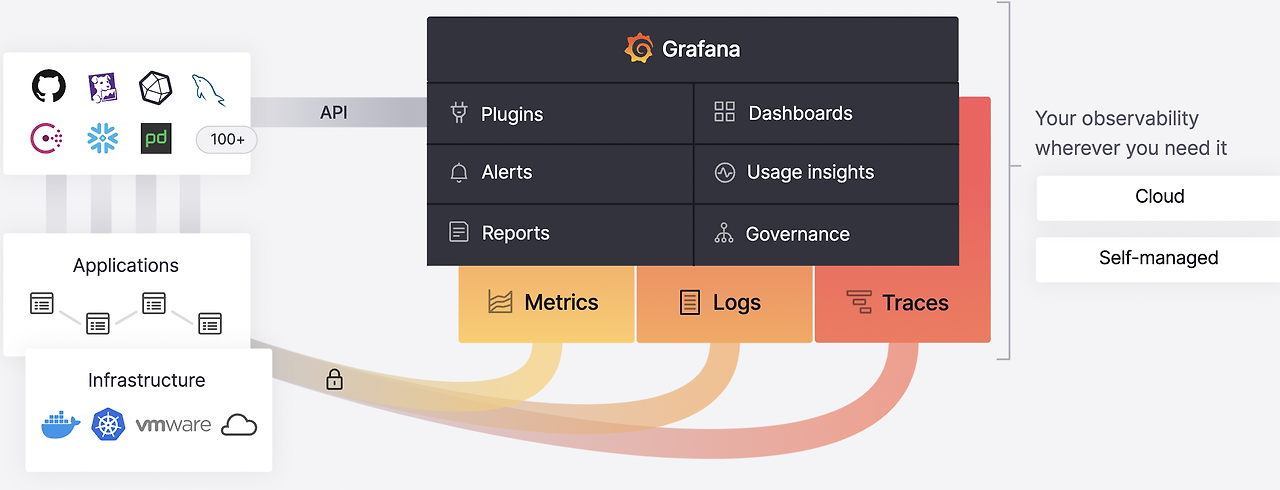
**PROJECT-3**

**Integrate Grafana with Linux Server for high cpu utilization and create a graph in Grafana.**

Linux is a family of open-source Unix-like operating systems based on the Linux kernel. Linux is the leading operating system on servers, and is one of the most prominent examples of free and open-source software collaboration.

The Grafana Linux Server integration collects metrics related to the operating system running on a node, including CPU usage, load average, memory usage, and disk and networking I/O using node\_exporter integration. You can also configure Alloy to collect logs.



**Linux Server integration for Grafana Cloud**

Linux is a family of open-source Unix-like operating systems based on the Linux kernel. Linux is the leading operating system on servers, and is one of the most prominent examples of free and open-source software collaboration.

Linux Server integration for Grafana Cloud enables you to collect metrics related to the operating system running on a node, including aspects like CPU usage, load average, memory usage, and disk and networking I/O using node\_exporter integration. It also allows you to use Grafana Alloy to scrape logs.

This integration includes 24 useful alerts and 7 pre-built dashboards to help monitor and visualize Linux Server metrics and logs.

**Before you begin**

Each Linux node being observed must have its dedicated Grafana Alloy running.

If you want to monitor more than one Linux Node with this integration, we recommend you to use the Ansible collection for Grafana Cloud to deploy Grafana Alloy to multiple machines, as described in this documentations.

**Install Linux Server integration for Grafana Cloud**

1. In your Grafana Cloud stack, click **Connections** in the left-hand menu.
2. Find **Linux Server** and click its tile to open the integration.
3. Review the prerequisites in the **Configuration Details** tab and set up Grafana Alloy to send Linux Server metrics and logs to your Grafana Cloud instance.
4. Click **Install** to add this integration’s pre-built dashboards and alerts to your Grafana Cloud instance, and you can start monitoring your Linux Server setup.

**Configuration snippets for Grafana Alloy**

**Simple mode**

These snippets are configured to scrape a single Linux Server instance running locally with default ports.

**Manually** copy and append the following snippets into your Grafana Alloy configuration file.

**Integrations snippets**

alloy

discovery.relabel "integrations\_node\_exporter" {

targets = prometheus.exporter.unix.integrations\_node\_exporter.targets

rule {

target\_label = "instance"

replacement = constants.hostname

}

rule {

target\_label = "job"

replacement = "integrations/node\_exporter"

}

}

prometheus.exporter.unix "integrations\_node\_exporter" {

disable\_collectors = ["ipvs", "btrfs", "infiniband", "xfs", "zfs"]

filesystem {

fs\_types\_exclude = "^(autofs|binfmt\_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|tmpfs|fusectl|hugetlbfs|iso9660|mqueue|nsfs|overlay|proc|procfs|pstore|rpc\_pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$"

mount\_points\_exclude = "^/(dev|proc|run/credentials/.+|sys|var/lib/docker/.+)($|/)"

mount\_timeout = "5s"

}

netclass {

ignored\_devices = "^(veth.\*|cali.\*|[a-f0-9]{15})$"

}

netdev {

device\_exclude = "^(veth.\*|cali.\*|[a-f0-9]{15})$"

}

}

prometheus.scrape "integrations\_node\_exporter" {

targets = discovery.relabel.integrations\_node\_exporter.output

forward\_to = [prometheus.relabel.integrations\_node\_exporter.receiver]

}

prometheus.relabel "integrations\_node\_exporter" {

forward\_to = [prometheus.remote\_write.metrics\_service.receiver]

rule {

source\_labels = ["\_\_name\_\_"]

regex = "node\_scrape\_collector\_.+"

action = "drop"

}

}

**Logs snippets**

**linux**

alloy

loki.relabel "integrations\_node\_exporter" {

forward\_to = [loki.write.grafana\_cloud\_loki.receiver]

rule {

target\_label = "job"

replacement = "integrations/node\_exporter"

}

rule {

target\_label = "instance"

replacement = constants.hostname

}

}

journal\_module "integrations\_node\_exporter" {

forward\_to = [loki.relabel.integrations\_node\_exporter.receiver]

}

//JOURNAL

declare "journal\_module" {

argument "forward\_to" {

optional = false

}

loki.source.journal "default" {

max\_age = "12h0m0s"

forward\_to = [loki.process.default.receiver]

relabel\_rules = loki.relabel.default.rules

}

loki.relabel "default" {

rule {

source\_labels = ["\_\_journal\_\_systemd\_unit"]

target\_label = "unit"

}

rule {

source\_labels = ["\_\_journal\_\_boot\_id"]

target\_label = "boot\_id"

}

rule {

source\_labels = ["\_\_journal\_\_transport"]

target\_label = "transport"

}

rule {

source\_labels = ["\_\_journal\_priority\_keyword"]

target\_label = "level"

}

forward\_to = []

}

loki.process "default" {

forward\_to = argument.forward\_to.value

}

}

**Advanced mode**

To instruct Grafana Alloy to scrape your Linux Server instance, go though the subsequent instructions.

The snippets provide examples to guide you through the configuration process.

First, **Manually** copy and append the following snippets into your Grafana Alloy configuration file.

Then follow the instructions below to modify the necessary variables.

**Advanced integrations snippets**

alloy

discovery.relabel "integrations\_node\_exporter" {

targets = prometheus.exporter.unix.integrations\_node\_exporter.targets

rule {

target\_label = "instance"

replacement = constants.hostname

}

rule {

target\_label = "job"

replacement = "integrations/node\_exporter"

}

}

prometheus.exporter.unix "integrations\_node\_exporter" {

disable\_collectors = ["ipvs", "btrfs", "infiniband", "xfs", "zfs"]

filesystem {

fs\_types\_exclude = "^(autofs|binfmt\_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|tmpfs|fusectl|hugetlbfs|iso9660|mqueue|nsfs|overlay|proc|procfs|pstore|rpc\_pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$"

mount\_points\_exclude = "^/(dev|proc|run/credentials/.+|sys|var/lib/docker/.+)($|/)"

mount\_timeout = "5s"

}

netclass {

ignored\_devices = "^(veth.\*|cali.\*|[a-f0-9]{15})$"

}

netdev {

device\_exclude = "^(veth.\*|cali.\*|[a-f0-9]{15})$"

}

}

prometheus.scrape "integrations\_node\_exporter" {

targets = discovery.relabel.integrations\_node\_exporter.output

forward\_to = [prometheus.relabel.integrations\_node\_exporter.receiver]

}

prometheus.relabel "integrations\_node\_exporter" {

forward\_to = [prometheus.remote\_write.metrics\_service.receiver]

rule {

source\_labels = ["\_\_name\_\_"]

regex = "node\_scrape\_collector\_.+"

action = "drop"

}

}

This integration uses prometheus.exporter.unix the component to collect system metrics.

The supplied configuration is tuned to exclude any metrics from the exporter which are not used by the integration’s dashboards, alerts, or recording rules. If a broader configuration which includes additional metrics is desired, the prometheus.exporter.unix component can be adjusted accordingly.

**Advanced logs snippets**

**linux**

alloy

loki.relabel "integrations\_node\_exporter" {

forward\_to = [loki.write.grafana\_cloud\_loki.receiver]

rule {

target\_label = "job"

replacement = "integrations/node\_exporter"

}

rule {

target\_label = "instance"

replacement = constants.hostname

}

}

journal\_module "integrations\_node\_exporter" {

forward\_to = [loki.relabel.integrations\_node\_exporter.receiver]

}

filelogs\_module "integrations\_node\_exporter" {

forward\_to = [loki.relabel.integrations\_node\_exporter.receiver]

}

//FILELOGS

declare "filelogs\_module" {

argument "forward\_to" {

optional = false

}

local.file\_match "default" {

path\_targets = [{

\_\_address\_\_ = "localhost",

\_\_path\_\_ = "/var/log/{syslog,messages,\*.log}",

}]

}

loki.source.file "default" {

targets = local.file\_match.default.targets

forward\_to = argument.forward\_to.value

}

}

//JOURNAL

declare "journal\_module" {

argument "forward\_to" {

optional = false

}

loki.source.journal "default" {

max\_age = "12h0m0s"

forward\_to = [loki.process.default.receiver]

relabel\_rules = loki.relabel.default.rules

}

loki.relabel "default" {

rule {

source\_labels = ["\_\_journal\_\_systemd\_unit"]

target\_label = "unit"

}

rule {

source\_labels = ["\_\_journal\_\_boot\_id"]

target\_label = "boot\_id"

}

rule {

source\_labels = ["\_\_journal\_\_transport"]

target\_label = "transport"

}

rule {

source\_labels = ["\_\_journal\_priority\_keyword"]

target\_label = "level"

}

forward\_to = []

}

loki.process "default" {

forward\_to = argument.forward\_to.value

}

}

This integration uses the [loki.source.journal](https://grafana.com/docs/alloy/latest/reference/components/loki.source.journal/), and [local.file\_match](https://grafana.com/docs/alloy/latest/reference/components/local.file_match/) components to collect system logs.

This includes the systemd journal and the file(s) matching /var/log/{syslog,messages,\*.log}.

If you wish to capture other log files, you must add new new maps to the path\_targets list parameter of the local.file\_match component. If you wish for these additionally captured logs to be labeled so that they can be seen in Linux Node integration logs dashboard, the entry must include the same instance and job labels.

**Grafana Agent static configuration (deprecated)**

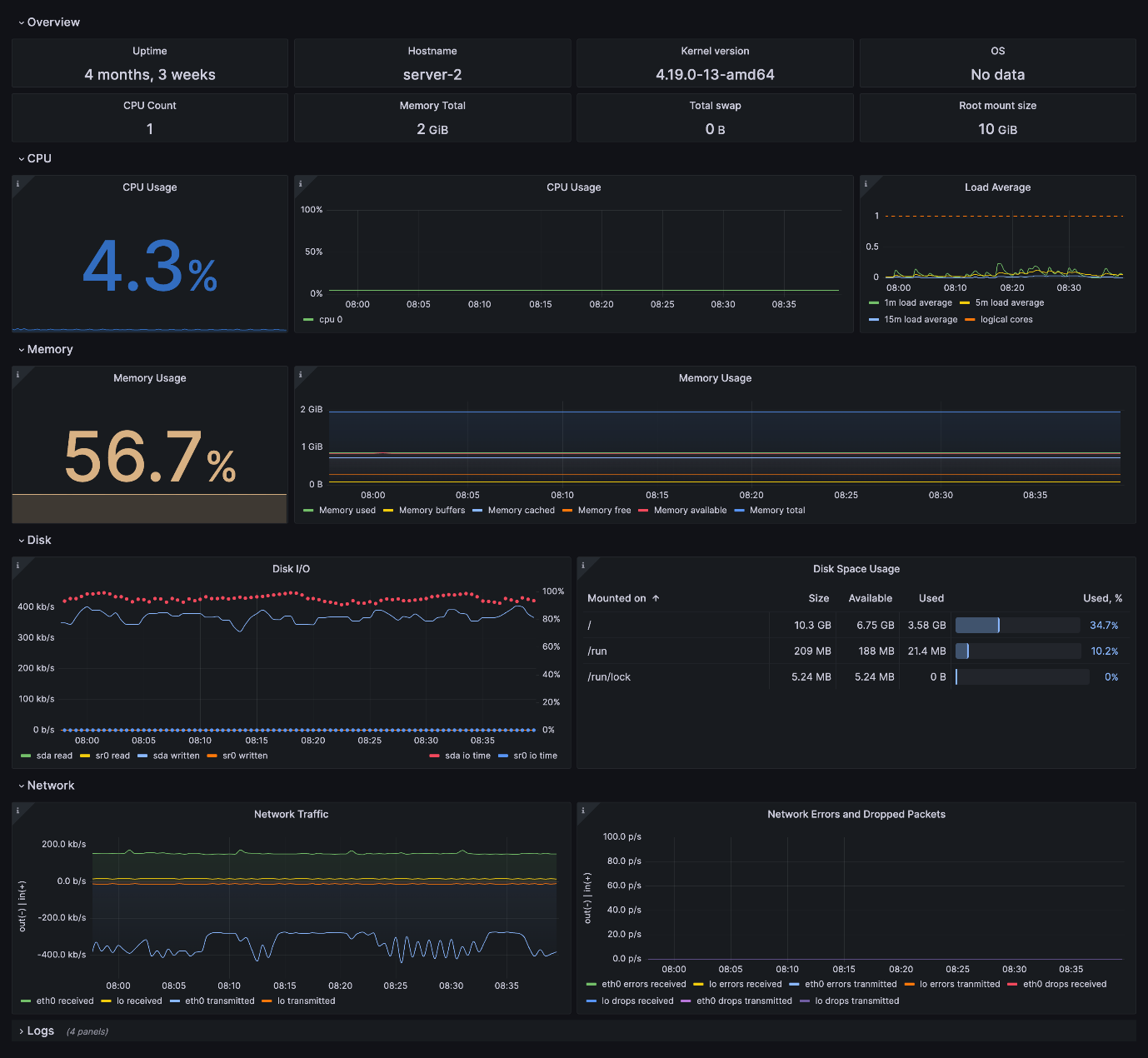
The following section shows configuration for running Grafana Agent in static mode which is deprecated. You should use Grafana Alloy for all new deployments.

**Dashboards**

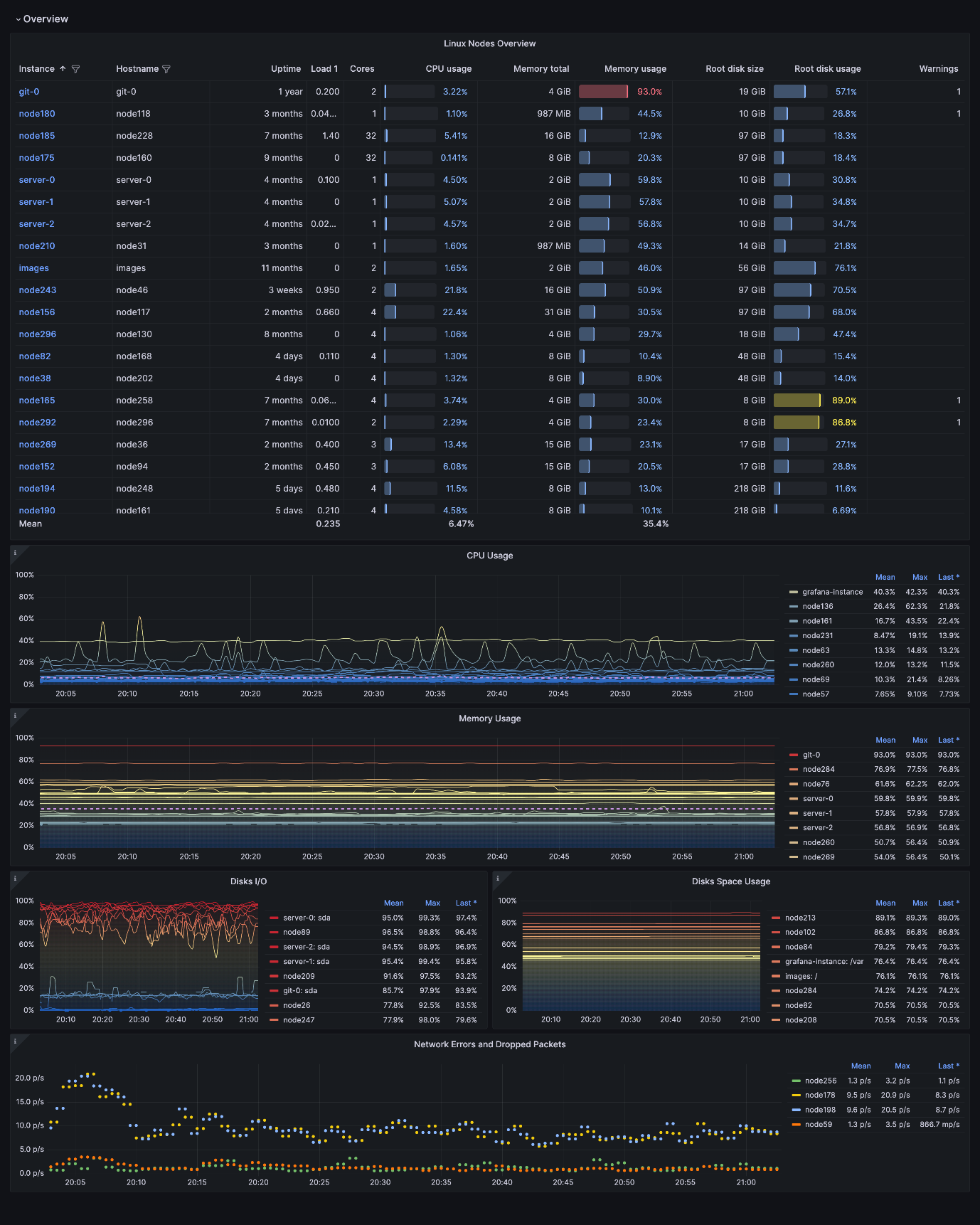
The Linux Server integration installs the following dashboards in your Grafana Cloud instance to help monitor your system.

* Linux node / CPU and system
* Linux node / filesystem and disks
* Linux node / fleet overview
* Linux node / logs
* Linux node / memory
* Linux node / network
* Linux node / overview

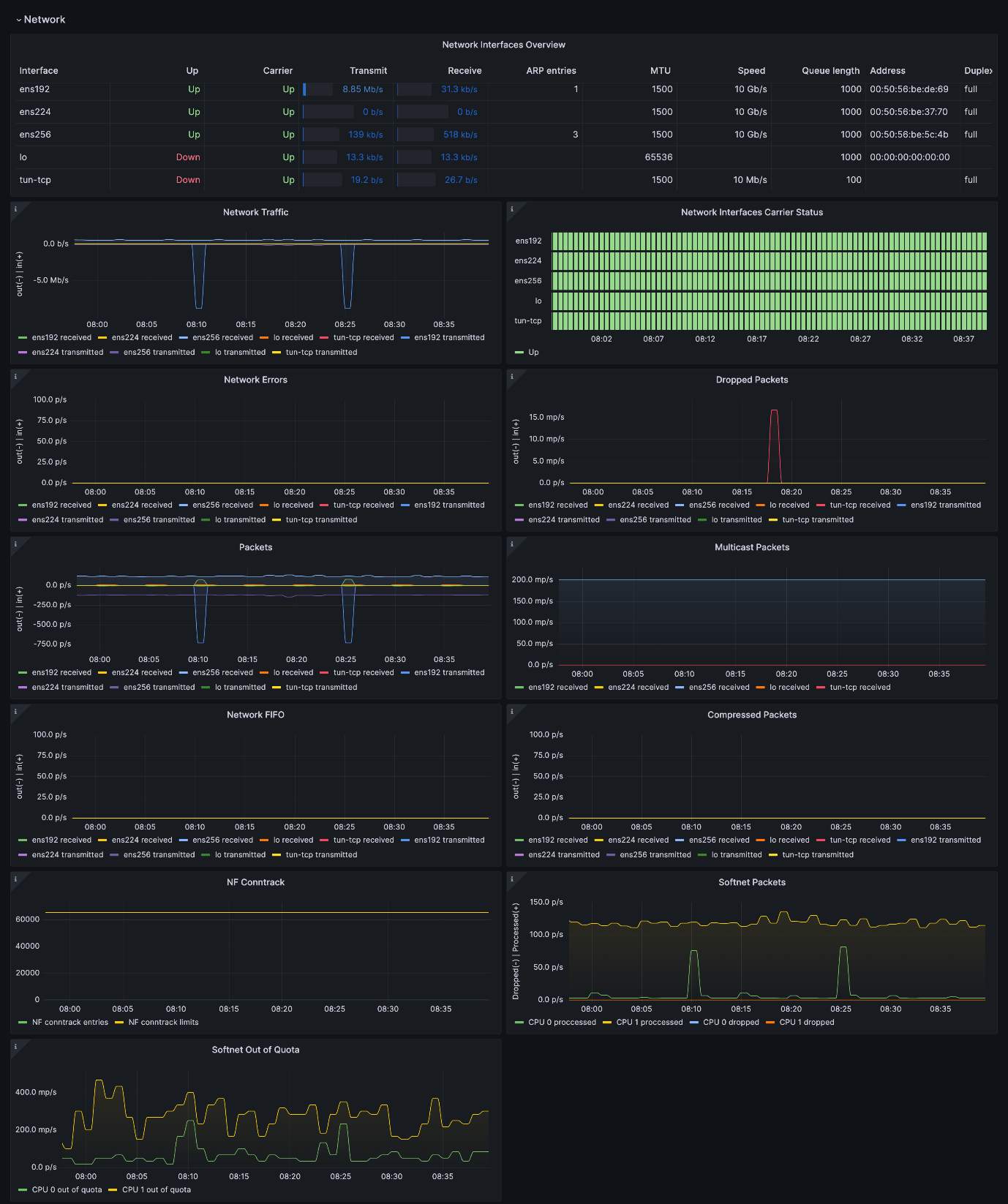
**Node overview dashboard**



**Fleet overview dashboard**



**Drill down dashboards: Network interfaces**



**Alerts**

The Linux Server integration includes the following useful alerts:

**node-exporter-filesystem**

| **Alert** | **Description** |
| --- | --- |
| NodeFilesystemAlmostOutOfSpace | Warning: Filesystem has less than 5% space left. |
| NodeFilesystemAlmostOutOfSpace | Critical: Filesystem has less than 3% space left. |
| NodeFilesystemFilesFillingUp | Warning: Filesystem is predicted to run out of inodes within the next 24 hours. |
| NodeFilesystemFilesFillingUp | Critical: Filesystem is predicted to run out of inodes within the next 4 hours. |
| NodeFilesystemAlmostOutOfFiles | Warning: Filesystem has less than 5% inodes left. |
| NodeFilesystemAlmostOutOfFiles | Critical: Filesystem has less than 3% inodes left. |

**node-exporter**

| **Alert** | **Description** |
| --- | --- |
| NodeCPUHighUsage | Info: High CPU usage. |
| NodeClockNotSynchronising | Warning: Clock not synchronising. |
| NodeClockSkewDetected | Warning: Clock skew detected. |
| NodeDiskIOSaturation | Warning: Disk IO queue is high. |
| NodeFileDescriptorLimit | Warning: Kernel is predicted to exhaust file descriptors limit soon. |
| NodeHasRebooted | Info: Node has rebooted. |
| NodeHighNumberConntrackEntriesUsed | Warning: Number of conntrack are getting close to the limit. |
| NodeMemoryHighUtilization | Warning: Host is running out of memory. |
| NodeMemoryMajorPagesFaults | Warning: Memory major page faults are occurring at very high rate. |
| NodeNetworkReceiveErrs | Warning: Network interface is reporting many receive errors. |
| NodeNetworkTransmitErrs | Warning: Network interface is reporting many transmit errors. |
| NodeProcessesCountIsHigh | Warning: There is more than 400 running processes on host. |
| NodeRAIDDegraded | Critical: RAID Array is degraded. |
| NodeRAIDDiskFailure | Warning: Failed device in RAID array. |
| NodeSystemSaturation | Warning: System saturated, load per core is very high. |
| NodeSystemdServiceCrashlooping | Warning: Systemd service keeps restaring, possibly crash looping. |
| NodeSystemdServiceFailed | Warning: Systemd service has entered failed state. |
| NodeTextFileCollectorScrapeError | Warning: Node Exporter text file collector failed to scrape. |

……………………………………………………………………………………………………………………………………………………………