sysdig

Checklist pour sécuriser Kubernetes

Kubernetes est de facto devenu le système d'exploitation du cloud. Il permet aux développeurs de regrouper facilement leurs applications sous forme de microservices portables. Toutefois, Kubernetes peut être difficile à utiliser. Les équipes DevOps délaissent souvent la sécurité jusqu'à ce qu'elles soient prêtes à déployer le code en production.

Kubernetes nécessite une nouvelle approche de la sécurité. Les processus et outils traditionnels n'arrivent pas à satisfaire aux exigences du cloud natif car ils ne fournissent pas la visibilité sur les environnements de container dynamiques.

54 % des containers ont une durée de vie inférieure ou égale à cinq minutes[1], ce qui rend très complexe l'investigation des comportements anormaux et violations.

L'un des points clés de la sécurité du cloud natif est de réagir le plus rapidement possible aux risques pesant sur la sécurité des containers. Le faire plus tard dans le cycle de vie du développement ralentit le rythme de l'adoption du cloud tout en augmentant les risques de sécurité et de conformité.

Les équipes **Cloud/DevOps/DevSecOps** sont généralement responsables de la sécurité et de la conformité ainsi que des applications cloud critiques passées en production. Il s'agit d'une tâche de plus dans leur calendrier déjà serré car elles doivent en outre assurer la bonne santé des applications et de l'infrastructure cloud.

Nous avons conçu cette checklist pour vous aider à choisir votre approche de la sécurité lorsque que vous vous lancez dans l'utilisation des containers et de Kubernetes.

[1] https://www.zdnet.com/article/technology-containers-short-lifespans-are-getting-even-shorter/

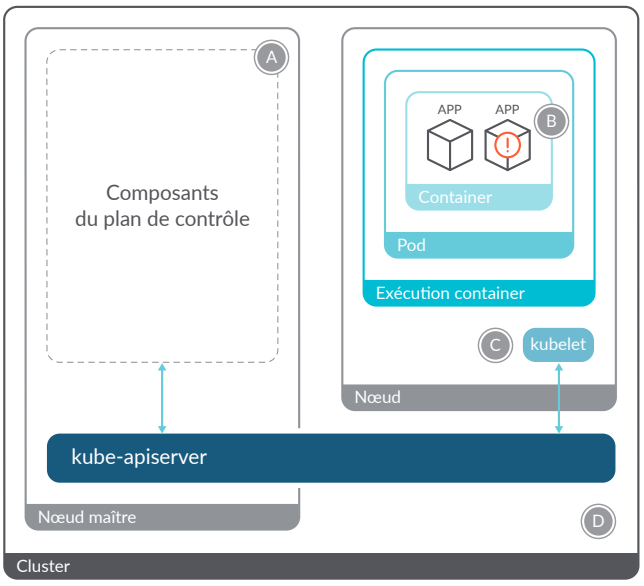
Décomposer les risques de sécurité Kubernetes

La possibilité de provisionner de manière déclarative et de configurer les paramètres d'infrastructure Kubernetes ainsi que les contraintes d'application via une approche « Infrastructure as a code » (IaC) permet aux entreprises de définir une baseline de sécurité pour tous les aspects clés de leurs clusters Kubernetes, quelle que soit la version et l'infrastructure sous-jacente. La configuration déclarative élimine également les erreurs des opérateurs pouvant conduire à de mauvaises configurations par essence inexploitables.

Observons maintenant un cluster Kubernetes pour comprendre quels éléments vous devez protéger.

Premièrement, vous devez protéger vos applications et bibliothèques. Les vulnérabilités dans vos images OS de base pour vos applications peuvent être exploitées pour voler des données, faire crasher vos serveurs ou obtenir frauduleusement des privilèges. Les bibliothèques tierces font également partie des éléments à sécuriser. Bien souvent, les hackers ne se compliqueront pas la vie à chercher des vulnérabilités dans votre code car il est plus simple d'exploiter les failles connues de vos bibliothèques d'applications.

Le domaine suivant est le control plane Kubernetes, le cerveau de votre cluster. Des programmes tels que le controller manager, etcd ou kubelet sont



A Accès via le proxy d'API Kubernetes API etcd

B Exploiter les vulnérabilités dans les apps ou bibliothèques tierces

C Accès via l'API

D Accès aux serveurs ou aux machines virtuelles

accessibles via l'API Kubernetes. Un hacker ayant accès à votre API pourrait mettre complètement à l'arrêt votre serveur, déployer des containers malveillants ou encore supprimer votre cluster en entier. En outre, comme votre cluster fonctionne sur des serveurs, l'accès à ces derniers doit être protégé. Un accès non désiré à ces serveurs ou aux machines virtuelles sur lesquelles les nœuds sont exécutés permettra à un hacker d'accéder à toutes vos ressources et de créer de graves failles de sécurité.

Maintenant que nous connaissons les éléments à sécuriser, penchons-nous sur les détails et examinons le cadre d'approche de la sécurité Kubernetes.

1

CHECKLIST POUR SÉCURISER KUBERNETES

Sécuriser l'infrastructure as code

De quoi s'agit-il ? **Infrastructure as Code (IaC)** est un moyen de gérer l'infrastructure Kubernetes et les applications au sein d'un système de contrôle des versions (p. ex. Git). Toutes les modifications de l'infrastructure sont effectuées via des requêtes pull qui changent les fichiers source.

Une fois approuvées et fusionnées, les requêtes pull reconfigureront et synchroniseront l'infrastructure de production pour correspondre au statut défini dans la repository source.

Cette tendance est une opportunité pour s'occuper de la sécurité davantage en amont dans le cadre d'un workflow DevOps sécurisé pour gérer les risques dans Kubernetes.

Avantages : L'approche IaC gagne rapidement en popularité car elle montre la voie vers plus de résilience par le biais d'un meilleur contrôle opérationnel. Avec la sécurité IaC, les équipes peuvent identifier et éliminer les risques de configuration avant le déploiement de l'infrastructure en production.

Les commandes commit de Git fournissent des mises à jour vérifiables et vous permettent d'appliquer également l'auto-remédiation via un workflow GitOps. Les équipes peuvent améliorer leur posture de sécurité Kubernetes et éliminer l'écart entre la source et la production.

Approche : Les équipes de DevOps peuvent automatiser la conformité et la gouvernance en utilisant une stratégie sous forme de code basée sur Open Policy Agent (OPA). Elles peuvent appliquer les stratégies pour analyser le prédéploiement des modèles IaC et détecter la dérive d'exécution qui peut être remédiée à la source avec une simple requête pull. Les correctifs peuvent être priorisés sur la base du contexte applicatif.

2

CHECKLIST POUR SÉCURISER KUBERNETES

Prévention des menaces avec les contrôleurs d'admission

De quoi s'agit-il ? **Les contrôleurs d'admission Kubernetes** sont des morceaux de code qui interceptent les appels d'API Kubernetes avant la création des objets. Ils peuvent être vus comme un gardien qui intercepte les requêtes API et durcit ce qui peut être exécuté sur le cluster.

L'analyse des images en CI/CD est une exigence critique de l'implémentation de la sécurité des containers et de Kubernetes. Cependant, les développeurs peuvent parfois contourner le pipeline CI/CD et déployer une image directement dans le cluster. Vous pouvez intégrer un **contrôleur d'admission** avec un moteur d'analyse pour empêcher le déploiement d'images risquées qui ne répondent pas à vos exigences de sécurité et de conformité.

Avantages : Lorsque **l'analyse d'images** est utilisée avec le **contrôleur d'admission**, vous pouvez bloquer les menaces avant qu'elles n'atteignent la production. Vous déclenchez ainsi immédiatement une analyse pour chaque image essayant de se déployer dans le cluster. Vous pouvez également utiliser un contexte d'environnement supplémentaire lors de la définition des critères d'admission comme le namespace, les métadonnées de pod, etc. En déclenchant une analyse d'image via le contrôleur d'admission, vous pouvez :

- Vérifier votre application, ses bibliothèques et d'autres fichiers afin de trouver des vulnérabilités bien connues.
- Analyser les métadonnées pour détecter les erreurs de configuration comme les identifiants exposés, les ports non-sécurisés ou l'utilisation de comptes root.
- Définir les contrôles personnalisés comme la mise sur liste noire de packages ou la détection des permissions erronées.

Si ces stratégies de sécurité ne sont pas satisfaites, vous pouvez empêcher l'image d'atteindre la production et avertir vos développeurs qui règleront les problèmes.

Approche : Activez le **contrôleur d'admission Kubernetes** en l'intégrant avec un moteur d'analyse pour empêcher le déploiement d'images risquées ou non analysées.

De quoi s'agit-il ? **PodSecurityPolicy (PSP)** est une ressource de niveau cluster qui contrôle les actions qu'un pod peut effectuer ou les ressources auxquelles il peut accéder. Cette ressource peut être utilisée pour implémenter l'accès à moindre privilège pour les pods.

Avantages : **PSP** peut prévenir les menaces sans affecter les performances en mettant en place un accès à moindre privilège pour les pods dans vos clusters. Vous pouvez mettre en place à ce niveau des contrôles préventifs comme retirer l'autorisation d'exécution des containers privilégiés, restreindre les ressources ou limiter l'accès aux volumes.

Approche : **PodSecurityPolicy** est implémentée comme un contrôleur d'admission optionnel (mais recommandé).

3

CHECKLIST POUR SÉCURISER KUBERNETES

Sécuriser le plan de contrôle (Control Plane) Kubernetes

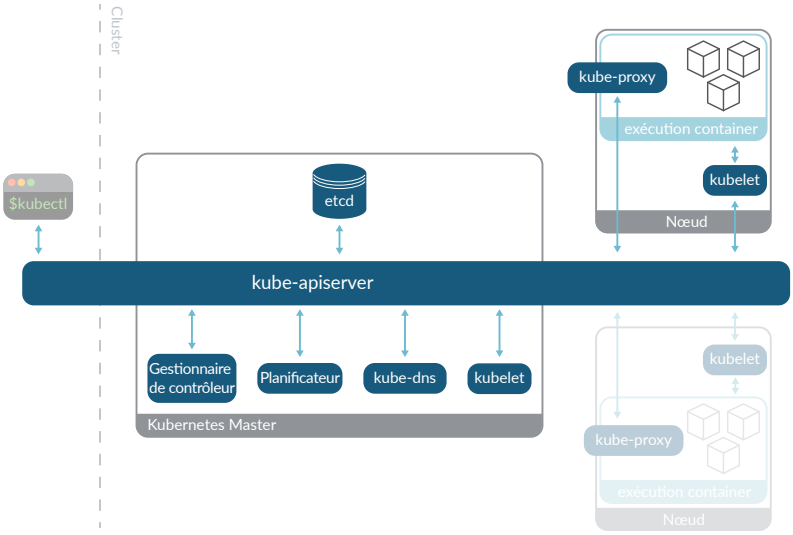
De quoi s'agit-il ? Le plan de contrôle Kubernetes est le cerveau de votre cluster Kubernetes. Il gère toutes vos ressources de cluster, peut planifier de nouveaux pods et peut lire tous les secrets stockés dans le cluster.

Avantages : Le plan de contrôle, comme son nom l'indique, contrôle votre cluster. En le sécurisant, il empêche un utilisateur malveillant d'extraire des informations, de faire crasher votre infrastructure ou de planifier des pods ayant accès au nœud parent.

Approche : Isolez le réseau de cluster, sécurisez l'API et auditez les commandes kubectl.

Les composants du plan de contrôle communiquent via l'API Kubernetes et les instructions kubectl peuvent être traduites en appels d'API. Sécurisation :

- Vérifiez la configuration **kubelet** : Désactivez `l'authentification anonyme`, définissez un `fichier-client-ca`, assurez-vous que le `mode d'autorisation` délègue au serveur API et désactivez le `port en lecture seule`.
- Activez `NodeRestriction` dans votre API afin que les agents kubelet soient les seuls autorisés à effectuer des modifications sur leur propre nœud.
- Activez l'autorisation via RBAC.



4

CHECKLIST POUR SÉCURISER KUBERNETES

Sécuriser les workloads lors de l'exécution

De quoi s'agit-il ? Gérer les risques de sécurité lors de l'exécution dans les containers et les environnements Kubernetes. **Sécurité d'exécution** détecte les comportements anormaux pouvant indiquer la compromission d'un container.

Avantages : Signalez les responsables et réagissez rapidement aux **vulnérabilités** récemment détectées avant qu'elles ne soient exploitées. Détectez et réglez les **attaques** dès qu'elles se produisent et avant qu'elles n'entraînent de sérieux dommages. Protégez le système des **erreurs de configurations ou des bugs** de logiciel causant des fuites de ressources et des comportements erratiques.

Approche : Analysez en continu pour pouvoir détecter les problèmes dès que possible. Définissez également des réponses automatiques aux incidents afin de réagir immédiatement. Enfin, capturez les données d'analyse forensique quand un incident se produit afin que vous puissiez enquêter sur la cause racine et l'empêcher de se reproduire. Penchons-nous plus en détail sur chacune de ces stratégies.

Signalement des vulnérabilités d'exécution : Après l'analyse initiale de l'image, de nouvelles vulnérabilités peuvent être détectées ou vos stratégies peuvent changer. Vous devez continuer à analyser vos images pour garantir leur sécurité au fil du temps. Certains dispositifs d'analyse des images exigent de vous d'effectuer une analyse complète à chaque fois, d'autres sauvegardent les métadonnées et vous alertent de la présence de nouveaux problèmes sans analyse supplémentaire. Vous devez être en mesure de mapper les vulnérabilités critiques pesant sur les applications spécifiques (p. ex. les CVE avec un fix disponible sur les images s'exécutant plus de 30 jours) et d'identifier les équipes devant régler ces problèmes. Cela nécessite de mapper les CVE dans le paysage de ressources Kubernetes (namespaces, déploiements, clusters, pods, etc. spécifiques).

Détection des comportements anormaux : Votre container exécute-t-il les tâches qu'il est censé réaliser ? Accède-t-il à des fichiers auxquels il ne devrait pas avoir accès ? A-t-il d'étranges connexions réseau ? Quelqu'un a-t-il ouvert un terminal ? En surveillant les activités de votre container, vous pouvez détecter des comportements anormaux.

Vous aurez besoin d'une **instrumentation** pour repérer ces problèmes. Votre instrumentation couvre-t-elle seulement vos applications ou également vos appels système ? Plus vous aurez de données, plus le nombre de comportements que vous pourrez détecter sera important. De combien de **ressources** votre instrumentation a-t-elle besoin ? Certaines solutions auront besoin de beaucoup de capacités de mémoire tandis que d'autres surchargeront votre CPU.

Falco est de facto le moteur de détection des menaces Kubernetes ; il détecte les comportements d'application inattendus et alerte sur les menaces lors de l'exécution. Falco capture les appels système en utilisant eBFP (entre autres sources), ce qui fournit de la visibilité sur les activités du système d'exécution avec le contexte d'application Kubernetes et le rend compatible avec les environnements de production hautes performances.

Créer des règles pour tous vos pods peut être une tâche chronophage. Disposer d'une vaste bibliothèque de **règles prêtes à l'emploi** peut faire la différence ici. Avec autant d'images, il est très facile de perdre le fil. Par conséquent, être capable d'utiliser **le machine learning** pour profiler les comportements attendus est un bon filet de sécurité.

Détection des menaces via la sécurité opérationnelle : Les services de journalisation comme AWS CloudTrail peuvent permettre la gouvernance, la conformité, l'audit opérationnel et l'audit des risques pour votre compte cloud. Ainsi, vous pouvez journaliser, surveiller et stocker les activités de compte liées aux actions (changements de configuration, création/suppression/modification d'événements) dans l'ensemble de votre infrastructure cloud. L'ensemble des règles Falco prêtes à l'emploi pour CloudTrail est une source de confiance pour l'audit opérationnel. Cela permet de minimiser les efforts de configuration, le temps de réponse et le volume de ressources demandé pour la visibilité du runtime. Visibilité dont vous avez besoin pour la détection des menaces AWS et l'investigation sur les événements liés à la sécurité.

5

CHECKLIST POUR SÉCURISER KUBERNETES

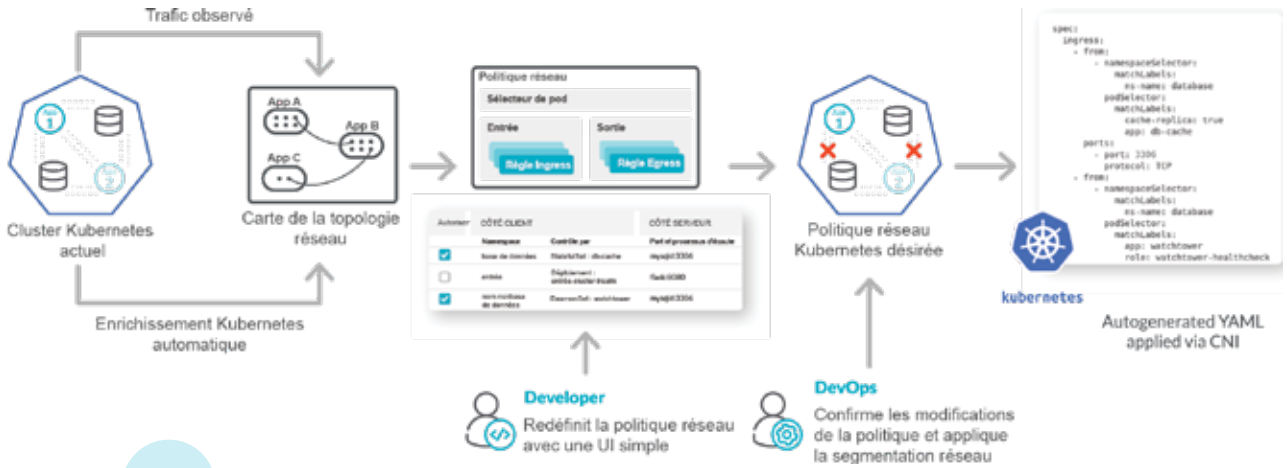
Segmentation réseau Kubernetes native

De quoi s'agit-il ? Par défaut, les pods Kubernetes sont non isolés, ce qui signifie qu'ils acceptent le trafic provenant de n'importe quelle source. Les politiques réseau Kubernetes sont un contrôle de sécurité natif sur la plateforme qui peut être utilisé pour définir la manière dont les applications et services communiquent explicitement entre eux.

Avantages : Protégez votre système des attaques comme le mouvement latéral dans les containers, l'exfiltration de données et l'élévation de privilèges. Vos équipes peuvent également satisfaire aux exigences de conformité (NIST, PCI, etc.) nécessitant une segmentation réseau.

Approche :

- Surveillez toutes les connexions réseau établies entre les applications et les services fonctionnant dans Kubernetes.
- Utilisez le contexte de l'application et des métadonnées Kubernetes. Utilisez les contrôles natifs des politiques réseau Kubernetes pour implémenter les politiques les moins privilégiées.
- Configurez les politiques réseau natives Kubernetes pour segmenter et restreindre le trafic entre, vers et depuis les pods.
- Simplifiez la gestion des politiques réseau et intégrez-le dans votre stratégie « Policy-as-code ».



6

CHECKLIST POUR SÉCURISER KUBERNETES

Réponse aux incidents et analyse forensique

De quoi s'agit-il ? Quand un événement non autorisé se produit, vous avez besoin de l'ensemble des enregistrements d'audit qui décrivent moment du changement, son type et qui l'a effectué. Vous devez également savoir comment les fichiers sensibles ont été modifiés et par qui.

Avantages : Vous pouvez devenir efficace opérationnellement et prêt pour l'audit avec votre environnement de containers. Cela vous aidera à résoudre les problèmes rapidement et à valider les exigences de conformité pour PCI, NIST, SOC2 etc.

Réponse automatique aux incidents : Réagissez immédiatement aux incidents avant qu'ils ne posent un problème plus important. Rien n'est plus rapide qu'une réponse automatique. Les incidents critiques exigeront que vous mettiez à l'arrêt les pods concernés mais pour les autres incidents, une notification suffit. Être capable de notifier les personnes pertinentes pour mener des enquêtes plus approfondies via les bons canaux est crucial.

Outils d'audit et d'analyse forensique : Vous devez capturer autant d'informations que possible sur un incident car le temps que vous enquêtiez, les containers pourraient déjà avoir disparu. Outre la capture, vous aurez besoin d'une manière de parcourir les données afin de pouvoir corréler les événements et trouver la source du problème plus rapidement. Par exemple, vous devriez être capable d'identifier les activités réseau inhabituelles, les corréler aux commandes shell exécutées au même moment et voir quels fichiers ont changé.

Réponse automatisée

- Signaler les incidents
- Mettre en pause / Tuer les containers

Données d'analyse forensic

- Qu'est-ce qui a changé ?
- Quand ?
- Par qui ?

Audit Tap : Suivez toutes les connexions réseau vers et depuis un processus spécifique même si la connexion est réussie. Faites un enregistrement de toutes les connexions acceptées/ayant échoué pour identifier les processus suspects ou inhabituels.

Surveillance de l'intégrité des fichiers : Vous donne de la visibilité sur toutes les activités liées à des fichiers sensibles. Elle est utilisée pour détecter la falsification de fichiers système critiques et de répertoires ainsi que les modifications non autorisées, que l'activité soit une attaque malveillante ou une opération non prévue.

- Intégrez les contrôles FIM dans votre stratégie d'analyse d'images.
- Créez des stratégies d'exécution pour surveiller les changements de fichier système.
- Implémentez un mécanisme de réponse automatique.
- Assurez-vous d'avoir des données d'analyse forensique complètes.

Options de sécurité Kubernetes : DIY vs Turnkey

Étapes pour sécuriser Kubernetes	Open source (DIY)	Sysdig Secure (Turnkey)
Prévention des menaces avec les contrôleurs d'admission	Le contrôleur d'admission Kubernetes peut s'intégrer à un moteur d'analyse pour vérifier si les images sont exemptes de vulnérabilités.	Sysdig Secure intègre l'analyse au pipeline CI/CD. Il fournit des politiques prêtes à l'emploi couvrant les meilleures pratiques de sécurité et normes de conformité.
	kube-psp-advisor est un outil qui simplifie la création de K8s Pod Security Policies (PSP) soit depuis un environnement K8s en direct, soit depuis un seul fichier .yaml contenant une spécification de pod.	Empêche les images risquées d'être déployées (via le contrôle des admissions Kubernetes). Vous pouvez analyser directement le pipeline et empêcher les images risquées d'aller dans la registry. Vous disposez d'intégrations et alertes prêtes à l'emploi avec des outils comme Slack, SNS, PagerDuty, etc.
Sécuriser le plan de contrôle Kubernetes	Validez la conformité de la configuration des clusters sur la base des benchmarks CIS pour Kubernetes (kube-bench). K8-security-configwatch peut évaluer les modifications de vos fichiers de configuration Kubernetes et mettre en évidence celles susceptibles d'affecter la sécurité du cluster. Utilisez Falco pour détecter les activités inattendues du plan de contrôle de Kubernetes.	Obtenez une visibilité approfondie sur des centaines de milliers de nœuds avec des tableaux de bord prêts à l'emploi pour surveiller les activités du plan de contrôle Kubernetes. Détectez les activités anormales plus rapidement avec des règles Falco gérées sur la base des journaux d'audit Kubernetes avec des intégrations de remédiation, d'alerte et de notification automatiques. Planifiez des évaluations de conformité en continu et générez des rapports basés sur les benchmarks CIS pour Kubernetes.

Étapes pour sécuriser Kubernetes	Open source (DIY)	Sysdig Secure (Turnkey)
Sécuriser les workloads lors de l'exécution	Falco, le projet open source de sécurisation du runtime en environnement cloud natif, est le moteur de détection des menaces Kubernetes de facto. Falco détecte les comportements d'application inattendus et alerte sur les menaces lors de l'exécution.	Détectez les nouvelles vulnérabilités lors de l'exécution et associez les images risquées à un namespace, cluster, déploiement, pod, etc. spécifique. Gagnez du temps concernant la détection des activités anormales en enrichissant Falco de règles prêtes à l'emploi. Améliorez la productivité DevOps en utilisant le profilage d'image basé sur le Machine Learning. Obtenez plus de visibilité sur l'ensemble du trafic réseau dans les containers fonctionnant dans des environnements hybrides/multicloud.
Segmentation réseau Kubernetes native	Les politiques réseau de Kubernetes sont une ressource native qui vous permet de définir comment un pod est autorisé à communiquer avec plusieurs « entités » dans le réseau. Les politiques réseau sont implémentées par un plugin comme Calico.	Réduisez les risques avec une visibilité réseau qui permet une microsegmentation en quelques minutes. La visibilité importante fournit des garde-fous pour les équipes n'ayant pas de compétences en matière de sécurité Kubernetes. Implémentez les bonnes politiques avec une vue unifiée et un contexte partagé dans les équipes. Simplifiez la gestion du réseau en automatisant les politiques réseau K8s.
Réponse aux incidents et analyse forensique	Sysdig est un outil open source sur systèmes Linux pour l'exploration et de dépannage des containers.	Accélérez la réponse aux incidents avec des enregistrements d'audit complets et des données d'analyse forensique approfondies. Réagissez plus rapidement via l'auto-remédiation et l'envoi d'alertes. Validez la conformité d'exécution avec des stratégies mappées sur différentes normes de conformité (NIST, PCI, SOC2).

Découvrez-en davantage sur la sécurité Kubernetes offerte par Sysdig.

Pour en savoir plus sur la manière de sécuriser Kubernetes, téléchargez notre Guide de sécurité Kubernetes.