DE-CIX im Interview
Das Internet der Zukunft

c't

MAGAZIN FÜR
PROFESSIONELLE IT

8
AUGUST
2023

Neues Tutorial
MLOps mit Kubeflow
KI-Pipelines mit Python und CRDs erstellen

4 199148 509005 08

9,90 €
Österreich 10,90 €
Schweiz 15.90 CHF
restl. Europa 11,60 €
www.ix.de

Sich selbst hacken

Der Werkzeugkasten für mehr IT-Sicherheit

Metaframeworks für JavaScript
Next.js und Co.: Schneller dank Server-Side Rendering

Terraform-Skripte absichern
Drei Schwachstellenscanner im Vergleich

Nvidia Grace Hopper Superchip
Energieeffizientes HPC für KI-Workloads

Confidential Computing
Intel TDX: Geschützt in der Public Cloud

Remote-Desktop-Software
Besser als Bordmittel: 15 Tools im Vergleich

DevOps: Agil trotz regulatorischer Vorgaben
Nushell: Objektorientierte Shell
Mercury macht Web-Apps aus Jupyter-Notebooks
Windows: Apps mit AppLocker sperren

Neuer Anlauf für Datentransfer in die USA

EU-US Data Privacy Framework

Rechtliche Analyse • Kommentar von Max Schrems

Regulatorische Compliance und DevOps: Geht das?

Straffe regulatorische Vorgaben sind oft ein großes Hindernis, bei der Softwareentwicklung die nötige Agilität für schnelle Innovationen zu erreichen. Mit modernen Entwicklungsmethoden und Automatisierung lassen sich Compliance und Geschwindigkeit verbinden.

Von Dr. Christoph Peters und Robert Ruzitschka

Auch in stark regulierten Branchen sind die IT-Abteilungen Innovationstreiber und stehen unter Druck, mit den Mitteln der agilen Softwareentwicklung kontinuierlich und in schnellen Zyklen Kundennutzen zu schaffen. In diesen Branchen soll die Regulierung in der IT sicherstellen, dass im Betrieb und bei der Änderung von IT-Systemen das Risiko für Ausfälle und Fehlfunktionen so gering wie möglich gehalten wird. Praktisch bedeutet das Vorgaben zum Entwicklungs- und Änderungsprozess sowie umfangreiche Dokumentations- und Reportingverpflichtungen.

Traditionell sehen solche Vorgaben manuelle Schritte vor. Diese Vorgehensweise ist allerdings für die moderne Softwareentwicklung, bei der Softwareentwicklungsteams oft mehrmals täglich Code für eine produktive Software liefern, völlig ungeeignet. Damit sind zwei zentrale Herausforderungen zu bewältigen: Wie können die Teams bereits in der Entwicklungsphase der Software die Risiken minimieren, die beim Ändern der Produktionssysteme entstehen? Und wie lässt sich garantieren, dass alle notwendigen Informationen für eine abschließende Beurteilung der Qualität vor der Freigabe zur Verfügung stehen, möglichst automatisiert bewertbar sind und die Dokumentation erstellt wird?

Softwareentwicklungsprozess automatisieren

Für Continuous Delivery benötigte Werkzeuge können hierbei helfen. Ein automatisierter Entwicklungsprozess bringt nicht nur Geschwindigkeit, Transparenz und Wiederholbarkeit, sondern erlaubt auch, die Ergebnisse der einzelnen Prozessschritte automatisiert zu evaluieren und zu protokollieren. Damit sind viele Informationen, die für die Dokumentation gemäß den regulatorischen Vorgaben nötig sind, bereits vorhanden. Die toolunterstützte Aggregation dieser Daten zusammen mit einer Standardisierung der möglichen Workflows erlaubt tatsächlich, schnelle Softwarelieferzyklen mit regulatorischer Compliance zu verbinden.

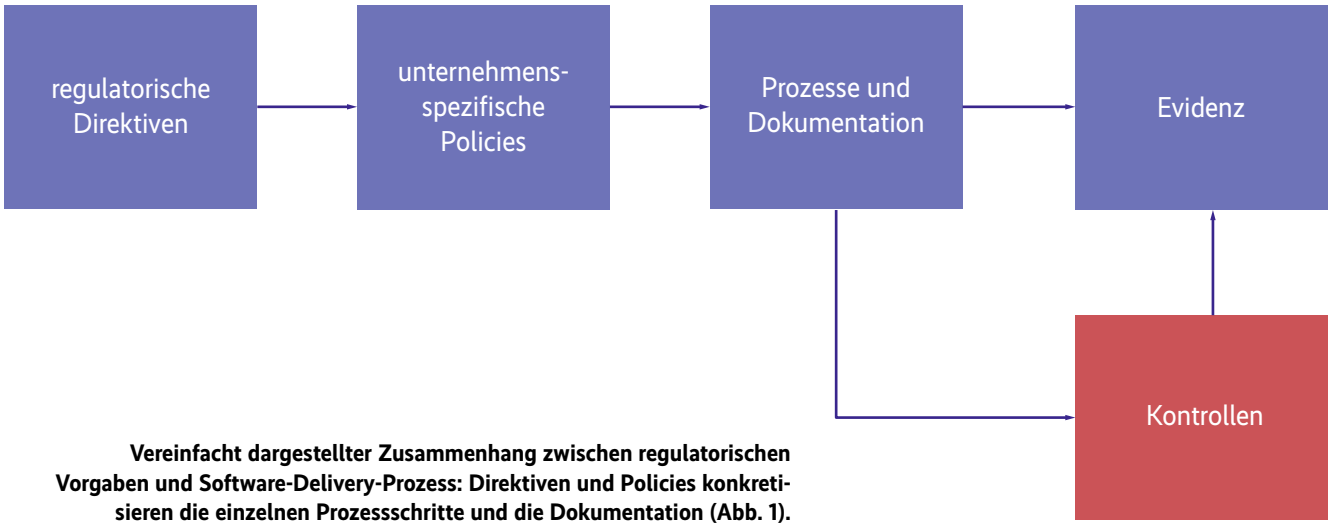
Ein weiterer wichtiger Vorteil von Automatisierung in diesem Kontext ist aber auch, dass die Entwicklungsteams real einen höheren Grad an Compliance erreichen können. Regulatorische Anforderungen zu erfüllen, empfinden viele Teams nicht als geschäftsrelevant, sondern eher als permanente Störung. Automatisierung kann hier zu höherer Akzeptanz und damit auch höherer tatsächlicher Compliance beitragen.

Die Regulierungsbehörden (beispielsweise auf nationaler oder europäischer Ebene) erstellen Anforderungsdokumente – Direktiven –, die bestimmte Vorgaben allgemein beschreiben. Um diese Anforderungen für Entwicklungsteams implementierbar zu machen, ist meist noch eine unternehmens- oder zumindest branchenspezifische Interpretation notwendig – die Policies. Sie präzisieren konkrete Prozessschritte und die notwendige Dokumentation.

Problematisch für Entwicklungsteams ist, dass derartige Policies von Risiko- oder Securityabteilungen verfasst werden, die keinen tieferen Einblick in die technischen Details von Softwareentwicklung haben und keine geeigneten Tools anbieten, die Policies umzusetzen.

iX-TRACT

- Die aus regulatorischen Anforderungen abgeleiteten Policies sind oft ohne Kenntnis der Abläufe in der Softwareentwicklung verfasst. Viele Entwicklerteams empfinden regulatorische Anforderungen als unrealistisch und nicht umsetzbar.
- Die üblichen Continuous-Delivery-Tools können auch Prozesse dokumentieren und Compliance sicherstellen.
- Quality Gates in den Pipelines sind Hürden, bei denen der CD-Prozess automatisch stoppt, sollten regulatorische Anforderungen nicht erfüllt sein. Sie lassen sich als Konfigurationsdateien, etwa in YAML, schreiben.
- Zentralisierte Plattformen zur Complianceautomatisierung können Teams entlasten, sollten aber nur angeboten, nicht vorgeschrieben werden.



Vereinfacht dargestellter Zusammenhang zwischen regulatorischen Vorgaben und Software-Delivery-Prozess: Direktiven und Policies konkretisieren die einzelnen Prozessschritte und die Dokumentation (Abb. 1).

Das macht die konkrete Anwendung schwierig und führt im schlimmsten Fall zu mangelnder Implementierung der regulatorischen Anforderungen, da die Teams die Vorgaben als unrealistisch und weltfremd empfinden.

Neben der Dokumentation der Prozesse ist es immer auch wesentlich, nachweisen zu können, dass die erforderlichen Prozessschritte tatsächlich durchgeführt wurden. Hier spricht man von Evidenz, die vorgehalten und auf Nachfrage eines Auditors jederzeit verfügbar sein muss. Kontinuierliche Kontrollen stellen das sicher (Abbildung 1).

Der Prozess in der Praxis

Als Beispiel dient ein vereinfachter Prozess aus der Finanzbranche. Die regulatorischen Vorgaben erfordern hier unter anderem, die Anforderungen zu beschreiben und sie mit den Änderungen in der Produktion zu verknüpfen, die Qualität der ausgerollten Software sicherzustellen und die Securityanforderungen zu berücksichtigen. Sie verlangen einen strukturierten Prozess, um Änderungen in Produktion auszurollen. Zusätzlich muss die Integrität der installierten Software sichergestellt sein und der Zugriff auf Produktionssysteme darf nur durch berechtigte Personen erfolgen.

Die zugehörige erforderliche Dokumentation umfasst die Beschreibung der Anforderungen, eine umfangreiche Testdokumentation mit Referenzen auf die Anforderungen, etwaige Änderungen in der Beschreibung der möglichen Sicherheitsrisiken und die Bestätigung der Freigabe durch eine autorisierte Person. Diese stellt sicher, dass alle regulatorischen Anforderungen eingehalten wurden und damit das Risiko der Änderungen so klein wie möglich ist.

Typischerweise dokumentieren Entwicklerinnen und Entwickler die Anforderungen in einem Tool wie Jira und brechen sie in Storys herunter. Zudem versehen diejenigen, die die Anforderungen implementieren, jede committete Codeänderung mit einer Referenz zur Jira-Story. Damit ist jede Codeänderung mit einer Anforderung verknüpft.

Im nächsten Schritt stellen Entwickler einen Pull Request, um die Änderung in die Codebasis zu übernehmen. Der Schritt garantiert, dass mindestens eine Person jede Codeänderung überprüft, bevor die Änderungen in den main-Branch gemergt werden. Als Teil des Pull Request erfolgt eine statische Codeanalyse und ein SAST-Check (Static Application Security Testing). Nur wenn diese Prozessschritte die geforderten Resultate liefern, setzt sich der Softwarelieferprozess fort. Der nächste Schritt baut die Software und führt als Teil des Builds die Unit-Tests aus.

Das Resultat dieser Tests wird zentral gespeichert. Sind die Tests erfolgreich durchgelaufen, wird das generierte Artefakt signiert und im Artefaktrepository abgelegt. Der Build-Prozess prüft auch, ob es Sicherheitsprobleme in den Dependencies, also den verwendeten Bibliotheken, gibt. Ist dies nicht der Fall, wird das Scanergebnis protokolliert.

Quality Gates lenken und messen Qualität

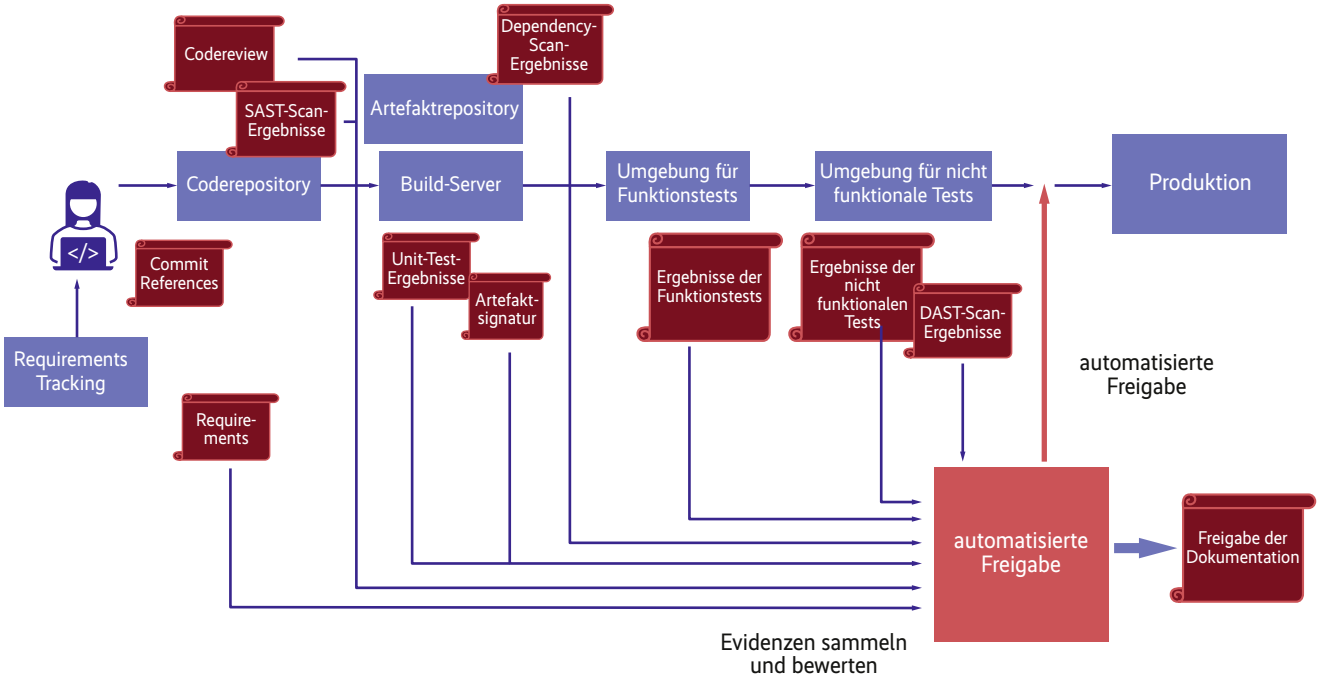
Die Pipeline wird nun fortgesetzt und das Artefakt in den folgenden Stages der Pipeline, beispielsweise in einer Testumgebung, installiert. Hier führt der Prozess dann die Regressionstests aus, idealerweise auch Performance- und andere Securitytests. Wird ein bestimmtes Quality Gate, also eine Qualitätshürde, nicht genommen, bricht der Prozess ab. Alle Ergebnisse der Prozessschritte werden protokolliert und abgelegt – sie sind die Evidenz für die erforderliche Compliance.

Sind alle definierten Schritte durchlaufen, wird aus den unterschiedlichen Systemen die Evidenz über APIs abgegriffen und in einem finalen Report konsolidiert. So kann nach einem finalen Approval (oder sogar ohne explizite Freigabe – alle definierten Qualitätskriterien sind erfüllt) die Applikation in Produktion gehen, nachdem die Signatur überprüft ist.

Quality Gates entscheiden also, ob der Softwareauslieferungsprozess abzubrechen ist. Idealerweise lassen sie sich als Konfigurationsdaten beschreiben. Strukturierte Formate wie etwa YAML garantieren gute Lesbarkeit auch für Nichtentwickler. Diese Konfigurationen werden in einem Codeversionierungssystem gespeichert, jede Änderung ist damit transparent nachvollziehbar. Sinnvoll ist es auch, dass die Schreibrechte für diese Konfigurationsdaten nicht bei den Entwicklungsteams liegen, sondern bei der Qualitäts- oder Risikoabteilung des Unternehmens. Mögliche Beispiele für solche Konfigurationsdaten sind etwa eine Mindestanzahl von Reviewern, eine Untergrenze für die Testabdeckung oder das Fehlen kritischer Vulnerabilities bei der statischen Codeanalyse.

Eine solche Konfiguration der Berechtigungen trennt die Verantwortlichkeiten: Die Risikoabteilung definiert in Kenntnis der regulatorischen Vorgaben die Regeln, die Entwicklungsteams müssen sie einhalten. Verletzen sie die Regeln, ist garantiert, dass die Software nicht die Produktion erreicht.

Während verletzte Compliancekriterien zu einem direkten Abbruch der Pipeline führen, gibt es weitere vor jeder Aus-



Der auf Prozessevidenzen basierende automatische Freigabeprozess gewährleistet hohe Qualität (Abb. 2).

lieferung der Software zu überprüfende Freigabekriterien. Sie reichen von Fragen des Projektmanagements (Gibt es offene Risiken?) über die Entwicklung (Sind alle geplanten Softwaretests durchgeführt?) und Softwarequalität (Gibt es Compilerwarnungen?) zu Fragen der Rechts (Wurde eine Open-Source-Software-Lizenzbewertung durchgeführt?).

Eine Softwarefreigabe in der Automobilindustrie besteht aus etwa hundert funktionalen und nicht funktionalen Kriterien. Die Sammlung der relevanten Prüfdaten und Nachweise erfolgt häufig in MS Excel und ist für jede Freigabe zu wiederholen. Ein interdisziplinäres Gremium mit Vertretern und Vertreterinnen des Qualitätsmanagements, der Entwicklung und des Kunden bewertet die Freigabefähigkeit. Jedes Freigabekriterium erhält eine Ampelfarbenbewertung: Rot unterbindet die Freigabe, Gelb ermöglicht sie mit Auflagen und Grün führt zu einer direkten Freigabe. Abzuarbeitende Maßnahmen, um Auflagen zu erfüllen, prüft das Gremium bei einem nachfolgenden Termin.

Der komplexe und strenge Softwarefreigabeprozess stellt sicher, dass die notwendigen Prozessschritte eingehalten werden, und hilft damit, hohe Qualität zu garantieren. Insbesondere verhindert er negative Auswirkungen wie Personen- oder Sachschäden durch den Einsatz der Software und trägt den Anforderungen zur Produkthaftung Rechnung. Allerdings fordern die Kunden, wie bereits erwähnt, eine rasche, hochfrequente Softwareauslieferung zu wettbewerbsfähigen Kosten. Damit muss das Augenmerk auch darauf liegen, die Softwarefreigabe zu automatisieren.

Automatisierung der Freigabe von Software

Die Automatisierung der Softwarefreigabe erfolgt in drei Schritten. Der erste Schritt sammelt die bei der Ausführung der einzelnen Prozessschritte generierten Evidenzen. Dazu spricht der Prozess Softwareentwicklertools wie GitHub und Projektmanagementtools wie Jira über APIs an. Andere mögliche Evidenzen sind beispielsweise Dateien im CVS-, JSON- oder YAML-Format, die versioniert im Artefaktrepository abgespeichert werden. Die unterstützten Dokumentformate hängen nur von der Implementierung des Automatisierungstools ab; so sind Metadaten von SharePoint-Files (Datenablage) oder Inhalte von Dokumenten in MS Word oder im PDF-Format genauso möglich.

Der nächste Schritt bewertet die Daten und ermittelt damit den Erfüllungsgrad der Freigabekriterien. So lassen sich beispielsweise Testreports und digitale Unterschriften in PDF-Dokumenten analysieren. Der abschließende Schritt schließlich aggregiert die Evidenzen und ihre Bewertung zu einem Bericht; der Freigabestatus ist als PDF-Dokument oder Website dargestellt. Dabei werden die zur Bewertung herangezogenen Evidenzen verlinkt.

Das von den Autoren in ihren Unternehmen eingesetzte Softwareprodukt ist modular und in allen drei Punkten erweiterbar. Es lassen sich zusätzliche Tools leicht einbinden, zusätzliche Freigabekriterien bewerten und Reports anpassen. Das Resultat des automatisierten Freigabeprozesses ist eine transparente, regelbasierte Entscheidung mit vollständiger Dokumentation inklusive aller Quellen. Eine Freigabesitzung muss nur noch bei Bedarf stattfinden.

Mit Feingefühl herangehen: Verpflichtung oder nicht?

Die Autonomie der Teams ist in der agilen Welt ein hohes Gut. Deshalb sollten Unternehmen strenge Vorgaben zum Tooling und zum Tech-Stack nur mit Bedacht einführen. Die effektivste Art, ohne Vorschriften eine Vereinheitlichung zu erreichen, ist, eine zentralisierte Lösung anzubieten. Sie sollte auf komfortable Weise die regulatorischen Vorschriften abdecken und somit die Entwicklungsteams spürbar entlasten.

Idealerweise ist die Complianceautomatisierung eng in die existierende Entwicklertoollandschaft integriert. Die IT-Sicherheits- und Risikoabteilungen des Unternehmens müssen sie abnehmen. Teams können sich darauf verlassen, dass sie alle erforderlichen Auflagen erfüllen, wenn sie die zentralisierte Plattform nutzen.

Die Erfahrung zeigt, dass Teams solch ein Angebot gut annehmen und sich eine

Vereinheitlichung über die Zeit von selbst ergibt. Man spricht hier von einem Golden Path, einem komfortablen goldenen Weg, der für Teams der bequemste ist und sie von Aufgaben, die nicht unmittelbar Business Value erzeugen, entlastet.

Es wird immer Teams geben, die die regulatorischen Vorgaben auf ihre eigene Weise lösen. Das ist auch in Ordnung, solange die Compliance gewährleistet bleibt. Hier stellt sich schnell heraus, dass der Complianceaufwand hoch ist, sofern jedes Team ihn selbst leisten muss. Er schlägt sich auch in den Kosten nieder. Und damit gibt es überzeugende Argumente auch aus geschäftlicher Sicht, eine zentralisierte, gut gemanagte Plattform einzuführen. Auch das ist ein wichtiger Faktor bei der Akzeptanz durch die Teams. Wie immer ist aber bei zentralen Plattformen darauf zu achten, Abhängigkeiten so weit wie möglich zu vermeiden [1].

Fazit

Moderne Softwareentwicklungsmethoden und dazugehörende Werkzeuge haben durch effektive Automatisierung ein hohes Innovationstempo ermöglicht. Die Prozessabläufe und die dabei generierten Daten sind als Basis für Softwarefreigabeentscheidungen verwendbar. Dazu ist es nötig, viele generierte Daten aus unterschiedlichen Systemen zu aggregieren und zu überprüfen. Mit Regeln, die auf den regulatorischen Anforderungen basieren, lässt sich somit oft automatisiert entscheiden, ob Änderungen an einer Anwendung die notwendigen Qualitätsanforderungen erfüllen.

Beim Implementieren und dem Design einer solchen Lösung ist eine enge und frühzeitige Abstimmung mit Security- und Risk-Abteilungen notwendig. Nur dann kann der automatisierte Prozess die Anforderungen aller Stakeholder erfüllen. Wichtig ist aber auch, dass Teams durch die Automatisierung compliancerelevanter Prozesse tatsächlich von den Tätigkeiten entlastet werden, die sie als nicht besonders erfüllend wahrnehmen.

Der Aufwand, diese Prozesse zu automatisieren, ist gut investiert. In Projekten ließ sich bereits ein Return on Investment von weniger als einem Jahr nachweisen. Schnellere Entwicklungszyklen, höhere Compliance und zufriedenere Mitarbeiter ergeben eine klassische Win-Win-Situation. (nb@ix.de)

Quellen

[1] Robert Ruzitschka; Internal Developer Platforms; iX 2/2023, S. 112

DR. CHRISTOPH PETERS

ist CEO des Corporate-Start-ups Bosch Software Flow. Sein Team entwickelt Lösungen zur automatisierten Freigabe von Software.

ROBERT RUZITSCHKA

ist DevOps Community Lead bei der Raiffeisen Bank International und beschäftigt sich mit den technischen und organisatorischen Aspekten agiler Softwareentwicklung.