Validator Guide: Setup Node Failover

A simple two machine instance failover method is described here, which allows you to:

- upgrade your validator software with virtually no down time, and
- failover to the secondary instance when your monitoring detects a problem with
- the primary instance
- without any safety issues that would otherwise be associated with running two
- instances of your validator.

You will need two validator-class machines for your primary and secondary validator. A third machine for running an etcd cluster, which is used to store the tower voting record for your validator.

Setup

etcd cluster setup

There is ample documentation regarding etcd setup and configuration at https://etcd.io/ , please generally familiarize yourself with etcd before continuing.

It's recommended that etcd be installed on a separate machine from your primary and secondary validator machines. This machine must be highly available, and depending on your needs you may wish to configure etcd with more than just one node.

First installetcd as desired for your machine. Then TLS certificates must be created for authentication between the etcd cluster and your validator. Here is one way to do this:

WithGolang installed, rungo install github.com/cloudflare/cfssl/cmd/cfssl@latest . Thecfssl program should now be available at~/go/bin/cfssl . Ensure~/go/bin is in your PATH by runningPATH=PATH:~/go/bin/ .

Now create a certificate directory and configuration file:

mkdir -p certs/ echo '{"CN":"etcd","hosts":["localhost", "127.0.0.1"],"key":{"algo":"rsa","size":2048}}' > certs/config.json then create certificates for the etcd server and the validator:

cfssl gencert -initca certs/config.json | cfssljson -bare certs/etcd-ca cfssl gencert -ca certs/etcd-ca.pem -ca-key certs/etcd-ca-key.pem certs/config.json | cfssljson -bare certs/validator cfssl gencert -ca certs/etcd-ca.pem -ca-key certs/etcd-ca-key.pem certs/config.json | cfssljson -bare certs/etcd Copy these files to your primary and secondary validator machines:

- certs/validator-key.pem
- certs/validator.pem
- certs/etcd-ca.pem

and these files to the machine running the etcd server:

- certs/etcd.pem
- certs/etcd-key.pem
- certs/etcd-ca.pem

With this configuration, both the validator and etdc will share the same TLS certificate authority and will each authenticate the other with it.

Startetcd with the following arguments:

etcd --auto-compaction-retention 2 --auto-compaction-mode revision \ --cert-file = certs/etcd.pem --key-file = certs/etcd-key.pem \ --client-cert-auth \ --trusted-ca-file = certs/etcd-ca.pem \ --listen-client-urls = https://127.0.0.1:2379 \ --advertise-client-urls = https://127.0.0.1:2379 and usecurl to confirm the etcd TLS certificates are properly configured:

curl --cacert certs/etcd-ca.pem https://127.0.0.1:2379/ --cert certs/validator.pem --key certs/validator-key.pem On success, curl will return a 404 response.

For more information on etcd TLS setup, please refer tohttps://etcd.io/docs/v3.5/op-guide/security/#example-2-client-to-server-authentication-with-https-client-certificates

Primary Validator

The following additionalsolana-validator parameters are required to enable tower storage into etcd:

solana-validator ... \ --tower-storage etcd \ --etcd-cacert-file certs/etcd-ca.pem \ --etcd-cert-file certs/validator.pem \ --etcd-

key-file certs/validator-key.pem \ --etcd-endpoint 127.0.0.1:2379 # <-- replace 127.0.0.1 with the actual IP address Note that once running your validatorwill terminate if it's not able to write its tower into etcd before submitting a vote transaction, so it's essential that your etcd endpoint remain accessible at all times.

Secondary Validator

Configure the secondary validator like the primary with the exception of the followingsolana-validator command-line argument changes:

- Generate and use a secondary validator identity:--identity secondary-validator-keypair.json
- Add--no-check-vote-account
- Add--authorized-voter validator-keypair.json
- (wherevalidator-keypair.json
- is the identity keypair for your primary validator)

Triggering a failover manually

When both validators are running normally and caught up to the cluster, a failover from primary to secondary can be triggered by running the following command on the secondary validator:

solana-validator wait-for-restart-window --identity validator-keypair.json \ && solana-validator set-identity validator-keypair.json The secondary validator will acquire a lock on the tower in etcd to ensure voting and block production safely switches over from the primary validator.

The primary validator will then terminate as soon as it detects the secondary validator using its identity.

Note: When the primary validator restarts (which may be immediate if you have configured your primary validator to do so) it will reclaim its identity from the secondary validator. This will in turn cause the secondary validator to exit. However if/when the secondary validator restarts, it will do so using the secondary validator identity and thus the restart cycle is broken.

Triggering a failover via monitoring

Monitoring of your choosing can invoke thesolana-validator set-identity validator-keypair.json command mentioned in the previous section.

It is not necessary to guarantee the primary validator has halted before failing over to the secondary, as the failover process will prevent the primary validator from voting and producing blocks even if it is in an unknown state.

Validator Software Upgrades

To perform a software upgrade using this failover method:

1. Install the new software version on your primary validator system but do not
2. restart it yet.
3. Trigger a manual failover to your secondary validator. This should cause your
4. primary validator to terminate.
5. When your primary validator restarts it will now be using the new software version.
6. Once the primary validator catches up upgrade the secondary validator at
7. your convenience. Previous Validator Guides: Vote Account ManagementNext Validator Guides: Restart a Cluster