LA SOLUTION CLÉ EN MAIN DEVSECOPS "AUTOMATISATOR"

Contexte

Pendant cette période de confinement, Campus Academy nous a proposé un workshop nommé Campus Track, d'une durée de deux semaines mélangeant les filières et les niveaux d'étude. Le but de ce track est de nous permettre de réinvestir nos compétences en situation professionnelle de collaboration et nous amener à concevoir, produire des réponses concrètes et fonctionnelles en lien avec nos domaines d'expertises respectifs.

Durant cet espace de temps, 7 projets (2 Projets Ops et 5 Projets Dev) ont été proposés pour un peu après 100 équipes composées de 3 à 4 personnes. Concernant notre équipe, il nous a été demandé de répondre aux besoins d'une association nommée la Cité de l'Environnement, qui souhaite **mettre en place une chaîne DevSecOps** tout en tenant compte de l'**aspect écologique**. Autant vous dire qu'on avait du pain sur la planche. Plus de détails sur les besoins du projet sur ce pdf .

À l'issue de cette épreuve et parmi tous ces projets, nous en sommes sortie gagnant en reportant le titre du projet favoris de Campus Track, on ne pouvait pas espérer mieux .

À travers cet article, je vous propose de décrire un peu plus en détails notre solution et à la fin de celui-ci je partagerai avec vous notre document d'architecture qui était un parmi des autres livrables attendus par le client.

La solution DevSecOps proposée

Le nom du projet

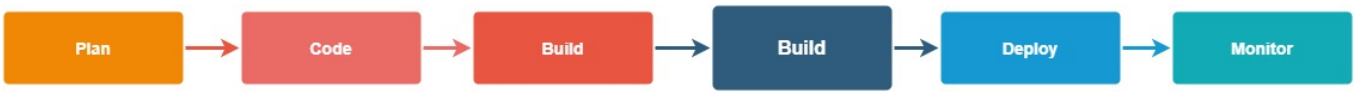
Durant les premières journées, nous avions passé beaucoup de temps à mieux **comprendre les besoins et exigences du client** via le logiciel de visioconférence WebEx de Cisco ou bien l'utilisation Slack par écrit. Nous avions aussi profité de cette journée pour mieux se connaître entre membres de l'équipe et assigner nos tâches respectives sur Trello. De ce fait, nous avions choisi de nommer notre solution "Automatisator". Je vous avoue que pour le nom de l'équipe, nous n'avions pas cherché très loin, la première chose qui nous aient venu à l'esprit est le film "Terminator" et nous avions décidé de fusionner ce nom avec le mot « automatisation » ce qui nous a donné le fameux « Automatisator ». Sans oublier bien sûr que l'automatisation est l'un des piliers fondamentaux du modèle DevOps . Bref, vous ne serez donc pas surpris si vous rencontrez ce nom dans notre document d'architecture.

Objectifs de la solution

L'objectif des outils de notre solution est de **rationaliser**, **raccourcir** et d'**automatiser** de façon **sécurisée** davantage les différentes étapes du flux de travail de livraison de logiciels (ou « pipeline »). Les outils proposés promeuvent également les **principes de base de la philosophie DevSecOps**, c'est-à-dire : l'automatisation, la collaboration et l'intégration entre les équipes de développement, sécurité et d'exploitation.

Fonctionnement de la solution

Nous avons choisi de séparer notre solution sous forme de six étapes consécutives :



Plan

Nous avons choisi cette phase afin d'aider l'entreprise à définir les besoins en matière d'infrastructure. Si elle souhaite par exemple installer une stack LAMP, ouvrir uniquement les ports 80/443, configurer des applications tierces et modifier les droits d'un fichier/utilisateur, sur une multitude de serveurs, sans nécessité de s'y connecter manuellement ? C'est tout ici le but de cette étape, car nul besoin qu'elle se connecte et exécute les commandes sur son parc de machines, elle exécutera à la place uniquement un fichier yaml depuis une machine maître qui s'occupera ensuite d'automatiser la configuration de ses machines cibles.

Cette prouesse technique est possible depuis l'utilisation de l'outil **Ansible** qui simplifiera à l'équipe d'opérations les tâches répétitives, complexes et fastidieuses et cela permettra à son équipe de gagner énormément de temps lorsqu'elle installera des packages ou configurera un grand nombre de serveurs.

Son architecture est simple et efficace. Il fonctionne en se connectant à ses nœuds et en y poussant de petits programmes. Ces programmes rendent le système conforme à l'état souhaité et, une fois les tâches terminées, elles sont ensuite supprimées.

Cet outil fonctionne sur SSH et ne nécessite aucun démon, serveur spécial ou bibliothèque pour fonctionner. Un éditeur de texte et un outil en ligne de commande sont généralement suffisants pour faire votre travail. Vous décrivez simplement votre infrastructure dans un fichier YAML puis toutes les informations sur l'état souhaité de ces machines sont organisées en playbooks. Il est également capable de rassembler des informations sur les nœuds (tels que les adresses IP ou les détails du système d'exploitation) dans ce que l'on appelle des Facts, qui aident à l'approvisionnement sélectif et automatisé de différentes configurations sur la plate-forme. Tout cela avec un langage très lisible par l'homme (le YAML) sans avoir

besoin d'écrire du code ou de déclarer des relations explicites.

> **Information**
>
> Pour ceux qui souhaitent apprendre cet outil, j'ai faits un cours complet sur Ansible !

Code, Build, Test, Deploy

Les étapes suivantes restent différentes les unes des autres mais nous avons choisi de les centraliser autour d'un seul outil appelé **Gitlab**.

Code

Cette phase implique la conception de logiciels et la création de code logiciel. Leurs équipes auront besoin de stocker et de suivre les modifications apportées de leur code. Cela est possible depuis l'outil Git qui est le système de contrôle de versions le plus utilisé au monde. Il suivra les modifications apportées aux fichiers, ils auront ainsi un enregistrement de ce qui a été fait et ils pourront revenir à des versions spécifiques s'ils en ont besoin. Git facilitera également la collaboration, en permettant aux modifications de plusieurs personnes d'être fusionnées en une seule source.

Cependant, git ne peut s'exécuter qu'en local, ils auront donc besoin d'un outil qui concentre les fonctionnalités de git dans un même outil distant (on-premise ou cloud). Pour ce faire le choix s'est porté vers le GitLab, une plate-forme entièrement intégrée basée sur Git pour le développement de logiciels. Outre les fonctionnalités de Git, GitLab possède de nombreuses fonctionnalités puissantes pour améliorer leur flux de travail que nous utiliserons dans les prochaines étapes du cycle

DevSecOps.

Build et Test

Au cours de cette phase de Build, elle gérera les versions et versions logicielles sur différentes branches (une branche test pour les équipes de développement et une branche master pour les équipes d'exploitation) et utilisera l'intégration continue pour l'aider à compiler et à empaqueter automatiquement le code pour une version future en production.

Passons ensuite à la phase de Test qui impliquera des tests automatisés continus pour assurer une qualité de code optimale. Ils auront ainsi une partie dédiée aux tests unitaires (phpunit dans le cas du projet car l'application cible est codée en php) permettant ainsi de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module ») et des tests de vulnérabilités automatisés afin de vérifier la sécurité de votre code, dans ce cas nous proposons au départ la méthodologie SAST qui examinera le code pour trouver des failles et des faiblesses logicielles telles que l'injection SQL et d'autres attaques connues. Le but de cette étape est donc de minimiser ainsi le risque de bugs et de vulnérabilités.

Pour ces deux étapes, cette automatisation est possible depuis la fonctionnalité d'intégration continue (CI) qui est intégré nativement dans Gitlab, donc pas besoin d'installer un outil annexe tel que Jenkins et se retrouver ensuite étranglé par une multitude d'outils à gérer. Ses développeurs auront besoin de fusionner tout le code produit, et cette fusion aura généralement lieu plusieurs fois par jour dans un projet Gitlab partagé. À partir de ce même projet, des tests automatisés seront effectués pour garantir l'absence de problèmes d'intégration et l'identification précoce de tout problème avant son envoie en production.

Deploy

Une fois les tests d'intégration continus réussis, la prochaine étape est la phase de déploiement qui inclut des outils qui aident à gérer, coordonner, planifier et automatiser les versions de produit en production. Cet exploit technique est possible grâce à la pratique de la déploiement continue (CD) qui est inclus également nativement dans Gitlab et garantit la livraison du code validé par l'intégration continue (CI) à leur application au moyen d'un pipeline de déploiement structuré.

le CI et CD agissent ensemble pour accélérer la rapidité avec laquelle leurs équipes peuvent fournir des résultats à leurs clients et parties prenantes. CI les aidera à détecter et à réduire les bugs et failles au début du cycle de développement, et le CD déplacera plus rapidement le code vérifié vers leurs applications. CI et CD doivent fonctionner de manière transparente afin que leurs équipes puissent se constituer rapidement et efficacement, tout en étant essentielle pour garantir une pratique de développement entièrement optimisée.

Monitor

Cette phase sera gérée par les outils open source **Prometheus** et **Grafana**. Cette étape consiste à identifier et à collecter des informations sur les problèmes d'une version spécifique du logiciel en production et de surveiller les métriques et les logs de leurs différentes applications afin de découvrir l'impact sur l'expérience de l'utilisateur final.

Conclusion

Dommage, que nous avions manqué de temps pour perfectionner notre solution, j'aurais bien aimé intégrer de l'infrastructure as Code avec du Terraform afin

d'automatiser aussi la création de notre infrastructure (SPOILER, c'est mon futur cours). Mais voilà, malheureusement il a fallu gérer plusieurs autres tâches, préparer les différents livrables , gérer les rendez-vous clients, mettre en pratique et documenter notre solution.

Néanmoins, je souhaite partager avec vous ci-dessous notre **dossier d'architecture**, où vous trouverez plus de détails sur le fonctionnement de notre solution et comprendre ce qui a pu orienter nos choix vers certains outils, bonne lecture et bon confinement à tous !

Document d'architecture