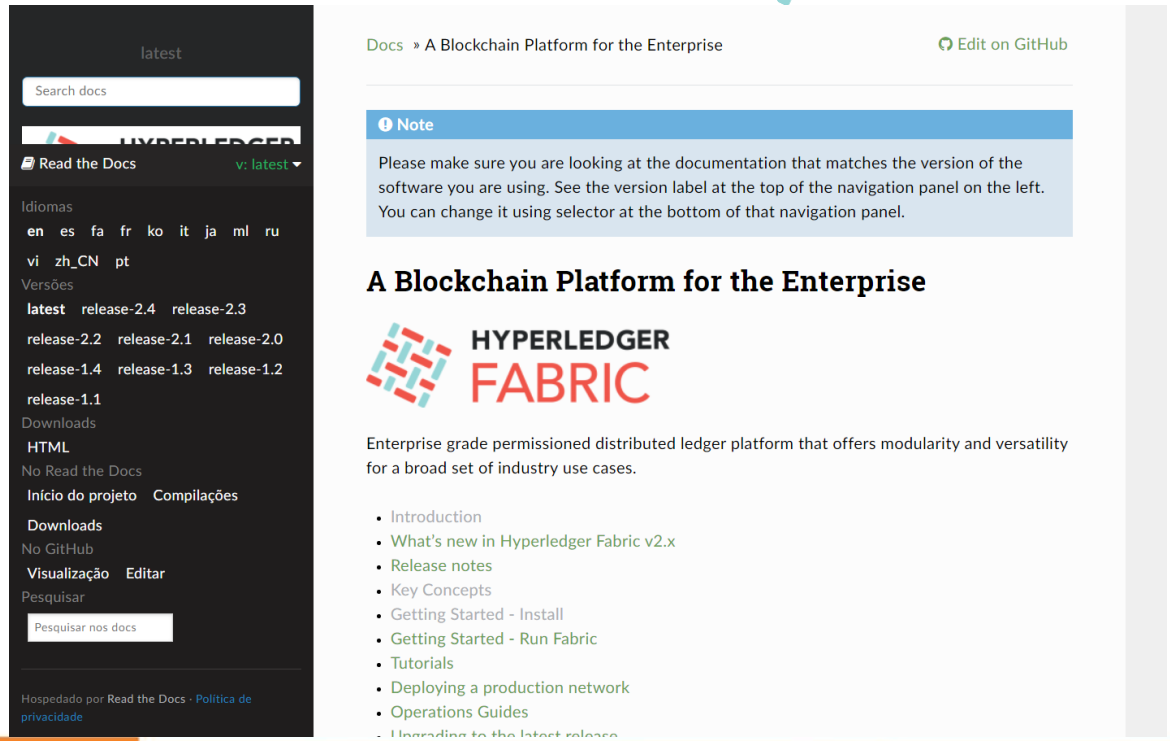
ESCOLA SUPERIOR DE REDES

Módulo 3
Hyperledger Fabric (parte 2)

GoLedger

Utilizando o readthedocs



Configuração inicial

Esse curso irá utilizar o ambiente Linux para desenvolvimento.

- Git
- Curl
- Docker
- Docker images do Fabric (na versão desejada)
- GoLang
- NodeJs

Verificando o docker

sudo systemctl enable docker

Limpando o docker

docker stop $(docker ps –a –q)
docker rm $(docker ps –a –q)
docker rmi –f $(docker images)
docker volume prune
docker system prune

Fabric-samples e Docker images v2.2

Instalando tudo

curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.2.9 1.5.5

Repositório fabric-samples

https://github.com/hyperledger/fabric-samples.git

Imagens oficiais

https://hub.docker.com/r/hyperledger/fabric-peer
https://hub.docker.com/r/hyperledger/fabric-orderer
https://hub.docker.com/r/hyperledger/fabric-orderer
https://hub.docker.com/r/hyperledger/fabric-tools

Test-network v2.2

Dentro do diretório *fabric-samples/test-network*

Utilizar o scritpt network *network.sh*

Se não for a 1ª vez limpar o ambiente

./network.sh down

Subir os containers com o comando

./network.sh up

Ferramenta *cryptogen*

Ferramenta de geração de MSPs simulados para ambiente de desenvolvimento

Utiliza arquivos de configuração (yaml)
Ex:

```
PeerOrgs:
- Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
Template:
      Count: 1
      SANS:
        - localhost
Users:
      Count: 1
```

Utilizando Fabric-CA

Se não for a 1ª vez limpar o ambiente

./network.sh up –ca

tree organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/

```
organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
└── msp
    ├── cacerts
    │   └── localhost-7054-ca-org1.pem
    ├── config.yaml
    ├── IssuerPublicKey
    ├── IssuerRevocationPublicKey
    ├── keystore
    │   └── 868e452a1fa47d12bfefbe306f7f2e9ebf383599c339914541b688b6ad5a3b6a_sk
    ├── signcerts
    │   └── cert.pem
    └── user
```

Certificados gerados

Os certificados gerados para uso na rede Hyperledger Fabric são identificados:
Common Name (CN): nome e domínio do certificado
Organization (O): organização do certificado
Organization Unit (OU): peer, orderer, client, admin

openssl x509 -in localhost-7054-ca-org1.pem -text –noout

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5f:d1:6a:5f:a4:5f:0a:0f:a9:cd:1b:1c:d8:90:ee:38:9d:d3:b5:37
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
        Validity
            Not Before: Nov  7 17:15:00 2022 GMT
            Not After : Nov  3 17:15:00 2037 GMT
        Subject: C = US, ST = North Carolina, L = Durham, O = org1.example.com, CN = ca.org1.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:c9:3a:2f:1f:65:2b:4f:b8:cb:8d:d9:52:63:36:...
```

Arquitetura proposta

Rede – *blockchain business network* - a ser criada

3 orgs

- ***org1***
- ***org2***
- ***orderer***

1 channel – *mychannel*

1 chaincode - *fabcar*

Ferramenta *configtxgen*

Gera artefatos de configuração da rede e inspeção da rede

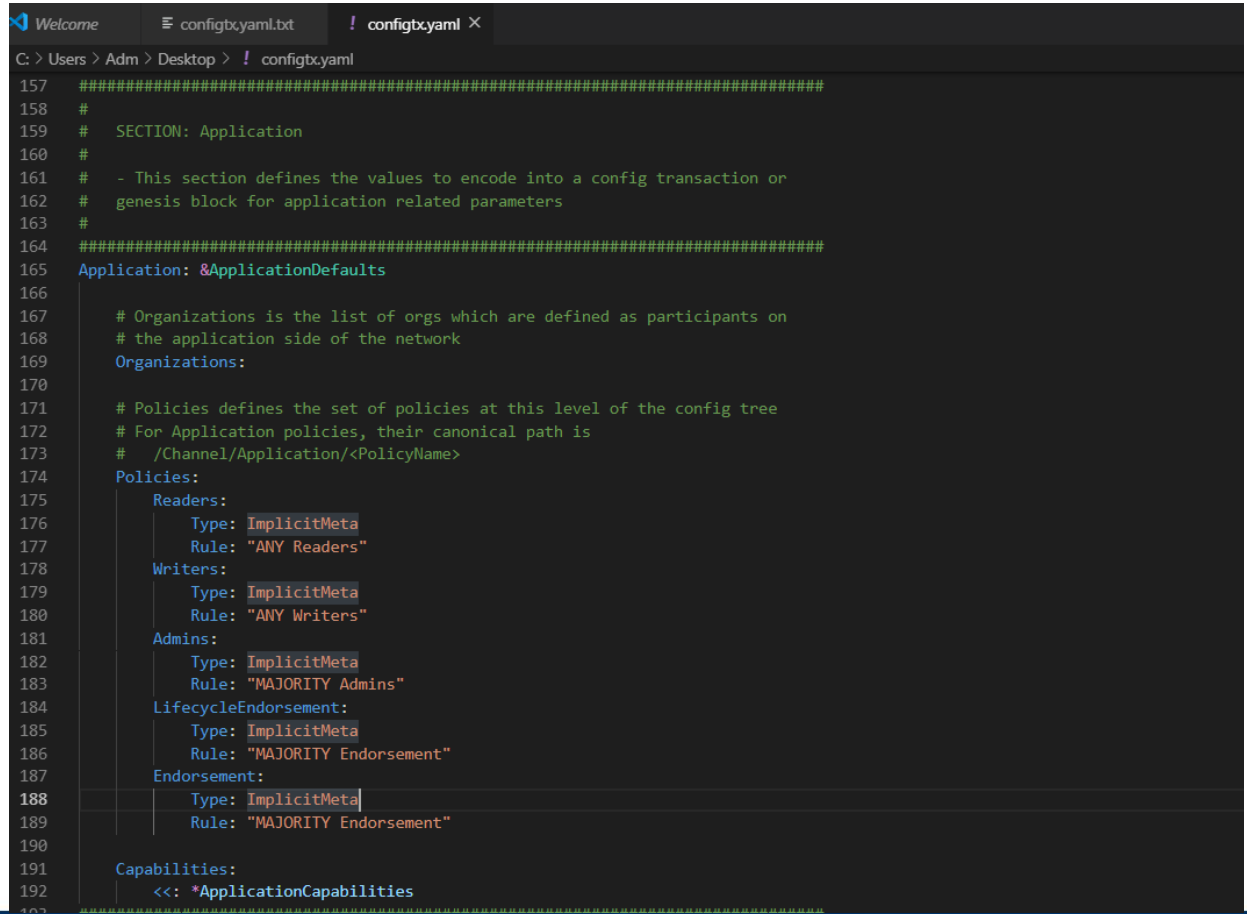
Os artefatos de configuração são propostos ao orderer para poder realizar ações tais como:

- Criar *genesis block*
- Criar novo channel
- Adicionar peer
- Adicionar nova organização

Configura o arquivo em *configtx.yaml*

Deve estar em *FABRIC_CFG_PATH* ou utilizar o parâmetro *-configPath*

Configtx.yaml



Arquivo configtx.yaml

O arquivo configtx.yaml possui diversas sessões para realizar a criação de artefatos de configuração.

- **Organizations**
- **Capabilities (Channel, Orderer, Application)**
- **Application**
- **Orderer**
- **Channel**
- **Profiles**

Criando o bloco gênesis

O *genesis block* possui as regras iniciais de uma Blockchain business network. No exemplo ele é criado no profile *TwoOrgsOrdererGenesis:*

```
TwoOrgsOrdererGenesis:
    <<: *ChannelDefaults
    Orderer:
        <<: *OrdererDefaults
        Organizations:
            - *OrdererOrg
        Capabilities:
            <<: *OrdererCapabilities
    Consortiums:
        SampleConsortium:
            Organizations:
                - *Org1
                - *Org2
```

Comando para gerar o artefato de configuração do genesis block

configtxgen -profile TwoOrgsOrdererGenesis -channelID system-channel -outputBlock ./system-genesis-block/genesis.block

Inspecionando o bloco gênesis

configtxgen -inspectBlock ./system-genesis-block/genesis.block

Iniciando a Business Network v2.2

Inspecionando o bloco gênesis

configtxgen -inspectBlock ./system-genesis-block/genesis.block

O orderer vai utilizar o *genesis block* para iniciar a rede.

Configuração do docker:

```
orderer.example.com:
    environment:
        - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
 volumes:
        - ../system-genesis-block/genesis.block:/var/hyperledger/orderer/orderer.genesis.block
```

Configuração do docker:

```
2022-11-07 20:37:39.800 UTC [orderer.commmon.multichannel] Initialize -> INFO 013 Starting
system channel 'system-channel' with genesis block hash
ca7bb35a2fd70f7c4c86b8f0b778d45758e249f93ab8d47b8621dbe2a009e174 and orderer type
etcdraft
```

Criando um channel na rede

Utilizando o comando configtxgen para o profile **TwoOrgsChannel** para gerar a transação de criação de channel.

```
TwoOrgsChannel:
    Consortium: SampleConsortium
    <<: *ChannelDefaults
    Application:
        <<: *ApplicationDefaults
        Organizations:
            - *Org1
            - *Org2
        Capabilities:
            <<: *ApplicationCapabilities
```

Criação da transação *mychannel.tx*

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

Criando o channel na rede

```
peer channel create -o localhost:7050 -c mychannel --ordererTLSHostnameOverride orderer.example.com -f ./channel-artifacts/mychannel.tx --outputBlock ./channel-artifacts/mychannel.block --tls --cafile ...
```

Bloco de configuração ***mychannel.block*** criado.

Criando um channel na rede

Utilizando o comando configtxgen para o profile **TwoOrgsChannel** para gerar a transação de criação de channel.

```
TwoOrgsChannel:
    Consortium: SampleConsortium
    <<: *ChannelDefaults
    Application:
        <<: *ApplicationDefaults
        Organizations:
            - *Org1
            - *Org2
        Capabilities:
            <<: *ApplicationCapabilities
```

Criação da transação *mychannel.tx*

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

Criando o channel na rede

```
peer channel create -o localhost:7050 -c mychannel --ordererTLSHostnameOverride orderer.example.com -f ./channel-artifacts/mychannel.tx --outputBlock ./channel-artifacts/mychannel.block --tls --cafile ...
```

Bloco de configuração ***mychannel.block*** criado.

Entrada de um peer no channel

Entrada de um peer de uma org no channel através da aplicação do bloco de configuração.

setGlobals $ORG
peer channel join -b ./channel-artifacts/mychannel.block

O peer dentro do channel representa a org dentro do channnel

2022-11-07 22:52:52.152 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel

Utilizando o container *cli*

Container cli é para da imagem hyperledger/fabric-tools

Possibilita a operação entre peers sem a mudança de containers

Variáveis de ambiente utilizadas pelo comando ***peer***

CORE_PEER_TLS_ENABLED
CORE_PEER_LOCALMSPID
CORE_PEER_TLS_ROOTCERT_FILE
CORE_PEER_MSPCONFIGPATH
CORE_PEER_ADDRESS

Exemplo

```
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
```

Atualizando blocos de configuração

Algumas operações são realizadas através da atualização dos blocos de configuração

peer channel fetch config | newest | oldest | blockNumber outputFile -o orderer.example.com:7050 –c mychannel ...

O bloco de configuração é modificado e atualizado no channel

peer channel update -o orderer.example.com:7050 -c mychannel -f txFile

Configuração do Anchor Peer

Os ***anchor peers*** representam os *peers* que vão receber os novos blocos do ***orderer*** após um commit bem sucedido.

A definição de anchor peer acontece com a atualização do bloco de configuração executando o script ***setAnchorPeer.sh*** dentro do container *cli*.

Atualização de blocos de configuração com o uso da ferramenta ***configtxlator*** para converter de json para protobuf.

Chaincode lifecycle

Ciclo de vida dos chaincodes instanciados no ***channel***

O channel possui uma política de endorso (***endorsing policy***) para a instanciação e atualização do chaincode no channel.

Cada chaincode possui uma ***endorsing policy*** para as suas transações.

Cada ***endorsing policy*** representa o conjunto mínimo de assinaturas para para completer a transação com sucesso. Ex:

Maioria simples
Todos
Qualquer um
Regra lógica, ex: org1 and [org2 or org3]

A instalação e instanciação do chaincode no channel requer as seguintes etapas:

Package: empacotamento do chaincode em um arquivo tar.
Install: instalação do chaincode nos endorsing peers.
Approve: aprovação do chaincode pelas orgs para validar o lifecycle endorsing policy
Commit: instanciar o chaincode no channel
Init (opcional): realizar uma transação inicial.

Instalando um chaincode no channel (FabCar-go)

As seguintes etapas serão realizadas após o comando:

./network.sh deployCC -ccn fabcar -ccp ../chaincode/fabcar/go -ccl go

Preparando o chaincode

go mod vendor

Criando um pacote para o peer

peer lifecycle chaincode package fabcar.tar.gz --path ../fabcar/go --lang golang --label fabcar_1.0

Instalando o chaincode no peer

setGlobals $ORG
peer lifecycle chaincode install fabcar.tar.gz

Verificando os chaincodes instalados

peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: fabcar_1.0:6c5c429e8a6734ff978f54d12a1d9e5e5296663e20dcacff6dd2276bb0e8b12b, Label: fabcar_1.0

Um chaincode instalado precisa ser instanciado para transformar o ***peer*** em ***endorsing peer***

Instanciando um chaincode

Aprovando o chaincode

setGloblals $ORG
peer lifecycle chaincode approveformyorg -o localhost:7050 ... --channelID mychannel --name fabcar --version 1.0 --package-id fabcar_1.0:6c5c429e8a6734ff978f54d12a1d9e5e5296663e20dcacff6dd2276bb0e8b12b --sequence 1

Aprovar em todas as orgs e verificar com a função:

peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name fabcar --version 1.0 --sequence 1 --output json

Comitar o chaincode

peer lifecycle chaincode commit -o localhost:7050 ... --channelID mychannel --peerAddresses localhost:7051 --peerAddresses localhost:9051 --name fabcar --version 1.0 --sequence 1

Inicializando um chaincode

Configuração das variáveis de ambiente para inicializar o chaincode

```
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

Chamando a primeira transação

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel
-n fabcar --peerAddresses localhost:7051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --
tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c
'{"function":"InitLedger","Args":[]}'
```

Testando o chaincode

Verificando o estado do peer

peer chaincode query -C mychannel -n fabcar -c '{"Args":["QueryAllCars",""]}'

[{"Key":"CAR0","Record":{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}},{"Key":"CAR1","Record":{"make":"Ford","model":"Mustang","colour":"red","owner":"Brad"}},{"Key":"CAR2","Record":{"make":"Hyundai","model":"Tucson","colour":"green","owner":"JinSoo"}},{"Key":"CAR3","Record":{"make":"Volkswagen","model":"Passat","colour":"yellow","owner":"Max"}},{"Key":"CAR4","Record":{"make":"Tesla","model":"S","colour":"black","owner":"Adriana"}},{"Key":"CAR5","Record":{"make":"Peugeot","model":"205","colour":"purple","owner":"Michel"}},{"Key":"CAR6","Record":{"make":"Chery","model":"S22L","colour":"white","owner":"Aarav"}},{"Key":"CAR7","Record":{"make":"Fiat","model":"Punto","colour":"violet","owner":"Pari"}},{"Key":"CAR8","Record":{"make":"Tata","model":"Nano","colour":"indigo","owner":"Valeria"}},{"Key":"CAR9","Record":{"make":"Holden","model":"Barina","colour":"brown","owner":"Shotaro"}}]

O operação query retorna o transaction proposal response.

ESCOLA SUPERIOR DE REDES

Tarefa

PRÓXIMO MÓDULO:
4
Hyperledger Fabric
(parte 3)

GoLedger