Metrics

Metrics are a powerful tool for monitoring the health and performance of a system. Celestia provides support for metrics to make sure, as an operator, your system continues to remain up and running. Metrics can also provide critical insight into how Celestia is used and how it can be improved.

Setup

Celestia usesPrometheus to publish metrics. It can be enabled through theconfig.toml file.

bash

Instrumentation Configuration Options

[instrumentation]

When true, Prometheus metrics are served under /metrics on

PrometheusListenAddr.

Check out the documentation for the list of available metrics.

prometheus

=

true

Address to listen for Prometheus collector(s) connections

prometheus_listen_addr

=

":26660"

Maximum number of simultaneous connections.

If you want to accept a larger number than the default, make sure

you increase your OS limits.

0 - unlimited.

max_open_connections

=

3

Instrumentation namespace

namespace

=

"celestia"

Instrumentation Configuration Options

[instrumentation]

When true, Prometheus metrics are served under /metrics on

PrometheusListenAddr.

Check out the documentation for the list of available metrics.

prometheus

=

true

Address to listen for Prometheus collector(s) connections

prometheus_listen_addr

=

":26660"

Maximum number of simultaneous connections.

If you want to accept a larger number than the default, make sure

you increase your OS limits.

0 - unlimited.

max_open_connections

=

3

Instrumentation namespace

namespace

=

"celestia" If you restart your node, you can check to see it's working by running:

bash curl

localhost:26660/metrics curl

localhost:26660/metrics

Visualization

Now your nodes are publishing metrics, we need something to scrape it and a visualizer to create a dashboard. Commonly, Prometheus is paired with Grafana.

First, you will need to install Prometheus either from theirdownloads page or through a package manager likebrew .

Next, create a config fileHOME/.celestia-app/config/prometheus.yml and fill out some basic settings as follows:

yml global : scrape_interval : 15s

By default, scrape targets every 15 seconds.

Attach these labels to any time series or alerts when communicating

with external systems (federation, remote storage, Alertmanager).

external_labels : monitor : "codelab-monitor"

A scrape configuration containing exactly one endpoint to scrape:

Here it's Prometheus itself.

scrape_configs :

The job name is added as a label job=<job_name> to any timeseries

scraped from this config.

- job_name : "prometheus"

Override the global default and scrape targets from this job every

5 seconds.

scrape_interval : 5s

static_configs :

Point to the same endpoint that Celestia is publishing on

- targets : ["localhost:26660"] global : scrape_interval : 15s

By default, scrape targets every 15 seconds.

Attach these labels to any time series or alerts when communicating

with external systems (federation, remote storage, Alertmanager).

external_labels : monitor : "codelab-monitor"

A scrape configuration containing exactly one endpoint to scrape:

Here it's Prometheus itself.

scrape_configs :

The job name is added as a label job=<job_name> to any timeseries

scraped from this config.

- job_name : "prometheus"

Override the global default and scrape targets from this job every

5 seconds.

scrape_interval : 5s

static_configs :

Point to the same endpoint that Celestia is publishing on

- targets : ["localhost:26660"] Note, that Prometheus by default runs its server on:9090 . If you are running this on the same machine as your consensus node, it will collide with gRPC which runs on the same port. To avoid this, either switch off gRPC (if it's not needed), change the gRPC port inapp.toml , or run Prometheus on a different port e.g.--web.listen-address="0.0.0.0:8000"

To run the prometheus server:

bash prometheus

--config.file= " HOME /.celestia-app/config/prometheus.yml" prometheus

--config.file= " HOME /.celestia-app/config/prometheus.yml" A prometheus server can scrape metrics from multiple nodes at once; a good way of bringing together information from many machines to one place.

To visualize the information, you can useGrafana : either with their cloud option or run theopen source code yourself.

Once setup, run:

bash grafana

server grafana

server which will begin a server onlocalhost:3000 . If you open the url on your browser you will see the Grafana login page. Useadmin for both the user and password to log in.

You will need to link the prometheus server as a data source. To do that go to "Configuration" in the sidebar and then "Data Sources". Add a new data source specifying the URL of the Prometheus instance (default atlocalhost:9090). Click "Save & test" to confirm.

Lastly, you will need to setup a dashboard. You can choose to do this yourself, handpicking the metrics that are important or you can simply export an existing design. Fortunately the cosmos ecosystem has conjured a "Cosmos Dashboard". On the sidebar, click "Dashboards" and then "import". Enter the following dashboard ID: 11036 and then link it to the "Prometheus" data source you just set up. Finally click the "Import" button and the "Cosmos Dashboard" should appear.

Node exporter

Celestia's metrics include a plethora of application specific trackers but it's also important to keep an eye on system level metrics such as memory usage and disk space. This can be best achieved by runningNode Exporter . Follow the guide in the link to get set up, adding the port number to theprometheus.yml file.

Alerts

The final cherry on the cake is to integrate your monitoring system with a mechanism for producing alerts to warn you if your node has crashed or is no longer able to stay at the head of the chain.

Since we're already using Grafana, we can install theGrafana OnCall plugin. OnCall allows you to setup integrations. It could be a webhook or a direct integration into Telegram or Slack. You can find more information on Grafana'sDocs Page .
[][Edit this page on GitHub] Last updated: Previous page Troubleshooting Next page Slashing mechanics []