STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

We Addressed the Elephant in the Room with OpenStack Sahara

Ajitha Robert
MSys Technologies, LLC

ABOUT
MSYS TECHNOLOGIES

PRODUCT ENGINEERING SERVICES AND DIGITAL TRANSFORMATION PARTNER

BAY AREA UNICORNS

rubrik PURESTORAGE® COHESITY Acronis

ENTERPRISES

amazon flex Google bakkt

AWARDS

BRONZE 2021 STEVIE WINNER

IAOP 2021 Global Outsourcing 100

RED HERRING 100 ASIA WINNER

GoodFirms

TiE ATLANTA FOSTERING ENTREPRENEURSHIP

IE20 INDIA EMERGING TWENTY

'TOP FASTEST GROWING STORAGE COMPANIES' FOR TWO CONSECUTIVE YEARS

Storage.com Newsletter

OUR WW STRENGTH 1000+ AND GROWING

MSYS OFFICES - USA | INDIA | VIETNAM

STORAGE DEVELOPER CONFERENCE SDC 21

Agenda

- Quick glance – Hadoop, Openstack
- Sahara – Openstack
- Sahara deployment – Deep dive
- Automate the building of distributed Hadoop using Sahara CLI
- Job Execution - Requirements
- A Case Study

Quick glance - Hadoop, Openstack

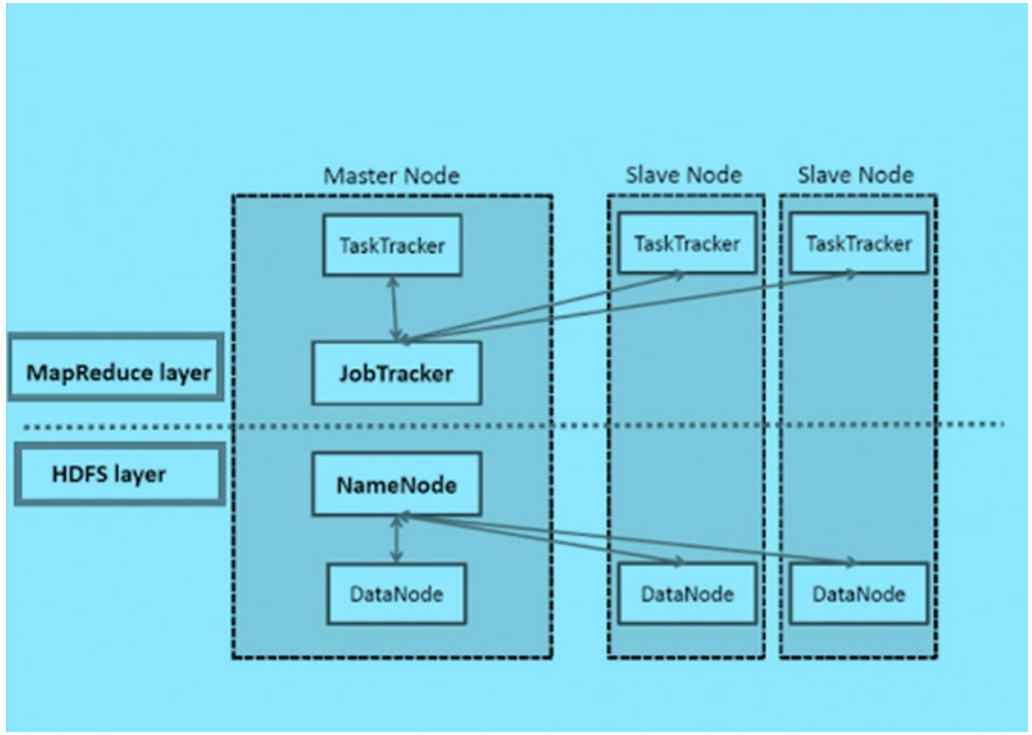
Hadoop Overview

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.. It includes:
  - MapReduce – offline computing engine
  - HDFS – Hadoop distributed file system - creates multiple replicas of data blocks
- Dimensions of big data - volume, velocity, variety
- Scalable, Economical, Efficient, Reliable
- Hadoop clusters - network of master and worker nodes
- Orchestrate and execute the various jobs across the Hadoop distributed file system

Example Applications and Organizations using Hadoop

- ✓A9.com – Amazon
- ✓AOL
- ✓Facebook
- ✓FOX Interactive Media

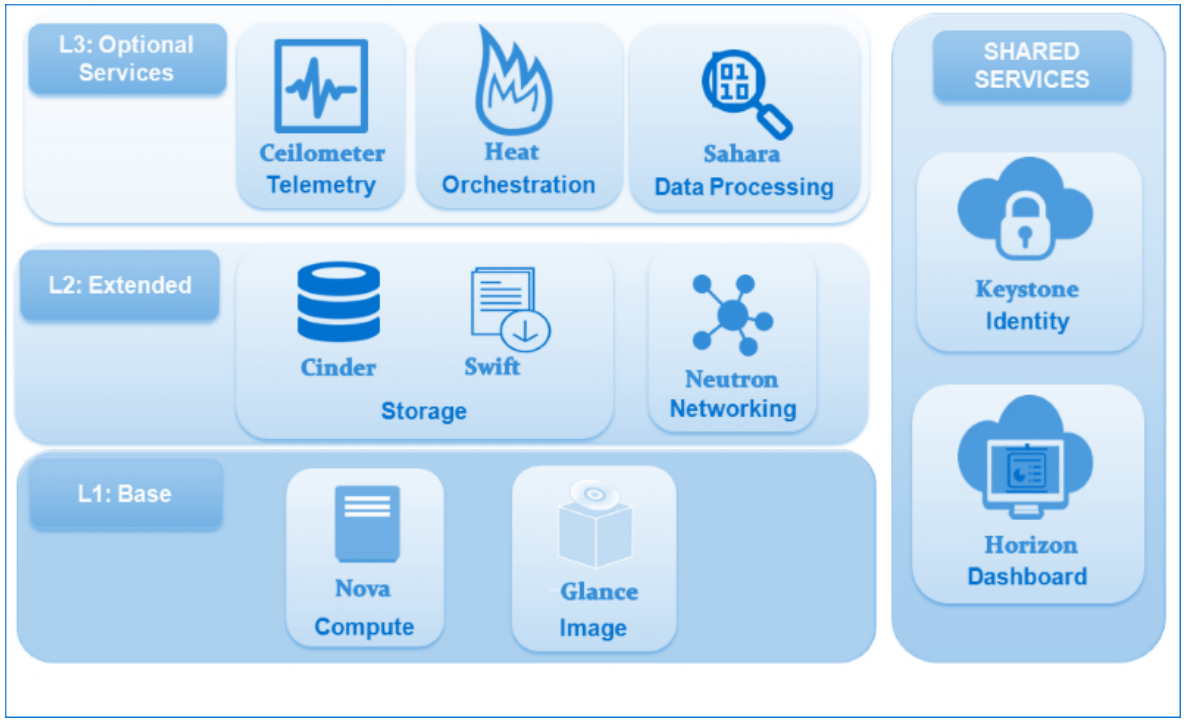
High Level Architecture



Openstack Overview

- Open-source platform that uses pooled virtual resources to build and manage private and public clouds

Openstack Architecture



Keystone(Identity as a Service) – authenticates users and provides security tokens that are used to work with OpenStack

Heat(Orchestration as a Service) – used to provision and orchestrate the deployment of data processing clusters.

Glance(Virtual Machine Image as a Service) – stores VM images with operating system and pre-installed Hadoop/spark software packages to create a data processing cluster.

Nova(Compute as a Service) – provisions a virtual machine for data processing clusters.

Neutron(Networking as a Service) – facilitates networking services from basic to advanced topology to access the data processing clusters.

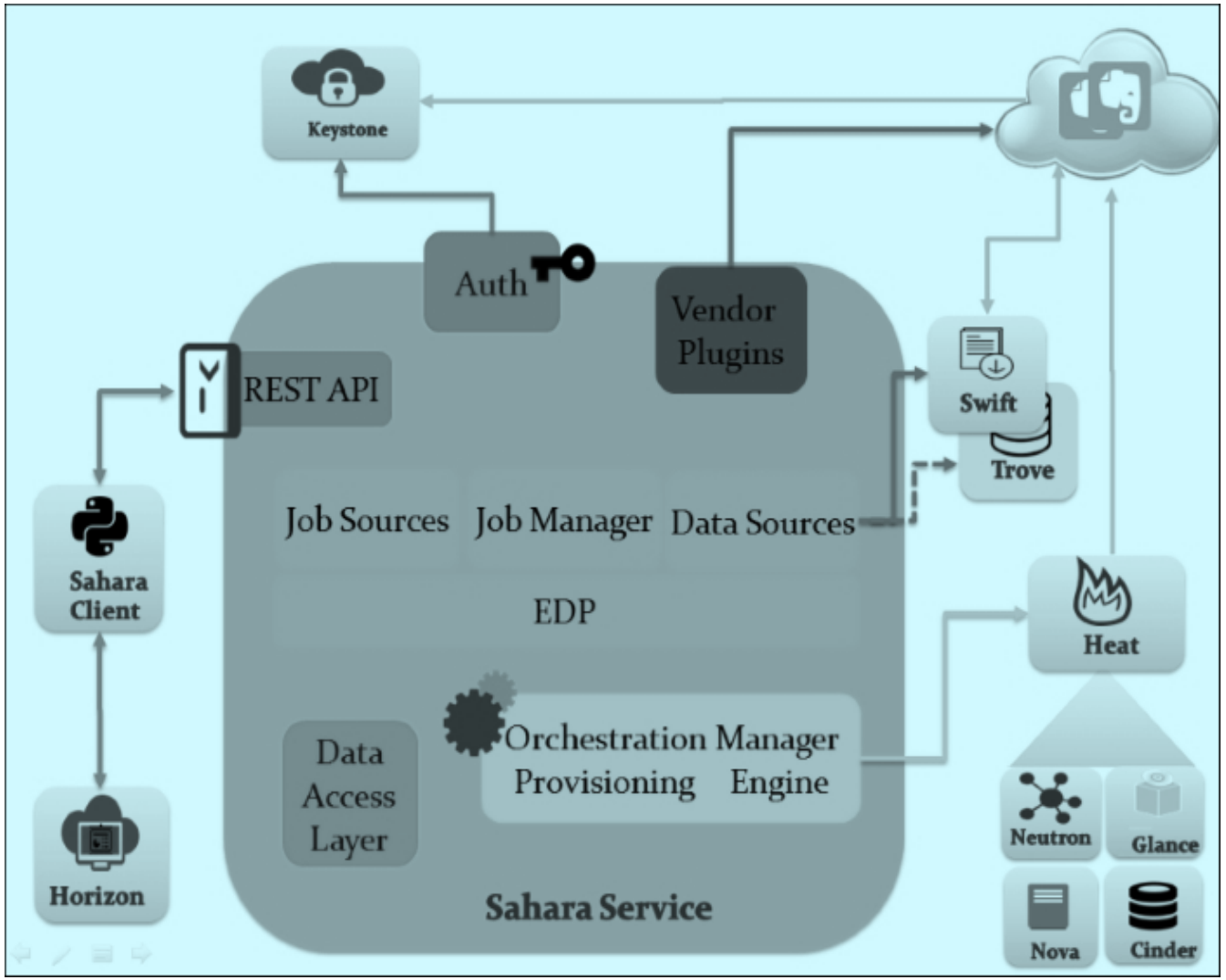
Cinder(Block Storage) – provides a persistent storage media for cluster nodes.

Swift(Object Storage) – provides a reliable storage to keep job binaries and the data processed by Hadoop/spark.

Ceilometer(Telemetry as a Service) – collects and stores the metrics about the cluster for metering and monitoring purposes.

Sahara - Openstack

Sahara Architecture



Auth component - responsible for client authentication & authorization, communicates with the OpenStack Identity service (keystone).
DAL - Data Access Layer: Persistent internal sahara data store.
Provisioning Engine - It instructs the **Heat** engine. Component responsible for communication with the OpenStack Compute (nova), Orchestration (heat), Block Storage (cinder), Image (glance), and DNS (designate) services.
Vendor Plugins - pluggable mechanism that exposes the type of cluster to be launched.
Elastic Data Processing (EDP*)* - Enables the running of jobs on an existing and launched Hadoop or Spark cluster in Sahara
REST API - exposes sahara functionality via REST HTTP interface.
Python Sahara Client - like other OpenStack components, sahara has its own python client.

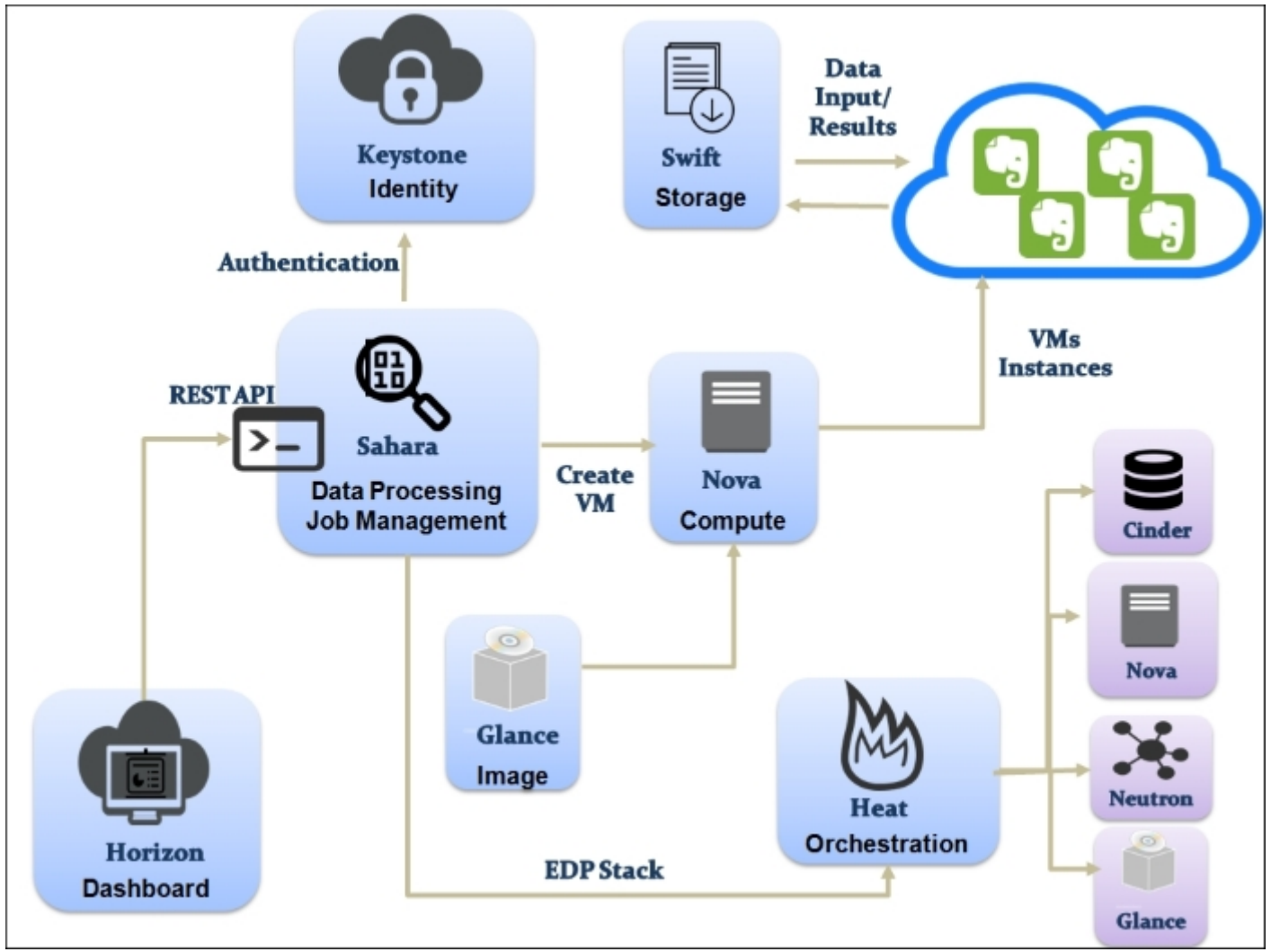
Sahara Features!!!

- Sahara – Data Processing as a Service.
- Provision and manage data processing frameworks such as Hadoop MapReduce, spark and storm in a cluster topology.
- Add-on component to Openstack that lets administrators deploy Spark and Hadoop on top of those VMs

TOP OPENSTACK SAHARA FEATURES:

- It is native to OpenStack and Completely opensource
- Like other OpenStack services, Sahara service can be managed through a powerful REST API, CLI and Horizon dashboard.
- It provides predefined configurable templates to install a Hadoop cluster in a few minutes just by specifying several parameters
- Plugins available to support multiple Hadoop vendor such as Vannila (Apache Hadoop), HDP(ambari), CDH(Cloudera), MapR, Spark, Storm.
- Cluster size can be scaled up and down based on demand.
- It can be integrated with OpenStack Swift to store the data processed by Hadoop and Spark.
- Apart from cluster provisioning, Sahara can be used as analytics as-a-service for ad-hoc or burst analytic workloads.
- Sahara supports different types of jobs: MapReduce, Hive, Pig and Oozie workflows. The data could be taken from various sources: Swift, HDFS, NoSQL and SQL databases. It also supports various provisioning plugins.

Sahara workflow for Hadoop



- Horizon Dashboard or the OpenStack command-line interface is used to access and manage big data resources
- Sahara retrieved and registered virtual machine images in its own image registry by contacting Glance
- Nova provisioned and launched virtual machines for the Hadoop cluster
- We used Heat component in order to automate the deployment of a Hadoop cluster

Sahara deployment – Deep dive

Our Installation of Sahara in Openstack environment

1. Package installation
   # yum install openstack-sahara

2. Create database
   MariaDB [(none)]> CREATE DATABASE sahara;
   MariaDB [(none)]> GRANT ALL ON sahara. TO 'sahara'@'%'*
   IDENTIFIED BY 'sahara';
   MariaDB [(none)]> flush privileges;

3. Update database schemas
   # sahara-db-manage –config-file /etc/sahara/sahara.conf upgrade head

4. Create Sahara service
   # keystone service-create --name=Sahara --type=Data-Processing --description="Elastic Data Processing“

5. Create endpoints for sahara service
   # keystone endpoint-create \
   --service sahara \
   --publicurl "http:// <ip>:8386/v1.1/%(tenant_id)s" \
   --adminurl "http:// <ip>:8386/v1.1/%(tenant_id)s" \
   --internalurl "http:// <ip>:8386/v1.1/%(tenant_id)s“

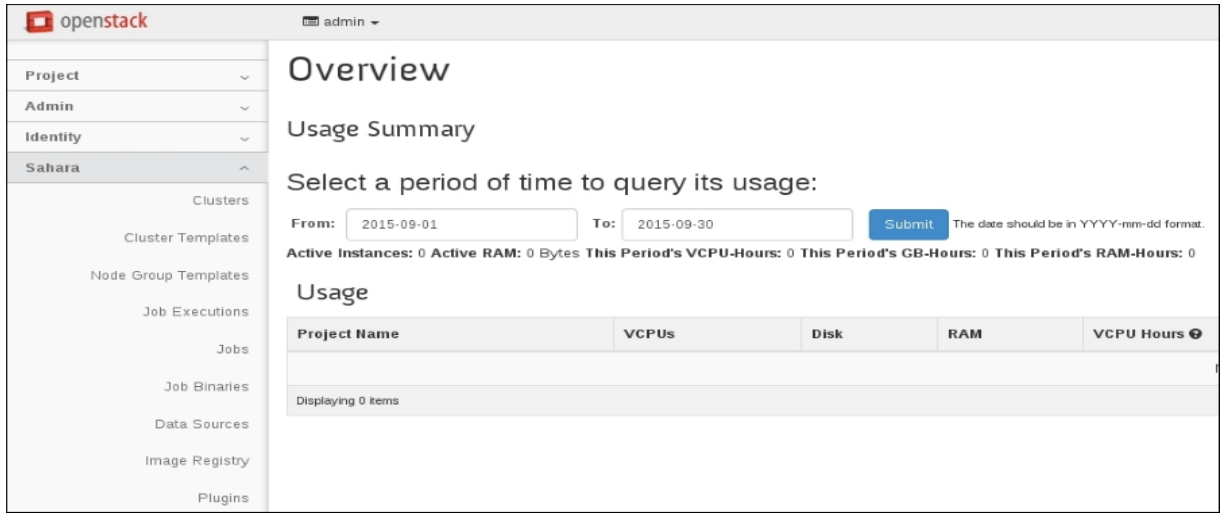
6. Enable it in horizon dashboard

   a) Package installation
   # yum install python-django-sahara –y

   b) Configuration changes in horizon files
   /usr/share/openstack-dashboard/openstack_dashboard/settings.py:
   HORIZON_CONFIG = {
   'dashboards': ('nova','syspanel','settings',...,'sahara')
   INSTALLED_APPS = (
   'saharadashboard',
   SAHARA_USE_NEUTRON=True
   SAHARA_URL='http://10.10.10.47:8386/v1.1’

   c) Restart httpd
   # systemctl restart httpd
   # systemctl start openstack-sahara-api

Sahara Dashboard



Hardware virtualization on computing nodes -Our Recommendation

To enable the hardware virtualization for x86 based systems, or to check that it is already enabled, follow these steps:

1. Check the output of the following command:
   # grep -oE 'svm|vmx' /proc/cpuinfo | uniq

2. Check that your hypervisor supports the hardware virtualization. For KVM, you can use the following command:
   # lsmod | grep kvm
   For KVM -'kvm-ok' tool
   $ kvm-ok
   INFO: /dev/kvm exists

KVM acceleration can be used

1. For KVM, check that OpenStack Compute service (Nova) use KVM as a hypervisor. The '/etc/nova/nova.conf' file should contain the following options:
   compute_driver = libvirt.LibvirtDriver
   [libvirt]
   virt_type = kvm
2. restart nova service

Our Hadoop deployment on VMs using OpenStack

Manual deployment (We faced multiple challenges)

Manual deployment of Hadoop in Openstack is time-consuming, which involves the below steps.

✓ Create instances – Image Creation, Launch instances

✓ Install Hadoop on each instance – Create data node, Worker node etc

✓ Required configuration changes

✓ For different frameworks like Hadoop, spark etc, we need to configure separately

Also, it involves manual interventions during re-creation of clusters

Sahara deployment

✓ We used OpenStack Sahara pre-provisioned images with an installed OS.

✓ Uploaded the image into OpenStack Image Service (Glance)

✓ OpenStack Sahara uses plugins(vannila in our case) for provisioning a data processing cluster. Registered the uploaded image with the corresponding plugin

✓ Created a new Hadoop cluster template (or you can edit the existing one) and launch it

Additional Benefits

✓ We used OpenStack Sahara to scale an Apache Hadoop cluster

✓ Very simple to add and remove nodes using a convenient interface.

Planning a Hadoop deployment

Hadoop cluster defines a set of networked nodes or instances that work together.

The building blocks in Hadoop clustering terminology can be categorized as Master and Slaves nodes.

Each category will run a specific number of daemons to store data and run parallel computations tasks on all this data (MapReduce). The cluster nodes can be

- **NameNode**: This is the orchestrator and centerpiece of the Hadoop cluster, which stores the filesystem metadata. **JobTracker**: This is where the parallel data processing occurs using MapReduce.
- **DataNode**: This role can be assigned to the majority of Slave nodes that present the horse-worker of the cluster. It might be presumed that a data node daemon is a slave to the name node.
- **TaskTracker**: Like data node, this role can be assigned to a slave node as well. A task tracker daemon is a slave to the job tracker node.

Automate the building of distributed Hadoop using Sahara CLI

Step 1: Created a Node Group Template using CLI

CLI to create and update Node Group Templates, which simply need the correct file syntax in JSON format
We created a master node group template, for this created a new file called master_template_pp.json with the following content:

```
# nano master_template_pp.json
{
  "name": "Master-PP-Template",
  "flavor_id": "2",
  "plugin_name": "vanilla",
  "hadoop_version": "2.7.1",
  "node_processes": ["namenode", "secondarynamenode" ,"resourcemanager", "historyserver",
"oozie"],
  "auto_security_group": true
}
```

```
# sahara node-group-template-create --json master_template_pp.json
```


Step 2: Created a worker Node Group Template using CLI

Created a worker node group template, created a new file called worker_template_pp.json with the following content:

```
# nano worker_template_pp.json
{
 " name": "PP-Worker-Template",
  "flavor_id": "2",
  "plugin_name": "vanilla",
  "hadoop_version": "2.7.1",
  "node_processes": ["nodemanager", "datanode"],
  "auto_security_group": true
}
```

Uploaded the new template file using Sahara CLI using the following,

sahara node-group-template-create –-json worker_template_pp.json

Step 3: Created a Node Cluster Template using CLI

Using the Sahara CLI, created a Node Cluster Template. Created a new file called master_template_pp.json with the following content:

```
# nano pp-hadoop-cluster-template.json
{
    "name": "pp- hadoop-cluster-template",
    "plugin_name": "vanilla",
    "hadoop_version": "2.7.1",
    "node_groups": [
        {
            "name": "master",
            "node_group_template_id": "53ed88e9-9f7f-43ab-a595-73a00a2ed630",
            "count": 1
        },
    {
            "name": "workers",
            "node_group_template_id": "2c0ba50f-5e93-4948-a4c9-39f29ef372ed",
            "count": 2
        }
    ]
}
```

Next, uploaded the new template file using the Sahara CLI as the following:

```
# sahara node-group-template-create --json
master_template_pp.json
```


Step 4: Launched our Hadoop Cluster

Launched the cluster

```
# nano pp_launch_cluster.json
 {
   "name": "pp-launch-cluster",
   "plugin_name": "vanilla",
   "hadoop_version": "2.7.1",
   "cluster_template_id" : "60524c6c-bea0-4065-bb7a-56f24ed91357",
   "default_image_id": "49fa54c0-18c0-4292-aa61-fa1a56dbfd24"
   "neutron_management_network": "4f9d08d2-80d5-485a-a601-e4135caf2eda"
 }
```

```
# sahara cluster-create --json pp_launch_cluster.json
```


Job execution- Requirements

Requirements to execute jobs in Sahara

Data Source

- Data localization: Path of input/output data
  - Swift: OpenStack object storage
  - HDFS: Native Hadoop Distributed File System storage
  - Manila: Network filesystem shares in OpenStack

- Code: Defines which code will be executed and run

Job binaries in Sahara

Scripts, JAR files, and file credentials will be referenced stores in Sahara's own database, or externally using Swift or Manila. The paths of scripts and JAR files are referenced by means of the URL

	Storage type	Access	Authentication
Internal	Sahara database	Sahara cluster	Sahara credentials
Swift	Object storage	Swift proxy	Swift credentials
Manila	File-shared storage	Mounted shares	Share access control

Job types in Sahara:

- Java
- Hive
- MapReduce
- MapReduce.Streaming
- Pig.
- Shell.
- Spark.

Our executed jobs via Horizon UI-Case study

Executing jobs via Horizon

We tested to neaten a simple text file by trimming and removing space in each line. The sample text file looks like the following:

- OpenStack
- SDC
- Sahara
- Swift
- Jobs

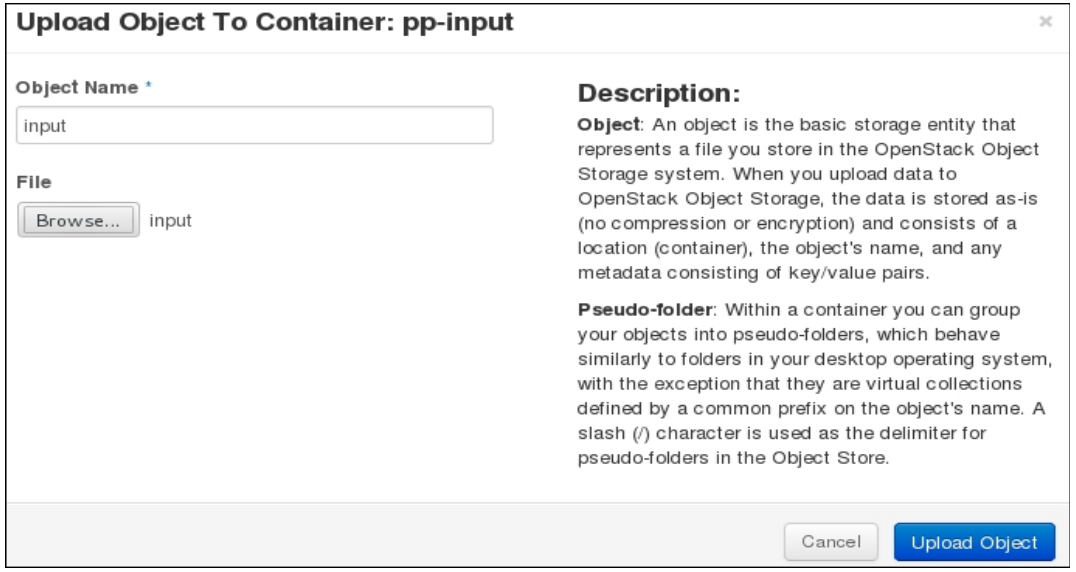
To do so, we executed a Pig Job in the Sahara cluster and designate the location of the text file in Swift named input.
The Pig script follows:

```
I = load '$INPUT' using PigStorage(':') as (cloud: chararray);
O = foreach I generate com.hadoopbook.pig.Trim(cloud);
store O into '$OUTPUT' USING PigStorage();
```


Executing jobs via Horizon- Cont…

1. Under the Object Store tab, we selected Containers:

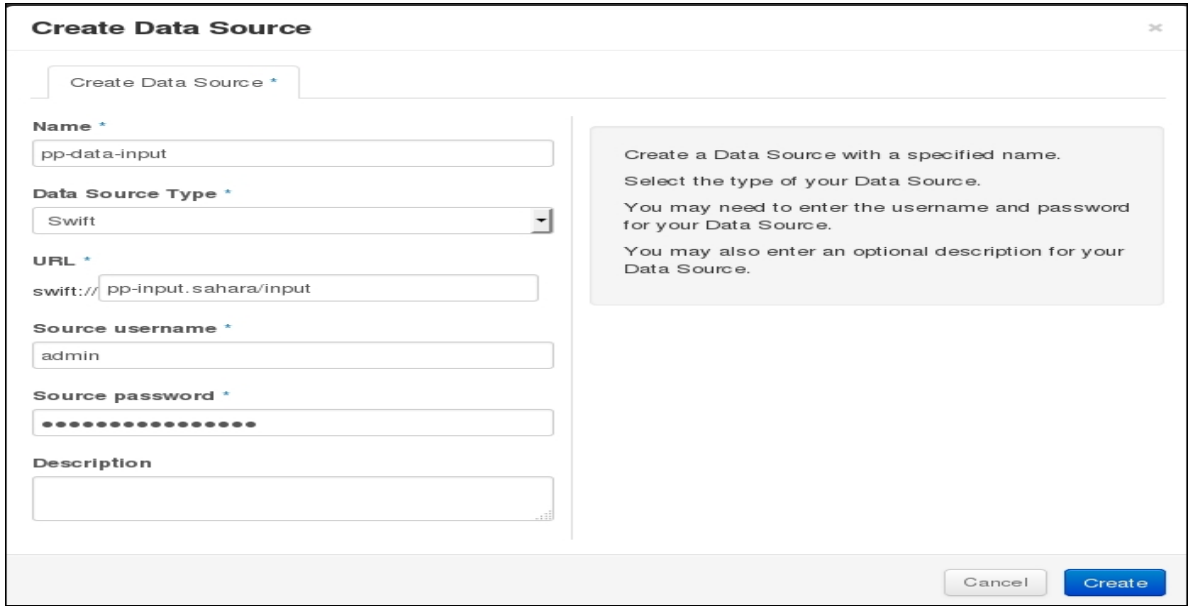
2. Created a new container in Swift and name it pp-input
3. We uploaded the input data



Executing jobs via Horizon- Cont…

4. Input file is uploaded in Swift:

5. Next, we informed sahara to take the data source from swift:

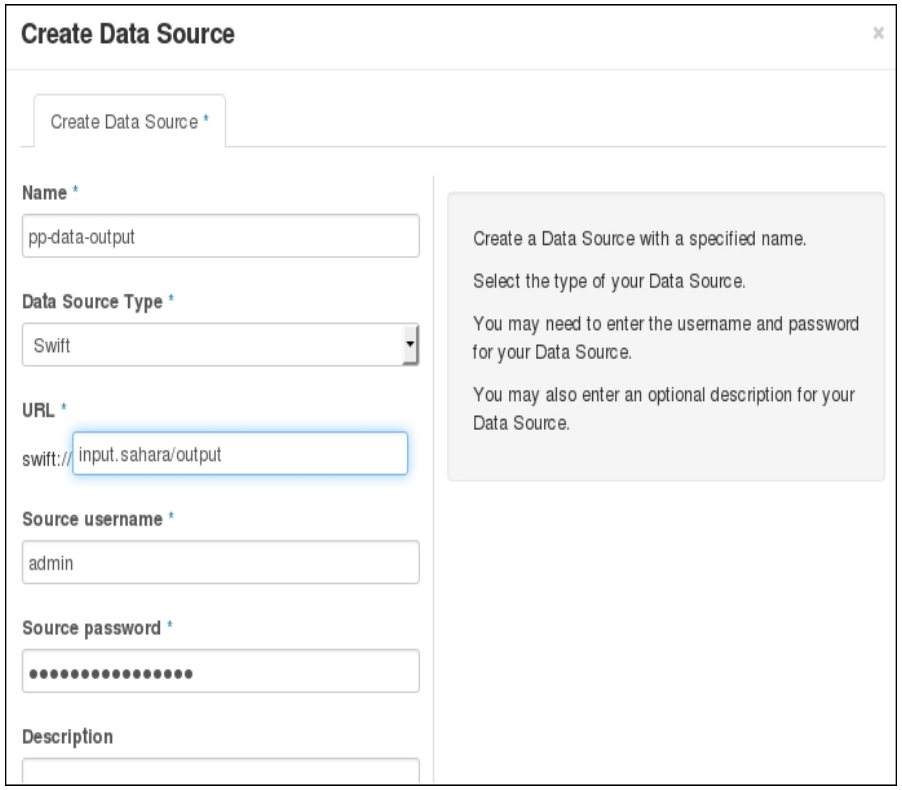


Note

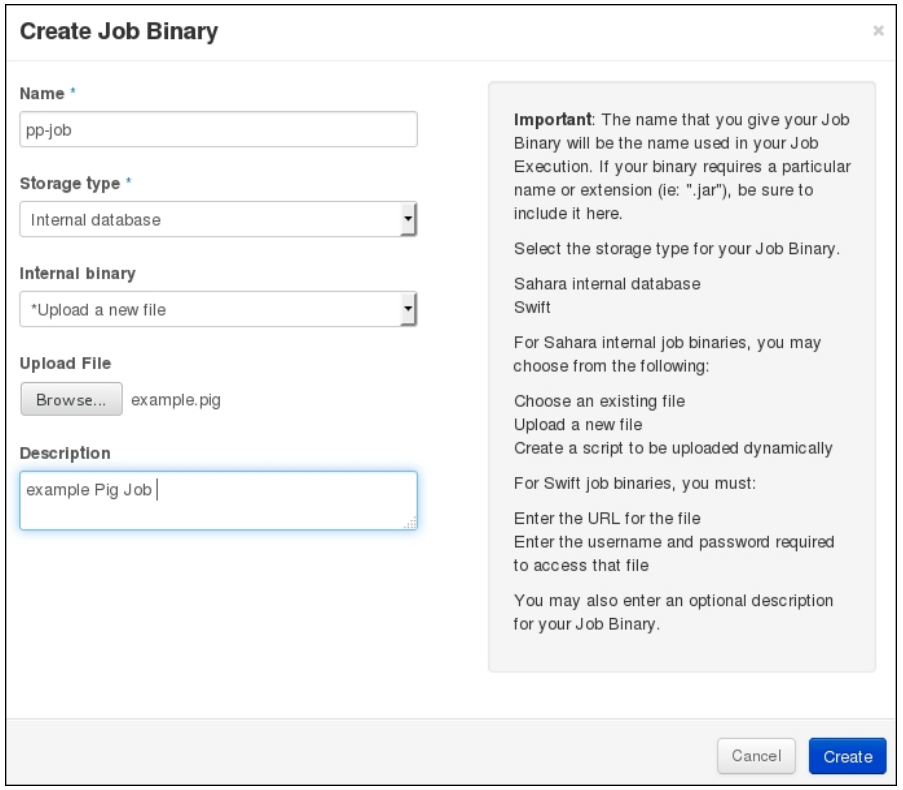
- The Swift URL has the following format: swift://<container_name.sahara>/<object_name>
- The Sahara Swift URLs might include a .sahara suffix.

Executing jobs via Horizon- Cont…

6. We created another data source for storing output

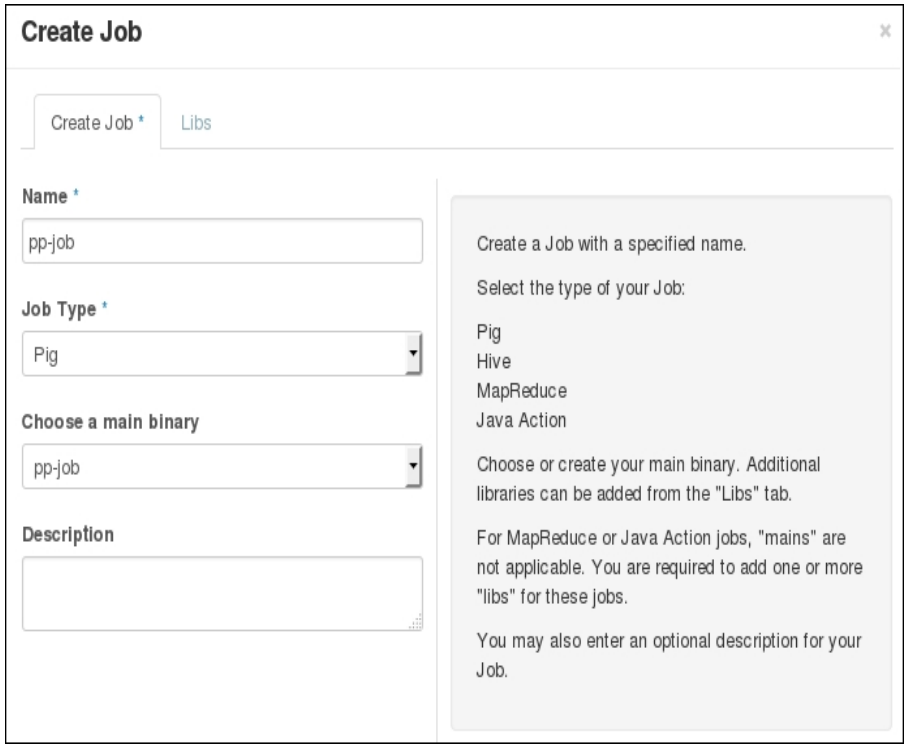


7. Next, We created a job binary for pig job

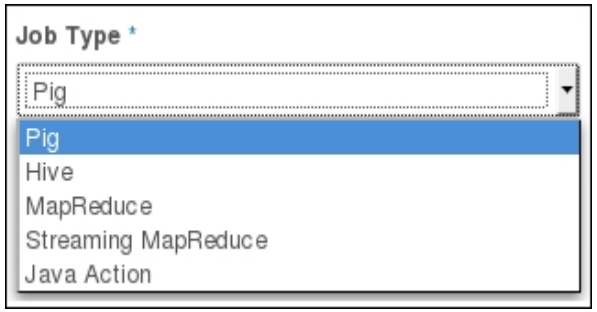


Executing jobs via Horizon- Cont…

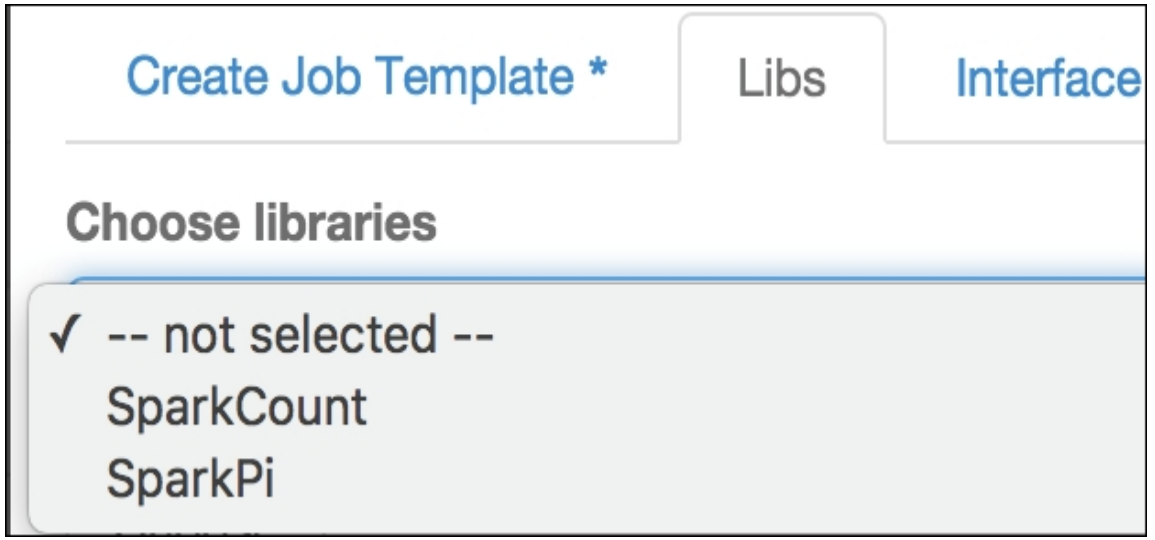
8. Created a pig job



9. Note that Sahara provides different job types from the drop-down menu list:

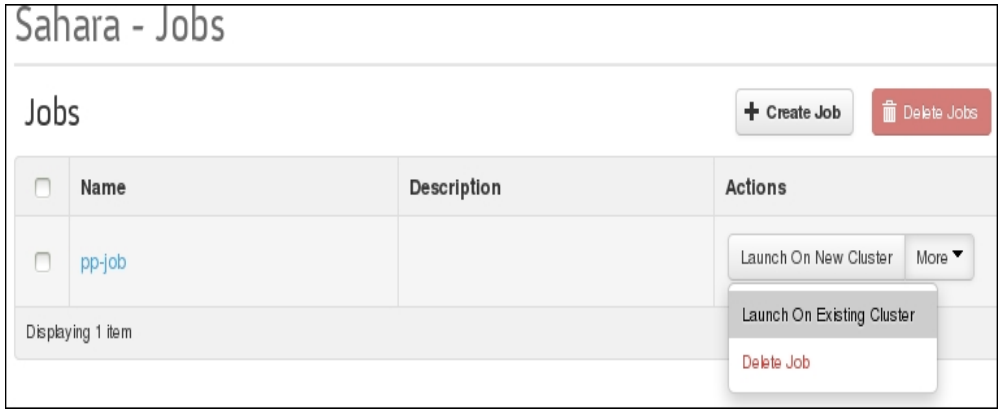


10. Optionally, it might be possible to upload additional libraries

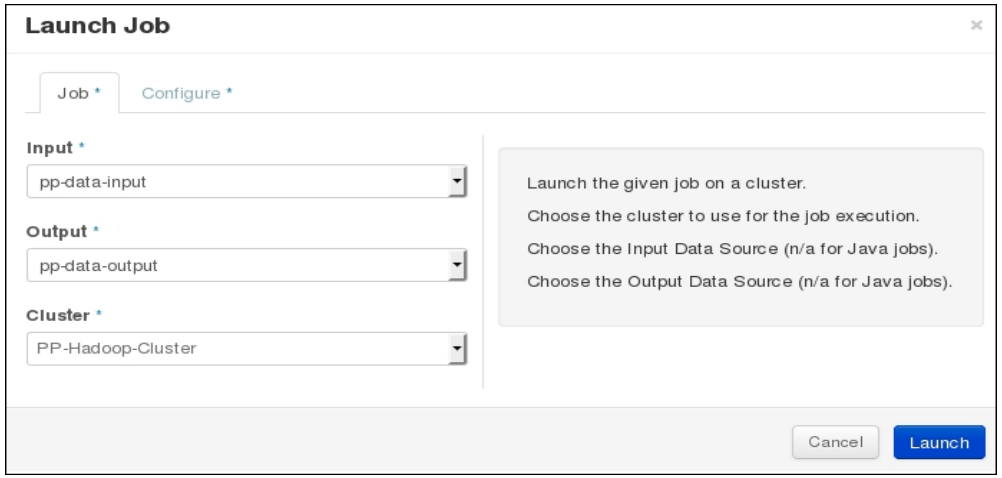


Executing jobs via Horizon- Cont…

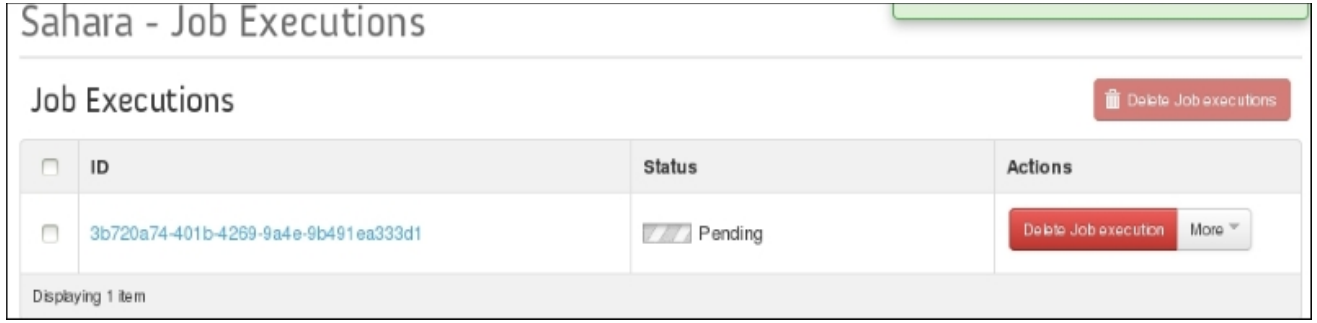
11. We launched the created job



12. We selected input, output and cluster for the job



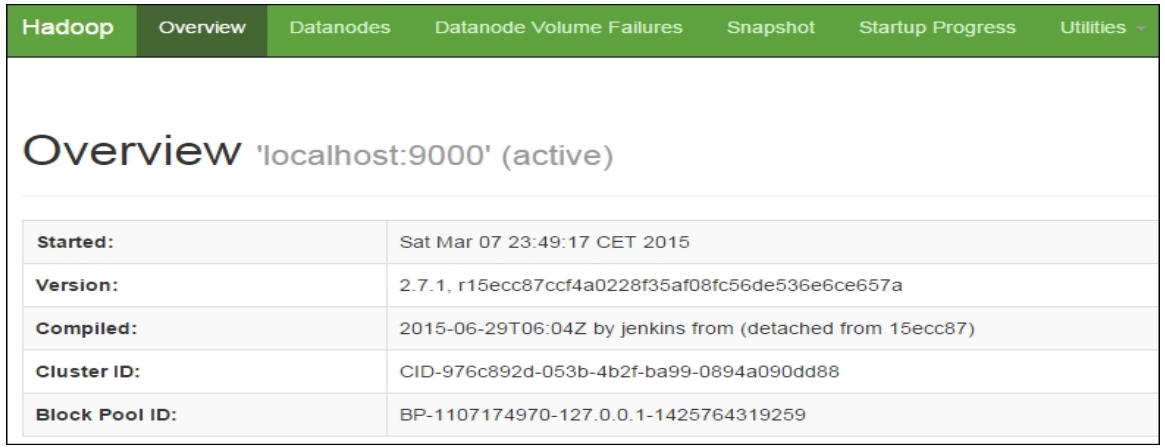
13. We Launched the cluster and job get executed on the specified Hadoop cluster. The job execution progress shows the current status of the job



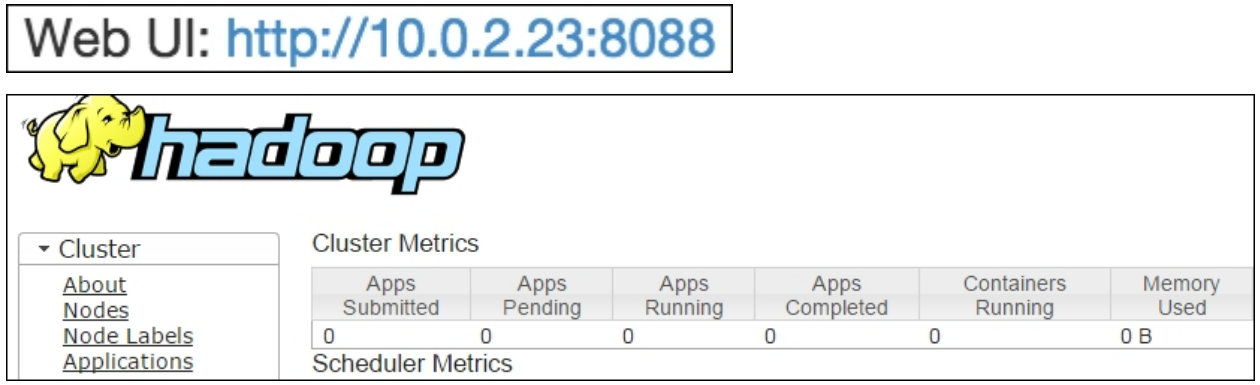
14. The job took a while in the Hadoop cluster by changing its progress state from Pending to Running. To collect results from the Swift output, the job progress should end up with a Succeeded status. An additional cluster details check can be performed by browsing to the Clusters tab in Sahara. In our case, HDFS web user interface is accessed on the Hadoop master node on port 50070.

Executing jobs via Horizon- Cont…

15.From the same cluster information tab, it is possible to access the HDFS NameNode web user interface



16.Accessing the YARN web user interface provides more details on running jobs in the Hadoop cluster, in our case Oozie:

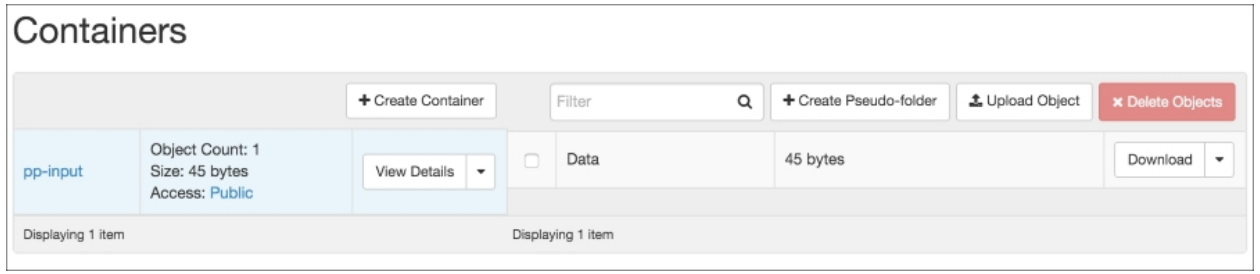


Executing jobs via Horizon- Cont…

17.The job execution ends with a Succeeded status as follows:

	Status
	Succeeded

18.We downloaded data from the Container output in Swift:



Output

- OpenStack
- SDC
- Sahar
- Swift
- Jobs

About the Speaker

Ajitha Robert

Ajitha is a Dynamic Cloud Stack Engineer with 8 Years of progressive IT Experience in Cloud Service Development. She has worked on assignments involving implementation and testing with Technical expertise for clients in Delivering Optimal Results across multiple domains like Security Auditing, Cloud Technologies, DevOps and Storage. At MSys, She has been working on Openstack Cinder driver and CI maintenance. She has good exposure in Openstack cloud, Ceph storage, java development and DevOps tools.

Please take a moment to rate this session.

Your feedback is important to us.

