supernova Documentation

Release trunk

Major Hayden

June 21, 2015

Contents

supernova helps you use novaclient (and other executables) with multiple environments the easy way.

You may like supernova if you regularly have the following problems:

- You hate trying to source multiple novarc files when using nova
- You get your terminals confused and do the wrong things in the wrong nova environment
- You don't like remembering things
- You want to keep sensitive API keys and passwords out of plain text configuration files (see the *Working with keyring storage* section)
- You need to share common skeleton environment variables for nova with your teams

If any of these complaints ring true, **supernova is for you**. supernova manages multiple nova environments without sourcing novarc files or mucking with environment variables.

CHAPTER 1

Documentation

1.1 Rackspace Quick Start

If you're already a Rackspace customer, you can get started quickly with supernova by following these steps:

```
pip install supernova rackspace-novaclient
wget -O ~/.supernova http://bit.ly/raxsupernova
supernova-keyring -s global RackspaceAccountUser
supernova-keyring -s global RackspaceAccountAPIKey
supernova-keyring -s global RackspaceAccountDDI
supernova dfw list
```

Let's break down these steps one by one.

We start by installing supernova as well as rackspace-novaclient (which is required for novaclient to talk to Rackspace's identity API):

```
pip install supernova rackspace-novaclient
```

There's a Rackspace example configuration file in the main repository and we can store that in our home directory as *.supernova*:

```
wget -O ~/.supernova http://bit.ly/raxsupernova
```

One of supernova's features is the ability to store sensitive data in your operating system's keyring. This also makes it handy when you want to re-use usernames, account numbers and passwords across multiple environments. We add that data to the keyring with these commands:

```
supernova-keyring -s global RackspaceAccountUser
supernova-keyring -s global RackspaceAccountAPIKey
supernova-keyring -s global RackspaceAccountDDI
```

Finally, we can test our configuration by listing our current instances in the DFW region:

```
supernova dfw list
```

1.2 Installing supernova

There are multiple options for installing supernova depending on whether you want to run a stable release or the latest available code.

1.2.1 Getting stable releases from PyPi

If you're installing for the first time, you can install supernova using pip:

```
pip install supernova
```

Be sure to install rackspace-novaclient if you plan to use supernova with Rackspace's Cloud Servers environments:

```
pip install supernova rackspace-novaclient
```

Upgrading supernova is easy as well:

```
pip install -U supernova
```

1.2.2 Latest available releases from GitHub

You can install the latest available version of supernova from GitHub using pip:

```
pip install git+git://github.com/major/supernova.git
```

You can also clone the repository and install supernova:

```
git clone https://github.com/major/supernova.git
cd supernova
python setup.py install
```

1.3 Configuring supernova

The following locations are valid configuration files for supernova.

- ${XDG_CONFIG_HOME}/supernova
- ~/.supernova
- ./.supernova

A complete list of nova environment variables can be located here.

In addition to the nova environment variables the config can include a key of 'BYPASS_URL' that will be passed to nova's –bypass-url command line option.

1.3.1 Working with keyring storage

With supernova, there are three ways you can store credentials:

- **Best:** Global keyring storage shared between multiple environments
- **Better:** Keyring storage specific to an environment
- **Awful:** Plain text in the supernova configuration file

NOTE: A temporary environment is spawned by supernova when it actually runs nova – your credentials aren't left in the environment variables of your current shell.

1.3.2 Global keyring storage

Storing a credential as a global credential allows you to use it across multiple environments. The greatest benefit of this option is that you only need to set credentials in one place within your keyring. If you need to change credentials frequently, this can be a big time saver.

Here's how we're already doing that in the Rackspace example configuration:

```
[dfw]
... snip ...
OS_PASSWORD=USE_KEYRING['RackspaceAccountAPIKey']
OS_USERNAME=USE_KEYRING['RackspaceAccountUser']
OS_TENANT_NAME=USE_KEYRING['RackspaceAccountDDI']

[ord]
... snip ...
OS_PASSWORD=USE_KEYRING['RackspaceAccountAPIKey']
OS_USERNAME=USE_KEYRING['RackspaceAccountUser']
OS_TENANT_NAME=USE_KEYRING['RackspaceAccountDDI']

[iad]
... snip ...
OS_PASSWORD=USE_KEYRING['RackspaceAccountAPIKey']
OS_USERNAME=USE_KEYRING['RackspaceAccountUser']
OS_TENANT_NAME=USE_KEYRING['RackspaceAccountDDI']
```

You can set global credentials with supernova-keyring:

```
supernova-keyring -s global RackspaceAccountAPIKey
```

Retrieving a credential you stored previously is easy as well:

```
supernova-keyring -g global RackspaceAccountAPIKey
```

WARNING: Retrieving a credential will display it on the screen in plain text. Don't worry: supernova will warn you thoroughly before you try this.

1.3.3 Environment-specific keyring storage

An older method of storing keyring data is to store it specifically for one environment. Using the Rackspace example configuration, your DFW configuration might look like this:

```
[dfw]
... snip ...
OS_PASSWORD=USE_KEYRING
OS_USERNAME=my_username
OS_TENANT_NAME=my_account_number
```

This tells supernova to look up the *OS_PASSWORD* value for the *dfw* environment. Setting that credential is done with supernova-keyring:

```
supernova-keyring -s dfw OS_PASSWORD
```

This was one of the first methods of keyring storage available in supernova and it's the least robust. It's recommended to consider using global keyring storage instead.

1.3.4 Plain text storage

This is obviously the easiest method, but it's generally not recommended. Any user with access to your home directory would have access to your credentials and could use them against your accounts. **Don't use plain text credential storage unless you know what you're doing.**

Using the Rackspace example configuration, plain text credential storage would look like this:

```
[dfw]
OS_AUTH_URL=https://identity.api.rackspacecloud.com/v2.0/
OS_AUTH_SYSTEM=rackspace
OS_COMPUTE_API_VERSION=1.1
NOVA_RAX_AUTH=1
OS_REGION_NAME=DFW
NOVA_SERVICE_NAME=cloudServersOpenStack
OS_PASSWORD=482aa30919030cafc08f2d2bda2193bf
OS_USERNAME=my_username
OS_TENANT_NAME=123456
```

When supernova runs, it will take the configuration options and pass them directly to nova (or the executable of your choice) in a subprocess.

1.4 Using supernova

1.4.1 Passing commands to nova

The structure of a supernova command is relatively simple:

```
supernova [enviromment_name] [commands]
```

Here are some examples:

- Listing instances in the *iad* environment:

```
supernova iad list
```

- Listing available images:

```
supernova iad image-list
```

- Listing available flavors:

```
supernova iad flavor-list
```

- Booting an instance:

```
supernova iad boot --image image_uuid --flavor flavor_id myserver.example.com
```

You can use supernova with long-running options and the output will be piped live to your terminal. For example, you can use `--poll` when you boot an instance and you'll see the novaclient output update as your instance is being built.

For debug output, supernova accepts `--debug` as a supernova option or as a novaclient option. Both of these are okay:

```
supernova --debug iad list
supernova iad list --debug
```

Be careful with sharing debug output, many clients will print credentials in plain text even if the credentials are not stored in plain text.

1.4.2 Listing available supernova environments

There's a convenience function provided that will dump out your parsed supernova configuration file to the screen:

```
supernova -l
```

This is a good way to test your configuration file syntax and diagnose problems. Keep in mind that any plain text credentials will be displayed on screen when you list environments (see *Working with keyring storage* for more details).

If you're still using plain text credentials, now is a good time to stop using them. :)

1.4.3 Grouped environments

Starting in supernova 0.9.6, you can "group" environments and run commands across all of the environments in a group. Using the Rackspace example configuration, we can group the DFW, IAD and ORD environments into a group called "raxusa":

```
[dfw]
SUPERNOVA_GROUP=raxusa
...snip...

[ord]
SUPERNOVA_GROUP=raxusa
...snip...

[iad]
SUPERNOVA_GROUP=raxusa
...snip...
```

Instead of referring to these environments one by one, you can now run commands across them as a group:

```
supernova raxusa list
```

This can be quite useful if you need to search multiple environments for an instance, or if you need to boot test instances in multiple datacenters. Just be careful with any actions that manipulate data, like rebuilds or instance deletions.

1.4.4 Using supernova with different executables

While supernova uses the nova executable by default, you can configure it to use any other executable when it runs. For example, you could use it to run glance, neutron, or keystone.

You can use the same executable for certain environments every time by adding *OS_EXECUTABLE* to your supernova configuration file:

```
[iadglance]
OS_EXECUTABLE=glance
...snip...
```

If you'd rather make a quick change at runtime, just use `-x` or `--executable`:

```
supernova -x glance iadglance image-list
```

1.4.5 Checking the supernova version

As of supernova 1.0.5, you can use the `--version` argument to have supernova's version printed on the command line:

```
$ supernova --version
supernova 1.0.5
```