



Artefakt-Management: **Best Practices für Skalierbarkeit und Wachstum**

Copyright © 2024 JFrog Ltd.
Publish: December 2023



Inhaltsverzeichnis

Einführung	2
Was ist Artefakt-Management?	3
Benefits und Unternehmenswert von Artefakt-Management	4
Glossar	5
Best Practices für Artefakt-Management	6-15
1. Pakete und Artefakte in einem Tool speichern, das für diesen speziellen Dateityp konzipiert ist	6
2. Trennung von Artefakten und Abhängigkeiten mit einer lokalen, remote und virtuellen Repository-Struktur	8
3. Öffentliche Proxy-Registries für ein lokal zwischengespeichertes Set von Artefakten	9
4. Repositories für jede Entwicklungsphase	10
5. Weisen Sie jedem Team ein eigenes Set von lokalen, remote und virtuellen Repositories zu	11
6. Artefakte zwischen SDLC-Umgebungen übertragen – statt sie neu zu generieren	12
7. Speichern und Verwalten der Metadaten zusammen mit dem dazugehörigen Artefakt	13
8. Erstellung von Richtlinien, die die Zugriffsrechte auf Repositories und das Ausführen von Aktionen regeln	14
9. Definition von Cleanup-Richtlinien für Repository-Hygiene und - Performance	15
Auswahl der richtigen Artefakt-Management-Lösung	16
JFrog für Artefakt-Management	19
Fazit	20

Einführung

Artefakte (auch bekannt als Binärdateien, Pakete, Librarys, Komponenten usw.) sind die Bausteine und Nebenprodukte der Softwareentwicklung. Jeden Tag erstellen Development-Teams neue Artefakte, indem sie eine Vielzahl von Technologien verwenden und bestehende Pakete, sowohl von Erst- als auch von Drittanbietern, in ihren Prozess integrieren.

Aufgrund des beschleunigten Development-Tempos, der Microservice-Ansätze, der Open-Source-Bewegung und weltweit verteilter Teams müssen Unternehmen eine explosionsartige Zunahme von Artefakten verwalten.

Leider führt die Verwaltung all dieser Artefakte zu einer Komplexität, die den Fortschritt des Developments verlangsamen oder blockieren kann. Verteilte Development-Teams müssen beispielsweise konsistente Paketversionen verwenden, um sicherzustellen, dass ihre Microservice-Apps kompatibel sind. Und Open-Source-Librarys bringen potenzielle Unsicherheitsfaktoren in die Software ein, darunter Schwachstellen sowie Versions- und Compliance-Probleme.

Solche Herausforderungen nehmen mit dem Wachstum von Unternehmen zu. Daher ist es von entscheidender Bedeutung, dass Unternehmen über Richtlinien, Prozesse und Tools verfügen, die eine unternehmensweite Transparenz und Kontrolle über die Artefakte, Binärdateien und Librarys ermöglichen, die Teil der Software-Lieferkette sind.

Dieses White Paper stellt die Konzepte und Gründe für die Einführung einer Artefakt-Management-Strategie vor, einschließlich 9 Best Practices für das Artefakt-Management. Diese bewährten Methoden wurden über mehr als 15 Jahre verfeinert, in denen wir Tausenden von großen und kleinen Unternehmen geholfen haben, die Verwendung von Artefakten in der Software-Development-Pipeline zu kontrollieren und gleichzeitig die Softwarebereitstellung zu beschleunigen. Wir werden auch darauf eingehen, worauf Sie bei den Tools achten sollten, die Sie zur Implementierung und Operationalisierung Ihres Artefakt-Management-Ansatzes verwenden.

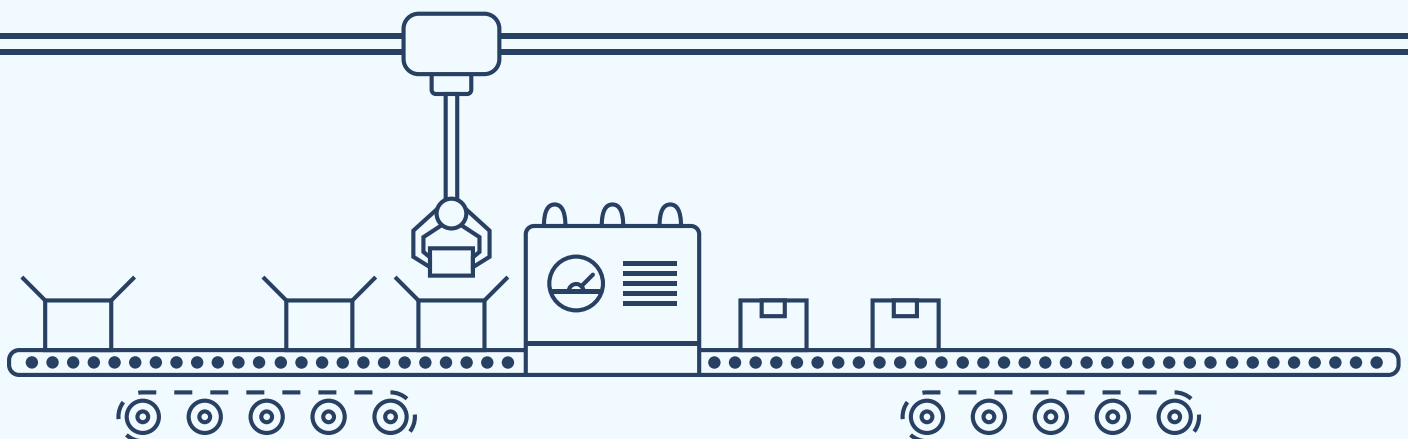
Was ist Artefakt-Management?

Bevor wir uns mit dem Management von Artefakten beschäftigen, muss klar sein, was ein Artefakt überhaupt ist. Ein Artefakt ist ein Software-Asset, das mit einem Software-Development-Projekt verbunden oder Teil davon ist. Artefakte liegen normalerweise in Form von Binärpaketen vor und werden verwendet, um die Funktion, die Architektur und das Design der Anwendung zu charakterisieren oder zu beschreiben. Typische Arten von Artefakten sind Images, ausführbare Dateien, Datenmodelle, Librarys, Metadaten, Use Cases und natürlich die kompilierte Anwendungsbinärdatei.

Unter Artefakt-Management versteht man die organisierte und strukturierte Speicherung der Artefakte eines Softwareprojekts, während gleichzeitig definiert und kontrolliert wird, wie und wo diese Artefakte von den verschiedenen an dem Projekt Beteiligten verwendet werden können. Artefakt-Management stellt sicher, dass alle Artefakte konsistent verwaltet, versioniert und teamübergreifend – manchmal auch über mehrere Standorte hinweg – bereitgestellt werden,

um Qualität, Zuverlässigkeit und Nachvollziehbarkeit zu gewährleisten.

Die naheliegendste Lösung für die Implementierung einer Artefakt-Management-Strategie ist, wenig überraschend, ein Artefakt-Repository-Manager. Dieser wird auch als Binär-Repository-Manager bezeichnet. Andere alternative Lösungen sind ein gemeinsam genutztes Laufwerk, allgemeine Speicherlösungen oder ein Tool zur Quellcodeverwaltung, die jedoch alle ihre Grenzen haben. Ein gemeinsam genutztes Laufwerk bietet beispielsweise nur eine eingeschränkte Versionskontrolle und keine Möglichkeit zur Bereitstellung von Artefakten. Ein Tool zur Quellcodeverwaltung wiederum eignet sich eigentlich nur für die Verwaltung von Quellcode-Textdateien und nicht für komplexe Artefakte wie große Binärdateien oder Docker-Images. Schmökern Sie weiter oder überfliegen Sie die nächsten Seiten, um zu erfahren, worauf Sie bei der Implementierung von Artefakt-Management achten sollten.



Benefits und Unternehmenswert von Artefakt-Management

Eine richtig umgesetzte Artefakt-Management-Strategie ermöglicht es einem Unternehmen, die Art und Weise zu kontrollieren, wie Artefakte in die Software-Lieferkette gelangen, sich weiterentwickeln und verwendet werden – von der Entwicklung bis zur Bereitstellung. Durch den Einsatz einer Artefakt-Management-Lösung sind Unternehmen in der Lage, die Wechselbeziehungen aller ihrer Binärdateien zu verstehen und erhalten so einen zuverlässigen Einblick in die Verwendung und Rückverfolgbarkeit von Binärdateien sowie die Möglichkeit, automatische Regeln oder Richtlinien auf der Grundlage von binären Metadaten zu erstellen. Darüber hinaus bietet das Artefakt-Management Development-Teams die folgenden vier konkreten Benefits:



Erhöhte Geschwindigkeit

Ein besseres Management von Abhängigkeiten und Build-Artefakten verkürzt die Build-Zeit und sorgt für Versionskonsistenz zwischen den Teams.



Verbesserte CI/CD Stabilität

Das Einrichten eines zentralen Hubs mit klaren Regeln für den Empfang und die Bereitstellung des Outputs Ihrer Development-Pipelines, einschließlich der Metadaten, trägt zur Verbesserung der Zuverlässigkeit von vorhersehbaren und wiederholbaren Ergebnissen bei.



Konsistente Sicherheit und Compliance

Wenn Sie wissen, wo und wie all Ihre Artefakte gespeichert und verwaltet werden, können Sie Sicherheitsrichtlinien und -tools effektiver anwenden.



Skalierbarkeit

Die Implementierung von Artefakt-Management ermöglicht Unternehmen eine einfache Skalierung des Developments an mehreren Standorten durch bessere Versionierung, Kontrolle und (mit den richtigen Tools) die Möglichkeit, Artefakte konsistent an mehreren Entwicklungsstandorten und Nutzungspunkten bereitzustellen.

Glossar

Im Folgenden sind einige gängige Begriffe zum Thema Artefakt-Management aufgeführt, die all jenen, die keine erfahrenen DevOps-Profis sind, möglicherweise nicht geläufig sind.

Binärdatei

Alle Anwendungen beginnen als von Menschen geschriebener Code, werden aber schließlich zu einer Binärdatei zusammengesetzt, die von Computern interpretiert und ausgeführt wird. Binärdateien bestehen aus einer Folge von Nullen und Einsen, die so angeordnet sind, dass sie als Teil eines größeren Computerprogramms gelesen und ausgeführt werden können. Binärdateien sind eine Art von Artefakt.

Lokale Repositorys

Lokale Repositorys sind Repositorys, in denen Sie proprietäre Artefakte zur Verfügung stellen können. Sie bieten einen zentralen Speicherort für Ihre internen Binärdateien und sollten, wie alle Repositorys, speziell auf einen bestimmten Artefakttyp zugeschnitten sein.

Remote-Repositorys

Remote-Repositorys sind ein Proxy für ein Repository, das sich auf einem Remote-Server befindet. Sie ermöglichen die Verwaltung von Artefakten, die von einer externen Quelle (z. B. Maven Central) importiert werden. Durch das Caching der erforderlichen Abhängigkeiten bieten Remote-Repositorys für Developer einen konsistenten und zuverlässigen Zugriff auf Pakete.

Virtuelle Repositorys

Virtuelle Repositorys fassen lokale und Remote-Repositorys für einen bestimmten Pakettyp in einer einzigen Repository-Struktur zusammen und vereinfachen so das Development und die Konfiguration mit einer einzigen URL für die Auflösung und Bereitstellung.

Hochverfügbarkeit

Hochverfügbarkeit (englisch "high availability", HA) bedeutet, dass ein System kontinuierlich und ohne Unterbrechungen arbeiten kann, normalerweise mit einer Uptime von 99,9 % oder mehr. Mit einem Paketverwaltungssystem, das die Ausfallsicherheit Ihrer Paket-Repositorys innerhalb einer Region oder eines Rechenzentrums gewährleistet, können Sie einen kontinuierlichen und zuverlässigen Zugriff auf wichtige Komponenten sicherstellen

Replikation

Hierbei geht es um die Replikation eines Artefakts zwischen einem oder mehreren Repositorys, die sich über mehrere Instanzen einer bestimmten Artefakt-Management-Lösung erstrecken. Die Replikation unterstützt wichtige Use Cases wie verteiltes Development (standortübergreifend), Hochverfügbarkeit und Disaster Recovery.

Föderation

Föderation bezieht sich auf einen Prozess, der Daten wie Artefakt-Metadaten, Identitäts- und Authentifizierungsinformationen über vernetzte Systeme hinweg synchronisiert. Zum Beispiel, wenn zwei oder mehr Instanzen der JFrog-Plattform über definierte Repositorys im Zuge einer bidirektionalen Synchronisation oder über User und Berechtigungen verbunden werden.

Best Practices für Artefakt-Management

1. Pakete und Artefakte in einem Tool speichern, das für diesen speziellen Dateityp konzipiert ist

Keine zwei Pakettypen sind gleich – sie unterscheiden sich in Struktur, Inhalt und Konfiguration. Es ist zwar möglich, mehrere Dateitypen in einem generischen Repository zu speichern, jedoch schränken Unternehmen, die sich für diesen Ansatz entscheiden, damit die Automatisierung ihrer Software-Pipelines ein und reduzieren die Daten, die während des Entwicklungsprozesses gesammelt werden. Aus diesem Grund wird im Allgemeinen empfohlen, Pakete und Artefakte in Repositories zu speichern, die für einen bestimmten Pakettyp entwickelt wurden.

Warum das wichtig ist

Jeder Paketmanager hat seine eigenen Befehle, Spezifikationen und Datenausgaben. Um Automatisierung und eine nahtlose Integration in Ihre Software-Pipelines zu ermöglichen, müssen die Repositories, in denen Sie Ihre Artefakte speichern, sich in die verschiedenen Tools, die Sie in Ihren Build-Prozessen verwenden, integrieren lassen und mit ihnen kommunizieren können.

Das Speichern von Paketen und Artefakten in einem Tool, das speziell für diesen Dateityp entwickelt wurde, erleichtert auch die Erfassung von Metadaten. Wenn Pakete und Artefakte in einem Tool gespeichert werden, das auf diesen Dateityp ausgelegt ist, kann das Tool Informationen über die Dateien

erfassen und speichern, z. B. den Dateityp, das Erstellungsdatum, wer sie erstellt hat und andere ergänzende Informationen.

Ein weiterer wichtiger Grund ist, dass es einfacher ist, organisiert zu bleiben. Ein Tool, das für einen bestimmten Dateityp entwickelt wurde, kann verwendet werden, um die Pakete und Artefakte in Ordnern, Tags und nach anderen Kategorien zu organisieren, sodass User die benötigten Pakete und Artefakte leicht finden können. Dies erleichtert auch den Überblick über die verschiedenen Versionen von Paketen und Artefakten.

Beispielimplementierung

Nehmen wir zum Beispiel Maven. Maven ist ein leistungsstarkes Tool für die Verwaltung von Softwareprojekten und unterstützt eine Vielzahl von Projekttypen, darunter Java EE-Projekte, Webanwendungen und mobile Apps. Maven wurde außerdem so konzipiert, dass es die Verwaltung von Abhängigkeiten, die Erstellung von Artefakten und die Paketierung von Anwendungen erleichtert.

Das Maven-Repository muss mehrere einzigartige Features unterstützen, um nativ mit dem Maven-Paketmanager zu arbeiten. Es muss in der Lage sein, Abhängigkeiten, einschließlich transitiver Abhängigkeiten, automatisch zu erkennen und sicherstellen, dass sie verfügbar und auf dem neuesten Stand sind. Es muss außerdem Multi-Modul-Projekte unterstützen, sodass Anwendungen in mehrere kleinere Teile verpackt werden können, die unabhängig voneinander gebaut werden.

Best Practices für Artefakt-Management

Fortsetzung

Maven muss in der Lage sein, Artefakte wie JARS und WARs zu erstellen und sie in verteilbare Formate wie ZIP-Dateien zu packen. Zu guter Letzt muss es eine breite Palette von Build-Tools, wie Ant, Gradle und natürlich Maven selbst, unterstützen.

Die Entscheidung, Pakete und Artefakte in einem Tool zu speichern, das für diesen speziellen Dateityp entwickelt wurde (in diesem Beispiel Maven), ermöglicht es Entwicklern, Abhängigkeiten, Build-Artefakte und Paketanwendungen einfach zu verwalten, ohne den Prozess manuell konfigurieren und managen zu müssen.

Nicht für die Verwaltung & Artefakten konzipiert



Public Registrys



Lokale Speicherung



Generic S3

Best Practices für Artefakt-Management

Fortsetzung

2. Trennung von Artefakten und Abhängigkeiten mit einer lokalen, remote und virtuellen Repository-Struktur

Moderne Softwareanwendungen bestehen aus einer Vielzahl von Paketen und Abhängigkeiten. Dazu gehören intern gefertigte Komponenten und Open-Source-Projekte, die Sie zur Beschleunigung Ihrer Releases verwenden. Als Teil eines Projekts benötigen Sie zum Beispiel Logging-Funktionen. Anstatt ein komplettes Logging-Framework von Grund auf neu zu entwickeln und zu testen, können Sie auf OSS-Logging-Frameworks wie log4j zurückgreifen, die bereits weiter entwickelt sind und sich bewährt haben.

Warum das wichtig ist

Aus Sicherheits-, Struktur- und Compliance-Gründen benötigen Unternehmen eine einfache Möglichkeit, Artefakte zu speichern und zu organisieren, damit klar ist, welche Artefakte proprietär sind und welche aus öffentlichen Quellen stammen (z. B. Abhängigkeiten). Um intellektuelle Artefakte besser von nicht-intellektuellen Artefakten zu trennen, ist es empfehlenswert, getrennte Repositories für diese beiden unterschiedlichen Klassifizierungen von Artefakten zu verwenden.

Beispielimplementierung

Es empfiehlt sich, eine Repository-Struktur zu verwenden, die sowohl lokale als auch remote und virtuelle Repositories umfasst.

In **lokalen** Repositories können Sie alle Ihre proprietären Artefakte speichern, die Sie entsprechend Ihren Anforderungen entwickeln.

Verwenden Sie **Remote**-Repositories, um Pakete aus öffentlichen OSS-Repositories wie Maven Central, Docker Hub, npm usw. zwischenspeichern.

Virtuelle Repositories bieten eine einzigartige Möglichkeit, die Verwaltung von Repositories zu vereinfachen. Da es sich um eine Kombination von lokalen und Remote-Repositories handelt, können Sie ein virtuelles Repository als den einzigen Endpunkt für die Erstellung neuer proprietärer Artefakte verwenden, indem Sie OSS-Paketabhängigkeiten auflösen und intellektuelle Artefakte in Artifactory veröffentlichen. Wenn ein Client eine Verbindung zu einem virtuellen Endpunkt herstellt, sollten es möglich sein, die Liste der Repositories, von denen Abhängigkeiten aufgelöst werden sollen, hinzuzufügen/zu entfernen/zu ändern/zu priorisieren, ohne eine Änderung an der Client-Konfiguration vorzunehmen.

Durch die Verwendung von lokalen und Remote-Repositories können Unternehmen ihre intellektuellen von nicht-intellektuellen Artefakten trennen. Der Einsatz virtueller Repositories vereinfacht den Administrations- und Governance-Aufwand bei Änderungen.

Best Practices für Artefakt-Management

Fortsetzung

3. Öffentliche Proxy-Registries für ein lokal zwischengespeicherte Set von Artefakten

Heutzutage bilden Open-Source-Pakete die Grundlage für jede im Einsatz befindliche Software. Developer und Build-Systeme verlassen sich häufig auf diese Open-Source- und Drittanbieter-Librarys, die in der Regel auf Remote-Systemen im Internet gehostet werden, um Softwareanwendungen zu erstellen. Obwohl diese Praxis weit verbreitet ist, stehen Unternehmen vor einigen Herausforderungen, wenn sie auf öffentlich gehostete Artefakte zurückgreifen.

Warum das wichtig ist

Unternehmen müssen Zuverlässigkeit, Verfügbarkeit, Sicherheit und Nachvollziehbarkeit berücksichtigen, wenn sie mit Artefakten aus öffentlichen Repositories arbeiten. Probleme können auftreten, wenn Artefakte/Abhängigkeiten in öffentlichen Repositories nicht mehr verfügbar sind oder Netzwerkengpässe zu Zuverlässigkeitsproblemen führen. Außerdem kann die Nachvollziehbarkeit schwierig sein, wenn Abhängigkeiten direkt aus dem Internet heruntergeladen werden. Die lokale Zwischenspeicherung von Artefakten in Remote-Repositories ermöglicht einen zuverlässigen und konsistenten Zugriff auf Pakete, ohne dass ständige Downloads erforderlich sind.

Beispielimplementierung

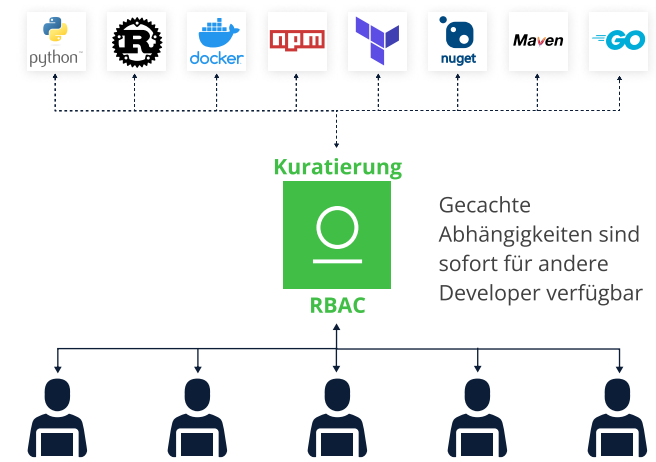
Wenn ein Ubuntu-Docker-Image von Docker Hub mittels dem Docker Pull ubuntu: 20.04

heruntergeladen wird, kommen die Downloads direkt von Docker Hub. Jedes Mal, wenn dasselbe Image angefordert wird, muss die Anfrage Docker Hub erreichen und der Download-Prozess beginnt von vorne. Am besten ist es, ein Remote-Repository zu erstellen, das als Proxy für alle Anfragen an Docker Hub fungiert. Sobald das Image im Remote-Proxy-Repository gecacht wurde, wird dasselbe zwischengespeicherte Image für alle nachfolgenden Anfragen aus beliebigen Quellen bereitgestellt, was die Ingress-Nutzung reduziert.

SECURITY BEST PRACTICES

Erlauben Sie Developern, Pakete direkt aus dem Internet herunterzuladen

Unternehmen, die Best Practices für die Sicherheit einführen möchten, setzen ihren Artefakt-Repository-Manager als Vermittler zwischen Developern und dem Internet ein, indem sie öffentliche Registries als Proxy-Server verwenden. Selbst wenn ein böses Paket nicht in den Build gelangt, kann ein Developer, der Malware auf sein Gerät herunterlädt, das gesamte Netzwerk und System gefährden. Der Zugriff auf alle Pakete über ein Tool wie Artifactory bietet eine erste Abwehr- und Kontrollebene.



Best Practices für Artefakt-Management

Fortsetzung

4. Repositorys für jede Entwicklungsphase

Unternehmen sollten Repositorys unterhalten, die auf jede Phase ihres SDLC zugeschnitten sind. Software-Releases bestehen aus mehreren Builds/ Paketen/Artefakten und Unternehmen bereiten Software für den Release vor, indem diese die verschiedenen Phasen des SDLC durchläuft und in jeder Phase getestet wird.

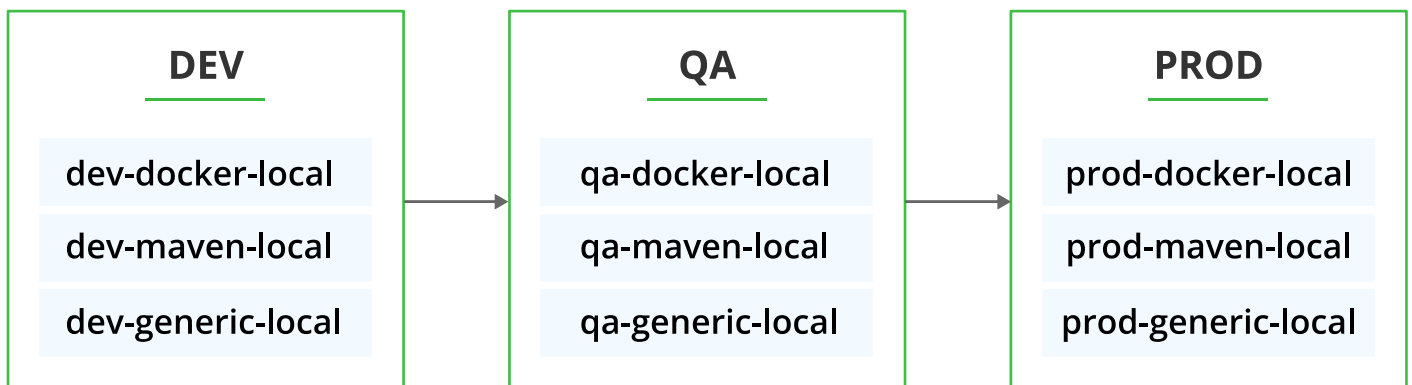
Warum das wichtig ist

Durch das Einrichten von Repositorys für jede Phase des SDLC können Unternehmen alle Komponenten für einen potenziellen Release in Repositorys verschieben, die für die jeweilige Phase geeignet sind. Außerdem kann

dadurch kontrolliert werden, wann die Software freigegeben wird (in der Regel, wenn sie bestimmte Validierungen bestanden hat), wer auf die Komponenten zugreifen kann – abhängig von der Phase des SDLC – und dass zu keinem Zeitpunkt ein Build für einen Teil des Releases erneut ausgeführt wird. Dieser Ansatz hilft Unternehmen, die Integrität und Sicherheit zu verbessern und den Status von Software-Releases transparent nachzuverfolgen.

Beispielimplementierung

Das folgende Diagramm zeigt den SDLC eines hypothetischen Unternehmens und die Komponenten, aus denen sich ein Release zusammensetzt. Stellen Sie sich vor, wie ein spezielles Repository für jede Phase dem Unternehmen hilft, sicher und nahtlos von einer Phase des Softwareentwicklungszyklus zur nächsten zu gelangen.



Best Practices für Artefakt-Management

Fortsetzung

5. Weisen Sie jedem Team ein eigenes Set von lokalen, remote und virtuellen Repositorys zu

Jedem Team oder "Projekt" sollte ein eigenes Set von Repositorys für die Softwareentwicklung zugewiesen werden. Darüber hinaus sollte die verwendete Artefakt-Management-Lösung über eine Art Verwaltungsorgan verfügen, um Ressourcen wie Repositorys, Builds, Release-Bundles und Pipelines zu gruppieren. Auf diese Weise können Ressourcen leichter an ein bestimmtes Team delegiert werden.

Warum das wichtig ist

Das gilt aus mehreren Gründen als Best Practice. Zum einen werden Artefakte in Unternehmen meist instinktiv in einem einzigen System gespeichert, obwohl es für bestimmte Teams aus Gründen der Einfachheit und Sicherheit sinnvoller sein kann, auf eine vordefinierte Untergruppe von Artefakten zuzugreifen. Zum anderen erleichtert die Zuweisung eigener Sets von Repositorys für jedes Team das Auffinden und den Zugriff auf die Komponenten, an denen gearbeitet werden soll, die gemeinsame Nutzung der Komponenten durch verschiedene Teams und – falls erforderlich – die Replikation oder Zusammenführung bestimmter Repositorys. Wenn Sie mit sehr großen Repositorys arbeiten, kann es kompliziert werden. Indem Sie den verschiedenen Teams ihr eigenes Set an Repositorys zuweisen, nehmen Sie dem zentralen Administrator einen Teil der Last und des Wartungsaufwands ab. So weiß ein zentraler Administrator möglicherweise nicht,

was er entfernen oder updaten kann, ohne dass dies Auswirkungen auf ein bestimmtes Team hat. Letztlich ist es auch finanziell ein kluger Schachzug, da es eine effizientere und nachvollziehbare Zuordnung von Kosten zu bestimmten Teams ermöglicht.

Beispielimplementierung

Team A verfügt über lokale, remote und virtuelle Repositorys für die benötigten Pakettypen (z. B. Maven, Docker, Npm und Generic). Sie haben auch ihre eigenen Administratoren, Berechtigungen und Richtlinien auf Projektebene. Team B verfügt über lokale, remote und virtuelle Repositorys für die benötigten Pakettypen (z. B. Rust, Conan, Docker, Helm und Generic). Wie Team A hat auch Team B eigene Administratoren auf Projektebene sowie eigene Berechtigungen und Richtlinien.

EINSATZ DER JFROG-PLATTFORM		
Globaler Administrator, Richtlinien, RBAC-Nutzungskontingente		
Projekt 1	Projekt 1	Projekt 1
Projektadministrator	Projektadministrator	Projektadministrator
User/Rollen	User/Rollen	User/Rollen
Repositorys	Repositorys	Repositorys
Richtlinien	Richtlinien	Richtlinien
Nutzung	Nutzung	Nutzung
Projekt 1	Projekt 1	Projekt 1
Projektadministrator	Projektadministrator	Projektadministrator
User/Rollen	User/Rollen	User/Rollen
Repositorys	Repositorys	Repositorys
Richtlinien	Richtlinien	Richtlinien
Nutzung	Nutzung	Nutzung

Best Practices für Artefakt-Management

Fortsetzung

6. Artefakte zwischen SDLC- Umgebungen übertragen – statt sie neu zu generieren

Der Softwareentwicklungs-Lebenszyklus (Software Development Life Cycle = SDLC) umfasst verschiedene Phasen, die von der anfänglichen Planung und dem Entwurf über die Programmierung bis hin zu Tests, der Bereitstellung und der Wartung reichen. Ein wesentliches Element des Softwareentwicklungs-Lebenszyklus ist die Übertragung von Artefakten von einer Umgebung in eine andere. Um den Prozess so effizient und effektiv wie möglich zu gestalten, empfiehlt es sich, die Artefakte zu promoten, anstatt sie für jede Phase neu zu erstellen.

Warum das wichtig ist

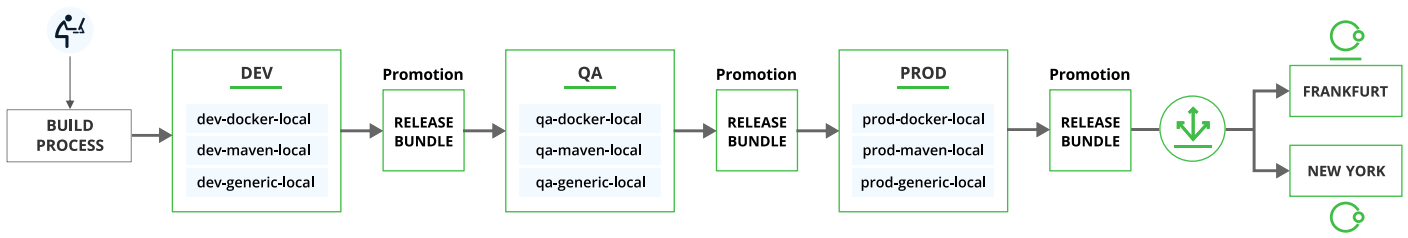
Das Übertragen von Artefakten über den gesamten Lebenszyklus der Softwareentwicklung hinweg trägt dazu bei, Konsistenz, Zuverlässigkeit und Rückverfolgbarkeit zu gewährleisten und gleichzeitig Zeit und Ressourcen zu sparen. Durch das Promoten anstelle der Neugenerierung von Artefakten können Developer sicher sein, dass der Code, den sie testen, mit der Version übereinstimmt, die letztendlich bereitgestellt wird, und gleichzeitig den Reifeprozess der Software klar nachvollziehen. Security-Scans als Teil des Promotion-Prozesses ermöglichen, dass die Scan-Ergebnisse als eine Art Ampel fungieren, die die Promotion blockieren oder genehmigen,

sodass die Builds sicher sind, wenn sie in die nächste Phase des SDLC übergehen. Dieser Ansatz trägt dazu bei, Fehler zu reduzieren, die Quality Assurance zu verbessern, die Produktivität zu steigern und die Zeit bis zur Markteinführung zu verkürzen – das macht ihn zu einem wertvollen Ansatz für jedes Development-Team.

Beispielimplementierung

Bei einem Unternehmen mit drei Phasen des SDLC – DEV, QA und PROD – würden diese Phasen in deren Artefakt-Manager zu erkennen sein. Das würde folgendermaßen aussehen:

- **DEV:** Build-Outputs und Abhängigkeiten, die in der Coding-Phase verwendet werden, werden in den DEV-Repositorys abgelegt.
 - **QA:** Die Builds und Pakete, aus denen ein Release besteht, werden gebündelt und dann für weitere Tests von DEV in die QA-Repositorys verschoben.
- * Anmerkungen:** Anmerkung: Es können Regeln definiert werden, damit nur Artefakte, die von DEV promotet werden, in die QA-Repositorys aufgenommen werden. Außerdem kann festgelegt werden, dass nur bestimmte User Lesezugriff auf das Repository haben, um Tests durchzuführen, oder dass sie nur Metadaten zu den Artefakten veröffentlichen können.
- **PROD:** Sobald eine ordnungsgemäße QA-Validierung stattgefunden hat, kann das Software-Release-Bundle in die Produktion promotet werden, die als der einzige Ort definiert werden kann, von dem Produktionssysteme Updates beziehen können.



Best Practices für Artefakt-Management

Fortsetzung

7. Speichern und Verwalten der Metadaten zusammen mit dem dazugehörigen Artefakt

Zur Erinnerung: Metadaten sind zusätzliche Attribute, die mit einem Artefakt verknüpft sind. Diese Attribute bieten Administratoren die Möglichkeit, Artefakte effektiv zu organisieren. Sehen wir uns jetzt die verschiedenen Arten von Metadaten an.

Allgemeine Metadaten

- Artefaktname
- Artefakt-Version
- Artefakt-Typ
- Artefakt-Größe
- Prüfsumme des Artefakts

Build-bezogene Metadaten

- Build-Name
- Build-Nummer
- Zeitstempel des Builds

Metadaten zu den Abhängigkeiten:

- Informationen über die Abhängigkeiten des Artefakts

Benutzerdefinierte Metadaten

- Zusätzliche projektspezifische Metadaten können definiert und mit Artefakten verknüpft werden

Warum das wichtig ist

Metadaten spielen bei der Softwareentwicklung eine wichtige Rolle. Sie ermöglichen es Unternehmen, die Historie eines Artefakts oder Builds zu verstehen und nachzuvollziehen, was damit geschehen ist, z. B. Validierungen der Software auf dem Weg zum Release. Metadaten sind auch wichtig, um festzustellen, ob sich an einem Paket etwas geändert hat, was darauf hinweisen könnte, dass die Erfassung detaillierter Metadaten für die Rückverfolgbarkeit von Software unerlässlich ist und sogar zum Triggern von Workflows, Webhooks oder Plugins verwendet werden kann.

Beispielimplementierung

Wir können Metadaten für Artefakte auf verschiedene Arten speichern und verwalten. Einige Möglichkeiten dazu sind im Folgenden angeführt:

1. Metadaten-Eigenschaften

Ihr Artefakt-Manager sollte in der Lage sein, Schlüssel-Wert-Paare mit Artefakten zu verknüpfen. Diese Eigenschaften können als Metadaten verwendet werden und sollten über die Benutzeroberfläche oder REST-APIs hinzugefügt oder aktualisiert werden – entweder einzeln oder alle auf einmal.

- **Manuelle Verwaltung via UI** - Über die Benutzeroberfläche können Sie Metadaten für einzelne Artefakte manuell hinzufügen, bearbeiten oder entfernen. So können Sie Schlüssel-Wert-Paare eingeben oder bestimmte Metadatenfelder direkt in der Benutzeroberfläche definieren.
- **REST API**- Umfangreiche REST-APIs ermöglichen die programmgesteuerte Verwaltung von Artefakt-Metadaten. Sie können die API verwenden,

Best Practices für Artefakt-Management

Fortsetzung

um Metadaten für bestimmte Artefakte oder eine Gruppe von Artefakten abzurufen, zu aktualisieren oder zu löschen.

- **File Upload Request** - Beim Hochladen von Artefakten in eine Artefakt-Management-Lösung sollte es möglich sein, Metadateninformationen in die Datei-Upload-Anfrage aufzunehmen. Dies lässt sich umsetzen, indem benutzerdefinierte Header oder Parameter in der Upload-Anfrage festgelegt werden, z. B. Artefaktnamen, Version oder andere gewünschte Metadatenfelder.

2. Integration von Build-Tools

Durch die Integration von verschiedenen Build-Tools wie Maven, Gradle oder Jenkins können Artefakt-Management-Lösungen automatisch Build-bezogene Metadaten während des Build-Prozesses erfassen. Dazu gehören Details wie Build-Name, Build-Nummer, Zeitstempel und mehr. Die Build-Informationen werden dann mit den hochgeladenen Artefakten verknüpft.

Mithilfe dieser Methoden können Sie Metadaten für Ihre Artefakte speichern, aktualisieren und verwalten und so die Organisation und das Abrufen von Informationen über Ihre Artefakte während ihres gesamten Lebenszyklus erleichtern.

8. Erstellung von Richtlinien, die die Zugriffsrechte auf Repositories und das Ausführen von Aktionen regeln

Unternehmen sollten über klare Regeln und Richtlinien verfügen, um steuern zu können, welche Personen und Systeme wann auf Software-Artefakte zugreifen dürfen. Es ist wichtig, dass Ihre Artefakt-Management-Lösung diese Richtlinien via RBAC und/oder Integrationen mit anderen Identity-Management-Lösungen umzusetzen kann.

Warum das wichtig ist

Software-Artefakte sind das Herzstück der Software-Entwicklung. Da die Pakete urheberrechtlich geschützten Code und Informationen enthalten, besteht ein großer Teil des Artefakt-Managements darin, sicherzustellen, dass keine Unbefugten auf Assets zugreifen können, die nicht für sie bestimmt sind.

Beispielimplementierung

User, Gruppen, Rollen und Berechtigungen hängen insofern zusammen, als Berechtigungen Rollen zugewiesen werden, die wiederum Usern und Gruppen zugewiesen werden können. Es ist möglich, Berechtigungen auf Repository-Ebene oder auf Projektebene zu vergeben. Das bedeutet, dass verschiedene User und Gruppen unterschiedliche Zugriffsrechte haben können – je nach Umfang des Projekts oder des Repositories.

Best Practices für Artefakt-Management

Fortsetzung

9. Definition von Cleanup-Richtlinien für Repository-Hygiene und -performance

Die Anzahl der heute produzierten Binärdateien ist nahezu unüberschaubar. Mit der Zeit sammeln sich immer mehr Binärdateien an, die irgendwann eine große Menge an Speicherplatz benötigen. Das Ausmisten der Dateien kann mehrerlei bedeuten. Einerseits kann es sich um das tatsächliche Löschen der Binärdateien aus dem Repository-Manager handeln oder es bedeutet, wie in den meisten Fällen, dass die Binärdateien in einer kostengünstigeren Speicher-Lösung archiviert werden (dies ist besonders für stark regulierte Branchen relevant. In diesem Szenario werden die archivierten Binärdateien von dem Repository-Manager in die Archivierungslösung übertragen).

Warum das wichtig ist

Genau wie in der physischen Welt wird unser "Zeug" zu einem ernsthaften Problem, wenn es sich immer weiter anhäuft und nicht richtig organisiert oder verwaltet wird. Cleanup-Richtlinien für Artefakt-Repositorys sind wichtig, um zu gewährleisten, dass die verwendeten Artefakte sicher und aktuell sind. Diese Richtlinien unterstützen die Integrität des Repositorys, indem sie verhindern, dass alte, anfällige oder schädliche Artefakte darin verbleiben. Durch das Entfernen von Artefakten, die nicht mehr verwendet werden, können Cleanup-Richtlinien auch die Gesamtgröße des Repositorys reduzieren. Dies trägt nicht nur dazu bei, dass das Repository

übersichtlich und effizient bleibt, sondern kann auch die Wartungskosten senken.

Beispielimplementierung

Die branchenweite Best Practice für ein Cleanup besteht darin, die mit der Binärdatei verknüpften Metadaten für das Cleanup zu verwenden. Das auf Metadaten basierende Cleanup ist äußerst leistungsfähig und flexibel, da es wichtige Informationen darüber liefert, was das Artefakt ist, wie und wo es verwendet wurde, wie alt es ist und vieles mehr. Um die besten Cleanup-Ergebnisse anhand von Metadaten zu erzielen, muss der verwendete Repository-Manager in der Lage sein, Metadaten für die verschiedenen Typen von gespeicherten Artefakten zu unterstützen.

Sie können Metadaten wie Datum/Uhrzeit, Ordnername, reguläre Ausdrücke, Anzahl der Downloads, letzter Download usw. verwenden.

Hier ist ein Beispiel dafür, wie Sie ungenutzte Artefakte aus Artifactory löschen können.

Auswahl der richtigen Artefakt-Management-Lösung

Richtiges Artefakt-Management ist ohne das richtige Tool eine Herausforderung. An dieser Stelle kommt ein Artefakt-Repository-Manager ins Spiel.

Ein Artefakt-Repository-Manager ist speziell dafür konzipiert, verschiedene Arten von Artefakten für Software-Builds an einem zentralen Ort zu speichern, zu verwalten, zu versionieren und bereitzustellen, und bietet die beste Möglichkeit, eine ständig wachsende Anzahl von Artefakten zu verwalten. Sie können zwar versuchen, Best Practices für das Artefakt-Management ohne einen Artefakt-Repository-Manager zu implementieren, aber Ihr Erfolg wird sich in Grenzen halten.

Bei der Auswahl eines Artefakt-Repository-Managers sollten Sie darauf achten, dass er folgende Funktionen bietet:

1. Vielseitigkeit

Fragen Sie sich: Unterstützt diese Lösung die erforderlichen Artefakttypen und Build-Tools, die ich bereits verwende und die ich in Zukunft verwenden könnte?

Beim Überprüfen, ob ein Repository-Manager universell einsetzbar ist, sind zwei Elemente zu berücksichtigen:

- Erstens sollte das Artefakt-Repository alle Paket- und Dateitypen, mit denen Sie arbeiten, nativ unterstützen. Native Unterstützung bedeutet, dass es sowohl dieselben Befehle wie das Tool (Paketmanager), mit dem das Paket erstellt wurde, als auch mehrere

Repository-Typen (lokal, remote, virtuell) zur Speicherung und Organisation der Artefakte unterstützt. Universelle Repository-Manager unterstützen in der Regel die meisten Pakettypen.

- Zweitens sollten Sie prüfen, ob die Lösung in die Tools integriert werden kann, die Sie bereits im Development-Prozess verwenden. Untersuchen Sie die Out-of-the-Box-Integrationen sowie die APIs und die CLI, die von jeder Repository-Lösung angeboten werden.

Auf den ersten Blick mag dies wie ein „Nice-to-have“ erscheinen, aber beide Elemente sind für die Pipeline-Automatisierung notwendig.

2. Automatisierung

Fragen Sie sich: Wie unterstützt diese Lösung die Automatisierung meiner Development-Pipelines?

Einer der Hauptgründe für die Umstellung des Artefakt-Managements von universellen Ad-Hoc-Speicherlösungen und mehreren privaten Registries auf ein maßgeschneidertes Artefakt-Repository ist die vollständige Automatisierung von Software-Pipelines.

Suchen Sie nach Lösungen, die eine Vielzahl von Triggern und Workflows über Webhooks, Plugins, Integrationen und CLI unterstützen. Stellen Sie außerdem sicher, dass die Lösung, in die Sie investieren, die gleichen Befehle unterstützt, die mit einem bestimmten Build-Tool verwendet werden, um eine vollständige Automatisierung zu ermöglichen.

Auswahl der richtigen Artefakt-Management-Lösung

Fortsetzung

3. Skalierbarkeit

Fragen Sie sich Kann diese Lösung skaliert werden, um ein verteiltes Development und eine erhöhte Auslastung zu bewältigen?

Bei den Überlegungen zur Skalierung sind einige Faktoren zu berücksichtigen, darunter:

- Wie viele Artefakte verwalten Sie?
Überlegen Sie, wie viele Artefakte Sie momentan haben und wie viele Sie in Zukunft haben werden. Laut JFrog-Daten verwalten 20 % der Unternehmen Artefakte von mindestens 5 TB, wobei die größten Unternehmen Daten im Petabyte-Bereich verwalten.
- Wie viele Standorte müssen unterstützt werden? Wenn Sie Development an mehreren Standorten betreiben, muss Ihr Artefakt-Repository auch etliche Optionen bieten, um die Teams synchron zu halten, z. B. Pull- und Push-Replikation, vollumfängliche bidirektionale Synchronisierung und eine Reihe von definierbaren Triggern für die Synchronisierung.
- Wie viele Clients werden zu einem bestimmten Zeitpunkt auf das System zugreifen? Berücksichtigen Sie die Mechanismen, die zur Bewältigung von einer erhöhten Systemauslastung zur Verfügung stehen, die durch Tausende von Clients, die gleichzeitig auf Artefakte zugreifen, entstehen kann. Suchen Sie nach Lösungen mit Caching oder anderen Verteilungsmechanismen, um diese Probleme zu entschärfen.

4. Hybride Bereitstellung

Fragen Sie sich: Wo kann ich diese Lösung bereitstellen und einsetzen?

Bei der Frage, wo Ihre Artefakt-Management-Lösung gehostet und betrieben werden soll, spielen verschiedene Faktoren eine Rolle.

Unternehmen, die besonderen Wert auf Sicherheit legen oder Branchenvorschriften unterliegen, müssen bestimmte Artefakte möglicherweise in einer selbst gehosteten, lokalen Umgebung speichern. Gleichzeitig werden unter Umständen Anwendungen in der Cloud ausgeführt, die erfordern, dass bestimmte Artefakt-Registries in der Nähe ihrer Laufzeitumgebung bleiben.

Um ein Höchstmaß an Flexibilität zu erreichen, sollten Sie nach Artefakt-Management-Lösungen suchen, die es ermöglichen, Instanzen überall dort bereitzustellen, wo sie benötigt werden (in der Cloud – hybrid oder Multicloud – und lokal), und die automatisch synchronisiert werden.

5. Sicherheit

Fragen Sie sich: Stellt diese Lösung sicher, dass die Komponenten gegen Zugriff von außen geschützt sind und keine Schwachstellen aufweisen?

Der Sicherheitsaspekt eines Artefakt-Management-Tools ist von entscheidender Bedeutung und muss aus zwei Blickwinkeln betrachtet werden: Plattformsicherheit und Anwendungssicherheit.

Auswahl der richtigen Artefakt-Management-Lösung

Fortsetzung

- Bei der Plattformssicherheit geht es um die Kontrolle des Zugriffs auf die Artefakte, die Sie in Ihrer Artefakt-Management-Lösung hosten und verwalten. Rollenbasierte Zugriffskontrollen (role-based access control = RBAC) mit granularen Berechtigungssätzen und Benutzerprofilen sind eine Grundvoraussetzung. Stellen Sie außerdem sicher, dass sich Ihre Artefakt-Management-Lösung in die von Ihnen benötigten Identity-Management-Lösungen wie LDAP, OAuth, SAML, SCIM, Crowd, Vault usw. integrieren lässt.
- Bei der Anwendungssicherheit geht es darum, sicherzustellen, dass die von Ihnen freigegebene Software – und damit auch die darin enthaltenen Komponenten – frei von Schwachstellen sind, die von Bedrohungsakteuren ausgenutzt werden könnten. Fragen Sie sich: Bietet die Application-Management-Lösung Integrationen mit verschiedenen heute weit verbreiteten Anwendungssicherheits-Tools (SCA, SAST, DAST usw.)? Oder noch besser: Bietet sie integrierte Funktionen, die in die von Ihnen gewählte Lösung nativ eingebettet sind? Einer der Vorteile der Konsolidierung von Artefakt-Management in einem einzigen System besteht darin, dass Maßnahmen zur Anwendungssicherheit an einem einzigen Ort durchgeführt werden können.



JFrog für Artefakt-Management

JFrog bietet eine hybride, universelle End-to-End-Software-Supply-Chain-Plattform, die den binären Lebenszyklus steuert und verwaltet. Die JFrog-Plattform mit ihrem Herzstück Artifactory verwaltet und sichert den gesamten Fluss von Software-Komponenten (einschließlich derer, die aus externen Quellen in Ihr Unternehmen fließen), die Erstellung neuer Software, alle Pipeline-Tasks und Promotions und zu guter Letzt die Auslieferung aus Ihrer Software-Factory. Mit fortschrittlicher, DevOps-zentrierter integrierter Sicherheit schützt JFrog die Software-Lieferkette in jeder Phase des SDLC und ermöglicht eine intelligente Behebung jeder gefundenen Schwachstelle.

Jede Binärdatei in der JFrog-Plattform ist vollständig rückverfolgbar und gibt Aufschluss darüber, welche Software erstellt wurde, wie sie erstellt wurde, von wem sie erstellt wurde, welche Abhängigkeiten verwendet wurden und welche Version gerade wo läuft. Upload, Download, Promotion, Inklusion, Runtime-Deployment, Retirement, Archivierung und Löschung jedes binären Artefakts werden vollständig aufgezeichnet und können über eine einfach zu bedienende Abfragesprache, REST-API, und die Benutzeroberfläche abgefragt werden. Durch robuste Metadaten und Beweisführung, unveränderliche Release-Bundle und kontinuierliche Sicherheitsüberprüfungen schafft JFrog Vertrauen bei jedem Release.

Die JFrog-Plattform für Artefakt-Management

Maßgebliches Artefakt-Management für flexibles Development und zuverlässige

Software-Bereitstellung in jeder Größenordnung.

DevOps-Flexibilität unterstützen

- Proxy und/oder Speicherung all Ihrer Artefakte mit über 30 integrierten Pakettypen
- Selbst gehostete, SaaS-, hybride und Multicloud-Bereitstellungen bieten die gleiche erstklassige Nutzererfahrung
- Integrieren Sie Artifactory in alle Ihre DevOps-Prozesse und -Tools

Ihre "Single Source of Truth"

- Ein einziges System für Ihre Artefakte mit erweiterter Suche/Tagging
- Verwaltung des gesamten Lebenszyklus von Binärdateien vom Build über Release bis zur Archivierung
- Umfangreiche Metadaten ermöglichen eine umfassende Verifizierbarkeit und Kontrolle Ihrer Assets

Schaffung einer Vertrauensbasis für DevSecOps

- Definieren Sie Richtlinien für den Zugriff, die Nutzung, die Verteilung und die Archivierung in verschiedenen Teams
- Integrierte DevOps-zentrierte Sicherheit wird von JFrog Xray zur Verfügung gestellt
- Sichern Sie kritische Prozesse mit unternehmensweitem Zugriff auf Ihre Artefakte von jedem Ort aus

JFrog bietet einen umfassenden Ansatz zur Verwaltung des sicheren Flusses von Artefakten in Ihrer Software Supply Chain – von der Eingabe bis zur Nutzung.

Fazit

Effektives Artefakt-Management ist eine entscheidende Komponente erfolgreicher Softwareentwicklung. Da sich die digitale Landschaft ständig weiterentwickelt, ist die Bedeutung eines guten Artefakt-Managements nicht von der Hand zu weisen.

Die vorgestellten Best Practices sind für Software-Teams jeder Größe anwendbar und dienen dazu, die Geschwindigkeit, Effizienz, Nachvollziehbarkeit und generell das Gesamtergebnis des Projekts zu verbessern.

Durch die Einhaltung dieser Prinzipien können Unternehmen Risiken minimieren, Workflows rationalisieren und die Langlebigkeit und Zugänglichkeit ihrer wertvollen Artefakte sicherstellen. Um den nächsten Schritt im Artefakt-Management zu machen, starten Sie die Testversion von JFrog und überzeugen Sie sich selbst, wie einfach es ist, diese Tipps in die Praxis umzusetzen.

Heben Sie Ihr Artefakt-Management auf das nächste Level:

[8 Gründe für DevOps, einen Artefakt-Repository-Manager zu verwenden >](#)

