

Maxime J.



Projet 10 : Déploiement de votre application sur un serveur comme un pro ! / Project 10 : Deploying your application on a server like a pro !



✚ Document écrit expliquant notre démarche de création, les difficultés rencontrées et la manière dont nous les avons résolues. Lien vers notre board Trello inclus. Le document est en format PDF n'excède pas 2 pages A4 (espaces non inclus). Il est rédigé en anglais. / Written document explaining our approach for the creation, the difficulties encountered and the way we resolved them. Link to our Trello board included. The document is in PDF format and must not exceed two A4 pages (spaces non included). It is written in English.

✚ Lien Trello : <https://trello.com/b/V1PyNAIr/projet-10-d%C3%A9ployez-votre-application-sur-un-serveur-comme-un-pro> / Link to Trello : <https://trello.com/b/V1PyNAIr/projet-10-d%C3%A9ployez-votre-application-sur-un-serveur-comme-un-pro>

I. Generalities

Project number 10 is a continuation of project number 8 in which we had developed a web application for your startup Purbeurre. Until then, this application was deployed on the Heroku platform. The aim of this new project is to redeploy your application by no longer using a "Platform as a Service" (PaaS) as Heroku is, but via our own server using the "Infrastructure as a Service" model (IaaS) as well as acquire some knowledge in devOps development, systems and network administration...


II. Creation approach


We started by creating an account on Digital Ocean, as well as a droplet named Nutella. How we use Windows operating system, the SSH public key location is quite different from MacOS or Linux. We had to generate one using the `ssh-keygen -t rsa` command. We copy-paste this SSH public key in a file named `id_rsa.pub`. We use this one to configure Digital Ocean. Then, we created the firewall for your droplet and activated it. We logged into SSH via the root user, moreover we created a new user named "maxime". To connect in SSH with this new user, we have created a new `.ssh` folder, inside it a new file named "authorized_keys" in which we insert our SSH public key. We made sure everything works by typing in the terminal `ssh username@IP` (in our case `ssh maxime@157.230.19.106`). And also, we downloaded the application on the server by cloning your Github project with the command `git clone https://github.com/Maximedu13/plateforme-pour-amateurs-de-Nutella.git`. After having downloaded the useful libraries, it's necessary to create a new virtual environment and to migrate the database as well as to generate the static files of the project with the command `python3 manage.py collectstatic`. Once the data has been imported, we configured the Nginx http server to direct the traffic related to Django to the correct IP address and port. We downloaded Nginx on the server and we created its configuration file in "sites-available" with a symbolic link pointing to sites-enabled. As Nginx does not interpret python code, we need a Python http server. We chose Gunicorn. We launched the Gunicorn server and then we configured Supervisor to restart the server in case of failure. Once Supervisor is installed, we configured it by creating a `nutella-gunicorn.conf` configuration file. Inside this same file it is necessary to add the command to execute to start Gunicorn, define the username that has the right to execute the command and the path from which the command is launched. Then, we need to separate the environments, indeed the production settings are different from the development settings. While working locally, we removed `settings.py` to paste its content into `__init__.py`. This file will be the development settings. For the production settings, we pulled the project from


the local to the server and we have created a file at the same level named `production.py`. Moreover, we decided to use the tool Travis to manage continuous integration. We created a Travis account using our Github account, we activated Continuous integration for the project, and we created a `.travis.yml` file at the root of the project. We launched a build on Travis and checked that it does not stop and exit with 0. To meet your expectations of monitoring the server, we have used three tools. The first one is digital ocean monitoring. We have created alerts policy to monitor in particular the CPU, RAM, and bandwidth . The second one is Sentry which allows to visualize what is happening in the application. We have configured it in order to obtain more information on user requests and to anticipate possible code errors. The third one is NewRelic, which allows to monitor your application by collecting metric analyzes. Finally, we created a CRON Job which will deal with the update of the products retrieved from Open Food Facts once a week, while maintaining your users' food preferences.

III. Difficulties encountered


First difficulty : Quickly done, badly done, exited too quickly.

 By launching the command `maxime@nutella:~/D-ployez-votre-application-sur-un-serveur-comme-un-pro$ sudo supervisorctl status`, we encountered the error “FATAL Exited too quickly (process log may have details)”.

 It is very difficult to know where the error comes from, without consulting the associated logs. The command allowing us to consult the logs is: `sudo vi /var/log/supervisor/supervisord.log`.

 It turns out that we have defined an incorrect directory in the file `/etc/supervisor/conf.d/nutella-gunicorn.conf`. The correct syntax is `directory=/home/maxime/D-ployez-votre-application-sur-un-serveur-comme-un-pro/NUTELLA` and not `directory=/home/maxime/D-ployez-votre-application-sur-un-serveur-comme-un-pro/`

Second difficulty : 500 Internal Server Error.

 After separating the environments, we encountered a 500 error on the application index.

❓ We had no idea where this error could come from, however we noticed that only the templates linked to the views which were supposed to return data coming from the database were affected by this error.

✅ We hadn't yet configured Travis, NewRelic or Sentry. So, we made the choice to temporarily change the DEBUG to True. By restarting the processes, we immediately found where this error came from. It was an incorrect port number for the database connection (5432 instead of 5433). We immediately made the corresponding changes and redefined DEBUG to False.

Third difficulty: Travis configuration, or how to switch from Ruby to Python.

❌ After launching the build of your project on Travis, it returned as error “No Gemfile found, skipping bundle install”.

❓ Travis did not recognize your project as a Python project, moreover our team made a small careless mistake by naming the file `travis.yml` instead of `.travis.yml`.

✅ We renamed this file correctly, then placed it this time at the root of the project.

Fourth difficulty: Raven to be or not to be ? That is the question.

❌ For the configuration of Sentry, we hesitated between using the Raven library or Sentry-sdk.

❓ Different tutorials recommend using the Raven library, while the Sentry documentation recommends Sentry-sdk.

✅ We decided to adopt Sentry's approach, that is to use the `sentry-sdk` library because the Sentry documentation is up to date, while some tutorials recommending using Raven are obsolete.