



Enterprise Blockchains: Redes permissionadas

[illegible]

Ü Satoshi Nakamoto — Bitcoin: A Peer-to-Peer Electronic Cash System.

Ü 2009 <https://bitcoin.org/bitcoin.pdf>

Ü P2P.

Ü Transferência sem intermediários.

Ü Impedir “double spending”.

Ü Criptografia versus Confiança.

Ü Anonimato (sou um hash).

Ü Imutabilidade (somente escrita).

Ü Prova de Trabalho (Mineração).



Satoshi Nakamoto

- Ü O que é?
- Ü Como é “gerado”/emitido?
- Ü Por que vale tanto?
 - Ü 21.000.000 => deflacionário => expectativa.
 - Ü => guardar > gastar: amanhã vale mais.
 - Ü Fosse a única opção => equilíbrio possível.
 - Ü “Convertido” em moeda real, especular parece melhor.
 - Ü Ainda: armazenar valor do que usar para troca.



ü Valor => HASH é tranquilo.

ü Achar um HASH tal que comece com N zeros é o desafio.

[illegible]

- Todos podem ter a cadeia completa.
- Mineradores possuem a cadeia completa.
- Somente escrita => rastreabilidade de transações, mas não dos nomes, apenas hashes.
- Não existe DESFAZER.

- ü O maior bloco ganha.
- ü Auto tuning: 12,5 BTC / 10 min, depois de 210.000, aumentar ZEROS => 6,25 / 10 min (jul/20).
- ü Finito: 21.000.000 => depois: tarifar.
- ü Qualquer coisa como 2140.

- “World Computer”, o que é.
- Acoplar lógica ao Blockchain.
- DLT, porém manipulado por códigos.
- Smart Contracts => código para manipular o estado, e.g. motorista + carro + 1/2L vodka => carro não liga & SEGURO inválido.

[illegible]

[illegible]

Cadeia

Ü Bloco:

Ü Árvore de Mele.

Ü Hash dos
hashes.

Ü Bloco:

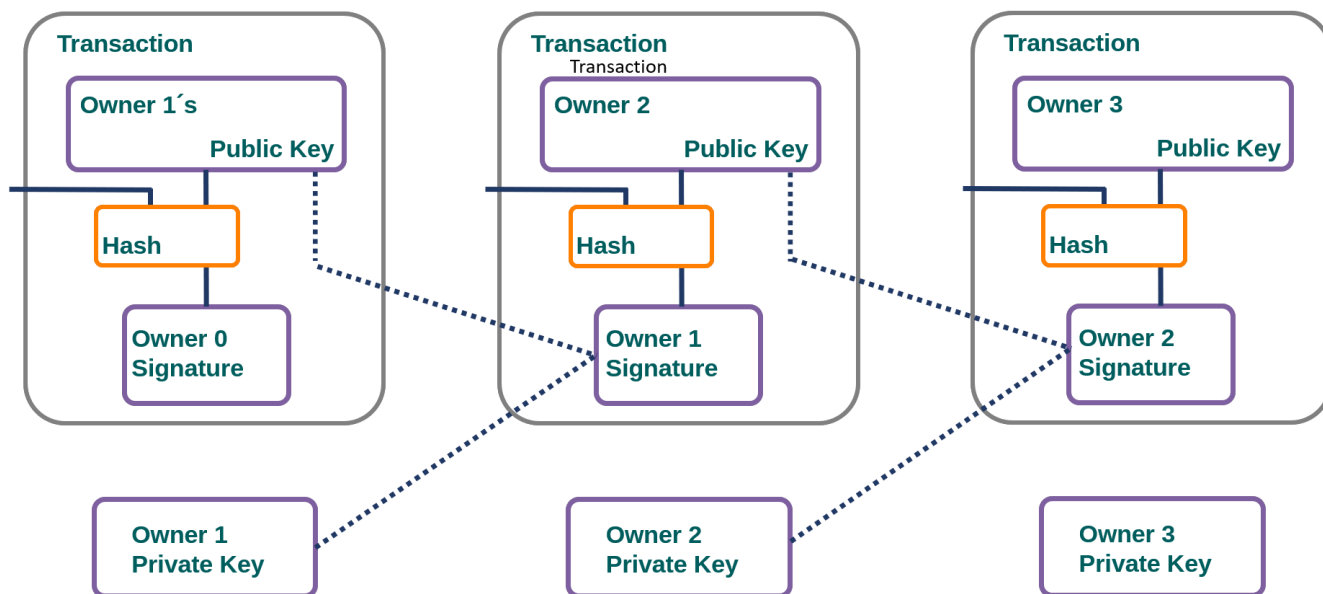
ü Árvore de Mele.

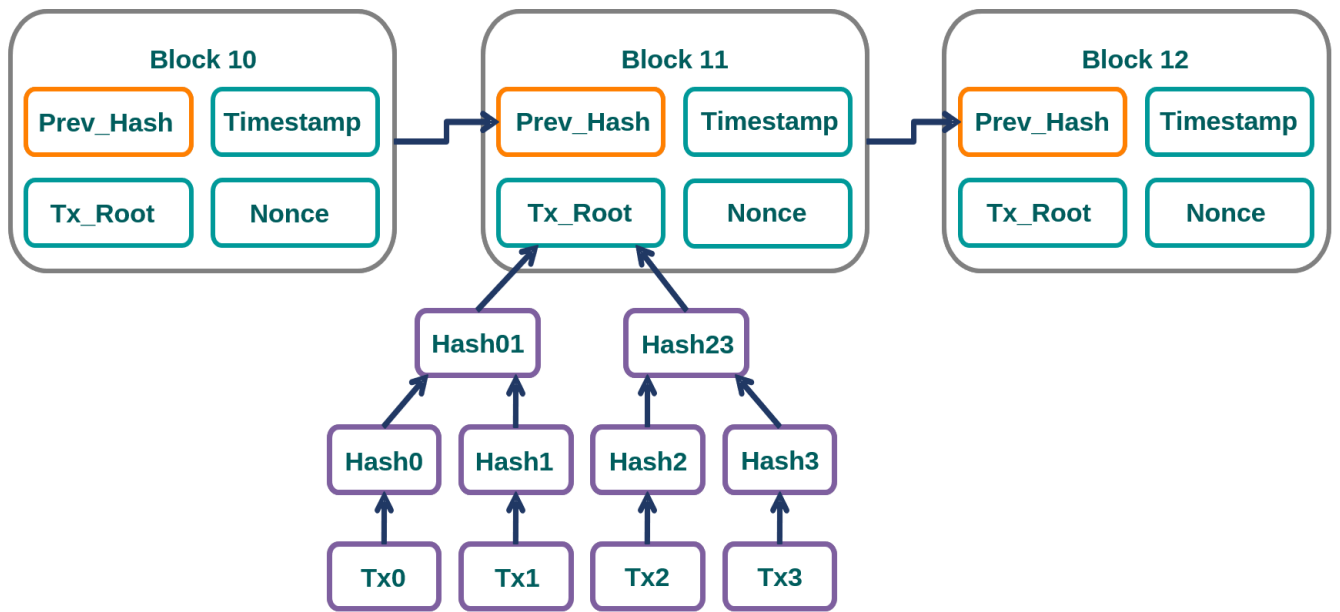
ü Hash dos hashes.

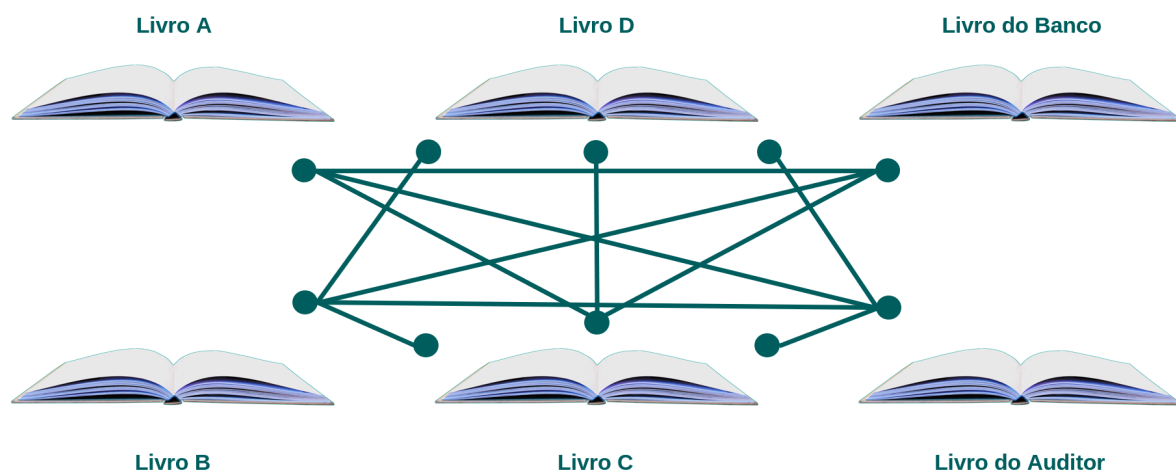
ü Hash do bloco anterior.

NONCES.

[illegible]







Ü Blockchain, os livros são cópias e o chaincode/smart contract é comum

Ü Sem Blockchain....

ü Cadeia de blocos:

Blockchain.

[illegible]



O que a tecnologia ajuda a resolver?

[illegible]

Ü Confiança sem confiança.

O que a tecnologia ajuda a resolver?

- Imutabilidade: livro compartilhado.
- Criptografia: validação, confiança, privacidade.
- Regras de negócios compartilhadas: smartContracts/Chaincode
- Consenso pós execução e no ciclo.

[illegible]

- Cadeia de suprimentos.
- Cadeia de donos: quem é o dono atual?
- O que aconteceu para ser o dono? Smart contract.
- O que aconteceu com algo que foi particionado?
- Qual o documento válido?
- Quais são os dados daquele cliente?

[illegible]

O que a tecnologia ajuda a resolver?

- ü Imagine um repositório git compartilhado entre Bancos.
- ü Imagine um repositório git compartilhado entre concorrentes.

[illegible]

[illegible]

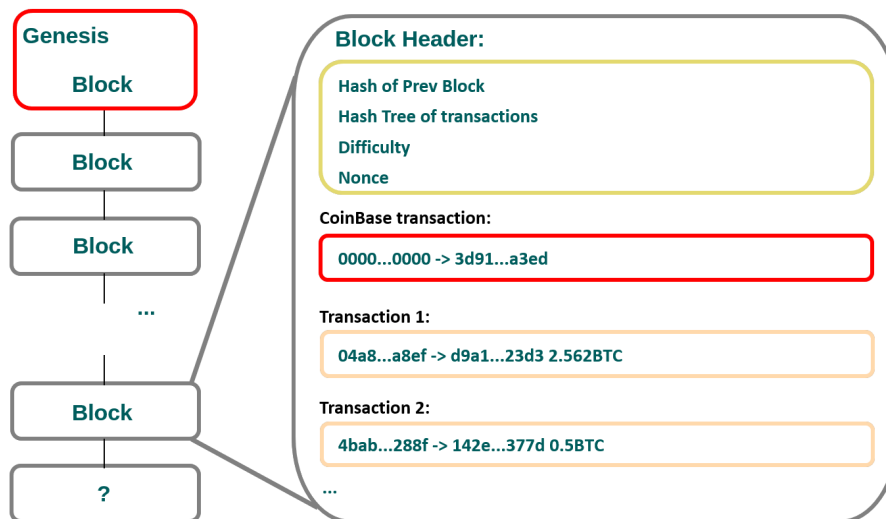
ü Smart Contract.

ü Código no bloco que modifica o bloco.

ü Consenso e conhecimento: Código compartilhado.

ü Primeiro cenário de git

Ü Prova de Trabalho.



- Redes Anônimas.
 - Bitcoin.
- Redes Permissionadas.
 - Hyperledger Fabric.
 - Consórcio.
 - Canais.



Projetos em Open Source

[illegible]

Hyperledger.

ü Fabric.

ü Sawtooth.

ü Iroha.

 Burrow.

ü Indy.

Ü Multichain.

ÜCorda.



Multichain

Multichain

[illegible]

Ü <https://www.multichain.com/download/multichain-latest.tar.gz>

Ü 123, Descompactar, criar cadeia, ligar serviço

Etapa (1) Primeira máquina

```
curl -OL https://www.multichain.com/download/multichain-latest.tar.gz
tar zxvf multichain-latest.tar.gz
multichain-1.0.6/multichain-util create chain1
multichain-1.0.6/multichaind chain1 -daemon
multichain-1.0.6/multichain-cli chain1
liststreams
listpermissions
create stream strTST false
create stream tstST false
listpermissions strTST.*
liststreams
publish strTST key1 vacamarela
publish strTST key1 48656c6c6f204d756c7469636861696e21
publish tstST key1 736f6d65206f7468657220646174615fd79f7
subscribe strTST
liststreamkeyitems strTST key1
issue hashpublishersacima cursos 10 0.1
listassets
getaddresses
###Vá para a segunda máquina Etapa (2)
Etapa(3)
multichain-1.0.6/multichain-cli chain1 grant 1GWAwdDXmn6i2vfBsTyXfHScJidBmo5R5XS6LM
connect,send,receive
```

###Vá para a segunda máquina Etapa (4) ligar o daemon multichain

Etapa (5)

```
##sendwithdata hashdasegundamáquina '{"cursos":.5}'
'{"for":"strtST","key":"transfer","data":"issuertxid"}'

sendwithdata 1GWAwdDXmn6i2vfBsTyXfHScJidBmo5R5XS6LM '{"cursos":.5}'
'{"for":"strtST","key":"transfer","data":"a8eca5d4fa1ee23d341cb35bbb1752621a7017c9c79b1c1a00c0f7
68b5d2a4e3"}'
```

gettotalbalances

###Vá para a segunda máquina Etapa (6) consultar

Repita a etapa 5 e a etapa 6

- Ü Criar um stream.
- Ü Verificar e atribuir permissões.
- Ü Transacionar chaves valores (metadados).

Na segunda máquina, Etapa (2)

```
curl -OL https://www.multichain.com/download/multichain-latest.tar.gz
tar zxvf multichain-latest.tar.gz
multichain-1.0.6/multichaind chain1@endereco:porta #para pegar o id
##
##MultiChain 1.0.6 Daemon (latest protocol 10011)
##
##Retrieving blockchain parameters from the seed node 192.168.56.102:8361 ...
##Blockchain successfully initialized.
```

```
##Please ask blockchain admin or user having activate permission to let you connect and/or transact:
##multichain-cli chain1 grant 1GWAwdDXmn6i2vfBsTyXfHScJidBmo5R5XS6LM connect
##multichain-cli chain1 grant 1GWAwdDXmn6i2vfBsTyXfHScJidBmo5R5XS6LM connect,send,receive
```

Execute na primeira máquina a última linha da mensagem acima: Etapa(3)

```
##multichain-cli chain1 grant 1GWAwdDXmn6i2vfBsTyXfHScJidBmo5R5XS6LM connect,send,receive
```

Etapa (4)

```
multichain-1.0.6/multichaind chain1 -daemon
```

Vá para a primeira máquina e transfira (sendwithdata) Etapa (5)

```
multichain-1.0.6/multichain-cli chain1
```

```
gettotalbalances
```

Repita a etapa 5 e a etapa 6

- Ü Criar assets (To issue).
- Ü Verificar e atribuir permissões.
- Ü Transacionar chaves valores (assets).

Criar e Listar Assets

```
multichain-1.0.6/multichain-cli chain1
chain1: assets (valores, tokens, criptomoedas)
```

Gere 10 cursos em posse de 1B, divisíveis em 10 cada um:

```
chain1: issue 1Bz2J2jR8b2i9cAja5LjNSWRhVbFNieTAti5No cursos 10 0.1
```

```
chain1: listassets
```

```
{"method":"listassets","params":[],"id":"12262334-1535122531","chain_name":"chain1"}

[
  {
    "name" : "cursos",
    "issuetxid" : "80556946bec20c28de647566344aaf5d6e15f84924f0eaf6afc9835d1ae35df4",
    "assetref" : "126-266-21888",
    "multiple" : 10,
    "units" : 0.100000000,
    "open" : false,
    "details" : {
    },
    "issueqty" : 10.000000000,
    "issueraw" : 100,
    "subscribed" : false
  }
]
```

```
chain1: gettotalbalances
```

```
{"method":"gettotalbalances","params":[],"id":"64945662-1535122551","chain_name":"chain1"}
```

```
[
  {
    "name" : "cursos",
    "assetref" : "126-266-21888",
    "qty" : 10.000000000
  }
]
```

Ü Transferir.

Segunda_{vm}

Segundo nó: tente conectar

```
multichain-1.0.6/multichaind chain1@192.168.56.101:6291&
```

Use as sugestões para ajustar as permissões no primeiro nó:

```
multichain-cli chain1
```

```
chain1: grant 15dmUQS353mMqEAUJkqq5n3JtfzczpReK2pXjk connect,send,receive
```

No primeiro nó

```
chain1: sendwithdata 15dmUQS353mMqEAUJkqq5n3JtfzczpReK2pXjk '{"cursos":.5}'
'{"for":"tstST","key":"transfer","data":"e9206a7573746f206d65746164696e6861"}'
```

Teste o segundo nó de novo:

```
multichain-1.0.6/multichaind chain1@192.168.56.101:6291&
```

Ainda no segundo nó:

```
multichain-1.0.6/multichain-cli chain1
```

```
chain1: gettotalbalances
```

```
chain1: listwallettransactions
```

```
chain1: sendwithdata 1Bz2J2jR8b2i9cAja5LjNSWRhVbFNieTAti5No '{"cursos":.5}'
'{"for":"tstST","key":"transfer","data":"6d65746164696e68612c206d7569746f20706f75636f"}'
```

...

```
This wallet contains no addresses with permission to write to this stream and global send
permission.
```

No primeiro nó

```
chain1: grant 15dmUQS353mMqEAUJkqq5n3JtfzczpReK2pXjk send
```

```
chain1: grant 15dmUQS353mMqEAUJkqq5n3JtfzczpReK2pXjk tstST.write
```

Ü Listar os blocos.

Nas dias VMs

chain1: listblocks 0-100



Corda

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- Criado do zero.
- Linguagem Kotlin, JVM.
- CordAPP.
- Privacidade, meio financeiro.
- Integração com ecossistema Java/JVM.
- Forte uso de gradle.

[illegible]

hyperledger.org não é só Hyperledger Fabric.

• Projetos com público alvo para aplicação e desenvolvimento distintos.

ü Projetos comuns de integração.

ü Sawtooth.

Ü Iroha.

 Burrow.

ü Indy.

ü Fabric.

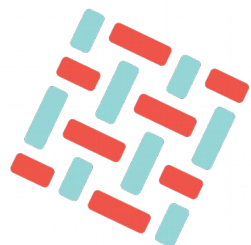
Ü Explorer.

Ü Composer.

Ü Cello.

_____ <https://lists.hyperledger.org/g/composer/message/125>

[illegible]



HYPERLEDGER
FABRIC

- Desenvolvido em Go: rede, concorrência.
- Segurança: TLS + Autenticação Certificados + Papéis.
- Modular.
- Permissionado.
- Canais e dados privados.
- Chaincodes (Smart Contracts).

- Ü Inicialmente, projeto IBM.
- Ü Local, AWS, Azure, Oracle, IBM.
- Ü Chaincode = Smart Contract: go, nodejs, java.
- Ü SDK para nodejs e java (python, go, rest disponíveis).
- Ü Baseado em docker (não suportado fora de docker).
- Ü POC, em bancos públicos e privados no Brasil.



Instalação Hyperledger Fabric

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Ü Execução suportada: contêineres Docker.

Ü Instalação Docker.

Centos 7

```
sudo yum install -y git java-headless yum-utils device-mapper-persistent-data lvm2
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install -y docker-ce
sudo systemctl start docker
sudo gpasswd -a aluno docker
sudo systemctl enable docker
#logout login
docker run hello-world
```

Ubuntu 16.04

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo adduser aluno docker
#logout e login
docker run hello-world
```

Debian 9

```
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo adduser diurno docker
```

Encerre a sessão gráfica e logue novamente

```
docker run hello-world
```

• Execução suportada: contêineres Docker.

• Instalação docker-compose.

```
curl -OL https://github.com/docker/compose/releases/download/1.22.0/docker-compose-Linux-x86_64
sudo mv docker-compose-Linux-x86_64 /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

ÜChaincode Nodejs.

ÜInstalação
nodejs.

Hyperledger Fabric nodejs via node version management tool

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

Instale a versão LTS de nodejs

```
nvm install --lts
```

Ajuste o nvm para usar a versão LTS

```
nvm use --lts
```

```
nvm alias default 'lts/*'
```

Instale a última versão de npm (instalador de módulos)

```
npm install npm@latest -g
```

Confirme as versões instaladas:

```
node --version
```

v8.11.4

```
npm --version
```

6.4.1

Ü Chaincode Go

Ü Instalação Go

```
curl -OL https://dl.google.com/go/go1.11.linux-amd64.tar.gz
sudo tar zxvf go1.11.linux-amd64.tar.gz -C /usr/local/
echo 'export PATH=/usr/local/go/bin:$PATH' >> ~/.bashrc
mkdir ~/go/src -p
mkdir ~/go/bin -p
echo 'export GOPATH=$HOME/go' >> ~/.bashrc
#logout e login
```

```
#Teste
cd ~/go/src
Crie um arquivo com o conteúdo abaixo e nome hw.go:
package main
import "fmt"
func main() {
    fmt.Printf("hello, world\n")
}
```

Execute diretamente
go run hw.go
Ou compile e execute:
go build hw.go
./hw

• Execução de:

• fabric-samples/first-network

• first-network

• fabcar

• fabric/examples/e2ecli

Download do script de instalação

```
curl -sSL http://bit.ly/2ysb0FE -o bootstrap.sh
```

Download de exemplos (fabric-samples) e binários

```
bash bootstrap.sh -d 1.2.0
```

Download de imagens docker

```
bash bootstrap.sh -s -b 1.2.0
```

Download código fonte Hyperledger Fabric

```
git clone https://github.com/hyperledger/fabric.git
```

➤ first-network

➤ ./byfn.sh -generate

➤ ./byfn.sh -up

➤ ./byfn.sh -down

geração de artefatos via configtxgen. Usado para bootstrap do orderer/canal

```
configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block
```

```
configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx  
-channelID $CHANNEL_NAME
```

```
configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-  
artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP
```

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

cryptogen

```
cryptogen generate --config=./crypto-config.yaml
```

Inspecione

channel-artifacts

crypt-config

```
ü fabcar
```

```
ü ./startFabric.sh
```

A rede iniciada é a de
fabric-samples/basic-network

Ü fabric/examples/e2e_cli

A rede iniciada usa Kafka como Broker
Veja o arquivo configtx.yaml

e2ecli:

Ajustar os caminhos de binários Hyperledger
Ajustar configtx.yaml



Exemplo: execução de rede

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- Ü Coloca ordem nas transações, timestamp + enfileiramento.
- Ü Pode ser executado fora de contêiner.
- Ü Um chaincode está em um canal.
- Ü Um canal está em uma organização.
- Ü Um canal é um dos itens de “privacidade”, a versão 1.2.0 também traz dados privados.

Orderer

Garante a entrega e ordenação das transações.

A organização que gerencia os orderers preferencialmente não é um dos consorciados, é uma entidade confiável como o próprio consórcio. Eventualmente, o orderer ou orderers podem ficar em ambiente apartado.

Em ambientes de desenvolvimento, é possível usar a forma mais simples: solo com apenas um orderer para todo o consórcio.

Ao ser iniciado, um orderer cria um canal de configuração para posteriormente permitir a criação de canais de aplicações.

Um canal de aplicações é um mecanismo de separação lógica de comunicações. Dentro de um canal, ocorre broadcast entre os membros do canal.

A função do orderer é entregar aos peers uma sequência única de mensagens.

O cliente Juquinha usa a aplicação Tragaqui que usa uma API para Hyperledger Fabric em nodejs.

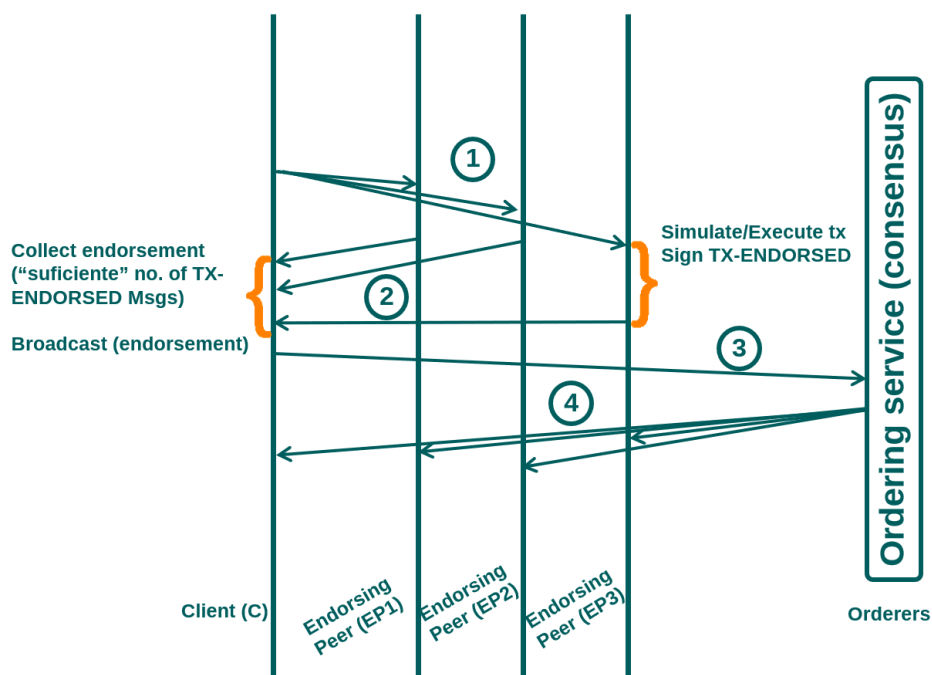
A aplicação solicita a encomenda de uma charrete 2019 ao fornecedor Charrete de Ouro

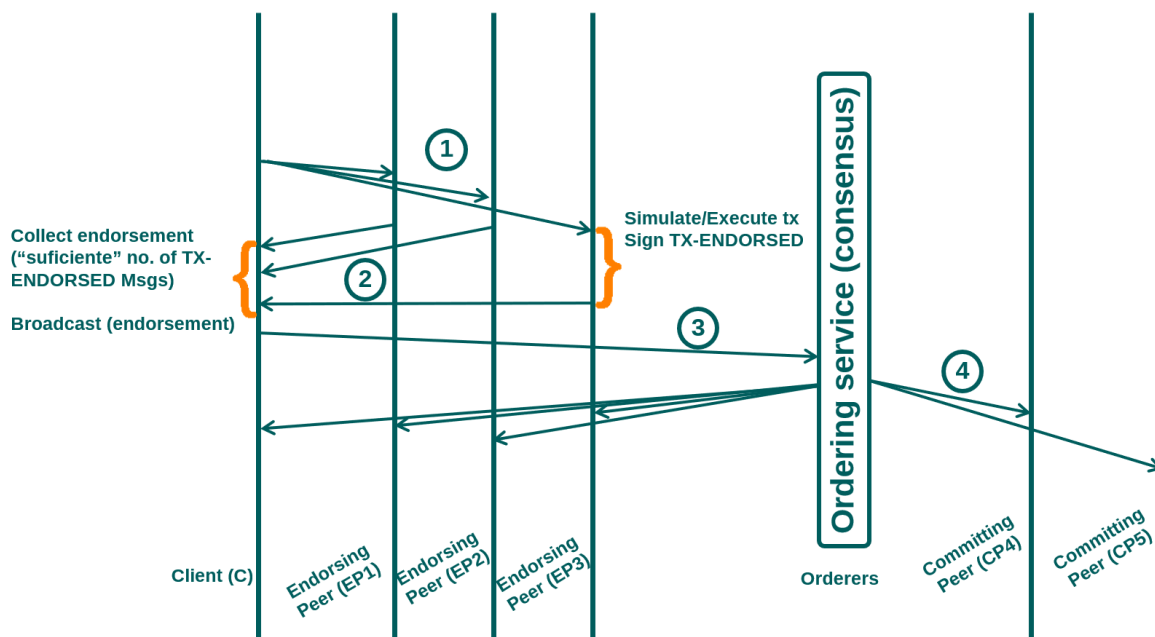
O peer da organização dos Amigos Compradores que Juquinha faz parte cria uma proposta de transação.

Esta proposta de transação deve ser validada pelo peer da organização do Juquinha e pelo peer da organização Charrete de Ouro.

A proposta de transação contém:

- Uma função de um chaincode comum as duas organizações;
- Os valores de entrada para essa função;
- Os valores finais dos ativos (1 charrete 2019 iria de Charrete de Ouro para Juquinha; 100 vaquinhas iriam de Juquinha para Charrete de Ouro)
- A API empacota tudo e junta as credenciais criptográficas.





Ü Validação.

A validação acontece pelos peers que endossam a transação:

- Testando os dados, a execução da função e a saída/estado final; Isso testa a validade dos valores de acordo com o estado atual da base de dados (vaquinhas e charretes)
- A assinatura da proposta;

Os endossos são enviados de volta ao canal que verifica a regra de endosso (política) e caso tudo esteja OK, o pacote confiável de transação é enviado ao orderer que ordena e enfileira os blocos que são enviados aos peers em broadcast.

Caso não ocorra endosso ou o solicitante não tenha direito a fazer a proposta por algum motivo (e.g. saldo) a transação INVÁLIDA também é passado ao orderer para ser inserida nos blocos.

O orderer não inspeciona o conteúdo das transações, ele simplesmente recebe as transações dos canais e as ordena em blocos por canal e por tempo.

Cada bloco enviado aos peers é anexado ao final da cadeia de blocos de um canal. Além disso, as bases de estado são alteradas para refletir as transações.

A escrita e mudança na base de estado gera um evento que pode ser usado pela aplicação para comunicar ao Juquinha o sucesso ou insucesso da operação.

Ü Em produção:

Ü Orderers + Kafka + zookeeper

Em uma arquitetura de produção, múltiplos orderers podem ser usados. Na arquitetura Hyperledger Fabric, são usados brokers Kafka para criar um cluster de mensageria que faz a função de manter os dados em uma ordem além de garantir replicação e distribuição/balanceamento dentro de uma rede.

Um cluster ZOOKEEPER (proxy e verificador de saúde) faz verificação dos nós Kafka.

Cada orderer Hyperledger Fabric deve conhecer o cluster Kafka para conseguir uma arquitetura multi orderer permitindo escalabilidade e o cluster Kafka vai garantir alta disponibilidade dos dados da ordenação.

Diagrama

Cliente

APPWEB

API

Proposta de Transação (chaincode+payload entrada/saida+assinaturas)

peer org1 endorsa(?)

peer org2 endorsa(?)

peer org3 endorsa(?)

Coleta endossos até conseguir o mínimo da política (ou não)

Orderer recebe a transação endorsada

peer org1 endorsou(?)

peer org2 endorsou(?)

peer org3 endorsou(?)

Orderer confirma que está OK, manda mudança de estado

e escreve no bloco e manda para os peers do canal.

Para saber mais <https://hyperledger-fabric.readthedocs.io/en/release-1.2/arch-deep-dive.html>

Ü fabric/examples/e2e_cli

Execução de um container orderer
cd ~/fabric-samples/basic-network
./generate.sh

docker-compose -f docker-compose.yml up orderer.example.com
CTRL + C
docker rm -f \$(docker ps -qa)

docker-compose -f docker-compose.yml up -d orderer.example.com
docker-compose -f docker-compose.yml down
docker ps
docker ps -qa

Ü ss -nlt

Ü sudo ss -nltop

```
docker-compose -f docker-compose.yml up -d orderer.example.com
```

```
docker ps
```

```
ss -nlt
```

State	Recv-Q	Send-Q	Local Address:Port	Peer
LISTEN	0	128	*:22	*:*
LISTEN	0	128	:::7050	:::*
LISTEN	0	128	:::22	:::*

```
sudo ss -nltop
```

State	Recv-Q	Send-Q	Local Address:Port	Peer
LISTEN	0	128	*:22	*:*
			users:(("sshd",pid=1121,fd=3))	
LISTEN	0	128	:::7050	:::*
			users:(("docker-proxy",pid=7393,fd=4))	
LISTEN	0	128	:::22	:::*
			users:(("sshd",pid=1121,fd=4))	

Ü ls -Rla crypto-config

ls -Rla crypto-config/ordererOrganizations/

```
Ü cat configtx.yaml
```

```
cat configtx.yaml
```

```
Ü cat docker-compose.yml
```

```
cat docker-compose.yml
```



Componentes: Peer e Client

[illegible]

- Ambiente docker criado a partir de Ubuntu.
- Binário peer.
- Configuração /etc/hyperledger.
- Variáveis definem peer e certificados.

Peers são processos intermediários de transações de negócios (chaincode) e de configuração (channel e outros).

Client é uma imagem especial criada para executar comandos diretamente sobre os peers.

Ao invés da imagem Client (cli), uma aplicação usa a API Hyperledger via uma de suas SDK.

- Nó de execução e de endosso (prova dos 9).
- Contêiner Docker.
- A execução de chaincode, inicia um contêiner de execução em separado (cc)

O comando básico para execução é
`peer node start`

O Hyperledger Fabric usa variáveis de ambiente para configurar os peers

Ü docker-compose

Ü docker exec

```
cd ~/fabric-samples/basic-network
docker-compose -f docker-compose.yml up
peer0.org1.example.com
```

CTRL + C

```
docker rm -f $(docker ps -qa)
```


Ü 7051

Ü Conexão com peers.

Ü 7052

Ü Acesso a chaincode.

Ü 7053

Ü Acesso a eventos.

```
docker-compose -f docker-compose.yml up -d peer0.org1.example.com
```

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
4298ffe6b14c	hyperledger/fabric-peer	"peer node start"	3 seconds ago	Up 2 seconds
0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp peer0.org1.example.com				

```
ss -nlt
```

State	Recv-Q	Send-Q	Local Address:Port	Peer
LISTEN	0	128	*:22	*:*
LISTEN	0	128	:::7050	:::*
LISTEN	0	128	:::7051	:::*
LISTEN	0	128	:::7053	:::*
LISTEN	0	128	:::22	:::*
LISTEN	0	128	:::5984	:::*

```
sudo ss -nltp
```

State	Recv-Q	Send-Q	Local Address:Port	Peer
LISTEN	0	128	*:22	*:*
users:(("sshd",pid=1121,fd=3))				
LISTEN	0	128	:::7050	:::*
users:(("docker-proxy",pid=10360,fd=4))				
LISTEN	0	128	:::7051	:::*
users:(("docker-proxy",pid=11256,fd=4))				
LISTEN	0	128	:::7053	:::*
users:(("docker-proxy",pid=11231,fd=4))				
LISTEN	0	128	:::22	:::*
users:(("sshd",pid=1121,fd=4))				
LISTEN	0	128	:::5984	:::*
users:(("docker-proxy",pid=10324,fd=4))				

Ü MSP

Ü TLS

Ü SHA256SUM Publico (Hexa=>String)

Exemplo de criação manual de certificados:

```
openssl ecparam -name prime256v1 -out prime256v1.pem #armazena o parametro de ec openssl
openssl ecparam -in prime256v1.pem -genkey -noout -out peer03.key
openssl req -new -key peer03.key -out peer03.csr -outform PEM
openssl x509 -req -CA /home/saito/Devel/1.2.0/fabric-samples/first-network/crypto-
config/peerOrganizations/org1.example.com/ca/ca.org1.example.com-cert.pem -CAkey
/home/saito/Devel/1.2.0/fabric-samples/first-network/crypto-
config/peerOrganizations/org1.example.com/ca/1597f18407c0006fb320d2673420efda02b73054
9d763bbdd1eaedc1a08b5207_sk -in peer03.csr -out peer03.crt -days 3650 -CAcreateserial
```

Ü Peers.

Ü Anchor.

Organizations:

- &Org1

Name: Org1MSP

ID: Org1MSP

MSPDir: crypto-config/peerOrganizations/org1.example.com/msp

AnchorPeers:

- Host: peer0.org1.example.com

Port: 7051

Ü cat docker-compose.yaml

Ü Executa comandos em nome de peers.

Ü Via docker-compose

[illegible]

- Um contêiner com CA completa.
- Permite todo o ciclo de certificados locais.

Ü ss -nlt

Ü sudo ss-nltop

Ü Início com crypto-config.yaml

Ü Pasta crypto-config

Ü service fabric-ca

Ü Admin via SDK

Ü Enroll via SDK



Orderer Kafka

4.1

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Ü Exemplo em e2e-cli

- Múltiplos orderers (mesma org).
- Múltiplos Kafka.
- Kafka como fila.
- ZooKeeper: Sentinela + HA.

Ü docker-compose.yml

Ü serviços Kafka

Ü serviços Zookeeper

- Cria artefatos.
- Executa docker-compose.
- `ss -nlt`

```
~/go/src/github.com/hyperledger/fabric/examples/e2e_cli  
./network_setup.sh down  
docker rm -f $(docker ps -qa)  
docker-compose down  
./network_setup.sh up -d  
docker ps  
ss -nlt  
sudo ss -nltop
```

Ü configtx.yaml

```
cat configtx.yaml
```

```
#####  
Orderer: &OrdererDefaults
```

```
# Orderer Type: The orderer implementation to start  
# Available types are "solo" and "kafka"  
OrdererType: kafka
```

```
...
```

```
Kafka:
```

```
# Brokers: A list of Kafka brokers to which the orderer connects. Edit  
# this list to identify the brokers of the ordering service.  
# NOTE: Use IP:port notation.
```

```
Brokers:
```

- kafka0:9092
- kafka1:9092
- kafka2:9092
- kafka3:9092

- O tipo é Kafka.
- Múltiplos orderers (mesma org).
- Múltiplos Kafka.
- Kafka como fila.
- ZooKeeper: Sentinela + HA.

```
zookeeper0:  
  container_name: zookeeper0  
  extends:  
    file: base/docker-compose-base.yaml  
    service: zookeeper  
  environment:  
    - ZOO_MY_ID=1  
    - ZOO_SERVERS=server.1=zookeeper0:2888:3888 server.2=zookeeper1:2888:3888  
      server.3=zookeeper2:2888:3888
```

e

```
kafka0:  
  container_name: kafka0  
  extends:  
    file: base/docker-compose-base.yaml  
    service: kafka  
  environment:  
    - KAFKA_BROKER_ID=0  
    - KAFKA_MIN_INSYNC_REPLICAS=2  
    - KAFKA_DEFAULT_REPLICATION_FACTOR=3  
    - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,zookeeper2:2181  
  depends_on:  
    - zookeeper0  
    - zookeeper1  
    - zookeeper2
```

[illegible]

- Couchdb x LevelDB
- LevelDB biblioteca para uso (memória e local).
- CouchDB é cliente servidor, produção.
- CouchDB permite consultas complexas.

A paginação não é possível via skip e limits do CouchDB. Isso é uma limitação artificial em código e prevista para ser removida em versões futuras do Hyperledger Fabric

Ü Docker-compose-cli.yaml do first-network

Ü 0.0.0.0:5984 → 5984/tcp

Ü 0.0.0.0:6984 -> 5984/tcp

Ü 0.0.0.0:7984 -> 5984/tcp

Ü 0.0.0.0:8984 -> 5984/tcp

Definição nos arquivos docker-compose

Ü `./byfn.sh -m up -s couchdb`

```
cd ~/fabric-samples/first-network
```

```
./byfn.sh -m down
```

```
docker rm -f $(docker ps -qa)
```

```
./byfn.sh -m up -s couchdb
```

Starting for channel 'mychannel' with CLI timeout of '10' seconds and CLI delay of '3' seconds and using database 'couchdb'
Continue? [Y/n]



Rede Kafka

4.3

[illegible]

Ü e2e_cli

O orderer é fundamental para as operações em Hyperledger Fabric.

A configuração inicial com orderer Solo não é escalável e apresenta ponto único de falha.

A solução do projeto Hyperledger Fabric é uma arquitetura baseada em Kafka em um padrão semelhante a filas de mensageria em Java EE.

Assim, múltiplos orderers podem ser criados e apontados para Cluster Kafka.

Isso permite distribuir orderers geograficamente, assim como manter um cluster kafka também distribuído.

Ü Todos os nós Kafka e orderers na mesma máquina

e2ecli:

Ajustar os caminhos de binários Hyperledger

Ajustar configtx.yaml

- Ü Docker: DNS interno é local, 127.0.0.11
- Ü Como resolver nomes da outra máquina/VM

- 1) Separação de docker-compose
- 2) external_hosts e a rede Docker

- Ü Cuidados: latência.
- Ü Mesmo processo de separação.
- Ü Necessário iniciar com o configtx que contém nomes que não são FQDN.

[illegible]

Ü Código desenvolvido para interagir com o Ledger.

Ü Modifica o estado/dados (CRUD).

ü Aplica lógica de negócios.

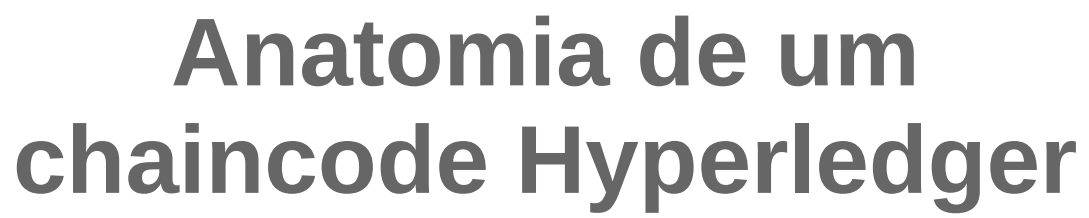
[illegible]

- Bitcoin => Apenas transferência.
- Ethereum => Transferência acontece se o resultado da execução de regras de negócios der OK (smart contracts).
- Também permite o uso de Oracles, terceiros que entregam informações confiáveis (cotações no tempo, estoque, indexadores).
- Tempo de transação: fora de controle.

Também questões como purging, backup, restore, cópia e garantia de execução ao longo do tempo estão fora de controle.

- Código desenvolvido para interagir com o Ledger.
- Modifica o estado/dados (CRUD).
- Aplica lógica de negócios.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



- Node.js

- package.json

- dependencies "fabric-shim": "unstable"

- App.js

- Init

- Inicializa valores de estado / dados (chaincode instantiate)

- Invoke

- Manipula os valores de estado/dados

ü fabric-samples/chaincode/
chaincode_example02/node
ü fabric-samples/chaincode/
marbles02/node

```
Ü const shim = require('fabric-shim');  
Ü shim.start(new Chaincode());
```


- golang principal linguagem Hyperledger Fabric.
- Também é base de projetos como Docker.
- C simplificado, mas com ponteiros...

- fabric-samples/chaincode-docker-devmode
- Execução manual do chaincode.
- Permite modificações a quente.

```
cd fabric-samples/chaincode-docker-devmode
docker-compose -f docker-compose-simple.yaml up -d
entre no container:
docker exec -it chaincode bash
apt update
apt install -y vim
```

```
cp -rv chaincode_example02 sacc
cd sacc/go
go build
```

```
Execute o chaincode
CORE_PEER_ADDRESS=peer:7052
CORE_CHAINCODE_ID_NAME=mycc:0 ./sacc
```

Ü Install, Instantiate, Invoke, Query.

em outro terminal
docker exec -it cli bash

Instale e instancie o chaincode

```
peer chaincode install -p chaincodedev/chaincode/sacc -n  
mycc -v 0
```

```
peer chaincode instantiate -n mycc -v 0 -c '{"Args":  
["a","10"]}' -C myc
```

Execute

```
peer chaincode invoke -n mycc -c '{"Args":["set", "foo",  
"bar"]}' -C myc
```

Consulte

```
peer chaincode query -n mycc -c '{"Args":["get","a"]}' -C myc  
# 10
```

```
peer chaincode query -n mycc -c '{"Args":["get","foo"]}' -C  
myc # bar
```



Lab com Javascript

5.3

[illegible]

Ü fabcar

Ü marbles



Instalar, Instanciar, Invocar e Consultar

[illegible]

- comandos peer
- peer chaincode install
- peer chaincode instantiate
- peer chaincode invoke
- peer chaincode query

[illegible]

[illegible]

Estrutura de certificados

MSP

define os certificados usados para autenticação e autorização;

No caso da pasta crypto-config, existem pastas para organizações:

`./ordererOrganizations/example.com/msp`

`./peerOrganizations/org2.example.com/msp`

para os nós:

`./ordererOrganizations/example.com/orderers/orderer.example.com/msp`

`./peerOrganizations/org2.example.com/peers/peer0.org2.example.com/msp`

e para usuários nomeados:

`./ordererOrganizations/example.com/users/Admin@example.com/msp`

`./peerOrganizations/org2.example.com/users/Admin@org2.example.com`

Estrutura de certificados

MSP

define os certificados usados para autenticação e autorização;

No caso da pasta crypto-config, existem pastas para organizações:

`./ordererOrganizations/example.com/msp`

`./peerOrganizations/org2.example.com/msp`

para os nós:

`./ordererOrganizations/example.com/orderers/orderer.example.com/msp`

`./peerOrganizations/org2.example.com/peers/peer0.org2.example.com/msp`

e para usuários nomeados:

`./ordererOrganizations/example.com/users/Admin@example.com/msp`

`./peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp`

Os certificados das entidades (entidade é o item que tem um certificado) são armazenadas da seguinte forma:

- O certificado público é armazenado em formato pem e em uma pasta chamada signcerts

`openssl x509 -text -noout -in`

`./ordererOrganizations/example.com/orderers/orderer.example.com/msp/signcerts/orderer.example.com-cert.pem`

- A chave privada é armazenada em um arquivo em uma pasta chamada keystore. O nome do arquivo é baseado no SKI (Subject Key Identifier) seguido de `_sk`, o ski é o sha256sum da chave pública (em string e não em HEX)

ü fabric/sampleconfig

fabric/sampleconfig



Permissões no blockchain: iniciando e reconfigurando

[illegible]

Ü Apenas na geração, a modificação necessitará de configtxlator.

Ü A maioria dos participantes do consórcio é necessária para adicionar um membro.

Inspeção:
um arquivo json de configuração via jq

[illegible]

Ü A partir da versão 1.1.0-preview, é possível alterar a configuração do consórcio, após início das transações

Antes da versão 1.1.0-preview não era possível modificar o consórcio depois de iniciado.
O procedimento é complexo, mas funcional.

Ü Protobuffers é uma linguagem google/golang de notação de objetos e configurações.

Os protobuffers, também são fonte de documentação.

Ü A sua análise, em um cenário, pode ajudar a entender que parâmetros são válidos.

[illegible]

ü jq é um manipulador em Java para arquivos JSON.

ü <https://stedolan.github.io/jq/>

Os dados no blockchain não são armazenados em JSON nativamente. É possível acessar os dados de bloco de configuração pelo uso da ferramenta configtxlator que permite a tradução de configurações de formato nativo para JSON que posteriormente podem ser manipulados e a partir de arquivos de diferença reinseridos nos blocos de configuração.

Ü Reconfigurar => coletar a cadeia de configuração, converter para JSON, modificar, conseguir o Delta de modificação, transformar e enviar via configtxlator.

O script `eyfn.sh`, `extends your first network`, usa a metodologia disponível para modificação de configurações no Hyperledger Fabric, no caso a adição de uma nova organização. As configurações são armazenadas em Blockchain e modificações de configurações são transações que serão armazenadas no Blockchain de configuração. A configuração final é a configuração em uso mais os blocos de modificação. O projeto disponibiliza um tradutor de configurações em Blockchain para JSON, o `configtxlator`. O `configtxlator` é uma ferramenta que traduz pedidos via API REST para chamadas de consulta ou modificação nos blocos de configuração.

Apesar do processo ainda complexo, a ação que deve ser feita é conseguir um JSON de diferença de configuração e transformar em arquivo configuração de bloco compatível com o Hyperledger Fabric.

É possível inspecionar os arquivos JSON criados dentro do container cli:

`config.json` representa a configuração atual.

`modified_config.json` representa a configuração após a adição da Org3

O processo pode ser modificado para que o arquivo da nova organização `org3.json` seja fornecido e não criado no próprio ambiente.

O processo de modificação de configuração também deve seguir essa metodologia.



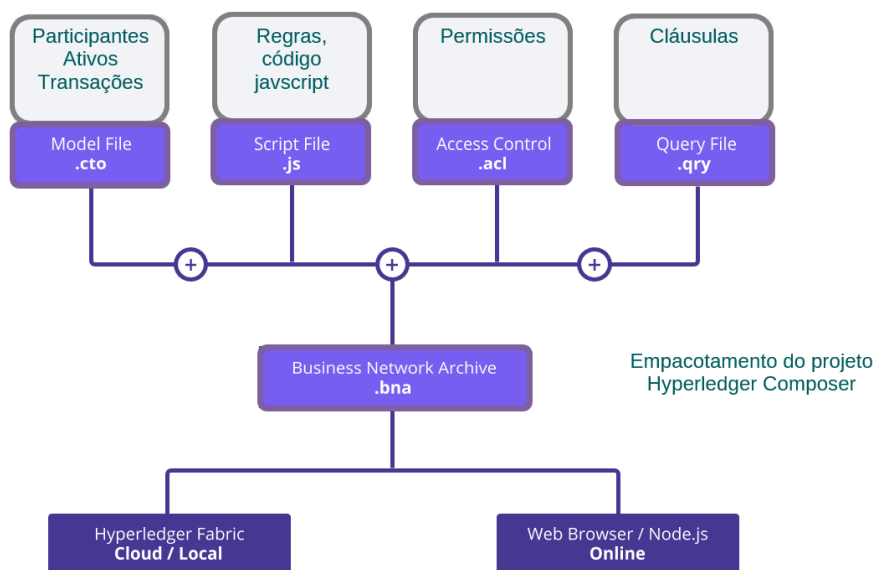
Perfil de consenso: Coleta de assinaturas

[illegible]

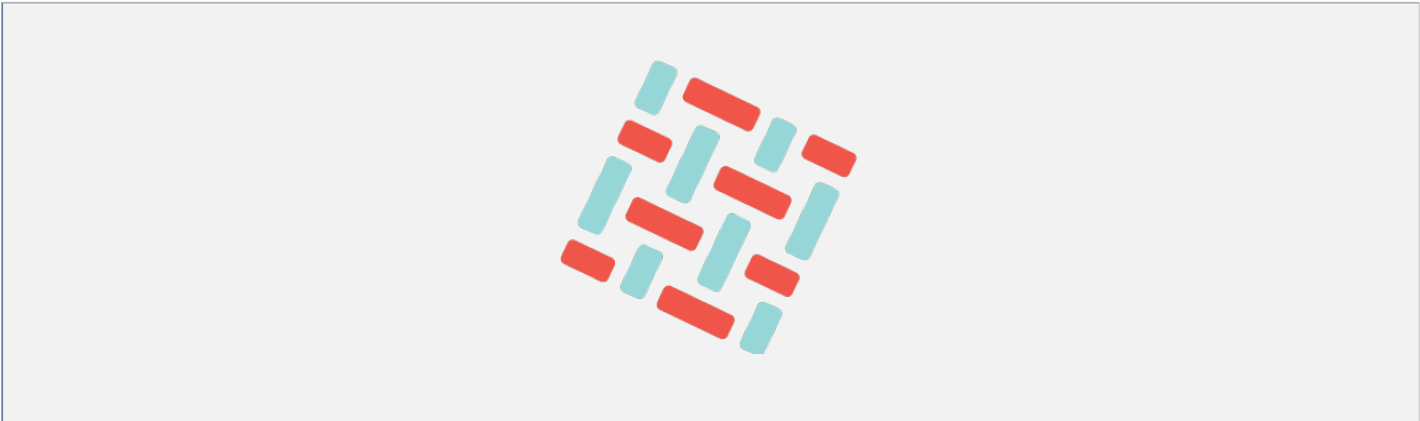
[illegible]

[illegible]

[illegible]



Deploy em redes existentes com uso de arquivos de credenciais



O que a tecnologia ajuda a resolver

[illegible]

O Hyperledger Composer é um projeto independente do Hyperledger Fabric. Inicialmente idealizado para funcionar com múltiplas plataformas de blockchain, atualmente funciona apenas com Hyperledger Fabric.

O objetivo do projeto é tornar a modelagem de blockchains mais próxima de analistas de negócios. Apesar de funcionar com Hyperledger Fabric, sua filosofia, linguagens de modelagem e maneiras de implementar regras e restrições é totalmente distinta do Hyperledger Fabric.

O ambiente Playground permite execução de transações baseadas em JSON sem a necessidade de seguir o ciclo de vida de chaincode. Isso permite a criação de modelos e testes diretamente em uma interface web.

Nota: O modelo separado do Hyperledger Composer torna razoavelmente complexa a inserção de redes criadas em Composer em redes Hyperledger Fabric existentes.

As definições de transações, atores e permissões são modelados em interface web e independentes da rede de fato.

Um dos objetivos é ser uma ferramenta para analistas de negócios. Atualmente exige conhecimento de Javascript e modelagem em JSON.

Permitir documentação e empacotamento: Ciclo de vida

O modelo baseado em arquivos descritores e empacotamento traz a figura possível no futuro de deploys de conjuntos completos de atores, código e permissionamento.

Além das ACL intrínsecas do Hyperledger Fabric, o Hyperledger Composer cria modelos de ACL baseados em código.



Execução

[illegible]

```
#com usuario que tenha direito a sudo: aluno
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
bash prereqs-ubuntu.sh
```

```
#logout e login
```

```
docker run hello-world
```

```
npm install -g composer-cli@0.20
```

```
npm install -g composer-rest-server@0.20
```

```
npm install -g generator-hyperledger-composer@0.20
```

```
npm install -g yo
```

```
npm install -g composer-playground@0.20
```

```
mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers
```

```
curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz  
tar -xvf fabric-dev-servers.tar.gz
```

```
cd ~/fabric-dev-servers  
export FABRIC_VERSION=hlfv12  
./downloadFabric.sh
```

```
cd ~/fabric-dev-servers  
export FABRIC_VERSION=hlfv12  
./startFabric.sh  
./createPeerAdminCard.sh
```

```
composer-playground  
~/fabric-dev-servers/stopFabric.sh
```

Acesse em
<http://192.168.56.101:8080>

[illegible]



Modelando Redes

9.2

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins or other markings visible.

[illegible]

Ü Javascript, mas sem acesso a bibliotecas.

////////////////////////////////////

Lógica @ não é comentário, é anotação!!!!

////////////////////////////////////

/**

* Track the trade of a commodity from one trader to another

* @param {org.example.Jogo.Bafo} bafo - the trade to be processed

* @transaction

*/

async function Tapa(Bafo) {

 Bafo.figurinha.owner=Bafo.newOwner;

 let assetRegistry = await getAssetRegistry('org.example.Jogo.Figurinha');

 await assetRegistry.update(Bafo.figurinha);

}

////////////////////////////////////



Modelando: Participantes e ativos

[illegible]

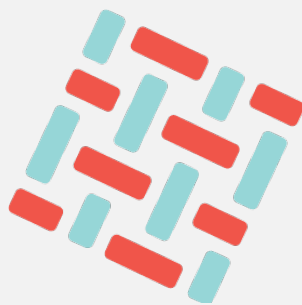
ü Modelados em JSON.

Os atores e ativos são modelados em um arquivo de modelo com extensão .cto. A declaração é basicamente uma declaração de tipos que envolvem os ativos (Assets), atores (Participants) e transações (Transactions) dentro de uma hierarquia de empacotamento semelhante a Java.

Ü Modelados em JSON.

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

namespace org.example.Jogo
asset Figurinha identified by figurinhaId {
  o String figurinhaId
  --> Jogador owner
  o String value
}
participant Jogador identified by jogadorId {
  o String jogadorId
  o String Nome
  o String Sobrenome
}
transaction Bafo {
  --> Figurinha figurinha
  --> Jogador newOwner
}
```

Deploy em rede Hyperledger Fabric

9.5

ü Desafio: criar um arquivo de perfil de rede Hyperledger Fabric para usar no deploy do arquivo Hyperledger Composer.

[illegible]



Projeto Final

[illegible]

