opie Documentation

Release

Alvaro Lopez Garcia

Apr 10, 2017

Contents

1 Installation **3**
1.1 Installation via Ubuntu/Debian packages . . . 3
1.2 Installation via pip . . . 3
1.3 Paching nova compute API . . . 3

2 Configuration **7**

3 Indices and tables **9**

opie is the materialization of the preemptible instances extension serving as a reference implementation. This package provides a set of pluggable extensions for OpenStack Compute (nova) making possible to execute premptible instances using a modified filter scheduler.

opie is completely pluggable, with the exception of a modification in the internal nova compute API that needs to be applied manually.

Contents:

CHAPTER 1

Installation

opie depends on the OpenStack Compute (nova) version that you are using. Currently it is only tested with OpenStack Compute Netwton (version 14), so you need to ensure that you are running that specific version.

Installation via Ubuntu/Debian packages

Add the opie PPA to your system manually, by copying the linbes below and adding them to your system's software sources:

```
deb http://ppa.launchpad.net/aloga/opie/ubuntu trusty main
deb-src http://ppa.launchpad.net/aloga/opie/ubuntu trusty main
```

Then, you can install it via apt-get:

```
apt-get update
apt-get install opie
```

Installation via pip

At the command line you can install:

```
$ pip install opie
```

Paching nova compute API

Unfortunately, even if opie is desinged to exploit nova's modularity, there is a manual patch that needs to be applied. Execute the following command to save the patch:

```
cat > /tmp/nova-compute-api.patch << EOF
--- /a/compute/api.py       2016-07-27 13:50:06.625151062 +0200
+++ /b/compute/api.py       2016-09-13 12:40:36.190353525 +0200
@@ -886,7 +886,7 @@
                                          access_ip_v4, access_ip_v6,
                                          requested_networks, config_drive,
                                          auto_disk_config, reservation_id,
-                                         max_count):
+                                         max_count, preemptible):
         """Verify all the input parameters regardless of the provisioning
         strategy being performed.
         """
@@ -948,6 +948,8 @@
                 instance_type, image_meta)

         system_metadata = {}
+        if preemptible:
+            system_metadata["preemptible"] = True

         # PCI requests come from two sources: instance flavor and
         # requested_networks. The first call in below returns an
@@ -1144,7 +1146,7 @@
                block_device_mapping, auto_disk_config,
                reservation_id=None, scheduler_hints=None,
                legacy_bdm=True, shutdown_terminate=False,
-               check_server_group_quota=False):
+               check_server_group_quota=False, preemptible=False):
         """Verify all the input parameters regardless of the provisioning
         strategy being performed and schedule the instance(s) for
         creation.
@@ -1184,7 +1186,7 @@
                 key_name, key_data, security_groups, availability_zone,
                 forced_host, user_data, metadata, access_ip_v4,
                 access_ip_v6, requested_networks, config_drive,
-                auto_disk_config, reservation_id, max_count)
+                auto_disk_config, reservation_id, max_count, preemptible)

         # max_net_count is the maximum number of instances requested by the
     # user adjusted for any network quota constraints, including
@@ -1553,7 +1555,8 @@
                block_device_mapping=None, access_ip_v4=None,
                access_ip_v6=None, requested_networks=None, config_drive=None,
                auto_disk_config=None, scheduler_hints=None, legacy_bdm=True,
-               shutdown_terminate=False, check_server_group_quota=False):
+               shutdown_terminate=False, check_server_group_quota=False,
+               preemptible=False):
         """Provision instances, sending instance information to the
         scheduler.  The scheduler will determine where the instance(s)
         go and will handle creating the DB entries.
@@ -1584,7 +1587,8 @@
                        scheduler_hints=scheduler_hints,
                        legacy_bdm=legacy_bdm,
                        shutdown_terminate=shutdown_terminate,
-                       check_server_group_quota=check_server_group_quota)
+                       check_server_group_quota=check_server_group_quota,
+                       preemptible=preemptible)

     def _check_auto_disk_config(self, instance=None, image=None,
```

```
                                    **extra_instance_updates):
--- /a/conf/scheduler.py
+++ /b/conf/scheduler.py
@@ -214,7 +214,7 @@ configuration.

 sched_driver_host_mgr_opt = cfg.StrOpt("scheduler_host_manager",
         default="host_manager",
-        choices=("host_manager", "ironic_host_manager"),
+        choices=("host_manager", "ironic_host_manager", "opie_host_manager"),
         help="""
 The scheduler host manager to use, which manages the in-memory picture of the
 hosts that the scheduler uses.
EOF
```

And apply the patch as follows:

- Ubuntu/Debian:

```
cd /usr/lib/python2.7/dist-packages/nova
patch -p1 < /tmp/nova-compute-api.patch
```

- RedHat/CentOS:

```
cd /usr/lib/python2.7/site-packages/nova
patch -p1 < /tmp/nova-compute-api.patch
```

Once this is done, ensure that you restart your `nova-api` service:

```
service nova-api restart
```

CHAPTER 2

Configuration

Once opie is installed you should enable it in your `nova.conf` configuration file.

opie implements a different host manager and scheduler, compatible with the original FilterScheduler, so ensure that your configuration file contains the following in the `[DEFAULT]` section:

```
[DEFAULT]
scheduler_driver = opie.scheduler.filter_scheduler.FilterScheduler
scheduler_host_manager = opie.scheduler.host_manager.HostManager
```

Moreover, in order to do a proper scheduling, opie implements two additional weighers:

- `PreemptibleDurationWeigher`: This weigher assumes that you are charging your users on 1h periods, therefire it will assign weights based on 1h-period consumptions. This way, the machines with the largest remainder (that is, time above 1h-periods) will get the lowest weight. For instance, if machine A has 1h 59min it will get a weight lower than a machine with 1h 01min running time.
- `PreemptibleCountWeigher`: This will assign weights based on the number of preemptible instances that are running on a host.

You can enable both weighers and its multipliers by adding the following configuration options to the `[DEFAULT]` section:

```
scheduler_weight_classes = (...), opie.scheduler.weights.preemptible.
↪PreemptibleDurationWeigher, opie.scheduler.weights.preemptible.
↪PreemptibleCountWeigher
preemptible_count_weight_multiplier = 1000.0
preemptible_duration_weight_multiplier = 1000.0
```

Once the scheduler is configured, ensure that you restart your `nova-scheduler` service:

```
service nova-scheduler restart
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search