Hewlett Packard
Enterprise

ADVANCED TOPICS FOR CRAY SYSTEM MANAGEMENT FOR HPE CRAY EX SYSTEMS

Harold Longley

CUG 2023

AGENDA

- HPE CRAY EX SYSTEM OVERVIEW
- ANSIBLE BEST PRACTICES
- MONITORING TOOLS
- SYSTEM MANAGEMENT HEALTH
- TUNING COMPUTE NODES
- SYSTEM ADMIN TOOLKIT
- TROUBLESHOOTING BOOT FAILURES
- COLLECTING DATA FOR HPE SERVICE

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

HPE CRAY EX SYSTEM OVERVIEW

- CSM Architecture
- HPE Cray EX Hardware
- Networks

CSM ARCHITECTURE

HPE CRAY SYSTEM MANAGEMENT FOR EXASCALE SUPERCOMPUTERS

Manage and extend Exascale supercomputer system management capabilities

Resilient, elastic, scalable systems management solution designed using extensible microservices cloud stack

Powerful
Comprehensive set of tools you need to manage all aspects of your Cray EX Supercomputer

Productive
Designed to maximize productivity of your HPC system, automate actions, and optimize running costs

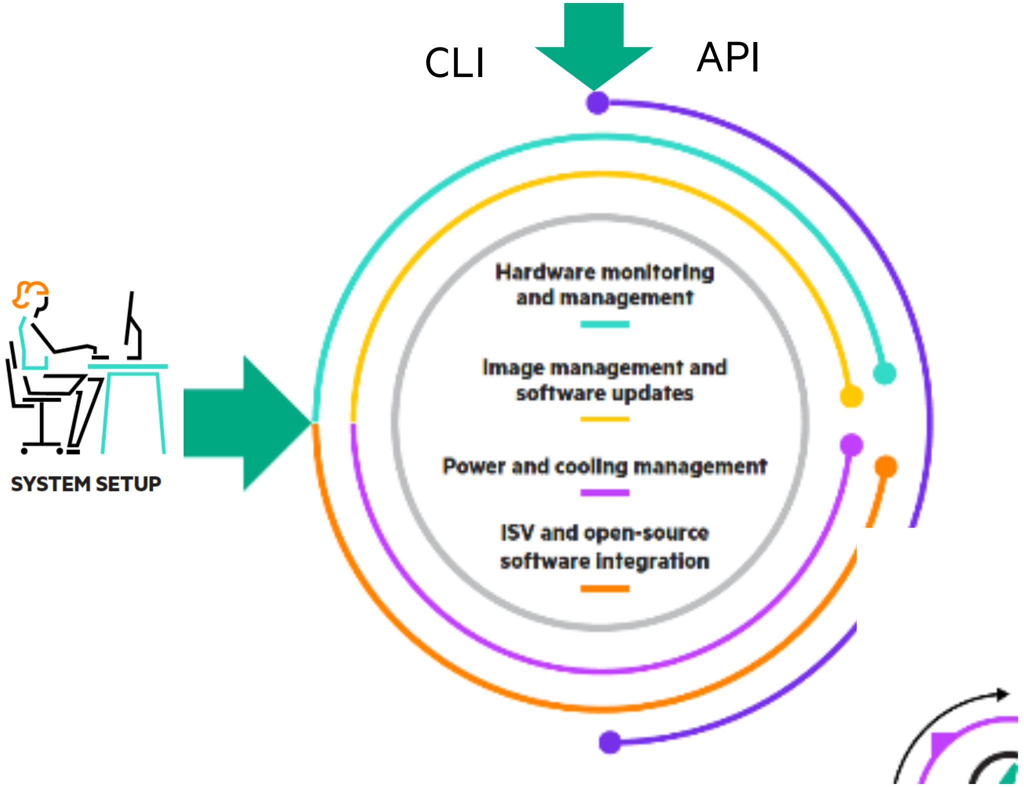
Secure
Support customizable role-based access control for systems management administration

Scalable
Manage Exascale systems with thousands of nodes

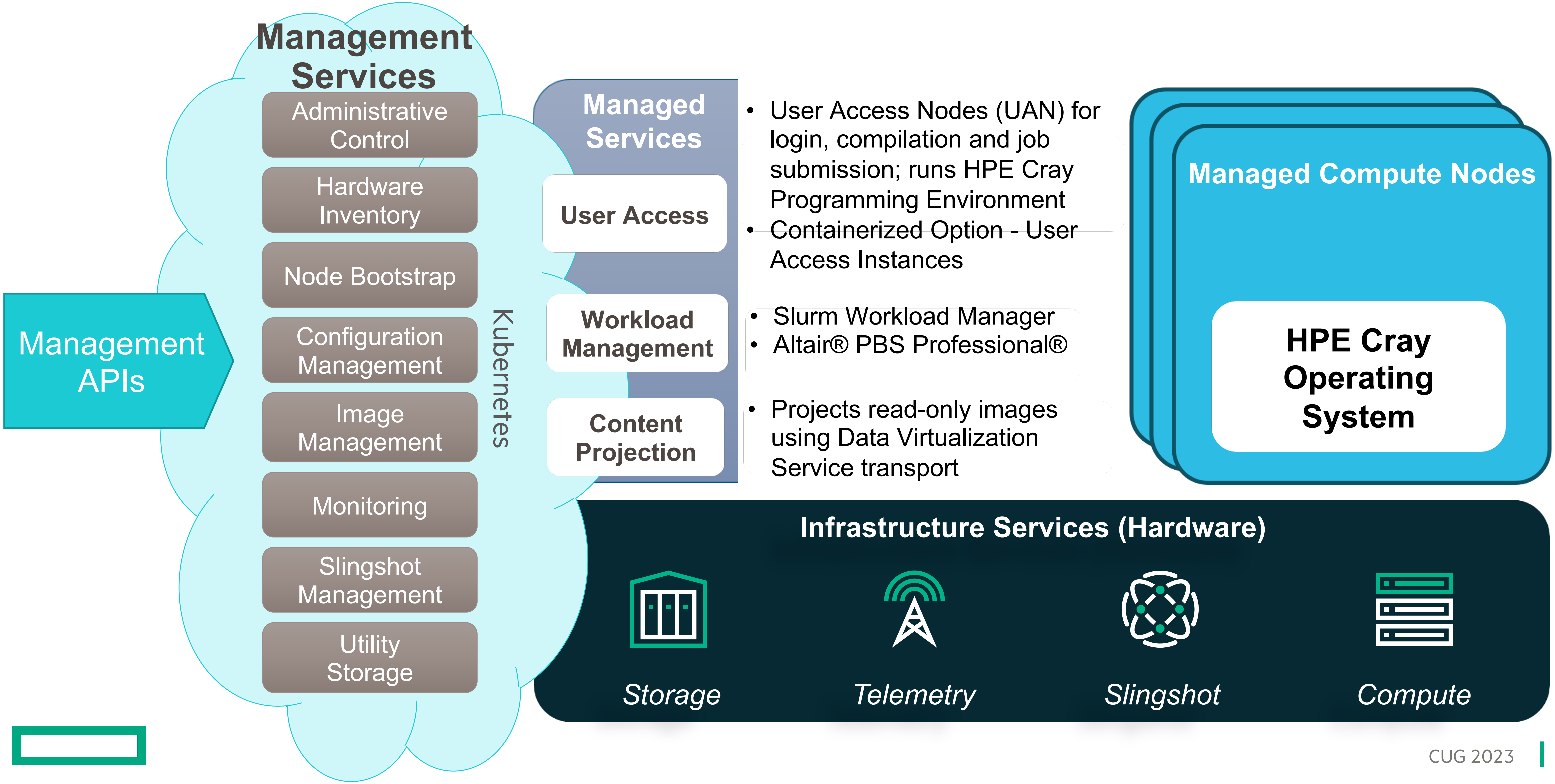
Flexible
Enable cloud-like secure multitenant operations with extensible microservices APIs

Proven
Used by customers globally with large supercomputing systems

Systems Administration & Automation



HPE CRAY SYSTEM MANAGEMENT SOLUTION OVERVIEW



Management APIs

Management Services

- Administrative Control
- Hardware Inventory
- Node Bootstrap
- Configuration Management
- Image Management
- Monitoring
- Slingshot Management
- Utility Storage

Kubernetes

Managed Services

User Access

- User Access Nodes (UAN) for login, compilation and job submission; runs HPE Cray Programming Environment
- Containerized Option - User Access Instances

Workload Management

- Slurm Workload Manager
- Altair® PBS Professional®

Content Projection

- Projects read-only images using Data Virtualization Service transport

Managed Compute Nodes

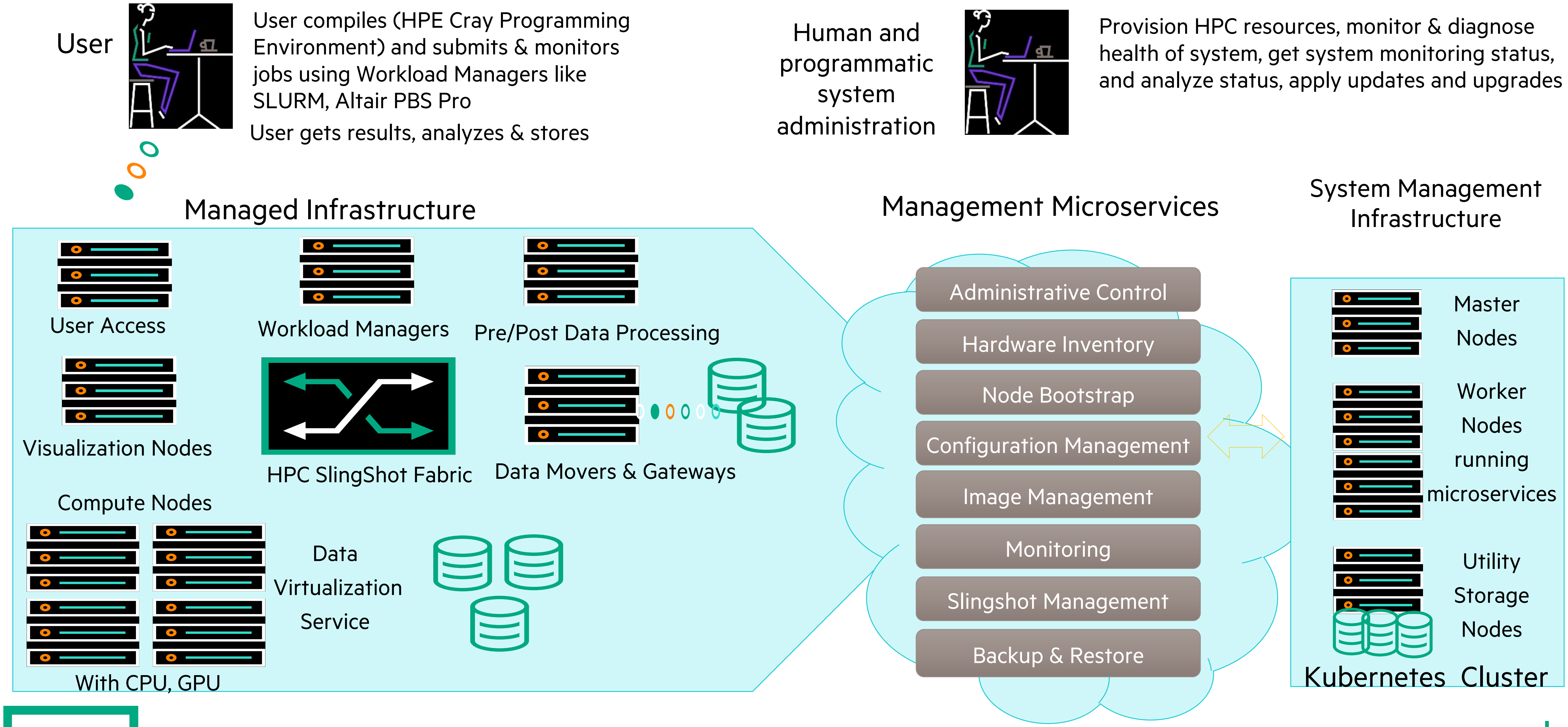
HPE Cray Operating System

Infrastructure Services (Hardware)

Storage *Telemetry* *Slingshot* *Compute*

HPE CRAY SYSTEMS MANAGEMENT COMPONENTS

Manage Exascale Supercomputers to deliver optimal performance for HPC workloads



User

User compiles (HPE Cray Programming Environment) and submits & monitors jobs using Workload Managers like SLURM, Altair PBS Pro

User gets results, analyzes & stores

Human and programmatic system administration

Provision HPC resources, monitor & diagnose health of system, get system monitoring status, and analyze status, apply updates and upgrades

Managed Infrastructure

- User Access
- Workload Managers
- Pre/Post Data Processing
- Visualization Nodes
- HPC SlingShot Fabric
- Data Movers & Gateways
- Compute Nodes With CPU, GPU
- Data Virtualization Service

Management Microservices

- Administrative Control
- Hardware Inventory
- Node Bootstrap
- Configuration Management
- Image Management
- Monitoring
- Slingshot Management
- Backup & Restore

System Management Infrastructure

- Master Nodes
- Worker Nodes running microservices
- Utility Storage Nodes

Kubernetes Cluster

HPE CRAY SYSTEM MANAGEMENT UNIQUE ATTRIBUTES

System management software designed for Exascale HPC and beyond

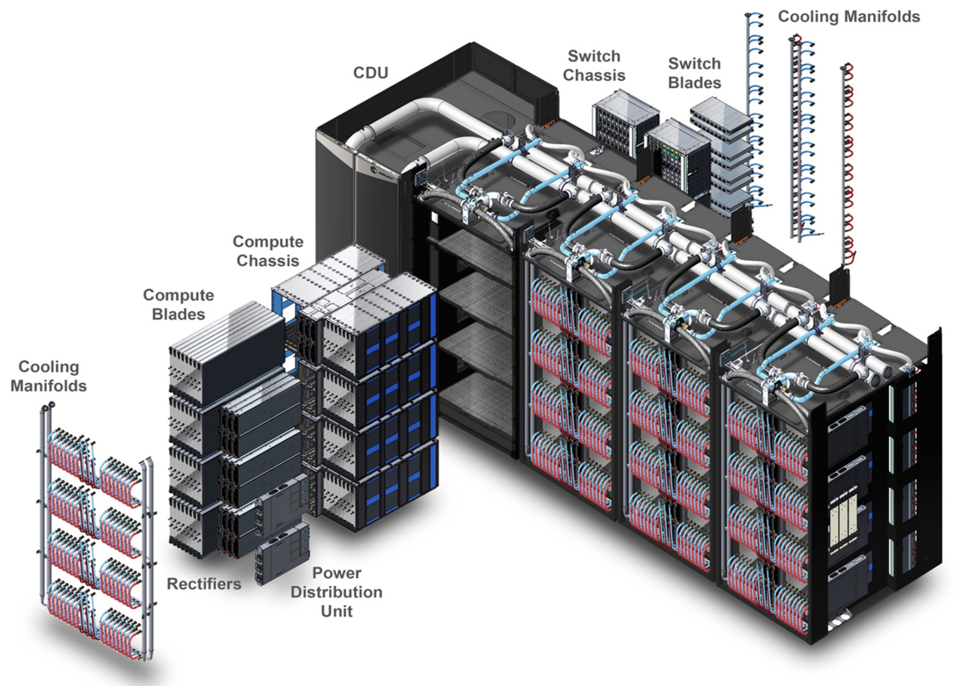
Key Capabilities

- Comprehensive monitoring and management of all aspects of the system: CPU/GPU, network (integrated Cray Slingshot Fabric Manager), power management and monitoring combined with provisioning for operational efficiency
- REST APIs & standard systems management protocols enable full interoperability and extensibility of monitoring, management, and automation capabilities
- Infrastructure-as-code: Login nodes as dynamic containers (User Access Instances), workload managers as containerized services
- Built from open-source software components, is open-source software

Unique Attributes

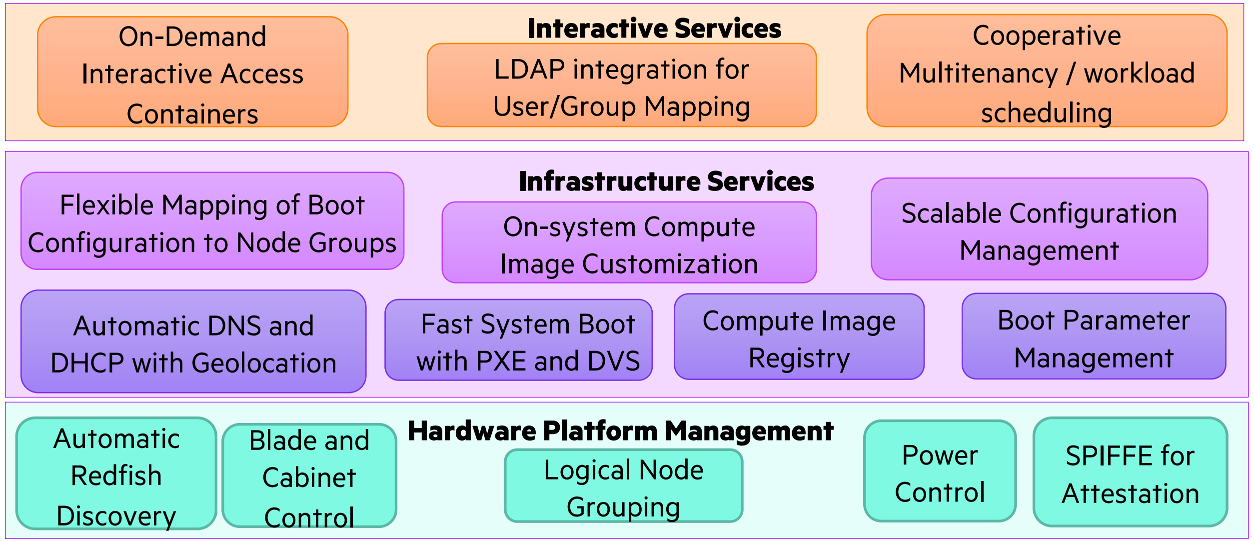
- Kubernetes platform for running system management and sysadmin tooling enabling infrastructure-as-code & CI/CD for jobs, tenants, and environments
- Declarative and dynamic inventory and state management represents single source of truth (configurations and artifacts), continuous delivery
- aaS Security with auditable access to all APIs
- Supports scalable deployment with massive system extensibility

HPE Cray EX Supercomputer &
HPE Cray Supercomputer with HPE Slingshot
Exascale and beyond scalable hardware architecture and infrastructure



HPE CRAY SYSTEM MANAGEMENT IS ELASTIC AND RESILIENT

- Flexible Deployment Options
  - Management Kubernetes cluster scales with more nodes, CPUs, memory, network, and storage
    - Proven to scale from small number of nodes to more than 50 worker nodes for very large customer deployments
- Elasticity
  - Services are continuously checked and updated to match state
  - When nodes are added or subtracted or the load suddenly changes, configuration is automatically modified
    - Autoscale Horizontally and Vertically within constraints
    - When the system is under-scaled, microservices fail according to defined priorities
- Resiliency
  - Microservices are active/active HA
    - Separate gateways and individual load balancers
    - Multiple Pods
    - Rolling deployments and rollbacks
  - Managed nodes running custom app services have HA



Interactive Services

On-Demand Interactive Access Containers

LDAP integration for User/Group Mapping

Cooperative Multitenancy / workload scheduling

Infrastructure Services

Flexible Mapping of Boot Configuration to Node Groups

On-system Compute Image Customization

Scalable Configuration Management

Automatic DNS and DHCP with Geolocation

Fast System Boot with PXE and DVS

Compute Image Registry

Boot Parameter Management

Hardware Platform Management

Automatic Redfish Discovery

Blade and Cabinet Control

Logical Node Grouping

Power Control

SPIFFE for Attestation

K8s Master Nodes

K8s Worker Nodes

Common footprint

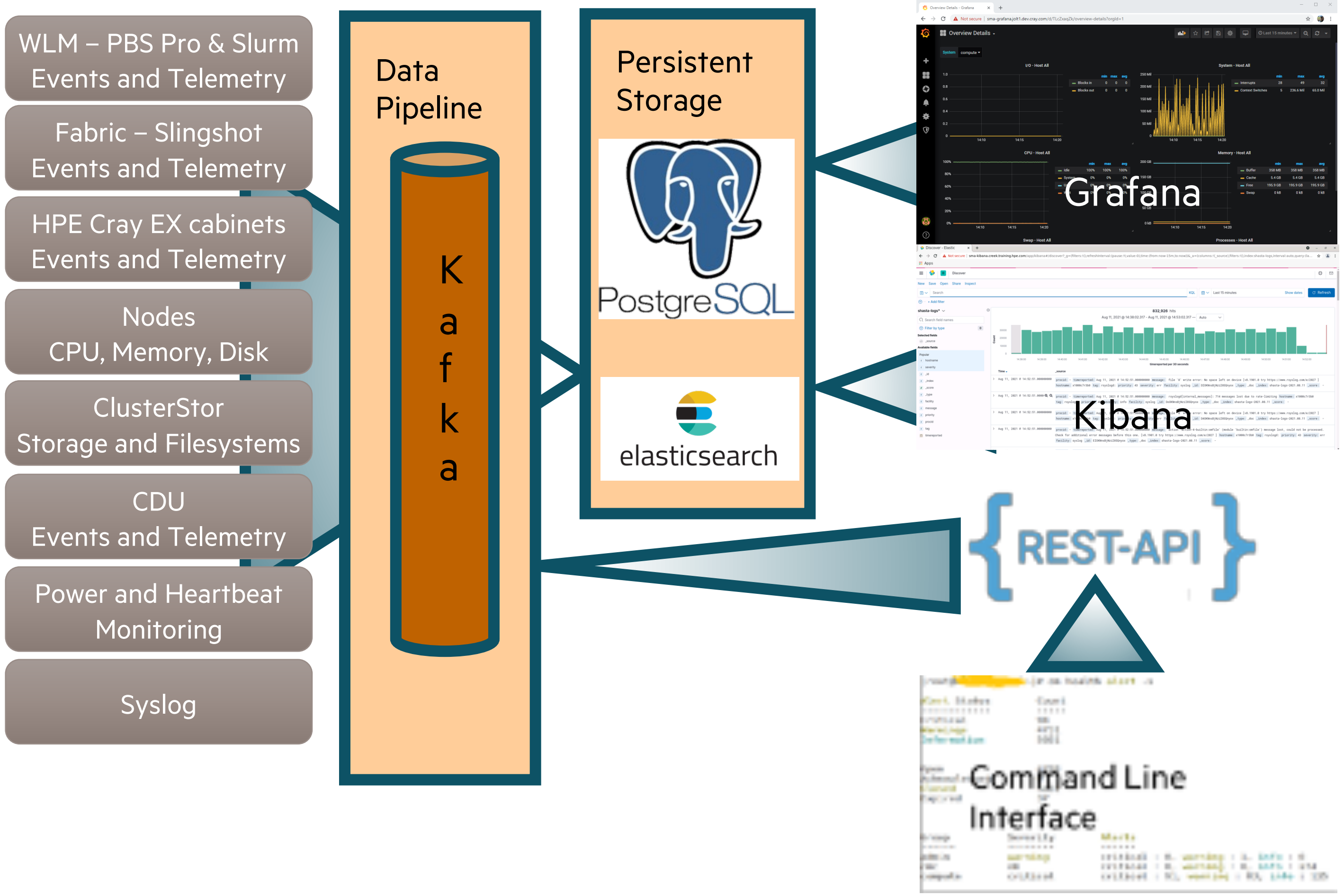
- 3 Kubernetes Master nodes for active failover
- 4+ K8s Worker nodes
- 3+ Utility storage nodes for state abstraction

SCALABLE MONITORING AND MANAGEMENT

HPE Cray Systems management offers fine-grained centralized monitoring and management of your Exascale HPC systems to keep it performing at its best

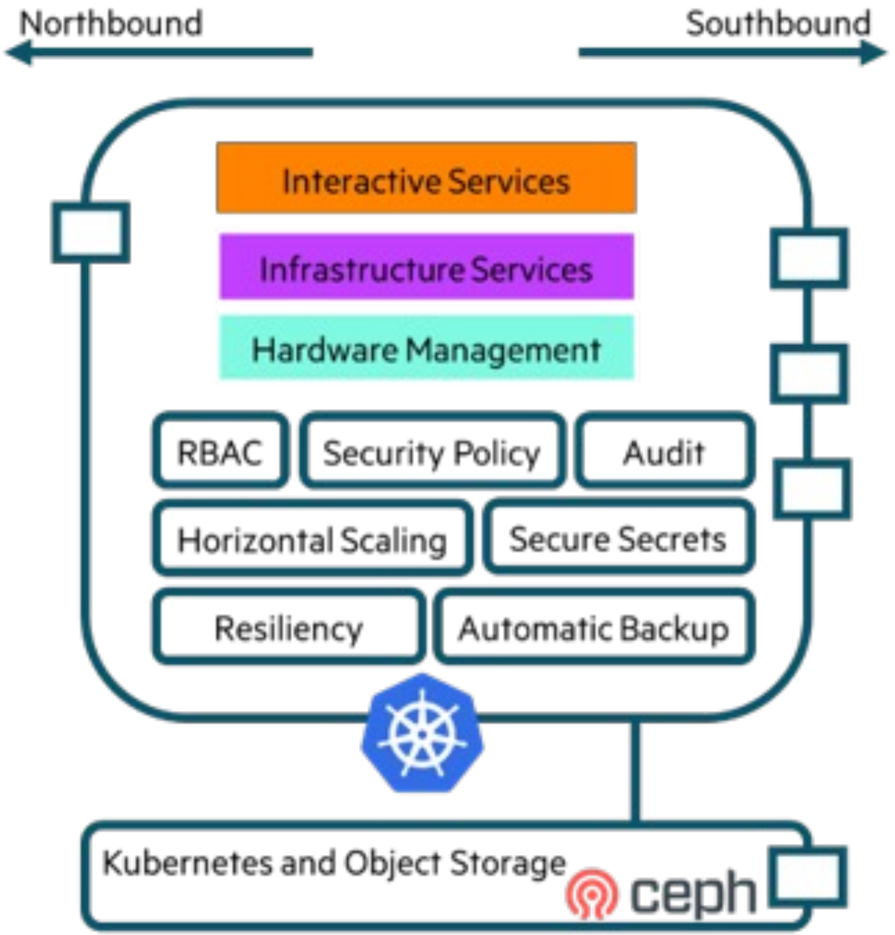
CPUs | **GPUs** | **Memory and disks** | **Cooling and networking** | **Software**

- In-band LDMS (Lightweight Distributed Metric Service) and out of band telemetry
- Access metrics and alerts via GUI, CLI, REST APIs
- Customize system telemetry and alerts to best suit your needs
- Set up automatic reactions to events to prevent failures

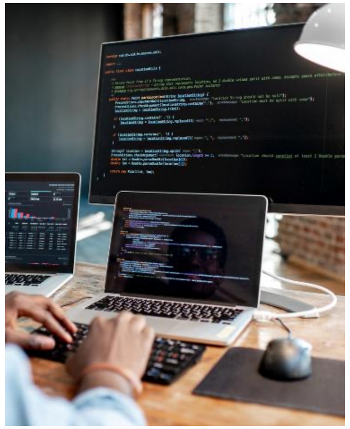


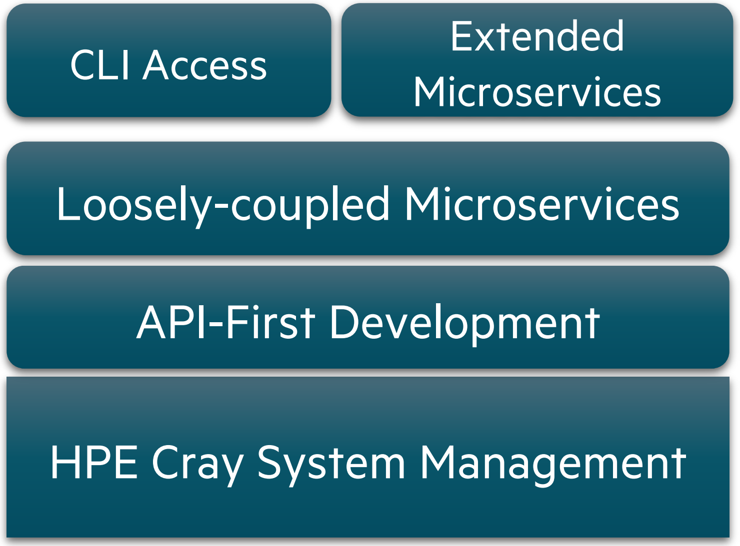
HPE CRAY SYSTEM MANAGEMENT DESIGNED FOR AS-A-SERVICE SECURITY

- CSM supports human and non-human IAM (Identity and Access Management)
- Fully supported custom RBAC (Role Based Access Control)
  - No limits to the group or role structure, infinite customization
  - Control managed entities with a URL
  - Programmatic interface for change control after upgrades, patches, etc.
- Multiple identity providers
- Credentials management
- Certificate management
- Mesh network encryption (TLS) and access policies
- DNS and external zone transfers
- Non-root users
- User traffic isolation - necessary for multitenancy
- Node attestation
  - SPIFFE (Secure Production Identity Framework For Everyone) provides a secure identity with X.509 certificate to every workload
  - SPIRE (SPIFFE Runtime Environment) manages platform and workload attestation, has API, and handles certificate issuance and rotation



CRAY SYSTEM MANAGEMENT EXTENSIBILITY FOR SYSTEM OPERATIONS





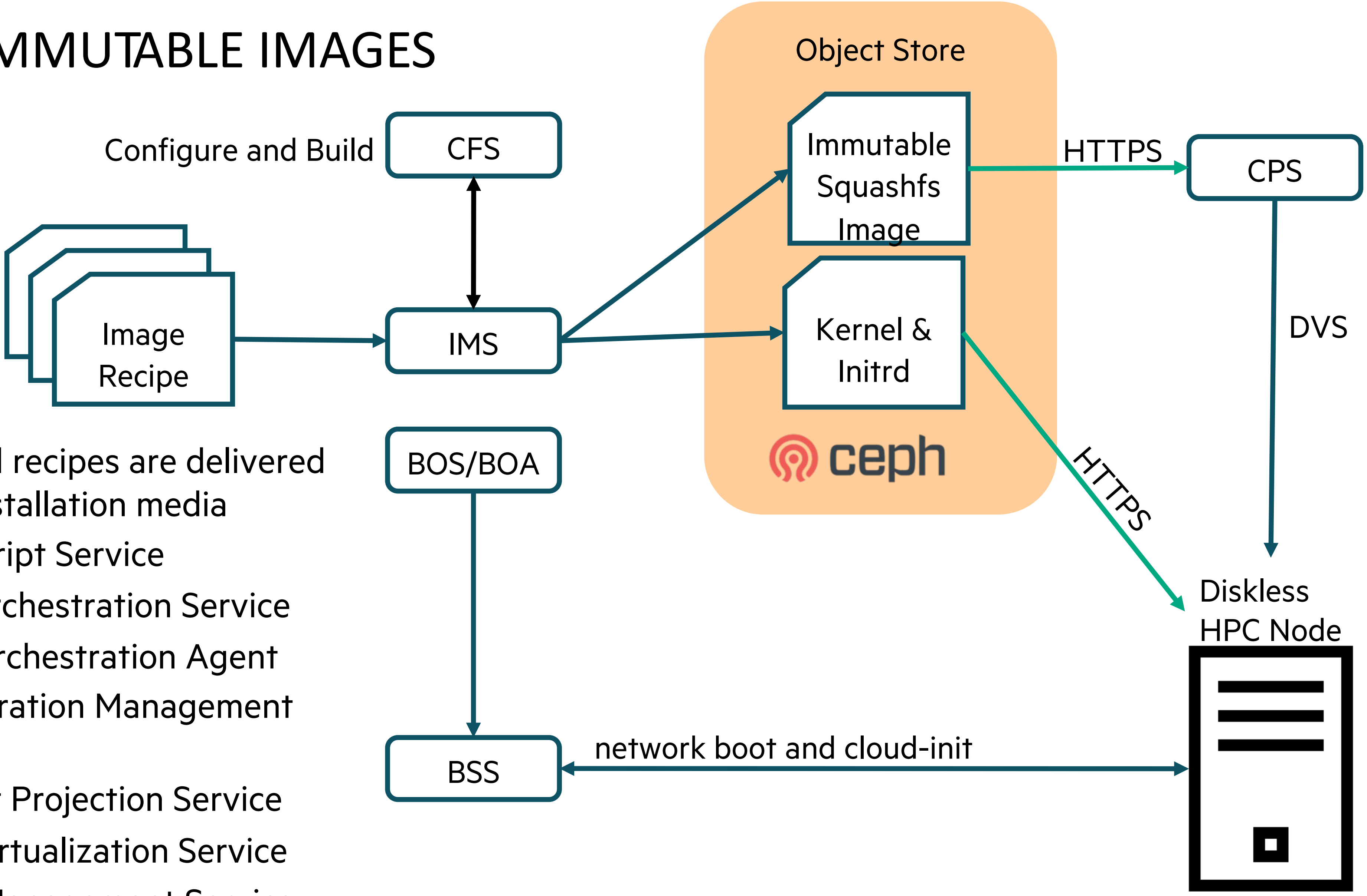
API-First Development

- Nearly 100% of the systems management functionality is exposed via API
- Machine readable Swagger API definitions are available for all
- Cray CLI– a tool for discovering and implementing the APIs
- System Administration Toolkit (SAT) – a CLI tool covering more common workflows spanning APIs

Loosely-coupled Microservices

- Customers are developing their own APIs to extend functionality
- Customers can pick and choose which HPE provided aspects to use or replace
- Enables granular deployment elasticity
  - Not limited because of a monolithic application design
  - “[this] functionality should scale and failover in [these] ways”
- Can be updated continuously with high confidence

BOOTING IMMUTABLE IMAGES



Both images and recipes are delivered as part of the installation media

- BSS: Boot Script Service
- BOS: Boot Orchestration Service
- BOA: Boot Orchestration Agent
- CFS: Configuration Management Service
- CPS: Content Projection Service
- DVS: Data Virtualization Service
- IMS: Image Management Service

HPE CRAY PROGRAMMING ENVIRONMENT

Essential toolset for HPC organizations developing HPC code in-house.

Fully integrated software suite with compilers, tools, and libraries designed to increase programmer productivity, application scalability, and performance.

Complete toolchain

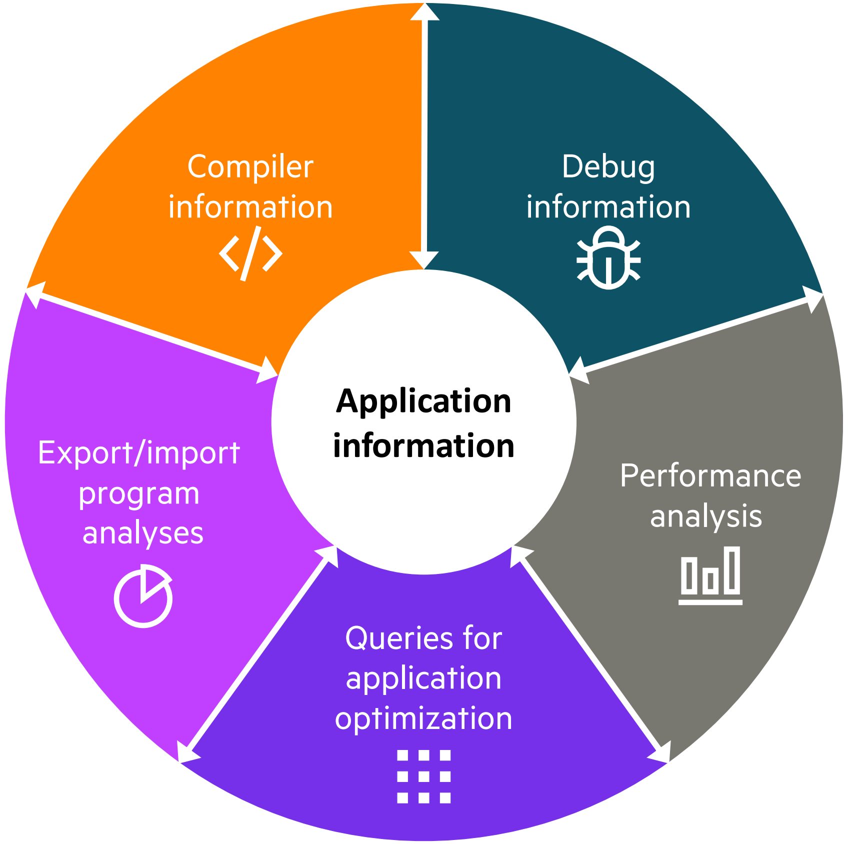
Cross platform

Programmable

Scalable

Holistic support

From HPC experts for HPC experts



Application information

Compiler information

Debug information

Performance analysis

Queries for application optimization

Export/import program analyses

COMPREHENSIVE TOOLCHAIN

HPE Cray Programming Environment

Software

Application Development

- C/C++ and Fortran compilers
- I/O, scientific, and math libraries
- HPE Cray MPI
- Machine learning plug-ins

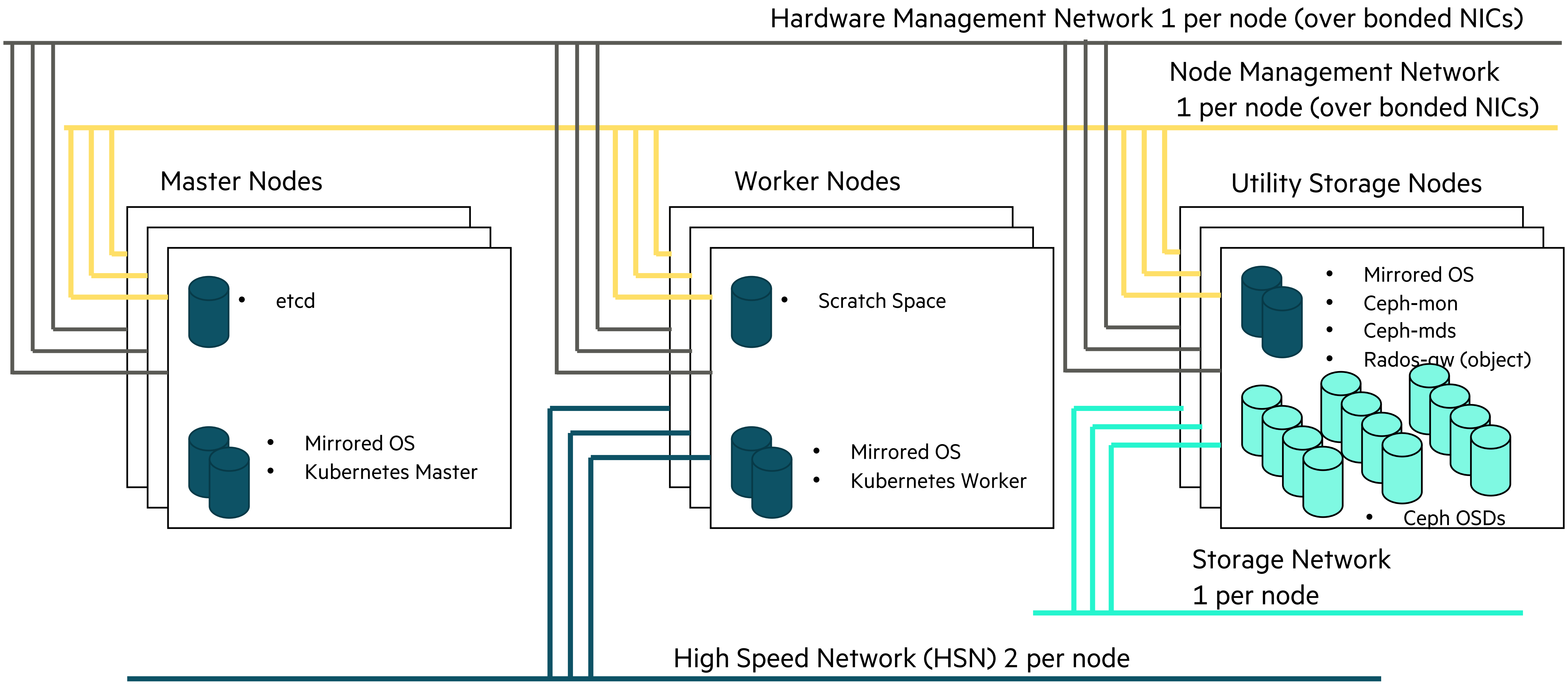
Debugging

- Stack tracing at scale
- Parallelized gdb for scale
- Compare two versions of an application
- Manage core files at scale
- Memory debugging at scale

Performance analysis and Optimization tools

- Whole program profiling and visualization
- HPC optimization tool for scale, parallelization, memory usage, bandwidth

MANAGEMENT NODES



COMMON COMMANDS

Command	Description
kubectl	CLI for Kubernetes cluster's control plane, using the Kubernetes API • jsonpath - kubectl uses JSONPath expressions to filter on specific fields in the JSON object and format the output
ceph	Control utility for manual deployment and maintenance of a Ceph cluster
cephadm	cephadm - deploys and manages a Ceph cluster
cray	CLI framework integrates system management REST APIs into easily usable commands • Outputs data in JSON, YAML, TOML
sat	CLI interacts with the REST APIs of many services to perform more complex system management tasks • Outputs data in JSON, YAML, TOML
fmctl	CLI for Slingshot fabric management
stt	CLI for Slingshot Topology Tool
jq	command works on JSON data to slice and filter and map and transform structured data like sed, awk, grep and friends let you play with text
Linux tools	systemctl, journalctl, pdsh/dshbak, curl

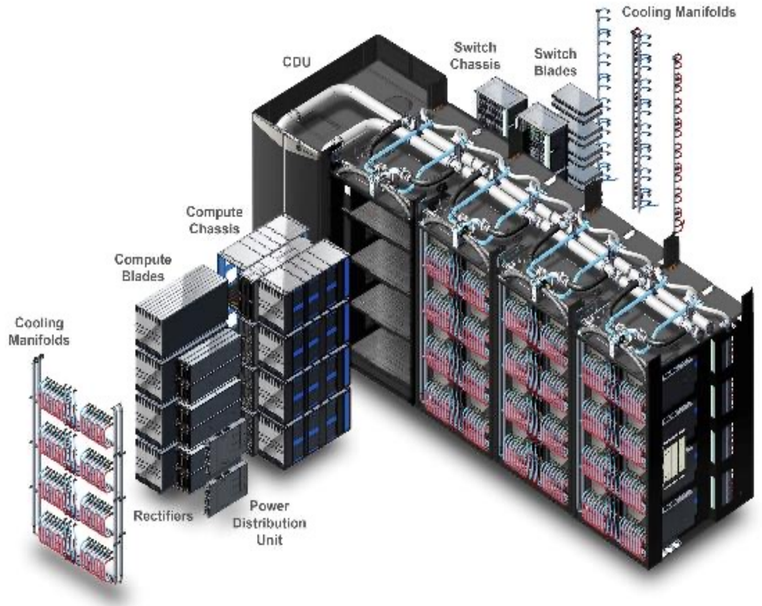
HPE CRAY EX HARDWARE

FLEXIBLE COMPUTE INFRASTRUCTURE

HPE Cray EX liquid-cooled optimized cabinet (Olympus)

- Up to 64 compute blades, and 512 processors per rack
- Flexible bladed architecture supports multiple generations of CPUs, GPUs, and interconnect
- Cableless interconnect between switches and nodes inside chassis
- 100% direct liquid-cooling – no fans
- Up to 400KW capability per rack
- Designed to provide an optimal solution for tens to hundreds of thousands of nodes, scales to hundreds of cabinets
- CEC (Cabinet Environment Controller)
- CMM (Chassis Management Module)
- CDU (Coolant Distribution Unit) supports up to 4 cabinets

Scaling building block



Choice of blade types for optimal density, efficiency, and cost per compute node

AIR-COOLED CABINETS

HPE Cray standard air-cooled cabinet (River)

- Standard 19” cabinet
- Air-cooled, but with optional liquid-cooled door
- One or more cabinets with Management infrastructure nodes
- One or more cabinets with high-performance and capacity Storage
- One or more cabinets with commodity compute nodes (CPU and GPU)
- PDU
- Management network switches
- Slingshot network switches

Management infrastructure, high-performance parallel filesystem, commodity compute nodes

HPE CRAY COMPONENT NAMES (XNAMES)

Component	Xname Scheme	Examples	Note
Cabinet	**`x#`**	`x1000 , x3000`	Cabinets don't have an X-Y grid
CDU	**`d#`**	`d0`	Up to 4 liquid-cooled cabinets per CDU
Chassis	**`x#c#`**	`x1000c3,` `x3000c0`	Air-cooled cabinets don't have chassis but for consistency always use c0 for chassis 0
Compute Blade Slot	**`x#c#s#`**	`x1000c3s4,` `x3000c0s22`	In air-cooled cabinets the slot is the lowest rack U height occupied by a server
Node card controller	**`x#c#s#b#`**	`x1000c3s4b1,` `x3000c0s22b2`	1st example - Node card 1 of blade 4 in chassis 3 2nd example - BMC in air-cooled 4 node server
Node	**`x#c#s#b#n#`**	`x1000c3s4b1n1,` `x3000c0s22b2n0`	Nodes are dependent on their BMCs
Processor	**`x#c#s#b#n#p#`**	`x1000c3s4b1n1p0,` `x3000c0s22b2n0p1`	Processor sockets are zero-based in xnames
Slingshot Switch	**`x#c#r#`**	`x1000c3r7,` `x3000c0r42`	Air-cooled Slingshot switches use rack "U" height just like air-cooled servers
Ethernet Switch	**`x#c#w#`**	`d0w1, x3000c0w38`	CDU, LeafBMC, and Leaf switches extend SMNet

NETWORKS

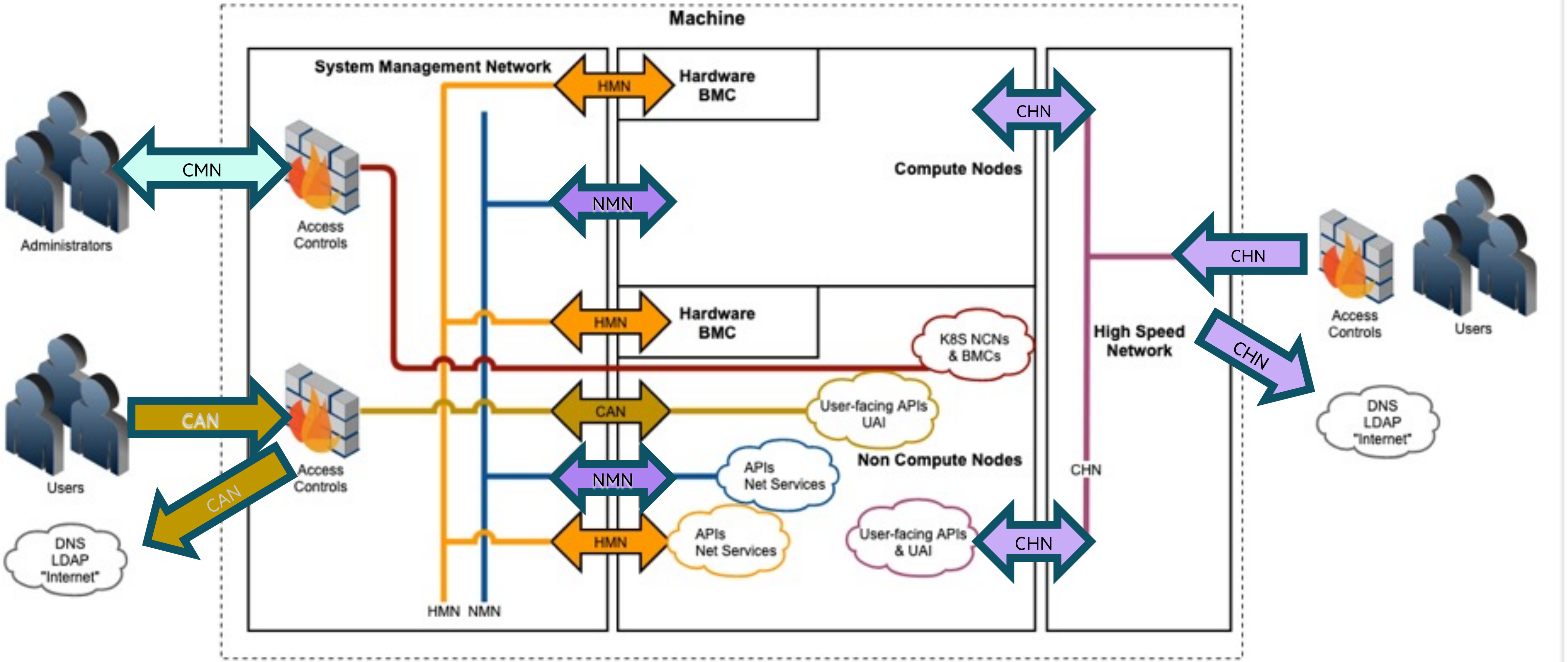
- Management
- Customer Access
- Slingshot

SYSTEM MANAGEMENT NETWORK (SMNET) OVERVIEW

- Standard Ethernet fabric directly connected to every node and controller in the system
  - Leaf/Spine topology implemented with commodity switches
  - Divided into multiple “Virtual Networks” implemented with VLANs and Access Control Lists

Virtual Network	Connections
Node Management Network (NMN)	• All Non-Compute Nodes (NCNs) • Air-cooled Compute Nodes • Liquid-cooled Compute Nodes
Hardware Management Network (HMN)	• Air-cooled Nodes (Compute and NCN) BMCs • All Slingshot Switch Controllers (sC) • Liquid-cooled Node Controllers (nC) • Liquid-cooled Chassis Controllers (cC) • Air-cooled Hardware Controllers (smart PDUs, CMCs, etc) • SMNet switch management ports
Customer Access Network (CAN)	• Upon upgrade to CSM 1.2, the old CAN will be split to create CMN and (either CAN or CHN) – Allows only user traffic and CAN API gateway
Customer Management Network (CMN)	• Allows administrative access to nodes and CMN API gateway
Customer High Speed Network (CHN)	• Allows user access to application nodes, UAI, compute nodes, and CHN API gateway from customer site via the HSN

CUSTOMER ACCESS OVER CMN, CAN, CHN



CSM BIFURCATED CAN HOSTNAMES

- User and administrative traffic segregation so URLs for certain services now include the network path in the fully qualified domain name
- Access to administrative services is restricted to the Customer Management Network (CMN)
- API access is available via the Customer Management Network (CMN), Customer Access Network (CAN), or Customer Highspeed Network (CHN)

Old Name (CSM 1.0)	**New Name (CSM 1.2 and 1.3)**
auth.shasta.dev.cray.com	auth.cmn.shasta.dev.cray.com
nexus.shasta.dev.cray.com	nexus.cmn.shasta.dev.cray.com
grafana.shasta.dev.cray.com	grafana.cmn.shasta.dev.cray.com
prometheus.shasta.dev.cray.com	prometheus.cmn.shasta.dev.cray.com
alertmanager.shasta.dev.cray.com	alertmanager.cmn.shasta.dev.cray.com
vcs.shasta.dev.cray.com	vcs.cmn.shasta.dev.cray.com
kiali-istio.shasta.dev.cray.com	kiali-istio.cmn.shasta.dev.cray.com
s3.shasta.dev.cray.com	s3.cmn.shasta.dev.cray.com
sma-grafana.shasta.dev.cray.com	sma-grafana.cmn.shasta.dev.cray.com
sma-kibana.shasta.dev.cray.com	sma-kibana.cmn.shasta.dev.cray.com
api.shasta.dev.cray.com	api.cmn.shasta.dev.cray.com api.chn.shasta.dev.cray.com api.can.shasta.dev.cray.com

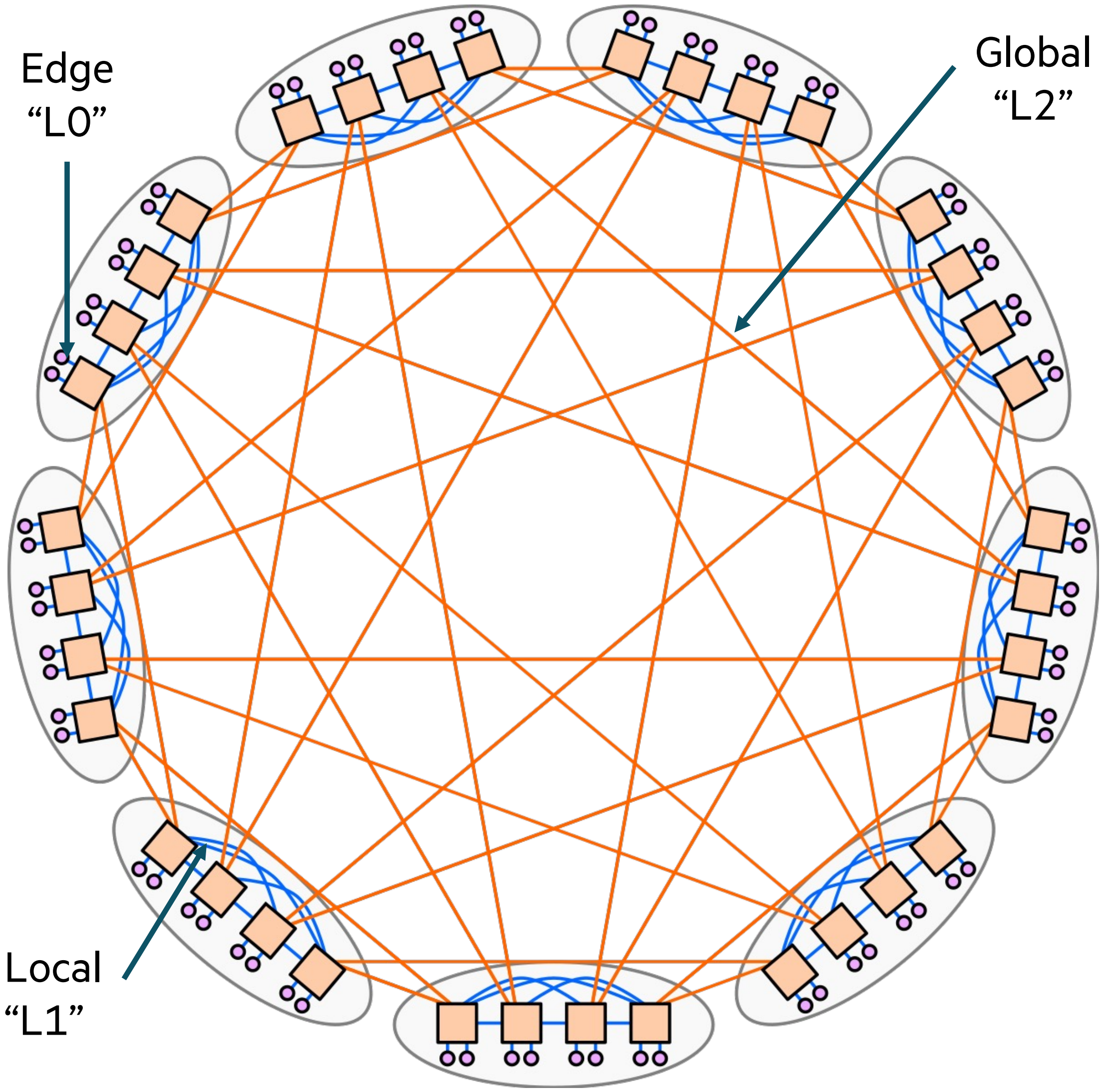
SLINGSHOT DRAGONFLY TOPOLOGY

Dragonfly Topology

- Provides All-to-All connectivity across the fabric
- Reduces costs of network hardware
- Efficient and consistent connectivity

Link Types

- Edge
  - Nodes are connected directly to Switches
  - *These are called "Edge" or "L0" Links*
- Local
  - Groups of Switches connected all-to-all
  - *All switches within a group have links between them*
  - *These are called "Local" , "Group" or "L1" Links*
- *Global*
  - *Links connect different groups together*
  - *These are called "Global" or "L2" Links*



HPE CRAY EX SYSTEM OVERVIEW

ANSIBLE BEST PRACTICES

MONITORING TOOLS

SYSTEM MANAGEMENT HEALTH

TUNING COMPUTE NODES

SYSTEM ADMIN TOOLKIT

TROUBLESHOOTING BOOT FAILURES

COLLECTING DATA FOR HPE SERVICE

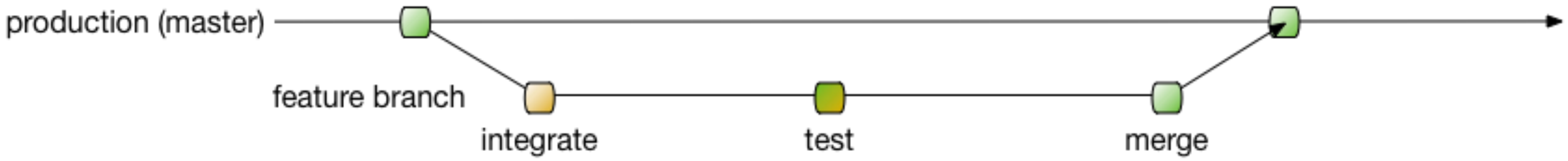
RESOURCES

ANSIBLE BEST PRACTICES

- Version Control Service (VCS)
- Configuration Framework Service (CFS)
- Ansible best practices
- Ansible profiling

CONFIGURATION WITH CFS AND VCS

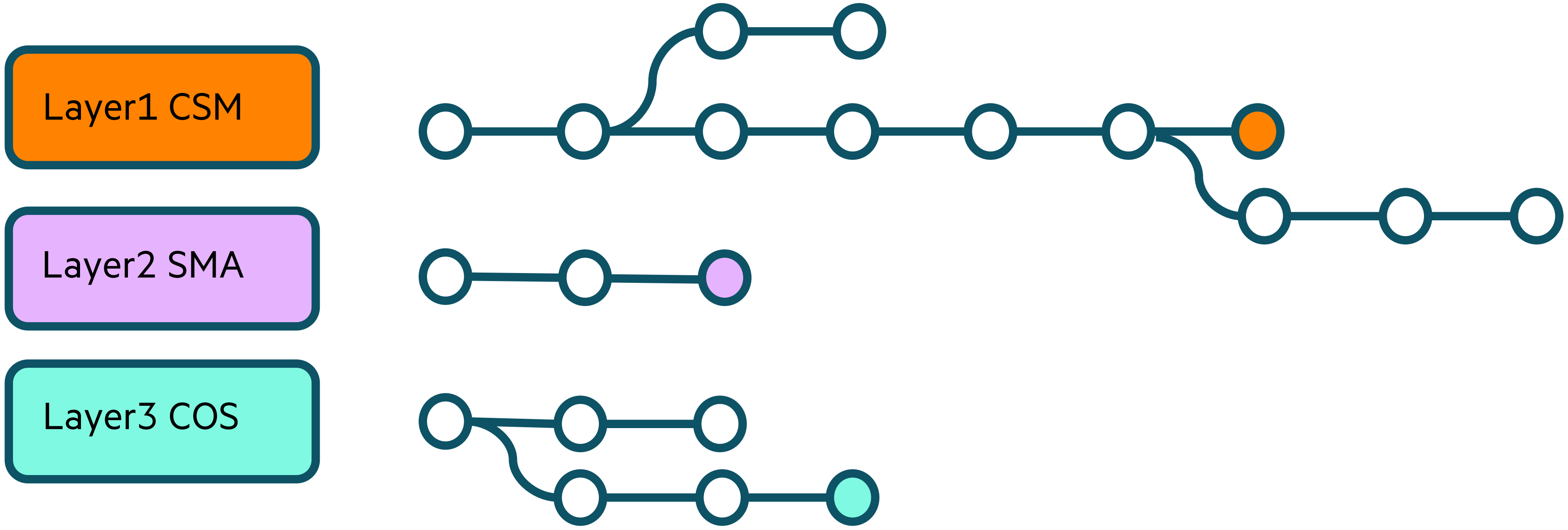
- Version Control Service (VCS)
  - Manages configuration data and content
    - Compute image configuration YAML files
  - *Gitea* server holds configuration content



- Configuration Framework Service (CFS)
  - Manages the launch of configuration actions
  - Does `git-clone` of configuration data and content from VCS
  - Launches Ansible Execution Environment (AEE) which runs Ansible playbook for target inventory
    - Either hostnames of nodes for node personalization or reconfiguration
    - Or IMS build environment for image customization
  - Aggregates status to show how many targets passed/failed the Ansible run

USING GIT FOR MANAGING CFS CONFIGURATION

- Stores Ansible to apply to nodes at lifecycle events
- All Ansible in git repositories with branches to allow site customization
- Ordered configuration management across multiple repositories
- CFS sessions as part of pre-boot Image Customization as well as post-boot Node Personalization



SAMPLE GIT SEQUENCE

```
ncn# git clone https://api-gw-service-nmn.local/vcs/cray/uan-config-management.git
Username for 'https://api-gw-service-nmn.local': crayvcs
Password for 'https://crayvcs@api-gw-service-nmn.local’:
```

Checkout from VCS

```
ncn# cd uan-config-management
ncn# git checkout cray/uan/2.4.3
ncn# git pull
ncn# git checkout -b integration && git merge cray/uan/2.4.3
```

Make local branch

```
ncn# vi <file(s) to be edited>
ncn# git mv <file(s) to be renamed>
ncn# git rm <file(s) to be removed>
```

Change something

```
ncn# git status
ncn# git diff
ncn# git add <file(s) in repo that were added or edited>
ncn# git status
```

Compare and Update

```
ncn# git commit -m "<some message about the change>"
```

Describe change

```
ncn# git push --set-upstream origin integration
```

Push changes to git

```
ncn# git rev-parse --verify HEAD
ecece54b1eb65d484444c4a5ca0b244b329f4667
```

Git commit ID to be used on CFS layer

CONFIGURATION FRAMEWORK SERVICE

- Provides a configuration framework for HPE and customers which integrates industry-standard configuration management tooling (Ansible) with Cray services
- Flexible workflow
  - Pre-boot image customization
  - Post-boot node personalization
- Provides dynamic inventory plugins to target Cray nodes for configuration
- CFS is integrated with other Cray Management Services:
  - Image Management Service (IMS)
  - Nexus Repository Manager
  - Version Control Service (VCS)
  - Boot Orchestration Service (BOS)
  - Artifact Repository / S3
- Configurations are applied in layers
- Configurations are processed in batches

CFS CONFIGURATIONS

```
ncn# cray cfs configurations describe compute-slurm-cpe-21.6.5 --format json
{
  "lastUpdated": "2021-06-24T18:58:25Z",
  "layers": [
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": "97209cb3e6c128e0b8c1eaae0e683227c57910ee",
      "name": "cos-integration-2.1.70",
      "playbook": "site.yml"
    },
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/slurm-config-management.git",
      "commit": "b302e1b672e27f74c36ceacfd2ed6bd50ed14c0a",
      "name": "slurm-integration-0.1.3",
      "playbook": "site.yml"
    },
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cpe-config-management.git",
      "commit": "43f3a36bca35d693a583d1643fe1cebb0ccaf7fe",
      "name": "cpe-integration-21.6.5",
      "playbook": "pe_deploy.yml"
    }
  ],
  "name": " compute-slurm-cpe-21.6.5 "
}
```

CFS COMPONENTS

```
ncn# cray cfs components describe x1000c0s5b0n1 --format json
{
  "configurationStatus": "configured",
  "desiredConfig": " compute-slurm-cpe-21.6.5 ",
  "enabled": true,
  "errorCount": 0,
  "id": "x1000c0s5b0n1",
  "retryPolicy": 3,
  "state": [
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": " 97209cb3e6c128e0b8c1eaae0e683227c57910ee",
      "lastUpdated": "2021-11-17T18:44:41Z",
      "playbook": "site.yml",
      "sessionName": "batcher-f80ebbdb-c4ec-4025-8156-68205b22ccdf"
    },
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/slurm-config-management.git",
      "commit": " b302e1b672e27f74c36ceacfd2ed6bd50ed14c0a",
      "lastUpdated": "2021-11-17T19:47:29Z",
      "playbook": "site.yml",
      "sessionName": "batcher-b57c437f-33e9-46d7-9416-8c955f773504"
    },
<< snip >>
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cpe-config-management.git",
      "commit": " 43f3a36bca35d693a583d1643fe1cebb0ccaf7fe",
      "lastUpdated": "2021-12-06T20:42:02Z",
      "playbook": "pe_deploy.yml",
      "sessionName": "batcher-bdea16db-dae5-4f7a-bffe-40f0a179d328"
    }
  ],
  "tags": {
    "bos_session": "d5f69110-dca6-4ecb-890f-3622957589fe"
```

The configuration for a component and whether it is enabled are set by BOS according to the `sessiontemplate`

If configuration fails it will be automatically retried up to the number specified in the `retryPolicy`

To see configuration (ansible) output check the cfs sessions to find configuration jobs and then check the logs of the ansible-N pods within those jobs.

CFS SESSIONS

```
ncn# cray cfs sessions describe batcher-080ba574-0a99-409b-a639-a45c73c25e63 --format json
{
  "ansible": {
    "config": "cfs-default-ansible-cfg",
    "limit": "x3000c0s26b0n0",
    "verbosity": 0
  },
  "configuration": {
    "limit": "",
    "name": "uan-config-2.0.0"
  },
  "name": "batcher-080ba574-0a99-409b-a639-a45c73c25e63",
  "status": {
    "artifacts": [],
    "session": {
      "completionTime": "2021-10-18T20:34:18",
      "job": "cfs-e78738d3-99a9-4b73-bce1-a720b34a714d",
      "startTime": "2021-10-18T20:31:15",
      "status": "complete",
      "succeeded": "true"
    }
  },
  "tags": {
    "bos_session": "bf88ad75-6a02-470c-85ca-4708a7f9fe0d"
  },
  "target": {
    "definition": "dynamic",
    "groups": null
  }
}
ncn# kubectl logs -n services cfs-e78738d3-99a9-4b73-bce1-a720b34a714d-ps4ls
error: a container name must be specified for pod cfs-e78738d3-99a9-4b73-bce1-a720b34a714d-ps4ls, choose one of: [inventory ansible-0 ansible-1
ansible-2 istio-proxy] or one of the init containers: [git-clone-0 git-clone-1 git-clone-2 istio-init]
```

The limit shows which node(s) are configured by each session

Kubernetes jobs control one or more pods and the job name is typically the start of the pod name

Each layer will be executed by a different container within the cfs job or possibly a different job

The containers names will have the format ansible-N (e.g., ansible-0)

CFS-BATCHER SCHEDULING RULES

- Every 10 seconds the batcher checks for components that need configuration
- Components (nodes) are assigned to a batch if:
  - They need configuration
  - They are not disabled
  - They are currently not assigned to a batch
- Components are grouped according to their desired state information.
- A new batch is created if
  - no partial batches match the desired state
  - all similar batches are full
- Batches are scheduled as CFS sessions when either
  - The batch is full
  - The batch window time has been exceeded

```
ncn# cray cfs options list --format json
{
  "additionalInventoryUrl": "",
  "batchSize": 25,
  "batchWindow": 60,
  "batcherCheckInterval": 10,
  "defaultAnsibleConfig": "cfs-default-ansible-
cfg",
  "defaultBatcherRetryPolicy": 1,
  "defaultPlaybook": "site.yml",
  "hardwareSyncInterval": 10,
  "sessionTTL": "7d"
}
```

WHY ISN’T CFS RUNNING?

```
ncn-m001:~ # kubectl logs -n services cray-cfs-batcher-5d58b8964c-tdsm2 -c cray-cfs-batcher
2021-09-16 09:19:54,225 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 09:20:54,910 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:21:15,163 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 09:21:25,250 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:22:15,759 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 09:23:26,546 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:23:26,547 - WARNING - batcher.batch - The 20 most recent configuration sessions have failed. Halting session creation for 60 seconds
2021-09-16 09:24:27,136 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:24:27,136 - WARNING - batcher.batch - The 20 most recent configuration sessions have failed. Halting session creation for 120 seconds
2021-09-16 09:26:28,170 - INFO    - batcher.batch - Successfully submited 2 batches for configuration
2021-09-16 09:27:49,098 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 09:28:49,865 - INFO    - batcher.batch - 2 batches/sessions have completed
2021-09-16 09:28:49,866 - WARNING - batcher.batch - The 20 most recent configuration sessions have failed. Halting session creation for 240 seconds
2021-09-16 09:29:50,468 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:32:52,036 - INFO    - batcher.batch - Successfully submited 2 batches for configuration
2021-09-16 09:34:53,393 - INFO    - batcher.batch - 2 batches/sessions have completed
2021-09-16 09:34:53,393 - WARNING - batcher.batch - The 20 most recent configuration sessions have failed. Halting session creation for 480 seconds
2021-09-16 09:42:57,206 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 09:44:28,008 - INFO    - batcher.batch - 1 batches/sessions have completed
2021-09-16 09:44:28,008 - WARNING - batcher.batch - The 20 most recent configuration sessions have failed. Halting session creation for 960 seconds
2021-09-16 10:00:35,565 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 10:00:45,689 - INFO    - batcher.batch - Successfully submited 1 batches for configuration
2021-09-16 10:02:26,775 - INFO    - batcher.batch - 2 batches/sessions have completed
2021-09-16 10:02:26,775 - INFO    - batcher.batch - A session has succeeded.  Resuming normal operations
```

CFS has implemented a crash loop back off style behavior to avoid creating an infinite number of failed configuration sessions

If the last 20 CFS session have failed, then it will pause increasing intervals to allow the problems to be corrected

WRITE ANSIBLE CODE FOR CFS

- CFS uses Ansible for configuration management
  - Create a configuration with one or more layers within a specific VCS git repository, and commit it to be executed by Ansible
  - Target a node, boot image, or group of nodes to apply the configuration
  - Create a configuration session to apply and track the status of Ansible, applying each configuration layer to the targets specified in the session metadata
- VCS is populated during software installation with Ansible code to configure each product
- Customers can write their own Ansible plays and roles to augment CFS configuration or implement new features
  - Ansible playbook best practices
    - https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html
  - Ansible Examples
    - https://github.com/ansible/ansible-examples

ANSIBLE – TERMS

- Playbook
  - One or more plays
- Play
  - Maps groups of hosts to tasks
- Task
  - Sequence of actions performed against group of hosts that match a pattern in the play
- Modules
  - Large Ansible library of common code
    - Manage basic system resources
    - Send notifications
- Roles
  - Abstraction for naming a group of things that perform same function
- Separate code from data
  - Jinja2 templates (code)
  - Variables (data)
- Jinja2
  - Python-based template engine
  - Templates have placeholders for parameter values which can be replaced with variables
- Data
  - Facts
    - Automatically available
    - Discovered at run time
  - Variables
    - User-defined

ANSIBLE CODE STRUCTURE

- Each repository directory matches Ansible documentation
  - https://docs.ansible.com/ansible/2.9/user_guide/playbooks_best_practices.html#content-organization
  - The default playbook site.yml is found at the top level, if it exists
  - Ansible roles and variables are in their appropriately named directories
  - Inventory directories like `group_vars` and `host_vars` may exist, but they are empty and left for variable overrides and customizations as needed by the customer

```
group_vars/
  group1.yml # here we assign variables to particular groups
  group2.yml
host_vars/
  hostname1.yml # here we assign variables to particular nodes
  hostname2.yml
site.yml # master playbook
roles/
  common/ # this hierarchy represents a "role"
    tasks/ #
      main.yml # <-- tasks file can include smaller files if warranted
    handlers/ #
      main.yml # <-- handlers file
    templates/ # <-- files for use with the template resource
      ntp.conf.j2 # <------- templates end in .j2
    files/ #
      bar.txt # <-- files for use with the copy resource
      foo.sh # <-- script files for use with the script resource
    vars/ #
      main.yml # <-- variables associated with this role
    defaults/ #
      main.yml # <-- default lower priority variables for this role
    meta/ #
      main.yml # <-- role dependencies
    library/ # roles can also include custom modules
    module_utils/ # roles can also include custom module_utils
    lookup_plugins/ # or other types of plugins, like lookup in this case
  fooapp/ # "" same kind of structure as "common" was above but for fooapp
```

ANSIBLE – BEST PRACTICES FOR PLAYBOOKS/ROLES

- Ansible expects that all tasks are idempotent
  - (action performed only once, even if play is run more than once)
  - Care should be taken to ensure that tasks prescribe the desired state of the running system, making changes only when necessary
  - See “Resource Model” at https://docs.ansible.com/ansible/latest/reference_appendices/glossary.html
- When modifying files on a running system
  - Keep in mind that other services may access the file
  - Take the appropriate measures to ensure the modifications do not interfere with other operations
  - Leave a breadcrumb that the file is updated by an automated process
    - The “insertbefore” or “insertafter” options in the Ansible “lineinfile” module are well-suited to help with this
- If you find that you are trying to do something that is difficult to achieve in a few simple steps
  - It is likely that Ansible already has a module that provides the functionality
  - Use existing Ansible modules rather than calling shell commands or scripts

WRITE PLAYBOOKS FOR MULTIPLE NODE TYPES

- Ansible playbook can designate which node groups the various tasks and roles will run against
  - This is designated using the `hosts` parameter
  - Users can create additional sections that target other node types, or adjust the hosts that the included roles will run against
  - Can target multiple groups within a section of a playbook or specify complex targets, such as nodes that are in one group and not in another group
    - https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html#common-patterns
  - Hosts can be in more than one group at a time if there are user-defined groups
    - Ansible will run all sections that match the node type against the node

DYNAMIC CFS INVENTORY

- Dynamic inventory generates Ansible hosts file with data from HSM
  - Can target an HSM group

```
ncn# cray hsm groups list --format json | jq .[].label
”blue"
”green"
```

  - Can target HSM-reported hardware roles and sub-roles
    - “Compute”, “Management”, “Application”
    - “Application_UAN”, “Management_Worker”, other Application subroles for the system
    - Consult the cray-hms-base-config Kubernetes ConfigMap in the services namespace for a listing of the available roles and sub-roles on the system
  - During a CFS session, the dynamic inventory is generated and placed in the hosts/01-cfs-generated.yaml file

STATIC CFS INVENTORY

- Static inventory can target specific groups of nodes
  - Good for testing configuration changes on a small scale in a configuration repository

```
ncn# mkdir -p hosts; cd hosts; cat > static <<EOF
[test_nodes]
x3000c0s25b0n0
EOF
ncn# cd ..; git add hosts/static
ncn# git commit -m "Added a single node to static inventory for test_nodes"
ncn# git push
```

  - The process can be used to include any nodes in the system reachable over the Node Management Network (NMN), which contains the public SSH key pair provisioned by the install process
  - This inventory information will only be located in the repository to which it is added
    - If the desired configuration contains multiple layers, use the additionalInventoryUrl option in CFS to provide inventory information on a per-session level instead of a per-repository level

IMAGE CUSTOMIZATION

- Use image customization to limit how many times a task is run and improve boot times
- Use image customization for configuration that is the same for all nodes of a type
  - Before CSM 1.3 (or for small playbooks or one-off testing with CSM 1.3)
    - Target a task to be run only when customizing image

      ```
      when: "{{ cray_cfs_image | default(false) }}"
      ```
    - Target a task to be run only on booted node during node personalization

      ```
      when: "{{ not cray_cfs_image | default(false) }}”
      ```
  - CSM 1.3 and later (better performance)
    - Use the cfs_image host group to distinguish between image customization and node personalization
      - Allows image customization to be identified in the hosts parameter
      - Removes the need to evaluate conditionals
      - Ensures that tasks are not accidentally running in both modes needlessly
- IMS image IDs are used as hosts and grouped according to input to the session creation

```
ncn# cray cfs sessions create --name example --configuration-name configurations-example
--target-definition image --target-group Compute IMS_IMAGE_ID
```

CFS PERFORMANCE AND SCALING TIPS

- Import roles rather than playbooks
  - Each time a new playbook starts, Ansible automatically gathers facts for all the systems it is running against
  - This is not necessary more than once and can slow down Ansible execution
- Turn off facts that are not needed in a playbook by setting `gather_facts: false`
  - If only a few facts are required, it is also possible to limit fact gathering by setting `gather_subset`
    - For more information on `gather_subset`, see https://docs.ansible.com/ansible/latest/modules/setup_module.html
  - Reducing fact gathering time is especially important when importing multiple playbooks from a top level playbook
    - Fact gathering will trigger for each imported playbook, potentially collecting the same information multiple times
- Use loops rather than individual tasks where modules are called multiple times
  - Some Ansible modules will optimize the command, such as grouping package installations into a single transaction https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

AVOID REPEATED CONDITIONALS WITH GROUP_BY

- The group_by and add_host modules can both be used to dynamically generate new hosts groups for the Ansible inventory
  - These modules prove useful when hosts can be grouped according to a common property
  - Then plays can be designed to only target that particular group.
    - Grouping by operating system, hardware type, or a hardware property such as the presence of a GPU
  - Ansible can then use these to skip roles and tasks more efficiently than if the when conditional is applied.
- group_by should be used when there are multiple named groups by which hosts can be grouped

```
-name: group by OS
  hosts: all
  tasks:
    - name: Classify hosts by OS
      group_by:
        key: os_{{ ansible_facts['distribution'] }}

- name: centOS playbook
  hosts: os_CentOS
  tasks:
   ...
```

AVOID REPEATED CONDITIONALS WITH ADD_HOST

- add_host is useful for cases where the property is true or false. It allows users to create a new group consisting of only the hosts where the property is true

```
- name: group by a sample variable
  hosts: all
  tasks:
    - name: Add all hosts where sample_var is true to the new Sample group
      add_host:
        name: '{{ inventory_hostname }}’
        groups: sample_group
      when: sample_var
- name: Sample playbook
  hosts: sample_group
  tasks:
  ...
```

- To target only a subset of a set of nodes, plays should use the following syntax
  - This play is targeting only nodes in the sample group that are also in the Compute nodes group
    - & takes the intersection of the Compute and sample_group groups

```
hosts: Compute:&sample_group
```

- To target a set of nodes except the ones in the new group, plays should use the following syntax
  - This play is targeting Compute nodes that are not a part of the sample group
    - ! negates the sample_group group, so that only Compute nodes that are not an image are targeted

```
hosts: Compute:!sample_group
```

AVOID REPEATED CONDITIONALS WITH INCLUDE_*

- Use include_* (dynamic re-use) to skip multiple tasks at once when using conditionals
  - Ansible evaluates conditionals for every node in every task
  - This includes when the conditional is applied to a block, or a role imported with roles: or the import_role task
    - This is because these are static imports that are compiled at the beginning of the playbook, and the conditional is inherited by every task in the role or block
    - Evaluating these conditionals for each task may only take a second or two, but across the hundreds of tasks that might be part of a playbook, this can add up to significant wasted time
  - Instead use dynamic imports with the include_* tasks
    - Because these are evaluated at runtime, a conditional can skip the import of the role or tasks entirely, and is only evaluated once
- See the Ansible documentation on Conditionals with re-use and Re-using files and roles for more information
  - Dynamic re-use is not possible when importing playbooks, so instead consider using group_by rather than a conditional static import

AVOID REPEATED CONDITIONALS WITH INCLUDE_* EXAMPLES

- BAD example: the role is imported statically and the when statement will be propagated down and evaluated for each task in the role
  - This wastes time by running the same check many times.

```
- name: Sample playbook
  hosts: all
  roles:
    - {role: sample_role, when: cray_cfs_image}
```

- GOOD example: the role should be imported dynamically so that the when conditional is only evaluated once

```
-name: Sample playbook
  hosts: all
  tasks:
  - include_role:
      role: sample_role
    when: cray_cfs_image
```

OTHER TIPS

- Use the included Ansible modules rather than making shell calls or running scripts
  - Ansible optimizes these and makes them flexible so the same module can be used for different systems
  - This will also improve the log output for debugging
- Use `loops` rather than individual tasks where modules are called multiple times
  - Some Ansible modules will optimize the command, such as grouping package installations into a single transaction
- Use Ansible retries for small, recoverable failures
  - CFS supports retries on a large scale, but it takes far more time for CFS to detect a failed component and spin up a new session than it does for Ansible to retry a task
- Do not use Ansible retries for failures that take a long time to recover from
  - Retrying for a significant amount of time on one node can hold up all the other successful nodes in a batch
  - If you cannot recover from a failure quickly, then let the node fail and CFS will separate it out from the successful nodes when new sessions are started
- Avoid `any_errors_fatal`
  - In addition to not working with all Ansible strategies, this can cause an Ansible run to exit early, and the nodes that did not have the error will have to start from the beginning of the playbook in the next session
- Design playbooks to be run with the `free` Ansible strategy
  - This means avoiding situations where all nodes in a batch need to complete a task before moving onto the next, and can save time by allowing nodes to proceed through a playbook at their own pace
- Avoid using the same CFS configuration/playbook for diverse node types
  - Ansible will skip sections of the playbook that have a `hosts` target that does not match any nodes in the current inventory/limit, but when multiple types of nodes are configured at the same time with the same configuration, they may end up in the same batch and Ansible run
  - This would mean that Ansible has to run through the sections for both types of nodes, taking more time than if the nodes were in separate batches and could skip past the unneeded code

ANSIBLE LIMITATIONS WITH CFS

- Because CFS splits components into multiple batches, and components may also configure at different times when they are rebooted, some keywords meant for coordinating the runs of multiple nodes may not work as expected.

Keyword	**Notes**
`any_errors_fatal`	This keyword is intended to stop execution as soon as any node reports a failure. However, this will only stop execution for the current CFS batch.
`run_once`	This keyword is intended to limit a task to running on a single node. However this will only cause the task to be run once per CFS batch.
`serial`	This keyword is intended to limit runs to a small number of nodes at a time, such as during a rolling upgrade. However, this will only function within the batch, so more nodes may be running the task than intended when multiple batches are running.

SELECTING AN ANSIBLE STRATEGY

- CFS supports two Ansible strategies
  - `cfs_linear` runs all task in a playbook serially, with all nodes completing a task before Ansible moves on to the next task
  - `cfs_free` decouples the nodes allowing each node to proceed through the playbook at its own pace
  - Switching to `cfs_free` from the default strategy of `cfs_linear` may result in better configuration time
    - Not all included playbooks currently support the `cfs_free` strategy, so this should only be done for playbooks that are confirmed to work correctly with the `cfs_free` strategy
    - In addition, the `cfs_free` strategy is limited by the fact that configuration in CFS is applied over multiple layers and multiple playbooks
      - This means that even when using the `cfs_free` strategy, all nodes must complete a playbook together before moving onto the next playbook
- The CFS Ansible strategies extend Ansible strategy
  - adding reporting callbacks that are used to track components' state
  - `cfs_linear` and `cfs_free` should always be used in place of `linear` and `free` to ensure that CFS functions correctly

ANSIBLE DEBUGGING

- Name tasks uniquely and use debug

```
tasks:
  - name: find nid match in external hosts file, capture IP address
    shell: “grep {{nid}} /etc/mysitelocal/hosts-external | head -1 | awk '{ print $4 }’”
    register: external_ipaddr
  - name: add ListenAddress/external options to file
    lineinfile:
      dest: /etc/sshd/sshd_config
      regexp="^SSHD_OPTS="
      line="SSHD_OPTS=‘-u0 -f /etc/ssh/sshd_config.external -o ListenAddress={{external_ipaddr}}’"
      backup: yes
    when:
      external_ipaddr is defined
  - debug: “Did not find external interface to start SSHD on...”
    when: external_ipaddr is not defined
```

TROUBLESHOOT ANSIBLE PLAY FAILURES IN CFS SESSIONS

- Find the CFS pod that is in an error state

```
ncn# kubectl get pods -n services | grep Error
    NAME                                            READY   STATUS   RESTARTS   AGE
    cfs-e8e48c2a-448f-4e6b-86fa-dae534b1702e-pnxmn  0/3     Error    0          25h
```

- Check to see what containers are in the pod

```
ncn# kubectl logs -n services $CFS_POD_NAME
    Error from server (BadRequest): a container name must be specified for pod cfs-e8e48c2a-448f-4e6b-86fa-dae534b1702e-pnxmn,
choose one of: [inventory ansible-0 istio-proxy] or one of the init containers: [git-clone-0 istio-init]
```

- Check the git-clone, inventory, ansible containers in that order

```
ncn# kubectl logs -n services CFS_POD_NAME git-clone
ncn# kubectl logs -n services CFS_POD_NAME inventory
        Sidecar available
        2019-12-05 15:00:12,160 - INFO    - cray.cfs.inventory - Starting CFS Inventory version=0.4.3, namespace=services
        2019-12-05 15:00:12,171 - INFO    - cray.cfs.inventory - Inventory target=dynamic for cfsession=boa-2878e4c0-39c2-4df0-989e-
053bb1edee0c
        2019-12-05 15:00:12,227 - INFO    - cray.cfs.inventory.dynamic - Dynamic inventory found a total of 2 groups
        2019-12-05 15:00:12,227 - INFO    - cray.cfs.inventory - Writing out the inventory to /inventory/hosts
ncn# kubectl logs -n services CFS_POD_NAME ansible
        Waiting for Inventory
        TASK [ncmp_hsn_cns : SLES Compute Nodes (HSN): Create/Update ifcfg-hsnx File(s)] ***
        fatal: [x3000c0s19b1n0]: FAILED! => {"msg": "'interfaces' is undefined"}
        fatal: [x3000c0s19b2n0]: FAILED! => {"msg": "'interfaces' is undefined"}
        NO MORE HOSTS LEFT *************************************************************
        PLAY RECAP *********************************************************************
        x3000c0s19b1n0             : ok=28   changed=20   unreachable=0    failed=1    skipped=77   rescued=0    ignored=1
        x3000c0s19b2n0             : ok=27   changed=19   unreachable=0    failed=1    skipped=63   rescued=0    ignored=1
```

ANSIBLE PROFILING

- Ansible tasks and playbooks can be profiled to determine execution times and identify poor runtime performance
  - Edit the default CFS Ansible.cfg

```
ncn# kubectl edit cm cfs-default-ansible-cfg -n services
```

  - Uncomment this line

```
#callback_whitelist    = cfs_aggregator, timer, profile_tasks, profile_roles
```

  - Comment this line by adding a # character to the beginning of the line

```
callback_whitelist = cfs_aggregator
```

  - New sessions will be created with profiling information available in the Ansible logs of the CFS session pods
- View end of CFS log to see PLAY RECAP

```
ncn# kubectl -n services --sort-by=.metadata.creationTimestamp get pods | grep cfs
ncn# kubectl logs -f -n services POD ansible
```

  - Find end of ansible log

```
2022-08-26T16:38:23 PLAY RECAP *********************************************************************
2022-08-26T16:38:23 x1203c2s6b1n1 : ok=23 changed=18 unreachable=0 failed=0 skipped=3 rescued=0 ignored=0
2022-08-26T16:38:23 x1203c3s0b0n1 : ok=8  changed=7  unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
2022-08-26T16:38:23 x1203c3s2b0n1 : ok=8  changed=7  unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

PROFILED ANSIBLE LOG

- After PLAY RECAP, ansible log shows profiling information by area

```
2022-08-26T16:38 ===============================================================================
2022-08-26T16:38 sma-ldms-compute ------------------------------------------------------ 756.74s
2022-08-26T16:38 gather_facts ------------------------------------------------------------ 2.35s
2022-08-26T16:38 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2022-08-26T16:38 total ----------------------------------------------------------------- 759.09s
2022-08-26T16:38 Friday 26 August 2022  22:38:23 +0000 (0:00:01.783)       0:12:39.142 *********
2022-08-26T16:38 ===============================================================================
2022-08-26T16:38 sma-ldms-compute : Install cray-ldms package -------------------------- 260.11s
2022-08-26T16:38 sma-ldms-compute : Restart LDMS on Computes --------------------------- 238.85s
2022-08-26T16:38 sma-ldms-compute : Copy LDMS config files from ansible environment to compute
nodes - 113.49s
2022-08-26T16:38 sma-ldms-compute : Add SMA zypper repositories ------------------------- 68.21s
2022-08-26T16:38 sma-ldms-compute : Delete SMA LDMS PVC pod ----------------------------- 38.93s
2022-08-26T16:38 sma-ldms-compute : Fetch files from worker node ------------------------ 11.14s
```

- Example is from the 22.07 software recipe release with SMA 1.6 and helped to focus attention to tasks and roles which were running slower than expected
  - Don't install rpms in post-boot CFS

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

MONITORING TOOLS

- Monitoring
  - System Monitoring Framework
  - LDMS
  - Telemetry API
  - SMA-Grafana
  - Dashboards
    - Drilling into dashboards
- LDMS extension
  - LDMS plugins
- Monasca alarms and notifications

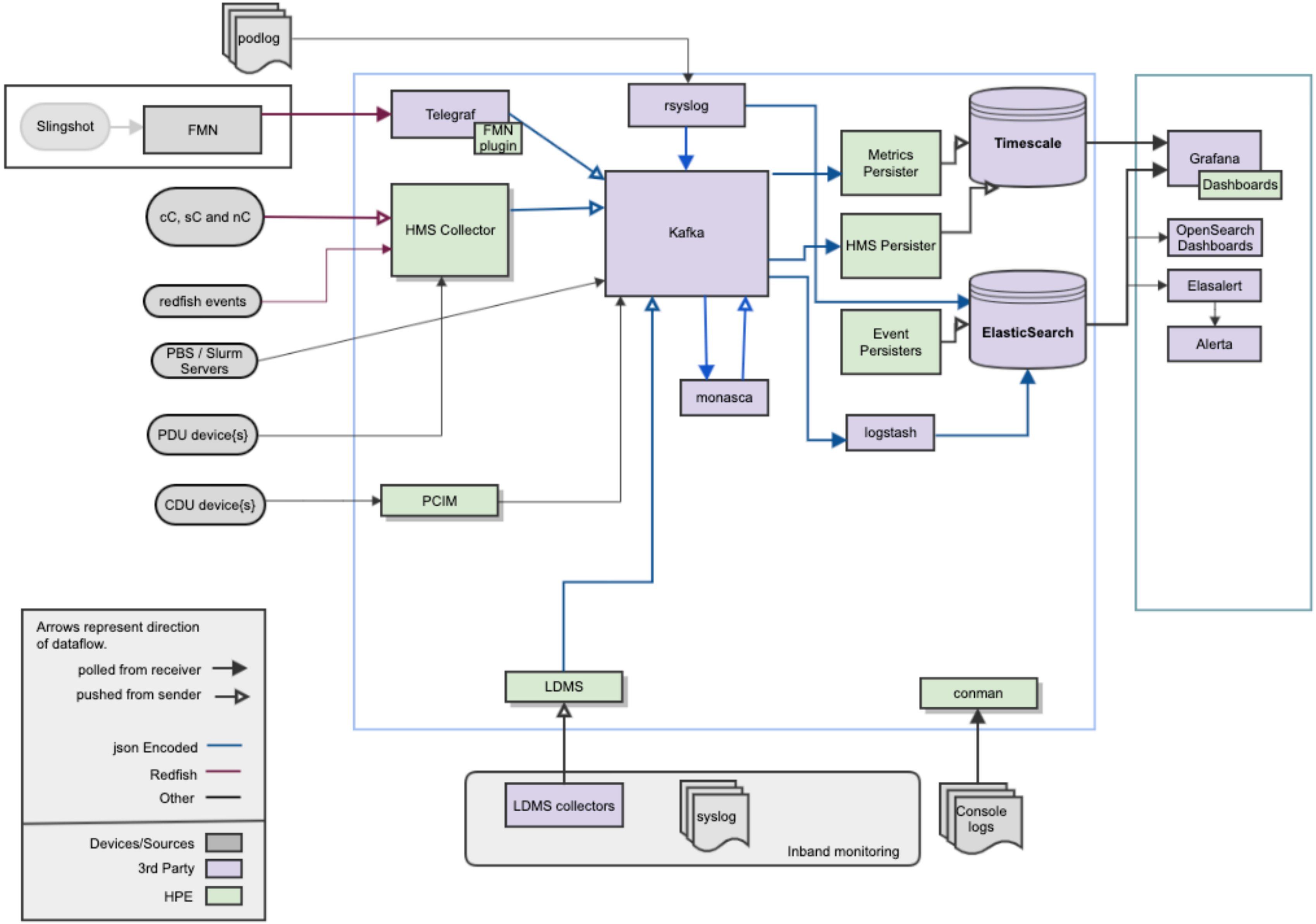
MONITORING

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

SYSTEM MONITORING FRAMEWORK

- Tightly-integrated monitoring system
- Provides detailed telemetry information from multiple subsystems:
  - Fabric
  - Environmental
  - Network
  - Storage
  - Operating systems (vmstat and iostat metrics)
- Incorporates the context necessary to understand telemetry data
- Feeds into a common message bus (Kafka), persistence, and minimal UI infrastructure
- 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 several pre-defined alarms
  - Can be extended with site defined alarms

SYSTEM MONITORING FRAMEWORK



Arrows represent direction of dataflow.

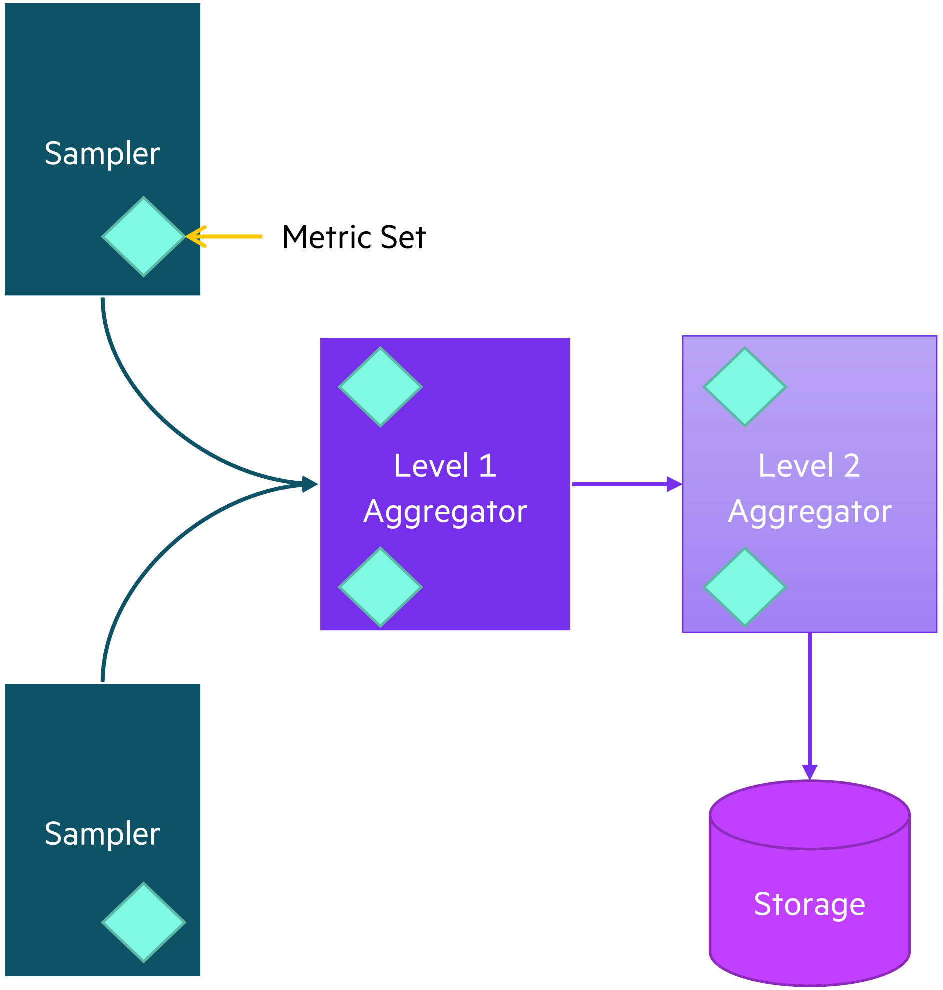
polled from receiver →
pushed from sender ⇾

json Encoded
Redfish
Other

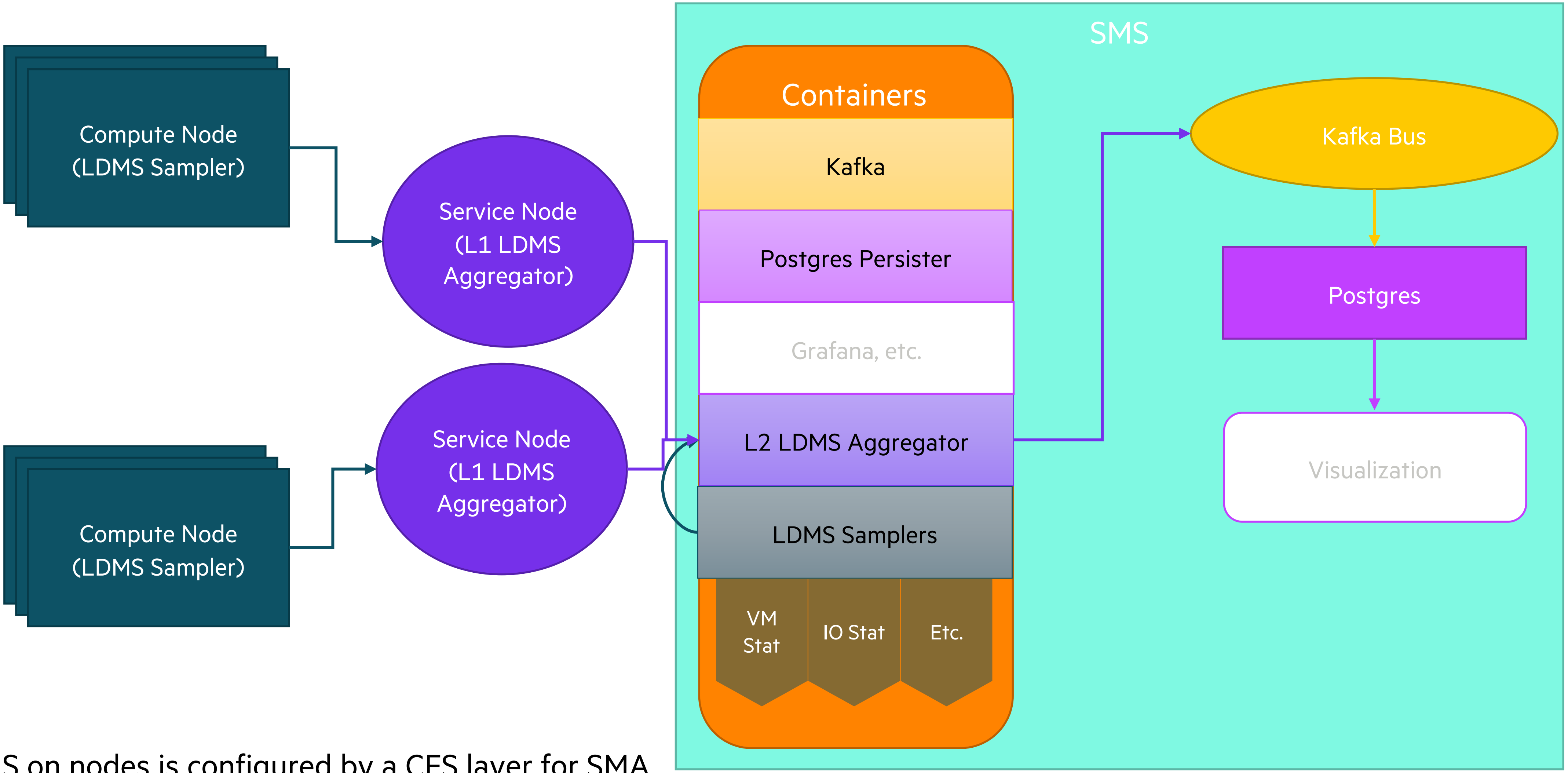
Devices/Sources ⬛
3rd Party ▨
HPE 🟩

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 MySQL 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

SMA-GRAFANA

- The HPE Cray EX system uses a Grafana web UI to provide system metric monitoring of:
  - LDMS statistics
  - 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 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:

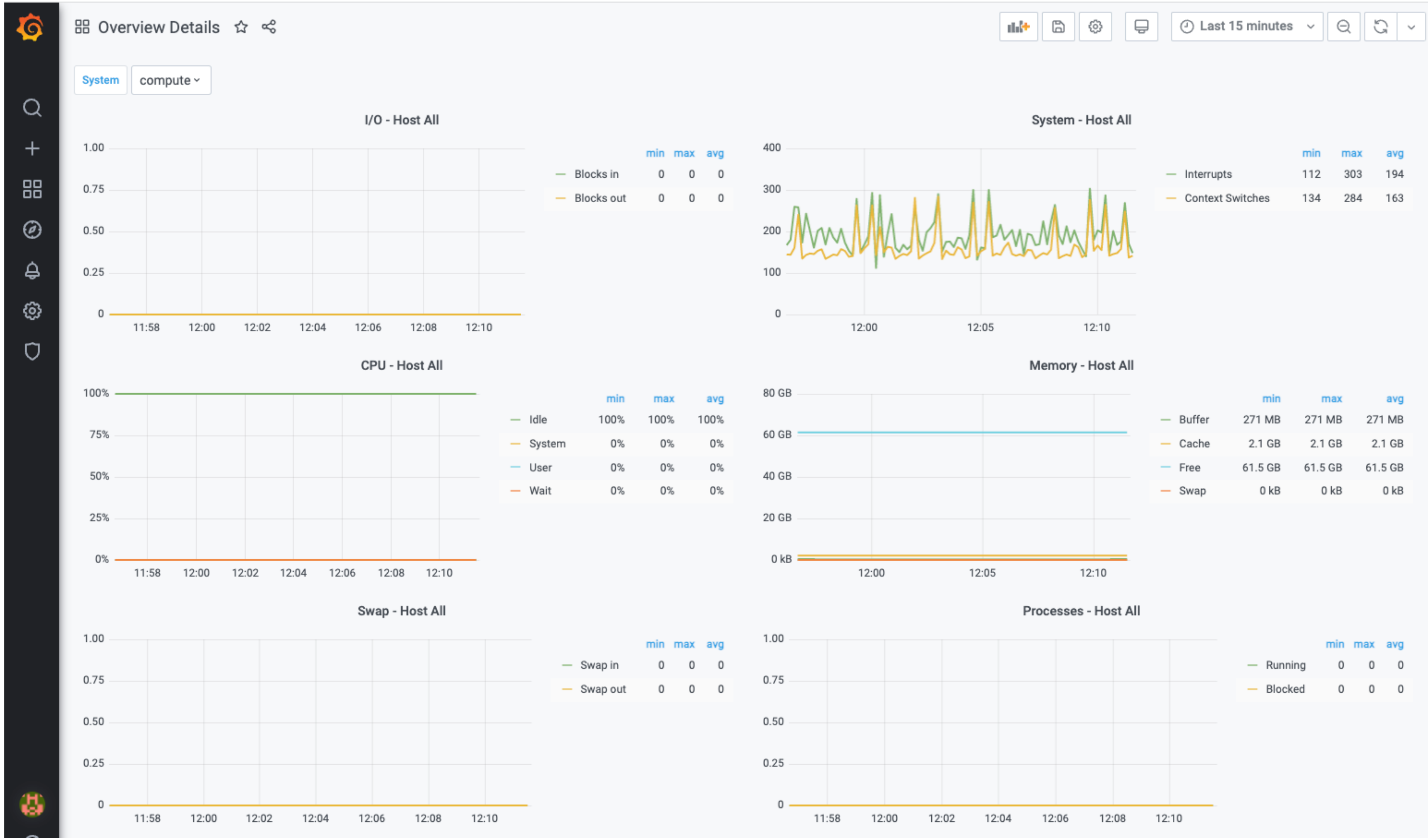
```
ncn-m001# kubectl get secret site-init -n loftsman \
-o jsonpath='{.data.customizations\.yaml}' | base64 -d | grep "external:"
external: SYSTEM_DOMAIN_NAME
```

  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



I/O - Host All

	min	max	avg
Blocks in	0	0	0
Blocks out	0	0	0

System - Host All

	min	max	avg
Interrupts	112	303	194
Context Switches	134	284	163

CPU - Host All

	min	max	avg
Idle	100%	100%	100%
System	0%	0%	0%
User	0%	0%	0%
Wait	0%	0%	0%

Memory - Host All

	min	max	avg
Buffer	271 MB	271 MB	271 MB
Cache	2.1 GB	2.1 GB	2.1 GB
Free	61.5 GB	61.5 GB	61.5 GB
Swap	0 kB	0 kB	0 kB

Swap - Host All

	min	max	avg
Swap in	0	0	0
Swap out	0	0	0

Processes - Host All

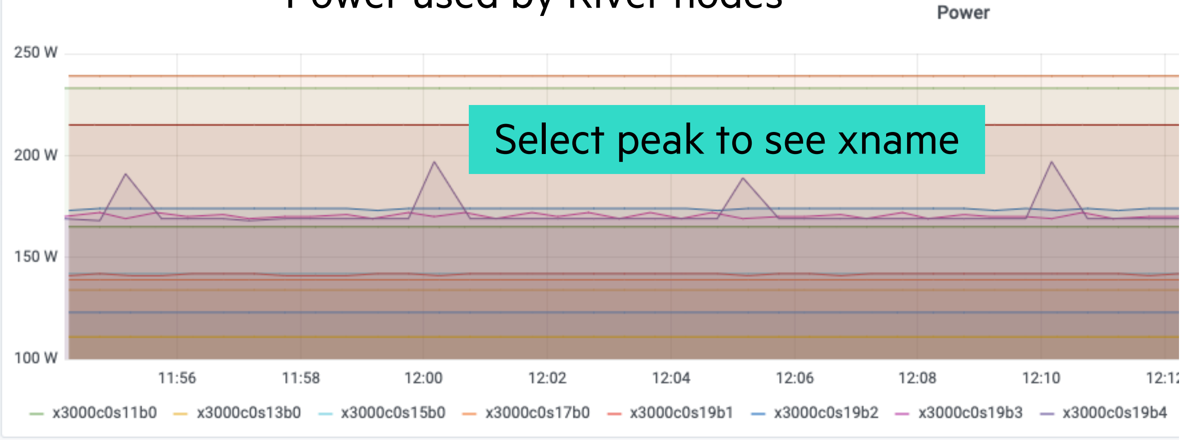
	min	max	avg
Running	0	0	0
Blocked	0	0	0

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)

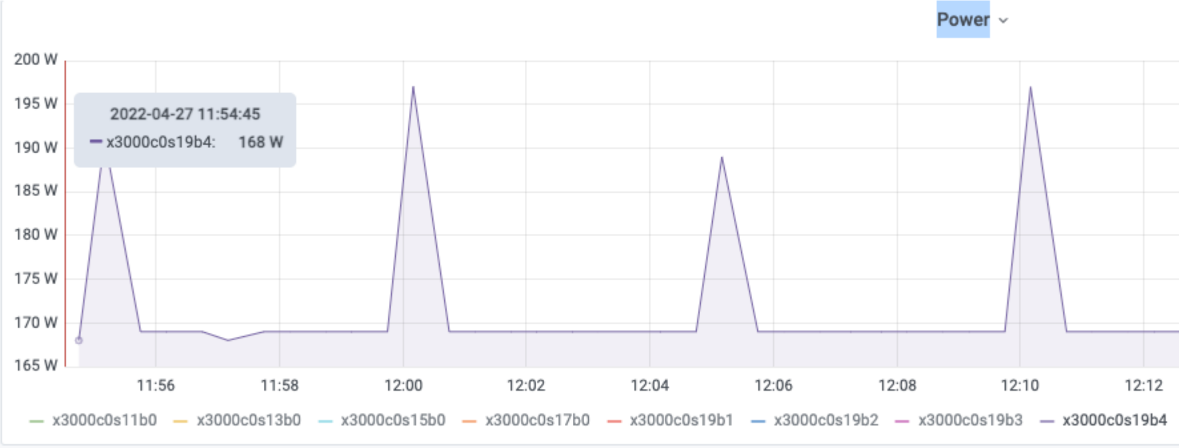
https://sma-grafana.cmn.SYSTEM_DOMAIN_NAME/dashboards

Power used by River nodes

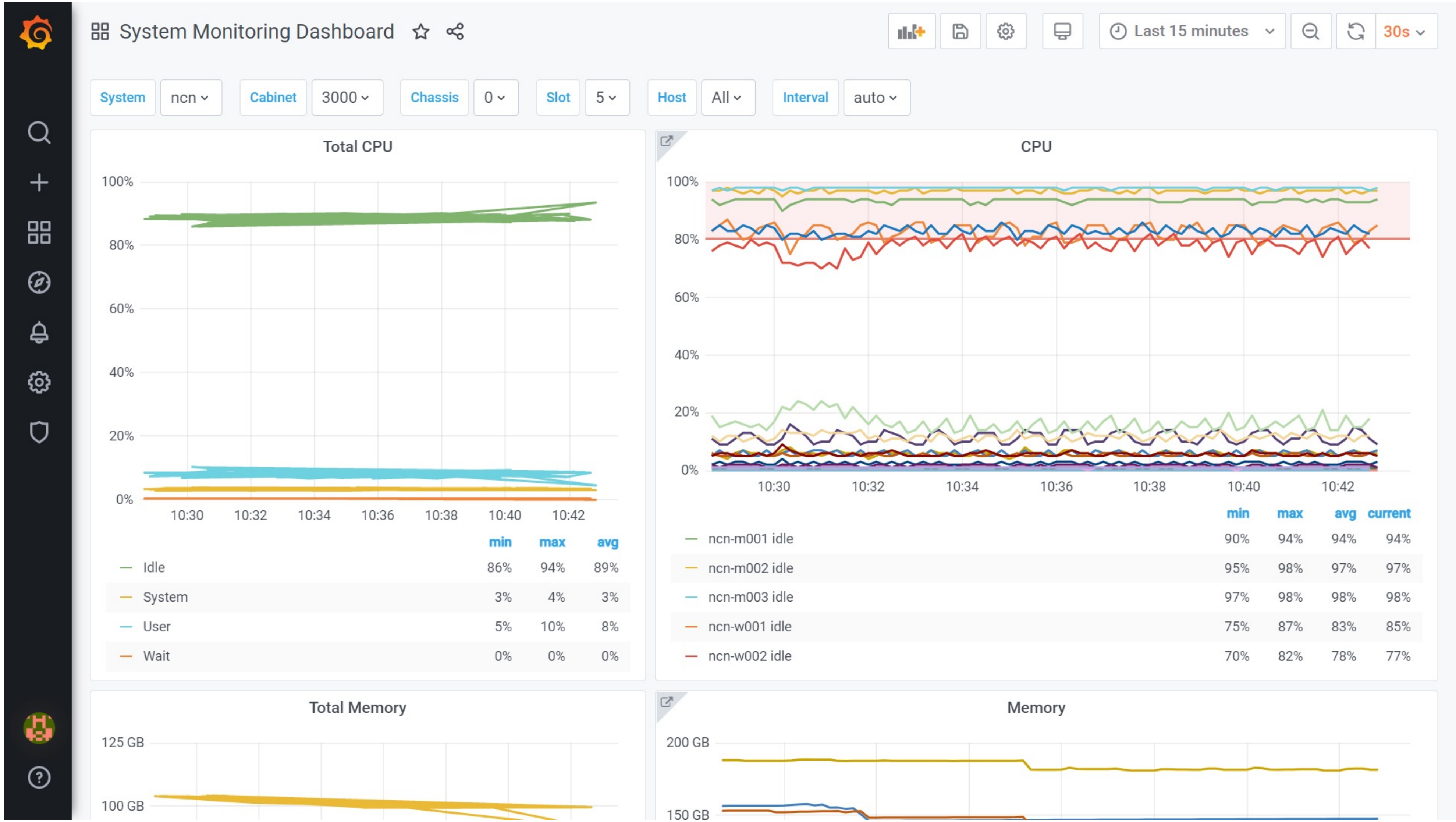


Select peak to see xname

Click on xname to drill into that node



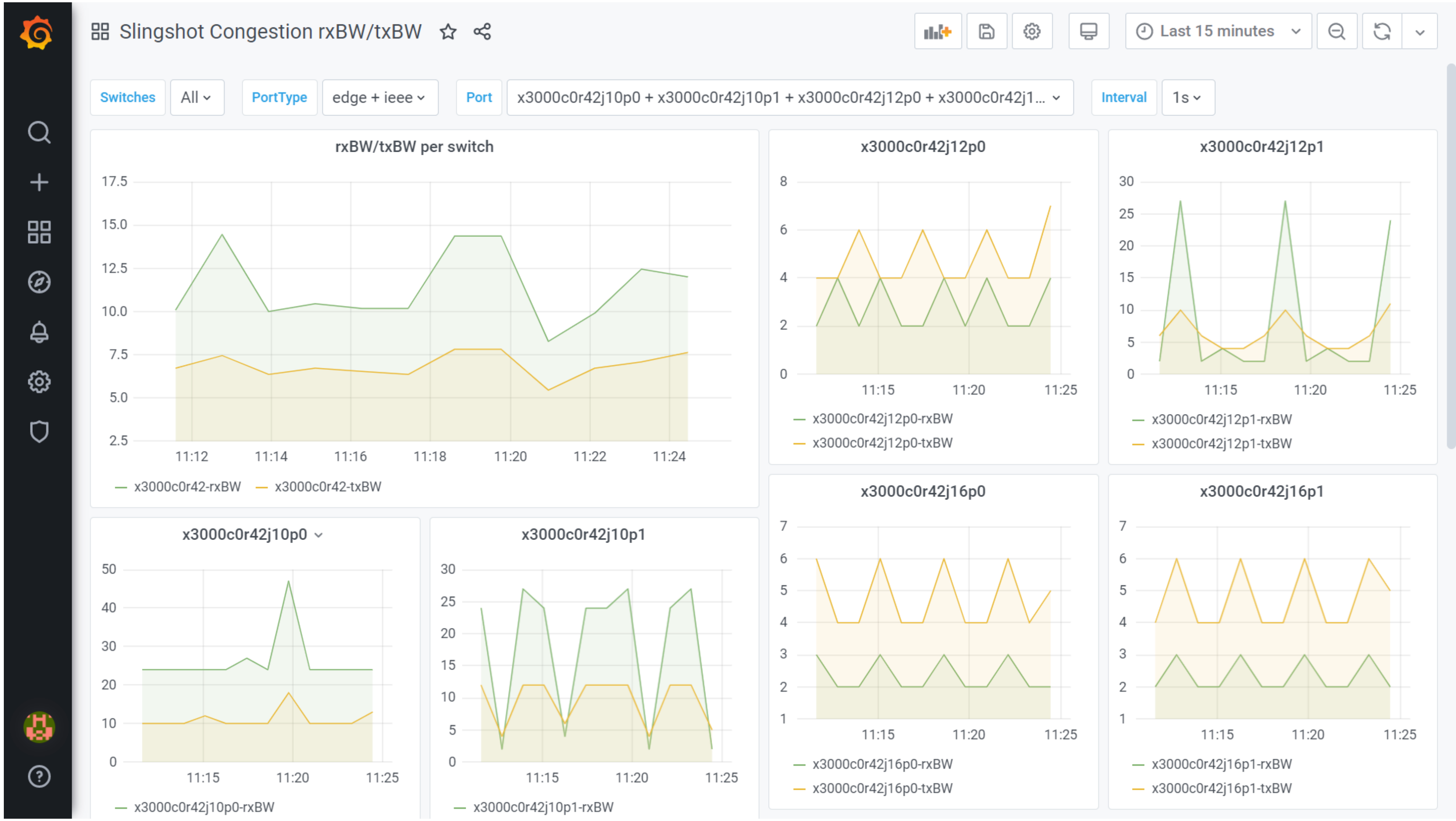
SMA-GRAFANA SYSTEM MONITORING DASHBOARD



	min	max	avg
— Idle	86%	94%	89%
— System	3%	4%	3%
— User	5%	10%	8%
— Wait	0%	0%	0%

	min	max	avg	current
— ncn-m001 idle	90%	94%	94%	94%
— ncn-m002 idle	95%	98%	97%	97%
— ncn-m003 idle	97%	98%	98%	98%
— ncn-w001 idle	75%	87%	83%	85%
— ncn-w002 idle	70%	82%	78%	77%

SMA-GRAFANA SLINGSHOT CONGESTION RECEIVE/TRANSMIT BANDWIDTH



Slingshot Congestion rxBW/txBW

Last 15 minutes

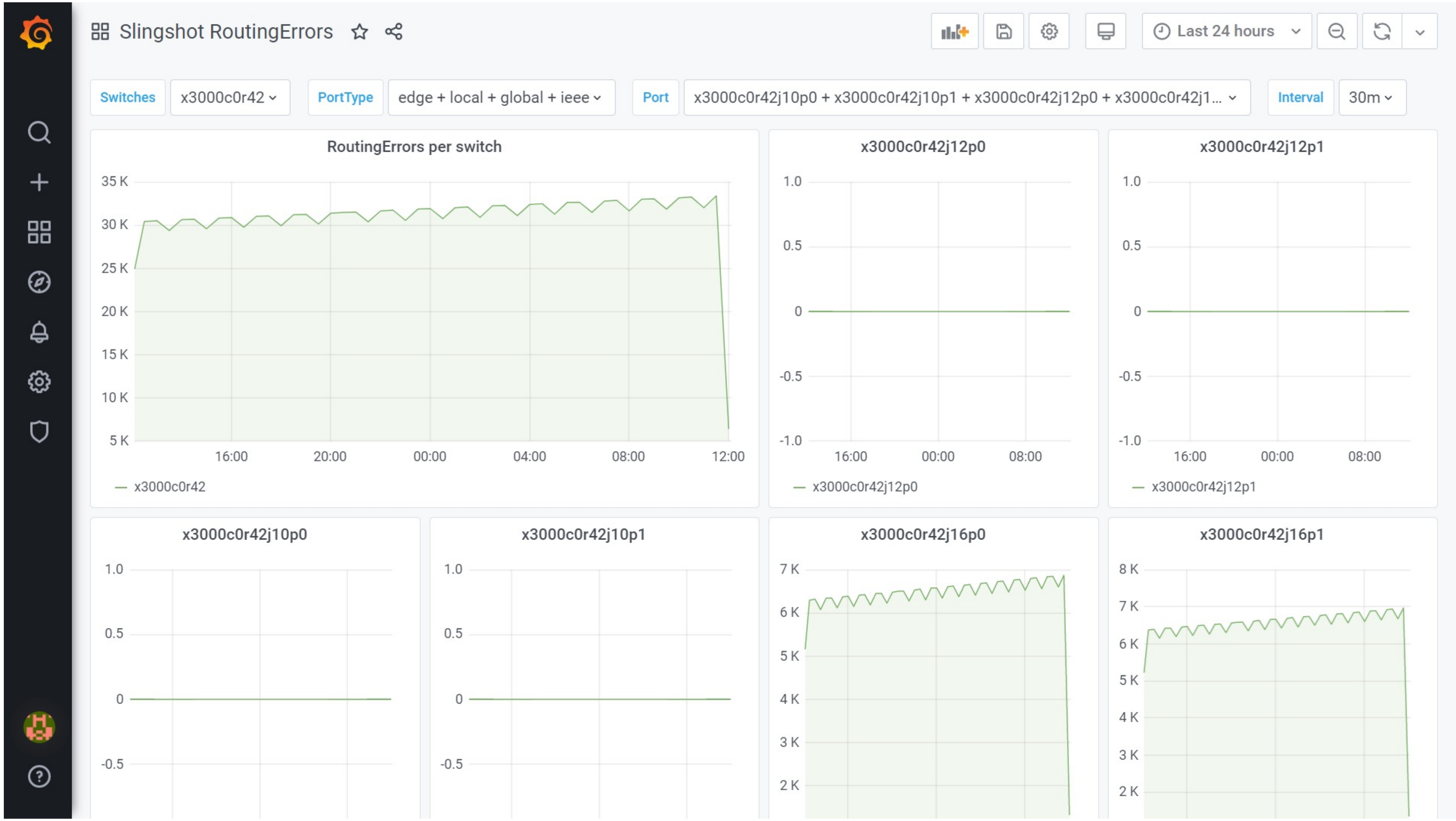
Switches: All

PortType: edge + ieee

Port: x3000c0r42j10p0 + x3000c0r42j10p1 + x3000c0r42j12p0 + x3000c0r42j1...

Interval: 1s

SMA-GRAFANA SLINGSHOT ROUTING ERRORS



Slingshot RoutingErrors

Last 24 hours

Switches: x3000c0r42

PortType: edge + local + global + ieee

Port: x3000c0r42j10p0 + x3000c0r42j10p1 + x3000c0r42j12p0 + x3000c0r42j1…

Interval: 30m

SMA-GRAFANA SWITCH CONTROLLER SENSORS

SMA-GRAFANA CABINET CONTROLLER SENSORS

SMA-GRAFANA NODE CONTROLLER SENSORS

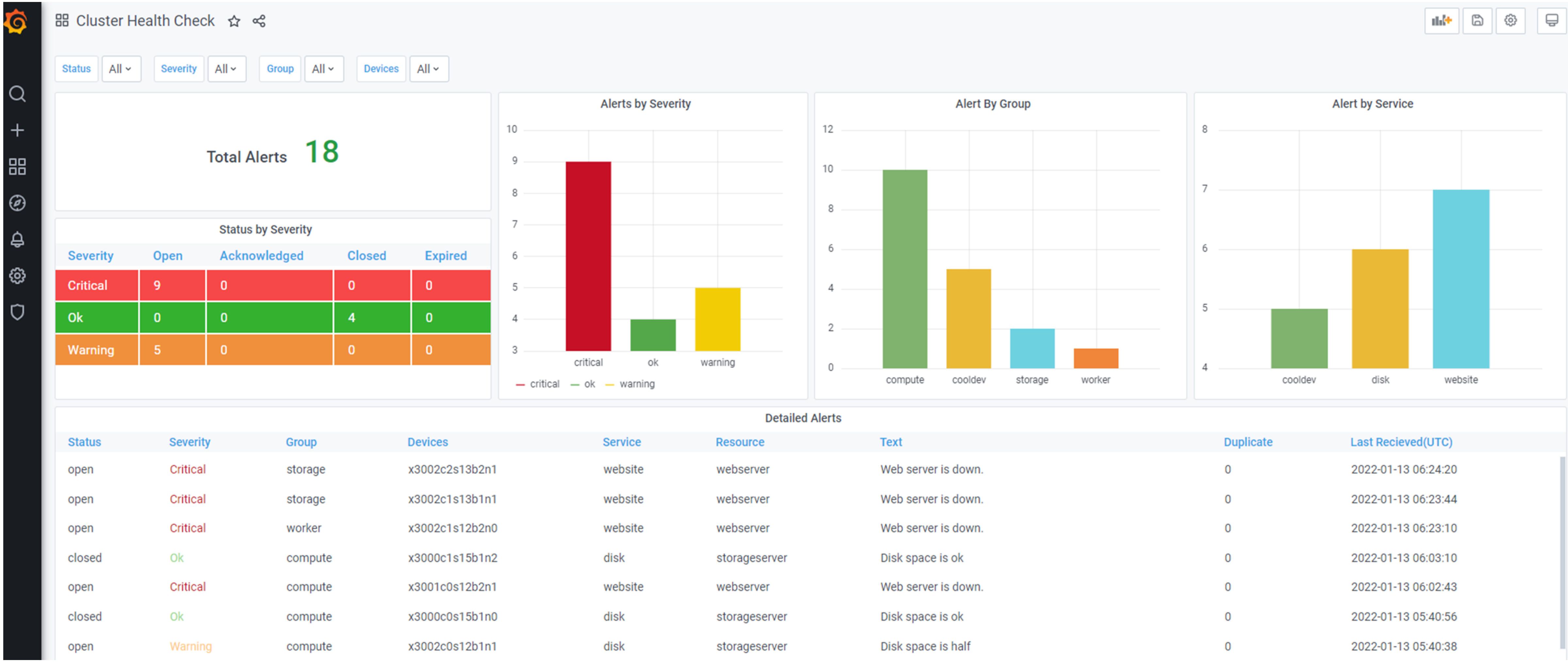

Interval auto

Locations x9000c1s0b1 + x9000c1s0b1n0 + x9000c1s0b1n1 + x9000c1s2b0 + x90...

Last 30 minutes

10s

CLUSTER HEALTH CHECK



Total Alerts 18

Status by Severity

Severity	Open	Acknowledged	Closed	Expired
Critical	9	0	0	0
Ok	0	0	4	0
Warning	5	0	0	0

Detailed Alerts

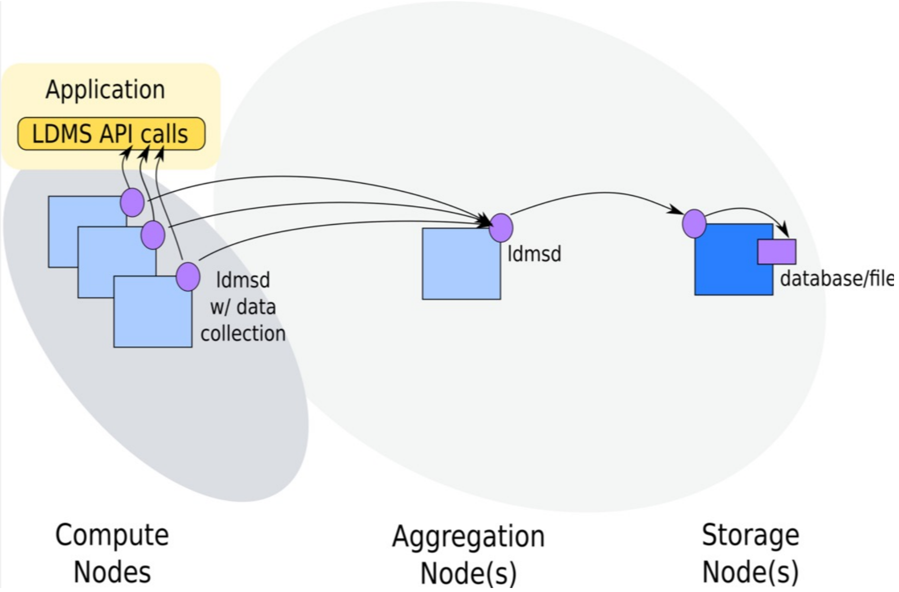
Status	Severity	Group	Devices	Service	Resource	Text	Duplicate	Last Recieved(UTC)
open	Critical	storage	x3002c2s13b2n1	website	webserver	Web server is down.	0	2022-01-13 06:24:20
open	Critical	storage	x3002c1s13b1n1	website	webserver	Web server is down.	0	2022-01-13 06:23:44
open	Critical	worker	x3002c1s12b2n0	website	webserver	Web server is down.	0	2022-01-13 06:23:10
closed	Ok	compute	x3000c1s15b1n2	disk	storageserver	Disk space is ok	0	2022-01-13 06:03:10
open	Critical	compute	x3001c0s12b2n1	website	webserver	Web server is down.	0	2022-01-13 06:02:43
closed	Ok	compute	x3000c0s15b1n0	disk	storageserver	Disk space is ok	0	2022-01-13 05:40:56
open	Warning	compute	x3002c0s12b1n1	disk	storageserver	Disk space is half	0	2022-01-13 05:40:38

LDMS EXTENSION

- LDMS plugins

LIGHTWEIGHT DISTRIBUTED METRIC SERVICE (LDMS)

- OVIS is a modular system for HPC data collection, transport, storage, log message exploration, and visualization as well as analysis
- LDMS is a low-overhead, low-latency framework for collecting, transfering, and storing metric data on a large distributed computer system
- The framework includes:
  - a public API with a reference implementation
  - tools for collecting, aggregating, transporting, and storing metric values
  - collectors for several common types of metrics
  - Data transport over socket, RDMA (IB/iWarp/RoCE), and Cray Gemini as well as Aries
- The API provides a way for vendors to expose system information in a uniform manner without being required to provide source code for accessing the information (although we advise it be included) which might reveal proprietary methods or information
- Metric information can be updated by a kernel module which runs only when applications yield the processor and transported using RDMA-like operations, resulting in minimal jitter during collection



LDMS PLUGINS

- SMA Administration guide
  - Add Customer Provided Samplers to LDMS v4 Configuration
    - Download the ovis-4 repo

      ```
      git clone https://github.com/ovis-hpc/ovis.git -b OVIS-4
      ```
    - Add dependencies
    - Make a directory for the new sampler
    - Create a Makefile
    - Change Configure script to include new makefile
    - Run autogen to build new sampler library files
    - Update `sma-ldms-map-pvc` pod with new sampler library files
    - Modify LDMS configuration to use new sampler libraries on NCNs and Compute nodes
    - Edit Compute and NCN LDMS files to include new samplers and add them to the PVCs
    - Restart `sma-ldms-map-pvc` pod
    - Customize compute images and reboot compute nodes
    - Restart LDMS aggregator pods
      - `sma-ldms-aggr-compute-0` and `sma-ldms-aggr-ncn-0`
- If there are problems, the new sampler can be removed using documented procedure

MONASCA ALARMS AND NOTIFCATIONS

- Email notifications
- Local alarms
- CLI for alarms
- Monasca tuning
- mon-alert

EMAIL NOTIFICATIONS

- SMA monitors metric data that is transmitted on the main telemetry bus
  - Provides a way to notify users when select metric data is outside of normal operating values
  - Includes several pre-defined alarms
- SMA configmap `sma-monasca-alarms-configdata-cm`
  - email_destination
  - sendmail_server
  - email_source

```
ncn# kubectl -n sma edit cm sma-monasca-alarms-configdata-cm
ncn# kubectl -n sma describe cm sma-monasca-alarms-configdata-cm
```

- Changes require deleting pods and job and creating new job

```
ncn# kubectl -n sma delete pod -l component=notification
ncn# kubectl -n sma delete job -l component=alarms-init-job
ncn# kubectl -n sma delete pod -l component=alarms-init-job
ncn# vi alarms-init-job.yaml
ncn# kubectl -n sma apply -f alarms-init-job.yaml
```

LOCAL ALARMS

- Local alarms can be created that send email notifications
  - Create local alarm definitions

```
ncn# vi customer-alarms-configmap.yaml
```

  - Deploy configmap

```
ncn-# kubectl -n sma apply -f customer-alarms-configmap.yaml
```

  - Create job definition

```
ncn# vi customer-alarms-init-job.yaml
```

  - Execute the SMA alarm initialization job

```
ncn# kubectl -n sma apply -f customer-alarms-init-job.yaml
```

  - Verify job succeeds

```
ncn# kubectl -n sma get po -l component=customer-alarms-init-job
NAME READY STATUS RESTARTS AGE
customer-alarms-init-job-dtrw5 0/1 Completed 0 5m
```

CLI FOR ALARMS

- List the state of all defined alarms

```
ncn# kubectl -n sma exec -it sma-monasca-agent-p9vcb -c collector -- sh -c 'monasca alarm-list'
+--------------------------------------+--------------------------------------+------------------+
| id                                   | alarm_definition_id                  | alarm_definition_name |
+--------------------------------------+--------------------------------------+------------------+
| 0881af14-5659-4468-813a-d99ac7f415c5 | cd72e681-995c-4f0a-9d29-c6a0a0e0dde8 | validation1Alarm |
| 64bbb62d-3cb1-466c-bed3-e7012f742683 | cd72e681-995c-4f0a-9d29-c6a0a0e0dde8 | validation1Alarm |
| b571cb91-2e1f-486e-9dc7-8b1e112cb530 | cd72e681-995c-4f0a-9d29-c6a0a0e0dde8 | validation1Alarm |
+--------------------------------------+--------------------------------------+------------------+
```

- List all defined alarms

```
ncn# kubectl -n sma exec -it sma-monasca-agent-p9vcb -c collector -- sh -c 'monasca alarm-definition-list'
+--------------------+--------------------------------------+--------------------------------+
| name               | id                                   | expression                     |
+--------------------+--------------------------------------+--------------------------------+
| metricsTestAlarm   | 7c101b85-0da9-48d8-a930-5f751645ca16 | avg(cray_test.other_test) < 20 |
| validation1Alarm   | cd72e681-995c-4f0a-9d29-c6a0a0e0dde8 | last(kubelet.health_status) < 0|
| vmstatTestAlarm    | f0c54a42-ba99-4d1b-a048-96400b55fbee | avg(cray_test.vmstat_test) < 20|
| SMA OST Free Files | f6b03e39-4529-4ae2-9e22-358b40aeea52 | avg(cray_storage.free_files_perc, 900) < 5.0|
| lustreTestAlarm    | fa7bdbe3-b53e-4bdc-bc5c-1bf1d780d4f3 | avg(cray_test.lustre_test) < 20|
+--------------------+--------------------------------------+--------------------------------+
```

- List all defined notifications

```
ncn# kubectl -n sma exec -it sma-monasca-agent-p9vcb -c collector -- sh -c 'monasca notification-list'
+----------------+--------------------------------------+---------+
| name           | id                                   | type    |
+----------------+--------------------------------------+---------+
| defaultWebhook | 7ad0fb59-3178-4947-9c1b-e340b1348176 | WEBHOOK |
| defaultEmail   | a3f71b75-0e3c-4dbc-8651-ef02ea3616e9 | EMAIL   |
+----------------+--------------------------------------+---------+
```

MONASCA TUNING

- Monasca service
  - Default memory 1216MB
  - Java heap size 870MB
- Under heavy load, OOM may happen

```
ncn# kubectl describe -n sma-monasca-thresh-dmtf
o.a.s.d.worker [ERROR] Error when processing event java.lang.OutOfMemoryError: Java heap space
```

- Change the configuration values in the sma-monasca section of customizations.yaml for permanent change

```
sma-monasca:
  thresh:
    maxHeapMB: ”990"
    resources:
      limits:
        memory: ”1600"Mi
```

- Change on running system

```
ncn# kubectl -n sma edit deployment sma-monasca-thresh-dmtf
- env:
  - name: MAX_HEAP_MB
   value: ”990"
...
resources:
  limits:
    memory: ”1600"Mi
```

MON-ALERT

- Mon-alert manages the life cycle of each alert
  - Looks for events in the data
  - Analyzes each event
  - Alerts the user
  - Stores the event in the alert dashboard
  - Retrieve alerts, process alerts, and close alerts
  - Disable alert mechanism during maintenance
- Display a summary of all alerts

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

MON-ALERT STATUS

- Display a status summary of all alerts

```
ncn# mon-alert status
METRIC.                     TYPE.  NAME                         VALUE   AVERAGE
--------------------------- ------ ---------------------------- ------- ---------
Total alerts                gauge  alerts.total                 33
Alert status change.        timer  alerts.status                4       26
Received alerts             timer  alerts.received              6594    25.7413
Count alerts                timer  alerts.counts                229     19.9607
Alert queries               timer  alerts.queries               1301.   21.6257
Deleted alerts.             timer  alerts.deleted               6303    26.3121
Alerta console auto-refresh text   switch.auto-refresh-allow    ON
API alert submission        text   switch.sender-api-allow      ON
```

- Display the most serious alerts

```
ncn# 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 on
```

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

MON-ALERT MANAGEMENT

- Query all alerts

```
ncn# mon-alert query [-f text=~criteria]
ncn# mon-alert query
ID STATUS SEVERITY GROUP ENV SERVICE RESOURCE EVENT VALUE DESCRIPTION DUPL LAST RECEIVED
-------- ------- --------- ------ ------------------------------------ ------
1d0d93c1 open warning fabric http://10.33.0.170:8000/fabric/agents/x3000c0r21b0 slingshot
```

  - More data with wide display
- Manage specific alert
  - Display single alert

```
ncn# mon-alert query -i 1d0d93c1
```

  - Acknowledge alert

```
ncn# mon-alert ack -i 1d0d93c1
```

  - Remedy the problem
  - Close alert

```
ncn# mon-alert close -i 1d0d93c1
```

  - Delete alert

```
ncn# mon-alert delete -i 1d0d93c1
```

MON-ALERT MAINTENANCE

- Clean up the alerts manually
  - This command looks for expired alerts and deletes the expired alerts

```
ncn# mon-alert housekeeping
```

- Add user comments to an alert

```
ncn# mon-alert tag -h
```

- Delete user comments from an alert

```
ncn# mon-alert untag -h
```

- Suspend alerting for a maintenance period
- If you need to perform maintenance, you can suppress the alerting mechanism during that time
  - Create blackout period

```
ncn# mon-alert blackout -h
```

  - Display blackout periods

```
ncn# mon-alert blackouts -h
```

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

SYSTEM MANAGEMENT HEALTH

- System health
  - Prometheus
  - Alertmanager
  - Grafana
  - Dashboards
- Slingshot network (HSN)
- System testing
  - CSM health checks
  - CSM diags

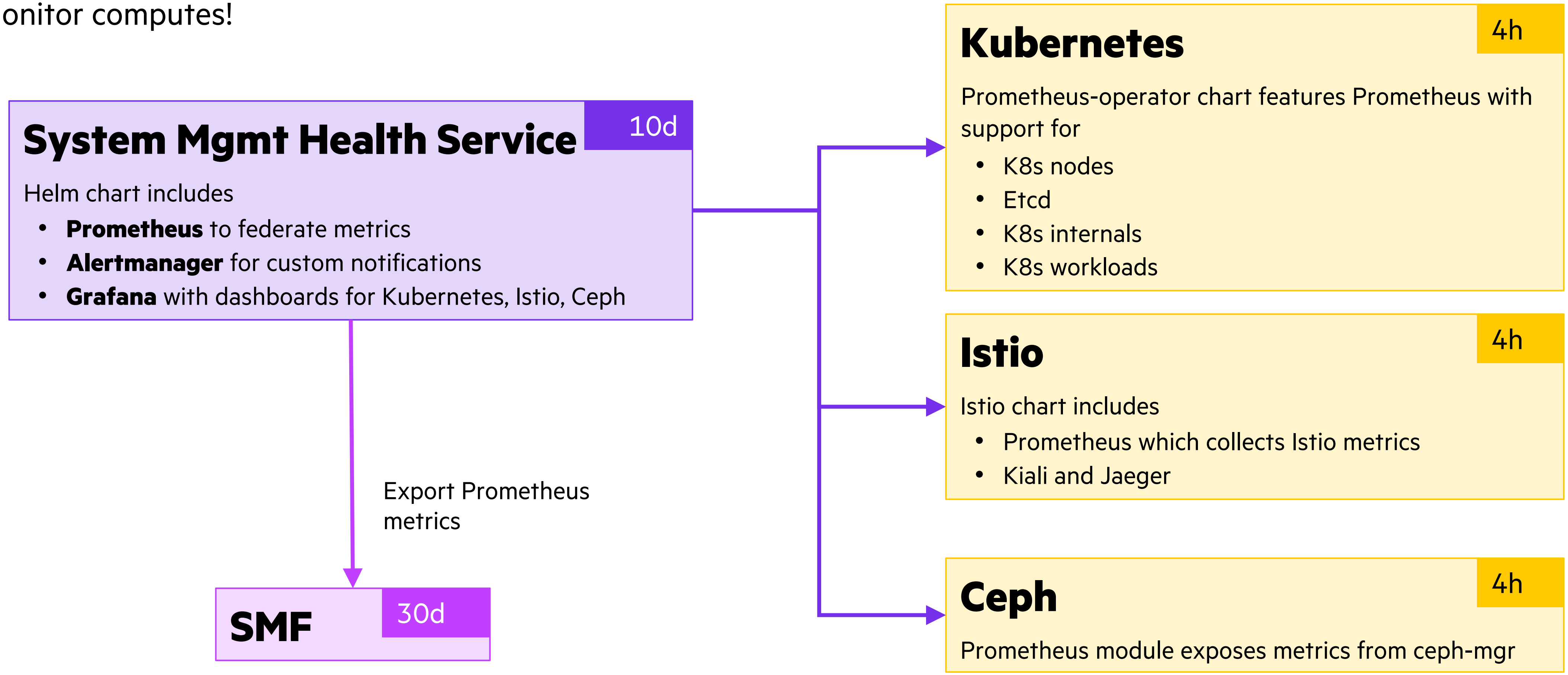
SYSTEM HEALTH

- Prometheus
- Alertmanager
- Grafana
- Dashboards

SYSTEM MANAGEMENT HEALTH SERVICE

Is the system healthy?

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



System Mgmt Health Service 10d

Helm chart includes

- **Prometheus** to federate metrics
- **Alertmanager** for custom notifications
- **Grafana** with dashboards for Kubernetes, Istio, Ceph

Export Prometheus metrics

SMF 30d

Kubernetes 4h

Prometheus-operator chart features Prometheus with support for

- K8s nodes
- Etcd
- K8s internals
- K8s workloads

Istio 4h

Istio chart includes

- Prometheus which collects Istio metrics
- Kiali and Jaeger

Ceph 4h

Prometheus module exposes metrics from ceph-mgr

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

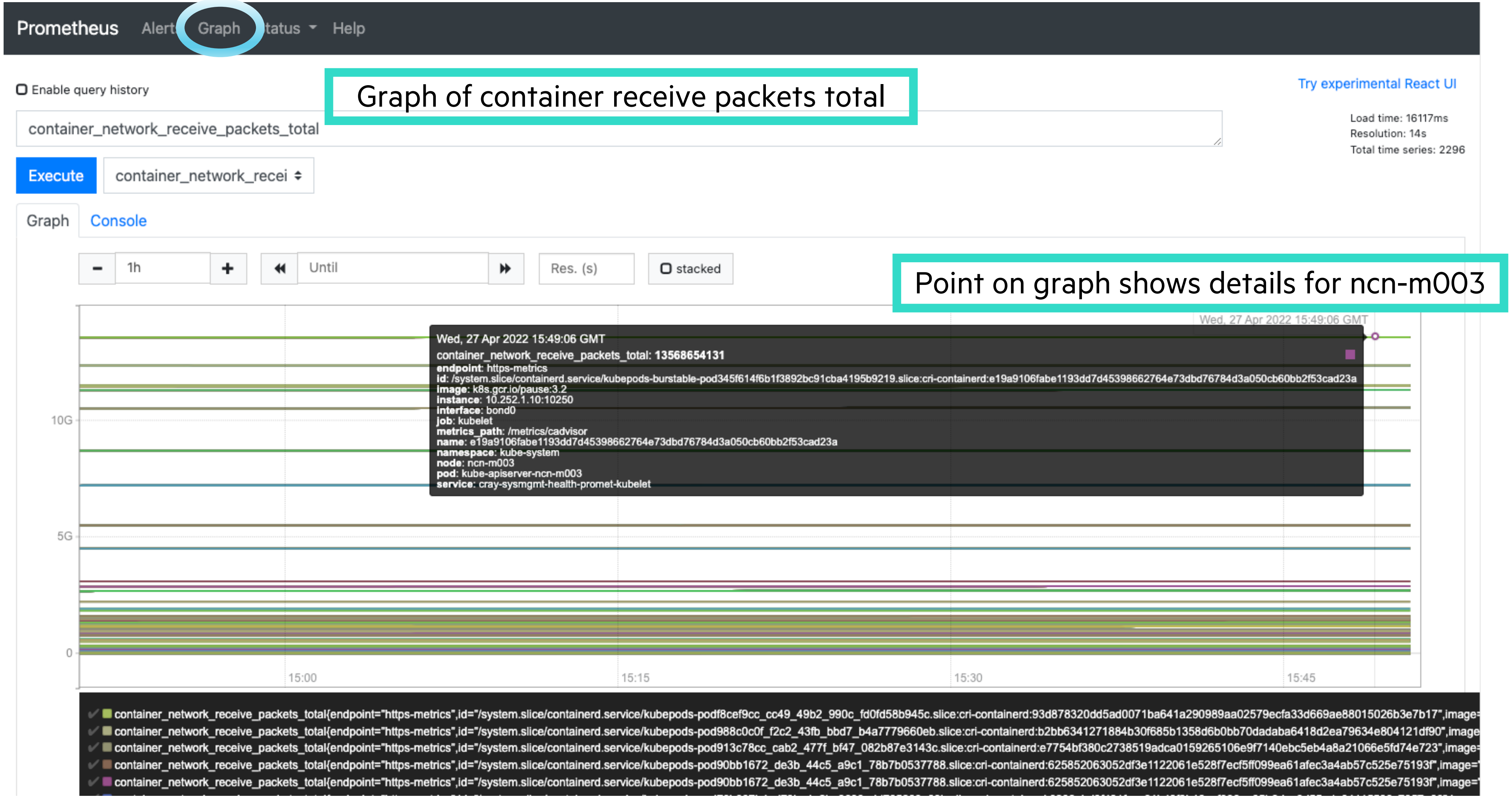
RETRIEVING ALERTS FROM PROMETHEUS

```
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



Graph of container receive packets total

Point on graph shows details for ncn-m003

container_network_receive_packets_total

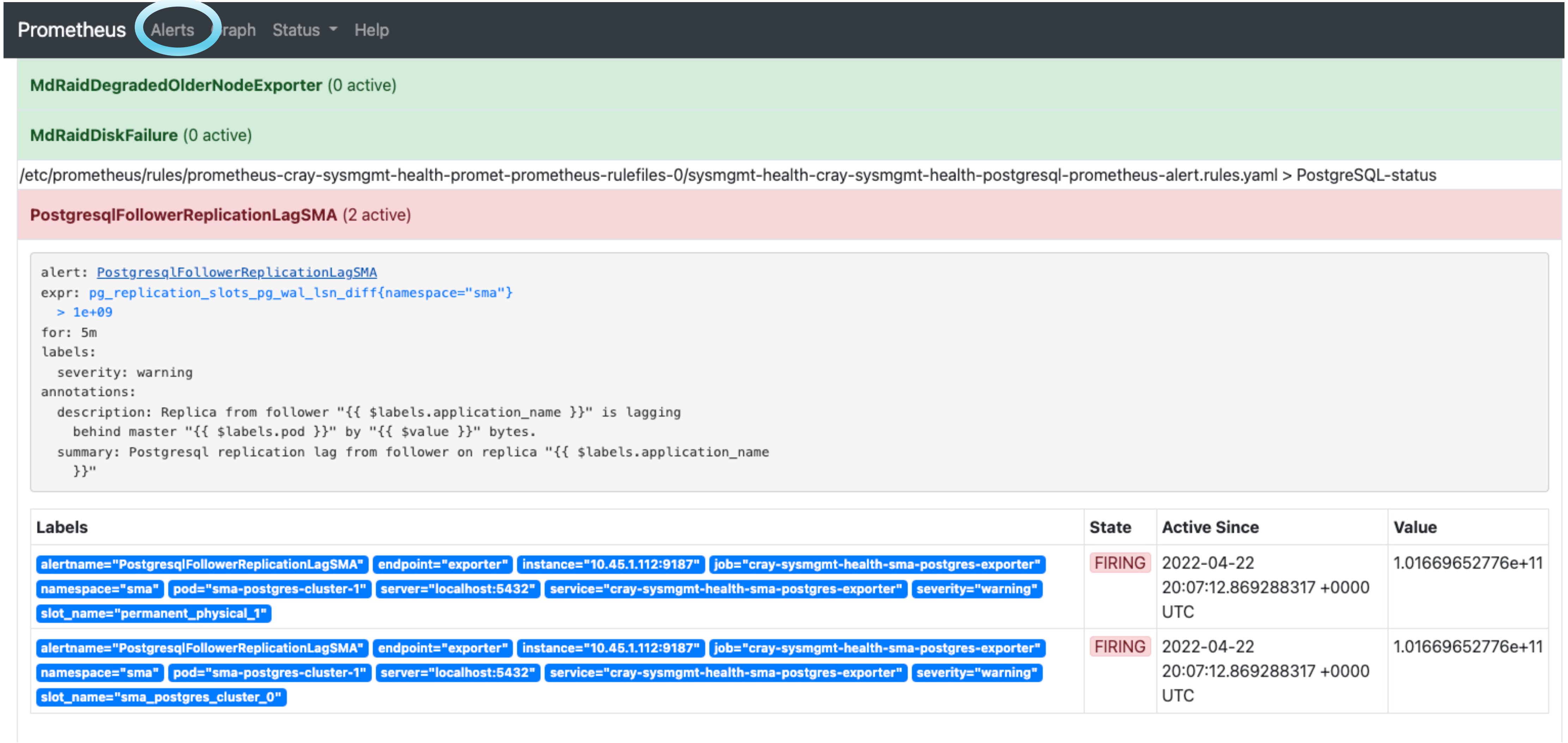
Load time: 16117ms
Resolution: 14s
Total time series: 2296

```
Wed, 27 Apr 2022 15:49:06 GMT
container_network_receive_packets_total: 13568654131
endpoint: https-metrics
id: /system.slice/containerd.service/kubepods-burstable-pod345f614f6b1f3892bc91cba4195b9219.slice:cri-containerd:e19a9106fabe1193dd7d45398662764e73dbd76784d3a050cb60bb2f53cad23a
image: k8s.gcr.io/pause:3.2
instance: 10.252.1.10:10250
interface: bond0
job: kubelet
metrics_path: /metrics/cadvisor
name: e19a9106fabe1193dd7d45398662764e73dbd76784d3a050cb60bb2f53cad23a
namespace: kube-system
node: ncn-m003
pod: kube-apiserver-ncn-m003
service: cray-sysmgmt-health-promet-kubelet
```

```
container_network_receive_packets_total{endpoint="https-metrics",id="/system.slice/containerd.service/kubepods-podf8cef9cc_cc49_49b2_990c_fd0fd58b945c.slice:cri-containerd:93d878320dd5ad0071ba641a290989aa02579ecfa33d669ae88015026b3e7b17",image=
container_network_receive_packets_total{endpoint="https-metrics",id="/system.slice/containerd.service/kubepods-pod988c0c0f_f2c2_43fb_bbd7_b4a7779660eb.slice:cri-containerd:b2bb6341271884b30f685b1358d6b0bb70dadaba6418d2ea79634e804121df90",image
container_network_receive_packets_total{endpoint="https-metrics",id="/system.slice/containerd.service/kubepods-pod913c78cc_cab2_477f_bf47_082b87e3143c.slice:cri-containerd:e7754bf380c2738519adca0159265106e9f7140ebc5eb4a8a21066e5fd74e723",image=
container_network_receive_packets_total{endpoint="https-metrics",id="/system.slice/containerd.service/kubepods-pod90bb1672_de3b_44c5_a9c1_78b7b0537788.slice:cri-containerd:625852063052df3e1122061e528f7ecf5ff099ea61afec3a4ab57c525e75193f",image="
container_network_receive_packets_total{endpoint="https-metrics",id="/system.slice/containerd.service/kubepods-pod90bb1672_de3b_44c5_a9c1_78b7b0537788.slice:cri-containerd:625852063052df3e1122061e528f7ecf5ff099ea61afec3a4ab57c525e75193f",image="
```

https://prometheus.cmn.SYSTEM_DOMAIN_NAME

PROMETHEUS - ALERTS



MdRaidDegradedOlderNodeExporter (0 active)

MdRaidDiskFailure (0 active)

/etc/prometheus/rules/prometheus-cray-sysmgmt-health-promet-prometheus-rulefiles-0/sysmgmt-health-cray-sysmgmt-health-postgresql-prometheus-alert.rules.yaml > PostgreSQL-status

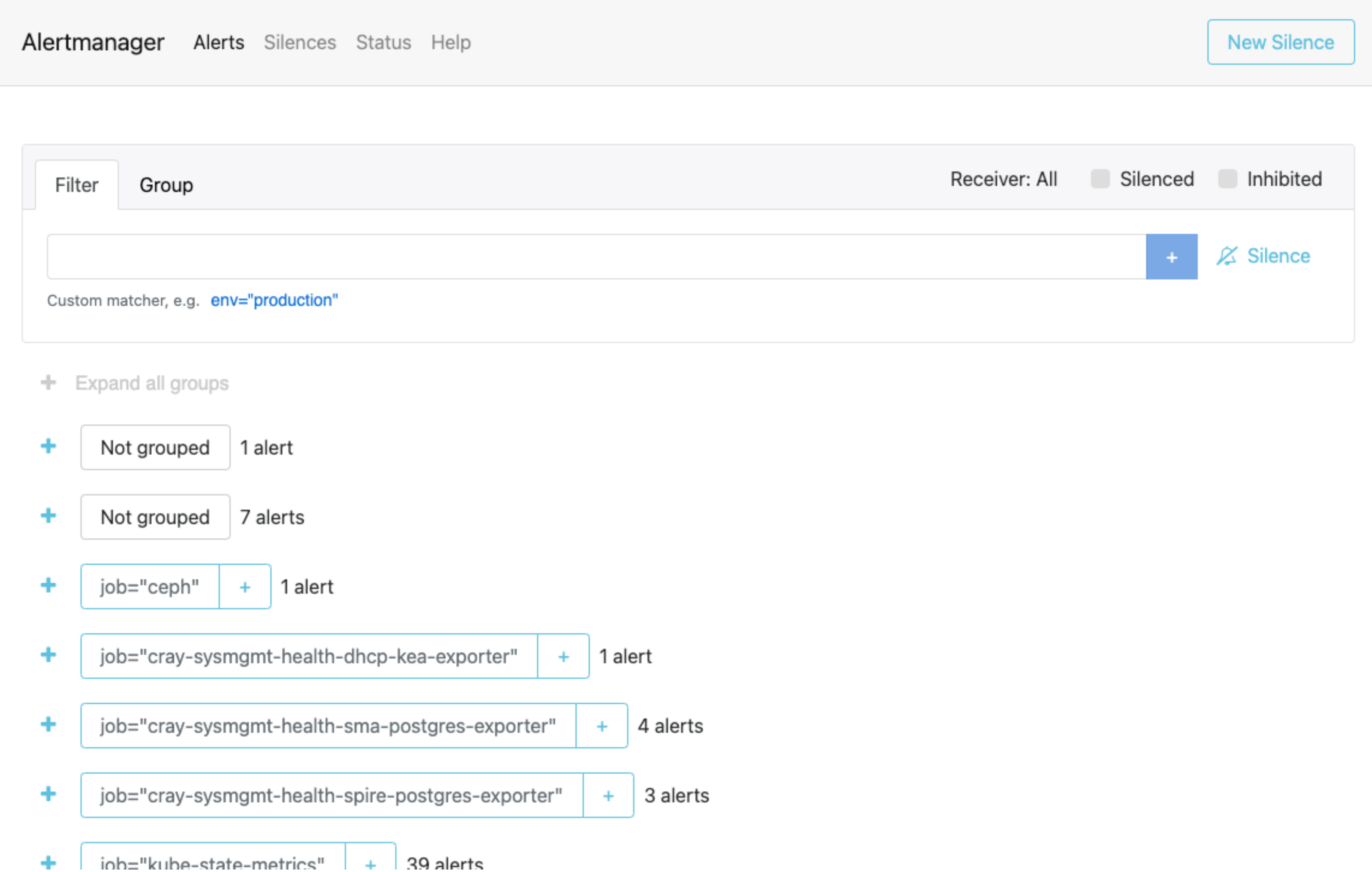
PostgresqlFollowerReplicationLagSMA (2 active)

```
alert: PostgresqlFollowerReplicationLagSMA
expr: pg_replication_slots_pg_wal_lsn_diff{namespace="sma"}
  > 1e+09
for: 5m
labels:
  severity: warning
annotations:
  description: Replica from follower "{{ $labels.application_name }}" is lagging
    behind master "{{ $labels.pod }}" by "{{ $value }}" bytes.
  summary: Postgresql replication lag from follower on replica "{{ $labels.application_name
    }}"
```

Labels	State	Active Since	Value
alertname="PostgresqlFollowerReplicationLagSMA" endpoint="exporter" instance="10.45.1.112:9187" job="cray-sysmgmt-health-sma-postgres-exporter" namespace="sma" pod="sma-postgres-cluster-1" server="localhost:5432" service="cray-sysmgmt-health-sma-postgres-exporter" severity="warning" slot_name="permanent_physical_1"	FIRING	2022-04-22 20:07:12.869288317 +0000 UTC	1.01669652776e+11
alertname="PostgresqlFollowerReplicationLagSMA" endpoint="exporter" instance="10.45.1.112:9187" job="cray-sysmgmt-health-sma-postgres-exporter" namespace="sma" pod="sma-postgres-cluster-1" server="localhost:5432" service="cray-sysmgmt-health-sma-postgres-exporter" severity="warning" slot_name="sma_postgres_cluster_0"	FIRING	2022-04-22 20:07:12.869288317 +0000 UTC	1.01669652776e+11

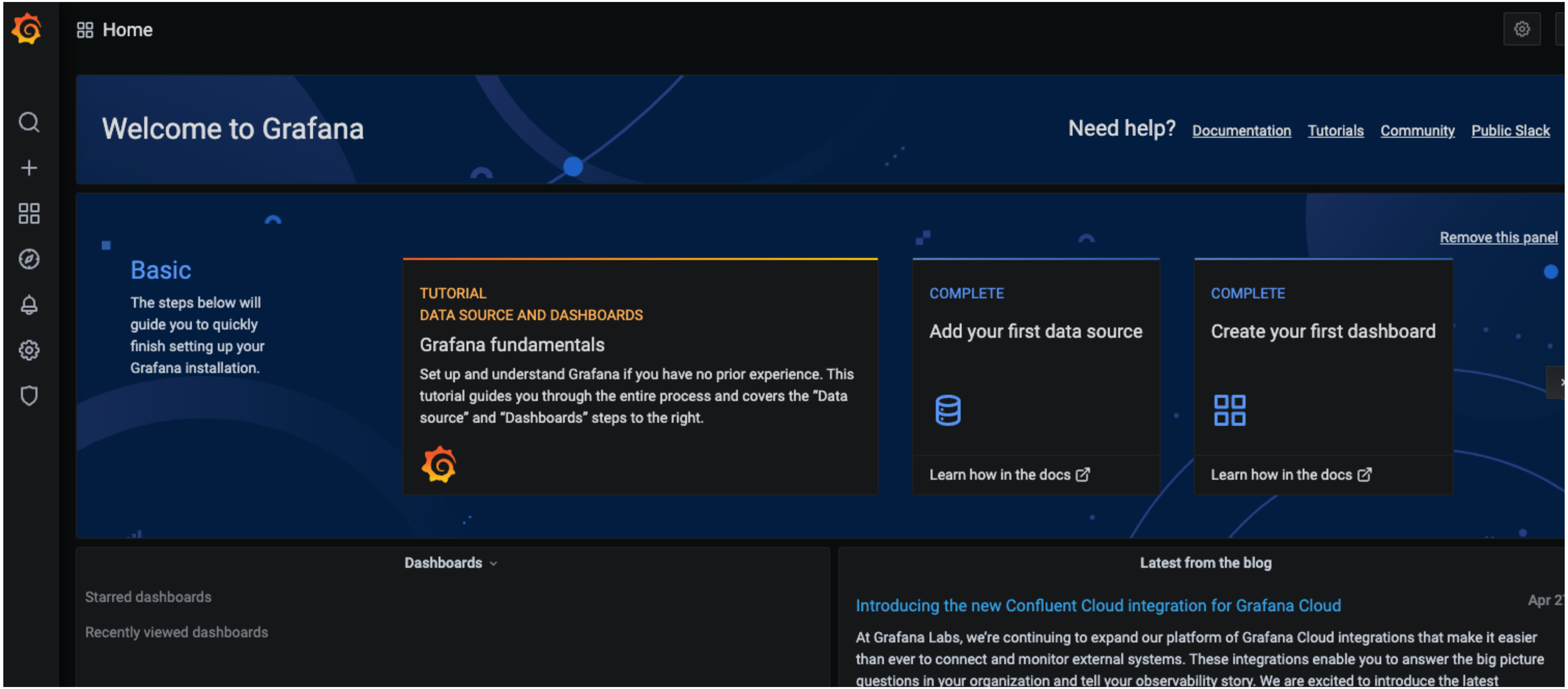
https://prometheus.cmn.SYSTEM_DOMAIN_NAME

ALERTMANAGER



https://alertmanager.cmn.SYSTEM_DOMAIN_NAME

GRAFANA



Home

Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

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

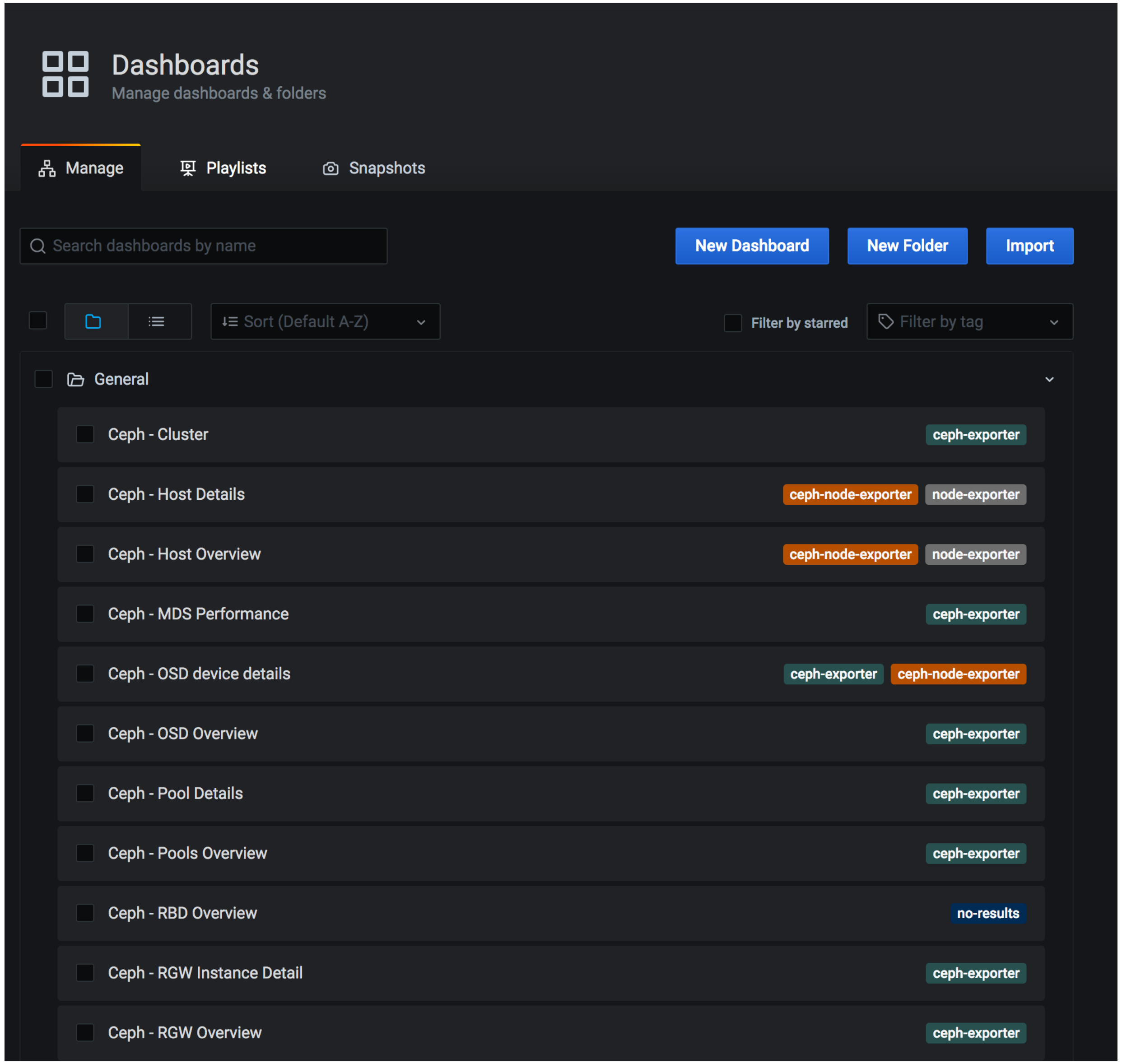
Apr 2

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



Ceph - Cluster	ceph-exporter
Ceph - Host Details	ceph-node-exporter node-exporter
Ceph - Host Overview	ceph-node-exporter node-exporter
Ceph - MDS Performance	ceph-exporter
Ceph - OSD device details	ceph-exporter ceph-node-exporter
Ceph - OSD Overview	ceph-exporter
Ceph - Pool Details	ceph-exporter
Ceph - Pools Overview	ceph-exporter
Ceph - RBD Overview	no-results
Ceph - RGW Instance Detail	ceph-exporter
Ceph - RGW Overview	ceph-exporter

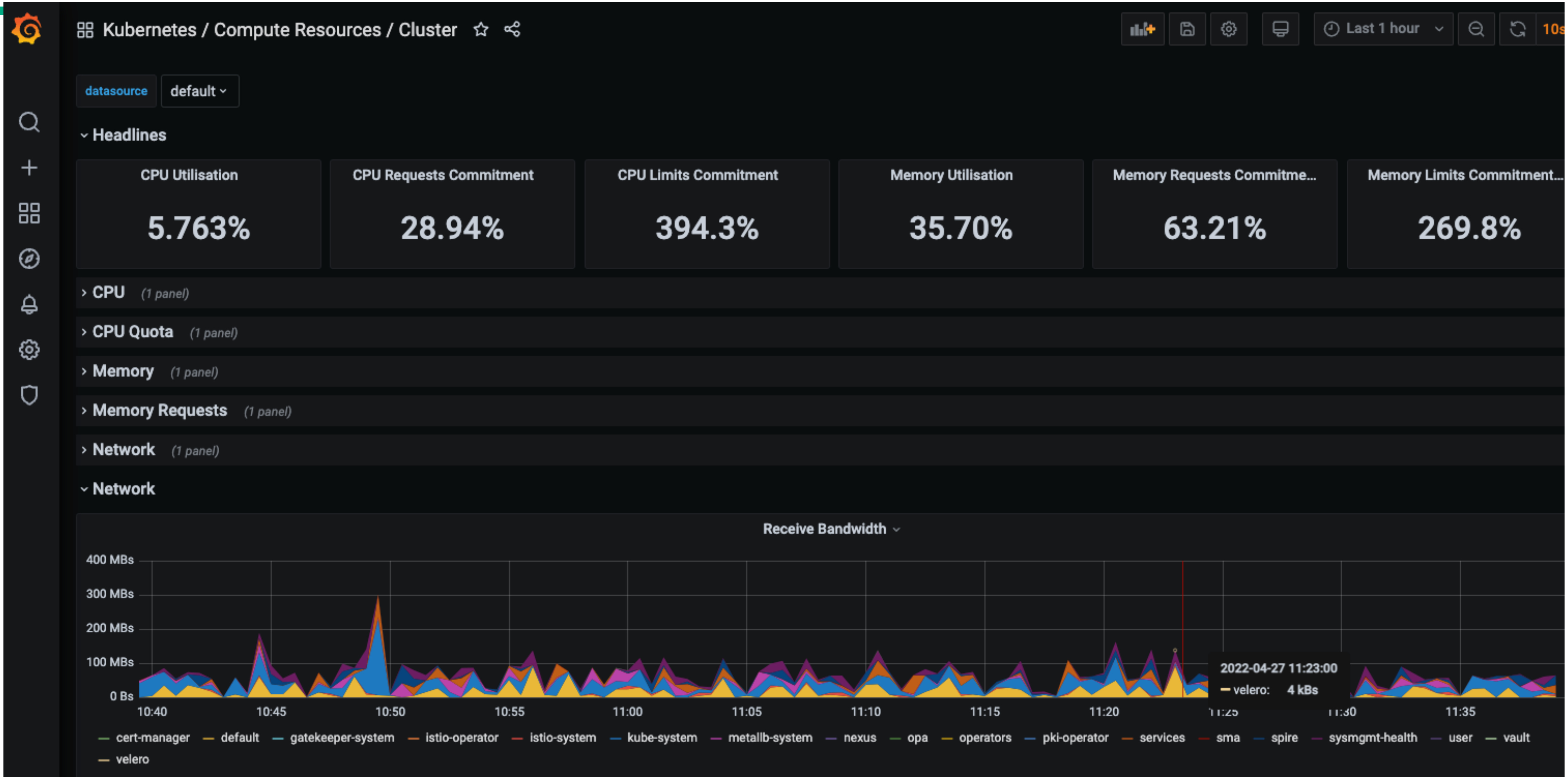
GRAFANA DASHBOARDS: ETCD

- 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

datasource: Prometheus
cluster: kube-etcd

2020-07-27 11:06:30
Watch Streams: 386
Lease Streams: 0

GRAFANA DASHBOARDS: KUBERNETES CLUSTER



CPU Utilisation: 5.763%

CPU Requests Commitment: 28.94%

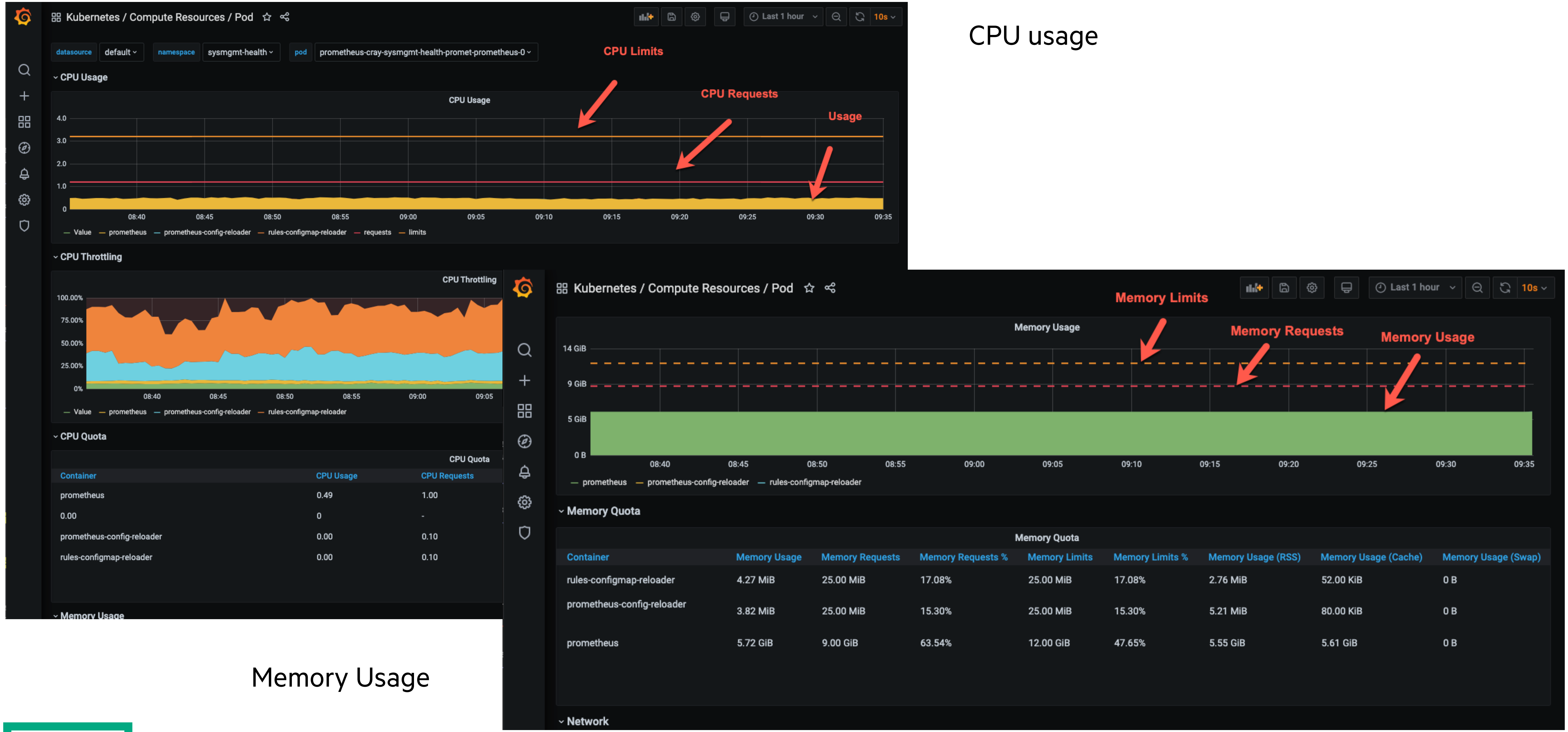
CPU Limits Commitment: 394.3%

Memory Utilisation: 35.70%

Memory Requests Commitme...: 63.21%

Memory Limits Commitment...: 269.8%

GRAFANA DASHBOARDS: KUBERNETES POD REQUESTS AND LIMITS



datasource: default
namespace: sysmgmt-health
pod: prometheus-cray-sysmgmt-health-promet-prometheus-0

CPU Limits

CPU Requests

Usage

CPU Quota

Container	CPU Usage	CPU Requests
prometheus	0.49	1.00
0.00	0	-
prometheus-config-reloader	0.00	0.10
rules-configmap-reloader	0.00	0.10

Memory Limits

Memory Requests

Memory Usage

Memory Quota

Container	Memory Usage	Memory Requests	Memory Requests %	Memory Limits	Memory Limits %	Memory Usage (RSS)	Memory Usage (Cache)	Memory Usage (Swap)
rules-configmap-reloader	4.27 MiB	25.00 MiB	17.08%	25.00 MiB	17.08%	2.76 MiB	52.00 KiB	0 B
prometheus-config-reloader	3.82 MiB	25.00 MiB	15.30%	25.00 MiB	15.30%	5.21 MiB	80.00 KiB	0 B
prometheus	5.72 GiB	9.00 GiB	63.54%	12.00 GiB	47.65%	5.55 GiB	5.61 GiB	0 B

CPU usage

Memory Usage

SLINGSHOT NETWORK (HSN)

SLINGSHOT TROUBLESHOOTING

- HPE Slingshot Troubleshooting Guide
  - Slingshot-Topology-Tool (STT)
  - River cable validator
  - linkdbg to debug downed links
  - DNS troubleshooting
  - Fabric manager
  - Slingshot NIC
  - Slingshot Switch
  - Network diagnostics

SLINGSHOT DIAGNOSTICS

- HPE Slingshot Troubleshooting Guide
  - dgnettest
    - Loopback bandwidth test, latency test, fabric bisection bandwidth test, fabric all-to-all
  - cxibwcheck.ch
    - Bi-directional loopback bandwidth for each Slingshot NIC on each node in a group of nodes
  - bwcheck.sh
    - Uni-directional loopback bandwidth for each Mellanox NIC on each node in a group of nodes
  - cxiberstat.sh
    - Measures NID link corrected and uncorrect bit error rates (BERs)
  - cxi_healthcheck
    - PCIe speed and width
    - Presence of PCIe errors
    - Algorithmic MAC assignment (optional)
    - Link state and speed
    - Link-layer retry setting is enabled
    - Internal loopback mode is disabled
    - Priority Flow Control (PFC) is enabled
    - Acceptable number of link flaps in the past hour (< 5) and the past 10 hours (< 10)
    - Presence of common error messages related to HPE Slingshot 200GB NIC in the kernel log
    - Services (retry handler, etc.) are in a running state (optional)
    - Resource and retry handler leak detection
    - Successful ping from HPE Slingshot 200 GB NIC interface to an external host / interface (optional)
    - Good, corrected, and uncorrected codeword rate check
    - Firmware revision check (optional)

FMN-SHOW-STATUS

- Check the current fabric status

```
ncn# kubectl exec -it -n services $(kubectl
get pods -A |grep fabric |awk '{print $2}')
-c slingshot-fabric-manager -- /bin/bash
slingshot-fabric-manager# fmn-show-status
Topology Status
Active: template-policy
Health
------
Runtime:HEALTHY
Configuration:WARNING
Traffic:HEALTHY
Security:HEALTHY
For more detailed Health - run 'fmctl get
health-engines/template-policy'

Port Policies (online / total ports for
each port-policy)
------------------------------------------
---------------
edge-policy: 0 / 0
fabric-policy: 10330 / 10340
cassini-policy: 4651 / 4664
qos-ll_be_bd_et-cassini-policy: 4651 /
4664
qos-hpc-cassini-policy: 0 / 0
qos-ll_be_bd_et-ethernet-policy: 0 / 0
qos-ll_be_bd_et-fabric-policy: 10330 /
10340
qos-hpc-fabric-policy: 0 / 0
lacp-policy: 0 / 0
offline-policy: 0 / 0

Edge: 4651 / 4664
Fabric: 10330 / 10340
Ports Reported: 15004 / 15004
Ports in Error State: 6 / 15004
Fully Synchronized Switches: 296 / 296
```

FMN-SHOW-STATUS DETAILS

- Check the current fabric status with details

```
ncn# kubectl exec -it -n services $(kubectl get pods -A
|grep fabric |awk '{print $2}') -c slingshot-fabric-
manager -- /bin/bash
slingshot-fabric-manager# fmn-show-status -d
Topology Status
… (Same as last slide for early part of output)

Edge: 4651 / 4664
Fabric: 10330 / 10340
Ports Reported: 15004 / 15004
Ports in Error State: 6 / 15004
Fully Synchronized Switches: 296 / 296

23 Downed links:
Fabric: x1000c5r5j5p0
Fabric: x1002c4r1j17p0
Fabric: x1003c4r1j17p0
Fabric: x1004c1r1j5p0
Fabric: x1006c0r7j13p1
Fabric: x1006c1r5j13p1
Fabric: x1006c6r1j13p1
Fabric: x1006c6r3j11p1
Fabric: x3000c0r33j3p0
Fabric: x3002c0r33j32p0
Edge: x1000c0r1j104p1
Edge: x1000c0r5j102p0
Edge: x1000c0r5j102p1
Edge: x1000c0r7j101p0
Edge: x1000c0r7j101p1
Edge: x1000c0r7j102p0
Edge: x1000c0r7j102p1
Edge: x1000c0r7j103p0
Edge: x1001c1r7j107p0
Edge: x1001c1r7j107p1
Edge: x1004c2r7j105p1
Edge: x1007c0r5j107p0
Edge: x1007c0r7j107p0

Port errors:
x1000c5r5j5p0 : Port capability degraded to prevent
excessive flapping
x1000c0r7j101p0 : Port disabled due to excessive
flapping
x3002c0r33j32p0 : Port disabled due to excessive
flapping
x1003c4r1j17p0 : Port capability degraded to prevent
excessive flapping
x3000c0r33j3p0 : Port disabled due to excessive
flapping
x1002c4r1j17p0 : Port capability degraded to prevent
excessive flapping
```

LINKDBG FABRIC ERRORS

- Check fabric link errors

```
slingshot-fabric-manager# linkdbg -L fabric
Querying downed links' link partners...

+--------+-----------------------------------------------+------------+----------+----------------
+--------------+-------------+------------------+-------------+----------------+
|  type  | rosprt xname (pport) <-> rosprt xname (pport) | rosswinfo  | sC firmW | sw_medtype-pw  |
lp rosswinfo | lp sC firmW | lp sw_medtype-pw | action_code | lp action_code |
+--------+-----------------------------------------------+------------+----------+----------------
+--------------+-------------+------------------+-------------+----------------+
| Fabric | x1000c5r5j5p0   (25) <-> x1004c1r1j5p0   (25) | tpml d S L |  2.0.2   |   Optical-
07   |  tpml d s L  |    2.0.2    |    Optical-07    |  otherport  |      ros6      |
| Fabric | x1002c4r1j17p0  (58) <-> x1003c4r1j17p0  (58) | tpml d S L |  2.0.2   |   Optical-
06   |  tpml d S L  |    2.0.2    |    Optical-07    |  otherport  |      ros5      |
| Fabric | x1006c0r7j13p1  (62) <-> x1006c6r1j13p1  (62) | tpml d s L |  2.0.2   | Electrical-N/A
|  tpml d S L  |    2.0.2    |  Electrical-N/A  |     ros6    |      ros5      |
| Fabric | x1006c1r5j13p1  (62) <-> x1006c6r3j11p1  (14) | tpml d s L |  2.0.2   | Electrical-N/A
|  tpml d S L  |    2.0.2    |  Electrical-N/A  |     ros6    |      ros5      |
| Fabric | x3000c0r33j3p0  (18) <-> x3002c0r33j32p0 (33) | tpml D S L |  2.0.2   |   Optical-
06   |  tpml D S L  |    2.0.2    |    Optical-06    |     ros4    |      ros4      |
+--------+-----------------------------------------------+------------+----------+----------------
+--------------+-------------+------------------+-------------+----------------+
```

LINKDBG EDGE ERRORS

- Check fabric link errors

```
slingshot-fabric-manager# linkdbg -L edge
Querying downed links' link partners...

+------+-------------------------------------------+------------+----------+----------------+-------------+
| type | rosprt xname (pport) <-> link_partner      | rosswinfo  | sC firmW | sw_medtype-pw  | action_code |
+------+-------------------------------------------+------------+----------+----------------+-------------+
| Edge | x1000c0r1j104p1 (16) <-> x1000c0s4b0n1h1  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r5j102p0 (49) <-> x1000c0s2b1n1h1  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r5j102p1 (48) <-> x1000c0s2b1n0h1  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r7j101p0 (34) <-> x1000c0s1b1n0h0  | tpml D S L |  2.0.2   | Electrical-N/A |     ros4    |
| Edge | x1000c0r7j101p1 (35) <-> x1000c0s1b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r7j102p0 (49) <-> x1000c0s2b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r7j102p1 (48) <-> x1000c0s2b1n0h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1000c0r7j103p0 (33) <-> x1000c0s3b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1001c1r7j107p0 ( 2) <-> x1001c1s7b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1001c1r7j107p1 ( 3) <-> x1001c1s7b1n0h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1004c2r7j105p1 ( 0) <-> x1004c2s5b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1007c0r5j107p0 ( 2) <-> x1007c0s7b1n1h1  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
| Edge | x1007c0r7j107p0 ( 2) <-> x1007c0s7b1n1h0  | tpml d S L |  2.0.2   | Electrical-N/A |  unkn_port  |
+------+-------------------------------------------+------------+----------+----------------+-------------+
```

LINKDBG ACTION CODES

- Get explanation of action codes

```
slingshot-fabric-manager# linkdbg -a ros4
PROBLEM SYNOPSIS: The link has been directed down.

WHY YOU GOT THIS ACTION CODE:
    The link monitor (lmon) state letter in the rosswinfo column, "D/d",
is
    capitalized.

HOW TO DIAGNOSE:
    Validate each issue, in order.

POSSIBLE ISSUES:
    1) The link direction could be in a transient state.  Rerun
    "linkdbg -t <portxname>" twice over a 10 second interval.
    2) Link auto retry has been exhausted.  This will be reported in fmn_status on
    the FMN.  The link was flapping too much, and needs hardware attention:

        (After each step, re-run linkdbg to see if the action resolved the issue.)

        Steps to debug Mountain Cabinet downed links between NIC and switch:
            1) Validate both switch and NIC are properly configured and attempting
            to bring up the HSN link.
            2) Perform group hug:
                Apply pressure to both the compute and switch blades simultaneously
                to seat the ExaMax connectors more firmly.
            3) Reseat the switch blade (this will act as an asic reset and reboot
as well).
            4) Reseat the compute blade.
            5) Swap NIC mezzanine cards between nodes. # look to see if follows NIC
or stays with cable.
                If failure follows NIC, replace the NIC.
                If failure stays with L0 cable, replace the L0 cable.
            6) Replace node card with known node card that has good link on the
            reporting errored HSN link.
            7) Replace switch.

        Steps to debug River Cabinet downed links between NIC and switch:
            1) Validate both switch and NIC are properly configured and attempting
            to bring up the HSN link.
            2) Reseat cable between NIC card and switch.
            3) Reboot compute node.
            4) Reset asic and reboot switch.
            5) Reseat cable between NIC card and switch a second time.
            6) Replace cable between NIC card and switch.
            7) Replace NIC card.
            8) Replace switch.

        Steps to debug downed links between switches:
            1) Validate both switches are properly configured and attempting to
            bring up the HSN link.
            2) Reseat cable between switches.
            3) Reset ASIC
            4) Slot power cycle
            5) Reseat cable between switches a second time.
            6) Replace cable between switches.
            7) Replace local switch.
            8) Replace remote switch.

    3) The link hasn't been commanded up.  Run the following from the switch, where
       "portnumber" is the number reported by linkdbg in parenthesis next to its xname:
        swtest -c "link $((1<<portnumber)) up"
```

SYSTEM TESTING

- CSM Health Checks
- CSM Diags

CSM HEALTH CHECKS

- CSM documentation describes system health validation
  - Run before rebooting or rebuilding a management node
  - Run before complete system graceful shutdown
  - Run during complete system graceful startup
  - Run during complete system non-graceful startup
  - Run as part of troubleshooting toolbox
  - See procedures in CSM documentation
    https://github.com/Cray-HPE/docs-csm/tree/release/1.3/operations/validate_csm_health.md
- Platform Health Checks
  - Automated NCN checks using goss servers on each NCN
    `/opt/cray/tests/install/ncn/automated/ncn-healthcheck`
  - Automated Kubernetes check using goss servers on each NCN
    `/opt/cray/tests/install/ncn/automated/ncn-kubernetes-check`
  - Manual ncnHealthChecks

    ```
    /opt/cray/platform-utils/ncnHealthChecks.sh
          -s ncn_uptimes
          -s node_resource_consumption
          -s pods_not_running
    ```
  - Manual ncnPostgresHealthChecks
    `/opt/cray/platform-utils/ncnPostgresHealthChecks.sh`
  - System management monitoring tools
    - Prometheus, Alertmanager, Grafana, Kiali
  - BGP Peering Status and Reset

MORE CSM HEALTH CHECKS

- See procedures in CSM documentation
  https://github.com/Cray-HPE/docs-csm/tree/release/1.3/operations/validate_csm_health.md
- Hardware Management Services
  - HMS Test execution
    `/opt/cray/csm/scripts/hms_verification/run_hms_ct_tests.sh`
  - HSM Discovery Validation
    `/opt/cray/csm/scripts/hms_verification/verify_hsm_discovery.py`
- Software Management Services
  - BOS, TFTP, cray-console, IMS, CFS, VCS, CRUS
    `– /usr/local/bin/cmsdev test -q all`
- Gateway health and SSH access checks
  - Gateway health tests from NCN
    `/usr/share/doc/csm/scripts/operations/gateway-test/ncn-gateway-test.sh`
  - Gateway health tests from outside the system
  - Internal SSH access
    `/usr/share/doc/csm/scripts/operations/pyscripts/start.py test_bican_internal`
  - External SSH access
- Booting CSM Barebones image
  - Tests whether the booting services infrastructure is functional to boot a compute node
- UAS/UAI tests
  - Validate basic UAS installation
  - Validate UAI creation
  - Troubleshooting UAS/UAI

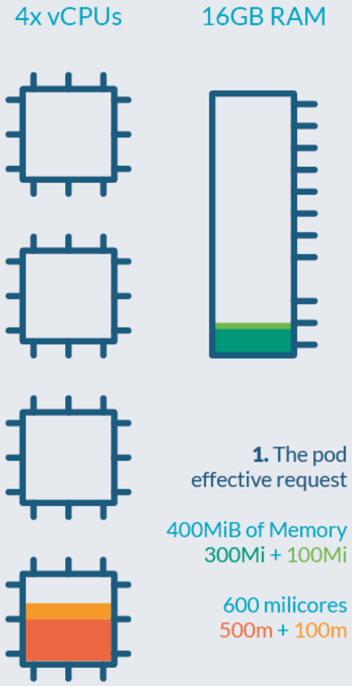
KUBERNETES LIMITS AND EXCEPTIONS

The pod - Deployment.yaml

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
 name: redis
 labels:
   name: redis-deployment
   app: example-voting-app
spec:
 replicas: 1
 selector:
   matchLabels:
    name: redis
    role: redisdb
    app: example-voting-app
 template:
   spec:
     containers:
       - name: redis
         image: redis:5.0.3-alpine
         resources:
           limits:
             memory: 600Mi
             cpu: 1
           requests:
             memory: 300Mi
             cpu: 500m
       - name: busybox
         image: busybox:1.28
         resources:
           limits:
             memory: 200Mi
             cpu: 300m
           requests:
             memory: 100Mi
             cpu: 100m
```

https://sysdig.com/blog/kubernetes-limits-requests/

A cluster node:



1. The pod effective request

400MiB of Memory
300Mi + 100Mi

600 milicores
500m + 100m

2. Kubernetes assigns 1024 shares per core.

1024 / 0.1 = 102 shares
1024 / 0.5 = 512 shares

3. Will be killed if allocates > 600MB.
The whole Pod will fail.

4. Will be throttled if uses more than "1 Core".
1 core = 1000 milicores = 1000m =
100ms of computing time every 100 real ms

Full computing time of the node:
4 vCPUs * 100 real ms =
400ms of computing time = 4000m

5. Killed if allocates > 200MB.

6. Throttled if uses > 30ms of computing time in 100ms

CPU AND MEMORY LIMITS

```
ncn# kubectl get LimitRange --all-namespaces
NAMESPACE        NAME                            CREATED AT
backups          cpu-mem-limit-range             2022-01-19T18:49:07Z
ceph-cephfs      cpu-mem-limit-range             2022-01-19T18:49:06Z
ceph-rbd         cpu-mem-limit-range             2022-01-19T18:49:07Z
default          cpu-mem-limit-range-requests    2022-01-19T18:49:08Z
ims              cpu-mem-limit-range             2022-01-19T18:49:07Z
istio-system     cpu-mem-limit-range             2022-01-19T18:49:07Z
loftsman         cpu-mem-limit-range             2022-01-19T18:49:07Z
metallb-system   cpu-mem-limit-range             2022-01-19T18:49:07Z
operators        cpu-mem-limit-range             2022-01-19T19:29:37Z
pki-operator     cpu-mem-limit-range             2022-01-19T19:29:37Z
services         cpu-mem-limit-range             2022-01-19T19:29:37Z
sma              cpu-mem-limit-range             2022-01-19T18:49:07Z
sysmgmt-health   cpu-mem-limit-range             2022-01-19T18:49:08Z
uas              cpu-mem-limit-range             2022-01-19T19:45:25Z
user             cpu-mem-limit-range             2022-01-19T19:45:25Z
vault            cpu-mem-limit-range             2022-01-19T19:29:37Z
velero           cpu-mem-limit-range             2022-01-19T18:49:08Z
```

POD MEMORY USAGE

```
ncn# kubectl top pod --all-namespaces --sort-by=memory
NAMESPACE           NAME                                                     CPU(cores)   MEMORY(bytes)
sma                 elasticsearch-master-1                                   56m          33242Mi
sma                 elasticsearch-master-0                                   172m         33163Mi
sma                 elasticsearch-master-2                                   166m         33160Mi
sma                 cluster-kafka-0                                          258m         7873Mi
sma                 cluster-kafka-1                                          177m         6813Mi
sma                 cluster-kafka-2                                          173m         6047Mi
sysmgmt-health      prometheus-cray-sysmgmt-health-promet-prometheus-0       383m         5760Mi
istio-system        prometheus-c6f686f44-287qm                               201m         4217Mi
istio-system        prometheus-c6f686f44-jz7xg                               182m         3585Mi
istio-system        prometheus-c6f686f44-8p7p5                               221m         3421Mi
nexus               nexus-7b948976d7-rgzbf                                   11m          2408Mi
sma                 sma-monasca-thresh-node-7594fcd77-wrz4d                  849m         1633Mi
kube-system         kube-apiserver-ncn-m001                                  300m         1563Mi
kube-system         kube-apiserver-ncn-m002                                  102m         1408Mi
services            cray-shared-kafka-kafka-2                                52m          1380Mi
services            slingshot-fabric-manager-6d7fbb785f-d7scw                50m          1348Mi
services            cray-shared-kafka-kafka-0                                41m          1283Mi
services            cray-shared-kafka-kafka-1                                40m          1257Mi
sma                 sma-postgres-cluster-1                                   14m          1172Mi
sma                 sma-monasca-thresh-dmtf-6c4fcc7c84-2vlzc                 845m         1152Mi
sma                 sma-monasca-thresh-metrics-69cf45c768-2kmq9              835m         1144Mi
sma                 cluster-zookeeper-1                                      17m          1031Mi
```

POD CPU USAGE

```
ncn# kubectl top pod --all-namespaces --sort-by=cpu
NAMESPACE           NAME                                                                   CPU(cores)   MEMORY(bytes)
sysmgmt-health      prometheus-cray-sysmgmt-health-promet-prometheus-0                     1562m        5762Mi
sma                 sma-monasca-thresh-node-7594fcd77-wrz4d                                874m         1634Mi
sma                 sma-monasca-thresh-dmtf-6c4fcc7c84-2vlzc                               839m         1152Mi
sma                 sma-monasca-thresh-metrics-69cf45c768-2kmq9                            832m         1144Mi
kube-system         kube-apiserver-ncn-m001                                                312m         1563Mi
sma                 cluster-kafka-0                                                        220m         7883Mi
istio-system        prometheus-c6f686f44-8p7p5                                             212m         3423Mi
istio-system        prometheus-c6f686f44-jz7xg                                             189m         3586Mi
istio-system        prometheus-c6f686f44-287qm                                             182m         4217Mi
sma                 elasticsearch-master-2                                                 167m         33160Mi
sma                 cluster-kafka-2                                                        161m         6050Mi
gatekeeper-system   gatekeeper-controller-manager-588d6476db-hrmns                         158m         119Mi
sma                 cluster-kafka-1                                                        153m         6819Mi
sma                 elasticsearch-master-0                                                 146m         33164Mi
kube-system         kube-apiserver-ncn-m003                                                113m         960Mi
sysmgmt-health      cray-sysmgmt-health-prometheus-node-exporter-5jjgw                     110m         212Mi
sysmgmt-health      cray-sysmgmt-health-prometheus-node-exporter-gpb8w                     109m         232Mi
kube-system         kube-apiserver-ncn-m002                                                102m         1408Mi
```

CEPH STATUS

- Ceph status shows health, expected and running services, storage information

```
ncn-s# ceph -s
  cluster:
    id:     b1781806-9370-43af-96aa-61447a4d9411
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum ncn-s003,ncn-s002,ncn-s001 (age 6w)
    mgr: ncn-s001(active, since 6w), standbys: ncn-s003, ncn-s002
    mds: cephfs:1 {0=ncn-s001=up:active} 2 up:standby
    osd: 24 osds: 24 up (since 6w), 24 in (since 6w)
    rgw: 3 daemons active (ncn-s001.rgw0, ncn-s002.rgw0, ncn-s003.rgw0)

  data:
    pools:   11 pools, 816 pgs
    objects: 357.05k objects, 786 GiB
    usage:   1.2 TiB used, 41 TiB / 42 TiB avail
    pgs:     816 active+clean

  io:
    client:   75 KiB/s rd, 10 MiB/s wr, 24 op/s rd, 1.07k op/s wr
```

STORAGE UTILIZATION

```
ncn-s# ceph df
--- RAW STORAGE ---
CLASS   SIZE    AVAIL   USED      RAW USED  %RAW USED
ssd     63 TiB  60 TiB  2.8 TiB    2.9 TiB       4.55
TOTAL   63 TiB  60 TiB  2.8 TiB    2.9 TiB       4.55

--- POOLS ---
POOL                         ID  PGS  STORED   OBJECTS  USED     %USED  MAX AVAIL
cephfs_data                   1  256  385 GiB  311.95k  1.1 TiB   1.96     19 TiB
cephfs_metadata               2  256  405 MiB   19.83k  1.2 GiB      0     19 TiB
default.rgw.buckets.data      3  256  103 GiB   27.96k  309 GiB   0.53     19 TiB
default.rgw.buckets.index     4   32  3.1 MiB      704  9.2 MiB      0     19 TiB
.rgw.root                     5   16  5.2 KiB       18  204 KiB      0     19 TiB
default.rgw.control           6   16      0 B        8      0 B      0     19 TiB
default.rgw.meta              7   16  788 KiB      171  3.9 MiB      0     19 TiB
default.rgw.log               8   16   30 KiB      210  624 KiB      0     19 TiB
kube                          9  256   36 GiB   18.30k   76 GiB   0.13     19 TiB
smf                          10  512  1.1 TiB  488.25k  1.3 TiB   2.28     28 TiB
default.rgw.buckets.non-ec   11   16      0 B        0      0 B      0     19 TiB
device_health_metrics        12    1   48 MiB       39  145 MiB      0     19 TiB
```

CSM DIAGS

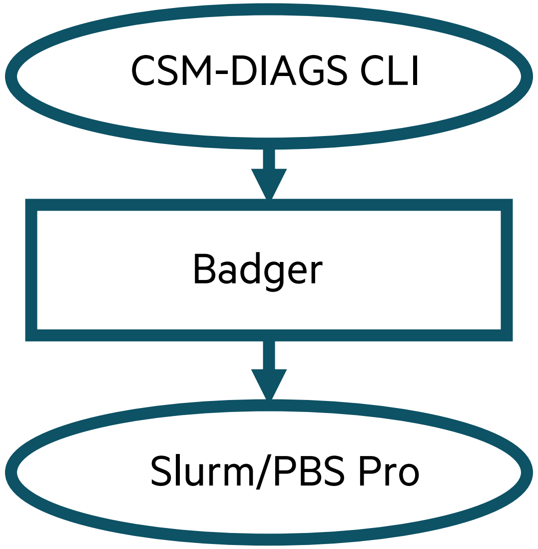
A set of diagnostic tools to perform various node level and system wide tests on compute nodes

- Functional test suites and performance test suites with both MPI and non MPI test suites
- Tests initiated using cray-hms-badger service to submit WLM jobs on compute nodes
  - Consistency checks
    cpuchk, memchk, fabricchk, netchk, fschk, mpichk
  - System Level Diagnostics
    linpack, cwlinpack, nodeperf, stream, olcmt, oldisk, olconf, cwolconf, rank, pandora, cwhpcc
  - Nvidia GPU Diagnostics
    gpu-burn, xkbandwidth, xkcheck, xkdgemm, xkmemtest, xbandwidthtest, xkstress, dgnettest
  - AMD GPU diagnostics
    amdgpubandwidth, amdgpuproperties, amdgpuedpp, amdgpufilechk, adgpukernelchk, amdgpulinkchk, amdgpumonitor, amdgpup2pchk, amdgpupciechk, amdgpupciemonitor, amdgpupkgchk, amdgpubioschk, amdgpustresstest, amdgpuuserchk
  - Fabric diagnostics
    dgnettest, check_excessive_pause
  - OSU Benchmark
    osu_startup, osu_bw_bibw, osu_single_multi_latency, osu_multiplebw_message_rate, osu_multithread_multiprocess_latency, osu_bw_latency_ops, osu_put_bibw, osu_get_acc_latency, osu_collective_blocking_barrier, osu_collective_MPI_blocking_ops, osu_collective_MPI_non_blocking_ibarrier, osu_collective_MPI_non_blocking_ops,
- sdiag_run.py using cray-hms-badger
  - Execute multiple diagnostics (MPI, NON_MPI, GPU, Slingshot) in one shot on multiple compute nodes

CSM DIAGS - CLI

- A CLI (which uses Badger framework) has been provided on the worker nodes to execute multiple diagnostics (MPI, NON_MPI, Slingshot) in a single instance on multiple compute nodes
  - Admin needs to modify the configuration files, with the list of diagnostics that need to be executed

– sdiag-list.json	(List of diagnostics which Admin needs to run)
– sdiag-arguments.json	(Argument Values for each Diagnostic Test)
– sdiag-gumball.json	(Badger Information, Session directory)
– nodes	(file with the list of node xnames or nids)
– nodes_gpu	(file with the list of GPU node xnames or nids)

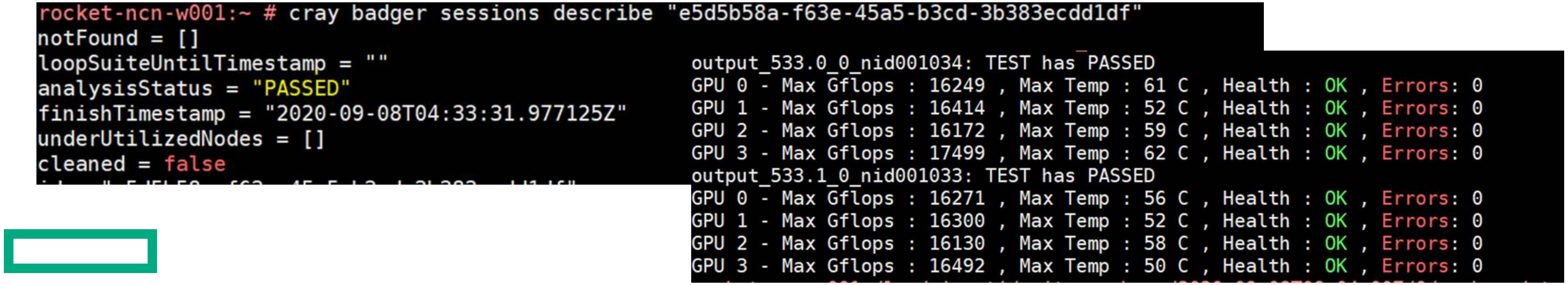

CSM-DIAGS CLI

Badger

Slurm/PBS Pro

- Can be run by:

```
ncn-w# /opt/cray/csm-diags/sdiag_run.py
 out_02:12:02.txt output file has been created in /var/log/cray/shasta-diag
 –Execution completed
     gpu-burn: cray badger sessions describe e5d5b58a-f63e-45a5-b3cd-3b383ecdd1df'
```



```
rocket-ncn-w001:~ # cray badger sessions describe "e5d5b58a-f63e-45a5-b3cd-3b383ecdd1df"
notFound = []
loopSuiteUntilTimestamp = ""
analysisStatus = "PASSED"
finishTimestamp = "2020-09-08T04:33:31.977125Z"
underUtilizedNodes = []
cleaned = false
```

```
output_533.0_0_nid001034: TEST has PASSED
GPU 0 - Max Gflops : 16249 , Max Temp : 61 C , Health : OK , Errors: 0
GPU 1 - Max Gflops : 16414 , Max Temp : 52 C , Health : OK , Errors: 0
GPU 2 - Max Gflops : 16172 , Max Temp : 59 C , Health : OK , Errors: 0
GPU 3 - Max Gflops : 17499 , Max Temp : 62 C , Health : OK , Errors: 0
output_533.1_0_nid001033: TEST has PASSED
GPU 0 - Max Gflops : 16271 , Max Temp : 56 C , Health : OK , Errors: 0
GPU 1 - Max Gflops : 16300 , Max Temp : 52 C , Health : OK , Errors: 0
GPU 2 - Max Gflops : 16130 , Max Temp : 58 C , Health : OK , Errors: 0
GPU 3 - Max Gflops : 16492 , Max Temp : 50 C , Health : OK , Errors: 0
```

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

TUNING COMPUTE NODES

- Performance
  - Low Noise Mode (LNM)
  - Cgroups
  - DVS
  - CPS tuning
  - Overlay preload for DVS
- Workload Managers
  - Slurm config for HPE 200GB (Cassini) NICs
    - VNI allocation, network resource reservation, and traffic class configuration
  - Application Task Orchestration and Management (ATOM)

PERFORMANCE

LOW NOISE MODE (LNM)

- Configures Linux kernel so OS tasks are migrated to one or more CPUs (on each node) which are excluded from application use
  - Full LNM
    - Kernel parameters at boot will reduce noise by moving system activities to CPU 0 (and potentially additional CPUs as well) and user space is configured to move any overhead processes to CPU 0
  - Lightweight mode
    - The kernel parameters are not used, but the user space configuration is still done
  - Must coordinate COS kernel parameters and WLM settings for LNM
- IRQs can be listed by name instead of by number, and there are new options for selecting to what CPUs the IRQs are directed

LNM KERNEL PARAMETERS

- In full LNM mode, the Linux kernel must be booted with parameters to direct noise overhead to CPU 0
  - The CPU range has to be specified for the number of CPUs on the node

```
nohz_full=1-255
rcu_nocbs=1-255
rcu_nocb_poll
```

- In addition, two parameters are specified on the kernel command line to guide the behavior of the user space configuration
  - The lnm parameter must be set to either full or lightweight.

```
lnm={ full, lightweight }
```

  - The lnm.cpu parameter is optional and is set to what CPU(s) to use to handle system overhead

```
[lnm.cpu=0]
```

- These boot parameters are set in a BOS session template

LNM CONFIGURATION FILE

- At boot, systemd runs lnmctl to configure the user space environment for low noise
- lnmctl reads default configuration from /etc/lnm-default.json and optional site configuration from /etc/lnm.json
  - Tunables – settings for sysctl and files in /sys
  - Processes – process names that should not be migrated to the system CPU(s)
  - IRQs – which hardware interrupt requests (IRQs) should be directed to the CPUs handling system overhead
    - cpu_0 for the first CPU on the node
    - cpu_last for the highest numbered CPU on the node.
    - cpu_all to use all of the CPUs listed in lnm.cpu or the CPU section
    - cpu_all:<list> to use all the listed CPUs
    - cpu_closest to use the CPU from lnm.cpu or the CPU section which is on the same NUMA node as the IRQ
    - cpu_closest:<list> use the CPU from the list which is on the same NUMA node as the IRQ. When using cpu_closest, there must be a system CPU specified for each NUMA node. The list is a comma-separated list of CPU numbers.
  - CPU - Which processors are used for system overhead
    - Single CPU, range of CPUs, comma separated list of CPUs, or MAX to use highest numbered CPU

```
{
  "Tunables": {
    "sysctl": {
      "vm.stat_interval": 120
    },
    "files": {
      "/sys/bus/workqueue/devices/writeback/cpumask": 1,
      "/sys/kernel/mm/transparent_hugepage/enabled": "never"
    }
  },
  "Processes": [".*watchdog.* ", " DVS-IPC_msg "],
  "IRQs": {
    "0": "cpu_0",
    "1": "cpu_last",
    "2": "cpu_all:1,2,4,8",
    ".*gpu": "cpu_closest:1,2,4,8"
  }
  "CPU": "0,128”
}
```

LOW NOISE MODE SLURM

- Avoid placing applications on CPU 0 if some or all system compute nodes are configured with the Low Noise Mode feature
- Slurm.conf
  - SchedulerParameters=spec_cores_first
    - Use core 0 instead of last core
  - AllowSpecResourcesUsage=YES
    - (Optional) allows users to override the specialized cores with `srun -S`
  - NodeName=nid000010 Sockets=2 CoresPerSocket=16 ThreadsPerCore=2 RealMemory=55296 Feature=Intel_Xeon_Gold_6130 CoreSpecCount=1
    - On each node configured with LNM, avoid one core by default

CGROUPS (CONTROL GROUPS)

- cgroups is a mechanism to organize processes hierarchically and distribute system resources along the hierarchy in a controlled and configurable manner
  - cgroup core is primarily responsible for hierarchically organizing processes
  - cgroup controller is usually responsible for distributing a specific type of system resource along the hierarchy although there are utility controllers which serve purposes other than resource distribution
- cgroups form a tree structure and every process in the system belongs to one and only one cgroup
  - All threads of a process belong to the same cgroup
  - On creation, all processes are put in the cgroup that the parent process belongs to at the time
  - A process can be migrated to another cgroup
  - Migration of a process doesn't affect already existing descendant processes
- Following certain structural constraints, controllers may be enabled or disabled selectively on a cgroup
  - All controller behaviors are hierarchical - if a controller is enabled on a cgroup, it affects all processes which belong to the cgroups consisting the inclusive sub-hierarchy of the cgroup
  - When a controller is enabled on a nested cgroup, it always restricts the resource distribution further
  - The restrictions set closer to the root in the hierarchy can not be overridden from further away

CGROUPS V2

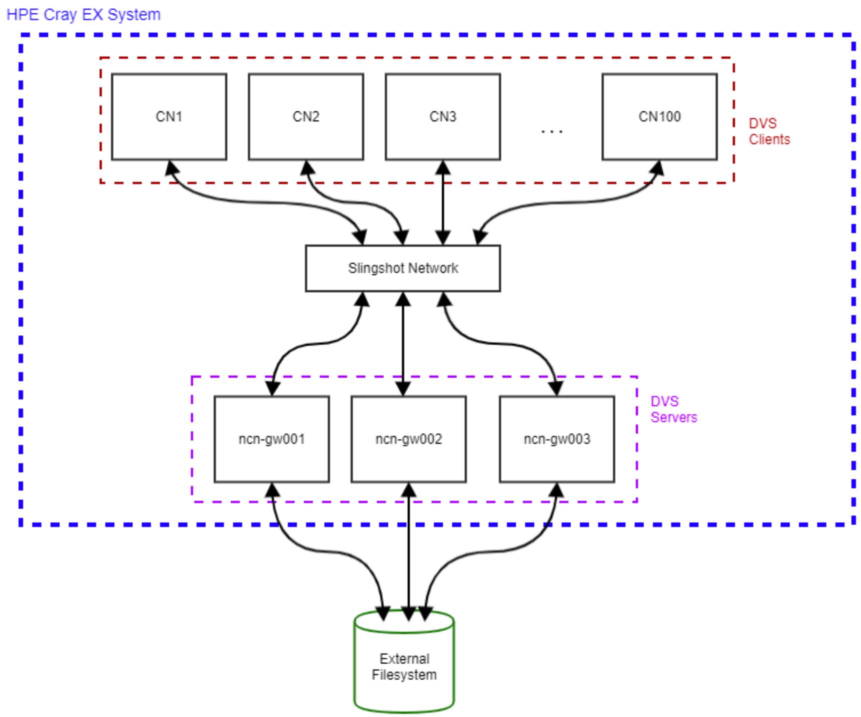
- cgroups v2 offers several improvements over cgroup v1
  - Single unified hierarchy design in API
  - Safer sub-tree delegation to containers
  - Newer features like Pressure Stall Information
  - Enhanced resource allocation management and isolation across multiple resources
    - Unified accounting for different types of memory allocations (network memory, kernel memory, etc)
    - Accounting for non-immediate resource changes such as page cache write backs
- cgroups v2 uses a different API than cgroup v1, so if there are any applications that directly access the cgroup file system, they need to be updated to newer versions that support cgroups v2
  - Identify the cgroup version on Linux Nodes
    - The cgroup version depends on the Linux distribution being used and the default cgroup version configured on the OS.

```
Compute# stat -fc %T /sys/fs/cgroup/
```

      - For cgroup v2, the output is cgroup2fs
      - For cgroup v1, the output is tmpfs
- COS 2.4 supports cgroups v2 in addition to cgroups v1
  - The system boots with only cgroups v2 by default
  - Set kernel parameters `systemd.unified_cgroup_hierarchy`, to boot with choose version of cgroups
    - In default mode, this parameter is set in the boot_parameter file with the value of 1, for cgroups version 2:
    - `systemd.unified_cgroup_hierarchy=1`
    - To override this kernel boot parameter to activate cgroups version 1, set the value to 0 in the BOS boot session template:
    - `systemd.unified_cgroup_hierarchy=0`
  - Nvidia open-source driver enables cgroups v2 support for CUDA managed memory

DVS

- Data Virtualization Service (DVS)
  - Distributed network service projects file systems mounted on NCNs to other nodes within the system
  - Projecting makes a file system available on nodes where it does not physically reside
  - DVS-specific configuration settings enable clients to access a file system projected by DVS servers
  - Represents a software layer that provides scalable transport for file system services
  - Uses Lustre Networking (LNet) to communicate over the network
    - LNet configuration is done by the code that configures DVS
- Works with CPS to project internal file systems to nodes
- Projecting external file systems from gateway nodes
  - DVS provides I/O performance and scalability to many nodes
    - Far beyond the number of clients supported by a single NFS server
  - HPE DVS configuration minimizes
    - Operating system noise
    - Impact on compute node memory resources
  - DVS
    - Uses Linux virtual file system (VFS) interface to process file system operations
    - Can project Any POSIX-compliant file system
      - such as Spectrum Scale (GPFS), NFS, and Lustre
  - Gateway nodes need custom OS images built to support Spectrum Scale

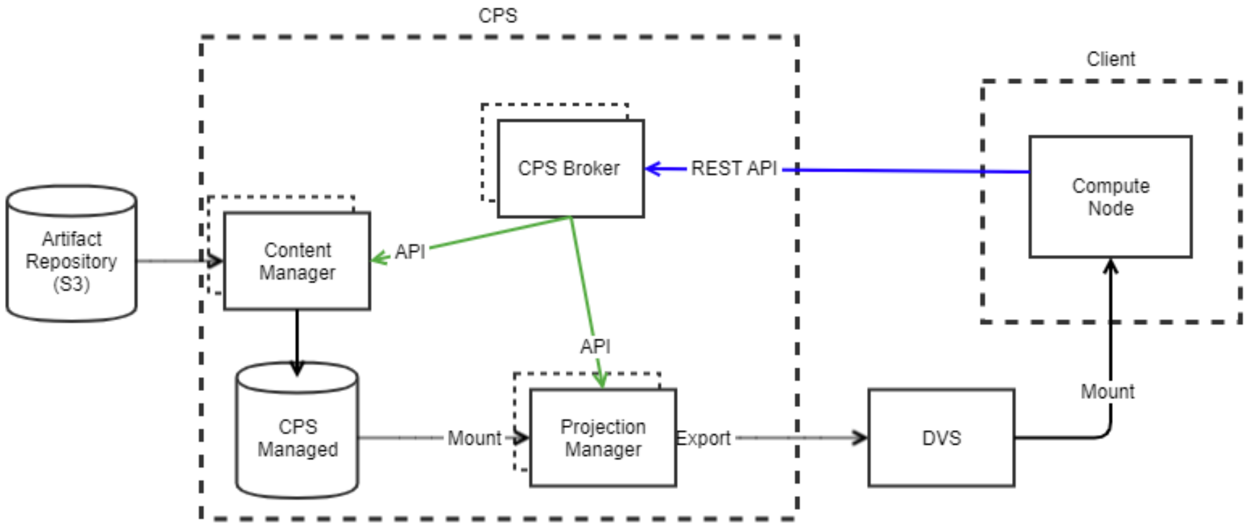


COMPUTE NODE ROOT FILE SYSTEM MOUNTS

- All files in the compute node root file system (rootfs) are provided from a squashFS image stored in S3 (Ceph)
- Compute node rootfs images are projected by CPS pods and mounted via DVS
- Rootfs images are mounted on compute nodes with `/opt/cray/cps-utils/bin/cpsmount.sh` and are mounted read-only
  - A compute node local overlay file system is configured to enable writes "on top of" the `rootfs` to an ephemeral in-memory file system
- DVS mount content is accessed over the network on demand
  - When a block is first referenced, DVS caches the content in the node-local Linux page cache so future references to that data will not involve the network
    - If available memory gets too low, Linux can evict these pages, and thus the data will be accessed over the network again (and cached again) if/when they are referenced again
    - Overlay Preload can permanently "pin" files in memory on the compute node at boot time so they can never be evicted
- DVS can also project other filesystems unrelated to CPS
  - Projections of user file systems using DVS can be configured as read-write or read-only

WHAT IS THE CONTENT PROJECTION SERVICE (CPS)

- The Content Projection Service (CPS) is a container-based microservice managed by Kubernetes
  - The main components of CPS are
    - CPS Brokers
    - Content Managers
    - Projection Managers
- At node boot Boot Script Service (BSS) provides
  - The Linux kernel
  - `initrd`
  - Boot parameter data
- CPS provides
  - Node's root file system image (operating system image)
  - HPE Cray Programming Environment (CPE) images
  - Analytics images



`cray cps contents` provides a list of images being managed by the content manager

`cray cps deployment` provides a list of CPS pods and their statuses

`cray cps transports` provides a list of images currently being exported (served) to nodes

CPS TUNING

- PodAntiAffinity ensures that there will be no more than one instance of cray-cps pods per worker node
- Scaling the number of cray-cps pods is helpful for maintaining resiliency and load-balancing
  - Default: 2 pods

```
ncn# kubectl -n services scale --current-replicas=2 --replicas=3 deployment/cray-cps
```

- Scaling the number of cm-pm pods and controller where they run is also useful for resiliency and load-balancing when using CPS
  - Default: 3 pods
  - Guidance: 1 pod supports about 512 nodes, but should have not less than 3 total
  - Can assign to specific worker nodes (in different cabinets) or let CPS choose from available worker nodes when scaling up

```
ncn# cray cps deployment update --nodes "ncn-w015,ncn-w016"
ncn# cray cps deployment update --numpods 2
```

CPS CONTENT

- Add content to CPS
  - Pre-stages the content to the cray-cps-cm-pm pods so it will be ready when the first client tries to mount the new content

```
ncn-w001# cray cps contents create --s3path s3://boot-images 08673352-fc26-4cc6-883a-f79e1ed3052b/rootfs --etag
90d7b9f298d1a638f5a80b3876691ccc-167 --transport dvs
```

- List all CPS content

```
ncn-w001# cray cps contents list
exportPath = "/var/lib/cps-local/76df050e1fde782a58365504477a7af6"
s3path = "s3://boot-images/08673352-fc26-4cc6-883a-f79e1ed3052b/rootfs"
ERROR = []
transports = [ "dvs",]
artifactID = "3c070d4f16dbd81e0c1870a751251880"
[[results.exportStatus]]
status = "ready"
type = "dvs"
[results.contentReplicas]
ready = 2
total = 2
[[results.contentReplicas.status]]
status = "ready"
replicaID = "10.252.1.5"
detail = "Artifact_id=3c070d4f16dbd81e0c1870a751251880 is ready"
[[results.contentReplicas.status]]
status = "ready"
replicaID = "10.252.1.6"
detail = "Artifact_id=3c070d4f16dbd81e0c1870a751251880 is ready"
```

- Remove CPS Content
  - Removing content downloaded by the cray-cps-cm-pm pods helps free up disk space on the nodes where those pods run

```
ncn-w001# cray cps contents delete --s3path s3://boot-images/08673352-fc26-4cc6-883a-f79e1ed3052b/rootfs
```

CLEAN UP CPS CONTENT

- CPS contents should be periodically checked and old contents removed from CPS to avoid running out of disk space
  - CPS contents expects the source data (file object) to be in the S3 storage, but they might get deleted by IMS before the CPS contents are removed especially the old contents
  - `cleanup_cps.py`
    - Can list all the CPS contents and which ones are currently DVS mounted or not used
    - Can remove CPS contents that are not in use
    - Scans all compute node
    - Scans with `--xname` option with list of comma separated xnames to search only those compute nodes
      - List at least one or two nodes for different boot images that are currently used to boot compute nodes and UAN nodes as well
  - List all contents

```
ncn-m001# cd /opt/cray/cps-utils/cps-cleanup
ncn-m001# ./cleanup_cps.py
```

  - Remove all unused contents by scanning all compute and application nodes

```
ncn-m001# ./cleanup_cps.py --delete
```

    - To delete manually instead of the above command

```
ncn-w001# cray cps contents delete --s3path s3://boot-images/08673352-fc26-4cc6-883a-
f79e1ed3052b/rootfs
```

CRAY OVERLAY PRELOAD

- Compute node root filesystem utilizes the Linux overlayfs architecture
  - Read-only lower layer that uses the Data Virtualization Service (DVS)
  - Read-write, RAM based upper layer
  - This architecture supports copying files from the lower layer to the upper layer to increase performance and support writes
- The Overlay Preload feature uses this copy operation to increase performance on frequently accessed files
  - A list of files is provided at boot, and they are all copied into local memory
  - All future references to those files are serviced by the local file system, rather than requiring remote data and/or metadata DVS operations
  - This improves system and application performance
    - However, the amount of memory available on the node is reduced by the cumulative size of all files copied into its memory
- The total amount of memory used by Overlay Preload can be configured by the system administrator to balance the performance and memory requirements of the system
- The system is shipped with a default list of files to be preloaded
  - This list is specific to the operating system release provided and the IO access recorded during system boot
  - The administrator can modify this list if desired
    - Sites may define their own file lists to optimize work for specific workloads
    - The Overlay Preload package ships with a script to aid in determining which files are accessed at boot time
      - It will analyze boot behavior and produce a list of files accessed

CONFIGURE CRAY OVERLAY PRELOAD

- Configuration Settings
  - Overlay Preload configuration is managed using the CFS overlay-preload Ansible role
    - overlay-preload-size-limit
      - The size, in MB, that limits the amount of file data that is promoted to the overlay cache
        - A value of 0 indicates ‘unlimited’ file data
- File lists
  - The list of files to be preloaded at boot are located in the file /opt/cray/overlay-preload/config/dist/overlay-preload.filelist
    - The file format is a list of file paths, one per line, with support for wildcard values
  - The file list in the default boot image may be modified
  - The file is read early in the boot process, and files will be processed in order
    - If there are constraints placed on total preload size, processing will stop once the limit is reached
      - In this case, files that are critical for preloading should be placed first
- The Overlay Preload Log File and Symlinks
  - Overlay Preload creates a log file on affected nodes at /var/log/cray/overlay-preload.log
    - The log file contains warnings for files that were not found, as well as the number and size of the files preloaded on the node
  - Any symlinks included in a file list may not be copied from the lower layer to the node-local RAM file system, which might look confusing
    - For example, if a site’s content list contains /etc/alternatives/unzip, which is a symlink to /usr/bin/unzip-plain
    - In this case, both the link and its target are present in lower layer, but neither of them appear in the node-local file system
    - This is expected and correct behavior
    - A site that is concerned about possible confusion for administrators can decide to exclude symlinks from file lists, or simply list the target of the symlink in a file list to ensure that it is present in the node-local file system

CUSTOM CRAY OVERLAY PRELOAD

- Create Custom Loads for Specific Workloads
  - Sites may define their own file lists to optimize work for specific workloads
    - Either create an Ansible play for CFS to run pre-boot for image customization or use the IMS method to jump into the image customization process via ssh to run a command
  - The following is a general workflow for this process:
    - Enable the cray-preload-strace service in the image that will be booted

```
image# systemctl enable cray-preload-strace
```

    - Boot a compute node with the new filesystem image
    - Log into the compute node as root and kill any strace process.
      - The strace log can be found at /cray-preload-strace.log.
    - Run the `preload-strace-analyze.sh` script with the strace log as input.

```
compute# preload-strace-analyze.sh /cray-preload-strace.log
```

      - The output will be a list of files, access counts, and sizes
      - The sum is included at the bottom
      - This can be used as the basis for creating or modifying an overlay file list
    - Disable the cray-preload-strace service

```
compute# systemctl disable cray-preload-strace
```

WORKLOAD MANAGEMENT

WORKLOAD MANAGEMENT

SLURM and PBS Pro

- Actively working with SchedMD and Altair on HPE Cray Ex system check-out and new APIs
- Cray providing integration through a new set of services and APIs
- Both WLMs supported
- Other WLMs can also use the same APIs

CRAY WLM SERVICES

- PALS – Parallel Application Launch Service
  - libpals is used for both PBS Pro and Slurm
  - Launcher part of PALS (mpiexec, aprun, palsd) is only used for PBS
- Application Task Orchestration and Management (ATOM)
  - application and job prologue and epilogue task runner
    - compute node cleanup
    - node health checking
    - energy usage reporting

SLURM CONFIG FOR HPE 200GB CASSINI NICS

- Set SwitchType=switch/hpe_slingshot in slurm.conf
- SwitchParameters determine behavior
  - vnis=<min>-<max> - Range of VNIs to allocate for jobs and applications
    - Default is 32768-65535.
  - tcs=<class1>[:<class2>]... - Set of traffic classes to configure for applications. Supported traffic classes are [DEDICATED_ACCESS], [LOW_LATENCY], [BULK_DATA] and [BEST_EFFORT].
  - single_node_vni=<all|user|none> - Allocates single node VNI as follows:
    - Not set - Does not allocate VNI for single-node job steps.
    - single_node_vni (no value) - Allocates a VNI for all job steps.
    - single_node_vni=all - Allocates a VNI for all job steps.
    - single_node_vni=user - Allocates a VNI for single-node job steps using the srun --network=single_node_vni option or SLURM_NETWORK=single_node_vni environment variable.
    - single_node_vni=none - Does not allocate VNI for single-node job steps.
  - job_vni=<all|user|none> - Allocates job VNI as follows:
    - Not set - Does not allocate additional VNI for jobs.
    - job_vni (no value) - Allocates an additional VNI for jobs, shared among all job steps.
    - job_vni=all - Allocates an additional VNI for jobs, shared among all job steps.
    - job_vni=user - Allocates an additional VNI for any job either using the srun --network=job_vni option or SLURM_NETWORK=job_vni environment variable.
    - job_vni=none - Does not allocate additional VNI for jobs.
    - adjust_limits - If set, slurmd sets an upper bound on network resource reservations by taking the per-NIC maximum
    - resource quantity and subtracting the reserved or used values (whichever is higher) for any system network services. This is
    - the default.

MORE SWITCHPARAMETERS

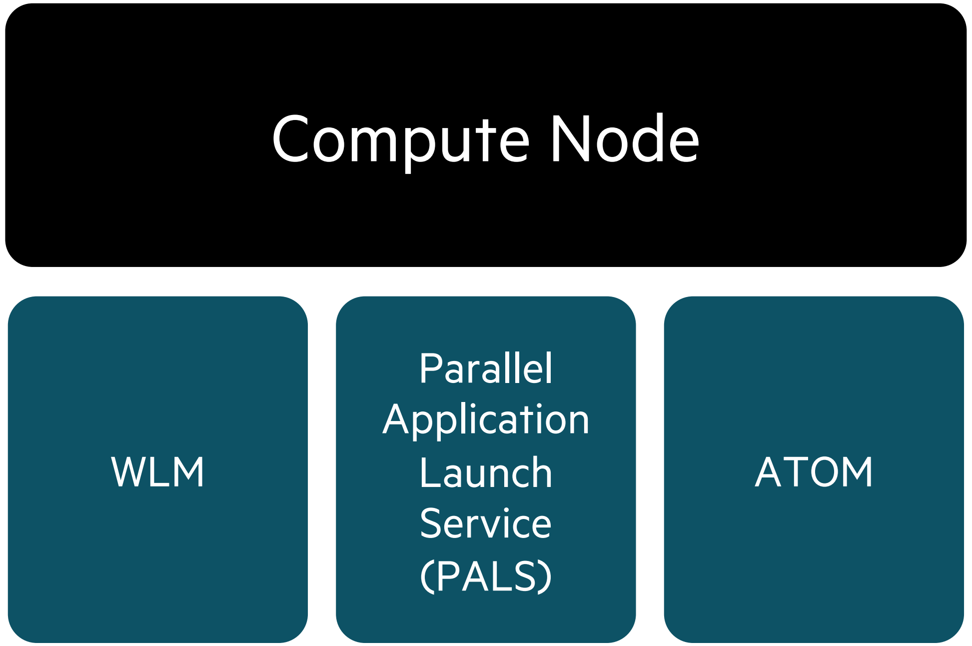
- Set SwitchType=switch/hpe_slingshot in slurm.conf
- SwitchParameters determine behavior
  - no_adjust_limits - If set, slurmd calculates network resource reservations based only upon the per-resource configuration default and number of tasks in the application; it does not set an upper bound based on resource usage of already-existing system network services. Setting no_adjust_limits can result in more application launch failures due to network resource exhaustion; but if an application requires a certain amount of resources, this option ensures it.
  - jlope_url=<url> - If set, slurmctld uses the configured URL to request Instant On NIC information, from the HPE jackalope daemon REST API, for each node in a job step.
  - jlope_auth=<BASIC|OAUTH> - HPE jackalope daemon REST API authentication type, default is OAUTH.
  - jlope_authdir=<directory> - Directory containing authentication information files. Default is /etc/jackaloped for BASIC authentication and /etc/wlm-client-auth for OAUTH authentication.
  - def_<rsrc>=<val> - Per-CPU reserved allocation for this resource.
  - res_<rsrc>=<val> - Per-node reserved allocation for this resource. If set, overrides the per-CPU allocation.
    - max_<rsrc>=<val> - Maximum per-node application for this resource.
- Resources are:
  - txqs - Transmit command queues. The default is 2 per-CPU, maximum 1024 per-node.
  - tgqs - Target command queues. The default is 1 per-CPU, maximum 512 per-node.
  - eqs - Event queues. The default is 2 per-CPU, maximum 1023 per-node.
  - cts - Counters. The default is 1 per-CPU, maximum 1023 per-node.
  - tles - Trigger list entries. The default is 1 per-CPU, maximum 2048 per-node.
  - ptes - Portable table entries. The default is 6 per-CPU, maximum 2048 per-node.
  - les - List entries. The default is 16 per-CPU, maximum 16384 per-node.
  - acs - Addressing contexts. The default is 4 per-CPU, maximum 1022 per-node.

PARALLEL APPLICATION LAUNCH SERVICE (PALS)

- Application launcher that enables WLMs to function normally
- WLM-specific plugins and configured to access the WLM interfaces
- Launch daemon (`palsd`) integrates with WLMs that have a compute node presence
  - PBS Pro's MoM
- Runs alongside the WLM daemon on the compute node
- Coordinates execution of parallel applications on multiple compute nodes
  - Treats these as a unit rather than separate processes
- Needed for WLMs that do not have a launcher or Cray PMI plugin
  - PBS Pro
- What about Slurm?
  - Already has a launcher (`srun`) and Cray PMI plugin
  - PALS will be disabled

APPLICATION TASK ORCHESTRATION AND MANAGEMENT (ATOM)

- Combines functionality of Cray XC system's compute node cleanup, node health, and RUR (Resource Usage and Reporting)
- General purpose job and application prologue and epilogue task runner
  - Configuration
  - Compute node cleanup
  - Node health testing
- ATOM is only called by PALS and WLMs
- ATOM REST API is not exposed on the network
  - Users cannot call ATOM APIs directly



WHY ATOM?

- Allows integration with PALS or the WLM compute node daemon
- Runs a task at a given time
  - ATOM service or daemon start-up (PBS Pro only)
  - Job start or end by WLM Daemon (PBS Pro and Slurm)
  - Application start or end by PALS (PBS Pro only)
- Does something if that task fails or succeeds
- Extensible and configurable by the customer
  - New tasks added by dropping in a new task configuration file
  - Runs tasks in lexical order, so sites can choose ordering
- Tasks can be disabled or enabled by site administrator or user
  - Site administrator can force some tasks to run or not permit others to be enabled
- ATOM: compute node daemon runs tasks in the configured order

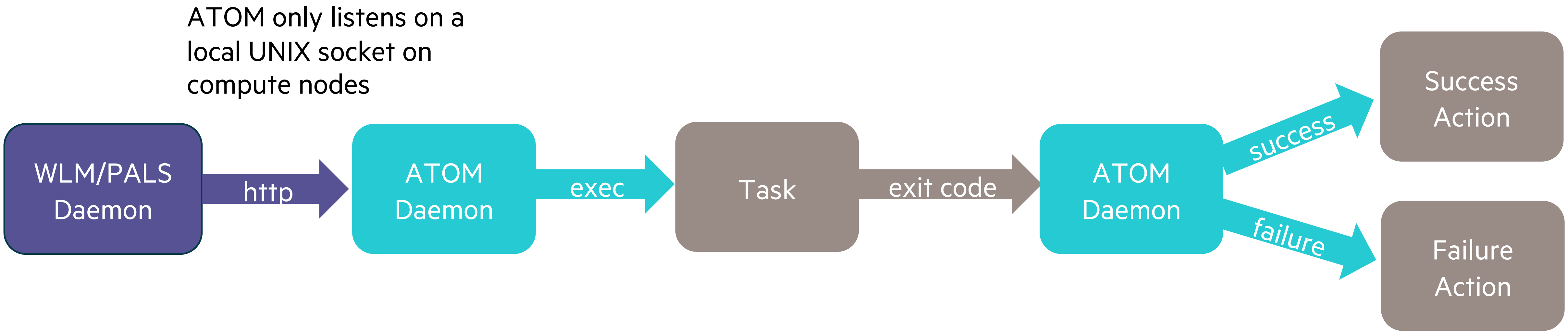
WHAT IS A TASK?

- ATOM daemon startup
  - Initialize Boot FreeMem
- Compute node cleanup
  - Clear VM/Lustre cache
  - Compact memory
- Node health
  - Free memory check
- Reporting
  - Task stats

- Any executable action that is run at a specified time
  - “On this event, run this script and if it fails, do this”
  - “On this event, run this script and if it succeeds, do this”
- Command can be inline commands or executed (Python/shell/binaries)
- Executed in filename lexical order

```
010_bootfreemem_init
{
    "name": “bootfreemem_init",
    "description": "Initialize /proc/boot_freemem",
    "onSuccess": [],
    "onFailure": [],
    "events": ["startup"],
    "timeout": 2,
    "command": ["/bin/sh", "-c", "echo 1 >/proc/boot_freemem"],
    "enabled": true,
    "userControl": false
}
```

ATOM ARCHITECTURE AND COMPONENTS


ATOM only listens on a local UNIX socket on compute nodes

- All tasks and actions run kept in a database only during a job or application's lifespan
  - Task details available through "tasks" endpoint
- All associated tasks and actions are deleted when a job or application is deleted!
- Tasks are considered successful if they exit with 0 exit status before their timeout period has elapsed
- In compute node image, `/etc/sysconfig/atomd` contains configurable variables which control file locations and settings for ATOM daemon

ATOM TASK CONFIGURATION FILE

- File names must begin with three decimal digits
  - Files are executed in numerical order
  - Configuration changes done via customizing the node image or via post-boot node personalization using Ansible JSON object with the following keys:

Key	Type	Required	Description
`name`	String	Yes	Unique task name
`description`	String	No	Human-readable task description
`onSuccess`	Array	No	List of action names to take upon successful completion
`onFailure`	Array	No	List of action names to take upon failure
`events`	Array	Yes	List of times to run this task (startup, jobStart, jobEnd, appStart, appEnd, action)
`timeout`	Number	No	Task timeout in seconds
`command`	Array	Yes	Task argv array
`enabled`	Boolean	No	Enable/disable task by default
`userControl`	Boolean	No	If true, allow users to enable/disable this task

ATOM TASK CONFIGURATION FILES

```
nid001000# ls -1 /etc/atom.d
010_bootfreemem_init.cfg
020_clear_lustre_caches.cfg
020_clear_lustre_caches_job.cfg
025_clean_tmpdirs.cfg
030_clear_vm_cache.cfg
040_compact_memory.cfg
040_compact_memory_job.cfg
090_hugepages_test.cfg
100_freemem_test.cfg
110_zeropage_test.cfg
120_pals_test.cfg
150_filesystem_test.cfg
200_energy_end.cfg
200_energy_start.cfg
800_admindown.cfg
850_reboot.cfg
900_panic.cfg
999_hello_atom.cfg
```

Example task, no actual action

ATOM TASK EXECUTION FILES

- Execution files are in /opt/cray/atom/sbin and are referenced in the "command" field
  - Test for zero page memory corruption at job end

```
nid001000# cat /etc/atom.d/110_zeropage_test.cfg
{
    "name": "zeropage_test",
    "description": "Check for zero page memory
corruption",
    "onSuccess": [],
    "onFailure": ["admindown"],
    "events": ["jobEnd"],
    "timeout": 5,
    "command": ["/opt/cray/atom/sbin/zeropage"],
    "enabled": true,
    "userControl": false,
    "exclusive": false
}
```

- Compact fragmented memory at end of every application and job so hugepage allocations remain efficient

```
nid000001# cat
/etc/atom.d/040_compact_memory.cfg
{
    "name": "compact_memory",
    "description": "Compact fragmented memory to
allow better hugepages allocation",
    "onSuccess": [],
    "onFailure": [],
    "events": ["appEnd", "jobEnd"],
    "timeout": 30,
    "command":
["/opt/cray/atom/sbin/compact_memory.py"],
    "enabled": true,
    "userControl": true
}
```

CFS CONFIGURATION FOR ATOM

- ATOM configuration is done by CFS, so add or change data in VCS (git)
  - Configuration settings can be used to specify directory paths
    - atom_filesystems
      - list of directory paths mounted on all compute nodes to check at application and job end time
    - atom_tmpdirs
      - list of directory paths to be cleaned up at job end time
  - Create the `group_vars/all/atom.yml` file in the pbs-config-management or slurm-config-management git repository
  - Edit and populate it with the desired settings. For example:

```
atom_filesystems:
  - "/scratch"

atom_tmpdirs:
  - "/tmp"
  - "/var/tmp"
  - "/dev/shm"
```

  - Can override or add new ATOM configuration files or tasks

```
roles/atom/files/config/
roles/atom/files/tasks/
```

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

SYSTEM ADMIN TOOLKIT (SAT)

- Provides filterable reports
  - Firmware and software versions
  - Hardware inventory and history
  - Current sensor data
  - System status
- Has automation for more dynamic workflows
  - Preparing boot artifacts
  - System boot and shutdown
  - Blade replacement,
  - BMC credential management
- Offers a command line utility which uses subcommands
  - Most commands require authentication to API gateway
  - Some commands require Kubernetes configuration and authentication

CRAY CLI FRAMEWORK FROM REST API SPECIFICATION

```
user@ncn> cray auth login --username UserWithAdminRole
Password:
user@ncn> cray --help
Usage: cray [OPTIONS] COMMAND [ARGS]...
  Cray management and workflow tool
Options:
  --version  Show the version and exit.
  --help     Show this message and exit.
Commands:
  init     Initialize/reinitialize the Cray CLI
```

- Documentation convention is that if the admin role is required for cray CLI or sat CLI, then the command prompt will use `hostname#` rather than `user@hostname>`
- Linux account and Keycloak authentication are different credentials

Management services which have API specifications

```
Groups:
  artifacts  Manage artifacts in S3
  auth       Manage OAuth2 credentials for the Cray CLI
  badger     Badger Service API
  bos        Boot Orchestration Service
  bss        Boot Script Service API
  capmc      Cray Advanced Platform Monitoring and Control API
  cfs        Configuration Framework Service
  config     View and edit Cray configuration properties
  cps        Content Projection Service
  crus       Compute Rolling Upgrade Service
  fas        Firmware Action Service
  hsm        Hardware State Manager API
  ims        Image Management Service
  nmd        Node Memory Dump Service
  scsd       System Configuration Service
  sls        System Layout Service
  uas        User Access Service
  vnid       Virtual Network Identifier Daemon
```

SAT CLI

- Runs on master nodes in a container using podman, a daemonless container runtime
  - Using either sat or sat bash always launches a container
  - The SAT container does not have access to the node's file system
- There are two ways to run sat
  - Interactive: Launching a container using sat bash, followed by sat commands

```
ncn-m# sat bash
 (CONTAINER-ID)sat-container# sat status
 (CONTAINER-ID)sat-container# sat hwinv
 (CONTAINER-ID)sat-container# exit
```

  - Non-interactive: Running a sat command directly on a master node

```
ncn-m# sat status
```

- Authentication using Keycloak credentials
  - `sat auth` and use Keycloak username and password per session
  - Account used needs to have admin role in Keycloak
- Man pages exist for `sat` and subcommands
  - Use to get more information on how to use options for subcommands

SAT COMMANDS

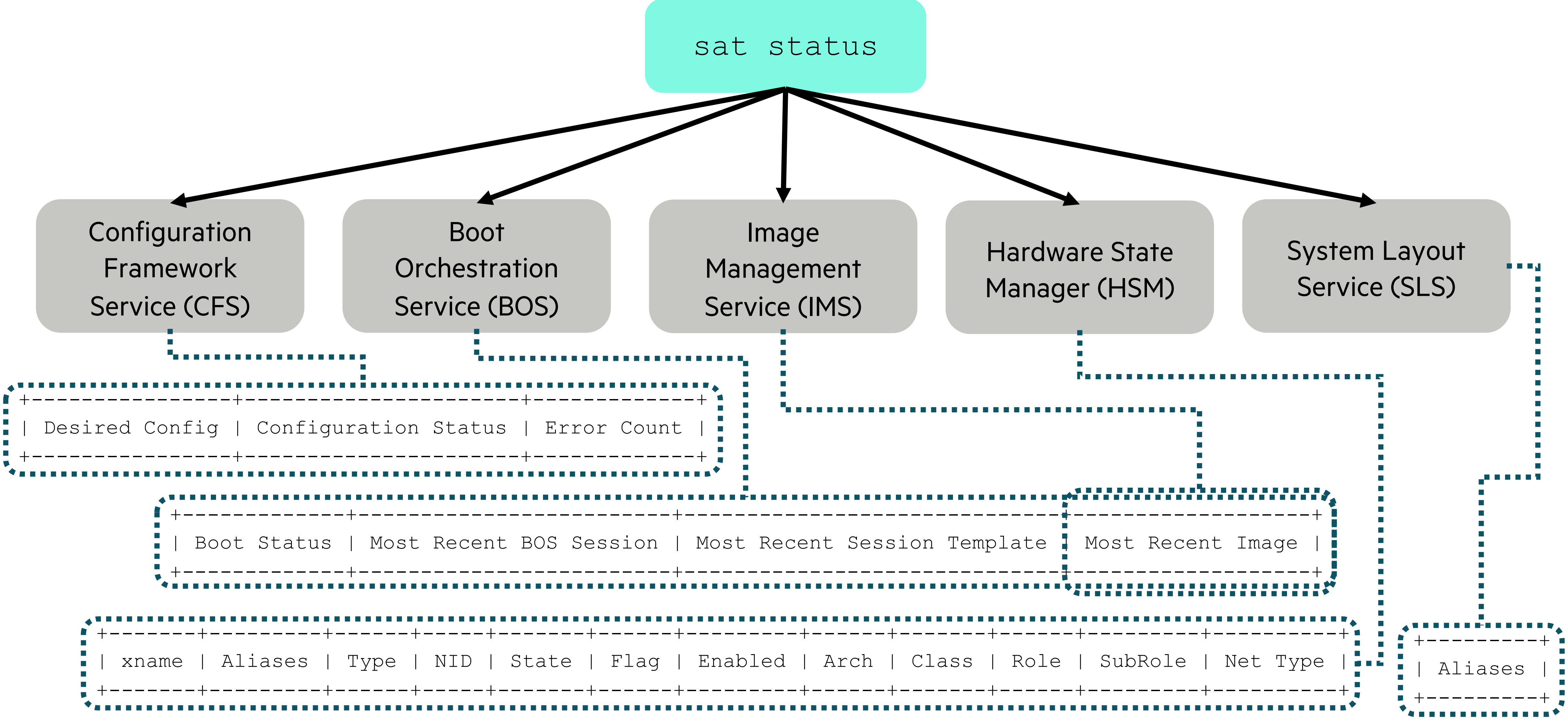
sat auth	Authenticate to the API gateway and save the token	sat k8s	Report on Kubernetes replicasets that have co-located replicas
sat bmccreds	Set BMC Redfish access credentials	sat nid2xname	Translate node IDs to node xnames
sat bootprep	Prepare to boot nodes with images and configurations	sat sensors	Report current sensor data
sat bootsys	Boot or shutdown the system (compute nodes, application nodes, and management nodes)	sat setrev	Set HPE Cray EX system revision information
sat diag	Launch diagnostics on the HSN switches and generate a report	sat showrev	Print revision information for the HPE Cray EX system
sat firmware	Report firmware version	sat slscheck	Perform a cross-check between SLS and HSM
sat hwhist	Report hardware component history	sat status	Report node status across the HPE Cray EX system
sat hwinv	Give a listing of the hardware of the HPE Cray EX system	sat swap	Prepare HSN switch or cable for replacement and bring HSN switch or cable into service
sat hwmatch	Report hardware mismatches for processors and memory	sat xname2nid	Translate node and node BMC xnames to node IDs
sat init	Create a default SAT configuration file		

Newest SAT commands

SAT STATUS API INTERACTIONS

- `sat status` gets information from the following APIs
  - Hardware State Manager (HSM)
  - System Layout Service (SLS)
  - Configuration Framework Service (CFS)
  - Boot Orchestration Service (BOS)
  - Image Management Service (IMS)
- Options exist to limit which APIs are queried (introduced in SAT 2.3)
  - `--hsm-fields`
  - `--sls-fields`
  - `--cfs-fields`
  - `--bos-fields`
    - Introduced in SAT 2.4
    - Only supported when using BOS v2 (with `--bos-version v2` or corresponding config file option)
    - This also includes the “Most Recent Image” field which is obtained using BOS and IMS

SAT STATUS API REQUESTS



`sat status`

Configuration Framework Service (CFS)

Boot Orchestration Service (BOS)

Image Management Service (IMS)

Hardware State Manager (HSM)

System Layout Service (SLS)

```
+----------------+----------------------+--------------+
| Desired Config | Configuration Status | Error Count |
+----------------+----------------------+--------------+
```

```
+-------------+----------------------+----------------------------+---------------------+
| Boot Status | Most Recent BOS Session | Most Recent Session Template | Most Recent Image |
+-------------+----------------------+----------------------------+---------------------+
```

```
+-------+---------+------+-----+-------+------+---------+------+-------+------+---------+----------+
| xname | Aliases | Type | NID | State | Flag | Enabled | Arch | Class | Role | SubRole | Net Type |
+-------+---------+------+-----+-------+------+---------+------+-------+------+---------+----------+
```

```
+---------+
| Aliases |
+---------+
```

SAT STATUS

- Shows current status of NCNs and CNs as reported by Hardware State Manager (HSM)
  - Information must be discovered by HSM
- Requires authentication to show any information

```
ncn-m# sat status --sort-by NID
+----------------+-----------+------|----------+-------+------+---------+------+----------+----------+---------+----------+
| xname          | Aliases   | Type | NID      | State | Flag | Enabled | Arch | Class    | Role     | Subrole | Net Type |
+----------------+-----------+------|----------+-------+------+---------+------+----------+----------+---------+----------+
| x3000c0s20b1n0 | nid000001 | Node | 1        | On    | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s20b2n0 | nid000002 | Node | 2        | Ready | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s20b3n0 | nid000003 | Node | 3        | On    | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s20b4n0 | nid000004 | Node | 4        | Ready | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s23b1n0 | nid000005 | Node | 5        | On    | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s23b2n0 | nid000006 | Node | 6        | Ready | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s23b3n0 | nid000007 | Node | 7        | On    | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x3000c0s23b4n0 | nid000008 | Node | 8        | On    | OK   | True    | X86  | River    | Compute  | None    | Sling    |
| x1000c0s1b0n0  | nid001004 | Node | 1004     | Ready | OK   | True    | X86  | Mountain | Compute  | None    | Sling    |
| x1000c0s1b0n1  | nid001005 | Node | 1005     | Ready | OK   | True    | X86  | Mountain | Compute  | None    | Sling    |
| x1000c0s1b1n0  | nid001006 | Node | 1006     | Ready | OK   | True    | X86  | Mountain | Compute  | None    | Sling    |
| x1000c0s1b1n1  | nid001007 | Node | 1007     | Ready | OK   | True    | X86  | Mountain | Compute  | None    | Sling    |
| x3000c0s1b0n0  | ncn-m001  | Node | 100001   | Ready | OK   | True    | X86  | River | Management  | Master  | Sling    |
| x3000c0s3b0n0  | ncn-m002  | Node | 100002   | Ready | OK   | True    | X86  | River | Management  | Master  | Sling    |
| x3000c0s5b0n0  | ncn-m003  | Node | 100003   | Ready | OK   | True    | X86  | River | Management  | Master  | Sling    |
| x3000c0s7b0n0  | ncn-w001  | Node | 100004   | Ready | OK   | True    | X86  | River | Management  | Worker  | Sling    |
| x3000c0s9b0n0  | ncn-w002  | Node | 100005   | Ready | OK   | True    | X86  | River | Management  | Worker  | Sling    |
| x3000c0s11b0n0 | ncn-w003  | Node | 100006   | Off   | OK   | True    | X86  | River | Management  | Worker  | Sling    |
| x3000c0s13b0n0 | ncn-s001  | Node | 100007   | Ready | OK   | True    | X86  | River | Management  | Storage | Sling    |
| x3000c0s15b0n0 | ncn-s002  | Node | 100008   | Ready | OK   | True    | X86  | River | Management  | Storage | Sling    |
| x3000c0s17b0n0 | ncn-s003  | Node | 100009   | Ready | OK   | True    | X86  | River | Management  | Storage | Sling    |
| x3000c0s27b0n0 | uan01     | Node | 49169248 | Off   | OK   | True    | X86  | River | Application | UAN     | Sling    |
+----------------+-----------+------|----------+-------+------+---------+------+-------+-------------+---------+----------+
```

SAT STATUS FILTERED

- Can filter by any of the columns with both “equal to” and “not equal to”
- Can remove some of the pretty printing

```
ncn-m# sat status --no-borders --filter nid=1000
 xname         Aliases   Type  NID      State   Flag  Enabled Arch Class    Role       Subrole Net Type
 x1000c0s0b0n0 nid001000 Node  1000     Ready   OK    True    X86  Mountain Compute     None   Sling
ncn-m# sat status --no-borders --no-headings --filter role=compute --filter state!=ready \
--filter enabled=true
 x1000c1s2b0n1 nid001041 Node  1041     Standby Alert True    X86  Mountain Compute     None   Sling
 x1000c2s1b0n0 nid001068 Node  1068     Off     OK    True    X86  Mountain Compute     None   Sling
 x1000c7s5b1n0 nid001246 Node  1246     On      OK    True    X86  Mountain Compute     None   Sling
ncn-m# sat status --no-borders --no-headings --filter class=river --filter role=application
 x3000c0s23b0n0 uan01    Node  49169120 Ready   OK    True    X86  River    Application UAN    Sling
```

- Can change fields displayed

```
ncn-m# sat status --no-borders --filter class=river --filter role=management \
--fields xname,aliases,nid,subrole,state
 xname          Aliases  NID    Subrole State
 x3000c0s3b0n0  ncn-m002 100002 Master  Ready
 x3000c0s7b0n0  ncn-w001 100004 Worker  Ready
 x3000c0s17b0n0 ncn-s003 100008 Storage Ready
```

- Can report status on different types of components, but default is “Node”
  - all, Chassis, ChassisBMC, ComputeModule, HSNBoard, Node, NodeBMC, NodeEnclosure, RouterBMC, RouterModule

```
ncn-m# sat status --no-borders --types RouterBMC
 xname        Type       State  Flag  Enabled  Arch  Class  Net Type
 x3000c0r21b0 RouterBMC  Ready  OK    True     X86   River  Sling
```

CHECKING SOFTWARE VERSIONS WITH KUBECTL

- Search for information in the product-catalog with jq filtering the output for only CSM

```
ncn# kubectl get cm cray-product-catalog -n services -o json | jq -r .data.csm
1.2.0:
  active: true
  configuration:
    clone_url: https://vcs.cmn.groot.dev.cray.com/vcs/cray/csm-config-management.git
    commit: 1069629a2682bb173c42c11c85d045797637806c
    import_branch: cray/csm/1.9.31
    import_date: 2022-07-12 13:54:16.725749
    ssh_url: git@vcs.cmn.groot.dev.cray.com:cray/csm-config-management.git
  images:
    cray-shasta-csm-sles15sp3-barebones.x86_64-csm-1.2:
      id: 0546c3fc-2928-497f-86ad-3d92085eb6ec
  recipes:
    cray-shasta-csm-sles15sp3-barebones.x86_64-csm-1.2:
      id: 3d2f5663-6190-4d8a-ad1d-63b09eae3fd8
```

CHECKING SOFTWARE VERSIONS WITH SAT

- Display information for all software products installed

```
ncn-m# sat showrev --products
################################################################################
Product Revision Information
################################################################################
+-------------------------+------------------+--------+----------------------------------------------------+----------------------------------------------------+
| product_name            | product          | active | images                                             | image_recipes                                      |
|                         |_version          |        |                                                    |                                                    +
+-------------------------+------------------+-------------------------------------------------------------+----------------------------------------------------+
| analytics               | 1.1.28           | N/A    | Cray-Analytics.x86_64-base                         | -                                                  |
| cos                     | 2.3.101          | N/A    | cray-shasta-compute-sles15sp3.x86_64-2.3.33        | cray-shasta-compute-sles15sp3.x86_64-2.3.33        |
| cpe                     | 21.12.3          | N/A    | cpe-barebones-sles15sp3.x86_64-21.12.2             | cpe-barebones-sles15sp3.x86_64-21.12.2             |
| cpe                     | 22.3.1           | N/A    | cpe-barebones-sles15sp3.x86_64-22.03.0             | cpe-barebones-sles15sp3.x86_64-22.03.0             |
| cpe                     | 22.6.6           | N/A    | cpe-barebones-sles15sp3.x86_64-22.06.4             | cpe-barebones-sles15sp3.x86_64-22.06.4             |
| cray-sdu-rda            | 2.0.0            | N/A    | -                                                  | -                                                  |
| csm                     | 1.0.11           | N/A    | cray-shasta-csm-sles15sp3-barebones.x86_64-csm-1.2 | cray-shasta-csm-sles15sp3-barebones.x86_64-csm-1.2 |
| hfp                     | 22.05.7          | N/A    | -                                                  | -                                                  |
| sat                     | 2.3.4            | True   | -                                                  | -                                                  |
| sle-os-backports-15-sp3 | 22.03.0          | N/A    | -                                                  | -                                                  |
| sle-os-products-15-sp3  | 22.03.0          | N/A    | -                                                  | -                                                  |
| sle-os-updates-15-sp3   | 22.03.0          | N/A    | -                                                  | -                                                  |
| slingshot               | 1.7.3-1934       | N/A    | -                                                  | -                                                  |
| slingshot-host-software | 1.7.3-55         | N/A    | -                                                  | -                                                  |
| slingshot-host-software | 1.7.3-55_cos-2.3 | N/A    | -                                                  | -                                                  |
| slingshot-host-software | 1.7.3-55_csm-1.2.0| N/A   | -                                                  | -                                                  |
| slurm                   | 1.1.10           | N/A    | -                                                  | -                                                  |
| sma                     | 1.6.22           | N/A    | -                                                  | -                                                  |
| uan                     | 2.4.3            | N/A    | -                                                  | -                                                  |
```

QUERYING HARDWARE INVENTORY

`sat` supports tab completion! From the podman pod, `sat bash`, but not from the `sat` CLI. Hitting tab twice provides a list of options

```
ncn-m# sat bash
(cab2475ed202) sat-container:/sat # source /etc/bash_completion.d/sat-completion.bash
(cab2475ed202) sat-container:/sat # sat hwinv --list-
--list-all             --list-drives            --list-node-accels                      --list-nodes
--list-chassis         --list-hsn-boards        --list-node-enclosure-power-supplies   --list-procs
--list-cmm-rectifiers  --list-mems              --list-node-enclosures                  --list-router-modules
--list-compute-modules --list-node-accel-risers  --list-node-hsn-nics
(cab2475ed202) sat-container:/sat # sat hwinv --list-nodes --node-fields xname,serial_number,memory_size
################################################################################
Listing of all nodes
################################################################################
+----------------+------------------+-------------------+
| xname          | Serial Number    | Memory Size (GiB) |
+----------------+------------------+-------------------+
| x1000c0s1b0n0  | HR19380063       | 256.0             |
| x1000c0s1b0n1  | HR19380063       | 256.0             |
| x1000c0s5b0n0  | HR19380023       | 256.0             |
(cab2475ed202) sat-container:/sat # sat hwinv --list-router-modules
################################################################################
Listing of all router modules
################################################################################
+-----------+--------------+
| xname     | Manufacturer |
+-----------+--------------+
| x1000c0r3 | Cray Inc     |
| x1000c0r7 | Cray Inc     |
```

SLINGSHOT SWITCH OR CABLE REPLACEMENT

- Disable a Slingshot switch before maintenance or enable a switch after maintenance is complete.

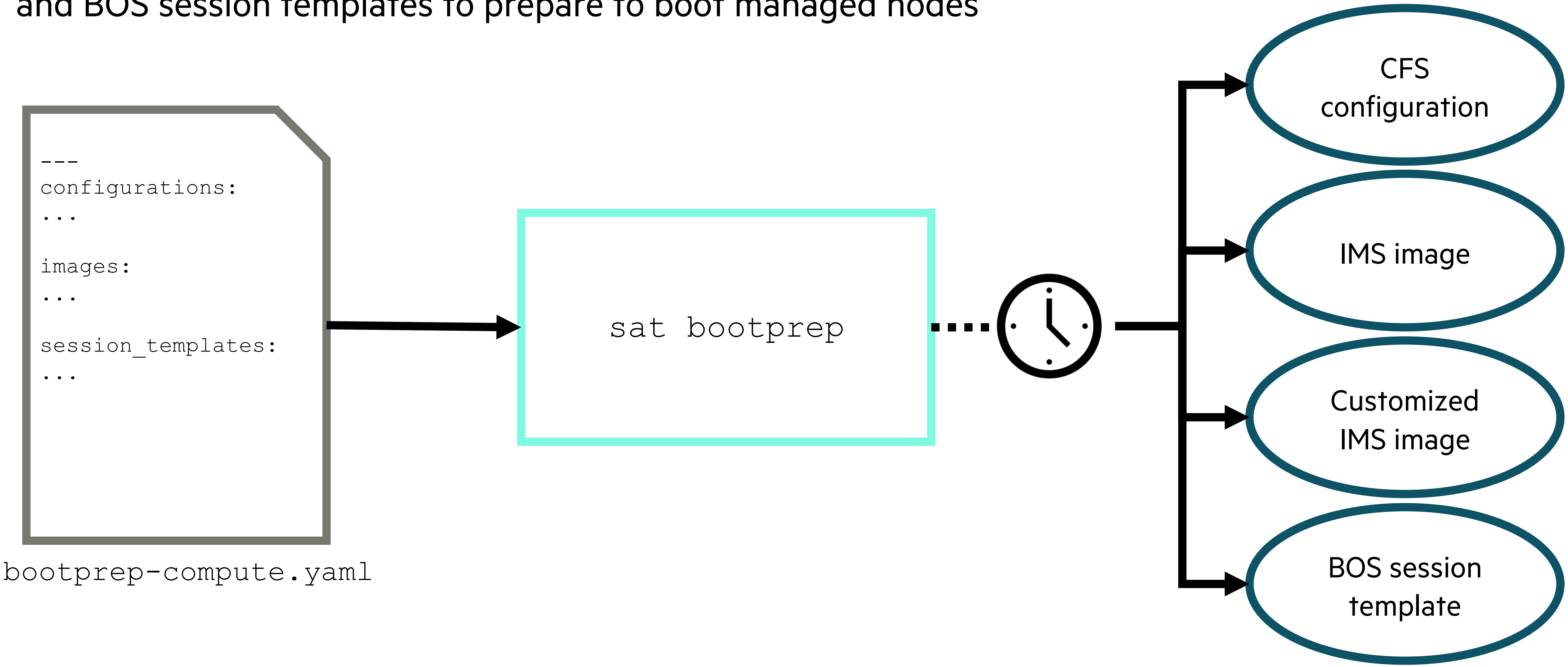
```
ncn-m# sat swap switch --dry-run x1000c3r3
Ports: x1000c3r3j104p1 x1000c3r3j105p0 x1000c3r3j105p1 x1000c3r3j106p0 x1000c3r3j106p1
x1000c3r3j107p0 x1000c3r3j107p1 x1000c3r3j100p1 x1000c3r3j101p0 x1000c3r3j101p1
x1000c3r3j102p0 x1000c3r3j102p1 x1000c3r3j103p0 x1000c3r3j103p1 x1000c3r3j104p0
x1000c3r3j100p0 x1000c3r3j9p0 x1000c3r3j8p1 x1000c3r3j8p0 x1000c3r3j6p1 x1000c3r3j6p0
x1000c3r3j4p1 x1000c3r3j4p0 x1000c3r3j2p1 x1000c3r3j2p0 x1000c3r3j22p1 x1000c3r3j22p0
x1000c3r3j20p1 x1000c3r3j20p0 x1000c3r3j24p1 x1000c3r3j24p0 x1000c3r3j18p1
x1000c3r3j18p0 x1000c3r3j16p1 x1000c3r3j12p0 x1000c3r3j11p1 x1000c3r3j10p1
x1000c3r3j11p0 x1000c3r3j10p0 x1000c3r3j16p0 x1000c3r3j14p1 x1000c3r3j14p0
x1000c3r3j13p1 x1000c3r3j12p1 x1000c3r3j13p0 x1000c3r3j9p1
Dry run completed with no action to enable/disable switch.
```

- Determine all linked ports from a single jack

```
ncn-m# sat swap cable --dry-run x5000c1r3j16
Ports: x5000c1r3j16p0 x5000c3r7j18p0 x5000c1r3j16p1 x5000c3r7j18p1
Dry run completed with no action to enable/disable cable.
```

SAT BOOTPREP HIGH-LEVEL DIAGRAM

- `sat bootprep` automates the creation of CFS configurations, IMS images, and BOS session templates to prepare to boot managed nodes



```
---
configurations:
...

images:
...

session_templates:
...
```

bootprep-compute.yaml

SAT BOOTPREP YAML

- Create a sample bootprep file with proper sections

```
ncn-m# sat bootprep generate-example
ncn-m# cp example-bootprep-input.yaml bootprep-input.yaml
ncn-m# vi bootprep-input.yaml
ncn-m# sat bootprep run bootprep_input.yaml
```

- SAT 2.4 (22.11.2 recipe) has several bootprep yaml files in the hpc-csm-software-recipe git repository

```
ncn-m# ls -l bootprep product_vars.yaml
-rw-r--r-- 1 root root  511 Apr 26 15:32 product_vars.yaml

bootprep:
total 28
-rw-r--r-- 1 root root 6711 Apr 28 14:21 compute-and-uan-bootprep.yaml
-rw-r--r-- 1 root root 1112 Mar 30 10:24 management-bootprep-image-customization.yaml
-rw-r--r-- 1 root root 2856 Apr 26 15:48 management-bootprep-node-personalization.yaml
-rw-r--r-- 1 root root 2665 Mar 28 08:39 management-bootprep.yaml
```

SAT COMPUTE-AND-UAN-BOOTPREP.YAML 1

```
# (C) Copyright 2022 Hewlett Packard Enterprise Development LP
---
schema_version: 1.0.2
configurations:
- name: compute-{{recipe.version}}
  layers:
  - name: shs-cassini_install-integration-{{shs.version}}
    playbook: shs_cassini_install.yml
    product:
      name: slingshot-host-software
      version: "{{shs.version}}"
      branch: integration-{{shs.version}}
  - name: cos-compute-integration-{{cos.version}}
    playbook: cos-compute.yml
    product:
      name: cos
      version: "{{cos.version}}"
      branch: integration-{{cos.version}}
  - name: csm-packages-integration-{{csm.version}}
    playbook: csm_packages.yml
    product:
      name: csm
      version: "{{csm.version}}"
  - name: csm-diags-compute-{{csm_diags.version}}
    playbook: csm-diags-compute.yml
    product:
      name: csm-diags
      version: "{{csm_diags.version}}"
  - name: sma-ldms-compute-{{sma.version}}
    playbook: sma-ldms-compute.yml
    product:
      name: sma
      version: "{{sma.version}}"
      branch: integration-{{sma.version}}
  - name: cpe-pe_deploy-integration-{{cpe.version}}
    playbook: pe_deploy.yml
    product:
      name: cpe
      version: "{{cpe.version}}"
      branch: cpe-{{cpe.version.split('.')[0]}}.{{cpe.version.split('.')[1].zfill(2)}}-integration
#  - name: analytics-site-integration-{{analytics.version}}
#    playbook: site.yml
#    product:
#      name: analytics
#      version: "{{analytics.version}}"
#      branch: integration
  - name: slurm-site-{{slurm.version}}
    playbook: site.yml
    product:
      name: slurm
      version: "{{slurm.version}}"
      branch: integration-{{slurm.version}}
  - name: cos-compute-last-integration-{{cos.version}}
    playbook: cos-compute-last.yml
    product:
      name: cos
      version: "{{cos.version}}"
      branch: integration-{{cos.version}}

- name: uan-{{recipe.version}}
  layers:
  (SAME type of layers)

#- name: gpu-{{recipe.version}}
#  layers:
#  - name: cos-gpu-customize-playbook-{{cos.version}}
#    playbook: gpu_customize_playbook.yml
#    product:
#      name: cos
#      version: "{{cos.version}}"
#      branch: integration
```

SAT COMPUTE-AND-UAN-BOOTPREP.YAML 2

```
images:
- name: "{{base.name}}"
  ref_name: base_cos_image
  base:
    product:
      name: cos
      type: recipe
      version: "{{cos.version}}"

- name: compute-{{base.name}}
  ref_name: compute_image
  base:
    image_ref: base_cos_image
  configuration: compute-{{recipe.version}}
  configuration_group_names:
  - Compute

# NOTE: In order for this image to contain GPU content you need the GPU content available in Nexus.
# NOTE: On a worker node check by running `gpu-nexus-tool repo check -v <vendor>` (e.g. AMD, Nvidia).
# NOTE: Additionally, you need to also uncomment the configuration gpu-{{recipe.version}} above.
#- name: gpu-image
#  base:
#    image_ref: compute_image
#  configuration: gpu-{{recipe.version}}
#  configuration_group_names:
#  - Compute

- name: uan-{{base.name}}
  ref_name: uan_image
  base:
    image_ref: base_cos_image
  configuration: uan-{{recipe.version}}
  configuration_group_names:
  - Application
  - Application_UAN

session_templates:
- name: compute-{{recipe.version}}
  image:
    image_ref: compute_image
  configuration: compute-{{recipe.version}}
  bos_parameters:
    boot_sets:
      compute:
        kernel_parameters: ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN} lnm=full lnm.cpu=0 no_hz_full=1-223
rcu_nocbs=1-223 rcu_nocb_poll cxi_core.disable_default_svc=0 cxi_core.enable_fgfc=1 cxi_core.ioi_enable=0
        node_roles_groups:
        - Compute
        rootfs_provider_passthrough: "dvs:api-gw-service-nmn.local:300:hsn0,nmn0:0"

- name: uan-{{recipe.version}}
  image:
    image_ref: uan_image
  configuration: uan-{{recipe.version}}
  bos_parameters:
    boot_sets:
      uan:
        kernel_parameters: spire_join_token=${SPIRE_JOIN_TOKEN} cxi_core.disable_default_svc=0 cxi_core.enable_fgfc=1
cxi_core.ioi_enable=0
        node_roles_groups:
        - Application
        rootfs_provider_passthrough: "dvs:api-gw-service-nmn.local:300:hsn0,nmn0:0"

        # If BOS v2 will be used to create the session from this session template,
        # you can target the UAN subrole here instead. E.g.:
        #
        # node_roles_groups:
        # - Application_UAN
        #
        # If not using BOS v2, and the system has other nodes with the
        # "Application" role in HSM that are not UANs, use node_list instead of
        # node_roles_groups. E.g.:
        #
        # node_list:
        # - xname1
        # - xname2
```

SAT BOOTPREP RUN

```
ncn-m# sat bootprep run compute-and-uan-bootprep.yaml
```

CFS creating configurations

```
INFO: Validating given input file compute-and-uan-bootprep.yaml
INFO: Input file successfully validated against schema
INFO: Creating 2 CFS configuration(s)
INFO: Creating CFS configuration with name "compute-22.11.2"
INFO: Creating CFS configuration with name "uan-22.11.2”
```

IMS building recipes

```
INFO: Using IMS public key with id b7a7edc8-9a32-4e32-85ce-0fc1ff76ce71
INFO: image at index 1 depends on image at index 0.
INFO: image at index 2 depends on image at index 0.
INFO: Found IMS base for image at index 0: recipe provided by version 2.4.109 of product cos
INFO: Found IMS base for image at index 1: image from input instance with ref_name="base_cos_image"
INFO: Found IMS base for image at index 2: image from input instance with ref_name="base_cos_image"
INFO: Of the 3 that will be created, 1 have no dependencies and will be created first.
INFO: Creating 3 images.
INFO: Creating images
INFO: Launching IMS job to create image
INFO: Created IMS image creation job with ID 4a6aedcc-da17-4337-947d-1eb8b527b45f
INFO: Creation of image cray-shasta-compute-sles15sp4.x86_64-2.4.17 succeeded: ID 29ef5db0-29b6-40e1-8219-5fb0909a0146
INFO: Image cray-shasta-compute-sles15sp4.x86_64-2.4.17 does not need configuration.
INFO: Creation of image cray-shasta-compute-sles15sp4.x86_64-2.4.17 succeeded: ID 29ef5db0-29b6-40e1-8219-5fb0909a0146
INFO: Base for image with name uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17 is a pre-built image.
INFO: Base for image with name compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17 is a pre-built image.
```

CFS customizing images

```
INFO: Creating CFS session sat-8302d116-42c1-41a2-b326-489b4b538c30 to configure image uan-cray-shasta-compute-
sles15sp4.x86_64-2.4.17
INFO: Created CFS session sat-8302d116-42c1-41a2-b326-489b4b538c30 to configure image uan-cray-shasta-compute-
sles15sp4.x86_64-2.4.17
INFO: Waiting for CFS to create Kubernetes job associated with session sat-a5e8f2c7-5c03-40e5-afba-025986334cb9.
INFO: Creating CFS session sat-6543c6a0-007a-4957-ad1a-8a906146bf53 to configure image compute-cray-shasta-compute-
sles15sp4.x86_64-2.4.17
INFO: Created CFS session sat-6543c6a0-007a-4957-ad1a-8a906146bf53 to configure image compute-cray-shasta-compute-
sles15sp4.x86_64-2.4.17
INFO: Waiting for CFS to create Kubernetes job associated with session sat-740406c1-6890-484c-8887-b8583d6271e5.
INFO: CFS session: sat-a5e8f2c7-5c03-40e5-afba-025986334cb9      Image: uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container git-clone   transitioned to succeeded
INFO:     Container istio-init  transitioned to succeeded
INFO:     Container ansible     transitioned to running
INFO:     Container inventory   transitioned to running
INFO:     Container istio-proxy transitioned to running
INFO:     Container teardown    transitioned to running
INFO: CFS session: sat-740406c1-6890-484c-8887-b8583d6271e5      Image: compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container git-clone   transitioned to succeeded
INFO:     Container istio-init  transitioned to succeeded
INFO:     Container ansible     transitioned to running
INFO:     Container inventory   transitioned to running
INFO:     Container istio-proxy transitioned to running
INFO:     Container teardown    transitioned to running
INFO: CFS session: sat-740406c1-6890-484c-8887-b8583d6271e5      Image: compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container inventory   transitioned to succeeded from running
INFO: CFS session: sat-a5e8f2c7-5c03-40e5-afba-025986334cb9      Image: uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container inventory   transitioned to succeeded from running
INFO: CFS session: sat-740406c1-6890-484c-8887-b8583d6271e5      Image: compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container ansible     transitioned to succeeded from running
INFO: CFS session: sat-a5e8f2c7-5c03-40e5-afba-025986334cb9      Image: uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container ansible     transitioned to succeeded from running
INFO: CFS session: sat-740406c1-6890-484c-8887-b8583d6271e5      Image: compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container teardown    transitioned to succeeded from running
INFO: CFS session: sat-740406c1-6890-484c-8887-b8583d6271e5      Image: compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container istio-proxy transitioned to succeeded from running
INFO: Renaming configured image with ID 132829e6-c071-4ca8-97c5-286a3f7be599 to compute-cray-shasta-compute-
sles15sp4.x86_64-2.4.17
INFO: Deleting image with ID 613ef053-46d5-4680-a072-0d024d797f8a which was overwritten by a new image named compute-cray-
shasta-compute-sles15sp4.x86_64-2.4.17
INFO: Creation of image compute-cray-shasta-compute-sles15sp4.x86_64-2.4.17 succeeded: ID b63c7856-5771-49ad-80b7-
c07b2e0ee40f
INFO: CFS session: sat-a5e8f2c7-5c03-40e5-afba-025986334cb9      Image: uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container teardown    transitioned to succeeded from running
INFO: CFS session: sat-a5e8f2c7-5c03-40e5-afba-025986334cb9      Image: uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17:
INFO:     Container istio-proxy transitioned to succeeded from running
INFO: Renaming configured image with ID 204aa950-277f-417c-8922-232019b6b9f0 to uan-cray-shasta-compute-sles15sp4.x86_64-
2.4.17
INFO: Deleting image with ID 5d4c1224-fe15-4e8e-8504-e8cc08a0a5b5 which was overwritten by a new image named uan-cray-
shasta-compute-sles15sp4.x86_64-2.4.17
INFO: Creation of image uan-cray-shasta-compute-sles15sp4.x86_64-2.4.17 succeeded: ID 4d09554c-0fd7-4d81-986d-08ee61f427d0^M
INFO: Image creation completed successfully
```

BOS creating BOS session templates with customized images

```
INFO: Creating 2 BOS session templates
INFO: Creating BOS session template with the name “compute-22.11.2”
INFO: Creating BOS session template with the name “uan-22.11.2”
```

FIRMWARE REPORTING

Node controller (or BMC) for two liquid-cooled nodes

```
ncn-m# sat firmware -x x1000c0s0b0
+-------------+--------------------------+--------------------------+------------------------------------------------------+
| xname       | name                     | target_name              | version                                              |
+-------------+--------------------------+--------------------------+------------------------------------------------------+
| x1000c0s0b0 | Node0.ManagementEthernet | Node0.ManagementEthernet | wnc.i210-p2sn01                                      |
| x1000c0s0b0 | Bootloader               | Bootloader               | 1.10-wnc                                             |
| x1000c0s0b0 | FPGA2                    | mFPGA1                   | 1.05                                                 |
| x1000c0s0b0 | BMC                      | BMC                      | nc.1.5-31-shasta-release.arm.2021-11.-               |
|                                                                                                03T03:49:30+00:00.b9ced71 |
| x1000c0s0b0 | FPGA1                    | mFPGA0                   | 1.05                                                 |
| x1000c0s0b0 | Node1.BIOS               | Node1.BIOS               | ex425.bios-1.6.1                                     |
| x1000c0s0b0 | Node0.BIOS               | Node0.BIOS               | ex425.bios-1.6.1                                     |
| x1000c0s0b0 | FPGA0                    | nFPGA                    | 5.02                                                 |
| x1000c0s0b0 | Recovery                 | Recovery                 | nc.1.5-31-shasta-release.arm.2021-11-                |
|                                                                                                03T03:49:30+00:00.b9ced71 |
| x1000c0s0b0 | Node1.ManagementEthernet | Node1.ManagementEthernet | wnc.i210-p2sn01                                      |
+-------------+--------------------------+--------------------------+------------------------------------------------------+
```

FIRMWARE REPORTING WITH XNAME LIST

List of xnames: cabinet controller and Slingshot switch

```
ncn-m# sat firmware -x x1003c6b0,x3001c0r11b0
+--------------+------------+---------------+-----------------------------------------------------------------------+
| xname        | name       | target_name   | version                                                               |
+--------------+------------+---------------+-----------------------------------------------------------------------+
| x1003c6b0    | Recovery   | Recovery      | cc.1.5-31-shasta-release.arm64.2021-11-03T03:50:18+00:00.b9ced71      |
| x1003c6b0    | Rectifier1 | Rectifier 1   | PFC_01.03-SEC_02.10                                                   |
| x1003c6b0    | Bootloader | Bootloader    | 1.7-cc-pass4                                                          |
| x1003c6b0    | Rectifier0 | Rectifier 0   | PFC_01.03-SEC_02.10                                                   |
| x1003c6b0    | BMC        | BMC           | cc.1.5-31-shasta-release.arm64.2021-11-03T03:50:18+00:00.b9ced71      |
| x1003c6b0    | FPGA0      | cFPGA         | 3.03                                                                  |
| x1003c6b0    | Rectifier2 | Rectifier 2   | PFC_01.03-SEC_02.10                                                   |
| x3001c0r11b0 | BMC        | BMC           | sc.1.7.0-45-slingshot-release.arm64.2022-03-05T22:28:42+00:00.9a31838 |
| x3001c0r11b0 | Recovery   | Recovery      | rec.1.4.22-shasta-release.arm64.2021-04-26T23:22:15+00:00.79c40dd     |
| x3001c0r11b0 | FPGA0      | sFPGA-ROS     | 1.08                                                                  |
| x3001c0r11b0 | Packages   | Packages      | na                                                                    |
| x3001c0r11b0 | Bootloader | Bootloader    | 1.9-sc-ros-tor                                                        |
| x3001c0r11b0 | FPGA1      | sFPGA-ROS-TOR | 1.04                                                                  |
+--------------+------------+---------------+-----------------------------------------------------------------------+
```

CHECK SENSORS

- Obtain sensor readings from BMCs (ChassisBMC, NodeBMC, RouterBMC)
  - Limit the telemetry topics queried to the topics listed
  - The default is to query all topics:
    - cray-telemetry-temperature, cray-telemetry-voltage, cray-telemetry-power, cray-telemetry-energy, cray-telemetry-fan, cray-telemetry-pressure

```
ncn-m# sat sensors -x x1003c2s6b1 -t NodeBMC -b 2 --timeout 10 --topic cray-telemetry-temperature
Telemetry data being collected for x1003c2s6b1
Please be patient...
Waiting for metrics for all requested xnames from cray-telemetry-temperature.
Receiving metrics from stream: cray-telemetry-temperature...
Telemetry data received from cray-telemetry-temperature for all requested xnames.
+-------------+---------+----------------------------+--------------------------------+---------------+---------+------------------+---------+------+
| xname       | Type    | Topic                      | Timestamp                 | Location | Parental Context | Physical Context | Index |Value |
+-------------+---------+----------------------------+--------------------------------+---------------+---------+------------------+---------+------+
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.079525696Z | x1003c2s6b1n0 | Chassis | VoltageRegulator | 0 | 55.4 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:56.585058025Z | x1003c2s6b1n0 | Chassis | VoltageRegulator | 2 | 45.8 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.081500532Z | x1003c2s6b1n1 | Chassis | VoltageRegulator | 0 | 51.2 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:56.580577726Z | x1003c2s6b1n1 | Chassis | VoltageRegulator | 2 | 45.8 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.072975044Z | x1003c2s6b1n0 | MISSING | CPU              | 0    | 30.875000 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.072913765Z | x1003c2s6b1n0 | MISSING | CPU              | 1    | 26.500000 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.073033042Z | x1003c2s6b1n1 | MISSING | CPU              | 0    | 29.750000 |
| x1003c2s6b1 | NodeBMC | cray-telemetry-temperature | 2022-04-01T18:17:57.073074561Z | x1003c2s6b1n1 | MISSING | CPU              | 1    | 27.500000 |
+-------------+---------+----------------------------+--------------------------------+---------------+---------+------------------+---------+------+
```

TRANSLATE XNAME AND NID

```
ncn-m# sat bash
(1e2360e3e3f0) sat-container:/sat # sat status | head -4
+---------------+-----------+------+------+-------+------+---------+------+----------+---------+---------+--------+
| xname         | Aliases   | Type | NID  | State | Flag | Enabled | Arch | Class    | Role    | Subrole | NetType|
+---------------+-----------+------+------+-------+------+---------+------+----------+---------+---------+--------+
| x1000c0s0b0n0 | nid001000 | Node | 1000 | Ready | OK   | True    | X86  | Mountain | Compute | None    | Sling  |

(1e2360e3e3f0) sat-container:/sat # sat xname2nid x1000c0s0b0n0
nid001000
(1e2360e3e3f0) sat-container:/sat # sat nid2xname 1000
x1000c0s0b0n0
(1e2360e3e3f0) sat-container:/sat # sat xname2nid x1000c0s0b0
nid001000,nid001001
```

This BMC has two nodes which would be affected by hardware work

```
 (1e2360e3e3f0) sat-container:/sat # sat xname2nid x3000c0s19,x1000c0s0b0n0
 nid[000001-000004,1000]
```

Recursively expand slot, chassis, and cabinet xnames to a range of nids

```
(1e2360e3e3f0) sat-container:/sat # sat xname2nid -f nid x3000c0s19,x1000c0s0b0n0
nid000001,nid000002,nid000003,nid000004,nid001000
```

Recursively expand slot, chassis, and cabinet xnames to a list of nids

TRACK HARDWARE

- Display hardware component history by xname or Field-Replaceable Unit (FRU) ID by querying HSM
  - FRU ID was added to output of `sat hwinv`

```
ncn-m# sat hwhist --help
usage: sat hwhist [-h] [-f PATH] [-x XNAME] [--format {pretty,yaml,json}] [--no-borders] [--no-headings]
       [--reverse] [--sort-by FIELD] [--show-empty] [--show-missing] [--fields FIELDS] [--filter QUERY]
       [--by-fru] [--fruid FRUID]

Report hardware component history.

optional arguments:
  -h, --help            show this help message and exit
  --by-fru              Display hardware component history by FRU.
  --fruid FRUID, --fruids FRUID
                        A comma-separated list of FRUIDs to include in the hardware component history report.
xnames:
  Options for specifying target xnames.

  -f PATH, --xname-file PATH
     Path to a newline-delimited file of xnames. In order to share the path between the host and container
     when sat is run in a container environment, the path should be either an absolute or relative path of
     a file in or below the home or current directory. Overrides value set in config file.
  -x XNAME, --xname XNAME, --xnames XNAME
     Specify an xname on which to operate. Multiple xnames may be specified via comma-separated entries or
     by providing this option multiple times.
```

ARGO – NODE LIFECYCLE SERVICE (NLS)

- Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes
  - https://argoproj.github.io/workflows/
  - Implemented as a Kubernetes CRD (Custom Resource Definition)
  - Easily orchestrate highly parallel jobs on Kubernetes
- Define workflows where each step in the workflow is a container
- Model multi-step workflows as a sequence of tasks or capture the dependencies between tasks using a graph (Directed Acyclic Graph)
- Argo UI with CSM
  - Requires authentication with Keycloak
  - Useful for watching the progress of an install or upgrade and debugging
- NLS workflows
  - Once a workflow is started, it will proceed through multiple steps in a set order
  - Most steps depend on previous steps and will wait for its dependencies to finish before starting
  - If any step fails, by default, that step will be continuously retried until it succeeds
    - There are two ways to make Argo not continuously retry a failed step
  - Logs in the Argo UI show output from individual stages of a workflow and are useful for debugging

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

TROUBLESHOOTING BOOT FAILURES

- Booting process
  - Booting overview
  - Boot Script Service (BSS)
  - Content Projection Services (CPS)
  - Boot Orchestration Service (BOS)
- Logs
  - Console logs and access
  - SMA-Kibana
- Troubleshooting tips

BOOTING PROCESS

- Booting overview
- Boot Script Service
- Content Projection Service
- Boot Orchestration Service

BOOT FLOWCHART WITH BOS AND S3

The Boot Orchestration Service (BOS) is responsible for booting, configuring, or shutting down collections of nodes.

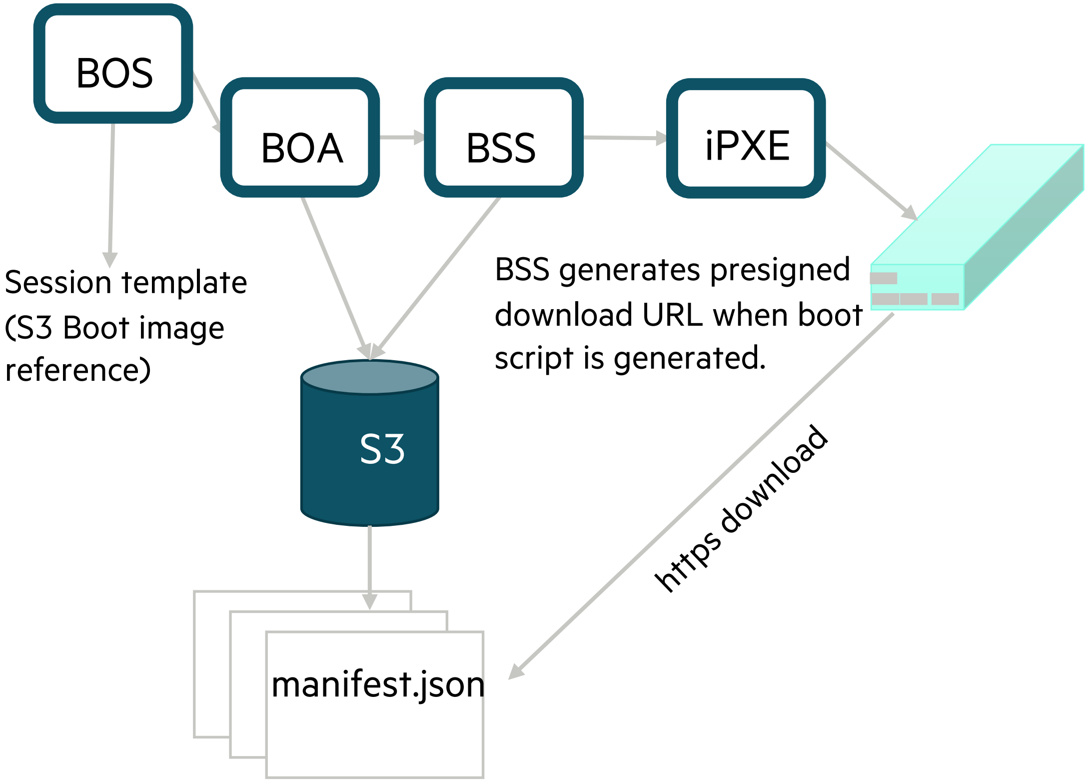
The Boot Orchestration Service has the following components:

- **Boot Orchestration Session Template** – a collection of one or more boot set objects
    - A boot set defines a collection of nodes and the information about the boot artifacts and parameters
- **Boot Orchestration Session** – An instance of a BOS operation that manages Boot Orchestration Agents
- **Boot Orchestration Agent** (BOA) – Executes actions submitted to the BOS API

BOS coordinates with several services to boot compute nodes:

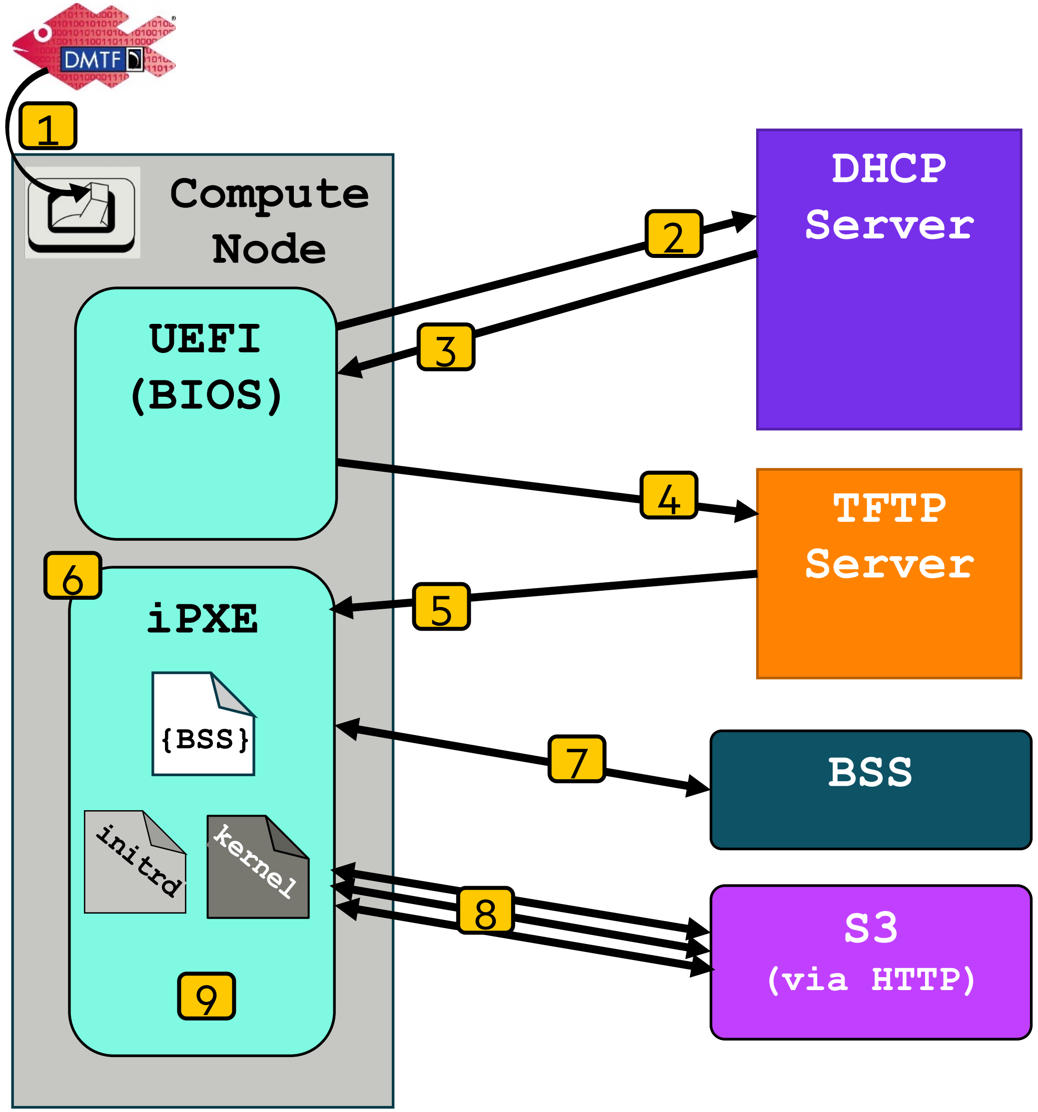
- **Hardware State Manager** (HSM) – Tracks the state of each node and holds their group and role associations
- **Image Management Service** (IMS) – Manages image records (kernel, `initrd`, image root)
- **Simple Storage Service** (S3) – Stores boot artifacts (kernel, `initrd`, image root)
- **Boot Script Service** (BSS) – Stores per-node information about iPXE boot script
- **Cray Advanced Platform and Monitoring Control** (CAPMC) – provides system-level power control for nodes in the system
- **Configuration Framework Service** (CFS) – Configures node(s) using configuration framework

During boot, BOS/BOA will get the S3 reference to boot image. BOA will need to access the image to read boot parameters. At the point that BSS generates the iPXE bootscript, BSS will generate the pre-signed S3 Download URL for the kernel and `initrd`. CPS will similarly need to be updated to project the `rootfs`.



COMPUTE NODE BOOT SEQUENCE

1. The compute node is powered on
2. The BIOS issues a DHCP discover request
3. DHCP Server responds with:
   - The IP address of the TFTP server
   - The name of the file to download
4. The node sends a request to the TFTP server
5. The TFTP server sends `ipxe.efi` to the node
6. The node chainloads the iPXE binary
7. iPXE downloads an `ipxe` boot script from BSS
8. Following the boot script, iPXE downloads the kernel, `initrd`, and kernel parameters from S3
9. The node attempts to boot using the boot artifacts pulled from S3



BOOT SCRIPT SERVICE (BSS)

Boot Script Service (BSS)

  - REST API to interact with HSM and provide nodes with boot artifacts and cloud-init payloads
  - Stores the configuration information that is used to boot each hardware component
  - Nodes consult BSS for their boot artifacts and boot parameters when nodes boot or reboot
- The BSS stores the current image and parameters that are assigned to each node
- The boot parameters stored in BSS for a node when a node is powered on will be used for that boot
- The Boot Orchestration Service (BOS) is used to update the boot script for a given node
  - Updating the boot script for a node in the BSS directly is not recommended
  - BSS does not have any information about how a node should be configured after it boots
  - Post-boot configuration (node personalization) is controlled by the Configuration Framework Service (CFS)
    - BOS calls CFS as part of the process of orchestrating the boot process

RETRIEVING A BOOT SCRIPT FROM BSS

- The boot script for a node includes the following boot artifacts (highlighted):

```
ncn# cray bss bootscript list --name x3000c0s23b2n0
#!ipxe
kernel --name kernel http://rgw-vip.nmn/boot-images/1c4f7f49-bfaf-4c25-9110-f5b46440c9a2/kernel?
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=L18PWYUE7B8KBQR3X4NB%2F20220105%2Fdefault%2Fs3%2Faws4_request&X-Amz-
Date=20220105T012211Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-
Signature=8aa3bdb208d5e216a0331c41c66f4346f6bf75b75b0f5f0addf0caf4bde3fd7e
initrd=initrd console=ttyS0,115200 bad_page=panic crashkernel=360M hugepagelist=2m-2g intel_iommu=off
intel_pstate=disable iommu=pt numa_interleave_omit=headless oops=panic pageblock_order=14 pcie_ports=native
rd.neednet=1 rd.retry=10 rd.shell turbo_boost_limit=999 biosdevname=0 ip=dhcp quiet
spire_join_token=8900a2f6-3bee-4757-bccb-75247893a6d0
root=craycps-s3:s3://boot-images/1c4f7f49-bfaf-4c25-9110-f5b46440c9a2/rootfs:
c91e4b1462822da009f191c206d8c9fa-205:dvs:api-gw-service-nmn.local:300:nmn0 nmd_data=url=s3://boot-images/1c4f7f49-bfaf-
4c25-9110-f5b46440c9a2/rootfs,etag=c91e4b1462822da009f191c206d8c9fa-205 bos_session_id=f8937b77-2c10-4a05-93bd-06cff8ee076b
xname=x3000c0s23b2n0 nid=6 ds=nocloud-net;s=http://10.92.100.81:8888/ || goto boot_retry
initrd --name initrd http://rgw-vip.nmn/boot-images/1c4f7f49-bfaf-4c25-9110-f5b46440c9a2/initrd?
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=L18PWYUE7B8KBQR3X4NB%2F20220105%2Fdefault%2Fs3%2Faws4_request&X-Amz-
Date=20220105T012211Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-
Signature=0dc66fb06761dd2e8f022446da6a5d31f9320c0bdb0c054cc2e7a10d0af4a972 || goto boot_retry
boot || goto boot_retry
:boot_retry
sleep 30
chain https://api-gw-service-nmn.local/apis/bss/boot/v1/bootscript?mac=b4:2e:99:7f:0d:24&retry=1
```

kernel image

Kernel parameters

root file system squashfs image

initrd image

MAC address of node’s NIC

BSS LOGS

- It is useful to monitor the logs of the cray-bss container within the BSS pods.

```
03/08 19:00:33 ncn# kubectl get pods -n services | grep bss
cray-bss-647fb9775f-jmxs7                                      2/2     Running      0          54d
cray-bss-647fb9775f-k4gl5                                      2/2     Running      0          53d
cray-bss-647fb9775f-qzxf5                                      2/2     Running      0          53d
cray-bss-etcd-4kvjphv69p                                       1/1     Running      0          53d
cray-bss-etcd-7lxvcq4drk                                       1/1     Running      0          54d
cray-bss-etcd-brp85brbnd                                       1/1     Running      0          119d
03/08 19:01:05 ncn# for POD in $(kubectl get pods -n services | grep bss |grep -v etcd | awk '{ print$1}');
do kubectl logs -n services --since 10m $POD -c cray-bss; done
03/08 19:01:23 ncn# ssh x1000c1s1b0n1 reboot
Connection to x1000c1s1b0n1 closed by remote host.
03/08 19:01:35 ncn# sleep 480
03/08 19:11:07 ncn# for POD in $(kubectl get pods -n services | grep bss |grep -v etcd | awk '{ print$1}');
do kubectl logs -n services --since 10m $POD -c cray-bss | grep -v DEBUG; done
2022/03/08 19:10:18 Retrieving state info from http://cray-smd/hsm/v1
2022/03/08 19:10:18 GET /meta-data, xname: x1000c1s1b0n1 ip: 10.100.0.114
2022/03/08 19:10:18 http: superfluous response.WriteHeader call from main.metaDataGetAPI
(cloudInitAPI.go:209)
'
', &spireResp): { 0  9a4f8130-7dee-4180-a4cd-63b22138c03c}
2022/03/08 19:07:34 BSS request succeeded for MAC 00:40:a6:83:63:34 (x1000c1s1b0n1)
```

Like other core boot services, BSS runs insides a Kubernetes pod

`reboot` is NOT the recommended way to reboot a node; BOS should be used

IDENTIFYING THE IMAGE IN USE BY A NODE

```
ncn# cray bss bootparameters  list --name x3000c0s14b0n0 --format json | jq '.[].kernel'
"s3://boot-images/1c329db9-3a32-49b8-be7c-2b09d47a609f/kernel"

ncn# cray bss bootparameters  list --name x3000c0s14b0n0 --format json | jq '.[].params'
"console=ttyS0,115200 bad_page=panic crashkernel=360M hugepagelist=2m-2g intel_iommu=off
intel_pstate=disable iommu=pt ip=nmn0:dhcp numa_interleave_omit=headless numa_zonelist_order=node
oops=panic pageblock_order=14 pcie_ports=native printk.synchronous=y quiet rd.neednet=1 rd.retry=10
rd.shell turbo_boost_limit=999 ifmap=net2:nmn0,lan0:hsn0,lan1:hsn1 spire_join_token=${SPIRE_JOIN_TOKEN}
root=craycps-s3:s3://boot-images/1c329db9-3a32-49b8-be7c-2b09d47a609f/rootfs:
4f862288a668ed8328158a438f276ab3-190:dvs:api-gw-service-nmn.local:300:nmn0 nmd_data=url=s3://boot-
images/1c329db9-3a32-49b8-be7c-2b09d47a609f/rootfs,etag=4f862288a668ed8328158a438f276ab3-190
bos_session_id=43254b57-d787-4797-8b45-ab621ca0b327”

ncn# ssh x3000c0s14b0n0 cat /proc/cmdline
kernel initrd=initrd console=ttyS0,115200 bad_page=panic crashkernel=360M hugepagelist=2m-2g
intel_iommu=off intel_pstate=disable iommu=pt ip=nmn0:dhcp numa_interleave_omit=headless
numa_zonelist_order=node oops=panic pageblock_order=14 pcie_ports=native printk.synchronous=y quiet
rd.neednet=1 rd.retry=10 rd.shell turbo_boost_limit=999 ifmap=net2:nmn0,lan0:hsn0,lan1:hsn1
spire_join_token=d399ee35-c191-46c7-9f40-da63f895d368 root=craycps-s3:s3://boot-images/1c329db9-3a32-49b8-
be7c-2b09d47a609f/rootfs:4f862288a668ed8328158a438f276ab3-190:dvs:api-gw-service-nmn.local:300:nmn0
nmd_data=url=s3://boot-images/1c329db9-3a32-49b8-be7c-2b09d47a609f/rootfs,
etag=4f862288a668ed8328158a438f276ab3-190 bos_session_id=43254b57-d787-4797-8b45-ab621ca0b327
xname=x3000c0s14b0n0 nid=49168832 ds=nocloud-net;s=http://10.92.100.81:8888/
```

TRACKING AN IMAGE FROM NODE TO CPS TO S3

```
ncn# cray cps contents list --format json | grep 1c329db9-3a32-49b8-be7c-2b09d47a609f/rootfs
    "s3path": "s3://boot-images/1c329db9-3a32-49b8-be7c-2b09d47a609f/rootfs",

ncn# cray cps contents list --format json | jq 'map(select(.s3path == "s3://boot-images/1c329db9-3a32-49b8-
be7c-2b09d47a609f/rootfs")) | .[].artifactID'
"e2e335eda4055fd1b293de4f2c9ab6ce”

ncn# cray cps contents list --format json | jq 'map(select(.s3path == "s3://boot-images/1c329db9-3a32-49b8-
be7c-2b09d47a609f/rootfs")) | .[].exportPath'
"/var/lib/cps-local/e2e335eda4055fd1b293de4f2c9ab6ce"


ncn# ssh ncn-w001
Last login: Thu Jul 15 04:53:58 2021 from 10.252.1.9

ncn-w001# file /var/lib/cps-local/e2e335eda4055fd1b293de4f2c9ab6ce/rootfs
/var/lib/cps-local/e2e335eda4055fd1b293de4f2c9ab6ce/rootfs: Squashfs filesystem, little endian, version
4.0, 1589565630 bytes, 90812 inodes, blocksize: 131072 bytes, created: Tue Jun 29 17:23:47 2021

ncn-w001# df /var/lib/cps-local/e2e335eda4055fd1b293de4f2c9ab6ce/rootfs
Filesystem             1K-blocks  Used         Available Use% Mounted on
s3fs           18014398509465600     0 18014398509465600   0% /var/lib/cps-local/boot-images
```

When a node requests a new image from CPS the content manager (CM) will cache the squashfs file from S3 bucket to s3fs available to each cray_cps_cm_pm_ pod. The squashfs files are stored on s3fs until CPS deletes the content.

TEMPLATE OF BOS SESSION TEMPLATE

- Use the provided empty session template template as a JSON framework and edit all the fields

```
ncn# cray bos sessiontemplatetemplate list --format json
{
  "boot_sets": {
    ”boot_set1": {
      "boot_ordinal": 1,
      "etag": "your_boot_image_etag",
      "kernel_parameters": "your-kernel-parameters",
      "network": "nmn",
      "node_list": ["x3000c0s19b1n0", "x3000c0s19b1n1", "x3000c0s19b2n0"]
      "path": "your-boot-path",
      "rootfs_provider": "your-rootfs-provider",
      "rootfs_provider_passthrough": "your-rootfs-provider-passthrough",
      "type": "your-boot-type"
    },
    "boot_set2": { ... }
  },
  "cfs": {
    "configuration": "desired-cfs-config"
  },
  "enable_cfs": true,
  "name": "name-your-template"
}
```

Multiple boot sets can be defined that will have same CFS configuration to be applied, but different kernel parameters or different path to boot artifacts

Can specify nodes one of these ways:

"node_list": ["x3000c0s19b1n0", "x3000c0s19b1n1", "x3000c0s19b2n0"]

"node_groups": ["green", "white", "pink"]

"node_roles_groups": ["Compute"]

BOS SESSION TEMPLATE DETAIL

```
ncn# cray bos sessiontemplate describe cos-2.3.101 --format json
{
  "boot_sets": {
    "compute": {
      "boot_ordinal": 2,
      "etag": "b29bb9e8cd8c64541f4ff025e108f7a6",
      "kernel_parameters": "ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN}",
      "network": "nmn",
      "node_roles_groups": [
        "Compute"
      ],
      "path": "s3://boot-images/c26034f1-4acf-4a45-b898-c5842d711ef6/manifest.json",
      "rootfs_provider": "cpss3",
      "rootfs_provider_passthrough": "dvs:api-gw-service-nmn.local:300:hsn0,nmn0:0",
      "type": "s3"
    }
  },
  "cfs": {
    "configuration": "cos-config-2.3.101"
  },
  "enable_cfs": true,
  "name": "cos-2.3.101"
}
```

boot_sets: A collection of nodes & the images they should boot with. One or more boot_sets may be specified per session template

etag: 'entity tag helps identify the version of the manifest.json file. Currently not used but cannot be left blank

network: The network over which the node will boot

kernel parameters : Kernel parameters passed to the operating system

Path: s3 location of the components of the boot image file ([IMS_Image_ID] `manifest.json`). Processed based on the “type”

rootfs_provider: The root file system provider

rootfs_provider_passthrough: Additional kernel parameters that will be appended to the 'rootfs=' kernel parameter

cfs or **cfs_url:** The repository configuration file or clone URL for the repository providing the configuration

enable_cfs: Whether to enable the Configuration Framework Service (CFS)

name: Name of the Session Template. The length of the name is restricted to 45 characters

CREATE A BOS SESSION TEMPLATE

```
ncn# cat INPUT_FILE.json
{
  "name": "cos-2.3.101",
  "boot_sets": {
    "test_compute": {
      "network": "nmn",
      "boot_ordinal": 1,
      "kernel_parameters": "ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN}",
      "rootfs_provider": "cpss3”,
      "node_list": [ "x3000c0s19b1n0" ],
      "etag": "90b2466ae8081c9a604fd6121f4c08b7",
      "path": "s3://boot-images/06901f40-f2a6-4a64-bc26-772a5cc9d321/manifest.json",
      "rootfs_provider_passthrough": "dvs:api-gw-service-nmn.local:300:hsn0,nmn0:0 ",
      "type": "s3" }
    },
  "cfs": {
    "configuration": "cos-config-2.3.101"
  },
  "enable_cfs": true
}
ncn# cray bos sessiontemplate create --file INPUT_FILE.json --name cos-2.3.101
ncn# cray bos sessiontemplate list --format json | jq '.[].name'
"cos-2.3.101"
"uan-2.4.3"
```

Display a list of all session templates in your system, filtering the output with jq for the `.name`

CREATE BOS SESSION

- A BOS Session represents an operation on a Session Template
  - boot – Boot nodes that are off
  - configure – Reconfigure the nodes using the Configuration Framework Service (CFS)
  - reboot – Gracefully power down nodes that are on and then power them back up
  - shutdown – Gracefully power down nodes that are on
- Use `cray bos session create` to create a BOS session

```
ncn# cray bos session create --template-uuid cos-2.3.101 --operation reboot
operation = "Reboot"
templateUuid = "cos-2.3.101"
[[links]]
href = "/v1/session/158fc371-d279-4494-a60e-fcac5612d605"
jobId = "boa-158fc371-d279-4494-a60e-fcac5612d605"
rel = "session"
type = "GET"

[[links]]
href = "/v1/session/158fc371-d279-4494-a60e-fcac5612d605/status"
rel = "status"
type = "GET“
```

When a BOS session is created it initiates one or more Boot Orchestration Agent (BOA) jobs. The name of the `session` created will be labeled `href` and included in the BOA `jobid` – which is part of the BOA pod name

- BOS supports an optional --limit parameter when creating a session
  - List of nodes, HSM groups, or HSM roles to limit the nodes that BOS runs against
  - Components are treated as OR operations unless preceded by “&” for AND or “|” for NOT

```
cray bos session create --template-uuid cos-2.3.101 --operation reboot --limit x3000c0s20b2n0
```

  - Operate on all except some nodes, HSM groups, or HSM roles

```
cray bos session create --template-uuid cos-2.3.101 --operation configure --limit all,!x3000c0s20b2n0
```

VIEW RUNNING BOS SESSION INFORMATION

- Use `cray bos session describe` to view progress of the BOS job
- Use `kubectl get pods` to view the status of the Boot Orchestration Agent (BOA) job associated with the BOS job

```
ncn# cray bos session describe 158fc371-d279-4494-a60e-fcac5612d605

boa_job_name = "boa-158fc371-d279-4494-a60e-fcac5612d605"
complete = false
error_count = 0
in_progress = true
operation = "Reboot"
start_time = "2022-06-28 08:40:14.949422"
status_link = "/v1/session/158fc371-d279-4494-a60e-fcac5612d605/status"
templateUuid = ”cos-2.3.101“
```

When a BOS session is created it initiates one or more Boot Orchestration Agent (BOA) jobs. The name of the session created will be labeled `href` and included in the BOA jobid – which is part of the BOA pod name

`cray bos session describe <JOB ID>` is used to view the status and progress of the job.
`boa_job_name` – Boot Orchestration Agent job name.

Monitoring the BOA JOB with **`kubectl get pods`** command.

```
ncn# kubectl get pods -n services -l job-name=boa-158fc371-d279-4494-a60e-fcac5612d605

NAME                                           READY   STATUS    RESTARTS   AGE
boa-158fc371-d279-4494-a60e-fcac5612d605-xw4xh 2/2     Running   0          2m47s
```

VIEW BOS SESSION STATUS

```
ncn# cray bos session status describe CATEGORY_NAME PHASE_NAME BOOT_SET_NAME SESSION_ID --format json
```

- BOS session status Phases
  - shutdown
  - boot
  - configure
- BOS session status Categories
  - not_started
  - succeeded
  - failed
  - excluded
  - in_progress

```
ncn# cray bos session status describe succeeded shutdown compute fb808925-2dd6-440d-8d6c-834892472036
name = "succeeded"
node_list = [ "x3000c0s19b4n0", "x3000c0s19b2n0", "x3000c0s19b3n0", "x3000c0s19b1n0",]
ncn# cray bos session status describe failed boot compute fb808925-2dd6-440d-8d6c-834892472036
name = ”failed"
node_list = [ "x3000c0s19b4n0",]
ncn# cray bos session status describe in_progress configure compute fb808925-2dd6-440d-8d6c-834892472036
name = "in_progress"
node_list = ["x3000c0s19b2n0", "x3000c0s19b3n0", "x3000c0s19b1n0",]
```

VIEW COMPLETED BOS SESSION INFORMATION

- Use `cray bos session describe` to view progress of the BOS job.
- Use `kubectl get pods` to view the status of the Boot Orchestration Agent (BOA) job associated with the BOS job.

```
ncn# cray bos session describe 158fc371-d279-4494-a60e-fcac5612d605

boa_job_name = "boa-158fc371-d279-4494-a60e-fcac5612d605"
complete = true
error_count = 0
in_progress = false
operation = "Reboot"
start_time = "2021-06-28 08:40:14.949422"
status_link = "/v1/session/158fc371-d279-4494-a60e-fcac5612d605/status"
stop_time = "2021-06-28 08:53:50.711327"
templateUuid = ”cos-2.3.101“
```

Monitoring the BOS job with **`cray bos session describe <JOB ID>`** to completion

Monitoring the BOA job with **`kubectl get pods`** command to completion

```
ncn# kubectl get pods -n services -l job-name=boa-158fc371-d279-4494-a60e-fcac5612d605
NAME                                           READY   STATUS      RESTARTS   AGE
boa-158fc371-d279-4494-a60e-fcac5612d605-xw4xh  0/2     Completed   0          14m
```

LOGS

- Console logs and access
- SMA-Kibana

CONTAINERIZED CONSOLE ACCESS

- ConMan is a serial console management program designed to support a large number of console devices and simultaneous users
- cray-console uses ConMan for interactive remote console access and console log collection
  - Automatically detects nodes which have been added or removed
  - Shared filesystem in Ceph for all cray-console pods to easily view log data
  - Console log data sent to SMA for other log processing
  - Dynamic autoscaling number of cray-console-node pods for size of system
    - Minimally, two pods are started
    - The number of PODs is scaled on
      - 750 Liquid-cooled nodes and/or 2000 “River” nodes
        - The Liquid-cooled nodes each require an ssh connection, so numbers are different
- Log locations:
  - Logs visible in any `cray-console-node-x` pod
  - Node logs: `/var/log/conman/console.XNAME`
  - ConMan damon logs: `/var/log/conman.log`

```
ncn# kubectl get pods -A |grep cray-console
services            cray-console-data-5cd59677d9-lf4f4
services            cray-console-data-postgres-0
services            cray-console-data-postgres-1
services            cray-console-data-postgres-2
services            cray-console-node-0
services            cray-console-node-1
services            cray-console-operator-7f9894f657-5psn5
```

CONSOLE LOGS WITH CRAY-CONSOLE-NODE

```
ncn# kubectl get pods -A |grep console-node
services            cray-console-node-0        3/3     Running       1      62d
services            cray-console-node-1        3/3     Running       0      68d
ncn# kubectl -it exec -n services cray-console-node-1 -c cray-console-node -- ls /var/log/conman
console.x1000c0s1b0n0  console.x1000c3s3b0n0   console.x3000c0s20b4n0
console.x1000c0s1b0n1  console.x1000c3s3b0n1   console.x3000c0s23b1n0
console.x1000c0s1b1n0  console.x1000c3s3b1n0   console.x3000c0s23b2n0
console.x1000c0s1b1n1  console.x1000c3s3b1n1   console.x3000c0s23b3n0
console.x1000c0s5b0n0  console.x1000c5s5b0n0   console.x3000c0s23b4n0
console.x1000c0s5b0n1  console.x1000c5s5b0n1   console.x3000c0s25b1n0
console.x1000c0s5b1n0  console.x1000c5s5b1n0   console.x3000c0s25b2n0
console.x1000c0s5b1n1  console.x1000c5s5b1n1   console.x3000c0s25b3n0
console.x1000c0s7b0n0  console.x1000c7s7b0n0   console.x3000c0s25b4n0
```

Each pod sees all the console files, only one cray-console-node pod is managing that node and writing its log file

```
ncn# kubectl -it exec -n services cray-console-node-1 -c cray-console-node -- \
tail -f /var/log/conman/console.x1000c0s1b0n0
```

Can view log without entering pod

```
ncn# kubectl -it exec -n services cray-console-node-1 -c cray-console-node -- /bin/bash
cray-console-node-1-pod# grep –i error /var/log/conman/console.x1000c0s1b0n0
```

Can view log by entering pod

- Access Console Log Data Via the SMA-kibana user interface
  https://github.com/Cray-HPE/docs-csm/blob/release/1.3/operations/conman/Access_Console_Log_Data_Via_the_System_Monitoring_Framework_SMF.md

INTERACTIVE CONSOLE EXAMPLE (LONG)

- To join the console, use `conman -j`
  - Retrieve the `cray-console-operator` pod ID

```
ncn# CONPOD=$(kubectl get pods -n services \
    -o wide|grep cray-console-operator|awk '{print $1}')
ncn# echo $CONPOD
cray-console-operator-79bf95964-qpcpp
```

  - Set the `XNAME` variable to the xname of the node whose console you wish to open

```
ncn# XNAME=x1000c0s0b0n0
```

  - Find the `cray-console-node` pod that is managing that node

```
ncn# NODEPOD=$(kubectl -n services exec $CONPOD -c cray-console-operator \
-- sh -c "/app/get-node $XNAME" | jq .podname | sed 's/"//g')
ncn# echo $NODEPOD
cray-console-node-1
```

  - Connect to the node's console using ConMan on the `cray-console-node` pod you found

```
ncn# kubectl exec -it -n services $NODEPOD -- conman -j $XNAME
<ConMan> Connection to console [x1000c0s0b0] opened.
nid000001 login:
```

  - To exit console use `&.` command

INTERACTIVE CONSOLE EXAMPLE (SHORT)

- Alternate form of previous slide

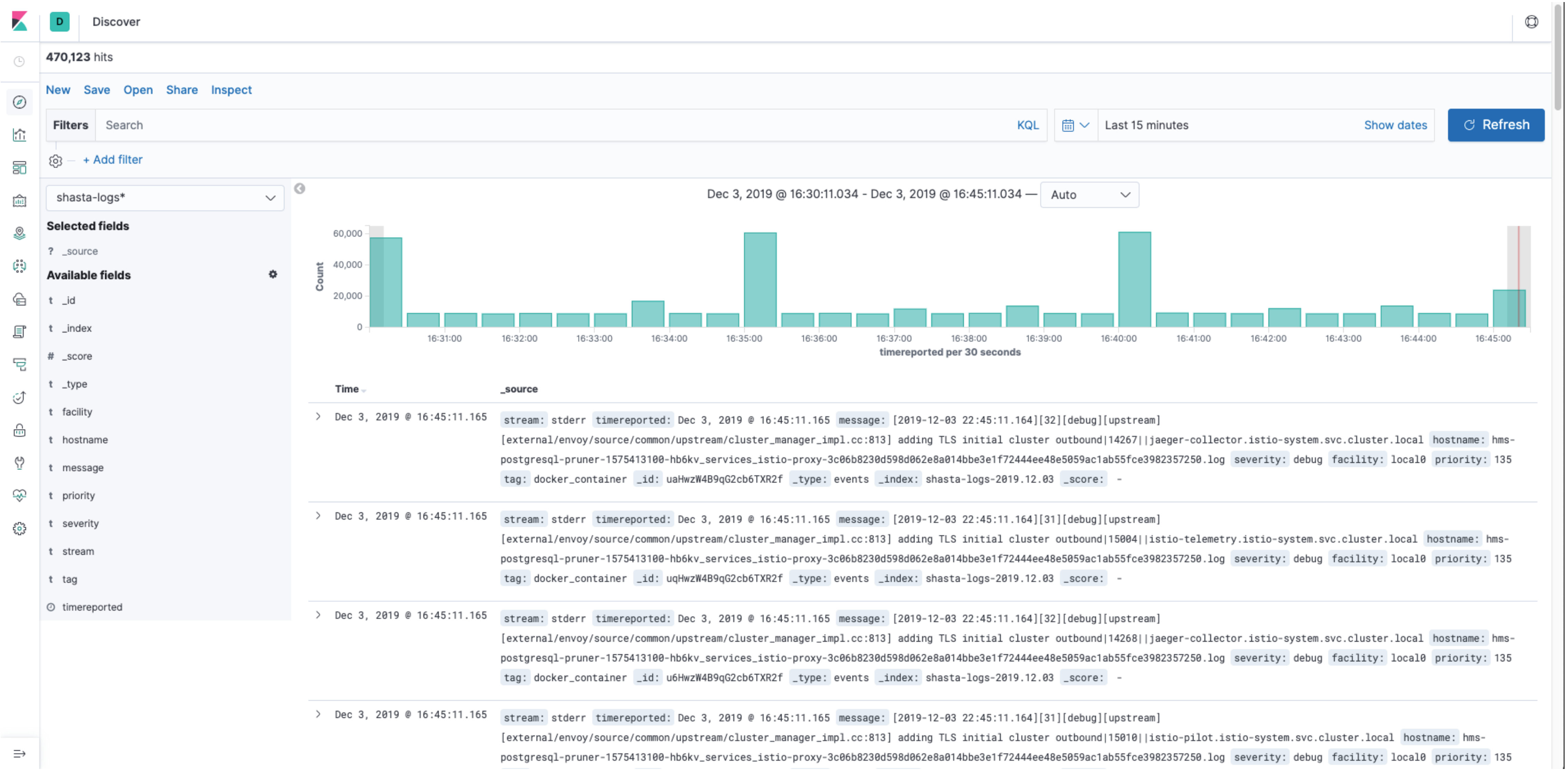
```
ncn# ConsoleJ ()
{
    XNAME=$@;
    CONPOD=$(kubectl get pods -n services -o wide|grep cray-console-operator|awk '{print $1}');
    NODEPOD=$(kubectl -n services -c cray-console-operator exec $CONPOD -- sh -c "/app/get-node $XNAME" | jq .podname | tr -d '"');
    echo conpod = $CONPOD nodepod = $NODEPOD;
    kubectl exec -it -n services $NODEPOD -c cray-console-node -- conman -j $XNAME
}
ncn# ConsoleJ x1000c0s0b0n0
<ConMan> Connection to console [x1000c0s0b0n0] opened.
nid000001 login:
```

  - To exit console use `&.` command
- To view the console read-only instead of joining it read-write, use `conman -m $XNAME`

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



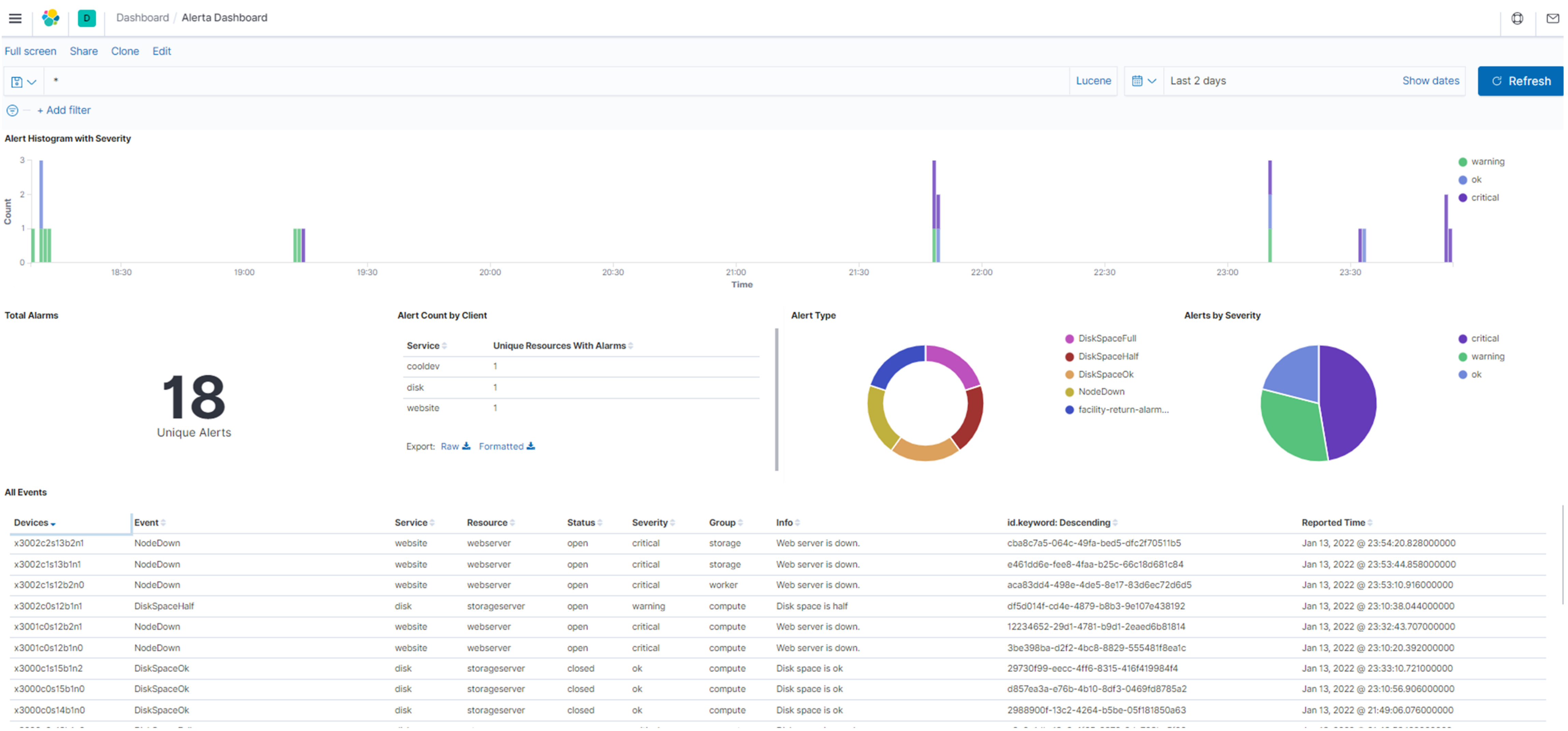
```
Dec 3, 2019 @ 16:45:11.165  stream: stderr timereported: Dec 3, 2019 @ 16:45:11.165 message: [2019-12-03 22:45:11.164][32][debug][upstream]
                            [external/envoy/source/common/upstream/cluster_manager_impl.cc:813] adding TLS initial cluster outbound|14267||jaeger-collector.istio-system.svc.cluster.local hostname: hms-
                            postgresql-pruner-1575413100-hb6kv_services_istio-proxy-3c06b8230d598d062e8a014bbe3e1f72444ee48e5059ac1ab55fce3982357250.log severity: debug facility: local0 priority: 135
                            tag: docker_container _id: uaHwzW4B9qG2cb6TXR2f _type: events _index: shasta-logs-2019.12.03 _score: -

Dec 3, 2019 @ 16:45:11.165  stream: stderr timereported: Dec 3, 2019 @ 16:45:11.165 message: [2019-12-03 22:45:11.164][31][debug][upstream]
                            [external/envoy/source/common/upstream/cluster_manager_impl.cc:813] adding TLS initial cluster outbound|15004||istio-telemetry.istio-system.svc.cluster.local hostname: hms-
                            postgresql-pruner-1575413100-hb6kv_services_istio-proxy-3c06b8230d598d062e8a014bbe3e1f72444ee48e5059ac1ab55fce3982357250.log severity: debug facility: local0 priority: 135
                            tag: docker_container _id: uqHwzW4B9qG2cb6TXR2f _type: events _index: shasta-logs-2019.12.03 _score: -

Dec 3, 2019 @ 16:45:11.165  stream: stderr timereported: Dec 3, 2019 @ 16:45:11.165 message: [2019-12-03 22:45:11.164][32][debug][upstream]
                            [external/envoy/source/common/upstream/cluster_manager_impl.cc:813] adding TLS initial cluster outbound|14268||jaeger-collector.istio-system.svc.cluster.local hostname: hms-
                            postgresql-pruner-1575413100-hb6kv_services_istio-proxy-3c06b8230d598d062e8a014bbe3e1f72444ee48e5059ac1ab55fce3982357250.log severity: debug facility: local0 priority: 135
                            tag: docker_container _id: u6HwzW4B9qG2cb6TXR2f _type: events _index: shasta-logs-2019.12.03 _score: -

Dec 3, 2019 @ 16:45:11.165  stream: stderr timereported: Dec 3, 2019 @ 16:45:11.165 message: [2019-12-03 22:45:11.164][31][debug][upstream]
                            [external/envoy/source/common/upstream/cluster_manager_impl.cc:813] adding TLS initial cluster outbound|15010||istio-pilot.istio-system.svc.cluster.local hostname: hms-
                            postgresql-pruner-1575413100-hb6kv_services_istio-proxy-3c06b8230d598d062e8a014bbe3e1f72444ee48e5059ac1ab55fce3982357250.log severity: debug facility: local0 priority: 135
```

SMA-KIBANA ALERTA DASHBOARD



Total Alarms

18
Unique Alerts

Alert Count by Client

Service	Unique Resources With Alarms
cooldev	1
disk	1
website	1

Export: Raw Formatted

All Events

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

TROUBLESHOOTING TIPS

TROUBLESHOOTING FAILED BOOT

- Tried to boot all compute nodes, but some failed to boot
- Where do you start looking?
  - BOA log shows widest view for BOSv1
    - Does the end of the BOA log have a traceback from some execution error?
    - Were there problems with BOA talking to BSS to assign the boot artifacts or to CFS to set desired configuration?
    - Were the power management calls to CAPMC smoothly done?
    - Were the node state status calls to HSM showing all nodes moving from OFF to ON to READY
      - With BOSv1, BOA has a 30-minute timeout to make these calls before giving up on the boot
    - Were the node configuration status calls to CFS showing all nodes (that had been in READY) moving to configured?
    - Did any of the calls from BOA to other services have 503 error messages?
      - This may mean that the are problems with the API gateway or that specific service might have an issue
        - Check the Kubernetes pod logs for the API gateway and the services which had 503 errors
        - Check whether any of the pods have an increasing restart count (or are in CrashLoopBackOff)
        - Check whether there have been OOMkill or CPUthrottling alerts during the boot
          - Some pods may need larger requests for memory or CPU resources if the system has recently been expanded with more nodes
    - Explore SMA-grafana dashboards for the time interval of the boot
    - Explore SMA-kibana dashboards for the time interval of the boot
  - If BOA reports power management errors, there might be more detail in the CAPMC logs
    - If there has been an Emergency Power Off (EPO) event, special handling may be needed to recover from it before trying to boot

TROUBLESHOOTING NODES NOT READY

- Check node state with HSM or SAT commands for which nodes are not in the READY state to find missing nodes

  ```
  ncn# sat status --filter role=compute --filter state!=ready
  ```

- For any nodes in the OFF state, check the power logs on their BMC (nodecontroller) for power up faults
  - Olympus nodes: (example x1000c0s0b0n1)

    ```
    ncn# ssh x1000c0s0b0
    x1000c0s0b0> egrep "\(partially powered up\)|Stopped at PS|already fully powered up" /var/log/powerfault_up.Node1
    ```

  - Refer to hardware service team if these message patterns are found
- For any nodes in the ON state, check the console logs for each node
  - After inspecting a few console logs to find a pattern, grep for that pattern in all the console logs conman

    ```
    ncn# kubectl -it exec -n services cray-console-node-1 -c cray-console-node -- grep pattern /var/log/conman/console.*
    ```

  - Repeat that grep command but count how many nodes have a problem matching that pattern

    ```
    ncn# kubectl -it exec -n services cray-console-node-1 -c cray-console-node -- grep pattern /var/log/conman/console.* | wc -l
    ```

  - Do the console messages indicate whether the node failed
    - To get a DHCP response and start downloading ipxe.efi binary? – check cray-kea and cray-ipxe pods
    - To contact BSS for the iPXE boot script once ipxe.efi started running? – check BSS pods and whether the MACaddr used has valid data in BSS
    - To download the boot artifacts (kernel, initrd) from S3? Check for presence of them with “`cray artifacts`” command
    - To configure the HSN NICs as tmp0, tmp1, etc while in Dracut? Check Slingshot fabric manager configuration and edge port health
    - To generate DVS node map and enumerate the DVS server addresses? Check DVS server health (on nodes running CPS cm-pm pods)
    - To mount the rootfs squashfs image from DVS servers? Check whether nodes with CPS cm-pm pods have the image cached in `/var/lib/cps-local`
  - Did the node have any of these errors in the console log? Some are kernel panics, some show node dropping into UEFI shell, and some indicate hardware errors

    ```
    startup.nsh|ernel panic|any other key to continue|Enter for maintenance|Entering emergency mode|query
    intf hsn|WHEA: Detected Memory Error|ASSERT|Shell\>|Unable to get TLV for interface|Machine Check"
    ```

TROUBLESHOOTING CFS

- If BOA reports some CFS failures
  - check the CFS batches that ran for this boot

```
ncn# kubectl -n services --sort-by=.metadata.creationTimestamp get pods | grep cfs
```

  - Confirm that CFS batcher started enough batches for the number of compute nodes divided by the CFS batch size
  - Check Ansible logs from the CFS pods looking for the `PLAY RECAP` summary to see how many tasks failed and on which nodes
    - Look in these logs for the tasks that failed with their failure messages and cross-reference messages with the Ansible code for this task
    - Confirm that the correct CFS configuration was assigned to the node (test versus production)
  - Check how many nodes completed configuration

```
ncn# sat status --filter role=compute | grep -i configured
```

  - Check how many nodes failed configuration

```
ncn# sat status --filter role=compute | grep -i failed
```

  - Check how many nodes are still trying to configure

```
ncn# sat status --filter role=compute | grep -i pending
```

- If the failures in the BOA log and other logs were transitory, then run the `cray bos` command from the end of the BOA log with the `--limit` option to retry the boot of the problematic nodes with the same boot artifacts

TROUBLESHOOTING SLOW BOOT

- Booted, but some nodes appeared to be slow to boot
- Where do you start looking?
  - Console logs can help to identify which nodes are the slowest in any part of the booting process
  - BOA log shows widest view
    - BOA gets status from HSM and reports how many nodes are in OFF, ON, READY state and when they change states, but not to the node name granularity
      - Are nodes of the same type all changing state mostly together?
        - The slow nodes may be on the path to hardware failure
        - A system with a mixture of compute node types will often show variation in boot time by node type
    - BOS gets configuration status from CFS and reports how many nodes are not yet configured
      - Are nodes of the same type all moving into configured state together?
      - There may be tuning problems to address with Ansible plays run by CFS
      - There may be tuning need for the CFS infrastructure for the size of the system

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

COLLECTING DATA FOR HPE SERVICE

- Node Memory Dump (NMD)
- Slingshot (hsn_triage_capture)
- System Diagnostic Utility (SDU)

NODE MEMORY DUMP (NMD)

- Standard Linux kdump mechanism
  - Uses `kexec` for booting into the dump-capture kernel (`kdump boot`) immediately after kernel crash
  - Standard `kdump` not scalable to large systems
    - Standard, each node decides on its own to produce a node memory dump
  - Needs a service to initiate dumps of selected nodes
- NMD controls the `kdump` process of the panicked node
  - Provides concurrent dump capability
  - Controls automated kdump so that the dump is generated only for the requested nodes
  - Provides a configurable makedumpfile dump level option for the selected node at dump time
    - Can specify dumplevel argument of the makedumpfile command
      - 31 by default
      - 16 if it is required to retrieve user process core dump (user data pages) or non-private cache pages

```
ncn# cray nmd dumps --help
Usage: cray nmd dumps [OPTIONS] COMMAND [ARGS]...
Options:
  --help  Show this message and exit.
Commands:
  create
  delete
  describe
  list
```

HSN_TRIAGE_CAPTURE

- If a problem occurs while following the HSN Debug Procedure, capture system log files by running `hsn_triage_capture` to capture state and logs from the fabric manager, switches and CMMs
  - When it completes, it prints the path to the tarball of captured data

```
ncn# kubectl exec -it -n services $(kubectl get pods -A |grep fabric |awk '{print
$2}') -c slingshot-fabric-manager -- /bin/bash
slingshot-fabric-manager# hsn_triage_capture -h
usage: ./hsn_triage_capture [-h] [-a] [-c] [-f] [-s] [-t
TARGET_XNAME[,TARGET_XNAME]]
Collect HSN debug information from the FMN, CMMs and switches,
then aggregate into a single tarball.
Note: it is assumed that passwordless ssh is configured to all devices.
optional arguments:
-h Show usage and exit
-a Collect FMN, console, and switch data (default)
-c Collect switch console logs
-f Collect FMN data
-s Collect switch data
-t TARGET_XNAME[,TARGET_XNAME] Specific targets only
```

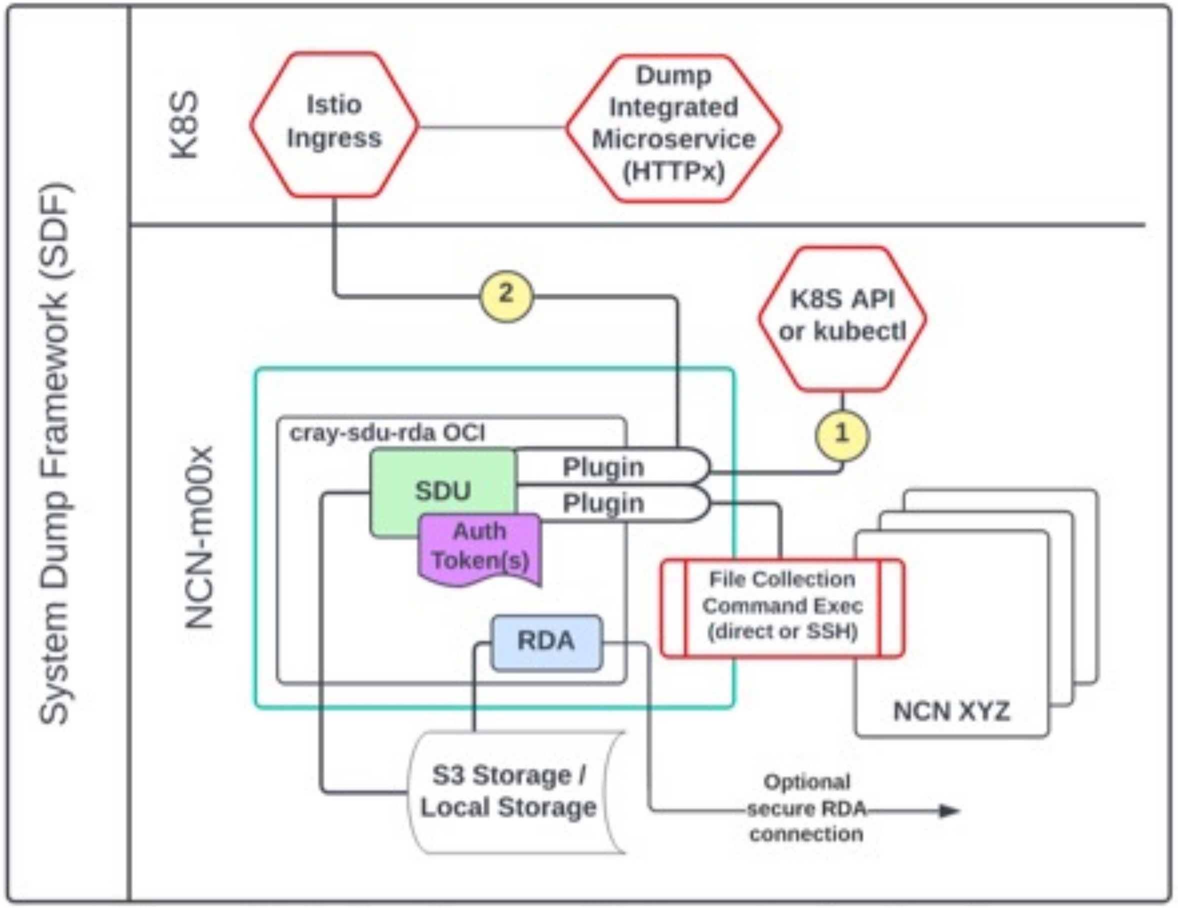
SYSTEM DIAGNOSTIC UTILITY (SDU)

- Pluggable architecture to collect logs, core files, register dumps, and more
  - Can package the output to `tar` to share any useful system triage information
  - Collects data from distributed parts of the system
- Remote Device Access (RDA) is capable of securely transporting this data to HPE
  - RDA Documentation: https://midway.ext.hpe.com/home
  - Security white paper: https://support.hpe.com/hpesc/public/docDisplay?docId=a00006791en_us
  - AFT (Asynchronous File Transport) is used to securely transport SDU data to HPE
  - IDA (Interactive Device Access) is used to tunnel TCP sessions with HPE
  - Independent from one another and both are opt-in features
- SDU runs in a podman container
  - Container is controlled as a service via systemd on master node
    - `/etc/sysconfig/cray-sdu-rda` – container settings
    - `ncn-m#` **`systemctl start cray-sdu-rda.service`**
      - `/usr/sbin/cray-sdu-rda` – used by systemd to configure, start, and stop container
    - `/usr/sbin/sdu` – passes commands into the cray-sdu-rda podman container
      - allows sdu commands to be run whether on NCN or in the container
- `sdu` commands can be run from the master node or within the container

```
ncn-m001: # sdu bash
ncn-m001-sdu: #   <--- prompt indicates you are inside the SDU/RDA container
```

SYSTEM DUMP FRAMEWORK (SDF)

- Provides a standard system dump feature
  - Onsite triage
  - Onsite to central support
  - Provides a structured data format
- Resiliency model
  - ncn-m001 and ncn-m002 (but SDU can be started on ANY master node, only 1 at a time)
  - Each eligible master node should have a unique RDA configuration



① Kubernetes Service Annotations are used to store systems dump integration discovery attributes.

② After dump-integrated microservices are discovered via K8S API/kubectl query for specific annotations, the dump protocol can be initiated and dump content downloaded via the SDU, via the Istio ingress (or equiv)

SDU SCENARIOS

- Health
  - Performs a system health collection to gather health information from the system
  - Useful times to run
    - After CSM install.sh completes
    - Before and after NCN reboots
    - After the system is brought back up
    - Any time there is unexpected behavior observed
    - In order to provide relevant information to create support tickets
- Inventory
  - Performs an inventory collection to gather version information for software, firmware, and hardware
  - Useful to run after system upgrades
    - The information collected is used by the HPE Cray Service and R & D organizations to improve customer support
- Triage
  - Performs a triage collection which will gather diagnostic information and logs necessary for HPE Cray Service and R & D to perform problem determination and isolation
  - You are encouraged to provide the --ref 'sfdc:<case number>' command line option to ensure that the snapshot is associated with your service case

SDU TRIAGE SCENARIO

```
ncn-m001# sdu --scenario triage --start_time '-2 days' --reason "Problem with system"
[stdout] INFO Configuration file "/etc/opt/cray/sdu/sdu.conf" and CLI Options Valid.
[stdout] INFO UI master_control status is (enabled) [no control file created]
[stdout] INFO MASTER CONTROLS -> (M:True, U:False)
[stdout] INFO UI CONTROLS -> (C:True, U:True)
[stdout] INFO Exclusive run: Lock file created
@ /var/opt/cray/sdu/lock/sdu.lock_channel-triage_system-devkit
[stdout] INFO COLLECT stage start
[...]
[stdout] INFO dir created in view /var/opt/cray/sdu/collection/triage/view/
2021-02-15T03-10-53_UTC-3c7c6d3040cef5b59b15f15f29c9eda2
[stdout] INFO starting purge
[stdout] INFO work directory removed from '/var/opt/cray/sdu/collection/triage/.work'
[stdout] INFO keeping 10 snapshot(s) max
[stdout] INFO Found 2 snapshot(s) to keep, 0 to purge
[stdout] INFO exiting purge, nothing to do
[stdout] INFO 1813098605.0 raw bytes collected.
[stdout] INFO SDU session stop successfully
[stdout] INFO run took 2431.83 seconds
ncn-m001# cd /var/opt/cray/sdu/collection/triage/view/\
2021-02-15T03-10-53_UTC-3c7c6d3040cef5b59b15f15f29c9eda2
```

All data collected from plugins will be in the view directory

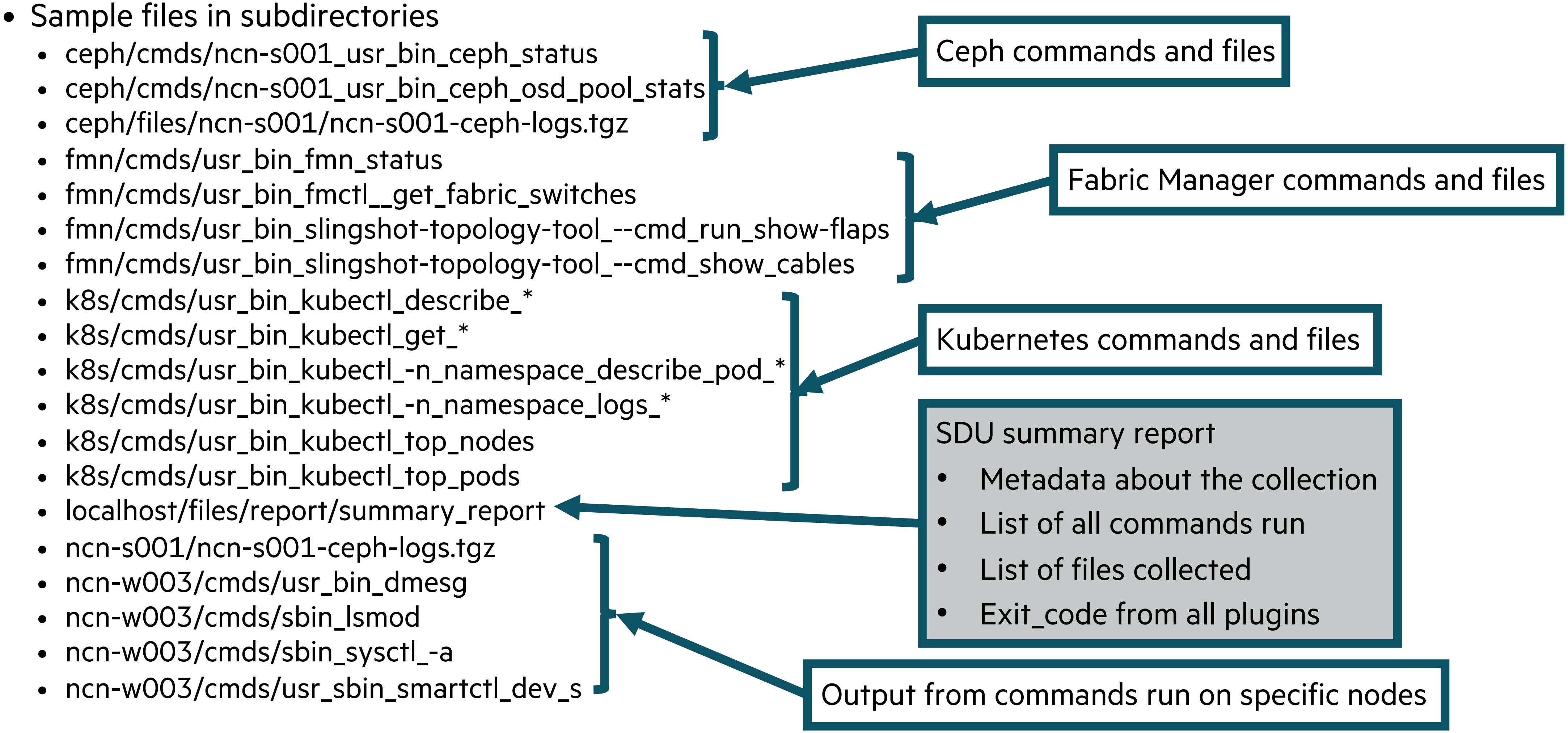
EXPLORE SDU VIEW

- Dump contents are organized first by host or system management component, and then by content type (files and cmds)
  - The following is an example of the directory path:

```
ncn-m001# ls -l
total 3576
drwxr-x--- 4 root root 31 Feb 15 03:51 ceph
drwxr-x--- 3 root root 18 Feb 15 03:51 fmn
drwxr-x--- 3 root root 18 Feb 15 03:51 k8s
drwxr-x--- 3 root root 19 Feb 15 03:51 localhost
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-m002
drwxr-x--- 4 root root 31 Feb 15 03:51 ncn-m003
drwxr-x--- 4 root root 31 Feb 15 03:51 ncn-m003-sdu
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-s001
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-s002
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-s003
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-w001
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-w002
drwxr-x--- 3 root root 19 Feb 15 03:51 ncn-w003
-rw-r--r-- 1 root root 3659206 Feb 15 03:51
session-1613358653-3c7c6d3040cef5b59b15f15f29c9eda2.json
```

- Additional subdirectories exist that contain the logs, core files, register dumps, and more

EXPLORE SDU DATA



- Sample files in subdirectories
  - ceph/cmds/ncn-s001_usr_bin_ceph_status
  - ceph/cmds/ncn-s001_usr_bin_ceph_osd_pool_stats
  - ceph/files/ncn-s001/ncn-s001-ceph-logs.tgz
  - fmn/cmds/usr_bin_fmn_status
  - fmn/cmds/usr_bin_fmctl__get_fabric_switches
  - fmn/cmds/usr_bin_slingshot-topology-tool_--cmd_run_show-flaps
  - fmn/cmds/usr_bin_slingshot-topology-tool_--cmd_show_cables
  - k8s/cmds/usr_bin_kubectl_describe_*
  - k8s/cmds/usr_bin_kubectl_get_*
  - k8s/cmds/usr_bin_kubectl_-n_namespace_describe_pod_*
  - k8s/cmds/usr_bin_kubectl_-n_namespace_logs_*
  - k8s/cmds/usr_bin_kubectl_top_nodes
  - k8s/cmds/usr_bin_kubectl_top_pods
  - localhost/files/report/summary_report
  - ncn-s001/ncn-s001-ceph-logs.tgz
  - ncn-w003/cmds/usr_bin_dmesg
  - ncn-w003/cmds/sbin_lsmod
  - ncn-w003/cmds/sbin_sysctl_-a
  - ncn-w003/cmds/usr_sbin_smartctl_dev_s

Ceph commands and files

Fabric Manager commands and files

Kubernetes commands and files

SDU summary report
- Metadata about the collection
- List of all commands run
- List of files collected
- Exit_code from all plugins

Output from commands run on specific nodes

SDU – KEY DIRECTORIES

- Service (manages SDU container)
  - `/usr/lib/systemd/system/cray-sdu-rda.service`
- Application (inside the container)
  - SDU core: /opt/cray/sdu/default/

    ```
    -ncn-m001-sdu:/ # ls /opt/cray/sdu
       3.3.12-20210624113255_6631f99  default
    ```

  - SDU Plugins: /opt/cray/sdu/default/plugins
- Configuration
  - `/etc/opt/cray/sdu/sdu.conf`
  - scenario_dir: `/etc/opt/cray/sdu/scenario` (defined in sdu.conf, may have changed from default)
- output (defined in sdu.conf, may have changed from default)
  - log_dir: `/var/opt/cray/sdu/log`
  - lock_dir: `/var/opt/cray/sdu/lock`
  - state_dir: `/var/opt/cray/sdu/run`
  - collection_dir: `/var/opt/cray/sdu/collection`

SDU – MOVING THE COLLECTION (TAR / RDA)

- Tar up collection

```
ncn-m001# cd /var/opt/cray/sdu/collection/<scenario>/view
ncn-m001# tar cvfzh test-system-2020-10-01T00-35-20_UTC-c410d30f1d5656ae006f657aa09d4d27.tgz
2020-10-01T00-35-20_UTC-c410d30f1d5656ae006f657aa09d4d27
```

- RDA Configuration (within the SDU container)
  - `/etc/rda/rda.conf` (if proxy settings are needed)

- RDA Outbox
  - `/var/opt/cray/sdu/outbox`

- Staging files to RDA (to send to HPE) (this will be automated in a future release)

```
ncn-m001-sdu# cd /var/opt/cray/sdu/collection/<scenario>/view
ncn-m001-sdu# sdu-stage-to-rda 2021-02-25T20-09-52_UTC-
f6cade95450824711405aa52dade8092
Staging files for RDA transport
Moving files from /var/tmp/RDA_STAGE.7gL3 to RDA outbox /var/tmp/rda/outbox
Done.
```

HPE CRAY EX SYSTEM OVERVIEW
ANSIBLE BEST PRACTICES
MONITORING TOOLS
SYSTEM MANAGEMENT HEALTH
TUNING COMPUTE NODES
SYSTEM ADMIN TOOLKIT
TROUBLESHOOTING BOOT FAILURES
COLLECTING DATA FOR HPE SERVICE
RESOURCES

RESOURCES

- Documentation
- Open-Source Software
- Training
- Related Presentations

DOCUMENTATION - INSTALLATION

- HPE Cray EX System Software Getting Started Guide S-8000
- HPE Cray System Management (CSM) Markdown
  - https://github.com/Cray-HPE/docs-csm/tree/release/1.3
- HPE Cray System Management (CSM) HTML
  - https://cray-hpe.github.io/docs-csm/en-13/
- HPE Cray EX System HPC Firmware Pack Installation Guide S-8037
- HPE Cray EX System Admin Toolkit HTML
  - https://cray-hpe.github.io/docs-sat/en-24/
- HPE Cray EX System Diagnostic Utility Installation Guide S-8034
- HPE Cray EX System Monitoring Application Installation Guide S-8030
- HPE SUSE Linux Enterprise Operating System Installation Guide S-8028
- HPE Slingshot Release Notes
- HPE Slingshot Operations Guide
- HPE Cray Operating System Installation Guide CSM on HPE Cray EX Systems S-8025
- HPE Cray User Access Node Software Installation Guide S-8032
- HPE Cray Programming Environment Installation Guide: CSM on HPE Cray EX S-8003
- HPE Cray EX Analytics Applications Guide S-8027

DOCUMENTATION - ADMINISTRATION

- HPE Cray System Management (CSM) Markdown
  - https://github.com/Cray-HPE/docs-csm/tree/release/1.3
  - https://github.com/Cray-HPE/docs-csm/blob/release/1.3/operations/kubernetes/Kubernetes.md
  - https://github.com/Cray-HPE/docs-csm/blob/release/1.3/glossary.md
- HPE Cray System Management (CSM) HTML
  - https://cray-hpe.github.io/docs-csm/en-13/
- HPE Cray EX System Admin Toolkit Guide S-8031
- HPE Cray EX System Diagnostic Utility Administration Guide S-8035
- HPE Cray EX System Monitoring Application Administration Guide S-8029
- HPE Cray EX Analytics Applications Guide S-8027
- HPE Cray Operating System Administration Guide CSM on HPE Cray EX Systems S-8024
- HPE Cray User Access Node Software Administration Guide S-8033
- HPE Cray System Management Diagnostics Guide S-8038
- HPE Slingshot Operations Guide
- HPE Slingshot Troubleshooting
- HPE Slingshot Hardware Guide
- HPE Cray Programming Environment User Guide: CSM on HPE Cray EX S-8005

DOCUMENTATION – OPEN SOURCE TOOLS

- CSM
  - MIT License
  - Github Hosted
    - https://github.com/Cray-HPE
  - Community Governance
    - https://github.com/Cray-HPE/community
- SAT
  - MIT License
  - Github Hosted
    - Primary repository for the sat CLI written in Python: https://github.com/Cray-HPE/sat
    - Podman wrapper script written in Bash: https://github.com/Cray-HPE/sat-podman
    - An important library used by sat CLI: https://github.com/Cray-HPE/python-csm-api-client
  - Documentation starting point:
    - https://github.com/Cray-HPE/sat/blob/integration/CONTRIBUTING.md
    - https://github.com/Cray-HPE/sat/blob/integration/docs/developer/README.md

- 3rd party open-source
  - https://kubernetes.io/docs/home/
  - https://kubernetes.io/docs/reference/kubectl/cheatsheet/
  - https://lmgtfy.com/?q=kubernetes+troubleshooting
  - https://www.elastic.co/guide/en/kibana/current/index.html
  - https://grafana.com/docs/
  - https://github.com/aelsabbahy/goss
  - http://docs.ansible.com/
  - https://kubernetes.io/docs/reference/kubectl/jsonpath/
  - https://stedolan.github.io/jq/manual/
  - http://www.compciv.org/recipes/cli/jq-for-parsing-json/
  - https://osinside.github.io/kiwi/

SUPERCOMPUTING: HPE CRAY EX TRAINING

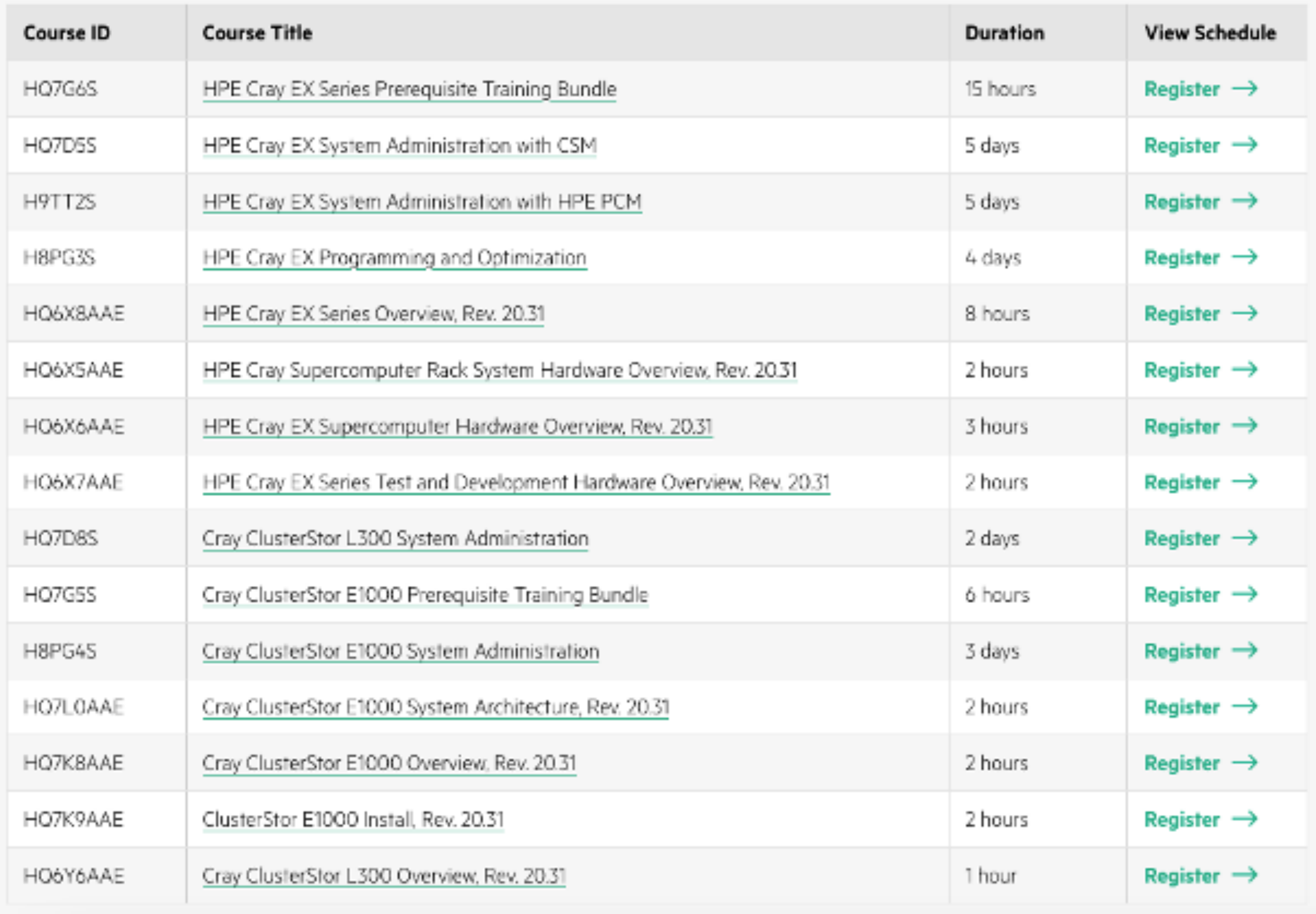
Where to start?

From HPE Edu

http://www.hpe.com/ww/training

- Select HPE Cray EX Series and ClusterStor Storage

https://education.hpe.com/ww/en/training/portfolio/servers.html#ServersLearningPathsIntro



Course ID	Course Title	Duration	View Schedule
HQ7G6S	HPE Cray EX Series Prerequisite Training Bundle	15 hours	Register →
HQ7D5S	HPE Cray EX System Administration with CSM	5 days	Register →
H9TT2S	HPE Cray EX System Administration with HPE PCM	5 days	Register →
H8PG3S	HPE Cray EX Programming and Optimization	4 days	Register →
HQ6X8AAE	HPE Cray EX Series Overview, Rev. 20.31	8 hours	Register →
HQ6X5AAE	HPE Cray Supercomputer Rack System Hardware Overview, Rev. 20.31	2 hours	Register →
HQ6X6AAE	HPE Cray EX Supercomputer Hardware Overview, Rev. 20.31	3 hours	Register →
HQ6X7AAE	HPE Cray EX Series Test and Development Hardware Overview, Rev. 20.31	2 hours	Register →
HQ7D8S	Cray ClusterStor L300 System Administration	2 days	Register →
HQ7G5S	Cray ClusterStor E1000 Prerequisite Training Bundle	6 hours	Register →
H8PG4S	Cray ClusterStor E1000 System Administration	3 days	Register →
HQ7L0AAE	Cray ClusterStor E1000 System Architecture, Rev. 20.31	2 hours	Register →
HQ7K8AAE	Cray ClusterStor E1000 Overview, Rev. 20.31	2 hours	Register →
HQ7K9AAE	ClusterStor E1000 Install, Rev. 20.31	2 hours	Register →
HQ6Y6AAE	Cray ClusterStor L300 Overview, Rev. 20.31	1 hour	Register →

RELATED PRESENTATIONS AND PAPERS

- CUG 2022
  - HPE Cray EX Shasta 22.03 Cray System Management Overview
  - Cray System Management for HPE Cray EX Systems
  - Dealing with Metrics Data – Where is it, How to get it, What to do with it?
- CUG 2021
  - User and Administrative Access Options for CSM-Based Shasta Systems
- CUG 2020
  - Advanced Topics in Configuration Management
- CUG 2019
  - Shasta Software Technical Workshop
  - Shasta System Management Overview
  - Reimagining Image Management in the New Shasta Environment
  - Hardware Discovery and Maintenance Workflows in Shasta Systems

THANK YOU

harold.longley@hpe.com