
Hyperledger Fabric

Deusimar Ferreira

<http://tiny.cc/3p80lz>

Versão 1.1 em 16 de agosto de 2020

Agradecimentos

Agradeço ao mestre Fernando Anselmo[12, 11, 13] por sempre compartilhar seus conhecimentos e assim motivar outras pessoas a fazerem o mesmo, e também por tornar público seus scripts LaTeX[10], confesso que nunca tinha se quer ouvido falar sobre e agora sou totalmente adepto.

Resumo

Neste artigo vou falar um pouco sobre essa tecnologia que vem cada vez mais ganhando atenção mundo a fora, atraindo olhares das mais renomadas empresas de tecnologia e também de desenvolvedores por toda parte do globo terrestre.

1 Blockchain

Blockchain é um livro-razão (ledger) de transações imutáveis, mantido dentro de uma rede distribuída de nós pares (peers). Esses nós mantêm uma cópia do livro (ledger), aplicando transações que foram validadas por um protocolo de consenso, agrupadas em blocos que incluem um hash que liga cada bloco ao bloco anterior.

Na Figura 1, faço uma analogia com um trem para facilitar o entendimento de uma cadeia de blocos. Primeiro teremos a cabine do trem, ela será nosso bloco genesis, agora perceba que a cada estação em que o trem vai passando é registrado e adicionado um novo vagão. Cada vagão adicionado será de forma analoga um novo bloco e cada estação será a conclusão de um trabalho e o registro no livro-razão (ledger) e assim vai formando uma cadeia onde todos os vagões ou blocos estarão interligados.



Figura 1: Exemplo de uma cadeia de blocos

1.1 Origen do Blockchain

Apesar de ter seu surgimento vinculado ao do Bitcoin[6] em 2008, o conceito teve origem bem antes, nos anos 90 em trabalho realizado por Stuart Haber[4] e W. Scott Stornetta[?] chamado *How to Time-Stamp a Digital Document*[5] .

1.2 Algoritmos de Consenso

Um algoritmo de consenso é um mecanismo que visa garantir que os participantes de uma rede cheguem em um acordo sobre uma única fonte de verdade, mesmo que ocorram falhas em alguns dos nós participantes.

Existem diferentes tipos de algoritmos de consenso, falarei mais sobre eles no futuro, porém aqui irei citar apenas os três mais conhecidos em minha opinião, que são:

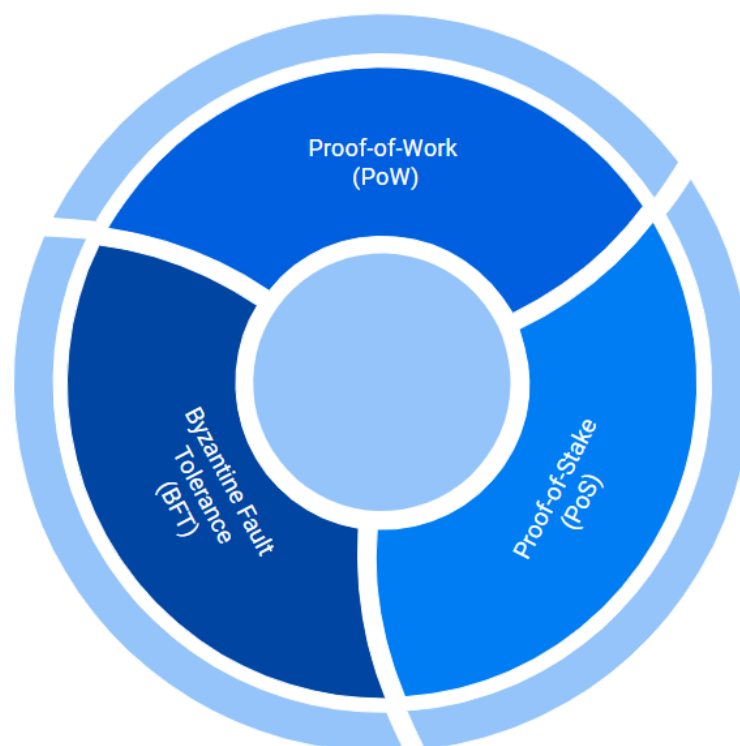


Figura 2: Os três algoritmos de consenso mais conhecidos

2 Hyperledger

Hyperledger[2] é uma comunidade de código aberto "estabelecida sob a Linux Foundation[1], que tem uma história longa e muito bem-sucedida de nutrir projetos de código aberto sob uma governança aberta que desenvolve comunidades sustentáveis e ecossistemas prósperos" focada no desenvolvimento de um conjunto de estruturas, ferramentas e bibliotecas para implantação de blockchain em nível corporativo.

2.1 Hyperledger Fabric

Hyperledger Fabric[3] é uma plataforma do tipo DLT (Distributed Ledger Technologies) de código aberto de nível empresarial, projetada para uso corporativo desde o início.



Figura 3: Principais requisitos para uso corporativo

2.2 Criando sua primeira rede Blockchain

Depois de contextualizarmos sobre o universo Blockchain é chega a hora de partimos para prática.

```
1 | # Oba!! Prática!!  
2 | $ echo "Vamos nessa!"
```

A topologia de nossa rede ficará assim:

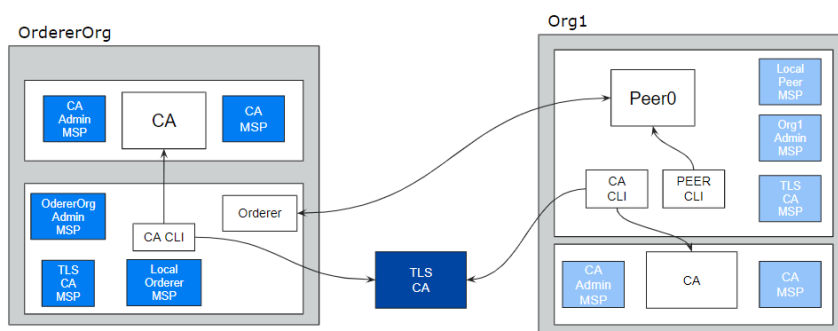


Figura 4: Topologia da nossa primeira rede

2.2.1 Pré-requisitos

Para facilitar, disponibilizei os comandos necessários para configurar o ambiente de desenvolvimento no meu GitHub[7], mas recomendo fortemente que acesse pré-requisitos[8] para se familiarizar com os requisitos básicos para rodar uma rede Hyperledger Fabric.

```
1 || Notas: Este post baseia-se em ambiente Linux (Ubuntu 18.04).
```

2.2.2 Definindo diretório de trabalho

Vale lembrar que os diretórios utilizados para armazenar os binários e arquivos de configuração da rede a seguir não é uma imposição, você pode definir outros caminhos caso deseje, porém será necessário alterar os comandos aqui demonstrados.

```
1 | $ sudo mkdir -p /opt/blockchain/hyperledger/  
2 | $ sudo chown -R aluno:aluno /opt/blockchain  
3 |  
4 | # Certifique que esteja no diretório correto  
5 | $ cd /opt/blockchain/hyperledger/
```

2.2.3 Baixando Imagens Hyperledger Fabric

Vamos usar os comandos abaixo para baixar as imagens docker[9] e os binários necessários para criar a cadeia de certificados e os demais artefatos necessários para configurar a rede Hyperledger[2] Fabric[3].

```
1 | $ curl -sSL http://bit.ly/2ysb0FE | bash -s 1.4.0  
2 |  
3 | # Após conclusão será criado diretório fabric-samples  
4 | $ tree -L 2
```

```
aluno@ip-172-31-3-84:/opt/blockchain/hyperledger$ tree -L 2
.
├── fabric-samples
│   ├── balance-transfer
│   ├── basic-network
│   ├── bin
│   ├── chaincode
│   ├── chaincode-docker-devmode
│   ├── CODE_OF_CONDUCT.md
│   ├── commercial-paper
│   ├── config
│   ├── CONTRIBUTING.md
│   ├── fabcar
│   ├── first-network
│   ├── high-throughput
│   ├── interest_rate_swaps
│   ├── Jenkinsfile
│   ├── LICENSE
│   ├── MAINTAINERS.md
│   ├── README.md
│   └── scripts
└── 13 directories, 6 files
```

Figura 5: Estrutura do diretório fabric-samples

2.2.4 Entendendo os binários Fabric[3]

Agora vamos entender um pouco dos binários Fabric[3] enquanto fazemos a geração dos artefatos para nossa rede.

```
1 # Clonando arquivos de configuracoes
2 $ git clone -b feature/lab-1 https://github.com/deusimarferreira/hyperledger-fabric.git
   network
3
4 # Acessando diretório dos arquivos
5 $ cd network
6
7 # Variáveis de ambiente
8 $ export PATH=$GOPATH/src/github.com/hyperledger/fabric/build/bin:${PWD}/../fabric-
   samples/bin:${PWD}:${PATH}
9 $ export FABRIC_CFG_PATH=${PWD}
10 $ export CHANNEL_NAME=villalabs-channel
```

O primeiro binário que iremos usar será o *cryptogen*, ele é responsável por gerar a cadeia de certificados Hyperledger[2] Fabrica[3] CA (Certificate Authority). Esse binário depende das configurações contidas no arquivo *crypto-config.yaml* e, após execução gera os artefatos no diretório *crypto-config*.

```
1 # Crie os diretórios crypto-config e channel-artifacts
2 $ mkdir crypto-config channel-artifacts
3
4 # Agora vamos gerar a cadeia de certificados para CA
5 $ cryptogen generate --config=./crypto-config.yaml
```

O segundo binário que iremos usar será o *configtxgen*, ele é responsável por gerar os artefatos de pré-configuração da rede Hyperledger[2] Fabrica[3]. Esse binário depende das configurações contidas no arquivo *configtx.yaml* e, após execução gerar os artefatos de configuração do canal no diretório *channel-artifacts*.

```

1 | # Gera o bloco genesis
2 | $ configtxgen -profile OrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block
3 |
4 | #
5 | $ configtxgen -profile OrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx
   | -channelID $CHANNEL_NAME
6 |
7 | #
8 | $ configtxgen -profile OrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/
   | Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP

```

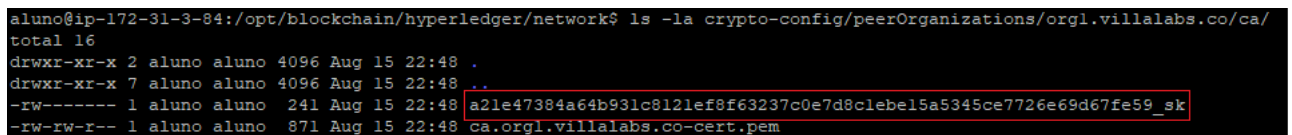
2.2.5 Iniciando nossa rede com docker-compose

Agora estamos perto de finalizar a criação de nossa rede, mas antes de executar o *docker-compose* precisamos atualizar a variável de ambiente *FABRIC_CA_SERVER_CA_KEYFILE* para a nova chave criada quando executamos o comando *cryptogen generate --config=./crypto-config.yaml*.

```

1 | # Comando para identificar a chave
2 | $ ls -la crypto-config/peerOrganizations/org1.villalabs.co/ca/

```



```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ ls -la crypto-config/peerOrganizations/org1.villalabs.co/ca/
total 16
drwxr-xr-x 2 aluno aluno 4096 Aug 15 22:48 .
drwxr-xr-x 7 aluno aluno 4096 Aug 15 22:48 ..
-rw----- 1 aluno aluno 241 Aug 15 22:48 a21e47384a64b931c8121ef8f63237c0e7d8c1eb15a5345ce7726e69d67fe59_sk
-rw-rw-r-- 1 aluno aluno 871 Aug 15 22:48 ca.org1.villalabs.co-cert.pem

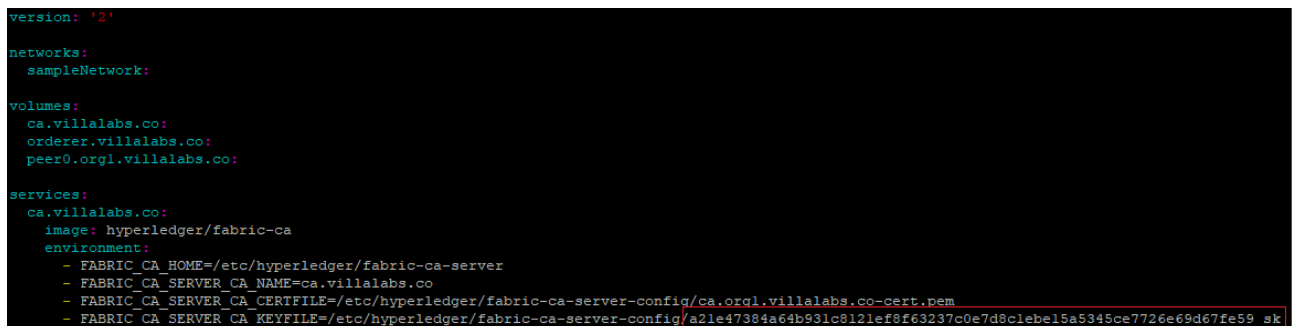
```

Figura 6: Saída do comando *ls -la*

```

1 | # Comando para alterar o docker-compose e incluir a chave
2 | $ vim docker-compose.yaml

```



```

version: '2'

networks:
  sampleNetwork:

volumes:
  ca.villalabs.co:
  orderer.villalabs.co:
  peer0.org1.villalabs.co:

services:
  ca.villalabs.co:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.villalabs.co
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.villalabs.co-cert.pem
      - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/a21e47384a64b931c8121ef8f63237c0e7d8c1eb15a5345ce7726e69d67fe59_sk

```

Figura 7: *docker-compose.yaml*

```

1 # Para e limpa todos os containers existentes
2 $ docker-compose -f docker-compose.yaml down
3
4 # Sobre containers
5 $ docker-compose -f docker-compose.yaml up -d

```

Pronto! Após executar o docker-compose, em caso de sucesso veremos o seguinte resultado.

```

1 # Execute esse comando para verificar os contêineres criados
2 $ docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"

```

CONTAINER ID	IMAGE	NAMES	STATUS	PORTS
7eb375f5bf7c	hyperledger/fabric-tools	cli	Up 12 minutes	
4784c382b498	hyperledger/fabric-peer	peer0.org1.villalabs.co	Up 12 minutes	0.0.0.0:7051->7051/tcp
87ac82e0f66b	hyperledger/fabric-ca	ca.villalabs.co	Up 12 minutes	0.0.0.0:7054->7054/tcp
606bf5dc576c	hyperledger/fabric-orderer	orderer.villalabs.co	Up 12 minutes	0.0.0.0:7050->7050/tcp

Figura 8: Contêineres criados

2.2.6 Criando canal (channel)

O terceiro binário tratado até aqui é *peer*, ele é utilizado para administrar o parceiro/par (peer), como cria um canal (channel), implantar um chaincode e etc..

```

1 # Cria o canal (channel)
2 $ docker exec \
3     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
5     peer0.org1.villalabs.co \
6     peer channel create -o orderer.villalabs.co:7050 -c $CHANNEL_NAME -f /etc/hyperledger
7     /configtx/channel.tx

```

Saída do comando:

```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
> -e "CORE_PEER_LOCALMSPID=Org1MSP" \
> -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer0.org1.villalabs.co \
> peer channel create -o orderer.villalabs.co:7050 -c $CHANNEL_NAME -f /etc/hyperledger/configtx/channel.tx
2020-08-15 23:58:25.678 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2020-08-15 23:58:25.721 UTC [cli.common] readBlock -> INFO 002 Received block: 0

```

Figura 9: Canal criado

```

1 # Uni o novo canal (channel) criado
2 $ docker exec \
3     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
5     peer0.org1.villalabs.co \
6     peer channel join -b $CHANNEL_NAME.block

```

Saída do comando:

```
aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
> -e "CORE_PEER_LOCALMSPID=Org1MSP" \
> -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer0.org1.villalabs.co \
> peer channel join -b $CHANNEL_NAME.block
2020-08-15 23:58:46.581 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2020-08-15 23:58:46.624 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
```

Figura 10: *Unindo canal*

Que tal usar o comando `peer channel list` para verificar o canal recém criado.

```
1 | # Verificar o canal criado
2 | $ docker exec \
3 |     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4 |     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
5 |     peer0.org1.villalabs.co \
    peer channel list
```

Saída do comando:

```
aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
> -e "CORE_PEER_LOCALMSPID=Org1MSP" \
> -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer0.org1.villalabs.co \
> peer channel list
2020-08-15 23:58:59.024 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
villalabs-channel
```

Figura 11: *Canal criado*

2.3 Vamos adicionar um novo *peer*

Até aqui construímos uma rede simples com apenas uma organização e um *peer*, mas quando falamos em rede distribuída pensamos em algo maior com várias organizações e *peer*'s. O que tenho a dizer por enquanto é “muita hora nessa calma, rss”, chegaremos lá.

Nosso próximo passo será adicionar um novo *peer* em nossa *Org1*.

Altere o arquivo `docker-compose.yaml` para adicionar as configurações necessárias para criar o contêiner para o novo *peer*.

Serão duas alterações necessárias, a primeira nas definições de **volumes:** e a segunda nos **services:**.

volumes: - Em volumes adicione o volume para o *peer1* (*peer1.org1.villalabs.co*):

```
1 | volumes:
2 |   ca.villalabs.co:
3 |   orderer.villalabs.co:
4 |   peer0.org1.villalabs.co:
5 |   peer1.org1.villalabs.co:
```

services: - Nos services vamos adicionar as configuração abaixo do **peer0.org1.villalabs.co**:


```

1  peer1.org1.villalabs.co:
2    container_name: peer1.org1.villalabs.co
3    image: hyperledger/fabric-peer
4    environment:
5      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
6      - CORE_PEER_ID=peer1.org1.villalabs.co
7      - CORE_LOGGING_PEER=info
8      - CORE_CHAINCODE_LOGGING=debug
9      - CORE_PEER_LOCALMSPID=Org1MSP
10     - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
11     - CORE_PEER_ADDRESS=peer1.org1.villalabs.co:7051
12     - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=network_sampleNetwork
13  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
14  command: peer node start
15  ports:
16    - 8051:7051
17    - 8053:7053
18  volumes:
19    - /var/run:/host/var/run/
20    - ./crypto-config/peerOrganizations/org1.villalabs.co/peers/peer1.org1.villalabs.co
21    /msp:/etc/hyperledger/msp/peer
22    - ./crypto-config/peerOrganizations/org1.villalabs.co/users:/etc/hyperledger/msp/
23    users
24    - ./channel-artifacts:/etc/hyperledger/configtx
25  depends_on:
26    - orderer.villalabs.co
27  networks:
28    - sampleNetwork

```

Altere também o arquivo *crypto-config.yaml* para aumentar o números de *peer*'s.

```

1  Template:
2  Count: 2

```

Usando o binário *cryptogen* para estender nossa rede e adicionar o novo *peer*.

```

1  # Realiza a geração da cadeia de certificados para o novo peer
2  $ cryptogen extend --config=./crypto-config.yaml
3
4  # Verifique novo peer adicionado
5  $ ls -l ./crypto-config/peerOrganizations/org1.villalabs.co/peers/

```

Saída do comando:

```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ ls -l ./crypto-config/peerOrganizations/org1.villalabs.co/peers/
total 8
drwxr-xr-x 4 aluno aluno 4096 Aug 15 22:48 peer0.org1.villalabs.co
drwxr-xr-x 4 aluno aluno 4096 Aug 16 15:30 peer1.org1.villalabs.co

```

Figura 12: *Peer's existentes*

Usando o binário *configtxgen* para atualizar nossa rede e permitir adicionar o *peer1*.

```

1 | # Vamos gerar nosso genesis block
2 | $ configtxgen -profile OrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block

```

Tudo pronto, vamos iniciar os contêineres:

```

1 | # Vamos iniciar os containers
2 | $ docker-compose -f docker-compose.yaml up \
3 |   -d peer0.org1.villalabs.co peer1.org1.villalabs.co cli
4 |
5 | # Visualize os contêineres com filtro 'name=peer'
6 | $ docker ps --filter name=peer --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"

```

Saída do comando:

```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker ps --filter name=peer -
CONTAINER ID          IMAGE                  NAMES                  STATUS
ae5ae9001616         hyperledger/fabric-peer peer1.org1.villalabs.co Up 6 seconds
7c7e27dc2ea2         hyperledger/fabric-peer peer0.org1.villalabs.co Up About an hour

```

Figura 13: Contêineres peer's

Agora vamos unir o canal criado no item **2.2 Criando sua primeira rede Blockchain** ao novo *peer* que acabamos de adicionar.

```

1 | # Primeiro vamos fazer um fetch do genesis block para o corrente channel
2 | $ docker exec \
3 |   -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4 |   -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
   peer1.org1.villalabs.co \
5 |   peer channel fetch oldest $CHANNEL_NAME.block -c $CHANNEL_NAME \
6 |   --orderer orderer.villalabs.co:7050

```

Saída do comando:

```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
>   -e "CORE_PEER_LOCALMSPID=Org1MSP" \
>   -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer1.org1.villalabs.co
>   peer channel fetch oldest $CHANNEL_NAME.block -c $CHANNEL_NAME \
>   --orderer orderer.villalabs.co:7050
2020-08-16 15:45:46.044 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2020-08-16 15:45:46.046 UTC [cli.common] readBlock -> INFO 002 Received block: 0

```

Figura 14: Saída do comando peer channel fetch

```

1 | # Segundo, vamos executar um join
2 | $ docker exec \
3 |   -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4 |   -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
   peer1.org1.villalabs.co \

```

```
5 || peer channel join -b $CHANNEL_NAME.block
```

Saída do comando:

```
aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
> -e "CORE_PEER_LOCALMSPID=Org1MSP" \
> -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer1.org1.villalabs.co \
> peer channel join -b $CHANNEL_NAME.block
2020-08-16 15:46:59.267 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2020-08-16 15:46:59.314 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
```

Figura 15: Saída do comando *peer channel join*

Que tal usar o comando *peer channel list* para verificar o canal recém criado.

```
1 # Verificar o canal
2 $ docker exec \
3   -e "CORE_PEER_LOCALMSPID=Org1MSP" \
4   -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
5   peer1.org1.villalabs.co \
   peer channel list
```

Saída do comando:

```
aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker exec \
> -e "CORE_PEER_LOCALMSPID=Org1MSP" \
> -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp" peer1.org1.villalabs.co \
> peer channel list
2020-08-16 15:49:15.947 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
villalabs-channel
```

Figura 16: Saída do comando *peer channel list*

2.4 Usando CouchDB como banco de estado

Por padrão o Hyperledger[2] Fabric[3] usa como banco de dados de estado o LevelDB, nesta seção iremos apresentar como configurar o CouchDB como nosso banco de dados de estado nos *peer's* de nossa rede.

Todo *peer* na rede deve possuir uma cópia do *ledger* e um *state database* próprio. Novamente vamos alterar o arquivo *docker-compose.yml* (O arquivo completo pode ser encontrado no endereço <https://github.com/deusimarferreira/hyperledger-fabric/blob/feature/lab-3/docker-compose.yml>) para alterar e incluir as configurações necessárias.

Adicione as variáveis de ambiente nas configurações dos *peer's* (*peer0* e *peer1*).

```
1 # Peer0
2 CORE_LEDGER_STATE_STATEDATABASE=CouchDB
3 CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb.peer0.org1.villalabs.co:5984
4 CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=peer0.org1
5 CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=password
6
7 # Peer1
```

```

8 CORE_LEDGER_STATE_STATEDATABASE=CouchDB
9 CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb.peer1.org1.villalabs.co:5984
10 CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=peer1.org1
11 CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=password

```

services: - Nos services vamos adicionar as configuração dos dois bancos de estado:

```

1 couchdb.peer0.org1.villalabs.co:
2   container_name: couchdb.peer0.org1.villalabs.co
3   environment:
4     - COUCHDB_USER=peer0.org1
5     - COUCHDB_PASSWORD=password
6   image: hyperledger/fabric-couchdb
7   ports:
8     - 5984:5984
9   networks:
10    - sampleNetwork
11
12 couchdb.peer1.org1.villalabs.co:
13   container_name: couchdb.peer1.org1.villalabs.co
14   environment:
15     - COUCHDB_USER=peer1.org1
16     - COUCHDB_PASSWORD=password
17   image: hyperledger/fabric-couchdb
18   ports:
19     - 6984:5984
20   networks:
21    - sampleNetwork

```

Tudo pronto, vamos iniciar os contêineres:

```

1 $ docker-compose -f docker-compose.yaml up \
2   -d ca.villalabs.co orderer.villalabs.co \
3   couchdb.peer0.org1.villalabs.co peer0.org1.villalabs.co \
4   couchdb.peer1.org1.villalabs.co peer1.org1.villalabs.co \
5   cli
6
7 # Visualize os contêineres
8 $ docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"

```

Saída do comando:

```

aluno@ip-172-31-3-84:/opt/blockchain/hyperledger/network$ docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"

```

CONTAINER ID	IMAGE	NAMES	STATUS
66ca3476c8e4	hyperledger/fabric-tools	cli	Up 47 seconds
f57d282cdcc6	hyperledger/fabric-peer	peer0.org1.villalabs.co	Up 48 seconds
77270a71e48b	hyperledger/fabric-peer	peer1.org1.villalabs.co	Up 48 seconds
f5440fd4a22a	hyperledger/fabric-couchdb	couchdb.peer0.org1.villalabs.co	Up 47 seconds
f8d6db55a341	hyperledger/fabric-couchdb	couchdb.peer1.org1.villalabs.co	Up 48 seconds
2da553ff5d84	hyperledger/fabric-ca	ca.villalabs.co	Up 2 hours
2a9e040769b7	hyperledger/fabric-orderer	orderer.villalabs.co	Up 2 hours

Figura 17: Contêineres

Use o comando *curl* para testar as bases CouchDB:

```
1 # Testar CouchDB
2 $ curl http://localhost:5984
3 $ curl http://localhost:6984
```

Como mudamos de base de dados iremos precisar atualizar os *peer*'s:

```
1 # PEER0
2 # Primeiro vamos fazer um fetch do genesis block para o corrente channel
3 $ docker exec \
4     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
5     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
6     peer0.org1.villalabs.co \
7     peer channel fetch oldest $CHANNEL_NAME.block -c $CHANNEL_NAME \
8     --orderer orderer.villalabs.co:7050
9
10 # Segundo, vamos executar um join
11 $ docker exec \
12     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
13     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
14     peer0.org1.villalabs.co \
15     peer channel join -b $CHANNEL_NAME.block
16
17 # PEER1
18 # Primeiro vamos fazer um fetch do genesis block para o corrente channel
19 $ docker exec \
20     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
21     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
22     peer1.org1.villalabs.co \
23     peer channel fetch oldest $CHANNEL_NAME.block -c $CHANNEL_NAME \
24     --orderer orderer.villalabs.co:7050
25
26 # Segundo, vamos executar um join
27 $ docker exec \
28     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
29     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
30     peer1.org1.villalabs.co \
31     peer channel join -b $CHANNEL_NAME.block
```

Que tal usar o comando *peer channel list* para verificar o canal recém criado.

```
1 # PEER0
2 # Verificar o canal
3 $ docker exec \
4     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
5     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
6     peer0.org1.villalabs.co \
7     peer channel list
8
9 # PEER1
10 # Verificar o canal
11 $ docker exec \
12     -e "CORE_PEER_LOCALMSPID=Org1MSP" \
13     -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.villalabs.co/msp"
```

3 Conclusão

Ne artigo tivemos a oportunidade de conhecer um pouco mais sobre a tecnologia Blockchain e ainda criar uma rede na prática usando a ferramenta Hyperledger[2] Fabric[3].

4 Sobre o autor

Praticante dos princípios ágeis para desenvolvimento de software, focado na entrega de valor.

Adepto do software **Open Source** (Software Livre), estou sempre em busca de novos conhecimentos e desafios relacionados ao desenvolvimento de software que agregam valor tanto no âmbito pessoal quanto profissional.

“Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade. Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser.” por Fernando Anselmo[12]

Referências

- [1] The Linux Foundation
<https://www.linuxfoundation.org/>
- [2] Hyperledger
<https://hyperledger.org/>
- [3] Hyperledger Fabric
<https://www.hyperledger.org/use/fabric>
- [4] Stuart Haber <https://www.linkedin.com/in/stuart-haber/>)\bibitem{scottstornetta}W. ScottStornetta\\url<https://www.linkedin.com/in/w-scott-stornetta-25186120/>)
- [5] How to Time-Stamp a Digital Document
https://link.springer.com/chapter/10.1007%2F3-540-38424-3_32
- [6] Bitcoin White paper
<https://bitcoin.org/en/bitcoin-paper>
- [7] Configurações de Ambiente
<https://github.com/deusimarferreira/hyperledger-fabric/blob/master/environment.md> <https://bitcoin.org/en/bitcoin-paper>

- [8] Pré-requisitos
<https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html>
- [9] Docker
<https://www.docker.com/>
- [10] LaTeX
<https://latex.net/>
- [11] Essa apostila utiliza scripts LaTeX[10] disponibilizados por Fernando Anselmo[12]
<https://github.com/fernandoans/publicacoes>
- [12] Fernando Anselmo - Blog Oficial de Tecnologia
<http://www.fernandoanselmo.blogspot.com.br/>
- [13] Encontre publicações de Fernando Anselmo[12] em
<https://cetrex.academia.edu/FernandoAnselmo>
- [14] Encontre essa e outras publicações minhas em
<https://github.com/deusimarferreira/artigos>