Setup a Solana Validator

This is a guide for getting your validator setup on the Solana testnet cluster for the first time. Testnet is a Solana cluster that is used for performance testing of the software before the software is used on mainnet. Since testnet is stress tested daily, it is a good cluster to practice validator operations.

Once you have a working validator on testnet, you will want to learn aboutoperational best practices in the next section. Although the guide is specific to testnet, it can be adapted to mainnet or devnet as well.

Refer to theAvailable Clusters section of the documentation to see example commands for each cluster. Now let's get started.

Open The Terminal Program

To start this guide, you will be running commands on your trusted computer, not on the remote machine that you plan to use for validator operations. First, locate the terminal program on yourtrusted computer .

- on Mac, you can search for the wordterminal
- in spotlight.
- on Ubuntu, you can typeCTRL + Alt + T
- .
- on Windows, you will have to open the command prompt as an Administrator.

Install The Solana CLI Locally

To create your validator vote account, you need to install theSolana command line interface on your local computer.

You can either useSolana's Install Tool section from the within these docs to install the CLI, or alternatively, you can alsobuild from source .

Building from source is a great option for those that want a more secure and potentially more performant executable. Once the Solana CLI is installed, you can return to this document once you are able to run the following command and get an answer on your terminal:

solana --version You should see an output that looks similar to this (note your version number may be higher):

solana-cli 1.14.17 (src:b29a37cf; feat:3488713414) Once you have successfully installed the cli, the next step is to change your config so that it is making requests to thetestnet cluster:

solana config set --url https://api.testnet.solana.com To verify that your config has change run:

solana config get You should see a line that says:RPC URL: https://api.testnet.solana.com

Create Keys

On your local computer, create the 3 keypairs that you will need to run your validator (docs for reference):

NOTE Some operators choose to make vanity keypairs for their identity and vote account using thegrind sub command (docs for reference). solana-keygen new -o validator-keypair.json solana-keygen new -o vote-account-keypair.json solana-keygen new -o authorized-withdrawer-keypair.json IMPORTANT theauthorized-withdrawer-keypair.json should be considered very sensitive information. Many operators choose to use a multisig, hardware wallet, or paper wallet for the authorized withdrawer keypair. A keypair is created on disk in this example for simplicity. Additionally, the withdrawer keypair should always be stored safely. The authorized withdrawer keypair shouldnever be stored on the remote machine that the validator software runs on. For more information, seevalidator security best practices

Create a Vote Account

Before you can create your vote account, you need to configure the Solana command line tool a bit more.

The below command sets the default keypair that the Solana CLI uses to thevalidator-keypair.json file that you just created in the terminal:

solana config set --keypair ./validator-keypair.json Now verify your account balance of0 :

solana balance Next, you need to deposit some SOL into that keypair account in order create a transaction (in this case, making your vote account):

solana airdrop 1 NOTE Theairdrop sub command does not work on mainnet, so you will have to acquire SOL and transfer it into this keypair's account if you are setting up a mainnet validator. Now, use the Solana cluster to create a vote account.

As a reminder, all commands mentioned so farshould be done on your trusted computer andNOT on a server where you intend to run your validator. It is especially important that the following command is done on atrusted computer :

solana create-vote-account -ut \ --fee-payer ./validator-keypair.json \ ./vote-account-keypair.json \ ./validator-keypair.json \ ./authorized-withdrawer-keypair.json Note-ut tells the cli command that we would like to use the testnet cluster.--fee-payer specifies the keypair that will be used to pay the transaction fees. Both flags are not necessary if you configured the solana cli properly above but they are useful to ensure you're using the intended cluster and keypair.

Save the Withdrawer Keypair Securely

Make sure yourauthorized-withdrawer-keypair.json is stored in a safe place. If you have chosen to create a keypair on disk, you should first backup the keypair and then delete it from your local machine.

IMPORTANT : If you lose your withdrawer key pair, you will lose control of your vote account. You will not be able to withdraw tokens from the vote account or update the withdrawer. Make sure to store theauthorized-withdrawer-keypair.json securely before you move on.

SSH To Your Validator

Connect to your remote server. This is specific to your server but will look something like this:

ssh user@ You will have to check with your server provider to get the correct user account and hostname that you will ssh into.

Update Your Ubuntu Packages

Make sure you have the latest and greatest package versions on your server

sudo apt update sudo apt upgrade

Sol User

Create a new Ubuntu user, namedsol , for running the validator:

sudo adduser sol It is a best practice to always run your validator as a non-root user, like thesol user we just created.

Hard Drive Setup

On your Ubuntu computer make sure that you have at least2TB of disk space mounted. You can check disk space using thedf command:

df -h If you have a drive that is not mounted/formatted, you will have to set up the partition and mount the drive. To see the hard disk devices that you have available, use the list block devices command:

lsblk -f You may see some devices in the list that have a name but do not have a UUID. Any device without a UUID is unformatted.

Drive Formatting: Ledger

Assuming you have an nvme drive that is not formatted, you will have to format the drive and then mount it.

For example, if your computer has a device located at/dev/nvme0n1 , then you can format the drive with the command:

sudo mkfs -t ext4 /dev/nvme0n1 For your computer, the device name and location may be different.

Next, check that you now have a UUID for that device:

lsblk -f In the fourth column, next to your device name, you should see a string of letters and numbers that look like this:6abd1aa5-8422-4b18-8058-11f821fd3967 . That is the UUID for the device.

Mounting Your Drive: Ledger

So far we have created a formatted drive, but you do not have access to it until you mount it. Make a directory for mounting your drive:

sudo mkdir -p /mnt/ledger Next, change the ownership of the directory to yoursol user:

sudo chown -R sol:sol /mnt/ledger Now you can mount the drive:

sudo mount /dev/nvme0n1 /mnt/ledger

Formatting And Mounting Drive: AccountsDB

You will also want to mount the accounts db on a separate hard drive. The process will be similar to the ledger example above.

Assuming you have device at/dev/nvme1n1 , format the device and verify it exists:

sudo mkfs -t ext4 /dev/nvme1n1 Then verify the UUID for the device exists:

lsblk -f Create a directory for mounting:

sudo mkdir -p /mnt/accounts Change the ownership of that directory:

sudo chown -R sol:sol /mnt/accounts And lastly, mount the drive:

sudo mount /dev/nvme1n1 /mnt/accounts

System Tuning

Linux

Your system will need to be tuned in order to run properly. Your validator may not start without the settings below.

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)

Copy Key Pairs

On your personal computer, not on the validator, securely copy yourvalidator-keypair.json file and yourvote-account-keypair.json file to the validator server:

scp validator-keypair.json sol@: scp vote-account-keypair.json sol@: Note : Thevote-account-keypair.json does not have any function other than identifying the vote account to potential delegators. Only the public key of the vote account is important once the account is created.

Switch to the sol User

On the validator server, switch to thesol user:

su - sol

Install The Solana CLI on Remote Machine

Your remote machine will need the Solana cli installed to run the validator software. Refer again toSolana's Install Tool orbuild from source . It is best for operators to build from source rather than using the pre built binaries.

Create A Validator Startup Script

In your sol home directory (e.g./home/sol/), create a folder calledbin . Inside that folder create a file calledvalidator.sh and make it executable:

mkdir -p /home/sol/bin touch /home/sol/bin/validator.sh chmod +x /home/sol/bin/validator.sh Next, open thevalidator.sh file for editing:

nano /home/sol/bin/validator.sh Copy and paste the following contents intovalidator.sh then save the file:

exec solana-validator \ --identity validator-keypair.json \ --vote-account vote-account-keypair.json \ --known-validator 5D1fNXzvv5NjV1ysLjirC4WY92RNsVH18vjmcszZd8on \ --known-validator 7XSY3MrYnK8vq693Rju17bbPkCN3Z7KvvfvJx4kdrsSY \ --known-validator Ft5fbkqNa76vnsjYNwjDZUXoTWpP7VYm3mtsaQckQADN \ --known-validator 9QxCLckBiJc783jnMvXZubK4wH86Eqqvashtrwvcsgkv \ --only-known-rpc \ --log /home/sol/solana-validator.log \ --ledger /mnt/ledger \ --rpc-port 8899 \ --dynamic-port-range 8000-8020 \ --entrypoint entrypoint.testnet.solana.com:8001 \ --entrypoint entrypoint2.testnet.solana.com:8001 \ --entrypoint entrypoint3.testnet.solana.com:8001 \ --expected-genesis-hash 4uhcVJyU9pJkvQyS88uRDiswHXSCkY3zQawwpjk2NsNY \ --wal-recovery-mode skip_any_corrupted_record \ --limit-ledger-size Refer tosolana-validator --help for more information on what each flag is doing in this script. Also refer to the section onbest practices for operating a validator .

Verifying Your Validator Is Working

Test that yourvalidator.sh file is running properly by executing thevalidator.sh script:

/home/sol/bin/validator.sh The script should execute thesolana-validator process. In a new terminal window, shh into your server, then verify that the process is running:

ps aux | grep solana-validator You should see a line in the output that includessolana-validator with all the flags that were added to yourvalidator.sh script.

Next, we need to look at the logs to make sure everything is operating properly.

Tailing The Logs

As a spot check, you will want to make sure your validator is producing reasonable log output (warning , there will be a lot of log output).

In a new terminal window, ssh into your validator machine, switch users to thesol user andtail the logs:

su - sol tail -f solana-validator.log Thetail command will continue to display the output of a file as the file changes. You should see a continuous stream of log output as your validator runs. Keep an eye out for any lines that say_ERROR_ .

Assuming you do not see any error messages, exit out of the command.

Gossip Protocol

Gossip is a protocol used in the Solana clusters to communicate between validator nodes. For more information on gossip, seeGossip Service . To verify that your validator is running properly, make sure that the validator has registered itself with the gossip network.

In a new terminal window, connect to your server via ssh. Identify your validator's pubkey:

solana-keygen pubkey ~/validator-keypair.json The commandsolana gossip lists all validators that have registered with the protocol. To check that the newly setup validator is in gossip, we willgrep for our pubkey in the output:

solana gossip | grep After running the command, you should see a single line that looks like this:

139.178.68.207 | 5D1fNXzvv5NjV1ysLjirC4WY92RNsVH18vjmcszZd8on | 8001 | 8004 | 139.178.68.207:80 | 1.14.17 | 3488713414 If you do not see any output after grep-ing the output of gossip, your validator may be having startup problems. If that is the case, start debugging by looking through the validator log output.

Solana Validators

After you have verified that your validator is in gossip, you can verify that your validator has joined the network using thesolana validators command. The command lists all validators in the network, but like before, we cangrep the output for the validator we care about:

solana validators | grep You should see a line of output that looks like this:

5D1fNXzvv5NjV1ysLjirC4WY92RNsVH18vjmcszZd8on FX6NNbS5GHc2kuzgTZetup6GZX6ReaWyki8Z8jC7rbNG 100% 197434166 (0) 197434133 (0) 2.11% 323614 1.14.17 2450110.588302720 SOL (1.74%)

Solana Catchup

Thesolana catchup command is a useful tool for seeing how quickly your validator is processing blocks. The Solana network has the capability to produce many transactions per second. Since your validator is new to the network, it has to ask another validator (listed as a--known-validator in your startup script) for a recent snapshot of the ledger. By the time you receive the snapshot, you may already be behind the network. Many transactions may have been processed and finalized in that time. In order for your validator to participate in consensus, it mustcatchup to the rest of the network by asking for the more recent transactions that it does not have.

Thesolana catchup command is a tool that tells you how far behind the network your validator is and how quickly you are catching up:

solana catchup If you see a message about trying to connect, your validator may not be part of the network yet. Make sure to check the logs and double checksolana gossip andsolana validators to make sure your validator is running properly.

Once you are happy that the validator can start up without errors, the next step is to create a system service to run thevalidator.sh file automatically. Stop the currently running validator by pressingCTRL+C in the window wherevalidator.sh is running.

Create a System Service

Follow these instructions forrunning the validator as a system service

Make sure to implement log rotate as well. Once you have the system service configured, start your validator using the newly configured service:

sudo systemctl enable --now sol Now verify that the validator is running properly by tailing the logs and using the commands mentioned earlier to check gossip and Solana validators:

tail -f /home/sol/solana-validator*.log

Monitoring

solana-watchtower is a command you can run on a separate machine to monitor your server. You can read more about handlingautomatic restarts and monitoring using Solana Watchtower here in the docs.

Common issues

Out of disk space

Make sure your ledger is on drive with at least2TB of space.

Validator not catching up

This could be a networking/hardware issue, or you may need to get the latest snapshot from another validator node.
Previous Validator Initiatives Next Setup an RPC Node