# Getting Started Monitoring your Node

Welcome to this comprehensive guide, designed to assist you in effectively monitoring your Charon cluster and nodes, and setting up alerts based on specified parameters.

## Pre-requisites

Ensure the following software are installed:

- Docker: Find the installation guide for Ubuntuhere
- Prometheus: You can install it using the guide availablehere
- Grafana: Follow thislink
- to install Grafana

## Import Pre-Configured Charon Dashboards

- Navigate to therepository
- that contains a variety of Grafana dashboards. For this demonstration, we will utilize the Charon Dashboard json.
- In your Grafana interface, create a new dashboard and select the import option.
- Copy the content of the Charon Dashboard json from the repository and paste it into the import box in Grafana. Click "Load" to proceed.
- Finalize the import by clicking on the "Import" button. At this point, your dashboard should begin displaying metrics. Ensure your Charon client and Prometheus are operational for this to occur.

## Example Alerting Rules

To create alerts for Node-Exporter, follow these steps based on the sample rules provided on the "Awesome Prometheus alerts" page:

1. Visit theAwesome Prometheus alerts
2. page. Here, you will find lists of Prometheus alerting rules categorized by hardware, system, and services.
3. Depending on your need, select the category of alerts. For example, if you want to set up alerts for your system's CPU usage, click on the 'CPU' under the 'Host & Hardware' category.
4. On the selected page, you'll find specific alert rules like 'High CPU Usage'. Each rule will provide the PromQL expression, alert name, and a brief description of what the alert does. You can copy these rules.
5. Paste the copied rules into your Prometheus configuration file under therules
6. section. Make sure you understand each rule before adding it to avoid unnecessary alerts.
7. Finally, save and apply the configuration file. Prometheus should now trigger alerts based on these rules.

For alerts specific to Charon/Alpha, refer to the alerting rules available on thisObolNetwork/monitoring .

## Understanding Alert Rules

1. ClusterBeaconNodeDown
2. This alert is activated when the beacon node in a specified Alpha cluster is offline. The beacon node is crucial for validating transactions and producing new blocks. Its unavailability could disrupt the overall functionality of the cluster.
3. ClusterBeaconNodeSyncing
4. This alert indicates that the beacon node in a specified Alpha cluster is synchronizing, i.e., catching up with the latest blocks in the cluster.
5. ClusterNodeDown
6. This alert is activated when a node in a specified Alpha cluster is offline.
7. ClusterMissedAttestations
8. :This alert indicates that there have been missed attestations in a specified Alpha cluster. Missed attestations may suggest that validators are not operating correctly, compromising the security and efficiency of the cluster.
9. ClusterInUnknownStatus
10. : This alert is designed to activate when a node within the cluster is detected to be in an unknown state. The condition is evaluated by checking whether the maximum of the app_monitoring_readyz metric is 0.
11. ClusterInsufficientPeers
12. :This alert is set to activate when the number of peers for a node in the Alpha M1 Cluster #1 is insufficient. The condition is evaluated by checking whether the maximum of theapp_monitoring_readyz
13. equals 4.
14. ClusterFailureRate
15. : This alert is activated when the failure rate of the Alpha M1 Cluster #1 exceeds a certain threshold.
16. ClusterVCMissingValidators
17. : This alert is activated if any validators in the Alpha M1 Cluster #1 are missing.
18. ClusterHighPctFailedSyncMsgDuty

19. : This alert is activated if a high percentage of sync message duties failed in the cluster. The alert is activated if the sum of the increase in failed duties tagged with "sync_message" in the last hour divided by the sum of the increase in total duties tagged with "sync_message" in the last hour is greater than 0.1.
20. ClusterNumConnectedRelays
21. : This alert is activated if the number of connected relays in the cluster falls to 0.
22. PeerPingLatency: 1. This alert is activated if the 90th percentile of the ping latency to the peers in a cluster exceeds 500ms within 2 minutes.

## Best Practices for Monitoring Charon Nodes & Cluster

- Establish Baselines
- : Familiarize yourself with the normal operation metrics like CPU, memory, and network usage. This will help you detect anomalies.
- Define Key Metrics
- : Set up alerts for essential metrics, encompassing both system-level and Charon-specific ones.
- Configure Alerts
- : Based on these metrics, set up actionable alerts.
- Monitor Network
- : Regularly assess the connectivity between nodes and the network.
- Perform Regular Health Checks
- : Consistently evaluate the status of your nodes and clusters.
- Monitor System Logs
- : Keep an eye on logs for error messages or unusual activities.
- Assess Resource Usage
- : Ensure your nodes are neither over- nor under-utilized.
- Automate Monitoring
- : Use automation to ensure no issues go undetected.
- Conduct Drills
- : Regularly simulate failure scenarios to fine-tune your setup.
- Update Regularly
- : Keep your nodes and clusters updated with the latest software versions.

## Third-Party Services for Uptime Testing

- updown.io
- Grafana synthetic Monitoring

## Key metrics to watch to verify node health based on jobs

- Node Exporter:

CPU Usage : High or spiking CPU usage can be a sign of a process demanding more resources than it should.

Memory Usage : If a node is consistently running out of memory, it could be due to a memory leak or simply under-provisioning.

Disk I/O : Slow disk operations can cause applications to hang or delay responses. High disk I/O can indicate storage performance issues or a sign of high load on the system.

Network Usage : High network traffic or packet loss can signal network configuration issues, or that a service is being overwhelmed by requests.

Disk Space : Running out of disk space can lead to application errors and data loss.

Uptime : The amount of time a system has been up without any restarts. Frequent restarts can indicate instability in the system.

Error Rates : The number of errors encountered by your application. This could be 4xx/5xx HTTP errors, exceptions, or any other kind of error your application may log.

Latency : The delay before a transfer of data begins following an instruction for its transfer.

It is also important to check:

- NTP clock skew
- Process restarts and failures (eg. throughnode_systemd
- )
- alert on high error and panic log counts.Edit this page Previous Split existing validator private keysNext Push Metrics

to Obol Monitoring