DevOps, Cloud public et sécurité de la supply chain

OSSIR Juillet 2021

Agenda

Présentation

Transformation numérique, encore plus vite !

La supply chain et ses faiblesses

Recommandations et hygiène Cloud

Présentation

CTO @ Beeware / Qualys / 42Crunch / Stacksciences

Staff DevSecOps @ Cloudbees

Web application security / malware detection / API Security / CI/CD and now cloud security

Transformation numérique: Plus vite, plus souvent !

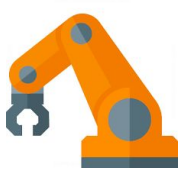
Agile !

De plus en plus rapide avec de moins en moins de gens…

Indice de performance de l’entreprise / être compétitif !

Automatisation

Dans tous les secteurs



DevOps: Une répartition différente des responsabilités

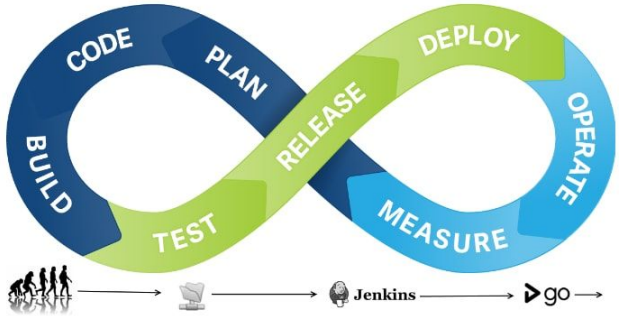
- Une définition qui reste floue et des interprétations différentes de ce qu'est le DevOps.
- Une réorganisation de l'entreprise
- Une seule équipe, plusieurs profils
- Automatisation maximale

Cloud public: La grande promesse !

- “Pay per use”
- Une gamme de produit très large, la sécurité intégrée, des coûts défiants toute concurrence.
- Suppression des intermédiaires, le développeur aux commandes !
- Kubernetes comme orchestrateur principal
- Stockage, base de données
- Security center

Le CI/CD - au coeur de l’automatisation

- Automatisation des tâches (audit, scan…)
- Automatisation de la construction d’images
- Automatisation des déploiements
- Automatisation de la sécurité
- etc.



Kubernetes - Application infrastructure as code

- Automatisation des déploiements
- Tolérance de panne et performances
- Load balancing
- Auto scaling
- etc.

L'équipe DevOps prend le contrôle de son infrastructure

La sécurité, un frein… ou pas...

La sécurité ralenti les itérations

Applications “legacy” transformées ou exposées sans se soucier de la sécurité

La sécurité bloque les mises en production

La sécurité coûte cher

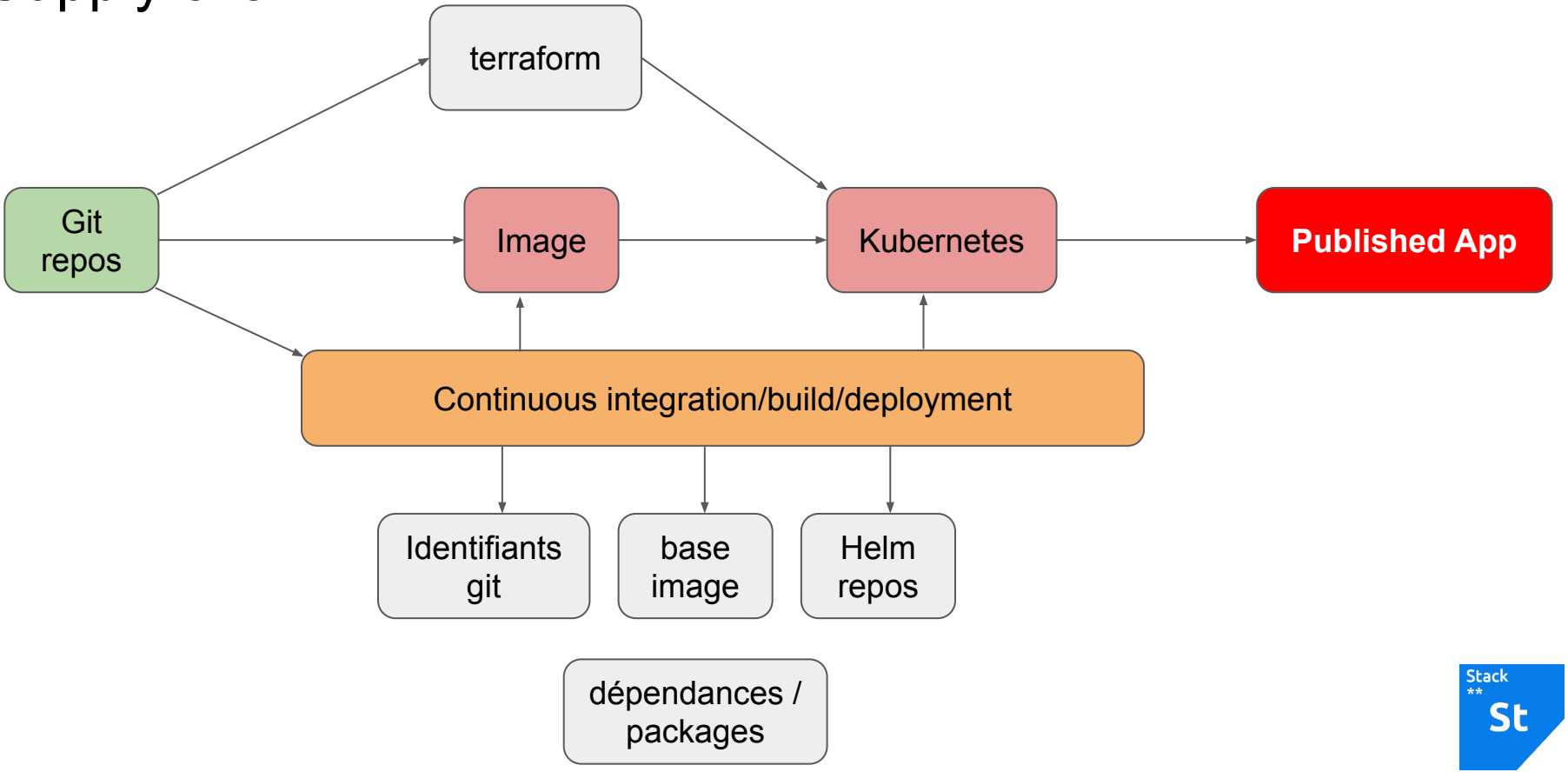
Alors on la supprime (on verra plus tard)

Explosion des menaces et compromissions

La supply chain

- Devient de plus en plus complexe
- Peut être 100% dans le cloud ou hybride
- Manipule des données sensibles
- Utilise des dépendances externes
- Se connecte à votre infrastructure on-prem et cloud
- Etc.

Supply chain



Une cible de choix

Permet de compromettre les applications

Permet de compromettre l’infrastructure

Permet de compromettre les clients

Fait parti de la surface d’attaque

Hors périmètre

“Shift left”

La responsabilité de la sécurité repose maintenant sur l'équipe DevOps.

Des outils inadaptés, trop d’information

En général, 10% du temps alloué à la sécurité

Des outils inadaptés, trop lents, cassent les SLAs du CI/CD

La supply chain dans le cloud public

- Des objets/assets déployés partout
- Des droits d'accès trop complexe
- Des versions non mises à jour
- Des dépendances imposées
- Difficile de savoir ce qui tourne réellement dans son infrastructure

Danger: Le code comme référentiel

Le code représentant l'infrastructure ne donne pas l'assurance que ce qui est déployé est exactement ce qui a été décrit.

Les APIs des Cloud providers évoluent ainsi que les options par défaut.

Les composant peuvent avoir des fonctionnalités par défaut (ex. services admin).

Un développeur peut avoir un accès étendu aux ressources cloud et deployer manuellement (ex. kubeconfig)

Les dépôts de code

As code = tout part du dépôt de code (webhooks).

Contient des applications, des définitions d'infrastructure, des secrets chiffrés etc.

Injection de code

- backdoor
- Ajout de définitions d'objet cloud
- Ajout de dépendances
- etc.

Le dépôt et les événements sur ce dernier vont déclencher toute une série d'actions jusqu'à la mise en production (Webhooks)!

Le CI/CD maillon faible ?

Exécution de binaires

Utilisation de crédentiels

Création d’images référence pour les autres build

Peut-être situé hors de l’infrastructure et déployer à distance: Jenkins

Peut-être situé dans l’infrastructure et builder/déployer du code: Jenkins-X

Analyse la sécurité à un instant T, et non sur la durée

N’est pas au courant de ce qui a pu être déployé manuellement

Un contexte d'exécution non sécurisé

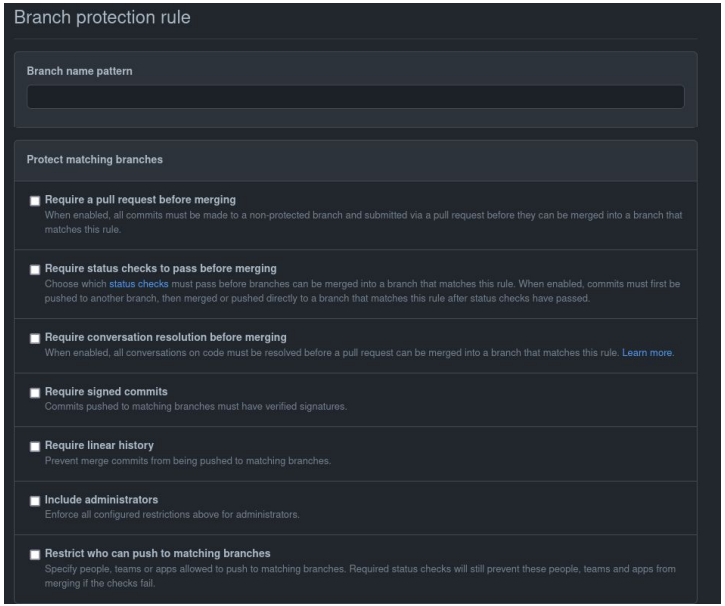
Un cluster dans le cloud public n’est pas sécurisé, il utilise des “nodes” basées sur des machines virtuelles, qui tournent sur des serveurs partagés entre les différents clients. Les ressources qui sont instanciées dans un cluster fonctionnent donc dans un environnement hostile.

Recommandations et hygiène dans le cloud

- Inventorier les différents objets
- Utiliser des projets / tags / labels par équipes pour mieux identifier les objets.
- Analyser la sécurité de chaque objet, ainsi que les dépendances et interconnections

Git repository

1. Signer les commits
2. Mettre en place un cycle d'approbation pour les merges/commit avec au moins deux validations.
3. Protéger les branches
4. Analyser les dépendances au même titre que son propre code
5. Limiter les dépendances externes
6. Ne pas se fier aux étoiles d'un projet Github pour la qualité de son code
7. Vérifier à chaque commit qu'un secret n'est pas hardcodé



Continuous (Build/integration/deployment)

- Privilégier une exécution dans un environnement maîtrisé
  - Dans le cluster k8s.
  - Dans un VPC accessible uniquement via rebond.
- Limiter le scope d’utilisation des secrets (ex: namespace k8s)
- Utiliser un KMS + Vault pour le stockage des secrets
- Définir des SLAs et limiter le temps d'exécution des pipelines
- Limiter l'accès aux ressources externes
  - Proxy registry
  - Proxy dépôt de dépendances
  - etc.

Images docker

- Utiliser des images de base à jour !
- Spécifier un utilisateur
- Supprimer tous les fichiers inutiles
- Une image ne doit contenir que le binaire à exécuter
- Vérifier qu’aucun secret utilisé lors de la construction de l’image ne finisse dans cette dernière (ex. variable d’environnement).
- Signer les images pour pouvoir valider leur déploiement plus tard.

Kubernetes: Contexte d'exécution

- Segmenter les déploiements grâce aux
  - Namespaces
  - Comptes de services (accès aux APIs et ressources)
  - RBAC
- Isoler les namespaces au niveau réseau (Network policies)
- Limiter l'exécution des conteneurs à leur stricts besoins (Pod security policies et Security context)
- Ne pas autoriser les containers à s'exécuter en ROOT !
- Utiliser de tags “immutables” pour deployer les images.
- Utiliser des connections TLS entre chaque POD (mTLS conseille)
- Définir des limites CPU et mémoire (max et request)

Kubernetes: Runtime

- Analyser le comportement des Pods
  - Sondes eBPF sur les nodes
  - Communications entrantes et sortantes
  - Analyse des exécutions de processus dans un container
  - Accès aux ressources du host
  - etc.

Stockage et bases de données

- Ne pas utiliser de mots de passe par défaut sur les bases de données
- Contrôler l'accès aux stockages de données
- Utiliser les comptes de services pour s’authentifier sur les objets de stockages du cloud provider
- Contrôler la sécurité du stockage des sauvegardes

DevOps: Les guildes

Au soutien des équipes pour les développeurs, les admins et la sécurité

Propose des templates et standardisation des processus

Transmet le savoir et l'expérience vers les équipes devops

Les anciennes équipes transverses se concentrent maintenant sur des solutions standard à destination des équipes DevOps.

“Staff Engineer”

Questions ?

mestrade@stacksciences.com