Monasca

monitoring-as-a-service (MONaaS) autoscaling with Heat

SFBay OpenStack
August 20, 2015

Dexter Fryar
Kanagaraj Manickam
Thomas Goepel

Outline

About me

Intro to Monasca

Architecture overview

Operational overview

Enough slideware let’s see the demo

About me

- 15 years at HP in systems software engineering roles
- Worked on internal and external embedded storage subsystems
- Heterogeneous solutions engineering
- Joined HP OpenStack engineering in 2011
- Foundation services – metering and billing
- Currently working on the Monasca project

Intro

- **Monasca the name**

  #monitoringatscale

- **How did we get here**

  public cloud at scale lessons learned, datadog, openstack

- **Goals for Monasca**

  open-source multi-tenant, highly scalable, performant, fault-tolerant monitoring-as-a-service solution that integrates with OpenStack

- **Uses**

  application, service, tenant, component

- **Model**

  REST API for high-speed metrics processing and querying and has a streaming alarm engine and notification engine
  access via → agent, rest API, statsd

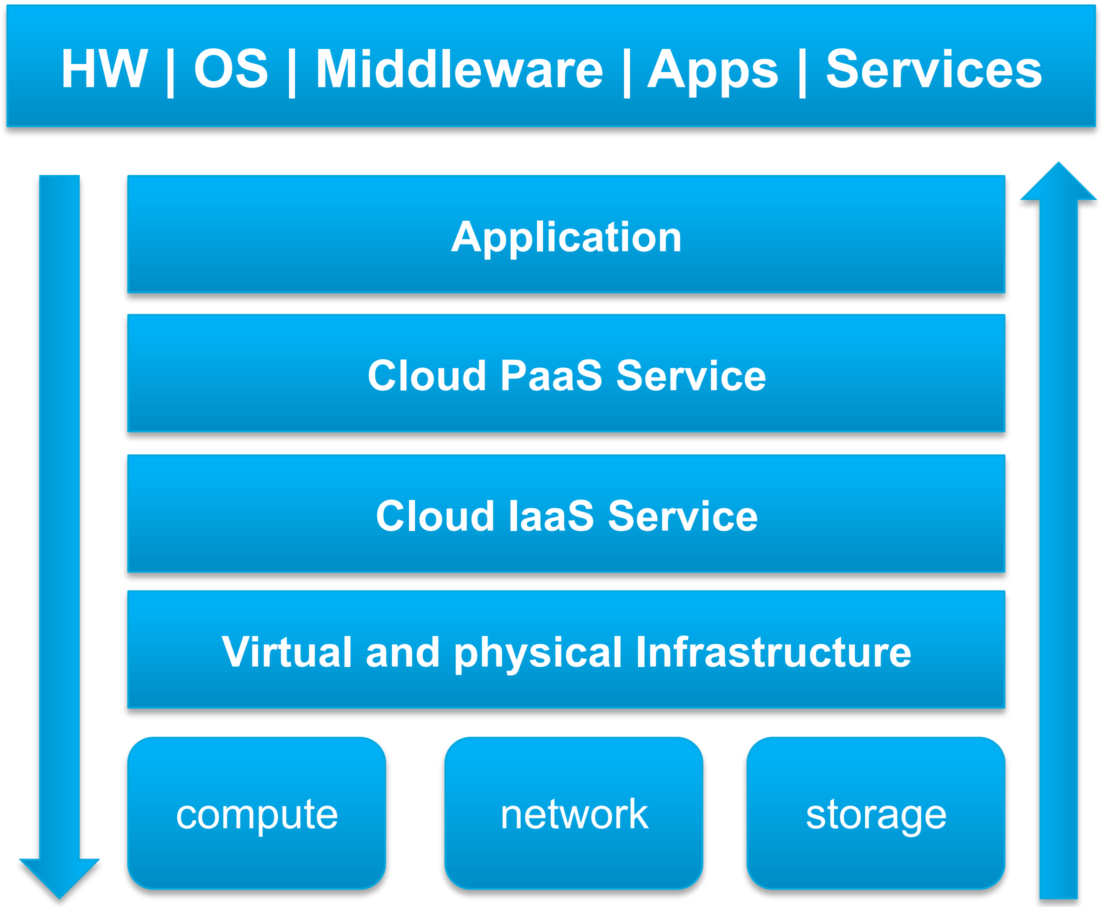
Monitoring Cloud Platforms

Monitoring, analyses, remediation, optimization

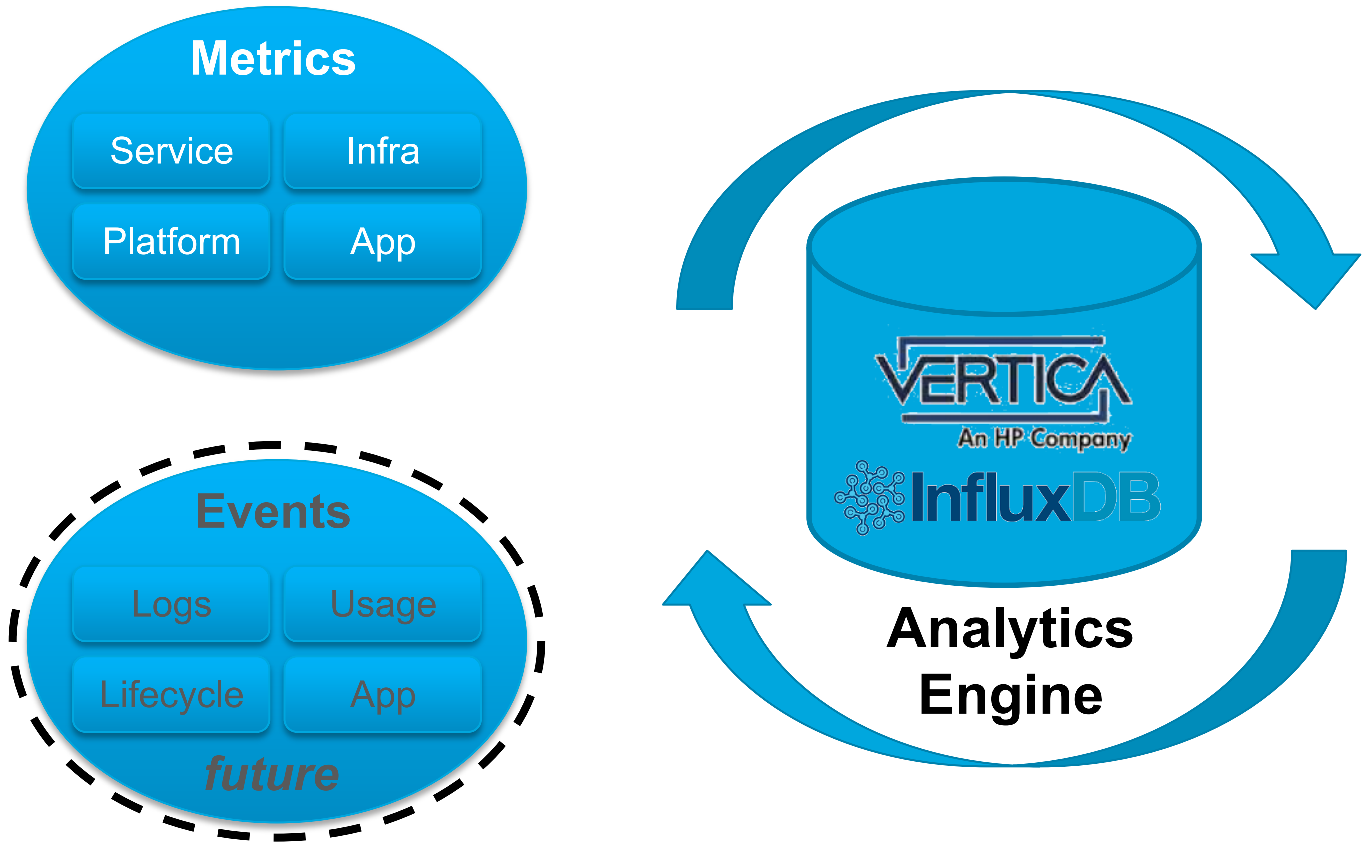
Workload: performance, availability, security, compliance

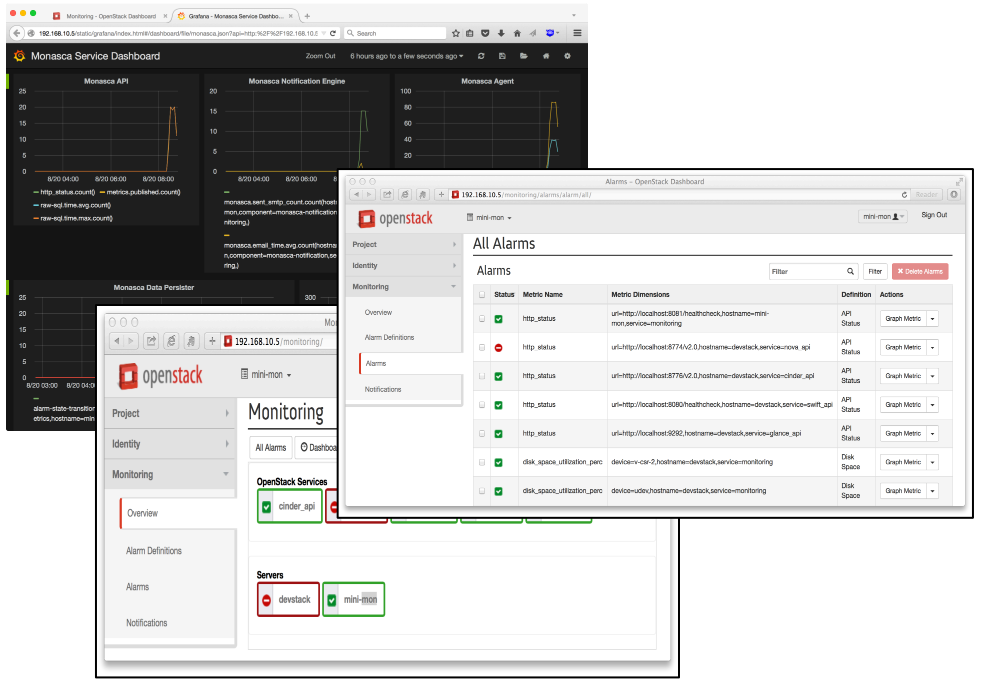
Cloud service health and availability

Virtual and physical compute, network, and storage monitoring


HW | OS | Middleware | Apps | Services
Application
Cloud PaaS Service
Cloud IaaS Service
Virtual and physical Infrastructure
compute
network
storage

What is Monasca?





192.168.10.5/static/grafana/index.html#/dashboard/file/monasca.json?api=http:%2F%2F192.168.10.5

- http_status.count()
- metrics.published.count()
- raw-sql.time.avg.count()
- raw-sql.time.max.count()

Alarms – OpenStack Dashboard

192.168.10.5/monitoring/alarms/alarm/all/

All Alarms

Alarms

Status	Metric Name	Metric Dimensions	Definition	Actions
✓	http_status	url=http://localhost:8081/healthcheck,hostname=mini-mon,service=monitoring	API Status	Graph Metric
⛔	http_status	url=http://localhost:8774/v2.0,hostname=devstack,service=nova_api	API Status	Graph Metric
✓	http_status	url=http://localhost:8776/v2.0,hostname=devstack,service=cinder_api	API Status	Graph Metric
✓	http_status	url=http://localhost:8080/healthcheck,hostname=devstack,service=swift_api	API Status	Graph Metric
✓	http_status	url=http://localhost:9292,hostname=devstack,service=glance_api	API Status	Graph Metric
✓	disk_space_utilization_perc	device=v-csr-2,hostname=devstack,service=monitoring	Disk Space	Graph Metric
✓	disk_space_utilization_perc	device=udev,hostname=devstack,service=monitoring	Disk Space	Graph Metric

Integrations

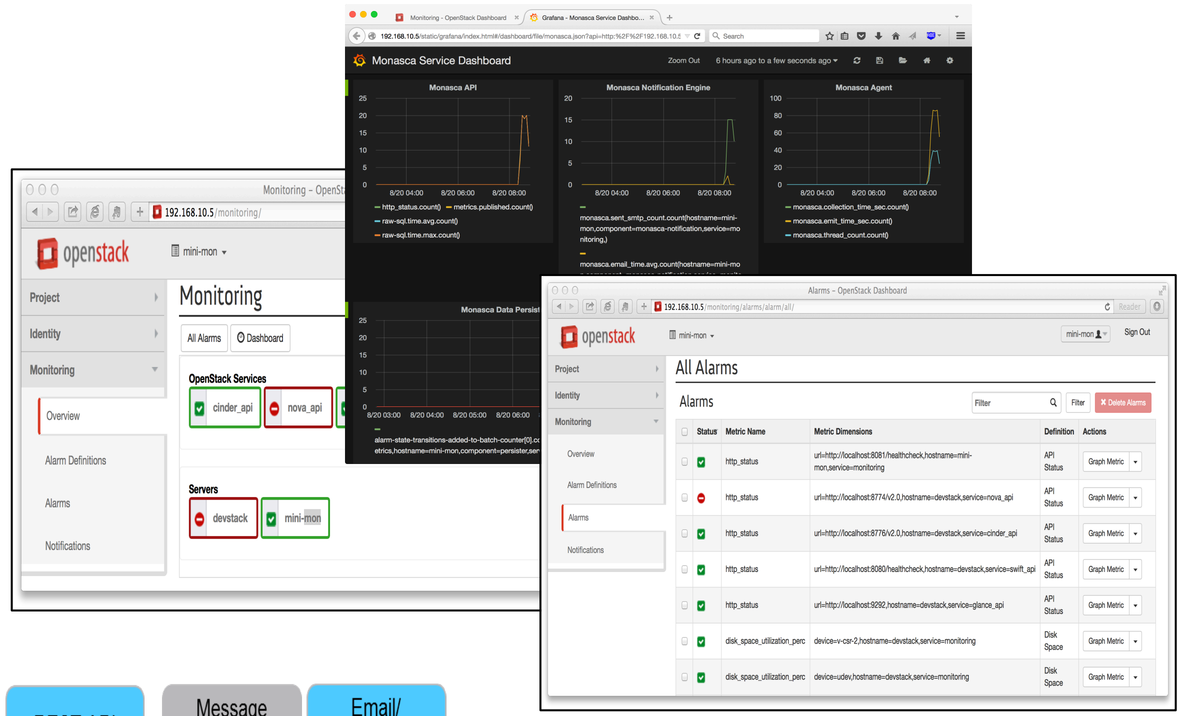
Atlassian

PAGERDUTY

Benefits

Monitoring as a service at scale

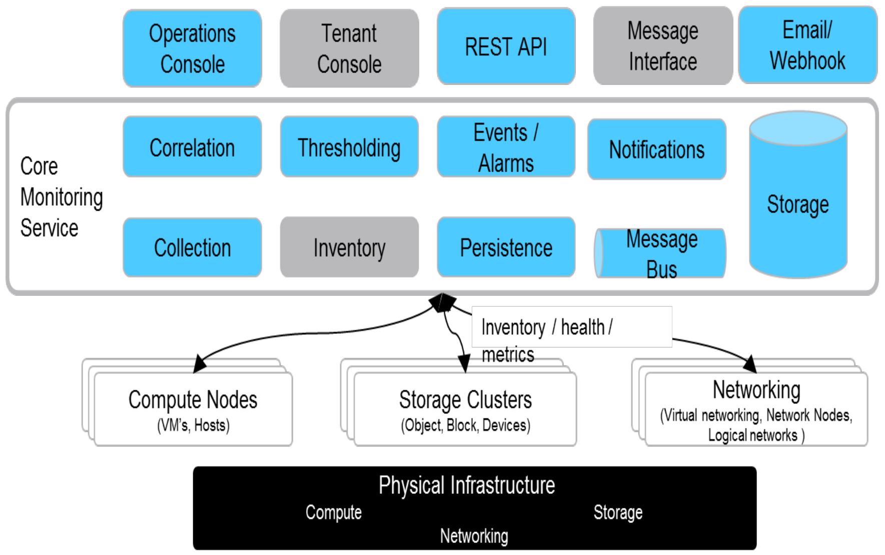
- Higher SLAs/ Increased reliability
- Lower MTTR w/faster troubleshooting
- Dynamic alarm management
- Compliance reporting
- Cloud scale: 100's today, 1000's tomorrow



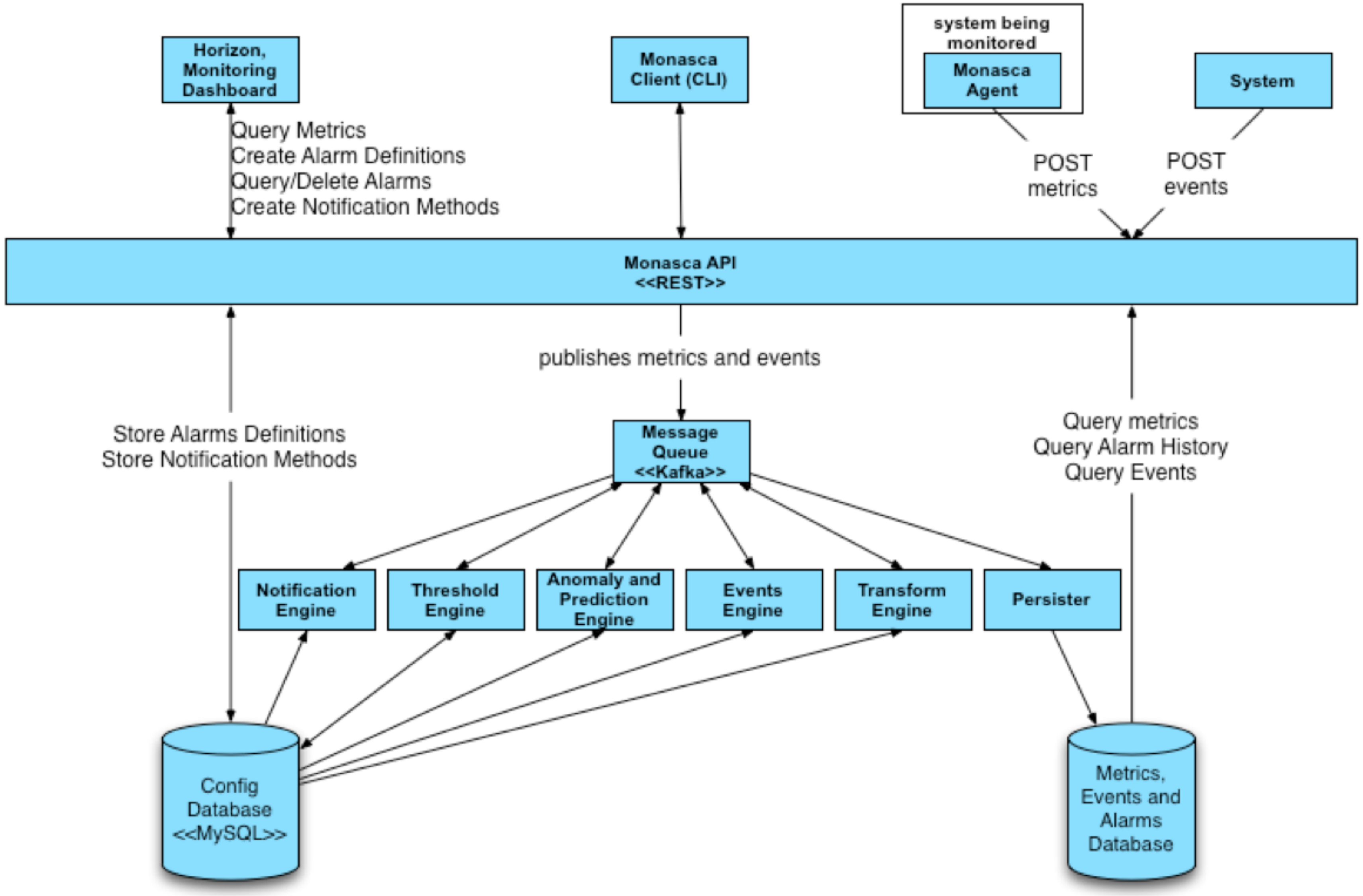
All Alarms

Alarms

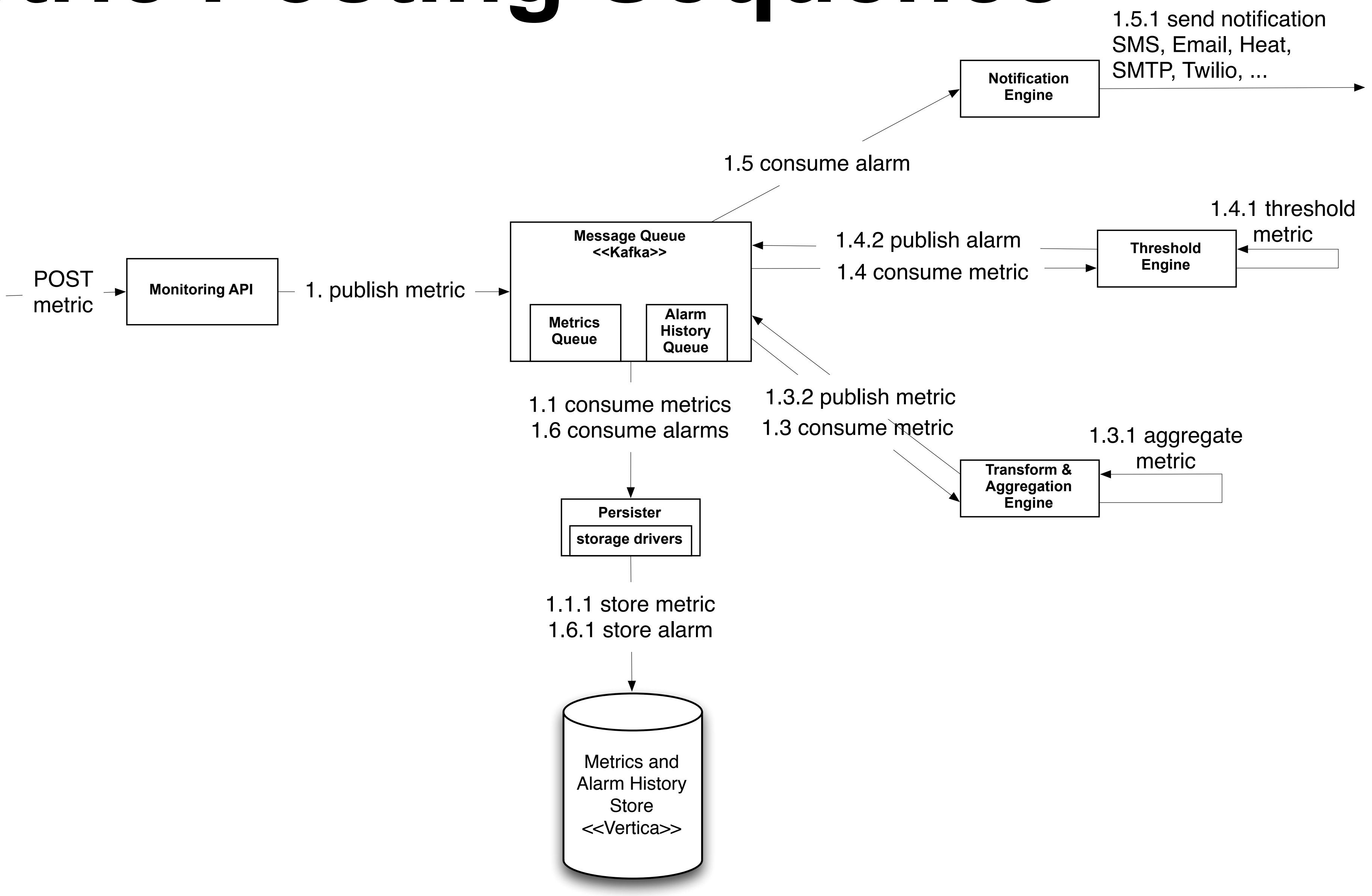
Status	Metric Name	Metric Dimensions	Definition	Actions
✓	http_status	url=http://localhost:8081/healthcheck,hostname=mini-mon,service=monitoring	API Status	Graph Metric
⛔	http_status	url=http://localhost:8774/v2.0,hostname=devstack,service=nova_api	API Status	Graph Metric
✓	http_status	url=http://localhost:8776/v2.0,hostname=devstack,service=cinder_api	API Status	Graph Metric
✓	http_status	url=http://localhost:8080/healthcheck,hostname=devstack,service=swift_api	API Status	Graph Metric
✓	http_status	url=http://localhost:9292,hostname=devstack,service=glance_api	API Status	Graph Metric
✓	disk_space_utilization_perc	device=v-csr-2,hostname=devstack,service=monitoring	Disk Space	Graph Metric
✓	disk_space_utilization_perc	device=udev,hostname=devstack,service=monitoring	Disk Space	Graph Metric



Architecture

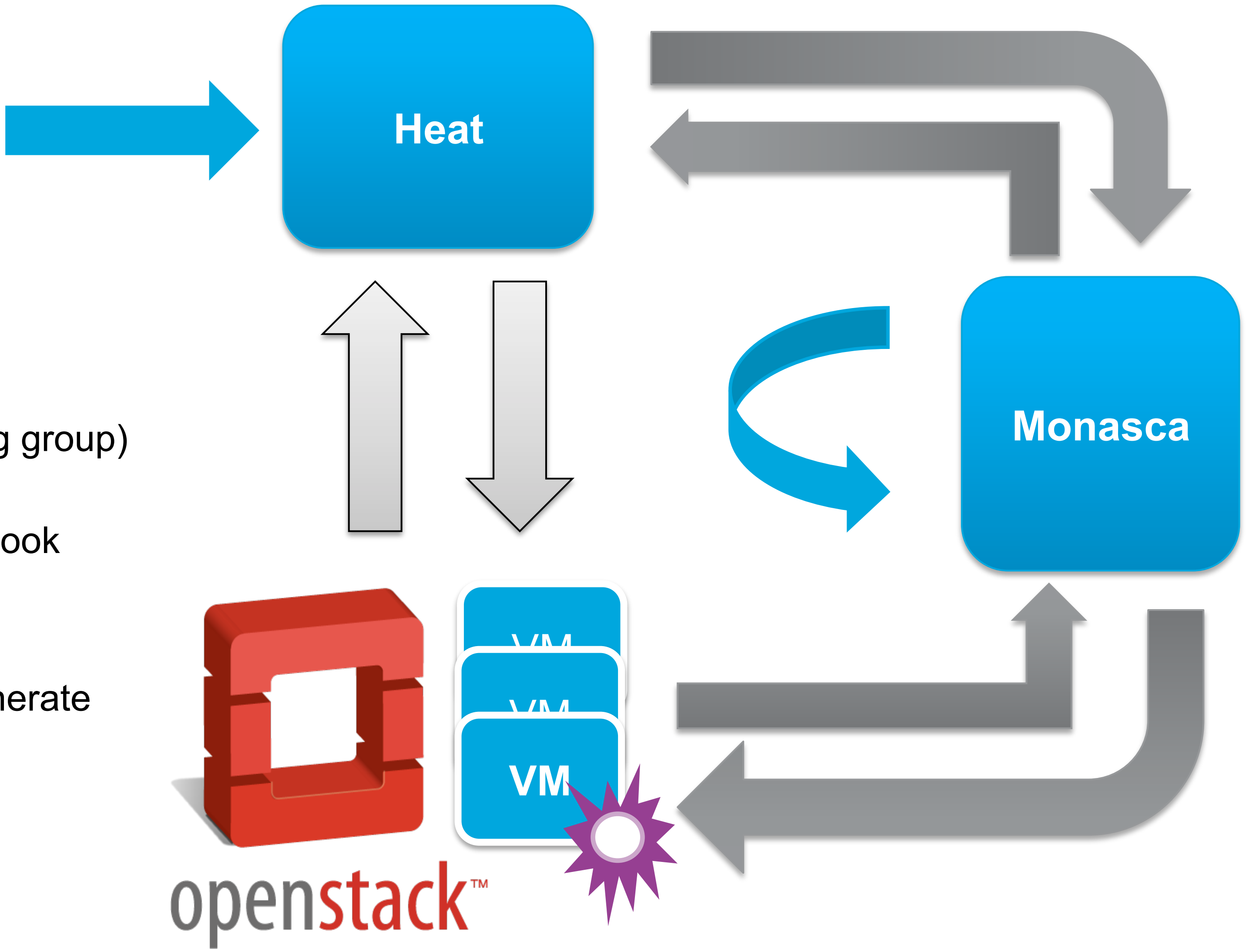


Metric Posting Sequence



AutoScale Demo

```
heat_template_version: 2013-05-23
resources:
  group:
    type: OS::Heat::AutoScalingGroup
  scaleup_policy:
    type: OS::Heat::ScalingPolicy
  notification:
    type: OS::Monasca::Notification
  cpu_alarm_high:
    type: OS::Monasca::AlarmDefinition
```



1. Heat create-stack auto-scale.yaml stack-1
2. Create desired nova instances (Autoscaling group) and auto-scaling for stack-1 in heat
3. Create monasca alarm definition and webhook notification
4. Monasca start to monitor nova instances
5. Instance reaches thersold and monasa generate alarm
6. Monasca calls heat webhook
7. Heat increase the instances count by 1

5-7 runs for ever ! (auto-scale)

Thank you!

Q&A

https://wiki.openstack.org/wiki/Monasca
https://launchpad.net/monasca

Core code
https://github.com/stackforge?query=monasca

Ancillary code
https://github.com/hpcloud-mon

Meetings Tuesdays 10 AM CST
https://wiki.openstack.org/wiki/Meetings/Monasca

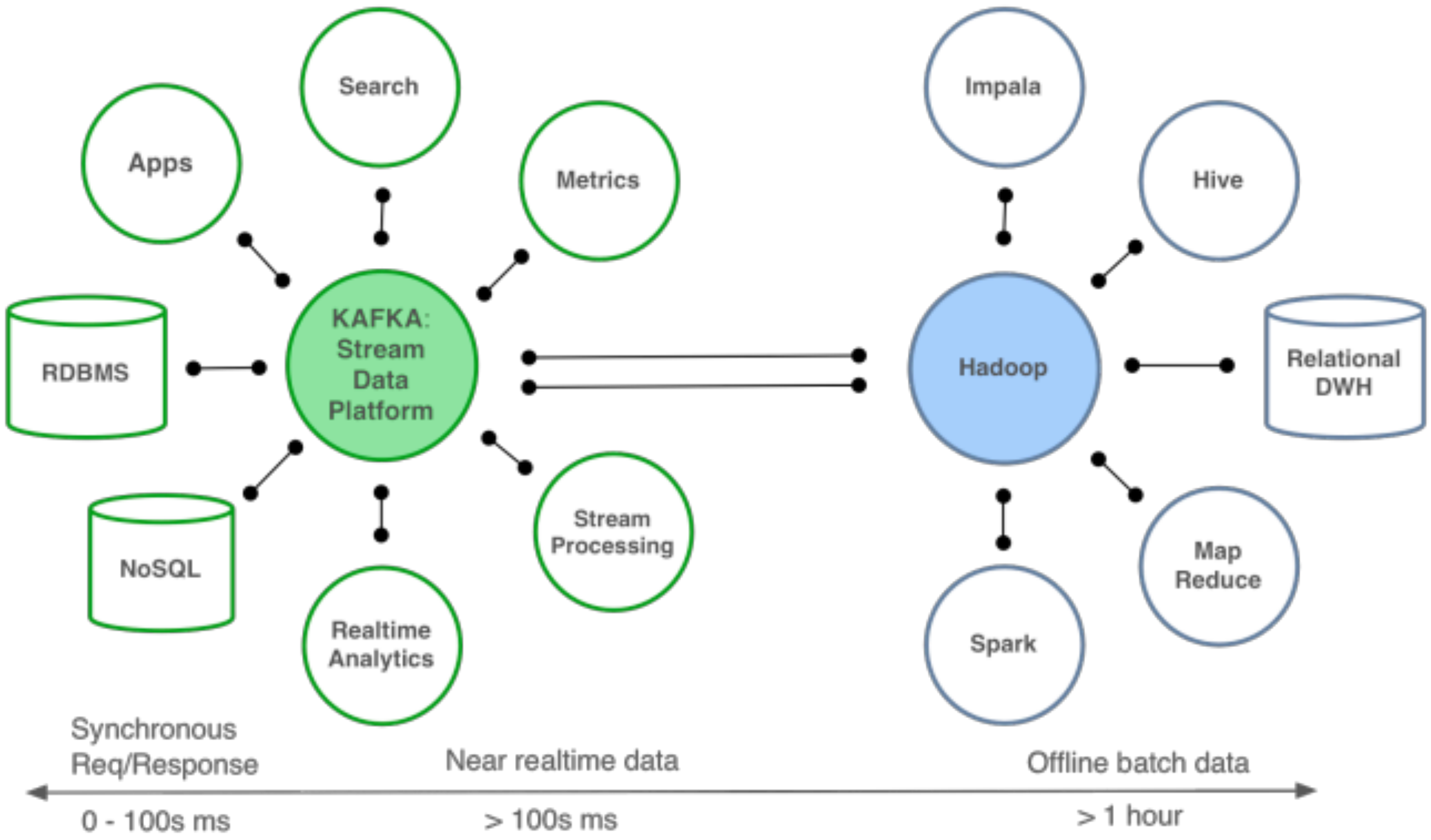
IRC #openstack-monasca on freenode.net

monasca-*.readthedocs.org

Backup

Stream Data Platform

http://www.confluent.io/blog/stream-data-platform-1/



RabbitMQ Issues & Limitations

- Performance:
  - RabbitMQ: 10K-20K messages/sec
  - Kafka: >100K messages/sec.
- Durability:
  - Performance of RabbitMQ with durable messages is very poor.
  - Kafka: Durable messages are always on.
- HA:
  - RabbitMQ does not cope seamlessly with network partitions and we’ve seen numerous failures.
  - Kafka: HA designed in based on a variant of PAXOS family of algorithms and handles network partitions based on consensus.
- Scalability:
  - Unable to scale RabbitMQ > 20K message/sec.
  - Easy to scale Kafka.
- RabbitMQ has been the biggest cause of failures and performance problems in a cloud at scale with a monitoring solution.