redhat.

Using Prometheus Operator to monitor OpenStack

Monitoring at Scale

Pradeep Kilambi & Franck Baudin / Anandeep Pannu
Engineering Mgr NFV Senior Principal Product Manager

15 November 2018

What we will be covering

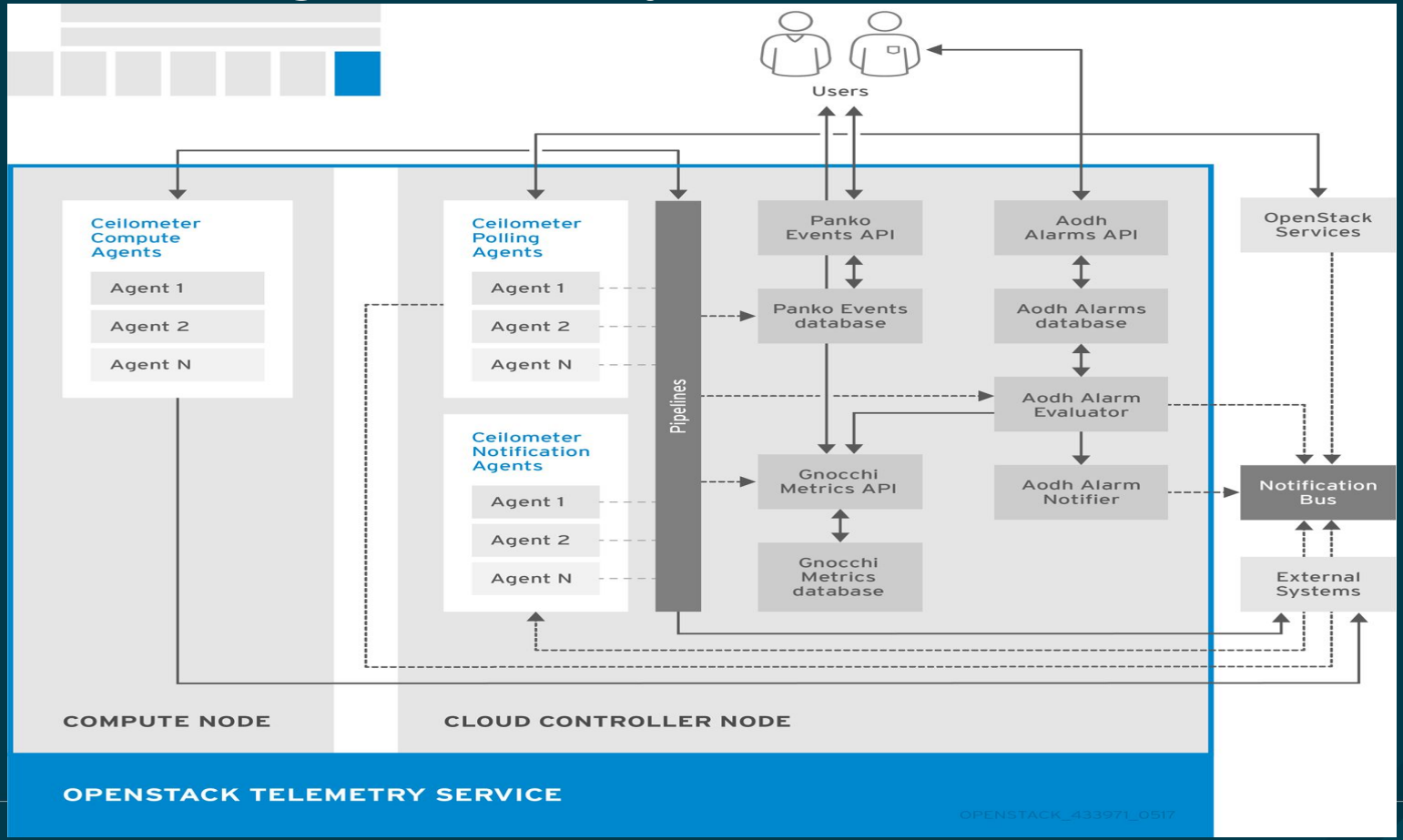
- Requirements
  - Why current OpenStack Telemetry is not adequate
  - Why Service Assurance Framework
- The solution approach
  - Platform solution approach
  - Multiple levels of API
- Detailed architecture
  - Overall architecture
  - Prometheus Operator
  - AMQ
  - Collectd plugins
- Configuration, Deployment & Perf results for scale
- Roadmap with future solutions

Issues & Requirements

Requirements for monitoring at scale

1. Address both telco (fault detection within few 100 ms) and enterprise requirements for monitoring
2. Handle sub-second monitoring of large scale clouds
3. Have well defined API access at multiple levels based on customer requirements
4. Time series database for storage of metrics/events should
    a. Handle the scale
        i. Every few hundred milliseconds, hundreds of metrics, hundreds of nodes, scores of clouds
    b. Be expandable to multi-cloud

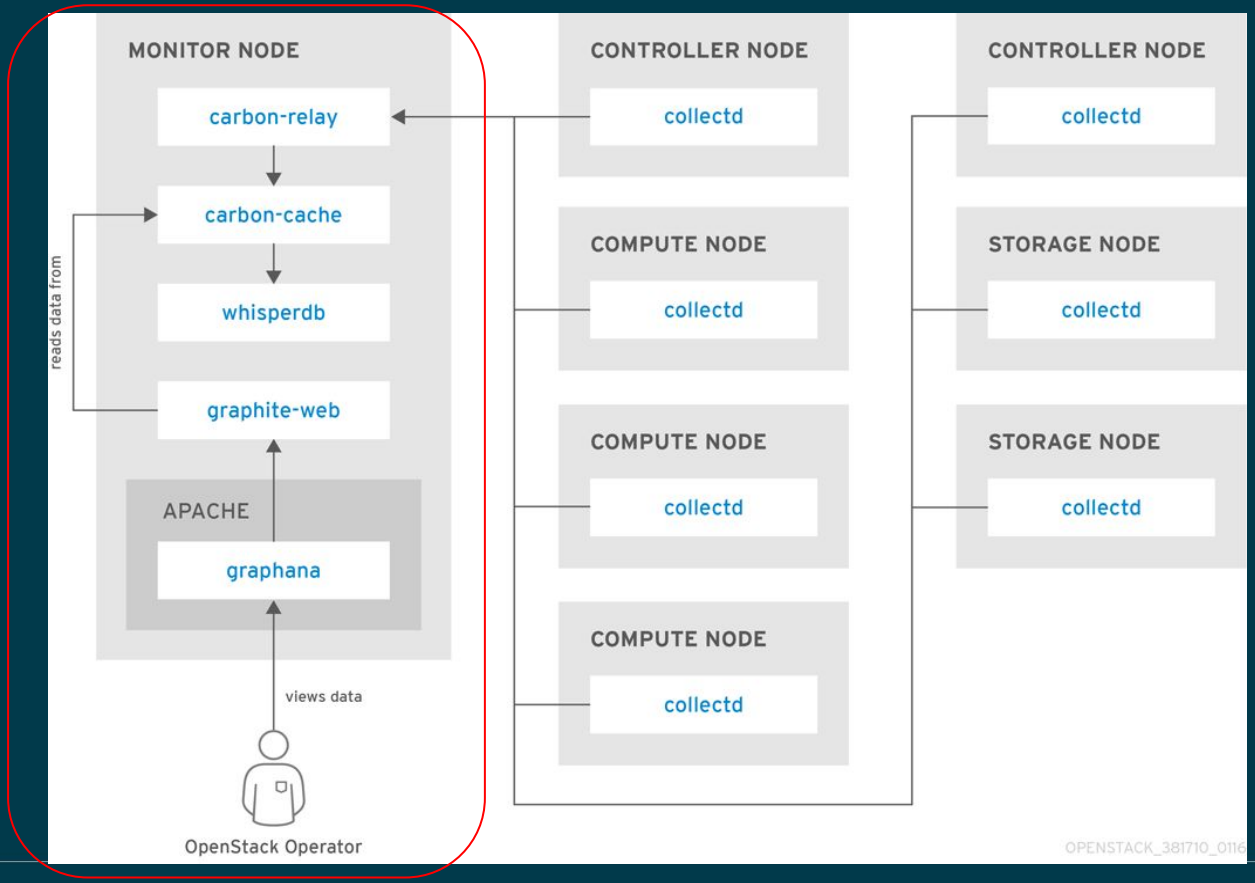
Monitoring / Telemetry - current stack



Monitoring at scale issues - Ceilometer

1. Current OpenStack telemetry & metrics/events mechanisms most suited for chargeback applications
2. A typical monitoring interval for Ceilometer/Panko/Aodh/Gnocchi combination is 10 minutes
3. Customers were asking for sub-second monitoring interval
    a. Implementing with current telemetry/monitoring stack resulted in “cloud down” situations
    b. Bottlenecks were
        i. Transport mechanism (http) to Gnocchi
        ii. Load on controllers by Ceilometer polling RabbitMQ

Monitoring / Telemetry - collectd



Monitoring at scale issues - collectd

1. Red Hat OpenStack Platform included collectd for performance monitoring using collectd plug-ins
    a. Collectd is deployed with RHEL on all nodes during a RHOSP deployment
    b. Collectd information can be
        i. Accessed via HTTP
        ii. Stored in Gnocchi
2. Similar issues as Ceilometer with monitoring at scale
    a. Bottlenecks were
        i. Transport mechanism (http)
            1. To consumers
            2. To Gnocchi
    b. Lack of a “server side” shipping with RHOSP

Platform & access issues

1. Ceilometer
    a. Ceilometer API doesn't exist anymore
    b. Separate Panko event API is being deprecated
    c. Infrastructure monitoring is minimal
        i. Ceilometer Compute provides limited Nova information
2. Collectd
    a. Access through http and/ or Gnocchi needs to be implemented by customer - no "server side"

Platform Solution Approach

Platform Approach to at scale monitoring

Problem:

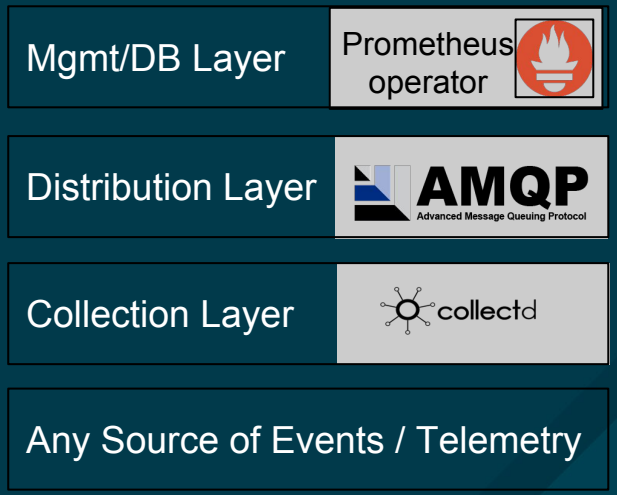
Current Openstack telemetry and metrics do not scale for large enterprises & to monitor the health of NFVi for telcos

Solution:

- Near real time Event and Performance monitoring at scale

Out of scope

- Mgmt application (Fault/Perf Mmgt) - Remediation - Root cause, Service Impact...



Platform Approach to at scale monitoring

1. APIs for 3 levels
   - At “sensor” (collectd agent) level
     - Provide plug-ins (Kafka, AMQP1) to allow connect to collectd via message bus of choice
   - At message bus level
     - Integrated, highly available AMQ Interconnect message bus with collectd
     - Message bus client for multiple languages
   - Time series database / management cluster level
     - Prometheus Operator
2. **CEILOMETER & GNOCCHI will continue to be used for chargeback and tenant metering**

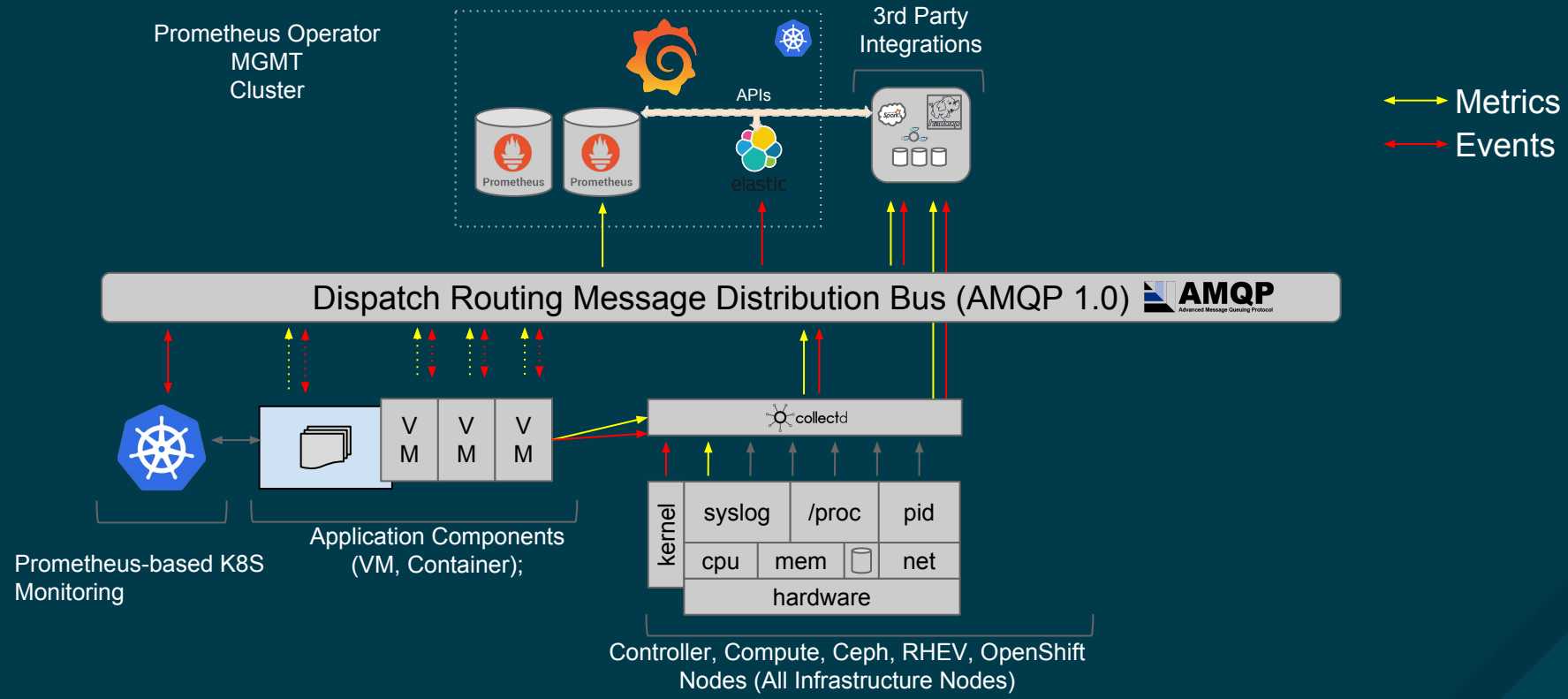
Service Assurance Framework Architecture

Architecture for infrastructure metrics & events

Based on the following elements

1. Collectd plug-ins for infrastructure & OpenStack services monitoring
2. AMQ Interconnect direct routing (QDR) message bus
3. Prometheus Operator database/management cluster
4. Ceilometer / Gnocchi for tenant/chargeback metering

Architecture for infrastructure metrics & events



Architecture for infrastructure metrics & events

Collectd Integration

- Collectd container -- Host / VM metrics collection framework
  - Collectd 5.8 with additional OPNFV Barometer specific plugins not yet in collectd project
    - Intel RDT, Intel PMU, IPMI
    - AMQP1.0 client plugin
    - Procevent -- Process state changes
    - Sysevent -- Match syslog for critical errors
    - Connectivity -- Fast detection of interface link status changes
  - Integrated as part of TripleO (OSP Director)

RHOSP 13 Collectd plug-ins

Pre-configured plug-ins:

1. Apache
2. Ceph
3. Cpu
4. Df (disk file system info)
5. Disk (disk statistics)
6. Memory
7. Load
8. Interface
9. Processes
10. TCPConns
11. Virt

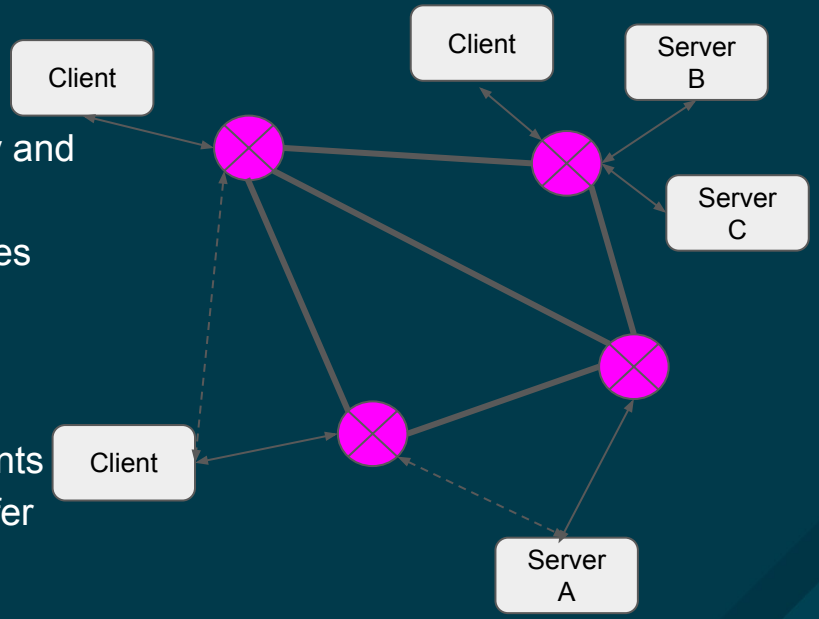
NFV specific plug-ins

1. OVS-events
2. OVS-stats
3. Hugepages
4. Ping
5. Connectivity
6. Procevent

Architecture for infrastructure metrics & events

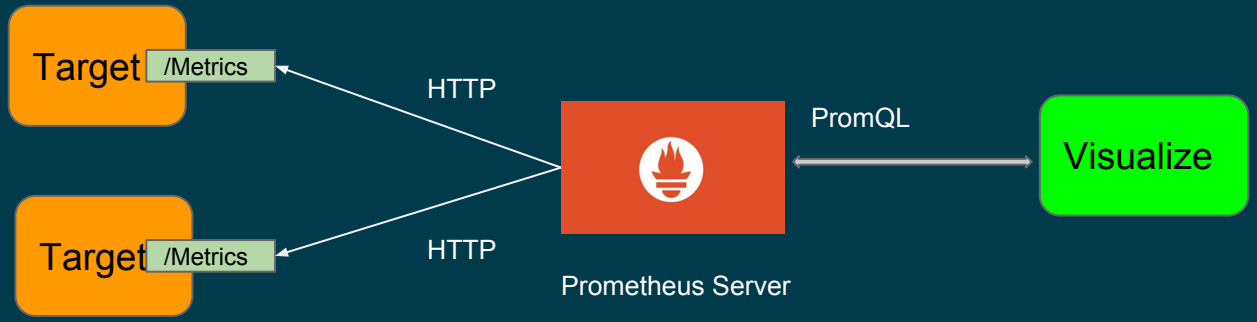
AMQ 7 Interconnect - Native AMQP 1.0 Message Router

- Large Scale Message Networks
  - Offers shortest path (least cost) message routing
  - Used without broker
  - **High Availability** through redundant path topology and re-route (not clustering)
  - Automatic recovery from network partitioning failures
  - **Reliable delivery without requiring storage**
- QDR Router Functionality
  - Apache Qpid Dispatch Router QDR
  - Dynamically learn addresses of messaging endpoints
  - Stateless - no message queuing, end-to-end transfer



High Throughput, Low Latency
Low Operational Costs

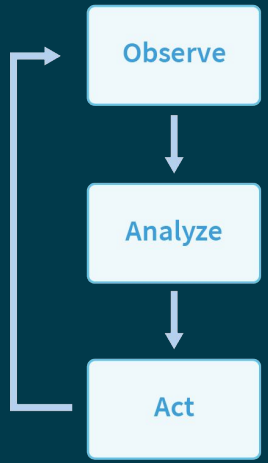
Prometheus



- Open Source Monitoring
- Only Metrics, Not Logging
- Pull based approach
- Multidimensional data model
- Time series database
- Evaluates rules for alerting and triggers alerts
- Flexible, Robust query language - *PromQL*

What is Operator?

- Automated Software Management
- purpose-built to run a Kubernetes application, with operational knowledge baked in
- Manage Installation & lifecycle of Kubernetes applications
- Extends native kubernetes configuration hooks
- Custom Resource definitions



Cluster “A” has 2 running pods:
- name: A-000, version 3.0.9
- name: A-001, version 3.1.0

Differences from desired config:
- should be version 3.1.0
- should have 3 members

How to get to desired config:
- Recover 1 member
- Back up cluster
- Upgrade to 3.1.0

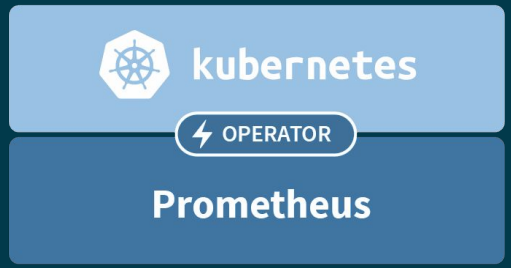
CoreOS

redhat.

Architecture for infrastructure metrics & events

Prometheus Operator

- Prometheus operational knowledge in software
- Easy deployment & maintenance of prometheus
- Abstracts out complex configuration paradigms
- Kubernetes native configuration
- Preserves the configurability



An Operator represents human operational knowledge in software, to reliably manage an application.

CoreOS

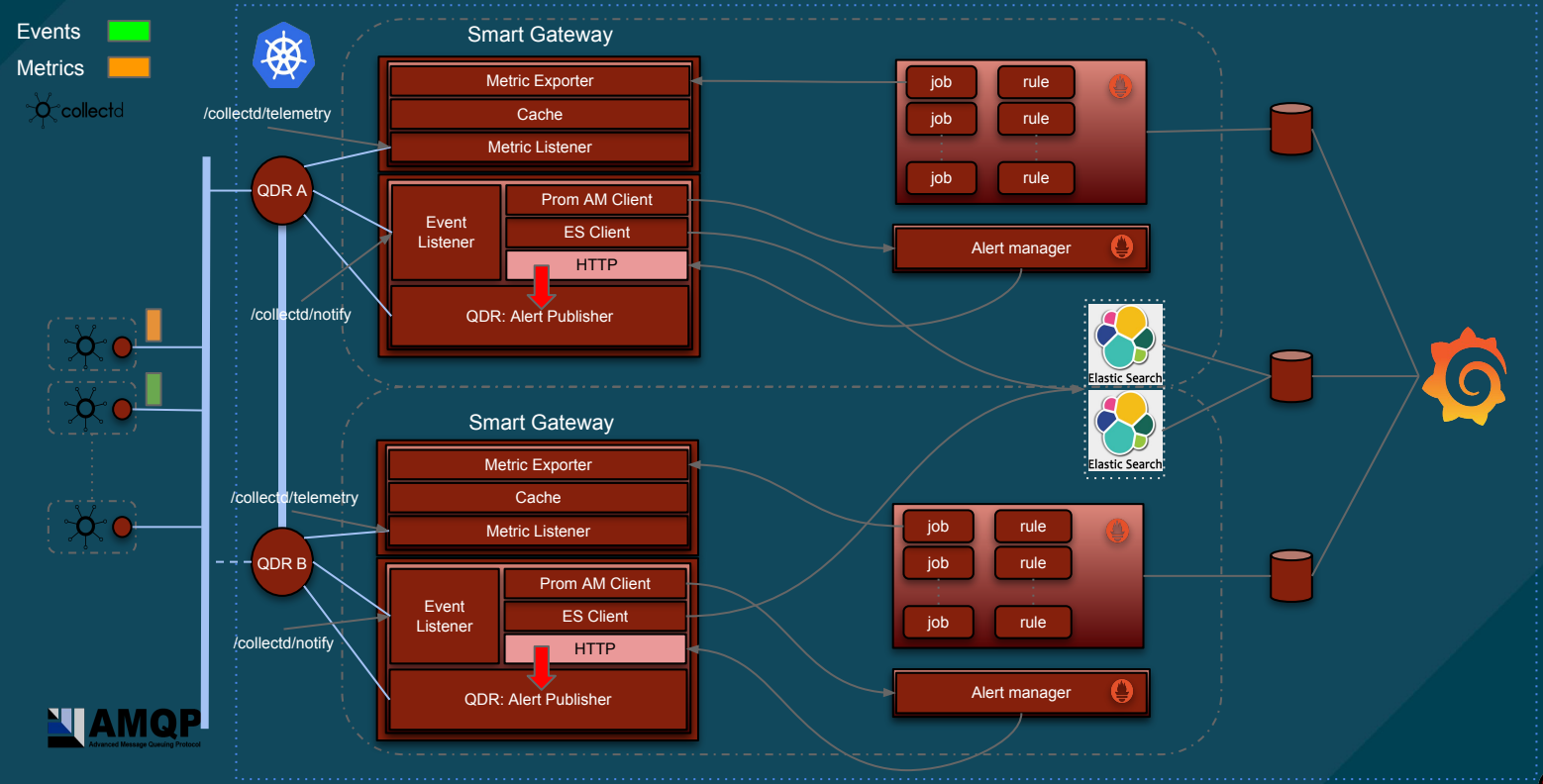
redhat.

Other Components

- ElasticSearch
  - System events and logs are stored in ElasticSearch as part of an ELK stack running in the same cluster as the Prometheus Operator
  - Events are stored in ElasticSearch and can be forwarded to Prometheus Alert Manager
  - Alerts that are generated from Prometheus Alert rule processing can be sent from Prometheus Alert Manager to the QDR bus
- Smart Gateway -- AMQP / Prometheus bridge
  - Receives metrics from AMQP bus, converts collectd format to Prometheus, coallates data from plugins and nodes, and presents the data to Prometheus through an HTTP server
  - Relay alarms from Prometheus to AMQP bus
- Grafana
  - Prometheus data source to visualize data

Architecture for infrastructure metrics & events

Prometheus Operator & AMQ QDR clustered

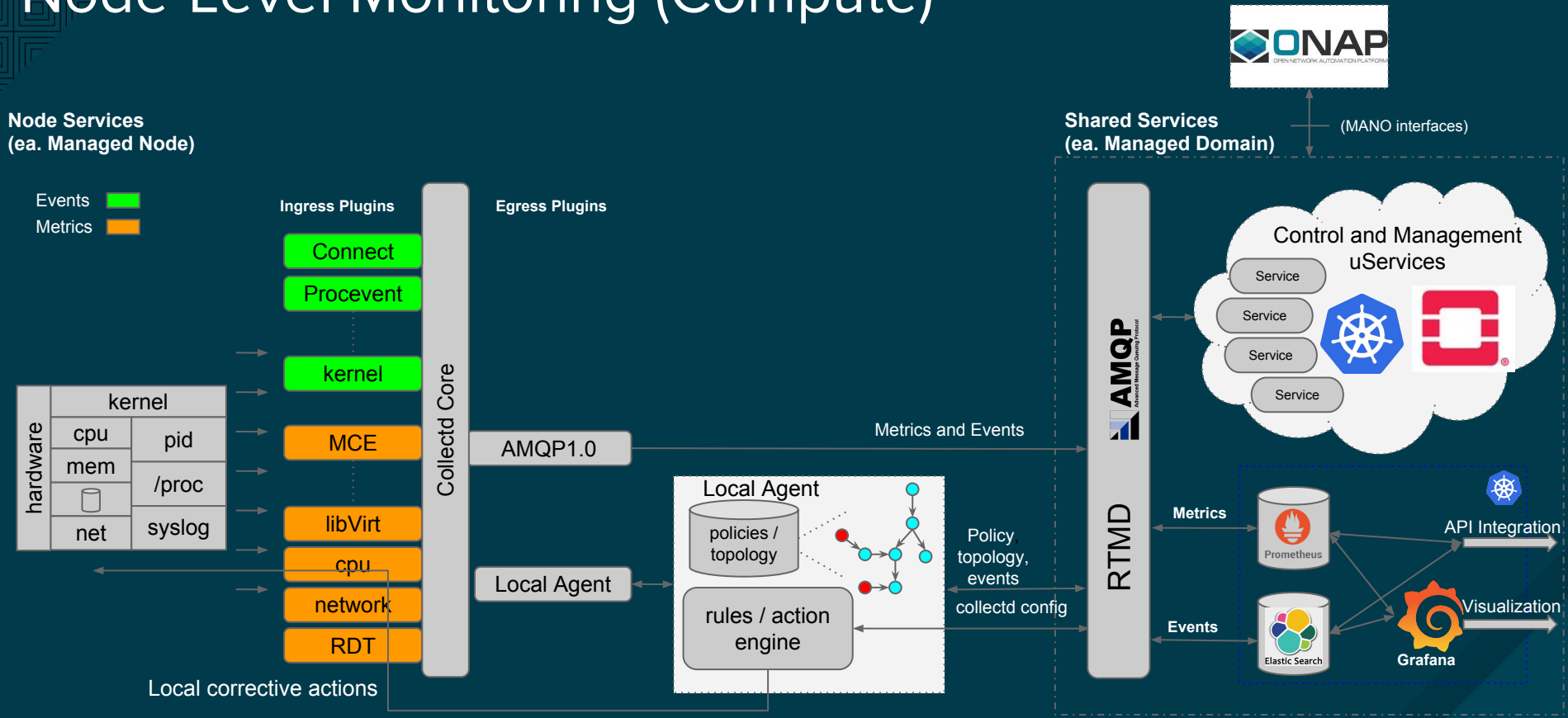


redhat.

Prometheus Management Cluster

- Runs Prometheus Operator on top of Kubernetes
- A collection of Kubernetes manifests and Prometheus rules combined to provide single-command deployments
- Introduces resources such as Prometheus, Alert Manager, ServiceMonitor
- Elasticsearch for storing Events
- Grafana dashboards for visualization
- Self-monitoring cluster

Node-Level Monitoring (Compute)



redhat.

Configuration & Deployment

TripleO Integration Of client side components

- Collectd and QDR profiles are integrated as part of the TripleO
- Collectd and QDRs run as containers on the openstack nodes
- Configured via heat environment file
- Each node will have a qpid dispatch router running with collectd agent
- Collectd is configured to talk to qpid dispatch router and send metrics and events
- Relevant collectd plugins can be configured via the heat template file

TripleO Client side Configuration

environments/metrics-collectd-qdr.yaml

```
## This environment template to enable Service Assurance Client side bits
resource_registry:
 OS::TripleO::Services::MetricsQdr: ../docker/services/metrics/qdr.yaml
 OS::TripleO::Services::Collectd: ../docker/services/metrics/collectd.yaml

parameter_defaults:
 CollectdConnectionType: amqp1
 CollectdAmqpInstances:
   notify:
     notify: true
     format: JSON
     presettle: true
   telemetry:
     format: JSON
     presettle: false
```

TripleO Client side Configuration

params.yaml

```
cat > params.yaml <<EOF
---
parameter_defaults:
  MetricsQdrConnectors:
    - host: 192.168.24.11
      port: 20001
      role: inter-router
    - host: 192.168.24.22
      port: 20001
      role: inter-router
EOF
```

Client side Deployment

Using overcloud deploy with collectd & qdr configuration and environment templates

```
cd ~/tripleo-heat-templates
git checkout master
cd ~
cp overcloud-deploy.sh overcloud-deploy-overcloud.sh
sed -i 's/usr\/share\/openstack-/home\/stack\//g' overcloud-deploy-overcloud.sh
./overcloud-deploy-overcloud.sh -e
/usr/share/openstack-tripleo-heat-templates/environments/metrics-collectd-qdr.yaml -e
/home/stack/params.yaml
```

Server Side SA Deployment

Deployed using TripleO (OSP Director) & Openshift-ansible

- Server side consists of OpenShift cluster running on 3 baremetal nodes
- is deployed using TripleO (OSP Director)
- Uses ironic to provision nodes
- tripleO to bootstrap openshift-ansible and deploy OpenShift cluster
- Prometheus Operator, grafana, elastic search, central QDR deployed on top of OpenShift as Ansible playbook bundles(apb)
- The server side Telemetry infrastructure is independent of OpenStack cloud

Deploying Telemetry Framework

Using tripleo overcloud deploy

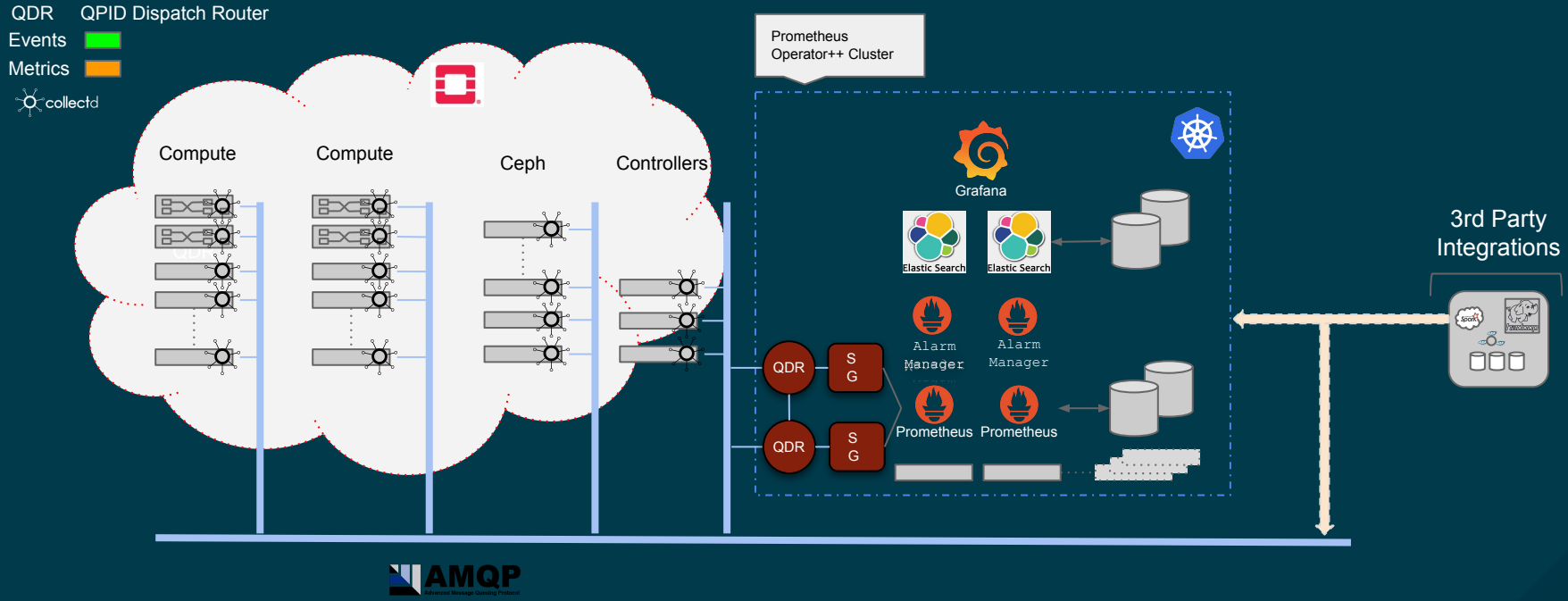
```
$ openstack overcloud deploy --stack telemetry --templates /home/stack/tripleo-heat-templates/ -r
/home/stack/tripleo-heat-templates/my_roles.yaml  -e
/home/stack/tripleo-heat-templates/environments/openshift.yaml -e
/home/stack/tripleo-heat-templates/environments/openshift-cns.yaml -e
/home/stack/tripleo-heat-templates/params.yaml -e
/home/stack/tripleo-heat-templates/environments/networks-disable.yaml -e
/home/stack/network-environment.yaml -e /home/stack/containers-prepare-parameter.yaml
```

Post deployment Overview

$ openstack server list

ID	Name	Status	Task State	Power State	Networks
91278733-73cb-44a3-8a7a-a82828414d12	overcloud-controller-0	ACTIVE	-	Running	ctlplane=192.168.24.13
332b5661-fe97-4ec3-8ba6-2cc2851a0039	overcloud-controller-1	ACTIVE	-	Running	ctlplane=192.168.24.11
a9100dae-f053-4266-8a80-0b417cc0c19d	overcloud-controller-2	ACTIVE	-	Running	ctlplane=192.168.24.22
55b1898e-381c-4037-90ee-4514bb28b277	overcloud-novacompute-0	ACTIVE	-	Running	ctlplane=192.168.24.17
dceaa6b2-963a-40a3-9f1f-07c179864786	telemetry-node-0	ACTIVE	-	Running	ctlplane=192.168.24.9
f67475a2-2b23-49e7-802d-1115eba39afa	telemetry-node-1	ACTIVE	-	Running	ctlplane=192.168.24.30
e0765d77-27fa-4bb2-92ff-d707aa7b19a2	telemetry-node-2	ACTIVE	-	Running	ctlplane=192.168.24.16

Service Assurance Cluster



redhat.

Deployment Summary

1. AMQP1 collectd plug-in
    - Proton (“send” side) client for AMQ integrated
2. Proton send/receive client for connecting to AMQ for consumers
3. Collectd plug-ins from Barometer project integrated
4. Separate management cluster running on OpenShift
    - At least two to three servers (for HA)
    - Each server has one QDR (Qpid Dispatch Router)
    - Prometheus Operator which consists of
        i. Prometheus
        ii. Prometheus Config
            1. Multiple Prometheus for HA (one per server)
        iii. Prometheus Alert Manager (one per server)
5. Director installation & configuration of all additional OpenStack components

Prometheus Metrics Scale

Hardware Test Setup

- 128G Memory
- 2.9GHz, 2 Socket, 12 physical cores, 24 hyperthreaded cores
- Drives -- sdc was used as the data drive for Prometheus
  - sda disk 447.1G SAMSUNG MZ7LM480
  - sdb disk 1.8T ST2000NX0403
  - sdc disk 1.8T ST2000NX0403
  - 7200RPM SATA3 6Gb/s 128M Internal
- 10Gbps Network interfaces

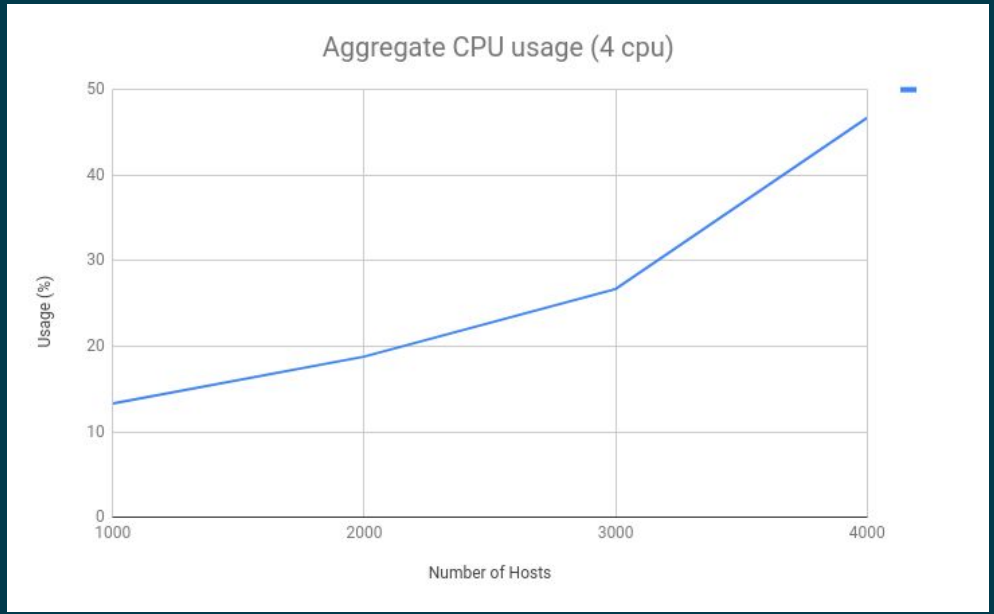
Prometheus Metrics Scale

Test Methodology

1. Prometheus scale dependent upon
    - Number of raw metrics
    - Number of labels per timeseries
    - Number of rules applied to each timeseries
        i. Data rewrite
        ii. Alerting
    - Data export load
2. Determine CPU load for each host tier for data ingestion only. Adjust GOMAXPROC
3. Add representative number of rules per timeseries
4. Target 4000 hosts with 100s of metrics each
5. Million metrics per second

Prometheus Metric Scale

Data Storage Only

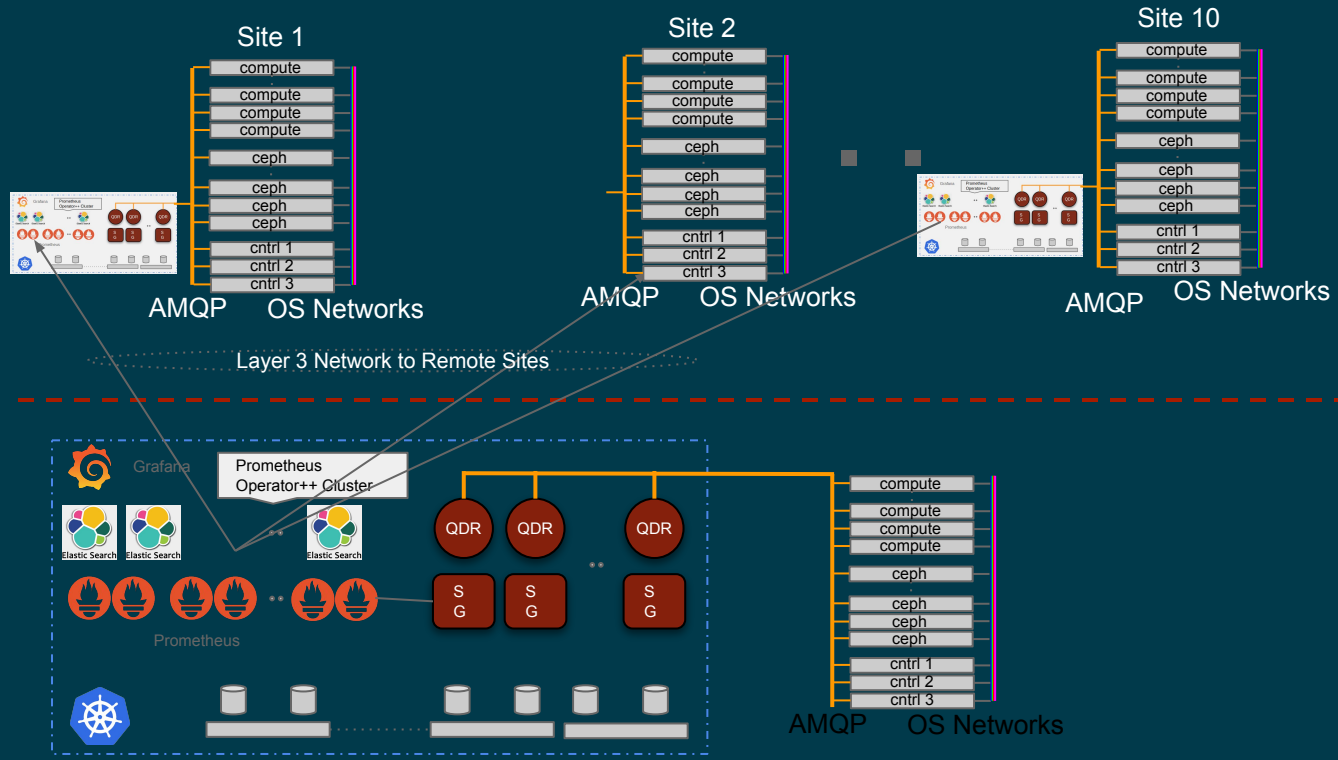


Roadmap to the future

Release Cadence

13 Queens	14 Rocky	15 Stein	BEYOND
* Backport OSP 14 Tech Preview SAF	* Service Assurance Framework TP * AMQ integration with SAF * Ansible based Prometheus Mgmt Cluster deployment	* Service Assurance Framework GA * Prometheus Mgmt Cluster Deployment by Director	Central SAF & Prometheus Mgmt Cluster for multi-site OSP deployment

Monitoring multiple cloud with multiple Prometheus Instances



redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos