Mise en œuvre d'un projet DevOps

Objectifs :

Notre objectif est de créer d'une équipe DevOps par projet aux allures et aux compétences diverses : des développeurs, des testeurs, des exploitants, des chefs de projets. C'est l'équipe toute entière qui est responsable et qui est chargée d'effectuer le déploiement continu.

1) Outils permettant la mise en œuvre de DevOps

Pour réussir un bon DevOps, il faut également des outils technologiques qui interviennent dans l'UDD (Usine De Développement)

- **Gestion du développement** : la production du code s'effectue avec des outils propres aux développeurs (IDE, Framework et extensions, DB, Paas, …)

- **Gestion du stockage du code** : le code doit être poussé sur dépôt central permettant la mutualisation entre les développeurs d'une même équipe. Des outils comme Git, GitLab, GitHub, Bitbucket, CVS, ou Mercurial peuvent ainsi être utilisés.

- **Gestion de l'intégration et Déploiement continue (CI/CD):** Le code doit générer automatiquement des builds à l'aide d'un gestionnaire d'intégration continue comme Jenkins (fork de Hudson), Ansible, TeamCity, CruiseControl ou Tinderbox. Avec Ansible ou Jenkins, le plugin buildbreaker permet de stopper la création du build si les analyses Sonar de qualité de code ne sont pas bonnes, au regard des critères de mesures choisies.

- **Gestion de la qualité de codes :** de nombreux outils comme whiteSource Bolt SonarQube ou Jacoco peuvent être utilisés. Ils permettent d'effectuer des analyses de codes au plus tôt dans le développement et servent à des fins d'amélioration continue.

- **Gestion des tests :** les tests unitaires ont des outils de la famille xUnit comme Junit (monde Java) et PHPUnit (monde PHP), JSUnit, PyUnit, et Test : More pour effectuer des TUs. Les tests métiers disposent d'outils comme Selenium, Behat, Cucumber, RFT (IBM), QF Test (Quality First Software), SilkTest (MicroFocus), Unified Functionnal Testing (UFT). D'autres types de tests doivent être effectués comme les tests de sécurité (OWASP, etc.), les tests de performances (JavaMelody, outils d'APM, JMeter, etc.).

- **Gestion du Monitoring/Analytics :** pour le monitoring plusieurs outils sont disponible AppDynamics, InfluxData, New Relic, Sensu, Nagios, BigPanda, DataDog, Dynatrace and more... et pour l'Analytics: Splunk and more... mais pour le Chat: Slack, HipChat, IRC and more...

2) Gestion de projets et collaboration : (Redmine, Atlassian, etc.)

D'autres outils complètent la panoplie et permettent d'augmenter l'automatisation et ainsi d'améliorer la productivité :

• **Les plateformes IaaS** (CloudForms, vRealize, AWS Ec2, Google App Engine, Mirosoft Azure, etc.) qui vont permettre de provisionner en automatiques les VMs nécessaires au fonctionnement de l'application.

• **Les plateformes PaaS** (Openshift, BlueMix, Cloud Foundry, Azure, etc.)

• **Les gestionnaires de configuration** (Puppet et Chef, Ansible)

• **Les gestionnaires de containerisation** (Docker, kubernetes services)

Projet fil rouge m2i formation

Introduction

Ce projet consiste à l'installation des 3 outils : Ansible, Jenkins et Docker;

Une machine virtuel "leader" sera créé avec Vagrant.

A partir des playbooks lancés sur la machine leader, on peut installer les autres logiciels sur les autres machines.

Les jobs sur jenkins font appel à des playbooks pour le déploiement des conteneurs docker.

Environnement avec Vagrant

Vagrantfile crée 4 machines virtuelles :

- leader : avec jenkins, ansible, docker et maven
- Vm pour nexus : pour le repository nexus et également le serveur sonar
- Vm pour gitlab : pour l'installation du gitlab (optionnel)
- VM cible[1:3] : infra cible pour le déploiement des conteneurs avec ansible depuis le leader.

Pour lancer les machines il suffit de faire un `vagrant up` Utiliser `vagrant provision leader` pour relancer les playbooks ansible. Le playbook lancé au démarrage de la machine est le playjenkins.yml qui contient les rôles suivants :

```
- java
- docker
- registry
- maven
```

Les playbook ansible

Ansible est le moyen le plus simple de déployer, Il vous donne le pouvoir de déployer des applications à plusieurs niveaux de manière fiable et cohérente. Vous pouvez configurer les services nécessaires ainsi que les artefacts d'application push à partir d'un système commun.

Dans le dossier ansible de la racine du projet se trouvent tous les fichier .yml plus les rôles respectives. Ce dossier est présent dans toutes les machines virtuelles sur /vagrant/ansible.

Pour lancer un playbook ansible il suffit de faire la commande `ansible-playbook /vagrant/ansible/<play-nom>` depuis la machine leader

Les différents playbook sont :

- playjenkins.yml
- playnex.yml, avec les rôles : docker, nexus et sonar (optional)
- playgit.yml, avec les rôles : docker et gitlab
- tomcat-deploy.yml, qui déploie un conteneur tomcat d'une image crée avec jenkins (voir configuration jenkins). **Remplacé par docker swarm**
- playswarm.yml, qui s'en charge de l'initiation du docker swarm avec leader en tant que manager et les cibles en tant que workers.
- swarm-deploy.yml, qui s'en charge du déploiement de l'image crée avec jenkins en mode service sur le swarm docker

Configuration jenkins

Après lancement de toutes les machines il faut configurer Jenkins pour la création d'un nœud ciblant la machine leader (configuration par ssh avec les clés publiques)

Le dossier jee-projet contient les fichier (dont Dockerfile) qu'il faut mettre sur GitLab
Variable : Utiliser le ficher Dockerfile dans le dossier docker-build

BUILD job sur jenkins avec commande bash:

```
mvn clean package #ou deploy
docker build (-f /path/to/Docker/file) -t 192.168.33.101:5000/tomcat-deploy .
docker push 192.168.33.101:5000/tomcat-deploy
```

RUN job sur jenkins avec déclenchement après job build :

```
ansible-playbook /vagrant/ansible/tomcat-deploy.yml
```

ou

```
ansible-playbook /vagrant/ansible/swarm-deploy.yml
```