Rally Documentation

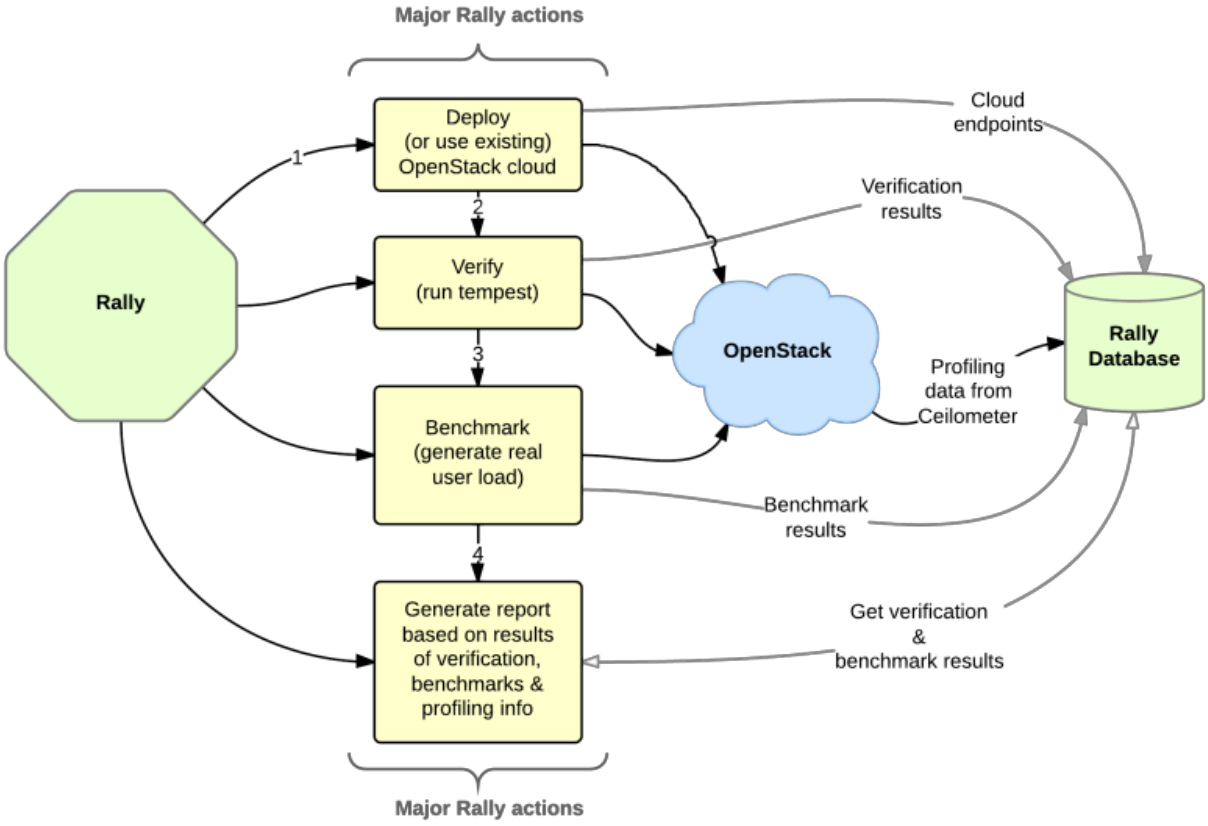
Release 0.9.0

OpenStack Foundation

Mar 21, 2017

Contents

OpenStack is, undoubtedly, a really *huge* ecosystem of cooperative services. **Rally** is a **benchmarking tool** that answers the question: **"How does OpenStack work at scale?"**. To make this possible, Rally **automates** and **unifies** multi-node OpenStack deployment, cloud verification, benchmarking & profiling. Rally does it in a **generic** way, making it possible to check whether OpenStack is going to work well on, say, a 1k-servers installation under high load. Thus it can be used as a basic tool for an *OpenStack CI/CD system* that would continuously improve its SLA, performance and stability.



CHAPTER 1

Contents

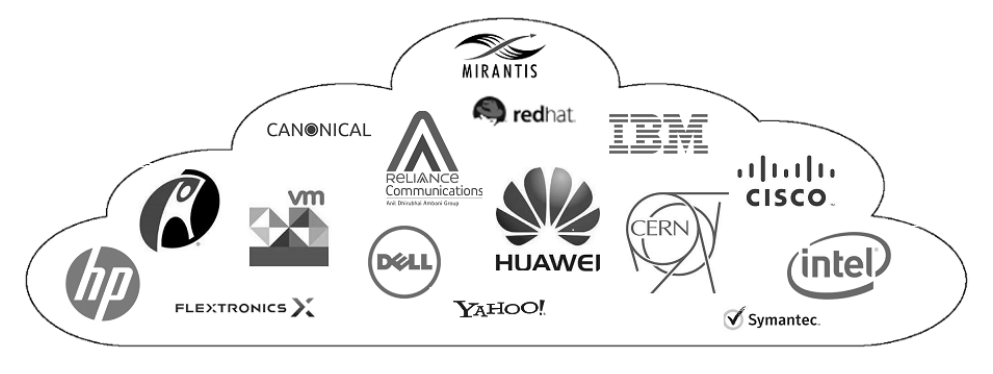
Rally project overview

Overview

Rally is a **benchmarking tool** that **automates** and **unifies** multi-node OpenStack deployment, cloud verification, benchmarking & profiling. It can be used as a basic tool for an *OpenStack CI/CD system* that would continuously improve its SLA, performance and stability.

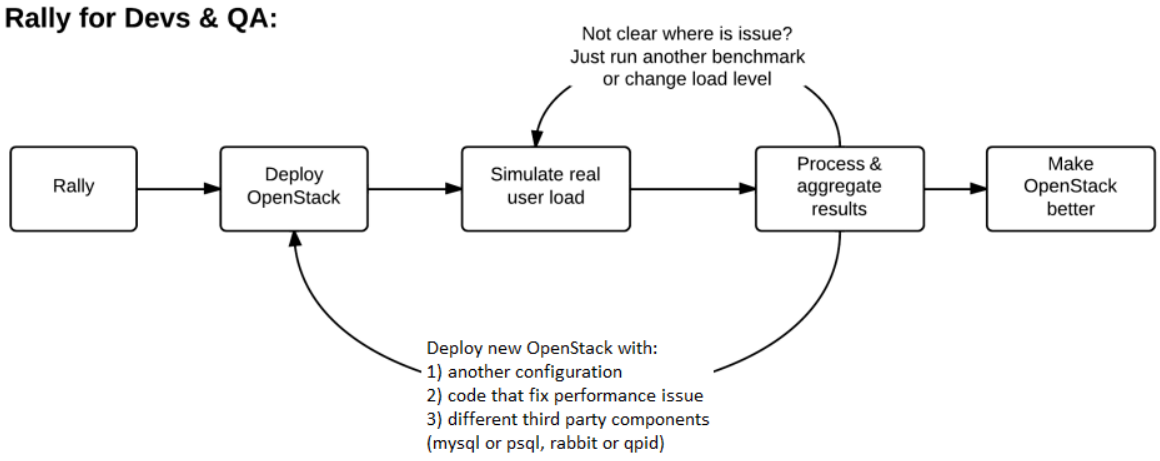
Who Is Using Rally

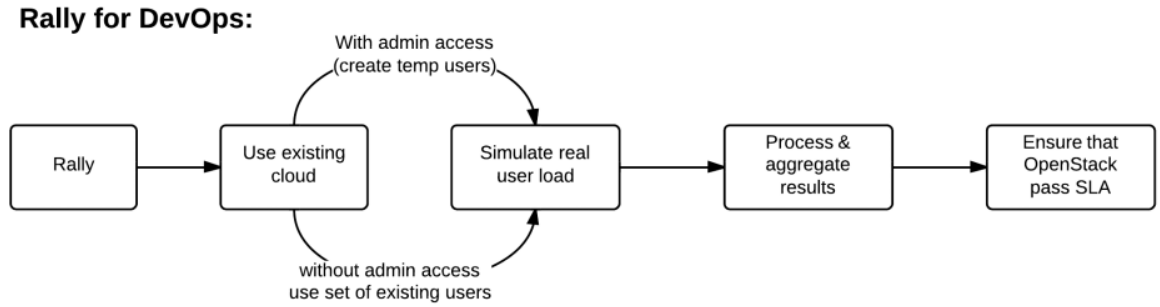
Here's a small selection of some of the many companies using Rally:

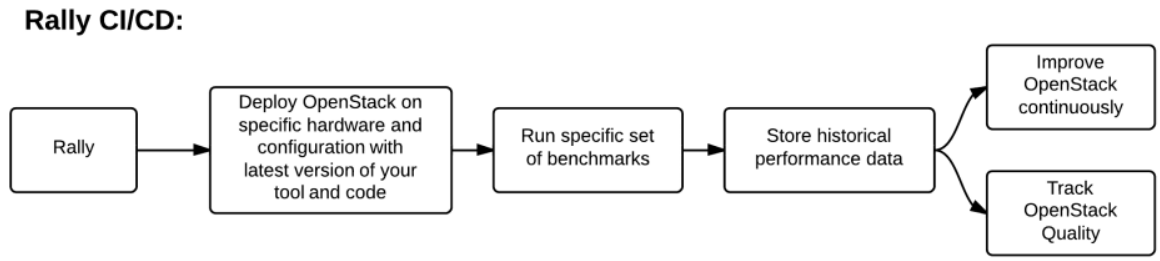


Use Cases

Let's take a look at 3 major high level Use Cases of Rally:







Generally, there are a few typical cases where Rally proves to be of great use:

1. Automate measuring & profiling focused on how new code changes affect the OS performance;
2. Using Rally profiler to detect scaling & performance issues;
3. Investigate how different deployments affect the OS performance:
   - Find the set of suitable OpenStack deployment architectures;
   - Create deployment specifications for different loads (amount of controllers, swift nodes, etc.);
4. Automate the search for hardware best suited for particular OpenStack cloud;
5. Automate the production cloud specification generation:
   - Determine terminal loads for basic cloud operations: VM start & stop, Block Device create/destroy & various OpenStack API methods;

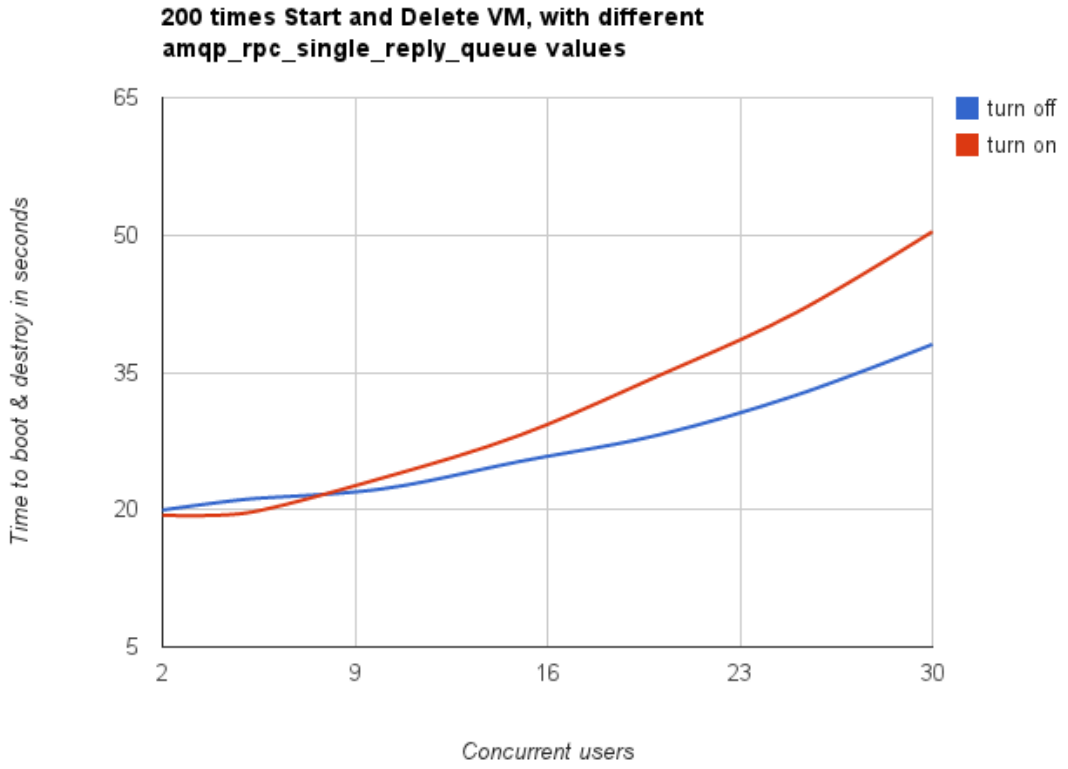
- Check performance of basic cloud operations in case of different loads.

Real-life examples

To be substantive, let's investigate a couple of real-life examples of Rally in action.

How does amqp_rpc_single_reply_queue affect performance?

Rally allowed us to reveal a quite an interesting fact about **Nova**. We used *NovaServers.boot_and_delete* benchmark scenario to see how the *amqp_rpc_single_reply_queue* option affects VM bootup time (it turns on a kind of fast RPC). Some time ago it was shown that cloud performance can be boosted by setting it on, so we naturally decided to check this result with Rally. To make this test, we issued requests for booting and deleting VMs for a number of concurrent users ranging from 1 to 30 with and without the investigated option. For each group of users, a total number of 200 requests was issued. Averaged time per request is shown below:



So Rally has unexpectedly indicated that setting the *amqp_rpc_single_reply_queue* option apparently affects the cloud performance, but in quite an opposite way rather than it was thought before.

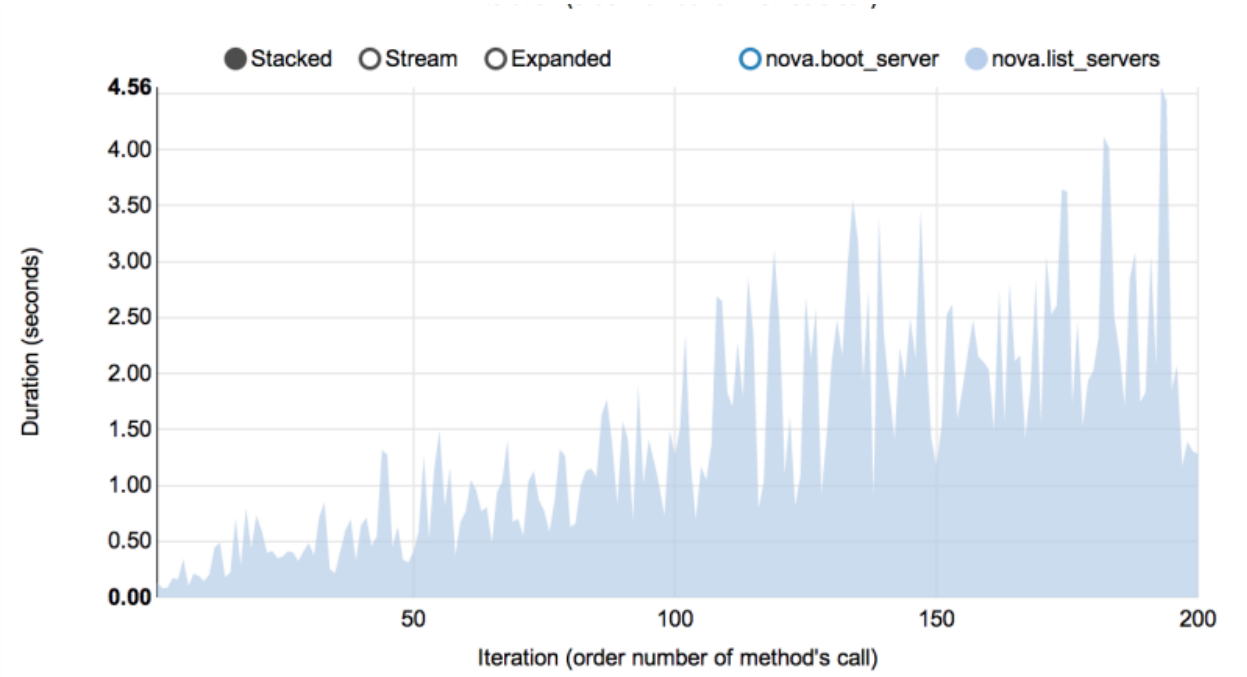
Performance of Nova list command

Another interesting result comes from the *NovaServers.boot_and_list_server* scenario, which enabled us to we launched the following benchmark with Rally:

- **Benchmark environment** (which we also call **"Context"**): 1 temporary OpenStack user.

- **Benchmark scenario**: boot a single VM from this user & list all VMs.
- **Benchmark runner** setting: repeat this procedure 200 times in a continuous way.

During the execution of this benchmark scenario, the user has more and more VMs on each iteration. Rally has shown that in this case, the performance of the **VM list** command in Nova is degrading much faster than one might expect:

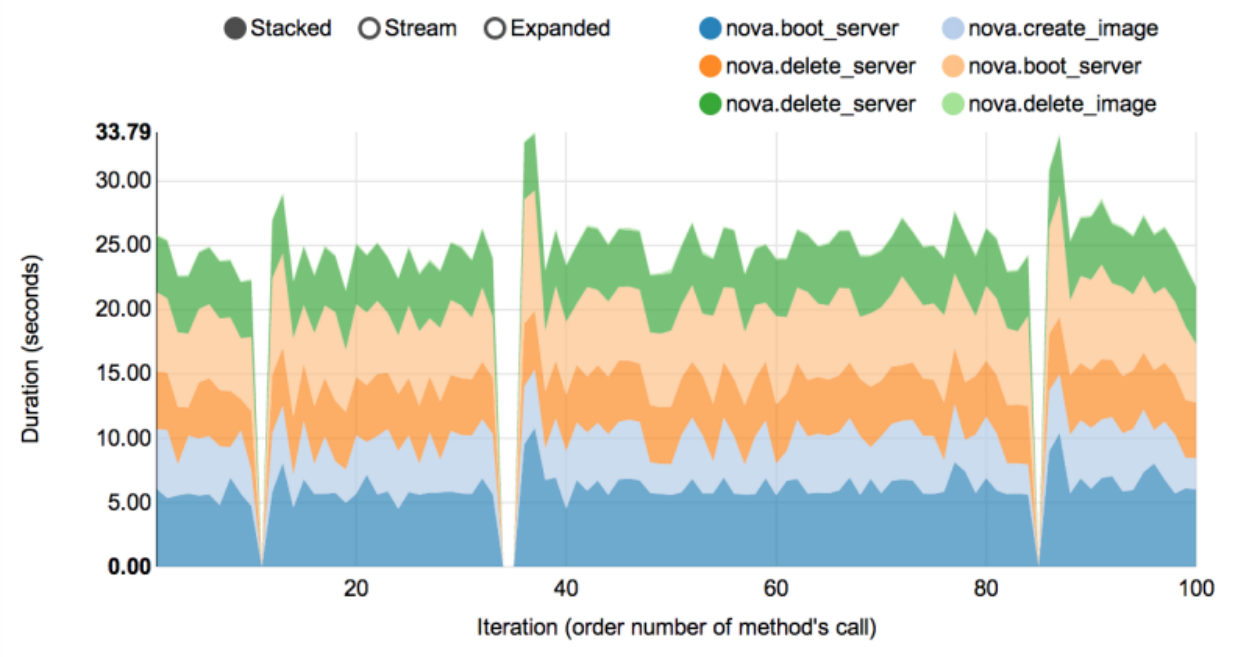


Complex scenarios

In fact, the vast majority of Rally scenarios is expressed as a sequence of **"atomic" actions**. For example, *NovaServers.snapshot* is composed of 6 atomic actions:

1. boot VM
2. snapshot VM
3. delete VM
4. boot VM from snapshot
5. delete VM
6. delete snapshot

Rally measures not only the performance of the benchmark scenario as a whole, but also that of single atomic actions. As a result, Rally also plots the atomic actions performance data for each benchmark iteration in a quite detailed way:

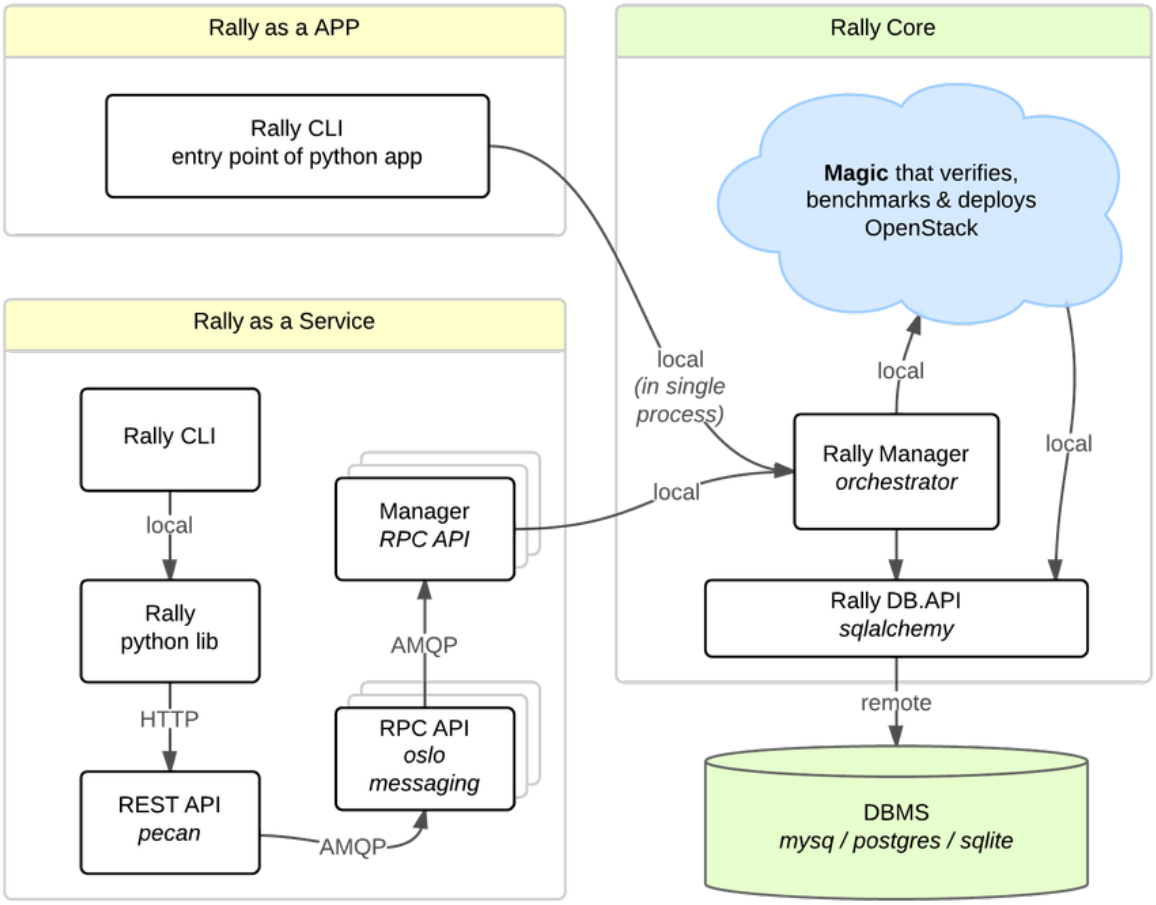


Architecture

Usually OpenStack projects are implemented *"as-a-Service"*, so Rally provides this approach. In addition, it implements a *CLI-driven* approach that does not require a daemon:

1. **Rally as-a-Service**: Run rally as a set of daemons that present Web UI *(work in progress)* so 1 RaaS could be used by a whole team.
2. **Rally as-an-App**: Rally as a just lightweight and portable CLI app (without any daemons) that makes it simple to use & develop.

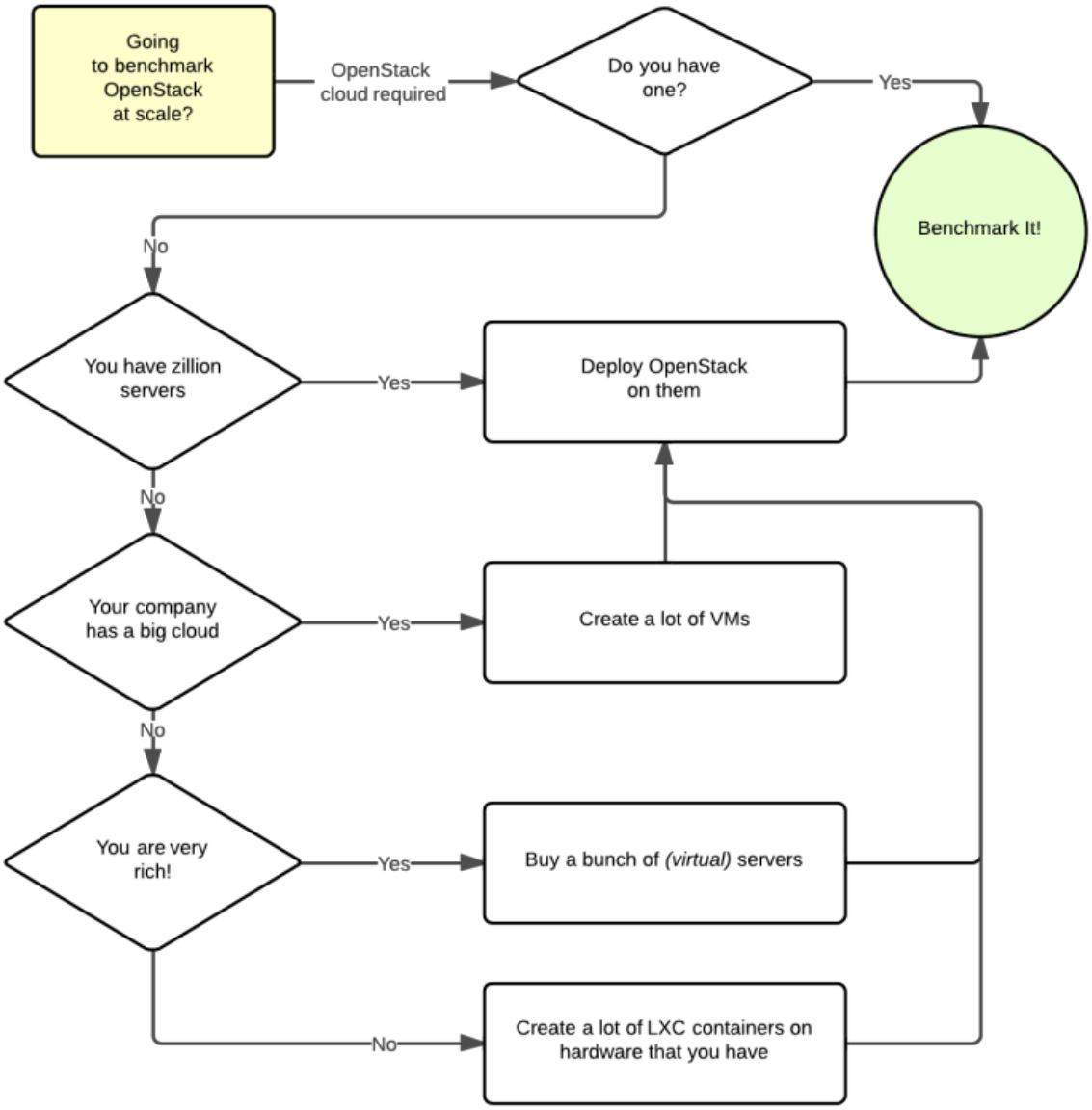
The diagram below shows how this is possible:



The actual **Rally core** consists of 4 main components, listed below in the order they go into action:

1. **Server Providers** - provide a **unified interface** for interaction with different **virtualization technologies** (*LXS*, *Virsh* etc.) and **cloud suppliers** (like *Amazon*): it does so via *ssh* access and in one *L3 network*;
2. **Deploy Engines** - deploy some OpenStack distribution (like *DevStack* or *FUEL*) before any benchmarking procedures take place, using servers retrieved from Server Providers;
3. **Verification** - runs *Tempest* (or another specific set of tests) against the deployed cloud to check that it works correctly, collects results & presents them in human readable form;
4. **Benchmark Engine** - allows to write parameterized benchmark scenarios & run them against the cloud.

It should become fairly obvious why Rally core needs to be split to these parts if you take a look at the following diagram that visualizes a rough **algorithm for starting benchmarking OpenStack at scale**. Keep in mind that there might be lots of different ways to set up virtual servers, as well as to deploy OpenStack to them.



Glossary

Warning: Unfortunately, our glossary is not full, but the Rally team is working on improving it. If you cannot find a definition in which you are interested, feel free to ping us via IRC (#openstack-rally channel at Freenode) or via E-Mail (openstack-dev@lists.openstack.org with tag [Rally]).

- *Task*
- *Verify*

Common

Alembic

A lightweight database migration tool which powers Rally migrations. Read more at Official Alembic documentation

DB Migrations

Rally supports database schema and data transformations, which are also known as migrations. This allows you to get your data up-to-date with latest Rally version.

Rally

A testing tool that automates and unifies multi-node OpenStack deployment and cloud verification. It can be used as a basic tool for an OpenStack CI/CD system that would continuously improve its SLA, performance and stability.

Rally Config

Rally behavior can be customized by editing its configuration file, *rally.conf*, in configparser format. While being installed, Rally generates a config with default values from its sample. When started, Rally searches for its config in "<sys.prefix>/etc/rally/rally.conf", "~/.rally/rally.conf", "/etc/rally/rally.conf"

Rally DB

Rally uses a relational database as data storage. Several database backends are supported: SQLite (default), PostgreSQL, and MySQL. The database connection can be set via the configuration file option *[database]/connection*.

Rally Plugin

Most parts of Rally are pluggable. Scenarios, runners, contexts and even charts for HTML report are plugins. It is easy to create your own plugin and use it. Read more at plugin reference.

Deployment

Deployment

A set of information about target environment (for example: URI and authentication credentials) which is saved in the database. It is used to define the target system for testing each time a task is started. It has a "type" value which changes task behavior for the selected target system; for example type "openstack" will enable OpenStack authentication and services.

Task

Cleanup

This is a specific context which removes all resources on target system that were created by the current task. If some Rally-related resources remain, please file a bug and attach the task file and a list of remaining resources.

Context

A type of plugin that can run some actions on the target environment before the workloads start and after the last workload finishes. This allows, for example, preparing the environment for workloads (e.g., create resources and change parameters) and restoring the environment later. Each Context must implement `setup()` and `cleanup()` methods.

Input task

A file that describes how to run a Rally Task. It can be in JSON or YAML format. The *rally task start* command needs this file to run the task. The input task is pre-processed by the Jinja2 templating engine so it is very easy to create repeated parts or calculate specific values at runtime. It is also possible to pass values via CLI arguments, using the *–task-args* or *–task-args-file* options.

Runner

This is a Rally plugin which decides how to run Workloads. For example, they can be run serially in a single process, or using concurrency.

Scenario

Synonym for *Workload*

Service

Abstraction layer that represents target environment API. For example, this can be some OpenStack service. A Service provides API versioning and action timings, simplifies API calls, and reduces code duplication. It can be used in any Rally plugin.

SLA

Service-Level Agreement (Success Criteria). Allows you to determine whether a subtask or workload is successful by setting success criteria rules.

Subtask

A part of a Task. There can be many subtasks in a single Task.

Task

An entity which includes all the necessary data for a test run, and results of this run.

Workload

An important part of Task: a plugin which is run by the runner. It is usually run in separate thread. Workloads are grouped into Subtasks.

Verify

Rally can run different subunit-based testing tools against a target environment, for example tempest for OpenStack.

Verification

A result of running some third-party subunit-based testing tool.

User stories

Many users of Rally were able to make interesting discoveries concerning their OpenStack clouds using our benchmarking tool. Numerous user stories presented below show how Rally has made it possible to find performance bugs and validate improvements for different OpenStack installations.

4x performance increase in Keystone inside Apache using the token creation benchmark

(Contributed by Neependra Khare, Red Hat)

Below we describe how we were able to get and verify a 4x better performance of Keystone inside Apache. To do that, we ran a Keystone token creation benchmark with Rally under different load (this benchmark scenario essentially just authenticate users with keystone to get tokens).

Goal

- Get the data about performance of token creation under different load.
- Ensure that keystone with increased public_workers/admin_workers values and under Apache works better than the default setup.

Summary

- As the concurrency increases, time to authenticate the user gets up.
- Keystone is CPU bound process and by default only one thread of *keystone-all* process get started. We can increase the parallelism by:
  1. increasing *public_workers/admin_workers* values in *keystone.conf* file
  2. running Keystone inside Apache
- We configured Keystone with 4 *public_workers* and ran Keystone inside Apache. In both cases we got up to 4x better performance as compared to default Keystone configuration.

Setup

Server : Dell PowerEdge R610

CPU make and model : Intel(R) Xeon(R) CPU X5650 @ 2.67GHz

CPU count: 24

RAM : 48 GB

Devstack - Commit#d65f7a2858fb047b20470e8fa62ddaede2787a85

Keystone - Commit#455d50e8ae360c2a7598a61d87d9d341e5d9d3ed

Keystone API - 2

To increase public_workers - Uncomment line with *public_workers* and set *public_workers* to 4. Then restart Keystone service.

To run Keystone inside Apache - Added *APACHE_ENABLED_SERVICES=key* in *localrc* file while setting up OpenStack environment with Devstack.

Results

1. Concurrency = 4

```
{'context': {'users': {'concurrent': 30,
                       'tenants': 12,
                       'users_per_tenant': 512}},
                       'runner': {'concurrency': 4, 'times': 10000, 'type': 'constant
↪'}}
```

action	min (sec)	avg (sec)	max (sec)	90 percentile	95 percentile	success	count	apache enabled keystone	public_workers
total	0.537	0.998	4.553	1.233	1.391	100.0%	10000	N	1
total	0.189	0.296	5.099	0.417	0.474	100.0%	10000	N	4
total	0.208	0.299	3.228	0.437	0.485	100.0%	10000	Y	NA

2. Concurrency = 16

```
{'context': {'users': {'concurrent': 30,
                       'tenants': 12,
                       'users_per_tenant': 512}},
                       'runner': {'concurrency': 16, 'times': 10000, 'type': 'constant
↪'}}
```

action	min (sec)	avg (sec)	max (sec)	90 percentile	95 percentile	success	count	apache enabled keystone	public_workers
total	1.036	3.905	11.254	5.258	5.700	100.0%	10000	N	1
total	0.187	1.012	5.894	1.61	1.856	100.0%	10000	N	4
total	0.515	0.970	2.076	1.113	1.192	100.0%	10000	Y	NA

3. Concurrency = 32

```
{'context': {'users': {'concurrent': 30,
                       'tenants': 12,
                       'users_per_tenant': 512}},
                       'runner': {'concurrency': 32, 'times': 10000, 'type': 'constant
↪'}}
```

action	min (sec)	avg (sec)	max (sec)	90 percentile	95 percentile	success	count	apache enabled keystone	public_workers
total	1.493	7.752	16.007	10.428	11.183	100.0%	10000	N	1
total	0.198	1.967	8.54	3.223	3.701	100.0%	10000	N	4
total	1.115	1.986	6.224	2.133	2.244	100.0%	10000	Y	NA

Finding a Keystone bug while benchmarking 20 node HA cloud performance at creating 400 VMs

(Contributed by Alexander Maretskiy, Mirantis)

Below we describe how we found a bug in Keystone and achieved 2x average performance increase at booting Nova servers after fixing that bug. Our initial goal was to benchmark the booting of a significant amount of servers on a cluster (running on a custom build of Mirantis OpenStack v5.1) and to ensure that this operation has reasonable performance and completes with no errors.

Goal

- Get data on how a cluster behaves when a huge amount of servers is started
- Get data on how good the neutron component is good in this case

Summary

- Creating 400 servers with configured networking
- Servers are being created simultaneously - 5 servers at the same time

Hardware

Having a real hardware lab with 20 nodes:

Vendor	SUPERMICRO SUPERSERVER
CPU	12 cores, Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
RAM	32GB (4 x Samsung DDRIII 8GB)
HDD	1TB

Cluster

This cluster was created via Fuel Dashboard interface.

Deployment	Custom build of Mirantis OpenStack v5.1
OpenStack release	Icehouse
Operating System	Ubuntu 12.04.4
Mode	High availability
Hypervisor	KVM
Networking	Neutron with GRE segmentation
Controller nodes	3
Compute nodes	17

Rally

Version

For this benchmark, we use custom Rally with the following patch:

https://review.openstack.org/#/c/96300/

Deployment

Rally was deployed for cluster using ExistingCloud type of deployment.

Server flavor

```
$ nova flavor-show ram64
+----------------------------+--------------------------------------+
| Property                   | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 0                                    |
| extra_specs                | {}                                   |
| id                         | 2e46aba0-9e7f-4572-8b0a-b12cfe7e06a1 |
| name                       | ram64                                |
| os-flavor-access:is_public | True                                 |
| ram                        | 64                                   |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
```

Server image

```
$ nova image-show TestVM
+----------------------------+-----------------------------------------------------+
| Property                   | Value                                               |
+----------------------------+-----------------------------------------------------+
| OS-EXT-IMG-SIZE:size       | 13167616                                            |
| created                    | 2014-08-21T11:18:49Z                                |
| id                         | 7a0d90cb-4372-40ef-b711-8f63b0ea9678                |
| metadata murano_image_info | {"title": "Murano Demo", "type": "cirros.demo"}     |
| minDisk                    | 0                                                   |
| minRam                     | 64                                                  |
| name                       | TestVM                                              |
| progress                   | 100                                                 |
| status                     | ACTIVE                                              |
| updated                    | 2014-08-21T11:18:50Z                                |
+----------------------------+-----------------------------------------------------+
```

Task configuration file (in JSON format):

```
{
   "NovaServers.boot_server": [
       {
           "args": {
               "flavor": {
                   "name": "ram64"
               },
               "image": {
                   "name": "TestVM"
```

```
                }
            },
            "runner": {
                "type": "constant",
                "concurrency": 5,
                "times": 400
            },
            "context": {
                "neutron_network": {
                    "network_ip_version": 4
                },
                "users": {
                    "concurrent": 30,
                    "users_per_tenant": 5,
                    "tenants": 5
                },
                "quotas": {
                    "neutron": {
                        "subnet": -1,
                        "port": -1,
                        "network": -1,
                        "router": -1
                    }
                }
            }
        }
    ]
}
```

The only difference between first and second run is that runner.times for first time was set to 500

Results

First time - a bug was found:

Starting from 142 server, we have error from novaclient: **Error <class 'novaclient.exceptions.Unauthorized'>: Unauthorized (HTTP 401).**

That is how a bug in Keystone was found.

action	min (sec)	avg (sec)	max (sec)	90 percentile	95 percentile	success	count
nova.boot_server	6.507	17.402	100.303	39.222	50.134	26.8%	500
total	6.507	17.402	100.303	39.222	50.134	26.8%	500

Second run, with bugfix:

After a patch was applied (using RPC instead of neutron client in metadata agent), we got **100% success and 2x improved average performance**:

action	min (sec)	avg (sec)	max (sec)	90 percentile	95 percentile	success	count
nova.boot_server	5.031	8.008	14.093	9.616	9.716	100.0%	400
total	5.031	8.008	14.093	9.616	9.716	100.0%	400

Installation and upgrades

Installation process

Automated installation

The easiest way to install Rally is by executing its installation script

```
wget -q -O- https://raw.githubusercontent.com/openstack/rally/master/install_rally.sh
↪| bash
# or using curl
curl https://raw.githubusercontent.com/openstack/rally/master/install_rally.sh | bash
```

The installation script will also check if all the software required by Rally is already installed in your system; if run as **root** user and some dependency is missing it will ask you if you want to install the required packages.

By default it will install Rally in a virtualenv in `~/rally` when run as standard user, or install system wide when run as root. You can install Rally in a **venv** by using the option `--target`:

```
./install_rally.sh --target /foo/bar
```

You can also install Rally system wide by running script as root and without `--target` option:

```
sudo ./install_rally.sh
```

Run `./install_rally.sh` with option `--help` to have a list of all available options:

```
 $ ./install_rally.sh --help
 Usage: install_rally.sh [options]

 This script will install rally either in the system (as root) or in a virtual
↪environment.

Options:
  -h, --help              Print this help text
  -v, --verbose           Verbose mode
  -s, --system            Instead of creating a virtualenv, install as
                          system package.
  -d, --target DIRECTORY  Install Rally virtual environment into DIRECTORY.
                          (Default: $HOME/rally).
  -f, --overwrite         Remove target directory if it already exists.
  -y, --yes               Do not ask for confirmation: assume a 'yes' reply
                          to every question.
  -D, --dbtype TYPE       Select the database type. TYPE can be one of
                          'sqlite', 'mysql', 'postgres'.
                          Default: sqlite
  --db-user USER          Database user to use. Only used when --dbtype
                          is either 'mysql' or 'postgres'.
  --db-password PASSWORD  Password of the database user. Only used when
                          --dbtype is either 'mysql' or 'postgres'.
  --db-host HOST          Database host. Only used when --dbtype is
                          either 'mysql' or 'postgres'
  --db-name NAME          Name of the database. Only used when --dbtype is
                          either 'mysql' or 'postgres'
  -p, --python EXE        The python interpreter to use. Default: /usr/bin/python.
```

Notes: the script will check if all the software required by Rally is already installed in your system. If this is not the case, it will exit, suggesting you the command to issue **as root** in order to install the dependencies.

You also have to set up the **Rally database** after the installation is complete:

```
rally-manage db recreate
```

Rally with DevStack all-in-one installation

It is also possible to install Rally with DevStack. First, clone the corresponding repositories:

```
git clone https://git.openstack.org/openstack-dev/devstack
git clone https://github.com/openstack/rally
```

Then, configure DevStack to run Rally. First, create your `local.conf` file:

```
cd devstack
cp samples/local.conf local.conf
```

Next, edit local.conf: add the following line to the `[[local|localrc]]` section.

```
enable_plugin rally https://github.com/openstack/rally master
```

Finally, run DevStack as usually:

```
./stack.sh
```

Rally & Docker

First you need to install Docker; Docker supplies installation instructions for various OSes.

You can either use the official Rally Docker image, or build your own from the Rally source. To do that, change directory to the root directory of the Rally git repository and run:

```
docker build -t myrally .
```

If you build your own Docker image, substitute `myrally` for `rallyforge/rally` in the commands below.

The Rally Docker image is configured to store local settings and the database in the user's home directory. For persistence of these data, you may want to keep this directory outside of the container. This may be done via the following steps:

```
sudo mkdir /var/lib/rally_container
sudo chown 65500 /var/lib/rally_container
docker run -it -v /var/lib/rally_container:/home/rally rallyforge/rally
```

Note: In order for the volume to be accessible by the Rally user (uid: 65500) inside the container, it must be accessible by UID 65500 *outside* the container as well, which is why it is created in `/var/lib/rally`. Creating it in your home directory is only likely to work if your home directory has excessively open permissions (e.g., `0755`), which is not recommended.

You can find all task samples, docs and certification tasks at /opt/rally/. Also you may want to save the last command as an alias:

```
echo 'alias dock_rally="docker run -it -v /var/lib/rally_container:/home/rally␣
↪rallyforge/rally"' >> ~/.bashrc
```

After executing `dock_rally`, or `docker run ...`, you will have bash running inside the container with Rally installed. You may do anything with Rally, but you need to create the database first:

```
user@box:~/rally$ dock_rally
rally@1cc98e0b5941:~$ rally-manage db recreate
rally@1cc98e0b5941:~$ rally deployment list
There are no deployments. To create a new deployment, use:
rally deployment create
rally@1cc98e0b5941:~$
```

In case you have SELinux enabled and Rally fails to create the database, try executing the following commands to put SELinux into Permissive Mode on the host machine

```
sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
setenforce permissive
```

Rally currently has no SELinux policy, which is why it must be run in Permissive mode for certain configurations. If you can help create an SELinux policy for Rally, please contribute!

More about docker: https://www.docker.com/

Database upgrade in Rally

Information for users

Rally supports DB schema versioning (schema versions are called *revisions*) and migration (upgrade to the latest revision).

End user is provided with the following possibilities:

- Print current revision of DB.

  ```
  rally-manage db revision
  ```

- Upgrade existing DB to the latest state.

  This is needed when previously existing Rally installation is being upgraded to a newer version. In this case user should issue command

  ```
  rally-manage db upgrade
  ```

  AFTER upgrading Rally package. DB schema will get upgraded to the latest state and all existing data will be kept.

  WARNING Rally does NOT support DB schema downgrade. One should consider backing up existing database in order to be able to rollback the change.

Information for developers

DB migration in Rally is implemented via package *alembic*.

It is highly recommended to get familiar with it's documentation available by the link before proceeding.

If developer is about to change existing DB schema they should create a new DB revision and a migration script with the following command.

```
alembic --config rally/common/db/sqlalchemy/alembic.ini revision -m <Message>
```

or

```
alembic --config rally/common/db/sqlalchemy/alembic.ini revision --autogenerate -m
↪<Message>
```

It will generate migration script – a file named *<UUID>_<Message>.py* located in *rally/common/db/sqlalchemy/migrations/versions*.

Alembic with parameter `--autogenerate` makes some "routine" job for developer, for example it makes some SQLite compatible batch expressions for migrations.

Generated script should then be checked, edited if it is needed to be and added to Rally source tree.

WARNING Even though alembic supports schema downgrade, migration scripts provided along with Rally do not contain actual code for downgrade.

Quick start

This section will guide you through all steps of using Rally - from installation to its advanced usage in different use cases (including running Rally in OpenStack CI system gates to control merges of patches submitted for review on Gerrit code review system).

Rally step-by-step

In the following tutorial, we will guide you step-by-step through different use cases that might occur in Rally, starting with the easy ones and moving towards more complicated cases.

Step 0. Installation

The easiest way to install Rally is by running its installation script:

```
wget -q -O- https://raw.githubusercontent.com/openstack/rally/master/install_rally.sh␣
↪| bash
# or using curl:
curl https://raw.githubusercontent.com/openstack/rally/master/install_rally.sh | bash
```

If you execute the script as regular user, Rally will create a new virtual environment in `~/rally/` and install in it Rally, and will use *sqlite* as database backend. If you execute the script as root, Rally will be installed system wide. For more installation options, please refer to the *installation* page.

Note: Rally requires Python version 2.7 or 3.4.

Now that you have Rally installed, you are ready to start *benchmarking OpenStack with it*!

Step 1. Setting up the environment and running a benchmark from samples

- *Registering an OpenStack deployment in Rally*
- *Benchmarking*
- *Report generation*

In this demo, we will show how to perform some basic operations in Rally, such as registering an OpenStack cloud, benchmarking it and generating benchmark reports.

We assume that you have gone through *Step 0. Installation* and have an already existing OpenStack deployment with Keystone available at <*KEYSTONE_AUTH_URL*>.

Registering an OpenStack deployment in Rally

First, you have to provide Rally with an OpenStack deployment it is going to benchmark. This should be done either through OpenRC files or through deployment configuration files. In case you already have an *OpenRC*, it is extremely simple to register a deployment with the *deployment create* command:

```
$ . openrc admin admin
$ rally deployment create --fromenv --name=existing
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| uuid                                 | created_at                 | name       |␣
↪status           | active |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| 28f90d74-d940-4874-a8ee-04fda59576da | 2015-01-18 00:11:38.059983 | existing   |␣
↪deploy->finished |        |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
Using deployment : <Deployment UUID>
...
```

Alternatively, you can put the information about your cloud credentials into a JSON configuration file (let's call it existing.json). The *deployment create* command has a slightly different syntax in this case:

```
$ rally deployment create --file=existing.json --name=existing
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| uuid                                 | created_at                 | name       |␣
↪status           | active |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| 28f90d74-d940-4874-a8ee-04fda59576da | 2015-01-18 00:11:38.059983 | existing   |␣
↪deploy->finished |        |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
Using deployment : <Deployment UUID>
...
```

Note the last line in the output. It says that the just created deployment is now used by Rally; that means that all the benchmarking operations from now on are going to be performed on this deployment. Later we will show how to switch between different deployments.

Finally, the *deployment check* command enables you to verify that your current deployment is healthy and ready to be benchmarked:

```
$ rally deployment check
keystone endpoints are valid and following services are available:
+----------+----------------+-----------+
| services | type           | status    |
+----------+----------------+-----------+
| cinder   | volume         | Available |
| cinderv2 | volumev2       | Available |
| ec2      | ec2            | Available |
| glance   | image          | Available |
| heat     | orchestration  | Available |
| heat-cfn | cloudformation | Available |
| keystone | identity       | Available |
| nova     | compute        | Available |
| novav21  | computev21     | Available |
| s3       | s3             | Available |
+----------+----------------+-----------+
```

Benchmarking

Now that we have a working and registered deployment, we can start benchmarking it. The sequence of benchmarks to be launched by Rally should be specified in a *benchmark task configuration file* (either in *JSON* or in *YAML* format). Let's try one of the sample benchmark tasks available in samples/tasks/scenarios, say, the one that boots and deletes multiple servers (*samples/tasks/scenarios/nova/boot-and-delete.json*):

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                "flavor": {
                    "name": "m1.tiny"
                },
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "force_delete": false
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                }
            }
        }
    ]
}
```

To start a benchmark task, run the `task start` command (you can also add the *-v* option to print more logging information):

```
$ rally task start samples/tasks/scenarios/nova/boot-and-delete.json
--------------------------------------------------------------------------------
```

```
 Preparing input task
--------------------------------------------------------------------------------

Input task is:
<Your task config here>

--------------------------------------------------------------------------------
 Task  6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996: started
--------------------------------------------------------------------------------

Benchmarking... This can take a while...

To track task status use:

  rally task status
  or
  rally task detailed

--------------------------------------------------------------------------------
 Task 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996: finished
--------------------------------------------------------------------------------

test scenario NovaServers.boot_and_delete_server
args position 0
args values:
{u'args': {u'flavor': {u'name': u'm1.tiny'},
           u'force_delete': False,
           u'image': {u'name': u'^cirros.*-disk$'}},
 u'context': {u'users': {u'project_domain': u'default',
                         u'resource_management_workers': 30,
                         u'tenants': 3,
                         u'user_domain': u'default',
                         u'users_per_tenant': 2}},
 u'runner': {u'concurrency': 2, u'times': 10, u'type': u'constant'}}
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| action             | min (sec) | avg (sec) | max (sec) | 90 percentile | 95␣
↪percentile | success | count |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| nova.boot_server   | 7.99      | 9.047     | 11.862    | 9.747         | 10.805     ␣
↪    | 100.0%  | 10    |
| nova.delete_server | 4.427     | 4.574     | 4.772     | 4.677         | 4.725      ␣
↪    | 100.0%  | 10    |
| total              | 12.556    | 13.621    | 16.37     | 14.252        | 15.311     ␣
↪    | 100.0%  | 10    |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
Load duration: 70.1310448647
Full duration: 87.545541048


HINTS:
* To plot HTML graphics with this data, run:
    rally task report 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996 --out output.html

* To get raw JSON output of task results, run:
    rally task results 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996
```

```
Using task: 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996
```

Note that the Rally input task above uses *regular expressions* to specify the image and flavor name to be used for server creation, since concrete names might differ from installation to installation. If this benchmark task fails, then the reason for that might a non-existing image/flavor specified in the task. To check what images/flavors are available in the deployment you are currently benchmarking, you might use the *rally show* command:

```
$ rally show images
+--------------------------------------+-----------------------+------------+
|                 UUID                 |          Name         |  Size (B)  |
+--------------------------------------+-----------------------+------------+
| 8dfd6098-0c26-4cb5-8e77-1ecb2db0b8ae |  CentOS 6.5 (x86_64)  | 344457216 |
| 2b8d119e-9461-48fc-885b-1477abe2edc5 | CirrOS 0.3.4 (x86_64) |  13287936 |
+--------------------------------------+-----------------------+------------+

$ rally show flavors

Flavors for user `admin` in tenant `admin`:
+----+-----------+-------+-----------+-----------+-----------+
| ID | Name      | vCPUs | RAM (MB)  | Swap (MB) | Disk (GB) |
+----+-----------+-------+-----------+-----------+-----------+
| 1  | m1.tiny   | 1     | 512       |           | 1         |
| 2  | m1.small  | 1     | 2048      |           | 20        |
| 3  | m1.medium | 2     | 4096      |           | 40        |
| 4  | m1.large  | 4     | 8192      |           | 80        |
| 5  | m1.xlarge | 8     | 16384     |           | 160       |
+----+-----------+-------+-----------+-----------+-----------+
```

Report generation

One of the most beautiful things in Rally is its task report generation mechanism. It enables you to create illustrative and comprehensive HTML reports based on the benchmarking data. To create and open at once such a report for the last task you have launched, call:

```
rally task report --out=report1.html --open
```

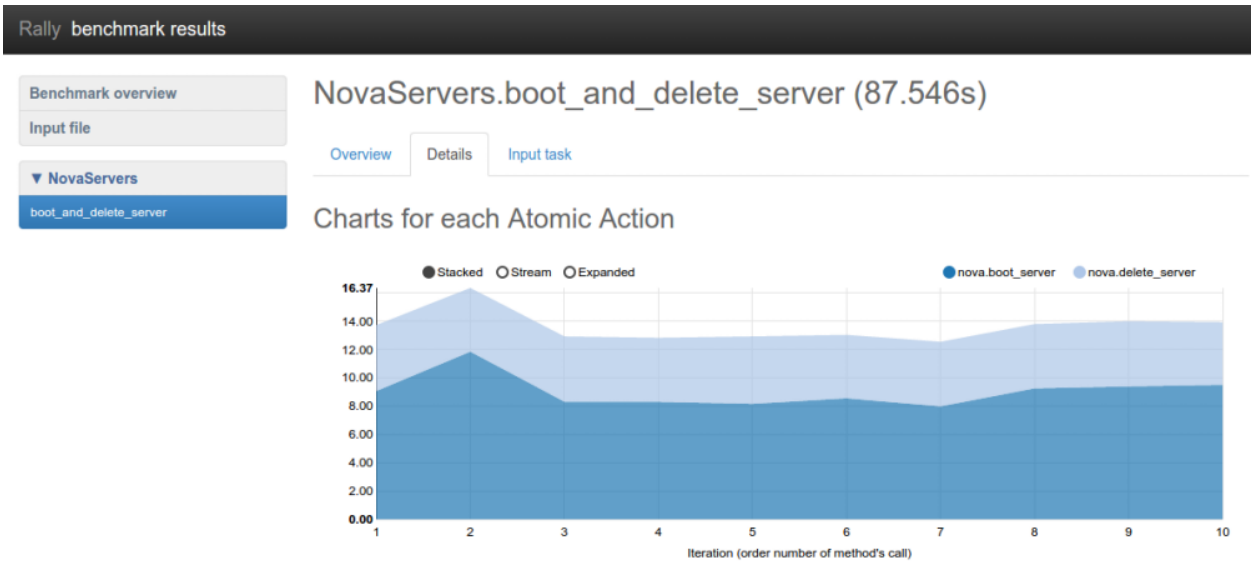
This will produce an HTML page with the overview of all the scenarios that you've included into the last benchmark task completed in Rally (in our case, this is just one scenario, and we will cover the topic of multiple scenarios in one task in *the next step of our tutorial*):

This aggregating table shows the duration of the load produced by the corresponding scenario (*"Load duration"*), the overall benchmark scenario execution time, including the duration of environment preparation with contexts (*"Full duration"*), the number of iterations of each scenario (*"Iterations"*), the type of the load used while running the scenario (*"Runner"*), the number of failed iterations (*"Errors"*) and finally whether the scenario has passed certain

Success Criteria (*"SLA"*) that were set up by the user in the input configuration file (we will cover these criteria in *one of the next steps*).

By navigating in the left panel, you can switch to the detailed view of the benchmark results for the only scenario we included into our task, namely **NovaServers.boot_and_delete_server**:

This page, along with the description of the success criteria used to check the outcome of this scenario, shows more detailed information and statistics about the duration of its iterations. Now, the *"Total durations"* table splits the duration of our scenario into the so-called **"atomic actions"**: in our case, the **"boot_and_delete_server"** scenario consists of two actions - **"boot_server"** and **"delete_server"**. You can also see how the scenario duration changed throughout its iterations in the *"Charts for the total duration"* section. Similar charts, but with atomic actions detailed are on the *"Details"* tab of this page:



Note that all the charts on the report pages are very dynamic: you can change their contents by clicking the switches above the graph and see more information about its single points by hovering the cursor over these points.

Take some time to play around with these graphs and then move on to *the next step of our tutorial*.

Step 2. Rally input task format

- *Basic input task syntax*
- *Multiple benchmarks in a single task*
- *Multiple configurations of the same scenario*

Basic input task syntax

Rally comes with a really great collection of *plugins* and in most real-world cases you will use multiple plugins to test your OpenStack cloud. Rally makes it very easy to run **different test cases defined in a single task**. To do so, use the following syntax:

```
{
    "<ScenarioName1>": [<benchmark_config>, <benchmark_config2>, ...]
    "<ScenarioName2>": [<benchmark_config>, ...]
}
```

where <*benchmark_config*>, as before, is a dictionary:

```
{
    "args": { <scenario-specific arguments> },
    "runner": { <type of the runner and its specific parameters> },
    "context": { <contexts needed for this scenario> },
    "sla": { <different SLA configs> }
}
```

Multiple benchmarks in a single task

As an example, let's edit our configuration file from *step 1* so that it prescribes Rally to launch not only the **NovaServers.boot_and_delete_server** scenario, but also the **KeystoneBasic.create_delete_user** scenario. All we have to do is to append the configuration of the second scenario as yet another top-level key of our JSON file:

multiple-scenarios.json

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                "flavor": {
                    "name": "m1.tiny"
                },
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "force_delete": false
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                }
            }
        }
    ],
    "KeystoneBasic.create_delete_user": [
        {
            "args": {},
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 3
            }
        }
    ]
}
```

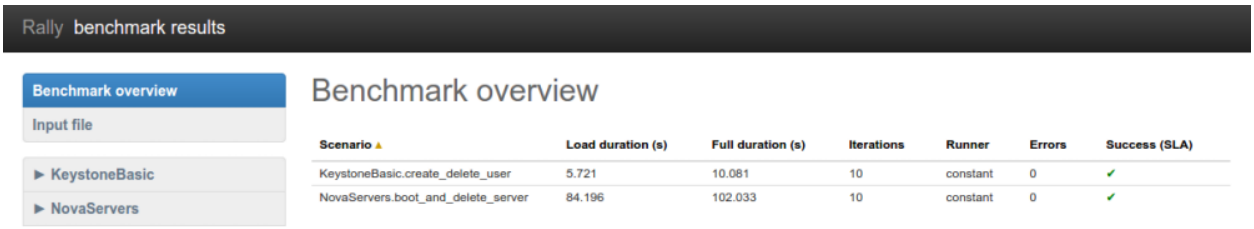
Now you can start this benchmark task as usually:

```
$ rally task start multiple-scenarios.json
...
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| action             | min (sec) | avg (sec) | max (sec) | 90 percentile | 95␣
↪percentile | success | count |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| nova.boot_server   | 8.06      | 11.354    | 18.594    | 18.54         | 18.567     ␣
↪   | 100.0%  | 10    |
| nova.delete_server | 4.364     | 5.054     | 6.837     | 6.805         | 6.821      ␣
↪   | 100.0%  | 10    |
```

```
| total                | 12.572    | 16.408    | 25.396    | 25.374         | 25.385
↪    | 100.0%  | 10    |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
Load duration: 84.1959171295
Full duration: 102.033041
-------------------------------------------------------------------------------

...

+----------------------+-----------+-----------+-----------+---------------+----------
↪-----+---------+-------+
| action               | min (sec) | avg (sec) | max (sec) | 90 percentile | 95
↪percentile | success | count |
+----------------------+-----------+-----------+-----------+---------------+----------
↪-----+---------+-------+
| keystone.create_user | 0.676     | 0.875     | 1.03      | 1.02          | 1.025
↪     | 100.0%  | 10    |
| keystone.delete_user | 0.407     | 0.647     | 0.84      | 0.739         | 0.79
↪     | 100.0%  | 10    |
| total                | 1.082     | 1.522     | 1.757     | 1.724         | 1.741
↪     | 100.0%  | 10    |
+----------------------+-----------+-----------+-----------+---------------+----------
↪-----+---------+-------+
Load duration: 5.72119688988
Full duration: 10.0808410645

...
```

Note that the HTML reports you can generate by typing **rally task report –out=report_name.html** after your benchmark task has completed will get richer as your benchmark task configuration file includes more benchmark scenarios. Let's take a look at the report overview page for a task that covers all the scenarios available in Rally:

```
rally task report --out=report_multiple_scenarios.html --open
```



Multiple configurations of the same scenario

Yet another thing you can do in Rally is to launch **the same benchmark scenario multiple times with different configurations**. That's why our configuration file stores a list for the key *"NovaServers.boot_and_delete_server"*: you can just append a different configuration of this benchmark scenario to this list to get it. Let's say, you want to run the **boot_and_delete_server** scenario twice: first using the *"m1.tiny"* flavor and then using the *"m1.small"* flavor:

multiple-configurations.json

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                "flavor": {
                    "name": "m1.tiny"
                },
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "force_delete": false
            },
            "runner": {...},
            "context": {...}
        },
        {
            "args": {
                "flavor": {
                    "name": "m1.small"
                },
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "force_delete": false
            },
            "runner": {...},
            "context": {...}
        }
    ]
}
```

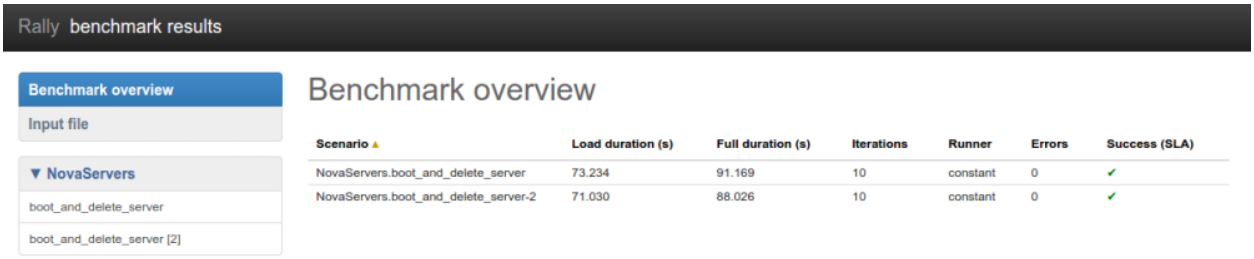
That's it! You will get again the results for each configuration separately:

```
$ rally task start --task=multiple-configurations.json
...
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| action             | min (sec) | avg (sec) | max (sec) | 90 percentile | 95␣
↪percentile | success | count |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| nova.boot_server   | 7.896     | 9.433     | 13.14     | 11.329        | 12.234     ␣
↪   | 100.0%  | 10    |
| nova.delete_server | 4.435     | 4.898     | 6.975     | 5.144         | 6.059      ␣
↪   | 100.0%  | 10    |
| total              | 12.404    | 14.331    | 17.979    | 16.72         | 17.349     ␣
↪   | 100.0%  | 10    |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
Load duration: 73.2339417934
Full duration: 91.1692159176
-------------------------------------------------------------------------------

...

+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| action             | min (sec) | avg (sec) | max (sec) | 90 percentile | 95␣
↪percentile | success | count |
```

```
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
| nova.boot_server   | 8.207     | 8.91      | 9.823     | 9.692         | 9.758
↪   | 100.0%  | 10    |
| nova.delete_server | 4.405     | 4.767     | 6.477     | 4.904         | 5.691
↪   | 100.0%  | 10    |
| total              | 12.735    | 13.677    | 16.301    | 14.596        | 15.449
↪   | 100.0%  | 10    |
+--------------------+-----------+-----------+-----------+---------------+------------
↪---+---------+-------+
Load duration: 71.029528141
Full duration: 88.0259010792
...
```

The HTML report will also look similar to what we have seen before:

```
rally task report --out=report_multiple_configuraions.html --open
```



Step 3. Benchmarking OpenStack with existing users

Motivation

There are two very important reasons from the production world of why it is preferable to use some already existing users to benchmark your OpenStack cloud:

1. *Read-only Keystone Backends:* creating temporary users for benchmark scenarios in Rally is just impossible in case of r/o Keystone backends like *LDAP* and *AD*.

2. *Safety:* Rally can be run from an isolated group of users, and if something goes wrong, this won't affect the rest of the cloud users.

Registering existing users in Rally

The information about existing users in your OpenStack cloud should be passed to Rally at the *deployment initialization step*. You have to use the **ExistingCloud** deployment plugin that just provides Rally with credentials of an already existing cloud. The difference from the deployment configuration we've seen previously is that you should set up the *"users"* section with the credentials of already existing users. Let's call this deployment configuration file *existing_users.json*:

```
{
    "type": "ExistingCloud",
    "auth_url": "http://example.net:5000/v2.0/",
    "region_name": "RegionOne",
    "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "pa55word",
        "tenant_name": "demo"
    },
    "users": [
        {
            "username": "b1",
            "password": "1234",
            "tenant_name": "testing"
        },
        {
            "username": "b2",
            "password": "1234",
            "tenant_name": "testing"
        }
    ]
}
```

This deployment configuration requires some basic information about the OpenStack cloud like the region name, auth url. admin user credentials, and any amount of users already existing in the system. Rally will use their credentials to generate load in against this deployment as soon as we register it as usual:

```
$ rally deployment create --file existings_users --name our_cloud
+--------------------------------------+----------------------------+-----------+-----
↪-------------+--------+
| uuid                                 | created_at                 | name      |␣
↪status           | active |
+--------------------------------------+----------------------------+-----------+-----
↪-------------+--------+
| 1849a9bf-4b18-4fd5-89f0-ddcc56eae4c9 | 2015-03-28 02:43:27.759702 | our_cloud |␣
↪deploy->finished |        |
+--------------------------------------+----------------------------+-----------+-----
↪-------------+--------+
Using deployment: 1849a9bf-4b18-4fd5-89f0-ddcc56eae4c9
~/.rally/openrc was updated
```

After that, the **rally show** command lists the resources for each user separately:

```
$ rally show images

Images for user `admin` in tenant `admin`:
+--------------------------------------+----------------------------------+------------+
| UUID                                 | Name                             | Size (B)   |
+--------------------------------------+----------------------------------+------------+
```

```
| 041cfd70-0e90-4ed6-8c0c-ad9c12a94191 | cirros-0.3.4-x86_64-uec          | 25165824  |
| 87710f09-3625-4496-9d18-e20e34906b72 | Fedora-x86_64-20-20140618-sda    | 209649664 |
| b0f269be-4859-48e0-a0ca-03fb80d14602 | cirros-0.3.4-x86_64-uec-ramdisk | 3740163   |
| d82eaf7a-ff63-4826-9aa7-5fa105610e01 | cirros-0.3.4-x86_64-uec-kernel  | 4979632   |
+--------------------------------------+----------------------------------+-----------+

Images for user `b1` in tenant `testing`:
+--------------------------------------+----------------------------------+-----------+
| UUID                                 | Name                             | Size (B)  |
+--------------------------------------+----------------------------------+-----------+
| 041cfd70-0e90-4ed6-8c0c-ad9c12a94191 | cirros-0.3.4-x86_64-uec          | 25165824  |
| 87710f09-3625-4496-9d18-e20e34906b72 | Fedora-x86_64-20-20140618-sda    | 209649664 |
| b0f269be-4859-48e0-a0ca-03fb80d14602 | cirros-0.3.4-x86_64-uec-ramdisk | 3740163   |
| d82eaf7a-ff63-4826-9aa7-5fa105610e01 | cirros-0.3.4-x86_64-uec-kernel  | 4979632   |
+--------------------------------------+----------------------------------+-----------+

Images for user `b2` in tenant `testing`:
+--------------------------------------+----------------------------------+-----------+
| UUID                                 | Name                             | Size (B)  |
+--------------------------------------+----------------------------------+-----------+
| 041cfd70-0e90-4ed6-8c0c-ad9c12a94191 | cirros-0.3.4-x86_64-uec          | 25165824  |
| 87710f09-3625-4496-9d18-e20e34906b72 | Fedora-x86_64-20-20140618-sda    | 209649664 |
| b0f269be-4859-48e0-a0ca-03fb80d14602 | cirros-0.3.4-x86_64-uec-ramdisk | 3740163   |
| d82eaf7a-ff63-4826-9aa7-5fa105610e01 | cirros-0.3.4-x86_64-uec-kernel  | 4979632   |
+--------------------------------------+----------------------------------+-----------+
```

With this new deployment being active, Rally will use the already existing users *"b1"* and *"b2"* instead of creating the temporary ones when launching benchmark task that do not specify the *"users"* context.

Running benchmark scenarios with existing users

After you have registered a deployment with existing users, don't forget to remove the *"users"* context from your benchmark task configuration if you want to use existing users, like in the following configuration file (*boot-and-delete.json*):

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                "flavor": {
                    "name": "m1.tiny"
                },
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "force_delete": false
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 2
            },
            "context": {}
        }
    ]
}
```

When you start this task, it will use the existing users *"b1"* and *"b2"* instead of creating the temporary ones:

```
rally task start samples/tasks/scenarios/nova/boot-and-delete.json
```

It goes without saying that support of benchmarking with predefined users simplifies the usage of Rally for generating loads against production clouds.

(based on: http://boris-42.me/rally-can-generate-load-with-passed-users-now/)

Step 4. Adding success criteria (SLA) for benchmarks

SLA - Service-Level Agreement (Success Criteria)

Rally allows you to set success criteria (also called *SLA - Service-Level Agreement*) for every benchmark. Rally will automatically check them for you.

To configure the SLA, add the *"sla"* section to the configuration of the corresponding benchmark (the check name is a key associated with its target value). You can combine different success criteria:

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                ...
            },
            "runner": {
                ...
            },
            "context": {
                ...
            },
            "sla": {
                "max_seconds_per_iteration": 10,
                "failure_rate": {
                    "max": 25
                }
            }
        }
    ]
}
```

Such configuration will mark the **NovaServers.boot_and_delete_server** benchmark scenario as not successful if either some iteration took more than 10 seconds or more than 25% iterations failed.

Checking SLA

Let us show you how Rally SLA work using a simple example based on **Dummy benchmark scenarios**. These scenarios actually do not perform any OpenStack-related stuff but are very useful for testing the behaviors of Rally. Let us put in a new task, *test-sla.json*, 2 scenarios – one that does nothing and another that just throws an exception:

```
{
    "Dummy.dummy": [
        {
            "args": {},
            "runner": {
                "type": "constant",
                "times": 5,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                }
            },
            "sla": {
                "failure_rate": {"max": 0.0}
            }
        }
    ],
    "Dummy.dummy_exception": [
        {
            "args": {},
            "runner": {
                "type": "constant",
                "times": 5,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                }
            },
            "sla": {
                "failure_rate": {"max": 0.0}
            }
        }
    ]
}
```

Note that both scenarios in these tasks have the **maximum failure rate of 0%** as their **success criterion**. We expect that the first scenario will pass this criterion while the second will fail it. Let's start the task:

```
rally task start test-sla.json
```

After the task completes, run *rally task sla_check* to check the results again the success criteria you defined in the task:

```
$ rally task sla_check
+----------------------+-----+-------------+--------+-------------------------------
↪-------------------------------------------------------------------------+
| benchmark            | pos | criterion   | status | detail                        ␣
↪                                                                          |
```

```
+----------------------+-----+--------------+--------+-------------------------------
↪-------------------------------------------------------------------------+
| Dummy.dummy            | 0   | failure_rate | PASS   | Maximum failure rate percent␣
↪0.0% failures, minimum failure rate percent 0% failures, actually 0.0%   |
| Dummy.dummy_exception | 0   | failure_rate | FAIL   | Maximum failure rate percent␣
↪0.0% failures, minimum failure rate percent 0% failures, actually 100.0% |
+----------------------+-----+--------------+--------+-------------------------------
↪-------------------------------------------------------------------------+
```

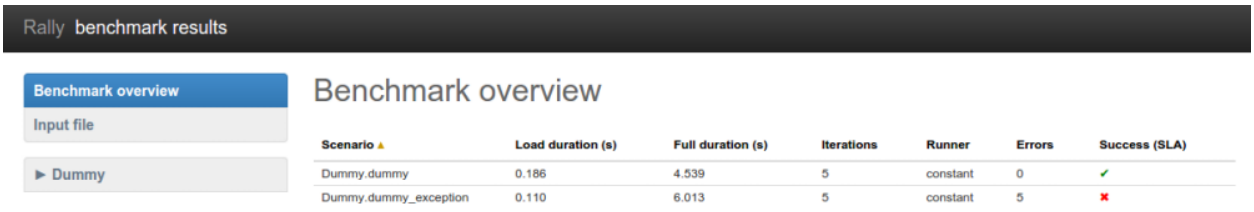
Exactly as expected.

SLA in task report

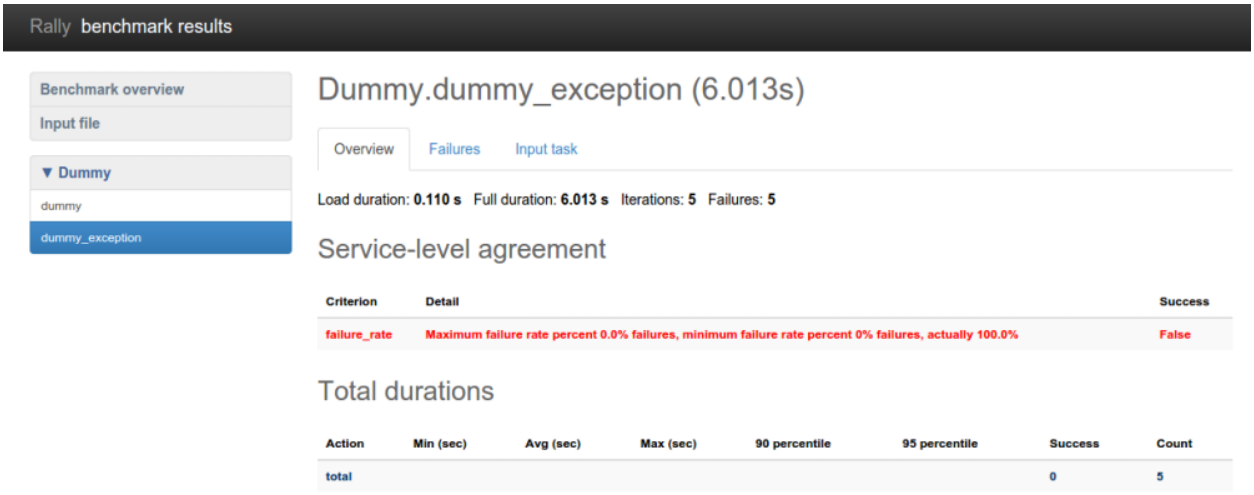
SLA checks are nicely visualized in task reports. Generate one:

```
rally task report --out=report_sla.html --open
```

Benchmark scenarios that have passed SLA have a green check on the overview page:



Somewhat more detailed information about SLA is displayed on the scenario pages:



Success criteria present a very useful concept that enables not only to analyze the outcome of your benchmark tasks, but also to control their execution. In *one of the next sections* of our tutorial, we will show how to use SLA to abort the load generation before your OpenStack goes wrong.

Step 5. Rally task templates

Basic template syntax

A nice feature of the input task format used in Rally is that it supports the **template syntax** based on Jinja2. This turns out to be extremely useful when, say, you have a fixed structure of your task but you want to parameterize this task in some way. For example, imagine your input task file (*task.yaml*) runs a set of Nova scenarios:

```
---
  NovaServers.boot_and_delete_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: "^cirros.*-disk$"
      runner:
        type: "constant"
        times: 2
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

  NovaServers.resize_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: "^cirros.*-disk$"
        to_flavor:
            name: "m1.small"
      runner:
        type: "constant"
        times: 3
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1
```

In both scenarios above, the *"^cirros.*-disk$"** image is passed to the scenario as an argument (so that these scenarios use an appropriate image while booting servers). Let's say you want to run the same set of scenarios with the same runner/context/sla, but you want to try another image while booting server to compare the performance. The most elegant solution is then to turn the image name into a template variable:

```
---
  NovaServers.boot_and_delete_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: {{image_name}}
      runner:
        type: "constant"
        times: 2
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

  NovaServers.resize_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: {{image_name}}
        to_flavor:
            name: "m1.small"
      runner:
        type: "constant"
        times: 3
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1
```

and then pass the argument value for **{{image_name}}** when starting a task with this configuration file. Rally provides you with different ways to do that:

1. Pass the argument values directly in the command-line interface (with either a JSON or YAML dictionary):

```
rally task start task.yaml --task-args '{"image_name": "^cirros.*-disk$"}'
rally task start task.yaml --task-args 'image_name: "^cirros.*-disk$"'
```

2. Refer to a file that specifies the argument values (JSON/YAML):

```
rally task start task.yaml --task-args-file args.json
rally task start task.yaml --task-args-file args.yaml
```

where the files containing argument values should look as follows:

args.json:

```
{
    "image_name": "^cirros.*-disk$"
}
```

args.yaml:

```
---
  image_name: "^cirros.*-disk$"
```

Passed in either way, these parameter values will be substituted by Rally when starting a task:

```
$ rally task start task.yaml --task-args "image_name: "^cirros.*-disk$""
--------------------------------------------------------------------------------
 Preparing input task
--------------------------------------------------------------------------------

Input task is:
---

  NovaServers.boot_and_delete_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: ^cirros.*-disk$
      runner:
        type: "constant"
        times: 2
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

  NovaServers.resize_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: ^cirros.*-disk$
        to_flavor:
            name: "m1.small"
      runner:
        type: "constant"
        times: 3
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

--------------------------------------------------------------------------------
 Task  cbf7eb97-0f1d-42d3-a1f1-3cc6f45ce23f: started
--------------------------------------------------------------------------------

Benchmarking... This can take a while...
```

Using the default values

Note that the `Jinja2` template syntax allows you to set the default values for your parameters. With default values set, your task file will work even if you don't parameterize it explicitly while starting a task. The default values should be set using the *{% set ... %}* clause (*task.yaml*):

```
{% set image_name = image_name or "^cirros.*-disk$" %}
---

  NovaServers.boot_and_delete_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: {{image_name}}
      runner:
        type: "constant"
        times: 2
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

    ...
```

If you don't pass the value for *{{image_name}}* while starting a task, the default one will be used:

```
$ rally task start task.yaml
--------------------------------------------------------------------------------
 Preparing input task
--------------------------------------------------------------------------------

Input task is:
---

  NovaServers.boot_and_delete_server:
    -
      args:
        flavor:
            name: "m1.tiny"
        image:
            name: ^cirros.*-disk$
      runner:
        type: "constant"
        times: 2
        concurrency: 1
      context:
        users:
          tenants: 1
          users_per_tenant: 1

    ...
```

Advanced templates

Rally makes it possible to use all the power of `Jinja2` template syntax, including the mechanism of **built-in functions**. This enables you to construct elegant task files capable of generating complex load on your cloud.

As an example, let us make up a task file that will create new users with increasing concurrency. The input task file (*task.yaml*) below uses the `Jinja2` **for-endfor** construct to accomplish that:

```
---
  KeystoneBasic.create_user:
  {% for i in range(2, 11, 2) %}
    -
      args: {}
      runner:
        type: "constant"
        times: 10
        concurrency: {{i}}
      sla:
        failure_rate:
          max: 0
  {% endfor %}
```

In this case, you don't need to pass any arguments via *–task-args/–task-args-file*, but as soon as you start this task, Rally will automatically unfold the for-loop for you:

```
$ rally task start task.yaml
--------------------------------------------------------------------------------
 Preparing input task
--------------------------------------------------------------------------------

Input task is:
---

  KeystoneBasic.create_user:

    -
      args: {}
      runner:
        type: "constant"
        times: 10
        concurrency: 2
      sla:
        failure_rate:
          max: 0

    -
      args: {}
      runner:
        type: "constant"
        times: 10
        concurrency: 4
      sla:
        failure_rate:
          max: 0

    -
      args: {}
      runner:
```

```
        type: "constant"
        times: 10
        concurrency: 6
      sla:
        failure_rate:
          max: 0

    -
      args: {}
      runner:
        type: "constant"
        times: 10
        concurrency: 8
      sla:
        failure_rate:
          max: 0

    -
      args: {}
      runner:
        type: "constant"
        times: 10
        concurrency: 10
      sla:
        failure_rate:
          max: 0


--------------------------------------------------------------------------------
 Task  ea7e97e3-dd98-4a81-868a-5bb5b42b8610: started
--------------------------------------------------------------------------------

Benchmarking... This can take a while...
```

As you can see, the Rally task template syntax is a simple but powerful mechanism that not only enables you to write elegant task configurations, but also makes them more readable for other people. When used appropriately, it can really improve the understanding of your benchmarking procedures in Rally when shared with others.

Step 6. Aborting load generation on success criteria failure

Benchmarking pre-production and production OpenStack clouds is not a trivial task. From the one side it is important to reach the OpenStack cloud's limits, from the other side the cloud shouldn't be damaged. Rally aims to make this task as simple as possible. Since the very beginning Rally was able to generate enough load for any OpenStack cloud. Generating too big a load was the major issue for production clouds, because Rally didn't know how to stop the load until it was too late.

With the **"stop on SLA failure"** feature, however, things are much better.

This feature can be easily tested in real life by running one of the most important and plain benchmark scenario called *"Authenticate.keystone"*. This scenario just tries to authenticate from users that were pre-created by Rally. Rally input task looks as follows (*auth.yaml*):

```
---
  Authenticate.keystone:
  -
    runner:
      type: "rps"
```

```
      times: 6000
      rps: 50
    context:
      users:
        tenants: 5
        users_per_tenant: 10
    sla:
      max_avg_duration: 5
```

In human-readable form this input task means: *Create 5 tenants with 10 users in each, after that try to authenticate to Keystone 6000 times performing 50 authentications per second (running new authentication request every 20ms). Each time we are performing authentication from one of the Rally pre-created user. This task passes only if max average duration of authentication takes less than 5 seconds.*

Note that this test is quite dangerous because it can DDoS Keystone. We are running more and more simultaneously authentication requests and things may go wrong if something is not set properly (like on my DevStack deployment in Small VM on my laptop).

Let's run Rally task with **an argument that prescribes Rally to stop load on SLA failure**:

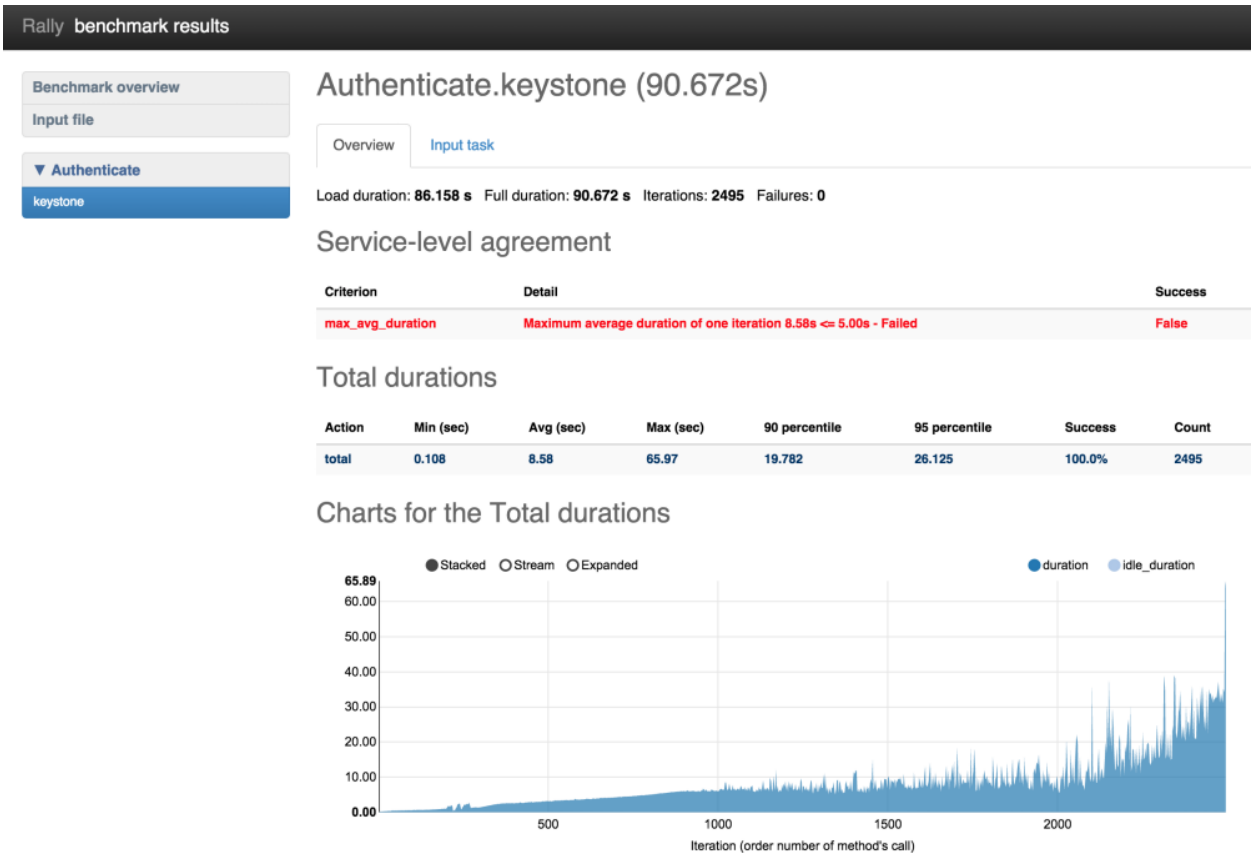
```
$ rally task start --abort-on-sla-failure auth.yaml

....
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
| action | min (sec) | avg (sec) | max (sec) | 90 percentile | 95 percentile |␣
↪success | count |
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
| total  | 0.108     | 8.58      | 65.97     | 19.782        | 26.125        | 100.0%␣
↪ | 2495  |
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
```

On the resulting table there are 2 interesting things:

1. Average duration was 8.58 sec which is more than 5 seconds
2. Rally performed only 2495 (instead of 6000) authentication requests

To understand better what has happened let's generate HTML report:

```
rally task report --out auth_report.html
```



On the chart with durations we can observe that the duration of authentication request reaches 65 seconds at the end of the load generation. **Rally stopped load at the very last moment just before bad things happened. The reason why it runs so many attempts to authenticate is because of not enough good success criteria.** We had to run a lot of iterations to make average duration bigger than 5 seconds. Let's chose better success criteria for this task and run it one more time.

```
---
  Authenticate.keystone:
  -
    runner:
      type: "rps"
      times: 6000
      rps: 50
    context:
      users:
        tenants: 5
        users_per_tenant: 10
    sla:
      max_avg_duration: 5
      max_seconds_per_iteration: 10
      failure_rate:
        max: 0
```

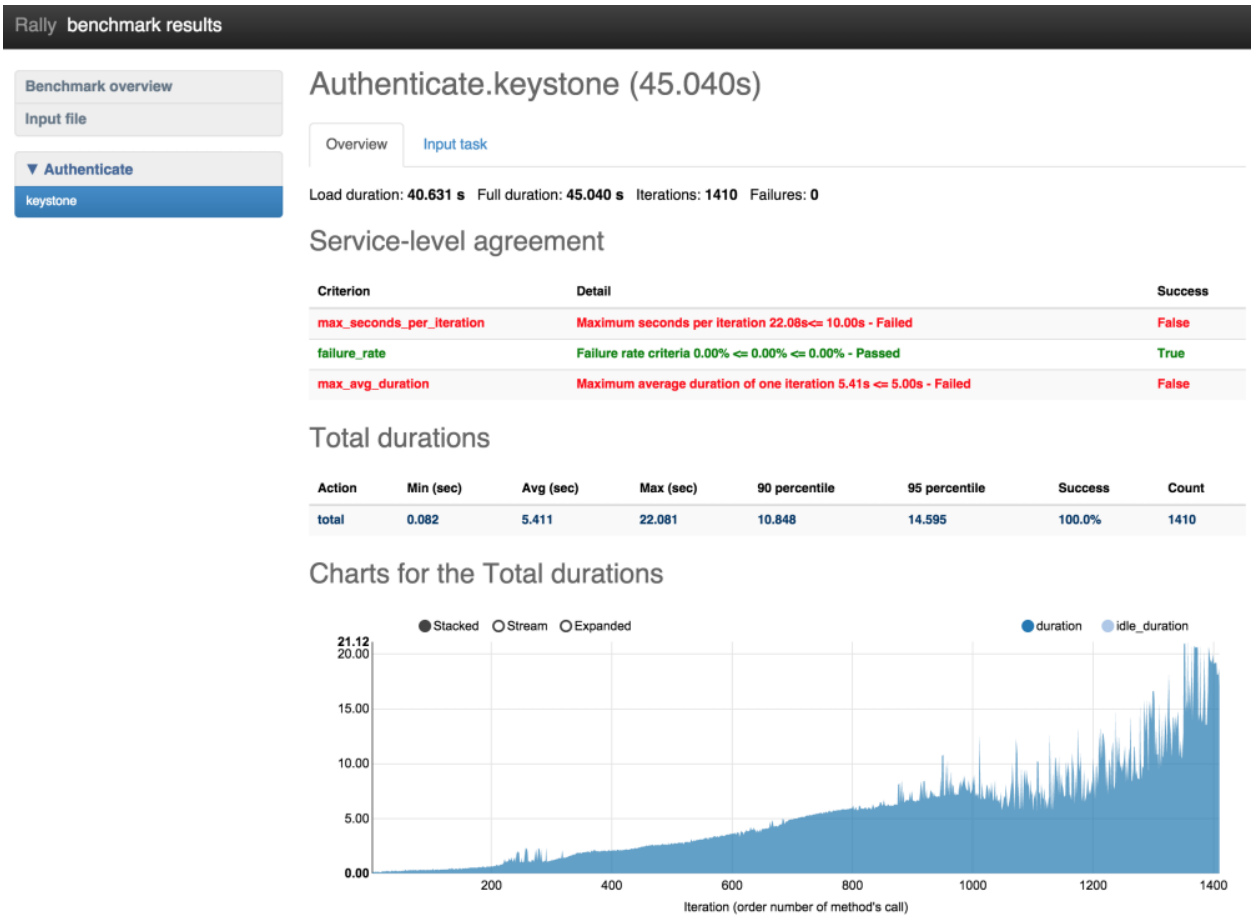
Now our task is going to be successful if the following three conditions hold:

1. maximum average duration of authentication should be less than 5 seconds
2. maximum duration of any authentication should be less than 10 seconds

3. no failed authentication should appear

Let's run it!

```
$ rally task start --abort-on-sla-failure auth.yaml

...
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
| action | min (sec) | avg (sec) | max (sec) | 90 percentile | 95 percentile |␣
↪success | count |
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
| total  | 0.082     | 5.411     | 22.081    | 10.848        | 14.595        | 100.0%␣
↪ | 1410  |
+--------+-----------+-----------+-----------+---------------+---------------+--------
↪-+-------+
```



This time load stopped after 1410 iterations versus 2495 which is much better. The interesting thing on this chart is that first occurrence of "> 10 second" authentication happened on 950 iteration. The reasonable question: "Why does Rally run 500 more authentication requests then?". This appears from the math: During the execution of **bad** authentication (10 seconds) Rally performed about 50 request/sec * 10 sec = 500 new requests as a result we run 1400 iterations instead of 950.

(based on: http://boris-42.me/rally-tricks-stop-load-before-your-openstack-goes-wrong/)

Step 7. Working with multiple OpenStack clouds

Rally is an awesome tool that allows you to work with multiple clouds and can itself deploy them. We already know how to work with *a single cloud*. Let us now register 2 clouds in Rally: the one that we have access to and the other that we know is registered with wrong credentials.

```
$ . openrc admin admin  # openrc with correct credentials
$ rally deployment create --fromenv --name=cloud-1
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| uuid                                 | created_at                 | name       |␣
↪status           | active |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| 4251b491-73b2-422a-aecb-695a94165b5e | 2015-01-18 00:11:14.757203 | cloud-1    |␣
↪deploy->finished |        |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
Using deployment: 4251b491-73b2-422a-aecb-695a94165b5e
~/.rally/openrc was updated
...

$ . bad_openrc admin admin  # openrc with wrong credentials
$ rally deployment create --fromenv --name=cloud-2
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| uuid                                 | created_at                 | name       |␣
↪status           | active |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| 658b9bae-1f9c-4036-9400-9e71e88864fc | 2015-01-18 00:38:26.127171 | cloud-2    |␣
↪deploy->finished |        |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
Using deployment: 658b9bae-1f9c-4036-9400-9e71e88864fc
~/.rally/openrc was updated
...
```

Let us now list the deployments we have created:

```
$ rally deployment list
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| uuid                                 | created_at                 | name       |␣
↪status           | active |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
| 4251b491-73b2-422a-aecb-695a94165b5e | 2015-01-05 00:11:14.757203 | cloud-1    |␣
↪deploy->finished |        |
| 658b9bae-1f9c-4036-9400-9e71e88864fc | 2015-01-05 00:40:58.451435 | cloud-2    |␣
↪deploy->finished | *      |
+--------------------------------------+----------------------------+------------+----
↪--------------+--------+
```

Note that the second is marked as **"active"** because this is the deployment we have created most recently. This means that it will be automatically (unless its UUID or name is passed explicitly via the *–deployment* parameter) used by the commands that need a deployment, like *rally task start ...* or *rally deployment check*:

```
$ rally deployment check
Authentication Issues: wrong keystone credentials specified in your endpoint␣
↪properties. (HTTP 401).

$ rally deployment check --deployment=cloud-1
keystone endpoints are valid and following services are available:
+----------+----------------+-----------+
| services | type           | status    |
+----------+----------------+-----------+
| cinder   | volume         | Available |
| cinderv2 | volumev2       | Available |
| ec2      | ec2            | Available |
| glance   | image          | Available |
| heat     | orchestration  | Available |
| heat-cfn | cloudformation | Available |
| keystone | identity       | Available |
| nova     | compute        | Available |
| novav21  | computev21     | Available |
| s3       | s3             | Available |
+----------+----------------+-----------+
```

You can also switch the active deployment using the **rally deployment use** command:

```
$ rally deployment use cloud-1
Using deployment: 658b9bae-1f9c-4036-9400-9e71e88864fc
~/.rally/openrc was updated
...

$ rally deployment check
keystone endpoints are valid and following services are available:
+----------+----------------+-----------+
| services | type           | status    |
+----------+----------------+-----------+
| cinder   | volume         | Available |
| cinderv2 | volumev2       | Available |
| ec2      | ec2            | Available |
| glance   | image          | Available |
| heat     | orchestration  | Available |
| heat-cfn | cloudformation | Available |
| keystone | identity       | Available |
| nova     | compute        | Available |
| novav21  | computev21     | Available |
| s3       | s3             | Available |
+----------+----------------+-----------+
```

Note the first two lines of the CLI output for the *rally deployment use* command. They tell you the UUID of the new active deployment and also say that the *~/.rally/openrc* file was updated – this is the place where the "active" UUID is actually stored by Rally.

One last detail about managing different deployments in Rally is that the *rally task list* command outputs only those tasks that were run against the currently active deployment, and you have to provide the *–all-deployments* parameter to list all the tasks:

```
$ rally task list
+--------------------------------------+-----------------+-----------------------------
↪+---------------+----------+--------+-----+
| uuid                                 | deployment_name | created_at                 ␣
↪| duration       | status   | failed | tag |
```

```
+--------------------------------------+----------------+----------------------------
↪+----------------+----------+--------+-----+
| c21a6ecb-57b2-43d6-bbbb-d7a827f1b420 | cloud-1        | 2015-01-05 01:00:42.099596␣
↪| 0:00:13.419226 | finished | False  |     |
| f6dad6ab-1a6d-450d-8981-f77062c6ef4f | cloud-1        | 2015-01-05 01:05:57.653253␣
↪| 0:00:14.160493 | finished | False  |     |
+--------------------------------------+----------------+----------------------------
↪+----------------+----------+--------+-----+
$ rally task list --all-deployment
+--------------------------------------+----------------+----------------------------
↪+----------------+----------+--------+-----+
| uuid                                 | deployment_name | created_at                ␣
↪| duration       | status   | failed | tag |
+--------------------------------------+----------------+----------------------------
↪+----------------+----------+--------+-----+
| c21a6ecb-57b2-43d6-bbbb-d7a827f1b420 | cloud-1        | 2015-01-05 01:00:42.099596␣
↪| 0:00:13.419226 | finished | False  |     |
| f6dad6ab-1a6d-450d-8981-f77062c6ef4f | cloud-1        | 2015-01-05 01:05:57.653253␣
↪| 0:00:14.160493 | finished | False  |     |
| 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996 | cloud-2        | 2015-01-05 01:14:51.428958␣
↪| 0:00:15.042265 | finished | False  |     |
+--------------------------------------+----------------+----------------------------
↪+----------------+----------+--------+-----+
```

Step 8. Discovering more plugins in Rally

- *Plugins in the Rally repository*
- *CLI: rally plugin show*
- *CLI: rally plugin list*

Plugins in the Rally repository

Rally currently comes with a great collection of plugins that use the API of different OpenStack projects like **Keystone**, **Nova**, **Cinder**, **Glance** and so on. The good news is that you can combine multiple plugins in one task to test your cloud in a comprehensive way.

First, let's see what plugins are available in Rally. One of the ways to discover these plugins is just to inspect their source code. another is to use build-in rally plugin command.

CLI: rally plugin show

Rally plugin CLI command is much more convenient way to learn about different plugins in Rally. This command allows to list plugins and show detailed information about them:

```
$ rally plugin show create_meter_and_get_stats

NAME
    CeilometerStats.create_meter_and_get_stats
NAMESPACE
    default
```

```
MODULE
    rally.plugins.openstack.scenarios.ceilometer.stats
DESCRIPTION
    Meter is first created and then statistics is fetched for the same
    using GET /v2/meters/(meter_name)/statistics.
PARAMETERS
+--------+--------------------------------------------------+
| name   | description                                      |
+--------+--------------------------------------------------+
| kwargs | contains optional arguments to create a meter    |
|        |                                                  |
+--------+--------------------------------------------------+
```

In case if multiple found benchmarks found command list all matches elements:

```
$ rally plugin show NovaKeypair

Multiple plugins found:
+-------------------------------------------------+-----------+-----------------------
↪---------------------------------+
| name                                            | namespace | title                 ␣
↪                                 |
+-------------------------------------------------+-----------+-----------------------
↪---------------------------------+
| NovaKeypair.boot_and_delete_server_with_keypair | default   | Boot and delete␣
↪server with keypair.                  |
| NovaKeypair.create_and_delete_keypair           | default   | Create a keypair with␣
↪random name and delete keypair. |
| NovaKeypair.create_and_list_keypairs            | default   | Create a keypair with␣
↪random name and list keypairs.  |
+-------------------------------------------------+-----------+-----------------------
↪---------------------------------+
```

CLI: rally plugin list

This command can be used to list filtered by name list of plugins.

```
$ rally plugin list --name Keystone

+-------------------------------------------------+-----------+-----------------------
↪----------------------------------------+
| name                                            | namespace | title                 ␣
↪                                      |
+-------------------------------------------------+-----------+-----------------------
↪----------------------------------------+
| Authenticate.keystone                           | default   | Check Keystone␣
↪Client.                                        |
| KeystoneBasic.add_and_remove_user_role          | default   | Create a user role␣
↪add to a user and disassociate.              |
| KeystoneBasic.create_add_and_list_user_roles    | default   | Create user role,␣
↪add it and list user roles for given user.    |
| KeystoneBasic.create_and_delete_ec2credential   | default   | Create and delete␣
↪keystone ec2-credential.                     |
| KeystoneBasic.create_and_delete_role            | default   | Create a user role␣
↪and delete it.                               |
| KeystoneBasic.create_and_delete_service         | default   | Create and delete␣
↪service.                                     |
```

```
| KeystoneBasic.create_and_list_ec2credentials     | default   | Create and List all␣
↪keystone ec2-credentials.                    |
| KeystoneBasic.create_and_list_services           | default   | Create and list␣
↪services.                                     |
| KeystoneBasic.create_and_list_tenants            | default   | Create a keystone␣
↪tenant with random name and list all tenants. |
| KeystoneBasic.create_and_list_users              | default   | Create a keystone␣
↪user with random name and list all users.     |
| KeystoneBasic.create_delete_user                 | default   | Create a keystone␣
↪user with random name and then delete it.     |
| KeystoneBasic.create_tenant                      | default   | Create a keystone␣
↪tenant with random name.                      |
| KeystoneBasic.create_tenant_with_users           | default   | Create a keystone␣
↪tenant and several users belonging to it.     |
| KeystoneBasic.create_update_and_delete_tenant    | default   | Create, update and␣
↪delete tenant.                               |
| KeystoneBasic.create_user                        | default   | Create a keystone␣
↪user with random name.                        |
| KeystoneBasic.create_user_set_enabled_and_delete | default   | Create a keystone␣
↪user, enable or disable it, and delete it.    |
| KeystoneBasic.create_user_update_password        | default   | Create user and␣
↪update password for that user.                |
| KeystoneBasic.get_entities                       | default   | Get instance of a␣
↪tenant, user, role and service by id's.       |
+--------------------------------------------------+-----------+----------------------
↪-----------------------------------------+
```

Step 9. Deploying OpenStack from Rally

Along with supporting already existing OpenStack deployments, Rally itself can **deploy OpenStack automatically** by using one of its *deployment engines*. Take a look at other deployment configuration file samples. For example, *devstack-in-existing-servers.json* is a deployment configuration file that tells Rally to deploy OpenStack with **Devstack** on the existing servers with given credentials:

```
{
    "type": "DevstackEngine",
    "provider": {
        "type": "ExistingServers",
        "credentials": [{"user": "root", "host": "10.2.0.8"}]
    }
}
```

You can try to deploy OpenStack in your Virtual Machine using this script. Edit the configuration file with your IP address/user name and run, as usual:

```
$ rally deployment create --file=samples/deployments/for_deploying_openstack_with_
↪rally/devstack-in-existing-servers.json --name=new-devstack
+--------------------------+--------------------------+--------------+-------------
↪-----+
|            uuid          |         created_at       |     name     |      status␣
↪     |
+--------------------------+--------------------------+--------------+-------------
↪-----+
|     <Deployment UUID>    | 2015-01-10 22:00:28.270941 | new-devstack | deploy->
↪finished |
+--------------------------+--------------------------+--------------+-------------
↪-----+
```

```
Using deployment : <Deployment UUID>
```

Step 10. Verifying cloud via Tempest verifier

- *Create/delete Tempest verifier*
- *Configure Tempest verifier*
- *Start a verification*

As you may know, Rally has a verification component (aka **'rally verify'**). Earlier the purpose of this component was to simplify work with Tempest framework (The OpenStack Integration Test Suite). Rally provided a quite simple interface to install and configure Tempest, run tests and build a report with results. But now the verification component allows us to simplify work not only with Tempest but also with any test frameworks or tools. All you need is to create a plugin for your framework or tool, and you will be able to use **'rally verify'** interface for it. At this point, Rally supports only one plugin in the verification component out of the box - as you might guess, Tempest plugin. In this guide, we will show how to use Tempest and Rally together via the updated **'rally verify'** interface. We assume that you already have a *Rally installation* and have already *registered an OpenStack deployment* in Rally. So, let's get started!

Create/delete Tempest verifier

Execute the following command to create a Tempest verifier:

```
$ rally verify create-verifier --type tempest --name tempest-verifier
2017-01-18 14:43:20.807 5125 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 14:43:21.203 5125 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from https://git.openstack.org/openstack/tempest.
2017-01-18 14:43:32.458 5125 INFO rally.verification.manager [-] Creating virtual␣
↪environment. It may take a few minutes.
2017-01-18 14:43:49.786 5125 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=cde1b03d-d1eb-47f2-a997-3fd21b1d8810) has been successfully created!
Using verifier 'tempest-verifier' (UUID=cde1b03d-d1eb-47f2-a997-3fd21b1d8810) as the␣
↪default verifier for the future operations.
```

The command clones Tempest from the **https://git.openstack.org/openstack/tempest** repository and installs it in a Python virtual environment for the current deployment by default. All information about the created verifier is stored in a database. It allows us to set up different Tempest versions and easily switch between them. How to do it will be described bellow. You can list all installed verifiers via the **rally verify list-verifiers** command.

The arguments below allow us to override the default behavior.

Use the **–source** argument to specify an alternate git repository location. The path to a local Tempest repository or a URL of a remote repository are both valid values.

```
$ rally verify create-verifier --type tempest --name tempest-verifier --source /home/
↪ubuntu/tempest/
2017-01-18 14:53:19.958 5760 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 14:53:20.166 5760 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from /home/ubuntu/tempest/.
2017-01-18 14:53:20.299 5760 INFO rally.verification.manager [-] Creating virtual␣
↪environment. It may take a few minutes.
```

```
2017-01-18 14:53:32.517 5760 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=3f878030-1edf-455c-ae5e-07836e3d7e35) has been successfully created!
Using verifier 'tempest-verifier' (UUID=3f878030-1edf-455c-ae5e-07836e3d7e35) as the␣
↪default verifier for the future operations.
```

```
$ rally verify create-verifier --type tempest --name tempest-verifier --source https:/
↪/github.com/openstack/tempest.git
2017-01-18 14:54:57.786 5907 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 14:54:57.990 5907 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from https://github.com/openstack/tempest.git.
2017-01-18 14:55:05.729 5907 INFO rally.verification.manager [-] Creating virtual␣
↪environment. It may take a few minutes.
2017-01-18 14:55:22.943 5907 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=e84a947c-b9d3-434b-853b-176a597902e5) has been successfully created!
Using verifier 'tempest-verifier' (UUID=e84a947c-b9d3-434b-853b-176a597902e5) as the␣
↪default verifier for the future operations.
```

Use the **–version** argument to specify a Tempest commit ID or tag.

```
$ rally verify create-verifier --type tempest --name tempest-verifier --version␣
↪198e5b4b871c3d09c20afb56dca9637a8cf86ac8
2017-01-18 14:57:02.274 6068 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 14:57:02.461 6068 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from https://git.openstack.org/openstack/tempest.
2017-01-18 14:57:15.356 6068 INFO rally.verification.manager [-] Switching verifier␣
↪repo to the '198e5b4b871c3d09c20afb56dca9637a8cf86ac8' version.
2017-01-18 14:57:15.423 6068 INFO rally.verification.manager [-] Creating virtual␣
↪environment. It may take a few minutes.
2017-01-18 14:57:28.004 6068 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=532d7ad2-902e-4764-aa53-335f67dadc7f) has been successfully created!
Using verifier 'tempest-verifier' (UUID=532d7ad2-902e-4764-aa53-335f67dadc7f) as the␣
↪default verifier for the future operations.
```

```
$ rally verify create-verifier --type tempest --name tempest-verifier --source /home/
↪ubuntu/tempest/ --version 13.0.0
2017-01-18 15:01:53.971 6518 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 15:01:54.180 6518 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from /home/ubuntu/tempest/.
2017-01-18 15:01:54.274 6518 INFO rally.verification.manager [-] Switching verifier␣
↪repo to the '13.0.0' version.
2017-01-18 15:01:54.336 6518 INFO rally.verification.manager [-] Creating virtual␣
↪environment. It may take a few minutes.
2017-01-18 15:02:06.623 6518 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=96ffc4bc-4ac2-4ae9-b3c2-d6b16b871027) has been successfully created!
Using verifier 'tempest-verifier' (UUID=96ffc4bc-4ac2-4ae9-b3c2-d6b16b871027) as the␣
↪default verifier for the future operations.
```

Use the **–system-wide** argument to perform system-wide Tempest installation. In this case, the virtual environment will not be created and Tempest requirements will not be installed. Moreover, it is assumed that requirements are already present in the local environment. This argument is useful when users don't have an Internet connection to install requirements, but they have pre-installed ones in the local environment.

```
$ rally verify create-verifier --type tempest --name tempest-verifier --source /home/
↪ubuntu/tempest/ --version 13.0.0 --system-wide
2017-01-18 15:22:09.198 7224 INFO rally.api [-] Creating verifier 'tempest-verifier'.
2017-01-18 15:22:09.408 7224 INFO rally.verification.manager [-] Cloning verifier␣
↪repo from /home/ubuntu/tempest/.
```

```
2017-01-18 15:22:09.494 7224 INFO rally.verification.manager [-] Switching verifier␣
↪repo to the '13.0.0' version.
2017-01-18 15:22:10.965 7224 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=14c94c12-633a-4522-bd3d-2508f2b9d681) has been successfully created!
Using verifier 'tempest-verifier' (UUID=14c94c12-633a-4522-bd3d-2508f2b9d681) as the␣
↪default verifier for the future operations.
```

To delete the Tempest verifier for all deployments execute the following command:

```
$ rally verify delete-verifier --id 14c94c12-633a-4522-bd3d-2508f2b9d681
2017-01-18 15:27:03.485 7474 INFO rally.api [-] Deleting verifier 'tempest-verifier'␣
↪(UUID=14c94c12-633a-4522-bd3d-2508f2b9d681).
2017-01-18 15:27:03.607 7474 INFO rally.api [-] Verifier has been successfully␣
↪deleted!
```

If you have any verifications, use the **–force** argument to delete the verifier and all stored verifications.

```
$ rally verify delete-verifier --id ec58af86-5217-4bbd-b9e5-491df6873b82
Failed to delete verifier 'tempest-verifier' (UUID=ec58af86-5217-4bbd-b9e5-
↪491df6873b82) because there are stored verifier verifications! Please, make sure␣
↪that they are not important to you. Use 'force' flag if you would like to delete␣
↪verifications as well.
```

```
$ rally verify delete-verifier --id ec58af86-5217-4bbd-b9e5-491df6873b82 --force
2017-01-18 15:49:12.840 8685 INFO rally.api [-] Deleting all verifications created by␣
↪verifier 'tempest-verifier' (UUID=ec58af86-5217-4bbd-b9e5-491df6873b82).
2017-01-18 15:49:12.843 8685 INFO rally.api [-] Deleting verification (UUID=c3d1408a-
↪a224-4d31-b38f-4caf8ce06a95).
2017-01-18 15:49:12.951 8685 INFO rally.api [-] Verification has been successfully␣
↪deleted!
2017-01-18 15:49:12.961 8685 INFO rally.api [-] Deleting verification (UUID=a437537e-
↪538b-4637-b6ab-ecb8072f0c71).
2017-01-18 15:49:13.052 8685 INFO rally.api [-] Verification has been successfully␣
↪deleted!
2017-01-18 15:49:13.061 8685 INFO rally.api [-] Deleting verification (UUID=5cec0579-
↪4b4e-46f3-aeb4-a481a7bc5663).
2017-01-18 15:49:13.152 8685 INFO rally.api [-] Verification has been successfully␣
↪deleted!
2017-01-18 15:49:13.152 8685 INFO rally.api [-] Deleting verifier 'tempest-verifier'␣
↪(UUID=ec58af86-5217-4bbd-b9e5-491df6873b82).
2017-01-18 15:49:13.270 8685 INFO rally.api [-] Verifier has been successfully␣
↪deleted!
```

Use the **–deployment-id** argument to remove the only deployment-specific data, for example, the config file, etc.

```
$ rally verify delete-verifier --deployment-id 351fdfa2-99ad-4447-ba31-22e76630df97
2017-01-18 15:30:27.793 7659 INFO rally.api [-] Deleting deployment-specific data for␣
↪verifier 'tempest-verifier' (UUID=ec58af86-5217-4bbd-b9e5-491df6873b82).
2017-01-18 15:30:27.797 7659 INFO rally.api [-] Deployment-specific data has been␣
↪successfully deleted!
```

When the **–deployment-id** and **–force** arguments are used together, the only deployment-specific data and only verifications of the specified deployment will be deleted.

```
$ rally verify delete-verifier --deployment-id 351fdfa2-99ad-4447-ba31-22e76630df97 --
↪force
2017-01-18 15:55:02.657 9004 INFO rally.api [-] Deleting all verifications created by␣
↪verifier 'tempest-verifier' (UUID=fbbd2bc0-dd92-4e1d-805c-672af7c5ec78) for␣
↪deployment '351fdfa2-99ad-4447-ba31-22e76630df97'.
```

```
2017-01-18 15:55:02.661 9004 INFO rally.api [-] Deleting verification (UUID=a3d3d53c-
↪79a6-4151-85ce-f4a7323d2f4c).
2017-01-18 15:55:02.767 9004 INFO rally.api [-] Verification has been successfully␣
↪deleted!
2017-01-18 15:55:02.776 9004 INFO rally.api [-] Deleting verification (UUID=eddea799-
↪bbc5-485c-a284-1747a30e3f1e).
2017-01-18 15:55:02.869 9004 INFO rally.api [-] Verification has been successfully␣
↪deleted!
2017-01-18 15:55:02.870 9004 INFO rally.api [-] Deleting deployment-specific data for␣
↪verifier 'tempest-verifier' (UUID=fbbd2bc0-dd92-4e1d-805c-672af7c5ec78).
2017-01-18 15:55:02.878 9004 INFO rally.api [-] Deployment-specific data has been␣
↪successfully deleted!
```

Configure Tempest verifier

Execute the following command to configure the Tempest verifier for the current deployment:

```
$ rally verify configure-verifier
2017-01-18 16:00:24.495 9377 INFO rally.api [-] Configuring verifier 'tempest-verifier
↪' (UUID=59e8bd5b-55e1-4ab8-b506-a5853c7a92e9) for deployment 'tempest'␣
↪(UUID=4a62f373-9ce7-47a3-8165-6dc7353f754a).
2017-01-18 16:00:27.497 9377 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=59e8bd5b-55e1-4ab8-b506-a5853c7a92e9) has been successfully configured for␣
↪deployment 'tempest' (UUID=4a62f373-9ce7-47a3-8165-6dc7353f754a)!
```

Use the **–deployment-id** argument to configure the verifier for any deployment registered in Rally.

```
$ rally verify configure-verifier --deployment-id <UUID or name of a deployment>
```

If you want to reconfigure the Tempest verifier, just add the **–reconfigure** argument to the command.

```
$ rally verify configure-verifier --reconfigure
2017-01-18 16:08:50.932 9786 INFO rally.api [-] Configuring verifier 'tempest-verifier
↪' (UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97).
2017-01-18 16:08:50.933 9786 INFO rally.api [-] Verifier is already configured!
2017-01-18 16:08:50.933 9786 INFO rally.api [-] Reconfiguring verifier.
2017-01-18 16:08:52.806 9786 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) has been successfully configured for␣
↪deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!
```

Moreover, it is possible to extend the default verifier configuration by providing the **–extend** argument.

```
$ cat extra_options.conf
[some-section-1]
some-option = some-value

[some-section-2]
some-option = some-value
```

```
$ rally verify configure-verifier --extend extra_options.conf
2017-01-18 16:15:12.248 10029 INFO rally.api [-] Configuring verifier 'tempest-
↪verifier' (UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97).
2017-01-18 16:15:12.249 10029 INFO rally.api [-] Verifier is already configured!
```

```
2017-01-18 16:15:12.249 10029 INFO rally.api [-] Adding extra options to verifier␣
↪configuration.
2017-01-18 16:15:12.439 10029 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) has been successfully configured for␣
↪deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!
```

```
$ rally verify configure-verifier --extend '{section-1: {option: value}, section-2:
↪{option: value}}'
2017-01-18 16:18:07.317 10180 INFO rally.api [-] Configuring verifier 'tempest-
↪verifier' (UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97).
2017-01-18 16:18:07.317 10180 INFO rally.api [-] Verifier is already configured!
2017-01-18 16:18:07.317 10180 INFO rally.api [-] Adding extra options to verifier␣
↪configuration.
2017-01-18 16:18:07.549 10180 INFO rally.api [-] Verifier 'tempest-verifier'␣
↪(UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) has been successfully configured for␣
↪deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!
```

In order to see the generated Tempest config file use the **–show** argument.

```
$ rally verify configure-verifier --show
2017-01-18 16:19:25.412 10227 INFO rally.api [-] Configuring verifier 'tempest-
↪verifier' (UUID=16b73e48-09ad-4a54-92eb-2f2708b72c54) for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97).
2017-01-18 16:19:25.412 10227 INFO rally.api [-] Verifier is already configured!

[DEFAULT]
debug = True
log_file = tempest.log
use_stderr = False

[auth]
use_dynamic_credentials = True
admin_username = admin
admin_password = admin
admin_project_name = admin
admin_domain_name = Default
...
```

Start a verification

In order to start a verification execute the following command:

```
$ rally verify start
2017-01-18 16:49:35.367 12162 INFO rally.api [-] Starting verification (UUID=0673ca09-
↪bdb6-4814-a33e-17731559ff33) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 16:49:44.404 12162 INFO tempest-verifier [-] {0} tempest.api.baremetal.
↪admin.test_chassis.TestChassis ... skip: TestChassis skipped as Ironic is not␣
↪available
2017-01-18 16:49:44.404 12162 INFO tempest-verifier [-] {0} tempest.api.baremetal.
↪admin.test_drivers.TestDrivers ... skip: TestDrivers skipped as Ironic is not␣
↪available
2017-01-18 16:49:44.429 12162 INFO tempest-verifier [-] {3} tempest.api.baremetal.
↪admin.test_ports_negative.TestPortsNegative ... skip: TestPortsNegative skipped as␣
↪Ironic is not available
```

```
2017-01-18 16:49:44.438 12162 INFO tempest-verifier [-] {2} tempest.api.baremetal.
↪admin.test_nodestates.TestNodeStates ... skip: TestNodeStates skipped as Ironic is␣
↪not available
2017-01-18 16:49:44.438 12162 INFO tempest-verifier [-] {2} tempest.api.baremetal.
↪admin.test_ports.TestPorts ... skip: TestPorts skipped as Ironic is not available
2017-01-18 16:49:44.439 12162 INFO tempest-verifier [-] {1} tempest.api.baremetal.
↪admin.test_api_discovery.TestApiDiscovery ... skip: TestApiDiscovery skipped as␣
↪Ironic is not available
2017-01-18 16:49:44.439 12162 INFO tempest-verifier [-] {1} tempest.api.baremetal.
↪admin.test_nodes.TestNodes ... skip: TestNodes skipped as Ironic is not available
2017-01-18 16:49:47.083 12162 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_availability_zone_negative.AZAdminNegativeTestJSON.test_get_availability_zone_
↪list_detail_with_non_admin_user ... success [1.013s]
2017-01-18 16:49:47.098 12162 INFO tempest-verifier [-] {1} tempest.api.compute.admin.
↪test_availability_zone.AZAdminV2TestJSON.test_get_availability_zone_list ...␣
↪success [1.063s]
2017-01-18 16:49:47.321 12162 INFO tempest-verifier [-] {1} tempest.api.compute.admin.
↪test_availability_zone.AZAdminV2TestJSON.test_get_availability_zone_list_detail ...␣
↪success [0.224s]
...
```

By default, the command runs the full suite of Tempest tests for the current deployment. Also, it is possible to run tests of any created verifier, and for any registered deployment in Rally, using the **–id** and **–deployment-id** arguments.

```
$ rally verify start --id <UUID or name of a verifier> --deployment-id <UUID or name␣
↪of a deployment>
```

Also, there is a possibility to run a certain suite of Tempest tests, using the **–pattern** argument.

```
$ rally verify start --pattern set=compute
2017-01-18 16:58:40.378 12631 INFO rally.api [-] Starting verification (UUID=a4bd3993-
↪ba3d-425c-ab81-38b2f627e682) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 16:58:44.883 12631 INFO tempest-verifier [-] {1} tempest.api.compute.admin.
↪test_auto_allocate_network.AutoAllocateNetworkTest ... skip: The microversion␣
↪range[2.37 - latest] of this test is out of the configuration range[None - None].
2017-01-18 16:58:47.330 12631 INFO tempest-verifier [-] {1} tempest.api.compute.admin.
↪test_availability_zone.AZAdminV2TestJSON.test_get_availability_zone_list ...␣
↪success [0.680s]
2017-01-18 16:58:47.416 12631 INFO tempest-verifier [-] {2} tempest.api.compute.admin.
↪test_availability_zone_negative.AZAdminNegativeTestJSON.test_get_availability_zone_
↪list_detail_with_non_admin_user ... success [0.761s]
2017-01-18 16:58:47.610 12631 INFO tempest-verifier [-] {1} tempest.api.compute.admin.
↪test_availability_zone.AZAdminV2TestJSON.test_get_availability_zone_list_detail ...␣
↪success [0.280s]
2017-01-18 16:58:47.694 12631 INFO tempest-verifier [-] {3} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_using_string_ram ... success␣
↪[1.015s]
2017-01-18 16:58:48.514 12631 INFO tempest-verifier [-] {3} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_verify_entry_in_list_details ..
↪. success [0.820s]
2017-01-18 16:58:48.675 12631 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_agents.AgentsAdminTestJSON.test_create_agent ... success [0.777s]
2017-01-18 16:58:49.090 12631 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_agents.AgentsAdminTestJSON.test_delete_agent ... success [0.415s]
2017-01-18 16:58:49.160 12631 INFO tempest-verifier [-] {3} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_int_id ... success [0.
↪646s]
```

```
2017-01-18 16:58:49.546 12631 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_agents.AgentsAdminTestJSON.test_list_agents ... success [0.455s]
...
```

Available suites for Tempest 14.0.0 (the latest Tempest release when this documentation was written) are **full**, **smoke**, **compute**, **identity**, **image**, **network**, **object_storage**, **orchestration**, **volume**, **scenario**. The number of available suites depends on Tempest version because some test sets move from the Tempest tree to the corresponding Tempest plugins.

Moreover, users can run a certain set of tests, using a regular expression.

```
$ rally verify start --pattern tempest.api.compute.admin.test_flavors.
↪FlavorsAdminTestJSON
2017-01-18 17:00:36.590 12745 INFO rally.api [-] Starting verification (UUID=1e12510e-
↪7391-48ed-aba2-8fefe1075a87) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:00:44.241 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_using_string_ram ... success␣
↪[1.044s]
2017-01-18 17:00:45.108 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_verify_entry_in_list_details ..
↪. success [0.868s]
2017-01-18 17:00:45.863 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_int_id ... success [0.
↪754s]
2017-01-18 17:00:47.575 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_none_id ... success [1.
↪712s]
2017-01-18 17:00:48.260 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_uuid_id ... success [0.
↪684s]
2017-01-18 17:00:50.951 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_list_flavor_without_extra_data ...␣
↪success [2.689s]
2017-01-18 17:00:51.631 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_server_with_non_public_flavor ...␣
↪success [0.680s]
2017-01-18 17:00:54.192 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_is_public_string_variations ... success [2.
↪558s]
2017-01-18 17:00:55.102 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_list_non_public_flavor ... success [0.911s]
2017-01-18 17:00:55.774 12745 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_list_public_flavor_with_other_user ...␣
↪success [0.673s]
2017-01-18 17:00:59.602 12745 INFO rally.api [-] Verification (UUID=1e12510e-7391-
↪48ed-aba2-8fefe1075a87) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 10 tests in 14.578 sec.
 - Success: 10
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
```

```
 - Failures: 0

Using verification (UUID=1e12510e-7391-48ed-aba2-8fefe1075a87) as the default␣
↪verification for the future operations.
```

In such a way it is possible to run tests from a certain directory or class, and even run a single test.

```
$ rally verify start --pattern tempest.api.compute.admin.test_flavors.
↪FlavorsAdminTestJSON.test_create_flavor_using_string_ram
2017-01-18 17:01:43.993 12819 INFO rally.api [-] Starting verification (UUID=b9a386e1-
↪d1a1-41b3-b369-9607173de63e) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:01:52.592 12819 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_using_string_ram ... success␣
↪[1.214s]
2017-01-18 17:01:57.220 12819 INFO rally.api [-] Verification (UUID=b9a386e1-d1a1-
↪41b3-b369-9607173de63e) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 1 tests in 4.139 sec.
 - Success: 1
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 0

Using verification (UUID=b9a386e1-d1a1-41b3-b369-9607173de63e) as the default␣
↪verification for the future operations.
```

In order to see errors of failed tests after the verification finished use the **–detailed** argument.

```
$ rally verify start --pattern tempest.api.compute.admin.test_aggregates.
↪AggregatesAdminTestJSON --detailed
2017-01-25 19:34:41.113 16123 INFO rally.api [-] Starting verification (UUID=ceb6f26b-
↪5830-42c5-ab09-bfd985ed4cb7) for deployment 'tempest-2' (UUID=38a397d0-ee11-475d-
↪ab08-e17be09d0bcd) by verifier 'tempest-verifier' (UUID=bbf51ada-9dd6-4b25-b1b6-
↪b651e0541dde).
2017-01-25 19:34:50.188 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_
↪az ... fail [0.784s]
2017-01-25 19:34:51.587 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_get_details ...␣
↪success [1.401s]
2017-01-25 19:34:52.947 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_list ... success [1.
↪359s]
2017-01-25 19:34:53.863 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_remove_host ... success␣
↪[0.915s]
2017-01-25 19:34:54.577 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete ... success [0.
↪714s]
2017-01-25 19:34:55.221 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete_with_az ...␣
↪success [0.643s]
```

```
2017-01-25 19:34:55.974 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_metadata_get_
 ↪details ... success [0.752s]
2017-01-25 19:34:56.689 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_with_az ...␣
 ↪success [0.714s]
2017-01-25 19:34:57.144 16123 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_verify_entry_in_list .
 ↪.. success [0.456s]
2017-01-25 19:35:01.132 16123 INFO rally.api [-] Verification (UUID=ceb6f26b-5830-
 ↪42c5-ab09-bfd985ed4cb7) has been successfully finished for deployment 'tempest-2'␣
 ↪(UUID=38a397d0-ee11-475d-ab08-e17be09d0bcd)!

==============================
Failed 1 test - output below:
==============================

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_
 ↪host_create_server_with_az
--------------------------------------------------------------------------------------
 ↪--------------------------
Traceback (most recent call last):
  File "tempest/api/compute/admin/test_aggregates.py", line 226, in test_aggregate_
 ↪add_host_create_server_with_az
    self.client.add_host(aggregate['id'], host=self.host)
  File "tempest/lib/services/compute/aggregates_client.py", line 95, in add_host
    post_body)
  File "tempest/lib/common/rest_client.py", line 275, in post
    return self.request('POST', url, extra_headers, headers, body, chunked)
  File "tempest/lib/services/compute/base_compute_client.py", line 48, in request
    method, url, extra_headers, headers, body, chunked)
  File "tempest/lib/common/rest_client.py", line 663, in request
    self._error_checker(resp, resp_body)
  File "tempest/lib/common/rest_client.py", line 775, in _error_checker
    raise exceptions.Conflict(resp_body, resp=resp)
tempest.lib.exceptions.Conflict: An object with that identifier already exists
Details: {u'message': u"Cannot add host to aggregate 2658. Reason: One or more hosts␣
 ↪already in availability zone(s) [u'tempest-test_az-34611847'].", u'code': 409}

======
Totals
======

Ran: 9 tests in 12.391 sec.
 - Success: 8
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 1

Using verification (UUID=ceb6f26b-5830-42c5-ab09-bfd985ed4cb7) as the default␣
 ↪verification for the future operations.
```

Also, there is a possibility to run Tempest tests from a file. Users can specify a list of tests in the file and run them, using the **–load-list** argument.

```
$ cat load-list.txt
tempest.api.identity.admin.v2.test_endpoints.EndPointsTestJSON.test_create_list_
 ↪delete_endpoint[id-9974530a-aa28-4362-8403-f06db02b26c1]
```

```
tempest.api.identity.admin.v2.test_endpoints.EndPointsTestJSON.test_list_endpoints[id-
↪11f590eb-59d8-4067-8b2b-980c7f387f51]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_assign_user_role[id-
↪0146f675-ffbd-4208-b3a4-60eb628dbc5e]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_get_role_by_id[id-
↪db6870bd-a6ed-43be-a9b1-2f10a5c9994f]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_list_roles[id-75d9593f-
↪50b7-4fcf-bd64-e3fb4a278e23]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_list_user_roles[id-
↪262e1e3e-ed71-4edd-a0e5-d64e83d66d05]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_remove_user_role[id-
↪f0b9292c-d3ba-4082-aa6c-440489beef69]
tempest.api.identity.admin.v2.test_roles.RolesTestJSON.test_role_create_delete[id-
↪c62d909d-6c21-48c0-ae40-0a0760e6db5e]
```

```
$ rally verify start --load-list load-list.txt
2017-01-18 17:04:13.900 12964 INFO rally.api [-] Starting verification (UUID=af766b2f-
↪cada-44db-a0c2-336ab0c17c27) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:04:21.813 12964 INFO tempest-verifier [-] {1} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_create_list_delete_endpoint ...␣
↪success [1.237s]
2017-01-18 17:04:22.115 12964 INFO tempest-verifier [-] {1} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_list_endpoints ... success [0.301s]
2017-01-18 17:04:24.507 12964 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_assign_user_role ... success [3.663s]
2017-01-18 17:04:25.164 12964 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_get_role_by_id ... success [0.657s]
2017-01-18 17:04:25.435 12964 INFO tempest-verifier [-] {2} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_roles ... success [0.271s]
2017-01-18 17:04:27.905 12964 INFO tempest-verifier [-] {2} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_user_roles ... success [2.468s]
2017-01-18 17:04:30.645 12964 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_remove_user_role ... success [2.740s]
2017-01-18 17:04:31.886 12964 INFO tempest-verifier [-] {3} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_role_create_delete ... success [1.239s]
2017-01-18 17:04:38.122 12964 INFO rally.api [-] Verification (UUID=af766b2f-cada-
↪44db-a0c2-336ab0c17c27) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 8 tests in 14.748 sec.
 - Success: 8
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 0

Using verification (UUID=af766b2f-cada-44db-a0c2-336ab0c17c27) as the default␣
↪verification for the future operations.
```

Moreover, it is possible to skip a certain list of Tempest tests, using the **–skip-list** argument.

```
$ cat skip-list.yaml
tempest.api.compute.admin.test_flavors.FlavorsAdminTestJSON.test_create_flavor_using_
 ↪string_ram[id-3b541a2e-2ac2-4b42-8b8d-ba6e22fcd4da]:
tempest.api.compute.admin.test_flavors.FlavorsAdminTestJSON.test_create_flavor_verify_
 ↪entry_in_list_details[id-8261d7b0-be58-43ec-a2e5-300573c3f6c5]: Reason 1
tempest.api.compute.admin.test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_
 ↪int_id[id-8b4330e1-12c4-4554-9390-e6639971f086]:
tempest.api.compute.admin.test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_
 ↪none_id[id-f83fe669-6758-448a-a85e-32d351f36fe0]: Reason 2
tempest.api.compute.admin.test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_
 ↪uuid_id[id-94c9bb4e-2c2a-4f3c-bb1f-5f0daf918e6d]:
```

```
$ rally verify start --pattern tempest.api.compute.admin.test_flavors.
 ↪FlavorsAdminTestJSON --skip-list skip-list.yaml
2017-01-18 17:13:44.475 13424 INFO rally.api [-] Starting verification (UUID=ec94b397-
 ↪b546-4f12-82ba-bb17f052c3d0) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
 ↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
 ↪2f2708b72c54).
2017-01-18 17:13:49.298 13424 INFO tempest-verifier [-] {-} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_int_id ... skip
2017-01-18 17:13:49.298 13424 INFO tempest-verifier [-] {-} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_none_id ... skip: Reason 2
2017-01-18 17:13:49.298 13424 INFO tempest-verifier [-] {-} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_using_string_ram ... skip
2017-01-18 17:13:49.298 13424 INFO tempest-verifier [-] {-} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_with_uuid_id ... skip
2017-01-18 17:13:49.299 13424 INFO tempest-verifier [-] {-} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_flavor_verify_entry_in_list_details ..
 ↪. skip: Reason 1
2017-01-18 17:13:54.035 13424 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_list_flavor_without_extra_data ...␣
 ↪success [1.889s]
2017-01-18 17:13:54.765 13424 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_create_server_with_non_public_flavor ...␣
 ↪success [0.732s]
2017-01-18 17:13:57.478 13424 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_is_public_string_variations ... success [2.
 ↪709s]
2017-01-18 17:13:58.438 13424 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_list_non_public_flavor ... success [0.962s]
2017-01-18 17:13:59.180 13424 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
 ↪test_flavors.FlavorsAdminTestJSON.test_list_public_flavor_with_other_user ...␣
 ↪success [0.742s]
2017-01-18 17:14:03.969 13424 INFO rally.api [-] Verification (UUID=ec94b397-b546-
 ↪4f12-82ba-bb17f052c3d0) has been successfully finished for deployment 'tempest-2'␣
 ↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 10 tests in 9.882 sec.
 - Success: 5
 - Skipped: 5
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 0

Using verification (UUID=ec94b397-b546-4f12-82ba-bb17f052c3d0) as the default␣
 ↪verification for the future operations.
```

Also, it is possible to specify the path to a file with a list of Tempest tests that are expected to fail. In this case, the specified tests will have the **xfail** status instead of **fail**.

```
$ cat xfail-list.yaml
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_
↪host_create_server_with_az[id-96be03c7-570d-409c-90f8-e4db3c646996]: Some reason␣
↪why the test fails
```

```
$ rally verify start --pattern tempest.api.compute.admin.test_aggregates.
↪AggregatesAdminTestJSON --xfail-list xfail-list.yaml
2017-01-18 17:20:04.064 13720 INFO rally.api [-] Starting verification (UUID=c416b724-
↪0276-4c24-ab60-3ba7078c0a80) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:20:17.359 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_
↪az ... xfail [6.328s]: Some reason why the test fails
2017-01-18 17:20:18.337 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_get_details ...␣
↪success [0.978s]
2017-01-18 17:20:19.379 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_list ... success [1.
↪042s]
2017-01-18 17:20:20.213 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_remove_host ... success␣
↪[0.833s]
2017-01-18 17:20:20.956 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete ... success [0.
↪743s]
2017-01-18 17:20:21.772 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete_with_az ...␣
↪success [0.815s]
2017-01-18 17:20:22.737 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_metadata_get_
↪details ... success [0.964s]
2017-01-18 17:20:25.061 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_with_az ...␣
↪success [2.323s]
2017-01-18 17:20:25.595 13720 INFO tempest-verifier [-] {0} tempest.api.compute.admin.
↪test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_verify_entry_in_list .
↪.. success [0.533s]
2017-01-18 17:20:30.142 13720 INFO rally.api [-] Verification (UUID=c416b724-0276-
↪4c24-ab60-3ba7078c0a80) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 9 tests in 17.118 sec.
 - Success: 8
 - Skipped: 0
 - Expected failures: 1
 - Unexpected success: 0
 - Failures: 0

Using verification (UUID=c416b724-0276-4c24-ab60-3ba7078c0a80) as the default␣
↪verification for the future operations.
```

Sometimes users may want to use the specific concurrency for running tests based on their deployments and available resources. In this case, they can use the **–concurrency** argument to specify how many processes to use to run Tempest tests. The default value (0) auto-detects CPU count.

```
$ rally verify start --load-list load-list.txt --concurrency 1
2017-01-18 17:05:38.658 13054 INFO rally.api [-] Starting verification (UUID=cbf5e604-
↪6bc9-47cd-9c8c-5e4c9e9545a0) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:05:45.474 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_create_list_delete_endpoint ...␣
↪success [0.917s]
2017-01-18 17:05:45.653 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_list_endpoints ... success [0.179s]
2017-01-18 17:05:55.497 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_assign_user_role ... success [2.673s]
2017-01-18 17:05:56.237 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_get_role_by_id ... success [0.740s]
2017-01-18 17:05:56.642 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_roles ... success [0.403s]
2017-01-18 17:06:00.011 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_user_roles ... success [3.371s]
2017-01-18 17:06:02.987 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_remove_user_role ... success [2.973s]
2017-01-18 17:06:04.927 13054 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_role_create_delete ... success [1.939s]
2017-01-18 17:06:11.166 13054 INFO rally.api [-] Verification (UUID=cbf5e604-6bc9-
↪47cd-9c8c-5e4c9e9545a0) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 8 tests in 23.043 sec.
 - Success: 8
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 0

Using verification (UUID=cbf5e604-6bc9-47cd-9c8c-5e4c9e9545a0) as the default␣
↪verification for the future operations.
```

Also, there is a possibility to rerun tests from any verification. In order to rerun tests from some verification execute the following command:

```
$ rally verify rerun --uuid cbf5e604-6bc9-47cd-9c8c-5e4c9e9545a0
2017-01-18 17:29:35.692 14127 INFO rally.api [-] Re-running tests from verification␣
↪(UUID=cbf5e604-6bc9-47cd-9c8c-5e4c9e9545a0) for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97).
2017-01-18 17:29:35.792 14127 INFO rally.api [-] Starting verification (UUID=51aa3275-
↪f028-4f2d-9d63-0db679fdf266) for deployment 'tempest-2' (UUID=351fdfa2-99ad-4447-
↪ba31-22e76630df97) by verifier 'tempest-verifier' (UUID=16b73e48-09ad-4a54-92eb-
↪2f2708b72c54).
2017-01-18 17:29:43.980 14127 INFO tempest-verifier [-] {1} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_create_list_delete_endpoint ...␣
↪success [2.172s]
```

```
2017-01-18 17:29:44.156 14127 INFO tempest-verifier [-] {1} tempest.api.identity.
↪admin.v2.test_endpoints.EndPointsTestJSON.test_list_endpoints ... success [0.177s]
2017-01-18 17:29:45.333 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_assign_user_role ... success [3.302s]
2017-01-18 17:29:45.952 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_get_role_by_id ... success [0.619s]
2017-01-18 17:29:46.219 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_roles ... success [0.266s]
2017-01-18 17:29:48.964 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_list_user_roles ... success [2.744s]
2017-01-18 17:29:52.543 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_remove_user_role ... success [3.578s]
2017-01-18 17:29:53.843 14127 INFO tempest-verifier [-] {0} tempest.api.identity.
↪admin.v2.test_roles.RolesTestJSON.test_role_create_delete ... success [1.300s]
2017-01-18 17:30:01.258 14127 INFO rally.api [-] Verification (UUID=51aa3275-f028-
↪4f2d-9d63-0db679fdf266) has been successfully finished for deployment 'tempest-2'␣
↪(UUID=351fdfa2-99ad-4447-ba31-22e76630df97)!

======
Totals
======
Ran: 8 tests in 14.926 sec.
 - Success: 8
 - Skipped: 0
 - Expected failures: 0
 - Unexpected success: 0
 - Failures: 0

Verification UUID: 51aa3275-f028-4f2d-9d63-0db679fdf266.
```

In order to rerun only failed tests add the **–failed** argument to the command.

```
$ rally verify rerun --uuid <UUID of a verification> --failed
```

A separated page about building verification reports: *Verification reports*.

Rally OpenStack Gates

Gate jobs

The **OpenStack CI system** uses the so-called **"Gate jobs"** to control merges of patches submitted for review on Gerrit. These **Gate jobs** usually just launch a set of tests – unit, functional, integration, style – that check that the proposed patch does not break the software and can be merged into the target branch, thus providing additional guarantees for the stability of the software.

Create a custom Rally Gate job

You can create a **Rally Gate job** for your project to run Rally benchmarks against the patchsets proposed to be merged into your project.

To create a rally-gate job, you should create a **rally-jobs/** directory at the root of your project.

As a rule, this directory contains only **{projectname}.yaml**, but more scenarios and jobs can be added as well. This yaml file is in fact an input Rally task file specifying benchmark scenarios that should be run in your gate job.

To make *{projectname}.yaml* run in gates, you need to add *"rally-jobs"* to the "jobs" section of *projects.yaml* in *openstack-infra/project-config*.

Example: Rally Gate job for Glance

Let's take a look at an example for the Glance project:

Edit *jenkins/jobs/projects.yaml:*

```
- project:
    name: glance
    node: 'bare-precise || bare-trusty'
    tarball-site: tarballs.openstack.org
    doc-publisher-site: docs.openstack.org

    jobs:
      - python-jobs
      - python-icehouse-bitrot-jobs
      - python-juno-bitrot-jobs
      - openstack-publish-jobs
      - translation-jobs
      - rally-jobs
```

Also add *gate-rally-dsvm-{projectname}* to *zuul/layout.yaml*:

```
- name: openstack/glance
  template:
    - name: merge-check
    - name: python26-jobs
    - name: python-jobs
    - name: openstack-server-publish-jobs
    - name: openstack-server-release-jobs
    - name: periodic-icehouse
    - name: periodic-juno
    - name: check-requirements
    - name: integrated-gate
    - name: translation-jobs
    - name: large-ops
    - name: experimental-tripleo-jobs
  check:
    - check-devstack-dsvm-cells
    - gate-rally-dsvm-glance
  gate:
    - gate-devstack-dsvm-cells
  experimental:
    - gate-grenade-dsvm-forward
```

To add one more scenario and job, you need to add *{scenarioname}.yaml* file here, and *gate-rally-dsvm-{scenarioname}* to *projects.yaml*.

For example, you can add *myscenario.yaml* to *rally-jobs* directory in your project and then edit *jenkins/jobs/projects.yaml* in this way:

```
- project:
    name: glance
    github-org: openstack
    node: bare-precise
```

```
    tarball-site: tarballs.openstack.org
    doc-publisher-site: docs.openstack.org

    jobs:
      - python-jobs
      - python-havana-bitrot-jobs
      - openstack-publish-jobs
      - translation-jobs
      - rally-jobs
      - 'gate-rally-dsvm-{name}':
        name: myscenario
```

Finally, add *gate-rally-dsvm-myscenario* to *zuul/layout.yaml*:

```
- name: openstack/glance
  template:
    - name: python-jobs
    - name: openstack-server-publish-jobs
    - name: periodic-havana
    - name: check-requirements
    - name: integrated-gate
  check:
    - check-devstack-dsvm-cells
    - check-tempest-dsvm-postgres-full
    - gate-tempest-dsvm-large-ops
    - gate-tempest-dsvm-neutron-large-ops
    - gate-rally-dsvm-myscenario
```

It is also possible to arrange your input task files as templates based on `Jinja2`. Say, you want to set the image names used throughout the *myscenario.yaml* task file as a variable parameter. Then, replace concrete image names in this file with a variable:

```
...

NovaServers.boot_and_delete_server:
  -
    args:
      image:
          name: {{image_name}}
    ...

NovaServers.boot_and_list_server:
  -
    args:
      image:
          name: {{image_name}}
    ...
```

and create a file named *myscenario_args.yaml* that will define the parameter values:

```
---

  image_name: "^cirros.*-disk$"
```

this file will be automatically used by Rally to substitute the variables in *myscenario.yaml*.

Plugins & Extras in Rally Gate jobs

Along with scenario configs in yaml, the **rally-jobs** directory can also contain two subdirectories:

- **plugins**: *Plugins* needed for your gate job;
- **extra**: auxiliary files like bash scripts or images.

Both subdirectories will be copied to *~/.rally/* before the job gets started.

Command Line Interface

Category: db

Commands for DB management.

rally-manage db create

Create Rally database.

rally-manage db recreate

Drop and create Rally database.

This will delete all existing data.

rally-manage db revision

Print current Rally database revision UUID.

rally-manage db upgrade

Upgrade Rally database to the latest state.

Category: deployment

Set of commands that allow you to manage deployments.

rally deployment check

Check keystone authentication and list all available services.

Command arguments:

- *–deployment* <*uuid*> [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

UUID or name of the deployment.

type: str

rally deployment config

Display configuration of the deployment.

Output is the configuration of the deployment in a pretty-printed JSON format.

Command arguments:

- *–deployment* <*uuid*> [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

UUID or name of the deployment.

type: str

rally deployment create

Create new deployment.

This command will create a new deployment record in rally database. In the case of ExistingCloud deployment engine, it will use the cloud represented in the configuration. If the cloud doesn't exist, Rally can deploy a new one for you with Devstack or Fuel. Different deployment engines exist for these cases.

If you use the ExistingCloud deployment engine, you can pass the deployment config by environment variables with `--fromenv`:

OS_USERNAME OS_PASSWORD OS_AUTH_URL OS_TENANT_NAME or OS_PROJECT_NAME OS_ENDPOINT_TYPE or OS_INTERFACE OS_ENDPOINT OS_REGION_NAME OS_CACERT OS_INSECURE OS_IDENTITY_API_VERSION

All other deployment engines need more complex configuration data, so it should be stored in a configuration file.

You can use physical servers, LXC containers, KVM virtual machines or virtual machines in OpenStack for deploying the cloud. Except physical servers, Rally can create cluster nodes for you. Interaction with virtualization software, OpenStack cloud or physical servers is provided by server providers.

Command arguments:

- *–name <name>* [*ref*]

  Name of the deployment.

  type: str

- *–fromenv* [*ref*]

  Read environment variables instead of config file.

- *–filename <path>* [*ref*]

  Path to the configuration file of the deployment.

  type: str

  default: none

- *–no-use* [*ref*]

  Don't set new deployment as default for future operations.

rally deployment destroy

Destroy existing deployment.

This will delete all containers, virtual machines, OpenStack instances or Fuel clusters created during Rally deployment creation. Also it will remove the deployment record from the Rally database.

Command arguments:

- *–deployment <uuid>* [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

UUID or name of the deployment.

type: str

rally deployment list

List existing deployments.

rally deployment recreate

Destroy and create an existing deployment.

Unlike 'deployment destroy', the deployment database record will not be deleted, so the deployment UUID stays the same.

Command arguments:

- *–filename* <*path*> [*ref*]

  Path to the configuration file of the deployment.

  type: str

  default: none

- *–deployment* <*uuid*> [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  UUID or name of the deployment.

  type: str

rally deployment show

Show the credentials of the deployment.

Command arguments:

- *–deployment* <*uuid*> [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  UUID or name of the deployment.

  type: str

rally deployment use

Set active deployment.

Command arguments:

- *–deployment* <*uuid*> [*ref*]

  UUID or name of a deployment.

  type: str

Category: plugin

Set of commands that allow you to manage Rally plugins.

rally plugin list

List all Rally plugins that match name and namespace.

Command arguments:

- *–name* <*name*> [*ref*]

  List only plugins that match the given name.

  type: str

  default: none

- *–namespace* <*namespace*> [*ref*]

  List only plugins that are in the specified namespace.

  type: str

  default: none

- *–plugin-base* <*plugin_base*> [*ref*]

  Plugin base class.

  type: str

  default: none

rally plugin show

Show detailed information about a Rally plugin.

Command arguments:

- *–name* <*name*> [*ref*]

  Plugin name.

  type: str

- *–namespace* <*namespace*> [*ref*]

  Plugin namespace.

  type: str

default: none

Category: task

Set of commands that allow you to manage benchmarking tasks and results.

rally task abort

Abort a running benchmarking task.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUID of task.

  type: str

- *–soft* [*ref*]

  Abort task after current scenario finishes execution.

rally task delete

Delete task and its results.

Command arguments:

- *–force* [*ref*]

  Force delete

- *–uuid* <*task-id*> [*ref*]

  UUID of task or a list of task UUIDs.

  type: str

rally task detailed

Print detailed information about given task.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUID of task. If –uuid is “last” the results of the most recently created task will be displayed.

  type: str

- *–iterations-data* [*ref*]

  Print detailed results for each iteration.

rally task export

Export task results to the custom task's exporting system.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUID of a the task.

  type: str

- *–connection* <*connection*> [*ref*]

  Connection url to the task export system.

  type: str

rally task list

List tasks, started and finished.

Displayed tasks can be filtered by status or deployment. By default 'rally task list' will display tasks from the active deployment without filtering by status.

Command arguments:

- *–deployment* <*uuid*> [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  UUID or name of a deployment.

  type: str

- *–all-deployments* [*ref*]

  List tasks from all deployments.

- *–status* <*status*> [*ref*]

  List tasks with specified status. Available statuses: aborted, aborting, crashed, finished, init, paused, running, sla_failed, soft_aborting, validated, validating, validation_failed

  type: str

  default: none

- *–uuids-only* [*ref*]

  List task UUIDs only.

rally task report

Generate report file for specified task.

Command arguments:

- *–tasks* <*tasks*> [*ref*]

  UUIDs of tasks, or JSON files with task results

  default: none

- *–out* <*path*> [*ref*]

  Path to output file.

  type: str

  default: none

- *–open* [*ref*]

  Open the output in a browser.

- *–html* [*ref*]

  Generate the report in HTML.

- *–html-static* [*ref*]

  Generate the report in HTML with embedded JS and CSS, so it will not depend on Internet availability.

- *–junit* [*ref*]

  Generate the report in the JUnit format.

rally task results

Display raw task results.

This will produce a lot of output data about every iteration.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUID of task.

  type: str

rally task sla-check

Display SLA check results table.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUID of task.

  type: str

- *–json* [*ref*]

  Output in JSON format.

rally task sla_check

DEPRECATED since Rally 0.8.0, use *rally task sla-check* instead.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUID of task.

  type: str

- *–json* [*ref*]

  Output in JSON format.

rally task start

Start benchmark task.

If both task_args and task_args_file are specified, they will be merged. task_args has a higher priority so it will override values from task_args_file.

Command arguments:

- *–deployment <uuid>* [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  UUID or name of a deployment.

  type: str

- *–task <path>, –filename <path>* [*ref*]

Note: The default value for the `--task` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally task start`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally task use <uuid>` (*ref*).

  Path to the input task file

- *–task-args <json>* [*ref*]

  Input task args (JSON dict). These args are used to render the Jinja2 template in the input task.

  default: none

- *–task-args-file <path>* [*ref*]

Path to the file with input task args (dict in JSON/YAML). These args are used to render the Jinja2 template in the input task.

default: none

- *–tag <tag>* [*ref*]

  Tag for this task

  default: none

- *–no-use* [*ref*]

  Don't set new task as default for future operations.

- *–abort-on-sla-failure* [*ref*]

  Abort the execution of a benchmark scenario whenany SLA check for it fails.

rally task status

Display the current status of a task.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUID of task

  type: str

rally task trends

Generate workloads trends HTML report.

Command arguments:

- *–out <path>* [*ref*]

  Path to output file.

  type: str

- *–open* [*ref*]

  Open the output in a browser.

- *–tasks <tasks>* [*ref*]

  UUIDs of tasks, or JSON files with task results

rally task use

Set active task.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUID of the task

  type: str

- *–task* [*ref*]

[deprecated since rally 0.2.0] use '–uuid' instead.

type: str

rally task validate

Validate a task configuration file.

This will check that task configuration file has valid syntax and all required options of scenarios, contexts, SLA and runners are set.

If both task_args and task_args_file are specified, they will be merged. task_args has a higher priority so it will override values from task_args_file.

Command arguments:

- *–deployment <uuid>* [*ref*]

Note: The default value for the `--deployment` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

UUID or name of a deployment.

type: str

- *–task <path>, –filename <path>* [*ref*]

Note: The default value for the `--task` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally task start`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally task use <uuid>` (*ref*).

Path to the input task file.

- *–task-args <json>* [*ref*]

  Input task args (JSON dict). These args are used to render the Jinja2 template in the input task.

  default: none

- *–task-args-file <path>* [*ref*]

  Path to the file with input task args (dict in JSON/YAML). These args are used to render the Jinja2 template in the input task.

  default: none

Category: verify

Verify an OpenStack cloud via a verifier.

rally verify add-verifier-ext

Add a verifier extension.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–source <source>* [*ref*]

  Path or URL to the repo to clone verifier extension from.

  type: str

  default: none

- *–version <version>* [*ref*]

  Branch, tag or commit ID to checkout before installation of the verifier extension (the 'master' branch is used by default).

  type: str

  default: none

- *–extra-settings <extra_settings>* [*ref*]

  Extra installation settings for verifier extension.

  type: str

  default: none

rally verify configure-verifier

Configure a verifier for a specific deployment.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–reconfigure* [*ref*]

  Reconfigure verifier.

- *–extend <path/json/yaml>* [*ref*]

  Extend verifier configuration with extra options. If options are already present, the given ones will override them. Can be a path to a regular config file or just a json/yaml.

  type: str

  default: none

- *–override <path>* [*ref*]

  Override verifier configuration by another one from a given source.

  type: str

  default: none

- *–show* [*ref*]

  Show verifier configuration.

rally verify create-verifier

Create a verifier.

Command arguments:

- *–name <name>* [*ref*]

  Verifier name (for example, 'My verifier').

  type: str

- *–type <type>* [*ref*]

  Verifier plugin name. HINT: You can list all verifier plugins, executing command *rally verify list-plugins*.

  type: str

- *–namespace <name>* [*ref*]

  Verifier plugin namespace. Should be specified in case of two verifier plugins with equal names but in different namespaces.

  type: str

  default:

- *–source <source>* [*ref*]

  Path or URL to the repo to clone verifier from.

  type: str

default: none

- *–version <version>* [*ref*]

  Branch, tag or commit ID to checkout before verifier installation (the 'master' branch is used by default).

  type: str

  default: none

- *–system-wide* [*ref*]

  Use the system-wide environment for verifier instead of a virtual environment.

- *–extra-settings <extra_settings>* [*ref*]

  Extra installation settings for verifier.

  type: str

  default: none

- *–no-use* [*ref*]

  Not to set the created verifier as the default verifier for future operations.

rally verify delete

Delete a verification or a few verifications.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUIDs of verifications. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

rally verify delete-verifier

Delete a verifier.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. If specified, only the deployment-specific data will be deleted for verifier. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–force* [*ref*]

  Delete all stored verifications of the specified verifier. If a deployment specified, only verifications of this deployment will be deleted. Use this argument carefully! You can delete verifications that may be important to you.

rally verify delete-verifier-ext

Delete a verifier extension.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–name <name>* [*ref*]

  Verifier extension name.

  type: str

  default: none

rally verify import

Import results of a test run into the Rally database.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–file <path>* [*ref*]

  File to import test results from.

  type: str

  default: none

- *–run-args <run_args>* [*ref*]

  Arguments that might be used when running tests. For example, '{concurrency: 2, pattern: set=identity}'.

  type: str

  default: none

- *–no-use* [*ref*]

  Not to set the created verification as the default verification for future operations.

rally verify list

List all verifications.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

  type: str

- *–tag <tag>* [*ref*]

  Tags to filter verifications by.

  type: str

  default: none

- *–status <status>* [*ref*]

  Status to filter verifications by.

type: str

default: none

rally verify list-plugins

List all plugins for verifiers management.

Command arguments:

- *–namespace* <*name*> [*ref*]

  Namespace name (for example, openstack).

  type: str

  default: none

rally verify list-verifier-exts

List all verifier extensions.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

rally verify list-verifier-tests

List all verifier tests.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–pattern* <*pattern*> [*ref*]

  Pattern which will be used for matching. Can be a regexp or a verifier-specific entity (for example, in case of Tempest you can specify 'set=smoke').

  type: str

  default:

rally verify list-verifiers

List all verifiers.

Command arguments:

- *–status* <*status*> [*ref*]

Status to filter verifiers by.

type: str

default: none

rally verify report

Generate a report for a verification or a few verifications.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUIDs of verifications. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–type <type>* [*ref*]

  Report type (Defaults to JSON). Out-of-the-box types: HTML, HTML-Static, JSON, JUnit-XML. HINT: You can list all types, executing *rally plugin list –plugin-base VerificationReporter* command.

  type: str

  default: none

- *–to <dest>* [*ref*]

  Report destination. Can be a path to a file (in case of HTML, JSON, etc. types) to save the report to or a connection string. It depends on the report type.

  type: str

  default: none

- *–open* [*ref*]

  Open the output file in a browser.

rally verify rerun

Rerun tests from a verification for a specific deployment.

Command arguments:

- *–uuid <uuid>* [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–failed* [*ref*]

  Rerun only failed tests.

- *–tag <tag>* [*ref*]

  Mark verification with a tag or a few tags.

  type: str

  default: none

- *–concurrency <N>* [*ref*]

  How many processes to be used for running verifier tests. The default value (0) auto-detects your CPU count.

  type: int

  default: none

- *–detailed* [*ref*]

  Show verification details such as errors of failed tests.

- *–no-use* [*ref*]

  Not to set the finished verification as the default verification for future operations.

rally verify show

Show detailed information about a verification.

Command arguments:

- *–uuid <uuid>* [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–sort-by <query>* [*ref*]

  Sort tests by 'name', 'duration' or 'status'.

  type: str

  default: name

- *–detailed* [*ref*]

  Show verification details such as run arguments and errors of failed tests.

rally verify show-verifier

Show detailed information about a verifier.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

rally verify start

Start a verification (run verifier tests).

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id* <*id*> [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

  type: str

- *–tag* <*tag*> [*ref*]

  Mark verification with a tag or a few tags.

  type: str

  default: none

- *–pattern* <*pattern*> [*ref*]

  Pattern which will be used for running tests. Can be a regexp or a verifier-specific entity (for example, in case of Tempest you can specify 'set=smoke').

  type: str

  default: none

- *–concurrency* <*N*> [*ref*]

How many processes to be used for running verifier tests. The default value (0) auto-detects your CPU count.

type: int

default: 0

- *–load-list* <*path*> [*ref*]

  Path to a file with a list of tests to run.

  type: str

  default: none

- *–skip-list* <*path*> [*ref*]

  Path to a file with a list of tests to skip. Format: json or yaml like a dictionary where keys are test names and values are reasons.

  type: str

  default: none

- *–xfail-list* <*path*> [*ref*]

  Path to a file with a list of tests that will be considered as expected failures. Format: json or yaml like a dictionary where keys are test names and values are reasons.

  type: str

  default: none

- *–detailed* [*ref*]

  Show verification details such as errors of failed tests.

- *–no-use* [*ref*]

  Not to set the finished verification as the default verification for future operations.

rally verify update-verifier

Update a verifier.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–update-venv* [*ref*]

  Update the virtual environment for verifier.

- *–version* <*version*> [*ref*]

  Branch, tag or commit ID to checkout. HINT: Specify the same version to pull the latest repo code.

  type: str

  default: none

- *–system-wide* [*ref*]

Switch to using the system-wide environment.

- *–no-system-wide* [*ref*]

  Switch to using the virtual environment. If the virtual environment doesn't exist, it will be created.

rally verify use

Choose a verification to use for the future operations.

Command arguments:

- *–uuid <uuid>* [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

rally verify use-verifier

Choose a verifier to use for the future operations.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

Task Component

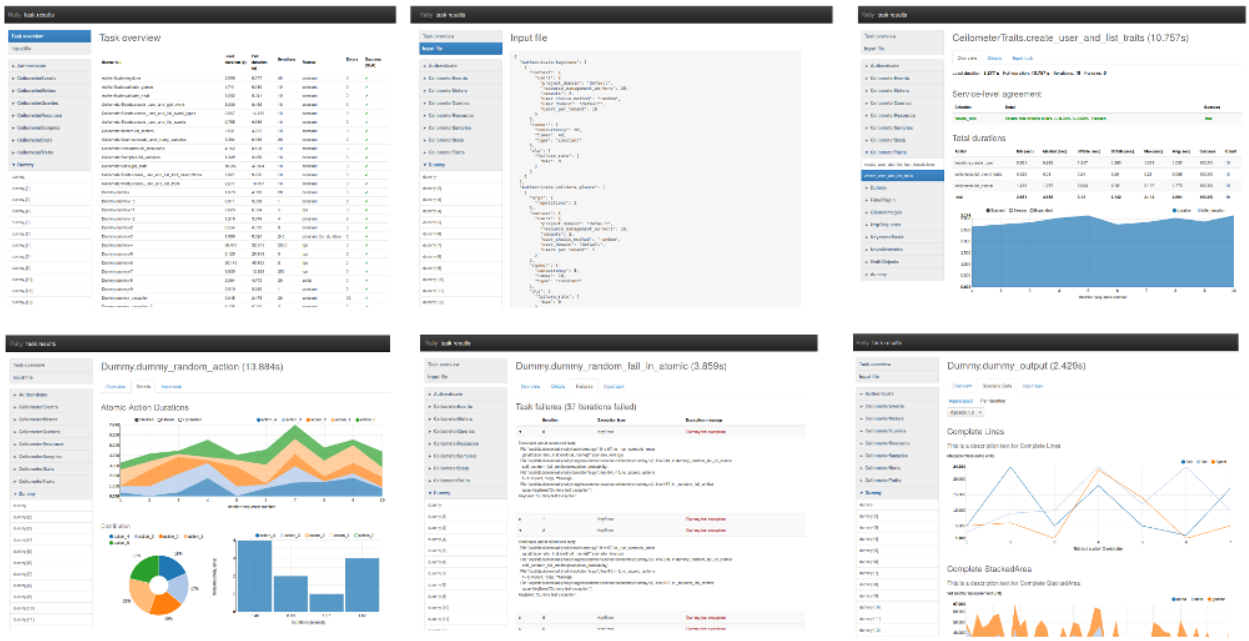
This section describes Rally Task Component (including feature presented since Rally v0.5.0, allowing to analyze statistics trends for the given tasks).

HTML Reports

HTML reports provide comprehensive analysis. Data is structured and displayed interactively, with charts and tables.

Task Report

Get the whole information about task workloads results, in pretty and convenient format!



Generate report for single task, using task UUID

Having a finished task, generate report with command:

```
$ rally task report <task-uuid> --out <report-file>
```

Example:

```
$ rally task report 6f63d9ec-eecd-4696-8e9c-2ba065c68535 --out report.html
```

Generate report for single task, using JSON file

Report can be generated from a task results JSON file. This file can be generated with command *rally task results*:

```
$ rally task results 6f63d9ec-eecd-4696-8e9c-2ba065c68535 > results.json
$ rally task report results.json --out report.html
```

Generate report for many tasks

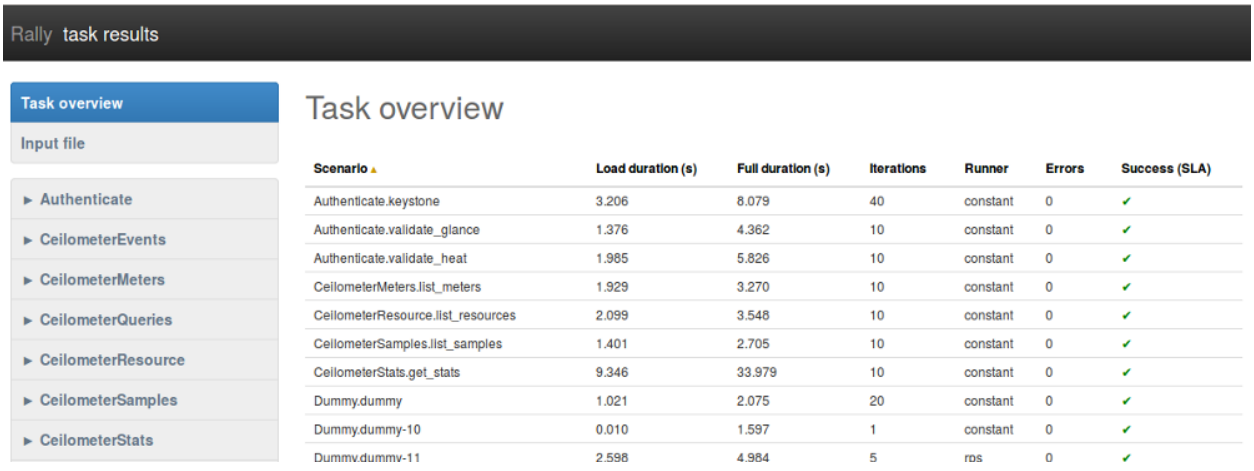
Report can be generated from many tasks. All workloads from specified tasks results will be composed into an entire report. To generate report, use *–tasks* argument with specified list of tasks UUIDs and/or tasks results JSON files.

Example:

```
$ rally task report --tasks 6f63d9ec-eecd-4696-8e9c-2ba065c68535 20ae7e95-7395-4be4-
↪aec2-b89220adee60 a5737eba-a204-43d6-a262-d5ea4b0065da results.json another_results.
↪json --out report.html
```

Task Overview

This is a table with brief summary of all workloads results. All columns are sortable and clickable.



Load duration

Time from first iteration start to last iteration end. In other words, this is a time of all workload iterations execution.

Full duration

This time includes iterations time (*Load duration*) plus time taken by another actions related to the task, mostly Contexts execution time.

Iterations

How many times the workload has run. This comes from the value of *runner.times* in task input file.

Failures

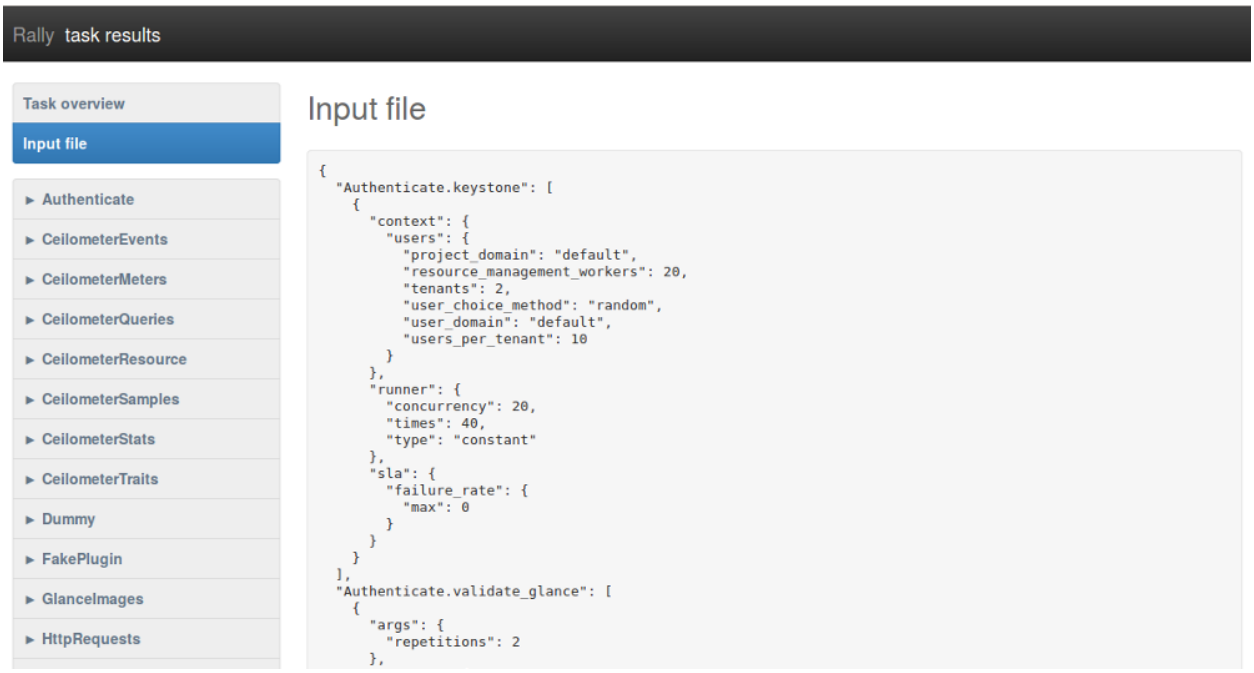
Number of failed iterations. Failure means that there was an Exception raised.

Success (SLA)

This is a boolean result of workload SLA. See *Service-level agreement explanation* below.

Input file

This shows JSON which can be used to run a task with exactly the same workloads list and configuration. This is not an exact copy (neither concatenation) of actually used input files (in command *rally task start*), however this is exactly what is needed to run workloads given in the report.



Tab «Overview»

Service-level agreement

SLA results appear in task report only if *"sla"* section is defined in task input file.

For example, having this in task input file:

```
"sla": {
  "performance_degradation": {
    "max_degradation": 50
  },
  "max_seconds_per_iteration": 1.0,
  "failure_rate": {
    "max": 0
  },
  "outliers": {
    "max": 1,
    "min_iterations": 10,
    "sigmas": 10
  },
  "max_avg_duration": 0.5
}
```

will result SLA section similar to the following:

Service-level agreement

Criterion	Detail	Success
performance_degradation	Current degradation: 0.952549% - Passed	True
max_seconds_per_iteration	Maximum seconds per iteration 0.25s <= 1.00s - Passed	True
failure_rate	Failure rate criteria 0.00% <= 0.00% <= 0.00% - Passed	True
outliers	Maximum number of outliers 0 <= 1 - Passed	True
max_avg_duration	Average duration of one iteration 0.25s <= 0.50s - Passed	True

What if workload has no "sla" configuration in input file?

If *"sla"* section is missed in input file, then block *Service-level agreement* is not displayed and its result is assumed to be always passed (no matter how many failures occurred).

Total durations

There is a durations analysis, which is represented by statistics table and duration StackedArea chart.

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
keystone.create_user	0.343	0.549	0.732	0.736	0.74	0.553	100.0%	10
keystone.delete_user	0.252	0.48	0.559	0.58	0.6	0.448	100.0%	10
total	**0.713**	**0.975**	**1.249**	**1.26**	**1.27**	**1.001**	**100.0%**	**10**

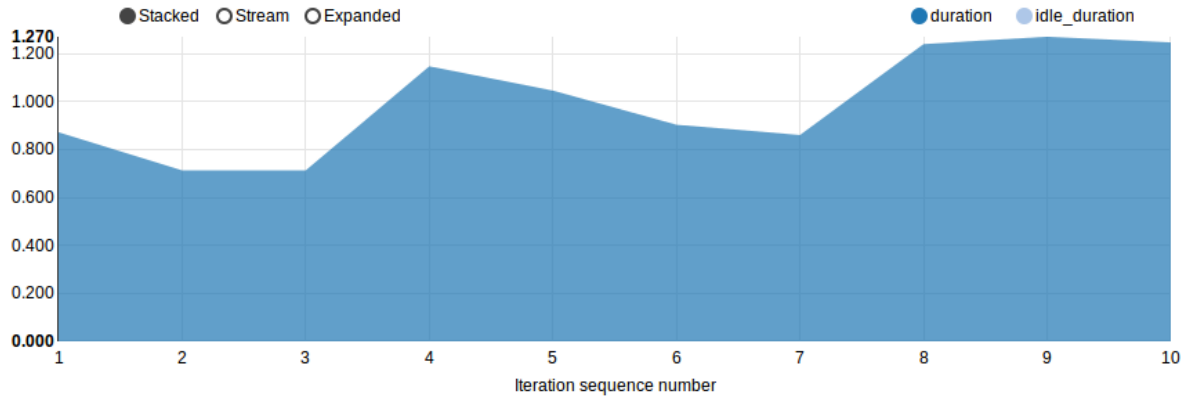


Table with statistics data

Action Name of the workload metric that has some duration saved. This is either an atomic action name or *Total* which points to workload *load duration*.

Min (sec) Minimal duration value

Median (sec) Median duration value

90%ile (sec) Percentile for 90% durations

95%ile (sec) Percentile for 95% durations

Max (sec) Maximal duration value

Avg (sec) Average duration value

Success Percent of successful runs. This is how many percent of this action runs (number of runs is given in *Count* column) were successful.

Count Number of actually run atomic actions. This can differ from *iterations count* because some atomic actions do not start if some exception is raised before in the workload runtime (for example in previous atomic action).

StackedArea with durations per iteration

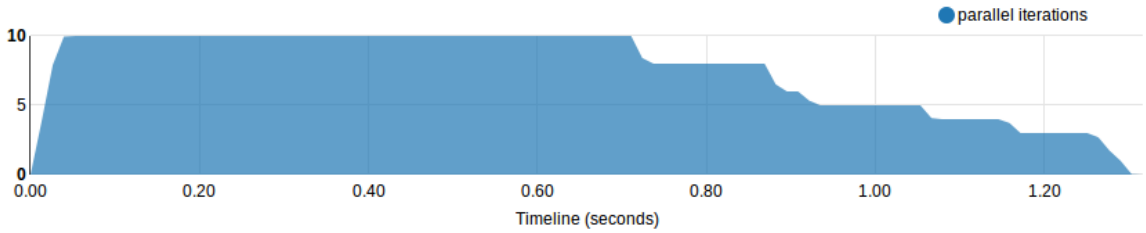
This chart shows *load_duration* and *idle_duration* values per iteration. If there is only one iteration, then chart is useless so it is hidden.

Idle duration

Sometimes workload does nothing for some reason (waiting for something or just making a dummy load). This is achieved by calling *time.sleep()* and spent time is called *idle duration*.

Load Profile

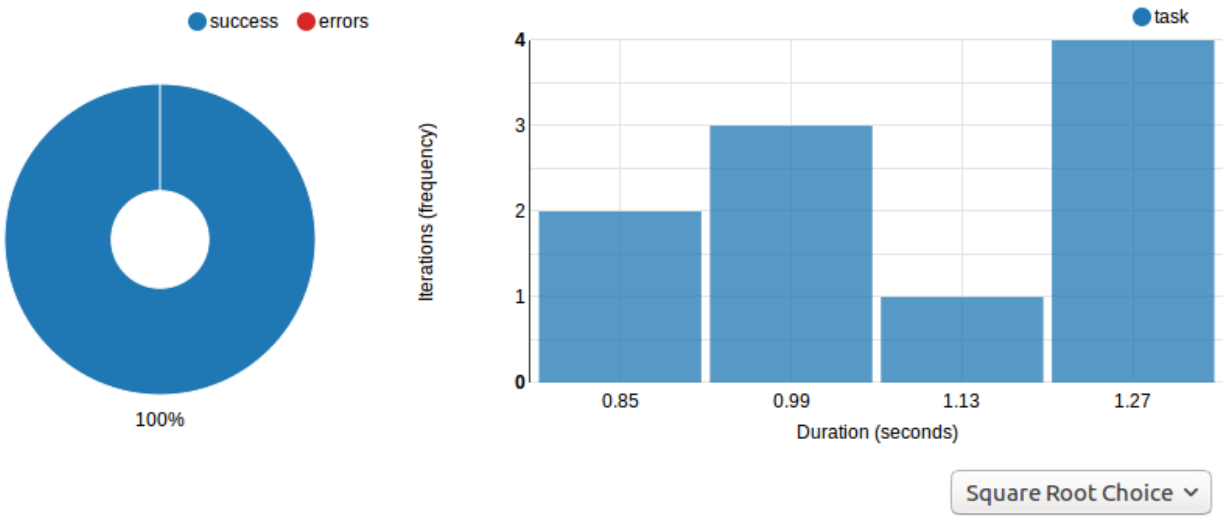
Load profile chart shows number of iterations running in parallel for each workload moment:



Distribution

Pie chart shows percent of successful and failed *iterations*.

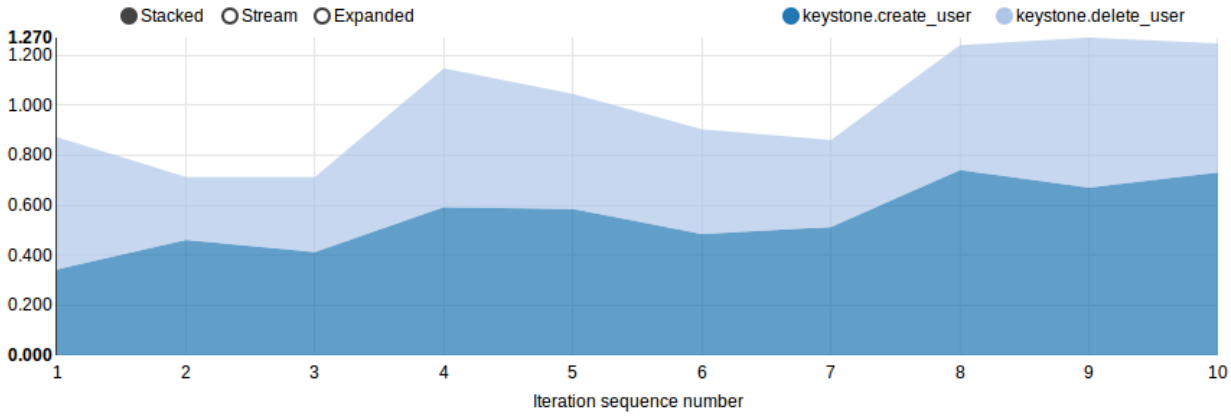
Histogram shows durations distribution with the following methods (selected in dropdown list): **Square Root Choice**, **Sturges Formula**, **Rise Rule**



Tab «Details»

Atomic Action Durations

There is a StackedArea chart that shows atomic actions durations per iteration. If there is only one iteration, then chart is useless so it is hidden.



Distribution

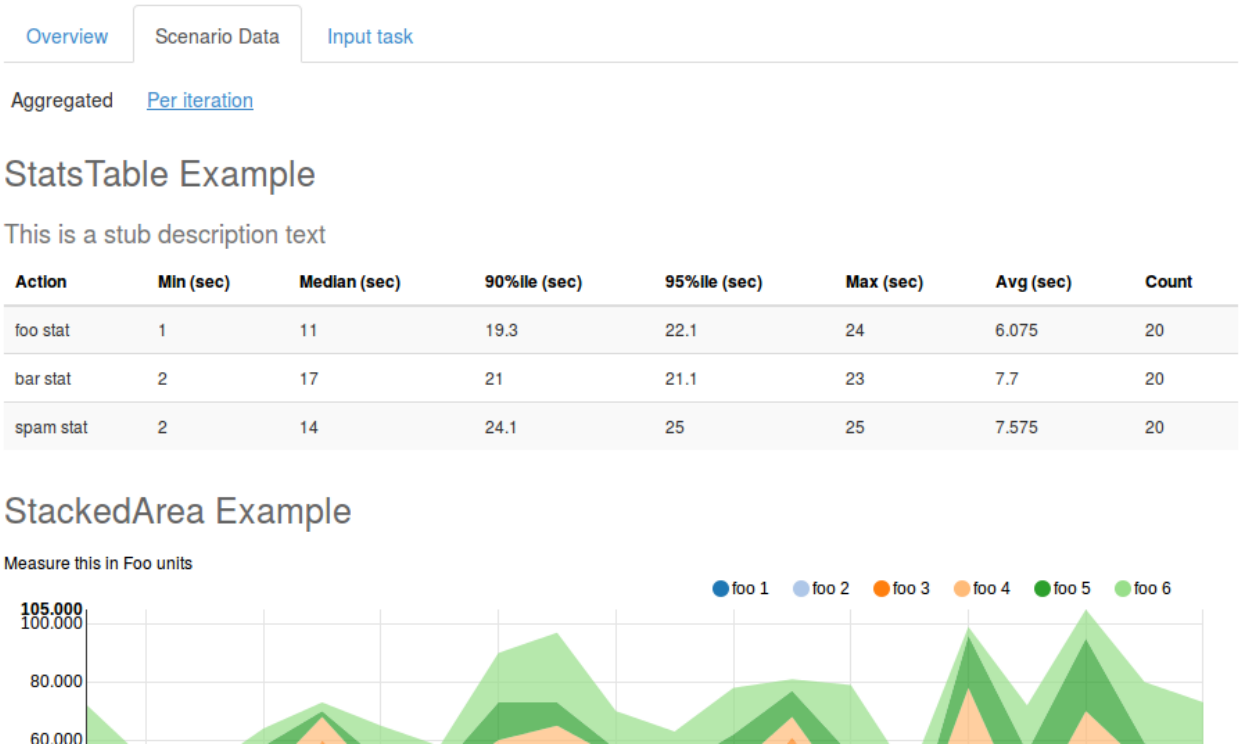
Distribution for atomic actions durations

Tab «Scenario Data»

This tab only appears if workload provides some custom output via method *Scenario.add_output()*.

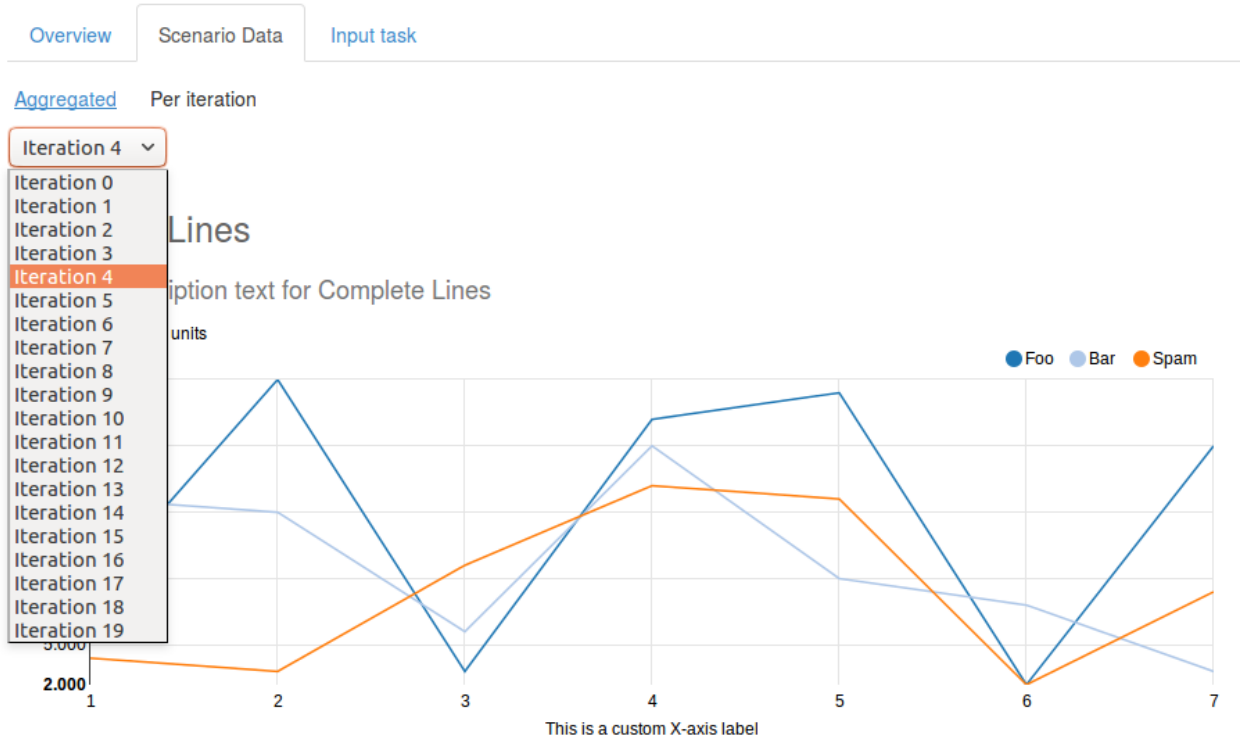
Aggregated

This shows charts with data aggregated from all iterations. This means that each X axis point represents an iteration, so each iteration provided some values that are aggregated into charts or tables.



Per iteration

Each iteration can create its own, complete charts and tables.



Tab «Failures»

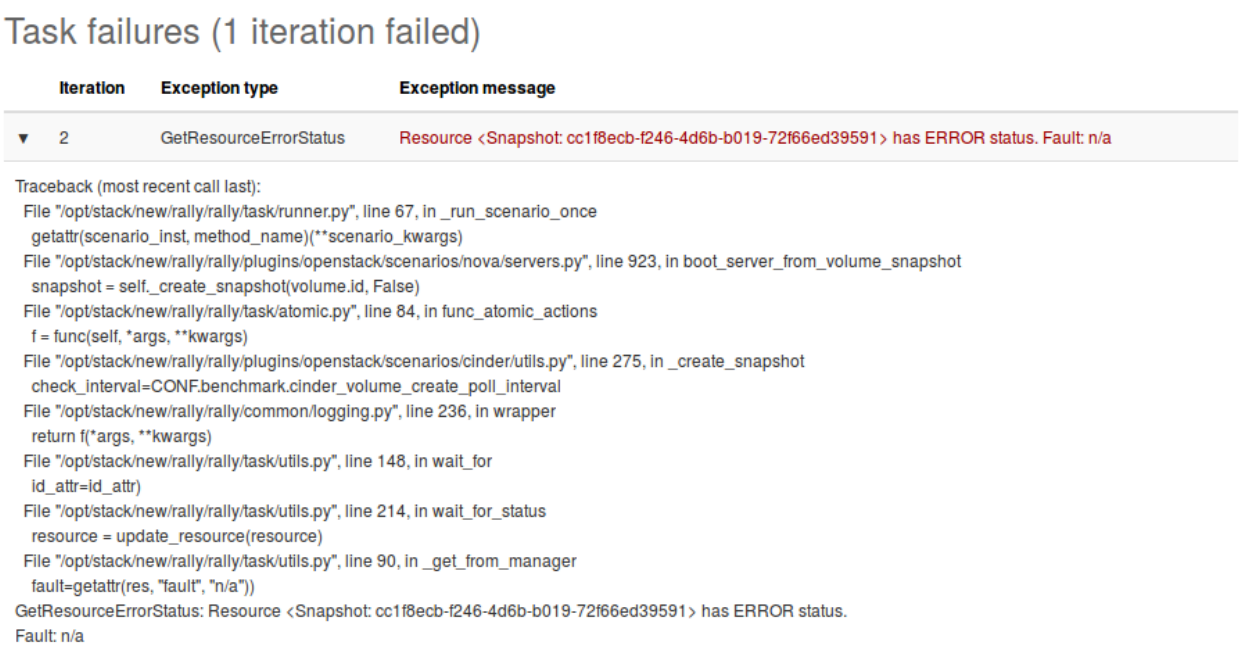
Complete information about exceptions raised during the workload run

Iteration Number of iteration where exception is occurred

Exception type Type of raised Exception subclass

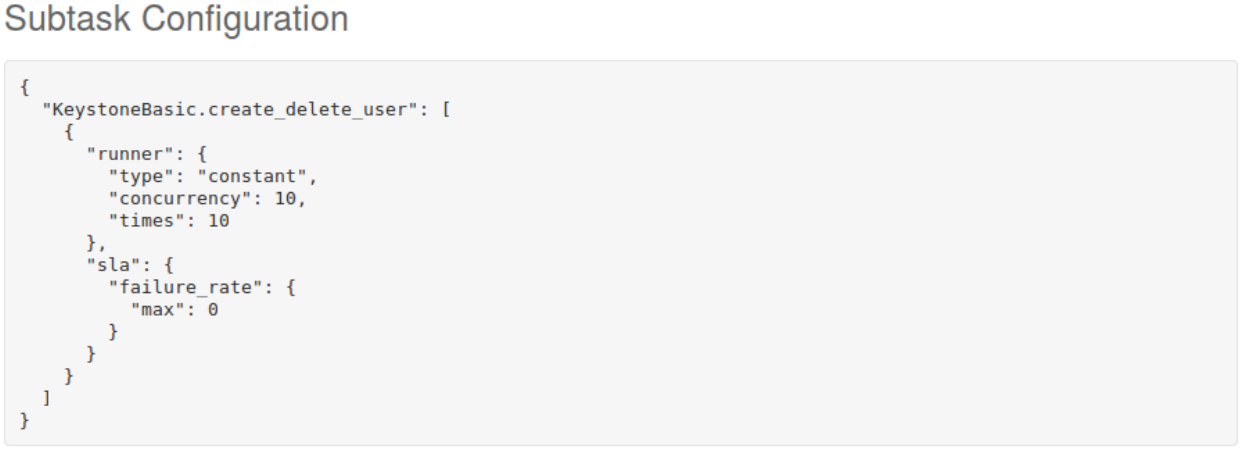
Exception message Message delivered by the exception

Click on a row expands it with exception traceback.



Tab «Input Task»

This shows JSON for input file which can be used to run current workload.



Trends Report

If same workload is run several times, some results of these runs can be compared. Compared metrics are ssuccess rate (percent of successful iterations) and statistics for durations.

How to generate trends report

Use command *rally task trends* with given tasks UUIDs and/or tasks results JSON files and the name of desired output file.

Example:

```
$ rally task trends --tasks 6f63d9ec-eecd-4696-8e9c-2ba065c68535 a5737eba-a204-43d6-
↪a262-d5ea4b0065da --out trends.html
```

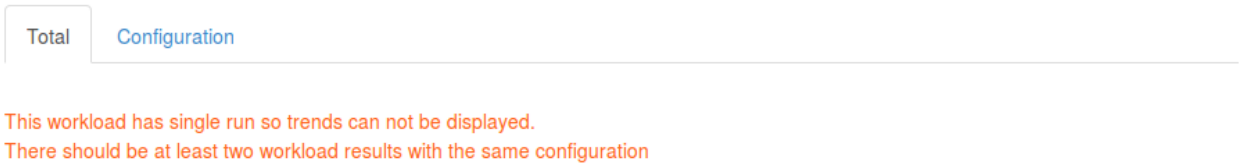
What is an order of workload runs?

Workload run number in shown on charts X axis, the order of runs is exactly as it comes from tasks data in the moment of report generation.

Trends overview

If workload has been actually run only once

That is obvious that it is not possible to have trend for a single value. There should be at least two workload runs to make results comparison possible. So in this case there is only a help message displayed.



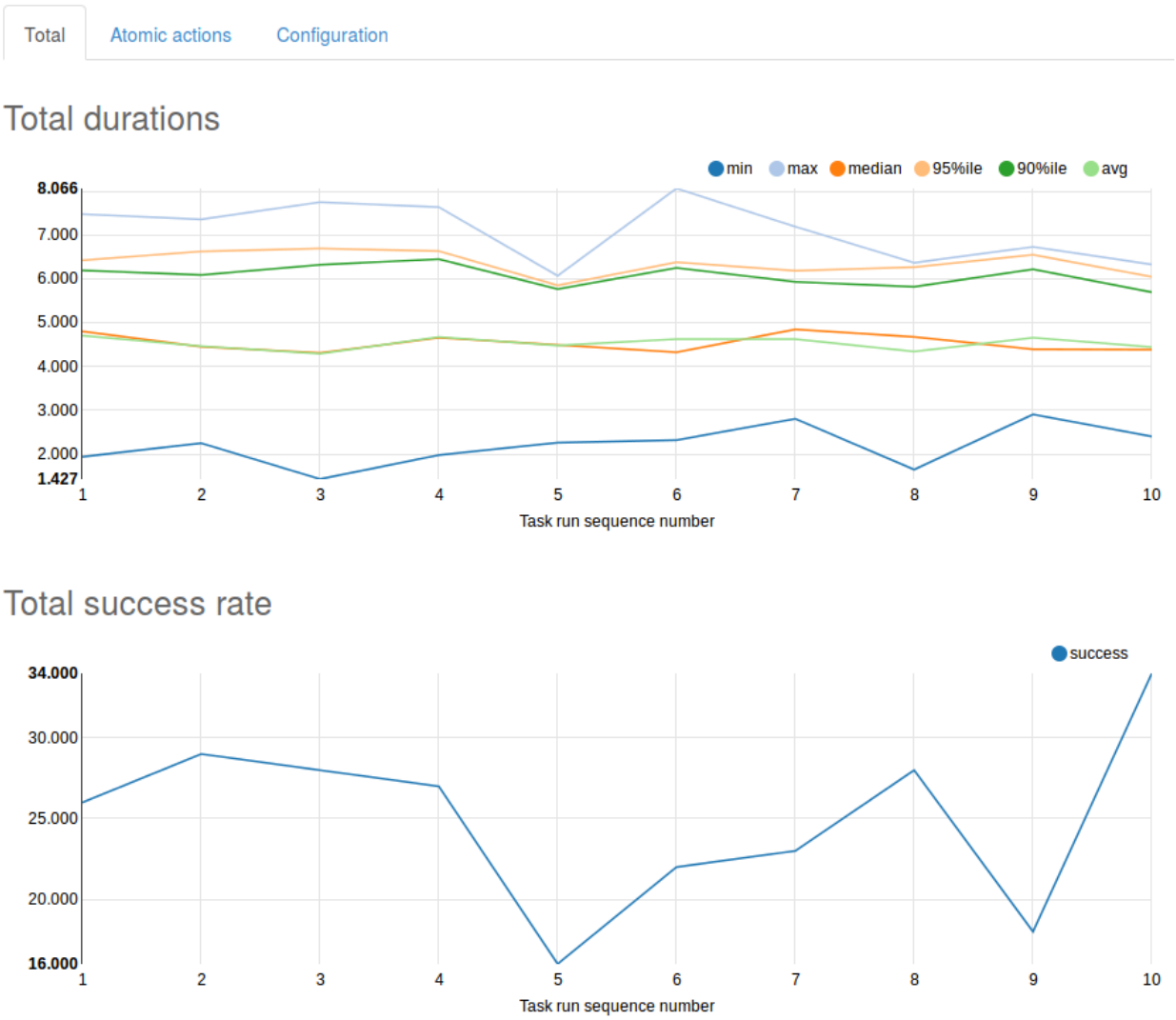
Tab «Total»

Total durations

Shows workload *load_duration* statistics trends.

Total success rate

Shows trends for percent of successful iterations

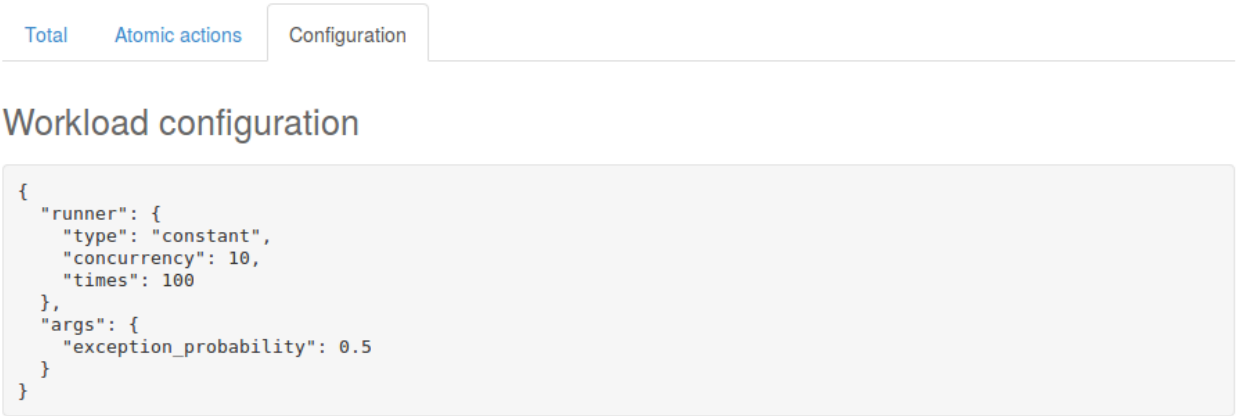


Tab «Atomic actions»

Statistics trends for atomic actions durations. Charts are same as for total durations.

Tab «Configuration»

Here is a configuration JSON for current workload.



CLI References

For more information regarding Rally Task Component CLI please proceed to CLI reference

Verification Component

Functional testing is a first step to ensuring that your product works as expected and API covers all use-cases. Rally Verification Component is all about this. It is not designed to generate a real big load (for this job we have *Task Component*), but it should be enough to check that your environment works by different tools (we call them *Verification*).

Verifiers

What is it?

Verifier Plugin is a compatibility layer between Rally and the specific tool (such as Tempest) which runs tests. It implements features like installation, configuration, upgrades, running, etc in terms of the tool. It is a driver in other words. It is a pluggable entity, which means that you can easily add support for whatever tool you want (see *HowTo add support for new tool* page for more information). Even more, you can deliver such plugin separately from Rally itself, but we firmly recommend to push a change to Rally upstream (see *Contribute to Rally* guide), so Rally core-team will able to review it and help to improve.

Verifier is an instance of the Verifier Plugin. It is an installed tool. For example, "Tempest" is a set of functional tests, it is Verifier Plugin (we have a plugin for it). Installed Tempest 12.0 from https://github.com/openstack/tempest in a virtual environment is the verifier.

Verifier is not aligned to any particular deployment like it was in the past, you can use one verifier for testing unlimited number of deployments (each deployment will have separate configuration files for the tool).

Verifier & Verifier Plugin are the main entities which Verification component operates with. Another one is the verifications results.

Verifier statuses

All verifiers can be in next statuses:

- *init* - Initial state. It appears while you call `rally verify create-verifier` command and installation step is not yet started.
- *installing* - Installation of the verifier is not a quick task. It is about cloning tool, checking packages or installing virtual environments with all required packages. This state indicates that this step is in the process.
- *installed* - It should be one of your favourite states. It means that everything is ok and you can start verifying your cloud.

- *updating* - This state identifies the process of updating verifier (version, source, packages, etc.).
- *extending* - The process of extending a verifier by its plugins.
- *failed* - Something went wrong while installation.

Verification statuses

- *init* - Initial state. It appears instantly after calling `rally verify start` command before the actual run of verifier's tool.
- *running* - Identifies the process of execution tool.
- *finished*- Verification is finished without errors and failures.
- *failed* - Verification is finished, but there are some failed tests.
- *crashed* - Unexpected error had happened while running verification.

Known verifier types

Out of the box

You can execute command `rally verify list-plugins` locally to check available verifiers in your environment.

Cut down from Global *Plugins Reference* page:

tempest

Tempest verifier.

Description:

Quote from official documentation:

> This is a set of integration tests to be run against a live OpenStack cluster. Tempest has batteries of tests for OpenStack API validation, Scenarios, and other specific tests useful in validating an OpenStack deployment.

Rally supports features listed below:

- *cloning Tempest*: repository and version can be specified
- *installation*: system-wide with checking existence of required packages or in virtual environment
- *configuration*: options are discovered via OpenStack API, but you can override them if you need
- *running*: pre-creating all required resources(i.e images, tenants, etc), prepare arguments, launching Tempest, live-progress output
- *results*: all verifications are stored in db, you can built reports, compare verification at whatever you want time.

Appeared in Rally 0.8.0 *(actually, it appeared long time ago with first revision of Verification Component, but 0.8.0 is mentioned since it is first release after Verification Component redesign)*

Running arguments:

- *concurrency*: Number of processes to be used for launching tests. In case of 0 value, number of processes will be equal to number of CPU cores.

- *load_list*: a list of tests to launch.
- *pattern*: a regular expression of tests to launch.
- *set*: Name of predefined set of tests. Known names: full, smoke, baremetal, clustering, compute, database, data_processing, identity, image, messaging, network, object_storage, orchestration, telemetry, volume, scenario
- *skip_list*: a list of tests to skip (actually, it is a dict where keys are names of tests, values are reasons).
- *xfail_list*: a list of tests that are expected to fail (actually, it is a dict where keys are names of tests, values are reasons).

Installation arguments:

- *system_wide*: Whether or not to use the system-wide environment for verifier instead of a virtual environment. Defaults to False.
- *source*: Path or URL to the repo to clone verifier from. Defaults to https://git.openstack.org/openstack/tempest
- *version*: Branch, tag or commit ID to checkout before verifier installation. Defaults to 'master'.

Namespace: openstack

Module: rally.plugins.openstack.verification.tempest.manager

Third-party

Nothing here yet.

Verification reports

Rally stores all verifications results in its DataBase so that you can access and process results at any time. No matter what verifier you use, results will be stored in a unified way and reports will be unified too.

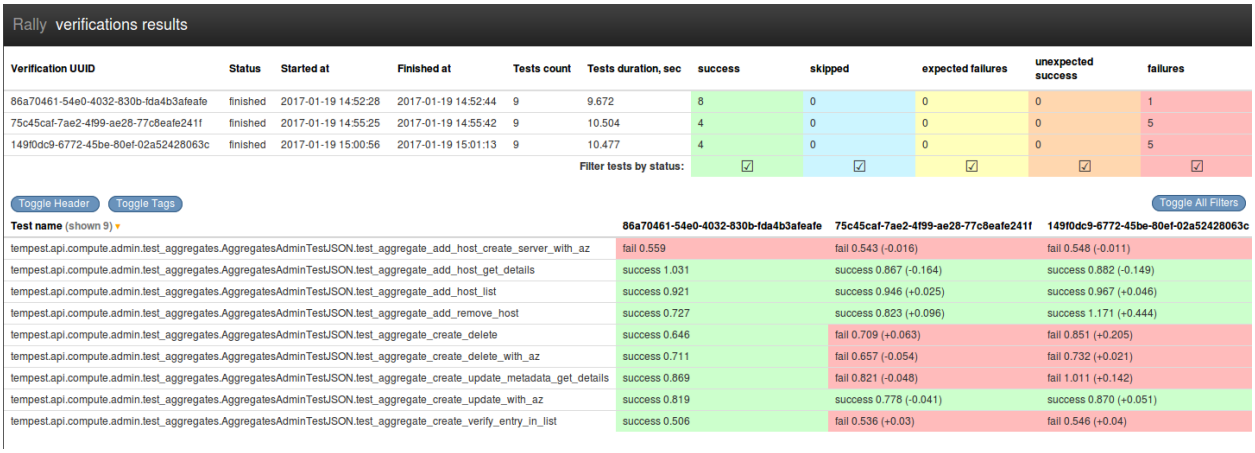
We support several types of reports out of the box: HTML, HTML-Static, JSON, JUnit-XML; but our reporting system is pluggable so that you can write your own plugin to build some specific reports or to export results to the specific system (see *HowTo add new reporting mechanism* for more details').

HTML reports

HTML report is the most convenient type of reports. It includes as much as possible useful information about Verifications.

Here is an example of HTML report for 3 verifications. It was generated by next command:

```
$ rally verify report --uuid <uuid-1> <uuid-2> <uuid-3> --type html \
  --to ./report.html
```



Rally verifications results

Verification UUID	Status	Started at	Finished at	Tests count	Tests duration, sec	success	skipped	expected failures	unexpected success	failures
86a70461-54e0-4032-830b-fda4b3afeafe	finished	2017-01-19 14:52:28	2017-01-19 14:52:44	9	9.672	8	0	0	0	1
75c45caf-7ae2-4f99-ae28-77c8eafe241f	finished	2017-01-19 14:55:25	2017-01-19 14:55:42	9	10.504	4	0	0	0	5
149f0dc9-6772-45be-80ef-02a52428063c	finished	2017-01-19 15:00:56	2017-01-19 15:01:13	9	10.477	4	0	0	0	5
					Filter tests by status:	☑	☑	☑	☑	☑

Toggle Header | Toggle Tags | Toggle All Filters

Test name (shown 9)	86a70461-54e0-4032-830b-fda4b3afeafe	75c45caf-7ae2-4f99-ae28-77c8eafe241f	149f0dc9-6772-45be-80ef-02a52428063c
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_az	fail 0.559	fail 0.543 (-0.016)	fail 0.548 (-0.011)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_get_details	success 1.031	success 0.867 (-0.164)	success 0.882 (-0.149)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_list	success 0.921	success 0.946 (+0.025)	success 0.967 (+0.046)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_remove_host	success 0.727	success 0.823 (+0.096)	success 1.171 (+0.444)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete	success 0.646	fail 0.709 (+0.063)	fail 0.851 (+0.205)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete_with_az	success 0.711	fail 0.657 (-0.054)	fail 0.732 (+0.021)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_metadata_get_details	success 0.869	fail 0.821 (-0.048)	fail 1.011 (+0.142)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_with_az	success 0.819	success 0.778 (-0.041)	success 0.870 (+0.051)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_verify_entry_in_list	success 0.506	fail 0.536 (+0.03)	fail 0.546 (+0.04)

The report consists of two tables.

First one is a summary table. It includes base information about verifications: UUIDs; numbers of tests; when they were launched; statuses; etc. Also, you can find detailed information grouped by tests statuses at the right part of the table.

If the size (height) of the summary table seems too large for you and hinders to see more tests results, you can push "Toggle Header" button.

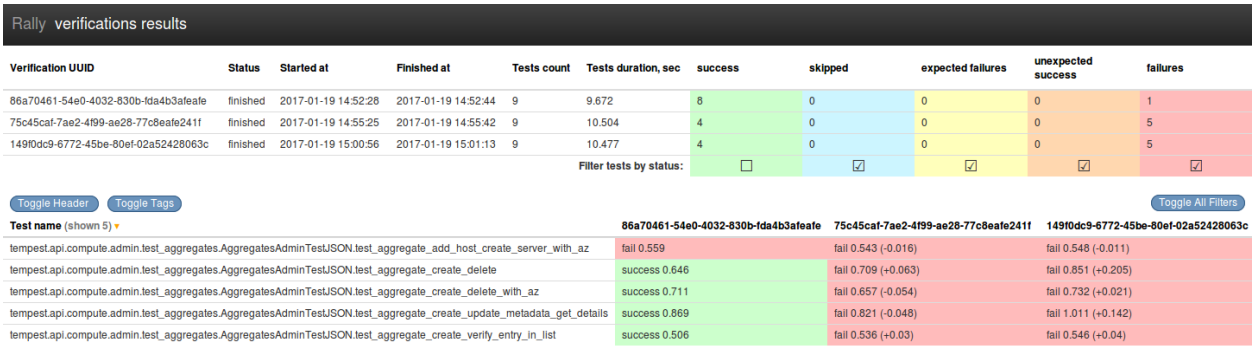
The second table contains actual verifications results. They are grouped by tests names. The result of the test for particular verification overpainted by one of the next colours:

- *Red* - It means that test has "failed" status
- *Orange* - It is "unexpected success". Most of the parsers calculates it just like failure
- *Green* - Everything is ok. The test succeeded.
- *Yellow* - It is "expected failure".
- *Light Blue* - Test is skipped. It is not good and not bad

Several verifications comparison is a default embedded behaviour of reports. The difference between verifications is displayed in brackets after actual test duration. Sign **+** means that current result is bigger that standard by the number going after the sign. Sign **-** is an opposite to **+**. Please, note that all diffs are comparisons with the first verification in a row.

Filtering results

You can filter tests by setting or removing a mark from check box of the particular status column of the summary table.



Rally verifications results

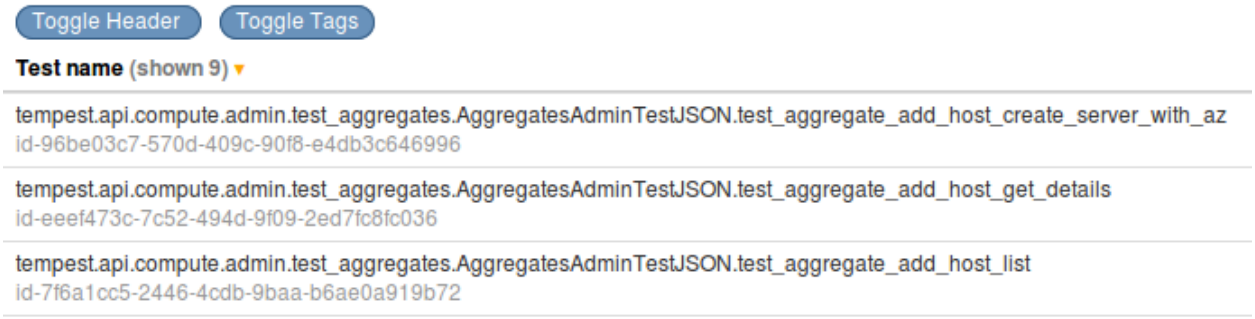
Verification UUID	Status	Started at	Finished at	Tests count	Tests duration, sec	success	skipped	expected failures	unexpected success	failures
86a70461-54e0-4032-830b-fda4b3afeafe	finished	2017-01-19 14:52:28	2017-01-19 14:52:44	9	9.672	8	0	0	0	1
75c45caf-7ae2-4f99-ae28-77c8eafe241f	finished	2017-01-19 14:55:25	2017-01-19 14:55:42	9	10.504	4	0	0	0	5
149f0dc9-6772-45be-80ef-02a52428063c	finished	2017-01-19 15:00:56	2017-01-19 15:01:13	9	10.477	4	0	0	0	5
					Filter tests by status:	☐	☑	☑	☑	☑

Toggle Header Toggle Tags Toggle All Filters

Test name (shown 5)	86a70461-54e0-4032-830b-fda4b3afeafe	75c45caf-7ae2-4f99-ae28-77c8eafe241f	149f0dc9-6772-45be-80ef-02a52428063c
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_az	fail 0.559	fail 0.543 (-0.016)	fail 0.548 (-0.011)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete	success 0.646	fail 0.709 (+0.063)	fail 0.851 (+0.205)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete_with_az	success 0.711	fail 0.657 (-0.054)	fail 0.732 (+0.021)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_update_metadata_get_details	success 0.869	fail 0.821 (-0.048)	fail 1.011 (+0.142)
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_verify_entry_in_list	success 0.506	fail 0.536 (+0.03)	fail 0.546 (+0.04)

Tests Tags

Some of the tests tools support tests tagging. It can be used for setting unique IDs, groups, etc. Usually, such tags are included in test name. It is inconvenient and Rally stores tags separately. By default they are hidden, but if you push "Toggle tags" button, they will be displayed under tests names.



Toggle Header Toggle Tags

Test name (shown 9)

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_az
id-96be03c7-570d-409c-90f8-e4db3c646996

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_get_details
id-eeef473c-7c52-494d-9f09-2ed7fc8fc036

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_list
id-7f6a1cc5-2446-4cdb-9baa-b6ae0a919b72

Tracebacks & Reasons

Tests with "failed" and "expected failure" statuses have tracebacks of failures. Tests with "skipped", "expected failure", "unexpected success" status has "reason" of events. By default, both tracebacks and reasons are hidden, but you can show them by clicking on the appropriate test.

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete | success 0.646 | fail 0.709 (+0.063) | fail 0.851 (+0.205)

75c45caf-7ae2-4f99-ae28-77c8eafe241f

```
Traceback (most recent call last):
  File "tempest/api/compute/admin/test_aggregates.py", line 76, in test_aggregate_create_delete
    self.assertEqual(1, 0)
  File "/home/rally-user/.rally/verification/verifier-c939f5aa-ee74-4eae-9864-7a6dac355de3/.venv/local/lib/python2.7/site-packages/testtools/testcase.py", line 411, in assertEqual
    self.assertThat(observed, matcher, message)
  File "/home/rally-user/.rally/verification/verifier-c939f5aa-ee74-4eae-9864-7a6dac355de3/.venv/local/lib/python2.7/site-packages/testtools/testcase.py", line 498, in assertThat
    raise mismatch_error
testtools.matchers._impl.MismatchError: 1 != 0
```

149f0dc9-6772-45be-80ef-02a52428063c

```
Traceback (most recent call last):
  File "tempest/api/compute/admin/test_aggregates.py", line 76, in test_aggregate_create_delete
    self.assertEqual(1, 0)
  File "/home/rally-user/.rally/verification/verifier-c939f5aa-ee74-4eae-9864-7a6dac355de3/.venv/local/lib/python2.7/site-packages/testtools/testcase.py", line 411, in assertEqual
    self.assertThat(observed, matcher, message)
  File "/home/rally-user/.rally/verification/verifier-c939f5aa-ee74-4eae-9864-7a6dac355de3/.venv/local/lib/python2.7/site-packages/testtools/testcase.py", line 498, in assertThat
    raise mismatch_error
testtools.matchers._impl.MismatchError: 1 != 0
```

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_create_delete_with_az | success 0.711 | fail 0.657 (-0.054) | fail 0.732 (+0.021)

Test name (shown 9)	f10e3bfa-443a-42fe-b7cf-1a134e8f4937
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_create_server_with_az	xfail 0.652

f10e3bfa-443a-42fe-b7cf-1a134e8f4937

Some reason why this test fails

```
Traceback (most recent call last):
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/api/compute/admin/test_aggregates.py", line 226, in test_aggregate_add_host_create_server_with_az
    self.client.add_host(aggregate['id'], host=self.host)
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/lib/services/compute/aggregates_client.py", line 95, in add_host
    post_body)
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/lib/common/rest_client.py", line 275, in post
    return self.request('POST', url, extra_headers, headers, body, chunked)
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/lib/services/compute/base_compute_client.py", line 48, in request
    method, url, extra_headers, headers, body, chunked)
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/lib/common/rest_client.py", line 663, in request
    self._error_checker(resp, resp_body)
  File "/home/ylobankov/.rally/verification/verifier-d7fd9aa0-6b4d-4606-bd7d-3aea8f7801ce/repo/tempest/lib/common/rest_client.py", line 775, in _error_checker
    raise exceptions.Conflict(resp_body, resp=resp)
tempest.lib.exceptions.Conflict: An object with that identifier already exists
Details: {u'message': u"Cannot add host to aggregate 2640. Reason: One or more hosts already in availability zone(s) [u'tempest-test_az-34611847'].", u'code': 409}
```

tempest.api.compute.admin.test_aggregates.AggregatesAdminTestJSON.test_aggregate_add_host_get_details	success 0.953

Plugins Reference for all out-of-the-box reporters

html

Generates verification report in HTML format.

Namespace: default

Module: rally.plugins.common.verification.reporters

html-static

Generates verification report in HTML format with embedded JS/CSS.

Namespace: default

Module: rally.plugins.common.verification.reporters

json

Generates verification report in JSON format.

An example of the report (All dates, numbers, names appearing in this example are fictitious. Any resemblance to real things is purely coincidental):

```
{"verifications": {
    "verification-uuid-1": {
        "status": "finished",
        "skipped": 1,
        "started_at": "2001-01-01T00:00:00",
        "finished_at": "2001-01-01T00:05:00",
        "tests_duration": 5,
        "run_args": {
            "pattern": "set=smoke",
            "xfail_list": {"some.test.TestCase.test_xfail":
                           "Some reason why it is expected."},
            "skip_list": {"some.test.TestCase.test_skipped":
                          "This test was skipped intentionally"},
        },
        "success": 1,
        "expected_failures": 1,
        "tests_count": 3,
        "failures": 0,
        "unexpected_success": 0
    },
```

```
    "verification-uuid-2": {
        "status": "finished",
        "skipped": 1,
        "started_at": "2002-01-01T00:00:00",
        "finished_at": "2002-01-01T00:05:00",
        "tests_duration": 5,
        "run_args": {
            "pattern": "set=smoke",
            "xfail_list": {"some.test.TestCase.test_xfail":
                           "Some reason why it is expected."},
            "skip_list": {"some.test.TestCase.test_skipped":
                          "This test was skipped intentionally"},
        },
        "success": 1,
        "expected_failures": 1,
        "tests_count": 3,
        "failures": 1,
        "unexpected_success": 0
    }
},
"tests": {
    "some.test.TestCase.test_foo[tag1,tag2]": {
        "name": "some.test.TestCase.test_foo",
        "tags": ["tag1","tag2"],
        "by_verification": {
            "verification-uuid-1": {
                "status": "success",
                "duration": "1.111"
            },
            "verification-uuid-2": {
                "status": "success",
                "duration": "22.222"
            }
        }
    },
    "some.test.TestCase.test_skipped[tag1]": {
        "name": "some.test.TestCase.test_skipped",
        "tags": ["tag1"],
        "by_verification": {
            "verification-uuid-1": {
                "status": "skipped",
                "duration": "0",
                "details": "Skipped until Bug: 666 is resolved."
            },
            "verification-uuid-2": {
                "status": "skipped",
                "duration": "0",
                "details": "Skipped until Bug: 666 is resolved."
            }
        }
    },
    "some.test.TestCase.test_xfail": {
        "name": "some.test.TestCase.test_xfail",
        "tags": [],
        "by_verification": {
            "verification-uuid-1": {
                "status": "xfail",
                "duration": "3",
```

```
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
            },
            "verification-uuid-2": {
                "status": "xfail",
                "duration": "3",
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
            }
        }
    },
    "some.test.TestCase.test_failed": {
        "name": "some.test.TestCase.test_failed",
        "tags": [],
        "by_verification": {
            "verification-uuid-2": {
                "status": "fail",
                "duration": "4",
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
                }
            }
        }
    }
}
```

Namespace: default

Module: rally.plugins.common.verification.reporters

junit-xml

Generates verification report in JUnit-XML format.

An example of the report (All dates, numbers, names appearing in this example are fictitious. Any resemblance to real things is purely coincidental):

```
<testsuites>
  <!--Report is generated by Rally 0.8.0 at 2002-01-01T00:00:00-->
  <testsuite id="verification-uuid-1"
             tests="9"
             time="1.111"
             errors="0"
             failures="3"
             skipped="0"
             timestamp="2001-01-01T00:00:00">
    <testcase classname="some.test.TestCase"
              name="test_foo"
              time="8"
              timestamp="2001-01-01T00:01:00" />
    <testcase classname="some.test.TestCase"
              name="test_skipped"
              time="0"
              timestamp="2001-01-01T00:02:00">
      <skipped>Skipped until Bug: 666 is resolved.</skipped>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_xfail"
              time="3"
              timestamp="2001-01-01T00:03:00">
      <!--It is an expected failure due to: something-->
      <!--Traceback:
HEEELP-->
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_uxsuccess"
              time="3"
              timestamp="2001-01-01T00:04:00">
      <failure>
          It is an unexpected success. The test should fail due to:
          It should fail, I said!
      </failure>
    </testcase>
  </testsuite>
  <testsuite id="verification-uuid-2"
             tests="99"
             time="22.222"
             errors="0"
             failures="33"
             skipped="0"
             timestamp="2002-01-01T00:00:00">
    <testcase classname="some.test.TestCase"
              name="test_foo"
              time="8"
```

```
              timestamp="2001-02-01T00:01:00" />
    <testcase classname="some.test.TestCase"
              name="test_failed"
              time="8"
              timestamp="2001-02-01T00:02:00">
      <failure>HEEEEEEELP</failure>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_skipped"
              time="0"
              timestamp="2001-02-01T00:03:00">
      <skipped>Skipped until Bug: 666 is resolved.</skipped>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_xfail"
              time="4"
              timestamp="2001-02-01T00:04:00">
      <!--It is an expected failure due to: something-->
      <!--Traceback:
HEEELP-->
    </testcase>
  </testsuite>
</testsuites>
```

Namespace: default

Module: rally.plugins.common.verification.reporters

Command Line Interface

Cut down from Global *Command Line Interface*

- *Category: verify*
  - *rally verify add-verifier-ext*
  - *rally verify configure-verifier*
  - *rally verify create-verifier*
  - *rally verify delete*
  - *rally verify delete-verifier*
  - *rally verify delete-verifier-ext*
  - *rally verify import*
  - *rally verify list*
  - *rally verify list-plugins*
  - *rally verify list-verifier-exts*
  - *rally verify list-verifier-tests*
  - *rally verify list-verifiers*
  - *rally verify report*

Category: verify

Verify an OpenStack cloud via a verifier.

rally verify add-verifier-ext

Add a verifier extension.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–source <source>* [*ref*]

  Path or URL to the repo to clone verifier extension from.

  type: str

  default: none

- *–version <version>* [*ref*]

  Branch, tag or commit ID to checkout before installation of the verifier extension (the 'master' branch is used by default).

  type: str

  default: none

- *–extra-settings <extra_settings>* [*ref*]

  Extra installation settings for verifier extension.

  type: str

  default: none

rally verify configure-verifier

Configure a verifier for a specific deployment.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

  type: str

- *–reconfigure* [*ref*]

  Reconfigure verifier.

- *–extend <path/json/yaml>* [*ref*]

  Extend verifier configuration with extra options. If options are already present, the given ones will override them. Can be a path to a regular config file or just a json/yaml.

  type: str

  default: none

- *–override <path>* [*ref*]

  Override verifier configuration by another one from a given source.

  type: str

  default: none

- *–show* [*ref*]

  Show verifier configuration.

rally verify create-verifier

Create a verifier.

Command arguments:

- *–name <name>* [*ref*]

  Verifier name (for example, 'My verifier').

  type: str

- *–type <type>* [*ref*]

Verifier plugin name. HINT: You can list all verifier plugins, executing command *rally verify list-plugins*.

type: str

- *–namespace <name>* [*ref*]

  Verifier plugin namespace. Should be specified in case of two verifier plugins with equal names but in different namespaces.

  type: str

  default:

- *–source <source>* [*ref*]

  Path or URL to the repo to clone verifier from.

  type: str

  default: none

- *–version <version>* [*ref*]

  Branch, tag or commit ID to checkout before verifier installation (the 'master' branch is used by default).

  type: str

  default: none

- *–system-wide* [*ref*]

  Use the system-wide environment for verifier instead of a virtual environment.

- *–extra-settings <extra_settings>* [*ref*]

  Extra installation settings for verifier.

  type: str

  default: none

- *–no-use* [*ref*]

  Not to set the created verifier as the default verifier for future operations.

rally verify delete

Delete a verification or a few verifications.

Command arguments:

- *–uuid <uuid>* [*ref*]

  UUIDs of verifications. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

rally verify delete-verifier

Delete a verifier.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

type: str

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. If specified, only the deployment-specific data will be deleted for verifier. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–force* [*ref*]

Delete all stored verifications of the specified verifier. If a deployment specified, only verifications of this deployment will be deleted. Use this argument carefully! You can delete verifications that may be important to you.

rally verify delete-verifier-ext

Delete a verifier extension.

Command arguments:

- *–id <id>* [*ref*]

Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

type: str

default: none

- *–name <name>* [*ref*]

Verifier extension name.

type: str

default: none

rally verify import

Import results of a test run into the Rally database.

Command arguments:

- *–id <id>* [*ref*]

Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

type: str

default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–file <path>* [*ref*]

  File to import test results from.

  type: str

  default: none

- *–run-args <run_args>* [*ref*]

  Arguments that might be used when running tests. For example, '{concurrency: 2, pattern: set=identity}'.

  type: str

  default: none

- *–no-use* [*ref*]

  Not to set the created verification as the default verification for future operations.

rally verify list

List all verifications.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id <id>* [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–tag* <*tag*> [*ref*]

  Tags to filter verifications by.

  type: str

  default: none

- *–status* <*status*> [*ref*]

  Status to filter verifications by.

  type: str

  default: none

rally verify list-plugins

List all plugins for verifiers management.

Command arguments:

- *–namespace* <*name*> [*ref*]

  Namespace name (for example, openstack).

  type: str

  default: none

rally verify list-verifier-exts

List all verifier extensions.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

rally verify list-verifier-tests

List all verifier tests.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

default: none

- *–pattern* <*pattern*> [*ref*]

  Pattern which will be used for matching. Can be a regexp or a verifier-specific entity (for example, in case of Tempest you can specify 'set=smoke').

  type: str

  default:

rally verify list-verifiers

List all verifiers.

Command arguments:

- *–status* <*status*> [*ref*]

  Status to filter verifiers by.

  type: str

  default: none

rally verify report

Generate a report for a verification or a few verifications.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  UUIDs of verifications. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–type* <*type*> [*ref*]

  Report type (Defaults to JSON). Out-of-the-box types: HTML, HTML-Static, JSON, JUnit-XML. HINT: You can list all types, executing *rally plugin list –plugin-base VerificationReporter* command.

  type: str

  default: none

- *–to* <*dest*> [*ref*]

  Report destination. Can be a path to a file (in case of HTML, JSON, etc. types) to save the report to or a connection string. It depends on the report type.

  type: str

  default: none

- *–open* [*ref*]

  Open the output file in a browser.

rally verify rerun

Rerun tests from a verification for a specific deployment.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–deployment-id* <*id*> [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

  Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

  type: str

- *–failed* [*ref*]

  Rerun only failed tests.

- *–tag* <*tag*> [*ref*]

  Mark verification with a tag or a few tags.

  type: str

  default: none

- *–concurrency* <*N*> [*ref*]

  How many processes to be used for running verifier tests. The default value (0) auto-detects your CPU count.

  type: int

  default: none

- *–detailed* [*ref*]

  Show verification details such as errors of failed tests.

- *–no-use* [*ref*]

  Not to set the finished verification as the default verification for future operations.

rally verify show

Show detailed information about a verification.

Command arguments:

- *–uuid* <*uuid*> [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

  default: none

- *–sort-by* <*query*> [*ref*]

  Sort tests by 'name', 'duration' or 'status'.

  type: str

  default: name

- *–detailed* [*ref*]

  Show verification details such as run arguments and errors of failed tests.

rally verify show-verifier

Show detailed information about a verifier.

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

rally verify start

Start a verification (run verifier tests).

Command arguments:

- *–id* <*id*> [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–deployment-id* <*id*> [*ref*]

Note: The default value for the `--deployment-id` argument is taken from the Rally environment. Usually, the default value is equal to the UUID of the last successful run of `rally deployment create`, if the `--no-use` argument was not used.

Hint: You can set the default value by executing `rally deployment use <uuid>` (*ref*).

Deployment name or UUID. HINT: You can list all deployments, executing command *rally deployment list*.

type: str

- *–tag <tag>* [*ref*]

  Mark verification with a tag or a few tags.

  type: str

  default: none

- *–pattern <pattern>* [*ref*]

  Pattern which will be used for running tests. Can be a regexp or a verifier-specific entity (for example, in case of Tempest you can specify 'set=smoke').

  type: str

  default: none

- *–concurrency <N>* [*ref*]

  How many processes to be used for running verifier tests. The default value (0) auto-detects your CPU count.

  type: int

  default: 0

- *–load-list <path>* [*ref*]

  Path to a file with a list of tests to run.

  type: str

  default: none

- *–skip-list <path>* [*ref*]

  Path to a file with a list of tests to skip. Format: json or yaml like a dictionary where keys are test names and values are reasons.

  type: str

  default: none

- *–xfail-list <path>* [*ref*]

  Path to a file with a list of tests that will be considered as expected failures. Format: json or yaml like a dictionary where keys are test names and values are reasons.

  type: str

  default: none

- *–detailed* [*ref*]

  Show verification details such as errors of failed tests.

- *–no-use* [*ref*]

  Not to set the finished verification as the default verification for future operations.

rally verify update-verifier

Update a verifier.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

  default: none

- *–update-venv* [*ref*]

  Update the virtual environment for verifier.

- *–version <version>* [*ref*]

  Branch, tag or commit ID to checkout. HINT: Specify the same version to pull the latest repo code.

  type: str

  default: none

- *–system-wide* [*ref*]

  Switch to using the system-wide environment.

- *–no-system-wide* [*ref*]

  Switch to using the virtual environment. If the virtual environment doesn't exist, it will be created.

rally verify use

Choose a verification to use for the future operations.

Command arguments:

- *–uuid <uuid>* [*ref*]

  Verification UUID. HINT: You can list all verifications, executing command *rally verify list*.

  type: str

rally verify use-verifier

Choose a verifier to use for the future operations.

Command arguments:

- *–id <id>* [*ref*]

  Verifier name or UUID. HINT: You can list all verifiers, executing command *rally verify list-verifiers*.

  type: str

HowTo

HowTo add new reporting mechanism

Reporting mechanism for verifications is pluggable. Custom plugins can be used for custom output formats or for exporting results to external systems.

We hardly recommend to read *Rally Plugins* page to understand how do Rally Plugins work.

Spec

All reporters should inherit `rally.verification.reporter.VerificationReporter` and implement all abstract methods. Here you can find its interface:

class `rally.verification.reporter.`**VerificationReporter**(*verifications*, *output_destination*)

Base class for all reporters for verifications.

base_ref
alias of `VerificationReporter`

generate()
Generate report
Returns

a dict with 3 optional elements:
- key "files" with a dictionary of files to save on disk. keys are paths, values are contents;
- key "print" - data to print at CLI level
- key "open" - path to file which should be open in case of –open flag

static make(*reporter_cls*, *verifications*, *output_destination*)
Initialize reporter, generate and validate report.

It is a base method which is called from API layer. It cannot be overridden. Do not even try! :)
Parameters
- **reporter_cls** – class of VerificationReporter to be used
- **verifications** – list of results to generate report for
- **output_destination** – destination of report

classmethod validate(*output_destination*)
Validate destination of report.
Parameters **output_destination** – Destination of report

Example of custom JSON Reporter

Basically, you need to implement only two methods "validate" and "generate".

Method "validate" should check that destination of the report is right. Method "generate" should build a report or export results somewhere; actually, it is up to you what it should do but return format is strict, see *Spec* section for what it can return.

```
import json

from rally.verification import reporter


@reporter.configure("summary-in-json")
class SummaryInJsonReporter(reporter.VerificationReporter):
    """Store summary of verification(s) in JSON format"""

    # ISO 8601
    TIME_FORMAT = "%Y-%m-%dT%H:%M:%S%z"

    @classmethod
    def validate(cls, output_destination):
        # we do not have any restrictions for destination, so nothing to
        # check
        pass

    def generate(self):
        report = {}

        for v in self.verifications:
            report[v.uuid] = {
                "started_at": v.created_at.strftime(self.TIME_FORMAT),
                "finished_at": v.updated_at.strftime(self.TIME_FORMAT),
                "status": v.status,
                "run_args": v.run_args,
                "tests_count": v.tests_count,
                "tests_duration": v.tests_duration,
                "skipped": v.skipped,
                "success": v.success,
                "expected_failures": v.expected_failures,
                "unexpected_success": v.unexpected_success,
                "failures": v.failures,
                # v.tests includes all information about launched tests,
                # but for simplification of this fake reporters, let's
                # save just names
                "launched_tests": [test["name"]
                                   for test in v.tests.values()]
            }

        raw_report = json.dumps(report, indent=4)

        if self.output_destination:
            # In case of output_destination existence report will be saved
            # to hard drive and there is nothing to print to stdout, so
            # "print" key is not used
            return {"files": {self.output_destination: raw_report},
                    "open": self.output_destination}
        else:
            # it is something that will be print at CLI layer.
            return {"print": raw_report}
```

HowTo add support for new tool

First of all, you should start from the reading of *Rally Plugins* page. After you learned basic things about Rally plugin mechanism, let's move to Verifier interface itself.

Spec

All verifiers plugins should inherit `rally.verification.manager.VerifierManager` and implement all abstract methods. Here you can find its interface:

class `rally.verification.manager.`**VerifierManager**(*verifier*)
Verifier base class.

This class provides an interface for operating specific tool.

configure(*extra_options=None*)
Configure a verifier.
Parameters **extra_options** – a dictionary with external verifier specific options for configuration.
Raises **NotImplementedError** – This feature is verifier-specific, so you should override this method in your plugin if it supports configuration

extend_configuration(*extra_options*)
Extend verifier configuration with new options.
Parameters **extra_options** – Options to be used for extending configuration
Raises **NotImplementedError** – This feature is verifier-specific, so you should override this method in your plugin if it supports configuration

get_configuration()
Get verifier configuration (e.g., the config file content).

install()
Clone and install a verifier.

install_extension(*source*, *version=None*, *extra_settings=None*)
Install a verifier extension.
Parameters
- **source** – Path or URL to the repo to clone verifier extension from
- **version** – Branch, tag or commit ID to checkout before verifier extension installation
- **extra_settings** – Extra installation settings for verifier extension

Raises **NotImplementedError** – This feature is verifier-specific, so you should override this method in your plugin if it supports extensions

is_configured()
Check whether a verifier is configured or not.

list_extensions()
List all verifier extensions.

list_tests(*pattern=''*)
List all verifier tests.
Parameters **pattern** – Filter tests by given pattern

override_configuration(*new_configuration*)
Override verifier configuration.

Parameters **new_configuration** – Content which should be used while overriding existing configuration

Raises **NotImplementedError** – This feature is verifier-specific, so you should override this method in your plugin if it supports configuration

run(*context*)

Run verifier tests.

Verification Component API expects that this method should return an object. There is no special class, you do it as you want, but it should have the following properties:

```
<object>.totals = {
  "tests_count": <total tests count>,
  "tests_duration": <total tests duration>,
  "failures": <total count of failed tests>,
  "skipped": <total count of skipped tests>,
  "success": <total count of successful tests>,
  "unexpected_success":
      <total count of unexpected successful tests>,
  "expected_failures": <total count of expected failed tests>
}

<object>.tests = {
  <test_id>: {
      "status": <test status>,
      "name": <test name>,
      "duration": <test duration>,
      "reason": <reason>,  # optional
      "traceback": <traceback>  # optional
  },
  ...
}
```

uninstall(*full=False*)

Uninstall a verifier.

Parameters **full** – If False (default behaviour), only deployment-specific data will be removed

uninstall_extension(*name*)

Uninstall a verifier extension.

Parameters **name** – Name of extension to uninstall

Raises **NotImplementedError** – This feature is verifier-specific, so you should override this method in your plugin if it supports extensions

validate_args(*args*)

Validate given arguments to be used for running verification.

Parameters **args** – A dict of arguments with values

Example of Fake Verifier Manager

FakeTool is a tool which doesn't require configuration and installation.

```
import random
import re

from rally.verification import manager
```

```
# Verification component expects that method "run" of verifier returns
# object. Class Result is a simple wrapper for two expected properties.
class Result(object):
    def __init__(self, totals, tests):
        self.totals = totals
        self.tests = tests


@manager.configure("fake-tool", default_repo="https://example.com")
class FakeTool(manager.VerifierManager):
    """Fake Tool \o/"""

    TESTS = ["fake_tool.tests.bar.FatalityTestCase.test_one",
             "fake_tool.tests.bar.FatalityTestCase.test_two",
             "fake_tool.tests.bar.FatalityTestCase.test_three",
             "fake_tool.tests.bar.FatalityTestCase.test_four",
             "fake_tool.tests.foo.MegaTestCase.test_one",
             "fake_tool.tests.foo.MegaTestCase.test_two",
             "fake_tool.tests.foo.MegaTestCase.test_three",
             "fake_tool.tests.foo.MegaTestCase.test_four"]

    # This fake verifier doesn't launch anything, just returns random
    #  results, so let's override parent methods to avoid redundant
    #  clonning repo, checking packages and so on.

    def install(self):
        pass

    def uninstall(self, full=False):
        pass

    # Each tool, which supports configuration, has the own mechanism
    # for that task. Writing unified method is impossible. That is why
    # `VerificationManager` implements the case when the tool doesn't
    # need (doesn't support) configuration at all. Such behaviour is
    # ideal for FakeTool, since we do not need to change anything :)

    # Let's implement method `run` to return random data.
    def run(self, context):
        totals = {"tests_count": len(self.TESTS),
                  "tests_duration": 0,
                  "failures": 0,
                  "skipped": 0,
                  "success": 0,
                  "unexpected_success": 0,
                  "expected_failures": 0}
        tests = {}
        for name in self.TESTS:
            duration = random.randint(0, 10000)/100.
            totals["tests_duration"] += duration
            test = {"name": name,
                    "status": random.choice(["success", "fail"]),
                    "duration": "%s" % duration}
            if test["status"] == "fail":
                test["traceback"] = "Ooooppps"
                totals["failures"] += 1
            else:
```

```
                totals["success"] += 1
            tests[name] = test
        return Result(totals, tests=tests)

    def list_tests(self, pattern=""):
        return [name for name in self.TESTS if re.match(pattern, name)]
```

HowTo migrate from Verification component 0.7.0 to 0.8.0

Note: This document describes migration process from 0.7.0 to 0.8.0 Rally version. You can apply this instruction for migration to later versions, but check all references and release notes before trying to do it.

Verification Component was introduced long time ago even before the first Rally release. It started as a small helper thing but became a big powerful tool. Since it was not designed to all features that were implemented there later, it contained a lot of workarounds and hacks.

New Verification Component, which we are happy to introduce, should fix all architecture issues and improve user-experience. Unfortunately, fixing all those obsolete architecture decisions could not be done in a backward-compatible way, or it would produce much more workarounds. That is why we decided to redesign the whole component in a clear way - remove old code and write a new one from scratch.

Migration to New Verification Component should be simple and do not take too much time. You can find description of made changes below.

Reports

We completely reworked verification reports and merged comparison to main report. Now you can build one report for multiple number of verifications.

For more details follow *Verification reports*

Verification statuses

Old Status	New Status	Description
init	init	Initial state. It appears instantly after calling `rally verify start` command before the actual run of verifier's tool.
running		It was used right after checking status of verifier. It is redundant in terms of new design.
verifying	running	Identifies the process of tool execution.
finished	finished	Previously, "finished" state was used for an identification of just finished verification. By "finished" meant that verification has any test result. Now it means that verification was executed and doesn't have failures, unexpected success or any kind of errors.
	failed	Old purpose is an identification of "errors", situations when results are empty. The right use is an identification of finished verification with tests in "failed" and "uxsuccess" (unexpected success) statuses.
failed	crashed	Something went wrong while launching verification.

The latest information about verification statuses you can find at *Verification statuses*.

Command Line Interface

You can find the latest information about Verification Component CLI here - *Command Line Interface*.

Installing verifier

Command for Rally 0.7.0 - rally verify install

```
$ rally verify install --deployment <uuid> --source <url> --version <vers> \
  --system-wide
```

Command since Rally 0.8.0:

```
$ rally verify create-verifier --type "tempest" --source <url> \
  --version <version> --system-wide --name <name>
```

Here you can find several important improvements:

1. Rally team introduced new entity - *Verifiers*. Verifier stores all information about installed tool (i.e., source, version, system-wide) in a database. You do not need to transmit the same arguments into all `rally verify` commands as it was previously with `--system-wide` flag.
2. You can use particular verifier for multiple deployments. `--deployment` flag moved to `rally verify start` command. Also, you can run it simultaneously (checking in parallel different sets, different cloud, etc)

3. Verification Component can use not only Tempest for verifying system. Check *Known verifier types* for full list of supported tools.
4. You can have unlimited number of verifiers.

Re-install verifier aka update

Command for Rally 0.7.0 - rally verify reinstall

```
$ rally verify reinstall --deployment <uuid> --source <url> --version <vers> \
  --system-wide
```

Command since Rally 0.8.0:

```
$ rally verify update-verifier --id <id> --source <url> --version <vers> \
  --system-wide --no-system-wide --update-venv
```

Changes:

1. `rally verify update-verifier` doesn't require deployment id
2. You can switch between usage of system-wide installation and virtual environment.
3. You can update just virtual environment without cloning verifier code again

Uninstall

Command for Rally 0.7.0 - rally verify uninstall

```
$ rally verify uninstall --deployment <uuid>
```

Command since Rally 0.8.0:

```
$ rally verify delete-verifier --id <id> --deployment-id <id> --force
```

Changes:

1. As it was mentioned before, Verifier doesn't have an alignment to any particular deployment, so deployment argument is optional now. If –deployment-id argument is specified only deployment specific data will be removed (i.e, configurations).
2. New –force flag for removing all verifications results for that verifier.

Installation extensions

Command for Rally 0.7.0 - rally verify installplugin

```
$ rally verify installplugin --deployment <uuid> --source <url> \
  --version <vers> --system-wide
```

Command since Rally 0.8.0:

```
$ rally verify add-verifier-ext --id <id> --source <url> --version <vers> \
  --extra-settings <data>
```

Changes:

1. –system-wide flag is removed. Rally checks the verifier information to identify where to install the extension - in a system-side way or use virtual environment.
2. New –extra-settings flag. In case of Tempest, it is redundant, but for other verifiers allows to transmit some extra installation settings for verifier extension.

Uninstall extensions

Command for Rally 0.7.0 - rally verify uninstallplugin

```
$ rally verify uninstallplugin --deployment <uuid> --repo-name <repo_name> \
  --system-wide
```

Command since Rally 0.8.0:

```
$ rally verify delete-verifier-ext --id <id> --name <name>
```

Changes:

1. It is one more place where you do not need to pass –system-wide flag anymore.
2. –deployment flag is gone.
3. –repo-name is renamed to just –name.

List extensions

Command for Rally 0.7.0 - rally verify listplugins

```
$ rally verify listplugins --deployment <uuid> --system-wide
```

Command since Rally 0.8.0:

```
$ rally verify list-verifier-exts --id <id>
```

Changes:

1. No need to specify –system-wide flag.
2. –deployment flag is gone.

Discover available tests

Command for Rally 0.7.0 - rally verify discover

```
$ rally verify discover --deployment <uuid> --system-wide --pattern <pattern>
```

Command since Rally 0.8.0:

```
$ rally verify list-verifier-tests --id <id> --pattern <pattern>
```

Changes:

1. No need to specify –system-wide flag.
2. –deployment flag is gone.

Configuring

Commands for Rally 0.7.0:

- The command for generating configs rally verify genconfig

```
$ rally verify genconfig --deployment <uuid> --tempest-config <path> \
  --add-options <path> --override
```

Command since Rally 0.8.0:

```
$ rally verify configure-verifier --id <id> --deployment-id <uuid> \
  --extend <path/json/yaml> --override <path> --reconfigure --show
```

Changes:

1. The argument `--override` replaces old `--tempest-config` name. First of all, argument name "override" is a unified word without alignment to any tool. Also, it describes in the best way the meaning of the action: use client specified configuration file.
2. The argument `--extend` replaces old `--add-options`. It accepts a path to config in INI format or JSON/YAML string. In future, it will be extended with the ability to specify a path to JSON/YAML file.
3. The argument `--reconfigure` replaces old `--override`. It means that existing file will be ignored and new one will be used/created.

Show config

Command for Rally 0.7.0 - rally verify showconfig

```
$ rally verify showconfig --deployment <uuid>
```

Command since Rally 0.8.0:

```
$ rally verify configure-verifier --id <id> --deployment-id <uuid> --show
```

Changes:

> We do not have a separate command for that task. `rally verify configure-verifier --show` shows an existing configuration (if it exists) if `--reconfigure` argument is not specified.

Running verification

Command for Rally 0.7.0 - rally verify start

```
$ rally verify start --deployment <uuid> --set <set_name> --regex <regex> \
  --load-list <path> --tests-file <path> --skip-list <path> \
  --tempest-config <path> --xfail-list <path> --system-wide \
  --concurrency <N> --failing --no-use
```

Command since Rally 0.8.0:

```
$ rally verify start --id <id> --deployment-id <uuid> --pattern <pattern> \
  --load-list <path> --skip-list <path> --xfail-list <path> \
  --concurrency <N> --no-use --detailed
```

Changes:

1. You need to pass verifier id
2. Arguments `--set` and `--regex` are merged in the new model to single `--pattern` argument. Name of tests set should be specified like `--pattern set=<set_name>`. It was done to provide a way for each verifier to support custom arguments.
3. The argument `--tests-file` was deprecated in Rally 0.6.0 and we are ready to remove it.
4. Arguments `--skip-list` and `--xfail-list` accept path to file in JSON/YAML format. Content should be a dictionary, where keys are tests names (full name with id and tags) and values are reasons.
5. The argument `--tempest-config` is gone. Use `rally verify configure-verifier --id <id> --deployment-id <uuid> --override <path>` instead.
6. The argument `--system-wide` is gone like in most of other commands.
7. In case of specified `--detailed` arguments, traces of failed tests will be displayed (default behaviour in old verification design)

Show verification result

Commands for Rally 0.7.0:

- The command for showing results of verification rally verify show

```
$ rally verify show --uuid <uuid> --sort-by <query> --detailed
```

- Separate command which calls `rally verify show` with hardcoded `--detailed` flag rally verify detailed

```
$ rally verify detailed --uuid <uuid> --sort-by <query>
```

Command since Rally 0.8.0:

```
$ rally verify show --uuid <uuid> --sort-by <query> --detailed
```

Changes:

1. Redundant `rally verify detailed` command is removed
2. Sorting tests via `--sort-by` argument is extended to name/duration/status

Listing all verifications

Command for Rally 0.7.0 - rally verify list

```
$ rally verify list
```

Command since Rally 0.8.0:

```
$ rally verify list --id <id> --deployment-id <id> --status <status>
```

Changes:

You can filter verifications by verifiers, by deployments and results statuses.

Importing results

Command for Rally 0.7.0 - rally verify import

```
$ rally verify import --deployment <uuid> --set <set_name> --file <path> --no-use
```

Command since Rally 0.8.0:

```
$ rally verify import --id <id> --deployment-id <uuid> --file <path> \
  --run-args <run_args> --no-use
```

Changes:

1. You need to specify verifier to import results for.
2. The argument `--set` is merged into unified `--run-args`.

Building reports

Commands for Rally 0.7.0:

- The command for building HTML/JSON reports of verification rally verify results

```
$ rally verify results --uuid <uuid> --html --json --output-file <path>
```

- The command for comparison two verifications rally verify compare

```
$ rally verify compare --uuid-1 <uuid_1> --uuid-2 <uuid_2> --csv --html \
  --json --output-file <output_file> --threshold <threshold>
```

Command since Rally 0.8.0:

```
$ rally verify report --uuid <uuid> --type <type> --to <destination> --open
```

Changes:

1. Building reports becomes pluggable. You can extend reporters types. See *Verification reports* for more details.
2. The argument `--type` expects type of report (HTML/JSON). There are no more separate arguments for each report type.

   Hint: You can list all supported types, executing `rally plugin list --plugin-base VerificationReporter` command.

   ---

3. Reports are not aligned to only local types, so the argument `--to` replaces `--output-file`. In case of HTML/JSON reports, it can include a path to the local file like it was previously or URL to some external system with credentials like `https://username:password@example.com:777`.
4. The comparison is embedded into main reports and it is not limited by two verifications results. There are no reasons for the separate command for that task.

The End

Have nice verifications!

Historical background

Tempest, OpenStack's official test suite, is a powerful tool for running a set of functional tests against an OpenStack cluster. Tempest automatically runs against every patch in every project of OpenStack, which lets us avoid merging changes that break functionality.

Unfortunately, it has limited opportunities to be used, to process its results, etc. That is why we started Verification Component initiative a long time ago (see a blog post for more details, but be careful as all user interface is changed completely since that time).

What is Verification Component and why do you need it?

The primary goal of Rally Product is to provide a simple way to do complex things. As for functional testing, Verification Component includes interfaces for:

- **Managing things**. Create an isolated virtual environment and install verification tool there? Yes, we can do it! Clone tool from Git repositories? Sure! Store several versions of one tool (you know, sometimes they are incompatible, with different required packages and so on)? Of course! In general, Verification Component allows to install, upgrade, reinstall, configure your tool. You should not care about zillion options anymore Rally will discover them via cloud UX and make the configuration file for you automatically.
- **Launching verifiers**. Launchers of specific tools don't always contain all required features, Rally team tries to fix this omission. Verification Component supports some of them like expected failures, a list of tests to skip, a list of tests to launch, re-running previous verification or just failed tests from it and so on. Btw, all verification runs arguments are stored in the database.
- **Processing results**. Rally DataBase stores all verifications and you can obtain unified (across different verifiers) results at any time. You can find a verification run summary there, run arguments which were used, error messages and etc. Comparison mechanism for several verifications is available too. Verification reports can be generated in several formats: HTML, JSON, JUnit-XML (see *Verification reports* for more details). Also, reports mechanism is expendable and you can write your own plugin for whatever system you want.

Rally Plugins

Rally has a plugin oriented architecture - in other words Rally team is trying to make all places of code pluggable. Such architecture leads to the big amount of plugins. *Plugins Reference* contains a full list of all official Rally plugins with detailed descriptions.

Plugins Reference

Deployment

Engines

DevstackEngine [Engine]

Deploy Devstack cloud.

Sample configuration:

```
{
    "type": "DevstackEngine",
    "devstack_repo": "https://example.com/devstack/",
    "local_conf": {
        "ADMIN_PASSWORD": "secret"
    },
    "provider": {
        "type": "ExistingServers",
        "credentials": [{"user": "root", "host": "10.2.0.8"}]
    }
}
```

Namespace: default

Parameters:

- *local_conf (dict)* [*ref*]
- *localrc (dict)* [*ref*]
- *devstack_branch (str)* [*ref*]
- *provider (dict)* [*ref*]
- *type (str)* [*ref*]
- *devstack_repo (str)* [*ref*]

Module: rally.deployment.engines.devstack

ExistingCloud [Engine]

Just use an existing OpenStack deployment without deploying anything.

To use ExistingCloud, you should put credential information to the config:

```
{
    "type": "ExistingCloud",
    "auth_url": "http://localhost:5000/v2.0/",
    "region_name": "RegionOne",
    "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "password",
        "tenant_name": "demo"
    },
    "https_insecure": False,
    "https_cacert": "",
}
```

Or, using keystone v3 API endpoint:

```
{
    "type": "ExistingCloud",
    "auth_url": "http://localhost:5000/v3/",
    "region_name": "RegionOne",
    "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "admin",
        "user_domain_name": "admin",
        "project_name": "admin",
        "project_domain_name": "admin",
    },
    "https_insecure": False,
    "https_cacert": "",
}
```

To specify extra options use can use special "extra" parameter:

```
{
    "type": "ExistingCloud",
    "auth_url": "http://localhost:5000/v2.0/",
    "region_name": "RegionOne",
    "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "password",
        "tenant_name": "demo"
    },
    "https_insecure": False,
    "https_cacert": "",
    "extra": {"some_var": "some_value"}
}
```

Namespace: default

Parameters:

- *https_insecure (bool)* [*ref*]
- *endpoint* [*ref*]
- *auth_url (str)* [*ref*]
- *region_name (str)* [*ref*]
- *endpoint_type* [*ref*]

  Set of expected values: 'admin', 'internal', 'public', 'None'.
- *extra (dict)* [*ref*]
- *admin* [*ref*]

  N/a
- *https_cacert (str)* [*ref*]
- *type (str)* [*ref*]
- *users (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - `{'$ref': '#/definitions/user'}`

Module: rally.deployment.engines.existing

LxcEngine [Engine]

Deploy with other engines in lxc containers.

Sample configuration:

```
{
    "type": "LxcEngine",
    "provider": {
        "type": "DummyProvider",
        "credentials": [{"user": "root", "host": "example.net"}]
    },
    "distribution": "ubuntu",
    "release": "raring",
    "tunnel_to": ["10.10.10.10", "10.10.10.11"],
    "start_lxc_network": "10.1.1.0/24",
    "container_name_prefix": "devstack-node",
    "containers_per_host": 16,
    "start_script": "~/start.sh",
    "engine": { ... }
}
```

Namespace: default

Parameters:

- *start_lxc_network (str)* [*ref*]

  Should follow next pattern: ^(d+.){3}d+/d+$.
- *containers_per_host (int)* [*ref*]
- *tunnel_to (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: str.
- *release (str)* [*ref*]
- *distribution (str)* [*ref*]
- *container_name (str)* [*ref*]
- *type (str)* [*ref*]
- *provider (dict)* [*ref*]

Module: rally.deployment.engines.lxc

MultihostEngine [Engine]

Deploy multihost cloud with existing engines.

Sample configuration:

```
{
    "type": "MultihostEngine",
    "controller": {
        "type": "DevstackEngine",
        "provider": {
            "type": "DummyProvider"
        }
    },
    "nodes": [
        {"type": "Engine1", "config": "Config1"},
        {"type": "Engine2", "config": "Config2"},
        {"type": "Engine3", "config": "Config3"},
    ]
}
```

If {controller_ip} is specified in configuration values, it will be replaced with controller address taken from credential returned by controller engine:

```
...
"nodes": [
    {
        "type": "DevstackEngine",
        "local_conf": {
            "GLANCE_HOSTPORT": "{controller_ip}:9292",
...
```

Namespace: default

Module: rally.deployment.engines.multihost

Provider Factories

CobblerProvider [Provider Factory]

Creates servers via PXE boot from given cobbler selector.

Cobbler selector may contain a combination of fields to select a number of system. It's user responsibility to provide selector which selects something. Since cobbler stores servers password encrypted the user needs to specify it configuration. All servers selected must have the same password.

Sample configuration:

```
{
    "type": "CobblerProvider",
    "host": "172.29.74.8",
    "user": "cobbler",
    "password": "cobbler",
    "system_password": "password"
    "selector": {"profile": "cobbler_profile_name", "owners": "user1"}
}
```

Namespace: default

Parameters:

- *host (str)* [*ref*]
- *password (str)* [*ref*]
- *selector (dict)* [*ref*]
- *user (str)* [*ref*]
- *system_password (str)* [*ref*]

Module: rally.deployment.serverprovider.providers.cobbler

ExistingServers [Provider Factory]

Just return endpoints from its own configuration.

Sample configuration:

```
{
    "type": "ExistingServers",
    "credentials": [{"user": "root", "host": "localhost"}]
}
```

Namespace: default

Parameters:

- *credentials (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

    ```
    {
       "required": [
           "host",
           "user"
       ],
       "type": "object",
       "properties": {
           "host": {
               "type": "string"
           },
    ```

```
            "password": {
                "type": "string"
            },
            "port": {
                "type": "integer"
            },
            "key": {
                "type": "string"
            },
            "user": {
                "type": "string"
            }
        }
}
```

- *type (str)* [*ref*]

Module: rally.deployment.serverprovider.providers.existing

LxcProvider [Provider Factory]

Provide lxc container(s) on given host.

Sample configuration:

```
{
    "type": "LxcProvider",
    "distribution": "ubuntu",
    "start_lxc_network": "10.1.1.0/24",
    "containers_per_host": 32,
    "tunnel_to": ["10.10.10.10"],
    "forward_ssh": false,
    "container_name_prefix": "rally-multinode-02",
    "host_provider": {
        "type": "ExistingServers",
        "credentials": [{"user": "root", "host": "host.net"}]
    }
}
```

Namespace: default

Parameters:

- *start_lxc_network (str)* [*ref*]

  Should follow next pattern: ^(d+.){3}d+/d+$.

- *containers_per_host (int)* [*ref*]
- *forward_ssh (bool)* [*ref*]
- *release (str)* [*ref*]
- *distribution (str)* [*ref*]
- *tunnel_to (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: str.

- *type (str)* [*ref*]
- *container_name_prefix (str)* [*ref*]
- *host_provider (dict)* [*ref*]

Module: rally.deployment.serverprovider.providers.lxc

OpenStackProvider [Provider Factory]

Provide VMs using an existing OpenStack cloud.

Sample configuration:

```
{
    "type": "OpenStackProvider",
    "amount": 42,
    "user": "admin",
    "tenant": "admin",
    "password": "secret",
    "auth_url": "http://example.com/",
    "flavor_id": 2,
    "image": {
        "checksum": "75846dd06e9fcfd2b184aba7fa2b2a8d",
        "url": "http://example.com/disk1.img",
        "name": "Ubuntu Precise(added by rally)",
        "format": "qcow2",
        "userdata": "disable_root: false"
    },
    "secgroup_name": "Rally"
}
```

Namespace: default

Parameters:

- *deployment_name (str)* [*ref*]
- *image (dict)* [*ref*]
- *auth_url (str)* [*ref*]
- *user (str)* [*ref*]
- *password (str)* [*ref*]
- *tenant (str)* [*ref*]
- *nics (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

    ```
    {
        "additionalProperties": false,
        "required": [
            "net-id"
        ],
        "type": "object",
        "properties": {
            "net-id": {
    ```

```
            "type": "string"
        }
    }
}
```

- *region (str)* [*ref*]
- *wait_for_cloud_init (bool)* [*ref*]
- *amount (int)* [*ref*]
- *flavor_id (str)* [*ref*]
- *secgroup_name (str)* [*ref*]
- *type (str)* [*ref*]
- *config_drive (bool)* [*ref*]

Module: rally.deployment.serverprovider.providers.openstack

VirshProvider [Provider Factory]

Create VMs from prebuilt templates.

Sample configuration:

```
{
    "type": "VirshProvider",
    "connection": "alex@performance-01",
    "template_name": "stack-01-devstack-template",
    "template_user": "ubuntu",
    "template_password": "password"
}
```

where :

- connection - ssh connection to vms host
- template_name - vm image template
- template_user - vm user to launch devstack
- template_password - vm password to launch devstack

Namespace: default

Parameters:

- *template_name (str)* [*ref*]
- *connection (str)* [*ref*]

  Should follow next pattern: ^.+@.+$.
- *template_password (str)* [*ref*]
- *type (str)* [*ref*]
- *template_user (str)* [*ref*]

Module: rally.deployment.serverprovider.providers.virsh

Task Component

Charts

Lines [Chart]

Display results as generic chart with lines.

This plugin processes additive data and displays it in HTML report as linear chart with X axis bound to iteration number. Complete output data is displayed as linear chart as well, without any processing.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    additive={"title": "Additive data as stacked area",
              "description": "Iterations trend for foo and bar",
              "chart_plugin": "Lines",
              "data": [["foo", 12], ["bar", 34]]},
    complete={"title": "Complete data as stacked area",
              "description": "Data is shown as stacked area, as-is",
              "chart_plugin": "Lines",
              "data": [["foo", [[0, 5], [1, 42], [2, 15], [3, 7]]],
                       ["bar", [[0, 2], [1, 1.3], [2, 5], [3, 9]]]],
              "label": "Y-axis label text",
              "axis_label": "X-axis label text"})
```

Namespace: default

Module: rally.task.processing.charts

Pie [Chart]

Display results as pie, calculate average values for additive data.

This plugin processes additive data and calculate average values. Both additive and complete data are displayed in HTML report as pie chart.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    additive={"title": "Additive output",
              "description": ("Pie with average data "
                              "from all iterations values"),
              "chart_plugin": "Pie",
              "data": [["foo", 12], ["bar", 34], ["spam", 56]]},
    complete={"title": "Complete output",
              "description": "Displayed as a pie, as-is",
              "chart_plugin": "Pie",
              "data": [["foo", 12], ["bar", 34], ["spam", 56]]})
```

Namespace: default

Module: rally.task.processing.charts

StackedArea [Chart]

Display results as stacked area.

This plugin processes additive data and displays it in HTML report as stacked area with X axis bound to iteration number. Complete output data is displayed as stacked area as well, without any processing.

Keys "description", "label" and "axis_label" are optional.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    additive={"title": "Additive data as stacked area",
              "description": "Iterations trend for foo and bar",
              "chart_plugin": "StackedArea",
              "data": [["foo", 12], ["bar", 34]]},
    complete={"title": "Complete data as stacked area",
              "description": "Data is shown as stacked area, as-is",
              "chart_plugin": "StackedArea",
              "data": [["foo", [[0, 5], [1, 42], [2, 15], [3, 7]]],
                       ["bar", [[0, 2], [1, 1.3], [2, 5], [3, 9]]]],
              "label": "Y-axis label text",
              "axis_label": "X-axis label text"})
```

Namespace: default

Module: rally.task.processing.charts

StatsTable [Chart]

Calculate statistics for additive data and display it as table.

This plugin processes additive data and compose statistics that is displayed as table in HTML report.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    additive={"title": "Statistics",
              "description": ("Table with statistics generated "
                              "from all iterations values"),
              "chart_plugin": "StatsTable",
              "data": [["foo stat", 12], ["bar", 34], ["spam", 56]]})
```

Namespace: default

Module: rally.task.processing.charts

Table [Chart]

Display complete output as table, can not be used for additive data.

Use this plugin for complete output data to display it in HTML report as table. This plugin can not be used for additive data because it does not contain any processing logic.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    complete={"title": "Arbitrary Table",
              "description": "Just show columns and rows as-is",
              "chart_plugin": "Table",
              "data": {"cols": ["foo", "bar", "spam"],
                       "rows": [["a row", 1, 2], ["b row", 3, 4],
                                ["c row", 5, 6]]}})
```

Namespace: default

Module: rally.task.processing.charts

TextArea [Chart]

Arbitrary text

This plugin processes complete data and displays of output in HTML report.

Examples of using this plugin in Scenario, for saving output data:

```
self.add_output(
    complete={"title": "Script Inline",
              "chart_plugin": "TextArea",
              "data": ["first output", "second output",
                       "third output"]]})
```

Namespace: default

Module: rally.task.processing.charts

Contexts

allow_ssh [Context]

Sets up security groups for all users to access VM via SSH.

Namespace: default

Parameters:

- *null* [*ref*]

Module: rally.plugins.openstack.context.network.allow_ssh

api_versions [Context]

Context for specifying OpenStack clients versions and service types.

Some OpenStack services support several API versions. To recognize the endpoints of each version, separate service types are provided in Keystone service catalog.

Rally has the map of default service names - service types. But since service type is an entity, which can be configured manually by admin(via keystone api) without relation to service name, such map can be insufficient.

Also, Keystone service catalog does not provide a map types to name (this statement is true for keystone < 3.3).

This context was designed for not-default service types and not-default API versions usage.

An example of specifying API version:

```
# In this example we will launch NovaKeypair.create_and_list_keypairs
# scenario on 2.2 api version.
{
    "NovaKeypair.create_and_list_keypairs": [
        {
```

```
            "args": {
                "key_type": "x509"
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                },
                "api_versions": {
                    "nova": {
                        "version": 2.2
                    }
                }
            }
        }
    ]
}
```

An example of specifying API version along with service type:

```
# In this example we will launch CinderVolumes.create_and_attach_volume
# scenario on Cinder V2
{
    "CinderVolumes.create_and_attach_volume": [
        {
            "args": {
                "size": 10,
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "flavor": {
                    "name": "m1.tiny"
                },
                "create_volume_params": {
                    "availability_zone": "nova"
                }
            },
            "runner": {
                "type": "constant",
                "times": 5,
                "concurrency": 1
            },
            "context": {
                "users": {
                    "tenants": 2,
                    "users_per_tenant": 2
                },
                "api_versions": {
                    "cinder": {
                        "version": 2,
                        "service_type": "volumev2"
                    }
                }
```

```
            }
        }
    ]
}
```

Also, it possible to use service name as an identifier of service endpoint, but an admin user is required (Keystone can return map of service names - types, but such API is permitted only for admin). An example:

```
# Similar to the previous example, but `service_name` argument is used
# instead of `service_type`
{
    "CinderVolumes.create_and_attach_volume": [
        {
            "args": {
                "size": 10,
                "image": {
                    "name": "^cirros.*-disk$"
                },
                "flavor": {
                    "name": "m1.tiny"
                },
                "create_volume_params": {
                    "availability_zone": "nova"
                }
            },
            "runner": {
                "type": "constant",
                "times": 5,
                "concurrency": 1
            },
            "context": {
                "users": {
                    "tenants": 2,
                    "users_per_tenant": 2
                },
                "api_versions": {
                    "cinder": {
                        "version": 2,
                        "service_name": "cinderv2"
                    }
                }
            }
        }
    ]
}
```

Namespace: default

Parameters:

Dictionary is expected. Keys should follow pattern(s) described bellow.

- *^[a-z]+$ (str)* [*ref*]

Module: rally.plugins.openstack.context.api_versions

audit_templates [Context]

Context class for adding temporary audit template for benchmarks.

Namespace: default

Parameters:

- *audit_templates_per_admin (int)* [*ref*]

  Min value: 1.

- *params (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "type": "object",
    "properties": {
        "goal": {
            "type": "object",
            "properties": {
                "name": {
                    "type": "string"
                }
            }
        },
        "strategy": {
            "type": "object",
            "properties": {
                "name": {
                    "type": "string"
                }
            }
        }
    }
}
```

- *fill_strategy* [*ref*]

  Set of expected values: 'round_robin', 'random', 'None'.

Module: rally.plugins.openstack.context.watcher.audit_templates

ceilometer [Context]

Context for creating samples and collecting resources for benchmarks.

Namespace: default

Parameters:

- *counter_name (str)* [*ref*]
- *metadata_list (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "type": "object",
    "properties": {
        "status": {
            "type": "string"
        },
        "deleted": {
            "type": "string"
        },
        "created_at": {
            "type": "string"
        },
        "name": {
            "type": "string"
        }
    }
}
```

- *samples_per_resource (int)* [*ref*]

  Min value: 1.

- *counter_unit (str)* [*ref*]
- *resources_per_tenant (int)* [*ref*]

  Min value: 1.

- *counter_volume (float)* [*ref*]

  Min value: 0.

- *batches_allow_lose (int)* [*ref*]

  Min value: 0.

- *timestamp_interval (int)* [*ref*]

  Min value: 1.

- *batch_size (int)* [*ref*]

  Min value: 1.

- *counter_type (str)* [*ref*]

Module: rally.plugins.openstack.context.ceilometer.samples

cluster_templates [Context]

Context class for generating temporary cluster model for benchmarks.

Namespace: default

Parameters:

- *docker_storage_driver (str)* [*ref*]
- *http_proxy (str)* [*ref*]
- *docker_volume_size (int)* [*ref*]
- *https_proxy (str)* [*ref*]

- *no_proxy (str)* [*ref*]
- *external_network_id (str)* [*ref*]
- *labels (str)* [*ref*]
- *dns_nameserver (str)* [*ref*]
- *server_type (str)* [*ref*]
- *network_driver (str)* [*ref*]
- *fixed_network (str)* [*ref*]
- *image_id (str)* [*ref*]
- *tls_disabled (bool)* [*ref*]
- *registry_enabled (bool)* [*ref*]
- *coe (str)* [*ref*]
- *flavor_id (str)* [*ref*]
- *volume_driver (str)* [*ref*]
- *master_lb_enabled (bool)* [*ref*]
- *public (bool)* [*ref*]
- *fixed_subnet (str)* [*ref*]
- *master_flavor_id (str)* [*ref*]

Module: rally.plugins.openstack.context.magnum.cluster_templates

clusters [Context]

Context class for generating temporary cluster for benchmarks.

Namespace: default

Parameters:

- *node_count (int)* [*ref*]

  Min value: 1.

- *cluster_template_uuid (str)* [*ref*]

Module: rally.plugins.openstack.context.magnum.clusters

dummy_context [Context]

Dummy context.

Namespace: default

Parameters:

- *fail_cleanup (bool)* [*ref*]
- *fail_setup (bool)* [*ref*]

Module: rally.plugins.common.context.dummy

ec2_servers [Context]

Context class for adding temporary servers for benchmarks.

Servers are added for each tenant.

Namespace: default

Parameters:

- *servers_per_tenant (int)* [*ref*]

  Min value: 1.

- *image (dict)* [*ref*]
- *flavor (dict)* [*ref*]

Module: rally.plugins.openstack.context.ec2.servers

existing_network [Context]

This context supports using existing networks in Rally.

This context should be used on a deployment with existing users.

Namespace: default

Parameters:

Module: rally.plugins.openstack.context.network.existing_network

flavors [Context]

Context creates a list of flavors.

Namespace: default

Parameters:

- *list* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "additionalProperties": false,
    "required": [
        "name",
        "ram"
    ],
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "ram": {
            "minimum": 1,
            "type": "integer"
        },
```

```
        "ephemeral": {
            "minimum": 0,
            "type": "integer"
        },
        "vcpus": {
            "minimum": 1,
            "type": "integer"
        },
        "extra_specs": {
            "additionalProperties": {
                "type": "string"
            },
            "type": "object"
        },
        "swap": {
            "minimum": 0,
            "type": "integer"
        },
        "disk": {
            "minimum": 0,
            "type": "integer"
        }
    }
}
```

Module: rally.plugins.openstack.context.nova.flavors

fuel_environments [Context]

Context for generating Fuel environments.

Namespace: default

Parameters:

- *release_id (int)* [*ref*]
- *network_provider (str)* [*ref*]
- *environments (int)* [*ref*]

  Min value: 1.
- *deployment_mode (str)* [*ref*]
- *net_segment_type (str)* [*ref*]
- *resource_management_workers (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.fuel

heat_dataplane [Context]

Context class for create stack by given template.

This context will create stacks by given template for each tenant and add details to context. Following details will be added:

id of stack; template file contents; files dictionary; stack parameters;

Heat template should define a "gate" node which will interact with Rally by ssh and workload nodes by any protocol. To make this possible heat template should accept the following parameters:

network_id: id of public network router_id: id of external router to connect "gate" node key_name: name of nova ssh keypair to use for "gate" node

Namespace: default

Parameters:

- *files (dict)* [*ref*]
- *context_parameters (dict)* [*ref*]
- *parameters (dict)* [*ref*]
- *template* [*ref*]
- *stacks_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.dataplane.heat

image_command_customizer [Context]

Context class for generating image customized by a command execution.

Run a command specified by configuration to prepare image.

Use this script e.g. to download and install something.

Namespace: default

Parameters:

- *username (str)* [*ref*]
- *floating_network (str)* [*ref*]
- *userdata (str)* [*ref*]
- *internal_network (str)* [*ref*]
- *workers (int)* [*ref*]

  Min value: 1.
- *port (int)* [*ref*]

  Min value: 1.

  Max value: 65535.
- *command* [*ref*]

  N/a
- *flavor (dict)* [*ref*]
- *password (str)* [*ref*]
- *image (dict)* [*ref*]

Module: rally.plugins.openstack.context.vm.image_command_customizer

images [Context]

Context class for adding images to each user for benchmarks.

Namespace: default

Parameters:

- *min_disk (int)* [*ref*]

  Min value: 0.

- *image_container (str)* [*ref*]
- *image_url (str)* [*ref*]
- *image_type* [*ref*]

  Set of expected values: 'qcow2', 'raw', 'vhd', 'vmdk', 'vdi', 'iso', 'aki', 'ari', 'ami'.

- *min_ram (int)* [*ref*]

  Min value: 0.

- *image_args (dict)* [*ref*]
- *image_name (str)* [*ref*]
- *images_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.glance.images

keypair [Context]

Namespace: default

Parameters:

Module: rally.plugins.openstack.context.nova.keypairs

lbaas [Context]

Namespace: default

Parameters:

- *pool (dict)* [*ref*]
- *lbaas_version (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.neutron.lbaas

manila_security_services [Context]

This context creates 'security services' for Manila project.

Namespace: default

Parameters:

- *security_services (list)* [*ref*]

  It is expected to be list of dicts with data for creation of security services.

  Elements of the list should follow format(s) described below:

  - **Type: dict. Data for creation of security services.** Example:

```
{'type': 'LDAP', 'dns_ip': 'foo_ip',
 'server': 'bar_ip', 'domain': 'quuz_domain',
 'user': 'ololo', 'password': 'fake_password'}
```

    Format:

```
{
    "additionalProperties": true,
    "required": [
        "type"
    ],
    "type": "object",
    "properties": {
        "type": {
            "enum": [
                "active_directory",
                "kerberos",
                "ldap"
            ]
        }
    }
}
```

Module: rally.plugins.openstack.context.manila.manila_security_services

manila_share_networks [Context]

This context creates share networks for Manila project.

Namespace: default

Parameters:

- *share_networks (dict)* [*ref*]
- *use_share_networks (bool)* [*ref*]

Module: rally.plugins.openstack.context.manila.manila_share_networks

manila_shares [Context]

This context creates shares for Manila project.

Namespace: default

Parameters:

- *shares_per_tenant (int)* [*ref*]

  Min value: 1.

- *share_proto (str)* [*ref*]

- *share_type (str)* [*ref*]
- *size (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.manila.manila_shares

monasca_metrics [Context]

Context for creating metrics for benchmarks.

Namespace: default

Parameters:

- *metrics_per_tenant (int)* [*ref*]

  Min value: 1.

- *value_meta (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "type": "object",
    "properties": {
        "value_meta_value": {
            "type": "string"
        },
        "value_meta_key": {
            "type": "string"
        }
    }
}
```

- *name (str)* [*ref*]
- *dimensions (dict)* [*ref*]

Module: rally.plugins.openstack.context.monasca.metrics

murano_environments [Context]

Context class for creating murano environments.

Namespace: default

Parameters:

- *environments_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.murano.murano_environments

murano_packages [Context]

Context class for uploading applications for murano.

Namespace: default

Parameters:

- *app_package (str)* [*ref*]

Module: rally.plugins.openstack.context.murano.murano_packages

network [Context]

Create networking resources.

This creates networks for all tenants, and optionally creates another resources like subnets and routers.

Namespace: default

Parameters:

- *network_create_args (dict)* [*ref*]
- *subnets_per_network (int)* [*ref*]

  Min value: 1.
- *start_cidr (str)* [*ref*]
- *dns_nameservers (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: str.
- *networks_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.network.networks

profiles [Context]

Context creates a temporary profile for Senlin test.

Namespace: default

Parameters:

- *version (str)* [*ref*]
- *type (str)* [*ref*]
- *properties (dict)* [*ref*]

Module: rally.plugins.openstack.context.senlin.profiles

quotas [Context]

Context class for updating benchmarks' tenants quotas.

Namespace: default

Parameters:

- *neutron (dict)* [*ref*]
- *cinder (dict)* [*ref*]
- *manila (dict)* [*ref*]
- *nova (dict)* [*ref*]
- *designate (dict)* [*ref*]

Module: rally.plugins.openstack.context.quotas.quotas

roles [Context]

Context class for assigning roles for users.

Namespace: default

Parameters:

- *list* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: str. The name of role to assign to user

Module: rally.plugins.openstack.context.keystone.roles

sahara_cluster [Context]

Context class for setting up the Cluster an EDP job.

Namespace: default

Parameters:

- *workers_count (int)* [*ref*]

  Min value: 1.
- *worker_flavor_id (str)* [*ref*]
- *use_autoconfig (bool)* [*ref*]
- *cluster_configs (dict)* [*ref*]
- *enable_proxy (bool)* [*ref*]
- *plugin_name (str)* [*ref*]
- *floating_ip_pool (str)* [*ref*]
- *volumes_size (int)* [*ref*]

  Min value: 1.
- *node_configs (dict)* [*ref*]

- *flavor_id (str)* [*ref*]
- *volumes_per_node (int)* [*ref*]

  Min value: 1.
- *enable_anti_affinity (bool)* [*ref*]
- *hadoop_version (str)* [*ref*]
- *auto_security_group (bool)* [*ref*]
- *security_groups (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  – Type: str.
- *master_flavor_id (str)* [*ref*]

Module: rally.plugins.openstack.context.sahara.sahara_cluster

sahara_image [Context]

Context class for adding and tagging Sahara images.

Namespace: default

Parameters:

- *username (str)* [*ref*]
- *image_uuid (str)* [*ref*]
- *hadoop_version (str)* [*ref*]
- *image_url (str)* [*ref*]
- *plugin_name (str)* [*ref*]

Module: rally.plugins.openstack.context.sahara.sahara_image

sahara_input_data_sources [Context]

Context class for setting up Input Data Sources for an EDP job.

Namespace: default

Parameters:

- *input_type* [*ref*]

  Set of expected values: 'swift', 'hdfs'.
- *swift_files (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  – Type: dict. Format:

```
{
    "additionalProperties": false,
    "required": [
        "name",
        "download_url"
```

```
    ],
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "download_url": {
            "type": "string"
        }
    }
}
```

- *input_url (str)* [*ref*]

Module: rally.plugins.openstack.context.sahara.sahara_input_data_sources

sahara_job_binaries [Context]

Context class for setting up Job Binaries for an EDP job.

Namespace: default

Parameters:

- *libs (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "additionalProperties": false,
    "required": [
        "name",
        "download_url"
    ],
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "download_url": {
            "type": "string"
        }
    }
}
```

- *mains (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: dict. Format:

```
{
    "additionalProperties": false,
    "required": [
        "name",
        "download_url"
    ],
```

```
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "download_url": {
            "type": "string"
        }
    }
}
```

Module: rally.plugins.openstack.context.sahara.sahara_job_binaries

sahara_output_data_sources [Context]

Context class for setting up Output Data Sources for an EDP job.

Namespace: default

Parameters:

- *output_type* [*ref*]

  Set of expected values: 'swift', 'hdfs'.

- *output_url_prefix (str)* [*ref*]

Module: rally.plugins.openstack.context.sahara.sahara_output_data_sources

servers [Context]

Context class for adding temporary servers for benchmarks.

Servers are added for each tenant.

Namespace: default

Parameters:

- *servers_per_tenant (int)* [*ref*]

  Number of servers to boot in each Tenant.

  Min value: 1.

- *image (dict)* [*ref*]

  Name of image to boot server(s) from.

- *auto_assign_nic (bool)* [*ref*]

  True if NICs should be assigned.

- *flavor (dict)* [*ref*]

  Name of flavor to boot server(s) with.

- *nics (list)* [*ref*]

  List of networks to attach to server.

  Elements of the list should follow format(s) described below:

```
- {'oneOf': [{'type': 'object', 'properties': {'net-id': {'type':
  'string'}}, 'description': 'Network ID in a format like OpenStack
  API expects to see.'}, {'type': 'string', 'description': 'Network
  ID.'}]}
```

Module: rally.plugins.openstack.context.nova.servers

stacks [Context]

Context class for create temporary stacks with resources.

Stack generator allows to generate arbitrary number of stacks for each tenant before test scenarios. In addition, it allows to define number of resources (namely OS::Heat::RandomString) that will be created inside each stack. After test execution the stacks will be automatically removed from heat.

Namespace: default

Parameters:

- *resources_per_stack (int)* [*ref*]

  Min value: 1.

- *stacks_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.heat.stacks

swift_objects [Context]

Namespace: default

Parameters:

- *object_size (int)* [*ref*]

  Min value: 1.

- *containers_per_tenant (int)* [*ref*]

  Min value: 1.

- *objects_per_container (int)* [*ref*]

  Min value: 1.

- *resource_management_workers (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.swift.objects

users [Context]

Context class for generating temporary users/tenants for benchmarks.

Namespace: openstack

Parameters:

- *user_domain (str)* [*ref*]

  ID of domain in which users will be created.

- *project_domain (str)* [*ref*]

  ID of domain in which projects will be created.

- *user_choice_method* [*ref*]

  The mode of balancing usage of users between scenario iterations. Set of expected values: 'random', 'round_robin'.

- *users_per_tenant (int)* [*ref*]

  The number of users to create per one tenant.

  Min value: 1.

- *tenants (int)* [*ref*]

  The number of tenants to create.

  Min value: 1.

- *resource_management_workers (int)* [*ref*]

  The number of concurrent threads to use for serving users context.

  Min value: 1.

Module: rally.plugins.openstack.context.keystone.users

volume_types [Context]

Context class for adding volumes types for benchmarks.

Namespace: default

Parameters:

- *list* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: str.

Module: rally.plugins.openstack.context.cinder.volume_types

volumes [Context]

Context class for adding volumes to each user for benchmarks.

Namespace: default

Parameters:

- *type* [*ref*]
- *volumes_per_tenant (int)* [*ref*]

  Min value: 1.

- *size (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.cinder.volumes

zones [Context]

Context to add *zones_per_tenant* zones for each tenant.

Namespace: default

Parameters:

- *zones_per_tenant (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.openstack.context.designate.zones

Exporters

file [Exporter]

Namespace: default

Module: rally.plugins.common.exporter.file_system

file-exporter [Exporter]

DEPRECATED.

Namespace: default

Module: rally.plugins.common.exporter.file_system

Hooks

fault_injection [Hook]

Performs fault injection using os-faults library.

Configuration: action - string that represents an action (more info in [1]) verify - whether to verify connection to cloud nodes or not

This plugin discovers extra config of ExistingCloud and looks for "cloud_config" field. If cloud_config is present then it will be used to connect to the cloud by os-faults.

Another option is to provide os-faults config file through OS_FAULTS_CONFIG env variable. Format of the config can be found in [1].

[1] http://os-faults.readthedocs.io/en/latest/usage.html

Namespace: default

Parameters:

- *action (str)* [*ref*]
- *verify (bool)* [*ref*]

Module: rally.plugins.openstack.hook.fault_injection

sys_call [Hook]

Performs system call.

Namespace: default

Parameters:

- *str* [*ref*]

  Command to execute.

Module: rally.plugins.common.hook.sys_call

SLAs

failure_rate [SLA]

Failure rate minimum and maximum in percents.

Namespace: default

Parameters:

- *max (float)* [*ref*]

  Min value: 0.0.

  Max value: 100.0.

- *min (float)* [*ref*]

  Min value: 0.0.

  Max value: 100.0.

Module: rally.plugins.common.sla.failure_rate

max_avg_duration [SLA]

Maximum average duration of one iteration in seconds.

Namespace: default

Parameters:

- *float* [*ref*]

  Min value: 0.0.

Module: rally.plugins.common.sla.max_average_duration

max_avg_duration_per_atomic [SLA]

Maximum average duration of one iterations atomic actions in seconds.

Namespace: default

Parameters:

Dictionary is expected. Keys should follow pattern(s) described bellow.

- . (str)* [*ref*]

  The name of atomic action.

Module: rally.plugins.common.sla.max_average_duration_per_atomic

max_seconds_per_iteration [SLA]

Maximum time for one iteration in seconds.

Namespace: default

Parameters:

- *float* [*ref*]

  Min value: 0.0.

Module: rally.plugins.common.sla.iteration_time

outliers [SLA]

Limit the number of outliers (iterations that take too much time).

The outliers are detected automatically using the computation of the mean and standard deviation (std) of the data.

Namespace: default

Parameters:

- *max (int)* [*ref*]

  Min value: 0.

- *min_iterations (int)* [*ref*]

  Min value: 3.

- *sigmas (float)* [*ref*]

  Min value: 0.0.

Module: rally.plugins.common.sla.outliers

performance_degradation [SLA]

Calculates performance degradation based on iteration time

This SLA plugin finds minimum and maximum duration of iterations completed without errors during Rally task execution. Assuming that minimum duration is 100%, it calculates performance degradation against maximum duration.

Namespace: default

Parameters:

- *max_degradation (float)* [*ref*]

  Min value: 0.0.

Module: rally.plugins.common.sla.performance_degradation

Scenarios

Authenticate.keystone [Scenario]

Check Keystone Client.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_ceilometer [Scenario]

Check Ceilometer Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_cinder [Scenario]

Check Cinder Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_glance [Scenario]

Check Glance Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated. In following we are checking for non-existent image.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_heat [Scenario]

Check Heat Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_monasca [Scenario]

Check Monasca Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_neutron [Scenario]

Check Neutron Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

Authenticate.validate_nova [Scenario]

Check Nova Client to ensure validation of token.

Creation of the client does not ensure validation of the token. We have to do some minimal operation to make sure token gets validated.

Namespace: openstack

Parameters:

- *repetitions* [*ref*]

  Number of times to validate

Module: rally.plugins.openstack.scenarios.authenticate.authenticate

CeilometerAlarms.create_alarm [Scenario]

Create an alarm.

This scenarios test POST /v2/alarms. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc that may be passed while creating an alarm.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.create_alarm_and_get_history [Scenario]

Create an alarm, get and set the state and get the alarm history.

This scenario makes following queries: GET /v2/alarms/{alarm_id}/history GET /v2/alarms/{alarm_id}/state PUT /v2/alarms/{alarm_id}/state

Initially alarm is created and then get the state of the created alarm using its alarm_id. Then get the history of the alarm. And finally the state of the alarm is updated using given state. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc that may be passed while alarm creation.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *state* [*ref*]

  An alarm state to be set

- *timeout* [*ref*]

  The number of seconds for which to attempt a successful check of the alarm state

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.create_and_delete_alarm [Scenario]

Create and delete the newly created alarm.

This scenarios test DELETE /v2/alarms/(alarm_id) Initially alarm is created and then the created alarm is deleted using its alarm_id. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc that may be passed while alarm creation.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.create_and_get_alarm [Scenario]

Create and get the newly created alarm.

These scenarios test GET /v2/alarms/(alarm_id) Initially an alarm is created and then its detailed information is fetched using its alarm_id. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc. that may be passed while creating an alarm.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.create_and_list_alarm [Scenario]

Create and get the newly created alarm.

This scenarios test GET /v2/alarms/(alarm_id) Initially alarm is created and then the created alarm is fetched using its alarm_id. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc. that may be passed while creating an alarm.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.create_and_update_alarm [Scenario]

Create and update the newly created alarm.

This scenarios test PUT /v2/alarms/(alarm_id) Initially alarm is created and then the created alarm is updated using its alarm_id. meter_name and threshold are required parameters for alarm creation. kwargs stores other optional parameters like 'ok_actions', 'project_id' etc that may be passed while alarm creation.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of the alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *kwargs* [*ref*]

  Specifies optional arguments for alarm creation.

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerAlarms.list_alarms [Scenario]

Fetch all alarms.

This scenario fetches list of all alarms using GET /v2/alarms.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.alarms

CeilometerEvents.create_user_and_get_event [Scenario]

Create user and gets event.

This scenario creates user to store new event and fetches one event using GET /v2/events/<message_id>.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.events

CeilometerEvents.create_user_and_list_event_types [Scenario]

Create user and fetch all event types.

This scenario creates user to store new event and fetches list of all events types using GET /v2/event_types.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.events

CeilometerEvents.create_user_and_list_events [Scenario]

Create user and fetch all events.

This scenario creates user to store new event and fetches list of all events using GET /v2/events.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.events

CeilometerMeters.list_matched_meters [Scenario]

Get meters that matched fields from context and args.

Namespace: openstack

Parameters:

- *filter_by_user_id* [*ref*]

  Flag for query by user_id

- *filter_by_project_id* [*ref*]

  Flag for query by project_id

- *filter_by_resource_id* [*ref*]

  Flag for query by resource_id

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *limit* [*ref*]

  Count of resources in response

Module: rally.plugins.openstack.scenarios.ceilometer.meters

CeilometerMeters.list_meters [Scenario]

Check all available queries for list resource request.

Namespace: openstack

Parameters:

- *metadata_query* [*ref*]

  Dict with metadata fields and values

- *limit* [*ref*]

  Limit of meters in response

Module: rally.plugins.openstack.scenarios.ceilometer.meters

CeilometerQueries.create_and_query_alarm_history [Scenario]

Create an alarm and then query for its history.

This scenario tests POST /v2/query/alarms/history An alarm is first created and then its alarm_id is used to fetch the history of that specific alarm.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *orderby* [*ref*]

  Optional param for specifying ordering of results

- *limit* [*ref*]

  Optional param for maximum number of results returned

- *kwargs* [*ref*]

  Optional parameters for alarm creation

Module: rally.plugins.openstack.scenarios.ceilometer.queries

CeilometerQueries.create_and_query_alarms [Scenario]

Create an alarm and then query it with specific parameters.

This scenario tests POST /v2/query/alarms An alarm is first created and then fetched using the input query.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Specifies meter name of alarm

- *threshold* [*ref*]

  Specifies alarm threshold

- *filter* [*ref*]

  Optional filter query dictionary

- *orderby* [*ref*]

  Optional param for specifying ordering of results

- *limit* [*ref*]

  Optional param for maximum number of results returned

- *kwargs* [*ref*]

  Optional parameters for alarm creation

Module: rally.plugins.openstack.scenarios.ceilometer.queries

CeilometerQueries.create_and_query_samples [Scenario]

Create a sample and then query it with specific parameters.

This scenario tests POST /v2/query/samples A sample is first created and then fetched using the input query.

Namespace: openstack

Parameters:

- *counter_name* [*ref*]

  Specifies name of the counter

- *counter_type* [*ref*]

  Specifies type of the counter

- *counter_unit* [*ref*]

  Specifies unit of the counter

- *counter_volume* [*ref*]

  Specifies volume of the counter

- *resource_id* [*ref*]

  Specifies resource id for the sample created

- *filter* [*ref*]

  Optional filter query dictionary

- *orderby* [*ref*]

  Optional param for specifying ordering of results

- *limit* [*ref*]

  Optional param for maximum number of results returned

- *kwargs* [*ref*]

  Parameters for sample creation

Module: rally.plugins.openstack.scenarios.ceilometer.queries

CeilometerResource.get_tenant_resources [Scenario]

Get all tenant resources.

This scenario retrieves information about tenant resources using GET /v2/resources/(resource_id)

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.resources

CeilometerResource.list_matched_resources [Scenario]

Get resources that matched fields from context and args.

Namespace: openstack

Parameters:

- *filter_by_user_id* [*ref*]

  Flag for query by user_id

- *filter_by_project_id* [*ref*]

  Flag for query by project_id

- *filter_by_resource_id* [*ref*]

  Flag for query by resource_id

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *start_time* [*ref*]

  Lower bound of resource timestamp in isoformat

- *end_time* [*ref*]

  Upper bound of resource timestamp in isoformat

- *limit* [*ref*]

  Count of resources in response

Module: rally.plugins.openstack.scenarios.ceilometer.resources

CeilometerResource.list_resources [Scenario]

Check all available queries for list resource request.

This scenario fetches list of all resources using GET /v2/resources.

Namespace: openstack

Parameters:

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *start_time* [*ref*]

  Lower bound of resource timestamp in isoformat

- *end_time* [*ref*]

  Upper bound of resource timestamp in isoformat

- *limit* [*ref*]

  Count of resources in response

Module: rally.plugins.openstack.scenarios.ceilometer.resources

CeilometerSamples.list_matched_samples [Scenario]

Get list of samples that matched fields from context and args.

Namespace: openstack

Parameters:

- *filter_by_user_id* [*ref*]

  Flag for query by user_id

- *filter_by_project_id* [*ref*]

  Flag for query by project_id

- *filter_by_resource_id* [*ref*]

  Flag for query by resource_id

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *limit* [*ref*]

  Count of samples in response

Module: rally.plugins.openstack.scenarios.ceilometer.samples

CeilometerSamples.list_samples [Scenario]

Fetch all available queries for list sample request.

Namespace: openstack

Parameters:

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *limit* [*ref*]

  Count of samples in response

Module: rally.plugins.openstack.scenarios.ceilometer.samples

CeilometerStats.create_meter_and_get_stats [Scenario]

Create a meter and fetch its statistics.

Meter is first created and then statistics is fetched for the same using GET /v2/meters/(meter_name)/statistics.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Contains optional arguments to create a meter

Module: rally.plugins.openstack.scenarios.ceilometer.stats

CeilometerStats.get_stats [Scenario]

Fetch statistics for certain meter.

Statistics is fetched for the using GET /v2/meters/(meter_name)/statistics.

Namespace: openstack

Parameters:

- *meter_name* [*ref*]

  Meter to take statistic for

- *filter_by_user_id* [*ref*]

  Flag for query by user_id

- *filter_by_project_id* [*ref*]

  Flag for query by project_id

- *filter_by_resource_id* [*ref*]

  Flag for query by resource_id

- *metadata_query* [*ref*]

  Dict with metadata fields and values for query

- *period* [*ref*]

  The length of the time range covered by these stats

- *groupby* [*ref*]

  The fields used to group the samples

- *aggregates* [*ref*]

  Name of function for samples aggregation

Returns: list of statistics data

Module: rally.plugins.openstack.scenarios.ceilometer.stats

CeilometerTraits.create_user_and_list_trait_descriptions [Scenario]

Create user and fetch all trait descriptions.

This scenario creates user to store new event and fetches list of all traits for certain event type using GET /v2/event_types/<event_type>/traits.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.traits

CeilometerTraits.create_user_and_list_traits [Scenario]

Create user and fetch all event traits.

This scenario creates user to store new event and fetches list of all traits for certain event type and trait name using GET /v2/event_types/<event_type>/traits/<trait_name>.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ceilometer.traits

CinderVolumeBackups.create_incremental_volume_backup [Scenario]

Create a incremental volume backup.

The scenario first create a volume, the create a backup, the backup is full backup. Because Incremental backup must be based on the full backup. finally create a incremental backup.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size in GB

- *do_delete* [*ref*]

  Deletes backup and volume after creating if True

- *create_volume_kwargs* [*ref*]

  Optional args to create a volume

- *create_backup_kwargs* [*ref*]

  Optional args to create a volume backup

Module: rally.plugins.openstack.scenarios.cinder.volume_backups

CinderVolumeTypes.create_and_delete_encryption_type [Scenario]

Create and delete encryption type

This scenario firstly creates an encryption type for a given volume type, then deletes the created encryption type.

Namespace: openstack

Parameters:

- *create_specs* [*ref*]

  The encryption type specifications to add

Module: rally.plugins.openstack.scenarios.cinder.volume_types

CinderVolumeTypes.create_and_delete_volume_type [Scenario]

Create and delete a volume Type.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional parameters used during volume type creation.

Module: rally.plugins.openstack.scenarios.cinder.volume_types

CinderVolumeTypes.create_and_list_encryption_type [Scenario]

Create and list encryption type

This scenario firstly creates a volume type, secondly creates an encryption type for the volume type, thirdly lists all encryption types.

Namespace: openstack

Parameters:

- *specs* [*ref*]

  The encryption type specifications to add

- *search_opts* [*ref*]

  Options used when search for encryption types

- *kwargs* [*ref*]

  Optional parameters used during volume type creation.

Module: rally.plugins.openstack.scenarios.cinder.volume_types

CinderVolumeTypes.create_and_set_volume_type_keys [Scenario]

Create and set a volume type's extra specs.

Namespace: openstack

Parameters:

- *volume_type_key* [*ref*]

  A dict of key/value pairs to be set

- *kwargs* [*ref*]

  Optional parameters used during volume type creation.

Module: rally.plugins.openstack.scenarios.cinder.volume_types

CinderVolumeTypes.create_volume_type_and_encryption_type [Scenario]

Create encryption type

This scenario first creates a volume type, then creates an encryption type for the volume type.

Namespace: openstack

Parameters:

- *specs* [*ref*]

  The encryption type specifications to add

- *kwargs* [*ref*]

  Optional parameters used during volume type creation.

Module: rally.plugins.openstack.scenarios.cinder.volume_types

CinderVolumes.create_and_accept_transfer [Scenario]

Create a volume transfer, then accept it

Measure the "cinder transfer-create" and "cinder transfer-accept" command performace.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB)

- *image* [*ref*]

  Image to be used to create initial volume

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_attach_volume [Scenario]

Create a VM and attach a volume to it.

Simple test to create a VM and attach a volume, then detach the volume and delete volume/VM.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

      min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Glance image name to use for the VM

- *flavor* [*ref*]

  VM flavor name

- *create_volume_params* [*ref*]

  Optional arguments for volume creation

- *create_vm_params* [*ref*]

  Optional arguments for VM creation

- *kwargs* [*ref*]

  (deprecated) optional arguments for VM creation

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_delete_snapshot [Scenario]

Create and then delete a volume-snapshot.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between snapshot creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *force* [*ref*]

  When set to True, allows snapshot of a volume when the volume is attached to an instance

- *min_sleep* [*ref*]

  Minimum sleep time between snapshot creation and deletion (in seconds)

- *max_sleep* [*ref*]

  Maximum sleep time between snapshot creation and deletion (in seconds)

- *kwargs* [*ref*]

  Optional args to create a snapshot

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_delete_volume [Scenario]

Create and then delete a volume.

Good for testing a maximal bandwidth of cloud. Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between volume creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

      min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Image to be used to create volume

- *min_sleep* [*ref*]

  Minimum sleep time between volume creation and deletion (in seconds)

- *max_sleep* [*ref*]

  Maximum sleep time between volume creation and deletion (in seconds)

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_extend_volume [Scenario]

Create and extend a volume and then delete it.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (in GB) or dictionary, must contain two values:

      min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *new_size* [*ref*]

  Volume new size (in GB) or dictionary, must contain two values:

      min - minimum size volumes will be created as; max - maximum size volumes will be created as.

  to extend. Notice: should be bigger volume size

- *min_sleep* [*ref*]

  Minimum sleep time between volume extension and deletion (in seconds)

- *max_sleep* [*ref*]

  Maximum sleep time between volume extension and deletion (in seconds)

- *kwargs* [*ref*]

  Optional args to extend the volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_get_volume [Scenario]

Create a volume and get the volume.

Measure the "cinder show" command performance.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Image to be used to create volume

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_list_snapshots [Scenario]

Create and then list a volume-snapshot.

Namespace: openstack

Parameters:

- *force* [*ref*]

  When set to True, allows snapshot of a volume when the volume is attached to an instance

- *detailed* [*ref*]

  True if detailed information about snapshots should be listed

- *kwargs* [*ref*]

  Optional args to create a snapshot

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_list_volume [Scenario]

Create a volume and list all volumes.

Measure the "cinder volume-list" command performance.

If you have only 1 user in your context, you will add 1 volume on every iteration. So you will have more and more volumes and will be able to measure the performance of the "cinder volume-list" command depending on the number of images owned by users.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *detailed* [*ref*]

  Determines whether the volume listing should contain detailed information about all of them

- *image* [*ref*]

  Image to be used to create volume

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_list_volume_backups [Scenario]

Create and then list a volume backup.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size in GB

- *detailed* [*ref*]

  True if detailed information about backup should be listed

- *do_delete* [*ref*]

  If True, a volume backup will be deleted

- *create_volume_kwargs* [*ref*]

  Optional args to create a volume

- *create_backup_kwargs* [*ref*]

  Optional args to create a volume backup

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_restore_volume_backup [Scenario]

Restore volume backup.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size in GB

- *do_delete* [*ref*]

  If True, the volume and the volume backup will be deleted after creation.

- *create_volume_kwargs* [*ref*]

  Optional args to create a volume

- *create_backup_kwargs* [*ref*]

  Optional args to create a volume backup

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_update_volume [Scenario]

Create a volume and update its name and description.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB)

- *image* [*ref*]

  Image to be used to create volume

- *create_volume_kwargs* [*ref*]

  Dict, to be used to create volume

- *update_volume_kwargs* [*ref*]

  Dict, to be used to update volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_and_upload_volume_to_image [Scenario]

Create and upload a volume to image.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integers, in GB), or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Image to be used to create volume.

- *force* [*ref*]

  When set to True volume that is attached to an instance could be uploaded to image

- *container_format* [*ref*]

  Image container format

- *disk_format* [*ref*]

  Disk format for image

- *do_delete* [*ref*]

  Deletes image and volume after uploading if True

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_from_volume_and_delete_volume [Scenario]

Create volume from volume and then delete it.

Scenario for testing volume clone.Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between volume creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (in GB), or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

  Should be equal or bigger source volume size

- *min_sleep* [*ref*]

  Minimum sleep time between volume creation and deletion (in seconds)

- *max_sleep* [*ref*]

  Maximum sleep time between volume creation and deletion (in seconds)

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_nested_snapshots_and_attach_volume [Scenario]

Create a volume from snapshot and attach/detach the volume

This scenario create volume, create it's snapshot, attach volume, then create new volume from existing snapshot and so on, with defined nested level, after all detach and delete them. volume->snapshot->volume->snapshot->volume ...

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size - dictionary, contains two values: min - minimum size volumes will be created as; max - maximum size volumes will be created as.

  default values: {"min": 1, "max": 5}

- *nested_level* [*ref*]

  Amount of nested levels

- *create_volume_kwargs* [*ref*]

  Optional args to create a volume

- *create_snapshot_kwargs* [*ref*]

  Optional args to create a snapshot

- *kwargs* [*ref*]

  Optional parameters used during volume snapshot creation.

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_snapshot_and_attach_volume [Scenario]

Create volume, snapshot and attach/detach volume.

Namespace: openstack

Parameters:

- *volume_type* [*ref*]

  Name of volume type to use

- *size* [*ref*]

  Volume size - dictionary, contains two values: min - minimum size volumes will be created as; max - maximum size volumes will be created as.

  default values: {"min": 1, "max": 5}

- *kwargs* [*ref*]

  Optional parameters used during volume snapshot creation.

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_volume [Scenario]

Create a volume.

Good test to check how influence amount of active volumes on performance of creating new.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Image to be used to create volume

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_volume_and_clone [Scenario]

Create a volume, then clone it to another volume.

This creates a volume, then clone it to anothor volume,

and then clone the new volume to next volume...

1. create source volume (from image)

2. clone source volume to volume1
3. clone volume1 to volume2
4. clone volume2 to volume3
5. ...

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB) or dictionary, must contain two values:

  min - minimum size volumes will be created as; max - maximum size volumes will be created as.

- *image* [*ref*]

  Image to be used to create initial volume

- *nested_level* [*ref*]

  Amount of nested levels

- *kwargs* [*ref*]

  Optional args to create volumes

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_volume_and_update_readonly_flag [Scenario]

Create a volume and then update its readonly flag.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size (integer, in GB)

- *image* [*ref*]

  Image to be used to create volume

- *read_only* [*ref*]

  The value to indicate whether to update volume to read-only access mode

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_volume_backup [Scenario]

Create a volume backup.

Namespace: openstack

Parameters:

- *size* [*ref*]

  Volume size in GB

- *do_delete* [*ref*]

  If True, a volume and a volume backup will be deleted after creation.

- *create_volume_kwargs* [*ref*]

  Optional args to create a volume

- *create_backup_kwargs* [*ref*]

  Optional args to create a volume backup

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.create_volume_from_snapshot [Scenario]

Create a volume-snapshot, then create a volume from this snapshot.

Namespace: openstack

Parameters:

- *do_delete* [*ref*]

  If True, a snapshot and a volume will be deleted after creation.

- *create_snapshot_kwargs* [*ref*]

  Optional args to create a snapshot

- *kwargs* [*ref*]

  Optional args to create a volume

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.list_transfers [Scenario]

List all transfers.

This simple scenario tests the "cinder transfer-list" command by listing all the volume transfers.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  If True, detailed information about volume transfer should be listed

- *search_opts* [*ref*]

  Search options to filter out volume transfers.

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.list_types [Scenario]

List all volume types.

This simple scenario tests the cinder type-list command by listing all the volume types.

Namespace: openstack

Parameters:

- *search_opts* [*ref*]

  Options used when search for volume types

- *is_public* [*ref*]

  If query public volume type

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.list_volumes [Scenario]

List all volumes.

This simple scenario tests the cinder list command by listing all the volumes.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if detailed information about volumes should be listed

Module: rally.plugins.openstack.scenarios.cinder.volumes

CinderVolumes.modify_volume_metadata [Scenario]

Modify a volume's metadata.

This requires a volume to be created with the volumes context. Additionally, `sets * set_size` must be greater than or equal to `deletes * delete_size`.

Namespace: openstack

Parameters:

- *sets* [*ref*]

  How many set_metadata operations to perform

- *set_size* [*ref*]

  Number of metadata keys to set in each set_metadata operation

- *deletes* [*ref*]

  How many delete_metadata operations to perform

- *delete_size* [*ref*]

  Number of metadata keys to delete in each delete_metadata operation

Module: rally.plugins.openstack.scenarios.cinder.volumes

DesignateBasic.create_and_delete_domain [Scenario]

Create and then delete a domain.

Measure the performance of creating and deleting domains with different level of load.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_delete_records [Scenario]

Create and then delete records.

Measure the performance of creating and deleting records with different level of load.

Namespace: openstack

Parameters:

- *records_per_domain* [*ref*]

  Records to create pr domain.

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_delete_recordsets [Scenario]

Create and then delete recordsets.

Measure the performance of creating and deleting recordsets with different level of load.

Namespace: openstack

Parameters:

- *recordsets_per_zone* [*ref*]

  Recordsets to create pr zone.

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_delete_server [Scenario]

Create and then delete a server.

Measure the performance of creating and deleting servers with different level of load.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_delete_zone [Scenario]

Create and then delete a zone.

Measure the performance of creating and deleting zones with different level of load.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_list_domains [Scenario]

Create a domain and list all domains.

Measure the "designate domain-list" command performance.

If you have only 1 user in your context, you will add 1 domain on every iteration. So you will have more and more domain and will be able to measure the performance of the "designate domain-list" command depending on the number of domains owned by users.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_list_records [Scenario]

Create and then list records.

If you have only 1 user in your context, you will add 1 record on every iteration. So you will have more and more records and will be able to measure the performance of the "designate record-list" command depending on the number of domains/records owned by users.

Namespace: openstack

Parameters:

- *records_per_domain* [*ref*]

  Records to create pr domain.

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_list_recordsets [Scenario]

Create and then list recordsets.

If you have only 1 user in your context, you will add 1 recordset on every iteration. So you will have more and more recordsets and will be able to measure the performance of the "openstack recordset list" command depending on the number of zones/recordsets owned by users.

Namespace: openstack

Parameters:

- *recordsets_per_zone* [*ref*]

  Recordsets to create pr zone.

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_list_servers [Scenario]

Create a Designate server and list all servers.

If you have only 1 user in your context, you will add 1 server on every iteration. So you will have more and more server and will be able to measure the performance of the "designate server-list" command depending on the number of servers owned by users.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_list_zones [Scenario]

Create a zone and list all zones.

Measure the "openstack zone list" command performance.

If you have only 1 user in your context, you will add 1 zone on every iteration. So you will have more and more zone and will be able to measure the performance of the "openstack zone list" command depending on the number of zones owned by users.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.create_and_update_domain [Scenario]

Create and then update a domain.

Measure the performance of creating and updating domains with different level of load.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.list_domains [Scenario]

List Designate domains.

This simple scenario tests the designate domain-list command by listing all the domains.

Suppose if we have 2 users in context and each has 2 domains uploaded for them we will be able to test the performance of designate domain-list command in this case.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.list_records [Scenario]

List Designate records.

This simple scenario tests the designate record-list command by listing all the records in a domain.

Suppose if we have 2 users in context and each has 2 domains uploaded for them we will be able to test the performance of designate record-list command in this case.

Namespace: openstack

Parameters:

- *domain_id* [*ref*]

  Domain ID

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.list_recordsets [Scenario]

List Designate recordsets.

This simple scenario tests the openstack recordset list command by listing all the recordsets in a zone.

Namespace: openstack

Parameters:

- *zone_id* [*ref*]

  Zone ID

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.list_servers [Scenario]

List Designate servers.

This simple scenario tests the designate server-list command by listing all the servers.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

DesignateBasic.list_zones [Scenario]

List Designate zones.

This simple scenario tests the openstack zone list command by listing all the zones.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.designate.basic

Dummy.dummy [Scenario]

Do nothing and sleep for the given number of seconds (0 by default).

Dummy.dummy can be used for testing performance of different ScenarioRunners and of the ability of rally to store a large amount of results.

Namespace: default

Parameters:

- *sleep* [*ref*]

  Idle time of method (in seconds).

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_exception [Scenario]

Throw an exception.

Dummy.dummy_exception can be used for test if exceptions are processed properly by ScenarioRunners and benchmark and analyze rally results storing process.

Namespace: default

Parameters:

- *size_of_message* [*ref*]

  Int size of the exception message

- *sleep* [*ref*]

  Idle time of method (in seconds).

- *message* [*ref*]

  Message of the exception

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_exception_probability [Scenario]

Throw an exception with given probability.

Dummy.dummy_exception_probability can be used to test if exceptions are processed properly by ScenarioRunners. This scenario will throw an exception sometimes, depending on the given exception probability.

Namespace: default

Parameters:

- *exception_probability* [*ref*]

  Sets how likely it is that an exception will be thrown. Float between 0 and 1 0=never 1=always.

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_output [Scenario]

Generate dummy output.

This scenario generates example of output data.

Namespace: default

Parameters:

- *random_range* [*ref*]

  Max int limit for generated random values

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_random_action [Scenario]

Sleep random time in dummy actions.

Namespace: default

Parameters:

- *actions_num* [*ref*]

  Int number of actions to generate

- *sleep_min* [*ref*]

  Minimal time to sleep, numeric seconds

- *sleep_max* [*ref*]

  Maximum time to sleep, numeric seconds

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_random_fail_in_atomic [Scenario]

Dummy.dummy_random_fail_in_atomic in dummy actions.

Can be used to test atomic actions failures processing.

Namespace: default

Parameters:

- *exception_probability* [*ref*]

  Probability with which atomic actions fail in this dummy scenario (0 <= p <= 1)

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.dummy_timed_atomic_actions [Scenario]

Run some sleepy atomic actions for SLA atomic action tests.

Namespace: default

Parameters:

- *number_of_actions* [*ref*]

  Int number of atomic actions to create

- *sleep_factor* [*ref*]

  Int multiplier for number of seconds to sleep

Module: rally.plugins.common.scenarios.dummy.dummy

Dummy.failure [Scenario]

Raise errors in some iterations.

Namespace: default

Parameters:

- *sleep* [*ref*]

  Float iteration sleep time in seconds

- *from_iteration* [*ref*]

  Int iteration number which starts range of failed iterations

- *to_iteration* [*ref*]

  Int iteration number which ends range of failed iterations

- *each* [*ref*]

  Int cyclic number of iteration which actually raises an error in selected range. For example, each=3 will raise error in each 3rd iteration.

Module: rally.plugins.common.scenarios.dummy.dummy

EC2Servers.boot_server [Scenario]

Boot a server.

Assumes that cleanup is done elsewhere.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.ec2.servers

EC2Servers.list_servers [Scenario]

List all servers.

This simple scenario tests the EC2 API list function by listing all the servers.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.ec2.servers

FuelEnvironments.create_and_delete_environment [Scenario]

Create and delete Fuel environments.

Namespace: openstack

Parameters:

- *release_id* [*ref*]

  Release id (default 1)

- *network_provider* [*ref*]

  Network provider (default 'neutron')

- *deployment_mode* [*ref*]

  Deployment mode (default 'ha_compact')

- *net_segment_type* [*ref*]

  Net segment type (default 'vlan')

- *delete_retries* [*ref*]

  Retries count on delete operations (default 5)

Module: rally.plugins.openstack.scenarios.fuel.environments

FuelEnvironments.create_and_list_environments [Scenario]

Create and list Fuel environments.

Namespace: openstack

Parameters:

- *release_id* [*ref*]

  Release id (default 1)

- *network_provider* [*ref*]

  Network provider (default 'neutron')

- *deployment_mode* [*ref*]

  Deployment mode (default 'ha_compact')

- *net_segment_type* [*ref*]

  Net segment type (default 'vlan')

Module: rally.plugins.openstack.scenarios.fuel.environments

FuelNodes.add_and_remove_node [Scenario]

Add node to environment and remove.

Namespace: openstack

Parameters:

- *node_roles* [*ref*]

  List. Roles, which node should be assigned to env with

Module: rally.plugins.openstack.scenarios.fuel.nodes

GlanceImages.create_and_delete_image [Scenario]

Create and then delete an image.

Namespace: openstack

Parameters:

- *container_format* [*ref*]

  Container format of image. Acceptable formats: ami, ari, aki, bare, and ovf

- *image_location* [*ref*]

  Image file location

- *disk_format* [*ref*]

  Disk format of image. Acceptable formats: ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, and iso

- *kwargs* [*ref*]

  Optional parameters to create image

Module: rally.plugins.openstack.scenarios.glance.images

GlanceImages.create_and_list_image [Scenario]

Create an image and then list all images.

Measure the "glance image-list" command performance.

If you have only 1 user in your context, you will add 1 image on every iteration. So you will have more and more images and will be able to measure the performance of the "glance image-list" command depending on the number of images owned by users.

Namespace: openstack

Parameters:

- *container_format* [*ref*]

  Container format of image. Acceptable formats: ami, ari, aki, bare, and ovf

- *image_location* [*ref*]

  Image file location

- *disk_format* [*ref*]

  Disk format of image. Acceptable formats: ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, and iso

- *kwargs* [*ref*]

  Optional parameters to create image

Module: rally.plugins.openstack.scenarios.glance.images

GlanceImages.create_image_and_boot_instances [Scenario]

Create an image and boot several instances from it.

Namespace: openstack

Parameters:

- *container_format* [*ref*]

  Container format of image. Acceptable formats: ami, ari, aki, bare, and ovf

- *image_location* [*ref*]

  Image file location

- *disk_format* [*ref*]

  Disk format of image. Acceptable formats: ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, and iso

- *flavor* [*ref*]

  Nova flavor to be used to launch an instance

- *number_instances* [*ref*]

  Number of Nova servers to boot

- *create_image_kwargs* [*ref*]

  Optional parameters to create image

- *boot_server_kwargs* [*ref*]

  Optional parameters to boot server

- *kwargs* [*ref*]

  Optional parameters to create server (deprecated)

Module: rally.plugins.openstack.scenarios.glance.images

GlanceImages.list_images [Scenario]

List all images.

This simple scenario tests the glance image-list command by listing all the images.

Suppose if we have 2 users in context and each has 2 images uploaded for them we will be able to test the performance of glance image-list command in this case.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.glance.images

HeatStacks.create_and_delete_stack [Scenario]

Create and then delete a stack.

Measure the "heat stack-create" and "heat stack-delete" commands performance.

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_and_list_stack [Scenario]

Create a stack and then list all stacks.

Measure the "heat stack-create" and "heat stack-list" commands performance.

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_check_delete_stack [Scenario]

Create, check and delete a stack.

Measure the performance of the following commands: - heat stack-create - heat action-check - heat stack-delete

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_snapshot_restore_delete_stack [Scenario]

Create, snapshot-restore and then delete a stack.

Measure performance of the following commands: heat stack-create heat stack-snapshot heat stack-restore heat stack-delete

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_stack_and_list_output [Scenario]

Create stack and list outputs by using new algorithm.

Measure performance of the following commands: heat stack-create heat output-list

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_stack_and_list_output_via_API [Scenario]

Create stack and list outputs by using old algorithm.

Measure performance of the following commands: heat stack-create heat output-list

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_stack_and_scale [Scenario]

Create an autoscaling stack and invoke a scaling policy.

Measure the performance of autoscaling webhooks.

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to template file that includes an OS::Heat::AutoScalingGroup resource

- *output_key* [*ref*]

  The stack output key that corresponds to the scaling webhook

- *delta* [*ref*]

  The number of instances the stack is expected to change by.

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template (dict of file name to file path)

- *environment* [*ref*]

  Stack environment definition (dict)

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_stack_and_show_output [Scenario]

Create stack and show output by using new algorithm.

Measure performance of the following commands: heat stack-create heat output-show

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *output_key* [*ref*]

  The stack output key that corresponds to the scaling webhook

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_stack_and_show_output_via_API [Scenario]

Create stack and show output by using old algorithm.

Measure performance of the following commands: heat stack-create heat output-show

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *output_key* [*ref*]

  The stack output key that corresponds to the scaling webhook

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_suspend_resume_delete_stack [Scenario]

Create, suspend-resume and then delete a stack.

Measure performance of the following commands: heat stack-create heat action-suspend heat action-resume heat stack-delete

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *files* [*ref*]

  Files used in template

- *environment* [*ref*]

  Stack environment definition

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.create_update_delete_stack [Scenario]

Create, update and then delete a stack.

Measure the "heat stack-create", "heat stack-update" and "heat stack-delete" commands performance.

Namespace: openstack

Parameters:

- *template_path* [*ref*]

  Path to stack template file

- *updated_template_path* [*ref*]

  Path to updated stack template file

- *parameters* [*ref*]

  Parameters to use in heat template

- *updated_parameters* [*ref*]

  Parameters to use in updated heat template If not specified then parameters will be used instead

- *files* [*ref*]

  Files used in template

- *updated_files* [*ref*]

  Files used in updated template. If not specified files value will be used instead

- *environment* [*ref*]

  Stack environment definition

- *updated_environment* [*ref*]

  Environment definition for updated stack

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.list_stacks_and_events [Scenario]

List events from tenant stacks.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.heat.stacks

HeatStacks.list_stacks_and_resources [Scenario]

List all resources from tenant stacks.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.heat.stacks

HttpRequests.check_random_request [Scenario]

Benchmark the list of requests

This scenario takes random url from list of requests, and raises exception if the response is not the expected response.

Namespace: default

Parameters:

- *requests* [*ref*]

  List of request dicts

- *status_code* [*ref*]

  Expected Response Code it will be used only if we doesn't specified it in request proper

Module: rally.plugins.common.scenarios.requests.http_requests

HttpRequests.check_request [Scenario]

Standard way to benchmark web services.

This benchmark is used to make request and check it with expected Response.

Namespace: default

Parameters:

- *url* [*ref*]

  Url for the Request object

- *method* [*ref*]

  Method for the Request object

- *status_code* [*ref*]

  Expected response code

- *kwargs* [*ref*]

  Optional additional request parameters

Module: rally.plugins.common.scenarios.requests.http_requests

IronicNodes.create_and_delete_node [Scenario]

Create and delete node.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional additional arguments for node creation

Module: rally.plugins.openstack.scenarios.ironic.nodes

IronicNodes.create_and_list_node [Scenario]

Create and list nodes.

Namespace: openstack

Parameters:

- *associated* [*ref*]

  Optional. Either a Boolean or a string representation of a Boolean that indicates whether to return a list of associated (True or "True") or unassociated (False or "False") nodes.

- *maintenance* [*ref*]

  Optional. Either a Boolean or a string representation of a Boolean that indicates whether to return nodes in maintenance mode (True or "True"), or not in maintenance mode (False or "False").

- *marker* [*ref*]

  Optional, the UUID of a node, eg the last node from a previous result set. Return the next result set.

- *limit* [*ref*]

  The maximum number of results to return per request, if:

  1. limit > 0, the maximum number of nodes to return.
  2. limit == 0, return the entire list of nodes.
  3. limit param is NOT specified (None), the number of items returned respect the maximum imposed by the Ironic API (see Ironic's api.max_limit option).

- *detail* [*ref*]

  Optional, boolean whether to return detailed information about nodes.

- *sort_key* [*ref*]

  Optional, field used for sorting.

- *sort_dir* [*ref*]

  Optional, direction of sorting, either 'asc' (the default) or 'desc'.

- *kwargs* [*ref*]

  Optional additional arguments for node creation

Module: rally.plugins.openstack.scenarios.ironic.nodes

KeystoneBasic.add_and_remove_user_role [Scenario]

Create a user role add to a user and disassociate.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.authenticate_user_and_validate_token [Scenario]

Authenticate and validate a keystone token.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_add_and_list_user_roles [Scenario]

Create user role, add it and list user roles for given user.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_delete_ec2credential [Scenario]

Create and delete keystone ec2-credential.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_delete_role [Scenario]

Create a user role and delete it.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_delete_service [Scenario]

Create and delete service.

Namespace: openstack

Parameters:

- *service_type* [*ref*]

  Type of the service

- *description* [*ref*]

  Description of the service

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_get_role [Scenario]

Create a user role and get it detailed information.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional additional arguments for roles creation

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_list_ec2credentials [Scenario]

Create and List all keystone ec2-credentials.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_list_roles [Scenario]

Create a role, then list all roles.

Namespace: openstack

Parameters:

- *create_role_kwargs* [*ref*]

  Optional additional arguments for roles create

- *list_role_kwargs* [*ref*]

  Optional additional arguments for roles list

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_list_services [Scenario]

Create and list services.

Namespace: openstack

Parameters:

- *service_type* [*ref*]

  Type of the service

- *description* [*ref*]

  Description of the service

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_list_tenants [Scenario]

Create a keystone tenant with random name and list all tenants.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_list_users [Scenario]

Create a keystone user with random name and list all users.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters to create users like “tenant_id”, “enabled”.

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_and_update_user [Scenario]

Create user and update the user.

Namespace: openstack

Parameters:

- *create_user_kwargs* [*ref*]

  Optional additional arguments for user creation

- *update_user_kwargs* [*ref*]

  Optional additional arguments for user updation

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_delete_user [Scenario]

Create a keystone user with random name and then delete it.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters to create users like “tenant_id”, “enabled”.

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_tenant [Scenario]

Create a keystone tenant with random name.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_tenant_with_users [Scenario]

Create a keystone tenant and several users belonging to it.

Namespace: openstack

Parameters:

- *users_per_tenant* [*ref*]

  Number of users to create for the tenant

- *kwargs* [*ref*]

  Other optional parameters for tenant creation

Returns: keystone tenant instance

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_update_and_delete_tenant [Scenario]

Create, update and delete tenant.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters for tenant creation

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_user [Scenario]

Create a keystone user with random name.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters to create users like “tenant_id”, “enabled”.

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_user_set_enabled_and_delete [Scenario]

Create a keystone user, enable or disable it, and delete it.

Namespace: openstack

Parameters:

- *enabled* [*ref*]

  Initial state of user ‘enabled’ flag. The user will be created with ‘enabled’ set to this value, and then it will be toggled.

- *kwargs* [*ref*]

  Other optional parameters to create user.

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.create_user_update_password [Scenario]

Create user and update password for that user.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.keystone.basic

KeystoneBasic.get_entities [Scenario]

Get instance of a tenant, user, role and service by id's.

An ephemeral tenant, user, and role are each created. By default, fetches the 'keystone' service. This can be overridden (for instance, to get the 'Identity Service' service on older OpenStack), or None can be passed explicitly to service_name to create a new service and then query it by ID.

Namespace: openstack

Parameters:

- *service_name* [*ref*]

  The name of the service to get by ID; or None, to create an ephemeral service and get it by ID.

Module: rally.plugins.openstack.scenarios.keystone.basic

MagnumClusterTemplates.list_cluster_templates [Scenario]

List all cluster_templates.

Measure the "magnum cluster_template-list" command performance.

Namespace: openstack

Parameters:

- *limit* [*ref*]

  (Optional) The maximum number of results to return per request, if:

  1. limit > 0, the maximum number of cluster_templates to return.
  2. limit param is NOT specified (None), the number of items returned respect the maximum imposed by the Magnum API (see Magnum's api.max_limit option).

- *kwargs* [*ref*]

  Optional additional arguments for cluster_templates listing

Module: rally.plugins.openstack.scenarios.magnum.cluster_templates

MagnumClusters.create_and_list_clusters [Scenario]

create cluster and then list all clusters.

Namespace: openstack

Parameters:

- *node_count* [*ref*]

  The cluster node count.

- *cluster_template_uuid* [*ref*]

  Optional, if user want to use an existing cluster_template

- *kwargs* [*ref*]

  Optional additional arguments for cluster creation

Module: rally.plugins.openstack.scenarios.magnum.clusters

MagnumClusters.list_clusters [Scenario]

List all clusters.

Measure the "magnum clusters-list" command performance.

Namespace: openstack

Parameters:

- *limit* [*ref*]

  (Optional) The maximum number of results to return per request, if:

  1. limit > 0, the maximum number of clusters to return.
  2. limit param is NOT specified (None), the number of items returned respect the maximum imposed by the Magnum API (see Magnum's api.max_limit option).

- *kwargs* [*ref*]

  Optional additional arguments for clusters listing

Module: rally.plugins.openstack.scenarios.magnum.clusters

ManilaShares.attach_security_service_to_share_network [Scenario]

Attaches security service to share network.

Namespace: openstack

Parameters:

- *security_service_type* [*ref*]

  Type of security service to use. Should be one of following: 'ldap', 'kerberos' or 'active_directory'.

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.create_and_delete_share [Scenario]

Create and delete a share.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between share creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *share_proto* [*ref*]

  Share protocol, valid values are NFS, CIFS, GlusterFS and HDFS

- *size* [*ref*]

  Share size in GB, should be greater than 0

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *kwargs* [*ref*]

  Optional args to create a share

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.create_and_list_share [Scenario]

Create a share and list all shares.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between share creation and list (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *share_proto* [*ref*]

  Share protocol, valid values are NFS, CIFS, GlusterFS and HDFS

- *size* [*ref*]

  Share size in GB, should be greater than 0

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *detailed* [*ref*]

  Defines whether to get detailed list of shares or not

- *kwargs* [*ref*]

  Optional args to create a share

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.create_security_service_and_delete [Scenario]

Creates security service and then deletes.

Namespace: openstack

Parameters:

- *security_service_type* [*ref*]

  Security service type, permitted values are 'ldap', 'kerberos' or 'active_directory'.

- *dns_ip* [*ref*]

  Dns ip address used inside tenant's network

- *server* [*ref*]

  Security service server ip address or hostname

- *domain* [*ref*]

  Security service domain

- *user* [*ref*]

  Security identifier used by tenant

- *password* [*ref*]

  Password used by user

- *description* [*ref*]

  Security service description

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.create_share_network_and_delete [Scenario]

Creates share network and then deletes.

Namespace: openstack

Parameters:

- *neutron_net_id* [*ref*]

  ID of Neutron network

- *neutron_subnet_id* [*ref*]

  ID of Neutron subnet

- *nova_net_id* [*ref*]

  ID of Nova network

- *description* [*ref*]

  Share network description

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.create_share_network_and_list [Scenario]

Creates share network and then lists it.

Namespace: openstack

Parameters:

- *neutron_net_id* [*ref*]

  ID of Neutron network

- *neutron_subnet_id* [*ref*]

  ID of Neutron subnet

- *nova_net_id* [*ref*]

  ID of Nova network

- *description* [*ref*]

  Share network description

- *detailed* [*ref*]

  Defines either to return detailed list of objects or not.

- *search_opts* [*ref*]

  Container of search opts such as “name”, “nova_net_id”, “neutron_net_id”, etc.

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.list_share_servers [Scenario]

Lists share servers.

Requires admin creds.

Namespace: openstack

Parameters:

- *search_opts* [*ref*]

  Container of following search opts: “host”, “status”, “share_network” and “project_id”.

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.list_shares [Scenario]

Basic scenario for ‘share list’ operation.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  Defines either to return detailed list of objects or not.

- *search_opts* [*ref*]

  Container of search opts such as “name”, “host”, “share_type”, etc.

Module: rally.plugins.openstack.scenarios.manila.shares

ManilaShares.set_and_delete_metadata [Scenario]

Sets and deletes share metadata.

This requires a share to be created with the shares context. Additionally, `sets * set_size` must be greater than or equal to `deletes * delete_size`.

Namespace: openstack

Parameters:

- *sets* [*ref*]

  How many set_metadata operations to perform

- *set_size* [*ref*]

  Number of metadata keys to set in each set_metadata operation

- *delete_size* [*ref*]

  Number of metadata keys to delete in each delete_metadata operation

- *key_min_length* [*ref*]

  Minimal size of metadata key to set

- *key_max_length* [*ref*]

  Maximum size of metadata key to set

- *value_min_length* [*ref*]

  Minimal size of metadata value to set

- *value_max_length* [*ref*]

  Maximum size of metadata value to set

Module: rally.plugins.openstack.scenarios.manila.shares

MistralExecutions.create_execution_from_workbook [Scenario]

Scenario tests execution creation and deletion.

This scenario is a very useful tool to measure the "mistral execution-create" and "mistral execution-delete" commands performance.

Namespace: openstack

Parameters:

- *definition* [*ref*]

  String (yaml string) representation of given file content (Mistral workbook definition)

- *workflow_name* [*ref*]

  String the workflow name to execute. Should be one of the to workflows in the definition. If no

  workflow_name is passed, one of the workflows in the definition will be taken.

- *wf_input* [*ref*]

  File containing a json string of mistral workflow input

- *params* [*ref*]

  File containing a json string of mistral params (the string is the place to pass the environment)

- *do_delete* [*ref*]

  If False than it allows to check performance in "create only" mode.

Module: rally.plugins.openstack.scenarios.mistral.executions

MistralExecutions.list_executions [Scenario]

Scenario test mistral execution-list command.

This simple scenario tests the Mistral execution-list command by listing all the executions.

Namespace: openstack

Parameters:

- *marker* [*ref*]

  The last execution uuid of the previous page, displays list of executions after "marker".

- *limit* [*ref*]

  Number Maximum number of executions to return in a single result.

- *sort_keys* [*ref*]

  Id,description

- *sort_dirs* [*ref*]

  [SORT_DIRS] Comma-separated list of sort directions. Default: asc.

Module: rally.plugins.openstack.scenarios.mistral.executions

MistralWorkbooks.create_workbook [Scenario]

Scenario tests workbook creation and deletion.

This scenario is a very useful tool to measure the "mistral workbook-create" and "mistral workbook-delete" commands performance.

Namespace: openstack

Parameters:

- *definition* [*ref*]

  String (yaml string) representation of given file content (Mistral workbook definition)

- *do_delete* [*ref*]

  If False than it allows to check performance in "create only" mode.

Module: rally.plugins.openstack.scenarios.mistral.workbooks

MistralWorkbooks.list_workbooks [Scenario]

Scenario test mistral workbook-list command.

This simple scenario tests the Mistral workbook-list command by listing all the workbooks.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.mistral.workbooks

MonascaMetrics.list_metrics [Scenario]

Fetch user's metrics.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional arguments for list query: name, dimensions, start_time, etc

Module: rally.plugins.openstack.scenarios.monasca.metrics

MuranoEnvironments.create_and_delete_environment [Scenario]

Create environment, session and delete environment.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.murano.environments

MuranoEnvironments.create_and_deploy_environment [Scenario]

Create environment, session and deploy environment.

Create environment, create session, add app to environment packages_per_env times, send environment to deploy.

Namespace: openstack

Parameters:

- *packages_per_env* [*ref*]

  Number of packages per environment

Module: rally.plugins.openstack.scenarios.murano.environments

MuranoEnvironments.list_environments [Scenario]

List the murano environments.

Run murano environment-list for listing all environments.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.murano.environments

MuranoPackages.import_and_delete_package [Scenario]

Import Murano package and then delete it.

Measure the "murano import-package" and "murano package-delete" commands performance. It imports Murano package from "package" (if it is not a zip archive then zip archive will be prepared) and deletes it.

Namespace: default

Parameters:

- *package* [*ref*]

  Path to zip archive that represents Murano application package or absolute path to folder with package components

Module: rally.plugins.openstack.scenarios.murano.packages

MuranoPackages.import_and_filter_applications [Scenario]

Import Murano package and then filter packages by some criteria.

Measure the performance of package import and package filtering commands. It imports Murano package from "package" (if it is not a zip archive then zip archive will be prepared) and filters packages by some criteria.

Namespace: default

Parameters:

- *package* [*ref*]

  Path to zip archive that represents Murano application package or absolute path to folder with package components

- *filter_query* [*ref*]

  Dict that contains filter criteria, lately it will be passed as **kwargs to filter method e.g. {"category": "Web"}

Module: rally.plugins.openstack.scenarios.murano.packages

MuranoPackages.import_and_list_packages [Scenario]

Import Murano package and get list of packages.

Measure the "murano import-package" and "murano package-list" commands performance. It imports Murano package from "package" (if it is not a zip archive then zip archive will be prepared) and gets list of imported packages.

Namespace: default

Parameters:

- *package* [*ref*]

  Path to zip archive that represents Murano application package or absolute path to folder with package components

- *include_disabled* [*ref*]

  Specifies whether the disabled packages will be included in a the result or not. Default value is False.

Module: rally.plugins.openstack.scenarios.murano.packages

MuranoPackages.package_lifecycle [Scenario]

Import Murano package, modify it and then delete it.

Measure the Murano import, update and delete package commands performance. It imports Murano package from "package" (if it is not a zip archive then zip archive will be prepared), modifies it (using data from "body") and deletes.

Namespace: default

Parameters:

- *package* [*ref*]

  Path to zip archive that represents Murano application package or absolute path to folder with package components

- *body* [*ref*]

  Dict object that defines what package property will be updated, e.g {"tags": ["tag"]} or {"enabled": "true"}

- *operation* [*ref*]

  String object that defines the way of how package property will be updated, allowed operations are "add", "replace" or "delete". Default value is "replace".

Module: rally.plugins.openstack.scenarios.murano.packages

NeutronLoadbalancerV1.create_and_delete_healthmonitors [Scenario]

Create a healthmonitor(v1) and delete healthmonitors(v1).

Measure the "neutron lb-healthmonitor-create" and "neutron lb-healthmonitor-delete" command performance. The scenario creates healthmonitors and deletes those healthmonitors.

Namespace: openstack

Parameters:

- *healthmonitor_create_args* [*ref*]

  Dict, POST /lb/healthmonitors request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_delete_pools [Scenario]

Create pools(v1) and delete pools(v1).

Measure the "neutron lb-pool-create" and "neutron lb-pool-delete" command performance. The scenario creates a pool for every subnet and then deletes those pools.

Namespace: openstack

Parameters:

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_delete_vips [Scenario]

Create a vip(v1) and then delete vips(v1).

Measure the "neutron lb-vip-create" and "neutron lb-vip-delete" command performance. The scenario creates a vip for pool and then deletes those vips.

Namespace: openstack

Parameters:

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

- *vip_create_args* [*ref*]

  Dict, POST /lb/vips request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_list_healthmonitors [Scenario]

Create healthmonitors(v1) and list healthmonitors(v1).

Measure the "neutron lb-healthmonitor-list" command performance. This scenario creates healthmonitors and lists them.

Namespace: openstack

Parameters:

- *healthmonitor_create_args* [*ref*]

  Dict, POST /lb/healthmonitors request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_list_pools [Scenario]

Create a pool(v1) and then list pools(v1).

Measure the "neutron lb-pool-list" command performance. The scenario creates a pool for every subnet and then lists pools.

Namespace: openstack

Parameters:

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_list_vips [Scenario]

Create a vip(v1) and then list vips(v1).

Measure the "neutron lb-vip-create" and "neutron lb-vip-list" command performance. The scenario creates a vip for every pool created and then lists vips.

Namespace: openstack

Parameters:

- *vip_create_args* [*ref*]

  Dict, POST /lb/vips request options

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_update_healthmonitors [Scenario]

Create a healthmonitor(v1) and update healthmonitors(v1).

Measure the "neutron lb-healthmonitor-create" and "neutron lb-healthmonitor-update" command performance. The scenario creates healthmonitors and then updates them.

Namespace: openstack

Parameters:

- *healthmonitor_create_args* [*ref*]

  Dict, POST /lb/healthmonitors request options

- *healthmonitor_update_args* [*ref*]

  Dict, POST /lb/healthmonitors update options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_update_pools [Scenario]

Create pools(v1) and update pools(v1).

Measure the "neutron lb-pool-create" and "neutron lb-pool-update" command performance. The scenario creates a pool for every subnet and then update those pools.

Namespace: openstack

Parameters:

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

- *pool_update_args* [*ref*]

  Dict, POST /lb/pools update options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV1.create_and_update_vips [Scenario]

Create vips(v1) and update vips(v1).

Measure the "neutron lb-vip-create" and "neutron lb-vip-update" command performance. The scenario creates a pool for every subnet and then update those pools.

Namespace: openstack

Parameters:

- *pool_create_args* [*ref*]

  Dict, POST /lb/pools request options

- *vip_create_args* [*ref*]

  Dict, POST /lb/vips request options

- *vip_update_args* [*ref*]

  Dict, POST /lb/vips update options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v1

NeutronLoadbalancerV2.create_and_list_loadbalancers [Scenario]

Create a loadbalancer(v2) and then list loadbalancers(v2).

Measure the "neutron lbaas-loadbalancer-list" command performance. The scenario creates a loadbalancer for every subnet and then lists loadbalancers.

Namespace: openstack

Parameters:

- *lb_create_args* [*ref*]

  Dict, POST /lbaas/loadbalancers request options

Module: rally.plugins.openstack.scenarios.neutron.loadbalancer_v2

NeutronNetworks.create_and_delete_floating_ips [Scenario]

Create and delete floating IPs.

Measure the "neutron floating-ip-create" and "neutron floating-ip-delete" commands performance.

Namespace: openstack

Parameters:

- *floating_network* [*ref*]

  Str, external network for floating IP creation

- *floating_ip_args* [*ref*]

  Dict, POST /floatingips request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_delete_networks [Scenario]

Create and delete a network.

Measure the "neutron net-create" and "net-delete" command performance.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_delete_ports [Scenario]

Create and delete a port.

Measure the "neutron port-create" and "neutron port-delete" commands performance.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *port_create_args* [*ref*]

  Dict, POST /v2.0/ports request options

- *ports_per_network* [*ref*]

  Int, number of ports for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_delete_routers [Scenario]

Create and delete a given number of routers.

Create a network, a given number of subnets and routers and then delete all routers.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

- *router_create_args* [*ref*]

  Dict, POST /v2.0/routers request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_delete_subnets [Scenario]

Create and delete a given number of subnets.

The scenario creates a network, a given number of subnets and then deletes subnets.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_list_floating_ips [Scenario]

Create and list floating IPs.

Measure the "neutron floating-ip-create" and "neutron floating-ip-list" commands performance.

Namespace: openstack

Parameters:

- *floating_network* [*ref*]

  Str, external network for floating IP creation

- *floating_ip_args* [*ref*]

  Dict, POST /floatingips request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_list_networks [Scenario]

Create a network and then list all networks.

Measure the "neutron net-list" command performance.

If you have only 1 user in your context, you will add 1 network on every iteration. So you will have more and more networks and will be able to measure the performance of the "neutron net-list" command depending on the number of networks owned by users.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_list_ports [Scenario]

Create and a given number of ports and list all ports.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *port_create_args* [*ref*]

  Dict, POST /v2.0/ports request options

- *ports_per_network* [*ref*]

  Int, number of ports for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_list_routers [Scenario]

Create and a given number of routers and list all routers.

Create a network, a given number of subnets and routers and then list all routers.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

- *router_create_args* [*ref*]

  Dict, POST /v2.0/routers request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_list_subnets [Scenario]

Create and a given number of subnets and list all subnets.

The scenario creates a network, a given number of subnets and then lists subnets.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_show_network [Scenario]

Create a network and show network details.

Measure the "neutron net-show" command performance.

Namespace: openstack

Parameters:

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_update_networks [Scenario]

Create and update a network.

Measure the "neutron net-create and net-update" command performance.

Namespace: openstack

Parameters:

- *network_update_args* [*ref*]

  Dict, PUT /v2.0/networks update request

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_update_ports [Scenario]

Create and update a given number of ports.

Measure the "neutron port-create" and "neutron port-update" commands performance.

Namespace: openstack

Parameters:

- *port_update_args* [*ref*]

  Dict, PUT /v2.0/ports update request options

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *port_create_args* [*ref*]

  Dict, POST /v2.0/ports request options

- *ports_per_network* [*ref*]

  Int, number of ports for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_update_routers [Scenario]

Create and update a given number of routers.

Create a network, a given number of subnets and routers and then updating all routers.

Namespace: openstack

Parameters:

- *router_update_args* [*ref*]

  Dict, PUT /v2.0/routers update options

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

- *router_create_args* [*ref*]

  Dict, POST /v2.0/routers request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.create_and_update_subnets [Scenario]

Create and update a subnet.

The scenario creates a network, a given number of subnets and then updates the subnet. This scenario measures the "neutron subnet-update" command performance.

Namespace: openstack

Parameters:

- *subnet_update_args* [*ref*]

  Dict, PUT /v2.0/subnets update options

- *network_create_args* [*ref*]

  Dict, POST /v2.0/networks request options. Deprecated.

- *subnet_create_args* [*ref*]

  Dict, POST /v2.0/subnets request options

- *subnet_cidr_start* [*ref*]

  Str, start value for subnets CIDR

- *subnets_per_network* [*ref*]

  Int, number of subnets for one network

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronNetworks.list_agents [Scenario]

List all neutron agents.

This simple scenario tests the "neutron agent-list" command by listing all the neutron agents.

Namespace: openstack

Parameters:

- *agent_args* [*ref*]

  Dict, POST /v2.0/agents request options

Module: rally.plugins.openstack.scenarios.neutron.network

NeutronSecurityGroup.create_and_delete_security_groups [Scenario]

Create and delete Neutron security-groups.

Measure the "neutron security-group-create" and "neutron security-group-delete" command performance.

Namespace: openstack

Parameters:

- *security_group_create_args* [*ref*]

  Dict, POST /v2.0/security-groups request options

Module: rally.plugins.openstack.scenarios.neutron.security_groups

NeutronSecurityGroup.create_and_list_security_groups [Scenario]

Create and list Neutron security-groups.

Measure the "neutron security-group-create" and "neutron security-group-list" command performance.

Namespace: openstack

Parameters:

- *security_group_create_args* [*ref*]

  Dict, POST /v2.0/security-groups request options

Module: rally.plugins.openstack.scenarios.neutron.security_groups

NeutronSecurityGroup.create_and_update_security_groups [Scenario]

Create and update Neutron security-groups.

Measure the "neutron security-group-create" and "neutron security-group-update" command performance.

Namespace: openstack

Parameters:

- *security_group_create_args* [*ref*]

  Dict, POST /v2.0/security-groups request options

- *security_group_update_args* [*ref*]

  Dict, PUT /v2.0/security-groups update options

Module: rally.plugins.openstack.scenarios.neutron.security_groups

NovaAgents.list_agents [Scenario]

List all builds.

Measure the "nova agent-list" command performance.

Namespace: openstack

Parameters:

- *hypervisor* [*ref*]

  List agent builds on a specific hypervisor. None (default value) means list for all hypervisors

Module: rally.plugins.openstack.scenarios.nova.agents

NovaAggregates.create_aggregate_add_and_remove_host [Scenario]

Create an aggregate, add a host to and remove the host from it

Measure "nova aggregate-add-host" and "nova aggregate-remove-host" command performance.

Namespace: openstack

Parameters:

- *availability_zone* [*ref*]

  The availability zone of the aggregate

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.create_aggregate_add_host_and_boot_server [Scenario]

Scenario to create and verify an aggregate

This scenario creates an aggregate, adds a compute host and metadata to the aggregate, adds the same metadata to the flavor and creates an instance. Verifies that instance host is one of the hosts in the aggregate.

Namespace: openstack

Parameters:

- *image* [*ref*]

  The image ID to boot from

- *metadata* [*ref*]

  The metadata to be set as flavor extra specs

- *availability_zone* [*ref*]

  The availability zone of the aggregate

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *boot_server_kwargs* [*ref*]

  Optional additional arguments to verify host aggregates

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.create_and_delete_aggregate [Scenario]

Create an aggregate and then delete it.

This scenario first creates an aggregate and then delete it.

Namespace: openstack

Parameters:

- *availability_zone* [*ref*]

  The availability zone of the aggregate

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.create_and_get_aggregate_details [Scenario]

Create an aggregate and then get its details.

This scenario first creates an aggregate and then get details of it.

Namespace: openstack

Parameters:

- *availability_zone* [*ref*]

  The availability zone of the aggregate

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.create_and_list_aggregates [Scenario]

Create a aggregate and then list all aggregates.

This scenario creates a aggregate and then lists all aggregates.

Namespace: openstack

Parameters:

- *availability_zone* [*ref*]

  The availability zone of the aggregate

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.create_and_update_aggregate [Scenario]

Create an aggregate and then update its name and availability_zone

This scenario first creates an aggregate and then update its name and availability_zone

Namespace: openstack

Parameters:

- *availability_zone* [*ref*]

  The availability zone of the aggregate

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAggregates.list_aggregates [Scenario]

List all nova aggregates.

Measure the "nova aggregate-list" command performance.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.nova.aggregates

NovaAvailabilityZones.list_availability_zones [Scenario]

List all availability zones.

Measure the "nova availability-zone-list" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the availability-zone listing should contain detailed information about all of them

Module: rally.plugins.openstack.scenarios.nova.availability_zones

NovaFlavors.create_and_delete_flavor [Scenario]

Create flavor and delete the flavor.

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.create_and_get_flavor [Scenario]

Create flavor and get detailed information of the flavor.

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.create_and_list_flavor_access [Scenario]

Create a non-public flavor and list its access rules

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.create_flavor [Scenario]

Create a flavor.

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.create_flavor_and_add_tenant_access [Scenario]

Create a flavor and Add flavor access for the given tenant.

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.create_flavor_and_set_keys [Scenario]

Create flavor and set keys to the flavor.

Measure the "nova flavor-key" command performance. the scenario first create a flavor,then add the extra specs to it.

Namespace: openstack

Parameters:

- *ram* [*ref*]

  Memory in MB for the flavor

- *vcpus* [*ref*]

  Number of VCPUs for the flavor

- *disk* [*ref*]

  Size of local disk in GB

- *extra_specs* [*ref*]

  Additional arguments for flavor set keys

- *kwargs* [*ref*]

  Optional additional arguments for flavor creation

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFlavors.list_flavors [Scenario]

List all flavors.

Measure the "nova flavor-list" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the flavor listing should contain detailed information

- *kwargs* [*ref*]

  Optional additional arguments for flavor listing

Module: rally.plugins.openstack.scenarios.nova.flavors

NovaFloatingIpsBulk.create_and_delete_floating_ips_bulk [Scenario]

Create nova floating IP by range and delete it.

This scenario creates a floating IP by range and then delete it.

Namespace: openstack

Parameters:

- *start_cidr* [*ref*]

  Floating IP range

- *kwargs* [*ref*]

  Optional additional arguments for range IP creation

Module: rally.plugins.openstack.scenarios.nova.floating_ips_bulk

NovaFloatingIpsBulk.create_and_list_floating_ips_bulk [Scenario]

Create nova floating IP by range and list it.

This scenario creates a floating IP by range and then lists all.

Namespace: openstack

Parameters:

- *start_cidr* [*ref*]

  Floating IP range

- *kwargs* [*ref*]

  Optional additional arguments for range IP creation

Module: rally.plugins.openstack.scenarios.nova.floating_ips_bulk

NovaHosts.list_and_get_hosts [Scenario]

List all nova hosts,and get detailed information fot this hosts.

Measure the "nova host-describe" command performance.

Namespace: openstack

Parameters:

- *zone* [*ref*]

  List nova hosts in an availability-zone. None (default value) means list hosts in all availability-zones

Module: rally.plugins.openstack.scenarios.nova.hosts

NovaHosts.list_hosts [Scenario]

List all nova hosts.

Measure the "nova host-list" command performance.

Namespace: openstack

Parameters:

- *zone* [*ref*]

  List nova hosts in an availability-zone. None (default value) means list hosts in all availability-zones

Module: rally.plugins.openstack.scenarios.nova.hosts

NovaHypervisors.list_and_get_hypervisors [Scenario]

List and Get hypervisors.

The scenario first lists all hypervisors, then get detailed information of the listed hypervisors in turn.

Measure the "nova hypervisor-show" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the hypervisor listing should contain detailed information about all of them

Module: rally.plugins.openstack.scenarios.nova.hypervisors

NovaHypervisors.list_and_get_uptime_hypervisors [Scenario]

List hypervisors,then display the uptime of it.

The scenario first list all hypervisors,then display the uptime of the listed hypervisors in turn.

Measure the "nova hypervisor-uptime" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the hypervisor listing should contain detailed information about all of them

Module: rally.plugins.openstack.scenarios.nova.hypervisors

NovaHypervisors.list_and_search_hypervisors [Scenario]

List all servers belonging to specific hypervisor.

The scenario first list all hypervisors,then find its hostname, then list all servers belonging to the hypervisor

Measure the "nova hypervisor-servers <hostname>" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the hypervisor listing should contain detailed information about all of them

Module: rally.plugins.openstack.scenarios.nova.hypervisors

NovaHypervisors.list_hypervisors [Scenario]

List hypervisors.

Measure the "nova hypervisor-list" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the hypervisor listing should contain detailed information about all of them

Module: rally.plugins.openstack.scenarios.nova.hypervisors

NovaHypervisors.statistics_hypervisors [Scenario]

Get hypervisor statistics over all compute nodes.

Measure the "nova hypervisor-stats" command performance.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.nova.hypervisors

NovaImages.list_images [Scenario]

List all images.

Measure the "nova image-list" command performance.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if the image listing should contain detailed information

- *kwargs* [*ref*]

  Optional additional arguments for image listing

Module: rally.plugins.openstack.scenarios.nova.images

NovaKeypair.boot_and_delete_server_with_keypair [Scenario]

Boot and delete server with keypair.

Plan of this scenario:

- create a keypair
- boot a VM with created keypair
- delete server
- delete keypair

Namespace: openstack

Parameters:

- *image* [*ref*]

  ID of the image to be used for server creation

- *flavor* [*ref*]

  ID of the flavor to be used for server creation

- *boot_server_kwargs* [*ref*]

  Optional additional arguments for VM creation

- *server_kwargs* [*ref*]

  Deprecated alias for boot_server_kwargs

- *kwargs* [*ref*]

  Optional additional arguments for keypair creation

Module: rally.plugins.openstack.scenarios.nova.keypairs

NovaKeypair.create_and_delete_keypair [Scenario]

Create a keypair with random name and delete keypair.

This scenario creates a keypair and then delete that keypair.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional additional arguments for keypair creation

Module: rally.plugins.openstack.scenarios.nova.keypairs

NovaKeypair.create_and_get_keypair [Scenario]

Create a keypair and get the keypair details.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional additional arguments for keypair creation

Module: rally.plugins.openstack.scenarios.nova.keypairs

NovaKeypair.create_and_list_keypairs [Scenario]

Create a keypair with random name and list keypairs.

This scenario creates a keypair and then lists all keypairs.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Optional additional arguments for keypair creation

Module: rally.plugins.openstack.scenarios.nova.keypairs

NovaNetworks.create_and_delete_network [Scenario]

Create nova network and delete it.

Namespace: openstack

Parameters:

- *start_cidr* [*ref*]

  IP range

- *kwargs* [*ref*]

  Optional additional arguments for network creation

Module: rally.plugins.openstack.scenarios.nova.networks

NovaNetworks.create_and_list_networks [Scenario]

Create nova network and list all networks.

Namespace: openstack

Parameters:

- *start_cidr* [*ref*]

  IP range

- *kwargs* [*ref*]

  Optional additional arguments for network creation

Module: rally.plugins.openstack.scenarios.nova.networks

NovaSecGroup.boot_and_delete_server_with_secgroups [Scenario]

Boot and delete server with security groups attached.

Plan of this scenario:

- create N security groups with M rules per group vm with security groups)
- boot a VM with created security groups
- get list of attached security groups to server
- delete server
- delete all security groups
- check that all groups were attached to server

Namespace: openstack

Parameters:

- *image* [*ref*]

  ID of the image to be used for server creation

- *flavor* [*ref*]

  ID of the flavor to be used for server creation

- *security_group_count* [*ref*]

  Number of security groups

- *rules_per_security_group* [*ref*]

  Number of rules per security group

- ****kwargs* [*ref*]

  Optional arguments for booting the instance

Module: rally.plugins.openstack.scenarios.nova.security_group

NovaSecGroup.boot_server_and_add_secgroups [Scenario]

Boot a server and add a security group to it.

Plan of this scenario:

- create N security groups with M rules per group
- boot a VM
- add security groups to VM

Namespace: openstack

Parameters:

- *image* [*ref*]

  ID of the image to be used for server creation

- *flavor* [*ref*]

  ID of the flavor to be used for server creation

- *security_group_count* [*ref*]

  Number of security groups

- *rules_per_security_group* [*ref*]

  Number of rules per security group

- ****kwargs* [*ref*]

  Optional arguments for booting the instance

Module: rally.plugins.openstack.scenarios.nova.security_group

NovaSecGroup.create_and_delete_secgroups [Scenario]

Create and delete security groups.

This scenario creates N security groups with M rules per group and then deletes them.

Namespace: openstack

Parameters:

- *security_group_count* [*ref*]

  Number of security groups

- *rules_per_security_group* [*ref*]

  Number of rules per security group

Module: rally.plugins.openstack.scenarios.nova.security_group

NovaSecGroup.create_and_list_secgroups [Scenario]

Create and list security groups.

This scenario creates N security groups with M rules per group and then lists them.

Namespace: openstack

Parameters:

- *security_group_count* [*ref*]

  Number of security groups

- *rules_per_security_group* [*ref*]

  Number of rules per security group

Module: rally.plugins.openstack.scenarios.nova.security_group

NovaSecGroup.create_and_update_secgroups [Scenario]

Create and update security groups.

This scenario creates 'security_group_count' security groups then updates their name and description.

Namespace: openstack

Parameters:

- *security_group_count* [*ref*]

  Number of security groups

Module: rally.plugins.openstack.scenarios.nova.security_group

NovaServerGroups.create_and_list_server_groups [Scenario]

Create a server group, then list all server groups.

Measure the "nova server-group-create" and "nova server-group-list" command performance.

Namespace: openstack

Parameters:

- *all_projects* [*ref*]

  If True, display server groups from all projects(Admin only)

- *kwargs* [*ref*]

  Server group name and policy

Module: rally.plugins.openstack.scenarios.nova.server_groups

NovaServers.boot_and_associate_floating_ip [Scenario]

Boot a server and associate a floating IP to it.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_bounce_server [Scenario]

Boot a server and run specified actions against it.

Actions should be passed into the actions parameter. Available actions are 'hard_reboot', 'soft_reboot', 'stop_start', 'rescue_unrescue', 'pause_unpause', 'suspend_resume', 'lock_unlock' and 'shelve_unshelve'. Delete server after all actions were completed.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *actions* [*ref*]

  List of action dictionaries, where each action dictionary speicifes an action to be performed in the following format: {"action_name": <no_of_iterations>}

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_delete_multiple_servers [Scenario]

Boot multiple servers in a single request and delete them.

Deletion is done in parallel with one request per server, not with a single request for all servers.

Namespace: openstack

Parameters:

- *image* [*ref*]

  The image to boot from

- *flavor* [*ref*]

  Flavor used to boot instance

- *count* [*ref*]

  Number of instances to boot

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for instance creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_delete_server [Scenario]

Boot and delete a server.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between volume creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_get_console_output [Scenario]

Get text console output from server.

This simple scenario tests the nova console-log command by retrieving the text console log output.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *length* [*ref*]

  The number of tail log lines you would like to retrieve. None (default value) or -1 means unlimited length.

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Returns: Text console log output for server

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_list_server [Scenario]

Boot a server from an image and then list all servers.

Measure the "nova list" command performance.

If you have only 1 user in your context, you will add 1 server on every iteration. So you will have more and more servers and will be able to measure the performance of the "nova list" command depending on the number of servers owned by users.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *detailed* [*ref*]

  True if the server listing should contain detailed information about all of them

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_live_migrate_server [Scenario]

Live Migrate a server.

This scenario launches a VM on a compute node available in the availability zone and then migrates the VM to another compute node on the same availability zone.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between VM booting and running live migration (of random duration from range [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *block_migration* [*ref*]

  Specifies the migration type

- *disk_over_commit* [*ref*]

  Specifies whether to allow overcommit on migrated instance or not

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_migrate_server [Scenario]

Migrate a server.

This scenario launches a VM on a compute node available in the availability zone, and then migrates the VM to another compute node on the same availability zone.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_rebuild_server [Scenario]

Rebuild a server.

This scenario launches a VM, then rebuilds that VM with a different image.

Namespace: openstack

Parameters:

- *from_image* [*ref*]

  Image to be used to boot an instance

- *to_image* [*ref*]

  Image to be used to rebuild the instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_show_server [Scenario]

Show server details.

This simple scenario tests the nova show command by retrieving the server details.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Returns: Server details

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_and_update_server [Scenario]

Boot a server, then update its name and description.

The scenario first creates a server, then update it. Assumes that cleanup is done elsewhere.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *description* [*ref*]

  Update the server description

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_lock_unlock_and_delete [Scenario]

Boot a server, lock it, then unlock and delete it.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between locking and unlocking the server (of random duration from min_sleep to max_sleep).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *min_sleep* [*ref*]

  Minimum sleep time between locking and unlocking in seconds

- *max_sleep* [*ref*]

  Maximum sleep time between locking and unlocking in seconds

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server [Scenario]

Boot a server.

Assumes that cleanup is done elsewhere.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *auto_assign_nic* [*ref*]

  True if NICs should be assigned

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_associate_and_dissociate_floating_ip [Scenario]

Boot a server associate and dissociate a floating IP from it.

The scenario first boot a server and create a floating IP. then associate the floating IP to the server.Finally dissociate the floating IP.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_attach_created_volume_and_live_migrate [Scenario]

Create a VM, attach a volume to it and live migrate.

Simple test to create a VM and attach a volume, then migrate the VM, detach the volume and delete volume/VM.

Optional ‘min_sleep’ and ‘max_sleep’ parameters allow the scenario to simulate a pause between attaching a volume and running live migration (of random duration from range [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Glance image name to use for the VM

- *flavor* [*ref*]

  VM flavor name

- *size* [*ref*]

  Volume size (in GB)

- *block_migration* [*ref*]

  Specifies the migration type

- *disk_over_commit* [*ref*]

  Specifies whether to allow overcommit on migrated instance or not

- *boot_server_kwargs* [*ref*]

  Optional arguments for VM creation

- *create_volume_kwargs* [*ref*]

  Optional arguments for volume creation

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_attach_created_volume_and_resize [Scenario]

Create a VM from image, attach a volume to it and resize.

Simple test to create a VM and attach a volume, then resize the VM, detach the volume then delete volume and VM. Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between attaching a volume and running resize (of random duration from range [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Glance image name to use for the VM

- *flavor* [*ref*]

  VM flavor name

- *to_flavor* [*ref*]

  Flavor to be used to resize the booted instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *force_delete* [*ref*]

  True if force_delete should be used

- *confirm* [*ref*]

  True if need to confirm resize else revert resize

- *do_delete* [*ref*]

  True if resources needs to be deleted explicitly else use rally cleanup to remove resources

- *boot_server_kwargs* [*ref*]

  Optional arguments for VM creation

- *create_volume_kwargs* [*ref*]

  Optional arguments for volume creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_from_volume [Scenario]

Boot a server from volume.

The scenario first creates a volume and then a server. Assumes that cleanup is done elsewhere.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *volume_type* [*ref*]

  Specifies volume type when there are multiple backends

- *auto_assign_nic* [*ref*]

  True if NICs should be assigned

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_from_volume_and_delete [Scenario]

Boot a server from volume and then delete it.

The scenario first creates a volume and then a server. Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between volume creation and deletion (of random duration from [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *volume_type* [*ref*]

  Specifies volume type when there are multiple backends

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_from_volume_and_live_migrate [Scenario]

Boot a server from volume and then migrate it.

The scenario first creates a volume and a server booted from the volume on a compute node available in the availability zone and then migrates the VM to another compute node on the same availability zone.

Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between VM booting and running live migration (of random duration from range [min_sleep, max_sleep]).

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *volume_type* [*ref*]

  Specifies volume type when there are multiple backends

- *block_migration* [*ref*]

  Specifies the migration type

- *disk_over_commit* [*ref*]

  Specifies whether to allow overcommit on migrated instance or not

- *force_delete* [*ref*]

  True if force_delete should be used

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_from_volume_and_resize [Scenario]

Boot a server from volume, then resize and delete it.

The scenario first creates a volume and then a server. Optional 'min_sleep' and 'max_sleep' parameters allow the scenario to simulate a pause between volume creation and deletion (of random duration from [min_sleep, max_sleep]).

This test will confirm the resize by default, or revert the resize if confirm is set to false.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *to_flavor* [*ref*]

  Flavor to be used to resize the booted instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *min_sleep* [*ref*]

  Minimum sleep time in seconds (non-negative)

- *max_sleep* [*ref*]

  Maximum sleep time in seconds (non-negative)

- *force_delete* [*ref*]

  True if force_delete should be used

- *confirm* [*ref*]

  True if need to confirm resize else revert resize

- *do_delete* [*ref*]

  True if resources needs to be deleted explicitly else use rally cleanup to remove resources

- *boot_server_kwargs* [*ref*]

  Optional arguments for VM creation

- *create_volume_kwargs* [*ref*]

  Optional arguments for volume creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.boot_server_from_volume_snapshot [Scenario]

Boot a server from a snapshot.

The scenario first creates a volume and creates a snapshot from this volume, then boots a server from the created snapshot. Assumes that cleanup is done elsewhere.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *volume_size* [*ref*]

  Volume size (in GB)

- *volume_type* [*ref*]

  Specifies volume type when there are multiple backends

- *auto_assign_nic* [*ref*]

  True if NICs should be assigned

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.list_servers [Scenario]

List all servers.

This simple scenario test the nova list command by listing all the servers.

Namespace: openstack

Parameters:

- *detailed* [*ref*]

  True if detailed information about servers should be listed

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.pause_and_unpause_server [Scenario]

Create a server, pause, unpause and then delete it

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.resize_server [Scenario]

Boot a server, then resize and delete it.

This test will confirm the resize by default, or revert the resize if confirm is set to false.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *to_flavor* [*ref*]

  Flavor to be used to resize the booted instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.resize_shutoff_server [Scenario]

Boot a server and stop it, then resize and delete it.

This test will confirm the resize by default, or revert the resize if confirm is set to false.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *to_flavor* [*ref*]

  Flavor to be used to resize the booted instance

- *confirm* [*ref*]

  True if need to confirm resize else revert resize

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.shelve_and_unshelve_server [Scenario]

Create a server, shelve, unshelve and then delete it

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.snapshot_server [Scenario]

Boot a server, make its snapshot and delete both.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServers.suspend_and_resume_server [Scenario]

Create a server, suspend, resume and then delete it

Namespace: openstack

Parameters:

- *image* [*ref*]

  Image to be used to boot an instance

- *flavor* [*ref*]

  Flavor to be used to boot an instance

- *force_delete* [*ref*]

  True if force_delete should be used

- *kwargs* [*ref*]

  Optional additional arguments for server creation

Module: rally.plugins.openstack.scenarios.nova.servers

NovaServices.list_services [Scenario]

List all nova services.

Measure the "nova service-list" command performance.

Namespace: openstack

Parameters:

- *host* [*ref*]

  List nova services on host

- *binary* [*ref*]

  List nova services matching given binary

Module: rally.plugins.openstack.scenarios.nova.services

Quotas.cinder_get [Scenario]

Get quotas for Cinder.

Measure the "cinder quota-show" command performance

Namespace: openstack

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.cinder_update [Scenario]

Update quotas for Cinder.

Namespace: openstack

Parameters:

- *max_quota* [*ref*]

  Max value to be updated for quota.

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.cinder_update_and_delete [Scenario]

Update and Delete quotas for Cinder.

Namespace: openstack

Parameters:

- *max_quota* [*ref*]

  Max value to be updated for quota.

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.neutron_update [Scenario]

Update quotas for neutron.

Namespace: openstack

Parameters:

- *max_quota* [*ref*]

  Max value to be updated for quota.

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.nova_get [Scenario]

Get quotas for nova.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.nova_update [Scenario]

Update quotas for Nova.

Namespace: openstack

Parameters:

- *max_quota* [*ref*]

  Max value to be updated for quota.

Module: rally.plugins.openstack.scenarios.quotas.quotas

Quotas.nova_update_and_delete [Scenario]

Update and delete quotas for Nova.

Namespace: openstack

Parameters:

- *max_quota* [*ref*]

  Max value to be updated for quota.

Module: rally.plugins.openstack.scenarios.quotas.quotas

SaharaClusters.create_and_delete_cluster [Scenario]

Launch and delete a Sahara Cluster.

This scenario launches a Hadoop cluster, waits until it becomes 'Active' and deletes it.

Namespace: openstack

Parameters:

- *flavor* [*ref*]

  Nova flavor that will be for nodes in the created node groups. Deprecated.

- *master_flavor* [*ref*]

  Nova flavor that will be used for the master instance of the cluster

- *worker_flavor* [*ref*]

  Nova flavor that will be used for the workers of the cluster

- *workers_count* [*ref*]

  Number of worker instances in a cluster

- *plugin_name* [*ref*]

  Name of a provisioning plugin

- *hadoop_version* [*ref*]

  Version of Hadoop distribution supported by the specified plugin.

- *floating_ip_pool* [*ref*]

  Floating ip pool name from which Floating IPs will be allocated. Sahara will determine automatically how to treat this depending on its own configurations. Defaults to None because in some cases Sahara may work w/o Floating IPs.

- *volumes_per_node* [*ref*]

  Number of Cinder volumes that will be attached to every cluster node

- *volumes_size* [*ref*]

  Size of each Cinder volume in GB

- *auto_security_group* [*ref*]

  Boolean value. If set to True Sahara will create a Security Group for each Node Group in the Cluster automatically.

- *security_groups* [*ref*]

  List of security groups that will be used while creating VMs. If auto_security_group is set to True, this list can be left empty.

- *node_configs* [*ref*]

  Config dict that will be passed to each Node Group

- *cluster_configs* [*ref*]

  Config dict that will be passed to the Cluster

- *enable_anti_affinity* [*ref*]

  If set to true the vms will be scheduled one per compute node.

- *enable_proxy* [*ref*]

  Use Master Node of a Cluster as a Proxy node and do not assign floating ips to workers.

- *use_autoconfig* [*ref*]

  If True, instances of the node group will be automatically configured during cluster creation. If False, the configuration values should be specify manually

Module: rally.plugins.openstack.scenarios.sahara.clusters

SaharaClusters.create_scale_delete_cluster [Scenario]

Launch, scale and delete a Sahara Cluster.

This scenario launches a Hadoop cluster, waits until it becomes 'Active'. Then a series of scale operations is applied. The scaling happens according to numbers listed in

Namespace: openstack

Parameters:

- *flavor* [*ref*]

  Nova flavor that will be for nodes in the created node groups. Deprecated.

- *master_flavor* [*ref*]

  Nova flavor that will be used for the master instance of the cluster

- *worker_flavor* [*ref*]

  Nova flavor that will be used for the workers of the cluster

- *workers_count* [*ref*]

  Number of worker instances in a cluster

- *plugin_name* [*ref*]

  Name of a provisioning plugin

- *hadoop_version* [*ref*]

  Version of Hadoop distribution supported by the specified plugin.

- *deltas* [*ref*]

  List of integers which will be used to add or remove worker nodes from the cluster

- *floating_ip_pool* [*ref*]

  Floating ip pool name from which Floating IPs will be allocated. Sahara will determine automatically how to treat this depending on its own configurations. Defaults to None because in some cases Sahara may work w/o Floating IPs.

- *neutron_net_id* [*ref*]

  Id of a Neutron network that will be used for fixed IPs. This parameter is ignored when Nova Network is set up.

- *volumes_per_node* [*ref*]

  Number of Cinder volumes that will be attached to every cluster node

- *volumes_size* [*ref*]

  Size of each Cinder volume in GB

- *auto_security_group* [*ref*]

  Boolean value. If set to True Sahara will create a Security Group for each Node Group in the Cluster automatically.

- *security_groups* [*ref*]

  List of security groups that will be used while creating VMs. If auto_security_group is set to True this list can be left empty.

- *node_configs* [*ref*]

  Configs dict that will be passed to each Node Group

- *cluster_configs* [*ref*]

  Configs dict that will be passed to the Cluster

- *enable_anti_affinity* [*ref*]

  If set to true the vms will be scheduled one per compute node.

- *enable_proxy* [*ref*]

  Use Master Node of a Cluster as a Proxy node and do not assign floating ips to workers.

- *use_autoconfig* [*ref*]

  If True, instances of the node group will be automatically configured during cluster creation. If False, the configuration values should be specify manually

Module: rally.plugins.openstack.scenarios.sahara.clusters

SaharaJob.create_launch_job [Scenario]

Create and execute a Sahara EDP Job.

This scenario Creates a Job entity and launches an execution on a Cluster.

Namespace: openstack

Parameters:

- *job_type* [*ref*]

  Type of the Data Processing Job

- *configs* [*ref*]

  Config dict that will be passed to a Job Execution

- *job_idx* [*ref*]

  Index of a job in a sequence. This index will be used to create different atomic actions for each job in a sequence

Module: rally.plugins.openstack.scenarios.sahara.jobs

SaharaJob.create_launch_job_sequence [Scenario]

Create and execute a sequence of the Sahara EDP Jobs.

This scenario Creates a Job entity and launches an execution on a Cluster for every job object provided.

Namespace: openstack

Parameters:

- *jobs* [*ref*]

  List of jobs that should be executed in one context

Module: rally.plugins.openstack.scenarios.sahara.jobs

SaharaJob.create_launch_job_sequence_with_scaling [Scenario]

Create and execute Sahara EDP Jobs on a scaling Cluster.

This scenario Creates a Job entity and launches an execution on a Cluster for every job object provided. The Cluster is scaled according to the deltas values and the sequence is launched again.

Namespace: openstack

Parameters:

- *jobs* [*ref*]

  List of jobs that should be executed in one context

- *deltas* [*ref*]

  List of integers which will be used to add or remove worker nodes from the cluster

Module: rally.plugins.openstack.scenarios.sahara.jobs

SaharaNodeGroupTemplates.create_and_list_node_group_templates [Scenario]

Create and list Sahara Node Group Templates.

This scenario creates two Node Group Templates with different set of node processes. The master Node Group Template contains Hadoop's management processes. The worker Node Group Template contains Hadoop's worker processes.

By default the templates are created for the vanilla Hadoop provisioning plugin using the version 1.2.1

After the templates are created the list operation is called.

Namespace: openstack

Parameters:

- *flavor* [*ref*]

  Nova flavor that will be for nodes in the created node groups

- *plugin_name* [*ref*]

  Name of a provisioning plugin

- *hadoop_version* [*ref*]

  Version of Hadoop distribution supported by the specified plugin.

- *use_autoconfig* [*ref*]

  If True, instances of the node group will be automatically configured during cluster creation. If False, the configuration values should be specify manually

Module: rally.plugins.openstack.scenarios.sahara.node_group_templates

SaharaNodeGroupTemplates.create_delete_node_group_templates [Scenario]

Create and delete Sahara Node Group Templates.

This scenario creates and deletes two most common types of Node Group Templates.

By default the templates are created for the vanilla Hadoop provisioning plugin using the version 1.2.1

Namespace: openstack

Parameters:

- *flavor* [*ref*]

  Nova flavor that will be for nodes in the created node groups

- *plugin_name* [*ref*]

  Name of a provisioning plugin

- *hadoop_version* [*ref*]

  Version of Hadoop distribution supported by the specified plugin.

- *use_autoconfig* [*ref*]

  If True, instances of the node group will be automatically configured during cluster creation. If False, the configuration values should be specify manually

Module: rally.plugins.openstack.scenarios.sahara.node_group_templates

SenlinClusters.create_and_delete_cluster [Scenario]

Create a cluster and then delete it.

Measure the "senlin cluster-create" and "senlin cluster-delete" commands performance.

Namespace: openstack

Parameters:

- *desired_capacity* [*ref*]

  The capacity or initial number of nodes owned by the cluster

- *min_size* [*ref*]

  The minimum number of nodes owned by the cluster

- *max_size* [*ref*]

  The maximum number of nodes owned by the cluster. -1 means no limit

- *timeout* [*ref*]

  The timeout value in seconds for cluster creation

- *metadata* [*ref*]

  A set of key value pairs to associate with the cluster

Module: rally.plugins.openstack.scenarios.senlin.clusters

SwiftObjects.create_container_and_object_then_delete_all [Scenario]

Create container and objects then delete everything created.

Namespace: openstack

Parameters:

- *objects_per_container* [*ref*]

  Int, number of objects to upload

- *object_size* [*ref*]

  Int, temporary local object size

- *kwargs* [*ref*]

  Dict, optional parameters to create container

Module: rally.plugins.openstack.scenarios.swift.objects

SwiftObjects.create_container_and_object_then_download_object [Scenario]

Create container and objects then download all objects.

Namespace: openstack

Parameters:

- *objects_per_container* [*ref*]

  Int, number of objects to upload

- *object_size* [*ref*]

  Int, temporary local object size

- *kwargs* [*ref*]

  Dict, optional parameters to create container

Module: rally.plugins.openstack.scenarios.swift.objects

SwiftObjects.create_container_and_object_then_list_objects [Scenario]

Create container and objects then list all objects.

Namespace: openstack

Parameters:

- *objects_per_container* [*ref*]

  Int, number of objects to upload

- *object_size* [*ref*]

  Int, temporary local object size

- *kwargs* [*ref*]

  Dict, optional parameters to create container

Module: rally.plugins.openstack.scenarios.swift.objects

SwiftObjects.list_and_download_objects_in_containers [Scenario]

List and download objects in all containers.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.swift.objects

SwiftObjects.list_objects_in_containers [Scenario]

List objects in all containers.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.swift.objects

VMTasks.boot_runcommand_delete [Scenario]

Boot a server, run script specified in command and delete server.

Namespace: openstack

Parameters:

- *image* [*ref*]

  Glance image name to use for the vm. Optional in case of specified "image_command_customizer" context

- *flavor* [*ref*]

  VM flavor name

- *username* [*ref*]

  Ssh username on server, str

- *password* [*ref*]

  Password on SSH authentication

- *command* [*ref*]

  Command-specifying dictionary that either specifies remote command path via *remote_path' (can be uploaded from a local file specified by 'local_path*), an inline script via 'script_inline' or a local script file path using 'script_file'. Both 'script_file' and 'local_path' are checked to be accessible by the 'file_exists' validator code.

  The 'script_inline' and 'script_file' both require an 'interpreter' value to specify the interpreter script should be run with.

  Note that any of 'interpreter' and 'remote_path' can be an array prefixed with environment variables and suffixed with args for the 'interpreter' command. 'remote_path's last component must be a path to a command to execute (also upload destination if a 'local_path' is given). Uploading an interpreter is possible but requires that 'remote_path' and 'interpreter' path do match.

  Examples:

```
# Run a `local_script.pl' file sending it to a remote
# Perl interpreter
command = {
    "script_file": "local_script.pl",
    "interpreter": "/usr/bin/perl"
}

# Run an inline script sending it to a remote interpreter
command = {
    "script_inline": "echo 'Hello, World!'",
    "interpreter": "/bin/sh"
}

# Run a remote command
```

```
command = {
    "remote_path": "/bin/false"
}

# Copy a local command and run it
command = {
    "remote_path": "/usr/local/bin/fio",
    "local_path": "/home/foobar/myfiodir/bin/fio"
}

# Copy a local command and run it with environment variable
command = {
    "remote_path": ["HOME=/root", "/usr/local/bin/fio"],
    "local_path": "/home/foobar/myfiodir/bin/fio"
}

# Run an inline script sending it to a remote interpreter
command = {
    "script_inline": "echo "Hello, ${NAME:-World}"",
    "interpreter": ["NAME=Earth", "/bin/sh"]
}

# Run an inline script sending it to an uploaded remote
# interpreter
command = {
    "script_inline": "echo "Hello, ${NAME:-World}"",
    "interpreter": ["NAME=Earth", "/tmp/sh"],
    "remote_path": "/tmp/sh",
    "local_path": "/home/user/work/cve/sh-1.0/bin/sh"
}
```

- *volume_args* [*ref*]

  Volume args for booting server from volume

- *floating_network* [*ref*]

  External network name, for floating ip

- *port* [*ref*]

  Ssh port for SSH connection

- *use_floating_ip* [*ref*]

  Bool, floating or fixed IP for SSH connection

- *force_delete* [*ref*]

  Whether to use force_delete for servers

- *wait_for_ping* [*ref*]

  Whether to check connectivity on server creation

- ****kwargs* [*ref*]

  Extra arguments for booting the server

- *max_log_length* [*ref*]

  The number of tail nova console-log lines user would like to retrieve

Returns: dictionary with keys ‘data’ and ‘errors’: data: dict, JSON output from the script errors: str, raw data from the script’s stderr stream

Module: rally.plugins.openstack.scenarios.vm.vmtasks

VMTasks.dd_load_test [Scenario]

Boot a server from a custom image, run a command that outputs JSON.

Example Script in rally-jobs/extra/install_benchmark.sh

Namespace: openstack

Parameters:

- *command* [*ref*]

  Default parameter from scenario

Module: rally.plugins.openstack.scenarios.vm.vmtasks

VMTasks.runcommand_heat [Scenario]

Run workload on stack deployed by heat.

Workload can be either file or resource:

```
{"file": "/path/to/file.sh"}
{"resource": ["package.module", "workload.py"]}
```

Also it should contain “username” key.

Given file will be uploaded to *gate_node* and started. This script should print *key value* pairs separated by colon. These pairs will be presented in results.

Gate node should be accessible via ssh with keypair *key_name*, so heat template should accept parameter *key_name*.

Namespace: openstack

Parameters:

- *workload* [*ref*]

  Workload to run

- *template* [*ref*]

  Path to heat template file

- *files* [*ref*]

  Additional template files

- *parameters* [*ref*]

  Parameters for heat template

Module: rally.plugins.openstack.scenarios.vm.vmtasks

Watcher.create_audit_and_delete [Scenario]

Create and delete audit.

Create Audit, wait until whether Audit is in SUCCEEDED state or in FAILED and delete audit.

Namespace: openstack

Module: rally.plugins.openstack.scenarios.watcher.basic

Watcher.create_audit_template_and_delete [Scenario]

Create audit template and delete it.

Namespace: openstack

Parameters:

- *goal* [*ref*]

  The goal audit template is based on

- *strategy* [*ref*]

  The strategy used to provide resource optimization algorithm

Module: rally.plugins.openstack.scenarios.watcher.basic

Watcher.list_audit_templates [Scenario]

List existing audit templates.

Audit templates are being created by Audit Template Context.

Namespace: openstack

Parameters:

- *name* [*ref*]

  Name of the audit template

- *goal* [*ref*]

  Name of the goal

- *strategy* [*ref*]

  Name of the strategy

- *limit* [*ref*]

  The maximum number of results to return per request, if:

  1. limit > 0, the maximum number of audit templates to return.
  2. limit == 0, return the entire list of audit_templates.
  3. limit param is NOT specified (None), the number of items returned respect the maximum imposed by the Watcher API

     (see Watcher's api.max_limit option).

- *sort_key* [*ref*]

  Optional, field used for sorting.

- *sort_dir* [*ref*]

  Optional, direction of sorting, either 'asc' (the default) or 'desc'.

- *detail* [*ref*]

  Optional, boolean whether to return detailed information about audit_templates.

Module: rally.plugins.openstack.scenarios.watcher.basic

ZaqarBasic.create_queue [Scenario]

Create a Zaqar queue with a random name.

Namespace: openstack

Parameters:

- *kwargs* [*ref*]

  Other optional parameters to create queues like "metadata"

Module: rally.plugins.openstack.scenarios.zaqar.basic

ZaqarBasic.producer_consumer [Scenario]

Serial message producer/consumer.

Creates a Zaqar queue with random name, sends a set of messages and then retrieves an iterator containing those.

Namespace: openstack

Parameters:

- *min_msg_count* [*ref*]

  Min number of messages to be posted

- *max_msg_count* [*ref*]

  Max number of messages to be posted

- *kwargs* [*ref*]

  Other optional parameters to create queues like "metadata"

Module: rally.plugins.openstack.scenarios.zaqar.basic

Scenario Runners

constant [Scenario Runner]

Creates constant load executing a scenario a specified number of times.

This runner will place a constant load on the cloud under test by executing each scenario iteration without pausing between iterations up to the number of times specified in the scenario config.

The concurrency parameter of the scenario config controls the number of concurrent iterations which execute during a single scenario in order to simulate the activities of multiple users placing load on the cloud under test.

Namespace: default

Parameters:

- *max_cpu_count (int)* [*ref*]

  The maximum number of processes to create load from.

  Min value: 1.

- *type (str)* [*ref*]

  Type of Runner.

- *timeout (float)* [*ref*]

  Operation's timeout.

- *concurrency (int)* [*ref*]

  The number of parallel iteration executions.

  Min value: 1.

- *times (int)* [*ref*]

  Total number of iteration executions.

  Min value: 1.

Module: rally.plugins.common.runners.constant

constant_for_duration [Scenario Runner]

Creates constant load executing a scenario for an interval of time.

This runner will place a constant load on the cloud under test by executing each scenario iteration without pausing between iterations until a specified interval of time has elapsed.

The concurrency parameter of the scenario config controls the number of concurrent iterations which execute during a single sceanario in order to simulate the activities of multiple users placing load on the cloud under test.

Namespace: default

Parameters:

- *duration (float)* [*ref*]

  The number of seconds during which to generate a load.

  Min value: 0.0.

- *type (str)* [*ref*]

  Type of Runner.

- *timeout (float)* [*ref*]

  Operation's timeout.

  Min value: 1.

- *concurrency (int)* [*ref*]

  The number of parallel iteration executions.

  Min value: 1.

Module: rally.plugins.common.runners.constant

rps [Scenario Runner]

Scenario runner that does the job with specified frequency.

Every single benchmark scenario iteration is executed with specified frequency (runs per second) in a pool of processes. The scenario will be launched for a fixed number of times in total (specified in the config).

An example of a rps scenario is booting 1 VM per second. This execution type is thus very helpful in understanding the maximal load that a certain cloud can handle.

Namespace: default

Parameters:

- *rps* [*ref*]
- *times (int)* [*ref*]

  Min value: 1.
- *timeout (float)* [*ref*]
- *max_cpu_count (int)* [*ref*]

  Min value: 1.
- *max_concurrency (int)* [*ref*]

  Min value: 1.
- *type (str)* [*ref*]

Module: rally.plugins.common.runners.rps

serial [Scenario Runner]

Scenario runner that executes benchmark scenarios serially.

Unlike scenario runners that execute in parallel, the serial scenario runner executes scenarios one-by-one in the same python interpreter process as Rally. This allows you to benchmark your scenario without introducing any concurrent operations as well as interactively debug the scenario from the same command that you use to start Rally.

Namespace: default

Parameters:

- *type (str)* [*ref*]
- *times (int)* [*ref*]

  Min value: 1.

Module: rally.plugins.common.runners.serial

Triggers

event [Trigger]

Triggers hook on specified event and list of values.

Namespace: default

Note: One of the following groups of parameters should be provided.

Option 1 of parameters:

Triage hook based on specified seconds after start of workload.

- *at (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: int.

- *unit* [*ref*]

  Set of expected values: 'time'.

Option 2 of parameters:

Triage hook based on specific iterations.

- *at (list)* [*ref*]

  Elements of the list should follow format(s) described below:

  - Type: int.

- *unit* [*ref*]

  Set of expected values: 'iteration'.

Module: rally.plugins.common.trigger.event

periodic [Trigger]

Periodically triggers hook with specified range and step.

Namespace: default

Note: One of the following groups of parameters should be provided.

Option 1 of parameters:

Periodically triage hook based on elapsed time after start of workload.

- *start (int)* [*ref*]

  Min value: 0.

- *step (int)* [*ref*]

  Min value: 1.

- *end (int)* [*ref*]

  Min value: 1.

- *unit* [*ref*]

  Set of expected values: 'time'.

Option 2 of parameters:

Periodically triage hook based on iterations.

- *start (int)* [*ref*]

  Min value: 1.

- *step (int)* [*ref*]

  Min value: 1.

- *end (int)* [*ref*]

  Min value: 1.

- *unit* [*ref*]

  Set of expected values: 'iteration'.

Module: rally.plugins.common.trigger.periodic

Verification Component

Verification Reporters

html [Verification Reporter]

Generates verification report in HTML format.

Namespace: default

Module: rally.plugins.common.verification.reporters

html-static [Verification Reporter]

Generates verification report in HTML format with embedded JS/CSS.

Namespace: default

Module: rally.plugins.common.verification.reporters

json [Verification Reporter]

Generates verification report in JSON format.

An example of the report (All dates, numbers, names appearing in this example are fictitious. Any resemblance to real things is purely coincidental):

```
{"verifications": {
    "verification-uuid-1": {
        "status": "finished",
        "skipped": 1,
        "started_at": "2001-01-01T00:00:00",
        "finished_at": "2001-01-01T00:05:00",
        "tests_duration": 5,
        "run_args": {
            "pattern": "set=smoke",
            "xfail_list": {"some.test.TestCase.test_xfail":
                           "Some reason why it is expected."},
            "skip_list": {"some.test.TestCase.test_skipped":
                          "This test was skipped intentionally"},
        },
        "success": 1,
        "expected_failures": 1,
        "tests_count": 3,
        "failures": 0,
        "unexpected_success": 0
    },
    "verification-uuid-2": {
        "status": "finished",
        "skipped": 1,
        "started_at": "2002-01-01T00:00:00",
        "finished_at": "2002-01-01T00:05:00",
        "tests_duration": 5,
        "run_args": {
            "pattern": "set=smoke",
            "xfail_list": {"some.test.TestCase.test_xfail":
                           "Some reason why it is expected."},
            "skip_list": {"some.test.TestCase.test_skipped":
                          "This test was skipped intentionally"},
        },
        "success": 1,
        "expected_failures": 1,
        "tests_count": 3,
        "failures": 1,
        "unexpected_success": 0
    }
 },
 "tests": {
    "some.test.TestCase.test_foo[tag1,tag2]": {
        "name": "some.test.TestCase.test_foo",
        "tags": ["tag1","tag2"],
        "by_verification": {
            "verification-uuid-1": {
                "status": "success",
                "duration": "1.111"
            },
            "verification-uuid-2": {
                "status": "success",
                "duration": "22.222"
            }
        }
    },
    "some.test.TestCase.test_skipped[tag1]": {
        "name": "some.test.TestCase.test_skipped",
        "tags": ["tag1"],
```

```
        "by_verification": {
            "verification-uuid-1": {
                "status": "skipped",
                "duration": "0",
                "details": "Skipped until Bug: 666 is resolved."
            },
            "verification-uuid-2": {
                "status": "skipped",
                "duration": "0",
                "details": "Skipped until Bug: 666 is resolved."
            }
        }
    },
    "some.test.TestCase.test_xfail": {
        "name": "some.test.TestCase.test_xfail",
        "tags": [],
        "by_verification": {
            "verification-uuid-1": {
                "status": "xfail",
                "duration": "3",
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
            },
            "verification-uuid-2": {
                "status": "xfail",
                "duration": "3",
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
            }
        }
    },
    "some.test.TestCase.test_failed": {
        "name": "some.test.TestCase.test_failed",
        "tags": [],
        "by_verification": {
            "verification-uuid-2": {
```

```
                "status": "fail",
                "duration": "4",
                "details": "Some reason why it is expected.\n\n"
                    "Traceback (most recent call last): \n"
                    "  File "fake.py", line 13, in <module>\n"
                    "    yyy()\n"
                    "  File "fake.py", line 11, in yyy\n"
                    "    xxx()\n"
                    "  File "fake.py", line 8, in xxx\n"
                    "    bar()\n"
                    "  File "fake.py", line 5, in bar\n"
                    "    foo()\n"
                    "  File "fake.py", line 2, in foo\n"
                    "    raise Exception()\n"
                    "Exception"
                }
            }
        }
    }
}
```

Namespace: default

Module: rally.plugins.common.verification.reporters

junit-xml [Verification Reporter]

Generates verification report in JUnit-XML format.

An example of the report (All dates, numbers, names appearing in this example are fictitious. Any resemblance to real things is purely coincidental):

```
<testsuites>
  <!--Report is generated by Rally 0.8.0 at 2002-01-01T00:00:00-->
  <testsuite id="verification-uuid-1"
             tests="9"
             time="1.111"
             errors="0"
             failures="3"
             skipped="0"
             timestamp="2001-01-01T00:00:00">
    <testcase classname="some.test.TestCase"
              name="test_foo"
              time="8"
              timestamp="2001-01-01T00:01:00" />
    <testcase classname="some.test.TestCase"
              name="test_skipped"
              time="0"
              timestamp="2001-01-01T00:02:00">
      <skipped>Skipped until Bug: 666 is resolved.</skipped>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_xfail"
              time="3"
              timestamp="2001-01-01T00:03:00">
      <!--It is an expected failure due to: something-->
      <!--Traceback:
```

```
HEEELP-->
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_uxsuccess"
              time="3"
              timestamp="2001-01-01T00:04:00">
      <failure>
          It is an unexpected success. The test should fail due to:
          It should fail, I said!
      </failure>
    </testcase>
  </testsuite>
  <testsuite id="verification-uuid-2"
             tests="99"
             time="22.222"
             errors="0"
             failures="33"
             skipped="0"
             timestamp="2002-01-01T00:00:00">
    <testcase classname="some.test.TestCase"
              name="test_foo"
              time="8"
              timestamp="2001-02-01T00:01:00" />
    <testcase classname="some.test.TestCase"
              name="test_failed"
              time="8"
              timestamp="2001-02-01T00:02:00">
      <failure>HEEEEEEELP</failure>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_skipped"
              time="0"
              timestamp="2001-02-01T00:03:00">
      <skipped>Skipped until Bug: 666 is resolved.</skipped>
    </testcase>
    <testcase classname="some.test.TestCase"
              name="test_xfail"
              time="4"
              timestamp="2001-02-01T00:04:00">
      <!--It is an expected failure due to: something-->
      <!--Traceback:
HEEELP-->
    </testcase>
  </testsuite>
</testsuites>
```

Namespace: default

Module: rally.plugins.common.verification.reporters

Verifier Contexts

Verifier Managers

tempest [Verifier Manager]

Tempest verifier.

Description:

Quote from official documentation:

> This is a set of integration tests to be run against a live OpenStack cluster. Tempest has batteries of tests for OpenStack API validation, Scenarios, and other specific tests useful in validating an OpenStack deployment.

Rally supports features listed below:

- *cloning Tempest*: repository and version can be specified
- *installation*: system-wide with checking existence of required packages or in virtual environment
- *configuration*: options are discovered via OpenStack API, but you can override them if you need
- *running*: pre-creating all required resources(i.e images, tenants, etc), prepare arguments, launching Tempest, live-progress output
- *results*: all verifications are stored in db, you can built reports, compare verification at whatever you want time.

Appeared in Rally 0.8.0 *(actually, it appeared long time ago with first revision of Verification Component, but 0.8.0 is mentioned since it is first release after Verification Component redesign)*

Running arguments:

- *concurrency*: Number of processes to be used for launching tests. In case of 0 value, number of processes will be equal to number of CPU cores.
- *load_list*: a list of tests to launch.
- *pattern*: a regular expression of tests to launch.
- *set*: Name of predefined set of tests. Known names: full, smoke, baremetal, clustering, compute, database, data_processing, identity, image, messaging, network, object_storage, orchestration, telemetry, volume, scenario
- *skip_list*: a list of tests to skip (actually, it is a dict where keys are names of tests, values are reasons).
- *xfail_list*: a list of tests that are expected to fail (actually, it is a dict where keys are names of tests, values are reasons).

Installation arguments:

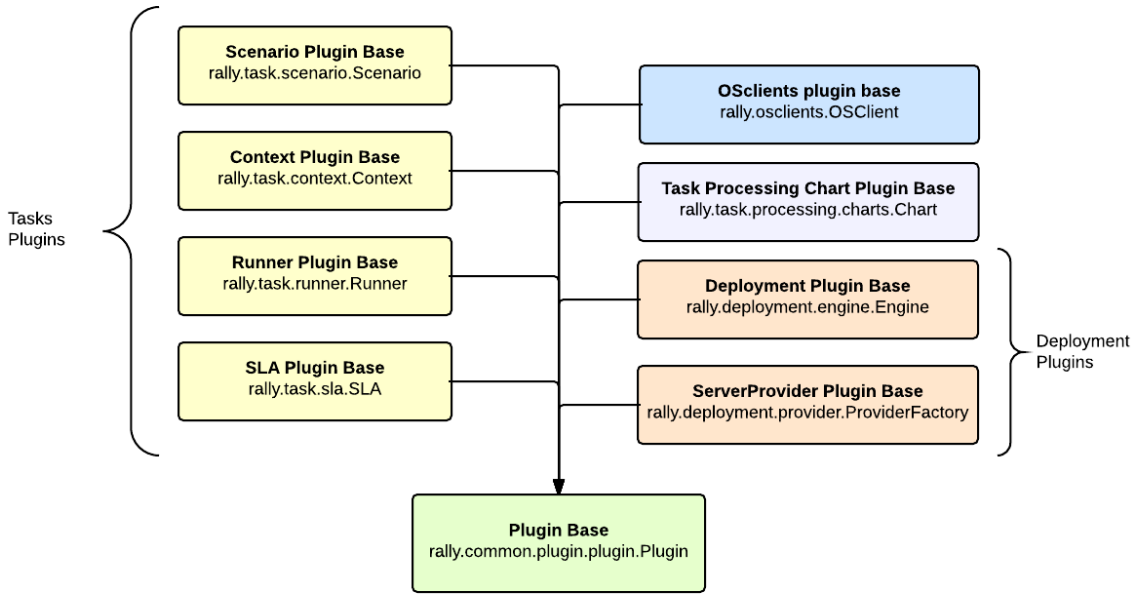
- *system_wide*: Whether or not to use the system-wide environment for verifier instead of a virtual environment. Defaults to False.
- *source*: Path or URL to the repo to clone verifier from. Defaults to https://git.openstack.org/openstack/tempest
- *version*: Branch, tag or commit ID to checkout before verifier installation. Defaults to 'master'.

Namespace: openstack

Module: rally.plugins.openstack.verification.tempest.manager

How plugins work

Rally provides an opportunity to create and use a **custom benchmark scenario, runner, SLA, deployment or context** as a **plugin**:



Placement

Plugins can be quickly written and used, with no need to contribute them to the actual Rally code. Just place a Python module with your plugin class into the `/opt/rally/plugins` or `~/.rally/plugins` directory (or its subdirectories), and it will be automatically loaded. Additional paths can be specified with the `--plugin-paths` argument, or with the `RALLY_PLUGIN_PATHS` environment variable, both of which accept comma-delimited lists. Both `--plugin-paths` and `RALLY_PLUGIN_PATHS` can list either plugin module files, or directories containing plugins. For instance, both of these are valid:

```
rally --plugin-paths /rally/plugins ...
rally --plugin-paths /rally/plugins/foo.py,/rally/plugins/bar.py ...
```

You can also use a script `unpack_plugins_samples.sh` from `samples/plugins` which will automatically create the `~/.rally/plugins` directory.

How to create a plugin

To create your own plugin you need to inherit your plugin class from plugin.Plugin class or its subclasses. Also you need to decorate your class with `rally.task.scenario.configure`

```
from rally.task import scenario

@scenario.configure(name="my_new_plugin_name")
class MyNewPlugin(plugin.Plugin):
    pass
```

Context as a plugin

So what are contexts doing? These plugins will be executed before scenario iteration starts. For example, a context plugin could create resources (e.g., download 10 images) that will be used by the scenarios. All created objects must be put into the *self.context* dict, through which they will be available in the scenarios. Let's create a simple context plugin that adds a flavor to the environment before the benchmark task starts and deletes it after it finishes.

Creation

Inherit a class for your plugin from the base *Context* class. Then, implement the Context API: the *setup()* method that creates a flavor and the *cleanup()* method that deletes it.

```
from rally.task import context
from rally.common import logging
from rally import consts
from rally import osclients

LOG = logging.getLogger(__name__)


@context.configure(name="create_flavor", order=1000)
class CreateFlavorContext(context.Context):
    """This sample creates a flavor with specified options before task starts
    and deletes it after task completion.

    To create your own context plugin, inherit it from
    rally.task.context.Context
    """

    CONFIG_SCHEMA = {
        "type": "object",
        "$schema": consts.JSON_SCHEMA,
        "additionalProperties": False,
        "properties": {
            "flavor_name": {
                "type": "string",
            },
            "ram": {
                "type": "integer",
                "minimum": 1
            },
            "vcpus": {
                "type": "integer",
                "minimum": 1
            },
            "disk": {
                "type": "integer",
                "minimum": 1
            }
        }
    }

    def setup(self):
        """This method is called before the task starts."""
        try:
            # use rally.osclients to get necessary client instance
            nova = osclients.Clients(self.context["admin"]["credential"]).nova()
```

```
            # and than do what you need with this client
            self.context["flavor"] = nova.flavors.create(
                # context settings are stored in self.config
                name=self.config.get("flavor_name", "rally_test_flavor"),
                ram=self.config.get("ram", 1),
                vcpus=self.config.get("vcpus", 1),
                disk=self.config.get("disk", 1)).to_dict()
            LOG.debug("Flavor with id '%s'" % self.context["flavor"]["id"])
        except Exception as e:
            msg = "Can't create flavor: %s" % e.message
            if logging.is_debug():
                LOG.exception(msg)
            else:
                LOG.warning(msg)

    def cleanup(self):
        """This method is called after the task finishes."""
        try:
            nova = osclients.Clients(self.context["admin"]["credential"]).nova()
            nova.flavors.delete(self.context["flavor"]["id"])
            LOG.debug("Flavor '%s' deleted" % self.context["flavor"]["id"])
        except Exception as e:
            msg = "Can't delete flavor: %s" % e.message
            if logging.is_debug():
                LOG.exception(msg)
            else:
                LOG.warning(msg)
```

Usage

You can refer to your plugin context in the benchmark task configuration files in the same way as any other contexts:

```
{
    "Dummy.dummy": [
        {
            "args": {
                "sleep": 0.01
            },
            "runner": {
                "type": "constant",
                "times": 5,
                "concurrency": 1
            },
            "context": {
                "users": {
                    "tenants": 1,
                    "users_per_tenant": 1
                },
                 "create_flavor": {
                    "ram": 1024
                }
            }
        }
    ]
}
```

Hooks. Hook trigger plugins

Why Hooks?

All Rally workloads repeat their actions as many times as it is configured by runner. Once run, there is no way to interrupt the runner to evaluate any change or restart event on the stability of the cloud under test. For example we would like to test how configuration change or cloud component restart would affect performance and stability.

Task hooks were added to fill this gap and allow to use Rally for reliability and high availability testing. Generally, hooks allow to perform any actions on specified iteration or specified time since the workload has been started.

Also, task html-report provides results of hook execution. They can contain graphical or textual information with timing and statistics.

Hooks & Triggers Overview

Architecture

Rally uses runners to specify how many times the workload should be executed. Hooks do not use runners, instead they rely on trigger plugins to specify when and how many times hook should be called. Therefore hooks are isolated from workload runners and do not affect them because each hook is executed in separate thread.

Sample of usage

Hooks can be added to the task configuration. Lets take a look at hook configuration:

```
{
    "name": "sys_call",
    "args": "/bin/echo 123",
    "trigger": {
        "name": "event",
        "args": {
            "unit": "iteration",
            "at": [5, 50, 200, 1000]
        }
    }
}
```

It specifies hook plugin with name “sys_call”. “args” field contains string that will be used by sys_call plugin, but in case of any other hook plugin it can contain any other Python object, that is assumed to be passed to the hook. “trigger” field specifies which trigger plugin should be used to run this hook. “trigger” contains similar fields “name” and “args” which represent trigger plugin name and arguments for trigger plugin. In this example “event” trigger is specified and configured to run the hook at 5th, 50th, 200th and 1000th iterations.

Here is a full task config that contains previous hook configuraiton:

```
{
    "Dummy.dummy": [
        {
            "args": {
                "sleep": 0.01
            },
            "runner": {
                "type": "constant",
                "times": 1500,
```

```
                "concurrency": 1
            },
            "hooks": [
                {
                    "name": "sys_call",
                    "args": "/bin/echo 123",
                    "trigger": {
                        "name": "event",
                        "args": {
                            "unit": "iteration",
                            "at": [5, 50, 200, 1000]
                        }
                    }
                }
            ]
        }
    ]
}
```

Note: In this example, runner is configured to run workload 1500 times. So there is a limit for iterations and hook will be triggered only if certain iteration is started by runner. In other words, if trigger specifies iteration out of runner iterations scope then such trigger will not be called.

Task report for this example will contain minimal information about hook execution: duration of each hook call and its status(success of failure).

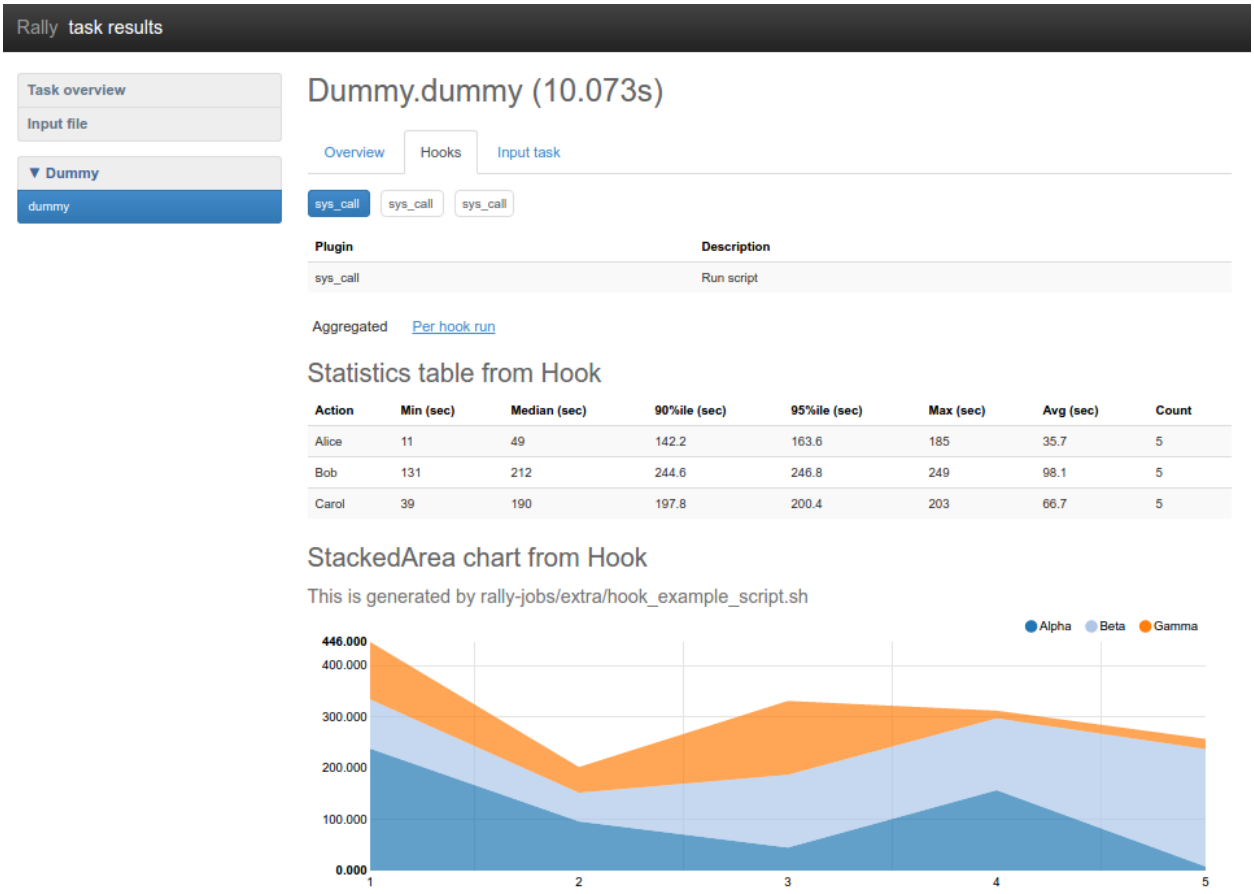
Let's take a look at more complicated config that can produce graphical and textual information.

```
---
  Dummy.dummy:
    -
      args:
        sleep: 0.75
      runner:
        type: "constant"
        times: 20
        concurrency: 2
      hooks:
        - name: sys_call
          description: Run script
          args: sh rally/rally-jobs/extra/hook_example_script.sh
          trigger:
            name: event
            args:
              unit: iteration
              at: [2, 5, 8, 13, 17]
        - name: sys_call
          description: Show time
          args: date +%Y-%m-%dT%H:%M:%S
          trigger:
            name: event
            args:
              unit: time
              at: [0, 2, 5, 6, 9]
        - name: sys_call
          description: Show system name
```

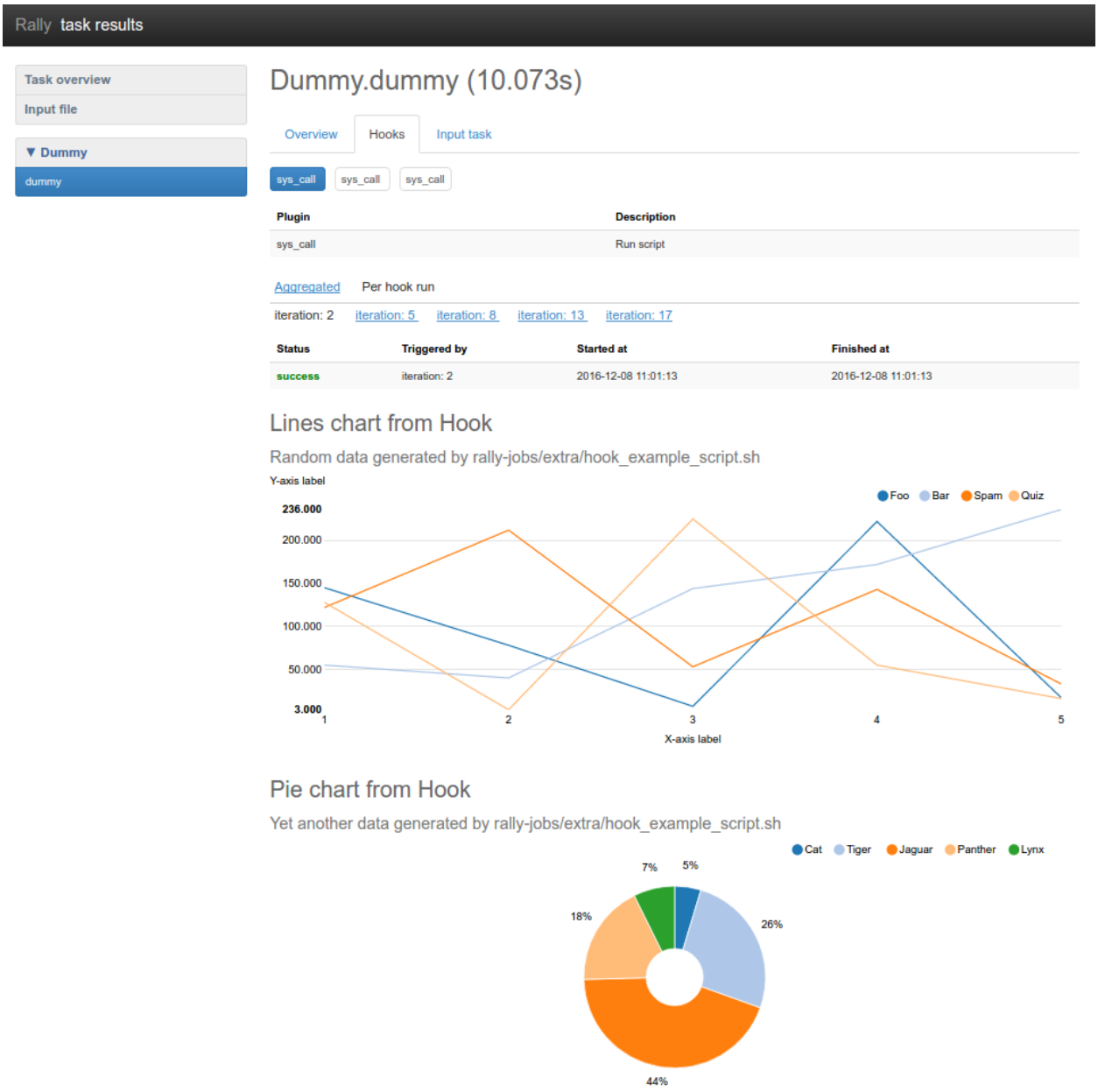
```
        args: uname -a
        trigger:
          name: event
          args:
            unit: iteration
            at: [2, 3, 4, 5, 6, 8, 10, 12, 13, 15, 17, 18]
    sla:
      failure_rate:
        max: 0
```

hook_example_script.sh generates dummy output in JSON format. Grafical information format is the same as for workloads and the same types of charts are supported for the hooks.

Here is a report that shows aggregated table and chart with hook results:



Here is report that shows lines chart and pie chart for first hook on the second iteration:



Browse existing Hooks and Triggers.

Writing your own Hook plugin

Problem description

Hook plugin should implement custom action that can be done one or multiple times during the workload. Examples of such actions might be the following:

- Destructive action inside cloud (Fault Injection)
- Getting information about current state of cloud (load/health)
- Upgrading/downgrading a component of cloud

- Changing configuration of cloud
- etc.

Plugin code

The following example shows simple hook code that performs system call. It is inherited from the base *Hook* class and contains implemented `run()` method:

```
import shlex
import subprocess

from rally import consts
from rally.task import hook


@hook.configure(name="simple_sys_call")
class SimpleSysCallHook(hook.Hook):
    """Performs system call."""

    CONFIG_SCHEMA = {
        "$schema": consts.JSON_SCHEMA,
        "type": "string",
    }

    def run(self):
        proc = subprocess.Popen(shlex.split(self.config),
                                stdout=subprocess.PIPE,
                                stderr=subprocess.STDOUT)
        proc.wait()
        if proc.returncode:
            self.set_error(
                exception_name="n/a",  # no exception class
                description="Subprocess returned {}".format(proc.returncode),
                details=proc.stdout.read(),
            )
```

Any exceptions risen during execution of `run` method will be caught by Hook base class and saved as a result. Although hook should manually call `Hook.set_error()` to indicate logical error in case if there is no exception raised.

Also there is a method for saving charts data: `Hook.add_output()`.

Plugin Placement

There are two folders for hook plugins:

- OpenStack Hooks
- Common Hooks

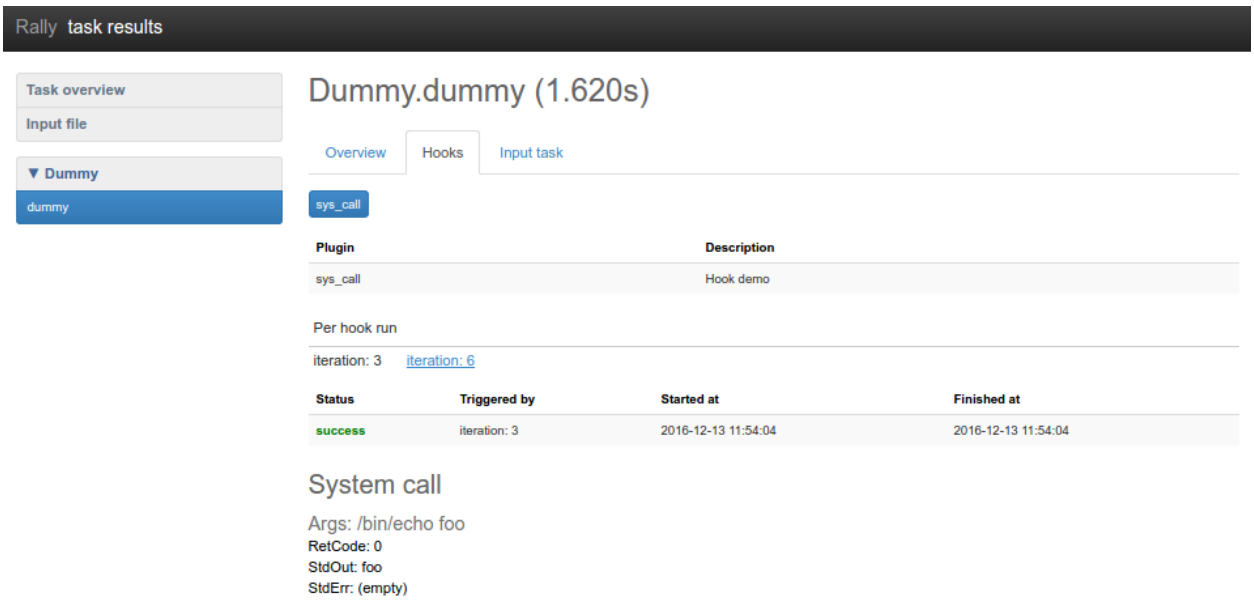
Sample of task that uses Hook

```
{
    "Dummy.dummy": [
```

```
        {
            "args": {
                "sleep": 0.01
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 1
            },
            "hooks": [
                {
                    "name": "simple_sys_call",
                    "args": "/bin/echo 123",
                    "trigger": {
                        "name": "event",
                        "args": {
                            "unit": "iteration",
                            "at": [3, 6]
                        }
                    }
                }
            ]
        }
    ]
}
```

Results of task execution

Result of previous task example:



Writing your own Trigger plugin

Problem description

Trigger plugin should implement an event processor that decides whether to start hook or not. Rally has two basic triggers that should cover most cases:

- Event Trigger
- Periodic Trigger

Plugin code

This example shows the code of the existing Event trigger:

```
from rally import consts
from rally.task import trigger


@trigger.configure(name="event")
class EventTrigger(trigger.Trigger):
    """Triggers hook on specified event and list of values."""

    CONFIG_SCHEMA = {
        "type": "object",
        "$schema": consts.JSON_SCHEMA,
        "oneOf": [
            {
                "properties": {
                    "unit": {"enum": ["time"]},
                    "at": {
                        "type": "array",
                        "minItems": 1,
                        "uniqueItems": True,
                        "items": {
                            "type": "integer",
                            "minimum": 0,
                        }
                    },
                },
                "required": ["unit", "at"],
                "additionalProperties": False,
            },
            {
                "properties": {
                    "unit": {"enum": ["iteration"]},
                    "at": {
                        "type": "array",
                        "minItems": 1,
                        "uniqueItems": True,
                        "items": {
                            "type": "integer",
                            "minimum": 1,
                        }
                    },
                },
                "required": ["unit", "at"],
                "additionalProperties": False,
            },
```

```
        ]
    }

    def get_listening_event(self):
        return self.config["unit"]

    def on_event(self, event_type, value=None):
        if not (event_type == self.get_listening_event()
                and value in self.config["at"]):
            # do nothing
            return
        super(EventTrigger, self).on_event(event_type, value)
```

Trigger plugins must override two methods:

- `get_listening_event` - this method should return currently configured event name. (So far Rally supports only "time" and "iteration")
- `on_event` - this method is called each time certain events occur. It calls base method when the hook is triggered on specified event.

Plugin Placement

All trigger plugins should be placed in Trigger folder.

Scenario runner as a plugin

Let's create a scenario runner plugin that runs a given benchmark scenario a random number of times (chosen at random from a given range).

Creation

Inherit a class for your plugin from the base *ScenarioRunner* class and implement its API (the *_run_scenario()* method):

```
import random

from rally.task import runner
from rally import consts


@runner.configure(name="random_times")
class RandomTimesScenarioRunner(runner.ScenarioRunner):
    """Sample scenario runner plugin.

    Run scenario random number of times, which is chosen between min_times and
    max_times.
    """

    CONFIG_SCHEMA = {
        "type": "object",
        "$schema": consts.JSON_SCHEMA,
        "properties": {
            "type": {
```

```
                "type": "string"
            },
            "min_times": {
                "type": "integer",
                "minimum": 1
            },
            "max_times": {
                "type": "integer",
                "minimum": 1
            }
        },
        "additionalProperties": True
    }

    def _run_scenario(self, cls, method_name, context, args):
        # runners settings are stored in self.config
        min_times = self.config.get('min_times', 1)
        max_times = self.config.get('max_times', 1)

        for i in range(random.randrange(min_times, max_times)):
            run_args = (i, cls, method_name,
                        runner._get_scenario_context(context), args)
            result = runner._run_scenario_once(run_args)
            # use self.send_result for result of each iteration
            self._send_result(result)
```

Usage

You can refer to your scenario runner in the benchmark task configuration files in the same way as any other runners. Don't forget to put your runner-specific parameters in the configuration as well (*"min_times"* and *"max_times"* in our example):

```
{
    "Dummy.dummy": [
        {
            "runner": {
                "type": "random_times",
                "min_times": 10,
                "max_times": 20,
            },
            "context": {
                "users": {
                    "tenants": 1,
                    "users_per_tenant": 1
                }
            }
        }
    ]
}
```

Different plugin samples are available here.

Scenario as a plugin

Let's create a simple scenario plugin that list flavors.

Creation

Inherit a class for your plugin from the base *Scenario* class and implement a scenario method inside it. In our scenario, we'll first list flavors as an ordinary user, and then repeat the same using admin clients:

```
from rally.task import atomic
from rally.task import scenario


class ScenarioPlugin(scenario.Scenario):
    """Sample plugin which lists flavors."""

    @atomic.action_timer("list_flavors")
    def _list_flavors(self):
        """Sample of usage clients - list flavors

        You can use self.context, self.admin_clients and self.clients which are
        initialized on scenario instance creation"""
        self.clients("nova").flavors.list()

    @atomic.action_timer("list_flavors_as_admin")
    def _list_flavors_as_admin(self):
        """The same with admin clients"""
        self.admin_clients("nova").flavors.list()

    @scenario.configure()
    def list_flavors(self):
        """List flavors."""
        self._list_flavors()
        self._list_flavors_as_admin()
```

Usage

You can refer to your plugin scenario in the benchmark task configuration files in the same way as any other scenarios:

```
{
    "ScenarioPlugin.list_flavors": [
        {
            "runner": {
                "type": "serial",
                "times": 5,
            },
            "context": {
                "create_flavor": {
                    "ram": 512,
                }
            }
        }
    ]
}
```

This configuration file uses the *"create_flavor"* context which we created in *Context as a plugin*.

SLA as a plugin

Let's create an SLA (success criterion) plugin that checks whether the range of the observed performance measurements does not exceed the allowed maximum value.

Creation

Inherit a class for your plugin from the base *SLA* class and implement its API (the *add_iteration(iteration)*, the *details()* method):

```
from rally.task import sla
from rally.common.i18n import _

@sla.configure(name="max_duration_range")
class MaxDurationRange(sla.SLA):
    """Maximum allowed duration range in seconds."""

    CONFIG_SCHEMA = {
        "type": "number",
        "minimum": 0.0,
    }

    def __init__(self, criterion_value):
        super(MaxDurationRange, self).__init__(criterion_value)
        self._min = 0
        self._max = 0

    def add_iteration(self, iteration):
        # Skipping failed iterations (that raised exceptions)
        if iteration.get("error"):
            return self.success   # This field is defined in base class

        # Updating _min and _max values
        self._max = max(self._max, iteration["duration"])
        self._min = min(self._min, iteration["duration"])

        # Updating successfulness based on new max and min values
        self.success = self._max - self._min <= self.criterion_value
        return self.success

    def details(self):
        return (_("%s - Maximum allowed duration range: %.2f%% <= %.2f%%") %
                (self.status(), self._max - self._min, self.criterion_value))
```

Usage

You can refer to your SLA in the benchmark task configuration files in the same way as any other SLA:

```
{
    "Dummy.dummy": [
        {
            "args": {
                "sleep": 0.01
            },
            "runner": {
```

```
                "type": "constant",
                "times": 5,
                "concurrency": 1
            },
            "context": {
                "users": {
                    "tenants": 1,
                    "users_per_tenant": 1
                }
            },
            "sla": {
                "max_duration_range": 2.5
            }
        }
    ]
}
```

Contribute to Rally

Where to begin

Please take a look our Roadmap to get information about our current work directions.

In case you have questions or want to share your ideas, be sure to contact us either at Rally-dev/Lobby channel on **Gitter** messenger (or, less preferably, at the `#openstack-rally` IRC channel on **irc.freenode.net**).

If you are going to contribute to Rally, you will probably need to grasp a better understanding of several main design concepts used throughout our project (such as **benchmark scenarios**, **contexts** etc.). To do so, please read this article.

How to contribute

1. You need a Launchpad account and need to be joined to the OpenStack team. You can also join the Rally team if you want to. Make sure Launchpad has your SSH key, Gerrit (the code review system) uses this.

2. Sign the CLA as outlined in the account setup section of the developer guide.

3. Tell git your details:

```
git config --global user.name "Firstname Lastname"
git config --global user.email "your_email@youremail.com"
```

4. Install git-review. This tool takes a lot of the pain out of remembering commands to push code up to Gerrit for review and to pull it back down to edit it. It is installed using:

```
pip install git-review
```

Several Linux distributions (notably Fedora 16 and Ubuntu 12.04) are also starting to include git-review in their repositories so it can also be installed using the standard package manager.

5. Grab the Rally repository:

```
git clone git@github.com:openstack/rally.git
```

6. Checkout a new branch to hack on:

```
git checkout -b TOPIC-BRANCH
```

7. Start coding

8. Run the test suite locally to make sure nothing broke, e.g. (this will run py34/py27/pep8 tests):

```
tox
```

(NOTE: you should have installed tox<=1.6.1)

If you extend Rally with new functionality, make sure you have also provided unit and/or functional tests for it.

9. Commit your work using:

```
git commit -a
```

Make sure you have supplied your commit with a neat commit message, containing a link to the corresponding blueprint / bug, if appropriate.

10. Push the commit up for code review using:

```
git review -R
```

That is the awesome tool we installed earlier that does a lot of hard work for you.

11. Watch your email or review site, it will automatically send your code for a battery of tests on our Jenkins setup and the core team for the project will review your code. If there are any changes that should be made they will let you know.

12. When all is good the review site will automatically merge your code.

(This tutorial is based on: http://www.linuxjedi.co.uk/2012/03/real-way-to-start-hacking-on-openstack.html)

Testing

Please, don't hesitate to write tests ;)

Unit tests

*Files: /tests/unit/**

The goal of unit tests is to ensure that internal parts of the code work properly. All internal methods should be fully covered by unit tests with a reasonable mocks usage.

About Rally unit tests:

- All unit tests are located inside /tests/unit/*
- Tests are written on top of: *testtools* and *mock* libs
- Tox is used to run unit tests

To run unit tests locally:

```
$ pip install tox
$ tox
```

To run py34, py27 or pep8 only:

```
$ tox -e <name>

#NOTE: <name> is one of py34, py27 or pep8
```

To run a single unit test e.g. test_deployment

```
$ tox -e <name> -- <test_name>

#NOTE: <name> is one of py34, py27 or pep8
#      <test_name> is the unit test case name, e.g tests.unit.test_osclients
```

To debug issues on the unit test:

- Add breakpoints on the test file using `import pdb; pdb.set_trace()`
- Then run tox in debug mode:

```
$ tox -e debug <test_name>
#NOTE: use python 2.7
#NOTE: <test_name> is the unit test case name

or
```

```
$ tox -e debug34 <test_name>
#NOTE: use python 3.4
#NOTE: <test_name> is the unit test case name
```

To get test coverage:

```
$ tox -e cover

#NOTE: Results will be in /cover/index.html
```

To generate docs:

```
$ tox -e docs

#NOTE: Documentation will be in doc/source/_build/html/index.html
```

Functional tests

*Files: /tests/functional/**

The goal of functional tests is to check that everything works well together. Functional tests use Rally API only and check responses without touching internal parts.

To run functional tests locally:

```
$ source openrc
$ rally deployment create --fromenv --name testing
$ tox -e cli

#NOTE: openrc file with OpenStack admin credentials
```

Output of every Rally execution will be collected under some reports root in directory structure like: reports_root/ClassName/MethodName_suffix.extension This functionality implemented in

tests.functional.utils.Rally.__call__ method. Use 'gen_report_path' method of 'Rally' class to get automatically generated file path and name if you need. You can use it to publish html reports, generated during tests. Reports root can be passed throw environment variable 'REPORTS_ROOT'. Default is 'rally-cli-output-files'.

Rally CI scripts

*Files: /tests/ci/**

This directory contains scripts and files related to the Rally CI system.

Rally Style Commandments

Files: /tests/hacking/

This module contains Rally specific hacking rules for checking commandments.

For more information about Style Commandments, read the OpenStack Style Commandments manual.

Request New Features

To request a new feature, you should create a document similar to other feature requests and then contribute it to the **doc/feature_request** directory of the Rally repository (see the *How-to-contribute tutorial*).

If you don't have time to contribute your feature request via Gerrit, please contact Boris Pavlovic (boris@pavlovic.me)

Active feature requests:

Capture Logs from services

Use case

A developer is executing various task and would like to capture logs as well as test results.

Problem description

In case of errors it is quite hard to debug what happened.

Possible solution

- Add special context that can capture the logs from tested services.

Check queue perfdata

Use case

Sometimes OpenStack services use common messaging system very prodigally. For example Neutron metering agent sending all database table data on new object creation i.e https://review.openstack.org/#/c/143672/. It cause to Neutron degradation and other obvious problems. It will be nice to have a way to track messages count and messages size in queue during tests/benchmarks.

Problem description

Heavy usage of queue isn't checked.

Possible solution

- Before running tests/benchmarks start process which will connect to queue topics and measure messages count, size and other data which we need.

Ability to compare results between task

Use case

During the work on performance it's essential to be able to compare results of similar task before and after change in system.

Problem description

There is no command to compare two or more tasks and get tables and graphs.

Possible solution

- Add command that accepts 2 tasks UUID and prints graphs that compares result

Distributed load generation

Use Case

Some OpenStack projects (Marconi, MagnetoDB) require a real huge load, like 10-100k request per second for benchmarking.

To generate such huge load Rally have to create load from different servers.

Problem Description

- Rally can't generate load from different servers
- Result processing can't handle big amount of data
- There is no support for chunking results

Explicitly specify existing users for scenarios

Use Case

Rally allows to reuse existing users for scenario runs. And we should be able to use only specified set of existing users for specific scenarios.

Problem Description

For the moment if used *deployment* with existing users then Rally chooses user for each scenario run randomly. But there are cases when we may want to use one scenario with one user and another with different one specific user. Main reason for it is in different set of resources that each user has and those resources may be required for scenarios. Without this feature Rally user is forced to make all existing users similar and have all required resources set up for all scenarios he uses. But it is redundant.

Possible solution

- Make it possible to use explicitly existing_users context

Historical performance data

Use case

OpenStack is really rapidly developed. Hundreds of patches are merged daily and it's really hard to track how performance is changed during time. It will be nice to have a way to track performance of major functionality of OpenStack running periodically rally task and building graphs that represent how performance of specific method is changed during the time.

Problem description

There is no way to bind tasks

Possible solution

- Add grouping for tasks
- Add command that creates historical graphs

Enhancements to installation script: `--version` and `--uninstall`

Use case

User might wish to control which rally version is installed or even purge rally from the machine completely.

Problem description

1. Installation script doesn't allow to choose version.
2. No un-install support.

Possible solution

1. Add `--version` option to installation script.
2. Add `--uninstall` option to installation script or create an un-installation script

Installation script: `--pypi-mirror`, `--package-mirror` and `--venv-mirror`

Use case

Installation is pretty easy when there is an Internet connection available. And there is surely a number of OpenStack uses when whole environment is isolated. In this case, we need somehow specify where installation script should take required libs and packages.

Problem description

1. Installation script can't work without direct Internet connection

Possible solution #1

1. Add `--pypi-mirror` option to installation script.
2. Add `--package-mirror` option to installation script.
3. Add `--venv-mirror` option to installation script.

Launch Specific Benchmark(s)

Use case

A developer is working on a feature that is covered by one or more specific benchmarks/scenarios. He/she would like to execute a rally task with an existing task template file (YAML or JSON) indicating exactly which benchmark(s) will be executed.

Problem description

When executing a task with a template file in Rally, all benchmarks are executed without the ability to specify one or a set of benchmarks the user would like to execute.

Possible solution

- Add optional flag to rally task start command to specify one or more benchmarks to execute as part of that test run.

Using multi scenarios to generate load

Use Case

Rally should be able to generate real life load. Simultaneously create load on different components of OpenStack, e.g. simultaneously booting VM, uploading image and listing users.

Problem Description

At the moment Rally is able to run only 1 scenario per benchmark. Scenario are quite specific (e.g. boot and delete VM for example) and can't actually generate real life load.

Writing a lot of specific benchmark scenarios that will produce more real life load will produce mess and a lot of duplication of code.

Possible solution

- Extend Rally task benchmark configuration in such way to support passing multiple benchmark scenarios in single benchmark context
- Extend Rally task output format to support results of multiple scenarios in single benchmark separately.
- Extend rally task plot2html and rally task detailed to show results separately for every scenario.

Multiple attach volume

Use Case

Since multiple volume attaching support to OpenStack Mitaka, one volume can be attached to several instances or hosts, Rally should add scenarios about multiple attach volume.

Problem Description

Rally lack of scenarios about multiple attach volume.

Possible solution

- Add nova scenarios "multi_attach_volume" and "multi_detach_volume"

Add support of persistence benchmark environment

Use Case

To benchmark many of operations like show, list, detailed you need to have already these resource in cloud. So it will be nice to be able to create benchmark environment once before benchmarking. So run some amount of benchmarks that are using it and at the end just delete all created resources by benchmark environment.

Problem Description

Fortunately Rally has already a mechanism for creating benchmark environment, that is used to create load. Unfortunately it's atomic operation: (create environment, make load, delete environment). This should be split to 3 separated steps.

Possible solution

- Add new CLI operations to work with benchmark environment: (show, create, delete, list)
- Allow task to start against benchmark environment (instead of deployment)

Production read cleanups

Use Case

Rally should delete in any case all resources that it created during benchmark.

Problem Description

- (implemented) Deletion rate limit

  You can kill cloud by deleting too many objects simultaneously, so deletion rate limit is required

- (implemented) Retry on failures

  There should be few attempts to delete resource in case of failures

- (implemented) Log resources that failed to be deleted

  We should log warnings about all non deleted resources. This information should include UUID of resource, it's type and project.

- (implemented) Pluggable

  It should be simple to add new cleanups adding just plugins somewhere.

- Disaster recovery

  Rally should use special name patterns, to be able to delete resources in such case if something went wrong with server that is running Rally. And you have just new instance (without old Rally DB) of Rally on new server.

Project Info and Release Notes

Maintainers

Project Team Lead (PTL)

Contact	Area of interest
Andrey Kurilin andreykurilin (irc) andreykurilin (gitter) andr.kurilin@gmail.com akurilin@mirantis.com	• Chief Architect • Release management • Community management • Core team management • Road Map

If you would like to refactor whole Rally or have UX/community/other issues please contact me.

Project Core maintainers

Contact	Area of interest
Alexander Maretskiy amaretskiy (irc) amaretskiy@mirantis.com	• Rally reports • Front-end
Anton Studenov tohin (irc) astudenov@mirantis.com	• Rally Deployment • Task Hooks
Boris Pavlovic boris-42 (irc) boris@pavlovic.me	• Founder and ideological leader • Architect • Rally task & benchmark
Chris St. Pierre stpierre (irc) cstpierr@cisco.com	• Rally task & benchmark • Bash guru ;)
Illia Khudoshyn ikhudoshyn (irc) ikhudoshyn@mirantis.com	• Rally task & benchmark
Kun Huang kun_huang (irc) gareth.huang@huawei.com	• Rally task & benchmark
Li Yingjun liyingjun (irc) yingjun.li@kylin-cloud.com	• Rally task & benchmark
Roman Vasilets rvasilets (irc) pomeo92@gmail.com	• Rally task & benchmark
Sergey Skripnick redixin (irc) sskripnick@mirantis.com	• Rally CI/CD • Rally deploy • Automation of everything
Yair Fried yfried (irc)	• Rally-Tempest integration • Rally task & benchmark

All cores from this list are reviewing all changes that are proposed to Rally. To avoid duplication of efforts, please contact them before starting work on your code.

Plugin Core reviewers

Contact	Area of interest
Ivan Kolodyazhny e0ne (irc) e0ne@e0ne.info	• Cinder plugins
Nikita Konovalov NikitaKonovalov (irc) nkonovalov@mirantis.com	• Sahara plugins
Oleg Bondarev obondarev (irc) obondarev@mirantis.com	• Neutron plugins
Sergey Kraynev skraynev (irc) skraynev@mirantis.com	• Heat plugins
Spyros Trigazis strigazi (irc) strigazi@gmail.com	• Magnum plugins

All cores from this list are responsible for their component plugins. To avoid duplication of efforts, please contact them before starting working on your own plugins.

Useful links

- Source code
- Rally roadmap
- Project space
- Bugs
- Patches on review
- **Meeting logs** **(server: irc.freenode.net, channel: #openstack-meeting)**

- IRC logs (server: **irc.freenode.net**, channel: **#openstack-rally**)
- Gitter chat
- Trello board

Where can I discuss and propose changes?

- Our IRC channel: **#openstack-rally** on **irc.freenode.net**;
- Weekly Rally team meeting (in IRC): **#openstack-meeting** on **irc.freenode.net**, held on Mondays at 14:00 UTC;
- OpenStack mailing list: **openstack-dev@lists.openstack.org** (see subscription and usage instructions);
- Rally team on Launchpad: Answers/Bugs/Blueprints.

Release Notes

All release notes

Rally v0.0.1

Information

Commits	**1039**
Bug fixes	**0**
Dev cycle	**547 days**
Release date	**26/Jan/2015**

Details

Rally is awesome tool for testing verifying and benchmarking OpenStack clouds.

A lot of people started using Rally in their CI/CD so Rally team should provide more stable product with clear strategy of deprecation and upgrades.

Rally v0.0.2

Information

Commits	**100**
Bug fixes	**18**
Dev cycle	**45 days**
Release date	**12/Mar/2015**

Details

This release contains new features, new benchmark plugins, bug fixes, various code and API improvements.

New Features

- rally task start **–abort-on-sla-failure**

  Stopping load before things go wrong. Load generation will be interrupted if SLA criteria stop passing.

- Rally verify command supports multiple Tempest sources now.
- python34 support
- postgres DB backend support

API changes

- [new] **rally [deployment | verify | task] use** subcommand

  It should be used instead of root command **rally use**

- [new] Rally as a Lib API

  To avoid code duplication between Rally as CLI tool and Rally as a Service we decide to make Rally as a Lib as a common part between these 2 modes.

  Rally as a Service will be a daemon that just maps HTTP request to Rally as a Lib API.

- [deprecated] **rally use** CLI command
- [deprecated] Old Rally as a Lib API

  Old Rally API was quite mixed up so we decide to deprecate it

Plugins

- **Benchmark Scenario Runners**:

  [improved] Improved algorithm of generation load in **constant runner**

  Before we used processes to generate load, now it creates pool of processes (amount of processes is equal to CPU count) after that in each process use threads to generate load. So now you can easily generate load of 1k concurrent scenarios.

  [improved] Unify code of **constant** and **rps** runners

  [interface] Added **abort()** to runner's plugin interface

  New method **abort()** is used to immediately interrupt execution.

- **Benchmark Scenarios**:

  [new] DesignateBasic.create_and_delete_server

  [new] DesignateBasic.create_and_list_servers

  [new] DesignateBasic.list_servers

  [new] MistralWorkbooks.list_workbooks

  [new] MistralWorkbooks.create_workbook

  [new] Quotas.neutron_update

  [new] HeatStacks.create_update_delete_stack

[new] HeatStacks.list_stacks_and_resources

[new] HeatStacks.create_suspend_resume_delete_stac

[new] HeatStacks.create_check_delete_stack

[new] NeutronNetworks.create_and_delete_routers

[new] NovaKeypair.create_and_delete_keypair

[new] NovaKeypair.create_and_list_keypairs

[new] NovaKeypair.boot_and_delete_server_with_keypair

[new] NovaServers.boot_server_from_volume_and_live_migrate

[new] NovaServers.boot_server_attach_created_volume_and_live_migrate

[new] CinderVolumes.create_and_upload_volume_to_image

[fix] CinderVolumes.create_and_attach_volume

Pass optional **kwargs only to create server command

[fix] GlanceImages.create_image_and_boot_instances

Pass optional **kwargs only to create server command

[fix] TempestScenario.* removed stress cleanup.

Major issue is that tempest stress cleanup cleans whole OpenStack. This is very dangerous, so it's better to remove it and leave some extra resources.

[improved] NovaSecGroup.boot_and_delete_server_with_secgroups

Add optional **kwargs that are passed to boot server comment

- **Benchmark Context**:

[new] **stacks**

Generates passed amount of heat stacks for all tenants.

[new] **custom_image**

Prepares images for benchmarks in VMs.

To Support generating workloads in VMs by existing tools like: IPerf, Blogbench, HPCC and others we have to have prepared images, with already installed and configured tools.

Rally team decide to generate such images on fly from passed to avoid requirements of having big repository with a lot of images.

This context is abstract context that allows to automate next steps:

1. runs VM with passed image (with floating ip and other stuff)
2. execute abstract method that has access to VM
3. snapshot this image

In future we are going to use this as a base for making context that prepares images.

[improved] **allow_ssh**

Automatically disable it if security group are disabled in neutron.

[improved] **keypair**

Key pairs are stored in “users” space it means that accessing keypair from scenario is simpler now:

self.context[”user”][”keypair”][”private”]

[fix] **users**

Pass proper EndpointType for newly created users

[fix] **sahara_edp**

The Job Binaries data should be treated as a binary content

- **Benchmark SLA**:

  [interface] SLA calculations is done in additive way now

  Resolves scale issues, because now we don’t need to have whole array of iterations in memory to process SLA.

  This is required to implement **–abort-on-sla-failure** feature

  [all] SLA plugins were rewritten to implement new interface

Bug fixes

18 bugs were fixed, the most critical are:

- Fix **rally task detailed –iterations-data**

  It didn’t work in case of missing atomic actions. Such situation can occur if scenario method raises exceptions

- Add user-friendly message if the task cannot be deleted

  In case of trying to delete task that is not in “finished” status users get traces instead of user-friendly message try to run it with –force key.

- Network context cleanups networks properly now

Documentation

- Image sizes are fixed
- New tutorial in “Step by Step” relate to **–abort-on-sla-failure**
- Various fixes

Rally v0.0.3

Information

Commits	**53**
Bug fixes	**14**
Dev cycle	**33 days**
Release date	**14/Apr/2015**

Details

This release contains new features, new benchmark plugins, bug fixes, various code and API improvements.

New Features & API changes

- Add the ability to specify versions for clients in benchmark scenarios

  You can call self.clients("glance", "2") and get any client for specific version.

- Add API for tempest uninstall

  $ rally-manage tempest uninstall # removes fully tempest for active deployment

- Add a –uuids-only option to rally task list

  $ rally task list –uuids-only # returns list with only task uuids

- Adds endpoint to –fromenv deployment creation

  $ rally deployment create –fromenv # recognizes standard OS_ENDPOINT environment variable

- Configure SSL per deployment

  Now SSL information is deployment specific not Rally specific and rally.conf option is deprecated

  Like in this sample https://github.com/openstack/rally/blob/14d0b5ba0c75ececfdb6a6c121d9cf2810571f77/samples/deployments/existing.json#L11-L12

Specs

- [spec] Proposal for new task input file format

  This spec describes new task input format that will allow us to generate multi scenario load which is crucial for HA and more real life testing:

  https://github.com/openstack/rally/blob/master/doc/specs/in-progress/new_rally_input_task_format.rst

Plugins

- **Benchmark Scenario Runners**:
  - Add a maximum concurrency option to rps runner

    To avoid running to heavy load you can set 'concurrency' to configuration and in case if cloud is not able to process all requests it won't start more parallel requests then 'concurrency' value.

- **Benchmark Scenarios**:

  [new] CeilometerAlarms.create_alarm_and_get_history

  [new] KeystoneBasic.get_entities

  [new] EC2Servers.boot_server

  [new] KeystoneBasic.create_and_delete_service

  [new] MuranoEnvironments.list_environments

  [new] MuranoEnvironments.create_and_delete_environment

  [new] NovaServers.suspend_and_resume_server

[new] NovaServers.pause_and_unpause_server

[new] NovaServers.boot_and_rebuild_server

[new] KeystoneBasic.create_and_list_services

[new] HeatStacks.list_stacks_and_events

[improved] VMTask.boot_runcommand_delete

restore ability to use fixed IP and floating IP to connect to VM via ssh

[fix] NovaServers.boot_server_attach_created_volume_and_live_migrate

Kwargs in nova scenario were wrongly passed

- **Benchmark SLA**:
  - [new] aborted_on_sla

  This is internal SLA criteria, that is added if task was aborted
  - [new] something_went_wrong

  This is internal SLA criteria, that is added if something went wrong, context failed to create or runner raised some exceptions

Bug fixes

14 bugs were fixed, the most critical are:

- Set default task uuid to running task. Before it was set only after task was fully finished.
- The "rally task results" command showed a disorienting "task not found" message for a task that is currently running.
- Rally didn't know how to reconnect to OpenStack in case if token expired.

Documentation

- New tutorial **task templates**

https://rally.readthedocs.org/en/latest/tutorial/step_5_task_templates.html

- Various fixes

Rally v0.0.4

Information

Commits	**87**
Bug fixes	**21**
Dev cycle	**30 days**
Release date	**14/May/2015**

Details

This release contains new features, new benchmark plugins, bug fixes, various code and API improvements.

New Features & API changes

- Rally now can generate load with users that already exist

  Now one can use Rally for benchmarking OpenStack clouds that are using LDAP, AD or any other read-only keystone backend where it is not possible to create any users. To do this, one should set up the "users" section of the deployment configuration of the ExistingCloud type. This feature also makes it safer to run Rally against production clouds: when run from an isolated group of users, Rally won't affect rest of the cloud users if something goes wrong.

- New decorator *@osclients.Clients.register* can add new OpenStack clients at runtime

  It is now possible to add a new OpenStack client dynamically at runtime. The added client will be available from osclients.Clients at the module level and cached. Example:

```
>>> from rally import osclients
>>> @osclients.Clients.register("supernova")
... def another_nova_client(self):
...   from novaclient import client as nova
...   return nova.Client("2", auth_token=self.keystone().auth_token,
...                      **self._get_auth_info(password_key="key"))
...
>>> clients = osclients.Clients.create_from_env()
>>> clients.supernova().services.list()[:2]
[<Service: nova-conductor>, <Service: nova-cert>]
```

- Assert methods now available for scenarios and contexts

  There is now a new *FunctionalMixin* class that implements basic unittest assert methods. The *base.Context* and *base.Scenario* classes inherit from this mixin, so now it is possible to use *base.assertX()* methods in scenarios and contexts.

- Improved installation script

  The installation script has been almost completely rewritten. After this change, it can be run from an unprivileged user, supports different database types, allows to specify a custom python binary, always asks confirmation before doing potentially dangerous actions, automatically install needed software if run as root, and also automatically cleans up the virtualenv and/or the downloaded repository if interrupted.

Specs & Feature requests

- [Spec] Reorder plugins

  The spec describes how to split Rally framework and plugins codebase to make it simpler for newbies to understand how Rally code is organized and how it works.

- [Feature request] Specify what benchmarks to execute in task

  This feature request proposes to add the ability to specify benchmark(s) to be executed when the user runs the *rally task start* command. A possible solution would be to add a special flag to the *rally task start* command.

Plugins

- **Benchmark Scenario Runners**:
  - Add limits for maximum Core usage to constant and rps runners

The new 'max_cpu_usage' parameter can be used to avoid possible 100% usage of all available CPU cores by reducing the number of CPU cores available for processes started by the corresponding runner.

- **Benchmark Scenarios**:
  - [new] KeystoneBasic.create_update_and_delete_tenant
  - [new] KeystoneBasic.create_user_update_password
  - [new] NovaServers.shelve_and_unshelve_server
  - [new] NovaServers.boot_and_associate_floating_ip
  - [new] NovaServers.boot_lock_unlock_and_delete
  - [new] NovaHypervisors.list_hypervisors
  - [new] CeilometerSamples.list_samples
  - [new] CeilometerResource.get_resources_on_tenant
  - [new] SwiftObjects.create_container_and_object_then_delete_all
  - [new] SwiftObjects.create_container_and_object_then_download_object
  - [new] SwiftObjects.create_container_and_object_then_list_objects
  - [new] MuranoEnvironments.create_and_deploy_environment
  - [new] HttpRequests.check_random_request
  - [new] HttpRequests.check_request
  - [improved] NovaServers live migrate benchmarks

    add 'min_sleep' and 'max_sleep' parameters to simulate a pause between VM booting and running live migration
  - [improved] NovaServers.boot_and_live_migrate_server

    add a usage sample to samples/tasks
  - [improved] CinderVolumes benchmarks

    support size range to be passed to the 'size' argument as a dictionary *{"min": <minimum_size>, "max": <maximum_size>}*
- **Benchmark Contexts**:
  - [new] MuranoPackage

    This new context can upload a package to Murano from some specified path.
  - [new] CeilometerSampleGenerator

    Context that can be used for creating samples and collecting resources for benchmarks in a list.
- **Benchmark SLA**:
  - [new] outliers

    This new SLA checks that the number of outliers (calculated from the mean and standard deviation of the iteration durations) does not exceed some maximum value. The SLA is highly configurable: the parameters used for outliers threshold calculation can be set by the user.

Bug fixes

21 bugs were fixed, the most critical are:

- Make it possible to use relative imports for plugins that are outside of rally package.
- Fix heat stacks cleanup by deleting them only 1 time per tenant (get rid of "stack not found" errors in logs).
- Fix the wrong behavior of 'rally task detailed –iterations-data' (it lacked the iteration info before).
- Fix security groups cleanup: a security group called "default", created automatically by Neutron, did not get deleted for each tenant.

Other changes

- Streaming algorithms that scale

  This release introduces the common/streaming_algorithms.py module. This module is going to contain implementations of benchmark data processing algorithms that scale: these algorithms do not store exhaustive information about every single benchmark iteration duration processed. For now, the module contains implementations of algorithms for computation of mean & standard deviation.

- Coverage job to check that new patches come with unit tests

  Rally now has a coverage job that checks that every patch submitted for review does not decrease the number of lines covered by unit tests (at least too much). This job allows to mark most patches with no unit tests with '-1'.

- Splitting the plugins code (Runners & SLA) into common/openstack plugins

  According to the spec "Reorder plugins" (see above), the plugins code for runners and SLA has been moved to the *plugins/common/* directory. Only base classes now remain in the *benchmark/* directory.

Documentation

- Various fixes
  - Remove obsolete *.rst* files (*deploy_engines.rst* / *server_providers.rst* / ...)
  - Restructure the docs files to make them easier to navigate through
  - Move the chapter on task templates to the 4th step in the tutorial
  - Update the information about meetings (new release meeting & time changes)

Rally v0.1.0

Information

Commits	**355**
Bug fixes	**90**
Dev cycle	**132 days**
Release date	**25/September/2015**

Details

This release contains new features, new 42 plugins, 90 bug fixes, various code and API improvements.

New Features & API changes

- **Improved installation script**
  - Add parameters:
    - `--develop` parameter to install rally in editable (develop) mode
    - `--no-color` to switch off output colorizing useful for automated output parsing and terminals that don't support colors.
  - Puts rally.conf under virtualenv etc/rally/ so you can have several rally installations in virtualenv
  - Many fixes related to access of different file, like: rally.conf, rally db file in case of sqlite
  - Update pip before Rally installation
  - Fix reinstallation
- **Separated Rally plugins & framework**

  Now plugins are here: https://github.com/openstack/rally/tree/master/rally/plugins

  Plugins are as well separated common/* for common plugins that can be use no matter what is tested and OpenStack related plugins
- **New Rally Task framework**
  - All plugins has the same Plugin base: rally.common.plugin.pluing.Plugin They have the same mechanisms for: discovering, providing information based on docstrings, and in future they will use the same deprecation/rename mechanism.
  - Some of files are moved:
    - rally/benchmark -> rally/task

      This was done to unify naming of rally task command and actually code that implements it.
    - rally/benchmark/sla/base.py -> rally/task/sla.py
    - rally/benchmark/context/base.py -> rally/task/context.py
    - rally/benchmark/scenarios/base.py -> rally/task/scenario.py
    - rally/benchmark/runners/base.py -> rally/task/runner.py
    - rally/benchmark/scenarios/utils.py -> rally/task/utils.py

    This was done to:
    - avoid doing rally.benchmark.scenarios import base as scenario_base
    - remove one level of nesting
    - simplify framework structure
  - Some of classes and methods were renamed
    - Plugin configuration:
      - context.context() -> context.configure()
      - scenario.scenario() -> scenario.configure()
      - Introduced runner.configure()
      - Introduced sla.configure()

      This resolves 3 problems:

- Unifies configuration of different types of plugins
- Simplifies plugin interface
- **Looks nice with new modules path:**

```
>>> from rally.task import scenario
>>> @scenario.configure()
```

– Atomic Actions were changed:

* New rally.task.atomic module

  This allow us in future to reuse atomic actions in Context plugins

* Renames:

  rally.benchmark.scenarios.base.AtomicAction -> rally.task.atomic.ActionTimer

  rally.benchmark.scenarios.base.atomic_action() -> rally.task.atomic.action_timer()

– **Context plugins decide how to map their data for scenario**

Now Context.map_for_scenario method can be override to decide how to pass context object to each iteration of scenario.

– Samples of NEW vs OLD context, sla, scenario and runner plugins:

* Context

```
# Old
from rally.benchmark.context import base

@base.context(name="users", order=100)
class YourContext(base.Context):

    def setup(self):
        # ...

    def cleanup(self):
        # ...

# New
from rally.task import context

@context.configure(name="users", order=100)
class YourContext(context.Context):

    def setup(self):
        # ...

    def cleanup(self):
        # ...

    def map_for_scenario(self):
        # Maps context object to the scenario context object
        # like context["users"] -> context["user"] and so on.
```

* Scenario

```
# Old Scenario
```

```
from rally.benchmark.scenarios import base
from rally.benchmark import validation

class ScenarioPlugin(base.Scenario):

    @base.scenario()
    def some(self):
        self._do_some_action()


    @base.atomic_action_timer("some_timer")
    def _do_some_action(self):
        # ...

# New Scenario

from rally.task import atomic
from rally.task import scenario
from rally.task import validation

# OpenStack scenario has different base now:
# rally.plugins.openstack.scenario.OpenStackScenario
class ScenarioPlugin(scenario.Scenario):

    @scenario.configure()
    def some(self):
        self._do_some_action()

    @atomic.action_timer("some_action")
    def _do_some_action(self):
        # ...
```

* Runner

```
## Old

from rally.benchmark.runners import base

class SomeRunner(base.ScenarioRunner):

     __execution_type__ = "some_runner"

    def _run_scenario(self, cls, method_name, context, args)
      # Load generation

    def abort(self):
      # Method that aborts load generation

## New

from rally.task import runner

@runner.configure(name="some_runner")
class SomeRunner(runner.ScenarioRunner):

    def _run_scenario(self, cls, method_name, context, args)
        # Load generation
```

```
    def abort(self):
        # Method that aborts load generation
```

* SLA

```
# Old

from rally.benchmark import sla

class FailureRate(sla.SLA):
    # ...

# New

from rally.task import sla

@sla.configure(name="failure_rate")
class FailureRate(sla.SLA):
    # ...
```

- **Rally Task aborted command**

  Finally you can gracefully shutdown running task by calling:

```
rally task abort <task_uuid>
```

- **Rally CLI changes**
  - [add] `rally --plugin-paths` specify the list of directories with plugins
  - [add] `rally task report --junit` - generate a JUnit report This allows users to feed reports to tools such as Jenkins.
  - [add] `rally task abort` - aborts running Rally task when run with the `--soft` key, the `rally task abort` command is waiting until the currently running subtask is finished, otherwise the command interrupts subtask immediately after current scenario iterations are finished.
  - [add] `rally plugin show` prints detailed information about plugin
  - [add] `rally plugin list` prints table with rally plugin names and titles
  - [add] `rally verify genconfig` generates tempest.conf without running it.
  - [add] `rally verify install` install tempest for specified deployment
  - [add] `rally verify reinstall` removes tempest for specified deployment
  - [add] `rally verify uninstall` uninstall tempest of specified deployment
  - [fix] `rally verify start --no-use` –no-use was always turned on
  - [remove] `rally use` now each command has subcommand `use`
  - [remove] `rally info`
  - [remove] `rally-manage tempest` now it is covered by `rally verify`
- **New Rally task reports**
  - New code is based on OOP style which is base step to make pluggable Reports
  - Reports are now generated for only one iteration over the resulting data which resolves scalability issues when we are working with large amount of iterations.

  - New Load profiler plot that shows amount of iterations that are working in parallel
  - Failed iterations are shown as a red areas on stacked are graphic.

Non backward compatible changes

- [remove] `rally use` cli command
- [remove] `rally info` cli command
- [remove] `--uuid` parameter from `rally deployment <any>`
- [remove `--deploy-id` parameter from: `rally task <any>`, `rally verify <any>`, `rally show <any>`

Specs & Feature requests

[feature request] Explicitly specify existing users for scenarios

[feature request] Improve install script and add –uninstall and –version

[feature request] Allows specific repos & packages in install-rally.sh

[feature request] Add ability to capture logs from tested services

[feature request] Check RPC queue perfdata

[spec] Refactoring Rally cleanup

[spec] Consistent resource names

Plugins

- **Scenarios**:

  [new] CinderVolumes.create_volume_backup

  [new] CinderVolumes.create_and_restore_volume_backup

  [new] KeystoneBasic.add_and_remove_user_role

  [new] KeystoneBasic.create_and_delete_role

  [new] KeystoneBasic.create_add_and_list_user_roles

  [new] FuelEnvironments.list_environments

  [new] CinderVolumes.modify_volume_metadata

  [new] NovaServers.boot_and_delete_multiple_servers

  [new] NeutronLoadbalancerV1.create_and_list_pool

  [new] ManilaShares.list_shares

  [new] CeilometerEvents.create_user_and_get_event

  [new] CeilometerEvents.create_user_and_list_event_types

  [new] CeilometerEvents.create_user_and_list_events

  [new] CeilometerTraits.create_user_and_list_trait_descriptions

[new] CeilometerTraits.create_user_and_list_traits

[new] NeutronLoadbalancerV1.create_and_delete_pools

[new] NeutronLoadbalancerV1.create_and_update_pools

[new] ManilaShares.create_and_delete_share

[new] ManilaShares.create_share_network_and_delete

[new] ManilaShares.create_share_network_and_list

[new] HeatStacks.create_and_delete_stack

[new] ManilaShares.list_share_servers

[new] HeatStacks.create_snapshot_restore_delete_stack

[new] KeystoneBasic.create_and_delete_ec2credential

[new] KeystoneBasic.create_and_list_ec2credentials

[new] HeatStacks.create_stack_and_scale

[new] ManilaShares.create_security_service_and_delete

[new] KeystoneBasic.create_user_set_enabled_and_delete

[new] ManilaShares.attach_security_service_to_share_network

[new] IronicNodes.create_and_delete_node

[new] IronicNodes.create_and_list_node

[new] CinderVolumes.create_and_list_volume_backups

[new] NovaNetworks.create_and_list_networks

[new] NovaNetworks.create_and_delete_network

[new] EC2Servers.list_servers

[new] VMTasks.boot_runcommand_delete_custom_imagea

[new] CinderVolumes.create_and_update_volume

- **Contexts**:

[new] ManilaQuotas

Add context for setting up Manila quotas: shares, gigabytes, snapshots, snapshot_gigabytes, share_networks

[new] ManilaShareNetworks

Context for share networks that will be used in case of usage deployment with existing users. Provided share networks via context option "share_networks" will be balanced between all share creations of scenarios.

[new] Lbaas

Context to create LBaaS-v1 resources

[new] ImageCommandCustomizerContext

Allows image customization using side effects of a command execution. E.g. one can install an application to the image and use these image for 'boot_runcommand_delete' scenario afterwards.

[new] EC2ServerGenerator

Context that creates servers using EC2 api

[new] ExistingNetwork

This context lets you use existing networks that have already been created instead of creating new networks with Rally. This is useful when, for instance, you are using Neutron with a dumb router that is not capable of creating new networks on the fly.

- **SLA**:

  [remove] max_failure_rate - use failure_rate instead

Bug fixes

90 bugs were fixed, the most critical are:

- Many fixes related that fixes access of rally.conf and DB files
- Incorrect apt-get "-yes" parameter in install_rally.sh script
- Rally bash completion doesn't exist in a virtualenv
- Rally show networks CLI command worked only with nova networks
- RPS runner was not properly generating load
- Check is dhcp_agent_scheduler support or not in network cleanup
- NetworkContext doesn't work with Nova V2.1
- Rally task input file was not able to use jinja2 include directive
- Rally in docker image was not able to
- Rally docker image didn't contain samples
- Do not update the average duration when iteration failed

Documentation

- **Add plugin reference page**

  Rally Plugins Reference page page contains a full list with

- **Add maintainers section on project info page**

  Rally Maintainers section contains information about core contributors of OpenStack Rally their responsibilities and contacts. This will help us to make our community more transparent and open for newbies.

- **Added who is using section in docs**
- **Many small fixes**

Rally v0.1.1

Information

Commits	**32**
Bug fixes	**9**
Dev cycle	**11 days**
Release date	**6/October/2015**

Details

This release contains new features, new 6 plugins, 9 bug fixes, various code and API improvements.

New Features

- **Rally verify generates proper tempest.conf file now**

  Improved script that generates tempest.conf, now it works out of box for most of the clouds and most of Tempest tests will pass without hacking it.

- **Import Tempest results to Rally DB**

  `rally verify import` command allows you to import already existing Tempest results and work with them as regular "rally verify start" results: generate HTML/CSV reports & compare different runs.

API Changes

Rally CLI changes

- [add] `rally verify import` imports raw Tempest results to Rally

Specs & Feature requests

There is no new specs and feature requests.

Plugins

- **Scenarios**:

  [new] NeutronNetworks.create_and_list_floating_ips

  [new] NeutronNetworks.create_and_delete_floating_ips

  [new] MuranoPackages.import_and_list_packages

  [new] MuranoPackages.import_and_delete_package

  [new] MuranoPackages.import_and_filter_applications

  [new] MuranoPackages.package_lifecycle

  [improved] NovaKeypair.boot_and_delete_server_with_keypair

  New argument `server_kwargs`, these kwargs are used to boot server.

  [fix] NeutronLoadbalancerV1.create_and_delete_vips

  Now it works in case of concurrency > 1

- **Contexts**:

  [improved] network

  Network context accepts two new arguments: `subnets_per_network` and `network_create_args`.

  [fix] network

Fix cleanup if nova-network is used. Networks should be dissociate from project before deletion

[fix] custom_image

Nova server that is used to create custom image was not deleted if script that prepares server failed.

Bug fixes

9 bugs were fixed, the most critical are:

- Fix install_rally.sh script

  Set 777 access to /var/lib/rally/database file if system-wide method of installation is used.
- Rally HTML reports Overview table had few mistakes
  - Success rate was always 100%
  - Percentiles were wrongly calculated
- Missing Ironic, Murano and Workload(vm) options in default config file
- `rally verify start` failed while getting network_id
- `rally verify genconfig` hangs forever if Horizon is not available

Documentation

- **Fix project maintainers page**

  Update the information about Rally maintainers
- **Document rally –plugin-paths CLI argument**
- **Code blocks in documentation looks prettier now**

Rally v0.1.2

Information

Commits	**208**
Bug fixes	**37**
Dev cycle	**77 days**
Release date	**23/December/2015**

Details

This release, as well as all previous ones, includes a lot of internal and external changes. Most important of them are listed below.

> **Warning:** Release 0.1.2 is the last release with Python 2.6 support.

Deprecations

- Class *rally.common.objects.Endpoint* was renamed to *Credentials*. Old class is kept for backward compatibility. Please, stop using the old class in your plugins.

  Warning: dict key was changed too in user context from "endpoint" to "credential"

- rally.task.utils: wait_is_ready(), wait_for(), wait_for_delete() deprecated you should use wait_for_status() instead.

Rally Verify

- Added possibility to run Tempest tests listed in a file(–tests-file argument in `verify start`)
- Added possibility to upload Tempest subunit stream logs into data base
- Improvements in generating Tempest config file
- Reworked subunit stream parser
- Don't install Tempest when *rally verify [gen/show]config*
- Rally team tries to simplify usage of each our component. Now Rally verification has some kind of a context like in Tasks. Before launching each verification, Rally checks existence of required resources(networks, images, flavours, etc) in Tempest configuration file and pre-creates them. Do not worry, all these resources will not be forgotten and left, Rally will clean them after verification.

Rally Task

- Add –html-static argument to `rally task report` which allows to generate HTML reports that doesn't require Internet.
- Rally supports different API versions now via api_versions context:

```
CinderVolumes.create_and_delete_volume:
  -
    args:
      size: 1
    runner:
      type: "constant"
      times: 2
      concurrency: 2
    context:
      users:
        tenants: 2
        users_per_tenant: 2
      api_versions:
        cinder:
          version: 2
          service_name: cinderv2
```

- Move rally.osclients.Clients to plugin base

  Rally OSclients is pluggable now and it is very easy to extend OSClient for your cloud out of Rally tree.

- Add 'merge' functionality to SLA

  All SLA plugins should implement merge() method now. In future this will be used for distributed load generation. Where SLA results from different runners will be merged together.

- New optional_action_timer decorator

  Allows to make the methods that can be both atomic_action or regular method. Method changes behavior based on value in extra key "atomic_action"

Rally Certification

- Fix Glance certification arguments
- Add Neutron Quotas only if Neutron service is available

Specs & Feature Requests

- Spec consistent-resource-names:

  Resource name is based on Task id now. It is a huge step to persistence and disaster cleanups.

- Add a spec for distributed load generation:

  https://github.com/openstack/rally/blob/master/doc/specs/in-progress/distributed_runner.rst

- Improvements for scenario output format

  https://github.com/openstack/rally/blob/master/doc/specs/in-progress/improve_scenario_output_format.rst

- Task and Verify results export command

  https://github.com/openstack/rally/blob/master/doc/specs/in-progress/task_and_verification_export.rst

Plugins

- **Scenarios**:
- [new] NovaServers.boot_and_get_console_output
- [new] NovaServers.boot_and_show_server
- [new] NovaServers.boot_server_attach_created_volume_and_resize
- [new] NovaServers.boot_server_from_volume_and_resize
- [new] NeutronSecurityGroup.create_and_delete_security_groups
- [new] NeutronSecurityGroup.create_and_list_security_groups
- [new] NeutronSecurityGroup.create_and_update_security_groups
- [new] NeutronLoadbalancerV1.create_and_delete_healthmonitors
- [new] NeutronLoadbalancerV1.create_and_list_healthmonitors
- [new] NeutronLoadbalancerV1.create_and_update_healthmonitors
- [new] SwiftObjects.list_and_download_objects_in_containers
- [new] SwiftObjects.list_objects_in_containers
- [new] FuelNodes.add_and_remove_node

- [new] CeilometerMeters.list_matched_meters
- [new] CeilometerResource.list_matched_resources
- [new] CeilometerSamples.list_matched_samples
- [new] CeilometerStats.get_stats
- [new] Authenticate.validate_monasca
- [new] DesignateBasic.create_and_delete_zone
- [new] DesignateBasic.create_and_list_zones
- [new] DesignateBasic.list_recordsets
- [new] DesignateBasic.list_zones
- **[fix] CinderVolumes.create_nested_snapshots_and_attach_volume** Remove random nested level which produce different amount of atomic actions and bad reports.
- Support for Designate V2 api
- A lot of improvements in Sahara scenarios
- **Context**:
- [new] api_versions

  Context allows us to setup client to communicate to specific service.
- [new] swift_objects

  Context pre creates swift objects for future usage in scenarios
- [update] sahara_cluster

  It supports proxy server which allows to use single floating IP for whole cluster.
- [fix] cleanup

  Fix cleanup of networks remove vip before port.

Bug fixes

37 bugs were fixed, the most critical are:

- Follow symlinks in plugin discovery
- Use sed without -i option for portability (install_rally.sh)
- Fixed race in rally.common.broker
- Fixed incorrect iteration number on “Failures” Tab
- Fixing issue with create_isolated_networks = False
- Fix docker build command

Documentation

Fixed some minor typos and inaccuracies.

Thanks

We would like to thank Andreas Jaeger for ability to provide Python 2.6 support in this release.

Rally v0.2.0

Information

Commits	**48**
Bug fixes	*6**
Dev cycle	**19 days**
Release date	**1/11/2015**

Details

This release, as well as all previous ones, includes a lot of internal and external changes. Most important of them are listed below.

Warning: Release 0.2.0 doesn't support python 26

Deprecations

- Option –system-wide-install for `rally verify start` was deprecated in favor of –system-wide
- ***rally show* commands were deprecated because of 3 reasons:**
  - It blocks us to make Rally generic testing tool
  - It complicates work on Rally as a Service
  - You can always use standard OpenStack clients to do the same

Rally Verify

- Add "xfail" mechanism for Tempest tests.

  This mechanism allows us to list some tests, that are expected to fail, in a YAML file and these tests will have "xfail" status instead of "fail".

  Use new argument "–xfails-file" of rally verify start command.

Rally Task

- –out argument of *rally task report* is optional now

  If you don't specify –out <file> it will just print the resulting report

- Better scenario output support

  As far as you know each scenario plugin are able to return data as a dict. This dict contained set of key-values {<name>: <float>} where each name was line on graph and each number was one of point. Each scenario run adds a single point for each line on that graph.

  This allows to add extra data to the Rally and see how some values were changed over time. However, in case when Rally was used to execute some other tool and collect it's data this was useless.

  To address this **Scenario.add_output(additive, complete)** was introduced:

  Now it is possible to generate as many as you need graphs by calling this method multiple times. There are two types of graph additive and complete. **Additive** is the same as legacy concept of output data which is generated from results of all iterations, **complete** are used when you would like to return whole chart from each iteration.

  HTML report has proper sub-tabs *Aggregated* and *Per iteration* inside *Scenario Data* tab.

  Here is a simple example how output can be added in any scenario plugin:

```
# This represents a single X point in result StackedArea.
# Values from other X points are taken from other iterations.
self.add_output(additive={"title": "How do A and B changes",
                          "description": ("Trend for A and B "
                                          "during the scenario run"),
                          "chart_plugin": "StackedArea",
                          "data": [["foo", 42], ["bar", 24]]})
# This is a complete Pie chart that belongs to this concrete iteration
self.add_output(
    complete={"title": "",
              "description": ("Complete results for Foo and Bar "
                              "from this iteration"),
              "chart_plugin": "Pie",
              "data": [["foo", 42], ["bar", 24]]})
```

Rally Certification

None.

Specs & Feature Requests

[Spec][Implemented] improve_scenario_output_format

https://github.com/openstack/rally/blob/master/doc/specs/implemented/improve_scenario_output_format.rst

Plugins

- **Scenarios**:
- [new] DesignateBasic.create_and_update_domain
- [improved] CinderVolumes.create_and_attach_volume

Warning: Use "create_vm_params" dict argument instead of `**kwargs` for instance parameters.

- **Context**:
- [improved] images

> **Warning:** The min_ram and min_disk arguments in favor of image_args, which lets the user specify any image creation keyword arguments they want.

Bug fixes

6 bugs were fixed:

- #1522935: CinderVolumes.create_and_attach_volume does not accept additional args for create_volume
- #1530770: "rally verify" fails with error 'TempestResourcesContext' object has no attribute 'generate_random_name'
- #1530075: cirros_img_url in rally.conf doesn't take effective in verification tempest
- #1517839: Make CONF.set_override with parameter enforce_type=True by default
- #1489059: "db type could not be determined" running py34
- #1262123: Horizon is unreachable outside VM when we are using DevStack + OpenStack

Documentation

None.

Thanks

2 Everybody!

Rally v0.3.0

Information

Commits	**69**
Bug fixes	**7**
Dev cycle	**29 days**
Release date	**2/16/2016**

Details

This release, as well as all previous ones, includes a lot of internal and external changes. Most important of them are listed below.

> **Warning:** In this release Rally DB schema migration is introduced. While upgrading Rally from previous versions it is required now to run `rally-manade db upgrade`. Please see 'Documentation' section for details.

CLI changes

- **Warning:** [Removed] `rally info` in favor of `rally plugin *`. It was deprecated for a long time.

- [Modified] `rally deployment check` now prints services, which don't have names, since such services can be used via api_versions context.

- **Warning:** [Modified] `rally verify [re]install` option –no-tempest-venv was deprecated in favor of –system-wide

- [Added] `rally-manage db revision` displays current revision of Rally database schema
- [Added] `rally-manage db upgrade` upgrades pre-existing Rally database schema to the latest revision
- [Added] `rally-manage db downgrade` to downgrades existing Rally database schema to previous revision
- [Added] `rally task export` exports task results to external services (only CLI command introduced, no real service support implemented yet, however one could write own plugins)
- [Added] `rally verify export` exports verification results to external services (only CLI command introduced, no real service support implemented yet, however one could write own plugins)

Rally Deployment

- **Warning:** `fuel` deployment engine is removed since it was outdated and lacked both usage and support

Rally Task

Add custom labels for "Scenario Output" charts

- X-axis label can be specified to add_output() by "axis_label" key of chart options dict. The key is named "axis_label" but not "x_label" because chart can be displayed as table, so we explicitly mention "axis" in option name to make this parameter useless for tables
- Y-axis label can be specified to add_output() by "label" key of chart options dict In some cases this parameter can be used for rendering tables - it becomes column name in case if chart with single iteration is transformed into table
- As mentioned above, if we have output chart with single iteration, then it is transformed to table, because chart with single value is useless
- OutputLinesChart is added, it is displayed by NVD3 lineChart()
- Chart "description" is optional now. Description is not shown if it is not specified explicitly
- Scenario Dummy.add_output is improved to display labels and OutputLinesChart
- Fix: If Y-values are too long and overlaps chart box, then JavaScript updates chart width in runtime to fit width of chart graphs + Y values to their DOM container

Rally Certification

None.

Specs & Feature Requests

- [Spec][Introduced] Export task and verification results to external services

  https://github.com/openstack/rally/blob/master/doc/specs/in-progress/task_and_verification_export.rst

- [Spec][Implemented] Consistent resource names

  https://github.com/openstack/rally/blob/master/doc/specs/implemented/consistent_resource_names.rst

- [Feature request][Implemented] Tempest concurrency

  https://github.com/openstack/rally/blob/master/doc/feature_request/implemented/add_possibility_to_specify_concurrency_for_tempest.rst

Plugins

- **Scenarios**:
- [added] VMTasks.workload_heat
- [added] NovaFlavors.list_flavors
- [updated] Flavors for Master and Worker node groups are now configured separately for SaharaCluster.* scenarios
- **Context**:
- **Warning:** [deprecated] rally.plugins.openstack.context.cleanup in favor of rally.plugins.openstack.cleanup
- [improved] sahara_cluster

  Flavors for Master and Worker node groups are now configured separately in `sahara_cluster` context

Miscellaneous

- Cinder version 2 is used by default
- Keystone API v3 compatibility improved
  - Auth URL in both formats http://foo.rally:5000/v3 and http://foo.rally:5000 is supported for Keystone API v3
  - Tempest configuration file is created properly according to Keystone API version used
- `install_rally.sh --branch` now accepts all git tree-ish, not just branches or tags
- VM console logs are now printed when Rally fails to connect to VM
- Add support for Rally database schema migration (see 'Documentation' section)

Bug fixes

7 bugs were fixed:

- #1540563: Rally is incompatible with liberty Neutron client

  The root cause is that in Neutron Liberty client, the _fx function doesn't take any explicit keyword parameter but Rally is passing one (tenant_id).

- #1543414: The *rally verify start* command fails when running a verification against Kilo OpenStack
- #1538341: Error in logic to retrieve image details in image_valid_on_flavor

Documentation

- Add documentation for DB migration

  https://github.com/openstack/rally/blob/master/rally/common/db/sqlalchemy/migrations/README.rst

Thanks

2 Everybody!

Rally v0.3.1

Information

Commits	**9**
Bug fixes	**6**
Dev cycle	**2 days**
Release date	**2/18/2016**

Details

This release is more about bug-fixes than features.

Warning: Please, update 0.3.0 to latest one.

Features

- Pass api_versions info to glance images context
- [Verify] Don't create new flavor when flavor already exists

Bug fixes

6 bugs were fixed, the most critical are:

- #1545889: Existing deployment with given endpoint doesn't work anymore

- #1547092: Insecure doesn't work with Rally 0.3.0
- #1547083: Rally Cleanup failed with api_versions context in 0.3.0 release
- #1544839: Job gate-rally-dsvm-zaqar-zaqar fails since the recent Rally patch
- #1544522: Non-existing "called_once_with" method of Mock library is used

Rally v0.3.2

Information

Commits	**55**
Dev cycle	**25 days**
Release date	**3/14/2016**

Details

This release, as well as all previous ones, includes a lot of internal and external changes. Most important of them are listed below.

CLI changes

- **Warning:** [Modified] Option '–tempest-config' for 'rally verify reinstall' command was deprecated for removal.

- **Warning:** [Removed] Option *–system-wide-install* was removed from *rally verify* commands in favor of *–system-wide* option.

- **Warning:** [Modified] Step of installation of Tempest during execution of the *rally verify start* command was deprecated and will be removed in the future. Please use *rally verify install* instead.

- Rework commands.task.TaskCommands.detailed. Now output of the command contains the same results as in HTML report.

Rally Verify

- Re-run failed Tempest tests

  Add the ability to re-run the tempest tests that failed in the last test execution. Sometimes Tempest tests fail due to a special temporary condition in the environment, in such cases it is very useful to be able to re-execute those tests.

  Running the following command will re-run all the test that failed during the last test execution regardless of what test suite was run.

```
rally verify start --failing
```

Specs & Feature Requests

- [Spec][Introduced] Refactoring scenario utils
- [Spec] Deployment unification

Plugins

- **Scenarios**:
- [updated] Fix flavor for cloudera manager

  Cloudera manager need master-node flavor
- [added] Expand Nova API benchmark in Rally

  Add support for listing nova hosts, agents, availability-zones and aggregates.
- [updated] Make sure VolumeGenerator uses the api version info while cleanup
- Designate V2 - Add recordset scenarios

  Add create_and_(list|delete)_recordset scenarios Remove the test also that checks the allowed methods, this is in order for us to be able to have a private method _walk_pages that will do fetching of pages for us vs attempting to fetch 1 giant list at once.
- unify *_kwargs name in scenarios

  When running a scenario, *kwargs* is used as default key-word arguments. But in some scenarios, there are more and one services being called, and we use xxx_kwargs for this case.

  However, some xxx_kwargs are not unified for same usage[0]. Unifying these could avoid misleading for end users. Another improvement is to add xxx_kwargs with empty settings for scenario config files.

  [0] http://paste.openstack.org/show/489505/
- **Warning:** Deprecated arguments 'script' and 'interpreter' were removed in favor of 'command' argument.

  VM task scenarios executes a script with a interpreter provided through a formatted argument called 'command' which expects the remote_path or local_path of the script and optionally an interpreter with which the script has to be executed.

Miscellaneous

- Avoid using *len(x)* to check if x is empty

  This cases are using *len()* to check if collection has items. As collections have a boolean representation too, directly check for true / false. And fix the wrong mock in its unit test.
- Fix install_rally.sh to get it to work on MacOSX

  On MacOSX, *mktemp* requires being passed a template. This change modifies the calls to *mktemp* to explicitly pass a template so that the code works on both MacOSX and linux.

- Use new-style Python classes

  There are some classes in the code that didn't inherited from nothing and this is an old-style classes. A "New Class" is the recommended way to create a class in modern Python.A "New Class" should always inherit from *object* or another new-style class.

  Hacking rule added as well.

- Make Rally cope with unversioned keystone URL

  With the change, the client version that's returned is now determined by the keystoneclient library itself based on whether you supply a URL with a version in it or not.

- Fix rally-mos job to work with mos-8.0

  Also remove hardcoded values for some other jobs.

- Add name() to ResourceManager

  This will allow us to perform cleanup based on the name.

- Add task_id argument to name_matches_object

  This will be used to ensure that we are only deleting resources for a particular Rally task.

- Extend api.Task.get_detailed

  Extend api.Task.get_detailed with ability to return task data as dict with extended results.

Bug fixes

The most critical fixed bugs are:

- #1547624: Wrong configuration for baremetal(ironic) tempest tests
- #1536800: openrc values are not quoted

  The openrc file created after rally deployment –fromenv did not quote the values for environment variables that will be exported.

- #1509027: Heat delete_stack never exits if status is DELETE_FAILED
- #1540545: Refactored atomic action in authenticate scenario
- #1469897: Incompatible with Keystone v3 argument in service create scenario
- #1550262: Different results in `rally task detailed`, `rally task report` and `rally task status` commands.
- #1553024: Backward incompatible change in neutronclient(release 4.1.0) broke Tempest config generation to support latest neutronclient.

Documentation

- Add documentation for DB migration
- Make documentation for output plugins
  - Add descriptive docstrings for plugins based on OutputChart
  - Register these plugins in Rally Plugins Reference

- Documentation tox fix

  Added information about debugging unit test with tox. Replace 3 references to py26 with py34 to reflect current rally tox configuration.

- Change structure of rally plugin and plugin references page
- Update the scenario development, runner and context sections
- The design of Rally Plugins Reference page was improved
- New page was added - CLI references

Thanks

To Everybody!

Rally v0.3.3

Information

Commits	**20**
Dev cycle	**10 days**
Release date	**3/24/2016**

Details

A half of patches relate to Cleanup. We have once again proved that ideal stuff can be improved. :)

Specs & Feature Requests

- [Spec][Introduced] Improve atomic actions format

Plugins

- **Cleanups**:
- Use proper attribute to get heat stack name
- Always assign a name to created images.

  This is necessary for name-based cleanup. If a name is not specified, one will be generated automatically.

- Improve filtering glance images in case of V2 API
- Delete only images created by images context

  Since the images context allows creating images with arbitrary names, name-based cleanup won't work for it, so we have to delete the exact list of images that it created instead.

- New config option to set cleanup threads

  Allow the user to change the number of cleanup threads via the rally config. When scaling out to thousands of instances, the cleanup can take forever with the static 20 threads.

- Add inexact matching to name_matches_object

  This will support places where we create resources with names that start with a given name pattern, but include some additional identifier afterwards. For instance, when bulk creating instances, Nova appends a UUID to each instance name.

- **Scenarios**:
- Add sample of template for testing for testing heat caching.
- Introduced new scenario Dummy.dummy_random_action. It is suitable for demonstration of upcoming trends report.
- **Contexts**:

  api_versions context was extended to support switch between Keystone V2 and V3 API versions. Now it is possible to use one Rally deployment to check both Keystone APIs.

- **Newcomer in the family**:

  All ResourceType classes are pluggable now and it is much easier to use and extend them.

  Warning: Decorator `rally.task.types.set` is deprecated now in favor of `rally.task.types.convert`.

Bug fixes

- #1536172: rally deployment destroy failed with traceback for failed deployments. At current moment it is impossible to delete deployment if for some reason deployment engine plugin cannot be found, because exception will be thrown.

Documentation

- Remove extra link in *All release notes*

  Previously, two links for latest release were presented.

- Update release notes for 0.3.2
  - Fixed indents for warning messages
  - Fixed all references

Thanks

To Everybody!

Rally v0.4.0

Information

Commits	**76**
Bug fixes	**12**
Dev cycle	**28 days**
Release date	**4/18/2016**

Details

> **Warning:** Rally DB schema was changed since previous release. See HOWTO about updating your database.

CLI changes

- Add status messages of db migration process
- Display task errors in human-friendly form
- Support OS_PROJECT_NAME as well as OS_TENANT_NAME

Messages

- Removed deprecation warning in case of transmitted "name" attribute while creation neutron resources.

  > **Warning:** Deprecated code was deleted.

- Suppress warning insecure URL messages

  Do not spam end users by insecure URL messages because it is quite valid case in testing process

Database

While preparing for deployment refactoring:

- db schema was changed;
- migration with new column *credentials* to deployment model was added;
- columns *users* and *admin* were dropped.

Rally Task

- Remove deprecated scenario output mechanism via returing value

  > **Warning:** Deprecated code was deleted.

- Friendlier error message with empty task file

  This is particularly useful when a Jinja2 template results in an empty task. The current error message isn't very helpful:

  Task config is invalid: *'NoneType' object has no attribute 'get'*

- Add Heat template validator

Plugins

Scenarios:

- Extend VM bind actions with "pause_unpause", "suspend_resume", "lock_unlock", "shelve_unshelve".
- Add exact error message into VMTasks.runcommand_heat scenario
- Add heat scenarios: output-show, output-list

  Current patch contains 4 scenarios from heat repo:

  - output-show for old algorithm
  - output-show for new algorithm
  - output-list for old algorithm
  - output-list for new algorithm

Contexts:

- Reduce default speed of users creation in users context from 30 to 20 by default.

SLAs:

- *NEW!!* MaxAverageDurationPerAtomic : Maximum average duration of one iterations atomic actions in seconds.

  Plugin Reference

Reports:

- Improve results calculation in charts.Table
- Use int instead of float for Y axis. It's number of parallel iterations and it can't be float.
- Remove accuracy that makes no sense, and creates a lot of noise on this graph
- Include failed iterations as well, otherwise we will calculate load incorrectly
- Graph should start from 0 (begging of experiment)
- Add 2 points at the end of graph to get at the end of graph 0 iterations in parallel

Task Exporter:

In previous release we introduced new mechanism to export results in various external systems and various formats.

In this release, we added first plugin for this stuff - *file_exporter*

Services:

Remove hardcoded timeout from heat service

Utils:

Make glance web uploads streamable

Without this change entire file get's downloaded into memory and can cause issues.

Rally Verify

- Set time precision to 3 digits (instead of 5) after dot.
- Don't use "–parallel" flag when concurrency == 1

  If concurrency equals to 1, it means that we use only one thread to run Tempest tests and the "–parallel" flag is not needed.

Plugin for DevStack

- Support to be enabled with different plugin name

  Allow rally to be installed by devstack through a different plugin name, e.g:

```
enable_plugin test-rally http://github.com/rally/rally.git master
```

- Removed uncalled code

  Devstack won't "source plugin.sh source" any more.

Bug fixes

12 bugs were fixed:

- X-Fail mechanism did not work for TestCase which failed on setUp step

  If Tempest fails in a test's setUpClass(), there is only one subunit event for each TestCase. In this case, Rally did not check partial test with x-fail list and marked test as "fail" insted of "x-fail".

  Launchpad bug-report #1568133

- Weak isolation of scenario arguments between iterations

  Input arguments for sub-task were shared between all iterations. Rally team found one scenario which modified mutable input variable.

  Affected scenario: NeutronNetworks.create_and_update_ports

- Incompatible filters between V1 and V2 for Glance images listing

  Glance V1 and V2 have different filters. For example, "owner" is a separate kwarg in V1, not a generic filter. Also, visibility has different labels in different APIs. We modified our Glance wrapper to support Glance V2 format of filters for both V1 and V2

- Wrong way to store validation errors

  Results of failed task validations saved in incorrect format. It broke and made un-userfriendly *rally task detailed* command.

  Launchpad bug-report #1562713

- Hardcoded task's status in *rally task results*

  If there are no results for task, *rally task results* printed message that task has failed status, but it can be not true(tasks in running state do not have results).

  Launchpad bug-report #1539096

- Tempest context failed to create network resources

  While we merged improvement for keystoneclient, we used wrong way to obtain tenant id in TempestContext.

  Launchpad bug-report #1550848

- Tasks based on Tempest failed to parse execution time.

  There is an ability in Rally to launch tasks based on Tempest. Since launch of Tempest is just subprocess, it is needed to parse subunit to set correct atomic actions.

  There was an issue while converting task execution time.

  Launchpad bug-report #1566712

- JSONSchema huge impact on task performance

  Before runner sent data to engine we were checking jsonschema. This operation is very expensive and in some cases it can take a lot of time.

  Here are test results, with Dummy.dummy_output scenario, sleep 0.5s (added manually), 8000 iterations, 400 in parallel:

  - **on master branch before the fix:** Load duration: 117.659588099 Full duration: 227.451056004
  - **on master before the fix but remove jsonschema validation in scenario:** Load duration: 12.5437350273 Full duration: 128.942219973
  - **on this patch before the fix (pure python validation):** Load duration: 11.5991640091 Full duration: 22.7199981213

- Wrong Calculation of running iterations in parallel

  Load profile chart was calculated wrongly. It showed more running iterations in parallel than actually are running.

- Rally did not show "missing argument" error raised by argparse while parsing cli args

  Launchpad bug-report #1562916

- Issue while checking required arguments in CLI

  There was a possible issue in case of several required arguments

  Launchpad bug-report #1555764

- Prepare step of verification did not check visibility of obtained image

  When we request a list of images to choose one of them for tests, we should make sure all images are active and they are PUBLIC. If images are not public, we will have failures of Tempest tests as described in the bug.

  Launchpad bug-report #1564431

Thanks

2 Everybody!

Rally v0.5.0

Information

Commits	**175**
Bug fixes	**19**
Dev cycle	**93 days**
Release date	**7/20/2016**

Details

This release took much more time than we expected, but we have a lot of reasons for such delay and if you look at our change-log, you will understand them.:)

Here is a quick introduction:

- To make our releases as much as possible stable, we added upper limits for each of our requirements;
- A lot of deprecated lines of code were removed, so be careful;
- Statistics trends for given tasks were introduced;
- Support for tempest plugins was added;
- Several new pages at docs.

Specs & Feature Requests

- [Introduced && implemented] Introduce class-based scenario implementation
- [Introduced] Rally Task Validation refactoring
- [Introduced] Scaling & Refactoring Rally DB
- [Introduced] SLA Performance degradation plugin

Logging

- disable urllib3 warnings only if the library provide them

Database

[doesn't require migration] Transform DB layer to return dicts, not SQLAlchemy models

Rally Deployment

- Support single-AZ deployment

  This supports the case where OpenStack is deployed with a single AZ for both controller(s) and compute(s), and not all hosts in the AZ that contains an instance are guaranteed to have the nova-compute service.

- Extend creation from environment with several new vars
  - OS_ENDPOINT_TYPE/OS_INTERFACE
  - OS_USER_DOMAIN_NAME

  - OS_PROJECT_DOMAIN_NAME
- Improve devstack plugin for Keystone V3

Rally Task

NEW!! Statistics trends for given tasks.

Rally Verify

- Remove '–tempest-config' arg from 'reinstall' command

  Warning: Using *–tempest-config* is became an error from this release. Use *rally verify genconfig* cmd for all config related stuff.

- Don't install Tempest when *rally verify start*

  Warning: Use should use *rally verify install* cmd to install tempest now

- Add ability to setup version of Tempest to install

  CLI argument to setup version

- Configure 'aodh' service in 'service_available' section
- Check existence of Tempest-tree in *rally verify discover* cmd
- Make Tempest work with auth url which doesn't include keystone version

  Tempest needs /v2.0 and /v3 at the end of URLs. Actually, we can't fix Tempest, so we extend our configuration module with workaround which allow to specify auth_url without version in rally deployment config.

- Use default list of plugins for sahara
- Move tempest related options of rally configuration to separate section.
- *NEW!!* Support for tempest plugins.

  CLI argument to install them

Plugins

In this release we are happy to introduce new entity - plugins Base classes

We have a lot of base plugin entities: Context, Scenario, SLA and etc. Sometimes plugins of different bases can have equal names(i.e ceilometer OSClient and ceilometer Context). It is normal and we should allow such conflicts. To support such cases we introduced new entity - plugin base. Statements of plugin bases:

- Each plugin base is unique entity;
- Names of plugin bases can't conflict with each other;
- Names of two or more plugins in one plugin base can't conflict with each other(in case of same namespace).
- Names of two or more plugins in different plugin base can conflict

Current list of plugin bases:

- rally.task.context.Context
- rally.task.scenario.Scenario
- rally.task.types.ResourceType
- rally.task.exporter.TaskExporter
- rally.task.processing.charts.Chart
- rally.task.runner.ScenarioRunner
- rally.task.sla.SLA
- rally.deployment.serverprovider.provider.ProviderFactory
- rally.deployment.engine.Engine
- rally.osclients.OSClient

OSClients

- *NEW!!* Support for Senlin client
- *NEW!!* Support for Gnocchi client
- *NEW!!* Support for Magnum client
- *NEW!!* Support for Watcher client
- Transmit endpoint_type to saharaclient

Scenarios:

- *NEW!!*:
- Authenticate.validate_ceilometer
- CinderVolumes.create_volume_from_snapshot
- CinderVolumes.create_volume_and_clone
- NovaFlavors.create_and_list_flavor_access
- NovaFlavors.create_flavor
- NovaServers.boot_and_update_server
- NovaServers.boot_server_from_volume_snapshot
- [Sahara] Add configs to MapR plugin
- Extend CinderVolumes.create_and_upload_volume_to_image with "image" argument

  Plugin Reference
- Deprecate Dummy.dummy_with_scenario_output scenario in favor of Dummy.dummy_output

  > **Warning:** Dummy.dummy_with_scenario_output scenario will be removed after several releases

  Deprecated Plugin Reference New Plugin Reference
- Extend CinderVolumes.create_volume_and_clone with nested_level

  Add nested_level argument for nested cloning volume to new volume

- Extend CinderVolumes.create_nested_snapshots_and_attach_volume

  Two new arguments were added: create_volume_kwargs and create_snapshot_kwargs

  > **Warning:** All arguments related to snapshot creation should be transmitted only via create_snapshot_kwargs.

- Introduce new style of scenarios - class based.

  Spec Reference

- Improve report for VMTasks.boot_runcommand_delete
- [Sahara] Added 5.5.0 version for cdh-plugin and 1.6.0 version for spark
- Extend boot_server_from_volume_and_delete, boot_server_from_volume, boot_server_from_volume_and_live_migrate, boot_server_from_volume_snapshot scenarios of NovaServers class with "volume_type" parameter.

Contexts:

- *NEW!!*:
  - Cinder volume_types
  - Murano environments
  - Heat dataplane
- Use Broker Pattern in Keystone roles context
- Use immutable types for locking context configuration

  Since context configuration passed to Context.__init__() was a mutable type (dict or list), sometimes we had unexpected changes done by unpredictable code (for example, in wrappers).

- Add possibility to balance usage of users

  For the moment all users for tasks were taken randomly and there was no way to balance them between tasks. It may be very useful when we have difference between first usage of tenant/user and all consecutive. In this case we get different load results.

  Therefore, "users" context was extended with new config option 'user_choice_method' that defines approach for picking up users.

  Two values are available: - random - round_robin

  Default one is compatible with old approach - "random".

- Make sahara_image and custom_image contexts glance v2 compatible
- Extend servers context with "nics" parameter
- Extend network context with "dns_nameservers" parameter
- Extend volume context with "volume_type" parameter

Cleanup:

- Mark several cleanup resources as tenant_resource

  Nova servers and security groups are tenant related resources, but resource decorator missed that fact which makes cleanup tries to delete one resources several times.

- Turn off redundant nova servers cleanup for NovaFlavors.list_flavors scenario
- Add neutron cleanup for NeutronSecurityGroup.create_and_delete_security_groups

Exporter:

Rename task-exporter "file-exporter" to "file".

> **Warning:** "file-exporter" is deprecated and will be removed in further releases.

Types:

Remove deprecated types.

> **Warning:** you should use rally.task.types.convert instead of rally.task.types.set decorator

Validators

- Add a required_api_version validator
- Add validators for scenario arguments

Utils:

Use glance wrapper where appropriate to support compatibility between V1 and V2

Bug fixes

19 bugs were fixed:

- Wrong arguments order of Keystone wrapper in case of V2 and V3
- AttributeError while disabling urllib3 warnings on old installations

  Launchpad bug-report #1573650
- install_rally.sh script is failed while obtaining setuptools
- "-inf" load duration in case of wrong runner plugin and failed start of contexts
- Strange input task in the report

  Launchpad bug-report #1570328
- Wrong behaviour of boot_server_from_volume scenarios in case of booting server from image.

  The arg of image must be None, when booting server from volume. Otherwise still boot server from image.

  Affected scenarios: NovaServers.boot_server_from_volume NovaServers.boot_server_from_volume_and_delete NovaServers.boot_server_from_volume_and_resize NovaServers.boot_server_from_volume_and_live_migrate

  Launchpad bug-report #1578556
- Weak validation of json schema of RPS runner

  JSON Schema of RPS runner doesn't have "required" field. It means that users are able to pass wrong configs and we will have runtime error while running task.
- Rally doesn't take cacert setting while creating keystone session

  Launchpad bug-report #1577360
- Heat scenarios fail when API uses TLS

  Launchpad bug-report #1585456

- Example in comment of context manila_share_networks wrong

  Launchpad bug-report #1587164

- There is no way to get UUID of a verification after it is created by "rally verify start" or "rally verify import_results" when –no-use is set

  Launchpad bug-report #1587034

- Exposed ssh timeout and interval in vm scenario

  Launchpad bug-report #1587728

- Ceilometer scenario doesn't require "ceilometer" ctx

  Launchpad bug-report #1557642

- "servers" context requires setting network id for multiple possible networks found.

  Launchpad bug-report #1592292

- nested_level data type incorrect in create_nested_snapshots_and_attach_volume

  Launchpad bug-report #1594656

- Rally cleanup servers raises exception

  Launchpad bug-report #1584104

- Stopping server is redundant before cold-migrating server

  Launchpad bug-report #1594730

- existing_users context doesn't work in case of Keystone v3
- Whether validates flavor's disk or not depends on booting type of the instance

  Launchpad bug-report #1596756

Documentation

- Re-use openstack theme for building docs outside rtd.

  Rally Docs at docs.openstack.org

- Add page for Verification component

  RTD page for Verification component

- Add glossary page

  RTD page for Glossary

- Adjust docs reference to "KeystoneBasic.authenticate" scenario

  Step 6. Aborting load generation on success criteria failure

Thanks

2 Everybody!

Rally v0.6.0

Overview

Release date	**9/05/2016**

Details

Common

- Added Python 3.5 support
- Sync requirements with OpenStack global-requirements
- Start using latest way of authentication - keystoneauth library
- Start porting all scenario plugins to class-based view.

Specs & Feature Requests

- [Implemented] SLA Performance degradation plugin
- [Proposed] New Tasks Configuration section - hook

Database

- disable db downgrade api
- [require migration] upgrade deployment config

Docker image

- Add sudo rights to rally user Rally is a pluggable framework. External plugins can require installation of additional python or system packages, so we decided to add sudo rights.
- Move from ubuntu:14.04 base image to ubuntu:16.04 . Ubuntu 16.04 is current/latest LTS release. Let's use it.
- pre-install vim Since there are a lot of users who like to experiment and modify samples inside container, rally team decided to pre-install vim
- configure/pre-install bash-completion Rally provides bash-completion script, but it doesn't work without installed *bash-completion* package and now it is included in our image.

Rally Deployment

- Add strict jsonschema validation for ExistingCloud deployments. All incorrect and unexpected properties will not be ignored anymore. If you need to store some extra parameters, you can use new "extra" property.
- Fix an issue with endpoint_type. Previously, endpoint type was not transmitted to keystone client. In this case, keystoneclient used default endpoint type (for different API calls it can differ). Behaviour after the fix:

  - None endpoint type -> Rally will initialize all clients without setting endpoint type. It means that clients will choose what default values for endpoint type use by itself. Most of clients have "public" as default values. Keystone use "admin" or "internal" by default.
  - Not none endpoint type -> Rally will initialize all clients with this endpoint. Be careful, by default most of keystone v2 api calls do not work with public endpoint type.

Rally Task

- [core] Iterations numbers in logging and reports must be synchronized. Now they start from 1 .
- [config] users_context.keystone_default_role is a new config option (Defaults to "member") for setting default user role for new users in case of Keystone V3.
- [Reports] Embed Rally version into HTML reports This adds Rally version via meta tag into HTML reports:

  <meta name="generator" content="Rally version {{ version }}">
- [Reports] Expand menu if there is only one menu group
- [logging] Remove deprecated rally.common.log module
- [Trends][Reports] Add success rate chart to trends report
- [Reports] Hide menu list if there is no data at all

Rally Verify

- Updating Tempest config file
- Some tests (for boto, horizon, etc.) were removed from Tempest and now there is no need to keep the corresponding options in Tempest config file.
- Some options in Tempest were moved from one section to another and we should to do the corresponding changes in Rally to be up to date with the latest Tempest version.
- Adding '–skip-list' arg to *rally verify start* cmd

  CLI argument for –skip-list
- *NEW!!*:
- Command for plugin listing
- Command to uninstall plugins
- Rename and deprecated several arguments for *rally verify start* cmd:
- tests-file -> load-list
- xfails-file -> xfail-list

Plugins

Scenarios:

- Extend Sahara scenarios with autoconfig param

  Affected plugins:
- SaharaClusters.create_and_delete_cluster

- SaharaClusters.create_scale_delete_cluster
- SaharaNodeGroupTemplates.create_and_list_node_group_templates
- SaharaNodeGroupTemplates.create_delete_node_group_templates
- *NEW!!*:
- MonascaMetrics.list_metrics
- SenlinClusters.create_and_delete_cluster
- Watcher.create_audit_template_and_delete
- Watcher.create_audit_and_delete
- Watcher.list_audit_templates
- Rename **murano.create_service** to **murano.create_services** atomic action

SLA:

NEW!!: performance degradation plugin

Contexts:

- *NEW!!*:
- Monasca monasca_metrics
- Senlin profiles
- Watcher audit_templates
- Extend manila_share_networks context with share-network autocreation support.
- Extend volumes context to allow volume_type to be None to allow using default value

Bug fixes

- [existing users] Quota context does not restore original settings on exit

  Launchpad bug-report #1595578
- [keystone v3] Rally task's test user role setting failed

  Launchpad bug-report #1595081
- [existing users] context cannot fetch 'tenant' and 'user' details from cloud deployment

  Launchpad bug-report #1602157
- UnboundLocalError: local variable 'cmd' referenced before assignment

  Launchpad bug-report #1587941
- [Reports] Fix trends report generation if there are n/a results

Documentation

- Add page about task reports

  RTD page for reports

Thanks

2 Everybody!

Rally v0.7.0

Overview

Release date	**10/11/2016**

Details

Specs & Feature Requests

- [Used] Ported all rally scenarios to class base

  Spec reference

- [Implemented] New Plugins Type - Hook

Database

Warning: Database schema is changed, you must run rally-manage db upgrade to be able to use old Rally installation with latest release.

- [require migration] fix for wrong format of "verification_log" of tasks
- [require migration] remove admin_domain_name from OpenStack deployments

Rally Deployment

- Remove admin_domain_name from openstack deployment Reason: admin_domain_name parameter is absent in Keystone Credentials.

Rally Task

- [Trends][Reports] Use timestamps on X axis in trends report
- [Reports] Add new OutputTextArea chart plugin

  New chart plugin can show arbitrary textual data on "Scenario Stata -> Per iteration" tab.

  This finally allows to show non-numeric data like IP addresses, notes and even long comments.

  Plugin Dummy.dummy_output is also updated to provide demonstration.

- [cli] Add version info to *rally task start* output

- [api] Allow to delete stopped tasks without force=True

  It is reasonable to protect deletion of running tasks (statuses INIT, VERIFYING, RUNNING, ABORTING and so on...) but it is strange to protect deletion for stopped tasks (statuses FAILED and ABORTED). Also this is annoying in CLI usage.

- Added hooks and triggers.

  Hook is a new entity which can be launched on specific events. Trigger is another new entity which processes events and launches hooks. For example, hook can launch specific destructive action - just execute cli command(we have sys_call hook for this task) and it can be launched by simple trigger on specific iteration(s) or time (there is event trigger).

Rally Verify

Scenario tests in Tempest require an image file. Logic of obtaining this image is changed:

- If CONF.tempest.img_name_regex is set, Rally tries to find an image matching to the regex in Glance and download it for the tests.
- If CONF.tempest.img_name_regex is not set (or Rally didn't find the image matching to CONF.tempest.img_name_regex), Rally downloads the image by the link specified in CONF.tempest.img_url.

Plugins

Scenarios:

- *Removed*: Dummy.dummy_with_scenario_output

  It was deprecated in 0.5.0

  > **Warning:** This plugin is not available anymore in 0.7.0

- *NEW!!*:
- MagnumClusterTemplates.list_cluster_templates
- MagnumClusters.list_clusters
- MagnumClusters.create_and_list_clusters
- NovaAggregates.create_aggregate_add_and_remove_host
- NovaAggregates.create_and_list_aggregates
- NovaAggregates.create_and_delete_aggregate
- NovaAggregates.create_and_update_aggregate
- NovaFlavors.create_and_get_flavor
- NovaFlavors.create_flavor_and_set_keys
- NovaHypervisors.list_and_get_hypervisors
- NovaServers.boot_server_associate_and_dissociate_floating_ip
- KeystoneBasic.authenticate_user_and_validate_token

Contexts:

- *NEW!!*:
- Manila manila_security_services
- Magnum cluster_templates
- Magnum clusters

OSClients:

Port all openstack clients to use keystone session.

Bug fixes

- [tasks] rally task detailed incorrect / inconsistent output

  Launchpad bug-report #1562713

Thanks

2 Everybody!

Rally v0.8.0

Overview

Release date	**1/25/2017**

Details

Specs & Feature Requests

- [Implemented] Refactor Verification Component
- [Implemented] Scaling & Refactoring Rally DB

Installation

We switched to use bindep library for checking required system packages. All our dependencies moved to separate file (like requirements.txt for python packages) bindep.txt.

Database

Warning: Database schema is changed, you must run rally-manage db upgrade to be able to use old Rally installation with latest release.

- change structure of database to be more flexible
- save raw task results in chunks (see raw_result_chunk_size option of [DEFAULT] rally configuration section)

- add db revision check in rally API, so it is impossible to use rally with wrong db now.

Rally API

Single entry point for Rally API is added - rally.api.API . Old API classes (`rally.api.Task`, `rally.api.Verification`, `rally.api.Deployment`) are deprecated now.

Rally CLI

- `rally task sla_check` is deprecated now in favor of `rally task sla-check`
- Deprecated category `rally show` was removed.
- *rally plugin list* is extended with plugin base column

Task Component

- [Random names] scenario for checking performance of generate_random_name method is added to our CI with proper SLA. Be sure, whatever number of random names you need, it will not affect performance of Rally at all, we checked.
- [atomic actions] scenario for checking performance of calculating atomic actions is added to our CI with proper SLA. Be sure, whatever number atomics you have in scenarios, it will not affect performance of Rally at all, we checked.
- [services] new entity is introduced for helping to provide compatibility layer between different API versions of one service.

Verification component

We completely redesign the whole Verification component. For more details see our new docs for that component

Unfortunately, such big change could not be done in backward compatible way, so old code is not compatible with new one. See HowTo migrate from Verification component 0.7.0 to 0.8.0

Plugins

Services:

- Glance:

  Switched from V1 to V2 API by default.

- Keystone:
- Transmit endpoint_type to keystoneclient
- Full keystone V3 support

Scenarios:

- *Updated*:
- The meaning of the volume_type argument is changes in CinderVolumes.create_snapshot_and_attach_volume scenario. It should contain actual volume type instead of boolean value to choose random volume type.

- Extend GlanceImages.create_image_and_boot_instances with create_image_kwargs and boot_server_kwargs arguments.
- *NEW!!*:
- CeilometerAlarms.create_and_get_alarm
- CinderVolumeBackups.create_incremental_volume_backup
- CinderVolumeTypes.create_and_delete_volume_type
- CinderVolumeTypes.create_volume_type_and_encryption_type
- CinderVolumes.create_and_accept_transfer
- CinderVolumes.create_and_get_volume
- CinderVolumes.create_volume_and_update_readonly_flag
- CinderVolumes.list_transfers
- CinderVolumes.list_types
- KeystoneBasic.create_and_get_role
- ManilaShares.create_and_list_share
- ManilaShares.set_and_delete_metadata
- MistralExecutions.create_execution_from_workbook
- MistralExecutions.list_executions
- NeutronLoadbalancerV2.create_and_list_loadbalancers
- NeutronNetworks.create_and_show_network
- NeutronNetworks.list_agents
- NovaAggregates.create_aggregate_add_host_and_boot_server
- NovaAggregates.create_and_get_aggregate_details
- NovaFlavors.create_and_delete_flavor
- NovaFlavors.create_flavor_and_add_tenant_access
- NovaHosts.list_and_get_hosts
- NovaHypervisors.list_and_get_uptime_hypervisors
- NovaHypervisors.list_and_search_hypervisors
- NovaHypervisors.statistics_hypervisors
- NovaSecGroup.boot_server_and_add_secgroups
- NovaServerGroups.create_and_list_server_groups
- Quotas.nova_get

Hooks:

- *NEW!!*:
- fault_injection

Runners

- *Updated*:

- RPS runner is extended with ability to increase 'rps' value by arithmetic progression across certain duration. Now it can be also a dict specifying progression parameters:

```
rps": {
    "start": 1,
    "end": 10,
    "step": 1,
    "duration": 2
}
```

  This will generate rps value: `start, start + step, start + 2 * step, .., end` across certain 'duration' seconds each step. If iteration count not ended at the last step of progression, then rps will continue to generate with "end" value. Note that the last rps could be generated smaller.

Fixed bugs

- [hooks] incorrect encoding of stdout/stderr streams opened by sys_call hook for py3
- [hooks] sorting Hook column at HTML report doesn't work
- [tasks][scenarios][neutron] L3 HA: Unable to complete operation on subnet

  Launchpad bug-report #1562878
- [tasks] JSON report doesn't save order of atomics
- [tasks][cleanup][nova] Failed to remove aggregate which has hosts in it
- [tasks] –abort-on-sla-failure mechanism works only for current workload, but does not stop the next ones.
- [hooks] hooks section isn't displayed in HTML report

Thanks

2 Everybody!

Rally v0.8.1

Overview

Release date	**1/27/2017**

Details

Fix for python requirements list.

Plugins

Scenarios:

- *Updated*:
- Use new network for each subnet at NeutronNetworks.create_and_list_subnets scenario.

- *NEW!!*:
- CinderVolumeTypes.create_and_list_encryption_type
- Quotas.cinder_get

Thanks

2 Everybody!

Rally v0.9.0

Overview

Release date	**3/20/2017**

Details

Command Line Interface

- *rally plugin list* now does not contain hidden plugins.

Task component

- Added check for duplicated keys in task files.
- The order of subtasks (scenarios/workloads) is not ignored any more. You can generate whatever you want load or use that feature for up the cloud (put small scenario to the start of task to wake up the cloud before the real load).
- Information about workload creation is added to HTML-reports.
- Task statuses is changed to be more clear and cover more cases:
- `verifying` is renamed to `validating`.
- `failed` is divided for 2 statuses - `validation_failed`, which means that task did not pass validation step, and `crashed`, which means that something went wrong in rally engine.
- Our awesome cleanup become more awesome! The filter mechanism is improved to discover resources in projects created only by Rally (it works for most of resources, except several network-related). It makes possible to run Rally with existing users in real tenants without fear to remove something important.

Verification component

- Fixed an issue with missed tests while listing all supported tests of specified verifier.
- Fixed an issue with displaying the wrong version of verifier in case of cloning from the local directory.
- Extend rally verify rerun with `--detailed`, `--no-use`, `--tag` and `--concurrency` arguments.
- Add output examples for JSON and JUnit-XML reporters.

Plugins

Contexts

- Extend cinder quotas to support `backups` and `backup_gigabytes`.

Deployment Engines:

Updated Extend DevstackEngine with `enable_plugin` option.

OpenStack clients:

- Extend support for auth urls like `https://example.com:35357/foo/bar/v3`
- Pass endpoint type to heatclient

Scenarios:

- *NEW!!*
- CinderVolumeTypes.create_and_delete_encryption_type
- CinderVolumeTypes.create_and_set_volume_type_keys
- KeystoneBasic.create_and_list_roles
- KeystoneBasic.create_and_update_user
- NovaKeypair.create_and_get_keypair
- NovaServers.resize_shutoff_server
- VMTasks.dd_load_test
- *UPDATED!!*
- Extend VMTasks.boot_runcommand_delete to display just raw text output of executed command.
- *DELETED*

  Scenario VMTasks.boot_runcommand_delete_custom_image is removed since VMTasks.boot_runcommand_delete covers the case of that particular scenario without adding any complexity.

Validators:

- Extend `required_contexts` validator to support `at least one of the` logic.
- Fix a bunch of JSON schemas which are used for validation of all plugins.

Documentation

We totally reworked Plugins Reference page. Now it looks more like Command Line Interface, which means that you can get links for particular parameter of particular plugin.

Also, you can find expected parameters and their types of all contexts, hooks, SLAs and so on! Most of them still miss descriptions, but we are working on adding them.

Fixed bugs

- [osclients] Custom auth mechanism was used for zaqarclient instead of unified keystonesession, which led to auth errors at some envs.
- [plugins] During running CinderVolumes.create_and_restore_volume_backup scenario we had a race problem with backup deleting due to wrong check of backup status.

- [plugins][verifications] Jenkins expexts “classname” JUnitXML attribute instead of “class_name”.

Thanks

2 Everybody!

Rally v0.9.0

Overview

Release date	**3/20/2017**

Details

Command Line Interface

- *rally plugin list* now does not contain hidden plugins.

Task component

- Added check for duplicated keys in task files.
- The order of subtasks (scenarios/workloads) is not ignored any more. You can generate whatever you want load or use that feature for up the cloud (put small scenario to the start of task to wake up the cloud before the real load).
- Information about workload creation is added to HTML-reports.
- Task statuses is changed to be more clear and cover more cases:
- `verifying` is renamed to `validating`.
- `failed` is divided for 2 statuses - `validation_failed`, which means that task did not pass validation step, and `crashed`, which means that something went wrong in rally engine.
- Our awesome cleanup become more awesome! The filter mechanism is improved to discover resources in projects created only by Rally (it works for most of resources, except several network-related). It makes possible to run Rally with existing users in real tenants without fear to remove something important.

Verification component

- Fixed an issue with missed tests while listing all supported tests of specified verifier.
- Fixed an issue with displaying the wrong version of verifier in case of cloning from the local directory.
- Extend rally verify rerun with `--detailed`, `--no-use`, `--tag` and `--concurrency` arguments.
- Add output examples for JSON and JUnit-XML reporters.

Plugins

Contexts

- Extend cinder quotas to support `backups` and `backup_gigabytes`.

Deployment Engines:

Updated Extend DevstackEngine with `enable_plugin` option.

OpenStack clients:

- Extend support for auth urls like `https://example.com:35357/foo/bar/v3`
- Pass endpoint type to heatclient

Scenarios:

- *NEW!!*
- CinderVolumeTypes.create_and_delete_encryption_type
- CinderVolumeTypes.create_and_set_volume_type_keys
- KeystoneBasic.create_and_list_roles
- KeystoneBasic.create_and_update_user
- NovaKeypair.create_and_get_keypair
- NovaServers.resize_shutoff_server
- VMTasks.dd_load_test
- *UPDATED!!*
- Extend VMTasks.boot_runcommand_delete to display just raw text output of executed command.
- *DELETED*

  Scenario VMTasks.boot_runcommand_delete_custom_image is removed since VMTasks.boot_runcommand_delete covers the case of that particular scenario without adding any complexity.

Validators:

- Extend `required_contexts` validator to support `at least one of the` logic.
- Fix a bunch of JSON schemas which are used for validation of all plugins.

Documentation

We totally reworked Plugins Reference page. Now it looks more like Command Line Interface, which means that you can get links for particular parameter of particular plugin.

Also, you can find expected parameters and their types of all contexts, hooks, SLAs and so on! Most of them still miss descriptions, but we are working on adding them.

Fixed bugs

- [osclients] Custom auth mechanism was used for zaqarclient instead of unified keystonesession, which led to auth errors at some envs.
- [plugins] During running CinderVolumes.create_and_restore_volume_backup scenario we had a race problem with backup deleting due to wrong check of backup status.

- [plugins][verifications] Jenkins expexts “classname” JUnitXML attribute instead of “class_name”.

Thanks

2 Everybody!

Index