

---

# **OpenStack-Ansible Documentation: haproxy\_server role**

*Release 18.1.0.dev297*

**OpenStack-Ansible Contributors**

**Feb 25, 2025**



# CONTENTS

<b>1</b>	<b>Configuring HAProxy (optional)</b>	<b>1</b>
1.1	Making HAProxy highly-available . . . . .	1
1.2	Configuring keepalived ping checks . . . . .	2
1.3	Securing HAProxy communication with SSL certificates . . . . .	3
1.4	Using Certificates from LetsEncrypt . . . . .	3
1.5	Configuring additional services . . . . .	5
1.6	Adding additional global VIP addresses . . . . .	5
1.7	Controlling HAProxy front-end binding . . . . .	6
1.8	Adding Access Control Lists to HAProxy front end . . . . .	7
1.9	Adding prometheus metrics to haproxy . . . . .	7
<b>2</b>	<b>Default variables</b>	<b>9</b>
<b>3</b>	<b>Required variables</b>	<b>17</b>
<b>4</b>	<b>Dependencies</b>	<b>19</b>
<b>5</b>	<b>Example playbook</b>	<b>21</b>



## CONFIGURING HAPROXY (OPTIONAL)

HAProxy provides load balancing services and SSL termination when hardware load balancers are not available for high availability architectures deployed by OpenStack-Ansible. The default HAProxy configuration provides highly- available load balancing services via keepalived if there is more than one host in the `haproxy_hosts` group.

### Important

Ensure you review the services exposed by HAProxy and limit access to these services to trusted users and networks only. For more details, refer to the [Securing network access to OpenStack services](#) section.

### Note

For a successful installation, you require a load balancer. You may prefer to make use of hardware load balancers instead of HAProxy. If hardware load balancers are in use, then implement the load balancing configuration for services prior to executing the deployment.

To deploy HAProxy within your OpenStack-Ansible environment, define target hosts to run HAProxy:

```
haproxy_hosts:
  infra1:
    ip: 172.29.236.101
  infra2:
    ip: 172.29.236.102
  infra3:
    ip: 172.29.236.103
```

There is an example configuration file already provided in `/etc/openstack_deploy/conf.d/haproxy.yml.example`. Rename the file to `haproxy.yml` and configure it with the correct target hosts to use HAProxy in an OpenStack-Ansible deployment.

### 1.1 Making HAProxy highly-available

If multiple hosts are found in the inventory, deploy HAProxy in a highly-available manner by installing keepalived.

To make keepalived work, edit at least the following variables in `user_variables.yml`:

```
haproxy_keepalived_external_vip_cidr: 192.168.0.4/25
haproxy_keepalived_internal_vip_cidr: 172.29.236.54/16
haproxy_keepalived_external_interface: br-flat
haproxy_keepalived_internal_interface: br-mgmt
```

- `haproxy_keepalived_internal_interface` and `haproxy_keepalived_external_interface` represent the interfaces on the deployed node where the keepalived nodes bind the internal and external vip. By default, use `br-mgmt`.
- On the interface listed above, `haproxy_keepalived_internal_vip_cidr` and `haproxy_keepalived_external_vip_cidr` represent the internal and external (respectively) vips (with their prefix length).
- Set additional variables to adapt keepalived in your deployment. Refer to the `user_variables.yml` for more descriptions.

To always deploy (or upgrade to) the latest stable version of keepalived. Edit the `/etc/openstack_deploy/user_variables.yml`:

```
keepalived_use_latest_stable: True
```

The HAProxy nodes have group vars applied that define the configuration of keepalived. This configuration is stored in `group_vars/haproxy_all/keepalived.yml`. It contains the variables needed for the keepalived role (master and backup nodes).

Keepalived pings a public and private IP address to check its status. The default address is `193.0.14.129`. To change this default, set the `keepalived_external_ping_address` and `keepalived_internal_ping_address` variables in the `user_variables.yml` file.

#### Note

The keepalived test works with IPv4 addresses only.

You can adapt keepalived to your environment by either using our override mechanisms (per host with `userspace host_vars`, per group with `userspace "group_vars"`, or globally using the `userspace user_variables.yml` file)

If you wish to deploy multiple haproxy hosts without keepalived and provide your own means for failover between them, edit `/etc/openstack_deploy/user_variables.yml` to skip the deployment of keepalived. To do this, set the following:

```
haproxy_use_keepalived: False
```

## 1.2 Configuring keepalived ping checks

OpenStack-Ansible configures keepalived with a check script that pings an external resource and uses that ping to determine if a node has lost network connectivity. If the pings fail, keepalived fails over to another node and HAProxy serves requests there.

The destination address, ping count and ping interval are configurable via Ansible variables in `/etc/openstack_deploy/user_variables.yml`:

```

keepalived_external_ping_address: # Public IP address to ping
keepalived_internal_ping_address: # Private IP address to ping
keepalived_ping_count: # ICMP packets to send (per interval)
keepalived_ping_interval: # How often ICMP packets are sent

```

By default, OpenStack-Ansible configures keepalived to ping one of the root DNS servers operated by RIPE. You can change this IP address to a different external address or another address on your internal network.

If external connectivity fails, it is important that internal services can still access an HAProxy instance. In a situation, when ping to some external host fails and internal ping is not separated, all keepalived instances enter the fault state despite internal connectivity being still available. Separate ping check for internal and external connectivity ensures that when one instance fails the other VIP remains in operation.

### 1.3 Securing HAProxy communication with SSL certificates

The OpenStack-Ansible project provides the ability to secure HAProxy communications with self-signed or user-provided SSL certificates. By default, self-signed certificates are used with HAProxy. However, you can provide your own certificates by using the following Ansible variables:

```

haproxy_user_ssl_cert: # Path to certificate
haproxy_user_ssl_key: # Path to private key
haproxy_user_ssl_ca_cert: # Path to CA certificate

```

Refer to [Securing services with SSL certificates](#) for more information on these configuration options and how you can provide your own certificates and keys to use with HAProxy. User provided certificates should be folded and formatted at 64 characters long. Single line certificates will not be accepted by HAProxy and will result in SSL validation failures. Please have a look here for information on [converting your certificate to various formats](#).

### 1.4 Using Certificates from LetsEncrypt

If you want to use [LetsEncrypt SSL Service](#) you can activate the feature by providing the following configuration in `/etc/openstack_deploy/user_variables.yml`. Note that this requires that `external_lb_vip_address` in `/etc/openstack_deploy/openstack_user_config.yml` is set to the external DNS address.

The following variables must be set for the haproxy hosts.

```

haproxy_ssl_letsencrypt_enable: True
haproxy_ssl_letsencrypt_email: example@example.com
haproxy_interval: 2000

```

The following variables serve as an example for how to configure a single HAProxy providing SSL termination for a service on the same host, served from 127.0.0.1:80. An additional HAProxy backend is configured which will receive the acme-challenge requests when certificates are renewed.

```

haproxy_service_configs:
  # the external facing service which serves the apache test site, with a acl
  ↪ for LE requests
  - haproxy_service_name: test

```

(continues on next page)

(continued from previous page)

```

haproxy_redirect_http_port: 80                                #redirect port 80
↳to port ssl
haproxy_redirect_scheme: "https if !{ ssl_fc } !{ path_beg /.well-known/
↳acme-challenge/ }" #redirect all non-ssl traffic to ssl except acme-
↳challenge
haproxy_port: 443
haproxy_frontend_acls:                                       #use a frontend
↳ACL specify the backend to use for acme-challenge
letsencrypt-acl:
rule: "path_beg /.well-known/acme-challenge/"
backend_name: letsencrypt
haproxy_ssl: True
haproxy_backend_nodes:                                       #apache is running
↳on locally on 127.0.0.1:80 serving a dummy site
- name: local-test-service
ip_addr: 127.0.0.1
haproxy_balance_type: http
haproxy_backend_port: 80
haproxy_backend_options:
- "httpchk HEAD /" # request to use
↳for health check for the example service

# an internal only service for acme-challenge whose backend is certbot on
↳the haproxy host
- haproxy_service_name: letsencrypt
haproxy_backend_nodes:
- name: localhost
ip_addr: {{ ansible_host }} #certbot binds to
↳the internal IP
backend_rise: 1 #quick rise and
↳fall time for multinode deployment to succeed
backend_fall: 2
haproxy_bind:
- 127.0.0.1 #bind to 127.0.0.1
↳as the local internal address will be used by certbot
haproxy_port: 8888 #certbot is
↳configured with http-01-port to be 8888
haproxy_balance_type: http

```

It is possible to use an HA configuration of HAProxy with certificates initialised and renewed using certbot by setting `haproxy_backend_nodes` for the LetsEncrypt service to include all HAProxy internal addresses. Each HAProxy instance will be checking for certbot running on its own node plus each of the others, and direct any incoming acme-challenge requests to the HAProxy instance which is performing a renewal.

Domains which will be covered by Lets Encrypt certificate are defined with `haproxy_ssl_letsencrypt_domains` variable, which can be set to a list. By default certificate will be issued only for `external_lb_vip_address`.

Another important aspect is defining a list of frontends, for which issued certificate will be used. By default, it is going to be used only for VIPs with type `external`. You can control and define type by



overriding a variable `haproxy_vip_binds`.

It is necessary to configure certbot to bind to the HAProxy node local internal IP address via the `haproxy_ssl_letsencrypt_certbot_bind_address` variable in a H/A setup.

## 1.5 Configuring additional services

Additional haproxy service entries can be configured by setting `haproxy_extra_services` in `/etc/openstack_deploy/user_variables.yml`

For more information on the service dict syntax, please reference `playbooks/vars/configs/haproxy_config.yml`

An example HTTP service could look like:

```
haproxy_extra_services:
- haproxy_service_name: extra-web-service
  haproxy_backend_nodes: "{{ groups['service_group'] | default([]) }}"
  haproxy_ssl: "{{ haproxy_ssl }}"
  haproxy_port: 10000
  haproxy_balance_type: http
  # If backend connections should be secured with SSL (default False)
  haproxy_backend_ssl: True
  haproxy_backend_ca: /path/to/ca/cert.pem
  # Or to use system CA for validation
  # haproxy_backend_ca: True
  # Or if certificate validation should be disabled
  # haproxy_backend_ca: False
```

Additionally, you can specify haproxy services that are not managed in the Ansible inventory by manually specifying their hostnames/IP Addresses:

```
haproxy_extra_services:
- haproxy_service_name: extra-non-inventory-service
  haproxy_backend_nodes:
  - name: nonInvHost01
    ip_addr: 172.0.1.1
  - name: nonInvHost02
    ip_addr: 172.0.1.2
  - name: nonInvHost03
    ip_addr: 172.0.1.3
  haproxy_ssl: "{{ haproxy_ssl }}"
  haproxy_port: 10001
  haproxy_balance_type: http
```

## 1.6 Adding additional global VIP addresses

In some cases, you might need to add additional internal VIP addresses to the load balancer front end. You can use the HAProxy role to add additional VIPs to all front ends by setting them in the `extra_lb_vip_addresses` or `extra_lb_tls_vip_addresses` variables.

The following example shows extra VIP addresses defined in the `user_variables.yml` file:

**extra\_lb\_vip\_addresses:**

- 10.0.0.10
- 192.168.0.10

The following example shows extra VIP addresses with TLS enabled defined in the `user_variables.yml` file:

**extra\_lb\_tls\_vip\_addresses:**

- 10.0.0.10
- 192.168.0.10

## 1.7 Controlling HAProxy front-end binding

HAProxy frontend can bind either to some specific IP (VIP) address or ethernet interface. A variable which controls this behaviour is `haproxy_vip_binds`. It is used for the service, unless `haproxy_bind` is defined on the service level. In that case `service.haproxy_bind` has precedence over `haproxy_vip_binds`.

`haproxy_vip_binds` is generated by the role from other convenience variables, like `haproxy_bind_external_lb_vip_address`, `haproxy_bind_external_lb_vip_interface`, `haproxy_bind_internal_lb_vip_address` and `haproxy_bind_internal_lb_vip_interface`.

Though you still can override `haproxy_vip_binds` to fine-control the binding process of HAProxy instance.

### 1.7.1 Overriding the address haproxy will bind to

In some cases you may want to override the default of having haproxy bind to the addresses specified in `external_lb_vip_address` and `internal_lb_vip_address`. For example if those are hostnames and you want haproxy to bind to IP addresses while preserving the names for TLS- certificates and endpoint URIs.

This can be set in the `user_variables.yml` file:

```
haproxy_bind_external_lb_vip_address: 10.0.0.10
haproxy_bind_internal_lb_vip_address: 192.168.0.10
```

### 1.7.2 Binding haproxy to interface

In some cases it might be more convenient to bind haproxy to the interface rather than a specific IP address. For example, this is handy if you decide to balance load between HAProxy instances using DNS RR, where each HAProxy will have its own VIP which will failover to others.

Binding to the interface can be set by providing following variables in the `user_variables.yml` file:

```
haproxy_bind_external_lb_vip_address: "*"
haproxy_bind_internal_lb_vip_address: "*"
haproxy_bind_external_lb_vip_interface: bond0
haproxy_bind_internal_lb_vip_interface: br-mgmt
```

## 1.8 Adding Access Control Lists to HAProxy front end

Adding ACL rules in HAProxy is easy. You just need to define `haproxy_acls` and add the rules in the variable

Here is an example that shows how to achieve the goal

```
- haproxy_service_name: influxdb-relay
  haproxy_acls:
    write_queries:
      rule: "path_sub -i write"
    read_queries:
      rule: "path_sub -i query"
    backend_name: "influxdb"
```

This will add two acl rules `path_sub -i write` and `path_sub -i query` to the front end and use the backend specified in the rule. If no backend is specified it will use a default `haproxy_service_name` backend.

If a frontend service directs to multiple backend services using ACLs, and a backend service does not require its own corresponding front-end, the `haproxy_backend_only` option can be specified:

```
- haproxy_service_name: influxdb
  haproxy_backend_only: true # Directed by the 'influxdb-relay' service above
  haproxy_backend_nodes:
    - name: influxdb-service
      ip_addr: 10.100.10.10
```

## 1.9 Adding prometheus metrics to haproxy

Since haproxy 2.0 its possible to exposes prometheus metrics. <https://www.haproxy.com/blog/haproxy-exposes-a-prometheus-metrics-endpoint/> if you need to create a frontend for it you can use the `haproxy_frontend_only` option:

```
- haproxy_service_name: prometheus-metrics
  haproxy_port: 8404
  haproxy_bind:
    - '127.0.0.1'
  haproxy_whitelist_networks: "{{ haproxy_whitelist_networks }}"
  haproxy_frontend_only: True
  haproxy_frontend_raw:
    - 'http-request use-service prometheus-exporter if { path /metrics }'
  haproxy_service_enabled: True
  haproxy_balance_type: 'http'
```

This Ansible role installs the HAProxy Load Balancer service.

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



## DEFAULT VARIABLES

```
# Validate Certificates when downloading hatop. May be set to "no" when proxy_
↪server
# is intercepting the certificates.
haproxy_hatop_download_validate_certs: true

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

## Haproxy Configuration
haproxy_rise: 3
haproxy_fall: 3
haproxy_interval: 12000

## Haproxy Stats
haproxy_stats_enabled: false
haproxy_stats_bind_address: 127.0.0.1
haproxy_stats_port: 1936
haproxy_stats_ssl: "{{ haproxy_ssl }}"
# haproxy_stats_ssl_cert_path: "{{ haproxy_ssl_cert_path }}/"
↪somecustomstatscert.pem"
# haproxy_stats_ssl_client_cert_ca: "{{ haproxy_ssl_cert_path }}/"
↪somecustomrootca.pem"
haproxy_username: admin
haproxy_stats_password: secrete
haproxy_stats_refresh_interval: 60
# Prometheus stats are supported from HAProxy v2
# Stats must be enabled above before this can be used
haproxy_stats_prometheus_enabled: false

# Default haproxy backup nodes to empty list so this doesn't have to be
# defined for each service.
haproxy_backup_nodes: []

# Configuration lines to write directly into all frontends
haproxy_frontend_extra_raw: []
haproxy_frontend_redirect_extra_raw: "{{ haproxy_frontend_extra_raw }}"
```

(continues on next page)

(continued from previous page)

```

# Default values for enabling HTTP/2 support
# Note, that while HTTP/2 will be enabled on frontends that are covered with
↪ TLS,
# backends can be configured to use HTTP/2 regardless of TLS.
haproxy_frontend_h2: true
haproxy_backend_h2: false

haproxy_service_configs: []
# Example:
# haproxy_service_configs:
#   - haproxy_service_name: haproxy_all
#     haproxy_backend_nodes: "{{ groups['haproxy_all'][0] }}"
#     # haproxy_backup_nodes: "{{ groups['haproxy_all'][1:] }}"
#     haproxy_port: 80
#     haproxy_balance_type: http
#     haproxy_backend_options:
#       - "forwardfor"
#       - "httpchk"
#       - "httplog"
#     haproxy_backend_server_options:
#       - "inter 3000"           # a contrived example, there are many
↪ server config options possible
#     haproxy_acls:
#       allow_list:
#         rule: "src 127.0.0.1/8 192.168.0.0/16 172.16.0.0/12 10.0.0.0/8"
#         backend_name: "mybackend"
#     haproxy_frontend_h2: True
#     haproxy_backend_h2: False
#     haproxy_frontend_acls:
#       letsencrypt-acl:
#         rule: "path_beg /.well-known/acme-challenge/"
#         backend_name: letsencrypt
#     haproxy_stick_table:
#       - "stick-table type ipv6 size 256k expire 10s store http_err_
↪ rate(10s)"
#       - "http-request track-sc0 src"
#       - "http-request deny deny_status 429 if { sc_http_err_rate(0) gt 20 }
↪ !{ src 10.0.0.0/8 } !{ src 172.16.0.0/12 } !{ src 192.168.0.0/16 }"
#       # https://www.haproxy.com/blog/haproxy-exposes-a-prometheus-metrics-
↪ endpoint/
#     - haproxy_service_name: prometheus-metrics
#     haproxy_port: 8404
#     haproxy_bind:
#       - '127.0.0.1'
#     haproxy_allowlist_networks: "{{ groups['haproxy_allowlist_networks'] }}"
#     haproxy_frontend_only: True
#     haproxy_balance_type: "http"
#     haproxy_frontend_raw:
#       - 'http-request use-service prometheus-exporter if { path /metrics }'

```

(continues on next page)

(continued from previous page)

```

#     haproxy_service_enabled: True

# HAProxy maps (unrelated keys are omitted but are required as the previous_
↪service example)
# Example:
# haproxy_service_configs:
#   - state: present                               # state 'absent' will remove map_
↪entries defined in this service
#     haproxy_service_enabled: true                # haproxy_service_enabled 'false'_
↪will remove map entries defined in this service
#     haproxy_service_name: "one"
#     haproxy_maps:
#       - 'use_backend %[req.hdr(host),lower,map(/etc/haproxy/route.map)]'
#     haproxy_map_entries:
#       - name: 'route'                             # this service contributes entries_
↪to the map called 'route'
#         order: 10                                 # prefix the name of the map_
↪fragment with this string to control ordering of the assembled map
#         entries:
#           - compute.example.com nova-api
#           - dashboard.example.com horizon
#     - haproxy_service_name: "two"
#     - haproxy_service_name: "three"
#     haproxy_map_entries:
#       - name: 'route'                             # this service contributes to the_
↪map called 'route'
#         entries:
#           - s3.example.com radosgw
#           - sso.example.com keycloak
#       - name: 'rate'                               # and also to the map called 'rate'
#         state: present                             # individual map entries can be_
↪removed with state 'absent'
#         entries:
#           - /api/foo 20
#           - /api/bar 40
#
# Results:
#
# /etc/haproxy/route.map
#   s3.example.com radosgw
#   sso.example.com keycloak
#   compute.example.com nova-api
#   dashboard.example.com horizon
#
# /etc/haproxy/rate.map
#   /api/foo 20
#   /api/bar 40

galera_monitoring_user: monitoring

```

(continues on next page)

(continued from previous page)

```

haproxy_bind_on_non_local: false

## haproxy SSL
haproxy_ssl: true
haproxy_ssl_all_vips: false
haproxy_ssl_dh_param: 2048
haproxy_ssl_cert_path: /etc/haproxy/ssl
haproxy_ssl_bind_options: "ssl-min-ver TLSv1.2 prefer-client-ciphers"
haproxy_ssl_server_options: "ssl-min-ver TLSv1.2"
# TLS v1.2 and below
haproxy_ssl_cipher_suite_tls12: >-
  {{ haproxy_ssl_cipher_suite | default(ssl_cipher_suite_tls12 | default(
  ↳ 'ECDH+AESGCM:ECDH+CHACHA20:ECDH+AES256:ECDH+AES128:!aNULL:!SHA1:!AESCCM')) }
  ↳ }
# TLS v1.3
haproxy_ssl_cipher_suite_tls13: "{{ ssl_cipher_suite_tls13 | default('TLS_AES_
↳ 128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256') }}"

# haproxy self signed certificate

# Storage location for SSL certificate authority
haproxy_pki_dir: "{{ openstack_pki_dir | default('/etc/pki/haproxy-ca') }}"

# Delegated host for operating the certificate authority
haproxy_pki_setup_host: "{{ openstack_pki_setup_host | default('localhost') }}"
↳ ""

# Create a certificate authority if one does not already exist
haproxy_pki_create_ca: "{{ openstack_pki_authorities is not defined | bool }}"
haproxy_pki_regen_ca: ""
haproxy_pki_authorities:
  - name: "HAProxyRoot"
    country: "GB"
    state_or_province_name: "England"
    organization_name: "Example Corporation"
    organizational_unit_name: "IT Security"
    cn: "HAProxy Root CA"
    provider: selfsigned
    basic_constraints: "CA:TRUE"
    key_usage:
      - digitalSignature
      - cRLSign
      - keyCertSign
    not_after: "+3650d"
  - name: "HAProxyIntermediate"
    country: "GB"
    state_or_province_name: "England"
    organization_name: "Example Corporation"
    organizational_unit_name: "IT Security"

```

(continues on next page)



(continued from previous page)

```

    cn: "HAProxy Intermediate CA"
    provider: ownca
    basic_constraints: "CA:TRUE,pathlen:0"
    key_usage:
      - digitalSignature
      - cRLSign
      - keyCertSign
    not_after: "+3650d"
    signed_by: "HAProxyRoot"

# Installation details for certificate authorities
haproxy_pki_install_ca:
  - name: "HAProxyRoot"
    condition: "{{ haproxy_pki_create_ca }}"

# HAProxy server certificate
haproxy_pki_keys_path: "{{ haproxy_pki_dir ~ '/certs/private/' }}"
haproxy_pki_certs_path: "{{ haproxy_pki_dir ~ '/certs/certs/' }}"
haproxy_pki_intermediate_cert_name: "{{ openstack_pki_service_intermediate_
  ↪cert_name | default('HAProxyIntermediate') }}"
haproxy_pki_intermediate_cert_path: >-
  {{ haproxy_pki_dir ~ '/roots/' ~ haproxy_pki_intermediate_cert_name ~ '/'
  ↪certs/' ~ haproxy_pki_intermediate_cert_name ~ '.crt' }}
haproxy_pki_regen_cert: ""
haproxy_pki_certificates: "{{ _haproxy_pki_certificates }}"

# SSL certificate creation
haproxy_pki_create_certificates: "{{ haproxy_user_ssl_cert is not defined and_
  ↪haproxy_user_ssl_key is not defined }}"

# Installation details for SSL certificates
haproxy_pki_install_certificates: "{{ _haproxy_pki_install_certificates }}"

# activate letsencrypt option
haproxy_ssl_letsencrypt_enable: false
haproxy_ssl_letsencrypt_certbot_binary: "certbot"
haproxy_ssl_letsencrypt_certbot_backend_port: 8888
haproxy_ssl_letsencrypt_pre_hook_timeout: 5
haproxy_ssl_letsencrypt_certbot_bind_address: "{{ ansible_host }}"
haproxy_ssl_letsencrypt_certbot_challenge: "http-01"
haproxy_ssl_letsencrypt_email: "example@example.com"
haproxy_ssl_letsencrypt_config_path: "/etc/letsencrypt/live"
haproxy_ssl_letsencrypt_setup_extra_params: ""
haproxy_ssl_letsencrypt_acl:
  letsencrypt-acl:
    rule: "path_beg /.well-known/acme-challenge/"
    backend_name: letsencrypt

# Use alternative CA that supports ACME, can be a public or private CA
# haproxy_ssl_letsencrypt_certbot_server: "https://acme-staging-v02.api.

```

(continues on next page)

(continued from previous page)

```
↪letsencrypt.org/directory"
haproxy_ssl_letsencrypt_domains:
  - "{{ external_lb_vip_address }}"

# hatop extra package URL and checksum
haproxy_hatop_download_url: "https://github.com/jhunt/hatop/archive/refs/tags/
↪v0.8.2.tar.gz"
haproxy_hatop_download_checksum:
↪"sha256:7fac1f593f92b939cfce34175b593e43862eee8e25db251d03a910b37721fc5d"

# Install hatop
haproxy_hatop_install: true

# The location where the extra packages are downloaded to
haproxy_hatop_download_path: "/opt/cache/files"

## haproxy default
# Set the number of retries to perform on a server after a connection failure
haproxy_retries: "3"
# Set the maximum inactivity time on the client side
haproxy_client_timeout: "50s"
# Set the maximum time to wait for a connection attempt to a server to succeed
haproxy_connect_timeout: "10s"
# Set the maximum allowed time to wait for a complete HTTP request
haproxy_http_request_timeout: "5s"
# Set the maximum inactivity time on the server side
haproxy_server_timeout: "50s"
# Set the HTTP keepalive mode to use
# Disable persistent connections by default because they can cause issues,
↪when the server side closes the connection
# at the same time a request is sent.
haproxy_keepalive_mode: "httpclose"

## haproxy tuning params
haproxy_maxconn: 4096

# Parameters below should only be specified if necessary, defaults are
↪programmed in the template
# haproxy_tuning_params:
#   tune.bufsize: 384000
#   tune.chksize: 16384
#   tune.comp_maxlevel: 1
#   tune.http_maxhdr: 101
#   tune.maxaccept: 64
#   tune.ssl_cachesize: 20000
#   tune.ssl_lifetime: 300
haproxy_tuning_params: {}

# Add extra VIPs to all services
```

(continues on next page)

(continued from previous page)

```
extra_lb_vip_addresses: []

# Add extra TLS VIPs to all services
extra_lb_tls_vip_addresses: []

# Option to override which address haproxy binds to for external vip.
haproxy_bind_external_lb_vip_address: "{{ external_lb_vip_address }}"

# Option to override which address haproxy binds to for internal vip.
haproxy_bind_internal_lb_vip_address: "{{ internal_lb_vip_address }}"

# Option to define if you need haproxy to bind on specific interface.
haproxy_bind_external_lb_vip_interface:
haproxy_bind_internal_lb_vip_interface:

# Option to override haproxy frontend binds
# Example:
# haproxy_vip_binds:
#   - address: '*'
#     interface: bond0
#     type: external
#   - address: '192.168.0.10'
#     pki_san_records:
#       - internal.cloud
haproxy_vip_binds: "{{ haproxy_tls_vip_binds | default(_haproxy_vip_binds) }}"

# Make the log socket available to the chrooted filesystem
haproxy_log_socket: "/dev/log"
haproxy_log_mount_point: "/var/lib/haproxy/dev/log"

# Ansible group name which should be used for distributing self signed SSL
↳Certificates
haproxy_ansible_group_name: haproxy_all

## security.txt
# When security risks in web services are discovered by independent security
# researchers who understand the severity of the risk, they often lack the
# channels to disclose them properly. As a result, security issues may be
# left unreported. security.txt defines a standard to help organizations
# define the process for security researchers to disclose security
# vulnerabilities securely. For more information see https://securitytxt.org/
# This content will be hosted at /security.txt and /.well-known/security.txt
haproxy_security_txt_dir: "/etc/haproxy"
haproxy_security_txt_headers: |
  HTTP/1.0 200 OK
  Cache-Control: no-cache
  Connection: close
  Content-Type: text/plain; charset=utf-8
```

(continues on next page)

(continued from previous page)

```
haproxy_security_txt_content: ""
# haproxy_security_txt_content: |
# # Please see https://securitytxt.org/ for details of the specification of
↪this file

# Allows to copy any static file to the destination hosts
haproxy_static_files_default:
- dest: "{{ haproxy_security_txt_dir }}/security.txt"
  content: "{{ haproxy_security_txt_headers + '\n' + haproxy_security_txt_
↪content }}"
  condition: "{{ haproxy_security_txt_content is truthy }}"
haproxy_static_files_extra: []
haproxy_static_files: "{{ haproxy_static_files_default + haproxy_static_files_
↪extra }}"
```

**REQUIRED VARIABLES**

None.



**DEPENDENCIES**

None.





## EXAMPLE PLAYBOOK

```
---
- name: Install haproxy
  hosts: haproxy
  user: root
  roles:
    - role: haproxy_server
      tags:
        - haproxy-server
  vars:
    haproxy_service_configs:
      - haproxy_service_name: group_name
        haproxy_backend_nodes: "{{ groups['group_name'][0] }}"
        haproxy_backup_nodes: "{{ groups['group_name'][1:] }}"
        haproxy_port: 80
        haproxy_balance_type: http
        haproxy_backend_options:
          - "forwardfor"
          - "httpchk"
          - "httplog"
        haproxy_backend_arguments:
          - "http-check expect string OK"
```