Sui Full Node Configuration

INFO

These instructions are for advanced users. If you just need a local development environment, you should instead follow the instructions in Create a Local Sui Network to create a local Full node, validators, and faucet.

Sui Full nodes validate blockchain activities, including transactions, checkpoints, and epoch changes. Each Full node stores and services the queries for the blockchain state and history.

This role enables validators to focus on servicing and processing transactions. When a validator commits a new set of transactions (or a block of transactions), the validator pushes that block to all connected Full nodes that then service the queries from clients.

Features

Sui Full nodes:

- Track and verify the state of the blockchain, independently and locally.
- Serve read requests from clients.

State synchronization

Sui Full nodes sync with validators to receive new transactions on the network.

A transaction requires a few round trips to 2f+1 validators to form a transaction certificate (TxCert).

This synchronization process includes:

1. Following 2f+1 validators and listening for newly committed transactions.
2. Making sure that 2f+1 validators recognize the transaction and that it reaches finality.
3. Executing the transaction locally and updating the local DB.

This synchronization process requires listening to at a minimum 2f+1 validators to ensure that a Full node has properly processed all new transactions. Sui will improve the synchronization process with the introduction of checkpoints and the ability to synchronize with other Full nodes.

Architecture

A Sui Full node is essentially a read-only view of the network state. Unlike validator nodes, Full nodes cannot sign transactions, although they can validate the integrity of the chain by re-executing transactions that a quorum of validators previously committed.

Today, a Sui Full node maintains the full history of the chain.

Validator nodes store only the latest transactions on the frontier of the object graph (for example, transactions with >0 unspent output objects).

Full node setup

Follow the instructions here to run your own Sui Full node. Sui Full nodes run using the `sui-node` binary.

Hardware requirements

Suggested minimum hardware to run a Sui Full node:

- CPUs: 8 physical cores / 16 vCPUs
- RAM: 128 GB
- Storage (SSD): 4 TB NVMe drive

Software requirements

Sui recommends running Sui Full nodes on Linux. Sui supports the Ubuntu and Debian distributions. You can run a Sui Full node on macOS, but this is only recommended for development and not for production use.

Make sure to update Rust.

Use the following command to install additional Linux dependencies.

```
$ sudo apt-get update \
&& sudo apt-get install -y --no-install-recommends \
tzdata \
libprotobuf-dev \
ca-certificates \
build-essential \
libssl-dev \
libclang-dev \
libpq-dev \
pkg-config \
openssl \
protobuf-compiler \
git \
clang \
cmake
```

Running a Full node

Instructions for building, installing, or downloading the `sui-node` binary are available at Sui Install. These install instructions are specific to the `sui` cli, but apply to the `sui-node` binary as well.

There are many ways to run a Sui Full node (bare metal, virtual machine, Kubernetes statefulset, and so on), and the solution that you choose depends on your specific needs as well as the infrastructure that you have available.

There are some specific considerations to keep in mind when running a Sui Full node that apply to all environments:

- Genesis: You must download the genesis blob for the network that you want to connect to, and make it available to the `sui-node`.
- Data Storage: Sui Full nodes *can* require a significant amount of disk space to store the blockchain history. If you plan to use your Full node to serve RPC requests, you must also plan for the storage of index files, which requires a significant amount of disk space.
- Monitoring: Sui Full nodes expose metrics about the node's health and the state of the Sui network.
- Updates: Sui Full nodes must be updated to the latest version to remain in sync with the network.
- Archival Fallback: The archival fallback allows you to sync checkpoints from any point in the chain's history. The network `seed-peers` below only keep a few epochs of history.

Using Docker Compose

There's a guide in the Sui repository on running a Full node via Docker Compose. This alone is not suitable for a production environment, but can be used to get a Full node up and running quickly on a virtual machine or local machine for development purposes. Refer to Running a Full node for instructions relevant to production use cases.

Setting up a Full node

When you are ready to run `sui-node` in your production environment, you can set up your Full node by completing the following steps:

1. Make a copy of the Full node YAML template: `cp crates/sui-config/data/fullnode-template.yaml fullnode.yaml`

2. Download the genesis blob for the network to use:

   - Devnet genesis blob: `curl -fLJO https://github.com/MystenLabs/sui-genesis/raw/main/devnet/genesis.blob`
   - Testnet genesis blob: `curl -fLJO https://github.com/MystenLabs/sui-genesis/raw/main/testnet/genesis.blob`
   - Mainnet genesis blob: `curl -fLJO https://github.com/MystenLabs/sui-genesis/raw/main/mainnet/genesis.blob`

3. For Testnet or Mainnet: Edit the `fullnode.yaml` file to include peer nodes for state synchronization. Append the following to the end of the current configuration:

Mainnet   Testnet

```
p2p-config:
  seed-peers:
    - address: /dns/mel-00.mainnet.sui.io/udp/8084
      peer-id:
d32b55bdf1737ec415df8c88b3bf91e194b59ee3127e3f38ea46fd88ba2e7849
    - address: /dns/ewr-00.mainnet.sui.io/udp/8084
      peer-id:
c7bf6cb93ca8fdda655c47ebb85ace28e6931464564332bf63e27e90199c50ee
    - address: /dns/ewr-01.mainnet.sui.io/udp/8084
      peer-id:
3227f8a05f0faa1a197c075d31135a366a1c6f3d4872cb8af66c14dea3e0eb66
    - address: /dns/lhr-00.mainnet.sui.io/udp/8084
      peer-id:
c619a5e0f8f36eac45118c1f8bda28f0f508e2839042781f1d4a9818043f732c
    - address: /dns/sui-mainnet-ssfn-1.nodeinfra.com/udp/8084
      peer-id:
0c52ca8d2b9f51be4a50eb44ace863c05aadc940a7bd15d4d3f498deb81d7fc6
    - address: /dns/sui-mainnet-ssfn-2.nodeinfra.com/udp/8084
      peer-id:
1dbc28c105aa7eb9d1d3ac07ae663ea638d91f2b99c076a52bbded296bd3ed5c
    - address: /dns/sui-mainnet-ssfn-ashburn-na.overclock.run/udp/8084
      peer-id:
5ff8461ab527a8f241767b268c7aaf24d0312c7b923913dd3c11ee67ef181e45
    - address: /dns/sui-mainnet-ssfn-dallas-na.overclock.run/udp/8084
      peer-id:
e1a4f40d66f1c89559a195352ba9ff84aec28abab1d3aa1c491901a252acefa6
    - address: /dns/ssn01.mainnet.sui.rpcpool.com/udp/8084
      peer-id:
fadb7ccb0b7fc99223419176e707f5122fef4ea686eb8e80d1778588bf5a0bcd
    - address: /dns/ssn02.mainnet.sui.rpcpool.com/udp/8084
      peer-id:
13783584a90025b87d4604f1991252221e5fd88cab40001642f4b00111ae9b7e
```

4. Optional: Set up the Archival Fallback, which allows you to sync checkpoints if you fall behind the network's `seed-peers`.

5. Optional: Skip this step to accept the default paths to resources. Edit the fullnode.yaml file to use custom paths.

6. Update the `db-path` field with the path to the Full node database. `db-path: "/db-files/sui-fullnode"`

7. Update the `genesis-file-location` with the path to genesis.blob.

```
genesis:
    genesis-file-location: "/sui-fullnode/genesis.blob"
```

Starting your Full node

You should not start syncing your Full node from the start of the genesis. This will take a very long time and consume a lot of resources (including likely filling up your disk).

Instead, start your Full node from a recent snapshot. You can find details on how to obtain a snapshot from the Sui Snapshots guide.

Now that you have your Full node config file set up, and you've obtained a snapshot, you can start your Full node by running the `sui-node` binary with your `fullnode.yaml` configuration file:

```
$ sui-node --config-path fullnode.yaml
```

It's a good idea to use something like systemd to manage your Full node in a production environment.

Troubleshooting

If you receive a `cannot find -lpq` error, you are missing the `libpq` library. Use `sudo apt-get install libpq-dev` to install on Linux, or `brew install libpq` on MacOS. After you install on MacOS, create a Homebrew link using `brew link --force libpq`. For further context, reference the issue on Stack Overflow.

If you receive the following error:

```
panicked at error binding to 0.0.0.0:9184: error creating server listener:
Address already in use (os error 98)
```

Then update the metrics address in your fullnode.yaml file to use port `9180`.

```
metrics-address: "0.0.0.0:9180"
```

Edit this page