OPENSHIFT
by Red Hat

Scaling Up Logging and Metrics

Ricardo Lourenço < rlourenc@redhat.com >
Elvir Kuric < ekuric@redhat.com >

Agenda

- Performance and Scale
- OCP Logging stack
  - Overview
  - Test harness
  - Large clusters
  - Scaling Up
  -
- OCP Metrics stack
  - Overview

- Tools used
- Sample results

Performance and Scale

We test individual components for limits. Then we scale them up.

We focus on OpenShift’s control plane.

Then we target any underlying systems.

Components: ElasticSearch, Kibana, Cassandra, FluentD, Heapster, Java, Ruby...

Logging stack overview

Users need a way to collect statistics about nodes in a cluster and send those logs to an external database.

- Elasticsearch is a distributed system by itself.
  - Collects logs from node services and all containers in the cluster.
  - Should be deployed with redundancy and persistent storage for scale and high availability.
  - Labels (region=infra) and Node Selectors.
- Fluentd
  - Gathers log entries from nodes, enriching them with metadata.
  - Feeds them into Elasticsearch.
  - Deployed using DaemonSets, Labels (logging-infra-fluentd).
- Kibana
  - presents a web UI for browsing and visualizing logs in Elasticsearch.

Large Clusters

MaxRequestsInFlight

```
servingInfo:
  bindAddress: 0.0.0.0:8443
  bindNetwork: tcp4
  certFile: master.server.crt
  clientCA: ca.crt
  keyFile: master.server.key
  maxRequestsInFlight: 500
  requestTimeoutSeconds: 3600
```

Partial list of consumers:

- FluentD pods talk to the apiserver to get metadata for the pods such as its namespace, pod name, host.
- openshift-node talks with the apiserver for command and control.
- SDN

```
$ kube-apiserver --help | grep inflight

          --max-requests-inflight int       The maximum number of requests in flight at a given

time. When the server exceeds this, it rejects requests. Zero for no limit. (default 400)
```

ReplicationController

The ReplicationController makes sure that a pod or homogeneous set of pods are always up and available. If there are too many pods, it will kill some. If there are too few, the ReplicationController will start more.

- The Replication Controller is limited by the scheduler and scheduling cycle.
- The DaemonSet binds directly to the node, and bypasses the scheduler entirely.

Burst replicas on both daemonsets and rc is set to 500.

This essentially creates a step f(n) from 0-500 ~O(1) time.

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort:80
```

DaemonSet

A way to run a daemon on every node, or on a certain set of nodes in a cluster.
Use a a DS instead of a ReplicationController for pods that provide a machine-level function, such as machine monitoring or machine logging.

Use cases:

- building a sharded datastore.
- running a logger on every node.
- cluster-level applications with strong node ties, and Openshift / Kubernetes node services.

The DaemonSet is used to run Fluentd on every node and send the data to a service like ElasticSearch for analysis.

pkg/controller/replication/replication_controller.go

// performance requirements for kubernetes 1.0.
BurstReplicas = 500

// The number of times we retry updating a replication controller's status.

// It resumes normal action after observing the watch events for them.

pkg/controller/daemon/daemoncontroller.go

// performance requirements for kubernetes 1.0.
BurstReplicas = 500

// If sending a status update to API server fails, we retry a finite number of times.

// It resumes normal action after observing the watch events for them.

Limits

ES

Queue capacity

At 10250 pods (50 logger pods @ ~205 OCP nodes) Logging a 256 Byte string per second we've hit the ES bulk queue capacity.

Fluentd

Buffer chunk limit

The memory buffer plugin provides a fast buffer implementation. It uses memory to store buffer chunks. When Fluentd is shut down, buffered logs that can’t be written quickly are deleted.

Logging throughput test

- Features:
  - Full logging project deployment.
  - Automated testing with pbench integration.
    - Loading up a cluster with logging traffic at different rates.
    - Collect pbench data while containers are logging.
  - Other logging drivers are also supported through the -d parameter.
    - Json-file, fluentd, journald

Test description

Phase 1 - Before the loggers and pbench start

Delete all indexed data so that data from previous runs doesn’t add up.

```
$ oc exec $es_pod -- curl -XDELETE "$es_url/*"
```

Gets ES node stats through the API

```
$ oc exec $es_pod -- curl $es_url/_stats?pretty
```

Captures disk usage under /var/lib/origin/openshift.local.volumes/pods/

Phase 2 - Generate load

Start X logger pods across N OCP cluster nodes, logging pre-defined random string at a chosen logging rate.
Registers pbench tools across chosen cluster nodes: iostat mpstat pidstat proc-vmstat sar turbostat

Phase 3 - After the run finishes.

Optimize and flush indexes.

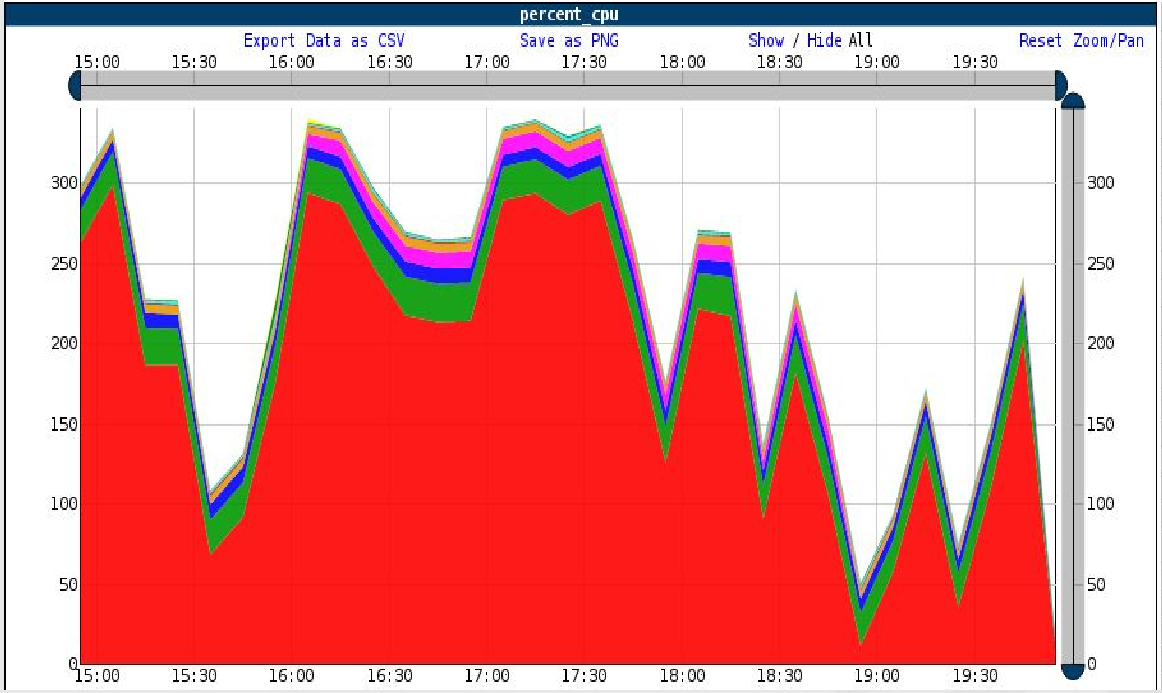
```
$ oc exec $es_pod -- curl -XPOST "$es_url/_refresh"
$ oc exec $es_pod -- curl -XPOST "$es_url/_optimize?only_expunge_deletes=true&flush=true"
```

Like in phase 1, gets ES, OC stats, logs and disk usage for your pbench --remote nodes.

Sample output

10n-no_limits-2048ll-1000wait-300s

pidstat - cpu_usage



percent_cpu

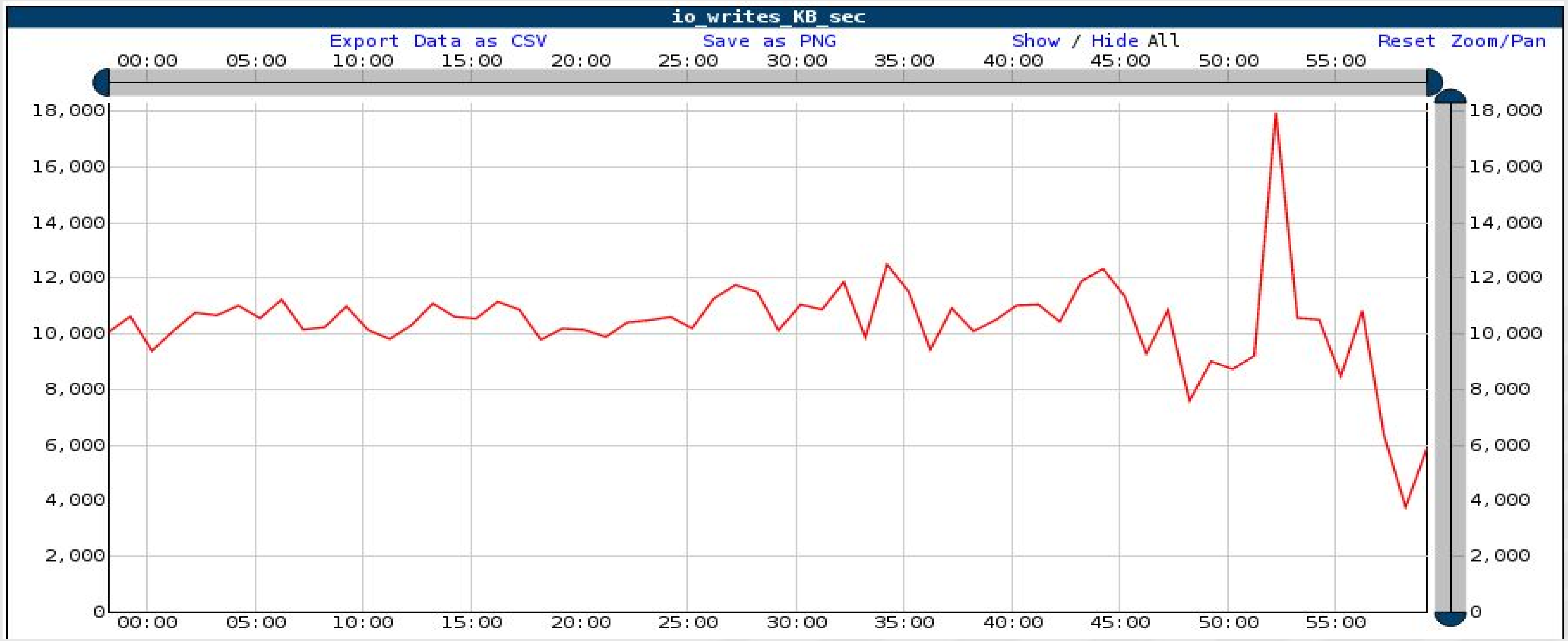
Threshold: 1 Apply Max Y Apply Y Average

Dataset Name Filter: Show Datasets Hide Datasets

+ View Port Controls

Data Sets	Value	Average	Median	Samples
4718-/bin/java_-Xms256m_-Xmx1g_-Djava.awt.hea		181.32	202.30	31
442-/usr/lib/systemd/systemd-journald		21.47	21.90	31
603-/usr/sbin/rsyslogd_-n		8.63	9.10	31
42271-/usr/bin/ruby_/opt/app-root/src/bin/flue		5.28	7.50	31
41891-/root/slstress_-tslstress-2048ll-1000w_-		4.72	4.80	31
2912-journalctl_-k_-f		1.34	1.30	31
2410-/usr/bin/openshift_start_node_--config=/		1.31	1.10	31
2652-/usr/bin/docker-current_daemon_--authori		0.94	0.70	31
2577-/usr/libexec/docker/rhel-push-plugin		0.21	0.20	31
41802-/usr/local/bin/pidstat_-l_-w_-u_-h_-d_-r		0.14	0.10	31
42255-/usr/bin/ruby_/opt/app-root/src/bin/flue		0.13	0.00	31
780-ovs-vswitchd_unix:/var/run/openvswitch/d		0.11	0.10	31
41860-/usr/local/bin/sar_-A_-o_/var/lib/pbench		0.06	0.10	31
41864-sadc_10_-z_-S_XALL_/var/lib/pbench-agent		0.03	0.00	31
1-/usr/lib/systemd/systemd_--switched-root		0.03	0.00	31
37940-/usr/sbin/dnsmasq_-k		0.03	0.00	31
41812-/usr/bin/perf_record_-a_--freq=100		0.02	0.00	31
42240-sh_run.sh		0.02	0.00	31
42112-/pod		0.02	0.00	31

30wppn - 205nodes - 1hour test.



Recommendations

- It is highly recommended to pre-pull the fluentd images (> 200 nodes).
  - ImagePullPolicy: IfNotPresent should be set in the deployer.yaml file.
- After deploying the region=infra pods, labeling your nodes should be done in steps/batches of 20 nodes at a time while keeping an eye open for any "MatchNodeSelector|CrashLoopBackOff|ImagePullFailed|Error" status.
- Use labels and selectors to differentiate infrastructure and application nodes
- Use large persistent volumes (PVs) for infrastructure nodes running logging (Elasticsearch) and metrics (Cassandra)

OpenShift Metrics

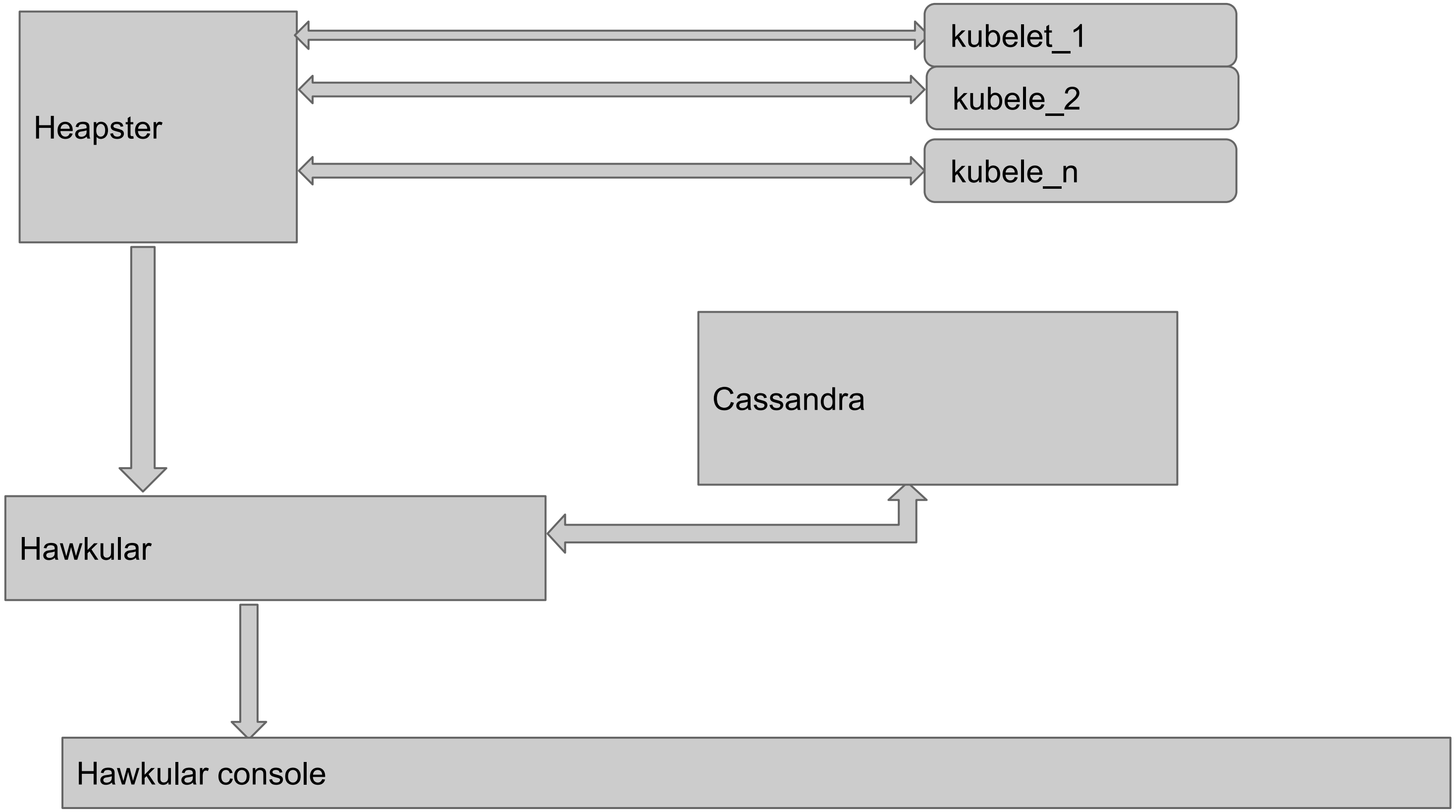
OCP Metrics stack - Overview

- Cassandra
  - Serves as datastore - all metrics data are written to Cassandra db

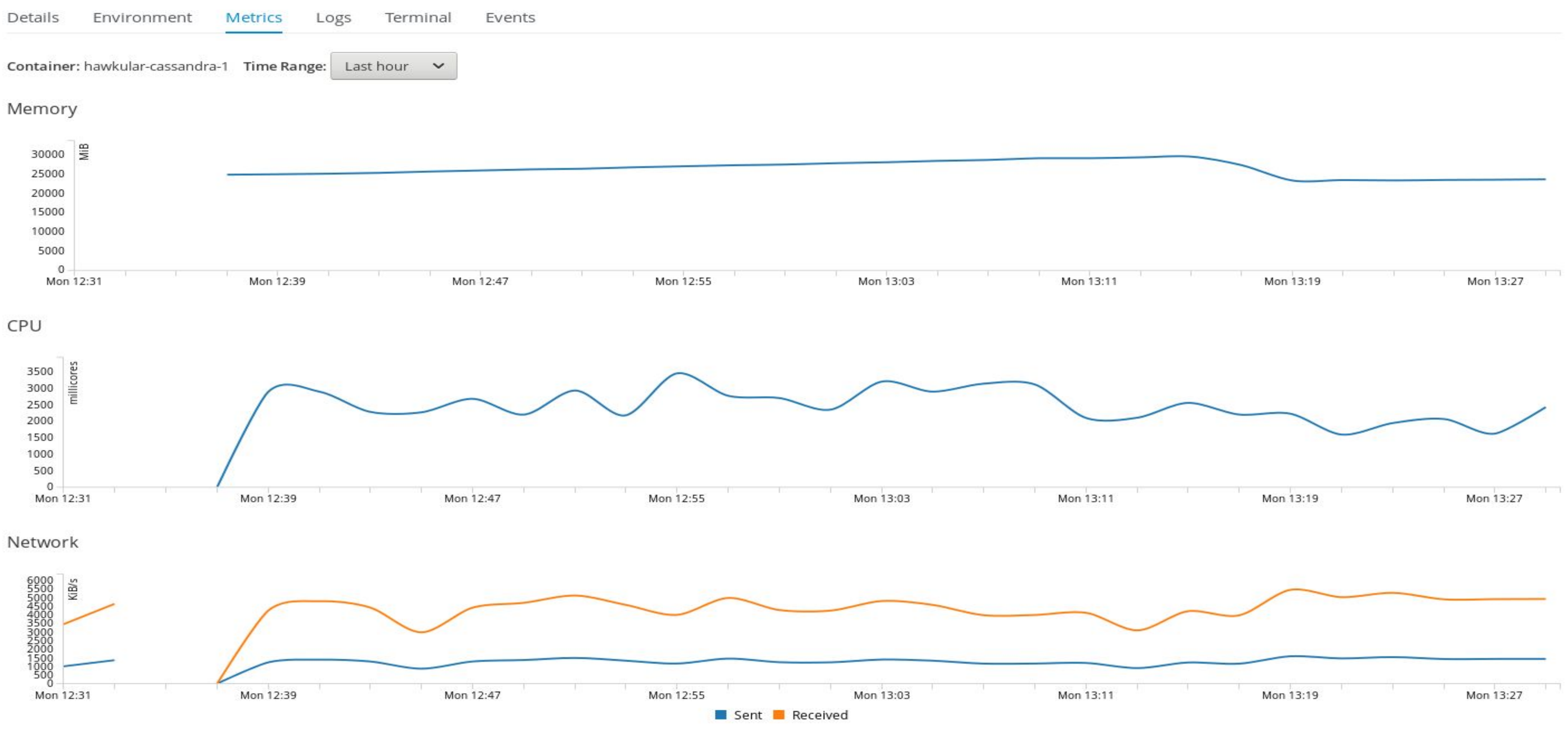
- Hawkular metrics storage engine

- Heapster
  - Collects metrics data from pods

Openshift metrics high level overview



OpenShift Metrics console



Details   Environment   Metrics   Logs   Terminal   Events

Container: hawkular-cassandra-1   **Time Range:** Last hour

OCP Metrics configuration

- All metrics components are are delivered as images and starting deploying metrics is easy as starting metrics pods
- metrics.yaml (from upstream : https://github.com/openshift/origin-metrics)

cassandra pod

- Pods (= cassandra nodes) form Cassandra cluster
- USE_PERSISTANT_STORAGE=True
  - PV will be mounted in cassandra pod /cassandra_data
- Persistent storage ensures data persistence
- Possible to scale out to more Cassandra pods (=nodes)

Cassandra pod

- USE_PERSISTANT_STORAGE=false
- No data persistence - if pod dies data will be lost
- Watch /var/lib/origin/ if persistent storage not used - can fill it over time

Heapster metrics component

- runs as Pod
- Heapster gathers metrics data from OCP cluster
- It gets metrics for every pod - across all namespaces
- Send these data to Hawkular metrics via API

OPENSHIFT
by Red Hat

Hawkular

"It provides means of creating, accessing and managing historically stored metrics via an easy to use json based REST interface."

Per : https://github.com/openshift/origin-metrics

OCP metrics setup

- Manual setup
  - Enter commands on master
  - Edit /etc/master/master-config.yaml
  - Restart master
- Using advanced installer
  - Set up all in ansible playbook

https://docs.openshift.com/enterprise/3.1/install_config/cluster_metrics.html

Scaling metrics pods

- We can add more
  - Cassandra pods
  - Hawkular pods
- Only one heapster pod per cluster

OpenShift metrics

- Use persistent storage for data - always
  - OpenShift metrics supports dynamic storage provisioning
  - DYNAMICALLY_PROVISION_STORAGE=true
- Run metrics pods on ‘infra’ nodes

OpenShift metrics configuration tips

- Watch system usage where metrics pods are running
- One set of metrics pods can handle ~10k pods - just guidelines

OpenShift metrics configuration tips

- For monitoring more pods, necessary to scale out metrics pods
- Cassandra is not best candidate for network storage, eg. NFS
  - http://docs.datastax.com/en/archived/cassandra/1.2/cassandra/architecture/architecturePlanningAntiPatterns_c.html

OPENSHIFT
by Red Hat

OpenShift metrics scale numbers

- It was successfully ran on clusters up to 1000 nodes (210 and 981 nodes cluster)
- Cassandra storage requirements are
  - 2.5 GB / day and monitored 1000 pods
  - 17.5 GB / day and 10k pods
- METRICS_DURATION=7(days)
- METRICS_RESOLUTION=15(s)

OPENSHIFT by Red Hat

OpenShift Metrics scaling issues

- Heapster is not catching up with massive number of pods monitored > 12k pods in OCP cluster
- Up to 12k pods it works good
- BZ 1388815
- **Monitoring Microservices on OpenShift with HOSA**

Hawkular project :

Thing to follow : HOSA

- Monitoring Microservices on OpenShift with HOSA
  - Hawkular OpenShift Agent (HOSA)

Blog post from hawkular team :

http://www.hawkular.org/blog/2017/01/17/obst-hosa.html?_lrsc=4c8f78df-8f03-4630-bd54-128a24998599&sc_cid=701600000011zELAAY

Thank you!
Q & A ?

GitHub

- https://github.com/redhat-performance
- http://github.com/openshift/svt

Ricardo Lourenço rlourenc@redhat.com; Elvir Kuric ekuric@redhat.com

OPENSHIFT by Red Hat