Spin up a lite-node

Lite-nodes are a simplified node option that allow developers to perform lightweight tasks on a local node. This page covers how to spin-up a lite node on your local machine.

In this guide, we're going to use the Lotus Filecoin implementation. We'll show how to install a lite-node on MacOS and Ubuntu. For other Linux distributions, check out the Lotus documentation . To run a lite-node on Windows, install WLS with Ubuntu on your system and follow the Ubuntu instructions below.

Prerequisites

Lite-nodes have relatively lightweight hardware requirements – it's possible to run a lite-node on a Raspberry Pi 4. Your machine should meet the following hardware requirements:

1. At least 2 GiB of RAM
2. A dual-core CPU.
3.

To build the lite-node, you'll need some specific software. Run the following command to install the software prerequisites:

MacOS Ubuntu 1. Ensure you have XCode 2. and Homebrew 3. installed. 4. Install the following dependencies: 5.6. Copy 7. brewinstallgobzrjqpkg-confighwloccoreutilsrust 8. 9. 1. Install the following dependencies: 2.3. Copy 4. sudoaptupdate-y 5. sudoaptinstallmesa-opencl-icdocl-icd-opencl-devgccgitbzrjqpkg-configcurlclangbuild-essentialhwloclibhwloc-devwget-y 6. 7. Install Go 8. and add/usr/local/go/bin 9. to yourPATH 10. variable: 11. 12. Copy 13. wgethttps://go.dev/dl/go1.21.1.linux-amd64.tar.gz 14. sudorm-rf/usr/local/go&&sudotar-C/usr/local-xzfgo1.21.1.linux-amd64.tar.gz 15. echo'export PATH=PATH:/usr/local/go/bin'>>~/.bashrc&&source~/.bashrc 16. 17. Install Rust 18. and source the~/.cargo/env 19. config file: 20. 21. Copy 22. curl--proto'=https'--tlsv1.2-sSfhttps://sh.rustup.rs|sh 23. source"HOME/.cargo/env" 24. 25.

Pre-build

Before we can build the Lotus binaries, there's some setup we need to do. MacOS users should select their CPU architecture from the tabs:

MacOS Intel MacOS ARM Ubuntu 1. Clone the repository, and move into thelotus 2. directory: 3.4. Copy 5. gitclonehttps://github.com/filecoin-project/lotus.git 6. cdlotus/ 7. 8. Switch to the branch representing the network you want to use. Mainnet always uses thereleases 9. branch: 10. 11. Copy 12. gitcheckoutreleases 13. 14. Or you can checkout to the Calibration testnet release using thentwk/calibration 15. branch: 16. 17. Copy 18. gitcheckoutntwk/calibration 19. 20. Done! You can move on to theBuild 21. section. 22. 1. Clone the repository, and move into thelotus 2. directory: 3.4. Copy 5. gitclonehttps://github.com/filecoin-project/lotus.git 6. cdlotus 7. 8. Switch to the branch representing the network you want to use. Mainnet always uses thereleases 9. branch: 10. 11. Copy 12. gitcheckoutreleases 13. 14. Or you can checkout to the Calibration testnet release using thentwk/calibration 15. branch: 16. 17. Copy 18. gitcheckoutntwk/calibration 19. 20. Create the necessary environment variables to allow Lotus to run on M1 architecture: 21. 22. Copy 23. exportLIBRARY_PATH=/opt/homebrew/lib 24. exportFFI_BUILD_FROM_SOURCE=1 25. exportPATH="(brew--prefixcoreutils)/libexec/gnubin:/usr/local/bin:PATH" 26. 27. Done! You can move on to the Build 28. section. 29. 1. Clone the repository, and move into thelotus 2. directory: 3. 4. Copy 5. gitclonehttps://github.com/filecoin-project/lotus.git 6. cdlotus 7. 8. Switch to the branch representing the network you want to use. Mainnet always uses thereleases 9. branch: 10. 11. Copy 12. gitcheckoutreleases 13. 14. Or you can checkout to the Calibration testnet release using thentwk/calibration 15. branch: 16.17. Copy 18. gitcheckoutntwk/calibration 19. 20. If your processor was released later than an AMD Zen or Intel Ice Lake CPU, enable the use of SHA extensions by adding these two environment variables. If in doubt, ignore this command and move on to the next section 21. . 22.23. Copy 24. exportRUSTFLAGS="-C target-cpu=native -g" 25. exportFFI_BUILD_FROM_SOURCE=1 26. 27. Done! You can move on to the Build section. 28.

Build the binary

The last thing we need to do to get our node setup is to build the package. The command you need to run depends on which network you want to connect to:

Mainnet Calibration 1. Remove or delete any existing Lotus configuration files on your system: 2.3. Copy 4. mv~/.lotus~/.lotus-backup 5. 6. Make the Lotus binaries and install them: 7.8. Copy 9. makecleanall 10. sudomakeinstall 11. 12. Once the installation finishes, query the Lotus version to ensure everything is installed successfully and for the correct network: 13. 14. Copy 15. lotus--version 16. 17. This will output something like: 18.19. Copy 20. lotus version 1.19.1-dev+mainnet+git.94b621dd5 21. 22. 1. Remove or delete any existing Lotus configuration files on your system: 2. 3. Copy 4. mv~/.lotus~/.lotus-backup 5. 6. Make the Lotus binaries and install them: 7. 8. Copy 9. makeclean&&makecalibrationnet 10. sudomakeinstall 11. 12. Once the installation finishes, query the Lotus version to ensure everything is installed successfully and for the correct network: 13. 14. Copy 15. lotus--version 16. 17. This will output something like: 18. 19. Copy 20. lotus version 1.19.1-dev+calibrationnet+git.94b621dd5.dirty 21. 22.

Start the node

Let's start the lite-node by connecting to a remote full-node. We can use the public full-nodes from glif.io :

Mainnet Calibration 1. Create an environment variable calledFULLNODE_API_INFO 2. and set it to the WebSockets address of the node you want to connect to. At the same time, start the Lotus daemon with the--lite 3. tag: 4. 5. Copy 6. FULLNODE_API_INFO=wss://wss.mainnet.node.glif.io/apigw/lotuslotusdaemon--lite 7. 8. This will output something like: 9.10. Copy 11. 2023-01-26T11:18:54.251-0400 INFO main lotus/daemon.go:219 lotus repo: /Users/johnny/.lotus 12. 2023-01-26T11:18:54.254-0400 WARN cliutil util/apiinfo.go:94 API Token not set and requested, capabilities might be limited. 13. ... 14. 15. The Lotus daemon will continue to run in this terminal window. All subsequent commands we use should be done in a separate terminal window. 16. 1. Create an environment variable calledFULLNODE_API_INFO 2. and set it to the WebSockets address of the node you want to connect to. At the same time, start the Lotus daemon with the--lite 3. tag: 4. 5. Copy 6. FULLNODE_API_INFO=wss://wss.calibration.node.glif.io/apigw/lotuslotusdaemon--lite 7. 8. This will output something like: 9.10. Copy 11. 2023-01-26T11:18:54.251-0400 INFO main lotus/daemon.go:219 lotus repo: /Users/johnny/.lotus 12. 2023-01-26T11:18:54.254-0400 WARN cliutil util/apiinfo.go:94 API Token not set and requested, capabilities might be limited. 13. ... 14. 15. The Lotus daemon will continue to run in this terminal window. All subsequent commands we use should be done in a separate terminal window. 16.

Expose the API

To send JSON-RPC requests to our lite-node we need to expose the API.

Mainnet Calibration 1. Open~/.lotus/config.toml 2. and uncommentListenAddress 3. on line 6: 4.5. Copy 6. [API] 7. # Binding address for the Lotus API 8. # 9. # type: string 10. # env var: LOTUS_API_LISTENADDRESS 11. ListenAddress="/ip4/127.0.0.1/tcp/1234/http" 12. # type: string 13. # env var: LOTUS_API_REMOTELISTENADDRESS 14. # RemoteListenAddress = "" 15. ... 16. 17. Open the terminal window where your lite-node is running and pressCTRL 18. +c 19. to close the daemon. 20. In the same window, restart the lite-node: 21. 22. Copy 23. FULLNODE_API_INFO=wss://wss.mainnet.node.glif.io/apigw/lotuslotusdaemon--lite 24. 25. This will output something like: 26. 27. Copy 28. 2023-01-26T11:18:54.251-0400 INFO main lotus/daemon.go:219 lotus repo: /Users/johnny/.lotus 29. 2023-01-26T11:18:54.254-0400 WARN cliutil util/apiinfo.go:94 API Token not set and requested, capabilities might be limited 30. ... 31. 32. The Lotus daemon will continue to run in this terminal window. All subsequent commands we use should be done in a separate terminal window. 33. 1. Open~/.lotus/config.toml 2. and uncommentListenAddress 3. on line 6: 4. 5. Copy 6. [API] 7. # Binding address for the Lotus API 8. # 9. # type: string 10. # env var: LOTUS_API_LISTENADDRESS 11. ListenAddress="/ip4/127.0.0.1/tcp/1234/http" 12. # type: string 13. # env var: LOTUS_API_REMOTELISTENADDRESS 14. # RemoteListenAddress = "" 15. ... 16. 17. Open the terminal window where your lite-node is running and pressCTRL 18. +c 19. to close the daemon. 20. In the same window restart the lite-node: 21. 22. Copy 23. FULLNODE_API_INFO=wss://wss.calibration.node.glif.io/apigw/lotuslotusdaemon--lite 24. 25. This will output something like: 26.27. Copy 28. 2023-01-26T11:18:54.251-0400 INFO main lotus/daemon.go:219 lotus repo: /Users/johnny/.lotus 29. 2023-01-26T11:18:54.254-0400 WARN cliutil util/apiinfo.go:94 API Token not set and requested, capabilities might be limited. 30. ... 31. 32. The Lotus daemon will continue to run in this terminal window. All subsequent commands we use should be done in a separate terminal window. 33. The lite-node is now set up to accept local JSON-RPC requests! However, we don't have an authorization key, so we won't have access to privileged JSON-RPC methods.

Create a key

To access privileged JSON-RPC methods, like creating a new wallet, we need to supply an authentication key with our Curl requests.

1. Create a new admin token and set the result to a newLOTUS_ADMIN_KEY
2. environment variable:

3.
   ```
4. Copy
5. lotusauthcreate-token--perm"admin"
6. ```
7. This will output something like:
8. ```
9. Copy
10. eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJBbGxvdyI6WyJyZWFkIiwid3JpdGUiLCJzaWduIiwiYWRtaW4iXX0.um-LqY7g-SDOsMheDRbQ9JIaFzus_Pan0J88VQ6ZLVE
11. ```
12. Keep this key handy. We're going to use it in the next section.
13.

Send requests

Let's run a couple of commands to see if the JSON-RPC API is set up correctly.

1. First, let's grab the head of the Filecoin network chain:
2. ```
3. Copy
4. curl-XPOST'127.0.0.1:1234/rpc/v0'\
5. -H'Content-Type: application/json'\
6. --data'{"jsonrpc":"2.0","id":1,"method":"Filecoin.ChainHead","params":[]}'\
7. |jq
8. ```
9. This will output something like:
10. ```
11. Copy
   ```

```
{
"jsonrpc": "2.0",
"result": {
"Cids": [
{
"/": "bafy2bzacead2v2y6yob7rkm4y4snthibuamzy5a5iuzlwvy7rynemtkdywfuo"
},
{
"/": "bafy2bzaced4zahevivrcdoefqlh2j45sevfh5g3zsw6whpqxqjig6dxxf3ip6"
},
...
```
Next, let's try to create a new wallet. Since this is a privileged method, we need to supply our auth keyeyJhbGc...
:
```
Copy
curl-XPOST'127.0.0.1:1234/rpc/v0'\
-H'Content-Type: application/json'\
-H'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJBbGxvdyI6WyJyZWFkIiwid3JpdGUiLCJzaWduIiwiYWRtaW4iXX0.um-LqY7g-SDOsMheDRbQ9JIaFzus_Pan0J88VQ6ZLVE'\
--data'{"jsonrpc":"2.0","id":1,"method":"Filecoin.WalletNew","params":["secp256k1"]}'\
|jq
```
This will output something like:
```
Copy
{
"jsonrpc": "2.0",
"result": "t1vuc4eu2wgsdnce2ngygyzuxky3aqijqe7gj5qqa",
"id": 1
}
```
The result field is the public key for our address. The private key is stored within our lite-node.
Set the new address as the default wallet for our lite-node:
```
Copy
curl-XPOST'127.0.0.1:1234/rpc/v0'\
-H'Content-Type: application/json'\
-H'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJBbGxvdyI6WyJyZWFkIiwid3JpdGUiLCJzaWduIiwiYWRtaW4iXX0.um-LqY7g-SDOsMheDRbQ9JIaFzus_Pan0J88VQ6ZLVE'\
--data'{"jsonrpc":"2.0","id":1,"method":"Filecoin.WalletSetDefault","params":["t1vuc4eu2wgsdnce2ngygyzuxky3aqijqe7gj5qqa"]}'\
|jq
```
This will output something like:
```
Copy
{
"jsonrpc": "2.0",
"id": 1
}
```

Next steps

You should now have a local lite-node connected to a remote full-node with an admin API key! You can use this setup to continue playing around with the JSON-RPC, or start building your applications on Filecoin!

Previous Lite-nodes Next Fundamentals

Last updated4 months ago