celestia-node metrics

This tutorial is for running metrics for your celestia-node data availability instance. This tutorial will focus on running metrics for a light node.

This tutorial assumes you have already setup your light node by following the tutorial in the celestia-node API tutorial .

Running metrics flags

You can enable thecelestia-node metric flags with the following command:

sh celestia

< node-typ e

start

--metrics.tls= < boolean

\ --metrics

--metrics.endpoint

< UR I

\ --p2p.network

< networ k

--core.ip

< UR I

celestia

< node-typ e

start

--metrics.tls= < boolean

\ --metrics

--metrics.endpoint

< UR I

\ --p2p.network

< networ k

--core.ip

< UR I

Add metrics flags to your node start command and restart your node to apply it. The metrics endpoint will gather your node's data to track your uptime.

Note that the--metrics flag enables metrics and expects an input into--metrics.endpoint .

We will go over what the endpoint will need to be in the metrics endpoint design considerations section.

Mainnet Beta

Here is an example for Mainnet Beta:

sh celestia

< node-typ e

start

--metrics.tls=true

\ --metrics

--metrics.endpoint

otel.celestia.observer

\ --core.ip

< UR I

celestia

< node-typ e

start

--metrics.tls=true

\ --metrics

--metrics.endpoint

otel.celestia.observer

\ --core.ip

< UR I

Mocha testnet

Here is an example for Mocha testnet:

sh celestia

< node-typ e

start

--metrics.tls=true

\ --metrics

--metrics.endpoint

otel.celestia-mocha.com

\ --core.ip

< UR I

--p2p.network

mocha celestia

< node-typ e

start

--metrics.tls=true

\ --metrics

--metrics.endpoint

otel.celestia-mocha.com

\ --core.ip

< UR I

--p2p.network

mocha

TLS connections

The--metrics.tls flag enables or disables a TLS connection to the OpenTelemetry Protocol metrics backend. You need to choose a boolean value (true orfalse) for this flag.

It's also common to set this flag tofalse when spinning up a local collector to check the metrics locally.

However, if the collector is hosted in the cloud as a separate entity (like in a DevOps environment), enabling TLS is a necessity for secure communication.

Here are examples of how to use it:

bash

To enable TLS connection

celestia

< node-typ e

start

--metrics.tls=true

--metrics

\ --metrics.endpoint

< UR l

\ --p2p.network

< networ k

--core.ip

< UR l

To disable TLS connection

celestia

< node-typ e

start

--metrics.tls=false

--metrics

\ --metrics.endpoint

< UR l

\ --p2p.network

< networ k

--core.ip

< UR l

To enable TLS connection

celestia

< node-typ e

start

--metrics.tls=true

--metrics

\ --metrics.endpoint

< UR I

\ --p2p.network

< networ k

--core.ip

< UR I

To disable TLS connection

celestia

< node-typ e

start

--metrics.tls=false

--metrics

\ --metrics.endpoint

< UR I

\ --p2p.network

< networ k

--core.ip

< UR I

Metrics endpoint design considerations

At the moment, the architecture of celestia-node metrics works as specified in the following ADR #010 .

Essentially, the design considerations here will necessitate running an OpenTelemetry (OTEL) collector that connects to Celestia light node.

For an overview of OTEL, check out the guide .

The ADR and the OTEL docs will help you run your collector on the metrics endpoint. This will then allow you to process the data in the collector on a Prometheus server which can then be viewed on a Grafana dashboard.

In the future, we do want to open-source some developer toolings around this infrastructure to allow for node operators to be able to monitor their data availability nodes. [][Edit this page on GitHub] Last updated: Previous page IBC relayers Next page config.toml guide []