Red Hat OpenStack Platform 16.0

Deployment Recommendations for Specific Red Hat OpenStack Platform Services

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

Last Updated: 2020-04-06

Red Hat OpenStack Platform 16.0 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

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. Red Hat is currently reviewing the information and procedures provided in this guide for this release. This document is based on the Red Hat OpenStack Platform 15 version, available at . If you require assistance for this release, please contact Red Hat support.

Table of Contents

CHAPTER 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.

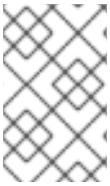
CHAPTER 2. TELEMETRY

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.

Telemetry is a CPU-intensive service that the director installs on the Controller. By default, Telemetry is not enabled in Red Hat OpenStack Platform 16.0.

To enable and optimize Telemetry for small overcloud environments, see Section 2.1, “Small overclouds”. To enable and optimize Telemetry for large overcloud environments, see Section 2.2, “Large and production-scale overclouds”.

2.1. SMALL OVERCLOUDS



NOTE

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.

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, complete the following steps:

1. Add the following to the **parameter_defaults:** of your environment file:

```
parameter_defaults:
  GnocchiBackend: *file*
```

2. Add the **enable-legacy-telemetry.yaml** to your **openstack overcloud deploy** command:

```
openstack overcloud deploy \
-e /home/stack/environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/enable-legacy-telemetry.yaml \
[...]
```

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 do not plan to enable Red Hat Ceph, Telemetry uses Object Storage (swift). By default, the director colocates Object Storage with Telemetry on the Controller.

2.2.1. Using separate, dedicated Telemetry nodes

The heavy use of CPU resources that Telemetry requires can affect the performance of other Controller services. To avoid this, complete the following steps to deploy Telemetry on a dedicated node:

1. 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**.

2. Remove the following lines from the **ServicesDefault** list of the Controller role:

```
    - 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
```

3. 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::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
```

4. Set the number of dedicated nodes for the Telemetry service in the environment file. For example, add **TelemetryCount: 3** to the **parameter_defaults** in the **/home/stack/storage-environment.yaml** file to deploy three dedicated Telemetry nodes:

```
parameter_defaults:
  TelemetryCount: *3*
```

You now have a custom **Telemetry** role. With this role, you can define a new flavor to tag and assign specific Telemetry nodes. For more information, see Creating a New Role in the *Advanced Overcloud Customization* guide.

5. When you deploy your overcloud, include the following files to apply the settings:

```
$ openstack overcloud deploy \
  -r /home/stack/templates/roles_data.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/enable-legacy-telemetry.yaml \
[...]
```

2.2.2. Configure Object Storage as recommended

If you cannot allocate dedicated nodes to Telemetry and you still need to use Object Storage as its back end, configure Object Storage to lower overall storage I/O on the Controller node. For more information, see Chapter 3, *Object Storage*.

CHAPTER 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

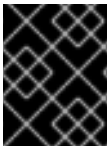
number of partitions that can be created.

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 3.1. 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 .

CHAPTER 4. FURTHER READING

- Director Installation and Usage
  - In particular, see Replacing Object Storage Nodes.
- Advanced Overcloud Customization
- Storage Guide