

Deploy de Smart Contract na rede de testes do Ethereum

Autor: Ramon Rocha Rezende

Deploy de Smart Contract na rede de testes do Ethereum	1
Introdução	1
Requisitos	1
Criação de conta e sincronização	2
Adquirindo Ether	4
Deploy de smart contract	5
Interação com smart contract	7
Blockchain explorer	8
Referências	10

Introdução

Para fazer deploy de um smart contract na rede de testes do Ethereum precisaremos de uma conta na rede em que faremos o deploy e de um smart contract. Neste caso utilizaremos a rede de testes Rinkeby. Este exemplo foi feito usando Ubuntu 18.04 LTS e **não** foram utilizados frameworks como o *Truffle*.

Requisitos

Precisamos de um cliente para interagir com a rede e criar uma conta, para isso utilizaremos o *geth*. O *geth*, ou *Go Ethereum* é uma das três(junto com C++ e Python) implementações originais do protocolo do Ethereum. Ele é escrito em Go, totalmente open source e licenciado sob a GNU LGPL v3.

Um método fácil para instalação do *geth* é a instalação por pacotes binários, para isso basta adicionar o repositório do Ethereum e fazer a instalação por “apt-get”.

```
sudo add-apt-repository -y ppa:ethereum/ethereum  
sudo apt-get update  
sudo apt-get install ethereum
```

O passo a passo para instalação foi retirado da documentação original do *geth* encontrada em:

<https://geth.ethereum.org/install-and-build/Installing-Geth>

Além de poder interagir com a rede precisaremos também compilar o nosso smart contract. Os smart contracts do Ethereum são escritos em Solidity e rodam na EVM(Ethereum Virtual Machine), no entanto, primeiro precisamos transformar o código fonte em bytecode. Logo precisaremos do compilador de Solidity para compilar o código fonte e fazer o deploy. Um método simples de instalação do compilador é a utilização de pacotes binários, para isso basta adicionar o repositório do ethereum e fazer a instalação por “apt-get”.

```
sudo add-apt-repository ppa:ethereum/ethereum  
sudo apt-get update  
sudo apt-get install solc
```

O passo a passo para a instalação foi retirado da documentação original do Solidity encontrada em:

<https://solidity.readthedocs.io/en/latest/installing-solidity.html#building-from-source>

Criação de conta e sincronização

Com o ambiente preparado estamos prontos para começar. O primeiro objetivo é criar uma conta na rede de testes Rinkeby. Existem alguns métodos para isso como por exemplo a utilização de uma extensão de browser chamada Metamask, mas utilizaremos somente o *geth*.

Para criar uma conta com *geth* na rede de testes Rinkeby podemos executar o comando:

```
geth --rinkeby account new
```

Podemos ver as contas já criadas com:

```
geth account list
```

Ou para ver contas criadas na rede Rinkeby podemos executar:

```
geth --rinkeby account list
```

Primeiramente precisamos sincronizar com a rede na qual pretendemos realizar nossos testes, para isso podemos utilizar:

```
geth --rinkeby --syncmode light --allow-insecure-unlock -rpc --rpcapi db,eth,net,web3,personal
```

Vamos entender nosso comando, a primeira flag *--rinkeby* indica que vamos nos conectar na rede de testes Rinkeby do Ethereum(o que já era esperado). A flag *--syncmode* indica o modo de sincronização que utilizaremos, neste caso utilizamos o modo *light*, pois ele é o

mais rápido e consome menos recursos da máquina, no entanto, ele é um modo **experimental**, e portanto **extremamente instável**. Embora o modo *light* seja suficiente para nossos testes é recomendada a utilização de algum outro modo de sincronização para melhor estabilidade, como o *fast*.

A próxima flag é *--allow-insecure-unlock*, ela permite que uma conta possa ser desbloqueada com um servidor http rodando, esta flag não era necessária até uma recente atualização. Logo em seguida temos *--rpc*, que ativa o HTTP-RPC server e *--rpcapi* que permite passar API's que poderão ser utilizadas na interface HTTP-RPC.

OBS.: Esta configuração apresenta alta insegurança, tendo em vista que as RPCs estão expostas via HTTP.

Feito isso a sincronização será iniciada, é necessário que ela termine para que sejam feitas transações na rede.

A partir de outro terminal podemos iniciar o console do *geth* para já começarmos a executar alguns comandos. No segundo terminal execute o comando:

```
geth attach ipc:/home/user/.ethereum/rinkeby/geth.ipc
```

O arquivo *geth.ipc* é um unix socket, ele só existe enquanto o *geth* está executando e é através dele que podemos nos conectar ao primeiro processo do *geth* que criamos. Observe que o caminho para o arquivo ipc pode ser alterado.

A partir deste ponto, se tudo correr como esperado, devemos ter acesso ao console do *geth* em um terminal e a sincronização com a rede rodando em outro terminal. Neste caso, podemos executar alguns comandos no console para verificar a situação. Utilizando o módulo *eth* que nós passamos ao *geth* podemos ver as contas criadas. O comando a seguir mostra as contas criadas:

```
eth.accounts
```

Ele retorna um array de strings com os endereços das contas criadas.

Para verificar se a sincronização está sendo feita com a rede podemos executar o comando:

```
eth.syncing
```

Caso a sincronização esteja sendo feita, deverá ser impresso na tela algo como:

```
{  
  currentBlock: 4795331,  
  highestBlock: 4795499,  
  knownStates: 0,  
  pulledStates: 0,  
  startingBlock: 4790723  
}
```

Caso contrário, um simples “*false*” será impresso.

A partir daí teremos um momento chave, onde teremos um terminal aberto rodando a sincronização com a rede e outro com o console do *geth*.

Adquirindo Ether

A rede de testes Rinkeby utiliza o protocolo de consenso “proof-of-authority”, diferente da rede principal que utiliza o “proof-of-work”. Isso torna a rede não totalmente descentralizada, mas como é uma rede de testes isso não é um problema. Sendo assim, não é permitido que todos os nós da rede possam “minerar” Ether. Como o Ether não tem valor monetário na rede de testes isso evita ataques de *spam*.

No entanto, ainda precisamos de Ether para realizar qualquer tipo de transação de escrita na rede, bem como fazer deploy do nosso smart contract. Para conseguir Ether existem algumas contas que disponibilizam o Ether de forma gratuita. Estas contas são chamadas de *faucets*. O *faucet* da Rinkeby pode ser acessado em:

<https://faucet.rinkeb>

Se executarmos, dentro do console do *geth*, o comando:

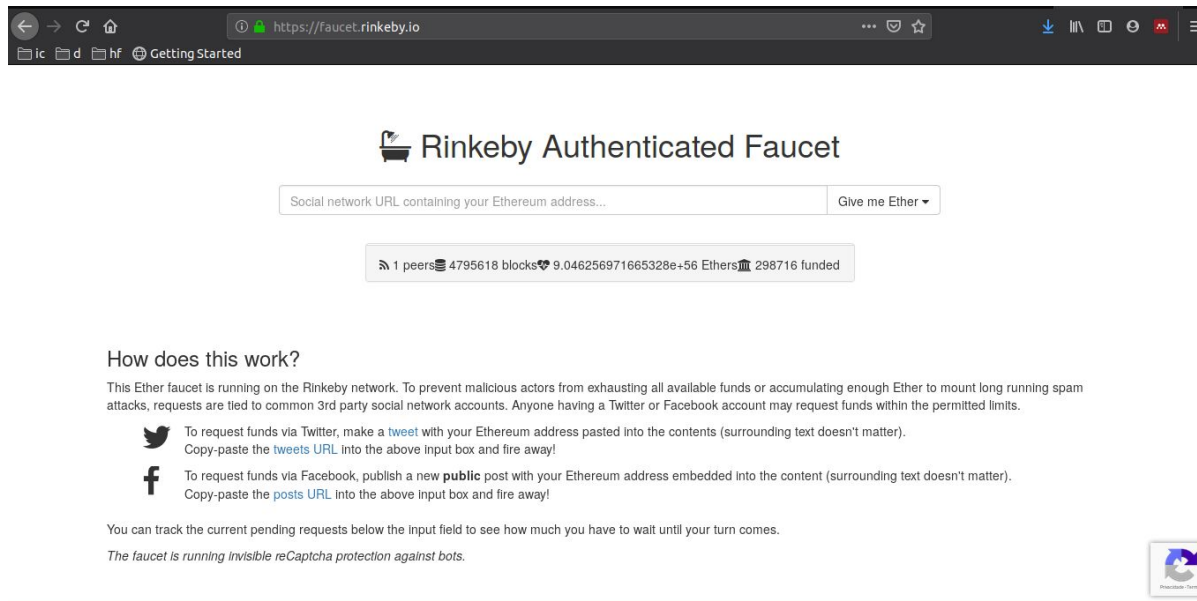
```
eth.getBalance(eth.accounts[0])
```

Retornamos a quantidade de Ether associada à primeira conta do nosso vetor de contas. O resultado é um número inteiro que representa a quantidade de *Wei* associada à conta. O *Wei* é a unidade mínima do ethereum, sendo que 1 *Ether* é equivalente a 10^{18} *Wei*. Para uma conta recém criada é esperado que esta função retorne zero.

Multiplier	Name
10^0	Wei
10^{12}	Szabo
10^{15}	Finney
10^{18}	Ether

Definição de Ether segundo Yellow Paper

Acessando a página do *faucet* podemos seguir as instruções para adquirir alguma quantidade de Ether.



É necessário fazer uma publicação em alguma rede social e colar o link da publicação no box indicado. Clicando em “*tweet*”, você será direcionado para um exemplo de publicação, ela basta alterar o “*0x00*” para o endereço da sua conta. É possível recuperar o endereço da conta com os comandos passados anteriormente, mas novamente, executar “*eth.accounts*” dentro do console do *geth* recupera as contas criadas.



Após feita a publicação é necessário colar o link dentro do box indicado no site do *faucet* e selecionar a quantidade de Ether desejada, observe que para cada quantidade existe um tempo de espera relacionado para a próxima solicitação.



Uma vez que os fundos foram transferidos a execução da função *eth.getBalance()* deverá retornar um número diferente de zero referente ao valor relacionado a conta.

A partir daí teremos um novo momento chave, onde teremos um terminal aberto rodando a sincronização com a rede, um outro com o console do *geth*, e uma conta desbloqueada e com Ether.

Deploy de smart contract

Para realizar transações na rede de testes é necessária uma conta desbloqueada, para isso utilizamos o comando:

```
personal.unlockAccount(eth.accounts[0])
```

Será solicitada a senha de desbloqueio que foi criada junto com a conta. Esta conta ficará desbloqueada por um período de tempo default, mas é possível especificar a quantidade de tempo do desbloqueio.

Em ordem de fazer deploy do smart contract precisamos primeiro compilar o mesmo. Para isso podemos abrir um novo terminal, navegar até a pasta onde está o código fonte do smart contract e executar o comando:

```
solc contract_name_file.sol --bin --abi --optimize -o compiled/
```

OBS.: Este comando deve ser executado no console do seu sistema linux, não dentro do console do *geth*

Este comando compila o contrato com nome de arquivo “ *contract_name_file.sol*”. A flag *--abi* indica que deve ser gerado um arquivo *.abi* com a interface do nosso contrato, a flag *--bin* indica que deve ser gerado um arquivo *.bin* com o bytecode do nosso contrato, e a flag *--optimize* habilita o otimizador de bytecode. O parâmetro *-o compiled/* indica que todas as saídas vão para a pasta */compiled*(esta pasta deve ser criada previamente). A interface do contrato e o bytecode serão utilizados posteriormente.

Voltando ao console do *geth* devemos criar uma variável para guardar o bytecode do nosso contrato e por conveniência criaremos uma para armazenar o endereço da nossa conta. Para isso podemos fazer:

```
var myAccount = eth.accounts[0]  
var contractBytecode = “Insira o bytecode do contrato aqui”
```

OBS.: É necessário adicionar *0x* no início da string para indicar o formato hexadecimal.

A partir daqui temos o necessário para fazer o deploy, só precisamos submeter uma nova transação(esta operação custará Ether, logo é necessário que se tenha fundos em sua conta). Podemos fazer isso executando:

```
var tx = eth.sendTransaction({from: myAccount, data: contractBytecode, gas: 470000})
```

Aqui passamos o endereço da nossa conta, o bytecode e a quantidade de gás. Observe que a quantidade de gás deve ser suficiente para a execução do contrato. É de certa forma

uma boa opção passar mais gas do que o necessário pois o que não for utilizado retorna para sua conta e problemas de enviar menos gas que o necessário são evitados. Se tudo correr corretamente já teremos feito o deploy do smart contract e executando o seguinte comando:

```
web3.eth.getTransactionReceipt(tx)
```

Se contrato tiver sido submetido será impressa uma saída como:

```
{
  blockHash:
    "0xfed7dcbd5e8c68e17bffa9f42cd30d95588674497ae719a04fd6a2ff219bb001d",
  blockNumber: 2534930,
  contractAddress: "0xbd3ffb07250634ba413e782002e8f880155007c8",
  cumulativeGasUsed: 1071323,
  from: "0x1db565576054af728b46ada9814b1452dd2b7e66",
  gasUsed: 458542,
  logs: [],
  logsBloom: "0x00000...",
  status: "0x1",
  to: null,
  transactionHash:
    "0x1a341c613c2f03a9bba32be3c8652b2d5a1e93f612308978bbff77ce05ab02c7",
  transactionIndex: 4
}
```

Interação com smart contract

Para interagir com um smart contract existente precisamos ter acesso ao seu endereço e à sua interface. Podemos ter acesso à interface do contrato utilizando o arquivo `.abi` criado após a compilação do código fonte.

Vamos associá-lo a uma variável:

```
var contractAbi = interface do contrato
```

OBS.: a variável `contractAbi` é um JSON object, logo não devem ser usadas aspas.

Exemplo:

```
var contractAbi =
[{"constant":true,"inputs":[],"name":"getCounter","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}, {"constant":false,"inputs":[],"name":"increment","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"}, {"inputs":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]
```

Feito isso também podemos associar o endereço do contrato a uma variável:

```
var contractAddress = "endereço do contrato"
```

O endereço do contrato é obtido quando é submetido e também pode ser recuperado através de um blockchain explorer.

Precisamos criar um objeto de contrato passando a nossa interface como parâmetro, podemos fazer isso executando a função:

```
var myContract = web3.eth.contract(contractAbi)
```

E a partir daí criar uma instância do contrato presente na rede através do endereço do contrato:

```
var contractInstance = myContract.at(contractAddress)
```

Feito isso precisamos de somente mais um passo para interagir com o contrato. Neste ponto, caso seja feita uma tentativa de execução de funções do contrato (de escrita), provavelmente iremos obter um erro. Isso ocorre porque nossa transação não sabe qual conta utilizar, para resolver isso podemos definir uma conta como padrão:

```
web3.eth.defaultAccount = eth.accounts[0]
```

Ou

```
web3.eth.defaultAccount = myAccount
```

Feito isso podemos executar as funções do nosso contrato normalmente. Lembrando que operações de leitura não custam Ether, mas para cada execução de função de escrita será descontada uma taxa do valor em sua conta.

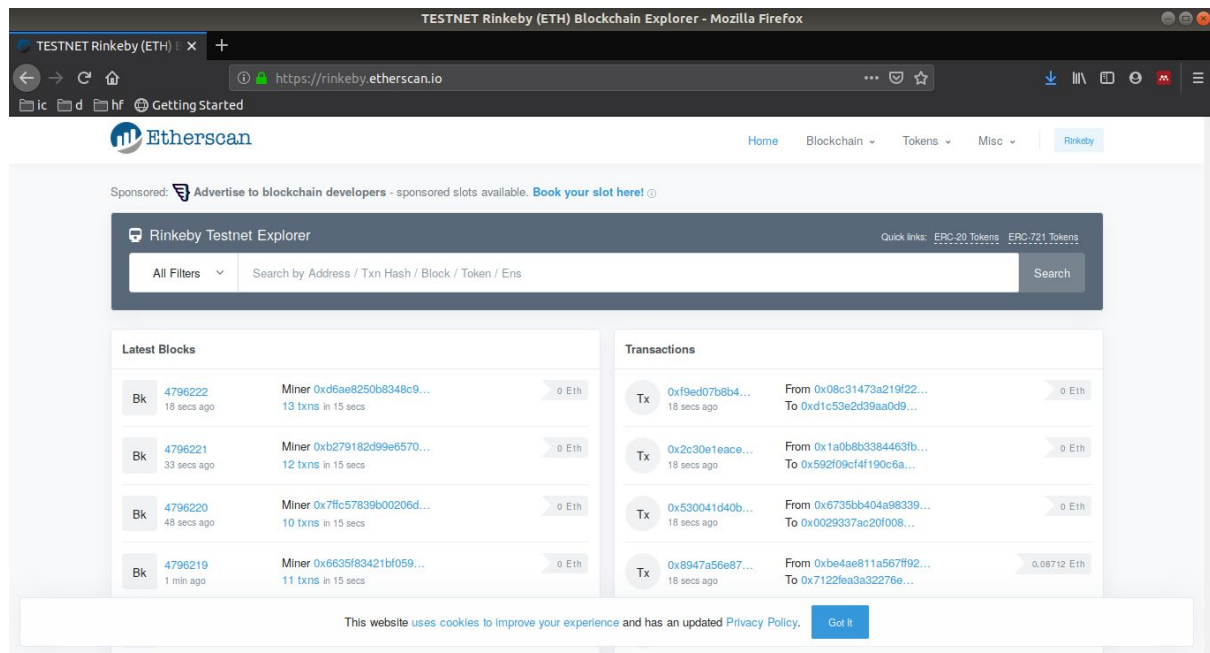
```
myContract.funcao_de_leitura()
```

```
myContract.funcao_de_escrita()
```

OBS.: Todas as operações dependem de estarmos devidamente sincronizados com a rede.

Blockchain explorer

A rede do Ethereum é pública, portanto todas as operações podem ser vistas pelo público. Neste caso podemos ver as operações de aquisição de fundos e criação de contrato, por exemplo. Para fazer isso basta acessar a página do Etherscan em: <https://rinkeby.etherscan.io/>



TESTNET Rinkeby (ETH) Blockchain Explorer - Mozilla Firefox

TESTNET Rinkeby (ETH) X +

https://rinkeby.etherscan.io

Etherscan

Home Blockchain Tokens Misc Rinkeby

Sponsored: Advertise to blockchain developers - sponsored slots available. [Book your slot here!](#)

Rinkeby Testnet Explorer

Quick links: [ERC-20 Tokens](#) [ERC-721 Tokens](#)

All Filters Search by Address / Txn Hash / Block / Token / Ens Search

Latest Blocks		
Bk	4796222 18 secs ago	Miner 0xd6ae8250b8348c9... 13 txns in 15 secs 0 Eth
Bk	4796221 33 secs ago	Miner 0xb279182d99e6570... 12 txns in 15 secs 0 Eth
Bk	4796220 48 secs ago	Miner 0x7ffc57839b00206d... 10 txns in 15 secs 0 Eth
Bk	4796219 1 min ago	Miner 0x6635f83421bf059... 11 txns in 15 secs 0 Eth

Transactions		
Tx	0xf9ed07b8b4... 18 secs ago	From 0x08c31473a219f22... To 0xd1c53e2d39aa0d9... 0 Eth
Tx	0x2c30e1eace... 18 secs ago	From 0x1a0b8b3384463fb... To 0x592f09c14f190c6a... 0 Eth
Tx	0x530041d40b... 18 secs ago	From 0x6735bb404a98339... To 0x0029337ac20f008... 0 Eth
Tx	0x8947a56e87... 18 secs ago	From 0xbe4ae811a567ff92... To 0x7122f6a3a32276e... 0.08712 Eth

This website uses cookies to improve your experience and has an updated Privacy Policy. [Get it](#)

Podemos fazer uma busca por uma conta ou por um contrato específico, basta ter o endereço correspondente.

Rinkeby Accounts, Addresses and Contracts - Mozilla Firefox

Rinkeby Accounts, Addr: x +

https://rinkeby.etherscan.io/address/0x4cb0e33aef7b1296a3b1cf11d206a1203badc9c4

Etherscan Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Address 0x4cb0e33aef7b1296a3b1cf11d206a1203badc9c4

Sponsored: Advertise to blockchain developers - sponsored slots available. [Book your slot here!](#)

Overview

Balance: 5.999625792 Ether

Token: \$0.00

More Info

My Name Tag: Not Available

Transactions Erc20 Token Txns

Latest 8 txns

Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0xb8ef0b7d7b6de77...	4795946	1 hr 16 mins ago	0x31b98d14007bdee...	IN 0x4cb0e33aef7b129...	3 Ether	0.000021
0x31c42cac584f090...	4790570	23 hrs 40 mins ago	0x4cb0e33aef7b129...	OUT 0x393c2836b8937cf...	0 Ether	0.000026874
0x172fed40e4517f40...	4790568	23 hrs 41 mins ago	0x4cb0e33aef7b129...	OUT 0x393c2836b8937cf...	0 Ether	0.000026874
0xcd02d70b45a8a32...	4790568	23 hrs 41 mins ago	0x4cb0e33aef7b129...	OUT 0x393c2836b8937cf...	0 Ether	0.000026874
0x2ba5b2bcf8ec835...	4790568	23 hrs 41 mins ago	0x4cb0e33aef7b129...	OUT 0x393c2836b8937cf...	0 Ether	0.000041874
0x98437a56b37243d...	4790462	1 day 7 mins ago	0x4cb0e33aef7b129...	OUT Contract Creation	0 Ether	0.000125856
0xd39b71a3dd6b8b6...	4783375	2 days 5 hrs ago	0x4cb0e33aef7b129...	OUT Contract Creation	0 Ether	0.000125856
0xfa0c66b74e8fb3bb...	4781690	2 days 12 hrs ago	0x31b98d14007bdee...	IN 0x4cb0e33aef7b129...	3 Ether	0.000021

[Download CSV Export]

Referências

<http://www.ethdocs.org/en/latest/>
<https://github.com/ethereum/go-ethereum/wiki>
<https://solidity.readthedocs.io/en/v0.5.10/>
<https://geth.ethereum.org/>