

DÉPLOIEMENT DE SMILE EN MODE SAAS SUR L'INFRASTRUCTURE DE L'UNIVERSITÉ DE STRASBOURG (PAAS)



Romain AMBROISE - Unicaen

12 – 13 Juin 2024



Sommaire



Contexte

1. Construction de l'application

1. Déploiement en local est organisation technique
2. CI/CD & runners gitlab

2. Infrastructure Kubernetes

- 1) Overlays : Création d'instances
- 2) SMILE instance manager

3. Retour d'expériences

1. Premières recommandations
2. Points de vigilances
3. Avantages du processus de déploiement en mode SaaS

Conclusion



Contexte



SMILE est un projet **open-source**, développé par l'Université de Caen, Normandie ; et déployé dans l'infrastructure de l'Université de Strasbourg.

Ce projet en mode **SaaS** (*Software as a Service*) est précurseur pour ce type de déploiement.

Il fait partie du mouvement pour le partage de solutions via le **EsupPortail** à destination des Universités demandeuses et ainsi éviter à tout le monde de « ré-inventer la roue ».

De fait, nous rencontrons tous les mêmes besoins. À savoir, **la mutualisation** d'un système de gestion des formations, des personnes, des structures, de la mobilité internationale, etc.

Avec ce système d'**hébergement**, le développement, l'installation, et la mise en **production opérationnelle** se fait par une unique équipe.

The screenshot shows the SMILE web interface. At the top, there is a navigation bar with a logo and menu items: My space, Etudiants, Cours, Configuration, Administration, a language selector (FR), and an Aide menu. The user is identified as Romain Ambroise, Administrateur technique, with a Déconnexion link.

The main content area is titled "COURS". Below the title, there is a help message: "Besoin d'aide ? Contactez-nous à l'adresse smile@unicaen.fr" and "Need help ? Ask us smile@unicaen.fr".

There are two main sections: "Ma composante principale" and "Choix". "Ma composante principale" has a dropdown menu labeled "Sélectionner...". "Choix" has a table with columns "Nom", "ECTS", and "Principal". Below this, a message says: "En attendant votre validation, vous pouvez simuler vos choix de cours".

Below these sections, there are several filters: "All", "Intitulé des cours", "Niveau", "Semestre", "Langue", and "ECTS".

At the bottom, there is a table header with columns: "Groupe", "Composante", "Nom", "Niveau", "Semestre", "Langue", "ECTS", "Voir", and "Sélectionner". Below the header, it says "No data available in table".

At the very bottom, there is a footer with logos for "RÉPUBLIQUE FRANÇAISE" and "UNICAEN UNIVERSITÉ CAEN NORMANDIE", and navigation links: "À propos", "Contact", "Plan de navigation", "Mentions légales", and "Vie privée".





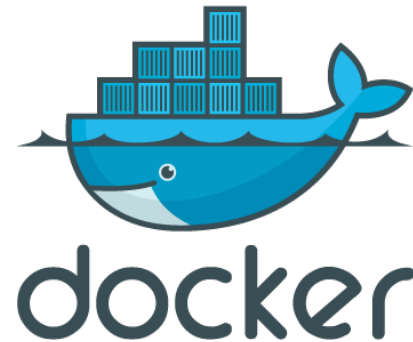
1. Construction de l'application



En amont du déploiement et de Kubernetes, il nous a d'abord été nécessaire de s'organiser côté développement.

1. Construction de l'application pour un **environnement** de développement en **local** (Dockerfiles, docker-compose) à destination des développeurs.
2. En parallèle, penser et prévoir du OnPremise et la destination vers Kubernetes.
3. Configuration d'un dépôt git pour le **versionning**.
4. Développement de la pipeline CI/CD pour la construction automatique des éléments de l'application et le déploiement vers un **registry** d'images.

- Dialogue avec l'équipe de développement de l'application
- Proxy ? Variables d'environnement, Gestion de la BDD, Démonstration
- Configuration des runners (docker in docker) pour la CI.
- Définir des règles de CI (releases, versions, nomenclatures) et règles de construction (branches concernées)





a. CI/CD & Runners gitlab



Construction et mise à disposition de l'application pour l'infrastructure Kubernetes

Le CI/CD de SMILE est basé sur le moteur **docker** pour la création des conteneurs.

Son code source est disponible sur le Github Esup (releases) mais aussi sur le Gitlab Unicaen pour le début de la chaîne CI/CD.

Pour pouvoir construire les images, gitlab a besoin de « runners ».

Ce runner est également déployé avec docker pour faciliter sa maintenance et sa montée de version.

On va facilement pouvoir lui attribuer des ressources (CPU, RAM) ainsi que les configurations proxy, cache, etc.

On le connecte ensuite au dépôt comme unique runner pour le projet.

```
version: '3'

services:
  runner: #smile
    image: gitlab/gitlab-runner:v15.2.1
    container_name: runner-smile
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /opt/docker/daemon.json:/daemon.json
      - /opt/docker/http-proxy.conf:/http-proxy.conf
      - /opt/runner-config.yaml:/etc/gitlab-runner/config.toml
    ports:
      - "8093:8093"
```

Assigned project runners

● #68 (8KtY_zuFX) 🔒

Remove runner

run jobs smile

docker local local-runner smile





b. Interfaces CI/CD et Quay Registry



a. Passage de la CI

All 485 Finished Branches Tags

Clear runner caches

CI lint

Run pipeline

Filter pipelines



Show Pipeline ID

Status	Pipeline	Created by	Stages
<p>Passed</p> <p>00:03:36</p> <p>1 week ago</p>	<p>replace DROP add multi service core config...</p> <p>#28367 master eef3835b</p> <p>latest</p>		<p>7 green checkmarks</p>

b. Les images arrivent sur le registry (Quay) d'Unistra

Repository Tags

Compact

Expanded

Show Signatures

1 - 20 of 20



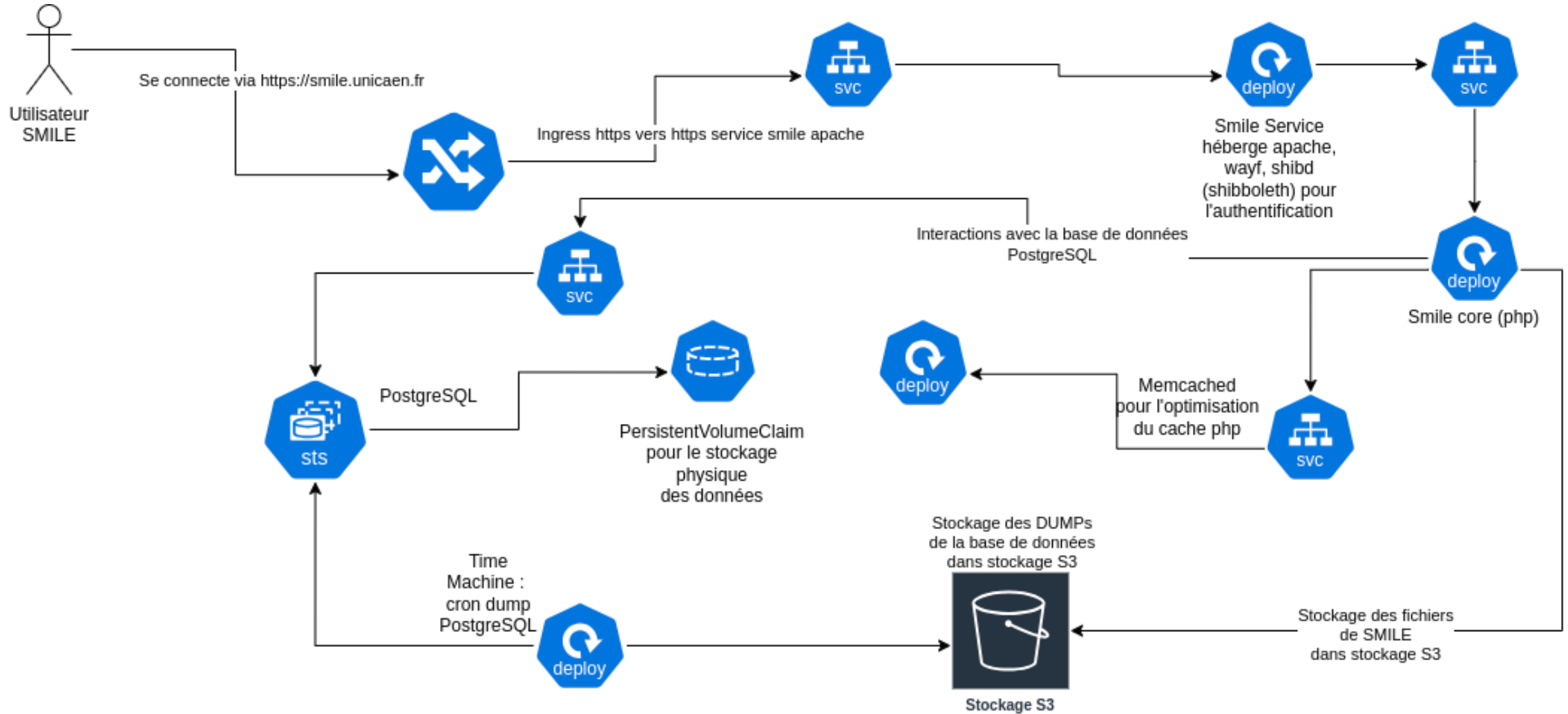
Filter Tags...

TAG	LAST MODIFIED	SECURITY SCAN	SIZE	EXPIRES	MANIFEST
1.2.6	14 days ago	Passed	546.3 MiB	Never	SHA256 b63d25d8211f
1.2.5	18 days ago	Passed	554.3 MiB	Never	SHA256 5def0471ce49
1.2.4	18 days ago	Passed	554.3 MiB	Never	SHA256 65986795d9a1





3. Infrastructure Kubernetes (k8s)





b. Overlays et création d'instances



SMILE est un projet qui est demandé par de nombreuses Universités.

Pour répondre à cette demande croissante, on va créer des « **instances** » du projet qui seront **uniques**, **cloisonnées** et **indépendantes** les unes des autres.

L'outil « **Kustomize** », en plus des **policies** de Kubernetes répond parfaitement à ces besoins.

Grâce à notre base, définie précédemment, on va charger cette **structure** et la modifier pour chacune des instances.

Cette méthode de fonctionnement permettra également à **SMILE** d'être servie sous différentes **versions** de l'application suivant les besoins de l'Université.

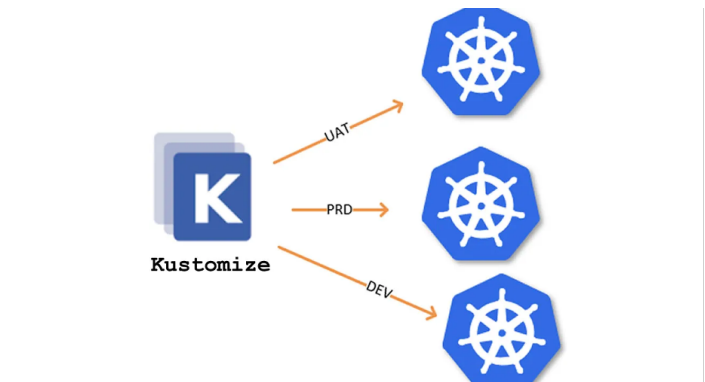




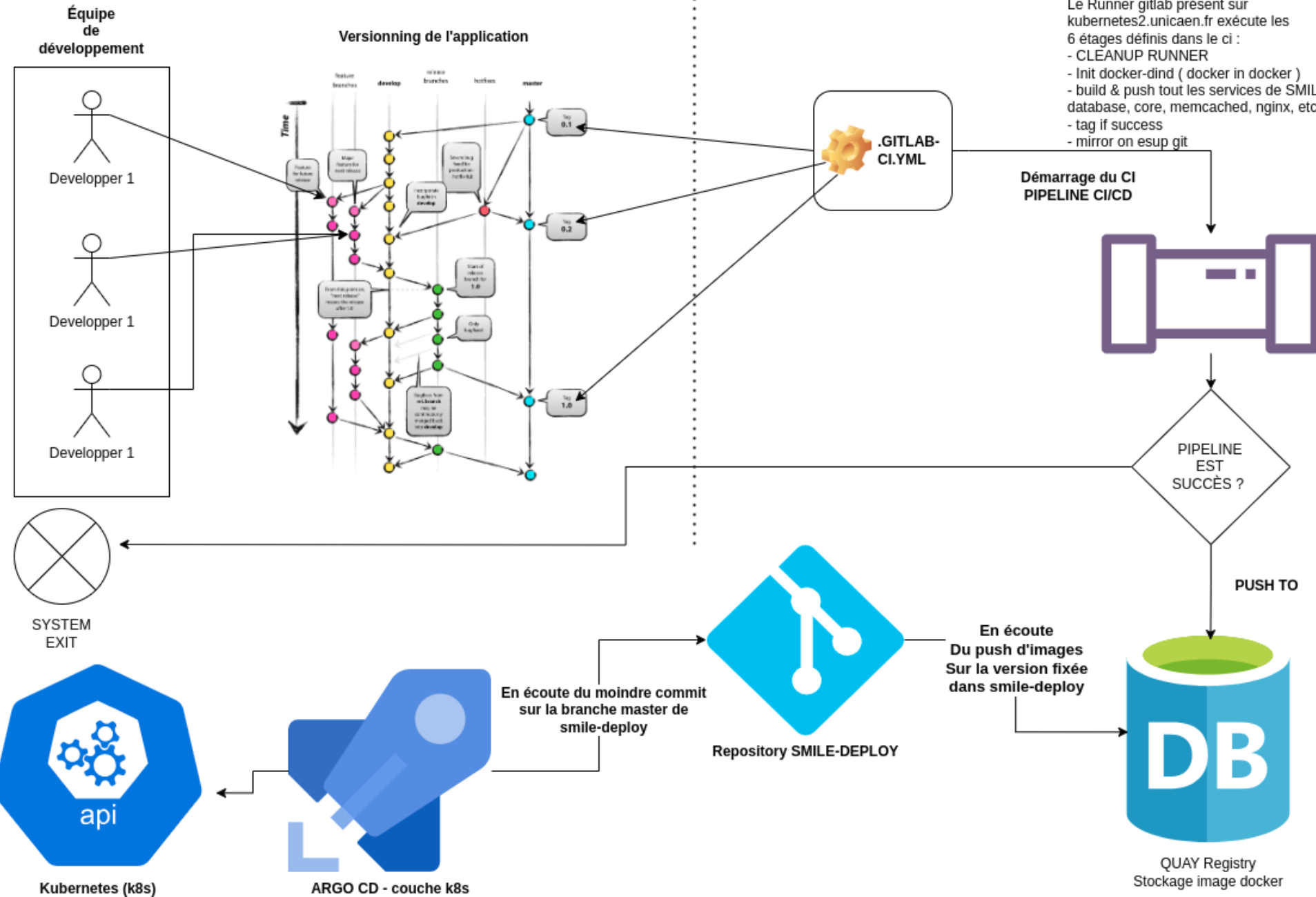
Diagramme du commit à la production

En écoute d'une merge request sur la branche MASTER

Exemple de SMILE



- Le Runner gitlab présent sur kubernetes2.unicaen.fr exécute les 6 étages définis dans le ci :
- CLEANUP RUNNER
 - Init docker-dind (docker in docker)
 - build & push tout les services de SMILE (database, core, memcached, nginx, etc)
 - tag if success
 - mirror on esup git





c. Automatisation : smile-manager



Dans l'optique de faire un déploiement automatique de bout en bout, on a pensé à un script en python à destination des administrateurs fonctionnels pour le déploiement.

Ce script est basé sur un dépôt git, où l'on va venir éditer un fichier texte en proposant des instructions. Puis un autre programme va venir régulièrement pull ce dépôt et **appliquer la création d'une instance ou encore sa mise à jour.**

Avantages :

- ✓ Libérer du temps aux développeurs.
- ✓ La communication mainteneurs / Université distante se fait par la même personne pour les mises à jour.
- ✓ Automatisation du process.
- ✓ Éviter de suivre une documentation qui peut être très longue pour la création d'une instance ou sa màj.

```
action|instance|versionSMILE|dd/MM/yyyy-hh:00
```

```
U|unistra|1.1.17|09/10/2023-12:00
```



4. Retour d'expérience : premières recommandations



Passage à Kubernetes :

- Nécessite de solides connaissances en Docker, système, réseau, un peu de développement et une formation à l'interaction avec un cluster Kubernetes.

Bonne communication avec l'équipe de développement :

- Déployer avec eux l'application localement et réfléchir à des **stratégies de déploiement** pour répondre aux besoins de mises à jour

S'assurer des dumps de données pour éviter les pannes et la haute disponibilité :

- Pour SMILE, nous avons un conteneur nommé « **smile-timemachine** » qui exécute des **dumps réguliers** de la base de données sur un **stockage distant** de l'infra de Kubernetes.



4. Retour d'expérience : points de vigilance



Prévoir des règles autour de la construction d'images de l'application à destination du registry

Dans le CI/CD il est crucial de :

- ✓ **filtrer les branches** sur lesquelles on déploie
- ✓ s'assurer de **ne pas surcharger des images existantes** (si un tag existe, alors on ne le remplace pas). Cela peut se faire avec l'API du registry d'images

Il est donc primordial de définir des règles strictes de déploiement autour de la construction d'images sachant que tout est connecté pour le déploiement de bout en bout.



4. Retour d'expérience : avantages du SaaS



- On gagne énormément de temps sur les **mises à jour** et la **maintenance**.
Actuellement, si il n'y a pas d'erreurs, la montée de version **se fait en quelques minutes**.
- Où avant on devait mettre à jour les dépendances, adapter les fichiers de configurations des dépendances, etc.
- On obtient également le **contrôle des versions** de chaque instance à un unique endroit, on a ainsi une meilleure **visibilité** sur le parc applicatif.
- En tant que développeur, on peut facilement avoir un environnement de développement, de test et de pré-prod se rapprochant au mieux du contexte **de production**.



5. Analyse de la montée de charge



Dans l'optique de tester, restreindre l'application à des quotas et de l'optimiser ; des tests de charge ont été réalisés côté application.

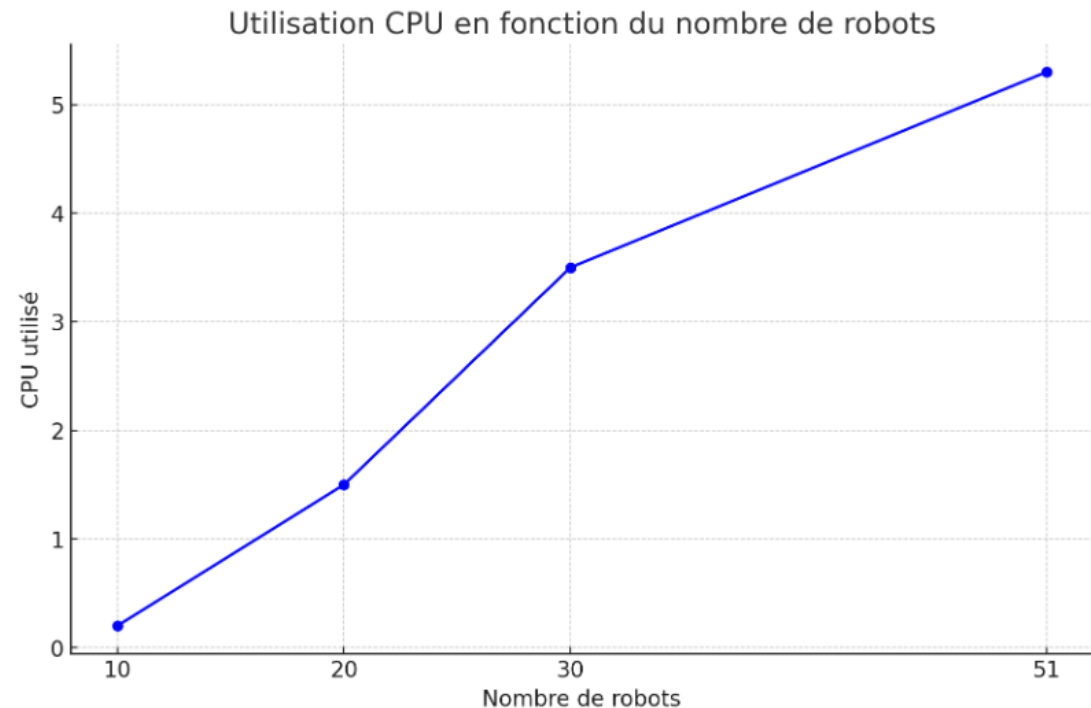
Script en **Python** utilisant **Selenium**

Première étape de login via le remplissage du formulaire de connexion

Il va **cliquer aléatoirement** sur les balises HTML de lien pour charger le plus de page de SMILE

Ce script Python est **encapsulé dans un conteneur docker** pour ainsi pouvoir le déployer et le dupliquer facilement.

Couplé à **Portainer.io** pour gérer les conteneurs en exécution





Conclusion



Retour sur la globalité du projet :

Deux équipes, de développement côté Unicaen et d'infrastructure côté Unistra, étaient en charge de la réalisation de ce projet en mode SaaS.

Stabilité de l'infrastructure ; disponibilité, bienveillance, conseils, maîtrise des différents sujets des deux équipes, ont fait que le projet s'est très bien déroulé.

Merci aux deux équipes !

