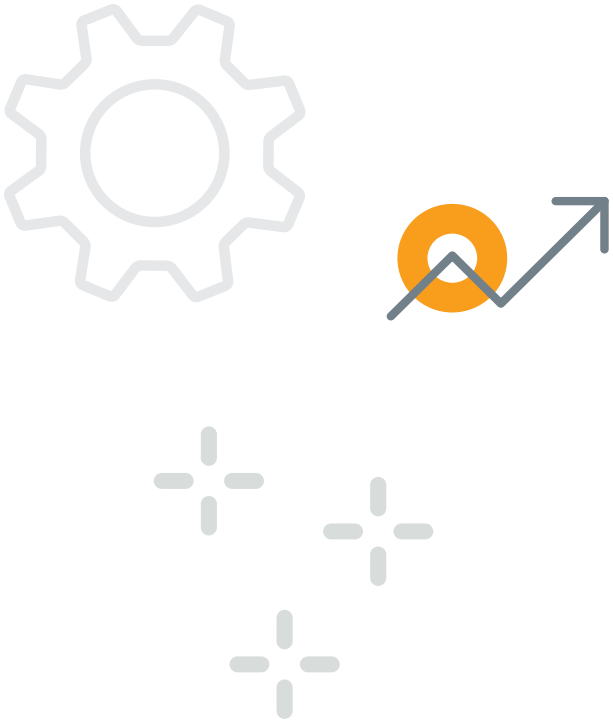
Le Guide moderne de la **supervision et de l'orchestration des conteneurs**

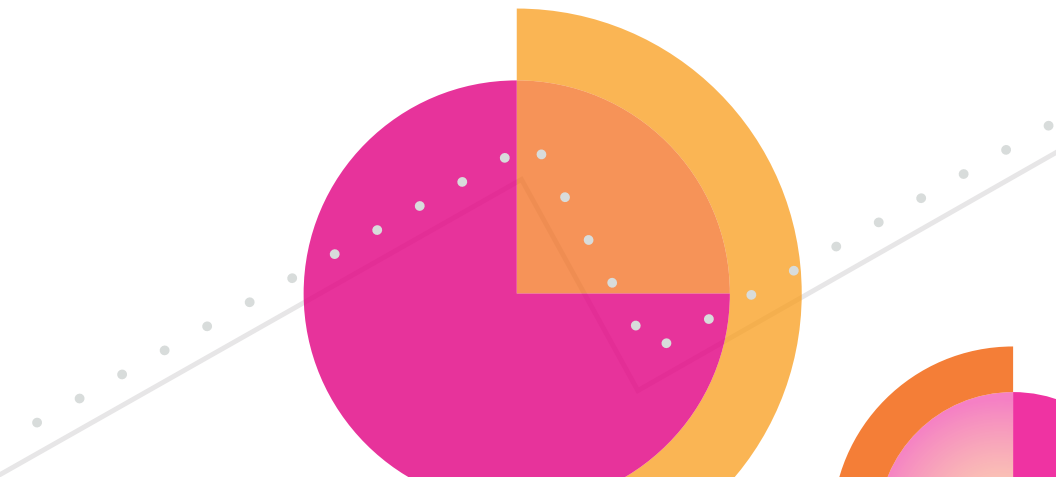
splunk>
turn data into doing



Depuis l'apparition du concept en 2013, les conteneurs sont devenus la principale tendance du monde informatique. Et on comprend aisément pourquoi : La technologie des conteneurs applicatifs révolutionne le développement des applications et apporte une flexibilité et une efficacité jusqu'ici impensable au processus de développement.

Les entreprises adoptent les conteneurs en grand nombre. Selon **Gartner**, plus de 85 % des entreprises du monde utiliseront des applications en conteneurs en production d'ici 2025, alors qu'elles étaient moins de 35 % à le faire en 2019. Cette adoption massive indique clairement que les entreprises ont besoin d'adopter une approche de développement basée sur les conteneurs pour rester compétitives.

Voyons ce qu'implique la conteneurisation et comment votre entreprise peut prendre l'avantage.

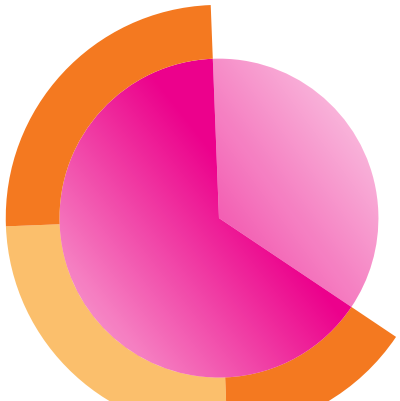


Qu’est-ce qu’un conteneur ?

La façon la plus simple de comprendre le concept de conteneur est de partir de sa désignation. Un conteneur physique est un réceptacle utilisé pour contenir et transporter des marchandises d’un emplacement à un autre.

Un conteneur logiciel remplit une fonction similaire. Il vous permet d’emballer le code d’une application, ses fichiers de configuration, des bibliothèques, des outils système et tout ce dont vous avez besoin pour exécuter cette application, dans une unité autonome que vous pouvez déplacer et exécuter où bon vous semble.

Les conteneurs sont un élément clé d’une approche en « microservices ». Cette approche divise les applications en modules à une seule fonction, utilisés uniquement lorsqu’il le faut. Le développeur peut modifier et redéployer un service particulier (et non l’application au complet) à chaque fois que des changements sont nécessaires.



Pourquoi les conteneurs font-ils autant de bruit ?

Les conteneurs apportent une solution à un problème bien trop courant des opérations : faire en sorte qu'un logiciel fonctionne de façon fiable et uniforme, quel que soit l'endroit où il est déployé. Lorsqu'une application passe d'un environnement à un autre, du staging à la production, par exemple, elle peut rencontrer des problèmes si le système d'exploitation, la topologie du réseau, les politiques de sécurité et autres aspects changent. Les conteneurs isolent l'application de son environnement et donc de ces variations. Comme chaque conteneur inclut ses dépendances, cette pratique évite également les problèmes qui surviennent lorsque des composants exigent des versions différentes de la même bibliothèque partagée ou d'autres dépendances.

Avant les conteneurs, les machines virtuelles (VM) étaient la principale méthode employée pour exécuter plusieurs applications isolées sur un même serveur. Comme les conteneurs, les VM sont isolées de l'infrastructure sous-jacente d'une machine, de sorte que les changements de matériel et de logiciel n'affectent pas les performances. Pour autant, conteneurs et VM ne procèdent pas du tout de la même façon.

Une VM s'affranchit du matériel pour transformer un serveur physique en une pluralité de serveurs virtuels. Pour ce faire, elle s'exécute au-dessus d'un hyperviseur, lui-même installé sur un ordinateur physique appelé « machine hôte ». L'hyperviseur est essentiellement un système de coordination qui arbitre l'accès aux ressources d'une machine hôte (CPU, RAM, etc.) et les met à disposition de la VM ou « machine invitée ». Les applications et tout ce qu'il faut pour les exécuter, y compris les bibliothèques et les binaires, sont contenus dans la machine invitée. Chaque machine invitée comprend également un système d'exploitation complet qui lui est propre. Un serveur qui héberge quatre VM, par exemple, va donc avoir quatre systèmes d'exploitation en plus de l'hyperviseur qui les coordonne tous. Cela exerce une forte pression sur les ressources de la machine et les choses peuvent rapidement s'enliser et limiter, à terme, le nombre de VM qu'un seul serveur peut supporter.

Les conteneurs, quant à eux, sont isolés du système d'exploitation. Un seul système d'exploitation est exécuté sur l'hôte (qui peut être un serveur physique, une VM ou un hôte cloud), et les conteneurs, grâce à un moteur de conteneurisation comme Docker Engine, partagent le noyau de cet OS avec d'autres conteneurs, chacun possédant son propre espace utilisateur isolé. Cette approche mobilise beaucoup moins de ressources qu'une machine virtuelle, ce qui rend les conteneurs beaucoup plus légers et efficaces et permet donc de bien mieux exploiter les ressources du serveur.

5 avantages du déploiement de conteneurs

Une infrastructure en conteneurs offre un grand nombre d'avantages. Voici les cinq principaux.

1. **Rapidité de livraison :** les applications installées sur une machine virtuelle mettent généralement plusieurs minutes à se lancer. Les conteneurs n'ont pas besoin d'attendre que le système d'exploitation démarre et peuvent donc être opérationnels en une fraction de seconde. Ils sont également plus rapides parce qu'ils utilisent moins de ressources d'OS, et quelques secondes suffisent à les créer, les cloner ou les détruire. Tout cela a un impact considérable sur le processus de développement et permet aux entreprises d'accélérer la mise sur le marché des logiciels, la correction des bugs et l'ajout de nouvelles fonctionnalités.

2. **Une approche orientée DevOps :** la vitesse des conteneurs en microservices, leur faible encombrement et leur économie de ressources les rendent parfaits pour un environnement DevOps. Une infrastructure basée sur des microservices permet aux développeurs de garder la main sur des parties spécifiques de l'application de bout en bout : ils sont ainsi en mesure de comprendre parfaitement son fonctionnement, d'optimiser ses performances et de résoudre les problèmes plus efficacement qu'avec des applications monolithiques.

3. **Portabilité :** les conteneurs englobent l'application et toutes ses dépendances. On peut ainsi facilement déplacer et exécuter des conteneurs sur du matériel Windows, Linux ou Mac, avec une grande fiabilité. Les conteneurs fonctionnent aussi bien sur des serveurs physiques locaux que sur des serveurs virtuels, dans un cloud privé ou public. Cela permet également de garder une certaine indépendance vis-à-vis des fournisseurs quand vous souhaitez migrer vos applications d'un environnement de cloud public vers un autre.

4. **Évolutivité renforcée :** les conteneurs restent généralement compacts parce qu'ils ne nécessitent pas d'OS propre, contrairement aux VM. La taille d'un conteneur se compte généralement en dizaines de mégaoctets, tandis qu'on parle plutôt de dizaines de gigaoctets pour les VM, soit un rapport de 1 à 1 000 environ. Cette efficacité permet de stocker bien plus de conteneurs sur un même système d'exploitation hôte, rendant cette approche bien plus évolutive.

5. **Cohérence :** comme les conteneurs conservent toutes les dépendances et configurations en interne, les développeurs peuvent travailler dans un environnement cohérent quel que soit l'endroit où les conteneurs sont déployés. Les développeurs ne perdent plus de temps à résoudre les problèmes liés aux différences d'environnement et peuvent se concentrer sur l'ajout de fonctionnalités aux applications. Il est également possible de faire passer un même conteneur du développement à la production le moment venu. Enfin, comme les conteneurs sont immuables une fois créés, les développeurs n'ont pas à s'inquiéter des différences de configuration lors du déploiement ni d'autres sources de difficultés.

Les fondamentaux de l'orchestration : utiliser Kubernetes

Pour bien démarrer avec l'orchestration de conteneurs, il faut un logiciel spécialisé pour déployer, gérer et faire évoluer les applications en conteneurs. Kubernetes est l'une des options les plus établies et les plus populaires aujourd'hui. Cette plateforme open source d'automatisation développée par Google est aujourd'hui gérée par la Cloud Native Computing Foundation.

Kubernetes peut considérablement améliorer le processus de développement en simplifiant la gestion des conteneurs, en automatisant les mises à jour et l'évolutivité et en minimisant les interruptions, pour permettre aux développeurs de se concentrer sur l'amélioration des applications et l'ajout de nouvelles fonctionnalités. Pour mieux comprendre comment, examinons les composants de base de Kubernetes et la façon dont ils fonctionnent ensemble.

Kubernetes utilise plusieurs couches d'abstraction définies dans son propre langage. Kubernetes présente de nombreuses facettes. La liste n'est pas exhaustive, mais elle offre un bon aperçu de la façon dont le matériel et le logiciel sont représentés dans le système.

Nœuds : dans le langage de Kubernetes, une « machine de travail » est un nœud. Il peut s'agir d'un serveur physique, d'une machine virtuelle ou d'un fournisseur de cloud comme AWS ou Microsoft Azure. À l'origine, les nœuds étaient appelés « minions » (laquais), ce qui vous donne une idée de leur usage. Ils reçoivent et accomplissent les tâches qui leur sont affectées par le nœud-maître et contiennent tous les services requis pour gérer et attribuer des ressources aux conteneurs.

Nœud-maître : c'est la machine qui orchestre tous les nœuds de travail et c'est votre point d'interaction avec Kubernetes. C'est de là que proviennent toutes les tâches affectées.

Cluster : un cluster est composé d'un nœud-maître et de plusieurs nœuds de travail. Les clusters regroupent toutes ces machines au sein d'une même unité puissante. Les applications en conteneurs sont déployées dans un cluster et le cluster distribue la charge aux différents nœuds, réorganisant le travail au fil de l'ajout et du retrait des conteneurs.

Pods : un pod est une collection de conteneurs mis en paquet et déployés sur un nœud. Tous les conteneurs d'un même pod partagent un réseau local et d'autres ressources. Ils peuvent se parler comme s'ils se trouvaient sur la même machine mais restent isolés les uns des autres. Dans le même temps, les pods isolent le réseau et le stockage du conteneur sous-jacent.

Un même nœud de travail peut contenir plusieurs pods. Si un nœud subit une défaillance, Kubernetes peut déployer un pod de remplacement sur un nœud opérationnel.

Bien qu'un pod puisse englober de nombreux conteneurs, on recommande de ne pas en inclure plus que nécessaire : un processus principal et ses conteneurs de soutien, appelés « sidecars ». Les pods grossissent quels que soient leurs besoins réels, et des pods surchargés peuvent exercer une pression importante sur les ressources.

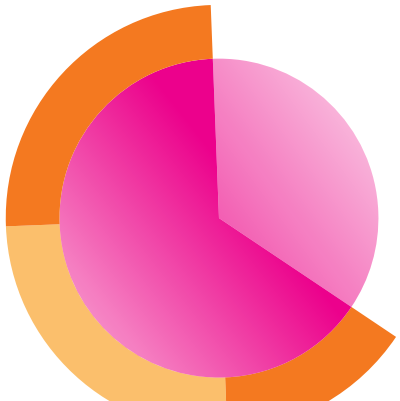
Déploiements : au lieu de déployer directement les pods sur un cluster, Kubernetes utilise une couche d'abstraction supplémentaire appelée « déploiement ». Un déploiement vous permet de préciser le nombre de répliques d'un pod que vous souhaitez exécuter simultanément. Une fois qu'il a déployé ce nombre de pods sur un cluster, il les supervise en continu et, en cas de défaillance d'un pod, le recrée et le redéploie automatiquement.

Ingress : Kubernetes isole les pods du monde extérieur, vous devez donc ouvrir un canal de communication vers tout service que vous souhaitez exposer. Cette nouvelle couche d'abstraction s'appelle « ingress ». Il existe plusieurs manières d'ajouter un ingress à un cluster, dont le LoadBalancer, le NodePort et le contrôleur d'ingress. Vous pouvez le voir comme le serveur web connecté à Internet que vous avez peut-être utilisé dans une architecture traditionnelle.

Quels défis Kubernetes et la conteneurisation présentent-ils pour la supervision ?

Si les conteneurs et les frameworks d'orchestration apportent une myriade d'avantages aux organisations, ils ont aussi tendance à complexifier la gestion des applications cloud. Ils s'accompagnent notamment de quelques difficultés.

- **Des angles morts importants :** les conteneurs sont faits pour être jetables. Pour cette raison, ils introduisent plusieurs couches d'abstraction entre l'application et le matériel sous-jacent, à des fins de portabilité et d'évolutivité. Tout cela crée un angle mort significatif du point de vue de la supervision conventionnelle. Les outils de supervision traditionnels ne sont pas capables de comprendre ces abstractions.



- **Augmentation du volume de données :** du fait de la portabilité aisée d'autant de composants interdépendants, il devient plus crucial encore de conserver des données de télémétrie pour assurer l'observabilité des performances et la fiabilité la plateforme d'applications, de conteneurs et d'orchestration. De nombreuses architectures de microservices sont conçues pour faire augmenter leur nombre en cas de besoin et les détruire lorsqu'ils ne sont plus utiles. Du fait de ce caractère éphémère, il est d'autant plus essentiel d'envoyer les données dans un système d'observabilité. L'ajout de composants supplémentaires au système augmente également le nombre d'éléments à superviser et à contrôler en cas de problème.

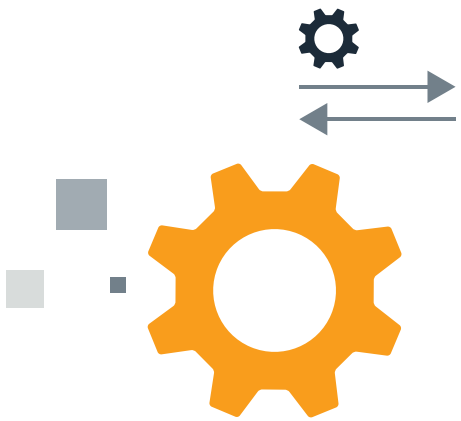
- **L'importance des visualisations :** pour faire face à l'échelle et à la complexité introduites par les microservices, les conteneurs et l'orchestration, il faut pouvoir à la fois visualiser l'environnement pour obtenir un aperçu immédiat de la santé de votre infrastructure, mais aussi déterminer de quelle façon le trafic circule dans votre environnement. Vous devez également pouvoir zoomer et consulter l'état de santé et les performances des conteneurs, des nœuds et des pods. La bonne solution de supervision doit proposer ce workflow.

- **Au rythme du DevOps :** les conteneurs peuvent être démultipliés et modifiés à la vitesse de l'éclair. Avec ce rythme de déploiement accéléré, les équipes DevOps ont plus de difficultés à suivre les fluctuations des performances de leurs applications sur les différents déploiements, ou même à savoir à quel moment de nouvelles dépendances de services sont ajoutées.

Comment implémenter des conteneurs

Une bonne solution de supervision des conteneurs vous permettra de conserver une bonne vision d'ensemble de votre environnement dynamique de conteneurs en unifiant les données des conteneurs à celles du reste de l'infrastructure pour offrir une meilleure contextualisation et rendre possible l'analyse des causes profondes. Voyons comment mettre en place plusieurs couches de supervision pour Docker, l'implémentation la plus répandue :

Hôtes : la disponibilité et les performances des machines physiques et virtuelles de vos clusters peuvent être supervisées. Pour cela, nous allons suivre des métriques clés comme la mémoire, l'utilisation du CPU, la taille du fichier d'échange et l'espace de stockage occupé. Ce doit être une capacité de base de n'importe quel outil de supervision des conteneurs.



Conteneurs : il est crucial d'avoir une visibilité sur vos conteneurs sous forme agrégée et individualisée. Un outil de supervision peut fournir des informations sur le nombre de conteneurs en cours d'exécution, ceux qui consomment le plus de mémoire et ceux qui ont démarré le plus récemment. Il doit également fournir des informations sur la consommation de CPU et de mémoire de chaque conteneur, ainsi que sur l'état de ses E/S réseau.

Framework d'orchestration : Kubernetes lui-même doit également être supervisé. Combien y a-t-il de nœuds disponibles ? Quel est l'état de santé du nœud maître ? Est-ce que des pods restent en attente de réaffectation pendant de longues périodes ? Quel volume de trafic transite par votre ingress ? Vous devez être en mesure de répondre rapidement à toutes ces questions pour exploiter durablement des services fiables. De plus, en raison de la nature de Kubernetes, les pods sont planifiés de façon à optimiser l'utilisation des ressources. On gagne ainsi en efficacité, mais l'emplacement de déploiement et d'exécution des pods devient plus imprévisible.

Points de terminaison des applications : il faut encore évaluer si votre service est capable de gérer les demandes des utilisateurs, et déterminer les performances et la latence de ces demandes. Votre solution de supervision doit également contrôler l'état de santé de l'application elle-même afin de mesurer la latence et d'autres critères de performances.

Quelles sont les fonctionnalités nécessaires pour superviser ces applications ?

Comme nous l'avons vu, la conteneurisation des applications et l'utilisation de frameworks d'orchestration comme Kubernetes apportent de nombreux avantages aux workflows de développement et de déploiement modernes. La supervision de ces environnements et applications reste toutefois bien plus compliquée que l'utilisation d'outils hérités. Une solution de supervision prête pour les conteneurs et les workflows d'orchestration doit offrir plusieurs fonctionnalités essentielles.

Collecte d'indicateurs clés

Métriques des pods : nombre de pods souhaités, nombre de pods disponibles, pods par phase (échec, attente, exécution), pods souhaités par déploiement

Métriques d'utilisation des ressources : métriques collectées par Docker Socket (à l'échelle des conteneurs et de chaque nœud, comme la consommation de CPU et de mémoire)

Métriques d'application : métriques RED (taux, erreur, durée) ; santé des applications ; disponibilité et performances de la base de données

Fonctions de consolidation, de corrélation et d'analyse

Déploiement simplifié : le déploiement des collecteurs doit être facile et cloud-native. Idéalement, on dispose d'un graphique Helm. Le déploiement peut être très simple :

```
helm repo add splunk-otel-collector-chart https://signalfx.github.io/splunk-otel-collector-chart
```

```
helm repo update
```

```
helm install --set splunkAccessToken='xxx' --set clusterName='sample' --set splunkRealm='us0' --set otelCollector.enabled='true' --generate-name splunk-otel-collector-chart/splunk-otel-collector
```

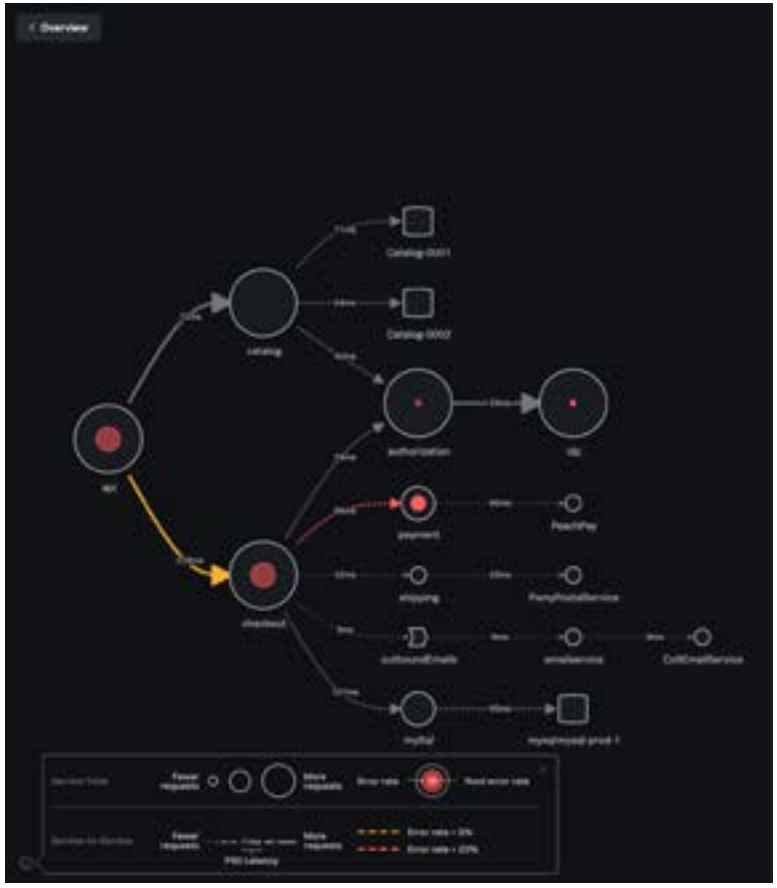
Indépendance vis-à-vis des fournisseurs : OpenTelemetry est l'avenir de la technologie de supervision, et l'instrumentation de votre environnement doit tenir compte du fait que vous pouvez décider de changer de fournisseur de supervision ou d'observabilité. La solution que vous adopterez doit impérativement prendre en charge OpenTelemetry afin que vous n'ayez pas à refaire de travaux d'instrumentation si vous changez d'outil.

Plateforme d'analyse de flux en temps réel : dans un monde où les temps d'arrêt peuvent coûter des centaines de milliers de dollars par heure, chaque seconde compte. Une plateforme moderne de supervision et d'observabilité doit être capable de produire des alertes et d'analyser les données en temps réel pour minimiser le MTTD et veiller à ce que les conclusions que vous tirez des données reposent bien sur l'état actuel du système.

Analyse prédictive : basés sur l'IA et le ML, les outils d'analyse prédictive peuvent vous prévenir à l'avance qu'un composant est sur le point de subir une défaillance. Compte tenu de la complexité des environnements modernes, ils peuvent même empêcher les problèmes de se produire et vous aider à détecter les défauts potentiels de performances et de disponibilité avant qu'ils n'affectent vos clients.

Tableaux de bord prédéfinis : vous ne devriez pas être contraint de documenter l'intégralité de votre infrastructure pour tirer parti de votre système de supervision. Des tableaux de bord automatisés et intégrés peuvent vous aider à comprendre les corrélations complexes entre les nœuds, les pods, les conteneurs et les applications, et à voir l'état de votre environnement en un coup d'œil :

Cartes des services automatisées : dans ce monde nouveau, il est extrêmement difficile de comprendre comment le trafic circule à travers l'ingress, les conteneurs et les applications. Votre outil de supervision doit être capable de comprendre les chemins empruntés par vos requêtes et de vous les montrer, en plus d'identifier les erreurs et autres problèmes se manifestant dans votre environnement :



Étapes suivantes

Les conteneurs sont un puissant outil de votre arsenal de développement, mais il faut impérativement comprendre comment fonctionnent vos environnements de conteneurs et quelles sont leurs performances. L'importance de la supervision de l'infrastructure et des performances des applications ne fait que croître avec le déploiement des conteneurs. Une solution de supervision à la hauteur des conteneurs doit répondre à plusieurs critères : elle doit être conçue pour comprendre les conteneurs, permettre un déploiement facile, et offrir une plateforme de streaming en temps réel, des analyses prédictives et une expérience prête à l'emploi offrant des données utiles. Si vous êtes prêt à adopter un système d'observabilité qui fait tout cela et peut fonctionner nativement avec les conteneurs, les clouds publics, les clouds privés et les environnements auto-hébergés, consultez une démo de Splunk Observability Cloud, ou commencez un essai gratuit dès aujourd'hui. Pour en savoir plus sur l'observabilité, visitez notre site web ou téléchargez notre Guide pratique de l'observabilité.

splunk>

21-14769-Splunk-Modern Guide