OpenStack-Ansible Documentation: memcached_server role

Release 18.1.0.dev122

OpenStack-Ansible Contributors

Dec 02, 2025

CONTENTS

CHAPTER

ONE

ALTERNATIVE MEMCACHED CONFIGURATIONS

By default Memcached servers are deployed on each controller host as a part of *shared-infra_containers* group. Drivers, like oslo_cache.memcache_pool support marking memcache backends as dead, however not all services allow you to select the driver which will be used for interaction with Memcached. In the meanwhile, you may face services API response delays or even unresponsive APIs while one of the memcached backends is down.

This is why you may want to use HAProxy for handling access and to check for backend aliveness or use always local to the service memcached server.

1.1 Configuring Memcached through HAProxy

Setting haproxy in front of the Memcached servers and relying on it for checking aliveness of the backends gives more reliable failover and minimize delays in case of backend failure. We need to define the following in your `user_variables.yml`:

```
haproxy_memcached_allowlist_networks: "{{ haproxy_allowlist_networks }}"
memcached_servers: "{{ internal_lb_vip_address ~ ':' ~ memcached_port }}"
haproxy_extra_services:
  - haproxy_service_name: memcached
    haproxy_backend_nodes: "{{ groups['memcached'] | default([]) }}"
    haproxy_bind: "{{ [internal_lb_vip_address] }}"
    haproxy_port: 11211
    haproxy_balance_type: tcp
    haproxy_balance_alg: source
    haproxy_backend_ssl: False
    haproxy_backend_options:
      - tcp-check
    haproxy_allowlist_networks: "{{ haproxy_memcached_allowlist_networks }}"
```

After setting this you will need to update haproxy and all services configuration to use new memcached backend:

```
# openstack-ansible playbooks/haproxy-install.yml
# openstack-ansible playbooks/setup-openstack.yml
```

1.2 Using only local Memcached

The idea behind this, is to configure services to use Memcached, that will reside only on the local control plane. Here local means not only having memcahed inside the container with the service itslef, but also having memcached inside a separate container on the same controller as the service well.

This will reduce latency and improve stability, since service and memcached instance will be running on the same control plane just on different containers that are connected through the same L2 bridge.

Among the cons of this approach is that there wont be any failover available in case of the memcached container crush, so caching on this controller wont work. For pros its only 1 controller that will be affected and not all of them in the case of remote memcached unavailability. Also, most the common scenario of API response delays is when whole controller goes down, since connection drop causes memcache_pool to wait for connection timeout rather then connection is rejected when memcached service goes down and memcache_pool instantly switches to another backend.

Note

In case some service wont have memcached server locally to the service that is running, behaviour will fallback to using all available memcached servers and memcached client will decide which one to use.

In order to always use local memcached you need to define the following in the `user_variables.yml` file:

```
memcached_servers: |-
    {% set service_controller_group = group_names | select('regex', '.*-host_
↪containers') | first | default('memcached') %}
    {{
      groups['memcached'] | intersect(groups[service_controller_group])
        | map('extract', hostvars, 'management_address')
        | map('regex_replace', '(.+)', '\1:' ~ memcached_port)
        | list | join(',')
    }}
```

After setting that you need to update all services configuration to use new memcached backend:

```
# openstack-ansible playbooks/setup-openstack.yml
```

Ansible role to install and configure Memcached.

To clone or view the source code for this repository, visit the role repository for memcached_server.

CHAPTER

TWO

DEFAULT VARIABLES

```yaml
## Logging level
debug: false

## APT Cache Options
cache_timeout: 600

# Set the package install state for distribution packages
# Options are 'present' and 'latest'
memcached_package_state: "latest"

# MemcacheD could set 'PrivateDevices=True' for its systemd unit by default␣
↪when
# installed into a container. This provides some additional security, but it
# causes problems with creating mount namespaces on CentOS 7 with systemd 219.
# While the security enhancements are helpful on bare metal hosts with
# multiple services running, they are not as helpful when MemcacheD is running
# in a container with its own isolated namespaces.
#
# Related bugs:
#   https://bugs.launchpad.net/openstack-ansible/+bug/1697531
#   https://github.com/lxc/lxc/issues/1623
#   https://github.com/systemd/systemd/issues/6121
#
# Setting the following variable to 'yes' will disable the PrivateDevices
# setting in the systemd unit file for MemcacheD on CentOS 7 hosts.
memcached_disable_privatedevices: "{{ ansible_facts['pkg_mgr'] == 'dnf' }}"

# The default memcache memory setting is to use .25 of the available system␣
↪ram
# as long as that value is < 8192. However you can set the `memcached_memory`
# value to whatever you like as an override.
base_memcached_memory: "{{ ansible_facts['memtotal_mb'] | default(4096) }}"
memcached_memory: "{{ base_memcached_memory | int // 4 if base_memcached_
↪memory | int // 4 < 8192 else 8192 }}"

memcached_port: 11211
memcached_listen: "127.0.0.1"
memcached_connections: 4096
```

(continues on next page)

(continued from previous page)

```
memcached_threads: 4
memcached_file_limits: "{{ memcached_connections | int + 1024 }}"

install_test_packages: false
```

CHAPTER

THREE

REQUIRED VARIABLES

None

CHAPTER

FOUR

EXAMPLE PLAYBOOK

```yaml
---
- name: Install Memcached
  hosts: memcached
  user: root
  roles:
    - { role: "memcached_server" }
```