Fakultät für Informatik

Studiengang Informatik

# DevSecOps mit Azure DevOps Pipelines und GitHub Actions

Projektarbeit im Fachwissenschaftlichen Wahlpflichtmodul DevOps

von

Florian Weidner

Datum der Abgabe: 06.07.2022
Prüfer: Daniel Kerschagl

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

IT-Sicherheit wird in der Software Entwicklung ein immer wichtigeres Thema. In Abbildung 1.1 ist der Anstieg der Ausgaben für IT-Sicherheit zu sehen. Dieser ist in den vergangen Jahren kontinuierlich gestiegen und auch in der Zukunft sollen die Ausgaben im Jahr 2025 im Vergleich zu 2020 um 57.5% steigen. Das zeigt, dass vielen Unternehmen die IT-Sicherheit immer wichtiger wird. Abbildung B.1 zeigt das IT-Projekte eine große Priorität bei Unternehmen hat. Die dafür anfallenden Kosten sollen aber natürlich trotzdem immer möglichst gering gehalten werden.

DevOps ist mittlerweile eine verbreitete und viel genutzte Entwicklungsmethode (siehe Abb. B.2) um den Entwicklungsprozess zu beschleunigen und somit auch Kosten zu sparen. Um den IT-Sicherheitsaspekt einfach in die Softwareentwicklung zu integrieren wurde so DevSecOps erschaffen. Der Prozess vereint DevOps mit Sicherheitsverfahren. [Ibr22] Zwei der größten DevOps-Plattformen sind GitHub und Azure DevOps. GitHub liefert GitHub Actions und Azure DevOps bietet Azure Pipelines um Operationen aus dem DevOps Prozess zu automatisieren. Diese Frage ist welches Tool den DevSecOps Prozess am besten integriert um sichere Software zu erstellen zu können.

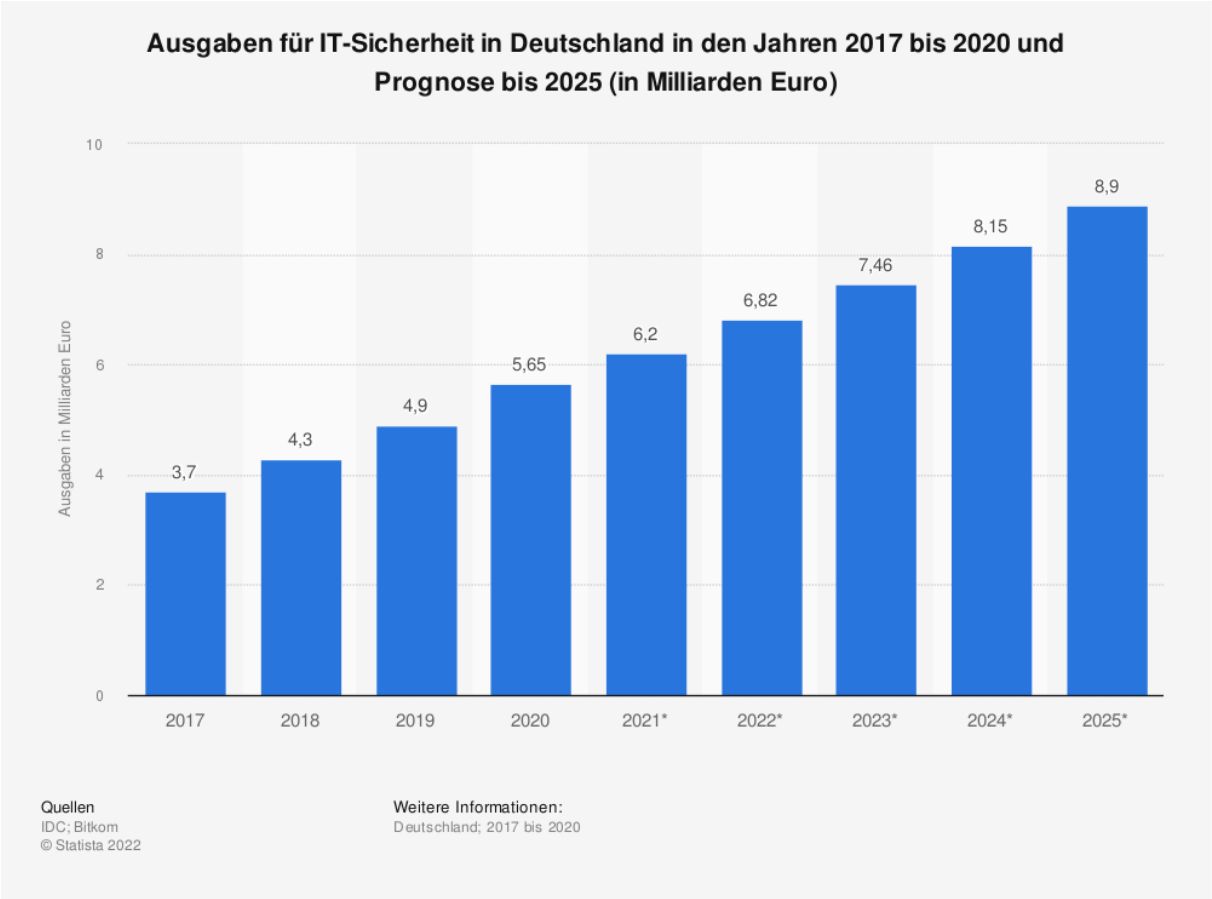


**Abbildung 1.1** Ausgaben für IT-Sicherheit in Deutschland in den Jahren 2017 bis 2020 und Prognose bis 2025 (in Milliarden Euro)

# 2 DevSecOps Allgemein

## 2.1 Zusammenfassung DevOps

DevOps setzt sich aus den Begriffen „Development "(Entwicklung) und „Operations "(Vorgänge) zusammen. Die traditionelle Trennung von Entwicklung und Softwarebetrieb führt oft zu Interessenskonflikten. Entwickler wollen stetig die Software verbessern, der Betrieb hingegen will Änderungen vermeiden um die Stabilität des System zu gewährleisten. [Bee21] Beim Einsetzten von DevOps wird das Entwickeln und der Betrieb von Software näher zusammengeführt. Es entsteht ein Softwareentwicklungs-Prozess, in dem man durch Automatisierungen das Bauen, Testen und Bereitstellen von Software beschleunigen will. Dies erreicht man durch Praktiken wie Continous Integration, Continous Delivery, Continous Deployment, automatisiertes Teste, Infrastructure-as-Code und automatische Veröffentlichungen. Durch die automatische Bereitstellung der Anwendung und der Infrastruktur entsteht eine schnellere Bereitstellung, bessere Softwarequalität und teilweise auch schon mehr IT-Sicherheit. [Mac20] Außerdem steht DevOps auch für offene Zusammenarbeit, Kommunikation, Transparenz, Eingestehen von

Fehlern und das gemeinsame lösen von Problemen, um Konflikte im Team zu vermeiden. Man will schnelles Feedback ermöglichen und somit das Risiko der Softwareentwicklung minimieren. DevOps beschränkt sich also nicht nur auf technische Hilfsmittel sondern liefert eine Kultur um den Entwicklungsprozess immer weiter zu verbessern.[Bee21]

## 2.2 Sicherheitsrisiken und Chancen mit DevOps

DevOps alleine bringt einige Herausforderungen zum Thema Softwaresicherheit mit sich. Oft wird Sicherheit für Entwicklungsgeschwindigkeit aufgegeben. Gründe dafür sind, das bei der Umstellung, die bestehende Sicherheitsmethodiken nicht mit DevOps integrierbar oder nicht in den agilen Prozess gepasst haben. Typischerweise werden dann Sicherheitstests erst am Ende mit der fertig entwickelten Software durchgeführt. Dadurch ist das Sicherheitsteam komplett vom DevOps Prozess ausgeschlossen. Außerdem besteht das Risiko, dass dann erkannte Fehler sich nur sehr schwer ausbessern lassen.

Je schneller die Release-Iterationen werden, desto weniger Aufwand wird in die Sicherheit gesteckt. Da mit DevOps geänderter Code auch schnell produktiv eingesetzt wird, hat das Entwicklungsteam meistens zu wenig Zeit um die Änderungen sicherheitsrelevant zu prüfen. Dies bietet dann eventuell neue Angriffsmöglichkeiten oder Risiken in der Software. Ein weiteres Risiko ist, dass das bei DevOps standardmäßige Einsetzten von Cloud Computing neue Sicherheitsmaßnahmen fordert, welche zusätzlich berücksichtigt werden müssen.

DevOps liefert aber auch Chancen um IT-Sicherheit in den Entwicklungsprozess zu integrieren. Feste, Zentrale und Standardisierte Bereitstellungspipelines helfen dem Security Team einen besseren Blick über die Anwendung und wie diese erstellt wird zu bekommen. Hier können dann verschieden Sicherheitsaspekte in die Pipeline eingebaut werden. DevOps bietet hier also die Grundlage IT-Sicherheit schon in den DevOps Prozess zu integrieren. [Mao20]

## 2.3 Definition von DevSecOps

Der Begriff DevSecOps baut auf den Prinzipien und Praktiken von DevOps auf und fügt den Sicherheitsaspekt in der Entwicklung noch weiter in den Vordergrund. Es ist also eine Erweiterung des DevOps Prozesses. Durch DevSecOps soll IT-Security schon vom Start eines Projektes mit geplant und auch in Bereitstellungsprozess integriert werden. Allerdings reicht es nicht, die Sicherheitsaspekte nur in frühere Phasen der Entwicklung zu verschieben. Es soll eine durchgängige Sicherheit durch ständiges Dazulernen und Verbessern des Sicherheitsprozesses im ganzen Projekt und dessen Bereitstellung entstehen. [Mao20] Am Ende ist also das ganze Team für die Applikationssicherheit verantwortlich und Probleme sollen auch im Team angegangen werden. [Ahm19]

Eine technologische Rahmenbedingungen für DevSecOps ist das Bedürfnis für die Automatisierung von Sicherheitsaufgaben im Entwicklungsprozess. Das Einbauen von Sicherheitschecks in die Bereitstellungspipelines reduzieren Zeit und Kosten von Fehlern, die sonst manuell gefunden werden müssen. Zusätzlich wird die IT-Sicherheit in den kompletten Lebenszyklus einer Software integriert und somit können auch Entwickler ohne viel Wissen über IT-Sicherheit einfach die automatischen Pipelines zur Sicherheitsanalyse nutzen. Das selbe gilt auch für die Bereitstellung der Infrastruktur. Infrastrucutre as Code muss auch sicher und zuverlässig

bereitgestellt werden, um auch hier Sicherheitslücken zu vermeiden. [Mao20].

## 2.4 Praktiken von DevSecOps

DevSecOps muss in verschiedenen, einzelnen Prozessen praktiziert werden, um es auf den ganzen Entwicklungsprozess abzubilden zu können. In der Planungsphase müssen schon bei der Risikoanalyse Sicherheitsaspekte berücksichtigt werden und eine passende Strategie für das restliche Projekt erstellt werden. Während der Entwicklung helfen statische Code Analysen und Code Reviews um Probleme zu finden, bevor der Code eingecheckt wurde. Die Produktivität der Entwickler wird dadurch nur minimal eingeschränkt, es können aber schon erste Fehler erkannt und ausgebessert werden. Nachdem der Code eingecheckt wurde sollten beim Erstellen der Software automatisierte Tests laufen, die feststellen ob es Fehler beim Erstellen gibt. Auch Abhängigkeitsanalysen und Unit Tests sollen kritische Sicherheitsprobleme aufdecken und den verantwortlichen Entwickler benachrichtigen. Auch die automatisierte Bereitstellung der Infrastruktur ist ein wichtiger Teil, der gesichert sein muss und auch essenzielle Auswirkungen auf das komplette System hat. Hier sind Themen wie Secret Management, Configuration Management oder Version Control wichtige Aspekte, um die man sich kümmern muss.[Mao20] [Ibr22] entwickelte ein Security Model um Sicherheitspraktiken in die komplette Infrastrukturbereitstellung zu bringen. Nachdem ein statischer Analyse Security Scanner den Terraform Code analysiert hat, wird dieser in einen verschlüsselten Objektspeicher in der Cloud ausgeführt. Hier werden alle benötigten Secure Shell (SSH) Schlüssel automatisch runter geladen. Die von Terraform erstellten IP-Adressen der Infrastruktur werden an Ansible gesendet welches die erstellten Server dann konfiguriert. Am Ende werden alle Geheimnisse von den Servern gelöscht. So entsteht eine sichere Bereitstellung der Infrastruktur, wo der komplette Prozess sicherheitstechnisch abgedeckt wird. Wenn die Applikation dann auf einer Testumgebung installiert wurde sollen statische und dynamische Security Tests laufen. Auch automatisierte Attacken sollten Bestandteil des automatischen Testzyklus sein. Im Live-Betrieb der Software sollen regelmäßig automatische Sicherheitschecks und eine Überwachung des Systems stattfinden, um einen Einblick in den Datenverkehr zu bekommen und eventuell bösartige Benutzerverhalten aufdecken zu können.

DevSecOps wirkt sich auch auf das Personal aus. Es soll in einem Entwicklungsteam immer sogenannte „Security Chanpions“[Mao20] geben. Diese sollen besonders Wissen über IT-Sicherheit und dem Entwicklungszyklus haben, um ein funktionsübergreifendes Team zu schaffen, was auch schon ein wichtiger Punkt von DevOps ist. Auch ein regelmäßiges Training der Entwickler und Bereitstellung wichtiger Tools führen dazu das IT-Sicherheit nicht nur nebenbei läuft, sondern zu einer Geisteshaltung die das ganze Unternehmen durchdringt.

Insgesamt entsteht eine zeitliche Linksverschiebung (Shift Left) der Sicherheitsmaßnahmen im Entwicklungsprozess. IT-Sicherheit soll schon von Anfang an in das Projekt mit einfließen um Fehler und Probleme am Ende zu vermeiden. Dadurch entsteht früh eine hohe Produktqualität. [Lub22]

Es gibt also viele verschiedene Praktiken, die man in die Softwareentwicklung integrieren kann und soll, wenn man DevSecOps integriert haben will.

## 2.5 Zero Trust Prinzip

Ein weiterer wichtiger Security Aspekt sind Berechtigungen. Neben einem schlichten und übersichtlichem Konzept sollen vor allem keine Berechtigungen vergeben werden, die gar nicht gebraucht werden. Im Vergleich zum gängigen „trust but verify“ spielt bei DevSecOps mehr das „Zero Trust“ Prinzip eine wichtige Rolle. Anstatt eine Vertrauensgrenze zu ziehen, überprüft man bei Zero Trust jeden Zugriff und jede Aktion. Es wird also zum Beispiel nicht unterschieden ob sich jemand in oder außerhalb des internen Netzwerks befindet. Vertrauen wird nur durch Authentifizierung, Verifikation und Autorisierung erreicht. Es steht also die Identität einer Person, eines Geräts oder eines Services im Vordergrund. So wird es Angreifern schwerer gemacht, auch innerhalb einem Netzwerk weitere Berechtigungen zu erlangen. [Wyl21]

## 2.6 DevSecOps Engineer als Beruf

DevOps wird oft auch als Jobbeschreibung verstanden. Als „DevOps Engineer“ arbeitet man in einem oder verschiedenen DevOps-Teams und -Abteilungen. Diese haben vor allem die Aufgabe, genannte Praktiken von DevOps durchzuführen. [Mac20] Ein DevSecOps Engineer setzt auf den Beruf eines DevOps Engineers mit dem Fokus auf IT-Security auf. Viele Unternehmen stoßen beim Einführen von DevSecOps in ihren Entwicklungsprozess auf Herausforderungen. Diese sind teilweise hohe Kosten, schon bestehende solide Organisationsstrukturen oder kultureller Widerstand von Entwicklern. Eine Möglichkeit dagegen ist es, einen DevSecOps Spezialisten anzustellen. Dieser kann die Transformation der Arbeitskultur vorantreiben und beschleunigen. Der Experte kann die Einführung verschiedener DevSecOps Tools übernehmen und mit seinem Wissen schneller Sicherheitsintegrationen in zum Beispiel bestehende Pipelines durchführen. [Mao20] Er muss ein tiefes Verständnis mitbringen wie sich IT-Security auf die einzelnen Phasen der Entwicklung und das Endprodukt auswirkt. Seine Aufgaben sind es, die automatisierten Sicherheitspraktiken in die Pipeline zu implementieren und immer wieder zu überprüfen und aktuell zu halten. Man sollte also gut in jeden Schritt der Softwareentwicklung integriert sein. Er muss sich natürlich aber auch mit normalen DevOps Prozessen und Prinzipien auskennen. Außerdem sollte er versuchen die restlichen, skeptischen Teammitglieder überzeugen, dass die neuen Sicherheitspraktiken wichtig sind und die Entwicklungsgeschwindigkeit nicht verlangsamen. Um für einen solchen Job angestellt zu werden helfen verschiedene Zertifikate über DevSecOps Engineering. [Cob19]

# 3 DevSecOps mit Github Actions

## 3.1 Zusammenfassung der Sicherheitsfunktionen

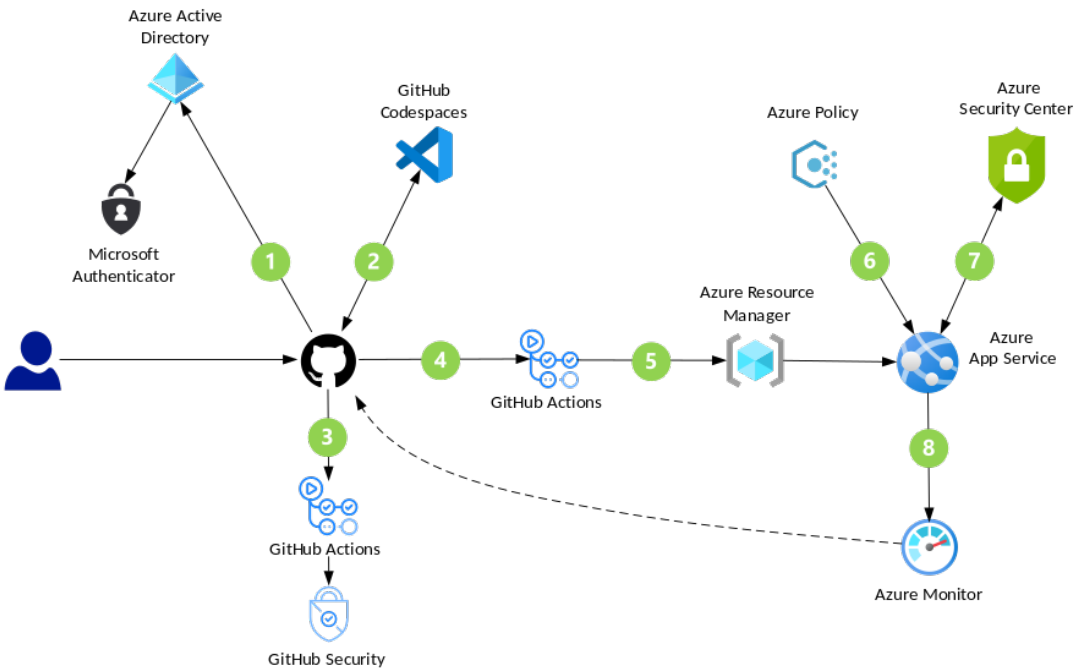


**Abbildung 3.1** Architektur von GitHub für DevSecOps Prozess

GitHub bietet mit GitHub Security ein Konzept DevSecOps abzubilden und in den Entwicklungsprozess zu integrieren. Der Aufbau ist in Abbildung 3.1 zu sehen und die einzelnen Schritte im folgendem Abschnitt grob beschrieben.

Wenn Entwickler auf GitHub Ressourcen zugreifen, müssen sie eine Authentifizierung über eine Azure Active Directory durchlaufen. Hier wird eine sichere, kennwortlose FIDO2-Authentifizierung verwendet, die nach aktuellen Standards entspricht. [Jus] Mit GitHub Codespaces können Entwickler auf vordefinierte Entwicklungsumgebungen in Containern zugreifen, die mit erforderlichen Erweiterungen für Sicherheitsscans ausgestattet sind. Durch GitHub Actions können automatisch Codierungsfehler und Schwachstellen durch automatische Scans beim Einchecken ermittelt werden. Außerdem können die Bildartefakte direkt in einen Azure App Service deployed werden. Durch Pull Requests können weitere automatisierte Tests ausgeführt werden. Außerdem bietet es eine gute Plattform um Code Reviews durchzuführen. Microsoft Defender for Cloud identifiziert Attacken in bereitgestellten Projekten. Durch den Azure Monitor können regelmäßig das Verhalten der laufenden Software überwacht evaluiert und Auffälligkeiten automatisch gemeldet werden.[Mig]

## 3.2 Codespaces

GitHub Codespaces bietet eine Softwareentwicklungsumgebung, die komplett in der Cloud läuft. Visual Studio Code ist so mit Terminal und Debugger im Browser nutzbar. Neben der hohen Skalierbarkeit und Standardisierung der Entwicklungsumgebung bietet das auch Sicherheitsvorteile. Visual Studio Code bietet hier viele Security Scanning Erweiterungen an, die den Entwicklungsprozess sicherer machen. Codespaces bieten Security Logs, in denen festgestellt werden kann, wer wann von wo mit der Entwicklungsumgebung gearbeitet hat. [codc] Außerdem kann man sensitive Daten (Secrets) verschlüsselt und sicher in den Umgebungsvariablen der Codespaces speichern. [codb]

## 3.3 Secret Scanning

Token oder private Schlüssel zur Kommunikation mit externen Services sollten immer geheim gehalten werden. Solche Secrets in einem öffentlichen Repository zu speichern ist demnach fatal für die Sicherheit der Anwendung. Dennoch kann es immer wieder passieren, das Secrets aus versehen eingecheckt werden. GitHub bietet hier ein Secret Scanning über GitHub Actions an. Dieses läuft automatisch bei öffentlichen Repositories und kann mit GitHub Advanced Security noch mit weiteren Secrets-Übereinstimmungsmustern erweitert werden. [sec]

## 3.4 Code Scanning

Automatisierte Security Code Analysen sind ein wichtiger Schritt im DevSecOps Prozess um Schwachstellen möglichst früh zu erkennen. GitHub bietet mit Code Scanning eine Möglichkeit über GitHub Actions verschiedene Workflows zu definieren, die Analysen durchführen. Es gibt im Moment 52 schon vordefinierte Workflows zum Thema Security, die zur Verfügung stehen. Neben dem eigenen Code Analyse Tool CodeQL können auch viele weitere externe Tools eingebunden werden.[codd] CodeQL kann Sicherheitsschwachstellen und generische Softwarefehler für die Sprachen C/C++, C#, Go, Java, JavaScript, Python, Ruby, Typescript identifizieren. Mit der Query Language wird der Code wie Daten behandelt und Sicherheitslücken dann als Abfragen modelliert, welche dann bei der Analyse ausgeführt werden können. Hier gibt es schon standardisierte Abfragen von der GitHub Community, man kann aber auch eigene Abfragen für die Analyse erstellen. [coda]

Alle gefundenen Schwachstellen werden dann im GitHub Repository unter dem Tab Security mit jeweiligen Schweregraden aufgelistet. Man wird direkt zur Codestelle weitergeleitet, die Schwachstelle wird beschrieben und eine Empfehlung abgegeben, um das Problem zu lösen. Es werden auch direkt passende Common Weakness Enumeration (CWE) Vulnerabilities mit angegeben, um weiter über die Schwachstelle Recherchieren zu können. [codd]

Eine weiterer Workflow der durch GitHub Actions zur Verfügung steht ist ein Dependency Review. Dieser Workflow analysiert alle Abhängigkeiten und möglich folgende Sicherheitsauswirkungen. Man hat also nicht nur einen Überblick, wann sich welche Abhängigkeiten ändern sondern bekommt auch direkt Informationen, ob die genutzten Pakete eventuelle Sicherheitsschwachstellen haben. Für alle öffentlichen Repositories wird automatisch bei jedem Pull Request ein Dependency Graph erstellt, der alle Abhängigkeiten mit kompatiblen

Packagemanagement Systemen beinhaltet. Zusätzlich gibt es den GitHub Dependabot. Dieser Analyseservice läuft in jedem öffentlichen Repository wenn man die Funktion aktiviert hat. Er erkennt alle Codeabhängigkeiten mit Sicherheitsschwachstellen. Es werden alle hinzugefügten, geänderten oder gelöschten Abhängigkeiten in einem Pull Request analysiert und dann im Security Tab des GitHub Repositories als Alert bereitgestellt.

## 3.5 DevSecOps Maturity Model

Das Open Web Application Security Project (OWASP) hat das DevSecOps Maturity Model (DSOMM) veröffentlicht, welches den aktuellen Reifegrad der DevSecOps Integration in ein Softwareprojekt angibt. Das Modell ordnet den Stand in vier verschiedene Stufen ein. Mit steigendem Level steigen die praktizierten Sicherheitsmaßnahmen. Es wird zu 18 verschiedenen Dimensionen (siehe Abb. 3.2) definiert, was man für die bestimmten Level umgesetzt haben muss. Es wird also der komplette Entwicklungsprozess abgedeckt. [owa] Mit GitHub Advanced Security lässt sich das DSOMM Level 1 einfach implementieren. Mit den schon genannten Features wie Code Scanning, Dependency graph, Secret Scanning und Anwenden der sonstigen DevSecOps Automatisierungen und einem integrierten, Open Source Dynamic Application Security Testing (DAST) Tool, hat man alle Wichtigen Maßnahmen für Level 1 abgedeckt. Wenn Level 1 erreicht wurde kann man sich dann versuchen Maßnahmen für Level 2 umzusetzen. [Alw20] Das Modell zeigt, das eine Einführung von DevSecOps ein lang laufender Prozess ist, in dem man neue Maßnahmen Schritt für Schritt immer umsetzen sollte. So kann sich auch die DevSecOps Kultur langsam mitentwickeln. Eine abrupte Einführung und Integration kostet viel Aufwand und führt schnell zu Gegenwind im Team.

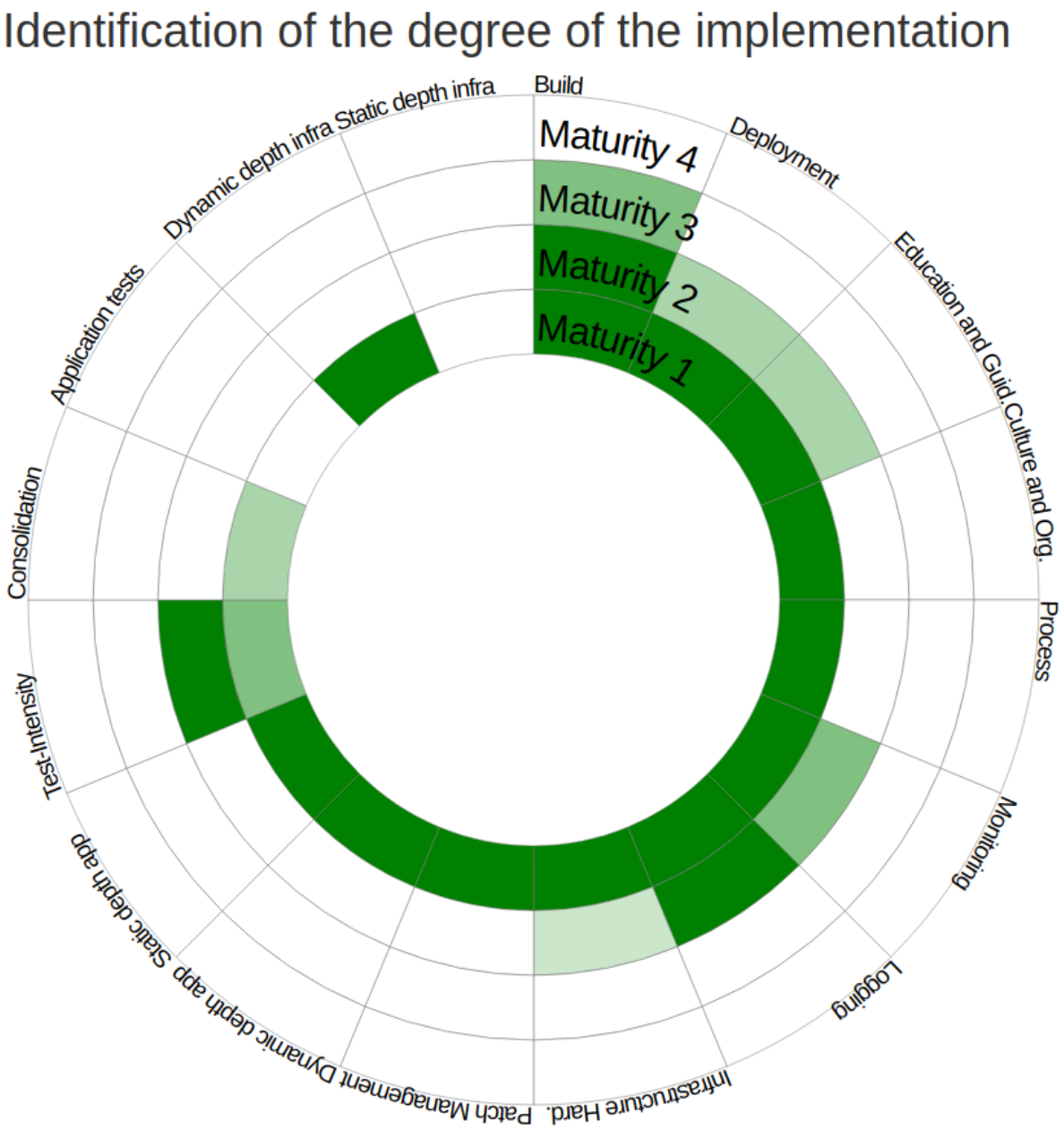


**Abbildung 3.2** OWASP DevSecOps Maturity Model

## 3.6 GitHub Security Lab

Mit dem GitHub Security Lab bietet die Plattform ein Portal, auf dem sie versuchen User für Security zu inspirieren und es ihnen ermöglichen Open Source Software leichter sicher zu implementieren. Auf der Plattform können Schwachstellen von großen, wichtigen Open Source Projekten berichtet werden. Das Portal koordiniert dann die Offenlegung der Schwachstellen mit den Sicherheitsteam der betroffenen Projekte. Es werden zusätzlich verschiedene Analysen mit CodeQL, einer semantischen Code-Analyse-Engine durchgeführt um die Sicherheitsforschung für Open Source Projekte zu verstärken. Zusätzlich werden Common Vulnerabilities und Exposures in einer Datenbank gesammelt. Außerdem werden viele Forschungen, Tutorials, Artikel oder Konferenzen veröffentlicht und der Community bereitgestellt. Es werden auch mit dem so genannten „CodeQL Bug Bounty program“ immer wieder Prämien ausgeschrieben, um die Community zu motivieren Sicherheitsforschung zu betreiben. Die Plattform bietet also

über die technischen Funktionen von GitHub hinaus die Möglichkeit für Entwickler sich über IT-Sicherheit zu vernetzten, über das Thema zu lernen oder sich an der Sicherheitsforschung für Open Source Projekte zu beteiligen. [gitb]

# 4 DevSecOps mit Azure DevOps Pipelines

Die Continous Delivery von Azure DevOps wird durch Azure Pipelines abgebildet. Pull Requests lösen diese aus und führen automatisch CI-Builds und automatisierte Tests durch. Grundlegend sind Azure Pipelines und GitHub Actions sehr ähnlich. Als GitHub von Microsoft 2018 gekauft wurde, entstanden GitHub Actions aus einem Fork der Azure Pipelines. [mica] Azure DevOps bietet ein breites Berechtigungskonzept an. Man sollte nur Berechtigungen für Ressourcen vergeben die benötigt werden und mit Rollen arbeiten, um einzelne Berechtigungen zu vermeiden und den Pflegeaufwand zu minimieren. Auch Pipelines können mit Berechtigungen eingeschränkt werden, sodass sie nur auf bestimmte Umgebungen zugreifen zu können. In Pipelines ist es außerdem möglich Checks zu integrieren. Diese Pausieren die Pipeline bis eine bestimmte Bedingung erfüllt wurde. Das kann zum Beispiel eine manuelle Zustimmung sein, die nötig ist, dass die Pipeline weiterläuft. So kann man garantieren das keine schädlicher Code bereitgestellt werden kann. [mice] Durch Templates können beim Erstellen von neuen Pipelines immer die selbe Struktur sichergestellt werden. So können auch Schutzmaßnahmen in den Templates vordefiniert werden. [micf] Für Azure Pipelines gibt es auch Open Source und lizenzierte DAST Tools, die über den Marketplace als Task in die Pipeline integriert werden können. Für potentielles Monitoring kann der Azure Monitor benutzt werden. Für eine sichere Bereitstellung der Infrastruktur kann in Azure Pipelines Terraform integriert werden. [EdP]

## 4.1 Microsoft Security Code Analysis

Zu Azure Pipelines werden im Moment noch Microsoft Security Code Analysis angeboten, mit der man IT-Sicherheit in die Bereitstellung einfach integrieren kann. Die Erweiterung macht das Einbinden von Security-Tasks simpel und man muss sich nicht um Updates der Tools kümmern. Nach der Installation der benötigten Azure DevOps Extension kann man neue Security-Tasks in die Pipelines hinzufügen. [micd] Es werden verschiedene Tasks angeboten. Der Credential Scanner scannt das Repository nach geheimen Informationen die noch im Repository stehen, wo sie nicht sein sollten. Dazu gibt es noch folgende weitere allgemeine Static Application Security Testing (SAST) Tools die integriert werden können: TSLint, Roslyn Analyzers, BinSkim und Anti-Malware Scanner. Auch die Nachbearbeitung der Analysen können mit der Erweiterung abgebildet werden. Security Analysis Logs können in den Azure Artifacts als .zip gespeichert werden. Ein Security Report erstellt eine Zusammenfassung von allen Security-Tasks die gelaufen sind und fügt diese für bestimmte Problemstufen zusammen. Mit der Post Analysis kann konfiguriert werden, ob der Erstellungsprozess aufgrund gefundener Schwachstellen fehlschlagen soll. Mann kann hier für die einzelnen Tasks oder auch die Problemstufen

unterscheiden. [micc]

Die Erweiterung ist noch bis 31. Dezember 2022 verfügbar. Bestehende Benutzer können diese zwar noch weiter nutzen. Neue Nutzer müssen sich selber um die Erstellung dieser Tasks kümmern oder auf GitHub migrieren. [micc]

# 5 Vergleich von GitHub Actions und Azure DevOps Pipelines

## 5.1 Kosten

Für GitHub Advanced Security ist GitHub Enterprise benötigt. Das kostet pro Benutzer 21$ im Monat. DevOps kostet grundlegend 6$ pro Person und die ersten 5 Nutzer sind gratis. Wenn man hier auch noch Funktionen des Azure Testplans benutzten will, kostet die Lizenz 52$ pro Person. Die Agenten für Pipelines unterscheiden sich bei den Kosten teilweise. GitHub bietet kostenlose gehostete Agenten für öffentlich Repositories an, DevOps allerdings nicht mehr. Bei privaten Repositories bietet GitHub 2.000 freie Minuten im Monat an, man kann diese aber mit GitHub Enterprise auf bis zu 50.000 Minuten erhöhen. DevOps bietet kostenlose 1.800 Minuten an. Bei bezahlten Jobs gibt es hier gar kein Limit mehr. GitHub bietet viel kostenlose für Open Source Repositories an. Um aber den kompletten DevSecOps Prozess gut abzudecken ist GitHub Enterprise nötig. [micb]

## 5.2 Benutzer Berechtigungen

Berechtigungen in Azure DevOps sind sehr vielseitig. Durch ein Aufteilung in Gruppen ist das Verwalten der Berechtigungen überschaubar. Da man Benutzer dann verschiedenen Gruppen zuweisen kann, sind sehr filigrane und spezielle Konfigurationen für viele Benutzer einfach umzusetzen. [Kat] Bei GitHub gibt es für private Repositories nur zwei Zugriffsberechtigungsstufen. Als Besitzer kann man „collaborator“ einladen und deren Berechtigungen steuern. Für Organisationen gibt es allerdings mehr Möglichkeiten. Hier gibt es Grundlegend einen „owner“, „billing manager“ und „member“. Außerdem gibt es dann auch die Möglichkeit über Rollen und Gruppen alle benötigten Berechtigungen zu konfigurieren. [gita] Für den privaten Gebrauch ist man hier also leicht eingeschränkt, allerdings habt man auch selten komplexe Berechtigungsstrukturen. Auf Organisationsebene bieten beide Plattformen allerdings das selbe Konzept an.

## 5.3 Automatisches Security Testing

Bei Azure DevOps muss man für Security Checks in der Build Pipeline auf eine Erweiterung oder andere Drittanbietersoftware wie zum Beispiel SonarQube zugreifen. GitHub bietet beim

Anlegen von GitHub Actions eine Vielzahl von vordefinierten Workflows, die sich um die IT-Sicherheit des Projektes und des Repositories kümmern. So kann man mit sehr wenig Aufwand und Eigenarbeit die Vorlagen Nutzen und zum Beispiel Secret-Scanning oder eine SAST-Analyse integrieren. GitHub bietet für Open Source Repositories hier schon viele kostenlose Möglichkeiten an. Ansonsten ist bei beiden Plattformen automatisches Security Scanning nur mit bezahlter Lizenz möglich. GitHub ist hier mit CodeQL auch viel Communitiyfreundlicher. Jeder Nutzer hat die Möglichkeit eigene CodeQL Queries zu erstellen oder welche von anderen zu Nutzen. [Por21] Wichtig ist allerdings auch , das man die erstellten Pipelines auch beachtet, aktuell hält und überprüft. Ansonsten hilft einem die einfachere Integration auch nichts.

## 5.4 Community Interesse

Seit GitHub 2018 von Microsoft gekauft wurde, wuchs das Community-Interesse an der Plattform nochmal stark an. In der Abbildung B.3 sind die StackOverflow Trends der beiden Portale zu sehen. Die Grafiken zeigen die Anzahl der gestellten Fragen zu den beiden Plattformen und deren CI/CD-Tools. Man erkennt, dass GitHub lange Zeit relevanter war, bis 2017 Azure DevOps an Relevanz gewinnt und GitHub kurzzeitig überholt. Seit der Übernahme von Microsoft hat GitHub allerdings wieder einen starken Zuwachs an gestellten Fragen. Auch die neuen GitHub Actions stoßen auf eine große Relevanz mit vielen Fragen in der Community und überholen die schon länger etablierten Azure Pipelines 2021. GitHub ist insgesamt näher mit der Entwicklungscommunity verbunden als Azure DevOps. Neben der Verbundenheit zu Open Source helfen GitHub Plattformen wie das Security Lab um die Community mehr einzubinden. Azure DevOps ist vor allem für Businesskunden ausgerichtet, und deckt hier teilweise die Anforderungen besser ab.

# 6 Fazit

IT-Sicherheit in den Softwareerstellungsprozess zu integrieren ist essenziell. DevSecOps ist hier eine gute Methode die Software sicherer zu machen, das Team für IT-Sicherheit zu sensibilisieren und auch alle zu integrieren. Es ist aber immer wichtig das Team mit einer plötzlichen Umstellung von vielen Punkten nicht zu überrennen. Die Einführung von DevSecOps hat keine zeitliche Einschränkung und jeder kleine Schritt, der in die richtige Richtung führt, bringt einen näher. So kann man Schritt für Schritt jeden im Team von DevSecOpos überzeugen und erfolgreich sichere Software erstellen.

Grundsätzlich ist das mit beiden Plattformen möglich. GitHub bietet insgesamt mit GitHub Advanced Security mehr benutzbare Funktionen, die einem das Einführen von DevSecOps erleichtern. Da die Einführung generell eine große Herausforderung ist, bekommt man von GitHub mehr Unterstützung durch vorgefertigte Workflows als in Azure DevOps. Hinzu kommt das GitHub mit GitHub Security Lab neben direkter Funktionalität eine Plattform für Entwickler bietet, um einerseits die IT-Sicherheitsforschung voranzubringen, als auch Nutzer

bei Securitythemen zu unterstützen. Azure DevOps ist mit Azure Pipelines ein Tool mit vielen Möglichkeiten. Der Aufwand Security zu integrieren ist hier aber mehr Arbeit. Insgesamt merkt man das Microsoft bei Thema Security und DevSecOps vor allem mit GitHub wirbt. Die Verbindung von DevSecOps und Azure Pipelines kommt in der Dokumentation von Microsoft nicht vor. Auch das Abschalten von der DevOps Erweiterung Microsoft Security Code Analysis zeigt, das die Entwicklung auf GitHub fokussiert wird. Neue Sicherheitstrends werden also in der Zukunft eher bei GitHub implementiert werden.

# A Abkürzungsverzeichnis

**SSH** Secure Shell

**CWE** Common Weakness Enumeration

**OWASP** Open Web Application Security Project

**DSOMM** DevSecOps Maturity Model

**DAST** Dynamic Application Security Testing

**SAST** Static Application Security Testing

# B Abbildungsverzeichnis

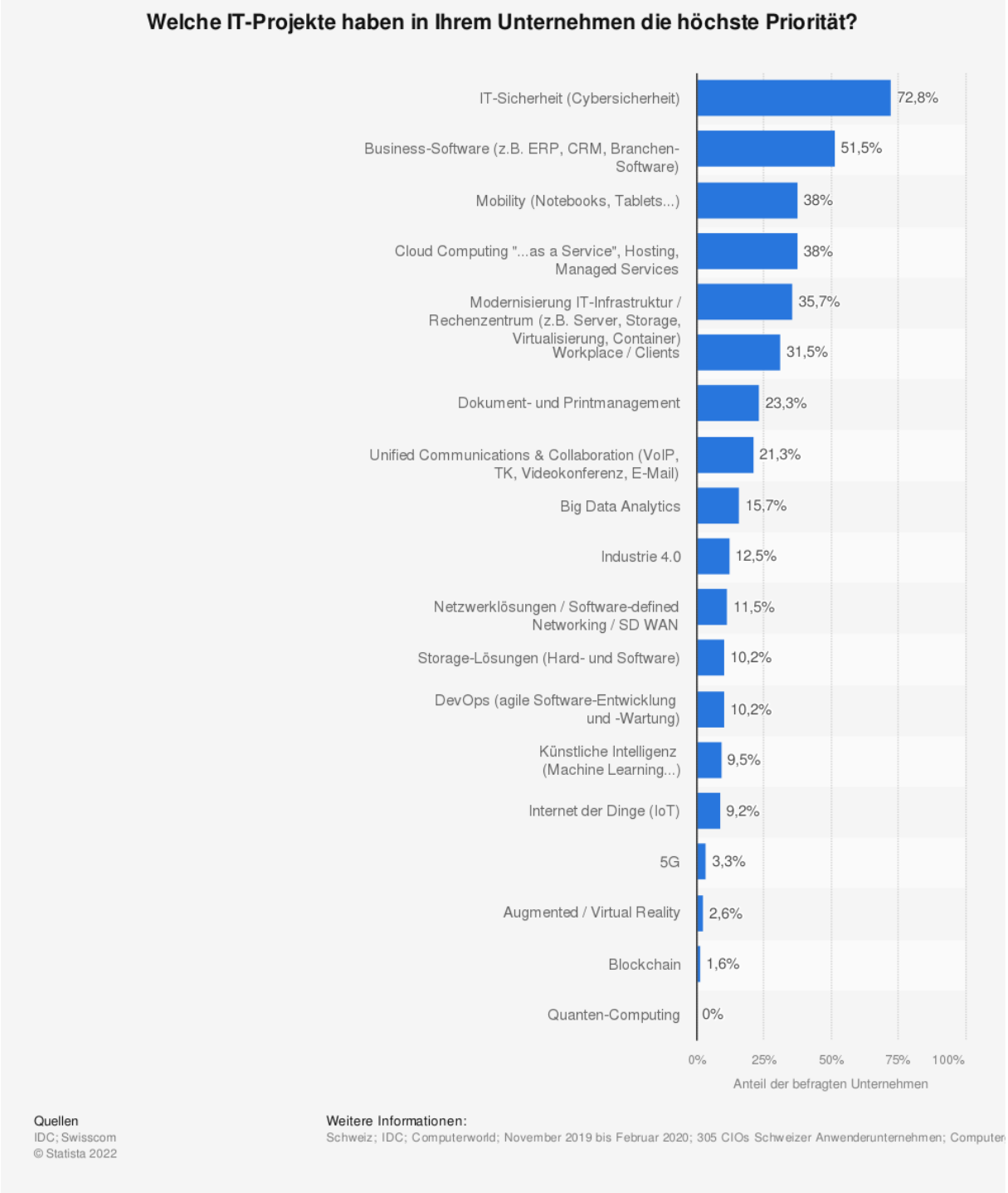


**Abbildung B.1** Welche IT-Projekte haben in Ihren Unternehmen die höchste Priorität

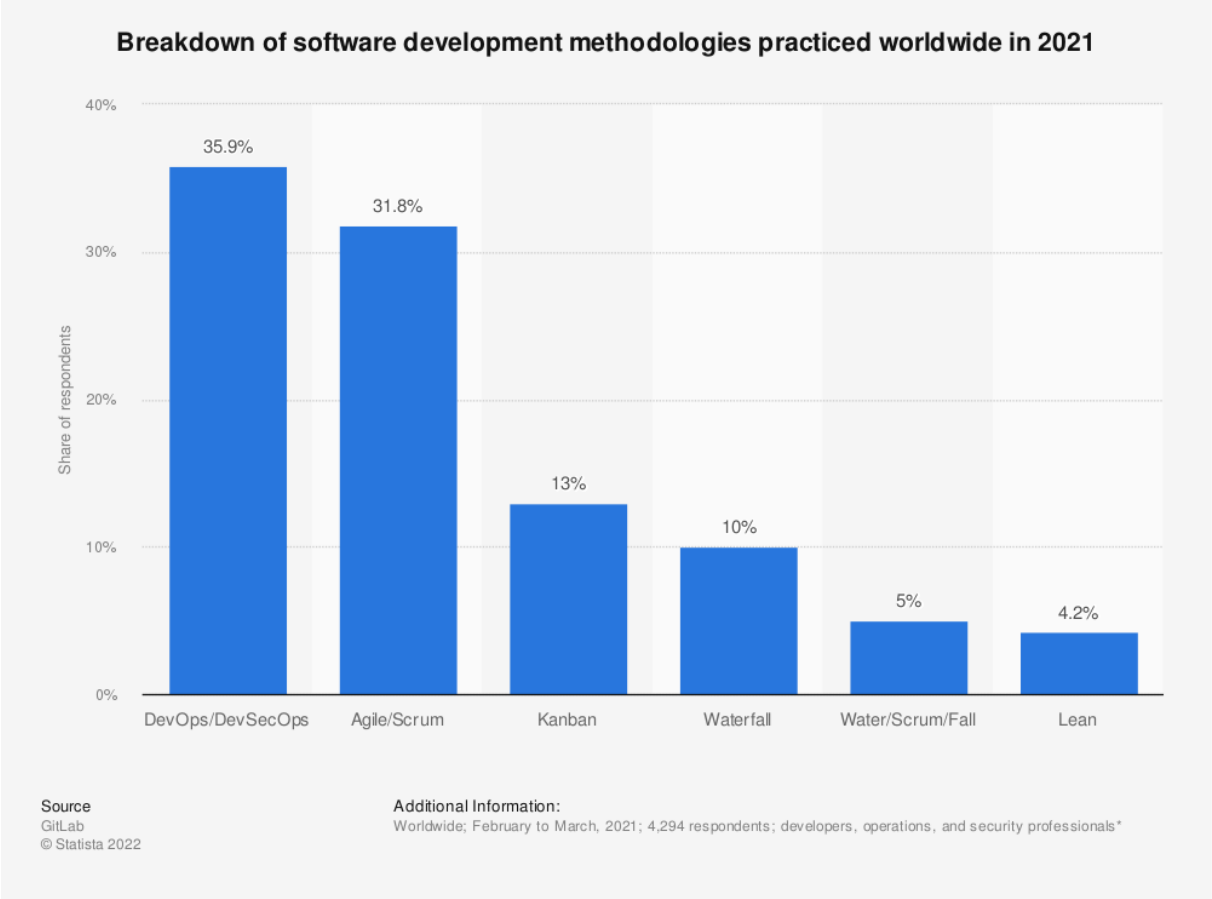


**Abbildung B.2** Breakdown of software development methodologies practiced wordlwide in 2021

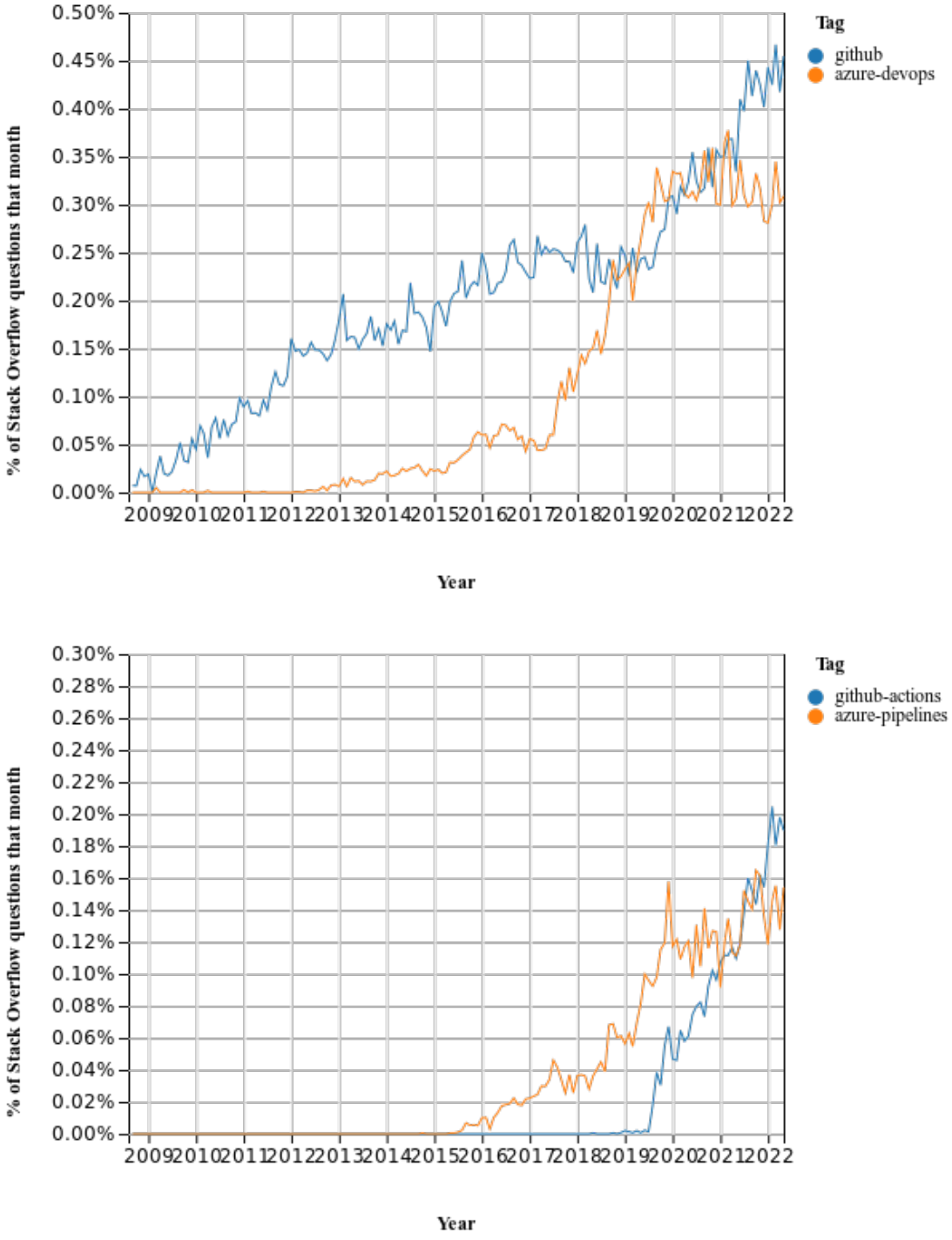


**Abbildung B.3** StackOverflow Trends von Github und Azure DevOps vom 18.06.2022

# Literaturverzeichnis


[Ahm19] Z. Ahmed und S. C. Francis. Integrating Security with DevSecOps: Techniques and Challenges. In *2019 International Conference on Digitization (ICD)*, S. 178–182. 2019.

[Alw20] K. Alwell. Achieving DevSecOps maturity with a developer-first, community-driven approach. https://github.blog/2020-08-06-achieving-devsecops-maturity-with-a-developer-first-communi Aug. 2020. Accessed: 2022-7-1.

[Bee21] F. Beetz und S. Harrer. GitOps: The Evolution of DevOps? *IEEE Software*, S. 0–0, 2021.

[Cob19] M. Cobb. What it takes to be a DevSecOps engineer. https://www.techtarget.com/searchsecurity/tip/What-it-takes-to-be-a-DevSecOps-engineer, Sept. 2019. Accessed: 2022-6-28.

[coda] About CodeQL. https://codeql.github.com/docs/codeql-overview/about-codeql/. Accessed: 2022-6-28.

[codb] Managing encrypted secrets for your codespaces. https://docs.github.com/en/codespaces/managing-your-codespaces/managing-encrypted-secrets-for-your-codespaces. Accessed: 2022-6-28.

[codc] Reviewing your security logs for Codespaces. https://docs.github.com/en/codespaces/managing-your-codespaces/reviewing-your-security-logs-for-codespaces. Accessed: 2022-6-28.

[codd] Setting up code scanning for a repository. https://docs.github.com/en/code-security/code-scanning/automatically-scanning-your-code-for-vulnerabilities-and-errors/setting-up-code-scanning-for-a-repository. Accessed: 2022-6-28.

[EdP] EdPrice-MSFT. DevSecOps in azure. https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/devsecops-in-azure. Accessed: 2022-7-1.

[gita] About access permissions on GitHub. https://docs.github.com/en/get-started/learning-about-github/access-permissions-on-github. Accessed: 2022-6-28.

[gitb] GitHub security lab. https://securitylab.github.com/. Accessed: 2022-6-28.

[Ibr22] A. Ibrahim, A. H. Yousef und W. Medhat. DevSecOps: A Security Model for Infrastructure as Code Over the Cloud. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, S. 284–288. 2022.

[Jus] Justinha. Kennwortlose Azure Active Directory-Anmeldung - Microsoft Entra. https://docs.microsoft.com/de-DE/azure/active-directory/authentication/concept-authentication-passwordless. Accessed: 2022-6-28.

[Kat] KathrynEE. Erste Schritte mit Berechtigungen, Zugriffsstufen und Sicherheitsgruppen - Azure DevOps. https://docs.microsoft.com/de-de/azure/devops/organizations/security/about-permissions?view=azure-devops&tabs=preview-page. Accessed: 2022-7-3.

[Lub22] S. Luber. Was ist Shift-Left? https://www.security-insider.de/was-ist-shift-left-a-1094279/, Febr. 2022. Accessed: 2022-7-1.

[Mac20] R. W. Macarthy und J. M. Bass. An Empirical Taxonomy of DevOps in Practice. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, S. 221–228. 2020.

[Mao20] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen und K. Lu. Preliminary Findings about DevSecOps from Grey Literature. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, S. 450–457. 2020.

[mica] DevOps with .NET and GitHub Actions - compare GitHub Actions with Azure Pipelines. https://docs.microsoft.com/en-us/dotnet/architecture/devops-for-aspnet-developers/actions-vs-pipelines. Accessed: 2022-7-1.

[micb] DevOps with .NET and GitHub Actions - compare GitHub Actions with Azure Pipelines. https://docs.microsoft.com/en-us/dotnet/architecture/devops-for-aspnet-developers/actions-vs-pipelines. Accessed: 2022-7-6.

[micc] Microsoft Security Code Analysis documentation overview. https://docs.microsoft.com/en-us/azure/security/develop/security-code-analysis-overview. Accessed: 2022-7-1.

[micd] Microsoft Security Code Analysis onboarding guide. https://docs.microsoft.com/en-us/azure/security/develop/security-code-analysis-onboard. Accessed: 2022-7-1.

[mice] Pipeline resource protection - Azure Pipelines. https://docs.microsoft.com/en-us/azure/devops/pipelines/security/resources?view=azure-devops. Accessed: 2022-7-1.

[micf] Security through templates. https://docs.microsoft.com/en-us/azure/devops/pipelines/security/templates?view=azure-devops. Accessed: 2022-7-1.

[Mig] F. Migacz. DevSecOps mit GitHub Security. https://docs.microsoft.com/de-de/azure/architecture/solution-ideas/articles/devsecops-in-github. Accessed: 2022-6-28.

[owa] OWASP Devsecops Maturity Model. https://owasp.org/www-project-devsecops-maturity-model/. Accessed: 2022-7-1.

[Por21] I. Porta. A side-by-side comparison of Azure DevOps and GitHub. https://medium.com/microsoftazure/a-side-by-side-comparison-of-azure-devops-and-github-834a223e0c16, Juni 2021. Accessed: 2022-7-3.

[sec] About Secret Scanning. https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning. Accessed: 2022-6-28.

[Wyl21] A. Wylde. Zero trust: Never trust, always verify. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, S. 1–4. 2021.

Eigenständigkeitserklärung / Declaration of Originality

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

*I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.*

Rosenheim, den tt.mm.jjjj

Vor- und Zuname