

Minicurso Introdução às Tecnologias Blockchain: Práticas

Prof. Rogério Aparecido Gonçalves Universidade Tecnológica Federal do Paraná (UTFPR)

Blockchain é uma tecnologia nova e considerada revolucionária e disruptiva, sendo até mesmo comparada, quanto ao impacto, ao surgimento da Internet. Neste minicurso serão apresentados conceitos e alguns fundamentos básicos relacionadas à Tecnologia Blockchain. Neste material complementar são apresentadas as práticas relacionadas ao desenvolvimento com *Ethereum*.

Sumário

1	Prática: Instalando o Cliente <i>Ethereum</i>: Geth	3
1.1	Instalando o Geth	3
1.2	Executando o Geth	3
1.3	Redes de Teste	5
1.4	Executando o Geth com uma Rede de Teste	7
1.5	Criando Contas na Rede	9
1.6	Listando as Contas	10
1.7	Clientes de Consenso	11
1.8	Instalando o cliente de consenso Prysm	11
1.9	Gerando um arquivo <i>JWT Secret</i>	11
1.10	Executando um Cliente de Execução	12
1.11	Executando um nó beacon usando Prysm	13
1.12	Executando o Console JavaScript	13
1.13	Verificação do Funcionamento da Rede	14
1.14	Listando as Contas pelo Console	14
1.15	Teste de transferência de valores entre carteiras	16
1.16	Leitura Recomendada	17
2	Prática: Criando uma Rede <i>Ethereum</i> Privada	18
2.1	Criando uma Rede Privada Local	18
2.2	Executando a nova Rede	19
2.3	Interagindo com a nova Rede	20
2.4	Criando contas na nova Rede	21
2.5	Verificando o saldo das carteiras	23
2.6	Gerar algum saldo para as carteiras	23
2.7	Transferências entre as carteiras	24
2.8	Leitura Recomendada	32
3	Prática: Instalando o Solidity	33
3.1	Compilando um Exemplo	33
3.2	Visualizando o <i>bytecode</i> gerado	33
3.3	Estimando a taxa gas	34
3.4	Gerando a ABI	34

3.5	Processo de Compilação Completo	35
3.6	Visualizando os <i>Opcodes</i>	35
3.7	Leitura Recomendada	36
4	Prática: Introdução ao Web3	37
4.1	Instalação das Ferramentas	37
4.2	Explorando Web3 com Geth	37
4.3	Web3 deployment	38
4.4	Web3 deployment: Executar o <i>Geth client</i>	38
4.5	Web3 deployment: Criar um script de <i>deployment</i>	38
4.6	Web3 deployment: Fazendo o <i>deploy</i> pelo <i>Geth console</i>	40
4.7	Web3 deployment: Interagindo com o contrato	42
4.8	Interagindo com contratos via frontends web	43
4.9	Biblioteca Javascript Web3.js	44
4.10	Development frameworks	44
4.11	Configuração do Ganache	45
4.12	Interagindo com um contrato	45
4.13	Developing a proof of idea project	46
4.14	Creating the ideap project	46
4.15	Patent DApp	47
4.16	IPFS	47
4.17	Atividade	47
4.18	Leitura Recomendada	48
5	Prática: Introdução à Tokenização	49
5.1	Desenvolvendo um Token	49
6	Word Cloud	49
	Referências	49

1 Prática: Instalando o Cliente *Ethereum*: Geth

A proposta desta prática é vermos o funcionamento do *software* cliente da rede *Ethereum*, o Geth, transformando a máquina em um nó da rede *Ethereum*.

1.1 Instalando o Geth

O cliente padrão Geth pode ser instalado em sistemas derivados do Debian e Ubuntu com o pacote `ethereum`:

```
1 $ sudo apt-get install -y software-properties-common
2 $ sudo add-apt-repository -y ppa:ethereum/ethereum
3 $ sudo apt-get update
4 $ sudo apt-get install -y ethereum
```

Em outros Sistemas como o Manjaro:

```
1 [rag@nitro-ryzen ~]$ sudo pacaaur -Ss ethereum
2 community/go-ethereum 1.10.25-1 [instalado]
3   Official Go implementation of the Ethereum protocol
4 [rag@nitro-ryzen ~]$ sudo pacaaur -S go-ethereum
5 [rag@nitro-ryzen ~]$ pacaaur -S go-ethereum
6 resolvendo dependencias...
7 procurando pacotes conflitantes...
8
9 Pacotes (1) go-ethereum-1.10.25-1
10
11 Tamanho total instalado: 197,38 MiB
12 Alteração no tamanho: 0,00 MiB
13
14 :: Continuar a instalação? [S/n]
```

Instruções para outros Sistemas Operacionais podem ser encontradas no site oficial da documentação do Ethereum, artigo [Installing Geth](#).

1.2 Executando o Geth

Executando o Geth diretamente ele irá sincronizar com a rede principal do Ethereum, a mainnet. Será criado o diretório `~/.ethereum` com os dados da rede.

```
1 [rag@nitro-ryzen ~]$ geth
2 INFO [10-20|21:07:12.911] Starting Geth on Ethereum mainnet...
3 INFO [10-20|21:07:12.912] Bumping default cache on mainnet provided=1024 updated=4096
4 INFO [10-20|21:07:12.914] Maximum peer count ETH=50 LES=0 total=50
5 INFO [10-20|21:07:12.915] Smartcard socket not found, disabling err="stat
   /run/pcscd/pcscd.comm: no such file or directory"
6 INFO [10-20|21:07:12.920] Set global gas cap cap=50,000,000
7 INFO [10-20|21:07:12.922] Allocated trie memory caches clean=614.00MiB dirty=1024.00MiB
8 INFO [10-20|21:07:12.923] Allocated cache and file handles
   database=/home/rag/.ethereum/geth/chaindata cache=2.00GiB handles=262,144
9 INFO [10-20|21:07:12.946] Opened ancient database
   database=/home/rag/.ethereum/geth/chaindata/ancient/chain readonly=false
10 INFO [10-20|21:07:12.950]
```

```

11 INFO [10-20|21:07:12.950]
-----
12 INFO [10-20|21:07:12.950] Chain ID: 1 (mainnet)
13 INFO [10-20|21:07:12.950] Consensus: Beacon (proof-of-stake), merged from Ethash
    (proof-of-work)
14 INFO [10-20|21:07:12.950]
15 INFO [10-20|21:07:12.950] Pre-Merge hard forks:
16 INFO [10-20|21:07:12.950] - Homestead: 1150000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead)
17 INFO [10-20|21:07:12.950] - DAO Fork: 1920000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/dao-fork)
18 INFO [10-20|21:07:12.950] - Tangerine Whistle (EIP 150): 2463000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle)
19 INFO [10-20|21:07:12.950] - Spurious Dragon/1 (EIP 155): 2675000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon-1)
20 INFO [10-20|21:07:12.950] - Spurious Dragon/2 (EIP 158): 2675000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon-2)
21 INFO [10-20|21:07:12.950] - Byzantium: 4370000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium)
22 INFO [10-20|21:07:12.950] - Constantinople: 7280000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantinople)
23 INFO [10-20|21:07:12.950] - Petersburg: 7280000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg)
24 INFO [10-20|21:07:12.950] - Istanbul: 9069000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanbul)
25 INFO [10-20|21:07:12.950] - Muir Glacier: 9200000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-glacier)
26 INFO [10-20|21:07:12.950] - Berlin: 12244000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin)
27 INFO [10-20|21:07:12.950] - London: 12965000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/london)
28 INFO [10-20|21:07:12.950] - Arrow Glacier: 13773000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/arrow-glacier)
29 INFO [10-20|21:07:12.950] - Gray Glacier: 15050000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/gray-glacier)
30 INFO [10-20|21:07:12.950]
31 INFO [10-20|21:07:12.950] Merge configured:
32 INFO [10-20|21:07:12.950] - Hard-fork specification:
    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris-spec
33 INFO [10-20|21:07:12.950] - Network known to be merged: true
34 INFO [10-20|21:07:12.950] - Total terminal difficulty: 5875000000000000000000000
35 INFO [10-20|21:07:12.950] - Merge netsplit block: <nil>
36 INFO [10-20|21:07:12.950]
-----
37 INFO [10-20|21:07:12.950]
38 INFO [10-20|21:07:12.952] Disk storage enabled for ethash caches
    dir=/home/rag/.ethereum/ethash count=3
39 INFO [10-20|21:07:12.952] Disk storage enabled for ethash DAGs dir=/home/rag/.ethash
    count=2
40 INFO [10-20|21:07:12.952] Initialising Ethereum protocol network=1 dbversion=8
41 INFO [10-20|21:07:12.963] Loaded most recent local header number=0 hash=d4e567..cb8fa3
    td=17,179,869,184 age=53y6mo3w
42 INFO [10-20|21:07:12.963] Loaded most recent local full block number=0
    hash=d4e567..cb8fa3 td=17,179,869,184 age=53y6mo3w
43 INFO [10-20|21:07:12.963] Loaded most recent local fast block number=0
    hash=d4e567..cb8fa3 td=17,179,869,184 age=53y6mo3w

```

```
44 INFO [10-20|21:07:12.964] Loaded local transaction journal transactions=0 dropped=0
45 INFO [10-20|21:07:12.964] Regenerated local transaction journal transactions=0
    accounts=0
46 INFO [10-20|21:07:12.965] Chain post-merge, sync via beacon client
47 INFO [10-20|21:07:12.965] Gasprice oracle is ignoring threshold set threshold=2
48 WARN [10-20|21:07:12.965] Engine API enabled protocol=eth
49 INFO [10-20|21:07:12.966] Starting peer-to-peer node
    instance=Geth/v1.10.25-stable-69568c55/linux-amd64/go1.19.1
50 INFO [10-20|21:07:12.991] New local node record seq=1,665,519,113,919
    id=da440578e33a2ce7 ip=127.0.0.1 udp=30303 tcp=30303
51 INFO [10-20|21:07:12.992] Started P2P networking
    self=enode://9ae8fcdad4a7243d1bd2308a159c5800ec170e588862be110152627c9ed3fa67376ef8c7526d7a56e9bb
52 INFO [10-20|21:07:12.993] IPC endpoint opened url=/home/rag/.ethereum/geth.ipc
53 INFO [10-20|21:07:12.993] Loaded JWT secret file
    path=/home/rag/.ethereum/geth/jwtsecret crc32=0xdeccafe4
54 INFO [10-20|21:07:12.994] WebSocket enabled url=ws://127.0.0.1:8551
55 INFO [10-20|21:07:12.994] HTTP server started endpoint=127.0.0.1:8551 auth=true
    prefix= cors=localhost vhosts=localhost
56 INFO [10-20|21:07:16.251] New local node record seq=1,665,519,113,920
    id=da440578e33a2ce7 ip=187.95.110.26 udp=2770 tcp=30303
57 INFO [10-20|21:07:22.992] Looking for peers peercount=0 tried=2 static=0
58 INFO [10-20|21:07:32.994] Looking for peers peercount=0 tried=3 static=0
59 INFO [10-20|21:07:43.205] Looking for peers peercount=0 tried=9 static=0
60 WARN [10-20|21:07:47.967] Post-merge network, but no beacon client seen. Please launch
    one to follow the chain!
61 INFO [10-20|21:07:53.281] Looking for peers peercount=0 tried=13 static=0
62 INFO [10-20|21:08:03.346] Looking for peers peercount=0 tried=9 static=0
```

O que vai levar um certo tempo para fazer toda a sincronização da rede completa.

1.3 Redes de Teste

Vimos as redes *Ethereum* disponíveis, uma lista completa pode ser encontrada em <https://ethereum.org/en/developers/docs/networks/>.

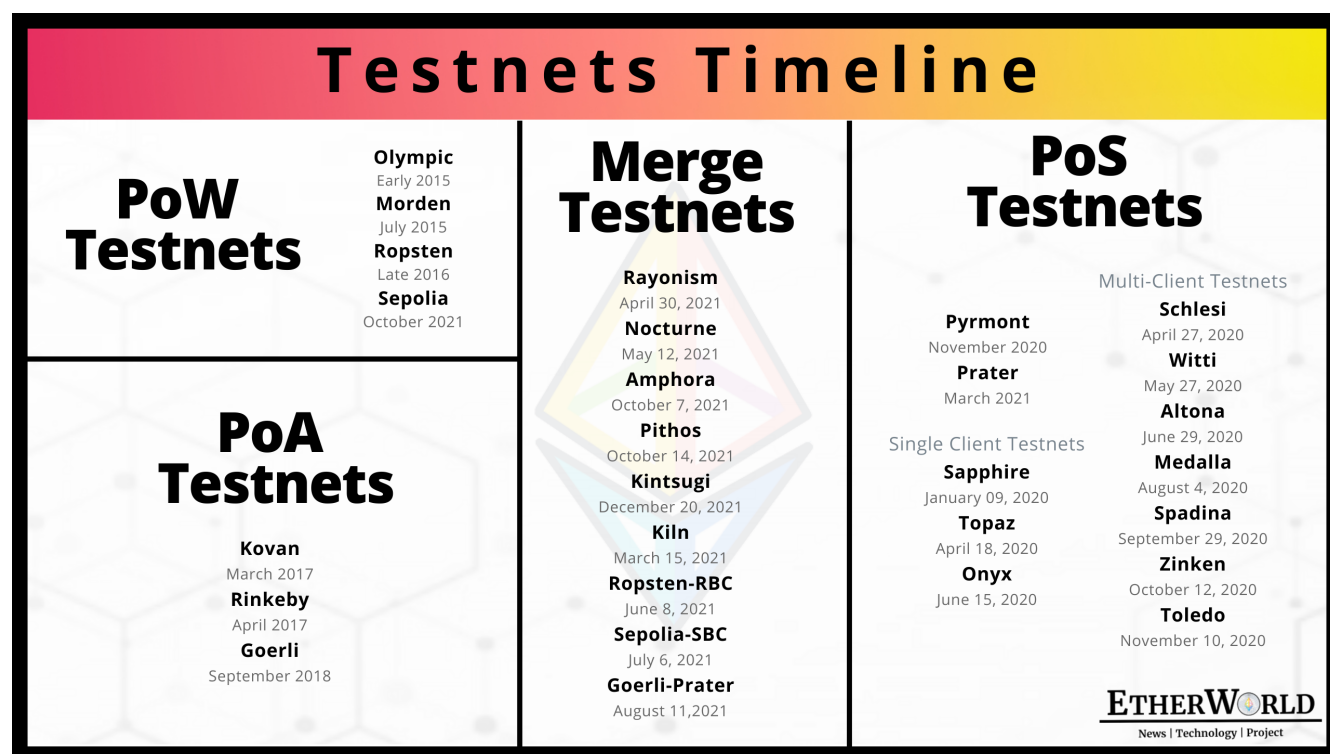


Figura 1: Timeline das Redes de Teste Fonte: Etherwold

A evolução das Redes de Teste pode ser vista nesse artigo [The Evolution of Ethereum Testnets](#).

Atualizar as redes de teste.

Rede	Id	Parâmetro	Descrição
mainnet	1	--mainnet	Rede principal
testnet		--testnet	O livro indica que para as redes de teste deve ser usado o parâmetro --testnet para acessar a rede ropsten por padrão ou fornecer o nome da rede, como --testnet rinkeby. Na versão atual os parâmetros são os seguintes.
ropsten		--ropsten	Ropsten network: pre-configured Proof of Work test network
rinkeby		--rinkeby	Rinkeby network: pre-configured Proof of Authority test network
goerli		--goerli	Görli network: pre-configured Proof of Authority test network
kiln		--kiln	Kiln network: pre-configured proof-of-work to proof-of-stake test network
sepolia		--sepolia	Sepolia network: pre-configured proof-of-work test network

A evolução das Redes de Teste pode ser vista nesse artigo [The Evolution of Ethereum Testnets](#).

1.4 Executando o Geth com uma Rede de Teste

Ao executar o geth com algum das redes de teste será criado um diretório dentro do diretório padrão `~/ethereum/` para cada rede de teste. Por exemplo, `~/ethereum/goerli` para `--goerli` e `~/ethereum/sepolia` para `--sepolia`.

Iremos executar a rede [Sepolia](#) que é recomendada como testnet padrão, uma vez que Goerli será descontinuada em 2023. Sepolia usa um conjunto validador permissionado. É uma rede nova, o que significa que seu estado e seu histórico são pequenos, o que deixa sua sincronização mais rápida e exige menos armazenamento.

Isto é útil para usuários que querem rapidamente montar um nó da rede e interagir com a rede diretamente.

```

1 [rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
  127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
  --keystore ~/.ethereum/sepolia/keystore
2 INFO [04-16|21:27:45.640] Starting Geth on Sepolia testnet...
3 INFO [04-16|21:27:45.641] Maximum peer count ETH=50 LES=0 total=50
4 INFO [04-16|21:27:45.643] Smartcard socket not found, disabling err="stat
  /run/pcscd/pcscd.comm: no such file or directory"
5 INFO [04-16|21:27:45.646] Set global gas cap cap=50,000,000
6 INFO [04-16|21:27:45.649] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
7 INFO [04-16|21:27:45.649] Using leveldb as the backing database
8 INFO [04-16|21:27:45.649] Allocated cache and file handles
  database=/home/rogerio/.ethereum/sepolia/geth/chaindata cache=512.00MiB
  handles=262,144
9 INFO [04-16|21:27:45.660] Using LevelDB as the backing database
10 INFO [04-16|21:27:45.663] Opened ancient database
  database=/home/rogerio/.ethereum/sepolia/geth/chaindata/ancient/chain
  readonly=false
11 INFO [04-16|21:27:45.664] Disk storage enabled for ethash caches
  dir=/home/rogerio/.ethereum/sepolia/geth/ethash count=3
12 INFO [04-16|21:27:45.664] Disk storage enabled for ethash DAGs
  dir=/home/rogerio/.ethash count=2
13 INFO [04-16|21:27:45.664] Initialising Ethereum protocol network=11,155,111 dbversion=8
14 INFO [04-16|21:27:45.667]
15 INFO [04-16|21:27:45.667]
-----
16 INFO [04-16|21:27:45.667] Chain ID: 11155111 (sepolia)
17 INFO [04-16|21:27:45.667] Consensus: Beacon (proof-of-stake), merged from Ethash
  (proof-of-work)
18 INFO [04-16|21:27:45.667]
19 INFO [04-16|21:27:45.667] Pre-Merge hard forks (block based):
20 INFO [04-16|21:27:45.667] - Homestead: #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead)
21 INFO [04-16|21:27:45.667] - Tangerine Whistle (EIP 150): #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle)
22 INFO [04-16|21:27:45.667] - Spurious Dragon/1 (EIP 155): #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon-1)
23 INFO [04-16|21:27:45.667] - Spurious Dragon/2 (EIP 158): #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon-2)
24 INFO [04-16|21:27:45.667] - Byzantium: #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium)
25 INFO [04-16|21:27:45.667] - Constantinople: #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantinople)
26 INFO [04-16|21:27:45.667] - Petersburg: #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg)

```



```

27 INFO [04-16|21:27:45.667] - Istanbul: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg)
28 INFO [04-16|21:27:45.667] - Muir Glacier: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanbul)
29 INFO [04-16|21:27:45.667] - Berlin: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-glacier)
30 INFO [04-16|21:27:45.667] - London: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin)
31 INFO [04-16|21:27:45.667]
32 INFO [04-16|21:27:45.667] Merge configured:
33 INFO [04-16|21:27:45.667] - Hard-fork specification:
    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.
34 INFO [04-16|21:27:45.667] - Network known to be merged: true
35 INFO [04-16|21:27:45.667] - Total terminal difficulty: 17000000000000000
36 INFO [04-16|21:27:45.667] - Merge netsplit block: #1735371
37 INFO [04-16|21:27:45.667]
38 INFO [04-16|21:27:45.667] Post-Merge hard forks (timestamp based):
39 INFO [04-16|21:27:45.667] - Shanghai: @1677557088
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/shanghai)
40 INFO [04-16|21:27:45.667]
41 INFO [04-16|21:27:45.667]
-----
42 INFO [04-16|21:27:45.667]
43 INFO [04-16|21:27:45.668] Loaded most recent local block number=0 hash=25a5cc..3e6dd9
    td=131,072 age=1y6mo2w
44 INFO [04-16|21:27:45.668] Loaded local transaction journal transactions=0 dropped=0
45 INFO [04-16|21:27:45.668] Regenerated local transaction journal transactions=0
    accounts=0
46 INFO [04-16|21:27:45.668] Chain post-merge, sync via beacon client
47 INFO [04-16|21:27:45.669] Gasprice oracle is ignoring threshold set threshold=2
48 WARN [04-16|21:27:45.669] Engine API enabled protocol=eth
49 INFO [04-16|21:27:45.670] Starting peer-to-peer node
    instance=Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
50 INFO [04-16|21:27:45.681] New local node record seq=1,681,600,532,512
    id=05b2a1f369961544 ip=127.0.0.1 udp=30303 tcp=30303
51 INFO [04-16|21:27:45.682] Started P2P networking
    self=enode://69520d55cfffcd600a5a46f15c8255ad6831dbe9cb11f73b6aef0847a5e048b99043950daede9c4a374a3
52 INFO [04-16|21:27:45.682] IPC endpoint opened
    url=/home/rogerio/.ethereum/sepolia/geth.ipc
53 INFO [04-16|21:27:45.683] Loaded JWT secret file
    path=/home/rogerio/.ethereum/sepolia/geth/jwtsecret crc32=0xa99ff4fd
54 INFO [04-16|21:27:45.683] HTTP server started endpoint=127.0.0.1:8559 auth=false
    prefix= cors= vhosts=localhost
55 INFO [04-16|21:27:45.686] WebSocket enabled url=ws://127.0.0.1:8551
56 INFO [04-16|21:27:45.686] HTTP server started endpoint=127.0.0.1:8551 auth=true
    prefix= cors=localhost vhosts=localhost
57 WARN [04-16|21:27:55.489] System clock seems off by -25.06956443s, which can prevent
    network connectivity
58 WARN [04-16|21:27:55.490] Please enable network time synchronisation in system
    settings.
59 WARN [04-16|21:28:20.672] Post-merge network, but no beacon client seen. Please launch
    one to follow the chain!

```

Note que está sendo gerada uma mensagem de aviso:


```
1 WARN [04-16|21:28:20.672] Post-merge network, but no beacon client seen. Please launch
   one to follow the chain!
```

Pesquisando na Internet por “*Post-merge network, but no beacon client seen. Please launch one to follow the chain!*”, encontramos essa solução: <https://github.com/ethereum/go-ethereum/issues/25791>.

A documentação do *Ethereum* sobre *Consensus Clients*, mostra que geth deve ser iniciado, com conexão RPC autenticada usando um arquivo jwtsecret. Por padrão esse arquivo está em `~/.ethereum/geth/jwtsecret`.

```
1 [rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
   127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
   --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
   8551 --authrpc.vhosts localhost --authrpc.jwtsecret
   ~/.ethereum/sepolia/geth/jwtsecret
2 INFO [04-16|21:27:45.640] Starting Geth on Sepolia testnet...
```

- Note que estou executando na rede de testes Sepolia, opção na minha versão do geth é diferente do livro. No livro ele diz para usar o parâmetro `--testnet` que por padrão usa a rede de testes Ropsten, na minha instalação não suporta mais o parâmetro `--ropsten`, tem o `--mainnet` e outras redes de testes.

1.5 Criando Contas na Rede

O comando `geth account new` cria uma nova conta. Indicaremos que a conta é para ser criada na rede de teste `--sepolia`.

```
1 [rogerio@ryzen-nitro execution]$ geth --sepolia account new
2 INFO [04-16|21:50:49.343] Maximum peer count ETH=50 LES=0 total=50
3 INFO [04-16|21:50:49.344] Smartcard socket not found, disabling err="stat
   /run/pcscd/pcscd.comm: no such file or directory"
4 Your new account is locked with a password. Please give a password. Do not forget this
   password.
5 Password:
6 Repeat password:
7
8 Your new key was generated
9
10 Public address of the key: 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95
11 Path of the secret key file:
   /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-51-24.036052785Z--a9e98368b44b371ce7
12
13 - You can share your public address with anyone. Others need it to interact with you.
14 - You must NEVER share the secret key with anyone! The key controls access to your
   funds!
15 - You must BACKUP your key file! Without the key, it's impossible to access account
   funds!
16 - You must REMEMBER your password! Without the password, it's impossible to decrypt
   the key!
```

A conta com a chave pública `0xa9e98368B44b371ceC7d205F9fE2b074b6134C95` foi criada. Iremos criar uma segunda conta.

```

1 [rogerio@ryzen-nitro execution]$ geth --sepolia account new
2 INFO [04-16|21:52:58.948] Maximum peer count ETH=50 LES=0 total=50
3 INFO [04-16|21:52:58.948] Smartcard socket not found, disabling err="stat
  /run/pcscd/pcscd.comm: no such file or directory"
4 Your new account is locked with a password. Please give a password. Do not forget this
  password.
5 Password:
6 Repeat password:
7
8 Your new key was generated
9
10 Public address of the key: 0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab
11 Path of the secret key file:
    /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852a26bedec5b
12
13 - You can share your public address with anyone. Others need it to interact with you.
14 - You must NEVER share the secret key with anyone! The key controls access to your
    funds!
15 - You must BACKUP your key file! Without the key, it's impossible to access account
    funds!
16 - You must REMEMBER your password! Without the password, it's impossible to decrypt
    the key!
17
18 [rogerio@ryzen-nitro execution]$

```

É para termos as duas contas 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95 e 0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab criadas com seus arquivos em /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--a9e98368b44b371ceC7d205F9fE2b074b6134C95 e /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852a26bedec5bd457b88c031c46a622f4ab.

1.6 Listando as Contas

As contas existentes ou que foram criadas podem ser listadas com o comando `geth account list`. Utilizaremos novamente o parâmetro que indica a rede de teste `--sepolia`.

```

1 [rogerio@ryzen-nitro execution]$ geth --sepolia account list
2 INFO [04-16|21:56:11.129] Maximum peer count ETH=50 LES=0 total=50
3 INFO [04-16|21:56:11.130] Smartcard socket not found, disabling err="stat
  /run/pcscd/pcscd.comm: no such file or directory"
4 INFO [04-16|21:56:11.130] Set global gas cap cap=50,000,000
5 Account #0: {a9e98368b44b371cec7d205f9fe2b074b6134c95}
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-51-24.036052785Z--a9e98368b44b371cec7d205f9fe2b074b6134c95
6 Account #1: {c061b852a26bedec5bd457b88c031c46a622f4ab}
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852a26bedec5bd457b88c031c46a622f4ab
7 [rogerio@ryzen-nitro execution]$

```

A documentação do `geth`, bem como comandos e parâmetros podem ser acessados em <https://geth.ethereum.org/docs>.

Executando com opção de responder a comandos via RPC. A documentação desta parte está disponível em <https://geth.ethereum.org/docs/rpc/server>.

```

1 [rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
  127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
  --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
  8551 --authrpc.vhosts localhost --authrpc.jwtsecret

```

```
~/ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15
```

1.7 Clientes de Consenso

Para terminar a configuração é necessário instalar algum cliente de consenso. Existem atualmente cinco clientes de consenso que podem ser executado em conjunto com o Geth:

- **Lighthouse**: escrito em Rust.
- **Nimbus**: escrito em Nim.
- **Prysm**: escrito em Go.
- **Teku**: escrito em Java.
- **Lodestar**: escrito em Typescript.

Por ser escrito na linguagem Go, assim como geth, testaremos o Prysm. O Prysm é uma implementação da especificação do consenso proof-of-stake do Ethereum.

A configuração do Prysm pode ser feita conforme descrito no material <https://docs.prylabs.network/docs/install/install-with-script>.

O material ensina a usar o Prysm para executar um nó *Ethereum*, portanto resolver o problema apresentado após a atualização do Merge e opcionalmente como um validador (*validator*).

1.8 Instalando o cliente de consenso Prysm¹

Para a instalação do Prysm, crie no diretório `~/ethereum/<rede>`, duas subpastas: `consensus` e `execution`. Acesse o diretório `consensus` e execute o comando para baixar o cliente Prysm e transformá-lo em executável:

```
1 $ mkdir prysm && cd prysm
2 $ curl https://raw.githubusercontent.com/prysmaticlabs/prysm/master/prysm.sh --output
   prysm.sh && chmod +x prysm.sh
```

1.9 Gerando um arquivo JWT Secret

- A conexão HTTP entre seu nó beacon e seu nó de execução precisa ser autenticada usando um *token* JWT. Existem diversas formas de gerar este *token*:
 - Usando um gerado *on line* como [este](#). Copie e cole o valor gerado dentro do arquivo `jwt.hex`.
 - Usando OpenSSL para criar o *token* via comando: `openssl rand -hex 32 | tr -d "\n" > "jwt.hex"`.
 - Usar o que foi gerado pelo cliente de execução geth: `~/ethereum/<<rede>>/geth/jwtsecret`.
 - Usar o próprio Prysm para gerar o `jwt.hex`:

¹Step 2: Install Prysm: <https://docs.prylabs.network/docs/install/install-with-script#step-2-install-prysm>

```

1 ## Optional. This command is necessary only if you've previously configured
  USE_PRYSM_VERSION
2 USE_PRYSM_VERSION=v4.0.0
3
4 ## Required.
5 ./prysm.sh beacon-chain generate-auth-secret

```

Nesta opção o Prysm irá mostrar o caminho onde o arquivo `jwt.hex` foi gerado.

1.10 Executando um Cliente de Execução²

(<https://docs.prylabs.network/docs/install/install-with-script#step-3-run-an-execution-client>)

Nesta etapa, você instalará um cliente de camada de execução (geth), se ainda não instalou, ao qual o nó beacon do Prysm se conectará.

Baixe e execute o a última versão 64-bit estável do **Geth installer** para seu Sistema Operacional do site [Geth downloads page](#).

Verifique se a versão mais nova já não foi instalada. O link [v1.11.5](#) lista as versões.

- Tenho instalado a versão 1.11.5-stable-a38f4108, pode verificar a sua com `geth --version` ou `geth version`:

```

1 [rogerio@ryzen-nitro execution]$ geth --version
2 geth version 1.11.5-stable-a38f4108
3 [rogerio@ryzen-nitro execution]$ geth version
4 Geth
5 Version: 1.11.5-stable
6 Git Commit: a38f4108571d1a144dc3cf3faf8990430d109bc4
7 Git Commit Date: 20230321
8 Architecture: amd64
9 Go Version: go1.20.2
10 Operating System: linux
11 GOPATH=
12 GOROOT=
13 [rogerio@ryzen-nitro execution]$

```

Navegue até o diretório `execution` e execute o comando para inicial o nó de execução. O comando padrão da documentação:

```
1 $ geth --http --http.api "eth,net,engine,admin" --authrpc.jwtsecret /path/to/jwt.hex
```

Comando que estou utilizando neste exemplo:

```

1 [rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
  127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
  --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
  8551 --authrpc.vhosts localhost --authrpc.jwtsecret
  ~/.ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15

```

²Step 3: Run an execution client: <https://docs.prylabs.network/docs/install/install-with-script#step-3-run-an-execution-client>

Veja as [Opções de linha de comando do Geth](#) para a definição de parâmetros.

Dependendo das opções a Sincronização pode levar um longo tempo, de horas até dias. Enquanto sincroniza, pode ir fazendo o próximo passo.

Congratulations

Você está agora executando um **nó de execução** na camada de execução da *Ethereum*.

1.11 Executando um nó beacon usando Prysm³

Use o comando para iniciar um nó beacon que conecta no seu nó de execução local:

```
1 ./prysm.sh beacon-chain --execution-endpoint=http://localhost:8551
  --jwt-secret=~/.ethereum/geth/jwtsecret --suggested-fee-recipient=<hash da minha
  conta>
```

Alterei o comando padrão para o conter o *hash* de uma das minhas contas e pelo tipo de rede de teste *sepolia* precisa de um *genesis state* para iniciar. O arquivo pode ser baixado de <https://github.com/eth-clients/merge-testnets/blob/main/sepolia/genesis.ssz> e deve ser colocado no diretório `consensus/prysm`.

```
1 [rogerio@ryzen-nitro prysm]$ wget
  https://github.com/eth-clients/merge-testnets/blob/main/sepolia/genesis.ssz -o
  genesis-sepolia.ssz
```

Então o comando pode ser utilizado para iniciar um nó *beacon* que conecta seu nó de execução local.

```
1 [rogerio@ryzen-nitro prysm]$ ./prysm.sh beacon-chain
  --execution-endpoint=http://localhost:8551 --sepolia
  --jwt-secret=~/.ethereum/sepolia/geth/jwtsecret
  --genesis-state=genesis-sepolia.ssz
  --suggested-fee-recipient=0xa9e98368b44b371cec7d205f9fe2b074b6134c95
```

1.12 Executando o Console JavaScript

O console Javascript pode também ser conectado ao nó Geth usando IPC. Quando o Geth é iniciado, um arquivo `geth.ipc` é criado automaticamente e salvo no diretório de dados. Este arquivo ou um caminho customizado para um arquivo IPC pode ser passado para o Geth usando o parâmetro `attach`:

```
1 [rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
```

³Step 4: Run a beacon node using Prysm: <https://docs.prylabs.network/docs/install/install-with-script#step-4-run-a-beacon-node-using-prysm>

```
6 datadir: /home/rogerio/.ethereum/sepolia
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
   txpool:1.0 web3:1.0
8
9 To exit, press ctrl-d or type exit
10 >
```

1.13 Verificação do Funcionamento da Rede

Para verificar o funcionamento da rede, utilize o comando `net.listening`:

```
1 [rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
6 datadir: /home/rogerio/.ethereum/sepolia
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
   txpool:1.0 web3:1.0
8
9 To exit, press ctrl-d or type exit
10 > net.listening
11 true
12 >
```

A mesma verificação pode ser feita via API JSON RPC no terminal do sistema:

```
1 curl -X POST --insecure --header "Content-Type: application/json" --data
   '{"jsonrpc":"2.0", "method":"net_listening","params":[], "id":64}' --location
   http://localhost:8559
2 {'jsonrpc':"2.0", "id":64, "result":true}
```

1.14 Listando as Contas pelo Console

A lista de contas pode ser recuperada através dos comandos no console `eth.accounts` e por RPC `{"method": "eth_accounts", "params": []}`. Em versões anteriores do geth era possível utilizar `personal.listAccounts`, mas `personal` foi [depreciado](#).

```
1 [rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
6 datadir: /home/rogerio/.ethereum/sepolia
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
   txpool:1.0 web3:1.0
8
9 To exit, press ctrl-d or type exit
10 > net.listening
11 true
12 > personal.listAccounts
13 ReferenceError: personal is not defined
```

```

14         at <eval>:1:1(0)
15
16 > eth.accounts
17 ["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]

```

O mesmo resultado pode ser obtido via comando `curl` no terminal para listar as contas usando RPC:

```

1 [rogerio@ryzen-nitro execution]$ curl -X POST --insecure --header "Content-Type:
    application/json" --data '{"jsonrpc":"2.0","method":"eth_accounts","params":[],
    "id":64}' --location http://localhost:8559
2 {"jsonrpc":"2.0","id":64,"result":["0xa9e98368b44b371cec7d205f9fe2b074b6134c95","0xc061b852a26bedec5b

```

Outros comandos podem ser executados da mesma maneira via Console JavaScript ou invocação RPC através do `curl`. Uma lista completa de comandos da API RPC estão disponíveis em [JSON-RPC API](#).

Como por exemplo, recuperar a versão do cliente `geth`:

```

1 [[rogerio@ryzen-nitro execution]$ curl -X POST --insecure --header "Content-Type:
    application/json" --data
    '{"jsonrpc":"2.0","method":"web3_clientVersion","params":[],"id":67}' --location
    http://localhost:8559
2 {"jsonrpc":"2.0","id":67,"result":"Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2"}

```

Ou a versão da Rede:

```

1 [rogerio@ryzen-nitro execution]$ curl -X POST --header "Content-Type:
    application/json" --data
    '{"jsonrpc":"2.0","method":"net_version","params":[],"id":67}' --location
    http://localhost:8559
2 {"jsonrpc":"2.0","id":67,"result":"11155111"}

```

O valor 11155111 em `result` indica que estamos executando sobre a rede Sepolia. Uma lista completa com os IDs das redes está disponível em <https://chainlist.org>. Os mais comuns são 1: *Ethereum Mainnet*, 2: *Morden testnet* (depreciado), 3: *Ropsten testnet*, 4: *Rinkeby testnet*, 5: *Goerli testnet* (será depreciada em 2023).

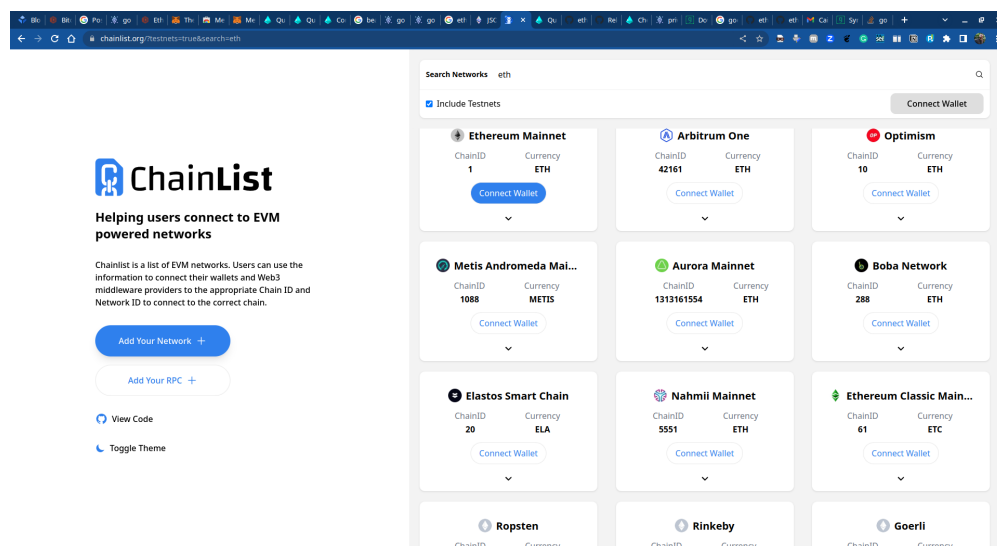


Figura 2: Lista de IDs Fonte: chainlist

1.15 Teste de transferência de valores entre carteiras

Corrigir os endereços.

Podemos testar uma transferência de valores solicitando valores para Faucets, utilizaremos o <https://faucet.sepolia.dev/> e faremos transferência de uma conta para outra e verificar se a transação aparece na rede de teste.

Verificação dos valores em cada carteira:

```
1 [rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
6 datadir: /home/rogerio/.ethereum/sepolia
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
      txpool:1.0 web3:1.0
8
9 To exit, press ctrl-d or type exit
10 > eth.accounts
11 ["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]
12 > eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95")
13 0
14 > eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab")
15 0
16 > web3.fromWei(eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95"), "ether")
17 0
18 > web3.fromWei(eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab"), "ether")
19 0
20 > web3.fromWei(eth.getBalance(eth.coinbase), "ether")
21 Error: etherbase must be explicitly specified
22     at web3.js:6365:9(39)
23     at get (web3.js:6265:66(14))
24     at <eval>:1:33(5)
```

```
25
26 > eth.coinbase
27 Error: etherbase must be explicitly specified
28     at web3.js:6365:9(39)
29     at get (web3.js:6265:66(14))
30     at <eval>:1:5(1)
31
32 > eth.coinbase.panic: Error: etherbase must be explicitly specified at
    web3.js:6365:9(39)
33
34 goroutine 82 [running]:
35 github.com/dop251/goja.(*baseJsFuncObject)._call(...)
36     github.com/dop251/goja@v0.0.0-20230122112309-96b1610dd4f7/func.go:396
```

- Enviar 100 ethers da primeira para a segunda carteira:

```
1 > eth.sendTransaction({from: "0xc4d51e4424c6d1e67f61742c63f0f42c1c3b1fb3", to:
    "0xbe6e6f1a8273286242efad565f63c2fb2ad08308", value: 100})
2
3 > eth.sendTransaction({from: "0xc4d51e4424c6d1e67f61742c63f0f42c1c3b1fb3", to:
    "0xbe6e6f1a8273286242efad565f63c2fb2ad08308", value: web3.toWei(100, "ether") })
```

1.16 Leitura Recomendada

Leitura Recomendada

Capítulo 11: Ethereum 101

Livro: IMRAN BASHIR. Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

Capítulo 12: Futher Ethereum

Livro: IMRAN BASHIR. Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

2 Prática: Criando uma Rede Ethereum Privada

O objetivo dessa prática é criarmos uma Rede *Ethereum* Privada.

2.1 Criando uma Rede Privada Local

Para a criação de uma nova Rede Privada Local é necessário fazermos algumas configurações. Precisamos criar um diretório `mkdir ~/.etherprivate` para ser a base de armazenamento para a nova rede. Temos que fornecer a configurações iniciais para a nova rede, criando um arquivo `privategenesis.json` em `~/.etherprivate`.

Passos

1. Criar um diretório `mkdir ~/.etherprivate`
2. Criar um arquivo `privategenesis.json` em `~/.etherprivate`.

O conteúdo do arquivo `privategenesis.json` deve ser o listado no Código 1.

```
1 {
2   "nonce": "0x0000000000000042",
3   "timestamp": "0x00",
4   "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
5   "extraData": "0x00",
6   "gasLimit": "0x8000000",
7   "difficulty": "0x0400",
8   "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
9   "coinbase": "0x3333333333333333333333333333333333333333333333333333333333333333",
10  "alloc": {},
11  "config": {
12    "chainId": 786,
13    "homesteadBlock": 0,
14    "eip150Block": 0,
15    "eip155Block": 0,
16    "eip158Block": 0
17  }
18 }
```

Código 1: Genesis File

Após a configuração inicial, o `geth` é utilizado para a criação e inicialização da nova Rede. O `geth` deve ser executado com os parâmetros `--datadir`, indicando o diretório onde os dados da nova rede serão armazenados e com o `init` indicando o caminho para o *genesis file*, conforme Código 2.

```
1 [rag@nitro-ryzen ~]$ geth --datadir ~/.etherprivate init
   ~/.etherprivate/privategenesis.json
2 INFO [10-27|19:59:19.049] Maximum peer count ETH=50 LES=0 total=50
3 INFO [10-27|19:59:19.051] Smartcard socket not found, disabling err="stat
   /run/pcscd/pcscd.comm: no such file or directory"
4 INFO [10-27|19:59:19.053] Set global gas cap cap=50,000,000
5 INFO [10-27|19:59:19.054] Allocated cache and file handles
   database=/home/rag/.etherprivate/geth/chaindata cache=16.00MiB handles=16
```

```

6 INFO [10-27|19:59:19.068] Opened ancient database
  database=/home/rag/.etherprivate/geth/chaindata/ancient/chain readonly=false
7 INFO [10-27|19:59:19.068] Writing custom genesis block
8 INFO [10-27|19:59:19.068] Persisted trie from memory database nodes=0 size=0.00B
  time="8.101us" gcnodes=0 gcsizes=0.00B gctime=0s livenodes=1 livesize=0.00B
9 INFO [10-27|19:59:19.069] Successfully wrote genesis state database=chaindata
  hash=6650a0..b5c158
10 INFO [10-27|19:59:19.069] Allocated cache and file handles
  database=/home/rag/.etherprivate/geth/lightchaindata cache=16.00MiB handles=16
11 INFO [10-27|19:59:19.080] Opened ancient database
  database=/home/rag/.etherprivate/geth/lightchaindata/ancient/chain readonly=false
12 INFO [10-27|19:59:19.081] Writing custom genesis block
13 INFO [10-27|19:59:19.081] Persisted trie from memory database nodes=0 size=0.00B
  time="7.613us" gcnodes=0 gcsizes=0.00B gctime=0s livenodes=1 livesize=0.00B
14 INFO [10-27|19:59:19.082] Successfully wrote genesis state database=lightchaindata
  hash=6650a0..b5c158
15 [rag@nitro-ryzen ~]$

```

Código 2: Inicialização da Rede Privada Local

2.2 Executando a nova Rede

O mesmo cliente Geth pode ser iniciado, executando com base na nova rede criada.

```

1 [rogerio@ryzen-nitro .etherprivate]$ geth --datadir ~/.etherprivate/ --syncmode full
  --allow-insecure-unlock --networkid 786 --http --http.addr 127.0.0.1 --http.port
  8559 --http.api "eth,net,web3,personal,engine,admin,debug" --keystore
  ~/.etherprivate/keystore --authrpc.addr localhost --authrpc.port 8551
  --authrpc.vhosts localhost --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret
  --nodiscover --maxpeers 15
2 INFO [04-17|09:14:28.084] Maximum peer count ETH=15 LES=0 total=15
3 INFO [04-17|09:14:28.085] Smartcard socket not found, disabling err="stat
  /run/pcscd/pcscd.comm: no such file or directory"
4 INFO [04-17|09:14:28.087] Set global gas cap cap=50,000,000
5 INFO [04-17|09:14:28.089] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
6 INFO [04-17|09:14:28.089] Using leveldb as the backing database
7 INFO [04-17|09:14:28.089] Allocated cache and file handles
  database=/home/rogerio/.etherprivate/geth/chaindata cache=512.00MiB handles=262,144
8 INFO [04-17|09:14:28.096] Using LevelDB as the backing database
9 INFO [04-17|09:14:28.097] Opened ancient database
  database=/home/rogerio/.etherprivate/geth/chaindata/ancient/chain readonly=false
10 INFO [04-17|09:14:28.097] Disk storage enabled for ethash caches
  dir=/home/rogerio/.etherprivate/geth/ethash count=3
11 INFO [04-17|09:14:28.097] Disk storage enabled for ethash DAGs
  dir=/home/rogerio/.ethash count=2
12 INFO [04-17|09:14:28.097] Initialising Ethereum protocol network=786 dbversion=8
13 INFO [04-17|09:14:28.098]
14 INFO [04-17|09:14:28.098]
-----
15 INFO [04-17|09:14:28.098] Chain ID: 786 (unknown)
16 INFO [04-17|09:14:28.098] Consensus: unknown
17 INFO [04-17|09:14:28.098]
18 INFO [04-17|09:14:28.098] Pre-Merge hard forks (block based):
19 INFO [04-17|09:14:28.098] - Homestead: #0
  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead)

```

```

20 INFO [04-17|09:14:28.098] - Tangerine Whistle (EIP 150): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine
21 INFO [04-17|09:14:28.098] - Spurious Dragon/1 (EIP 155): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious
22 INFO [04-17|09:14:28.098] - Spurious Dragon/2 (EIP 158): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious
23 INFO [04-17|09:14:28.098] - Byzantium: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium
24 INFO [04-17|09:14:28.098] - Constantinople: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantin
25 INFO [04-17|09:14:28.098] - Petersburg: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersbur
26 INFO [04-17|09:14:28.098] - Istanbul: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanbul
27 INFO [04-17|09:14:28.098] - Berlin: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin
28 INFO [04-17|09:14:28.098] - London: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/london
29 INFO [04-17|09:14:28.098]
30 INFO [04-17|09:14:28.098] The Merge is not yet available for this network!
31 INFO [04-17|09:14:28.098] - Hard-fork specification:
    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.m
32 INFO [04-17|09:14:28.098]
33 INFO [04-17|09:14:28.098] Post-Merge hard forks (timestamp based):
34 INFO [04-17|09:14:28.098]
35 INFO [04-17|09:14:28.098]
-----
36 INFO [04-17|09:14:28.098]
37 INFO [04-17|09:14:28.098] Loaded most recent local block number=0 hash=6650a0..b5c158
    td=1024 age=54y3w3d
38 INFO [04-17|09:14:28.099] Loaded local transaction journal transactions=0 dropped=0
39 INFO [04-17|09:14:28.099] Regenerated local transaction journal transactions=0
    accounts=0
40 INFO [04-17|09:14:28.101] Gasprice oracle is ignoring threshold set threshold=2
41 WARN [04-17|09:14:28.101] Engine API enabled protocol=eth
42 WARN [04-17|09:14:28.101] Engine API started but chain not configured for merge yet
43 INFO [04-17|09:14:28.101] Starting peer-to-peer node
    instance=Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
44 INFO [04-17|09:14:28.111] IPC endpoint opened url=/home/rogerio/.etherprivate/geth.ipc
45 INFO [04-17|09:14:28.112] Loaded JWT secret file
    path=/home/rogerio/.etherprivate/geth/jwtsecret crc32=0x612a2337
46 INFO [04-17|09:14:28.112] New local node record seq=1,681,733,283,410
    id=ffcbe33c92bbd417 ip=127.0.0.1 udp=0 tcp=30303
47 INFO [04-17|09:14:28.112] Started P2P networking
    self="enode://4e6d847eea6db022f2a8a44453f4009a52141bc87eae8204f84d968e91f18a60c9ed3f60890e4e6d189
48 INFO [04-17|09:14:28.112] HTTP server started endpoint=127.0.0.1:8559 auth=false
    prefix= cors= vhosts=localhost
49 INFO [04-17|09:14:28.113] WebSocket enabled url=ws://127.0.0.1:8551
50 INFO [04-17|09:14:28.113] HTTP server started endpoint=127.0.0.1:8551 auth=true
    prefix= cors=localhost vhosts=localhost

```

2.3 Interagindo com a nova Rede

O console pode ser utilizado na interação com a instância da nova rede em execução.

```
1 $ geth attach ~/.etherprivate/geth.ipc
```

2.4 Criando contas na nova Rede

Vamos criar duas contas para testes na nova rede. Utilizaremos para fins de teste a senha “admin12345”. O Código abaixo mostra a execução quando era possível utilizar o método `personal.newAccount(...)` via console JavaScript.

```
1 > personal.newAccount("admin12345")
2 "0xedbc36d74d5a1cd64db36e53798bd1781f0c4955"
3
4 > eth.accounts
5 ["0xedbc36d74d5a1cd64db36e53798bd1781f0c4955"]
6
7 > personal.newAccount("admin12345")
8
9 "0x1478d95f8754b3ba7127100dd0bb46578fe7d22a"
10
11 > eth.accounts
12 ["0xedbc36d74d5a1cd64db36e53798bd1781f0c4955",
    "0x1478d95f8754b3ba7127100dd0bb46578fe7d22a"]
```

Com a depreciação do `personal` é recomendado utilizar o `clef` com o parâmetro `newaccount`. É importante indicar o diretório `keystore` do `.etherprivate`.

```
1 [rogerio@ryzen-nitro .etherprivate]$ clef newaccount --keystore keystore
2
3 WARNING!
4
5 Clef is an account management tool. It may, like any software, contain bugs.
6
7 Please take care to
8 - backup your keystore files,
9 - verify that the keystore(s) can be opened with your password.
10
11 Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
12 without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
13 PURPOSE. See the GNU General Public License for more details.
14
15 Enter 'ok' to proceed:
16 > ok
17
18 ## New account password
19
20 Please enter a password for the new account to be created (attempt 0 of 3)
21 >
22 -----
23 INFO [04-17|09:22:12.882] Your new key was generated
    address=0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
24 WARN [04-17|09:22:12.892] Please backup your key file!
    path=/home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e44b03b37755
25 WARN [04-17|09:22:12.892] Please remember your password!
26 Generated account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
```

A conta 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 foi gerada, utilizei a senha admin12345.

```

1 [rogerio@ryzen-nitro .etherprivate]$ clef newaccount --keystore keystore
2
3 WARNING!
4
5 Clef is an account management tool. It may, like any software, contain bugs.
6
7 Please take care to
8 - backup your keystore files,
9 - verify that the keystore(s) can be opened with your password.
10
11 Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
12 without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
13 PURPOSE. See the GNU General Public License for more details.
14
15 Enter 'ok' to proceed:
16 > ok
17
18 ## New account password
19
20 Please enter a password for the new account to be created (attempt 0 of 3)
21 >
22 -----
23 INFO [04-17|09:28:56.607] Your new key was generated
24   address=0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
25 WARN [04-17|09:28:56.607] Please backup your key file!
26   path=/home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad451d2865a2
27 WARN [04-17|09:28:56.607] Please remember your password!
28 Generated account 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
29 [rogerio@ryzen-nitro .etherprivate]$

```

Neste ponto é para termos duas contas criadas, que na minha máquina são: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 e 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c, ambas com a senha admin12345.

Verifiquemos no console JavaScript se elas são listadas:

```

1 [rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
6 datadir: /home/rogerio/.etherprivate
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
8   txpool:1.0 web3:1.0
9
10 To exit, press ctrl-d or type exit
11 > eth.accounts
12 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
13   "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]

```


2.5 Verificando o saldo das carteiras

Vamos verificar os valores em cada uma das carteiras:

```
1 [rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
6 datadir: /home/rogerio/.etherprivate
7 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
8          txpool:1.0 web3:1.0
9
10 To exit, press ctrl-d or type exit
11 > eth.accounts
12 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
13  "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
14 >
15 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
16 0
17 > web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
18 0
19 >
```

2.6 Gerar algum saldo para as carteiras

Para acumular algum valor é necessário minerar. Vamos iniciar o geth indicando a carteira que irá receber as recompensas pela mineração utilizando o parâmetro `--miner.etherbase`, utilizei aqui a primeira conta criada `0x2db017e44b03b37755a4b15e14cd799f83de4c13`. Note que é possível colocar uma identificação para sua rede com `--identity "RAGPrivateEtherem"`.

```
1 [rogerio@ryzen-nitro .etherprivate]$ geth --datadir ~/.etherprivate/ --syncmode full
2 --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
3 --http.addr 127.0.0.1 --http.port 8559 --http.api
4 "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
5 --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
6 --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
7 --miner.etherbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
```

O cliente geth irá iniciar normalmente.

Em um console JavaScript vamos verificar o saldo inicial e iniciar a mineração.

```
1 [rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/RAGPrivateEtherem/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 coinbase: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
6 at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
7 datadir: /home/rogerio/.etherprivate
8 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
9          txpool:1.0 web3:1.0
10
11 To exit, press ctrl-d or type exit
```

```
11 > eth.accounts
12
13 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
14 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
15 0
16 > miner.start()
17
18 null
19 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
20 2890
21 >
```

2.7 Transferências entre as carteiras

Vamos enviar 100 *ethers* da primeira para a segunda carteira.

```
1 [rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/RAGPrivateEthereum/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
5 coinbase: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
6 at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
7 datadir: /home/rogerio/.etherprivate
8 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
          txpool:1.0 web3:1.0
9
10 To exit, press ctrl-d or type exit
11 > eth.accounts
12
13 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
14 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
15 0
16 > miner.start()
17
18 null
19 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
20 2890
21 > miner.stop()
22
23 null
24 > web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
25 0
26 > web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
27 0
28 > web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
29 4335
30 > eth.sendTransaction({from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13", to:
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c", value: 100})
31 Error: authentication needed: password or unlock
32   at web3.js:6365:9(39)
33   at send (web3.js:5099:62(29))
34   at <eval>:1:20(9)
```

```

35
36 > personal.unlockAccount(eth.accounts[0])
37
38
39 ReferenceError: personal is not defined
40     at <eval>:1:1(0)
41
42 > eth.accounts
43 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
44  "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
45 > eth.sendTransaction({
46   to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
47   from: eth.accounts[0],
48   value: 100
49 });
50 Error: authentication needed: password or unlock
51     at web3.js:6365:9(39)
52     at send (web3.js:5099:62(29))
53     at <eval>:1:20(12)
54 > eth.sendTransaction({
55   to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
56   from: eth.accounts[0],
57 > eth.accounts
58
59 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
60  "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]

```

O erro **“Error: authentication needed: password or unlock”** ocorre por que precisamos autorizar a transação. Em versões anteriores era possível desbloquear as contas via console JavaScript, conforme tentamos no Código acima `personal.unlockAccount(eth.accounts[0])`. Como o `personal` foi depreciado não pode ser mais utilizado.

Na versão corrente é preciso utilizar o `clef` para fazer a autenticação em um console. Em um outro terminal inicie a instância do `clef` com o comando:

```

1 [rogerio@ryzen-nitro .etherprivate]$ clef --chainid 786 --keystore
2   ~/.etherprivate/keystore --configdir ~/.etherprivate/clef --http
3
4 WARNING!
5
6 Clef is an account management tool. It may, like any software, contain bugs.
7
8 Please take care to
9 - backup your keystore files,
10 - verify that the keystore(s) can be opened with your password.
11
12 Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
13 without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
14 PURPOSE. See the GNU General Public License for more details.
15
16 Enter 'ok' to proceed:
17 > ok
18
19 INFO [04-17|10:23:09.630] Using CLI as UI-channel
20 INFO [04-17|10:23:10.050] Loaded 4byte database embeds=268,621 locals=0

```

```

    local=./4byte-custom.json
20 WARN [04-17|10:23:10.050] Failed to open master, rules disabled err="failed stat on
    /home/rogerio/.etherprivate/clef/masterseed.json: stat
    /home/rogerio/.etherprivate/clef/masterseed.json: no such file or directory"
21 INFO [04-17|10:23:10.050] Starting signer chainid=786
    keystore=/home/rogerio/.etherprivate/keystore light-kdf=false advanced=false
22 INFO [04-17|10:23:10.052] Smartcard socket file missing, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
23 INFO [04-17|10:23:10.052] Audit logs configured file=audit.log
24 INFO [04-17|10:23:10.053] HTTP endpoint opened url=http://127.0.0.1:8550/
25 INFO [04-17|10:23:10.053] IPC endpoint opened
    url=/home/rogerio/.etherprivate/clef/clef.ipc
26
27 ----- Signer info -----
28 * extapi_version : 6.1.0
29 * extapi_http : http://127.0.0.1:8550/
30 * extapi_ipc : /home/rogerio/.etherprivate/clef/clef.ipc
31 * intapi_version : 7.0.1
32
33 ----- Available accounts -----
34 0. 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 at
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e44b03
35 1. 0x7A7686ad451d2865A2246E239B674aeFd4c6c27c at
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad451d
36 ----- List Account request-----
37 A request has been made to list all accounts.
38 You can select which accounts the caller can see
39 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
40 URL:
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
41 [x] 0x7A7686ad451d2865A2246E239B674aeFd4c6c27c
42 URL:
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
43 -----

```

E volte ao terminal onde iniciou o geth e indique que as autenticações serão via clef com o parâmetro `--signer=/home/rogerio/.etherprivate/clef/clef.ipc` passando o caminho dado pelo clef.

```

1 [rogerio@ryzen-nitro .etherprivate]$ geth --datadir ~/.etherprivate/ --syncmode full
    --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
    --http.addr 127.0.0.1 --http.port 8559 --http.api
    "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
    --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
    --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.ethereumbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
    --signer=/home/rogerio/.etherprivate/clef/clef.ipc

```

Cada transação executada no console JavaScript deverá ser autorizada no console do clef.

```

1
2 > eth.sendTransaction({
3   to: '0x7A7686ad451d2865a2246e239b674aeFd4c6c27c',
4   from: eth.accounts[0],
5 > eth.accounts

```

```

6
7 ["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
  "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
8 > eth.sendTransaction({
9   to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
10  from: eth.accounts[0],
11  value: 100
12 });
13 "0xb579cc595601e4aca546ce4e46bdcdd7841bd7f50a0a78c505e839dd039b8b9"
14 > eth.sendTransaction({
15   to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
16   from: eth.accounts[0],
17 > eth.sendTransaction({
18   to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
19   from: eth.accounts[0],
20 > eth.getBalance(eth.accounts[1])
21
22 0

```

No console do clef é possível autorizar e ver a transação assinada com hash: 0xb579cc595601e4aca546ce4e46bdcdd7841bd7f50a0a78c505e839dd039b8b9 que é o mesmo id devolvido no console JavaScript.

```

1 ----- List Account request-----
2 A request has been made to list all accounts.
3 You can select which accounts the caller can see
4 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
5   URL:
6     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
7 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
8   URL:
9     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
8 -----
9 Request context:
10   NA -> ipc -> NA
11
12 Additional HTTP header data, provided by the external caller:
13   User-Agent: ""
14   Origin: ""
15 Approve? [y/N]:
16 > y
17 ----- List Account request-----
18 A request has been made to list all accounts.
19 You can select which accounts the caller can see
20 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
21   URL:
22     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
23 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
24   URL:
25     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
24 -----
25 Request context:
26   NA -> ipc -> NA
27
28 Additional HTTP header data, provided by the external caller:
29   User-Agent: ""
30   Origin: ""

```

```

31 Approve? [y/N]:
32 > y
33 ----- Transaction request-----
34 to: 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
35 from: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 [chksum ok]
36 value: 100 wei
37 gas: 0x5208 (21000)
38 gasprice: 1000000000 wei
39 nonce: 0x0 (0)
40 chainid: 0x312
41
42 Request context:
43     NA -> ipc -> NA
44
45 Additional HTTP header data, provided by the external caller:
46     User-Agent: ""
47     Origin: ""
48 -----
49 Approve? [y/N]:
50 > y
51 ## Account password
52
53 Please enter the password for account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
54 >
55 -----
56 Transaction signed:
57 {
58     "type": "0x0",
59     "nonce": "0x0",
60     "gasPrice": "0x3b9aca00",
61     "maxPriorityFeePerGas": null,
62     "maxFeePerGas": null,
63     "gas": "0x5208",
64     "value": "0x64",
65     "input": "0x",
66     "v": "0x648",
67     "r": "0xe497ab329bf31af61f371e2eb251ca979ec8ba45e099318d44468e8703358418",
68     "s": "0x518b4e7e5906e4abc86e486523a8ad290727529f240a201905acc84c5f95417f",
69     "to": "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
70     "hash": "0xb579cc595601e4aca546ce4e46bcdcded7841bd7f50a0a78c505e839dd039b8b9"
71 }

```

No console JavaScript é possível recuperar o recibo da transação:

[illegible]

```

12  root: "0x9d1d510b5517e7c772e36c67e99d47de6d10c25ba9d24311d39eb85b7db6c071",
13  to: "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
14  transactionHash:
      "0xb579cc595601e4aca546ce4e46bdcdd7841bd7f50a0a78c505e839dd039b8b9",
15  transactionIndex: 0,
16  type: "0x0"
17 }

```

É preciso minerar para a transação ser efetivada:

```

1 > miner.start()
2 null
3 > eth.getBalance(eth.accounts[1])
4 100
5 > miner.stop()
6 null

```

Ao enviar outra quantida para a segunda conta é preciso autorizar a transação no console do `clef` e minerar novamente para ela ser efetivada.

```

1
2 > eth.sendTransaction({from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13", to:
      "0x7a7686ad451d2865a2246e239b674aefd4c6c27c", value: 100})
3 "0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980"
4 > eth.getBalance(eth.accounts[1])
5
6 100
7 > miner.start()
8 null
9 > eth.getBalance(eth.accounts[1])
10 200
11 > miner.stop()

```

Console do `clef` para a segunda transação:

```

1 ----- List Account request-----
2 A request has been made to list all accounts.
3 You can select which accounts the caller can see
4 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
5   URL:
      keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
6 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
7   URL:
      keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
8 -----
9 Request context:
10   NA -> ipc -> NA
11
12 Additional HTTP header data, provided by the external caller:
13   User-Agent: ""
14   Origin: ""
15 Approve? [y/N]:
16 > y
17 ----- List Account request-----
18 A request has been made to list all accounts.

```



```

19 You can select which accounts the caller can see
20 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
21 URL:
22     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
22 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
23 URL:
24     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
24 -----
25 Request context:
26     NA -> ipc -> NA
27
28 Additional HTTP header data, provided by the external caller:
29     User-Agent: ""
30     Origin: ""
31 Approve? [y/N]:
32 > y
33 ----- Transaction request-----
34 to: 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
35 from: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 [chksum ok]
36 value: 100 wei
37 gas: 0x5208 (21000)
38 gasprice: 1000000000 wei
39 nonce: 0x1 (1)
40 chainid: 0x312
41
42 Request context:
43     NA -> ipc -> NA
44
45 Additional HTTP header data, provided by the external caller:
46     User-Agent: ""
47     Origin: ""
48 -----
49 Approve? [y/N]:
50 > y
51 ## Account password
52
53 Please enter the password for account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
54 >
55 -----
56 Transaction signed:
57 {
58     "type": "0x0",
59     "nonce": "0x1",
60     "gasPrice": "0x3b9aca00",
61     "maxPriorityFeePerGas": null,
62     "maxFeePerGas": null,
63     "gas": "0x5208",
64     "value": "0x64",
65     "input": "0x",
66     "v": "0x647",
67     "r": "0x33d621389272cdbee73d2c50d91b846b325bb2d7b94f2c32d726c6fa21151f9a",
68     "s": "0x5510f0a5611c8c7640904b5a489422352fae1517eb7307bb6496468cff545726",
69     "to": "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
70     "hash": "0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980"
71 }
72 ----- List Account request-----

```

```

73 A request has been made to list all accounts.
74 You can select which accounts the caller can see
75 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
76 URL:
77     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
78 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
79 URL:
80     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
81 -----
82 Request context:
83     NA -> ipc -> NA
84
85 Additional HTTP header data, provided by the external caller:
86     User-Agent: ""
87     Origin: ""
88 Approve? [y/N]:
89 > y
90 ----- List Account request-----
91 A request has been made to list all accounts.
92 You can select which accounts the caller can see
93 [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
94 URL:
95     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
96 [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
97 URL:
98     keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
99 -----
100 Request context:
101     NA -> ipc -> NA
102
103 Additional HTTP header data, provided by the external caller:
104     User-Agent: ""
105     Origin: ""
106 Approve? [y/N]:
107 > y

```

A transação 0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980 foi autorizada e podemos verificar o saldo das carteiras:

[illegible]

```
15   transactionIndex: 0,  
16   type: "0x0"  
17 }  
18 > web3.fromWei(eth.getBalance("0xdb017e44b03b37755a4b15e14cd799f83de4c13"), "ether")  
19 5484.9999999999999998  
20 > web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")  
21 2e-16  
22 >
```

Mais detalhes de como implantar uma rede privada local do *Ethereum* podem ser visto em [Private Networks](#).

2.8 Leitura Recomendada

Leitura Recomendada

Capítulo 12: *Further Ethereum* (Imran 2018)

Livro: IMRAN BASHIR. *Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained*, 2nd Edition.

3 Prática: Instalando o Solidity

Para os testes com o desenvolvimento de Contratos Inteligentes iremos utilizar a linguagem Solidity. O Compilador para a linguagem Solidity é o `solc`. O `solc` converte código de alto nível escrito em Solidity para *bytecode* da *Ethereum Virtual Machine (EVM)*.

O comando para instalação em distribuições Ubuntu ou derivados do Debian:

```
1 $ sudo apt-get install solc
```

Outras distribuições como o Manjaro Linux, o pacote `solidity` deve ser instalado:

```
1 $ pacaaur -S solidity
```

Feita a instalação, para verificar a versão instalada execute o comando:

```
1 $ solc --version
2 solc, the solidity compiler commandline interface
3 Version: 0.8.19+commit.7dd6d404.Linux.g++
```

3.1 Compilando um Exemplo

Para verificar o funcionamento e algumas funcionalidades vamos criar um contrato simples, com o nome `Addition.sol` e com o seguinte conteúdo:

```
1 pragma solidity ^0.8.19;
2
3 contract Addition {
4     uint8 x;
5
6     function addx(uint8 y, uint8 z ) public {
7         x = y + z;
8     }
9     function retrievex() view public returns (uint8) {
10         return x;
11     }
12 }
```

Se a versão do `solidity` na sua máquina for diferente, basta ajustar no arquivo fonte colocando a versão correta.

Para a compilação simples execute:

```
1 [rag@ryzen-nitro]$ solc Addition.sol
2 Compiler run successful. No output generated.
3 [rag@ryzen-nitro]$
```

3.2 Visualizando o bytecode gerado

O `solc` tem alguns parâmetros interessantes que nos permite verificar o formato binário do contrato, que é a sequência dos *bytecodes* gerados para a **EVM**:

```

1 $ solc --bin Addition.sol
2 Warning: SPDX license identifier not provided in source file. Before publishing,
   consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to
   each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source
   code. Please see https://spdx.org for more information.
3 --> Addition.sol
4
5
6 ===== Addition.sol:Addition =====
7 Binary:
8 608060405234801561001057600080fd5b506101f6806100206000396000f3fe608060405234801561001057
9 600080fd5b50600436106100365760003560e01c806336718d801461003b578063ac04e0a014610057575b60
10 0080fd5b610055600480360381019061005091906100f2565b610075565b005b61005f61009e565b60405161
11 006c9190610141565b60405180910390f35b8082610081919061018b565b6000806101000a81548160ff0219
12 16908360ff1602179055505050565b60008060009054906101000a900460ff16905090565b600080fd5b6000
13 60ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b6000813590506100ec81
14 6100c6565b92915050565b60008060408385031215610109576101086100b4565b5b60006101178582860161
15 00dd565b9250506020610128858286016100dd565b9150509250929050565b61013b816100b9565b82525050
16 565b60006020820190506101566000830184610132565b92915050565b7f4e487b71000000000000000000
17 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000
18 506101a1836100b9565b9250828201905060ff8111156101ba576101b961015c565b5b9291505056fea26469
19 70667358221220e0ec16eaf684603f4f7c74f327a27e4a1a981dfac0cb258479ffe452abda2e4964736f6c63
20 430008110033

```

3.3 Estimando a taxa gas

Como uma taxa de gas é cobrada para cada operação que a EVM executa, é uma boa prática estimar o gas antes de implantar um contrato em uma rede ativa. O parâmetro `--gas` pode ser utilizado para fazer essa estimativa.

```

1 $ solc --gas Addition.sol
2 ===== Addition.sol:Addition =====
3 Gas estimation:
4 construction:
5   147 + 100400 = 100547
6 external:
7   addx(uint8,uint8): infinite
8   retrievex(): 2479

```

3.4 Gerando a ABI

A *Application Binary Interface (ABI)* é uma forma padrão de interagir com os contratos, sabermos como os métodos estão disponíveis e quais parâmetros utilizam. Para a gerar a ABI do contrato utilize o `solc` com o parâmetro `--abi`.

```

1 $ solc --abi Addition.sol
2 ===== Addition.sol:Addition =====
3 Contract JSON ABI
4 [{"inputs":[{"internalType":"uint8","name":"y","type":"uint8"},{"internalType":"uint8",
5 "name":"z","type":"uint8"}],"name":"addx","outputs":[],"stateMutability":"nonpayable",
6 "type":"function"},{"inputs":[],"name":"retrievex","outputs":[{"internalType":"uint8",

```

```
7 "name":"","type":"uint8"}],"stateMutability":"view","type":"function"}]
```

3.5 Processo de Compilação Completo

O processo de compilação completo do contrato `Addition.sol` pode ser feito com o comando:

```
1 $ solc --bin --abi -o bin Addition.sol
2 Compiler run successful. Artifact(s) can be found in directory "bin".
```

Se erros ocorrerem serão mostrados no terminal, caso contrário o compilador irá mostrar uma mensagem de sucesso. Com o parâmetro de diretório de saída `-o bin`, serão gerados os arquivos no diretório `bin`:

- **Addition.abi:** Contém a ABI do contrato no formato JSON.
- **Addition.bin:** Contém a representação binária do código do contrato.

O conteúdo de cada um dos arquivos pode ser visualizado:

```
1 $ cat bin/Addition.bin
2 608060405234801561001057600080fd5b506101f6806100206000396000f3fe608060405234801561001057
3 600080fd5b50600436106100365760003560e01c806336718d801461003b578063ac04e0a014610057575b60
4 0080fd5b610055600480360381019061005091906100f2565b610075565b005b61005f61009e565b60405161
5 006c9190610141565b60405180910390f35b8082610081919061018b565b6000806101000a81548160ff0219
6 16908360ff1602179055505050565b60008060009054906101000a900460ff16905090565b600080fd5b6000
7 60ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b6000813590506100ec81
8 6100c6565b92915050565b60008060408385031215610109576101086100b4565b5b60006101178582860161
9 00dd565b9250506020610128858286016100dd565b9150509250929050565b61013b816100b9565b82525050
10 565b60006020820190506101566000830184610132565b92915050565b7f4e487b7100000000000000000000
11 0000000000000000000000000000000000600052601160045260246000fd5b6000610196826100b9565b91
12 506101a1836100b9565b9250828201905060ff8111156101ba576101b961015c565b5b9291505056fea26469
13 70667358221220e0ec16eaf684603f4f7c74f327a27e4a1a981dfac0cb258479ffe452abda2e4964736f6c63
14 430008110033
15
16 $ cat bin/Addition.abi
17 [{"inputs":[{"internalType":"uint8","name":"y","type":"uint8"},{"internalType":"uint8",
18 "name":"z","type":"uint8"}],"name":"addx","outputs":[],"stateMutability":"nonpayable",
19 "type":"function"},{"inputs":[],"name":"retrieveX","outputs":[{"internalType":"uint8",
20 "name":"","type":"uint8"}],"stateMutability":"view","type":"function"}]
```

3.6 Visualizando os Opcodes

Os *opcodes* das instruções geradas para a **EVM** podem ser visualizados compilando-se com o parâmetro `--opcodes`:

```
1 [rag@ryzen-nitro ]$ solc --opcodes Addition.sol
2
3 ===== Addition.sol:Addition =====
4 Opcodes:
5 PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0 DUP1
   REVERT JUMPDEST POP PUSH2 0x1f6 DUP1 PUSH2 0x20 PUSH1 0x0 CODECOPY PUSH1 0x0
   RETURN INVALID PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI
```

[illegible]

```
6 [rag@ryzen-nitro]$
```

Para uma lista completa de parâmetros aceitos pelo `solc` execute no terminal o comando `solc --help`.

3.7 Leitura Recomendada

Leitura Recomendada

Capítulo 14: *Development Tools and Frameworks* (Imran 2018)

Livro: IMRAN BASHIR. Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

4 Prática: Introdução ao Web3

Nesta prática são apresentadas algumas ferramentas de Desenvolvimento e Frameworks para o desenvolvimento e implantação de Contratos Inteligentes. Apresenta uma introdução ao Web3, métodos de desenvolvimento, teste e verificação de contratos inteligentes com Ganache, console do cliente Geth e Remix IDE. Introduce o Truffle *framework*, que também pode ser usado para testar, migrar contratos inteligentes e o Drizzle, para criar *frontends* de DApps de maneira mais fácil, com IPFS, para hospedar as páginas web da aplicação.

A proposta é explorarmos a biblioteca Web3 com o cliente Geth, e os métodos de desenvolvimento, teste e verificação de contratos inteligentes com Ganache, console do cliente Geth. Fazer o *deploy* de contratos inteligentes utilizando o *console* Geth e O Truffle. O Truffle pode ser usado para testar, migrar contratos inteligentes.

4.1 Instalação das Ferramentas

3. Instale as outras ferramentas: Node.js, Ganache e Ganache-CLI, Truffle, Drizzle, Embark e outras ferramentas indicadas no capítulo. O truffle utiliza o nodejs nas versões v14-v18 (<https://trufflesuite.com/docs/truffle/how-to/install/>)
4. Instale o Node.js e as bibliotecas necessárias.

```
1 $ sudo apt-get install nvm
2 $ nvm install 18
```

2. Utilizando o Truffle baixar o exemplo de projeto MetaCoin e fazer o deploy no Ganache.

4.2 Explorando Web3 com Geth

- Web3 é uma biblioteca JavaScript que pode ser usada na comunicação com um Nó *Ethereum* via comunicação RPC. Web3 expõe métodos que o acesso está disponível sobre RPC.
- A interação com o cliente Geth é possível via *Geth JavaScript Console*, que expõe vários métodos de consulta e gerenciamento do blockchain.
- Vimos os comandos para execução do Geth e do Console JavaScript na **Aula 021 - Prática sobre Ethereum: Ambiente de Desenvolvimento**.
- Iniciar o Nó Geth com suporte ao web3:

```
1 $ geth --datadir ~/.etherprivate/ --allow-insecure-unlock --networkid 786 --http
  --http.addr 127.0.0.1 --http.port 8559 --http.api
  "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
  --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
  --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
```

- Iniciar um console para a interação com a execução:

```
1 $ geth attach ~/.etherprivate/geth.ipc
2 Welcome to the Geth JavaScript console!
3
4 instance: Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.19.3
5 coinbase: 0xedbc36d74d5a1cd64db36e53798bd1781f0c4955
6 at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
7 datadir: /home/rag/.etherprivate
8 modules: admin:1.0 debug:1.0 engine: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
9
10 To exit, press ctrl-d or type exit
```

- Verificando se os recursos `web3` estão disponíveis:

```
1 > web3.version
2 {
3   api: "0.20.1",
4   ethereum: undefined,
5   network: "786",
6   node: "Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.19.3",
7   whisper: undefined,
8   getEthereum: function(callback),
9   getNetwork: function(callback),
10  getNode: function(callback),
11  getWhisper: function(callback)
12 }
13 >
```

4.3 Web3 deployment

- Faremos um *deploy* usando o *Geth console*.
- O passo a passo pode ser visto no livro e iremos reproduzir aqui, seguindo a sequência de passos:
 - Executar o *Geth client*.
 - Criar um script de *deployment*, usando a *ABI* e o *bytecode*, e algum código JavaScript.
 - Faremos o *deploy* do contrato via linha de comando pelo *Geth console*.
 - Interagir com o contrato via um *frontend web*.

4.4 Web3 deployment: Executar o Geth client

- Executar o *Geth client*. [✓]
- Executar o *Geth console*. [✓]

4.5 Web3 deployment: Criar um script de deployment

- Compile o contrato com o `solc` ou utilizando o `Remix IDE`, gerando o binário e a *ABI*:

```

1 $ solc --bin --abi -o bin ValueChecker.sol
2 $ ls
3 bin deploy.js ValueChecker.sol
4 $ cd bin
5 $ ls
6 valueChecker.abi valueChecker.bin
7 $ cat valueChecker.bin
8 6080604052600a60005534801561001557600080fd5b5061018b806100256000396000f3fe60806040523480
9 1561001057600080fd5b506004361061002b5760003560e01c8063f9d55e2114610030575b600080fd5b6100
10 4a600480360381019061004591906100f2565b610060565b604051610057919061013a565b60405180910390
11 f35b600080548260ff16106100ae577f3eb1a229ff7995457774a4bd31ef7b13b6f4491ad1ebb8961af120b8
12 b4b6239c600160405161009d919061013a565b60405180910390a1600190506100af565b5b919050565b6000
13 80fd5b600060ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b6000813590
14 506100ec816100c6565b92915050565b600060208284031215610108576101076100b4565b5b600061011684
15 8285016100dd565b91505092915050565b60008115159050919050565b6101348161011f565b8252050565b
16 600060208201905061014f600083018461012b565b9291505056fea264697066735822122088a7e63726327b
17 857c0d0a6d073976f05d5073826c629671c857a375db35d51c64736f6c63430008110033
18
19 $ cat valueChecker.abi
20 [{"anonymous":false,"inputs":[{"indexed":false,"internalType":"bool","name":"returnValue",
21 "type":"bool"}],"name":"valueEvent","type":"event"},{"inputs":[{"internalType":"uint8",
22 "name":"x","type":"uint8"}],"name":"Matcher","outputs":[{"internalType":"bool",
23 "name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"}]

```

- Preparação do código *JavaScript*:

```

1 var valuecheckerContract = web3.eth.contract([{"anonymous": false, "inputs": [{"
  "indexed": false, "internalType": "bool", "name": "returnValue", "type": "bool"
}], "name": "valueEvent", "type": "event" }, {"inputs": [{"internalType":
  "uint8", "name": "x", "type": "uint8" }], "name": "Matcher", "outputs": [{"
  "internalType": "bool", "name": "", "type": "bool" }], "stateMutability":
  "nonpayable", "type": "function" }]);
2 var valuechecker = valuecheckerContract.new({
3   from: web3.eth.accounts[0],
4   data:
5     '0x6080604052600a60005534801561001557600080fd5b5061010d806100256000396000f3006080
6     60405260043610603f576000357c01000000000000000000000000000000000000000000000000000000000000000000
7     000000900463ffffffff168063f9d55e21146044575b600080fd5b348015604f57600080fd5b5060
8     6f600480360381019080803560ff16906020019092919050506089565b60405180821515151581
9     5260200191505060405180910390f35b600080548260ff1610151560db577f3eb1a229ff79954577
10    74a4bd31ef7b13b6f4491ad1ebb8961af120b8b4b6239c6001604051808215151515815260200191
11    505060405180910390a16001905060dc565b5b9190505600a165627a7a723058209ff756514f1ef4
12    6f5650d800506c4eb6be2d8d71c0e2c8b0ca50660fde82c7680029', gas: '4700000'
13 },
14 function (e, contract) {
15   console.log(e, contract);
16   if (typeof contract.address !== 'undefined') {
17     console.log('Contract mined! address: ' + contract.address +
18       'transactionHash: ' + contract.transactionHash);
19   }
20 })

```

4.6 Web3 deployment: Fazendo o deploy pelo Geth console

- No *Geth console* dê um *unlock* na conta:

```
1 > personal.listAccounts[0]
2 "0xedbc36d74d5a1cd64db36e53798bd1781f0c4955"
3 > personal.unlockAccount(personal.listAccounts[0])
4 Unlock account 0xedbc36d74d5a1cd64db36e53798bd1781f0c4955
5 Passphrase:
6 true
7 >
```

- Cole o código JavaScript para fazer o *deploy*:

[illegible]

- Na execução do *deploy* deu uma mensagem de erro `Error: insufficient funds for gas * price + value undefined`, pois a carteira da conta selecionada não tem saldo suficiente. É necessário minerar para ganhar algum saldo:

```
1 > miner.start()
2 null
3 > miner.stop()
4 > null
```

- Repetindo o processo de *deploy*:

[illegible]

- Nos logs do Nó Geth irá aparecer a mensagem de que o contrato foi submetido:

```
1 INFO [11-24|12:44:17.115] Submitted contract creation
  hash=0x975501f4b6c24a46d13ead5840f40bc03460c6be4139cbd6c3d902e73790796c
  from=0xeDbC36d74d5a1Cd64DB36E53798bd1781f0C4955 nonce=1
  contract=0xfbe4899126470AF8dd4d37e878f0De486a6CFA71 value=0
```

- Iniciando a mineração o contrato será minerado:

```

1 > miner.start()
2 null
3 > null [object Object]
4 Contract mined! address: 0xfbe4899126470af8dd4d37e878f0de486a6cfa71 transactionHash:
    0x975501f4b6c24a46d13ead5840f40bc03460c6be4139cbd6c3d902e73790796c
5 > miner.stop()

```

4.7 Web3 deployment: Interagindo com o contrato

- Interagir com o contrato via *Geth console*
 - Após o *deployment* através da sua ABI o contrato estará disponível no *console*:

```

1 > valuechecker.
2 valuechecker.Matcher valuechecker.address valuechecker.transactionHash
3 valuechecker._eth valuechecker.allEvents valuechecker.valueEvent
4 valuechecker.abi valuechecker.constructor
5 > valuechecker.address
6 "0xfbe4899126470af8dd4d37e878f0de486a6cfa71"
7 > valuechecker.transactionHash
8 "0x975501f4b6c24a46d13ead5840f40bc03460c6be4139cbd6c3d902e73790796c"

```

Percebam o mesmo *address* e *transactionHash* que foram devolvidos no processo de *deploy*.

- A ABI do *valuechecker* está disponível:

```

1 > valuechecker.abi
2 [{
3   anonymous: false,
4   inputs: [{
5     indexed: false,
6     internalType: "bool",
7     name: "returnValue",
8     type: "bool"
9   }],
10  name: "valueEvent",
11  type: "event"
12 }, {
13   inputs: [{
14     internalType: "uint8",
15     name: "x",
16     type: "uint8"
17   }],
18   name: "Matcher",
19   outputs: [{
20     internalType: "bool",
21     name: "",
22     type: "bool"
23   }],
24   stateMutability: "nonpayable",
25   type: "function"

```

```
26 }]  
27 >
```

- A função `Matcher` pode ser invocada para a verificação de valores:

```
1 > eth.getBalance(valuechecker.address)  
2 0  
3 > valuechecker.Matcher.call(12)  
4 true  
5 > valuechecker.Matcher.call(10)  
6 true  
7 > valuechecker.Matcher.call(5)  
8 false  
9 >
```

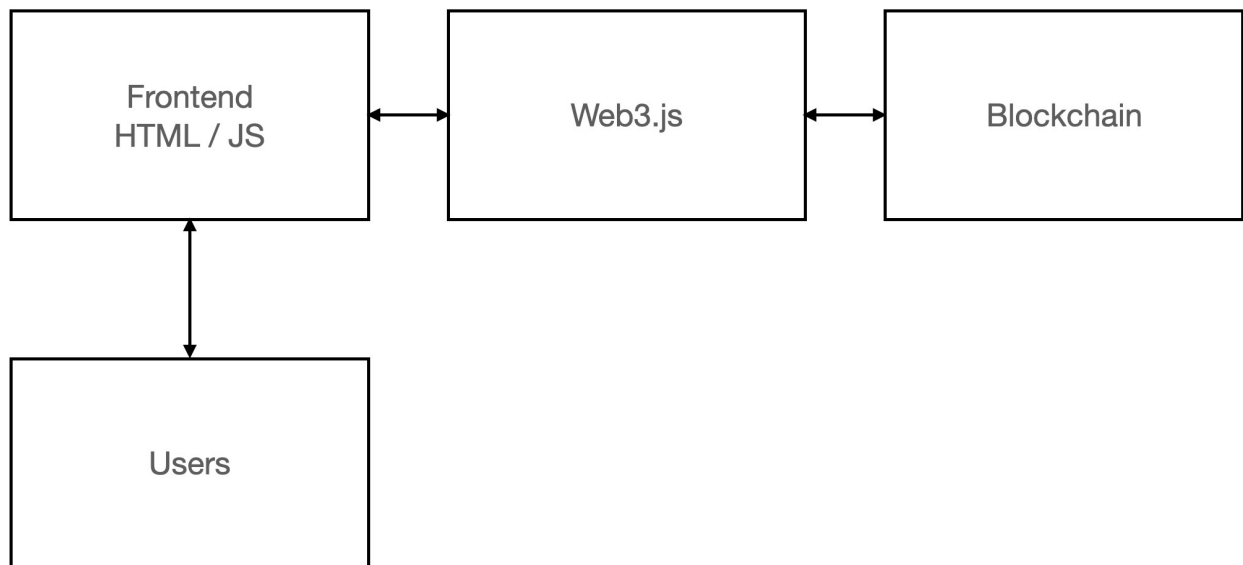
- Interagir com o contrato via um *frontend* web.
- *POST requests* É possível interagir com o Geth via JSON RPC sobre o HTTP. Para esse teste utilizaremos o `curl`. Lembrando que a porta utilizando foi a 8559.
- **Recuperando a lista de contas:** A lista de contas pode ser obtida utilizando o método `personal_listAccounts`, conforme o comando:

```
1 $ curl --request POST --data  
    '{"jsonrpc":"2.0","method":"personal_listAccounts","params": [],"id":4}'  
    localhost:8559 -H "Content-Type: application/json"  
2 {"jsonrpc":"2.0","id":4,"result":["0xedbc36d74d5a1cd64db36e53798bd1781f0c4955",  
3  "0x1478d95f8754b3ba7127100dd0bb46578fe7d22a"]}]}
```

Um objeto JSON é retornado com a lista de contas. No comando `curl`, o parâmetro `--request` é usado para especificar que o comando é uma requisição do tipo `POST` e `--data` é usado para especificar os parâmetros e valores. Finalmente, o `localhost:8559` é usado para indicar o endereço que o HTTP endpoint do Geth está aberto.

4.8 Interagindo com contratos via frontends web

- A interação com *smart contracts* como parte de uma DApps é normalmente feito usando uma interface web desenvolvida utilizando HTML/JS/CSS. Algumas bibliotecas e *frameworks* como React, Redux, e Drizzle, podem também ser usadas.



4.9 Biblioteca Javascript Web3.js

- Se ainda não instalou a biblioteca `web3.js`, pode instalá-la via `npm` com o comando:

```
1 $ npm install web3
```

A biblioteca `Web.js` disponibiliza alguns módulos, sendo eles:

- **web3-eth:** Ethereum blockchain e smart contracts.
- **web3-shh:** Protocolo `Whisper` (Comunicação e *broadcast* P2P).
- **web3-bzz:** Protocolo `Swarm`, que fornece armazenamento descentralizado.
- **web3-utils:** Fornece funções úteis para o desenvolvimento de DApps.
- Criando um servidor `http` para testar a app.

```
1 # Python 3.x
2 python3 -m http.server 7777
3 # If Python version returned above is 2.X
4 python -m SimpleHTTPServer 7777
```

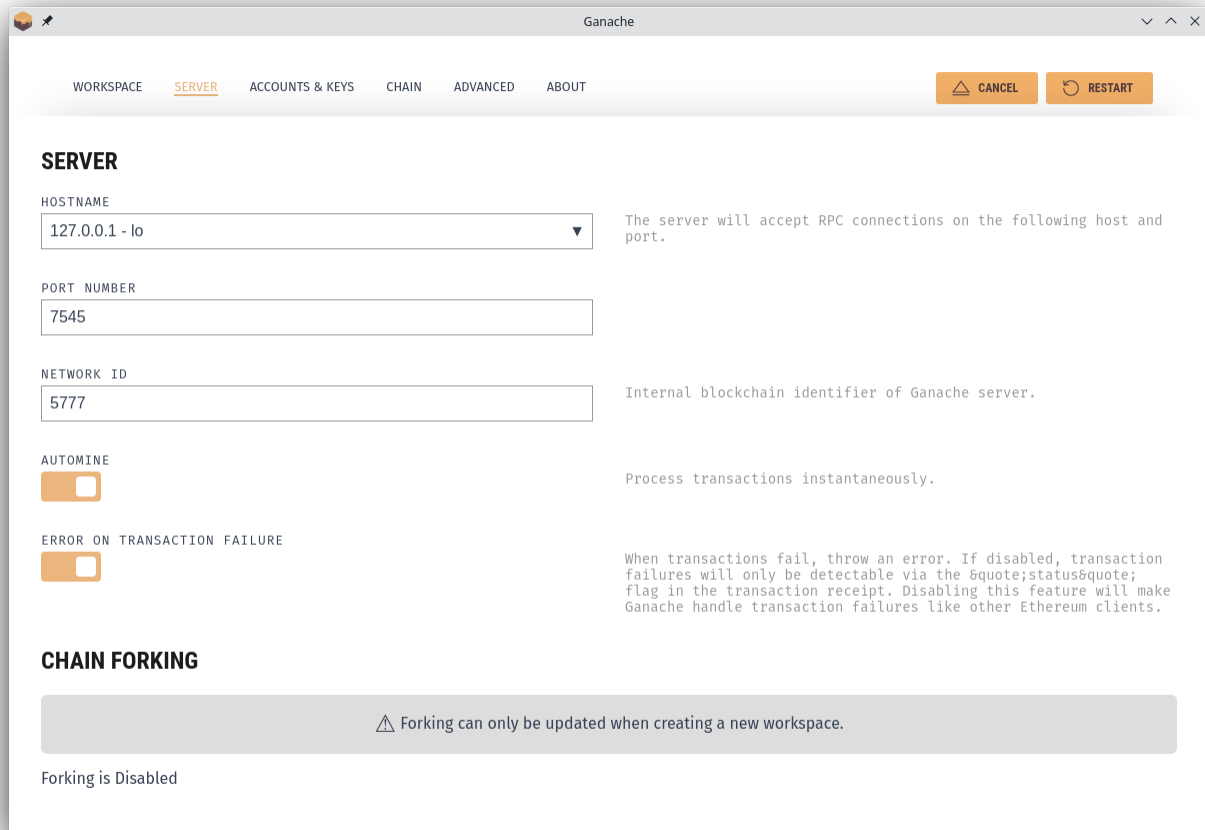
4.10 Development frameworks

- Installing and initializing Truffle
 - Truffle initialization is performed using the `Truffle init` command, which generates a skeleton structure for a project

- Compiling, testing, and migrating using Truffle
 - Several commands available in Truffle can be used to compile, test and deploy smart contracts

4.11 Configuração do Ganache

- We can use Ganache as a local blockchain to provide the RPC interface.



4.12 Interagindo com um contrato

- O console do Truffle expõe vários métodos que podem ser usados para interagir com contratos.

```

(truffle(development)> MetaCoin.
MetaCoin.__defineGetter__
MetaCoin.__proto__
MetaCoin.toLocaleString

MetaCoin.apply
MetaCoin.toString

MetaCoin._constructorMethods
MetaCoin.abi
MetaCoin.ast
MetaCoin.bytecode
MetaCoin.compiler
MetaCoin.currentProvider
MetaCoin.deployedBinary
MetaCoin.devdoc
MetaCoin.hasNetwork
MetaCoin.length
MetaCoin.name
MetaCoin.networks
MetaCoin.resetAddress
MetaCoin.setNetworkType
MetaCoin.sourceMap
MetaCoin.transactionHash
MetaCoin.userdoc

MetaCoin.__defineSetter__
MetaCoin.hasOwnProperty
MetaCoin.valueOf

MetaCoin.bind

MetaCoin.call

MetaCoin._json
MetaCoin.addProp
MetaCoin.at
MetaCoin.caller
MetaCoin.configureNetwork
MetaCoin.decodeLogs
MetaCoin.deployedBytecode
MetaCoin.ens
MetaCoin.interfaceAdapter
MetaCoin.link
MetaCoin.network
MetaCoin.new
MetaCoin.schemaVersion
MetaCoin.setProvider
MetaCoin.sourcePath
MetaCoin.unlinked_binary
MetaCoin.web3

MetaCoin.__lookupGetter__
MetaCoin.isPrototypeOf

MetaCoin._properties
MetaCoin.address
MetaCoin.autoGas
MetaCoin.class_defaults
MetaCoin.contractName
MetaCoin.defaults
MetaCoin.deployedSourceMap
MetaCoin.events
MetaCoin.isDeployed
MetaCoin.links
MetaCoin.networkType
MetaCoin.numberFormat
MetaCoin.schema_version
MetaCoin.setWallet
MetaCoin.timeoutBlocks
MetaCoin.updatedAt

MetaCoin.__lookupSetter__
MetaCoin.propertyIsEnumerable

MetaCoin.constructor

MetaCoin._property_values
MetaCoin.arguments
MetaCoin.binary
MetaCoin.clone
MetaCoin.contract_name
MetaCoin.deployed
MetaCoin.detectNetwork
MetaCoin.gasMultiplier
MetaCoin.legacyAST
MetaCoin.metadata
MetaCoin.network_id
MetaCoin.prototype
MetaCoin.setNetwork
MetaCoin.source
MetaCoin.toJSON
MetaCoin.updated_at

```

4.13 Developing a proof of idea project

The screenshot displays the Remix IDE interface during the deployment of a smart contract named 'PatentIdea'. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing the 'Web3 Provider' environment, account '0x236...A9ADA' with a balance of 98.817 ETH, and a gas limit of 3,000,000. The contract 'PatentIdea - browser/patent.sol' is selected for deployment. Below this, a list of 'Deployed Contracts' shows 'PatentIdea at 0x10F...96Ed2 (blockchain)'. The main editor displays the Solidity code for 'PatentIdea', which includes a mapping of hashes, a tracker, and functions for saving and checking hashes. The bottom status bar indicates the transaction was successful: '[block:113 txIndex:0] from:0x236...A9ADA to:PatentIdea.(constructor) value:0 wei data:0x608...'.

4.14 Creating the ideap project

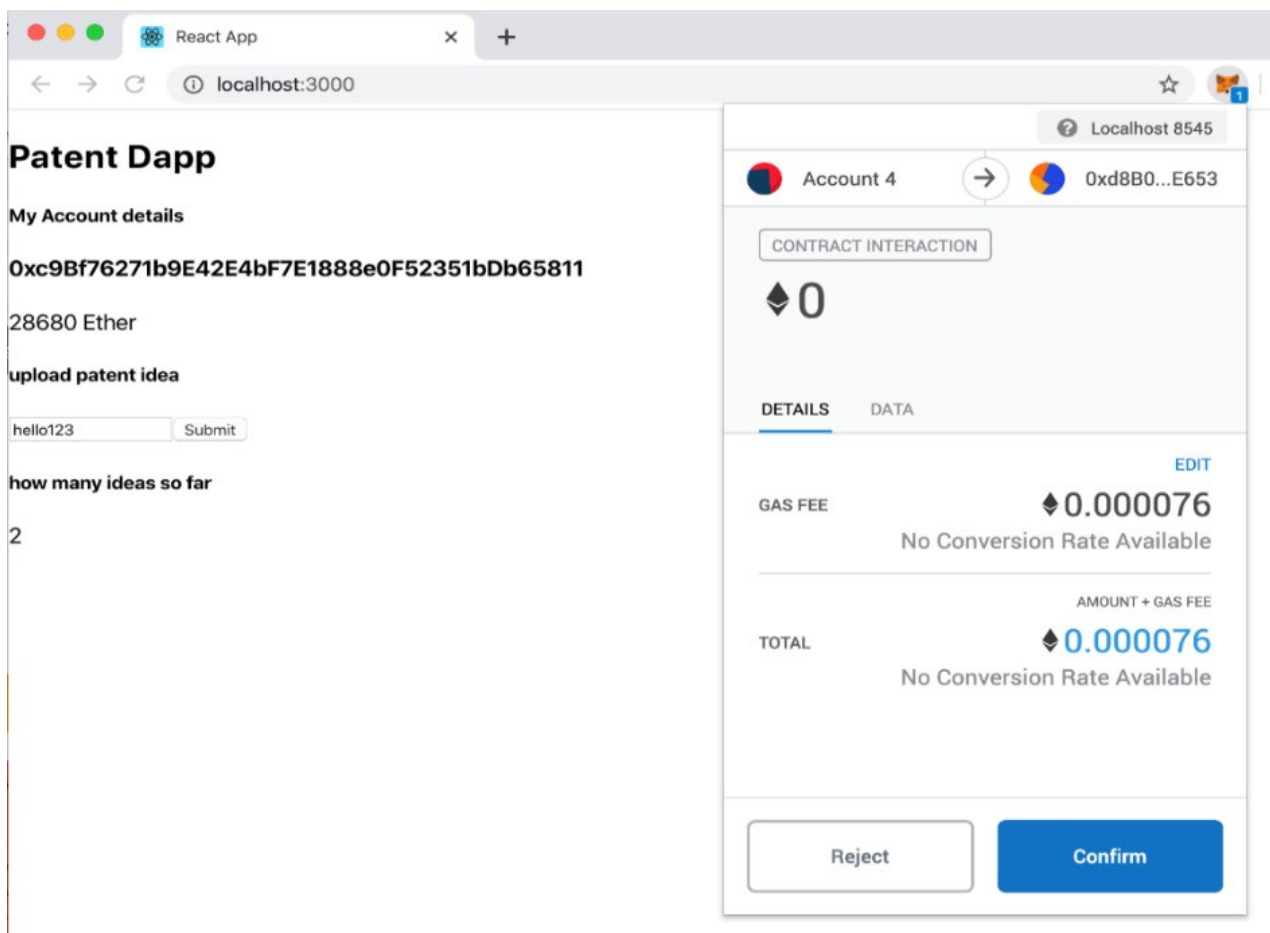
The necessary steps to create a proof of idea project, as detailed in the core Mastering Blockchain book, are as follows:

- Write the ideap smart contract
- Compile and test it in the Remix IDE

- Deploy to Ganache using Truffle
- Deploy to your network of choice (this is optional)
- Build a web frontend using Drizzle
- Run the DApp!

4.15 Patent DApp

This is the resulting interactive frontend of the proof of idea DApp.



4.16 IPFS

- Traditionally, storage is centralized.
- In order to decentralize the entire blockchain ecosystem, storage services should also be decentralized, and serve as decentralized storage layer of the blockchain.
- DApps can benefit from decentralized storage, where backend data can be stored without fear of censorship or centralized control.

4.17 Atividade

- Instalar as ferramentas do Capítulo e implementar os projetinhos de exemplos.
- Utilizando o Truffle baixar o exemplo de projeto MetaCoin e fazer o deploy no Ganache.

4.18 Leitura Recomendada

Leitura Recomendada

Capítulo 15: Introducing Web3

Livro: [IMRAN BASHIR](#). Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

