# GETH

**Curso Blockchain Developer - Turma JUN2018**

23 de Julho de 2018

## Material produzido por [bbchain](bbchain).



# 1. Objetivo

Criar uma rede privada utilizando o client Geth.

# 2. Materiais

- [Geth] (https://geth.ethereum.org/downloads/)
- [Terminal de Comando]

# 3. Instruções

# 3.1 Validação de instalação

No terminal de comando execute o comando:

```
geth help
```

O resultado deve ser:

```
NAME:
   geth.exe - the go-ethereum command line interface

   Copyright 2013-2018 The go-ethereum Authors

USAGE:
   geth.exe [options] command [command options] [arguments...]

VERSION:
   1.8.11-stable-dea1ce05

COMMANDS:
   account           Manage accounts
   attach            Start an interactive JavaScript environment (connect t
   bug               opens a window to report a bug on the geth repo
   console           Start an interactive JavaScript environment
   copydb            Create a local chain from a target chaindata folder
   dump              Dump a specific block from storage
   dumpconfig        Show configuration values
   export            Export blockchain into file
   export-preimages  Export the preimage database into an RLP stream
   import            Import a blockchain file
   import-preimages  Import the preimage database from an RLP stream
   init              Bootstrap and initialize a new genesis block
   js                Execute the specified JavaScript files
   license           Display license information
   makecache         Generate ethash verification cache (for testing)
   makedag           Generate ethash mining DAG (for testing)
  .
  .
  .
```

# 3.2 Setup inicial

## 3.2.1 Criação de conta

Execute o comando:

```
geth account new
```

Será pedido a inserção de uma senha. Está senha irá proteger a sua chave privada, criptografando o arquivo. Mas caso você perca a senha, será a mesma coisa que perder a chave, é muito importante mante-la em um lugar que você possa recuperar, preferencialmente fora de sua máquina.

Resultado esperado:

```
Your new account is locked with a password. Please give a password. Do not
Passphrase:
Repeat passphrase:
Address: {073e0b83225719e9216c1cf424c8eaf4ad5a3e44}
```

Com um client Ethereum, não há necessidade de utilizar serviços na rede para a geração de suas contas, este é um dos processos mais seguros para a criação de uma carteira, desde que você consiga garantir que a sua máquina não está comprometida.

### 3.2.2 Criação do bloco Genesis

Para construir uma rede privada, é necessário construir o bloco que Genesis, que representa o estado inicial da rede. Neste bloco, é possível determinar o nível de complexidade de mineração e valores em Ether iniciais para carteiras de sua escolha.

Para fazer isto, vamos criar uma nova pasta com o nome `Geth` em uma pasta de trabalho.

Crie um novo arquivo de texto com o nome `genesis.json`, salvo com encoding `UTF-8`.

Preencha este arquivo com as informações abaixo:

```
{
 "config": {
   "chainID": 1234,
   "homesteadBlock": 0,
   "eip155Block": 0,
   "eip158Block": 0
 },
 "alloc": {
    "0x<ENDERECO DA SUA CARTEIRA>": {
      "balance": "1000000000000000000000000000"
```

```
    }
  },
  "difficulty": "0x4000",
  "gasLimit": "0xffffffff",
  "nonce": "0x0000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000
  "extraData": "0x123458db4e347b1234537c1c8370e4b5ed33adb3db69cbdb7a38e1e50b
  "timestamp": "0x00"
}
```

Em redes privadas, normalmente não é necessário definir um valor inicial de Ether para uma carteira, já que a mineração normalmente é de baixa complexidade, mas vamos inicializar a conta que criamos com algum valor de Ether para facilitar os nossos testes.

### 3.2.3 Criação de um novo nó

Para criar um nó Geth, é necessário construir um novo diretório de dados.
Cada diretório de dados representa um nó na rede, mesmo trabalhando local, é necessário colocar cada nó em sua pasta para simular um cenário mais próximo do real.

Para criar um novo nó, execute o seguinte comando da pasta onde se encontra o arquivo `genesis.json` criado anteriormente:

```
geth --datadir "<Caminho para pasta>/LocalNode1" init genesis.json
```

Onde `<Caminho para pasta>` é o caminho onde você deseja armazenar o seu nó.

Este comando irá criar na pasta LocalNode1, um diretório `geth` , onde fica armazenada a *blockchain*, e a pasta `keystore` , onde ficam armazenados as suas chaves privadas.

Agora já conseguimos acessar esta rede, para isto utilize o comando:

```
geth --datadir "<Caminho para pasta>/LocalNode1" --networkid 1234 --port
11111 --nodiscover console
```

Onde `--datadir` representa o diretório onde está armazenada a blockchain, `--networkid` é o número da rede que utilizamos na especificação do bloco **genesis**, `--port` é a porta que será aberta para comunicação com este nó, `--nodiscover` informa que não queremos que haja auto-discovery de outros nós da rede e por fim o comando `console` que irá inicializar o `geth` com um console para executarmos transações.

**Resultado esperado**

```
INFO [07-23|14:22:47] Maximum peer count                       ETH=25 LES=0
INFO [07-23|14:22:47] Starting peer-to-peer node               instance=Get
INFO [07-23|14:22:47] Allocated cache and file handles         database=C:\
INFO [07-23|14:22:47] Initialised chain configuration          config="{Cha
INFO [07-23|14:22:47] Disk storage enabled for ethash caches   dir=C:\\User
INFO [07-23|14:22:47] Disk storage enabled for ethash DAGs     dir=C:\\User
INFO [07-23|14:22:47] Initialising Ethereum protocol           versions="[6
INFO [07-23|14:22:47] Loaded most recent local header          number=0 has
INFO [07-23|14:22:47] Loaded most recent local full block      number=0 has
INFO [07-23|14:22:47] Loaded most recent local fast block      number=0 has
INFO [07-23|14:22:47] Regenerated local transaction journal    transactions
INFO [07-23|14:22:48] Starting P2P networking
INFO [07-23|14:22:48] RLPx listener up                         self="enode:
INFO [07-23|14:22:48] IPC endpoint opened                      url=\\\\.\\p
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.11-stable-dea1ce05/windows-amd64/go1.10.2
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.
```

## 3.2.4 Acesso a informações do nó

As estruturas locais podem ser acessadas através da variável `personal`, por ela, você consegue acessar as suas contas e wallets armazenadas na pasta `keystore`, usando o comando `personal.listAccounts` devemos ter o resultado abaixo:

```
> personal.listAccounts
[]
```

Como não possuímos nada em nossa keystore, o retorno é vazio.

Vamos pegar a conta que criamos no começo deste exercício e mover para a nossa keystore.

Para isto, precisamos acessar as pastas globais do Geth:

```
OSX: ~/Library/Ethereum
Linux: ~/.ethereum
Windows: %APPDATA%\Ethereum.
```

Acesse a pasta `keystore` e copie o arquivo para a pasta `keystore` do nó que você criou.

Execute o comando para listar a conta novamente:

```
> personal.listAccounts
["0x281b76c2c1a3af4aecd0a944202cce9853e9bf67"]
```

Agora já conseguimos ver o endereço que foi criado, podemos verificar o saldo presente na carteira em *ether* com o seguinte comando:

```
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether");
100000000000
```

Como inicializamos este endereço com *Ether*, já é possível ver seu saldo.

### 3.2.5 Mineração

Por padrão, o Geth não inicializa fazendo a mineração. para que o processo comece, utilize o comando `miner.start(8)` , onde o número passado é a quantidade de threads de mineração você quer utilizar. Para parar a mineração, basta utilizar o comando `miner.stop()`.

Também é possível inicializar o Geth já minerando utilizando o comando:

```
geth --mine --minerthreads=4
```

A primeira coisa que é realizada no inicio da mineração é a geração do grafo de transações, após isto, blocos novos serão gerados a todo momento, independente de haver transações ou não, desta forma, mantendo sempre a mesma dificuldade e evitando que um nó específico consiga explorar está transação.

Caso você verifique novamente a sua conta inicial, verá que ela possuí mais *Ethers* após minerar por alguns minutos.

```
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether");
100000000035
```

Já que não foi definido nenhum endereço como padrão para recebimento do *Ether*, o Geth irá selecionar a primeira conta disponível.

## 3.3 Comunicação

### 3.3.1 Inicialização

Para falar com o Geth, alguns outros parâmetros são necessários para habilitar o RPC, execute o comando abaixo no seu terminal:

```
geth --datadir "<Caminho para pasta>/LocalNode1" --networkid 1234 --port 11
```

Os comandos adicionais são:

- `rpc` : habilita o acesso rpc no nó;
- `rpcport` : especifica qual porta vai ser utilizada para comunicação rpc;
- `rpccorsdoamin` : especifica quais domínios podem se comunicar com seu nó, no caso, qualquer um;
- `rpcapi` : especifica quais serviços serão disponibilizados via rpc;
  Consigo, agora é possível acessar o seu nó através da porta 8080.

### 3.3.2 Conexão

Vamos agora conectar diretamente do **Remix** em nosso nó.
Para fazer isto, deve-se selecionar, dentro da aba `run` a opção, `Web3Provider` . Será questionado se você autoriza o acesso. Selecione `Ok` e acesse através do endereço `http://localhost:8080` .

Agora já é possível acessar diretamente o seu nó. Sua conta irá aparecer carregada com os valores de *Ether* disponível.

Porém isto não é o suficiente para executar uma transação. Primeiro é preciso desbloquear a sua conta neste nó. Para fazer isto execute no console do Geth o comando:

```
> personal.unlockAccount(eth.accounts[0], "<password>")
true
```

Já podemos fazer transações, caso o seu `miner` esteja ligado, ele irá minerar a sua transação e te entregar uma receita.

# 4 Desafio

Habilite o seu Dapp para conversar diretamente com um nó em sua máquina. Será necessário:

- [ ] Inicializar o seu nó com a opção de RPC ligada.
- [ ] Apontar o seu código para a porta RPC do seu nó.
- [ ] Desbloquear a sua conta.
- [ ] Inicializar o miner.

Passando por todos estes passos, já deve ser possível utilizar o jogo da velha a partir de seu nó.

# 5 Próximos passos

Algumas ferramentas irão auxiliar no desenvolvimento, segue abaixo links para elas.

- [ ] https://truffleframework.com/
- [ ] https://embark.readthedocs.io/en/2.6.6/
- [ ] https://truffleframework.com/ganache
- [ ] https://plugins.jetbrains.com/plugin/9475-intellij-solidity
- [ ] https://marketplace.visualstudio.com/items?itemName=ConsenSys.Solidity

Links de referência:

- [ ] https://github.com/ethereum/go-ethereum
- [ ] http://solidity.readthedocs.io/en/v0.4.24/
- [ ] https://github.com/ethereum/web3.js/