TEST PLAN

for

fuel-plugin-groundwork-monitor 7.1-7.1.0-1

Mirantis OpenStack 8.0

Contents

Revision history

Version	***Revision date***	***Editor***	***Comment***
0.1	2015-01-23	Irina Povolotskaya (ipovolotskaya@mirantis.com)	Created the template structure.
0.2	2016-02-29	Irina Povolotskaya (ipovolotskaya@mirantis.com)	Updated the template with testing approach and tools recommended by Mirantis.
1.0	2016-07-13	GroundWork, Inc.	Cleaned up the formatting. Adapted to the fuel-plugin-groundwork-monitor plugin.
1.1	2016-08-02	GroundWork, Inc.	List the GroundWork Installer's own log file as a potential source of troubleshooting information. Fixed mistakenly duplicate test-case IDs. Added test cases that are peripheral to operation of GroundWork Monitor and instead have to do with changes to the broader OpenStack deployment: *fuel_create_mirror_update_core_repos* *apply_maintenance_update*

fuel-plugin-groundwork-monitor Plugin

The GroundWork Monitor plugin for Fuel provides the functionality to add GroundWork Monitor to Mirantis OpenStack as a monitoring backend option, using the Fuel Web UI in a user-friendly manner. GroundWork Monitor is a monitoring solution that can be used to watch over many different types of resources, including computer hardware, network devices, application presence and responsiveness, and so forth.

This Fuel plugin is hot-pluggable, meaning it can be installed on a deployed cluster (in ready state), not just on an undeployed cluster. Then to obtain a node running GroundWork Monitor, it is only necessary to enable the plugin under the Settings tab for the given environment in the Fuel Web UI, add a node with the new GROUNDWORK_MONITOR role, and deploy changes. More information on the installation and configuration process is presented in the Plugin Guide. Once the GroundWork Monitor software itself is configured, the local monitoring setup on the GroundWork Monitor machine will constitute precious cargo, and that node should never be deleted capriciously.

Developer's specification

The writ to write this plugin was simple: wrap the standard GroundWork installer in a Fuel Plugin package for easy deployment to a node. It is normally expected that the GroundWork install will occupy its own node, so as not to have other activity on the machine compromise the monitoring function. There is no other formal specification.

Limitations

- Deployment of a Remote PostgreSQL database is not supported by this Fuel plugin. The working assumption is that you are installing a standalone monitoring server.
- The GroundWork Monitor software is deployed using fixed standard administrative credentials, which are not settable during the install itself. It is necessary to perform some manual post-install steps to establish site-local credentials for proper security. See the Plugin Guide for details.
- As installed by the Fuel plugin when it is deployed on a node, the GroundWork Monitor software is initially configured to only monitor the machine on which it runs. Extending the setup to monitor a larger set of resources involves manual post-install steps.
- The Fuel plugin can be used to install a child server, but it will not be configured as such immediately after the installation. That is, manual configuration is required to make it interact as desired with a parent server.
- This plugin is not equipped to handle an upgrade from a previous release of GroundWork Monitor. See Upgrading to 7.1.0 from a previous version for details of the procedures for upgrading to the current release.

Test strategy

Testing of this plugin involves primarily checking the validity of install and deploy actions. It does not cover functionality within the GroundWork Monitor component itself, except for simple verification that it is running after a successful deploy action.

Types of tests included

The GroundWork installer is not incorporated directly into the Fuel plugin. Hence we must test the plugin behavior at install time both with and without the GroundWork installer being present where the plugin can find it. The plugin also verifies the GroundWork installer checksum at plugin install time, to ensure that only the expected software is in play. So testing must cover both an intact and a corrupted installer. The checksum verification also occurs again at deploy time, so that may be tested.

Deployment idempotency on a given node can also be tested, essentially to verify that an attempt to deploy on a node which already has the plugin deployed does not disturb the existing setup.

The universe is complex, so there are ways in which a deploy action might fail because the GroundWork installer fails. The simplest way to force this might be to consider that monitoring of any significant infrastructure is a resource-intensive operation, so GroundWork Monitor has certain minimum system requirements. Some of those are soft limits, which generate only warnings if the recommended resources are not available. But if the machine is sufficiently restricted that we can force the GroundWork installer to fail, that fact should be reflected in the failure of the plugin deployment.

GroundWork Monitor does not provide any OpenStack-related APIs, so it is not appropriate to use a tool such as Tempest to perform this testing. Hence none of the tests currently described here are automated.

Types of tests not included

The complexity of GroundWork Monitor precludes any realistic testing of its internal functionality, once deployed. Also, the current releases of GroundWork Monitor do not provide a fully-automated upgrade option without manual steps, so it is not possible to test a plugin-version upgrade.

Because of the potentially intense ongoing resource requirements of monitoring, we generally expect GroundWork Monitor to be deployed on its own node, or at least not on a Controller or Compute node where other OpenStack services are running that might give reason for the node to be removed. We therefore do not test for situations where a Controller or Compute node on which the plugin is deployed gets removed, with the hope that the plugin's resources would be migrated to some other node. (In fact, such migration would necessarily involve some internal reconfiguration of the GroundWork Monitor application, at a minimum to recognize its own new hostname. There are no hooks available in the product to trigger such a switchover and execute it thoroughly.) *Removal of a GroundWork node in general will cause destruction of carefully constructed monitoring setup, representing tens to thousands of man-hours of administrative investment.* This represents a loss of probably business-critical data, and can destroy the ability of the site to monitor for a considerable amount of time until the setup is restored.

Acceptance criteria

Each test should end with a simple pass/fail indication. The simplest plugin acceptance test is to check that ordinary plugin install and deploy actions with an intact GroundWork installer available, targeted to a node configured with sufficient resources, run to completion without error. Everything else is essentially a boundary case.

Test environment and infrastructure

All test cases can be run in a single Fuel environment. The tests can be run in as minimal environment as is possible to construct within Fuel. GroundWork Monitor does not depend on any external OpenStack services, so no other nodes beyond the GroundWork node are absolutely required in the environment other than whatever primary-controller and related setup is required by Fuel itself.

The central GroundWork Monitor software runs on Linux; it can monitor a variety of other operating systems. For purposes of testing, a reasonably recent release of either Ubuntu or CentOS can be used. A single copy of the GroundWork installer contains software which has been compiled to be portable across these Linux distributions, and it adapts as necessary to the differences between the platforms.

A production GroundWork node should ordinarily provide at least the following minimum resources. Larger setups are certainly acceptable, but these resources should be sufficient for simple plugin testing:

- GroundWork recommends at least 4GB of memory.
- GroundWork recommends at least 160GB of free disk space.
- GroundWork recommends at least 2 CPUs, running at 3 GHz.

That said, not all of these recommendations are rigorously enforced, For instance, testing is fine on a node with just 80GB of total disk space, including an installed OS. And it works as well on a node with 4x2.1GHz CPUs, so the individual-CPU speed is not a limiting factor, either. It's possible that the minimum-memory recommendation might be strictly enforced.

Obviously, the GroundWork node will require network access both to probe other infrastructure resources and to present its UI for configuration and monitoring-data display purposes.

Troubleshooting forensics

Certain files and directories may be important in diagnosing problems with a plugin deployment. Unless otherwise specified, they reside on the plugin target node.

`/var/log/fuel-plugin-groundwork-monitor/deployment.log`
: This file is used by the plugin as a safe place to record deployment activity. It's the first place to look if you experience a deployment failure.

`/tmp/groundwork-installer-test.log`
: This file is used by this plugin if the official GroundWork installer is not run and the installer emulator is run instead. It records a few details of the context in which the emulator was run.

`/usr/local/groundwork/Info.txt`
: This file is a normal part of the full GroundWork Monitor product. It is used by the plugin deployment scripting as a sentinel file to check for a full proper install. A dummy copy is generated by the installer emulator, with very simple content, to provide proof to the deployment scripting that the emulator ran successfully.

`/tmp/bitrock_installer*.log`
: These files are where the GroundWork Installer and Uninstaller record the details of their own progress. The first file created is unnumbered; subsequent filenames include an underscore and a process ID (so the filename will look similar to `bitrock_installer_2659.log`). Since process IDs are limited to 5 digits and wrap around, look for the most recently written file (`ls -ltr /tmp/bitrock_installer*.log`), no matter what the exact form of its filename is.

`/etc/fuel/plugins/fuel-plugin-groundwork-monitor-7.1/`
: This directory on the target node is where the Fuel Master drops the deployment files so they can be run on that node.

`/var/www/nailgun/plugins/fuel-plugin-groundwork-monitor-7.1/`
: This directory on the Fuel Master is where the complete plugin code is installed.

Product compatibility matrix

The fuel-plugin-groundwork-monitor plugin does not depend on any specific OpenStack facilities or additional plugins other than installing an operating system with external network access, with the basic memory, disk, and CPU resources noted in the previous section. It may depend on certain capabilities of the 8.0 Fuel plugin infrastructure that were not available in previous releases. It does, however, depend on a particular version of the companion GroundWork installer.

Component	***Version***
Mirantis OpenStack	*8.0*
Fuel plugin	*`fuel-plugin-groundwork-monitor-7.1-7.1.0-1.noarch.rpm`*
GroundWork installer	*`groundworkenterprise-7.1.0-br391-gw2842-linux-64-installer.run`*

Install/deploy test cases

The GroundWork Monitor deployment does not directly interact with the OpenStack infrastructure, other than to potentially monitor its resources and applications once locally configured to do so. Therefore, the tests currently listed here all refer to potential Fuel plugin install and deploy actions.

Attempt plugin install with missing GroundWork installer

The GroundWork installer is not bundled directly into the Fuel plugin, but must be available when the plugin is installed so it can be saved away for later deployment. If the GroundWork installer is not present as expected, that fact alone should cause the Fuel plugin install to fail.

Test Case ID	*install_no_groundwork_installer*
Description	*Verify that the Fuel plugin will not install on the Fuel Master if it cannot see the GroundWork installer where it expects to find it, in the `/tmp` directory.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Ensure that the `/tmp` directory contains no file or symlink with the name of the GroundWork installer. See the Product compatibility matrix section above for the exact name of this file.* 4. *Attempt to install the plugin. The attempt should fail with an error message: `ERROR: Installation of the fuel-plugin-groundwork-monitor RPM failed because the GroundWork installer is not present in the /tmp directory.`* 5. *Verify that the plugin was **not** installed successfully by running the `fuel plugins --list` command in the Fuel CLI.*
Expected Result	*Plugin is not installed on the Fuel Master node and the corresponding output appears in the CLI.*

Install plugin and deploy environment

This will be one of the most-common courses of action for using the plugin.

Test Case ID	*install_plugin_deploy_env*
Description	*Install the plugin on the Fuel Master with an intact GroundWork installer present in the `/tmp` directory. Deploy the plugin in a new environment with an intact GroundWork installer still present in the installed software.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Place it directly into the `/tmp` directory.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 12. *Find the fully-qualified hostname of the GroundWork node, and make sure it is correctly resolvable on a machine where you can run a browser.* 13. *Navigate your browser to the hostname of the GroundWork node, and verify that you see the GroundWork login splash screen.*
Expected Result	*Plugin is installed successfully on 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.*

Install symlinked plugin and deploy environment

The GroundWork installer approaches a gigabyte in size. The plugin install process expects it to reside (during the install itself) in the `/tmp` directory, to provide a standard place to look for it. But if there is not enough space in that directory on the Fuel Master, we also support parking the GroundWork installer elsewhere in the filesystem and just leaving a symlink with the same name in the `/tmp` directory, pointing to wherever the plugin physically resides.

Test Case ID	*install_symlinked_plugin_deploy_env*
Description	*Install the plugin on the Fuel Master with an intact GroundWork installer symlinked in the `/tmp` directory. Deploy the plugin in a new environment with an intact GroundWork installer still present in the installed software.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Place it in the filesystem outside of the `/tmp` directory itself and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 12. *Find the fully-qualified hostname of the GroundWork node, and make sure it is correctly resolvable on a machine where you can run a browser.* 13. *Navigate your browser to the hostname of the GroundWork node, and verify that you see the GroundWork login splash screen.*
Expected Result	*Plugin is installed successfully on 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.*

Deploy environment and install/deploy plugin

This plugin can be installed and deployed after an environment is created.

Test Case ID	*deploy_env_install_plugin*
Description	*The fuel-plugin-groundwork-monitor plugin is hot-pluggable, meaning it can be both installed on the Fuel Master and deployed to a node some time after the cluster as a whole has already been deployed. Test this capability by installing on the Fuel Master after a cluster has been deployed, and deploying on a previously uncommitted node within that existing deployed cluster.*
Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.* 2. *Create an environment in the Fuel Web UI.* 3. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, and 1 node with Telemetry-MongoDB role. Keep one node uncommitted; you will later use it for GroundWork Monitor. Make sure the uncommitted node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 4. *Finalize environment configuration (e.g., networking, node interfaces).* 5. *Run network verification check.* 6. *Deploy the cluster.* 7. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 8. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 9. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.* 10. *Install the plugin.* 11. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.* 12. *Enable the plugin in the environment, following the instructions from the Plugin Guide.* 13. *Add 1 node with GROUNDWORK_MONITOR role.* 14. *Deploy the cluster.* 15. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 16. *Find the fully-qualified hostname of the GroundWork node, and make sure it is correctly resolvable on a machine where you can run a browser.* 17. *Navigate your browser to the hostname of the GroundWork node, and verify that you see the GroundWork login splash screen.*

Expected Result	*Cluster is created and network verification check is passed.* *Plugin is installed successfully on the Fuel Master node and the corresponding output appears in the CLI.* *Plugin is enabled and configured in the Fuel Web UI.* *OSTF tests (Health Checks) are passed.* *Environment is deployed successfully.*

Install plugin with corrupted GroundWork installer

It is possible that the copy of the GroundWork installer being used for the plugin install phase has been corrupted, either through accident (e.g., network disruption during file transfer) or malice. We therefore take pains to ensure that what we are installing is actually what we expect to install.

The checksum of the GroundWork installer is verified in an RPM post-install scriplet. (We don't verify the checksum in a pre-install scriptlet, because that would constitute a TOCTTOU race condition. See https://en.wikipedia.org/wiki/Time_of_check_to_time_of_use for details.) However, due to limitations of the RPM software, there is no way for a post-install scriptlet failure to successfully signal that the installation of this RPM must be reverted. The best we can do is to emit an error message to the terminal, and prevent the corrupted GroundWork installer from becoming part of the installed plugin.

In the situation just described, the RPM will still be installed, and the Fuel system will still record the plugin as being installed, so it is available for deployments. If the plugin is then deployed to a target node, there will be no accompanying GroundWork installer. However, this situation is detected by the target-node deployment scripting, and instead an installer emulator will be run. It provides a few simple actions to pretend that the full GroundWork Monitor product was being installed. This behavior is completely non-toxic, and can be quite helpful in development testing of the rest of the plugin deployment code.

Test Case ID	*install_corrupted_groundwork_installer*
Description	*Verify that the Fuel plugin will not copy a corrupted GroundWork installer into the plugin install directory on the Fuel Master if it finds a corrupted GroundWork installer in the `/tmp` directory.*

Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.*
	2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).*
	3. *Copy the GroundWork installer to the Fuel Master node. Place it directly into the `/tmp` directory.*
	4. *Intentionally corrupt this copy of the GroundWork installer, by adding a small block of zero-value bytes to the end of the file. To do so, execute this command (all on one line):*
	`dd if=/dev/zero bs=1k count=1 >> /tmp/groundworkenterprise-7.1.0-br391-gw2842-linux-64-installer.run`
	5. *Install the plugin. The attempt should generate an error message:*
	`ERROR: The GroundWork installer fails sha256 checksum verification.`
	However, the plugin install will apparently succeed in spite of that failure.
	6. *Verify that the plugin was installed "successfully" by running the `fuel plugins --list` command in the Fuel CLI.*
	7. *Verify that the GroundWork installer is **not** present in the `/var/www/nailgun/plugins/fuel-plugin-groundwork-monitor-7.1/deployment_scripts/` directory.*

Expected Result	*Plugin is installed on the Fuel Master node but does not include the GroundWork installer as part of the plugin; the corresponding output appears in the CLI. Testing deployment of such a partially-installed plugin is covered in the separate Install plugin and deploy environment with omitted GroundWork installer test, so that action is not duplicated here.*

Install plugin with symlinked corrupted GroundWork installer

It is possible that the copy of the GroundWork installer being used for the plugin install phase has been corrupted, either through accident (e.g., network disruption during file transfer) or malice. We therefore take pains to ensure that what we are installing is actually what we expect to install. This applies in the case of a symlinked installer as well. See the discussion in the previous test for more detail on this scenario.

Test Case ID	*install_symlinked_corrupted_groundwork_installer*
Description	*Verify that the Fuel plugin will not install on the Fuel Master if it finds a corrupted GroundWork installer symlinked into the `/tmp` directory.*

Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.*
	2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).*
	3. *Copy the GroundWork installer to the Fuel Master node. Place it in the filesystem outside of the `/tmp` directory itself and place a symlink to it in the `/tmp` directory using the same filename for the symlink.*
	4. *Intentionally corrupt this copy of the GroundWork installer, by adding a small block of zero-value bytes to the end of the file. To do so, execute this command (all on one line):*
	`dd if=/dev/zero bs=1k count=1 >> /tmp/groundworkenterprise-7.1.0-br391-gw2842-linux-64-installer.run`
	5. *Install the plugin. The attempt should generate an error message:*
	`ERROR: The GroundWork installer fails sha256 checksum verification.`
	However, the plugin install will apparently succeed in spite of that failure.
	6. *Verify that the plugin was installed "successfully" by running the `fuel plugins --list` command in the Fuel CLI.*
	7. *Verify that the GroundWork installer is **not** present in the `/var/www/nailgun/plugins/fuel-plugin-groundwork-monitor-7.1/deployment_scripts/` directory.*

Expected Result	*Plugin is installed on the Fuel Master node but does not include the GroundWork installer as part of the plugin; the corresponding output appears in the CLI. Testing deployment of such a partially-installed plugin is covered in the separate Install plugin and deploy environment with omitted GroundWork installer test, so that action is not duplicated here.*

Install plugin and deploy environment with omitted GroundWork installer

Once the plugin has been installed on the Fuel Master, it is possible for the associated GroundWork installer to go missing before the plugin is deployed on a node. This is unlikely to happen by accident, but it can happen on purpose. If the GroundWork installer is removed from its installed location on the Fuel Master, a deploy action must respond appropriately. In our case, we allow the plugin to be deployed under such a condition. In this situation, a fallback installer-emulator script will be run on the deployed node instead of the GroundWork installer. Since the GroundWork installer itself normally runs for many minutes while being deployed, having a fast-acting emulator invoked in this situation allows a much quicker development and testing cycle for the Fuel plugin itself (or would, if OpenStack-related Fuel actions did not completely dominate the deployment time), irrespective of the ultimate behavior of the official GroundWork installer payload.

Test Case ID	*install_plugin_deploy_no_groundwork_installer*
Description	*Verify that the Fuel plugin will still deploy if it was previously installed on the Fuel Master but the GroundWork installer was subsequently removed from the installed distribution. This test case also covers the situation that results from installing the plugin on the Fuel Master with a corrupted accompanying GroundWork installer. In either case, at deployment time an installer emulator will be run so the rest of the plugin deployment actions can still be tested.*

Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.* 6. *Intentionally remove the GroundWork installer from the installed plugin. To do so, execute the following command (all on one line):* `rm /var/www/nailgun/plugins/fuel-plugin-groundwork-monitor-7.1/deployment_scripts/groundworkenterprise-7.1.0-br391-gw2842-linux-64-installer.run` 7. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 8. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role.* 9. *Finalize environment configuration (e.g., networking, node interfaces).* 10. *Run network verification check.* 11. *Deploy the cluster.* 12. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 13. *On the target node:* a. *Check that the* `/etc/fuel/plugins/fuel-plugin-groundwork-monitor-7.1/` *directory does* ***not*** *contain a copy of the actual GroundWork installer.* b. *Check that only the* `/usr/local/groundwork/Info.txt` *file is present in that directory.* c. *Check that the* `/tmp/groundwork-installer-test.log` *file is present.* d. *Check that the* `/var/log/fuel-plugin-groundwork-monitor/deployment.log` *file exists, but does* ***not*** *contain an* `http` *URL.*
Expected Result	*Plugin is installed successfully on 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.* *Remains from running the installer emulator are present on the target node.*

Install plugin and attempt deployment with corrupted GroundWork installer

Once the plugin has been installed on the Fuel Master, it is possible for the associated GroundWork installer to become corrupted before the plugin is deployed on a node. This is unlikely to happen by accident, but it could happen on purpose, if it was replaced by some Trojan horse. If the GroundWork installer is still present but corrupted in its installed location on the Fuel Master, a deploy action must respond appropriately. In our case, the corrupted file should be detected, and the deploy action should fail.

Test Case ID	*install_plugin_deploy_corrupted_groundwork_installer*
Description	*Verify that the Fuel plugin will fail to deploy if it was previously installed on the Fuel Master but the GroundWork installer in the installed distribution was subsequently corrupted.*

Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.*
	2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).*
	3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.*
	4. *Install the plugin.*
	5. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.*
	6. *Intentionally corrupt the GroundWork installer in the installed plugin, by adding a small block of zero-value bytes to the end of the file. To do so, execute the following command (all on one line):*
	`dd if=/dev/zero bs=1k count=1 >> /var/www/nailgun/plugins/fuel-plugin-groundwork-monitor-7.1/deployment_scripts/groundworkenterprise-7.1.0-br391-gw2842-linux-64-installer.run`
	7. *Verify that the GroundWork installer you just modified has execute permissions, as it should have had before you corrupted it. (If you chose to corrupt the file in some other manner, this might not be the case. And if the installer is not executable, then when the plugin is deployed, instead of detecting a checksum failure as we expect it to do in the following steps, the deployment scripting will happily ignore the GroundWork installer and simply run the installer emulator instead.)*
	8. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.*
	9. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role.*
	10. *Finalize environment configuration (e.g., networking, node interfaces).*
	11. *Run network verification check.*
	12. *Deploy the cluster.*
	13. *Observe that deployment of this plugin fails.*
	14. *On the target node, check that the* `/var/log/fuel-plugin-groundwork-monitor/deployment.log` *file exists, and contains this message:*
	`ERROR: The GroundWork installer fails sha256 checksum verification.`
	15. *Run OSTF, as desired. However, no GroundWork-specific tests are available.*

Expected Result	*Plugin is installed successfully on 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 (perhaps), but this plugin fails deployment.*

Install plugin and attempt deployment on an already-deployed node

Once the plugin has been deployed on a node, it might be possible for the Fuel administrator to accidentally attempt to re-deploy it on the same node. This situation should be detected and the new deploy action should fail, so the existing monitoring configuration data is not lost.

This is currently a logical test description, which appears to be impossible to execute. I don't know how you would use Fuel to re-deploy on the same node, since that would effectively require that you assign the same GROUNDWORK_MONITOR role to the node which already has that node.

Test Case ID	*install_plugin_deploy_on_deployed_node*
Description	*Verify that the Fuel plugin will fail to deploy if it was previously deployed on the same node.*
Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 12. *Attempt to re-deploy the plugin on the same GROUNDWORK_MONITOR node. (I don't know how to do this.) Observe that the attempt fails.*
Expected Result	*Plugin is installed successfully on 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.* *Attempting to re-deploy the plugin on the same node fails.*

Install plugin and attempt deployment on an underresourced node

GroundWork Monitor requires certain minimum resources on the deployed node. If they are not available, this situation should be detected and the new deploy action should fail, to limit the potential for later confused troubleshooting.

Test Case ID	*install_plugin_deploy_underresourced_node*
Description	*Verify that the Fuel plugin will fail to deploy if deployment is attempted on a node with sufficiently inadequate resources to run GroundWork Monitor.*
Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node does not meet the minimum resource requirements, as listed in the Test environment and infrastructure section. Perhaps the easiest way to do this is to provision the node with only 2GB memory and 80 GB of disk space.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Observe that the plugin deployment fails.* 12. *Run OSTF, as desired. However, no GroundWork-specific tests are available.*
Expected Result	*Plugin is installed successfully on 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 (perhaps), but this plugin fails deployment.*

Fuel create mirror and update (setup) of core repos

This test has to do with some potential undesired interaction between Fuel and its management of already-deployed nodes. See https://bugs.launchpad.net/fuel/+bug/1527330 for details of the problems that triggered the inclusion of this test case.

Test Case ID	*fuel_create_mirror_update_core_repos*
Description	*Verify that an already-deployed GroundWork Monitor node will remain effectively undisturbed by changes elsewhere in the Fuel subsystem.*

Steps	1. *Verify that the plugin is not already installed by running the* `fuel plugins --list` *command in the Fuel CLI.*
	2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).*
	3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the* `/tmp` *directory, or place it elsewhere in the filesystem and place a symlink to it in the* `/tmp` *directory using the same filename for the symlink.*
	4. *Install the plugin.*
	5. *Ensure that the plugin is installed successfully by running the* `fuel plugins --list` *command in the Fuel CLI.*
	6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.*
	7. *Add 2 nodes with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.*
	8. *Finalize environment configuration (e.g., networking, node interfaces).*
	9. *Run network verification check.*
	10. *Deploy the cluster.*
	11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.*
	12. *Go to each controller / compute / storage / etc. node using a terminal shell, and get the process IDs of services which were launched by plugins. Store them for later reference. In the case of a GroundWork node, the following command will generate a reasonable list with just the most-important data.*
	`ps -u nagios,postgres -C svscan -C supervise -C syslog-ng -C snmptrapd --forest -o pid,user,comm`
	13. *Launch the following command on the Fuel Master node:*
	`fuel-createmirror -M`
	14. *Launch the following command on the Fuel Master node:*
	a. *For MOS 8.0:* `fuel --env <ENV_ID> node --node-id <NODE_ID1> <NODE_ID2> <NODE_IDn> --tasks setup_repositories`
	15. *Go to each controller / plugin / storage node using a terminal shell, and check if all of the plugin's services noted earlier are still alive and have not changed their process IDs.*
	16. *Check with the* `fuel nodes` *command that all nodes remain in ready status.*
	17. *Rerun OSTF.*

Expected Result	*Plugin is installed successfully on 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.* *Each plugin's services should not be restarted when the Fuel Master tasks are executed. If they are restarted as some exception, this information should be added to the plugin's User Guide. In the case of a GroundWork node, most of the monitoring processes should have remained stable through this exercise. The exceptions might be certain transient processes run by cron jobs and the like, that might happen to be caught by probing at a particular moment.* *Cluster (nodes) should remain in ready state.* *OSTF test should be passed on rerun.*

Attempt plugin uninstall in a deployed environment

It is generally a **Very Bad Idea** to go casually removing this plugin within an active deployed environment. That's because of the significant administrative investment the site has probably made in the monitoring configuration data of GroundWork Monitor. It would be a **Very Good Idea** if there were some way for a deployment of this plugin to mark the node as requiring extraordinary measures (on the order of two-person, simultaneously activated launch keys for a nuclear strike) before such a node can be undeployed.

That notion goes beyond what is being tested here, which is simply that an attempted uninstall on the Fuel Master should fail if the plugin is currently deployed on some node in some environment. As far as I can tell, the current uninstall-failure feature checked by the following test appears to be entirely controlled by Fuel itself, and not by some particular feature specified within the plugin.

Test Case ID	*uninstall_plugin_with_deployed_env*
Description	*Verify that the Fuel plugin will fail to uninstall on the Fuel Master if it is currently deployed in the environment.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the `/tmp` directory, or place it elsewhere in the filesystem and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 12. *Attempt to uninstall the plugin on the Fuel Master by running the following command (all on one line):* `fuel plugins --remove fuel-plugin-groundwork-monitor==7.1.0` 13. *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 plugin in a non-deployed environment

If you really wish to uninstall the plugin completely from the Fuel Master, it is necessary to first delete all nodes on which it is already deployed, then disable the plugin in all environments controlled by the Fuel Master. *But that first step, of removing the plugin from the nodes on which it is deployed, must never be taken casually, as noted earlier.* In some future release of Mirantis OpenStack, there ought to be a feature that allows the plugin to block undeployment from a deployed node without special extra actions being taken by the administrator, to protect the precious asset which is the existing GroundWork configuration data. And then that feature ought to be tested here.

Test Case ID	*uninstall_plugin*
Description	*Completely uninstall the Fuel plugin from the Fuel Master. This is supposed to require first removing all active associations the plugin has to every environment managed by the Fuel Master. For simplicity, we don't fully create any associations.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the `/tmp` directory, or place it elsewhere in the filesystem and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role.* 8. *Finalize environment configuration (e.g. networking, nodes interfaces).* 9. *Run network verification check.* 10. *Delete the listed environment without first deploying it.* 11. *Uninstall the plugin on the Fuel Master by running the following command (all on one line):* `fuel plugins --remove fuel-plugin-groundwork-monitor==7.1.0`
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`").*

Uninstall plugin after disabling in environment

This test is similar to the earlier Uninstall plugin in a non-deployed environment test. See that test for more information about installing the plugin. However, we include this extra test because it demonstrates some issues with the Fuel plugin-management infrastructure (not this plugin itself).

Test Case ID	*uninstall_plugin_after_disabling*
Description	*Completely uninstall the Fuel plugin from the Fuel Master, and verify that it completely disappears from the Fuel UI after being manually disabled in an environment which was never fully deployed.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the `/tmp` directory, or place it elsewhere in the filesystem and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Navigate to Settings > Other in the new environment, disable the plugin (uncheck the checkbox), and Save Settings.* 8. *Navigate to Dashboard (just so you're not sitting in Other for the next step).* 9. *Uninstall the plugin on the Fuel Master by running the following command (all on one line):* `fuel plugins --remove fuel-plugin-groundwork-monitor==7.1.0` 10. *Verify that the plugin is no longer installed by running the `fuel plugins --list` command in the Fuel CLI.* 11. *Navigate to Settings > Other. Observe that in spite of the fact that the plugin is now gone from the Fuel Master, this environment still believes it is available, contrary to expectations.* 12. *Enable the plugin by checking the checkbox, and Save Settings. Observe that the Save succeeds, even though the plugin does not exist in the Fuel Master.*
Expected Result	*Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.* *Cluster is created.* *Plugin is initially available in the Fuel Web UI.* *When uninstalling the plugin, no plugin-related elements should be left in the environment (e.g. UI elements disappear, Nailgun database is restored to the default state, no output for command "`fuel plugins --list`").*

Uninstall plugin from never-enabled environment

This test is similar to the earlier Uninstall plugin in a non-deployed environment test. See that test for more information about installing the plugin. However, we include this extra test because it demonstrates some issues with the Fuel plugin-management infrastructure (not this plugin itself).

Test Case ID	*uninstall_plugin_when_never_enabled*
Description	*Completely uninstall the Fuel plugin from the Fuel Master, and verify that it completely disappears from the Fuel UI in an environment where it was never enabled in the first place.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Create an environment in the Fuel Web UI, before the plugin is installed.* 3. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 4. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the `/tmp` directory, or place it elsewhere in the filesystem and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 5. *Install the plugin.* 6. *Ensure that plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 7. *Navigate to Settings in the new environment. Observe that no Other category is available, even though the plugin now exists in the Fuel Master, presumably because the plugin was not available when this environment as a whole was created.* 8. *Navigate to Nodes > Add Node. (Don't actually add any nodes; visiting this screen seems to trigger recognition in this environment of the new plugin, as a side effect.)* 9. *Navigate to Settings. Observe that the Other category now appears.* 10. *Uninstall the plugin on the Fuel Master by running the following command (all on one line):* `fuel plugins --remove fuel-plugin-groundwork-monitor==7.1.0` 11. *Verify that the plugin is no longer installed by running the `fuel plugins --list` command in the Fuel CLI.* 12. *Navigate to Settings > Other. Observe that in spite of the fact that the plugin is now gone from the Fuel Master, this environment still believes it is available, contrary to expectations.* 13. *Enable the plugin by checking the checkbox, and Save Settings. Observe that the Save succeeds, even though the plugin does not exist in the Fuel Master.*

Expected Result	*Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI.* *Cluster is created.* *Plugin is initially available in the Fuel Web UI.* *When uninstalling the plugin, no plugin-related elements should be 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 test cases

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

The Fuel Master node upgrade testing

The fuel-plugin-groundwork-monitor plugin is currently only supported on Mirantis OpenStack 8.0. In the future, when it is updated to also run under later releases of MOS, this section will be updated to include Fuel Master upgrade testing. In general, as long as all the installed plugin files are left intact by the Fuel upgrade, and as long as the plugin is internally specified as supporting the new release of Fuel, there should be no problem with the upgrade. If the new Fuel version changes certain critical details, such as the plugin deployment directory on the deployed node, that will require some changes to the plugin and corresponding test cases.

Update the plugin to minor version in the deployed environment

The Fuel Plugin Framework allows updating the plugin to a new 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 to learn about the procedure.

This being the first iteration of the fuel-plugin-groundwork-monitor plugin, there is no support included for minor-version upgrades. This scenario should be revisited in the future.

More generally, no support will be provided until we have a full explanation from Mirantis as to what a `fuel plugins --update` *command actually does. The link above provides only a template command, with no information whatsoever on what actually happens when you execute it.*

Currently, the GroundWork installer is not equipped to handle an upgrade from a previous release of GroundWork Monitor in a completely-automated fashion. (Some steps are automated, but a significant amount of application-specific manual work is also required, both before and after the upgrade.) Hence we do not provide any deployed-plugin-upgrade tests at this time. For the time being, an attempted deployment of the plugin on a node that already has the plugin deployed should fail by recognizing that GroundWork Monitor is already installed on the node, without affecting the deployed software. That test case is already covered earlier in this document.

Apply maintenance updates to a deployed environment

Mirantis OpenStack features the ability to receive patches via the common flow called Maintenance Updates. See https://dcs.mirantis.com/openstack/fuel/fuel-8.0/maintenance-updates.html for general information about MOS MUs, and links from there to the process for applying such an update. The test here has to do with some potential undesired interaction between Fuel and its management of already-deployed nodes. It has nothing to do with upgrading the plugin itself.

Test Case ID	*apply_maintenance_update*
Description	*Verify that an already-deployed GroundWork Monitor node will remain effectively undisturbed by changes elsewhere in the Fuel subsystem, specifically an MOS Maintenance Update.*
Steps	1. *Verify that the plugin is not already installed by running the `fuel plugins --list` command in the Fuel CLI.* 2. *Copy the plugin to the Fuel Master node (please refer to the User Guide for more details).* 3. *Copy the GroundWork installer to the Fuel Master node. Either place it directly into the `/tmp` directory, or place it elsewhere in the filesystem and place a symlink to it in the `/tmp` directory using the same filename for the symlink.* 4. *Install the plugin.* 5. *Ensure that the plugin is installed successfully by running the `fuel plugins --list` command in the Fuel CLI.* 6. *Create an environment with enabled plugin in the Fuel Web UI, following the instructions from the Plugin Guide.* 7. *Add 1 node with Controller role, 1 node with Compute role, 1 node with Storage-Cinder role, 1 node with Telemetry-MongoDB role, and 1 node with GROUNDWORK_MONITOR role. Make sure the GROUNDWORK_MONITOR node addresses the minimum resource requirements, as listed in the Test environment and infrastructure section.* 8. *Finalize environment configuration (e.g., networking, node interfaces).* 9. *Run network verification check.* 10. *Deploy the cluster.* 11. *Run OSTF, as desired. However, no GroundWork-specific tests are available.* 12. *Once the environment is deployed, apply Mirantis OpenStack maintenance updates, following the published instructions (see https://docs.mirantis.com/openstack/fuel/fuel-8.0/maintenance-updates.html and links from there to details on particular updates and the processes for applying updates).* 13. *Check that plugin services continue running. In the case of a GroundWork node, as a simple check you should still be able to log into its Web UI and observe current monitoring data.* 14. *Make sure that all nodes are in ready state (you can check this in the Fuel UI, or via the `fuel nodes` command in a terminal window on the Fuel Master node), and that no regression is observed.* 15. *Run OSTF checks.*

Expected Result	*Plugin is installed successfully on 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 are not restarted). In the case of a GroundWork node, if you check in detail, most of the monitoring processes should have remained stable through this exercise. The exceptions might be certain transient processes run by cron jobs and the like, that might happen to be caught by probing at a particular moment.* *Cluster (nodes) remain in the fully operational state after applying Maintenance Updates.*

Appendix

The following external resources are available.

Item	***Resource***
1	GroundWork Monitor capabilities are described on the GroundWork web site.
2	The GroundWork installer can be obtained from GroundWork, Inc., starting from the GroundWork web site.
3	The fuel-plugin-groundwork-monitor documentation describes this plugin, and how to install and deploy it.