

Task-API

Präsentation Development Operations

Jenny Wagner



Inhaltsverzeichnis



- Projektidee
- Techstack
- Dockerfile
- CI Pipeline
- Lessons Learned
- Nächste Schritte

Projektidee

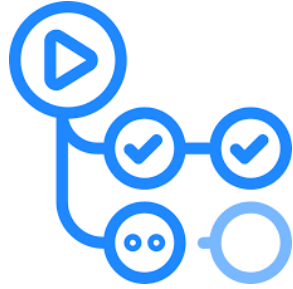
- REST-API zum Verwalten von Tasks
- Speichert in Sqlite Datenbank
- Deploybar als einzelner Docker Container

```
GET http://localhost:3000/api/tasks

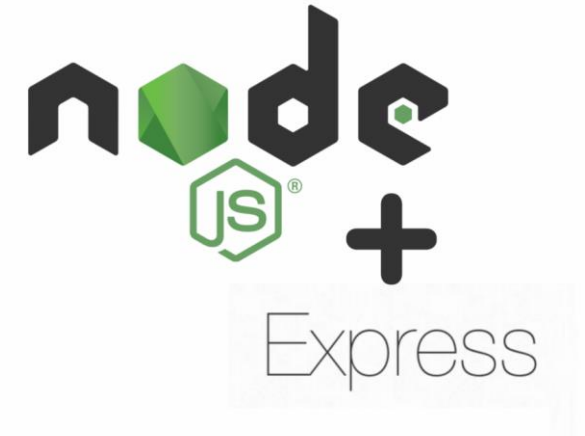
1  [
2    {
3      "id": 1,
4      "title": "Wäsche machen",
5      "description": "Heute noch",
6      "completed": 0
7    },
8    {
9      "id": 2,
10     "title": "Essen kochen",
11     "description": null,
12     "completed": 0
13   }
14 ]
```

Techstack

- Typescript
- NodeJS
- ExpressJS
- Prettier + ESLint
- Sqlite
- Docker
- Vitest
- GitHub Actions



Prettier



Dockerfile

- Node Base Image
- Installieren von Abhängigkeiten
- Source Code kopieren
- Vorkonfigurierte DB Kopieren
- Port 3000 öffnen
- Applikation starten

```
1  FROM node:20
2  WORKDIR /app
3
4  COPY package*.json ./
5
6  RUN npm install
7
8  COPY src/ ./src/
9  COPY tsconfig.json ./
10 COPY template.data.sqlite ./data.sqlite
11
12 EXPOSE 3000
13
14 CMD ["npm", "run", "dev"]
15
```

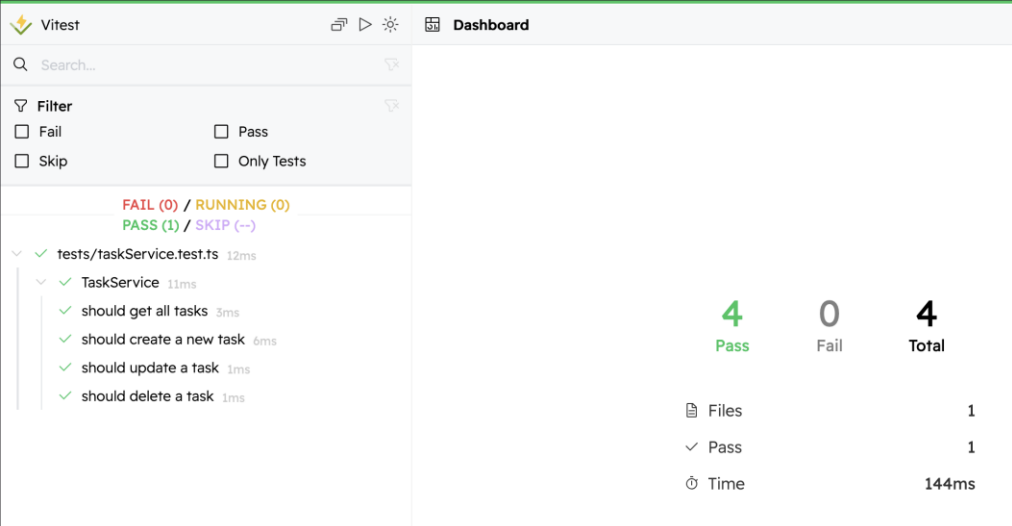
CI Pipeline

- 2 Jobs
- Trigger Main Branch & Pull Request
- „Build and test“
 - Baut mit npm
 - Linting + Prettier
 - Tests + Coverage
- „Docker Build“
 - Baut Docker Image
 - Pushed Docker Image

```
1 docker-build:
2   runs-on: ubuntu-latest
3
4   steps:
5     - name: Checkout code
6       uses: actions/checkout@v4
7
8     - name: Login to DockerHub
9       uses: docker/login-action@v3
10      with:
11        username: ${secrets.DOCKER_USERNAME}
12        password: ${secrets.DOCKER_PASSWORD}
13
14     - name: Build Docker Image
15       run: docker build -t ${secrets.DOCKER_USERNAME }/task-api:latest .
16
17     - name: Push Docker Image
18       run: docker push ${secrets.DOCKER_USERNAME }/task-api:latest
```

```
1 build-and-test:
2   runs-on: ubuntu-latest
3
4   steps:
5     - name: Checkout code
6       uses: actions/checkout@v4
7
8     - name: Setup Node.js
9       uses: actions/setup-node@v4
10      with:
11        node-version: 20
12        cache: 'npm'
13
14     - name: Install dependencies
15       run: npm ci
16
17     - name: Run Linter and Prettier Check
18       run: npm run check
19
20     - name: Run Unit Tests
21       run: npm run test
```

Aktueller Stand



Vitest Dashboard

Search...

Filter

- ☐ Fail
- ☐ Pass
- ☐ Skip
- ☐ Only Tests

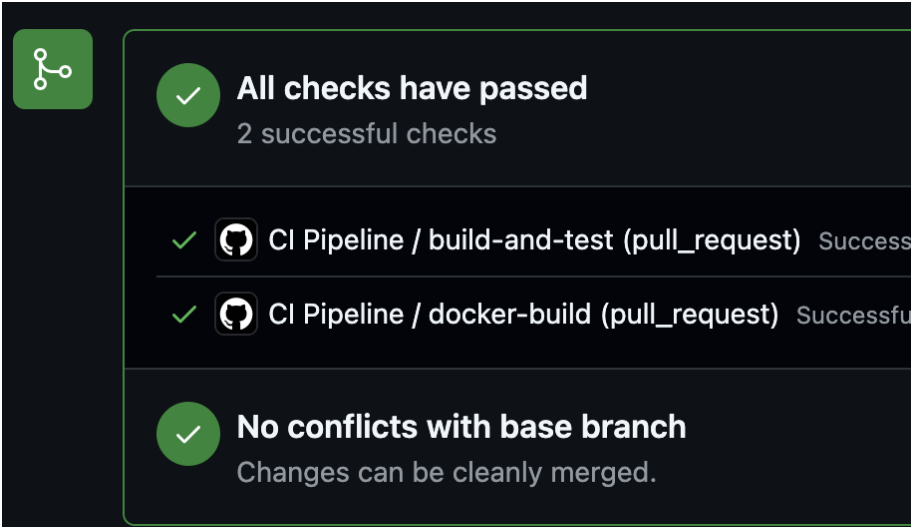
FAIL (0) / RUNNING (0)
PASS (1) / SKIP (--)

tests/taskService.test.ts 12ms

- TaskService 11ms
 - should get all tasks 3ms
 - should create a new task 6ms
 - should update a task 1ms
 - should delete a task 1ms

4 Pass 0 Fail 4 Total

Files 1
Pass 1
Time 144ms



Share icon

- ☒ All checks have passed
2 successful checks
- ☒ CI Pipeline / build-and-test (pull_request) Success
- ☒ CI Pipeline / docker-build (pull_request) Success
- ☒ No conflicts with base branch
Changes can be cleanly merged.

Lessons Learned

- **Früh mit DevOps anfangen**

→ CI/CD & Git-Struktur direkt zu Beginn einrichten.

- **Keep it simple**

→ Nur Tools nutzen, die man wirklich versteht & braucht und unnötigen Code vermeiden.

- **Versionskontrolle ernst nehmen**

→ Regelmäßig committen, Branches sauber halten, Pull Requests nutzen.

- **Deploy regelmäßig, nicht am Ende**

→ Kleine, häufige Deployments halten alles stabil & testbar. Momentan nur lokal.

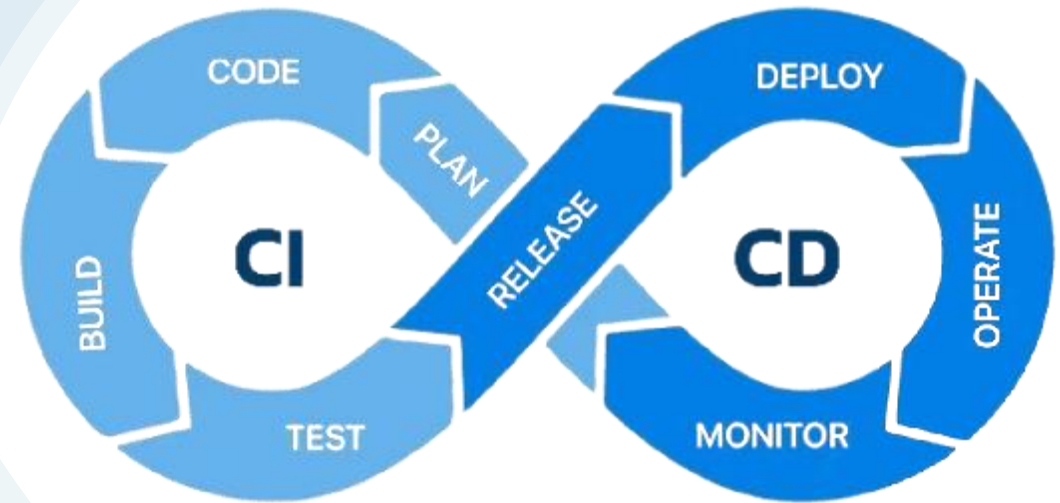
Nächste Schritte

CI Pipeline

- Test Coverage
- SonarQube

CD Pipeline

- 1 Job auf Main Branch
- Pullt latest Docker Image
- Aktualisiert Docker Container
- Test Curl Request



Vielen Dank für eure
Aufmerksamkeit

Gibt es noch Fragen?

