Validator Guide: Starting a Validator

Configure Solana CLI

The solana cli includesget andset configuration commands to automatically set the--url argument for cli commands. For example:

solana config set --url http://api.devnet.solana.com While this section demonstrates how to connect to the Devnet cluster, the steps are similar for the otherSolana Clusters .

Confirm The Cluster Is Reachable

Before attaching a validator node, sanity check that the cluster is accessible to your machine by fetching the transaction count:

solana transaction-count View themetrics dashboard for more detail on cluster activity.

Enabling CUDA

If your machine has a GPU with CUDA installed(Linux-only currently), include the--cuda argument tosolana-validator .

When your validator is started look for the following log message to indicate that CUDA is enabled:"[solana::validator] CUDA is enabled"

System Tuning

Linux

If you would prefer to manage system settings on your own, you may do so with the following commands.

Optimize sysctl knobs

sudo

bash -c "cat >/etc/sysctl.d/21-solana-validator.conf <<EOF

Increase UDP buffer sizes

net.core.rmem_default = 134217728 net.core.rmem_max = 134217728 net.core.wmem_default = 134217728 net.core.wmem_max = 134217728

Increase memory mapped files limit

vm.max_map_count = 1000000

Increase number of allowed open file descriptors

fs.nr_open = 1000000 EOF" sudo sysctl -p /etc/sysctl.d/21-solana-validator.conf

Increase systemd and session file limits

Add

LimitNOFILE=1000000 to the[Service] section of your systemd service file, if you use one, otherwise add

DefaultLimitNOFILE=1000000 to the[Manager] section of/etc/systemd/system.conf .

sudo systemctl daemon-reload sudo

bash -c "cat >/etc/security/limits.d/90-solana-nofiles.conf <<EOF

Increase process file descriptor count limit

-
  - nofile 1000000 EOF"

Close all open sessions (log out then, in again)

System Clock

Large system clock drift can prevent a node from properly participating in Solana's gossip protocol . Ensure that your system clock is accurate. To check the current system clock, use:

timedatectl Operators commonly use an ntp server to maintain an accurate system clock.

Generate identity

Create an identity keypair for your validator by running:

solana-keygen new -o ~/validator-keypair.json The identity public key can now be viewed by running:

solana-keygen pubkey ~/validator-keypair.json Note: The "validator-keypair.json” file is also your(ed25519)private key.

Paper Wallet identity

You can create a paper wallet for your identity file instead of writing the keypair file to disk with:

solana-keygen new --no-outfile The corresponding identity public key can now be viewed by running:

solana-keygen pubkey ASK and then entering your seed phrase.

SeePaper Wallet Usage for more info.

Vanity Keypair

You can generate a custom vanity keypair using solana-keygen. For instance:

solana-keygen grind --starts-with e1v1s:1 You may request that the generated vanity keypair be expressed as a seed phrase which allows recovery of the keypair from the seed phrase and an optionally supplied passphrase (note that this is significantly slower than grinding without a mnemonic):

solana-keygen grind --use-mnemonic --starts-with e1v1s:1 Depending on the string requested, it may take days to find a match...

Your validator identity keypair uniquely identifies your validator within the network.It is crucial to back-up this information.

If you don’t back up this information, you WILL NOT BE ABLE TO RECOVER YOUR VALIDATOR if you lose access to it. If this happens, YOU WILL LOSE YOUR ALLOCATION OF SOL TOO.

To back-up your validator identify keypair,back-up your "validator-keypair.json” file or your seed phrase to a secure location.

More Solana CLI Configuration

Now that you have a keypair, set the solana configuration to use your validator keypair for all following commands:

solana config set --keypair ~/validator-keypair.json You should see the following output:

Config File: /home/solana/.config/solana/cli/config.yml RPC URL: http://api.devnet.solana.com WebSocket URL: ws://api.devnet.solana.com/ (computed) Keypair Path: /home/solana/validator-keypair.json Commitment: confirmed

Airdrop & Check Validator Balance

Airdrop yourself some SOL to get started:

solana airdrop 1 Note that airdrops are only available on Devnet and Testnet. Both are limited to 1 SOL per request.

To view your current balance:

solana balance Or to see in finer detail:

solana balance --lamports Read more about thedifference between SOL and lamports here.

Create Authorized Withdrawer Account

If you haven't already done so, create an authorized-withdrawer keypair to be used as the ultimate authority over your validator. This keypair will have the authority to withdraw from your vote account, and will have the additional authority to change all other aspects of your vote account. Needless to say, this is a very important keypair as anyone who possesses it can make any changes to your vote account, including taking ownership of it permanently. So it is very important to keep your authorized-withdrawer keypair in a safe location. It does not need to be stored on your validator, and should not be stored anywhere from where it could be accessed by unauthorized parties. To create your authorized-withdrawer keypair:

solana-keygen new -o ~/authorized-withdrawer-keypair.json

Create Vote Account

If you haven’t already done so, create a vote-account keypair and create the vote account on the network. If you have completed this step, you should see the “vote-account-keypair.json” in your Solana runtime directory:

solana-keygen new -o ~/vote-account-keypair.json The following command can be used to create your vote account on the blockchain with all the default options:

solana create-vote-account ~/vote-account-keypair.json ~/validator-keypair.json ~/authorized-withdrawer-keypair.json Remember to move your authorized withdrawer keypair into a very secure location after running the above command.

Read more aboutcreating and managing a vote account.

Known validators

If you know and respect other validator operators, you can specify this on the command line with the--known-validator argument tosolana-validator . You can specify multiple ones by repeating the argument--known-validator --known-validator . This has two effects, one is when the validator is booting with--only-known-rpc , it will only ask that set of known nodes for downloading genesis and snapshot data. Another is that in combination with the--halt-on-known-validators-accounts-hash-mismatch option, it will monitor the merkle root hash of the entire accounts state of other known nodes on gossip and if the hashes produce any mismatch, the validator will halt the node to prevent the validator from voting or processing potentially incorrect state values. At the moment, the slot that the validator publishes the hash on is tied to the snapshot interval. For the feature to be effective, all validators in the known set should be set to the same snapshot interval value or multiples of the same.

It is highly recommended you use these options to prevent malicious snapshot state download or account state divergence.

Connect Your Validator

Connect to the cluster by running:

solana-validator \ --identity ~/validator-keypair.json \ --vote-account ~/vote-account-keypair.json \ --rpc-port 8899

\ --entrypoint entrypoint.devnet.solana.com:8001 \ --limit-ledger-size \ --log ~/solana-validator.log To force validator logging to the console add a--log - argument, otherwise the validator will automatically log to a file.

The ledger will be placed in theledger/ directory by default, use the--ledger argument to specify a different location.

Note: You can use apaper wallet seed phrase for your--identity and/or--authorized-voter keypairs. To use these, pass the respective argument assolana-validator --identity ASK ... --authorized-voter ASK ... and you will be prompted to enter your seed phrases and optional passphrase. Confirm your validator is connected to the network by opening a new terminal and running:

solana gossip If your validator is connected, its public key and IP address will appear in the list.

Controlling local network port allocation

By default the validator will dynamically select available network ports in the 8000-10000 range, and may be overridden with--dynamic-port-range . For example,solana-validator --dynamic-port-range 11000-11020 ... will restrict the validator to ports 11000-11020.

Limiting ledger size to conserve disk space

The--limit-ledger-size parameter allows you to specify how many ledgershreds your node retains on disk. If you do not include this parameter, the validator will keep all received ledger data until it runs out of disk space. Otherwise, the validator will continually purge the oldest data once to stay under the specified--limit-ledger-size value.

The default value attempts to keep the blockstore (data within the rocksdb directory) disk usage under 500 GB. More or less disk usage may be requested by adding an argument to--limit-ledger-size if desired. More information about selecting a custom limit value isavailable here .

Note that the above target of 500 GB does not account for other items that may reside in theledger directory, depending on validator configuration. These items may include (but are not limited to):

- Persistent accounts data
- Persistent accounts index
- Snapshots

Systemd Unit

Running the validator as a systemd unit is one easy way to manage running in the background.

Assuming you have a user calledsol on your machine, create the file/etc/systemd/system/sol.service with the following:

[Unit] Description=Solana Validator After=network.target StartLimitIntervalSec=0

[Service] Type=simple Restart=always RestartSec=1 User=sol LimitNOFILE=1000000 LogRateLimitIntervalSec=0 Environment="PATH=/bin:/usr/bin:/home/sol/.local/share/solana/install/active_release/bin" ExecStart=/home/sol/bin/validator.sh

[Install] WantedBy=multi-user.target Now create/home/sol/bin/validator.sh to include the desiredsolana-validator command-line. Ensure that the 'exec' command is used to start the validator process (i.e. "exec solana-validator ..."). This is important because without it, logrotate will end up killing the validator every time the logs are rotated.

Ensure that running/home/sol/bin/validator.sh manually starts the validator as expected. Don't forget to mark it executable withchmod +x /home/sol/bin/validator.sh

Start the service with:

sudo systemctl enable --now sol

Logging

Log output tuning

The messages that a validator emits to the log can be controlled by theRUST_LOG environment variable. Details can by found in thedocumentation for theenv_logger Rust crate.

Note that if logging output is reduced, this may make it difficult to debug issues encountered later. Should support be sought from the team, any changes will need to be reverted and the issue reproduced before help can be provided.

Log rotation

The validator log file, as specified by--log ~/solana-validator.log , can get very large over time and it's recommended that log rotation be configured.

The validator will re-open its log file when it receives theUSR1 signal, which is the basic primitive that enables log rotation.

If the validator is being started by a wrapper shell script, it is important to launch the process withexec (exec solana-validator ...) when using logrotate. This will prevent theUSR1 signal from being sent to the script's process instead of the validator's, which will kill them both.

Using logrotate

An example setup for thelogrotate , which assumes that the validator is running as a systemd service calledsol.service and writes a log file at /home/sol/solana-validator.log:

Setup log rotation

cat

> logrotate.sol << EOF /home/sol/solana-validator.log { rotate 7 daily missingok postrotate systemctl kill -s USR1 sol.service endscript } EOF sudo

cp logrotate.sol /etc/logrotate.d/sol systemctl restart logrotate.service As mentioned earlier, be sure that if you use logrotate, any script you create which starts the solana validator process uses "exec" to do so (example: "exec solana-validator ..."); otherwise, when logrotate sends its signal to the validator, the enclosing script will die and take the validator process with it.

Using a ramdisk with spill-over into swap for the accounts database to reduce SSD wear

If your machine has plenty of RAM, a tmpfs ramdisk (tmpfs) may be used to hold the accounts database

When using tmpfs it's essential to also configure swap on your machine as well to avoid running out of tmpfs space periodically.

A 300GB tmpfs partition is recommended, with an accompanying 250GB swap partition.

Example configuration:

1. sudo mkdir /mnt/solana-accounts
2. Add a 300GB tmpfs partition by adding a new line containingtmpfs /mnt/solana-accounts tmpfs rw,size=300G,user=sol 0 0
3. to/etc/fstab
4. (assuming your validator is running under the user "sol").CAREFUL: If you
5. incorrectly edit /etc/fstab your machine may no longer boot
6. Create at least 250GB of swap space
7. Choose a device to use in place ofSWAPDEV
8. for the remainder of these instructions.
9. Ideally select a free disk partition of 250GB or greater on a fast disk. If one is not
10. available, create a swap file withsudo dd if=/dev/zero of=/swapfile bs=1MiB count=250KiB
11. ,
12. set its permissions withsudo chmod 0600 /swapfile
13. and use/swapfile
14. asSWAPDEV
15. for
16. the remainder of these instructions
17. Format the device for usage as swap withsudo mkswap SWAPDEV
18. Add the swap file to/etc/fstab
19. with a new line containingSWAPDEV swap swap defaults 0 0
20. Enable swap withsudo swapon -a
21. and mount the tmpfs withsudo mount /mnt/solana-accounts/
22. Confirm swap is active withfree -g
23. and the tmpfs is mounted withmount

Now add the--accounts /mnt/solana-accounts argument to yoursolana-validator command-line arguments and restart the validator.

Account indexing

As the number of populated accounts on the cluster grows, account-data RPC requests that scan the entire account set -- likegetProgramAccounts andSPL-token-specific requests -- may perform poorly. If your validator needs to support any of these requests, you can use the--account-index parameter to activate one or more in-memory account indexes that significantly improve RPC performance by indexing accounts by the key field. Currently supports the following parameter values:

- program-id
- : each account indexed by its owning program; used bygetProgramAccounts
- spl-token-mint
- : each SPL token account indexed by its token Mint; used bygetTokenAccountsByDelegate
- , andgetTokenLargestAccounts
- spl-token-owner
- : each SPL token account indexed by the token-owner address; used bygetTokenAccountsByOwner
- , andgetProgramAccounts

- requests that include an spl-token-owner filter. Previous Best Practices: Validator Security Next Validator Guides: Publishing Validator Info