



# Red Hat OpenStack Platform 17.0

## External Load Balancing for the Overcloud

Configuring a Red Hat OpenStack Platform environment to use an external load balancer



# Red Hat OpenStack Platform 17.0 External Load Balancing for the Overcloud

---

Configuring a Red Hat OpenStack Platform environment to use an external load balancer

OpenStack Team  
rhos-docs@redhat.com

## Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

Configure a Red Hat OpenStack Platform (RHOSP) environment to use an external load balancer for the overcloud. This includes configuration guidelines for your load balancer and configuration of the overcloud with Red Hat OpenStack Platform director.

---

## Table of Contents

<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>3</b>
<b>PROVIDING FEEDBACK ON RED HAT DOCUMENTATION</b> .....	<b>4</b>
<b>CHAPTER 1. CONFIGURING THE OVERCLOUD TO USE AN EXTERNAL LOAD BALANCER</b> .....	<b>5</b>
1.1. PREPARING YOUR ENVIRONMENT FOR AN EXTERNAL LOAD BALANCER	5
1.2. CONFIGURING THE OVERCLOUD NETWORK FOR AN EXTERNAL LOAD BALANCER	7
1.3. CREATING AN EXTERNAL LOAD BALANCER ENVIRONMENT FILE	8
1.4. CONFIGURING SSL FOR EXTERNAL LOAD BALANCING	10
1.5. DEPLOYING THE OVERCLOUD WITH AN EXTERNAL LOAD BALANCER	11
1.6. ADDITIONAL RESOURCES	12
<b>CHAPTER 2. EXAMPLE CONFIGURATION: OVERCLOUD WITH AN EXTERNAL HAPROXY LOAD BALANCER</b>	<b>13</b>
2.1. EXAMPLE HAPROXY CONFIGURATION FILE	13
2.1.1. Global configuration parameters: Example HAProxy configuration file	17
2.1.2. Default values configuration parameters: Example HAProxy configuration file	18
2.1.3. Service-level configuration parameters: Example HAProxy configuration file	18
2.2. CONFIGURATION PARAMETERS FOR SERVICES THAT USE LOAD BALANCING	19



## MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

## PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

### Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.



# CHAPTER 1. CONFIGURING THE OVERCLOUD TO USE AN EXTERNAL LOAD BALANCER

In Red Hat OpenStack Platform (RHOSP), the overcloud uses multiple Controller nodes together as a high availability cluster to ensure maximum operational performance for your OpenStack services. The cluster also provides load balancing for OpenStack services, which evenly distributes traffic to the Controller nodes and reduces server overload for each node.

By default, the overcloud uses an open source tool called HAProxy to manage load balancing. HAProxy load balances traffic to the Controller nodes that run OpenStack services. The **haproxy** package contains the **haproxy** daemon that listens to incoming traffic, and includes logging features and sample configurations.

The overcloud also uses the high availability resource manager Pacemaker to control HAProxy as a highly available service. This means that HAProxy runs on each Controller node and distributes traffic according to a set of rules that you define in each configuration.

You can also use an external load balancer to perform this distribution. For example, your organization might use a dedicated hardware-based load balancer to handle traffic distribution to the Controller nodes. To define the configuration for an external load balancer and the overcloud creation, you perform the following processes:

1. Install and configure an external load balancer.
2. Configure and deploy the overcloud with heat template parameters to integrate the overcloud with the external load balancer. This requires the IP addresses of the load balancer and of the potential nodes.

Before you configure your overcloud to use an external load balancer, ensure that you deploy and run high availability on the overcloud.

## 1.1. PREPARING YOUR ENVIRONMENT FOR AN EXTERNAL LOAD BALANCER

To prepare your environment for an external load balancer, first create a node definition template and register blank nodes with director. Then, inspect the hardware of all nodes and manually tag nodes into profiles.

Use the following workflow to prepare your environment:

- Create a node definition template and register blank nodes with Red Hat OpenStack Platform director. A node definition template **instackenv.json** is a JSON format file and contains the hardware and power management details to register nodes.
- Inspect the hardware of all nodes. This ensures that all nodes are in a manageable state.
- Manually tag nodes into profiles. These profile tags match the nodes to flavors. The flavors are then assigned to a deployment role.

### Procedure

1. Log in to the director host as the **stack** user and source the director credentials:

```
$ source ~/stackrc
```

2. Create a node definition template **instackenv.json** and copy and edit the following example based on your environment:

```
{
  "nodes":[
    {
      "mac":[
        "bb:bb:bb:bb:bb:bb"
      ],
      "cpu":"4",
      "memory":"6144",
      "disk":"40",
      "arch":"x86_64",
      "pm_type":"ipmi",
      "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":"ipmi",
      "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":"ipmi",
      "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":"ipmi",
      "pm_user":"admin",
      "pm_password":"p@55w0rd!",
      "pm_addr":"192.0.2.208"
    }
  ]
}
```

```

    }
  ]
}

```

3. Save the file to the home directory of the **stack** user, **/home/stack/instackenv.json**, then import it into director and register the nodes to the director:

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

4. Assign the kernel and ramdisk images to all nodes:

```
$ openstack overcloud node configure
```

5. Inspect the hardware attributes of each node:

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



### IMPORTANT

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

6. Get the list of your nodes to identify their UUIDs:

```
$ openstack baremetal node list
```

7. Manually tag each node to a specific profile by adding a profile option in 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:

```

$ openstack baremetal node set 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 --property
capabilities='profile:control,boot_option:local'
$ openstack baremetal node set 6faba1a9-e2d8-4b7c-95a2-c7fbd12129a --property
capabilities='profile:control,boot_option:local'
$ openstack baremetal node set 5e3b2f50-fcd9-4404-b0a2-59d79924b38e --property
capabilities='profile:control,boot_option:local'
$ openstack baremetal node set 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13 --property
capabilities='profile:compute,boot_option:local'

```

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

#### Additional resources

- [Planning your overcloud](#)

## 1.2. CONFIGURING THE OVERCLOUD NETWORK FOR AN EXTERNAL LOAD BALANCER

To configure the network for the overcloud, isolate your services to use specific network traffic and then configure the network environment file for your local environment. This file is a heat environment file that describes the overcloud network environment, points to the network interface configuration templates, and defines the subnets and VLANs for your network and the IP address ranges.

## Procedure

1. To configure the node interfaces for each role, customize the following network interface templates:

- To configure a single NIC with VLANs for each role, use the example templates in the following directory:

```
/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans
```

- To configure bonded NICs for each role, use the example templates in the following directory:

```
/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans
```

2. Create a network environment file by copying the file from **/home/stack/network-environment.yaml** and editing the content based on your environment.

## Additional resources

- [Basic network isolation](#)
- [Custom composable networks](#)
- [Custom network interface templates](#)
- [Overcloud networks](#)

## 1.3. CREATING AN EXTERNAL LOAD BALANCER ENVIRONMENT FILE

To deploy an overcloud with an external load balancer, create a new environment file with the required configuration. In this example file, several virtual IPs are configured on the external load balancer, one virtual IP on each isolated network, and one for the Redis service, before the overcloud deployment starts. Some of the virtual IPs can be identical if the overcloud node NICs configuration supports the configuration.

## Procedure

- Use the following example environment file **external-lb.yaml** to create the environment file, and edit the content based on your environment.

```
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
```



## NOTE

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

## 1.4. CONFIGURING SSL FOR EXTERNAL LOAD BALANCING

To configure encrypted endpoints for the external load balancer, create additional environment files that enable SSL to access endpoints and then install a copy of your SSL certificate and key on your external load balancing server. By default, the overcloud uses unencrypted endpoints services.

### Prerequisites

- If you are using an IP address or domain name to access the public endpoints, choose one of the following environment files to include in your overcloud deployment:
  - To access the public endpoints with a domain name service (DNS), use the file **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-dns.yaml**.
  - To access the public endpoints with an IP address, use the file **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-ip.yaml**.

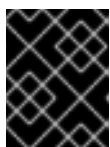
### Procedure

1. If you use a self-signed certificate or if the certificate signer is not in the default trust store on the overcloud image, inject the certificate into the overcloud image by copying 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/
```

2. Open the file in a text editor and copy the contents of the root certificate authority file to the **SSLRootCertificate** parameter:

```
parameter_defaults:
  SSLRootCertificate: |
    -----BEGIN CERTIFICATE-----
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQEBCwUAMFgx CzAJBgNV
    ...
    sFW3S2roS4X0Af/kSSD8mIBBTFTCMBAj6rtLBKLaQbIx Eplzrgvp
    -----END CERTIFICATE-----
```



## IMPORTANT

The certificate authority content requires the same indentation level for all new lines.

3. Change the resource URL for the **OS::TripleO::NodeTLSCAData:** parameter to an absolute URL:

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

4. Optional: If you use a DNS hostname to access the overcloud through SSL/TLS, create a new environment file **~/templates/cloudname.yaml** and define the hostname of the overcloud endpoints:

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

Replace **overcloud.example.com** with the DNS hostname for the overcloud endpoints.

## 1.5. DEPLOYING THE OVERCLOUD WITH AN EXTERNAL LOAD BALANCER

To deploy an overcloud that uses an external load balancer, run the **openstack overcloud deploy** and include the additional environment files and configuration files for the external load balancer.

### Prerequisites

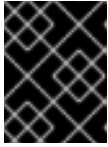
- Environment is prepared for an external load balancer. For more information about how to prepare your environment, see [Section 1.1, "Preparing your environment for an external load balancer"](#)
- Overcloud network is configured for an external load balancer. For information about how to configure your network, see [Section 1.2, "Configuring the overcloud network for an external load balancer"](#)
- External load balancer environment file is prepared. For information about how to create the environment file, see [Section 1.3, "Creating an external load balancer environment file"](#)
- SSL is configured for external load balancing. For information about how to configure SSL for external load balancing, see [Section 1.4, "Configuring SSL for external load balancing"](#)

### Procedure

1. Deploy the overcloud with all the environment and configuration files for an external load balancer:

```
$ 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 /
  -e <SSL/TLS endpoint environment file> /
  -e <DNS hostname environment file> /
  -e <root certificate injection environment file> /
  -e <additional_options_if_needed>
```

Replace the values in angle brackets `<>` with the file paths you defined for your environment.



### IMPORTANT

You must add the network environment files to the command in the order listed in this example.

This command includes the following environment files:

- **network-isolation.yaml**: Network isolation configuration file.
- **network-environment.yaml**: Network configuration file.
- **external-loadbalancer-vip.yaml**: External load balancing virtual IP addresses configuration file.
- **external-lb.yaml**: External load balancer configuration file. You can also set the following options for this file and adjust the values for your environment:
  - **--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.
- SSL/TLS environment files:
  - **SSL/TLS endpoint environment file**: Environment file that defines how to connect to public endpoint. Use **tls-endpoints-public-dns.yaml** or **tls-endpoints-public-ip.yaml**.
  - (Optional) **DNS hostname environment file**: The environment file to set the DNS hostname.
  - **Root certificate injection environment file**: The environment file to inject the root certificate authority.

During the overcloud deployment process, Red Hat OpenStack Platform director provisions your nodes. This process takes some time to complete.

2. To view the status of the overcloud deployment, enter the following commands:

```
$ source ~/stackrc
$ openstack stack list --nested
```

## 1.6. ADDITIONAL RESOURCES

- [Load balancing traffic with HAProxy](#).



## CHAPTER 2. EXAMPLE CONFIGURATION: OVERCLOUD WITH AN EXTERNAL HAPROXY LOAD BALANCER

This example configuration shows an overcloud that uses a federated HAProxy server to provide external load balancing. You can choose a different external load balancer based on your environment requirements.

The example configuration includes the following elements:

- An external load balancing server that runs HAProxy.
- One Red Hat OpenStack Platform (RHOSP) director node.
- An overcloud that consists of 3 Controller nodes in a highly available cluster and 1 Compute node.
- Network isolation with VLANs.

The example 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 include IP assignments for the Controller nodes and virtual IPs that the load balancer binds to OpenStack services.

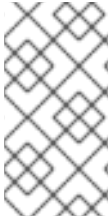
### 2.1. EXAMPLE HAPROXY CONFIGURATION FILE

The example file shows the internal HAProxy configuration parameters. You can use the sample configuration parameters as a basis for configuring your external load balancer.

The HAProxy configuration file contains the following sections:

- Global configuration
- Defaults configuration
- Services configurations

Director provides this configuration in the **/etc/haproxy/haproxy.conf** file on each Controller node for non-containerized environments, and in the **/var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg** file for containerized environments.

**NOTE**

In addition to the global, default, and services parameters, you must also configure other HAProxy parameters. For more information about HAProxy parameters, see the *HAProxy Configuration Manual* located in `/usr/share/doc/haproxy-*/configuration.txt` on the Controller nodes or on any system where the **haproxy** package is installed.

**Example HAProxy configuration file**

```
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\r\n p@55w0rd!\r\n
tcp-check send PING\r\n
tcp-check expect string +PONG
tcp-check send info\r\n 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

```

### 2.1.1. Global configuration parameters: Example HAProxy configuration file

The global configuration parameters section defines a set of process-wide parameters for the load balancer. You can use the example parameters from the configuration file to configure your external load balancer. Adjust the parameter values based on your environment.

#### Global configuration parameters

```

global
daemon
user haproxy
group haproxy

```

```
log /dev/log local0
maxconn 10000
pidfile /var/run/haproxy.pid
```

The example shows the following parameters:

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

### 2.1.2. Default values configuration parameters: Example HAProxy configuration file

The default values configuration parameters section defines a set of default values to use when running the external load balancer services. You can use the example parameters from the configuration file to configure your external load balancer. Adjust the parameter values based on your environment.

#### Default values configuration parameters

```
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
```

The example shows the following parameters:

- **log**: Enables logging for the service. The **global** value means that the logging functions use the **log** parameters from the **global** section.
- **mode**: Defines 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.

### 2.1.3. Service-level configuration parameters: Example HAProxy configuration file

The service-level configuration parameters section defines a set of parameters to use when load balancing traffic to a specific Red Hat OpenStack Platform (RHOSP) service. You can use the example parameters from the configuration file to configure your external load balancer. Adjust the parameter values based on your environment, and copy the section for each service that you want to load balance.

#### Service-level configuration parameters

This example shows the configuration parameters for the **ceilometer** service.

```
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
```

Each service that you want to load balance must correspond to a section in the configuration file. Each service configuration includes the following parameters:

- **listen**: The name of the service that listens for requests.
- **bind**: The IP address and TCP port number the on which the service listens. Each service binds a different address that represents a different network traffic type.
- **server**: The name of each server that provides the service, the server IP address and listening port, and connection parameters:
- **check**: (Optional) Enables health checks.
- **fall 5**: (Optional) After five failed health checks, the service is considered offline.
- **inter 2000**: (Optional) The interval between two consecutive health checks set to 2000 milliseconds, or 2 seconds.
- **rise 2**: (Optional) After two successful health checks, the service is considered operational.

In the **ceilometer** example, the service identifies the IP addresses and ports on which the ceilometer service is offered as **172.16.20.250:8777** and **172.16.23.250:8777**. HAProxy directs the requests 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).

#### Additional resources

- [Section 2.2, "Configuration parameters for services that use load balancing"](#)

## 2.2. CONFIGURATION PARAMETERS FOR SERVICES THAT USE LOAD BALANCING

For each service in the overcloud that uses load balancing, use the following examples as a guide to configure your external load balancer. Adjust the parameter values based on your environment, and copy the section for each service that you want to load balance.

**NOTE**

Most services use the 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 offline.

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

**aodh**

**Port number:** 8042

**Binds to:** internal\_api, external

**Target network or 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
```

**ceilometer**

**Port number:** 8777

**Binds to:** internal\_api, external

**Target network or 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
```



-

**cinder****Port number:** 8776**Binds to:** internal\_api, external**Target network or 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
```

**glance\_api****Port number:** 9292**Binds to:** storage, external**Target network or 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
```

**glance\_registry****Port number:** 9191**Binds to:** internal\_api**Target network or 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
```

**gnocchi****Port number:** 8041**Binds to:** internal\_api, external**Target network or 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
```

**heat\_api****Port number:** 8004**Binds to:** internal\_api, external**Target network or 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
```

**heat\_cfn**

**Port number:** 8000

**Binds to:** internal\_api, external

**Target network or 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
```

## heat\_cloudwatch

**Port number:** 8003

**Binds to:** internal\_api, external

**Target network or 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
```

## horizon

**Port number:** 80

**Binds to:** internal\_api, external

**Target network or 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
```

**keystone\_admin**

**Port number:** 35357

**Binds to:** internal\_api, external

**Target network or 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
```

**keystone\_admin\_ssh**

**Port number:** 22

**Binds to:** internal\_api

**Target network or 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
```

## keystone\_public

**Port number:** 5000

**Binds to:** internal\_api, external

**Target network or 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
```

## mysql

**Port number:** 3306

**Binds to:** internal\_api

**Target network or 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 only when all other non-backup servers are unavailable.
- If the server is offline, all connections are immediately terminated.
- You must enable the sending of TCP keepalive packets on both sides.
- You must enable HTTP protocol to check on the servers health.
- You can configure a stickiness table to store IP addresses, to help maintain persistence.



### IMPORTANT

The **mysql** service uses Galera to provide a highly available database cluster. Galera supports an active-active configuration, but to avoid lock contention, you must use an active-passive configuration that is enforced by the load balancer.

### 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
```

## neutron

**Port number:** 9696

**Binds to:** internal\_api, external

**Target network or 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
```

## nova\_ec2

**Port number:** 8773

**Binds to:** internal\_api, external

**Target network or 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
```

### nova\_metadata

**Port number:** 8775

**Binds to:** internal\_api

**Target network or 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
```

### nova\_novncproxy

**Port number:** 6080

**Binds to:** internal\_api, external

**Target network or 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 load 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 method also designates the server that receives the request and ensures that the same client IP address always reaches the same server unless server goes down or up. If the hash result changes due to a change in the number of running servers, the load balancer redirects the 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
```

### nova\_osapi

**Port number:** 8774

**Binds to:** internal\_api, external

**Target network or 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
```

## nova\_placement

**Port number:** 8778

**Binds to:** internal\_api, external

**Target network or 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
```

## panko

**Port number:** 8779

**Binds to:** internal\_api, external

**Target network or 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
```

## redis

**Port number:** 6379

**Binds to:** internal\_api (redis service IP)

**Target network or 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 the AUTH method and the Redis administration password. Director normally generates a random password, but you can define a custom Redis password.
- The default balancing method is **round-robin**. However, for this service, use the **first** method. This ensures that the first server that has 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
```

## swift\_proxy\_server

**Port number:** 8080

**Binds to:** storage, external

**Target network or 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
```