Red Hat OpenStack Platform 12

OpenStack Data Processing

Manually provisioning and scaling Hadoop clusters in Red Hat OpenStack Platform

Last Updated: 2019-03-05

Red Hat OpenStack Platform 12 OpenStack Data Processing

Manually provisioning and scaling Hadoop clusters in Red Hat OpenStack Platform

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

The OpenStack Data Processing feature allows you to easily provision and scale Hadoop clusters to process large datasets. This guide walks you through the entire OpenStack Data Processing workflow, which includes registering the Data Processing requirements (image, input data, job binaries), configuring templates used to provision clusters, processing data on those clusters, and scaling those clusters as necessary. This release of OpenStack Data Processing includes a Guides tab. This tab features wizards that will help you create the templates necessary in order to launch clusters and run jobs on them. The objective of this guide is to provide a more in-depth look at the OpenStack Data Processing workflow, and will therefore walk you through the template creation and component registration without the use of the Guides tab feature. Using the OpenStack Data Processing feature requires basic knowledge of data processing within the Hadoop framework. Further, users also need to be familiar with the particulars of their chosen Hadoop plug-in.

Table of Contents

1. OVERVIEW

OpenStack Data Processing is provided by the OpenStack Sahara project, which provides a robust interface to easily provision and scale Hadoop clusters. Such clusters can then be used to run resource-intensive jobs, typically for processing large data sets. As an OpenStack component, OpenStack Data Processing is fully integrated into the OpenStack ecosystem; for example, users can administer the entire Hadoop data processing workflow through the OpenStack dashboard — from configuring clusters, all the way to launching and running jobs on them.

For more information about Hadoop, see http://hadoop.apache.org/.

OpenStack Data Processing uses different plug-ins for provisioning specific clusters of each Hadoop distribution. The Hadoop parameters available for configuring clusters differ depending on the Hadoop distribution (and, by extension, the plug-in used). As of this release (Red Hat OpenStack Platform 12), OpenStack Data Processing supports the following plug-ins:

- Cloudera (CDH)
  - version 5.7
  - version 5.9
  - version 5.11
- Hortonworks Data Platform
  - version 2.3
  - version 2.4
- MapR 5.2

OpenStack Data Processing also includes a **Guides** tab. This tab features wizards that will help you create the templates necessary in order to launch clusters and run jobs on them. However, you will still need to register the components necessary for using OpenStack Data Processing, such as Hadoop images and job binaries. As such, if you intend to use the **Guides** feature, we recommend you read Section 5, “Register the Required Components” first.

2. INSTALLATION

This section assumes that you are deploying OpenStack Data Processing on the overcloud, through the director. It also assumes that you created a *director installation user* named **stack**, identical to Creating a Director Installation User (from Director Installation and Usage).

The director uses *environment files* to configure the overcloud during deployment. These environment files are stored in **`/usr/share/openstack-tripleo-heat-templates/`**, along with the Puppet scripts and heat templates used to orchestrate each service.

The OpenStack Data Processing service is disabled on the overcloud by default. This is because the OpenStack Data Processing components are registered as null operations (**`OS::Heat::None`**) in the main overcloud environment file (**`/usr/share/openstack-tripleo-heat-templates/overcloud-resource-registry-puppet.yaml`**):

```
  OS::TripleO::Services::SaharaApi: OS::Heat::None
  OS::TripleO::Services::SaharaEngine: OS::Heat::None
```

To enable these services, include an environment file that links these resources to their respective Heat templates in the **/usr/share/openstack-tripleo-heat-templates/puppet/services** directory. Some services have predefined environment files in the **environments** directory. For example, the Data Processing service (**sahara**) use the **/usr/share/openstack-tripleo-heat-templates/environments/services/sahara.yaml** file, which contains the following:

```
resource_registry:
  OS::TripleO::Services::SaharaApi: ../../docker/services/sahara-api.yaml
  OS::TripleO::Services::SaharaEngine: ../../docker/services/sahara-
engine.yaml
```

This overrides the default null operation resources and enables the services. Include this environment file when running the **openstack overcloud deploy** command.

```
$ openstack overcloud deploy --templates -e \
  /usr/share/openstack-tripleo-heat-
templates/environments/services/sahara.yaml
```

NOTE

You can also enable *instance locality* to force clusters to use volumes local to the same node. Doing so could improve the performance of your cluster. See Appendix A, *Enabling Instance Locality* for related infrormation.

IMPORTANT

If you passed any extra environment files when you created the overcloud, pass them again here using the **-e** option to avoid making undesired changes to the overcloud. For more information, see Modifying the Overcloud Environment (from Director Installation and Usage).

3. WORKFLOW

OpenStack Data Processing provisions and scales Hadoop clusters using pre-configured cluster templates that define specifically designed instances. These instances form the individual nodes that make up Hadoop clusters; you can then use these Hadoop clusters to run the jobs/binaries that will process your data.

If you intend to use OpenStack Data Processing, you should already be familiar with the necessary components for working within the Hadoop framework. As such, the general workflow described in this section assumes that you already have the following components prepared:

- A Hadoop image; specifically, a Red Hat Enterprise Linux image containing a Hadoop data processing plug-in. See Section 1, “Overview” for a list of supported plug-ins.
- The input data you wish to process, preferably uploaded to the Object Storage service.
- The job binaries and libraries you will use to process the input data, preferably uploaded to the Object Storage service.

NOTE

For details on how to upload content to the Object Storage service, see Upload an Object.

In addition, you should also have a general idea of the computational resources required to run the job. This will help you determine what type of nodes (and how many of each) you will need.

The following high-level workflow describes how to configure and use the OpenStack Data Processing service to launch clusters and run jobs on those clusters:

1. Create an image containing the necessary plug-in components for OpenStack Data Processing (Section 4, “Create Hadoop Image”). This will be your Hadoop image.
   The procedure for creating this image differs depending on your chosen Hadoop plug-in.
2. Register the following required components to the OpenStack Data Processing service:
   - Hadoop image
   - Data sources (namely, your input data and where the output data should go)
3. Create node group templates. Each template defines many useful Hadoop-specific settings for any given node, most notably:
   - What Hadoop plug-in and version should the node group use?
   - Which processes should run on the node?
4. Create or upload cluster templates. A cluster template defines, among other things:
   - *Node group composition*: namely, how many nodes of each node group should make up the cluster.
   - *Cluster-scoped Hadoop configurations*: specific parameters you need to set for each Hadoop component (HIVE, AMBARI, HDFS, and the like).
5. Launch a Hadoop cluster (using a cluster template), and run a job on the cluster (namely, running a registered job binary on a data source). You can also scale the cluster (as in, add or remove nodes of any type) as needed.
6. Register job binaries, scripts, or libraries to the OpenStack Data Processing Service, create jobs, and launch them on Hadoop clusters. Jobs define which job binaries, scripts, or libraries should be used to process registered data sources.

The next few sections describe each workflow step in greater detail.

4. CREATE HADOOP IMAGE

This release supports the following plug-ins:

- Cloudera (CDH)
  - version 5.7
  - version 5.9
  - version 5.11

- Hortonworks Data Platform
  - version 2.3
  - version 2.4
- MapR 5.2

To create a Hadoop image based on any of these plug-ins, you need to first download a Red Hat Enterprise Linux 7 image. You can get one from the following link (requires a Red Hat subscription):

https://access.redhat.com/downloads/content/69/ver=/rhel---7/7.5/x86_64/product-software

After downloading the image, configure your system to use your Red Hat subscription details. Doing so will allow your system to further download the tools necessary for creating a Hadoop image from your downloaded Red Hat Enterprise Linux image. To do so, enter the following commands from a terminal (as root):

```
# export REG_USER=USERNAME
# export REG_PASSWORD=PASSWORD
# export REG_POOL_ID=POOLID
```

Where:

- *USERNAME* and *PASSWORD* are your Red Hat subscription credentials.
- *POOLID* is your Red Hat subscription's Pool ID. To determine your subscription's Pool ID, enter following command:

```
# subscription-manager list --available
```

This command will display important details about your subscription, including the **Pool ID** value.

NOTE

For more information about manually configuring your Red Hat subscription, see Using and Configuring Red Hat Subscription Manager.

Next, install the **sahara-image-elements** package:

```
# yum install sahara-image-elements -y
```

Finally, create the Hadoop image. To do this, go to the same directory as your Red Hat Enterprise Linux image and run the following command:

```
# sahara-image-create -p PLUGIN -v VERSION
```

Replace *PLUGIN* with the plug-in that the Hadoop image should use. The *VERSION* argument should be specified only when OpenStack Data Processing supports multiple versions of the same plug-in.

The following values can be used for *PLUGIN*:

- **cloudera** (CDH)

- Include **-v 5.7** (for version 5.7), **-v 5.9** (for version 5.9) or **-v 5.11** (for version 5.11) when you run the command.

- **ambari** (Apache Ambari for Hortonworks Data Platform)
  - Include **-v 2.3** (for version 2.3) or **-v 2.4** (for version 2.4) when you run the command.

- **mapr** (MapR 5.2). The -v option should not be specified.

The **sahara-image-create** command will create an image containing the name of the plug-in (for example, **sahara-rhel-7-5-cdh-5-11.qcow2** for **cloudera**).

5. REGISTER THE REQUIRED COMPONENTS

OpenStack Data Processing requires a Hadoop image containing the necessary elements to launch and use Hadoop clusters. Specifically, Red Hat OpenStack Platform requires an image containing Red Hat Enterprise Linux with the necessary data processing plug-in.

Once you have a Hadoop image suitable for the jobs you wish to run, register it to the OpenStack Data Processing service. To do so:

1. Upload the image to the Image service. For instructions on how to do so, see Upload an Image.
2. After uploading the image, select **Project > Data Processing > Image Registry** in the dashboard.
3. Click **Register Image**, and select the Hadoop image from the **Image** drop-down menu.
4. Enter the user name that the OpenStack Data Processing service should use to apply settings and manage processes on each instance/node. The user name set for this purpose on the official images provided by Red Hat Enterprise Linux (which you used in Section 4, “Create Hadoop Image”) is cloud-user.
5. By default, the OpenStack Data Processing service will add the necessary plug-in and version tags in the **plug-in** and **Version** drop-down menu. Verify that the tag selection is correct, then click **Add plugin tags** to add them. The OpenStack Data Processing service also allows you to use custom tags to either differentiate or group registered images. Use the **Add custom tag** button to add a tag; tags appear in the box under the Description field.
   To remove a custom tag, click the **x** beside its name.
6. Click **Done**. The image should now appear in the **Image Registry** table.

5.1. Register Input and Output Data Sources

After registering an image, register your data input source and output destination. You can register both as objects from the Object Storage service; as such, you need to upload both as objects first. For instructions on how to do so, see Upload an Object.

NOTE

You can also register data objects straight from another Hadoop-compatible distributed file system (for example, HDFS). For information on how to upload data to your chosen distributed file system, see its documentation.

1. In the dashboard, select **Project > Data Processing > Data Sources**.

2. Click **Create Data Source**. Enter a name for your data source in the **Name** field.

3. Use the **Description** field to describe the data source (optional).

4. Select your data source's type and URL. The procedure for doing so depends on your source's location:

   - If your data is located in the Object Storage service, select **Swift** from the **Data Source Type** drop-down menu. Then:

     a. Provide the container and object name of your data source as **swift://*CONTAINER*/*OBJECT*** in the **URL** field.

     b. If your data source requires a login, supply the necessary credentials in the **Source username** and **Source password** fields.

   - If your data is located in a Hadoop Distributed File System (HDFS), select the corresponding source from the Data Source Type drop-down menu. Then, enter the data source's URL in the **URL** field as **hdfs://*HDFSHOST:PORT/OBJECTPATH***, where:

     - ***HDFSHOST*** is the host name of the HDFS host.

     - ***PORT*** is the port on which the data source is accessible.

     - ***OBJECTPATH*** is the available path to the data source on ***HDFSHOST***.

5. Click **Done**. The data source should now be available in the **Data Sources** table.

Perform this procedure for each data input/output object required for your jobs.

6. CONFIGURE NODE GROUP TEMPLATES

Data processing through Hadoop involves running jobs on clusters. Each cluster is made up of specially-configured nodes (or instances). OpenStack Data Processing allows you to configure templates defining the different types of nodes required for each cluster. These templates are called *node group templates*.

A node group templates defines the settings required for each node in a Hadoop cluster. The following procedure describes how to configure a node group template:

1. In the dashboard, select **Project > Data Processing > Node Group Templates**.

2. Click **Create Template**.

3. Use the **Plugin Name** and **Version** drop-down menus to select the name and version of the Hadoop plug-in that the node will use.

4. Click **Create**.

5. Enter a name for your template in the **Template Name** field.

6. Use the **Description** field to describe the node group template you are creating (optional).

7. Select a flavor that the nodes should use from the **OpenStack Flavor** drop-down menu. The flavor should be appropriate to the computing, memory, and storage needs of the node. For more information about flavors, see Manage Flavors.

8. Select a storage location for your node from the **Storage location** drop-down menu:

- **Ephemeral Drive**: the OpenStack Data Processing service will generate the necessary ephemeral storage for the node.
- **Cinder Volume**: with this option, you can configure the Block Storage service to create a set number of volumes for each node. Use the Volumes per node field to set the number of volumes, and the Volumes size (GB) field to specify the size of each volume.
  For most deployments, we recommend the **Cinder Volume** option. Data on ephemeral drives are dependent on the integrity of the host, and therefore vulnerable to host failures.

9. Use the **Floating IP pool** drop-down menu to set whether the node should use floating IPs; and if so, which one. For more information about floating IPs, see Configure IP Addressing.
10. Next, configure the node's security. To create a security group for the node group, select the **Auto Security Group** check box. You can also launch the node in existing security groups by selecting their corresponding check boxes from the **Security Groups** list.
    See Project Security Management for more details about security groups.
11. Finally, choose which Hadoop processes should be launched in the node group. To do so, select each chosen processes' check box from the **Processes** list. Consult your chosen Hadoop plug-in's documentation for more information about each process.
12. Click **Create**. The template should now appear in the **Node Group Template** table.

The **Node Group Template** table lists all available node group templates, their respective Hadoop plug-ins, and versions. This table also lists which Hadoop processes will run on each template's nodes.

7. CONFIGURE CLUSTER TEMPLATES

After registering the required components (Section 5, "Register the Required Components") and configuring the types of nodes you need (Section 6, "Configure Node Group Templates"), you can now configure cluster templates. Cluster templates define the following settings for a cluster:

- The node composition of the cluster (as in, how many of each node type). Available node types are listed **Project > Data Processing > Node Group Templates**.
- The Hadoop plug-in and version used by the cluster.
- A list of processes that should only be launched once on a single host (*anti-affinity*).
- Cluster-level parameters for Hadoop services.

To create and configure a cluster template:

1. In the dashboard, select **Project > Data Processing > Cluster Templates**.
2. Use the **Plugin Name** and **Version** drop-down menus to select the name and version of the Hadoop plug-in that the cluster will use.
3. Click **Create**.
4. Enter a name for your template in the **Template Name** field.
5. Use the **Description** field to describe the cluster template you are creating (optional).
6. From the **Use anti-affinity groups for** list, check the box of each process that should **not** be launched more than once on a host.

7. Click the Node Groups tab. From here, configure how many nodes of each node type should run in the cluster. To do this:

   a. Use the **Select a Node Group Template to add:** drop-down menu to select a template (see Section 6, “Configure Node Group Templates” for related information).

   b. Click **+** to create a node group entry based on the template. Do this for each node group template you want to use in the cluster template.

   c. For each node group entry, set how many instances of that node type should launch in the cluster.
   You can also enter a name for each node group entry (optional).

8. Configure Hadoop cluster settings by component as needed. Each component (for example, ZOOKEEPER, AMBARI, HUE) has its own parameters tab. Each tab contains related parameters, most of which also have brief contextual help. For more information on these parameters, consult your selected Hadoop plug-in’s documentation on supported functions for each component.

9. Once you are satisfied with the configuration of your cluster template, click **Create**. Your template should now appear in the **Cluster Templates** table.

Alternatively, you can also use the **Upload Template** button to upload an existing cluster template from your local file system.

You can also launch a cluster using any existing templates from the **Cluster Templates** table. To do this, select **Launch Cluster** from the **Actions** drop-down menu of a cluster template. To view launched clusters, select **Project > Data Processing > Clusters**. For more information about launching a cluster, see Section 8, “Launch a Cluster”.

8. LAUNCH A CLUSTER

Once you have a cluster template (see Section 7, “Configure Cluster Templates”), you can launch a cluster. To do so, open the **Launch Cluster** wizard. There are several ways to open this wizard:

- Select **Project > Data Processing > Cluster Templates**. From there, select **Launch Cluster** from the **Action** drop-down menu of a template.

- Select **Project > Data Processing > Clusters**. From there, click **Launch Cluster**; then, use the **Plugin Name** and Hadoop **Version** drop-down menus to select the name and version of the Hadoop plug-in that the cluster will use.

Afterwards, click **Create**. This will open the **Launch Cluster** wizard:

1. In the **Launch Cluster** wizard, enter a name for the cluster in the **Cluster Name** field.

2. Use the **Description** field to describe the cluster you are launching (optional).

3. If you opened the **Launch Cluster** wizard through the **Cluster** table in **Project > Data Processing > Clusters**, you can use the **Cluster Template** drop-down menu to select a cluster template. You will only be able to select templates that are compatible with the Hadoop plug-in and version you selected earlier.

4. Select a Hadoop image that the cluster should use from the **Base Image** drop-down menu. For details on creating and registering a Hadoop image, see Section 4, “Create Hadoop Image” and Section 5, “Register the Required Components”.

5. If needed, select a key pair from the **Keypair** drop-down menu. You can also click **+** beside this menu to create a new key pair. While key pairs are not required to launch a cluster, you will need them to log into cluster nodes (for example, through SSH).
   For information on key pairs, see Manage Key Pairs.

6. Select which network the cluster should use from the **Neutron Management Network** drop-down menu. For more details on adding and managing networks in OpenStack, see Common Administrative Tasks.

7. Click **Create** to launch the cluster.

To view launched clusters, select **Project > Data Processing > Clusters**.

8.1. Scale or Delete a Cluster

OpenStack Data Processing allows you to easily scale an existing cluster to suit your resourcing needs. Scaling allows you to add or remove nodes of any type (or number) from a running cluster.

To view launched clusters, select **Project > Data Processing > Clusters**. You can scale or delete clusters from this page. For instructions on how to launch a cluster, see Section 8, “Launch a Cluster”.

To scale a cluster:

1. On the **Clusters** table, choose a cluster to scale. Then, select **Scale Cluster** from that cluster’s **Action** drop-down menu. .Scale the cluster as necessary:

   - To add a new node type to the cluster, select its template from the **Select a Node Group Template** to add drop-down menu. Then, click the **+** button to add the node type to the cluster; you can then set how many nodes of that type should be added.

   - To add or remove nodes from an existing node group, use the **+** or **-** buttons on the node group’s row. Alternatively, you can simply set the number of nodes in the node group’s **Count** field.

2. Click **Scale**.

To delete a cluster, select **Delete Cluster** from its **Action** drop-down menu. You can also delete multiple clusters by selecting their check boxes and clicking the **Delete Clusters** button.

9. CONFIGURE JOBS

With the OpenStack Data Processing service, jobs define the actual data processing tasks. Each job specifies a job type (for example, Pig, Hive, or MapReduce), binary, script, and library. A job can only use binaries, scripts, or libraries that are registered with the OpenStack Data Processing service.

After creating a job, you can then launch it on a cluster and run it against an input data source. Input and output data sources, like job binaries, must also be registered first with the OpenStack Data Processing service (see Section 5.1, “Register Input and Output Data Sources”).

9.1. Register Job Binaries, Scripts, or Libraries

The process for registering job binaries, scripts, and libraries is similar to image and data source registration. You can register them directly from the Object Storage service; for instructions on how to upload objects to the Object Storage service, see Upload an Object. Alternatively, you can also upload binaries and libraries directly from your local file system directly into the OpenStack Data Processing service.

1. In the dashboard, select **Project > Data Processing > Job Binaries**.

2. Click **Create Job Binary**.

3. Enter a name for the object (namely, your script, binary, or library). This name will be used when selecting the object. If your object requires a particular name or extension (for example, **.jar**), include it here.

4. Use the **Description** field to describe the script, binary, or library you are registering (optional).

5. Configure the object depending on its storage type.

   a. If the object is available through the Object Storage service, select **Swift** from the Storage type drop-down menu. Then:

      - Provide the container and object name of your script, binary, or library as **swift://*CONTAINER*/*OBJECT*** in the URL field.

      - If your script, binary, or library requires a login, supply the necessary credentials in the **Username** and **Password** fields

   b. Otherwise, select Internal database from the Storage type drop-down menu. Then, use the Internal binary drop-down menu to either:

      - Select an available binary, library, or script from the OpenStack Data Processing service, or

      - Input a script directly into the dashboard (**Create a script**), or

      - Upload a binary, library, or script directly from your local file system (**Upload a new file**).

6. Click **Create**. The binary, library, or script should now be available in the **Job Binaries** table.

9.2. Create a Job Template

Once the required binaries, scripts, and libraries are registered with OpenStack Data Processing, perform the following steps:

1. In the dashboard, select **Project > Data Processing > Job Templates**.

2. Click **Create Job Template**.

3. Enter a name for your job in the **Name** field.

4. Select the correct type from the **Job Type** drop-down menu. For more information about job types, consult your chosen plug-in's documentation regarding supported job types.

5. Select the binary that should be used for this job from the **Choose a main binary** drop-down menu. The options in this menu are populated with job binaries and scripts registered with the OpenStack Data Processing service; for more information, see Section 9.1, "Register Job Binaries, Scripts, or Libraries".

6. Use the **Description** field to describe the job you are creating (optional).

7. If the job binary you specified requires libraries, add them. To do so, click the **Libs** tab and select a library from the **Choose libraries** drop-down menu. Then, click **Choose** to add the library to the job; the library should be included in the Chosen libraries list. Repeat this for every library required by the job binary. Like binaries, the options in the **Choose** libraries drop-down menu are populated with libraries registered with the OpenStack Data Processing service. For more information, see Section 9.1, "Register Job Binaries, Scripts, or Libraries".

8. Click **Create**. The job should now be available in the **Jobs** table.

10. LAUNCH JOBS

After creating a job, you can launch it to process data registered with the OpenStack Data Processing service (see Section 5.1, “Register Input and Output Data Sources”. Jobs require a cluster; you can launch the job on an existing cluster (Section 10.1, “Launch a Job on an Existing Cluster”) or an entirely new one (Section 10.2, “Launch a Job on a New Cluster”).

NOTE

Launching a job involves specifying input data sources and output data destination. Both objects must first be registered with the OpenStack Data Processing service. For more information, see Section 5.1, “Register Input and Output Data Sources”.

10.1. Launch a Job on an Existing Cluster

To view a list of existing clusters in the dashboard, select **Project > Data Processing > Clusters**. For information on how to launch a cluster, see Section 8, “Launch a Cluster”.

To launch a job on an existing cluster:

1. In the dashboard, select **Project > Data Processing > Jobs**. The **Jobs** table displays all available job templates; see Section 9.2, “Create a Job Template” for details on creating new job templates.

2. Choose which job template to use; then, select **Launch On Existing Cluster** from the job template’s **Actions** drop-down menu.

3. On the **Launch Job** wizard, select your input data source from the **Input** drop-down menu. Then, select your output destination from the **Output** drop-down menu.
   If needed, you can also register your input data source or output destination from here. To do so, click the **+** on either **Input** or **Output** drop-down menus. Doing so will open the **Create Data Source** wizard; for more information, see Section 5.1, “Register Input and Output Data Sources”.

4. From the **Cluster** drop-down menu, select which cluster the job should run on.

5. If you need to set any special job properties for this job, click the **Configure** tab. From there, click **Add** under either **Configuration** or **Parameters** to specify any special name/value pairs. You can specify multiple name/value pairs through this tab.
   For more information about supported job properties, consult your chosen Hadoop plug-in’s documentation.

6. Click **Launch**.

To view the status of launched jobs, select **Project > Data Processing > Jobs**. See Section 10.3, “Delete or Re-Launch Launched Jobs” for instructions on how to re-launch or delete a launched job.

10.2. Launch a Job on a New Cluster

After creating a job template, you can also use it to launch a job on an entirely new cluster. Doing so gives you the option to automatically kill the cluster after the job is finished.

1. In the dashboard, select **Project > Data Processing > Jobs**. The **Jobs** table displays all available jobs; see Section 9.2, “Create a Job Template” for details on creating new jobs.

2. Choose which job template to use; then, select **Launch On New Cluster** from the job template's **Actions** drop-down menu.

3. Use the plug-in **Name** and **Version** drop-down menus to select the name and version of the Hadoop plug-in that the job will use.

4. Click **Create**.

5. Enter a name for your cluster in the **Cluster Name** field.

6. Select a Hadoop image that the cluster should use from the **Base Image** drop-down menu. For details on creating and registering a Hadoop image, see Section 4, "Create Hadoop Image" and Section 5, "Register the Required Components".

7. If needed, select a key pair from the **Keypair** drop-down menu. You can also click **+** beside this menu to create a new key pair. While key pairs are not required to launch a cluster, you will need them to log into cluster nodes (for example, through SSH).
   For information on key pairs, see Manage Key Pairs.

8. Select which network the cluster should use from the **Neutron Management Network** drop-down menu. For more details on adding and managing networks in OpenStack, see Common Administrative Tasks.

9. By default, the OpenStack Data Processing service will delete the cluster as soon as the job finishes. To prevent this from happening, select the **Persist cluster after job** exit check box.

10. Next, click the **Job** tab. From there, select your input data source from the **Input** drop-down menu. Then, select your output destination from the **Output** drop-down menu.

11. If needed, you can also register your input data source or output destination from here. To do so, click the **+** on either **Input** or **Output** drop-down menus. Doing so will open the **Create Data Source** wizard; for more information, see Section 5.1, "Register Input and Output Data Sources".

12. If you need to set any special job properties for this job, click the **Configure** tab. From there, click **Add** under either **Configuration** or **Parameters** to specify any special name/value pairs. You can specify multiple name/value pairs through this tab.
    For more information about supported job properties, consult your chosen Hadoop plug-in's documentation.

13. Click **Launch**.

To view the status of launched jobs, select **Project > Data Processing > Jobs**. See Section 10.3, "Delete or Re-Launch Launched Jobs" for instructions on how to re-launch or delete a launched job.

10.3. Delete or Re-Launch Launched Jobs

To view the status of launched jobs, select **Project > Data Processing > Jobs**. From here, you can delete or re-launch a job.

To delete a launched job, select **Delete job execution** from its **Action** drop-down menu. You can also delete multiple launched jobs by selecting their check boxes and clicking the **Delete job executions** button.

To re-launch a job on an existing cluster, select **Relaunch on Existing Cluster** from its **Action** drop-down menu. For instructions on how to continue, see Section 10.1, "Launch a Job on an Existing Cluster".

Alternatively, you can re-launch a job execution on a completely new cluster. To do so, select **Relaunch on New Cluster** from its **Action** drop-down menu. For instructions on how to continue, see Section 10.2, “Launch a Job on a New Cluster”.

A. ENABLING INSTANCE LOCALITY

The performance of a cluster can be greatly improved when it only uses volumes local to the same node. You can force a cluster to only use such volumes through the node group template; specifically, by setting the **volume_local_to_instance** property to **True**.

Before you can use this property, you need to enable its corresponding filter in the Block Storage service. To do so, **`InstanceLocality`** must be included to the list of enabled filter schedulers in **`/etc/cinder/cinder.conf`**:

```
scheduler_default_filters =
AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter,InstanceLocality
```

To configure this through the director, add an environment file to your deployment containing:

```
parameter_defaults:
  ControllerExtraConfig: # 1
    cinder::config::cinder_config:
      DEFAULT/scheduler_default_filters:
        value:
'AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter,InstanceLocality'
```

1 You can also add the **`ControllerExtraConfig:`** hook and its nested sections to the **`parameter_defaults:`** section of an existing environment file.

NOTE

For information about the filter scheduler, see Configure How Volumes are Allocated to Multiple Back Ends (from the Storage Guide).

See Section 2, “Installation” to learn more about deploying the overcloud with OpenStack Data Processing enabled.

After enabling the **`InstanceLocality`** filter, you can now force clusters to use volumes local to the same node. To do so:

1. From the command line, list the names of all available node group templates:

```
$ sahara node-group-template-list
    +------------+--------------------------------------+
    | name       | id                                   ...
    +------------+--------------------------------------+
    | mytemplate | 3d6b4b7c-bca7-4f3a-a6ae-621a31ab7a75 ...
    +------------+--------------------------------------+
```

2. View the template’s properties:

```
$ sahara node-group-template-show --name mytemplate
    +---------------------------+---------------+
    | Property                  | Value         |
    +---------------------------+---------------+
    | volume_local_to_instance  | False         |
    | volumes_availability_zone | None          |
    | description               |               |
```

```
    | availability_zone         |               |
    | volume_mount_prefix       | /volumes/disk |
    | updated_at                | None          |
    ...
```

Here, **volume_local_to_instance** is set to **False**.

3. Create a file named **setting.json** containing the following string:

```
{"volume_local_to_instance": true}
```

4. Update the node group template with the string from **setting.json**:

```
$ sahara node-group-template-update --json setting.json --id 3d6b4b7c-
bca7-4f3a-a6ae-621a31ab7a75
    +---------------------------+---------------+
    | Property                  | Value         |
    +---------------------------+---------------+
    | volume_local_to_instance  | True          |
    | volumes_availability_zone | None          |
    | description               |               |
    | availability_zone         |               |
    | volume_mount_prefix       | /volumes/disk |
    | updated_at                | None          |
    ...
```

Doing so will set the **volume_local_to_instance** property to **True**.