
Rally Documentation

Release 3.1.0

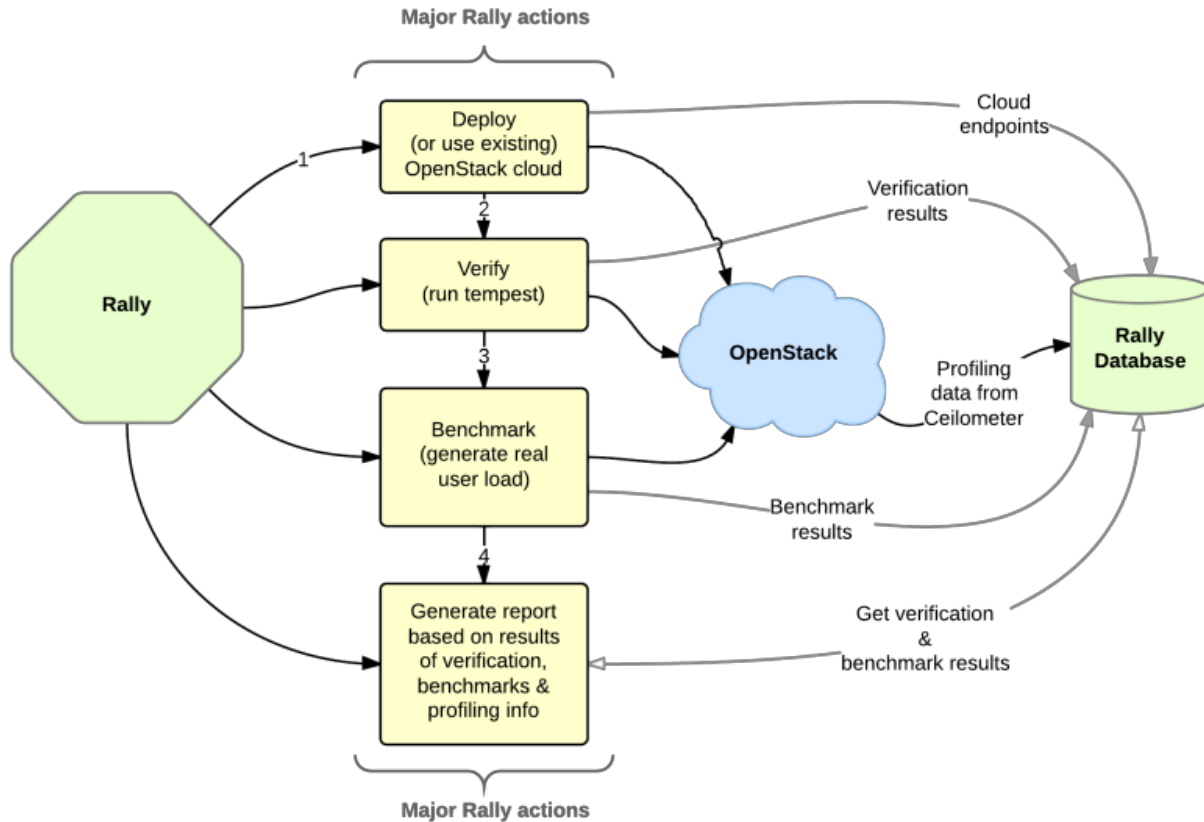
OpenStack Foundation

May 08, 2020

Contents

1	Contents	3
1.1	Rally project overview	3
1.2	Installation and upgrades	18
1.3	Quick start	21
1.4	Command Line Interface	68
1.5	Task Component	94
1.6	Verification Component	106
1.7	Rally Plugins	138
1.8	Contribute to Rally	177
1.9	Request New Features	180
1.10	Project Info and Release Notes	185
	Index	259

OpenStack is, undoubtedly, a really *huge* ecosystem of cooperative services. **Rally** is a **testing 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, testing & 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.



1.1 Rally project overview

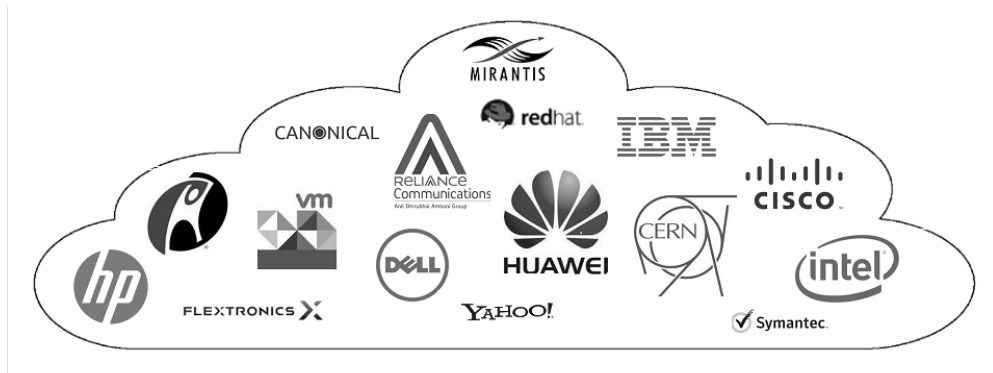
- *Overview*

1.1.1 Overview

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

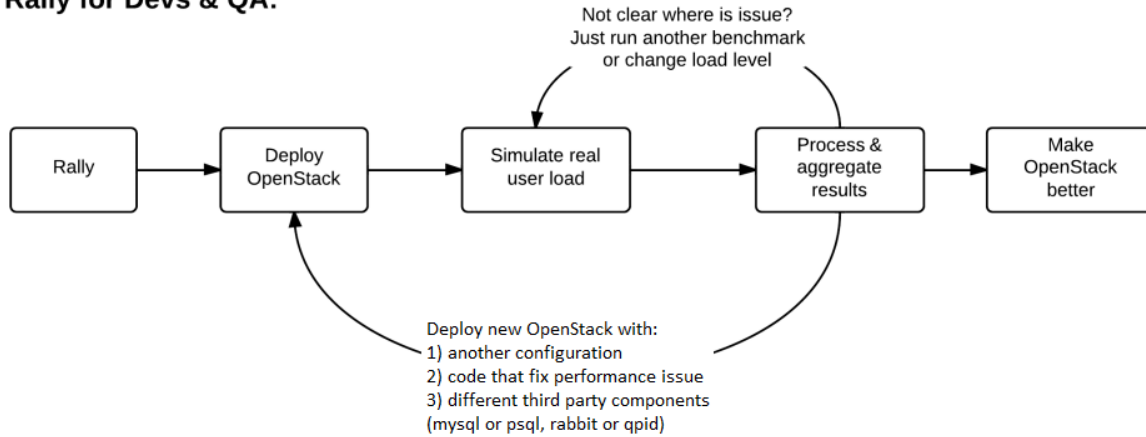
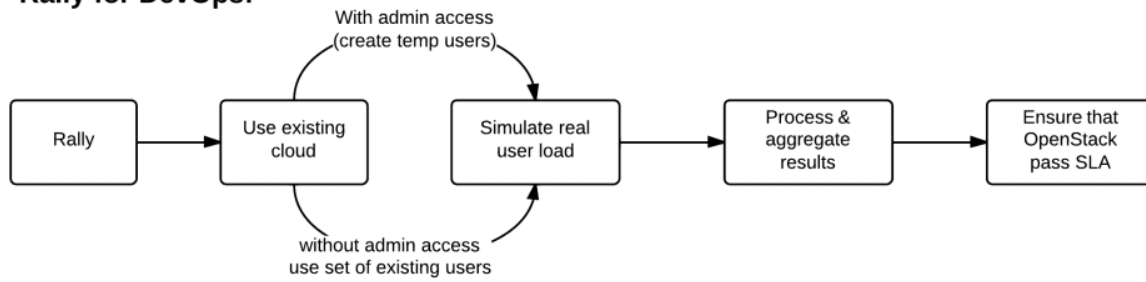
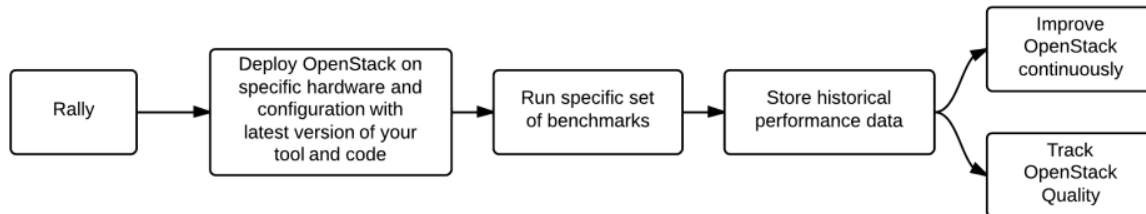
Who Is Using Rally

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:

Rally for Devs & QA:**Rally for DevOps:****Rally CI/CD:**

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;

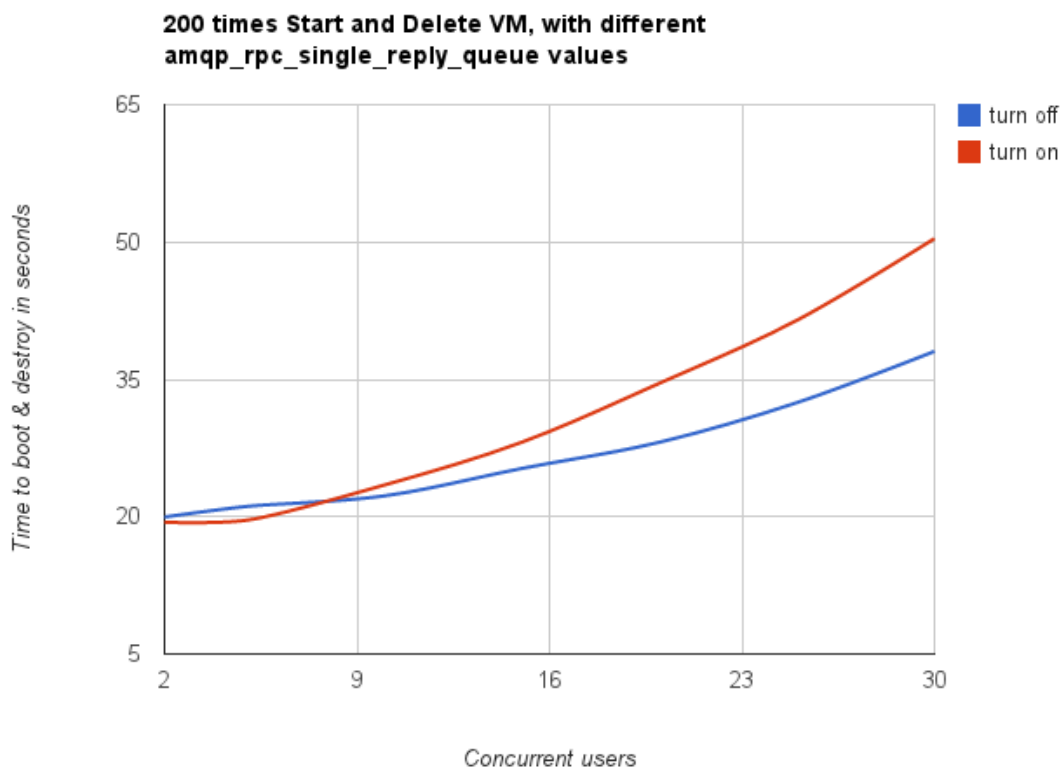
- Check performance of basic cloud operations in case of different loads.

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* 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.

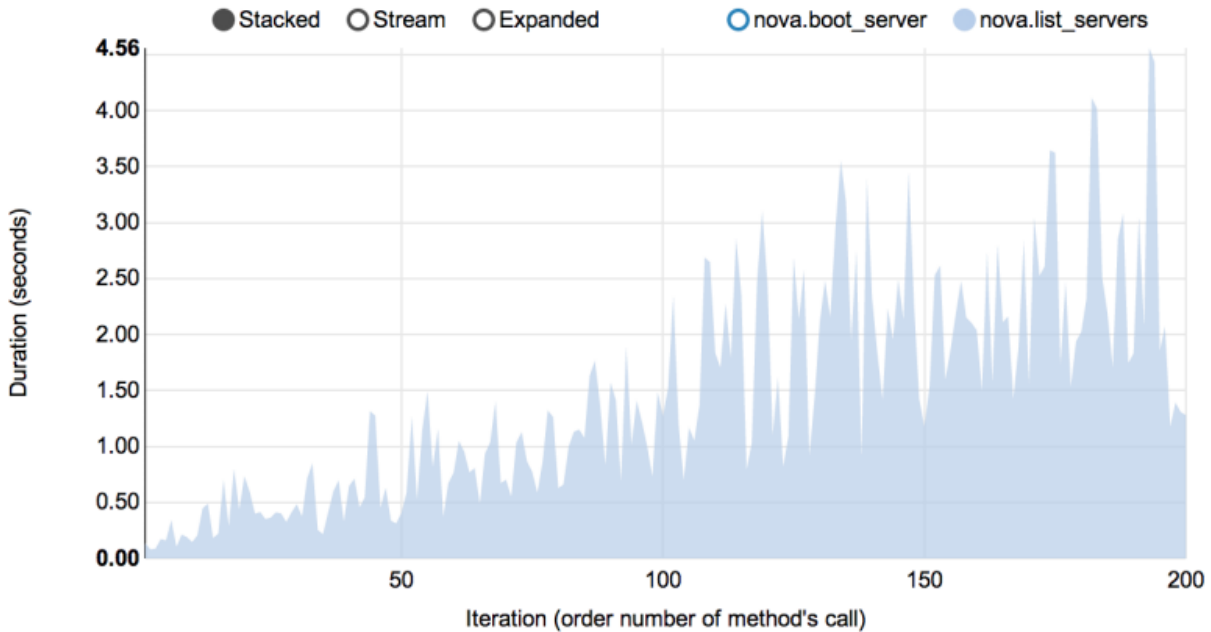
Performance of Nova list command

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

- **Task context:** 1 temporary OpenStack user.

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

During the execution of this task, 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:

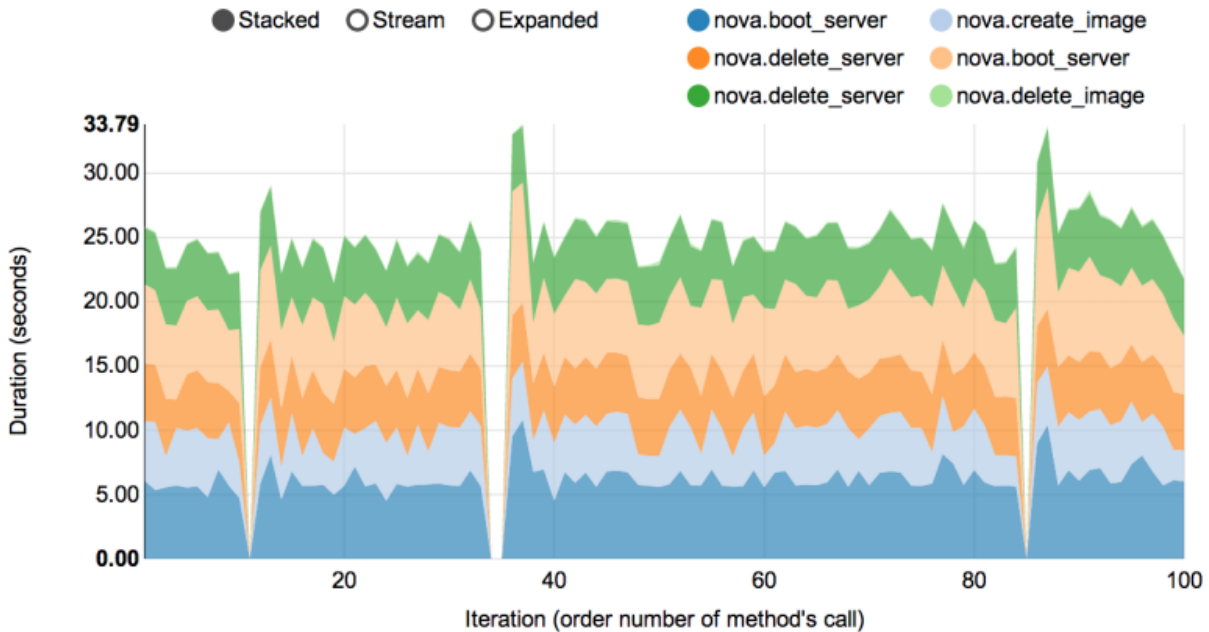


Complex scenarios

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 scenario as a whole, but also that of single atomic actions. As a result, Rally also displays the atomic actions performance data for each scenario iteration in a quite detailed way:

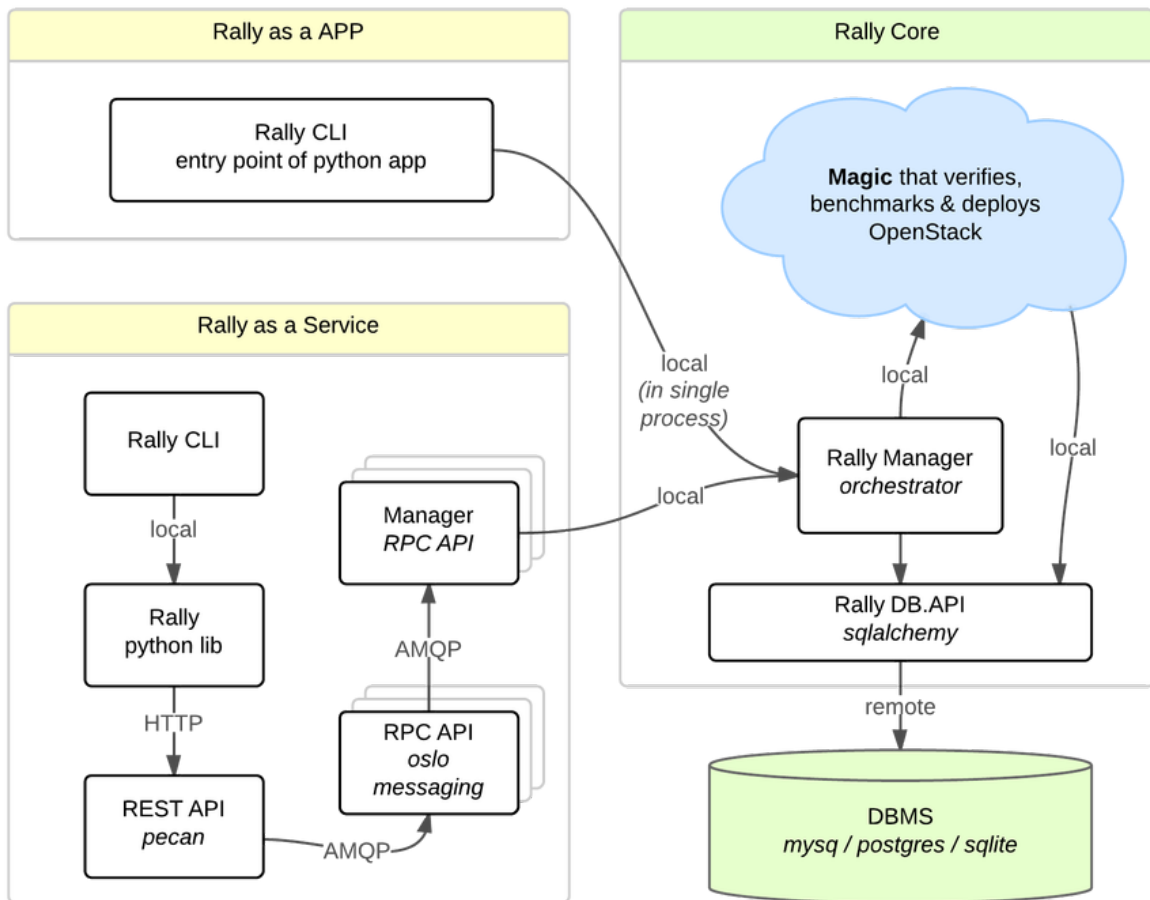


Architecture

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.

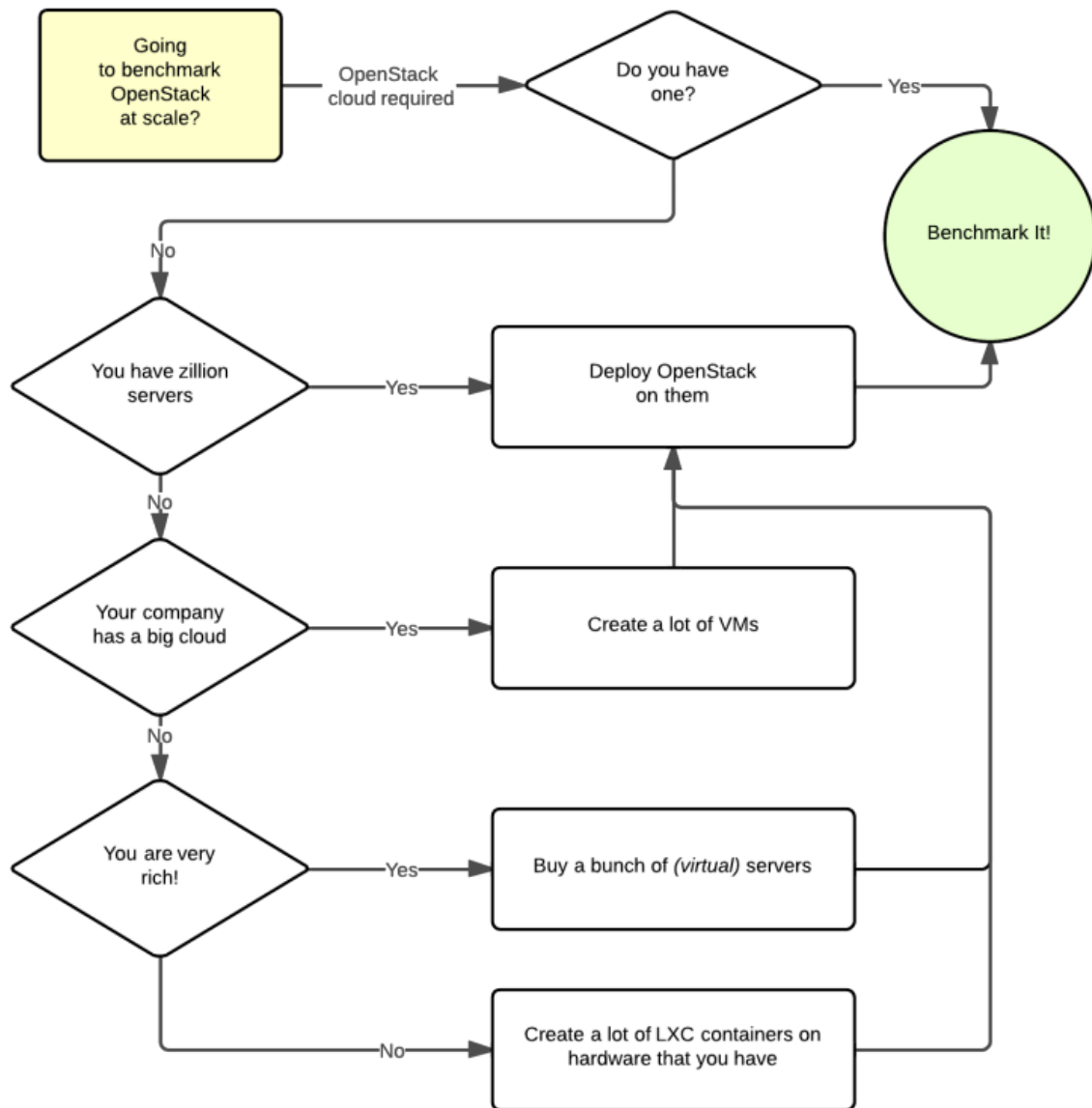
The diagram below shows how this is possible:



The actual **Rally core** consists of 3 main components, listed below in the order they go into action:

1. **Deploy** - store credentials about your deployments, credentials are used by verify and task commands. It has plugable mechanism that allows one to implement basic LCM for testing environment as well.
2. **Verify** - wraps unittest based functional testing framework to provide complete tool with result storage and reporting. Currently has only plugin implemented for OpenStack Tempest.
3. **Task** - framework that allows to write parametrized plugins and combine them in complex test cases using YAML. Framework allows to produce all kinds of tests including functional, concurrency, regression, load, scale, capacity and even chaos testing.

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 testing clouds at scale**.



1.1.2 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-discuss@lists.openstack.org with tag [Rally]).

- *Common*
- *Deployment*

- *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.

1.1.3 User stories

Rally has made it possible to find performance bugs and validate improvements for different OpenStack installations. You can read some stories below:

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
  ↳ }}
```

ac- tion	min (sec)	avg (sec)	max (sec)	90 per- centile	95 per- centile	suc- cess	count	apache keystone	enabled	pub- lic_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
  ↳ }}
```

ac- tion	min (sec)	avg (sec)	max (sec)	90 per- centile	95 per- centile	suc- cess	count	apache keystone	enabled	pub- lic_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
  ↳ '}}
```

ac- tion	min (sec)	avg (sec)	max (sec)	90 per- centile	95 per- centile	suc- cess	count	apache keystone	enabled	pub- lic_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 testing 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 measure performance 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 test case, 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

```
$ glance image-show d1c116f4-3c38-4aa6-8fa1-f7a28c4e72a6
+-----+-----+
| Property                | Value                                |
+-----+-----+
| checksum                | 053ad369d58aa98afb1d355aa16b0663   |
| container_format        | bare                                 |
| created_at              | 2018-01-09T06:23:18Z                 |
| disk_format              | qcow2                                |
| id                      | d1c116f4-3c38-4aa6-8fa1-f7a28c4e72a6 |
| min_disk                | 0                                     |
| min_ram                 | 0                                     |
| name                    | TestVM                               |
| owner                   | 01cb845eee6449cea4381865a1270736   |
| protected               | False                                |
+-----+-----+
```

(continues on next page)

(continued from previous page)

size	5254208	
status	active	
tags	[]	
updated_at	2018-01-09T06:23:18Z	
virtual_size	None	
visibility	public	
+-----+-----+-----+		

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 per-centile	95 per-centile	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 per-centile	95 per-centile	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

1.2 Installation and upgrades

1.2.1 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
```

(continues on next page)

(continued from previous page)

<code>-s, --system</code>	Install system-wide.
<code>-d, --target DIRECTORY</code>	Install Rally virtual environment into DIRECTORY. (Default: /root/rally if not root).
<code>--url</code>	Git repository public URL to download Rally from. This is useful when you have only installation script and
<code>↪want</code>	to install Rally from custom repository. (Default: https://git.openstack.org/openstack/rally). (Ignored when you are already in git repository).
<code>--branch</code>	Git branch name, tag (Rally release), commit hash, ref, or
<code>↪other</code>	tree-ish to install. (Default: master) Ignored when you are already in git repository.
<code>-f, --overwrite</code>	Deprecated. Use <code>-r</code> instead.
<code>-r, --recreate</code>	Remove target directory if it already exist. If neither <code>'-r'</code> nor <code>'-R'</code> is set default behaviour is to ask.
<code>-R, --no-recreate</code>	Do not remove target directory if it already exist. If neither <code>'-r'</code> nor <code>'-R'</code> is set default behaviour is to ask.
<code>-y, --yes</code>	Do not ask for confirmation: assume a 'yes' reply to every question.
<code>-D, --dbtype TYPE</code>	Select the database type. TYPE can be one of 'sqlite', 'mysql', 'postgresql'. Default: sqlite
<code>--db-user USER</code>	Database user to use. Only used when <code>--dbtype</code> is either 'mysql' or 'postgresql'.
<code>--db-password PASSWORD</code>	Password of the database user. Only used when <code>--dbtype</code> is either 'mysql' or 'postgresql'.
<code>--db-host HOST</code>	Database host. Only used when <code>--dbtype</code> is either 'mysql' or 'postgresql'
<code>--db-name NAME</code>	Name of the database. Only used when <code>--dbtype</code> is either 'mysql' or 'postgresql'
<code>-p, --python EXE</code>	The python interpreter to use. Default: /usr/bin/python.
<code>--develop</code>	Install Rally with editable source code try. (Default: false)
<code>--no-color</code>	Disable output coloring.

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 db recreate
```

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 xrrally/xrrally-openstack in the commands below.

The Rally Docker image is configured to store the database in the user's home directory - /home/rally/data/rally.sqlite. For persistence of these data, you may want to keep this directory outside of the container. This may be done via 2 ways:

- use a docker image. In this case you do not need to initialize the database

```
docker volume create --name rally_volume
docker run -v rally_volume:/home/rally/data xrally/xrally-openstack deployment_
↳ create --name "foo"
```

- mount the directory to container. In this case, there is ability to transmit task files inside the container, but you will need to initialize the database by yourself

```
sudo mkdir /var/lib/rally_container
sudo chown 65500 /var/lib/rally_container
docker run -v /var/lib/rally_container:/home/rally/data xrally/xrally-openstack_
↳ db create
docker run -v /var/lib/rally_container:/home/rally/data xrally/xrally-openstack_
↳ deployment create --name "foo"
```

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_container`. 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 pre created tasks at `/home/rally/source` 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/>

1.2.2 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 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 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/alembic.ini revision -m <Message>
```

or

```
alembic --config rally/common/db/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.

1.3 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).

1.3.1 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 *testing OpenStack with Rally!*

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

- *Installing rally-openstack package*
- *Registering an OpenStack deployment in Rally*
- *Running Rally Tasks*
- *Report generation*

In this demo basic operations in Rally are performed, such as adding OpenStack cloud deployment, running task against it and generating report.

It's assumed that you have gone through *Step 0. Installation* and have an already existing OpenStack deployment with Keystone available at `<KEYSTONE_AUTH_URL>`.

Installing rally-openstack package

First, you have to provider Rally with `rally-openstack` package, to be done with `pip install rally-openstack` command.

Registering an OpenStack deployment in Rally

After successful installation, you have to provide Rally with an OpenStack deployment that should be tested. 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                    |                     |
+-----+-----+-----+-----+-----+
(continues on next page)
```

(continues on next page)

(continued from previous page)

```
| 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 tasks or verify commands are going to be run against it. Later in tutorial is described how to use multiple deployments.

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

```
$ rally deployment check
keystone endpoints are valid and following services are available:
+-----+-----+-----+
| Service | Service 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 |
+-----+-----+-----+
```

Running Rally Tasks

Now that we have a working and registered deployment, we can start testing it. The sequence of subtask to be launched by Rally should be specified in a *task input file* (either in *JSON* or in *YAML* format). Let's try one of the task sample 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": {
```

(continues on next page)

(continued from previous page)

```

        "users": {
            "tenants": 3,
            "users_per_tenant": 2
        }
    }
}
]
}

```

To start a 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
-----

Running Task... 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    |

```

(continues on next page)

(continued from previous page)

```

| 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 generate a JUnit report, run:
    rally task export 6fd9a19f-5cf8-4f76-ab72-2e34bb1d4996 --type junit-xml
    --to output.xml

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

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 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 might use the the following commands:

```

$ . ~/.rally/openrc
$ openstack image list
+-----+-----+-----+-----+-----+-----+
| ID                                     | Name                                     | Status |
+-----+-----+-----+-----+-----+-----+
| 30dc3b46-4a4b-4fcc-932c-91fa87753902 | cirros-0.3.4-x86_64-uec                | active |
| d687fc2a-75bd-4194-90c7-1619af255b04 | cirros-0.3.4-x86_64-uec-kernel         | active |
| c764d543-027d-47a3-b46e-0c1c8a68635d | cirros-0.3.4-x86_64-uec-ramdisk        | active |
+-----+-----+-----+-----+-----+-----+

$ openstack flavor list
+-----+-----+-----+-----+-----+-----+
| ID | Name      | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+
| 1  | m1.tiny   | 512 | 1    | 0         | 1     | True      |
| 2  | m1.small  | 2048 | 20   | 0         | 1     | True      |
| 3  | m1.medium | 4096 | 40   | 0         | 2     | True      |
| 4  | m1.large  | 8192 | 80   | 0         | 4     | True      |
| 42 | m1.nano   | 64   | 0    | 0         | 1     | True      |
| 5  | m1.xlarge | 16384 | 160  | 0         | 8     | True      |
| 84 | m1.micro  | 128  | 0    | 0         | 1     | True      |
+-----+-----+-----+-----+-----+-----+

```

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 task data. To create and open at once such a report for the last task you have launched, call:

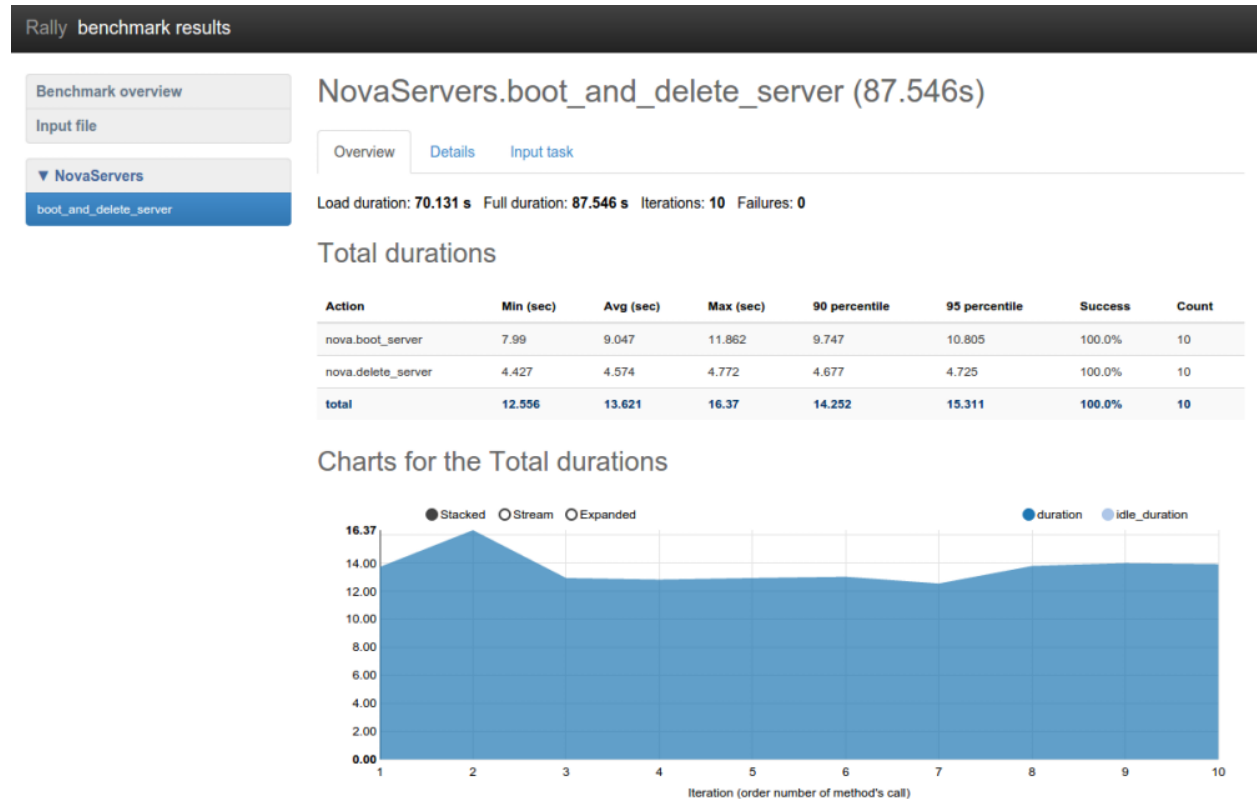
```
rally task report --out=report1.html --open
```

This is going produce an HTML page with the overview of all the scenarios that you’ve included into the last 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*):



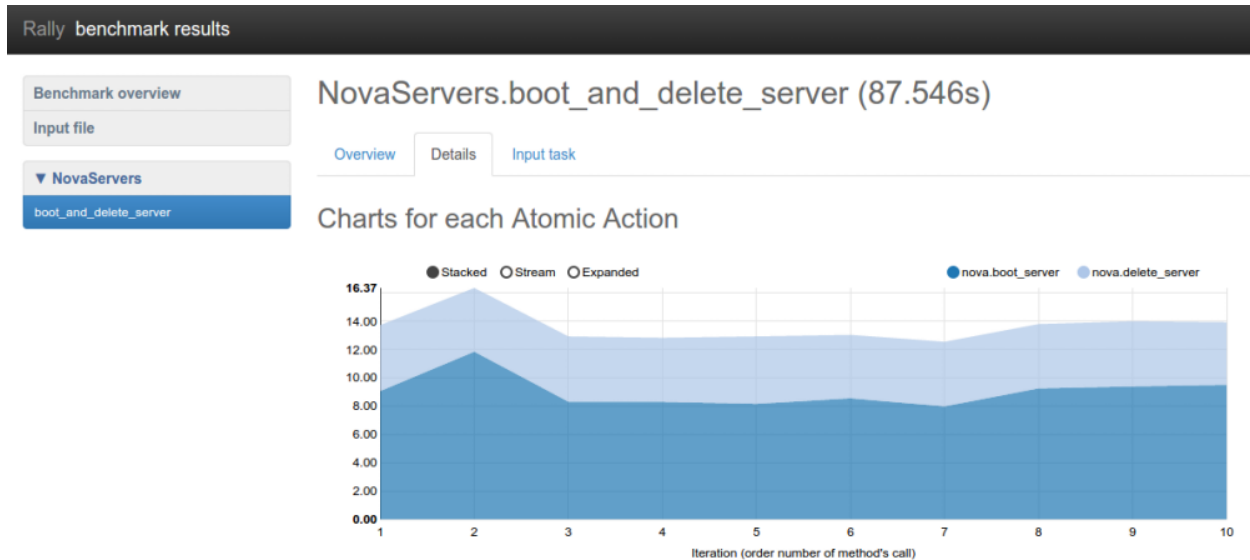
This aggregating table shows the duration of the load produced by the corresponding scenario (“*Load duration*”), the overall subtask execution time, including the duration of context creation (“*Full duration*”), the number of iterations of each scenario (“*Iterations*”), the type of the load used while 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 task 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 subtasks 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>": [<config>, <config2>, ...]
  "<ScenarioName2>": [<config>, ...]
}
```

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

```
{
  "args": { <scenario-specific arguments> },
  "runner": { <type of the runner and its specific parameters> },
```

(continues on next page)

(continued from previous page)

```

    "context": { <contexts needed for this scenario> },
    "sla": { <different SLA configs> }
}

```

Multiple subtasks 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 task as usually:

```

$ rally task start multiple-scenarios.json
...
+-----+-----+-----+-----+-----+-----+-----+
↪--+-----+-----+

```

(continues on next page)

(continued from previous page)

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 task reports can be generated by typing **rally task report --out=report_name.html**. This command works even if not all subtask are done.

Let's take a look at the report overview page for a task with multiple subtasks

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

Rally benchmark results							
Benchmark overview							
Input file							
▶ KeystoneBasic							
▶ NovaServers							
Scenario ▲	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)	
KeystoneBasic.create_delete_user	5.721	10.081	10	constant	0	✓	
NovaServers.boot_and_delete_server	84.196	102.033	10	constant	0	✓	

Multiple configurations of the same scenario

Yet another thing you can do in Rally is to launch **the same 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 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     |
+-----+-----+-----+-----+-----+-----+
(continues on next page)
```

(continued from previous page)

```

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
```

Rally benchmark results							
<div> <div>Benchmark overview</div> <div>Input file</div> <div> <div>▼ NovaServers</div> <div>boot_and_delete_server</div> <div>boot_and_delete_server [2]</div> </div> </div>							
Benchmark overview							
Scenario ▲	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)	
NovaServers.boot_and_delete_server	73.234	91.169	10	constant	0	✓	
NovaServers.boot_and_delete_server-2	71.030	88.026	10	constant	0	✓	

Step 3. Running Task against OpenStack with read only users

- *Motivation*
- *Registering deployment with existing users in Rally*
- *Running tasks that uses 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 test your OpenStack cloud:

1. *Read-only Keystone Backends*: creating temporary users for running 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 deployment with existing users in Rally

The information about existing users in your OpenStack cloud should be passed to Rally at the *deployment initialization step*. 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*:

```
{
  "openstack": {
    "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 existing_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
```

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

Running tasks that uses existing users

After you have registered a deployment with existing users, don't forget to remove the “*users*” context from your task input file 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 is going to use “*b1*” and “*b2*” for running subtask instead of creating the temporary users:

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

It goes without saying that support of running 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 subtasks

- *SLA - Service-Level Agreement (Success Criteria)*
- *Checking SLA*
- *SLA in task report*

SLA - Service-Level Agreement (Success Criteria)

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

To configure the SLA, add the “*sla*” section to the configuration of the corresponding subtask (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** task 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 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",
```

(continues on next page)

(continued from previous page)

```

        "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 is going to 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 against the success criteria you defined in the task:

```
$ rally task sla_check
```

subtask	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 100.0% 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
```

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

Rally benchmark results							
Benchmark overview		Benchmark overview					
Input file							
► Dummy							
Scenario	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)	
Dummy.dummy	0.186	4.539	5	constant	0	✓	
Dummy.dummy_exception	0.110	6.013	5	constant	5	✗	

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

Benchmark overview

Input file

▼ Dummy

dummy

dummy_exception

Dummy.dummy_exception (6.013s)

Overview

Failures

Input task

Load duration: 0.110 s Full duration: 6.013 s Iterations: 5 Failures: 5

Service-level agreement

Criterion	Detail	Success
failure_rate	Maximum failure rate percent 0.0% failures, minimum failure rate percent 0% failures, actually 100.0%	False

Total durations

Action	Min (sec)	Avg (sec)	Max (sec)	90 percentile	95 percentile	Success	Count
total						0	5

Success criteria present a very useful concept that enables not only to analyze the outcome of your 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*
- *Using the default values*
- *Advanced 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

```

(continues on next page)

(continued from previous page)

```
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:
```

(continues on next page)

(continued from previous page)

```

    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-alf1-3cc6f45ce23f: started
-----

Running Task... 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"

```

(continues on next page)

(continued from previous page)

```
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:
```

(continues on next page)

(continued from previous page)

```

    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

```

(continues on next page)

(continued from previous page)

```

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

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

Running Task... 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 testing procedures in Rally when shared with others.

Step 6. Aborting load generation on success criteria failure

Testing 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 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

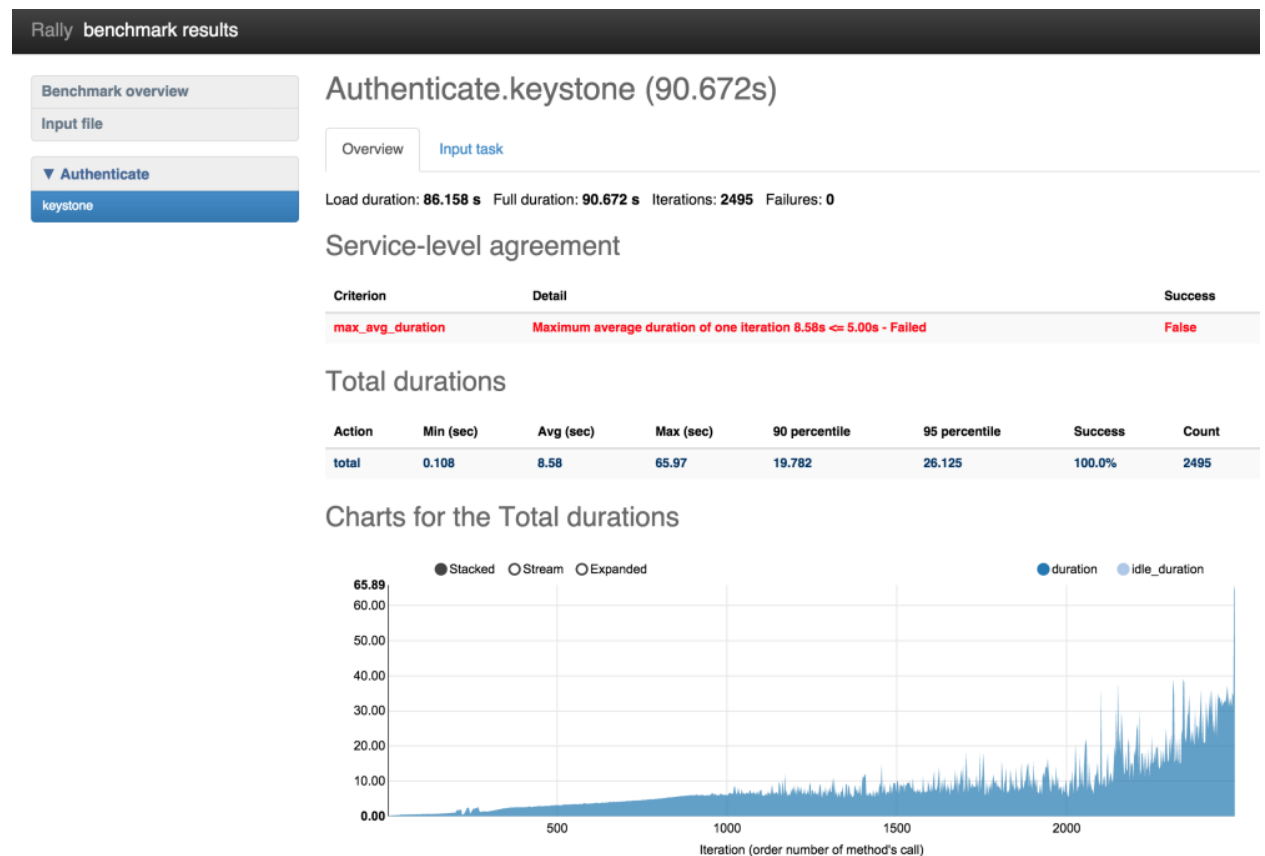
....
+-----+-----+-----+-----+-----+-----+-----+
| action | min (sec) | avg (sec) | max (sec) | 90 percentile | 95 percentile | success |
+-----+-----+-----+-----+-----+-----+-----+
| total  | 0.108     | 8.58      | 65.97     | 19.782        | 26.125        | 100.0%  |
+-----+-----+-----+-----+-----+-----+-----+
| 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:

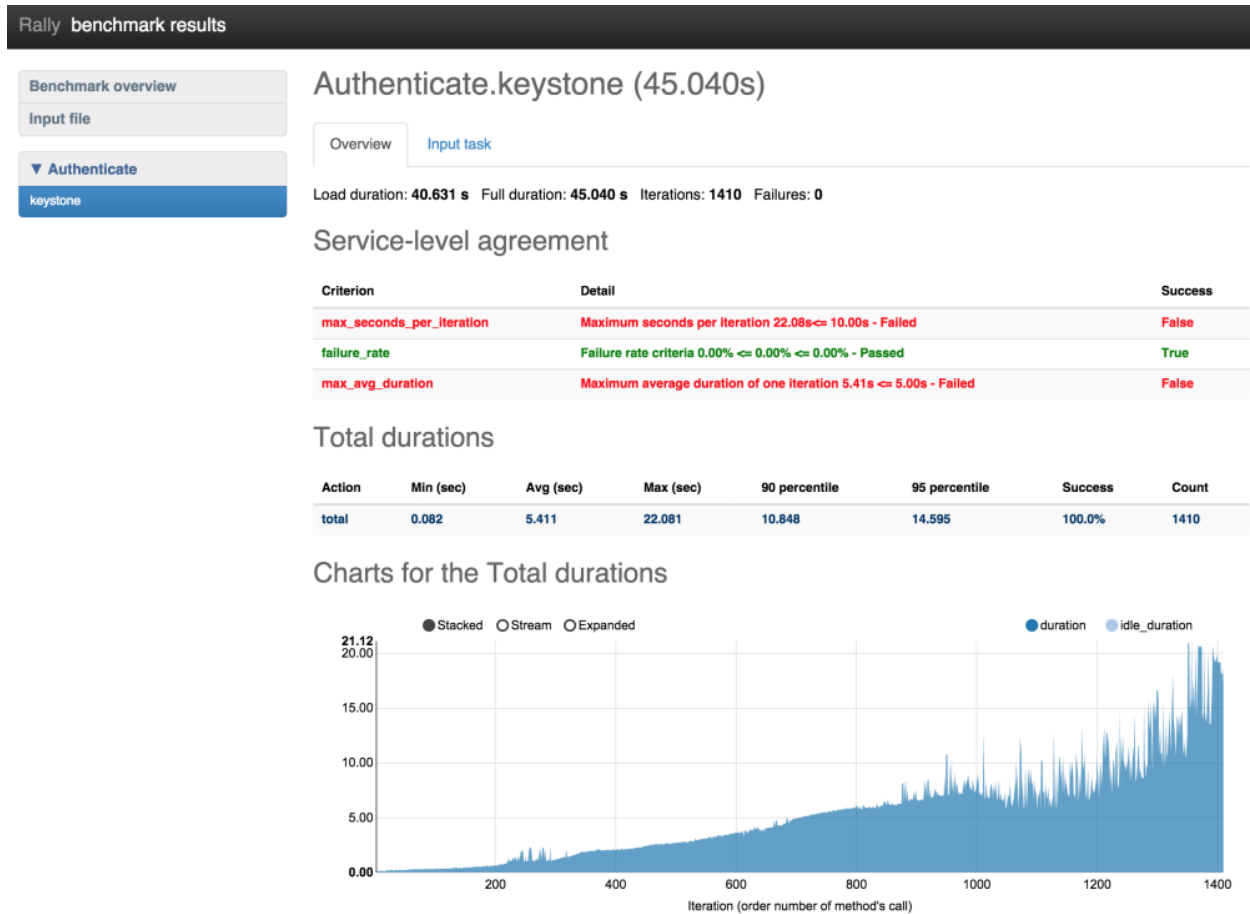
1. maximum average duration of authentication should be less than 5 seconds
2. maximum duration of any authentication should be less than 10 seconds
3. no failed authentication should appear

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	status	active	created_at	name
4251b491-73b2-422a-aecb-695a94165b5e	deploy->finished		2015-01-18 00:11:14.757203	cloud-1

(continues on next page)

(continued from previous page)

```

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      |
+-----+-----+-----+-----+-----+-----+
| 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      |
+-----+-----+-----+-----+-----+-----+
| 4251b491-73b2-422a-aecb-695a94165b5e | 2015-01-05 00:11:14.757203 | cloud-1   |
+-----+-----+-----+-----+-----+-----+
| 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 |
+-----+-----+-----+-----+-----+-----+

```

(continues on next page)

(continued from previous page)

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 |
+-----+
```

(continues on next page)

(continued from previous page)

```

| 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

-----
Create a meter and fetch its statistics.
-----

NAME
    CeilometerStats.create_meter_and_get_stats
PLATFORM
    openstack
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                               |
+-----+-----+-----+-----+-----+

```

(continues on next page)

(continued from previous page)

```
| kwargs | contains optional arguments to create a meter |
+-----+-----+-----+-----+-----+-----+
```

In case if multiple plugins were found, all matched elements are listed:

```
$ rally plugin show NovaKeypair

Multiple plugins found:
+-----+-----+-----+-----+
↪-----+
| Plugin base | Name                                     | Platform | Title |
↪                                     |
+-----+-----+-----+-----+
↪-----+
| Scenario    | NovaKeypair.boot_and_delete_server_with_keypair | openstack | Boot↪
↪and delete server with keypair.          |
| Scenario    | NovaKeypair.create_and_delete_keypair          | openstack | Create↪
↪a keypair with random name and delete keypair. |
| Scenario    | NovaKeypair.create_and_get_keypair              | openstack | Create↪
↪a keypair and get the keypair details.        |
| Scenario    | NovaKeypair.create_and_list_keypairs           | openstack | 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
```

Plugin base	Name	Platform
OSClient	keystone	openstack
Wrapper for KeystoneClient which hides OpenStack auth details.		
Scenario	Authenticate.keystone	openstack
Check Keystone Client.		
Scenario	KeystoneBasic.add_and_remove_user_role	openstack
Create a user role add to a user and disassociate.		
Scenario	KeystoneBasic.authenticate_user_and_validate_token	openstack
Authenticate and validate a keystone token.		
Scenario	KeystoneBasic.create_add_and_list_user_roles	openstack
Create user role, add it and list user roles for given user.		
Scenario	KeystoneBasic.create_and_delete_ec2credential	openstack
Create and delete keystone ec2-credential.		
Scenario	KeystoneBasic.create_and_delete_role	openstack
Create a user role and delete it.		
Scenario	KeystoneBasic.create_and_delete_service	openstack
Create and delete service.		
Scenario	KeystoneBasic.create_and_get_role	openstack
Create a user role and get it detailed information.		

(continues on next page)

(continued from previous page)

Scenario	Code	Environment	Access
Scenario KeystoneBasic.create_and_list_ec2_credentials	↪Create and list all keystone ec2-credentials.	openstack	└
Scenario KeystoneBasic.create_and_list_roles	↪Create a role, then list all roles.	openstack	└
Scenario KeystoneBasic.create_and_list_services	↪Create and list services.	openstack	└
Scenario KeystoneBasic.create_and_list_tenants	↪Create a keystone tenant with random name and list all tenants.	openstack	└
Scenario KeystoneBasic.create_and_list_users	↪Create a keystone user with random name and list all users.	openstack	└
Scenario KeystoneBasic.create_and_update_user	↪Create user and update the user.	openstack	└
Scenario KeystoneBasic.create_delete_user	↪Create a keystone user with random name and then delete it.	openstack	└
Scenario KeystoneBasic.create_tenant	↪Create a keystone tenant with random name.	openstack	└
Scenario KeystoneBasic.create_tenant_with_users	↪Create a keystone tenant and several users belonging to it.	openstack	└
Scenario KeystoneBasic.create_update_and_delete_tenant	↪Create, update and delete tenant.	openstack	└
Scenario KeystoneBasic.create_user	↪Create a keystone user with random name.	openstack	└
Scenario KeystoneBasic.create_user_set_enabled_and_delete	↪Create a keystone user, enable or disable it, and delete it.	openstack	└
Scenario KeystoneBasic.create_user_update_password	↪Create user and update password for that user.	openstack	└
Scenario KeystoneBasic.get_entities	↪instance of a tenant, user, role and service by id's.	openstack	Get└

Step 9. 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 below. 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-aa53-335f67dadcf7f) has been successfully created!
```

(continues on next page)

(continued from previous page)

```
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!
```

(continues on next page)

(continued from previous page)

```

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
```

(continues on next page)

(continued from previous page)

```
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 Ironi
↳available
2017-01-18 16:49:44.404 12162 INFO tempest-verifier [-] {0} tempest.api.baremetal.
↳admin.test_drivers.TestDrivers ... skip: TestDrivers skipped as Ironi
↳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
↳Ironi 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 Ironi
↳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 Ironi 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
↳Ironi 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 Ironi 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-aba2-8fefef1075a87) 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]
```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```
- 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
```

(continues on next page)

(continued from previous page)

```

    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: {'message': 'Cannot add host to aggregate 2658. Reason: One or more hosts
↳ already in availability zone(s) [u'tempest-test_az-34611847'].', '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]

```

(continues on next page)

(continued from previous page)

```

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; this should be a yaml file containing key-value pairs, where the keys are a regex to match against the test name. The values are the reason why the test was skipped. If an invalid regex is supplied the key is treated as a test id. For example:

```

$ 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]:

# Example: Skipping several tests with a partial match:
^tempest\.api\.compute.servers\.test_attach_interfaces: skip using a regex

```

The first five keys are invalid regular expressions and are included in the skip list as is.

```

$ 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).

```

(continues on next page)

(continued from previous page)

```

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).

```

(continues on next page)

(continued from previous page)

```

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]

```

(continues on next page)

(continued from previous page)

```

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]
```

(continues on next page)

(continued from previous page)

```
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](#).

Step 10. Profiling OpenStack Internals

- [Workflow](#)
- [Registering the HMAC key](#)
- [Getting the full trace](#)
- [Disabling the profiler](#)

Rally leverage [OSprofiler](#) to generate traces of OpenStack internal calls happening during the run of a scenario. Integration of OSProfiler in Rally can help to dig into concurrency problems of OpenStack which is a huge ecosystem of cooperative services.

Workflow

Enabling the profiler is based on a shared secret between the clients (here Rally) and the various Openstack services : the HMAC key. In the following we assume that your OpenStack services have been configured to enable OSprofiler and that the secret HMAC key is `SECRET_KEY`. This key is stored alongside the credentials of your deployment. Once Rally is instructed about the HMAC key, a new trace can be initialized for each iteration of the workload. Rally will then store in its reports a profiler trace id. This id can be finally used to query OSprofiler in order to get the full trace of the iteration.

Registering the HMAC key

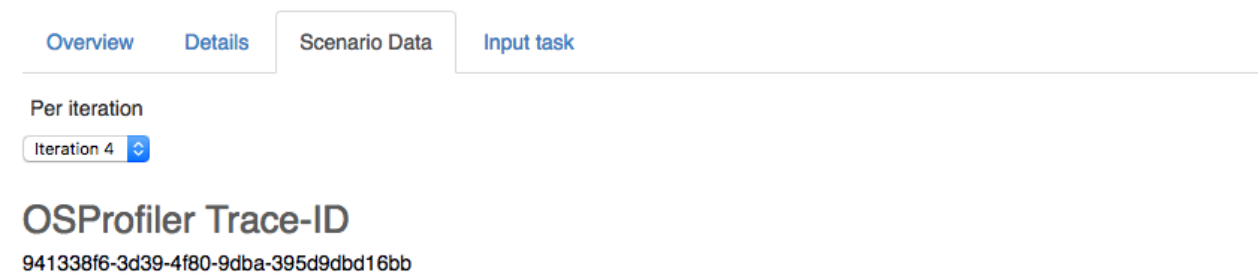
You can store your HMAC key in the environment variable `OSPROFILER_HMAC_KEY`. This variable will be loaded if you create your deployment with the `--from-env` option.

Alternatively if you create your deployment using the `--file` option you can add the HMAC key with the following :

```
{
  "openstack": {
    [...]
    "profiler_hmac_key": "SECRET_KEY"
  }
}
```

Getting the full trace

A trace id is stored on a per-iteration basis and can be found in the JSON report as-well as the HTML report :



Overview Details **Scenario Data** Input task

Per iteration

Iteration 4

OSProfiler Trace-ID

941338f6-3d39-4f80-9dba-395d9dbd16bb

OSProfiler can be asked to generate the full trace using this trace id:

```
osprofiler trace show --html --out trace.html 941338f6-3d39-4f80-9dba-395d9dbd16bb
```

Disabling the profiler

Setting `enable_profiler = False` under the `benchmark` group in the configuration file will disabled the profiler.

1.3.2 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 tasks 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 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.

1.4 Command Line Interface

- *Category: db*
- *Category: deployment*
- *Category: env*
- *Category: plugin*
- *Category: task*
- *Category: verify*

1.4.1 Category: db

CLI commands for DB management.

rally db create

Create Rally database.

rally db ensure

Creates Rally database if it doesn't exist.

rally db recreate

Drop and create Rally database.

This will delete all existing data.

rally db revision

Print current Rally database revision UUID.

rally db show

Show the connection string.

Command arguments:

- `-creds` *[ref]*
Do not hide credentials from connection string

rally db upgrade

Upgrade Rally database to the latest state.

1.4.2 Category: deployment

Set of commands that allow you to manage deployments.

rally deployment check

Check all credentials 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 (see *rally plugin list --plugin-base Engine* for more details).

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

1.4.3 Category: env

Set of commands that allow you to manage envs.

rally env check

Check availability of all platforms in environment.

Command arguments:

- `-env <uuid> [ref]`

UUID or name of the env.

type: str

default: none

- `-json [ref]`

Format output as JSON.

- `-detailed [ref]`

Show detailed information.

rally env cleanup

Perform disaster cleanup for specified environment.

Cases when Rally can leave undeleted resources after performing workload:

- Rally execution was interrupted and cleanup was not performed
- The environment or a particular platform became unreachable which fail Rally execution of cleanup

Command arguments:

- `-json [ref]`
Format output as JSON.
- `-env <uuid> [ref]`
UUID or name of the env.
type: str
default: none

rally env create

Create new environment.

Command arguments:

- `-name <name>, -n <name> [ref]`
Name of the env.
type: str
- `-description <description>, -d <description> [ref]`
Env description
type: str
default: none
- `-extras <extras>, -e <extras> [ref]`
JSON or YAML dict with custom non validate info.
type: str
default: none
- `-from-sysenv [ref]`
Iterate over all available platforms and check system environment for credentials.
- `-spec <path>, -s <path> [ref]`
Path to env spec.
type: str
default: none
- `-json [ref]`
Format output as JSON.
- `-no-use [ref]`

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

rally env delete

Deletes all records related to the environment from db.

Command arguments:

- *-env <uuid> [ref]*
UUID or name of the env.
type: str
default: none
- *-force [ref]*
Delete DB records even if env is not destroyed.

rally env destroy

Destroy existing environment.

Command arguments:

- *-env <uuid> [ref]*
UUID or name of the env.
type: str
default: none
- *-skip-cleanup [ref]*
Do not perform platforms cleanup before destroy.
- *-json [ref]*
Format output as JSON.
- *-detailed [ref]*
Show detailed information.

rally env info

Retrieve and show environment information.

Command arguments:

- *-env <uuid> [ref]*
UUID or name of the env.
type: str
default: none
- *-json [ref]*
Format output as JSON.

rally env list

List existing environments.

Command arguments:

- `-json [ref]`
Format output as JSON.

rally env show

Show base information about the environment record.

Command arguments:

- `-env <uuid> [ref]`
UUID or name of the env.
type: str
default: none
- `-json [ref]`
Format output as JSON.
- `-only-spec [ref]`
Print only a spec for the environment.

rally env use

Set default environment.

Command arguments:

- `-env <uuid> [ref]`
UUID or name of a env.
type: str
- `-json [ref]`
Format output as JSON.

1.4.4 Category: plugin

Set of commands that allow you to manage Rally plugins.

rally plugin list

List all Rally plugins that match name and platform.

Command arguments:

- `-name <name> [ref]`
List only plugins that match the given name.

type: str

default: none

- *-platform <platform>* [*ref*]

List only plugins that are in the specified platform.

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

- *-platform <platform>* [*ref*]

Plugin platform.

type: str

default: none

1.4.5 Category: task

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

rally task abort

Abort a running 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

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.
- `-filter-by <filter_by>` [*ref*]
Filter the displayed workloads.<sla-failures>: only display the failed workloads.
type: str
default: none

rally task export

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

Command arguments:

- `-uuid <uuid>` [*ref*]
UUIDs of tasks or json reports of tasks
type: str
default: none
- `-type <type>` [*ref*]
Report type. Out-of-the-box types: JSON, HTML, HTML-Static, Elastic, JUnit-XML. HINT: You can list all types, executing *rally plugin list -plugin-base TaskExporter* command.
type: str
default: none
- `-to <dest>` [*ref*]
Report destination. Can be a path to a file (in case of JSON, HTML, HTML-Static, JUnit-XML, Elastic etc. types) to save the report to or a connection string. It depends on the report type.

type: str

default: none

- `--deployment <deployment>` [*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*).

Report all tasks with defined deployment

type: str

rally task import

Import json results of a test into rally database

Command arguments:

- `-file <path>` [*ref*]

JSON file with task results

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 a deployment.

type: str

- `-tag <tag>` [*ref*]

Mark the task with a tag or a few tags.

type: str

default: none

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

- `--tag <tag> [ref]`

Tags to filter tasks by.

type: str

default: none

- `--uuids-only [ref]`

List task UUIDs only.

rally task report

Generate a report for the specified task(s).

Command arguments:

- `--out <path> [ref]`

Report destination. Can be a path to a file (in case of HTML, HTML-STATIC, etc. types) to save the report to or a connection string.

type: str

default: none

- `-open` [[ref](#)]
Open the output in a browser.
- `-html` [[ref](#)]
- `-html-static` [[ref](#)]
- `-json` [[ref](#)]
- `-uuid <uuid>` [[ref](#)]
UUIDs of tasks or json reports of tasks
type: str
default: none
- `-deployment <deployment>` [[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](#)).

Report all tasks with defined deployment

type: str

rally task results

DEPRECATED since Rally 3.0.0.

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 start

Run task.

If both `task_args` and `task_args_file` are specified, they are going to be merged. `task_args` has a higher priority so it overrides values from `task_args_file`. There are 3 kinds of return codes, 0: no error, 1: running error, 2: sla check failed.

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>` [\[ref\]](#)

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>` [\[ref\]](#)

Mark the task with a tag or a few tags.

type: str

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 task when any SLA check for it fails for subtask or workload.

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
- `--html-static [ref]`

rally task use

Set active task.

Command arguments:

- `--uuid <uuid> [ref]`
UUID of the task
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]`

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

1.4.6 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

- `-platform <platform>` [[ref](#)]

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

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:

- `--platform <platform>` [[ref](#)]

Required platform (e.g. 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 regexes matching 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

1.5 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).

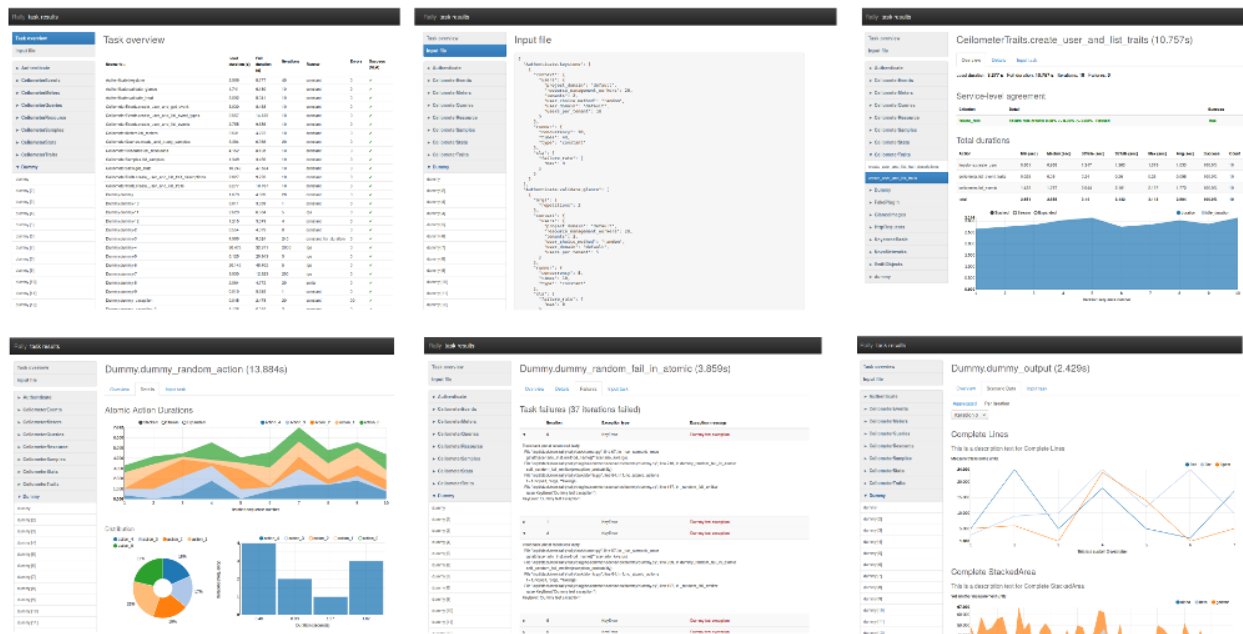
- *HTML Reports*
 - *Task Report*
 - *Trends Report*
- *CLI References*

1.5.1 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-eeed-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-eeed-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-eeed-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.

Rally task results							
Task overview							
Input file							
Task overview							
Scenario ▲	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)	
Authenticate.keystone	3.206	8.079	40	constant	0	✓	
Authenticate.validate_glance	1.376	4.362	10	constant	0	✓	
Authenticate.validate_heat	1.985	5.826	10	constant	0	✓	
CeilometerMeters.list_meters	1.929	3.270	10	constant	0	✓	
CeilometerResource.list_resources	2.099	3.548	10	constant	0	✓	
CeilometerSamples.list_samples	1.401	2.705	10	constant	0	✓	
CeilometerStats.get_stats	9.346	33.979	10	constant	0	✓	
Dummy.dummy	1.021	2.075	20	constant	0	✓	
Dummy.dummy-10	0.010	1.597	1	constant	0	✓	
Dummy.dummy-11	2.598	4.984	5	ros	0	✓	

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.

Rally task results

Task overview

Input file

- ▶ Authenticate
- ▶ CeilometerEvents
- ▶ CeilometerMeters
- ▶ CeilometerQueries
- ▶ CeilometerResource
- ▶ CeilometerSamples
- ▶ CeilometerStats
- ▶ CeilometerTraits
- ▶ Dummy
- ▶ FakePlugin
- ▶ GlanceImages
- ▶ HttpRequests

Input file

```
{
  "Authenticate.keystone": [
    {
      "context": {
        "users": {
          "project_domain": "default",
          "resource_management_workers": 20,
          "tenants": 2,
          "user_choice_method": "random",
          "user_domain": "default",
          "users_per_tenant": 10
        }
      },
      "runner": {
        "concurrency": 20,
        "times": 40,
        "type": "constant"
      },
      "sla": {
        "failure_rate": {
          "max": 0
        }
      }
    }
  ],
  "Authenticate.validate_glance": [
    {
      "args": {
        "repetitions": 2
      }
    }
  ]
}
```

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

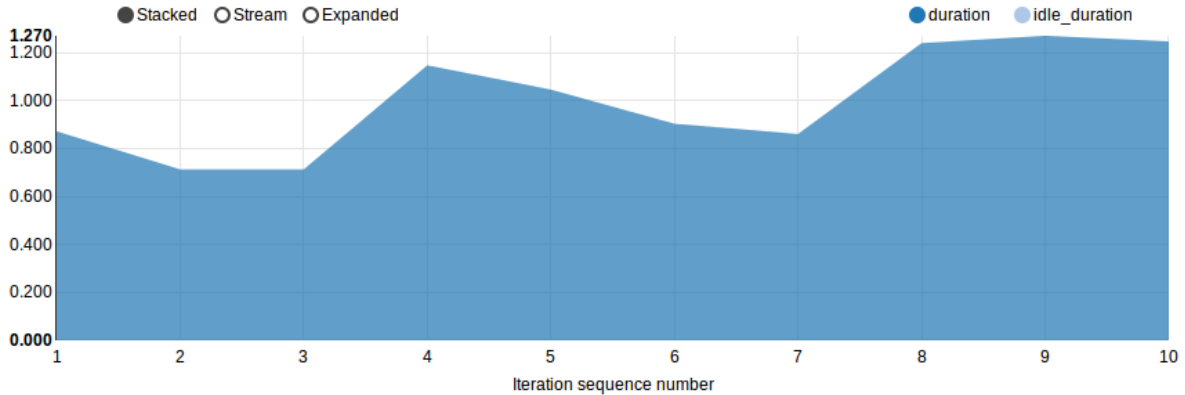


Table with statistics data

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

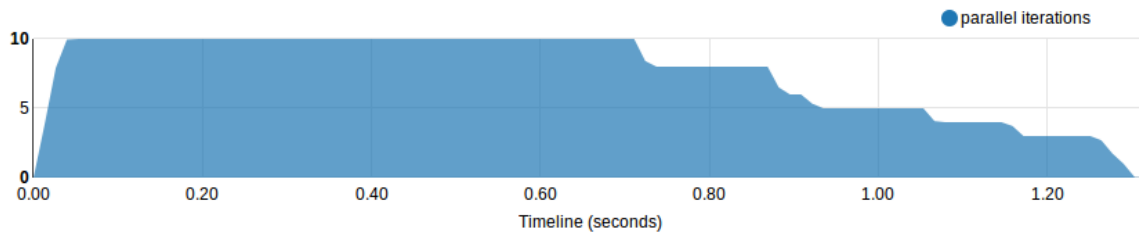
This chart shows *load_duration* and *idle_duration* values per iteration. If there is only one iteration, then chart is useless so it is hidden.

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

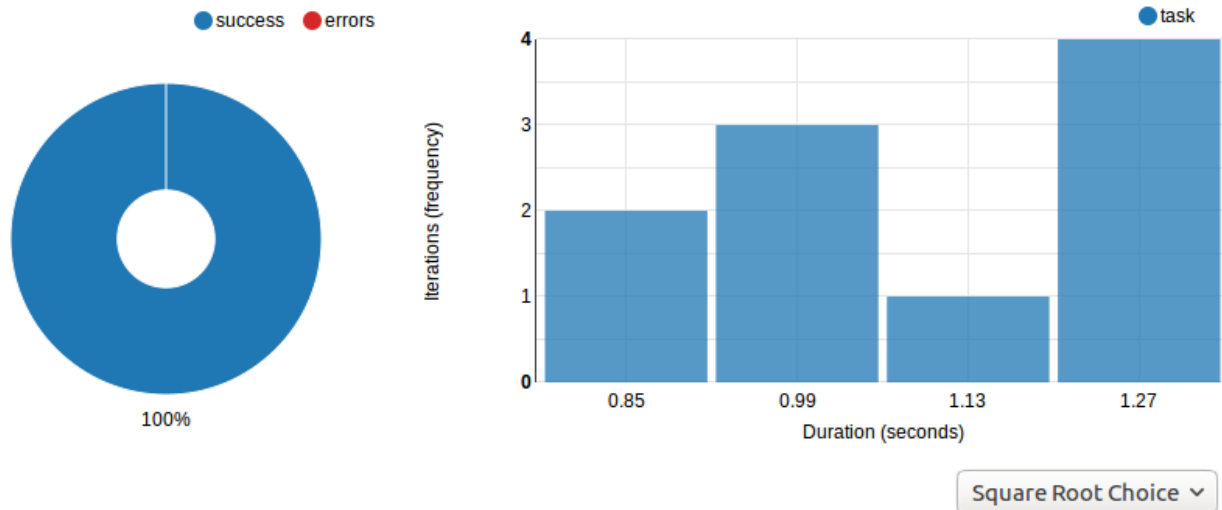
Load profile chart shows number of iterations running in parallel for each workload moment:



Distribution

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

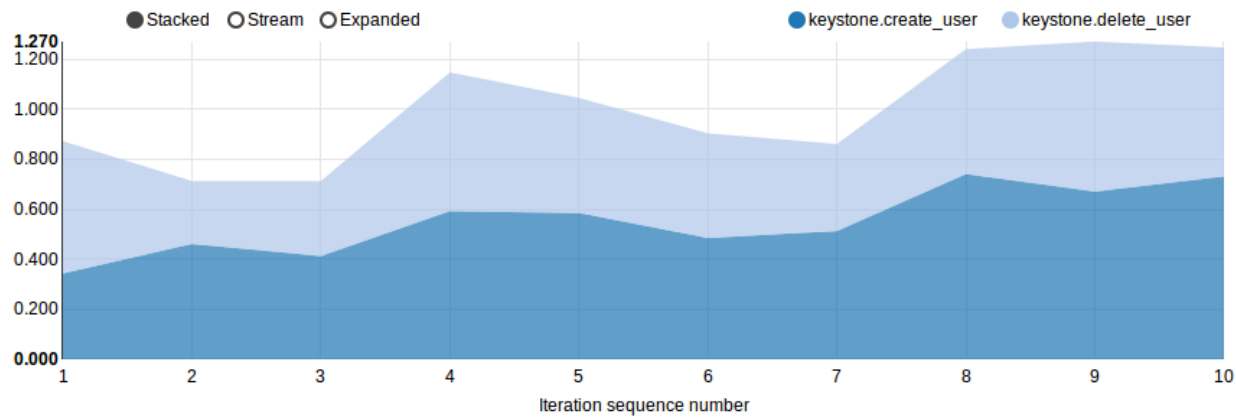
Histogram shows durations distribution with the following *methods* (selected in dropdown list): **Square Root Choice, Sturges Formula, Rise Rule**



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.

[Overview](#) [Scenario Data](#) [Input task](#)

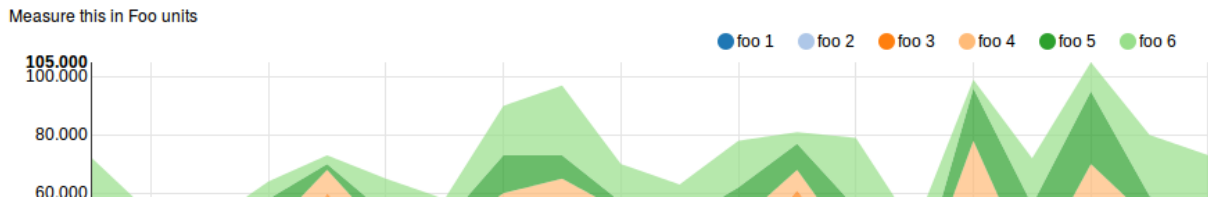
[Aggregated](#) [Per iteration](#)

StatsTable Example

This is a stub description text

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Count
foo stat	1	11	19.3	22.1	24	6.075	20
bar stat	2	17	21	21.1	23	7.7	20
spam stat	2	14	24.1	25	25	7.575	20

StackedArea Example



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.

Task failures (1 iteration failed)

Iteration	Exception type	Exception message
▼ 2	GetResourceErrorStatus	Resource <Snapshot: cc1f8ecb-f246-4d6b-b019-72f66ed39591> has ERROR status. Fault: n/a

Traceback (most recent call last):
File "/opt/stack/new/rally/rally/task/runner.py", line 67, in _run_scenario_once
 getattr(scenario_inst, method_name)(**scenario_kwargs)
File "/opt/stack/new/rally/rally/plugins/openstack/scenarios/nova/servers.py", line 923, in boot_server_from_volume_snapshot
 snapshot = self._create_snapshot(volume.id, False)
File "/opt/stack/new/rally/rally/task/atomic.py", line 84, in func_atomic_actions
 f = func(self, *args, **kwargs)
File "/opt/stack/new/rally/rally/plugins/openstack/scenarios/cinder/utlis.py", line 275, in _create_snapshot
 check_interval=CONF.benchmark.cinder_volume_create_poll_interval
File "/opt/stack/new/rally/rally/common/logging.py", line 236, in wrapper
 return f(*args, **kwargs)
File "/opt/stack/new/rally/rally/task/utlis.py", line 148, in wait_for
 id_attr=id_attr)
File "/opt/stack/new/rally/rally/task/utlis.py", line 214, in wait_for_status
 resource = update_resource(resource)
File "/opt/stack/new/rally/rally/task/utlis.py", line 90, in _get_from_manager
 fault=getattr(res, "fault", "n/a")
GetResourceErrorStatus: Resource <Snapshot: cc1f8ecb-f246-4d6b-b019-72f66ed39591> has ERROR status.
Fault: n/a

Tab «Input Task»

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

Subtask Configuration

```
{
  "KeystoneBasic.create_delete_user": [
    {
      "runner": {
        "type": "constant",
        "concurrency": 10,
        "times": 10
      },
      "sla": {
        "failure_rate": {
          "max": 0
        }
      }
    }
  ]
}
```

Trends Report

If same workload is run several times, some results of these runs can be compared. Compared metrics are success 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-eeed-4696-8e9c-2ba065c68535 a5737eba-a204-43d6-
↪a262-d5ea4b0065da --out trends.html
```

What is an order of workload runs?

Workload run number is 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

Trends overview						
▼ Dummy						
dummy						
dummy_random_fail_in_atomic						

Scenario ▲	Number of runs	Min duration	Max duration	Avg duration	SLA
Dummy.dummy	1	-	-	-	✓
Dummy.dummy_random_fail_in_atomic	10	1.4270	8.0660	4.5303	✓

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.

Total	Configuration
-------	---------------

This workload has single run so trends can not be displayed.
There should be at least two workload results with the same configuration

Tab «Total»

Total durations

Shows workload *load_duration* statistics trends.

Total success rate

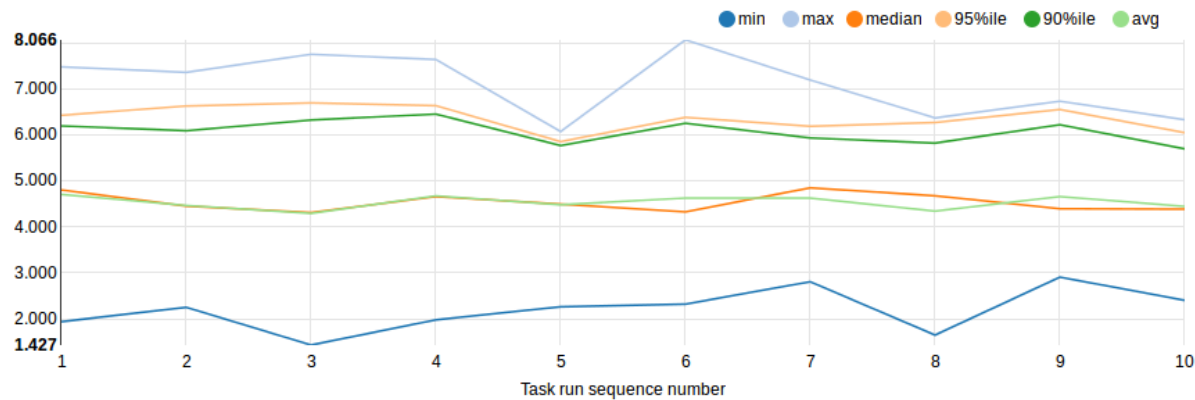
Shows trends for percent of successful iterations

Total

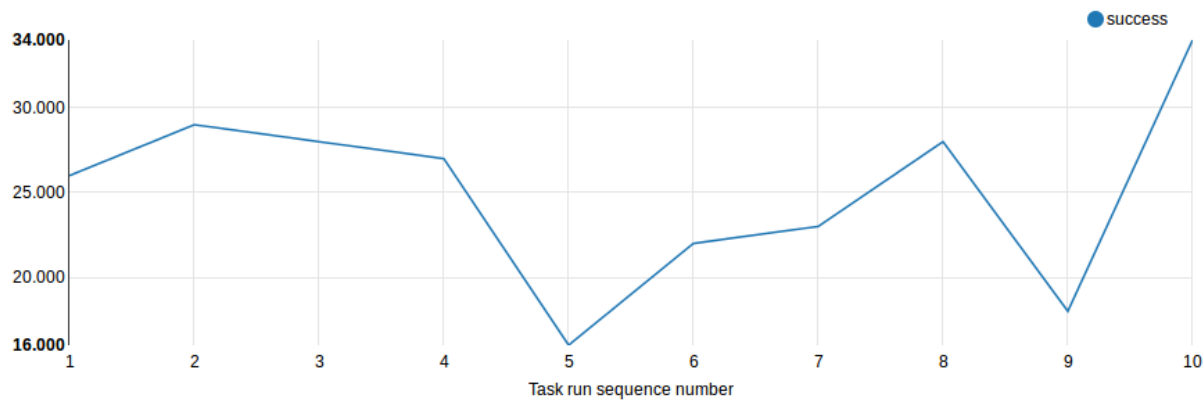
Atomic actions

Configuration

Total durations



Total success rate



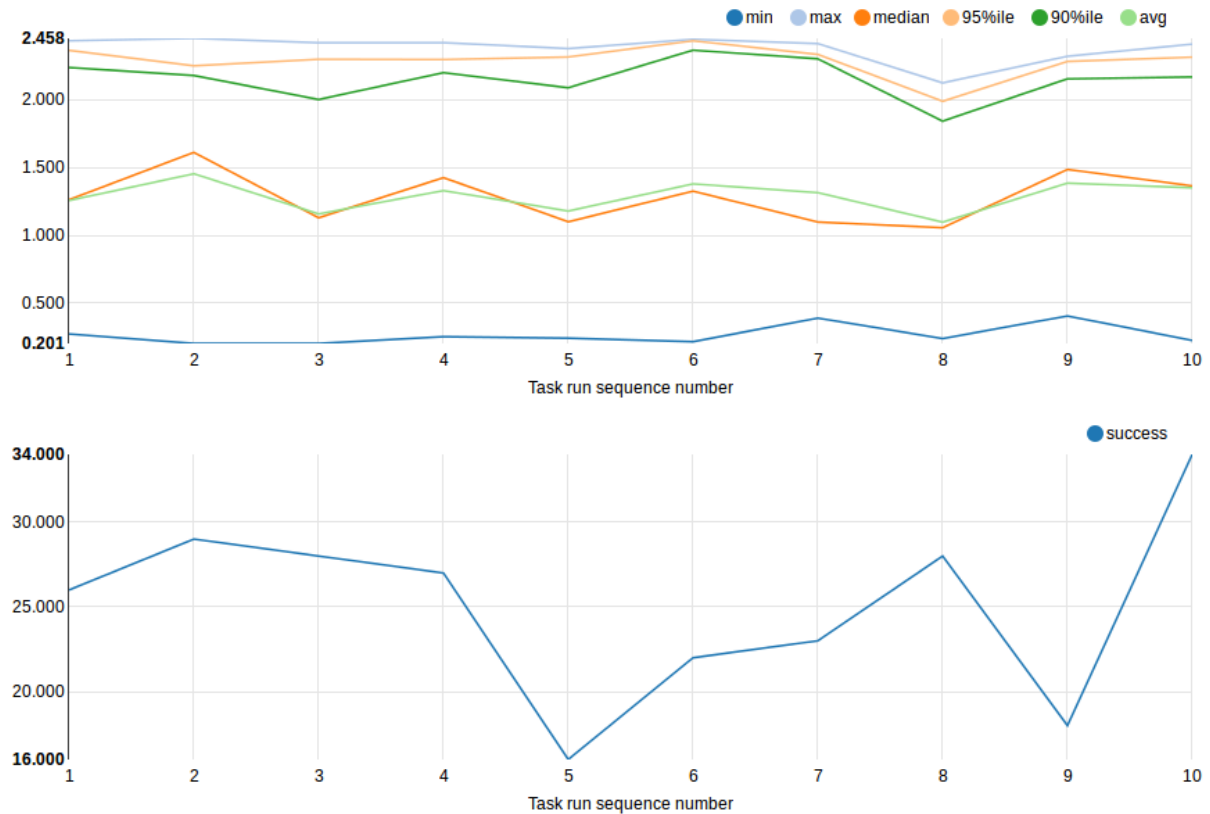
Tab «Atomic actions»

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

Total Atomic actions Configuration

Atomic actions durations / success rate

dummy_fail_test



Tab «Configuration»

Here is a configuration JSON for current workload.

Total Atomic actions Configuration

Workload configuration

```
{
  "runner": {
    "type": "constant",
    "concurrency": 10,
    "times": 100
  },
  "args": {
    "exception_probability": 0.5
  }
}
```

1.5.2 CLI References

For more information regarding Rally Task Component CLI please proceed to [CLI reference](#)

1.6 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*).

1.6.1 Verifiers

- *What is it?*
- *Verifier statuses*
- *Verification statuses*
- *Known verifier types*

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 be 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:

Third-party

Nothing here yet.

1.6.2 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*
 - *Filtering results*
 - *Tests Tags*
 - *Tracebacks & Reasons*
- *Plugins Reference for all out-of-the-box reporters*
 - *html*
 - *html-static*
 - *json*

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-1da4b3afeafe	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:						<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<div> <div>Toggle Header</div> <div>Toggle Tags</div> <div>Toggle All Filters</div> </div>										
Test name (shown 9)										
						86a70461-54e0-4032-830b-1da4b3afeafe	75c45caf-7ae2-4f99-ae28-77c8eafe241f	149f0dc9-6772-45be-80ef-02a52428063c		
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestUSON.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.AggregatesAdminTestUSON.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.AggregatesAdminTestUSON.test_aggregate_add_host_list						success 0.921	success 0.946 (+0.025)	success 0.967 (+0.046)		
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestUSON.test_aggregate_add_remove_host						success 0.727	success 0.823 (+0.096)	success 1.171 (+0.444)		
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestUSON.test_aggregate_create_delete						success 0.646	fail 0.709 (+0.063)	fail 0.851 (+0.205)		
tempest.api.compute.admin.test_aggregates.AggregatesAdminTestUSON.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.AggregatesAdminTestUSON.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.AggregatesAdminTestUSON.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.AggregatesAdminTestUSON.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 assertEquals 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 assertEquals 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/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/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/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/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/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/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: {'message': '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.

Platform: default

Module: `rally.plugins.verification.reporters`

html-static

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

Platform: default

Module: `rally.plugins.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
    }
  },
}
```

(continues on next page)

(continued from previous page)

```

    "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",

```

(continues on next page)

(continued from previous page)

```

        "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"
        }
    }
}
}

```

(continues on next page)

(continued from previous page)

```

    }
}

```

Platform: default**Module:** rally.plugins.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"

```

(continues on next page)

(continued from previous page)

```

        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>HEEEEEELP</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>

```

Platform: default**Module:** rally.plugins.verification.reporters

1.6.3 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*
- *rally verify rerun*
- *rally verify show*
- *rally verify show-verifier*
- *rally verify start*
- *rally verify update-verifier*
- *rally verify use*
- *rally verify use-verifier*

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

- `-platform <platform>` [[ref](#)]

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

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:

- `-platform <platform>` [*ref*]

Required platform (e.g. 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 regexes matching 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

1.6.4 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](#)
- [Example of custom JSON Reporter](#)

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,  
                                                    out-  
                                                    put_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](#)
- [Example of Fake Verifier Manager](#)

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 in-
                stallation
            • 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.

    Every extension is a dict object which contains name and entry_point keys. example:
```

```
{ "name": p.name, "entry_point": p.entry_point_target
}
```

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.FatalilityTestCase.test_one",
             "fake_tool.tests.bar.FatalilityTestCase.test_two",
             "fake_tool.tests.bar.FatalilityTestCase.test_three",
             "fake_tool.tests.bar.FatalilityTestCase.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
    # cloning 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 = {}
```

(continues on next page)

(continued from previous page)

```

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"] = "Ooooppss"
        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*
- *Verification statuses*
- *Command Line Interface*
 - *Installing verifier*
 - *Re-install verifier aka update*
 - *Uninstall*
 - *Installation extensions*
 - *Uninstall extensions*
 - *List extensions*
 - *Discover available tests*
 - *Configuring*

- *Show config*
- *Running verification*
- *Show verification result*
- *Listing all verifications*
- *Importing results*
- *Building reports*
- *The End*

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 <code>rally verify start</code> 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!

1.6.5 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).

1.6.6 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.

1.7 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.

1.7.1 Plugins Reference

- *Common*
 - *Platforms*
- *Task Component*
 - *Charts*
 - *Contexts*
 - *Hook Actions*
 - *Hook Triggers*
 - *SLAs*
 - *Scenarios*
 - *Scenario Runners*
 - *Task Exporters*
 - *Validators*
- *Verification Component*
 - *Verification Reporters*
 - *Verifier Contexts*
 - *Verifier Managers*

Common

Platforms

Task Component

Charts

EmbeddedChart [Chart]

Chart for embedding custom html as a complete chart.

Example of usage:

```
self.add_output (
    complete={
        "title": "Embedding link to example.com",
        "chart_plugin": "EmbeddedChart",
        "data": "<a href='example.com'>"
                "To see external logs follow this link"
                "</a>"
    }
)
```

Platform: default

Module: `rally.task.processing.charts`

EmbeddedExternalChart [Chart]

Chart for embedding external html page as a complete chart.

Example of usage:

```
self.add_output(  
    complete={  
        "title": "Embedding external html page",  
        "chart_plugin": "EmbeddedExternalChart",  
        "data": "https://example.com"  
    }  
)
```

Platform: default

Module: `rally.task.processing.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"  
    })
```

Platform: 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]]})
```

Platform: 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"})
```

Platform: 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]]})
```

Platform: 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}]}})
```

Platform: 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"]})
```

Platform: default

Module: `rally.task.processing.charts`

Contexts

dummy_context [Context]

Dummy context.

Platform: default

Parameters:

- *fail_setup* (bool) [ref]
- *fail_cleanup* (bool) [ref]

Module: `rally.plugins.task.contexts.dummy`

Hook Actions

sys_call [Hook Action]

Performs system call.

Platform: default

Parameters:

- *str* [ref]
Command to execute.

Module: rally.plugins.task.hooks.sys_call

Hook Triggers

event [Hook Trigger]

Triggers hook on specified event and list of values.

Platform: 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.

- *unit* [ref]
Set of expected values: 'time'.
- *at (list)* [ref]
Elements of the list should follow format(s) described below:
 - Type: int. Format:

```
{
  "type": "integer",
  "minimum": 0
}
```

Option 2 of parameters:

Triage hook based on specific iterations.

- *unit* [ref]
Set of expected values: 'iteration'.
- *at (list)* [ref]
Elements of the list should follow format(s) described below:
 - Type: int. Format:

```
{  
  "type": "integer",  
  "minimum": 1  
}
```

Module: `rally.plugins.task.hook_triggers.event`

periodic [Hook Trigger]

Periodically triggers hook with specified range and step.

Platform: 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.

- *unit* [ref]
Set of expected values: 'time'.
- *start (int)* [ref]
Min value: 0.
- *end (int)* [ref]
Min value: 1.
- *step (int)* [ref]
Min value: 1.

Option 2 of parameters:

Periodically triage hook based on iterations.

- *unit* [ref]
Set of expected values: 'iteration'.
- *start (int)* [ref]
Min value: 1.
- *end (int)* [ref]
Min value: 1.
- *step (int)* [ref]
Min value: 1.

Module: `rally.plugins.task.hook_triggers.periodic`

SLAs

failure_rate [SLA]

Failure rate minimum and maximum in percents.

Platform: default

Parameters:

- *min (float) [ref]*
Min value: 0.0.
Max value: 100.0.
- *max (float) [ref]*
Min value: 0.0.
Max value: 100.0.

Module: `rally.plugins.task.sla.failure_rate`

max_avg_duration [SLA]

Maximum average duration of one iteration in seconds.

Platform: default

Parameters:

- *float [ref]*

Module: `rally.plugins.task.sla.max_average_duration`

max_avg_duration_per_atomic [SLA]

Maximum average duration of one iterations atomic actions in seconds.

Platform: default

Parameters:

Dictionary is expected. Keys should follow pattern(s) described bellow.

- *. (str)* [ref]*
The name of atomic action.

Module: `rally.plugins.task.sla.max_average_duration_per_atomic`

max_seconds_per_iteration [SLA]

Maximum time for one iteration in seconds.

Platform: default

Parameters:

- *float* [*ref*]

Min value: 0.0.

Module: `rally.plugins.task.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.

Platform: 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.task.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.

Platform: default

Parameters:

- *max_degradation (float)* [*ref*]

Min value: 0.0.

Module: `rally.plugins.task.sla.performance_degradation`

Scenarios

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.

Platform: default

Parameters:

- *sleep* [ref]

Idle time of method (in seconds).

Module: `rally.plugins.task.scenarios.dummy.dummy`

Dummy.dummy_exception [Scenario]

Throws an exception.

Dummy.dummy_exception used for testing if exceptions are processed properly by task engine and analyze rally results storing & displaying capabilities.

Platform: 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.task.scenarios.dummy.dummy`

Dummy.dummy_exception_probability [Scenario]

Throws an exception with given probability.

Dummy.dummy_exception_probability used for testing if exceptions are processed properly by task engine and analyze rally results storing & displaying capabilities.

Platform: 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.task.scenarios.dummy.dummy`

Dummy.dummy_output [Scenario]

Generate dummy output.

This scenario generates example of output data.

Platform: default

Parameters:

- *random_range* [ref]
Max int limit for generated random values

Module: `rally.plugins.task.scenarios.dummy.dummy`

Dummy.dummy_random_action [Scenario]

Sleep random time in dummy actions.

Platform: 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.task.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.

Platform: default

Parameters:

- *exception_probability* [ref]
Probability with which atomic actions fail in this dummy scenario ($0 \leq p \leq 1$)

Module: `rally.plugins.task.scenarios.dummy.dummy`

Dummy.dummy_timed_atomic_actions [Scenario]

Run some sleepy atomic actions for SLA atomic action tests.

Platform: 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.task.scenarios.dummy.dummy`

Dummy.failure [Scenario]

Raise errors in some iterations.

Platform: 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.task.scenarios.dummy.dummy`

HttpRequests.check_random_request [Scenario]

Executes random HTTP requests from provided list.

This scenario takes random url from list of requests, and raises exception if the response is not the expected response.

Platform: 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.task.scenarios.requests.http_requests`

HttpRequests.check_request [Scenario]

Standard way for testing web services using HTTP requests.

This scenario is used to make request and check it with expected Response.

Platform: 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.task.scenarios.requests.http_requests`

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.

Platform: default

Parameters:

- *concurrency (int) [ref]*
The number of parallel iteration executions.
Min value: 1.
- *times (int) [ref]*
Total number of iteration executions.
Min value: 1.
- *timeout (float) [ref]*
Operation's timeout.
- *max_cpu_count (int) [ref]*
The maximum number of processes to create load from.
Min value: 1.

Module: [rally.plugins.task.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 scenario in order to simulate the activities of multiple users placing load on the cloud under test.

Platform: default

Parameters:

- *concurrency (int) [ref]*
The number of parallel iteration executions.
Min value: 1.
- *duration (float) [ref]*
The number of seconds during which to generate a load. If the duration is 0, the scenario will run once per parallel execution.

Min value: 0.0.

- *timeout (float)* [ref]

Operation's timeout.

Min value: 1.

Module: `rally.plugins.task.runners.constant`

rps [Scenario Runner]

Scenario runner that does the job with specified frequency.

Every single 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.

Platform: default

Parameters:

- *times (int)* [ref]

Min value: 1.

- *rps* [ref]

- *timeout (float)* [ref]

- *max_concurrency (int)* [ref]

Min value: 1.

- *max_cpu_count (int)* [ref]

Min value: 1.

Module: `rally.plugins.task.runners.rps`

serial [Scenario Runner]

Scenario runner that executes 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 execute scenario without introducing any concurrent operations as well as interactively debug the scenario from the same command that you use to start Rally.

Platform: default

Parameters:

- *times (int)* [ref]

Min value: 1.

Module: `rally.plugins.task.runners.serial`

Task Exporters

elastic [Task Exporter]

Exports task results to the Elasticsearch 2.x, 5.x or 6.x clusters.

The exported data includes:

- Task basic information such as title, description, status, deployment uuid, etc. See `rally_task_v1_data` index.
- Workload information such as scenario name and configuration, runner type and configuration, time of the start load, success rate, sla details in case of errors, etc. See `rally_workload_v1_data` index.
- Separate documents for all atomic actions. See `rally_atomic_action_data_v1` index.

The destination can be a remote server. In this case specify it like:

```
https://elastic:changeme@example.com
```

Or we can dump documents to the file. The destination should look like:

```
/home/foo/bar.txt
```

In case of an empty destination, the `http://localhost:9200` destination will be used.

Platform: default

Module: `rally.plugins.task.exporters.elastic.exporter`

html [Task Exporter]

Generates task report in HTML format.

Platform: default

Module: `rally.plugins.task.exporters.html`

html-static [Task Exporter]

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

Platform: default

Module: `rally.plugins.task.exporters.html`

json [Task Exporter]

Generates task report in JSON format.

Platform: default

Module: `rally.plugins.task.exporters.json_exporter`

junit-xml [Task Exporter]

Generates task 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 0.10.0 at 2017-06-04T05:14:00-->
  <testsuite id="task-uu-ii-dd"
    errors="0"
    failures="1"
    skipped="0"
    tests="2"
    time="75.0"
    timestamp="2017-06-04T05:14:00">
    <testcase classname="CinderVolumes"
      name="list_volumes"
      id="workload-1-uuid"
      time="29.9695231915"
      timestamp="2017-06-04T05:14:44" />
    <testcase classname="NovaServers"
      name="list_keypairs"
      id="workload-2-uuid"
      time="5"
      timestamp="2017-06-04T05:15:15">
      <failure>oops</failure>
    </testcase>
  </testsuite>
</testsuites>

```

Platform: default

Module: rally.plugins.task.exporters.junit

old-json-results [Task Exporter]

Generates task report in JSON format as old *rally task results*.

Platform: default

Module: rally.plugins.task.exporters.old_json_results

trends-html [Task Exporter]

Generates task trends report in HTML format.

Platform: default

Module: rally.plugins.task.exporters.trends

trends-html-static [Task Exporter]

Generates task trends report in HTML format with embedded JS/CSS.

Platform: default

Module: `rally.plugins.task.exporters.trends`

Validators

args-spec [Validator]

Scenario arguments validator

Platform: default

Module: `rally.plugins.common.validators`

check_constant [Validator]

Additional schema validation for constant runner

Platform: default

Module: `rally.plugins.task.runners.constant`

check_rps [Validator]

Additional schema validation for rps runner

Platform: default

Module: `rally.plugins.task.runners.rps`

enum [Validator]

Checks that parameter is in a list.

Ensure a parameter has the right value. This value need to be defined in a list.

Platform: default

Parameters:

- *param_name* [ref]
Name of parameter to validate
- *values* [ref]
List of values accepted
- *missed* [ref]
Allow to accept optional parameter
- *case_insensitive* [ref]
Ignore case in enum values

Module: `rally.plugins.common.validators`

es_exporter_destination [Validator]

Validates the destination for ElasticSearch exporter.

In case when the destination is ElasticSearch cluster, the version of it should be 2.* or 5.*

Platform: default

Module: `rally.plugins.task.exporters.elastic.exporter`

file_exists [Validator]

Validator checks parameter is proper path to file with proper mode.

Ensure a file exists and can be accessed with the specified mode. Note that path to file will be expanded before access checking.

Platform: default

Parameters:

- *param_name* [ref]
Name of parameter to validate
- *mode* [ref]
Access mode to test for. This should be one of: * os.F_OK (file exists) * os.R_OK (file is readable) * os.W_OK (file is writable) * os.X_OK (file is executable)
If multiple modes are required they can be added, eg: mode=os.R_OK+os.W_OK
- *required* [ref]
Boolean indicating whether this argument is required.

Module: `rally.plugins.common.validators`

jsonschema [Validator]

JSON schema validator

Platform: default

Module: `rally.plugins.common.validators`

map_keys [Validator]

Check that parameter contains specified keys.

Platform: default

Parameters:

- *param_name* [ref]
Name of parameter to validate
- *required* [ref]
List of all required keys

- *allowed* [[ref](#)]
List of all allowed keys
- *additional* [[ref](#)]
Whether additional keys are allowed. If list of allowed keys are specified, defaults to False, otherwise defaults to True
- *missed* [[ref](#)]
Allow to accept optional parameter

Module: `rally.plugins.common.validators`

number [Validator]

Checks that parameter is a number that pass specified condition.

Ensure a parameter is within the range [minval, maxval]. This is a closed interval so the end points are included.

Platform: default

Parameters:

- *param_name* [[ref](#)]
Name of parameter to validate
- *minval* [[ref](#)]
Lower endpoint of valid interval
- *maxval* [[ref](#)]
Upper endpoint of valid interval
- *nullable* [[ref](#)]
Allow parameter not specified, or parameter=None
- *integer_only* [[ref](#)]
Only accept integers

Module: `rally.plugins.common.validators`

required_contexts [Validator]

Validator checks if required contexts are specified.

Platform: default

Parameters:

- *contexts* [[ref](#)]
List of strings and tuples with context names that should be specified. Tuple represent 'at least one of the'.

Module: `rally.plugins.common.validators`

required_param_or_context [Validator]

Validator checks if required image is specified.

Platform: default

Parameters:

- *param_name* [ref]
Name of parameter
- *ctx_name* [ref]
Name of context

Module: `rally.plugins.common.validators`

required_params [Validator]

Scenario required parameter validator.

This allows us to search required parameters in subdict of config.

Platform: default

Parameters:

- *subdict* [ref]
Sub-dict of “config” to search. if not defined - will search in “config”
- *params* [ref]
List of required parameters

Module: `rally.plugins.common.validators`

required_platform [Validator]

Validates specification of specified platform for the workload.

Platform: default

Parameters:

- *platform* [ref]
Name of the platform

Module: `rally.common.validation`

restricted_parameters [Validator]

Validates that parameters is not set.

Platform: default

Parameters:

- *param_names* [ref]
Parameter or parameters list to be validated.

- *subdict* [*ref*]

Sub-dict of “config” to search for param_names. if not defined - will search in “config”

Module: `rally.plugins.common.validators`

Verification Component

Verification Reporters

html [Verification Reporter]

Generates verification report in HTML format.

Platform: default

Module: `rally.plugins.verification.reporters`

html-static [Verification Reporter]

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

Platform: default

Module: `rally.plugins.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",
```

(continues on next page)

(continued from previous page)

```

    "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"
      }
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

        "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"
        }
    }
}
}
}
}

```

Platform: default

Module: rally.plugins.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"
```

(continues on next page)

(continued from previous page)

```
        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>HEEEEEEEELP</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>
```

Platform: default

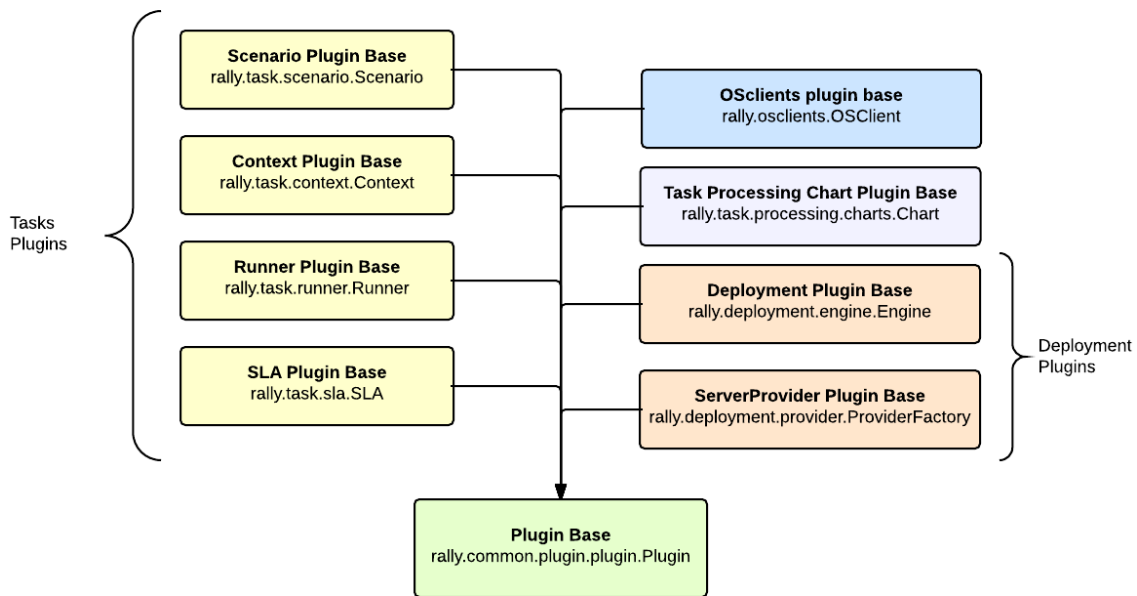
Module: rally.plugins.verification.reporters

Verifier Contexts

Verifier Managers

1.7.2 How plugins work

Rally provides an opportunity to create and use a **custom task scenario, runner, SLA, deployment or context** as a **plugin**:



1.7.3 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.

1.7.4 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 runner start first iteration and deletes it after runner finishes execution of all iterations.

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.plugins.openstack import osclients

LOG = logging.getLogger(__name__)

@context.configure(name="create_flavor", order=1000)
class CreateFlavorContext(context.Context):
    """This sample creates a flavor with specified option."""

    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 then 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()
```

(continues on next page)

(continued from previous page)

```

        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

The new plugin can be used by specifying it in context section. Like below:

```

{
  "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 configuration:

```
{
  "Dummy.dummy": [
    {
      "args": {
        "sleep": 0.01
      },
      "runner": {
        "type": "constant",
        "times": 1500,
        "concurrency": 1
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    "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

```

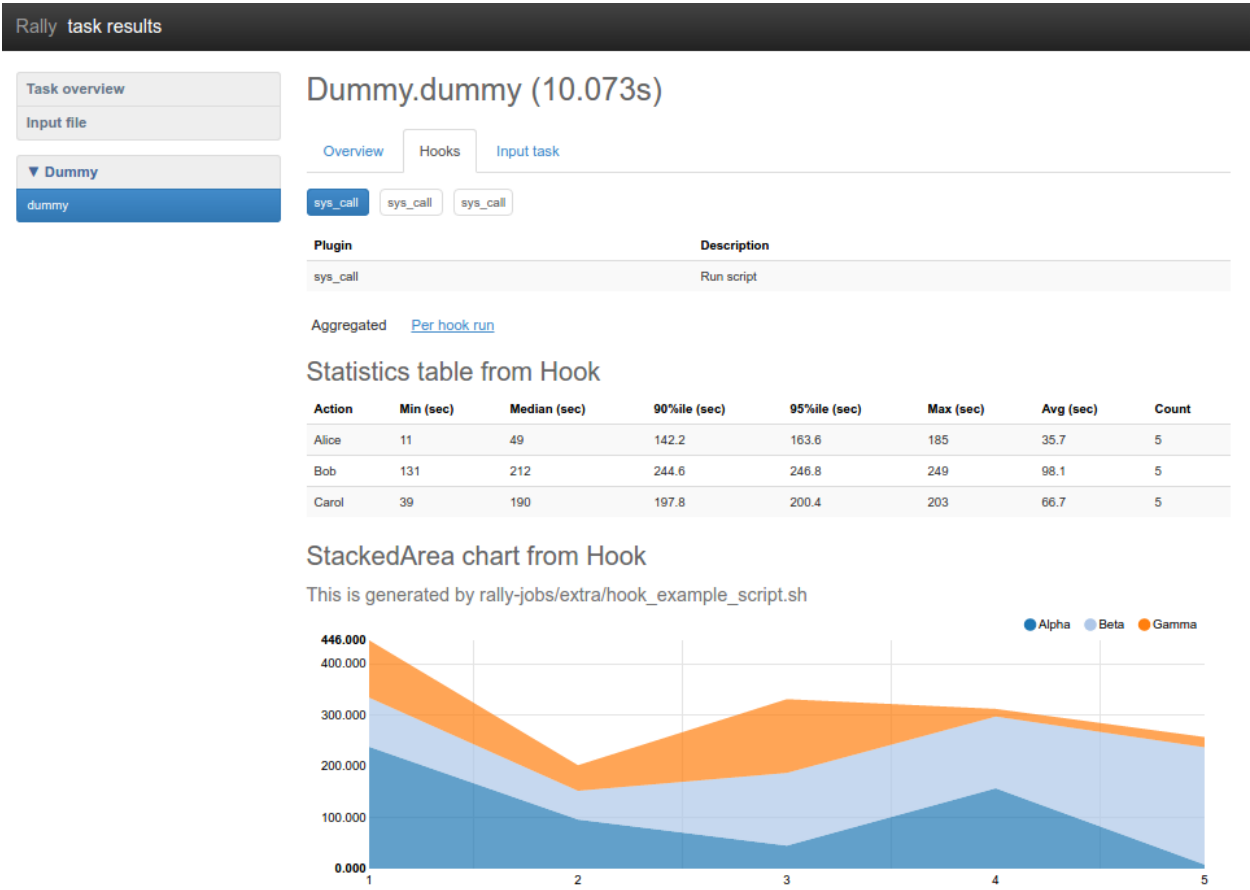
(continues on next page)

(continued from previous page)

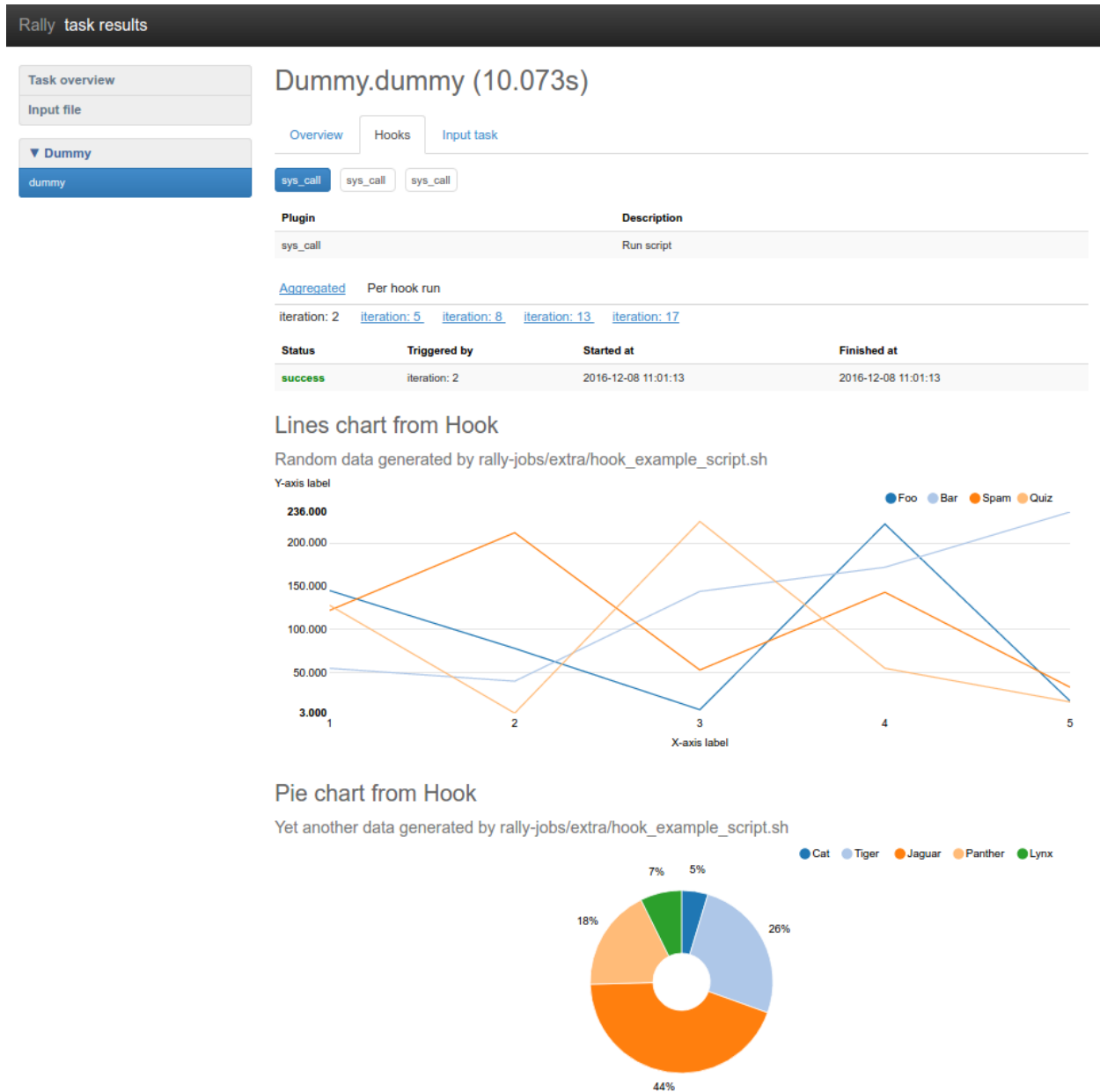
```
args: uname -a
trigger:
  name: event
  args:
    unit: iteration
    at: [2, 3, 4, 5, 6, 8, 10, 12, 13, 15, 17, 18]
sla:
  failure_rate:
    max: 0
```

hook_example_script.sh generates dummy output in JSON format. Graphical 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:

Rally task results

Task overview

Input file

▼ Dummy

dummy

Dummy.dummy (1.620s)

Overview Hooks Input task

sys_call

Plugin	Description
sys_call	Hook demo

Per hook run

iteration: 3 [iteration: 6](#)

Status	Triggered by	Started at	Finished at
success	iteration: 3	2016-12-13 11:54:04	2016-12-13 11:54:04

System call

Args: /bin/echo foo
 RetCode: 0
 StdOut: foo
 StdErr: (empty)

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,
            },
        ],
    }
```

(continues on next page)

(continued from previous page)

```

    ]
}

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 runner plugin that runs a given 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 (between min_times and max_times)
    """

    CONFIG_SCHEMA = {
        "type": "object",
        "$schema": consts.JSON_SCHEMA,
        "properties": {
            "type": {

```

(continues on next page)

(continued from previous page)

```

        "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 input task 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 *OpenStackScenario* 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 import consts
from rally.plugins.openstack import scenario
from rally.task import atomic
from rally.task import validation

@validation.add("required_services", services=[consts.Service.NOVA])
@validation.add("required_platform", platform="openstack", users=True)
@scenario.configure(name="ScenarioPlugin.list_flavors_useless")
class ListFlavors(scenario.OpenStackScenario):
    """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()

    def run(self):
        """List flavors."""
        self._list_flavors()
        self._list_flavors_as_admin()
```

Usage

You can refer to your plugin scenario in the task input 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

@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

The new plugin can be used by specifying it in SLA section. Like below:

```
{
    "Dummy.dummy": [
        {
            "args": {
                "sleep": 0.01
            },
            "runner": {
```

(continues on next page)

(continued from previous page)

```

        "type": "constant",
        "times": 5,
        "concurrency": 1
    },
    "context": {
        "users": {
            "tenants": 1,
            "users_per_tenant": 1
        }
    },
    "sla": {
        "max_duration_range": 2.5
    }
}
]
}

```

1.8 Contribute to Rally

1.8.1 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 **scenarios**, **contexts** etc.). To do so, please read this article.

1.8.2 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>)

1.8.3 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 through 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](#).

1.9 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:

1.9.1 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.

1.9.2 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 tasks.

Problem description

Heavy usage of queue isn't checked.

Possible solution

- Before running task start process which will connect to queue topics and measure messages count, size and other data which we need.

1.9.3 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

1.9.4 Distributed load generation

Use Case

Some OpenStack projects (Marconi, Magnetodb) require a real huge load, like 10-100k request per second for load testing.

To generate such huge load Rally has 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

1.9.5 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

1.9.6 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

1.9.7 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

1.9.8 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.

1.9.9 Launch Specific SubTask

Use case

A developer is working on a feature that is covered by one or more specific subtask. He/she would like to execute a rally task with an existing task template file (YAML or JSON) indicating exactly what subtask will be executed.

Problem description

When executing a task with a template file in Rally, all subtasks are executed without the ability to specify one or a set of subtasks the user would like to execute.

Possible solution

- Add optional flag to rally task start command to specify one or more subtasks to execute as part of that test run.

1.9.10 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 subtask. 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 subtask scenarios that produces more real life load will produce mess and a lot of duplication of code.

Possible solution

- Extend Rally subtask configuration in such way to support passing multiple scenarios in single subtask context
- Extend Rally task output format to support results of multiple scenarios in single subtask separately.
- Extend rally task plot2html and rally task detailed to show results separately for every scenario.

1.9.11 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”

1.9.12 Add support of persistence task environment

Use Case

There are situations when same environment is used across different tasks. For example you would like to improve operation of listing objects. For example:

- Create hundreds of objects
- Collect baseline of list performance
- Fix something in system
- Repeat the performance test
- Repeat fixing and testing until things are fixed.

Current implementation of Rally will force you to recreate task context which is time consuming operation.

Problem Description

Fortunately Rally has already a mechanism for creating task environment via contexts. Unfortunately it's atomic operation: - Create task context - Perform subtask scenario-runner pairs - Destroy task context

This should be split to 3 separated steps.

Possible solution

- Add new CLI operations to work with task environment: (show, create, delete, list)
- Allow task to start against existing task context (instead of deployment)

1.9.13 Production read cleanups

Use Case

Rally should delete in all cases all resources that it creates during tasks.

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.

1.10 Project Info and Release Notes

1.10.1 Maintainers

Project Team Lead (PTL)

Contact	Area of interest
Andrey Kurilin andreykurilin (irc) andreykurilin (gitter) andr.kurilin@gmail.com	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none"> • Rally reports • Front-end
Anton Studenov tohin (irc) astudenov@mirantis.com	<ul style="list-style-type: none"> • Rally Deployment • Task Hooks
Boris Pavlovic boris-42 (irc) boris@pavlovic.me	<ul style="list-style-type: none"> • Founder and ideological leader • Architect • Rally task & plugins
Chen Haibing chenhb-zte (gitter) chen.haibing1@zte.com.cn	<ul style="list-style-type: none"> • Rally task & plugins
Chris St. Pierre stpierre (irc) cstpierr@cisco.com	<ul style="list-style-type: none"> • Rally task & plugins • Bash guru ;)
Hai Shi shihai1991 (gitter) shihai1992@gmail.com	<ul style="list-style-type: none"> • Rally task & plugins
Illia Khudoshyn ikhudoshyn (irc) ikhudoshyn@mirantis.com	<ul style="list-style-type: none"> • Rally task & plugins
Kun Huang kun_huang (irc) gareth.huang@huawei.com	<ul style="list-style-type: none"> • Rally task & plugins
Li Yingjun liyingjun (irc) yingjun.li@kylin-cloud.com	<ul style="list-style-type: none"> • Rally task & plugins
1.10. Project Info and Release Notes	187
Roman Vasilets rvasilets (irc)	<ul style="list-style-type: none"> • Rally task & plugins

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	<ul style="list-style-type: none">• Cinder plugins
Nikita Kononov NikitaKononov (irc) nkononov@mirantis.com	<ul style="list-style-type: none">• Sahara plugins
Oleg Bondarev obondarev (irc) obondarev@mirantis.com	<ul style="list-style-type: none">• Neutron plugins
Sergey Kraynev skraynev (irc) skraynev@mirantis.com	<ul style="list-style-type: none">• Heat plugins
Spyros Trigazis strigazi (irc) strigazi@gmail.com	<ul style="list-style-type: none">• 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.

1.10.2 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

1.10.3 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-discuss@lists.openstack.org](#) (see [subscription and usage instructions](#));
- [Rally team on Launchpad](#): Answers/Bugs/Blueprints.

1.10.4 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 generic testing of 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 task 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

- **Task 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.
- **Task 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

- **Task Context:**

- [new] **stacks**
 - Generates passed amount of heat stacks for all tenants.
- [new] **custom_image**
 - Prepares images for internal VMs testing.
 - 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

- **Task 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 task plugins, bug fixes, various code and API improvements.

New Features & API changes

- Add the ability to specify versions for clients in 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/14d0b5ba0c75ececfd6a6c121d9cf2810571f77/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

- **Task 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.
- **Task 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

- **Task 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 task 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 testing 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 subtasks to execute in task

This feature request proposes to add the ability to specify subtask(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

- **Task 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.

- **Task 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 scenarios
 - 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 scenarios
 - support size range to be passed to the ‘size’ argument as a dictionary {“min”: <minimum_size>, “max”: <maximum_size>}

- **Task 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 testing of list operations.

- **Task 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 task data processing algorithms that scale: these algorithms do not store exhaustive information about every single subtask 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 info 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):
```

(continues on next page)

(continued from previous page)

```
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 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 `Ironi`, `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:
```

(continues on next page)

(continued from previous page)

```
tenants: 2
users_per_tenant: 2
api_versions:
  cinder:
    version: 2
    service_name: cinderv2
```

- Move rally.osclients.Clients to plugin base

Rally OSClient 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 v0.10.0

Overview

Release date	10/20/2017
--------------	------------

- Ability to use OpenStack deployments without admin credentials
- Better validation process of tasks
- New task format (with an ability to set description for workloads)
- New JSON report
- ElasticSearch exporter
- [OSProfiler](#) support
- Restructure of configuration options.

Details

Command Line Interface

- Introduce `rally task import` command for importing task results into database.
- Extend tags support for tasks. Now you can specify several tags for a single task using `-tag argument`. Also filtering tasks by tags is now available.
- Move DB category from `rally-manage db` to `rally db` and introduce `rally db show` command for printing the used connection string.

Deployments

This release we started a huge work related to simplification of deployment component of Rally. There is a good progress which includes several nice features:

- The format. “ExistingCloud” deployment type covers 99.99% cases and is used as a base for all new things. Also, it will be extended to support different platforms soon. The new format looks like (for OpenStack case):

```
{
  "openstack": {
    "admin": {
      "username": "admin",
      "password": "changeme",
      "tenant_name": "foo",
    },
    "auth_url": "https://example.com",
  }
}
```

- admin user is optional in case of setting existing users. From the beginning, setting admin credentials was a required section of Rally deployment configuration. Even with introducing existing users feature, this behaviour left. Finally, we finished a big code refactoring and admin credential become optional section. If a set of plugins for particular workload doesn't require admin user, you can launch this task at deployment without setting it.

The information about the requirements of plugins you can find at [Plugins Reference page](#) (see `Requires platform(s) :` section at the bottom of each plugin).

- Originally, Rally project was designed to check performance of OpenStack and we succeeded in building awesome tool. We do not plan to stop and just want to inform about our future plans to expand a number of supported platforms. Subscribe to our [GitHub organization](#) to not miss new plugins.

Task component

- The new task format is introduced. It includes a bunch of improvements, unification, etc. All the docs and samples will be updated soon.
As for now, you can check [a spec](#) for this big change.
- SLA failure_rate max=0 become a default if nothing else is specified.
- Totally reworked atomic actions. The atomic actions now supports nested actions which allows to measure durations inside the scenario even more precise. You can find them in HTML report or in our new json report (see `rally task report --json`).
- Generation of names for new resources takes care about particular workload id, so it helps to provide a better cleanup and prepare for new feature - disaster cleanup.

Plugins

We started supporting discovering plugins by entry-points, so you can easily deliver your custom plugins as a simple python package.

To make you package after-discoverable, you need to specify the proper entry-point at your `setup.cfg` file:

Deployment Engines:

Remove serverproviders & rarely used deployers

Unfortunately, seems like nobody is using deployers for deploying their clouds and mostly people would like just to execute their code.

- 1) Remove server provides
- 2) Remove engines that uses server providers

OpenStack clients:

- Deprecate EC2 client. It wasn't used in any of plugins and doesn't support keystone v3
- Move `rally.osclients` module to `rally.plugins.openstack.osclients`

Scenarios:

The old way to describe scenario plugin via method is finally removed. Initially Rally scenario plugins were methods of special class, like below:

```
from rally.task import scenario

class SomeBasicClass(scenario.Scenario):

    @scenario.configure()
    def some_scenario(self, arg1):
        """An implementation of SomeBasicClass.foo scenario."""

    @scenario.configure()
    def another_scenario(self):
        """Implementation of another scenario, SomeBasicClass.bar."""
```

However to unify scenarios with other plugins we moved to model where plugin is class. It was done long time ago.

```
from rally.task import scenario

@scenario.configure(name="CustomName")
class Some(scenario.Scenario):

    def run(self, arg1):
        """An implementation of the scenario."""
```

We had a bunch of code that was used for transition and backward compatibility that we finally removed.

- *NEW!!*
- CinderQos.create_and_get_qos
- CinderQos.create_and_list_qos
- CinderQos.create_and_set_qos
- CinderQos.create_qos_associate_and_disassociate_type
- CinderVolumeTypes.create_and_get_volume_type
- CinderVolumeTypes.create_and_list_volume_types
- CinderVolumeTypes.create_and_update_encryption_type
- CinderVolumeTypes.create_and_update_volume_type
- CinderVolumeTypes.create_get_and_delete_encryption_type
- CinderVolumeTypes.create_volume_type_add_and_list_type_access
- Dummy.openstack
- GlanceImages.create_and_deactivate_image
- GlanceImages.create_and_download_image
- GlanceImages.create_and_get_image
- GlanceImages.create_and_update_image

- `K8sPods.create_pods`
- `K8sPods.create_rcs`
- `K8sPods.list_pods`
- `ManilaShares.create_and_extend_share`
- `ManilaShares.create_and_shrink_share`
- `ManilaShares.create_share_then_allow_and_deny_access`
- `NeutronBGPVPN.create_and_delete_bgpvpns`
- `NeutronBGPVPN.create_and_list_bgpvpns`
- `NeutronBGPVPN.create_and_list_networks_associations`
- `NeutronBGPVPN.create_and_list_routers_associations`
- `NeutronBGPVPN.create_and_update_bgpvpns`
- `NeutronBGPVPN.create_bgpvpn_assoc_disassoc_networks`
- `NeutronBGPVPN.create_bgpvpn_assoc_disassoc_routers`
- `NeutronNetworks.create_and_show_ports`
- `NeutronNetworks.create_and_show_routers`
- `NeutronNetworks.create_and_show_subnets`
- `NeutronNetworks.set_and_clear_router_gateway`
- `NeutronSecurityGroup.create_and_delete_security_group_rule`
- `NeutronSecurityGroup.create_and_list_security_group_rules`
- `NeutronSecurityGroup.create_and_show_security_group`
- `NeutronSecurityGroup.create_and_show_security_group_rule`
- `NovaServerGroups.create_and_delete_server_group`
- `NovaServerGroups.create_and_get_server_group`
- `NovaServers.boot_and_get_console_url`
- `NovaServers.boot_server_and_attach_interface`
- `NovaServers.boot_server_and_list_interfaces`
- `NovaServers.boot_server_attach_volume_and_list_attachments`
- *UPDATED!!*
- The new argument `properties` is added to scenario `IronicNodes.create_and_list_node`
- *DELETED*

Fuel and Nova-Network are not alive any more. So we removed those scenarios. If any of those scenarios a critical for you, please contact us.

- `FuelEnvironments.create_and_delete_environment`
- `FuelEnvironments.create_and_list_environments`
- `FuelNodes.add_and_remove_node`
- `NovaFloatingIpsBulk.create_and_delete_floating_ips_bulk`

- NovaFloatingIpsBulk.create_and_list_floating_ips_bulk
- NovaNetworks.create_and_delete_network
- NovaNetworks.create_and_list_networks
- NovaSecGroup.boot_and_delete_server_with_secgroups
- NovaSecGroup.boot_server_and_add_secgroups
- NovaSecGroup.create_and_delete_secgroups
- NovaSecGroup.create_and_list_secgroups
- NovaSecGroup.create_and_update_secgroups

Validators:

The validators refactoring was a long-term task which blocked us to abandon alignment to only OpenStack platform and requirements of setting admin credentials. In this release, we made a great progress and fixed a lot of issues and blockers which made possible to refactor validators. Now validation step is equal for all types of plugins (Scenario, SLA, Context, Hooks, Runners, etc).

The old way to add validator for scenario is deprecated. The new unified way looks like:

```
import time

from rally.common import validation
from rally.task import scenario

@validation.add("number", param_name="timeout", minval=0)
@scenario.configure(name="Foo.bar")
class FooScenario(scenario.Scenario):
    def run(self, timeout):
        time.sleep()
```

The old validators from `rally.task.validators` module is deprecated too, see equivalents which can be used with add decorator:

- required_openstack -> required_platform with setting platform argument to “openstack”
- external_network_exists -> ‘external_network_exists <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#external-network-exists-validator>’_
- file_exists -> ‘file_exists <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#file-exists-validator>’_
- flavor_exists -> ‘flavor_exists <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#flavor-exists-validator>’_
- image_exists -> ‘image_exists <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#image-exists-validator>’_
- image_valid_on_flavor -> ‘image_valid_on_flavor <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#image-valid-on-flavor-validator>’_
- number -> ‘number <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#number-validator>’_
- required_api_versions -> ‘required_api_versions <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-api-versions-validator>’_
- required_cinder_services -> ‘required_cinder_services <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-cinder-services-validator>’_

- `required_clients` -> `'required_clients` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-clients-validator>‘_
- `required_contexts` -> `'required_contexts` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-contexts-validator>‘_
- `required_neutron_extensions` -> `'required_neutron_extensions` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-neutron-extensions-validator>‘_
- `required_param_or_context` -> `'required_param_or_context` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-param-or-context-validator>‘_
- `required_services` -> `'required_services` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#required-services-validator>‘_
- `restricted_parameters` -> `'restricted_parameters` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#restricted-parameters-validator>‘_
- `validate_heat_template` -> `'validate_heat_template` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#validate-heat-template-validator>‘_
- `volume_type_exists` -> `'volume_type_exists` <https://rally.readthedocs.io/en/0.10.0/plugins/plugin_reference.html#volume-type-exists-validator>‘_
- `workbook_contains_workflow` -> `workbook_contains_workflow`
- `network_exists` is removed, since we do not find any customers for it. Please contact us if it was useful for you.
- `validate_share_proto` is removed in favor of enum validator

Fixed bugs

- [plugins] JSON schema of `servers` context allows to transmit a list of nics in two formats. First one is a format that novaclient expects to see (each network should be represented like `{"nic-id": "the id of the network"}`). The second one is more user-friendly - just list of strings (each network should be represented just by id of the network). Unfortunately, the second case was not covered by our code base.

Also, the first described format works in limited cases due to bad serialization.

[Launchpad bug-report #1695245](#)

- [deployment] `~/rally/.openrc` not working for keystone v3
- [Launchpad bug-report #1683820](#)
- [plugins] Failed to list volumes in case of missed name in the object.
 - [backported into 0.9.1][deployment] Credentials is not updated as soon as deployment is recreated. Need to call recreate request twice.
- [Launchpad bug-report #1675271](#)
- [backported into 0.9.1][plugins] Scenario `IronicNodes.create_and_list_node` had a wrong check that list of all nodes contains newly created one.
 - [backported into 0.9.1][task][cleanup] Do not remove quotas in case of existing users
 - [backported into 0.9.1][task][cleanup] Various traces of neutron resources
 - [backported into 0.9.1][core] Keystone v3, authentication error for Rally users if the value of `project_domain_name` of admin user isn't equal "default"

[Launchpad bug-report #1680837](#)

- [backported into 0.9.1][task] Scenario `NovaHosts.list_and_get_hosts` obtains hostname for all hosts. But it fails in some environments if host is not compute.

[Launchpad bug-report #1675254](#)

- [backported into 0.9.1][verification] Rally fails to run on systems on which python-virtualenv is not installed

[Launchpad bug-report #1678047](#)

- [backported into 0.9.1][verification] CLI `rally verify rerun` fails with `TypeError` due to wrong integration with Rally API.

- [plugins] Rally fails while creating neutron router on the clouds where ext-gw-mode extension is not installed.

- [plugins] Scenario `CinderVolumes.create_nested_snapshots_and_attach_volume` fails on a big load due to the fact that one server is used for several iterations. In such case we are facing 2 issues: the maximum number of volumes per VM is 26 (which is a number of available names for volumes); detaching volume of one iteration can block attaching of other iterations.

[Launchpad bug-report #1708160](#)

Thanks

2 Everybody!

Rally v0.10.1

Overview

Release date	12/05/2017
--------------	------------

Details

This release is fully backward compatible with Rally 0.10.0 and contains just bug fixes.

Fixed bugs

- [deployment] Suppress deprecation warning about an old format in case of using `-fromenv` option of `rally deployment create`
- [deployment] Failure `rally deployment show` while displaying the information about deployment with a config in an old format.

- [task] New json report processed the hook results in a wrong way

[Launchpad bug-report #1734336](#)

- [task] Failure while generating trends reports in case of failures in setup method of any context

[Launchpad bug-report #1732193](#)

- [task] Failure to export results in Elasticsearch 5.x cluster in case of extra `/` in the end of destination url.

Thanks

2 Everybody!

Rally v0.11.0

Overview

Release date	02/16/2018
--------------	------------

- Stabilize Rally releases (see requirements section)
- Publishing docker images is returned!
- Environment introduction (see Deployment section)

Details

Requirements

As for long time we tried to make our releases stable and workable even after a year from release date. For this purpose, upper limits for all our requirements were used. Such solution helped to make releases more stable, but it did not work well and created more pain than a profit. The main issue was related to new releases of not-direct rally dependencies which can be incompatible with each other.

From Rally 0.11.0 we are stopping adding upper version limits for our [requirements](#) (this does not apply to cases when we are sure that some new releases of dependency broke us for sure).

As an alternative solution, the [upper-constraints file](#) is selected. It is a file with a list of packages and their versions which we used in our CI while testing. Versions from this list are suggested to use in production. Please note, that it also contains not direct Rally dependencies (dependencies of rally dependencies and dependencies of dependencies of rally dependencies as well) and packages which possibly doesn't relate to Rally at all.

The example of usage:

Logging

The default value of `use_stderr` option of Rally config is changed to **True**. It is done to ensure that json output of rally commands will not be messed with logging and integration with third-party tools and scripts should become simpler.

Docker

Unfortunately, we lost access to [rallyforge organization at DockerHub](#), so our docker images were not published anywhere officially for some time. Docker Support did not help us a lot. At least, the original repo is removed now and we can start new organization at DockerHub from the scratch.

The new docker images for Rally+OpenStack plugins with an updated HowTo you can find at [xrally/xrally-openstack repo](#). It contains all Rally releases + latest tag which maps to master branch.

Command Line Interface

- Introduce `rally db ensure` command. It is going to create Rally DB if it doesn't exist, otherwise it does nothing.
- Various improvements for `rally db`. Such as printing results of operations as well as connection string to DB, hiding credentials by default, etc.
- Various changes in returning error codes. Error codes become different for different exceptions. Also, `rally task start` began to return 1 in case of any runtime issues and 2 if any workload doesn't pass it's SLA.
- The new CLI for new component (see `Environments & Deployments` section for more details) - `rally env`

Environments & Deployments

Deployment Component is fundamental part of Rally which is used by Task and Verification components. Whereas Task and Verification components experienced redesigns (some parts were redesigned even several times), Deployment component was only extended over the time. Currently, we are at the point when further extending requires much work and user experience become worth. It is a hard decision, as like we had done with Verification component in Rally 0.8, the Deployment component is re-written from the scratch. To be clear, the new version of the component has not only the new code, but the new name as well - it is the Environment.

We are at the stage when Rally is suitable not only for OpenStack but for various platforms and systems. The Environment is some entity against which Rally should perform load. The specific Environment can include credentials for OpenStack or for Kubernetes or for both. The Environment is a way to describe the complex system/cloud which all of us have. As well it can be used for simple systems as easy as for complex.

If you are regular Rally user which tests OpenStack clusters, nothing is changing for you at this moment. `rally deployment` CLI still works and it is not even deprecated. It will be kept for a while. But you can start looking at our new CLI `rally env`.

As for writing plugins for external platforms - we are working on updating our documentary.

Task component

- The json results are extended with context execution stats (i.e when and which context was executed, how much time setup and cleanup methods took, etc). Also, `volumes@openstack` and `volume_types@openstack` are ported to include detailed stats of executed actions. In further releases, all existing contexts will be adopted to the similar behavior.
- Better OSProfiler integration
Rally environment&deployment config began accept `profiler_conn_str` property which is used to generate OSProfiler native html-report and to embed it to Rally's html-report.
- HTML report is extended to include a timestamp of failures.

Plugins

As it mentioned above, the Deployment Component will be replaced soon. Such decision led to abandoning all deployment-related plugins (engines, providers, etc).

Scenarios:

- *NEW!!*
`NovaServers.boot_server_attach_volume_and_list_attachments`

- *UPDATED!!*
- Extend several Nova&Neutron related scenarios with `create_floating_ip_args` parameter
`NovaServers.boot_and_associate_floating_ip` `NovaServers.boot_server_associate_and_dissociate_floating_ip`
- Modify Bgpvpn scenarios to test true bgpvpn
All Bgpvpn scenarios began to boot a server to add active port in the network associated to the bgpvpn which allows to test not only the record in the database, but true bgpvpn

Contexts:*UPDATED!!*`network@openstack` context is extended with ability to specify external router information.**Fixed bugs**

- [backported into 0.10.1][deployment] Suppress deprecation warning about an old format in case of using `--fromenv` option of rally deployment create
- [backported into 0.10.1][deployment] Failure `rally deployment show` while displaying the information about deployment with a config in an old format.
- [backported into 0.10.1][task] New json report processed the hook results in a wrong way
[Launchpad bug-report #1734336](#)
- [backported into 0.10.1][task] Failure while generating trends reports in case of failures in setup method of any context
[Launchpad bug-report #1732193](#)
- [backported into 0.10.1][task] Failure to export results in Elasticsearch 5.x cluster in case of extra / in the end of destination url.
- [deployment] OpenStack deployment creation with `--fromenv` option used old deprecated format.
- [verify] Rally did not support creating verifiers from Gerrit/SSH source.
[Launchpad bug-report #1737529](#)
- [task][openstack] Removing default security group in `users@openstack` context did not take into account that neutron can return multiple resources in some configuration instead of one security group which relates to requested tenant.
- [task][openstack] Existing openstack users get their roles un-assigned if they are equal to what `roles@openstack` context is configured to assign.
[Launchpad bug-report #1720270](#)
- [task][openstack] Validation step ignores `roles@openstack` context and marks as “invalid” valid cases
Some actions in openstack can be performed only if the user has specific role. Since these roles can be different in different OpenStack installations Rally has `roles@openstack context` context which can assign roles to the users. Validation step did not check for specified roles in workload config and made wrong assumption about accessibility of resources
[Launchpad bug-report #1539878](#)
- [task][openstack] Wrong identifiers were used for filtering Mistral resources while cleanup step.

- [task][openstack] [NovaServers.boot_and_live_migrate_server](#) does wrong target host selection
[Launchpad bug-report #1734914](#)

Thanks

2 Everybody!

Rally v0.11.1

Overview

Release date	02/27/2018
--------------	------------

- Fix database migration
- Un-cup kubernetes client version in requirements
- Add support for sTestr for verifiers
- Add several new scenarios for Gnocchi

Details

DataBase

Rally <0.10.0 was hardcoded to support only OpenStack platform. That is why deployment config had a flat schema (i.e openstack credentials were at the same top-level as other properties).

Rally 0.10 includes an attempt to unify deployment component for supporting multiple platforms. The deployment config was extended with a new top level property `creds` which was designed to include credentials for different platforms. Since Task and Verification components used `deployment.credentials` object from database instead of using deployment config directly, Rally 0.10 did not provide a database migration of deployment config.

While developing Rally 0.11.0 with new Environment component, we made a wrong assumption and forgot about an old format. That is why a 7287df262dbc migration relied on “creds” property of `deployment.config`

If the database was created before Rally<0.10, the introduced assumption leads to `KeyError failure[0]` for old deployment configuration:

```
File ".../7287df262dbc_move_deployment_to_env.py", line 137, in upgrade
    and (set(spec["creds"]) == {"openstack"})
KeyError: 'creds'
```

We fixed this issue and you should easily migrate from Rally < 0.11.0 to Rally 0.11.1 without any issues.

Verification component

OpenStack Tempest team made a decision to switch from `testrepository` test runner to `stestr` which is fork of `testrepository`.

Despite the fact that `stestr` is not 100% backwards compatible with `testrepository`, it is not a hard task to make `Tempest verifier` work with both of them (to support new releases of tempest tool as like old ones) and it is what we did in Rally 0.11.1

Plugins

Scenarios:

- *NEW!!*

```
GnocchiArchivePolicyRule.list_archive_policy_rule      GnocchiArchivePolicyRule.create_delete_archive_policy_rule
GnocchiArchivePolicyRule.create_delete_archive_policy_rule
```

Thanks

2 Everybody!

Rally v0.11.2

Release date	03/29/2018
--------------	-------------------

- OpenStack plugins moved to the separate [repo](#) and [package](#) . In-tree OpenStack plugins are deprecated now and will be removed soon. All further development should be done in the new repository.
- Environment manager and Platform plugins are extended with a new feature - creating a spec based on system environment variables. We had similar feature in Deployment component like below, but it handles only OpenStack platform.

```
$ rally deployment create --name "my-deployment" --fromenv
```

The new feature is not limited by one platform and each platform plugin can implement it. The usage of the feature is still pretty simple:

```
$ rally env create --name "my-env" --from-sysenv
```

- There is a new argument for *rally env show* command: **—only-spec**. It is a trigger for showing only a specification of the environment
- Methods for association and dissociation floating ips were deprecated in novaclient a year ago and latest major release (python-novaclient 10) [doesn't include them](#). These actions should be performed via neutronclient now. It is not as simple as it was via Nova-API and you can find more neutron-related atomic actions in results of scenarios.
- *os-hosts* CLIs and python API bindings had been deprecated in python-novaclient 9.0.0 and became removed in [10.0.0 release](#). This decision affected 2 scenarios [NovaHosts.list_hosts](#) and [NovaHosts.list_and_get_hosts](#) which become redundant and we cannot leave them (python-novaclient doesn't have proper interfaces any more).
- Improvements of elasticsearch task exporter to cover the case when success rate of workload is not in range of 0-100. For example, it can happen when context fails.

Rally v0.12.0

Release date	05/08/2018
--------------	-------------------

Warning: It is friendly reminder about the future of in-tree OpenStack plugins. All further development is done in a [separate project](#). In-tree plugins are deprecated and will be removed in next major release!

- Improve performance of *rally task import* command.
- Port internals of Verification component to support pip 10
- Extend plugins interface to provide config options to load. An example of *setup.cfg/setup.py*:
Method *list_opts* in the above example, returns a dict where key is a category name, value is a list of options.
- Rework *ResourceType* plugin type. Previously, it was hard-coded for OpenStack resources only (initialization of OpenStack clients).

An old interface looked like:

```
from rally.common.plugin import plugin
from rally.task import type

@plugin.configure(name="glance")
class GlanceResource(type.ResourceType):
    @classmethod
    def transform(cls, clients, resource_config):
        """Transform the resource config to id.

        :param clients: Initialized OpenStack clients
        :param resource_config: a dict with resource description
            taken from workload
        """
        pass
```

The new one:

```
from rally.common.plugin import plugin
from rally.task import type

@plugin.configure(name="glance")
class GlanceResource(type.ResourceType):
    def __init__(self, context, cache=None):
        """init method

        :param context: A context object as like other plugins accept.
        :param cache: A global cache which can be used for listing
            the similar resources.
        """
        super(GlanceResource, self).__init__(context, cache)
        # NOTE #1: the next code is copy-pasted from
        # *rally_openstack.types.OpenStackResourceType* class and
        # there is no need to copy it to plugins itself, just inherit
        # from the right parent.
        # NOTE #2: the following code is equal to what we have in
        # an old ResourceType implementation. Property *self._clients*
        # is what was transmitted to transform method as *clients*
        # argument
        self._clients = None
        if self._context.get("admin"):
            self._clients = osclients.Clients(
                self._context["admin"]["credential"])
```

(continues on next page)

(continued from previous page)

```

elif self._context.get("users"):
    self._clients = osclients.Clients(
        self._context["users"][0]["credential"])

def pre_process(self, resource_spec, config):
    """Pre process the resource config to id.

    :param resource_spec: a dict with resource description
        taken from workload
    :param config: A resource specification from scenario
        plugin. Usually it contains only *type* of resource.
    """
    #
    pass

```

Fixed bugs

- Fix deprecated `--tasks` argument of `rally task report`. Use `--uuid` instead.
- Fix printing warning of an old deprecated deployment configuration format.

Thanks

2 Everybody!

Rally v0.12.1

Release date	05/15/2018
--------------	------------

Warning: It is friendly reminder about the future of in-tree OpenStack plugins. All further development is done in a [separate project](#). In-tree plugins are deprecated and will be removed in next major release!

- Fix loading configuration options from external plugins.

Thanks

2 Everybody!

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] Add more Nova API scenarios

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 returning 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 (beginning 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" instead 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 OSCClient 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 platform).
- 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 keystone session, 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 expects “classname” JUnitXML attribute instead of “class_name”.

Thanks

2 Everybody!

Rally v0.9.1

Overview

Release date	4/12/2017
--------------	-----------

Details

Unfortunately, Rally 0.9.0 contains various bugs. We work hard to fix them, improve our CI to avoid such issues in future and ready to present a new Rally release which includes only bug-fixes.

Fixed bugs

- [deployment] Credentials is not updated as soon as deployment is recreated. Need to call recreate request twice.
[Launchpad bug-report #1675271](#)
- [task] Scenario [IronicNodes.create_and_list_node](#) had a wrong check that list of all nodes contains newly created one.
- [task][cleanup] Do not remove quotas in case of existing users
- [task][cleanup] Various traces of neutron resources
- [core] Keystone v3, authentication error for Rally users if the value of project_domain_name of admin user isn't equal “default”
[Launchpad bug-report #1680837](#)
- [task] Scenario [NovaHosts.list_and_get_hosts](#) obtains hostname for all hosts. But it fails in some environments if host is not compute.
[Launchpad bug-report #1675254](#)
- [verification] Rally fails to run on systems on which python-virtualenv is not installed
[Launchpad bug-report #1678047](#)
- [verification] CLI [rally verify rerun](#) fails with TypeError due to wrong integration with Rally API.

Thanks

2 Everybody!

Rally v0.9.2

Overview

Release date	10/20/2017
--------------	------------

Details

The latest OpenStack merged a bunch of incompatible changes. This release is an attempt to fix compatibility issues in Rally 0.9. Now it works well for old and new OpenStack releases.

Note: OpenStack Nova abandoned networking and image API. It is impossible to do anything with it, so we suggest you to use Neutron and Glance instead.

Fixed bugs

- [broken dependency] One of hook plugins required ansible which released incompatible version. To fix this, the proper version of os-faults lib should be used
- [keystone] The format of keystone URL's completely change which resulted in a wrong processing it from our side.

Thanks

2 Everybody!

Rally v1.0.0

Release date	06/20/2018
--------------	------------

Warning: There is no in-tree OpenStack plugins anymore. They moved to the separate project .

Starting from the current release all notes are written in a single file [CHANGELOG.rst](#) Follow it for more details.

Thanks

2 Everybody!

Rally v1.0.0

Release date	06/20/2018
--------------	------------

Warning: There is no in-tree OpenStack plugins anymore. They moved to [the separate project](#).

Starting from the current release all notes are written in a single file [CHANGELOG.rst](#) Follow it for more details.

Thanks

2 Everybody!

B

`base_ref (rally.verification.reporter.VerificationReporter`
attribute), 126

C

`configure () (rally.verification.manager.VerifierManager`
method), 128

E

`extend_configuration ()`
(rally.verification.manager.VerifierManager
method), 128

G

`generate () (rally.verification.reporter.VerificationReporter`
method), 126

`get_configuration ()`
(rally.verification.manager.VerifierManager
method), 128

I

`install () (rally.verification.manager.VerifierManager`
method), 128

`install_extension ()`
(rally.verification.manager.VerifierManager
method), 128

`is_configured () (rally.verification.manager.VerifierManager`
method), 128

L

`list_extensions ()`
(rally.verification.manager.VerifierManager
method), 128

`list_tests () (rally.verification.manager.VerifierManager`
method), 129

M

`make () (rally.verification.reporter.VerificationReporter`
static method), 126

O

`override_configuration ()`
(rally.verification.manager.VerifierManager
method), 129

R

`run () (rally.verification.manager.VerifierManager`
method), 129

U

`uninstall () (rally.verification.manager.VerifierManager`
method), 129

`uninstall_extension ()`
(rally.verification.manager.VerifierManager
method), 129

V

`validate () (rally.verification.reporter.VerificationReporter`
class method), 126

`validate_args () (rally.verification.manager.VerifierManager`
method), 129

`VerificationReporter (class in`
rally.verification.reporter), 126

`VerifierManager (class in`
rally.verification.manager), 128