MAGAZIN FÜR PROFESSIONELLE
INFORMATIONSTECHNIK

GitLab Enterprise Ultimate als DevOps-Toolchain für Entwickler

28.05.2020 07:00 Uhr Christoph Puppe

GitLab präsentiert sich als die Zentrale für den Prozess der Softwareentwicklung – inklusive Auditing, Weiterentwicklung, Betrieb und Überwachung.

Schon der Name deutet es an: GitLab bietet den Git-Server – aber auch Benutzerverwaltung, Runner, Wiki, Issue Tracker, Kanban-Board, Container-Registry, Sicherheitsscanner für Container und Sourcecode, Multi-Cluster-Management und -Monitoring sowie Analyse der Entwicklung. Dazu ist es in fast alle bekannte Produkte für Entwickler integrierbar und selbst über eine API zu steuern. Für diesen Artikel kam die "GitLab Enterprise Edition 12.8.1-ee" mit der Ultimate-Lizenz zum Einsatz.

Laut Webseite sollte eine Installation auf Ubuntu wenige Befehle erfordern. Das Skript hat aber einen Bug und so war ein anderer Weg angebracht. Das Einspielen per Helm auf einem GKE-Cluster verlief reibungslos und war in unter 30 Minuten erledigt. Das Passwort für root steht als Secret bereit und nach kurzem Einschwingen des Service steht eine vollständige Installation zur Verfügung.

Wer möchte, kann jetzt georedundante Instanzen hinzufügen, die alle Daten vorhalten, und auf diese Weise eine Lastverteilung über Standorte realisieren. Die Instanzen leiten allerdings alle Schreibvorgänge an die Zentrale weiter, die diese dann auf die Geo-Nodes repliziert.

Grundeinstellungen wie die LDAP-Integration für die Benutzerverwaltung sind ebenso vorhanden wie sehr viele Services, die Projekte mit externen Diensten wie Slack, Prometheus, Jira, Bugzilla und rund zwei Dutzend weiteren verbinden.

IX-TRACT

- CI/CD und Container bestimmen immer mehr den Alltag von Entwicklern und Administratoren.
- GitLab vereint eine komplette DevOps-Toolchain unter einer Oberfläche.
- Die kommerziellen GitLab-Varianten bieten zusätzliche Features, wie Code- und Vulnerability-Scanner oder mehr Monitoring.

MEHR ZUM THEMA GITLAB

GitLab mit passender Continuous-Integration-Pipeline automatisiert aufsetzen [1]

GitLab CI sowie CD installieren und konfigurieren [2]

GitLab als One-Stop-Toolchain für Entwickler [3]

Community versus Enterprise

Gegenüber der Community- und der Premium-Version bietet die umfangreichste Lizenz (Ultimate) einige große und viele kleine Features, die das Code-, Release- und Projektmanagement vereinfachen.

Ende März hat GitLab angekündigt, **18 Features aus den bezahlten Versionen in die FOSS Edition zu übernehmen [4]**. Das soll in den kommenden Wochen geschehen. Ein Beispiel dafür sind Related Issues, die auf einen Blick Abhängigkeiten zwischen Aufgaben zeigen. Hierfür werden verwandte und blockierende Issues einfach referenziert. Weiter sollen folgende Funktionen in die FOSS Edition wandern: Export Issues, Issue Board Focus Mode, Service Desk, Web Terminal für Web IDE, File Syncing zum Web Terminal, Design Management, Code Quality, Package Managers, Canary Deployments, Incremental Rollout, Feature Flags, Deploy Boards sowie mehrere Kubernetes-Cluster und Network Policies für Kubernetes.

Scoped Labels ermöglichen es, komplexere Arbeitsabläufe mit Labels abzubilden. Hierfür bekommt eine zusammengehörende Liste von Labels einen Scope zugewiesen. Nun kann es nur noch ein Label aus dieser Liste an einem Issue geben. Eine erneute Zuweisung eines Labels aus der Gruppe entfernt automatisch das alte Label, beispielsweise beim Übergang eines Issue von Entwicklung nach Testing oder Betriebssystem.

Epic Boards für Projektgruppen beinhalten Gantt-Diagramme für eine zeitliche Übersicht. Ebenfalls nur in der Ultimate enthalten sind mehrere Issue Boards für Projektgruppen, die sich auch spezifischer konfigurieren lassen, etwa in der Ansicht aller Issues eines Mitarbeiters oder eines definierten Meilensteines.

Die größeren Unterschiede zeigen sich beim Steuern und Verwalten der Deployments und der Automatisierung von Codeanalysen. So enthält GitLab Ultimate Scanner für Fehler im Code (SAST – Static Application Security Testing), Verwundbarkeiten in Abhängigkeiten und Containern sowie einen für Schwachstellen in Webseiten (DAST – Dynamic Application Security Testing). Auch sind Security- und Compliance-Dashboards, Monitoring von Kubernetes und Pod Logs Teil des Funktionsumfangs. **Ein ausführlicher Vergleich der GitLab-Versionen findet sich auf der GitLab-Website [5]**.

[6]

Berechtigungsmanagement

GitLab kann entweder die interne Benutzerverwaltung verwenden oder eine externe über OAuth 2, LDAP et cetera anbinden. Eine Besonderheit ist, dass GitLab per OAuth 2 nicht nur diverse Identity Manager nutzt, sondern auch selbst ein OAuth-2-Provider ist. Das heißt, andere Anwendungen können ihre Benutzer über GitLab authentifizieren. Das Benutzermanagement kann für Gruppen und Benutzer Zwei-Faktor-Authentifizierung erzwingen. Das Einrichten eines FIDO2-Tokens klappte reibungslos. Wer in seiner Organisation Zugriffsrechte über Labels definiert – wie dies in Multi-Level-Security-Umgebungen üblich ist –, kann eine URL hinterlegen, bei der GitLab erfragt, ob der Benutzer über die Berechtigungen für dieses Label und Projekt verfügt.

Grundsätzlich folgt GitLab den Regeln der Role-based Access Control (RBAC). Benutzer sind entweder direkt oder als Mitglied einer Gruppe für Objekte autorisiert. Ein Repository ist ein Projekt, das entweder zu einer Gruppe oder zu einem Benutzer gehört. Diese Gruppen sind dieselben, in denen auch Benutzer zusammengefasst sind. Das ist zu Anfang etwas verwirrend, aber im Alltags praktisch, da die Benutzer in der Gruppe Berechtigungen haben, die dann automatisch für alle Projekte dieser Gruppe gelten.

Das Autorisieren der Benutzer für ein Projekt erfolgt – entweder direkt oder über die Gruppe – in den Rollen Gast, Reporter, Developer, Maintainer oder Owner. Auch lässt sich der Zugriff zeitlich limitieren

und erlischt am gesetzten Enddatum. Um hier noch mehr Flexibilität zu haben, erlaubt GitLab, auch Gruppen in ein Projekt einzuladen. Deren Benutzer erhalten dann die Berechtigungen, die sie in ihrer Ursprungsgruppe haben, bis zu einem gesetzten Maximalwert. Wenn also die Gruppe "Dev-Team A" eine Einladung in das "Projekt B" erhält, besitzen die Mitglieder weiterhin ihre Rolle: Ein Reporter aus der Gruppe A ist auch in "Projekt B" nur ein Reporter oder ein Developer ist auch weiterhin ein Developer, aber nur bis zum gesetzten Maximalwert. Ist der Maximalwert beispielsweise Developer, kann ein Maintainer aus dem "Dev-Team A" über diese Einladung nicht zum Maintainer in "Projekt B" aufsteigen. Ein weiteres nettes Detail ist die Unterscheidung in interne und externe Benutzer.

GIT

Alle notwendigen Funktionen wie SSH-Server et cetera sind nach der Installation vorhanden und funktionieren einwandfrei. So ist es einfach, eine alte und stark verwundbare Version von WordPress in ein Testprojekt einzuchecken. Die für diesen Test gewählte Version ist die 4.8.1 und **cvedtails.com listet dafür 31 Verwundbarkeiten mit Scores von 3,5 bis 7,5 [7]**.

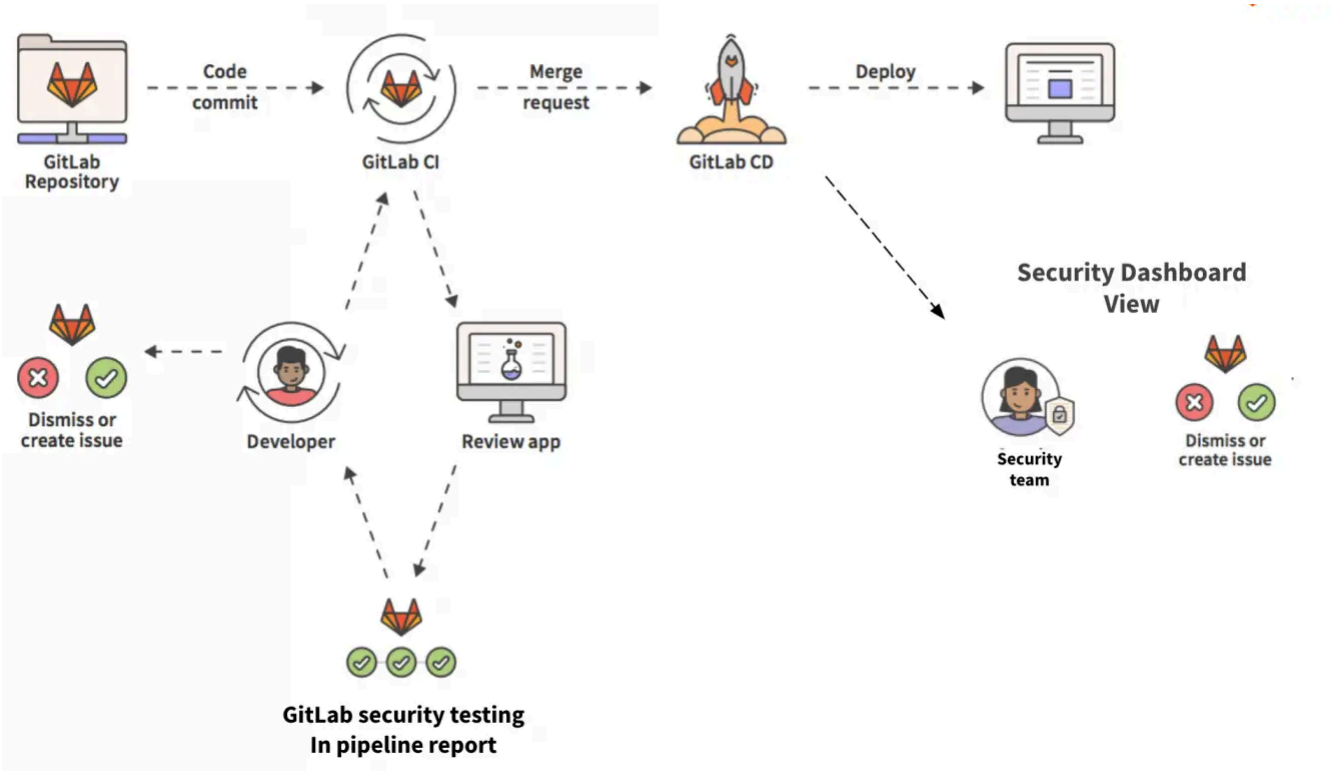
Mit diesem Code soll GitLab zeigen, wie es Verwundbarkeiten erkennt und die Bearbeitung anstößt. Statischer Codescan und Unterstützung der Programmierer durch automatische Pull Requests für alle betroffenen Branches gehören zum Leistungsumfang, den man bei einer kommerziellen Git-Software erwarten kann. Alle Ereignisse im Git können mit WebHooks externe Programme triggern, das gilt auch für Kommentare im Code. Die WebHooks sind dabei über Token gesichert.

Um den Code vor (un-)absichtlichen Fehlern zu schützen, wendet GitLab das Prinzip des Protected Branch an: Nur die Maintainer dürfen in diesen Branch mergen und commiten. Zusätzlich zeigt das Compliance-Dashboard jede Veränderung an.

CI/CD

Wer in einem größeren Projekt den Überblick behalten will, findet eine umfangreiche Sammlung von Auswertungen über die Commits, Issues, Merges und so weiter. Leider sind diese Insights nicht so umfangreich wie die, die etwa GitHub bietet.

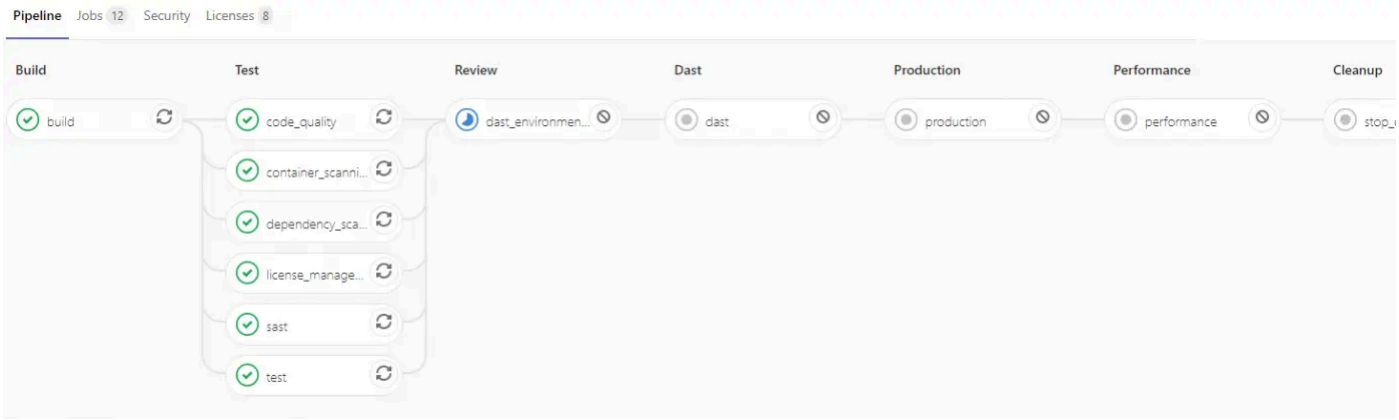
Im Unterschied zu anderen GIT-Servern oder Container-Image-Registries enthält GitLab eine vollständige Umgebung für das Steuern automatisierter Jobs bis hin zum getriggerten Ausrollen neuer Anwendungsversionen in diverse Umgebungen und nach ausgiebigen Tests. Dafür muss im obersten Verzeichnis des Repo eine Datei .gitlab-ci.yml liegen. In dieser sind alle Stages zu definieren.



GitLab-Vorschlag für einen beschleunigten Entwicklungszyklus: Auch bei Left-Shift Security – also Sicherheit als Teil des Entwicklungsprozesses mit Automatisierung – sollte hier nach ITIL das Securityteam spätestens den Merge Request auch manuell bewerten.
(Bild: GitLab.com)

Für CI/CD benötigt GitLab Ressourcen, die es als Kubernetes-Cluster oder lokal auf dem gleichen Docker-Daemon nutzen kann. Dort startet der Runner die Jobs der Pipelines, kann aber auch die Anwendung für Tests, Staging und produktiven Betrieb ausrollen. Umgebungen für diese Deployments kann es beliebig viele geben und es lassen sich auch Kubernetes-Cluster bestimmten Umgebungen zuordnen. Typisch wäre hier, jeweils einen oder mehrere Cluster für Test, Integration und Produktion zu betreiben. Allen ist dann jedoch gemeinsam, dass GitLab nur einen administrativen Service-Account für Konfiguration und Überwachung dieser Cluster nutzt. Auch Canary Deployments sind so möglich.

Da jede Umgebung auch einige – natürlich möglichst wenige – Unterschiede aufweist, enthält GitLab eine Verwaltung für Umgebungsvariablen, die es auch vor neugierigen Blicken schützt. Nur wer ausreichende Berechtigungen hat, kann diese Variablen setzen und einsehen. So ist es etwa üblich, die Token oder Passwörter für die von der Anwendung genutzten Dienste in jeder Umgebung unterschiedlich zu halten. Die Skripte für die Jobs der Pipelines greifen dann auf diese Variablen zurück und so kann ein Entwickler nicht direkt deren Inhalte sehen. Allerdings kann, wer die Pipelines kontrolliert, auch diese Inhalte einsehen, somit ist bei der Berechtigungsvergabe darauf zu achten, dass nur wenige vertrauenswürdige Personen im Team die Jobs und die Variablen editieren können.



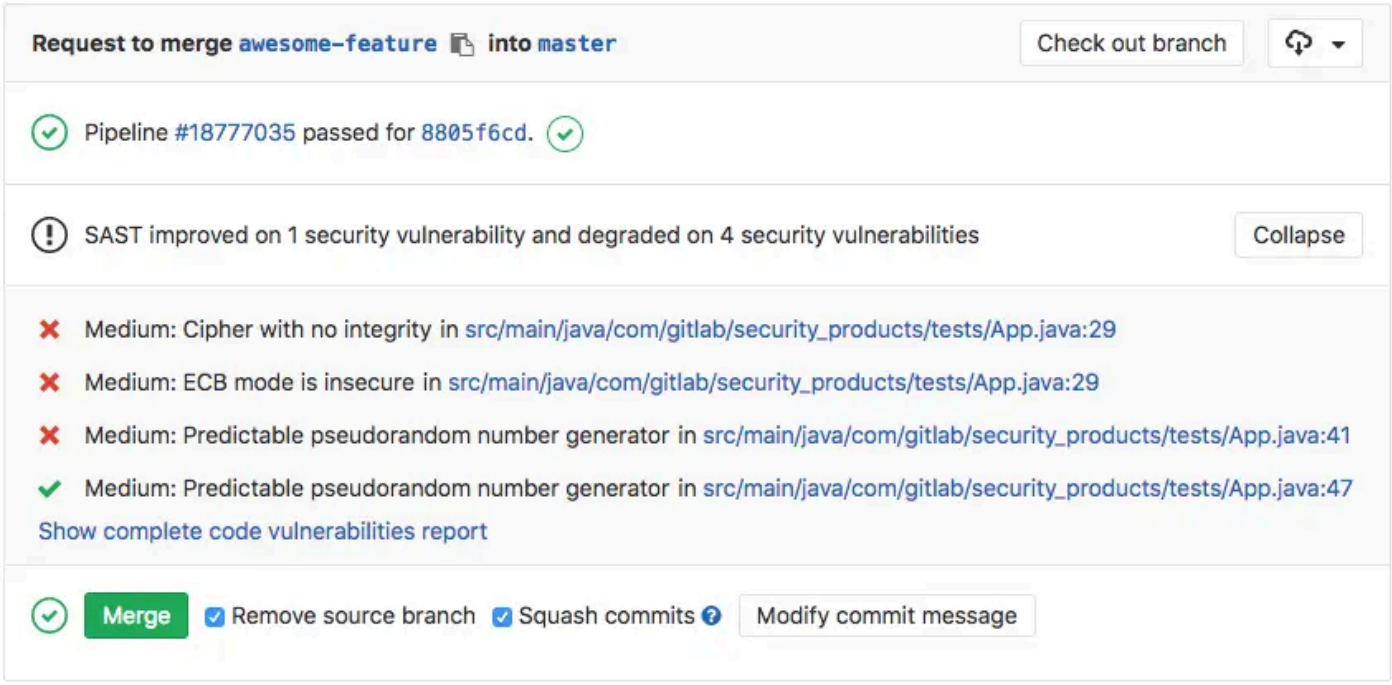
Nach einem Commit in den Master Branch startet automatisch, sofern nichts anderes hinterlegt ist, die Pipeline Auto DevOps. Bei einfachen Projekten läuft diese direkt durch, komplexere Projekte erfordern Anpassungen.

Image-Registry mit integriertem Sicherheitsscan

Alle aus den Projekten entstehenden Images landen in der Registry von GitLab. Die ist zwar rudimentär, bietet aber alle wichtigen Funktionen. So startet – wenn gewünscht – ein Scan der Images nach Sicherheitsproblemen aus der Pipeline heraus und die Images sind mit den gleichen Berechtigungen wie der Code geschützt. Sofern Code oder Images für nicht von GitLab gesteuerte Deployments bereitstehen sollen, wird ein Token vergeben, mit dem die Authentifizierung und Autorisierung erfolgt.

Immerhin lässt sich einstellen, wie lange die Registry die Images aufhebt. Dies ist auch in Abhängigkeit zu den vergebenen Tags konfigurierbar. So bleiben Releases erhalten und Debug- oder Test-Images sind schnell verschwunden.

Ebenfalls in GitLab enthalten sind diverse Scans, die die Sicherheit der Anwendungen und des Betriebs erhöhen. Die Scans sind in die CI/CD-Pipelines zu integrieren und **im Template "Auto DevOps" bereits enthalten [8]**. Aus den Ergebnissen erstellt GitLab eine Übersicht im Security-Dashboard, aber auch der Merge Request, der die Codeänderungen einbringen soll, zeigt einen Vorher-nachher-Vergleich.



Der SAST-Scanner zeigt Schwachstellen im Code und einen automatisch erstellten Merge Request. (Bild: GitLab.com)

Der Test der Codequalität erfolgt mit Code Climate, das bereits vollständig konfiguriert enthalten ist. Schon während der Programmierung kontrolliert der integrierte Dependency Scanner, ob der Code über Bibliotheken bekannte Schwachstellen importiert. Vor dem Bau der Anwendung kann der SAST-Scanner den Code auf typische Programmierfehler untersuchen. Nach dem Test-Deployment von Webanwendungen prüft ein DAST-Scanner diese auf per Web ausnutzbare Lücken. Sofern aus dem Projekt auch ein Container gebaut wird, überprüft GitLab diesen mit Clair und Klar auf bekannte Schwachstellen. Die Automatisierung der Compliance Checks runden ein Scan nach den Lizenzen der Bibliotheken und das Compliance-Dashboard ab, das eine Übersicht der letzten Merges und deren Genehmigung liefert, inklusive der Information, wer diese erteilt hat.

Diese Sicherheitsscans sind in der Pipeline "Auto DevOps" enthalten. Diese ist auf Wunsch für alle Projekte aktiv, sofern die keine eigene Konfiguration haben. Im Test war Auto DevOps allerdings nur selten ohne Anpassungen nutzbar und brach entweder beim Build oder bei einer der nächsten Stages ab. Getestet wurde das mit zwei selbst geschriebenen Anwendungen (einem IDp in PHP und einem TG-Bot in JS) sowie zwei fremden Projekten (F-Droid und WordPress). Von den vieren kam die Auto-DevOps-Pipeline beim TG-Bot am weitesten, allerdings gab es auch hier den Fehler, dass die gefundenen Schwachstellen und Probleme im Code nicht in den Dashboards oder Issues auftauchten. Wenn dies funktioniert, werden aus gefundenen Problemen automatisch Issues und Merge Requests. Ein Support-Call beim Hersteller war zum Redaktionsschluss noch offen.

Web IDE, Wiki und Issue Board

Wer schnell ein paar Änderungen machen möchte oder eine mobile IDE benötigt, die überall funktioniert, findet mit der in GitLab eingebauten Web IDE eine nicht ideale, aber benutzbare Option. Immerhin sind sehr viele Templates vorhanden, die für Betriebsthemen wie Pipelines und Codescans brauchbare Startpunkte setzen.

Ähnliches ist über das Wiki und das Issue Board zu sagen. Es funktioniert, hat die meisten benötigten Funktionen und ist für den Einstieg brauchbar. Wer größere Projekte planen und dokumentieren möchte, wird wohl eher eine dedizierte Software dafür einsetzen, die auch über Scrum oder Kanban-Boards mit Backlog und Analysen verfügt.

Multi-Cluster-Management

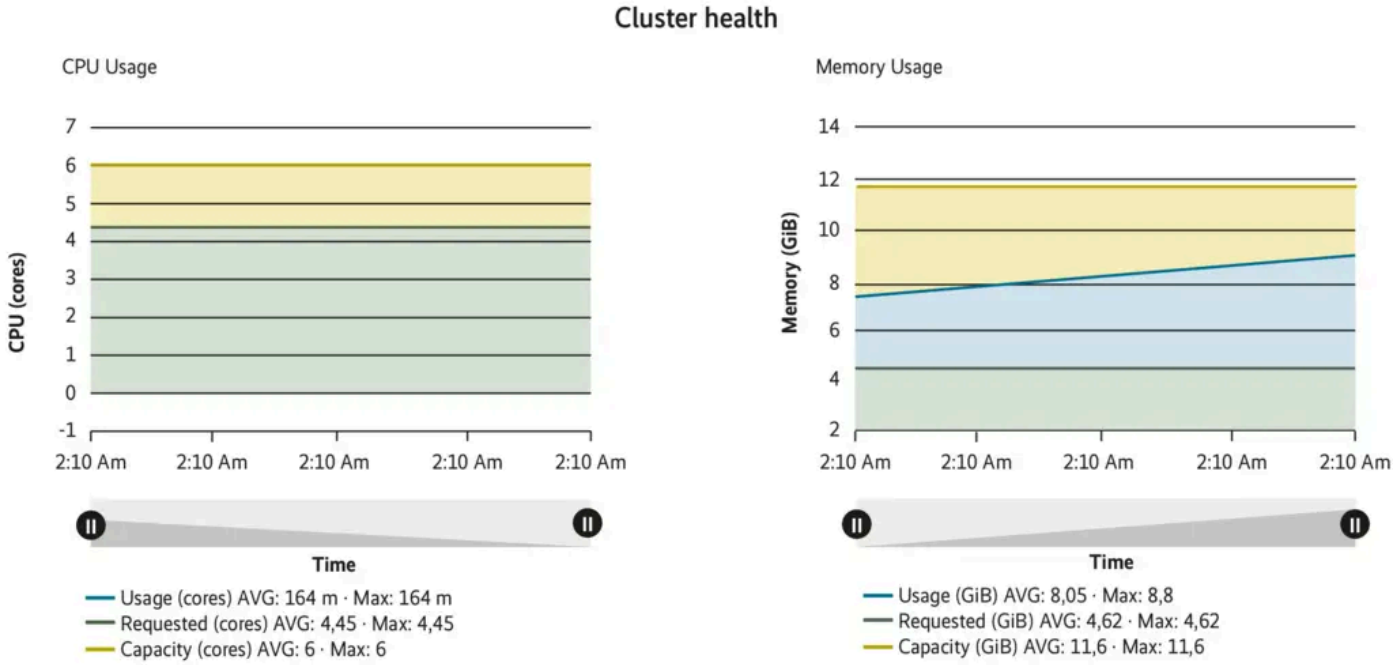
In der Administration bietet GitLab die Option, entweder einen Kubernetes-Cluster bei AWS oder GCE zu erzeugen oder einen existierenden anzubinden. Um selbst Cluster einzurichten, benötigt GitLab einen Benutzer mit den erforderlichen Berechtigungen bei diesen Diensten. Dies wurde nicht getestet, sondern stattdessen mit wenigen Klicks ein manuell erstellter Cluster angebunden. Ein regulärer Ausdruck legt dabei fest, welche Umgebungen auf diesem Cluster laufen dürfen. Die Umgebungen richten die Projekte allerdings selbst ein. Wer genauer steuern will, welches Projekt auf welchem Cluster läuft, muss das zuordnen. Aber auch die Projekte selbst können für sich eigene, exklusiv nutzbare Cluster hinzufügen.

GitLab fordert danach zum Installieren von Helm auf, mit dem dann die weiteren Dienste auf den Cluster kommen. GitLab Runner ist essenziell, damit GitLab auf dem Cluster Jobs ausführen kann. Gibt man dem Cluster eine Base Domain mit, ist er auch über DNS adressierbar und man kann dort seine Anwendungen für Tester oder Kunden bereitstellen. Für das Überwachen des Clusters bietet GitLab an, Prometheus und ELK zu installieren sowie Knative für Functions, Crossplane für das Verwalten von Secrets und CertManager für das Management von Let’s-Encrypt-Zertifikaten. JupyterHub ist auch als Helm Chart vorbereitet; eine Spezialität ist hier, dass ein Projekt in GitLab auch die Verwaltungsberechtigung für den Cluster erhalten kann, also eine Gruppe von Administratoren potenziell die Konfiguration aller Dienste des Clusters übernimmt.

Im Test liefen diese Funktionen weitgehend reibungslos, wenn die Installationen in der vorgegebenen Reihenfolge erfolgten und man dem Cluster etwas Zeit zwischen den Installationen ließ. Auch müssen die Nodes ausreichend groß sein, um alle Managementdienste plus die eigentlichen Aufgaben starten zu können.

Ist alles installiert, bietet GitLab aus dem Projekt heraus eine Sicht auf die Ausführung in den unterschiedlichen Umgebungen: Traces, Logfiles der Pods, Fehlermeldungen und eine gute Auswahl von Metriken. Enthalten sind Response-Zeiten, HTTP-Fehlercodes, CPU- und Speicherauslastung der Nodes. Auch hier liefert beispielsweise StackDriver zwar mehr Details, aber für eine Entwicklungsumgebung ist die Auswahl schon treffend und hilfreich. Schönes Detail: Sofern der Cluster es erlaubt, wird beim Installieren von Istio die öffentliche IP-Adresse direkt mit erstellt und eingetragen. Gleiches gilt für die Knative-Installation.

Bei der Verwaltung der Cluster gab es dann den einzigen Fehler 500 während des Tests. Der Cluster ließ sich nicht entfernen, da er nicht mehr verfügbar war und die installierten Dienste nicht deinstallieren konnte. Nur das Löschen der Integration war dann erfolgreich, aber auch nur bei einer aktiven Verbindung. Generell ist darauf zu achten, ein nicht ablaufendes Service-Token einzutragen; das ist eigentlich offensichtlich, fällt aber vielleicht nicht sofort jedem auf.



In einer grafischen Übersicht fasst GitLab den Status des Clusters zusammen.

Fazit

GitLab versucht alles zu sein und zu können. Zwar gelingt das auch weitgehend, aber zulasten des Funktionsumfangs in den einzelnen Bereichen: Fast immer existieren andere Werkzeuge, die etwas bis sehr viel mehr können. Allerdings sind diese dann selten wirklich gut integriert, und eine Arbeitsumgebung aus mehreren Werkzeugen zu bauen, ist oft sehr aufwendig. An einigen Stellen würde man sich auch etwas mehr Integration wünschen, etwa das Wiki mit den Releases des Codes zusammen zu versionieren oder auf einzelne Issues einfacher zu referenzieren

Wer also gerne innerhalb einer Oberfläche seine Anwendungen schreiben, verwalten, deployen und auch überwachen möchte, ist mit GitLab sehr gut bedient. Ob man damit auch die Produktionsumgebung betreiben und überwachen will, ist abhängig von den Wünschen an den Betrieb. Möglich ist es durchaus. Die Kosten dafür sind auch dementsprechend hoch (monatlich 99 US-Dollar pro User). Die Arbeitsersparnis, weil beispielsweise die Berechtigungen auf den Code automatisch auch für die Container-Images gelten und man sich nur um ein Werkzeug kümmern muss, kann dies wohl in den meisten Fällen ausgleichen. Der ständige Medienbruch – unausweichlich, wenn die Toolchain aus einem Dutzend unterschiedlicher Tools besteht – ist im Alltag allerdings oft sehr hinderlich.

iX-Wertung

GitLab Enterprise Edition 12.8.1-e Ultimate, eine integrierte Software für DevOps-Umgebungen

faktisch alle für DevOps benötigten Fähigkeiten in einer Software

alle Funktionen sind praxisnah und gut bedienbar

integrierte Sicherheitsscanner und Berechtigungsmodelle

- im Vergleich zu Spezialsoftware teils geringere Detailtiefe

Christoph Puppe ist Security Consultant bei SVA, Auditteamleiter für ISO 27001 nach Grundschutz, Mitautor des Grundschutz-Kompendiums und ehemaliger Penetrationstester. (**avr** [9])

URL dieses Artikels:

https://www.heise.de/-4735485

Links in diesem Artikel:

[1] https://www.heise.de/ratgeber/GitLab-mit-passender-Continuous-Integration-Pipeline-automatisiert-aufsetzen-4224990.html
[2] https://www.heise.de/ratgeber/GitLab-CI-sowie-CD-installieren-und-konfigurieren-4340888.html
[3] https://www.heise.de/tests/GitLab-Enterprise-Ultimate-als-DevOps-Toolchain-fuer-Entwickler-4735485.html
[4] https://about.gitlab.com/blog/2020/03/30/new-features-to-core/
[5] https://about.gitlab.com/pricing/self-managed/feature-comparison/
[6] https://www.heise.de/ix/
[7] https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/version_id-224335/Wordpress-Wordpress-4.8.1.html
[8] https://gitlab.com/gitlab-org/gitlab/blob/master/lib/gitlab/ci/templates/Auto-DevOps.gitlab-ci.yml
[9] mailto:avr@ix.de