



ONE REPO TO RULE THEM ALL

JAVA UND ANGULAR UNTER EINEM GEMEINSAMEN DACH VEREINT

MOTIVATION

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup
- ⚡ Zwei Branches für ein einzelnes Feature

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup
- ⚡ Zwei Branches für ein einzelnes Feature
 - ⚡ Aufwendig, diese Branches synchron zu halten

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup
- ⚡ Zwei Branches für ein einzelnes Feature
 - ⚡ Aufwendig, diese Branches synchron zu halten
 - ⚡ Beide Projekte müssen gleichzeitig deployt werden

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup
- ⚡ Zwei Branches für ein einzelnes Feature
 - ⚡ Aufwendig, diese Branches synchron zu halten
 - ⚡ Beide Projekte müssen gleichzeitig deployt werden
- ⚡ Code in verschiedenen Projekten zu teilen ist nervig

BACKEND UND FRONTEND IN SEPARATEN REPOSITORIES

- ⚡ Kein einheitliches Projektsetup
- ⚡ Zwei Branches für ein einzelnes Feature
 - ⚡ Aufwendig, diese Branches synchron zu halten
 - ⚡ Beide Projekte müssen gleichzeitig deployt werden
- ⚡ Code in verschiedenen Projekten zu teilen ist nervig
- ⚡ Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

MÖGLICHE LÖSUNG



BE UND FE IM SELBEN REPO

- 💡 Einfacheres Setup/Tooling?
- 💡 BE und FE immer synchron?
- 💡 Code leichter teilen?
- 💡 Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

BE UND FE IM SELBEN REPO

- 💡 Einfacheres Setup/Tooling?
- 💡 BE und FE immer synchron?
- 💡 Code leichter teilen?
- 💡 Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

BE UND FE IM SELBEN REPO

- ✔ Einfacheres Setup/Tooling?
- ✔ BE und FE immer synchron?
- 💡 Code leichter teilen?
- 💡 Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

BE UND FE IM SELBEN REPO

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- 💡 Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

BE UND FE IM SELBEN REPO

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✗ Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach

BE UND FE IM SELBEN REPO

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✗ Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach
- ✗ Pipeline weniger performant?

BE UND FE IN NX-MONOREPO

- ? Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ? Code leichter teilen?
- ? Projektübergreifendes Suchen oder Nachverfolgen ist nicht so einfach
- ? Pipeline weniger performant?

WAS IST NX?

WER BIN ICH?

WER BIN ICH?



Marco Sieben

- Fullstack-Entwickler
- Arbeitet seit 8 Jahren in München
- Erste Programmiererfahrung mit Lego Mindstorms
- Gehegt von vielen Repos
- **Kein Nx-Experte, sondern interessierter Laie**

WER BIN ICH?



Marco Sieben

- Fullstack-Entwickler
- Arbeitet seit 8 Jahren in München
- Erste Programmiererfahrung mit Lego Mindstorms
- Generiert von vielen Repos
- **Kein** Nx-Experte, sondern interessierter Laie

Shoutout an meinen Arbeitgeber andrena objects ❤️

PRAKTISCHES BEISPIEL

UNSER PLAN FÜR DEN REST DES TALKS

1. Minimales Backend und Frontend anlegen

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren
3. API robuster machen/Code teilen

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren
3. API robuster machen/Code teilen
4. Lokales Caching einbauen

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren
3. API robuster machen/Code teilen
4. Lokales Caching einbauen
5. CI/CD-Setup

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren
3. API robuster machen/Code teilen
4. Lokales Caching einbauen
5. CI/CD-Setup
6. Verteilten Cache hinzufügen

1. Minimales Backend und Frontend anlegen
2. In Nx-Monorepo migrieren
3. API robuster machen/Code teilen
4. Lokales Caching einbauen
5. CI/CD-Setup
6. Verteilten Cache hinzufügen
7. Was haben wir erreicht?

1. MINIMALES BE UND FE

DER EINFACHE TEIL

FANCY AF



2. IN NX-MONOREPO MIGRIEREN

JETZT KOMMT NX

Automatische Migration eines Angular-Projekts:

```
npx nx@latest init --integrated
```

Migration des Backends:

Migration des Backends: Leider nicht ganz so einfach 😞

Migration des Backends: Leider nicht ganz so einfach 😞

1. Java-Unterstützung hinzufügen:

```
npm install @nxrocks/nx-spring-boot --save-dev
```

Migration des Backends: Leider nicht ganz so einfach 😞

1. Java-Unterstützung hinzufügen:

```
npm install @nxrocks/nx-spring-boot --save-dev
```

2. Neue Java-App im Monorepo anlegen:

```
nx g @nxrocks/nx-spring-boot:project apps/backend
```

Migration des Backends: Leider nicht ganz so einfach 😞

1. Java-Unterstützung hinzufügen:

```
npm install @nxrocks/nx-spring-boot --save-dev
```

2. Neue Java-App im Monorepo anlegen:

```
nx g @nxrocks/nx-spring-boot:project apps/backend
```

3. Quellcode von Backend-Projekt in neu erstellte App kopieren 🤝

ERGEBNIS

ERGEBNIS

Monorepo, Yey!

ERGEBNIS

Monorepo, Yey!



3. API ROBUSTER MACHEN/CODE TEILEN

JETZT WIRD ES SPANNEND

PROBLEM

API und Typen an zwei Stellen konfiguriert (Backend und Frontend)

PROBLEM

API und Typen an zwei Stellen konfiguriert (Backend und Frontend)

LÖSUNG

API nur im BE definieren und im FE generieren

OPENAPI-SPEC IM BACKEND ERZEUGEN

Durch Plugin

```
id("org.springdoc.openapi-gradle-plugin") version "1.9.0"
```

und Dependency

```
implementation("org.springdoc:springdoc-openapi-starter-webmvc-ui:2.6.0")
```

=>

```
./gradlew generateOpenApiDocs
```

Eigenes Nx-Target im BE dafür definieren

Eigenes Nx-Target im BE dafür definieren

=>

```
nx generateApiSpec backend
```

FRONTEND-CODE DURCH SPEC ERZEUGEN

```
npm install @openapitools/openapi-generator-cli --save-dev
```

=>

```
npx openapi-generator-cli generate -i apps/backend/build/openapi.json -g typescript
```

FRONTEND-CODE DURCH SPEC ERZEUGEN

```
npm install @openapitools/openapi-generator-cli --save-dev
```

=>

```
npx openapi-generator-cli generate -i apps/backend/build/openapi.json -g typescript
```

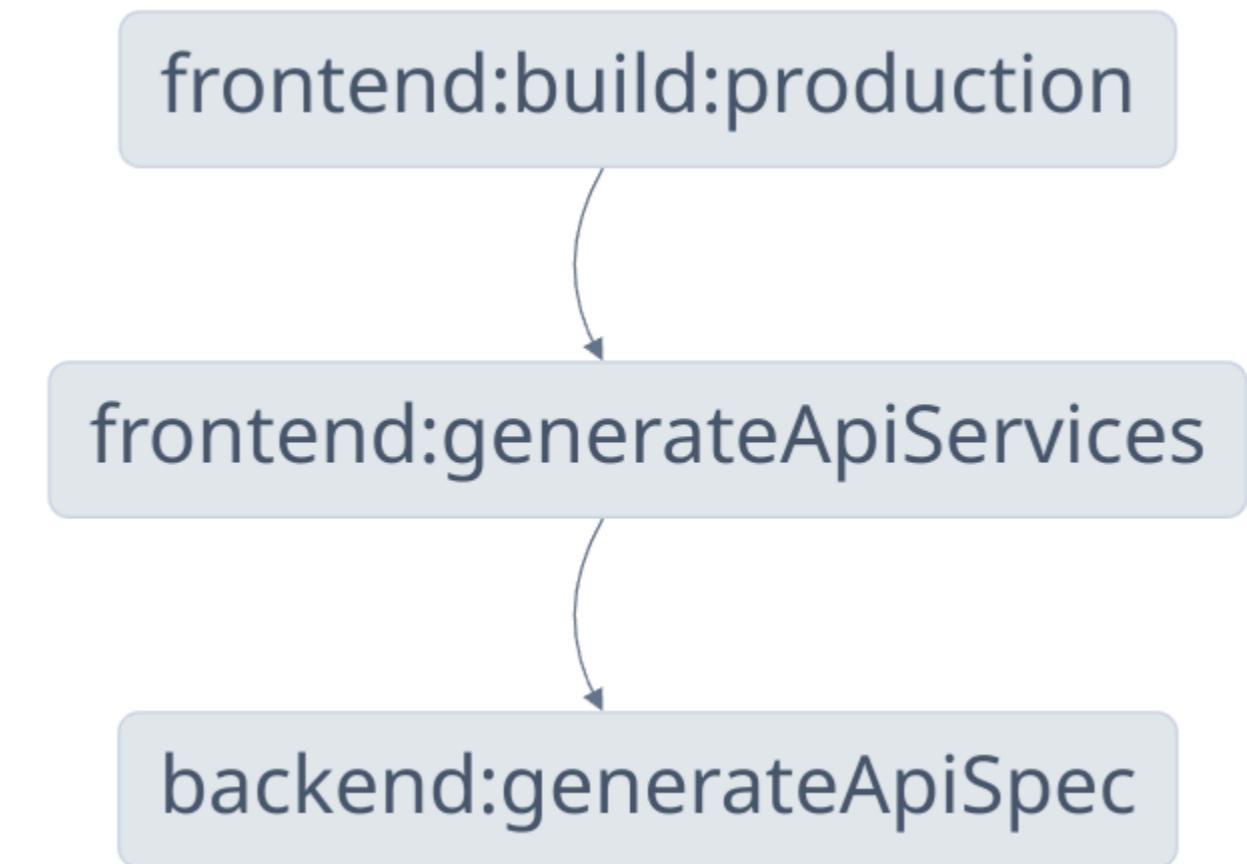
Eigens Target im Frontend =>

```
nx generateApiServices frontend
```

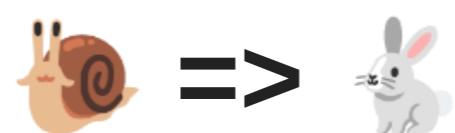
ABHÄNGIGKEITEN DER TASKS

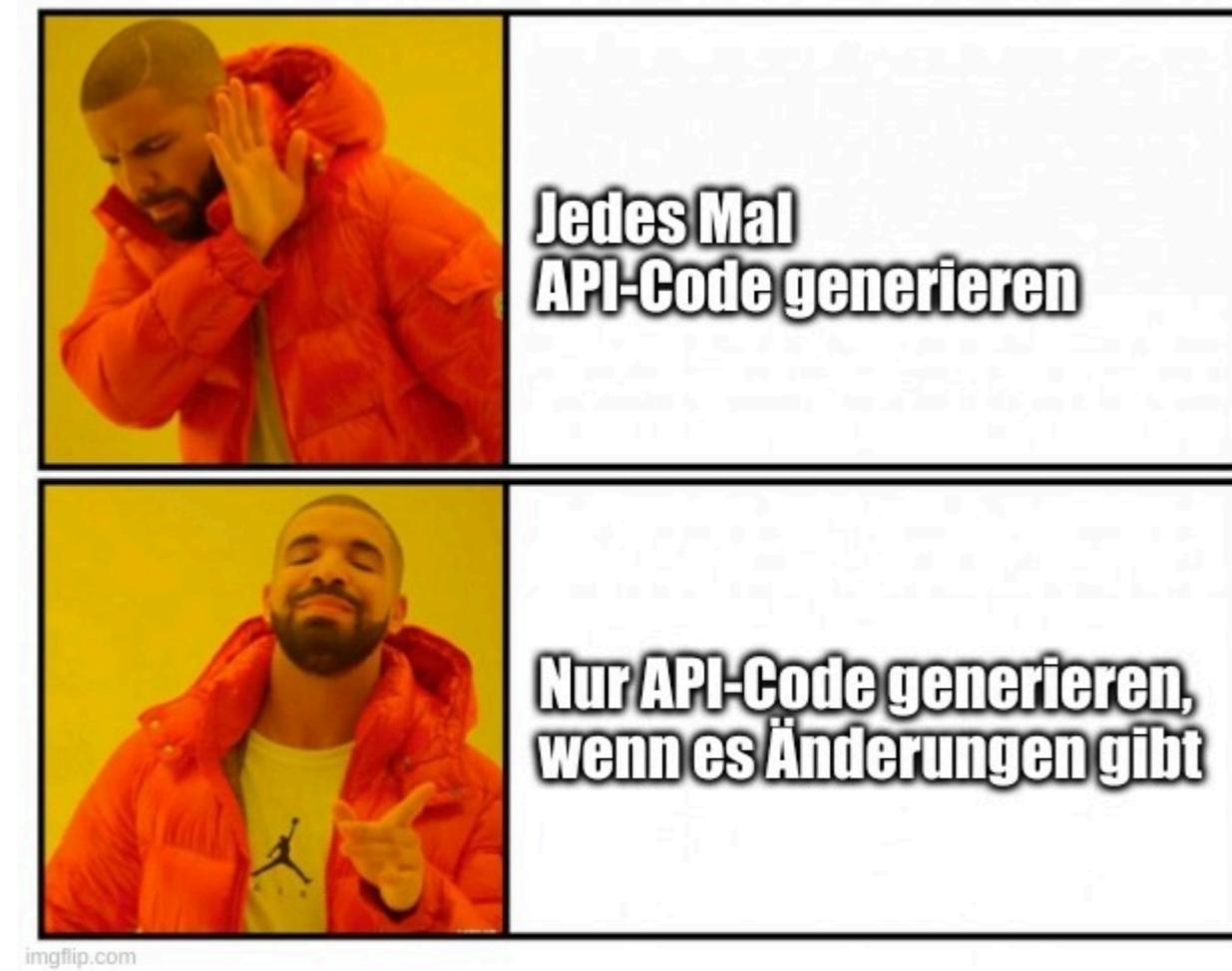
ABHÄNGIGKEITEN DER TASKS

```
nx graph
```



4. CACHING





5. CI/CD-SETUP

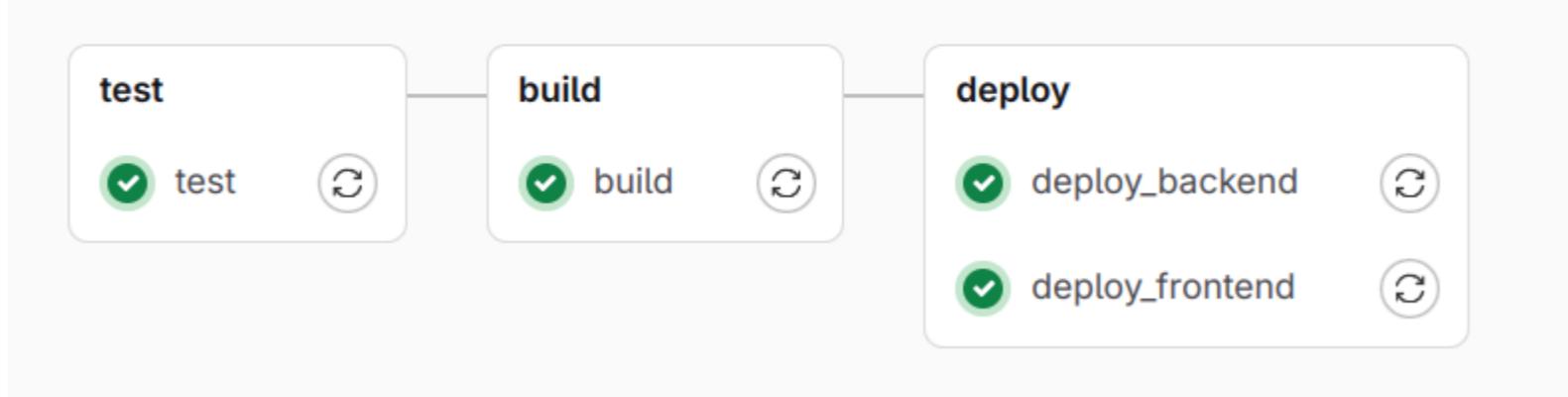
WIR SIND SCHLIESSLICH PROFIS!

ERSTER EINFACHER VERSUCH

ERSTER EINFACHER VERSUCH

latest ⚡ 4 jobs ⏲ 6 minutes 26 seconds, queued for 0 seconds

Pipeline Jobs 4 Tests 0

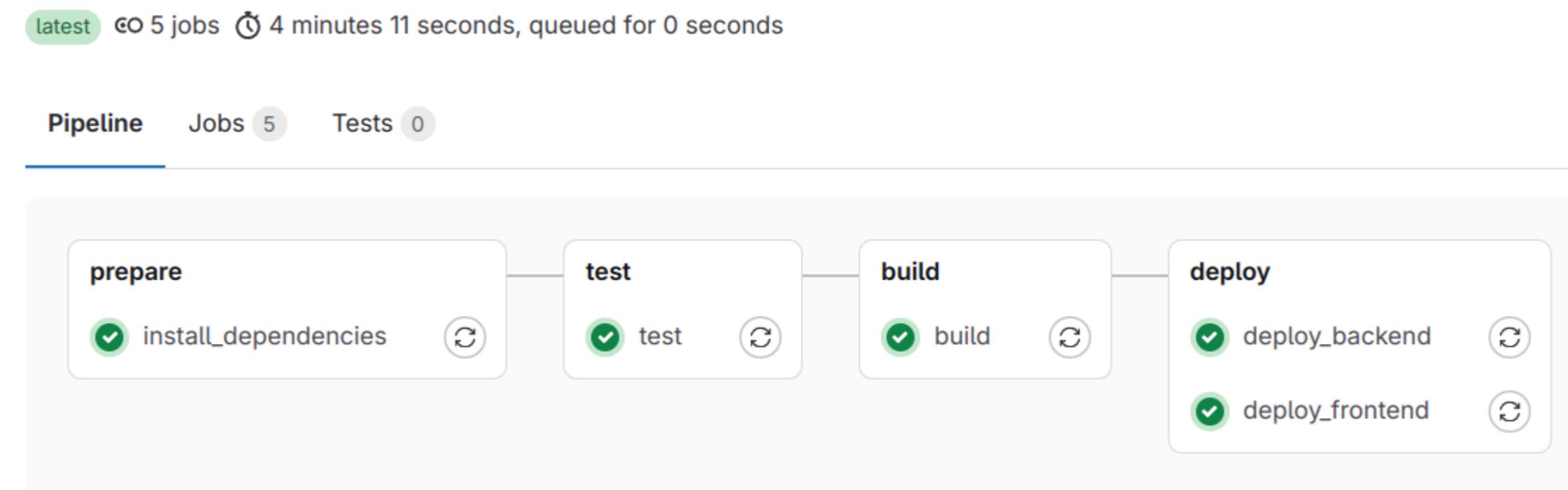


GITLAB-OPTIMIERUNG

- Dependencies zwischen Schritten Cachen
- Gitlab Cache optimieren

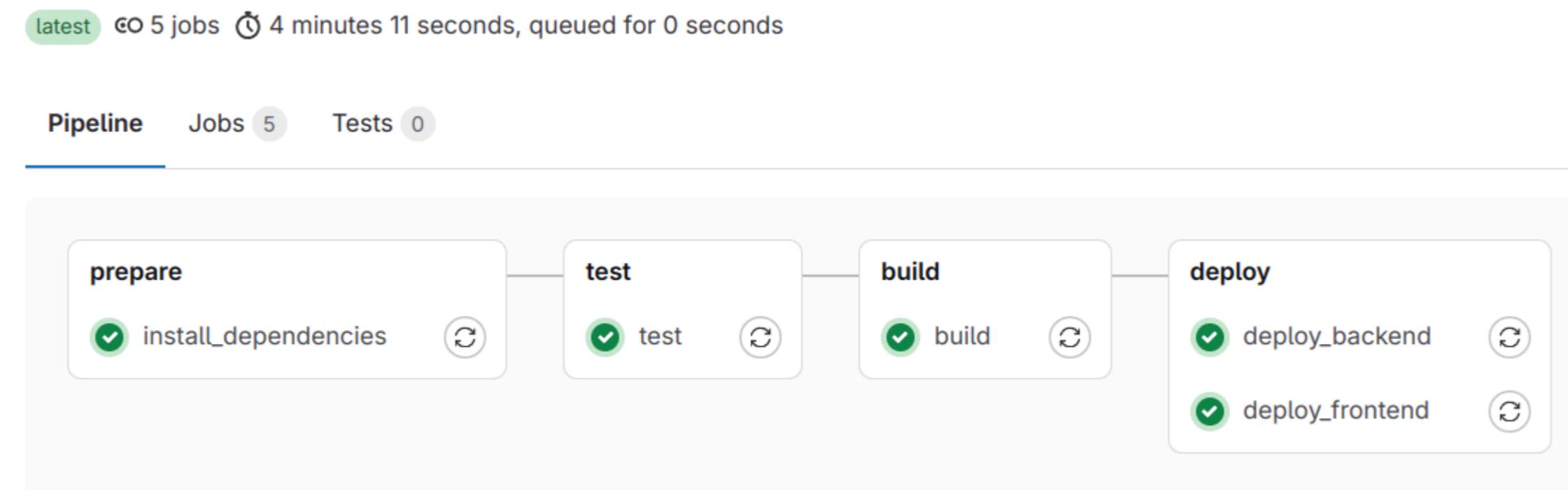
GITLAB-OPTIMIERUNG

- Dependencies zwischen Schritten Cachen
- Gitlab Cache optimieren



GITLAB-OPTIMIERUNG

- Dependencies zwischen Schritten Cachen
- Gitlab Cache optimieren



6.5 Minuten => 4.5 Minuten

6. VERTEILTER CACHE

AB IN DIE CLOUD

- ⚡ Pipeline hat keinen lokalen Nx Cache

- ⚡ Pipeline hat keinen lokalen Nx Cache
- 💡 Verwende verteilten Cache (hier: in S3)

⚡ Pipeline hat keinen lokalen Nx Cache

💡 Verwende verteilten Cache (hier: in S3)

Auch möglich:

- Nx Cloud (der Standard)
- Azure
- Google Cloud
- Redis
- und viele mehr
- sowie eine Bibliothek, um einfach eigene Lösungen zu implementieren

⚡ Pipeline hat keinen lokalen Nx Cache

💡 Verwende verteilten Cache (hier: in S3)

Auch möglich:

- Nx Cloud (der Standard)
- Azure
- Google Cloud
- Redis
- und viele mehr
- sowie eine Bibliothek, um einfach eigene Lösungen zu implementieren

! Ab Nx 21 nur noch über Powerpack !

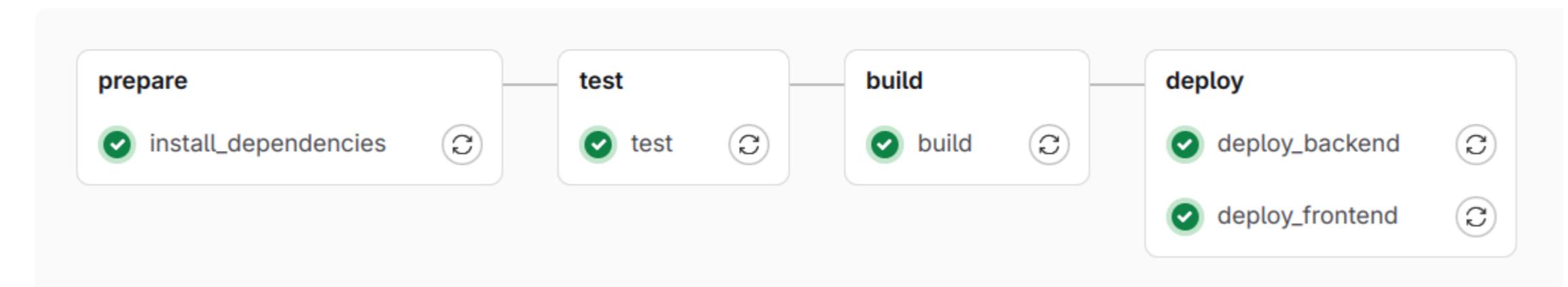
- S3-Bucket in AWS
- IAM-User in AWS
- Remote-Cache-Plugin

```
npm install --save-dev @pellegrims/nx-remotecac
```

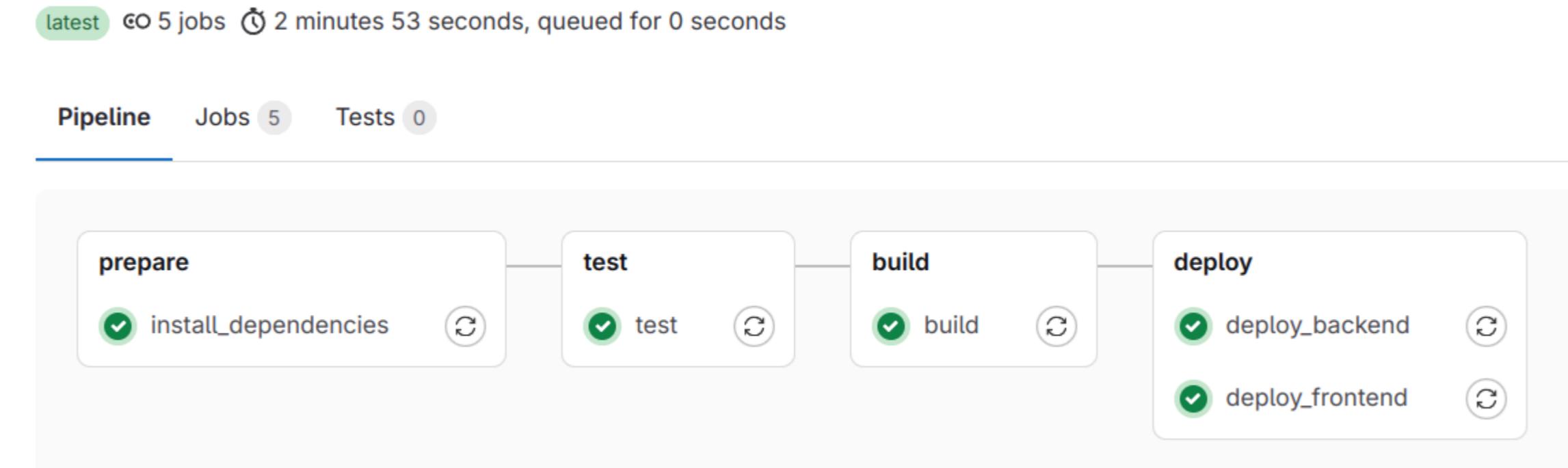
- Cache konfigurieren

latest ⚡ 5 jobs ⏱ 2 minutes 53 seconds, queued for 0 seconds

Pipeline Jobs 5 Tests 0



(nur Frontend-Änderung)

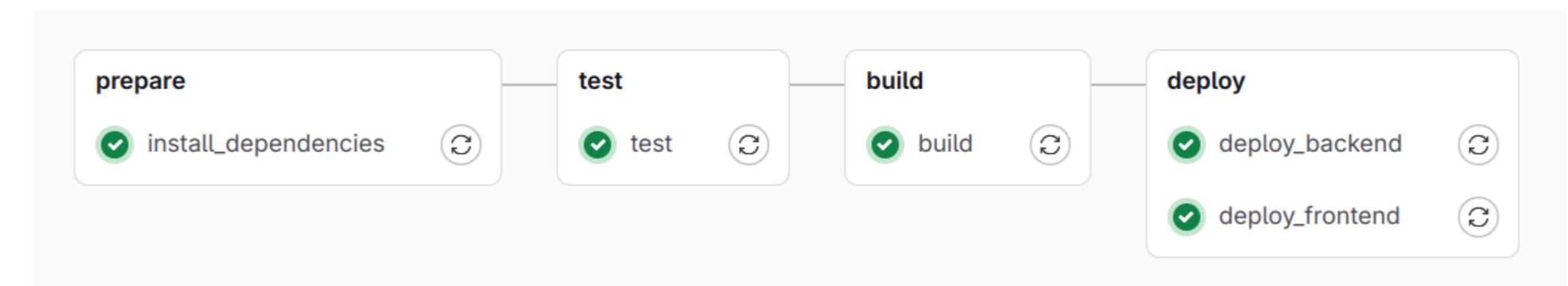


(nur Frontend-Änderung)

4.5 Minuten => 3 Minuten

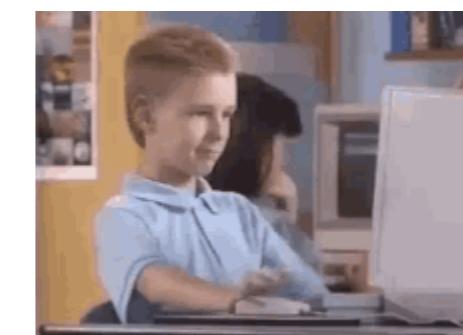
latest ⚡ 5 jobs ⏱ 2 minutes 53 seconds, queued for 0 seconds

Pipeline Jobs 5 Tests 0



(nur Frontend-Änderung)

4.5 Minuten => 3 Minuten



7. ERGEBNIS

WAS HABEN WIR ERREICHT?

- ? Einfacheres Setup/Tooling?
- ? BE und FE immer synchron?
- ? Code leichter teilen?
- ? Projektübergreifendes Suchen oder Nachverfolgen
- ? Pipeline weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✗ BE und FE immer synchron?
- ✗ Code leichter teilen?
- ✗ Projektübergreifendes Suchen oder Nachverfolgen
- ✗ Pipeline weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ? Code leichter teilen?
- ? Projektübergreifendes Suchen oder Nachverfolgen
- ? Pipeline weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ? Projektübergreifendes Suchen oder Nachverfolgen
- ? Pipeline weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ? Pipeline weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)
- ✓ Globale Dependencyverwaltung

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)
- ✓ Globale Dependencyverwaltung
- ✓ Neue Projekte leicht hinzuzufügen

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)
- ✓ Globale Dependencyverwaltung
- ✓ Neue Projekte leicht hinzuzufügen
- ⚡ Initiales Setup etwas aufwendiger

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)
- ✓ Globale Dependencyverwaltung
- ✓ Neue Projekte leicht hinzuzufügen
- ⚡ Initiales Setup etwas aufwendiger
- ⚡ Weiter Abstraktionsebene

- ✓ Einfacheres Setup/Tooling?
- ✓ BE und FE immer synchron?
- ✓ Code leichter teilen?
- ✓ Projektübergreifendes Suchen oder Nachverfolgen
- ✓ Pipeline **nicht** weniger performant?
- ✓ Caching (lokal und verteilt)
- ✓ Globale Dependencyverwaltung
- ✓ Neue Projekte leicht hinzuzufügen
- ⚡ Initiales Setup etwas aufwendiger
- ⚡ Weiter Abstraktionsebene
- ⚡ Von der Entwicklung des Nx-Projekts abhängig (Beispiel "verteilter Cache")

FAZIT

FAZIT NICE, ODER?

FAZIT NICE, ODER?

(Mit einem Wermutstropfen)

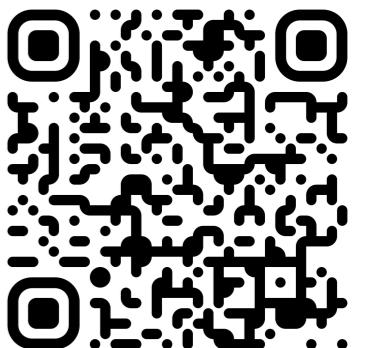
SIEHE AUCH:

- "Java und Angular unter einem Dach mit Nx"
Marco Sieben, Java Magazin (entwickler.de)
- "Fantastische Monorepos und wo sie zu finden sind"
Simon Hauck und Thomas Czogalik, XP Days
- Die Nx-Seite
<https://nx.dev>
- "Monorepos & Spaceships - Navigating successfully through Code and Cosmos"
Max Kless, MunichJS November Meetup, 14.11.2024

Marco Sieben
andrena objects ag

 marco.sieben@andrena.de

 <https://www.linkedin.com/in/stnimmerlein>



VIELEN DANK

Marco Sieben
andrena objects ag

 marco.sieben@andrena.de

 <https://www.linkedin.com/in/stnimmerlein>

