STORAGE DEVELOPER CONFERENCE



Virtual Conference September 28-29, 2021

We Addressed the Elephant in the Room with OpenStack Sahara

Ajitha Robert MSys Technologies, LLC A SNIA, Event



PRODUCT ENGINEERING SERVICES AND DIGITAL TRANSFORMATION PARTNER

BAY AREA UNICORNS

'TOP FASTEST GROWING STORAGE COMPANIES' FOR TWO CONSECUTIVE YEARS

OUR WW STRENGTH 1000+ AND GROWING





Storage 5

Newsletter

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



4 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

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







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



7 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

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 commandline 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



11 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

Our Installation of Sahara in Openstack environment b) Configuration changes in horizon files

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;

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

🧧 opensta	ick	📼 admin 👻					
Project	~	Overview					
Admin	~						
Identity	~	Usage Summary					
Sahara	^	Select a period	of time to	auen ite usaa	0.		
	Clusters	select a period of time to query its usage:					
Cluster Templates		From: 2015-09-01	To:	2015-09-30	Submit	The date should be in 1	YYYY-mm-dd format.
blada	Course Templetes	Active Instances: 0 Active RAM: 0 Bytes This Period's VCPU-Hours: 0 This Period's GB-Hours: 0 This Period's RAM-Hours: 0					
14006	Group Templates	Usage					
	Job Executions	Project Name		VCPUs	Disk	RAM	VCPU Hours @
	Jobs	1 rejour name		10100	DIOR		
	Job Binaries						1
	Data Sources	Displaying 0 items					
	Incore Deviate -						
	image Registry						
	Plugins						



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:

Ismod | grep kvm For KVM -'kvm-ok' tool \$ kvm-ok INFO: /dev/kvm exists

KVM acceleration can be used

 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
- \checkmark 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



16 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

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



21 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

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	:	Pig. Shell.
•	MapReduce MapReduce.Streaming	•	Spark.



Our executed jobs via Horizon UI-Case study



23 | ©2021 Storage Developer Conference ©. MSys Technologies, LLC. All Rights Reserved.

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();



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

	penstack admin -					
Ρ	roject	•	Containers			
Compute +		+	Containers		+ Create Container	Objects
	Network	- F				
	Object Store	-		No items to display.		
Containant			Displaying 0 items			Displaying 0 items
	Containers	_				

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

Upload Object To Container: pp-input	×		
Object Name * input	Description: Object: An object is the basic storage entity that represents a file you store in the OpenStack Object		
File Browse input	Storage system. When you upload data to OpenStack Object Storage, the data is stored as-is (no compression or encryption) and consists of a location (container), the object's name, and any metadata consisting of key/value pairs.		
	Pseudo-folder : Within a container you can group your objects into pseudo-folders, which behave similarly to folders in your desktop operating system, with the exception that they are virtual collections defined by a common prefix on the object's name. A slash (<i>i</i>) character is used as the delimiter for pseudo-folders in the Object Store.		
	Cancel Upload Object		



4. Input file is uploaded in Swift:

Containers							
Containers			+ Create Container	Ob	jects	Filt	9ľ
pp-input	Object Count: 1 Size: 45 bytes Access: Private	View Details	More 👻		input		45 bytes
Displaying 1 item				Display	ying 1 ilem		

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

Create Data Source	24
Create Data Source *	
Name *	
pp-data-input	Create a Data Source with a specified name.
Data Source Type * Swift	Select the type of your Data Source. You may need to enter the username and password for your Data Source.
URL * swift:// pp-input.sahara/input	You may also enter an optional description for your Data Source.
Source username *	
admin	
Source password *	
•••••	
Description	
	Cancel Create

Note

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



×

6. We created another data source for storing output

Create Data Source	
Create Data Source *	
Name *	
pp-data-output	Create a Data Source with a specified name.
Data Sauraa Tura t	Select the type of your Data Source.
Swift	You may need to enter the username and password for your Data Source.
URL *	You may also enter an optional description for your Data Source.
swift://_Input.sanara/output	
Source username *	
admin	
Source password *	
•••••	
Description	

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

Create Job Binary	×
Name *	
pp-job Storage type *	Important : The name that you give your Job Binary will be the name used in your Job Execution. If your binary requires a particular name or extension (ie: ".jar"), be sure to
Internal database	Include it here.
Internal binary *Upload a new file	Select the storage type for your Job Binary. Sahara internal database Swift
Upload File	For Sahara internal job binaries, you may choose from the following:
Browse example.pig	Choose an existing file Upload a new file Create a script to be uploaded dynamically
example Pig Job	For Swift job binaries, you must:
aii	Enter the URL for the file Enter the username and password required to access that file
	You may also enter an optional description for your Job Binary.
	Cancel



8. Created a pig job

Create Job	
Create Job * Libs	
Name *	
pp-job	Create a Job with a specified name.
loh Type *	Select the type of your Job:
	Pig
1 19	Hive MapReduce
Choose a main binary	Java Action
pp-job •	Choose or create your main binary. Additional libraries can be added from the "Libs" tab.
Description	For MapReduce or Java Action jobs, "mains" are not applicable. You are required to add one or more "libs" for these jobs.
E.	You may also enter an optional description for your Job.

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

Job Type *
Pig
Pig
Hive
MapReduce
Streaming MapReduce
Java Action

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





11. We launched the created job

Sah	ara - Jobs		
Job	S		+ Create Job
	Name	Description	Actions
	pp-job		Launch On New Cluster More
Display	ring 1 item	Launch On Existing Cluster	
			Delete Job

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

Launch Job	ж
Job * Configure *	
p-data-input Dutput pp-data-output	Launch the given job on a cluster. Choose the cluster to use for the job execution. Choose the Input Data Source (n/a for Java jobs). Choose the Output Data Source (n/a for Java jobs).
Cluster * PP-Hadoop-Cluster	
	Cancel

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

Sahara - Job Executions					
Job Executions					
	ID	Status	Actions		
	3b720a74-401b-4269-9a4e-9b491ea333d1	Pending	Delete Job execution More *		
Displaying 1 item					

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.

HDFS

Web UI: http://10.0.2.23:50070



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

Hadoop	doop Overview Datanodes		Datanode Volume Failures	Snapshot	Startup Progress	Utilities -		
Overview 'localhost:9000' (active)								
Started:	Sat Mar 07 23:49:17 CET 2015							
Version:		2	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a					
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)							
Cluster ID:	:	C	CID-976c892d-053b-4b2f-ba99-0894a090dd88					
Block Poo	I ID:	В	P-1107174970-127.0.0.1-1425	764319259				

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

Web UI: http://10.0.2.23:8088





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

Status	
Succeeded	

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

Containers						
		+ Create Container		Filter Q	+ Create Pseudo-folder	t X Delete Objects
pp-input	Object Count: 1 Size: 45 bytes Access: Public	View Details 👻		Data	45 bytes	Download -
Displaying 1 item			Displa	ying 1 item		

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.

