**Automated Report Generation for OPS Dashboard**

## **Overview:**

* Create an Operations report dashboard to allow MOC staff to view daily status about activity in OpenShift & Openstack clusters.
* Reports are snapshots of activity on a statically configured list of servers (OpenStack) and OpenShift Cluster metrics.
* The reporting period is configurable, and defaults to three times every day.
* Each report will identify pods that are active, running and have failed
* Gather Red Hat OpenShift cluster metrics on each configured pod, node and project
* Get the usage statistics for servers and resources (e.g. memory, virtual CPUs) of each active project (tenant) on OpenStack
* Display the information collected in the form of visualizations
* Use OpenStack object storage to store the gathered metrics

## **Prerequisites:**

* Should be able to login to <https://kaizen.massopen.cloud/dashboard/auth/login/>
* Should be able to use OpenStack CLI
* Make sure ‘OpenStack server list’ is working properly
* Should have k-OpenShift account
* Should have Access to OpenStack Compute API
* Link to add users to the OSP <https://osticket.massopen.cloud/kb/faq.php?id=22>
* Should be able to download Ec2 credentials from onboarding.mass open.cloud.
* Have basic understanding of <https://osticket.massopen.cloud/kb/faq.php?id=24>
* Have basic understanding of <https://osticket.massopen.cloud/kb/faq.php?id=16>
* Node js should be installed in the development and staging environment where React is running (the VM running the user dashboards), React npm modules should also be configured in the same environment.
* Ensure that hello world React application runs on your operating system
* Ensure that pip runs on your operating system

## **List of dependencies**

* Python OpenShift client
* OpenStack keystone client

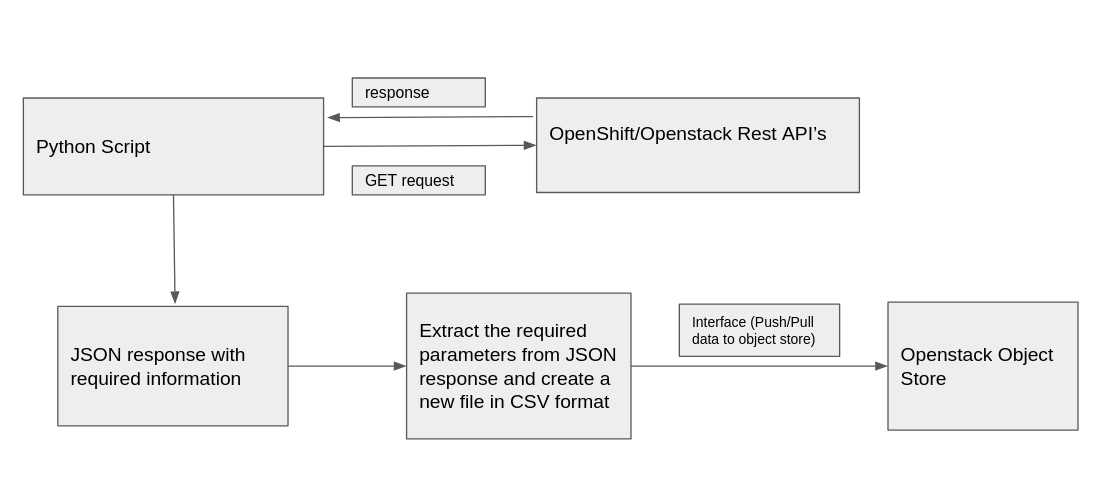
**Methodology:**

The whole process of automated report generation is segmented into 3 steps:

* Data Generation
* Data Collection
* Dashboard Building

### **Data Generation**

A ***cron job*** will run 3 times a day, triggering a python script that queries the corresponding APIs for the MOC***Kaizen-OpenStack* & *OpenShift cluster***).



#### **OpenShift Data Generation**

To get authorisation to the OpenShift master API server

* ***Kubeconfig file*** to execute outside the cluster
* We will create a ***Service Account*** to execute

[https://docs.OpenShift.com/container-platform/3.11/welcome/index.html](https://docs.openshift.com/container-platform/3.11/welcome/index.html)

Authorization: Bearer $TOKEN

Accept: application/json

Connection: close

OpenShift API (Data to be Collected)

* To get Node Data

[https://$ENDPOINT/api/v1/nodes/$NAME](about:blank)

* To get Pod Data

GET /api/v1/namespaces/$NAMESPACE/pods/$NAME HTTP/1.1

* To list all Builds

GET /apis/build.OpenShift.io/v1/builds HTTP/1.1

* To list all Routes

GET /apis/route.OpenShift.io/v1/routes HTTP/1.1

#### **OpenStack Data Generation**

OpenStack Compute API version 2.73(Data to be collected)

### To get server usage details for each tenant running on OpenStack *Compute API:*

GET https://<compute\_api\_url>/os-simple-tenant-usage/details

To see more details about the tenant usage statistics, go [here](https://docs.openstack.org/api-ref/compute/?expanded=list-tenant-usage-statistics-for-all-tenants-detail#list-tenant-usage-statistics-for-all-tenants)

* To list of all the servers running on OpenStack:

GET https://<compute\_api\_url>/servers/details

For each server, shows server details including config drive, extended status, and server usage information.

The extended status information appears in the OS-EXT-STS:vm\_state, OS-EXT-STS:power\_state, and OS-EXT-STS:task\_state attributes.

The server usage information appears in the OS-SRV-USG:launched\_at and OS-SRV-USG:terminated\_at attributes.

* Filtered API query to extract specific details:

GET https://<compute\_api\_url>/os-simple-tenant-usage/{tenant\_id}?marker={current\_date}

It is necessary to filter the data that this API query returns, because it is more data than we want to report. We process the JSON response received and extract only the corresponding values for the following parameters for each tenant:

* Tenant\_id
* Tenant.start\_date
* tenant\_stop\_date
* Server\_uuid
* Server.status
* Server.start\_date
* Server.stop\_date
* Instance\_name
* Flavor
* server.Memory\_usage
* server.uptime
* tenant.uptime
* Tenant.vcpus
* Server.vcpus

Save the new object created to a file in CSV format. Push the csv file to the object store created on OpenStack to the corresponding folder.

## **Data Collection**

A python script will act as an interface to push and pull the data (Collected Metrics) from the OpenStack object store.

When Push is used : Push is used to push the metrics collected to the OpenStack object store

When Pull is used : Pull is used to pull the metrics collected from the OpenStack object store, to build the dashboards.

Why & What Object Store?

The Object Store is an interface where users can upload data directly to our Ceph cluster for storage.

**OpenStack object cli commands**

* Object Creation Command

OpenStack object create [--name <name>] <container> <filename> [<filename> ...]

* Object Delete Command

OpenStack object delete <container> <object> [<object> ...]

* Object Save Command

OpenStack object create [--name <name>] <filename> [<filename> ...] <container> object

* Object show Command

OpenStack object show <container> object

To Download an uploaded File

*curl -o local\_file.txt* [*http://rdgw.kaizen.massopencloud.org/swift/v1/tutorial\_container\_unique/test\_file*](http://rdgw.kaizen.massopencloud.org/swift/v1/tutorial_container_unique/test_file)

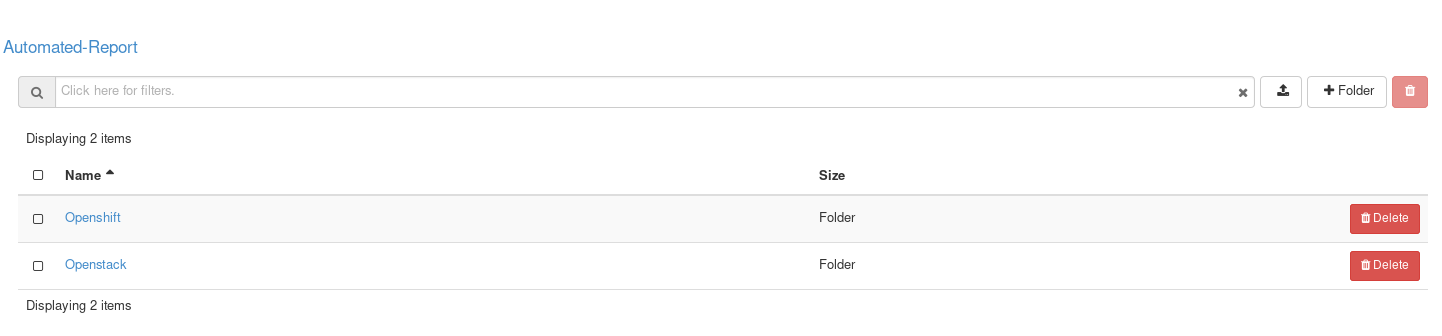
Once I have uploaded the file on to the object store, other new users can use curl to download the file.

Below is the folder structure for how we will store the data

Two Folders

1.) OpenShift

2.) OpenStack

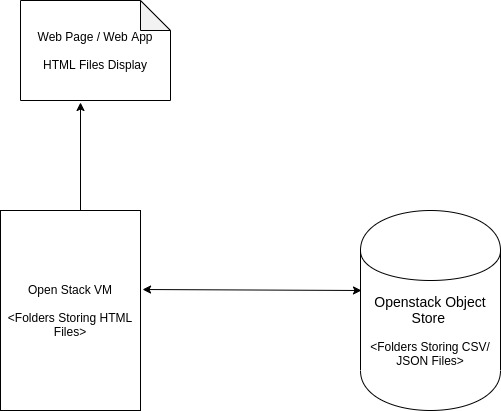


Sub Folders Inside OpenShift for Cron Job (Morning, Noon & Evening)

***Files inside this folder :*** *File\_name\_YY\_MM\_DD.csv*

## **Dashboard Building**

Implementation:

****

OPS Dashboard Diagram

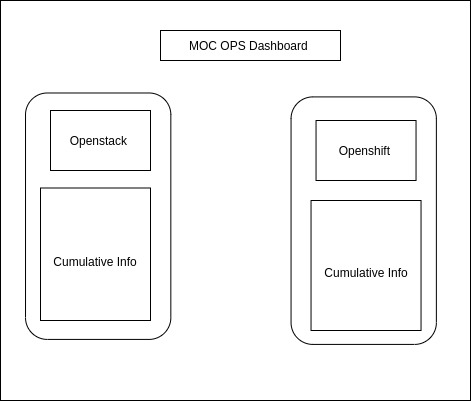
*Note: Apache Service should be installed and running on the VM*

The React Web App has the Javascript and asynchronous functions which will fetch the data from CSV/JSON.

### **Frontend design templates**

This is the Overall Block Diagram of the front end of the design. UI Framework : React

The Webapp will be deployed on the VM and then the dashboard interface will be accessed through the Floating IP. There is a common access link which will be shared and could be accessible by anyone who has it.



The above dashboard contains cumulative data collected from OpenShift and OpenStack

This is the overall OpenStack and OpenStack UI Display Diagram which focuses on the display of the Nodes, Pods and Project Information

