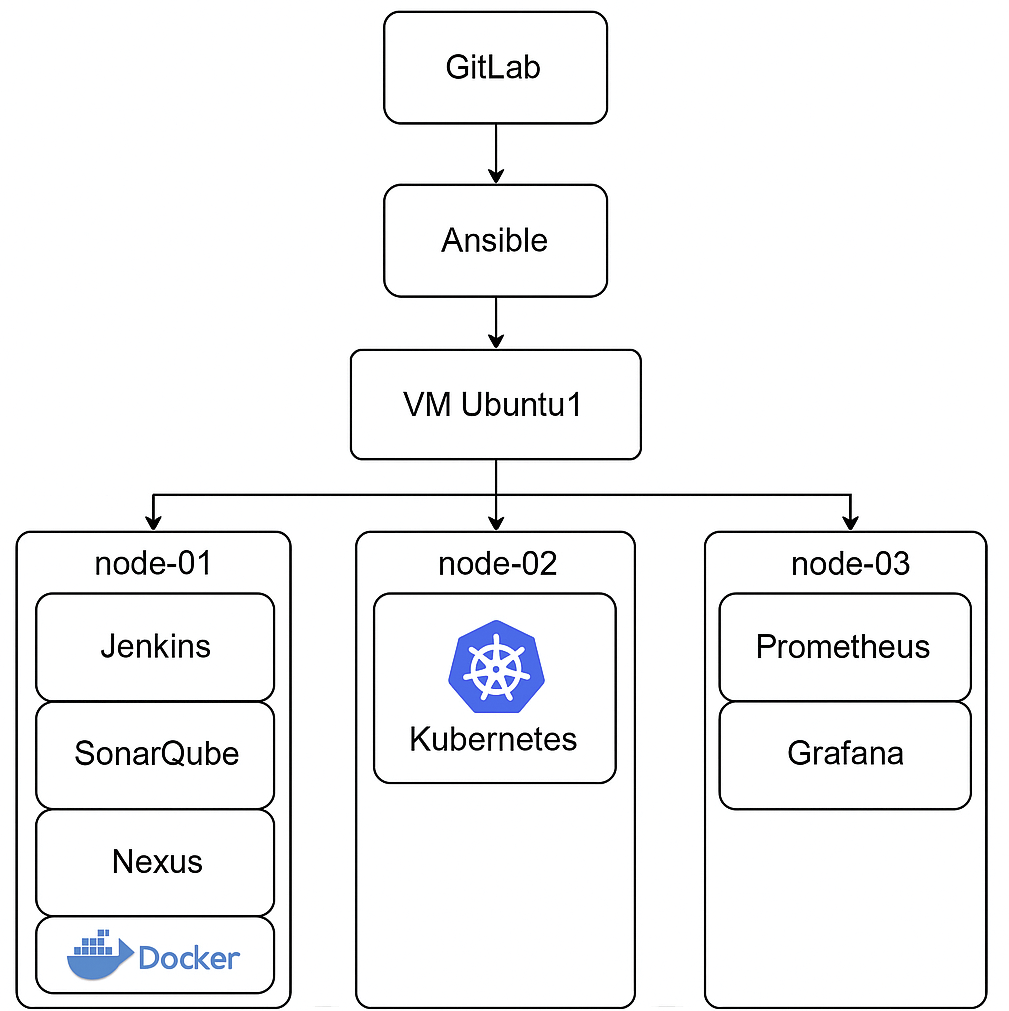
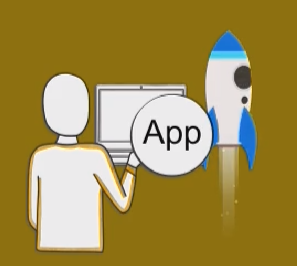
L’objectif de mon projet est de concevoir et de mettre en œuvre une chaîne d’intégration et de déploiement continus (CI/CD) pour une application web composée d’un front-end et d’un back-end. Ce projet vise à automatiser les étapes de build, test, analyse de code, stockage des artefacts, déploiement et supervision, en s’appuyant sur des outils DevOps tels que Jenkins, SonarQube, Nexus, Docker, Kubernetes, Prometheus et Grafana. Cette démarche permet de fiabiliser les déploiements, d'améliorer la qualité du code, et de faciliter le suivi du bon fonctionnement des services en production.

1. Liste des compétences acquises dans mon projet :

* Définir les objectifs du projet et ses livrables
* Identifier les technologies principales à utiliser (GitLab, Ansible, Docker, Kubernetes, Jenkins, etc.)
* Établir un diagramme d'architecture du système cible
* Création et gestion d’un dépôt GitLab personnel
* Installer et configurer Ansible sur une machine de contrôle
* Écrire des playbooks Ansible pour la configuration de serveurs Linux
* Automatiser l'installation de Docker avec Ansible
* Installer Docker sur les machines cibles
* Construire et tester une image Docker personnalisée
* Configurer un registre Docker (ex : Docker Hub) pour stocker les images
* Installer un cluster Kubernetes (en local)
* Créer des fichiers YAML pour définir des déploiements Kubernetes
* Configurer des services Kubernetes (NodePort)
* Déployer une application conteneurisée sur Kubernetes
* Installer Jenkins sur un serveur dédié
* Configurer un pipeline Jenkins pour builder des images Docker
* Automatiser les tests
* Déployer les applications directement depuis Jenkins vers Kubernetes
* Installer et configurer un serveur SonarQube
* Configurer l’intégration de SonarQube dans le pipeline Jenkins
* Analyser la qualité du code de l’application avec SonarQube
* Stockage d’artefacts et d’images Docker
* Installer et configurer Prometheus dans le cluster Kubernetes
* Configurer des alertes basées sur les métriques Prometheus
* Installer Grafana pour visualiser les données de Prometheus
* Créer des dashboards Grafana pour surveiller les performances des applications
* Documenter les procédures de déploiement et de maintenance
* Valider la conformité du système avec les critères du titre professionnel



3- TABLEAU ADRESSE IP OU ENVIRONNEMENT DE TRAVAIL (local, cloud, vm, docker etc…)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nom de la machine / Service** | |  | | --- | | **Adresse IP** | | |  | | --- | | **Rôle / Fonction** | | |  | | --- | | **Environnement** | |
| vm-ubuntu1 (poste de contrôle) | 192.168.8.142 | Machine de gestion avec Ansible | VM locale (VMware) |
| Node-01 | 192.168.8.147 | Jenkins, SonarQube, Nexus (via Docker) | |  | | --- | |  |  |  | | --- | | VM – Kubernetes | |
| Node-02 | 192.168.8.148 | Nœud Kubernetes | VM – Docker |
| Node-03 | 192.168.8.149 | Supervision avec Prometheus & Grafana | Machine physique – Docker |
| |  | | --- | | Machine physique (GitLab) |  |  | | --- | |  | | 192.168.227.1 | Hébergement de GitLab (projet DevOps) | Docker (Node-01) |
| |  | | --- | | Jenkins |  |  | | --- | |  | | 192.168.8.147:8080 | Intégration et déploiement continu (CI/CD) | Docker (Node-01) |
| SonarQube | 192.168.8.147:9000 | |  | | --- | |  |  |  | | --- | | Analyse de la qualité du code | | Docker (Node-01) |
| Nexus | 192.168.8.147:8081 | Stockage des artefacts et images Docker | Docker (Node-01) |
| Prometheus | 192.168.8.149:9090 | Supervision des métriques système/K8s | Docker (Node-03) |
| Grafana | 192.168.8.149:3000 | Visualisation des métriques (dashboards) | Docker (Node-03) |

1. L’application concernée par la chaîne DevOps est une application web développée pour une **association**. Elle est composée de deux parties :

* **Un back-end** développé en Java, initialement déployé sur un serveur Tomcat, puis conteneurisé avec Docker et orchestré via Kubernetes.
* **Un front-end** que j’ai également pris en charge, déployé de manière automatisée à l’aide de Docker et Kubernetes.

Cette application associative a servi de base pour la mise en place d’une chaîne CI/CD complète. L’objectif était d’automatiser les étapes de build, de test, d’analyse de code, de stockage des artefacts, de déploiement, puis de superviser le bon fonctionnement des services via Prometheus et Grafana.

1. Technologies utilisées dans le projet

Développement de l’application :

* Serveur d’application : Tomcat (pour le déploiement initial)
* Front-end : Servi via **Nginx** (utilisé comme serveur web pour les fichiers statiques et pour le reverse proxy)

Chaîne CI/CD :

* Outil d’intégration continue : Jenkins
* Analyse de la qualité de code : SonarQube
* Gestionnaire de référentiels : Nexus (pour les artefacts et images Docker)
* Contrôle de version : Git (hébergé sur GitLab)

Automatisation et configuration :

* Outil d'automatisation : Ansible (utilisé pour l’installation de Docker, Kubernetes, et la configuration des machines)

Conteneurisation et orchestration :

* Conteneurs : Docker
* Orchestrateur : Kubernetes

Supervision et monitoring :

* Collecte des métriques : Prometheus
* Visualisation : Grafana

Infrastructure :

* Environnement de travail : VMs locales créées sous VMware
* Hébergement GitLab : Machine physique personnelle

1. Que fait chaque techno de manière concrète ?

Application à déployer

* Application front-end (statique) : mise à disposition via Nginx, qui sert de serveur web pour distribuer les fichiers du front-end.
* Application back-end : une application existante, fournie dans le cadre du projet, que j’ai intégrée et déployée sans la développer moi-même.

Intégration et déploiement continu (CI/CD)

* Git (GitLab) : utilisé pour héberger le code source de l’application et déclencher automatiquement les pipelines CI/CD.
* Jenkins : utilisé pour automatiser les étapes du pipeline, notamment le build, l'analyse de code (via SonarQube), le push d’artefacts dans Nexus, et le déploiement dans Kubernetes.
* SonarQube : outil d’analyse statique du code permettant de vérifier la qualité, la maintenabilité et les vulnérabilités dans le code source existant.
* Nexus : utilisé pour stocker les artefacts (comme des fichiers .war ou images Docker), en tant que repository privé.

Automatisation et configuration

* Ansible : utilisé pour automatiser l’installation des outils (Docker, Kubernetes, etc.) sur les différentes machines, et assurer une configuration homogène des environnements.

Conteneurisation et orchestration

* Docker : utilisé pour conteneuriser les composants de l’application et les services associés (Jenkins, SonarQube, Nexus, Prometheus, etc.).
* Kubernetes : utilisé pour orchestrer le déploiement des conteneurs, assurer leur haute disponibilité, et faciliter la gestion des services sur les différents nœuds.

Supervision et monitoring

* Prometheus : outil de collecte de métriques utilisé pour surveiller l’état des services, des conteneurs, et des ressources système.
* Grafana : utilisé pour visualiser les métriques collectées sous forme de tableaux de bord personnalisés.