W. Caban, *Architecting and Operating OpenShift Clusters*
https://doi.org/10.1007/978-1-4842-4985-7_9

9. Day-2 Operations

William Caban [1]

(1) Columbia, MD, USA

As seen in the previous chapter, OpenShift provides features or capabilities to enhance developer experience like the CI/CD Pipelines covered in Chapter

8 and the self-service Templates in Chapter 7 . The day-to-day work of developers may leave a high number of objects behind. In very active development environments, the garbage collection processes might need tuning. For example, when executing CI/CD Pipelines or building Containers using features like source to image (s2i), there might be intermediate Containers or Image layers that get created and left behind, consuming the Node ephemeral storage and increasing the size of the etcd database. To work with this, once the OpenShift cluster is in operation, there are certain tasks required for the proper maintenance, operations, and fine-tuning of the cluster. This chapter covers some of these common tasks.

Managing Leftover Objects

During the normal operation and utilization of the cluster and cluster services, objects created in OpenShift can accumulate. Maintaining all previous versions of all the objects may end up consuming significant amount of storage which may have an impact on the performance of elements of the platform. For example:

- High storage consumption of the *etcd data store* may add additional pressure on *etcd* response time which leads to higher latency per request.

> **NOTE** The upstream OSS *etcd* project provides the *benchmark* [1] tool that can be used to measure etcd performance.

- Depending on the storage backend used by the internal Container Registry, high storage consumption of the backend storage may yield to slower upload (push) time for new images being build or onboard into the platform.

> **TIP** Using object storage as the storage backend for the internal Container Registry regularly is the most resilient and cost-effective storage backend for this job.

- High storage utilization of `/var/lib/containers` which is used by the Container Runtime to cached Container Images and for the Container

ephemeral storage will have an impact on the ability to instantiate new Containers in the node or the ability to download new Images.

> **TIP** Use a dedicated disk or partition to map to the `/var/lib/containers` directory to avoid saturating the *root* disk of the *Node*.

The high storage consumptions can be the result of normal cluster operations by users of the platform. This is particularly relevant when using objects like *Deployments*, *Builds*, manipulating *Images* (i.e., tagging and keeping multiple releases, etc.), groups, *CronJobs,* and others.

The *OpenShift* client CLI provides a mechanism for cluster administrator to prune [2] older versions of some of this resource (see Figure 9-1).

```
$ oc adm prune
Remove older versions of resources from the server

 The commands here allow administrators to manage the older versions of resources on the system by removing them.

Usage:
  oc adm prune [flags]

Available Commands:
  auth        Removes references to the specified roles, clusterroles, users, and groups.
  builds      Remove old completed and failed builds
  deployments Remove old completed and failed deployments
  groups      Remove old OpenShift groups referencing missing records on an external provider
  images      Remove unreferenced images

```

Figure 9-1 Removing older version of resources

The execution of the prune command will perform a *dry run* by default (see line #2 on Figure 9-2). During the run it identifies the resources of the particular time that will be removed (see line #4 on Figure 9-2) during the actual process.

```
1  $ oc adm prune images (1)
2  Dry run enabled - no modifications will be made. Add --confirm to remove images (2) (3)
3  Deleting istags openshift-sdn/node: v3.11
4  Deleting istags cicd-staging/podcicd-v4: v4
5  Deleting istags openshift-node/node: v3.11
6  Deleting istags cicd-staging/podcicd-v3: v3
7  Deleting istags cicd-staging/podcicd-v1: v1 (4)
8  Deleting istags cicd-staging/podcicd-v5: v5
9  Deleting istags cicd-prod/podcicd-v6: v6
10 Deleting istags cicd-prod/podcicd-v1: v1
11 Deleting istags cicd-staging/podcicd-v6: v6
12 Deleting istags cicd-prod/podcicd-v5: v5
13 Deleting istags cicd-staging/podcicd-v2: v2
14 Deleted 11 objects. (5)
```

Figure 9-2 Command to prune Images

The “confirm” flag must be appended to the *prune* command for the actual process to be executed (see line #2 on Figure 9-3). Additional flags are available to provide higher control and granularity of which objects should be removed or maintained (see lines #4 and #7 on Figure 9-3).

```
1  # To execute the actual prune operation the "confirm" flag must be appended
2  $ oc adm prune images --keep-tag-revisions=3 --keep-younger-than=2h --confirm
3
4  --keep-tag-revisions=3  Specify the number of image revisions for a tag in an image
5                          stream that will be preserved
6
7  --keep-younger-than=3h  Specify the minimum age of an image and its referrers for it
8                          to be considered a candidate for pruning.
9
```

Figure 9-3 Confirming the prune command

NOTE The optional flags for the `oc adm prune` commands are object specific. Refer to the CLI command help for details.

Garbage Collection

There are two types of garbage collection [3] performed by the *OpenShift Nodes*:

- **Container garbage collection**: Removes terminated containers. This is enabled by default and it is executed automatically.
- **Image garbage collection**: Removes Images no longer referenced by any running Pods. It relies on disk usage as reported by cAdvisor on the Node to choose which Images to remove from the Node.

When the garbage collection is executed, the oldest images get deleted first until the stopping threshold is met. Both of these garbage collection types are configurable by modifying the *Kubelet* argument settings at the *Node ConfigMap* (see Figure 9-4).

```
# oc edit cm -n openshift-node node-config-compute
apiVersion: v1
kind: ConfigMap
data:
  node-config.yaml: |
    apiVersion: v1
...
    kind: NodeConfig
    kubeletArguments:
      minimum-container-ttl-duration:        # Minimum age that a container is eligible for garbage collection
      - "10s"
      maximum-dead-containers-per-container: # Number of instances to retain per pod container
      - "2"
      maximum-dead-containers:               # Maximum number of total dead containers in the node
      - "240"
      image-gc-high-threshold:               # Percent of disk usage which triggers image garbage collection
      - "85"
      image-gc-low-threshold:                # Percent of disk usage to which image garbage collection attempts to free
      - "80"
      bootstrap-kubeconfig:
      - /etc/origin/node/bootstrap.kubeconfig
      cert-dir:
      - /etc/origin/node/certificates
      enable-controller-attach-detach:
      - 'true'
      feature-gates:
      - RotateKubeletClientCertificate=true,RotateKubeletServerCertificate=true
      node-labels:
      - node-role.kubernetes.io/compute=true
      pod-manifest-path:
      - /etc/origin/node/pods
      rotate-certificates:
      - 'true'
...
```

Figure 9-4 Garbage collection settings in the Node ConfigMap

Node Optimizations

There are multiple ways to optimize *Nodes* to deliver the performance required for the workloads and the experience required by an organization. The specific settings to modify to achieve certain optimization are tied to the specifications of the *Hosts* and the characteristics of the workload that will be running in those *Nodes*.

OpenShift provides many settings to tune the performance of the Platform. The following subtopics are some of the common settings available for cluster administrators to configure to achieve desired *Node* optimizations.

NODE RESOURCE ALLOCATION

OpenShift provides configuration [4] parameters to allocate per *Node* resources to maintain reliable scheduling of workloads to a Node while

minimizing overcommitting compute and memory resources. There are two types of resource allocations:

- **kube-reserved**: Allocation of resources reserved for Node components (i.e., kubelet, kube-proxy, Container Runtime, etc.). The default is None.

- **system-reserved**: Allocation of resources reserved for Host system components (i.e., sshd, NetworkManager, etc.). The default is None.

Both of these resource reservation types are configurable by modifying the *Kubelet* argument settings at the *Node ConfigMap* (see Figure 9-5).

```
# oc edit cm -n openshift-node node-config-compute
apiVersion: v1
kind: ConfigMap
data:
  node-config.yaml: |
    apiVersion: v1
...
    kind: NodeConfig
    kubeletArguments:
      kube-reserved:              # Resources reserved for node components.
      - "cpu=200m,memory=512Mi"
      system-reserved:            # Resources reserved for the remaining system components.
      - "cpu=200m,memory=512Mi"
...
```

Figure 9-5 Node resource reservation

SETTING MAX PODS PER NODE

OpenShift provides two *Kubelet* configuration setting to control the maximum number of Pods that can be scheduled into a Node:

- **pods-per-core**: Configures the maximum number of Pods the Node can run per core on the Node. When using this parameter, the maximum number of Pods allowed in the Node will be `<pods-per-core> x <number-of-cores-in-node>`

> **NOTE** To disable this limit, set *pods-per-core* to 0.

- **max-pods**: Configures a fixed number as the maximum number of Pods that can run on the Node. The default value is 250.

> **NOTE** When both of these settings are configured, the lower of the two is used.

These settings are configurable by modifying the *Kubelet* arguments at the *Node ConfigMap* (see Figure 9-6).

```
# oc edit cm -n openshift-node node-config-compute
apiVersion: v1
kind: ConfigMap
data:
  node-config.yaml: |
    apiVersion: v1
...
    kind: NodeConfig
    kubeletArguments:
      pods-per-core: # <max. number of running Pods> =  <pods-per-core> * <num. cores on Node>
      - "10"
      max-pods:      # explicit max. number of Pods running on Node
      - "250"
...
```

Figure 9-6 Maximum number of running Pods per Node

USING THE TUNED PROFILE

Tuned[5] is a daemon that monitors devices connected to the Host and statically and dynamically tunes system settings based on a selected Profile.

During the deployment of *OpenShift*, the installer configures the *Nodes* with *Tuned* profiles[6] for *OpenShift* (see Figure 9-7) and assigns them to the *Nodes* based on their role.

```
[root@ocp ~]# tuned-adm active          (1)
Current active profile: openshift-control-plane

[root@ocp ~]# tuned-adm list (2)
Available profiles:
- balanced                      - General non-specialized tuned profile
- desktop                       - Optimize for the desktop use-case
- latency-performance           - Optimize for deterministic performance at the cost of increased power consumption
- network-latency               - Optimize for deterministic performance at the cost of increased power consumption, focused on low latency network performance
- network-throughput            - Optimize for streaming network throughput, generally only necessary on older CPUs or 40G+ networks
- openshift                     - Optimize systems running OpenShift (parent profile)
- openshift-control-plane       - Optimize systems running OpenShift control plane
- openshift-node                - Optimize systems running OpenShift nodes
- powersave                     - Optimize for low power consumption
- throughput-performance        - Broadly applicable tuning that provides excellent performance across a variety of common server workloads
- virtual-guest                 - Optimize for running inside a virtual guest
- virtual-host                  - Optimize for running KVM guests
Current active profile: openshift-control-plane

```

Figure 9-7 The tuned profiles for OpenShift

Eviction Policy

The *Eviction Policy*[7] enables the *Node* to reclaim needed resources by failing one or more *Pods* when the *Node* is running low on available resources. OpenShift supports two types of eviction policy:

- **hard**: The *Node* takes immediate action to reclaim resources from a Pod that exceeds predefined thresholds (see #1 Figure 9-8).

- **soft**: The *Node* waits for a grace period (see #3 Figure 9-8) before reclaiming resources from a Pod exceeding the thresholds (see #2 Figure 9-8).

```
# oc edit cm -n openshift-node node-config-compute
apiVersion: v1
kind: ConfigMap
data:
  node-config.yaml: |
    apiVersion: v1
...
    kind: NodeConfig
    kubeletArguments:
      eviction-hard: 1
      - memory.available<100Mi
      - nodefs.available<10%
      - nodefs.inodesFree<5%
      - imagefs.available<15%
...
```

```
# oc edit cm -n openshift-node node-config-compute
apiVersion: v1
kind: ConfigMap
data:
  node-config.yaml: |
    apiVersion: v1
...
    kind: NodeConfig
    kubeletArguments:
      eviction-soft: 2
      - memory.available<500Mi
      - nodefs.available<500Mi
      - nodefs.inodesFree<100Mi
      - imagefs.available<100Mi
      - imagefs.inodesFree<100Mi
      eviction-soft-grace-period: 3
      - memory.available=1m30s
      - nodefs.available=1m30s
      - nodefs.inodesFree=1m30s
      - imagefs.available=1m30s
      - imagefs.inodesFree=1m30s
...
```

Figure 9-8 Eviction Policies

The *Eviction Policy* settings are configurable by modifying the *Kubelet* arguments at the *Node ConfigMap* (see Figure 9-8).

Pod Scheduling

OpenShift Pod Scheduler [8] is an internal process responsible for determining the placement of new *Pods* onto *Nodes*. It does this by identifying a *Node* that can provide the *Pod's* requirements while complying with configured policies.

The available *Nodes* are filtered by rules known as *Predicates*. The resulting list is sorted by rules that rank *Nodes* according to preferences and determine a *Priority*.

The configuration for the default scheduler policy containing the default *Predicates* and *Priorities* is on the *Master Nodes* at `/etc/origin/master/scheduler.json`.

In addition to the default scheduler, there are several ways to invoke advanced scheduling of Pods using

- **Pod Affinity and Anti-affinity**[9] : Pods specify affinity or anti-affinity toward a group of *Pods* (e.g., for an application's latency requirement) using labels on *Nodes* and label selectors on *Pods* to control where a *Pod* can be placed.
- **Node Affinity**[10]: Pods specify affinity or anti-affinity toward a group of Nodes using labels on *Nodes* and label selectors on *Pods* to control where a *Pod* can be placed.
- **Node Selectors**[11]: Use labels on *Nodes* and label selectors on *Pods* to control the scheduling on where a *Pod* can be placed.
- **Taints and Tolerations**[12]: *Taints* are labels on a *Node* to refuse Pods to be scheduled onto the *Node* unless the *Pod* has a matching *Toleration*. *Tolerations* are labels on a *Pod*. The *Taints* and *Tolerations* labels on the *Node* and on the *Pod* must match in order to be able to schedule the *Pod* onto the *Node*.

Pod Priority

Pod Priority[13] is used to indicate the relative importance of a *Pod* compared to other *Pods*. The *Scheduler* orders *Pods* in queues by their *Priority* with higher *priority Pods* ahead of other lower *priority Pods*.

The *PriorityClass* are cluster-level (non-namespaced) objects defining a mapping between a *name* and an integer representing the *Priority* of the class. The higher the number, the higher the *priority*.

The *priority* number is any 32-bit integer with a value smaller than or equal to 1,000,000,000 (one billion). Higher values are reserved for critical *Pods* that should not be preempted or evicted.

OpenShift has two reserved *PriorityClasses* for critical system *Pods* as seen in Table 9-1.

Table 9-1 OpenShift Reserved PriorityClasses

PriorityClass Name	Priority Value	Description
system-node-critical	2,000,001,000	Used for all Pods that should never be evicted from a Node. This includes Pods like sdn-ovs, sdn, and others.
system-cluster-critical	2,000,000,000	Used with Pods that are important for the normal operations of the cluster. Pods with this priority include fluentd, descheduler, and others.

The *PriorityClass name* field is used by the *Priority Admission Controller* to identify the integer value of the *priority*. If the named *PriorityClass* is not found, the *Pod* is rejected. An example of the definition and utilization of a *PriorityClass* can be seen in Figure 9-9.

```
1  apiVersion: scheduling.k8s.io/v1beta1
2  kind: PriorityClass (1)
3  metadata:
4    name: demo-high-priority (2) # name of the PriorityClass object
5  value: 1000000               # priority actual value
6  globalDefault: false         # default for Pods not specifying a PriorityClass name?
7  description: "This is a demo priority class."
```

```
1  apiVersion: v1
2  kind: Pod (3)
3  metadata:
4    name: my-demo-app
5  spec:
6    containers:
7    - name: myy-demo-app
8      image: my-demo-app
9      imagePullPolicy: IfNotPresent
10   priorityClassName: demo-high-priority (4)
```

Figure 9-9 Defining and using a PriorityClass

When a high-priority *Pod* disrupts the *Node resource budget* , the scheduler attempts to preempt *Pods,* starting with lower-priority *Pods,* avoiding violating the *Pod disruption budget.*

When the scheduling of a new high-priority Pod requires the eviction of a lower-priority *Pod* that has a *Pod Affinity* rule with a high-priority *Pod* running in the *Node*, the scheduler attempts to identify a different *Node* to schedule the new high-priority *Pod*.

Summary

This chapter documents some of the Day-2 operations tasks for the maintenance and operation of OpenShift clusters. In addition, the chapter presents some of the settings a cluster administrator can use to allocate resources for system or platform critical tasks.

There are many more settings available for the reader to discover from the official OpenShift documentation. The settings covered in this chapter are applicable for the most common scenarios.

The OpenShift platform provides sensible defaults optimized for what is sometimes referred to as general Cloud-native workloads, meaning the workloads for which Kubernetes has been designed which were expected to be TCP-based, web-enabled, and entirely agnostic to the underlying hardware infrastructure. With the adoption of Kubernetes outside the web-

based application, there is the need to support hardware acceleration (i.e., GPUs, FPGAs, etc.) or multiple NICs per Container, and much more. Chapter 10 explores how some of these advanced compute and networking capabilities are supported in OpenShift.

Footnotes

1 Measuring performance of *etcd*, refer to the documentation at https://github.com/etcd-io/etcd/blob/master/Documentation/op-guide/performance.md

2 Additional details on pruning object are available at the online documentation: https://docs.openshift.com/container-platform/3.11/admin_guide/pruning_resources.html

3 Additional details are available in the documentation at https://docs.openshift.com/container-platform/3.11/admin_guide/garbage_collection.html

4 Additional information about configuring Node resources is available at the online documentation: https://docs.openshift.com/container-platform/3.11/admin_guide/allocating_node_resources.html

5 Additional information about Tuned is available at the RHEL documentation (requires a valid RHN subscription): https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/performance_tuning_guide/index#chap-Red_Hat_Enterprise_Linux-Performance_Tuning_Guide-Tuned

6 Additional information about the OpenShift Tuned profiles is available at the scaling and performance documentation (Requires a valid RHN subscription) https://access.redhat.com/documentation/en-us/openshift_container_platform/3.11/html-single/scaling_and_performance_guide/index#scaling-performance-capacity-tuned-profile

7 Additional information on OpenShift Eviction Policies is available online at https://docs.openshift.com/container-platform/3.11/admin_guide/out_of_resource_handling.html#out-of-resource-eviction-policy

8 The OpenShift default scheduler is described in more detail in the online documentation: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/scheduler.html

9 Advanced Scheduling using Pod Affinity and Anti-Affinity: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/pod_affinity.html#admin-guide-sched-pod-affinity

10 Advanced Scheduling using Node Affinity: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/node_affinity.html#admin-guide-sched-affinity

11 Advanced Scheduling using Node Selector: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/node_selector.html#admin-guide-sched-selector

12 Advanced Scheduling using Taints and Tolerations: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/taints_tolerations.html#admin-guide-taints

13 Additional information about Pod Priority is available at the online documentation: https://docs.openshift.com/container-platform/3.11/admin_guide/scheduling/priority_preemption.html#priori priority-about_priority-preemption