OpenStack Infrastructure Optimization Service

August 2018

AUTHOR:
Mohammed Henni

SUPERVISOR:
Jose Castro Leon

CERN openlab

PROJECT SPECIFICATION

OpenStack Watcher provides a flexible and scalable resource optimization service for multi-tenant OpenStack-based clouds. Watcher provides a complete optimization loop—including everything from a metrics receiver, complex event processor and profiler, optimization processor and an action plan applier. This provides a robust framework to realize a wide range of cloud optimization goals, including the reduction of data center operating costs, increased system performance via intelligent virtual machine migration, increased energy efficiency—and more!

This project aims to investigate the features of OpenStack Watcher, and evaluate how they could be integrated into the CERN Cloud Service.

ABSTRACT

CERN operates an OpenStack based private cloud to provide its users with resources on demand. It is one of the largest OpenStack deployments in the world, with more than 300,000 cores over 9,000 hypervisors [1].

Managing such a large deployment is a very challenging work, and as the need for computing resources grows, the infrastructure is planned to grow accordingly.

One of the main challenges is to optimize and maximize the resource utilization. A lot of effort, such as the work on preemptible virtual machines [2], is being done at CERN to improve that.

This report presents another work done in that scope, which is the integration of a framework for infrastructure optimization for OpenStack, called Watcher.

The report starts by giving an overview of Watcher, describing its architecture and explaining how it works. Then, focus is put on the integration of Watcher for CERN's cloud. Finally, a conclusion summarizes the key points of this project, and what future work could be done to further integrate Watcher at CERN.

TABLE OF CONTENTS

1. Introduction

a. OpenStack

OpenStack is a set of free and open source software that allow the deployment and management of cloud computing infrastructures.

OpenStack consists of many independent components, named the OpenStack services. These services interact with each other through APIs.

OpenStack is backed by some of the largest companies in tech. Among the top contributors to this open source project are Red Hat, HP, IBM, Rackspace, and many others.

Many large companies rely on OpenStack, including AT&T, PayPal, NTT, and of course, CERN.

b. OpenStack at CERN

CERN moved from grid computing to cloud computing in order to efficiently fulfil the computing and storage needs of its users on demand. It has been running OpenStack in production for managing its private cloud since 2013.

Although OpenStack at CERN started with only a few projects (Nova, Glance, Cinder, Keystone), it is now running more than 12 different OpenStack projects in production. Some 90 percent of the CERN resources are delivered on top of OpenStack [1].

c. Motivation for this project

The OpenStack deployment at CERN is one of the largest in the world, with more than 300,000 cores over 9,000 hypervisors [1]. The infrastructure runs across two CERN data centers, one in Geneva and the other one in Budapest, separated by approximately 22 ms.

A cloud environment is very dynamic, as virtual machines are allocated and liberated at a high rate. At CERN, VMs are created/deleted every 10s.

All of this makes it challenging to keep resources' usage optimal. This is the target of this project: try out the OpenStack project for infrastructure optimization, in order to maximize and optimize resource utilization at CERN.

2. OpenStack Watcher

a. Overview

Watcher is the official infrastructure optimization service for OpenStack [3]. It's a scalable framework that provides a pluggable architecture to realize a wide range of optimization goals, such as reducing the energy consumption and increasing system performance [4].

Figure 1. Watcher project mascot [6]

b. Architecture

Watcher has 3 main components, as illustrated in figure 2: decision engine, applier, and the api. The decision engine is responsible for computing the potential optimization actions needed to fulfil a certain goal [5]. The applier is responsible of actually performing those actions on the infrastructure to be optimized.

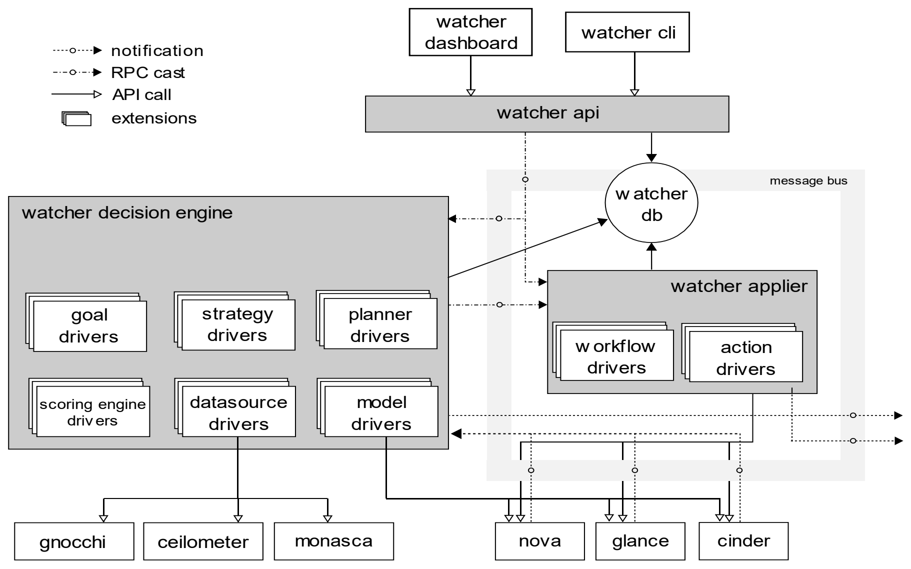


Figure 2. Watcher architecture

Each of these components offers pluggable sub-components so that it can easily be extended. One can for example add new strategies to the decision engine, and new actions to the applier.

Interaction with Watcher is possible through its command line interface, and through its dashboard that integrates with Horizon.

Watcher leverages services provided by other OpenStack projects such as Nova for live migration and Ceilometer for getting metrics.

c. How it works

Watcher performs in an optimization loop (depicted in figure 3). It starts by getting relevant metrics of the infrastructure to optimize from the datasource drivers (figure 1). Then it analyses those metrics to profile virtual machines resource usage.

Then, Watcher's decision engine builds a modal of the infrastructure that describes its state, and tries to compute an optimal equivalent modal, based on specified goals and constrains.

After computing an optimal modal, the decision engine plans the different actions necessary to transition from the current modal to the optimal one, and finally Watcher's applier executes those actions on the infrastructure.

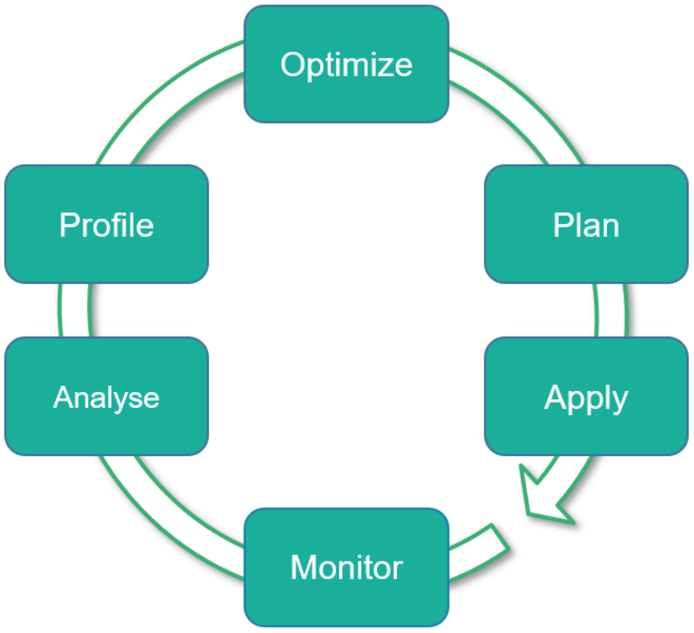


Figure 3. Watcher optimization loop

3. Watcher at CERN

a. Deploying Watcher in CERN's cloud

After trying out Watcher on a Devstack environment, we deployed it in a preproduction environment in the CERN cloud. Deployment steps can be found in [5].

Since we do not want to try things out on the whole CERN infrastructure, we tweaked Watcher to restrict it to the hyperconverged servers[1] environment. The size of the environment to be optimized is 9 servers, hosting 40 virtual machines.

b. Extending Watcher

Out of the box, Watcher comes with a set of optimization strategies, most of which rely on some monitoring metrics, obtained from the datasource drivers (figure 2).

By default, Watcher relies on Ceilometer, Gnocchi or Monasca to retrieve metrics. CERN doesn't use these services for monitoring.

Since the goal is to first try out Watcher on production before fully integrating it, instead of developing a new datasource plugin for CERN monitoring tools, we extended Watcher with a new optimization strategy that doesn't rely on monitoring metrics.

The implemented strategy is illustrated in figure 4. It balances the number of VMs between the servers, and allows the administrator to specify some VMs not to be moved, by tagging them as "critical". The desired result of the strategy is to have the same VM count per server, prioritizing the servers with more "critical" VMs to be less loaded when the VM count is not a multiple of servers count.

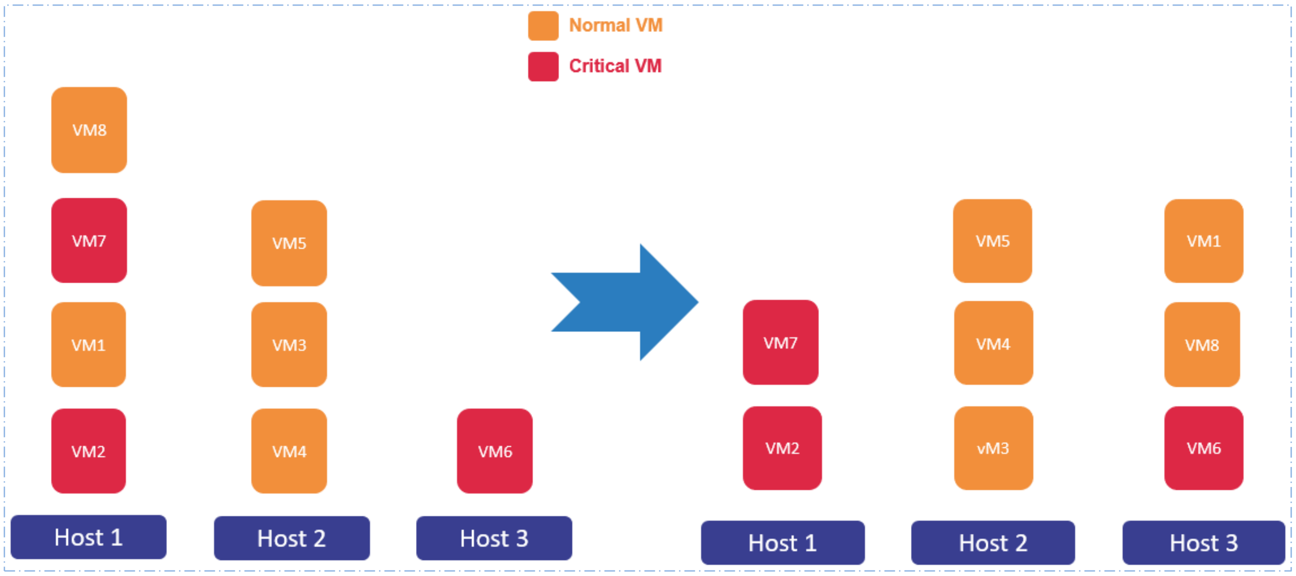


Figure 4. VM count balancing strategy

[1] Hyperconverged servers: an architecture where compute and storage workloads are combined to try to use the servers more efficiently.

c. Testing Watcher

i. Test scenario

As mentioned in 3.a., the test bed is 9 identical servers hosting 40 virtual machines. The resource utilization across is imbalanced across the servers due to the VMs distribution. In this test we will try to improve that with Watcher.

We start by launching an audit of the infrastructure with Watcher to see what actions Watcher recommends in order to rebalance the VMs distribution, then we apply those actions and see the resulting resource utilization.

ii. Initial state

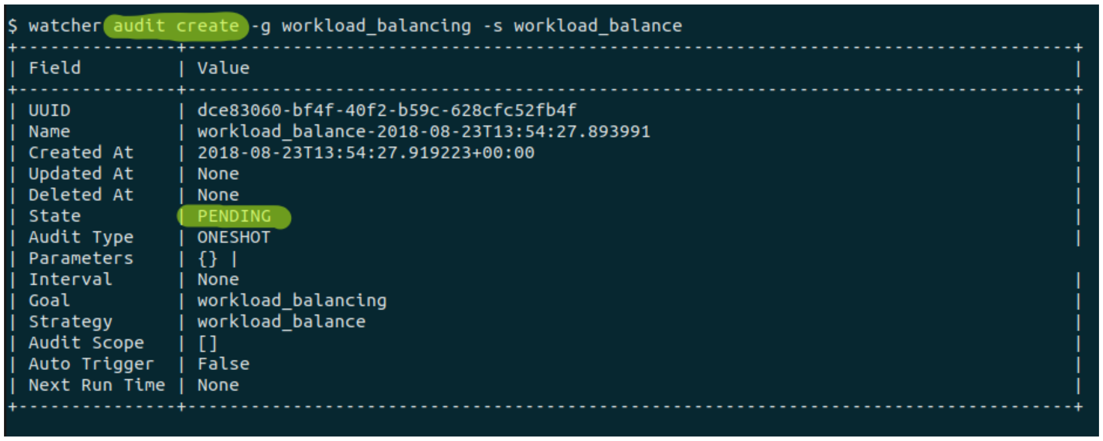
The table below summarizes the resource utilisation across the 9 servers:

hypervisor_hostname	vcpus	vcpus_used	memory_mb	memory_mb_used
h69231632006657.cern.ch	64	5	262048	23948
h69231633297344.cern.ch	64	8	262048	31384
h69231634667726.cern.ch	64	12	262048	38884
h69231630784724.cern.ch	64	20	262048	53884
h69231636936635.cern.ch	64	24	262048	61384
h69231636521310.cern.ch	64	24	262048	61384
h69231633349254.cern.ch	64	28	262048	68884
h69231639712607.cern.ch	64	32	262048	76384
h69231639979288.cern.ch	64	32	262048	76384

We notice a clear imbalance in the vcpus and memory used between the hosts. Even though the servers have the same capacity, we see that the difference in resource utilisation: 32 vcpus_used on one server (out of 64) while only 5 are used on another (out of 64).

iii. Launching an audit with Watcher

The following command creates an audit with Watcher by specifying the goal to achieve and the strategy to use. In our case, the strategy we want is workload_balance.

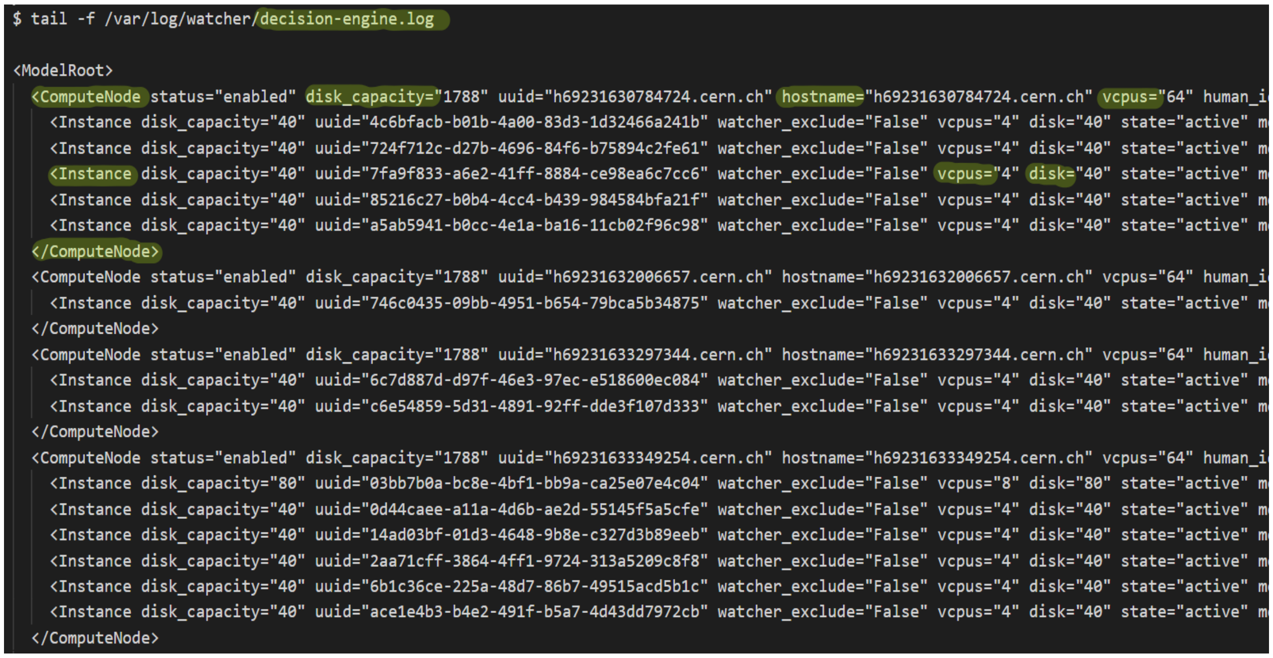


```
$ watcher audit create -g workload_balancing -s workload_balance
+---------------+--------------------------------------------------------------------------+
| Field         | Value                                                                    |
+---------------+--------------------------------------------------------------------------+
| UUID          | dce83060-bf4f-40f2-b59c-628cfc52fb4f                                     |
| Name          | workload_balance-2018-08-23T13:54:27.893991                              |
| Created At    | 2018-08-23T13:54:27.919223+00:00                                         |
| Updated At    | None                                                                     |
| Deleted At    | None                                                                     |
| State         | PENDING                                                                  |
| Audit Type    | ONESHOT                                                                  |
| Parameters    | {} |
| Interval      | None                                                                     |
| Goal          | workload_balancing                                                       |
| Strategy      | workload_balance                                                         |
| Audit Scope   | []                                                                       |
| Auto Trigger  | False                                                                    |
| Next Run Time | None                                                                     |
+---------------+--------------------------------------------------------------------------+
```

Figure 5. Creating an audit with Watcher

When we create an audit, Watcher's decision engine first builds a model of the infrastructure's current state, then based on that model, and on the given strategy, it builds an equivalent optimized modal, and computes the set of actions needed to move from the current model to the optimized one.

The figure below shows an example of the infrastructure model, taken from the decision engine's logs: every compute node (server) is listed with its characteristics, as well as all the virtual machine instances it is hosting, with their respective characteristics.

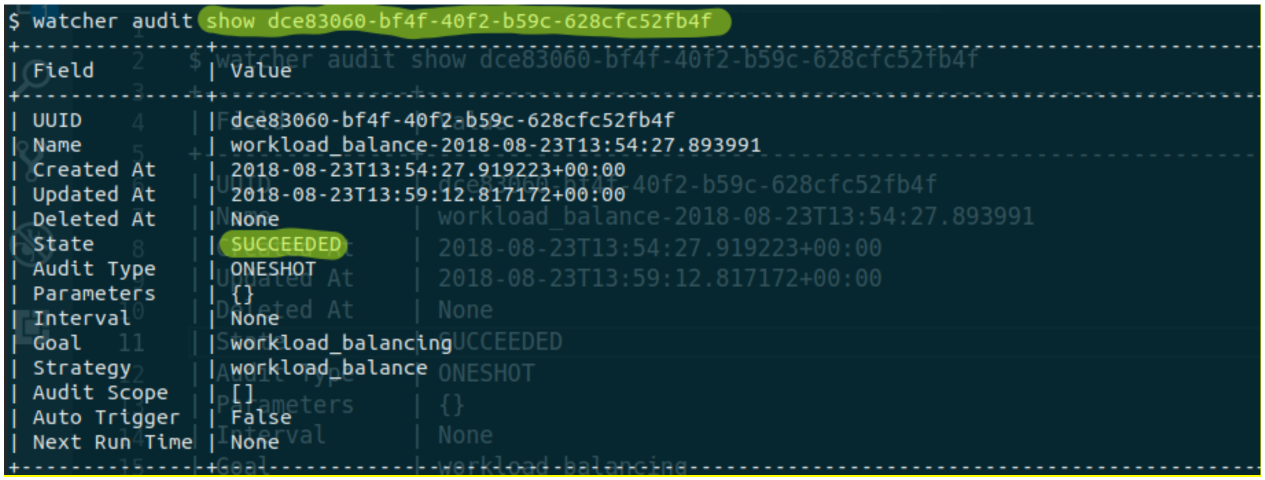


```
$ tail -f /var/log/watcher/decision-engine.log

<ModelRoot>
  <ComputeNode status="enabled" disk_capacity="1788" uuid="h69231630784724.cern.ch" hostname="h69231630784724.cern.ch" vcpus="64" human_i
    <Instance disk_capacity="40" uuid="4c6bfacb-b01b-4a00-83d3-1d32466a241b" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="724f712c-d27b-4696-84f6-b75894c2fe61" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="7fa9f833-a6e2-41ff-8884-ce98ea6c7cc6" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="85216c27-b0b4-4cc4-b439-984584bfa21f" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="a5ab5941-b0cc-4e1a-ba16-11cb02f96c98" watcher_exclude="False" vcpus="4" disk="40" state="active" m
  </ComputeNode>
  <ComputeNode status="enabled" disk_capacity="1788" uuid="h69231632006657.cern.ch" hostname="h69231632006657.cern.ch" vcpus="64" human_i
    <Instance disk_capacity="40" uuid="746c0435-09bb-4951-b654-79bca5b34875" watcher_exclude="False" vcpus="4" disk="40" state="active" m
  </ComputeNode>
  <ComputeNode status="enabled" disk_capacity="1788" uuid="h69231633297344.cern.ch" hostname="h69231633297344.cern.ch" vcpus="64" human_i
    <Instance disk_capacity="40" uuid="6c7d887d-d97f-46e3-97ec-e518600ec084" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="c6e54859-5d31-4891-92ff-dde3f107d333" watcher_exclude="False" vcpus="4" disk="40" state="active" m
  </ComputeNode>
  <ComputeNode status="enabled" disk_capacity="1788" uuid="h69231633349254.cern.ch" hostname="h69231633349254.cern.ch" vcpus="64" human_i
    <Instance disk_capacity="80" uuid="03bb7b0a-bc8e-4bf1-bb9a-ca25e07e4c04" watcher_exclude="False" vcpus="8" disk="80" state="active" m
    <Instance disk_capacity="40" uuid="0d44caee-a11a-4d6b-ae2d-55145f5a5cfe" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="14ad03bf-01d3-4648-9b8e-c327d3b89eeb" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="2aa71cff-3864-4ff1-9724-313a5209c8f8" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="6b1c36ce-225a-48d7-86b7-49515acd5b1c" watcher_exclude="False" vcpus="4" disk="40" state="active" m
    <Instance disk_capacity="40" uuid="ace1e4b3-b4e2-491f-b5a7-4d43dd7972cb" watcher_exclude="False" vcpus="4" disk="40" state="active" m
  </ComputeNode>
```

Figure 6. Infrastructure model built by the Watcher's decision engine

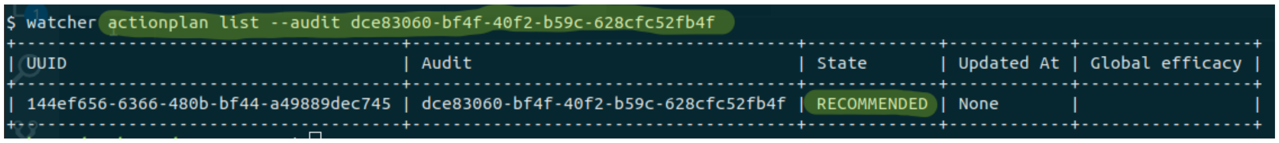
After the decision engine successfully computes the action plan for the given strategy, the audit's state changes to "succeeded", which can be viewed with the following command:



```
$ watcher audit show dce83060-bf4f-40f2-b59c-628cfc52fb4f
+---------------+--------------------------------------------------------------
| Field         | Value
+---------------+--------------------------------------------------------------
| UUID          | dce83060-bf4f-40f2-b59c-628cfc52fb4f
| Name          | workload_balance-2018-08-23T13:54:27.893991
| Created At    | 2018-08-23T13:54:27.919223+00:00
| Updated At    | 2018-08-23T13:59:12.817172+00:00
| Deleted At    | None
| State         | SUCCEEDED
| Audit Type    | ONESHOT
| Parameters    | {}
| Interval      | None
| Goal          | workload_balancing
| Strategy      | workload_balance
| Audit Scope   | []
| Auto Trigger  | False
| Next Run Time | None
+---------------+--------------------------------------------------------------
```

Figure 7. Showing an audit with Watcher

Now that the audit succeeded, we can check that the decision engine created an action plan for our audit:

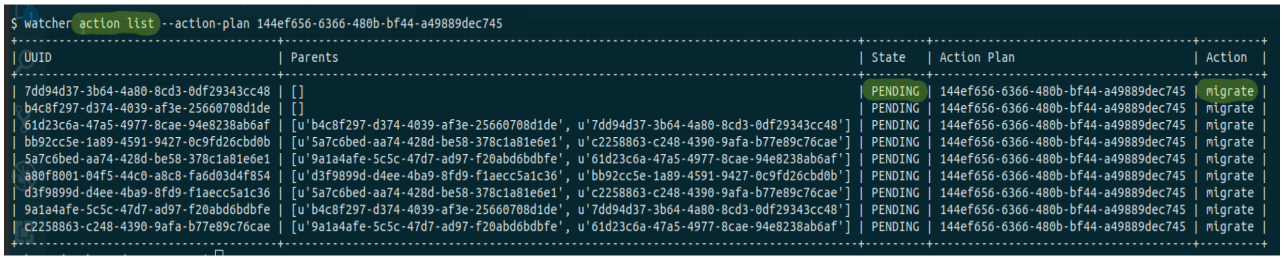


```
$ watcher actionplan list --audit dce83060-bf4f-40f2-b59c-628cfc52fb4f
+--------------------------------------+--------------------------------------+-------------+------------+-----------------+
| UUID                                 | Audit                                | State       | Updated At | Global efficacy |
+--------------------------------------+--------------------------------------+-------------+------------+-----------------+
| 144ef656-6366-480b-bf44-a49889dec745 | dce83060-bf4f-40f2-b59c-628cfc52fb4f | RECOMMENDED | None       |                 |
+--------------------------------------+--------------------------------------+-------------+------------+-----------------+
```

Figure 8. Showing an audit's action plan

iv. Executing Watcher's optimization plan

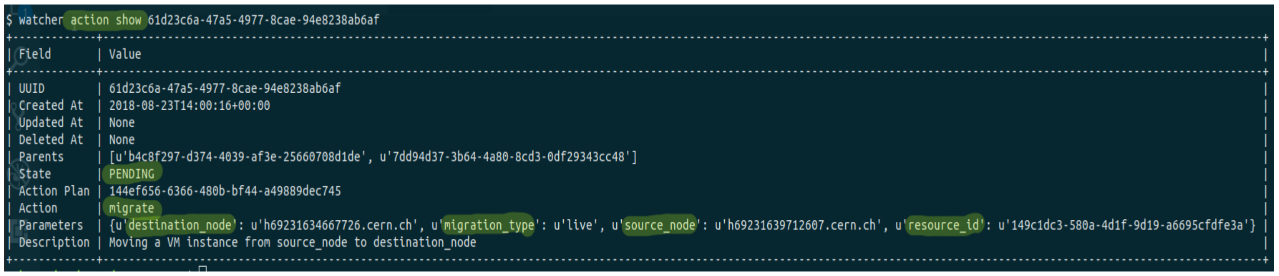
Before executing the action plan, let us see what actions will be executed. The following command does that:



```
$ watcher action list --action-plan 144ef656-6366-480b-bf44-a49889dec745
+--------------------------------------+----------------------------------------------------------------------------------------------+---------+--------------------------------------+---------+
| UUID                                 | Parents                                                                                      | State   | Action Plan                          | Action  |
+--------------------------------------+----------------------------------------------------------------------------------------------+---------+--------------------------------------+---------+
| 7dd94d37-3b64-4a80-8cd3-0df29343cc48 | []                                                                                           | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| b4c8f297-d374-4039-af3e-25660708d1de | []                                                                                           | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| 61d23c6a-47a5-4977-8cae-94e8238ab6af | [u'b4c8f297-d374-4039-af3e-25660708d1de', u'7dd94d37-3b64-4a80-8cd3-0df29343cc48']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| bb92cc5e-1a89-4591-9427-0c9fd26cbd0b | [u'5a7c6bed-aa74-428d-be58-378c1a81e6e1', u'c2258863-c248-4390-9afa-b77e89c76cae']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| 5a7c6bed-aa74-428d-be58-378c1a81e6e1 | [u'9a1a4afe-5c5c-47d7-ad97-f20abd6bdbfe', u'61d23c6a-47a5-4977-8cae-94e8238ab6af']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| a80f8001-04f5-44c0-a8c8-fa6d03d4f854 | [u'd3f9899d-d4ee-4ba9-8fd9-f1aecc5a1c36', u'bb92cc5e-1a89-4591-9427-0c9fd26cbd0b']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| d3f9899d-d4ee-4ba9-8fd9-f1aecc5a1c36 | [u'5a7c6bed-aa74-428d-be58-378c1a81e6e1', u'c2258863-c248-4390-9afa-b77e89c76cae']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| 9a1a4afe-5c5c-47d7-ad97-f20abd6bdbfe | [u'b4c8f297-d374-4039-af3e-25660708d1de', u'7dd94d37-3b64-4a80-8cd3-0df29343cc48']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
| c2258863-c248-4390-9afa-b77e89c76cae | [u'9a1a4afe-5c5c-47d7-ad97-f20abd6bdbfe', u'61d23c6a-47a5-4977-8cae-94e8238ab6af']          | PENDING | 144ef656-6366-480b-bf44-a49889dec745 | migrate |
+--------------------------------------+----------------------------------------------------------------------------------------------+---------+--------------------------------------+---------+
```

Figure 9. Showing an action-plan's list of actions

We see that all the actions are pending, and each action is a migration of a VM. We can zoom into one of the actions to see it in detail:

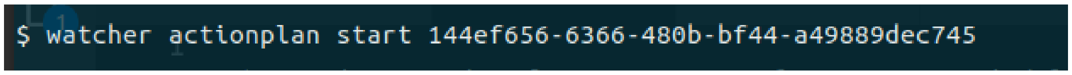


```
$ watcher action show 61d23c6a-47a5-4977-8cae-94e8238ab6af
+-------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Field       | Value                                                                                                                                                                        |
+-------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| UUID        | 61d23c6a-47a5-4977-8cae-94e8238ab6af                                                                                                                                         |
| Created At  | 2018-08-23T14:00:16+00:00                                                                                                                                                    |
| Updated At  | None                                                                                                                                                                         |
| Deleted At  | None                                                                                                                                                                         |
| Parents     | [u'b4c8f297-d374-4039-af3e-25660708d1de', u'7dd94d37-3b64-4a80-8cd3-0df29343cc48']                                                                                          |
| State       | PENDING                                                                                                                                                                      |
| Action Plan | 144ef656-6366-480b-bf44-a49889dec745                                                                                                                                         |
| Action      | migrate                                                                                                                                                                      |
| Parameters  | {u'destination_node': u'h69231634667726.cern.ch', u'migration_type': u'live', u'source_node': u'h69231639712607.cern.ch', u'resource_id': u'149c1dc3-580a-4d1f-9d19-a6695cfdfe3a'} |
| Description | Moving a VM instance from source_node to destination_node                                                                                                                    |
+-------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

Figure 10. Showing an action with Watcher

In the parameters section, we can find information about the migration including the id of the VM instance to be moved, the source node (host in which the instance is), and the destination node (where it will be migrated).

Now, we execute the action plan with the next command:

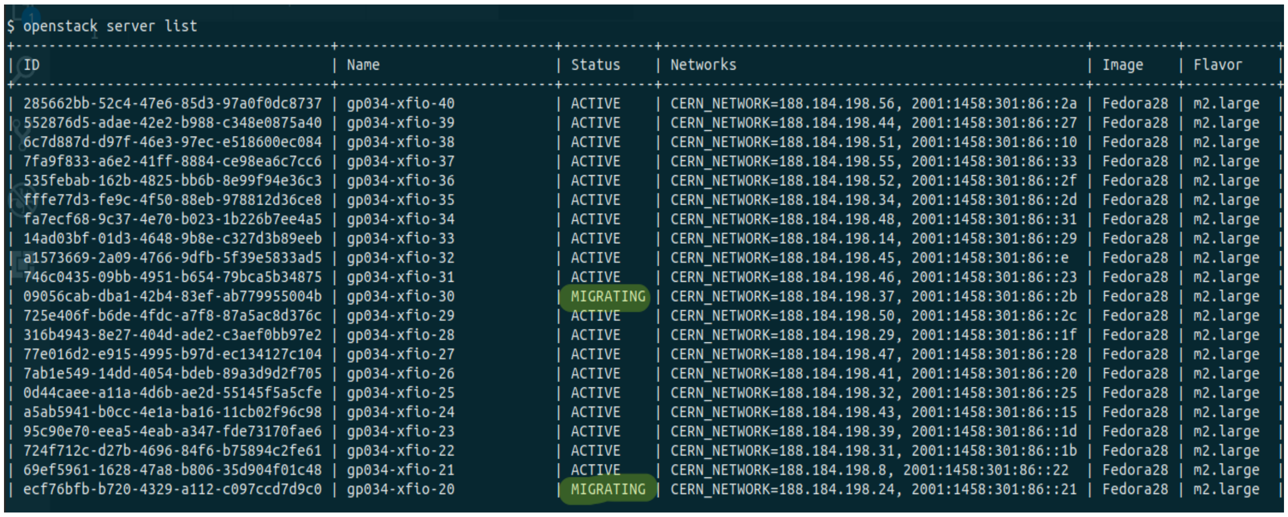


```
$ watcher actionplan start 144ef656-6366-480b-bf44-a49889dec745
```

Figure 11. Starting an action plan with Watcher

When we start an action plan, Watcher's applier engine executes the actions in the predefined order, making changes on the infrastructure.

By running the command below after starting the action plan, we can see that some of the VM instances are migrating:

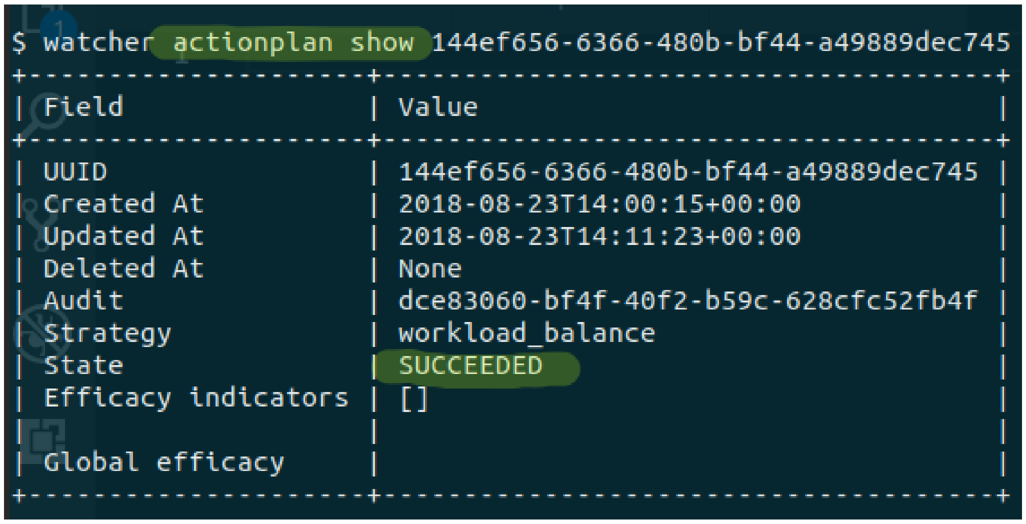


```
$ openstack server list
+--------------------------------------+------------------------+-----------+----------------------------------------------------+----------+------------+
| ID                                   | Name                   | Status    | Networks                                           | Image    | Flavor     |
+--------------------------------------+------------------------+-----------+----------------------------------------------------+----------+------------+
| 285662bb-52c4-47e6-85d3-97a0f0dc8737 | gp034-xfio-40          | ACTIVE    | CERN_NETWORK=188.184.198.56, 2001:1458:301:86::2a  | Fedora28 | m2.large   |
| 552876d5-adae-42e2-b988-c348e0875a40 | gp034-xfio-39          | ACTIVE    | CERN_NETWORK=188.184.198.44, 2001:1458:301:86::27  | Fedora28 | m2.large   |
| 6c7d887d-d97f-46e3-97ec-e518600ec084 | gp034-xfio-38          | ACTIVE    | CERN_NETWORK=188.184.198.51, 2001:1458:301:86::10  | Fedora28 | m2.large   |
| 7fa9f833-a6e2-41ff-8884-ce98ea6c7cc6 | gp034-xfio-37          | ACTIVE    | CERN_NETWORK=188.184.198.55, 2001:1458:301:86::33  | Fedora28 | m2.large   |
| 535febab-162b-4825-bb6b-8e99f94e36c3 | gp034-xfio-36          | ACTIVE    | CERN_NETWORK=188.184.198.52, 2001:1458:301:86::2f  | Fedora28 | m2.large   |
| fffe77d3-fe9c-4f50-88eb-978812d36ce8 | gp034-xfio-35          | ACTIVE    | CERN_NETWORK=188.184.198.34, 2001:1458:301:86::2d  | Fedora28 | m2.large   |
| fa7ecf68-9c37-4e70-b023-1b226b7ee4a5 | gp034-xfio-34          | ACTIVE    | CERN_NETWORK=188.184.198.48, 2001:1458:301:86::31  | Fedora28 | m2.large   |
| 14ad03bf-01d3-4648-9b8e-c327d3b89eeb | gp034-xfio-33          | ACTIVE    | CERN_NETWORK=188.184.198.14, 2001:1458:301:86::29  | Fedora28 | m2.large   |
| a1573669-2a09-4766-9dfb-5f39e5833ad5 | gp034-xfio-32          | ACTIVE    | CERN_NETWORK=188.184.198.45, 2001:1458:301:86::e   | Fedora28 | m2.large   |
| 746c0435-09bb-4951-b654-79bca5b34875 | gp034-xfio-31          | ACTIVE    | CERN_NETWORK=188.184.198.46, 2001:1458:301:86::23  | Fedora28 | m2.large   |
| 09056cab-dba1-42b4-83ef-ab779955004b | gp034-xfio-30          | MIGRATING | CERN_NETWORK=188.184.198.37, 2001:1458:301:86::2b  | Fedora28 | m2.large   |
| 725e406f-b6de-4fdc-a7f8-87a5ac8d376c | gp034-xfio-29          | ACTIVE    | CERN_NETWORK=188.184.198.50, 2001:1458:301:86::2c  | Fedora28 | m2.large   |
| 316b4943-8e27-404d-ade2-c3aef0bb97e2 | gp034-xfio-28          | ACTIVE    | CERN_NETWORK=188.184.198.29, 2001:1458:301:86::1f  | Fedora28 | m2.large   |
| 77e016d2-e915-4995-b97d-ec134127c104 | gp034-xfio-27          | ACTIVE    | CERN_NETWORK=188.184.198.47, 2001:1458:301:86::28  | Fedora28 | m2.large   |
| 7ab1e549-14dd-4054-bdeb-89a3d9d2f705 | gp034-xfio-26          | ACTIVE    | CERN_NETWORK=188.184.198.41, 2001:1458:301:86::20  | Fedora28 | m2.large   |
| 0d44caee-a11a-4d6b-ae2d-55145f5a5cfe | gp034-xfio-25          | ACTIVE    | CERN_NETWORK=188.184.198.32, 2001:1458:301:86::25  | Fedora28 | m2.large   |
| a5ab5941-b0cc-4e1a-ba16-11cb02f96c98 | gp034-xfio-24          | ACTIVE    | CERN_NETWORK=188.184.198.43, 2001:1458:301:86::15  | Fedora28 | m2.large   |
| 95c90e70-eea5-4eab-a347-fde73170fae6 | gp034-xfio-23          | ACTIVE    | CERN_NETWORK=188.184.198.39, 2001:1458:301:86::1d  | Fedora28 | m2.large   |
| 724f712c-d27b-4696-84f6-b75894c2fe61 | gp034-xfio-22          | ACTIVE    | CERN_NETWORK=188.184.198.31, 2001:1458:301:86::1b  | Fedora28 | m2.large   |
| 69ef5961-1628-47a8-b806-35d904f01c48 | gp034-xfio-21          | ACTIVE    | CERN_NETWORK=188.184.198.8, 2001:1458:301:86::22   | Fedora28 | m2.large   |
| ecf76bfb-b720-4329-a112-c097ccd7d9c0 | gp034-xfio-20          | MIGRATING | CERN_NETWORK=188.184.198.24, 2001:1458:301:86::21  | Fedora28 | m2.large   |
```

Figure 12. Listing the list of VM instances with openstack

v. Result

Once the action plan finishes and no error occurs, its state changes from "ongoing" to "succeeded":



```
$ watcher actionplan show 144ef656-6366-480b-bf44-a49889dec745
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| UUID                | 144ef656-6366-480b-bf44-a49889dec745 |
| Created At          | 2018-08-23T14:00:15+00:00            |
| Updated At          | 2018-08-23T14:11:23+00:00            |
| Deleted At          | None                                 |
| Audit               | dce83060-bf4f-40f2-b59c-628cfc52fb4f |
| Strategy            | workload_balance                     |
| State               | SUCCEEDED                            |
| Efficacy indicators | []                                   |
|                     |                                      |
| Global efficacy     |                                      |
+---------------------+--------------------------------------+
```

Figure 13. Showing an action plan with Watcher

This means that all the actions included in this action plan were successfully executed on our infrastructure. Now to conclude this test, we check again the resource utilization to see how well our strategy performed:

hypervisor_hostname	vcpus	vcpus_used	memory_mb	memory_mb_used
h69231632006657.cern.ch	64	20	262048	53884
h69231633297344.cern.ch	64	20	262048	53884
h69231634667726.cern.ch	64	20	262048	53884
h69231630784724.cern.ch	64	20	262048	53884
h69231636936635.cern.ch	64	20	262048	53884
h69231636521310.cern.ch	64	20	262048	53884
h69231633349254.cern.ch	64	20	262048	53884
h69231639712607.cern.ch	64	20	262048	53884
h69231639979288.cern.ch	64	24	262048	61384

The resource utilization is now balanced across the servers, our strategy worked as expected on this set up.

4. Conclusion

Resource optimization in cloud infrastructures reduces operations cost and leads to more efficient usage of the available computing power and storage. A lot of effort is being done at CERN to optimize resource utilization in its private OpenStack cloud deployment.

In this work, we investigated Watcher, the resource optimization service for OpenStack. We deployed Watcher in pre-production, and showed it is easily extensible by adding our own custom optimization strategy to it to fit a specific use case at CERN.

A good continuation of this work would be to further integrate Watcher with CERN monitoring tools in order to get more metrics, then find more use cases for Watcher at CERN, and develop optimization strategies for those use cases. And finally, combine Watcher with other works that are being conducted at CERN to optimize utilization of its cloud resources.

References

[1] http://superuser.openstack.org/articles/openstack-production-cern-lightning-talk/

[2] http://openstack-in-production.blogspot.com/2018/02/maximizing-resource-utilization-with.html

[3] https://governance.openstack.org/tc/reference/projects/

[4] https://wiki.openstack.org/wiki/Watcher

[5] https://docs.openstack.org/watcher/latest/architecture.html

[6] https://www.openstack.org/project-mascots/