Hewlett Packard Enterprise

CSM AND SMA MONITORING FOR HPE CRAY EX SYSTEMS

Matt Silvia
SMA Software Quality Engineer

April, 2023

AGENDA

- CSM SYSTEM HEALTH
- SMA ARCHITECTURE
- SMA COMPONENTS
- SMA ALARMS AND NOTIFICATIONS
- SLINGSHOT MONITORING IN SMA

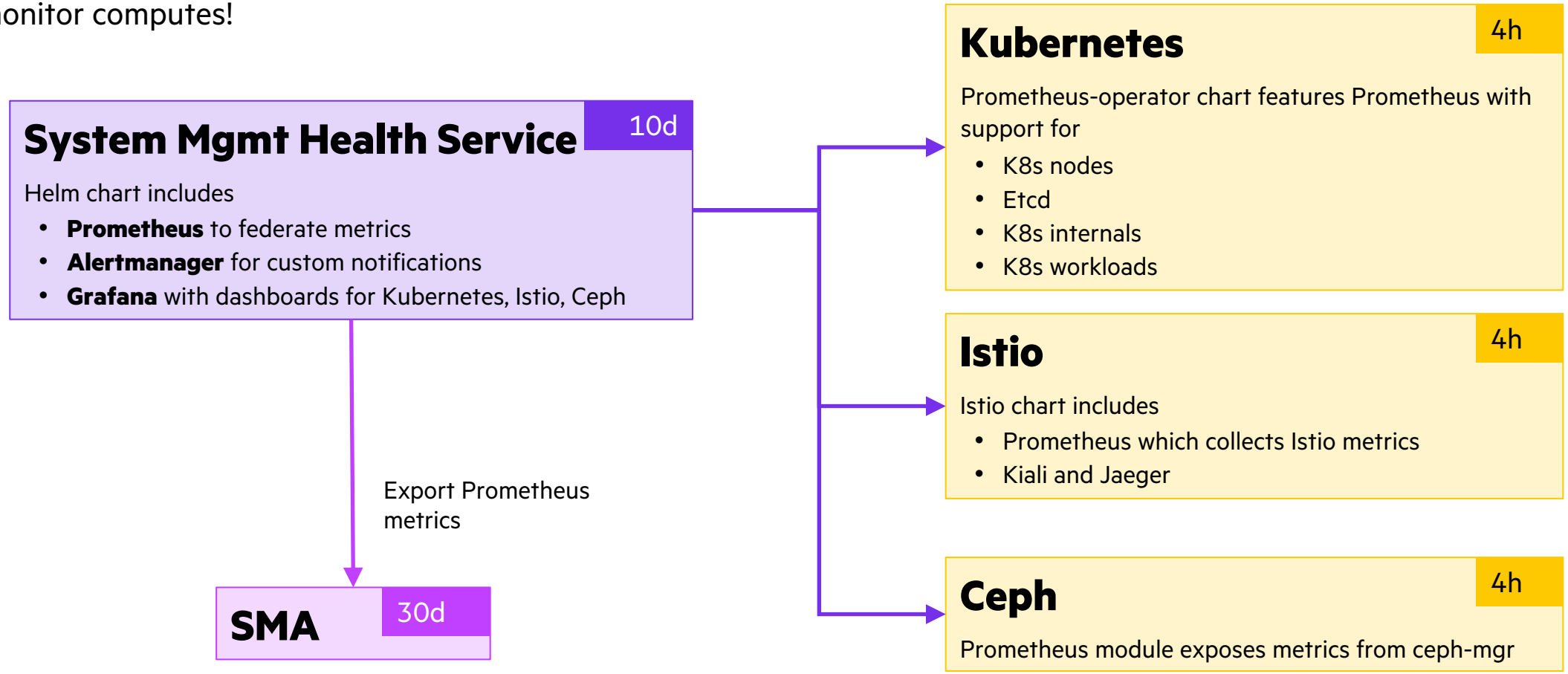
CSM SYSTEM HEALTH

- Prometheus
- Grafana Dashboards

SYSTEM MANAGEMENT HEALTH SERVICE

Is the system healthy?

- Independent from the System Monitoring Application (SMA)
- Does not monitor computes!



INDUSTRY STANDARD TOOLS

- Prometheus is the de-facto standard cloud-native metrics and monitoring tool
  - Prometheus operator provides custom resource definitions
    - Scrape metrics from service endpoints
  - Prometheus alerting rules triggers alerts to Alertmanager
  - Alertmanager manages the silencing, inhibition, aggregation, and sending out of notifications
- Grafana supports pulling data from Prometheus
  - Dashboards are readily available
- Istio supports service mesh tracing with Jaeger and observability with Kiali
- Customer integration
  - Customize Alertmanager notifications
    - Email, Slack, custom web hook
  - Run components “off system”
    - Integrate with existing Prometheus infrastructure

ISTIO WITH KIALI , JAEGER , AND PROMETHEUS

- Kiali
  - Observability console for Istio with service mesh configuration and validation capabilities
  - Helps you understand the structure and health of your service mesh by monitoring traffic flow to infer the topology and report errors
  - Provides detailed metrics and a basic Grafana integration, which can be used for advanced queries
  - Distributed tracing is provided by integration with Jaeger
  - https://kiali-istio.cmn.SYSTEM_DOMAIN_NAME
  - Documentation https://kiali.io/
- Jaeger
  - Distributed transaction monitoring
  - Performance and latency optimization
  - Root cause analysis
  - Service dependency analysis
  - Distributed context propagation
  - https://jaeger-istio.cmn.SYSTEM_DOMAIN_NAME
  - Documentation https://www.jaegertracing.io/
- Prometheus
  - Monitoring system and time series database
  - Record metrics that track the health of Istio and of applications within the service mesh
  - https://prometheus-istio.cmn.SYSTEM_DOMAIN_NAME
  - Documentation https://prometheus.io/

HEALTH CHECKS

- Prometheus alerts provide coverage across infrastructure and platform
- Coarse-grained and comprehensive, as opposed to fine-grained and exhaustive
- Supports preventive and diagnostic use cases

NON-COMPUTE NODES	UTILITY STORAGE	CONTAINER ORCHESTRATION	SERVICE MESH	WORKLOADS
• CPU and memory utilization • Local storage utilization • Network I/O errors and latency • Clock skew	• Ceph status • Storage utilization • Disk I/O errors and latency	• Kubernetes status • API errors • CPU and memory overcommitments	• Istio status • Service availability • Service request rates • Service response statuses and latency	• Status of pods, deployments, stateful sets, daemon sets, jobs • CPU, memory, network, and storage utilization and errors

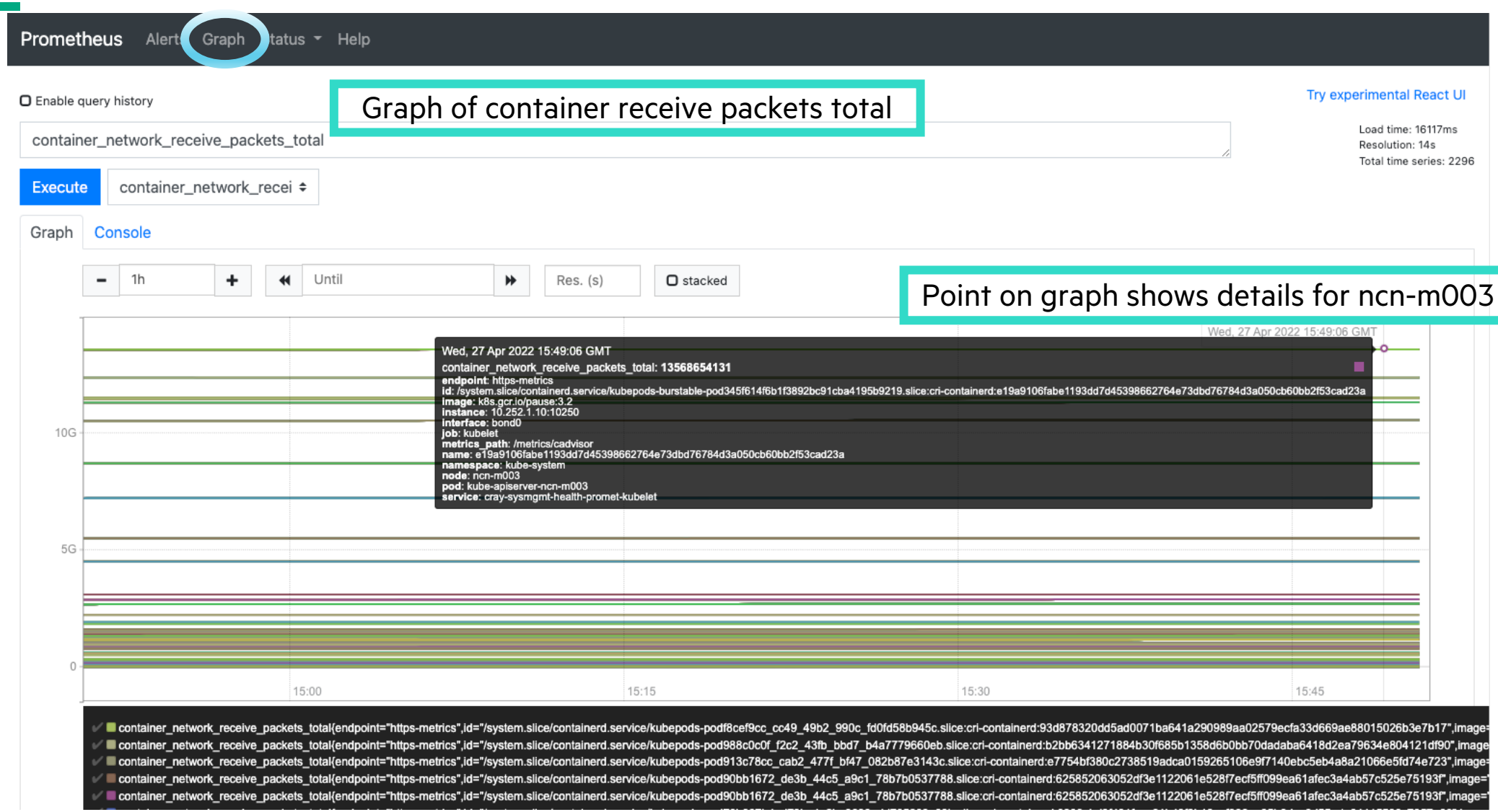
RETRIEVING ALERTS FROM PROMETHEUS

```
ncn# kubectl -n sysmgmt-health get svc cray-sysmgmt-health-promet-prometheus
NAME                                      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
cray-sysmgmt-health-promet-prometheus     ClusterIP   10.21.141.187    <none>        9090/TCP   34d
ncn# curl -s http://10.21.141.187:9090/api/v1/alerts |jq -j '.data' | grep alertname | sort | uniq -c
     12          "alertname": "CPUThrottlingHigh",
    108          "alertname": "IstioHighRequestLatency",
    103          "alertname": "IstioLatency99Percentile",
      1          "alertname": "IstioLowTotalRequestRate",
      1          "alertname": "KubeAPIErrorBudgetBurn",
      1          "alertname": "KubeDeploymentReplicasMismatch",
    131          "alertname": "KubeJobCompletion",
    130          "alertname": "KubeJobFailed",
      2          "alertname": "KubePersistentVolumeFillingUp",
      1          "alertname": "KubePodCrashLooping",
      1          "alertname": "NodeClockNotSynchronising",
      1          "alertname": "PodReadinessProbeFailure",
      1          "alertname": "PostgresqlFollowerReplicationLagSMA",
      2          "alertname": "PostgresqlHighRollbackRate",
      1          "alertname": "PostgresqlInactiveReplicationSlot",
      3          "alertname": "PostgresqlNotEnoughConnections",
      3          "alertname": "TargetDown",
      1          "alertname": "Watchdog",
```

RETRIEVING THE LATEST ALERT FROM PROMETHEUS

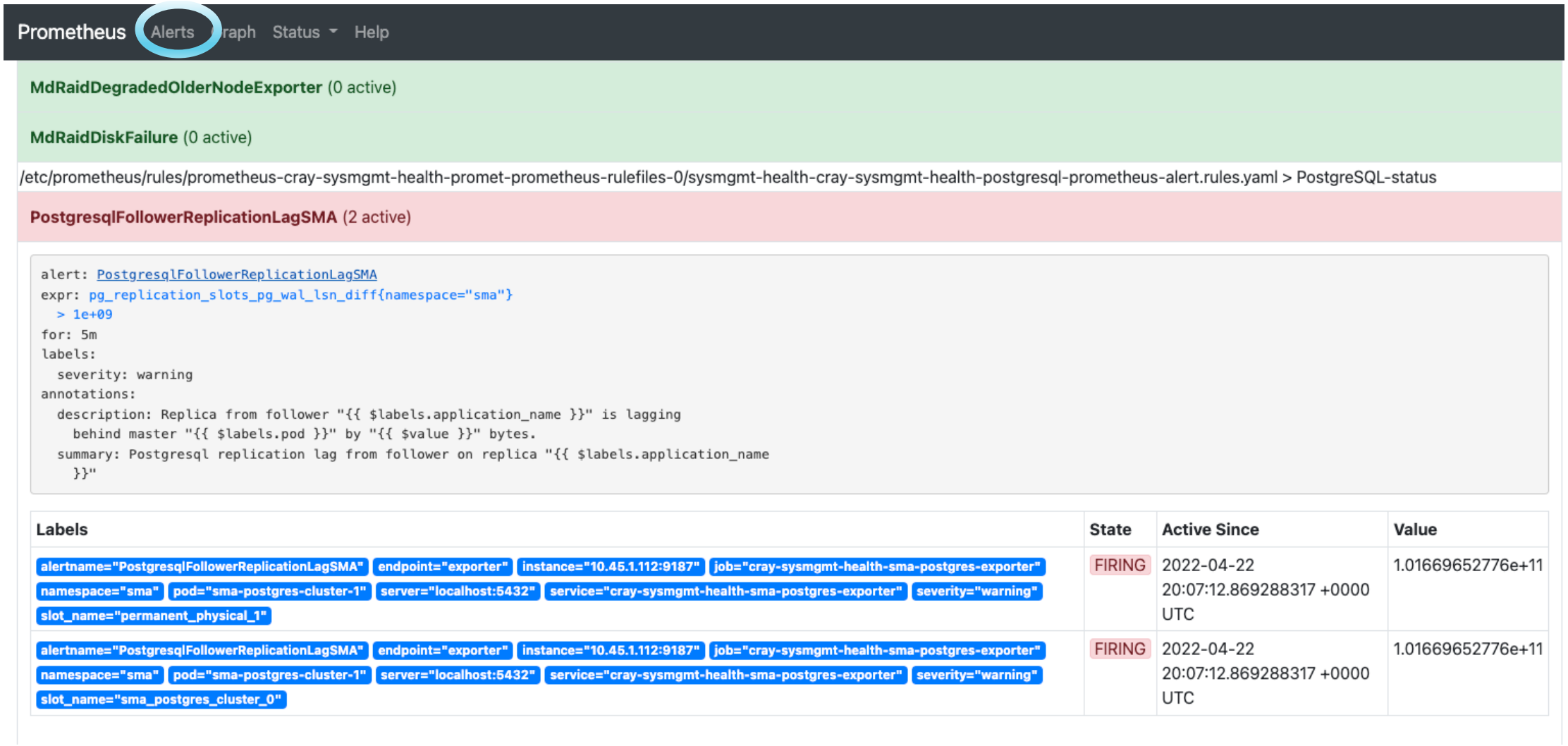
```
ncn# curl -s http://10.21.141.187:9090/api/v1/alerts |jq -j '.data.alerts \
| map(select(.labels.alertname == "CPUThrottlingHigh")) | max_by(.activeAt)'
{
  "labels": {
    "alertname": "CPUThrottlingHigh",
    "container": "manager",
    "namespace": "gatekeeper-system",
    "pod": "gatekeeper-controller-manager-588d6476db-d5g8v",
    "severity": "info"
  },
  "annotations": {
    "message": "28.03% throttling of CPU in namespace gatekeeper-system for container manager
in pod gatekeeper-controller-manager-588d6476db-d5g8v.",
    "runbook_url": "https://github.com/kubernetes-monitoring/kubernetes-
mixin/tree/master/runbook.md#alert-name-cputhrottlinghigh"
  },
  "state": "pending",
  "activeAt": "2022-04-27T16:11:07.129355508Z",
  "value": "2.8030608135320173e-01"
}
```

PROMETHEUS - GRAPH



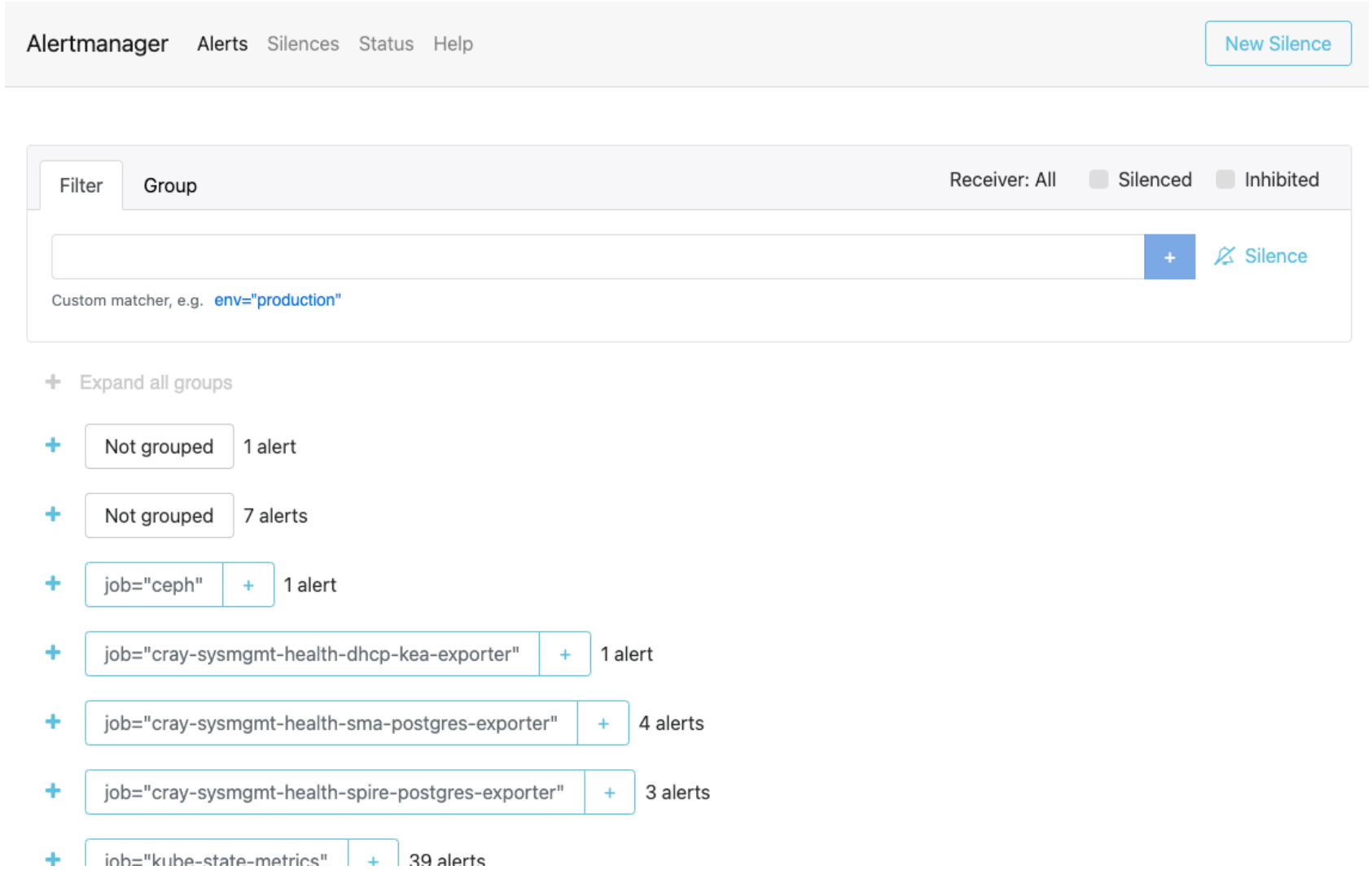
https://prometheus.cmn.SYSTEM_DOMAIN_NAME

PROMETHEUS - ALERTS



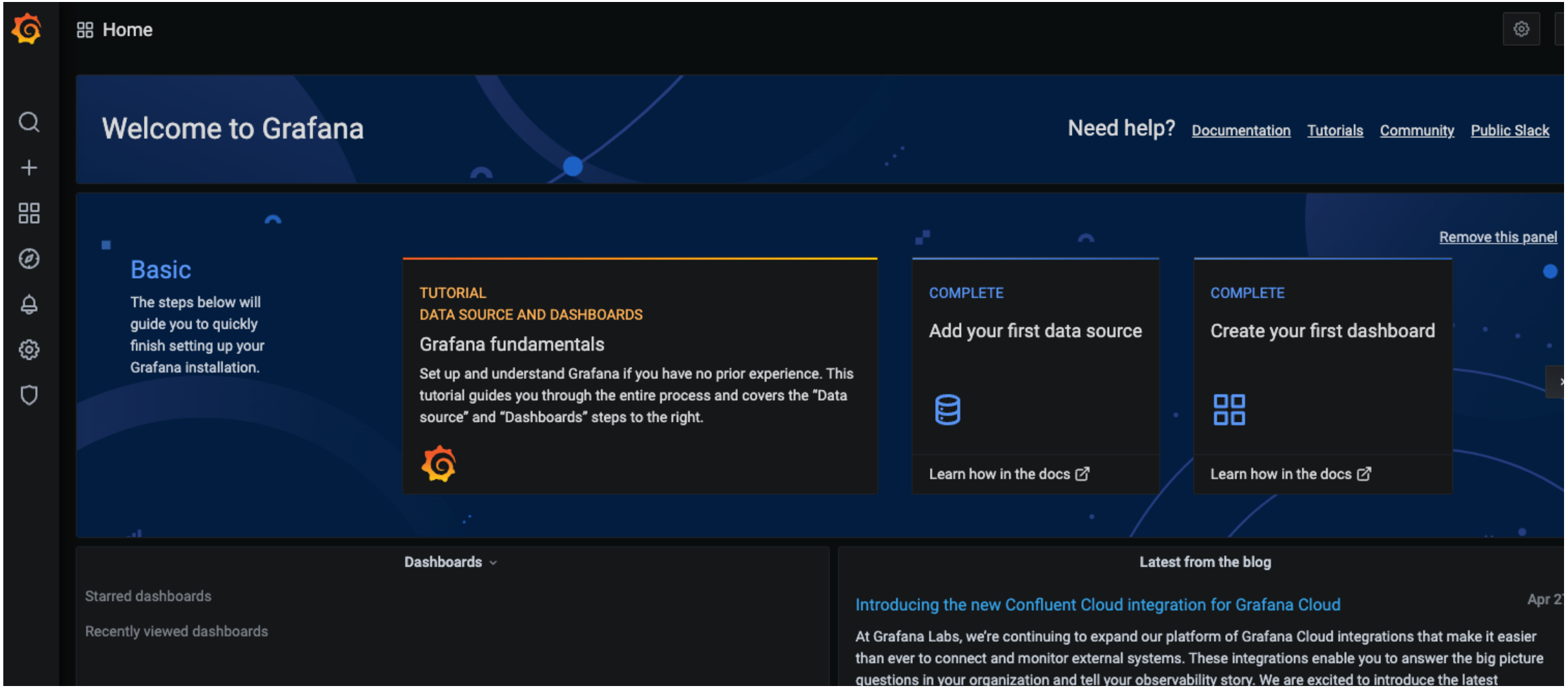
https://prometheus.cmn.SYSTEM_DOMAIN_NAME

ALERTMANAGER



https://alertmanager.cmn.SYSTEM_DOMAIN_NAME

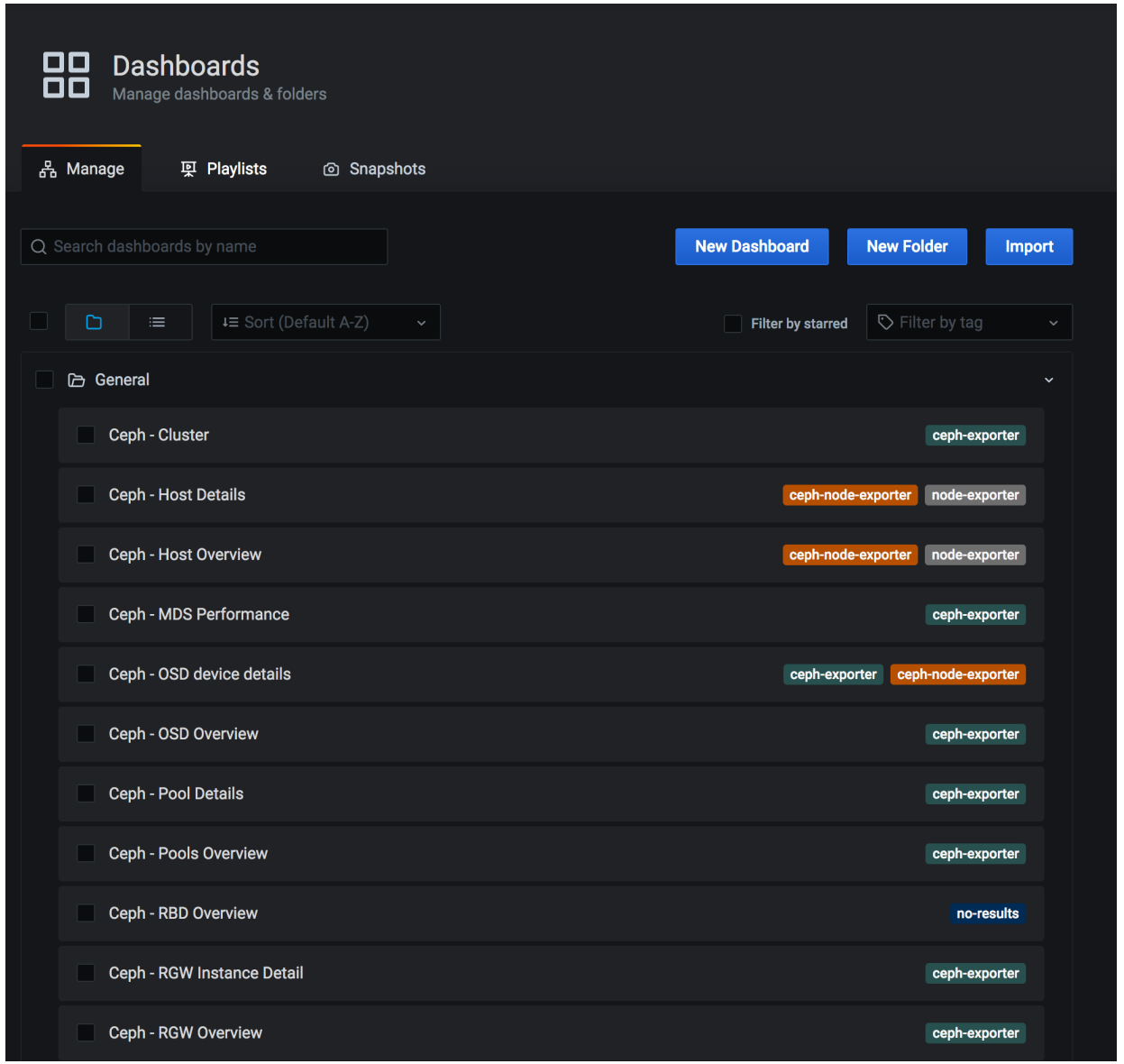
GRAFANA


Home
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
Remove this panel
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Introducing the new Confluent Cloud integration for Grafana Cloud
At Grafana Labs, we're continuing to expand our platform of Grafana Cloud integrations that make it easier than ever to connect and monitor external systems. These integrations enable you to answer the big picture questions in your organization and tell your observability story. We are excited to introduce the latest

GRAFANA DASHBOARDS CATALOG

- Uses Keycloak authentication/authorization
- Secured with TLS sharing cluster certificate bundle
- About 40 included dashboards
  - Ceph
  - CoreDNS
  - Etcd
  - ETCD Clusters
  - Istio
  - Kea-dhcp
  - Kubernetes
  - Node Exporter
  - Nodes
  - PostgreSQL
  - Prometheus

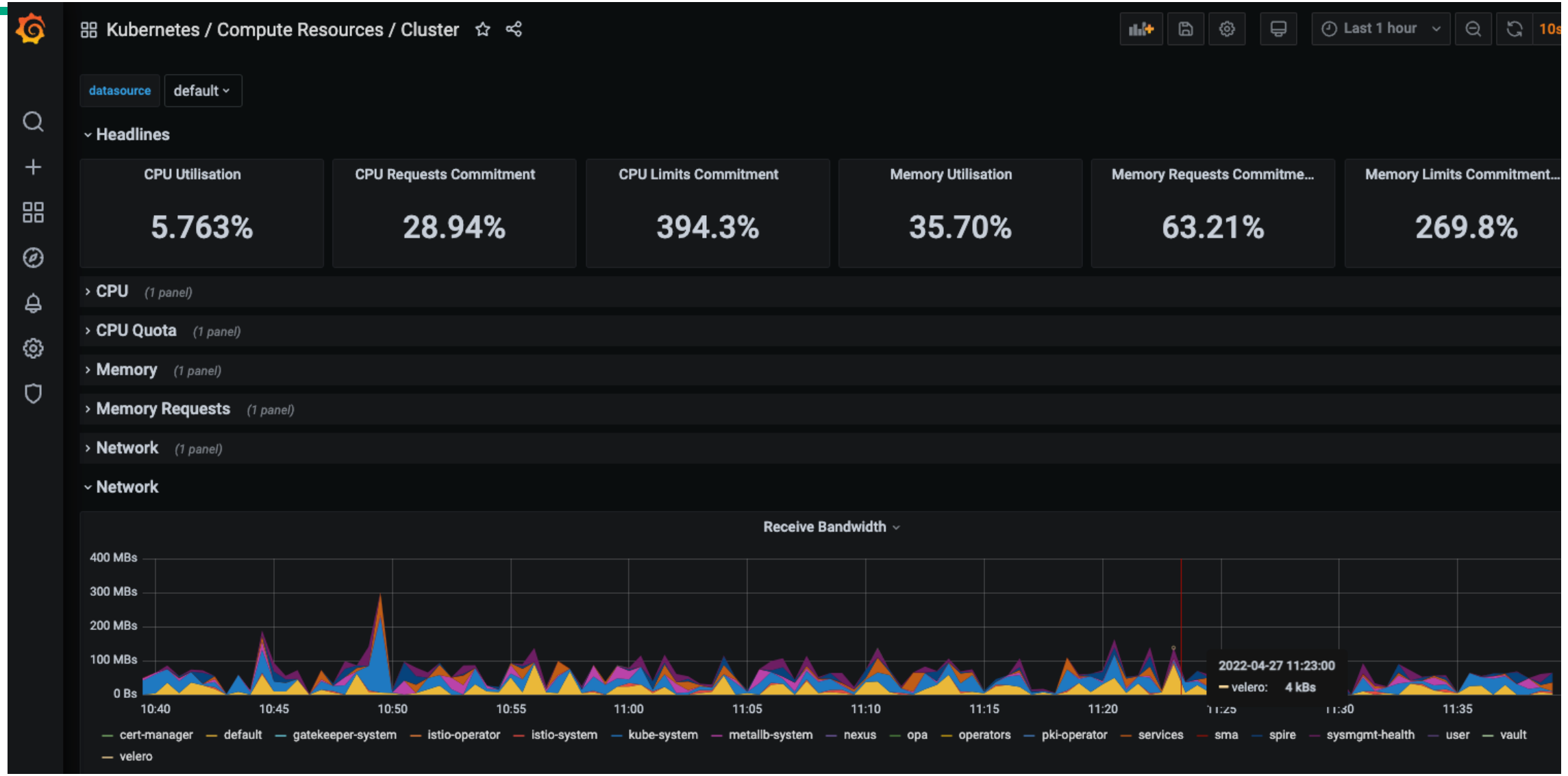
https://grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards



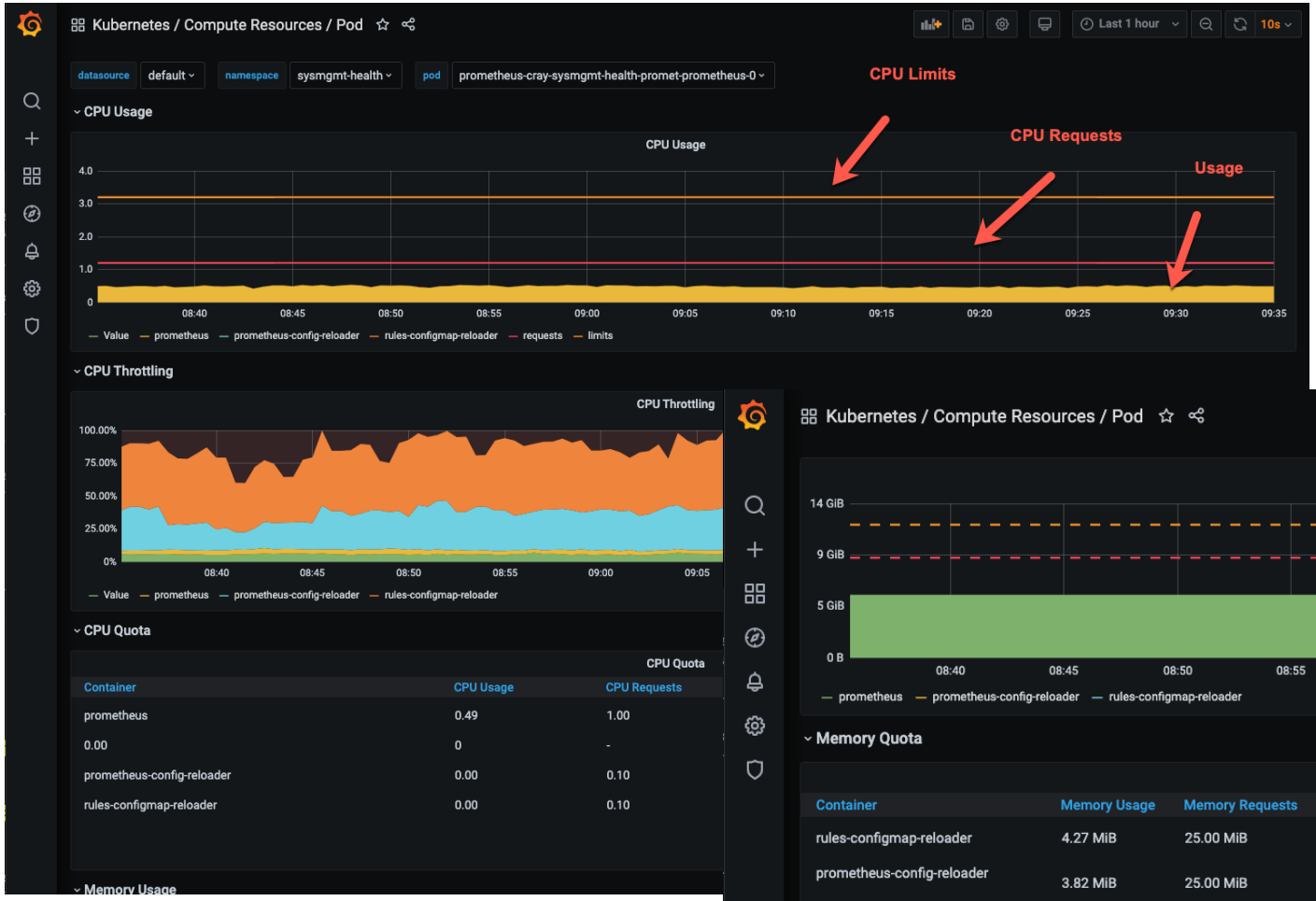
GRAFANA DASHBOARDS: ETCD

- Nodes up (quorum)
- RPC Rate
- Active Streams
- DB Size
- Disk Sync Duration
- Memory
- Client Traffic in
- Client Traffic Out
- Peer Traffic In
- Peer Traffic Out
- Raft proposals
- Total Leader Elections Per day

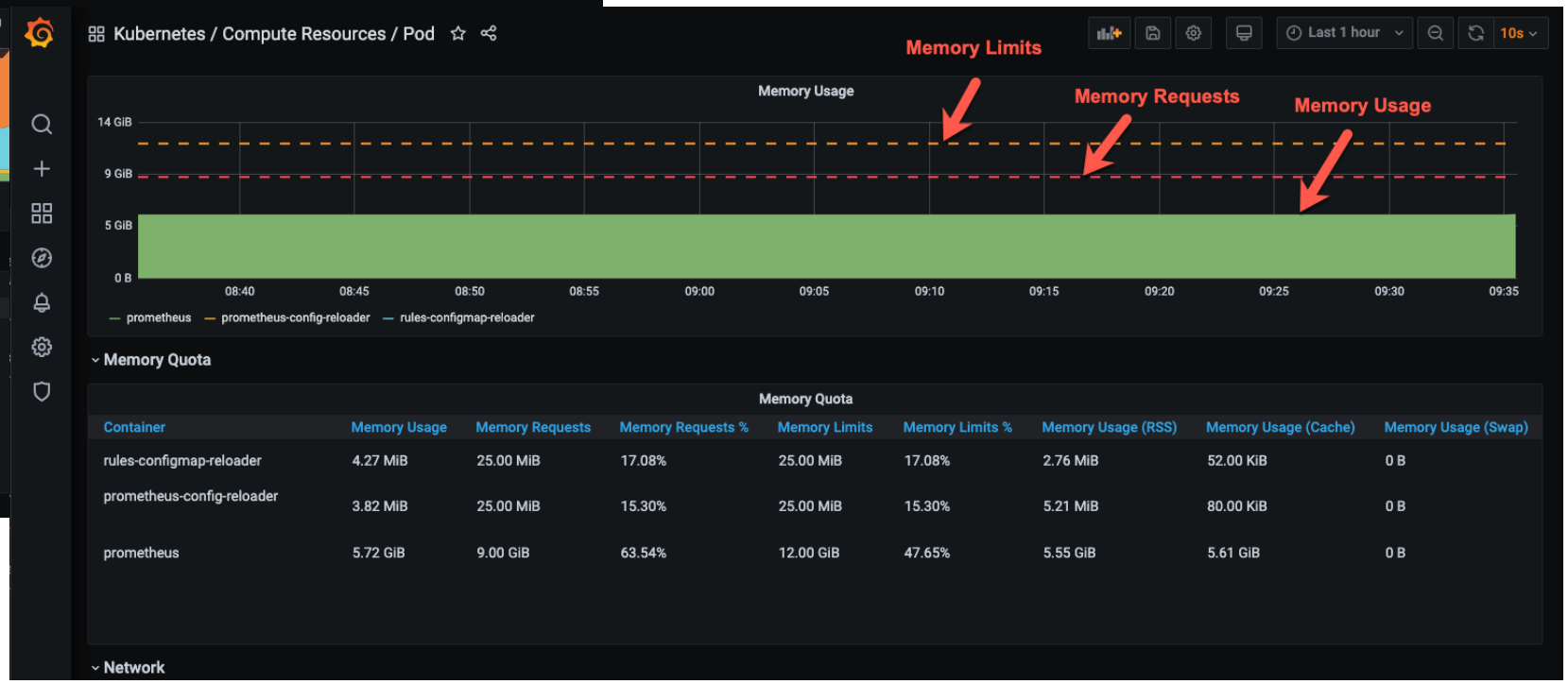
GRAFANA DASHBOARDS: KUBERNETES CLUSTER



GRAFANA DASHBOARDS: KUBERNETES POD REQUESTS AND LIMITS



CPU usage



Memory Usage

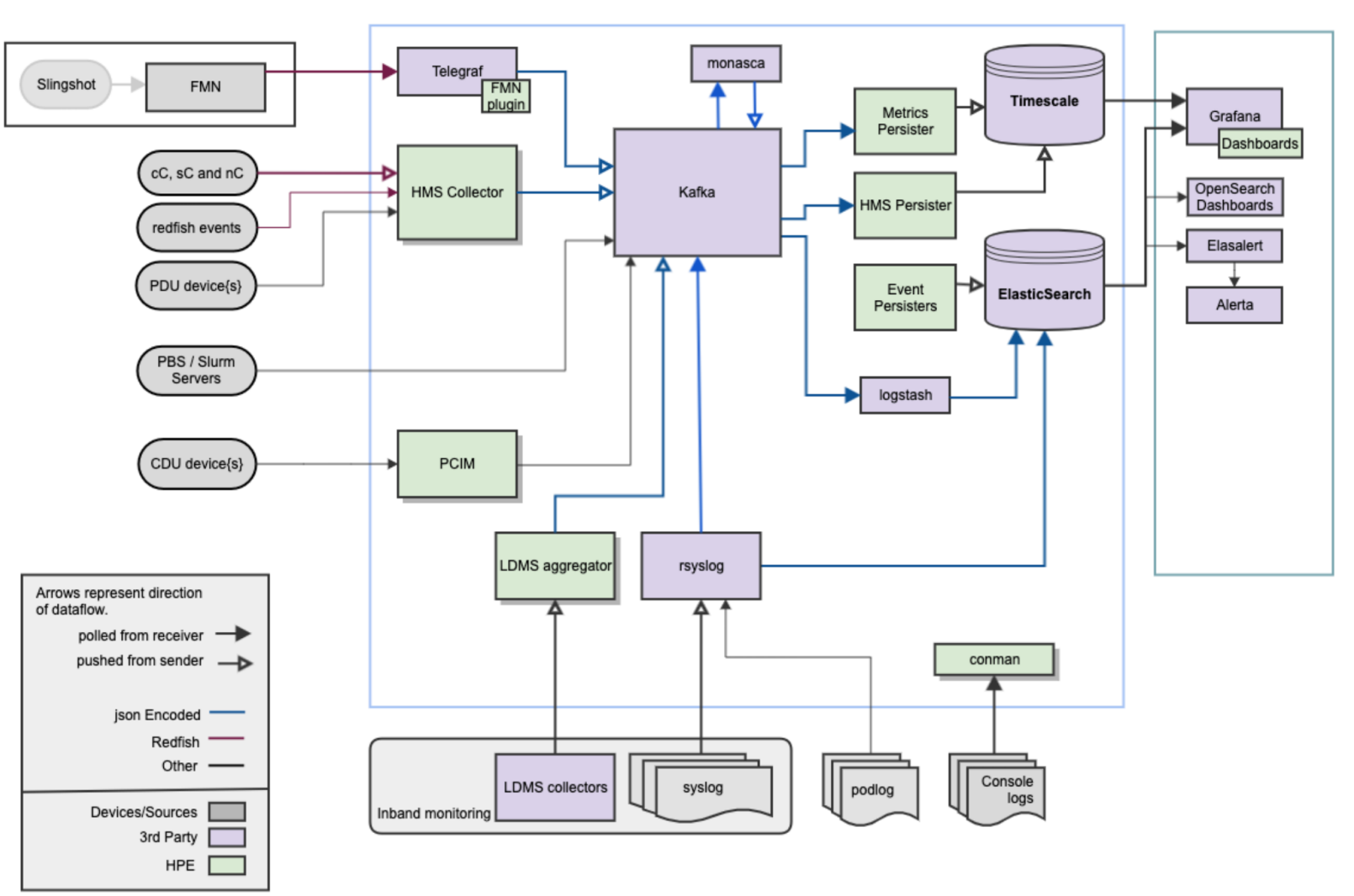
SMA MONITORING

- System Monitoring Application
- LDMS
- Telemetry API
- SMA-Grafana
- Dashboards

SYSTEM MONITORING APPLICATION

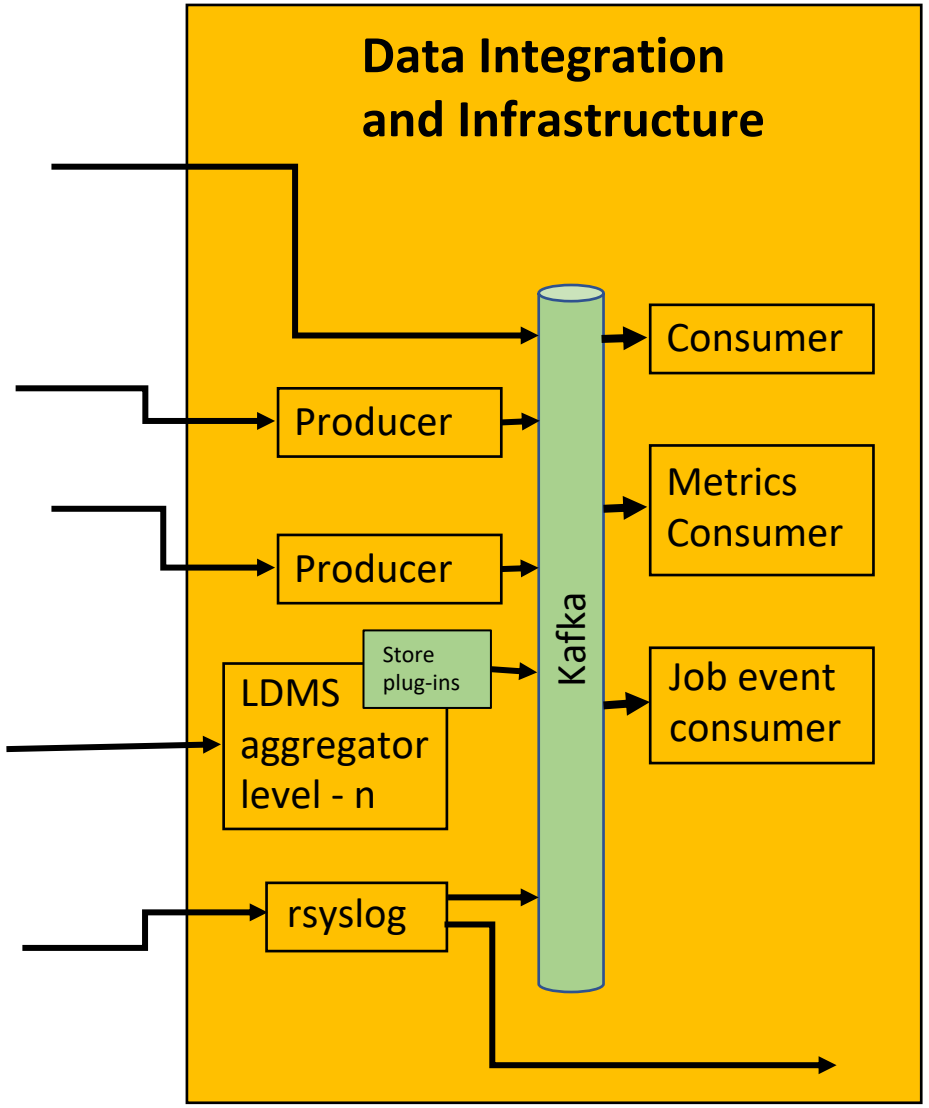
- Tightly-integrated monitoring system
- Provides detailed telemetry information from multiple subsystems:
  - Fabric
  - Environmental
  - Network
  - Storage
  - Operating systems (vmstat and iostat metrics)
- Feeds into a common message bus (Kafka), is persisted, and available via UI infrastructure
  - SAT has user interfaces that integrate with the System Monitoring Framework
- SMA alarms and notifications subsystem monitors metric data
  - Provides a way to notify administrators when select metric data is outside of normal operating values
  - SMA includes some example alarms
  - Can be extended with site defined alarms

SYSTEM MONITORING APPLICATION FRAMEWORK DIAGRAM



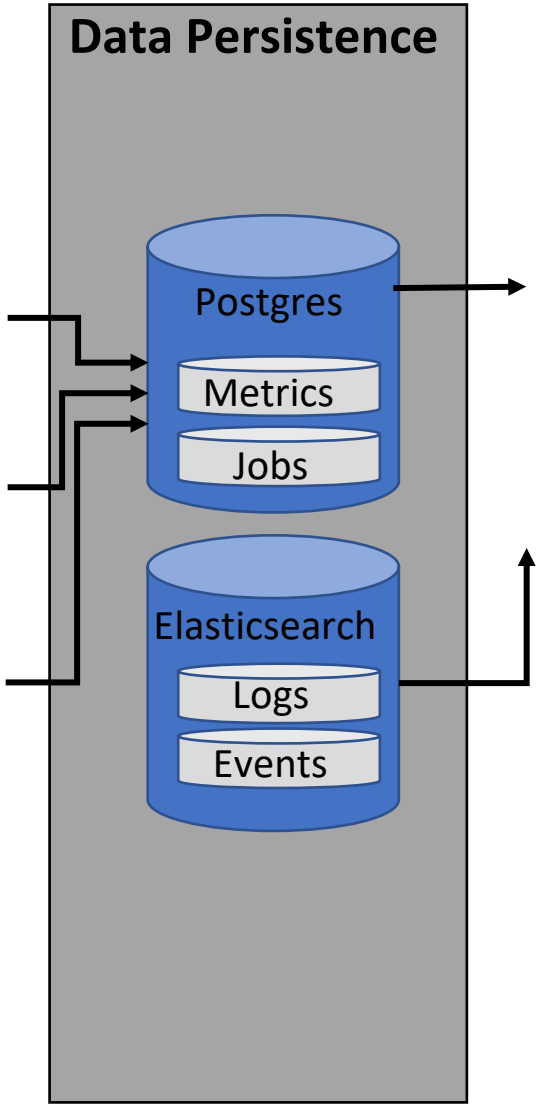
DATA INTEGRATION AND INFRASTRUCTURE LAYER

- Uses a distributed streaming platform to publish and subscribe to streams of records
- Apache Kafka
  - A distributed publish-subscribe messaging system
  - Easy to scale horizontally
  - Supports multiple subscribers and balances consumers during failure
  - Persists messages on disk
  - Supports multiple client-side APIs for consumers and producers
  - Commonly referred to as the “Kafka Bus”



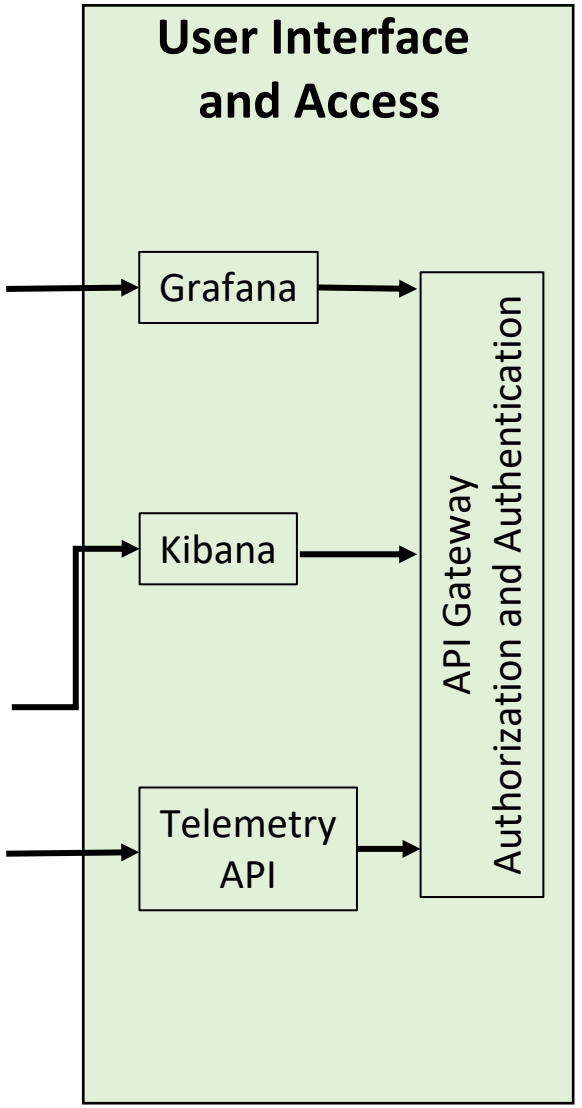
DATA PERSISTENCE LAYER

- Store telemetry data from Cray defined producers, collectors and aggregators
  - It is possible for customers to develop their own data collectors but the data they collect would not be stored in the SMF data persistence databases
    - Data from custom collectors can be streamed via the Telemetry API
- Two main responsibilities:
  - Time scale database (TSDB) optimized for handling time series data
  - Convert raw data into internal documents and store them with full text search
- Two main technologies:
  - Postgres for TSDB
  - Elasticsearch search engine
- Administrators and users should NOT attempt to read or update these databases directly

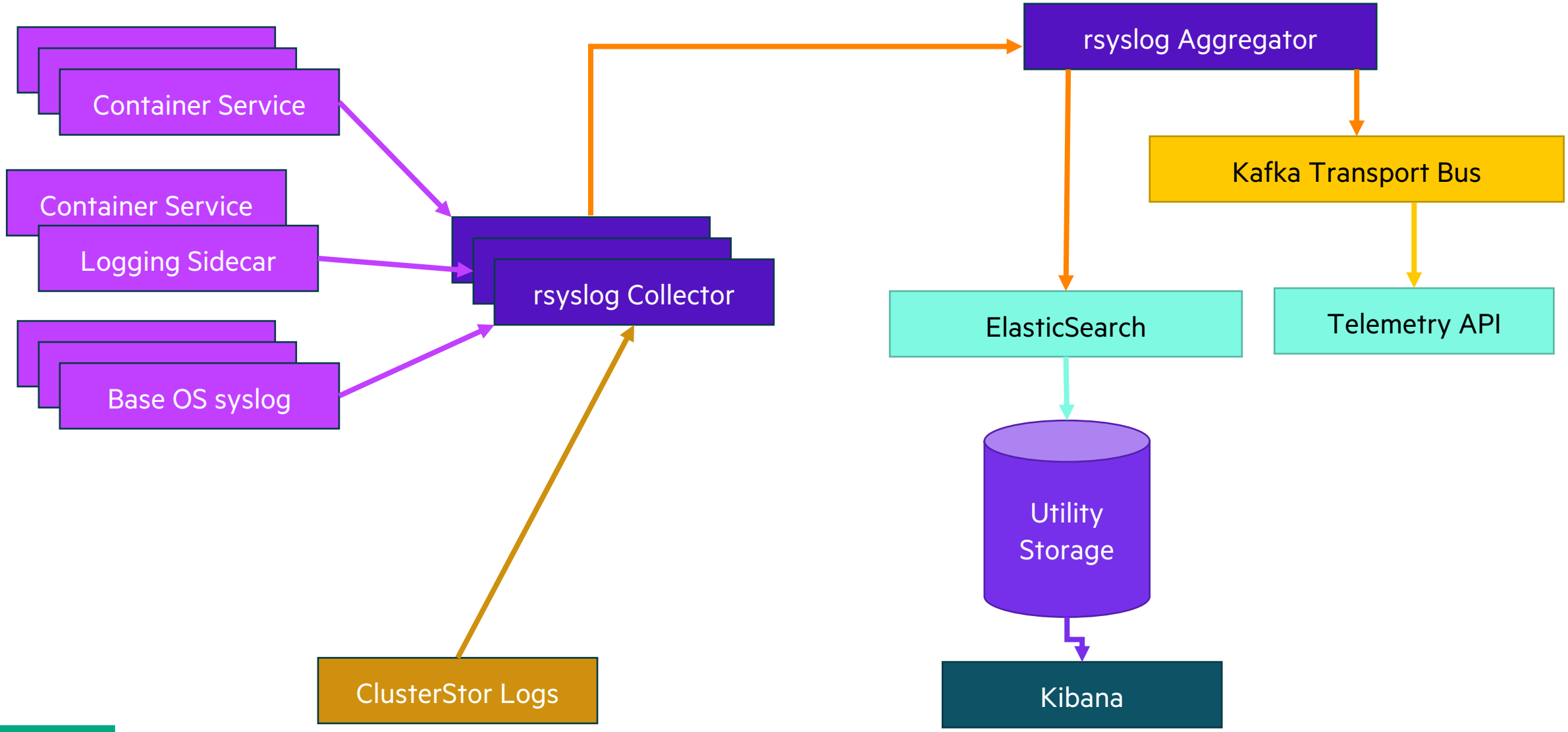


USER INTERFACE AND ACCESS LAYER

- Provides limited end user access to data stored in the SMA
- Allows consumption of streaming data and data that was persisted
  - Creation of custom graphs and panels
  - Generation of custom tables and search dialog boxes
  - Email notification generation for alerts.
- LDMS, IO stat, and vmstat metrics via Grafana
- Log analysis via Kibana
- AuthN and AuthZ provided by API gateway and Keycloak
- Telemetry API used for access to streaming telemetry and data stored in kafka

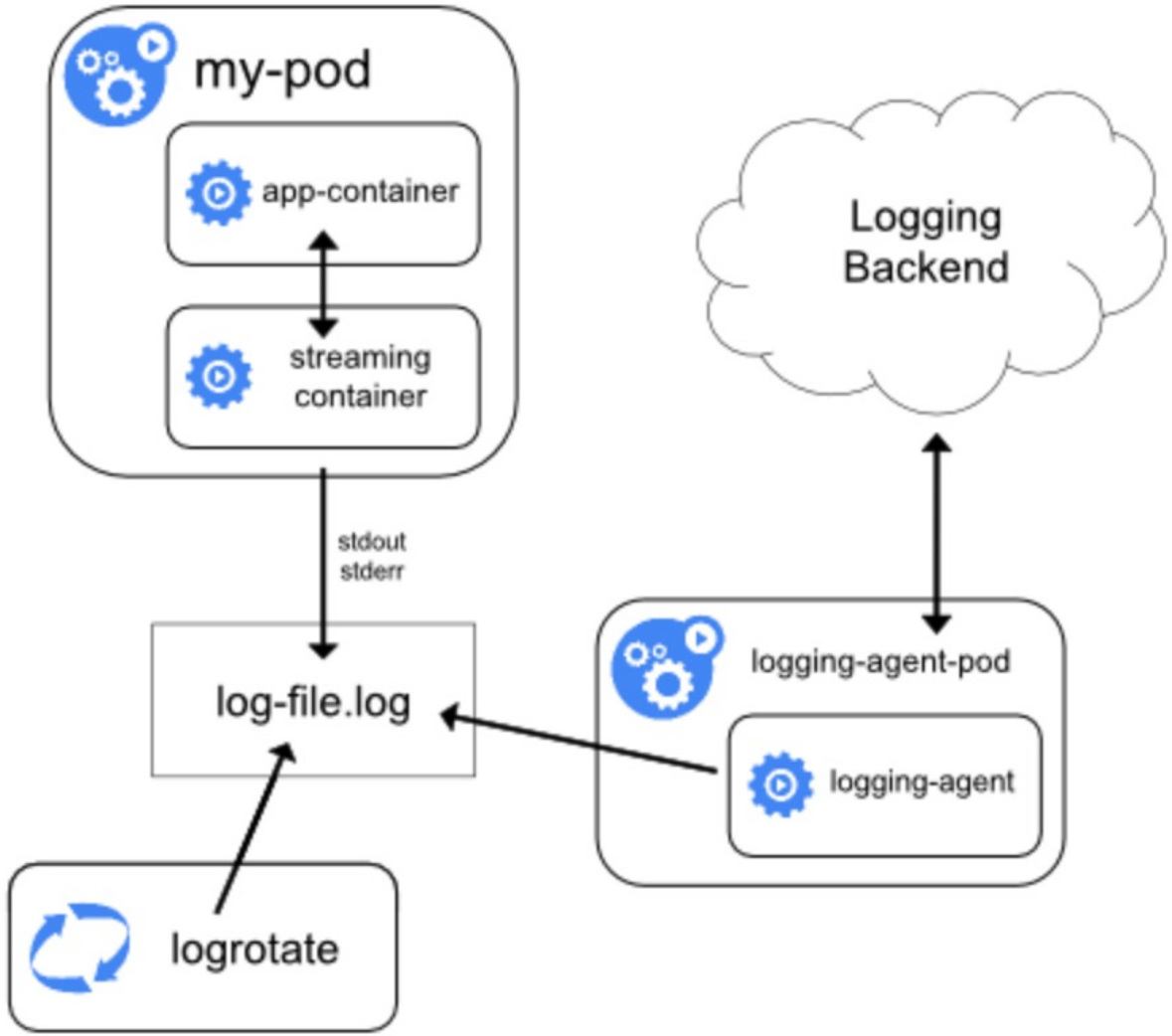


LOG AGGREGATION



KUBERNETES CONTAINER SIDECAR

- Sidecar runs a logging agent
- Picks up logs from application containers in pod
- Can separate several logs streams from different parts of the application



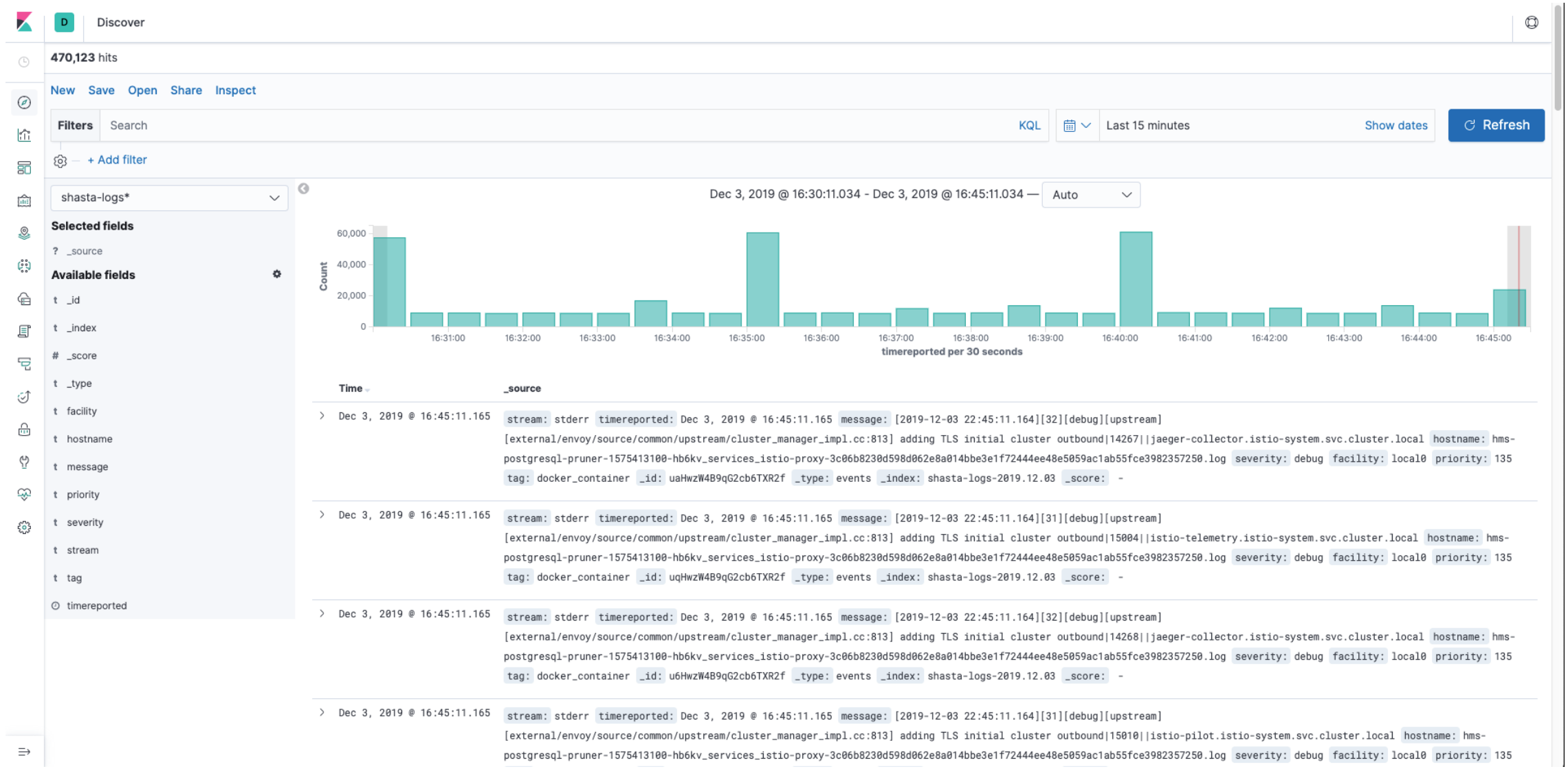
SMA-KIBANA

- Sma-kibana enables
  - Viewing all logs from CNs, NCNs, and Kubernetes pods in Kibana
  - Sorting and searching through log information from multiple sources to help troubleshoot issues
- View and analyze Shasta system logs in the web UI provided by the Kibana service
- Access sma-kibana
  1. Determine the external domain name by running the following command on any NCN:

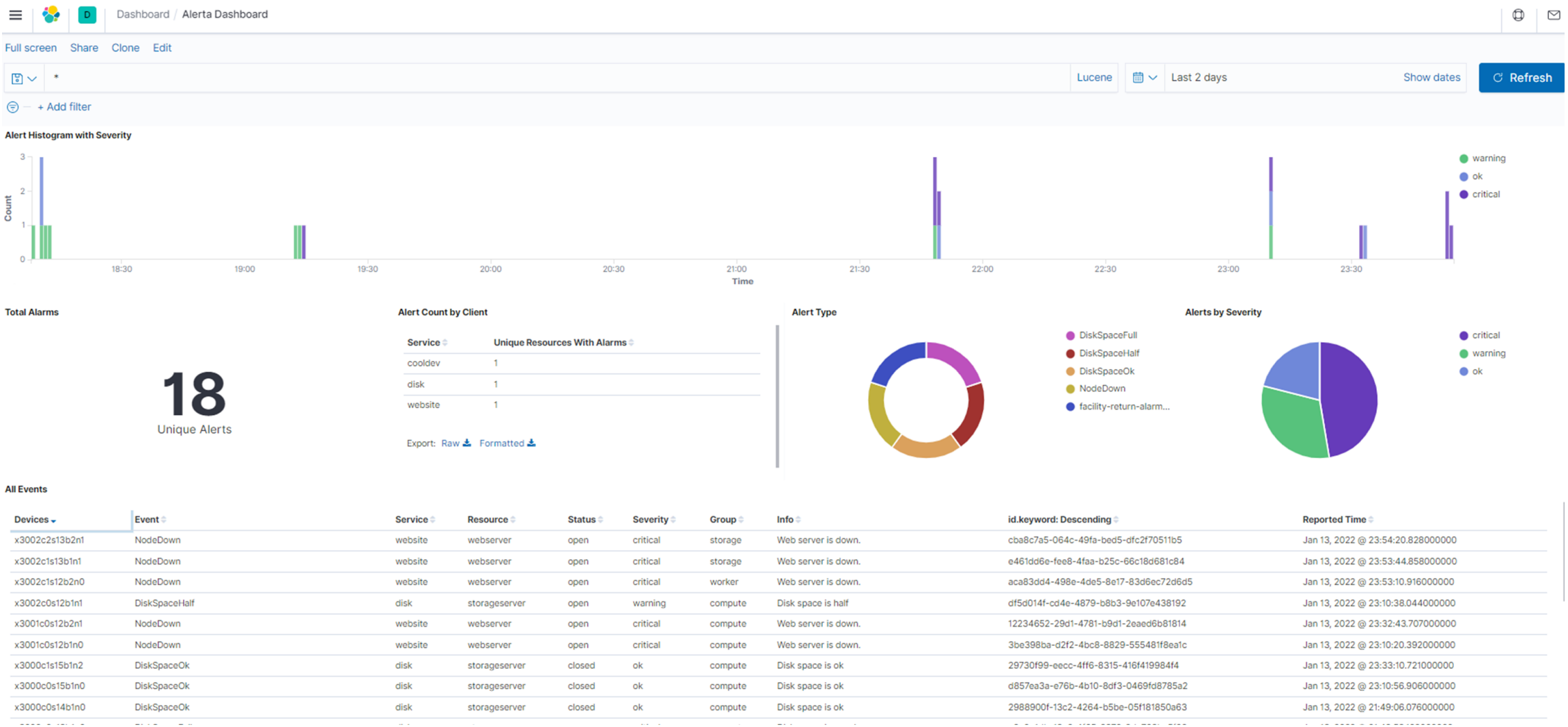
     ```
     ncn-m001# kubectl get secret site-init -n loftsman \
     -o jsonpath='{.data.customizations\.yaml}' | base64 -d | grep "external:"
     external: SYSTEM_DOMAIN_NAME
     ```
  2. Navigate to the following URL in a web browser:

     `https://sma-kibana.cmn.SYSTEM_DOMAIN_NAME/`
  3. Login by entering a valid username and password
  4. Select the index for the type of logs desired (Shasta or ClusterStor) from the drop-down list to search that data source
  5. Refine the displayed results by entering Search terms, which can be simple or complex
  6. Expand displayed log entries for more details
  7. Click a field from the list of Available Fields to see a list of the most common entries in that field
  8. Click the time range drop-down menu to select the time period for which logs are displayed
- https://www.elastic.co/kibana to further explore and analyze the system logs

SMA-KIBANA DISCOVER



SMA-KIBANA ALERTA DASHBOARD



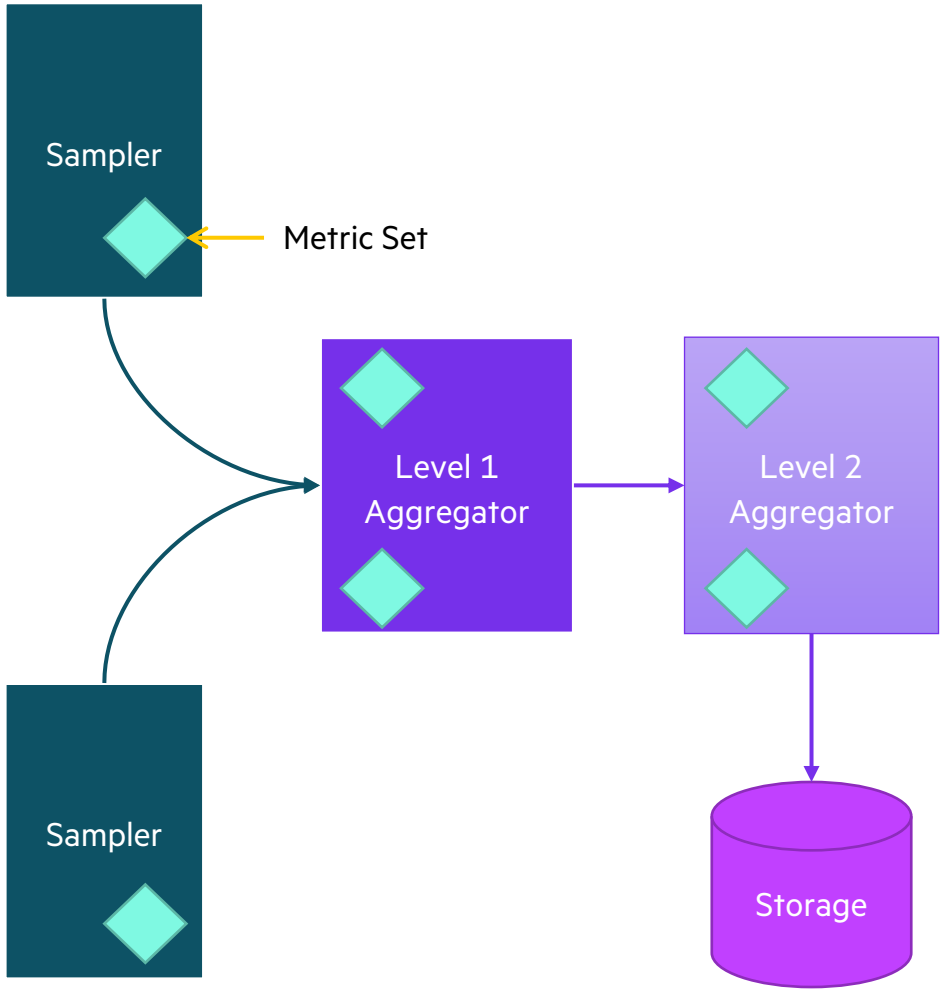
Devices	Event	Service	Resource	Status	Severity	Group	Info	id.keyword: Descending	Reported Time
x3002c2s13b2n1	NodeDown	website	webserver	open	critical	storage	Web server is down.	cba8c7a5-064c-49fa-bed5-dfc2f70511b5	Jan 13, 2022 @ 23:54:20.828000000
x3002c1s13b1n1	NodeDown	website	webserver	open	critical	storage	Web server is down.	e461dd6e-fee8-4faa-b25c-66c18d681c84	Jan 13, 2022 @ 23:53:44.858000000
x3002c1s12b2n0	NodeDown	website	webserver	open	critical	worker	Web server is down.	aca83dd4-498e-4de5-8e17-83d6ec72d6d5	Jan 13, 2022 @ 23:53:10.916000000
x3002c0s12b1n1	DiskSpaceHalf	disk	storageserver	open	warning	compute	Disk space is half	df5d014f-cd4e-4879-b8b3-9e107e438192	Jan 13, 2022 @ 23:10:38.044000000
x3001c0s12b2n1	NodeDown	website	webserver	open	critical	compute	Web server is down.	12234652-29d1-4781-b9d1-2eaed6b81814	Jan 13, 2022 @ 23:32:43.707000000
x3001c0s12b1n0	NodeDown	website	webserver	open	critical	compute	Web server is down.	3be398ba-d2f2-4bc8-8829-555481f8ea1c	Jan 13, 2022 @ 23:10:20.392000000
x3000c1s15b1n2	DiskSpaceOk	disk	storageserver	closed	ok	compute	Disk space is ok	29730f99-eecc-4ff6-8315-416f419984f4	Jan 13, 2022 @ 23:33:10.721000000
x3000c0s15b1n0	DiskSpaceOk	disk	storageserver	closed	ok	compute	Disk space is ok	d857ea3a-e76b-4b10-8df3-0469fd8785a2	Jan 13, 2022 @ 23:10:56.906000000
x3000c0s14b1n0	DiskSpaceOk	disk	storageserver	closed	ok	compute	Disk space is ok	2988900f-13c2-4264-b5be-05f181850a63	Jan 13, 2022 @ 21:49:06.076000000

SAT DASHBOARDS IN SMA-KIBANA

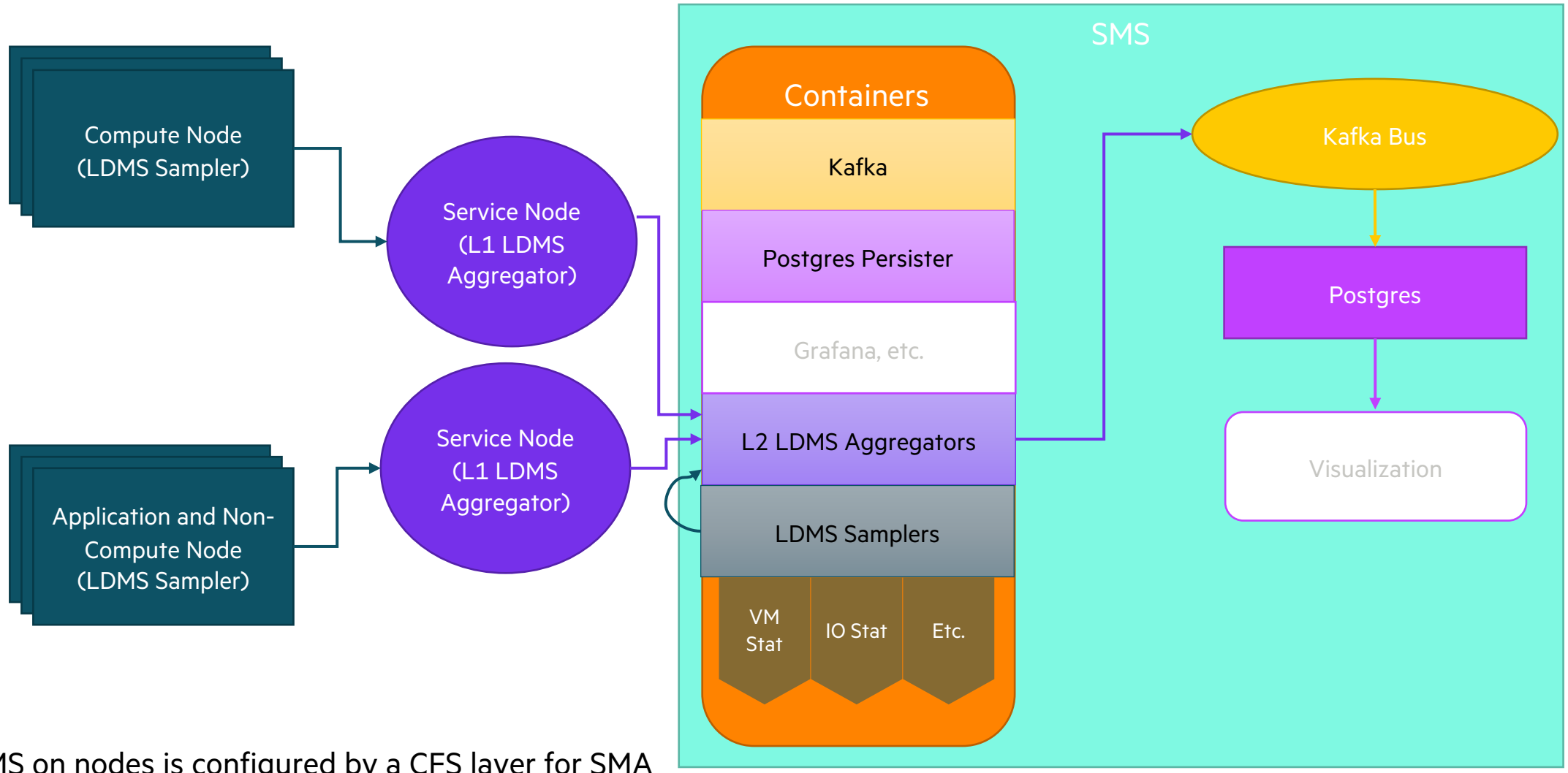
Dashboard	Short Description	Long Description
sat-aer	AER corrected	Corrected Advanced Error Reporting messages from PCI Express devices on each node
sat-aer	AER fatal	Fatal Advanced Error Reporting messages from PCI Express devices on each node
sat-atom	ATOM failures	Application Task Orchestration and Management tests are run on a node when a job finishes. Test failures are logged
sat-atom	ATOM admindown	ATOM test failures can result in nodes being marked admindown. An admindown node is not available for job launch
sat-heartbeat	Heartbeat loss events	Heartbeat loss event messages reported by the hbtd pods that monitor for heartbeats across nodes in the system
sat-kernel	Kernel assertions	The kernel software performs a failed assertion when some condition represents a serious fault. The node goes down
sat-kernel	Kernel panics	The kernel panics when something is seriously wrong. The node goes down
sat-kernel	Lustre bugs (LBUGs)	The Lustre software in the kernel stack performs a failed assertion when some condition related to file system logic represents a serious fault. The node goes down
sat-kernel	CPU stalls	CPU stalls are serous conditions that can reduce node performance, and sometimes cause a node to go down. Technically these are Read-Copy-Update stalls where software in the kernel stack holds onto memory for too long
sat-kernel	Out of memory	An Out Of Memory (OOM) condition has occurred. The kernel must kill a process to continue. The kernel will select an expendable process when possible. If there is no expendable process the node usually goes down in some manner. Even if there are expendable processes the job is likely to be impacted. OOM conditions are best avoided
sat-mce	MCE	Machine Check Exceptions (MCE) are errors detected at the processor level
sat-rasdaemon	rasdaemon errors	Errors from the rasdaemon service on nodes. The rasdaemon service is the Reliability, Availability, and Serviceability Daemon, and it is intended to collect all hardware error events reported by the linux kernel, including PCI and MCE errors
sat-rasdaemon	rasdaemon messages	All messages from the rasdaemon service on nodes

LIGHTWEIGHT DISTRIBUTED METRIC SERVICE (LDMS)

- Developed by Sandia National Lab for Blue Waters Cray XE/XK
- Distributed data collection, transport, and storage tool
- **Samplers** run one or more sampling plugins that periodically sample data on monitored nodes
  - Defines a metric set (a collection of metrics)
  - HA configuration supported
- **Aggregators** periodically collect data in a pull fashion from samplers or other aggregators
- **Storage** plugins periodically write in database or flat file (file per metric name or CSV file per metric set)
  - Incomplete or not updated metric set data is not written to storage



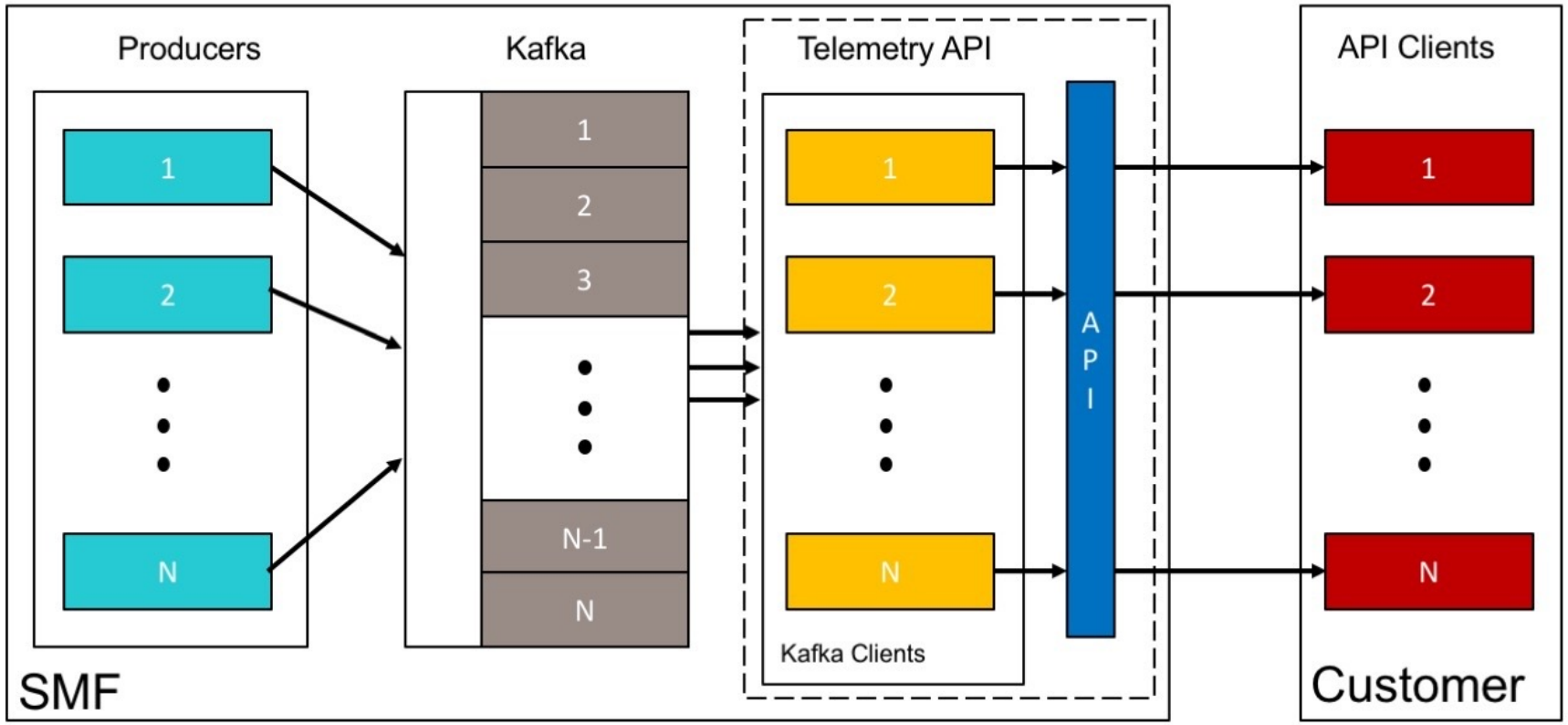
LDMS



- LDMS on nodes is configured by a CFS layer for SMA

TELEMETRY API

Provides access to metrics
Accessible through a RESTful JSON interface
Authenticated using bearer tokens, and token must be included in all HTML requests to the API
Streams telemetry data to clients using Server-Side Events (SSE)



ACCESSING THE TELEMETRY API WITH CURL

```
ncn# CLIENT_SECRET=`kubectl get secrets admin-client-auth -o jsonpath='{.data.client-secret}'| base64 -d`
ncn# TOKEN=$(curl -s -d grant_type=client_credentials -d client_id=admin-client \
-d client_secret=${CLIENT_SECRET} \
https://api-gw-service-nmn.local/keycloak/realms/shasta/protocol/openid-connect/token)
ncn# ACCESS_TOKEN=$(echo ${TOKEN} | jq -r .access_token)
ncn# curl -k -s -H "Authorization: Bearer ${ACCESS_TOKEN}" \
https://api-gw-service-nmn.local/apis/sma-telemetry-api/v1/ping |jq
{
  "api_version": "v1",
  "timestamp": 1591990968
}
ncn# curl -k -s  -H "Authorization: Bearer ${ACCESS_TOKEN}" \
https://api-gw-service-nmn.local/apis/sma-telemetry-api/v1/stream | jq '' |head
{
  "streams": [
    {
      "name": "cray-node",
      "scale_factor": 4
    },
    {
      "name": "cray-logs-clusterstor",
      "scale_factor": 4
    },
```

READING FROM THE TELEMETRY API WITH CURL

```
ncn# curl -ks --compressed -H "Authorization: Bearer ${ACCESS_TOKEN}" \
https://api-gw-service-nmn.local/apis/sma-telemetry-api/v1/stream/cray-node |head -3 \
| tail -1 | fold -80 | head -5
data: { "metrics": { "messages": [{"metric":{"name":"cray_storage.cray_vmstat.me
m_swpd","dimensions":{"product":"shasta","system":"compute","service":"ldms","co
mponent":"cray_vmstat","hostname":"nid001255","cname":"x1000c7s7b1n1","job_id":"
0"},"timestamp":1599767510102,"value":0},"meta":{"tenantId":"6305a7f186e74d849ad
3f00ade0242a9","region":"RegionOne"},"creation_time":3386706919782612992},{"metr
ncn# curl -ks --compressed -H "Authorization: Bearer ${ACCESS_TOKEN}" \
https://api-gw-service-nmn.local/apis/sma-telemetry-api/v1/stream/cray-node |head -3 \
| tail -1 | cut -c 7- | jq '' | head
{
  "metrics": {
    "messages": [
      {
        "metric": {
          "name": "cray_storage.cray_vmstat.mem_cache",
          "dimensions": {
            "product": "shasta",
            "system": "compute",
            "service": "ldms",
```

To get output from telemetry stream the `--compressed` option is needed

Telemetry stream output is one json object per line of output.

Linux formatting tools are helpful

PMDB DATA

MessageId	Topic Name	Requires Collector Setup	Telemetry Source(s)	Storage
CrayTelemetry. Temperature	cray-telemetry-temperature	No	cC, nC, sC, River	Postgres
CrayTelemetry.Voltage	Cray-telemetry-voltage	No	cC, nC, sC, River	Postgres
CrayTelemetry.Power/Current	cray-telemetry-power	No	cC, nC, sC, River	Postgres
CrayTelemetry.Energy	cray-telemetry-energy	No	cC, nC, River	Postgres
CrayTelemetry.Fan/Rotational	cray-telemetry-fan	No	cC, sC, River	Postgres
CrayTelemetry.Pressure	cray-telemetry-pressure	No	cC	Postgres
CrayTelemetry.Humidity	cray-telemetry-humidity	No	cC	Postgres
CrayTelemetry.LiquidFlow	cray-telemetry-liquidflow	No	cC	Postgres
CrayFabricTelemetry.*	cray-fabric-telemetry	Yes	Fabric	Postgres
CrayFabricPerfTelemetry.*	cray-fabric-perf-telemetry	Yes	FabricPerf	Postgres
CrayFabricCriticalTelemetry.*	cray-fabric-crit-telemetry	Yes	FabricCrit	Postgres
Anything except the other rows of this table	cray-dmtf-resource-event	No	Hardware events - No message ID (Different format from all the other metrics)	Postgres
CrayFabricHealth.*	cray-fabric-health	Yes	FabricHealth	Opensearch

PMDB DATA FORMAT


```
{
  "Context": "/",
  "Events": [
    {
      "EventTimestamp": "2023-02-06T10:40:08.543Z",
      "MessageId": "CrayFabricCriticalTelemetry.RoutingErrors",
      "Oem": {
        "Sensors": [
          {
            "Timestamp": "2023-02-06T10:40:08.532Z",
            "Location": ,
            "SensorType": ”Power",
            "ParentalContext": ”SystemBoard",
            "ParentalIndex": 2,
            "PhysicalContext": ”NetworkingDevice",
            "Index": 8,
            "PhysicalSubContext": "Output",
            "DeviceSpecificContext": "ieee",
            "SubIndex": 0,
            "Value": "0"
          }
        ],
        "TelemetrySource": "FabricCrit"
      }
    }
  ]
}
```


SMA-GRAFANA

- The HPE Cray EX system uses a Grafana web UI to provide system metric monitoring of:
  - LDMS metrics
  - Job and Lustre performance metrics for any attached and monitored ClusterStor storage systems
  - HSN fabric performance, errors, congestion, and other statistics
  - Power, temperature and other PMDB sensor data from node, cabinet, and switch controllers
- Access sma-grafana
  1. Determine the external domain name by running the following command on any NCN:

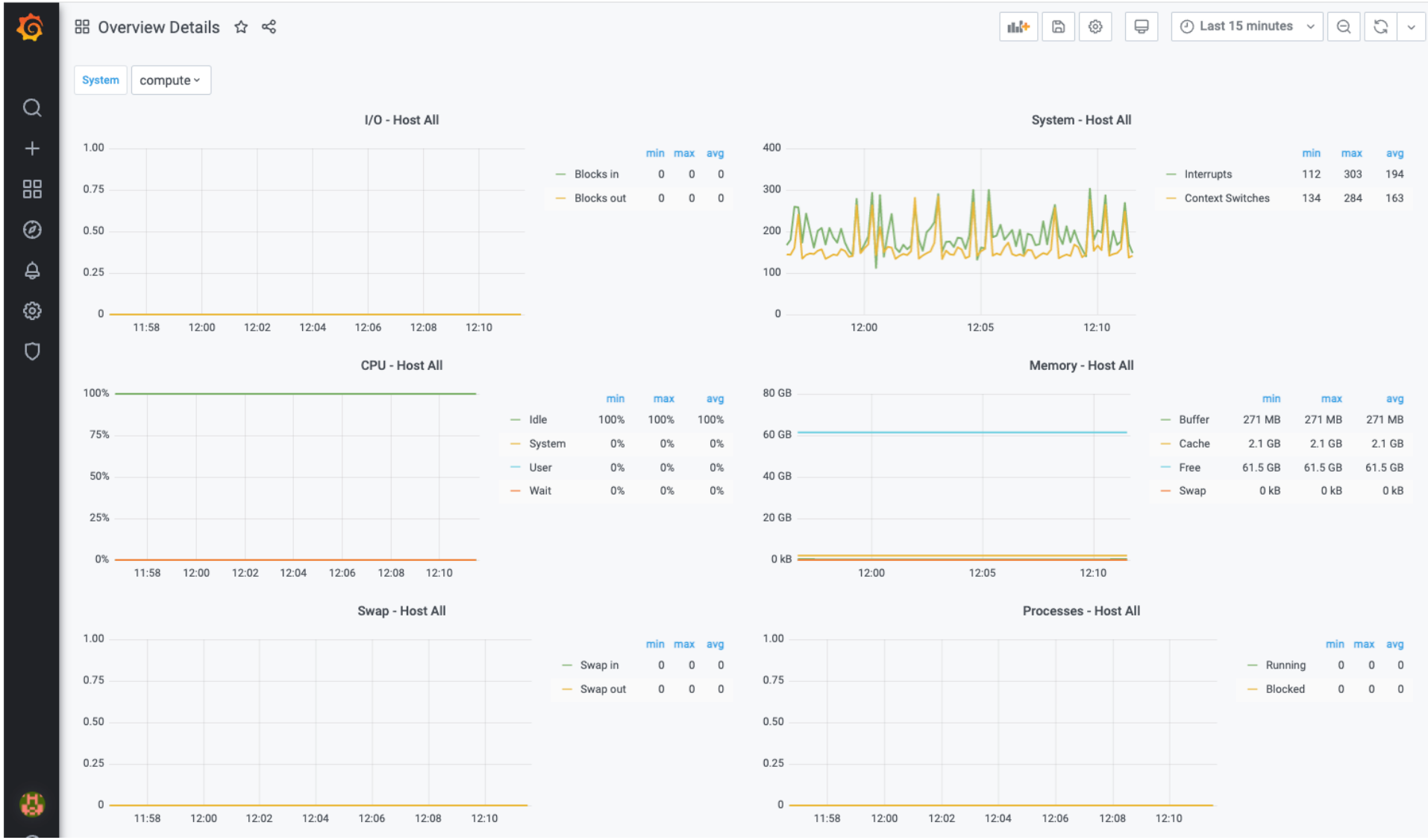
     ```
     ncn-m001# kubectl get secret site-init -n loftsman \
     -o jsonpath='{.data.customizations\.yaml}' | base64 -d | grep "external:"
     external: SYSTEM_DOMAIN_NAME
     ```

  2. Navigate to the following URL in a web browser:

     `https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/`

  3. Login by entering a valid username and password
  4. Select a dashboard from the Overview Details drop-down menu

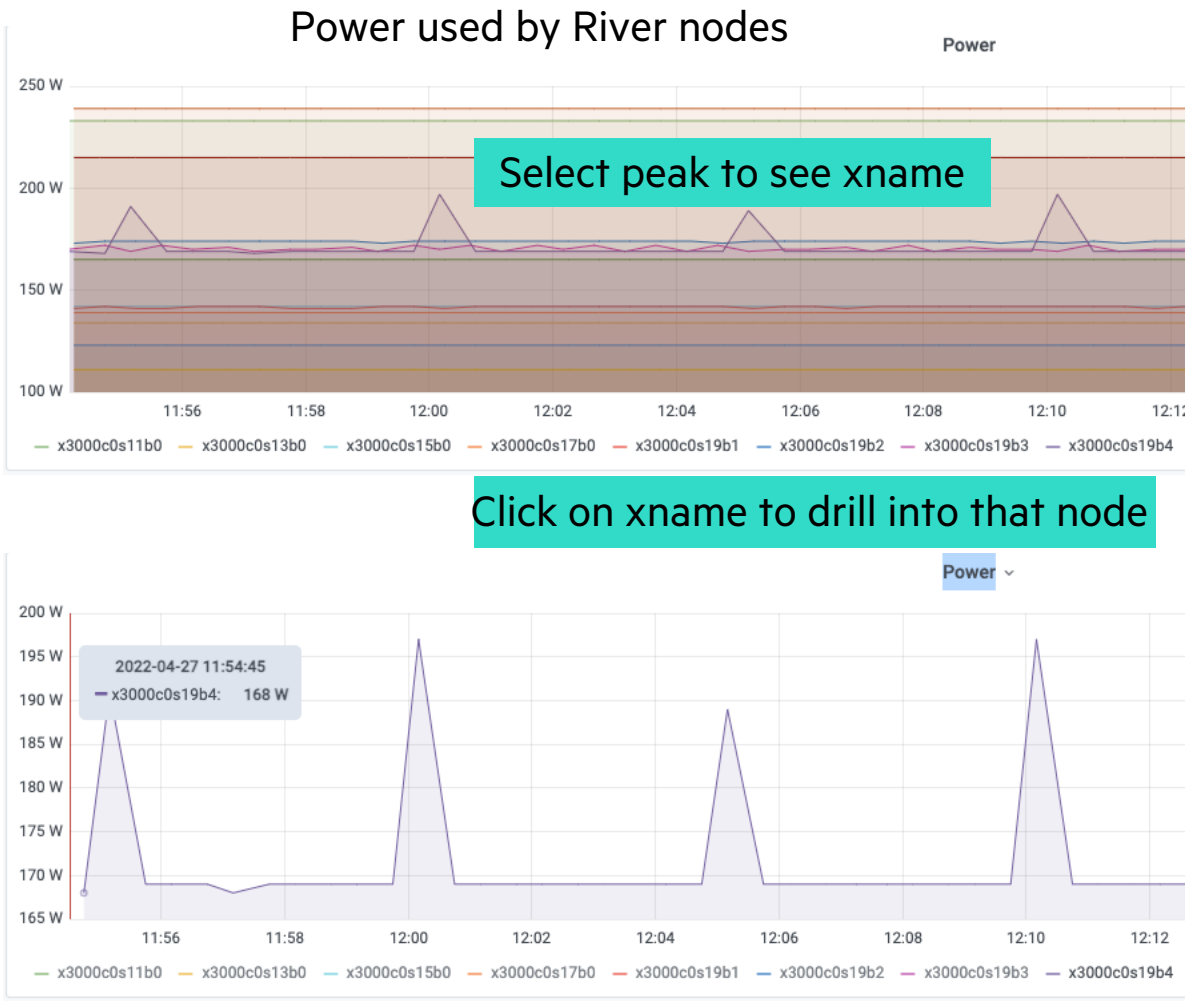
SMA-GRAFANA OVERVIEW DETAILS



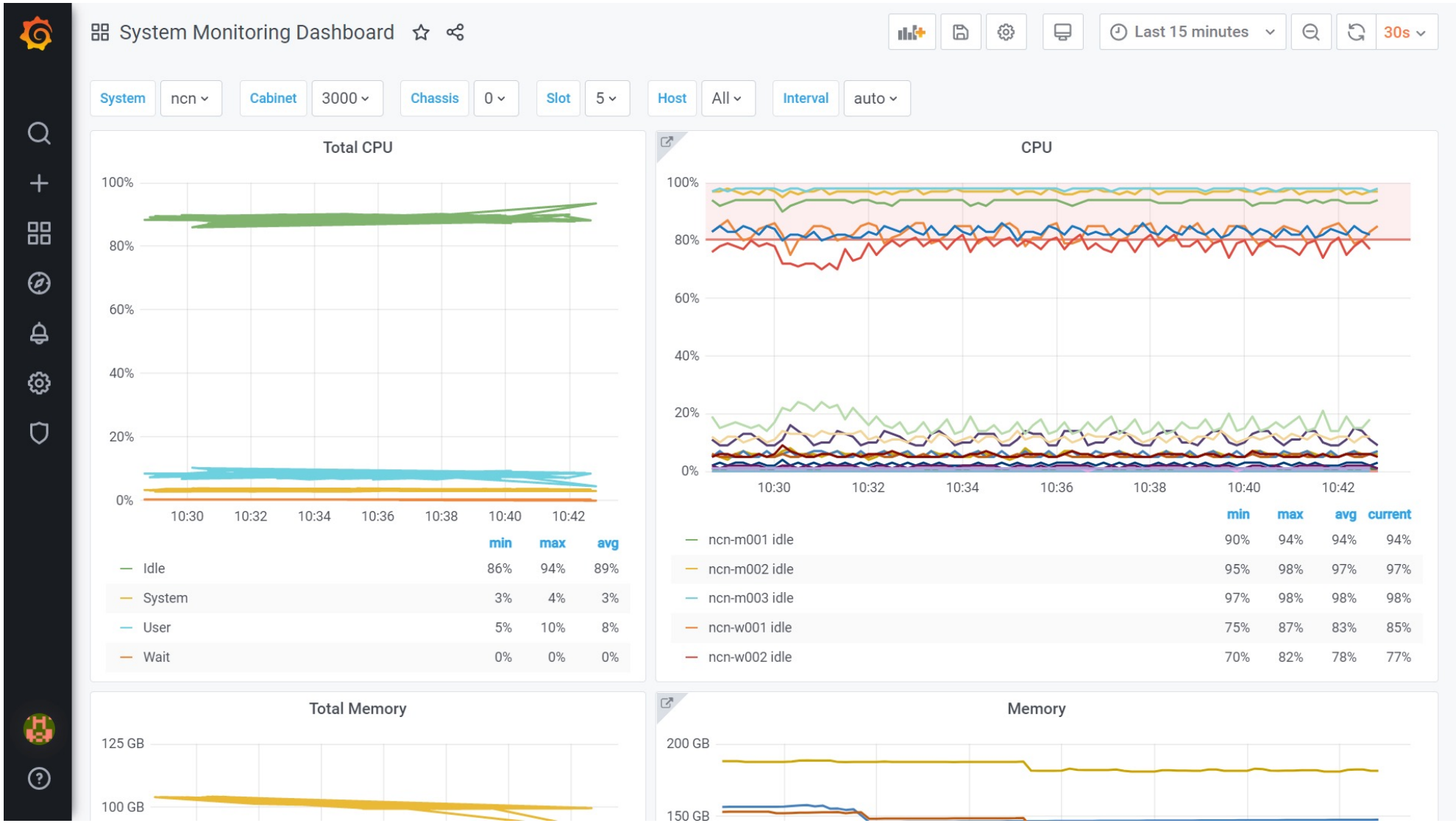
SMA-GRAFANA DASHBOARDS

- About 20 included dashboards
  - System CPU, I/O, Kernel, Memory, Processes, Swap
  - Cabinet Controller Sensors
  - CDU Monitoring
  - Fabric Telemetry
  - Fabric Performance Telemetry
  - Fabric Critical Telemetry
  - Fabric Switch Hardware Telemetry
  - Node Controller Sensors
  - Overview Details
  - Overview Device I/O Stats
  - PDU Monitoring
  - Redfish Events
  - River Sensors
  - Switch Controller Sensors
  - System Monitoring Dashboard
  - Cluster Health Check (Alerta alerts)

https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards



SMA-GRAFANA SYSTEM MONITORING DASHBOARD

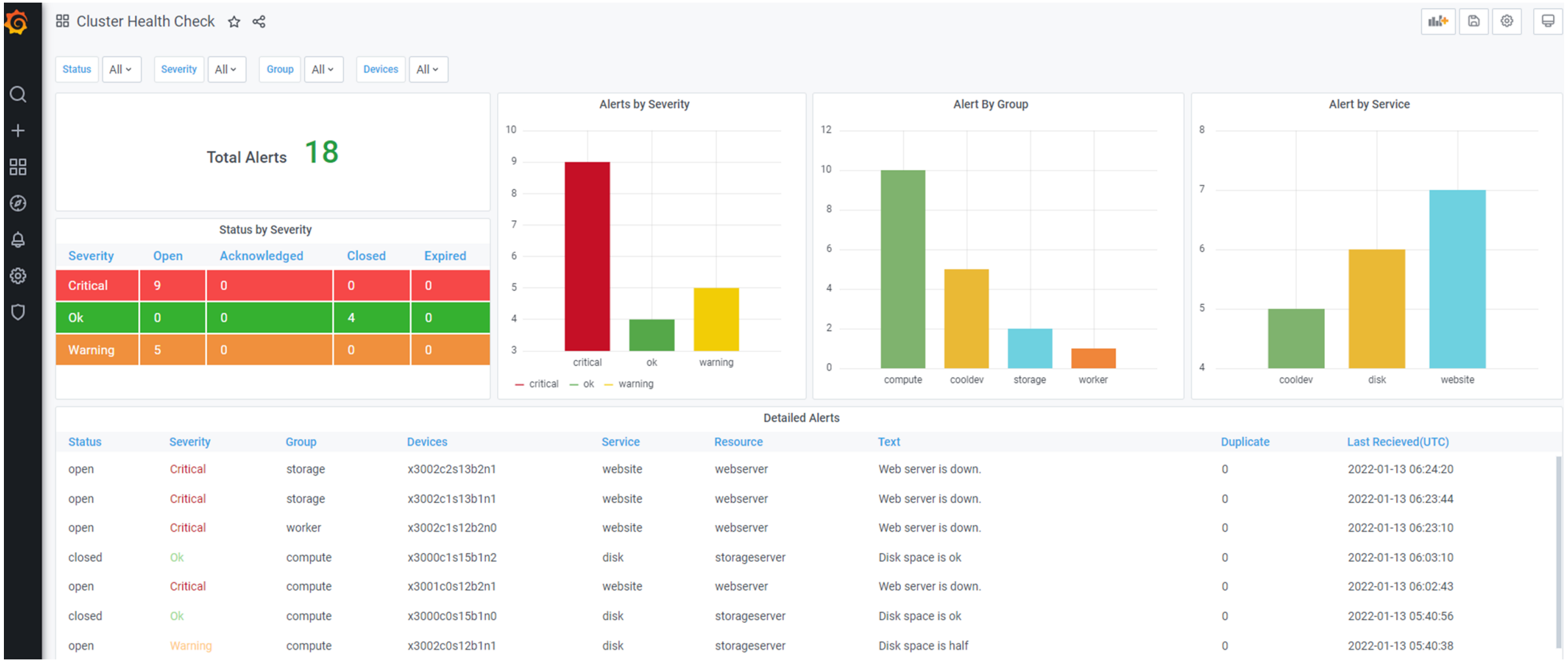


SMA-GRAFANA SWITCH CONTROLLER SENSORS

SMA-GRAFANA CABINET CONTROLLER SENSORS

SMA-GRAFANA NODE CONTROLLER SENSORS

CLUSTER HEALTH CHECK



MONASCA

The Monasca alarms and notifications subsystem of SMA monitors metric data on the kafka telemetry bus and provides a way to notify users via email when select metric data is outside of normal operating values.

The alarms rules engine subscribes to a metric topic in kafka and compares values against thresholds.

Metric data is processed one reading at a time, but alarm transitions can be set to be triggered for averages over time periods which can be defined within the alarm.

When the alarm-state changes, a state-transition is sent to notification engine.

CONFIGURING MONASCA

Alarm and notification definitions are created in a configmap that should be instantiated in the SMA namespace.

It is likely that administrators will want to create custom alarm definitions so alarm notifications are sent when meaningful thresholds are crossed.

In order to define new alarms, create and save a text file named customer-alarms-configmap.yaml with contents like those shown, and initialize it as described in the HPE Cray EX System Monitoring Application Administration Guide.pdf.

```
# Source: sma-monasca/templates/alarms-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: customer-alarms
  labels:
    app: customer-sma-monasca
    component: "alarms"
data:
  definitions.yml.j2: |
   notifications:
     - name: customerEmail
       type: email
       address: "user@customer.com"
   alarm_definitions:
     - name: "customerTestAlarm"
       expression: "avg(cray_test.heartbeat) < 20"
       description: "Customer Alarm test metrics topic"
       severity: "MEDIUM"
       match_by:
         - "hostname"
       alarm_actions:
         - "customerEmail"
       ok_actions:
         - "customerEmail"
       undetermined_actions:
         - "customerEmail"
```

MONASCA CLI

System administrators can use the Monasca alarm-list , alarm-definition-list , and notification-list commands to view the state of defined alarms, the definitions of all SMA alarms on a HPE Cray EX system, and the list of all defined SMA notifications, respectively. These commands, however, all have to be issued through Kubectl since SMA runs Monasca in Kubernetes pods.

List defined alarms:

```
# kubectl -n sma exec -it sma-monasca-agent-p9vcb -c collector -- sh -c 'monasca alarm-definition-list'
```

```
+------------------------------+--------------------------------------+-----------------------------------------------------------+-------------+-----------------+
| name                         | id                                   | expression                                                | match_by    | actions_enabled |
+------------------------------+--------------------------------------+-----------------------------------------------------------+-------------+-----------------+
| lustreTestAlarm              | 28c5d704-deb9-4a99-a6b0-e74a42ad34e6 | avg(cray_test.lustre_test) < 20                           | hostname    | True            |
| SMA Alarm Test 1             | 3e00a3c6-655f-410e-8bf9-137b9edde23d | last(cray_test.other_test) > 20                           | hostname    | True            |
| SMA ClusterStor Metric Health | 6b046908-a8b7-486a-8fb2-1dfbecab9743 | min(cray_storage.link_rate{device=MDT0000}) < 0 times 5   | system_name | True            |
| vmstatTestAlarm              | 9b05bef3-f732-4617-a79f-9bc8a164b92b | avg(cray_test.vmstat_test) < 20                           | hostname    | True            |
| SMA OST Free Files           | 9b81138f-3da6-4253-a3ad-ac78e2c4ae6f | avg(cray_storage.free_files_perc, 900) < 5.0              | system_name | True            |
|                              |                                      |                                                           | device      |                 |
| SMA OST Free Space           | c4d5d9e2-dbec-447d-9695-069aed4960f4 | avg(cray_storage.free_space_perc, 900) < 5.0              | system_name | True            |
|                              |                                      |                                                           | device      |                 |
| validation1Alarm             | de216b43-52bf-4f35-a86f-1dc9b3655341 | last(kubelet.health_status) < 0                           | hostname    | True            |
| metricsTestAlarm             | f424ec27-291e-4b60-bba5-b020a510ae77 | avg(cray_test.other_test) < 20                            | hostname    | True            |
+------------------------------+--------------------------------------+-----------------------------------------------------------+-------------+-----------------+
```

List active alarms and their state:

```
# kubectl -n sma exec -it sma-monasca-agent-p9vcb -c collector -- sh -c 'monasca alarm-list'
```

MON_ALERT

Apache Kafka and Elasticsearch process alert information from Slingshot and make it available to the `mon-alert` command.

The `mon-alert` command does the following:

- Looks for events in the data
- Analyzes each event
- Alerts the user regarding the event
- Stores the event in the alert dashboard
- Allows you to manage the life cycle of Slingshot alerts received by the cluster manager.

For an overview of the mon-alert command, enter the following:

```
ncn-w001# mon-alert -h
```

MON_ALERT EXAMPLES

Display a summary of all alerts:

```
ncn-w001# mon-alert -s
Alert Status        Count
------------        -----
Critical            0
Warnings            6288
Information         1
Open                6288
Acknowledged        0
Closed              2
Expired             0
```

Display the information for a specific alert from the node(environment).

```
ncn-w001# mon-alert query -i 1d0d93c1
ID          1d0d93c1
STATUS      open
SEVERITY    warning
GROUP       fabric
ENV         http://10.33.0.170:8000/fabric/agents/x3000c0r21b0
SERVICE     slingshot
RESOURCE    http://10.33.0.170:8000/fabric/agents/x3000c0r21b0
EVENT       crayfabrichealthtelemetry-configuration Telemetry error- Telemetry failure:
            Service https://x3000c0r21b0/fabric/v1/switch/telemetry returned error 400
            for PUT. Id 72954577 Message Unsupported Statistics value ["35535"] -critical
VALUE       Telemetry
DESCRIPTION Telemetry failure: Service https://x3000c0r21b0/fabric/v1/switch/telemetry
            returned error 400 for PUT. id 72954577 message Unsupported statistics value
            ["3635","35535"]
DUPL        0
LAST RECEIVED 2021/12/16 14:08:11
```

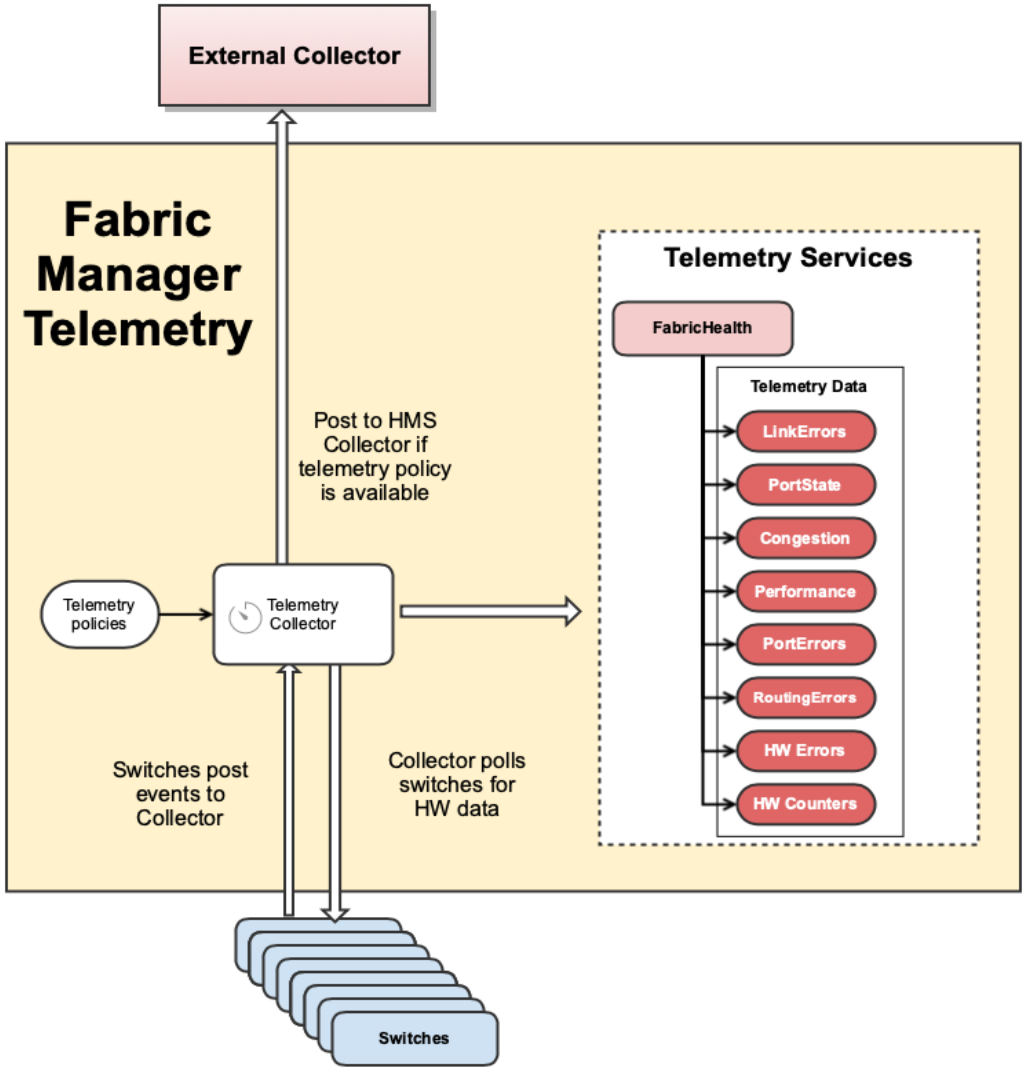
Display the most serious alerts:

```
ncn-w001# mon-alert top
http://localhost:8080          alerta 8.6.0                    14:16:16 16/12/21
Sev  Time      Dupl. Customer  Env.                  Service Resource    Group      Event      Value      Text
Warn 14:08:35 0        -           x3000c0s12b1n1 disk    rsyslog      compu      SpaceHal   ERROR      Disk space is half
Warn 14:07:12 0        -           x3000c0s13b1n0 disk    rsyslog      compu      SpaceFul   ERROR      Disk space is half
Warn 14:07:10 0        -           x3000c0s14b1n0 disk    rsyslog      compu      SpaceOk    OK         Disk space is ok
```

This command produces many lines of output. To return to the system prompt, press CTRL-c.

SLINGSHOT MONITORING

- The Fabric Manager collects event and metric data from the Slingshot switch and generates fabric health events.
- While telemetry data can be accessed using the Fabric Telemetry API, the Fabric Manager can also be configured to stream telemetry data to an external collector such as the Hardware Management Service (HMS) in HPE Cray EX system software which routes the telemetry data to the Shasta Monitoring Application (SMA).
- The Fabric Manager exposes telemetry data via the REST API endpoints. All telemetry data is cached in the fabric manager and can be queried via odata queries.
- When Fabric Manager is configured as telemetry collector, the Slingshot switches push critical and non-critical telemetry data to persistent storage within the Fabric Manager.



SLINGSHOT MONITORING CONFIGURATION

- Configure the collector URI

```
ncn# fmctl update /switch-telemetry/settings collector.value=http://10.94.100.71:80
ncn# fmctl update /switch-telemetry/settings collector.value=http://istio-ingressgateway.hmnlb:80
```

- Set the periodicity for data collection

```
ncn# fmctl update /switch-telemetry/settings periodicity.value=10 (0-300)
```

- Set up telemetry collection – Sets collector URI

```
ncn# fmn-update-telemetry-config --collector https://api.hmnlb.shandy/apis/fabric-manager/telemetry-collector:80 –enable
```

- Enable optional telemetry

```
ncn# fmctl update /switch-telemetry/settings statistics.values=["3635", "1213", "2819", "2863", "4188", "PortErrors",
"HardErrors", "RoutingErrors"]
```

- Enable zero-valued metric streaming (Disabled by default)

```
ncn# fmctl get switches
ncn# for sw in <switch-names>; do echo $sw; ssh $sw fmctl update /switch-telemetry/metrics-collection
zeroValuedMetricsStreamingEnabled=true/false; done
```

SLINGSHOT DATA

Fabric Telemetry

- PortState Status, PortState Speed, LinkErrors
- Slingshot 2.0 no longer supports these metrics

Fabric Hardware Telemetry

- Current, power, temperature, voltage, etc.
- Available data varies with hardware.

Fabric Critical Telemetry

- Telemetry data generated by the switch agents associated with critical port errors.
- By default, the counters associated with port errors are not streamed.
  - Enabled in the “enable optional telemetry” step on the previous slide.

Fabric Performance Telemetry

- Switch agents generate performance telemetry per port utilization/congestion.
- Congestion metrics are collected and posted based on a configurable periodic timer.
  - Configured in the “set the periodicity for data collection” step on the previous slide.

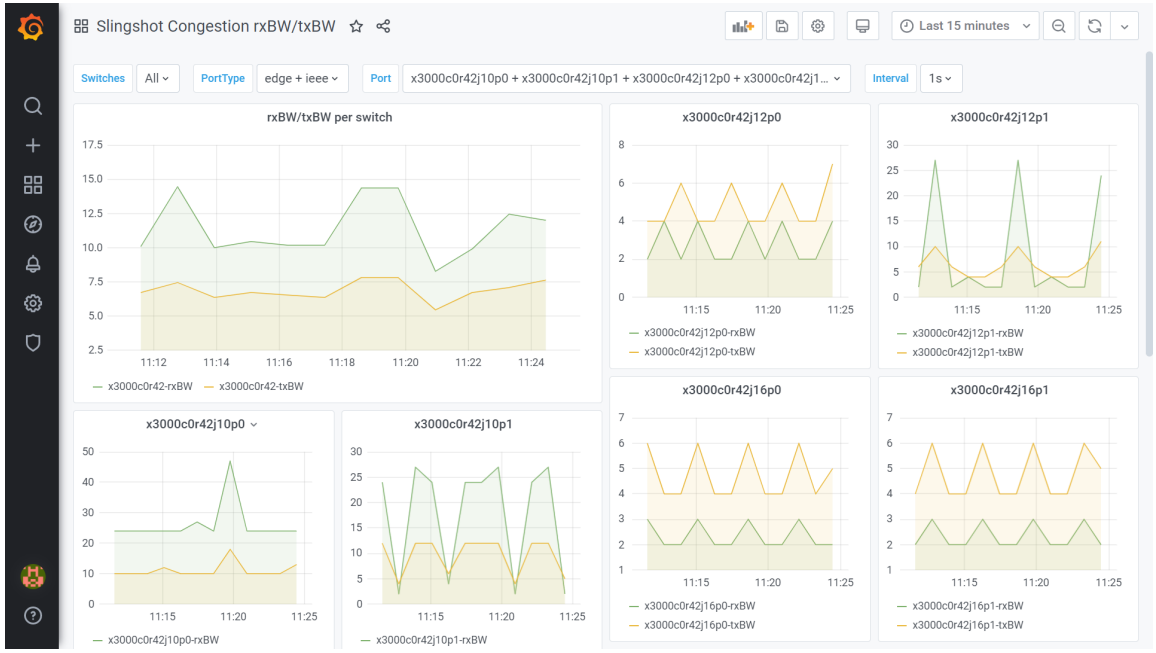
SLINGSHOT DASHBOARDS

- Included Slingshot Dashboards:
  - Bit Error Rate (BER)
  - Congestion rx/tx BW (Receive/Transmit Bandwidth)
  - Congestion rxCongestion
  - Congestion rxPausePercent/txPausePercent
  - Current
  - HardErrors
  - LinkErrors
  - PortSpeed
  - PortStatus
  - Power
  - RFC3635 IfHcInOctets/IfHcOutOctets
  - RFC3635 RxBroadcastPkts/TxBroadcastPkts
  - RFC3635 RxMulticastPkts/TxMulticastPkts
  - RFC3635 RxPauseFrames/TxPauseFrames
  - RFC3635 RxUcastPkts/TxUcastPkts
  - Rotational
  - RoutingErrors
  - Temperature
  - Voltage

https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards

Use the Slingshot monitoring dashboards for monitoring, real-time analytics, alerting, and visualization of Slingshot data. This includes fabric, fabric hardware, fabric critical, and fabric performance Telemetry data.

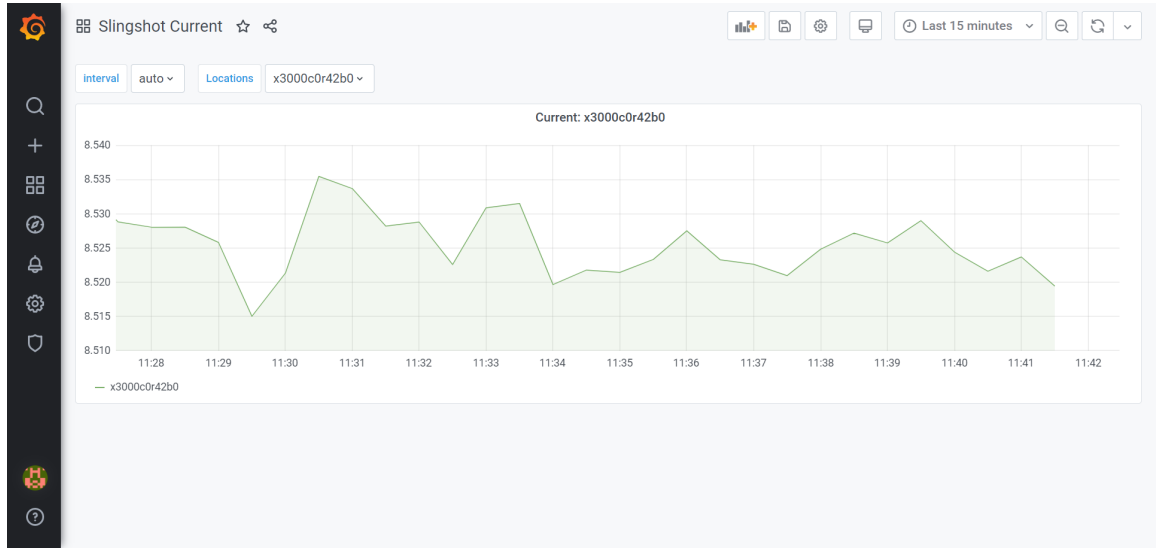
Dashboards provide aggregated metrics for all switches and also offer filtration options at the port and port type levels.

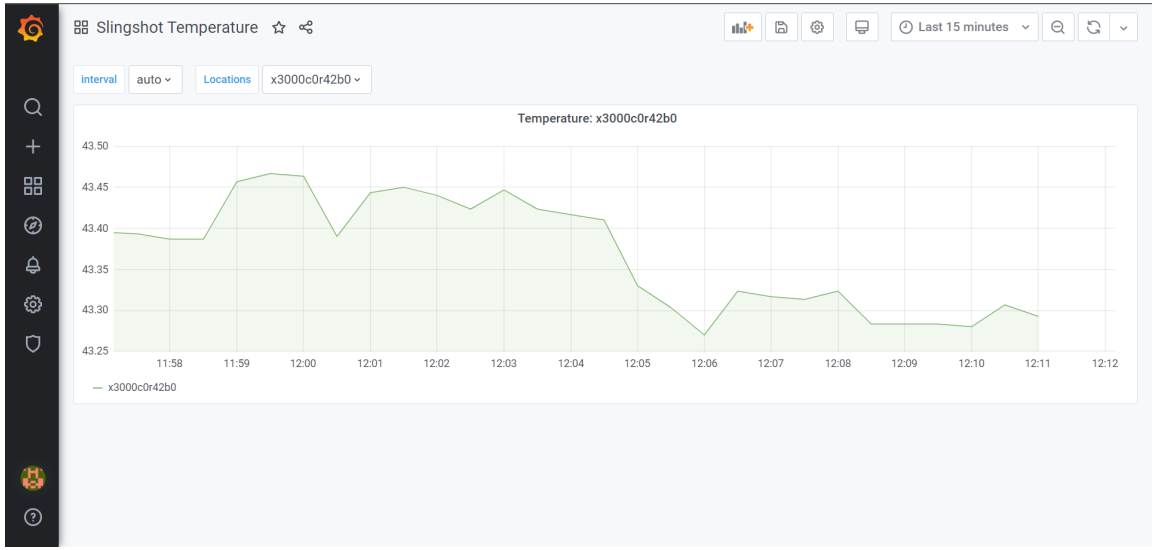


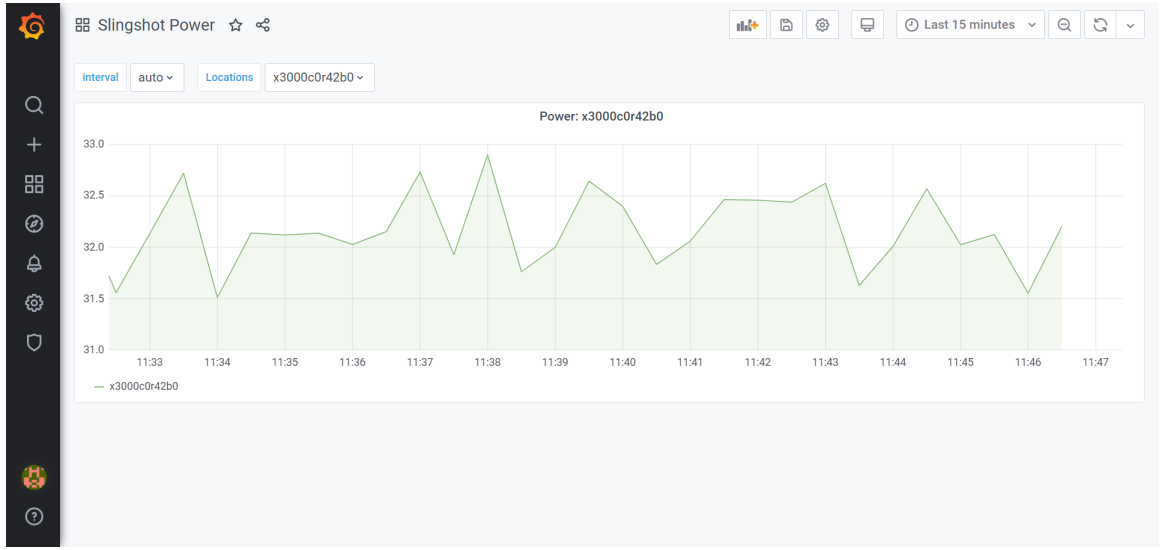
SLINGSHOT HARDWARE TELEMETRY DASHBOARDS

- CraySwitchHardwareTelemetry dashboards include:
  - Current
  - Energy
  - Power
  - Rotational
  - Temperature
  - Voltage

https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards







SLINGSHOT FABRIC AND PERFORMANCE TELEMETRY DASHBOARDS

CrayFabricTelemetry dashboards include:

- PortState Status
- PortState Speed
- LinkErrors

NOTE: Slingshot 2.0 no longer supports these metrics

CrayFabricPerformanceTelemetry dashboards include:

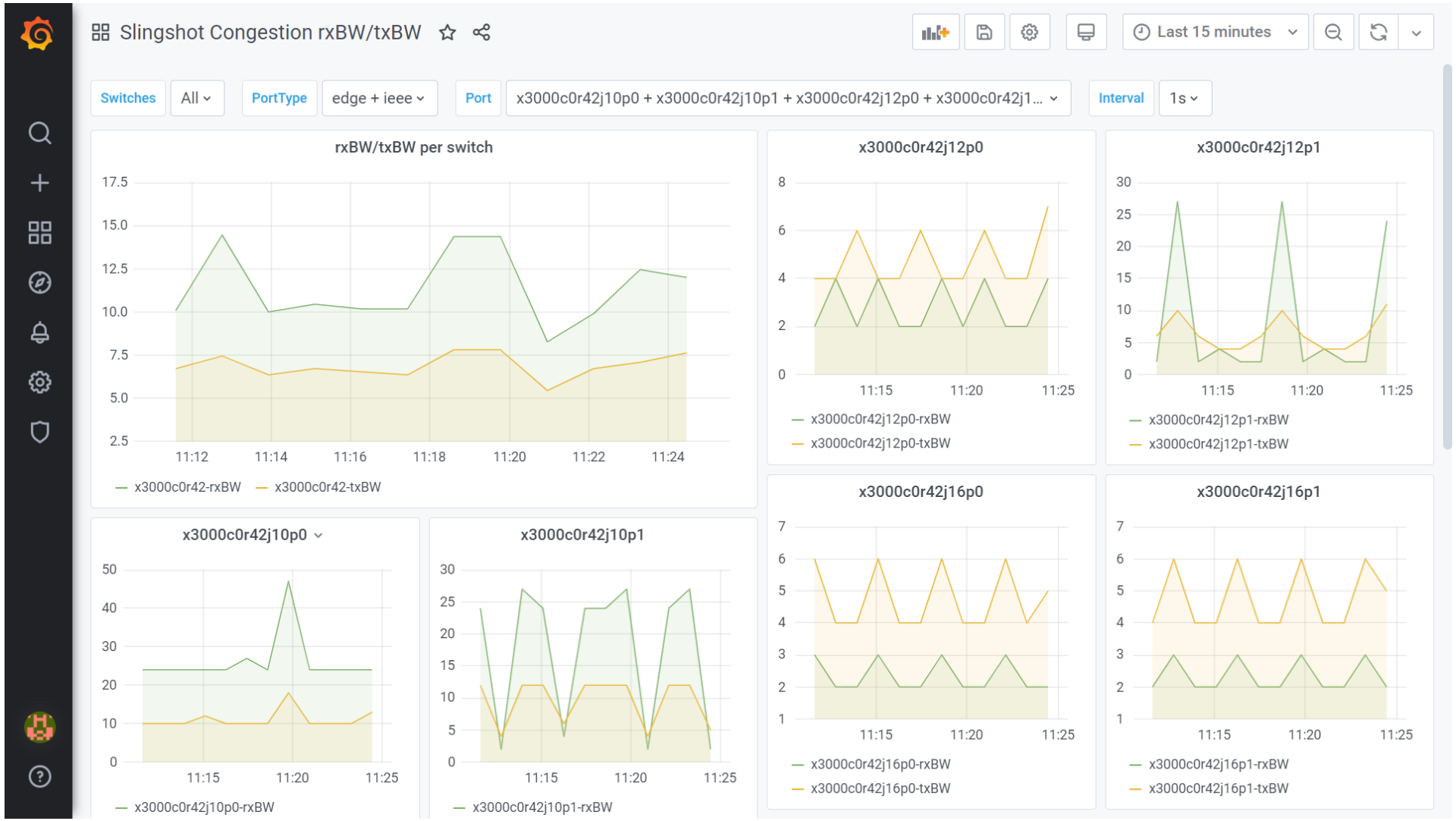
- Congestion rx/tx BW (Receive/Transmit Bandwidth)
- Congestion Receive/Transmit PausePercent
- RFC3635 rx/tx UcastPkts (Receive/Transmit Unicast Packets)
- RFC3635 rx/tx MulticastPkts (Receive/Transmit Multicast Packets)
- RFC3635 rx/tx BroadcastPkts (Receive/Transmit Broadcast Packets)
- RFC3635 rx/tx PauseFrames (Receive/Transmit Pause Frames)
- RFC3635 Receive/Transmit IfHCOutOctets (The average number of octets)

https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards

SLINGSHOT CONGESTION RX/TX BW (RECEIVE/TRANSMIT BANDWIDTH)

This dashboard shows the average receive bandwidth and transmit bandwidth values of a selection of switches and ports over a specified period of time.

Multiple graph panels are used to display switch-level and port-level averages.



SLINGSHOT FABRIC CRITICAL TELEMETRY DASHBOARDS

Cray Fabric Critical Telemetry is telemetry data generated by the switch agents associated with critical port errors.

CraySwitchHardwareTelemetry dashboards include:

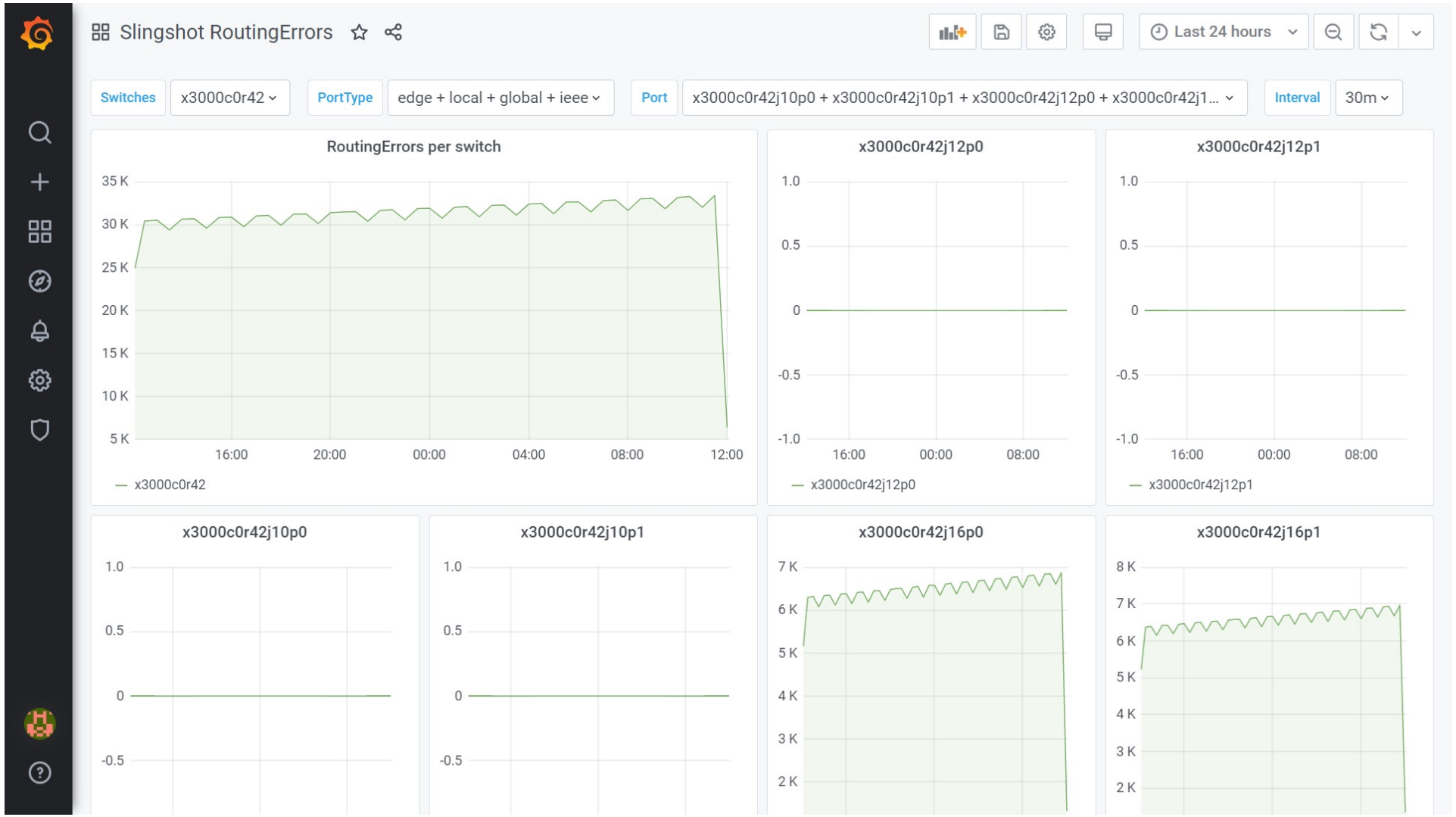
- PortErrors
- RoutingErrors
- HardErrors

https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards

SLINGSHOT ROUTING ERRORS DASHBOARD

This dashboard shows the number of RoutingErrors that have occurred in each switch over a specified period of time.

Multiple graph panels are used to display port-level RoutingErrors statistics.



THANK YOU

matthew.silvia@hpe.com