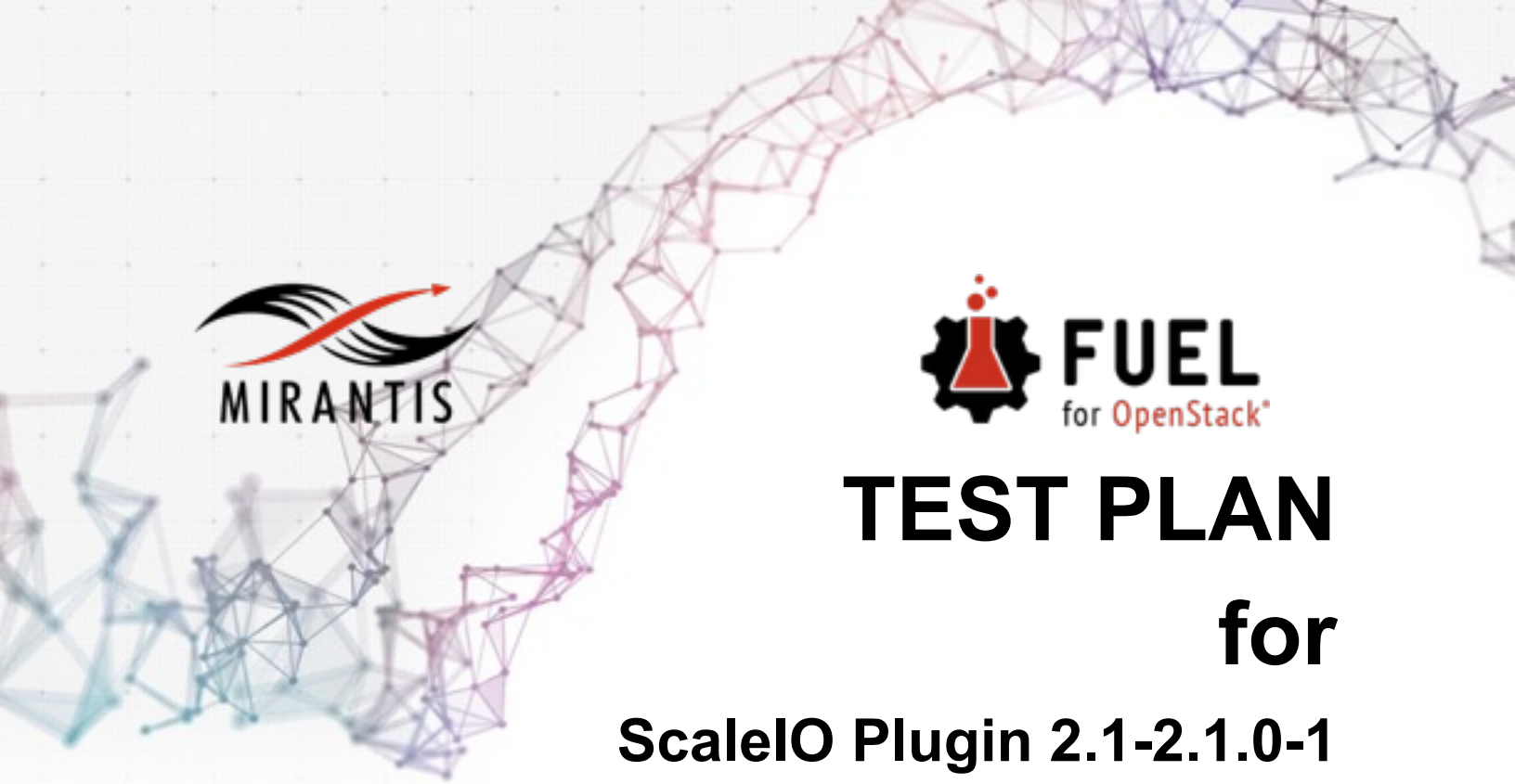

TEST PLAN

for

ScaleIO Plugin 2.1-2.1.0-1

Mirantis OpenStack 8.0

Revision history

Version	Revision date	Editor	Comment
0.1	28.03.2016	Alexey Morlang (alexey.morlang@emc.com)	Initial version.
0.2	07.04.2016	Alexey Morlang (alexey.morlang@emc.com)	Added exceptions for OSTF.
0.3	08.04.2016	Alexey Morlang (alexey.morlang@emc.com)	Added maintenance update and create mirror tests.
0.4	08.04.2016	Alexey Morlang (alexey.morlang@emc.com)	Added examples for SCLI tool.
0.5	12.04.2016	Alexey Morlang (alexey.morlang@emc.com)	Add/remove 1 controllers in case 'modify_env_with_plugin_remove_add_controller'
0.6	02.06.2016	Alexey Morlang (alexey.morlang@emc.com)	Plugin version for Fuel8.0

ScaleIO Plugin

EMC ScaleIO is a software-only server-based storage area network (SAN) that converges storage and compute resources to form a single-layer, enterprise-grade storage product. ScaleIO storage is elastic and delivers linearly scalable performance. Its scale-out server SAN architecture can grow from a few to thousands of servers. ScaleIO uses servers' direct-attached storage (DAS) and aggregates all disks into a global, shared, block storage. ScaleIO features single-layer compute and storage architecture without requiring additional hardware or cooling/ power/space.
Breaking traditional barriers of storage scalability, ScaleIO scales out to hundreds and thousands of nodes and multiple petabytes of storage. The parallel architecture and distributed volume layout delivers a massively parallel system that deliver I/O operations through a distributed system. As a result, performance can scale linearly with the number of application servers and disks, leveraging fast parallel rebuild and rebalance without interruption to I/O. ScaleIO has been carefully designed and implemented with ScaleIO software components so as to consume minimal computing resources.
With ScaleIO, any administrator can add, move, or remove servers and capacity on demand during I/O operations. The software responds automatically to any infrastructure change and rebalances data accordingly across the grid nondisruptively. ScaleIO can add capacity on demand, without capacity planning or data migration and grow in small or large increments and pay as you grow, running on any server and with any storage media.
ScaleIO natively supports all leading Linux distributions and hypervisors. It works agnostically with any solid-state drive (SSD) or hard disk drive (HDD) regardless of type, model, or speed.

ScaleIO Components

ScaleIO Data Client (SDC)

- Acts as Block Device Driver
- Exposes volumes to applications
- Service must run to provide access to volumes
- Over TCP/IP

ScaleIO Data Service (SDS)

- Abstracts storage media
- Contributes to storage pools

- Performs I/O operations

ScaleIO Metadata Manager (MDM)
- Not located in the data path
- Provides Monitoring and Configuration management
- Holds cluster-wide component mapping

ScaleIO Gateway (GW)
- Provides REST API for Configuration management

ScaleIO Cinder and Nova Drivers

ScaleIO includes a Cinder driver, which interfaces between ScaleIO and OpenStack, and presents volumes to OpenStack as block devices which are available for block storage. It also includes an OpenStack Nova driver, for handling compute and instance volume related operations. The ScaleIO driver executes the volume operations by communicating with the backend ScaleIO MDM through the ScaleIO REST Gateway.

Developer's specification

Is available on GitHub repository.

Limitations

There are following limitations in plugin functionality:
- plugin is compatible with Mirantis 8.0
- plugin supports the only Ubuntu environment
- the only hyper converged environment is supported - there is no separate ScaleIO Storage nodea
- there is no ability to disable completely Cinder-Lvm because of FUEL limitations
- disks for SDS-es should be unallocated, they will be cleaned up
- MDMs and Gateways are deployed together and only onto controller nodes
- there is no ability to separate data network traffic from replication traffic
- there is no fault sets support

Test strategy

The ScaleIO plugin creates a GUI element to collect the information necessary to deploy and configure EMC ScaleIO in the cluster nodes. The testing strategy is to confirm that all options in the GUI are handled properly and ScaleIO is successfully deployed (if not selected option to use existing ScaleIO) and Cinder and Nova are properly configure to use the ScaleIO cluster as the block storage service and as ephemeral storage service.

Acceptance criteria

All tests should pass.

Test environment, infrastructure and tools

The test lab should include 5 nodes. The following designations for the nodes:

1) Fuel master node (w/ 50GB Disk, 2 Network interfaces [Mgmt, PXE])
2) OpenStack Controller #1 node
3) OpenStack Controller #2 node
4) OpenStack Controller #3 node
5) OpenStack Compute node

Each node shall have at least 2 CPUs, 4GB RAM, 2x100GB disks, 3 Network interfaces. The 3 interfaces will be used for the following purposes:

- **Admin (PXE) network**: Mirantis OpenStack uses PXE booting to install the operating system, and then loads the OpenStack packages for you.
- **Private, Management and Storage networks**: All of the OpenStack management traffic will flow over this network (“Management” and “Storage” will be separated by VLANs). To re-use the network it will also host the private network. It will be added into each Virtual Machines when they boot. It will therefore be the route where traffic flows in and out of the VM.
- **Private network**: This network is used by OpenStack service nodes and the floating IP address range. Public network must have access to the internet.

In order to perform management operations with ScaleIO cluster there is the tool ‘scli’. It is a management tool that is available on all controller nodes after deployment.

How to do checks with ScaleIO CLI tool (scli):

1) How to check ScaleIO cluster state with help of ‘scli’ tool (the tool is available on all controller nodes after deployment):

- Find controller node with minimal management IP (initially it is master ScaleIO MDM), ssh to this controller node
- read cluster information: scli —query_cluster
- login to cluster: scli —login —mdm_ip <ip_of_master_mdm_from_query_cluster> —username admin —password <password_from_plugin_settings>
- query information about all SDS: scli —mdm_ip <ip_of_master_mdm_from_query_cluster> —query_all_sds
- query information about all SDC: scli —mdm_ip <ip_of_master_mdm_from_query_cluster> —query_all_sdc

There should controlled ‘State: Normal’. For 3 controllers configuration there should be 1 master MDM, 1 slave MDMs, 1 Tie-Breakes. For 5 controllers configurations - 1 master MDM, 2 slave MDMs, 2 Tie-Breakers. Below there is an example of output for 3 controllers and 1 compute cluster.

root@node-17:~# scli --query_cluster
Cluster:
Mode: 3_node, State: Normal, Active: 3/3, Replicas: 2/2
Master MDM:
Name: 192.168.0.3, ID: 0x65d3369a2fde5cf0
IPs: 192.168.0.3, Management IPs: 192.168.0.3, Port: 9011
Version: 2.0.5014
Slave MDMs:
Name: 192.168.0.4, ID: 0x1bd565225d6178e1
IPs: 192.168.0.4, Management IPs: 192.168.0.4, Port: 9011
Status: Normal, Version: 2.0.5014
Tie-Breakers:
Name: 192.168.0.5, ID: 0x49871a8b5be6d032
IPs: 192.168.0.5, Port: 9011
Status: Normal, Version: 2.0.5014

root@node-17:~# scli —mdm_ip 192.168.0.3,192.168.0.4 —login —username admin —password qwe123QWE

root@node-17:~# scli —mdm_ip 192.168.0.3,192.168.0.4 —query_all_sds
Query-all-SDS returned 4 SDS nodes.

Protection Domain 0da7f39f00000000 Name: default
SDS ID: 2a99509e00000003 Name: node-17 State: Connected, Joined
IP: 192.168.1.2,192.168.0.3 Port: 7072 Version: 2.0.5014

SDS ID: 2a99509d00000002 Name: node-19 State: Connected, Joined IP: 192.168.1.3,192.168.0.5 Port: 7072 Version: 2.0.5014
SDS ID: 2a99509c00000001 Name: node-18 State: Connected, Joined IP: 192.168.1.4,192.168.0.6 Port: 7072 Version: 2.0.5014
SDS ID: 2a99509b00000000 Name: node-20 State: Connected, Joined IP: 192.168.1.1,192.168.0.4 Port: 7072 Version: 2.0.5014

root@node-17:~# scli —mdm_ip 192.168.0.3,192.168.0.4 —query_all_sdc
MDM restricted SDC mode: Disabled
Query all SDC returned 4 SDC nodes.
SDC ID: adcc709800000003 Name: N/A IP: 192.168.0.3 State: Connected GUID: 7342402A-771A-458A-9938-FD75DFE24488 Version: 2.0.5014
Read bandwidth: 0 IOPS 0 Bytes per-second
Write bandwidth: 0 IOPS 0 Bytes per-second
SDC ID: adcc709700000002 Name: N/A IP: 192.168.0.4 State: Connected GUID: C967D14E-6F40-4FF9-858B-033AB57B6FB4 Version: 2.0.5014
Read bandwidth: 0 IOPS 0 Bytes per-second
Write bandwidth: 0 IOPS 0 Bytes per-second
SDC ID: adcc709600000001 Name: N/A IP: 192.168.0.5 State: Connected GUID: 44285A52-F063-4934-A004-48A51E9B9BAD Version: 2.0.5014
Read bandwidth: 0 IOPS 0 Bytes per-second
Write bandwidth: 0 IOPS 0 Bytes per-second
SDC ID: adcc709500000000 Name: N/A IP: 192.168.0.6 State: Connected GUID: 6706B573-F41C-4CFF-850A-C077394E2034 Version: 2.0.5014
Read bandwidth: 0 IOPS 0 Bytes per-second
Write bandwidth: 0 IOPS 0 Bytes per-second

2) How to remove disconnected SDS and SDC from ScaleIO cluster state with help of ‘scli’ tool. Interception is presented for SDS, for SDC the actions are the same with just replacing ‘sds’ words with ‘sdc’ in cli:

- login to cluster: scli —login —mdm_ip <ip_of_master_mdm_from_query_cluster> —username admin —password <password_from_plugin_settings>
- find disconnected SDS/SDC by quering: scli —mdm_ip <ip_of_master_mdm_from_query_cluster> —query_all_sds
- remove node from cluster scli —mdm_ip <ip_of_master_mdm_from_query_cluster> —remove_sds —sds_id <id_of_sds_from_query> —i_am_sure

Product compatibility matrix

ScaleIO Plugin version	Compatible Fuel version	OpenStack and OS Version	ScaleIO version
2.0.0	6.1	Juno on Ubuntu14.04	2.0
2.0.0	7.0	Kilo on Ubuntu14.04	2.0
2.1.0	8.0	Liberty on Ubuntu14.04	2.0

System testing

Install plugin and deploy environment

Test Case ID	install_plugin_deploy_env
Steps	1. Upload plugin to the master fuel node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled ScaleIO plugin in fuel ui a. Provide passwords for Admin and for Gateway, e.g. qwe123QWE b. Keep protection domain name and storage pool names in default values c. Input '/dev/sdb' in Storage devices d. Select checkbox 'Controller as SDS' 5. Add 3 node with Controller and Cinder role 6. Add 1 node with Compute role 7. Make second disk (sdb) unallocated for all four nodes 8. Apply network settings a. IP addresses and assigning networks to interfaces depend on actual network environment of test lab 9. Run network verification 10. Deploy the cluster 11. Check ScaleIO cluster state from the controller node with minimal management IP (initially it is master ScaleIO MDM) 12. Login to Horizon with the admin user when the OpenStack deployment is finished a. Create volume from image b. Create flavor with 8GB disks and zero swap c. Run instance with just created flavour d. Delete volume and instance created above 13. Run OSTF excepting test with launch of instances
Expected Result	1. Plugin is installed successfully, cluster is created, network verification. 2. ScaleIO cluster is configured in 'node_3' mode: 1. Two MDMs on the two of controller 2. One TieBreaker on the one of controller 3. SDS and SDC on each node and they are in connected state 4. Volume and instance are created and deleted successfully via Horizon. 3. OSTF are passed. Tests with launch of instances should be excluded because they require special flavour.

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

Test Case ID	modify_env_with_plugin_remove_add_controller
Environment	Fuel master node (w/ 50GB Disk, 2 Network interfaces [Mgmt, PXE]) 1) OpenStack Controller #1 node 2) OpenStack Controller #2 node 3) OpenStack Controller #3 node 4) OpenStack Controller #4 node 5) OpenStack Controller #5 node 6) OpenStack Compute Network and disks configuration is the same as described in common section
Steps	1. Upload plugin to the master fuel node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled ScaleIO plugin in fuel ui a. Provide passwords for Admin and for Gateway, e.g. qwe123QWE b. Keep protection domain name and storage pool names in default values c. Input '/dev/sdb' in Storage devices d. Select checkbox 'Controller as SDS' 5. Add 3 node with Controller and Cinder role 6. Add 1 node with Compute role 7. Make second disk (sdb) unallocated for all four nodes 8. Apply network settings a. IP addresses and assigning networks to interfaces depend on actual network environment of test lab 9. Run network verification 10. Deploy the cluster 11. Run OSTF excepting test with launch of instances 12. Remove 1 controller node. 13. Check ScaleIO cluster state 14. Re-deploy cluster 15. Check ScaleIO cluster state 16. Run OSTF excepting test with launch of instances 17. Add 1 new node with Controller role 18. Check ScaleIO cluster state 19. Re-deploy cluster 20. Run OSTF excepting test with launch of instances

Expected Result	1. Plugin is installed successfully, cluster is created, network verification.
	2. ScaleIO cluster is configured in '5_node' mode:
	    1. Three MDMs on the two of controller
	    2. Two TieBreaker on the one of controller
	    3. SDS and SDC on each node and they are in connected state
	    4. Volume and instance are created and deleted successfully via Horizon.
	3. Removed controller(s) is removed from ScaleIO cluster.
	4. Newly added controller is added to ScaleIO cluster.
	5. OSTF are passed in all runs. Tests with launch of instances should be excluded because they require special flavour.

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 1. Create environment with enabled ScaleIO plugin in fuel ui a. Provide passwords for Admin and for Gateway, e.g. qwe123QWE b. Keep protection domain name and storage pool names in default values c. Input '/dev/sdb' in Storage devices d. Select checkbox 'Controller as SDS' 2. Add 3 node with Controller and Cinder role 3. Add 1 node with Compute role 4. Make second disk (sdb) unallocated for all four nodes 5. Apply network settings a. IP addresses and assigning networks to interfaces depend on actual network environment of test lab 6. Run network verification 7. Deploy the cluster 4. Check ScaleIO cluster state 5. Run OSTF excepting test with launch of instances 6. Remove 1 compute node 7. Re-deploy cluster 8. Check ScaleIO cluster state 9. Remove disconnected SDS and SDC from ScaleIO cluster via cli 10. Run OSTF excepting test with launch of instances 11. Add 1 compute node 12. Re-deploy cluster 13. Update controller nodes by running task update_hosts via fuel cli 14. Check ScaleIO cluster state 15. Run OSTF excepting test with launch of instances
Expected Result	1. Plugin is installed successfully, cluster is created, network verification. 2. ScaleIO cluster is configured in 'node_3' mode: 1. Two MDMs on the two of controller 2. One TieBreaker on the one of controller 3. SDS and SDC on each node and they are in connected state 4. Volume and instance are created and deleted successfully via Horizon. 3. OSTF are passed. Tests with launch of instances should be excluded because they require special flavour.

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 excepting test with launch of instances 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 2. Check that it was installed successfully 3. Remove plugin 4. Check that it was successfully removed
Expected Result	Plugin was installed and then removed successfully

Upgrade/update

The Fuel Master node upgrade testing

Test Case ID	upgrade_Master_node
Steps	1. Install the version of MOS6.1 2. Install plugin 3. Deploy environment with enabled plugin functionality 4. Run OSTF excepting test with launch of instances 5. Upgrade the Fuel Master node 6.1 -> 7.0 6. Verify cluster and plugin functionality a. make sure all nodes are left in ready state b. run OSTF checks excepting test with launch of instances
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).

Apply maintenance updates to deployed environment

Test Case ID	apply_mu
Steps	1. Install plugin 2. Deploy environment with enabled plugin functionality 3. Run OSTF excepting test with launch of instances 4. Once environment is deployed, apply maintenance updates following the instructions. 5. Make sure all nodes are in ready state and no regression is observed. 6. Run OSTF excepting test with launch of instances

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.

Appendix

№	Resource title
1	ScaleIO Fuel Plugin GitHub repository
2	ScaleIO User Guide
3	ScaleIO OpenStack Information
4	