Celestia Light Node Monitoring Using Prometheus, Grafana, & Docker

Table of Contents

What is Celestia

Celestia is a modular consensus and data network, built to enable anyone to easily deploy their own blockchain with minimal overhead. Celestia provides consensus and security on-demand, enabling anyone to deploy a blockchain without the overhead of bootstrapping a new consensus network.

What is Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit, originally developed by SoundCloud. It is designed to monitor and collect metrics from various components of a system, including applications, servers, databases, and services. Prometheus uses a time-series database to store and query metrics data, and it provides a flexible querying language called PromQL for extracting and aggregating metrics.

Prometheus is commonly used:

Metrics Collection: Prometheus is built to gather metrics from different sources using a pull model. It supports a wide range of data exporters that can be integrated with applications or services to expose metrics in a format that Prometheus understands. This allows developers and system administrators to instrument their systems and collect valuable performance data.

Data Model: Prometheus uses a time-series data model, where metrics data is stored as timestamped values associated with specific labels. This model enables efficient storage and querying of metrics, allowing users to analyze and visualize data over time and across different dimensions.

Powerful Querying Language: PromQL, the query language of Prometheus, allows users to perform flexible and expressive queries on the collected metrics. It supports functions for data aggregation, filtering, mathematical operations, and more. PromQL enables users to create custom queries and generate meaningful insights from the collected metrics.

Alerting and Monitoring Rules: Prometheus provides a rule-based alerting system that allows users to define alerting conditions based on metric values. When these conditions are met, Prometheus triggers alerts that can be sent to various notification channels, such as email, chat platforms, or incident management systems. Monitoring rules can also be defined to detect and respond to specific events or conditions in the system.

What is Grafana

Grafana is an open-source data visualization and monitoring tool used to create interactive and customizable dashboards for analyzing and displaying metrics from various data sources. It supports a wide range of data sources, including time-series databases, relational databases, cloud monitoring services, log files, and more. Grafana provides a user-friendly interface that allows users to create visually appealing graphs, charts, and alerts to gain insights into their data.

Grafana is commonly used:

Data Visualization: Grafana enables users to create rich and interactive visualizations of their data. It offers a variety of visualization options, including graphs, tables, heatmaps, gauges, and more. Users can customize the appearance, style, and layout of their dashboards, allowing them to effectively present data and monitor key metrics.

Multi-Data Source Support: Grafana supports a wide range of data sources, allowing users to bring data from various systems into a single dashboard for analysis. It integrates with popular time-series databases like Prometheus, InfluxDB, and Graphite, as well as relational databases such as MySQL, PostgreSQL, and Microsoft SQL Server. It also supports cloud monitoring services like Amazon CloudWatch, Azure Monitor, and Google Cloud Monitoring.

Dashboard Templating: Grafana provides flexible dashboard templating capabilities, allowing users to create dynamic and reusable dashboards. Templating enables users to create parameterized dashboards that can be filtered, customized, and reused across different environments or for different entities, making it easier to manage and explore large amounts of data.

Alerting and Notifications: Grafana offers robust alerting functionalities, allowing users to set up alert rules based on metric thresholds or conditions. When an alert is triggered, Grafana can send notifications through various channels such as email, Slack, PagerDuty, or other messaging platforms. This helps users proactively monitor their systems and respond to critical events.

User Access Control and Sharing: Grafana provides features for user authentication, access control, and sharing of dashboards. It allows users to define roles and permissions, restricting access to sensitive data or limiting certain actions to specific user groups. Dashboards can be shared with others, either by providing a direct link or embedding them in other applications or websites.

What is Docker

Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization. It provides an environment to package applications and their dependencies into lightweight, isolated containers. Containers are self-contained, portable units that include everything needed to run an application, such as the code, runtime, system tools, libraries, and settings.

Docker is commonly used:

Application Isolation: Docker containers provide application isolation, allowing you to encapsulate your application and its dependencies. Each container runs in its own isolated environment, separate from the underlying host system and other containers. This isolation ensures that applications are portable and can run consistently across different environments, regardless of the underlying infrastructure.

Portability and Consistency: Docker enables you to create containers that are consistent across different development, testing, and production environments. Containers are self-contained and encapsulate all the required dependencies, eliminating the "it works on my machine" problem. This portability simplifies the deployment process and ensures that applications run consistently across different environments, reducing compatibility issues.

Efficient Resource Utilization: Docker utilizes operating system-level virtualization, allowing multiple containers to run on the same host system without the need for a separate virtual machine (VM) for each application. This lightweight approach results in efficient resource utilization, as containers share the host system's kernel and only require the necessary resources to run the application, making it more resource-efficient compared to traditional virtualization.

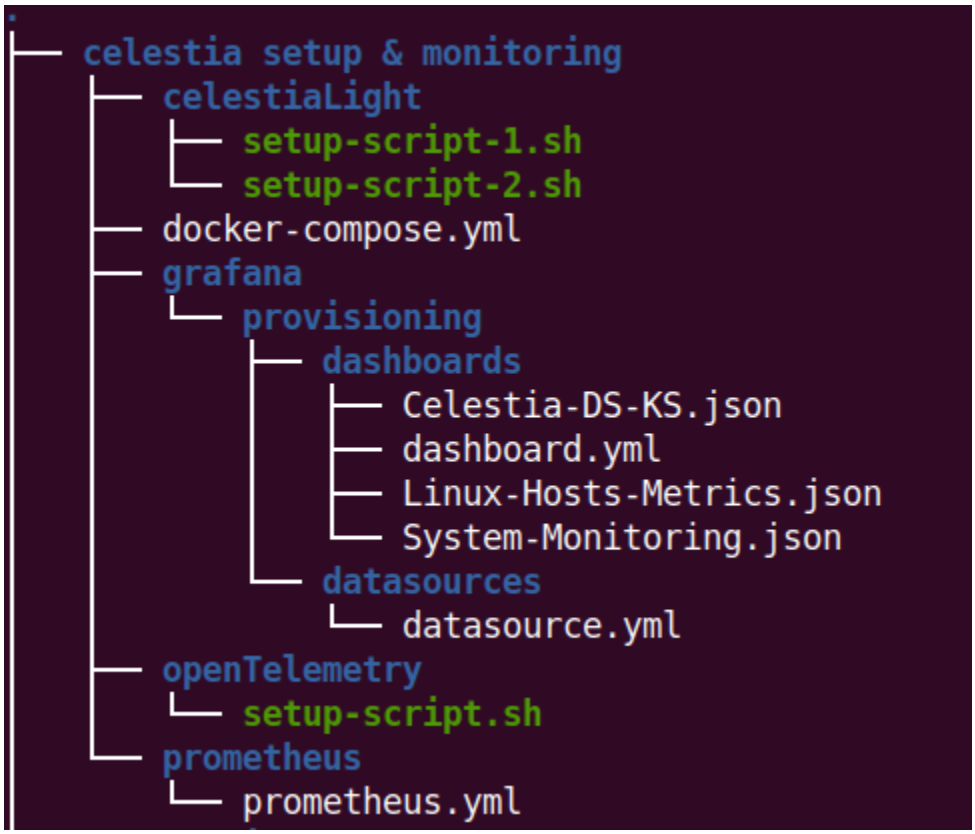
Scalability and Microservices Architecture: Docker's containerization model is well-suited for scalable and distributed architectures, such as microservices. With Docker, you can easily scale individual services by creating multiple instances of the corresponding containers. This flexibility allows applications to scale horizontally, handling increased workloads by adding more containers, and provides a modular approach to building and managing complex systems.

Versioning and Rollback: Docker provides versioning capabilities, allowing you to version control your Docker images and containers. This feature enables you to track changes, rollback to previous versions if necessary, and maintain a history of your

application's state. Versioning in Docker simplifies the release management process and provides greater control over deployments.

Directory Structure of This Project

This is the directory structure of this project:



Necessary Commands to Setup

Setup Docker in Ubuntu

Set up the repository

1. Update the apt package index and install packages to allow apt to use a repository over HTTPS

```
sudo apt-get update

sudo apt-get install ca-certificates curl gnupg
```

2. Add Docker's official GPG key:

```
sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
echo \
  "deb [arch="$(dpkg --print-architecture)"
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine in Ubuntu

1. Update the apt package index:

```
sudo apt-get update
```

2. Install Docker Engine, containerd, and Docker Compose

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

3. Install docker-compose

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-co
mpose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose; sudo
chmod +x /usr/local/bin/docker-compose
```

4. Add ubuntu user in docker group

```
sudo usermod -aG docker ubuntu

exit
```

Now login again to the VM.

Setup Celestia Light Node in Ubuntu

In the project directory go inside into the **celestiaLight** directory. Here you will find a sciprt named **setup-script.sh**. You need to run that script.

1. Go inside the celestiaLight direcotry

```
cd celestiaLight
```

2. Run script-1

```
./setup-script-1.sh
```

3. You will be asked for some prompt in the terminal.

```
- Press 2 for selecting blockspacerace
- Press 1 for default
- Press 1 for auto generated wallet
- Press y for showing the log.
```

4. Run script-2

```
./setup-script-2.sh
```

5. When the setup will be completed successfully, you will show the output the service is **active and running**.

Setup OpenTelemetry Collector in Ubuntu

In the project directory go inside into the **openTelemetry** directory. Here you will find a sciprt named **setup-script.sh**. You need to run that script.

1. Go inside the celestiaLight direcotry

```
cd ..

cd openTelemetry
```

2. Run script

```
./setup-script.sh
```

3. When the setup will be completed successfully, you will show the output the service is **active and running**.

Setup Prometheus, Grafana, and Node Exporter

1. Go to the project directory **celestia setup & monitoring**.

```
cd ..
```

2. If run ubuntu from any cloud provider then you have public IP. If you run in local then you have localhost or 127.0.0.1 IP. Provide that IP in prometheus.yml file on the placeholder.

```
nano prometheus/prometheus.yml
```

3. Now you have to run all of the necessary container in docker. Run this command.

```
docker-compose up -d
```

4. Check the running containers

```
docker ps
```

Now you will see all of the containers are running.

Visualization From Grafana

To see the dashboard you need to open your browser and paste this url

- If you run in cloud ubuntu server

```
http://your-server-public-ip:3000
```

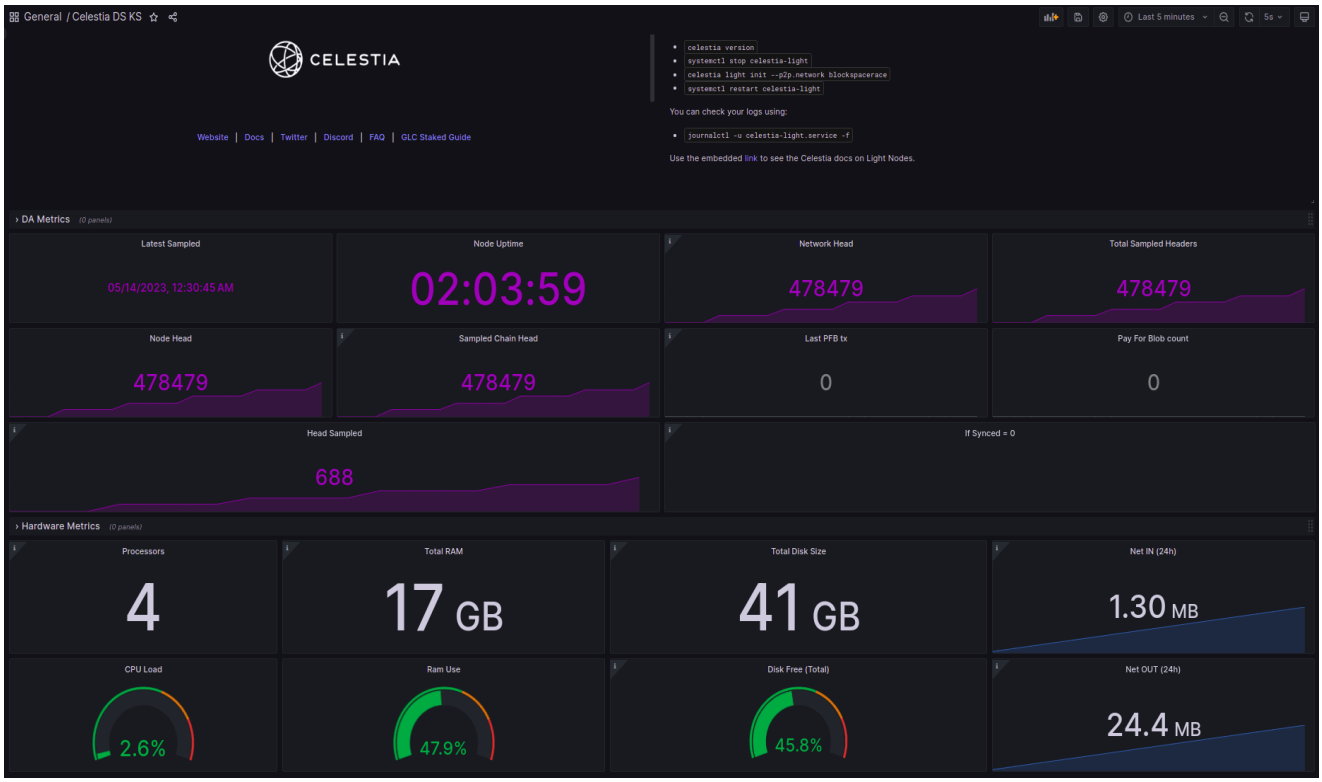
- If you run in local

```
http://localhost:3000
```

- Credentials of Grafana

```
Username: admin
Password: admin
You can change it from project directory .env file
```

Screen Capture of Celestia Dashboard



References

1. https://docs.docker.com/engine/install/ubuntu/

2. https://github.com/GLCNI/celestia-node-scripts/tree/main/multi-network#readme

3. https://github.com/GLCNI/celestia-node-scripts/blob/main/multi-network/monitoring/prometheus/README.md