Aula 20 - Ethereum

Ambiente de Desenvolvimento

Prof. Rogério Aparecido Gonçalves[1]
rogerioag@utfpr.edu.br

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento de Computação (DACOM)
Campo Mourão - Paraná - Brasil

Programa de Pós Graduação em Ciência da Computação
Mestrado em Ciência da Computação
PPGCC17 - Tópicos em Redes de Computadores e Cibersegurança

Este trabalho está licenciado com uma Licença Creative Commons Teste - Atribuição-NãoComercial-SemDerivações 4.0 Internacional.

Agenda i

Introdução

Objetivos

- Apresentação do Ambiente de Desenvolvimento para *Ethereum*.

Ethereum – Redes de Teste i

Parâmetro	Descrição
`--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 network: pre-configured Proof of Work test network
`--rinkeby`	Rinkeby network: pre-configured Proof of Authority test network
`--goerli`	Görli network: pre-configured Proof of Authority test network
`--kiln`	Kiln network: pre-configured proof-of-work to proof-of-stake test network

Ethereum – Redes de Teste ii

`--sepolia`	Sepolia network: pre-configured proof-of-work test network

Ethereum – Redes de Teste iii

- Testando a execução:

```
geth --ropsten --syncmode snap --http --http.addr 127.0.0.1 --http.port 8559 --http
INFO [10-25|16:24:31.929] Starting Geth on Ropsten testnet...
```

- Estava dando o seguinte *Warning*:

```
WARN [10-25|17:23:25.126] Post-merge network, but no beacon c
```

Pesquisando na Internet: **Post-merge network, but no beacon client seen. Please launch one to follow the chain!**

- Encontrei essa solução na internet:

https://github.com/ethereum/go-ethereum/issues/25791

- Indicando a documentação do *Ethereum* sobre *Consensus Clients*

Ethereum – Redes de Teste iv

- Mostra como `geth` deve ser iniciado, com conexão RPC autenticada usando um arquivo `jwtsecret`.
- Por padrão esse arquivo está em `~/.ethereum/geth/jwtsecret`.

```
geth --ropsten --syncmode snap --http --http.addr 127.0.0.1 -
INFO [10-25|16:24:31.929] Starting Geth on Ropsten testnet...
```

- Note que estou executando na rede de testes `Ropsten`, 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 tem o parâmetro `--ropsten`, como tem o `--mainnet` e outras redes de testes.

Clientes 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`

- Testei o `Prysm` por ser escrito em `Go`, assim como o `Geth`.
- `Prysm` é uma implementação da especificação do consenso `proof-of-stake` do `Ethereum`.
- Este link apresenta como configurar o `Prism`:

Step 2: Instalando o Prysm#

- Crie no diretório `~/.ethereum`, duas subpastas: `consensus` e `execution`:
- Acesse o diretório `consensus` e execute o comando para baixar o cliente `Prysm` e transformá-lo em executável:

```
$ mkdir prysm && cd prysm
$ curl https://raw.githubusercontent.com/prysmaticlabs/prysm/
```

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/geth/jwtsecret`.
  - Usar o próprio `Prysm` para gerar o `jwt.hex`:

```
## Optional. This command is necessary only if you've previou

## Required
../prysm.sh beacon-chain generate-auth-secret
```

Step 3: Executando um Cliente de Execução#

- 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.
- Note que `Geth 1.10.22` contém uma regressão. Atualize para v1.10.23+ se você já não tiver uma mais nova.
- Tenho instalado a versão `1.10.25-stable`:

```
$ geth version
Geth
Version: 1.10.25-stable
Git Commit: 69568c554880b3567bace64f8848ff1be27d084d
Git Commit Date: 20220915
```

Step 4: Executando um nó beacon usando Prysm#

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

```
./prysm.sh beacon-chain --execution-endpoint=http://localhost
```

- Alterei o comando padrão para o conter o *hash* de uma das minhas contas:

```
./prysm.sh beacon-chain --execution-endpoint=http://localhost
```

- As contas podem ser listadas com o comando:

```
$ geth account list
```

- If you're running a validator, specifying a `suggested-fee-recipient` wallet address will allow you to earn what were previously miner transaction fee tips. See How to configure

Step 5: Executando um validator usando Prysm#

- Next, we'll create your validator keys with the Ethereum Staking Deposit CLI.
- Download the latest stable version of the deposit CLI from the Staking Deposit CLI Releases page.
- Run the following command to create your mnemonic phrase and keys:

```
./deposit new-mnemonic --num_validators=1 --mnemonic_language
```

- Follow the CLI prompts to generate your keys. This will give you the following artifacts:

1. A **new mnemonic seed phrase**. This is **highly sensitive** and should never be exposed to other people or networked hardware.

Congratulations!

- You're now running a **full Ethereum node** and a **validator**.
- It can a long time (from days to months) for your validator to become fully activated. To learn more about the validator activation process, see Deposit Process. See Check node and validator status for detailed status monitoring guidance.
- You can leave your **execution client**, **beacon node**, and **validator client** terminal windows open and running. Once your validator is activated, it will automatically begin proposing and validating blocks.

Atividade

- Leitura do Capítulo 13.

Word Cloud

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.

Próximas Aulas

Próximas Aulas

- Ambientes de Desenvolvimento e Ferramentas.

Referências

Referências i

Imran, Bashir. 2018. *Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.* Packt Publishing. https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1789486&lang=pt-br&site=eds-live&scope=site.