

[docs](#) / [getting-started](#) / [consensus-clients](#)

# Connecting to Consensus Clients

Last edited on April 25, 2023

Geth is an [execution client](#). Historically, an execution client alone was enough to run a full Ethereum node. However, since Ethereum swapped from [proof-of-work](#) (PoW) to [proof-of-stake](#) (PoS) based consensus, Geth needs to be coupled to another piece of software called a "[consensus client](#)".

There are five consensus clients available, all of which connect to Geth in the same way. This page will outline how Geth can be set up with a consensus client to form a complete Ethereum node.

## Configuring Geth

Geth can be downloaded and installed according to the instructions on the [Installing Geth](#) page. In order to connect to a consensus client, Geth must expose a port for the inter-client RPC connection.

The RPC connection must be authenticated using a `jwtsecret` file. This is created and saved to `<datadir>/geth/jwtsecret` by default but can also be created and saved to a custom location or it can be self-generated and provided to Geth by passing the file path to `--authrpc.jwtsecret`. The `jwtsecret` file is required by both Geth and the consensus client.

The authorization must then be applied to a specific address/port. This is achieved by passing an address to `--authrpc.addr` and a port number to `--authrpc.port`. It is also safe to provide either `localhost` or a wildcard `*` to `--authrpc.vhosts` so that incoming requests from virtual hosts are accepted by Geth because it only applies to the port authenticated using `jwtsecret`.

A complete command to start Geth so that it can connect to a consensus client looks as follows:

```
geth --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost --
```

## Consensus clients

There are currently five consensus clients that can be run alongside Geth. These are:

[Lighthouse](#): written in Rust

[Nimbus](#): written in Nim

[Prysm](#): written in Go

[Teku](#): written in Java

[Lodestar](#): written in Typescript

It is recommended to consider [client diversity](#) when choosing a consensus client. Instructions for installing each client are provided in the documentation linked in the list above.

The consensus client must be started with the right port configuration to establish an RPC connection to the local Geth instance. In the example above, `localhost:8551` was authorized for this purpose. The consensus clients all have a command similar to `--http-webprovider` that takes the exposed Geth port as an argument.

The consensus client also needs the path to Geth's `jwt-secret` in order to authenticate the RPC connection between them. Each consensus client has a command similar to `--jwt-secret` that takes the file path as an argument. This must be consistent with the `--authrpc.jwtsecret` path provided to Geth.

The consensus clients all expose a [Beacon API](#) that can be used to check the status of the Beacon client or download blocks and consensus data by sending requests using tools such as [Curl](#). More information on this can be found in the documentation for each consensus client.

## Validators

Validators are responsible for securing the Ethereum blockchain. Validators have staked at least 32 ETH into a deposit contract and run validator software. Each consensus client has its own validator software that is described in detail in its respective documentation. The easiest way to handle staking and validator key generation is to use the Ethereum Foundation [Staking Launchpad](#). The Launchpad guides users through the process of generating validator keys and connecting the validator to the consensus client.

## Syncing

Geth cannot sync until the connected consensus client is synced. This is because Geth needs a target head to sync to. The fastest way to sync a consensus client is using checkpoint sync. To do this, a checkpoint or a url to a checkpoint provider can be provided to the consensus client on startup. There are several sources for these checkpoints. The ideal scenario is to get one from a trusted node operator, organized out-of-band, and verified against a third node or a block explorer or checkpoint provider. Some clients also allow checkpoint syncing by HTTP API access to an existing Beacon node. There are also several [public checkpoint sync endpoints](#).

Please see the pages on [syncing](#) for more detail. For troubleshooting, please see the [Syncing](#) section on the [console log messages](#) page.




# Using Geth

Geth is the portal for users to send transactions to Ethereum. The Geth Javascript console is available for this purpose, and the majority of the [JSON-RPC API](#) will remain available via web3js or HTTP requests with commands as json payloads. These options are explained in more detail on the [Javascript Console page](#). The Javascript console can be started using the following command in a separate terminal (assuming Geth's IPC file is saved in `datadir`):

```
geth attach datadir/geth.ipc
```

## Summary

Now that Ethereum has implemented proof-of-stake, Geth users are required to install and run a consensus client. Otherwise, Geth will not be able to track the head of the chain. There are five consensus clients to choose from. This page provided an overview of how to choose a consensus client and configure Geth to connect to it.

| Downloads   |   | Documentation   |  |
|---|---|---|--|
|  |  |  |  |
| © 2013–2023. The go-ethereum Authors   <a href="#">Do-not-Track</a>                 |   |   |  |