Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Praxistransferbericht

Einbindung von Code Compliance Checks in der SAP Software Entwicklung

Justin Becker

vorgelegt am 8. August 9834

•

Fachbereich Duales Studium Wirtschaft / Technik
Hochschule für Wirtschaft und Recht Berlin

Name:	Justin Becker
Matrikelnummer:	77204368351
Ausbildungsbetrieb:	SAP SE
Fachbereich:	FB2: Duales Studium — Technik
Studienjahrgang:	2023
Studiengang:	Informatik
Betreuerin Unternehmen:	Jenny Peter
Betreuerin Hochschule:	Lara Maria Stricker
Wortanzahl:	2174

Abstract

Das Ziel dieser Arbeit besteht darin den Entwicklungsprozess von SAP Software zu vereinfachen, indem die Dienste des HANA & HANA Cloud Quality Engineering Teams der Commen Service Infrastructre (CSI) Abteilung verbessert werden. Diese Dienste stellen sogenannte Code Compliance, also ob Code den SAP-Richtlinien folgt, fest. Dafür führen sie Checks durch, die einen Scan des Codes und die Bewertung der Ergebnisse des Scans umfassen.

Dieser Bericht baut darauf hinaus, wie man die Bewertung der Scan Ergebnisse auf einer technischen Ebenen so gestaltet, dass sie so streng wie nötig und so tolerant wie möglich sind. Dabei soll die Verantwortung für die Bewertung klar in dafür zuständigen Händen liegen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

JSON Programmiersprachen unabhähniges Format zum Speicher von Daten durch eine Key-Value-Pair Zuweisung

Pipeline Abfolge von Prozessen, die aufeinander aufbauen

Policy As Code Prinzip, Richtlinien als Code mit einer Programmiersprache aufzuschreiben

Public Cloud Cloud, die von mehrern Unternehmen genutzt wird

Software as a Service komplettes Software Programm in der Cloud mit Nutzung über einen Browser

Akronyme

CCC Code Compliance Check

CI/CD Continuous Integration / Continuous Delivery

CSI Commen Service Infrastructre

EU Europäischen Union

HHCQE HANA & HANA Cloud Quality Engineering

OPA Open Policy Agent

SaaS Software as a Service

1 Einleitung

Im Unternehmen SAP SE besteht die Notwendigkeit, dass alle Software-Entwickler die globale Entwicklungsrichtlinie von SAP und insbesondere die in dieser Richtlinie festgelegten Regeln befolgen. Die Einhaltung dieser Regeln und damit die Einhaltung der SAP Global Development-Richtlinie für alle Entwicklungseinheiten ist ungemein wichtig und obligatorisch, um jegliche rechtlichen und finanziellen Risiken, sowie Ansehensverluste für das Unternehmen zu vermeiden.

Die SAP-Abteilung CSI unterstützt Service- und Produktteams beim Programmieren, Versenden und Ausführen ihrer Services und Produkte in der Cloud. Es bietet ein Portfolio von Diensten, die eine Plattform für die Entwicklung, Veröffentlichung und den Betrieb von Cloud-nativen, konformen und produktionsbereiten Diensten und Anwendungen bilden.

Die Dienste zur Überprüfung der SAP-Entwicklungsrichtlinien werden vom Team HANA & HANA Cloud Quality Engineering (HHCQE) entwickelt und betreut. Mit sogenannten Code Compliance Checks (CCCs) werden die Richtlinien in automatisierten Entwicklungs- und Bereitstellungsprozessen überprüft und die SAP konforme Auslieferung neuer oder aktualisierter Software, z.B. zur Fehlerbehebung, gewährleistet.

In diesem Praxistransferbericht wird erarbeitet, wie solche Checks gestaltet werden, damit sie alle genannten Anforderungen erfüllen und gleichzeitig so minimalistisch bleiben, dass kein Entwickler mehr als nötig durch negative Check Ergebnisse aufgehalten wird.

2 Software Entwicklung bei SAP

Um den Umfang der Anforderung an die CCCs zu verstehen, muss erst ein Verständnis für die Entwicklung der SAP Produkte und damit auch für die Produkte selbst entstehen.

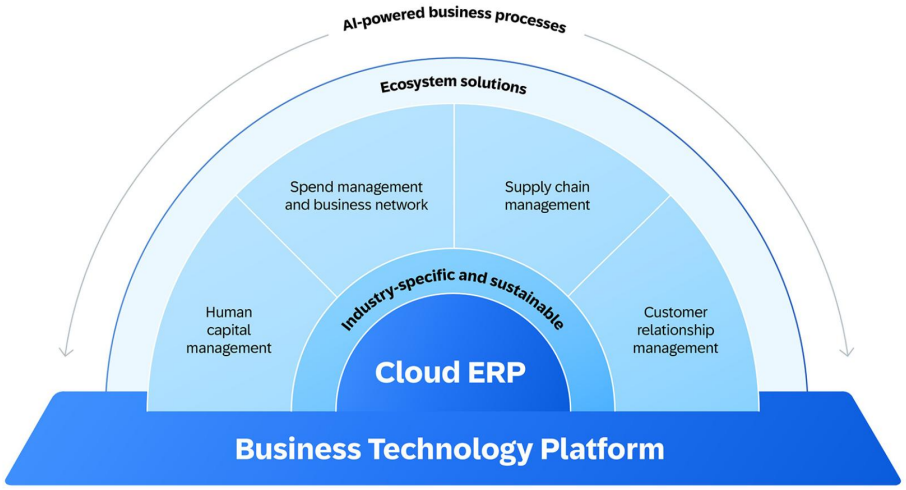

Abbildung 1: Produktportfolio Übersicht von SAP
Quelle: https://www.sap.com/intelligent-enterprise.html?url_id=banner-glo-homepage-row3-cta-who-we-are-230510

Die SAP ist eine der größten Software-Entwicklungsunternehmen weltweit und hat, wie in Abbildung 1 dargestellt, ein sehr breites Produktportfolio für ein sehr breites Kundenfeld [2]. Diese Produkte umfassen Bereich wie Human Resources, Customer Relationship Management, Branchen spezifische Lösungen und noch vieles mehr.

Damit Kunden in der Benutzung eine klare und einheitliche Handhabung erfahren können, muss jedes Produkt als Software Komponente genauso gestaltet sein. Nicht nur das – sämtliche SAP Produkte sollten kongruent zueinander sein, da bei Kunden nicht das Gefühl entstehen darf, sie müssten mit jedem Produkt etwas komplett neuartiges erlernen. Dies würde Kunden davon abhalten in weitere SAP Produkte zu investieren. [**Hier ne passende quelle waere cool**]

2.1 Cloud Produkte

Die Strategie der SAP sieht es vor, dass möglichst viele dieser Produkte in einer Public Cloud als Software as a Service (SaaS) verfügbar sind [5]. Die Entwicklung von Cloud Produkten hat unter anderem den Vorteile, dass Kunden nicht in eigen Hardware und Infrastruktur investieren müssen und damit flexibel bleiben. Da Cloud Produkte meist in einem Abonnement Modell verkauft werden, ist ein Vorteil für die SAP konstante und kalkulierbare Einnahmen. Deshalb bauen aktuelle SAP Produkte, wie in Abbildung 1 zu sehen, auf diesem Prinzip auf. [4]

Damit die Entwicklung der Produkte entsprechend der erklärten Prinzipien abläuft, gibt es Abteilungen, die sich darauf fokussieren den Cloud Entwicklungsprozess zu vereinfachen. So bietet CSI eine einheitliche Infrastruktur für diese Entwicklungsprozess. So kann als Teil dieser Infrastruktur überprüft werden, ob alle Anforderungen an den Code des Produktes erfüllt sind – ob der Code compliant ist.

Umgesetzt werden diese Anforderungen als Teil des Continuous Integration / Continuous Delivery (CI/CD) Prozesses.

2.2 CI/CD

Da die Cloud Produkte komplett von SAP verwaltet werden, ergibt sich der wichtige Vorteil für die SAP und deren Kunden, dass die Entwicklungsteams bei SAP den Code von Produkten jederzeit anpassen und an den Kunden ausliefern kann. Somit können schnell Fehler verbessert, Feedback des Kunden umgesetzt und agil gearbeitet werden. So entsteht mehr Produktwert für die Kunden und ihre Zufriedenheit steigt.

Diesen Prozess wird CI/CD genannt. Jeder Entwickler integriert kontinuierlich neue Änderungen in den Code, welche dann kontinuierlich an den Kunden geliefert werden. Umgesetzt wird dieser Prozess durch die Automatisierung mit sogenannten CI/CD Pipelines. Diese Pipelines übernehmen alle für die Auslieferung nötigen Schritte wie zum Beispiel die Build-, Test- und Deploy-Vorgänge und ermöglichen damit erst CI/CD. [6]

Durch den schnellen Entwicklungsprozess entsteht schnell Code der nicht mehr compliant ist. Deshalb ist es ein unerlässlicher Schritt die Code Compliance vor dem

Ausliefern als Teil der CI/CD Pipelines festzustellen. Deshalb werden CCCs als eben solch ein Schritt eingeführt.

3 Code Compliance

In den CCCs geht es darum - wie durch die Bezeichnung bereits vorweggenommen - die Compliance von Code festzustellen. Allerdings wurde noch nicht geklärt, was damit gemeint ist.

Wenn Code compliant ist, heißt das, dass der Code alle ihn betreffenden Richtlinien erfüllt. Das können Richtlinien von Staatlichen Institutionen wie der EU oder Deutschland sein, Richtlinien, die für das ganze Unternehmen, hier SAP, gelten, aber auch Richtlinien die nur für bestimmte Abteilungen gelten.

Deshalb ist für diese Arbeit interessant, wie die Richtlinien von SAP genutzt werden können. Diese umfassen unter anderem Regeln zum Qualitätsmanagement, zur Export Kontrollklassifizierung, Opensource Lizenzvalidierung und diverse Überprüfungen der Sicherheitsanforderungen.

3.1 Beispiel

Ein konkretes Beispiel Code der ein Sicherheitsrisiko darstellt und damit eventuell für nicht Compliance verantwortlich ist, ist die in Abbildung 2 dargestellte Funktion.

```
def readFile(filePath):
    fileObj = open(filePath, "rb")
    return fileObj.readlines()
```

Abbildung 2: Python Risikocode, da eine geöffnete Datei nicht richtig geschlossen wird (selbst geschrieben)

In dem Beispiel wird mit Python eine Datei geöffnet und ihr Inhalt gelesen. Die Datei wird allerdings nicht geschlossen. Solche Fehler könnten zu unerwartetem Verhalten während der Laufzeit führen. Deshalb müssen sie bereits davor erkannt werden. Da eine Überprüfung per Hand fehleranfällig und langsam ist, braucht es hierfür eine Automatisierung. Diese Automatisierung wird Check genannt.

3.2 Checks

Damit ergibt sich als Ziel von den CCCs in den CI/CD Pipelines, dass sie das Ergebnis liefern sollen, ob der Code, der mit der Pipeline ausgeliefert werden soll, allen gegebenen Richtlinien folgt und damit compliant ist. Außerdem müssen sie eine Begründung für diese Entscheidung an die Entwickler liefern, damit dieser seinen Code entsprechend verbessern kann.

Dafür sind CCCs in zwei Teile aufgeteilt:

- ein Scan, der den Code analysiert und feststellt welche Richtlinien relevante Risikostellen existieren
- die Bewertung von den Ergebnissen diesen Scans nach den Richtlinien

Da die Richtlinien Anforderungen unterschiedlichster Themenbereich wie zum Beispiel von der Export Kontrollklassifizierung bis zu Sicherheit enthalten, wird die Gesamtfeststellung - compliant oder nicht - in mehrere Teilchecks unterteilt, sodass solch ein Check einen Aspekt der Richtlinien übernimmt.

3.2.1 Scans

Das Scannen von Code wird von externen Tools übernommen. Zum aufrufen dieser Tools wird eine interne Version des Project Piper genutzt. Piper vereinfacht unterschiedlichste Schritte, die in einer Pipeline ausgeführt werden können. So zum Beispiel auch das aufrufen von eben solchen externen Scan Tools.

Es gibt zum Beispiel in dem Bereich der Sicherheitsanalyse von Code eine Reihe geeigneterTools wie Checkmarx oder Fortify, die unter anderem in dem HHCQE Team genutzt werden [3]. Solche Tools würden ein Risikostelle wie in Abbildung 2 erkennen und als „Improper_Resource_Shutdown_or_Release“ einordnen [1].

Dann wird von dem Scan eine Datei erzeugt, die enthält welche Risikostellen gefunden wurden. Zudem ist in solcher Datei meist eine Priorisierung, wie schwerwiegend die Risikostellen sind, enthalten.

Diese Datei stellt nun das Ergebnis des Scans da und liegt im JSON Format vor.

Außerdem besitzen externe Tools wie Checkmarx oft eine Weboberfläche, in der Details eines Scans mit den genauen Zeilen an Code, die Risikostellen sind, eingesehen werden können. In diesen Rahmen gibt es dann die Möglichkeit diese Risikostellen als unbedenklich zu markieren. Richtlinien beziehen sich oft darauf, wie viele von den Risikostellen als unbedenklich eingestuft werden müssen.

3.2.2 Scan Ergebnis Bewertung

In der Bewertung der Ergebnisse muss nun auf Grundlage der erzeugten Datei die Entscheidung getroffen, ob der Code compliant oder nicht ist. Bisher wurde diese Entscheidung nicht eindeutig nach den Richtlinien getroffen. Stattdessen wurde sie weitgehend von Piper übernommen. Problem ist, dass dieser Prozess mit Piper nicht durchsichtig ist und keine Verantwortungen abgeklärt sind. Es bedarf einer Lösung, die flexibel, falls sich die Anforderungen an Code und damit die Richtlinien ändern, und nachvollziehbar, sodass immer klar ist auf welcher Grundlage die Entscheidung getroffen wurde, ist.

Von der SAP Abteilung Hyperspace werden seit neusten Richtlinien in einem Policy As Code Format bereitgestellt. Diese Richtlinien sind in der sogenannten Rego Programmiersprache formuliert, die mit dem Open Policy Agent (OPA) angewandt werden können. Dieser OPA wird von Hyperspace genutzt.

Abbildung 3: Beispiel Rego Script mit Ein- und Ausgabe
(selbst geschrieben)

Das in Abbildung 3 zu sehende Script ist immer eindeutig, transparent und schnell anzupassen. Die verwendete Rego Programmiersprache ist außerdem dafür optimiert Richtlinien anzuwenden. Damit is der OPA mit Rego für die genannten Anforderungen geeignet. [7]

Genutzt werden können diese Richtlinien von Hyperspace in zwei Schritten.

Als erstes wird die Ergebnisdatei des Scans auf einen Server von Hyperspace hochgeladen. In dem Upload ist noch eine sogenannter Policy Key (Deutsch: Richtlinien Schlüssel) enthalten, der eindeutig zuordnet, welche Richtlinie für das Ergebnis genutzt werden soll. So können dann die Ergebnisse des Scans bei Hyperspace mit Hilfe von OPA ausgewertet werden.

Im zweiten Schritt müssen die Auswertungsergebnisse dann innerhalb der Pipeline nutzbar gemacht werden. Dafür wird eine Anfrage an den Hyperspace Server geschickt, der eine Datei mit der Auswertung zurückgibt. Diese Datei enthält im JSON Format die Information, ob der gescannte Code der Richtlinie mit gegebenen Policy Key entspricht oder nicht.

Da die Evaluation auf dem Hyperspace Server asynchron läuft und nicht klar ist, wann die Auswertung zur Verfügung steht, wird ein Fallback-Mechanismus implementiert, der diese zweite Anfrage in größer werdenden Zeitabständen bis zu fünf Mal wiederholt. Wenn es nach dem fünften Versuch kein Ergebnis gibt, bricht die Bewertung ab und das Check Gesamtergebnis wird als unbekannt gesetzt.

Damit für den Endbenutzer dieser Entscheidungsfindungsprozess nachvollziehbar bleibt, werden mit einer dritten Anfrage an den Hyperspace Server Details zu der Auswertung abgerufen. Diese Details werden dem Benutzer zusätzlich zu dem eigentlichen Auswertungsergebnis - compliant oder nicht - angezeigt. Sie enthalten Informationen wie den Namen der Richtlinie, eine Kurzbeschreibung, die URL zu der genutzten Richtlinie im Policy As Code Format sowie noch einiges mehr.

4 Herausforderungen

Während der Erarbeitung von diesem Projekt treten immer wieder Probleme auf, die diese Arbeit wohl erst interessant machen. Neben alltägliche Problemen mit dem Code gibt es zwei Herausforderungen, die besonders Lehrreich waren.

4.1 Externe Tools

Im Verlauf der Praxisphase wurde mit einer breiten Auswahl an Tools von anderen Teams gearbeitet. Dazu gehören zum Beispiel das Project Piper, interne GitHub Varianten und unter anderem auch die Richtlinien, die von dem Hyperspace Team zur Verfügung gestellt wurden.

Zu Beginn der Praxisphase wurde ein Plan zur Nutzung der Hyperspace Richtlinien aufgestellt, der da war, dass die Evaluation der Scan Ergebnisse direkt in der Pipeline stattfindet und nicht bei Hyperspace. Dafür sollten die Richtlinien im Policy As Code Format von einem Hyperspace GitHub Repository in den Pipeline Workspace heruntergeladen werden. Danach wird die Ergebnisdatei des Scans mit OPA gegen die heruntergeladene Richtlinie local evaluiert. Dieser Vorgang hat den Vorteil, dass die Auswertungsergebnisse zuverlässig und schnell verfügbar sind.

In der Vorstellung dieses Prozesses nach guten zwei Wochen Entwicklung der Anwendung für die neu verfügbaren Policy As Code Richtlinien gegenüber Hyperspace, wurde vom Hyperspace Team angemerkt, dass diese Idee nicht skalieren wird. Grund dafür ist, dass nicht alle Richtlinien wie angenommen in einem GitHub Repository zu Verfügung stehen werden, sonder verteilt sind. So gibt es keinen zentralen Anlaufpunkt von dem man Richtlinien herunterladen könnte.

Deshalb wurde dann der in diesem Bericht unter Kapitel 3.2.2 vorgestellte Lösungsansatz umgesetzt. Mitzunehmen aus dieser Erfahrung ist, dass man bei der Nutzung von externen Tools möglichst viel über dessen Funktionsweise herausfinden sollte. Direkte Kommunikation und am Besten Absprache mit den Teams über die Nutzung kann viel Zeit sparen und ist ein gute Weg dafür.

4.2 Code Ausführung in Pipelines

Bei der Entwicklung von dem Feature wurde viel lokal getestet, da Tests in Pipelines viel Zeit kosten. So war eine Herausforderungen, dass sich die geschriebenen Code Scripte anders in den beiden Umgebungen verhalten.

```
String resultAsText = new File(POLICY_COLLECT_RESULT_FILE_PATH).getText()
```

Abbildung 4: Beispiel Script für Code, der nicht in einer Pipeline funktioniert (selbst geschrieben)

In dem Beispiel Script in Abbildung 4 ist zu sehen, wie mit einer standard Java Funktionalität die Auswertungsergebnisdatei gelesen wird. Dieses Implementierung funktioniert lokal problemlos, scheitert aber in einer Pipeline, da diese Java Funktionalität nicht in der Pipeline Umgebung verfügbar ist.

```
String resultAsText = PipelineEnvironment.instance.getScript().readFile([file: POLICY_COLLECT_RESULT_FILE_PATH])
```

Abbildung 5: Beispiel Script für Code, der Pipeline Funktionalität benutzt (selbst geschrieben)

Um dieses Problem zu umgehen, stehen dafür Funktionalitäten von der Pipeline Umgebung zu Verfügung. Der Aufruf aus Abbildung 5 zeigt wie so eine Funktionalität benutzt werden kann. Bei dieser Implementierung ist zu beachten, dass sie vorerst nur in der Pipeline funktioniert. Lokal besteht die Möglichkeit bei diesem Aufruf im Hintergrund den Code aus Abbildung 4 auszuführen, damit auch weiterhin getestet werden kann. Alle Code Abschnitte, die mit Dateien arbeiten, mussten entsprechend angepasst werden.

Durch diese Herausforderung ist das Verständnis für die Funktionsweise von Pipelines und das Bewusstsein für die Bedeutung der Umgebung, in der gearbeitet wird, gewachsen.

5 Fazit

Abschließend lässt sich festhalten, dass die umgesetzte Implementierung alle erarbeiteten Anforderungen erfüllt. D.h., dass innerhalb einer Pipeline entschieden werden kann, ob Code compliant oder nicht ist. Der Entscheidungsfindungsprozess wird durch von dafür Verantwortlichen in dem unmissverständlichen Policy As Code Format deutlich gemacht. Für tiefere Einblicke kann in den genutzten Scan Tools eine Weboberfläche genutzt werden, die alle Risikostellen dokumentiert und einordnet.

Trotzdem bleiben noch Fragen offen. Was machen die Teams die eigene strengere oder laschere Anforderungen haben? Gibt es auch Richtlinien, die man gar nicht in einem Policy As Code Format darlegen kann?

Literatur

[1] Dharmendra Ahuja. *Python Security Made Easy: Fixing Security Vulnerabilities with Checkmarx !* 2023. URL: https://medium.com/@dharmendradevops11/python-security-made-easy-fixing-security-vulnerabilities-with-checkmarx-aa7b3ff28b7a (besucht am 19. 03. 2024).

[2] SAP Globale Corporate Cummunications. *SAP Corporate Fact Sheet*. Techn. Ber. SAP SE, 2024. URL: https://www.sap.com/documents/2017/04/4666ecdd-b67c-0010-82c7-eda71af511fa.html.

[3] Florian Maler John Breeden. *Die besten DAST- und SAST-Tools*. 2024. URL: https://www.csoonline.com/de/a/die-besten-dast-und-sast-tools,3673878 (besucht am 19. 03. 2024).

[4] Sebastian Minnich. *Die Vorteile und Nachteile des Cloud-Computing*. 2017. URL: https://www.heise.de/download/blog/Die-Vorteile-und-Nachteile-des-Cloud-Computing-3713041 (besucht am 17. 03. 2024).

[5] o.V. *SAP Cloud Strategy*. URL: https://sapinsider.org/topic/sap-platform-technology/sap-cloud-strategy/ (besucht am 17. 03. 2024).

[6] o.V. *What is CI/CD?* 2023. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd (besucht am 17. 03. 2024).

[7] *Policy Language*. URL: https://www.openpolicyagent.org/docs/latest/policy-language/ (besucht am 19. 03. 2024).

6 Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich:

1. dass ich meine Bachelor-Thesis selbstständig verfasst habe,

2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,

3. dass ich meine Bachelor-Thesis bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum	Justin Becker