Le guide DevSecOps de l'Infrastructure as Code

Le DevSecOps a permis aux équipes d'automatiser la sécurité et de l'intégrer au cycle DevOps. Dans ce guide, nous examinerons les obstacles à une sécurisation du cloud à l'aide de ces pratiques, ainsi que le rôle de l'Infrastructure as Code dans ce domaine.

Sommaire

L'Infrastructure as Code : introduction 3

Avantages de l'Infrastructure as Code 3

Risques liés à l'Infrastructure as Code 4

Problèmes de sécurité de l'IaC open-source 5

Problématiques DevSecOps avec l'IaC 6

Adopter l'IaC et le DevSecOps dans le cloud : nos conseils 7

Sécurité du cloud dans le cycle DevOps 8

Conclusion 10

L'Infrastructure as Code : introduction

L'Infrastructure as Code (IaC) désigne les technologies et processus utilisés pour gérer et provisionner l'infrastructure à l'aide du code. Elle rend possible des processus DevOps comme le contrôle de versions, la vérification par les pairs, les tests automatisés, l'étiquetage, l'intégration et le déploiement continus.

Essor de l'infrastructure IaC

C'est Puppet qui a donné le jour à l'Infrastructure as Code en 2009 comme alternative aux méthodes traditionnelles de déploiement et de gestion de l'infrastructure. Pour l'entreprise DevOps :

> « Autrefois, les anciennes méthodes de gestion de l'infrastructure – basées sur une documentation et des processus manuels, des scripts fragiles à usage unique et des outils à interface graphique – s'avéraient utiles. Aujourd'hui, face au besoin d'infrastructures ultra-évolutives, à l'adoption d'infrastructures éphémères et à la complexité croissante des systèmes applicatifs, cette gestion doit passer par d'autres moyens. »

Depuis, l'infrastructure IaC a servi de base à plusieurs entreprises comme Ansible, Chef et Salt. Plus récemment, c'est Terraform, un framework IaC open-source signé HashiCorp, surtout utilisé pour définir des ressources dans des services de cloud public, qui a contribué à populariser ce concept. De fait, Terraform a rendu l'Infrastructure as Code accessible et personnalisable à l'infini, au point d'ouvrir la voie à un écosystème IaC.

Dans le même temps, les fournisseurs cloud ont créé leurs propres frameworks de configuration afin de simplifier et d'automatiser l'orchestration et la gestion de l'infrastructure. AWS CloudFormation, Azure Resource Manager (ARM) et Google Cloud Deployment Manager permettent tous aux ingénieurs d'infrastructures de créer des environnements reproductibles plus facilement.

Principe de fonctionnement

L'infrastructure IaC peut être déclarative (définir ce qui sera provisionné) ou impérative (définir la façon de provisionner ces éléments). Terraform et CloudFormation sont deux exemples de frameworks déclaratifs, tandis que AWS Cloud Development Kit (CDK) fait partie des frameworks IaC impératifs. Kubernetes est lui aussi proche d'une infrastructure IaC puisqu'il est possible de le configurer en passant par le code.

Si chaque framework repose sur une syntaxe et des conventions qui lui sont propres, une Infrastructure as Code se compose généralement de déclarations de ressources, de variables d'entrée, de valeurs de sortie, de paramètres de configuration et d'autres encore. Basée le plus souvent sur JSON, HCL ou YAML, l'IaC contient tous les éléments de configuration nécessaires pour déployer votre infrastructure – serveurs, réseau, stockage, sécurité, gestion des identités et des accès (IAM), etc.

```
module "s3_bucket" {
  source = "terraform-aws-modules/s3-bucket/aws"

  bucket = "my-s3-bucket"
  acl    = "private"

  versioning = {
    enabled = true
  }

}
```

Voici un exemple de module Terraform qui crée un compartiment S3 privé avec le versionnage activé.

Avantages de l'Infrastructure as Code

Puisque l'infrastructure IaC utilise du code pour définir les conditions nécessaires au provisionnement des ressources, elle permet une automatisation et des montées en charge homogènes du cloud.

Automatisation

Aujourd'hui, les entreprises déploient d'innombrables applications au quotidien, ce qui modifie constamment leurs besoins d'infrastructure.

C'est pourquoi l'IaC intègre toutes les configurations manuelles au code afin de simplifier le provisionnement du cloud. En transformant les configurations d'infrastructure manuelles en modèles lisibles par les machines, elle épargne aux développeurs le provisionnement et la gestion manuels de ces infrastructures. Quant aux ingénieurs, ils peuvent compter sur des workflows automatisés pour développer, tester et déployer de nouvelles infrastructures.

Évolutivité

L'infrastructure IaC a le mérite de faciliter la tâche des équipes qui doivent configurer des ressources cloud à grande échelle et de réduire le risque d'erreurs sans pour autant gaspiller du temps et des moyens. L'automatisation et la configuration dans le code homogénéisent et simplifient grandement le déploiement de services cloud.

Tout cela facilite également le déprovisionnement d'infrastructures inutilisées, avec à la clé une réduction des coûts de maintenance et d'informatique globaux.

Reproductibilité

En matière d'infrastructure cloud, l'homogénéité s'avère essentielle. L'Infrastructure as Code permet justement de déployer des services réseau, de stockage et de serveur de façon homogène à travers différentes ressources et même différents environnements multicloud. Ainsi, elle réduit les erreurs humaines au maximum tout en ouvrant la voie à une journalisation et un contrôle de versions complets. Grâce à la reproductibilité, il est possible de provisionner davantage de ressources plus facilement, mais aussi de respecter des normes strictes de qualité, les bonnes pratiques de sécurité et d'éventuelles obligations sectorielles.

Sécurité

L'infrastructure IaC joue un rôle clé dans la collaboration entre les équipes. Lorsqu'ils provisionnent des ressources cloud à travers plusieurs environnements et clouds à l'aide d'un seul et même langage unifié, les développeurs et les opérationnels peuvent rester en phase plus facilement et unir leurs forces pour protéger les applications cloud-native.

En bref

En résumé, l'infrastructure IaC vous fait gagner du temps et économise vos ressources. Explications :

- **L'automatisation accroît l'évolutivité et la rapidité du provisionnement de ressources.**
- **La réduction des erreurs humaines se traduit par des déploiements d'infrastructure homogènes et prévisibles.**
- **L'IaC favorise le collaboratif et la codification des connaissances qui minimisent les risques futurs dans l'entreprise.**

Risques liés à l'Infrastructure as Code

En dépit de sa flexibilité et de ses avantages, l'infrastructure IaC présente un certain nombre de points faibles à connaître – surtout en matière de sécurité et de conformité.

Manque d'adoption

Comme toute nouvelle bibliothèque open-source ou plateforme SaaS, l'IaC a le pouvoir de booster votre efficacité. Toutefois, elle nécessite une adhésion et une sensibilisation suffisantes. Parce qu'il s'agit d'une technologie relativement nouvelle, les équipes qui l'adoptent ont du mal à intégrer correctement les nouveaux frameworks à l'infrastructure existante.

Ainsi, l'intégration de l'Infrastructure as Code à votre stack pourra accroître la complexité et la confusion autour du provisionnement, de la gouvernance et de la sécurisation de vos ressources.

Dérives immuables

Avec l'infrastructure IaC, au lieu de gérer votre infrastructure à partir d'une seule console centralisée, vous possédez un autre framework exécuté en parallèle, censé être votre référentiel. Lorsque vous modifiez l'infrastructure en dehors du code qui la provisionne, vous risquez de créer des écarts entre les ressources en cours d'exécution et leur configuration IaC sous-jacente.

Sans les bons outils, ces dérives pourront également entraîner des erreurs de configuration et un risque pour votre environnement.

Sécurité négligée

Les erreurs de configuration représentent la principale cause de compromission de données dans le cloud. Or, d'après certains, 99 % de ces erreurs passent inaperçues[1].

Si les outils de sécurité du cloud offrent la visibilité et les moyens de surveillance nécessaires pour réduire ce chiffre, les éclairages fournis sont parfois en décalage avec l'infrastructure IaC.

Pour les équipes qui utilisent l'Infrastructure as Code, la découverte d'erreurs de configuration post-déploiement et post-requête merge pourra compliquer les relations entre elles et se traduire par une charge de travail supplémentaire. De fait, il faudra bien investiguer, prioriser, planifier et effectuer la correction de ces erreurs.

1. "What is Cloud Security?" McAfee, consulté le 9 novembre 2021, https://www.mcafee.com/enterprise/en-us/security-awareness/cloud.html.

Le meilleur moyen de prévenir ces erreurs reste encore d'adopter une approche « shift-left » de la sécurité du cloud, en recherchant d'éventuelles erreurs de configuration dans le code au lieu de surveiller l'infrastructure en cours d'exécution. Or, comme nous le verrons dans la partie suivante, la résolution des problèmes de sécurité au niveau de l'IaC accuse un certain retard.

En bref

En résumé, l'infrastructure IaC peut créer des risques et des crispations. En cause :

- Des problèmes d'adoption et d'intégration de l'Infrastructure as Code aux environnements, aux équipes et aux workflows
- Une communication inefficace sur les modalités de gouvernance de l'infrastructure et d'application des politiques
- Des outils de sécurité du cloud à la traîne, incapables de fournir un feedback proactif et actionnable

Problèmes de sécurité de l'IaC open-source

Bien que l'Infrastructure as Code donne vie à des modules et modèles open-source préconfigurés, il est important de comprendre que ces composants font rarement la part belle à la sécurité.

L'économie IaC open-source gagne du terrain sur des plateformes comme GitHub et dans des référentiels spécialisés comme Terraform Registry et Artifact Hub. Si l'IaC open-source simplifie et accélère le déploiement de services cloud pour les développeurs, elle relègue souvent la sécurité au second plan.

Ainsi, près de la moitié des modules Terraform open-source dans Terraform Registry et des graphiques Helm dans Artifact Hub contiennent des erreurs de configuration. Les études qui ont servi de source à la figure ci-dessous soulignent les lacunes en matière de sécurité du cloud. Elles montrent également que la sécurité accuse un certain retard et explorent l'impact potentiellement considérable du DevSecOps sur la sécurisation de l'infrastructure cloud.

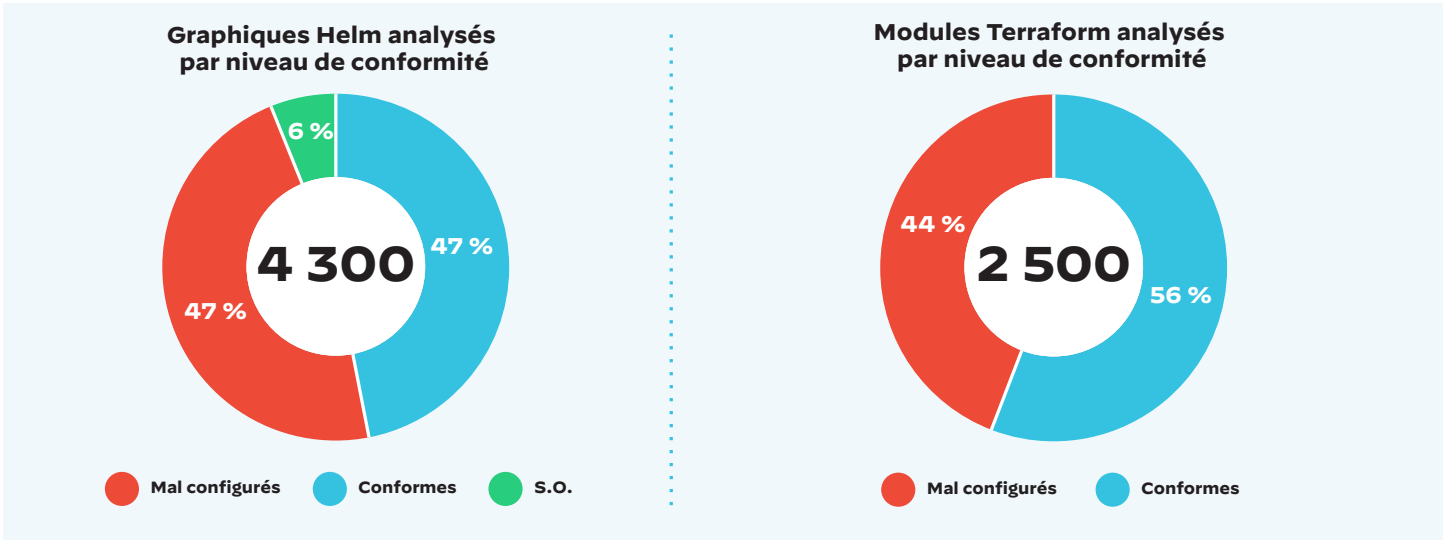

Figure 1 : Conclusions générales de l'étude Open Source Helm Security de Bridgecrew (gauche) ; conclusions générales du rapport State of Open Source Terraform Security de Bridgecrew (droite)

Problématiques DevSecOps avec l'IaC

Sans la bonne approche, la bonne stratégie et les bons outils, l'adoption du DevSecOps pour sécuriser le cloud risque de créer davantage de goulets d'étranglement et de crispations entre les équipes.

Comme chacun sait, les motivations des développeurs sont souvent en décalage avec celles des équipes de sécurité. Les équipes DevOps souhaitent progresser rapidement de manière itérative, tandis que le feedback réactif et hors sprint des équipes de sécurité vient freiner cette progression.

Le DevSecOps vise à intégrer la sécurité à ces équipes DevOps afin d'éviter les erreurs de configuration et les implémentations fragiles qui pourraient exposer les applications cloud-native aux menaces. Or sans la bonne approche, l'adoption d'une stratégie DevSecOps s'avère problématique, surtout pour la sécurisation du cloud.

Il existe deux grandes raisons à cela. D'une part, les entreprises ne disposent pas toujours en interne de l'expérience et de l'expertise nécessaires pour suivre l'évolution rapide des technologies d'infrastructure. D'autre part, puisque de nombreux processus et outils existants ne tiennent pas compte des impératifs de sécurité du cloud, ils pourront engendrer des goulets d'étranglement pour les équipes de sécurité et d'ingénierie cloud-native.

Gouvernance incohérente

Le manque de processus de développement d'infrastructures évolutives complique l'implémentation des pratiques DevSecOps dans le cloud. En répondant aux problématiques de performance et de coût liées au déploiement d'infrastructures à grande échelle, l'infrastructure IaC offre une solution.

La gouvernance des politiques est une question centrale puisque l'IaC crée des permutations supplémentaires d'infrastructure cloud gérées par différentes équipes dans le cadre de différents workflows. Ces workflows accroissent la complexité et l'ambiguïté autour de l'application des politiques. Dans le meilleur des cas, cette ambiguïté engendre une redondance. Mais il arrive aussi qu'elle entraîne des risques.

En pratique :

- Les équipes d'ingénierie supposent que les politiques de sécurité et de conformité sont appliquées au niveau du cloud, à l'aide d'outils des fournisseurs cloud ou d'autres solutions de sécurité.
- Les équipes de sécurité n'ont pas connaissance de tous les frameworks de provisionnement et ignorent si les ressources qu'elles déploient s'accompagnent de politiques adaptées.

Il est donc essentiel de disposer d'un workflow pour combler ces écarts. Sans quoi les ressources cloud déployées avec du code mal configuré viendront alourdir la charge de travail des ingénieurs.

Conseil : éliminez les problèmes à la source

D'après nos données, lorsqu'une erreur de configuration n'est corrigée qu'en cours d'exécution de l'infrastructure IaC, elle a 70 % de chances de réapparaître. En éliminant les problèmes à la source, vous empêchez les erreurs de configuration de refaire surface.

Écarts de compétences et d'accès

Les outils automatisés détectent tout, des problèmes de sécurité critiques aux cas de non-respect des bonnes pratiques informatiques. Toutefois, même les meilleurs outils dernier cri ne peuvent remplacer des workflows et processus DevSecOps fiables dans le cloud.

C'est parce que les erreurs de configuration identifiées par un outil doivent encore être contextualisées, priorisées et corrigées. Or, il n'est pas toujours aisé de réunir les équipes de sécurité et d'ingénierie autour de ces missions. Les effectifs chargés de la sécurité et de la conformité n'ont généralement pas accès aux référentiels de code et aux consoles cloud pour procéder à des corrections. Et même lorsqu'ils y ont accès, ils ne disposent pas toujours du savoir-faire et du contexte applicatif complet nécessaires pour éliminer eux-mêmes les erreurs de configuration. De leur côté, les ingénieurs ne comprennent pas suffisamment les enjeux de sécurité pour prioriser correctement les erreurs. Et même s'ils maîtrisaient le sujet, ils auraient bien du mal à trouver le temps de les corriger après avoir changé de contexte.

En somme, il arrive que l'automatisation resserre les goulets d'étranglement qu'elle était censée éliminer. Par exemple :

- La sécurité ouvre un ticket et l'attribue à un chef d'équipe qui pourra travailler ou non dans l'équipe concernée par le problème.
- Dans Jira, la correction du problème finit par être intégrée à un sprint numéroté.
- Pendant ce temps, dix autres tickets « P0 » ont été créés.
- Avec l'analyse automatique, ces tickets sont facilement générés par centaines. Tout le processus se répète ad nauseam.

S'ajoute à cela la fameuse pénurie de compétences en cybersécurité ! Résultat : les tâches de sécurité et de conformité sont assignées à des développeurs souvent mal préparés.

Quel que soit l'état de vos ressources de sécurité, vous ne pouvez pas attendre des développeurs qu'ils maîtrisent tous les problèmes de configuration possibles dans le cadre de la création d'une infrastructure cloud. Lorsque vous vous reposez trop sur eux, le déploiement d'erreurs de configuration est quasi assuré. Or, non seulement ces erreurs non corrigées comportent des risques, mais elles deviennent aussi bien plus coûteuses à éliminer à mesure qu'elles progressent à travers le cycle de développement.

En bref

En résumé, le DevSecOps peut aggraver la situation dans les cas suivants :

- Gouvernance des politiques peu claire et incohérente entre le code d'infrastructure et les ressources cloud déployées
- Écarts de compétences ou d'accès sur le plan de l'identification et de la correction des erreurs de configuration qui exposent le cloud aux dangers

Adopter l'IaC et le DevSecOps dans le cloud : nos conseils

Après avoir passé en revue les éventuelles problématiques liées au DevSecOps dans le cloud, voici maintenant trois mesures que vous pouvez prendre pour les résoudre :

1. **Tout automatiser**

   Vos équipes d'ingénierie automatisent sans doute déjà une grande partie de vos tests logiciels, des tests unitaires aux recherches de dépendances. La sécurité du cloud ne devrait pas faire exception. En effet, même les professionnels de la sécurité les plus chevronnés ne peuvent connaître toutes les bonnes pratiques de sécurité les plus récentes de tous les fournisseurs cloud, couches de service, plateformes d'orchestration, ressources, etc., et encore moins les appliquer à leur propre architecture.

   L'analyse automatisée représente le seul moyen de programmer l'application de l'intégralité des règles de sécurité et de conformité sans consacrer trop de temps et de ressources à une lecture minutieuse de la documentation. Sans feedback actionnable, l'automatisation pourra engendrer davantage de soucis et de travail pour les ingénieurs. C'est pourquoi il est essentiel de trouver les bons outils – qu'il s'agisse de solutions commerciales ou open-source comme Checkov – pour automatiser la détection et la correction des problèmes.

2. **Capitaliser sur les processus existants**

   Même les éclairages les plus utiles peuvent s'avérer dérangeants et stériles lorsqu'ils sont présentés au mauvais endroit, au mauvais moment. Le seul moyen de favoriser l'adoption du DevSecOps dans le cloud est donc de l'intégrer aux outils et processus dont les développeurs dépendent au quotidien. Heureusement, la clé de voûte de l'adoption de l'automatisation et du Security-as-Code est probablement déjà en place.

   Pour savoir où et comment appliquer les garde-fous, vous devrez tenir compte de plusieurs facteurs : suite d'outils existants, objectifs de l'organisation, maturité de la sécurité, etc.

3. **Intégrer la sécurité au code**

   La plupart des points de friction DevSecOps s'expliquent par le caractère traditionnellement réactif des approches de sécurité, qui reposent sur la recherche des problèmes plutôt que sur leur prévention. L'infrastructure IaC permet de remplacer ces méthodes par une démarche proactive. Que l'on parle de Policy-as-Code ou de Security-as-Code, la transposition de l'automatisation de la gouvernance dans un seul et même langage fournit tant aux professionnels de la sécurité qu'aux ingénieurs DevOps des éclairages rapides et des solutions exploitables.

   L'intégration des politiques à la couche de code garantit une application homogène des règles de sécurité du cloud tout en permettant de les étendre à tout l'environnement au fil du temps. À long terme, vous économiserez également des ressources puisque la correction d'une faille logicielle coûte 100 fois plus cher en production qu'au moment de l'écriture du code[2]. Toutefois, il est vrai que, plus vous êtes en amont du cycle de développement, moins vous disposez d'informations sur l'infrastructure et ses éventuelles erreurs de configuration.

   C'est pourquoi il est impératif de traiter les questions de sécurité de l'infrastructure à chaque phase du cycle DevOps.

2. Mukesh Soni, *Defect Prevention: Reducing Cost and Enhancing Quality*, https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/.

Sécurité du cloud dans le cycle DevOps

Si chaque entreprise a besoin d'une solution qui lui est propre, pour favoriser l'adoption du DevSecOps dans le cloud, il est primordial d'appliquer des garde-fous en amont de l'infrastructure IaC et de générer des éclairages actionnables tout au long du cycle DevOps.

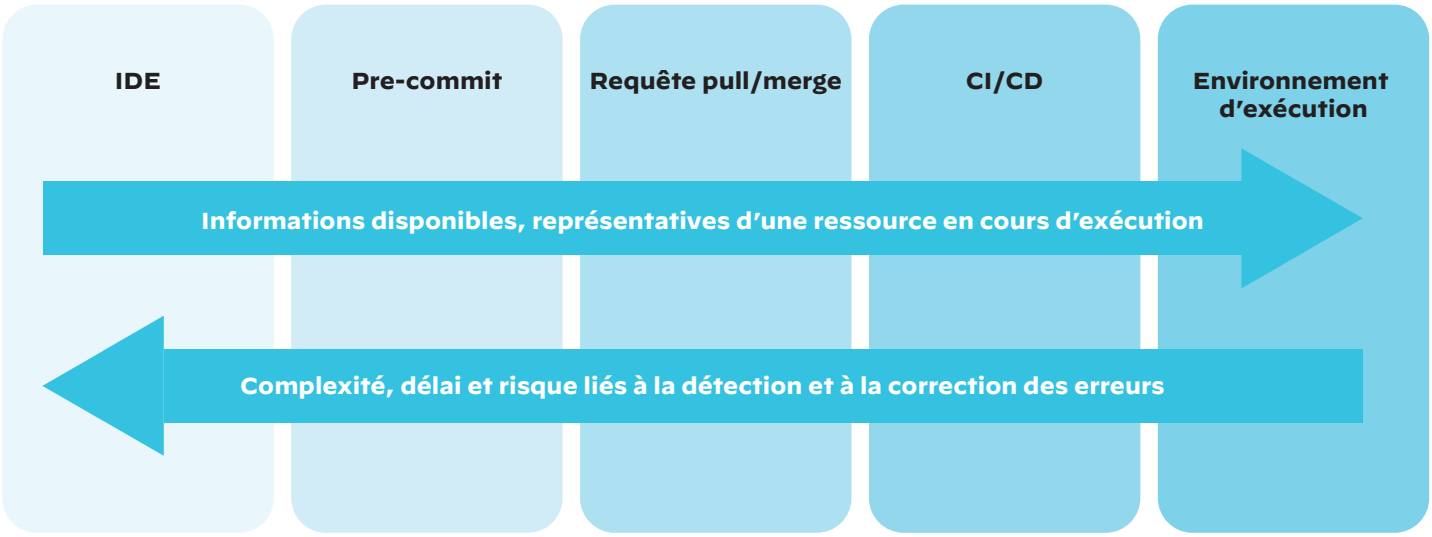

Figure 2 : Avantages et inconvénients de l'identification des erreurs de configuration du cloud à chaque stade du cycle de développement

Analyse IDE

Alerter en amont : voilà le grand principe de la sécurité « shift-left ». Et quels meilleurs emplacement et moment que celui où les développeurs écrivent du code pour générer ces alertes ?

Pour intégrer la sécurité du cloud le plus en amont possible (en dehors des phases de conception et de planification), mieux vaut intégrer des garde-fous à votre environnement de développement intégré (IDE). Ces garde-fous prendront certainement la forme d'un plug-in ou d'une extension comme Checkov VS Code. Puisqu'elle minimise les changements de contexte, l'analyse IDE reste le moyen le plus fiable et le plus économique d'identifier les problèmes.

Hooks pre-commit

Les tests unitaires et d'intégration pre-commit font partie des bonnes pratiques généralement admises. Alors pourquoi pas une analyse de sécurité de l'IaC pre-commit ?

En recherchant d'éventuelles erreurs de configuration dans l'Infrastructure as Code localement, vous pouvez corriger ces erreurs en toute sécurité, dans votre propre espace de travail, avant d'intégrer le code à un référentiel partagé. L'analyse pre-commit vous évite également de perdre du temps sur des builds défaillants (et de retarder les builds de vos collègues), ou encore de stopper net les vérifications de requêtes pull.

Que vous effectuiez des modifications au moment du feedback ou que vous utilisiez les suggestions de l'IDE pour vous guider dans la création d'une infrastructure sécurisée, les inconvénients sont très peu nombreux. Le seul point faible de l'analyse locale, c'est qu'il revient au développeur de la lancer et d'effectuer les changements.

Avec le bon mélange d'analyses passives et de feedback actionnable, vous pouvez renforcer votre sécurité sans déployer des efforts colossaux.

Vérifications de requêtes pull/merge

Pour les équipes qui ne jurent que par leur système de contrôle de versions (VCS), l'intégration de la sécurité aux processus de revue de code approuvé présente de nombreux avantages. En fonction des éléments déclencheurs de vos builds CI/CD, cette approche pourra être associée à l'analyse via le pipeline ou la remplacer.

Votre système VCS pourra également offrir des contrôles uniques pour l'implémentation de garde-fous. Les trois principales plateformes (GitHub, GitLab et Bitbucket) proposent le contrôle de versions, la création de branches, la revue de code intégré et des contrôles des autorisations. Ainsi, les développeurs peuvent tester le code sans compromettre les systèmes en production et continuer à modifier ce code avant toute requête merge.

Conseil : ajoutez des garde-fous

Il est bon de paramétrer votre revue de code pour bloquer les requêtes merge en cas d'échec des vérifications, afin d'éviter d'intégrer une IaC mal configurée à votre branche de version principale.

Tâches CI/CD

Les pipelines CI/CD jouent un rôle essentiel dans la compilation de l'infrastructure et le test du code compilé avant son déploiement. Il est important d'analyser ce niveau d'abstraction pour identifier les erreurs de configuration dans les ressources clés, les variables et les modules dépendants sur le point d'être provisionnés.

L'intégration de la sécurité de l'infrastructure au pipeline CI/CD permet également de l'automatiser et de l'adapter entièrement à votre workflow. Vous pouvez choisir les points à valider impérativement sur vos builds et accéder au feedback directement dans votre solution de pipeline CI/CD. Il s'agit aussi d'un outil collaboratif que toute votre équipe utilise pour examiner, rejeter et approuver les changements.

Que votre analyse s'effectue dans l'outil CI/CD ou dans votre système VCS, ce niveau de contrôle favorise la collaboration et l'accessibilité du point de vue des développeurs.

Sécurité dans l'environnement d'exécution

Attention : la sécurité « developer-first » ne remplace pas pour autant les méthodes « traditionnelles » de recherche d'erreurs de configuration nuisibles à la sécurité et à la conformité dans les ressources cloud en cours d'exécution.

Même avec l'infrastructure IaC, des modifications manuelles peuvent survenir, ce qui pourra entraîner des dérives de configuration involontaires entre le code et les ressources cloud en cours d'exécution. La gestion des dérives cloud peut vous éclairer sur ces ressources afin d'identifier les écarts susceptibles de présenter un risque.

En somme, vous ne pouvez pas vous baser uniquement sur les éclairages générés en cours de développement sans les rapprocher d'états réels en cours d'exécution, au risque d'entraîner de véritables conflits. Puisque l'analyse en cours d'exécution suit l'état réel des configurations, elle offre le seul moyen viable d'évaluer les changements de configuration au fil du temps en cas de méthodes multiples de gestion des configurations. C'est également une excellente façon de satisfaire à vos obligations d'audit et de traçage du contrôle continu des modifications.

En bref

Un programme DevSecOps complet dans le cloud doit comprendre les points suivants :

- **Accès des développeurs à des outils économiques capables de générer des éclairages en amont**
- **Application collective de garde-fous dans votre système VCS ou votre pipeline CI/CD**
- **Visibilité continue sur les ressources en cours d'exécution pour remédier aux dérives liées à des modifications de configuration manuelles**

Conclusion

L'IaC permet de déployer et de gérer les infrastructures plus efficacement. Elle joue également un rôle crucial dans l'implémentation du DevSecOps dans le cloud. Si elle génère des problématiques qui lui sont propres, l'Infrastructure as Code contribue à sécuriser votre infrastructure.

En intégrant les analyses d'IaC et les corrections Security-as-Code à tous les stades du cycle DevOps, vous pouvez adopter une approche plus moderne de la sécurité du cloud.

Comme toute nouvelle technologie, l'infrastructure IaC peut accroître la complexité et les risques – en particulier lorsque diverses équipes utilisent différents frameworks. Puisqu'elle s'exécute aussi en parallèle avec des outils d'orchestration manuelle du cloud, elle nécessite une adoption et une visibilité complètes pour éviter les écarts de provisionnement et, surtout, de sécurité des ressources.

Automatisation, évolutivité, reproductibilité… les avantages de l'IaC pour la gestion et le provisionnement du cloud en compensent généralement les coûts.

Pour l'équipe Prisma Cloud, l'Infrastructure as Code a également le pouvoir d'impulser un rapprochement entre les équipes DevOps, de sécurité et d'ingénierie de l'infrastructure. Grâce aux avantages de l'IaC, les équipes peuvent en effet automatiser la sécurité du cloud et l'intégrer au cycle DevOps. Notre plateforme de sécurité cloud codifiée intègre les analyses d'IaC et les corrections Security-as-Code aux outils et workflows des développeurs pour permettre cette collaboration.

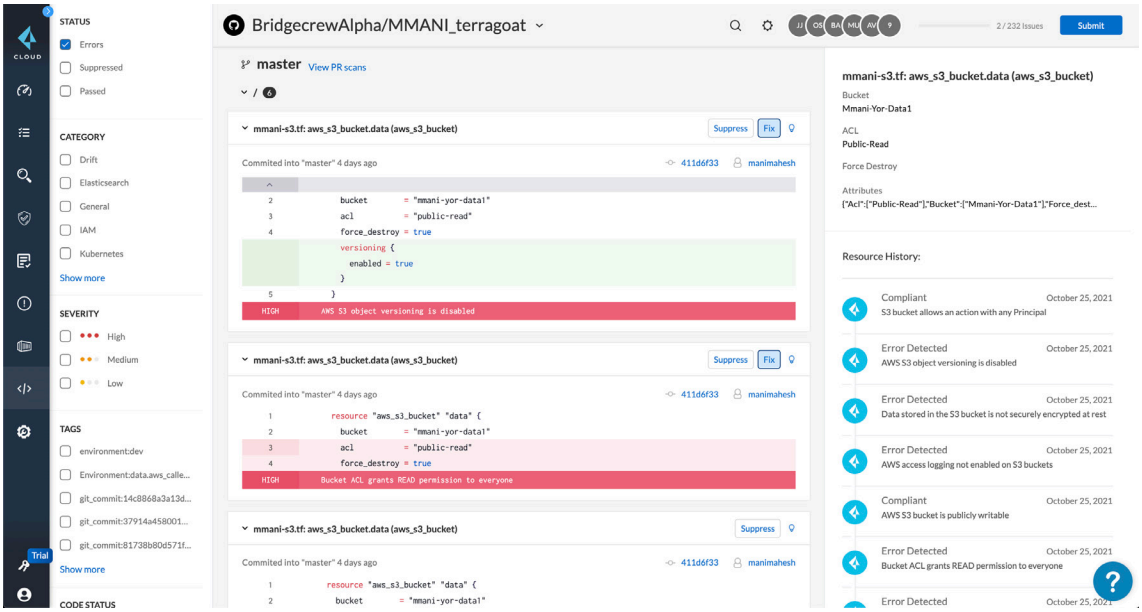

Figure 3 : Fonctionnalités de sécurité de l'IaC dans Prisma Cloud

Choisissez Prisma Cloud pour :

- Identifier et corriger les erreurs de configuration dans les ressources cloud et l'IaC
- Appliquer des centaines de politiques intégrées de sécurité et de conformité
- Intégrer des garde-fous via les plug-ins IDE, les hooks pre-commit et les intégrations natives VCS et CI/CD

Essai gratuit

Oval Tower, De Entrée 99 – 197
1101HE Amsterdam,
Pays-Bas

Téléphone : +31 20 888 1883
www.paloaltonetworks.fr