



Red Hat OpenStack Platform

11

Deployment Recommendations for Specific Red Hat OpenStack Platform Services

Maximizing the performance of specific Red Hat OpenStack Platform services in an enterprise environment

OpenStack Team

Red Hat OpenStack Platform 11 Deployment Recommendations for Specific Red Hat OpenStack Platform Services

Maximizing the performance of specific Red Hat OpenStack Platform services in an enterprise environment

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2017 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This article enumerates several recommendations for deploying specific Red Hat OpenStack Platform services. These recommendations aim to address many performance bottlenecks when deploying OpenStack through the director.

Table of Contents

| | |
|--|----------|
| 1. OVERVIEW | 2 |
| 2. TELEMETRY | 2 |
| 2.1. Small Overclouds | 2 |
| 2.1.1. Disable Telemetry Entirely | 2 |
| 2.1.2. Use a File Back End | 3 |
| 2.2. Large and Production-Scale Overclouds | 3 |
| 2.2.1. Use Separate, Dedicated Telemetry Nodes | 4 |
| 2.2.2. Configure Object Storage as Recommended | 5 |
| 3. OBJECT STORAGE | 5 |
| 3.1. Use Separate Disks for the Object Storage Service | 5 |
| 3.2. Use Separate, Dedicated Storage Nodes | 6 |
| 3.3. Increase Default Partition Power | 6 |
| 4. FURTHER READING | 7 |

1. OVERVIEW

The Red Hat OpenStack Platform director aims to provide customers with as much configuration flexibility, allowing them to tailor-fit the overcloud to address their specific needs. The director also does this while trying to make the deployment process as easy and quick as possible.

To allow for an easy and painless deployment, the director manages the configuration of many service settings and applies sane, thoroughly-tested defaults. These defaults were selected for their suitability in deploying small overcloud environments, mostly because many overclouds start out small and scale out according to the needs of the business. In addition, a majority of overcloud deployments are test environments that businesses use to study OpenStack and evaluate its suitability for their operations.

If you are planning to scale to or deploy a large overcloud, optimize your overcloud to prevent any potential bottlenecks as its workload increases. The recommendations in this article help you do so, further preventing scale from affecting the performance of specific services within the overcloud.

2. TELEMETRY

The defaults applied by the director for the Telemetry service are typically suitable for small deployments. Such deployments generally translate to proof-of-concept or testing environments, and are useful for environments with a limited number of nodes.



Note

For reference, a *small* deployment is a Red Hat OpenStack Platform overcloud built to support less than 100 instances, with a maximum of 12 physical cores (or 24 cores with hyperthreading enabled) per Controller node.

Many of these defaults are not optimal for heavy production workloads, and addressing them early in the overall overcloud design can help prevent performance issues later on. Telemetry is a CPU-intensive service that the director enables and installs on the Controller by default — this, in turn, can impact the performance of other Controller services significantly.

The following subsections describe recommendations for optimizing Telemetry for both small and large overcloud environments:

2.1. Small Overclouds

By default, the director configures Telemetry with the minimum necessary metrics to gather. This is suitable for small deployments where Telemetry will be used sparingly or not at all.

2.1.1. Disable Telemetry Entirely

If you are not planning to use Telemetry in your overcloud at all, prevent the director from enabling it altogether. This prevents Telemetry from affecting the performance of other services.

To disable Telemetry, add the following to the **resource_registry:** of your environment file:

```
resource_registry:
```

```

OS::TripleO::Services::CeilometerApi: OS::Heat::None
OS::TripleO::Services::CeilometerCollector: OS::Heat::None
OS::TripleO::Services::CeilometerExpirer: OS::Heat::None
OS::TripleO::Services::CeilometerAgentCentral: OS::Heat::None
OS::TripleO::Services::CeilometerAgentNotification: OS::Heat::None
OS::TripleO::Services::CeilometerAgentIpmi: OS::Heat::None
OS::TripleO::Services::GnocchiApi: OS::Heat::None
OS::TripleO::Services::GnocchiMetricd: OS::Heat::None
OS::TripleO::Services::GnocchiStatsd: OS::Heat::None
OS::TripleO::Services::AodhApi: OS::Heat::None
OS::TripleO::Services::AodhEvaluator: OS::Heat::None
OS::TripleO::Services::AodhNotifier: OS::Heat::None
OS::TripleO::Services::AodhListener: OS::Heat::None
OS::TripleO::Services::ComputeCeilometerAgent: OS::Heat::None
OS::TripleO::Services::PankoApi: OS::Heat::None

```

Then, add the following to the **parameter_defaults**: of your environment file to disable notifications:

```

parameter_defaults:
  ExtraConfig:
    neutron::notification_driver: noop
    nova::notification_driver: noop
    keystone::notification_driver: noop
    glance::notify::rabbitmq::notification_driver: noop
    cinder::ceilometer::notification_driver: noop
    manila::notification_driver: noop
    sahara::notify::notification_driver: noop
    barbican::api::notification_driver: noop
    ceilometer::notification_driver: noop

```

For more information about disabling Red Hat OpenStack Platform services, see [Adding and Removing Services from Roles](#). For details on disabling Telemetry services, see [How to disable gnocchi, aodh and ceilometer in Red Hat OpenStack Platform 10](#).

2.1.2. Use a File Back End

If you need to enable Telemetry, you can lower its performance impact by using a file back end for the **gnocchi** service. To do this, add the following to the **parameter_defaults**: of your environment file:

```

parameter_defaults:
  GnocchiBackend: file

```

Important

This is only advisable if you are deploying small, proof-of-concept overcloud deployments with High Availability disabled.

2.2. Large and Production-Scale Overclouds

Telemetry uses the enabled object store as its storage back end. If you are not enabling Red Hat Ceph, this means Telemetry will use Object Storage (**swift**). By default, the director will co-locate Object Storage with Telemetry on the Controller.

2.2.1. Use Separate, Dedicated Telemetry Nodes

Consider deploying Telemetry on its own dedicated node. This will prevent its heavy use of CPU time from affecting the performance of other Controller services.

To set dedicated Telemetry nodes, remove the Telemetry services from the Controller role. Copy `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml` to `/home/stack/templates/roles_data.yaml`. Then, remove the following lines from the `ServicesDefault` list of the Controller role:

```
- OS::TripleO::Services::CeilometerApi
- OS::TripleO::Services::CeilometerCollector
- OS::TripleO::Services::CeilometerExpirer
- OS::TripleO::Services::CeilometerAgentCentral
- OS::TripleO::Services::CeilometerAgentNotification
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::AodhApi
- OS::TripleO::Services::AodhEvaluator
- OS::TripleO::Services::AodhNotifier
- OS::TripleO::Services::AodhListener
- OS::TripleO::Services::PankoApi
- OS::TripleO::Services::CeilometerAgentIpmi
```

Next, add the following snippet to `/home/stack/templates/roles_data.yaml`:

```
- name: Telemetry
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::GnocchiApi
    - OS::TripleO::Services::GnocchiMetricd
    - OS::TripleO::Services::GnocchiStatsd
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
```


- OS::Triple0::Services::AodhNotifier
- OS::Triple0::Services::AodhListener
- OS::Triple0::Services::PankoApi
- OS::Triple0::Services::CeilometerAgentIpmi

Set how many dedicated nodes for the Telemetry service through your environment file. For example, add **TelemetryCount: 3** to the **parameter_defaults** in your environment file (for example, `/home/stack/storage-environment.yaml`) to deploy 3 dedicated Telemetry nodes:

```
parameter_defaults:
  TelemetryCount: 3
```

At this point, you now have a custom **Telemetry** role. With this, you can define a new flavor that you can use to tag and assign specific Telemetry nodes. For instructions, see [Creating a New Role](#).

When you deploy your overcloud, include both files to apply the settings:

```
$ openstack overcloud deploy \
  -r /home/stack/templates/roles_data.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  [...]
```

2.2.2. Configure Object Storage as Recommended

If you cannot allocate dedicated nodes to Telemetry and still need to use Object Storage as its back end, configure Object Storage to lower overall storage I/O on the Controller node. See [Section 3, “Object Storage”](#) for details.

3. OBJECT STORAGE

If you do not deploy Red Hat OpenStack Platform with Ceph, the director will deploy the Object Storage service (**swift**). This will also serve as the object store for several other OpenStack services, including (but not limited to) Telemetry and RabbitMQ.

3.1. Use Separate Disks for the Object Storage Service

By default, the director uses the directory `/srv/node/d1` on the system disk for Object Storage. On the Controller, this disk is used by other services as well; this could become a performance bottleneck once Telemetry starts recording events in an enterprise setting. To mitigate this, use one or more separate disks for the Object Storage service.

The following environment file snippet, for example, uses two separate disks on each Controller node for the Object Storage service. It creates an XFS file system on these disks (using the whole disk), and configures the Object Storage service use them as storage devices as well:

```
parameter_defaults:
  SwiftRawDisks: {"sdb": {}, "sdc": {}}
```

SwiftRawDisks defines each storage disk on the node. This example defines both **sdb** and **sdc** disks on each Controller node.

When configuring multiple disks, ensure that the Bare Metal service (**ironic**) uses the intended root disk. See [Defining the Root Disk for Nodes](#) for more information.

3.2. Use Separate, Dedicated Storage Nodes

You can also set dedicated nodes for the Object Storage service. Doing so will prevent any disk I/O by the Telemetry service from affecting any other services on the Controller node.

To do set dedicated Object Storage nodes, create a custom `roles_data.yaml` file (based on the default `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml`) and edit it by removing the Object Storage service entry from the Controller node. Specifically, remove the following line from the `ServicesDefault` list of the `Controller` role:

```
- OS::TripleO::Services::SwiftStorage
```

Then, use the `ObjectStorageCount` resource in your custom environment file to set how many dedicated nodes to allocate for the Object Storage service. For example, add `ObjectStorageCount: 3` to the `parameter_defaults` in your environment file to deploy 3 dedicated Object Storage nodes:

```
parameter_defaults:
  ObjectStorageCount: 3
```

For more information about configuring custom roles, see [Composable Services and Custom Roles](#) and [Adding and Removing Services from Roles](#).

3.3. Increase Default Partition Power

The Object Storage service distributes data across disks and nodes using modified *hash rings* (see [The Rings](#) for more details). There are three rings by default - one for accounts, one for containers, and one for objects. Each ring uses a fixed parameter called *partition power*. This parameter sets the maximum number of partitions that can be created; for an overview, see [OpenStack Object Storage \(swift\)](#).

The partition power is important and can only be changed for new containers and their objects. As such, it is important to set this value before **initial deployment**.

The default value for director-deployed environments is 10. This is a reasonable value for smaller deployments, especially if you only plan to use disks on the Controller nodes for Swift. With larger deployments (for example, when using separate Object Storage service nodes), use a higher value. The following table will help you to select an appropriate partition power if you use three replicas.

Table 1. Appropriate partition power values per number of available disks

| Partition Power | Maximum number of disks |
|-----------------|-------------------------|
| 10 | ~ 35 |
| 11 | ~ 75 |
| 12 | ~ 150 |

| | |
|----|-------|
| 13 | ~ 250 |
| 14 | ~ 500 |



Important

Setting an excessively high value (for example, a partition power of **14** for only 40 disks) will negatively impact replication times.

To set the partition power, use the following resource:

```
parameter_defaults:  
  SwiftPartPower: 11
```

You can also configure an additional object server ring for new containers. This is useful if you want to scale up (that is, add more disks to) an Object Storage deployment that initially uses a low partition power. For more information, see [Configure an Object Storage Ring](#).

4. FURTHER READING

- ✦ [Director Installation and Usage](#)
 - In particular, see [Replacing Object Storage Nodes](#).
- ✦ [Advanced Overcloud Customization](#)
- ✦ [Storage Guide](#)