# Test Plan for ScaleIO Cinder 1.0.0 Fuel Plugin

[Test Plan for <Plugin name> <plugin\_version> Fuel Plugin](#h.9sjw9gygbth8)

[Revision history](#h.6rt6kxi0q67o)

[NOTE: /\* Please do not copy description of section in your reports, just provide required information from description \*/](#h.3dxu23cwxr7r)

[<Plugin name> Plugin](#h.u6oeahaeh3zs)

[Developer’s specification](#h.2tgi2odyzskc)

[Limitations](#h.2b0cy5ovolma)

[Test strategy](#h.9jbr2unh6i9)

[Acceptance criteria](#h.nzhxdderq67)

[Test environment, infrastructure and tools](#h.pw7thep7oq4y)

[Product compatibility matrix](#h.cut28xqenhu1)

[Type of testing](#h.l24plbdk72x)

[<Case 1 title>](#h.ud54pvm25504)

[<Case 2 title>](#h.me240mqtn8ni)

[System testing](#h.sb63vn8gl4ym)

[Install plugin and deploy environment](#h.sb63vn8gl4ym)

[Modifying env with enabled plugin (removing/adding controller nodes)](#h.i7qfs69lnngb)

[Modifying env with enabled plugin (removing compute node)](#h.ljpmaxg12tdy)

[Uninstall of plugin with deployed environment](#h.sf9irtsbnouf)

[Uninstall of plugin](#h.3oa18jdmatyj)

[The Fuel Master node upgrade testing](#h.md720k5dygbn)

[Appendix](#h.sq6iwh656wfp)

# Revision history

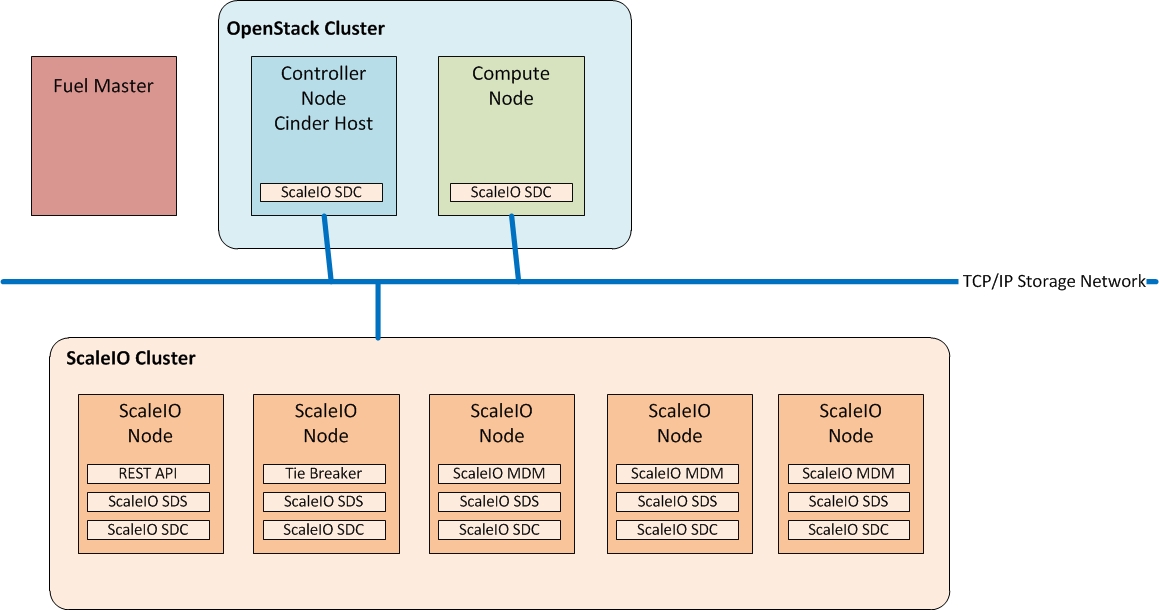
|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Revision date** | **Editor** | **Comment** |
| 0.1 | 23.01.2015 | Irina Povolotskaya  (ipovolotskaya@mirantis.com) | Created the template structure. |
| 0.2 |  |  |  |
|  |  |  |  |

# ScaleIO Cinder Plugin

This Fuel plugin for ScaleIO enables OpenStack to use with an external ScaleIO cluster for block storage. This ScaleIO plugin for Fuel extends Mirantis OpenStack functionality by adding and configuring a Cinder driver for EMC’s ScaleIO software defined storage.

ScaleIO is a software-only solution that uses existing servers' local disks and LAN to create a virtual SAN that has all the benefits of external storage—but at a fraction of cost and complexity. ScaleIO utilizes the existing local internal storage and turns it into internal shared block storage.

The following diagram shows the plugin's high level architecture:



The figure above shows the required OpenStack roles and services:

Service/Role Name Description Installed in

Controller Node + Cinder Host A node that runs network, volume, API, scheduler, and image services. Each service may be broken out into separate nodes for scalability or availability. In addition this node is a Cinder Host, that contains the Cinder Volume Manager OpenStack Cluster

Compute Node A node that runs the nova-compute daemon that manages Virtual Machine (VM) instances that provide a wide range of services, such as web applications and analytics. OpenStack Cluster

In the external ScaleIO cluster we have installed the following roles and services:

Service Name Description Installed in

SclaeIO Gateway (REST API) The ScaleIO Gateway Service, includes the REST API to communicate storage commands to the SclaeIO Cluster, in addtion this service is used for authentication and certificate management. ScaleIO Cluster

Meta-data Manager (MDM) Configures and monitors the ScaleIO system. The MDM can be configured in redundant Cluster Mode, with three members on three servers, or in Single Mode on a single server. ScaleIO Cluster

Tie Breaker (TB) Tie Breaker service helps determining what service runs as a master vs. a slave ScaleIO Cluster

Storage Data Server (SDS) Manages the capacity of a single server and acts as a back-end for data access.The SDS is installed on all servers contributing storage devices to the ScaleIO system. These devices are accessed through the SDS. ScaleIO Cluster

Storage Data Client (SDC) A lightweight device driver that exposes ScaleIO volumes as block devices to the application that resides on the same server on which the SDC is installed. Openstack Cluster

Note: for more information in how to deploy a ScaleIO Cluster, please refer to the ScaleIO manuals located in the download packages for your platform: http://www.emc.com/products-solutions/trial-software-download/scaleio.htm and/or watch the demo

## 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).*

# Test strategy

*Provide information on types of implemented tests (that enter test cases below). Specify which tests will be automated. Add plans for introducing more tests in the future (like compatibility, security, performance).*

## Acceptance criteria

*Provide information on the set of basic tests that should be performed to consider that the feature has passed the testing process. Add information about test coverage and metrics.*

## Test environment, infrastructure and tools

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

## Product compatibility matrix

*Provide information on interoperability for the plugin. (\* not a mandatory field). For example, if your plugin is compatible with specific Mirantis OpenStack releases or other plugins.*

**NOTE: /\* MUST HAVE TESTS.** **These scenarios of tests mandatory must be added in your test plans. \*/**

# System testing

## Install plugin and deploy environment

|  |  |
| --- | --- |
| Test Case ID | install\_plugin\_deploy\_env |
| Steps | 1. Upload plugin to the master node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 1 node with Compute and another role     Picture of the External ScaleIO Cluster Running    Retrive the external ScaleIO Cluster information. For our example these are the configuration settings:    Use the ScaleIO Cluster information to update the ScaleIO Plugin information     1. Apply network settings   Use the networking settings that are appropriate for your environment. For our example we used the default settings provided by Fuel:     1. Run network verification      1. Deploy the cluster      1. Check plugin health using cli 2. 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 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 1 node with Compute and another role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Check plugin services using cli 10. Run OSTF 11. Remove 1 nodes with Controller role   /\*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\*/   1. Re-deploy cluster 2. Check plugin services using cli 3. Run OSTF 4. Add 1 new node with Controller role 5. Re-deploy cluster 6. Check plugin services using cli 7. 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 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 2 nodes with compute roles and 1 another role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Check plugin services using cli 10. Run OSTF 11. Remove 1 compute node 12. Re-deploy cluster 13. Check plugin services using cli 14. Run OSTF 15. Add 1 compute node 16. Re-deploy cluster 17. Check plugin services using cli 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 3. run ostf 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.)" 5. remove environment 6. remove plugin 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      1. check that it was installed successfully 2. remove plugin 3. check that it was successfully removed |
| Expected Result | *Plugin was installed and then removed successfully* |

# The Fuel Master node upgrade testing

# Appendix

Provide any links to external resources or documentation here.

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

