The Murano Plugin for Fuel Documentation

Release 1.2.0

Mirantis Inc.

December 19, 2016

CHAPTER ONE

OVERVIEW

Murano plugin for Fuel is a plugin to install the Application Catalog Service for OpenStack. Murano plugin provides flexibility for the OpenStack Application Catalog service by supporting multiple releases and versions (for some Fuel version). Also, the plugin provides the capability to detach Murano from the controller node as a separate node.

Software prerequisites

To use Murano plugin, verify that your environment meets the following prerequisites:

Prerequisites	Version/Comment
Fuel	9.0

Limitations

- Murano plugin is known to conflict with Contrail plugin. The reason is that both `murano-api` and `contrail-api` services use 8082 as their default port. To resolve this, please specify a custom port for `contrail-api` in Contrail plugin.

See also release-notes for any release-specific known issues.

Licenses

Component	**License**
Murano plugin	Apache 2.0

References

For more information about Murano plugin for Fuel described in this document, see:

- Specification
- Launchpad project
- GitHub project

CHAPTER

TWO

INSTALLING THE MURANO PLUGIN

Before you install the Murano plugin, verify that your environment meets the requirements described in *Software prerequisites*. You must have the Fuel Master node installed and configured before you can install the plugin. Typically, you install a Fuel plugin before you deploy an OpenStack environment.

To install the Murano plugin:

1. Download the Murano plugin from the Fuel Plugins Catalog.
2. Copy the plugin `.rpm` package to the Fuel Master node:

   Example:

   ```
   # scp detach-murano-<plugin_version> root@fuel-master:/tmp
   ```

3. Log in to the Fuel Master node CLI as root.
4. Install the plugin by typing:

   ```
   # fuel plugins --install detach-murano-<plugin_version>
   ```

5. Verify that the plugin is installed correctly:

   ```
   # fuel plugins
   id | name           | version | package_version
   ---|----------------|---------|----------------
   1  | detach-murano  | 1.2.0   | 4.0.0
   ```

6. Proceed to *Configuring the Murano plugin*.

CHAPTER

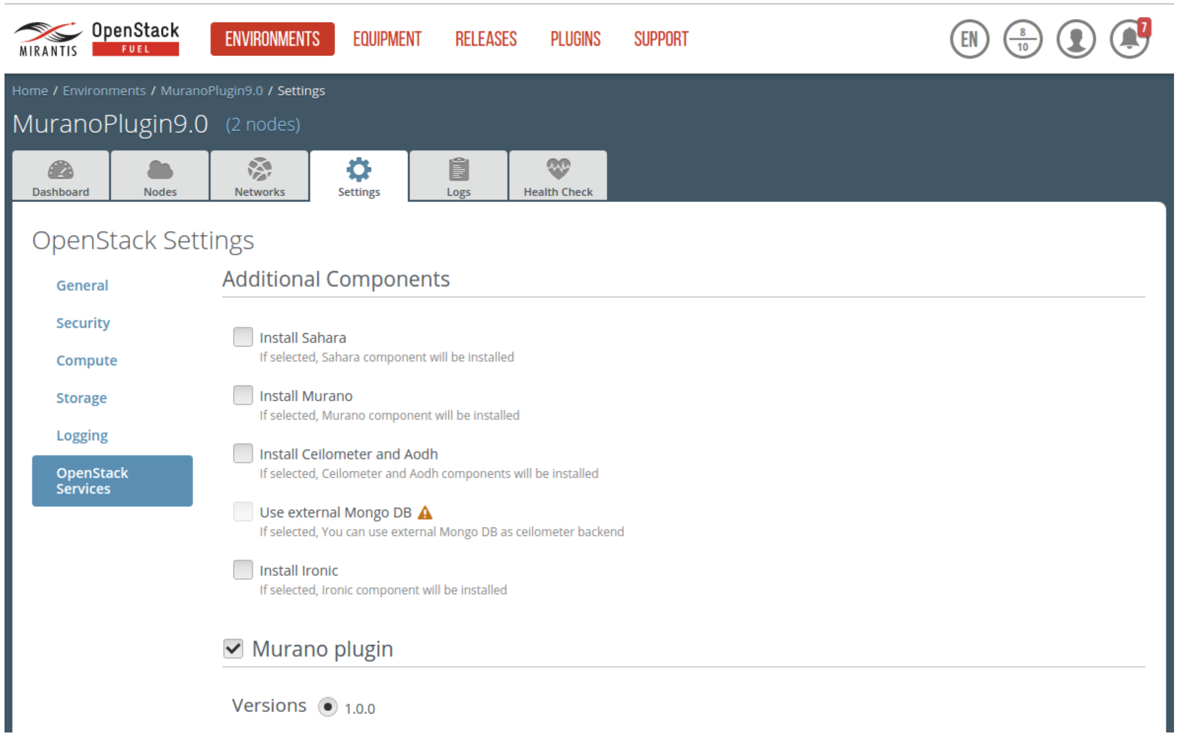
THREE

CONFIGURING THE MURANO PLUGIN

Once the Murano plugin is installed, follow the instruction below to create an OpenStack environment with Murano services.

To configure the plugin:

1. Create an OpenStack environment as described in the Fuel User Guide or use an existing one.
2. To enable the Murano plugin, navigate to the *Environments* tab and select the *Murano plugin* checkbox:



- [ ] Install Sahara
  If selected, Sahara component will be installed
- [ ] Install Murano
  If selected, Murano component will be installed
- [ ] Install Ceilometer and Aodh
  If selected, Ceilometer and Aodh components will be installed
- [ ] Use external Mongo DB
  If selected, You can use external Mongo DB as ceilometer backend
- [ ] Install Ironic
  If selected, Ironic component will be installed

- [x] Murano plugin

Versions ● 1.0.0

3. Configure the Murano plugin as required selecting the following available settings.

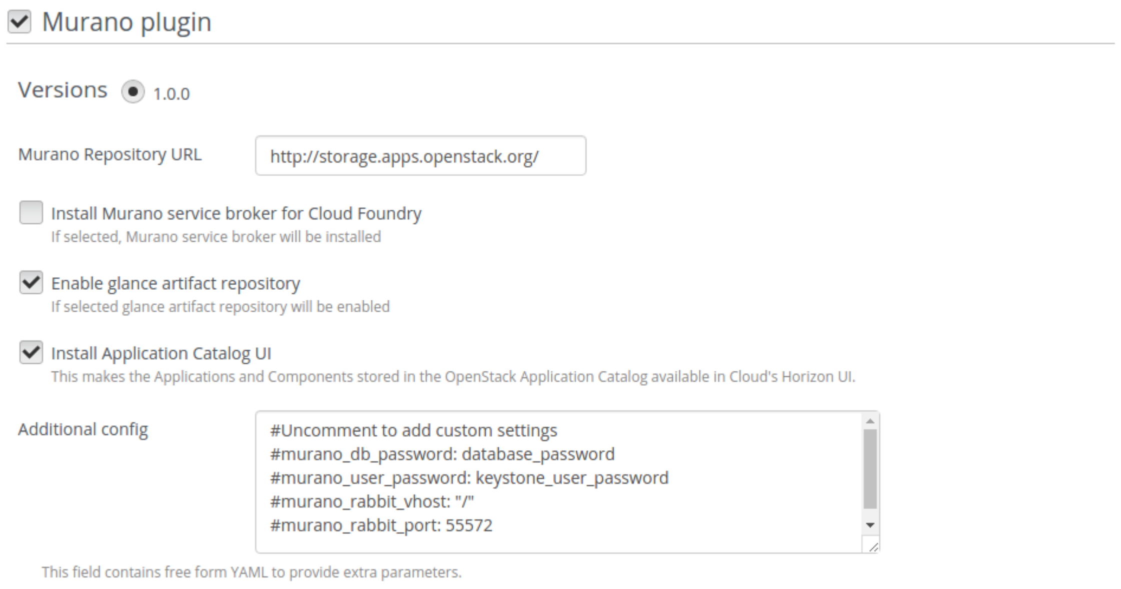
   Murano Repository URL specifies the Murano applications repository to import a package.

   Install Murano service broker for Cloud Foundry enables Cloud Foundry Service Broker API. Cloud Foundry is PaaS which supports full lifecycle from initial development through all testing stages to deployment. Most well-known Cloud Foundry flavors are Cloud Foundry OSS, Pivotal Cloud Foundry, and Pivotal Web Services. If Cloud Foundry Service Broker API is enabled, Murano apps are available at Cloud Foundry as services.

   Enable glance artifact repository enables usage of new Glance API, which stores not only the VM images but also data assets and their metadata for other OpenStack projects. This specification defines the usage of this feature in Murano. Therefore, Murano may store its packages in Glance and benefit from all its features.

Install Application Catalog UI enables the OpenStack Community App Catalog, which will help make applications available on OpenStack cloud by providing a community-driven catalog containing Glance images, Heat templates, and Murano applications.

Additional config allows specifying Murano end-user credentials, such as names of Murano users (db, keystone, rabbit), passwords, and others.



✓ Murano plugin

Versions ● 1.0.0

Murano Repository URL: http://storage.apps.openstack.org/

☐ Install Murano service broker for Cloud Foundry
If selected, Murano service broker will be installed

✓ Enable glance artifact repository
If selected glance artifact repository will be enabled

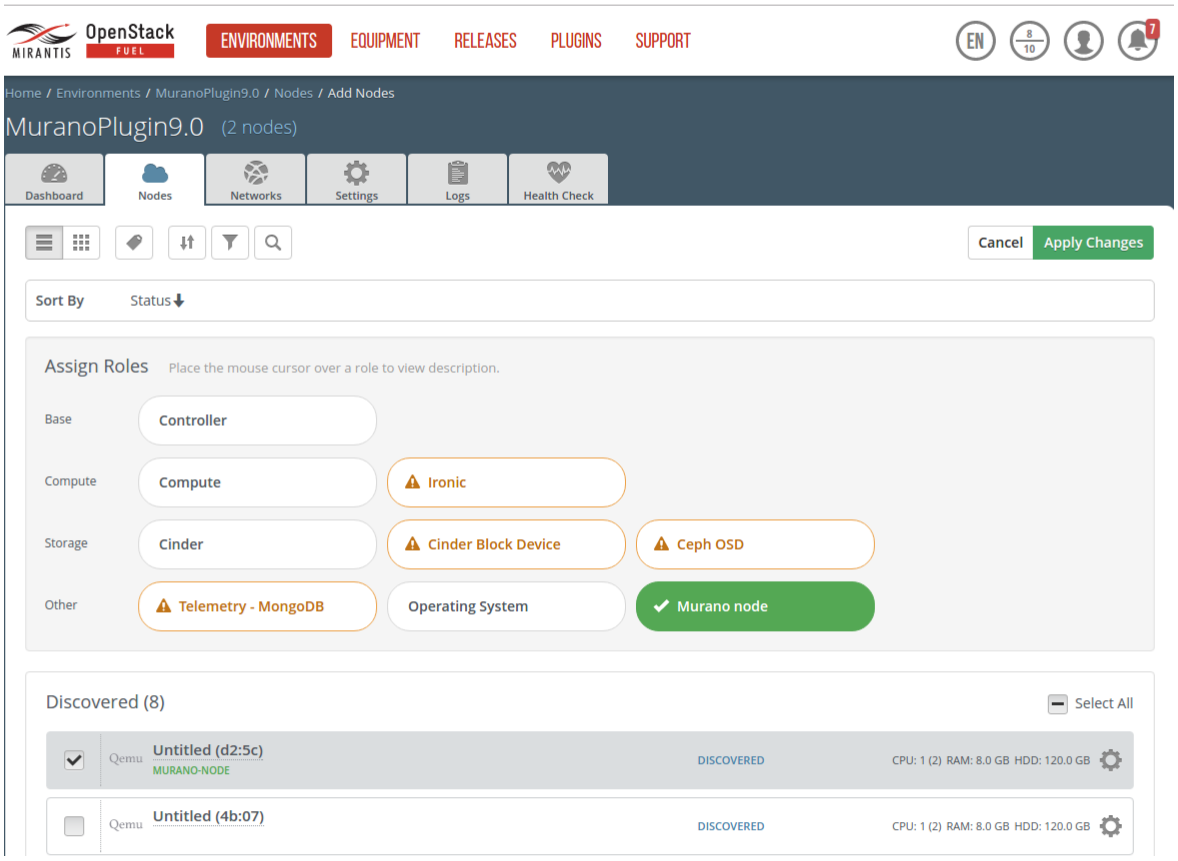
✓ Install Application Catalog UI
This makes the Applications and Components stored in the OpenStack Application Catalog available in Cloud's Horizon UI.

Additional config

```
#Uncomment to add custom settings
#murano_db_password: database_password
#murano_user_password: keystone_user_password
#murano_rabbit_vhost: "/"
#murano_rabbit_port: 55572
```

This field contains free form YAML to provide extra parameters.

4. To deploy Murano services on a particular OpenStack node, assign the *Murano node* role to this node in the *Nodes* tab. There are no restrictions to combining the Murano role with other node roles. Otherwise, Murano services will be deployed on controller nodes.



5. Configure your environment as described in the Fuel User Guide.

Now, you can deploy your OpenStack environment with Murano as described in the Fuel user Guide.

CHAPTER

FOUR

UPDATING THE MURANO PLUGIN

To update the plugin, follow the steps described in the Fuel Plugins Guide.

CHAPTER

FIVE

RELEASE NOTES

1.2.0

Murano Dashboard has been renamed to App Catalog and now allows seamless integration and single panel structure with App Catalog UI dashboard.

New Features

- CLI command :command:`env-templat-create-env` now supports `--region` flag
- Murano dashboard has been renamed to App Catalog, monolithic config file has been split into multiple small files. Every such file defines either a panel group or adds general murano-related settings to horizon.
- /environments/ENV_ID/model/PATH endpoint added. GET request responds with the subsection of ENV_ID's object model located in its PATH. PATCH request applies json-patch from request body to ENV_ID's model. It does not contain PATH in the URL.
- Support for environment model edit API was added
- New Murano CLI command `murano environment-model-show <ID> [--path <PATH>] [--session-id <SESSION_ID>]`
- New Murano CLI command `murano environment-model-edit <ID> <FILE> --session-id <SESSION_ID>`
- New OSC command `openstack environment model show <ID> [--path <PATH>] [--session-id <SESSION_ID>]`
- New OSC command `openstack environment model edit <ID> <FILE> --session-id <SESSION_ID>`
- io.murano.system.HeatStack.push can be called with async => true flag for asynchronous push
- Added a MetadataAware mixin class capable to retrieve the metadata attributes from the implementing objects and all its parents.
- Added a `metadata()` yaql function to retrieve the meta information about the object, stored in the ”?/metadata” section of object model.
- New OSC command `package list`.

Known Issues

- Deploying Cloud Foundry integration service with this release of `fuel-plugin-murano` does not work correctly.

- Community App Catalog UI panels, delivered as part of `fuel-plugin-murano` do not work correctly with this release.

Upgrade Notes

- To upgrade to Newton version of app catalog you need to remove old `_50_murano.py` config file, that defined in murano dashboard. Be sure to also remove any .pyc and .po files. After that you need to copy all new config files from `muranodashboard/local/enabled/*.py` to `openstack_dashboard/local/enabled/` and restart horizon

Bug Fixes

- added default rules to NeutronSecurityGroupManager to avoid error if *createDefaultInstanceSecurity-GroupRules()* method isn't extended in inheritor and SecurityGroups isn't created in application with call of *addGroupIngress()* method.
- The issue with adding already deployed components to environment via dropdown is fixed with applying changes for the new type format.
- It is now possible to make a PUT request with body equal to '[]' to '/environments/<env_id>/services' end-point. This will result in removing all apps from current session. This allows deleting the last application from environment from CLI.
- Previously Cinder Volumes created in MuranoPL were not released correctly on object destruction. The issue is now fixed.

Other Notes

- Bumped the RUNTIME_VERSION attribute to 1.5

1.1.0

New Features

- New on-request garbage collector for MuranoPL objects were implemented. Garbage collection is triggered by io.murano.system.GC.collect() static method. Garbage collector destroys all object that are not reachable anymore. GC can handle objects with cross-references and isolated object graphs. When portion of object model becomes not reachable it destroyed in predictable order such that child objects get destroyed before their parents and, when possible, before objects that are subscribed to their destruction notifications.
- Internally, both pre-deployment garbage collection (that was done by comparision of `Objects` and `ObjectsCopy`) and post-deployment orphan object collection are now done through the new GC.
- io.murano.system.GC.isDoomed() static method was added. It can be used within the `.destroy` method to test if other object is also going to be destroyed.
- io.murano.system.GC.isDestroyed() static method was added. It checks if the object is destroyed and thus no methods can be invoked on it.
- Ability to load package from directory was added. If specified directory contains all the needed files then package will be imported as usual.

Known Issues

- Deploying Cloud Foundry integration service with this release of `fuel-plugin-murano` does not work correctly.
- If you update from murano plugin version 1.0.0 with Murano deployed on controller to 1.1.0 and add deploy murano on a separate node Murano will not work correctly.
- Community App Catalog UI panels, delivered as part of `fuel-plugin-murano` do not work correctly with this release.

Bug Fixes

- It was impossible to use *–owned* flag when Glare was used to filter packages. This issue is fixed now.
- CLI flag `--is-public` had no effect when importing a package with 'glare' package service. This flag now correctly marks the package public on upload.
- It was possible to import the same murano package from the CLI into the same project multiple times if glare was used and the package was imported as a private one. The issue is now fixed.
- Murano engine no longer logs methods `string()`, `json()`, and `yaml()` of the 'io.murano.system.Resources' class. This is done to prevent UnicodeDecodeError's when transferring binary files to murano agent.

Other Notes

- Murano Dashboard relies on Glance v1 API for image uploads. In case it is not available an error will be shown and all the image-related functionality will be unavailable.

1.0.0

New Features

- Added a new manifest format 1.4.0. Introduced the `Scope` keyword for class methods to declare a method's accessibility from outside through the API call.
- New OSC command **environment list [–all-tenants]**
- New OSC command **environment show <ID or NAME> [–session-id <SESSION_ID>] [–only-apps]**
- New OSC command **environment rename <ID> <NAME>**
- New OSC command **environment session create <ID>**
- Implemented the capability for API endpoint `/catalog/packages` to filter 'id', 'category', 'tag' properties using the 'in' operator. An example of using the 'in' operator for 'id' is 'id=in:id1,id2,id3'. Added this filter using the syntax that conforms to the latest guidelines from the OpenStack API-WG.
- Added the `timeout` parameter to `runCommand` and `putFile` methods of the `io.murano.configuration.Linux` class.
- New OSC command **category list**
- New OSC command **category show <ID>**
- New OSC command **category create <NAME>**

- New OSC command **category delete <ID> [<ID> ...]**
- New OSC command **environment create [–join-net-id <NET_ID>] [–join-subnet-id <SUBNET_ID>] [–region <REGION_NAME>] <ENVIRONMENT_NAME>**
- New OSC command **environment delete [–abandon] <NAME_or_ID> [<NAME_or_ID> ...]**
- Added support for class/method JSON-schema API.
- New murano CLI command **murano class-schema [–package-name PACKAGE_NAME] [–class-version PACKAGE_VERSION] <CLASS> [<METHOD> [<METHOD> ...]]**
- New OSC command **openstack class-schema [–package-name PACKAGE_NAME] [–class-version PACKAGE_VERSION] <CLASS> [<METHOD> [<METHOD> ...]]**
- Added support for static actions API.
- New Murano CLI command **murano static-action-call [–arguments [<KEY=VALUE> [<KEY=VALUE> ...]]] [–package-name <PACKAGE>] [–class-version CLASS_VERSION] <CLASS> <METHOD>**
- New OSC command **openstack static-action call [–arguments [<KEY=VALUE> [<KEY=VALUE> ...]]] [–package-name <PACKAGE>] [–class-version CLASS_VERSION] <CLASS> <METHOD>**
- Added Cinder volume classes to the core library. Now, it s possible to create a new volume from various sources or use an existing volume. Also, it is possible to attach volumes to instances and boot instances from volumes.
- Added the `driver` configuration option to the `networking` group. It allows to explicitly select the networking driver. It supports 'neutron' and 'nova' options. If set to `None` (default), Murano attempts to use 'neutron' if available, 'nova' otherwise. The change is backward compatible.
- Added the *`--dep-exists-action`* option to murano **package-import** and **bundle-import** commands. It allows specifying a different default action for an existing main package and existing dependencies. In case of specifying only *`--exists-action`*, the action also applies to dependencies.
- Added the `DISPLAY_MURANO_REPO_URL` setting that is used as a user-visible link to `apps.openstack.org` or any other murano applications repository.
- Added the `description_text` field to environment and environment templates database tables and respective API objects.
- Introduced a new MuranoPL `io.murano.system.GC` class. Now, MuranoPL garbage collector can be used to set up destruction dependencies between murano objects. If object Foo is subscribed to object Bar's destruction, it will be notified through a specific handler. If both Foo and Bar are going to be destroyed during one execution session, Foo will be destroyed after Bar. You can omit the handler, in this case destruction order will also be preserved. Handler can be a static or a usual function.
- Implemented the capability for the helper methods of Linux class to run concurrently if executed for different VM agents.
- Added the capability to execute actions (delete, abandon or deploy) on multiple selected environments.
- Added the following meta-classes to the core library - `Title` `Description` `HelpText` `Hidden` `Section` `Position` `ModelBuilder`. These classes will later be used to implement dynamic object model generation.
- Added an overload of the `new` function - `new($model, $owner)`. It loads a complete object graph in a single call. Objects in the model can have cross references. In that case, this is the only way to instantiate the graph. Objects may be specified either in object model format (with '?' attribute or in MuranoPL format (used for Meta definitions).
- The contract `class()` now uses the same approach to load classes from dictionaries. Thus the same two syntaxes apply there as well.
- Added support for application deployment across OpenStack regions. Now, all OpenStack resource classes inherit from `io.murano.CloudResource` that provides `.getRegion()` method and

`regionName` property. This allows assigning resources to different regions. `.getRegion()` returns `io.murano.CloudRegion` instance that resource or its parent belongs to. The interface of `CloudRegion` is similar to the `Environment` class and should be used to get `HeatStack` instance associated with the region, default network configuration, security group manager, and agent listener instances. `Environment` now acts as a default region so backward compatibility is not broken. However, new applications should not use the `Environment` to set security group rules but rather a region(s) of their instance(s) in order to work correctly when their instances were configured to use region other than the default.

- Added the `api_workers` option to the `murano` configuration group. It controls the number of API workers launched by murano-api. If not set, it defaults to the number of CPUs available.
- Added a new engine RPC call to generate json-schema from MuranoPL class. The schema may be generated either from the entire class or for specific model builders - static actions that can be used to generate object model from their input. Class schema is built by inspecting class properties and method schema using the same algorithm but applied to its arguments.
- Implemented a new framework for MuranoPL contracts. Now, instead of several independent implementations of the same YAQL methods (`string()`, `class()` and others) all implementations of the same method are combined into a single class. Therefore, we now have a class per contract method. This also simplifies the development of new contracts. Each such class can provide methods for data transformation (default contract usage), validation that is used to decide if the method can be considered an extension method for the value, and JSON schema generation method that was moved from the schema generator script.
- Previously, when a class overrode a property from its parent class the value was stored separately for both of them, transformed by each of the contracts. Thus each class accessed the value of its contract. In absolute majority of the cases, the observed value was the same. However, if the contracts were compatible on the provided value (say int() and string() contracts on the value “123”), they were different. This is considered to be a bad pattern. Now, the value is stored only once per object and transformed by the contract defined in the actual object, type. All base contracts are used to validate the transformed object thus this pattern will not work anymore.
- The value that is stored in the object’s properties is obtained by executing a special `finalize` contract implementation that, by default returns the input value unmodified. Because validation happens on the transformed value before it gets finalized, the transformation can return a value that will pass the validation, though the final value will not. This is used to relax the `template()` contract limitation that prevented child class from excluding additional properties from the template.
- The `string()` contract no longer converts everything to string values. Now, it only converts scalar values to strings. Previous behavior allowed the `string()` property to accept lists and convert them to their Python string representation, which is clearly not what developers expected.
- Due to refactoring, contracts work a little bit faster because there is no more need to generate YAQL function definition for each contract method on each call.
- Changed the type representation in the object model. Previously, there were three attributes in the ”?” section of the object - `type`, `classVersion`, and `package`, where only the `type` was mandatory. Now, these attributes are merged into single attribute `type` that has the `typeName/version@package` format. The `version` and `package` parts are still optional.
- Previously, when pre-deployment garbage collection occurred it executed `.destroy` method for objects that were present in the `ObjectsCopy` section of the object model (which is the snapshot of the model after last deployment) and not present in the current model anymore (because they were deleted through the API between deployments). If the destroyed objects were to access another object that was not deleted it was accessing its copy from the `ObjectsCopy`. Thus any changes to the internal state made by that object were lost after the garbage collection finished (that is, before the `.deploy` method call) and could not affect the deployment. Now, if the object is present in both `Objects` and `ObjectsCopy`, a single instance (the one from `Objects`) is used for both garbage collection and deployment. As a consequence, instances (in their `.destroy` method) now may observe changes made to other objects they refer if they were not deleted, but modified through the

API. In some rare cases, it may break existing applications.

- OSC plugin now accepts `--murano-url` and `MURANO_URL` to allow using a custom murano-api endpoint
- Murano dashboard now comes with the `muranodashboard/local/local_settings.d/_50_murano.py` file that contains murano-specific settings for horizon (for example, `MURANO_API_URL`).
- Separated murano service broker from murano-api into a murano-cfapi service. Created a separate database and `paste.ini` for service broker.
- Added a new API endpoint `v1/actions` to call static public methods. It accepts class name, method name, method arguments, and optionally package name and class version in the request body. This call does not create an environment, object instances or database records.
- Implemented a new contract function `template()`. `template()` works similar to the `class()` in regards to the data validation but does not instantiate objects. Instead, the data is left in the object model in dictionary format so that it could be instantiated later with the `new()` function. Additionally, the function allows excluding specified properties from validation and from the resulting template so that they could be provided later. Objects that are assigned to the property or argument with `template()` contract will be automatically converted to their object model representation.
- Split `Instance`‘s `.deploy()` method into two phases - `beginDeploy()` and `endDeploy()`. This allows the application developer to provision multiple instances at once without the need to push the stack for each instance.
- Added `/templates/{env_template_id}/services/{path:.*?}` API endpoint for environment template application update operation.
- Added the capability to declare MuranoPL YAML methods with variable length, positional, and keyword arguments. This is done using argument `Usage` attribute. Regular arguments have `Standard` usage which is the default. Variable length args (args in Python) should have `Usage: VarArgs` and keyword args (kwargs) are declared with `Usage: KwArgs`. Inside the method they are seen as a list and a dictionary correspondingly. For such arguments contracts are written for individual argument values, thus no need to write them as lists or dicts.

Known Issues

- Murano services will not operate correctly when installed on a separate detached node, if you’re installing `fuel-plugin-murano` in an environment that contains Murano, bundled with Fuel 9.0. This happens because `murano-engine` service on controller nodes is still active. As a workaround, you need to disable `murano-engine` service on controller nodes manually after installing `fuel-plugin-murano`.
- Deploying Cloud Foundry integration service with this release of `fuel-plugin-murano` does not work correctly.

Upgrade Notes

- New database migration 015 has to be applied.
- Previously, all murano-specific horizon settings had to be kept in `local_settings.py` file of Horizon. You need to remove those settings from `local_settings.py` and copy `muranodashboard/local/local_settings.d/_50_murano.py` to `openstack_dashboard/local/local_settings.d/` directory in horizon, and keep all murano-related changes there.

Deprecation Notes

- Deprecated the `Usage Action` keyword. For format versions 1.4.0 and newer, use `Scope Public` instead.
- Renamed the `workers` option from the `engine` group to `engine_workers` to reduce ambiguity with the `api_workers` option.

Security Issues

- Addressed cve-2016-4972. In several places Murano used loaders inherited directly from `yaml.Loader` when parsing MuranoPL and UI files from packages. This is unsafe, because this loader is capable of creating custom python objects from specifically constructed yaml files. With this change all yaml loading operations are done using safe loaders instead.

Bug Fixes

- When glare is used, specifying a base class in the UI definition now also fetches all the packages with classes that inherit from that class. For example, if you specify the 'io.example.Parent' class, the dashboard fetches 'io.example.Child1' and 'io.example.Child2', and any other descendants of 'io.example.Parent' that are present.
- Fixed the issue with sequential download of packages. Dashboard is now using 'tables.LinkAction' instead of 'tables.Action' for DownloadPackage table action.
- The API call for deleting a service from environment template did not return the result of its operation. The issue was fixed.
- Fixed the issue with the UI dialog that did not display in Murano Dashboard for applications without UI definitions bundled in the package. This usually affected HOT-based packages and other non-muranopl-based applications.
- Importing a package into glare now fills the *'inherited'* field with full inheritance info. Previously, it only contained immediate parent classes.
- Fixed the issue that prevented the Murano Dashboard from finding certain horizon filters, such as 'parse_isotime', 'timesince_or_never', and others.
- Removed the need for Keystone v2 options (`admin_user`, `admin_password`, `admin_tenant_name`) when Keystone v3 is in use.
- Previously, Murano assumed that the service user and service project are in the 'Default' domain. These values can now be set in the `keystone_authtoken` configuration group.
- The equality check (assertEqual) in the test-runner can now properly compare two MuranoPl objects.
- Prevented the resource leak for objects created during deployment with `new()` function call.
- Murano is now able to deploy applications in the environments with disabled Neutron Security Groups. Detection is based on the presence of the `security-group` Neutron extension.
- Fixed the password check in dynamic UI forms. Previously, the dashboard did not validate the password fields with IDs not ending with 'password'. Now, to determine whether to add default password validators to the field or not, the dashbord only checks the field type itself, instead of both field type and ending of the field ID.
- The *Environment components* page now reloads after an empty environment deployment. This allows adding new components to the empty environment without having to reload the page manually.
- Fixed the order of packages import. The main package is now imported before all its dependencies. Cyclic dependencies are imported in a random order.

- Added the capability to simultaneously use the `--resources-dir` and `--template` arguments for the **package-create** command. This allows creating HOT-packages with the `Resources` folder.
- Whenever murano-engine accesses script files, text script files are opened in 'rU' mode that recognizes all types of newlines and binary files are opened in 'rb' mode to prevent their corruption.
- Improved the performance of the *Recent Activity* panel on the *Browse Catalog* page.
- Fixed the issue that reset all environment changes from the previous session (adding or removing components without deployment) after consequent login. Also, the *Ready to deploy* status now only displays if there are changes pending in the current session.
- The test-runner now outputs the tests it runs and their results to stdout directly, instead of the logging system.
- The test-runner now does not output logs to stderr by default unless a 'use_stderr' parameter is specified in the configuration file.
- Fixed the issue that prevented the test-runner from properly invoking `setUp` and `tearDown` methods of fixtures in some cases.