Red Hat OpenStack Platform 16.1

External Load Balancing for the Overcloud

Configuring an OpenStack Platform Environment to Use an External Load Balancer

Last Updated: 2020-09-08

Red Hat OpenStack Platform 16.1 External Load Balancing for the Overcloud

Configuring an OpenStack Platform Environment to Use an External Load Balancer

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide provides information on configuring a Red Hat OpenStack Platform environment to use an external load balancer for the Overcloud. This includes configuration guidelines for your load balancer and configuration of the Overcloud through the OpenStack Platform director.

Table of Contents

CHAPTER 1. INTRODUCTION

Red Hat OpenStack Platform director creates a cloud environment called the **Overcloud**. The Overcloud contains a set of different node types that perform certain roles. One of these node types is the **Controller** node. The Controller is responsible for Overcloud administration and uses specific OpenStack components. An Overcloud uses multiple Controllers together as a high availability cluster, which ensures maximum operational performance for your OpenStack services. In addition, the cluster provides load balancing for access to the OpenStack services, which evenly distributes traffic to the Controller nodes and reduces server overload for each node.

It is also possible to use an external load balancer to perform this distribution. For example, an organization might use their own hardware-based load balancer to handle traffic distribution to the Controller nodes. This guide provides the necessary details to help define the configuration for both an external load balancer and the Overcloud creation. This involved the following process:

1. **Installing and Configuring the Load Balancer**- This guide includes some HAProxy options for load balancing and services. Translate the settings to the equivalent of your own external load balancer.

2. **Configuring and Deploying the Overcloud**- This guide includes some Heat template parameters help the Overcloud integrate with the external load balancer. This mainly involves the IP addresses of the load balancer and potential nodes. This guide also includes the command to start the Overcloud deployment and its configuration to use the external load balancer.

1.1. USING LOAD BALANCING IN THE OVERCLOUD

The Overcloud uses a open source tool called **HAProxy**. HAProxy load-balances traffic to Controller nodes running OpenStack services. The **haproxy** package contains the **haproxy** daemon, which is started from the haproxy systemd service, along with logging features and sample configurations. However, the Overcloud also uses a high availability resource manager (Pacemaker) to control HAProxy itself as a highly available service (haproxy-clone). This means HAProxy runs on each Controller node and distributes traffic according to a set of rules defined in each configuration.

1.2. DEFINING AN EXAMPLE SCENARIO

This article uses the following scenario as an example:

- An external load balacing server using HAProxy. This demonstrates how to use a federated HAProxy server. You can substitute this for another supported external load balancer.

- One OpenStack Platform director node

- An Overcloud that consists of:

- 3 Controller nodes in a highly available cluster

- 1 Compute node

- Network isolation with VLANs

The scenario uses the following IP address assignments for each network:

- Internal API: 172.16.20.0/24

- Tenant: 172.16.22.0/24
- Storage: 172.16.21.0/24
- Storage Management: 172.16.19.0/24
- External: 172.16.23.0/24

These IP ranges will include IP assignments for the Controller nodes and virtual IPs that the load balancer binds to OpenStack services.

CHAPTER 2. DEFINING THE DEFAULT CONFIGURATION

When creating and configuring an Overcloud without an external load balancer, the director configures HAProxy to distribute traffic to multiple OpenStack services. The director provides this configuration in the **/etc/haproxy/haproxy.conf** file on each Controller node. The default configuration contains three main parts: global, defaults, and multiple service configurations.

The next few sections examine the default parameters from each configuration section. This provides an example of the configuration settings for installing and configuring your external load balancer. Note that these parameters are only a fraction of the total HAProxy parameters. For details about these and other parameters, see the "HAProxy Configuration Manual" located in **/usr/share/doc/haproxy-*/configuration.txt** on the Controller nodes (or any system where the **haproxy** package is installed).

2.1. GLOBAL CONFIGURATION

```
global
  daemon
  group  haproxy
  log  /dev/log local0
  maxconn  10000
  pidfile  /var/run/haproxy.pid
  user  haproxy
```

This section defines a set of process-wide parameters. This includes the following:

- **daemon**: Run as a background process.
- **user haproxy**, **group haproxy**: Defines the Linux user and group that owns the process.
- **log**: Defines syslog server to use.
- **maxconn**: Sets the maximum number of concurrent connections to the process.
- **pidfile**: Sets file to use for the process IDs.

2.2. DEFAULTS CONFIGURATION

```
defaults
  log  global
  mode  tcp
  retries  3
  timeout  http-request 10s
  timeout  queue 1m
  timeout  connect 10s
  timeout  client 1m
  timeout  server 1m
  timeout  check 10s
```

This section defines a default set of parameters for each service. This includes the following:

- **log**: Enables logging for the service. The **global** value means that the logging functions use the **log** parameters in the **global** section.
- **mode**: Sets the protocol to use. In this case, the default is TCP.

- **retries**: Sets the number of retries to perform on a server before reporting a connection failure.
- **timeout**: Sets the maximum time to wait for a particular function. For example, **timeout http-request** sets the maximum time to wait for a complete HTTP request.

2.3. SERVICES CONFIGURATION

```
listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter 2000 rise 2
```

There are multiple service configuration sections in the default file. Each service configuration includes the following:

- **listen**: The name of the service listening for requests
- **bind**: The IP address and TCP port number the on which the service listens
- **server**: The name of each server providing the service, the server's IP address and listening port, and other information.

The example above shows the HAProxy settings for the **ceilometer** service. This services identifies the IP addresses and ports on which the ceilometer service is offered (port 8777 on 172.16.20.2500 and 172.16.23.250). HAProxy directs the requests made for those addresses to **overcloud-controller-0** (172.16.20.150:8777), **overcloud-controller-1** (172.16.20.151:8777), or **overcloud-controller-2** (172.16.0.152:8777).

In addition, the example **server** parameters enable the following:

- **check**: Enables health checks
- **fall 5**: After five failed health checks, the service is considered dead.
- **inter 2000**: The interval between two consecutive health checks set to 2000 milliseconds (or 2 seconds).
- **rise 2**: After two successful health checks, a server is considered operational.

Each service binds to different addresses, representing different network traffic types. Also some services contain additional configuration options. The next chapter examines each specific service configuration so that you can replicate these details on your external load balancer.

CHAPTER 3. SERVICES CONFIGURATION REFERENCE

This chapter outlines the configuration for each specific service in the overcloud that uses load balancing. Use this configuration as a guide to configure your external load balancer. For more information about parameters, see the "HAProxy Configuration Manual" located in **/usr/share/doc/haproxy-*/configuration.txt** on the Controller nodes (or any system where the haproxy package is installed).

NOTE

Most services use a default health check configuration:

- The interval between two consecutive health checks set to 2000 milliseconds (or 2 seconds).
- After two successful health checks, a server is considered operational.
- After five failed health checks, the service is considered dead.

Each service indicates the default health check or additional options in the **Other information** section of each service.

3.1. AODH

Port Number: 8042

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen aodh
  bind 172.16.20.250:8042
  bind 172.16.23.250:8042
  server overcloud-controller-0 172.16.20.150:8042 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8042 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8042 check fall 5 inter 2000 rise 2
```

3.2. CEILOMETER

Port Number: 8777

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter 2000 rise 2
```

3.3. CINDER

Port Number: 8776

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen cinder
  bind 172.16.20.250:8776
  bind 172.16.23.250:8776
  server overcloud-controller-0 172.16.20.150:8776 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8776 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8776 check fall 5 inter 2000 rise 2
```

3.4. GLANCE_API

Port Number: 9292

Binds to: storage, external

Target network/server: storage on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen glance_api
  bind 172.16.23.250:9292
  bind 172.16.21.250:9292
  server overcloud-controller-0 172.16.21.150:9292 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.21.151:9292 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.21.152:9292 check fall 5 inter 2000 rise 2
```

3.5. GLANCE_REGISTRY

Port Number: 9191

Binds to: internal_api

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen glance_registry
  bind 172.16.20.250:9191
  server overcloud-controller-0 172.16.20.150:9191 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:9191 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:9191 check fall 5 inter 2000 rise 2
```

3.6. GNOCCHI

Port Number: 8041

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen gnocchi
  bind 172.16.20.250:8041
  bind 172.16.23.250:8041
  server overcloud-controller-0 172.16.20.150:8041 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8041 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8041 check fall 5 inter 2000 rise 2
```

3.7. HEAT_API

Port Number: 8004

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

- This service uses HTTP mode instead of the default TCP mode

HAProxy Example:

```
listen heat_api
  bind 172.16.20.250:8004
  bind 172.16.23.250:8004
  mode http
  server overcloud-controller-0 172.16.20.150:8004 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8004 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8004 check fall 5 inter 2000 rise 2
```

3.8. HEAT_CFN

Port Number: 8000

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen heat_cfn
  bind 172.16.20.250:8000
  bind 172.16.23.250:8000
  server overcloud-controller-0 172.16.20.150:8000 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.152:8000 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.151:8000 check fall 5 inter 2000 rise 2
```

3.9. HEAT_CLOUDWATCH

Port Number: 8003

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen heat_cloudwatch
  bind 172.16.20.250:8003
  bind 172.16.23.250:8003
```

```
server overcloud-controller-0 172.16.20.150:8003 check fall 5 inter 2000 rise 2
server overcloud-controller-1 172.16.20.151:8003 check fall 5 inter 2000 rise 2
server overcloud-controller-2 172.16.20.152:8003 check fall 5 inter 2000 rise 2
```

3.10. HORIZON

Port Number: 80

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check
- This service uses HTTP mode instead of the default TCP mode
- This service uses cookie-based persistence for interactions with the UI

HAProxy Example:

```
listen horizon
  bind 172.16.20.250:80
  bind 172.16.23.250:80
  mode http
  cookie SERVERID insert indirect nocache
  server overcloud-controller-0 172.16.20.150:80 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:80 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:80 check fall 5 inter 2000 rise 2
```

3.11. KEYSTONE_ADMIN

Port Number: 35357

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen keystone_admin
  bind 172.16.23.250:35357
  bind 172.16.20.250:35357
  server overcloud-controller-0 172.16.20.150:35357 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:35357 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:35357 check fall 5 inter 2000 rise 2
```

3.12. KEYSTONE_ADMIN_SSH

Port Number: 22

Binds to: internal_api

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen keystone_admin_ssh
  bind 172.16.20.250:22
  server overcloud-controller-0 172.16.20.150:22 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:22 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:22 check fall 5 inter 2000 rise 2
```

3.13. KEYSTONE_PUBLIC

Port Number: 5000

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen keystone_public
  bind 172.16.20.250:5000
  bind 172.16.23.250:5000
  server overcloud-controller-0 172.16.20.150:5000 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:5000 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:5000 check fall 5 inter 2000 rise 2
```

3.14. MYSQL

Port Number: 3306

Binds to: internal_api

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check. However, the health checks use port 9200.

- This service is load balanced to only one server at a time.
- Each server is only used in load balancing when all other non-backup servers are unavailable.
- If the server is marked down, all connections are immediately terminated.
- Enable the sending of TCP keepalive packets on both sides.
- Enable HTTP protocol to check on the servers health.
- Configure a stickiness table to store IP address. This helps maintain persistence.

IMPORTANT

The **mysql** service uses Galera to provide a highly available database cluster. While Galera supports an **active/active** configuration, we recommend using an **active/passive** enforced by the load balancer to avoid lock contention.

HAProxy Example:

```
listen mysql
  bind 172.16.20.250:3306
  option tcpka
  option httpchk
  stick on dst
  stick-table type ip size 1000
  timeout client 0
  timeout server 0
  server overcloud-controller-0 172.16.20.150:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2
  server overcloud-controller-1 172.16.20.151:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2
  server overcloud-controller-2 172.16.20.152:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2
```

3.15. NEUTRON

Port Number: 9696

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen neutron
  bind 172.16.20.250:9696
  bind 172.16.23.250:9696
```

```
  server overcloud-controller-0 172.16.20.150:9696 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:9696 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:9696 check fall 5 inter 2000 rise 2
```

3.16. NOVA_EC2

Port Number: 8773

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen nova_ec2
  bind 172.16.20.250:8773
  bind 172.16.23.250:8773
  server overcloud-controller-0 172.16.20.150:8773 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8773 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8773 check fall 5 inter 2000 rise 2
```

3.17. NOVA_METADATA

Port Number: 8775

Binds to: internal_api

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen nova_metadata
  bind 172.16.20.250:8775
  server overcloud-controller-0 172.16.20.150:8775 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8775 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8775 check fall 5 inter 2000 rise 2
```

3.18. NOVA_NOVNCPROXY

Port Number: 6080

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check
- The default balancing method is round-robin. However, for this service, use a **source** method. This method hashes the source IP address and divides it by the total weight of the running servers. This designates the server that receives the request. This ensures the same client IP address always reaches the same server as long as no server goes down or up. If the hash result changes due to a change in the number of running servers, the balancer redirects many clients to a different server.

HAProxy Example:

```
listen nova_novncproxy
  bind 172.16.20.250:6080
  bind 172.16.23.250:6080
  balance source
  server overcloud-controller-0 172.16.20.150:6080 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:6080 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:6080 check fall 5 inter 2000 rise 2
```

3.19. NOVA_OSAPI

Port Number: 8774

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen nova_osapi
  bind 172.16.20.250:8774
  bind 172.16.23.250:8774
  server overcloud-controller-0 172.16.20.150:8774 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8774 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8774 check fall 5 inter 2000 rise 2
```

3.20. NOVA_PLACEMENT

Port Number: 8778

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen nova_placement
  bind 172.16.20.250:8778
  bind 172.16.23.250:8778
  server overcloud-controller-0 172.16.20.150:8778 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8778 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8778 check fall 5 inter 2000 rise 2
```

3.21. PANKO

Port Number: 8779

Binds to: internal_api, external

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen panko
  bind 172.16.20.250:8779
  bind 172.16.23.250:8779
  server overcloud-controller-0 172.16.20.150:8779 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8779 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8779 check fall 5 inter 2000 rise 2
```

3.22. REDIS

Port Number: 6379

Binds to: internal_api (redis service IP)

Target network/server: internal_api on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check.
- Perform health checks using **tcp-check** send/expect sequences. The string to send is "info\ replication\r\n" and the response is "role:master"
- The Redis service uses a password for authentication. For example, the HAProxy configuration uses a **tcp-check** with and **AUTH** method and the Redis administration password. The director normally generates a random password, but you can define a custom Redis password. See Section 4.2.2, “Configuring Load Balancing Options” for more information.

- The default balancing method is round-robin. However, for this service, use a **first** method. This ensures the first server with available connection slots receives the connection.

HAProxy Example:

```
listen redis
  bind 172.16.20.249:6379 transparent
  balance first
  option tcp-check
  tcp-check send AUTH\ p@55w0rd!\r\n
  tcp-check send PING\r\n
  tcp-check expect string +PONG
  tcp-check send info\ replication\r\n
  tcp-check expect string role:master
  tcp-check send QUIT\r\n
  tcp-check expect string +OK
  server overcloud-controller-0 172.16.20.150:6379 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:6379 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:6379 check fall 5 inter 2000 rise 2
```

3.23. SWIFT_PROXY_SERVER

Port Number: 8080

Binds to: storage, external

Target network/server: storage on overcloud-controller-0, overcloud-controller-1, and overcloud-controller-2

Other information:

- Each target server uses a default health check

HAProxy Example:

```
listen swift_proxy_server
  bind 172.16.23.250:8080
  bind 172.16.21.250:8080
  server overcloud-controller-0 172.16.21.150:8080 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.21.151:8080 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.21.152:8080 check fall 5 inter 2000 rise 2
```

CHAPTER 4. CONFIGURING THE OVERCLOUD

This section runs through the process of creating an Overcloud that uses the external load balancer. This includes registering the nodes, configuring the network setup, and the configuring options required for the Overcloud creation command.

4.1. SETTING UP YOUR ENVIRONMENT

This section uses a cutdown version of the process from the *Director Installation and Usage* guide.

Use the following workflow to set up your environment:

- Create a node definition template and register blank nodes in the director.
- Inspect hardware of all nodes.
- Manually tag nodes into roles.
- Create flavors and tag them into roles.

4.1.1. Initializing the Stack User

Log into the director host as the **stack** user and run the following command to initialize your director configuration:

```
$ source ~/stackrc
```

This sets up environment variables containing authentication details to access the director's CLI tools.

4.1.2. Registering Nodes

A node definition template (**instackenv.json**) is a JSON format file and contains the hardware and power management details for registering nodes. For example:

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
            ],
            "cpu":"4",
```

```
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.206"
        },
        {
            "mac":[
                "dd:dd:dd:dd:dd:dd"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.207"
        },
        {
            "mac":[
                "ee:ee:ee:ee:ee:ee"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.208"
        }
    ]
}
```

After creating the template, save the file to the stack user's home directory (**/home/stack/instackenv.json**), then import it into the director. Use the following command to accomplish this:

```
$ openstack overcloud node import ~/instackenv.json
```

This imports the template and registers each node from the template into the director.

Assign the kernel and ramdisk images to all nodes:

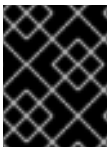
```
$ openstack overcloud node configure
```

The nodes are now registered and configured in the director.

4.1.3. Inspecting the Hardware of Nodes

After registering the nodes, inspect the hardware attribute of each node. Run the following command to inspect the hardware attributes of each node:

```
$ openstack overcloud node introspect --all-manageable
```



IMPORTANT

The nodes must be in the **manageable** state. Make sure this process runs to completion. This process usually takes 15 minutes for bare metal nodes.

4.1.4. Manually Tagging the Nodes

After registering and inspecting the hardware of each node, tag them into specific profiles. These profile tags match our nodes to flavors, and in turn the flavors are assigned to a deployment role.

Retrieve a list of your nodes to identify their UUIDs:

```
$ ironic node-list
```

To manually tag a node to a specific profile, add a profile option to the properties/capabilities parameter for each node. For example, to tag three nodes to use a controller profile and one node to use a compute profile, use the following commands:

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13 add
properties/capabilities='profile:compute,boot_option:local'
```

The addition of the **profile:compute** and **profile:control** options tag the nodes into each respective profiles.

4.2. CONFIGURING THE NETWORK

This section examines the network configuration for the Overcloud. This includes isolating our services to use specific network traffic and configuring the Overcloud with our load balancing options.

4.2.1. Isolating the Network

The director provides methods to configure isolated overcloud networks. This means the Overcloud environment separates network traffic types into different networks, which in turn assigns network traffic to specific network interfaces or bonds. After configuring isolated networks, the director configures the OpenStack services to use the isolated networks. If no isolated networks are configured, all services run on the Provisioning network.

First, the Overcloud requires a set of network interface templates. You customize these templates to configure the node interfaces on a per role basis. These templates are standard Heat templates in YAML format. The director contains a set of example templates to get you started:

- **/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans** - Directory containing templates for single NIC with VLANs configuration on a per role basis.

- **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans** - Directory containing templates for bonded NIC configuration on a per role basis.

For more information on network interface configuration, see the *Director Installation and Usage* guide.

Next, create a network environment file. This file is a Heat environment file that describes the Overcloud's network environment and points to the network interface configuration templates. This file also defines the subnets and VLANs for our network along with IP address ranges. You customize these values for the local environment.

This scenario uses the following network environment file saved as **/home/stack/network-environment.yaml**:

```
resource_registry:
  OS::TripleO::BlockStorage::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/cinder-storage.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/controller.yaml
  OS::TripleO::ObjectStorage::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/swift-storage.yaml
  OS::TripleO::CephStorage::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/ceph-storage.yaml

parameter_defaults:
  InternalApiNetCidr: 172.16.20.0/24
  TenantNetCidr: - 172.16.22.0/24
  StorageNetCidr: 172.16.21.0/24
  StorageMgmtNetCidr: 172.16.19.0/24
  ExternalNetCidr: 172.16.23.0/24
  InternalApiAllocationPools: [{'start': '172.16.20.10', 'end': '172.16.20.200'}]
  TenantAllocationPools: [{'start': '172.16.22.10', 'end': '172.16.22.200'}]
  StorageAllocationPools: [{'start': '172.16.21.10', 'end': '172.16.21.200'}]
  StorageMgmtAllocationPools: [{'start': '172.16.19.10', 'end': '172.16.19.200'}]
  # Leave room for floating IPs in the External allocation pool
  ExternalAllocationPools: [{'start': '172.16.23.10', 'end': '172.16.23.60'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 172.16.23.1
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.0.2.254
  # The IP address of the EC2 metadata server. Generally the IP of the Undercloud
  EC2MetadataIp: 192.0.2.1
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["8.8.8.8","8.8.4.4"]
  InternalApiNetworkVlanID: 201
  StorageNetworkVlanID: 202
  StorageMgmtNetworkVlanID: 203
  TenantNetworkVlanID: 204
  ExternalNetworkVlanID: 100
  # Set to "br-ex" if using floating IPs on native VLAN on bridge br-ex
  NeutronExternalNetworkBridge: """"
  # Customize bonding options if required
  BondInterfaceOvsOptions:
    "bond_mode=balance-tcp lacp=active other-config:lacp-fallback-ab=true"
```

For more information on network environment configuration, see the *Director Installation and Usage* guide.

Ensure the director host has access to the Internal API network so that it can connect to the **keystone_admin_ssh** VIP.

4.2.2. Configuring Load Balancing Options

The director provides a method for creating an Overcloud where an external load balancers hosts the virtual IPs instead of HAProxy managing them internally. This configuration assumes a number of virtual IPs are configured on the external load balancer, one per isolated network, plus one for the Redis service, before the Overcloud deployment starts. Some of the Virtual IPs can be identical if the Overcloud node NICs configuration permits so.

You previously configured the external load balancer using settings from the previous chapter. These settings include the IPs that the director assigns to the Overcloud nodes and uses for service configuration.

The following is an example Heat environment file (**external-lb.yaml**) that contains the Overcloud configuration for using the external load balancer:

```
parameter_defaults:
  ControlFixedIPs: [{'ip_address':'192.0.2.250'}]
  PublicVirtualFixedIPs: [{'ip_address':'172.16.23.250'}]
  InternalApiVirtualFixedIPs: [{'ip_address':'172.16.20.250'}]
  StorageVirtualFixedIPs: [{'ip_address':'172.16.21.250'}]
  StorageMgmtVirtualFixedIPs: [{'ip_address':'172.16.19.250'}]
  RedisVirtualFixedIPs: [{'ip_address':'172.16.20.249'}]
  # IPs assignments for the Overcloud Controller nodes. Ensure these IPs are from each respective
allocation pools defined in the network environment file.
  ControllerIPs:
    external:
    - 172.16.23.150
    - 172.16.23.151
    - 172.16.23.152
    internal_api:
    - 172.16.20.150
    - 172.16.20.151
    - 172.16.20.152
    storage:
    - 172.16.21.150
    - 172.16.21.151
    - 172.16.21.152
    storage_mgmt:
    - 172.16.19.150
    - 172.16.19.151
    - 172.16.19.152
    tenant:
    - 172.16.22.150
    - 172.16.22.151
    - 172.16.22.152
    # CIDRs
    external_cidr: "24"
    internal_api_cidr: "24"
    storage_cidr: "24"
    storage_mgmt_cidr: "24"
```

```
  tenant_cidr: "24"
RedisPassword: p@55w0rd!
ServiceNetMap:
  NeutronTenantNetwork: tenant
  CeilometerApiNetwork: internal_api
  AodhApiNetwork: internal_api
  GnocchiApiNetwork: internal_api
  MongoDbNetwork: internal_api
  CinderApiNetwork: internal_api
  CinderIscsiNetwork: storage
  GlanceApiNetwork: storage
  GlanceRegistryNetwork: internal_api
  KeystoneAdminApiNetwork: internal_api
  KeystonePublicApiNetwork: internal_api
  NeutronApiNetwork: internal_api
  HeatApiNetwork: internal_api
  NovaApiNetwork: internal_api
  NovaMetadataNetwork: internal_api
  NovaVncProxyNetwork: internal_api
  SwiftMgmtNetwork: storage_mgmt
  SwiftProxyNetwork: storage
  HorizonNetwork: internal_api
  MemcachedNetwork: internal_api
  RabbitMqNetwork: internal_api
  RedisNetwork: internal_api
  MysqlNetwork: internal_api
  CephClusterNetwork: storage_mgmt
  CephPublicNetwork: storage
  ControllerHostnameResolveNetwork: internal_api
  ComputeHostnameResolveNetwork: internal_api
  BlockStorageHostnameResolveNetwork: internal_api
  ObjectStorageHostnameResolveNetwork: internal_api
  CephStorageHostnameResolveNetwork: storage
```

The **parameter_defaults** section contains the VIP and IP assignments for each network on OpenStack. These settings must match the same IP configuration for each service on the load balancer. This section also defines an administrative password for the Redis service (**RedisPassword**). This section also contains the **ServiceNetMap** parameter, which maps each OpenStack service to a specific networks. The load balancing configuration requires this services remap.

4.3. CONFIGURING SSL FOR LOAD BALANCING

The Overcloud uses unencrypted endpoints for its services by default. This means the Overcloud configuration requires an additional environment file to enable SSL/TLS for its endpoints.

NOTE

Ensure your external load balancer has a copy of your SSL certificate and key installed.

An overcloud with an internal load balancer uses the **enable-tls.yaml** environment file from the Heat template collection to install the key and certificate pair. An external load balancer does not require this file. However, the overcloud still requires a list of SSL/TLS endpoints. Depending on whether you are using an IP address or domain name to access the public endpoints, use the following environment file:

- If using a DNS name for accessing the public endpoints, use **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-dns.yaml**
- If using a IP address for accessing the public endpoints, use **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-ip.yaml**

If using a self-signed certificate or the certificate signer is not in the default trust store on the Overcloud image, inject the certificate into the Overcloud image. Copy the **inject-trust-anchor.yaml** environment file from the Heat template collection:

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/inject-trust-anchor.yaml
~/templates/.
```

Edit this file and make the following changes for these parameters:

SSLRootCertificate

Copy the contents of the root certificate authority file into the **SSLRootCertificate** parameter. For example:

```
parameter_defaults:
  SSLRootCertificate: |
  -----BEGIN CERTIFICATE-----
  MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQEBCwUAMFgxCzAJBgNV
  ...
  sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvp
  -----END CERTIFICATE-----
```

IMPORTANT

The certificate authority contents require the same indentation level for all new lines.

OS::TripleO::NodeTLSCAData

Change the resource URL for **OS::TripleO::NodeTLSCAData:`** to an absolute URL:

```
resource_registry:
  OS::TripleO::NodeTLSCAData: /usr/share/openstack-tripleo-heat-
templates/puppet/extraconfig/tls/ca-inject.yaml
```

If using a DNS hostname to access the Overcloud through SSL/TLS, create a new environment file (**~/templates/cloudname.yaml**) to define the hostname of the Overcloud's endpoints. Use the following parameters:

CloudName

The DNS hostname for the Overcloud endpoints.

DnsServers

A list of DNS servers to use. The configured DNS servers must contain an entry for the configured CloudName that matches the IP for the Public API.

The following is an example of the contents for this file:

```
parameter_defaults:
 CloudName: overcloud.example.com
 DnsServers: 10.0.0.1
```

The deployment command (**openstack overcloud deploy**) in Section 4.4, "Creating the Overcloud" uses the **-e** option to add environment files. Add the environment files from this section in the following order:

- The environment file with the SSL/TLS endpoints (**tls-endpoints-public-dns.yaml** or **tls-endpoints-public-ip.yaml**)
- The environment file to set the DNS hostname (**cloudname.yaml**)
- The environment file to inject the root certificate authority (**inject-trust-anchor.yaml**)

For example:

```
$ openstack overcloud deploy --templates [...] -e /usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-dns.yaml -e ~/templates/cloudname.yaml -e ~/templates/inject-trust-anchor.yaml
```

4.4. CREATING THE OVERCLOUD

The creation of an Overcloud that uses an external load balancer requires additional arguments to the **openstack overcloud deploy** command. For example:

```
$ openstack overcloud deploy --templates -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml -e ~/network-environment.yaml  -e /usr/share/openstack-tripleo-heat-templates/environments/external-loadbalancer-vip.yaml -e ~/external-lb.yaml --control-scale 3 --compute-scale 1 --control-flavor control --compute-flavor compute [ADDITIONAL OPTIONS]
```

The above command uses the following options:

- **--templates** - Creates the Overcloud from the default Heat template collection.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml** - Adds an additional environment file to the Overcloud deployment. In this case, it is an environment file that initializes the resources network isolation configuration.
- **-e ~/network-environment.yaml** - Adds an additional environment file to the Overcloud deployment. In this case, it is the network environment file created previously.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/external-loadbalancer-vip.yaml** - Adds an additional environment file to the Overcloud deployment. In this case, it is an environment file that initializes the external load balancing configuration. Note that you should include this environment file after the network configuration files.
- **-e ~/external-lb.yaml** - Adds an additional environment file to the Overcloud deployment. In this case, it is the environment file containing our external load balancer configuration. Note that you should include this environment file after the network configuration files.
- **--control-scale 3** - Scale the Controller nodes to three.
- **--compute-scale 3** - Scale the Compute nodes to three.

- **--control-flavor control** - Use a specific flavor for the Controller nodes.
- **--compute-flavor compute** - Use a specific flavor for the Compute nodes.

NOTE

For a full list of options, run:

```
$ openstack help overcloud deploy
```

See also the Director Installation and Usage guide for parameter examples.

The Overcloud creation process begins and the director provisions your nodes. This process takes some time to complete. To view the status of the Overcloud creation, open a separate terminal as the stack user and run:

```
$ source ~/stackrc
$ heat stack-list --show-nested
```

4.5. ACCESSING THE OVERCLOUD

The director generates a script to configure and help authenticate interactions with your Overcloud from the director host. The director saves this file, overcloudrc, in your stack user's home directory. Run the following command to use this file:

```
$ source ~/overcloudrc
```

This loads the necessary environment variables to interact with your Overcloud from the director host's CLI. To return to interacting with the director's host, run the following command:

```
$ source ~/stackrc
```

4.6. COMPLETING THE OVERCLOUD CONFIGURATION

This concludes the creation of the Advanced Overcloud.

For fencing the high availability cluster, see the Director Installation and Usage guide.

For post-creation functions, see the Director Installation and Usage guide.

APPENDIX A. EXAMPLE DEFAULT HAPROXY CONFIGURATION

The following is an example of the default configuration file for HAProxy on the Overcloud's Controller nodes. This file is located at **/etc/haproxy/haproxy.conf** on each Controller node.

```
global
  daemon
  group  haproxy
  log  /dev/log local0
  maxconn  10000
  pidfile  /var/run/haproxy.pid
  user  haproxy

defaults
  log  global
  mode  tcp
  retries  3
  timeout  http-request 10s
  timeout  queue 1m
  timeout  connect 10s
  timeout  client 1m
  timeout  server 1m
  timeout  check 10s

listen aodh
  bind 172.16.20.250:8042
  bind 172.16.20.250:8042
  mode http
  server overcloud-controller-0 172.16.20.150:8042 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8042 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.252:8042 check fall 5 inter 2000 rise 2

listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter 2000 rise 2

listen cinder
  bind 172.16.20.250:8776
  bind 172.16.23.250:8776
  server overcloud-controller-0 172.16.20.150:8776 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8776 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8776 check fall 5 inter 2000 rise 2

listen glance_api
  bind 172.16.23.250:9292
  bind 172.16.21.250:9292
  server overcloud-controller-0 172.16.21.150:9292 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.21.151:9292 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.21.152:9292 check fall 5 inter 2000 rise 2

listen glance_registry
```

```
  bind 172.16.20.250:9191
  server overcloud-controller-0 172.16.20.150:9191 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:9191 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:9191 check fall 5 inter 2000 rise 2

listen gnocchi
  bind 172.16.23.250:8041
  bind 172.16.21.250:8041
  mode http
  server overcloud-controller-0 172.16.20.150:8041 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8041 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8041 check fall 5 inter 2000 rise 2

listen heat_api
  bind 172.16.20.250:8004
  bind 172.16.23.250:8004
  mode http
  server overcloud-controller-0 172.16.20.150:8004 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8004 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8004 check fall 5 inter 2000 rise 2

listen heat_cfn
  bind 172.16.20.250:8000
  bind 172.16.23.250:8000
  server overcloud-controller-0 172.16.20.150:8000 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.152:8000 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.151:8000 check fall 5 inter 2000 rise 2

listen heat_cloudwatch
  bind 172.16.20.250:8003
  bind 172.16.23.250:8003
  server overcloud-controller-0 172.16.20.150:8003 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8003 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8003 check fall 5 inter 2000 rise 2

listen horizon
  bind 172.16.20.250:80
  bind 172.16.23.250:80
  mode http
  cookie SERVERID insert indirect nocache
  server overcloud-controller-0 172.16.20.150:80 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:80 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:80 check fall 5 inter 2000 rise 2

listen keystone_admin
  bind 172.16.23.250:35357
  bind 172.16.20.250:35357
  server overcloud-controller-0 172.16.20.150:35357 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:35357 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:35357 check fall 5 inter 2000 rise 2

listen keystone_admin_ssh
  bind 172.16.20.250:22
  server overcloud-controller-0 172.16.20.150:22 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:22 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:22 check fall 5 inter 2000 rise 2
```

```
listen keystone_public
  bind 172.16.20.250:5000
  bind 172.16.23.250:5000
  server overcloud-controller-0 172.16.20.150:5000 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:5000 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:5000 check fall 5 inter 2000 rise 2

listen mysql
  bind 172.16.20.250:3306
  option tcpka
  option httpchk
  stick on dst
  stick-table type ip size 1000
  timeout client 0
  timeout server 0
  server overcloud-controller-0 172.16.20.150:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2
  server overcloud-controller-1 172.16.20.151:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2
  server overcloud-controller-2 172.16.20.152:3306 backup check fall 5 inter 2000 on-marked-down
shutdown-sessions port 9200 rise 2

listen neutron
  bind 172.16.20.250:9696
  bind 172.16.23.250:9696
  server overcloud-controller-0 172.16.20.150:9696 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:9696 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:9696 check fall 5 inter 2000 rise 2

listen nova_ec2
  bind 172.16.20.250:8773
  bind 172.16.23.250:8773
  server overcloud-controller-0 172.16.20.150:8773 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8773 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8773 check fall 5 inter 2000 rise 2

listen nova_metadata
  bind 172.16.20.250:8775
  server overcloud-controller-0 172.16.20.150:8775 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8775 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8775 check fall 5 inter 2000 rise 2

listen nova_novncproxy
  bind 172.16.20.250:6080
  bind 172.16.23.250:6080
  balance source
  server overcloud-controller-0 172.16.20.150:6080 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:6080 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:6080 check fall 5 inter 2000 rise 2

listen nova_osapi
  bind 172.16.20.250:8774
  bind 172.16.23.250:8774
  server overcloud-controller-0 172.16.20.150:8774 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8774 check fall 5 inter 2000 rise 2
```

```
  server overcloud-controller-2 172.16.20.152:8774 check fall 5 inter 2000 rise 2

listen nova_placement
  bind 172.16.20.250:8778
  bind 172.16.23.250:8778
  mode http
  server overcloud-controller-0 172.16.20.150:8778 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8778 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8778 check fall 5 inter 2000 rise 2

listen panko
  bind 172.16.20.250:8779 transparent
  bind 172.16.23.250:8779 transparent
  server overcloud-controller-0 172.16.20.150:8779 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:8779 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:8779 check fall 5 inter 2000 rise 2

listen redis
  bind 172.16.20.249:6379
  balance first
  option tcp-check
  tcp-check send AUTH\ p@55w0rd!\r\n
  tcp-check send PING\r\n
  tcp-check expect string +PONG
  tcp-check send info\ replication\r\n
  tcp-check expect string role:master
  tcp-check send QUIT\r\n
  tcp-check expect string +OK
  server overcloud-controller-0 172.16.20.150:6379 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.20.151:6379 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.20.152:6379 check fall 5 inter 2000 rise 2

listen swift_proxy_server
  bind 172.16.23.250:8080
  bind 172.16.21.250:8080
  server overcloud-controller-0 172.16.21.150:8080 check fall 5 inter 2000 rise 2
  server overcloud-controller-1 172.16.21.151:8080 check fall 5 inter 2000 rise 2
  server overcloud-controller-2 172.16.21.152:8080 check fall 5 inter 2000 rise 2
```