

Service Assurance Framework

MVP Overview

SA Team

Executive Summary

- Deliver a Service Assurance Framework aligned to Telco “Carrier Grade” requirements but just as applicable to Enterprises who run business critical workloads. ([5K nodes, 200 metrics, 1 sec interval](#))
- Foundation for future applications:
 - Efficient, “near real-time” Fault Management
 - Enables enhanced High Availability
- Cross-platform (RHOSP, RHEV, OCP) with 3rd party integration
 - Initially delivered as RHOSP project (to fulfill urgent NFV need)
 - Designed and deployed as standalone solution (i.e. as independent of products as possible)
- Strong integration with Community and partners
 - OPNFV Barometer, Doctor
 - Intel, SevOne, KDDI
- Leverage popular open-source technologies: Collectd, AMQP1.0, Prometheus, ...
- Eventual integration with ONAP

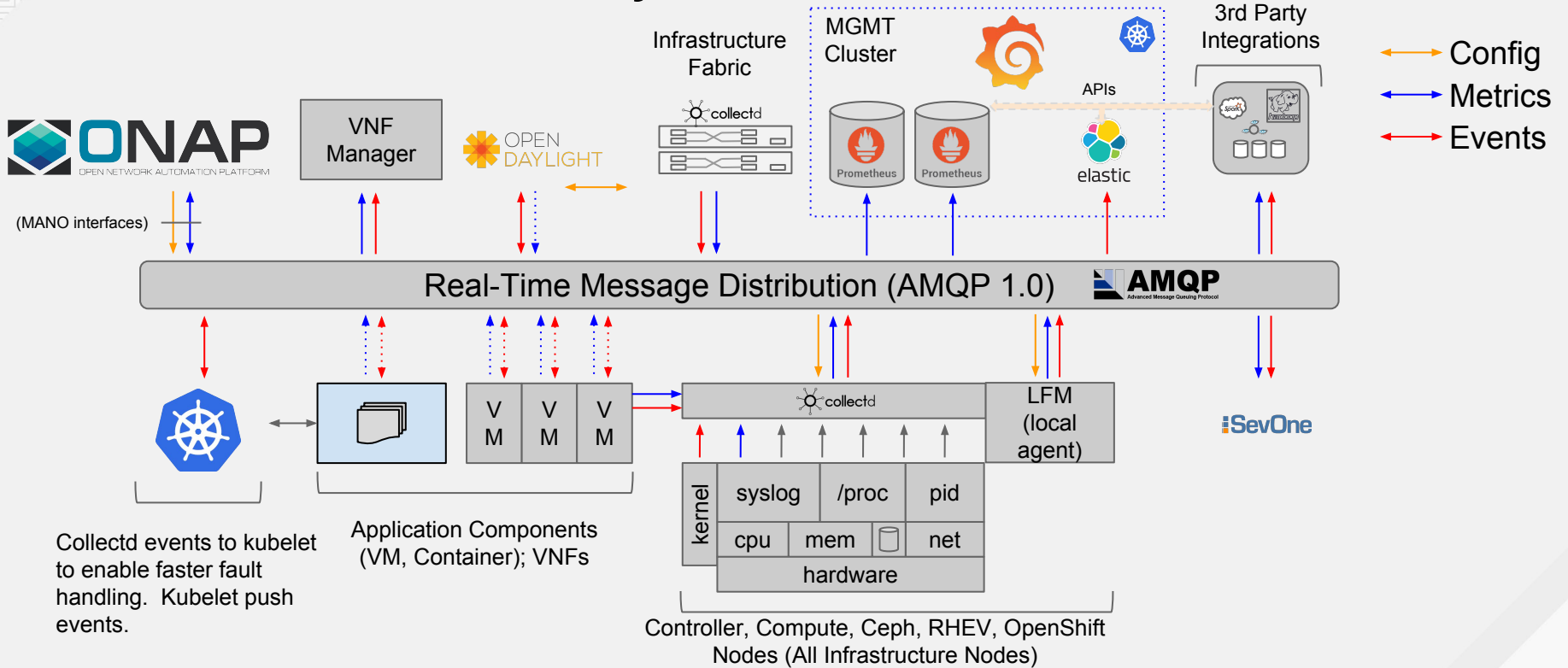
Overview

- Goal is the productization of an infrastructure monitoring and event distribution framework, which enables applications to run with reliability and availability approaching traditional telco requirements
- The framework enables the collection and distribution of metrics and events
 - Data usage include analysis, storage and an automated remediation, PM, billing etc. processes
 - Our initial focus is to provide the ability to detect, transport and act upon fault conditions in less than 100s of milliseconds (with a target of 50ms for low-level network faults)
 - Fault events are more important to SA performance than metrics (failed = service down vs. performance SLA violation = service quality degraded)
- The framework leverages existing tools:
 - Collectd for both event detection and metrics acquisition
 - A common message bus, AMQP 1.0-based, for both metrics and events
 - Prometheus for metric storage (short-term)
- Applications themselves **cannot** do this alone
 - Can detect that “something” is faulty, but not what; no info & knowledge about the infrastructure
 - High frequency polling for fast detection is too resource intensive

Service Assurance / Fault Management Goals

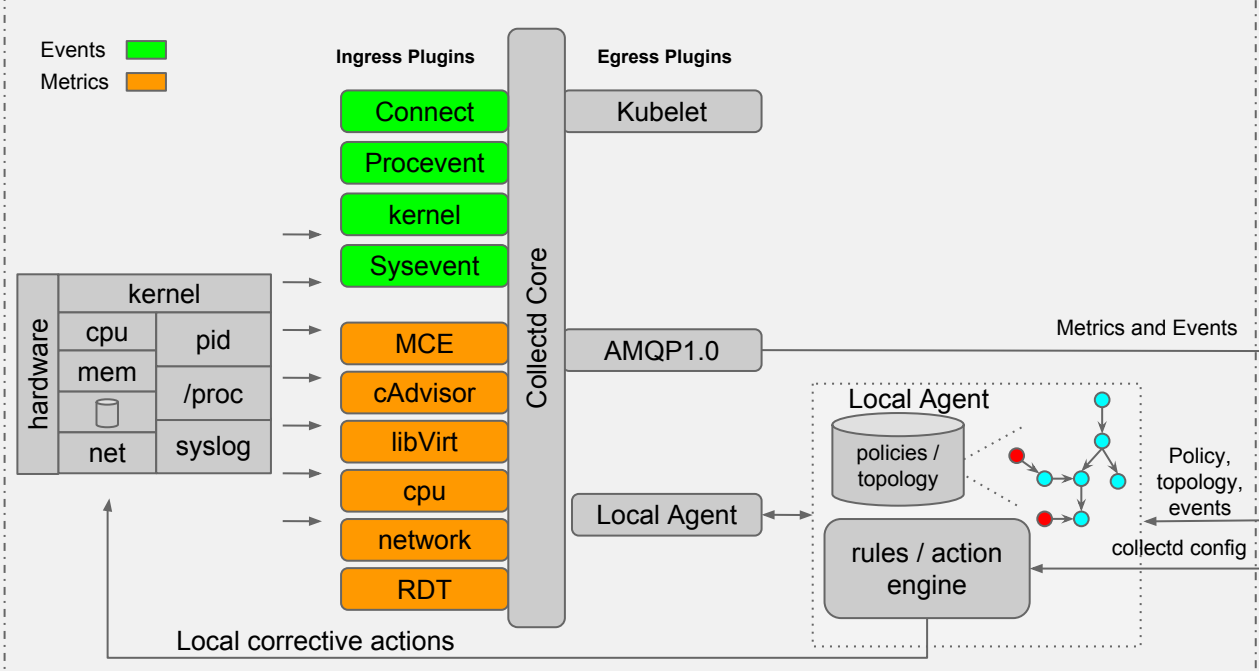
- Keys to enabling telco levels of service availability in a cloud environment
 - Requires a monitoring and event detection framework that distributes correlated fault information to components at the infrastructure and user layers with an end-to-end latency target of < 50 milliseconds
 - Requires a local fault manager (LFM) that can react quickly (<10's of milliseconds) to local host faults by applying local fault policies specified by the higher service layers. LFM can communicate locally to VMs and/or containers to enable fast fault detection and remediation.
 - For a cloud infrastructure, the network assumes a larger role in availability than individual compute hosts. A fast node failure detection mechanism is needed to differentiate between host and switch failures and interact with the underlying network control (SDN/ODL)
- Leverage a Real-Time, Message Distribution framework (AMQP1.0) to provide a common transport mechanism for events and metrics
- A high-availability framework cannot have humans in the remediation / recovery process if 5NINES Service Availability is a goal.
- Logging information is provided for detailed, post-mortem analysis of infrastructure problems.

Service Assurance System Overview

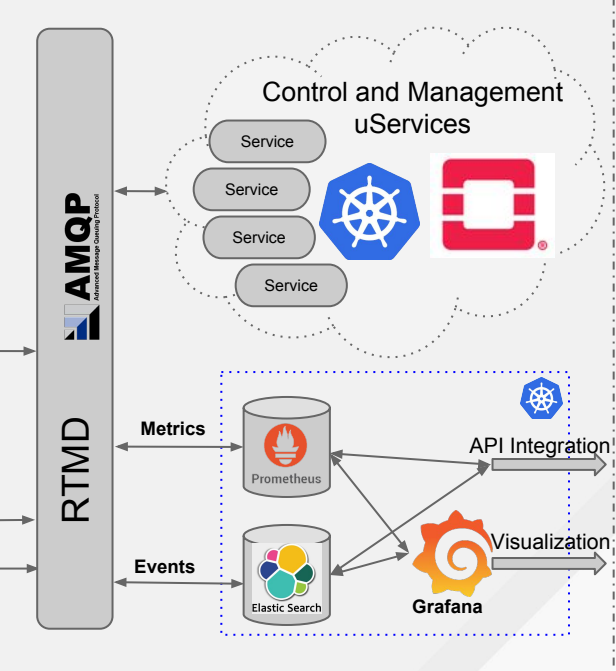


Node-Level Monitoring (Compute)

Node Services (ea. Managed Node)



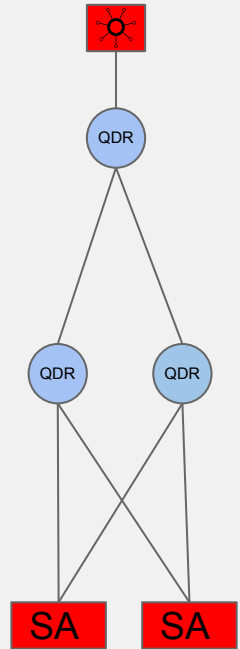
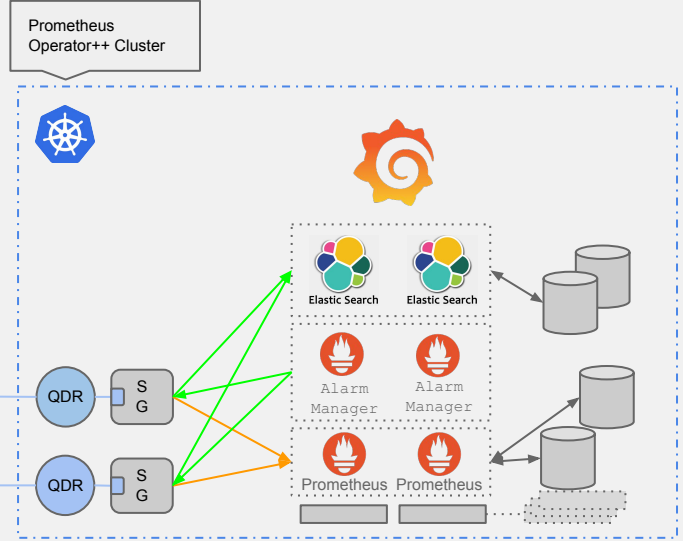
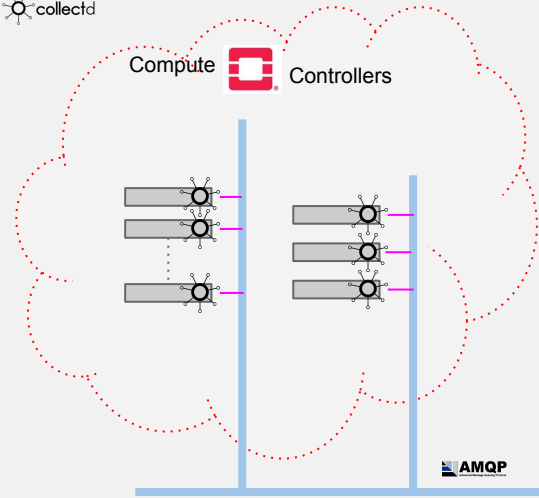
Shared Services (ea. Managed Domain)



(MANO interfaces)

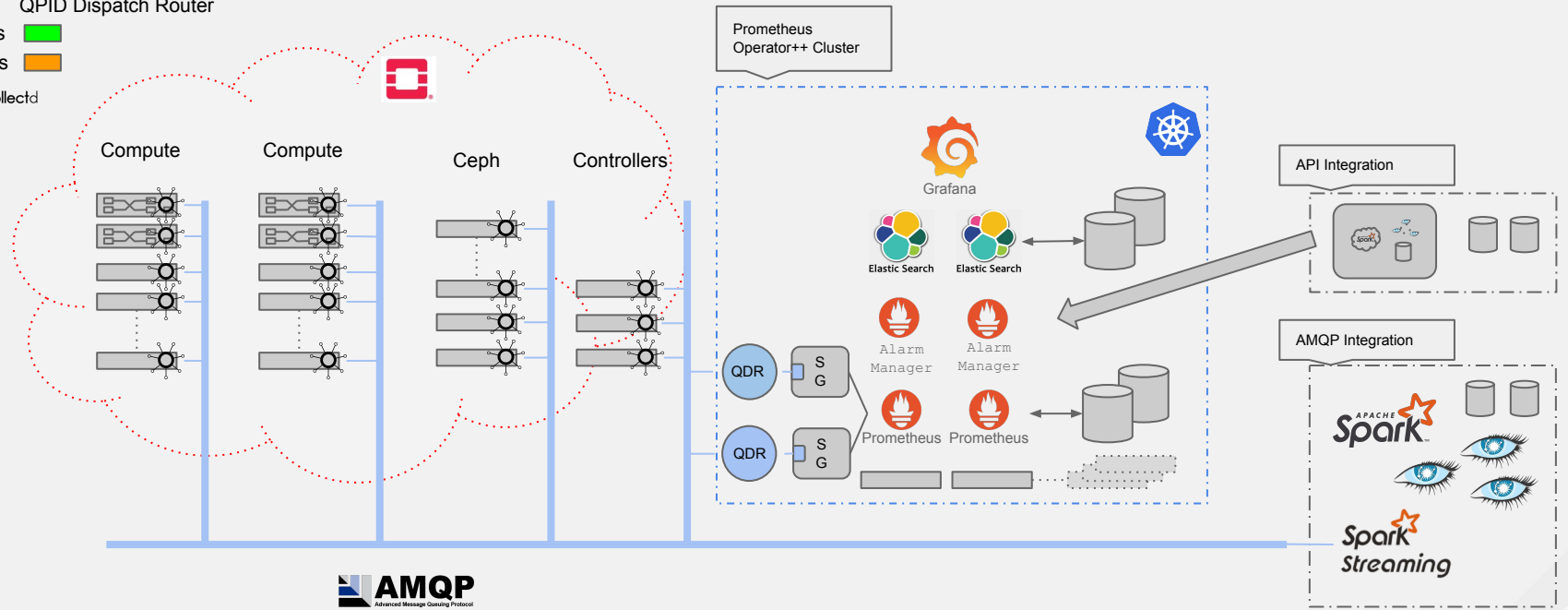
Service Assurance PoC

QDR Qpid Dispatch Router
 Events 
 Metrics 
 collectd






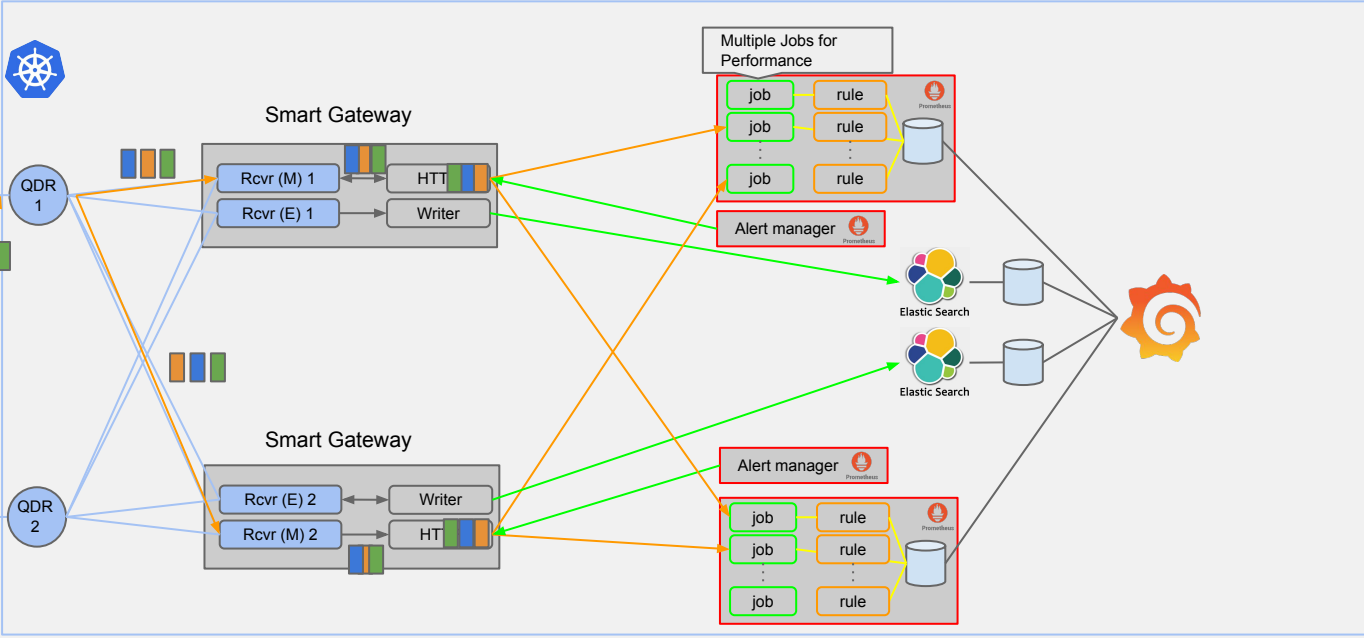
Service Assurance Cluster

QDR QPID Dispatch Router
 Events ■
 Metrics ■
 collectd

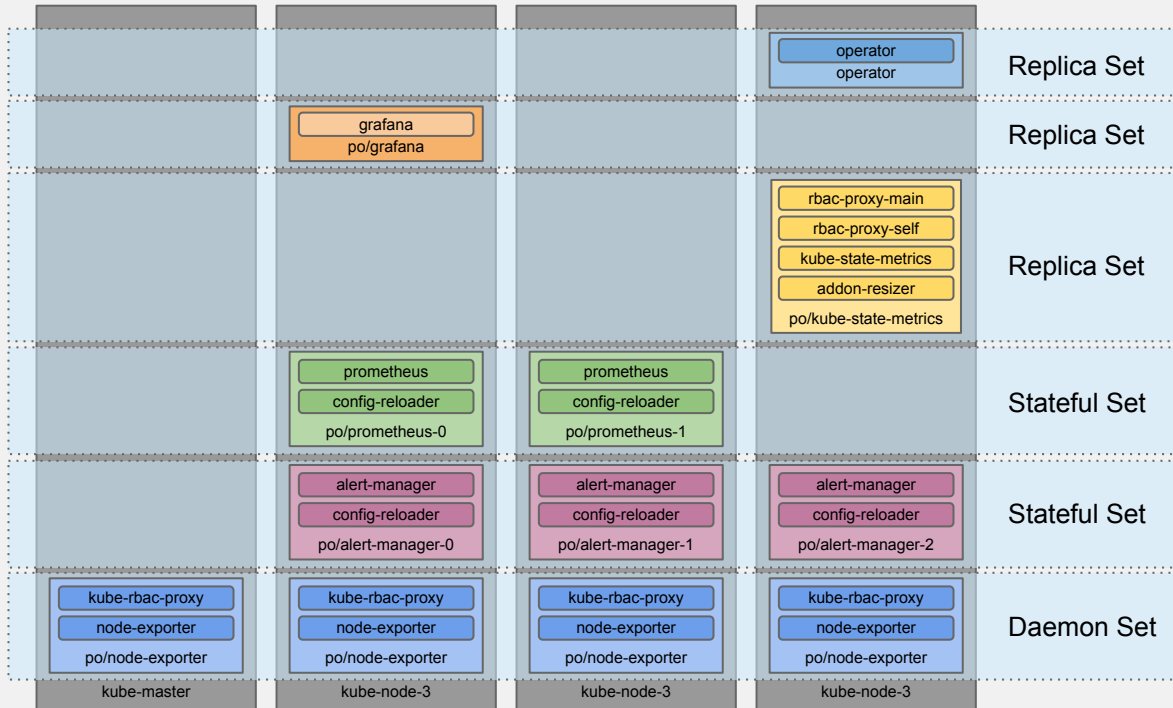


Smart Gateway (SG) / Prometheus

Events 
Metrics 
 collectd



Prometheus Operator



MVP Components



- [AMQP1.0](#) - based messaging
 - AMQP1.0 is available upstream through the [Apache QPID](#) project and as a Red Hat product (newer protocol than AMQP0.9 RabbitMQ)
 - AMQP1.0 is not currently envisioned as a replacement for AMQP0.9 (RabbitMQ)
 - Partners can integrate with the the SA project by using one of the many language flavors of the AMQP1.0 [client](#) (C, Golang, Python, Java)
 - The message bus will carry metrics and events and third-party applications can subscribe to either or both.
 - Metrics are delivered “best-effort” and events are reliable

MVP Components

- Barometer Containers
 - [Collectd](#) container -- Host / VM metrics collection framework
 - Currently Collectd 5.8 with additional 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
 - [Apache QPID dispatch router](#) container -- AMQP1.0 message bus router
 - Ansible scripts for configuration of containers integrated with installation process

MVP Components

- [Prometheus-Operator](#) -- Prometheus in Kubernetes
 - A collection of Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide single-command deployments of end-to-end Kubernetes cluster monitoring with Prometheus
 - Self-monitoring of cluster
- ElasticSearch
 - System events and logs are stored in ElasticSearch as part of an ELK stack running in the same cluster as the Prometheus Operator
- [Smart Gateway](#) -- AMQP / Prometheus bridge
 - Receives metrics from AMQP bus, converts collected format to Prometheus, collates data from plugins and nodes, and presents the data to Prometheus through an HTTP server
 - Relays alarms from Prometheus to AMQP bus

MVP Functionality

- Metrics at Scale
 - Store 100's of metrics from 1000 hosts at 1 second granularity
- Alerting at Scale
 - Allow several (simple) threshold rules per series
- Application subscription to simple events
- Integration with 3rd party systems
 - Metrics
 - Events

Beyond MVP Functionality

- Physical / Virtual event correlation
- Fault suppression

Productization Plan

RHOSP 13: Deliver SA Framework MVP (stretch goal)

- limited availability to a few partners (Intel, SevOne, ...).
- Supported by SA team.
- RDO Integration (OpenStack Q)

RHOSP 14: Deliver SA Framework v1.0 TP

RHOSP 15: Deliver SA Framework v1.0 GA

RHOSP 16: Deliver SA Framework v1.1 GA (Ready for LTS)

RHOSP 13 - Spring 18

RHOSP 14 - Fall 18

RHOSP 15 - Spring 19

RHOSP 16 - Fall 19

Evolution

The complete Service Assurance framework will be developed in stages:

- 1.) MVP: Low-latency monitoring framework operating at the infrastructure level. Only coarse-grained models of physical hosts and control plane services exists. SA Policies are initially fixed.
 - Upstream collectd functionality
 - Inclusion of the RTMD bus
 - Prometheus RTMD bus interface
 - Visualization tools integration
- 2.) Fault correlation service (to relate the faults to affected virtual entities / applications)
- 3.) Support for distribution, instantiation and management of user-supplied FM/PM policies
 - Being worked in ETSI NFV & GANA, OPNFV & ONAP (Drools based policies)
- 4.) TOSCA based Application (VNF) & Service (multiple applications) models
 - Models of events & policies allow FM/PM information to be used by VNFs & Services
 - Being worked on in ETSI NFV, OASIS & ONAP



THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHatNews



youtube.com/user/RedHatVideos