Installation

Step 1: Node Installation

Step 1.1: Node Configuration

This step provides procedures to install BandChain's executable and sync blocks with other peers.

Assuming to run on Ubuntu 22.04 LTS allowing connection on port 26656 for P2P connection.

Before beginning instructions, the following variables should be set to be used in further instructions.Please make sure that these variables are set every time when using the new shell session.

Chain ID of Laozi Mainnet

export

CHAIN_ID

laozi-mainnet

Wallet name to be used as validator's account, please change this into your name (no whitespace).

export

WALLET_NAME

< YOUR_WALLET_NAME

Name of your validator node, please change this into your name.

export

MONIKER

< YOUR_MONIKER

URL of genesis file for Laozi Mainnet

export

GENESIS_FILE_URL

https://raw.githubusercontent.com/bandprotocol/launch/master/laozi-mainnet/genesis.json

Data sources/oracle scripts files

export

BIN_FILES_URL

https://raw.githubusercontent.com/bandprotocol/launch/master/laozi-mainnet/files.tar.gz The following applications are required to build and run the BandChain node.

- make, gcc, g++ (can be obtained from the build-essential package on linux)
- wget, curl for downloading files

install required tools

sudo

apt-get update &&

\ sudo

apt-get upgrade -y &&

\ sudo

apt-get

install -y build-essential curl

wget jq * Install Go 1.19.1

Install Go 1.19.1

wget https://go.dev/dl/go1.19.1.linux-amd64.tar.gz tar xf go1.19.1.linux-amd64.tar.gz sudo

mv go /usr/local/go

Set Go path to PATH variable

echo

"export PATH= PATH :/usr/local/go/bin:~/go/bin"

HOME /.profile source ~/.profile Go binary should be at /usr/local/go/bin and any executable compiled by go install command should be at ~/go/bin

Step 1.2: Clone & Install BandChain Laozi

cd ~

Clone BandChain Laozi version v2.5.4

git clone https://github.com/bandprotocol/chain cd chain git fetch &&

git checkout v2.5.4

Install binaries to GOPATH/bin

make

install

Step 1.3: Initialize the BandChain and download the genesis file

cd

HOME

Initialize configuration and genesis state

bandd init --chain-id CHAIN_ID

" MONIKER "

Replace genesis file with our genesis file

wget

GENESIS_FILE_URL -O HOME /.band/config/genesis.json

Download data sources / oracle scripts files, and store in HOME/.band/files

wget -qO- BIN_FILES_URL

|

tar xvz -C HOME /.band/

Create new account

bandd keys add

WALLET_NAME

Step 1.4: Configure general settings

This can be done by editing fields inHOME/.band/config/config.toml andHOME/.band/config/app.toml .

For a list of seeds and peers, please seehere .

For minimum-gas-prices, you can set any number but the recommendation is 0.0025uband (according tothis proposal).

List of seeds and persistent peers you want to add

e.g. SEEDS="8d42bdcb6cced03e0b67fa3957e4e9c8fd89015a@34.87.86.195:26656,543e0cab9c3016

export

SEEDS

",,..." export

PERSISTENT_PEERS

",,..."

Add seeds and persistent peers to config.toml

sed -E -i \ "s/seeds = \" .* \" /seeds = \" {SEEDS} \" /"

\ HOME /.band/config/config.toml

sed -E -i \ "s/persistent_peers = \" .* \" /persistent_peers = \" {PERSISTENT_PEERS} \" /"

\ HOME /.band/config/config.toml

Add minimum gas price

sed -E -i \ "s/^minimum-gas-prices =./minimum-gas-prices = \" 0.0025uband \" /"

\ HOME /.band/config/app.toml

Step 2: Setup Cosmovisor

Pre-Requisite Readings * Cosmoviser Cosmovisor is a small process manager for Cosmos SDK application binaries that monitors the governance module viastdout for incoming chain upgrade proposals

Step 2.1: Setup environment variables

Add required environment variables for Cosmovisor into your profile

cd ~ echo

"export DAEMON_NAME=bandd"

~/.profile echo

"export DAEMON_HOME= HOME /.band"

~/.profile source ~/.profile

Step 2.2: Install and provide binaries

Install Cosmovisor and provide bandd binary to Cosmovisor

Install Cosmovisor

go install github.com/cosmos/cosmos-sdk/cosmovisor/cmd/cosmovisor@v1.0.0

Setup folder and provide bandd binary for Cosmovisor Genesis

mkdir -p HOME /.band/cosmovisor/genesis/bin mkdir -p HOME /.band/cosmovisor/upgrades cp

HOME /go/bin/bandd HOME /.band/cosmovisor/genesis/bin

Setup folder and provide bandd binary for Cosmovisor Upgrades

mkdir -p HOME /.band/cosmovisor/upgrades/v2_5/bin cp

HOME /go/bin/bandd DAEMON_HOME /cosmovisor/upgrades/v2_5/bin

Step 3: Sync with the network

There are three main ways to sync a node on the BandChain; Blocksync, State Sync, and Snapshots.

info It is highly recommended to useState Sync orSnapshots as it's faster. However, if you want to use Blocksync, please read the following section. * Block Sync * State Sync * Snapshot - HighStakes

Block Sync

Blocksync is faster than traditional consensus and syncs the chain from genesis by downloading blocks and verifying against the Merkle tree of validators. For more information seeTendermint's Fastsync Docs

When syncing via Blocksync, node operators will need to provide the binary of each upgrade version for Cosmovisor to switch when it reaches the upgrade heights.

You can see the detail of genesis and each upgrade in the table below.

Upgrade name Upgrade Height Upgrade detail Go version Bandd version Binary path genesis 0 - 1.16.7 v2.3.6 ~/.band/cosmovisor/genesis/bin v2_4 11525000link 1.19.1 v2.4.1 ~/.band/cosmovisor/upgrades/v2_4/bin v2_5 16562500 link 1.19.1 v2.5.4 ~/.band/cosmovisor/upgrades/v2_5/bin Before doing the next step, you have to build and provide each correct bandd binary version to Cosmovisor in the binary path so that Cosmovisor can automatically switch it correctly.

State Sync

State Sync is an efficient and fast way to bootstrap a new node, and it works by replaying larger chunks of application state directly rather than replaying individual blocks or consensus rounds. For more information, seeTendermint's State Sync docs .

Setup variables

Get trust height and trust hash

LATEST_HEIGHT

(curl -s https://rpc.laozi4.bandchain.org/block | jq -r .result.block.header.height) ; TRUST_HEIGHT = ((LATEST_HEIGHT - 30000)) TRUST_HASH = (curl -s "https://rpc.laozi4.bandchain.org/block?height= TRUST_HEIGHT "

| jq -r .result.block_id.hash)

show trust height and trust hash

echo

"TRUST HEIGHT: TRUST_HEIGHT " echo

"TRUST HASH: TRUST_HASH "

Setup State Sync config

Enable State Sync

sed -i \ '/[statesync]/,+34 s/enable = false/enable = true/'

\ HOME /.band/config/config.toml

Set RPC Endpoint for State Sync

sed -E -i \ "/[statesync]/,+34 s/rpc_servers = \" .* \" /rpc_servers = \" http\:\/\/rpc.laozi1.bandchain.org\:80,http\:\/\/rpc.laozi2.bandchain.org\:80,https\:\/\/rpc.laozi3.bandchain.org\:443,https\:\/\/rpc.laozi4.bandchain.org\:443 \" /"

\ HOME /.band/config/config.toml

Set Trust Height for State Sync

sed -i \ "/[statesync]/,+34 s/trust_height = .*/trust_height = {TRUST_HEIGHT} /"

\ HOME /.band/config/config.toml

Set Trust Hash for State Sync

sed -i \ "/[statesync]/,+34 s/trust_hash = \" .* \" /trust_hash = \" {TRUST_HASH} \" /"

\ HOME /.band/config/config.toml

Snapshot - HighStakes

HighStakes provides the latest chain data snapshot with a very small size every day. You can download snapshot data and start to sync the block from snapshot height without syncing from the first

block.

For more information, click on the link belowhttps://tools.highstakes.ch/snapshots/bandprotocol

Download snapshots

cd ~/.band/

wget https://tools.highstakes.ch/files/bandprotocol.tar.gz tar -xvf bandprotocol.tar.gz

Step 4: Setup daemon service

We do recommend running the Bandchain node as a daemon, which can be set up usingsystemctl .

Step 4.1: Create BandChain service

Run the following command to create a new daemon forcosmovisor that runs bandd (This script work on non-root user).

Write bandd service file to /etc/systemd/system/bandd.service

export

USERNAME

(whoami) sudo -E bash -c 'cat << EOF > /etc/systemd/system/bandd.service [Unit] Description=BandChain Node Daemon After=network-online.target

[Service] Environment="DAEMON_NAME=bandd" Environment="DAEMON_HOME={HOME}/.band" Environment="DAEMON_RESTART_AFTER_UPGRADE=true" Environment="DAEMON_ALLOW_DOWNLOAD_BINARIES=false" Environment="UNSAFE_SKIP_BACKUP=true" User=USERNAME ExecStart={HOME}/go/bin/cosmovisor start Restart=always RestartSec=3 LimitNOFILE=4096

[Install] WantedBy=multi-user.target EOF'

Step 4.2: Register and start bandd service

In this step, we will register and start bandd service

Register bandd to systemctl

sudo systemctl enable bandd

Start bandd daemon

sudo systemctl start bandd Oncebandd service has been started, logs can be queried by runningjournalctl -u bandd.service -f command. You will see your node beginning to sync.

Step 5: Setup Yoda

Pre-requisite Readings * Yoda Yoda is a program used by BandChain's validator nodes to help automatically query data from data providers by executing data source script, then submitting the result to fulfill the request.

Since a subset of validators who are selected for a data request must send the data they received as a transaction of MsgReportData to BandChain.

Yoda requires indexer to run properly. Please make sure if your node has set indexer in config.toml file as "kv"

Step 5.1: Installation

Before setting up Yoda, the Lambda function executor need to be set up to execute data sources. If this step has not been done yet, please follow the instructions on the following pages (select either one of these methods):

- AWS Lambda Function
- Google Cloud Function

To check Yoda version, use the following command.

yoda version

v2.5.4

Step 5.2: Set the Yoda configurations

Use the command below to config your Yoda, replacingVARIABLES with their actual values.

rm -rf ~/.yoda

clear old config if exist

yoda config chain-id CHAIN_ID yoda config node http://localhost:26657 yoda config broadcast-timeout "5m" yoda config rpc-poll-interval "1s" yoda config max-try 5 yoda config validator (bandd keys show WALLET_NAME -a --bech val) Then, add multiple reporter accounts to allow Yoda to submit transactions concurrently.

yoda keys add REPORTER_1 yoda keys add REPORTER_2 yoda keys add REPORTER_3 yoda keys add REPORTER_4 yoda keys add REPORTER_5 Lastly, configure the Lambda Executor endpoint to helps running data source scripts and return results to Yoda. More details about the executor can be found in thissection .

export

EXECUTOR_URL

< YOUR_EXECUTOR_URL

yoda config executor "rest: {EXECUTOR_URL} ?timeout=10s"

Step 5.3: Start Yoda

To start Yoda, it's also recommended to usesystemctl .

Write yoda service to /etc/systemd/system/yoda.service

export

USERNAME

(whoami) sudo -E bash -c 'cat << EOF > /etc/systemd/system/yoda.service [Unit] Description=Yoda Daemon After=network-online.target

[Service] User=USERNAME ExecStart=/homeUSERNAME/go/bin/yoda run Restart=always RestartSec=3 LimitNOFILE=4096

[Install] WantedBy=multi-user.target EOF' The first time running Yoda, you will need to register and startyoda services by running the following commands.

Register yoda to systemctl

sudo systemctl enable yoda

Start yoda daemon

sudo systemctl start yoda Afteryoda service has been started, logs can be queried by runningjournalctl -u yoda.service -f command. The log should be similar to the following log example below. Once verified, you can stop tailing the log by typingControl-C .

.. . systemd [.. .] : Started Yoda Daemon. .. . yoda [.. .] : I [.. .] ⋆ Creating HTTP client with node URI: tcp://localhost:26657 .. . yoda [.. .] : I [.. .] Starting WebSocket subscriber .. . yoda [.. .] : I [.. .] Subscribing to events with query: tm.event =

'Tx' .. .

Step 5.4: Wait for the latest blocks to be synced

It is imperative to exercise caution and allow adequate time for the newly started BandChain node to synchronize its blocks until it has reached the latest block. The latest block can be verified onCosmoScan . Previous Getting Started Next Become a Validator