

Rapport de Projet

DevOps -

Déploiement

Zeineb Chabchoub

22 Janvier 2021

TABLE DES MATIÈRES

Introduction Générale	5
Chapitre 1 : Contexte Général du projet	6
Introduction	6
Problème	7
Problématique : Fonctionnement traditionnel sans DevOps	7
Solution	7
DevOps : Évolution du terme DevOps	7
Objectif du projet : Projet de fin de semestre	7
Exigence du projet	8
Conclusion	8
Chapitre 2 : Étude préalable	11
Introduction	11
Concepts clés	11
Service	11
DevOps	11
Déploiement	11
Integration continue	11
Déploiement continue	11
Monitoring	11
Conclusion	11
Chapitre 3 : Conception de la solution	11
Introduction	11
Architecture de l'application	11
Services utilisées	11
Stratégie de déploiement	11
Conclusion	11
Chapitre 4 : Mise en oeuvre et validation	13
Conclusion Générale et Perspective	13

LISTE DES FIGURES

Figure 1. Architecture projet Django

Figure 2. Flux d'intégration et de déploiement continue

Figure 3 : Flux de transfert des métriques de surveillance

Figure 4. Déploiement sur Heroku

Figure 5 : Route /client -1-.

Figure 6 : Route /client -2-.

Figure 7 : Route /admin -1- .

Figure 8 : Route /admin -2- .

Figure 9 : Route /metrics.

Figure 10 : GitHub Actions

Figure 11 : Configuration Grafana Prometheus



Introduction Générale

Les logiciels passent par plusieurs étapes pour être créés. Il s'agit des étapes de définition, de développement et de maintenance. Dans le cadre du cycle de vie du logiciel, les projets informatiques sont toujours menés selon plusieurs phases bien définies à savoir: recueillir les besoins, la conception du produit, le développement et faire des tests avant de le mettre en production

En outre, les changements rapides des technologies et les besoins du marché ont modifié l'environnement des opérateurs d'éditeurs de logiciels. Les méthodes habituelles sont très strictes et ne tolèrent pas la flexibilité face aux changements des besoins. Quoique les entreprises sont confrontées au défi d'accélérer la livraison des logiciels.

Par ce fait, les clients attendent des réponses rapides à leurs exigences qui sont en constante évolution. De plus, les versions fréquentes sont essentielles pour que le client puisse fournir un retour continu. Des entreprises tel que Facebook déploient régulièrement des logiciels auprès de leurs clients. Bien que la fourniture continue de logiciels aux clients offre un avantage commercial par rapport à d'autres sociétés concurrentes, il n'est pas toujours aisé de le faire, vu les défis techniques et non techniques, doivent être d'abord surmontées.


Conséquemment, les entreprises doivent faire preuve de légèreté et d'agilité tout au long du cycle de développement des logiciels pour pouvoir relever ces défis.

La mise en production d'une application est un processus bien élaboré faisant intervenir, assez souvent, des équipes différentes, ce qui peut mener à des conflits tant que le but de chaque équipe diffère de celui de l'autre.

Il devient, donc, évident qu'il faut adopter une approche qui permet d'unifier le processus de développement et de production afin d'éviter tous les problèmes précédemment cités.

Le développement logiciel agile est dans plusieurs entreprises en raison de ses nombreux avantages, tels que la réduction du temps de développement, l'augmentation du taux de réussite du projet, la réduction des coûts de développement et la satisfaction accrue de la clientèle et qui permet d'assurer des bons suivis.

De ce fait, la notion de DevOps répond à l'ensemble des problématiques précédentes. Il s'agit d'une approche qui se base sur la synergie entre la partie opérationnelle et la partie production de plus l'alignement de l'ensemble des équipes du système d'information sur un objectif commun



a pour effet de réduire les conflits entre ces différents intervenants, d'éviter les retards dû à la communication entre eux, et d'améliorer par conséquent les délais des livrables.

C'est dans ce cadre que s'inscrit ce projet de fin de semestre pour assurer une bonne préparation à l'intégration de la vie professionnelle. En effet, en tant qu'ingénieur Génie Logiciel se spécialiser et maîtriser l'ensemble de pratiques qui permettent d'unifier le processus de développement et de déploiement d'un produit logiciel, allier les connaissances théoriques des différentes technologies avec des compétences pratiques pour la mise en place d'un pipeline DevOps et être capable de comparer, étudier et prendre des décisions stratégiques pour le déploiement des produits logiciels sont des atouts très important qui font la différence.

Ce rapport s'articule autour de quatre chapitres

- Le premier chapitre intitulé "Contexte général du projet" met le fléau sur les contraintes et les problématiques de développement des produits logiciels et donne une présentation générale du contexte du projet ainsi qu'aux exigences de ce projet.
- Le deuxième chapitre intitulé "Étude préalable" consiste à introduire les concepts clés nécessaires à la compréhension de notre projet ainsi que les exigences spécifiées pour la validation de ce projet.
- Le troisième chapitre intitulé "Conception de la solution" présente l'architecture de l'application en précisant les services, les outils et les technologies utilisés pour répondre aux exigences du projet et argumente le choix de la stratégie de déploiement.
- Le quatrième chapitre intitulé "Mise en œuvre et validation" permet de présenter le fonctionnement de l'application.

On clôture ce rapport par une conclusion générale dans laquelle on résume la solution et on expose quelques perspectives futures.

Chapitre 1 : Contexte Général du projet

1. Introduction

Ce projet est un projet académique développé dans le cadre de la validation des connaissances théoriques et des compétences pratiques acquises dans les 2 matières ; DevOps et Déploiement pour les étudiants en 5ème année de génie logiciel à l'INSAT ayant l'option : DevOps et tests de logiciels.

2. Problème

Problématique : Fonctionnement traditionnel sans DevOps

L'équipe de développement (dev) collectent les exigences du métier pour développer le code, cette même équipe de dev teste le nouveau code puis livre l'application à l'équipe de production et d'exploitation (ops), les ops mettent cette application en production.

En suivant ce modèle plusieurs problèmes se posent car les deux équipes ont des objectifs différents, d'une côté l'équipe dev veut faire évoluer rapidement les applications en cherchant à réduire le time en market réduit, et de l'autre côté l'objectif principal de l'équipe ops est de garantir la stabilité du système et ce par des contrôles très sévères de changement apporté au système d'information.

Cet antagonisme de ces objectifs entre l'équipe de dev et l'équipe des ops créer des conflits d'intérêt, les dev blâment les ops pour les problème de retard de livraison et les ops tiennent l'équipe de dev pour des responsable des incidents en production ..

3. Solution

a. DevOps : Évolution du terme DevOps

Au début de son apparition le terme DevOps est directement lié par la création d'une nouvelle culture et une nouvelle organisation d'entreprise cherchant à réconcilier les objectifs de l'équipe de développement et les objectifs de l'équipe d'exploitation.

Comme cet ensemble de pratique a prouvé son efficacité et vu l'évolution technologique exponentielle allié avec la demande et l'exigence du marché qui évoluent dans un environnement concurrentiel aigu, la philosophie DevOps est devenue de nos jours un état d'esprit qui impacte le cycle de vie d'un produit logiciel indépendamment de la grandeur du projet ou des équipes intervenantes.

b. Objectif du projet : Projet de fin de semestre

L'objectif de ce projet est de valider une formation complète pour garantir une meilleure intégration des étudiants dans la vie professionnelle.

En tant que étudiant en 5ème année logiciel, spécialiste en DevOps, nous devons être capable de mener à bien des mission de mise en place d'une automatisation complète des process (depuis le commit d'un fichier par un développeur) en réduisant au maximum les interventions humaines pour accélérer les livraisons.

Notamment, ces missions incluent l'assurance qualité, c'est à dire nous seront appeler à fiabiliser les livrables en assurant qu'un processus répétable est mis en place pour vérifier la qualité du logiciel développé et ce par exemple via une chaîne d'outils de contrôles qui s'exécutent à chaque étape.

A chaque étape, les outils doivent décider automatiquement d'accepter ou de rejeter le build candidat à la mise en production.

Pour mener à bien notre responsabilité d'homme-orchestre, qui est de plus en plus demandé dans le marché des produits logiciel, et faire preuve de nos compétences nous avons besoin d'une excellente compréhension des étapes de fabrication d'un logiciel ainsi que les enjeux du déploiement de ce même logiciel sur divers environnements avant d'atteindre la production.

c. Exigence du projet

Développement :

- 2 points finaux fonctionnels.
- 1 point final pour les métriques qui expose les métriques du niveau d'application.

Automatisation :

- Assurer une intégration continue.
- Assurer un déploiement continu et automatique.

Surveillance :

- Recueillir des données dans une base de données orientée séries pour la collecte des métriques.
- Créer un tableau de bord pour les métriques collectés.
- Créer une alerte.



Déploiement :

- Concevoir et mettre en œuvre une architecture.
- Choisir un service pour assurer le déploiement.
- Mettre en place une stratégie de déploiement.
- **Conclusion**

Dans ce chapitre on a présenté le contexte du projet ainsi que les problèmes posées avant l'intervention de la culture DevOps pour mettre le fléau sur l'importance du rôle d'un ingénieur génie logiciel spécialiste en DevOps et d'où la nécessité d'une validation pratique qui simule les cas réels de la vie professionnelle pour garantir la robustesse des connaissances théoriques acquise durant notre formation académique.

Pour réaliser ce projet, quels sont les outils utilisés afin de répondre aux exigences demandées et atteindre les résultats désirés ?

Chapitre 2 : Étude préalable

1. Introduction

Pour mettre en place un processus d'intégration continue (CI) et de déploiement continu (CD) nous avons besoin de définir quelques concepts et mots clés utilisés dans ce projet.

2. Concepts clés

a. Service

Un service est un périmètre fonctionnel qu'un fournisseur souhaite exposer à un certain type de consommateur.

Cet ensemble de fonctionnalités doit avoir un sens en exposant un nombre d'opérations offrant un traitement de bout en bout.

b. DevOps

Le DevOps est un mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops), notamment l'administration système.


c. Déploiement

Le déploiement en informatique est la mise en œuvre d'un nouveau système, service ou tout type de produit logiciel.

d. Intégration continue (Continuous Integration CI)

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier qu'à chaque modification de code source le résultat des modifications ne produit pas de régression dans l'application développée. Le principal but de cette pratique est de détecter les problèmes d'intégration au plus tôt lors du développement. De plus, elle permet d'automatiser l'exécution des suites de tests et de voir l'évolution du développement du logiciel.

L'intégration continue repose souvent sur la mise en place d'une brique logicielle permettant l'automatisation de tâches : compilation, tests unitaires et fonctionnels, validation produit, tests de performances... A chaque changement de code, cette brique logicielle va exécuter un ensemble de tâches, étapes et produire un ensemble de résultats, que le développeur peut par la suite



consulter. Cette intégration permet ainsi de ne pas oublier d'éléments lors de la mise en production et donc ainsi d'améliorer la qualité du produit.

Les principaux avantages d'une telle technique de développement sont :

- Le test immédiat des modifications.
- La notification rapide en cas de code incompatible ou manquant ; les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de dernière minute.
- Une version est toujours disponible pour un test, une démonstration ou une distribution.

e. Déploiement continue (Continuous Deployment **CD**)

Le déploiement continue est une approche qui vise la construction des produits logiciel d'une manière à ce qu'ils puissent être envoyé à la production d'une façon automatique à tout moment càd à partir du moment où les applications réussissent les test fonctionnels et d'acceptation automatisés ils doivent être déployés automatiquement à la production. L'idée est d'avoir un rythme de mise en production plus régulier, plus rapide et plus petit ce qui est beaucoup moins risqué qu'une grosse mise jour en mode big bang, l'objectif donc du déploiement continue et l'accélération de time to market.

f. La surveillance : Monitoring

Le processus de collecte, d'agrégation, de stockage et d'analyse des informations de suivi (métriques) pour mesurer l'état de santé et la performance du produit.

3. Conclusion

Dans ce chapitre on a défini quelques notions de base utilisées pendant ce projet, ceux qui sont considérés comme une connaissance requise pour pouvoir élaborer le travail.



Chapitre 3 : Conception de la solution

1. Introduction

On va présenter dans ce chapitre l'architecture de l'application, les outils et les technologies utilisés pour l'implémentation de notre solution ainsi que le choix de la stratégie de déploiement afin de répondre aux exigences fonctionnelles et non fonctionnelles spécifiées dans le chapitre précédent.

2. Conception de la solution

a. Architecture de l'application développée

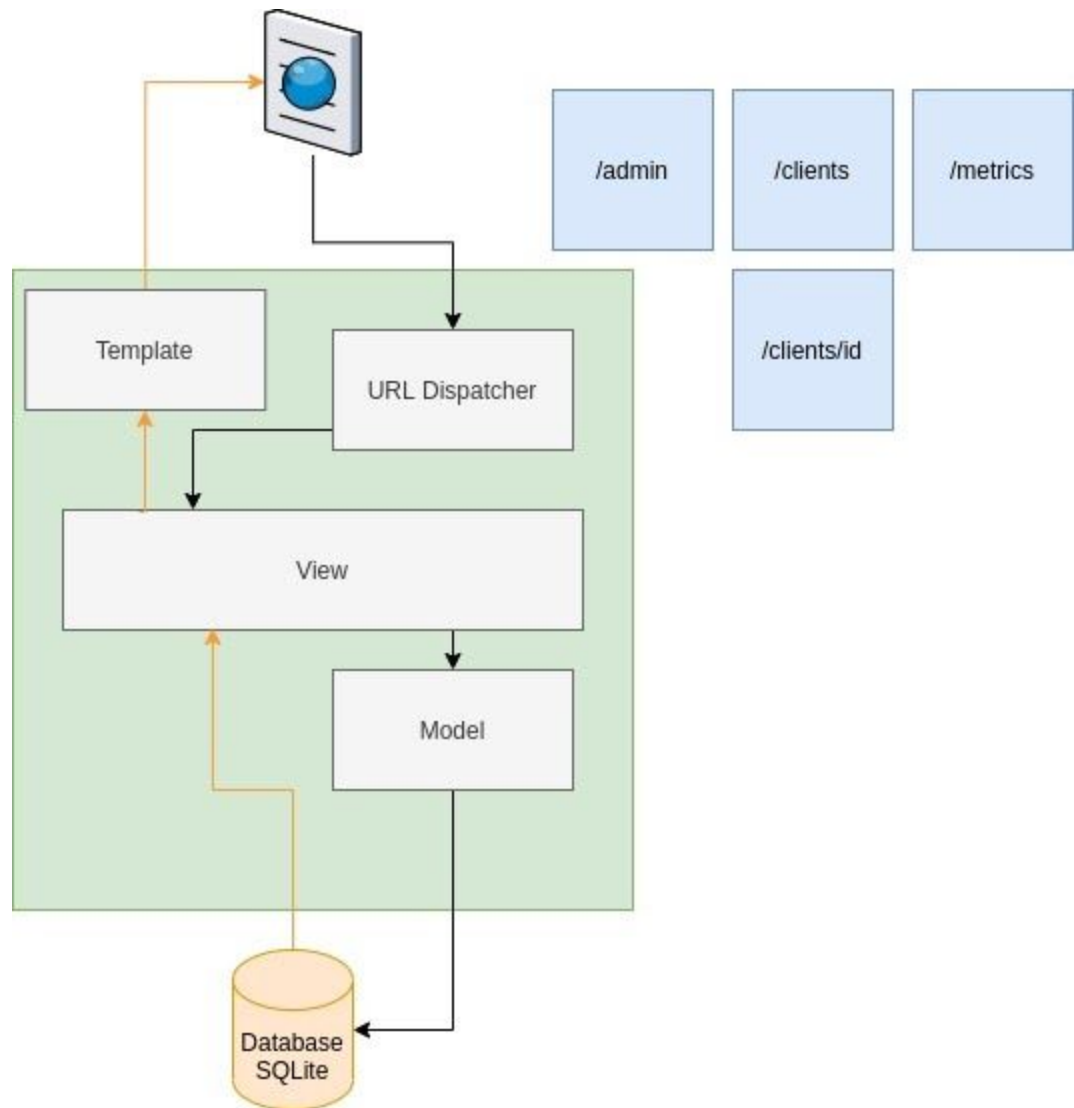


Figure 1. Architecture projet Django

b. Flux d'automatisation de l'intégration et du déploiement

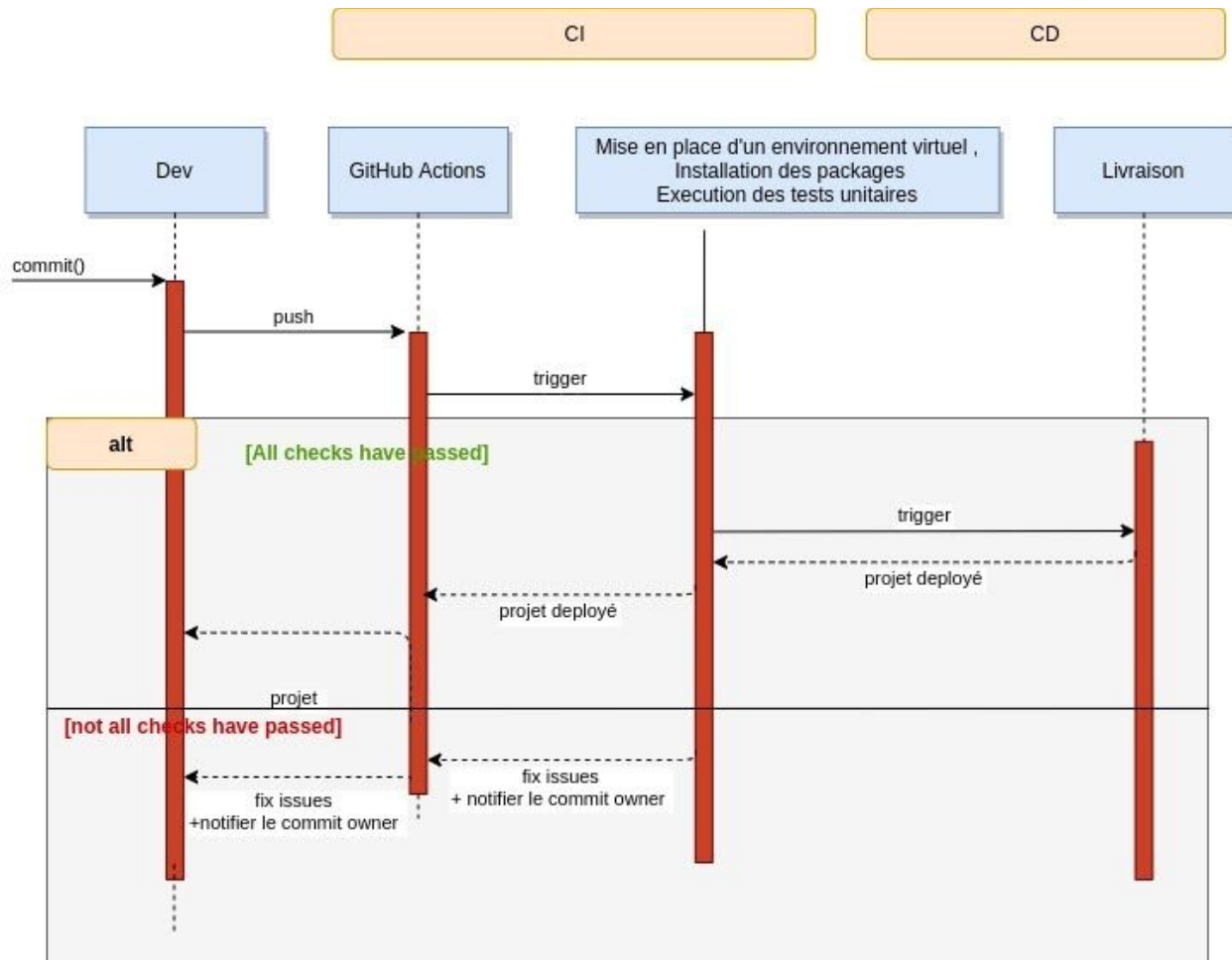


Figure 2. Flux d'intégration et de déploiement continue

c. Surveillance

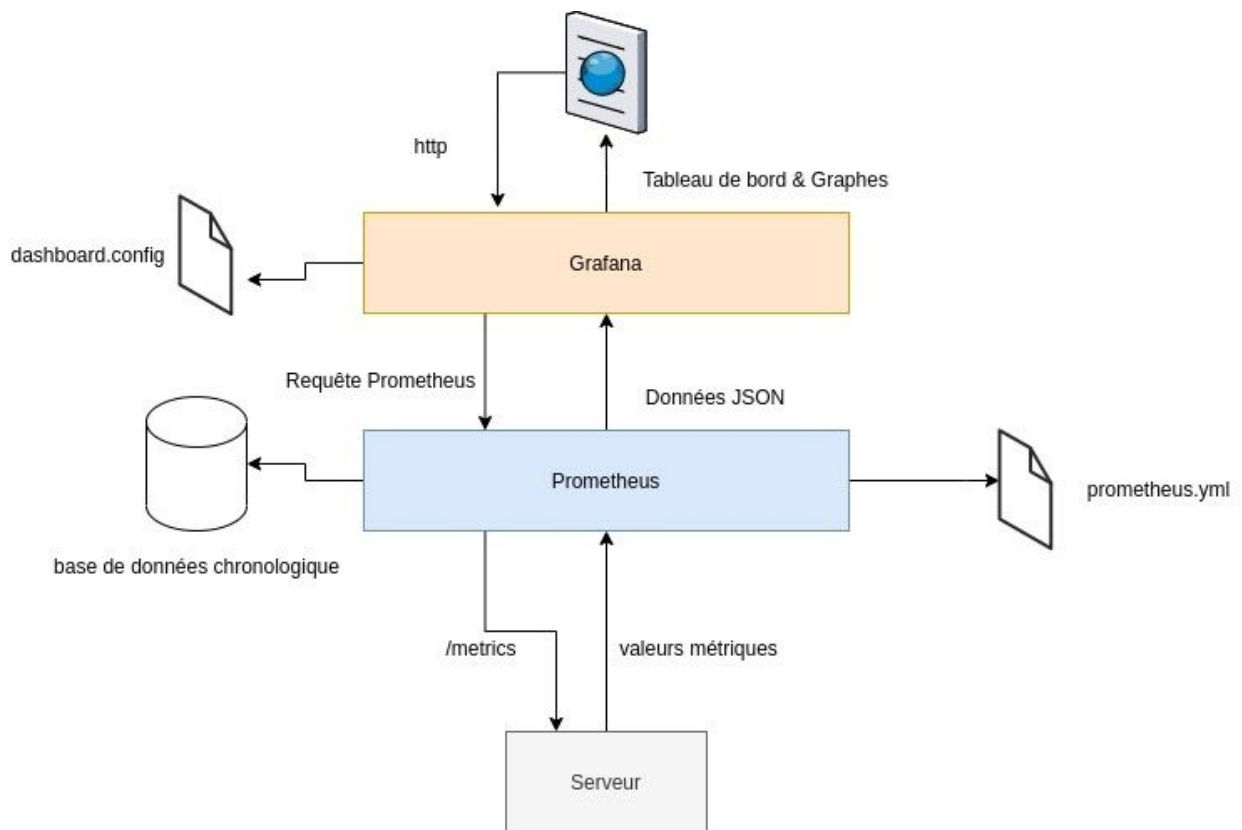


Figure 3 : Flux de transfert des métriques de surveillance

d. Déploiement sur Heroku

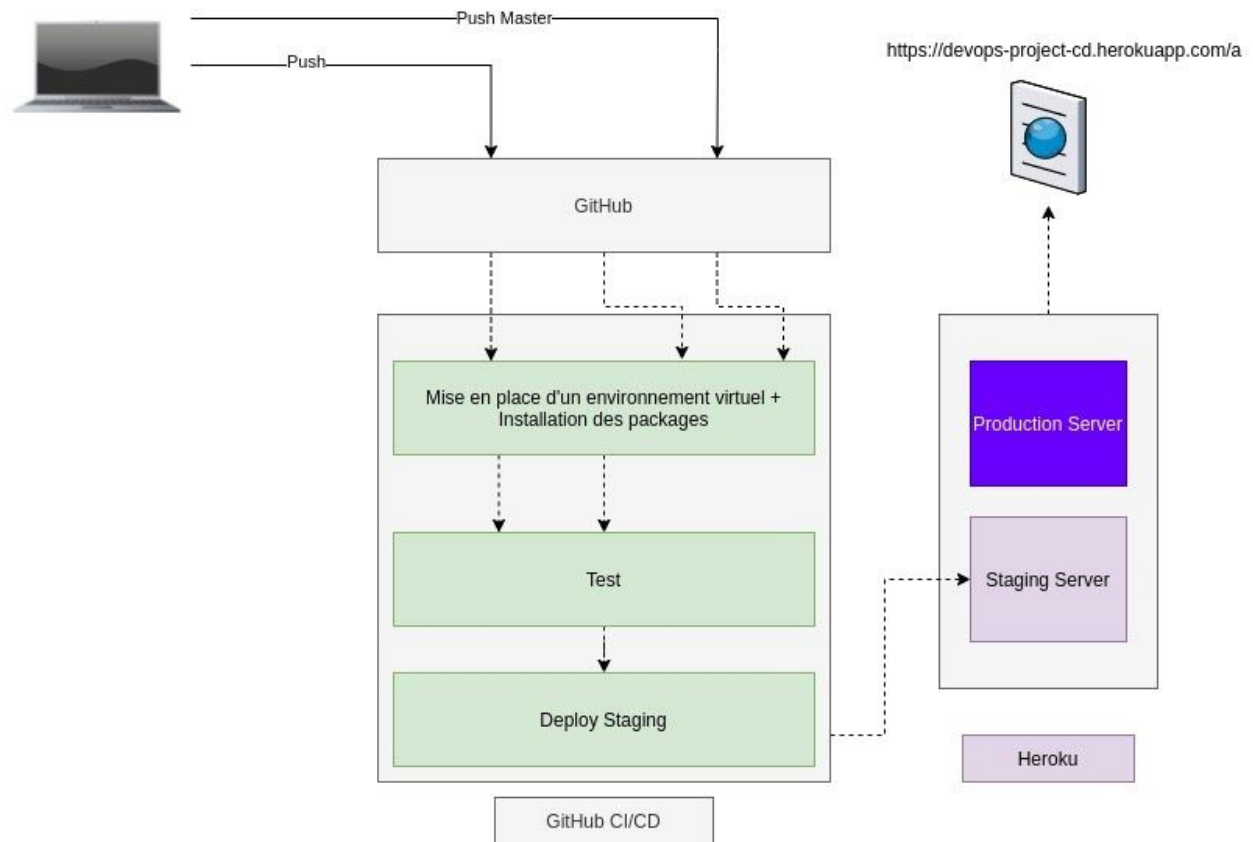


Figure 4. Déploiement sur Heroku

3. Conclusion

Dans ce chapitre on a défini les outils et les technologies utilisés ainsi que l'architecture choisie pour l'implémentation de notre solution dans ses 4 phases : développement, intégration, déploiement et surveillance.

Chapitre 4 : Mise en oeuvre et validation

1. Introduction

Dans ce chapitre on va présenter comment notre système fonctionne.

2. Mise en place de la solution

a. Developpement

/client

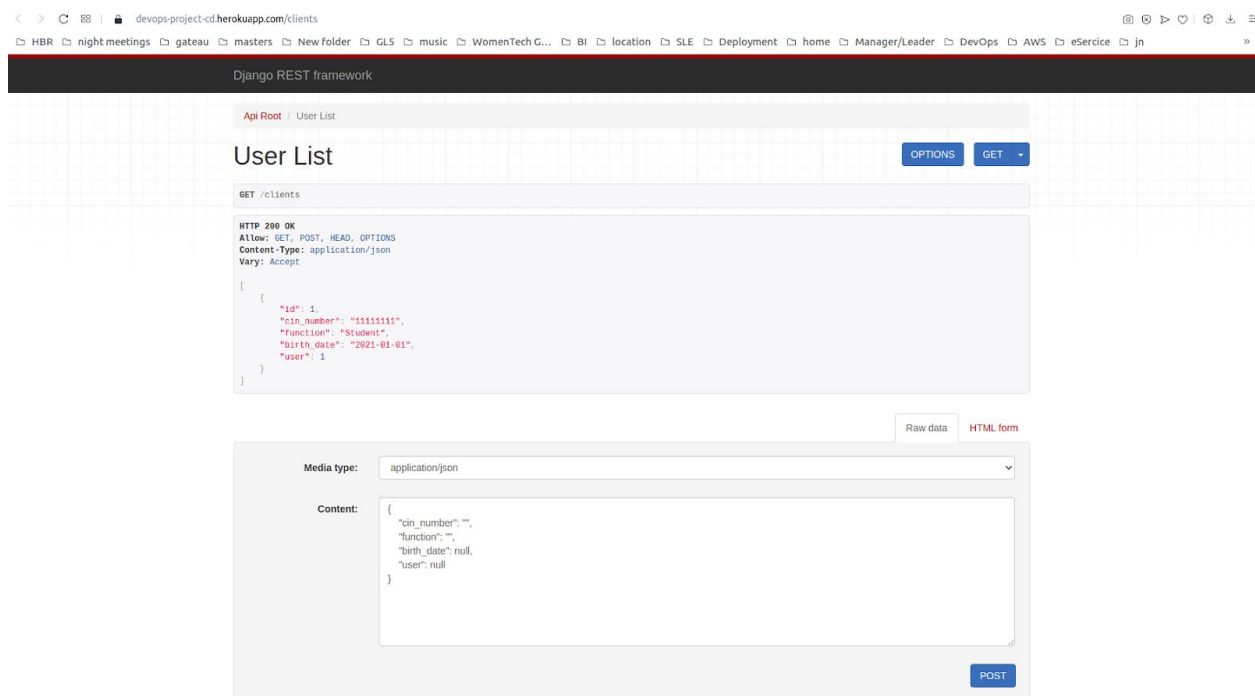


Figure 5 : Route /client -1.

/client/:id

The screenshot shows the Django REST framework API browser interface. The breadcrumb trail is "Api Root / User List / User Instance". The route being viewed is "GET /clients/1". The response status is "HTTP 200 OK". The allowed methods are "GET, PUT, PATCH, DELETE, HEAD, OPTIONS". The content type is "application/json". The response body is a JSON object:

```
{  "id": 1,  "cin_number": "11111111",  "function": "Student",  "birth_date": "2021-01-01",  "user": 1}
```

. Below the response, there is a section for "Media type" set to "application/json" and a "Content" field showing the same JSON object. There are buttons for "DELETE", "OPTIONS", "GET", "PUT", and "PATCH".

Figure 6 : Route /client -2-.

/admin

The screenshot shows the Django administration login page. The title is "Django administration". There are two input fields: "Username:" with the value "admin" and "Password:" with a masked password. Below the input fields is a "Log in" button.

Figure 7 : Route /admin -1- .

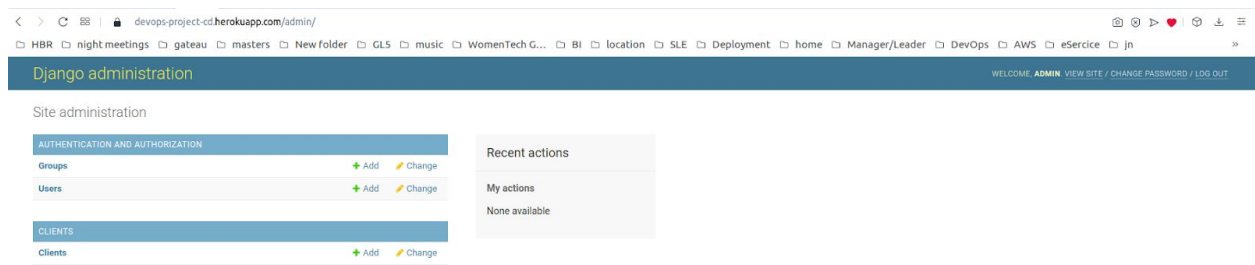


Figure 8 : Route /admin -2- .

/metrics

< > ☰ | 🔒 devops-project-cd.herokuapp.com/metrics

📁 HBR 📁 nightmeetings 📁 gateau 📁 masters 📁 Newfolder 📁 GLS 📁 music 📁 WomenTech G... 📁 BI 📁 location 📁 SLE 📁 Deployment 📁 home 📁 Man

```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 1710.0
python_gc_objects_collected_total{generation="1"} 1011.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable object found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 179.0
python_gc_collections_total{generation="1"} 16.0
python_gc_collections_total{generation="2"} 1.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="6",patchlevel="12",version="3.6.12"} 1.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 1.60415744e+08
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 4.5645824e+07
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.61117821546e+09
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 1.4100000000000001
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 14.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 10000.0
# HELP django_model_inserts_total Number of insert operations by model.
# TYPE django_model_inserts_total counter
django_model_inserts_total{model="client"} 0.0
# HELP django_model_inserts_created Number of insert operations by model.
# TYPE django_model_inserts_created gauge
django_model_inserts_created{model="client"} 1.6111782168230126e+09
# HELP django_model_updates_total Number of update operations by model.
# TYPE django_model_updates_total counter
django_model_updates_total{model="client"} 0.0
# HELP django_model_updates_created Number of update operations by model.
# TYPE django_model_updates_created gauge
django_model_updates_created{model="client"} 1.6111782168230777e+09
# HELP django_model_deletes_total Number of delete operations by model.
# TYPE django_model_deletes_total counter
django_model_deletes_total{model="client"} 0.0
# HELP django_model_deletes_created Number of delete operations by model.
# TYPE django_model_deletes_created gauge
django_model_deletes_created{model="client"} 1.611178216823122e+09
# HELP django_migrations_unapplied_total Count of unapplied migrations by database connection
# TYPE django_migrations_unapplied_total gauge
django_migrations_unapplied_total{connection="default"} 0.0
# HELP django_migrations_applied_total Count of applied migrations by database connection
# TYPE django_migrations_applied_total gauge
django_migrations_applied_total{connection="default"} 20.0
# HELP django_db_new_connections_total Counter of created connections by database and by vendor.
# TYPE django_db_new_connections_total counter
django_db_new_connections_total{alias="default",vendor="sqlite"} 1.0
```

Figure 9 : Route /metrics.

b. Automatisisation

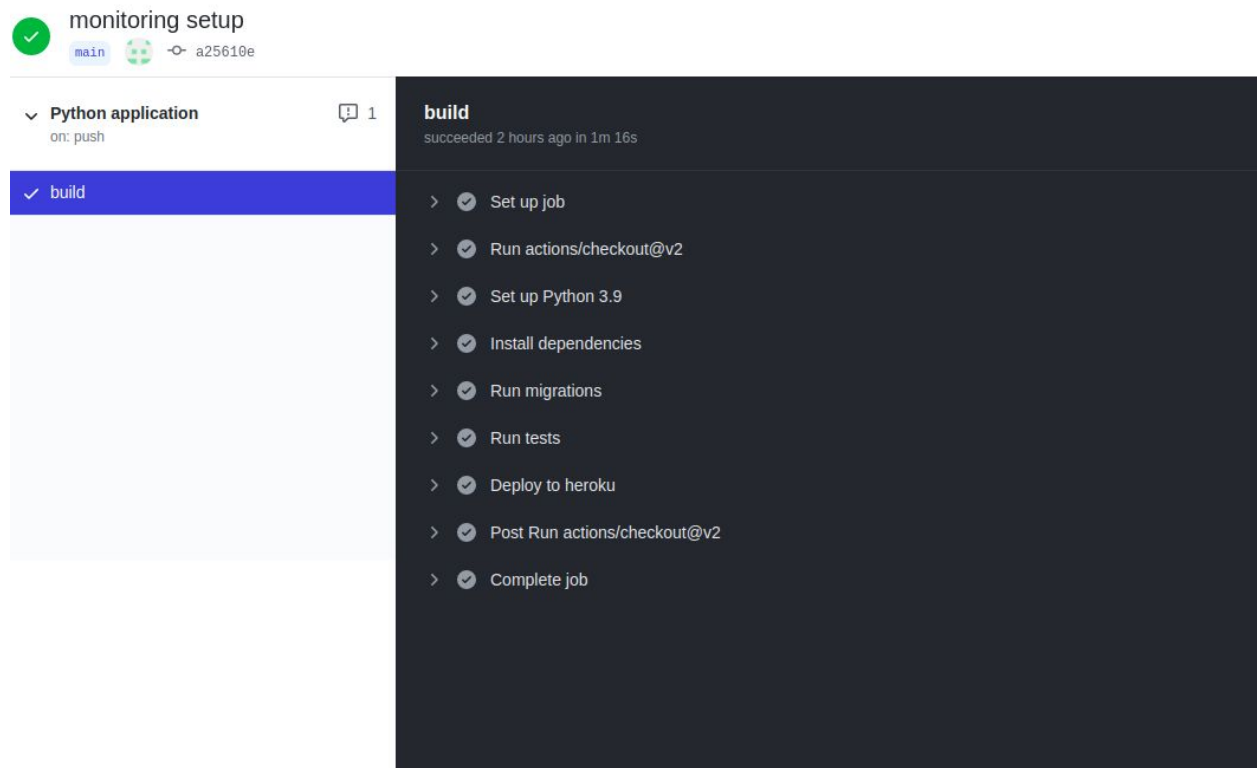


Figure 10 : GitHub Actions

c. Surveillance

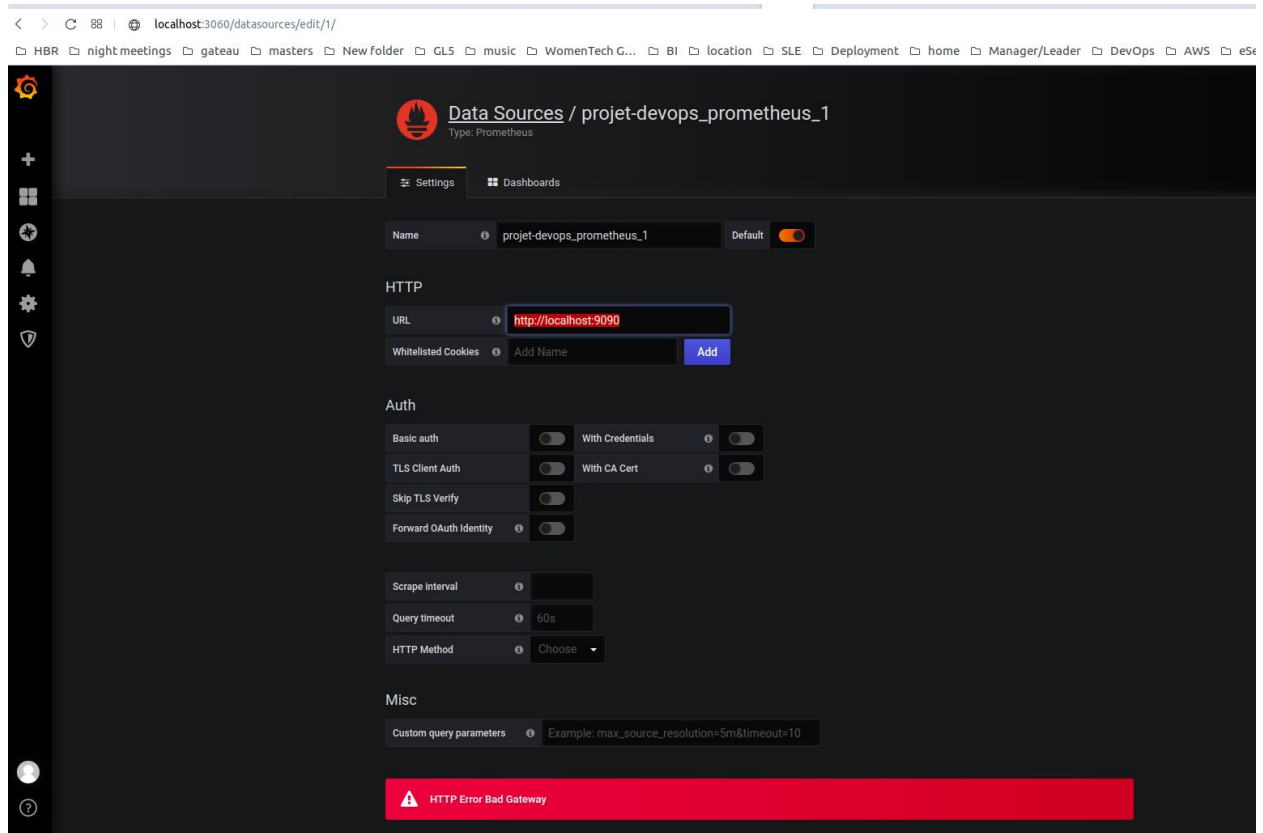


Figure 11 : Configuration Grafana Prometheus

3. Conclusion

Dans ce chapitre on a présenté le résultat du travail accompli pour répondre aux exigences spécifiées ainsi que l'output final du projet.

Conclusion Générale et Perspective

Le développement des applications représente un défi dans l'ingénierie logicielle. Il impose de fournir les bons outils et les processus cohérents aux développeurs pour assurer la livraison du logiciel.

Ce travail s'inscrit dans le cadre de finalisation du cursus académique, a été réalisé pour assurer qu'un étudiant en 5ème année génie logiciel à l'INSAT ayant choisi comme option : DevOps et tests logiciel, a touché le processus entier d'implantation d'une pipeline DevOps pour un produit logiciel.

On a décomposé le travail en 4 phases itératives : développement, intégration, déploiement et surveillance.

Le première phase est relative au développement d'un service fournissant 3 points finaux.

La deuxième phase est l'automatisation du processus d'intégration de l'application en précisant les étapes à suivre pour assurer la compilation et l'exécution des tests unitaires à chaque pull/push de/vers l'outil de gestion des versions.

La troisième phase est l'automatisation du déploiement de l'application en utilisant Hreuko comme plateforme de déploiement.

Et dernièrement, la quatrième phase est l'assurance d'une dashboard de surveillance en collectant les données avec Prometheus et les afficher dans une interface intuitive : Grafana.

Ce travail nous a aidé à mieux comprendre les enjeux de l'implémentation d'une pipeline DevOps et avoir une vision sur la meilleure façon de mettre en place une plateforme d'intégration et de déploiement continu pour améliorer la livraison des produits logiciels.

On tient à présent à souligner quelques extensions intéressantes pour améliorer ce projet et renforcer nos compétences pour mener à bien des missions professionnelles en tant qu'ingénieur DevOps.

Pour conclure, on souhaite que ce travail apporte la satisfaction aux enseignants, et qu'il le trouve à la hauteur de leurs attentes.



Bibliographie/Netographie

<https://skillvalue.com/fr/blog/tech-skills/ingenieur-devops-definition/>

<http://docs.rapidsms.org/en/v0.10.0/topics/architecture.html>

<https://fnproject.io/tutorials/grafana/>

<https://www.toptal.com/devops/effective-ci-cd-deployment-pipeline>

<https://www.sipios.com/blog-tech/monitoring>

<https://docs.docker.com/compose/>

https://medium.com/@humble_bee/django-basics-for-a-beginner-5d864e6aa084

<https://www.freecodecamp.org/news/sphinx-for-django-documentation-2454e924b3bc/>

■ ■ ■