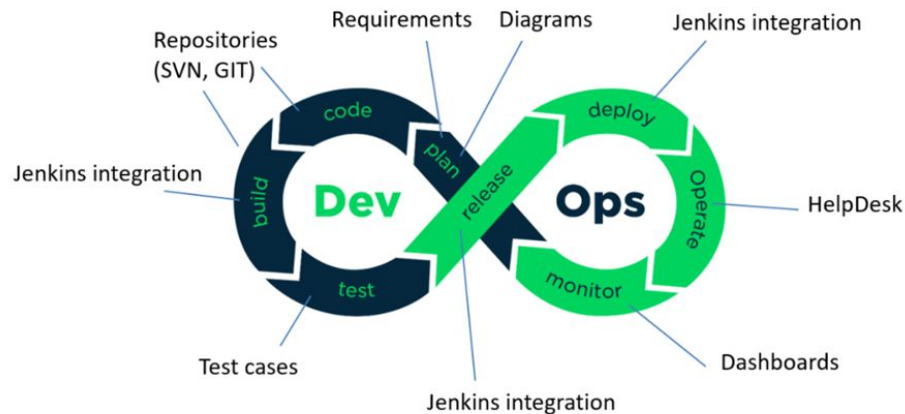


Mise en place d'une chaîne complète CI/CD pour une application Node EJS

Projet proposé et encadré par Mr. SADO



18 Janvier 2022

Présentation des membres du groupe 3



Abdelkader RAHMANI

Consultant DevOps
Ing. Dr. Génie Chimique



Aurélien DAIX

Consultant DevOps



Oussama ZAID

Consultant Devops
Admin Systemes et
reseaux



Renaud SAUTOUR

Consultant Devops
Artisan
Consultant CRM

Analyse du projet

Devops-project2:

- Fichier ejs (embedded javascript)
- Modules node
- Package json (manifest)

=> site web à déployer

The screenshot shows the GitHub repository page for 'sadofrazer/devops-project2'. The repository is public and has 1 watch, 5 forks, and 0 stars. The main navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The repository is currently on the 'master' branch with 1 branch and 0 tags. The file list shows the following files and their commit history:

File	Commit	Time
.idea	first commit	6 days ago
bin	first commit	6 days ago
node_modules	first commit	6 days ago
public	first commit	6 days ago
routes	first commit	6 days ago
views	first commit	6 days ago
README.MD	edit Readme	6 days ago
app.js	first commit	6 days ago
package-lock.json	first commit	6 days ago
package.json	first commit	6 days ago

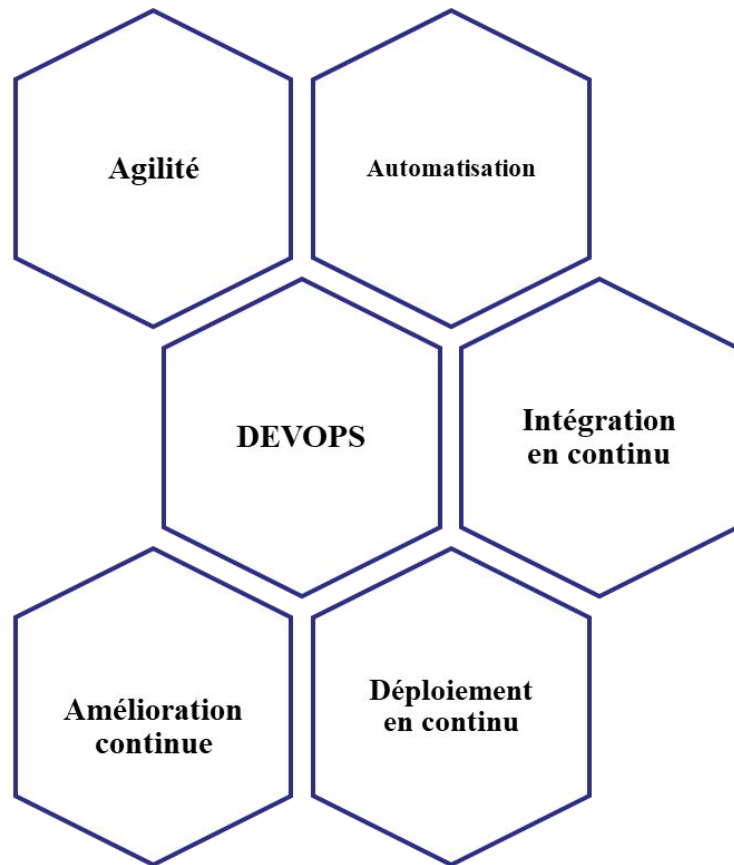
The right sidebar contains the 'About' section with the text 'No description, website, or topics provided.' and the 'Releases' section with the text 'No releases published'. The 'Packages' section also shows 'No packages published'. The 'Languages' section shows a bar chart with EJS at 90.8% and CSS at 5.4%.

<https://github.com/sadofrazer/devops-project2.git>

Le projet orienté DevOps

Plan:

- Outils utilisés
- Infrastructure
- Workflow et chaine CICD
- Demo



Outils utilisés



- **Gestion de versions**
- **Travail collaboratif**
- **Possibilité de forker un projet**
- **Trigger de Pipelines CICD**
- **https://github.com/projetajc-group3/projetajc_node.git**



- **Cloud provider**
- **Multitude de services**
- **EC2**
- **S3**



- **Serveur d'automatisation**
- **Open source**
- **Nombreux plugins**
- **Pipeline CICD**

Outils utilisés



- **Conteneurisation**
- **Micro services**
- **Agilité et flexibilité**
- **Déploiement**
- **Analyse de vulnérabilités**
- **Scanner des paquets et des langages**
- **Une large communauté**
- **Open source**
- **Registry publique**
- **Images Docker**
- **<https://hub.docker.com/u/projetajcgroup3>**

Outils utilisés



- **créer et versionner l'infrastructure**
- **Open source**
- **Infrastruce As Code**

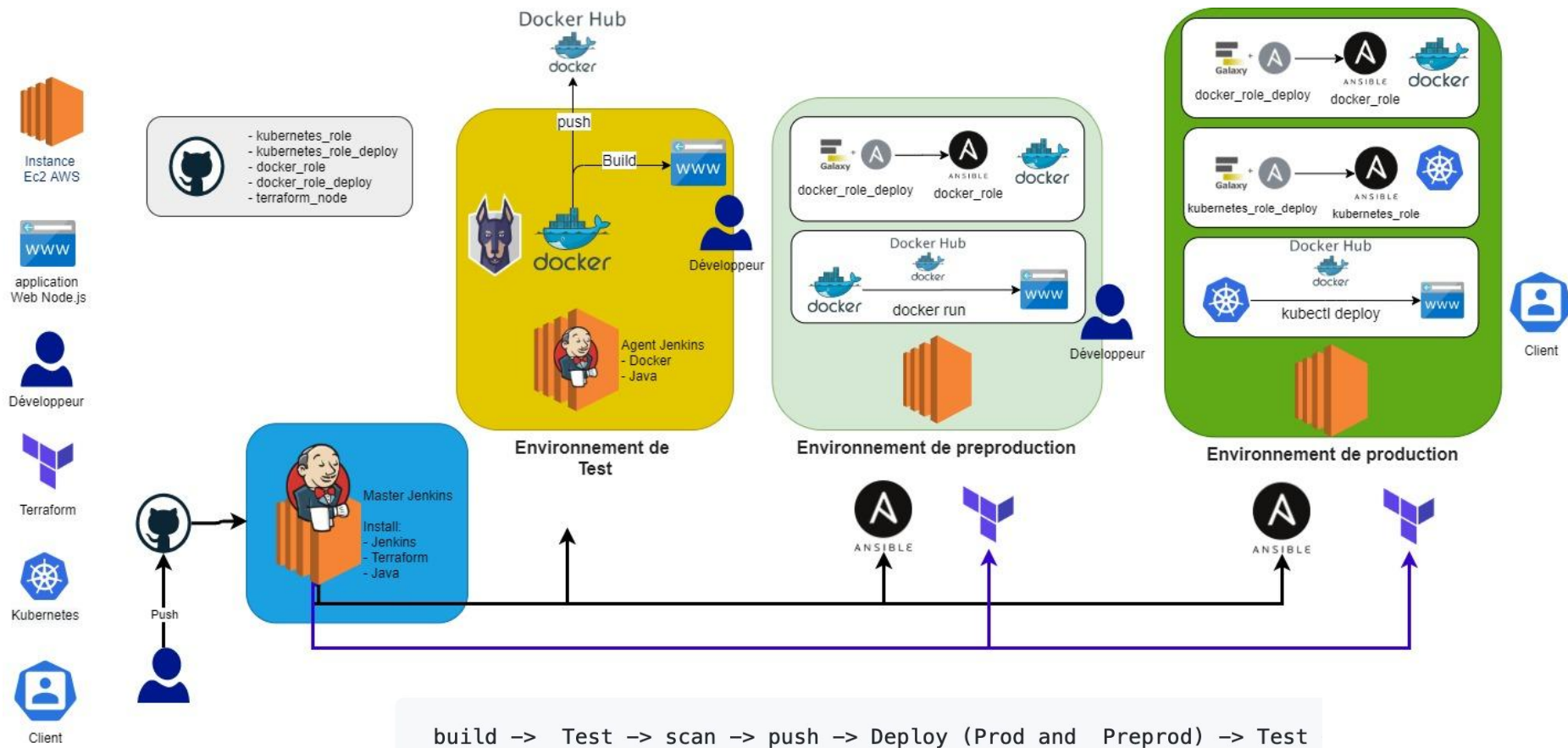


- **Gestion des configurations**
- **Service Agentless**
- **Infrastructure As Code**
- **Rôles Ansible**



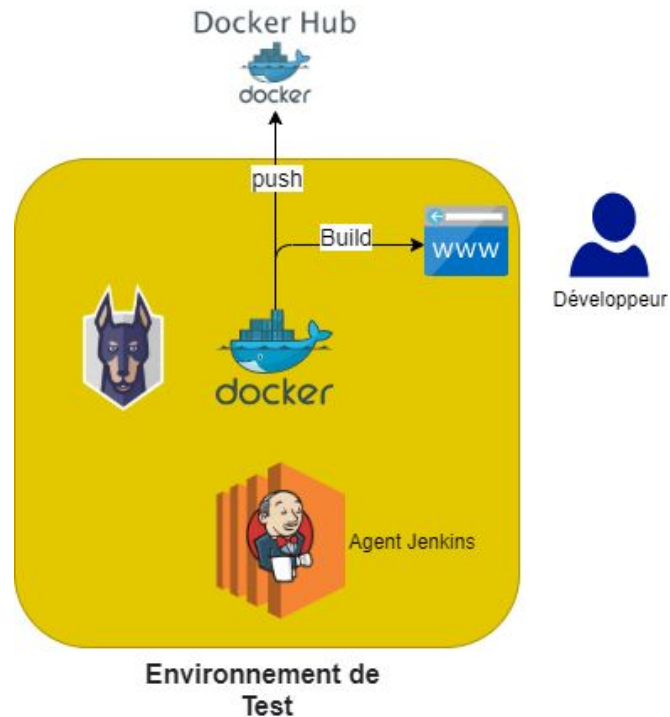
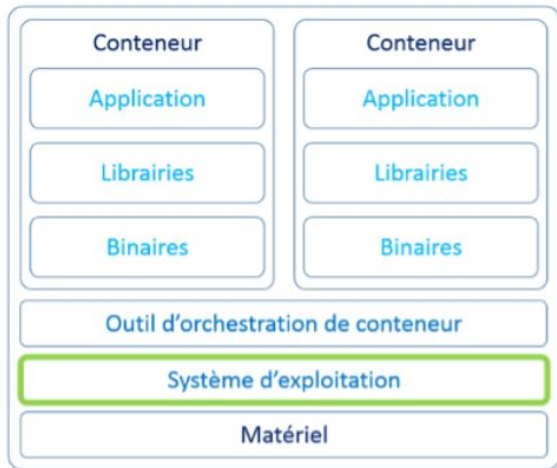
- **Orchestrateur**
- **Scalabilité, Load Balancing**
- **Infrastruce As Code**

Infrastructure & Workflow



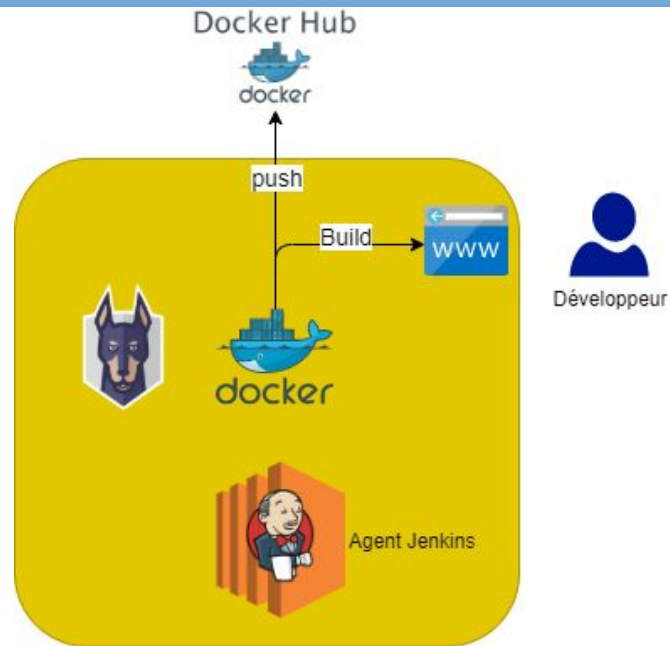
Environnement de test

- Création d'un agent Jenkins : “Agent-test”
- Construire une image à l'aide d'un dockerfile
- Tester le fonctionnement de l'application

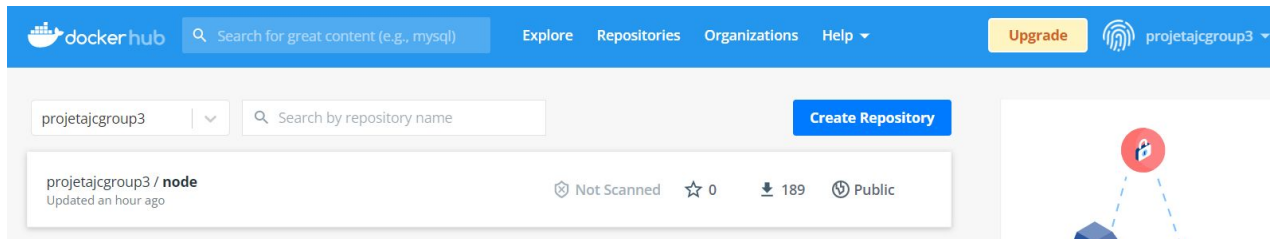


Environnement de test

- Scan de sécurité et de vulnérabilité
=> Snyk Security Report
- Push vers Docker Hub
=> Sauvegarder notre artefact



Environnement de Test

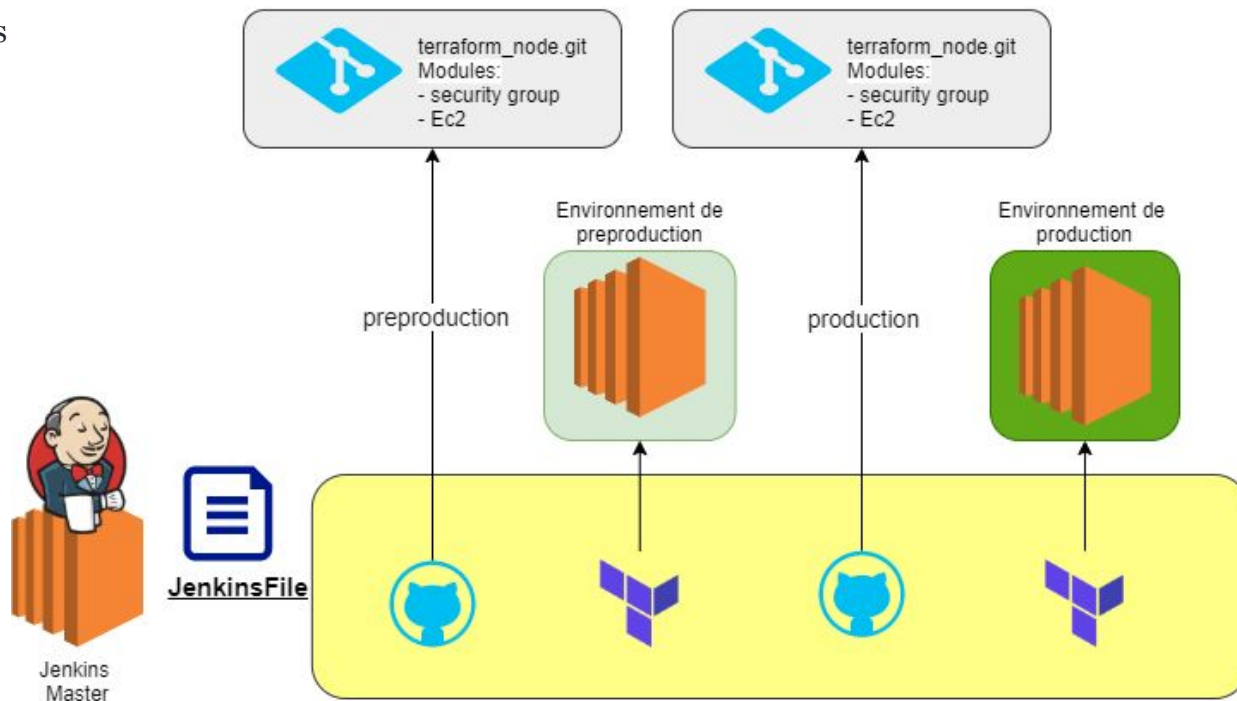


Terraform (Architecture)

Installation de Terraform sur Jenkins

Déploiement automatique sur AWS

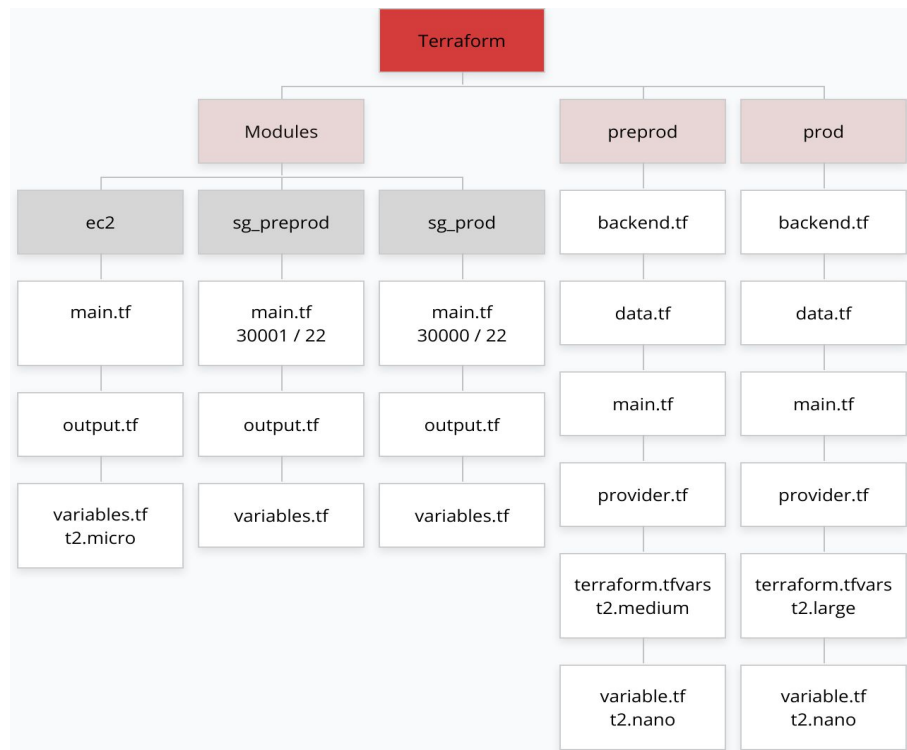
- Serveur de preproduction
- Serveur de production



Terraform (Repository)

Code Terraform réutilisable:

- **Modules Terraform:**
 - Security group Pre-Production
 - Security group Production
 - ec2
- **Objectifs de notre module:**
 - Créer deux instance Ec2
 - Pre Production (preprod/main.tf)
 - Production (prod/main.tf)
 - Créer bucket S3 (Backend)



Environnement de pré-prod

- Installation de Ansible
- Création d'un rôle Ansible "docker_role"
 - (Installation de Docker)

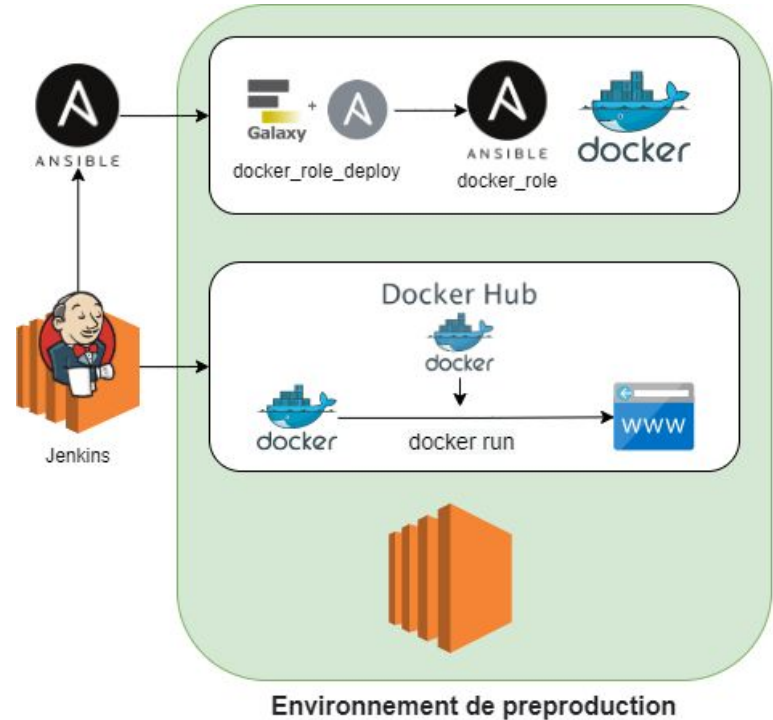
=> https://github.com/projetajc-group3/docker_role.git

- Déploiement et lancement du rôle

=> https://github.com/projetajc-group3/docker_role_deploy.git

```
ansible-galaxy install -r docker_role_deploy/roles/requirements.yml
ansible-playbook -i hosts.yml docker.yml
```

- Lancement du conteneur avec Image Docker de l'application



Environnement de prod

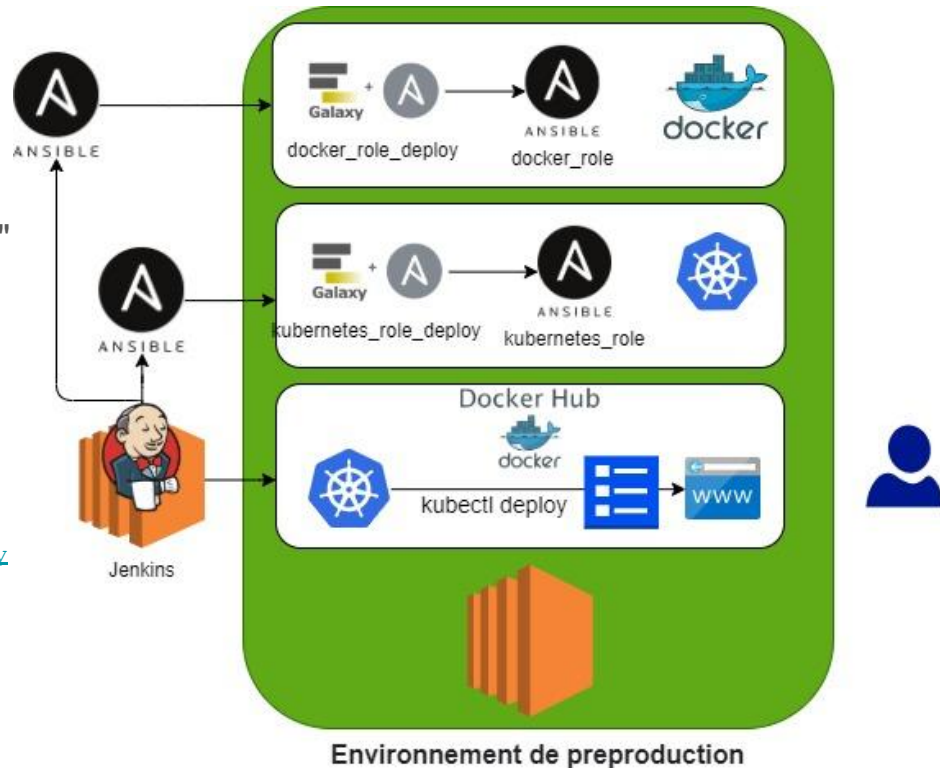
- Déploiement Rôle Ansible "docker_role"
- Déploiement Rôle Ansible "kubernetes_role"

Depuis ce manifeste installer Kubernetes

- `kubernetes_role_deploy/requirements.yml`
- `- src: kubernetes_role.git`

https://github.com/projetaic-group3/kubernetes_role_deploy

https://github.com/projetajc-group3/kubernetes_role



Kubernetes

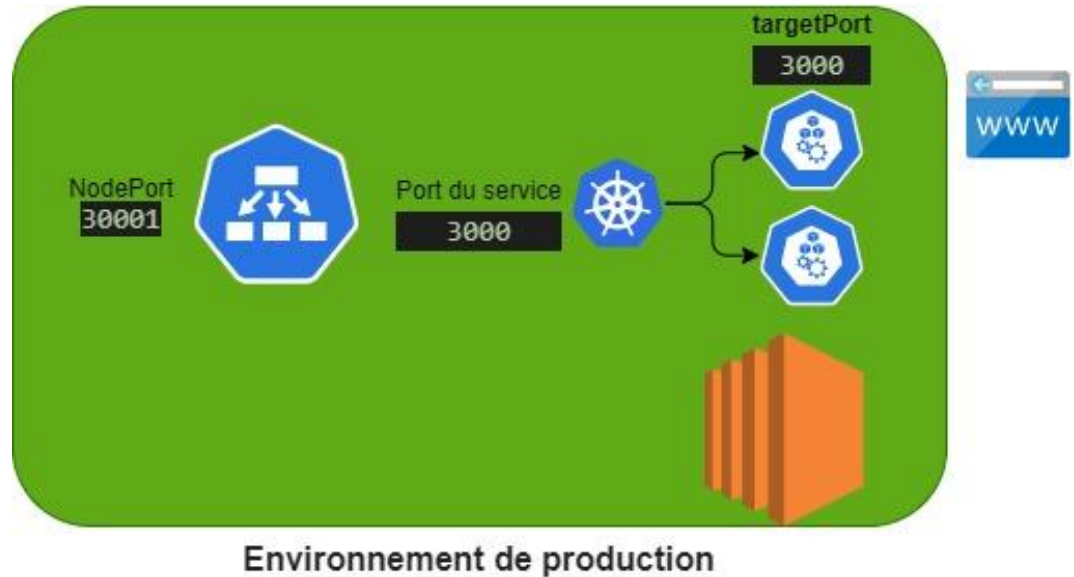
Le rôle installe kubernetes et un template

Déployer Node app

- templates/nodeapp.yml.j2
- manifest déploiement application Node

Caractéristiques Deployment

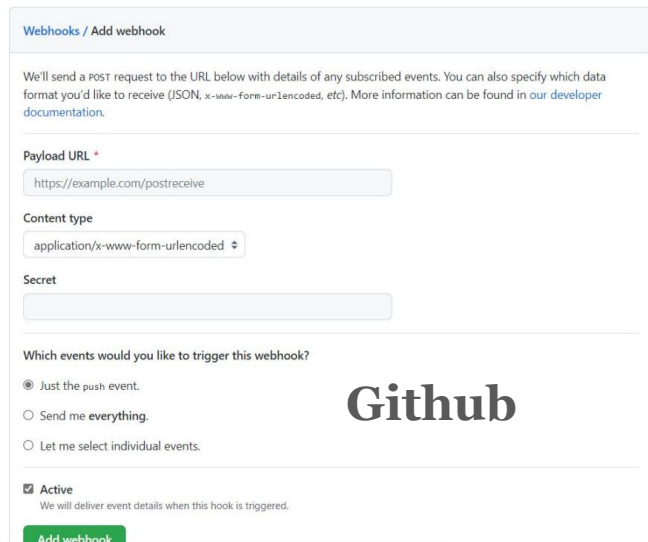
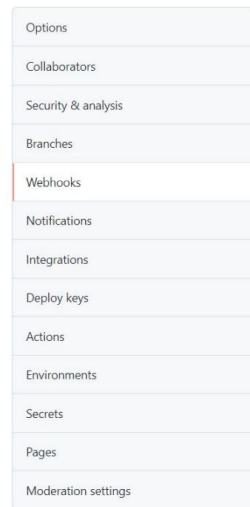
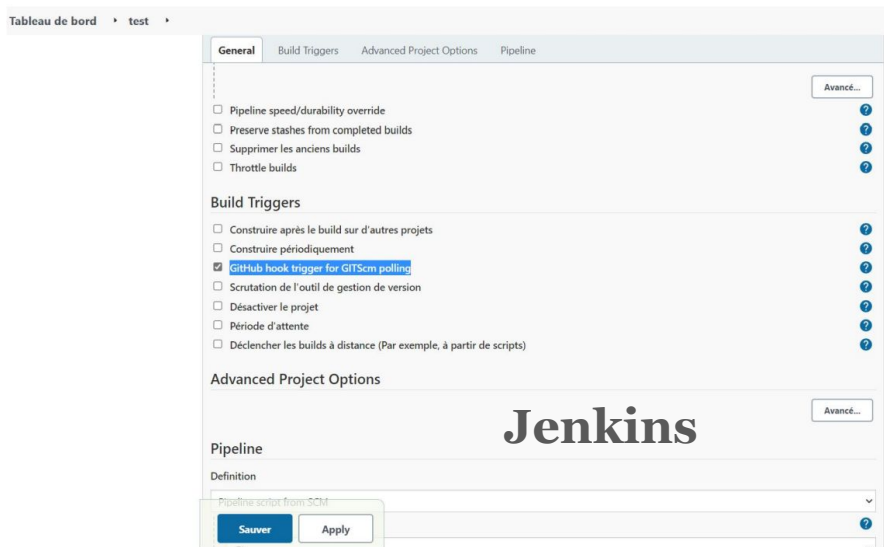
- 2 replicas
- Service de type Nodeport



Webhook trigger

- Configuration de Jenkins-server et de notre projet
- Configuration de notre repo Github

=> Déployer automatiquement une nouvelle version de l'application après chaque modification au niveau du code source => déploiements plus récurrentes et réduire le time to market



Notifications slack

- Configuration de Jenkins server
- Configuration de notre compte slack créé pour ce projet

=> Recevoir des notifications après le déploiement en pré-prod => on teste l'application sur le web=> on valide le déploiement en prod

Une fois le déploiement terminé en prod, nous recevons une autre notification sur slack



jenkins APPLI 17 h 29

STAGING APPLICATION PENDING VALIDATION ON <http://107.21.172.98:30001>

VALIDATION : Job 'projetajc-group3 [2]' (<http://54.174.144.82:8080/job/projetajc-group3/2/>)



jenkins APPLI 17 h 36

SUCCESSFUL: Job 'projetajc-group3 [2]' (<http://54.174.144.82:8080/job/projetajc-group3/2/>)

APPLICATION AVAILABLE ON <http://44.202.23.129:30000>

DÉMONSTRATION

- **github:** https://github.com/projetajc-group3/projetajc_node
- **Jenkins:** <http://54.174.144.82:8080/job/projetajc-group3/>
- **DockerHub:** <https://hub.docker.com/repository/docker/projetajcgroup3/node>
- **Slack:** <https://app.slack.com/>

Axes d'amélioration

- **Améliorer le manifeste kubernetes en ajoutant une stratégie de mise à jour pour ne pas détruire les deux pods au mm moment**
- **Simuler des attaques pour étudier les axes d'améliorations**
- **Pousser l'image sur un registry privé (un server gitlab par exemple)**
- **Ajouter un stage monitoring dans notre pipeline**
- **Les stages au niveau de Jenkinsfile pourraient mieux gérer les erreurs potentiels avec plus de gestion d'erreurs**

