Table of Contents

# 1 INTRODUCTION

This document provides the overall testing strategy and approach required to ensure that the requirements of the rolling upgrade OpenStack Mitaka – Newton (master) for Cinder, Swift and Nova projects – For now on referred as “The Projects” – are tested adequately, and that the required level of quality and reliability of the software deliverables is attained.

Master Test Plan is initiated in the Planning phase, however, this document could be updated throughout the project.

## 1.1 PURPOSE AND SCOPE

The purpose of this document is to communicate activities related to the planning, staffing, managing and execution of testing activities for the rolling upgrade joint deliverable.

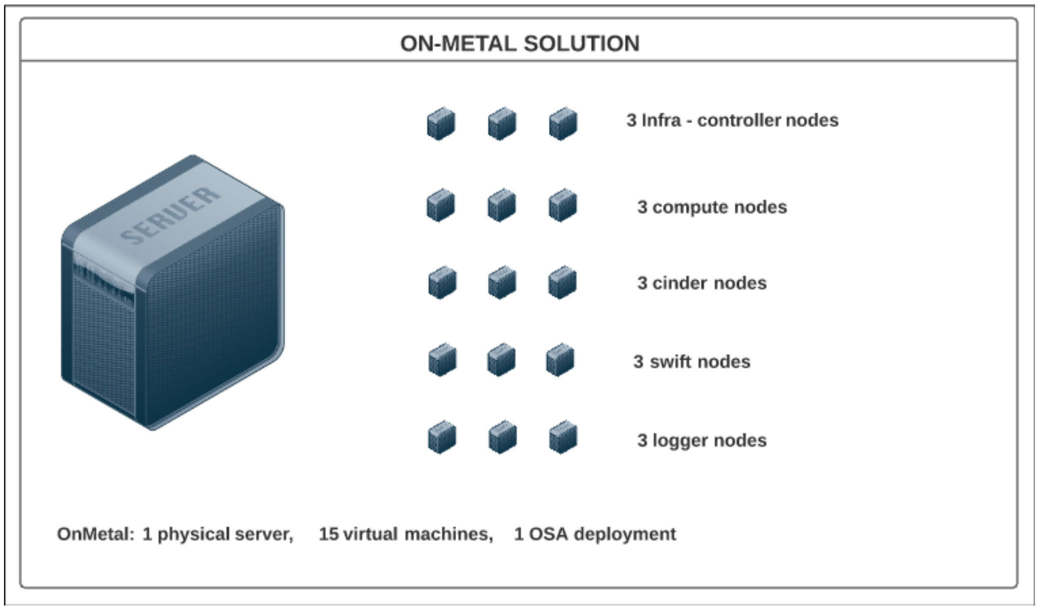
This document focuses on:

- Overall testing strategy
- Levels of testing to be performed
- Entry and Exit criteria for each test level
- Supporting testing tools
- Roles and Responsibilities of supporting testing resources
- System resources
- Assumptions and risks

During the OpenStack (OS) Newton release cycle. QE team will test upgradability of OpenStack (Cinder, Swift and Nova “The projects”) and will measure API downtime.

To accomplish this goal, team will create a third party multi-node rolling upgrade CI (See maturity steps and flows below).

All-in-one onMetal means – 1 physical server, multiple vms on it, create multi-node OS deployment using those VMs


ON-METAL SOLUTION
SERVER
3 Infra - controller nodes
3 compute nodes
3 cinder nodes
3 swift nodes
3 logger nodes
OnMetal: 1 physical server, 15 virtual machines, 1 OSA deployment

CI maturity stages:

1. CI all-in-one onMetal – Liberty to Mitaka simple upgrade (not rolling)

   Goal: Have baseline to prove that a simple upgrade flow within the CI works and can be tested.

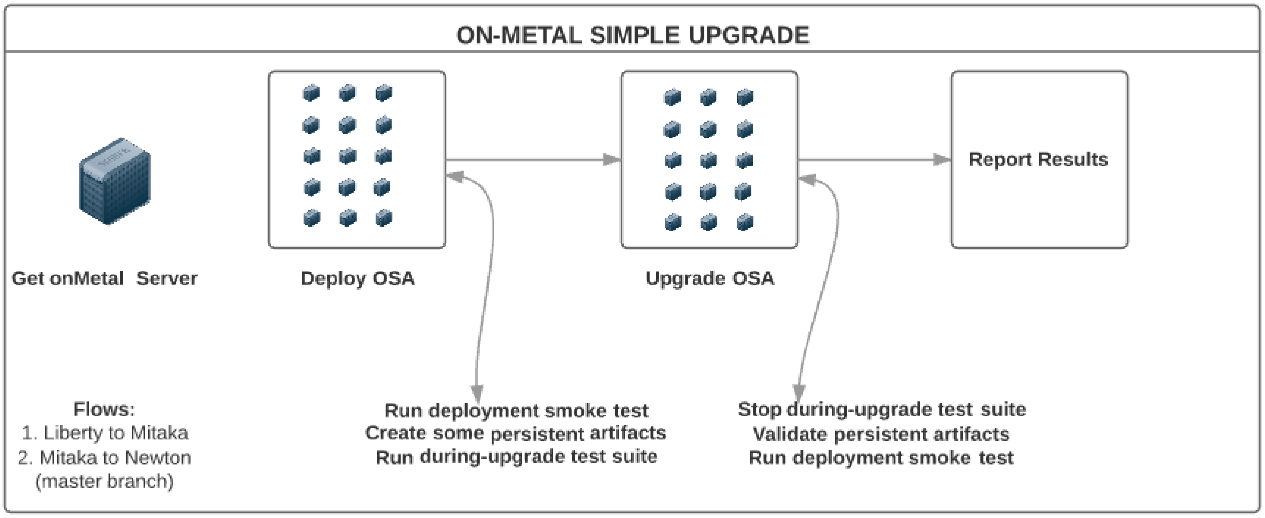
   This CI should be capable of performing pre-upgrade and post-upgrade testing.

   a. Deploy Liberty release on an all-in-one onMetal.
   b. Run smoke test (Tempest)
   c. Create persistent objects at Liberty
      i. VM
      ii. Object container
      iii. Upload object
      iv. Create and attach volume to a second VM.
   d. Upgrade deployment from Liberty to latest stable release (Mitaka).
   e. Run smoke test (Tempest)
   f. Verify sanity of persistent objects at Liberty
      i. ssh to existing VM
      ii. Add new object to existing container
      iii. Download existing object
      iv. Write a file into existing volume (second VM).R4n

2. CI all-in-one onMetal – Mitaka to Master-Newton simple upgrade (not rolling):

   Goal: Baseline from latest stable release (Mitaka) to daily branch Newton (Master)

   Same flow as above (a-f) changing OS versions – Expect several issues


ON-METAL SIMPLE UPGRADE
Get onMetal Server
Deploy OSA
Upgrade OSA
Report Results
Flows:
1. Liberty to Mitaka
2. Mitaka to Newton (master branch)
Run deployment smoke test
Create some persistent artifacts
Run during-upgrade test suite
Stop during-upgrade test suite
Validate persistent artifacts
Run deployment smoke test

3. CI all-in-one onMetal – Mitaka to Master-Newton rolling upgrade.

   End goal – prove rolling upgrade, measure downtime on the control plane at different stages of the upgrade, measure time on each upgrade stage, prove stability of the environment.

   Depends on deployment team deliverables – upgrade process and stages.

   This CI should be capable of performing pre-upgrade and post-upgrade testing.

   On a daily basis, CI should:

   a. Deploy Mitaka release on an all-in-one onMetal.
   b. Run smoke test (Tempest)
   c. Create persistent objects at Liberty
      i. VM
      ii. Object container
      iii. Upload object
      iv. Create and attach volume to a second VM.

   Project upgrade order might change

   d. Fire Nova project rolling upgrade from Mitaka to Newton
      i. Start Nova during-upgrade testing (To be run during the whole process – See Matrix HERE.
      ii. On each defined upgrade stage

         Depends on project upgrade steps and upgrade permutation matrix

         - Measure time to perform the stage
         - Run smoke test (Tempest) – ensure environment sanity, send requests for both: nodes at Mitaka and Newton releases.
   e. Finish Nova project rolling upgrade
      i. Stop Nova during-upgrade testing
      ii. Run smoke test (Tempest)
      iii. Run post-upgrade tests - TBD
   f. Fire Cinder project rolling upgrade from Mitaka to Newton
      i. Start Cinder during-upgrade testing (To be run during the whole process – See Matrix HERE.
      ii. On each defined upgrade stage

         Depends on project upgrade steps and upgrade permutation matrix

         - Measure time to perform the stage
         - Run smoke test (Tempest) – ensure environment sanity, send requests for both: nodes at Mitaka and Newton releases.
   g. Finish Cinder project rolling upgrade
      i. Stop Cinder during-upgrade testing
      ii. Run smoke test (Tempest)

iii. Run Cinder post-upgrade tests

h. Fire Swift project rolling upgrade from Mitaka to Newton

i. Start Swift during-upgrade testing (To be run during the whole process – See Matrix HERE)

ii. On each defined upgrade stage

Depends on project upgrade steps and upgrade permutation matrix

- Measure time to perform the stage
- Run smoke test (Tempest) – ensure environment sanity, send requests for both: nodes at Mitaka and Newton releases.

i. Finish Swift project rolling upgrade

i. Stop Swift during-upgrade testing

ii. Run smoke test (Tempest)

iii. Run post-upgrade tests

j. Verify sanity of persistent objects at Liberty

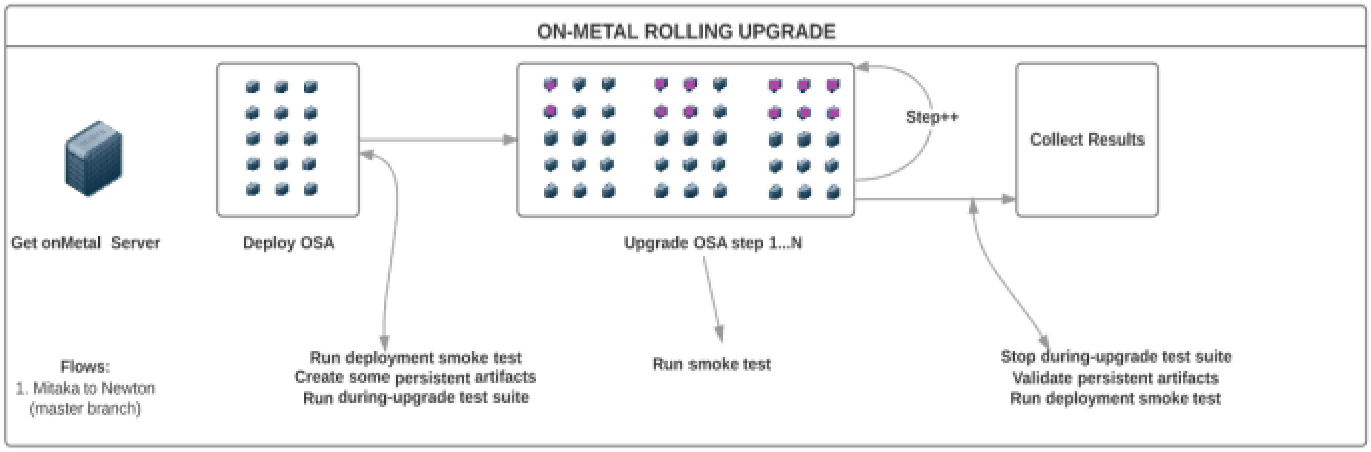
i. ssh to existing VM

ii. Add new object to existing container

iii. Download existing object

iv. Write a file into existing volume (second VM).

k. Store all results from the upgrade and testing process


ON-METAL ROLLING UPGRADE
Get onMetal Server
Deploy OSA
Upgrade OSA step 1...N
Step++
Collect Results
Flows:
1. Mitaka to Newton (master branch)
Run deployment smoke test
Create some persistent artifacts
Run during-upgrade test suite
Run smoke test
Stop during-upgrade test suite
Validate persistent artifacts
Run deployment smoke test

***DEPENDENCIES WITH OTHER TEAMS***:

Nova, Cinder and Swift teams: Provide rolling upgrade steps. Provide test scenarios for the “during-upgrade” testing and “post-upgrade” test.

Deployment team: Provide Deployment and Upgrade mechanisms (scripts, playbooks, etc).

Deployment team will assist on the stabilization of the CI flow (troubleshooting, script changes).

**IN SCOPE**

QA team to provide: CI infrastructure, configuration and workflows for all maturity CI phases. Integration of deployment and upgrade mechanisms into the CI. Integration of automated test scenarios into the CI. Collect metrics and test results. Normalization of results into elastic-search. Presentation of results via Kibana reporter.

For the first two CI maturity levels OpenStackAnsible will be the underlying deployment mechanism. Last maturity level technology is to be confirmed with deployment team TBD.

Additional details on Trello Epic Cards: https://trello.com/c/7bwNwAQr

**TEST CASE MATRIX - TBD**

## 1.2 OUT OF SCOPE

Hardware provisioning

Any special or custom OpenStack configuration

Manual test cases or scenarios

New features availability (TBD)

Deprecated features (TBD)

## 1.3 BACKGROUND

Rolling upgrade consists of upgrading progressively the servers of a distributed system to reduce service downtime. Upgrading a subset of servers requires a well-engineered cluster membership protocol to maintain, in the meantime, the availability of the system state.

Rolling upgrades imply that during some interval of time there will be services or components of a service running and interacting at different code versions in the same cloud. It puts multiple constraints onto the software.

- older services should be able to talk with newer services
- older services should not require the database to have older schema (otherwise newer services that require the newer schema would not work).

Testing rolling upgrades may include several combinations, permutations, scenarios and areas of focus. Hence a priority or risk matrix is good way to select which scenarios and test cases will be executed at each upgrade stage.

Additional test cases and implementation details will be provided by each of the teams.

# 2 TEST STRATEGY

This section addresses test level selection, characteristics and testing tools.

## 2.1 SOFTWARE DEVELOPMENT LIFECYCLE MODEL

Waterfall lifecycle – Release is under development; upgrade testing will come as stable versions become available.
Tracking of QA activities will be done with scrum methodology, having 2 week sprints, daily standups, backlog grooming, and sprint planning's.

## 2.2 TEST COVERAGE STRATEGY

| Coverage Strategy | Choose One (x) |
|---|---|
| 100% Feature Coverage | |
| Testing Risk Based Analysis | X |

## 2.3 TEST LEVEL COVERAGE

This section contains specific information relating to the selection of the test levels. Refer to the testing guidelines document for the objective and detailed description of each test level.

| Test Level | Applicable? | Rationale for omitting test level |
|---|---|---|
| Unit Testing (UT) | ☐ *Yes* ☒ *No* | Taken care by the projects |
| Component Integration Testing (CIT) | ☐ *Yes* ☒ *No* | *Taken care by the projects* |
| System Testing (ST) | ☒ *Yes* ☐ *No* | *Rolling upgrade will be treated per project as individual systems.* *No special configurations will be tested but just a single systematic approach for all.* |
| System Integration Testing (SIT) | ☒ *Yes* ☐ *No* | *Project upgrade (CI maturity 1 and 2) will be treated as a complete system (system of systems) instead of testing each project it will test them all as a single entity.* *No special configurations will be tested but just a single systematic approach for all* |
| User Acceptance Testing (UAT) | ☐ *Yes* ☒ *No* | |

## 2.4 TEST LEVEL CHARACTERISTICS

| Test Level | Owner | Entry Criteria | Exit Criteria |
|---|---|---|---|
| ST | OSIC QE Team<br><br>And<br><br>OSIC each Team Focal Point | • *Projects have unit testing, and system testing passing with all the versions involved.*<br>• *The projects are in compliance with the OpenStack rolling upgrades guidelines <u>HERE</u>*<br>• *System test environment is established (CI)*<br>• *Adequate Test data is available (Test Cases, DBs, etc)* | • 100% of planned test specifications (test cases/scripts/scenarios) for system test level must be executed and/or dispositioned with an agreement of the testing stakeholders.<br>• *Defects were documented and reported in launchpad*<br>• *All severity 1 (critical) and 2 (major) defects are triaged.* |

| | | | |
|---|---|---|---|
| | | • *Completed and reviewed test cases / test scripts*<br>• *All scenarios to be tested are identified, and automated.*<br>• *Test scenarios are included on the CI* | |
| SIT | OSIC QE Team<br><br>And<br><br>OSIC each Team Focal Point | • *"System of systems" test environment is established (CI)*<br>• *All scenarios to be tested are identified, and automated.*<br>• *Test scenarios are included on the CI* | • *All items in scope were tested*<br>• *All test cases (100%) are executed: failed cases have a resolution.*<br>• *Defects were documented and reported in launchpad*<br>• *All severity 1 (critical) and 2 (major) defects are triaged.* |

## 2.5 TEST SCOPE

| Test Level | In Scope | Out of Scope |
|---|---|---|
| ST | CI maturity level 3 – Automated rolling upgrade – Per project using different upgrade stages and test matrix.<br>Control Plane<br>Before, during and after upgrade testing: Functional, Sanity and Persistent testing. | See out of scope section |
| SIT | CI maturity level 1 y 2 – Automated Upgrade Testing.<br>Before and after upgrade testing.<br>Basic during upgrade testing<br>Validation of resources from release A when system is upgraded to release B. | See out of scope section |

## 2.6 PRIORITIZATION

This section describes the methodology that will be used prioritize test execution and bugs.

### 2.6.1 PRIORITIZATION FOR TEST EXECUTION

*CI will go through the mentioned maturity levels mentioned on Section 1.*
*All exiting automated test cases and scenarios will be run.*
*Non-automated scenarios will be prioritized based on a test case risk assessment. The test case risk will be a combination of likelihood of failure and impact if it fails. All test cases will receive an overall score that will be grouped into a high, medium, or low category. These categories will be used to determine the order that test cases will be automated and then executed.*
*As CI matures more and more Test scenarios will be automated and included into the execution. The order of test case execution will be based on each project upgrade stages (if any) and in the order in which each project gets upgraded.*

*Score will be done using:*

***Impact***
*1 = None – No noticeable impact to features*
*2 = Little – Low impact to features*
*3 = Moderate – Medium impact to features*
*4 = Severe – High impact to features*
*5 = Extreme – Critical impact to features*

***Likelihood of feature failure***
*1 = Somewhat Likely – Liittle chance that the feature will fail*
*2 = Likely – Feature will probably fail, but not certain*
*3 = Very Likely – Very high probability that the feature will fail*

***Overall score is the product of the impact value times the likelihood value***
*High = 9-15*
*Medium = 5-8*
*Low = 1-4*

### 2.6.2 PRIORITIZATION FOR BUGS

*Bug priority will be suggested and documented on Launchpad, following OpenStack community guidelines http://docs.openstack.org/contributor-guide/doc-bugs.html#doc-bug-triaging-guidelines*

*Assistance of each of the project will be required for bugs that:*
- *Causes all upgrade, CI or testing activities to be halted.*
- *Severely affects the functionality.*

## 2.7 TESTING TOOLS

| Tool(s) | Description |
|---|---|
| *Jenkins 2.0* | *Jenkins pipelines will be used to automate the complete flow including OS deployment, upgrade and testing activities.* |
| *Tempest* | *OS integration test suite – To be used for the sanity of the environment and potentially persistent testing during the upgrade* |
| *Python Scripts* | *Additional automated test suites and scripts.* |
| *Groovy Scripts* | *Language used by Jenkins pipelines* |
| *OSA* | *OpenStack-Ansible OS deployer to perform environment deployment and upgrade.* *Uses Ansible tool.* |
| *Ironic* | *OpenStack baremetal project to provision operating system into physical nodes.* *Potentially replaced by cobbler if going on virtual instead of physical hardware.* |
| *Elastic-Search* | *Non SQL DB to store results information* |
| *Kibana* | *Reporter server to display results* |

# 3 ROLES AND RESPONSIBILITIES

| Role/ Group | Responsibilities | Name(s) |
|---|---|---|
| *QA Test Lead* | *Provides testing management oversight. Responsibilities:*<br>• *provide technical direction*<br>• *acquire appropriate resources*<br>• *provide management reporting* | Daryl Walleck |
| *Product Owner* | *Represents customer's interest and represents the product to the outside world (Customer).*<br>*Responsibilities:*<br>• *Responsible for market, business case, and competitive analysis*<br>• *Responsible for long and short term product vision*<br>• *Prioritizes features for releases based upon expected ROI*<br>• *Writes Acceptance Criteria*<br>• *Writes user stories*<br>• *Makes trade-off decisions between scope and schedule* | Kenny<br>Krish<br>Sonu |
| *QA Team* | *Responsible for qualification of product.*<br>*Responsibilities:*<br>• *decide on the scope of the testing in agreement with Project Manager*<br>• *Configure CI infrastructure*<br>• *Create CI flows*<br>• *Integrate test cases into the CI*<br>• *Automates additional test cases and scenarios*<br>• *log results*<br>• *open and verify bugs*<br>• *help troubleshooting error* | OSIC QA |

| | | |
|---|---|---|
| *Deployment Team* | *Responsible for creation of OpenStack deployment and rolling upgrade automated mechanism.*<br>*Responsibilities:*<br>• *Get and configure infrastructure(baremetal 22 nodes)*<br>• *Provide OpenStack architecture*<br>• *Create automated way to deploy OpenStack.*<br>• *Create automated way to rolling upgrade the selected projects.*<br>• *Help with integration and stabilization of the CI flow* | OSIC Deployment |
| *Nova, Cinder, and swift focal points* | *Responsible to provide the rolling upgrade information and test plan for their own projects:*<br>• *Provide deployment team with the steps and knowledge about "How to live upgrade the project" - Documentation, links, release notes, locate any other relevant information.*<br>• *Identify the scenarios that should run during and after the rolling upgrade (either automated or manual)*<br>• *If needed help with the automation of the manual identified test scenarios* | Shashi<br>Pushkar<br>Shiva<br>Szimon |

# 4 TEST ENVIRONMENT AND RESOURCES

The following tables are used to identify the system resources (hardware, software etc. required for the test environment.

## 4.1 SYSTEM TEST ENVIRONMENT

### 4.1.1 HARDWARE

| Component | Description | Server name (Optional) | Network Information (Optional) | Notes |
|---|---|---|---|---|
| Bare metal Server | Rackspace onMetal I/O V2 | Variable | Variable | Need credential to spin it up |
| Jenkins master | Principal Jenkins component | Cloud1 | 172.99.106.115 | |
| Jenkins agents | Jenkins slaves – perform actual work | Variable – automated | Private-net | |

### 4.1.2 SOFTWARE

| Environment | Component | Product/Application | Versions |
|---|---|---|---|
| *System under test* | *Platform* | *OpenStack* | *Liberty*<br>*Mitaka*<br>*Newton (Master)* |
| *CI* | *Web Server* | *Jenkins* | *2.0* |
| CI | Programming languages | Ansible<br>Python<br>Groovy<br>Shell | |

## 4.2 TEST DATA ACQUISITION

The following table is used to identify the approach for acquiring and securing the test data to be used for each test level.

| Source of Test Data | Extraction approach | Type of test data (input or pre-existing) | Security controls |
|---|---|---|---|
| *TBD – Might be OSIC Cloud1 DBs but not confirmed* | *TBD* | *input* | *TBD* |
| | | | |

# 5 TEST ASSUMPTIONS AND RISKS

## 5.1 ASSUMPTIONS

This section lists assumptions that are specific to the test planning.

| # | Assumption |
|---|---|
| **1** | *A stable mechanism to deploy OpenStack (all-in-one onMetal or any other) is owned and provided by the deployment team.* |
| **2** | *A stable mechanism to upgrade OpenStack is owned and provided by the deployment team.* |
| **3** | *A stable mechanism to rolling upgrade OpenStack is owned and provided by the* |

| | |
|---|---|
| | *deployment team.* |
| **4** | Deployment team will assist on the stabilization of the CI flow |
| **5** | Deployment team will help troubleshooting and find root cause analysis of issues. |
| **6** | Issues won't be fixed unless caused by the deployment tools (deployment team) or the CI (QA team) |
| **7** | Nova, Cinder and Swift projects met the OpenStack community requirements to perform upgrades and rolling upgrades efficiently |
| **8** | Upgrade steps/ stages are provided by the projects and agreed for implementation with the deployment team. |
| **9** | Test Plans are provided by the projects assisted by QA team |
| | |

## 5.2 RISKS

The following risks to the testing plan have been identified and the supporting contingency plans included to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is the milestone or event that would cause the risk to become an issue to be addressed.

| # | Risk | Impact | Trigger | Mitigation/ Contingency Plan |
|---|---|---|---|---|
| **1** | *Fail to deliver rolling upgrade mechanism.* | *Unable to complete the CI flow* | | *Have CI with simple upgrade (Maturity level 1 and 2)* |
| **2** | Unstable OpenStack upgrade and rolling upgrade | *Unable to complete the CI flow*<br><br>*Delay testing*<br><br>*Untrusty results* | *Unstable branches*<br><br>*Bugs on the projects* | *TBD* |
| **3** | *CI stability* | *Delay testing*<br><br>*Untrusty results* | *Unstable branches*<br><br>*Issues on the deployment tools*<br><br>*Issues on the CI* | *TBD* |
| **4** | Selected scenarios not reflecting critical areas | Untrusty results | Blind spots<br><br>Lack of knowledge | Working with technical leaders to validate the scenarios |
| **5** | | | | |

# 6 TEST SCHEDULE

| Testing Level | Test Activity | Timeframe |
|---|---|---|
| *System Integration Test* | *Sprint 1 – Id test scenarios, test cases and upgrade procedure.* | *Sep 2* |
| | Sprint 1 – Automation of an all-In-one OnMetal maturity level 1 – Liberty to Mitaka | Sep 2 |
| | Sprint 2 – Automation of test cases and integration into Jenkins CI flows | Sep 16 |
| | Sprint 2 – Automation of an all-In-one OnMetal maturity level 1 – Mitaka to Newton Master branch | Sep 16 |
| | Sprint X – One server with multi node VMs – rolling upgrade<br>DEPENDS ON DEPLOYMENT PLAYBOOKS (specifically rolling upgrade steps) | TBD |
| | Sprint X – Multi node bare metal – rolling upgrade<br>DEPENDS ON DEPLOYMENT PLAYBOOKS | TBD |

# 7 TEST REPORTING

Following measurements will be collected and reported.

| # | Metrics | Measurement Data | Frequency | Responsible | Reported To |
|---|---|---|---|---|---|
| | API downtime | Time End to End - Trending | Daily | CI | |
| | Playbooks elapsed times | Time End to End - Trending | | | |
| | Test suites failure ratio | Trending | | | |

# 8 REFERENCES

*Main repository https://github.com/osic/osic-upgrade-test*
*http://docs.openstack.org/contributor-guide/doc-bugs.html*
*http://docs.openstack.org/index.html#install-guides*
*http://docs.openstack.org/developer/grenade/readme.html#basic-flow*
*http://www.danplanet.com/blog/2015/06/26/upgrading-nova-to-kilo-with-minimal-downtime/*
*http://docs.openstack.org/ops-guide/ops-upgrades.html*
*http://docs.openstack.org/developer/neutron/devref/upgrade.html*
*https://governance.openstack.org/reference/tags/assert_follows-standard-deprecation.html*
*https://governance.openstack.org/reference/tags/assert_supports-rolling-upgrade.html*

## 8.1 EXTERNAL REFERENCES

This section lists references to the relevant policies or laws that give rise to the need for this plan.
NA

# 9 GLOSSARY

| Item | Description |
|---|---|
| Black box testing | Focus is on the external attributes and behavior of the software. Such testing examines the software from the user perspective. UAT is the classical example of this type of testing |
| Bug | A bug is a flaw, error or omission identified during the testing process. Bugs are typically classified by level of severity ranging from non-critical to “show stopper” |
| Negative Testing (destructive) | Testing attempts to prove that the software can be broken using invalid or erroneous input conditions. Both defined and undefined error conditions should be generated. |
| Positive Testing | Testing attempts to prove that the software satisfies the requirements |
| S&P testing | Stress and Performance testing |
| Test Case | A test case is a specific test designed to verify a particular condition or requirement. It identifies input data with predicted results and describes the testing objective. |
| Test Script | Provide the step by step procedures comprising the actions to be taken and the verification of the results |
| White-box testing | It tests software with knowledge of internal data structures, logical flow at the source code level. Unit testing is the classical example of this type of testing. |