Blazar Documentation

Release 7.0.2.dev2

OpenStack Foundation

Sep 19, 2022

CONTENTS

Blazar is an OpenStack service to provide resource reservations in the OpenStack cloud for different resource types - both virtual (instances, volumes, stacks) and physical (hosts).

CHAPTER

ONE

INSTALLATION GUIDE

1.1 Blazar Installation Guide

1.1.1 Installation using DevStack

This section includes instructions for Blazar installation using DevStack. DevStack configures both the host reservation and the instance reservation.

1. Download DevStack:

```
git clone https://opendev.org/openstack/devstack.git
```

2. Create a local.conf file in the devstack directory. You can use the following sample local.conf:

```
[[local|localrc]]
ADMIN_PASSWORD=password
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
DEST=/opt/stack/
LOGFILE=$DEST/logs/stack.sh.log
HOST_IP=127.0.0.1
GIT_BASE=https://opendev.org/
RECLONE=yes
enable_plugin blazar https://opendev.org/openstack/blazar
```

3. Run DevStack as the stack user:

```
./stack.sh
```

4. Source the admin credentials:

```
. openrc admin admin
```

5. Now you can add hosts to Blazar:

```
blazar host-create hostname
```

1.1.2 Installation without DevStack

This section includes instructions for Blazar installation. You can use the host reservation and the instance reservation once you finish the install guide.

Download all Blazar related repos:

```
git clone https://opendev.org/openstack/blazar
git clone https://opendev.org/openstack/blazar-nova
git clone https://opendev.org/openstack/python-blazarclient
```

Install all these projects to your working environment via:

```
python setup.py install
```

or

```
python setup.py develop
```

Next you need to configure Blazar and Nova. First, generate a blazar.conf sample:

```
cd /path/to/blazar
tox -e genconfig
mv etc/blazar/blazar.conf.sample /etc/blazar/blazar.conf
```

Then edit */etc/blazar/blazar.conf* using the following example:

```
[DEFAULT]
host=<blazar_host>
port=<blazar_port>
os_auth_host=<auth_host>
os_auth_port=<auth_port>
os_auth_protocol=<http, for example>
os_auth_version=v3
os_admin_username=<username>
os_admin_password=<password>
os_admin_project_name=<project_name>
identity_service=<identity_service_name>
os_region_name=<region_name>

[manager]
plugins=physical.host.plugin,virtual.instance.plugin

[keystone_authtoken]
auth_type=<password, for example>
project_domain_name=<project_domain_name>
project_name=<project_name>
user_domain_name=<user_domain_name>
username=<username>
password=<password>
auth_url=<identity_service_url>
```

*os_admin_** flags refer to the Blazar service user. If you do not have this user, create it:

```
openstack user create --password <password> --project <project_name> --
↪email <email-address> <username>
openstack role add --project <project_name> --user <username> <admin_role>
```

Next you need to configure Nova. Please add the following lines to nova.conf file:

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
available_filters = blazarnova.scheduler.filters.blazar_filter.BlazarFilter
enabled_filters = AvailabilityZoneFilter,ComputeFilter,
↪ComputeCapabilitiesFilter,ImagePropertiesFilter,
↪ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,SameHostFilter,
↪DifferentHostFilter,BlazarFilter
```

Restart nova-scheduler to use the new configuration file.

Next you need to create a Nova aggregate to use as a free pool for host reservation:

```
openstack aggregate create freepool
```

And we need to create the reservation service in Keystone with its endpoints:

```
openstack service create --name blazar --description "OpenStack␣
↪Reservation Service" reservation
openstack endpoint create --region <region> blazar admin "<auth_protocol>:/
↪/<blazar_host>:<blazar_port>/v1"
openstack endpoint create --region <region> blazar internal "<auth_
↪protocol>://<blazar_host>:<blazar_port>/v1"
openstack endpoint create --region <region> blazar public "<auth_protocol>
↪://<blazar_host>:<blazar_port>/v1"
```

And, finally, we need to create a database for Blazar:

```
mysql -u<user> -p<password> -h<host> -e "DROP DATABASE IF EXISTS blazar;"
mysql -u<user> -p<password> -h<host> -e "CREATE DATABASE blazar CHARACTER␣
↪SET utf8;"
```

Then edit the database section of */etc/blazar/blazar.conf*:

```
[database]
connection=mysql+pymysql://<user>:<password>@<host>/blazar?charset=utf8
```

To start Blazar services use:

```
blazar-api --config-file /etc/blazar/blazar.conf
blazar-manager --config-file /etc/blazar/blazar.conf
```

Now you can use python-blazarclient to communicate with Blazar.

1.1.3 Install Blazar Dashboard

Please see Blazar Dashboard installation guide (external link).

CHAPTER

TWO

CONFIGURATION REFERENCE

2.1 Blazar Configuration Reference

2.1.1 Configuration

Reference

blazar.conf

DEFAULT

auth_strategy

Type string

Default `keystone`

The strategy to use for auth: noauth or keystone.

port

Type integer

Default `1234`

Minimum Value 0

Maximum Value 65535

Port that will be used to listen on

enable_v1_api

Type boolean

Default `True`

Deploy the v1 API.

host

Type host address

Default `0.0.0.0`

Name of this node. This can be an opaque identifier. It is not necessarily a hostname, FQDN, or IP address. However, the node name must be valid within an AMQP key, and if using ZeroMQ (will be removed in the Stein release), a valid hostname, FQDN, or IP address

log_exchange

Type boolean

Default `False`

Log request/response exchange details: environ, headers and bodies

cleaning_time

Type integer

Default `0`

Minimum Value 0

The minimum interval [minutes] between the end of a lease and the start of the next lease for the same resource. This interval is used for cleanup.

os_auth_protocol

Type string

Default `http`

Protocol used to access OpenStack Identity service

os_auth_host

Type host address

Default `127.0.0.1`

IP or hostname of machine on which OpenStack Identity service is located

os_auth_port

Type string

Default `5000`

Port of OpenStack Identity service.

os_auth_prefix

Type string

Default `''`

Prefix of URL to access OpenStack Identity service.

os_admin_username

Type string

Default `admin`

This OpenStack user is used to treat trusts. The user must have admin role in <os_admin_project_name> project.

os_admin_password

Type string

Default `blazar`

Password of the admin user to treat trusts.

os_admin_project_name

Type string

Default `admin`

Name of project where the user is admin.

os_auth_version

Type string

Default `v3`

Blazar uses API v3 to allow trusts using.

os_admin_user_domain_name

Type string

Default `Default`

A domain name the os_admin_username belongs to.

os_admin_project_domain_name

Type string

Default `Default`

A domain name the os_admin_project_name belongs to

db_driver

Type string

Default `blazar.db`

Driver to use for database access

command

Type unknown type

Default `<None>`

Available commands

identity_service

Type string

Default `identity`

Identity service to use.

os_region_name

Type string

Default `<None>`

Region name of this node. This is used when picking the URL in the service catalog.

Possible values:

- Any string representing region name

keystone_client_version

Type string

Default 3

Keystoneclient version

rpc_conn_pool_size

Type integer

Default 30

Minimum Value 1

Size of RPC connection pool.

Table 1: Deprecated Variations

Group	Name
DEFAULT	rpc_conn_pool_size

conn_pool_min_size

Type integer

Default 2

The pool size limit for connections expiration policy

conn_pool_ttl

Type integer

Default 1200

The time-to-live in sec of idle connections in the pool

executor_thread_pool_size

Type integer

Default 64

Size of executor thread pool when executor is threading or eventlet.

Table 2: Deprecated Variations

Group	Name
DEFAULT	rpc_thread_pool_size

rpc_response_timeout

Type integer

Default 60

Seconds to wait for a response from a call.

transport_url

Type string

Default `rabbit://`

The network address and optional user credentials for connecting to the messaging backend, in URL format. The expected format is:

driver://[user:pass@]host:port[,[userN:passN@]hostN:portN]/virtual_host?query

Example: rabbit://rabbitmq:password@127.0.0.1:5672//

For full details on the fields in the URL see the documentation of oslo_messaging.TransportURL at https://docs.openstack.org/oslo.messaging/latest/reference/transport.html

control_exchange

Type string

Default `openstack`

The default exchange under which topics are scoped. May be overridden by an exchange name specified in the transport_url option.

rpc_ping_enabled

Type boolean

Default `False`

Add an endpoint to answer to ping calls. Endpoint is named oslo_rpc_server_ping

debug

Type boolean

Default `False`

Mutable This option can be changed without restarting.

If set to true, the logging level will be set to DEBUG instead of the default INFO level.

log_config_append

Type string

Default `<None>`

Mutable This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 3: Deprecated Variations

Group	Name
DEFAULT	log-config
DEFAULT	log_config

log_date_format

Type string

Default `%Y-%m-%d %H:%M:%S`

Defines the format string for %(asctime)s in log records. Default: the value above . This option is ignored if log_config_append is set.

log_file

Type string

Default `<None>`

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

Table 4: Deprecated Variations

Group	Name
DEFAULT	logfile

log_dir

Type string

Default `<None>`

(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

Table 5: Deprecated Variations

Group	Name
DEFAULT	logdir

watch_log_file

Type boolean

Default `False`

Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if log_file option is specified and Linux platform is used. This option is ignored if log_config_append is set.

use_syslog

Type boolean

Default `False`

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

use_journal

Type boolean

Default `False`

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages.This option is ignored if log_config_append is set.

syslog_log_facility

Type string

Default `LOG_USER`

Syslog facility to receive log lines. This option is ignored if log_config_append is set.

use_json

Type boolean

Default `False`

Use JSON formatting for logging. This option is ignored if log_config_append is set.

use_stderr

Type boolean

Default `False`

Log output to standard error. This option is ignored if log_config_append is set.

use_eventlog

Type boolean

Default `False`

Log output to Windows Event Log.

log_rotate_interval

Type integer

Default `1`

The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is setto interval.

log_rotate_interval_type

Type string

Default `days`

Valid Values Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

max_logfile_count

Type integer

Default `30`

Maximum number of rotated log files.

max_logfile_size_mb

Type integer

Default `200`

Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

`log_rotation_type`

Type string

Default `none`

Valid Values interval, size, none

Log rotation type.

Possible values

interval Rotate logs at predefined time intervals.

size Rotate logs once they reach a predefined size.

none Do not rotate log files.

`logging_context_format_string`

Type string

Default `%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [%(request_id)s %(user_identity)s] %(instance)s%(message)s`

Format string to use for log messages with context. Used by oslo_log.formatters.ContextFormatter

`logging_default_format_string`

Type string

Default `%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-] %(instance)s%(message)s`

Format string to use for log messages when context is undefined. Used by oslo_log.formatters.ContextFormatter

`logging_debug_format_suffix`

Type string

Default `%(funcName)s %(pathname)s:%(lineno)d`

Additional data to append to log message when logging level for the message is DEBUG. Used by oslo_log.formatters.ContextFormatter

`logging_exception_prefix`

Type string

Default `%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s %(instance)s`

Prefix each line of exception output with this format. Used by oslo_log.formatters.ContextFormatter

`logging_user_identity_format`

Type string

Default `%(user)s %(tenant)s %(domain)s %(user_domain)s %(project_domain)s`

Defines the format string for %(user_identity)s that is used in logging_context_format_string. Used by oslo_log.formatters.ContextFormatter

`default_log_levels`

Type list

Default `['amqp=WARN', 'amqplib=WARN', 'boto=WARN', 'qpid=WARN', 'sqlalchemy=WARN', 'suds=INFO', 'oslo.messaging=INFO', 'oslo_messaging=INFO', 'iso8601=WARN', 'requests.packages.urllib3.connectionpool=WARN', 'urllib3.connectionpool=WARN', 'websocket=WARN', 'requests.packages.urllib3.util.retry=WARN', 'urllib3.util.retry=WARN', 'keystonemiddleware=WARN', 'routes.middleware=WARN', 'stevedore=WARN', 'taskflow=WARN', 'keystoneauth=WARN', 'oslo.cache=INFO', 'oslo_policy=INFO', 'dogpile.core.dogpile=INFO']`

List of package logging levels in logger=LEVEL pairs. This option is ignored if log_config_append is set.

`publish_errors`

Type boolean

Default `False`

Enables or disables publication of error events.

`instance_format`

Type string

Default `"[instance: %(uuid)s] "`

The format for an instance that is passed with the log message.

`instance_uuid_format`

Type string

Default `"[instance: %(uuid)s] "`

The format for an instance UUID that is passed with the log message.

`rate_limit_interval`

Type integer

Default `0`

Interval, number of seconds, of log rate limiting.

`rate_limit_burst`

Type integer

Default `0`

Maximum number of logged messages per rate_limit_interval.

rate_limit_except_level

Type string

Default `CRITICAL`

Log level name used by rate limiting: CRITICAL, ERROR, INFO, WARNING, DEBUG or empty string. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

fatal_deprecations

Type boolean

Default `False`

Enables or disables fatal status of deprecations.

backdoor_port

Type string

Default `<None>`

Enable eventlet backdoor. Acceptable values are 0, <port>, and <start>:<end>, where 0 results in listening on a random tcp port number; <port> results in listening on the specified port number (and not enabling backdoor if that port is in use); and <start>:<end> results in listening on the smallest unused port number within the specified range of port numbers. The chosen port is displayed in the services log file.

backdoor_socket

Type string

Default `<None>`

Enable eventlet backdoor, using the provided path as a unix socket that can receive connections. This option is mutually exclusive with backdoor_port in that only one should be provided. If both are provided then the existence of this option overrides the usage of that option. Inside the path {pid} will be replaced with the PID of the current process.

log_options

Type boolean

Default `True`

Enables or disables logging values of all registered options when starting a service (at DEBUG level).

graceful_shutdown_timeout

Type integer

Default `60`

Specify a timeout after which a gracefully shutdown server will exit. Zero value means endless wait.

api

api_v2_controllers

Type list

Default `['oshosts', 'leases']`

API extensions to use

cors

allowed_origin

Type list

Default `<None>`

Indicate whether this resource may be shared with the domain received in the requests origin header. Format: <protocol>://<host>[:<port>], no trailing slash. Example: https://horizon.example.com

allow_credentials

Type boolean

Default `True`

Indicate that the actual request can include user credentials

expose_headers

Type list

Default `[]`

Indicate which headers are safe to expose to the API. Defaults to HTTP Simple Headers.

max_age

Type integer

Default `3600`

Maximum cache age of CORS preflight requests.

allow_methods

Type list

Default `['OPTIONS', 'GET', 'HEAD', 'POST', 'PUT', 'DELETE', 'TRACE', 'PATCH']`

Indicate which methods can be used during the actual request.

allow_headers

Type list

Default `[]`

Indicate which header field names may be used during the actual request.

database

sqlite_synchronous

Type boolean

Default `True`

If True, SQLite uses synchronous mode.

Table 6: Deprecated Variations

Group	Name
DEFAULT	sqlite_synchronous

backend

Type string

Default `sqlalchemy`

The back end to use for the database.

Table 7: Deprecated Variations

Group	Name
DEFAULT	db_backend

connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the database.

Table 8: Deprecated Variations

Group	Name
DEFAULT	sql_connection
DATABASE	sql_connection
sql	connection

slave_connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the slave database.

mysql_sql_mode

Type string

Default `TRADITIONAL`

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: mysql_sql_mode=

mysql_enable_ndb

Type boolean

Default `False`

If True, transparently enables support for handling MySQL Cluster (NDB).

connection_recycle_time

Type integer

Default `3600`

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

Table 9: Deprecated Variations

Group	Name
DATABASE	idle_timeout
database	idle_timeout
DEFAULT	sql_idle_timeout
DATABASE	sql_idle_timeout
sql	idle_timeout

max_pool_size

Type integer

Default `5`

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

Table 10: Deprecated Variations

Group	Name
DEFAULT	sql_max_pool_size
DATABASE	sql_max_pool_size

max_retries

Type integer

Default `10`

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

Table 11: Deprecated Variations

Group	Name
DEFAULT	sql_max_retries
DATABASE	sql_max_retries

retry_interval

Type integer

Default `10`

Interval between retries of opening a SQL connection.

Table 12: Deprecated Variations

Group	Name
DEFAULT	sql_retry_interval
DATABASE	reconnect_interval

`max_overflow`

Type integer

Default `50`

If set, use this value for max_overflow with SQLAlchemy.

Table 13: Deprecated Variations

Group	Name
DEFAULT	sql_max_overflow
DATABASE	sqlalchemy_max_overflow

`connection_debug`

Type integer

Default `0`

Minimum Value 0

Maximum Value 100

Verbosity of SQL debugging information: 0=None, 100=Everything.

Table 14: Deprecated Variations

Group	Name
DEFAULT	sql_connection_debug

`connection_trace`

Type boolean

Default `False`

Add Python stack traces to SQL as comment strings.

Table 15: Deprecated Variations

Group	Name
DEFAULT	sql_connection_trace

`pool_timeout`

Type integer

Default `<None>`

If set, use this value for pool_timeout with SQLAlchemy.

Table 16: Deprecated Variations

Group	Name
DATABASE	sqlalchemy_pool_timeout

use_db_reconnect

Type boolean

Default `False`

Enable the experimental use of database reconnect on connection lost.

db_retry_interval

Type integer

Default `1`

Seconds between retries of a database transaction.

db_inc_retry_interval

Type boolean

Default `True`

If True, increases the interval between retries of a database operation up to db_max_retry_interval.

db_max_retry_interval

Type integer

Default `10`

If db_inc_retry_interval is set, the maximum seconds between retries of a database operation.

db_max_retries

Type integer

Default `20`

Maximum retries in case of connection error or deadlock error before error is raised. Set to -1 to specify an infinite retry count.

connection_parameters

Type string

Default `''`

Optional URL parameters to append onto the connection URL at connect time; specify as param1=value1&param2=value2&

enforcement

max_lease_duration

Type integer

Default `-1`

Maximum lease duration in seconds. If this is set to -1, there is not limit.

max_lease_duration_exempt_project_ids

Type list

Default `[]`

Allow list of project ids exempt from filter constraints.

enabled_filters

Type list

Default `[]`

List of enabled usage enforcement filters.

healthcheck

path

Type string

Default `/healthcheck`

The path to respond to healtcheck requests on.

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.

detailed

Type boolean

Default `False`

Show more detailed information as part of the response. Security note: Enabling this option may expose sensitive details about the service being monitored. Be sure to verify that it will not violate your security policies.

backends

Type list

Default `[]`

Additional backends that can perform health checks and report that information back as part of a request.

disable_by_file_path

Type string

Default `<None>`

Check the presence of a file to determine if an application is running on a port. Used by DisableByFileHealthcheck plugin.

`disable_by_file_paths`

Type list

Default `[]`

Check the presence of a file based on a port to determine if an application is running on a port. Expects a port:path list of strings. Used by DisableByFilesPortsHealthcheck plugin.

keystone_authtoken

`www_authenticate_uri`

Type string

Default `<None>`

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint.

Table 17: Deprecated Variations

Group	Name
keystone_authtoken	auth_uri

`auth_uri`

Type string

Default `<None>`

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint. This option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

Warning: This option is deprecated for removal since Queens. Its value may be silently ignored in the future.

Reason The auth_uri option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

auth_version

Type string

Default <None>

API version of the Identity API endpoint.

interface

Type string

Default internal

Interface to use for the Identity API endpoint. Valid values are public, internal (default) or admin.

delay_auth_decision

Type boolean

Default False

Do not handle authorization requests within the middleware, but delegate the authorization decision to downstream WSGI components.

http_connect_timeout

Type integer

Default <None>

Request timeout value for communicating with Identity API server.

http_request_max_retries

Type integer

Default 3

How many times are we trying to reconnect when communicating with Identity API Server.

cache

Type string

Default <None>

Request environment key where the Swift cache object is stored. When auth_token middleware is deployed with a Swift cache, use this option to have the middleware share a caching backend with swift. Otherwise, use the memcached_servers option instead.

certfile

Type string

Default <None>

Required if identity server requires client certificate

keyfile

Type string

Default <None>

Required if identity server requires client certificate

cafile

Type string

Default <None>

A PEM encoded Certificate Authority to use when verifying HTTPs connections. Defaults to system CAs.

insecure

Type boolean

Default False

Verify HTTPS connections.

region_name

Type string

Default <None>

The region in which the identity server can be found.

memcached_servers

Type list

Default <None>

Optionally specify a list of memcached server(s) to use for caching. If left undefined, tokens will instead be cached in-process.

Table 18: Deprecated Variations

Group	Name
keystone_authtoken	memcache_servers

token_cache_time

Type integer

Default 300

In order to prevent excessive effort spent validating tokens, the middleware caches previously-seen tokens for a configurable duration (in seconds). Set to -1 to disable caching completely.

memcache_security_strategy

Type string

Default None

Valid Values None, MAC, ENCRYPT

(Optional) If defined, indicate whether token data should be authenticated or authenticated and encrypted. If MAC, token data is authenticated (with HMAC) in the cache. If ENCRYPT, token data is encrypted and authenticated in the cache. If the value is not one of these options or empty, auth_token will raise an exception on initialization.

memcache_secret_key

Type string

Default `<None>`

(Optional, mandatory if memcache_security_strategy is defined) This string is used for key derivation.

memcache_pool_dead_retry

Type integer

Default `300`

(Optional) Number of seconds memcached server is considered dead before it is tried again.

memcache_pool_maxsize

Type integer

Default `10`

(Optional) Maximum total number of open connections to every memcached server.

memcache_pool_socket_timeout

Type integer

Default `3`

(Optional) Socket timeout in seconds for communicating with a memcached server.

memcache_pool_unused_timeout

Type integer

Default `60`

(Optional) Number of seconds a connection to memcached is held unused in the pool before it is closed.

memcache_pool_conn_get_timeout

Type integer

Default `10`

(Optional) Number of seconds that an operation will wait to get a memcached client connection from the pool.

memcache_use_advanced_pool

Type boolean

Default `False`

(Optional) Use the advanced (eventlet safe) memcached client pool. The advanced pool will only work under python 2.x.

include_service_catalog

Type boolean

Default `True`

(Optional) Indicate whether to set the X-Service-Catalog header. If False, middleware will not ask for service catalog on token validation and will not set the X-Service-Catalog header.

enforce_token_bind

Type string

Default `permissive`

Used to control the use and type of token binding. Can be set to: disabled to not check token binding. permissive (default) to validate binding information if the bind type is of a form known to the server and ignore it if not. strict like permissive but if the bind type is unknown the token will be rejected. required any form of token binding is needed to be allowed. Finally the name of a binding method that must be present in tokens.

service_token_roles

Type list

Default `['service']`

A choice of roles that must be present in a service token. Service tokens are allowed to request that an expired token can be used and so this check should tightly control that only actual services should be sending this token. Roles here are applied as an ANY check so any role in this list must be present. For backwards compatibility reasons this currently only affects the allow_expired check.

service_token_roles_required

Type boolean

Default `False`

For backwards compatibility reasons we must let valid service tokens pass that dont pass the service_token_roles check as valid. Setting this true will become the default in a future release and should be enabled if possible.

service_type

Type string

Default `<None>`

The name or type of the service as it appears in the service catalog. This is used to validate tokens that have restricted access rules.

auth_type

Type unknown type

Default `<None>`

Authentication type to load

Table 19: Deprecated Variations

Group	Name
keystone_authtoken	auth_plugin

auth_section

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

manager

rpc_topic

Type string

Default `blazar.manager`

The topic Blazar uses for blazar-manager messages.

plugins

Type list

Default `['dummy.vm.plugin']`

All plugins to use (one for every resource type to support.)

minutes_before_end_lease

Type integer

Default `60`

Minimum Value 0

Minutes prior to the end of a lease in which actions like notification and snapshot are taken. If this is set to 0, then these actions are not taken.

event_max_retries

Type integer

Default `1`

Minimum Value 0

Maximum Value 50

Number of times to retry an event action.

notifications

publisher_id

Type string

Default `blazar.lease`

Publisher ID for notifications

nova

nova_client_version

Type string

Default `2`

Novaclient version

Table 20: Deprecated Variations

Group	Name
DEFAULT	nova_client_version

compute_service

Type string

Default `compute`

Nova name in keystone

Table 21: Deprecated Variations

Group	Name
DEFAULT	compute_service

image_prefix

Type string

Default `reserved_`

Prefix for VM images if you want to create snapshots

Table 22: Deprecated Variations

Group	Name
DEFAULT	image_prefix

aggregate_freepool_name

Type string

Default `freepool`

Name of the special aggregate where all hosts are candidate for physical host reservation

Table 23: Deprecated Variations

Group	Name
physical:host	aggregate_freepool_name

project_id_key

Type string

Default `blazar:project`

Aggregate metadata value for key matching project_id

Table 24: Deprecated Variations

Group	Name
physical:host	project_id_key

blazar_owner

Type string

Default `blazar:owner`

Aggregate metadata key for knowing owner project_id

Table 25: Deprecated Variations

Group	Name
physical:host	blazar_owner

az_aware

Type boolean

Default `True`

A flag to store original availability zone

oslo_concurrency

disable_process_locking

Type boolean

Default `False`

Enables or disables inter-process locks.

Table 26: Deprecated Variations

Group	Name
DEFAULT	disable_process_locking

lock_path

Type string

Default `<None>`

Directory to use for lock files. For security, the specified directory should only be writable by the user running the processes that need locking. Defaults to environment variable OSLO_LOCK_PATH. If external locks are used, a lock path must be set.

Table 27: Deprecated Variations

Group	Name
DEFAULT	lock_path

oslo_messaging_amqp

container_name

Type string

Default <None>

Name for the AMQP container. must be globally unique. Defaults to a generated UUID

Table 28: Deprecated Variations

Group	Name
amqp1	container_name

idle_timeout

Type integer

Default 0

Timeout for inactive connections (in seconds)

Table 29: Deprecated Variations

Group	Name
amqp1	idle_timeout

trace

Type boolean

Default False

Debug: dump AMQP frames to stdout

Table 30: Deprecated Variations

Group	Name
amqp1	trace

ssl

Type boolean

Default False

Attempt to connect via SSL. If no other ssl-related parameters are given, it will use the systems CA-bundle to verify the servers certificate.

ssl_ca_file

Type string

Default ''

CA certificate PEM file used to verify the servers certificate

Table 31: Deprecated Variations

Group	Name
amqp1	ssl_ca_file

ssl_cert_file

Type string

Default `''`

Self-identifying certificate PEM file for client authentication

Table 32: Deprecated Variations

Group	Name
amqp1	ssl_cert_file

ssl_key_file

Type string

Default `''`

Private key PEM file used to sign ssl_cert_file certificate (optional)

Table 33: Deprecated Variations

Group	Name
amqp1	ssl_key_file

ssl_key_password

Type string

Default `<None>`

Password for decrypting ssl_key_file (if encrypted)

Table 34: Deprecated Variations

Group	Name
amqp1	ssl_key_password

ssl_verify_vhost

Type boolean

Default `False`

By default SSL checks that the name in the servers certificate matches the hostname in the transport_url. In some configurations it may be preferable to use the virtual hostname instead, for example if the server uses the Server Name Indication TLS extension (rfc6066) to provide a certificate per virtual host. Set ssl_verify_vhost to True if the servers SSL certificate uses the virtual host name instead of the DNS name.

sasl_mechanisms

Type string

Default `''`

Space separated list of acceptable SASL mechanisms

Table 35: Deprecated Variations

Group	Name
amqp1	sasl_mechanisms

`sasl_config_dir`

Type string

Default `''`

Path to directory that contains the SASL configuration

Table 36: Deprecated Variations

Group	Name
amqp1	sasl_config_dir

`sasl_config_name`

Type string

Default `''`

Name of configuration file (without .conf suffix)

Table 37: Deprecated Variations

Group	Name
amqp1	sasl_config_name

`sasl_default_realm`

Type string

Default `''`

SASL realm to use if no realm present in username

`connection_retry_interval`

Type integer

Default 1

Minimum Value 1

Seconds to pause before attempting to re-connect.

`connection_retry_backoff`

Type integer

Default 2

Minimum Value 0

Increase the connection_retry_interval by this many seconds after each unsuccessful failover attempt.

connection_retry_interval_max

Type integer

Default 30

Minimum Value 1

Maximum limit for connection_retry_interval + connection_retry_backoff

link_retry_delay

Type integer

Default 10

Minimum Value 1

Time to pause between re-connecting an AMQP 1.0 link that failed due to a recoverable error.

default_reply_retry

Type integer

Default 0

Minimum Value -1

The maximum number of attempts to re-send a reply message which failed due to a recoverable error.

default_reply_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc reply message delivery.

default_send_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc cast or call message delivery. Only used when caller does not provide a timeout expiry.

default_notify_timeout

Type integer

Default 30

Minimum Value 5

The deadline for a sent notification message delivery. Only used when caller does not provide a timeout expiry.

default_sender_link_timeout

Type integer

Default `600`

Minimum Value 1

The duration to schedule a purge of idle sender links. Detach link after expiry.

`addressing_mode`

Type string

Default `dynamic`

Indicates the addressing mode used by the driver. Permitted values: legacy - use legacy non-routable addressing routable - use routable addresses dynamic - use legacy addresses if the message bus does not support routing otherwise use routable addressing

`pseudo_vhost`

Type boolean

Default `True`

Enable virtual host support for those message buses that do not natively support virtual hosting (such as qpidd). When set to true the virtual host name will be added to all message bus addresses, effectively creating a private subnet per virtual host. Set to False if the message bus supports virtual hosting using the hostname field in the AMQP 1.0 Open performative as the name of the virtual host.

`server_request_prefix`

Type string

Default `exclusive`

address prefix used when sending to a specific server

Table 38: Deprecated Variations

Group	Name
amqp1	server_request_prefix

`broadcast_prefix`

Type string

Default `broadcast`

address prefix used when broadcasting to all servers

Table 39: Deprecated Variations

Group	Name
amqp1	broadcast_prefix

`group_request_prefix`

Type string

Default `unicast`

address prefix when sending to any server in group

Table 40: Deprecated Variations

Group	Name
amqp1	group_request_prefix

rpc_address_prefix

Type string

Default `openstack.org/om/rpc`

Address prefix for all generated RPC addresses

notify_address_prefix

Type string

Default `openstack.org/om/notify`

Address prefix for all generated Notification addresses

multicast_address

Type string

Default `multicast`

Appended to the address prefix when sending a fanout message. Used by the message bus to identify fanout messages.

unicast_address

Type string

Default `unicast`

Appended to the address prefix when sending to a particular RPC/Notification server. Used by the message bus to identify messages sent to a single destination.

anycast_address

Type string

Default `anycast`

Appended to the address prefix when sending to a group of consumers. Used by the message bus to identify messages that should be delivered in a round-robin fashion across consumers.

default_notification_exchange

Type string

Default `<None>`

Exchange name used in notification addresses. Exchange name resolution precedence: Target.exchange if set else default_notification_exchange if set else control_exchange if set else notify

default_rpc_exchange

Type string

Default `<None>`

Exchange name used in RPC addresses. Exchange name resolution precedence: Target.exchange if set else default_rpc_exchange if set else control_exchange if set else rpc

reply_link_credit

Type integer

Default `200`

Minimum Value 1

Window size for incoming RPC Reply messages.

rpc_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming RPC Request messages

notify_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming Notification messages

pre_settled

Type multi-valued

Default `rpc-cast`

Default `rpc-reply`

Send messages of this type pre-settled. Pre-settled messages will not receive acknowledgement from the peer. Note well: pre-settled messages may be silently discarded if the delivery fails. Permitted values: rpc-call - send RPC Calls pre-settled rpc-reply- send RPC Replies pre-settled rpc-cast - Send RPC Casts pre-settled notify - Send Notifications pre-settled

oslo_messaging_kafka

kafka_max_fetch_bytes

Type integer

Default `1048576`

Max fetch bytes of Kafka consumer

kafka_consumer_timeout

Type floating point

Default `1.0`

Default timeout(s) for Kafka consumers

pool_size

Type integer

Default `10`

Pool Size for Kafka Consumers

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_min_size

Type integer

Default `2`

The pool size limit for connections expiration policy

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_ttl

Type integer

Default `1200`

The time-to-live in sec of idle connections in the pool

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

consumer_group

Type string

Default `oslo_messaging_consumer`

Group id for Kafka consumer. Consumers in one group will coordinate message consumption

producer_batch_timeout

Type floating point

Default `0.0`

Upper bound on the delay for KafkaProducer batching in seconds

producer_batch_size

Type integer

Default `16384`

Size of batch for the producer async send

`compression_codec`

Type string

Default `none`

Valid Values none, gzip, snappy, lz4, zstd

The compression codec for all data generated by the producer. If not set, compression will not be used. Note that the allowed values of this depend on the kafka version

`enable_auto_commit`

Type boolean

Default `False`

Enable asynchronous consumer commits

`max_poll_records`

Type integer

Default `500`

The maximum number of records returned in a poll call

`security_protocol`

Type string

Default `PLAINTEXT`

Valid Values PLAINTEXT, SASL_PLAINTEXT, SSL, SASL_SSL

Protocol used to communicate with brokers

`sasl_mechanism`

Type string

Default `PLAIN`

Mechanism when security protocol is SASL

`ssl_cafile`

Type string

Default `''`

CA certificate PEM file used to verify the server certificate

`ssl_client_cert_file`

Type string

Default `''`

Client certificate PEM file used for authentication.

`ssl_client_key_file`

Type string

Default `''`

Client key PEM file used for authentication.

`ssl_client_key_password`

Type string

Default `''`

Client key password file used for authentication.

oslo_messaging_notifications

`driver`

Type multi-valued

Default `''`

The Drivers(s) to handle sending notifications. Possible values are messaging, messagingv2, routing, log, test, noop

Table 41: Deprecated Variations

Group	Name
DEFAULT	notification_driver

`transport_url`

Type string

Default `<None>`

A URL representing the messaging driver to use for notifications. If not set, we fall back to the same configuration used for RPC.

Table 42: Deprecated Variations

Group	Name
DEFAULT	notification_transport_url

`topics`

Type list

Default `['notifications']`

AMQP topic used for OpenStack notifications.

Table 43: Deprecated Variations

Group	Name
rpc_notifier2	topics
DEFAULT	notification_topics

`retry`

Type integer

Default `-1`

The maximum number of attempts to re-send a notification message which failed to be delivered due to a recoverable error. 0 - No retry, -1 - indefinite

oslo_messaging_rabbit

amqp_durable_queues

Type boolean

Default `False`

Use durable queues in AMQP.

amqp_auto_delete

Type boolean

Default `False`

Auto-delete queues in AMQP.

Table 44: Deprecated Variations

Group	Name
DEFAULT	amqp_auto_delete

ssl

Type boolean

Default `False`

Connect over SSL.

Table 45: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_use_ssl

ssl_version

Type string

Default `''`

SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1_1, and TLSv1_2 may be available on some distributions.

Table 46: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_version

ssl_key_file

Type string

Default `''`

SSL key file (valid only if SSL enabled).

Table 47: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_keyfile

`ssl_cert_file`

Type string

Default `''`

SSL cert file (valid only if SSL enabled).

Table 48: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_certfile

`ssl_ca_file`

Type string

Default `''`

SSL certification authority file (valid only if SSL enabled).

Table 49: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_ca_certs

`heartbeat_in_pthread`

Type boolean

Default `True`

Run the health check heartbeat thread through a native python thread by default. If this option is equal to False then the health check heartbeat will inherit the execution model from the parent process. For example if the parent process has monkey patched the stdlib by using eventlet/greenlet then the heartbeat will be run through a green thread.

Warning: This option is deprecated for removal. Its value may be silently ignored in the future.

`kombu_reconnect_delay`

Type floating point

Default `1.0`

How long to wait before reconnecting in response to an AMQP consumer cancel notification.

Table 50: Deprecated Variations

Group	Name
DEFAULT	kombu_reconnect_delay

kombu_compression

Type string

Default <None>

EXPERIMENTAL: Possible values are: gzip, bz2. If not set compression will not be used. This option may not be available in future versions.

kombu_missing_consumer_retry_timeout

Type integer

Default 60

How long to wait a missing client before abandoning to send it its replies. This value should not be longer than rpc_response_timeout.

Table 51: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_reconnect_timeout

kombu_failover_strategy

Type string

Default round-robin

Valid Values round-robin, shuffle

Determines how the next RabbitMQ node is chosen in case the one we are currently connected to becomes unavailable. Takes effect only if more than one RabbitMQ node is provided in config.

rabbit_login_method

Type string

Default AMQPLAIN

Valid Values PLAIN, AMQPLAIN, RABBIT-CR-DEMO

The RabbitMQ login method.

Table 52: Deprecated Variations

Group	Name
DEFAULT	rabbit_login_method

rabbit_retry_interval

Type integer

Default 1

How frequently to retry connecting with RabbitMQ.

rabbit_retry_backoff

Type integer

Default `2`

How long to backoff for between retries when connecting to RabbitMQ.

Table 53: Deprecated Variations

Group	Name
DEFAULT	rabbit_retry_backoff

rabbit_interval_max

Type integer

Default `30`

Maximum interval of RabbitMQ connection retries. Default is 30 seconds.

rabbit_ha_queues

Type boolean

Default `False`

Try to use HA queues in RabbitMQ (x-ha-policy: all). If you change this option, you must wipe the RabbitMQ database. In RabbitMQ 3.0, queue mirroring is no longer controlled by the x-ha-policy argument when declaring a queue. If you just want to make sure that all queues (except those with auto-generated names) are mirrored across all nodes, run: rabbitmqctl set_policy HA ^(?!amq.).* {ha-mode: all}

Table 54: Deprecated Variations

Group	Name
DEFAULT	rabbit_ha_queues

rabbit_transient_queues_ttl

Type integer

Default `1800`

Minimum Value 1

Positive integer representing duration in seconds for queue TTL (x-expires). Queues which are unused for the duration of the TTL are automatically deleted. The parameter affects only reply and fanout queues.

rabbit_qos_prefetch_count

Type integer

Default `0`

Specifies the number of messages to prefetch. Setting to zero allows unlimited messages.

heartbeat_timeout_threshold

Type integer

Default `60`

Number of seconds after which the Rabbit broker is considered down if heartbeats keep-alive fails (0 disables heartbeat).

`heartbeat_rate`

Type integer

Default `2`

How often times during the heartbeat_timeout_threshold we check the heartbeat.

`direct_mandatory_flag`

Type boolean

Default `True`

(DEPRECATED) Enable/Disable the RabbitMQ mandatory flag for direct send. The direct send is used as reply, so the MessageUndeliverable exception is raised in case the client queue does not exist.MessageUndeliverable exception will be used to loop for a timeout to lets a chance to sender to recover.This flag is deprecated and it will not be possible to deactivate this functionality anymore

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Mandatory flag no longer deactivable.

`enable_cancel_on_failover`

Type boolean

Default `False`

Enable x-cancel-on-ha-failover flag so that rabbitmq server will cancel and notify consumerswhen queue is down

oslo_middleware

`max_request_body_size`

Type integer

Default `114688`

The maximum body size for each request, in bytes.

Table 55: Deprecated Variations

Group	Name
DEFAULT	osapi_max_request_body_size
DEFAULT	max_request_body_size

`secure_proxy_ssl_header`

Type string

Default `X-Forwarded-Proto`

The HTTP Header that will be used to determine what the original request protocol scheme was, even if it was hidden by a SSL termination proxy.

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.

`enable_proxy_headers_parsing`

Type boolean

Default `False`

Whether the application is behind a proxy or not. This determines if the middleware should parse the headers or not.

oslo_policy

`enforce_scope`

Type boolean

Default `False`

This option controls whether or not to enforce scope when evaluating policies. If `True`, the scope of the token used in the request is compared to the `scope_types` of the policy being enforced. If the scopes do not match, an `InvalidScope` exception will be raised. If `False`, a message will be logged informing operators that policies are being invoked with mismatching scope.

`enforce_new_defaults`

Type boolean

Default `False`

This option controls whether or not to use old deprecated defaults when evaluating policies. If `True`, the old deprecated defaults are not going to be evaluated. This means if any existing token is allowed for old defaults but is disallowed for new defaults, it will be disallowed. It is encouraged to enable this flag along with the `enforce_scope` flag so that you can get the benefits of new defaults and `scope_type` together

`policy_file`

Type string

Default `policy.yaml`

The relative or absolute path of a file that maps roles to permissions for a given service. Relative paths must be specified in relation to the configuration file setting this option.

Table 56: Deprecated Variations

Group	Name
DEFAULT	policy_file

`policy_default_rule`

Type string

Default `default`

Default rule. Enforced when a requested rule is not found.

Table 57: Deprecated Variations

Group	Name
DEFAULT	policy_default_rule

policy_dirs

Type multi-valued

Default `policy.d`

Directories where policy configuration files are stored. They can be relative to any directory in the search path defined by the config_dir option, or absolute paths. The file defined by policy_file must exist for these directories to be searched. Missing or empty directories are ignored.

Table 58: Deprecated Variations

Group	Name
DEFAULT	policy_dirs

remote_content_type

Type string

Default `application/x-www-form-urlencoded`

Valid Values application/x-www-form-urlencoded, application/json

Content Type to send and receive data for REST based policy check

remote_ssl_verify_server_crt

Type boolean

Default `False`

server identity verification for REST based policy check

remote_ssl_ca_crt_file

Type string

Default `<None>`

Absolute path to ca cert file for REST based policy check

remote_ssl_client_crt_file

Type string

Default `<None>`

Absolute path to client cert for REST based policy check

remote_ssl_client_key_file

Type string

Default `<None>`

Absolute path client key file REST based policy check

physical:host

`blazar_az_prefix`

Type string

Default `blazar_`

Prefix for Availability Zones created by Blazar

`before_end`

Type string

Default `''`

Actions which we will be taken before the end of the lease

`enable_notification_monitor`

Type boolean

Default `False`

Enable notification-based resource monitoring. If it is enabled, the blazar-manager monitors states of compute hosts by subscribing to notifications of Nova.

`notification_topics`

Type list

Default `['notifications', 'versioned_notifications']`

Notification topics to subscribe to.

`enable_polling_monitor`

Type boolean

Default `False`

Enable polling-based resource monitoring. If it is enabled, the blazar-manager monitors states of compute hosts by polling the Nova API.

`polling_interval`

Type integer

Default `60`

Minimum Value 1

Interval (seconds) of polling for health checking.

`healing_interval`

Type integer

Default `60`

Minimum Value 0

Interval (minutes) of reservation healing. If 0 is specified, the interval is infinite and all the reservations in the future is healed at one time.

nova.conf

Please add the following lines to the nova.conf configuration file:

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
available_filters = blazarnova.scheduler.filters.blazar_filter.BlazarFilter
enabled_filters = AvailabilityZoneFilter,ComputeFilter,
↪ComputeCapabilitiesFilter,ImagePropertiesFilter,
↪ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,SameHostFilter,
↪DifferentHostFilter,BlazarFilter
```

2.1.2 Policy

Reference

Policies

Warning: Using a JSON-formatted policy file is deprecated since Blazar 7.0.0 (Wallaby). This oslopolicy-convert-json-to-yaml tool will migrate your existing JSON-formatted policy file to YAML in a backward-compatible way.

The following is an overview of all available policies in Blazar. For a sample configuration file, refer to *Sample Policy File*.

To change policies, please create a policy file in */etc/blazar/* and specify the policy file name at the *oslo_policy/policy_file* option in *blazar.conf*.

blazar

`admin`

Default `is_admin:True or role:admin`

Default rule for most Admin APIs.

`admin_or_owner`

Default `rule:admin or project_id:%(project_id)s`

Default rule for most non-Admin APIs.

`blazar:leases:get`

Default `rule:admin_or_owner`

Operations

- **GET** `/{api_version}/leases`

- **GET** /{api_version}/leases/{lease_id}

Policy rule for List/Show Lease(s) API.

blazar:leases:post

Default rule:admin_or_owner

Operations

- **POST** /{api_version}/leases

Policy rule for Create Lease API.

blazar:leases:put

Default rule:admin_or_owner

Operations

- **PUT** /{api_version}/leases/{lease_id}

Policy rule for Update Lease API.

blazar:leases:delete

Default rule:admin_or_owner

Operations

- **DELETE** /{api_version}/leases/{lease_id}

Policy rule for Delete Lease API.

blazar:oshosts:get

Default rule:admin

Operations

- **GET** /{api_version}/os-hosts
- **GET** /{api_version}/os-hosts/{host_id}

Policy rule for List/Show Host(s) API.

blazar:oshosts:post

Default rule:admin

Operations

- **POST** /{api_version}/os-hosts

Policy rule for Create Host API.

blazar:oshosts:put

Default rule:admin

Operations

- **PUT** /{api_version}/os-hosts/{host_id}

Policy rule for Update Host API.

blazar:oshosts:delete

Default `rule:admin`

Operations

- **DELETE** `/{api_version}/os-hosts/{host_id}`

Policy rule for Delete Host API.

`blazar:oshosts:get_allocations`

Default `rule:admin`

Operations

- **GET** `/{api_version}/os-hosts/allocations`
- **GET** `/{api_version}/os-hosts/{host_id}/allocation`

Policy rule for List/Get Host(s) Allocations API.

`blazar:floatingips:get`

Default `rule:admin`

Operations

- **GET** `/{api_version}/floatingips`
- **GET** `/{api_version}/floatingips/{floatingip_id}`

Policy rule for List/Show FloatingIP(s) API.

`blazar:floatingips:post`

Default `rule:admin`

Operations

- **POST** `/{api_version}/floatingips`

Policy rule for Create Floating IP API.

`blazar:floatingips:delete`

Default `rule:admin`

Operations

- **DELETE** `/{api_version}/floatingips/{floatingip_id}`

Policy rule for Delete Floating IP API.

CHAPTER

THREE

CLI REFERENCE

3.1 Command-Line Interface Reference

3.1.1 Host Reservation

Prerequisites

The following packages should be installed:

- blazar
- blazar-nova
- python-blazarclient

1. Add hosts into the freepool

1. Add hosts into the Blazar freepool using the host-create command:

```
blazar host-create compute-1
```

2. Check hosts in the freepool:

```
blazar host-list
```

Result:

```
+----+---------------------+-------+-----------+----------+
| id | hypervisor_hostname | vcpus | memory_mb | local_gb |
+----+---------------------+-------+-----------+----------+
| 1  | compute-1           |     2 |      3951 |       38 |
+----+---------------------+-------+-----------+----------+
```

2. Create a lease

1. Create a lease (compute host reservation) using lease-create command:

```
blazar lease-create --physical-reservation min=1,max=1,hypervisor_
↪properties='[">=", "$vcpus", "2"]' --start-date "2020-06-08 12:00" --end-
↪date "2020-06-09 12:00" lease-1
```

Result:

```
+---------------+-------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪----------+
| Field         | Value                                                       ␣
↪                                                                              ␣
↪          |
+---------------+-------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪----------+
| action        |                                                             ␣
↪                                                                              ␣
↪          |
| created_at    | 2020-06-08 02:43:40                                         ␣
↪                                                                              ␣
↪          |
| end_date      | 2020-06-09T12:00:00.000000                                  ␣
↪                                                                              ␣
↪          |
| events        | {"status": "UNDONE", "lease_id": "6638c31e-f6c8-4982-
↪9b98-d2ca0a8cb646", "event_type": "before_end_lease", "created_at":
↪"2020-06-08         |
|               | 02:43:40", "updated_at": null, "time": "2020-06-
↪08T12:00:00.000000", "id": "420caf25-dba5-4ac3-b377-50503ea5c886"}          ␣
↪                   |
|               | {"status": "UNDONE", "lease_id": "6638c31e-f6c8-4982-
↪9b98-d2ca0a8cb646", "event_type": "start_lease", "created_at": "2020-06-
↪08 02:43:40",   |
|               | "updated_at": null, "time": "2020-06-08T12:00:00.000000",
↪ "id": "b9696139-55a1-472d-baff-5fade2c15243"}                              ␣
↪          |
|               | {"status": "UNDONE", "lease_id": "6638c31e-f6c8-4982-
↪9b98-d2ca0a8cb646", "event_type": "end_lease", "created_at": "2020-06-08␣
↪02:43:40",     |
|               | "updated_at": null, "time": "2020-06-09T12:00:00.000000",
↪ "id": "ff9e6f52-db50-475a-81f1-e6897fdc769d"}                              ␣
↪          |
| id            | 6638c31e-f6c8-4982-9b98-d2ca0a8cb646                        ␣
↪                                                                              ␣
↪          |
| name          | lease-1                                                     ␣
↪                                                                              ␣
↪          |
| project_id    | 4527fa2138564bd4933887526d01bc95                            ␣
↪                                                                              ␣
↪          |
| reservations  | {"status": "pending", "lease_id": "6638c31e-f6c8-4982-
↪9b98-d2ca0a8cb646", "resource_id": "8", "max": 1, "created_at": "2020-06-
↪08           |
```

(continues on next page)

(continued from previous page)

```
|               | 02:43:40", "min": 1, "updated_at": null, "hypervisor_
↪properties": "[\">=\", \"$vcpus\", \"2\"]", "resource_properties": "",
↪"id":           |
|               | "4d3dd68f-0e3f-4f6b-bef7-617525c74ccb", "resource_type":
↪"physical:host"}                                                              ␣
↪          |
| start_date    | 2020-06-08T12:00:00.000000                                   ␣
↪                                                                              ␣
↪          |
| status        |                                                              ␣
↪                                                                              ␣
↪          |
| status_reason |                                                              ␣
↪                                                                              ␣
↪          |
| trust_id      | ba4c321878d84d839488216de0a9e945                             ␣
↪                                                                              ␣
↪          |
| updated_at    |                                                              ␣
↪                                                                              ␣
↪          |
| user_id       |                                                              ␣
↪                                                                              ␣
↪          |
+---------------+------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-----------+
```

2. Check leases:

```
blazar lease-list
```

Result:

```
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
| id                                   | name    | start_date              ␣
↪   | end_date                    |
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
| 6638c31e-f6c8-4982-9b98-d2ca0a8cb646 | lease-1 | 2020-06-08T12:00:00.
↪000000 | 2020-06-09T12:00:00.000000 |
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
```

3. Use the leased resources

1. Create a server: Please specify the reservation id as a scheduler hint.

```
openstack server create --flavor <flavor> --image <image> --network
↪<network> --hint reservation=4d3dd68f-0e3f-4f6b-bef7-617525c74ccb
↪<server-name>
```

3.1.2 Instance Reservation

Prerequisites

The following packages should be installed:

- blazar
- blazar-nova
- python-blazarclient

1. Add hosts into the freepool

1. Add hosts into the Blazar freepool using the host-create command:

```
blazar host-create compute-1
```

2. Check hosts in the freepool:

```
blazar host-list
```

Result:

```
+----+---------------------+-------+-----------+-----------+
| id | hypervisor_hostname | vcpus | memory_mb | local_gb |
+----+---------------------+-------+-----------+-----------+
| 1  | compute-1           |     2 |      3951 |        38 |
+----+---------------------+-------+-----------+-----------+
```

2. Create a lease

1. Create a lease (instance reservation) using lease-create command:

```
blazar lease-create --reservation resource_type=virtual:instance,vcpus=1,
↪memory_mb=1024,disk_gb=20,amount=1 --start-date "2020-07-24 20:00" --end-
↪date "2020-08-09 21:00" lease-1
```

Result:

```
+---------------+-------------------------------------------------------------
↪-----------------------------------------------------------+
| Field         | Value                                                       ␣
↪                                                           |
```

(continues on next page)

(continued from previous page)

```
+---------------+-------------------------------------------------------------
↪--------------------------------------------------------------+
| action        |                                                              ␣
↪                                                             |
| created_at    | 2017-07-31 07:55:59                                          ␣
↪                                                             |
| end_date      | 2020-08-09T21:00:00.000000                                   ␣
↪                                                             |
| events        | {"status": "UNDONE", "lease_id": "becf2f3b-0177-4c0f-
↪a7e7-0123370849a3", "event_type": "end_lease", "created_at":        |
|               | "2017-07-31 07:55:59", "updated_at": null, "time": "2020-
↪08-09T21:00:00.000000", "id": "0f269526-c32d-4e53-bc6b-          |
|               | 09fb7adf4354"}                                               ␣
↪                                                             |
|               | {"status": "UNDONE", "lease_id": "becf2f3b-0177-4c0f-
↪a7e7-0123370849a3", "event_type": "start_lease", "created_at":      |
|               | "2017-07-31 07:55:59", "updated_at": null, "time": "2020-
↪07-24T20:00:00.000000", "id": "7dbf3904-7d23-4db3-bfbd-          |
|               | 5cc8cb9d4d92"}                                               ␣
↪                                                             |
|               | {"status": "UNDONE", "lease_id": "becf2f3b-0177-4c0f-
↪a7e7-0123370849a3", "event_type": "before_end_lease", "created_at": |
|               | "2017-07-31 07:55:59", "updated_at": null, "time": "2020-
↪08-07T21:00:00.000000", "id": "f16151d4-04b4-403c-              |
|               | b0d7-f60d3810e37e"}                                          ␣
↪                                                             |
| id            | becf2f3b-0177-4c0f-a7e7-0123370849a3                         ␣
↪                                                             |
| name          | lease-1                                                      ␣
↪                                                             |
| project_id    | 6f6f9b596d47441294eb40f565063833                             ␣
↪                                                             |
| reservations  | {"status": "pending", "memory_mb": 1024, "lease_id":
↪"becf2f3b-0177-4c0f-a7e7-0123370849a3", "disk_gb": 20,              |
|               | "resource_id": "061198b0-53e4-4545-9d85-405ca93a7bdf",
↪"created_at": "2017-07-31 07:55:59", "updated_at": "2017-07-31    |
|               | 07:55:59", "aggregate_id": 3, "server_group_id":
↪"ba03ebb4-e55c-4da4-9d39-87e13354f3b7", "amount": 1, "affinity": null,  |
|               | "flavor_id": "db83d6fd-c69c-4259-92cf-012db2e55a58", "id
↪": "db83d6fd-c69c-4259-92cf-012db2e55a58", "vcpus": 1,           |
|               | "resource_type": "virtual:instance"}                         ␣
↪                                                             |
| start_date    | 2020-07-24T20:00:00.000000                                   ␣
↪                                                             |
| status        |                                                              ␣
↪                                                             |
| status_reason |                                                              ␣
↪                                                             |
| trust_id      | 65da707498914c7992ee7170647a3472                             ␣
↪                                                             |
| updated_at    |                                                              ␣
↪                                                             |
| user_id       |                                                              ␣
↪                                                             |
+---------------+-------------------------------------------------------------
↪--------------------------------------------------------------+
```

2. Check leases:

```
blazar lease-list
```

Result:

```
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
| id                                   | name    | start_date              ␣
↪   | end_date                    |
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
| becf2f3b-0177-4c0f-a7e7-0123370849a3 | lease-1 | 2020-07-24T20:00:00.
↪000000 | 2020-08-09T21:00:00.000000 |
+--------------------------------------+---------+-------------------------
↪---+-----------------------------+
```

3. Use the leased resources

While the reservation you created is active you can see and use the flavor of your reservation.

```
openstack flavor list
```

Result:

```
+--------------------------------------+-----------------------------------
↪---------------+-----------+------+-----------+------+-------+-----------
↪--+-----------+
| ID                                   | Name                              ␣
↪                | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_
↪Factor | Is_Public |
+--------------------------------------+-----------------------------------
↪---------------+-----------+------+-----------+------+-------+-----------
↪--+-----------+
| 1                                    | m1.tiny                           ␣
↪                | 512       | 1    | 0         |      | 1     | 1.0       ␣
↪  | True      |
| 2                                    | m1.small                          ␣
↪                | 2048      | 20   | 0         |      | 1     | 1.0       ␣
↪  | True      |
| 3                                    | m1.medium                         ␣
↪                | 4096      | 40   | 0         |      | 2     | 1.0       ␣
↪  | True      |
| 4                                    | m1.large                          ␣
↪                | 8192      | 80   | 0         |      | 4     | 1.0       ␣
↪  | True      |
| 5                                    | m1.xlarge                         ␣
↪                | 16384     | 160  | 0         |      | 8     | 1.0       ␣
↪  | True      |
| c1                                   | cirros256                         ␣
↪                | 256       | 0    | 0         |      | 1     | 1.0       ␣
↪  | True      |
| d1                                   | ds512M                            ␣
↪                | 512       | 5    | 0         |      | 1     | 1.0       ␣
↪  | True      |
```

(continues on next page)

(continued from previous page)

```
| d2                                   | ds1G
↪                | 1024      | 10   | 0          |      | 1     | 1.0
↪   | True      |
| d3                                   | ds2G
↪                | 2048      | 10   | 0          |      | 2     | 1.0
↪   | True      |
| d4                                   | ds4G
↪                | 4096      | 20   | 0          |      | 4     | 1.0
↪   | True      |
| db83d6fd-c69c-4259-92cf-012db2e55a58 | reservation:db83d6fd-c69c-4259-
↪92cf-012db2e55a58 | 1024      | 20   | 0          |      | 1     | 1.0
↪     | False     |
+--------------------------------------+-------------------------------------
↪---------------+-----------+------+-----------+------+-------+-----------
↪--+-----------+
```

1. Create a server: Please specify the flavor of the reservation.

```
openstack server create --flavor db83d6fd-c69c-4259-92cf-012db2e55a58 --
↪image <image> --network <network> <server-name>
```

3.1.3 Floating IP Reservation

Prerequisites

The following packages should be installed:

- blazar
- neutron
- python-blazarclient

The floating IP plugin should be enabled in `blazar.conf`:

```
[manager]
plugins = virtual.floatingip.plugin
```

1. Create reservable floating IPs

1. The operator should create floating IPs as reservable resources using the floatingip-create command. They must select floating IPs that are not part of an allocation pool in Neutron. For example, to create a reservable floating IP with address `172.24.4.2` from the Neutron network with ID `81fabec7-00ae-497a-b485-72f4bf187d3e`, run:

```
blazar floatingip-create 81fabec7-00ae-497a-b485-72f4bf187d3e 172.24.4.2
```

2. Check reservable floating IPs:

```
blazar floatingip-list
```

Result:

```
+--------------------------------------+---------------------+-------------
↪-------------------------+
| id                                   | floating_ip_address | floating_
↪network_id                  |
+--------------------------------------+---------------------+-------------
↪-------------------------+
| 67720c36-4d53-41e6-acec-7d3fb9436fd5 | 172.24.4.2          | 81fabec7-
↪00ae-497a-b485-72f4bf187d3e |
+--------------------------------------+---------------------+-------------
↪-------------------------+
```

2. Create a lease

1. Create a lease (floating IP reservation) using the lease-create command. Note that `python-blazarclient` version 2.2.1 or greater is required to use this feature. When you use `resource_type=virtual:floatingip`, the following parameters are supported:
   - `network_id`: UUID of the external network to reserve from (required)
   - `amount`: number of floating IPs to reserve (optional, defaults to 1)
   - `required_floatingips`: list of specific floating IPs to allocate (optional, must be formatted as a JSON array)

```
blazar lease-create --reservation 'resource_type=virtual:floatingip,
↪network_id=81fabec7-00ae-497a-b485-72f4bf187d3e,amount=2,required_
↪floatingips=["172.24.4.2","172.24.4.3"]' fip-lease
```

Result:

```
Created a new lease:
+--------------+-------------------------------------------------------------
↪--+
| Field        | Value                                                       ␣
↪   |
+--------------+-------------------------------------------------------------
↪--+
| created_at   | 2019-09-23 08:33:22                                         ␣
↪   |
| degraded     | False                                                       ␣
↪   |
| end_date     | 2019-09-24T08:33:00.000000                                  ␣
↪   |
| events       | {                                                           ␣
↪   |
|              |     "status": "UNDONE",                                     ␣
↪   |
|              |     "lease_id": "d67f3bcf-cb82-4c7d-aa4d-49cc48586d89",     ␣
↪   |
|              |     "event_type": "before_end_lease",                       ␣
↪   |
|              |     "created_at": "2019-09-23 08:33:22",                    ␣
↪   |
|              |     "updated_at": null,                                     ␣
↪   |
```

(continues on next page)

(continued from previous page)

```
|               |     "time": "2019-09-24T07:33:00.000000",                    ␣
↪  |
|               |     "id": "628e6eec-d157-4e6a-9238-47c008f357be"             ␣
↪  |
|               | }                                                            ␣
↪  |
|               | {                                                            ␣
↪  |
|               |     "status": "UNDONE",                                      ␣
↪  |
|               |     "lease_id": "d67f3bcf-cb82-4c7d-aa4d-49cc48586d89",      ␣
↪  |
|               |     "event_type": "end_lease",                               ␣
↪  |
|               |     "created_at": "2019-09-23 08:33:22",                     ␣
↪  |
|               |     "updated_at": null,                                      ␣
↪  |
|               |     "time": "2019-09-24T08:33:00.000000",                    ␣
↪  |
|               |     "id": "d8a56235-3171-4097-8dd6-425788f4dd73"             ␣
↪  |
|               | }                                                            ␣
↪  |
|               | {                                                            ␣
↪  |
|               |     "status": "UNDONE",                                      ␣
↪  |
|               |     "lease_id": "d67f3bcf-cb82-4c7d-aa4d-49cc48586d89",      ␣
↪  |
|               |     "event_type": "start_lease",                             ␣
↪  |
|               |     "created_at": "2019-09-23 08:33:22",                     ␣
↪  |
|               |     "updated_at": null,                                      ␣
↪  |
|               |     "time": "2019-09-23T08:33:00.000000",                    ␣
↪  |
|               |     "id": "f7322caf-9470-4281-b980-dcd76b3e476c"             ␣
↪  |
|               | }                                                            ␣
↪  |
| id            | d67f3bcf-cb82-4c7d-aa4d-49cc48586d89                         ␣
↪  |
| name          | fip-lease                                                    ␣
↪  |
| project_id    | 10b4b88b67e141aeb093fec48c93232c                             ␣
↪  |
| reservations | {                                                             ␣
↪  |
|               |     "status": "pending",                                     ␣
↪  |
|               |     "lease_id": "d67f3bcf-cb82-4c7d-aa4d-49cc48586d89",      ␣
↪  |
|               |     "resource_id": "ae205735-970e-4f91-a2fc-c99fc7cc45fc",
↪  |
```

(continues on next page)

(continued from previous page)

```
|               |     "network_id": "81fabec7-00ae-497a-b485-72f4bf187d3e",␣
↪   |
|               |     "created_at": "2019-09-23 08:33:22",                  ␣
↪   |
|               |     "updated_at": "2019-09-23 08:33:22",                  ␣
↪   |
|               |     "required_floatingips": [                             ␣
↪   |
|               |         "172.24.4.2",                                     ␣
↪   |
|               |         "172.24.4.3"                                      ␣
↪   |
|               |     ],                                                    ␣
↪   |
|               |     "missing_resources": false,                           ␣
↪   |
|               |     "amount": 2,                                          ␣
↪   |
|               |     "id": "30f72423-db81-4f13-bc78-b931c4a96b48",         ␣
↪   |
|               |     "resource_type": "virtual:floatingip",                ␣
↪   |
|               |     "resources_changed": false                            ␣
↪   |
|               | }                                                         ␣
↪   |
| start_date    | 2019-09-23T08:33:00.000000                                ␣
↪   |
| status        | PENDING                                                   ␣
↪   |
| trust_id      | 0617c18ba83d4ec29832b0ec19c5ae5e                          ␣
↪   |
| updated_at    | 2019-09-23 08:33:23                                       ␣
↪   |
| user_id       | 9e43ffa598d14bac91fc889c2e15cd13                          ␣
↪   |
+---------------+-----------------------------------------------------------
↪--+
```

2. Check leases:

```
blazar lease-list
```

Result:

```
+--------------------------------------+-----------+-----------------------
↪-----+----------------------------+
| id                                   | name      | start_date            ␣
↪     | end_date                   |
+--------------------------------------+-----------+-----------------------
↪-----+----------------------------+
| d67f3bcf-cb82-4c7d-aa4d-49cc48586d89 | fip-lease | 2019-09-23T08:33:00.
↪000000 | 2019-09-24T08:33:00.000000 |
+--------------------------------------+-----------+-----------------------
↪-----+----------------------------+
```

3. Update a lease

1. Update a lease (floating IP reservation) using the lease-update command. Note that `python-blazarclient` version 2.2.1 or greater is required to use this feature. After passing the existing reservation ID to the `--reservation` option, you can modify start or end dates as well as some reservation parameters:

   - `amount`: you can modify the number of floating IPs to reserve. Reducing `amount` is supported only for pending reservations.
   - `required_floatingips`: you can only reset the list of specific floating IPs to allocate to an empty list

```
blazar lease-update --reservation 'id=e80033e6-5279-461d-9573-dec137233434,
↪amount=3,required_floatingips=[]' fip-lease
```

Result:

```
Updated lease: fip-lease
```

2. Check updated lease:

```
blazar lease-show fip-lease
```

Result:

```
+--------------+--------------------------------------------------------------
↪--+
| Field        | Value                                                        ␣
↪   |
+--------------+--------------------------------------------------------------
↪--+
| created_at   | 2019-09-23 08:09:51                                          ␣
↪   |
| degraded     | False                                                        ␣
↪   |
| end_date     | 2019-09-24T08:09:00.000000                                   ␣
↪   |
| events       | {                                                            ␣
↪   |
|              |     "status": "UNDONE",                                      ␣
↪   |
|              |     "lease_id": "5d528d8d-c023-4792-ae77-cb6d4dc2c162",      ␣
↪   |
|              |     "event_type": "before_end_lease",                        ␣
↪   |
|              |     "created_at": "2019-09-23 08:09:51",                     ␣
↪   |
|              |     "updated_at": null,                                      ␣
↪   |
|              |     "time": "2019-09-24T07:09:00.000000",                    ␣
↪   |
|              |     "id": "352521cc-bfe9-4881-9a3e-2ac770671144"             ␣
↪   |
|              | }                                                            ␣
↪   |
|              | {                                                            ␣
↪   |
```

(continues on next page)

(continued from previous page)

```
|                | "status": "DONE",                                                 ␣
↪  |
|                | "lease_id": "5d528d8d-c023-4792-ae77-cb6d4dc2c162",               ␣
↪  |
|                | "event_type": "start_lease",                                      ␣
↪  |
|                | "created_at": "2019-09-23 08:09:51",                              ␣
↪  |
|                | "updated_at": "2019-09-23 08:10:10",                              ␣
↪  |
|                | "time": "2019-09-23T08:09:00.000000",                             ␣
↪  |
|                | "id": "59e1e170-660e-4a2d-a9e7-167fd5741ff5"                      ␣
↪  |
|                | }                                                                 ␣
↪  |
|                | {                                                                 ␣
↪  |
|                | "status": "UNDONE",                                               ␣
↪  |
|                | "lease_id": "5d528d8d-c023-4792-ae77-cb6d4dc2c162",               ␣
↪  |
|                | "event_type": "end_lease",                                        ␣
↪  |
|                | "created_at": "2019-09-23 08:09:51",                              ␣
↪  |
|                | "updated_at": null,                                               ␣
↪  |
|                | "time": "2019-09-24T08:09:00.000000",                             ␣
↪  |
|                | "id": "fda0d28d-afe5-4ebb-bea0-50ab1f8d7182"                      ␣
↪  |
|                | }                                                                 ␣
↪  |
| id             | 5d528d8d-c023-4792-ae77-cb6d4dc2c162                              ␣
↪  |
| name           | fip-lease                                                         ␣
↪  |
| project_id     | 10b4b88b67e141aeb093fec48c93232c                                  ␣
↪  |
| reservations | {                                                                 ␣
↪  |
|                | "status": "active",                                               ␣
↪  |
|                | "lease_id": "5d528d8d-c023-4792-ae77-cb6d4dc2c162",               ␣
↪  |
|                | "resource_id": "543a350b-c703-48c9-a97e-2e787c26e385",
↪  |
|                | "network_id": "81fabec7-00ae-497a-b485-72f4bf187d3e",             ␣
↪  |
|                | "created_at": "2019-09-23 08:09:51",                              ␣
↪  |
|                | "updated_at": "2019-09-23 08:10:10",                              ␣
↪  |
|                | "required_floatingips": [],                                       ␣
↪  |
```

(continues on next page)

(continued from previous page)

```
|               |     "missing_resources": false,                                 ␣
↪   |
|               |     "amount": 3,                                                ␣
↪   |
|               |     "id": "e80033e6-5279-461d-9573-dec137233434",                ␣
↪   |
|               |     "resource_type": "virtual:floatingip",                      ␣
↪   |
|               |     "resources_changed": false                                  ␣
↪   |
|               | }                                                               ␣
↪   |
| start_date    | 2019-09-23T08:09:00.000000                                      ␣
↪   |
| status        | ACTIVE                                                          ␣
↪   |
| trust_id      | 707391571cd14bd9bfc8eaf986163b37                                ␣
↪   |
| updated_at    | 2019-09-23 08:15:51                                             ␣
↪   |
| user_id       | 9e43ffa598d14bac91fc889c2e15cd13                                ␣
↪   |
+---------------+-----------------------------------------------------------------
↪--+
```

4. Use the leased resources

1. Once the lease becomes active, the allocated floating IPs are tagged with the reservation ID, in this case `e80033e6-5279-461d-9573-dec137233434`, and can be displayed with the following command:

```
openstack floating ip list --tags reservation:e80033e6-5279-461d-9573-
↪dec137233434
```

Result:

```
+--------------------------------------+---------------------+-------------
↪-----+------+--------------------------------------+----------------------
↪-------------+
| ID                                   | Floating IP Address | Fixed IP␣
↪Address | Port | Floating Network                     | Project              ␣
↪             |
+--------------------------------------+---------------------+-------------
↪-----+------+--------------------------------------+----------------------
↪-------------+
| 3954b799-4957-4e9f-96b7-46f72604c973 | 172.24.4.4          | None         ␣
↪     | None | 81fabec7-00ae-497a-b485-72f4bf187d3e |␣
↪10b4b88b67e141aeb093fec48c93232c |
| ae26069c-f7e9-4b8d-8ca0-6770c025dfae | 172.24.4.3          | None         ␣
↪     | None | 81fabec7-00ae-497a-b485-72f4bf187d3e |␣
↪10b4b88b67e141aeb093fec48c93232c |
| b427c171-30fe-45c4-a00b-3d5ca9b00306 | 172.24.4.2          | None         ␣
↪     | None | 81fabec7-00ae-497a-b485-72f4bf187d3e |␣
↪10b4b88b67e141aeb093fec48c93232c |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+--------------------+--------------
↪------+-------+--------------------------------------+--------------------
↪--------------+
```

2. Use the reserved floating IP like a regular one, for example by attaching it to an instance with `openstack server add floating ip`.

CHAPTER

FOUR

API REFERENCE

4.1 Blazar REST API docs

CHAPTER

FIVE

USER GUIDE

5.1 User Guide

5.1.1 Introduction

Blazar is a resource reservation service for OpenStack. Idea of creating Blazar originated with two different use cases:

- Compute host reservation (when user with admin privileges can reserve hardware resources that are dedicated to the sole use of a project)
- Virtual machine (instance) reservation (when user may ask reservation service to provide him working VM not necessarily now, but also in the future)

Now these ideas have been transformed to more general view: with Blazar, user can request the resources of cloud environment to be provided (leased) to his project for specific amount of time, immediately or in the future.

Both virtual (Instances, Volumes, Networks) and hardware (full hosts with specific characteristics of RAM, CPU, etc) resources can be allocated via lease.

In terms of benefits added, Resource Reservation Service will:

- improve visibility of cloud resources consumption (current and planned for future);
- enable cloud resource planning based on current and future demand from end users;
- automate the processes of resource allocation and reclaiming;
- provide energy efficiency for physical hosts (both compute and storage ones);
- potentially provide leases as billable items for which customers can be charged a flat fee or a premium price depending on the amount of reserved cloud resources and their usage.

Glossary of terms

Reservation is an allocation of certain cloud resource (Nova instance, Cinder volume, compute host, etc.) to a particular project. Speaking about virtual reservations, we may have not only simple, solid ones (like already mentioned instances and volumes), but also complex ones - like Heat stacks and Savanna clusters. Reservation is characterized by status, resource type, identifier and lease it belongs to.

Lease is a negotiation agreement between the provider (Blazar, using OpenStack resources) and the consumer (user) where the former agrees to make some kind of resources (both virtual and physical) available to the latter, based on a set of lease terms presented by the consumer. Here lease may be

described as a contract between user and reservation service about cloud resources to be provided right now or later. Technically speaking, lease is a group of reservations granted to a particular project upon request. Lease is characterized by start time, end time, set of individual reservations and associated events.

Event is simply something that may happen to a lease. In most simple case, event might describe lease start and lease end. Also it might be a notification to user (e.g. about soon lease expiration) and some extra actions.

Rationale

Blazar is created to:

- manage cloud resources not only right now, but also in the future;
- have dedicated resources for a certain amount of time;
- prepare for the peak loads and perform capacity planning;
- optimize energy consumption.

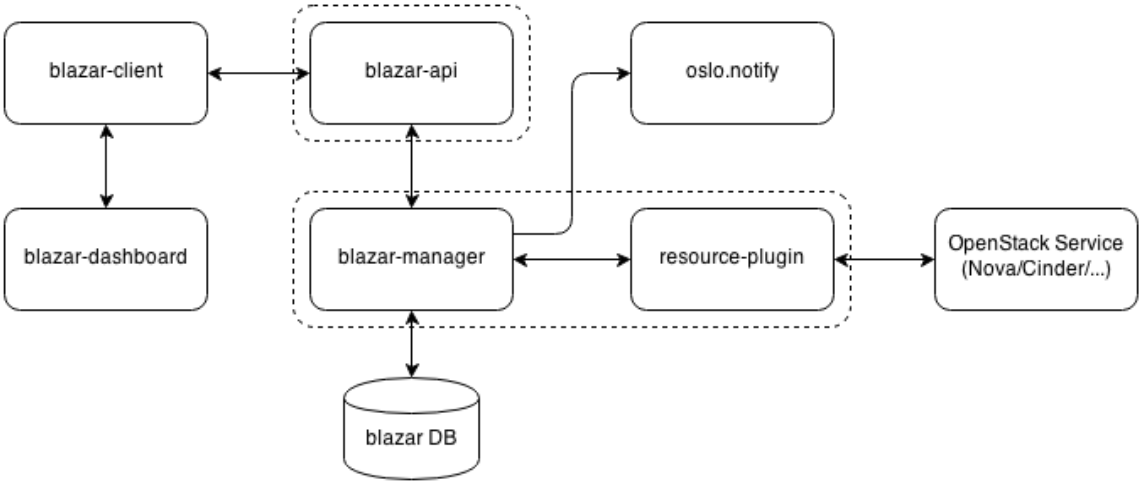
Lease types (concepts)

- **Immediate reservation**. Resources are provisioned immediately (like VM boot or moving host to reserved user aggregate) or not at all. If request can be fulfilled, lease is created and **success** status is returned. Lease should be marked as **active** or **to_be_started**. Otherwise (if request resource cannot be provisioned right now) failure status for this request should be returned.
- **Reservation with retries**. Mostly looks like previous variant, but in case of failure, user may want to have several (configurable number) retries to process lease creation action. In this case request will be processed till that will be possible to create lease but not more than set in configuration file number of times.
- **Best-effort reservation**. Also might have place if lease creation request cannot be fulfilled immediately. Best-effort mechanism starts something like scavenger hunt trying to find resources for reservations. For compute hosts reservation that makes much sense, because in case there are instances belonging to other project on eligible hosts, and without them there will be possible to reserve these hosts, Blazar may start instances migration. This operation can be timely and fairly complex and so different strategies may be applied depending on heuristic factors such as the number, type and state of the instances to be migrated. Also Blazar should assert that there are at least enough potential candidates for the migration prior to starting the actual migration. If Blazar decides to start migration, it returns **success** state and marks lease as **in_progress**, otherwise - **failure**. If this hunting ends successfully before configurable timeout has passed, lease should be marked as **active**, otherwise its status is set to **timedout**.
- **Delayed resource acquiring** or **scheduled reservation**. In this reservation type, lease is created successfully if Blazar thinks there will be enough resources to process provisioning later (otherwise this request returns **failure** status). Lease is marked as **inactive** till all resources will be actually provisioned. That works pretty nice and predictable speaking about compute hosts reservation (because hosts as resources are got not from common cloud pool, but from admin defined pool). So it is possible for Blazar to predict these physical resources usage and use that information during lease creation. If we speak about virtual reservations, here situation is more complicated, because all resources are got from common cloud resources pool, and Blazar cannot

guarantee there will be enough resources to provision them. In this failure case lease state will be marked as **error** with appropriate explanation.

5.1.2 Blazar architecture

Blazar design can be described by following diagram:



blazar-client - provides the opportunity to communicate with Blazar via *REST API* (blazar-api service).

blazar-api - waits for the REST calls from the outside world to redirect them to the manager. blazar-api communicates with blazar-manager via RPC. Runs as a separated process.

blazar-manager - implements all logic and operations with leases, reservations and events. Communicates with Blazar DB and stores there data structure of connected leases, reservations (both physical and virtual) and events. blazar-manager service is responsible for running events created for lease and process all actions that should be done this moment. Manager uses resource-plugins to work with concrete resources (instances, volumes, compute hosts).

resource-plugin - responsible for exact actions to do with reserved resources (VMs, volumes, etc.) When working knows only about resource ID and token to use. All resource plugins work in the same process as blazar-manager.

Virtual instance reservation

Note virtual instance reservation feature is not available in current release. Expected to be available in the future (bug tracker).

Virtual instance reservation mostly looks like usual instance booting for user - he/she only passes special hints to Nova containing information about future lease - lease start and end dates, its name, etc. Special Nova API extensions parse these parameter and use them to call Blazar, passing to it ID of just created instance. If there is a need to reserve all instances in cloud (like in developer labs to automate process of resource reclaiming), default reservation extension might be used. By default it starts lease at the moment of request and gives it one month of lifetime.

During the time lease has not started yet, instance will be shelved.

Compute host reservation

Now process of compute hosts reserving contains two steps:

- admin marks hosts from common pool as possible to be reserved. That is implemented by moving these hosts to special aggregate;
- user asks for reserving of host with specified characteristics like:
  - the region
  - the availability zone
  - the host capabilities extra specs (scoped and non-scoped format is accepted)
  - the number of CPU cores
  - the amount of free RAM
  - the amount of free disk space
  - the number of hosts

Technically speaking, resource ID here will be not host ID, because there might be many of them wanted. Resource here will be new aggregate containing reserved hosts. The time lease starts, user may use reserved compute capacity to run his/her instances on it passing special scheduler hint to Nova. When host is reserved, its not used for usual instance running, it might be used only when lease starts and only by passing reservation ID to Nova. That is implemented using special Nova Scheduler filter, that passes reservation ID to Blazar and checks if user really can use reserved compute capacity.

5.1.3 State machines

Blazar objects (leases, reservations, and events) each have a status. This document describes statuses in detail.

Lease status

Lease statuses are categorized into two types: stable or transitional. In the state machine shown below, stable statuses are drawn as black nodes while transitional statuses are drawn as gray nodes. Status transitions are triggered by an API call or an event in a lease.

A lease has the following four stable statuses:

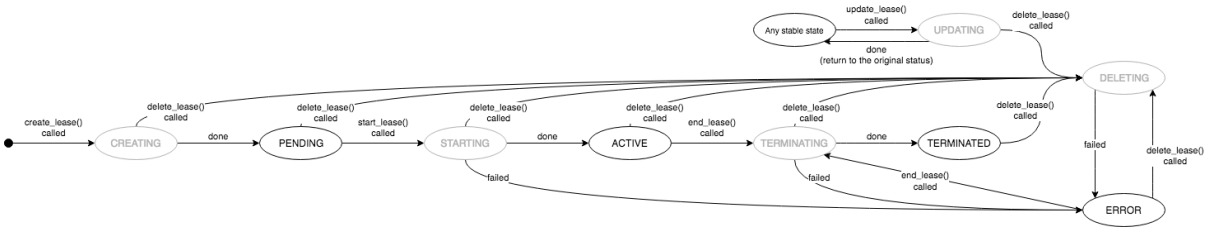
- **PENDING**: A lease has been successfully created and is ready to start. The lease stays in this status until it starts.
- **ACTIVE**: A lease has been started and is active.
- **TERMINATED**: A lease has been successfully terminated.
- **ERROR**: Unrecoverable failures happened to the lease.

Transitional statuses are as follows:

- **CREATING**: A lease is being created.
- **STARTING**: A lease is being started.
- **UPDATING**: A lease is being updated.

- **TERMINATING**: A lease is being terminated.
- **DELETING**: A lease is being deleted. Any status can change to this status because delete is the highest prioritized operation. e.g. when a lease hangs up in the STARTING status, delete should be allowed.

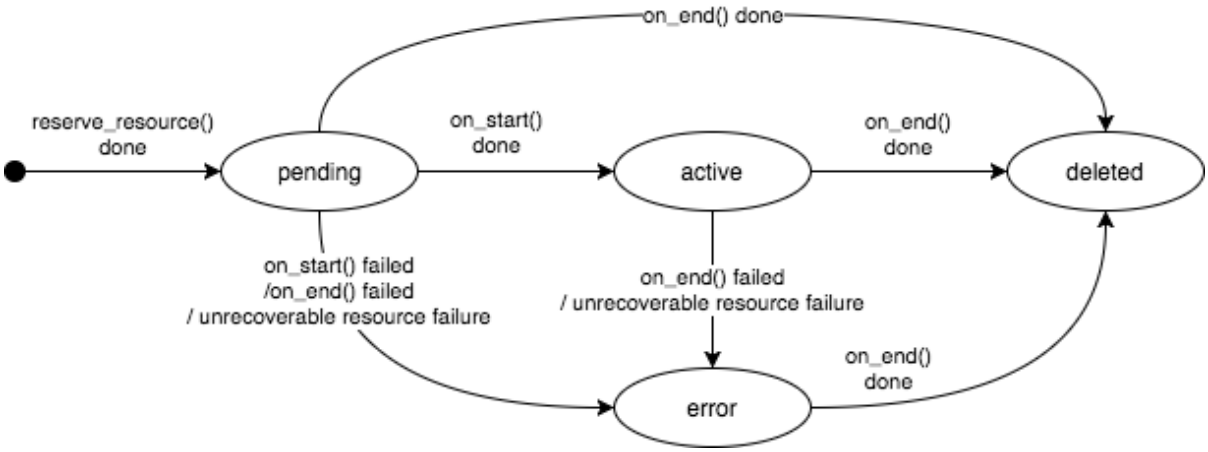
If some action can cause an invalid status transition, the action is denied. E.g. If a user sends an Update Lease request while it is starting, the Update Lease request is denied because the transition from **STARTING** to **UPDATING** is invalid.



Reservation status

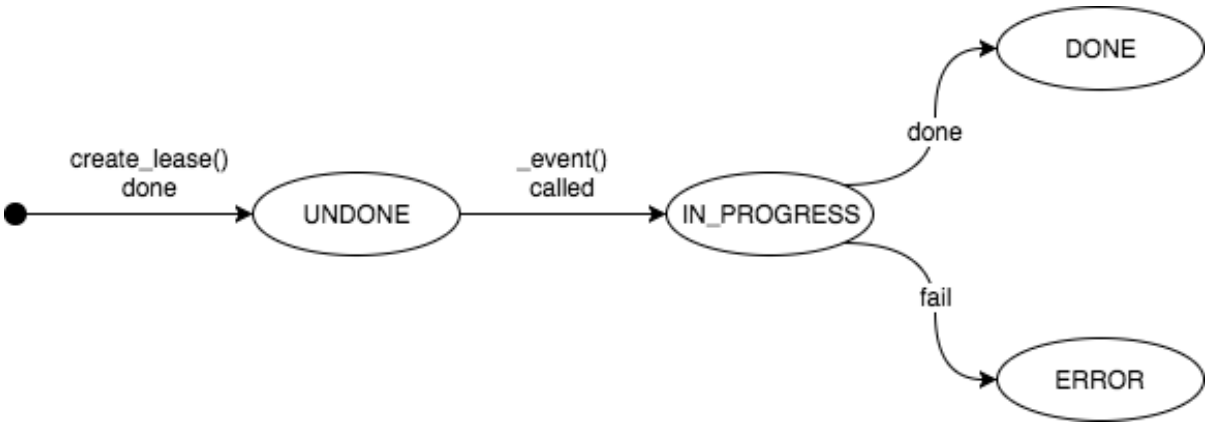
A reservation has the following four statuses. Lowercase letters are used for backward compatibility:

- **pending**: A reservation has been successfully created and is ready to start. The reservation stays in this status until it starts.
- **active**: A reservation has been started and is active.
- **deleted**: Reserved resources have been successfully released.
- **error**: Unrecoverable failures happened to resources.



Event status

Event statuses are as follows.



Relationships between statuses

The following table shows conditions of statuses of reservations and events that have to be satisfied for each lease status.

Lease	Reservations	Events
CREATING	pending	start_lease: UNDONE , end_lease: UNDONE
PENDING	pending	start_lease: UNDONE , end_lease: UNDONE
STARTING	pending or active or error	start_lease: IN_PROGRESS , end_lease: UNDONE
ACTIVE	active	start_lease: DONE , end_lease: UNDONE
TERMINATING	active or deleted or error	start_lease: DONE , end_lease: IN_PROGRESS
TERMINATED	deleted	start_lease: DONE , end_lease: DONE
DELETING	Any status	Any status
UPDATING	Any status	Any status other than IN_PROGRESS

blazar/status module

The *blazar/status* module defines and manages these statuses.

5.1.4 Resource Monitoring

Blazar monitors states of resources and heals reservations which are expected to suffer from resource failure. Resource specific functionality, e.g., calling Nova APIs, is provided as a monitoring plugin. The following sections describe the resource monitoring feature in detail.

Monitoring Type

Blazar supports 2 types of monitoring - push-based and polling-based.

1. Push-based monitoring

   The monitor listens to notification messages sent by other components, e.g., sent by Nova for the host monitoring plugin. And it picks up messages which refer to the resources managed by Blazar. Event types, topics to subscribe and notification callbacks are provided by monitoring plugins.

2. Polling-based monitoring

   The blazar-manager periodically calls a states check method of monitoring plugins. Then, the monitoring plugins check states of resources, e.g., *List Hypervisors* of the Compute API is used for the host monitoring plugin.

Admins can enable/disable these monitoring by setting configuration options.

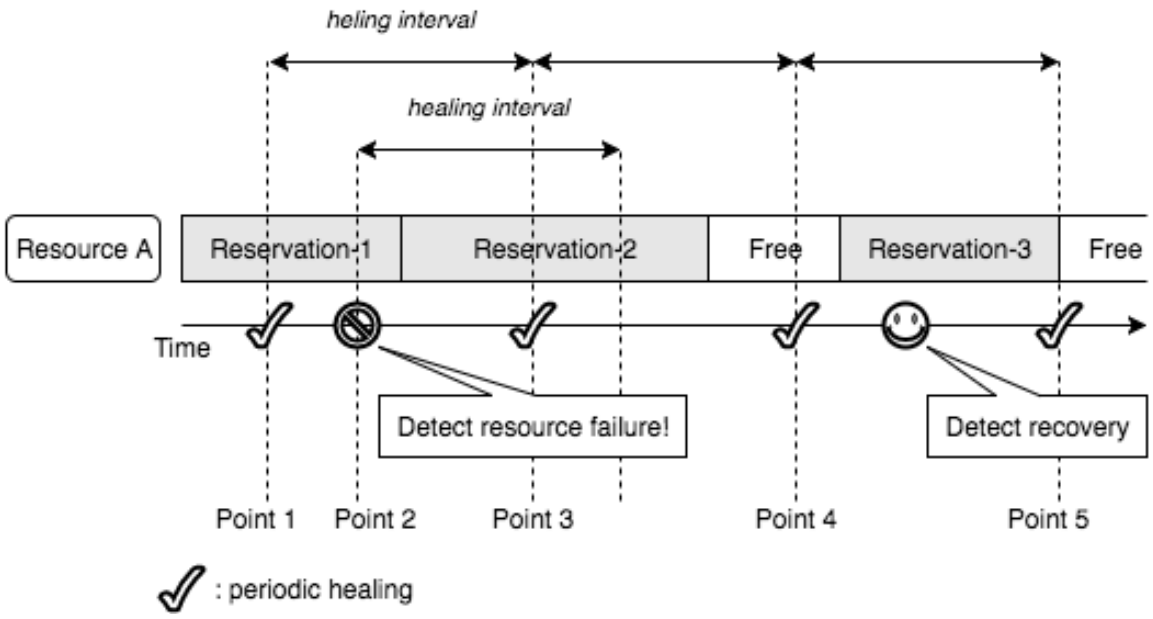
Healing

When the monitor detects a resource failure, it heals reservations which are expected to suffer from the failure. Note that it does not immediately heal all of reservations for the failed resource because the resource is expected to recover sometime in the future, i.e., the monitor heals only reservations which are active or will start soon.

In addition, the monitor periodically checks validity of reservations and heals invalid reservations. Therefore, even though the failed resource did not recover in the last interval, the periodic task heals invalid reservations which will start in the next interval.

The healing flow is as follows:

1. Resource A is reserved for the *Reservation-1*, *Reservation-2* and *Reservation-3* as shown in the following diagram.
2. At the point 1, the periodic task in the manager checks if there is any reservation to heal and it detects there is not.
3. At the point 2, the manager detects a failure of the resource A. Then, it heals active reservations and reservations which will start in the *healing interval*. In this case, *Reservation-1* and *Reservation-2* are healed immediately.
4. At the point 3, the periodic task checks if there is any reservation to heal. In this case, the task finds out there is no reservation to heal because the resource has not yet recovered but no reservation will start in next interval. *Reservation-2* has been already healed in step 3.
5. At the point 4, the periodic task checks if there is any reservation to heal again. In this case, the task finds out *Reservation-3* needs to be healed because it will start in the next interval and the resource has not yet recovered.
6. Before the point 5, the manager detects a recovery of the resource.
7. At the point 5, the periodic task finds out there is no failed resource and nothing to do.



Flags

Leases and reservations have flags that indicate states of reserved resources. Reservations have the following two flags:

- **missing_resources**: If any resource allocated to the reservation fails and no alternative resource found, this flag is set *True*.
- **resources_changed**: If any resource allocated to the *active* reservation and alternative resource is reallocated, this flag is set *True*.

Leases have the following flag:

- **degraded**: If the **missing_resources** or the **resources_changed** flags of any reservation included in the lease is *True*, then it is *True*.

Lease owners can see health of the lease and reservations included in the lease by checking these flags.

Monitoring Resources

Resource specific functionality is provided as a monitoring plugin. The following resource is currently supported.

Compute Host Monitor

Compute host monitor detects failure and recovery of compute hosts. If it detects failures, it triggers healing of host reservations and instance reservations. This document describes the compute host monitor plugin in detail.

Monitoring Type

Both of the push-based and the polling-based monitoring types are supported for the compute host monitor. These monitors can be enabled/disabled by the following configuration options:

- **enable_notification_monitor**: Set *True* to enable it.
- **enable_polling_monitor**: Set *True* to enable it.

Failure Detection

Compute host monitor detects failure and recovery hosts by subscribing Nova notifications or polling the *List Hypervisors* of Nova API. If any failure is detected, Blazar sets the *reservable* field of the failed host *False* and heals suffering reservations as follows.

Reservation Healing

If a host failure is detected, Blazar tries to heal host/instance reservations which use the failed host by reserving alternative host. The length of the *healing interval* can be configured by the *healing_interval* option.

Configurations

To enable the compute host monitor, enable *enable_notification_monitor* or *enable_polling_monitor* option, and set *healing_interval* as appropriate for your cloud. See also the *blazar.conf* in detail.

CHAPTER

SIX

ADMIN GUIDE

6.1 Administrator Guide

6.1.1 blazar-status

Synopsis

```
blazar-status <category> <command> [<args>]
```

Description

`blazar-status` is a tool that provides routines for checking the status of a Blazar deployment.

Options

The standard pattern for executing a **`blazar-status`** command is:

```
blazar-status <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
blazar-status
```

Categories are:

- `upgrade`

Detailed descriptions are below.

You can also run with a category argument such as `upgrade` to see a list of all commands in that category:

```
blazar-status upgrade
```

These sections describe the available categories and arguments for **`blazar-status`**.

Upgrade

`blazar-status upgrade check` Performs a release-specific readiness check before restarting services with new code. This command expects to have complete configuration and access to databases and services.

Return Codes

Return code	Description
0	All upgrade readiness checks passed successfully and there is nothing to do.
1	At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.
2	There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.
255	An unexpected error occurred.

History of Checks

3.0.0 (Stein)

- Placeholder to be filled in with checks as they are added in Stein.

6.1.2 Usage Enforcement

Synopsis

Usage enforcement and lease constraints can be implemented by operators via custom usage enforcement filters.

Description

Usage enforcement filters are called on `lease_create`, `lease_update` and `on_end` operations. The filters check whether or not lease values or allocation criteria pass admin defined thresholds. There is currently one filter provided out-of-the-box. The `MaxLeaseDurationFilter` restricts the duration of leases.

Options

All filters are a subclass of the BaseFilter class located in `blazar/enforcement/filter/base_filter.py`. Custom filters must implement methods for `check_create`, `check_update`, and `on_end`. The `MaxLeaseDurationFilter` is a good example to follow. Filters are enabled in `blazar.conf` under the `[enforcement]` group. For example, enabling the `MaxLeaseDurationFilter` to limit lease durations to only one day would work as follows:

```
[enforcement]
enabled_filters = MaxLeaseDurationFilter
max_lease_duration = 86400
```

MaxLeaseDurationFilter

This filter simply examines the lease `start_date` and `end_date` attributes and rejects the lease if its duration exceeds a threshold. It supports two configuration options:

- `max_lease_duration`
- `max_lease_duration_exempt_project_ids`

See the *blazar.conf* page for a description of these options.

CHAPTER

SEVEN

FOR CONTRIBUTORS

7.1 Contributor Guide

7.1.1 How to contribute

Getting started

- Read the OpenStack Developers Guide
- Login to OpenStack Gerrit using your Launchpad ID
  - Sign the OpenStack Individual Contributor License Agreement
  - Check that your email is listed in Gerrit identities
- Subscribe to Blazar-related projects on OpenStack Gerrit. Go to your settings and in the watched projects add *openstack/blazar*, *openstack/blazar-nova*, *openstack/python-blazarclient* and *openstack/blazar-dashboard*.

As all bugs/blueprints are listed in Blazar Launchpad, you may keep track on them and choose some to work on.

How to keep in touch with community

- If youre not yet subscribed to the OpenStack general mailing list or to the OpenStack development mailing list, please do. Blazar-related emails must be send with **[blazar]** in the subject.
- All questions may be asked on our IRC channel #openstack-blazar on OFTC.
- We also have weekly meetings on #openstack-meeting-alt. Please check meeting details.

Your first commit to Blazar

- Read the OpenStack development workflow documentation
- Clone the corresponding Blazar repository: blazar, blazar-nova, client, blazar-dashboard
- Apply and commit your changes
- Make sure all code checks and tests have passed
- Send your patch for review
- Monitor the status of your change on https://review.openstack.org/

7.1.2 Development guidelines

Coding Guidelines

PEP8 checks should pass for all Blazar code. You may check it using the following command:

```
tox -e pep8
```

Also you should keep your code clear using more code style checks via pylint:

```
tox -e pylint
```

If you see any pep8/pylint errors in your code, it is mandatory to fix them before sending your change for review.

Testing Guidelines

Blazar repositories have unit tests that are run on all submitted code, and it is recommended for developers to execute them themselves to catch regressions early. Developers are also expected to keep the test suite up-to-date with any submitted code changes.

Unit tests might be run in TOX environments via the commands:

```
tox -e py36
tox -e py37
```

for Python 3.6 and Python 3.7 accordingly.

Note that the Blazar code base is not yet compatible with Python 3, so tests will be failing.

Note that some tests might use databases, the script `tools/test-setup.sh` sets up databases for the unit tests.

Documentation Guidelines

Currently Blazar docs are partially written on OpenStack wiki pages, and partially using Sphinx / RST located in the main repo in *doc* directory.

To build Sphinx / RST docs locally run the following command:

```
tox -e docs
```

Then you can access generated docs in the *doc/build/* directory, for example, the main page would be *doc/build/html/index.html*.

7.2 References

The blazar project has lots of complicated parts in it where it helps to have an overview to understand how the internals of a particular part work.

7.2.1 Internals

The following is a dive into some of the internals in blazar.

- *REST API Version History*: How blazar uses API microversion.

REST API Version History

This documents the changes made to the REST API with every microversion change. The description for each version should be a verbose one which has enough information to be suitable for use in user documentation.

1.0

This is the initial version of the v1.0 API which supports microversions. The v1.0 API is from the REST API userss point of view exactly the same as v1 except with strong input validation.

A user can specify a header in the API request:

```
OpenStack-API-Version: <version>
```

where `<version>` is any valid api version for this API.

If no version is specified then the API will behave as if a version request of v1.0 was requested.

CHAPTER

EIGHT

SPECS

CHAPTER

NINE

INDICES AND TABLES

- genindex
- search