# Gemnasium docs Documentation

*Release 1.2.0*

**Gemnasium**

**Apr 25, 2017**

**1 Welcome** **3**

**2 Release Notes** **5**
2.1 1.2.1 - 2017-04-19 . . . 5
2.2 1.2.0 - 2017-04-10 . . . 5
2.3 1.1.3 - 2017-03-21 . . . 5
2.4 1.1.2 - 2017-02-16 . . . 5
2.5 1.1.1 - 2017-02-03 . . . 6
2.6 1.1.0 - 2017-01-31 . . . 6
2.7 1.0.0 - 2017-01-27 . . . 6
2.8 1.0.0-rc1 - 2017-01-16 . . . 6
2.9 1.0.0-beta4 - 2016-12-15 . . . 6
2.10 1.0.0-beta3 - 2016-11-29 . . . 7
2.11 1.0.0-beta2 - 2016-11-18 . . . 7
2.12 1.0.0-beta1 - 2016-10-21 . . . 7

**3 Getting Started** **9**
3.1 What is Gemnasium Enterprise? . . . 9
3.2 First steps . . . 9

**4 Prerequisites** **11**
4.1 Subscriptions . . . 11
4.2 System Requirements . . . 11
4.3 License key . . . 11

**5 Firewall configuration** **13**
5.1 Incoming traffic . . . 13
5.2 Outgoing traffic . . . 13

**6 Run Gemnasium Enterprise** **15**
6.1 Download Gemnasium Enterprise . . . 15
6.2 Preparing volumes . . . 15
6.3 Configuring SSL . . . 15
6.4 Running without SSL . . . 16
6.5 SELinux . . . 17
6.6 Volumes . . . 17
6.7 Logging . . . 18
6.8 Obtaining a shell . . . 18

**7 SMTP setup** **19**
7.1 Configuration . . . 19

**8 Environment Variables** **21**
8.1 Variables persistence . . . 21
8.2 Variables List . . . 21

**9 Upgrade Gemnasium Enterprise** **23**
9.1 Using latest version . . . 23
9.2 Using nightly version . . . 23
9.3 Using a tagged version . . . 23

**10 Troubleshooting** **25**
10.1 Known issues . . . 25

**11 Data Backup** **27**
11.1 Snapshots . . . 27
11.2 Using docker . . . 27

**12 Proxy configuration** **29**
12.1 NO_PROXY configuration . . . 29

**13 Self-Signed Certificates** **31**
13.1 Installing a CA CERT . . . 31
13.2 Getting a valid certificate . . . 31

**14 Let's Encrypt Certificates** **33**
14.1 Integrated Let's Encrypt client . . . 33
14.2 Custom process . . . 33

**15 GitHub** **35**
15.1 GitHub.com . . . 35
15.2 GitHub Enterprise . . . 36

**16 GitLab** **37**
16.1 GitLab.com . . . 37
16.2 GitLab CE/EE . . . 38

**17 Bitbucket Cloud** **39**
17.1 Adding OAuth consumers on Bitbucket.org . . . 39
17.2 Configure Gemnasium Enterprise to use Bitbucket.org . . . 39
17.3 Advanced configuration . . . 40

**18 Bitbucket Server** **41**
18.1 Adding OAuth consumers on Bitbucket Server . . . 41
18.2 Configure Gemnasium Enterprise to use Bitbucket Server . . . 46
18.3 Adding project webhooks . . . 46

**19 Slack** **51**

Contents:

CHAPTER 1

# Welcome

Welcome to the Gemnasium Enterprise 1.2 documentation, where you can find information and guides to help you with Gemnasium Enterprise and start exploring its features.

Use the left navigation bar to browse the documentation, the Search bar in the top-left to look for something specific, or the links below to access some highlights.

**Note:** Gemnasium Enterprise Edition will be referred as "GEE" in this documentation

CHAPTER 2

# Release Notes

## 1.2.1 - 2017-04-19

- [FEATURE] Add *Let's Encrypt Certificates* support
- [FEATURE] Support Python "compatible release" operator (~=) as specified in PEP 440
- [BUG] Invitations to team were sent again if role was updated

## 1.2.0 - 2017-04-10

- [FEATURE] Add package pages
- [FEATURE] Add Packages changelogs
- [FEATURE] Project page revamped
- [FEATURE] Add Yarn support
- [DOC] Fix documentation for integrations
- [BUG] Fix a bug with PHP dependencies using the caret operator
- [BUG] The button "Refresh repositories" was emptying the list with bitbucket.org

## 1.1.3 - 2017-03-21

- Gemnasium Enterprise is now also available on Quay.io

## 1.1.2 - 2017-02-16

- [BUG] Minor bug and fixes
- [BUG] UI pages now expose an error to the user if the backend is not available.
- [FEATURE] New button to copy the notification channels of a project to all the projects of the team
- [FEATURE] New `MAILER_EMAIL_FROM` env var to specify the sender of GEE email notifications
- [DOC] Added documentation for CA certs used in integrations

- [DOC] Added documentation for users behind proxies

## 1.1.1 - 2017-02-03

- [BUG] Fix webhook handler. The service in charge of receiving and triggering a project sync was returning a 200 and dropping the event in some cases.

## 1.1.0 - 2017-01-31

- [FEATURE] Add Bitbucket Server support
- [BUG] Weekly digests are now sent on Monday mornings, 8am, instead of Sunday at midnight
- [BUG] Adding an empty project from GitHub/Gitlab/bitbucket.org was causing the webhook registration to fail. The project bootstrapping was considered as finished, and the project was not synced after the first commit.

Note: We have switched to Webpack 2 for assets bundling, this is transparent for users.

## 1.0.0 - 2017-01-27

Same as 1.0.0-rc1.

## 1.0.0-rc1 - 2017-01-16

This is the last pre-release before 1.0.0, if no bug is found.

- [FEATURE] Add Bitbucket.org (Bitbucket Cloud) Support
- [FEATURE] Add project logs with realtime update
- [FEATURE] Improve project notification channels configuration
- [FEATURE] Allow to edit existing project notification channel
- [FEATURE] Improve user notifications configuration
- [BUG] Fix various UI bugs
- [BUG] Some PHP packages were not fully synced

## 1.0.0-beta4 - 2016-12-15

- [FEATURE] "New package release" notifications via Slack and email
- [BUG] Fix file upload form when adding unsupported file
- [BUG] Fix left menu bar behavior on small devices layout
- [BUG] Fix oauth signup error handling

## 1.0.0-beta3 - 2016-11-29

- [FEATURE] GitLab Support
- [FEATURE] New notifications in the UI

Known issues:

- [BUG][GITLAB] Symlinks on dependency files are not followed
- [BUG][GITLAB] Dependency files greater than 2MB are ignored
- [BUG] Can't sign-in using an oauth account if the same email is already used

## 1.0.0-beta2 - 2016-11-18

- [FEATURE] Display commits in project page
- [FEATURE] Internal logging (live feeds will be available in beta3)
- [BUG] Fix a security issue when adding a project to a team. The tokens of the team owner were used instead of the user's.
- [BUG] Fix display issues in Firefox
- [BUG] Fix UI Cache issues
- [BUG] Offline projects color was not updated when pushing new dependency files
- [BUG] Sync was failing when commit already existed
- [BUG] Fix a bug preventing to upload new files in Offline projects

Known issues:

- [FEATURE] Gitlab support is delayed to beta3
- [BUG] Can't sign-in using an oauth account if the same email is already used

## 1.0.0-beta1 - 2016-10-21

- First private beta
- GitHub.com and GitHub Enterprise support
- Slack integration for notifications

CHAPTER 3

# Getting Started

## What is Gemnasium Enterprise?

Gemnasium Enterprise Edition (GEE) is the self-hosted version of Gemnasium, designed to run on your own servers, inside your network. A private license key is required to use the software, otherwise no data will be synced with gemnasium.com (making your instance unusable).

To get a valid license key, please contact our support.

## First steps

Gemnasium Enterprise is designed as a collaborative tool. Each project must be added inside a team context, that's why the first thing to do is to create your team. The name must be unique in the Gemnasium Enterprise instance.

### Add your collegues

Go to the "Team" section of the main menu, and press the "+ Add member" button to invite co-workers. The role of the invited user will grant him/her access to your team:

- "Admin" users will be able to manage project and users
- "Basic" users will be able to access projects only

### Add projects

The first shortcut in the main menu "+ Add project" will allow you to add new projects to a team. Once the team is selected, a source must be chosen. By default, only "offline" projects are available, because your instance doesn't know anything about external sources.

Offline projects will allow to upload dependencies files (Gemfile, Gemfile.lock, package.json,...) directly to a project. This kind of project is called "offline", because Gemnasium can't synchronize the files with an external source.

To add projects from github.com, or GitHub Enterprise, an external source must be created. Please refer to the corresponding pages in the "INSTALLATION AND CONFIGURATION" section of this documentation. More source types will be available in the next versions of Gemnasium Enterprise.

CHAPTER 4

# Prerequisites

## Subscriptions

- As Gemnasium Enterprise is provided as a docker image, you must have a Docker Hub or Quay account to download it.
- A Gemnasium Enterprise license is required. If do not have your license yet, please contact our support.

## System Requirements

### Hardware

- Physical or virtual system, or an instance running on a public or private IaaS.
- 1 vCPU
- Minimum of 2GB RAM
- Minimum 20GB hard disk space

### Software

- OS: RedHat (RHEL, Atomic Host, Centos & Fedora flavors), Debian Stable. Any OS running docker should work but is not officially supported.
- Docker >= 1.9.1

## License key

As you will see, in all the `docker run` commands, there is a `-e LICENSE_KEY=YOUR_OWN_LICENSE_KEY \`. You need to replace `YOUR_OWN_LICENSE_KEY` with the license we gave you. If you don't have one, please get in touch and we'll get that sorted.

CHAPTER 5

# Firewall configuration

Gemnasium Enterprise is designed to run behind your firewalls, inside your network. It should be completely isolated from the outside, especially from incoming connections.

## Incoming traffic

Gemnasium only exposes two ports:

| Port | Usage | Protocol |
|---|---|---|
| 80 | clear connection | http |
| 443 | secure connection | https |

It's your responsibility to configure your network and firewall to restrict access to these ports. By default, Gemnasium exposes port 80 only to redirect to port 443. Custom ports might be used, refer to *Configuring SSL* if needed.

## Outgoing traffic

While Gemnasium Enterprise should be completely isolated from the outside for incoming connections, some outgoing traffic is necessary for normal operation. The following ports must be open on your firewall for outgoing traffic:

| Address | Port | Protocol | Usage |
|---|---|---|---|
| sync.gemnasium.com | 443 | tcp | Sync with Gemnasium main DB |
| index.docker.io | 443 | tcp | Pull updates of gemnasium/enterprise image, if used. |
| quay.io | 443 | tcp | Quay docker repository, if used. |

### What data is sent to sync.gemnasium.com?

Your content is private, and will remain private. But synchronizing all packages, versions, changelogs, advisories, etc. with Gemnasium's main DB would require a huge amount of storage and a lot of bandwidth.

To avoid this situation, your instance of Gemnasium Enterprise periodically sends a list of the packages (dependencies) used in your projects to https://sync.gemnasium.com In return, Gemnasium syncer provides all the metadata corresponding to this request, including the security advisories. Gemnasium keeps the following information in private log entries:

- Timestamp of the current sync request
- Customer (based on license key)

- Gemnasium version used
- Number of packages per type (gems, npms, ...)

**Note:** Private dependencies are completely ignored by the syncer.

Advisories can be created at any time on Gemnasium.com; that's why your instance synchronizes hourly. If one of your projects is using a dependency affected by a security issue, you will be notified by your instance within the hour.

**Note:** Gemnasium is using data (advisories) from security companies (partnerships only). Your data will never be submitted directly to these companies, but Gemnasium may share anonymized statistical data.

CHAPTER 6

# Run Gemnasium Enterprise

Gemnasium Enterprise is shipped as a single, all-included docker image.

## Download Gemnasium Enterprise

In order to be able to download the Gemnasium Enterprise docker image, you must have one of these accounts:

- Docker Hub
- Quay

Send us the name of the account used and we will share the image with that user.

**Note:** This documentation is based on Docker hub image URLs, ie: `gemnasium/enterprise`. If Quay is being used instead, prefix the image name to have `quay.io/gemnasium/enterprise`.

## Preparing volumes

Persistent volumes are needed to store Gemnasium Enterprise data. The easiest way to get started, is to create local volumes on your server, but it can be any kind of volume supported by the docker engine.

**See also:**

Please refer to Docker Volumes for more information: https://docs.docker.com/engine/tutorials/dockervolumes/

To create local volumes, on you server:

```
docker volume create --name gemnasium-data
docker volume create --name gemnasium-logs
```

## Configuring SSL

A valid certificate must be provided to run Gemnasium Enterprise with the integrated SSL webserver. If you don't have a valid certificate available, you can obtain one from Let's Encrypt for free. Please refer to the *Let's Encrypt Certificates* section. If you don't need Gemnasium Enterprise to serve content on https directly, go directly to the section: *Running without SSL*.

The certificate files **must** be named after the server name.

Example: for *gemnasium.example.com*, the certificate files **must** be named:

- `gemnasium.example.com.cert.pem` for the certificate
- `gemnasium.example.com.key.pem` for its private key

Gemnasium will look for 2 files with the `.cert.pem` and `.key.pem` suffix.

If the certificate has an intermediate chain, it must concatenated after the server certificate:

```
cat server.cert.pem ca-chain.cert.pem > gemnasium.example.com.cert.pem
```

The 2 files **must** be available in `/etc/gemnasium/ssl`, inside the container.

```
docker run --detach  \
  --name gemnasium \
  --restart always \
  -v /host/path/to/ssl/:/etc/gemnasium/ssl \
  -p 80:80 -p 443:443 \
  -e LICENSE_KEY=YOUR_OWN_LICENSE_KEY \
  -v gemnasium-data:/var/opt/gemnasium/ \
  -v gemnasium-logs:/var/log/ \
  -v /var/run/docker.sock:/var/run/docker.sock \
  gemnasium/enterprise:latest
```

**Note:** Gemnasium needs the docker socket to be mounted only if the Reports feature is being used. If not, the line `-v /var/run/docker.sock:/var/run/docker.sock` can be safely removed.

This will pull and start Gemnasium Enterprise. Your instance will be available at https://gemnasium.example.com after a few seconds.

If you need to use a different port for https than 443, use the `EXTERNAL_URL` env var to specify the full URL of your Gemnasium Enterprise server, including the port used:

```
docker run --detach  \
  --name gemnasium \
  --restart always \
  -v /host/path/to/ssl/:/etc/gemnasium/ssl \
  -p 80:80 -p 8443:443 \
  -e LICENSE_KEY=YOUR_OWN_LICENSE_KEY \
  -e EXTERNAL_URL=https://gemnasium.example.com:8443/ \
  -v gemnasium-data:/var/opt/gemnasium/ \
  -v gemnasium-logs:/var/log/ \
  -v /var/run/docker.sock:/var/run/docker.sock \
  gemnasium/enterprise:latest
```

and start browsing https://gemnasium.example.com:8443/

## Running without SSL

**Warning:** We strongly discourage running Gemnasium Enterprise without any SSL termination. This section is present if you already have SSL terminations, like secured reverse-proxies, ssl appliances, etc.

Run the image:

```
docker run --detach  \
  --name gemnasium \
  --restart always \
  -e REDIRECT_HTTP_TO_HTTPS=false \
  -e EXTERNAL_URL=http://gemnasium.example.com/ \
  -p 80:80 \
  -e LICENSE_KEY=YOUR_OWN_LICENSE_KEY \
  -v gemnasium-data:/var/opt/gemnasium/ \
  -v gemnasium-logs:/var/log/ \
  -v /var/run/docker.sock:/var/run/docker.sock \
  gemnasium/enterprise:latest
```

**Note:** The environment variable `REDIRECT_HTTP_TO_HTTPS` is *true* by default, and must be `false` in this case.

The service is available after a few seconds on the port 80 of your server. Use the `EXTERNAL_URL` variable to specify the full URL of your Gemnasium Enterprise server, including the port if necessary.

## SELinux

Gemnasium Enterprise can't be run directly on SELinux servers, because:

1. The volumes will be readonly by default
2. The docker socket will be restricted to the host

Use this command instead:

```
docker run --detach  \
  --name gemnasium \
  --restart always \
  -v /host/path/to/ssl/:/etc/gemnasium/ssl \
  -p 80:80 -p 443:443 \
  -e LICENSE_KEY=YOUR_OWN_LICENSE_KEY \
  -v gemnasium-data:/var/opt/gemnasium/:Z \
  -v gemnasium-logs:/var/log/:Z \
  -v /var/run/docker.sock:/var/run/docker.sock:Z \
  gemnasium/enterprise:latest
```

This will label the content inside the container with the exact MCS label that the container will run with, basically it runs `chcon -Rt svirt_sandbox_file_t -l s0:c1,c2 /var/db` where `s0:c1,c2` differs for each container.

**See also:**

More info: http://www.projectatomic.io/blog/2015/06/using-volumes-with-docker-can-cause-problems-with-selinux/

Please refer to this project to install the proper SELinux module to fix the second point.

## Volumes

Gemnasium is storing data in two folders, which should be mounted as volumes

| Local location | Location in container | Usage |
|---|---|---|
| gemnasium-data (volume) | /var/opt/gemnasium | Gemnasium data |
| gemnasium-logs (volume) | /var/log | Gemnasium logs |

Gemnasium data is composed mostly of the PostgreSQL database files, but also nsq data, etc. These files must be backed up, refer to the *Data Backup*. section.

The `/var/log` contains the OS logs, and everything dedicated to gemnasium in `/var/log/gemnasium`.

Finally, as explained in the *Configuring SSL* section, your certificate and key must be available in the `/etc/gemnasium/ssl` folder.

# Logging

By default, all logs will be sent to the standard output of the container (`stdout`), along with files in `/var/log`. This makes it easier to troubleshoot if needed.

## Graylog

Gemnasium Enterprise can be configured to log to a distant Graylog server. To enable this feature, use the following environment variables:

| Env variables | Usage |
|---|---|
| GRAYLOG_SERVICE_HOST | Graylog input hostname/ip |
| GRAYLOG_SERVICE_PORT | Graylog input port |

Example:

```
docker run --detach  \
  --name gemnasium \
  --restart always \
  -v /host/path/to/ssl/:/etc/gemnasium/ssl \
  -p 80:80 -p 443:443 \
  -v gemnasium-data:/var/opt/gemnasium/ \
  -v gemnasium-logs:/var/log/ \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -e GRAYLOG_SERVICE_HOST=logs.example.log
  -e GRAYLOG_SERVICE_PORT=1515
  gemnasium/enterprise:latest
```

Both variables must be set to activate the GELF output.

# Obtaining a shell

The docker image doesn't have a SSH server, because docker provides everything needed to get a shell console inside the container:

```
docker exec -it gemnasium bash
```

will create a new bash session, with the root user.

> **Warning:** With great power comes great responsibility: as root, you can damage files inside the container, including your persisted data.

CHAPTER 7

# SMTP setup

In order to send notifications (team invitations, digests, etc.), Gemnasium Enterprise needs a SMTP server. There is no such server included in the docker image, so an external SMTP server should be used. Note that SMTP connectivity is not fully required, just recommended.

## Configuration

SMTP can be configured by passing environment variable to the docker container:

| Env variables | Usage |
|---|---|
| SMTP_SERVICE_HOST | Server host address |
| SMTP_SERVICE_PORT | Server port |
| SMTP_USER_NAME | Username |
| SMTP_PASSWORD | Password |
| SMTP_INSECURE | Skip SSL verification |

The SMTP password can be plain auth, or a secret (CRAM-MD5 auth). These variables are all optional.

SSL and TLS are supported, and will be automatically used if the port (`SMTP_SERVICE_PORT`) is `465` or `587`.

The sender for email notifications will be by default *app@gemnasium.com* which can be an issue with your smtp server. It can be updated by passing the `MAILER_EMAIL_FROM` environment var to your Gemnasium Enterprise container.

CHAPTER 8

# Environment Variables

Gemnasium Enterprise is entirely configured using environment variables.

## Variables persistence

If you don't want to specify by hand all environment variables in docker run, you can create a file including all the variables to be set: Compose expects each line in an env file to be in VAR=VAL format. Lines beginning with # (i.e. comments) are ignored, as are blank lines.

To use the environment file, add the `--env-file=[file name]` to your `docker run` command line.

With docker-compose, the name should be named `.env` in the same directory as your `docker-compose.yaml` file, and will be loaded automatically. Remember to specify the env variables expected in the file, otherwise docker-compose will simply ignore them.

## Variables List

| Env variables | Usage | Required | Default |
|---|---|---|---|
| Main | | | |
| `LICENSE_KEY` | Your GEE private license key | required | |
| `EXTERNAL_URL` | Your GEE full URL | required | |
| Logging | | | |
| `GRAYLOG_SERVICE_HOST` | Graylog input hostname/ip | optional | |
| `GRAYLOG_SERVICE_PORT` | Graylog input port | if `GRAYLOG_SERVICE_HOST` is set | |
| SMTP | | | |
| `SMTP_SERVICE_HOST` | SMTP server host | optional | |
| `SMTP_SERVICE_PORT` | SMTP server port | optional | 25 |
| `SMTP_USER_NAME` | SMTP user | optional | |
| `SMTP_PASSWORD` | SMTP password (plain auth) or secret (CRAM-MD5 auth) | optional | |
| `SMTP_INSECURE` | SMTP skip SSL verification | optional | false |
| `MAILER_EMAIL_FROM` | Email notifications sender | optional | app@gemnasium.com |

CHAPTER 9

# Upgrade Gemnasium Enterprise

While the data about packages, versions, etc. is automatically updated, and stored in your local database, Gemnasium Enterprise won't upgrade itself automatically.

## Using latest version

To upgrade Gemnasium Enterprise to a new version:

1- Pull the new image:

```
docker pull gemnasium/enterprise:latest
```

2- Stop and remove the container:

```
docker rm -f gemnasium
```

3- Run the image again (see *Preparing volumes*.). Gemnasium Enterprise will update your data automatically, if necessary.

```
docker run [...]
```

## Using nightly version

Every night, a new build of Gemnasium Enterprise will be generated, as a `gemnasium/enterprise:nightly` image. This image is intended for debugging only, and should not be used for production.

## Using a tagged version

The available tags are listed here: https://hub.docker.com/r/gemnasium/enterprise/tags/

It is recommended to always use the `latest` tag.

CHAPTER 10

# Troubleshooting

Read container logs:

```
docker logs -f gemnasium
```

Enter running container:

```
docker exec -it gemnasium /bin/bash
```

## Known issues

Gemnasium Enterprise is using `setcap` to allow our process running on port 80 and 443 (unless SSL is disabled via `REDIRECT_HTTP_TO_HTTPS` to false). Some kernels don't support capacities operations inside containers, especially when AUFS is being used. To avoid an error while running Gemnasium Enterprise, the api server will fallback to use a setuid bit on the server, meaning in the case the service is running as root inside the container. While this is not a security issue for your host, it means the api has full control inside the container, including reading passwords and tokens.

If you are unsure your system is affected by this issue, check the logs of the api service in `/var/log/gemnasium/api/current`. If `setcap` is failing, the message `Warning: setcap not available,falling back to setuid` will be displayed at the top of the log file.

If you want to avoid this issue, you can bind your own ports, higher then 1024, using the env vars `GEMNASIUM_API_PORT_8080_TCP_PORT` `GEMNASIUM_API_SSL_PORT_8443_TCP_PORT`. If they are both above 1024, no setcap or setuid method will be used, and the webserver will run as a limited-rights user.

CHAPTER 11

# Data Backup

Gemnasium Enterprise will store its data in a persistent volume (cf. *Volumes*.).

It is YOUR responsibility to backup this volume. Gemnasium Enterprise has no capacities of backing up data.

## Snapshots

Disk snapshots are probably the best option to backup your data. All the data stored in the persistent volume will saved at once, and can be restored at anytime.

If a restore is needed, remember to stop the container before restoring anything:

```
docker stop gemnasium
```

and remove it completely:

```
docker rm gemnasium
```

Once the data is restored, run again the image: *Preparing volumes*

## Using docker

As explained in the "Manage data in containers" docker page, docker may be used to create a tarball:

> docker run –rm –volumes-from gemnasium -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /var/opt/gemnasium

It is highly recommanded to stop the container before doing this operation.

### Restoring data with docker

With your backup in the local directly (pwd), untar your archive to restore the data in the `gemnasium-data` volume:

```
$ docker run --rm -v $(pwd):/backup -v gemnasium-data:/var/opt/gemnasium/ gemnasium/
↪enterprise bash -c "cd /var/opt/gemnasium && tar xvf /backup/backup.tar --strip 1"
```

Once the data is restored, run again the image: *Preparing volumes*

CHAPTER 12

# Proxy configuration

Gemnasium Enterprise is designed to run behind your firewalls, inside your network. If your network is using a proxy to access http or https resources, you may need to adapt GEE configuration.

By default, docker will use a default network bridge, so the network stack is shared with the host where docker is running.

The proxy configuration for Docker is beyond the scope of this documentation, please refer to Docker and your operating system documentation if needed.

Gemnasium Enterprise is using the posix environment vars `http_proxy` (/`HTTP_PROXY`) and `no_proxy` (/`NO_PROXY`) directly. Set these variables according to your network configuration. All the services inside the container will be aware of these two variables.

## NO_PROXY configuration

As Gemnasium Enterprise is composed of several services running inside the container, the communication between the services might be affected by a proxy configuration on your network.

Theses hosts must be added to your current `no_proxy` var inside the container:

- Your Gemnasium Enterprise full URL
- `api`
- `gemnasium-api`
- `gemnasium-auth`
- `gemnasium-badges`
- `gemnasium-billing`
- `gemnasium-notifier`
- `gemnasium-repo-syncer`
- `nsqlookupd`
- `nsqd`
- `postgresql`

Theses hostnames are added to `/etc/hosts` when the container is starting, so it's important that a `curl gemnasium-api` is working inside the container.

```
root@60bdf6bb1ab5:/# curl gemnasium-api
<a href="http://docs.gemnasium.apiary.io">Moved Permanently</a>.
```

Remember that you can also pass an environment file to your docker container with `--env-file=[file name]`, so this file should contain a complete `no_proxy` definition.

CHAPTER 13

# Self-Signed Certificates

Gemnasium Enterprise doesn't require a certificate signed by a (well-)known authority. As with any other service running on your network, it's not recommanded to use such certificate, unless a CA certificate is being used to sign it.

Gemnaasium Enterprise will fail to contact your local servers if they are using such certificate, unless the right CA is available in the container.

**Note:** Only valid (ie: signed by a known authority) certificates will work with Gemnasium Enterprise.

## Installing a CA CERT

Gemnasium Enterprise is using a Debian Stable as the operating system. To make sure all services inside the container are using the right certificates, they must be installed into `/etc/ssl/certs/`. Do not mount a folder on this path directly, as you may hide the current certificates already present (Debian default ones). Each certificate can actually be mounted:

```
docker run [...] -v [ca-cert file path]:/etc/ssl/certs/[ca-cert file path]:ro␣
↪gemnasium/enterprise
```

**Note:** If you have more than one CA certificate, replicate this parameter for each.

## Getting a valid certificate

If you don't have a valid authority to sign your certificates, you may want to use Let's Encrypt. They are delivering free certificates.

CHAPTER 14

# Let's Encrypt Certificates

Let's Encrypt is a free, automated, and open Certificate Authority (CA). You can obtain a valid certificate for your Gemnasium Enterprise instance. There are two different ways to configure your instance to use Let's Encrypt: Using the integrated client, or using a custom process.

**See also:**

Let's Encrypt documentation online for detailled information

## Integrated Let's Encrypt client

Gemnasium Entreprise features a Let's Encrypt client with:

- Automatic renewal (one week before expiration)
- TLS-SNI domain validation

The generated certificate will be saved in `/etc/gemnasium/ssl` along with its private key. If you want to cache/persist these files, mount a volume into your container for this path.

### Limitations

By default, the client proves control by answering an HTTPS-based challenge. This requires the host name to resolve to the IP address of a (single) computer running Gemnasium Enterprise. Typically, the challenge won't work on a shared SSL LoadBalancer with SNI, because let's encrypt will use server names like `*.acme.invalid`.

This also means you will probably not be able to get a certificate for your local machine (unless it's exposed directly to the internet).

## Custom process

EFF is providing a full-featured acme client:

https://certbot.eff.org/

Getting the certificate, renew it, etc. must be done outside the Gemnasium container, otherwise your changes will be lost during next Gemnasium update deployment. The generated certificate and key should be installed (ie: mounted into the docker container) directly in `/etc/gemnasium/ssl` as explained in the *Configuring SSL* section.

CHAPTER 15

# GitHub

## GitHub.com

Before being able to add projects from github.com, Gemnasium Enterprise needs to be configured to access it.

This is a two steps process: firs you need to create OAuth applications on GitHub and then configure Gemnasium Enterprise to use them.

### 1. Adding OAuth applications on GitHub

Gemnasium Enterprise needs **two different OAuth applications**. One to enable "Login with GitHub" and one to synchronize projects.

To do that:

- go on https://github.com/settings/developers (or alternatively, add the application to an organization: https://github.com/organizations/{[}your_org{]}/settings/applications)
- click on the "Register a new application" and fill the form:
  - **Name:** "Gemnasium Enterprise Login"
  - **Homepage URL:** your Gemnasium Enterprise base URL (example: *https://gemnasium.example.com/*)
  - **Authorization callback URL:** `{GEMNASIUM_INSTANCE_URL}/auth/auth/github.com/login/callback` where you replace `{GEMNASIUM_INSTANCE_URL}` with the url of your Gemnasium Enterprise instance (example: *https://gemnasium.example.com/auth/auth/github.com/login/callback*)
- Click on the "Register application" button.

You will need the "Client ID" and "Client Secret" you see on the confirmation page for the next section. You can keep that tab open and open a new tab to https://github.com/settings/developers to create the second application.

To create the second application required:

- go on https://github.com/settings/developers
- click on the "Register a new application" and fill the form:
  - **Name:** "Gemnasium Enterprise Sync"
  - **Homepage URL:** your Gemnasium Enterprise base URL (example: *https://gemnasium.example.com/*)
  - **Authorization callback URL:** `{GEMNASIUM_INSTANCE_URL}/auth/auth/github.com/sync/callback` where you replace `{GEMNASIUM_INSTANCE_URL}` with the url of your Gemnasium Enterprise instance (example: *https://gemnasium.example.com/auth/auth/github.com/sync/callback*)

- Click on the "Register application" button.

### 2. Configure Gemnasium Enterprise to use GitHub

A convenient script is provided to configure your instance:

```
docker exec -it gemnasium configure
```

Select "GitHub.com", and then fill the corresponding fields with the values from the applications created above. Be careful not to confuse **Sync** and **Login** applications!

Your Gemnasium Enterprise users are now able to login using their GitHub account. A new source named "GitHub" is also available on the "Add Project" screen.

## GitHub Enterprise

GitHub Enterprise is no different than GitHub.com, the steps are the same.

In step *1. Adding OAuth applications on GitHub*, just replace `github.com` with your private GitHub Enterprise instance host (example: *https://gemnasium.example.com/auth/auth/private-github.example.com/login/callback*).

In step *2. Configure Gemnasium Enterprise to use GitHub*, make sure to select "GitHub Enterprise" instead of "GitHub.com". The script will ask for your GitHub Enterprise instance URL, and to name it.

Several GitHub Enterprise instances can be configured, just name them accordingly to avoid confusion.

### Requirements

Gemnasium Enterprise has been tested against GitHub Enterprise >=2.7.X. If your version is older than 2.7 series, please contact our support.

CHAPTER 16

# GitLab

## GitLab.com

Before being able to add projects from gitlab.com, Gemnasium Enterprise needs to be configured to access it.

This is a two steps process: firs you need to create an OAuth application on GitLab and then configure Gemnasium Enterprise to use that application.

### 1. Adding the OAuth application on GitLab

- go on https://gitlab.com/profile/applications
- fill the form "Add new application":
  - **Name:** "Gemnasium Enterprise"
  - **Redirect URI:** `{GEMNASIUM_INSTANCE_URL}/auth/auth/gitlab.com/sync/callback` where you replace `{GEMNASIUM_INSTANCE_URL}` with the url of your Gemnasium Enterprise instance (example: *https://gemnasium.example.com/auth/auth/gitlab.com/sync/callback*)
  - **Scopes:** api
- Click on the "Save application" button.

You will need the "Application Id" and the "Secret" you see on the confirmation page for the next section.

### 2. Configure Gemnasium Enterprise to use GitLab

#### A. Automatic configuration

A convenient script is provided to add both Sign In/up and project synchronization with Gitlab at once. If you don't want both features, check *B. Advanced configuration*.

```
docker exec -it gemnasium configure
```

Select "GitLab.com", and then fill the corresponding fields with the values from the application created above.

Your Gemnasium Enterprise users are now able to login using their GitLab account. A new source named "GitLab" is also available on the "Add Project" screen.

### B. Advanced configuration

The automatic configuration described above enables both Sign In/Up and project synchronization with Gitlab. This advanced configuration makes it possible to restrict the integration to one of these two features.

To enable Gitlab Sign In/Up only, run these commands:

```
docker exec -it gmenasium auth providers create gitlab.com gitlab 2.0 https://gitlab.
↪com/oauth/authorize https://gitlab.com/oauth/token
docker exec -it gemnasium auth clients create gitlab.com login {OAUTH_APPLICATION_ID}
↪{OAUTH_APPLICATION_SECRET} api
```

To enable Gitlab Synchronization only, run these commands:

```
docker exec -it gmenasium auth providers create gitlab.com gitlab 2.0 https://gitlab.
↪com/oauth/authorize https://gitlab.com/oauth/token
docker exec -it gemnasium auth clients create gitlab.com sync {OAUTH_APPLICATION_ID}
↪{OAUTH_APPLICATION_SECRET} api
docker exec -it gemnasium repo-syncer sources create gitlab.com "Gitlab" gitlab https:
↪//api.gitlab.com/
```

`{OAUTH_APPLICATION_ID}` and `{OAUTH_APPLICATION_SECRET}` must be replaced with the id and the secret of the OAuth application created on Gitlab. See *1. Adding the OAuth application on GitLab* above.

## GitLab CE/EE

GitLab CE/EE is no different than GitLab.com, the steps are the same.

In step *1. Adding the OAuth application on GitLab*, just replace `gitlab.com` with your private GitLab CE/EE instance host (example: *https://gemnasium.example.com/auth/auth/private-gitlab.example.com/sync/callback*).

When using *A. Automatic configuration*, make sure to select "GitLab CE/EE" instead of "GitLab.com". The script will ask for your GitLab instance URL, and to name it.

When using *B. Advanced configuration*, make sure to replace all occurences of `gitlab.com` with your private GitLab CE/EE instance host. You also need to replace `"Gitlab"` with the name you want in the last command to enable synchronization feature. This is the name of the source that will show up in the "Add projects" page.

Several GitLab CE/EE instances can be configured, just name them accordingly to avoid confusion.

### Requirements

Gemnasium Enteprise is compatible with GitLab >= 8.14. There is a bug in nginx affecting lower versions of GitLab.

CHAPTER 17

# Bitbucket Cloud

Before being able to add projects from bitbucket.org (A.K.A. Bitbucket Cloud), Gemnasium Enterprise needs to be configured to be able to access it.

First add an OAuth consumer on bitbucket.org, then configure Gemnasium Enterprise to use that OAuth consumer.

## Adding OAuth consumers on Bitbucket.org

The first step is to go to open the "Add OAuth consumer" form:

- Go to the Bitbucket Settings page.
- If needed, use the dropdown list and select the Bitbucket account of your company.
- Click on the "OAuth" item in the sidebar.
- Click on the "Add consumer" button.

Or simply visit: https://bitbucket.org/account/user/{[}your_account{]}/oauth-consumers/new

Then fill the form:

- Set the name to "Gemnasium Enterprise".
- Set the callback URL to be `{homepage URL}/auth/auth/bitbucket.org` where you replace the home with your Gemnasium Enterprise host (example: *https://gemnasium.example.com/auth/auth/bitbucket.org*)
- Set the URL to your Gemnasium Enterprise base URL (example: "https://gemnasium.example.com/").
- Grant permissions to: read account, read repositories, read and write webhooks.
- Save the new OAuth consumer.

You'll then be redirected to the OAuth Consumers page and the consumer named "Gemnasium Enterprise" will be visible in the list. Click on its name to expand the item and retrieve the credentials: the consumer key and its secret.

## Configure Gemnasium Enterprise to use Bitbucket.org

A convenient script is provided to configure your instance:

```
docker exec -it gemnasium configure
```

Your Gemnasium Enterprise users are now able to login using their Bitbucket.org account. A new source named "Bitbucket.org" is also available on the "Add Project" screen.

## Advanced configuration

The default configuration described above enables both Bitbucket.org Sign In and project synchronization with Bitbucket.org. The advanced configuration makes it possible to restrict the integration to one of these two features.

To enable Bitbucket.org Sign Up, add an OAuth consumer with the permissions "account" and "email", then run these commands to configure Gemnasium Enterprise accordingly:

```
docker exec -it gemnasium auth provider create bitbucket.org bitbucket https://
↪bitbucket.org/site/oauth2/authorize https://bitbucket.org/site/oauth2/access_token
docker exec -it gemnasium auth clients create bitbucket.org login OAUTH_CONSUMER_KEY␣
↪OAUTH_CONSUMER_SECRET account,email
```

To enable Bitbucket.org Synchronization, add an OAuth consumer with the permissions "repository" and "webhook", then run these commands to configure Gemnasium Enterprise accordingly:

```
docker exec -it gemnasium auth provider create bitbucket.org bitbucket https://
↪bitbucket.org/site/oauth2/authorize https://bitbucket.org/site/oauth2/access_token
docker exec -it gemnasium auth clients create bitbucket.org sync OAUTH_CONSUMER_KEY␣
↪OAUTH_CONSUMER_SECRET repository,webhook
docker exec -it gemnasium repo-syncer sources create bitbucket.org "Bitbucket Cloud"␣
↪bitbucket https://api.bitbucket.org
```

`OAUTH_CONSUMER_KEY` and `OAUTH_CONSUMER_SECRET` must replaced with the key and the secret of the OAuth consumer created on Bitbucket.org. See *Adding OAuth consumers on Bitbucket.org* above.

CHAPTER 18

# Bitbucket Server

Before being able to add projects from Bitbucket Server (formerly know as Atlassian Stash), Gemnasium Enterprise needs to be configured to be able to access it.

Bitbucket Server is different from the other integrations since OAuth 1.0 is being used, instead of OAuth 2.0.

First add an OAuth consumer on Bitbucket Server, then configure Gemnasium Enterprise to use that OAuth consumer.

## Adding OAuth consumers on Bitbucket Server

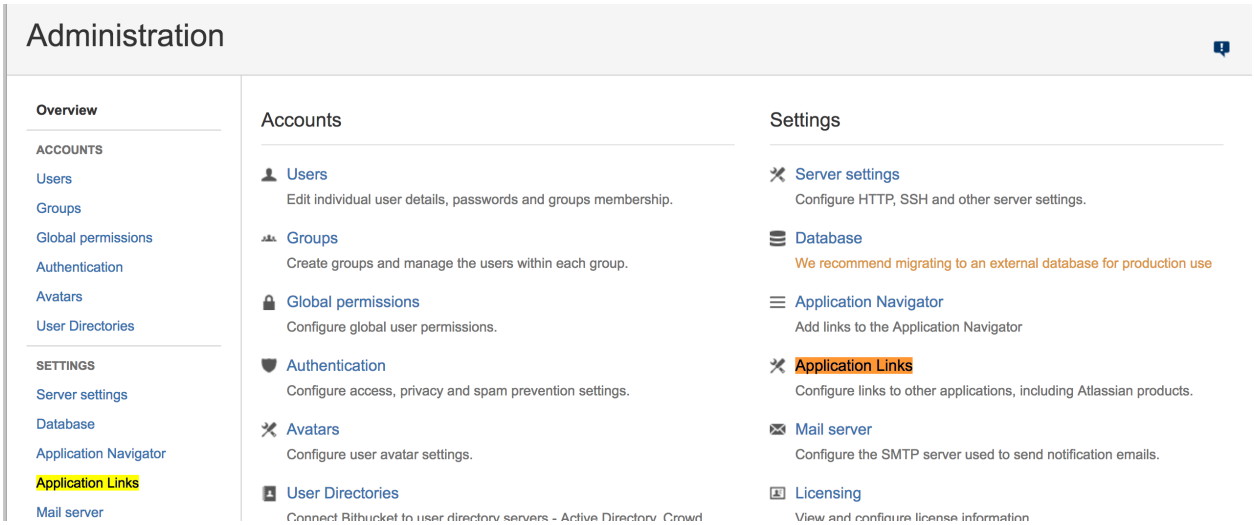
Before configuring Bitbucket Server, make sure you have the public key of the certificate Gemnasium Enterprise uses to sign off all the requests it sends to OAuth 1.0 providers like Bitbucket Server. This public key is displayed when running the Gemnasium Enterprise configure script (see *Configure Gemnasium Enterprise to use Bitbucket Server* below).

On Bitbucket Server, OAuth consumers are registered as a special kind of "Application Links" with an "Incoming Authentication" configuration.

To access the "Application Links" settings:

- Log in Bitbucket Server as an admin,
- Go to the settings page, or click on the gear icon that's in the upper-right corner of the page:

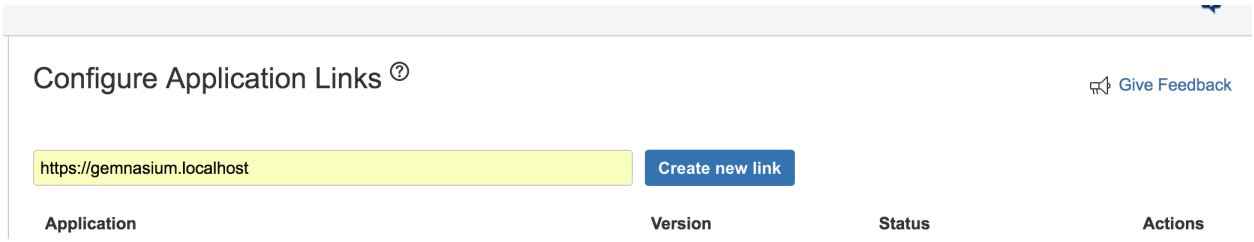
- Click on the "Application Links" item that's under the "SETTINGS" section of the sidebar:




Or simply visit https://{[}bitbucket_server{]}/plugins/servlet/applinks/listApplicationLinks where `bitbucket_server` is the hostname of your Bitbucket Server instance.

Then create a new Application Link for Gemnasium Enterprise:

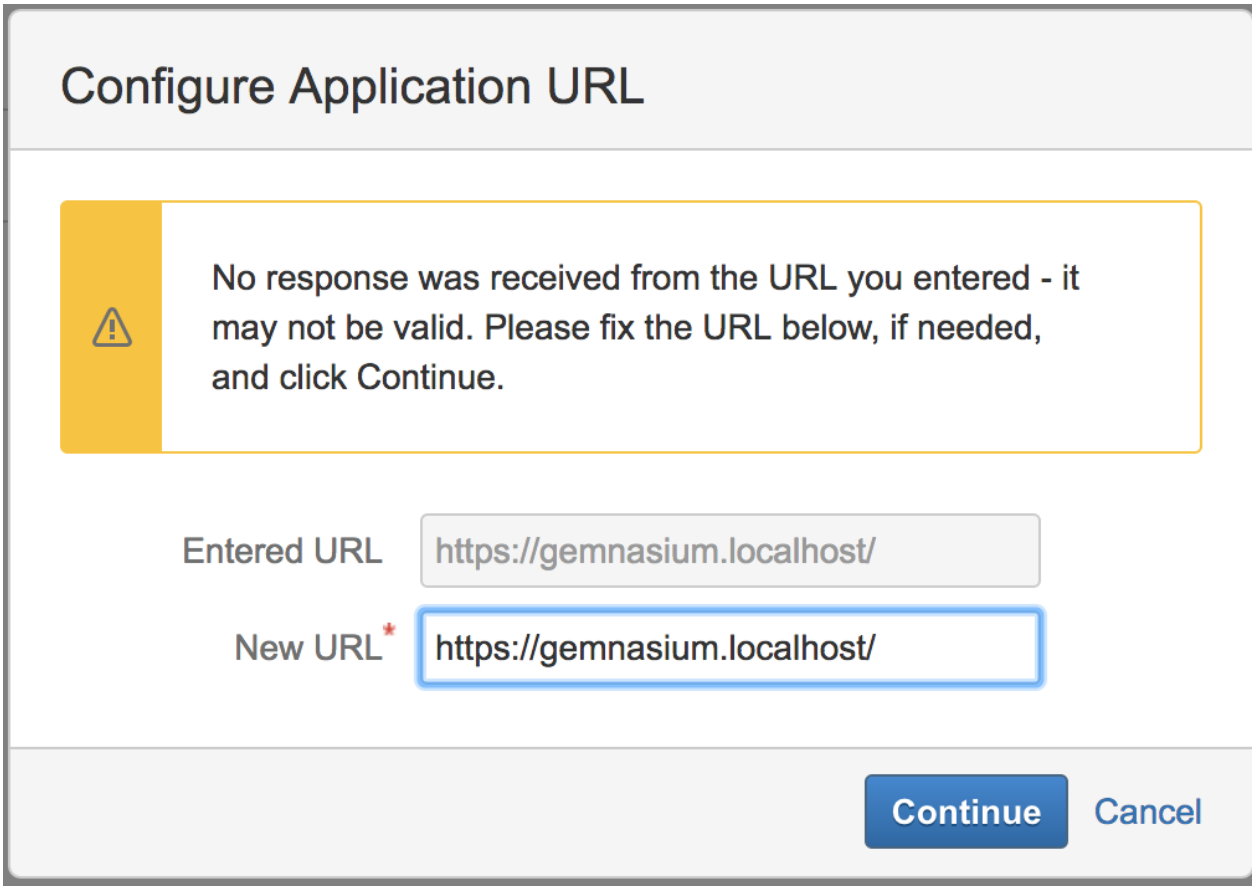
- Enter the URL of your Gemnasium Enterprise instance,




(https://gemnasium.localhost is an example, use your instance URL here)

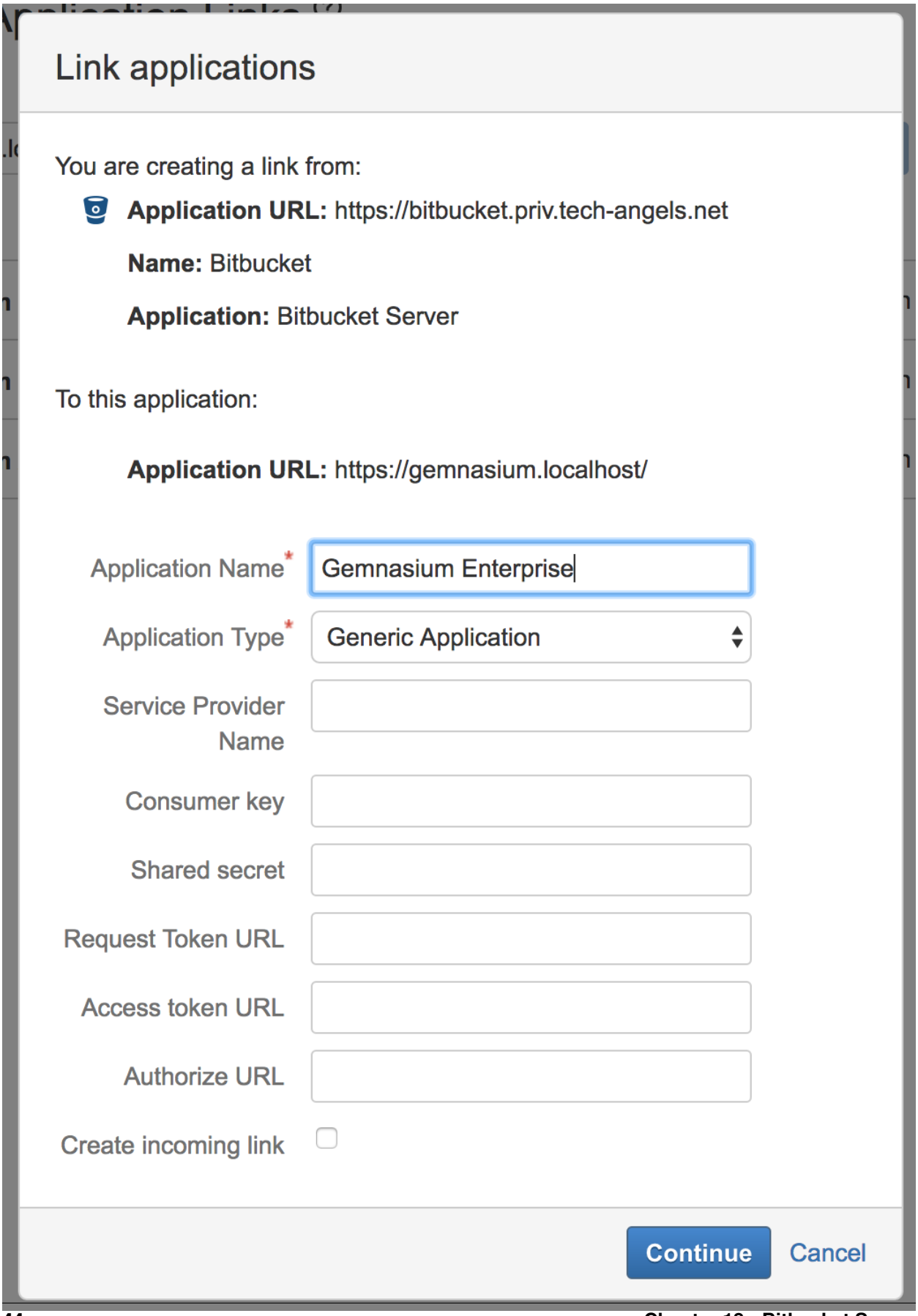
- Click on "Create new link".

Bitbucket Server then shows a warning because it's not able to probe the URL in order to configure the Application Link automatically. You can safely ignore this warning and proceed by clicking on "Continue".




Then, fill in the mandatory fields of the "Link application" form:

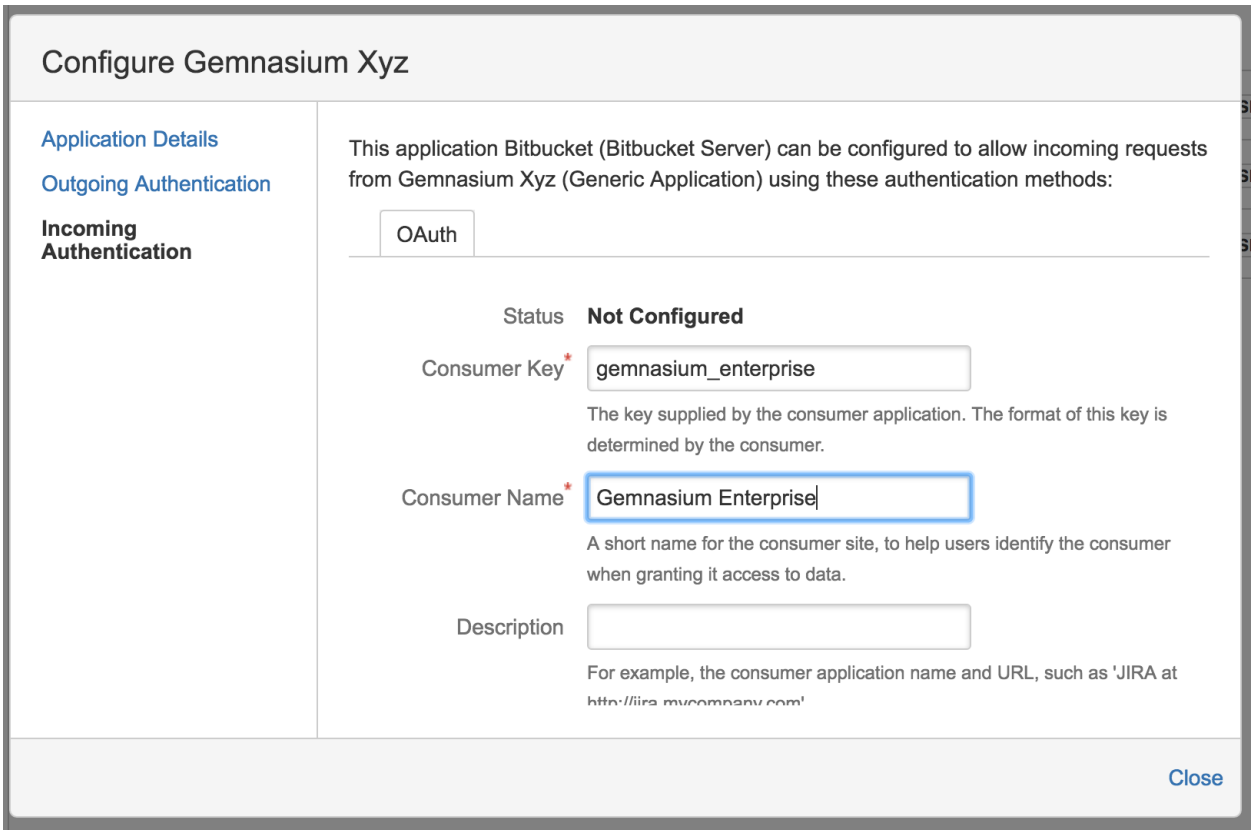
- Set the Application Name to "Gemnasium Enterprise",
- Set the Application Type to "Generic Application",
- Let the other fields empty and click on "Continue".


Link applications
You are creating a link from:
Application URL: https://bitbucket.priv.tech-angels.net
Name: Bitbucket
Application: Bitbucket Server
To this application:
Application URL: https://gemnasium.localhost/
Application Name*
Gemnasium Enterprise
Application Type*
Generic Application
Service Provider Name
Consumer key
Shared secret
Request Token URL
Access token URL
Authorize URL
Create incoming link
Continue
Cancel

A new Application Link shows up in the list. Click on pen icon in order to edit its properties.

Then click on "Incoming Authentication" and fill in the form:

- Set the Consumer Key to "gemnasium_enterprise",
- Set the Consumer Name to "Gemnasium Enterprise",
- Paste the Public Key that was given when configuring Gemnasium Enterprise (see *Configure Gemnasium Enterprise to use Bitbucket Server* below),
- Set the Consumer Callback URL to `{gemnasium_enterprise_url}/auth/auth/{bitbucket_hostname}` where you replace *gemnasium_enterprise_url* with the base URL of your Gemnasium Enterprise and *bitbucket_hostname* with the hostname of your Bitbucket Server instance (example: *https://gemnasium.example.com/auth/auth/bitbucket.example.org*)
- Click on "Save".




**Important:** Make sure to use "gemnasium_enterprise" as the Consumer Key

Gemnasium Enterprise can now connect to Bitbucket Server using the OAuth 1.0 protocol.

# Configure Gemnasium Enterprise to use Bitbucket Server

A convenient script is provided to configure your instance:

```
docker exec -it gemnasium configure
```

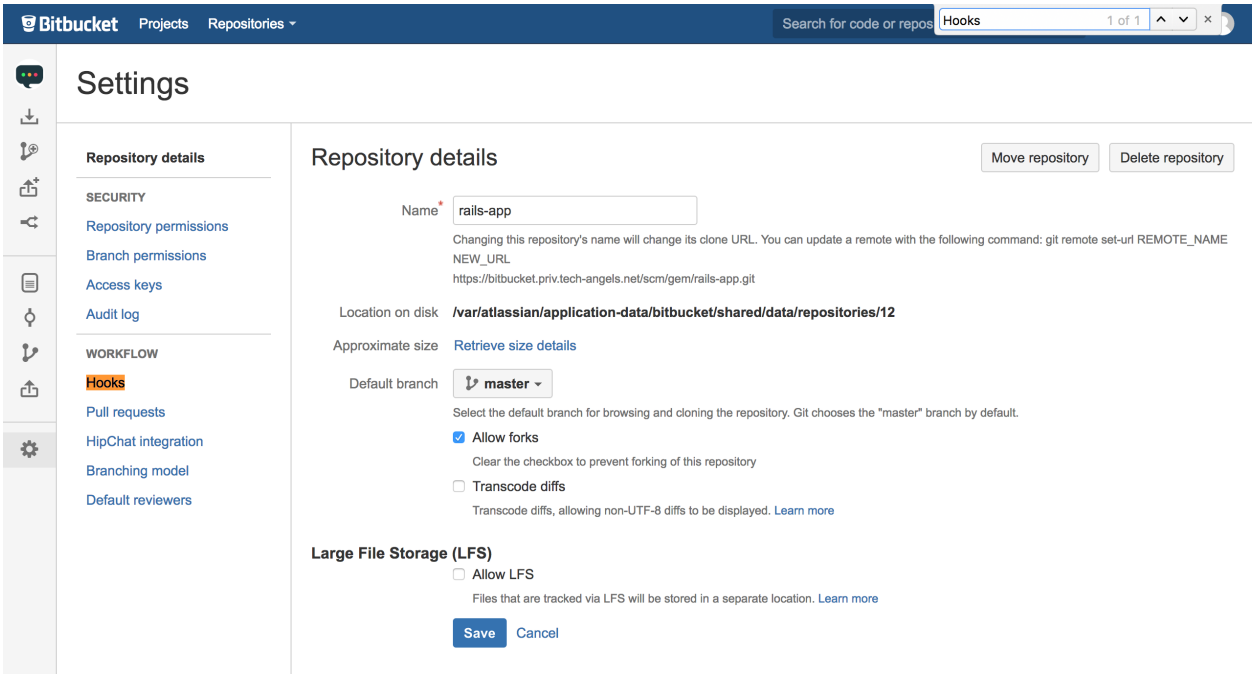
When prompted, give the hostname of your Bitbucket Server instance and the OAuth Consumer Key you have registered Gemnasium Enterprise with (example: *gemnasium_enterprise*).

Your Gemnasium Enterprise users are now able to login using their Bitbucket Server account. Also, a new repository source named "Bitbucket Server" is available on the "Add Project" screen.

# Adding project webhooks

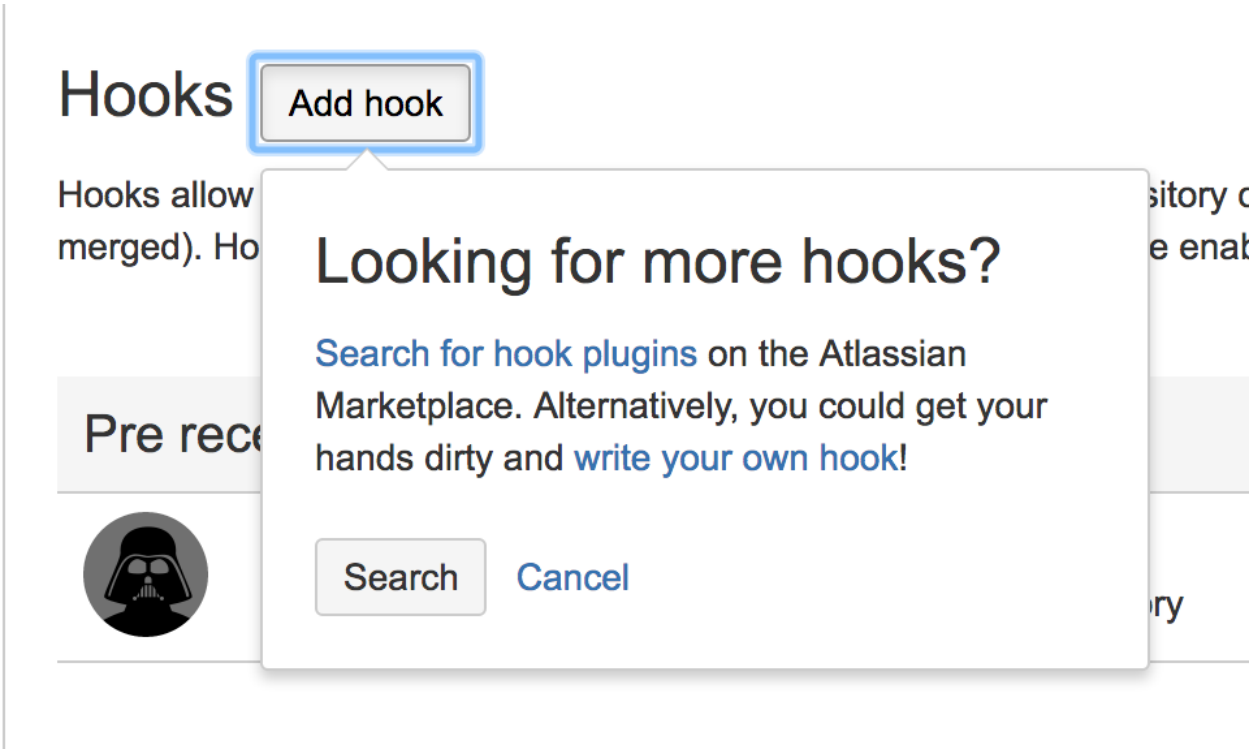
> **Danger:** Bitbucket Server API does not provide any way to create webhooks in projects like GitHub or Gitlab (or even bitbucket.org). This operation is therefore manual, and must be executed for each project added to Gemnasium Enterprise.

Go to your Bitbucket Server project setting page, and click on the "Hooks" link.

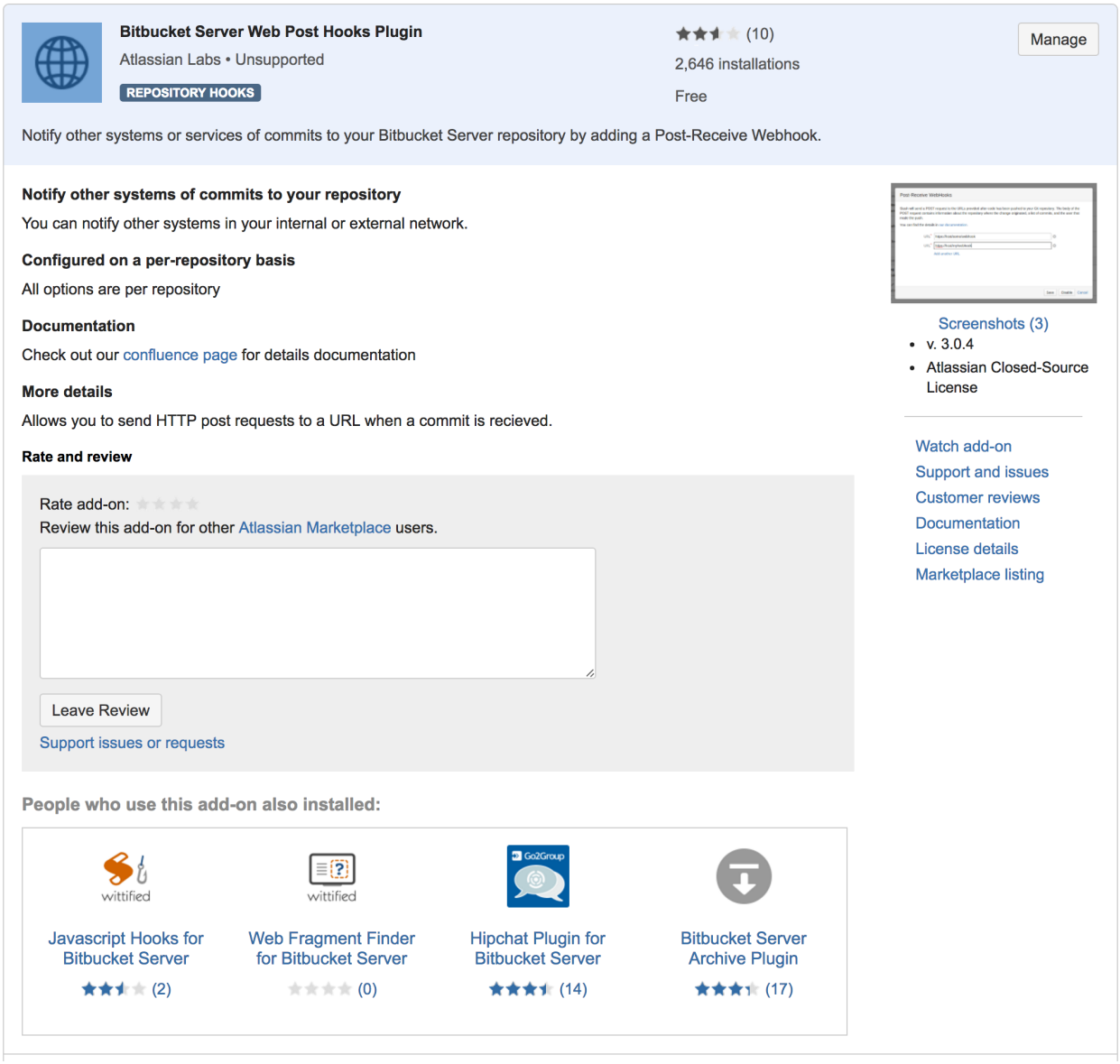



A webhook plugin must be installed (once) to be able to notify URLs.

- Click on the "Add Hook" button, then "Search"

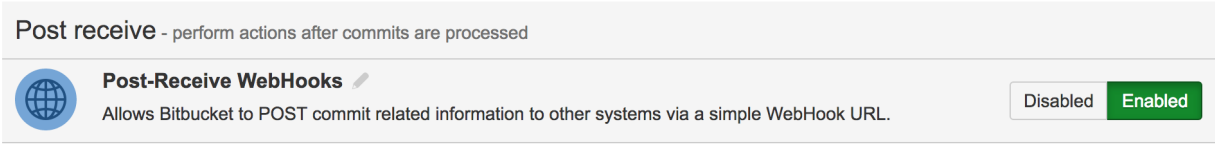



- A new tab will open with the Atlassian plugins marketplace
- Search for "Web Post Hooks", and select the "Bitbucket Server Web Post Hooks Plugin" plugin

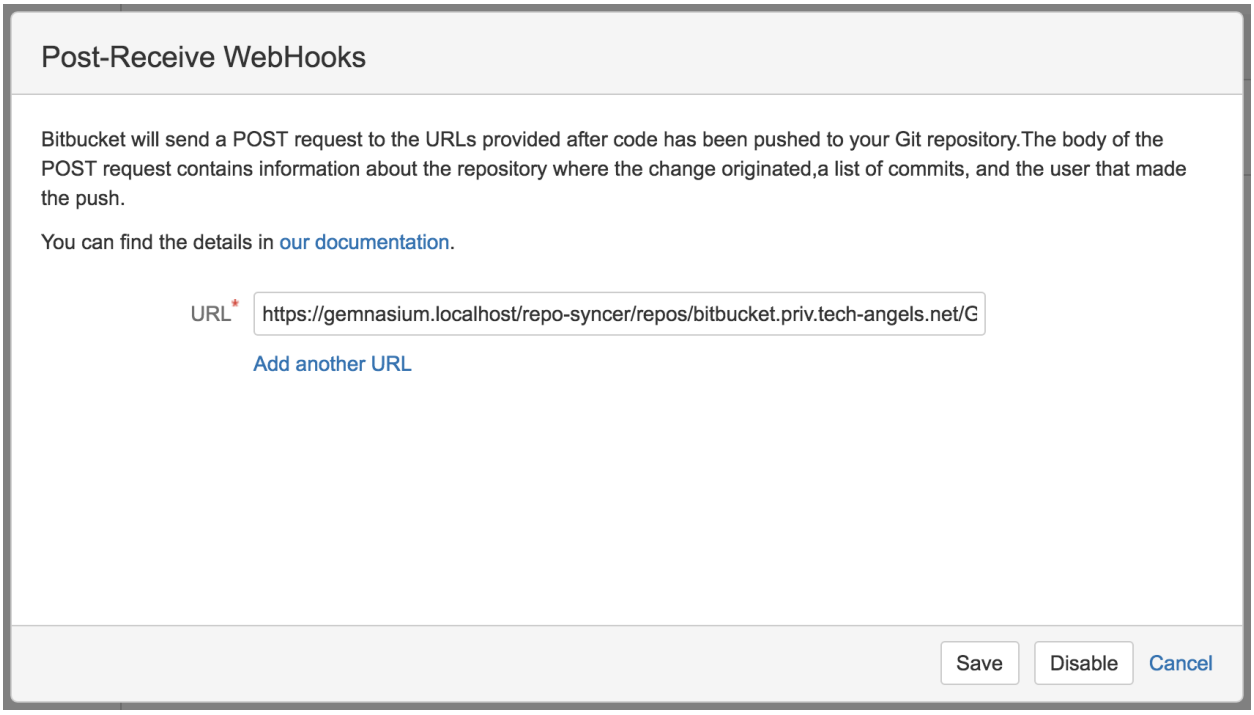



Make sure you have the required permission level to install this plugin, or request the installation to an admin of your Bitbucket Server.

- Once installed, enable the plugin in the project settings:




- Click on "Post-Receive WebHooks" (or the pen next to the name) to configure the plugin




Enter the following URL:

> {gemnasium_enterprise_url}/repo-syncer/repos/{project_slug}/events

where:

- `gemnasium_enterprise_url` is the URL of your Gemnasium Enteprise instance, starting with https://
- `project_slug` is composed of `{bitbucket_hostname}/{project}/{repo}`

In our example, the hook URL is https://gemnasium.localhost/repo-syncer/repos/bitbucket.priv.tech-angels.net/GEM/rails-app/events and the repo URL is https://bitbucket.priv.tech-angels.net/projects/GEM/repos/rails-app/browse

CHAPTER 19

# Slack

Because it's an enterprise version installed on your server, you need to do a few things in order to add Slack support.

The first step is to create a new Slack application, on Slack itself. Here are the steps:

- Go on https://api.slack.com/apps?new_app=1
- Set the App Name to be "Gemnasium Enterprise", and then click on the "Create App" button.
- You will need the Client ID and Client Secret in a moment. You can keep them around or go back to see them when you need them.
- (optional) By default, there is no Gemnasium icon for the messages sent on Slack. If you want it, which we suggest, you can upload it on the "Basic Information" screen of the app you just created, in the "Display Information" section. `You can use this icon.`.
- Go in the "OAuth & Permissions" section (in the left menu) and set the redirect URL to https://gemnasium.example.com/auth/auth/slack.com/webhook/callback

Now, we need to configure Gemnasium Enterprise to use it:

```
docker exec -it gemnasium configure
```

Select "Slack", and then fill the corresponding fields with the values from the apps created above.

Now you only need to configure Slack notifications for a project in the web UI. You can do that in the "Settings" of each project.