Software Quality Assurance (SQA) Report

22-26 Aug 2016

opie (OpenStack Preemptible Instances Extension)

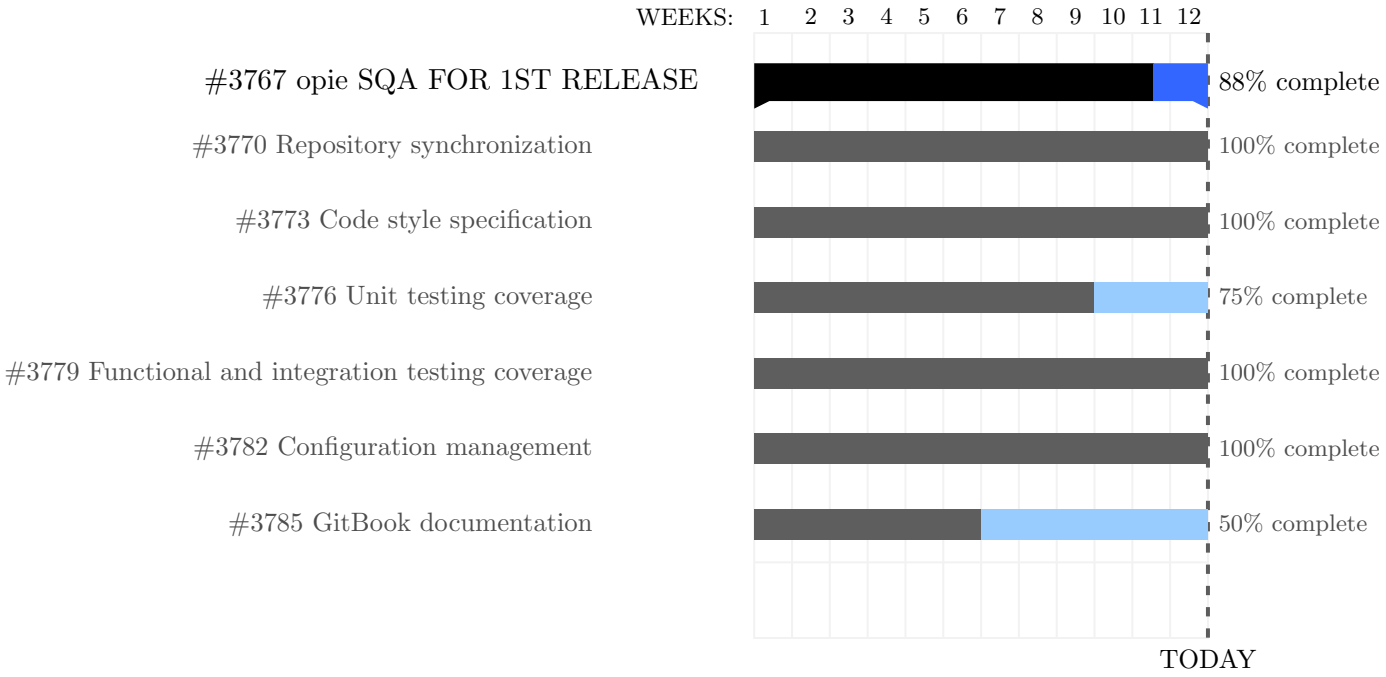
SQA Progress Status COMPLETE

88% done

GitHub repository	COMPLETE
Code style adherence	COMPLETE
Code coverage	92%
Functional/integration testing	COMPLETE
GitBook documentation	COMPLETE
Automated deployment	NOT COMPLETE

Part I

Task Progress for the 1st Release



1 Repository synchronization

Products contributing to INDIGO-DataCloud project must have their code avaiable under GitHub's `indigo-dc` organization.

Repository exists under `indigo-dc` GitHub organization:

- https://github.com/indigo-dc/opie

2 Code Style

Products contributing to INDIGO-DataCloud project are expected to be adhered to a community or de-facto standard code style definition. Exceptions can be made to the selected standard. Custom style guides are accepted but nonetheless not recommended.

Code style definition	OpenStack Style Guidelines
Community/de-facto standard	Yes
Exceptions	0
Richness	111 (+pep8 +flake8) Errors None Warnings None link

2.1 Build status

Last build status on Jenkins CI opie-codestyle.

3 Unit Testing

Code coverage will be tracked for the INDIGO-DataCloud related products and must not decrease during the project's duration. Recommended threshold is 70%.

4 Functional/Integration testing

Functional testing must cover at least the basic functionalities that the product was requested to fulfill within the INDIGO-DataCloud project scope. Integration testing must cover the interactions with other components. Both types of testing will be automated whenever feasible by integrating them in the project's continuous integration service.

No functional or integration testing provided.

4.1 Test coverage

1. add_retry_host
2. hosts_up
3. invalid_max_attempts
4. max_attempts
5. post_select_populate
6. schedule_chooses_best_host
7. schedule_happy_day
8. schedule_host_pool
9. schedule_large_host_pool
10. schedule_with_pci_requests
11. select_destinations

12. select_destinations_no_valid_host
13. select_destinations_no_valid_host_not_enough
14. select_destinations_notifications
15. detect_not_overcommit
16. detect_overcommit_cpu
17. detect_overcommit_disk
18. detect_overcommit_ram
19. detect_preemptible
20. detect_preemptible_empty
21. detect_preemptible_false
22. hosts_up
23. schedule_happy_day
24. schedule_happy_day_preemptible
25. select_destinations
26. select_destinations_kill_preemptible
27. select_destinations_kill_preemptible_empty
28. select_destinations_premptible
29. terminate_preemptible_instances

4.2 Observations

- Product team has provided the command/s to deploy the automated testing.

5 GitBook documentation

Product-related documentation must be uploaded to GitBook's `indigo-dc` central repository. Types of documentation includes a) Developer b) Deployment and Administration c) Command-line Interface (CLI) and Application Program Interface (API) d) User Documentation. All these types may not be applied for every product. Those products that offer functionalities out of the scope of INDIGO-DataCloud project needs may not provide all the spectrum, but links to the official documentation.

Documentation available under `indigo-dc` GitBook organization:

https://indigo-dc.gitbooks.io/opie/content/

5.1 Types of documentation currently provided

Readme Developer guide

5.2 Observations

- Points to improve in the documentation:
  - Describe the installation steps using the INDIGO repository.
  - Explain what are preemptible instances, benefits, etc.
  - Also required a description of the features/workflow (VMs are killed or suspended, how to enable/disable preemptible extension, could it be enabled for a certain tenant?, etc.)

6 Configuration Management

Those products released by INDIGO-DataCloud project that need to be deployed by the end user must rely on a maintained open-source configuration management tool to provide an automated means to install and configure the product. The recommended tool is `Ansible`.

Product does not currently have an automated deployment at INDIGO-DataCloud's Jenkins CI.

6.1 Observations

- From the product team: '*opie needs manual configuration of the installed nova configuration files and the patching of one file, so a manual action is needed anyway. In this case I think that documentation is a better option, as the setup is not complicated at all. If I manage those files (i.e. /etc/nova/nova.conf) and they are already managed by the configuration management (i.e. puppet or ansible) there would be conflicts (at least in puppet you cannot define a resource twice). I could modify the file using augeas, but this is an external dependency for Puppet, that is often not installed. With this in mind, the configuration management module will only install the packages, so I do not know how to proceed.*'

Part II

How to read this document

1 Summary (front) page

Both the overall product's SQA adherence and per-task status codes are explained below:

COMPLETE	Task has been successfully completed and fulfills the project's SQA requirements, listed in Deliverable D3.1 and Extensions to Software Quality Assurance documents.
NOT COMPLETE	Task has not been completed, yet some missing required bits have not been provided.
IN PROGRESS	Task has not been completed, but can proceed as it is.
WP3 PENDING	Task has some pending work from WP3 side, meaning that the product team already submitted the required data but it has not been yet consumed by WP3.

2 Task Progress

2.1 Code style

Code style definition	Name and link of the standard to which the product is adhered.
Community/de-facto standard	Whether the adopted standard is community-wide accepted.
Exceptions	Number of exceptions from the standard definition.
Richness	Number of rules defined in the adopted standard. Additionally (whenever available) the number of errors, number of warnings documented in the standard will be displayed as well as the link to the latest definition.

2.2 Unit testing

This section will display the a) trend graph with the evolution of the code coverage over time and b) the Cobertura report, with the coverage results of different methods. Both are taken from the project's Jenkins continuous integration service.
Note: resultant coverage value is the lowest of the ones for the different methods: packages, files, classes, lines, conditionals.

2.3 Functional/Integration testing

2.4 GitBook documentation

Whenever the documentation of the product is available at the project's GitBook repository, both the a) **link** to the documentation index and b) **type of documentation** provided will be displayed in the report.

2.5 Configuration Management

Whenever the product has an recipe to be deployed automatically the following information will be available:

Tool	Configuration management tool used.
Manifest link	URL pointing to the manifest/s.
Deployment level	Whether `installation`, `configuration` or both.
Build status	Current build status for the project's supported OS distributions.