title: How to turn your Raspberry Pi 4 into a node just by flashing the MicroSD card description: Flash your Raspberry Pi 4, plug in an ethernet cable, connect the SSD disk and power up the device to turn the Raspberry Pi 4 into a full Ethereum node + validator author: "EthereumOnArm" tags: ["clients", "execution layer", "consensus layer", "nodes"] lang: en skill: intermediate published: 2022-06-10 source: Ethereum on ARM sourceUrl: https://ethereum-on-arm-documentation.readthedocs.io/en/latest/

Ethereum on Arm is a custom Linux image that can turn a Raspberry Pi into an Ethereum node.

To use Ethereum on Arm to turn a Raspberry Pi into an Ethereum node, the following hardware is recommended:

- Raspberry 4 (model B 8GB), Odroid M1 or Rock 5B (8GB/16GB RAM) board
- MicroSD Card (16 GB Class 10 minimum)
- 2 TB SSD minimum USB 3.0 disk or an SSD with a USB to SATA case.
- Power supply
- Ethernet cable
- Port forwarding (see clients for further info)
- A case with heatsink and fan
- USB keyboard, Monitor and HDMI cable (micro-HDMI) (Optional)

Why run Ethereum on ARM? {#why-run-ethereum-on-arm}

ARM boards are very affordable, flexible, small computers. They are good choices for running Ethereum nodes because they can be bought cheaply, configured so that all their resources focus just on the node, making them efficient, they consume low amounts of power and are physically small so they can fit unobtrusively in any home. It is also very easy to spin up nodes because the Raspberry Pi's MicroSD can simply be flashed with a prebuilt image, with no downloading or building software required.

How does it work? {#how-does-it-work}

The Raspberry Pi's memory card is flashed with a prebuilt image. This image contains everything needed to run an Ethereum node. With a flashed card, all the user needs to do is power-on the Raspberry Pi. All the processes required to run the node are automatically started. This works because the memory card contains a Linux-based operating system (OS) on top of which system-level processes are automatically run that turn the unit into an Ethereum node.

Ethereum cannot be run using the popular Raspberry Pi Linux OS "Raspbian" because Raspbian still uses a 32-bit architecture which leads Ethereum users to run into memory issues and consensus clients do not support 32-bit binaries. To overcome this, the Ethereum on Arm team migrated to a native 64-bit OS called "Armbian".

Images take care of all the necessary steps, from setting up the environment and formatting the SSD disk to installing and running the Ethereum software as well as starting the blockchain synchronization.

Note on execution and consensus clients {#note-on-execution-and-consensus-clients}

The Ethereum on Arm image includes prebuilt execution and consensus clients as services. An Ethereum node requires both clients to be synced and running. You are only required to download and flash the image and then start the services. The image is preloaded with the following execution clients:

- Geth
- Nethermind
- Besu

and the following consensus clients:

- Lighthouse
- Nimbus
- Prysm
- Teku

You should choose one of each to run - all execution clients are compatible with all consensus clients. If you do not explicitly select a client, the node will fall back to its defaults - Geth and Lighthouse - and run them automatically when the board is powered up. You must open port 30303 on your router so Geth can find and connect to peers.

Downloading the Image {#downloading-the-image}

The Raspberry Pi 4 Ethereum image is a "plug and play" image that automatically installs and sets up both the execution and consensus clients, configuring them to talk to each other and connect to the Ethereum network. All the user needs to do is start their processes using a simple command.

Download the Raspberry Pi image from Ethereum on Arm and verify the SHA256 hash:

```sh

# From directory containing the downloaded image
```

shasum -a 256 ethonarm_22.04.00.img.zip

Hash should output: fb497e8f8a7388b62d6e1efbc406b9558bee7ef46ec7e53083630029c117444f

~~~

Note that images for Rock 5B and Odroid M1 boards are available at the Ethereum-on-Armdownloads page.

Flashing the MicroSD {#flashing-the-microsd}

The MicroSD card that will be used for the Raspberry Pi should first be inserted into a desktop or laptop so it can be flashed. Then, the following terminal commands will flash the downloaded image onto the SD card:

```shell

# check the MicroSD card name

sudo fdisk -l

sdxxx ```

It is really important to get the name correct because the next command includes `dd` which completely erases the existing content of the card before pushing the image onto it. To continue, navigate to the directory containing the zipped image:

```shell

# unzip and flash image

unzip ethonarm_22.04.00.img.zip sudo dd bs=1M if=ethonarm_22.04.00.img of=/dev/ conv=fdatasync status=progress ```

The card is now flashed, so it can be inserted into the Raspberry Pi.

## Start the node {#start-the-node}

With the SD card inserted into the Raspberry Pi, connect the ethernet cable and SSD then switch the power on. The OS will boot up and automatically start performing the preconfigured tasks that turn the Raspberry Pi into an Ethereum node, including installing and building the client software. This will probably take 10-15 minutes.

Once everything is installed and configured, log in to the device via an ssh connection or using the terminal directly if a monitor and keyboard is attached to the board. Use the `ethereum` account to log in, as this has permissions required to start the node.

~~~
shell User: ethereum Password: ethereum
```

The default execution client, Geth, will start automatically. You can confirm this by checking the logs using the following terminal command:

```
sh sudo journalctl -u geth -f
```

The consensus client does need to be started explicitly. To do this, first open port 9000 on your router so that Lighthouse can find and connect to peers. Then enable and start the lighthouse service:

```
sh sudo systemctl enable lighthouse-beacon sudo systemctl start lighthouse-beacon
```

Check the client using the logs:

```
sh sudo journalctl -u lighthouse-beacon
```

Note that the consensus client will sync in a few minutes because it uses checkpoint sync. The execution client will take longer - potentially several hours, and it will not start until the consensus client is already finished syncing (this is because the execution client needs a target to sync to, which the synced consensus client provides).

With the Geth and Lighthouse services running and synced, your Raspberry Pi is now an Ethereum node! It is most common to interact with the Ethereum network using Geth's Javascript console, which can be attached to the Geth client on port 8545. It is also possible to submit commands formatted as JSON objects using a request tool such as Curl. See more in the Geth documentation.

Geth is preconfigured to report metrics to a Grafana dashboard which can be viewed in the browser. More advanced users might wish to use this feature to monitor the health of their node by navigating to `ipaddress:3000`, passing `user: admin` and `passwd: ethereum`.

Validators {#validators}

A validator can also be optionally added to the consensus client. The validator software allows your node to participate actively in consensus and provides the network with cryptoeconomic security. You get rewarded for this work in ETH. To run a validator, you must first have 32 ETH, which must be deposited
```

into the deposit contract. **This is a long-term commitment - it is not yet possible to withdraw this ETH!** The deposit can be made by following the step-by-step guide on the Launchpad. Do this on a desktop/laptop, but do not generate keys — this can be done directly on the Raspberry Pi.

Open a terminal on the Raspberry Pi and run the following command to generate the deposit keys:

```
sudo apt-get update sudo apt-get install staking-deposit-cli cd && deposit new-mnemonic --num_validators 1
```

Keep the mnemonic phrase safe! The command above generated two files in the node's keystore: the validator keys and a deposit data file. The deposit data needs to be uploaded into the launchpad, so it must be copied from the Raspberry Pi to the desktop/laptop. This can be done using an ssh connection or any other copy/paste method.

Once the deposit data file is available on the computer running the launchpad, it can be dragged and dropped onto the on the launchpad screen. Follow the instructions on the screen to send a transaction to the deposit contract.

Back on the Raspberry Pi, a validator can be started. This requires importing the validator keys, setting the address to collect rewards, and then starting the preconfigured validator process. The example below is for Lighthouse—instructions for other consensus clients are available on the Ethereum on Arm docs:

```shell

## import the validator keys

lighthouse account validator import --directory=/home/ethereum/validator_keys

## set the reward address

sudo sed -i 's/' /etc/ethereum/lighthouse-validator.conf

## start the validator

sudo systemctl start lighthouse-validator ```

Congratulations, you now have a full Ethereum node and validator running on a Raspberry Pi!

### More details {#more-details}

This page gave an overview of how to set up a Geth-Lighthouse node and validator using Raspberry Pi. More detailed instructions are available on the Ethereum-on-Arm website.

### Feedback appreciated {#feedback-appreciated}

We know the Raspberry Pi has a massive user base that could have a very positive impact on the health of the Ethereum network. Please dig into the details in this tutorial, try running on testnets, check out the Ethereum on Arm GitHub, give feedback, raise issues and pull requests and help advance the technology and documentation!

### References {#references}

1. https://ubuntu.com/download/raspberry-pi
2. https://wikipedia.org/wiki/Port_forwarding
3. https://prometheus.io
4. https://grafana.com
5. https://forum.armbian.com/topic/5565-zram-vs-swap/
6. https://geth.ethereum.org
7. https://nethermind.io
8. https://www.hyperledger.org/projects/besu
9. https://github.com/prysmaticlabs/prysm
10. https://lighthouse.sigmaprime.io
11. https://ethersphere.github.io/swarm-home
12. https://raiden.network
13. https://ipfs.io
14. https://status.im
15. https://vipnode.org
```