Palantir

APOLLO DEMO-TAG

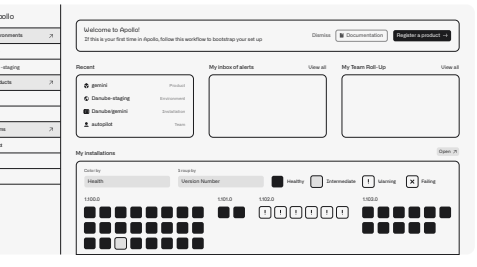
April 27, 2022

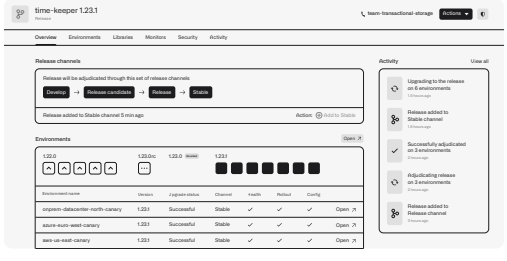
→ German Transcript

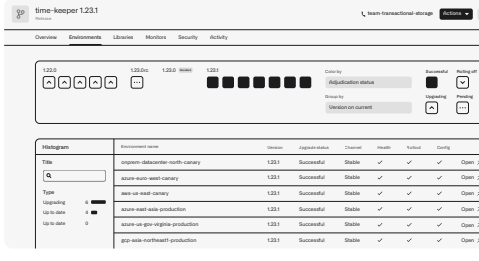
This transcript is provided as a courtesy and is intended to be viewed, and is subject to, the accompanying oral presentation and related materials, including any legal disclaimers.

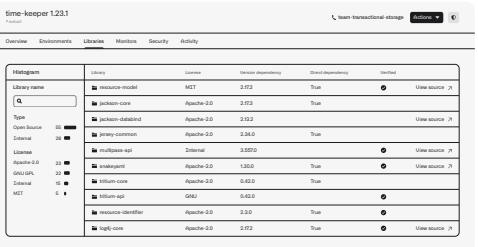
Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

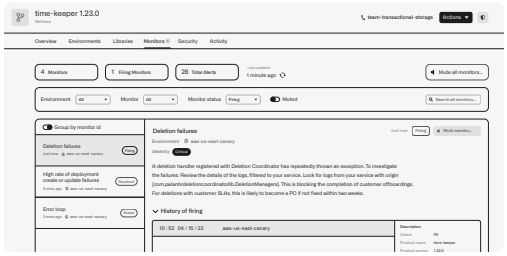
Index

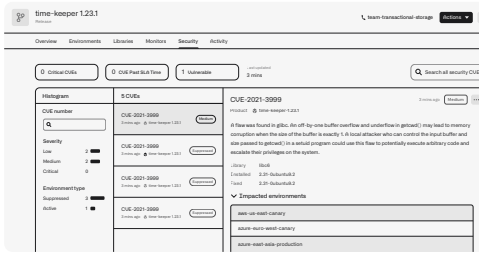












Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

Why We Built Apollo

Shyam Sankar
→ Chief Operating Officer

Bei Palantir sprechen wir immer davon, an den schwierigsten Problemen der Welt zu arbeiten. Und mit den schwierigsten meinen wir nicht die kompliziertesten oder anspruchvollsten oder technisch komplexesten. Wir meinen die existenziellsten. Wir arbeiten seit sechs Jahren an Apollo, weil wir der Meinung sind, dass die Bereitstellung der Software eines dieser existenziellen Probleme ist. Und ehrlich gesagt bin ich mir nicht sicher, ob irgendjemand anderes daran gedacht hat. Dann passierte SolarWinds, dann Log4j. Und es ist erschreckend deutlich, dass die Softwarebereitstellung und -verwaltung — d. h. die Softwarelieferkette — ein existenzielles Problem für moderne Unternehmen ist.

Wir können den Nutzen und die Wirksamkeit von Apollo schlichtweg dadurch bezeugen, dass wir es selbst zur Bereitstellung unserer eigenen Software verwenden. Apollo verwaltet, implementiert und wartet Foundry und Gotham weltweit und unterstützt geschäftskritische Arbeiten in einigen der schwierigsten Umgebungen, die man sich vorstellen kann. Von Satelliten über U-Boote, High-Side-Netzwerke bis hin zu Humvees und natürlich jeder Art von Cloud- und lokalen Umgebungen — Apollo verwaltet Software in mehr als 250 verschiedenen Kundenumgebungen. Dies ermöglicht es uns, den Anforderungen unserer Zeit gerecht zu werden.

Diese Fragmentierung der Bereitstellungsumgebungen verändert die Spielregeln für Softwareanbieter grundlegend. Apollo löst dieses Problem. Apollo ermöglicht es Ingenieur*innen, Code einmal zu schreiben, der in allen Umgebungen funktioniert. Es ermöglicht derselben Software, die individuellen Anforderungen jeder Umgebung zu erfüllen, indem es Ingenieur*innen ermöglicht, ihre eigenen Voraussetzungen für das erwartete Verhalten der Umgebung zu kodieren, sei es vor Ort, in der Cloud oder am Netzwerkrand, sodass sich Ingenieur*innen darauf konzentrieren können, bessere Produkte zu entwickeln und all die menschlichen Mühen zu vermeiden, um die Produkte dahin zu bringen, wo sie benötigt werden. Apollo war das Rückgrat der Bereitstellung unsere Software in den letzten sechs Jahren und das wird in den nächsten 60 Jahren weiter der Fall sein. Ich freue mich sehr darüber, das volle Potential der Plattform heute mit Ihnen teilen zu dürfen.

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

Wissen Sie, Microservices und serviceorientierte Architekturen sind so beliebt geworden, weil sie schnellere Innovation durch Arbeitsteilung versprechen. Damit Sie diese Vorteile tatsächlich nutzen können, müssen diese Teams in der Lage sein, sich mit unterschiedlichen Geschwindigkeiten weiterzuentwickeln. Teams sollten in der Lage sein, neue Funktionen und Fähigkeiten dann bereitzustellen, wenn sie bereit sind, anstatt dadurch verlangsamt zu werden auf einen wöchentlichen, monatlichen oder vierteljährlichen Veröffentlichungsplan warten zu müssen, der von dem langsamsten Beteiligten oder der Fähigkeit, die Qualität der Platform zu überprüfen, bestimmt wird. In einer serviceorientierten Architektur handelt es sich lediglich um API-Abhängigkeiten zwischen den verschiedenen Diensten, aber man möchte, dass diese in der Lage sind, Neuerscheinungen zu unterschiedlichen Frequenzen zu senden.

Apollo wurde von Anfang an entwickelt, diese intrinsischen Probleme, die mit verteilten Systemen einhergehen, zu lösen, indem es den Entwickler-Teams ermöglicht, die Vorbedingungen und die Erwartungen an ihre Software neben dem Code selbst zu kodieren. Apollo übernimmt dann diese eingebetteten Einschränkungen und stellt sicher, dass sie beim Upgrade eingehalten werden, und ermöglicht es, das Rollback eines bestimmten Dienstes sicher zu automatisieren, ohne die Verfügbarkeit der gesamten Plattform zu beeinträchtigen. Zur Kompilierungszeit können Speichersysteme so die Reihe von Schemas deklarieren, aus denen sie lesen und in denen sie schreiben können, spezifische Migrationsschritte, die sie unterstützen, und eine Möglichkeit, die aktuelle Version ihrer Schemas zur Laufzeit zu melden. Auf dies Weise kann Apollo sicher Speicherdienste durch Schemamigrationen aktualisieren und diese Schritte bei Bedarf durch Schutzmechanismen wieder rückgängig machen, die Datenbeschädigungen verhindern.

Das zweite Prinzip: Ein*e Softwareentwickler*in sollte kein tiefgreifendes Verständnis der Umgebung oder des Infrastruktursystems benötigen, in dem seine*ihre Software ausgeführt wird, um Funktionen zu entwickeln, die erstklassige Software unterstützen. Als die Welt Software-Monolithen baute und auslieferte und in ein oder zwei virtuellen Maschinen oder Server einsetzte, war das schwierigste Bereitstellungsproblem für die Entwickler*innen, Linux zu verstehen. Aber heute ist die Infrastruktur, auf der Software aufgebaut wird, weitaus komplizierter, und wird immer komplizierter. Vor diesem Hintergrund ermöglichen die heute verfügbaren

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

Infrastruktursysteme die Bereitstellung der nicht verhandelbaren Funktionen moderner Software. Dinge wie Datenverschlüsselung und Netzwerksicherheit, Software, die hoch verfügbar ist und ohne Ausfallzeiten der Benutzenden aktualisiert werden kann, und die Möglichkeit, sie über mehrere Cloud-Anbieter bis zum Netzwerkrand bereitzustellen. All dies wird für geschäftskritische Software immer wichtiger. Damit Entwickler*innen jedoch eine moderne Java- oder Python-Anwendung erstellen und bereitstellen können, die hoch verfügbar sein muss — vielleicht wird eine Konfiguration und Zertifikate benötigt, um HTTPS-Datenverkehr zu bedienen und durch ein Front-Door-Proxy verfügbar zu machen — müssten sie sechs oder sieben verschiedene Kubernetes-Objekte verstehen, konfigurieren und warten, zusätzlich zu den Cloud-Infrastrukturkomponenten, die diese Netzwerk-Load-Balancer verfügbar machen würden. Das ist einfach keine gute Verwendung der Zeit der meisten Entwickler*innen. Daher bietet Apollo ein SDK, mit dem Entwickler*innen moderne Softwareanwendungen erstellen können. Das SDK von Apollo ermöglicht es Entwickler*innen, die Eigenschaften ihrer Software zu deklarieren, z. B.: „Vielleicht brauche ich ein lokales persistentes Volume oder ich brauche drei Repliken meines Dienstes, und das Quorum für meinen Dienst besteht darin, dass ein beliebiges Replikat verfügbar ist." Apollo kann dann basierend auf diesen deklarierten Eigenschaften entsprechenden Kubernetes-Pod-Controller generieren und verwalten. Ein Zertifikat und die entsprechenden öffentlichen und privaten Schlüsseln werden generiert und dem Dienst zur Verfügung gestellt, sodass der Dienst die TLS-Verschlüsselung für jeden Gastdatenverkehr konfigurieren kann, ohne verstehen zu müssen, wie PKI-Material von der Plattform erstellt oder verwaltet wird. Und der*die Entwickler*in kann deklarieren, dass sein Dienst einen API-Endpunkt erzeugt, und anfordern, dass dieser Endpunkt über einen Reverse-Proxy auf einem bestimmten Pfad verfügbar gemacht wird, ohne die Komplexität der Einrichtung des Kubernetes-Cluster-Ingress-Systems oder der Funktionsweise von Ingress-Objekten verstehen zu müssen. All diese Funktionen sind möglich, ohne eine einzige Zeile Code zu ändern.

Das dritte Prinzip besteht darin, Software dort einzusetzen, wo sie gebraucht wird, indem Softwareplattformen wiederholbar, reproduzierbar und über Cloud-Anbieter und Infrastrukturtypen hinweg portierbar sind. Beispielsweise finden viele Unternehmen die Idee, ihre Software in einer neuen Umgebung bereitzustellen, möglicherweise abschreckend, da es fast unmöglich ist, ihre Plattform von Grund auf neu einzurichten,

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

geschweige denn mehrere Installationen gleichzeitig zu betreiben. Andere verfügen über Bereitstellungspipelines, die stark auf eine einzelne Umgebung zugeschnitten sind, in der die Einführung einer neuen Produktionsumgebung ihren bestehenden Bereitstellungsprozess grundlegend unterbrechen würde. Der Marktzugang ist für viele einfach kein erreichbares Ziel, da die zur Bereitstellung der Software verwendeten Technologien und Tools, für das Paradigma einer einzelnen SaaS-Umgebung entwickelt wurden. Und während sich die Welt weiterentwickelt hat, haben sich ihre Werkzeuge nicht weiterentwickelt. Daher verfolgt Apollo einen ganz anderen Ansatz als andere Tools im DevOps- oder Plattform-Engineering-Bereich. Apollo beginnt mit der Modellierung der Software, die Sie in Ihren Umgebungen und Ihrem Unternehmen ausführen möchten, in einem einzigen Softwarekatalog. Auf diese Weise können alle Parteien eine einzige Ansicht von allem teilen, was über eine Software bekannt ist, von den Angaben der Entwickler*innen — Dinge wie dienstübergreifende Abhängigkeiten, unterstützte Schemaversionen, Ressourcenanforderungen usw. — bis hin zu Informationen, die durch externe Systeme wie Viren oder Schwachstellenscanner oder die Sicherheitsscanner, die Sie bereits in Ihrem Unternehmen einsetzen. Verschiedene Softwareplattformen können dann als Zusammensetzung der einzelnen Dienstleistungen und Produkte definiert werden, die im Apollo-Katalog vorhanden sind. Apollo modelliert dann jede Umgebung, in der Sie Software bereitstellen und verwalten möchten, und deklariert eine Reihe von Einheiten als eindeutige Installationen eines bestimmten Softwareprodukts aus dem Katalog. Release-Kanäle werden verwendet, um die verschiedenen Phasen darzustellen, die die Software bei der Einführung in Ihren Umgebungen durchläuft. Mit Release-Kanälen können Sie nichtlineare Rollout-Prozesse modellieren. Für Geschäftsmodelle, die eher denen von Palantir entsprechen, müssen Sie eine große Anzahl von Umgebungen verwalten und Ihre Software dort einsetzen, wo Ihre Kunden sind. In diesem Fall können Sie freigegebene Kanäle verwenden, um das vertrauen in neue Versionen zu modellieren, wo Sie Ihre Software zuerst in Test-Umgebungen anwenden, bevor Sie sie in mehr geschäftskritischen Umgebungen einsetzen, in denen eine geringere Risikotoleranz besteht. Dieser Ansatz bei der Software-Einführung vermeidet die schmerzhaftesten Fehlermodi herkömmlicher Pipeline-gesteuerter Continous-Deployment-Modelle. Wo Pipelines in der Regel spezifisch auf einzelne oder kleinere Sets an Services zugeschnitten sind und sich in Silos entlang der Teamgrenzen entwickeln, bieten Release-Kanäle einen konsistenten Ansatz für Ihre Produktionsmanagement-Story über Teams

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

The Core Principles that Unlock Autonomous Software Deployment

Greg DeArment,
→ Chief Architect &
Head of Apollo Platform

[CONT.]

hinweg, während jedes Entwicklerteam immer noch die Eingaben in das System steuern kann, die vorschreiben, wie die Software eingeführt wird.

Und das letzte Prinzip: Die effektive Verwaltung von Softwareproduktion geht über die einfache Bereitstellung einer Software hinaus und hängt von der aktiven Zusammenarbeit von Softwareentwicklern*innen, Betreiber*innen, Sicherheits- und Compliance-Teams ab. Wenn wir in den letzten Jahren eines gelernt haben, dann ist es, dass die Herausforderung der Gewährleistung der Sicherheit der Softwarelieferkette und die damit verbundenen Probleme nicht verschwinden. So wie nicht jede*r Ingenieur*in in Details der Infrastruktur verstehen muss, auf der seine*ihre Software ausgeführt wird, sollten sie auch nicht Zeit damit verbringen müssen, Expert*innen für die Funktionsweise von Softwaresicherheit zu werden. Es sollte ein integrierter Bestandteil der Produktionsmanagement-Story und frei verfügbar sein. Mit Apollo wird der Softwarekatalog zu einem zentralen Gateway, das auf der Grundlage der von den Sicherheits- und Compliance-Teams festgelegten Richtlinien regelt, welche Software in der Produktion eingesetzt werden kann und welche nicht. Ohne den natürlichen Softwareentwicklungsprozess zu beeinträchtigen, lässt sich Apollo in Ihre bestehende DevOps-Toolchain integrieren, z. B. in Ihr kontinuierliches Integrationssystem, oder Artefakt-Container-Registrierungen und Ihre Schwachstellen- und Sicherheitsscanner. Ihre Sicherheitsteams können Richtlinien für Sicherheitslücken definieren, die es Apollo ermöglichen, Versionen, die gegen diese Richtlinien verstoßen, automatisch zurückzurufen, oder das Sicherheitsteam zu benachrichtigen, wenn es Versionen gibt, die nicht den Richtlinien entsprechen, damit sie umgehend jährlich geprüft werden.

Nach der Softwarebereitstellung überprüft Apollo kontinuierlich den Zustand jeder Softwareinstallation, wenn sich das Entwicklungsteam für das Health-System entschieden hat. Auf diese Weise können Entwickler*innen SLOs und andere Kriterien codieren, die ihre Software erfüllen sollte, damit sie als fehlerfrei angesehen werden kann. Darüber hinaus können sie auch innerhalb des Software-Artefakts selbst angeben, wie ihre Software überwacht werden soll, wodurch es möglich wird, das erwartete Verhalten zu definieren, ohne den Code ändern zu müssen. Apollo fasst diese Informationen in einem einzigen Fenster zusammen und kann das Rollback und andere aktive und passive Korrekturmaßnahmen automatisieren, wenn die SLOs verletzt werden, Zustandsprüfungen Probleme melden oder wenn die eingebetteten Monitore ausgelöst werden.

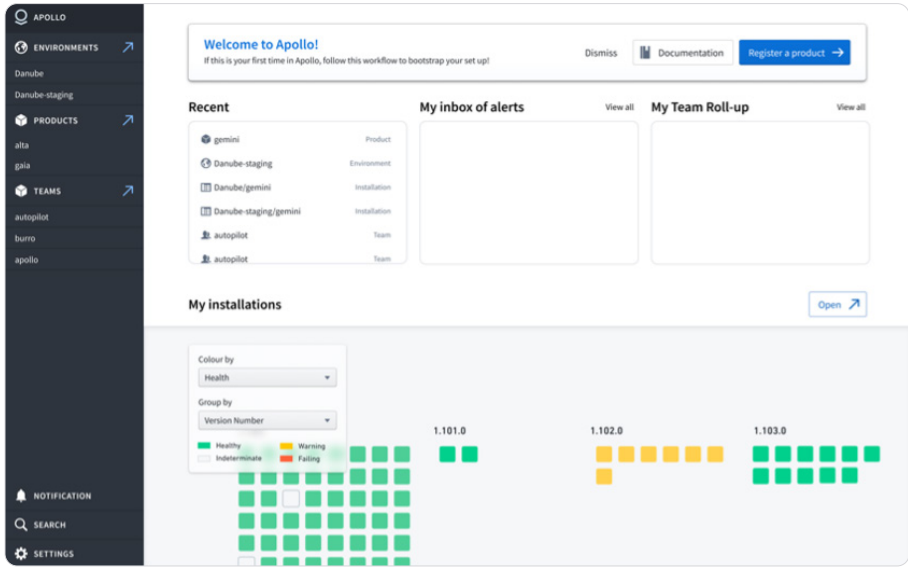
Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

Lassen Sie uns in eine Demo einsteigen, um zu verstehen, wie jede dieser Gruppen Apollo über den Lebenszyklus einer Version nutzt. Stellen wir uns vor, ich bin ein Entwickler, der Apollo nutzt, um meine Software kontinuierlich in allen meinen Umgebungen bereitzustellen. Ich habe gerade eine Version des Timekeeper-Produkts auf Version 1.23.0 veröffentlicht. Ich kann sofort erkennen, dass mein Version gemäß den Bereitstellungsrichtlinien meines Teams für alle gekennzeichneten Software-Releases dem entsprechenden Release-Kanal hinzugefügt wurde. Infolgedessen hat Apollo damit begonnen, meine drei Umgebungen, die diesen Release-Kanal abonnieren, automatisch für Updates des Timekeeper-Dienstes zu aktualisieren.

• Palantir Apollo Control Center



Lassen Sie mich Ihnen die Release-Seite meines Produkts zeigen, auf der ich alles über diese Softwareversion von einem einzigen Ort verstehen kann. Von hier aus kann ich sehen, dass meine aktuelle Version gegen meiner definierten Test-Umgebungen beurteilt und automatisch als stabil markiert wird, sobald das Rollout abgeschlossen ist und alle Installationen während des angegebenen „Soak"-Zeitraums fehlerfrei bleiben. Wie Sie sehen können, hat sich mein Team für Test-Umgebungen entschieden, die mehrere verschiedene Cloud-Anbieter sowie ein Rechenzentrum umfassen. Es sieht so aus, als sei einer meiner Monitore ausgelöst worden, was wahrscheinlich bedeutet, dass mein Upgrade in einer oder mehreren meiner Test-Umgebungen fehlgeschlagen ist. Wenn dies passiert, gehe ich als Erstes in meine

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

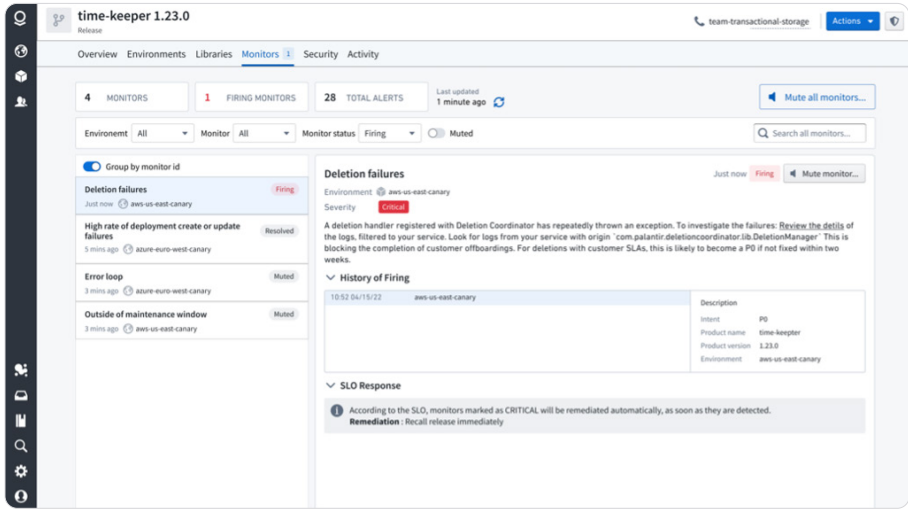
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

Registerkarte „Monitors“ Meiner Version, um besser zu verstehen, welche Benachrichtigungen ausgelöst werden und was das für meine Veröffentlichung bedeutet. Monitore ermöglichen es mir, service level objectives für mein Produkt in Zustandsprüfungen kodieren, die Apollo durchführen kann. Wie wir hier sehen können, sieht es so aus, als ob einer der kritischen Monitore, die ich in meinem Produkt kodiert habe, fehlgeschlagen ist. Das Auslösen eines kritischen Monitors ist in der Regel ein Zeichen dafür, dass bei meinem Release etwas ernsthaft schiefgelaufen ist und der Release nicht stabil genug ist, um in der gesamten Flotte eingeführt zu werden. In diesem Fall war ich derjenige, der den Monitor beim Auslösen erwischt hat, aber oft ist es ein Mitglied eines unserer Betriebsteams, das für die Verwaltung dieser Umgebungen verantwortlich ist. Da ich meine SLOs innerhalb des Monitors codieren und von Apollo automatisch verfolgen lassen konnte, sind meine Kollegen im Betriebsteam genauso in der Lage wie ich, Warnmeldungen zu sichten und entsprechende Maßnahmen zu ergreifen, um sicherzustellen, dass meine Software für Kunden in der Produktion zuverlässig einen Mehrwert liefert.

• Palantir Apollo Monitoring



Während wir uns unterhielten, hat Apollo meine Veröffentlichung zurückgerufen, und nicht nur sicherzustellen, dass Timekeeper 1.23.0 nicht in keiner neuen Umgebungen installiert wird, sondern auch ein automatisches Downgrade für die bereits aktualisierten Test-Umgebungen gestartet. Wenn ich zurück zur Übersichtsseite für meine Version 1.23.0 navigiere, kann ich sehen, dass Apollo mich

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

jetzt an prominenter Stelle darüber informiert, dass dieser Release zurückgerufen wurde, den Grund für den Rückruf, sowie die Strategie, die Apollo verfolgte, um zu einer bewährten Version zurückzukehren. All diese Informationen an einem einzigen Ort zu haben, bietet für mich und mein Team einen entscheidenden historischen Kontext. Wie Sie sehen können, wurde das Aktivitätsfenster aktualisiert, da Apollo automatische Maßnahmen ergriffen hat, um meinen auslösenden Monitor zu erkennen, die Version abzurufen und mit dem Rollback auf die letzte funktionierende Version meines Produkts zu beginnen. Die Fähigkeit von Apollo, alle Aktionen in meinen verwalteten Umgebungen proaktiv zu verfolgen und zu prüfen, ermöglicht es den von mir entwickelten Produkten, in Umgebungen ausgeführt zu werden, die strenge Sicherheits- und Compliance-Kontrollen erfordern.

Wie wir in dem „Environment"-Panel sehen können, hat Apollo gerade das Rollback meiner Test-Umgebungen von meiner schlechten Version 1.23.0 abgeschlossen. Da Apollo in der Lage war, automatisch mit dem Rollout meiner Version in Test-Umgebungen zu beginnen, ein Problem mit meiner Version zu erkennen, bevor sie kritische Produktionsumgebungen erreichte, und Diese Version dann sofort zurückzurufen und rückgängig zu mahen, habe ich jetzt genügend Zeit, um Fehler zu beheben, und mich darauf zu konzentrieren mein Produkt verbessern. Ich habe das Problem behoben und eine neue Version von Timekeeper 1.23.1 erstellet. Diese Version wurde erfolgreich in allen drei meiner Test-Umgebungen aktualisiert und hat für meine definierte „Soak"-Zeit keine Monitore ausgelöst. Apollo erhält daher automatisch Version 1.23.1 und wird als stabil markiert. In meinem Team ist dies der Release-Kanal, den wir verwenden, um zu signalisieren, dass ein Veröffentlichung produktionsbereit ist, und Apollo hat automatisch mit dem Rollout in der gesamten Flotte begonnen.

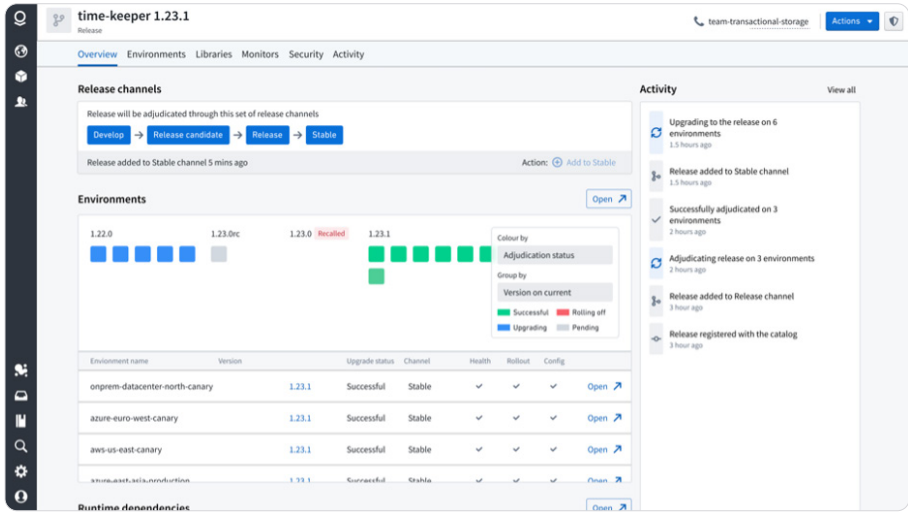
Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

• Palantir Apollo Software Catalog



Neben den Release-Kanälen haben Sie bereits gesehen, wie ich die „Environments"-Ansicht von Apollo nutzte, um den Status meiner Releases in allen Umgebungen schnell nachzuvollziehen. Umgebungen in Apollo sind ein kritisches Konzept und können viele Formen annehmen. Einige der Umgebungen, in dennen Apollo meine Software einsetzt, sind traditionelle Entwicklungs- oder Testumgebungen, die Teil der Grundausstattung für viele Entwicklerteams sind. Andere Umgebungen sind echte Produktionsumgebungen, und erlauben mir, dieselbe Version meiner Software über verschiedene Cloud-Regionen und Cloud-Anbietern hinweg bereitzustellen — zum Beispeil in der Region USA Ost von Amazon und die Region Europa West von Azure — und einige der Umgebungen sind Produktionsumgebungen von Kunden, vielleicht innerhalb ihres VPC oder Rechenzentrums. Indem ich all diese Informationen innerhalb meiner Veröffentlichung im Apollo-Katalog kodiere, kann ich Apollo erlauben, diese Komplexität für mich zu verwalten. Das gibt mir Zeit zurück und ermöglicht es mir, Software zuverlässig und effizient an jede Umgebung zu liefern, in der sie benötigt wird.

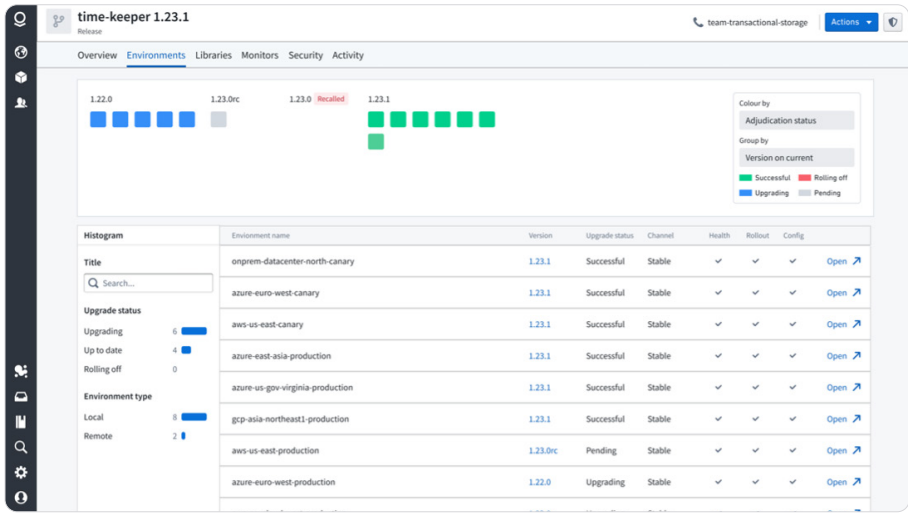
Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

Apollo in Action: Software Demo

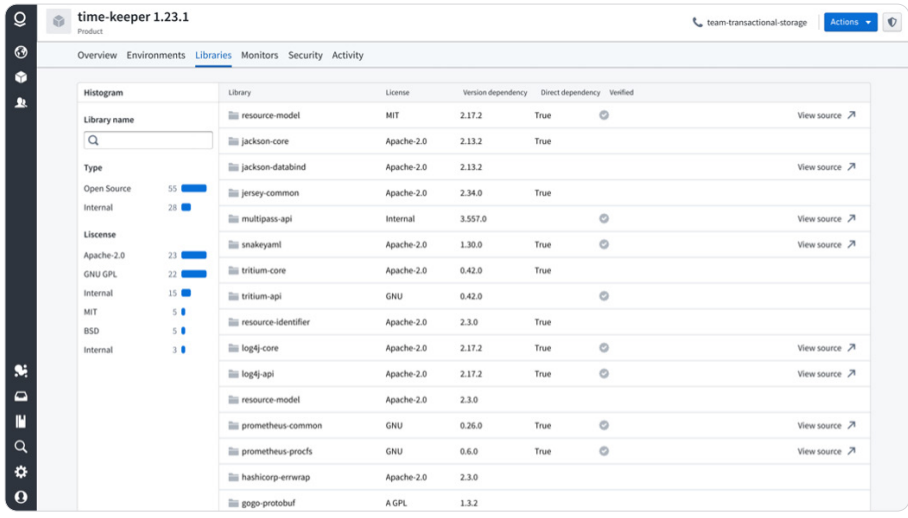
Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

• Palantir Apollo Delivery



• Palantir Apollo Software Supply Chain



Wie wir alle wissen, wird es im Anbetracht von Log4j und SolarWinds für Unternehmen immer wichtiger, ihre Software-Lieferkette ganzheitlich zu verstehen. Apollo stellt mir eine benutzerfreundliche Software-Stückliste für meine Version zur Verfügung, die mir Einblick in alle meine Software-Abhängigkeiten gibt. Ich kann schnell herausfiltern und verstehen, welche Versionen von Log4j mein Release nutzt. In diesem Fall sehen wir, dass ich die sichere Version 2.17.2 verwende, sodass keine Handlung erforderlich ist. Apollo gibt mir 360-Grad-Transparenz über alle Software, die ich in der Produktion habe, unabhängig von der Umgebung. Apollo ermöglicht mir nicht nur, die Lieferkette meiner Software zu verstehen, sondern bietet auch Einblick in die Sicherheit

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

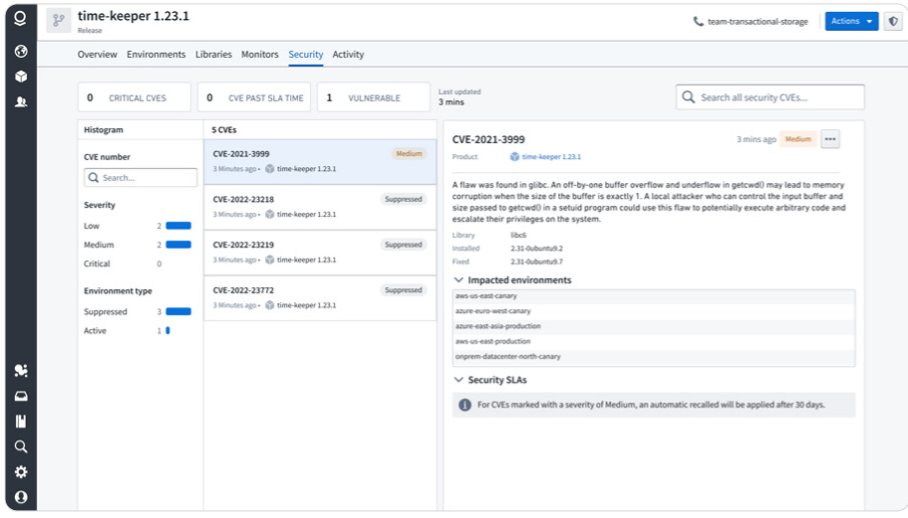
Apollo in Action: Software Demo

Sean Hacker,
→ Forward Deployed Engineer

[CONT.]

meiner Versionen. Als Entwickler*in ist es für mich unglaublich wertvoll, diese Informationen zusammen mit all den anderen Informationen zu haben, die wir uns gerade angesehen haben, damit ich meine Software von einem einzigen Fenster aus verstehen kann. Apollo wird meine Version im Laufe der Zeit weiterhin scannen um sicherzustellen, dass wir alle neuen Sicherhetistlücken erkennen, die in Zukunft entdeckt werden, und diese Version automatisch zurückrufen oder mich benachrichtigen, dass eine Schwachstelle meine Aufmerksamkeit erfordert.

• Palantir Apollo Security



In der Zeit, in der ich Sie durch die Nutzung von Apollo durn mein Team geführt habe, hat Apollo hart daran gearbeitet, meine Produktionsumgebungen auf meine Timekeeper-Version 1.23.1 zu aktualisieren. Wir können sehen, dass Rollout jetzt abgeschlossen ist und Apollo meine Software in allen Umgebungen, in denen sie bereitgestellt wird, aktualisiert hat. Die Welt, in der wir heute Software einsetzen, ist viel dynamischer als noch vor wenigen Jahren, und diese Dynamik nimmt weiter zu. Apollo wurde entwickelt, um Software in großem Maßstab und Tempo zu verwalten und so Ergebnisse weiter, schneller und effizienter zu liefern als je zuvor.

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

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.

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

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.

Copyright © 2022 Palantir Technologies Inc. and / or affiliates. All rights reserved. All data is notional or publicly available.

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.

Copyright © 2022 Palantir Technologies Inc. and / or affiliates ("Palantir"). All rights reserved.