Using Auto heal to improve cluster uptime for Nginx

Most noderunners have the common problem that stale nodes, meaning nodes that are stuck at a certain block height and will not continue keeping up the chain tip, provide horrible UX to users. To mitigate this, we can add some extra monitoring tools to dynamically add or remove nodes from the cluster.

The provided software gives you an entry level solution for this. As everything is written in Python, you can adjust this to your API setup.

To install, please do the following:

Prerequisites

Ensure you have Python installed. If not, download and install Python frompython.org . You'll also need pip, Python's package manager, to install required libraries. If you're using a Linux-based system, ensure you have NGINX installed and properly configured.

Since we edit the nginx config directly, we need to give pyhton3 sudo rights.

Be aware when you follow this tutorial.

Clone Repository

To clone the AutoHealBot repository, use the following steps:

1. Install Git
2. : Ensure Git is installed on your system. If not, install it fromgit-scm.com
3. .
4. Clone the Repository
5. : Open a terminal and run:

6.
   ```
7. Copy
8. git clone https://github.com/SecretSaturn/AutoHealBot.git
9. ```
10. Navigate to the Directory
11. : After cloning, change to the repository directory:
12. ```
13. Copy
14. cdAutoHealBot
15. ```
16.

This clones the entire repository to your local machine, allowing you to access all files and resources. To proceed with the tutorial, follow additional setup instructions provided in the repository's README or other documentation.

Install Required Libraries for Python

When installing, make sure to install this under root withsudo , otherwise the script will later not find the libraries later on.

```

Copy sudopipinstallaiohttppython-dotenvdateutil

```

Setup the enviroment file

To configure your environment variables, copy over the.env.example in the repository.

```

Copy

Path to your Nginx Config

NGINX_CONFIG_PATH='/path/to/your/nginx.conf'

Base rate limit in requests per second

BASE_RATE=8

Rate increase per healthy node

NODE_MULTIPLIER=1

Port numbers for RPC, gRPC, and LCD

RPC_PORT=26657 GRPC_PORT=9091 LCD_PORT=1317

Path to the file containing node URLs

FILE_PATH='/path/to/your/file.txt'

Time in seconds to determine if a node has fallen behind

TIME_BEFORE_FALLEN_BEHIND=18

Time interval in seconds for updating the status

UPDATE_TIME=30

```

Replace the placeholders with actual values:

- NGINX_CONFIG_PATH
- : The path to your NGINX configuration file.
- BASE_RATE
- andNODE_MULTIPLIER
- : Adjust as needed.
- RPC_PORT
- ,GRPC_PORT
- ,LCD_PORT
- : Set to your specific ports.
- FILE_PATH
- : Path to the text file with node URLs.
- TIME_BEFORE_FALLEN_BEHIND
- : Maximum allowed time before a node is considered unhealthy.
- UPDATE_TIME
- : Time between health checks.
-

Create a File with Node URLs

Create a text file with the node URLs. For example, createnodes.txt with one URL per line, make sure to include theRPC port to each node here as well:

```

Copy 192.168.0.1:26657 192.168.0.2:26657 192.168.0.2:26657

```

In the AutoHealBot script, "upstream blocks" refer to sections in the NGINX configuration that specify which backend servers handle different types of traffic. This setup divides the backend nodes into separate streams: RPC, gRPC, and LCD. The script checks the health of these nodes and updates the corresponding upstream blocks to reflect the healthy nodes for each stream. It ensures that traffic is routed to servers that are online and functional.

As a reference, the upstream blocks are defined as:
```

```

Copy upstream_blocks={ "rpc_stream":healthy_nodes_rpc, "grpc_stream":healthy_nodes_grpc,
"lcd_stream":healthy_nodes_lcd, }

```

Execute the Script

Run the script to start the asynchronous health checks and NGINX updates:

```

Copy sudopython3main.py

```

Troubleshooting

- Environment Variables Not Loaded
- : Ensure your.env
- file is in the same directory as the script or specify its path explicitly withdotenv_path
- .
- NGINX Not Reloading
- : Check if you have the necessary permissions to reload NGINX and ensure systemctl or other command-line utilities are in your PATH.
-

With this setup, the script will run asynchronously, periodically checking node health, updating the NGINX configuration, and reloading the NGINX service as needed.

Last updated18 days ago On this page * Prerequisites * Clone Repository * Install Required Libraries for Python * Setup the enviroment file * Create a File with Node URLs * Execute the Script

Was this helpful? Edit on GitHub Export as PDF