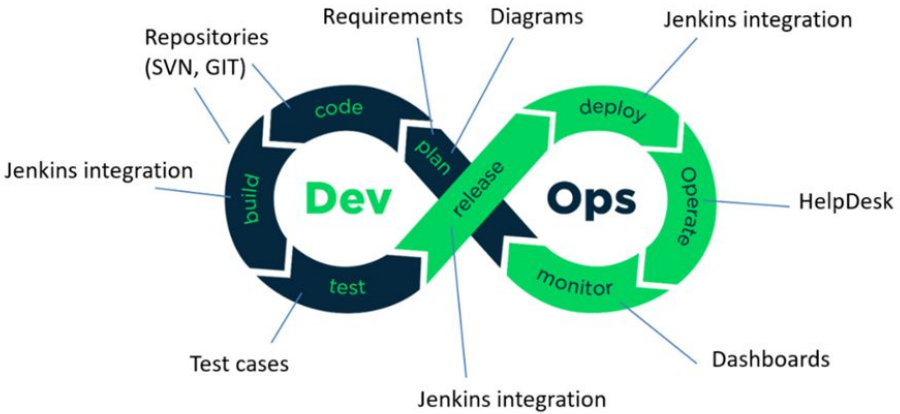
# Mise en place d'une chaîne complète CICD pour une application Node EJS

**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

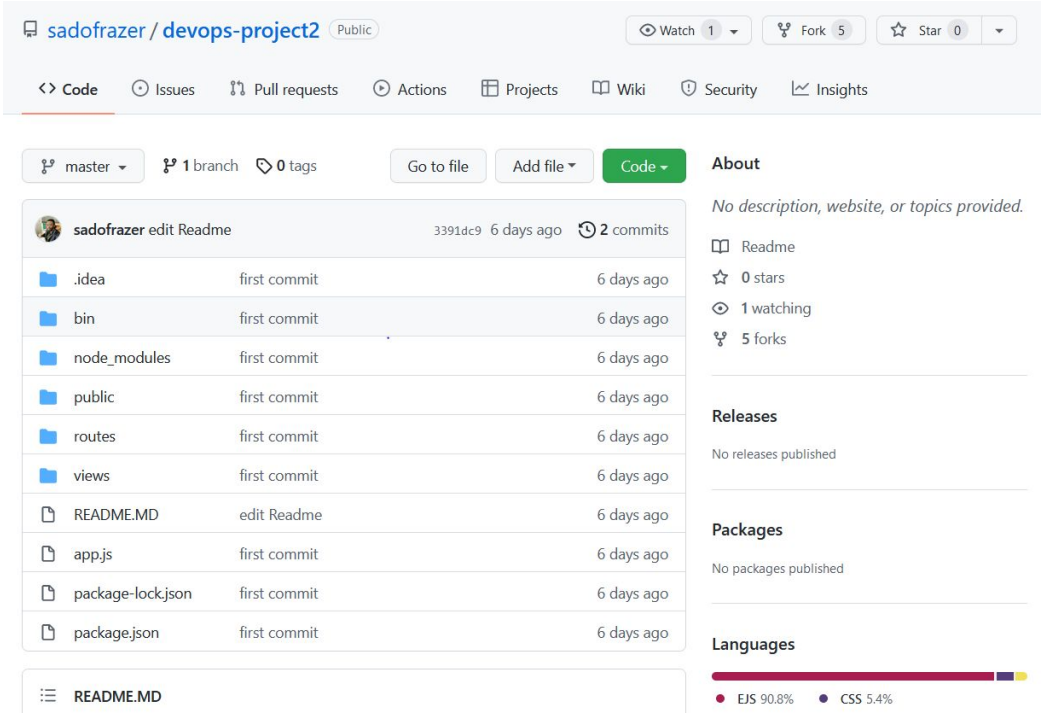
Consultant Devops
Artisan
Consultant CRM

# Analyse du projet

**Devops-projet2:**

- **Fichier ejs (embeded javascript)**
- **Modules node**
- **Package json (manifest)**

**=> site web à déployer**

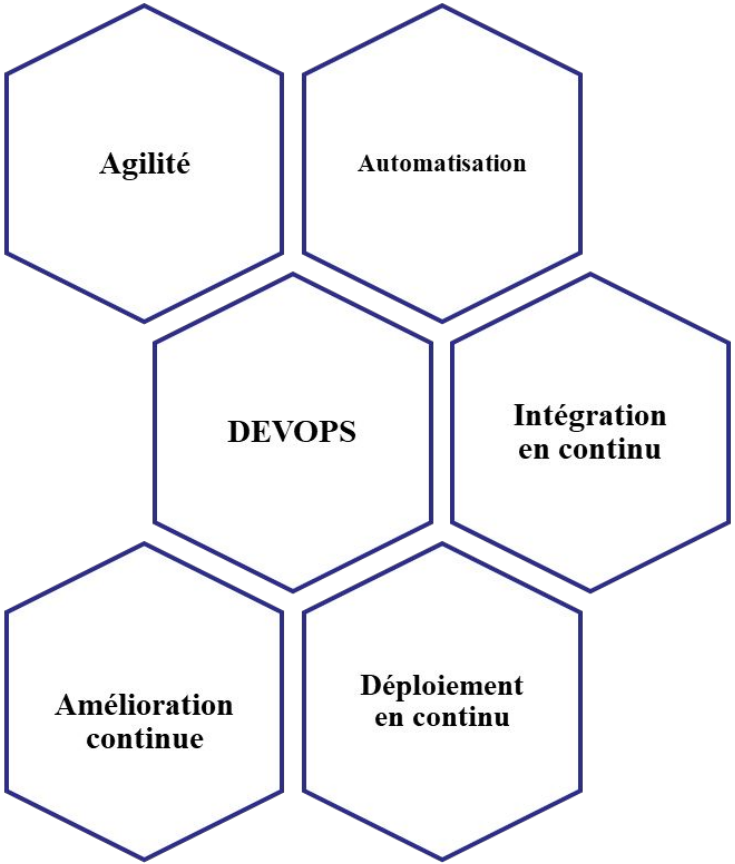



**https://github.com/sadofrazer/devops-project2.git**

# 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**

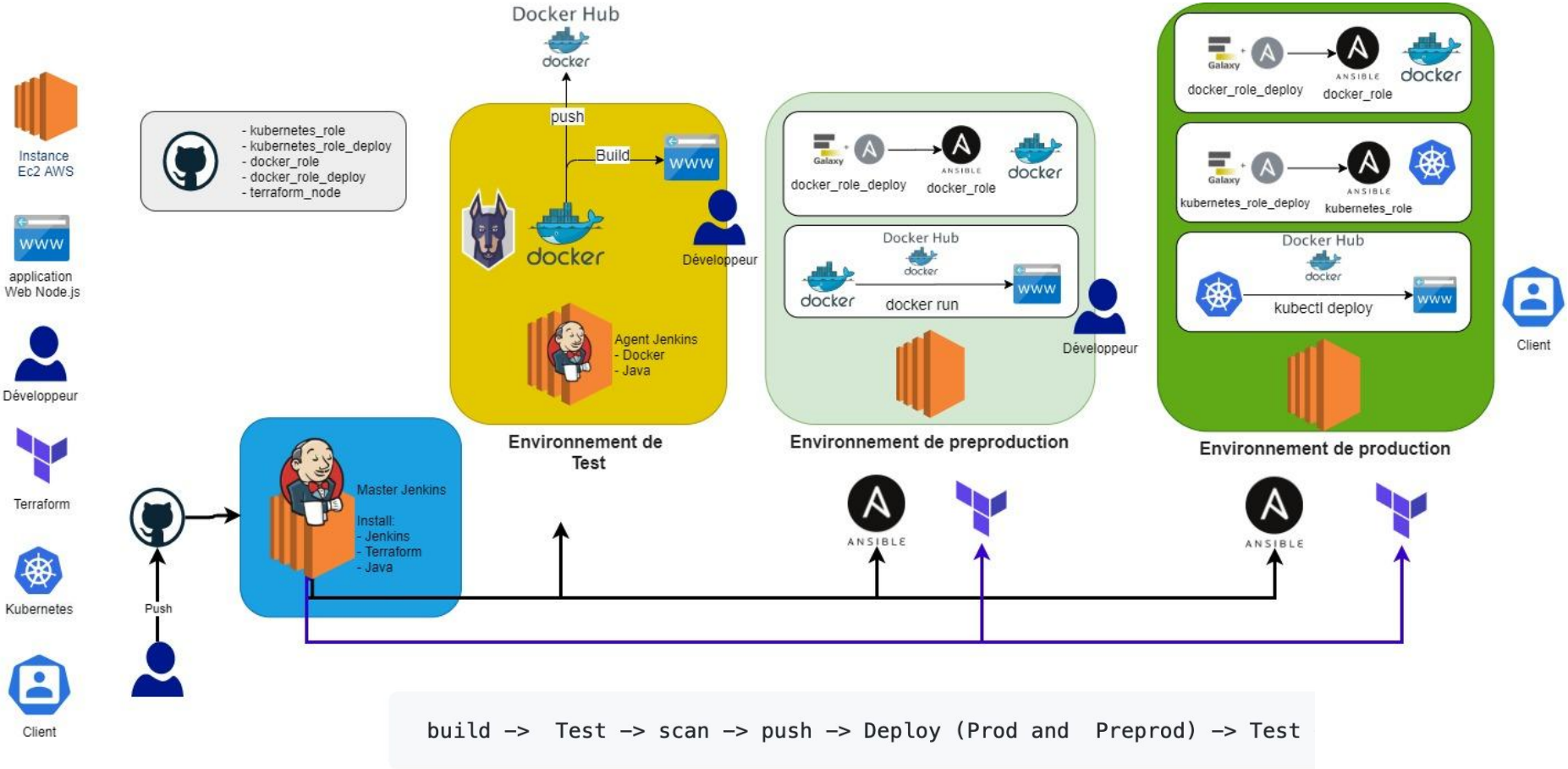
# Outils utilisés


- **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**

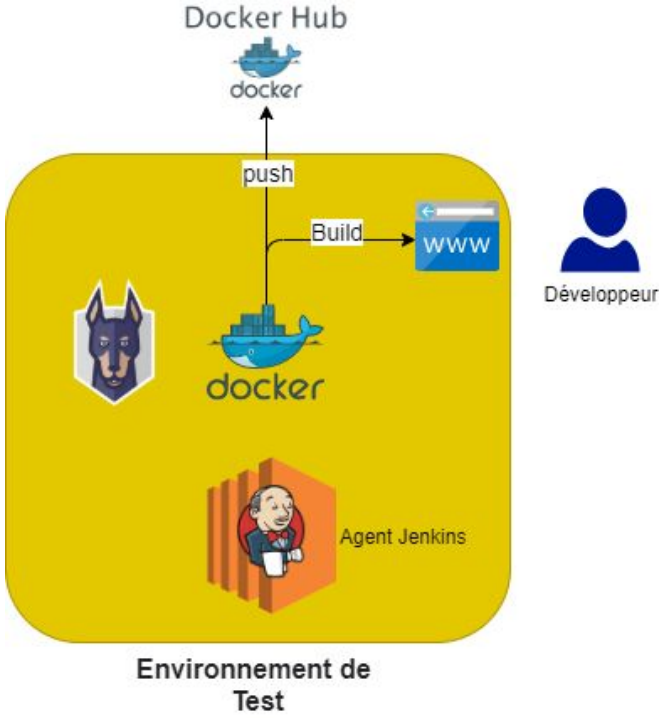
# Infrastructure & Workflow




```
build ->  Test -> scan -> push -> Deploy (Prod and  Preprod) -> Test
```

# Environnement de test

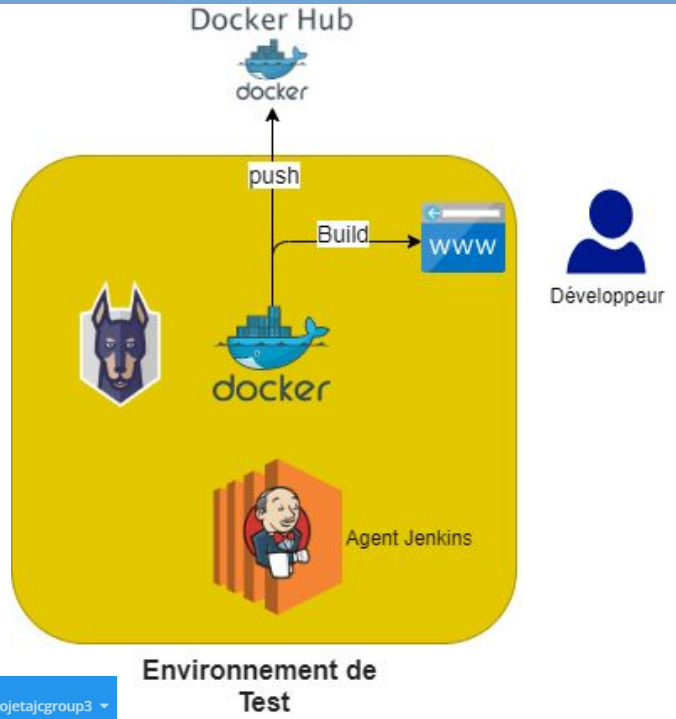
- Création d'un agent Jenkins : “Agent-test”
- Construire une image à l’aide d’un dockerfile
- Tester le fonctionnement de l'application

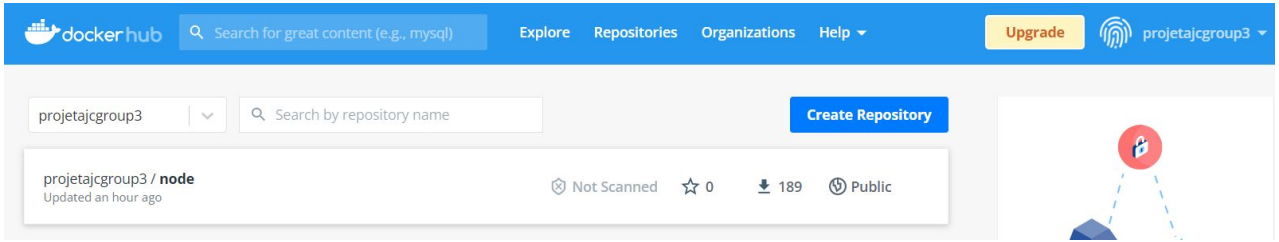



# Environnement de test

- Scan de sécurité et de vulnérabilité
  => Snyk Security Report

- Push vers Docker Hub
  => Sauvegarder notre artefact



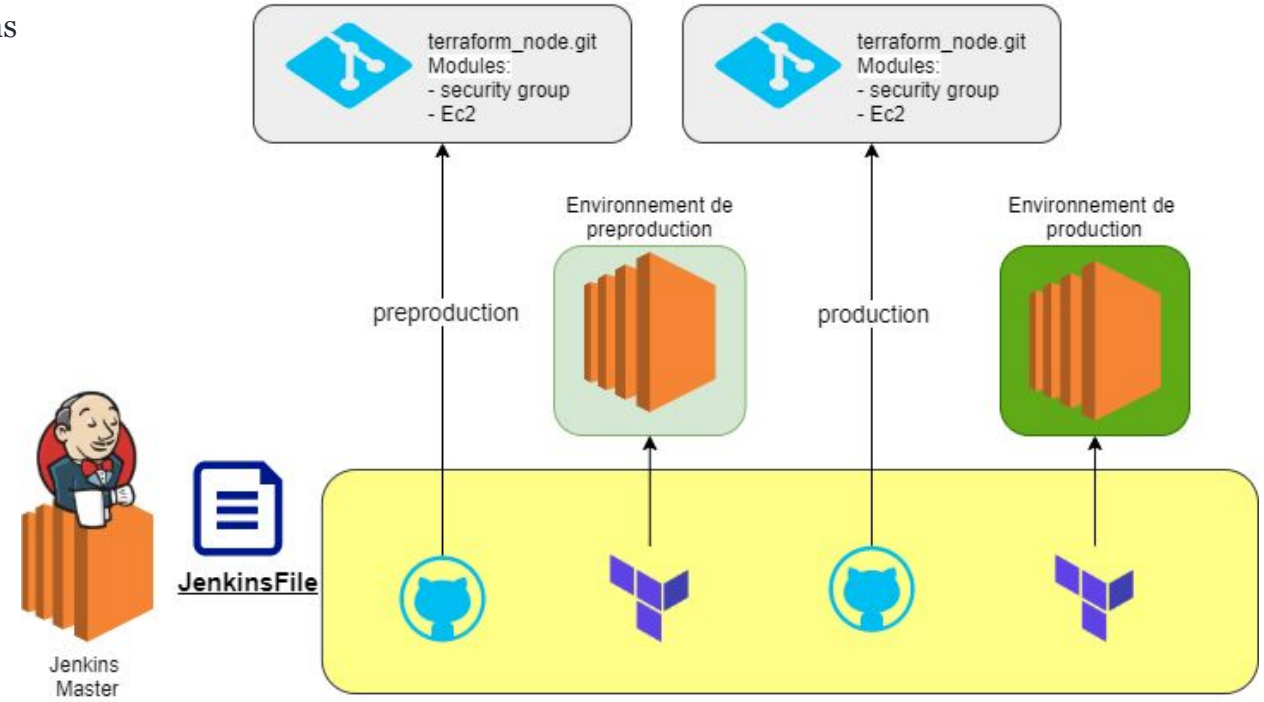



# Terraform (Architecture)

Installation de Terraform sur Jenkins

Déploiement automatique sur AWS

- Serveur de preproduction
- Serveur de production

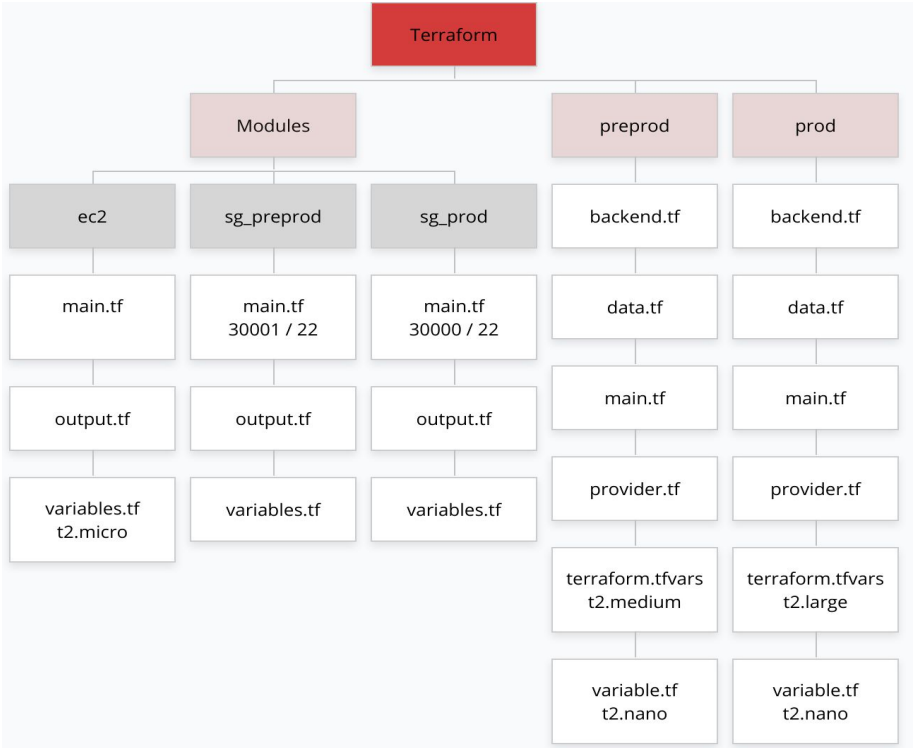


# Terraform (Repository)

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)

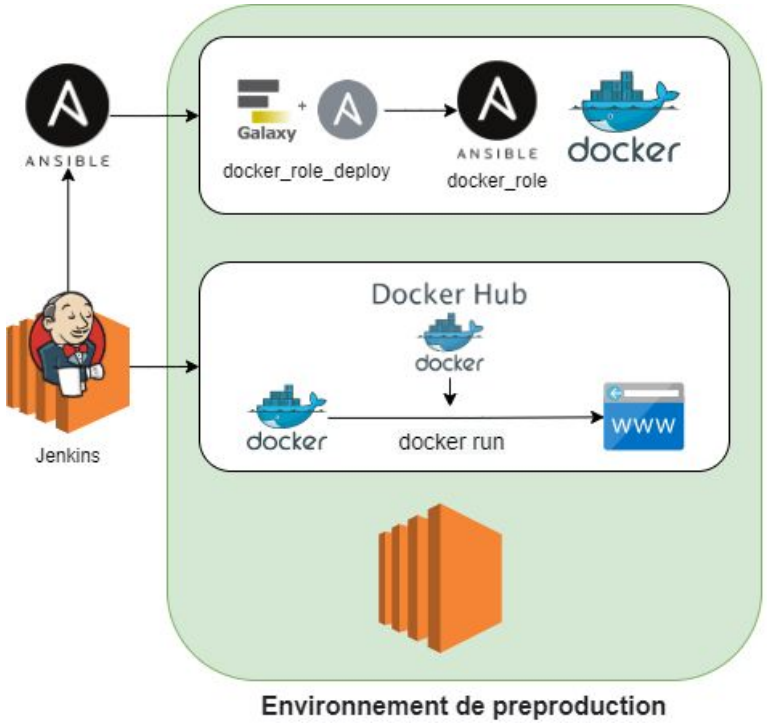
  => https://github.com/projetajc-group3/docker_role.git

- 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"

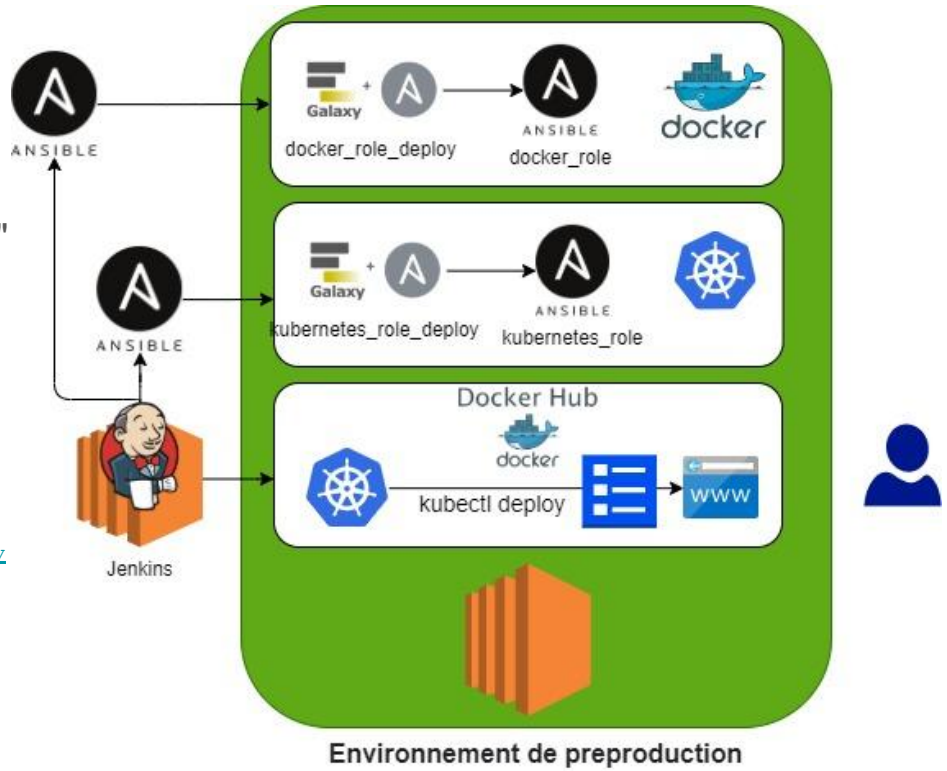
- Déploiement  Rôle Ansible "kubernetes_role"

Depuis ce manifeste installer Kubernetes

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

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

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



# Kubernetes

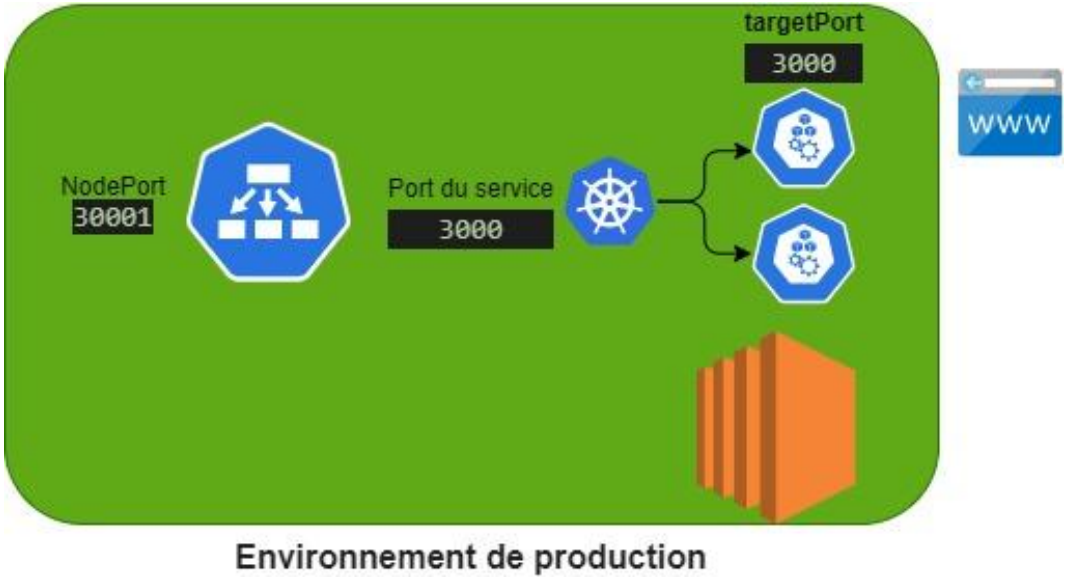
Le rôle installe kubernetes et un template

## Déployer Node app

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

Caractéristiques Deployment

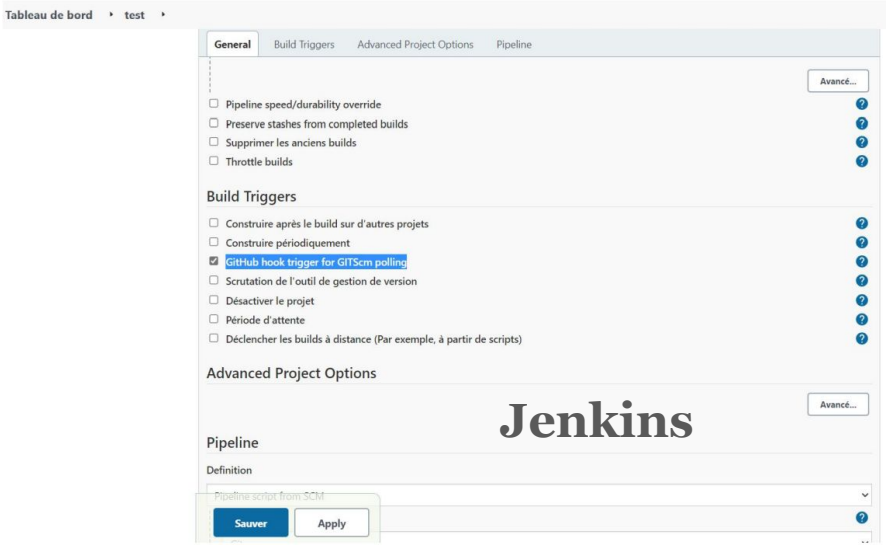
- 2 replicas
- Service de type Nodeport

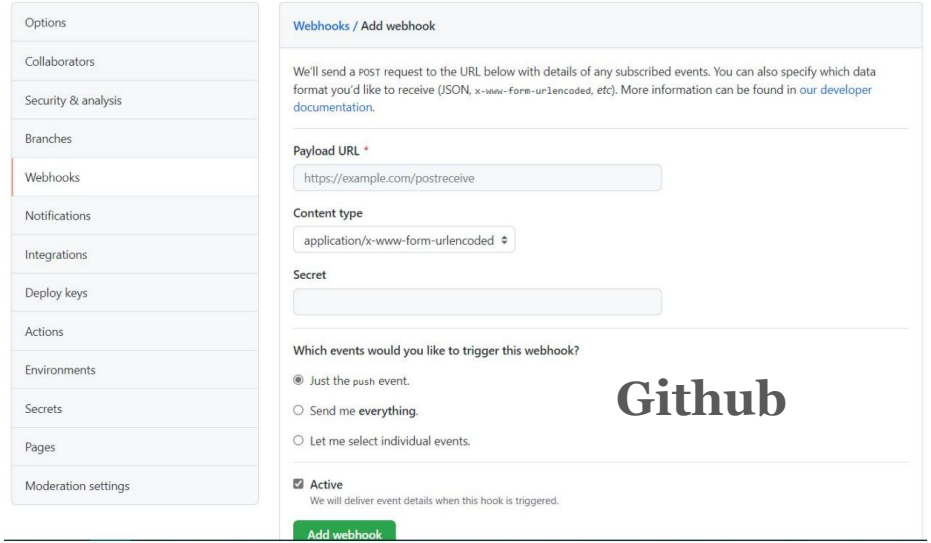


# Webhook trigger

- 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

# 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**


Merci de
votre
attention