Лабораторная работа №2

Установка узла Ethereum в тестовой сети Goerli и низкоуровневое управление аккаунтами и транзакциями.

# Вступление

После перехода сети Ethereum на протокол Proof of Stake в 2022 году, данная лабораторная работа крайне усложнилась. Во-первых, усложнился процесс разворачивания локального узла, т.к. теперь требуется дополнительно установить модуль consensus client и синхронизировать его с execution client. Во-вторых, в результате увеличения пропускной способности сети резко усложнился сам процесс синхронизации, теперь он требует существенно время (порядка нескольких дней) и существенные аппаратные ресурсы (до 500 Гб SSD + достаточно широкий канал связи). В связи с этим данная работа переводится в разряд опциональной и выполняется при желании и наличии необходимых аппаратных ресурсов.

# Установка узла Ethereum

Для работы локального узла необходимы:

execution client - например geth

consensus client - например lighthouse

**Также необходимо быстрое (SSD) хранилище вместимостью минимум 500 Гб (тестовая сеть для примера на момент 2023 г. занимает 350-400 Гб)**

## Предварительная настройка

Создадим рабочие директории для хранения данных узла:

sudo mkdir -p /data/ethereum

cd /data

sudo mkdir -p lighthouse

cd ethereum/

Сгенерируем JWT secret для защищённого взаимодействия execution и consensus клиентов:

sudo mkdir -p secrets

openssl rand -hex 32 | tr -d "\n" | sudo tee /secrets/jwt.hex

## Установка geth

Для установки geth вводим следующие команды:

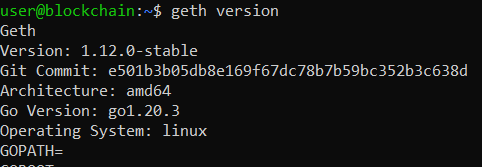
sudo add-apt-repository -y ppa:ethereum/ethereum

sudo apt-get update

sudo apt-get install ethereum

Для проверки и получения версии вводим:

geth version



Получить доступные параметры и команды geth можно, выполнив:

geth help

Пример команды для запуска узла тестовой сети goerli с активированной консолью и подключением http:

geth --goerli console --http --http.addr 127.0.0.1 --datadir /data/ethereum --authrpc.jwtsecret /data/ethereum/secrets/jwt.hex

## Установка lighthouse

Подробнее про работу с lighthouse можно прочитать [тут](https://lighthouse-book.sigmaprime.io/intro.html)

Перед установкой lighthouse - необходимо установить rust и некоторые утилиты:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

sudo apt install -y git gcc g++ make cmake pkg-config llvm-dev libclang-dev clang protobuf-compiler

Теперь скачиванием исходники и выполняем сборку в любую удобную вам папку:

git clone https://github.com/sigp/lighthouse.git

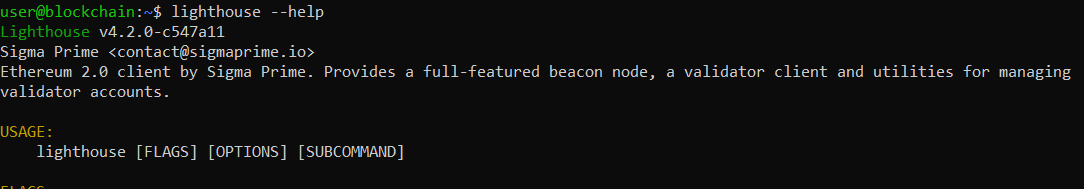
cd lighthouse

git checkout stable

make

Проверяем установку lighthouse:

lighthouse --help



Тут же отображаются доступные параметры запуска, пример команды для взаимодействия с узлом тестовой сети goerli:

lighthouse bn --network goerli --datadir /data/lighthouse --execution-endpoints http://127.0.0.1:8551 --checkpoint-sync-url https://goerli.checkpoint-sync.ethpandaops.io --http --http-address 127.0.0.1 --jwt-secrets /data/ethereum/secrets/jwt.hex --purge-db

## Мониторинг процесса синхронизации и взаимодействие с узлом

Процесс синхронизации занимает несколько полных дней. Отслеживать его можно по логам geth и lighthouse, а также через консоль geth. Для взаимодействия с узлом тестовой сети, настроенной ранее, подойдёт команда:

geth --goerli attach ipc://data/ethereum/geth.ipc

Для мониторинга состояния в консоли можно использовать следующие команды:

eth.syncing

net.peerCount

eth.blockNumber

eth.syncing и eth.blockNumber начнут отображать информацию после первичной синхронизации, она занимает несколько часов. А спустя несколько дней синхронизации, окончание этого процесса можно отследить также через eth.syncing, в конце он должен возвращать false.

## Создание и запуск демонов

На данный момент geth и lighthouse будут работать лишь пока запущены терминалы, в которых они выполняются. Для решения этой проблемы - их можно запустить в виде демонов.

Демон для geth создаётся следующим образом. Создаём новый юнит в systemd и копируем параметру снизу внутрь:

sudo systemctl edit --force --full Ethereum

[Unit]

Description=Ethereum Node Service

After=network.target

[Service]

User=user

Type=simple

ExecStart=/usr/bin/geth --goerli --http --http.addr 127.0.0.1 --datadir /data/ethereum --authrpc.jwtsecret /data/ethereum/secrets/jwt.hex

WorkingDirectory=/data/ethereum

Restart=on-failure

[Install]

WantedBy=multi-user.target

Для запуска вводим команды:

sudo systemctl daemon-reload

sudo systemctl enable ethereum

sudo systemctl start ethereum

Для отслеживания статуса и мониторинга логов соответственно вводим:

sudo systemctl status ethereum

journalctl -u ethereum -f -n 100

Аналогично создаётся демон для lighthouse, отличия будут лишь в названии и содержимом файла сервиса:

[Unit]

Description=Eth beacon chain service

After=network.target

[Service]

User=user

Type=simple

ExecStart=/home/user/.cargo/bin/lighthouse bn --network goerli --datadir /data/lighthouse --execution-endpoints http://127.0.0.1:8551 --checkpoint-sync-url https://goerli.checkpoint-sync.ethpandaops.io --http --http-address 127.0.0.1 --jwt-secrets /data/ethereum/secrets/jwt.hex -->WorkingDirectory=/data/lighthouse

Restart=on-failure

[Install]

WantedBy=multi-user.target

Мониторинг логов:

journalctl -u lighthouse-bc -f -n 100

P.S. Самая частая критическая ошибка, связанная с запуском этих сервисов - Permission denied /data/ethereum/geth/LOCK. Исправляется расширением доступа к созданным ранее папкам /data

## Подготовка окружения

В домашнем каталоге выполняем

mkdir labaEth

mkdir labaEth/keystore

Создаем два аккаунта (дважды выполняем!)

clef newaccount --keystore labaEth/keystore

нужно будет ввести пароль (запомните его, а лучше запишите!) и на выходе получите адрес аккаунта вида:

Generated account 0x48AfEc7F73b3e1F7E0728a965A37df283E423594

clef newaccount --keystore labaEth/keystore

…

Generated account 0x323346f9B5eCb00C4dff7B81A2f21699b1c75500

Запускаем кошелек (в отдельном терминале!)

clef --keystore labaEth/keystore --configdir labaEth/clef --chainid 5

Когда узел синхронизировался (см. выше), можно двигаться далее.

Идем на https://learnweb3.io/faucets/goerli/ и просим перечислить нам на созданный выше кошелек (например, 0x323346f9B5eCb00C4dff7B81A2f21699b1c75500) средств тестовой сети (понадобится аккаунт на Github’е).

В новом терминале подключаемся к узлу и пробуем управлять им:

geth --goerli attach ipc://data/ethereum/geth.ipc

Проверяем баланс

> web3.fromWei(eth.getBalance("0x48AfEc7F73b3e1F7E0728a965A37df283E423594"), "ether")

0

> web3.fromWei(eth.getBalance("0x323346f9B5eCb00C4dff7B81A2f21699b1c75500"), "ether")

0.1

Создаем транзакцию

> eth.sendTransaction({

from: "0x323346f9B5eCb00C4dff7B81A2f21699b1c75500",

to: "0x48AfEc7F73b3e1F7E0728a965A37df283E423594",

value: web3.toWei(0.01, "ether")

})

"0x6e17ac45bbab20979c0ff87adb4868a82566e0e5d5c002f4dada143141e8e8d0"

Внимание, после ввода команды взаимодействие в этом терминале прекратится до того, как вы не подтвердите подпись транзакции в вашем кошельке (который запущен в другом терминале). Так что, придется переключится в нужный терминал

В терминале кошелька придет сообщение об авторизации транзакции:

--------- Transaction request-------------

to: 0x48AfEc7F73b3e1F7E0728a965A37df283E423594

from: 0x323346f9B5eCb00C4dff7B81A2f21699b1c75500 [chksum ok]

value: 10000000000000000 wei

gas: 0x5208 (21000)

maxFeePerGas: 1500000014 wei

maxPriorityFeePerGas: 1500000000 wei

nonce: 0x0 (0)

chainid: 0x5

Accesslist

Request context:

NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:

User-Agent: ""

Origin: ""

-------------------------------------------

Approve? [y/N]:

> y

## Account password

Please enter the password for account 0x323346f9B5eCb00C4dff7B81A2f21699b1c75500

>

Здесь необходимо согласиться подписать транзакцию (ответ «y») и ввести пароль кошелька, после чего процесс продолжится:

-----------------------

Transaction signed:

{

"type": "0x2",

"nonce": "0x0",

"gasPrice": null,

"maxPriorityFeePerGas": "0x59682f00",

"maxFeePerGas": "0x59682f0e",

"gas": "0x5208",

"value": "0x2386f26fc10000",

"input": "0x",

"v": "0x1",

"r": "0x2803063401a9e9c579a8a9b4dc0087a09f69ce87baf524a3e49338ad2a9a11c",

"s": "0x7adad029430dd9a9527e2f4d0786082c27c958cdf3c48adf94eb1041707a485f",

"to": "0x48afec7f73b3e1f7e0728a965a37df283e423594",

"chainId": "0x5",

"accessList": [],

"hash": "0x6e17ac45bbab20979c0ff87adb4868a82566e0e5d5c002f4dada143141e8e8d0"

}

DEBUG[04-08|14:06:19.631] Served account\_signTransaction reqid=5 duration=17.494035099s

После успешного подписания транзакции нужно переключиться обратно в терминал, где запущена JS-консоль и увидеть, что команда успешно завершилась. После этого можно снова проверить балансы аккаунтов.

> web3.fromWei(eth.getBalance("0x323346f9B5eCb00C4dff7B81A2f21699b1c75500"), "ether")

0.089968499999853

> web3.fromWei(eth.getBalance("0x48AfEc7F73b3e1F7E0728a965A37df283E423594"), "ether")

0.01

Источники:

<https://geth.ethereum.org/docs/install-and-build/installing-geth>

<https://geth.ethereum.org/docs/getting-started>

<https://github.com/goerli/testnet>