERSIONS=4.2
/home/osboxes/Desktop/fabric-samples/test-network/./bin/cryptogen
generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done

Creating peer0.org1.example.com ... done

Creating orderer.example.com ... done

Creating cli ... done

CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          NAMES
a06b6bcd690   hyperledger/fabric-tools:latest     "/bin/bash"             Less than a second ago   Up Less than a second   cli
51e58096f7ff   hyperledger/fabric-peer:latest      "peer node start"       2 seconds ago          Up Less than a second   peer0.org2.example.com
, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp
41d2586dd8ec   hyperledger/fabric-orderer:latest   "orderer"               2 seconds ago          Up Less than a second   orderer.example.com
, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp
0eac4e094a7f   hyperledger/fabric-peer:latest      "peer node start"       2 seconds ago          Up Less than a second   peer0.org1.example.com
, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp
osboxes@osboxes:~/Desktop/fabric-samples/test-network$

```

Figura 22 - Inicialização da rede de exemplo

Para poder correr os scripts existentes na diretoria *fabric-samples* é necessário utilizar o comando apresentado.

```
export PATH=/home/<nome de utilizador>/Desktop/tese/fabric-samples/bin:$PATH
```

No caso apresentado o utilizador é *osboxes*, ficando o comando com a seguinte forma.

```
export PATH=/home/osboxes/Desktop/tese/fabric-samples/bin:$PATH
```

5.2.2.5 Node.JS e NPM

Para utilizar a linguagem *Javascript* em *chaincodes* é necessário instalar *Node.js* e *npm*.

A instalação inicia-se com o comando.

```
wget http://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz seguido
por sudo tar -C /usr/local --strip-components 1 -xzf node-v16.14.0-linux-x64.tar.gz
```

Deste modo fica-se então com *Node.js* e *npm* instalados na diretoria */usr/local/bin*. Isto pode ser verificado utilizando o comando `ls -l /usr/local/bin/node ls -l /usr/local/bin/npm`.


```

osboxes@osboxes:~/Desktop$ wget http://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz
--2022-03-11 09:13:08-- http://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz
Resolving nodejs.org (nodejs.org)... 104.20.23.46, 104.20.22.46, 2606:4700:10::6814:172e, ...
Connecting to nodejs.org (nodejs.org)|104.20.23.46|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz [following]
--2022-03-11 09:13:08-- https://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz
Connecting to nodejs.org (nodejs.org)|104.20.23.46|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32813953 (31M) [application/gzip]
Saving to: 'node-v16.14.0-linux-x64.tar.gz'

node-v16.14.0-linux 100%[=====] 31.29M 2.18MB/s in 15s

2022-03-11 09:13:23 (2.15 MB/s) - 'node-v16.14.0-linux-x64.tar.gz' saved [32813953/32813953]

osboxes@osboxes:~/Desktop$ sudo tar -C /usr/local --strip-components 1 -xzf node-v16.14.0-linux-x64.tar.gz
osboxes@osboxes:~/Desktop$
osboxes@osboxes:~/Desktop$ ls -l /usr/local/bin/node ls -l /usr/local/bin/npm
ls: cannot access 'ls': No such file or directory
-rwxr-xr-x 1 1001 1001 80310904 Feb  8 12:49 /usr/local/bin/node
lrwxrwxrwx 1 1001 1001 38 Feb  8 12:49 /usr/local/bin/npm -> ../lib/node_modules/npm/bin/npm-cli.js
osboxes@osboxes:~/Desktop$

```

Figura 23 - Instalação de NPM e Node.JS

5.2.2.6 GO

Para utilizar *Go* na implementação do *chaincode* é necessário instalar a linguagem. Com esse intuito seguem-se as instruções presentes em [21].

Para verificar a versão instalada usa-se `go version`.

```

osboxes@osboxes:~/Desktop/fabric-samples/fabcar$ go version
go version go1.18.1 linux/amd64
osboxes@osboxes:~/Desktop/fabric-samples/fabcar$

```

Figura 24 - Verificação da versão de Go

A diretoria onde os comandos são corridos é irrelevante.

5.2.2.7 jq

Comando usado para ler dados de um ficheiro *JSON*. Isto é necessário para a adição de organizações a uma rede pré-existente. Para instalar basta utilizar o comando `sudo apt-get install jq`.

5.2.2.8 Github Desktop

Apesar de ser bastante útil, a criação de um repositório de *GitHub* não é um pré-requisito proibitivo. Contudo numa das implementações, que irá ser descrita posteriormente, é vantajosa a criação de um repositório a partir de uma diretoria local.

No entanto, o processo de criação do repositório, não foi um processo linear.

Num primeiro momento existiu uma tentativa de seguir o tutorial presente em [22] onde é descrito como criar o repositório a partir do terminal, fazendo uso de comandos como `git init` (para criar um repositório), `git add` (para adicionar ficheiros ao repositório) e `git commit` (para guardar as modificações feitas ao repositório). Esta tentativa acabou por dar erros de permissão negada.

Para simplificar o processo e ter acesso a um modo mais visual de criação de repositórios, utilizou-se o *GitHub Desktop*. Para instalar foi-se a [23] fazer download e instalação consequente do executável.

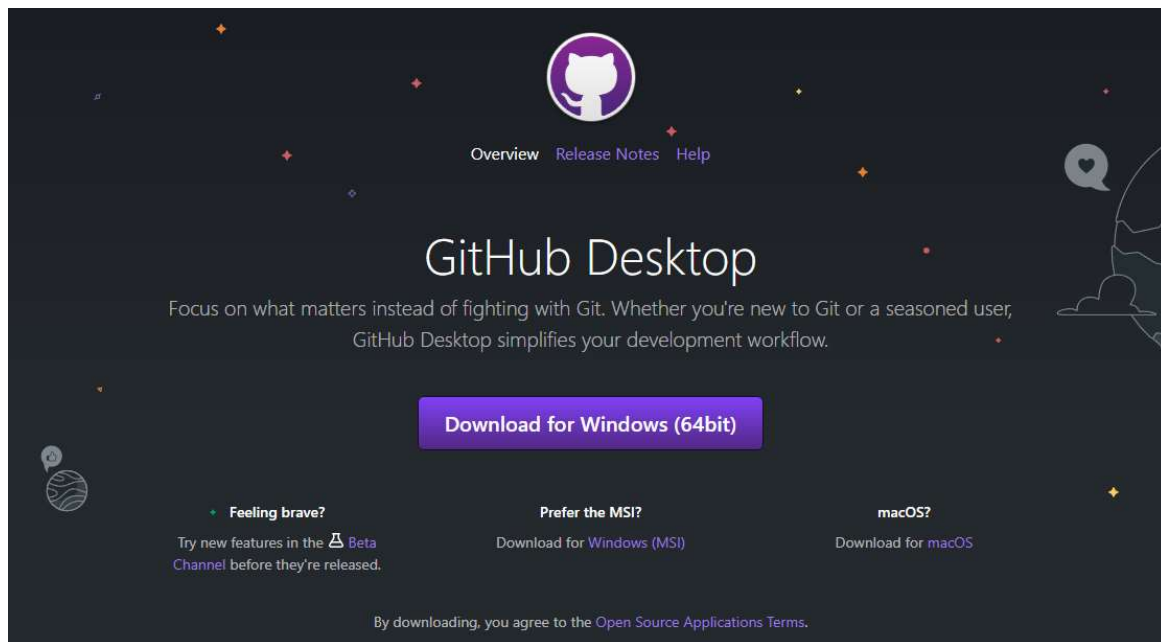


Figura 25 - Snapshot da webpage de instalação de Github Desktop [23]

Após a instalação criou-se o repositório utilizando a diretoria (onde se localizam os materiais da rede). Mas ao clicar em *fetch* (de modo a passar as mudanças para o repositório no *GitHub*) apareceram erros de autenticação, não sendo mencionada explicitamente qual era a autenticação a causar problemas. Todavia, em algumas das linhas de erros era mencionado *SSH*.

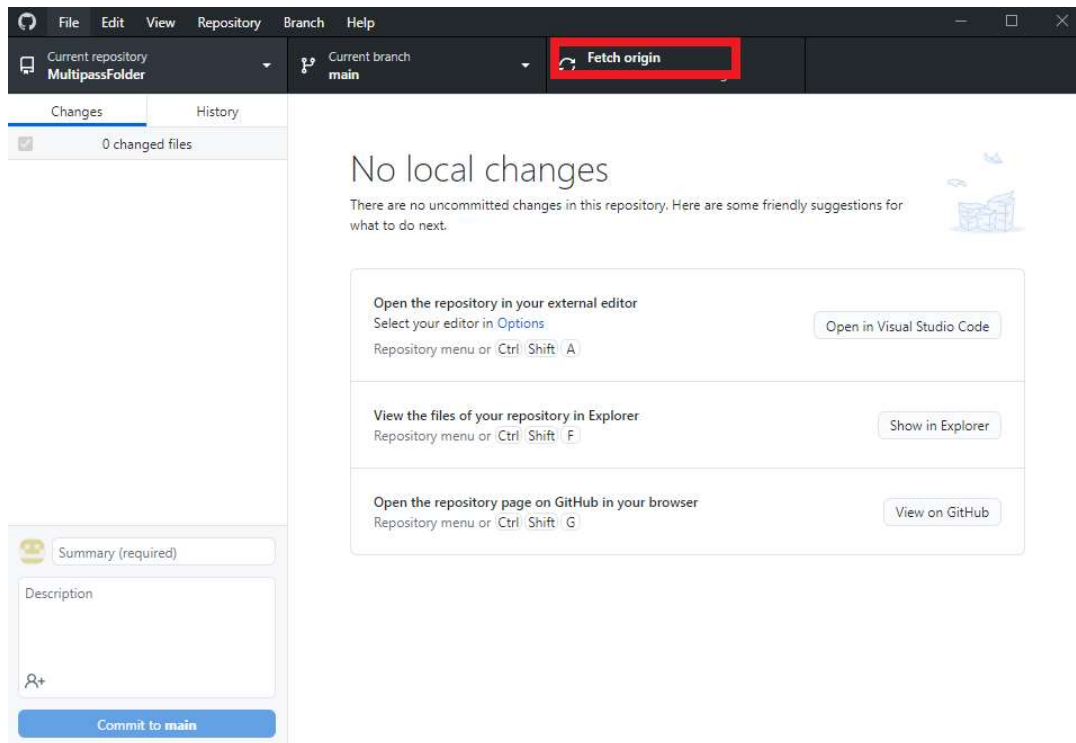


Figura 26 - Snapshot da página de Github Desktop

Numa tentativa de ultrapassar esta dificuldade utilizou-se [24] para criar uma chave *SSH* fazendo uso do comando `ssh-keygen` (usado na criação da chave pública e privada necessárias para a conexão *SSH*) e [25] para associar a chave pública à conta de *GitHub*.

Após percorrer os passos descritos continuou a aparecer o erro de autenticação. Nessa altura experimentou-se criar um repositório de raiz (sem nada), com o *GitHub Desktop* e seguidamente copiar os ficheiros necessários para a diretoria do repositório, conseguindo finalmente realizar o *fetch*.

Durante o *fetch* surgiu o *warning* "LF will be replaced by CRLF" que se deve ao facto de se terem *line endings* de *Linux* em *Windows*.

5.2.2.8 Subversion

Ferramenta normalmente utilizada na gestão e acompanhamento de mudanças de código em projetos. Neste caso vai ser utilizada para fazer download de uma diretoria de um repositório de *GitHub*.

Para instalar utilizou-se `sudo apt-get install subversion -y`


```

ubuntu@vm1:~$ sudo apt-get install subversion -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapr1 libaprutil1 libserf-1-1 libsvn1
Suggested packages:
  db5.3-util libapache2-mod-svn subversion-tools
The following NEW packages will be installed:
  libapr1 libaprutil1 libserf-1-1 libsvn1 subversion
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 2237 kB of archives.
After this operation, 9910 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 libapr1 amd64 1.6.3-2 [90.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 libaprutil1 amd64 1.6.1-2 [84.4 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/universe amd64 libserf-1-1 amd64 1.3.9-6 [44.4 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/universe amd64 libsvn1 amd64 1.9.7-4ubuntu1 [1183 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/universe amd64 subversion amd64 1.9.7-4ubuntu1 [834 kB]
Fetched 2237 kB in 4s (601 kB/s)
Selecting previously unselected package libapr1:amd64.
(Reading database ... 91423 files and directories currently installed.)
Preparing to unpack .../libapr1_1.6.3-2_amd64.deb ...
Unpacking libapr1:amd64 (1.6.3-2) ...
Selecting previously unselected package libaprutil1:amd64.
Preparing to unpack .../libaprutil1_1.6.1-2_amd64.deb ...
Unpacking libaprutil1:amd64 (1.6.1-2) ...
Selecting previously unselected package libserf-1-1:amd64.
Preparing to unpack .../libserf-1-1_1.3.9-6_amd64.deb ...
Unpacking libserf-1-1:amd64 (1.3.9-6) ...
Selecting previously unselected package libsvn1:amd64.
Preparing to unpack .../libsvn1_1.9.7-4ubuntu1_amd64.deb ...
Unpacking libsvn1:amd64 (1.9.7-4ubuntu1) ...
Selecting previously unselected package subversion.
Preparing to unpack .../subversion_1.9.7-4ubuntu1_amd64.deb ...
Unpacking subversion (1.9.7-4ubuntu1) ...
Setting up libapr1:amd64 (1.6.3-2) ...
Setting up libaprutil1:amd64 (1.6.1-2) ...
Setting up libserf-1-1:amd64 (1.3.9-6) ...
Setting up libsvn1:amd64 (1.9.7-4ubuntu1) ...
Setting up subversion (1.9.7-4ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.5) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

```

Figura 27 - Instalação de Subversion

5.2.2.8 yq

Versão de jq utilizada na edição de ficheiros do tipo yaml. Este comando foi utilizado na criação do script criador dos documentos consoante a configuração (secção 5.2.3.7). Para instalar utiliza-se o comando `sudo snap install yq`.

5.2.3 Criação, implementação e execução de diferentes configurações de rede

5.2.3.1 Implementação inicial de rede com três organizações

A implementação inicial foi realizada com a ajuda do MSc. Jorge Rivadeneira, onde usando [6], foi implementada em linguagem *go* uma rede onde existem três organizações, entre as quais uma das organizações apenas consegue ler a informação do canal, não podendo escrever para o canal, ao contrário das outras duas que podem ler e escrever.

O processo de criação da rede descrita irá ser explicado passo a passo de seguida:

- 1) Inicialmente recorreu-se a *cryptogen* [26], para gerar material chave do HF, onde se utilizou a configuração presente em *crypto-config.yaml*, que descreve quantas

organizações existem, número de utilizadores dentro das organizações, nome e domínio das organizações.

```
#####
# crypto-config.yaml
#####
OrdererOrgs:
- Name: Orderer
  Domain: acme.com
  EnableNodeOUs: true
  Specs:
  - Hostname: orderer
    SANS:
    - localhost
PeerOrgs:
- Name: Org1
  Domain: org1.acme.com
  EnableNodeOUs: true
  Template:
    Count: 1
    SANS:
    - localhost
  Users:
    Count: 1
- Name: Org2
  Domain: org2.acme.com
  EnableNodeOUs: true
  Template:
    Count: 1
    SANS:
    - localhost
  Users:
    Count: 1
- Name: Org3
  Domain: org3.acme.com
  EnableNodeOUs: true
  Template:
    Count: 1
    SANS:
    - localhost
  Users:
    Count: 1
```

Figura 28 - Ficheiro *crypto-config.yaml*

```
linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ cryptogen generate --config=./crypto-config.yaml
org1.acme.com
org2.acme.com
org3.acme.com
```

Figura 29 - Comando *cryptogen* utilizado

- 2) Seguidamente, usou-se *configtxgen* [27] que, fazendo uso dos perfis criados em *configtx.yaml* (onde se define informação necessária à criação do canal, como o comportamento das políticas, os *hosts* e os *ports* usados), se cria a configuração do canal para cada uma das organizações.

```

#####
# Profile
#####
Profiles:

ThreeOrgsOrdererGenesis:
  <<: *ChannelDefaults
Orderer:
  <<: *OrdererDefaults
Organizations:
  - *OrdererOrg
Capabilities:
  <<: *OrdererCapabilities
Consortiums:
  SampleConsortium:
    Organizations:
      - *Org1
      - *Org2
      - *Org3
ThreeOrgsChannel:
  Consortium: SampleConsortium
  <<: *ChannelDefaults
Application:
  <<: *ApplicationDefaults
Organizations:
  - *Org1
  - *Org2
  - *Org3
Capabilities:
  <<: *ApplicationCapabilities

```

Figura 30 - Excerto de configtx.yaml relativo aos perfis

```

linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ configtxgen -profile ThreeOrgsOrdererGenesis -channelID system-channel -outputblock ./channel-artifacts/genesis.block
1821-07-08 15:42:15.384 EDT [common.tools.configtxgen] main -- INFO 001 Loading configuration
1821-07-08 15:42:15.402 EDT [common.tools.configtxgen.localconfig] load -- INFO 002 orderer type: solo
1821-07-08 15:42:15.402 EDT [common.tools.configtxgen.localconfig] load -- INFO 003 Loaded configuration: configtx.yaml
1821-07-08 15:42:15.459 EDT [common.tools.configtxgen] doOutputBlock -- INFO 004 Generating genesis block
1821-07-08 15:42:15.459 EDT [common.tools.configtxgen] doOutputBlock -- INFO 005 Creating system channel genesis block
1821-07-08 15:42:15.459 EDT [common.tools.configtxgen] doOutputBlock -- INFO 006 Writing genesis block
linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ configtxgen -profile ThreeOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID marketplace
1821-07-08 15:42:15.558 EDT [common.tools.configtxgen] main -- INFO 001 Loading configuration
1821-07-08 15:42:15.596 EDT [common.tools.configtxgen.localconfig] load -- INFO 002 Loaded configuration: configtx.yaml
1821-07-08 15:42:15.696 EDT [common.tools.configtxgen] doOutputChannelCreateTx -- INFO 003 Generating new channel configtx
1821-07-08 15:42:15.696 EDT [common.tools.configtxgen] doOutputChannelCreateTx -- INFO 004 Writing new channel tx
linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID marketplace -asOrg Org1MSP
1821-07-08 15:42:15.721 EDT [common.tools.configtxgen] main -- INFO 001 Loading configuration
1821-07-08 15:42:15.730 EDT [common.tools.configtxgen.localconfig] load -- INFO 002 Loaded configuration: configtx.yaml
1821-07-08 15:42:15.730 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 003 Generating anchor peer update
1821-07-08 15:42:15.731 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 004 Writing anchor peer update
linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID marketplace -asOrg Org2MSP
1821-07-08 15:42:15.852 EDT [common.tools.configtxgen] main -- INFO 001 Loading configuration
1821-07-08 15:42:15.871 EDT [common.tools.configtxgen.localconfig] load -- INFO 002 Loaded configuration: configtx.yaml
1821-07-08 15:42:15.871 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 003 Generating anchor peer update
1821-07-08 15:42:15.873 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 004 Writing anchor peer update
linuxmint@linuxmint201:~/Desktop/tutorial/acme-network$ configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -channelID marketplace -asOrg Org3MSP
1821-07-08 15:42:21.139 EDT [common.tools.configtxgen] main -- INFO 001 Loading configuration
1821-07-08 15:42:21.148 EDT [common.tools.configtxgen.localconfig] load -- INFO 002 Loaded configuration: configtx.yaml
1821-07-08 15:42:21.148 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 003 Generating anchor peer update
1821-07-08 15:42:21.188 EDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -- INFO 004 Writing anchor peer update

```

Figura 31 - Comandos configtxgen utilizados

- Depois, usou-se *Docker* para criar um *container* com uma BD *couchDB* para cada organização, usando a informação presente em *docker-compose-base.yaml* (informação relativa ao containers das organizações) e *docker-compose-clicouchdb.yaml* (informação dos containers, BD, *Certificate Authority* (CA) e *cli* que é o container onde se irá interagir com a rede).

```

# ---CHANGED--- The peer name is taken from the name generated by the "cryptogen" certs - it indicates the peer org 3 and one peer "peer0"
peer0.org3.acme.com:
# ---CHANGED--- Container name - same as the peer name
container_name: peer0.org3.acme.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - CORE_PEER_ID=peer0.org3.acme.com
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - CORE_PEER_ADDRESS=peer0.org3.acme.com:7051
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org3.acme.com:7051
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org3.acme.com:7051
  # ---CHANGED--- ensure that the MSP ID is correctly set of Org3
  - CORE_PEER_LOCALMSPID=Org3MSP
volumes:
  - /var/run/./host/var/run/
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - ../crypto-config/peerOrganizations/org3.acme.com/peers/peer0.org3.acme.com/msp/etc/hyperledger/fabric/msp
  # ---CHANGED--- changed to reflect peer name, org name and our company's domain
  - ../crypto-config/peerOrganizations/org3.acme.com/peers/peer0.org3.acme.com/tls/etc/hyperledger/fabric/tls
ports:
  - 9051:7051
  - 9053:7053

```

Figura 32 - Excerto de `docker-compose-base.yaml` relativo à terceira organização

- 4) Utilizando o *container cli* previamente inicializado começou-se a criar o canal, recorrendo ao comando *peer channel create* [28] usando a organização *orderer (Org1)* para o gerar.
- 5) Posteriormente utiliza-se *peer channel join* em cada uma das organizações para as juntar ao canal criado.
- 6) Após isso, fez-se uso de *peer channel update* com o intuito de definir os *Anchor Peers* (usado para *peers* de diferentes organizações reconhecerem a existência uns dos outros). Neste caso para todas as organizações tendo em conta a necessidade de alta disponibilidade e redundância.
- 7) Passando para a implementação do *chaincode*, usa-se *peer lifecycle chaincode package*, onde utilizando *scripts* criados da linguagem *go* (que descrevem o comportamento no *chaincode*, o formato das mensagens e funções a serem invocadas mais à frente), cria-se um ficheiro comprimido do tipo *tar.gz* com o *chaincode*, que é instalado usando *peer lifecycle chaincode install* em cada uma das organizações.

```

func (s *SmartContract) Query(ctx contractapi.TransactionContextInterface, foodId string) (*Food, error) {
    foodAsBytes, err := ctx.GetStub().GetState(foodId)

    if err != nil {
        return nil, fmt.Errorf("Failed to read from world state. %s", err.Error())
    }

    if foodAsBytes == nil {
        return nil, fmt.Errorf("%s does not exist", foodId)
    }

    food := new(Food)

    err = json.Unmarshal(foodAsBytes, food)
    if err != nil {
        return nil, fmt.Errorf("Unmarshal error. %s", err.Error())
    }

    return food, nil
}

```

Figura 33 - Excerto de ficheiro `.go` usado na criação da função *query*

- 8) Após a instalação do *chaincode* em cada organização tomou-se a decisão de quais vão ter acesso de escrita. Para tal usa-se *peer lifecycle chaincode approveformyorg* para aprovar a definição do *chaincode*, em cada uma das organizações. Com este comando, ao definir as *signature policie*s apenas para duas organizações, a terceira ficará impedida de escrever para o canal, algo que se pode observar com *peer lifecycle chaincode checkcommitreadiness*.
- 9) Finalmente para fazer *commit* do *chaincode* criado até aí, utiliza-se *peer lifecycle chaincode commit*.
- 10) Para testar o que foi implementado usam-se as funções presentes no *chaincode go* denominadas: *set* (para escrever no canal) e *query* (para ler do canal), onde se verificou que uma das organizações não tem autorização para escrever apesar de todas poderem ler do canal criado.

The screenshot shows a database record for 'marketplace_foodcontrol' with ID 'id:3'. The record is a JSON object with the following fields: '_id' (id:3), '_rev' (1-5cfd7a5133712f53fd4ae2a657109e70), 'farmer' (jooo), 'variety' (bananas), and '~version' (CgM8BwA=). The interface includes 'Save Changes' and 'Cancel' buttons.

```

1 {
2   "_id": "id:3",
3   "_rev": "1-5cfd7a5133712f53fd4ae2a657109e70",
4   "farmer": "jooo",
5   "variety": "bananas",
6   "~version": "CgM8BwA="
7 }

```

Figura 34 - Exemplo de valor introduzido na BD utilizando a função *set*

5.2.3.2 Implementação fazendo uso de scripts de fabric-samples com chaincode modificado

Nesta implementação existiu um foco na mudança do *chaincode*, tendo-se como objetivo correr a rede pré-existente em *fabric-samples* com esse *chaincode* novo, implementado em linguagem Go.

De seguida explica-se o processo:

- 1) Em *fabric-samples* deverá aceder-se à diretoria *./chaincode/fabcar/go* de modo a poder modificar o ficheiro do *chaincode* em linguagem Go denominado *fabcar.go* da maneira representada nas figuras 34-38 , sendo o código original o assinalado a vermelho e o novo código a verde.


```

20 // Car describes basic details of what makes up a car
21 type Car struct {
22     Make string `json:"make"`
23     Model string `json:"model"`
24     Colour string `json:"colour"`
25     Owner string `json:"owner"`
26 }
27
28 // QueryResult structure used for handling result of query
29 type QueryResult struct {
30     Key string `json:"Key"`
31     Record *Car
32 }

```

```

22 // Asset describes basic details of what makes up an Asset
23 type Asset struct {
24     UserID string `json:"UserID"`
25     DataControllerID string `json:"DataControllerID"`
26     PhoneResource string `json:"PhoneResource"`
27 }
28
29 // QueryResult structure used for handling result of query
30 type QueryResult struct {
31     Key string `json:"Key"`
32     Record *Asset
33 }

```

Figura 35 - Excerto do ficheiro fabcar.go relativo à estrutura dos dados

Ir-se-á modificar a estrutura e o seu nome de modo a passar a receber *UserID*, *DataControllerID* e *PhoneResource*.

```

33
34 // InitLedger adds a base set of cars to the ledger
35 func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
36     cars := []Car{
37         Car{Make: "Toyota", Model: "Prius", Colour: "blue", Owner: "Tomoko"},
38         Car{Make: "Ford", Model: "Mustang", Colour: "red", Owner: "Brad"},
39         Car{Make: "Hyundai", Model: "Tucson", Colour: "green", Owner: "Jin Soo"},
40         Car{Make: "Volkswagen", Model: "Passat", Colour: "yellow", Owner: "Max"},
41         Car{Make: "Tesla", Model: "S", Colour: "black", Owner: "Adriana"},
42         Car{Make: "Peugeot", Model: "205", Colour: "purple", Owner: "Michel"},
43         Car{Make: "Chery", Model: "S22L", Colour: "white", Owner: "Aarav"},
44         Car{Make: "Fiat", Model: "Punto", Colour: "violet", Owner: "Pari"},
45         Car{Make: "Tata", Model: "Nano", Colour: "indigo", Owner: "Valeria"},
46         Car{Make: "Holden", Model: "Barina", Colour: "brown", Owner: "Shotaro"},
47     }
48
49     for i, car := range cars {
50         carAsBytes, _ := json.Marshal(car)
51         err := ctx.GetStub().PutState("CAR"+strconv.Itoa(i), carAsBytes)
52
53         if err != nil {
54             return fmt.Errorf("Failed to put to world state. %s", err.Error())
55         }
56     }
57
58     return nil
59 }

```

```

34
35 // InitLedger adds a base set of info to the ledger
36 func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
37     db_info := []Asset{
38         Asset{UserID: "1", DataControllerID: "1", PhoneResource: "camera & microphone"},
39         Asset{UserID: "2", DataControllerID: "2", PhoneResource: "camera & microphone"},
40         Asset{UserID: "3", DataControllerID: "3", PhoneResource: "microphone"},
41         Asset{UserID: "4", DataControllerID: "4", PhoneResource: "camera & microphone"},
42         Asset{UserID: "5", DataControllerID: "5", PhoneResource: "camera & microphone"},
43         Asset{UserID: "6", DataControllerID: "3", PhoneResource: "camera & microphone"},
44         Asset{UserID: "7", DataControllerID: "4", PhoneResource: "camera & microphone"},
45         Asset{UserID: "8", DataControllerID: "2", PhoneResource: "camera & microphone"},
46         Asset{UserID: "9", DataControllerID: "1", PhoneResource: "camera & microphone"},
47         Asset{UserID: "10", DataControllerID: "3", PhoneResource: "camera & microphone"},
48     }
49
50     for i, asset := range db_info {
51         assetAsBytes, _ := json.Marshal(asset)
52         err := ctx.GetStub().PutState("ASSET"+strconv.Itoa(i), assetAsBytes)
53
54         if err != nil {
55             return fmt.Errorf("Failed to put to world state. %s", err.Error())
56         }
57     }
58
59     return nil
60 }

```

Figura 36 - Excerto do ficheiro fabcar.go relativo à inicialização do ledger

Modifica-se a inicialização para poplar a base de dados com valores relevantes.

```

60
61 // CreateCar adds a new car to the world state with given details
62 func (s *SmartContract) CreateCar(ctx contractapi.TransactionContextInterface, carNumber string, make string, model string, colour string, owner string) error {
63     car := Car{
64         Make:    make,
65         Model:   model,
66         Colour:  colour,
67         Owner:   owner,
68     }
69
70     carAsBytes, _ := json.Marshal(car)
71
72     return ctx.GetStub().PutState(carNumber, carAsBytes)
73 }
74
75 // QueryCar returns the car stored in the world state with given id
76 func (s *SmartContract) QueryCar(ctx contractapi.TransactionContextInterface, carNumber string) (*Car, error) {
77     carAsBytes, err := ctx.GetStub().GetState(carNumber)
78
79     if err != nil {
80         return nil, fmt.Errorf("Failed to read from world state. %s", err.Error())
81     }
82
83     if carAsBytes == nil {
84         return nil, fmt.Errorf("%s does not exist", carNumber)
85     }
86
87     car := new(Car)
88     _ = json.Unmarshal(carAsBytes, car)
89
90     return car, nil
91 }
92
93 // QueryAllCars returns all cars found in world state
94 func (s *SmartContract) QueryAllCars(ctx contractapi.TransactionContextInterface) ([]QueryResult, error) {
95     startKey := ""
96     endKey := ""
97
98     resultsIterator, err := ctx.GetStub().GetStateByRange(startKey, endKey)
99
100    if err != nil {
101        return nil, err
102    }
103    defer resultsIterator.Close()
104
105    results := []QueryResult{}
106
107    for resultsIterator.HasNext() {
108        queryResponse, err := resultsIterator.Next()
109
110        if err != nil {
111            return nil, err
112        }
113
114        car := new(Car)
115        _ = json.Unmarshal(queryResponse.Value, car)
116
117        queryResult := QueryResult{Key: queryResponse.Key, Record: car}
118        results = append(results, queryResult)
119    }
120
121    return results, nil
122 }
123
60
61 // CreateCar adds a new Asset to the world state with given details
62 func (s *SmartContract) CreateCar(ctx contractapi.TransactionContextInterface, assetNumber string, uid string, dcid string, pr string) error {
63     asset := Asset{
64         UserID:    uid,
65         DataControllerID: dcid,
66         PhoneResource: pr,
67     }
68
69
70     assetAsBytes, _ := json.Marshal(asset)
71
72     return ctx.GetStub().PutState(assetNumber, assetAsBytes)
73 }
74
75 // QueryCar returns the car stored in the world state with given id
76 func (s *SmartContract) QueryCar(ctx contractapi.TransactionContextInterface, assetNumber string) (*Asset, error) {
77     assetAsBytes, err := ctx.GetStub().GetState(assetNumber)
78
79     if err != nil {
80         return nil, fmt.Errorf("Failed to read from world state. %s", err.Error())
81     }
82
83     if assetAsBytes == nil {
84         return nil, fmt.Errorf("%s does not exist", assetNumber)
85     }
86
87     asset := new(Asset)
88     _ = json.Unmarshal(assetAsBytes, asset)
89
90     return asset, nil
91 }
92
93 // QueryAllCars returns all db_info found in world state
94 func (s *SmartContract) QueryAllCars(ctx contractapi.TransactionContextInterface) ([]QueryResult, error) {
95     startKey := ""
96     endKey := ""
97
98     resultsIterator, err := ctx.GetStub().GetStateByRange(startKey, endKey)
99
100    if err != nil {
101        return nil, err
102    }
103    defer resultsIterator.Close()
104
105    results := []QueryResult{}
106
107    for resultsIterator.HasNext() {
108        queryResponse, err := resultsIterator.Next()
109
110        if err != nil {
111            return nil, err
112        }
113
114        asset := new(Asset)
115        _ = json.Unmarshal(queryResponse.Value, asset)
116
117        queryResult := QueryResult{Key: queryResponse.Key, Record: asset}
118        results = append(results, queryResult)
119    }
120
121    return results, nil
122 }
123

```

Figura 37 - Excerto do ficheiro fabcar.go com diversas funções

Muda-se a denominação das variáveis a utilizar nas diferentes funções.

<pre> 123 124 // ChangeCarOwner updates the owner field of car with given id in world state 125 func (s *SmartContract) ChangeCarOwner(ctx contractapi.TransactionContextInterface, carNumber string, newOwner string) error { 126 car, err := s.QueryCar(ctx, carNumber) 127 128 if err != nil { 129 return err 130 } 131 132 car.Owner = newOwner 133 134 carAsBytes, _ := json.Marshal(car) 135 136 return ctx.GetStub().PutState(carNumber, carAsBytes) 137 } 138 </pre>	<pre> 123 /* There are no owners 124 // ChangeCarOwner updates the owner field of car with given id in world state 126 func (s *SmartContract) ChangeCarOwner(ctx contractapi.TransactionContextInterface, assetNumber string, newOwner string) error { 127 asset, err := s.QueryCar(ctx, assetNumber) 128 129 if err != nil { 130 return err 131 } 132 133 asset.Owner = newOwner 134 135 assetAsBytes, _ := json.Marshal(asset) 136 137 return ctx.GetStub().PutState(assetNumber, assetAsByte s) 138 } 139 */ 140 </pre>
--	---

Figura 38 - Excerto do ficheiro *fabcar.go* relativo à função que realiza a mudança de owner

No caso desta implementação não é necessário ter a função para mudar de *owner* (uma vez que, com a mudança da estrutura, esse valor já não é representado), sendo eliminada, ou seja, comentada.

<pre> 138 139 func main() { 140 141 chaincode, err := contractapi.NewChaincode(new(SmartCon tract)) 142 143 if err != nil { 144 fmt.Printf("Error create fabcar chaincode: %s", err.Error()) 145 return 146 } 147 148 if err := chaincode.Start(); err != nil { 149 fmt.Printf("Error starting fabcar chaincode: % s", err.Error()) 150 } 151 } </pre>	<pre> 140 141 func main() { 142 143 chaincode, err := contractapi.NewChaincode(new(SmartCon tract)) 144 145 if err != nil { 146 fmt.Printf("Error create chain-db-network chain code: %s", err.Error()) 147 return 148 } 149 150 if err := chaincode.Start(); err != nil { 151 fmt.Printf("Error starting chain-db-network cha incode: %s", err.Error()) 152 } 153 } </pre>
--	---

Figura 39 - Excerto do final do ficheiro *fabcar.go*

Muda-se de novo os nomes das variáveis para apresentar mensagens de erro mais adequadas.

- 2) Voltando à diretoria *fabric-samples* vai-se para *fabcar* e modifica-se *startFabric.sh* (script usado na inicialização da rede) da seguinte maneira:


```

36 # launch network; create channel and join peer to channel
37 pushd ../test-network
38 ./network.sh down
39 ./network.sh up createChannel -ca -s couchdb
40 ./network.sh deployCC -ccn fabcar -ccv 1 -cci initLedger -ccl
  ${CC_SRC_LANGUAGE} -ccp ${CC_SRC_PATH}
41 popd
42

```

```

36 # launch network; create channel and join peer to channel
37 pushd ../test-network
38 ./network.sh down
39
40
41 #CHANGED to have channel name "db-network" and deploy to it, al
  so changed chaincode name to chain-db-network
42 ./network.sh up createChannel -c db-network -ca -s couchdb
43 ./network.sh deployCC -c db-network -ccn chain-db-network -ccv
  1 -cci initLedger -ccl ${CC_SRC_LANGUAGE} -ccp ${CC_SRC_PATH}
44
45 popd
46

```

Figura 40 - Excerto do ficheiro startFabric.sh relativo à inicialização da network e do chaincode

Para atribuir o nome *db-network* ao *channel* e o nome *chain-db-network* ao *chaincode*. Caso se pretendam nomes diferentes é fulcral ter em conta que os nomes não podem incluir *underscore* (“_”).

```

1 cat <<EOF
2
3 Total setup execution time : ${({date +%s} - starttime)} secs
...
4
5 Next, use the FabCar applications to interact with the deployed
FabCar contract.
6 The FabCar applications are available in multiple programming la
nguaes.
7 Follow the instructions for the programming language of your cho
ice:
8
9 JavaScript:
10
11 Start by changing into the "javascript" directory:
12 cd javascript
13
14 Next, install all required packages:
15 npm install
16
17 Then run the following applications to enroll the admin user,
and register a new user
18 called appUser which will be used by the other applications to
interact with the deployed
19 FabCar contract:
20 node enrollAdmin
21 node registerUser
22
23 You can run the invoke application as follows. By default, the
invoke application will
24 create a new car, but you can update the application to submit
other transactions:
25 node invoke
26
27 You can run the query application as follows. By default, the
query application will
28 return all cars, but you can update the application to evaluat
e other transactions:
29 node query
30
31 TypeScript:
32
33 Start by changing into the "typescript" directory:
34 cd typescript
35
36 Next, install all required packages:
37 npm install
38
39 Next, compile the TypeScript code into JavaScript:
40 npm run build
41
42 Then run the following applications to enroll the admin user,
and register a new user
43 called appUser which will be used by the other applications to
interact with the deployed
44 FabCar contract:
45 node dist/enrollAdmin
46 node dist/registerUser
47
48 You can run the invoke application as follows. By default, the
invoke application will
49 create a new car, but you can update the application to submit
other transactions:
50 node dist/invoke
51
52 You can run the query application as follows. By default, the
query application will
53 return all cars, but you can update the application to evaluat
e other transactions:
54 node dist/query
55
56 Java:

```

```

1 cat <<EOF
2
3 Total setup execution time : ${({date +%s} - starttime)} secs
...
4
5 Next, use the FabCar applications to interact with the deployed
chain-db-network contract.
6 Follow the instructions:
7
8 JavaScript:
9
10 Start by changing into the "javascript" directory:
11 cd javascript
12
13 Next, install all required packages:
14 npm install
15
16 Then run the following applications to enroll the admin user,
and register a new user
17 called appUser which will be used by the other applications to
interact with the deployed contract:
18 node enrollAdmin.js
19 node registerUser.js
20
21 You can run the invoke application as follows. Which will crea
te a new database entry
22 with the information provided by the user:
23 node invoke.js <UserID> <DataControlerID> <PhoneResources>
24 ex: node invoke.js 13 14 microphone
25
26 You can run the query application as follows.
27 In this function you can either query the whole database as it
is presented:
28 node query.js
29 Or you can query a specific Asset from the database:
30 node query.js <Key>
31 ex: node query ASSET4

```

Figura 41 - Excerto do ficheiro startFabric.sh relativo às instruções de uso da aplicação em JavaScript

Para apresentar diferentes instruções de uso ao correr o *script*. Adicionalmente retiram-se as instruções de uso relativas a todas as linguagens de programação, exceto *JavaScript*, uma vez que será essa a linguagem utilizada.

- 3) Ainda na diretoria *fabcar* corre-se o comando `./startFabric.sh`. Este script faz uso de *network.sh* presente na diretoria *fabric-samples/test-network* de modo a apagar conteúdos de redes pré-existent e inicializar a nova rede com *channel* denominado

db-network. Adicionalmente, é instalado o *chaincode chain-db-network* em 2 organizações.

```

osboxes@osboxes:~/Desktop/fabric-samples/fabcar$ ./startFabric.sh
~/Desktop/fabric-samples/test-network --Desktop/fabric-samples/fabcar
Using Docker and docker-compose
Stopping network
Stopping cli ... done
Stopping peer0.org1.example.com ... done
Stopping peer0.org2.example.com ... done
Stopping couchdb1 ... done
Stopping orderer.example.com ... done
Stopping couchdb0 ... done
Stopping ca.org2 ... done
Stopping ca.org1 ... done
Stopping ca.orderer ... done
Removing cli ... done
Removing peer0.org1.example.com ... done
Removing peer0.org2.example.com ... done
Removing couchdb1 ... done
Removing orderer.example.com ... done
Removing couchdb0 ... done
Removing ca.org2 ... done
Removing ca.org1 ... done
Removing ca.orderer ... done
Removing network fabric test
Removing network compose default
WARNING: Network compose_default not found.
Removing volume compose_orderer.example.com
Removing volume compose_peer0.org1.example.com
Removing volume compose_peer0.org2.example.com
Removing volume compose_peer0.org3.example.com
WARNING: Volume compose_peer0.org3.example.com not found.
Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com
Cleaning remaining containers
Cleaning generated chaincode docker images
'docker kill' requires at least 1 argument.
See 'docker kill --help'.

Usage: docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
Creating channel db-network
Using organization 1
+ osnadmin channel join --channelID db-network --config-block ./channel-artifacts/db-network.block -o localhost:7053 --ca-file /home/osboxes/Desktop/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --client-cert /home/osboxes/Desktop/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt --client-key /home/osboxes/Desktop/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.key
+ res=0
Status: 201
{
  "name": "db-network",
  "url": "/participation/v1/channels/db-network",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
channel 'db-network' created
Joining org1 peer to the channel...
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'db-network'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org2 on channel 'db-network'...
Attempting to check the commit readiness of the chaincode definition on peer0.org2. Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID db-network --name chain-db-network --version 1 --sequence 1 --init-required --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org2 on channel 'db-network'
Using organization 1
Using organization 2
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/osboxes/Desktop/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --channelID db-network --name chain-db-network --peerAddresses localhost:7051 --tlsRootCertFiles /home/osboxes/Desktop/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem --peerAddresses localhost:9051 --tlsRootCertFiles /home/osboxes/Desktop/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/tlsca/tlsca.org2.example.com-cert.pem --version 1 --sequence 1 --init-required
+ res=0
2022-05-11 10:31:31.982 WET 0001 INFO [chaincodeCmd] ClientWait -> txid [6d6560cabf6f82e6aaffc946b8fea6af036380cf2189de894d540eda1bcece4f] committed with status (VALID) at localhost:7051
2022-05-11 10:31:31.998 WET 0002 INFO [chaincodeCmd] ClientWait -> txid [6d6560cabf6f82e6aaffc946b8fea6af036380cf2189de894d540eda1bcece4f] committed with status (VALID) at localhost:9051
chaincode definition committed on channel 'db-network'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'db-network'...
Attempting to query committed status on peer0.org1. Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID db-network --name chain-db-network
+ res=0
Committed chaincode definition for chaincode 'chain-db-network' on channel 'db-network':
Version: 1, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vssc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'db-network'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'db-network'...
Attempting to query committed status on peer0.org2. Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID db-network --name chain-db-network
+ res=0
Committed chaincode definition for chaincode 'chain-db-network' on channel 'db-network':
Version: 1, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vssc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'db-network'

```

Figura 42 - Conjunto de excertos relativos à execução do script *startFabric.sh*

- Depois da criação da rede, ainda em *fabcar*, na diretoria *JavaScript*, *query.js* modifica-se da seguinte maneira.

```

36
37 // Get the network (channel) our contract is deployed t
o.
38 const network = await gateway.getNetwork('mychannel');
39
40 // Get the contract from the network.
41 const contract = network.getContract('fabcar');
42
43 // Evaluate the specified transaction.
44 // queryCar transaction - requires 1 argument, ex: ('qu
eryCar', 'CAR4')
45 // queryAllCars transaction - requires no arguments, e
x: ('queryAllCars')
46 const result = await contract.evaluateTransaction('quer
yAllCars');
47 console.log(`Transaction has been evaluated, result is:
${result.toString()}`);
48
36
37 //CHANGED network name from mychannel to db-network
38 // Get the network (channel) our contract is deployed t
o.
39 const network = await gateway.getNetwork('db-network');
40
41 //CHANGED network name from fabcar to chain-db-network
42 // Get the contract from the network.
43 const contract = network.getContract('chain-db-networ
k');
44
45 // Evaluate the specified transaction.
46 // queryCar transaction - requires 1 argument, ex: node
query.js ASSET4
47 // queryAllCars transaction - requires no arguments, no
de query.js
48
49 //CHANGED in order to be able to choose what query to u
se in command line
50 const myArgs = process.argv.slice(2);
51 if (myArgs[0] !=null){
52     const result = await contract.evaluateTransaction
('QueryCar', myArgs[0]);
53     console.log(`Transaction has been evaluated, result
is: ${result.toString()}`);
54 } else if (myArgs[0] ==null){
55     const result = await contract.evaluateTransaction
('QueryAllCars');
56     console.log(`Transaction has been evaluated, result
is: ${result.toString()}`);
57 }
58 else{
59     console.log("To evaluate the specified transactio
n");
60     console.log("queryCar transaction - requires 1 argu
ment, ex: node query.js ASSET4");
61     console.log("queryAllCars transaction - requires no
arguments, node query.js");
62     console.log("Run query.js as specified above");
63 }
64
65
48

```

Figura 43 - Excerto do ficheiro query.js

Muda-se o nome da rede e *chaincode* para o que foi definido anteriormente. Modifica-se também o comportamento do código de modo a poder definir que tipo de *query* fazer e especificar o *ASSET* a ser pesquisado, tudo isto na linha de comandos.

5) Ainda na diretoria *JavaScript*, muda-se o ficheiro *invoke.js* consoante o apresentado de seguida.

```

35
36 // Get the network (channel) our contract is deployed t
o.
37 const network = await gateway.getNetwork('mychannel');
38
39 // Get the contract from the network.
40 const contract = network.getContract('fabcar');
41
42 // Submit the specified transaction.
43 // createCar transaction - requires 5 arguments, ex: {cr
eateCar, 'CAR12', 'Honda', 'Accord', 'Black', 'Tom'}
44 // changeCarOwner transaction - requires 2 args , ex:
'changeCarOwner', 'CAR12', 'Dave'
45 await contract.submitTransaction('createCar', 'CAR12',
'Honda', 'Accord', 'Black', 'Tom');
46 console.log('Transaction has been submitted');
47
48 // Disconnect from the gateway.
49 await gateway.disconnect();
50
51 } catch (error) {
52 console.error('Failed to submit transaction: ${error}');
53 process.exit(1);
54 }
55 }

```

```

36 //CHANGED network name from mychannel to db-network
37 // Get the network (channel) our contract is deployed t
o.
39 const network = await gateway.getNetwork('db-network');
40
41 //CHANGED network name from fabcar to chain-db-network
42 // Get the contract from the network.
43 const contract = network.getContract('chain-db-networ
k');
44
45 // Submit the specified transaction.
46 // createCar transaction - requires 3 arguments, ex: nod
e invoke.js 13 14 microphone
47
48 //CHANGED to automatically choose a new key value taking
into account the last ASSET in the database
49 const result = await contract.evaluateTransaction('Query
AllCars');
50 let s_result=result.toString();
51 //Registers the number of times the string 'key' occurs
in string 's_result'
52 var count=s_result.match(/key/g).length;
53 let new_asset="ASSET"+ count;
54
55 //DEBUG CONSOLES
56 //console.log(result.toString());
57 //console.log(count);
58 //console.log(new_asset);
59
60 //CHANGED to take UserID, DatacontrolerID and phoneResou
rces directly in the terminal
61 const myArgs = process.argv.slice(2);
62 await contract.submitTransaction('createCar', new_asset,
myArgs[0], myArgs[1], myArgs[2]);
63 console.log('Transaction has been submitted');
64
65 // Disconnect from the gateway.
66 await gateway.disconnect();
67
68 } catch (error) {
69 console.error('Failed to submit transaction: ${error}');
70 process.exit(1);
71 }
72 }

```

Figura 44 - Excerto do ficheiro *invoke.js*

Em que se modifica o código de modo a criar uma *key* para o novo valor na base de dados e para poder especificar as informações do novo *ASSET* a ser criado no terminal.

6) Finalmente, basta apenas seguir as instruções apresentadas no fim de correr o script *startFabric.sh*.

```

Total setup execution time : 79 secs ...

Next, use the FabCar applications to interact with the deployed chain-db-network contract.
Follow the instructions:

JavaScript:

Start by changing into the "javascript" directory:
cd javascript

Next, install all required packages:
npm install

Then run the following applications to enroll the admin user, and register a new user
called appUser which will be used by the other applications to interact with the deployed contract:
node enrollAdmin.js
node registerUser.js

You can run the invoke application as follows. Which will create a new database entry
with the information provided by the user:
node invoke.js <UserID> <DataControlerID> <PhoneResources>
ex: node invoke.js 13 14 microphone

You can run the query application as follows.
In this function you can either query the whole database as it is presented:
node query.js
Or you can query a specific Asset from the database:
node query.js <Key>
ex: node query ASSET4

```

Figura 45 - Instruções apresentadas ao executar *startFabric.sh*

- 7) Após fazer `npm install`, na diretoria `./fabcar/javascript` (para instalar os packages necessários para correr os passos seguintes) utilizar `node enrollAdmin.js` e `node registerUser.js` para registrar o `admin` e criar um novo utilizador para interagir com a aplicação.

```
osboxes@osboxes:~/Desktop/fabric-samples/fabcar$ cd javascript/
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'fabric-ca-client@2.2.11',
npm WARN EBADENGINE   required: { node: '^10.15.3 || ^12.13.1 || ^14.13.1', npm: '^6.4.1' },
npm WARN EBADENGINE   current: { node: 'v16.14.0', npm: '8.3.1' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'fabric-common@2.2.11',
npm WARN EBADENGINE   required: { node: '^10.15.3 || ^12.13.1 || ^14.13.1', npm: '^6.4.1' },
npm WARN EBADENGINE   current: { node: 'v16.14.0', npm: '8.3.1' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'fabric-network@2.2.11',
npm WARN EBADENGINE   required: { node: '^10.15.3 || ^12.13.1 || ^14.13.1', npm: '^6.4.1' },
npm WARN EBADENGINE   current: { node: 'v16.14.0', npm: '8.3.1' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'fabric-protos@2.2.11',
npm WARN EBADENGINE   required: { node: '^10.15.3 || ^12.13.1 || ^14.13.1', npm: '^6.4.1' },
npm WARN EBADENGINE   current: { node: 'v16.14.0', npm: '8.3.1' }
npm WARN EBADENGINE }

up to date, audited 345 packages in 1s

19 packages are looking for funding
  run `npm fund` for details

13 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ node enrollAdmin.js
Wallet path: /home/osboxes/Desktop/fabric-samples/fabcar/javascript/wallet
Successfully enrolled admin user "admin" and imported it into the wallet
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ node registerUser.js
Wallet path: /home/osboxes/Desktop/fabric-samples/fabcar/javascript/wallet
Successfully registered and enrolled admin user "appUser" and imported it into the wallet
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$
```

Figura 46 - Execução de `npm install`, registo de `admin` e criação de novo utilizador

- 8) Finalmente pode-se usar `node query.js` e `node invoke.js` para utilizar as funcionalidades descritas anteriormente.

```
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ node query.js
Wallet path: /home/osboxes/Desktop/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: [{"Key": "ASSET0", "Record": {"UserID": "1", "DataControllerID": "1", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET1", "Record": {"UserID": "2", "DataControllerID": "2", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET2", "Record": {"UserID": "3", "DataControllerID": "3", "PhoneResource": "microphone"}}, {"Key": "ASSET3", "Record": {"UserID": "4", "DataControllerID": "4", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET4", "Record": {"UserID": "5", "DataControllerID": "5", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET5", "Record": {"UserID": "6", "DataControllerID": "3", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET6", "Record": {"UserID": "7", "DataControllerID": "4", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET7", "Record": {"UserID": "8", "DataControllerID": "2", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET8", "Record": {"UserID": "9", "DataControllerID": "1", "PhoneResource": "camera \u0026 microphone"}}, {"Key": "ASSET9", "Record": {"UserID": "10", "DataControllerID": "3", "PhoneResource": "camera \u0026 microphone"}}]
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ node invoke.js 13 14 microphone
Wallet path: /home/osboxes/Desktop/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$ node query.js ASSET10
Wallet path: /home/osboxes/Desktop/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"UserID": "13", "DataControllerID": "14", "PhoneResource": "microphone"}
osboxes@osboxes:~/Desktop/fabric-samples/fabcar/javascript$
```

Figura 47 - Execução de `query.js` e `invoke.js`

5.2.3.3 Implementação de uma rede com quatro organizações sem scripts

Nesta implementação criou-se a seguinte estrutura de diretorias.



Figura 48 - Estrutura de diretorias criada

A diretoria *chaincode* contém o código *chaincode* a correr em cada organização, passo esse que será descrito mais á frente.

Na diretoria *jivr-network* está toda a configuração da rede. Entrando nessa diretoria observa-se o seguinte:



Figura 49 - Estrutura da diretoria *jivr-network*

A configuração é feita seguindo os seguintes passos a partir da diretoria apresentada anteriormente:

- 1) Inicialmente irá ser necessário definir o comportamento do ficheiro *crypto-config.yaml*, da seguinte forma.

```
jnv-network > ! crypto-config.yaml
1 #####
2 #   crypto-config.yaml
3 #####
4 OrdererOrgs:
5 - Name: Orderer
6   Domain: jnvr.com
7   EnableNodeOUs: true
8   Specs:
9     - Hostname: orderer
10      SANS:
11        - localhost
12 PeerOrgs:
13 - Name: Org1
14   Domain: org1.jnvr.com
15   EnableNodeOUs: true
16   Template:
17     Count: 1
18     SANS:
19       - localhost
20   Users:
21     Count: 1
22 - Name: Org2
23   Domain: org2.jnvr.com
24   EnableNodeOUs: true
25   Template:
26     Count: 1
27     SANS:
28       - localhost
29   Users:
30     Count: 1
31 - Name: Org3
32   Domain: org3.jnvr.com
33   EnableNodeOUs: true
34   Template:
35     Count: 1
36     SANS:
37       - localhost
38   Users:
39     Count: 1
40 - Name: Org4
41   Domain: org4.jnvr.com
42   EnableNodeOUs: true
43   Template:
44     Count: 1
45     SANS:
46       - localhost
47   Users:
48     Count: 1
49
```

Figura 50 - Ficheiro *crypto-config.yaml*

No ficheiro *crypto-config.yaml* define-se as organizações da rede. No caso da implementação existe uma organização *Orderer* e quatro organizações *Peer*. Em cada um explicita-se o nome, o domínio da organização, o número de utilizadores e *EnableNodeOUs* é colocado *true* para permitir a classificação de identidades. Neste caso são utilizados *peer*, *client*, *admin* e *member* [29], como se verifica futuramente.

Após ter o ficheiro preparado, corre-se o comando `cryptogen generate --config=./crypto-config.yaml` para criar materiais chave da rede.

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ cryptogen generate --config=./crypto-config.yaml
org1.jnvr.com
org2.jnvr.com
org3.jnvr.com
org4.jnvr.com
```

Figura 51 - Execução do comando `cryptogen generate`

- 2) No próximo passo ir-se-á criar artefactos relacionados à configuração do canal. Para isso, vai ser utilizado o comando `configtxgen` de diversas maneiras. Para utilizar estes comandos irá ser necessária a criação de `configtx.yaml`, que descreve toda a rede e o seu comportamento.

Identifica-se cada organização e denomina-se-lhes um ID.

```
22
23     - &Org1
24       Name: Org1MSP
25       ID: Org1MSP
26       MSPDir: crypto-config/peerOrganizations/org1.jnvr.com/msp
27       Policies:
28         Readers:
29           Type: Signature
30           Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
31         Writers:
32           Type: Signature
33           Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
34         Admins:
35           Type: Signature
36           Rule: "OR('Org1MSP.admin')"
37         Endorsement:
38           Type: Signature
39           Rule: "OR('Org1MSP.peer')"
40       AnchorPeers:
41         - Host: peer0.org1.jnvr.com
42           Port: 7051
```

Figura 52 - Excerto do ficheiro `configtx.yaml` relativo à identificação da organização 1

Apresentam-se de seguida as capacidades do canal, que irão ser definidas nas próximas figuras 53-55.

```

106 #####
107 # SECTION: Capabilities
108 #####
109 Capabilities:
110   Channel: &ChannelCapabilities
111     V2_0: true
112
113   Orderer: &OrdererCapabilities
114     V2_0: true
115
116   Application: &ApplicationCapabilities

```

Figura 53 - Excerto do ficheiro configtx.yaml relativo à definição das capacidades do canal

Uma das definições de capacidades é a secção *Application*, que identifica os recursos usados pelos *peer nodes*.

```

118 #####
119 # SECTION: Application
120 #####
121 Application: &ApplicationDefaults
122
123   # Organizations is the list of orgs which are defined as participants on
124   # the application side of the network
125   Organizations:
126
127   # Policies defines the set of policies at this level of the config tree
128   # For Application policies, their canonical path is
129   # /Channel/Application/<PolicyName>
130   Policies:
131     Readers:
132       Type: ImplicitMeta
133       Rule: "ANY Readers"
134     Writers:
135       Type: ImplicitMeta
136       Rule: "ANY Writers"
137     Admins:
138       Type: ImplicitMeta
139       Rule: "MAJORITY Admins"
140     LifecycleEndorsement:
141       Type: ImplicitMeta
142       Rule: "MAJORITY Endorsement"
143     Endorsement:
144       Type: ImplicitMeta
145       Rule: "MAJORITY Endorsement"
146
147   Capabilities:
148     <<: *ApplicationCapabilities
149
150

```

Figura 54 - Excerto do ficheiro configtx.yaml relativo à definição da secção da aplicação

Outra das definições de capacidades é a secção *Orderer*, usada para identificar os recursos usados pelos *orderer nodes*.

```
150
151 #####
152 # SECTION: Orderer
153 #####
154 Orderer: &OrdererDefaults
155 # Orderer Type: The orderer implementation to start
156 OrdererType: solo
157
158 # OrdererType: etcdraft
159
160 EtcdRaft:
161   Consenters:
162     - Host: orderer.jnvr.com
163       Port: 7050
164       ClientTLS Cert: ../organizations/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/tls/server.crt
165       ServerTLS Cert: ../organizations/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/tls/server.crt
166
167   Addresses:
168     - orderer.jnvr.com:7050
169   BatchTimeout: 2s
170   BatchSize:
171     MaxMessageCount: 10
172     AbsoluteMaxBytes: 99 MB
173     PreferredMaxBytes: 512 KB
174
175   Kafka:
176     Brokers:
177       - 127.0.0.1:9092
178   Organizations:
179
180   Policies:
181     Readers:
182       Type: ImplicitMeta
183       Rule: "ANY Readers"
184     Writers:
185       Type: ImplicitMeta
186       Rule: "ANY Writers"
187     Admins:
188       Type: ImplicitMeta
189       Rule: "MAJORITY Admins"
190   # BlockValidation specifies what signatures must be included in the block
191   # from the orderer for the peer to validate it.
192   BlockValidation:
193     Type: ImplicitMeta
194     Rule: "ANY Writers"
195
```

Figura 55 - Excerto do ficheiro *configtx.yaml* relativo à definição dos *orderers*

A última definição das capacidades do canal é a secção *Channel*, que define os valores a codificar para parâmetros relativos ao canal.

```
196 #####
197 #
198 # CHANNEL
199 #
200 # This section defines the values to encode into a config transaction or
201 # genesis block for channel related parameters.
202 #
203 #####
204 Channel: &ChannelDefaults
205 # Policies defines the set of policies at this level of the config tree
206 # For Channel policies, their canonical path is
207 # /Channel/<PolicyName>
208 Policies:
209 # Who may invoke the 'Deliver' API
210 Readers:
211   Type: ImplicitMeta
212   Rule: "ANY Readers"
213 # Who may invoke the 'Broadcast' API
214 Writers:
215   Type: ImplicitMeta
216   Rule: "ANY Writers"
217 # By default, who may modify elements at this config level
218 Admins:
219   Type: ImplicitMeta
220   Rule: "MAJORITY Admins"
221
222 # Capabilities describes the channel level capabilities, see the
223 # dedicated Capabilities section elsewhere in this file for a full
224 # description
225 Capabilities:
226   <<: *ChannelCapabilities
```

Figura 56 - Excerto do ficheiro *configtx.yaml* relativo à definição do canal

Finalmente define-se as políticas que são utilizadas pelo comando *configtxgen* na criação do canal e seus elementos.

```
227 #####
228 # Profile
229 #
230 #####
231 Profiles:
232
233   FourOrgsOrdererGenesis:
234     <<: *ChannelDefaults
235     Orderer:
236       <<: *OrdererDefaults
237       Organizations:
238         - *OrdererOrg
239       Capabilities:
240         <<: *OrdererCapabilities
241     Consortiums:
242       SampleConsortium:
243         Organizations:
244           - *Org1
245           - *Org2
246           - *Org3
247           - *Org4
248   FourOrgsChannel:
249     Consortium: SampleConsortium
250     <<: *ChannelDefaults
251     Application:
252       <<: *ApplicationDefaults
253       Organizations:
254         - *Org1
255         - *Org2
256         - *Org3
257         - *Org4
258       Capabilities:
259         <<: *ApplicationCapabilities
```

Figura 57 - Excerto do ficheiro *configtx.yaml* relativo à definição das políticas do canal

Após a configuração de *configtx.yaml* usa-se o seguinte comando, para criar o bloco de génese do canal.

```
configtxgen -profile FourOrgsOrdererGenesis -channelID system-channel
-outputBlock ./channel-artifacts/genesis.block
```

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ configtxgen -profile FourOrgsOrdererGenesis -channelID system-channel -outputBlock ./channel-artifacts/genesis.block
2022-04-12 09:13:19.492 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:13:19.502 WEST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: solo
2022-04-12 09:13:19.502 WEST 0003 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:13:19.531 WEST 0004 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2022-04-12 09:13:19.531 WEST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Creating system channel genesis block
2022-04-12 09:13:19.532 WEST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
```

Figura 58 - Criação do bloco de gênese

O comando seguinte irá ser utilizado para criar o canal, que neste caso se denomina *channel1*.

```
configtxgen -profile FourOrgsChannel -outputCreateChannelTx
./channel-artifacts/channel.tx -channelID channel1,
```

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ configtxgen -profile FourOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID channel1
2022-04-12 09:15:34.110 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:15:34.121 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:15:34.121 WEST 0003 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Generating new channel configtx
2022-04-12 09:15:34.166 WEST 0004 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Writing new channel tx
```

Figura 59 - Criação do canal *channel1*

Os comandos apresentados subsequentemente são usados para criar os *anchor peers* de cada organização.

```
configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org1MSPanchors.tx -channelID channel1 -asOrg
Org1MSP
```

```
configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org2MSPanchors.tx -channelID channel1 -asOrg
Org2MSP
```

```
configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org3MSPanchors.tx -channelID channel1 -asOrg
Org3MSP
```

```
configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org4MSPanchors.tx -channelID channel1 -asOrg
Org4MSP
```



```

osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID channel1 -asOrg Org1MSP
2022-04-12 09:15:46.381 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:15:46.391 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:15:46.391 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-12 09:15:46.413 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID channel1 -asOrg Org2MSP
2022-04-12 09:15:53.459 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:15:53.468 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:15:53.468 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-12 09:15:53.491 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -channelID channel1 -asOrg Org3MSP
2022-04-12 09:16:06.885 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:16:06.898 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:16:06.898 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-12 09:16:06.924 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org4MSPanchors.tx -channelID channel1 -asOrg Org4MSP
2022-04-12 09:16:16.445 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-12 09:16:16.454 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-12 09:16:16.454 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-12 09:16:16.480 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update

```

Figura 60 - Criação dos anchor peers em todas as organizações

- 3) Passando para a criação dos containers *Docker* onde a rede e o *channel* correm. É necessário criar o documento *docker-compose-cli-couchdb.yaml*, que define o nome da rede, estabelece informação relativa a todos os elementos da rede, inclui todas as organizações, fixa o CA da *Org1*, determina os *couchdb* (utilizados para armazenar a informação de cada organização) e descreve o container *cli* (usado na interação com a rede) que estará a correr na *Org1*.

```

85 peer0.org4.jnvr.com:
86 # ---CHANGED--- Container name - same as the peer name
87 container_name: peer0.org4.jnvr.com
88 extends:
89   file: base/docker-compose-base.yaml
90   # ---CHANGED--- Refers to peer name
91   service: peer0.org4.jnvr.com
92 environment:
93   - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
94   - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb3:5984
95   - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
96   - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
97 depends_on:
98   - orderer.jnvr.com
99   - couchdb3
100 networks:
101   # ---CHANGED--- our network is called "basic"
102   - basic

```

Figura 61 - Excerto do ficheiro *docker-compose-cli-couchdb.yaml* a definir um peer de uma organização

```

105 ca.org1.jnvr.com:
106   image: hyperledger/fabric-ca:1.4.8
107   environment:
108     - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
109     - FABRIC_CA_SERVER_CA_NAME=ca.org1.jnvr.com
110     - FABRIC_CA_SERVER_TLS_ENABLED=true
111     - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.jnvr.com-cert.pem
112     - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/priv_sk
113     - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.jnvr.com-cert.pem
114     - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/priv_sk
115   ports:
116     - "7054:7054"
117   command: sh -c 'fabric-ca-server start -b admin:adminpw'
118   volumes:
119     - ./crypto-config/peerOrganizations/org1.jnvr.com/ca/:/etc/hyperledger/fabric-ca-server-config
120   container_name: ca.org1.jnvr.com
121   networks:
122     - basic

```

Figura 62 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir a configuração do CA da organização 1

```

170 couchdb0:
171   image: couchdb:3.1
172   environment:
173     - COUCHDB_USER=admin
174     - COUCHDB_PASSWORD=adminpw
175   ports:
176     - 5984:5984
177   container_name: couchdb0
178   networks:
179     - basic

```

Figura 63 - Excerto do ficheiro docker-compose-cli-couchdb.yaml a definir uma das bases de dados

```

124 cli:
125   container_name: cli
126   image: hyperledger/fabric-tools:2.2
127   tty: true
128   stdin_open: true
129   environment:
130     - GOPATH=/opt/gopath
131     - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
132     - FABRIC_LOGGING_SPEC=DEBUG
133     - CORE_PEER_ID=cli
134     # ---CHANGED--- peer0 from Org1 is the default for this CLI container
135     - CORE_PEER_ADDRESS=peer0.org1.jnvr.com:7051
136     - CORE_PEER_LOCALMSPID=Org1MSP
137     - CORE_PEER_TLS_ENABLED=true
138     # ---CHANGED--- changed to reflect peer0 name, org1 name and our company's domain
139     - CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/server.crt
140     # ---CHANGED--- changed to reflect peer0 name, org1 name and our company's domain
141     - CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/server.key
142     # ---CHANGED--- changed to reflect peer0 name, org1 name and our company's domain
143     - CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/ca.crt
144     # ---CHANGED--- changed to reflect peer0 name, org1 name and our company's domain
145     - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/users/Admin@org1.jnvr.com/msp
146   working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
147   # ---CHANGED--- command needs to be connected out as we will be issuing commands explicitly, not using by any script
148   # command: /bin/bash -c './scripts/script.sh ${CHANNEL_NAME}; sleep $TIMEOUT'
149   command: /bin/bash
150   volumes:
151     - /var/run:/host/var/run/
152     # ---CHANGED--- chaincode path adjusted
153     - ../chaincode:/opt/gopath/src/github.com/chaincode
154     - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
155     - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts
156   depends_on:
157     # ---CHANGED--- reference to our orderer
158     - orderer.jnvr.com
159     # ---CHANGED--- reference to peer0 of Org1
160     - peer0.org1.jnvr.com
161     # ---CHANGED--- reference to peer0 of Org2
162     - peer0.org2.jnvr.com
163     # ---CHANGED--- reference to peer0 of Org3
164     - peer0.org3.jnvr.com
165     # ---CHANGED--- reference to peer0 of Org4
166     - peer0.org4.jnvr.com
167   networks:
168     # ---CHANGED--- our network is called "basic"
169     - basic

```

Figura 64 - Excerto do ficheiro `docker-compose-cli-couchdb.yaml` a definir configuração de cli

- Adicionalmente é necessário criar o documento `docker-compose-base.yaml` na diretoria base, que é mencionado anteriormente na definição dos containers das organizações. Neste documento é definido em maior detalhe informações sobre a inicialização dos `dockers` das organizações e onde são guardados os materiais criados.

```

39 peer0.org1.jnvr.com:
40   # ---CHANGED--- Container name - same as the peer name
41   container_name: peer0.org1.jnvr.com
42   extends:
43     file: peer-base.yaml
44     service: peer-base
45   environment:
46     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
47     - CORE_PEER_ID=peer0.org1.jnvr.com
48     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
49     - CORE_PEER_ADDRESS=peer0.org1.jnvr.com:7051
50     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
51     - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.jnvr.com:7051
52     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
53     - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.jnvr.com:7051
54     - CORE_PEER_LOCALMSPID=Org1MSP
55   volumes:
56     - /var/run:/host/var/run/
57     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
58     - ../crypto-config/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/msp:/etc/hyperledger/fabric/msp
59     # ---CHANGED--- changed to reflect peer name, org name and our company's domain
60     - ../crypto-config/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls:/etc/hyperledger/fabric/tls
61   ports:
62     - 7051:7051
63     - 7053:7053

```

Figura 65 - Excerto do ficheiro `docker-compose-base.yaml` a definir um peer de uma organização

- 5) Novamente, menciona-se outro ficheiro denominado *peer-base.yaml*, que vai ser necessário criar, ainda na diretoria *base*. Este ficheiro é usado para estabelecer os diferentes modos de trabalho, a diretoria onde a rede trabalha e a localização de ficheiros chave para a rede.

```
6  version: '2'
7
8  services:
9    peer-base:
10     image: hyperledger/fabric-peer:2.2.0
11     environment:
12       - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
13       # the following setting starts chaincode containers on the same
14       # bridge network as the peers
15       # https://docs.docker.com/compose/networking/
16       # ---CHANGED---
17       - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=jnvr-network_basic
18       - FABRIC_LOGGING_SPEC=INFO
19       # - FABRIC_LOGGING_SPEC=DEBUG
20       - CORE_PEER_TLS_ENABLED=true
21       - CORE_PEER_GOSSIP_USELEADERELECTION=true
22       - CORE_PEER_GOSSIP_ORGLEADER=false
23       - CORE_PEER_PROFILE_ENABLED=true
24       - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
25       - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
26       - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
27     working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
28     command: peer node start
```

Figura 66 - Excerto do ficheiro *peer-base.yaml* a definir o comportamento de um peer

- 6) Tendo os ficheiros criados, correm-se os comandos

```
export CHANNEL_NAME=channel1
```

```
export VERBOSE=false
```

```
export FABRIC_CFG_PATH=$PWD
```

Seguidos por `CHANNEL_NAME=$CHANNEL_NAME docker-compose -f docker-compose-cli-couchdb.yaml up -d`, que irá iniciar os containers da rede.

```

osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ CHANNEL_NAME=channel1 docker-compose -f docker-compose-cli-couchdb.yaml up -d
[+] Running 12/12
  # Network jnvr-network-basic Created 0.1s
  # Container couchdb0 Started 2.0s
  # Container couchdb1 Started 2.0s
  # Container couchdb3 Started 2.0s
  # Container orderer-jnvr.com Started 1.7s
  # Container couchdb2 Started 1.8s
  # Container ca.org1.jnvr.com Started 1.7s
  # Container peer0.org3.jnvr.com Start... 3.4s
  # Container peer0.org2.jnvr.com Start... 4.1s
  # Container peer0.org4.jnvr.com Start... 4.1s
  # Container peer0.org1.jnvr.com Start... 4.0s
  # Container cli Started 5.3s
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-networks$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
8a943ab9af66       hyperledger/fabric-tools:2.2    "/bin/bash"        10 seconds ago     Up 4 seconds
3d40123dd577       hyperledger/fabric-peer:2.2.0    "peer node start"  11 seconds ago     Up 6 seconds       0.0.0.0:18051->7051/tcp, :::18051->7051/tcp, 0.0.0.0:18053->7053/tcp, :::18053->7053/tcp
35d8af7d1d98       hyperledger/fabric-peer:2.2.0    "peer node start"  11 seconds ago     Up 6 seconds       0.0.0.0:18051->7051/tcp, :::18051->7051/tcp, 0.0.0.0:18053->7053/tcp, :::18053->7053/tcp
b406d16a154c       hyperledger/fabric-peer:2.2.0    "peer node start"  11 seconds ago     Up 6 seconds       0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp
26edab7eb04b       hyperledger/fabric-peer:2.2.0    "peer node start"  11 seconds ago     Up 6 seconds       0.0.0.0:9051->7051/tcp, :::9051->7051/tcp, 0.0.0.0:9053->7053/tcp, :::9053->7053/tcp
acd72b8935a1       couchdb:3.1          "tini -- /docker-ent..."  11 seconds ago     Up 8 seconds       4369/tcp, 9100/tcp, 0.0.0.0:5986->5984/tcp, :::5986->5984/tcp
33776127c33a       couchdb:3.1          "tini -- /docker-ent..."  11 seconds ago     Up 8 seconds       4369/tcp, 9100/tcp, 0.0.0.0:5987->5984/tcp, :::5987->5984/tcp
297bcfb5b5f3       hyperledger/fabric-ca:1.4.8      "sh -c 'fabric-ca-se..."  11 seconds ago     Up 8 seconds       0.0.0.0:7054->7054/tcp, :::7054->7054/tcp
04424ecc6a10       hyperledger/fabric-orderer:2.2.0 "orderer"           11 seconds ago     Up 8 seconds       0.0.0.0:7050->7050/tcp, :::7050->7050/tcp
b592f36259c2       couchdb:3.1          "tini -- /docker-ent..."  11 seconds ago     Up 8 seconds       4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp, :::5984->5984/tcp
036ee170376       couchdb:3.1          "tini -- /docker-ent..."  11 seconds ago     Up 8 seconds       4369/tcp, 9100/tcp, 0.0.0.0:5985->5984/tcp, :::5985->5984/tcp

```

Figura 67 - Inicialização dos containers da rede

- 7) Após o início dos containers da rede, ir-se-á entrar no container *cli*, criado para ser possível interagir com a rede, com o comando `docker exec -it cli /bin/sh`.
- 8) Dentro de *cli* é necessário voltar a exportar o nome do canal com o comando `export CHANNEL_NAME=channel1`.
- 9) Depois cria-se o canal usando o seguinte comando.

```
peer channel create -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f
./channel-artifacts/channel.tx --tls true --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.j
nvr.com-cert.pem
```

```

osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ docker exec -it cli /bin/sh
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CHANNEL_NAME=channel1
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel create -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
2022-04-14 16:56:10.030 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 16:56:10.036 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 16:56:10.063 UTC [bccsp] GetDefault -> DEBU 003 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 16:56:10.070 UTC [bccsp_sw] openKeyStore -> DEBU 004 KeyStore opened at [/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/users/Admin@org1.jnvr.com/msp/keystore]...done
2022-04-14 16:56:10.120 UTC [grpc] Infof -> DEBU 03a Subchannel Connectivity change to READY
2022-04-14 16:56:10.121 UTC [grpc] UpdateSubConnState -> DEBU 03b pickfirstBalancer: HandleSubConnStateChange: 0xc000487000, {READY <nil>}
2022-04-14 16:56:10.121 UTC [grpc] Infof -> DEBU 03c Channel Connectivity change to READY
2022-04-14 16:56:10.158 UTC [msp.identity] Sign -> DEBU 03d Sign: plaintext: 0AE0060A1408051A0608AAA4E1920622..B42970C50BAB12080A021A0012021A00
2022-04-14 16:56:10.158 UTC [msp.identity] Sign -> DEBU 03e Sign: digest: 48754C344C3B07BC6101B524C02CC9868FC72D70DD38D94922BED9651880777
2022-04-14 16:56:10.185 UTC [cli.common] readBlock -> INFO 03f Received block: 0

```

Figura 68 - Excertos da criação do canal

Para verificar que o comando executou corretamente basta utilizar o comando `ls` e verificar que se criou o ficheiro, neste caso, *channel1.block*. Se o nome do canal fosse diferente, aparecia o nome de canal escolhido.

- 10) Seguidamente juntam-se as organizações 1, 2, 3 e 4 ao canal através dos seguintes comandos, onde se utiliza o ficheiro do tipo `.block` obtido anteriormente.

```
peer channel join -b channel1.block
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org2.jnvr.com:7051  
CORE_PEER_LOCALMSPID="Org2MSP"  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt peer channel join -b channel1.block
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051  
CORE_PEER_LOCALMSPID="Org3MSP"  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer channel join -b channel1.block
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051  
CORE_PEER_LOCALMSPID="Org4MSP"  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel join -b channel1.block
```

```

/opt/gopath/src/github.com/hyperledger/fabric/peer # CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER_LOCALMSPID="Org4MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel join -b channel1.block
2022-04-14 17:07:04.281 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-14 17:07:04.282 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-14 17:07:04.294 UTC [bccsp] GetDefault -> DEBU 003 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-14 17:07:04.298 UTC [bccsp_sw] openKeyStore -> DEBU 004 KeyStore opened at [/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/keystore]...done
2022-04-14 17:07:04.311 UTC [grpc] Infof -> DEBU 024 Channel Connectivity change to CONNECTING
2022-04-14 17:07:04.319 UTC [comm.tls] ClientHandshake -> DEBU 025 Client TLS handshake completed in 2.936276ms remoteaddress=172.23.0.9:7051
2022-04-14 17:07:04.320 UTC [grpc] Infof -> DEBU 026 Subchannel Connectivity change to READY
2022-04-14 17:07:04.323 UTC [grpc] UpdateSubConnState -> DEBU 027 pickfirstBalancer: HandleSubConnStateChange: 0xc0000f2230, {READY <nil>}
2022-04-14 17:07:04.323 UTC [grpc] Infof -> DEBU 028 Channel Connectivity change to READY
2022-04-14 17:07:04.323 UTC [channelCmd] InitCmdFactory -> INFO 029 Endorser and orderer connections initialized
2022-04-14 17:07:04.329 UTC [msp.identity] Sign -> DEBU 02a Sign: plaintext: 0AA8070A5C08011A0C08B8A9E1920610...7EE7F08D1A0A0A000A000A000A000A000A000
2022-04-14 17:07:04.329 UTC [msp.identity] Sign -> DEBU 02b Sign: digest: D4265868770066C762F096D9231EBC55B05506993B3A9AAA786E3EE81D3D5E12
2022-04-14 17:07:04.794 UTC [channelCmd] executeJoin -> INFO 02c Successfully submitted proposal to join channel

```

Figura 69 - Excertos da junção de uma organização ao canal

Deve-se notar que devido a *cli* estar a correr na organização 1, não é necessária toda a definição de variáveis que se verifica para as outras organizações.

- 11) De seguida utilizam-se os seguintes comandos para criar os *anchor peers* para cada uma das organizações.

```

peer channel update -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f
./channel-artifacts/Org1MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.j
nvr.com-cert.pem

```

```

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.jnvr.
com/msp/ CORE_PEER_ADDRESS=peer0.org2.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.j
nvr.com/tls/ca.crt peer channel update -o orderer.jnvr.com:7050 -c
$CHANNEL_NAME -f ./channel-artifacts/Org2MSPanchors.tx --tls --
cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.j
nvr.com-cert.pem

```



```

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org3MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer channel update -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org3MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem

```

```

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org4MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel update -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org4MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem

```

```

/opt/gopath/src/github.com/hyperledger/fabric/peer # CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER_LOCALMSPID="Org4MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel update -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org4MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
2022-04-14 17:09:41.966 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 17:09:41.968 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 17:09:41.983 UTC [bccsp] GetDefault -> DEBU 003 Before using BCCSP, please call InitFactories (). Falling back to bootBCCSP.
2022-04-14 17:09:41.989 UTC [bccsp sw] openKeyStore -> DEBU 004 KeyStore opened at [/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/keystore]...done
2022-04-14 17:09:41.997 UTC [grpc] Infof -> DEBU 025 ClientConn switching balancer to "pick first"
2022-04-14 17:09:41.997 UTC [grpc] Infof -> DEBU 026 Channel switches to new LB policy "pick first"
2022-04-14 17:09:41.997 UTC [grpc] Infof -> DEBU 027 Subchannel Connectivity change to CONNECTING
2022-04-14 17:09:41.998 UTC [grpc] Infof -> DEBU 028 Subchannel picks a new address "orderer.jnvr.com:7050" to connect
2022-04-14 17:09:42.000 UTC [grpc] UpdateSubConnState -> DEBU 029 pickfirstBalancer: HandleSubConnStateChange: 0xc00002fa40, {CONNECTING <nil>}
2022-04-14 17:09:42.000 UTC [grpc] Infof -> DEBU 02a Channel Connectivity change to CONNECTING
2022-04-14 17:09:42.006 UTC [comm.tls] ClientHandshake -> DEBU 02b Client TLS handshake completed in 3.101788ms remoteaddress=172.23.0.7:7050
2022-04-14 17:09:42.009 UTC [grpc] Infof -> DEBU 02c Subchannel Connectivity change to READY
2022-04-14 17:09:42.010 UTC [grpc] UpdateSubConnState -> DEBU 02d pickfirstBalancer: HandleSubConnStateChange: 0xc00002fa40, {READY <nil>}
2022-04-14 17:09:42.010 UTC [grpc] Infof -> DEBU 02e Channel Connectivity change to READY
2022-04-14 17:09:42.039 UTC [channelCmd] update -> INFO 02f Successfully submitted channel update

```

Figura 70 - Excertos da criação de um anchor peer numa organização

Volta-se a observar que na organização 1 não é necessário a definição das variáveis, que se verificam em todas as outras organizações, tal como quando se juntaram as organizações ao canal, anteriormente.

12) Neste exemplo vai ser utilizado um *chaincode* em linguagem *go*, que está localizado na diretoria *chaincode* mencionada no início do exemplo. Nessa diretoria cria-se uma pasta com o nome do *chaincode*, que neste caso se denomina *foodcontrol*.

Dentro da pasta anterior cria-se um ficheiro do tipo *.go* com o nome do *chaincode*, ou seja, *foodcontrol.go* com o seguinte conteúdo, onde estão funções para inicializar valores em cada *peer*, permitir realização de pesquisas, introdução e edição de novos valores.

```
1 package main
2
3 import (
4     "encoding/json"
5     "fmt"
6     "github.com/hyperledger/fabric-contract-api-go/contractapi"
7 )
8
9 // SmartContract provides functions for control the food
10 type SmartContract struct {
11     contractapi.Contract
12 }
13
14 //Food describes basic details of what makes up a food
15 type Food struct {
16     Farmer string `json:"farmer"`
17     Variety string `json:"variety"`
18 }
19
20 func (s *SmartContract) Set(ctx contractapi.TransactionContextInterface, foodId string, farmer string, variety string) error {
21
22     //Validaciones de sintaxis
23
24     //Validaciones de negocio
25
26     food := Food{
27         Farmer: farmer,
28         Variety: variety,
29     }
30
31     foodAsBytes, err := json.Marshal(food)
32     if err != nil {
33         fmt.Printf("Marshal error: %s", err.Error())
34         return err
35     }
36
37     return ctx.GetStub().PutState(foodId, foodAsBytes)
38 }
39
40 func (s *SmartContract) Query(ctx contractapi.TransactionContextInterface, foodId string) (*Food, error) {
41
42     foodAsBytes, err := ctx.GetStub().GetState(foodId)
43
44     if err != nil {
45         return nil, fmt.Errorf("Failed to read from world state. %s", err.Error())
46     }
47
48     if foodAsBytes == nil {
49         return nil, fmt.Errorf("%s does not exist", foodId)
50     }
51
52     food := new(Food)
53
54     err = json.Unmarshal(foodAsBytes, food)
55     if err != nil {
56         return nil, fmt.Errorf("Unmarshal error. %s", err.Error())
57     }
58
59     return food, nil
60 }
61
62 func main() {
63
64     chaincode, err := contractapi.NewChaincode(new(SmartContract))
65
66     if err != nil {
67         fmt.Printf("Error al crear el chaincode: %s", err.Error())
68         return
69     }
70
71     if err := chaincode.Start(); err != nil {
72         fmt.Printf("Error al crear el: %s", err.Error())
73     }
74 }
```

Figura 71 - Documento *foodcontrol.go*

- 13) Depois cria-se `go.mod` que identifica as versões e *packages* necessários para correr o ficheiro `.go`, neste caso `foodcontrol.go`.

```
1 module foodcontrol
2
3 go 1.13
4
5 require github.com/hyperledger/fabric-contract-api-go v1.1.0
6
```

Figura 72 - Documento `go.mod`

- 14) Após a criação dos ficheiros fazem-se os seguintes exports no terminal, ainda dentro de cli.

```
export CHANNEL_NAME=channel1
export CHAINCODE_NAME=foodcontrol
export CHAINCODE_VERSION=1
export CC_RUNTIME_LANGUAGE=golang
export CC_SRC_PATH="../../chaincode/${CHAINCODE_NAME}/"
export
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/
tlsca.jnvr.com-cert.pem
```

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CHANNEL_NAME=channel1
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CHAINCODE_NAME=js_chain
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CHAINCODE_VERSION=1
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CC_RUNTIME_LANGUAGE=node
/opt/gopath/src/github.com/hyperledger/fabric/peer # export CC_SRC_PATH="../../chaincode/${CHAINCODE_NAME}/"
```

Figura 73 - Execução dos exports dentro de cli

- 15) Passando para a génese do package com o *chaincode* a ser instalado em cada peer utiliza-se o seguinte comando, que faz uso dos *exports* realizados anteriormente.

```
peer lifecycle chaincode package ${CHAINCODE_NAME}.tar.gz --path
${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label
${CHAINCODE_NAME}_${CHAINCODE_VERSION} >&log.txt
```

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer lifecycle chaincode package ${CHAINCODE_NAME}.tar.gz --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label ${CHAINCODE_NAME}_${CHAINCODE_VERSION} >&log.txt
```

Figura 74 – Criação do package com o *chaincode* a instalar nos peers

- 16) Seguidamente pode-se verificar os resultados utilizando o comando `ls`, de onde se verifica a criação de dois ficheiros, um denominado `log.txt` (onde estão os `logs` do funcionamento do comando) e outro chamado `<nome do chaincode>.tar.gz`, neste caso `foodcontrol.tar.gz` (`chaincode` a ser instalado em cada `peer`).

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # ls
channel-artifacts  crypto          log.txt
channel1.block     js chain.tar.gz
```

Figura 75 - Verificação da criação do package do chaincode

- 17) Finalmente instala-se o `chaincode` em cada `peer` com os seguintes comandos.

```
peer lifecycle chaincode install foodcontrol.tar.gz
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org2.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt peer lifecycle chaincode install
foodcontrol.tar.gz
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org3MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer lifecycle chaincode install
foodcontrol.tar.gz
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org4MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincode install
foodcontrol.tar.gz
```



```

/opt/gopath/src/github.com/hyperledger/fabric/peer # CORE_PEER_MSPCONFIGPATH=/opt/gopath/s
rc/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@o
rg4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER_LOCALMSPID="Org4MSP
" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/pe
erOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincod
e install js_chain.tar.gz
2022-04-16 13:36:46.548 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-16 13:36:46.550 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-16 13:36:46.562 UTC [bccsp] GetDefault -> DEBU 003 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-04-16 13:36:46.571 UTC [bccsp_sw] openKeyStore -> DEBU 004 KeyStore opened at [/opt/gopath/src/github.com/hyperledger/fabric/peer/cryp
to/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/keystore]...done
2022-04-16 13:36:46.571 UTC [msp] getPemMaterialFromDir -> DEBU 005 Reading directory /opt/gopath/src/github.com/hyperledger/fabric/peer/cr
ypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/signcerts
2022-04-16 13:36:46.571 UTC [msp] getPemMaterialFromDir -> DEBU 006 Inspecting file /opt/gopath/src/github.com/hyperledger/fabric/peer/cryp
to/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/signcerts/Admin@org4.jnvr.com-cert.pem
2022-04-16 13:36:46.599 UTC [comm.tls] ClientHandshake -> DEBU 032 Client TLS handshake completed in 2.841933ms remoteaddress=172.24.0.9:70
51
2022-04-16 13:36:46.600 UTC [grpc] Infof -> DEBU 033 Subchannel Connectivity change to READY
2022-04-16 13:36:46.600 UTC [grpc] UpdateSubConnState -> DEBU 034 pickfirstBalancer: HandleSubConnStateChange: 0xc00015fa40, {READY <nil>}
2022-04-16 13:36:46.600 UTC [grpc] Infof -> DEBU 035 Channel Connectivity change to READY
2022-04-16 13:36:46.603 UTC [msp.identity] Sign -> DEBU 036 Sign: plaintext: 0AAE070A6208031A0C08EE8CEB920610...863200040000FFFF3C9CA3D4001
00000
2022-04-16 13:36:46.604 UTC [msp.identity] Sign -> DEBU 037 Sign: digest: F73652D8F4C5046AD7D8E996D6A6E77A08D5601CBD536EC6741BF4615A3D3F4F
nKjs_chain_1:04140a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78eaf5c2e903f2\022\njs_chain_1" >
2022-04-16 13:36:55.718 UTC [cli.lifecycle.chaincode] submitInstallProposal -> INFO 038 Installed remotely: response:<status:200 payload:"\
a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78eaf5c2e903f2

```

Figura 76 - Excertos da instalação do chaincode em um dos peers

Destes comandos é importante registrar o *package identifier* que aparece no final do comando. No caso desta implementação é:

```
foodcontrol_1:04140a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78eaf5c2e903f2
```

- 18) Tendo o *package identifier* pode-se aprovar a definição do *chaincode* para cada organização. Neste caso como se pretende manter a Organização 2 em modo *read-only*, esta organização não irá ter o seu *chaincode* aprovado. Isto vai ser realizado com os seguintes comandos.

```
peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA
--channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version
$CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy
"OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --package-id
<package identifier>
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.
com/msp/ CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org3MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.j
nvr.com/tls/ca.crt peer lifecycle chaincode approveformyorg --tls --
cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME
--version $CHAINCODE_VERSION --sequence 1 --waitForEvent --
signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')
--package-id <package identifier>
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.
```

```

com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org4MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --package-id <package identifier>

```

Sabendo o valor do *package identifier* do passo anterior, o comando fica da seguinte forma.

```

peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --package-id foodcontrol_1:04140a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78eaf5c2e903f2

```

```

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org3MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --package-id foodcontrol_1:04140a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78eaf5c2e903f2

```

```

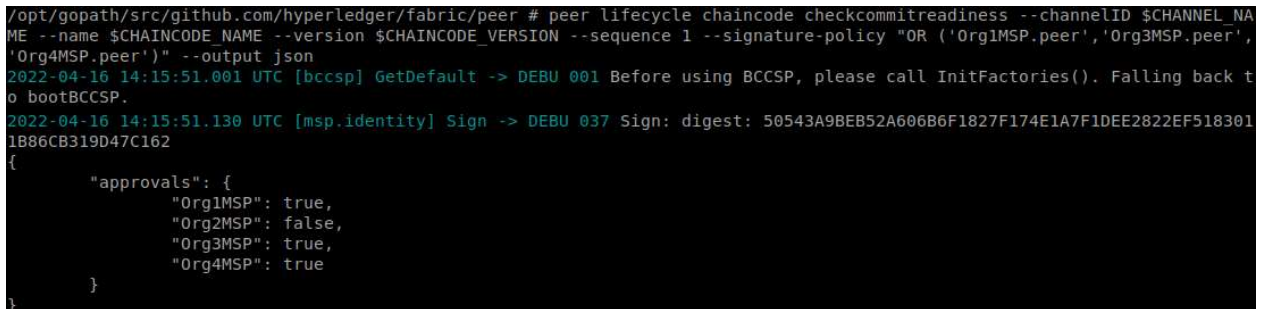
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.jnvr.com/msp/ CORE_PEER_ADDRESS=peer0.org4.jnvr.com:7051
CORE_PEER_LOCALMSPID="Org4MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --package-id

```

```
foodcontrol_1:04140a7bd8625ae0ec79e9f51a34571ccaf0d7121bb608907b78ea  
f5c2e903f2
```

Para verificar que apenas a Organização 2 não está aprovada corre-se o comando.

```
peer lifecycle chaincode checkcommitreadiness --channelID  
$CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --  
sequence 1 --signature-policy "OR  
( 'Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer' )" --output json.
```



```
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer')" --output json  
2022-04-16 14:15:51.001 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.  
2022-04-16 14:15:51.130 UTC [msp.identity] Sign -> DEBU 037 Sign: digest: 50543A98EB52A606B6F1827F174E1A7F1DEE2822EF5183011B86CB319D47C162  
{  
  "approvals": {  
    "Org1MSP": true,  
    "Org2MSP": false,  
    "Org3MSP": true,  
    "Org4MSP": true  
  }  
}
```

Figura 77 - Excertos da verificação da aprovação das organizações

Onde se verifica que a Organização 2 não tem *approval*.

- 19) Para fazer o *commit* do *chaincode* usa-se o seguinte comando, onde se volta a verificar a ausência da Organização 2 devido a esta ser *read-only*.

```
peer lifecycle chaincode commit -o orderer.jnvr.com:7050 --tls --  
cafile $ORDERER_CA --peerAddresses peer0.org1.jnvr.com:7051 --  
tlsRootCertFiles  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani  
zations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/ca.crt --  
peerAddresses peer0.org3.jnvr.com:7051 --tlsRootCertFiles  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani  
zations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt --  
peerAddresses peer0.org4.jnvr.com:7051 --tlsRootCertFiles  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani  
zations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt --  
channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version  
$CHAINCODE_VERSION --sequence 1 --signature-policy "OR  
( 'Org1MSP.peer', 'Org3MSP.peer', 'Org4MSP.peer' )"
```

```

/opt/gopath/src/github.com/hyperledger/fabric/peer # peer lifecycle chaincode commit -o orderer.jnvr.com:7050 --tls --cafile
le $ORDERER_CA --peerAddresses peer0.org1.jnvr.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/p
eer/crypto/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/ca.crt --peerAddresses peer0.org3.jnvr.com:7051 -
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.or
g3.jnvr.com/tls/ca.crt --peerAddresses peer0.org4.jnvr.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/
fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt --channelID $CHANNEL_NAME --name $
CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.pe
er')"
2022-04-16 14:42:42.009 UTC [msp.identity] Sign -> DEBU 081 Sign: digest: 4D702565538E8448E322EF40B8828A65FB1350E05CE4B1AB
F402A12CC4272146
2022-04-16 14:42:44.298 UTC [chaincodeCmd] ClientWait -> INFO 082 txid [a767edfd001464c7f58a420e06a59787b4fad114b9d94f5a46
91834263dcfd02] committed with status (VALID) at peer0.org3.jnvr.com:7051
2022-04-16 14:42:44.614 UTC [chaincodeCmd] ClientWait -> INFO 083 txid [a767edfd001464c7f58a420e06a59787b4fad114b9d94f5a46
91834263dcfd02] committed with status (VALID) at peer0.org4.jnvr.com:7051
2022-04-16 14:42:44.676 UTC [chaincodeCmd] ClientWait -> INFO 084 txid [a767edfd001464c7f58a420e06a59787b4fad114b9d94f5a46
91834263dcfd02] committed with status (VALID) at peer0.org1.jnvr.com:7051

```

Figura 78 - Excertos do commit do chaincode

20) Após ter o *chaincode* preparado utiliza-se o seguinte comando para utilizar a função *Set* definida no *chaincode* e para criar no *ledger* o valor de *id:3*.

```

peer chaincode invoke -o orderer.jnvr.com:7050 --tls --cafile
$ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c
'{"Args":["Set","id:3","jorge","naranja"]}'

```

```

/opt/gopath/src/github.com/hyperledger/fabric/peer # peer chaincode invoke -o orderer.jnvr.com:7050 --tls --cafile $OR
DERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","id:3","jorge","naranja"]}'

```

Figura 79 - Utilização da função set do chaincode

21) De modo a observar os resultados deste comando pode-se ir ao url `localhost:5984/_utils/` em qualquer navegador de Internet, podendo assim aceder à base de dados *couchdb* do *peer1*.

Nessa página será necessário introduzir o nome de utilizador e password do *couchdb*, que foram definidos no *setup* da rede.

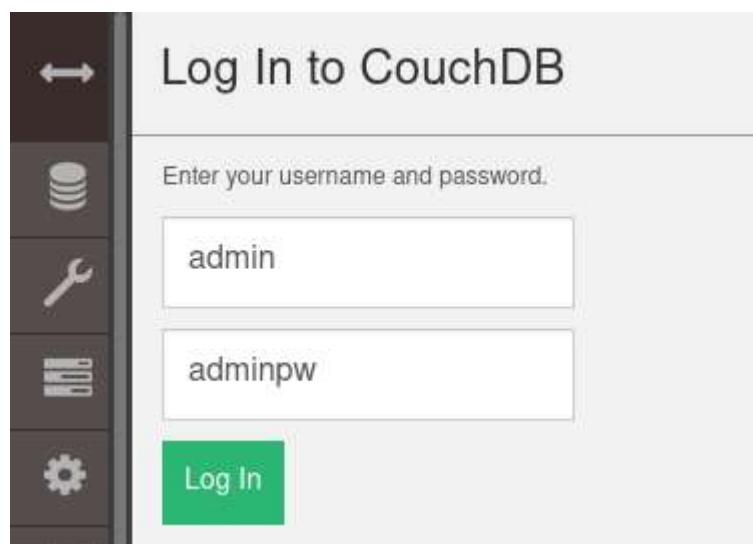


Figura 80 - Página inicial de couchdb

Após o log in no *couchdb* irá ser apresentada a seguinte página onde se observa o *chaincode* instalado.

Name	Size	# of Docs
<i>_replicator</i>	2.3 KB	1
<i>_users</i>	2.3 KB	1
channel1_	155.5 KB	3
channel1_lifecycle	2.1 KB	5
channel1_lifecycle\$\$h_implicit_org_\$org1\$m\$\$p	2.6 KB	6
channel1_lifecycle\$\$h_implicit_org_\$org2\$m\$\$p	0 bytes	0
channel1_lifecycle\$\$h_implicit_org_\$org3\$m\$\$p	2.6 KB	6
channel1_lifecycle\$\$h_implicit_org_\$org4\$m\$\$p	2.6 KB	6
channel1_lifecycle\$\$p_implicit_org_\$org1\$m\$\$p	2.5 KB	6
channel1_foodcontrol	310 bytes	1
channel1_lsc	0 bytes	0
fabric_internal	291 bytes	1

Figura 81 - Chaincode em couchdb

Ao entrar em *channel1_foodcontrol* observa-se o *id:3* e os valores definidos criados com o comando.

id	key	value
id:3	id:3	{ "rev": "1-923ec604d07d33f69891b2221370cf1" }

Figura 82 - Valor previamente criado, guardado em couchdb

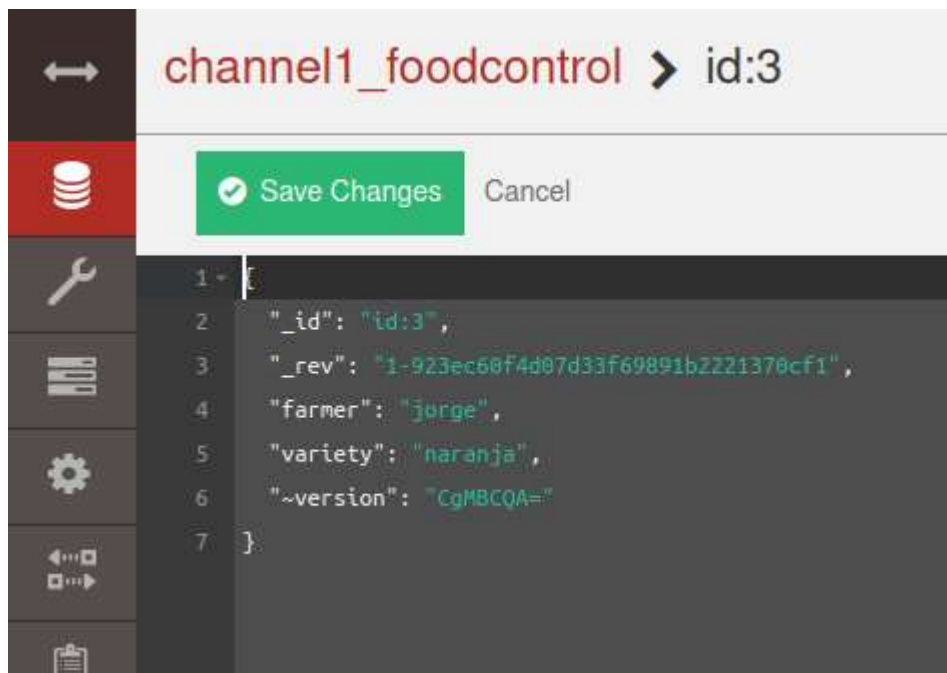


Figura 83 - Detalhes do valor criado, guardado em couchdb

5.2.3.4 Criação de scripts para conceber uma rede com quatro organizações

Neste caso usa-se a estrutura de pastas apresentadas na implementação anterior (secção 5.2.3.3) com os mesmos documentos *.yaml* de configuração.

Tendo estes documentos de configuração cria-se scripts de modo a inicializar uma rede com quatro organizações, que é explicada de seguida:

- 1) Inicialmente irá ser criado o ficheiro *script_jnvr_network.sh* na diretoria *jnvr_network* que se pode observar de seguida.


```

home > osboxes > Desktop > jnvr_net > jnvr-network > $ script-jnvr-network.sh
1  #!/bin/sh
2
3  #This scripts assumes it is being run in the jnvr_network directory.
4  #There also is the assumption that the structure of the example is being followed.
5
6  #####!!!!ATTENTION!!!!#####
7  #This script eliminates ALL DOCKER CONTAINERS AND VOLUMES IN YOUR MACHINE, meaning if
8  #there is a network already running, THAT NETWORK'S CONTAINERS AND VOLUMES will be DELETED
9  #####
10
11  rm -r ./channel-artifacts
12  mkdir ./channel-artifacts
13  sudo rm -r ./crypto-config
14  mkdir ./crypto-config
15  docker rm -f $(docker ps -aq)
16  docker volume rm -f $(docker volume ls -q)
17  NETWORK_NAME=$(grep 'CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE' /home/osboxes/Desktop/jnvr_net/jnvr-network/base/peer-base.yaml)
18  #IFS variable is changed in order to modify how Bash recognizes word boundaries while splitting a sequence of character strings, normally it is set to ' ' as I've defined it is set to '*'
19  old_IFS=$IFS
20  IFS==
21  #Here doc is being used here to pass the value of NETWORK_NAME as an input to the read comand, EOF here functions as a delimiter token
22  read -r thing NETWORK_NAME <<EOF
23  $NETWORK_NAME
24  EOF
25  IFS=$old_IFS
26  docker network rm $NETWORK_NAME
27
28
29  #Generate Cryptographic Material:
30  cryptogen generate --config=./crypto-config.yaml
31
32  #Create Genesis Block
33  configtxgen -profile FourOrgsOrdererGenesis -channelID system-channel -outputBlock ./channel-artifacts/genesis.block
34  configtxgen -profile FourOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID channel1
35  configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID channel1 -asOrg Org1MSP
36  configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID channel1 -asOrg Org2MSP
37  configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -channelID channel1 -asOrg Org3MSP
38  configtxgen -profile FourOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org4MSPanchors.tx -channelID channel1 -asOrg Org4MSP
39
40  #Start Docker containers
41  export CHANNEL_NAME=channel1
42  export VERBOSE=false
43  export FABRIC_CFG_PATH=$PWD
44  CHANNEL_NAME=${CHANNEL_NAME} docker-compose -f docker-compose-cli-couchdb.yaml up -d
45
46  #Copy Docker Script into container cli
47  docker cp ./script-jnvr-network-docker.sh cli:/opt/gopath/src/github.com/hyperledger/fabric/peer
48
49  #Initiate cli shell
50  docker exec -it cli /bin/sh

```

Figura 84 - Documento *script-jnvr-network.sh*

Neste script removem-se quaisquer ficheiros previamente criados na configuração da *jnvr_network*, todos os containers e volumes presentes na máquina e irá ser removida a rede previamente criada com a configuração presente na diretoria. Caso o script corra numa máquina nova, nada será feito nesta fase inicial.

De seguida irão ser corridos os comandos apresentados na implementação anterior (secção 5.2.3.3) até à criação dos *containers*.

Finalmente, o *script_jnvr_network_docker.sh* irá ser copiado para o *container cli* e entra-se no container.

- 2) Passando para a criação de *script_jnvr_network_docker.sh*, que também se encontra na diretoria *jnvr_network*, observa-se de seguida a sua constituição.

```

1 #!/bin/sh
2
3 #Create the Channel
4 export CHANNEL_NAME=channel1
5 peer channel create -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls true --cafile /opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
6
7 #Join Org1 Org2 Org3 and Org4 to the Channel
8 peer channel join -b ${CHANNEL_NAME}.block
9 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org2.jnvr.com:7051 CORE_PEER LOCALMSPID="Org2MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt peer channel join -b ${CHANN
EL_NAME}.block
10 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org3.jnvr.com:7051 CORE_PEER LOCALMSPID="Org3MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer channel join -b ${CHANN
EL_NAME}.block
11 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER LOCALMSPID="Org4MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel join -b ${CHANN
EL_NAME}.block
12
13 #Set the Anchors Peers in all Orgs
14 peer channel update -o orderer.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /opt/gopath/src
github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.
pem
15 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org2.jnvr.com:7051 CORE_PEER LOCALMSPID="Org2MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt peer channel update -o order
er.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
16 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org3.jnvr.com:7051 CORE_PEER LOCALMSPID="Org3MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer channel update -o order
er.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org3MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
17 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER LOCALMSPID="Org4MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer channel update -o order
er.jnvr.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org4MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem
18
19 #Make chaincode package
20 export CHANNEL_NAME=channel1
21 export CHAINCODE_NAME=foodcontrol
22 export CHAINCODE_VERSION=1
23 export CC_RUNTIME_LANGUAGE=golang
24 export CC_SRC_PATH="./../chaincode/${CHAINCODE_NAME}/"
25 export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/
msp/tlscacerts/tlsca.jnvr.com-cert.pem
26 peer lifecycle chaincode package ${CHAINCODE_NAME}.tar.gz --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label ${CHAINCODE_NA
ME} ${CHAINCODE_VERSION} >&log.txt
27
28 #Install Chaincode
29 peer lifecycle chaincode install ${CHAINCODE_NAME}.tar.gz && pckID.txt
30 pckID=$(grep -o 'identifier.*' pckID.txt|cut -f2,3 -d:) #grab string 'identifier' and forwards, after, the output is cut with delimit
er ':' in this case there are 3 parts, where parts 2 and 3 are chosen
31 rm pckID.txt
32 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org2.jnvr.com:7051 CORE_PEER LOCALMSPID="Org2MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt peer lifecycle chaincode ins
tall ${CHAINCODE_NAME}.tar.gz
33 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org3.jnvr.com:7051 CORE_PEER LOCALMSPID="Org3MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer lifecycle chaincode ins
tall ${CHAINCODE_NAME}.tar.gz
34 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER LOCALMSPID="Org4MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincode ins
tall ${CHAINCODE_NAME}.tar.gz
35
36 #Endorsing Policies
37 peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAI
NCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.peer')" --package-id $pckID
38 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org3.jnvr.com:7051 CORE_PEER LOCALMSPID="Org3MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt peer lifecycle chaincode app
roveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --wa
itForEvent --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.peer')" --package-id $pckID
39 CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/users/Admin@org4.j
nvr.com/msp/ CORE_PEER ADDRESS=peer0.org4.jnvr.com:7051 CORE_PEER LOCALMSPID="Org4MSP" CORE_PEER TLS ROOTCERT FILE=/opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt peer lifecycle chaincode app
roveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --wa
itForEvent --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.peer')" --package-id $pckID
40
41 #Check Endorsing Policies
42 peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequen
ce 1 --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.peer')" --output json
43
44 #Commit chaincode
45 peer lifecycle chaincode commit -o orderer.jnvr.com:7050 --tls --cafile $ORDERER_CA --peerAddresses peer0.org1.jnvr.com:7051 --tlsRo
otCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/peers/peer0.org1.jnvr.com/tls/ca.
crt --peerAddresses peer0.org3.jnvr.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizat
ions/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt --peerAddresses peer0.org4.jnvr.com:7051 --tlsRootCertFiles /opt/gopath/src/gith
ub.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com/peers/peer0.org4.jnvr.com/tls/ca.crt --channelID $CHANNEL_NAME --
name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --signature-policy "OR ('Org1MSP.peer','Org3MSP.peer','Org4MSP.peer')"
46
47 #Using chaincode functions
48 peer chaincode invoke -o orderer.jnvr.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","i
d:3","jorge","naranja"]}'

```

Figura 85 - Documento script_jnvr_network_docker.sh

Este script irá correr os comandos dentro de *cli*, presentes na implementação anterior (secção 5.2.3.3). Onde se cria o canal, adicionam-se as organizações ao canal, definem-se os *Anchor Peers* para cada organização (de modo a poder existir comunicação entre organizações), faz-se o *setup* do *chaincode* e coloca-se um valor no *ledger* utilizando a função *Set* presente no *chaincode*.

- 3) Após ter os ficheiros prontos basta executar *script_jnvr_network.sh* a partir da diretoria *jnvr_network* com o comando `./script_jnvr_network.sh`.

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ ./script_jnvr_network.sh
788b031780b0
96c918443162
5a36e6024f57
3e4807b0f839
3231735ac3b1
138419b9521b
58307a33f1c9
5d83fa5d25b0
2aba4cc48106
ac6e2261d430
6e5215708aa5
0e8626158efec8e60317d4d869c6647388460e25f48df99d0b3109fb48d0582d
02f37224e51c78c083d793f6df5b2fadddd8088dc3d0ff0fbee8ae3b56dbdb19
4abd8e540c5e8ce47bfaceba60cf89abaf9ad677db573590188b60e984ede86e
6bb185b4e9103efa99f7f9fd5e5e1fac03bc985cbe1553015d04fc0ed9212e51
8a61880fc2ef46a0afde8ad0794deb3b2f375fa5356ed1b52a0feef1d06f66de
8d0ad3fdb56a9f84676ea512f8e304060a4ef802f218587e5fdc35d9c5991b50
96c51b6fb2e85cb117c4072b138a6e727bbf2c9d13237b311b4efd10b0f6a005
780ca21f0579533f6f45e0a230a61afce7a96adaeb4b2f103aedae4db9a329d9
892ad43336be95ba0cdaa42813a0b7ef33a6883b48cae2f652dca3912acf577c
35740bcc4f73b47a2a50b65a36a03fe428c5clfbc7b34a82956cc717a8
424306f265530abf4d9f1c64ee2f9a39e0d4a739284da1ca0a4bdea8aa92b9d4
a042d893c348a33d714d072f8d5aae535757ea55778d60cd43ed887b85ec687e
aded19ca88ba9725f42326900646ba4cd4457173433370a67e499eedeed376b9
bb61cf618a2a6581c9d2351e91840db3b5a18a198e61fb02b31454be77bd6644
cee5921f6f000991b2d1b65cbeaf3e39e44a13747b8c6fe713b039cf58c2b1cd
jnvr-network_basic
org1.jnvr.com
org2.jnvr.com
org3.jnvr.com
org4.jnvr.com
2022-04-17 23:02:03.541 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:03.560 WEST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: solo
2022-04-17 23:02:03.560 WEST 0003 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:03.590 WEST 0004 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2022-04-17 23:02:03.590 WEST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Creating system channel genesis block
2022-04-17 23:02:03.590 WEST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
2022-04-17 23:02:03.633 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:03.646 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:03.646 WEST 0003 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Generating new channel configtx
2022-04-17 23:02:03.696 WEST 0004 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Writing new channel tx
2022-04-17 23:02:03.747 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:03.759 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:03.759 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-17 23:02:03.783 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
2022-04-17 23:02:03.837 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:03.850 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:03.850 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-17 23:02:03.879 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
2022-04-17 23:02:03.925 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:03.935 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:03.936 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-17 23:02:03.960 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
2022-04-17 23:02:04.003 WEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-04-17 23:02:04.016 WEST 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-04-17 23:02:04.016 WEST 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-04-17 23:02:04.039 WEST 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
[+] Running 12/12
# Network jnvr-network_basic Created 0.1s
# Container couchdb0 Started 1.8s
# Container couchdb3 Started 1.5s
# Container orderer.jnvr.com Started 1.9s
# Container couchdb1 Started 2.1s
# Container ca.org1.jnvr.com Started 2.1s
# Container couchdb2 Started 2.1s
# Container peer0.org2.jnvr.com Start... 4.1s
# Container peer0.org4.jnvr.com Start... 3.5s
# Container peer0.org3.jnvr.com Start... 4.1s
# Container peer0.org1.jnvr.com Start... 3.8s
# Container cli Started 5.2s
```

Figura 86 - Excertos da execução de *script_jnvr_network.sh*

- 4) Depois, verifica-se que o `script_jnvr_network_docker.sh` foi enviado para o container utilizando o comando `ls`.

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # ls
channel-artifacts      crypto                  script_jnvr_network_docker.sh
```

Figura 87 - Verificação da presença de `script_jnvr_network_docker.sh`

- 5) Finalmente utilizando o comando `./script_jnvr_network_docker.sh` será terminada a configuração da rede, obtendo-se o mesmo resultado da implementação anterior (secção 5.2.3.3).

Durante a criação dos scripts foram encontrados problemas de acesso quando se utilizava o comando

```
docker exec --user root cli
/opt/gopath/src/github.com/hyperledger/fabric/peer/script_jnvr_netwo
rk_docker.sh
```

Este comando iria permitir a execução automática de `script_jnvr_network_docker.sh` ao executar `script_jnvr_network.sh`. O facto deste comando não funcionar é a razão de ser necessário correr o `script` dentro do container `cli`.

Adicionalmente, é importante ter em conta que os `exports` estão a ser declarados dentro do `script`. Caso seja necessário de correr novos comandos que usem variáveis exportadas nos scripts, é necessário voltar a exportá-las.

5.2.3.5 Adicionar uma organização a uma rede em funcionamento

A tentativa de implementar a adição de uma organização a uma rede em funcionamento foi baseada no tutorial presente em [30] onde é utilizada a rede descrita na implementação anterior (secção 5.2.3.4), que contém quatro organizações de raiz, tendo-se o objetivo de adicionar uma quinta organização. O processo percorrido é descrito de seguida:

0) Antes de qualquer passo é necessário inicializar a rede, para isso utiliza-se os scripts descritos anteriormente (secção 5.2.3.4), ou colocar os comandos manualmente (secção 5.2.3.3).

- 1) Tendo a rede a correr cria-se a diretoria *AddOrg5* em *jnvr-network*. Depois começa-se por criar o ficheiro *org5-crypto-config.yaml*, que descreve a nova organização a ser criada, como se observa de seguida.

```
home > osboxes > Desktop > jnvr_net > jnvr-network > AddOrg5 > ! org5-crypto-config.yaml
1 | PeerOrgs:
2 |   - Name: Org5
3 |     Domain: org5.jnvr.com
4 |     EnableNodeOUs: true
5 |     Template:
6 |       Count: 1
7 |       SANS:
8 |         - localhost
9 |     Users:
10 |       Count: 1
11 |
```

Figura 88 - Ficheiro *org5-crypto-config.yaml*

- 2) Com o ficheiro obtido anteriormente utiliza-se o comando `cryptogen generate --config=./org5-crypto-config.yaml`, na diretoria acabada de criar (*AddOrg5*) para gerar os materiais chave da organização 5 na diretoria *crypto-config*.

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network$ cd AddOrg5/
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ cryptogen generate
--config=./org5-crypto-config.yaml
org5.jnvr.com
```

Figura 89 - Execução do comando `cryptogen generate` para a organização 5

- 3) Depois copia-se a diretoria *crypto-config* da rede inicializada anteriormente para o *crypto-config* da organização cinco, acabado de criar, com o comando `cp -r ../crypto-config/ordererOrganizations crypto-config/`.

- 4) De seguida utiliza-se o seguinte comando para criar o ficheiro *org5.json* na diretoria *channel-artifacts* da rede, onde se descreve a configuração da organização cinco em formato *json*.

```
export FABRIC_CFG_PATH=$PWD && configtxgen -printOrg Org5MSP >
../channel-artifacts/org5.json
```

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ cp -r ../crypto-co
nfig/ordererOrganizations crypto-config/.
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ export FABRIC_CFG_
PATH=$PWD && configtxgen -printOrg Org5MSP > ../channel-artifacts/org5.json
2022-05-27 14:43:17.756 WEST 0001 INFO [common.tools.configtxgen] main -> L
oading configuration
2022-05-27 14:43:17.762 WEST 0002 INFO [common.tools.configtxgen.localconfi
g] LoadTopLevel -> Loaded configuration: /home/osboxes/Desktop/jnvr_net/jnvr
-network/AddOrg5/configtx.yaml
```

Figura 90 - Criação de *org5.json*

- 5) Nesta altura, entra-se no container cli (que se encontra a correr na Org1), com `docker exec -it cli /bin/sh` e correm-se os seguintes comandos.

```
export
```

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscac
erts/tlsca.jnvr.com-cert.pem && export CHANNEL_NAME=channel1
```

```
peer channel fetch config config_block.pb -o orderer.jnvr.com:7050 -c
$CHANNEL_NAME --tls --cafile $ORDERER_CA
```

Este processo irá criar o ficheiro *config_block.pb* que contém o último bloco de configuração do *blockchain*.

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ docker exec -it cli /bin/sh
/opt/gopath/src/github.com/hyperledger/fabric/peer # export ORDERER_CA=/opt
/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/
jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem &
& export CHANNEL_NAME=channel1
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel fetch con
fig config_block.pb -o orderer.jnvr.com:7050 -c $CHANNEL_NAME --tls --cafil
e $ORDERER_CA
2022-05-27 14:01:08.015 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactori
es(). Falling back to bootBCCSP.
2022-05-27 14:01:08.122 UTC [grpc] Infof -> DEBU 029 Channel Connectivity change to READY
2022-05-27 14:01:08.124 UTC [channelCmd] InitCmdFactory -> INFO 02a Endorser and orderer connections i
nitialized
2022-05-27 14:01:08.126 UTC [msp.identity] Sign -> DEBU 02b Sign: plaintext: 0AE060A1408051A0608A4B3C
3940622...A6EB9AEF819E12080A020A0012020A00
2022-05-27 14:01:08.126 UTC [msp.identity] Sign -> DEBU 02c Sign: digest: 7B67578CBAE3C5AF560F4B31305F
DA999F80683601B9199EBD3804474F695EAB
2022-05-27 14:01:08.162 UTC [cli.common] readBlock -> INFO 02d Received block: 9
2022-05-27 14:01:08.162 UTC [channelCmd] fetch -> INFO 02e Retrieving last config block: 4
2022-05-27 14:01:08.163 UTC [msp.identity] Sign -> DEBU 02f Sign: plaintext: 0AE060A1408051A0608A4B3C
3940622...A91C120C0A041A02080412041A020804
2022-05-27 14:01:08.163 UTC [msp.identity] Sign -> DEBU 030 Sign: digest: 1C13C27C0FF5637C324792558BFF
9806C01E46A03FC7B8C2807D2E446FA9FA3C
2022-05-27 14:01:08.170 UTC [cli.common] readBlock -> INFO 031 Received block: 4
```

Figura 91 - Excertos da criação de *config_block.pb*

Como se observa na Figura 91 apresentada anteriormente, o último bloco de configuração é o número quatro. Isto deve-se a nesse momento existirem cinco blocos de configuração na rede, como se observa na seguinte figura (Figura 92).

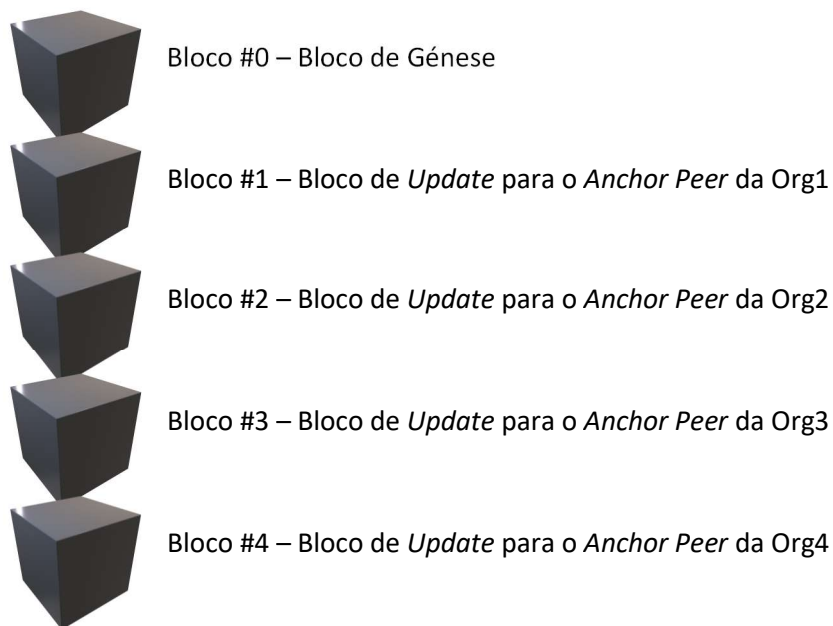


Figura 92 - Representação dos blocos presentes na rede

- 6) Seguidamente usa-se o seguinte comando, para extrair a porção de código relevante para o *update*, ou seja, a configuração de *ORG1MSP*, *ORG2MSP*, *ORG3MSP*, *ORG4MSP*, entre outros.

```
configtxlator proto_decode --input config_block.pb --type common.Block  
| jq .data.data[0].payload.data.config > config.json
```

- 7) Depois corre-se o comando apresentado para adicionar-se a definição de *ORG5MSP* (presente no *org5.json* criado anteriormente) antecedida do envelope definido no ficheiro *config.json*, criando o ficheiro *modified_config.json*.

```
jq -s '.[0] * {"channel_group":{"groups":{"Application":{"groups":  
{"Org5MSP":.[1]}}}}}' config.json ./channel-artifacts/org5.json >  
modified_config.json,
```

```

    },
    "type": 0
  },
  "version": "0"
}
},
"version": "1"
},
"Org5MSP": {
  "groups": {},
  "mod_policy": "Admins",
  "policies": {
    "Admins": {
      "mod_policy": "Admins",
      "policy": {
        "type": 1,
        "value": {
          "identities": [
            {
              "principal": {
                "msp_identifier": "Org5MSP",
                "role": "ADMIN"
              },
              "principal_classification": "ROLE"
            }
          ],
          "rule": {
            "n_out_of": {
              "n": 1,
              "rules": [
                {
                  "signed_by": 0
                }
              ]
            }
          }
        }
      },
      "version": 0
    }
  },
  "version": "0"
},
"Endorsement": {
  "mod_policy": "Admins",
  "policy": {
    "type": 1,
    "value": {
      "identities": [

```

Figura 93 - Excerto de *modified_config.json* relativo à organização 5

- 8) Seguidamente usam-se os seguintes comandos, de modo a transformar os ficheiros *config.json* e *modified_config.json* para ficheiros do tipo *.pb*.

```
configtxlator proto_encode --input config.json --type common.Config --output config.pb
```

```
configtxlator proto_encode --input modified_config.json --type common.Config --output modified_config.pb
```

- 9) Após ter os ficheiros da configuração inicial e da configuração com a organização cinco, usa-se o comando

```
configtxlator compute_update --channel_id $CHANNEL_NAME --original config.pb --updated modified_config.pb --output org5_update.pb,
```

Com o comando apresentado anteriormente, cria-se o ficheiro *org5_update.pb*, onde se computa a atualização da configuração, usando-se *modified_config.json*.

10) Usam-se os seguintes comandos para criar envelope a rodear *org5_update.pb*. Para isso transformou-se *org5_update.pb* em *JSON*, criou-se *org5_update_in_envelope.json* (já com o envelope) e transformou-se esse ficheiro em *.pb*.

```
configtxlator proto_decode --input org5_update.pb --type
common.ConfigUpdate | jq . > org5_update.json
```

```
echo '{"payload":{"header":{"channel_header":{"channel_id":"channel1",
"type":2}},"data":{"config_update":'$(cat org5_update.json)'}}}' | jq .
> org5_update_in_envelope.json
```

```
configtxlator proto_encode --input org5_update_in_envelope.json --type
common.Envelope --output org5_update_in_envelope.pb
```

11) Para terminar a ação da Org1, usa-se o comando `peer channel signconfigtx -f org5_update_in_envelope.pb`, que assina a modificação à rede, ou seja, declara que está de acordo com esta alteração.

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel signconfigtx -f org5_update_in_envelope.pb
2022-05-27 16:55:56.920 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling ba
ck to bootBCCSP.
2022-05-27 16:55:56.922 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories(). Falling ba
ck to bootBCCSP.
AiEA+6UxWokyURym+TpCftgggorZ2v3/MAtch3XLE9648usCIGQu06sNReBSrWUT
ZL2hoDbHYqjDUR/YJScsy8hx68ot
-----END CERTIFICATE-----
2022-05-27 16:55:56.948 UTC [msp] setupSigningIdentity -> DEBU 01a Signing identity expires at 2032-05-24 12:25:00 +00
00 UTC
2022-05-27 16:55:56.949 UTC [msp] GetDefaultSigningIdentity -> DEBU 01b Obtaining default signing identity
2022-05-27 16:55:56.949 UTC [channelCmd] InitCmdFactory -> INFO 01c Endorser and orderer connections initialized
2022-05-27 16:55:56.950 UTC [msp.identity] Sign -> DEBU 01d Sign: plaintext: 0AAA060A074F7267314D5350129E062D...726974
65727312002A0641646D696E73
2022-05-27 16:55:56.950 UTC [msp.identity] Sign -> DEBU 01e Sign: digest: 10B8064F12AC5E6D76BA7CB9430CB313997FB02EED34
9F1E368EC3FAF62C44B2
2022-05-27 16:55:56.951 UTC [msp.identity] Sign -> DEBU 01f Sign: plaintext: 0AE0060A1408021A06089C85C4940622...803896
BECAB4C14000191EAF74780711
2022-05-27 16:55:56.952 UTC [msp.identity] Sign -> DEBU 020 Sign: digest: CB89814C040BF52123ED608FC380C9909DD30E09C2EA
B0358CBE39BA0209A3CC
```

Figura 94 - Excertos da assinatura da modificação aplicada à rede

12) Devido à maneira de como foram definidas as políticas da rede, é necessário uma maioria das organizações autorizar uma alteração para esta entrar em vigor, como se observa na figura apresentada (Figura 95).

```

home > osboxes > Desktop > jnvr_net > jnvr-network > ! configtx.yaml
---
196 #####
197 #
198 # CHANNEL
199 #
200 # This section defines the values to encode into a config transaction or
201 # genesis block for channel related parameters.
202 #
203 #####
204 Channel: &ChannelDefaults
205 # Policies defines the set of policies at this level of the config tree
206 # For Channel policies, their canonical path is
207 # /Channel/<PolicyName>
208 Policies:
209 # Who may invoke the 'Deliver' API
210 Readers:
211   Type: ImplicitMeta
212   Rule: "ANY Readers"
213 # Who may invoke the 'Broadcast' API
214 Writers:
215   Type: ImplicitMeta
216   Rule: "ANY Writers"
217 # By default, who may modify elements at this config level
218 Admins:
219   Type: ImplicitMeta
220   Rule: "MAJORITY Admins"
221
222 # Capabilities describes the channel level capabilities, see the
223 # dedicated Capabilities section elsewhere in this file for a full
224 # description
225 Capabilities:
226   <<: *ChannelCapabilities
227

```

Figura 95 - Excerto da configuração da rede que dita ser necessário existir uma maioria para verificar alterações na rede

Como a rede a correr é constituída por quatro organizações é necessário a alteração ser assinada por pelo menos três organizações (para atingir a maioria). Para isso é necessário entrar em outras organizações e assinar a modificação, como se descreve de seguida:

- a) Usam-se os comandos, para entrar no *container cli* através da Org2 e assinar a modificação

```

docker exec -e CORE_PEER_LOCALMSPID="Org2MSP" -e
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt -e
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.jnvr.com/msp/ -e CORE_PEER_ADDRESS=peer0.org2.jnvr.com:7051 -it cli /bin/sh

```

```
peer channel signconfigtx -f org5_update_in_envelope.pb
```

```

osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ docker exec -e CORE_PEER_LOCALMSPID="Org2MSP" -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/peers/peer0.org2.jnvr.com/tls/ca.crt -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com/users/Admin@org2.jnvr.com/msp/ -e CORE_PEER_ADDRESS=peer0.org2.jnvr.com:7051 -it cli /bin/sh /opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel signconfigtx -f org5_update_in_envelope.pb
2022-05-27 17:09:14.474 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-05-27 17:09:14.477 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.

```

Figura 96 - Assinatura da modificação da rede por parte da organização 2

- b) Finalmente utilizam-se os comandos apresentados de seguida, para entrar no container *cli* através da *Org3* e fazer *update*, que automaticamente assina a modificação por parte da *Org3*.

```
docker exec -e CORE_PEER_LOCALMSPID="Org3MSP" -e
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt -e
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ -e CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051 -it cli
/bin/sh

export
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem && export CHANNEL_NAME=channel1

peer channel update -f org5_update_in_envelope.pb -c $CHANNEL_NAME -o orderer.jnvr.com:7050 --tls --cafile $ORDERER_CA
```

```
osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ docker exec -e CORE_PEER_LOCALMSPID="Org3MSP" -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/peers/peer0.org3.jnvr.com/tls/ca.crt -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com/users/Admin@org3.jnvr.com/msp/ -e CORE_PEER_ADDRESS=peer0.org3.jnvr.com:7051 -it cli /bin/sh /opt/gopath/src/github.com/hyperledger/fabric/peer # export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem && export CHANNEL_NAME=channel1 /opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel update -f org5_update_in_envelope.pb -c $CHANNEL_NAME -o orderer.jnvr.com:7050 --tls --cafile $ORDERER_CA
2022-05-27 17:12:59.689 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-05-27 17:12:59.721 UTC [grpc] Infof -> DEBU 026 Channel switches to new LB policy "pick_first"
2022-05-27 17:12:59.721 UTC [grpc] Infof -> DEBU 027 Subchannel Connectivity change to CONNECTING
2022-05-27 17:12:59.721 UTC [grpc] Infof -> DEBU 028 Subchannel picks a new address "orderer.jnvr.com:7050" to connect
2022-05-27 17:12:59.726 UTC [grpc] UpdateSubConnState -> DEBU 029 pickfirstBalancer: HandleSubConnStateChange: 0xc00002f2b0, {CONNECTING <nil>}
2022-05-27 17:12:59.726 UTC [grpc] Infof -> DEBU 02a Channel Connectivity change to CONNECTING
2022-05-27 17:12:59.734 UTC [comm.tls] ClientHandshake -> DEBU 02b Client TLS handshake completed in 7.254138ms remote address=172.21.0.7:7050
2022-05-27 17:12:59.736 UTC [grpc] Infof -> DEBU 02c Subchannel Connectivity change to READY
2022-05-27 17:12:59.737 UTC [grpc] UpdateSubConnState -> DEBU 02d pickfirstBalancer: HandleSubConnStateChange: 0xc00002f2b0, {READY <nil>}
2022-05-27 17:12:59.737 UTC [grpc] Infof -> DEBU 02e Channel Connectivity change to READY
2022-05-27 17:12:59.898 UTC [channelCmd] update -> INFO 02f Successfully submitted channel update
```

Figura 97 - Excertos da execução de *update* por parte da organização 3

- 13) Após ter a rede atualizada com a *Org5* usa-se o comando `docker-compose -f docker-compose-couch-org5.yaml up -d` para inicializar um *peer* da organização, uma base de dados de *couchdb* e uma linha de comandos denominada *Org5cli*.


```

1 version: '2'
2
3 # ---CHANGED--- our network is called "basic"
4 networks:
5   basic:
6 services:
7
8 # ---CHANGED--- The peer name is taken from the name generated by the "cryptogen" certs – it indicates the peer org 5 and one peer "peer0"
9 peer0.org5.jnvr.com:
10 # ---CHANGED--- Container name – same as the peer name
11   container_name: peer0.org5.jnvr.com
12   extends:
13     file: base/docker-compose-base.yaml
14     # ---CHANGED--- Refers to peer name
15     service: peer0.org5.jnvr.com
16   environment:
17     - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
18     - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb4:5984
19     - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
20     - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
21   depends_on:
22     - orderer.jnvr.com
23     - couchdb4
24   networks:
25     # ---CHANGED--- our network is called "basic"
26     - basic
27
28 couchdb4:
29   image: couchdb:3.1
30   environment:
31     - COUCHDB_USER=admin
32     - COUCHDB_PASSWORD=adminpw
33   ports:
34     - 5988:5984
35   container_name: couchdb4
36   networks:
37     - basic
38
39 Org5cli:
40   container_name: Org5cli
41   image: hyperledger/fabric-tools:2.2
42   tty: true
43   stdin_open: true
44   environment:
45     - GOPATH=/opt/gopath
46     - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
47     - FABRIC_LOGGING_SPEC=INFO
48     #- FABRIC_LOGGING_SPEC=DEBUG
49     - CORE_PEER_ID=Org5cli
50     # ---CHANGED--- peer0 from Org5 is the default for this CLI container
51     - CORE_PEER_ADDRESS=peer0.org5.jnvr.com:11051
52     - CORE_PEER_LOCALMSPID=Org5MSP
53     - CORE_PEER_TLS_ENABLED=true
54     # ---CHANGED--- changed to reflect peer0 name, org5 name and our company's domain
55     - CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org5.jnvr.com/peers/peer0.org5.jnvr.com/tls/server.crt
56     # ---CHANGED--- changed to reflect peer0 name, org5 name and our company's domain
57     - CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org5.jnvr.com/peers/peer0.org5.jnvr.com/tls/server.key
58     # ---CHANGED--- changed to reflect peer0 name, org5 name and our company's domain
59     - CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org5.jnvr.com/peers/peer0.org5.jnvr.com/tls/ca.crt
60     # ---CHANGED--- changed to reflect peer0 name, org5 name and our company's domain
61     - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org5.jnvr.com/users/Admin@org5.jnvr.com/msp
62   working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
63   # ---CHANGED--- command needs to be connected out as we will be issuing commands explicitly, no t using by any script
64   # command: /bin/bash -c './scripts/script.sh ${CHANNEL_NAME}; sleep $TIMEOUT'
65   command: /bin/bash
66   volumes:
67     - /var/run/./host/var/run/
68     # ---CHANGED--- chaincode path adjusted
69     - ./../chaincode:/opt/gopath/src/github.com/chaincode
70     - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
71     # ---CHANGED--- added paths for each orgs 1 2 3 4
72     - ./../crypto-config/peerOrganizations/org1.jnvr.com:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com
73     - ./../crypto-config/peerOrganizations/org2.jnvr.com:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.jnvr.com
74     - ./../crypto-config/peerOrganizations/org3.jnvr.com:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.jnvr.com
75     - ./../crypto-config/peerOrganizations/org4.jnvr.com:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.jnvr.com
76   depends_on:
77     # ---CHANGED--- reference to peer0 of Org5
78     - peer0.org5.jnvr.com
79
80   networks:
81     # ---CHANGED--- our network is called "basic"
82     - basic

```

Figura 98 - Definição das configurações do novo peer, novo couchdb e org5cli

```

osboxes@osboxes:~/Desktop/jnvr_net/jnvr-network/AddOrg5$ docker-compose -f docker-compose-couch-org5.yaml up -d
[+] Running 3/3
  # Container couchdb4           Started
  # Container peer0.org5.jnvr.com Started
  # Container Org5cli            Started

```

Figura 99 - Inicialização dos componentes da organização 5

14) Seguidamente usa-se `docker exec -it Org5cli /bin/sh` para entrar na linha de comandos acabada de criar e faz-se os seguintes *exports*.

```
export
```

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/jnvr.com/orderers/orderer.jnvr.com/msp/tlscacerts/tlsca.jnvr.com-cert.pem && export CHANNEL_NAME=channel1
```

15) Depois usa-se o comando `peer channel fetch 0 channel1.block -o orderer.jnvr.com:7050 -c $CHANNEL_NAME --tls --cafile $ORDERER_CA` para ir trazer o bloco de g nese do *blockchain*, ou seja, para ir buscar o bloco#0.

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel fetch 0 channel1.block -o orderer.jnvr.com:7050 -c $CHANNEL_NAME --tls --cafile $ORDERER_CA
2022-05-27 20:17:53.231 UTC [bccsp] GetDefault -> DEBU 001 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-05-27 20:17:53.234 UTC [bccsp] GetDefault -> DEBU 002 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-05-27 20:17:53.244 UTC [bccsp] GetDefault -> DEBU 003 Before using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
2022-05-27 20:17:53.249 UTC [bccsp_sw] openKeyStore -> DEBU 004 KeyStore opened at [/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/users/Admin@org1.jnvr.com/msp/keystore]...done
2022-05-27 20:17:53.249 UTC [msp] getPemMaterialFromDir -> DEBU 005 Reading directory /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.jnvr.com/users/Admin@org1.jnvr.com/msp/signcerts
2022-05-27 20:17:53.254 UTC [grpc] Infof -> DEBU 020 ClientConn switching balancer to "pick_first"
2022-05-27 20:17:53.254 UTC [grpc] Infof -> DEBU 021 Channel switches to new LB policy "pick_first"
2022-05-27 20:17:53.255 UTC [grpc] Infof -> DEBU 022 Subchannel Connectivity change to CONNECTING
2022-05-27 20:17:53.255 UTC [grpc] Infof -> DEBU 023 Subchannel picks a new address "orderer.jnvr.com:7050" to connect
2022-05-27 20:17:53.255 UTC [grpc] UpdateSubConnState -> DEBU 024 pickfirstBalancer: HandleSubConnStateChange: 0xc000020fe0, {CONNECTING <nil>}
2022-05-27 20:17:53.256 UTC [grpc] Infof -> DEBU 025 Channel Connectivity change to CONNECTING
2022-05-27 20:17:53.267 UTC [comm.tls] ClientHandshake -> DEBU 026 Client TLS handshake completed in 4.311385ms remoteaddress=172.21.0.7:7050
2022-05-27 20:17:53.269 UTC [grpc] Infof -> DEBU 027 Subchannel Connectivity change to READY
2022-05-27 20:17:53.269 UTC [grpc] UpdateSubConnState -> DEBU 028 pickfirstBalancer: HandleSubConnStateChange: 0xc000020fe0, {READY <nil>}
2022-05-27 20:17:53.269 UTC [grpc] Infof -> DEBU 029 Channel Connectivity change to READY
2022-05-27 20:17:53.270 UTC [channelCmd] InitCmdFactory -> INFO 02a Endorser and orderer connections initialized
2022-05-27 20:17:53.271 UTC [msp.identity] Sign -> DEBU 02b Sign: plaintext: 0AE0060A1408051A0608F1E3C4940622...E63F6D49802912080A021A0012021A00
2022-05-27 20:17:53.272 UTC [msp.identity] Sign -> DEBU 02c Sign: digest: C05B2D0A48600E0643164983BAE10B94EB32AA0EA8F29F9B02C8E4653C44E0CE
2022-05-27 20:17:53.276 UTC [cli.common] readBlock -> INFO 02d Received block: 0
```

Figura 100 - Excertos do uso de `peer channel fetch 0`

16) Passando para o *localhost* correm-se os seguintes comandos para copiar `channel1.block` de *cli* para *Org5cli*.

```
docker cp
```

```
cli:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel1.block
./
```

```
docker cp channel1.block
```

```
Org5cli:/opt/gopath/src/github.com/hyperledger/fabric/peer/
```

- 17) No passo seguinte, deparou-se com um problema que não se conseguiu resolver devido ao *host peer0.prg5.jnvr.com* não reconhecido, ao correr o comando `peer channel join -b channel1.block`.

```
/opt/gopath/src/github.com/hyperledger/fabric/peer # peer channel join -b channel1.block
Error error getting endorser client for channel: endorser client failed to connect to peer0.org5.jnvr.com:11051: failed to create new connection: connection error: desc = "transport: error while dialing: dial tcp: lookup peer0.org5.jnvr.com: no such host"
```

Figura 101 - Erro verificado ao tentar juntar o novo peer da organização 5 ao canal

Concluindo esta implementação, apesar de não ter sido possível completar a implementação e execução, é importante destacar que a operação de adição de uma organização a uma rede em funcionamento operação é algo que deve ser evitado. Isto deve-se à sua complexidade, implicando bastante *tinkering*. Esta sofisticação deve-se à natureza *permissioned* das redes em *hyperledger-fabric*, onde a entrada a um *channel* da rede é restringida.

5.2.3.6 Implementação de uma rede a correr em várias máquinas

Nesta implementação pretendeu-se criar uma rede com duas organizações, cada uma com dois *peers*, correndo cada *peer* numa máquina diferente. A implementação foi baseada no tutorial presente em [31]. Os passos utilizados para implementar uma rede com esta configuração são descritos de seguida:

- 1) Para criar a configuração utilizou-se o comando `multipass launch 18.04 --name vm<#> -d 7G`, quatro vezes, de modo a criar VMs de *Ubuntu 18.04* onde apenas se especifica o espaço de disco (sete *Gigabytes*) utilizado de modo a ter espaço suficiente para os documentos que irão ser utilizados no exemplo, sendo os restantes parâmetros colocados nos seus valores predefinidos, para os quatro *peers* presentes na rede.
- 2) De seguida criou-se uma diretoria local denominada *MultipassFolder*. Nesta diretoria estão os seguintes ficheiros, que podem ser obtidos com o comando `git clone https://github.com/jnvr/MultipassFolder.git` (não esquecer de dar permissões aos ficheiros a utilizar posteriormente). Neste caso apenas alguns irão ser utilizados entre os quais:

```
PS C:\Users\2015232235\Documents\GitHub\MultipassFolder> ls

Directory: C:\Users\2015232235\Documents\GitHub\MultipassFolder

Mode                LastWriteTime         Length Name
----                -
d-----           25/05/2022    21:32         4host-swarm-clis
d-----           22/06/2022    19:36         config_maker_script
d-----           18/05/2022    08:59         jnvr_net
-a----           18/05/2022    08:53             66         .gitattributes
-a----           04/06/2022    16:41          3554         beginMultiHost.sh
-a----           22/06/2022    22:06          3013         initSetup.sh
-a----           18/05/2022    09:36           243         README.md
```

Figura 102 - Ficheiros presentes na diretoria MultipassFolder

- initSetup.sh: Usado na instalação de todos os pré-requisitos da implementação.

```
1 #!/bin/bash
2
3 echo " ===== "
4 echo "---Update and Upgrade---"
5 sudo apt update -y && sudo apt upgrade -y
6 echo " ===== "
7 echo "---Install Git---"
8 sudo apt-get install git
9 echo " ===== "
10 echo " ===== " >>versions.txt
11 echo "---Install Curl---"
12 sudo apt install curl
13 curl --version >versions.txt
14 echo " ===== " >>versions.txt
15 echo " ===== "
16 echo "---Installing Docker---"
17 sudo apt-get remove docker docker-engine docker.io
18 sudo apt install docker.io -y
19 sudo systemctl start docker
20 sudo systemctl enable docker
21 USER_NAME=$(whoami)
22 sudo usermod -a -G docker $USER_NAME
23 docker --version >>versions.txt
24 echo " ===== " >>versions.txt
25 echo " ===== "
26 echo "---Installing Docker-Compose---"
27 sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)
28 -$(uname -m)" -o /usr/local/bin/docker-compose
29 sudo chmod +x /usr/local/bin/docker-compose
30 sudo docker-compose --version >>versions.txt
31 echo " ===== " >>versions.txt
32 echo " ===== "
33 echo "---Install Fabric-Samples---"
34 sudo curl -sSL https://bit.ly/2ysb0FE | sudo bash -s
35 export PATH=./fabric-samples/bin:$PATH
36 echo " ===== "
37 echo "---Install NodeJS---"
38 wget http://nodejs.org/dist/v16.14.0/node-v16.14.0-linux-x64.tar.gz
39 sudo tar -C /usr/local --strip-components 1 -xzf node-v16.14.0-linux-x64.tar.gz
40 npm --version >>versions.txt
41 echo " ===== " >>versions.txt
42 echo " ===== "
43 echo "---Install Go---"
44 wget https://go.dev/dl/go1.18.1.linux-amd64.tar.gz
45 sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.18.1.linux-amd64.tar.gz
46 export PATH=$PATH:/usr/local/go/bin
47 go version >>versions.txt
48 echo " ===== "
49 echo "---Install Subversion---"
50 sudo apt-get install subversion -y
51 echo " ===== "
52 echo "---Install yq---"
53 sudo snap install yq
54 rm node-v16.14.0-linux-x64.tar.gz
55 rm go1.18.1.linux-amd64.tar.gz
56
57 export PATH=/home/ubuntu/fabric-samples/bin:$PATH
```

Figura 103 - Ficheiro initSetup.sh

- Diretoria 4host-swarm-clis: Onde estão os ficheiros utilizados para criar a rede em várias máquinas, que irão ser descritos futuramente. Caso não se pretenda fazer download da totalidade do repositório pode ser utilizado o comando `sudo svn checkout https://github.com/jnvr/MultipassFolder/trunk/4host-swarm-clis`. Seguido de `sudo chmod -R 755 4host-swarm-clis`, para dar permissões aos ficheiros.

3) Após a criação da diretoria *MultipassFolder* utilizou-se o seguinte comando para cada VM.

```
multipass mount <MultipassFolder directory location>\MultipassFolder\
vm<#>:/home/ubuntu/sharedFolder
```

No caso da implementação usa-se o seguinte comando, para criar uma diretoria partilhada entre a VM criada (localizada em `~/sharedFolder`) e a máquina local (localizada em `C:\Users\2015232235\Documents\Github\MultipassFolder\`).

```
multipass mount C:\Users\2015232235\Documents\Github\MultipassFolder\
vm<#>:/home/ubuntu/sharedFolder
```

Verifica-se que as VM estão a correr, a versão de ubuntu a ser usada e os seus *IPs* com o comando `multipass ls`.

```
Administrator: Windows PowerShell
PS C:\Windows\system32> multipass launch 18.04 --name vm1 -d 7G
Launched: vm1
PS C:\Windows\system32> multipass launch 18.04 --name vm2 -d 7G
Launched: vm2
PS C:\Windows\system32> multipass launch 18.04 --name vm3 -d 7G
Launched: vm3
PS C:\Windows\system32> multipass launch 18.04 --name vm4 -d 7G
Launched: vm4
PS C:\Windows\system32> multipass mount C:\Users\2015232235\Documents\Github\MultipassFolder\ vm1:/home/ubuntu/sharedFolder
PS C:\Windows\system32> multipass mount C:\Users\2015232235\Documents\Github\MultipassFolder\ vm2:/home/ubuntu/sharedFolder
PS C:\Windows\system32> multipass mount C:\Users\2015232235\Documents\Github\MultipassFolder\ vm3:/home/ubuntu/sharedFolder
PS C:\Windows\system32> multipass mount C:\Users\2015232235\Documents\Github\MultipassFolder\ vm4:/home/ubuntu/sharedFolder
PS C:\Windows\system32> multipass ls
Name      State      IPv4      Image
vm1       Running   172.31.118.1  Ubuntu 18.04 LTS
vm2       Running   172.31.124.170  Ubuntu 18.04 LTS
vm3       Running   172.31.123.54   Ubuntu 18.04 LTS
vm4       Running   172.31.118.243  Ubuntu 18.04 LTS
PS C:\Windows\system32>
```

Figura 104 - Inicialização das VM, criação de diretorias partilhadas e listagem de máquinas existentes

Após a criação das VM e das suas ligações à máquina local é fulcral ter bastante cuidado na forma como as VM são utilizadas. Durante as diversas tentativas de uso, estas ficavam bloqueadas ao inicializar, ou os comandos ficavam bloqueados. Nestes casos inicialmente deve-se abrir e fechar o *shell*. Caso volte a bloquear o comando, deverá parar-se a VM, da maneira apresentada na Figura 105.

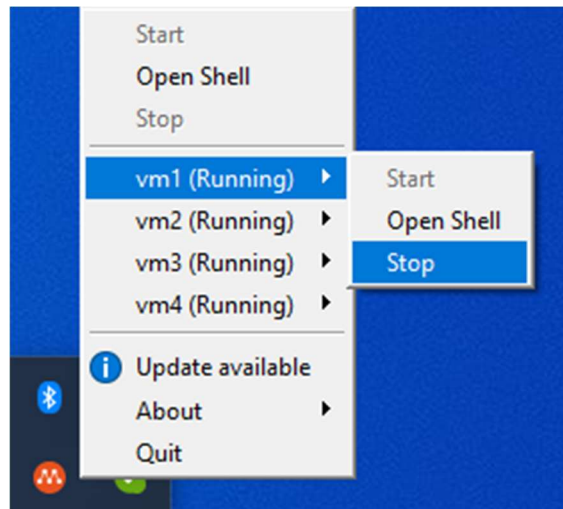


Figura 105 - Demonstração da paragem de uma máquina multipass

Adicionalmente, é de extrema importância não fechar o Shell enquanto este está a inicializar (como se observa na imagem). Isso poderá levar a que a VM fique corrompida não voltando a inicializar mais.

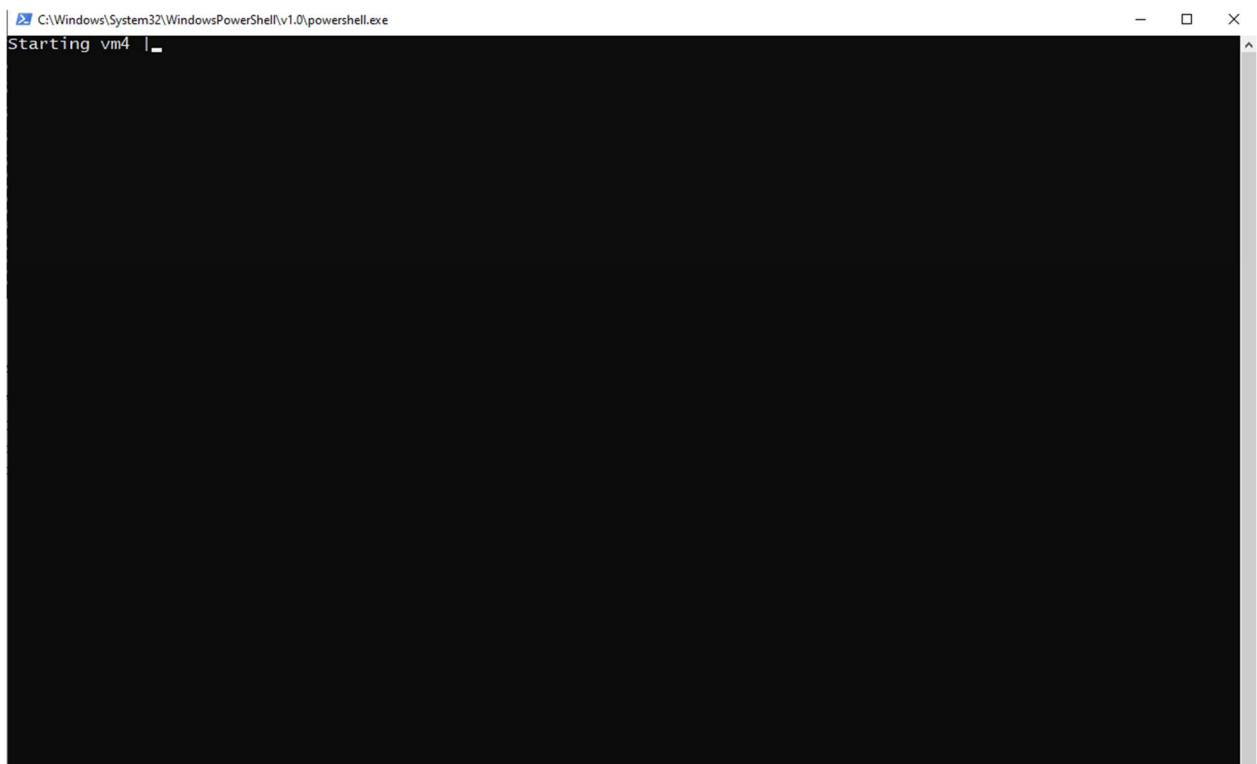


Figura 106 - Inicialização de uma máquina multipass

- 4) Após a criação das VMs e tendo em conta os cuidados ter, vai-se correr o *setup* inicial em cada VM. Para isso utiliza-se o seguinte comando, de modo a copiar o script de *setup* inicial, uma vez que é impossível mudar permissões e executar comandos na diretoria partilhada em *Multipass*.

```
cp ./sharedFolder/initSetup.sh ./
```


De seguida mudam-se as permissões do ficheiro com `chmod 755 initSetup.sh` e corre-se o script com `sudo ./initSetup.sh`.

Caso no decorrer da execução do script apareçam erros semelhantes ao apresentado de seguida, deve-se voltar a correr o script.

```
E: Could not get lock /var/lib/dpkg/lock-frontent - open (11: Resource temporarily unavailable)
```

```
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), is another process using it?
```

Depois de executar `initSetup.sh` utiliza-se o comando `go version`. Caso o comando não seja reconhecido, corre-se novamente as linhas do script `initSetup.sh` relativas ao "--- Install Go---" (ou seja, linhas 43-45).

A linguagem de programação `Go` estará bem instalada quando se utilizar `go version` e se obtiver o output: `go version go1.18.1 linux/amd64`

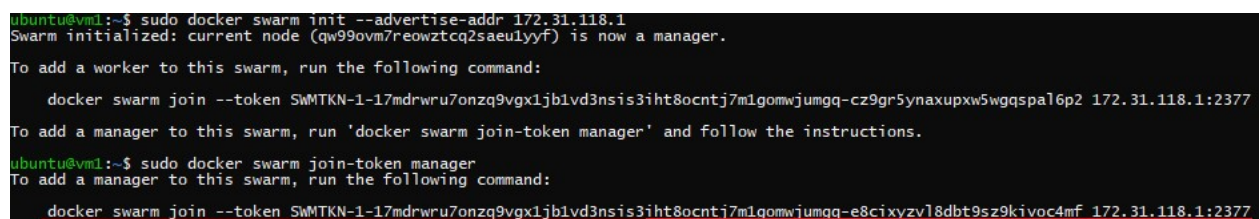
- 5) Após correr a configuração inicial em cada VM passa-se para a criação de um *overlay* de *Docker Swarm*, que permite a criação de uma rede distribuída em diversos containers de *Docker*, de modo a ser possível comunicar entre os *hosts*. Devido às redes de *overlay* encriptadas não funcionarem em *Windows*, para se ter uma solução mais flexível não se utilizou encriptação. Tendo isso em conta utilizou-se o comando `docker swarm init --advertise-addr <vm1 IP address>`.

Pode-se verificar o IP das VMs através do uso do comando `ifconfig` (dentro da VM) ou do uso de `multipass ls` (fora da VM, ou seja, na máquina local). Nesta implementação usa-se o comando.

```
sudo docker swarm init --advertise-addr 172.31.118.1
```

De modo a iniciar um *Docker Swarm* com o *host* da primeira VM. Seguido por `sudo docker swarm join-token manager` para obter o *token* para outros *hosts* entrarem na rede como *managers* (permite que cada nodo tenha permissões para gerir a rede). Neste exemplo obtém-se o seguinte comando.

```
docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzvl8dbt9sz9kivoc4mf 172.31.118.1:2377
```



```
ubuntu@vm1:~$ sudo docker swarm init --advertise-addr 172.31.118.1
Swarm initialized: current node (qw99ovm7reowztcq2saeu1yyf) is now a manager.
To add a worker to this swarm, run the following command:
    docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-cz9gr5ynaxupxw5wgqspa16p2 172.31.118.1:2377
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
ubuntu@vm1:~$ sudo docker swarm join-token manager
To add a manager to this swarm, run the following command:
    docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzvl8dbt9sz9kivoc4mf 172.31.118.1:2377
```

Figura 107 - Obtenção do comando para adicionar um manager à rede

Seguidamente utiliza-se o comando `sudo <output from join-token manager> --advertise-addr <VM# IP address>` para as outras três VMs. Especificamente, nesta circunstância, usa-se o seguinte comando para vm2.

```
sudo docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzvl8dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.124.170
```

O seguinte comando para vm3.

```
sudo docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzvl8dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.123.54
```

E o seguinte comando para vm4.

```
sudo docker swarm join --token SWMTKN-1-17mdrwru7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzvl8dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.118.243
```

```

ubuntu@vm2:~
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri May 20 11:17:32 WEST 2022
System load:  0.0          Processes:    107
Usage of /:   54.9% of 6.61GB  Users logged in:  0
Memory usage: 40%          IP address for eth0:  172.31.124.170
Swap usage:  0%            IP address for docker0: 172.17.0.1

0 updates can be applied immediately.
New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Fri May 20 10:27:28 2022 from 172.31.112.1
ubuntu@vm2:~$ sudo docker swarm join --token SWMTKN-1-17mdrwrw7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzv18dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.124.170
This node joined a swarm as a manager.
ubuntu@vm2:~$

ubuntu@vm3:~
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri May 20 11:20:26 WEST 2022
System load:  0.0          Processes:    107
Usage of /:   54.9% of 6.61GB  Users logged in:  0
Memory usage: 41%          IP address for eth0:  172.31.123.54
Swap usage:  0%            IP address for docker0: 172.17.0.1

0 updates can be applied immediately.
New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Fri May 20 10:39:13 2022 from 172.31.112.1
ubuntu@vm3:~$ sudo docker swarm join --token SWMTKN-1-17mdrwrw7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzv18dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.123.54
This node joined a swarm as a manager.
ubuntu@vm3:~$

ubuntu@vm4:~
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri May 20 11:21:03 WEST 2022
System load:  0.0          Processes:    107
Usage of /:   54.9% of 6.61GB  Users logged in:  0
Memory usage: 41%          IP address for eth0:  172.31.118.243
Swap usage:  0%            IP address for docker0: 172.17.0.1

0 updates can be applied immediately.
New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Fri May 20 10:49:38 2022 from 172.31.112.1
ubuntu@vm4:~$ sudo docker swarm join --token SWMTKN-1-17mdrwrw7onzq9vgx1jb1vd3nsis3iht8ocntj7m1gomwjumgq-e8cixyzv18dbt9sz9kivoc4mf 172.31.118.1:2377 --advertise-addr 172.31.118.243
This node joined a swarm as a manager.
ubuntu@vm4:~$

```

Figura 108 - Junção das VM 2,3,4 ao swarm criado pela máquina 1

Observa-se as máquinas ligadas ao nodo utilizando o comando `sudo docker node ls`. Este comando pode ser utilizado a partir de qualquer *host*. Para correr este comando é necessário uma maioria dos *hosts* do tipo *manager* estarem acessíveis.

```

ubuntu@vm1:~$ sudo docker node ls
ID                HOSTNAME          STATUS      AVAILABILITY      MANAGER STATUS      ENGINE VERSION
qw99ovm7reowztcq2saeulyyf *  vm1              Ready        Active              Leader                20.10.7
xwtddlriywnkww1do0hvi66g0    vm2              Ready        Active              Reachable             20.10.7
bmy0xckw4a8201l63qpa1k8yg    vm3              Ready        Active              Reachable             20.10.7
wbxixbynv67npziaof005ufv4    vm4              Ready        Active              Reachable             20.10.7
ubuntu@vm1:~$

```

Figura 109 - Listagem de máquinas ligadas a nodo

Caso seja necessário modificar o leader utilizar `sudo docker node demote <VMs que não se quer como leader>` a partir da VM que se quer como leader. Seguido de `sudo docker node promote <as mesmas VMs introduzidas anteriormente>`.

Finalmente passa-se para a criação da rede de *overlay* em si, usando `sudo docker network create --attachable --driver overlay first-network`, no primeiro *host*.

Com o comando `sudo docker network ls`, observa-se que foi criada a rede de *overlay* denominada *first-network* em cada *host*.

```
ubuntu@vm1: ~
ubuntu@vm1:~$ sudo docker network create --attachable --driver overlay first-network
oieoj9oq21gz5zlp4zeckpj6p
ubuntu@vm1:~$ sudo docker network ls
NETWORK ID      NAME                DRIVER              SCOPE
046366d03c52    bridge              bridge              local
f81ab9122877    docker_gwbridge     bridge              local
oieoj9oq21gz    first-network       overlay             swarm
e43fe7ba2c5f    host                host                local
ujomdh4u5x2k    ingress             overlay             swarm
69395c63a7fa    none                null                local
ubuntu@vm1:~$

ubuntu@vm2: ~
ubuntu@vm2:~$ sudo docker network ls
NETWORK ID      NAME                DRIVER              SCOPE
cf145f96480d    bridge              bridge              local
5c3e9cce7fb7    docker_gwbridge     bridge              local
oieoj9oq21gz    first-network       overlay             swarm
44365bb3e99f    host                host                local
ujomdh4u5x2k    ingress             overlay             swarm
2a3e1582d244    none                null                local
ubuntu@vm2:~$

ubuntu@vm3: ~
ubuntu@vm3:~$ sudo docker network ls
NETWORK ID      NAME                DRIVER              SCOPE
5ab0d4a3aa03    bridge              bridge              local
f99470649ca4    docker_gwbridge     bridge              local
oieoj9oq21gz    first-network       overlay             swarm
a47c3f620d4d    host                host                local
ujomdh4u5x2k    ingress             overlay             swarm
d85b691b2a43    none                null                local
ubuntu@vm3:~$

ubuntu@vm4: ~
ubuntu@vm4:~$ sudo docker network ls
NETWORK ID      NAME                DRIVER              SCOPE
ce095ce49c9e    bridge              bridge              local
a768997d6db8    docker_gwbridge     bridge              local
oieoj9oq21gz    first-network       overlay             swarm
460b33562db5    host                host                local
ujomdh4u5x2k    ingress             overlay             swarm
7322cdd494a4    none                null                local
ubuntu@vm4:~$
```

Figura 110 - Listagem de redes presentes a partir de cada máquina

- 6) Seguidamente, utiliza-se `cd fabric-samples/4host-swarm-clis/` para entrar na diretoria onde se encontram ficheiros de configuração da rede.

```

ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~$ cd fabric-samples/4host-swarm-clis/
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ ls -la
total 100
drwxr-xr-x  6 root root  4096 May 22 19:30 .
drwxr-xr-x 32 root root  4096 May 22 19:29 ..
-rwxr-xr-x  1 root root    17 May 22 19:29 .env
drwxr-xr-x  4 root root  4096 May 22 19:29 .svn
drwxr-xr-x  2 root root  4096 May 22 19:29 base
drwxr-xr-x  3 root root  4096 May 22 19:30 channel-artifacts
-rwxr-xr-x  1 root root 16021 May 22 19:29 configtx.yaml
drwxr-xr-x  4 root root  4096 May 22 19:30 crypto-config
-rwxr-xr-x  1 root root  4015 May 22 19:29 crypto-config.yaml
-rwxr-xr-x  1 root root  2994 May 22 19:29 host1.yaml
-rwxr-xr-x  1 root root   101 May 22 19:29 host1down.sh
-rwxr-xr-x  1 root root    35 May 22 19:29 host1up.sh
-rwxr-xr-x  1 root root  3060 May 22 19:29 host2.yaml
-rwxr-xr-x  1 root root   101 May 22 19:29 host2down.sh
-rwxr-xr-x  1 root root    35 May 22 19:29 host2up.sh
-rwxr-xr-x  1 root root  3063 May 22 19:29 host3.yaml
-rwxr-xr-x  1 root root   101 May 22 19:29 host3down.sh
-rwxr-xr-x  1 root root    35 May 22 19:29 host3up.sh
-rwxr-xr-x  1 root root  3071 May 22 19:29 host4.yaml
-rwxr-xr-x  1 root root   101 May 22 19:29 host4down.sh
-rwxr-xr-x  1 root root    35 May 22 19:29 host4up.sh
-rwxr-xr-x  1 root root  2518 May 22 19:29 mychannelup.sh

```

Figura 111 - Listagem dos ficheiros presentes na directoria 4host-swarm-clis

Os ficheiros denominados *host<#>up.yaml* iniciam os *containers* de *Docker* para cada *host* (ou seja, cada VM). Cada um destes documentos utiliza o *host<#>.yaml* correspondente, definindo a configuração de cada *host*, que se encontra caracterizado na Figura 112.

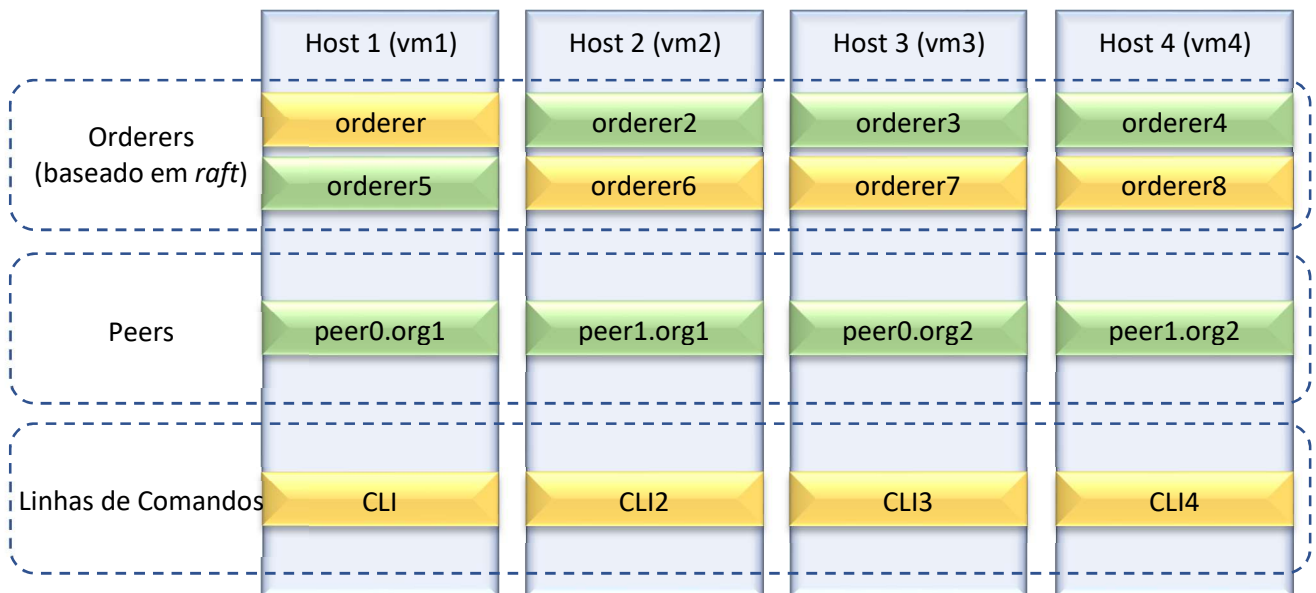


Figura 112 - Caracterização da configuração da rede

Cada VM tem então uma linha de comandos, um *peer* e dois *orderers* (um para o *peer* e outro para a linha de comandos) baseados em *raft* que significa que qualquer *orderer* pode ser acessado por qualquer *host*.

Caso seja necessário modificar a configuração, basta alterar os ficheiros *configtx.yaml*, *crypto-config.yaml*, *base/docker-compose-base.yaml* e *base/peer-base.yaml*, podendo modificar as organizações, número de *peers*, linhas de comandos, organizações, entre outros.

Após ter as novas configurações utilizam-se os comandos:

-----Geração material criptográfico-----

```
cryptogen generate --config=./crypto-config.yaml
export FABRIC_CFG_PATH=$PWD
```

-----Criação do bloco de génese-----

```
configtxgen -profile SampleMultiNodeEtcdRaft -channelID system-
channel -outputBlock ./channel-artifacts/genesis.block

configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-
artifacts/channel.tx -channelID mychannel

configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org1MSPanchors.tx -channelID mychannel -asOrg
Org1MSP

configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org2MSPanchors.tx -channelID mychannel -asOrg
Org2MSP
```

É de salientar que os comandos a serem corridos podem ser diferentes consoante a configuração. Pode-se desejar mais que dois *anchor peers* e ter o nome do *profiles* (presentes em *configtx.yaml*) diferentes.

- 7) Empregando-se o comando `./host<#>up.sh` para cada respetiva VM, são inicializados os *containers* de *Docker* em cada *host*.


```

ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo ./host1up.sh
WARNING: The SYS_CHANNEL variable is not set. Defaulting to a blank string.
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use 'docker stack deploy'.

Creating peer0.org1.example.com ... done
Creating orderer.example.com ... done
Creating orderer5.example.com ... done
Creating cli ... done
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$

ubuntu@vm2: ~/fabric-samples/4host-swarm-clis
ubuntu@vm2:~/fabric-samples/4host-swarm-clis$ sudo ./host2up.sh
WARNING: The SYS_CHANNEL variable is not set. Defaulting to a blank string.
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use 'docker stack deploy'.

Creating orderer6.example.com ... done
Creating orderer2.example.com ... done
Creating peer1.org1.example.com ... done
Creating cli2 ... done
ubuntu@vm2:~/fabric-samples/4host-swarm-clis$

ubuntu@vm3: ~/fabric-samples/4host-swarm-clis
ubuntu@vm3:~/fabric-samples/4host-swarm-clis$ sudo ./host3up.sh
WARNING: The SYS_CHANNEL variable is not set. Defaulting to a blank string.
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use 'docker stack deploy'.

Creating peer0.org2.example.com ... done
Creating orderer3.example.com ... done
Creating orderer7.example.com ... done
Creating cli3 ... done
ubuntu@vm3:~/fabric-samples/4host-swarm-clis$

ubuntu@vm4: ~/fabric-samples/4host-swarm-clis
ubuntu@vm4:~/fabric-samples/4host-swarm-clis$ sudo ./host4up.sh
WARNING: The SYS_CHANNEL variable is not set. Defaulting to a blank string.
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use 'docker stack deploy'.

Creating peer1.org2.example.com ... done
Creating orderer4.example.com ... done
Creating orderer8.example.com ... done
Creating cli4 ... done
ubuntu@vm4:~/fabric-samples/4host-swarm-clis$

```

Figura 113 - Inicialização dos componentes em cada peer

Caso dê erro devido a um dos clis não conseguir ligar-se à rede com ID igual ao da *first-network* criada anteriormente, deverá parar-se e reinicializar a VM e tentar correr o script novamente.

8) Seguidamente usa-se o script `mychannelup.sh` para:

- Criar o canal denominado *mychannel*, fazendo uso do comando:

```

docker exec cli peer channel create -o orderer.example.com:7050 -c
mychannel -f ./channel-artifacts/channel.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem

```

- Juntar as organizações ao canal criado, que é realizado através dos seguintes comandos:

-----peer0.org1-----

```
docker exec cli peer channel join -b mychannel.block
```

-----peer1.org1-----

```
docker exec -e CORE_PEER_ADDRESS=peer1.org1.example.com:8051 -e  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f  
abric/peer/crypto/peerOrganizations/org1.example.com/peers/peer1.org  
1.example.com/tls/ca.crt cli peer channel join -b mychannel.block
```

-----peer0.org2-----

```
docker exec -e  
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri  
c/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.ex  
ample.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e  
CORE_PEER_LOCALMSPID="Org2MSP" -e  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f  
abric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org  
2.example.com/tls/ca.crt cli peer channel join -b mychannel.block
```

-----peer1.org2-----

```
docker exec -e  
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri  
c/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.ex  
ample.com/msp -e CORE_PEER_ADDRESS=peer1.org2.example.com:10051 -e  
CORE_PEER_LOCALMSPID="Org2MSP" -e  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f  
abric/peer/crypto/peerOrganizations/org2.example.com/peers/peer1.org  
2.example.com/tls/ca.crt cli peer channel join -b mychannel.block
```

- Definir os Anchor Peers, um para cada organização, com os comandos apresentados de seguida:

```
docker exec cli peer channel update -o orderer.example.com:7050 -c  
mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg  
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t  
lsca.example.com-cert.pem
```

```
docker exec -e  
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri  
c/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.ex  
ample.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e  
CORE_PEER_LOCALMSPID="Org2MSP" -e  
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f  
abric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org  
2.example.com/tls/ca.crt cli peer channel update -o
```

```
orderer.example.com:7050 -c mychannel -f ./channel-  
artifacts/Org2MSPanchors.tx --tls --cafile  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg  
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t  
lsca.example.com-cert.pem
```

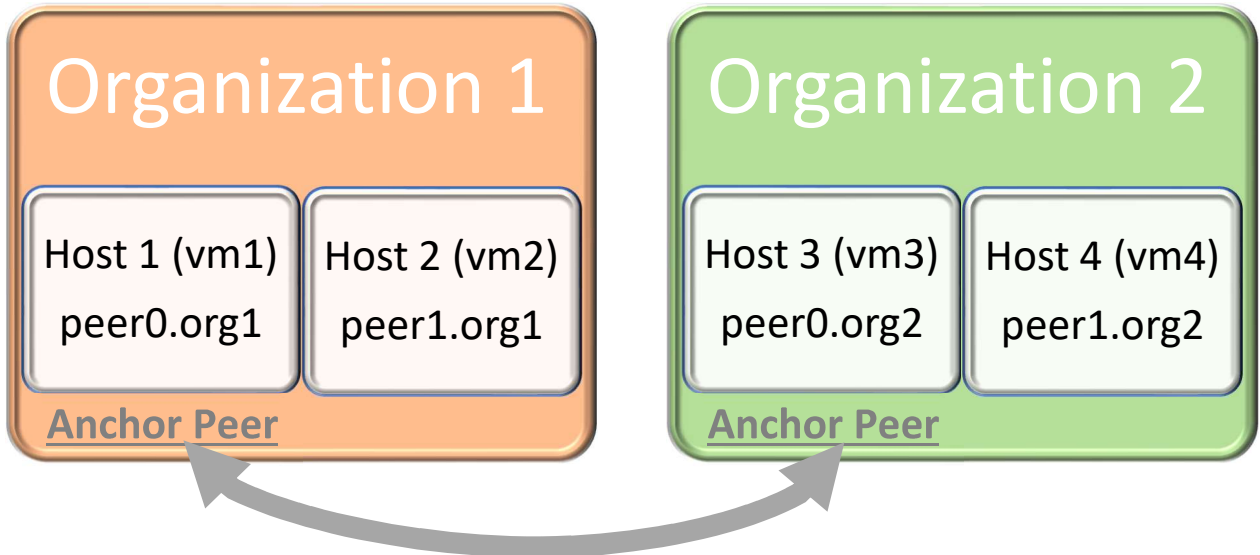


Figura 114 - Representação do funcionamento da rede

Utilizando então o comando `sudo ./mychannelup.sh` corre-se o *script* obtendo-se o seguinte resultado.

```

ubuntu@vml:~/fabric-samples/4host-swarm-clis$ sudo ./mychannelup.sh
2022-05-24 19:57:12.897 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:12.920 UTC 0002 INFO [cli.common] readBlock -> Expect block, but got status: &{NOT_FOUND}
2022-05-24 19:57:12.925 UTC 0003 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:13.128 UTC 0004 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:13.133 UTC 0005 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:13.336 UTC 0006 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:13.342 UTC 0007 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:13.545 UTC 0008 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:13.549 UTC 0009 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:13.752 UTC 000a INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:13.759 UTC 000b INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:13.961 UTC 000c INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:13.969 UTC 000d INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:14.172 UTC 000e INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:14.178 UTC 000f INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:14.382 UTC 0010 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:14.389 UTC 0011 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:14.593 UTC 0012 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:14.601 UTC 0013 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:14.804 UTC 0014 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:14.808 UTC 0015 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:15.012 UTC 0016 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:15.019 UTC 0017 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:15.230 UTC 0018 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:15.241 UTC 0019 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:15.443 UTC 001a INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:15.450 UTC 001b INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:15.653 UTC 001c INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:15.660 UTC 001d INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:15.863 UTC 001e INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:15.869 UTC 001f INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:16.073 UTC 0020 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:16.082 UTC 0021 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:16.284 UTC 0022 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:16.291 UTC 0023 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:16.494 UTC 0024 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:16.501 UTC 0025 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:16.704 UTC 0026 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:16.710 UTC 0027 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:16.913 UTC 0028 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:16.919 UTC 0029 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:17.122 UTC 002a INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:17.128 UTC 002b INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:17.331 UTC 002c INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:17.337 UTC 002d INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:17.540 UTC 002e INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:17.549 UTC 002f INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:17.752 UTC 0030 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:17.758 UTC 0031 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:17.961 UTC 0032 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:17.967 UTC 0033 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:18.170 UTC 0034 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:18.176 UTC 0035 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:18.379 UTC 0036 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:18.386 UTC 0037 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:18.594 UTC 0038 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-05-24 19:57:18.611 UTC 0039 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:18.818 UTC 003a INFO [cli.common] readBlock -> Received block: 0
2022-05-24 19:57:24.142 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:24.241 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-05-24 19:57:24.514 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:24.604 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-05-24 19:57:24.899 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:24.984 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-05-24 19:57:25.217 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:25.320 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-05-24 19:57:25.484 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:25.554 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
2022-05-24 19:57:25.727 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-05-24 19:57:25.745 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update

```

Figura 115 - Inicialização do canal

Como se observa na Figura 115, quando se correu o comando ocorreram algumas falhas de conexão, mas acabou por se realizar o pretendido. Para confirmar a inicialização do canal usa-se `sudo docker exec peer<#>.org<#>.example.com peer channel getinfo -c mychannel`, em cada *host*, consoante o *peer* presente.

Este comando deverá retornar a mesma *height* de *Blockchain* e o mesmo *BlockHash* em todos os *hosts*. Tem-se *height* igual a três devido ao primeiro bloco ser o bloco de génese, o segundo reter o *update* realizado para adicionar o *anchor peer* da organização 1 e o terceiro ter sido criado com o *update* realizado para adicionar o *anchor peer* da organização 2.


```

ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo docker exec peer0.org1.example.com peer channel getinfo -c mychannel
002-05-24 19:58:58.474 UTC INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Blockchain info: {"height":3,"currentBlockHash":"AF0DjaCkmYhsVzmqCHyscpHsMSQKRqgfqrE0J+Mh24=", "previousBlockHash": "Fk+5Vct6D1kf9iyyZyQPBL7ZpGtvzYka0pgG5hHOHE="}
ubuntu@vm2:~/fabric-samples/4host-swarm-clis
ubuntu@vm2:~/fabric-samples/4host-swarm-clis$ sudo docker exec peer1.org1.example.com peer channel getinfo -c mychannel
002-05-24 19:58:53.998 UTC INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Blockchain info: {"height":3,"currentBlockHash":"AF0DjaCkmYhsVzmqCHyscpHsMSQKRqgfqrE0J+Mh24=", "previousBlockHash": "Fk+5Vct6D1kf9iyyZyQPBL7ZpGtvzYka0pgG5hHOHE="}
ubuntu@vm3:~/fabric-samples/4host-swarm-clis
ubuntu@vm3:~/fabric-samples/4host-swarm-clis$ sudo docker exec peer0.org2.example.com peer channel getinfo -c mychannel
002-05-24 19:58:52.754 UTC INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Blockchain info: {"height":3,"currentBlockHash":"AF0DjaCkmYhsVzmqCHyscpHsMSQKRqgfqrE0J+Mh24=", "previousBlockHash": "Fk+5Vct6D1kf9iyyZyQPBL7ZpGtvzYka0pgG5hHOHE="}
ubuntu@vm4:~/fabric-samples/4host-swarm-clis
ubuntu@vm4:~/fabric-samples/4host-swarm-clis$ sudo docker exec peer1.org2.example.com peer channel getinfo -c mychannel
002-05-24 19:58:51.133 UTC INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Blockchain info: {"height":3,"currentBlockHash":"AF0DjaCkmYhsVzmqCHyscpHsMSQKRqgfqrE0J+Mh24=", "previousBlockHash": "Fk+5Vct6D1kf9iyyZyQPBL7ZpGtvzYka0pgG5hHOHE="}

```

Figura 116 - Verificação das informações do blockchain a correr em cada máquina

- 9) Passando para o *chaincode*, utiliza-se *fabcar* que vem com *fabric-samples*. Caso se quisesse um *chaincode* diferente, utilizava-se os passos referentes ao *chaincode* realizados à implementação anterior da secção 5.2.3.2.

No *host 1* usa-se `sudo chmod -R 777 ../chaincode/fabcar/go`, a partir da diretoria *fabric-samples/4host-swarm-clis*, para evitar problemas de acesso.

Com os seguintes comandos ainda no *host 1*.

```

pushd ../chaincode/fabcar/go
GO111MODULE=on go mod vendor
popd

```

Muda-se a diretoria para a localização do *chaincode*, coloca-se a variável de estado *GO111MODULE* em *on* (para poder modificar a maneira como o *go* importa pacotes), usa-se `go mod vendor` (para criar uma diretoria localizada onde *go* será invocado, que contem cópias dos *packages* necessários para a *build* e *testa-os*) e volta-se para a diretoria inicial (neste caso *fabric-samples/4host-swarm-clis*).

Caso o comando *go* volte a não ser reconhecido utilizar novamente os comandos referidos no final do quarto passo desta implementação.

```

ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ pushd ../chaincode/fabcar/go
~/fabric-samples/chaincode/fabcar/go ~/fabric-samples/4host-swarm-clis ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/chaincode/fabcar/go$ GO111MODULE=on go mod vendor
go: downloading github.com/hyperledger/fabric-contract-api-go v1.1.0
go: downloading github.com/hyperledger/fabric-chaincode-go v0.0.0-20200424173110-d7076418f212
go: downloading github.com/hyperledger/fabric-protos-go v0.0.0-20200424173316-dd554ba3746e
go: downloading github.com/golang/protobuf v1.3.2
go: downloading google.golang.org/grpc v1.23.0
go: downloading github.com/xeipuuv/gojsonschema v1.2.0
go: downloading github.com/go-openapi/spec v0.19.4
go: downloading github.com/gobuffalo/packr v1.30.1
go: downloading github.com/xeipuuv/gojsonreference v0.0.0-20180127040603-bd5ef7bd5415
go: downloading github.com/go-openapi/jsonpointer v0.19.3
go: downloading github.com/go-openapi/jsonreference v0.19.2
go: downloading github.com/go-openapi/swag v0.19.5
go: downloading github.com/gobuffalo/envy v1.7.0
go: downloading github.com/gobuffalo/packd v0.3.0
go: downloading golang.org/x/net v0.0.0-20190827160401-ba9fcec4b297
go: downloading google.golang.org/genproto v0.0.0-20180831171423-11092d34479b
go: downloading github.com/xeipuuv/gojsonpointer v0.0.0-20180127040702-4e33ac2762d5f
go: downloading github.com/PuerkitoBio/purell v1.1.1
go: downloading github.com/mailru/easyjson v0.0.0-20190626092158-b2ccc519800e
go: downloading gopkg.in/yaml.v2 v2.2.8
go: downloading github.com/joho/godotenv v1.3.0
go: downloading github.com/rogppe/go-internal v1.3.0
go: downloading golang.org/x/sys v0.0.0-20190710143415-6ec70d6a5542
go: downloading github.com/PuerkitoBio/urlesc v0.0.0-20170810143723-de5bf2ad4578
go: downloading golang.org/x/text v0.3.2
ubuntu@vm1:~/fabric-samples/chaincode/fabcar/go$ popd
~/fabric-samples/4host-swarm-clis ~/fabric-samples/4host-swarm-clis

```

Figura 117 - Geração de *packages* necessários para a execução do *chaincode*


```

sudo docker exec cli peer lifecycle chaincode approveformyorg --tls
--cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem --channelID mychannel --name fabcar --
version 1 --sequence 1 --waitForEvent --package-id <package-id
obtido nos comandos anteriores>

sudo docker exec -e
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.ex
ample.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e
CORE_PEER_LOCALMSPID="Org2MSP" -e
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org
2.example.com/tls/ca.crt cli peer lifecycle chaincode
approveformyorg --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem --channelID mychannel --name fabcar --
version 1 --sequence 1 --waitForEvent --package-id <package-id
obtido nos comandos anteriores>

```

Nesta implementação o *package-id* obtido foi:

```
fabcar_1:1146b4b491871bf18b23dd67dd8cc058655b36cc0e2274f165ed06b796a
8f276
```

Usando-se então os comandos apresentados.

```

sudo docker exec cli peer lifecycle chaincode approveformyorg --tls
--cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem --channelID mychannel --name fabcar --
version 1 --sequence 1 --waitForEvent --package-id
fabcar_1:1146b4b491871bf18b23dd67dd8cc058655b36cc0e2274f165ed06b796a
8f276

sudo docker exec -e
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.ex
ample.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e
CORE_PEER_LOCALMSPID="Org2MSP" -e
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org
2.example.com/tls/ca.crt cli peer lifecycle chaincode
approveformyorg --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem --channelID mychannel --name fabcar --

```



```
version 1 --sequence 1 --waitForEvent --package-id
fabcar_1:1146b4b491871bf18b23dd67dd8cc058655b36cc0e2274f165ed06b796a
8f276
```

```
ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
b23dd67dd8cc058655b36cc0e2274f165ed06b796a8f276
[cli.lifecycle.chaincode] setOrdererClient -> Retrieved channel (mychannel) orderer endpoint: orderer.example.com:7050
[chaincodeCmd] ClientWait -> txid [835753d86813f7543a81d106785412b432719483b07b72bae4082f29333cf8f3] committed with status (VALID) at peer0.org1.example.com:7051
ubuntu@vm1: ~/fabric-samples/4host-swarm-clis$ sudo docker exec -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e CORE_PEER_LOCALMSPID="Org2MSP" -e CORE_PEER_TLS_ROOTCERT_FILES=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt cli peer lifecycle chaincode approveformyorg --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychannel --name fabcar --version 1 --sequence 1 --waitForEvent --package-id fabcar_1:1146b4b491871bf18b23dd67dd8cc058655b36cc0e2274f165ed06b796a8f276
[cli.lifecycle.chaincode] setOrdererClient -> Retrieved channel (mychannel) orderer endpoint: orderer.example.com:7050
[chaincodeCmd] ClientWait -> txid [4d1122c11b203517ccff4314e2dfb4fbc2dd3a611cbf8f95dccc007edd19ea83] committed with status (VALID) at peer0.org2.example.com:9051
```

Figura 119 – Aprovação do chaincode em ambas as organizações

Para verificar que as organizações aprovaram o *chaincode*, corre-se o comando.

```
sudo docker exec cli peer lifecycle chaincode checkcommitreadiness -
-channelID mychannel --name fabcar --version 1 --sequence 1
```

```
ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo docker exec cli peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name fabcar --version 1 --sequence 1
Chaincode definition for chaincode 'fabcar', version '1', sequence '1' on channel 'mychannel' approval status by org:
Org1MSP: true
Org2MSP: true
```

Figura 120 - Verificação da aprovação do chaincode

Finalmente falta apenas fazer *commit* do *chaincode*, que é feito com o comando, no host 1.

```
sudo docker exec cli peer lifecycle chaincode commit -o
orderer.example.com:7050 --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses peer0.org1.example.com:7051 --
tlsRootCertFiles
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --
peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --
channelID mychannel --name fabcar --version 1 --sequence 1
```

```
ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo docker exec cli peer lifecycle chaincode commit -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --channelID mychannel --name fabcar --version 1 --sequence 1
[chaincodeCmd] ClientWait -> txid [d55b1967849cd3d1f366aeecc512fcb5c6e14796a081d2c036cf9431eed3981] committed with status (VALID) at peer0.org1.example.com:7051
[chaincodeCmd] ClientWait -> txid [d55b1967849cd3d1f366aeecc512fcb5c6e14796a081d2c036cf9431eed3981] committed with status (VALID) at peer0.org2.example.com:9051
```

Figura 121 - Aprovação do chaincode

10) Passando à avaliação da rede, começa-se por utilizar a função do *chaincode* denominada *initLedger*, utilizada para popular o *ledger*, com o comando apresentado, no primeiro *host*.

```
sudo docker exec cli peer chaincode invoke -o
orderer3.example.com:9050 --tls true --cafile
```

```

/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer3.example.com/msp/tlscacerts/
tlsca.example.com-cert.pem -C mychannel -n fabcar --peerAddresses
peer0.org1.example.com:7051 --tlsRootCertFiles
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --
peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c
 '{"Args":["initLedger"]}'

```

Seguido pelo comando a utilizar a função *queryCar*, usada na procura de *items* no *ledger*, retornando a informação desse item, no *host* 1.

```

sudo docker exec cli peer chaincode query -n fabcar -C mychannel -c
 '{"Args":["queryCar","CAR0"]}'

```

```

ubuntu@vm1: ~/fabric-samples/4host-swarm-clis
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo docker exec cli peer chaincode invoke -o orderer3.example.com:9050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer3.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n fabcar --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["initLedger"]}'
2022-05-24 23:58:30.317 UTC INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
ubuntu@vm1:~/fabric-samples/4host-swarm-clis$ sudo docker exec cli peer chaincode query -n fabcar -C mychannel -c '{"Args":["queryCar","CAR0"]}'
{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}

```

Figura 122 - Inicialização de dados no ledger e invocação da função *queryCar*

Para verificar que as linhas de comando definidas na configuração funcionam, corresse o seguinte comando, em *vm2*.

```

sudo docker exec cli2 peer chaincode query -n fabcar -C mychannel -c
 '{"Args":["queryCar","CAR0"]}'

```

O seguinte comando em *vm3*.

```

sudo docker exec cli3 peer chaincode query -n fabcar -C mychannel -c
 '{"Args":["queryCar","CAR0"]}'

```

O seguinte comando em *vm4*.

```

sudo docker exec cli4 peer chaincode query -n fabcar -C mychannel -c
 '{"Args":["queryCar","CAR0"]}'

```

```

ubuntu@vm2: ~
ubuntu@vm2:~$ sudo docker exec cli2 peer chaincode query -n fabcar -C mychannel -c '{"Args":["queryCar","CAR0"]}'
{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}

ubuntu@vm3: ~
ubuntu@vm3:~$ sudo docker exec cli3 peer chaincode query -n fabcar -C mychannel -c '{"Args":["queryCar","CAR0"]}'
{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}

ubuntu@vm4: ~
ubuntu@vm4:~$ sudo docker exec cli4 peer chaincode query -n fabcar -C mychannel -c '{"Args":["queryCar","CAR0"]}'
{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}

```

Figura 123 - Testagem da funcionalidade da rede em todos os *hosts*

Onde observa-se que o valor invocado pelo *host* 1 está presente em todos os outros *hosts*, usando as linhas de comando de cada organização.

5.2.3.7 Criação de um script para gerar redes consoante o número de Peers e Organizações são definidos pelo utilizador

Nesta implementação pretendeu-se criar um script que cria documentos de configuração de acordo com o número de organizações e *peers* definidos pelo utilizador.

Para simplificar a execução da implementação, é escolhida a configuração com duas organizações e dois *peers* em cada organização tal como foi usado na prévia implementação (secção 5.2.3.6). Os passos utilizados na implementação são descritos de seguida:

- 0) Antes da criação da implementação é necessário, tal como na secção 5.2.3.6, ter os documentos da directoria *MultipassFolder* prontos. Para tal usa-se o comando `git clone https://github.com/jnvr/MultipassFolder.git`. Após o download do repositório mudar as permissões de acesso na directoria `./MultipassFolder/config_maker_script` com o comando `sudo chmod -R 777 ./MultipassFolder/config_maker_script`.
- 1) A criação dos documentos de configuração, irá ocorrer com a execução do script presente na directoria `config_maker_script` denominado `makeConfigFiles.sh`, script este que utiliza uma mistura de comandos `yq` (para editar ficheiros `.yaml` e adicionar estruturas) e `sed` (para editar valores nos ficheiros `.yaml`). O script encontra-se dividido em várias secções, que são descritas de seguida:
 - Inicialização dos valores de *peer* e organizações: nesta secção é pedido ao utilizador para declarar o número de organizações e *peers* por organização
 - Configuração de `configtx.yaml`: inicialmente faz-se *reset* de `configtx.yaml`. Depois introduzem-se estruturas *template* por cada organização (presente em `OrgStructureConftx.yaml`) e editam-se os valores do *template* consoante a organização a ser declarada. Adicionam-se as novas organizações ao *profile* `TwoOrgsChannel`. Depois adiciona-se a estrutura *template* de `orderer` (presente em `OrdererStructureConftx.yaml`) para cada *peer* e editam-se os valores tendo em conta os *peers*. Seguidamente criam-se novos `orderers` no *profile* `SampleMultiNodeEtcdRaft.Orderer` e `SampleMultiNodeEtcdRaft.EtcdRaft` para cada *peer* e para `cli` (linha de

Figura 124 - Script `makeConfigFiles.sh`, inicialização dos valores de *peer* e organizações

```
1 #!/bin/sh
2
3 printf "\v $(tput setaf 7)$(tput setab 4)$(tput bold) All configurations are meant to be multi-h
ost and will run a peer in each machine $(tput sgr 0)\n\n"
4 echo "Introduce number of Organizations"
5 read -r OrgNUM
6 echo "Introduce number of Peers per Organization"
7 read -r PeerNUM
8
9 lastPeerjter=$((PeerNUM-1))
```

comandos). Finalmente adicionam-se organizações ao *profile* *SampleMultiNodeEtcDRaft*.

```
11 #####
12 ##### Configuration of configtx.yaml #####
13 #####
14 #####
15 cat configtx1.yaml > configtx.yaml # Reset modifications in configtx.yaml
16
17 iter=1
18 ordererPort=7050
19 orgPort=7051
20 ordererNumber=2
21 while [ "$iter" -le "$OrgNUM" ];do
22   printf "\n$(tput setaf 5)$(tput bold)Adding OrgSite to Organizations in configtx.yaml $(tput sgr 0)"
23   item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Organizations += env(item)" configtx.yaml # Add new Org structure t
24   o configtx.yaml
25   sed -i -s//merge //g' configtx.yaml # To remove an unexpected result of the previous command
26
27   # Change values of the new Organization structure
28   sed -i s/'orgName'/'OrgSite' / configtx.yaml
29   sed -i s/'orgName'/'OrgSiteMSP' / configtx.yaml
30   sed -i s/'readRule'/'\OR('OrgSiteMSP.admin', 'OrgSiteMSP.peer', 'OrgSiteMSP.client)'\*/ configtx.yaml
31   sed -i s/'writeRule'/'\OR('OrgSiteMSP.admin', 'OrgSiteMSP.client)'\*/ configtx.yaml
32   sed -i s/'adminRole'/'\OR('OrgSiteMSP.admin)'\*/ configtx.yaml
33   sed -i s/'endorseRule'/'\OR('OrgSiteMSP.peer)'\*/ configtx.yaml
34   sed -i s/'anchorPeer'/'peer0.orgsite.example.com' / configtx.ya
35   sed -i s/'portNum'/'OrgPort' / configtx.ya
36
37
38   # Add new Org to TwoOrgChannel profile
39   printf "\n$(tput setaf 5)$(tput bold)Adding OrgSite to TwoOrgChannel profile in configtx.yaml $(tput sgr 0)"
40   m="OrgSite" yq -i ".Profiles.TwoOrgChannel.Application.Organizations += strenv(m)" configtx.yaml
41   sed -i s/"\OrgSite"/"OrgSite"/ configtx.yaml # Remove quotes created from the previous command
42   sed -i -s/. Textsample/d' configtx.yaml
43
44   jter=0
45   while [ "$jter" -lt "$PeerNUM" ]; do
46     if [ "$iter" -eq 1 ] && [ "$jter" -eq 0 ]; then
47       printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer0.Org0 to SampleMultiNodeEtcDRaft.EtcDRa
48       ft profile in configtx.yaml $(tput sgr 0)"
49       item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.EtcDRaft.Conse
50       nters += env(item)" configtx.yaml # Add new Orderer structure to configtx.yaml
51       sed -i s/'hostOrderer'/'orderer.example.com' / configtx.yaml
52       sed -i s/'ordPort'/'SordererPort' / configtx.yaml
53       sed -i s/'tlsOrderer'/'crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/v
54       tls/server.crt' / configtx.yaml
55       sed -i -s/. InsertText/d' configtx.yaml
56
57       # Add an orderer to a peer in SampleMultiNodeEtcDRaft.Orderer profile
58       printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer0.Org0 to SampleMultiNodeEtcDRaft.Ordere
59       r profile in configtx.yaml $(tput sgr 0)"
60       m="orderer.example.com:SordererPort" yq -i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.Addresses += stren
61       v(m)" configtx.yaml
62       sed -i -s/. Textinsert/d' configtx.yaml
63       ordererPort=$((ordererPort+1000))
64
65       elif [ "$jter" -eq "$OrgNUM" ] && [ "$jter" -eq "$LastPeerjter" ]; then
66
67         # Add an orderer to a peer in SampleMultiNodeEtcDRaft.EtcDRaft profile
68         printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer$jter.OrgSite to SampleMultiNodeEtcDRaf
69         t.EtcDRaft profile in configtx.yaml $(tput sgr 0)"
70         item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.EtcDRaft.Conse
71         nters += env(item)" configtx.yaml # Add new Orderer structure to configtx.yaml
72         sed -i s/'hostOrderer'/'orderer.SordererNumber.example.com' / configtx.yaml
73         sed -i s/'ordPort'/'SordererPort' / configtx.yaml
74         sed -i s/'tlsOrderer'/'crypto-config/ordererOrganizations/example.com/orderers/orderer.SordererNumber
75         .example.com/vtls/server.crt' / configtx.yaml
76
77         # Add an orderer to a peer in SampleMultiNodeEtcDRaft.Orderer profile
78         printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer$jter.OrgSite to SampleMultiNodeEtcDRaf
79         t.Orderer profile in configtx.yaml $(tput sgr 0)"
80         m="orderer.SordererNumber.example.com:SordererPort" yq -i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.Addr
81         esses += strenv(m)" configtx.yaml
82         ordererNumber=$((ordererNumber+1))
83         ordererPort=$((ordererPort+1000))
84
85         # Add an orderer for cli to SampleMultiNodeEtcDRaft.EtcDRaft profile
86         printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for cli to SampleMultiNodeEtcDRaft.EtcDRaft prof
87         ile in configtx.yaml $(tput sgr 0)"
88         item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.EtcDRaft.Conse
89         nters += env(item)" configtx.yaml # Add new Orderer structure to configtx.yaml
90         sed -i s/'hostOrderer'/'orderer.SordererNumber.example.com' / configtx.yaml
91         sed -i s/'ordPort'/'SordererPort' / configtx.yaml
92         sed -i s/'tlsOrderer'/'crypto-config/ordererOrganizations/example.com/orderers/orderer.SordererNumber
93         .example.com/vtls/server.crt' / configtx.yaml
94
95         # Add an orderer for cli to SampleMultiNodeEtcDRaft.Orderer profile
96         printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for cli to SampleMultiNodeEtcDRaft.Orderer profi
97         le in configtx.yaml $(tput sgr 0)"
98         m="orderer.SordererNumber.example.com:SordererPort" yq -i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.Addr
99         esses += strenv(m)" configtx.yaml
100        ordererNumber=$((ordererNumber+1))
101        ordererPort=$((ordererPort+1000))
102
103        else
104
105          # Add an orderer to a peer in SampleMultiNodeEtcDRaft.EtcDRaft profile
106          printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer$jter.OrgSite to SampleMultiNodeEtcDRaf
107          t.EtcDRaft profile in configtx.yaml $(tput sgr 0)"
108          item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.EtcDRaft.Conse
109          nters += env(item)" configtx.yaml # Add new Orderer structure to configtx.yaml
110          sed -i s/'hostOrderer'/'orderer.SordererNumber.example.com' / configtx.yaml
111          sed -i s/'ordPort'/'SordererPort' / configtx.yaml
112          sed -i s/'tlsOrderer'/'crypto-config/ordererOrganizations/example.com/orderers/orderer.SordererNumber
113          .example.com/vtls/server.crt' / configtx.yaml
114
115          # Add an orderer to a peer in SampleMultiNodeEtcDRaft.Orderer profile
116          printf "\n$(tput setaf 5)$(tput bold)Adding new orderer for Peer$jter.OrgSite to SampleMultiNodeEtcDRaf
117          t.Orderer profile in configtx.yaml $(tput sgr 0)"
118          m="orderer.SordererNumber.example.com:SordererPort" yq -i ".Profiles.SampleMultiNodeEtcDRaft.Orderer.Addr
119          esses += strenv(m)" configtx.yaml
120          ordererNumber=$((ordererNumber+1))
121          ordererPort=$((ordererPort+1000))
122
123          fi
124          jter=$((jter+1))
125        done
126
127        # Add new Org to SampleMultiNodeEtcDRaft profile
128        printf "\n$(tput setaf 5)$(tput bold)Adding new OrgSite to Organizations in configtx.yaml $(tput sgr 0)"
129        item=$(cat OrdererStructureConfix.yaml) yq -e-i ".Organizations += env(item)" configtx.yaml # Add new Org structure t
130        o configtx.yaml
131        sed -i s/"\OrgSite"/"OrgSite"/ configtx.yaml # Remove quotes created from the previous command
132        sed -i -s/. sampleText/d' configtx.yaml
133        orgPort=$((orgPort+PeerNUM*1000))
134        iter=$((iter+1))
135      done
136    sed -i -s//merge //g' configtx.yaml # To remove unexpected results from the previous while loop
```

Figura 125 - Script *makeConfigFiles.sh*, configuração de *configtx.yaml*

- Configuração de *crypto-config.yaml*: inicialmente faz-se *reset* de *crypto-config.yaml*. Depois adicionam-se os valores de *orderer* consoante o número de *peers*. Finalmente, adiciona-se a estrutura *template* (presente em *createOrgCryCon.yaml*) para cada organização e editam-se os valores do *template* conforme a organização.

```

116 #####
117 ##### Configuration of crypto-config.yaml #####
118 #####
119
120 cat crypto-config1.yaml > crypto-config.yaml # Reset modifications in crypto-config.yaml
121
122 lastPortOrderer=$((ordererPort-1000))
123 ordererNumber=$((ordererNumber-1))
124 iter=2
125 while [ "$iter" -le "$ordererNumber" ]; do
126     printf "\n$(tput setaf 6)$$(tput bold)Adding orderer$iter to crypto-config.yaml $(tput sgr 0)"
127     m="Hostname: orderer$iter" yq -i '.OrdererOrgs[0].Specs += strenv(m)' crypto-config.yaml
128     sed -i s/"\Hostname: orderer$iter"/"Hostname: orderer$iter"/ crypto-config.yaml # Remove quotes created
        from the previous command
129     iter=$((iter+1))
130 done
131
132 printf "\n$(tput setaf 6)$$(tput bold)Adding Org1 to crypto-config.yaml $(tput sgr 0)"
133 yq -i ".PeerOrgs[0].Template.Count = $PeerNUM" crypto-config.yaml # Edit peer number of Org1
134 yq -i ".PeerOrgs[0].Users.Count = $lastPeerjter" crypto-config.yaml # Edit user number of Org1
135
136 iter=1
137 while [ "$iter" -lt "$OrgNUM" ];do
138     i=$((iter+1))
139     printf "\n$(tput setaf 6)$$(tput bold)Adding Org$i to crypto-config.yaml $(tput sgr 0)"
140     item=$(cat createOrgCryCon.yaml) yq -ei '.PeerOrgs += env(item)' crypto-config.yaml # Add new Org structur
        e to crypto-config.yaml "https://stackoverflow.com/questions/68321476/insert-multiple-lines-of-one-attribute-i
        n-yaml-file-with-yq-v4-x"
141     yq -ei ".PeerOrgs[${iter}].Template.Count = ${PeerNUM}" crypto-config.yaml # Edit peer number of Org<#>
142     yq -ei ".PeerOrgs[${iter}].Users.Count = ${lastPeerjter}" crypto-config.yaml # Edit user number of Org<#>
143
144     # First try where there were problems passing strings: "Error: parsing expression: Lexer error: could not
        match text"
145     # yq -i ".PeerOrgs[${iter}].Domain = "org$i.example.com"" crypto-config.yaml #
146     # yq -ei ".PeerOrgs[${iter}].Name |= Org$i" crypto-config.yaml
147
148     sed -i s/"nameInput"/"Org$i"/ crypto-config.yaml # Changing organization Name
149     sed -i s/"domainInput"/"org$i.example.com"/ crypto-config.yaml # Changing organization Domain
150     iter=$((iter+1))
151 done

```

Figura 126 - Script *makeConfigFiles.sh*, configuração de *crypto-config.yaml*

- Configuração de *docker-compose-base.yaml*: inicialmente apagam-se as modificações realizadas previamente. Depois adiciona-se a estrutura *template* (presente em *peerStructureDocker.yaml*) todos *peers* na rede e editam-se os valores para cada *peer*.


```

154 #####
155 ##### Configuration of base/docker-compose-base.yaml #####
156 #####
157
158 sed -i '22,$d' docker-compose-base.yaml # Reset docker-compose-base.yaml file to its original form
159
160 iter=1
161 peerPort=7051
162 initpeerPort=7051
163 while [ "$iter" -le "$OrgNUM" ]; do
164     bootOrgIter=$((iter-1))
165     jter=0
166     while [ "$jter" -lt "$PeerNUM" ]; do
167         # Add peers to the Organization
168         printf "\n$(tput setaf 3)$(tput bold)Adding peer$jter.Org$iter to docker-compose-base.yaml $(tput sgr 0)"
169         item=$(cat peerStructureDocker.yaml) yq -ei '.services += env(item)' docker-compose-base.yaml
170         sed -i s/"peerName"/"peer$jter"/ docker-compose-base.yaml
171         sed -i s/"orgName"/"org$iter.example.com"/ docker-compose-base.yaml
172         sed -i s/"peerPort"/"$peerPort"/g docker-compose-base.yaml #In this command it is necessary to use 'g' mod
173         #ifier due to sed only replacing the first instance in a line by default
174         sed -i s/"peerchainPort"/"$((peerPort+1))"/ docker-compose-base.yaml
175
176         # Define CORE_PEER_GOSSIP_BOOTSTRAP as every other peer in the Organization
177         bootIter=0
178         bootPeerPort=$((initpeerPort+1000*PeerNUM*bootOrgIter))
179         while [ "$bootIter" -lt "$PeerNUM" ]; do
180             if [ "$bootIter" -eq "$jter" ]; then
181                 bootIter=$((bootIter+1))
182                 bootPeerPort=$((bootPeerPort+1000))
183             else
184                 sed -i s/"peerBoot"/"peer$bootIter"/ docker-compose-base.yaml
185                 sed -i s/"orgBoot"/"org$iter.example.com"/ docker-compose-base.yaml
186                 sed -i s/"portBoot"/"$bootPeerPort peerBoot.orgBoot:portBoot"/ docker-compose-base.yaml
187                 bootIter=$((bootIter+1))
188                 bootPeerPort=$((bootPeerPort+1000))
189             fi
190         done
191         sed -i 's/ peerBoot.orgBoot:portBoot/' docker-compose-base.yaml
192
193         sed -i s/"orgMSP"/"Org$iter\MSP"/ docker-compose-base.yaml
194         sed -i s/"mspVolume"/"../crypto-config/peerOrganizations/org$iter.example.com/peers/peer$jter.orgsite
195         r.example.com/msp:/etc/hyperledger/fabric/msp"/ docker-compose-base.yaml
196         sed -i s/"tlsVolume"/"../crypto-config/peerOrganizations/org$iter.example.com/peers/peer$jter.orgsite
197         r.example.com/tls:/etc/hyperledger/fabric/tls"/ docker-compose-base.yaml
198         peerPort=$((peerPort+1000))
199         jter=$((jter+1))
200     done
201     iter=$((iter+1))
202 done

```

Figura 127 - Script makeConfigFiles.sh, configuração de docker-compose-base.yaml

- Criação dos *host<#>.yaml* de cada *peer*: inicialmente faz-se *reset* dos documentos de *host<#>.yaml* previamente criados. Depois utiliza-se a estrutura presente em *host.yaml* para criar cada um dos documentos e editam-se os valores consoante o *host*.

```

202 #####
203 ##### Creation of host<#>.yaml for each peer #####
204 #####
205
206 cat host1Structure.yaml > host1.yaml # Reset host1.yaml config
207 rm -r ./configDocs/host*.yaml 2> /dev/null # Reset the rest of host<#>.yaml, the error output is being ignored
as it is irrelevant here
208
209 iter=1
210 number=2
211 numPort=8050
212 while [ "$siter" -le "$orgNUM" ]; do
213     jter=0
214     while [ "$sjter" -lt "$PeerNUM" ]; do
215         if [ "$siter" -eq 1 ] && [ "$sjter" -eq 0 ]; then
216             # Create host1.yaml
217             printf "\n$(tput setaf 1)$(tput bold)Configuring host1.yaml $(tput sgr 0)"
218             sed -i s/"lastOrderer"/"orderer$ordererNumber.example.com"/ host1.yaml
219             sed -i s/"lastPortOrderer"/"$lastPortOrderer"/g host1.yaml
220             mv ./host1.yaml ./configDocs
221             jter=$((jter+1))
222         else
223             # Create the rest of the host<#>.yaml
224             printf "\n$(tput setaf 1)$(tput bold)Configuring host$number.yaml $(tput sgr 0)"
225             cat host.yaml > host$number.yaml
226             chmod 777 host$number.yaml
227             sed -i s/"ordererName"/"orderer$number.example.com"/ host$number.yaml
228             sed -i s/"peerName"/"peer$sjter"/ host$number.yaml
229             sed -i s/"orgName"/"org$siter.example.com"/ host$number.yaml
230             sed -i s/"portOrderer"/"$numPort"/g host$number.yaml
231             mv ./host$number.yaml ./configDocs
232             number=$((number+1))
233             numPort=$((numPort+1000))
234             jter=$((jter+1))
235         fi
236     done
237     iter=$((iter+1))
238 done
239

```

Figura 128 - Script `makeConfigFiles.sh`, criação dos ficheiros `host<#>.yaml`

- Criação de `mychannel.sh`: inicialmente apagam-se as modificações realizadas previamente. Depois adicionam-se os comandos de juntar todos os `peers` ao canal, conforme os `peers` existentes e os comandos de adicionar `anchor peers`, um por organização.

```

242 #####
243 ##### Creation of mychannel.sh #####
244 #####
245
246 sed -i '5,5d' mychannel.sh # Reset mychannel.sh to its inicial form
247
248 # Join all peers to channel
249 printf "\n$(tput setaf 4)$(tput bold)Configuring mychannel.sh $(tput sgr 0)"
250 iter=2
251 ordPort=9051
252 while [ "$siter" -le "$orgNUM" ]; do
253     jter=0
254     while [ "$sjter" -lt "$PeerNUM" ]; do
255         if [ "$siter" -eq 1 ]; then
256             echo "docker exec -e CORE_PEER_ADDRESS=peer$sjter.org$siter.example.com:$ordPort -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org$siter.example.com/peers/peer$sjter.org$siter.example.com/tls/ca.crt cli peer channel join -b mychannel.block" >> mychannel.sh
257             ordPort=$((ordPort+1000))
258             jter=$((jter+1))
259         else
260             echo "docker exec -e CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org$siter.example.com/users/Admin@org$siter.example.com/msp -e CORE_PEER_ADDRESS=peer$sjter.org$siter.example.com:$ordPort -e CORE_PEER_LOCALMSPID=\"Org${siter}MSP\" -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org$siter.example.com/peers/peer0.org$siter.example.com/tls/ca.crt cli peer channel join -b mychannel.block" >> mychannel.sh
261             ordPort=$((ordPort+1000))
262             jter=$((jter+1))
263         fi
264     done
265     iter=$((iter+1))
266 done
267
268 echo "docker exec cli peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" >> mychannel.sh
269
270 corePeerOrg=7051
271 while [ "$siter" -le "$orgNUM" ]; do
272     corePeerOrg=$((corePeerOrg+PeerNUM*1000))
273     echo "docker exec -e CORE_PEER MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org$siter.example.com/users/Admin@org$siter.example.com/msp -e CORE_PEER_ADDRESS=peer0.org2.example.com:$corePeerOrg -e CORE_PEER_LOCALMSPID=\"Org${siter}MSP\" -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org$siter.example.com/peers/peer0.org$siter.example.com/tls/ca.crt cli peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" >> mychannel.sh
274     iter=$((iter+1))
275 done

```

Figura 129 - Script `makeConfigFiles.sh`, criação de `mychannel.sh`

- Criação de *host<#>up.sh* e *host<#>down.sh* de cada *peer*: inicialmente eliminam-se os documentos previamente criados. Seguidamente criam-se e movem-se os scripts para a directoria *./MultipassFolder/config_maker_script/configDocs*.

```

278 #####
279 ##### Creation of host<#>up.sh & host<#>down.sh #####
280 #####
281
282 rm -r ./configDocs/host*.sh 2> /dev/null # Reset host<#>up.sh & host<#>down.sh files created previously, the e
rror output is being ignored as it is irrelevant here
283
284 hostNumber=1
285 iter=1
286 while [ "$siter" -le "$OrgNUM" ]; do
287     jter=0
288     while [ "$jter" -lt "$PeerNUM" ]; do
289         # Creation of host<#>up.sh
290         printf "\n$(tput setaf 0)$(tput bold)Creating host${hostNumber}up.sh $(tput sgr 0)"
291         echo "docker-compose -f host${hostNumber}.yaml up -d" > host${hostNumber}up.sh
292
293         # Creation of host<#>down.sh
294         printf "\n$(tput setaf 0)$(tput bold)Creating host${hostNumber}down.sh $(tput sgr 0)"
295         sed "li docker-compose -f host${hostNumber}.yaml down -v" hostDownStructure.sh > host${hostNumber}down.s
h
296
297         mv ./host${hostNumber}up.sh ./configDocs && mv ./host${hostNumber}down.sh ./configDocs
298         hostNumber=$((hostNumber+1))
299         jter=$((jter+1))
300     done
301     iter=$((iter+1))
302 done

```

Figura 130 - Script *makeConfigFiles.sh*, criação dos ficheiros *host<#>up.sh* e *host<#>down.sh*

- Envio de documentos de configuração modificados para *configDocs*: finalmente enviam-se os ficheiros criados que restam para a directoria *configDocs*.

```

304 echo # To add a new line at the end
305 cp ./configtx.yaml ./configDocs/configtx.yaml
306 cp ./crypto-config.yaml ./configDocs/crypto-config.yaml
307 cp ./docker-compose-base.yaml ./configDocs/docker-compose-base.yaml
308 cp ./mychannel.sh ./configDocs/mychannel.sh

```

Figura 131 - Script *makeConfigFiles.sh*, envio de documentos de configuração modificados para *configDocs*

2) Passa-se então para a execução em si do script *makeConfigFiles.sh*.

```

ubuntu@vm1:~/MultipassFolder/config_maker_script$ ./makeConfigFiles.sh
All configurations are meant to be multi-host and will run a peer in each machine
Introduce number of Organizations
2
Introduce number of Peers per Organization
2
Adding Org1 to Organizations in configtx.yaml
Adding Org1 to TwoOrgsChannel profile in configtx.yaml
Adding new orderer for Peer0.Org0 to SampleMultiNodeEtcdRaft.EtcdRaft profile in configtx.yaml
Adding new orderer for Peer0.Org0 to SampleMultiNodeEtcdRaft.Orderer profile in configtx.yaml
Adding new orderer for Peer1.Org1 to SampleMultiNodeEtcdRaft.EtcdRaft profile in configtx.yaml
Adding new orderer for Peer1.Org1 to SampleMultiNodeEtcdRaft.Orderer profile in configtx.yaml
Adding Org2 to Organizations in configtx.yaml
Adding Org2 to TwoOrgsChannel profile in configtx.yaml
Adding new orderer for Peer0.Org2 to SampleMultiNodeEtcdRaft.EtcdRaft profile in configtx.yaml
Adding new orderer for Peer0.Org2 to SampleMultiNodeEtcdRaft.Orderer profile in configtx.yaml
Adding new orderer for Peer1.Org2 to SampleMultiNodeEtcdRaft.EtcdRaft profile in configtx.yaml
Adding new orderer for Peer1.Org2 to SampleMultiNodeEtcdRaft.Orderer profile in configtx.yaml
Adding new orderer for cli to SampleMultiNodeEtcdRaft.EtcdRaft profile in configtx.yaml
Adding new orderer for cli to SampleMultiNodeEtcdRaft.Orderer profile in configtx.yaml
Adding orderer2 to crypto-config.yaml
Adding orderer3 to crypto-config.yaml
Adding orderer4 to crypto-config.yaml
Adding orderer5 to crypto-config.yaml
Adding Org1 to crypto-config.yaml
Adding Org2 to crypto-config.yaml
Adding peer0.Org1 to docker-compose-base.yaml
Adding peer1.Org1 to docker-compose-base.yaml
Adding peer0.Org2 to docker-compose-base.yaml
Adding peer1.Org2 to docker-compose-base.yaml
Configuring host1.yaml
Configuring host2.yaml
Configuring host3.yaml
Configuring host4.yaml
Configuring mychannel.sh
Creating host1up.sh
Creating host1down.sh
Creating host2up.sh
Creating host2down.sh
Creating host3up.sh
Creating host3down.sh
Creating host4up.sh
Creating host4down.sh

```

Figura 132 - Execução do script makeConfigFiles.sh

Criando os ficheiros obtém-se a configuração de *hosts* presente na Figura 133.

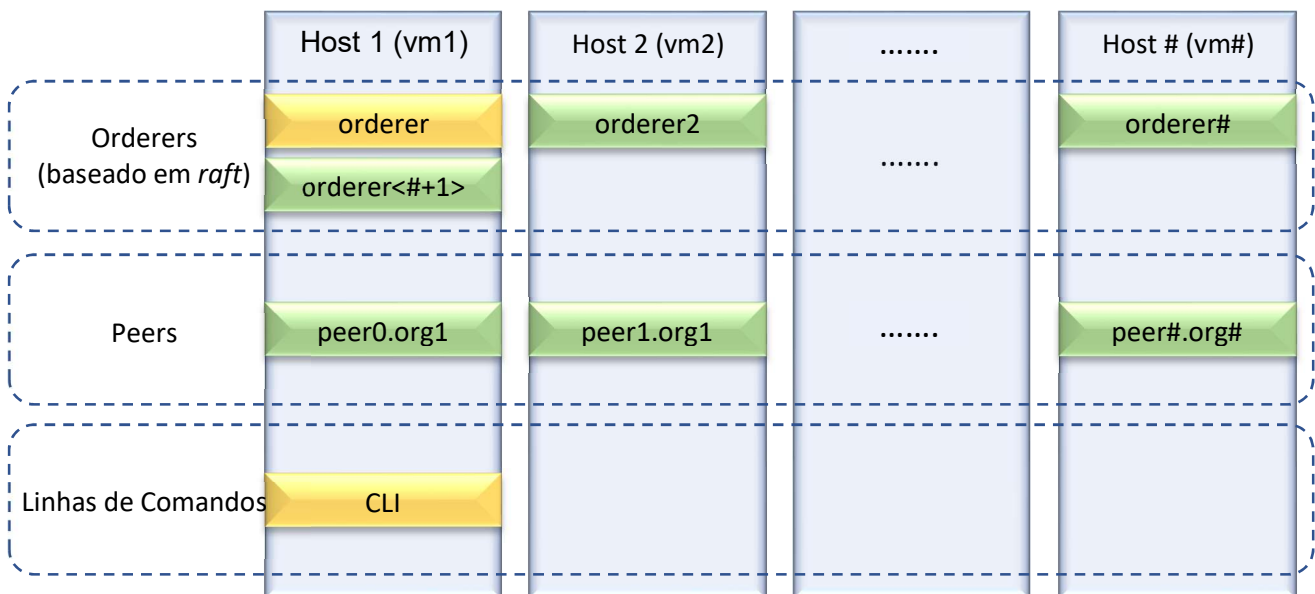


Figura 133 - Configuração de hosts obtida utilizando makeConfigFiles.sh

- 3) Finalmente, em todas as máquinas, substituem-se os ficheiros em *configDocs* na diretoria *4host-swarm-clis* mencionada na secção 5.2.3.6, exceto o documento *docker-compose-base.yaml* que irá substituir para a diretoria *4host-swarm-clis/base*.
- 4) Para fazer a testagem basta voltar a percorrer os passos da secção 5.2.3.6, obtendo-se os mesmos resultados.

5.3 Ethereum-Geth

A opção do uso de Ethereum foi apenas apresentada e reconhecida numa fase bastante tardia no processo da realização da dissertação, pelo que não foi possível fazer uma implementação aprofundada da ideia descrita na secção 4.2 deste documento.

A implementação realizada baseia-se na realização o tutorial presente em [32], de seguida ir-se-á descrever o sistema operativo utilizado, o pré-requisito e a implementação.

5.3.1 Sistema operativo

5.3.1.1 Multipass - Ubuntu

Nesta implementação, tal como na secção 5.2.1.2, utiliza-se *multipass* que corre Ubuntu 20.04. Neste caso são criadas duas máquinas virtuais, *machine1* e *machine2*.

```
PS C:\Users\Develop> multipass launch -n machine1
Launched: machine1
PS C:\Users\Develop> multipass launch -n machine2
Launched: machine2
```

Figura 134 - Criação de máquinas virtuais

5.3.2 Pré-requisito

Na implementação é adquirido que se está a utilizar um sistema atualizado. Para isso corre-se o comando `sudo apt update && sudo apt upgrade`.

5.3.1.1 Geth

Tal como foi mencionado anteriormente na secção 2.2.6, *Geth* é um *standalone client* empregado para poder usar a implementação em linguagem *go* do protocolo *Ethereum*. Para instalar existem diversas possibilidades presentes em [33]. Neste caso usaram-se os seguintes comandos.

```
sudo add-apt-repository -y ppa:ethereum/Ethereum
sudo apt-get update
sudo apt-get install Ethereum
```

5.3.3 2.1.1 Implementação de uma rede privada com dois peers em máquinas diferentes

Nesta implementação pretendeu-se criar uma rede privada com dois *peers*, cada um numa VM. De seguida são descritos os passos necessários.

- 1) Inicialmente, na VM *machine1* ir-se-á criar a diretoria `/tmp/eth/60` onde irá correr o comando apresentado de seguida.

```
geth --datadir="/tmp/eth/60/01" -verbosity 6 --ipcdisable --port
30301 --http.port 8101 console 2>> /tmp/eth/60/01.log
```


Com este comando inicia-se a consola de JavaScript do Geth, no comando descreve-se onde a instância corre, os portos a serem utilizados, entre outros. No final os erros são enviados para a diretoria `/tmp/eth/60/01.log`.

```
ubuntu@machine1:~$ mkdir -p /tmp/eth/60
ubuntu@machine1:~$ geth --datadir="/tmp/eth/60/01" --verbosity 6 --ipcdisable --port 30301 --http.port 8101 console 2>> /tmp/eth/60/01.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.20-stable-8f2416a8/linux-amd64/go1.18.1
at block: 0 (Thu Jan 01 1970 01:00:00 GMT+0100 (CET))
datadir: /tmp/eth/60/01
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

Figura 135 - Inicialização da consola de geth na VM machine1

- 2) Seguidamente utiliza-se o comando apresentado de seguida para obter o `enode`, que identifica o nodo de *Ethereum* onde se está presentemente.

`admin.nodeInfo.enode`

No caso desta implementação obtém-se o seguinte valor.

"enode://a0f8152465752aa0b5d3b2b8de5b45c94ae057291684b8a25452de82cc4033f35ad505f3466cde51401c04dac815759dadf3c73335db104073505f13e5d7d@127.0.0.1:30301"

```
> admin.nodeInfo.enode
"enode://a0f8152465752aa0b5d3b2b8de5b45c94ae057291684b8a25452de82cc4033f35ad505f3466cde51401c04dac815759dadf3c73335db104073505f13e5d7d@127.0.0.1:30301"
```

Figura 136 - Execução do comando `admin.nodeInfo.enode`

- 3) Posteriormente repete-se o passo 1 na VM machine2.

```
ubuntu@machine2:~$ mkdir -p /tmp/eth/60
ubuntu@machine2:~$ geth --datadir="/tmp/eth/60/02" --verbosity 6 --ipcdisable --port 30302 --http.port 8102 console 2>> /tmp/eth/60/02.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.20-stable-8f2416a8/linux-amd64/go1.18.1
at block: 0 (Thu Jan 01 1970 01:00:00 GMT+0100 (CET))
datadir: /tmp/eth/60/02
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

Figura 137 - Inicialização da consola geth na VM machine2

- 4) Ainda na VM machine2, corre-se o comando apresentado de seguida para criar a rede privada constituída por *machine1* e *machine2*. Neste comando vai-se utilizar o enode obtido no passo 2, substituindo-se o valor de *localhost* (127.0.0.1) com o IP da VM *machine1*. Se ambas as máquinas estiverem na mesma rede de internet utiliza-se o IP, caso contrário utiliza-se IP externo (pode ser obtido através de pesquisa na internet). Estando a ser utilizadas VM de *multipass*, o IP é obtido com o comando `multipass ls`, executado fora das VM.

```
PS C:\Users\Develop> multipass ls
Name      State      IPv4          Image
machine1  Running   172.20.250.152 Ubuntu 20.04 LTS
machine2  Running   172.20.242.31 Ubuntu 20.04 LTS
```

Figura 138 - Execução do comando `multipass ls`

`admin.addPeer(<enode obtido no passo 2 modificado >)`

No caso específico da implementação foi usado o seguinte comando.

```
admin.addPeer("enode://a0f8152465752aa0b5d3b2b8de5b45c94ae057291684b8a25452de82cc4033f35ad505f3466cdedba51401c04dac815759dadf3c73335db104073505f13e5d7d@172.20.250.152:30301")
```

```
> admin.addPeer("enode://a0f8152465752aa0b5d3b2b8de5b45c94ae057291684b8a25452de82cc4033f35ad505f3466cdedba51401c04dac815759dadf3c73335db104073505f13e5d7d@172.20.250.152:30301")
true
```

Figura 139 - Execução do comando `admin.addPeer`

- 5) Finalmente, observa-se que se criou uma rede com ambas as VM utilizando o comando `net.peerCount` na VM *machine2*, onde se observa a presença de dois *peers*.

```
> net.listening
true
> net.peerCount
2
```

Figura 140 - Execução do comando `net.peerCount`

5.4 Avaliações

Comparando as implementações realizadas usando HF e a pequena implementação realizada em *Ethereum*, verifica-se que ambas as plataformas têm características interessantes e características desfavoráveis.

Enquanto HF é uma opção bastante interessante devido à segurança inerente de uma rede de natureza *permissioned* e por se programar os *chaincodes* com linguagens de programação bastantes conhecidas (*java*, *javascript*, *go*). Esta plataforma também tem alguns problemas relativos à sua falta de flexibilidade (para mudar a configuração da rede é necessários bastantes passos) e falta de recursos existentes derivado da sua relativa obscuridade.

No caso do *Ethereum* tem-se mais flexibilidade (uma vez que foi essa a intenção na sua criação, onde se podem criar diferentes *chaincodes* e *dApps*) e, mais importante, a grande facilidade de acesso a recursos, de modo a ultrapassar problemas encontrados, devido à enorme comunidade de *developers* e utilizadores. Porém, para programar em *Ethereum* tem de se aprender uma nova linguagem de programação, denominada *Solidity* e é necessário criar nodos dedicados a estabelecer a segurança de uma rede.

Concluindo apesar de ambas as plataformas serem alternativas interessantes, *Ethereum* seria a melhor opção, muito por causa da abundância de recursos disponíveis.

6 Trabalho

6.1 Trabalho realizado

Durante a Dissertação este trabalho focou-se maioritariamente na criação das diferentes implementações usando *Hyperledger Fabric*, seguido por uma breve familiarização e implementação com *Ethereum*, durante todo este período o documento da dissertação foi sendo escrito.

	Março			Abril					Maio				Junho				Julho	
Semanas:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Tarefas de Implementação em Hyperledger Fabric	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█		
Familiarização e implementação com Ethereum																	█	█
Escrita da Dissertação			█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█

Figura 141 - Tabela do trabalho realizado

6.2 Trabalho futuro

No futuro seria interessante realizar a implementação de *Ethereum* mencionada na secção 4.2 da dissertação. Adicionalmente penso que seria uma boa ideia acompanhar a evolução de *Hyperledger Fabric* com novas implementações e configurações ainda mais detalhadas.

Referências

- [1] Rivadeneira-Muñoz, Jorge Eduardo [et al.]; "A Privacy-Aware Framework integration into a Human-in-the-Loop IoT System", ACM Surveys (submitted), June 2021.
- [2] Plotly, "Dash". Disponível em: <https://plotly.com/dash/>. Acedido em 30-junho-2021.
- [3] Xorum.io, "Cross-platform mobile apps development in 2021: Xamarin vs React Native vs Flutter vs Kotlin Multiplatform". Disponível em: <https://medium.com/xorum-io/cross-platform-mobile-apps-development-in-2021-xamarin-vs-react-native-vs-flutter-vs-kotlin-ca8ea1f5a3e0>. Acedido em 30-junho-2021.
- [4] Microsoft. "What is the Xamarin.Forms technology stack?". Disponível em: <https://www.microsoft.com/pt-pt/vidoplayer/embed/RE4LC7e?postJsIIlMsg=true&autoCaptions=pt-pt>. Acedido em 30-junho-2021.
- [5] Wikipedia, "Bitcoin". Disponível em: <https://en.wikipedia.org/wiki/Bitcoin>. Acedido em 30-junho-2021.
- [6] Hyperledger Fabric, "How Fabric networks are structured" Disponível em: <https://hyperledger-fabric.readthedocs.io/en/latest/network/network.html>. Acedido em 30-junho-2021.
- [7] Canonical. "Multipass". Disponível em: <https://multipass.run/>. Acedido em 23-abril-2022
- [8] J. Fernandes, D. Raposo, N. Armando, S. Sinche, J. Sá Silva, A. Rodrigues, V. Pereira, H. Oliveira, L. Macedo and F. Boavida, "ISABELA – A Socially-Aware Human-in-the-Loop Advisor System", Journal Online Social Networks and Media, Elsevier, ISSN: 2468-6964, Jan. 2020.
- [9] Opencast, "Features". Disponível em: <https://opencast.org/features/>. Acedido em 30-junho-2021.
- [10] Opencast, "Integrations". Disponível em: <https://opencast.org/integrations/>. Acedido em 30-junho-2021.
- [11] Matomo, "What Is Matomo?". Disponível em: https://matomo.org/faq/new-to-piwik/faq_13/. Acedido em 30-junho-2021.
- [12] Matomo, "BigBlueButton". Disponível em: <https://docs.bigbluebutton.org/>. Acedido em 30-junho-2021.
- [13] ALEKS, "About ALEKS". Disponível em: https://www.aleks.com/about_aleks. Acedido em 30-junho-2021.
- [14] Kelly Parke, "Designing AI Solutions". Disponível em: <https://www.kellyeparke.com/can-you-really-find-friends-on-the-web/>. Acedido em 30-junho-2021.
- [15] STHEMBrasil, "Quem somos?". Disponível em: <https://www.sthembrasil.com/quem-somos/>. Acedido em 30-junho-2021.
- [16] STHEMBrasil, "Projeto-piloto do STHEM Brasil para o uso da ALEKS é aprovado e elogiado por instituições consorciadas". Disponível em: <https://www.sthembrasil.com/projeto-piloto-do-sthem-brasil-para-o-uso-da-aleks-e-aprovado-e-elogiado-por-instituicoes-consorciadas/>. Acedido em 30-junho-2021.
- [17] Microsoft, "Criar aplicações móveis com o Xamarin.Forms". Disponível em: <https://docs.microsoft.com/pt-pt/learn/paths/build-mobile-apps-with-xamarin-forms/>. Acedido em 30-junho-2021.
- [18] OsBoxes, "Linux Mint". Disponível em: <https://www.osboxes.org/linux-mint/#linux-mint-20-3-vmware>. Acedido em 10-março-2022.
- [19] Canonical, "Multipass". Disponível em: <https://multipass.run/>. Acedido em 23-março-2022.
- [20] Hyperledger Fabric, "Install Samples, Binaries, and Docker Images". Disponível em: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/install.html>. Acedido em 23-março-2022.
- [21] Go, "Download and install". Disponível em: <https://go.dev/doc/install>. Acedido em 23-março-2022.
- [22] Karl Broman, "Start a new git repository". Disponível em: https://kbroman.org/github_tutorial/pages/init.html. Acedido em 23-abril-2022.
- [23] GitHub, "GitHub Desktop". Disponível em: <https://desktop.github.com/>. Acedido em 23-abril-2022.

- [24] Github Docs, “Gerar uma nova chave SSH e adicioná-la ao ssh-agent”. Disponível em: <https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>. Acedido em 23-abril-2022.
- [25] Github Docs, “Adicionar uma nova chave SSH à sua conta do GitHub”. Disponível em: <https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>. Acedido em 23-abril-2022.
- [26] Hyperledger-Fabric, “cryptogen”. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/latest/commands/cryptogen.html>. Acedido em 30-junho-2021.
- [27] Hyperledger-Fabric, “configtxgen”. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/latest/commands/configtxgen.html>. Acedido em 30-junho-2021.
- [28] Hyperledger-Fabric, “peer”. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/latest/commands/peercommand.html>. Acedido em 30-junho-2021.
- [29] Hyperledger-Fabric, “Membership Service Provider (MSP)”. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/membership/membership.html>. Acedido em 10-junho-2022.
- [30] KC Tam, “Add a New Organization on Existing Hyperledger Fabric Network”. Disponível em: <https://kctheservant.medium.com/add-a-new-organization-on-existing-hyperledger-fabric-network-2c9e303955b2>. Acedido em 15-abril-2022.
- [31] KC Tam, “Multi-Host Deployment for First Network (Hyperledger Fabric v2)”. Disponível em: <https://kctheservant.medium.com/multi-host-deployment-for-first-network-hyperledger-fabric-v2-273b794ff3d>. Acedido em 28-abril-2022.
- [32] Go Ethereum, “Private Network Tutorial”. Disponível em: <https://geth.ethereum.org/docs/getting-started/private-net>. Acedido em 2-julho-2022.
- [33] Go Ethereum, “Installing Geth”. Disponível em: <https://geth.ethereum.org/docs/install-and-build/installing-geth>. Acedido em 2-julho-2022.