Renovate hält die Dependencies frisch

Um Software sicher und Updates schlank zu halten, müssen Abhängigkeiten regelmäßig auf neue Versionen geprüft werden. Mit dem Renovate Bot geht das automatisch.

Von Danny Steinbrecher

Ohne Abhängigkeiten (Dependencies) geht es in der Softwarewelt kaum mehr. Fertige Pakete von außerhalb sparen zwar Entwicklungszeit, bergen jedoch die Gefahr leicht zu übersehender Schwachstellen, die als Common Vulnerabilities and Exposures (CVEs) geführt werden. Abhängigkeiten erfordern also einen gewissen Wartungsaufwand. Das eindrücklichste Beispiel für die tickende Zeitbombe, die vernachlässigte Dependencies darstellen, ist der Vorfall um das Java-Logging-Framework Log4j im Jahr 2021. Von Katastrophen wie dieser abgesehen, müssen Updates und Bugfixes aber auch deshalb regelmäßig nachgeladen werden, weil bei größeren Versionssprüngen die Gefahr besteht, dass Teile der Software inkompatibel werden – aufwendiges Nachjustieren ist die Folge. Um so eine Regelmäßigkeit gewährleisten zu können, automatisiert man am besten: Das kann der Renovate Bot leisten.

Der Renovate Bot, den der Anbieter des Tools mend.io eigentlich Mend Renovate getauft hat, kümmert sich um die Dependencies innerhalb einer Anwendung. Er schaut nach, auf welchem Stand sie sind, und vergleicht sie mit den neusten Versionen. Zusätzlich gibt es noch eine Einschätzung, wie groß das Update sein würde – von Patch (v1.0.0 -> 1.0.1) über Minor (v1.0.0 -> 1.1.0) bis zu Major (v1.0.0 -> 2.0.0). Anschließend erstellt Renovate einen Pull Request in GitHub, der je nach Konfiguration automatisch oder manuell gemergt wird. Den Prozess kann man auf die Anwendung zuschneiden, sodass Renovate Bot beispielsweise unkritische Minor Updates automatisch und größere Major Updates manuell mergt.

Das Konzept ist nicht neu: Schon 2019 hat GitHub den 2017 entwickelten Dependabot übernommen, der seitdem die Dependencies der Open-Source-Welt auf Updatebedarf abklopft. Im Vergleich spricht für Renovate, dass das Programm flexibler ist. Es hat mehr Konfigurationsmöglichkeiten und ist besser in andere Plattformen neben GitHub integriert, darunter Azure DevOps und Bitbucket. Der Dependabot ist hingegen auf GitHub spezialisiert und wird ständig um neue Funktionen erweitert. Der Einsatz der beiden Bots fühlt sich durch Renovates Automerge-Feature direkt über eine Konfigurationsdatei unterschiedlich an – Renovate nimmt Entwicklern nach der Konfiguration etwas mehr Arbeit ab.

Onboarding

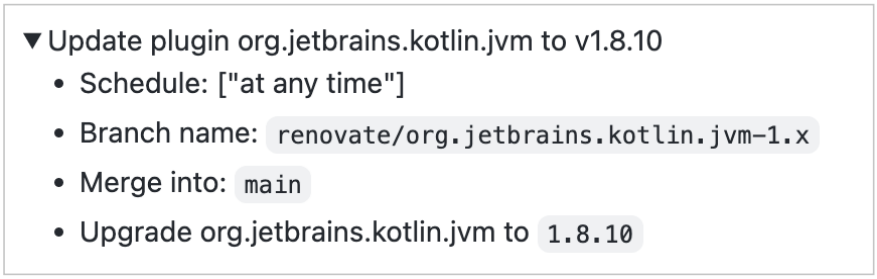
Als Beispiel für seinen Einsatz im eigenen Projekt dient eine Anwendung, die in GitHub liegt (siehe ix.de/zj3b). Am Ende steht dann eine Konfigurationsdatei, die auch eine gute Grundlage für den Einsatz im eigenen Projekt bilden sollte. Doch zunächst steht der Einrichtungsprozess von Renovate in GitHub an. Der erfolgt über eine Installation der MEND-App, die sich im GitHub-Verzeichnis von Renovate findet. Sie wird mit einem Klick auf den Configure-Button gestartet, anschließend leitet die Installation durch die Ersteinrichtung – bis zum ersten Pull Request.

Bei näherer Betrachtung fällt auf, dass der erste Pull Request dabei keine Dependencies aktualisiert, sondern als Onboarding fungiert: Zum einen erzeugt er eine renovate.json-Konfigurationsdatei und zum anderen enthält der erzeugte GitHub Pull Request einen ausführlichen Kommentar, der auch alle erkannten Package-Dateien auflistet. In denen schaut Renovate nach Dependencies und prüft deren Version. Hier lässt sich bereits erkennen, ob der Bot Dependencies berücksichtigt oder ob man gegebenenfalls unerkannte Pakete hinzufügen muss. Kubernetes-Dateien sind solche Fälle, ihre speziellen Anforderungen werden an späterer Stelle noch einmal beschrieben. Der zweite Abschnitt des Kommentars gibt einen kleinen Überblick darüber, was Renovate in der Standardkonfiguration macht. Im anschließenden „What to Expect“-Abschnitt steht, welche Dependencies im Detail nicht auf dem neusten Stand sind und auf welche Version sie aktualisiert werden.

iX-TRACT

- Fast jedes Softwareprojekt nutzt Open-Source-Bibliotheken und kann sich schnell Sicherheitslücken und Bugs einhandeln, wenn diese Pakete nicht mehr auf dem neuesten Stand sind.
- Der kostenlos nutzbare Renovate Bot automatisiert Aktualisierungen von Dependencies.
- Renovate zeichnet sich durch einen größeren Konfigurationsspielraum aus. Updates werden in Form von Pull Requests durchgeführt, auch ein automatisches Merge ist möglich.
- Der Artikel zeigt, wie man Renovate sinnvoll einrichtet. Am Ende steht eine Konfiguration, die eine gute Grundlage für eigene Projekte ist.

Die einzelnen Updates kann man sich außerdem im Detail anschauen (Abbildung 1) und erhält dabei wertvolle Hinweise für die spätere Konfiguration: Neben dem Schedule, also der Einstellung, wie oft der Renovate Bot die Dependencies prüft, dem Branch Name, in dem der Pull Request erstellt wird, und der Version, auf die aktualisiert wird, findet sich auch Merge into. Alle Anpassungen landen standardmäßig im Main Branch.



Der Kommentar des ersten Pull Request enthält bereits wertvolle Informationen zu den Updates (Abb. 1).

Konfiguration

Wechselt man in den Reiter Commits, sieht man, dass die Datei renovate.json angelegt wurde. Darin lässt sich der Bot passend zu den Anforderungen des Projektes anpassen. In Listing 1 ist die Standardkonfigurationsdatei zu sehen, die sich um einige Parameter erweitern lässt. Am einfachsten ist es, die Presets zu benutzen, die in der Dokumentation von Renovate zu finden sind, darunter sind Default-, Monorepo-, Package- und Schedule-Presets. Hat man ein passendes Preset gefunden, kann es in das `extends` `Array` der JSON-Datei eingebunden werden. In der Standardkonfiguration ist ein Full-Config-Preset (`config:base`) enthalten. Das Beispielpaket enthält mehrere Presets (in Listing 1 zu sehen), die später überschrieben werden können. Sollen Aktualisierungen zum Beispiel zunächst in einem Dev-Branch landen, ergänzt man renovate.json um `"base Branches": ["dev"]`, wobei gegebenenfalls der Name des Branch angepasst werden muss.

Wenn die Konfiguration fertig ist, kann der Onboarding Pull Request gemergt werden. Gibt es Updates der Abhängigkeiten, erstellt das Tool nach ein paar Minuten zwei neue Pull Requests, aber auch ein neues Issue – das Dependency Dashboard Issue, das den aktuellen Stand des Renovate Bots zeigt und offen gehalten werden sollte. Es gibt einen Überblick darüber, welche Dependencies aktualisiert werden müssen und welche zurückgestellt wurden, und es bietet ein paar Interaktionsmöglichkeiten. Zum Beispiel könnte man damit Dependency-Updates neu anstoßen. Außerdem listet es noch einmal alle untersuchten Dependencies auf. Damit lässt sich prüfen, ob die Konfiguration gewisse Dependencies ausgeschlossen hat oder ob die Änderung an der Konfigurationsdatei erfolgreich war und jetzt neue Dependencies einbezieht. Ab jetzt erzeugt Renovate für jedes neue Update einer Dependency einen Pull Request – und optional eine E-Mail.

Sollte ein syntaktischer Fehler in der Konfiguration auftauchen, hilft Renovate, indem es ein neues Issue anlegt, das von einer fehlerhaften Konfiguration berichtet und zeigt, welcher Eintrag betroffen ist. Bis zur Behebung des Fehlers arbeitet Renovate nicht weiter. Außerdem ist die Reihenfolge der Presets wichtig: Weiter unten stehende Presets überschreiben solche, die weiter oben stehen. Damit lassen sich zum Beispiel die Presets überschreiben, die in den Full-Config-Presets enthalten sind.

Noise Reduction

Je nach Set-up kann es vorkommen, dass Renovate jede Stunde eine Benachrichtigung erzeugt, die darauf hinweist, dass zwei neue Updates vorhanden sind – auch wenn der Renovate Bot nur in einem einzigen Projekt eingesetzt wird. Das ist nicht nur nervig, sondern kann sich auch zu einem echten Problem auswachsen. Denn zu viele Benachrichtigungen haben den gleichen Effekt wie keine. Wichtige Updates, die unter Umständen einen Fix für eine CVE beinhalten, können in einer Flut von unwichtigen Benachrichtigungen untergehen. Renovate hat ein paar Ansätze parat, um den sprichwörtlichen Lärm zu reduzieren, darunter vier essenzielle: das Gruppieren von Dependencies, die Anpassung des Pull-Request-Limits, ein Scheduling des Renovate Bots und Automerging. Dafür muss man die renovate.json-Datei erweitern.

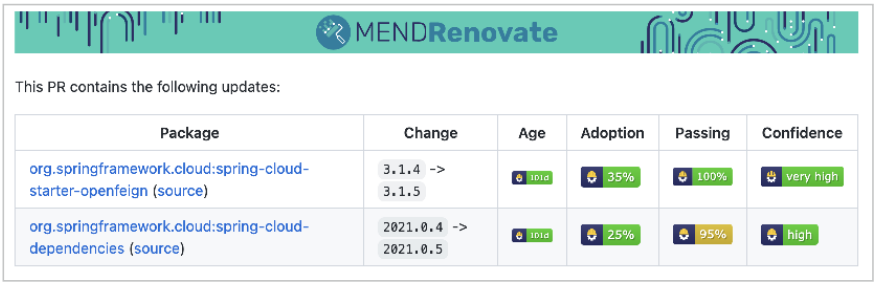
Die einfachste Option ist bereits in der Standardkonfiguration enthalten: Innerhalb des `config:base`-Presets befindet sich das `group:monorepos`-Preset (Listing 2). Dadurch werden Dependencies, die in Abhängigkeit zu anderen stehen, in einen einzigen Pull Request zusammengeführt (Abbildung 2). Die Gruppierung erfolgt nach verschiedenen Regeln, die in dem bereits eingebauten Preset enthalten sind. Darunter fallen Gruppierungen für Android, Angular, AWS, JUnit, Kotlin, Rust und einige mehr. Das Preset legt also bereits eine valide Grundlage für Gruppierungen.

Sollte das `config:base`-Preset nicht zum Einsatz kommen, können die einzelnen Gruppierungen auch manuell eingerichtet werden (siehe ix.de/zj3b). Die Gruppierungen kann man auch noch deutlich detaillierter vornehmen. Zum Beispiel könnte man über die `package Rules` eigene Gruppierungen anlegen und damit Dependencies, die aus verschiedenen Repositorys und von verschiedenen Autoren stammen, zusammenführen. Dafür muss man die `matchPackage Patterns` ausfüllen. Dabei ist aber Vor-

Listing 1: renovate.json ist bereits mit einigen Presets ausgestattet

```
{
  "$schema": "https://docs.renovatebot.com/renovate-schema.json",
  "baseBranches": [
    "dev"
  ],

  "extends": [
    ":dependencyDashboard",
    ":semanticPrefixFixDepsChoreOthers",
    ":ignoreModulesAndTests",
    ":autodetectRangeStrategy",
    ":prHourlyLimit2",
    ":prConcurrentLimit10",
    "group:monorepos",
    "group:recommended",
    "workarounds:all"
  ]
}
```



Um nicht in einer Benachrichtigungsflut zu ersticken, sollte man Dependencies in einem gemeinsamen Pull Request vereinen (Abb. 2).

sicht geboten, es ergeben sich nämlich durchaus ein paar Nachteile: Setzt man Repository-übergreifende Gruppierungen ein, dann erhöht sich auch die Wahrscheinlichkeit, dass der Merge einen Konflikt hat. Das liegt dann vielleicht nur an einer einzigen Dependency innerhalb der Gruppierung, aber die Detektivarbeit kostet Zeit. Es kann auch sein, dass man gar nicht alle Dependencies einer Gruppierung aktualisieren möchte. Gruppierungen sollten also mit Bedacht eingesetzt werden.

Mehr Updates bitte!

Ein weiterer Weg, die Anzahl an Benachrichtigungen zu reduzieren, ist es, mehr gleichzeitig erstellte Pull Requests zu erlauben. In der Standardkonfiguration erzeugt der Renovate Bot maximal zwei Requests zur gleichen Zeit, auch wenn es mehr als diese beiden gibt. Sobald er sie gemergt hat, legt er nach einer Stunde zwei Pull Requests nach. Bei 20 verschiedenen Dependency-Updates kämen also zehn Stunden lang jede Stunde zwei Benachrichtigungen an.

Mit der Konfigurationsoption `prHourlyLimit` kann man das Verhalten jedoch anpassen. Sie lässt sich in der Konfiguration ergänzen. Als Parameter erhält sie die Anzahl an gleichzeitigen Pull Requests. Übergibt man den Wert `0`, fällt das Limit weg und Renovate erzeugt alle nötigen Pull Requests auf einmal.

Hier gibt es auch wieder Default-Presets, die man in das `extends`-Array einbindet. Das Preset `prHourlyLimitNone` weist das gleiche Verhalten auf wie `prHourlyLimit: 0`. Mit `prHourlyLimit: 1` wird immer nur ein Pull Request pro Stunde erzeugt. Alternativ gibt es noch `prHourlyLimit: 2` und `prHourlyLimit: 4`. In diesem Beispiel steht die Anzahl der Pull Requests auf 8.

Zeitplan

Wer alle oder einen Teil der Dependencies nicht stündlich, sondern beispielsweise täglich scannt, reduziert die Benachrichtigungen des Renovate Bots weiter. Dazu muss man die Regeln um die `schedule`- und `timezone`-Parameter erweitern (Listing 3). Zunächst wird die Zeitzone nach dem IANA-Standard festgelegt. Für Schedule gibt es wieder ein paar Optionen. Am einfachsten sind die Standard-Presets wie `daily`, `weekly`, `monthly` oder `nonOfficeHours`. Es besteht aber auch die Option, sich eigene Presets zu erzeugen. Im Beispiel soll der Renovate Bot nur an Werktagen laufen und bis Mittag fertig sein. So besteht nicht die Gefahr, am Wochenende oder kurz vor Feierabend Benachrichtigungen zu erhalten, die dann später in Vergessenheit geraten. Auch hier ist es wieder möglich, über die `packageRules` Gruppen oder einzelne Dependencies anderen Zyklen zuzuordnen. Renovate testet unwichtigere Dependencies dann seltener – und kritischere öfter.

Automerging

Wenn schon automatisieren, dann doch bitte richtig. Warum soll Renovate überhaupt noch Benachrichtigungen senden, wenn es einen Pull Request erstellt hat? Wäre ein direkter, automatischer Merge nicht praktischer? Um so eine hohe Automatisierungsstufe zu erreichen, muss die Anwendung ein paar wichtige Voraussetzungen erfüllen: Nur wenn sie eine gute Testabdeckung und eine gut funktionierende CI-Pipeline besitzt, sollte man ihr das Automergen zutrauen. Das gilt allgemein für den Renovate Bot, aber insbesondere für das Automerging. Um es zu aktivieren, muss man den bestehenden Code um die entsprechenden `packageRules` erweitern und drei Parameter darin einbinden (Listing 4).

Die erste Zeile, `"automerge": true`, aktiviert die Automatisierung, sorgt aber noch nicht für weniger Benachrichtigungen. Renovate erstellt weiterhin Pull Requests, die es lediglich automatisch mergt. Beides erzeugt immer noch Benachrichtigungen. Um diese zu reduzieren, ist zusätzlich die Konfiguration von `"automergeType": "branch"` erforderlich. Sie sorgt dafür, dass der Renovate Bot keinen Pull Request erstellt, sondern einen eigenen Branch eröffnet. Wenn die Tests mit den neuen Dependencies durchlaufen, erstellt er automatisch ei-

Listing 2: Voneinander abhängige Dependencies können in einem einzigen Pull Request gruppiert werden

```
{
  "$schema": "https://docs.renovatebot.com/renovate-schema.json",
  "extends": [
    "config:base"
  ]
}
```

Listing 3: Ein Zeitplan regelt die Scans, die Renovate durchführt

```
{
  ...
  "timezone": "Europe/Berlin",
  "schedule": [
    "after 5am every weekday",
    "before 1pm every weekday"
  ]
}
```

Listing 4: Automerging erspart Arbeit, sollte aber mit Bedacht eingesetzt werden

```
{
  ...
  "packageRules": [
    {
      "automerge": true,
      "automergeType": "branch",
      "matchUpdateTypes": [
        "minor",
        "patch"
      ]
    }
  ]
}
```

Listing 5: Mit dem Kubernetes-Tag erkennt Renovate auch K8s-Dateien

```
{
  ...
  "kubernetes": {
    "fileMatch": [
      "\\.yml$",
      "\\.yaml$"
    ]
  }
}
```

Listing 6: Richtig konfiguriert arbeitet Renovate dauerhaft sehr zuverlässig

```
{
  "$schema": "https://docs.renovatebot.com/renovate-schema.json",
  "extends": [
    "config:base"
  ],
  "prHourlyLimit": 8,
  "timezone": "Europe/Berlin",
  "schedule": [
    "after 5am every weekday",
    "before 1pm every weekday"
  ],
  "packageRules": [
    {
      "automerge": true,
      "automergeType": "branch",
      "matchUpdateTypes": ["minor", "patch"]
    }
  ],
  "kubernetes": {
    "fileMatch": ["\\.yml$", "\\.yaml$"]
  }
}
```

nen Commit in den Main Branch. Nur wenn ein Test fehlschlägt, erzeugt er einen Pull Request. So entsteht nur noch eine Nachricht – dass ein Commit erstellt wurde –, und die ist so wichtig, dass man sie nicht ausschalten sollte. Dadurch herrscht immer Klarheit darüber, dass es ein Update gab. Zuletzt legt man noch fest, welcher Updatetyp automatisch gemergt werden soll – sinnvoll ist das für Patch- und Minor Updates mit der Konfiguration `"matchUpdateTypes": ["minor", "patch"]`. Major Updates könnten Änderungen beinhalten, die die Software unbrauchbar machen, und sollten lieber manuell gemergt werden.

Einsatzfeld Kubernetes

Wie eingangs erwähnt, gibt es Dependencies, die Renovate nicht ohne Weiteres erkennt. Kubernetes ist ein solcher Fall. Existieren in einem Projekt Kubernetes-Manifeste, findet man sie weder im Onboarding Pull Request noch auf dem Dependency-Dashboard. Da Kubernetes-Dateien anhand ihres Dateinamens und dessen Endung nicht eindeutig zu erkennen sind, müssen sie für Renovate explizit konfiguriert werden. Dafür fügt man den Kubernetes-Tag zur renovate.json-Datei hinzu (Listing 5). Die Option `fileMatch` gibt an, dass alle Dateien mit der Endung `.yml` und `.yaml` Kubernetes-Dateien sind. Optional könnte hier auch ein Pfadname stehen, wenn sich die betreffenden Dateien nur in einem speziellen Ordner befinden.

Jetzt scannt Renovate auch Kubernetes-Dependencies und aktualisiert sie bei Bedarf. Das gilt jedoch nur für Dependencies, die Kubernetes aus der Registry lädt. Zum heutigen Zeitpunkt werden Versionen der Kubernetes-Ressourcen nicht erkannt. Dabei wäre es wünschenswert, wenn der Bot von `apiVersion: extensions/v1beta1` auf `apiVersion: networking.k8s.io/v1` aktualisieren würde oder wenigstens einen Hinweis auf die veraltete Version gäbe.

Fazit

Mit der Hilfe von Renovate lassen sich Bugs sowie Common Vulnerabilities and Exposures durch notwendige Updates verhindern, die das Programm schnell und in kleinen Schritten einarbeiten kann. Die Automatisierung dieser Aufgabe spart mit der richtigen Konfiguration Zeit. Wie immer ist es dabei aber wichtig, die goldene Mitte zu finden, damit das Tool seinen Einsatzzweck erfüllt und bei der Instandhaltung der Software tatsächlich unterstützt – und nicht stattdessen zusätzliche Probleme bereitet. Dafür bietet Renovate umfangreiche Konfigurationsmöglichkeiten an, die am Anfang etwas abschreckend wirken können. Mit der abschließenden Konfiguration aus dem Artikel, die in Listing 6 zu sehen ist, steht aber schon eine gute Grundlage bereit, um die Auffrischung der Abhängigkeiten in eigener Software zu automatisieren. (kki@ix.de)

Quellen

Ressourcen für die Konfiguration sind unter ix.de/zj3b zu finden

DANNY STEINBRECHER

ist IT-Consultant und Softwareentwickler bei der codecentric.