GitLab

Der ultimative CI/CD-Leitfaden

Von den Grundlagen bis zur Integration
von Sicherheitstests und KI

Inhalt

Einführung

Die kontinuierliche Integration und kontinuierliche Lieferung (CI/CD) hat die Zusammenarbeit von Softwareteams bei der Umsetzung von Code in Anwendungen revolutioniert. Vorbei sind die Zeiten des Kopfzerbrechens bei der Code-Integration und den sich wiederholenden manuellen Prozessen. CI/CD macht moderne Softwareentwicklung möglich – schnell, zuverlässig und automatisiert.

Im Kern geht es bei CI/CD darum, eine nahtlose Pipeline zu schaffen, die den Code von der Entwicklungsumgebung bis zur Produktivumgebung bringt und Feedback in Echtzeit berücksichtigt. CI hilft Teams dabei, Probleme frühzeitig zu erkennen, bevor sie zu kostspieligen Problemen werden, indem es sicherstellt, dass Codeänderungen häufig in einem gemeinsamen Repository zusammengeführt, automatisch getestet und validiert werden. Mit CD wird dies noch erweitert, indem die Bereitstellung automatisiert wird, sodass Releases vorhersehbar und stressfrei sind.

Anstatt auf manuelle Prozesse und komplexe Toolchains für die Softwareentwicklung zu setzen, können Teams eine robuste CI/CD-Pipeline nutzen, um Software zu erstellen, zu testen und bereitzustellen. Und KI kann den Prozess noch weiter optimieren, indem sie automatisch CI/CD-Pipelines für konsistente Qualitäts-, Compliance- und Sicherheitsprüfungen erstellt.

Das Ergebnis sind hochwertigere Software, kürzere Lieferzeiten und häufigere Veröffentlichungen, mit denen man schneller auf Benutzer- und Marktanforderungen reagieren kann.

In diesem Leitfaden werden moderne CI/CD-Pipelines von den Grundprinzipien über Best Practices bis hin zu fortschrittlichen Strategien erklärt. Was du in diesem Leitfaden lernst, wird dir helfen, deine DevSecOps-Umgebung zu skalieren, um Software auf agile, automatisierte und effiziente Weise zu entwickeln und bereitzustellen.

Grundlagen von CI/CD

Mit kontinuierlicher Integration können Teams Fehler und Sicherheitsprobleme leichter und viel früher im Entwicklungsprozess erkennen und beheben.

Was ist kontinuierliche Integration?

Unter kontinuierlicher Integration (CI) versteht man das frühzeitige und häufige Integrieren aller Codeänderungen in den main-Branch eines gemeinsamen Quellcode-Repositorys, das automatische Testen von Änderungen bei der Übergabe oder Zusammenführung und das automatische Starten eines Builds. Mit kontinuierlicher Integration können Teams Fehler und Sicherheitsprobleme leichter und viel früher im Entwicklungsprozess erkennen und beheben.

Was ist kontinuierliche Lieferung?

Die kontinuierliche Lieferung (CD) – manchmal auch kontinuierliche Bereitstellung genannt – ermöglicht es Unternehmen, ihre Anwendungen automatisch bereitzustellen, so dass die Entwickler(innen) mehr Zeit haben, sich auf die Überwachung des Bereitstellungsstatus zu konzentrieren und den Erfolg sicherzustellen. Bei der kontinuierlichen Lieferung legen die DevSecOps-Teams die Kriterien für die Freigabe des Codes im Voraus fest. Wenn diese Kriterien erfüllt und validiert sind, wird der Code in der Produktivumgebung bereitgestellt. Auf diese Weise können Unternehmen flexibler agieren und neue Funktionen schneller an die Benutzer(innen) weitergeben – mit der Gewissheit, dass ihre Bereitstellungen alle Sicherheitstests bestanden haben.

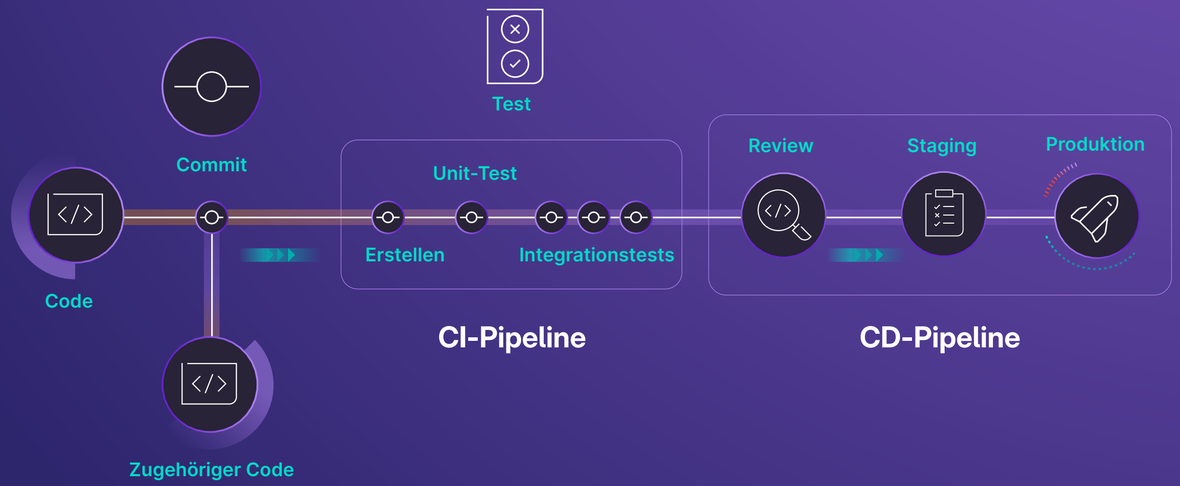
Was ist eine CI/CD-Pipeline?

Eine CI/CD-Pipeline besteht aus einer Reihe von Schritten, wie z. B. Erstellen, Testen und Bereitstellen, die die Softwarebereitstellung automatisieren und optimieren. Jede Phase dient als Qualitätsüberprüfung und stellt sicher, dass nur validierter Code weiterentwickelt wird. In den frühen Phasen werden in der Regel grundlegende Überprüfungen wie Kompilierung und Unit-Tests durchgeführt, während spätere Phasen Integrationstests, Leistungstests, Konformitätstests und gestaffelte Bereitstellungen in verschiedenen Umgebungen umfassen können.

Die Pipeline kann so konfiguriert werden, dass manuelle Genehmigungen an kritischen Punkten erforderlich sind, z. B. vor der Bereitstellung für die Produktion, während gleichzeitig Routineaufgaben automatisiert werden und Entwickler(innen) schnelles Feedback über den Zustand ihrer Änderungen erhalten. Dieser strukturierte Ansatz sorgt für Konsistenz, reduziert menschliche Fehler und bietet einen klaren Audit-Trail, wie Codeänderungen sich von der Entwicklung bis in die Produktion bewegen. Moderne Pipelines werden oft als Code implementiert, so dass sie wie Anwendungscode versioniert, getestet und gepflegt werden können.

Die folgenden Begriffe sind im Zusammenhang mit CI/CD wichtig:

- **Commit:** Eine Codeänderung.
- **Job:** Anweisungen, die ein Runner ausführen muss.
- **Runner:** Ein Agent oder Server, der jeden Job einzeln ausführt und je nach Bedarf hoch- oder herunterfahren kann.
- **Phasen:** Ein Schlüsselwort, das bestimmte Jobphasen definiert, z. B. „Erstellen“ und „Bereitstellen“. Jobs der gleichen Phase werden parallel ausgeführt. Pipelines werden mithilfe der versionierten YAML-Datei .gitlab-ci.yml auf der Root-Ebene eines Projekts konfiguriert.



Die Vorteile von CI/CD in der modernen Softwareentwicklung

CI/CD bringt der modernen Softwareentwicklung entscheidende Vorteile, indem es die Zeit und das Risiko für die Bereitstellung neuer Funktionen und Korrekturen drastisch reduziert. Durch die kontinuierliche Feedbackschleife können DevSecOps-Teams sicher sein, dass ihre Änderungen automatisch für die gesamte Codebase validiert werden.

Einer der wichtigsten Aspekte ist jedoch, dass CI/CD eine Kultur der Zusammenarbeit und Transparenz innerhalb von Softwareentwicklungsteams fördert. Wenn jeder den Status von Builds, Tests und Bereitstellungen in Echtzeit sehen kann, wird es einfacher, Engpässe im Bereitstellungsprozess zu identifizieren und zu beheben. Die von CI/CD ermöglichte Automatisierung reduziert auch die kognitive Belastung für Entwickler(innen) und gibt ihnen die Möglichkeit, sich auf das Schreiben von Code zu konzentrieren, anstatt manuelle Bereitstellungsprozesse zu verwalten. Dies führt zu einer höheren Zufriedenheit und Produktivität der Entwickler(innen) und reduziert gleichzeitig das Risiko, das traditionell mit dem gesamten Softwareveröffentlichungsprozess verbunden ist. Teams können freier experimentieren, wenn sie wissen, dass schnelle Code Reviews Teil des Prozesses sind, und sie können Änderungen bei Bedarf schnell zurücknehmen, was Innovationen und kontinuierliche Verbesserungen fördert.

Hier sind einige der wichtigsten Vorteile von CI/CD:

Häufigere Bereitstellungen. Teams können kleinere Änderungen häufiger bereitstellen. Das Risiko und die Komplexität werden dadurch geringer. Außerdem können sie schneller auf Rückmeldungen reagieren und schneller entwickeln.

Schnellere Markteinführung. Durch automatisierte Prozesse kann man neue Funktionen und Fehler schnell bereitstellen. Das spart Zeit, die man sonst für die Produktion braucht.

Schnelleres Testen. Durch automatisierte Tests bei Codeänderungen und die gleichzeitige Ausführung mehrerer Tests in verschiedenen Umgebungen verkürzt CI/CD die Testzeit im Vergleich zu manuellen Ansätzen erheblich.

V**erbesserte Codequalität.** Durch automatisierte Tests werden Fehler sofort erkannt und beseitigt. So ist sichergestellt, dass der Code besser wird und jede Version richtig funktioniert.

Schnellere Wiederherstellung. Mit CI/CD werden Probleme einfacher und schneller gelöst. Auch der Zeitraum bis zur Wiederherstellung nach Vorfällen wird kürzer. Kleine Software-Updates helfen dabei, Fehler schneller zu finden. Entwickler(innen) können Programmierfehler schneller beheben oder zu einer früheren Version zurückkehren, damit die Kundschaft schnell weiterarbeiten kann.

Einfachere Compliance und Audits. Compliance-Aufgaben können in den Lebenszyklus der Entwicklung integriert werden, wodurch das Risiko der Veröffentlichung nicht konformer Anwendungen verringert wird. CI/CD-Systeme zeichnen alle Builds, Tests und Bereitstellungen detailliert auf und bieten so einen umfassenden Audit-Trail, der die Problembehebung, die Einhaltung von Compliance-Anforderungen und die Durchführung von Audits erleichtert.

Weniger Kontextwechsel. Entwickler(innen) können sich leichter auf etwas konzentrieren und sind weniger belastet, wenn sie Echtzeit-Feedback zu ihrem Code bekommen. Kleine Codeabschnitte, die automatisch getestet werden, helfen ihnen, Probleme zu finden, während sie noch daran arbeiten. Sie müssen weniger Code überprüfen, was die Suche nach Fehlern einfacher macht.

Konsistente Prozesse. Mit CI/CD wird die Erstellung von Software einfacher. Teams können besser zusammenarbeiten, wichtige Termine einhalten und bessere Software liefern.

Zufriedenere (und produktivere) Entwickler(innen). Da ein Großteil des Bereitstellungsprozesses automatisiert ist und weniger Kontextwechsel anfallen, hat das Team mehr Zeit für Projekte, die für die Entwickler(innen) und ihr Unternehmen lohnender sind.

Zufriedenere Benutzer(innen) und Kund(inn)en. Wenn weniger Fehler in die Produktion gelangen und neue Funktionen und Fehlerbehebungen schneller und häufiger veröffentlicht werden, ist es einfacher, die Kundenzufriedenheit und -bindung zu verbessern und neue Kund(inn)en zu gewinnen.

Hauptunterschiede zwischen CI/CD und traditioneller Entwicklung

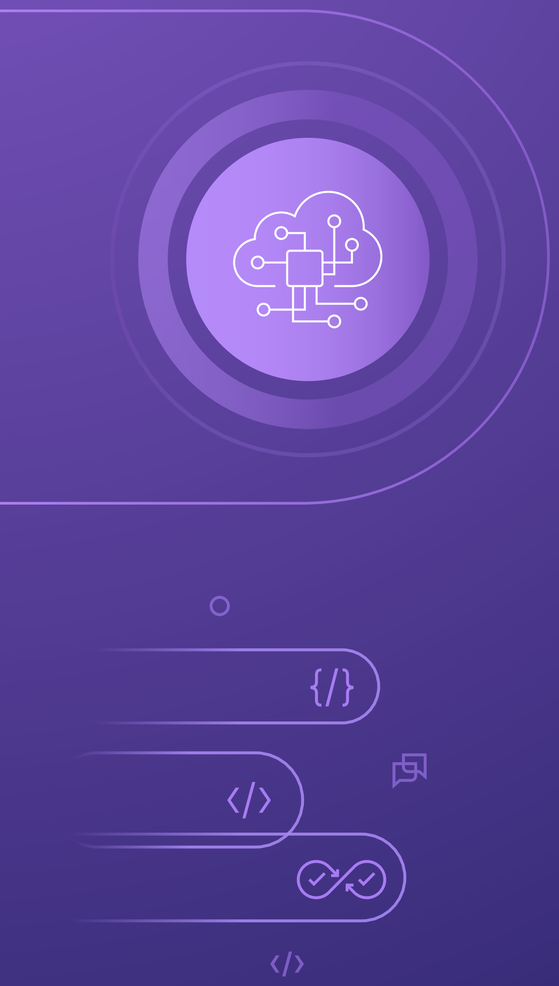
CI/CD unterscheidet sich in vielerlei Hinsicht von der traditionellen Softwareentwicklung:

Häufige Code-Commits
Entwickler(innen) arbeiten oft unabhängig und laden ihren Code selten in eine main-Codebase hoch, was zu Merge-Konflikten und anderen zeitaufwändigen Problemen führt. Mit CI/CD pushen Entwickler(innen) Commits den ganzen Tag über, um sicherzustellen, dass Konflikte frühzeitig erkannt werden und die Codebase auf dem neuesten Stand bleibt.

Reduziertes Risiko
Langwierige Testzyklen und eine umfassende Vorabplanung kennzeichnen die traditionelle Softwareentwicklung. Dadurch sollen Risiken minimiert werden, jedoch wird die Möglichkeit eingeschränkt, Probleme zu finden und zu beheben. Das Risikomanagement in CI/CD basiert darauf, dass kleine, inkrementelle Änderungen vorgenommen werden, die genau überwacht und leicht rückgängig gemacht werden können.

Automatisiertes und kontinuierliches Testen
In der traditionellen Softwareentwicklung werden die Tests erst durchgeführt, wenn die Entwicklung abgeschlossen ist. Dies führt jedoch zu Problemen wie verspäteter Lieferung und kostspieliger Fehlerbehebungen. CI/CD unterstützt automatisierte Tests, die während der gesamten Entwicklung kontinuierlich durchgeführt und durch jeden Code-Commit ausgelöst werden. Entwickler(innen) erhalten außerdem Feedback, auf das sie schnell reagieren können.

Automatisierte, wiederholbare und häufige Bereitstellungen
Mit CI/CD sind Bereitstellungen automatisierte Prozesse, die den typischen Stress und Aufwand reduzieren, der mit großen Software-Rollouts verbunden ist. Der gleiche Bereitstellungsprozess kann in allen Umgebungen wiederholt werden, was Zeit spart und Fehler und Inkonsistenzen reduziert.



Best Practices für CI/CD-Implementierung und -Management

Wie erfolgreich du mit CI/CD bist, hängt stark von den Best Practices ab, die du implementierst.

- Committe früh und oft.
- Optimiere die Pipeline-Phasen.
- Erstelle Builds schnell und einfach.
- Nutze Fehlschläge, um Prozesse zu verbessern.
- Stelle sicher, dass die Testumgebung die Produktion widerspiegelt.
- Achte auf die Integration mit deinem Quellcodeverwaltungssystem (SCM).
- Beginne dort, wo du gerade bist – du kannst jederzeit nachbessern.
- Die beste kontinuierliche Lieferung erfolgt mit minimalen Mitteln.
- Verfolge, was passiert, damit Probleme und Merge Requests nicht außer Kontrolle geraten.
- Optimiere Benutzerakzeptanztests und Staging mit Automatisierung.
- Verwalte die Release-Pipeline durch Automatisierung.
- Implementiere Überwachung für Transparenz und Effizienz.
- Beginne mit KI zu experimentieren, um deinen Prozess effizienter und sicherer zu gestalten.

Sehen wir uns ein paar dieser Best Practices genauer an.

Integration von Quellcodeverwaltung und CI/CD

Quellcodeverwaltung (SCM) und CI/CD bilden die Grundlage moderner Softwareentwicklungspraktiken. SCM-Systeme wie Git bieten eine zentrale Möglichkeit, Änderungen zu verfolgen, verschiedene Codeversionen zu verwalten und die Zusammenarbeit zwischen den Teammitgliedern zu erleichtern, während gleichzeitig ein stabiler main-Branch gepflegt wird, der immer produktionsreifen Code enthält.

CI/CD verwendet den von SCM-Systemen verwalteten Code, um ihn zu erstellen, zu testen und zu validieren. Wenn das SCM-System und das CI/CD-Tool eng integriert sind – oder idealerweise Teil derselben DevSecOps-Plattform – geschieht dies automatisch. Wenn Entwickler(innen) Codeänderungen einreichen, ruft das CI/CD-System automatisch den neuesten Code ab, kombiniert ihn mit der vorhandenen Codebase und führt eine Reihe automatisierter Überprüfungen durch. Dazu gehören in der Regel das Kompilieren des Codes, das Ausführen von Unit-Tests, das Durchführen einer statischen Codeanalyse und das Überprüfen der Testabdeckung. Wenn einer dieser Schritte fehlschlägt, wird das Team sofort benachrichtigt, sodass es Probleme beheben kann, bevor sie sich auf andere Entwickler(innen) auswirken oder in die Produktivumgebung überführt werden.

Diese enge Integration von Versionskontrolle und kontinuierlicher Integration schafft eine Feedbackschleife, die zur Aufrechterhaltung der Codequalität beiträgt, die Behebung von Sicherheitslücken erleichtert und verhindert, dass sich Integrationsprobleme häufen. Außerdem erhöht sie die Produktivität und beschleunigt den Entwicklungsprozess.

Wenn SCM- und CI/CD-Tools getrennt oder nicht eng integriert sind, müssen die Entwickler(innen) ständig zwischen ihnen hin- und herwechseln, Builds nach Commits manuell auslösen und Build-Informationen in Merge Requests zurückkopieren. Dieser Kontextwechsel erhöht die Wahrscheinlichkeit menschlicher Fehler und schafft Lücken in der Nachvollziehbarkeit zwischen Codeänderungen und Bereitstellung. Die fehlende Integration erschwert auch die Umsetzung automatisierter Richtlinien, die den gesamten Software-Entwicklungsprozess abdecken, und zwingt die Teams dazu, eigene Skripte oder Webhooks zu erstellen, um die Lücke zwischen den Systemen zu schließen. Bei der Problembehandlung müssen die Entwickler(innen) Informationen aus verschiedenen, nicht miteinander verbundenen Tools zusammenstellen, was es zeitaufwändiger macht, zu verstehen, was schiefgelaufen ist und wo in der Pipeline ein Fehler aufgetreten ist.

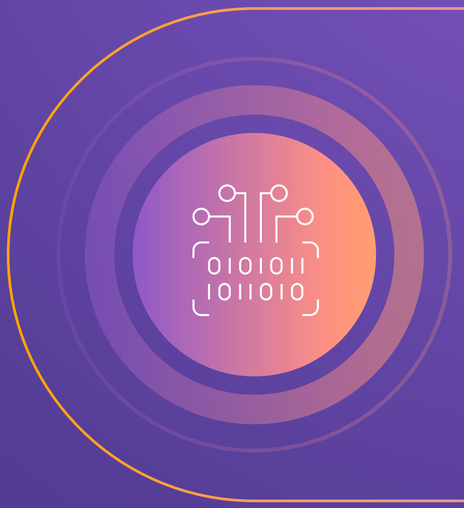
Verbesserung von CI/CD mit KI

Die KI kann noch einen Schritt weiter gehen und die CI/CD-Prozesse über den gesamten Entwicklungslebenszyklus hinweg auf verschiedene Weise verbessern.

Eine wichtige Anwendung von KI ist die Problembehandlung in den CI/CD-Pipelines. Anstatt sich durch Job-Protokolle, Fehlermeldungen und Ausführungstraces zu wühlen, um herauszufinden, warum ein CI/CD-Job fehlgeschlagen ist, können Entwickler(innen) mithilfe von KI die Grundursache ermitteln und sich eine Lösung vorschlagen lassen.

Entwickler(innen) können KI auch nutzen, um Sicherheitsprobleme und Sicherheitslücken, die in ihrem Code entdeckt werden, zu verstehen, zu priorisieren und zu beheben. Wenn ein Sicherheitsscan in der CI/CD-Pipeline eine Schwachstelle entdeckt, kann KI die Sicherheitslücke zusammenfassen, Beispiele für ihre Ausnutzung nennen und eine Lösung vorschlagen, einschließlich des erforderlichen Codes zur Behebung der Sicherheitslücke.

CI/CD allein ist bereits eine leistungsstarke Methode, um bessere Software schneller auszuliefern. Mit KI kannst du deinen Softwareentwicklungsprozess noch effizienter und sicherer machen.



Erste Schritte mit CI/CD

Für den Einstieg in CI/CD verwendest du ein einfaches, aber repräsentatives Projekt als Pilotprojekt. Wähle eine einfache Anwendung mit simplen Testanforderungen aus, damit du dich auf das Erlernen der Pipeline-Mechanismen konzentrieren kannst, anstatt dich mit komplexen Bereitstellungsszenarien zu befassen. Stelle zunächst sicher, dass dein Code versioniert ist und über einige grundlegende automatisierte Tests verfügt – ein paar Unit-Tests reichen für den Anfang aus. Das Ziel ist es, eine minimale Pipeline zu erstellen, die du nach und nach erweitern kannst, wenn dein Verständnis wächst.

Im Fall von GitLab beginnt der Prozess mit dem Erstellen der Datei .gitlab-ci.yml im Stammverzeichnis deines Projekts. Diese YAML-Datei definiert deine Pipeline-Phasen (grundlegende Phasen wie Erstellen, Testen und Bereitstellen) und Jobs. Eine einfache Pipeline könnte so aussehen: In der Phase „Erstellen“ wird dein Code kompiliert und es werden Artefakte erstellt, die Phase „Test“ führt deine Unit-Tests aus und die Phase „Bereitstellen“ pusht deine Anwendung in eine Staging-Umgebung. GitLab erkennt diese Datei automatisch und beginnt, deine Pipeline auszuführen, wenn Änderungen in dein Repository gepusht werden. Die Plattform bietet integrierte Runner zur Ausführung deiner Pipeline-Aufträge, du kannst aber auch deine eigenen Runner einrichten, um mehr Kontrolle zu haben.

Wenn du mit den Grundlagen vertraut bist, kannst du nach und nach anspruchsvollere Elemente zu deiner Pipeline hinzufügen. Dazu gehören z. B. Codequalitätsprüfungen, Sicherheitsscans oder die automatische Bereitstellung für die Produktion. Die DevSecOps-Plattform von GitLab bietet Funktionen wie Compliance-Management, Bereitstellungsvariablen und manuelle Approval-Gates, die du einbauen kannst, wenn deine Pipeline ausgereift ist. Achte auf die Ausführungszeit der Pipeline und suche nach Möglichkeiten, Jobs parallel auszuführen. Denke daran, die richtige Fehlerbehandlung und Benachrichtigungen hinzuzufügen, damit deine Teammitglieder umgehend über alle Pipeline-Ausfälle informiert werden. Beginne damit, häufige Probleme und Lösungen zu dokumentieren, wenn sie auftauchen – dies wird von unschätzbarem Wert sein, wenn dein Team wächst.

Erste Schritte mit GitLab CI/CD.

Registriere dich für GitLab Ultimate und teste die KI-basierte DevSecOps-Plattform 60 Tage lang kostenlos.

Mehr erfahren

Über GitLab

GitLab ist die umfassendste, KI-gestützte DevSecOps-Plattform für Softwareinnovationen. GitLab bietet eine Schnittstelle, einen Datenspeicher, ein Berechtigungsmodell, eine Wertschöpfungskette, eine Serie von Berichten, einen Ort zum Sichern deines Codes, einen Ort für die Bereitstellung in jeder Cloud und einen Ort, an dem jede und jeder einen Beitrag leisten kann. Die Plattform ist die einzige echte Cloud-agnostische, durchgängige Plattform für DevSecOps, die alle DevSecOps-Funktionen an einem Ort vereint.

Mit GitLab können Unternehmen Code schnell und kontinuierlich erstellen, bereitstellen und verwalten, um ihre Geschäftsvision in die Realität umzusetzen. GitLab ermöglicht Kund(inn)en und Anwender(inne)n schnellere Innovationen, eine einfachere Skalierung sowie eine effektivere Betreuung und Bindung von Kund(inn)en. Als Open-Source-Lösung agiert GitLab an der Seite seiner wachsenden Community aus Tausenden Entwickler(inne)n und Millionen Anwender(inne)n, um kontinuierlich neue Innovationen zu liefern.

GitLab