|  |  |
| --- | --- |
|  |  |

****

**TEST PLAN**

**for**

**Plugin Name <version in 6-digit format>**

**Mirantis OpenStack <release>**

Content

Revision history 4

<Plugin name> Plugin 5

Developer’s specification 5

Limitations 5

Test strategy 5

Types of tests included 5

Types of tests not included 5

Acceptance criteria 5

Test environment and infrastructure 5

Testing tools recommended by Mirantis 6

Product compatibility matrix 7

Functional testing 7

Tempest 7

<Case 2 title> 8

System testing 8

Install plugin and deploy environment 9

Modifying env with enabled plugin (removing/adding controller nodes) 9

Modifying environment with enabled plugin (removing/adding Compute node) 11

Fuel create mirror and update (setup) of core repos 12

Uninstall of plugin in the deployed environment 13

Uninstall of plugin in the non-deployed environment 14

Upgrade/update 15

The Fuel Master node upgrade testing 15

Update the plugin to minor version in the deployed environment 16

Apply maintenance updates to deployed environment 17

Other 18

Fuel plugin interoperability testing 18

Testing with different plugin types 18

Performance testing 18

Networking performance 18

API benchmarking 18

Storage performance 18

Appendix 19

Tests included into the tools 20

OSTF (Health Checks) 20

Sanity group 20

Smoke group 20

HA group 21

Platform group 21

Cloud validation group 21

Tempest 21

Rally 21

Ceilometer 22

Cinder 22

Heat 22

Glance 22

Keystone 22

Murano 22

Nova 22

Neutron 23

Swift 23

Rally 23

Shaker 23

Storage performance 24

IOPS for block devices 24

Resiliency Tests 26

# 

# Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Revision date** | **Editor** | **Comment** |
| 0.1 | 23.01.2015 | Irina Povolotskaya  (ipovolotskaya@mirantis.com) | Created the template structure. |
| 0.2 | 29.02.2016 | Irina Povolotskaya  (ipovolotskaya@mirantis.com) | Updated the template with testing approach and tools recommended by Mirantis. |
|  |  |  |  |

# <Plugin name> Plugin

*Provide common information about the plugin (what functionality it provides, what component it extends). You can also add schemes with a detailed explanation of plugin architecture.*

*You can copy this information from the Plugin Guide or refer to the specific section of the Plugin Guide (the former is more preferable).*

## Developer’s specification

*Provide a link to developer’s specification (the specification itself should be formed according to* [*the template*](https://github.com/stackforge/fuel-specs/blob/master/specs/template.rst) *and put into the plugin’s repo).*

## Limitations

*Provide information about possible limitations (for example, plugin can be enabled only in a specific environment configuration).*

*You can copy this information from the Plugin Guide or refer to the specific section of the Plugin Guide (the former is more preferable).*

# Test strategy

*Provide information on types of implemented tests (that enter test cases below). Specify which tests will be automated.*

## Types of tests included

*Provide information on specific type of tests included into the current scope.*

## Types of tests not included

*Provide information on specific types of tests excluded from the scope.*

## Acceptance criteria

*Provide information on the set of basic tests that should be performed to consider that the feature has passed the testing process.*

## Test environment and infrastructure

*If all cases have one environment, put all information about hardware, software and other issues necessary for the plugin into this section and remove ‘Environment’ row from every test case . If not, then skip this section and fill in ‘Environment’ section in every case table.*

***Note: If you plugin is hardware-specific, provide instructions on accessing your lab.***

## Testing tools recommended by Mirantis

*Mirantis recommends that you use a series of testing tools to make sure your plugin meets both Mirantis and OpenStack Community testing standards.*

*For your convenience, the table also provides the specific categories of the plugin to apply these tools to:*

|  |  |  |
| --- | --- | --- |
| **Tool name** | **Comment** | **Plugin category** |
| [**OSTF**](https://github.com/openstack/fuel-ostf/tree/master/fuel_health/tests) | Mirantis OpenStack has built-in system called Health Check for a quick high-level verification of the cloud. Health Check will test the basic operations of the OpenStack API, HA validation (check MySQL and RabbitMQ clusters status) and other critical functions.  [OSTF](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) (aka Health Checks) are run automatically on deploying your cluster with the plugin installed.  This means, OSTF is obligatory step in all test cases. | Suitable for **any** plugin. |
| [**Shaker**](http://shaker-docs.readthedocs.org/en/latest/) | Shaker is used to test network performance. It’s the distributed data-plane testing tool for OpenStack. | Suitable for Networking/SDN plugins. |
| [**Rally**](https://rally.readthedocs.org/en/latest/) | Rally is used to test API functionality and performance. Rally is a benchmarking tool developed by OpenStack community and is widely used in OpenStack ecosystem. | Suitable for **any** plugin. |
| [**Tempest**](https://github.com/Mirantis/mos-tempest-runner) | Tempest is an original test framework developed by community and aimed to cover all possible API functionality available in Openstack components or added by third-party modules/drivers/plugins. It contains hundreds of tests. | Suitable for **any** plugin. |
| [**Fio**](https://github.com/axboe/fio) | Fio is used for disk performance measurement | Suitable for storage plugins. |
| [**Wally**](https://github.com/Mirantis/disk_perf_test_tool) | Wally enables testing framework for measuring block storage devices performance. | Suitable for storage plugins. |

***Please be sure to see the list of all tests run within every tool in*** [***the corresponding section***](#h.mqcui0a8862n)***.***

## Product compatibility matrix

*Provide information on interoperability for the plugin with specific Mirantis OpenStack release.*

*If applicable, please specify other plugins versions to test with. For more information on inter-compatibility testing, see* [*the corresponding section*](#h.gkvyu1kc3exi)*.*

|  |  |
| --- | --- |
| **Issue** | **Version** |
| Mirantis OpenStack |  |
| Plugin 1 name |  |

# Functional testing

*Functional testing is* ***obligatory*** *for your plugin and highly depends on the plugin's use case.*

## Tempest

*Case title should be short. Provide description in the table below.*

|  |  |
| --- | --- |
| Test Case ID | *tempest\_run* |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Create an environment](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#create-a-new-openstack-environment) with enabled plugin in the Fuel Web UI. 5. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 6. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. Install Tempest suite at the Fuel Master node according to the instructions (found in [README.md](https://github.com/Mirantis/mos-tempest-runner) file). |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *All Tempest tests are passed successfully except for those expected to fail (the latter are listed* [*here*](https://github.com/Mirantis/mos-tempest-runner/tree/master/shouldfail)*).* |

## <Case 2 title>

|  |  |
| --- | --- |
| Test Case ID | *Provide unique id of test case.* |
| Description | *Provide description for the case (what will be checked with this specific test case).* |
| Prerequisites | *Provide prerequisite steps for setting up the required environment (configuring software, hardware)* |
| Steps | *Provide step-by-step instructions in enumerated list:*   1. ... 2. ... |
| Expected Result | *Describe what result is expected.* |

# System testing

* *This section provides information on the test cases that are* ***obligatory*** *for your plugin.*
  + *These cases serve to make sure that your plugin does not break the Mirantis OpenStack environments.*
* *Please note that Mirantis recommends that you used the* ***HA environment configuration*** *for testing purposes:*
  + *At least three controllers must be configured to have a reliable HA environment; this is the minimum requirement for quorum-based clusters, such as Pacemaker and Galera. The controller cluster can include more than three servers to increase the level of reliability. For more information about how Fuel deploys HA, see* [*Multi-node with HA Deployment*](https://docs.mirantis.com/openstack/fuel/fuel-7.0/reference-architecture.html#multi-node-ha)*.*

*Note that system testing includes OSTF as the tool recommended by Mirantis.*

## Install plugin and deploy environment

|  |  |
| --- | --- |
| Test Case ID | install\_plugin\_deploy\_env |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Create an environment](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#create-a-new-openstack-environment) with enabled plugin in the Fuel Web UI. 5. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 6. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.* |

## Modifying env with enabled plugin (removing/adding controller nodes)

|  |  |
| --- | --- |
| Test Case ID | modify\_env\_with\_plugin\_remove\_add\_controller |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that the plugin is installed successfully using CLI with running fuel plugins --list command in the Fuel CLI. 4. [Create an environment](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#create-a-new-openstack-environment) with enabled plugin in the Fuel Web UI. 5. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 6. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 7. Enable the plugin and configure it following the instructions from the Plugin Guide. 8. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 9. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 10. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 11. Remove 1 node with Controller role (*i.e. remove the primary Controller node which should have the lowest ID, where plugin’s services are running to ensure that all plugins resources are migrated to another Controller node*). 12. [Re-deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 13. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 14. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 1 new node with Controller role. 15. [Re-deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 16. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *When adding/removing Controller node (where plugin-related services are run):*   * *all plugins resources are migrated to another Controller node* * *the environment is redeployed successfully when adding/removing Controller node.* |

## Modifying environment with enabled plugin (removing/adding Compute node)

|  |  |
| --- | --- |
| Test Case ID | modify\_env\_with\_plugin\_remove\_add\_compute |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins)   [Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details).   1. Install the plugin. 2. Ensure that the plugin is installed successfully using CLI with running fuel plugins --list command in the Fuel CLI. 3. [Create an environment](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#create-a-new-openstack-environment) with enabled plugin in the Fuel Web UI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 1 new node with Compute role. 11. [Re-deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 12. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 13. Remove 1 node with Compute role (*i.e. remove the node, where plugin’s services are running to ensure that all plugins resources are migrated to another Compute node*). 14. [Re-deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 15. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *When adding/removing Compute node (where plugin-related services are run):*   * *all plugins resources are migrated to another Compute node* * *the environment is re-deployed successfully when adding/removing Compute node.* |

## Fuel create mirror and update (setup) of core repos

|  |  |
| --- | --- |
| Test Case ID | Fuel\_create\_mirror\_update\_core\_repos |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins)   [Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details).   1. Install the plugin. 2. Ensure that the plugin is installed successfully using CLI with running fuel plugins --list command in the Fuel CLI. 3. [Create an environment](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#create-a-new-openstack-environment) with enabled plugin in the Fuel Web UI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. Go in cli through controller / compute / storage /etc nodes and get pid of services which were launched by plugin and store them. 11. Launch the following command on the Fuel Master node:   fuel-createmirror -M   1. Launch the following command on the Fuel Master node:    1. For MOS < 8.0:   fuel --env <ENV\_ID> node --node-id <NODE\_ID1> <NODE\_ID2> <NODE\_ID\_N> --tasks upload\_core\_repos   * 1. For MOS 8.0:   fuel --env <ENV\_ID> node --node-id <NODE\_ID1> <NODE\_ID2> <NODE\_ID\_N> --tasks setup\_repositories     1. Go to controller/plugin/storage node and check if plugin's services are alive and aren't changed their pid. 2. Check with fuel nodes command that all nodes are remain in ready status. 3. Rerun OSTF. |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *Plugin's services shouldn't be restarted after corresponding task was executed. If they are restarted as some exception, this information should be added to plugin's User Guide.*  *Cluster (nodes) should remain in ready state.*  *OSTF test should be passed on rerun.* |

## Uninstall of plugin in the deployed environment

|  |  |
| --- | --- |
| Test Case ID | uninstall\_plugin\_with\_deployed\_env |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. Uninstall the plugin with running fuel plugins --remove <plugin-name>==<plugin\_version> (e.g. 1.0.1). 11. Ensure that the following output appears in CLI: "400 Client Error: Bad Request (Can't delete plugin which is enabled for some environment.)" |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *Alert is displayed when trying the uninstall the plugin.* |

## Uninstall of plugin in the non-deployed environment

|  |  |
| --- | --- |
| Test Case ID | uninstall\_plugin |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. Delete listed environment 9. Uninstall the plugin with running fuel plugins --remove <plugin-name>==<plugin\_version> (e.g. 1.0.1) |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *When uninstalling the plugin, no plugin-related elements are left in the environment (e.g. UI elements disappear, Nailgun database is restored to the default state, no output for command “fuel plugins --list”).* |

# Upgrade/update

*This section provides test cases recommended in terms of update and upgrade procedure.*

## The Fuel Master node upgrade testing

*Please refer* [*to the official Mirantis OpenStack documentation*](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#upgrading-and-updating-from-earlier-releases) *to learn more about the procedure and make up the corresponding test case for your plugin.*

*In general, this case if for the situation when e.g. your plugin is targeted at 6.1 release of Mirantis OpenStack and you'd like to perform upgrade of the Fuel Master node to 7.0 Mirantis OpenStack.*

|  |  |
| --- | --- |
| Test Case ID | upgrade\_Master\_node |
| Steps | *Your plugin is compatible with 6.1 MOS.*   1. Install the version of MOS that is compatible with your plugin (e.g. 6.1). 2. Install a plugin that is compatible with this version (e.g., 6.1) using standard flow. 3. Finalize the environment configuration. 4. Deploy an environment. 5. Upgrade the Fuel Master node *(e.g., 6.1->7.0).* 6. Verify cluster and plugin functionality    1. make sure all nodes are left in ready state    2. run OSTF checks    3. make sure all plugin-related services are running. |
| Expected Result | Cluster and plugin stay fully operational.  When the upgrade is complete, the following messages will appear under the Releases tab in the Fuel Web UI:  New release available: Kilo on Ubuntu 14.04 (2015.1.0-7.0). |

## Update the plugin to minor version in the deployed environment

*Fuel Plugin Framework allows updating the plugin to the minor version.*

*Please consider adding this case because bug fixes for Fuel plugins are delivered as new minor versions. Refer to* [*the Fuel CLI reference on Fuel plugins*](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#fuel-plugins-cli) *to learn about the procedure.*

|  |  |
| --- | --- |
| Test Case ID | update\_plugin\_to\_minor |
| Steps | 1. Copy the first version of the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. Copy the second version of the plugin (minor one) to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 11. Run the following command fuel plugins --update <fuel-plugin-file> 12. Make sure all nodes are left in ready state. 13. Run OSTF checks. 14. Make sure all plugin-related services are running. |
| Expected Result | The new plugin version is displayed in the output of the fuel plugins --list command. |

## Apply maintenance updates to deployed environment

*Mirantis OpenStack features the ability to receive patches via the common flow called* [*Maintenance Updates*](https://docs.mirantis.com/openstack/fuel/fuel-7.0/maintenance-updates.html#maintenance-updates)*. Please reach out to Partner Enablement team about the latest Maintenance Updates portion to test against with you plugin.*

|  |  |
| --- | --- |
| Test Case ID | apply\_mu |
| Steps | 1. Copy the plugin to the Fuel Master node (please refer to [the User Guide](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#install-fuel-plugins) for more details). 2. Install the plugin. 3. Ensure that plugin is installed successfully with running fuel plugins --list command in the Fuel CLI. 4. [Add](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#add-nodes-to-the-environment) 3 nodes with Controller role and 1 node with Compute and another role. 5. [Finalize environment configuration](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#configure-your-environment) (e.g. networking, nodes interfaces). 6. Enable the plugin and configure it following the instructions from the Plugin Guide. 7. Run [network verification check](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#verify-networks). 8. [Deploy the cluster](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#deploy-changes). 9. [Run OSTF.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/user-guide.html#post-deployment-check) 10. Once environment is deployed, apply maintenance updates following [the instructions.](https://docs.mirantis.com/openstack/fuel/fuel-7.0/maintenance-updates.html#maintenance-updates) 11. Check is plugin services continue running. 12. Make sure all nodes are in ready state and no regression is observed. 13. Run OSTF checks. |
| Expected Result | *Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.*  *Cluster is created and network verification check is passed.*  *Plugin is enabled and configured in the Fuel Web UI.*  *OSTF tests (Health Checks) are passed.*  *Environment is deployed successfully.*  *Maintenance Updates do not affect running services related to the plugin (e.g. the services aren't restarted).*  *Cluster remains in the fully operational state after applying Maintenance Updates.* |

# Other

## Fuel plugin interoperability testing

### Testing with different plugin types

*Mirantis recommends that you test your plugin with any other plugins that belong to different category.*

*For instance, if you have SDN solution turned into the plugin, you could take up the one that provides some application to run at the top of Mirantis OpenStack. Below you can find the list of most commonly used plugin combinations:*

* *LDAP*
* *Contrail*
* *LMA Toolchain*
* *Zabbix*

*The latest versions of these plugins can be downloaded from* [*the Fuel Plugin Catalog*](https://www.mirantis.com/products/openstack-drivers-and-plugins/fuel-plugins/)*.*

*Please, be sure to reach out to Partner Enablement team to get the latest information on the most frequently used plugin combination to plan your testing procedure accordingly.*

## Performance testing

*Please note that all instructions on installing and configuring the tools are provided in the* [*Testing tools recommended by Mirantis*](#h.3dhziox4g0js) *section of this document (click the tools names to access documentation).*

***NOTE:***

***Performance testing is supposed to be run on 20+ nodes (specifically, when using Rally and Shaker).***

### Networking performance

*Be sure to download and install* [*Shaker*](http://shaker-docs.readthedocs.org/en/latest/installation.html)*.*

### API benchmarking

*Be sure to download and install* [*Rally*](https://rally.readthedocs.org/en/latest/install.html#automated-installation)*.*

### Storage performance

*Be sure to download and install* [*Fio*](https://github.com/axboe/fio) *or* [*Wally*](https://github.com/Mirantis/disk_perf_test_tool/blob/master/INSTALL.md)*.*

# Appendix

*Provide any links to external resources or documentation here.*

|  |  |
| --- | --- |
| **№** | **Resource title** |
| 1 | Resource title should be clickable (must contain the link). |
|  |  |

# Tests included into the tools

***Please do not copy this list into your Test Plan.***

*This section provides the detailed information on the test cases run within every specific tool listed in* [*the corresponding section*](#h.3dhziox4g0js)*.*

## OSTF (Health Checks)

### Sanity group

* Ceilometer test to list meters, alarms and resources
* Request flavor list
* Request image list using Nova
* Request instance list
* Request absolute limits list
* Request snapshot list
* Request volume list
* Request image list using Glance v1
* Request image list using Glance v2
* Request stack list
* Request active services list
* Request user list
* Check that required services are running
* Request list of networks

### Smoke group

* Create instance flavor
* Check create, update and delete image actions using Glance v1
* Check create, update and delete image actions using Glance v2
* Create volume and boot instance from it
* Create volume and attach it to instance
* Instance live migration
* Check network connectivity from instance via floating IP
* Create keypair
* Create security group
* Check network parameters
* Launch instance
* Launch instance, create snapshot, launch instance from snapshot
* Create user and authenticate with it to Horizon

### HA group

* Check data replication over mysql
* Check if amount of tables in databases is the same on each node
* Check galera environment state
* Check pacemaker status
* RabbitMQ availability
* RabbitMQ replication

### Platform group

* Ceilometer test to check alarm state and get Nova metrics
* Ceilometer test to check notifications from Glance
* Ceilometer test to check notifications from Keystone
* Ceilometer test to check notifications from Neutron
* Ceilometer test to check notifications from Cinder
* Ceilometer test to create, check and list samples
* Ceilometer test to create, update, check and delete alarm
* Typical stack actions: create, delete, show details, etc.
* Advanced stack actions: suspend, resume and check
* Check stack autoscaling
* Check stack rollback
* Update stack actions: in-place, replace and update whole template

### Cloud validation group

* Check disk space outage on controller and compute nodes
* Check log rotation configuration on all nodes

## Tempest

All Tempest tests are running except a small number of those known as ‘should-fail’ for a particular MOS version.

Tempest also covers negative scenarios such as authorization violation or parameters’ mismatch. The following tests are expected to fail:

* The tests listed here<https://github.com/Mirantis/mos-tempest-runner/tree/master/shouldfail/7_0>
* Tests for Swift because there is RadosGW instead of Swift.

## Rally

The following test cases are tested using Rally:

### Ceilometer

* A user is able to define, modify, check a status and history, get a list and delete alarms
* A user is able to create meters and get statistics for them
* A user is able to define list of meters and resources

### Cinder

* A user is able to create, clone, extend, attach and delete volumes
* A user is able to create a volume backup and restore from it
* A user is able to convert a volume to an image
* A user is able to clone volumes
* A user is able to create and delete snapshots of volumes
* A user is able to convert snapshots back to new volumes and delete the volumes afterwards
* A user is able to get a list of volumes and snapshots

### Heat

* A user is able to create, suspend, resume, check and delete stacks
* A user is able to add, modify, and delete resources from stacks
* A user is able to manage resource groups

### Glance

* A user is able to create, modify their metadata, and delete images
* A user is able to get a list of images

### Keystone

* A user is able to manage (create, modify, and delete) services and roles
* A user is able to manage users and tenants
* A user is able to assign roles to users

### Murano

* A user is able to import, update, and delete packages
* A user is able to create, deploy, and delete an environment

### Nova

* A user is able to create (one by one or multiple at once) and destroy VMs
* A user is able to do power management of VM (hard/soft reboot, power off/on)
* A user is able to pause/unpause a VM
* A user is able to assign Security Groups to a VM
* A user is able to migrate a VM between hypervisors
* A user is able to rebuild a VM using another image
* A user is able to resize a VM
* A user is able to attach a volume to a VM
* A user is able to migrate a VM with attached volume
* A user is able to boot a VM from volume
* A user is able to manage (add, update, and delete) quotas
* A user is able to manage keypairs
* A user is able to assign a keypair to a VM

### Neutron

* A user is able to manage networks and subnets
* A user is able to manage ports, attach them to networks
* A user is able to manage routers, attach them to networks
* A user is able to update quotas

### Swift

* A user is able to create, list, and delete containers
* A user is able to upload objects into container, list objects, download, and delete them

## Rally

The same functional test cases as above are running concurrently to measure an API response time while it’s under load.

Two runs should be performed with the concurrency set to 5 and 10.

## Shaker

The next test cases are run by Shaker:

1. Full\_l2.yaml scenario tests the bandwidth between pairs of instances in the same virtual network (L2 domain). Each instance is deployed on own compute node. The test increases the load from 1 pair until all available instances are used.
2. Full\_l3\_east\_west.yaml scenario tests the bandwidth between pairs of instances deployed in different virtual networks plugged into the same router. Each instance is deployed on its own compute node. The test increases the load from 1 pair pair until all available instances are used.
3. Full\_l3\_north\_south.yaml tests the bandwidth between pairs of instances deployed in different virtual networks. Instances with master agents are located in one network, instances with slave agents are reached via their floating IPs. Each instance is deployed on its own compute node. The test increases the load from 1 pair pair until all available instances are used.
4. TCP scenario tests TCP bandwidth to the destination host. By default it sends traffic to one of public iperf servers
5. UDP scenario tests UDP packets per second to the destination host. By default it sends traffic to one of public iperf servers.

### Storage performance

Image Uploading to Glance

Testing on 20G raw image via **time glance image-create…** command

### IOPS for block devices

Mixed sequential reads and writes and mixed random reads and writes.

1. IOPS of Ephemeral disks of a single VM.
2. IOPS of Ephemeral disks of 10, 20 VMs.
3. IOPS for Cinder volumes of a single VM.
4. IOPS for Cinder volumes of 10, 20 VMs.

The tests are done by **fio** utility with the next configs:

[random\_reads\_writes]

ramp\_time=30

runtime=120

rw=randrw

direct=1

buffered=0

numjobs=32

blocksize=4k

group\_reporting

iodepth=64

norandommap=1

thread=1

time\_based=1

wait\_for\_previous

randrepeat=0

filename=/dev/vdc

size=22g

[sequential\_reads\_writes]

ramp\_time=30

runtime=120

rw=readwrite

direct=1

buffered=0

numjobs=32

blocksize=4k

group\_reporting

iodepth=64

norandommap=1

thread=1

time\_based=1

wait\_for\_previous

randrepeat=0

filename=/dev/vdc

size=22g

[random-writers]

ramp\_time=30

runtime=120

rw=randwrite

direct=1

buffered=0

numjobs=32

blocksize=4k

group\_reporting=1

iodepth=64

unified\_rw\_reporting=1

norandommap=1

thread=1

time\_based=1

wait\_for\_previous=1

randrepeat=0

filename=/dev/vdc

size=22g

## Resiliency Tests

1. **Graceful shutdown one of the controller nodes.**
   1. Gracefully shut down one of the controller nodes. (Controller 1)
      1. (target controller): shutdown -h 0
   2. Boot 5 VMs and verify that they all are up and running.
      1. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   3. Turn on controller and wait until all services are up (Time limit 30 min)
      1. (IPMI): power on
      2. (other controller): pcs status
   4. Boot 5 VMs and verify that they all are up and running.
      1. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   5. Return the cluster to initial state.
      1. (other controller): for uuid in `nova list|grep ha-test|awk '{print $2}'`; do nova delete $uuid; done
2. **Hard power off one of the controller nodes.**
   1. Hard power off one of the controller nodes. (Controller 2)
      1. (IPMI): power off
   2. Boot 5 VMs and verify that they all are up and running.
      1. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   3. Turn on controller and wait until all services are up, but not longer than 30 min.
      1. (IPMI): power on
      2. (other controller): pcs status
   4. Boot 5 VMs and verify that they all are up and running.
      1. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   5. Return the cluster to initial state.
      1. (other controller): for uuid in `nova list|grep ha-test|awk '{print $2}'`; do nova delete $uuid; done
3. **Cut network communication on one of the controller nodes.**
   1. Cut network communication on one of the controller nodes. (Controller 3)
   2. (IPMI): ip link set down <NIC> #For every physical NIC
   3. Boot 5 VMs and verify that they all are up and running.
   4. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   5. Reboot controller
   6. (IPMI): power reset
   7. Boot 5 VMs and verify that they all are up and running.
   8. (other controller): nova boot --flavor 1 --image TestVM --min-count 5 --nic net-id=<net04 ID> ha-test
   9. Return the cluster to initial state.
   10. (other controller): for uuid in `nova list|grep ha-test|awk '{print $2}'`; do nova delete $uuid; done