B3LB

Release 1.2

Jan 18, 2022

CONTENTS

B3LB is a open source BigBlueButton API load balancer similar to Scalelite. B3LB is based on the Django Python Web framework and is designed to work in large scale-out deployments with 100+ BigBlueButton nodes and high attendee join rates. The project started at IBH IT-Service GmbH in the fall 2020 and has been published in February 2021 during the second lock down in Germany. B3LB is licensed under the GNU Affero General Public License v3.0.

Note: This project uses BigBlueButton and is not endorsed or certified by BigBlueButton Inc. BigBlueButton and the BigBlueButton Logo are trademarks of BigBlueButton Inc.

CHAPTER

ONE

ABOUT

B3LB is based on the Django Python Web framework and is designed to work in large scale-out deployments with 100+ BigBlueButton nodes and high attendee join rates.

1.1 Architecture

To scale for a huge number of attendees it is possible to:

- scale-out the B3LB API frontends
- scale-out the B3LB polling workers
- scale-out your BBB nodes

1.2 Features

- multiple b3lb frontend instances
- backend BBB node polling using Celery
- extensive caching based on Redis
- robust against high BBB node response times (i.e. due to ongoing DDoS attacks)

1.2.1 BBB Clustering

- supports a high number of BBB nodes
- different load balancing factors per cluster
- load calculation by attendees, meetings and CPU load metrics
- maintenance mode allows to disable BBB nodes gracefully

1.2.2 BBB Frontend API

- deployed on ASGI with uvicorn
- HTTP call-outs are implementated async using aiohttp
- support API key rollover using a second secret
- prebuild responses for expensive API calls (`getMeetings`)
- limiting attendees or meetings per tenant
- does not implement but blocks recording API calls

1.2.3 Multitenancy

- per-tenant API hostnames
- start presentation injection
- branding logo injection
- multiple API secrets per tenant

1.2.4 Monitoring

- simple health-check URL
- simple json statistics URL
- prometheus metrics URL

1.3 Load Calculation

To select a BBB node for new meetings B3LB calculates a load value for the BBB nodes. The BBB node with the lowest load value is choosen. The load is based on three metrics:

- number of attendees
- number of meetings
- cpu utilization (base 10.000)

Each of the metrics is important for deciding where to spawn new meetings. The cpu utilization depends on the current load caused by running meetings and also respects external effects of the BBB nodes. The number of meetings is important since it is an indicator that more attendees may join and cause even more load in the future.

Metric	Description	Origin
cpu_{15s}	cpu utilization in the last 15s	node
cpu_{1m}	cpu utilization in the last minute	node
n_{atn}	number of active attendees	node
n_{mtg}	number of active meetings	node

Tunable	Default	Description	Origin
cpu_{max}	5.000	target max cpu utilization	cluster
cpu_{order}	6	order of the polynomial	cluster
f_{atn}	1	load factor for a single attendee	cluster
f_{mtg}	30	load factor for a single meeting	cluster

$$load_{node} = f_{atn} * n_{atn} + f_{mtg} * n_{mtg} + \frac{cpu_{max}}{cpu_{order}} * \sum_{n=1}^{cpu_{order}} \left[\frac{\max\left(cpu_{1m}, cpu_{15s}\right)}{10.000} \right]^n$$

The cpu utilization is reinforced to get a slow increase as long the cpu utilization is low and increases more and more steep. The following plot shows the load value for a BBB node depending on it's CPU utilization (base 10.000) for different attendee and meeting counts.

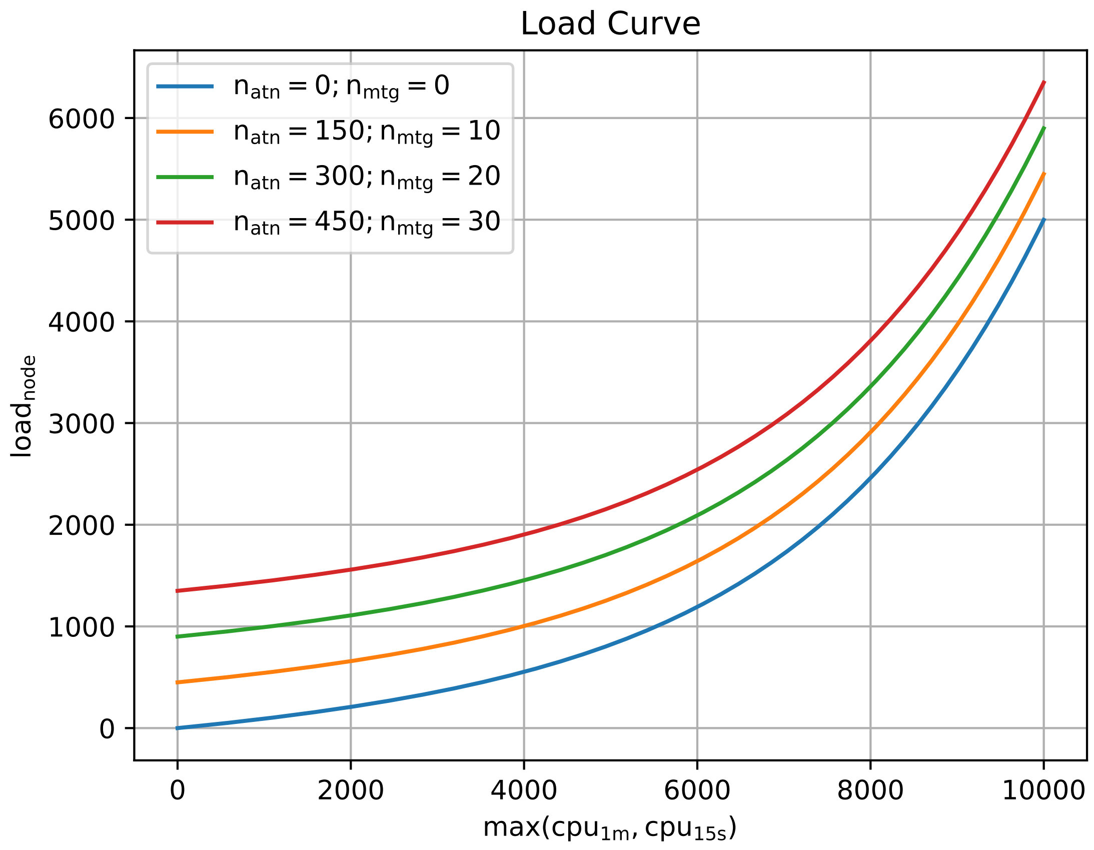
Tuning the polynomial order changes the load balancing to be more or less cpu load sensitive:

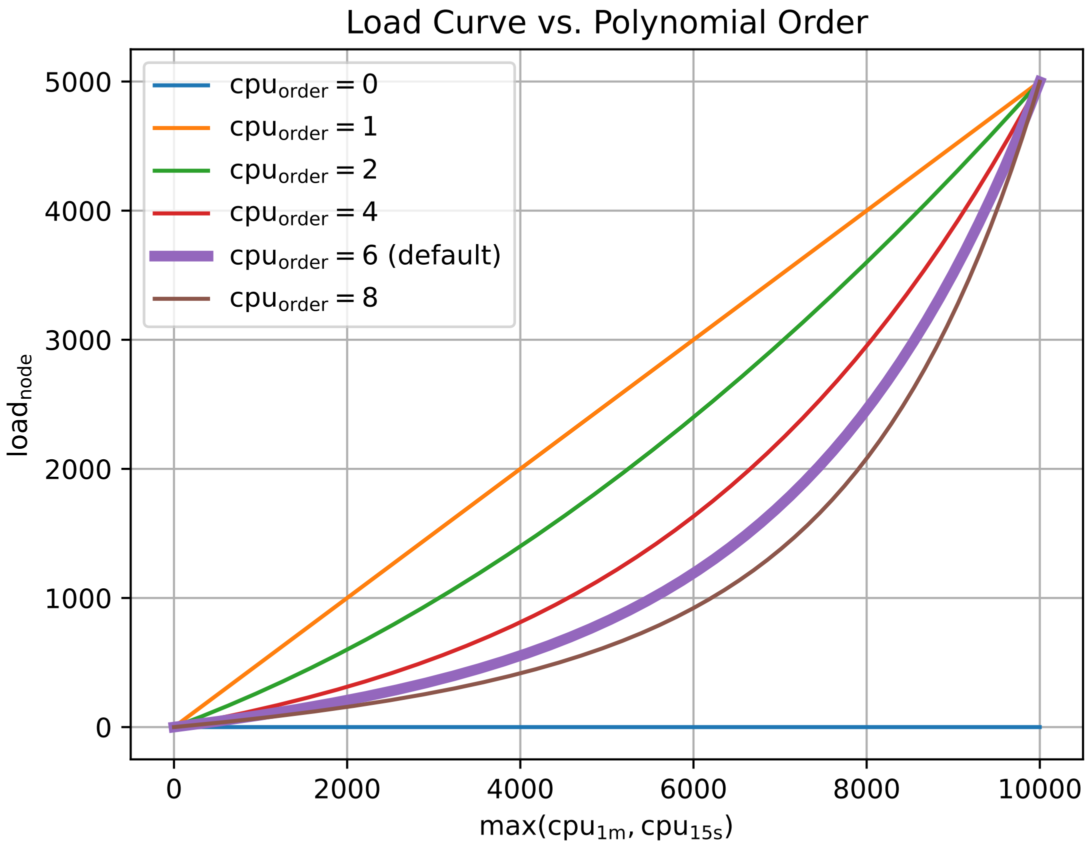
1.4 Container Images

B3LB provides in three different docker image provided on Quay.io and GitHub Packages. The images can be build from source using the provided Dockerfiles.

Hint: It is intentional that there are no *b3lb:latest* nor *b3lb-static:latest* image tags available. You should always pick a explicit version for your deployment.

Warning: Since Docker has stopped to support OSS no images on *Docker Hub* are provided any more for b3lb 2.2.1!


Load Curve
$n_{atn} = 0; n_{mtg} = 0$
$n_{atn} = 150; n_{mtg} = 10$
$n_{atn} = 300; n_{mtg} = 20$
$n_{atn} = 450; n_{mtg} = 30$
$load_{node}$
$\max(cpu_{1m}, cpu_{15s})$
0
1000
2000
3000
4000
5000
6000
0
2000
4000
6000
8000
10000



Load Curve vs. Polynomial Order
$cpu_{order} = 0$
$cpu_{order} = 1$
$cpu_{order} = 2$
$cpu_{order} = 4$
$cpu_{order} = 6$ (default)
$cpu_{order} = 8$
$load_{node}$
$\max(cpu_{1m}, cpu_{15s})$
0
1000
2000
3000
4000
5000
0
2000
4000
6000
8000
10000

1.4.1 b3lb

This image contains the Django files of b3lb to run the ASGI application, Celery tasks and manamgenet CLI commands.

Quay.io

```
docker pull quay.io/ibh/b3lb:2.2.2
```

GitHub Packages

```
docker pull docker.pkg.github.com/de-ibh/b3lb/b3lb:2.2.2
```

1.4.2 b3lb-static

Uses the Caddy webserver to provide static assets for the Django admin UI and can be used to publish per-tenant assets.

Quay.io

```
docker pull quay.io/ibh/b3lb-static:2.2.2
```

GitHub Packages

```
docker pull docker.pkg.github.com/de-ibh/b3lb/b3lb-static:2.2.2
```

1.4.3 b3lb-pypy

This image contains the Django files of b3lb and uses *PyPy <https://www.pypy.org/>*_ instead of CPython. This boosts the performance for the celery worker if the need to process a huge number of nodes or attendees.

Quay.io

```
docker pull quay.io/ibh/b3lb-pypy:2.2.2
```

GitHub Packages

```
docker pull docker.pkg.github.com/de-ibh/b3lb/b3lb-pypy:2.2.2
```

Warning: It is recommended to use *b3lb-pypy* for the celery workers, only. It is not well-tested for any other task and is known to waste memory. You should run it only with cgroup based memory limits engaged to prevent excessive memory swapping or OOM killing.

1.4.4 b3lb-dev

This is the development build of b3lb using Djangos single threaded build-in webserver. You should never use this in production.

Quay.io

```
docker pull ibhde/b3lb-dev:latest
```

GitHub Packages

```
docker pull docker.pkg.github.com/de-ibh/b3lb/b3lb-dev:latest
```

CHAPTER

TWO

PREREQUISITES

Hint: Some of the examples and templates are containing *Jinja2*-like variables notations `{{ variable_name }}` - you need to replace them appropriately if not used with a deployment or template engine like *ansible* with *Jinja2*.

2.1 Docker

To deploy *b3lb* you need a running Docker environment:

- Docker Engine
- Docker Swarm
- Kubernetes
- ...

This documentation expects to use *Docker Compose* for single host container deployment.

2.2 DNS

The BBB API of *B3LB* can be used with a wildcard DNS entry as well as with a single domain and different URL paths. A wildcard DNS entry is recommended as it is most similar to a standalone BBB server. Both variants can be used at the same time.

2.2.1 Wildcard DNS Entry

B3LB uses the following domain scheme:

`https://api.bbbconf.de/admin/` the *Django Admin*

`https://api.bbbconf.de/b3lb/ping` checks the health of *B3LB* including database access

`https://api.bbbconf.de/b3lb/metrics` global Prometheus metrics

`https://api.bbbconf.de/b3lb/stats` global JSON statistics

`https://tenant1.api.bbbconf.de/bigbluebutton/` BBB API URL for the tenant `tentant1`

`https://tenant1-001.api.bbbconf.de/bigbluebutton/` BBB API URL for a additional secret for the tenant `tentant1`

It is recommended to add corresponding DNS RR using a wildcard to your zone file:

```
; address records of reverse proxy instances
{{ api_base_domain }}.    A       192.0.2.1
                          A       192.0.2.2
                          A       192.0.2.3
                          AAAA    2001:db8::1
                          AAAA    2001:db8::2
                          AAAA    2001:db8::3
; wildcard used by tenants
*.{{ api_base_domain }}.  CNAME   api
```

You need to support dynamic zone updates to use wildcard certificates from *Let's'Encrypt*. The following example could be used with *bind9* to create a TSIG update key and allow zone updates.

The TSIG key can be created using the *tsig-keygen* binary:

```
root@ns:~# tsig-keygen -a hmac-sha512 "{{ tsig_key }}" > /etc/bind/traefik.key
```

Hint:

- `{{ tsig_key }}` is the name of the TSIG key
- `{{ tsig_secret }}` the secret value from the key file

Example zone definition:

```
include "/etc/bind/traefik.key";

zone "{{ api_base_domain }}" {
    type master;
    file "{{ api_base_domain }}.zone";

    allow-update { key "{{ tsig_key }}"; };
};
```

2.2.2 Single Domain Name

A single domain name can be used if the use of a wildcard DNS entry is not possible or not desired. The following URL patterns are used:

```
``https://api.bbbconf.de/admin/``
```

the *Django Admin*

`https://api.bbbconf.de/b3lb/ping` checks the health of *B3LB* including database access

`https://api.bbbconf.de/b3lb/metrics` global Prometheus metrics

`https://api.bbbconf.de/b3lb/stats` global JSON statistics

`https://api.bbbconf.de/b3lb/t/tenant1/bbb/` BBB API URL for the tenant `tentant1`

`https://api.bbbconf.de/b3lb/t/tenant1-001/bbb/` BBB API URL for a additional secret for the tenant `tentant1`

2.3 Reverse Proxy

A reverse proxy with the following features is required:

- to get a wildcard certificate from *Let's'Encrypt* the use of the ACME DNS-01 challenge is required (*recommended*)
- access ACLs to protect *b3lb* admin & metrics urls

traefik has proven to work very well for *b3lb*.

2.4 PostgreSQL Database

b3lb requires a database backend supported by Django. It needs to be accessible by all *b3lb* frontend and worker instances.

Hint: Using PostgreSQL 9.5+ is highly recommended.

CHAPTER

THREE

CONFIGURATION

b3lb can be configured using environment variables or a .env file. The following settings are available:

`SECRET_KEY` The secret key for your Django deployment.

REQUIRED

Warning: The secret key needs to be unique, secure and kept confidential. A key could be generated using the *pwgen* command:

```
pwgen -ys 50 1
```

`DATABASE_URL` The databases setting for your Django deployment. The `default` database setting is used and should point to a PostgreSQL database.

REQUIRED

`CELERY_BROKER_URL` The broker URL used by *Celery*. The broker needs to be the same on all *b3lb* instances.

REQUIRED

`CACHE_URL` The caches setting for your Django deployment. The `default` cache is used by the worker nodes to cache BBB's `getMeetings` XML data. It is recommended to use a redis backend.

Default: `locmemcache://b3lb-default-cache`

Hint: It is highly recommended to configure a powerful caching backend like *redis*. Running the worker nodes with limited caching will result in many database read requests with huge result sets. The worker nodes should use a shared cache if network throughput is non-expensive.

`CACHEOPS_REDIS` This settings needs to configure a redis backend used for ORM caching using django-cacheops. For frontend nodes it is recommend to use a local redis instance to separate failure domains.

Default: `redis://redis/2`

`CACHEOPS_DEGRADE_ON_FAILURE` The ORM caching layer will continue to work with degraded performance if the redis backend is not available.

Default: `True`

`LANGUAGE_CODE` The language ID to be used for the admin pages.

Default: `en-us`

`TIME_ZONE` The time zone to be used.

Default: `UTC`

B3LB_API_BASE_DOMAIN The b3lb *base domain*.

REQUIRED

Hint: *redis* is used in the three different settings CACHES, CELERY_BROKER_URL and CACHEOPS_REDIS. It is highly recommended to use unique redis database identifiers for each setting.

Create your own .env file using the following template:

Hint: You might use the templates with Jinja2 and set the variables api_base_domain, db_passwd and secret_key appropriately.

```
# Uvicorn: number of ASGI workers
WEB_CONCURRENCY=4

# Django Basics
SECRET_KEY={{ secret_key }}
DEBUG=False

# Django I18N
LANGUAGE_CODE=de-de
TIME_ZONE=Europe/Berlin

# Django Cache
CACHE_URL=redis://:{{ redis_secret }}@redis:6379/1

# Django ORM
DATABASE_URL=psql://b3lb-user:{{ db_passwd }}@db-host:5432/b3lb-db

# Django ORM Caching
CACHEOPS_REDIS=redis://:{{ redis_secret }}@redis:6379/2

# Configure Celery
CELERY_BROKER_URL=redis://:{{ redis_secret }}@redis:6379/3

# B3LB
B3LB_API_BASE_DOMAIN=api.bbbconf.de
```

CHAPTER

FOUR

DEPLOYMENT

b3lb is distributed as Docker image and the following deployment blueprint is based on Docker Compose. As reverse proxy traefik is used. The following templates are provided:

docker-compose.yml The compose file deploying all services on a single node. For scale-out you need some deployment environment allowing you to scale the components.

conf.d/traefik/config.yml The traefik configuration.

conf.d/traefik/traefik.yml Static middlewares for traefik used in the compose file.

Hint: You need at least to change the options tagged with `TODO:` in the following templates. You might use the templates with Jinja2 and set the variables `ansible_fqdn`, `api_base_domain`, `assets_domain`, `tsig_key` and `tsig_secret` appropriately. For single domain name setups you need to update the traefik rules labels and change the certificat resolver setting to `traefik.http.routers.*.tls.certResolver=acmeTLS`.

docker-compose.yml

```
version: '3'

services:
  # reverse proxy
  traefik:
    image: traefik:2.5.6
    read_only: true
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./conf.d/traefik/config.yml:/etc/traefik/config.yml:ro
      - ./conf.d/traefik/traefik.yml:/etc/traefik/traefik.yml:ro
      - ./data.d/traefik/acme:/etc/traefik/acme
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      RFC2136_DNS_TIMEOUT: 21
      # TODO: nameserver to be used for DNS-01 challenge
      RFC2136_NAMESERVER: 192.0.2.53
      RFC2136_TSIG_ALGORITHM: hmac-sha512.
      # TODO: tsig key used for auth
      RFC2136_TSIG_KEY: {{ tsig_key }}
      RFC2136_TSIG_SECRET: {{ tsig_secret }}
    labels:
```

(continues on next page)

(continued from previous page)

```
      - traefik.enable=true

      # HOST: publish traefik dashboard
      - traefik.http.routers.traefik.entrypoints=https
      - traefik.http.routers.traefik.rule=Host(`{{ ansible_fqdn }}`)
      - traefik.http.routers.traefik.middlewares=management-chain@file
      - traefik.http.routers.traefik.tls=true
      - traefik.http.routers.traefik.tls.options=default
      - traefik.http.routers.traefik.tls.certResolver=acmeTLS
      - traefik.http.routers.traefik.service=api@internal

      # HOST: publish traefik ping service
      - traefik.http.routers.traefik-ping.entrypoints=https
      - traefik.http.routers.traefik-ping.rule=Host(`{{ ansible_fqdn }}`) &&␣
↪PathPrefix(`/ping`)
      - traefik.http.routers.traefik-ping.middlewares=endpoint-chain@file
      - traefik.http.routers.traefik-ping.tls=true
      - traefik.http.routers.traefik-ping.tls.options=default
      - traefik.http.routers.traefik-ping.tls.certResolver=acmeTLS
      - traefik.http.routers.traefik-ping.service=ping@internal

      # HOST: publish traefik prometheus metrics
      - traefik.http.routers.prometheus.entrypoints=https
      - traefik.http.routers.prometheus.rule=Host(`{{ ansible_fqdn }}`) &&␣
↪PathPrefix(`/metrics`)
      - traefik.http.routers.prometheus.middlewares=management-chain@file
      - traefik.http.routers.prometheus.tls=true
      - traefik.http.routers.prometheus.tls.options=default
      - traefik.http.routers.prometheus.tls.certResolver=acmeTLS
      - traefik.http.routers.prometheus.service=prometheus@internal
    networks:
      - rp
    restart: always
    logging: &default_logging
      driver: "json-file"
      options:
        max-size: "1M"
        max-file: "5"

  # b3lb frontend
  django:
    image: quay.io/ibh/b3lb:2.2.2
    env_file:
      - ./conf.d/b3lb/env
    labels:
      # TENANT: /bigbluebutton/api
      #         /b3lb/t/TENANT/bbb/api
      - traefik.enable=true
      - traefik.http.routers.api.entrypoints=https
      - traefik.http.routers.b3lb-api.rule=(HostRegexp(`{[a-z0-9-]+}.{{ api_base_
↪domain }}`) && PathPrefix(`/bigbluebutton/api/`)) || (Host(`{{ api_base_domain }}`)␣
↪&& PathPrefix(`/b3lb/t/{[a-z0-9-]+}/bbb/api/`))
      - traefik.http.routers.api.middlewares=endpoint-chain@file
      - traefik.http.routers.api.tls=true
      - traefik.http.routers.api.tls.options=default
      - traefik.http.routers.api.tls.certResolver=acmeDNS
      - "traefik.http.routers.api.tls.domains[0].main={{ api_base_domain }}"
```

(continues on next page)

(continued from previous page)

```
      - "traefik.http.routers.api.tls.domains[0].sans=*.{{ api_base_domain }}"
      - traefik.http.routers.api.service=api
      - traefik.http.services.api.loadbalancer.server.port=8000

      # TENANT: /b3lb/t/TENANT/logo
      #         /b3lb/t/TENANT/slide
      - traefik.http.routers.b3lb-assets.entrypoints=https
      - traefik.http.routers.b3lb-assets.rule=Host(`{{ api_base_domain }}`) && Path(`/
↪b3lb/t/{[a-z0-9-]+}/logo`, `/b3lb/t/{[a-z0-9-]+}/slide`)
      - traefik.http.routers.b3lb-assets.middlewares=endpoint-chain@file
      - traefik.http.routers.b3lb-assets.tls=true
      - traefik.http.routers.b3lb-assets.tls.options=default
      - traefik.http.routers.b3lb-assets.tls.certResolver=acmeDNS
      - traefik.http.routers.b3lb-assets.service=b3lb-assets
      - traefik.http.services.b3lb-assets.loadbalancer.server.port=8000
      - "traefik.http.routers.b3lb-assets.tls.domains[0].main={{ api_base_domain }}"
      - "traefik.http.routers.b3lb-assets.tls.domains[0].sans=*.{{ api_base_domain }}"

      # GLOBAL: /b3lb/ping
      # TENANT: /b3lb/ping
      - traefik.http.routers.api-ping.entrypoints=https
      - traefik.http.routers.api-ping.rule=(HostRegexp(`{{ api_base_domain }}`) ||␣
↪HostRegexp(`{tenant:[a-z0-9-]+}.{{ api_base_domain }}`)) && Path(`/b3lb/ping`)
      - traefik.http.routers.api-ping.middlewares=endpoint-chain@file
      - traefik.http.routers.api-ping.tls=true
      - traefik.http.routers.api-ping.tls.options=default
      - traefik.http.routers.api-ping.tls.certResolver=acmeDNS
      - "traefik.http.routers.api-ping.tls.domains[0].main={{ api_base_domain }}"
      - "traefik.http.routers.api-ping.tls.domains[0].sans=*.{{ api_base_domain }}"
      - traefik.http.routers.api-ping.service=api-ping
      - traefik.http.services.api-ping.loadbalancer.server.port=8000

      # TENANT: /b3lb/stats
      # TENANT: /b3lb/metrics
      - traefik.http.routers.stats.entrypoints=https
      - traefik.http.routers.b3lb-stats.rule=(HostRegexp(`{[a-z0-9-]+}.{{ api_base_
↪domain }}`) && Path(`/b3lb/stats`, `/b3lb/metrics`)) || (Host(`{{ api_base_domain }}
↪`) && Path(`/b3lb/t/{[a-z0-9-]+}/stats`, `/b3lb/t/{[a-z0-9-]+}/metrics`))
      - traefik.http.routers.stats.middlewares=endpoint-chain@file
      - traefik.http.routers.stats.tls=true
      - traefik.http.routers.stats.tls.options=default
      - traefik.http.routers.stats.tls.certResolver=acmeDNS
      - traefik.http.routers.stats.service=stats
      - traefik.http.services.stats.loadbalancer.server.port=8000
      - "traefik.http.routers.stats.tls.domains[0].main={{ api_base_domain }}"
      - "traefik.http.routers.stats.tls.domains[0].sans=*.{{ api_base_domain }}"

      # GLOBAL: /admin/ /files/ /b3lb/metrics
      - traefik.http.routers.admin.entrypoints=https
      - traefik.http.routers.b3lb-admin.rule=Host(`{{ api_base_domain }}`) &&␣
↪(PathPrefix(`/admin/`, `/files/`) || Path(`/b3lb/metrics`))
      - traefik.http.routers.admin.middlewares=management-chain@file
      - traefik.http.routers.admin.tls=true
      - traefik.http.routers.admin.tls.options=default
      - traefik.http.routers.admin.tls.certResolver=acmeDNS
      - traefik.http.routers.admin.service=admin
      - traefik.http.services.admin.loadbalancer.server.port=8000
```

(continues on next page)

(continued from previous page)

```
      - "traefik.http.routers.admin.tls.domains[0].main={{ api_base_domain }}"
      - "traefik.http.routers.admin.tls.domains[0].sans=*.{{ api_base_domain }}"
    networks:
      - rp
      - lb
    restart: always
    logging:
      <<: *default_logging

  # static assets: logos, slides and Django admin
  static:
    image: quay.io/ibh/b3lb-static:2.2.2
    labels:
      # Django admin static assets
      - traefik.enable=true
      - traefik.http.routers.static.entrypoints=https
      - traefik.http.routers.static.rule=Host(`{{ api_base_domain }}`) &&␣
↪PathPrefix(`/static/`)
      - traefik.http.routers.static.middlewares=management-chain@file,static-strip
      - traefik.http.routers.static.tls=true
      - traefik.http.routers.static.tls.options=default
      - traefik.http.routers.static.tls.certResolver=acmeDNS
      - traefik.http.middlewares.static-strip.stripprefix.prefixes=/static
      - traefik.http.services.static.loadbalancer.server.port=8001
      - "traefik.http.routers.static.tls.domains[0].main={{ api_base_domain }}"
      - "traefik.http.routers.static.tls.domains[0].sans=*.{{ api_base_domain }}"
    networks:
      - rp
    restart: always
    logging:
      <<: *default_logging

  # celery scheduling
  celery-beat:
    image: quay.io/ibh/b3lb:2.2.2
    command: celery-beat
    env_file:
      - ./conf.d/b3lb/env
    networks:
      - lb
    restart: always
    logging:
      <<: *default_logging

  # celery worker
  celery-tasks:
    image: quay.io/ibh/b3lb:2.2.2
    #
    # ---==] PyPy [==---
    #
    # Consider to change to PyPy if the processing speed is to slow.
    # You need to replace the image and add a reasonable high cgroup
    # memory limit:
    #
    # image: quay.io/ibh/b3lb-pypy:2.2.2
    # mem_limit: 10g
    #
```

(continues on next page)

(continued from previous page)

```
    # ---==] PyPy [==---
    #
    command: celery-tasks
    env_file:
      - ./conf.d/b3lb/env
    networks:
      - lb
    restart: always
    logging:
      <<: *default_logging

  # cache
  redis:
    image: redis:6.0.16-alpine
    # TODO: Adjust max memory!
    # TODO: Set your redis secret!
    command: redis-server --maxmemory 4096mb --maxmemory-policy volatile-lfu --
↪requirepass {{ redis_secret }}
    networks:
      - lb
    restart: always
    logging:
      <<: *default_logging

networks:
  # b3lb internal
  lb:

  # reverse proxy
  rp:
```

conf.d/traefik/config.yml

```
http:
  middlewares:
    # add security related http headers
    # https://doc.traefik.io/traefik/middlewares/headers/#configuration-options
    security-headers:
      headers:
        frameDeny: true
        sslRedirect: true
        browserXssFilter: true
        contentTypeNosniff: true
        forceSTSHeader: true
        stsSeconds: 31536000
        stsIncludeSubdomains: true
        stsPreload: true

    # prevent search enginge indexing
    x-robots-tag:
       headers:
          customResponseHeaders:
             X-Robots-Tag: "noindex, nofollow, noarchive, nosnippet, notranslate,␣
↪noimageindex"
```

(continues on next page)

(continued from previous page)

```
    # list of ip prefixes allowed to access management and metrics
    mgmt-whitelist:
      ipWhiteList:
        sourceRange:
          # TODO: Add your management ip prefixes!
          - 127.0.0.0/8


    # middleware chain used for public endpoints
    endpoint-chain:
      chain:
        middlewares:
        - security-headers
        - x-robots-tag

    # middleware chain used for management endpoints
    management-chain:
      chain:
        middlewares:
        - security-headers
        - x-robots-tag
        - mgmt-whitelist


tls:
  options:
    # TLS settings
    default:
      sniStrict: true
      minVersion: VersionTLS12
      preferServerCipherSuites: true
      cipherSuites:
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
        - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
      curvePreferences:
        - X25519
        - CurveP521
        - CurveP384
```

conf.d/traefik/traefik.yml

```
# disable traefik call home
global:
  checkNewVersion: false
  sendAnonymousUsage: false

# enable traefik dashboard
api:
  dashboard: true
```

(continues on next page)

(continued from previous page)

```yaml
# fix traefik's dashboard privacy issue
# (https://github.com/traefik/traefik/issues/7699)
pilot:
  dashboard: false

# enable traefik ping handler
ping:
  manualRouting: true

# enable traefik prometheus metrics export
metrics:
  prometheus:
    manualRouting: true


# entrypoints for http and https
entryPoints:
  http:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: https
          scheme: https
  https:
    address: ":443"
    http:
      tls:
        options: default

# add docker and file providers
providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    watch: true
    exposedByDefault: false
    # Needs to match the network name created by
    # docker-compose!
    network: b3lb_rp
  file:
    filename: /etc/traefik/config.yml

# frontend certificates
certificatesResolvers:
  # required for wildcard DNS entries
  acmeDNS:
    acme:
      # TODO: Adding an email address is required!
      #email:
      storage: /etc/traefik/acme/acmeDNS.json
      dnsChallenge:
        provider: rfc2136
  #  sufficient for single domain name setups
  acmeTLS:
    acme:
      # TODO: Adding an email address is required!
      #email:
```

(continues on next page)

(continued from previous page)

```
      storage: /etc/traefik/acme/acmeTLS.json
      tlsChallenge: {}

# use default logging
log: {}

# enable access logging only for failed or high latency requests
accessLog:
  filters:
    statusCodes:
      - "400-499"
      - "500-599"
    retryAttempts: true
    minDuration: "500ms"
```

CHAPTER

FIVE

DJANGO ADMIN

You need to do the following initial operations to get your B3LB instance operational.

5.1 Initial fixture

B3LB ships with a initial fixture to configure the celery scheduled tasks:

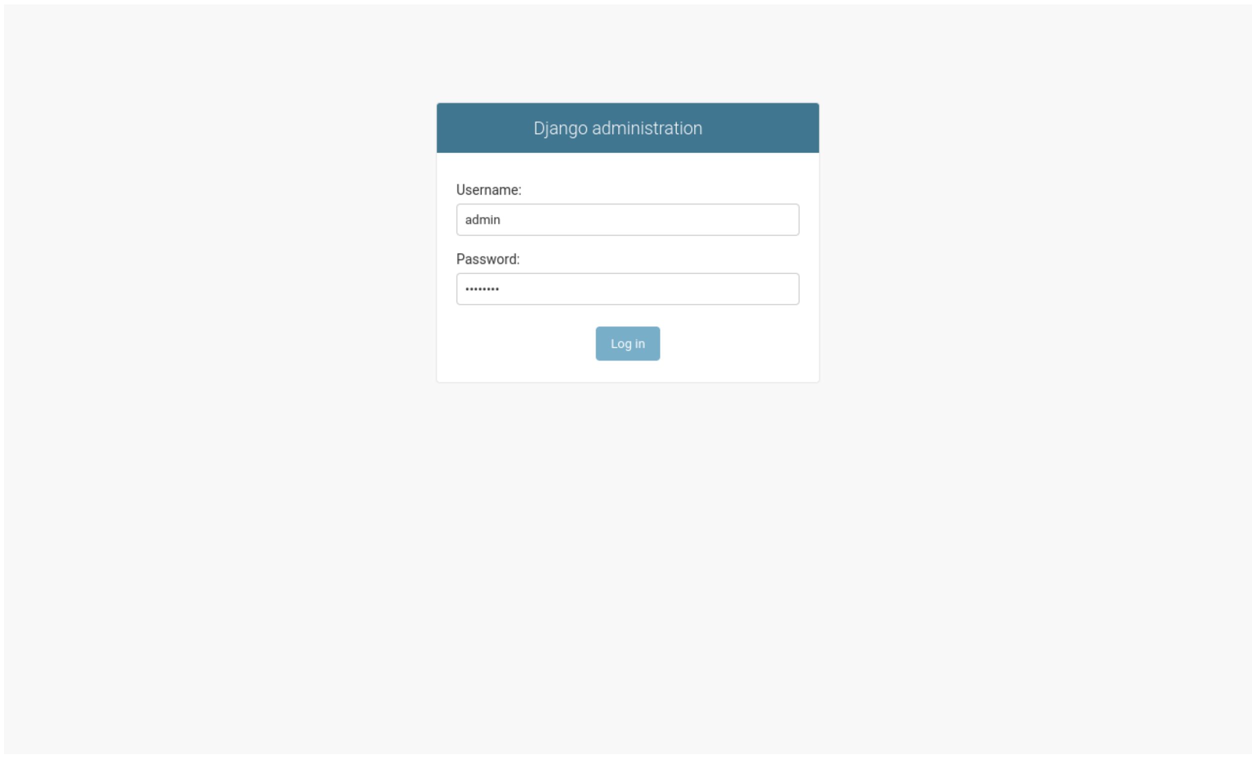
```
$ docker-compose exec django ./manage.py loaddata periodictasks
Installed 6 object(s) from 1 fixture(s)
```

5.2 Create a superuser

A initial Django superuser login needs to be created on the CLI to be able to login to the Django admin pages:

```
$ docker-compose exec django ./manage.py createsuperuser
Username (leave blank to use 'root'): admin
Email address: admin@bbbconf.de
Password:
Password (again):
Superuser created successfully.
```

Login at `https://{{ api_base_domain }}/admin/` using the admin credentials:


Django administration

Username:

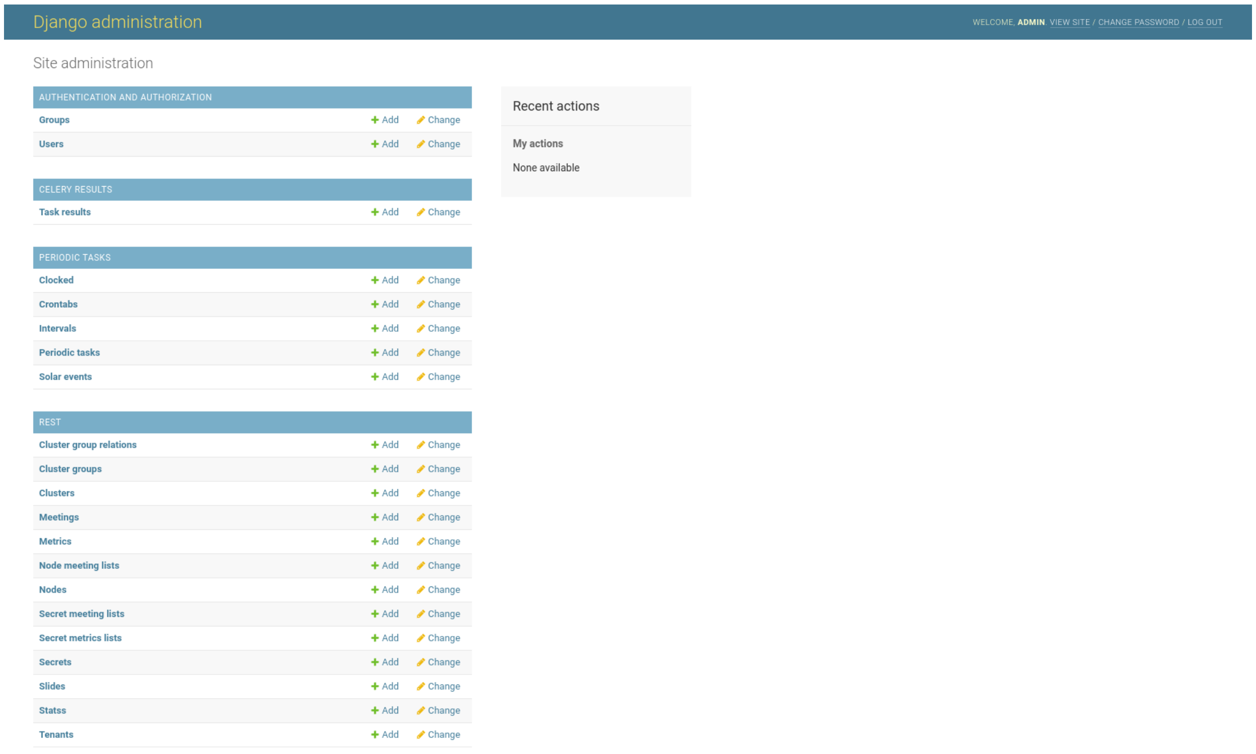
admin

Password:

••••••••

Log in

After logging in you are able to view and edit the Django models:


Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

CELERY RESULTS		
Task results	+ Add	Change

PERIODIC TASKS		
Clocked	+ Add	Change
Crontabs	+ Add	Change
Intervals	+ Add	Change
Periodic tasks	+ Add	Change
Solar events	+ Add	Change

REST		
Cluster group relations	+ Add	Change
Cluster groups	+ Add	Change
Clusters	+ Add	Change
Meetings	+ Add	Change
Metrics	+ Add	Change
Node meeting lists	+ Add	Change
Nodes	+ Add	Change
Secret meeting lists	+ Add	Change
Secret metrics lists	+ Add	Change
Secrets	+ Add	Change
Slides	+ Add	Change
Statss	+ Add	Change
Tenants	+ Add	Change

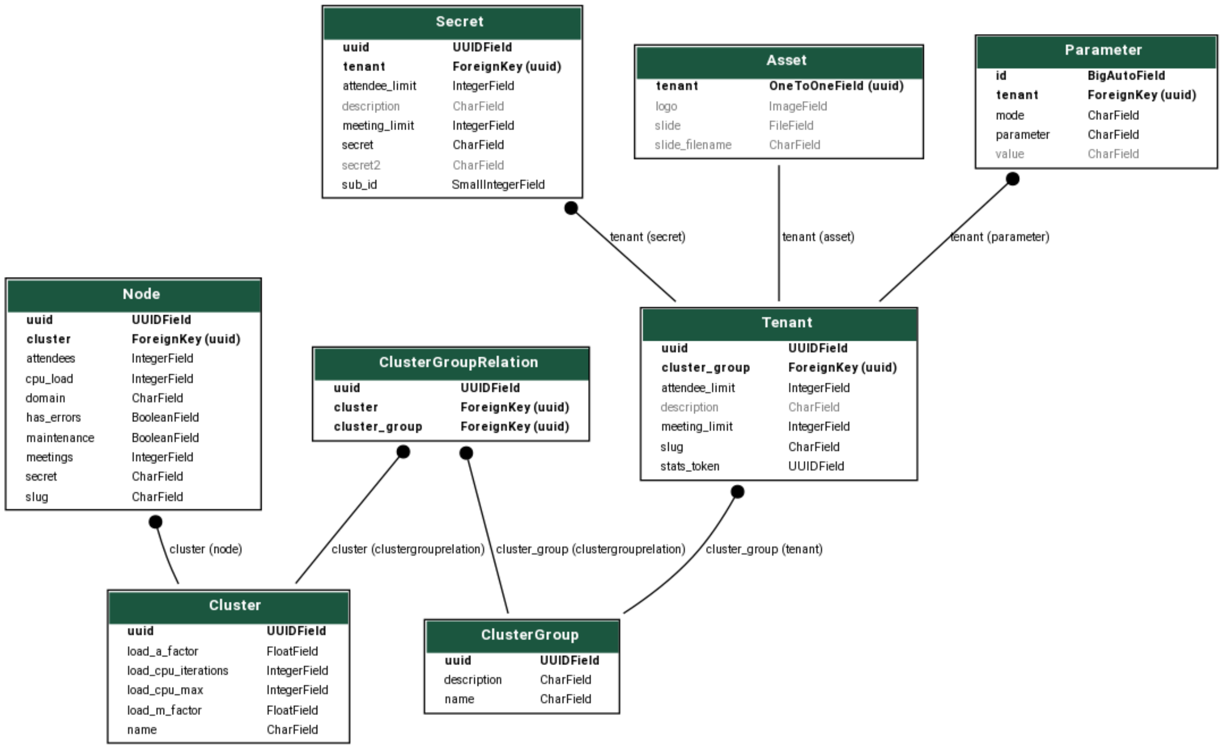
Recent actions

My actions

None available

5.3 Models

With the admin login you can configure the required backend instances. The following schema shows the relation between models used in *B3LB* and to be used by the admin:



The models are used for:

Cluster

A group of BBB *Nodes* with the same load balancing parameters. If running different types of hardware or VM nodes you should consider to put group them into diffent *Clusters* so that the load balancing parameters can be tuned per *Cluster*.

ClusterGroup

A group of *Clusters* which can be used by *Tenants*. This is just a indirection layer if you have many clusters. In small environemnts there might only by a single *ClusterGroup*. *Cluster* can be assigned to multple *ClusterGroups*.

ClusterGroupRelation

Maps a *Cluster* to a *ClusterGroup*.

Node

A single host running a BigBlueButton instance. It is related to a single *Cluster*. The *Node*'s FQDN is build from the *slug* and the *domain* property.

Secret

A BigBlueButton API secret related to a tenant. A tenant might have multple secrets. Attendee and meeting limits are enforced at the secret level and at *Tenant* level - whatever exceeds first.

Asset

A *Tenant* can have an *Asset*. With an *Asset* it is possible to assign a startup slide and a branding logo to a *Tenant*.

Parameter

With *Parameter*s are assigned to a *Tenant* and can be used to set, override or block BBB API *create parameters <https://docs.bigbluebutton.org/dev/api.html#create>*_.

Tenant

Is allowed to use the B3LB's BigBlueButton API. A *Tenant* requires at least one *Secret* with sub_id of `0` before the API can be used. Attendee and meeting limits are enforced over all *Secrets* assigned to the *Tenant*.

CHAPTER

SIX

CLI COMMANDS

The *b3lb* container image provides the following additional django-admin commands:

addnode Creates a BBB node and adds it to a existing cluster.

```
usage: manage.py addnode [-h] --slug SLUG --secret SECRET --cluster CLUSTER [--
↪version] [-v {0,1,2,3}]
                  [--settings SETTINGS] [--pythonpath PYTHONPATH] [--traceback] [--
↪no-color]
                  [--force-color] [--skip-checks]

Add new BBB cluster node.

optional arguments:
  --slug SLUG           hostname
  --secret SECRET       BBB API secret
  --cluster CLUSTER     cluster name
```

addsecrets Generates tenant API sub secrets using random credentials.

```
usage: manage.py addsecrets [-h] --tenant-slug TENANT_SLUG --sub-id SUB_ID [--
↪version] [-v {0,1,2,3}]
                             [--settings SETTINGS] [--pythonpath PYTHONPATH] [--
↪traceback] [--no-color]
                             [--force-color] [--skip-checks]

Add new tenant secret(s).

optional arguments:
  --tenant-slug TENANT_SLUG
                        Slug
  --sub-id SUB_ID       Ids from N-M or single id, with 0 <= N, M < 1000
```

checkslides Synchronizes the slides objects to the existing slide files.

```
usage: manage.py checkslides [-h] [--version] [-v {0,1,2,3}] [--settings SETTINGS]
                              [--pythonpath PYTHONPATH] [--traceback] [--no-color]␣
↪[--force-color]
                              [--skip-checks]

Check slides in slides folder
```

getloadvalues Dumps the BBB node's load values.

```
usage: manage.py getloadvalues [-h] [--version] [-v {0,1,2,3}] [--settings␣
↪SETTINGS]
                               [--pythonpath PYTHONPATH] [--traceback] [--no-color] [--
↪force-color]
                               [--skip-checks]

Get calculated load values of nodes
```

gettenantsecrets Dumps all API secrets of a single tenant.

```
usage: manage.py gettenantsecrets [-h] [--version] [-v {0,1,2,3}] [--settings␣
↪SETTINGS]
                                  [--pythonpath PYTHONPATH] [--traceback] [--no-color] [--
↪force-color]
                                  [--skip-checks]

Get first secret and hostnames of tenants.
```

listalltenantsecrets Dumps API secrets of all tenants.

```
usage: manage.py listalltenantsecrets [-h] [--version] [-v {0,1,2,3}] [--settings␣
↪SETTINGS]
                                      [--pythonpath PYTHONPATH] [--traceback] [--no-color]
                                      [--force-color] [--skip-checks]

List all secrets of all tenants
```

meetingstats Dump statistics.

```
usage: manage.py meetingstats [-h] [--version] [-v {0,1,2,3}] [--settings␣
↪SETTINGS]
                              [--pythonpath PYTHONPATH] [--traceback] [--no-color] [--
↪force-color]
                              [--skip-checks]

Get status information of tenant and meetings
```