## Update the Server

sudo apt update && sudo apt upgrade

sudo apt dist-upgrade && sudo apt autoremove

## Secure the Server

Find your available port.

sudo ss -tulpn | grep ':<yourSSHportnumber>'

Update the firewall to allow inbound traffic on <yourSSHportnumber>. SSH requires TCP.

sudo ufw allow <yourSSHportnumber>/tcp

Next change the default SSH port.

sudo nano /etc/ssh/sshd\_config

Find the line with # Port 22 or Port 22 and change it to Port <yourSSHportnumber>. Remove the # if it was present.

Restart the SSH service.

sudo systemctl restart ssh

Next time you log in via SSH use <yourSSHportnumber> for the port.

Optional: If you were already using UFW with port 22/TCP allowed then update the firewall to deny inbound traffic on that port. Only do this after you log in using the new SSH port.

sudo ufw deny 22/tcp

Create and run this script.

#!/bin/bash

#Allow Go Ethereum

sudo ufw allow 30303

#Allow Prysm

sudo ufw allow 13000/tcp

sudo ufw allow 12000/udp

#Allow Grafana

sudo ufw allow 3000/tcp

#Allow Prometheus

sudo ufw allow 9090/tcp

#Enable Firewall

sudo ufw enable

sudo ufw status numbered

## Mount USB Drive

Check filesystem and name.

lsblk -f

Make mount point

sudo mkdir /media/usb

Mount drive

sudo mount /dev/sda1 /media/usb

Permanently mount drive

Get UUID

sudo blkid

Edit fstab

sudo nano /etc/fstab

At the bottom of that file, add an entry that contains the UUID:

UUID=75963928-cb91-4f6d-af0f-8e6412813383 /media/usb ext4 auto uid=1000,gid=1000,umask=022,nosuid,nodev,nofail,x-gvfs-show 0 0

Details:

UUID - is the UUID of the drive. You don't have to use the UUID here. You could just use /dev/sdj, but it's always safer to use the UUID as that will never change (whereas the device name could).

/data - is the mount point for the device.

auto - automatically mounts the partition at boot

nosuid - specifies that the filesystem cannot contain set userid files. This prevents root escalation and other security issues.

nodev - specifies that the filesystem cannot contain special devices (to prevent access to random device hardware).

nofail - removes the errorcheck.

x-gvfs-show - show the mount option in the file manager. If this is on a GUI-less server, this option won't be necessary.

0 - determines which filesystems need to be dumped (0 is the default).

0 - determine the order in which filesystem checks are done at boot time (0 is the default).

## Install Prysm

Create a working directory and enter it:

sudo mkdir prysm && cd prysm

Fetch the prysm.sh script from Github and make it executable:

curl https://raw.githubusercontent.com/prysmaticlabs/prysm/master/prysm.sh --output prysm.sh && sudo chmod +x prysm.sh

## Run your beacon node

Create Directory

sudo mkdir beaconchain

Create and Configure the Service

Create a systemd service file to store the service config.

sudo nano /etc/systemd/system/prysm-beaconchain.service

[Unit]

Description=Prysm Beaconchain

Wants=network-online.target

After=network-online.target

[Service]

Type=simple

User=ubuntu

Group=ubuntu

Restart=always

RestartSec=5

Environment="ClientIP=$(curl -s v4.ident.me)"

ExecStart=/bin/bash -c '/media/usb/prysm/prysm.sh beacon-chain --pyrmont --datadir=/media/usb/prysm/beaconchain --http-web3provider=https://eth-goerli.alchemyapi.io/v2/DyY0KSEap\_-bfoByvSqF-v2vbUkp3Gvc --accept-terms-of-use'

[Install]

WantedBy=multi-user.target

Exit and save.

The --pyrmont flag is required to indicate we are running against the testnet.

The --p2p-host-ip flag is recommended to improve peer networking. We use an Environment variable Environment="ClientIP=$(curl -s v4.ident.me)" to get the client IP address because ExecStart doesn’t allow the call in-line. Using --p2p-host-ip=${ClientIP} is the work-around.

The --http-web3provider flag defines the endpoint of the Eth1 node. If you installed one locally the value is http://127.0.0.1:8545. If you’re using a third party use the external endpoint address (e.g. Infura or Prysmatic’s Eth1 node: https://goerli.prylabs.net).

The --accept-terms-of-use flag is required in order to be able to run the binary as a service. Using this flag indicates acceptance of the Prysm terms of use.

Reload systemd to reflect the changes.

sudo systemctl daemon-reload

Start the service and check to make sure it’s running correctly.

sudo systemctl start prysm-beaconchain

sudo systemctl status prysm-beaconchain

Enable the service to automatically start on reboot.

sudo systemctl enable prysm-beaconchain

The beacon-chain will begin to sync. It may take several hours for the node to fully sync. You can check the progress by running the journal command. Press Ctrl+C to quit.

sudo journalctl -fu prysm-beaconchain.service

## Create the Validator Wallet

First create a directory to store the validator wallet and give the current user permissions to access it. Change <yourusername> to your logged in username. This is a temporary assignment while we create the wallet. We will reassign permissions to the validator service later.

sudo mkdir -p /var/lib/prysm/validator

sudo chown -R ubuntu:ubuntu /var/lib/prysm/validator

…

Create a directory and copy your validator keys.

sudo mkdir -p /var/lib/prysm/validator/validator\_keys

sudo chown -R ubuntu:ubuntu /var/lib/prysm/validator/validator\_keys

Mount USB with Keys

sudo mkdir /media/usb1

sudo mount /dev/sdb1 /media/usb1

Copy Keys to /var/lib/prysm/validator/validator\_keys

sudo cp keystore-m\_12381\_3600\_0\_0\_0-1605793834.json /var/lib/prysm/validator/validator\_keys

sudo cp keystore-m\_12381\_3600\_1\_0\_0-1605793834.json /var/lib/prysm/validator/validator\_keys

Import accounts\keys

./prysm.sh validator accounts import --keys-dir/var/lib/prysm/validator/validator\_keys --accept-terms-of-use –pyrmont

You will be asked to create a validator wallet password and to enter the password when you generated the keys and your mnemonic.

Verify

./prysm.sh validator accounts list --pyrmont --wallet-dir /var/lib/prysm/validator --accept-terms-of-use

Create a file to store the wallet password so the validator can access the wallet without having to manually supply the password. The file will be named password.txt.

touch /var/lib/prysm/validator/password.txt

sudo nano /var/lib/prysm/validator/password.txt

Protect the document by removing access for g(roup)o(ther).

sudo chmod go-rw /var/lib/prysm/validator/password.txt

## Configure the Validator

Setup Accounts and Directories

We will run the validator as a service so if the system restarts the process will automatically start back up again.

Create an account for the service to run under. This type of account can’t log into the server.

sudo useradd --no-create-home --shell /bin/false prysm-validator

We created the data directory for the validator in the previous step: /var/lib/prysm/validator. Now set directory permissions so the prysm-validator account can modify the validator account data directory.

sudo chown -R ubuntu:ubuntu /var/lib/prysm/validator

Create a systemd service file to store the service config.

sudo nano /etc/systemd/system/prysm-validator.service

[Unit]

Description=Validator

Wants=network-online.target

After=network-online.target

[Service]

Type=simple

User=ubuntu

Group=ubuntu

Restart=always

RestartSec=5

ExecStart=/bin/bash -c '/media/usb/prysm/prysm.sh validator --pyrmont --datadir /media/usb/prysm/ --wallet-dir /var/lib/prysm/validator --wallet-password-file /var/lib/prysm/validator/password.txt --accept-terms-of-use'

[Install]

WantedBy=multi-user.target

Reload systemd to reflect the changes.

sudo systemctl daemon-reload

Start the service and check to make sure it’s running correctly.

sudo systemctl start prysm-validator

sudo systemctl status prysm-validator

Enable the validator service to automatically start on reboot.

sudo systemctl enable prysm-validator

You can check the progress by running the journal command. Press Ctrl+C to quit.

sudo journalctl -fu prysm-validator.service