



This is an **application providing tools not available in GitLab CE** (Gitlab push/pull mirror and more).

- ***Allows to automatically mirror your Git or SVN repositories to GitLab by hook trigger or periodically.***
- ***Allows to automatically mirror your GitLab repository to any remote Git repository.***

Hi there, my name is João Marques!

I have graduated from University of Minho in Informatics Engineering and I am currently pursuing the Master's Degree.

Deployed on 29/July/2020



Source:

<https://github.com/Salamek/gitlab-tools>



Salamek/gitlab-tools is licensed under the  
**GNU General Public License v3.0**

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

**Version 1.0.0**

08/August/2020

It is possible to deploy it in a containerized version towards a web app hosting platform  
(Docker-Compose with Kubernetes)

That version is not cover in this one

For future reference:

<https://blog.ssdnodes.com/blog/host-multiple-websites-docker-nginx/>  
<https://medium.com/@cirolini/docker-flask-e-uwsgi-d10e58c56489>

[https://docs.docker.com/engine/examples/postgresql\\_service/](https://docs.docker.com/engine/examples/postgresql_service/)  
<https://stackoverflow.com/questions/33711818/embed-sqlite-database-to-docker-container>  
<https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/docker-mysql-getting-started.html>

<https://github.com/qubitron/flask-webapp-quickstart>

<https://docs.microsoft.com/azure/app-service/app-service-web-tutorial-python-postgresql>  
<https://medium.com/@nikovrdoljak/deploy-your-flask-app-on-azure-in-3-easy-steps-b2fe388a589e>



Image 1: Index Page (Sign in with GitLab Account)

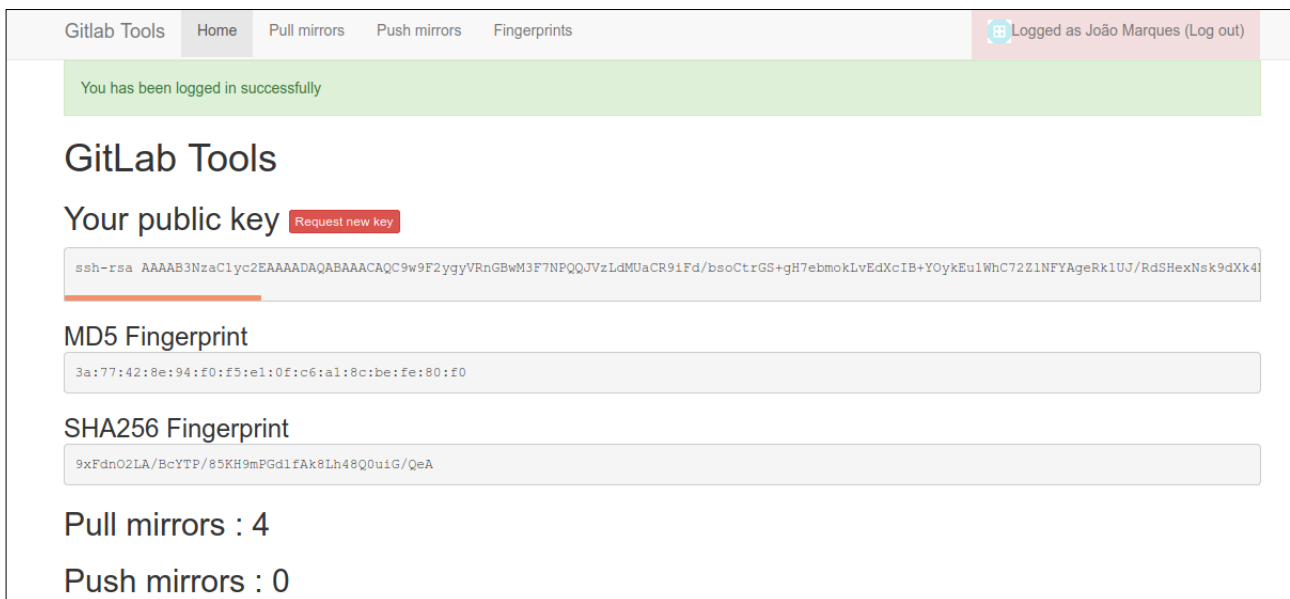


Image 2{ Main Page (public key will be used) }

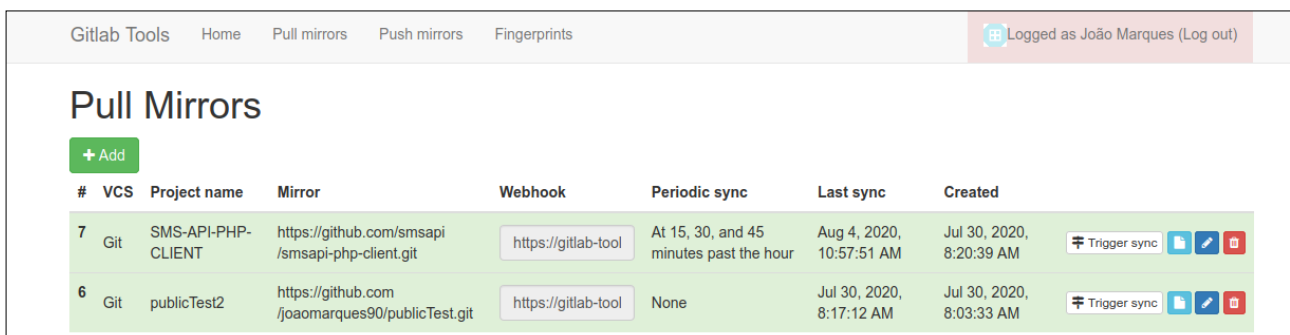


Image 3{ Pull Mirrors Page (Push Mirrors Page is similar) }

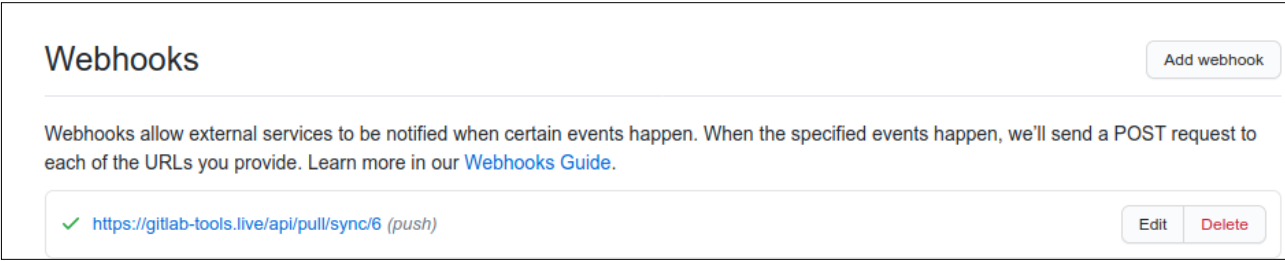


Image 4{ GitHub Webhook Example }

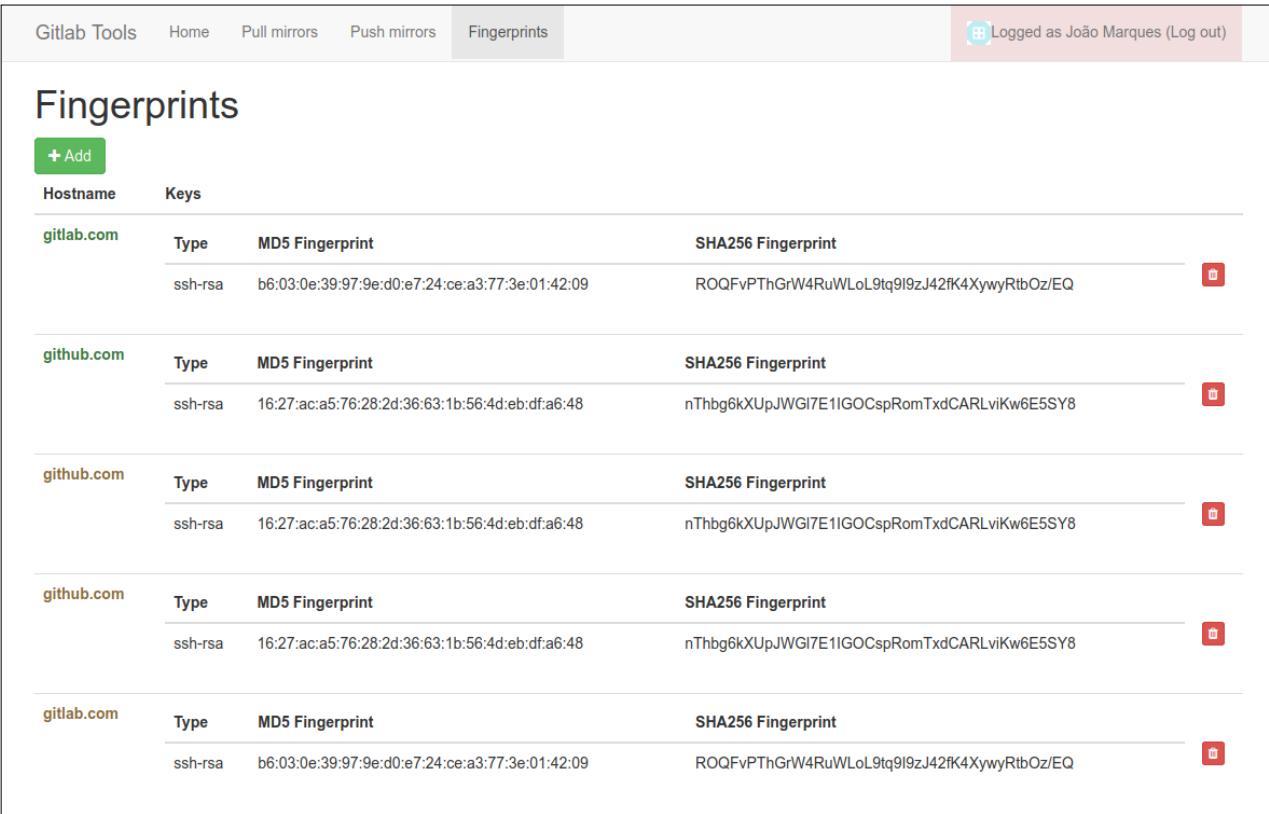


Image 6{ Fingerprints Page }



Image 5{ New pull/push mirror Page }

# Table of Contents

1.	Git .....	6
1.1.	SVN & Git.....	6
1.2.	Git mirroring.....	7
2.	Synchronization .....	8
3.	Client's Customs .....	8
4.	GitLab CE.....	8
5.	Database .....	9
5.1.	Accepted DB .....	10
5.1.1.	PostgreSQL.....	11
5.1.2.	MySQL .....	11
5.1.3.	SQLite.....	11
5.2.	DB SaaS free tiers providers .....	12
6.	Locally, Cloud or On-Premise .....	13
6.1.	nginx.....	14
6.1.1.	Installation .....	15
6.1.2.	Configuration.....	16
6.2.	uwsgi.....	17
6.2.1.	Installation .....	17
6.2.2.	Configuration.....	17
6.3.	Flask.....	17
6.4.	Domain name .....	17
6.5.	Repository Access .....	18
6.5.1.	Webhooks.....	18
6.5.1.1	Description.....	18
6.5.1.2	Configuration.....	18
6.5.2.	Cron expressions.....	18
6.5.2.1.	Description .....	18
6.5.2.2.	Examples.....	18
6.6.	Security .....	19
6.6.1.	IP Filtering .....	19
6.6.2.	Basic Authentication.....	19
6.6.3.	SSL .....	20
6.6.4.	DNSSEC.....	21
6.6.4.1.	Enabling DNSSEC.....	21
6.6.5.	TLS .....	21
6.6.6.	CDN Edge.....	23
6.6.7.	HTTP-S/x.....	24
6.6.7.1.	Quic.....	25
6.6.7.2.	ssllabs.com/ssltest.....	27
6.7.	Monitoring Dashboard Tools.....	28
6.7.1.	Amplify nginx.....	28
6.7.2.	CDN Edge .....	29
6.7.3.	Database.....	31
6.7.4.	Flask.....	32
7..	Installation & Configuration .....	33
8.	Logs .....	37
8.1.	ufw .....	37
8.2.	nginx.....	37
8.2.1.	access .....	37
8.2.2.	error.....	37
8.3.	uwsgi.....	37
8.4.	app_crash.....	37
9.	Target Audience .....	38
10.	Time To Deploy .....	38
11.	Total Cost of Ownership (TCO).....	38
12.	Time To Detect and Time To Mitigate (TTD, TTM).....	39
13.	Credits .....	40
	Image Index .....	41

# WHAT? [A]

## • [1] Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows

### • [1.1] SVN & Git

Apache Subversion, also known as Subversion, SVN represents the most popular centralized version control system on the market. With a centralized system, all files and historical data are stored on a central server. Developers can commit their changes directly to that central server repository.

Whether using Git, SVN, or another alternative, you will benefit from being able to track and review your code for better releases. Just be sure to choose an issue tracking software that supports your choice, so you are able to track properly that work over time.

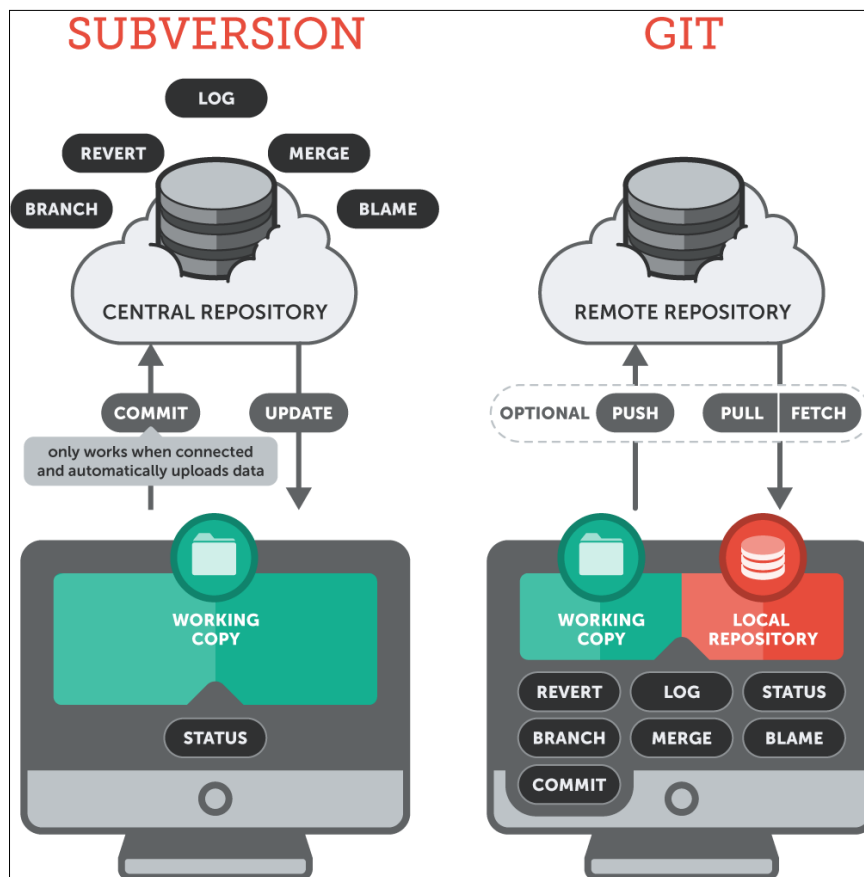


Image 7 {SVN vs Git}

## • [1.2] Git Mirroring

Repository mirroring is very useful when, for some reason, you must use a project from another source.

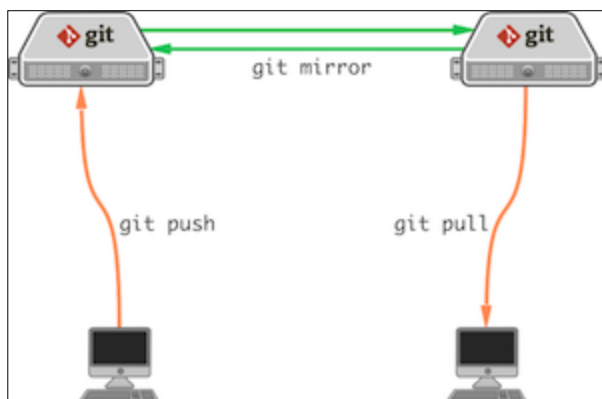
**There are two kinds of repository mirroring features supported by GitLab: push and pull, the latter being only available in GitLab Enterprise Edition (EE).**

The push method mirrors the repository in GitLab to another location.

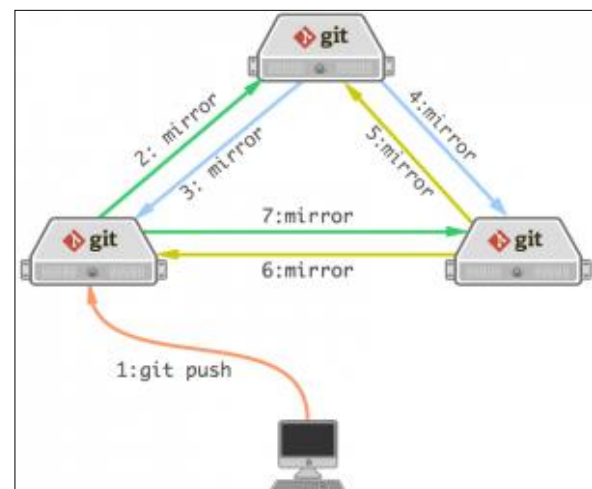
Once the mirror repository is updated, all new branches, tags, and commits will be visible in the project's activity feed.

Users with at least developer access to the project can also force an immediate update with the click of a button. This button will not be available if the mirror is already being updated or 5 minutes still haven't passed since its last update.

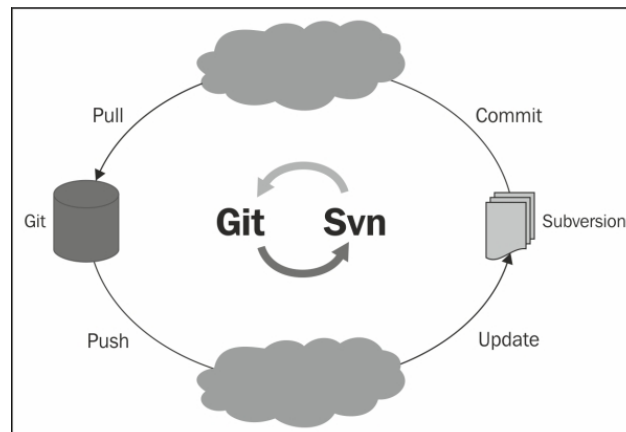
You need to use a personal access token instead of a password when you want to mirror to GitLab and have 2FA enabled.



*Image 8 { Git Mirror example 1 }*



*Image 9 { Git Mirror example 2 }*



*Image 10 { Git -Svn Mirror }*

# WHY? [B]

## • [2] Synchronization

You want to maintain synchronizations between GitLab and other git/svn platforms.

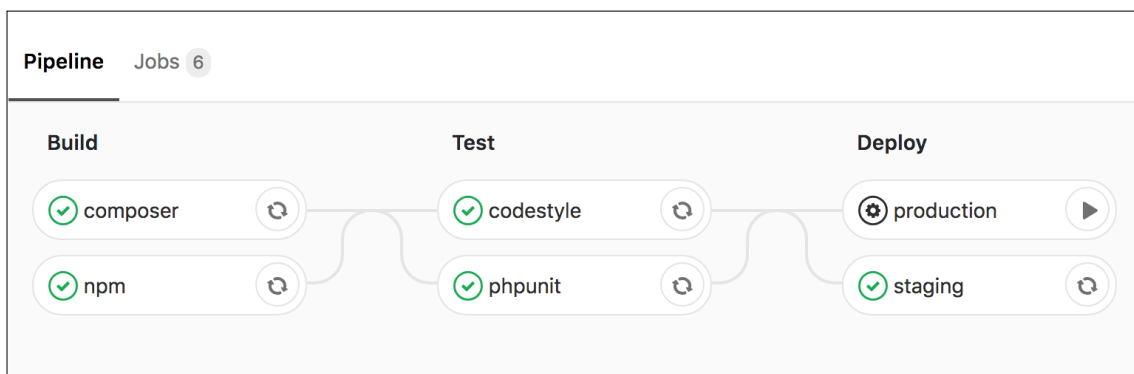
So if you update the origin repository then the mirror repository will also update. You can also maintain both doing pull and (request) pushes/merge. It all depends on your objectives.

It is highly customizable to the user needs.

## • [3] Client's Customs (Repository or GitLab Pipeline)

As mentioned in [2] but you may also be working as a team for a company that does not want to migrate to GitLab.

May also be the case you or your team are more familiar or prefers using GitLab Pipeline rather than GitHub Actions for some reason. Once the file '[.gitlab-ci.yml](#)' is updated on the source repository, the GitLab-destiny repository will also be updated and hence will trigger the CI pipeline execution.



*Image 11 { GitLab Pipeline example }*

## • [4] GitLab CE (free)

As mentioned in [1.2] these features are totally available only in GitLab EE which is paid.

This free tool will give you access to those features without any cost associated while using the GitLab Community Edition.

Feature name	Gitlab CE	Gitlab EE	Gitlab Tools	Description
Pull mirror <a href="#">(doc)</a>	No	Yes	Yes	Allows to automatically mirror your Git or SVN repositories to GitLab by hook trigger or periodicaly
Push mirror <a href="#">(doc)</a>	Yes(10.8)	Yes	Yes	Allows to automatically mirror your GitLab repository to any remote Git repository

*Image 12 { Comparison between service features }*



# HOW? [C]

## • [5] Database (logs)

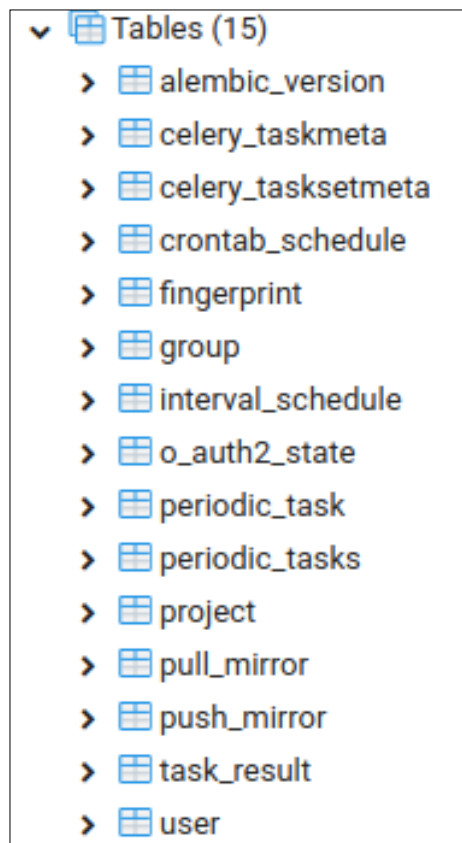
Databases are used to hold administrative information and more specialized data.

Data within the most common types of databases in operation today is typically modelled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized.

From a security point of view, the purpose of a log is to act as a red flag when something bad is happening. Reviewing logs regularly could help identify malicious attacks on your system. Given the large of amount of log data generated by systems, it is impractical to review all of these logs manually each day.

This application tool stores the following tables inside the database. It is quite easy also to see when anything went bad by viewing the stored data. Specially the “celery\_taskmeta” table where in case of failure shows the specific warning associated. Never the less, all information and operations are logged and detailed.

You can install or alternatively you can use a DBaaS provider instead.



*Image 13 {Database Log example }*

## • [5.1] Accepted DB



*Image 14 { Comparison between DB }*

The creator of this tool has supported three database types:

- ✓ **SQLite**
- ✓ **PostgreSQL**
- ✓ **MySQL.**

He states the recommended database type is PostgreSQL.

In the following table you may see the advantages, disadvantages, when to use and when not to accordingly to each database type.

Name	Advantages	Disadvantages	When to use	When not to use
SQLite	<ul style="list-style-type: none"> <li>• File based</li> <li>• Standards-aware</li> <li>• Great for developing and even testing</li> </ul>	<ul style="list-style-type: none"> <li>• No user management</li> <li>• Lack of possibility to tinker with for additional performance</li> </ul>	<ul style="list-style-type: none"> <li>• Embedded applications</li> <li>• Disk access replacement</li> <li>• Testing</li> </ul>	<ul style="list-style-type: none"> <li>• Multi-user applications</li> <li>• Applications requiring high write volumes</li> </ul>
MySQL	<ul style="list-style-type: none"> <li>• Easy to work with</li> <li>• Feature rich</li> <li>• Secure</li> <li>• Scalable and powerful</li> <li>• Speedy</li> </ul>	<ul style="list-style-type: none"> <li>• Known limitations</li> <li>• Reliability issues</li> <li>• Stagnated development</li> </ul>	<ul style="list-style-type: none"> <li>• Distributed operations</li> <li>• High Security</li> <li>• Web-sites and Web-applications</li> <li>• Custom solutions</li> </ul>	<ul style="list-style-type: none"> <li>• SQL compliance</li> <li>• Concurrency</li> <li>• Lack of features</li> </ul>
PostgreSQL	<ul style="list-style-type: none"> <li>• An open source SQL standard compliant RDBMS</li> <li>• Strong Community</li> <li>• Strong third-party support</li> <li>• Extensible</li> <li>• Objective</li> </ul>	<ul style="list-style-type: none"> <li>• Performance</li> <li>• Popularity</li> <li>• Hosting</li> </ul>	<ul style="list-style-type: none"> <li>• Data Integrity</li> <li>• Complex, custom procedures</li> <li>• Integration</li> <li>• Complex designs</li> </ul>	<ul style="list-style-type: none"> <li>• Speed</li> <li>• Simple to sets up</li> <li>• Replication</li> </ul>

*Image 15 { Detailed Comparison between DB }*

Notes:

- The end user has the option to have a local database or a remote database.
- The authentication (host, door, user and password) will be needed for further installation
- **During the mirroring, all data will be locally downloaded and kept!**  
**! Be aware of the repository size !**

### • [5.1.1] PostgreSQL

```
$ sudo apt update && sudo apt install postgresql postgresql-contrib -y  
$ sudo -u postgres createuser --interactive
```

Now fill it with the desired and required information.

«done»

Check if the database service is running:

```
$ systemctl status postgresql.service
```

### • [5.1.2] MySQL

**Pre-requisites:** non-root user with sudo privileges.

```
$ sudo apt update && sudo apt install mysql-server -y  
$ sudo mysql_secure_installation
```

Let us create a user with the desired username and password:

```
$ sudo mysql  
$$ CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';  
$$ CREATE USER 'username'@'%' IDENTIFIED BY 'password';  
$$ exit
```

«done»

```
$ systemctl status mysql.service
```

### • [5.1.3] SQLite (Local)

```
$ sudo apt update && sudo apt install sqlite3 -y
```

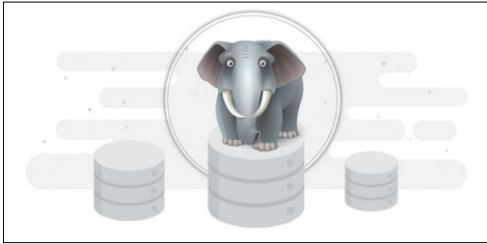
«done»

## • [5.2] DB SaaS free tiers providers

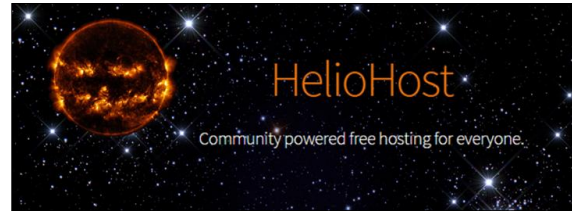
There are many free DBaaS providers and each one of them have different limitations. So choose, if that is the case, the one that is more appropriated according to your need.

Have a look here: <https://www.hostingadvice.com/how-to/best-free-database-hosting/>

Here is a bunch of providers, which I recommend:



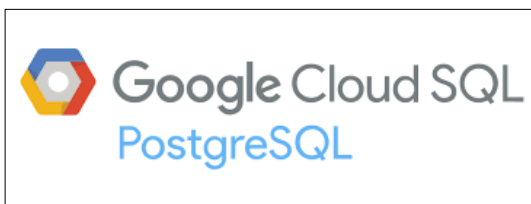
*Image 17 { ElephantSQL }*



*Image 16 { HelioHost }*



*Image 19 { Amazon RDS }*



*Image 18 { Google Cloud SQL }*



*Image 21 { Heroku }*



*Image 20 { Azure PostgreSQL }*



*Image 22 { GoogieHost }*



*Image 23 { GigaRocket }*



*Image 24 { FreeHosting }*

## • [6] Locally, Cloud or On-Premise

**Local server** is a server, which resides locally into your machine. You can test your website as many times as you want before uploading it to the web server.

**Web servers** are computers that deliver Web pages. Every Web server has an IP address and possibly a domain name. In this context, you may have a **Cloud** or **On-Premise** web server.

Depending on your needs and resources, choose which one is the best option for your goal.

Note: Do not forget that if you want to make it available on the internet via *ANAME record*, you will need to possess an internet domain. It can also work without it, working with *IP address* and *door* as long as it has a *static IP* or that you have a work around for *dynamic IP* issues. The doors need to be open towards internet IP addresses.

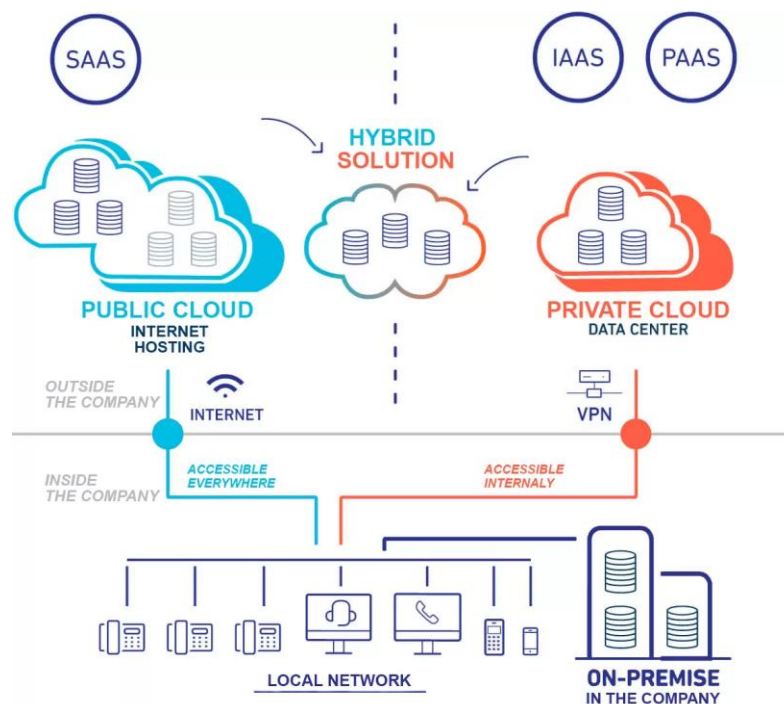


Image 25 { Computing Location }

Currently there are not free tiers for cloud IaaS due to safety precautions (hacking, ddos, xss, phishing and so forth). There are strict limitations such as credit duration, which is usually between 15 and 30 days.

The remaining time must be paid, even if other services might be free for 12 months (SaaS, PaaS).

Students usually receive 12 months to use the free credit (Azure, AWS, Google Cloud and so forth)

## • [6.1] nginx

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption

The following images illustrate quite well how it works and how all the pieces (**Flask**, **uWSGI**, **Bootstrap**, Databases) combine in this project. They are self-explanatory.

**Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. It is being used by this tool to the fullest.

uWSGI and Flask usage and details are explained on the next chapters.

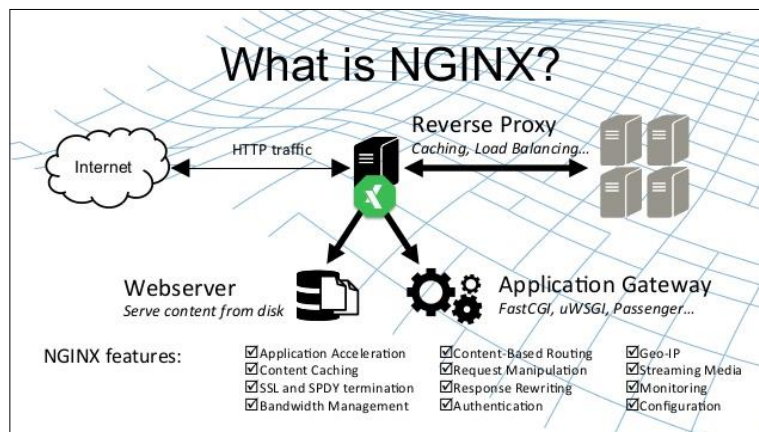


Image 27 { nginx description }

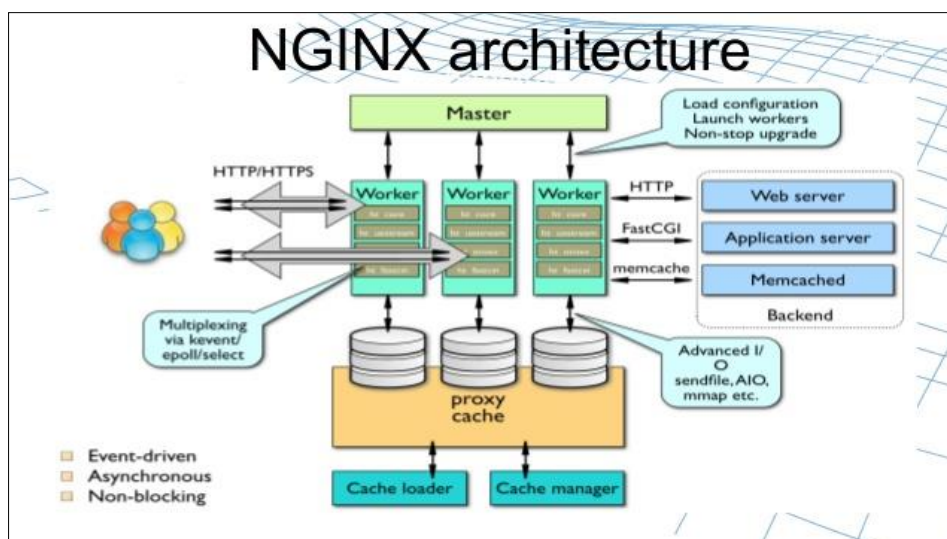


Image 26 { nginx architecture }



Image 28 { Server architecture details 1}

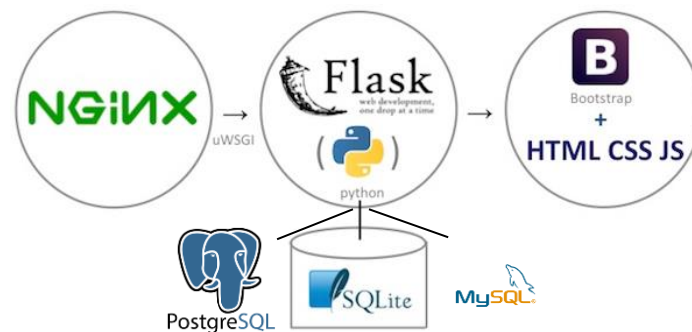


Image 29 { Server architecture details 2}

### • [6.1.1] nginx Installation

```
$ sudo apt update
$ sudo apt install nginx -y
```

Adjusting the firewall:

```
$ sudo ufw app list
```

«Output

Available applications:

```
Nginx Full
Nginx HTTP
Nginx HTTPS
OpenSSH
```

Output»

Recommended action:

```
$ sudo ufw allow 'Nginx Full'
```

Uncomment 'server\_names\_hash\_bucket\_size 64' to avoid nginx complains

```
$ sudo nano /etc/nginx/nginx.conf
```

Managing nginx service:

```
$ systemctl status/enable/disable/start/stop/reload/restart nginx
```



## • [6.1.2] nginx Configuration

### General Configuration

- ❖ **/etc/nginx:** The Nginx configuration directory. All of the Nginx configuration files reside here.
- ❖ **/etc/nginx/nginx.conf:** The main Nginx configuration file. This can be modified to make changes to the Nginx global configuration.
- ❖ **/etc/nginx/sites-available/:** The directory where per-site server blocks can be stored. Nginx will not use the configuration files found in this directory unless they are linked to the sites-enabled directory. Typically, all server block configuration is done in this directory, and then enabled by linking to the other directory.
- ❖ **/etc/nginx/sites-enabled/:** The directory where enabled per-site server blocks are stored. Typically, these are created by linking to configuration files found in the sites-available directory.
- ❖ **/etc/nginx/snippets:** This directory contains configuration fragments that can be included elsewhere in the Nginx configuration. Potentially repeatable configuration segments are good candidates for refactoring into snippets.

Do the following steps:

```
$ sudo nano /etc/nginx/sites-available/gitlab-tools.example.com
```

Add this and save (*ctrl + x, y, Enter*):

```
server {  
    listen 80;  
    listen [::]:80;  
    #Change the underscore by the given server name  
    server_name gitlab-tools.example.com;  
  
    root /usr/lib/python3/dist-packages/gitlab_tools;  
}
```

After the installation of this tool, you can check if everything is okay with

```
$ sudo nginx -t
```

Do not forget to “reload” your *nginx* after changing this file.

If you are using also *Uwsgi* my recommendation is to *reload* it as well after reloading *nginx*.



## • [6.2] Uwsgi

### • [6.2.1] Uwsgi Installation

Just run the following command:

```
$ sudo apt install uwsgi uwsgi-plugin-python3
```

### • [6.2.2] Uwsgi Configuration

**Do the following steps:**

Change the underscore by the given server name

```
$ sudo /etc/uwsgi/apps-available/gitlab-tools.example.com.ini
```

Add this:

```
[uwsgi]
uid = gitlab-tools
master = true
chdir = /usr/lib/python3/dist-packages/gitlab_tools
socket = /tmp/gitlab-tools.sock
module = wsgi
callable = app
plugins = python3
buffer-size = 32768
```

## • [6.3] Flask

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website. In other words, Flask is the prototype used to create instances of web application or web applications if you want to put it simple. Therefore, once we import Flask, we need to create an instance of the Flask class for our web app.

## • [6.4] Domain name

Where you need to have an internet domain name, there are several free options for it.

The most known one is [www.dot.tk](http://www.dot.tk).

There are many free 12 months tier plans on other providers such as [www.namescheap.com](http://www.namescheap.com) or [www.name.com](http://www.name.com) .

A good one to compare prices is [www.tld-list.com](http://www.tld-list.com) .

After you obtained a domain name you will need to provide the IP address of your web server for the A (v4), AAAA (v6) records and a CNAME records [DNS] with page ruling (www Permanent Redirect).

## • [6.5] Repository Access

If you have or get repository access (owner, maintainer/developer) you should use *Webhooks*. If you do not, you may use a *Cron expression* to retrieve the repository content in a scheduled way.

### • [6.5.1] Webhooks

#### • [6.5.1.1] Description

A **webhook** in web development is a method of augmenting or altering the behaviour of a web page or web application with custom call-backs.

Webhooks are "user-defined HTTP call-backs". They are usually triggered by some event, such as pushing code to a repository or a comment being posted to a blog. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. Users can configure them to cause events on one site to invoke behaviour on another.

The format is usually JSON. The request is done as an HTTP POST request.

#### • [6.5.1.2] Configuration

When creating a push/pull mirror on the application website, inside that section it will be provided the link to where the webhook should respond to listed with the name "webhook".

GitHub example:

[https://github.com/USERNAME/REPO\\_NAME/settings/hooks](https://github.com/USERNAME/REPO_NAME/settings/hooks)

1. Click on "Add webhook"
2. Paste the link provided from "webhook" field inside the created *gitlab-tools website*
3. Choose "JSON type"
4. Choose "Just push Event"
5. Choose "Active"
6. Click "Add Webhook"

You can wait or force it by pressing on the "Trigger sync" button

### • [6.5.2] Cron expressions

#### • [6.5.2.1] Description

A **cron expression** is a string consisting of six or seven sub-expressions (fields) that describe individual details of the schedule. These fields, separated by white space, can contain any of the allowed values with various combinations of the allowed characters for that field.

<seconds> <minutes> <hours> <days of month> <months> <days of week> <years>

#### • [6.5.2.2] Examples

0 * * ? * *	Every minute
0 0 * ? * *	Every hour
0 0,15,30,45 * ? * *	Every hour at minutes 0,15, 30 and 45
0 0 */12 ? * *	Every twelve hours
0 0 0 * * ?	Every day at midnight - 12am
0 0 12 * * SUN	Every Sunday at noon
0 0 12 1 * ?	Every month on the 1st, at noon

## • [6.6] Security

Web server security is the protection of information assets that can be accessed from a Web server. Web server security is important for any organization that has a physical or virtual Web server connected to the Internet. It requires a layered defence and is especially important for organizations with customer-facing websites.

Separate servers should be used for internal and external-facing applications and servers for external-facing applications should be hosted on a CDN Edge or containerized service network to prevent an attacker from exploiting a vulnerability to gain access to sensitive internal information.

### • [6.6.1] IP Filtering

NGINX can allow or deny access based on a particular IP address or the range of IP addresses of client computers.

<https://docs.nginx.com/nginx/admin-guide/security-controls/controlling-access-proxied-tcp/#restrict>

Take as example:

```
server {  
deny 192.168.1.2;  
allow 192.168.1.1/24;  
allow 2001:0db8::/32;  
deny all;  
}
```

The rules are processed in sequence, from top to bottom: if the first directive in the sequence is deny all, then all further allow directives have no effect.

In this example, the subnet 192.168.1.1/24 is allowed access, with the exception of 192.168.1.2.

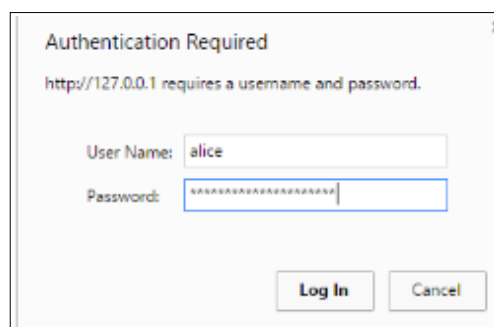
The 2001:0db8::/32 range of IPv6 addresses is also allowed, and access to any other IP addresses is denied.

### • [6.6.2] Basic Authentication

When setting up a web server, there are often sections of the site that you wish to restrict access to. Web applications often provide their own authentication and authorization methods, but the web server itself can be used to restrict access if these are inadequate or unavailable.

Consider the following information inside this link:

<https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-http-basic-authentication/>



*Image 30 { nginx Basic Auth }*

**However**, since we are using **Flask** we are required to do the following while baring in mind the main target folder with the file to be modified is `/usr/lib/python3/dist-packages/gitlab_tools/` This application already has a login manager. Its authentication is based on GitLab account. However that does not stop from filtering accounts that you wish to have access to this tool. For further understanding the layers of the login process please visit : <https://flask-login.readthedocs.io/en/latest/>

The next step is to only allow connections to specific accounts. That will require to modify the database and to add such constrains. After the client's authentication resourcing GitLab App Auth, we may check if it is one of those who are allowed or blocked from using this application.

The target file is `/usr/lib/python3/dist-packages/gitlab_tools/models/gitlab_tools.py` and the *block* to be modified is `"class User(BaseTable)"`.

Add `"is_allowed= db.Column(db.Boolean, default=False)"`

Modify: `"def is_authenticated(self)"` instead of returning `"True"` to return `"self.is_allowed"`.

With this, we assume nobody is allowed to use it after the registration. The admin will later manually change this inside the database to allow this access to users chosen by the owner.

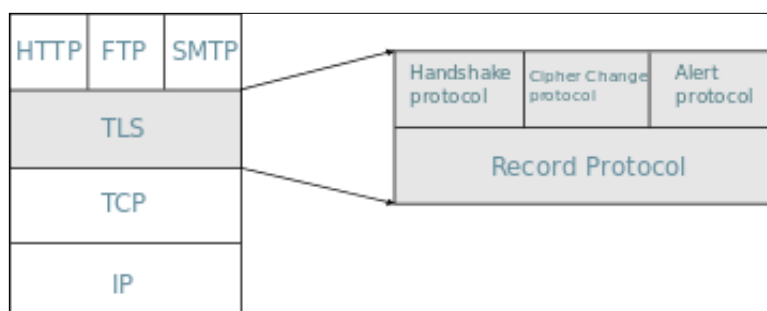
Example:

**UPDATE user SET is\_allowed = True WHERE gitlab\_id = X;**

### • [6.6.3] SSL

Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser, or a mail server and a mail client (e.g., Outlook).

SSL allows sensitive information such as credit card numbers, social security numbers, and login credentials to be transmitted securely. Normally, data sent between browsers and web servers is sent in plain text—leaving you vulnerable to eavesdropping. If an attacker is able to intercept all data being sent between a browser and a web server, they can see and use that information.



*Image 31 { IP Protocol Framework }*

## • [6.6.4] DNSSEC

The Domain Name System Security Extensions (DNSSEC) is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol (IP) networks. It is a set of extensions to DNS which provide to DNS clients (resolvers) cryptographic authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.

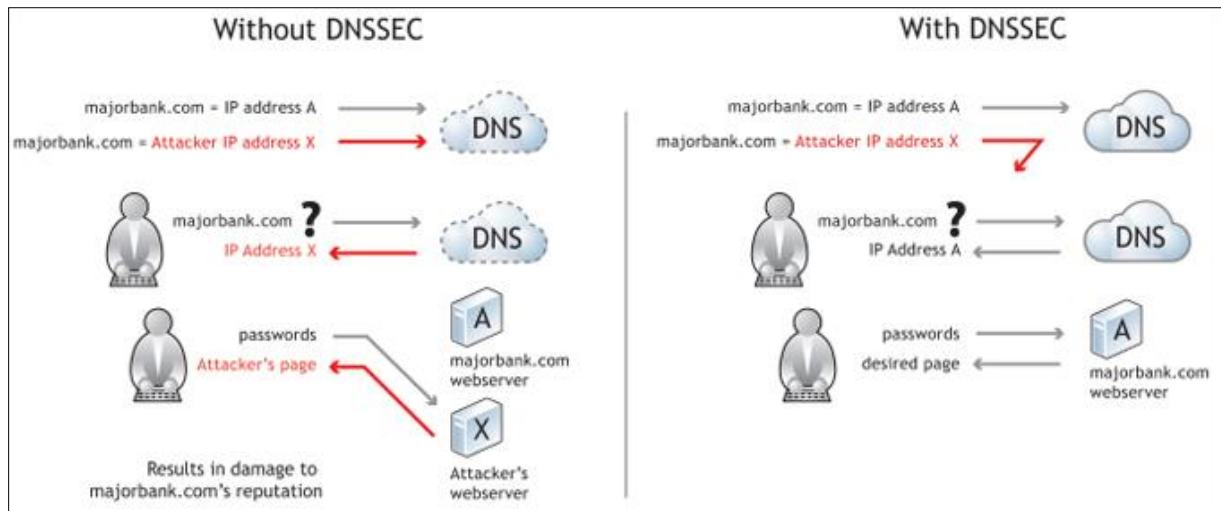


Image 32 { DNSSEC Protection }

### • [6.6.4.1] Enabling DNSSEC

Each Domain Name Registrar has a specific DNSSEC configuration. Most of the cases they have their own tutorial to follow along, therefore varying considerably.

Since I strongly recommend using a CDN Edge, they too have tutorial it and providing the extra protection needed.

Consider the following:

<https://developers.cloudflare.com/registrar/cloudflare-registrar/dnssec/>

## • [6.6.5] TLS

Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network. The TLS protocol aims primarily to provide privacy and data integrity between two or more communicating computer applications. When secured by TLS, connections between a client (e.g., a web browser) and a server (e.g., wikipedia.org) should have one or more of the following properties:

- ❖ The connection is *private* (or *secure*) because symmetric cryptography is used to encrypt the data transmitted.
- ❖ The identity of the communicating parties can be *authenticated* using public-key cryptography.
- ❖ The connection is *reliable* because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

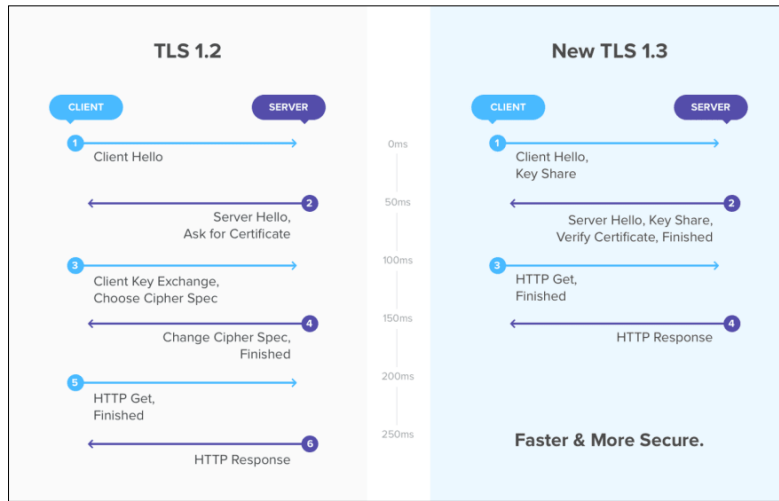


Image 33 { TLS versions 1.2 1.3 }

## moz://a SSL Configuration Generator

Image 34 { Mozilla SSL Configuration Generator }

<https://ssl-config.mozilla.org> (nginx, PostgreSQL, MySQL)

The best practice is to follow the link above and inside there adjust the values to your applications version.

If you choose to have (recommended) certificates on your web server, follow the next steps.



Image 36 { Let's Encrypt }

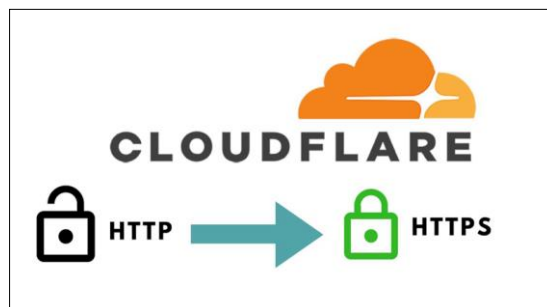


Image 35 { Cloudflare SSL }

There are many certificates providers. The most used is “letsencrypt”.

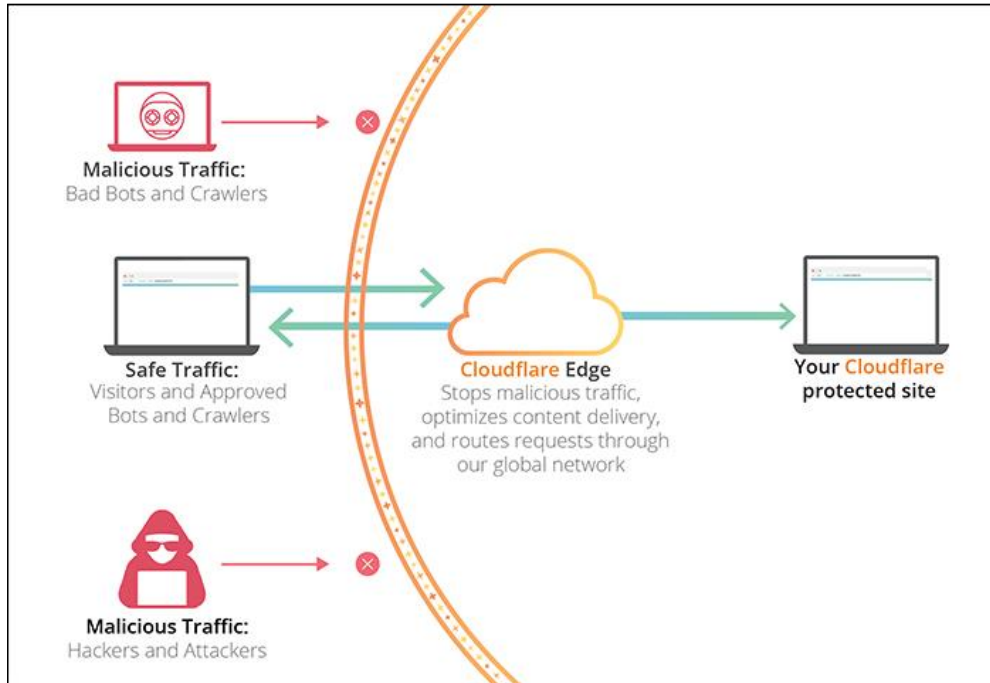
```
sudo add-apt-repository ppa:certbot/certbot
sudo apt update && sudo apt install python-certbot-nginx
sudo certbot --nginx -d gitlabtools.EXAMPLE.COM
fill with the desired and required information
«after filling it will take a few minutes»
sudo certbot renew --dry-run
```

There is a maximum requests per week using “letsencrypt”. If you will be using a CDN such as ‘Cloudflare’ (recommended) then you may also do this by following these instructions:

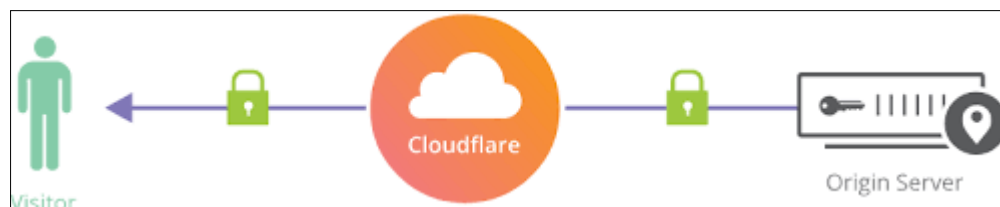
<https://www.howtoforge.com/tutorial/nginx-with-letsencrypt-ciphersuite/>  
Follow the steps of Cloudflare in SSL/TLS dashboard

- **[6.6.6] CDN Edge**

As more content delivery network (CDN) operators move applications and business processes to the cloud, CDN security is a growing concern. CDN security solutions must not only protect data from theft and loss, but ensure optimal CDN availability by fending off the many attacks that can diminish or interrupt network performance. The stakes are high – when poor CDN security prevent users from accessing content they want with the speed they expect, it can have a lasting negative affect on revenues and reputation for the CDN operator.



*Image 37 { Cloudflare CDN Edge }*



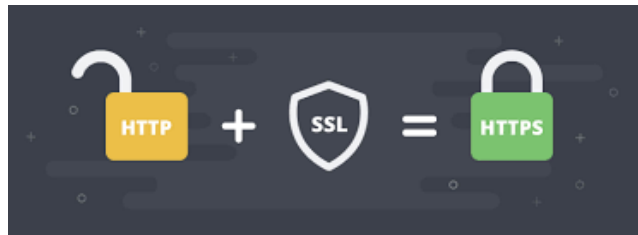
*Image 38 { Cloudflare CDN Edge }*

## • [6.6.7] HTTP-S/x

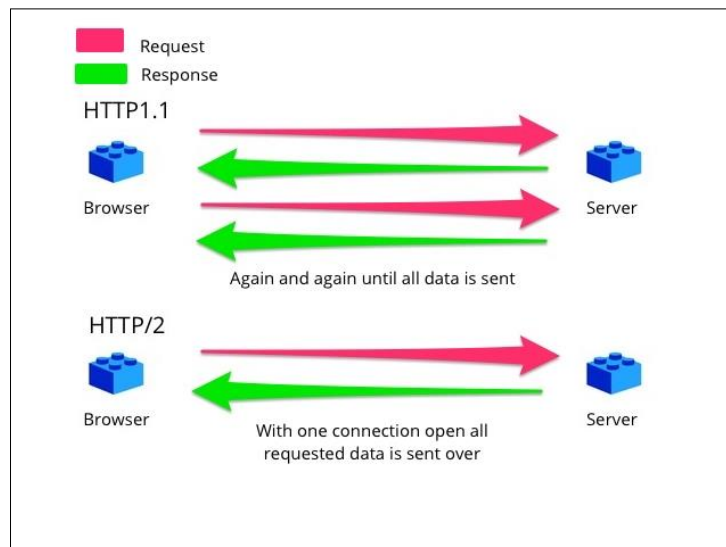
**Hypertext Transfer Protocol Secure (HTTPS)** is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL). The protocol is therefore also referred to as **HTTP over TLS**, or **HTTP over SSL**.

The principal motivations for HTTPS are authentication of the accessed website, and protection of the privacy and integrity of the exchanged data while in transit. It protects against man-in-the-middle attacks, and the bidirectional encryption of communications between a client and server protects the communications against eavesdropping and tampering.

In practice, this provides a reasonable assurance that one is communicating with the intended website without interference from attackers.



*Image 39 { HTTP + SSL = HTTPS }*



*Image 40 { HTTP/2 }*



- [6.6.7.1] Quic

It was announced the technology preview of QUIC+HTTP/3 for NGINX at a special open source repository. This is pre-release software, based on the IETF QUIC draft and is maintained in a development branch, isolated from the stable and mainline branches. The release is the culmination of several months of initial development, and is now ready for interoperability testing, feedback, and code contributions.

HTTP/3 is based on the QUIC transport protocol, which is designed specifically to support multiplexed connections without depending on a single TCP connection. QUIC instead uses UDP as the low-level transport mechanism for moving packets between client and server, and implements a reliable connection upon which HTTP requests are made. Notably, QUIC also incorporates TLS as an integral component, not as an additional layer as with HTTP/1.1 and HTTP/2.

The goal of QUIC is to provide a high-performance, high-reliability, high-security transport protocol for HTTP/3 (although QUIC is also suitable for non-HTTP traffic).

Semantically speaking, HTTP/3 itself is very similar to HTTP/2.

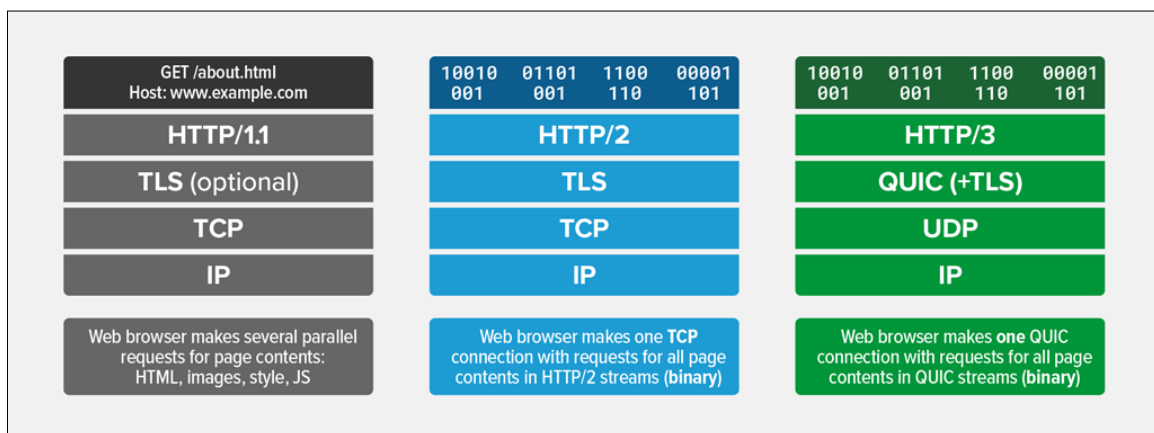


Image 41 { HTTP versions comparison 1 }

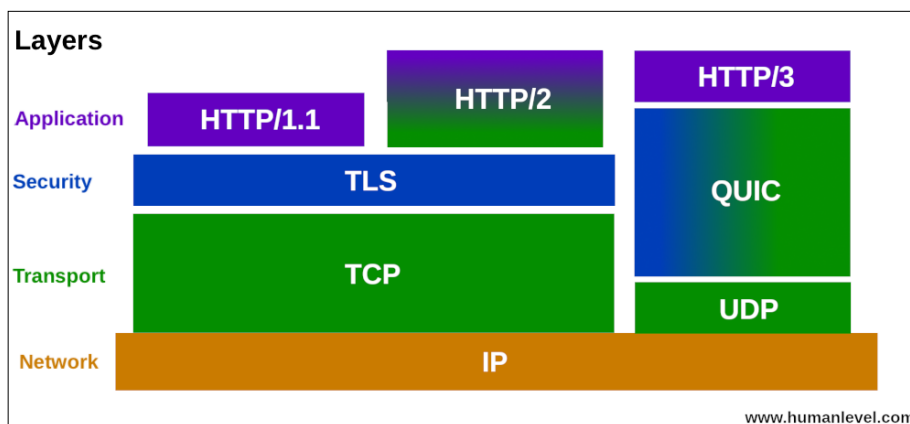


Image 42 { HTTP versions comparison 2 }

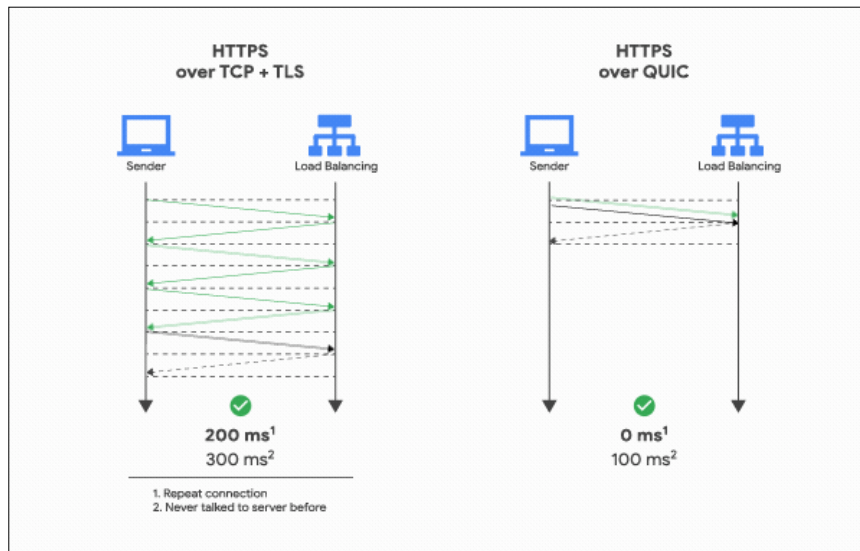


Image 43 { TCP/TLS VS QUIC }

If you wish to use this version, bare in mind that you need to update the nginx version ( $\geq 1.19.x$ ) and modify configurations.

```
server {
    listen 443 ssl;                # TCP listener for HTTP/1.1
    listen 443 http3 reuseport;    # UDP listener for QUIC+HTTP/3

    ssl_protocols      TLSv1.3; # QUIC requires TLS 1.3

    ssl_certificate     ssl/www.example.com.crt;
    ssl_certificate_key ssl/www.example.com.key;

    add_header Alt-Svc 'quic=":443"'; # Advertise that QUIC is available
    add_header QUIC-Status $quic;    # Sent when QUIC was used
}
```

Here are the official links that allow you to do and test it:

<https://quic.nginx.org/readme.html> && <https://http3check.net/>

As for 08/August/2020, states the following:

#### Not (yet) supported features:

- Version negotiation
- ECN, Congestion control and friends as specified in quic-recovery [5]
- A connection with the spin bit succeeds and the bit is spinning
- Structured Logging
- QUIC recovery (proper congestion and flow control)
- NAT Rebinding
- Address Mobility
- HTTP/3 trailers

- [6.6.7.2] Testing the Security

<https://www.ssllabs.com/ssltest>

This free online service performs a deep analysis of the configuration of any SSL web server on the public Internet.

For perfect score and safety, take note of the information provided by the links beneath:

<https://foxontherock.com/fix-dns-caa-issue-ssl-labs-test/>

<https://michael.lustfield.net/nginx/getting-a-perfect-ssl-labs-score>

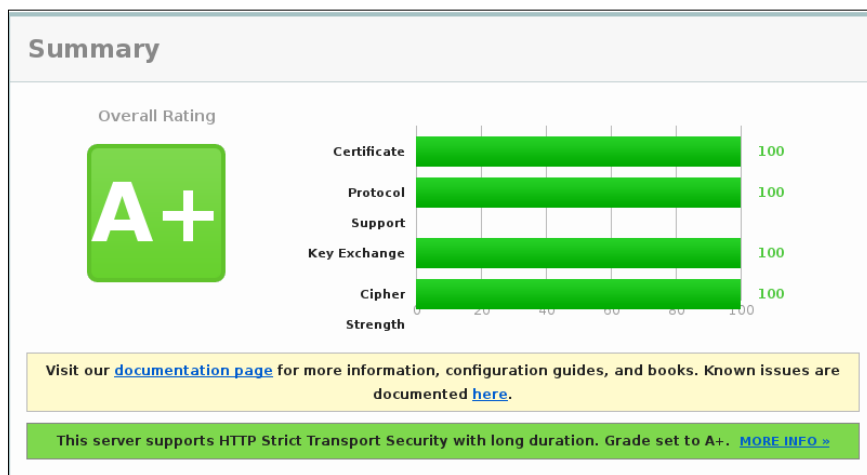


Image 44 { sslabs perfect score test }

	Server	Test time	Grade
1	<a href="#">2606:4700:3037:0:0:6818:74c6</a> Ready	Wed, 05 Aug 2020 14:37:20 UTC Duration: 43.399 sec	A+
2	<a href="#">2606:4700:3037:0:0:ac43:ad69</a> Ready	Wed, 05 Aug 2020 14:38:03 UTC Duration: 42.833 sec	A+
3	<a href="#">2606:4700:3033:0:0:6818:75c6</a> Ready	Wed, 05 Aug 2020 14:38:46 UTC Duration: 43.600 sec	A+
4	<a href="#">172.67.173.105</a> Ready	Wed, 05 Aug 2020 14:39:30 UTC Duration: 43.397 sec	A+
5	<a href="#">104.24.116.198</a> Ready	Wed, 05 Aug 2020 14:40:13 UTC Duration: 45.346 sec	A+
6	<a href="#">104.24.117.198</a> Ready	Wed, 05 Aug 2020 14:40:58 UTC Duration: 45.674 sec	A+

Image 45 { sslabs during testing }

- **[6.7] Monitoring Dashboard Tools**

“Telemetry” is the mechanism for collecting data from monitoring. Telemetry can use agents that are installed in the deployment environments, an SDK that relies on markers inserted into source code, server logging, or a combination of these. Typically, telemetry will distinguish between the data pipeline optimized for real-time alerting and dashboards and higher-volume data needed for troubleshooting or usage analytics.

- **[6.7.1] Amplify nginx**

Nginx amplify is a collection of useful tools for extensively monitoring a open source Nginx web server and NGINX Plus. With NGINX Amplify you can monitor performance, keep track of systems running Nginx and enables for practically examining and fixing problems associated with running and scaling web applications.

It can be used to visualize and determine a Nginx web server performance bottlenecks, overloaded servers, or potential DDoS attacks; enhance and optimize Nginx performance with intelligent advice and recommendations.

In addition, it can notify you when something is wrong with the any of your application setup, and it also serves as a web application capacity and performance planner.

This software provides a free tier as well.

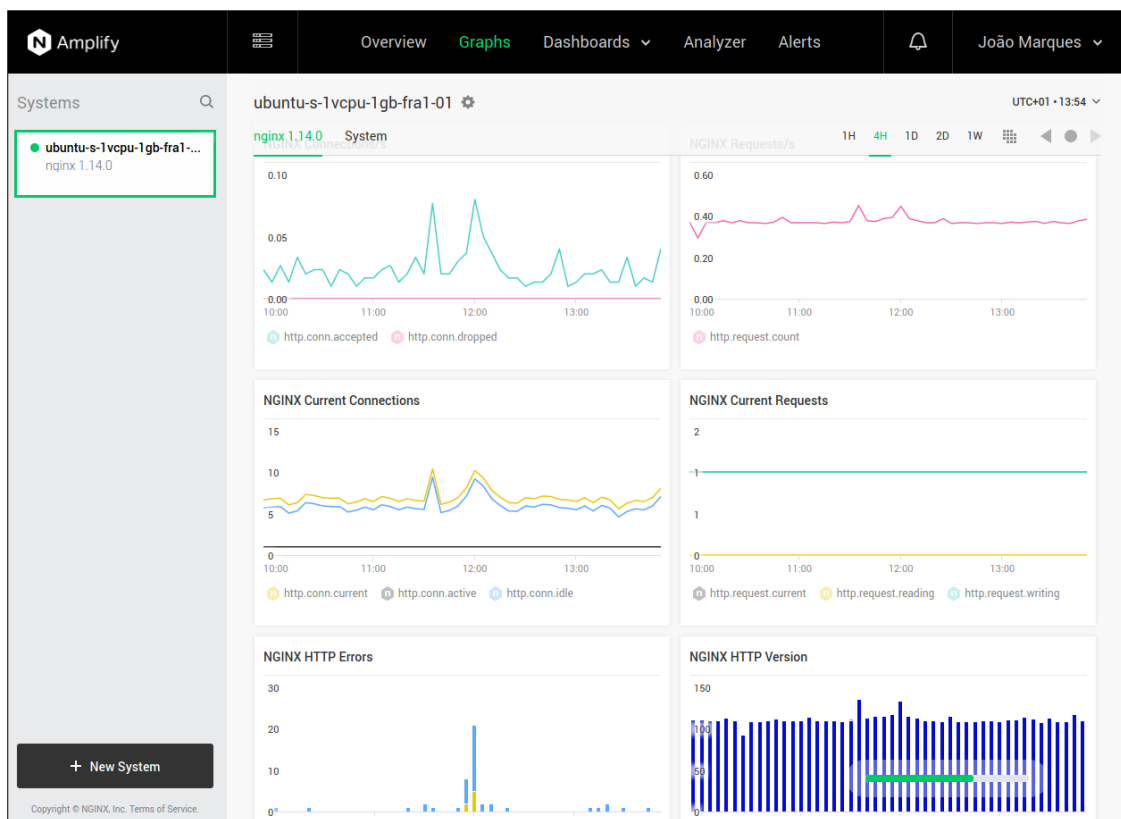


Image 46 { Amplify nginx graph }

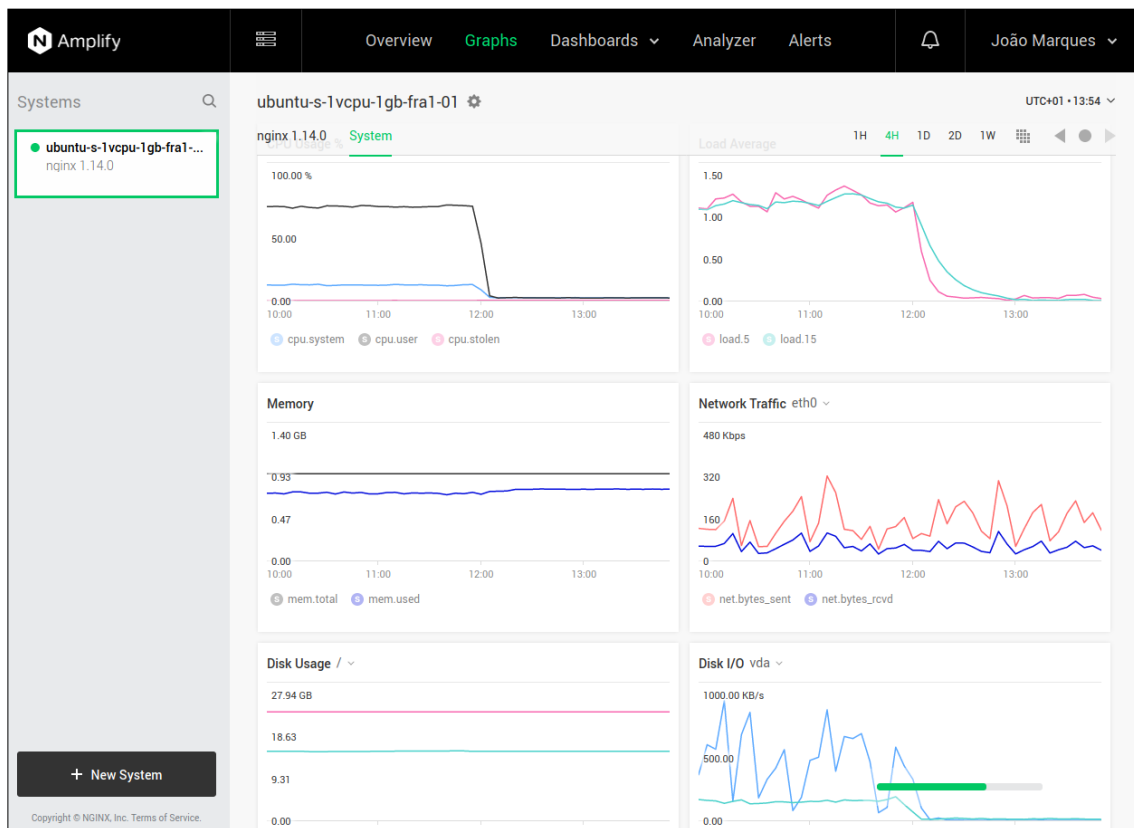


Image 47 { Amplify system graph }

There are more software with these capabilities. So if you can find more alternatives here:  
<https://stackshare.io/nginx-amplify/alternatives>

## • [6.7.2] CDN Edge

A CDN edge server is a computer that exists at the logical extreme or “edge” of a network. An edge server often serves as the connection between separate networks. A primary purpose of a CDN edge server is to store content as close as possible to a requesting client machine, thereby reducing latency and improving page load times.

They usually also provide tools that show website traffic, visitor analytics, monitor threats and logging. My personal recommendation would be [Cloudflare](#).

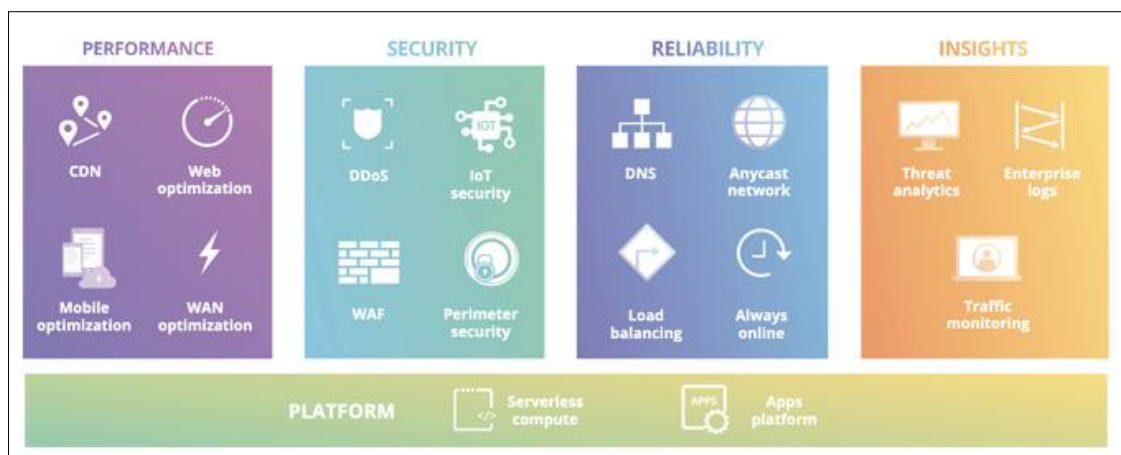


Image 48 { Cloudflare services }

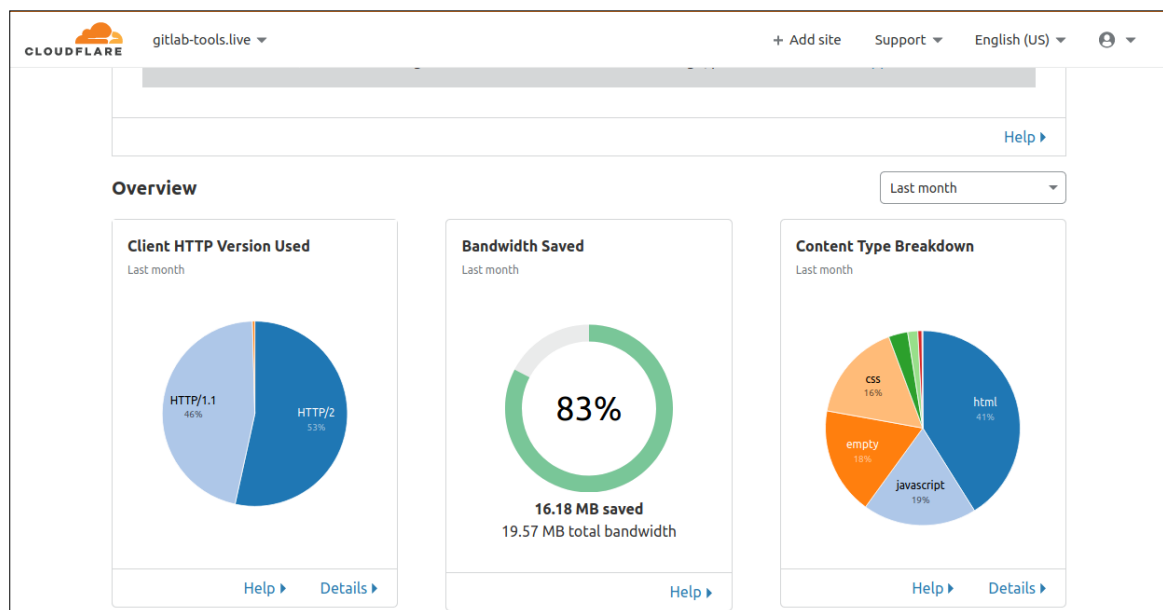


Image 49 { Cloudflare dashboard 1 }

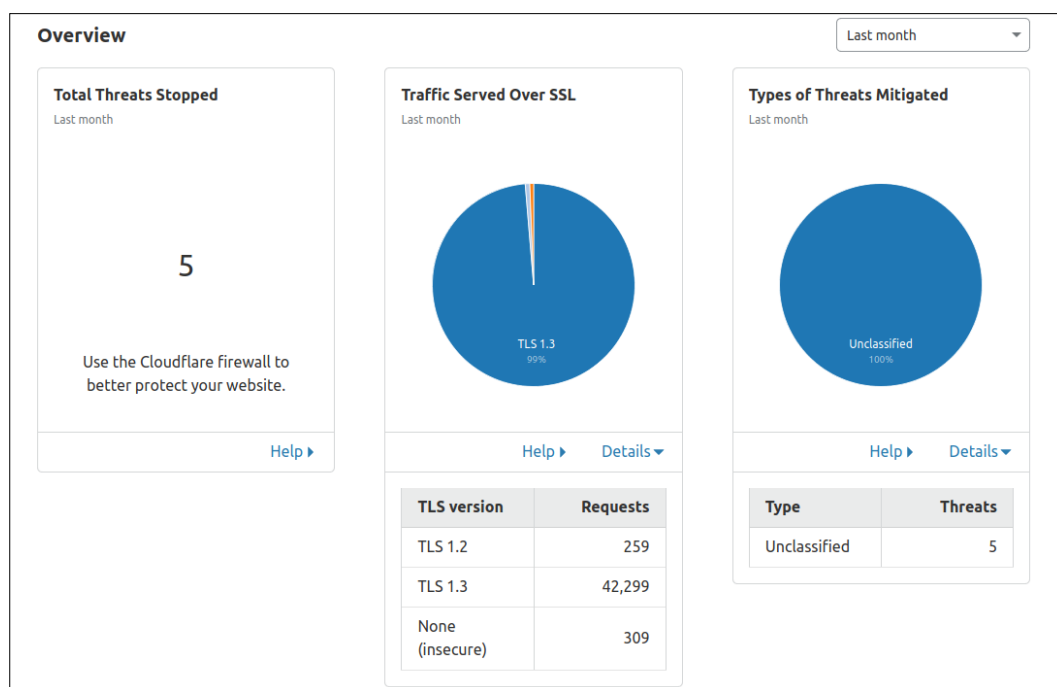


Image 50 { Cloudflare dashboard 2 }

ALTERNATIVES (FREE TIER – CDN):



Image 52 { Imperva }



Image 53 { Amazon CDN }



Image 51 { Azure CDN }

### • [6.7.3] Database

Database monitoring is the tracking of database performance and resources in order to create and maintain a high performance and highly available application infrastructure. For example, database monitoring include:

- ❖ Query details (top CPU, slow running, and most frequent)
- ❖ Session details (current user connections and locks)
- ❖ Scheduled jobs
- ❖ Replication details
- ❖ Database performance (buffer, cache, connection, lock, and latch)

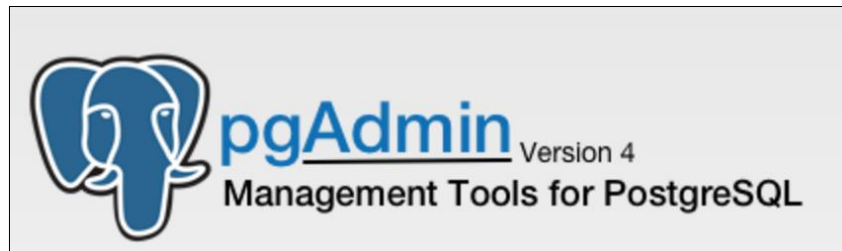


Image 54 { pgAdmin }

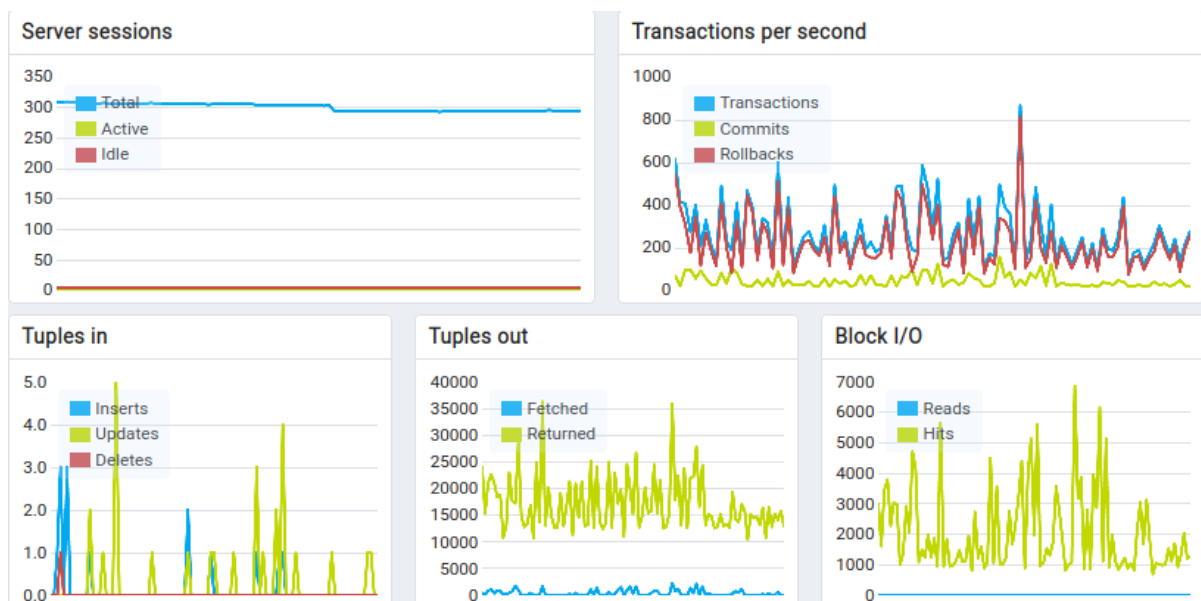


Image 55 { pgAdmin Dashboard }

#### Recommended database-monitoring tools:

- ❖ MySQL → [Workbench](#) OR [Percona Monitoring and Management \(PMM\)](#)
- ❖ SQLite → <https://github.com/vbalyasny/sqlite-monitor>
- ❖ PostgreSQL → [pgAdmin](#)

- **[6.7.4] Flask**

Software teams are able to trace performance issues back to a poor performing API call as well as surface all related code errors. Engineering Managers and Developers now have a single tool to optimize the performance of their code and deliver fast customer experiences.

See local variables in the stack for prod errors, just like in your dev environment.

Introspect more deeply into the runtime and jump into the frame to get additional data for any local variable. Filter and group Flask exceptions intuitively to eliminate noise.

Sentry is an open-source error tracking service which boosts developer efficiency and facilitates continuous iteration. The action sends release notifications to a self-hosted instance of Sentry, helping developers to monitor website performance and fix crashes in real-time.

The creator used this tool for monitoring, has you can clearly see the data inside the database. He resorted to use it as local server.

<https://github.com/getsentry/onpremise/>



*Image 56 { Sentry }*



## • [7] Installation & Configuration

Having you reach this level and have everything prepared, the installation will be very fast. During it will be requested all the credentials. It is going to create a new user (let's keep it by default: gitlab-tools).

Follow the author's Setup Instructions After it, you will need to configure the nginx web file as mentioned before while taking in mind the default configuration:

*« Author's nginx default web page configuration file »*

```
server {  
    listen 80;  
    listen [::]:80;  
    server_name gitlab-tools.example.com;  
  
    root /usr/lib/python3/dist-packages/gitlab_tools;  
}
```

If chosen to run uwsgi, reconfigure the same file accordingly:

```
#root /usr/lib/python3/dist-packages/gitlab_tools;  
location/ {  
    uwsgi_pass unix:///tmp/gitlab-tools.sock  
    include uwsgi_params;  
}
```

Now link the file by running:

```
$ ln -s /etc/uwsgi/apps-available/gitlab-tools.example.com.ini /etc/uwsgi/apps-enabled/
```

If you are using uwsgi you are required to take this action to avoid conflicts with nginx  
Restart the engine by doing:

```
$ systemctl restart uwsgi
```

Now you should have the socket file created at `/tmp/gitlab-tools.sock`  
Check that by running:

```
$ file /tmp/gitlab-tools.sock
```

You can check the status of your server by doing

```
$ systemctl status gitlab-tools_celery*
```

If everything is okay, it should look like this:

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/usr/lib/python3/dist-packages/gitlab_tools# systemctl status gitlab-tools_celery*
● gitlab-tools_celerybeat.service - GitLab Tools Service Celery beat
  Loaded: loaded (/etc/systemd/system/gitlab-tools_celerybeat.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2020-08-04 17:03:18 UTC; 18min ago
  Main PID: 9763 (gitlab-tools)
  Tasks: 1 (limit: 1151)
  CGroup: /system.slice/gitlab-tools_celerybeat.service
          └─9763 /usr/bin/python3 /usr/bin/gitlab-tools celerybeat --config_prod --pid=/home/gitlab-tools/celerybeat.pid --schedule=/home/gitlab-tools/celery_beat.db

Aug 04 17:03:18 ubuntu-s-1vcpu-1gb-fra1-01 systemd[1]: Started GitLab Tools Service Celery beat.

● gitlab-tools_celeryworker.service - GitLab Tools Service Celery worker
  Loaded: loaded (/etc/systemd/system/gitlab-tools_celeryworker.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2020-08-04 17:03:13 UTC; 18min ago
  Main PID: 9766 (gitlab-tools)
  Tasks: 2 (limit: 1151)
  CGroup: /system.slice/gitlab-tools_celeryworker.service
          └─9766 /usr/bin/python3 /usr/bin/gitlab-tools celeryworker --config_prod
             └─9742 /usr/bin/python3 /usr/bin/gitlab-tools celeryworker --config_prod

Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: - ** ----- .> transport: amqp://quest:**@127.0.0.1:5672//
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: - ** ----- .> results: postgres://bwihmujrgnqaus:**@ec2-52-212-157-46.eu-west-1.compute.amazonaws.com:5432/de7ktgguoulvdm
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: - *** --- * --- .> concurrency: {min=1, max=10} (prefork)
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: -- ***** ---- .> task events: OFF (enable -E to monitor tasks in this worker)
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: --- ***** ----
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: ----- [queues]
Aug 04 17:15:09 ubuntu-s-1vcpu-1gb-fra1-01 gitlab-tools[9766]: .> gitlab_tools exchange=gitlab_tools(direct) key=gitlab_tools
```

*Image 57 { GitLab-Tools status }*

At this point, you should apply the security that is mentioned above (TLS, certificates and so on).

**This tool will act like an intermediary, keeping the data and checking if any repository listed (pull/push mirror) was updated. So be aware about the actual size you may need for your operations.**

As mentioned in DB chapter, the directory is `/home/gitlab-tools/repositories` and if you wish to know how much space was consume run the following command:

```
$ du -sh /home/gitlab-tools/repositories
```

**In order to use this tool you will need to set up the GitLab API configuration:**

1. Go to <https://gitlab.com/profile/applications>
2. Add a name (Ex: GitLab-Tools)
3. Put the corresponding redirect URI (Ex: `https://gitlab-tools.EXAMPLE.COM/sign/in/do`)
4. Select the scope “API”
5. Click save application
6. Save the provided “Application ID” and “Secret”

**In order to use pull from the mirrored repository tool you will need to add the public key, which is present in the home page.**

Examples:

- ❖ If you want to pull from GitHub you need to add a public key inside that repository
- ❖ If you want to push to GitHub you need to add a public key inside that repository
- ❖ If you want to pull from a GitLab repository you need to add a public key inside that repository

**You will need also to create a group(s) inside GitLab.**

Just go to <https://gitlab.com/groups/new> and create a new one (example: “Mirrors”). When you are pulling a repository to GitLab it will appear on the webpage inside the list (just type “mirrors” and it will show).

Having it all prepared, here is the *author's Setup Instructions*:

After successful installation you will need to run setup to configure your installation:

```
$ gitlab-tools setup
```

This will start simple setup utility where you can/must configure

1. **gitlab-tools user** (default: gitlab-tools, should be created after install)
2. **Database type to use** (PostgreSQL is recommended database to use (you can use other database types only on your own risk)
  - a. **Database host** (PostgreSQL, MySQL) or path (Only SQLite)
  - b. **Database name** (PostgreSQL, MySQL)
  - c. **Database user** (PostgreSQL, MySQL)
  - d. **Database password** (PostgreSQL, MySQL)
3. **Webserver configuration** (Host and port makes sense only when using integrated web server controlled by gitlab-tools service)
4. **Server name** (eg.: <https://gitlab-tools.example.com> or IP:PORT when using integrated webserver)
5. **GitLab API configuration** (you can follow this guide till point 7. <https://docs.gitlab.com/ee/integration/gitlab.html>) redirect url is <https://gitlab-tools.example.com/sign/in/do> where example.com is your server domain
  - a. **Gitlab APP ID**: Application Id
  - b. **Gitlab APP SECRET**: Secret
6. **Save new configuration ?** -> y (this will create config file in /etc/gitlab-tools/config.yml)
7. **Recreate database ?** -> y (this will create new empty database)
8. **Restart services to load new configuration ?** -> y this will restart all gitlab-tools services:
  - a. **gitlab-tools**: Controlling integrated webserver, disable this one if you want to use uwsgi or so.
  - b. **gitlab-tools\_celeryworker**: Controlling background workers, must be enabled and running to perform mirroring
  - c. **gitlab-tools\_celerybeat**: Controlling celery scheduler

This creates/updates config file in `/etc/gitlab-tools/config.yml`, you can modify this file manually  
After this you should have gitlab-tools running on your "server name"

### « Author's Pull Mirror Configuration Guide »

Pull mirror pulls changes from remote GIT repository (GitHub, etc.) to your GitlabCE installation  
To create your first pull mirror follow these steps

1. Click on "Pull mirror" in main menu
2. Click on "Add +"
3. Fill in form required fields:
  - a. Fill in "Project mirror" that is GIT repo URI you want to mirror (eg. [git@github.com](mailto:git@github.com):Salamek/gitlab-tools.git)
  - b. Fill in "Project name" that is name of project that will be created in your GitLab CE installation
  - c. Fill in "Search for group" select group in your GitLab CE installation where this mirrored project will be created. It can be separated group like "Mirrors" or any other group or subgroup.
4. Click on "Save"
5. Now you should see newly created mirror in "Pull mirror" section
6. To get actual mirroring work, we must register push web hook to mirrored repository and allow your GitLab-Tools user to access that repository via configured "Project mirror" URI.

### « Author's Push Mirror Configuration Guide »

Push mirror pushes changes from your GitlabCE installation to remote GIT repository (GitHub, etc.) To create your first push mirror follow these steps

1. Click on "Push mirror" in main menu
2. Click on "Add +"
3. Fill in form required fields:
  - a. Fill in "Project mirror" that is GIT repo URI you want to push to (eg. [git@github.com](mailto:git@github.com):Salamek/gitlab-tools.git), it must exists and be empty!
  - b. Fill in "Search for project" select project in your GitLab CE installation you want to mirror.
4. Click on "Save"
5. Now you should see newly created mirror in "Push mirror" section
6. To get actual mirroring work you must allow your GitLab-Tools user to access that repository via configured "Project mirror" URI:

#### **How to allow access to mirrored repository:**

For SSH access you will need GitLab-Tools user public key, you can find it on "Home" page. If you use SSH, make sure your deploy key on target repository has writable permission.

## • [8] Logs

As it was mentioned on the “Monitoring” chapter, logging data is quite important to both monitoring as to understanding and fixing issues.

Here are some paths to take into consideration.

### • [8.1] ufw

UFW, or uncomplicated firewall, is a frontend for managing firewall rules in Arch Linux, Debian or Ubuntu. It logs all the connections attempted or made and its contents.

You might notice there is increasingly more and more attacks. Therefore, it is a good idea to have a CDN backing it up. This log is located at `/var/log/ufw.log`

### • [8.2] nginx

#### • [8.2.1] Access

NGINX saves all access calls at `/var/log/nginx/access.log`

#### • [8.2.2] Error

NGINX also saves all errors it encountered at `/var/log/nginx/error.log`

### • [8.3] uwsgi

Uwsgi saves all connections made, contents, actions done by its workers and its results. Also very good to monitor the sequence of requests. (The database only saves error or success warnings).

This log is located at `/var/log/uwsgi/app/gitlab-tools.EXAMPLE.COM.log`

### • [8.4] app\_crash

If for some reason the application stops and crashes really bad, inside it might be the reason why.

This log is located at `/var/crash/___usr_bin_gitlab-tools.EXAMPLE.COM_x.crash`

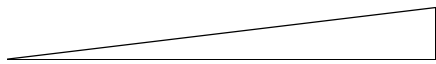
## WHO? [D]

### • [9] Target Audience

Singular developers and Team developers.  
Anyone who is familiar with GitLab.  
Anyone wanting to learn.

## WHEN? [E]

### • [10] Time To Deploy

	<i>Basic</i>	<i>Advanced</i> <sup>*recommended</sup>
<i>Local</i>	~ 30 min	~ 60 min
<i>Remote</i>	~ 60 min	~ 120 min

*Basic* meaning deploying from zero not caring so much about security and monitoring.  
*Advanced* meaning deploying from zero with all security, monitoring and configuration

This will depend a lot on your expertise and also if you already have something pre-built.  
In the best scenario where you already have a database, nginx/uwsgi working then you may only need to install and configure one or two files (site-available/apps-available), restart the engines and there you go. In less then 5 minutes it is done. The more advanced more time it will take.

## COST? [F]

### • [11] Total Cost of Ownership (TCO)

			DataBase			Domain			Monitoring			TOOL
	Local	Remote	Local	Cloud	Premise	Local	Cloud	Premise	Local	Cloud	Premise	
FREE	✓	✓ *1	✓	✓ *2	✓	✓	✓ *3	✓	✓	✓ *4	✓	✓

This will depend on which components you will choose. It is proven you can deploy and combine all components and still be totally cost free

✓ \*1 - Azure, AWS, GCP, DigitalOcean, etc (free 2 months trials or 12 months for student credit) [IaaS]

✓ \*2 - Many free tiers [SaaS]

✓ \*3 - “dot.tk” (free domains website search website)

✓ \*4 - Free Monitoring tools provided [DB, CDN, nginx, System, Flask]

## Errors & Fixes [G]

### • [12] Time To Detect and Time To Mitigate (TTD, TTM)

✖ **Error lines received while fetching: error:** failed to push some references to a 'git@gitlab.com\_X:GroupY/external\_repository\_name.git'

- In general, *you* cannot fix this. Only the owner of the repository can do that, because a pre-receive *'hook'* is something *they* control and you don't.
- Solution: <https://stackoverflow.com/questions/24114676/git-error-failed-to-push-some-refs-to-remote>

✖ During the writing of this document it was noticed the online database provider changed both host, username and password. Even though all data was not lost this was to me an unpredictable behaviour. Therefore leading to crashes and presentation of 500-X HTML errors. Be aware of such notifications by your provider and set such notifications to redirect towards your e-mail.

- Once noticed the fix is quite simple: Just update it with the newest values:

```
sudo nano /etc/gitlab-tools/config.yml
```

Restart the engine:

✚ if you are using *'uwsgi'*

```
$ sudo systemctl restart uwsgi
```

✚ if you are **not** using *'uwsgi'*

```
$ sudo systemctl restart nginx
```

✖ If the default value of `server_names_hash_bucket_size` used at the server is not enough, means nginx complained with could not build the `server_names_hash`, you should increase `server_names_hash_bucket_size`: 32, the directive value should be increased to the next power of two (e.g. in this case to 64).

- To do that you should uncomment that line inside the file `"/etc/nginx/nginx.conf"`.
- My personal recommendation is to leave the value as `'64'`

✖ To upload designs, you will need to enable LFS. More information:

[https://gitlab.com/help/user/project/issues/design\\_management#requirements](https://gitlab.com/help/user/project/issues/design_management#requirements)

## Credits [H]

### • 13 Credits

This work was based on:

<https://github.com/Salamek/gitlab-tools> (Original Source)

developed by [Adam Schubert](#) aka [Salamek](#)

Please consider of sponsoring the creator:

<https://www.patreon.com/salamek/creators>

Special thanks to [Eurotux,SA](#) for welcoming me into the company and in particular to [Bruno Costa](#) which guide me and helped me throughout this process.

All of images contain raw metadata, which were obtain using google image search engine.

This tool was deployed using [GitHub Student Pack](#) in many instances ([DigitalOcean.com](#), [Names.com](#), etc).

Another website, which was incredibly useful with full of resources that I also recommend, is: <https://free-for.dev/>



# Image Index

Image 1: Index Page (Sign in with GitLab Account) .....	2
Image 2{ Main Page (public key will be used) } .....	2
Image 3{ Pull Mirrors Page (Push Mirrors Page is similar) } .....	2
Image 4{ GitHub Webhook Example } .....	2
Image 6{ New pull/push mirror Page } .....	2
Image 5{ Fingerprints Page } .....	2
Image 7 {SVN vs Git} .....	2
Image 8 { Git Mirror example 1 } .....	2
Image 9 { Git Mirror example 2 } .....	2
Image 10 { Git -Svn Mirror } .....	2
Image 11 { GitLab Pipeline example } .....	2
Image 12 { Comparison between service features } .....	2
Image 13 {Database Log example } .....	2
Image 14 { Comparison between DB } .....	2
Image 15 { Detailed Comparison between DB } .....	2
Image 16 { HelioHost } .....	2
Image 17 { ElephantSQL } .....	2
Image 18 { Google Cloud SQL } .....	2
Image 19 { Amazon RDS } .....	2
Image 20 { Azure PostgreSQL } .....	2
Image 21 { Heroku } .....	2
Image 23 { GoogieHost } .....	2
Image 22 { GigaRocket } .....	2
Image 24 { FreeHosting } .....	2
Image 25 { Computing Location } .....	2
Image 26 { nginx architecture } .....	2
Image 27 { nginx description } .....	2
Image 28 { Server architecture details 1 } .....	2
Image 29 { Server architecture details 2 } .....	2
Image 30 { nginx Basic Auth } .....	2
Image 31 { IP Protocol Framework } .....	2
Image 33 { DNSSEC Protection } .....	2
Image 34 { TLS versions 1.2 1.3 } .....	2
Image 35 { Mozilla SSL Configuration Generator } .....	2
Image 37 { Cloudflare SSL } .....	2
Image 36 { Let's Encrypt } .....	2
Image 38 { Cloudflare CDN Edge } .....	2
Image 39 { Cloudflare CDN Edge } .....	2
Image 40 { HTTP + SSL = HTTPS } .....	2
Image 41 { HTTP/2 } .....	2
Image 42 { HTTP versions comparison 1 } .....	2
Image 43 { HTTP versions comparison 2 } .....	2
Image 44 { TCP/TLS VS QUIC } .....	2
Image 45 { sslabs perfect score test } .....	2
Image 46 { sslabs during testing } .....	2
Image 47 { Amplify nginx graph } .....	2
Image 48 { Amplify system graph } .....	2
Image 49 { Cloudflare services } .....	2
Image 50 { Cloudflare dashboard 1 } .....	2
Image 51 { Cloudflare dashboard 2 } .....	2
Image 54 { Azure CDN } .....	2
Image 52 { Imperva } .....	2
Image 53 { Amazon CDN } .....	2
Image 55 { pgAdmin } .....	2
Image 56 { pgAdmin Dashboard } .....	2
Image 57 { Sentry } .....	2
Image 58 { GitLab-Tools status } .....	2