

CI/CD Setup Überblick

Warum GitHub Actions?

GitHub Actions hat sich als die ideale Wahl für uns erwiesen, da es nahtlos in unsere bestehende Arbeitsumgebung integriert ist. Die Möglichkeit, Workflows direkt in unserem GitHub-Repository zu definieren, vereinfacht die Einrichtung und Verwaltung der Pipelines erheblich. Außerdem ermöglicht es uns, unsere gesamte CI/CD-Infrastruktur an einem zentralen Ort zu halten, was die Wartung und Überwachung erleichtert.

Unsere CI/CD-Pipeline: Ein Überblick

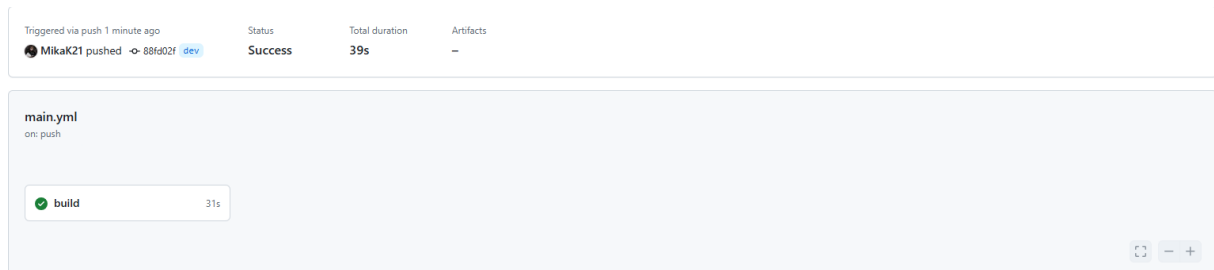
Unsere aktuelle CI/CD-Pipeline ist so konfiguriert, dass bei jedem Push und Pull auf den master und dev Branch das Projekt automatisch gebaut und unsere Unit Tests ausgeführt werden. Dies gewährleistet, dass jede Änderung, die in den Hauptzweig integriert wird, stabil und funktional ist. Die Pipeline besteht aus zwei Hauptkomponenten:

Build-Pipeline: Diese Pipeline übernimmt den Build-Prozess unseres Projekts. Sie prüft, ob alle Abhängigkeiten korrekt aufgelöst und das Projekt erfolgreich kompiliert wird. Hierdurch stellen wir sicher, dass der Code in einem funktionalen Zustand ist, bevor weitere Tests durchgeführt werden.

Test-Pipeline: Parallel zur Build-Pipeline führt diese Pipeline unsere Unit Tests aus. Dies hilft uns sicherzustellen, dass der Code nicht nur erfolgreich gebaut, sondern auch umfassend getestet wird. Durch diese Trennung können wir Builds und Tests unabhängig voneinander überwachen und Probleme gezielt angehen.

Konfiguration der Pipeline: Die Konfiguration unserer CI/CD-Pipeline erfolgt über die main.yml-Datei, die in unserem Repository unter .github/workflows gespeichert ist. In dieser Datei sind alle notwendigen Spezifikationen und Schritte definiert, um den Build- und Testprozess zu steuern. Die Verwendung von YAML als Konfigurationssprache macht es einfach, unsere Workflows zu lesen und zu ändern. Zudem können wir durch die deklarative Natur von YAML klar nachvollziehen, welche Schritte in welcher Reihenfolge ausgeführt werden.

Überblick eines Beispiels + Testresultate



The screenshot displays a GitHub Actions workflow run interface. At the top, a summary bar indicates the workflow was triggered by a push 1 minute ago, is in a 'Success' status, and took 39 seconds to complete. Below this, the workflow file 'main.yml' is shown, triggered on a 'push' event. A single job named 'build' is visible, which has completed successfully in 31 seconds. The interface includes standard GitHub UI elements like a copy icon and zoom controls.

Triggered via	Status	Total duration	Artifacts
push 1 minute ago	Success	39s	—

main.yml
on: push

build
31s

Run tests1s

```
1 ▶ Run npm test
4
5 > next@0.1.0 test /home/runner/work/DHBM_RezeptApp/DHBM_RezeptApp/next
6 > jest
7
8 PASS __test__/previews.test.js
9 PASS __test__/register.test.js
10 PASS __test__/recipe.test.js
11 PASS __test__/login.test.js
12 PASS __test__/api.test.js
13
14 Test Suites: 5 passed, 5 total
15 Tests:      5 passed, 5 total
16 Snapshots: 0 total
17 Time:       0.083 s
18 Ran all test suites.
```