Test Plan for Manila Fuel plugin

Table of contents

Introduction

1.1.Purpose

This document describes Master Test Plan for Manila Fuel Plugin. The scope of this plan defines the following objectives:

- describe testing activities;
- outline testing approach, test types, test cycles that will be used;
- test mission;
- deliverables;

1.2.Intended Audience

This document is intended for Manila project team staff (QA and Dev engineers and managers) all other persons who are interested in testing results.

Governing Evaluation Mission

There is the Manila project that provides a shared file system service for OpenStack. It's the new OpenStack project that provides coordinated access to shared or distributed file systems. The Manila has multiple storage backends (to support vendor or file system specific nuances/capabilities) that could work simultaneously on one environment. The Fuel Manila plugin adds utilization the corresponding OpenStack service in an Fuel environment.

2.1. Evaluation Test Mission

- Lab environment deployment.
- Deploy MOS with developed plugin installed.
- Create and run specific tests for plugin/deployment.

- Documentation

2.2. Test items

- Manila UI;
- Fuel CLI;
- Fuel API;
- Fuel UI;
- MOS UI;
- MOS API.

Test Approach

The project test approach consists of BVT, Acceptance Integration/System and Regression test levels.

3.1 Criteria for test process starting

Before test process can be started it is needed to make some preparation actions - to execute important preconditions.The following steps must be executed successfully for starting test phase:

- all project requirements are reviewed and confirmed;
- implementation of features has finished (a new build is ready for testing);
- implementation code is stored in GIT;
- bvt-tests are executed successfully (100% success);
- test environment is prepared with correct configuration;
- test environment contains the last delivered build for testing;
- test plan is ready and confirmed internally;
- implementation of manual tests and necessary autotests has finished.

3.2. Suspension Criteria

Testing of a particular feature is suspended if there is a blocking issue which prevents tests execution. Blocking issue can be one of the following:

- Feature has a blocking defect, which prevents further usage of this feature and there is no workaround available;
- CI test automation scripts failure.

3.3. Feature Testing Exit Criteria

Testing of a feature can be finished when:

- all planned tests (prepared before) for the feature are executed;
- no defects are found during this run;
- all planned tests for the feature are executed;
- defects found during this run are verified or confirmed to be acceptable (known issues);

The time for testing of that feature according to the project plan has run out and Project Manager confirms that no changes to the schedule are possible.

Deliverables

4.1 List of deliverables

Project testing activities are to be resulted in the following reporting documents:

- Test plan;
- Test run report from TestRail;

4.2. Acceptance criteria

90% of tests cases should be with status - passed. Critical and high issues are fixed. Manual tests should be executed and pass:

Steps:

1. Deploy cluster with Manila plugin enabled.
2. TBD

Test Cycle Structure

An ordinary test cycle for each iteration consists of the following steps:

- Smoke testing of each build ready for testing;
- Verification testing of each build ready for testing;
- Regression testing cycles in the end of iteration;
- Creation of a new test case for covering of a new found bug (if such test does not exist).

5.1.1 Smoke Testing

Smoke testing is intended to check a correct work of a system after new build delivery. Smoke tests allow to be sure that all main system functions/features work correctly according to customer requirements.

5.1.2 Verification testing

Verification testing includes functional testing covering the following: new functionality (implemented in the current build); critical and major defect fixes (introduced in the current build). Some iteration test cycles also include non-functional testing types described in Overview of Planned Tests.

5.1.3 Regression testing

Regression testing includes execution of a set of test cases for features implemented before current iteration to ensure that following modifications of the system haven't introduced or uncovered software defects. It also includes verification of minor defect fixes introduced in the current iteration.

5.1.4 Bug coverage by new test case

Bug detection starts after all manual and automated tests are prepared and test process initiated. Ideally, each bug must be clearly documented and covered by test case. If a bug without a test coverage was found it must be clearly documented and covered by custom test case to prevent occurrence of this bug in future deployments/releases etc. All custom manual test cases suppose to be added into TestRail and automated tests suppose to be pushed to Git/Gerrit repo.

5.2 Metrics

Test case metrics are aimed to estimate a quality of bug fixing;detect not executed tests and schedule their execution.

- passed / Failed test cases - this metric shows results of test cases execution, especially, a ratio between test cases passed successfully and failed ones. Such statistics must be gathered after each delivered build test. This will help to identify a progress in successful bugs fixing. Ideally, a count of failed test cases should aim to a zero.
- Not Run test cases - this metric shows a count of test cases which should be run within a current test phase (have not run yet). Having such statistics, there is an opportunity to detect and analyze a scope of not run test cases, causes of their non execution and planning of their further execution (detect time frames, responsible QA)

Smoke/BVT tests

Instalation test

ID

manila_install

Description

Check install/uninstall Manila Plugin functionality.

Complexity

core

Steps

1. Upload plugins to the master node and install.
2. Create a new environment
3. Enable Manila plugin for new environment.
4. Attempt to remove enabled plugin.
5. Disable plugin
6. Remove Plugin Manila

Expected results

All steps must be completed successfully, without any errors.

Smoke test

ID

manila_smoke

Description

Deploy a cluster with Manila Plugin.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with "Neutron with VLAN segmentation" as a network configuration.
3. Enable plugin.
4. **Configure environment:**
    - Add a node with Controller + role.
    - Add a node with Compute + Cinder + Manila-share + Manila-data roles.
    - Add a node with Compute + Cinder roles.
5. Run network check
6. Deploy cluster with plugin.
7. Verify Manila service basic functionality (share add/mount)

Expected results

All steps must be completed successfully, without any errors.

BVT test

ID

manila_bvt

Description

BVT test for manila plugin. Deploy cluster with 3 controller and install Manila plugin.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with "Neutron with tunneling segmentation" as a network configuration.
3. Enable plugin.
4. **Configure environment:**
    - Add a node with Controller + Ceph-OSD roles.
    - Add a node with Controller + Ceph-OSD roles.
    - Add a node with Controller + Ceph-OSD roles.
    - Add a node with Compute + Ceph-OSD roles.
    - Add a node with Ceph-osd + Manila-share + Manila-data roles.
5. Deploy cluster with plugin.
6. Run OSTF.
7. Verify Manila service basic functionality (share add/mount)

Expected results

All steps must be completed successfully, without any errors.

Integration tests

Multiple manila share nodes test

ID

manila_share_ha

Description

Deploy a cluster with at least two nodes with Manila-share role.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with at least 2 Manila-share and 1 Manila-data roles.
3. Deploy cluster with plugin.
4. Run OSTF.
5. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Multiple manila data nodes test

ID

manila_data_ha

Description

Deploy a cluster with two nodes with Manila-data role.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with at least 2 Manila-data and 1 Manila-share roles.
3. Deploy cluster with plugin.
4. Run OSTF.
5. Verify Manila service basic functionality (share create/mount)

Expected results

All steps must be completed successfully, without any errors.

Both Cinder and Ceph test

ID

manila_both_cinder_ceph

Description

Deploy a cluster using Ceph as a backend for block storage and cinder for other (image, object and ephemeral).

Complexity

core

Steps

1. Upload plugins and install.
2. Set Ceph as a backend for block storage.
3. Create environment with at least 1 Manila-data 1 Manila-share 1 Cinder and 3 Ceph-OSD.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Ceilometer enabled test

ID

manila_with_ceilometer

Description

Deploy a cluster with additional component Ceilometer.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with enabled component Ceilometer.
3. Configure nodes with at least 3 Mongo-DB 1 Manila-data and 1 Manila-share roles.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Murano enabled test

ID

manila_with_murano

Description

Deploy a cluster with additional component Murano.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with enabled component Murano.
3. Configure nodes with at least 1 Manila-data and 1 Manila-share.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Sahara enabled test

ID

manila_with_sahara

Description

Deploy a cluster with additional component Sahara.

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with enabled component Sahara.
3. Configure nodes: at least 1 Manila-data 1 Manila-share.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Enabled driver and redeploy

ID

manila_enable_after_deploy

Description

Verify redeploy of cluster with enabled driver

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with disabled manila plugin.
3. Deploy cluster.
4. Run OSTF.
5. Enable and configure plugin for deployed environment.
6. Configure nodes: at least 1 Manila-data 1 Manila-share.
7. Re-deploy cluster.
8. Verify Manila service basic functionality (share create/mount).
9. Run OSTF.

Expected results

All steps must be completed successfully, without any errors.

Functional tests

Controller delete/add test

ID

manila_del_add_controllers

Description

Verify that node with controller role can be deleted and added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 3 nodes with controller role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Delete a Controller node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).
11. Add a Controller node and deploy changes.
12. Deploy cluster with plugin.
13. Run OSTF.
14. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Compute delete/add test

ID

manila_del_add_compute

Description

Verify that node with compute role can be deleted and added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 2 nodes with compute role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Delete a compute node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).
11. Add a compute node and deploy changes.
12. Deploy cluster with plugin.
13. Run OSTF.
14. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Cinder delete/add test

ID

manila_del_add_cinder

Description

Verify that node with cinder role can be deleted and added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 2 nodes with cinder role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Delete a cinder node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).
11. Add a cinder node and deploy changes.
12. Deploy cluster with plugin.
13. Run OSTF.
14. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Manila-share delete/add test

ID

manila_del_add_share

Description

Verify that node with manila-share role can be deleted and added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 2 nodes with manila-share role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Delete a manila-share node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).
11. Add a manila-share node and deploy changes.
12. Deploy cluster with plugin.
13. Run OSTF.
14. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Manila-data delete/add test

ID

manila_del_add_data

Description

Verify that node with manila-data role can be deleted and added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 2 nodes with manila-data role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Delete a manila-data node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).
11. Add a manila-data node and deploy changes.
12. Deploy cluster with plugin.
13. Run OSTF.
14. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Ceph add test

ID

manila_add_ceph

Description

Verify that node with ceph-osd role can added after deploying.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment.
3. Add at least 3 nodes with ceph-osd role.
4. Deploy cluster with plugin.
5. Run OSTF.
6. Verify Manila service basic functionality (share create/mount).
7. Add another ceph-osd node and deploy changes.
8. Deploy cluster with plugin.
9. Run OSTF.
10. Verify Manila service basic functionality (share create/mount).

Expected results

All steps must be completed successfully, without any errors.

Failover

Check cluster and manila functionality after controller shutdown/reboot

ID

manila_shut_reb_controller

Description

Verify that manila-service works after controllers shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.
2. Create a new ha environment.
3. Add at least 3 nodes controller, 1 with manila-data and 1 with manila-share and 1 compute node roles.
4. Deploy cluster.
5. Shutdown primary controller node.
6. Run OSTF.
7. Verify Manila service basic functionality (share add/mount).
8. Reboot controller node which becomes primary.
9. Run OSTF.
10. Verify Manila service basic functionality (share add/mount).

Expected result

All steps must be completed successfully, without any errors.

Check cluster and manila functionality after compute shutdown/reboot

ID

manila_shut_reb_compute

Description

Verify that manila-service works after compute shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.
2. Create a new environment.
3. Add at least 2 nodes compute role, 1 with manila-data and 1 with manila-share roles.
4. Deploy cluster.
5. Reboot first and shutdown second compute node.
6. Run OSTF.
7. Verify Manila service basic functionality (share add/mount).
8. Bring up compute second node, and shutdown first compute node.
9. Verify Manila service basic functionality (share add/mount).
10. Run OSTF.

Expected result

All steps must be completed successfully, without any errors.

Check cluster and manila functionality after manila-share shutdown/reboot

ID

manila_shut_reb_share

Description

Verify that manila-service works after manila-share shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.
2. Create a new environment.
3. Add at least 2 nodes manila-share role, 1 with manila-data role.
4. Deploy cluster.
5. Shutdown first manila-share and reboot second manila-share node.
6. Run OSTF.
7. Verify Manila service basic functionality (share add/mount).
8. Bring up first manila-share node, and shutdown second manila-share node.
9. Verify Manila service basic functionality (share add/mount).
10. Run OSTF.

Expected result

All steps must be completed successfully, without any errors.

Check cluster and manila functionality after manila-data shutdown/reboot

ID

manila_shut_reb_data

Description

Verify that manila-service works after manila-data shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.
2. Create a new environment.
3. Add at least 2 nodes manila-data role, 1 with manila-share role.
4. Deploy cluster.
5. Shutdown first manila-data and reboot second manila-data node.
6. Run OSTF.
7. Verify Manila service basic functionality (share add/mount).
8. Bring up first manila-data node, and shutdown second manila-data node.
9. Verify Manila service basic functionality (share add/mount).
10. Run OSTF.

Expected result

All steps must be completed successfully, without any errors.

Check cluster and manila functionality after cinder shutdown/reboot

ID

manila_shut_reb_cinder

Description

Verify that manila-service works after cinder shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.
2. Create a new environment.
3. Add at least 2 nodes cinder roles, 1 with manila-share and 1 with manila-share role.
4. Deploy cluster.
5. Run OSTF.
6. Shutdown first and reboot second cinder node.
7. Verify Manila service basic functionality (share add/mount).
8. Run OSTF.
9. Bring up first cinder node, and shutdown second cinder node.
10. Verify Manila service basic functionality (share add/mount).
11. Run OSTF.

Expected result

All steps must be completed successfully, without any errors.

Check cluster and manila functionality after ceph-osd reboot

ID

manila_shut_reb_ceph-osd

Description

Verify that manila-service works after ceph-osd shutdown/reboot.

Complexity

core

Steps

1. Install Manila plugin on master node.

2. Create a new environment.
3. Add at least 3 nodes ceph-osd roles, 1 with manila-share and 1 with manila-share role.
4. Deploy cluster.
5. Shutdown first ceph-osd node.
6. Verify Manila service basic functionality (share add/mount).
7. Run OSTF.
8. Bring up first ceph-osd node and reboot second
9. Verify Manila service basic functionality (share add/mount).
10. Run OSTF.

Expected result

All steps must be completed successfully, without any errors.

System tests

NFS share with generic driver

ID

manila_nfs_generic

Description

Verify generic driver functionality with NFS share type.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment with manila plugin and configured generic driver.
3. Add at least 1 manila-share 1 manila-data 1 compute nodes role.
4. Deploy cluster with plugin.
5. Create configure and mount NFS share.
6. Verify I/O to share according to configured ACL(IP).

Expected results

All steps must be completed successfully, without any errors.

CIFS share with generic driver

ID

manila_cifs_generic

Description

Verify generic driver functionality with CIFS share type.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment with manila plugin and configured generic driver.
3. Add at least 1 manila-share 1 manila-data 1 compute nodes role.
4. Deploy cluster with plugin.
5. Create configure and mount CIFS share.
6. Verify I/O to share according to configured ACL(IP).

Expected results

All steps must be completed successfully, without any errors.

NFS share with netapp driver

ID

manila_nfs_netapp

Description

Verify netapp driver functionality with NFS share type.

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment with manila plugin and configured netapp driver.
3. Add at least 1 manila-share 1 manila-data 1 compute nodes role.
4. Deploy cluster with plugin.
5. Create configure and mount NFS share.
6. Verify I/O to share according to configured ACL(IP).

Expected results

All steps must be completed successfully, without any errors.

ID

manila_cifs_netapp

Description

Verify netapp driver functionality with CIFS and NFS share types

Complexity

core

Steps

1. Upload plugins and install.
2. Create an environment with manila plugin and configured netapp driver.
3. Add at least 1 manila-share 1 manila-data 1 compute nodes role.
4. Deploy cluster with plugin.
5. Create configure and mount CIFS share.
6. Verify I/O to NFS share according to configured ACL(IP).

Expected results

All steps must be completed successfully, without any errors.

NFS CIFS share with both drivers

ID

manila_nfs_cifs_both

Description

Verify manila functionaity with and both generic and netapp drivers enabled

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with both Generic and NetApp drivers enabled.
3. Add at least 1 manila-share 1 manila-data 1 compute nodes role.
4. Deploy cluster with plugin.
5. Create configure and mount NFS share using generic driver.
6. Verify I/O to NFS share according to configured ACL(IP).
7. Create configure and mount CIFS using NetApp driver.
8. Verify I/O to NFS share according to configured ACL(IP).

Expected results

All steps must be completed successfully, without any errors.

Instance live migration

ID

manila_live_migration

Description

Verify manila functionaity after instance live migration

Complexity

core

Steps

1. Upload plugins and install.
2. Create environment with both Generic.
3. Add at least 2 nodes with compute role 1 node with manila-share and 1 with manila-data role
4. Deploy cluster with plugin.
5. Create configure and mount NFS share using generic driver.
6. Verify I/O to NFS share.
7. Execute live migration for instance to other compute.
8. Verify I/O to NFS share after migration.

Expected results

All steps must be completed successfully, without any errors.

GUI tests

GUI defaults test

ID

manila_gui_defalts

Description

Check default settings for Manila Plugin via GUI.

Complexity

core

Steps

1. Upload plugins to the master node and install.
2. Create a new environment.
3. Verify default settings for Plugin: Plugin is enabled, generic driver is enabled, default values for the rest fields are present.

Expected results

All steps must be completed successfully, without any errors.

GUI field validation test

ID

manila_gui_validation

Description

Check validation messages for Manila Plugin settings via GUI.

Complexity

core

Steps

1. Upload plugins to the master node and install.
2. Create a new environment.
3. Check validation for setting Image name.
4. Check validation for setting NetApp server Hostname.
5. Check validation for setting NetApp transport type.
6. Check validation for setting NetApp server port.
7. Check validation for setting NetApp username.
8. Check validation for setting NetApp password.
9. Check validation for setting NetApp volume aggregate.
10. Check validation for setting NetApp pattern for aggregation names.
11. Check validation for setting NetApp pattern for storage port names.

Expected results

All steps must be completed successfully, without any errors.