Efficient Monitoring and Root Cause Analysis in Complex Systems

Witek Bedyk

Agenda

- Benefits of robust monitoring
- Measurements vs. Alarms
- Importance of Alarms Correlation
- Effective Alerting
- Self-healing

Why is Monitoring useful?

- Improve system / application uptime
- Reduce administration burden
- Resource optimization
- Prevent bottlenecks
- Make use of collected data (e.g. billing)

Why is Monitoring useful?

- Improve system / application uptime
- Reduce administration burden
- Resource optimization
- Prevent bottlenecks
- Make use of collected data (e.g. billing)

Use Case

Customer escalation:

“We have cloud outage! Keystone is flapping up and down continuously and many requests get 503 service unavailable error.”

Healthcheck

Simple HTTP endpoint up or down checks on services.

http_status	[0, 1]
http_response_time	

Metrics

- *Metrics measure and report on quantifiable data from your system*

- cpu, memory, network, filesystem, disk IO
- Services
  - MySQL, RabbitMQ, Apache, MemcacheD, etc.
- LibVirt, Open vSwitch
- Applications:
  - StatsD, Prometheus
- Custom checks

Dimensions

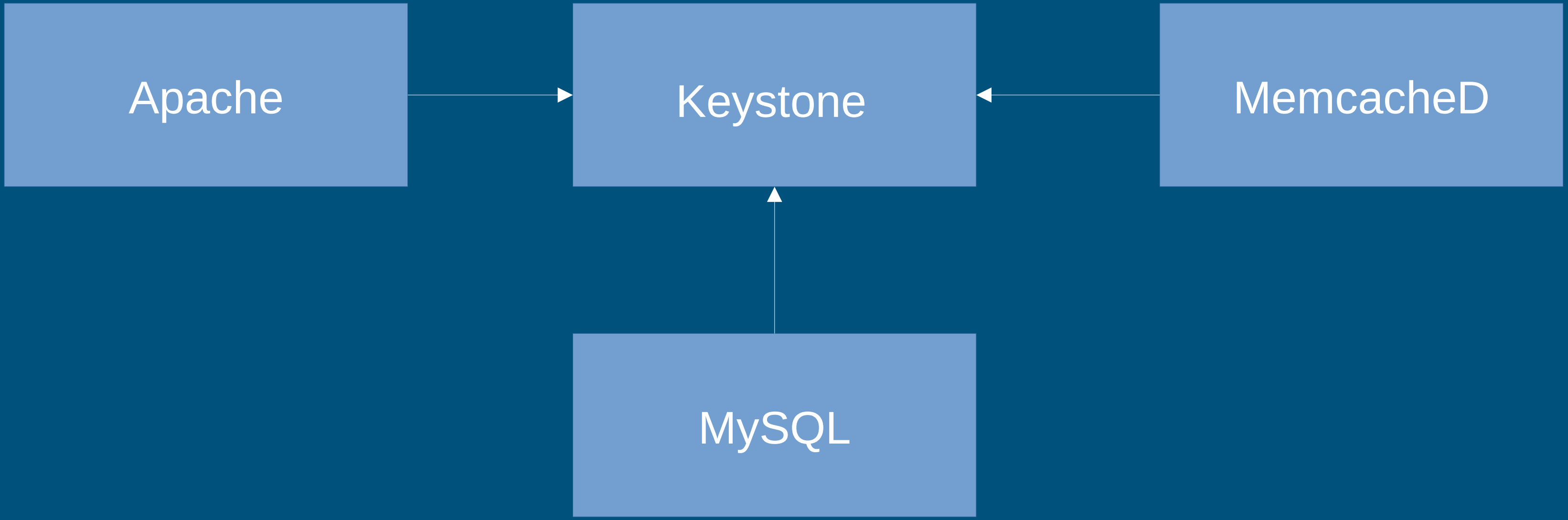
- *Dimensions are a dictionary of key, value pairs used to describe metrics.*

- hostname
- service
- component
- url
- device

Transaction-level vs. System-level metrics

- Transaction-level: end user perspective
  - Is Horizon working correctly?
- System-level: administrator perspective
  - Reveals failures of service components

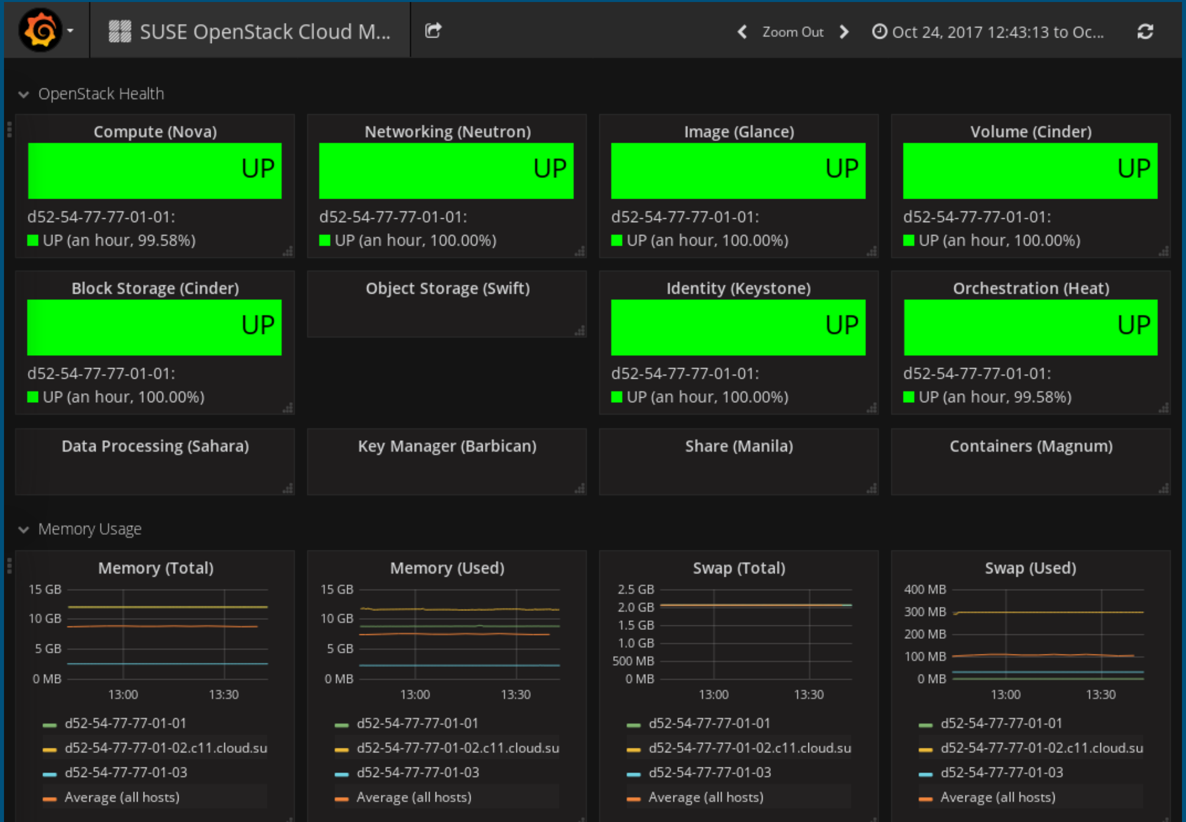
Dependencies



Gathered metrics

http_status
http_response_time
apache.net.hits
apache.performance.idle_worker_count
mysql.performance.open_files
mysql.net.connections
memcache.curr_connections
memcache.get_misses_rate
process.cpu_perc
process.open_file_descriptors

Dashboards



OpenStack Health

Compute (Nova)	Networking (Neutron)	Image (Glance)	Volume (Cinder)
UP	UP	UP	UP
d52-54-77-77-01-01: 🟩 UP (an hour, 99.58%)	d52-54-77-77-01-01: 🟩 UP (an hour, 100.00%)	d52-54-77-77-01-01: 🟩 UP (an hour, 100.00%)	d52-54-77-77-01-01: 🟩 UP (an hour, 100.00%)

Block Storage (Cinder)	Object Storage (Swift)	Identity (Keystone)	Orchestration (Heat)
UP		UP	UP
d52-54-77-77-01-01: 🟩 UP (an hour, 100.00%)		d52-54-77-77-01-01: 🟩 UP (an hour, 100.00%)	d52-54-77-77-01-01: 🟩 UP (an hour, 99.58%)

Data Processing (Sahara)	Key Manager (Barbican)	Share (Manila)	Containers (Magnum)

Memory Usage

Memory (Total) — Memory (Used) — Swap (Total) — Swap (Used)

- d52-54-77-77-01-01
- d52-54-77-77-01-02.c11.cloud.su
- d52-54-77-77-01-03
- Average (all hosts)

Alarms

Status of the system or resource meets criteria indicating an action is required.

Alarm definitions

- *Alarm definitions are templates specifying how alarms should be created.*
- grouping
- http_status > 0, match_by: ["service", "component", "hostname", "url"]
- filtering
- avg(cpu.idle_perc{service=monitoring}) < 20

Use case (alarms)

MemcacheD number of connections is high on node A.
MemcacheD hit rate is low on node A.
Keystone API is down on node A.
Keystone API is down on node A. Keystone API is up on node A.
Keystone API is down on node A. Keystone API is up on node A.
Keystone API is down on node A. Keystone API is up on node A.

Alarms correlation

- *“80% of the mean time to repair is wasted on trying to locate the issue”* Gartner

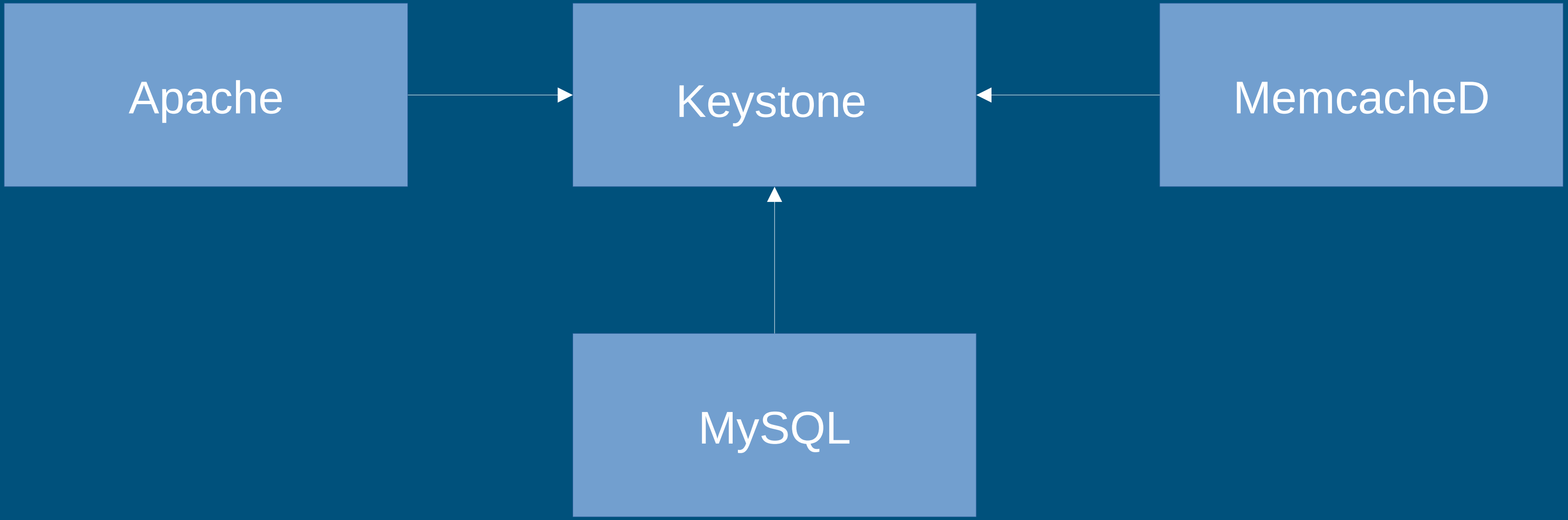
- Remove noise from the environment
- Alerts should be:
  - meaningful
  - actionable
  - indicate the point of failure

Vitrage

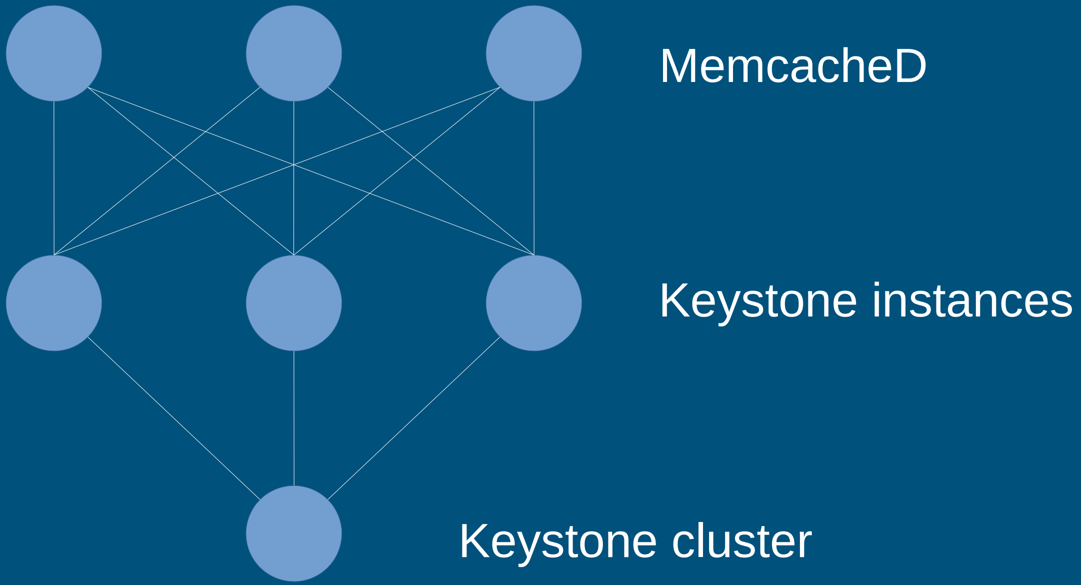
- *OpenStack Root Cause Analysis service*

- organize alarms
  - define relationships between alarms
  - represent as an entity graph
- analyze
  - represent system health
- find root cause
  - graphical visualization

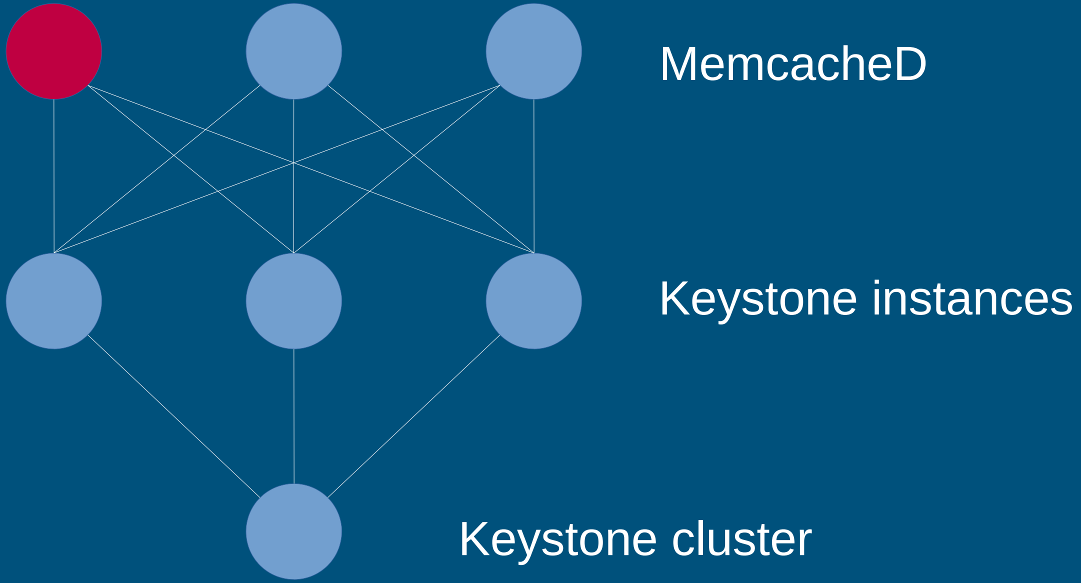
Dependencies



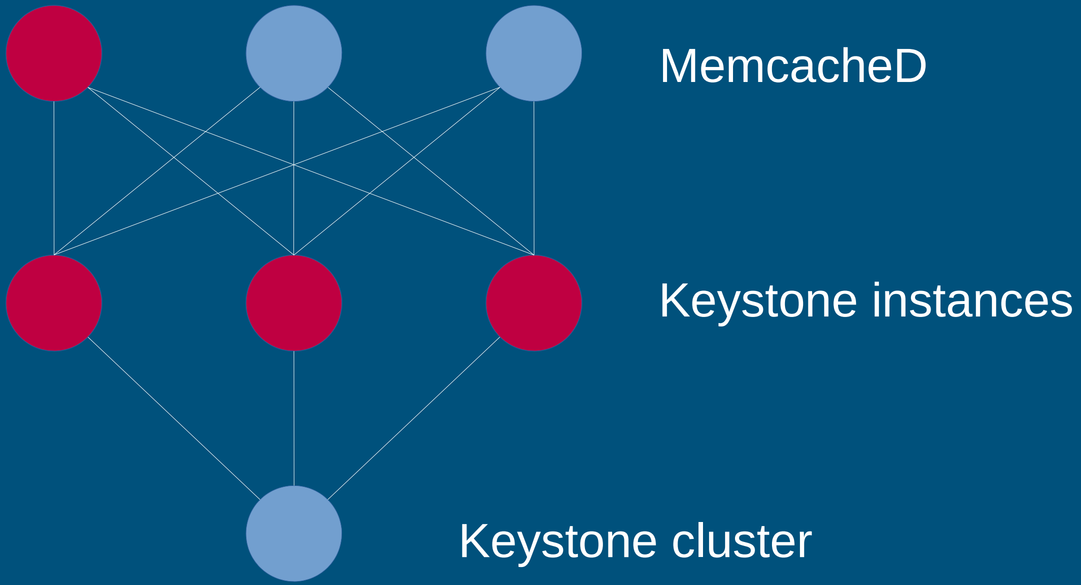
Dependencies



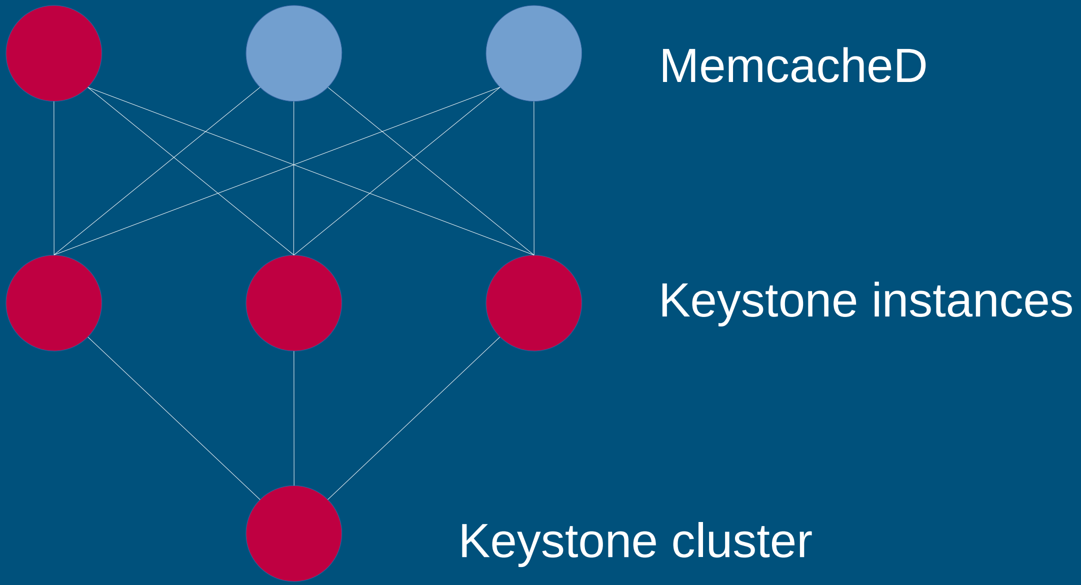
Dependencies



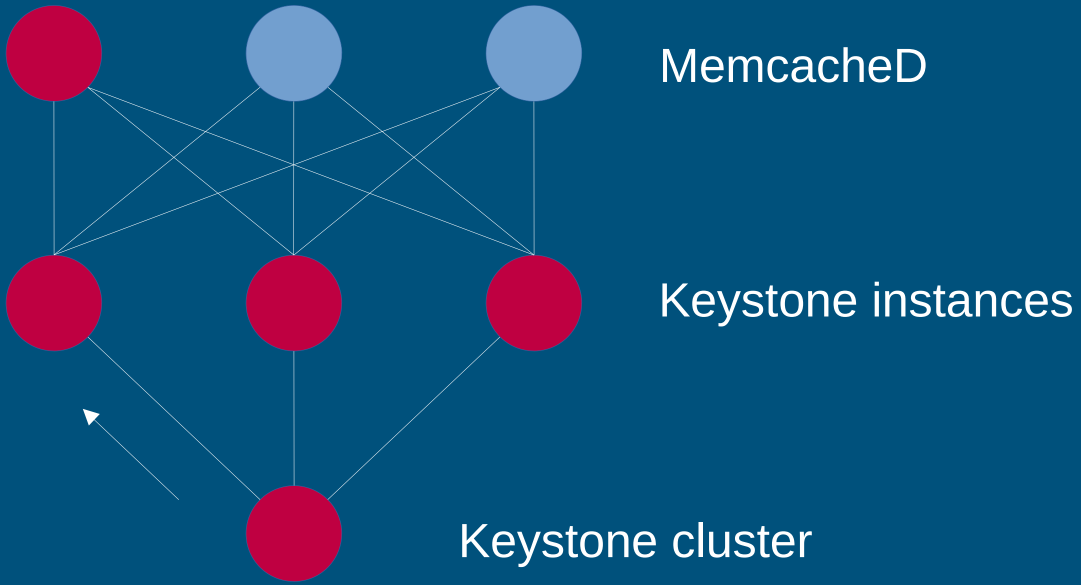
Dependencies



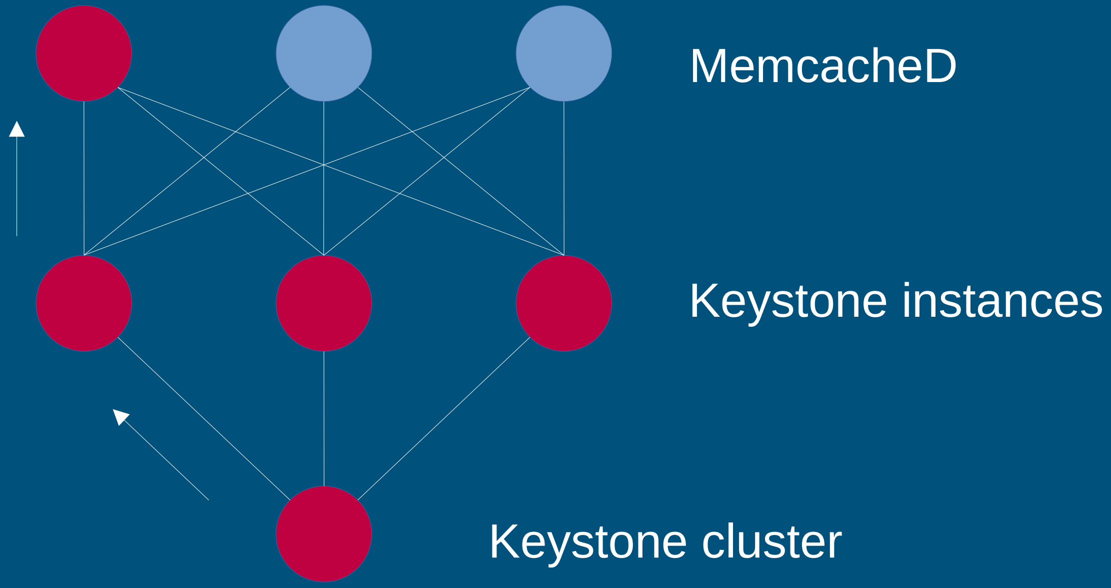
Dependencies



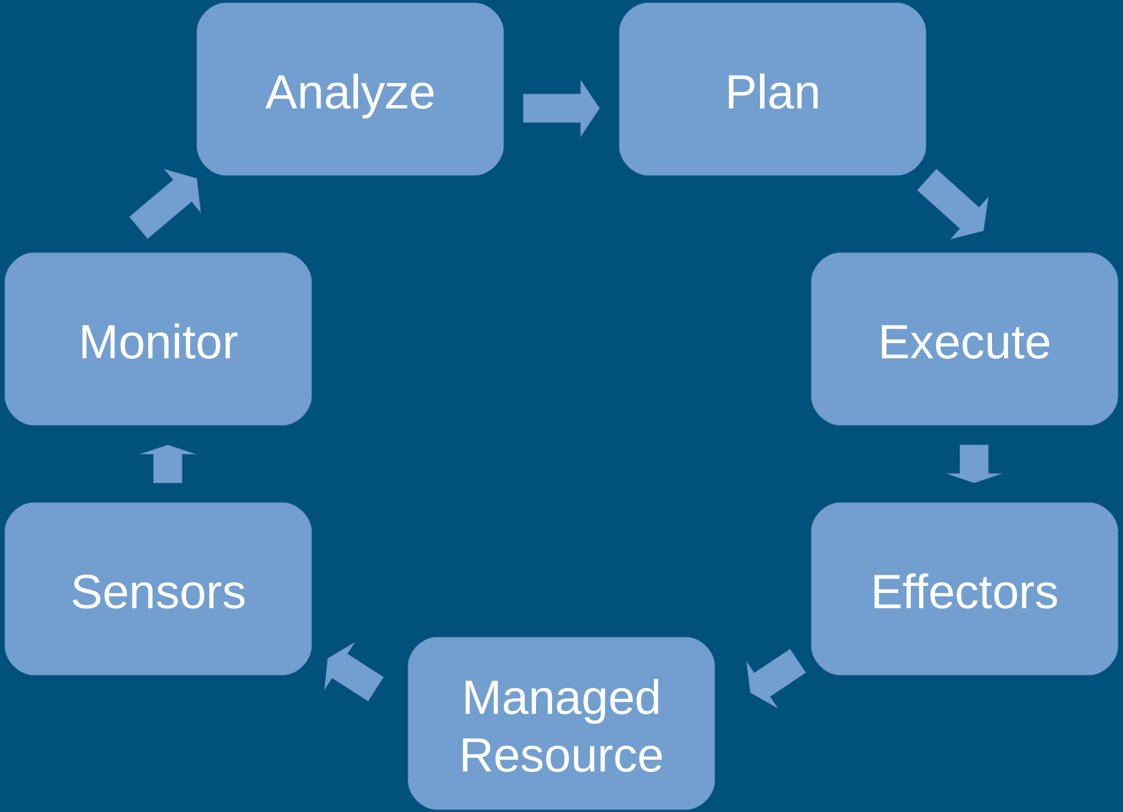
Dependencies



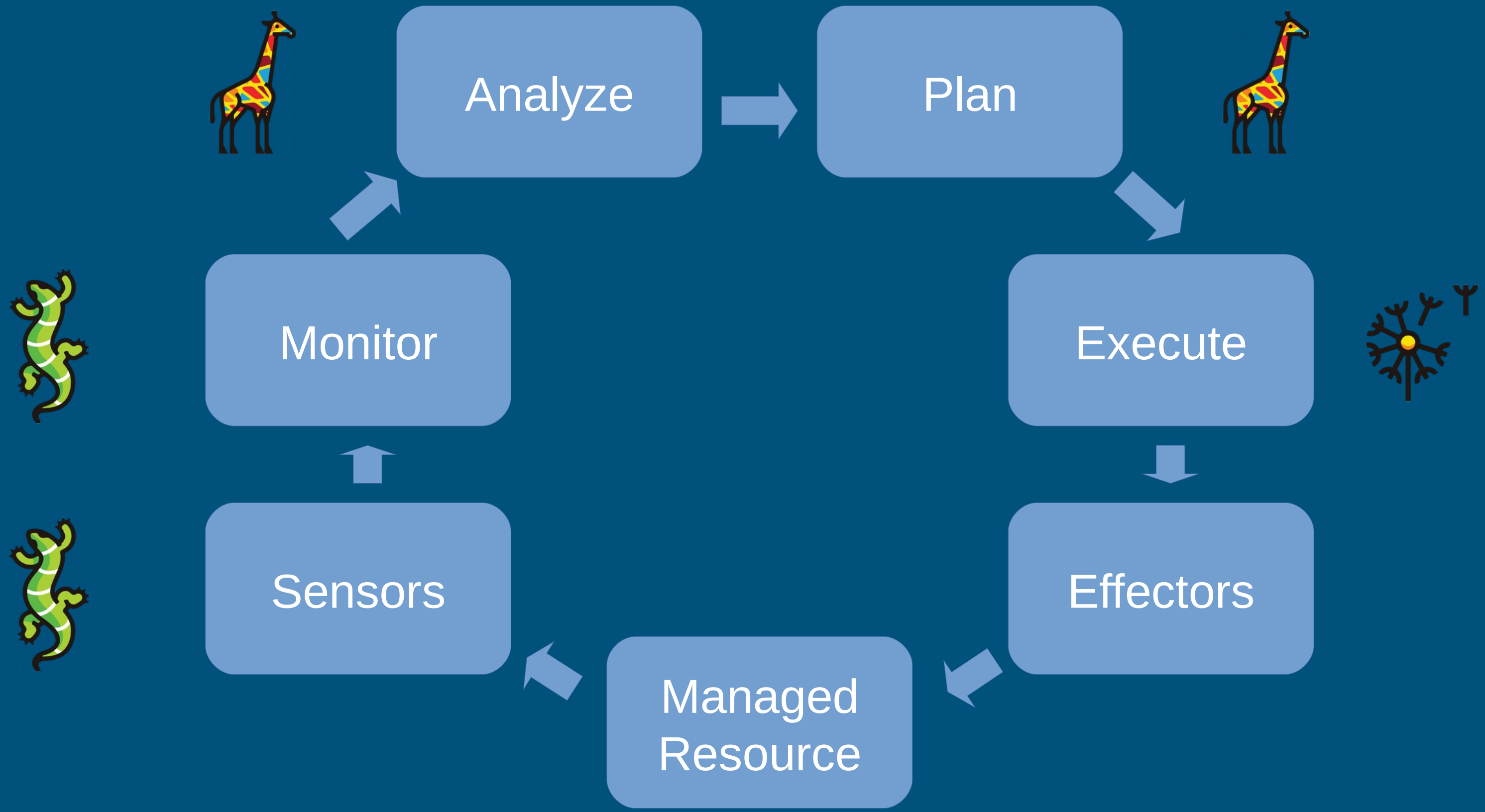
Dependencies



Monitor Analyze Plan Execute (MAPE)



Monitor Analyze Plan Execute (MAPE)

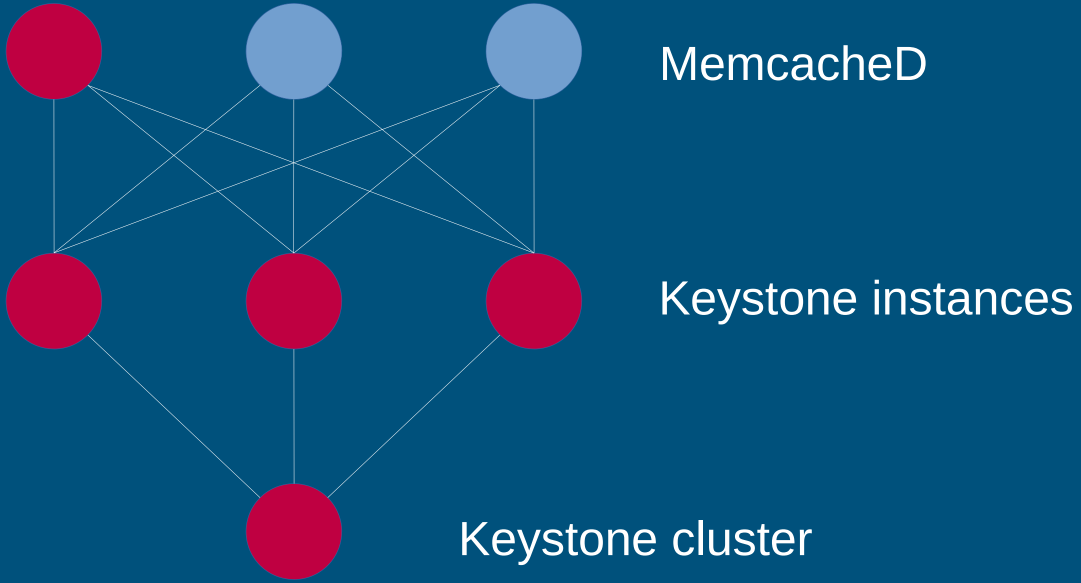


Vitrage Templates

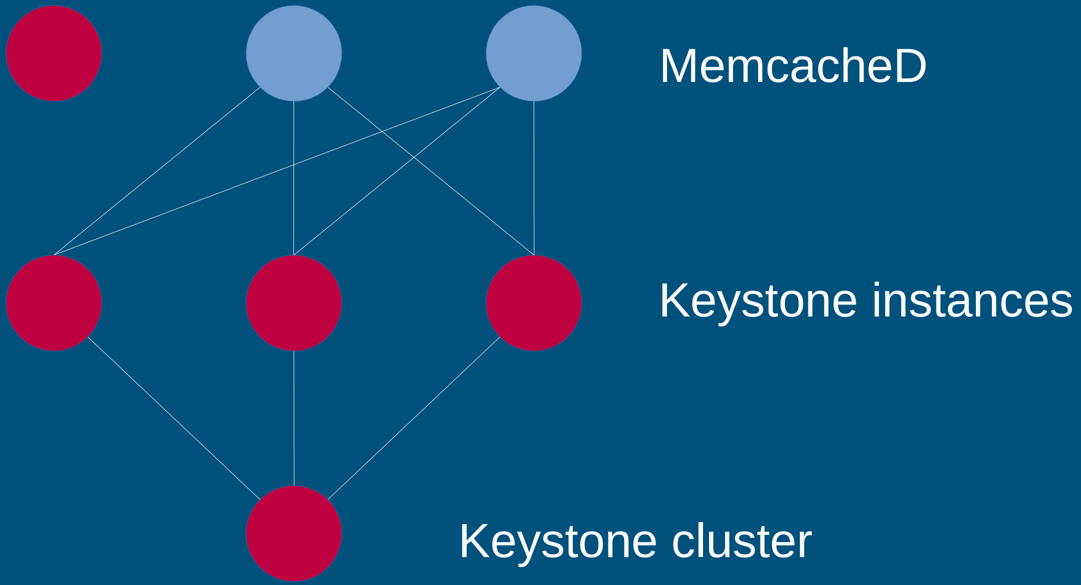
- *Vitrage Templates are used to express Condition → Action scenarios.*

- if <condition> then raise deduced alarm
- if <condition> then set deduced state
- if <condition> then add causal relationship (used for RCA capability)
- if <condition> then execute Mistral workflow

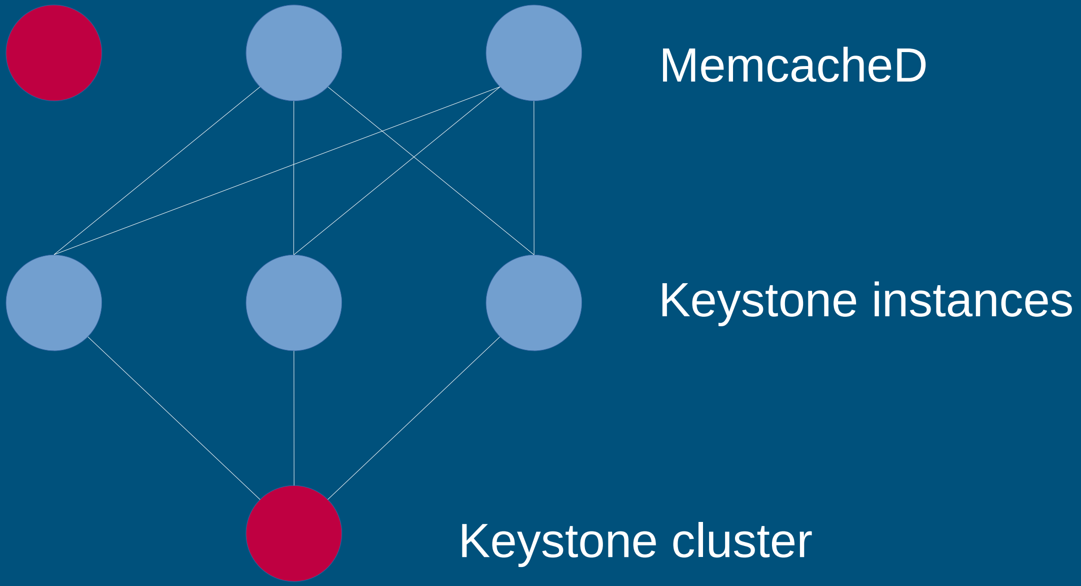
Self-healing



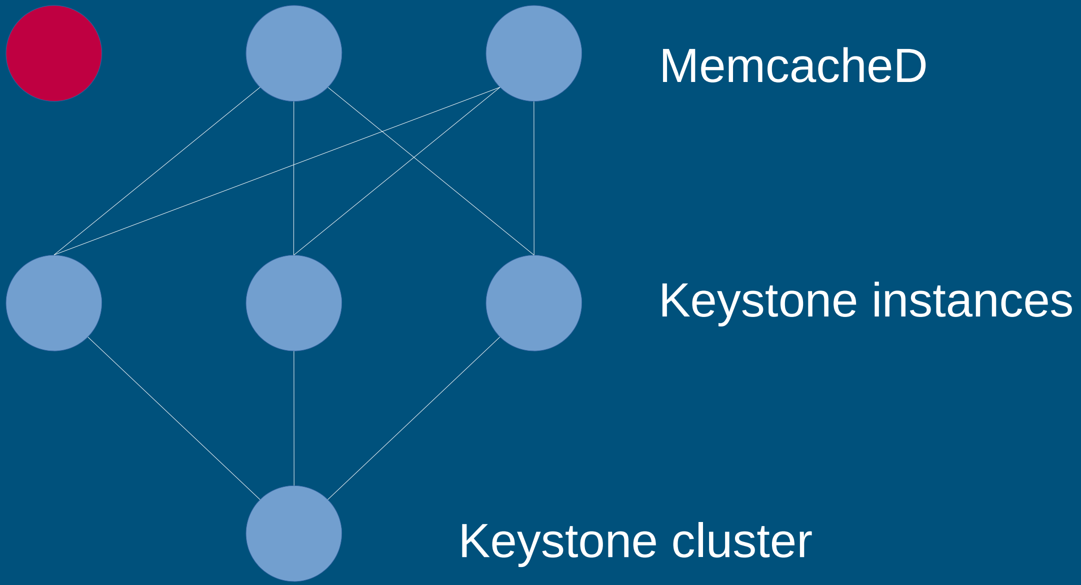
Self-healing



Self-healing



Self-healing



OpenStack Healthcheck APIs

- more detailed checks would be useful for most OpenStack services
- common middleware should get implemented in Oslo
- existing old effort:
    - https://storyboard.openstack.org/#!/story/2001439
    - https://review.opendev.org/617924

Summary

- Robust monitoring is essential
- Measurements vs. Alarms
- Importance of Alarms Correlation
- Self-healing

Thank You
谢谢

Questions and Answers