# Test Plan for Openbook 1.1.0 Fuel Plugin

# Revision history

| Version | Revision date | Editor | Comment |
|---|---|---|---|
| 0.1 | 23.01.2015 | Irina Povolotskaya (ipovolotskaya@mirantis.com) | Created the template structure. |
| 0.2 | 22.10.2015 | Jeremy Fluhmann (jeremy@talligent.com) | Filled template for Openbook Fuel Plugin. |
| 0.3 | 29.10.2015 | Jeremy Fluhmann (jeremy@talligent.com) | Updated for MOS 7.0 |

# Openbook Plugin

This plugin extends Mirantis OpenStack functionality by adding Openbook customer onboarding, self-service, and cloud billing / charge-back services. Openbook is a fully-functional, simple to use cloud management solution that has been built specifically for OpenStack. It allows users to measure, manage, and monetize clouds built on OpenStack.

## Developer's specification

Available on Github repo for openstack/fuel-plugin-openbook [1].

## Limitations

The Openbook Fuel Plugin requires Ceilometer (OpenStack Telemetry) to be included in the OpenStack environment. It also requires a Sharefile account for Talligent [2] and access to the Internet.

# Test strategy

Openbook Fuel Plugin pulls information from several OpenStack endpoints. The communication between Openbook and OpenStack is tested through the Openbook API. One test verifies initial communication with the Keystone admin endpoint, while the other verifies communication with the other OpenStack endpoints.

## Acceptance criteria

An OpenStack Resource Manager should be added to Openbook. This is verified by reviewing output from the deployment.

## Test environment, infrastructure and tools

All cases run in a single environment consisting of a Fuel master node, one or more controller nodes with Ceilometer, at least one compute node, and one Openbook node.

## Product compatibility matrix

The Openbook Fuel Plugin is developed for Mirantis OpenStack 7.0 Kilo on Ubuntu 14.04.

# Type of testing

The puppet modules for the plugin test the connectivity to OpenStack prior to creating an OpenStack Resource Manager. The output of these tests is directed to files residing on the node.

## Verify Communication with OpenStack

| | |
|---|---|
| Test Case ID | verify_comm_with_openstack |
| Description | Verify that Openbook can communicate with Keystone on the admin port |
| Prerequisites | OpenStack nodes have been deployed.  Operating System node has been deployed with the name 'openbook'. |
| Steps | 1. SSH onto the Fuel Master node<br>2. Identify the IP address of the 'openbook' node via: `fuel nodes`<br>3. SSH to the 'openbook' node<br>`4. cat /tmp/connectivity-test-results.txt` |
| Expected Result | Output of step 4 contains service catalog results for OpenStack |

## Verify OpenStack Resource Manager Creation

| | |
|---|---|
| Test Case ID | verify_os_resource_manager_creation |
| Description | OpenStack nodes have been deployed.  Operating System node has been deployed with the name 'openbook'. |
| Prerequisites | OpenStack nodes have been deployed.  Operating System node has been deployed with the name 'openbook'. |
| Steps | 1. SSH onto the Fuel Master node<br>2. Identify the IP address of the 'openbook' node via: `fuel nodes`<br>3. SSH to the 'openbook' node<br>`4. cat /tmp/resource_manager_results.txt` |
| Expected Result | Output of step 4 contains “`HTTP/1.1 201 Created`” |

# System testing

## Install plugin and deploy environment

| Test Case ID | install_plugin_deploy_env |
|---|---|
| Steps | 1. Upload plugin to the master node<br>2. Install plugin<br>3. Ensure that plugin is installed successfully using cli<br>4. Create environment with enabled plugin in fuel ui<br>5. Add 3 nodes with Controller role and 1 node with Compute and another role<br>6. Apply network settings<br>7. Run network verification<br>8. Deploy the cluster<br>9. Check plugin health using cli<br>10. Run OSTF |
| Expected Result | *Plugin is installed successfully, cluster is created,* network verification and OSTF are passed, and all plugin services is enabled and worked as expected. |

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

| Test Case ID | modify_env_with_plugin_remove_add_controller |
|---|---|
| Steps | 1. Upload plugin to the master node<br>2. Install plugin<br>3. Ensure that plugin is installed successfully using cli<br>4. Create environment with enabled plugin in fuel ui<br>5. Add 3 nodes with Controller role and 1 node with Compute and another role<br>6. Apply network settings<br>7. Run network verification<br>8. Deploy the cluster<br>9. Check plugin services using cli<br>10. Run OSTF<br>11. Remove 1 nodes with Controller role<br>/*remove node, where plugin's services available, to ensure that according to ha mode all plugins resources will be replaced and available on another live node and continue to work as expected*/<br>12. Re-deploy cluster<br>13. Check plugin services using cli |

| | |
|---|---|
| | 14. Run OSTF<br>15. Add 1 new  node with Controller role<br>16. Re-deploy cluster<br>17. Check plugin services using cli<br>18. Run OSTF |
| Expected Result | *Plugin is installed successfully,  cluster is created,  network verification and OSTF are passed, and all plugin services is enabled after migration in ha mode and worked as expected after modifying of environment.* |

## Modifying env with enabled plugin (removing/adding compute node)

| | |
|---|---|
| Test Case ID | modify_env_with_plugin_remove_add_compute |
| Steps | 1. Upload plugin to the master node<br>2. Install plugin<br>3. Ensure that plugin is installed successfully using cli<br>4. Create environment with enabled plugin in fuel ui<br>5. Add 3 nodes with Controller role and 2 nodes with compute roles and 1 another role<br>6. Apply network settings<br>7. Run network verification<br>8. Deploy the cluster<br>9. Check plugin services using cli<br>10. Run OSTF<br>11. Remove 1 compute node<br>12. Re-deploy cluster<br>13. Check plugin services using cli<br>14. Run OSTF<br>15. Add 1 compute node<br>16. Re-deploy cluster<br>17. Check plugin services using cli<br>18. Run OSTF |
| Expected Result | *Plugin is installed successfully,  cluster is created,  network verification and OSTF are passed, and all plugin services is enabled and worked as expected after modifying of environment.* |

## Uninstall of plugin with deployed environment

| | |
|---|---|
| Test Case ID | uninstall_plugin_with_deployed_env |
| Steps | 1. install plugin |

| | |
|---|---|
| | 2. deploy environment with enabled plugin functionality<br>3. run ostf<br>4. try to delete plugin and ensure that present in cli alert: "400 Client Error: Bad Request (Can't delete plugin which is enabled for some environment.)"<br>5. remove environment<br>6. remove plugin<br>7. check that it was successfully removed |
| Expected Result | *Plugin was installed successfully. Alert is present when we trying to delete plugin which is attached to enabled environment. When environment was removed, plugin is removed successfully too.* |

## Uninstall of plugin

| | |
|---|---|
| Test Case ID | uninstall_plugin |
| Steps | 1. install plugin<br>2. check that it was installed successfully<br>3. remove plugin<br>4. check that it was successfully removed |
| Expected Result | *Plugin was installed and then removed successfully* |

# Appendix

Provide any links to external resources or documentation here.

| № | Resource title |
|---|---|
| 1 | https://github.com/openstack/fuel-plugin-openbook/blob/master/specs/openbook-plugin.rst |
| 2 | https://talligent.sharefile.com/ |