Rally

Introduction

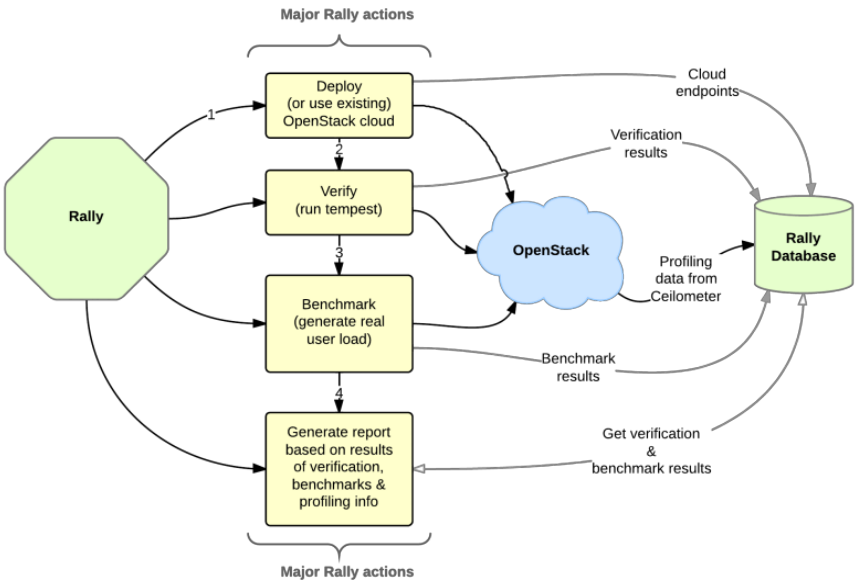
Rally is a Benchmark-as-a-Service project for OpenStack.

Rally is intended to provide the community with a benchmarking tool that is capable of performing **specific**, **complicated** and **reproducible** test cases on **real deployment** scenarios.

If you are here, you are probably familiar with OpenStack and you also know that it's a really huge ecosystem of cooperative services. When something fails, performs slowly or doesn't scale, it's really hard to answer different questions on "what", "why" and "where" has happened. Another reason why you could be here is that you would like to build an OpenStack CI/CD system that will allow you to improve SLA, performance and stability of OpenStack continuously.

The OpenStack QA team mostly works on CI/CD that ensures that new patches don't break some specific single node installation of OpenStack. On the other hand it's clear that such CI/CD is only an indication and does not cover all cases (e.g. if a cloud works well on a single node installation it doesn't mean that it will continue to do so on a 1k servers installation under high load as well). Rally aims to fix this and help us to answer the question "How does OpenStack work at scale?". To make it possible, we are going to automate and unify all steps that are required for benchmarking OpenStack at scale: multi-node OS deployment, verification, benchmarking & profiling.

Rally workflow can be visualized by the following diagram:



Installation

Execute the installation script

$ git clone https://github.com/noironetworks/rally.git

Change git branch

$ git checkout vCPE

To install Rally system wide by running script as root

$ sudo ./install_rally.sh

To change configurations, edit */etc/rally/rally.conf*

Running rally

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.

Example :- *$. openrc admin*

$ rally deployment create --fromenv --name=testing

OR

$ rally deployment create --file=existing.json --name=testing

Sample existing.json file:

```
{
        "openstack": {
        "auth_url": "http://keystone_url:5000/v3",
        "region_name": "RegionOne",
        "endpoint_type": "public",
        "admin": {
        "username": "admin",
        "password": "password",
        "user_domain_name": "admin_domain",
        "project_name": "admin",
        "project_domain_name": "admin_domain"
        },
        "users": [
        {
                "username": "non_admin_user",
                "password": "password",
                "tenant_name": "some_tenant"
        }
        ]
        }
}
```

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

```
$ rally deployment check
```

Rally Task

1. Running Rally Tasks

Now that we have a working and registered deployment, we can start testing it. The sequence of subtasks to be launched by Rally should be specified in a task input file (either in JSON or YAML format). To start a task, run the *task start* command.

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

2. Rally input task format

Rally comes with a collection of plugins and in most cases we use multiple plugins to test your OpenStack cloud.To do so the following syntax is used.

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

3. 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.*uec$"}'*
   *$ rally task start task.yaml --task-args 'image_name: "^cirros.*uec$"'*

   Or 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.*uec$"
   }

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

Benchmarking

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

```
{
```

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

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

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

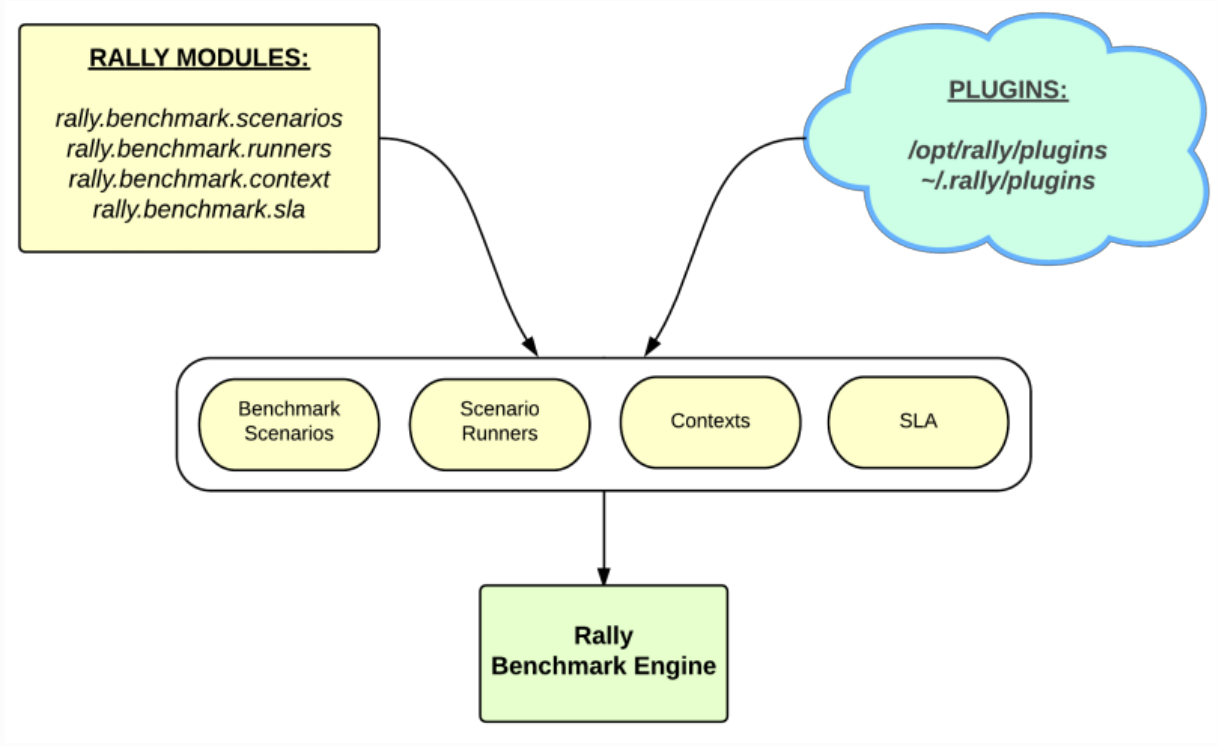
Report generation

Rally enables you to create illustrative and comprehensive HTML reports based on the task data.
This is going to produce an HTML page with the overview of all the scenarios that you've included into the last task completed in Rally

Example :- *$ rally task report --uuid --out=report1.html --open*

Rally Plugins

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



Just place a python module with your plugin class into the **~/.rally/plugins** directory (or it's subdirectories), and it will be autoloaded. Additional paths can be specified with the ***--plugin-paths*** argument ;

$ rally --plugin-paths /rally/aci_plugins ...

Verifying cloud via Tempest verifier

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.

Tempest Integration with Rally

- Execute the following command to create a Tempest verifier:
  $ rally verify create-verifier --type tempest --name tempest-verifier
- Execute the following command to configure the Tempest verifier for the current deployment:
  $ rally verify configure-verifier
- In order to start a verification execute the following command:
  $ rally verify start

CLI commands and its function

Command	Function
rally deployment create --fromenv --name=existing	Create deployment from environment
rally deployment create --file=<credentials.json> --name=<name>	Create deployment from json file
rally deployment check	Verify current deployment is healthy and ready to be tested
rally deployment list	List all deployments
rally deployment use <deployment name>	Use a specific deployment
rally task start <task.json/task.yaml>	To start a task
rally task list	List all tasks
rally task detailed --uuid<UUID>	Details of a task

rally task report <task-UUID> --out output.html	Report of task in html
rally task trends --tasks <uuid-1> <uuid-2> <uuid-3> --out trends.html	To compare between different tasks
rally plugin list --name <name>	List all Rally plugins filtered by name
rally plugin show <plugin name>	Details of a plugin
rally verify list-verifiers	List all verifiers
rally verify list-plugins	List all plugins for verifiers management
rally verify create-verifier --type tempest --name tempest-verifier	To create tempest verifier
rally verify delete-verifier --id <verifier id>	To delete a verifier
rally verify configure-verifier --deployment-id <UUID or name of a deployment>	To configure verifier
rally verify configure-verifier --show	To show configuration file
rally verify start --id <UUID or name of a verifier> --deployment-id <UUID or name of a deployment>	To start verification
rally verify start --pattern set=<set name>	To verify a specific set
rally verify start --pattern <pattern>	To verify a specific pattern
rally verify rerun --uuid <verification UUID>	To rerun a verification
rally verify report --uuid <uuid-1> <uuid-2> <uuid-3> --type html --to ./report.html	Report of verification in html

References

Github link: https://github.com/openstack/rally.git , https://github.com/noironetworks/rally.git
Official doc : https://docs.openstack.org/rally/latest/
Setting up env :
https://docs.openstack.org/rally/latest/quick_start/tutorial/step_1_setting_up_env_and_running_benchmark_from_samples.html
Sample existing.json files :
https://github.com/openstack/rally/tree/stable/0.12/samples/deployments