



Grundlegendes zu GitHub Actions

Hier erfährst du mehr über die Grundlagen von GitHub Actions, einschließlich der Kernkonzepte und wesentlichen Terminologie.

In diesem Artikel

Übersicht

Die Komponenten von GitHub Actions

Erstellen eines Beispielworkflows

Grundlegendes zu Workflowdateien

Anzeigen der Aktivität für eine Workflowausführung

Nächste Schritte

Übersicht

GitHub Actions ist eine Plattform für Continuous Integration und Continuous Delivery (CI/CD), mit der du deine Build-, Test- und Bereitstellungspipeline automatisieren kannst. Du kannst Workflows erstellen, mit denen du alle Pull Requests für dein Repository erstellen und testen sowie gemergte Pull Requests für die Produktion bereitstellen kannst.

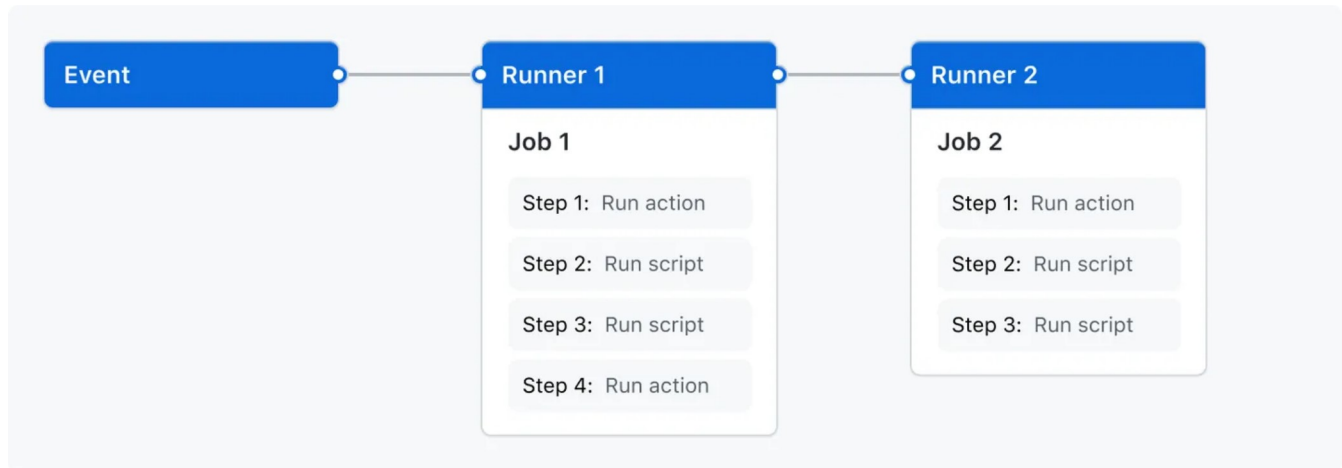
GitHub Actions ist nicht auf DevOps beschränkt und kann auch für andere Ereignisse in deinem Repository Workflows ausführen. So kannst du z. B. einen Workflow für das automatische Hinzufügen geeigneter Bezeichnungen ausführen, sobald in deinem Repository ein neues Issue erstellt wird.

GitHub stellt virtuelle Linux-, Windows- und macOS-Computer für die Ausführung deiner Workflows bereit. Alternativ dazu kannst du eigene selbstgehostete Runner in deinem Rechenzentrum oder deiner Cloudinfrastruktur hosten.

Die Komponenten von GitHub Actions

Du kannst konfigurieren, dass bei einem *Ereignis* in deinem Repository ein GitHub Actions-*Workflow* ausgelöst wird. Dabei kann es sich etwa um das Öffnen eines Pull Requests oder das Erstellen eines Issues handeln. Der Workflow enthält einen oder mehrere *Aufträge*, die nacheinander oder gleichzeitig ausgeführt werden können. Jeder Auftrag wird innerhalb eines eigenen *Runners* der VM oder in einem Container ausgeführt und verfügt über einen oder mehrere *Schritte*. Diese führen entweder ein von Ihnen definiertes Skript oder eine *Aktion* aus. Dabei handelt es sich um eine wiederverwendbare Erweiterung zur Vereinfachung des

Dabei handelt es sich um eine wiederverwendbare Erweiterung zur Vereinfachung des Workflows.



Workflows

Ein Workflow ist ein konfigurierbarer automatisierter Prozess zur Ausführung eines oder mehrerer Aufträge. Workflows werden durch eine im Repository eingetragene YAML-Datei definiert. Die Auslösung ihrer Ausführung erfolgt durch ein Ereignis in deinem Repository, manuell oder nach einem definierten Zeitplan.

Workflows werden im Verzeichnis `.github/workflows` in einem Repository definiert, und ein Repository kann mehrere Workflows aufweisen, die jeweils eine andere Gruppe von Aufgaben ausführen können. So kannst du etwa einen Workflow zum Erstellen und Testen von Pull Requests erstellen, einen zweiten zum Bereitstellen deiner Anwendung bei jeder neuen Erstellung eines Releases und einen dritten zum Hinzufügen von Bezeichnungen bei jeder Öffnung eines weiteren Issues.

Du kannst innerhalb eines anderen Workflows auf einen Workflow verweisen. Weitere Informationen findest du unter [Wiederverwenden von Workflows](#).

Weitere Informationen zu Workflows findest du unter [Verwenden von Workflows](#).

Ereignisse

Bei einem Ereignis handelt es sich um eine bestimmte Aktivität in einem Repository, die die Ausführung eines Workflows auslöst. Die Aktivität kann beispielsweise von GitHub stammen, wenn ein Pull Request erstellt, ein Issue geöffnet oder ein Commit per Push in ein Repository übertragen wird. Die Ausführung eines Workflows kann auch nach einem [Zeitplan](#), durch [Posten in einer REST-API](#) oder manuell ausgelöst werden.

Eine vollständige Liste der Ereignisse zum Auslösen von Workflows findest du unter [Ereignisse, die Workflows auslösen](#).

Aufträge

Ein Auftrag umfasst mehrere *Schritte* in einem Workflow, die in demselben Runner ausgeführt werden. Jeder Schritt besteht entweder aus einem Shellskript oder aus einer *Aktion*, die ausgeführt werden. Die Schritte werden nacheinander ausgeführt und sind voneinander abhängig. Da alle Schritte im gleichen Runner ausgeführt werden, kannst du Daten eines Schritts für andere Schritte freigeben. So kann z. B. in einem Schritt eine Anwendung erstellt und im nächsten Schritt die erstellte Anwendung getestet werden.

Du kannst einen Auftrag so konfigurieren, dass er Abhängigkeiten zu anderen Aufträgen enthält. Standardmäßig verfügen Aufträge nicht über Abhängigkeiten und werden gleichzeitig ausgeführt. Verfügt ein Auftrag über eine Abhängigkeit zu einem anderen Auftrag, wird er erst nach dessen Abschluss ausgeführt. Du kannst z. B. mehrere Buildaufträge für unterschiedliche Architekturen ohne Abhängigkeiten erstellen und anschließend einen Auftrag zur Paketerstellung, der von diesen abhängig ist. Dadurch werden die Buildaufträge gleichzeitig ausgeführt, und der Auftrag zur Paketerstellung wird erst ausgeführt, nachdem diese erfolgreich abgeschlossen wurden.

Weitere Informationen zu Aufträgen findest du unter [Verwenden von Aufträgen](#).

Aktionen

Bei einer *Aktion* handelt es sich um eine benutzerdefinierte Anwendung für die GitHub Actions-Plattform zur Ausführung einer komplexen und häufig ausgeführten Aufgabe. Mit einer Aktion kannst du die Menge des Codes in deinen Workflowdateien reduzieren. Mit einer Aktion kannst du dein Git-Repository aus GitHub abrufen, die korrekte Toolkette für deine Buildumgebung einrichten oder die Authentifizierung bei deinem Cloudanbieter einrichten.

Du kannst eigene Aktionen schreiben oder im GitHub Marketplace nach Aktionen für deine Workflows suchen.

Weitere Informationen findest du unter [Erstellen von Aktionen](#).

Runner

Ein Runner ist ein Server, auf dem deine Workflows ausgeführt werden, wenn sie ausgelöst werden. Ein Runner kann immer nur einen Auftrag ausführen. GitHub stellt Runner für Ubuntu Linux, Microsoft Windows und macOS zum Ausführen von Workflows bereit. Jede Workflowausführung erfolgt in einer neu bereitgestellten VM. GitHub bietet zudem größerer Runner, die in größeren Konfigurationen verfügbar sind. Weitere Informationen findest du unter [Informationen zu großen Runnern](#). Wenn du ein anderes Betriebssystem oder eine bestimmte Hardwarekonfiguration benötigst, kannst du deine eigenen Runner hosten. Weitere Informationen zu selbstgehosteten Runnern findest du unter [Deinen eigenen Runner hosten](#).

Erstellen eines Beispielworkflows

In GitHub Actions werden Workflows mithilfe der YAML-Syntax definiert. Die einzelnen Workflows werden in deinem Coderepository jeweils als separate YAML-Datei in einem Verzeichnis mit dem Namen `.github/workflows` gespeichert.

Du kannst in deinem Repository einen Beispielworkflow erstellen, der immer dann automatisch eine Reihe von Befehlen auslöst, wenn Code per Push übertragen wird. In diesem Workflow checkt GitHub Actions den per Push übertragenen Code aus, installiert das Testframework [bats](#) und führt einen einfachen Befehl aus, um die bats-Version auszugeben: `bats -v`.

- 1 Erstelle in deinem Repository das Verzeichnis `.github/workflows/`, um die Workflowdateien zu speichern.
- 2 Erstelle im Verzeichnis `.github/workflows/` eine neue Datei mit dem Namen `learn-github-actions.yml`, und füge den folgenden Code hinzu:

YAML




```
name: learn-github-actions
run-name: ${{{ github.actor }}} is learning GitHub Actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```

- 3 Führe für diese Änderungen einen Commit aus, und übertrage sie per Push in dein GitHub-Repository.

Die neue GitHub Actions-Workflowdatei ist jetzt in deinem Repository installiert und wird immer dann automatisch ausgeführt, wenn eine Änderung per Push in dein Repository übertragen wird. Informationen zum Ausführungsverlauf eines Workflows findest du unter [Anzeigen der Aktivität einer Workflowausführung](#).

Grundlegendes zu Workflowdateien

In diesem Abschnitt wird anhand der einzelnen Zeilen des Einführungsbeispiels erläutert, wie du mithilfe der YAML-Syntax eine Workflowdatei erstellst.

YAMLBesideInline

```
name: learn-github-actions
```

Optional - The name of the workflow as it will appear in the "Actions" tab of the GitHub repository. If this field is omitted, the name of the workflow file will be used instead.

```
run-name: ${ github.actor } is learning GitHub Actions
```

Optional - The name for workflow runs generated from the workflow, which will appear in the list of workflow runs on your repository's "Actions" tab. This example uses an expression with the `github` context to display the username of the actor that triggered the workflow run. For more information, see "[Workflowsyntax für GitHub Actions](#)."

```
on: [push]
```

Specifies the trigger for this workflow. This example uses the `push` event, so a workflow run is triggered every time someone pushes a change to the repository or merges a pull request. This is triggered by a push to every branch; for examples of syntax that runs only on pushes to specific branches, paths, or tags, see "[Workflowsyntax für GitHub Actions](#)."

```
jobs:
```

Groups together all the jobs that run in the `learn-github-actions` workflow.

```
check-bats-version:
```

Defines a job named `check-bats-version`. The child keys will define properties of the job.

```
runs-on: ubuntu-latest
```

Configures the job to run on the latest version of an Ubuntu Linux runner. This means that the

job will execute on a fresh virtual machine hosted by GitHub. For syntax examples using other runners, see "[Workflowsyntax für GitHub Actions](#)"

```
steps:
```

Groups together all the steps that run in the `check-bats-version` job. Each item nested under this section is a separate action or shell script.

```
- uses: actions/checkout@v4
```

The `uses` keyword specifies that this step will run `v4` of the `actions/checkout` action. This is an action that checks out your repository onto the runner, allowing you to run scripts or other actions against your code (such as build and test tools). You should use the checkout action any time your workflow will use the repository's code.

```
- uses: actions/setup-node@v3
  with:
    node-version: '14'
```

This step uses the `actions/setup-node@v3` action to install the specified version of the Node.js. (This example uses version 14.) This puts both the `node` and `npm` commands in your `PATH`.

```
- run: npm install -g bats
```

The `run` keyword tells the job to execute a command on the runner. In this case, you are using `npm` to install the `bats` software testing package.

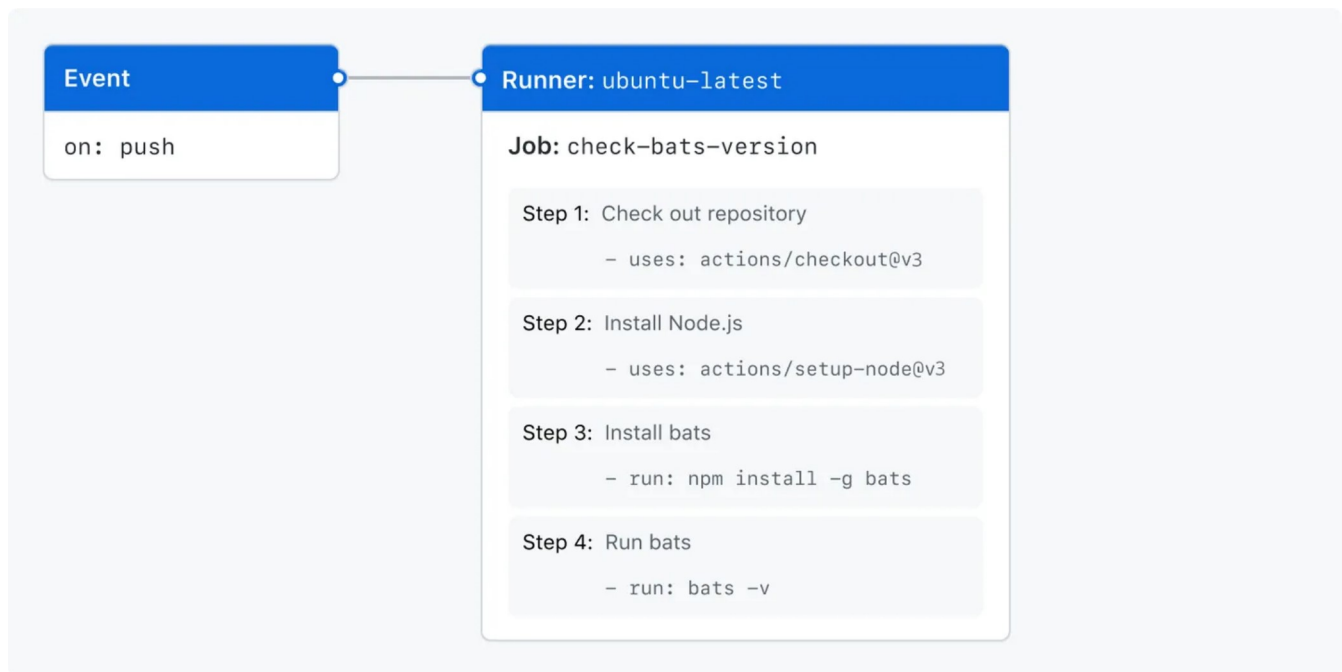
```
- run: bats -v
```

Finally, you'll run the `bats` command with a parameter that outputs the software version.

Visualisieren der Workflowdatei

Im folgenden Diagramm siehst du die zuvor erstellte Workflowdatei und die hierarchische Organisation der GitHub Actions-Komponenten. In jedem Schritt wird eine einzelne Aktion

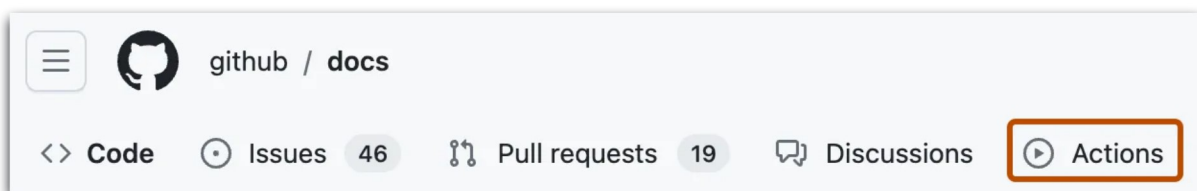
oder ein einzelnes Shellskript ausgeführt. In den Schritten 1 und 2 werden Aktionen ausgeführt, in den Schritten 3 und 4 Shellskripts. Weitere vordefinierte Aktionen für deine Workflows findest du unter [Suchen und Anpassen von Aktionen](#).



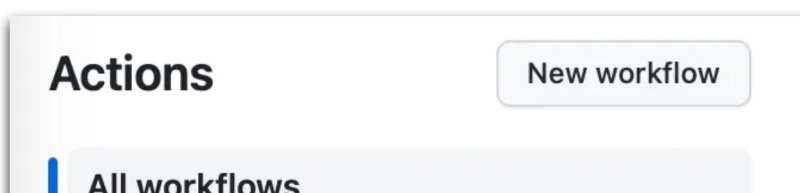
Anzeigen der Aktivität für eine Workflowausführung

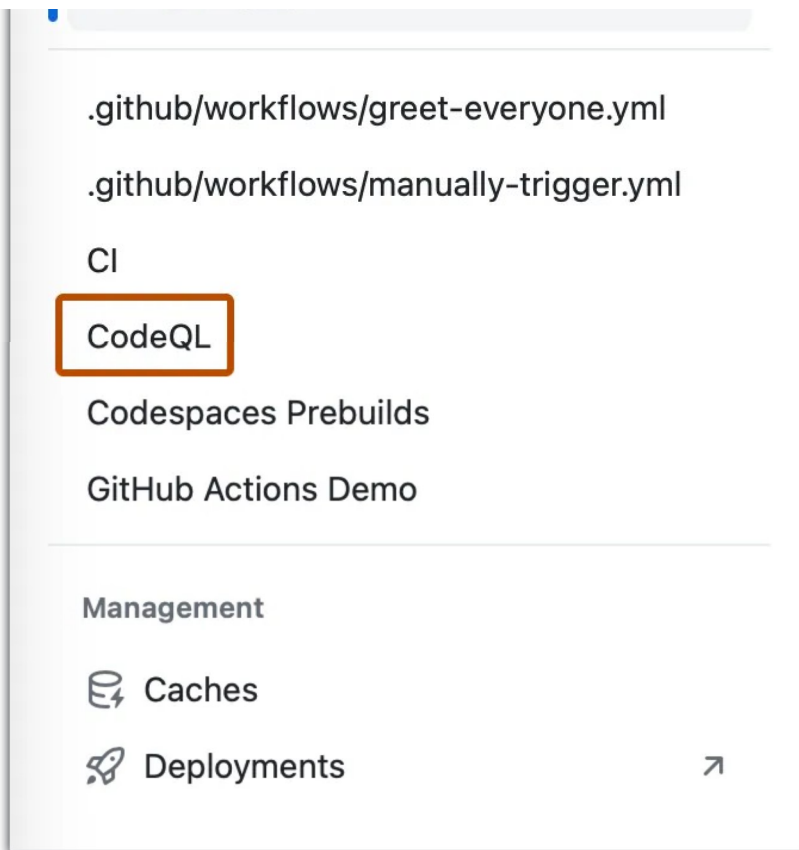
Wenn dein Workflow ausgelöst wird, wird eine *Workflowausführung* erstellt, die den Workflow ausführt. Nachdem eine Workflowausführung gestartet wurde, wird auf GitHub eine grafische Darstellung des Ausführungsfortschritts sowie der Aktivitäten der einzelnen Schritte dargestellt.

- 1 Navigiere auf GitHub.com zur Hauptseite des Repositorys.
- 2 Klicke unter dem Namen deines Repositorys auf **Aktionen**.



- 3 Klicke in der linken Seitenleiste auf den Workflow, den Du sehen willst.





- 4 Klicke in der Liste der Workflowausführungen auf den Namen der Ausführung, um die Zusammenfassung der Workflowausführung anzuzeigen.
- 5 Klicke auf der linken Randleiste oder im Visualisierungsdiagramm auf den Auftrag, den du anzeigen möchtest.
- 6 Klicke auf den Schritt, um seine Ergebnisse anzuzeigen.

Nächste Schritte

GitHub Actions kann dir dabei helfen, nahezu alle Aspekte deines Anwendungsentwicklungsprozesses zu automatisieren. Willst du loslegen? Hier findest du einige hilfreiche Ressourcen für deine nächsten Schritte mit GitHub Actions:

- Eine schnelle Möglichkeit zum Erstellen eines GitHub Actions-Workflows ist unter [„Verwenden von Startworkflows“](#) zu finden.
- Informationen zu CI-Workflows (Continuous Integration) zum Erstellen und Testen deines Codes findest du unter [Automatisieren von Builds und Tests](#).
- Informationen zum Erstellen und Veröffentlichen von Paketen findest du unter [Veröffentlichen von Paketen](#).
- Informationen zum Bereitstellen von Projekten findest du unter [Bereitstellung](#).

- Informationen zum Automatisieren von Aufgaben und Prozessen auf GitHub findest du unter [Verwalten von Issues und Pull Requests](#).
- Beispiele, die komplexere Features von GitHub Actions veranschaulichen, darunter viele der oben genannten Anwendungsfälle, findest du unter [Beispiele](#). Du erfährst anhand detaillierter Beispiele, wie du deinen Code auf einem Runner testest, auf die GitHub-CLI zugreifst und erweiterte Funktionen wie Parallelität und Testmatrizen verwendest.
- Wenn du deine Kenntnisse in der Automatisierung von Arbeitsabläufen und der Beschleunigung der Entwicklung mit GitHub Actions nachweisen möchtest, kannst du ein GitHub Actions Zertifikat mit GitHub Certifications erwerben. Weitere Informationen findest du unter [Informationen zu GitHub Certifications](#).

Rechtliche Hinweise

© 2024 GitHub, Inc. [Impressum](#) [Begriffe](#) [Datenschutz](#) [Status](#) [Preise](#)
[Experten-Dienstleistungen](#) [Blog](#)