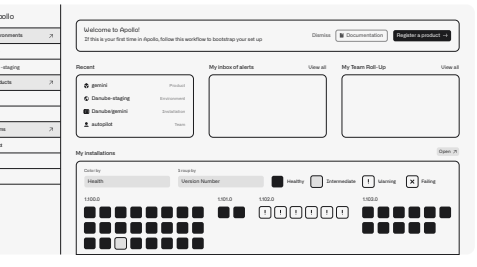
Palantir

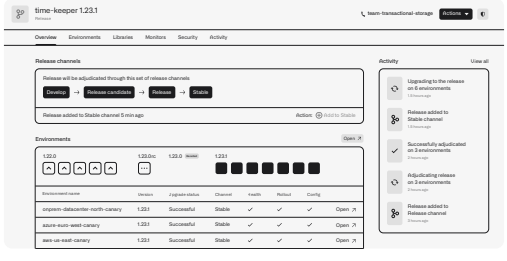
JOURNÉE DE DÉMONSTRATION APOLLO

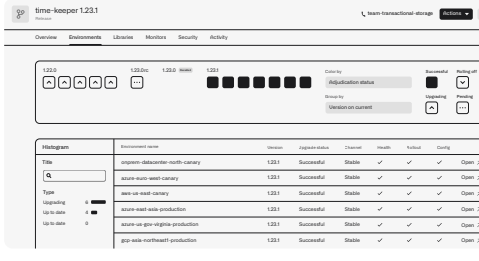
April 27, 2022
→ French Transcript

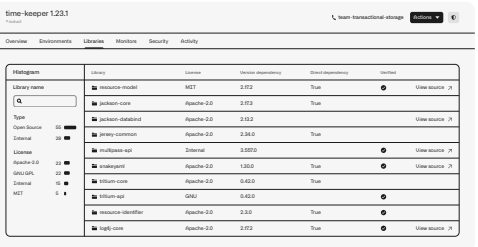
This transcript is provided as a courtesy and is intended to be viewed, and is subject to, the accompanying oral presentation and related materials, including any legal disclaimers.

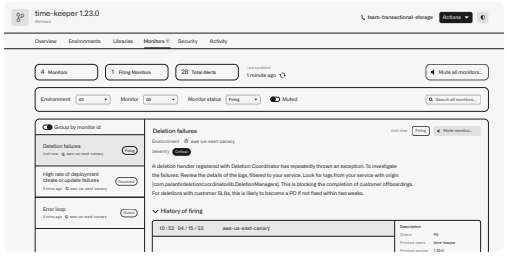
Index

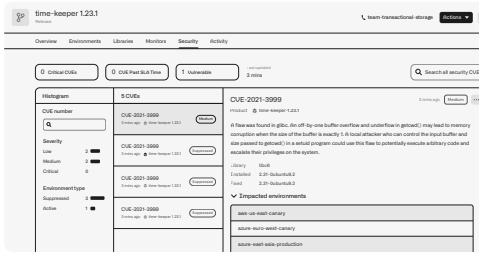












Why We Built Apollo

Shyam Sankar
→ Chief Operating Officer

Chez Palantir, nous parlons toujours de travailler sur les problèmes les plus difficiles du monde, et par le plus dur, nous ne parlons pas de ce qui est le plus complexe, le plus cérébral ou le plus technique. Nous parlons de ce qui est existentiel. Nous construisons Apollo depuis six ans parce que nous pensons que la fourniture de logiciels est l'un de ces problèmes existentiels. Et franchement, je ne suis pas sûr que quelqu'un d'autre l'ait vraiment fait. Puis SolarWinds est arrivé, puis Log4j. Et il est malheureusement évident que la livraison et la gestion de logiciels, c'est-à-dire la chaîne d'approvisionnement des logiciels constitue un problème existentiel pour les entreprises modernes.

Nous pouvons simplement attester de l'utilité et de l'efficacité d'Apollo parce que nous l'utilisons pour déployer nos propres logiciels. Apollo assure la gestion, le déploiement et la maintenance de Foundry et Gotham dans le monde entier, et prend en charge des missions critiques dans certains des environnements les plus difficiles que l'on puisse imaginer, des satellites aux sous-marins, des réseaux haute altitude aux Humvees et, bien sûr, tous les types de cloud et sur site. Apollo gère des logiciels dans plus de 250 environnements clients distincts. C'est ce qui nous permet de tenir nos promesses et d'être au rendez-vous.

Cette fragmentation des environnements de diffusion change fondamentalement la donne pour les fournisseurs de logiciels. Apollo résout ce problème. Il permet aux ingénieurs d'écrire une seule fois un code qui fonctionne dans tous les environnements. Il permet au même logiciel de répondre aux besoins uniques de chaque environnement en permettant au développeur d'encoder ses propres conditions préalables dans le comportement attendu de l'environnement, que ce soit sur site, dans le cloud ou à la périphérie, afin que les développeurs puissent se concentrer sur la création de meilleurs produits et éviter tout le travail humain nécessaire pour amener ces produits là où ils doivent se trouver. Apollo est à la base de la manière dont nous avons produit nos logiciels au cours des six dernières années, et il va le rester au cours des 60 prochaines années. Je suis ravi de partager toute la puissance de la plateforme avec vous aujourd'hui.

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

Vous savez, les microservices et les architectures orientées services sont devenus si populaires en raison de la promesse d'une innovation plus rapide grâce à la distribution des tâches, et pour que vous puissiez réellement profiter de ces avantages, les équipes doivent être en mesure d'évoluer à des vitesses différentes. Les équipes doivent être en mesure de livrer les nouvelles fonctionnalités et capacités dès qu'elles sont prêtes, au lieu d'être ralenties et d'attendre un calendrier de sortie hebdomadaire, mensuel ou trimestriel qui est déterminé par l'élément le plus lent, ou par votre capacité à, par exemple, assurer la qualité de la plateforme. Et dans une architecture orientée vers les services, il va y avoir des dépendances d'API entre les différents services. Mais il faut une capacité à expédier les nouvelles versions à des rythmes différents.

Ainsi, dès le départ, Apollo a été conçu pour résoudre ces problèmes intrinsèques liés aux systèmes distribués en permettant aux équipes d'ingénieurs logiciels d'encoder les conditions préalables et les attentes de leurs logiciels à côté du code lui-même. Apollo prend ensuite en compte ces contraintes intégrées et veille à ce qu'elles soient respectées lors de la mise à niveau, ce qui permet d'automatiser en toute sécurité le retour en arrière d'un service donné sans avoir d'impact sur la disponibilité de l'ensemble de la plate-forme. Ainsi, au moment de la compilation, les systèmes de stockage sont en mesure de déclarer l'ensemble des schémas qu'ils peuvent lire et dans lesquels ils peuvent écrire, les étapes de migration spécifiques qu'ils prennent en charge et un moyen de signaler la version actuelle de leurs schémas lors de l'exécution. Apollo peut ainsi mettre à niveau en toute sécurité les services de stockage par l'intermédiaire de migrations de schémas et de retours en arrière si nécessaire, avec des mécanismes de sécurité qui empêchent la corruption des données.

Le deuxième principe. Un développeur de logiciel ne devrait pas avoir besoin d'une compréhension approfondie de l'environnement ou du système d'infrastructure dans lequel son logiciel sera exécuté pour renforcer des capacités qui permettent de réaliser des logiciels de classe mondiale. Lorsque le monde construisait et distribuait des monolithes logiciels et les déployait dans une ou deux machines ou serveurs virtuels, le problème de déploiement le plus difficile auquel les développeurs étaient confrontés était d'essayer de comprendre Linux. Mais aujourd'hui, l'infrastructure sur laquelle les logiciels sont construits est beaucoup plus compliquée, et cela ne fait qu'empirer. Cela étant dit, les systèmes

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

d'infrastructure disponibles aujourd'hui permettent de fournir les capacités non négociables des logiciels modernes. Il s'agit notamment de logiciels de sécurité des données, de chiffrement et de réseau hautement disponibles et pouvant être mis à jour sans temps d'arrêt pour l'utilisateur, ainsi que de la capacité à se déployer sur plusieurs fournisseurs de cloud et à la périphérie. Tous ces éléments sont de plus en plus cruciaux pour les logiciels critiques, mais pour qu'un développeur construise et déploie une application Java ou Python moderne qui doit être hautement disponible, elle a peut-être besoin d'une certaine configuration, de certificats pour servir le trafic HTTPS et s'exposer via un proxy de porte d'entrée, il doit comprendre, configurer et maintenir six ou sept objets Kubernetes différents, en plus des composants de l'infrastructure en cloud qui exposent les équilibreurs de charge du réseau. Ce n'est tout simplement pas une bonne utilisation emploi du temps pour la plupart des développeurs. Apollo fournit donc un SDK que les développeurs peuvent utiliser pour créer des applications logicielles modernes. Le SDK d'Apollo permet au développeur de déclarer les propriétés de son logiciel, par exemple, vous savez, j'ai peut-être le besoin d'un volume local persistant, ou j'ai besoin de trois répliques du service, ouet quel lee quorum pour mon le service est soit que chaque réplique soit disponible. Apollo peut ensuite générer et gérer les contrôleurs de pods Kubernetes appropriés en fonction de ces propriétés déclarées. Un certificat et les clés publiques et privées correspondantes sont générés et fournis au service, qui peut ainsi configurer le chiffrement TLS pour chaque trafic invité sans avoir besoin de comprendre comment le matériel PKI est créé ou géré par la plateforme. Le développeur est en mesure de déclarer que son service produit un point de terminaison API et de demander que ce point de terminaison soit exposé via un proxy inverse à un certain chemin, le tout sans avoir besoin de comprendre les complexités de la configuration du système d'entrée du cluster Kubernetes ou le fonctionnement des objets d'entrée. Toutes ces capacités sont possibles sans modifier une seule ligne de code.

Le troisième principe consiste à rendre possible le déploiement des logiciels là où ils sont nécessaires en permettant aux plateformes logicielles d'être répétables, reproductibles et portables entre les fournisseurs de services en cloud et les types d'infrastructure. Par exemple, de nombreuses entreprises peuvent trouver décourageante l'idée de livrer leur logiciel dans un nouvel environnement, car leur plateforme est presque impossible à configurer à partir de zéro, et encore moins à exploiter plusieurs installations à la fois. D'autres ont des

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

pipelines de déploiement hautement adaptés à un seul environnement, où l'introduction d'un nouvel environnement de production briserait fondamentalement leur processus de déploiement existant. L'accès au marché n'est tout simplement pas un objectif réalisable pour beaucoup, car la technologie et les outils utilisés pour fournir le logiciel ont été conçus pour ce paradigme d'environnement SaaS unique. Et alors que le monde a évolué, l'outillage n'a pas changé. Apollo adopte donc une approche très différente de celle des autres outils de l'espace DevOps ou de l'ingénierie des plateformes. Apollo commence par modéliser les logiciels que vous souhaitez exécuter dans vos environnements et votre entreprise dans un catalogue logiciel unique. Cela permet à toutes les parties de partager un point de vue unique de tout ce qui est connu à propos d'un logiciel, depuis ce qui est déclaré par les développeurs, des choses comme, vous savez, les dépendances entre services, les versions de schéma prises en charge, les demandes de ressources, etc. jusqu'aux informations qui sont complétées par des systèmes externes comme les scanners de virus ou de vulnérabilité ou les scanners de sécurité que vous utilisez déjà dans votre entreprise. Les différentes plateformes logicielles peuvent alors être définies comme la composition des services et produits individuels qui existent dans le catalogue Apollo. Apollo modélise ensuite chaque environnement dans lequel vous souhaitez déployer et gérer des logiciels et déclare un ensemble d'entités installations uniques d'un produit logiciel donné à partir du catalogue. Les canaux de publication sont utilisés pour représenter les différentes étapes par lesquelles passe un logiciel lorsqu'il est déployé dans vos environnements. Les canaux de diffusion vous permettent de modéliser des processus de déploiement non linéaires. Pour les modèles d'entreprise plus proches de celui de Palantir, qui vous obligent à gérer un grand nombre d'environnements et à déployer votre logiciel là où se trouvent vos clients, vous pouvez choisir d'utiliser des canaux de diffusion pour modéliser la confiance dans les nouvelles versions, où vous déployez d'abord votre logiciel dans des environnements canaris avant de le déployer dans des environnements plus critiques qui sont plus réticents à prendre des risques. Cette approche du déploiement des logiciels permet d'éviter les modes d'échec les plus douloureux des modèles traditionnels de déploiement continu axés sur le pipeline. Alors que les pipelines ont tendance à être propres à un seul service ou à un petit ensemble de services et à évoluer en silos le long des frontières de l'équipe, les canaux de diffusion offrent une approche cohérente pour votre historique de gestion de la production à travers les équipes, tout en permettant à chaque équipe de développement de contrôler les entrées

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

du système qui dictent la façon dont leur logiciel est déployé.
Enfin, le dernier principe, la gestion efficace des logiciels et de la production, dépasse le simple fait de déployer un logiciel et dépend de la collaboration active des développeurs, des opérateurs et des équipes chargées de la sécurité et de la conformité. Si nous avons appris quelque chose ces dernières années, c'est que les problèmes liés à la sécurité de la chaîne d'approvisionnement des logiciels et aux questions connexes ne sont pas près de disparaître. Tout comme nous n'avons pas besoin que chaque ingénieur comprenne les détails de l'infrastructure sur laquelle tourne son logiciel, nous n'avons pas non plus besoin qu'ils consacrent du temps à devenir des experts en matière de sécurité logicielle. Cela devrait faire partie intégrante de la gestion de la production et devrait être gratuit. Avec Apollo, le catalogue de logiciels devient un portail central qui régit les logiciels qui peuvent et ne peuvent pas être déployés en production, en fonction des politiques déterminées par les équipes de sécurité et de conformité. Sans entraver le processus naturel de développement des logiciels, il s'intègre à votre chaîne d'outils DevOps existante, comme votre système d'intégration continu ou vos registres de conteneurs d'artéfacts et vos scanners de vulnérabilité et de sécurité. Vos équipes de sécurité peuvent définir des politiques de vulnérabilité qui permettent à Apollo de rappeler automatiquement les versions qui violent ces politiques, ou de notifier à l'équipe de sécurité les versions qui ne sont pas conformes afin qu'elles soient soumises à un examen annuel rapide.

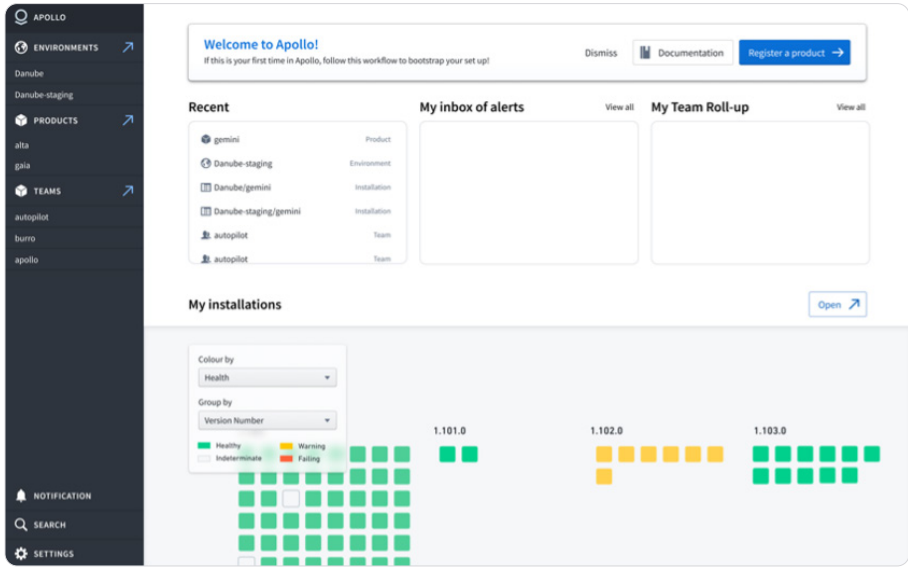
Après le déploiement du logiciel, Apollo vérifie en permanence la santé de chacune des installations logicielles si l'équipe de développement a opté pour le système de santé. Cela permet aux développeurs d'encoder les SLO et les autres critères que leur logiciel doit respecter pour être considéré comme sain. En outre, les développeurs peuvent également déclarer comment leur logiciel doit être surveillé dans l'artéfact logiciel lui-même, ce qui permet de définir le comportement attendu sans avoir à modifier le code. Apollo regroupe ces informations dans un seul écran et peut automatiser le retour en arrière et d'autres actions de remédiation actives et passives lorsque les SLO ne sont pas respectés, lorsque les contrôles de santé signalent des problèmes ou lorsque les moniteurs intégrés se déclenchent.

Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

Commençons par une démonstration pour comprendre comment chacun de ces groupes utilise Apollo au cours du cycle de vie d'une version. Imaginons que je sois un développeur qui utilise Apollo pour déployer continuellement mes logiciels dans tous mes environnements. Je viens de couper une version du produit Timekeeper sur la version 1.23.0. Immédiatement, je peux voir que ma version a été ajoutée au canal de diffusion approprié, conformément aux politiques de déploiement de mon équipe, pour toutes les versions logicielles étiquetées. En conséquence, Apollo a commencé à mettre automatiquement à niveau mes trois environnements, qui sont abonnés à ce canal de diffusion, pour les mises à jour du service timekeeper.

• Palantir Apollo Control Center



Je vais vous montrer la page de lancement de mon produit, qui me permet de tout comprendre sur cette version du logiciel à partir d'un seul point. D'ici, je peux voir que ma version actuelle est évaluée par rapport à mes environnements canaris définis et qu'elle est configurée pour s'identifier automatiquement comme stable, une fois que le déploiement est terminé et que toutes les installations restent saines pendant la période d'essai déclarée. Comme vous pouvez le voir, mon équipe a choisi des environnements canary qui couvrent plusieurs fournisseurs de clouds différents, ainsi qu'un centre de données sur site. Il semble qu'un de mes moniteurs soit en panne, ce qui signifie probablement que ma mise à niveau a échoué dans un ou plusieurs de mes environnements canaris. Lorsque cela se produit, la première chose

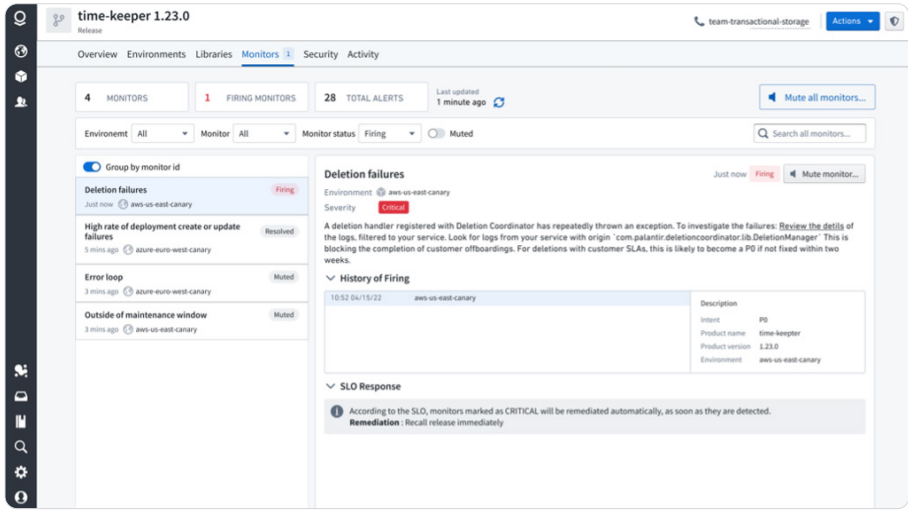
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

que je fais est de fouiller dans l'onglet « Moniteurs de versions » pour mieux comprendre quelles notifications sont déclenchées et ce que cela signifie pour ma diffusion. Les moniteurs me permettent d'encoder les objectifs de niveau de service de mon produit en contrôles de santé sur lesquels Apollo peut agir. Comme nous le voyons là, il semble que l'un des moniteurs critiques que j'ai encodé dans mon produit a échoué. Une erreur critique du moniteur est généralement un signe que quelque chose a sérieusement mal tourné sur ma version, et que la version n'est pas assez stable pour être déployée dans toute la flotte.

• Palantir Apollo Monitoring



Dans ce cas précis, c'est moi qui ai pris le contrôle de l'erreur du moniteur, mais c'est souvent un membre de l'une de nos équipes opérationnelles qui est chargé de gérer ces environnements. Comme j'ai pu encoder mes SLO dans le moniteur et qu'Apollo en assure automatiquement le suivi pour moi, mes collègues de l'équipe d'exploitation sont tout aussi capables que moi de trier les alertes et de prendre les mesures qui s'imposent pour garantir que mon logiciel apporte une valeur ajoutée fiable aux clients en production.

Pendant que nous parlions, Apollo a automatiquement rappelé ma version, non seulement en s'assurant que timekeeper 1.23.0 ne soit pas installé sur de nouveaux environnements, mais aussi en déclenchant une mise à niveau automatique pour les environnements canaris qui avaient déjà été mis à niveau. En retournant à la page de présentation de ma version 1.23.0, je constate qu'Apollo m'informe maintenant de

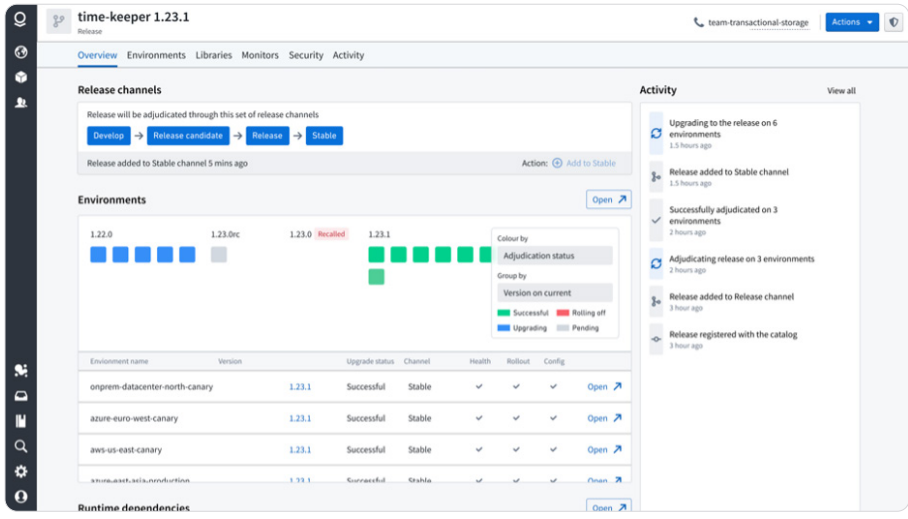
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

façon bien visible que cette version a été rappelée, de la raison pour laquelle elle a été rappelée, ainsi que de la stratégie adoptée par Apollo pour revenir à une bonne version connue. Le fait de disposer de toutes ces informations en un seul point fournit un contexte historique crucial pour moi et mon équipe. Comme vous pouvez le voir, le panneau d'activité s'est mis à jour au fur et à mesure qu'Apollo a entrepris une action automatique pour détecter mon moniteur, rappeler la version et commencer le retour à la dernière bonne version connue de mon produit. La capacité d'Apollo à suivre et à auditer de manière proactive toutes les actions au sein de mes environnements gérés permet aux produits que je développe de fonctionner dans des environnements qui peuvent exiger des contrôles de sécurité et de conformité stricts.

• Palantir Apollo Software Catalog



Comme nous pouvons le constater dans le panneau des environnements, Apollo vient de terminer le retour en arrière de ma mauvaise version 1.23.0 dans mes environnements canaris. Depuis qu'Apollo a pu lancer automatiquement le processus de déploiement de ma version dans les environnements canaris, détecter un problème sur ma version avant qu'elle n'atteigne les environnements de production critiques, puis rappeler et annuler immédiatement cette version, j'ai maintenant tout le temps de déboguer ce qui n'a pas fonctionné et de me concentrer sur l'amélioration de mon produit. J'ai corrigé ce problème et créé une nouvelle version de timekeeper 1.23.1. Cette version a été mise à niveau avec succès dans mes trois environnements

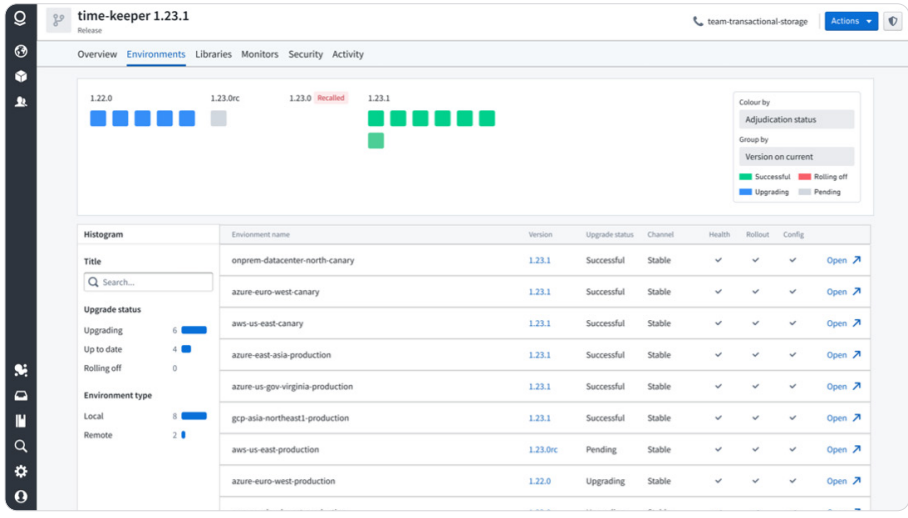
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

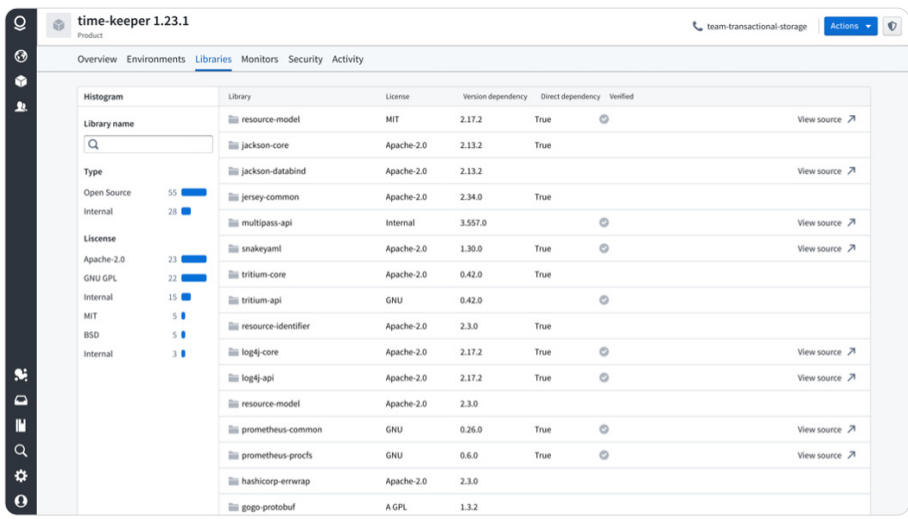
[CONT.]

canaris et n'a provoqué le déclenchement d'aucun moniteur pendant la période de stabilisation définie. Apollo a donc approuvé automatiquement la version 1.23.1 et l'a identifiée comme stable. Au sein de mon équipe, c'est le canal de diffusion que nous utilisons pour signifier qu'une version est prête à être mise en production, et qu'Apollo aura automatiquement lancé le processus de déploiement dans toute la flotte.

• Palantir Apollo Delivery



• Palantir Apollo Software Supply Chain



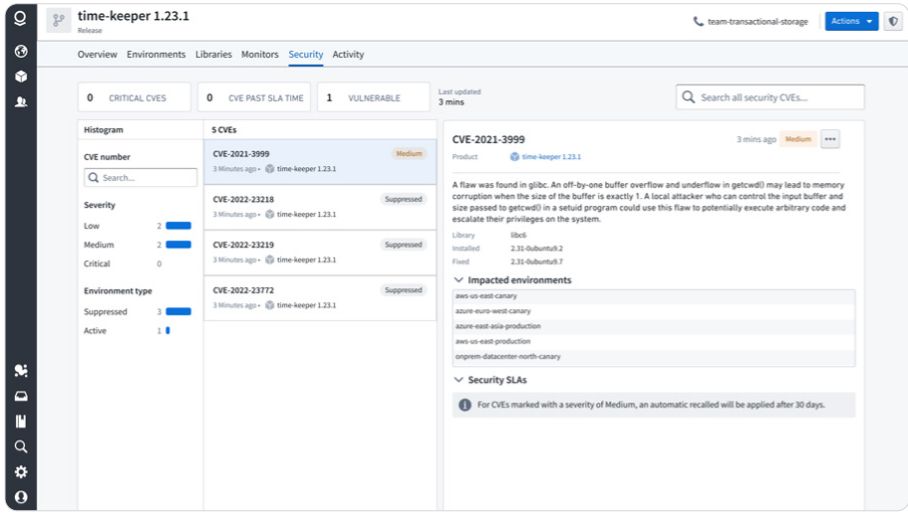
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

En plus des canaux de diffusion, vous m'avez déjà vu utiliser la vue des environnements d'Apollo pour comprendre rapidement l'état de mes versions dans les différents environnements. Les environnements dans Apollo sont des concepts critiques et peuvent prendre de nombreuses formes. Certains des environnements dans lesquels Apollo déploie mes logiciels sont des environnements de développement ou de test traditionnels qui constituent un élément de base pour de nombreuses équipes de développeurs. D'autres environnements sont de véritables environnements de production, mais me permettent de déployer la même version de mon logiciel dans différentes régions de cloud, chez différents fournisseurs de services en cloud par exemple, la région Est des États-Unis pour Amazon et la région d'Europe de l'Ouest pour Azure. Et certains de ces environnements sont des environnements de production qui ont été mis en place pour chaque client, peut-être à l'intérieur de leur VPC ou de leur centre de données. En encodant toutes ces informations dans ma version dans le catalogue Apollo, je peux permettre à Apollo de gérer cette complexité pour moi. Cela me permet de récupérer du temps et d'envoyer des logiciels dans tous les environnements où ils sont nécessaires, de manière fiable et efficace.

• Palantir Apollo Security



Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

Comme nous le savons tous, dans le sillage de Log4j et SolarWinds, il est de plus en plus crucial pour les organisations de comprendre de manière globale leur chaîne d'approvisionnement en logiciels. Apollo m'offre une nomenclature logicielle facile à utiliser pour ma version, ce qui me donne une visibilité sur toutes mes dépendances logicielles. Je suis en mesure de filtrer rapidement et de comprendre quelles versions de Log4j ma version exploite. Dans ce cas, nous constatons que j'utilise la version sûre 2.17.2, il n'y a donc aucune action nécessaire. Apollo me donne une transparence à 360 degrés sur tous les logiciels que j'ai en production, quel que soit l'environnement. En plus de me permettre de comprendre la chaîne d'approvisionnement de mes logiciels, Apollo offre également une visibilité sur la sécurité de mes versions. En tant que développeur, il est extrêmement précieux pour moi de disposer de ces informations en plus de toutes les autres que nous venons d'examiner, afin de pouvoir vraiment comprendre mon logiciel à partir d'un seul et même écran. Apollo continuera à analyser ma version au fil du temps, en s'assurant que nous détectons toutes les nouvelles vulnérabilités qui seront découvertes à l'avenir et que nous pouvons rappeler automatiquement cette version ou me signaler qu'une vulnérabilité nécessite mon attention.

Pendant le temps que j'ai passé à expliquer comment mon équipe utilise Apollo, il a travaillé dur à la mise à niveau de mes environnements de production vers la version 1.23.1. Nous pouvons constater que le déploiement est maintenant terminé et qu'Apollo a mis à jour mon logiciel dans tous les environnements sur lesquels il est déployé.
Le monde dans lequel nous déployons les logiciels aujourd'hui est beaucoup plus dynamique qu'il y a quelques années, et ce dynamisme ne fait que croître. Apollo a été conçu pour gérer les logiciels à l'échelle et au rythme voulus, afin de livrer plus, plus vite et plus efficacement que jamais auparavant.

Disclaimer

This presentation and the accompanying oral commentary include discussion of Palantir products, features and capabilities, including recent updates to our products, as well as potential product direction. They are intended for information purposes only and shall not be deemed to be incorporated into any contract or agreement and do not constitute a guarantee or warranty of any kind. They are not a commitment to deliver any material, code, or functionality, and should not be relied upon in making procurement, purchasing or investment decisions. The development, release, and timing of any features, capability, or functionality mentioned herein remains at our sole discretion.

This presentation and the accompanying oral commentary contain "forward-looking" statements within the meaning of the federal securities laws, and these statements involve substantial risks and uncertainties. All statements other than statements of historical fact could be deemed forward-looking, including, but not limited to, expectations of future operating results or financial performance, market size and growth opportunities, plans for future operations, competitive position, technological capabilities, and strategic relationships, as well as assumptions relating to the foregoing. Forward-looking statements are inherently subject to risks and uncertainties, some of which cannot be predicted or quantified. In some cases, you can identify forward-looking statements by terminology such as "guidance," "expect," "anticipate," "should," "believe," "hope," "target," "project," "plan," "goals," "estimate," "potential," "predict," "may," "will," "might," "could," "intend," "shall," and variations of these terms or the negative of these terms and similar expressions. You should not put undue reliance on any forward-looking statements. Forward-looking statements should not be read as a guarantee of future performance or results and will not necessarily be accurate indications of the times at, or by, which such performance or results will be achieved, if at all.

Disclaimer

Forward-looking statements are subject to a number of risks and uncertainties, many of which involve factors or circumstances that are beyond our control. Our actual results could differ materially from those stated or implied in forward-looking statements due to a number of factors, including but not limited to risks detailed in our filings with the Securities and Exchange Commission (the "SEC"), including in our quarterly report on Form 10-Q for the quarter ended September 30, 2020 and other filings and reports that we may file from time to time with the SEC. You can locate these reports on our investor relations website (https://investors.palantir.com/financials/sec-filings/) or on the SEC website (https://www.sec.gov). If the risks or uncertainti-es ever materialize or the assumptions prove incorrect, our results may differ materially from those expressed or implied by such forward-looking statements. Except as required by law, we assume no obligation and do not intend to update these forward-looking statements or to conform these statements to actual results or to changes in our expectations.

This presentation contains statistical data, esti-mates and forecasts that are based on independent industry publications or other publicly available information, as well as other information based on our internal sources. This information involves many assumptions and limitations, and you are cautioned not to give undue weight to these estimates. We have not independently verified the accuracy or completeness of the data contained in these industry publications and other publicly available information. Accordingly, we make no representations as to the accuracy or completeness of that data nor do we undertake to update such data after the date of this presentation. All data shown in product demonstrations is notional or publicly available and any resemblance to actual persons, entities or events is purely coincidental and should not be inferred. Certain visualizations and capabilities shown in product demonstrations may rely on or reflect third party data sources that are not included as part of Palantir's standard product offering.

Disclaimer

This presentation also contains links to publicly available websites, data, or other information. We have not independently verified the accuracy or completeness of such websites, data, or information and accordingly we make no representations as to their accuracy or completeness nor do we undertake to update such data or information after the date of this presentation. The inclusion of external links does not constitute endorsement by Palantir of the linked websites or the data or information contained therein.

By attending or receiving this presentation you acknowledge that you will be solely responsible for your own assessment of the market and our market position and that you will conduct your own analysis and be solely responsible for forming your own view of the potential future performance of our business.

Unless otherwise noted, all product, feature, or service names, logos, and trademarks, including without limitation Palantir and the Palantir logo are the intellectual property of Palantir and / or its affiliates in the United States and/or other jurisdictions. Any non-Palantir logos or trademarks included herein are the property of the owners thereof and are used for reference purposes only. Such use should not be construed as an endorsement of Palantir or the platforms and products of Palantir.