

IT Enrichment

Concept CI/CD



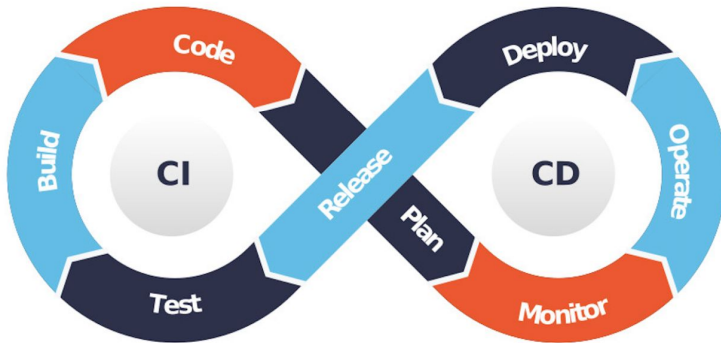


Continuous integration and continuous delivery

Concept CI/CD

CI/CD adalah singkatan dari Continuous Integration (CI) dan Continuous Delivery (CD). Kedua konsep ini merujuk pada praktik-praktik dalam pengembangan perangkat lunak yang bertujuan untuk meningkatkan kecepatan, kualitas, dan keandalan pengiriman perangkat lunak.

CI/CD sering kali digunakan bersama-sama sebagai bagian dari praktik pengembangan perangkat lunak yang dikenal sebagai "CI/CD pipeline," yang mencakup langkah-langkah dari penggabungan kode hingga pengiriman ke produksi secara otomatis.



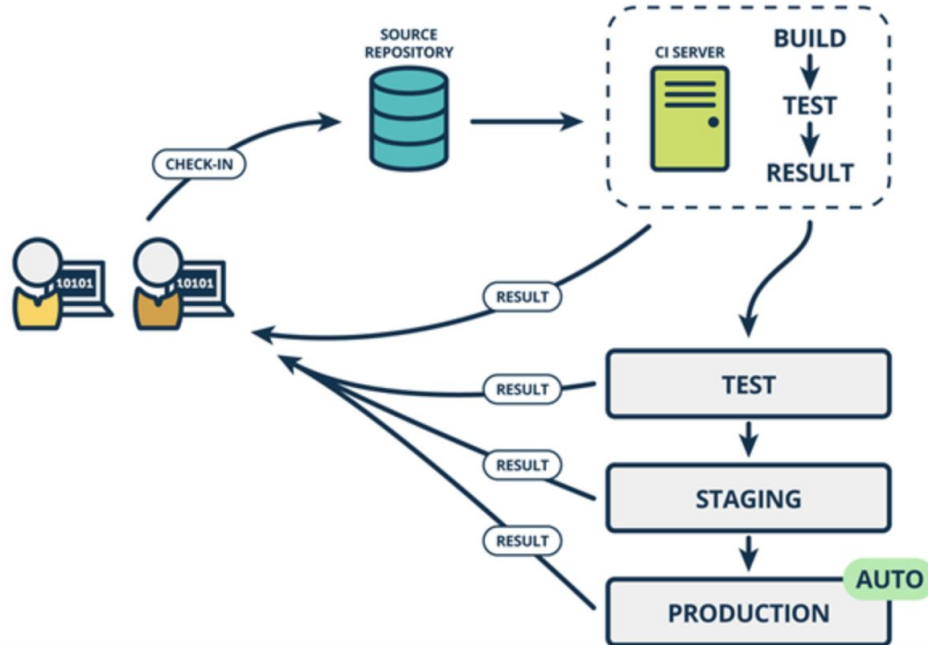
Continuous Integration (CI):

- CI adalah praktik di mana para pengembang secara teratur menggabungkan kode mereka ke dalam repository bersama, seperti Git.
- Setiap kali ada perubahan kode, sistem CI akan otomatis membangun (compile) dan menjalankan rangkaian tes otomatis untuk memastikan bahwa perubahan tersebut tidak merusak fungsionalitas yang sudah ada.
- Tujuan CI adalah untuk mendeteksi dan memperbaiki konflik atau kesalahan lebih awal dalam siklus pengembangan, sehingga tim dapat mengatasi masalah segera setelah mereka muncul.

Continuous Delivery (CD):

- CD melibatkan otomatisasi seluruh proses pengiriman perangkat lunak, mulai dari pembangunan, pengujian, hingga penyebaran ke lingkungan produksi.
- Setelah perubahan lolos dari proses CI, CD memastikan bahwa perangkat lunak tersebut dapat dikirim ke lingkungan produksi kapan saja dengan cepat dan dengan risiko sekecil mungkin.
- CD menghilangkan manual dan proses yang rentan terhadap kesalahan, memastikan konsistensi antara lingkungan pengembangan, pengujian, dan produksi.

Concept CI/CD



CI/CD *pipeline* ini sangat lazim digunakan dalam pengembangan perangkat lunak. CI/CD *pipeline* ini menjadi penghubung antara tim pengembang dengan tim operasional yang di dalamnya terdapat tiga fase yang berupa *continuous integration*, *continuous delivery*, dan *continuous deployment*.

Ketiga fase tersebut akan dilakukan secara terus menerus dan otomatis untuk mendapatkan perangkat lunak yang andal dan bebas dari *bug*.



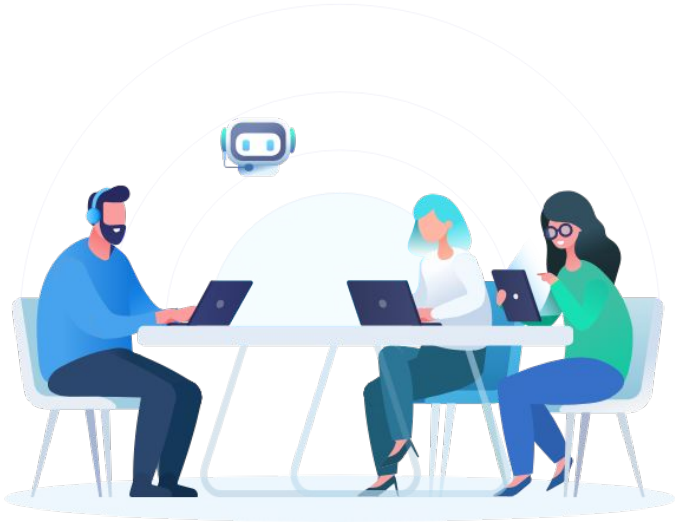
CI/CD Tools

+

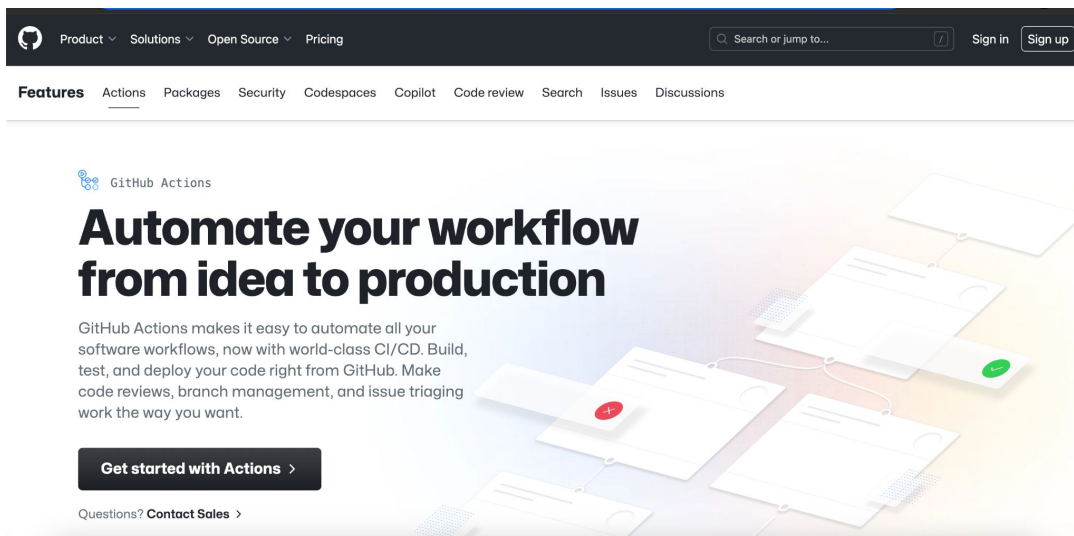
CI/CD Tools

Ada beberapa *tools* yang dapat digunakan dalam proses CI/CD. Berikut ini adalah *tools* yang dapat kamu gunakan:

1. **Jenkins** : <https://www.jenkins.io/>
2. **AWS Codebuild** :
<https://aws.amazon.com/codebuild/>
3. **Azure devops** :
<https://azure.microsoft.com/en-us/products/devops>
4. **Gitlab CI/CD** : <https://docs.gitlab.com/ee/ci/>
5. **Github Action** : <https://github.com/features/actions>



GitHub Actions adalah platform otomatisasi yang disediakan oleh GitHub untuk mendukung alur kerja pengembangan perangkat lunak (CI/CD) dan otomatisasi tugas-tugas lainnya dalam siklus pengembangan perangkat lunak. Dengan GitHub Actions, Kamu dapat membuat serangkaian tindakan (actions) yang dijalankan secara otomatis berdasarkan peristiwa tertentu dalam repositori GitHub.



The screenshot shows the GitHub Actions landing page. At the top is a dark navigation bar with the GitHub logo, links for Product, Solutions, Open Source, and Pricing, a search bar, and Sign in / Sign up buttons. Below this is a secondary navigation bar with links for Features, Actions, Packages, Security, Codespaces, Copilot, Code review, Search, Issues, and Discussions. The main content area features the GitHub Actions logo and the headline "Automate your workflow from idea to production". A paragraph of text describes the capabilities of GitHub Actions. A prominent "Get started with Actions" button is located below the text. At the bottom left, there is a link for "Questions? Contact Sales". The background of the main content area features a stylized illustration of workflow cards connected by lines.

Product Solutions Open Source Pricing

Search or jump to... Sign in Sign up

Features Actions Packages Security Codespaces Copilot Code review Search Issues Discussions

GitHub Actions

Automate your workflow from idea to production

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

[Get started with Actions >](#)

Questions? [Contact Sales >](#)

Kita akan mempelajari deployment CI/CD dengan contoh case menggunakan Javascript dan NodeJS sederhana, dimana dalam prosesnya kita asumsikan developer sudah menyiapkan server lewat docker ataupun ada server lain yang akan digunakan untuk deployment CI/CD. Untuk menyiapkan server sederhana jika tidak memiliki shared hosting atau cloud server kita akan coba buat server dan repository dummy di **Docker Hub**.

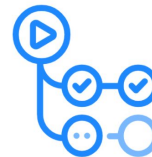
Dalam case ini stack yang terlibat akan seperti ini :



Atau stack language lainnya
seperti golang, laravel, dll



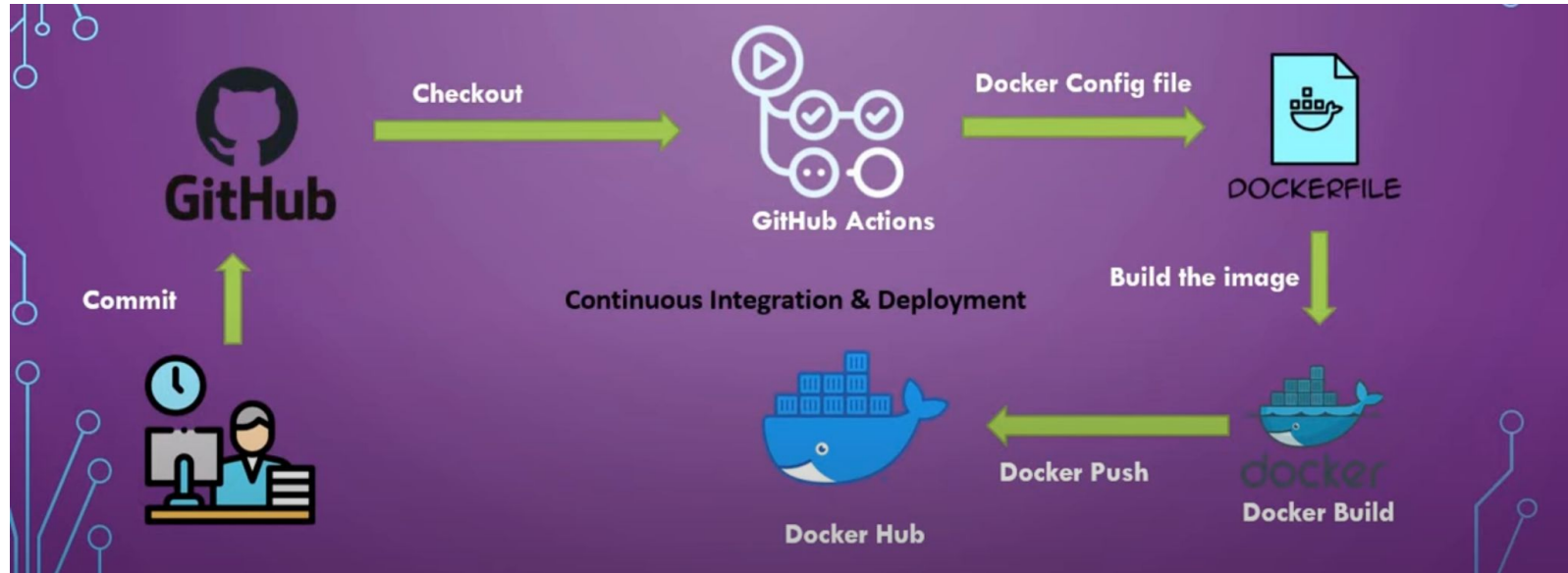
Atau server lainnya

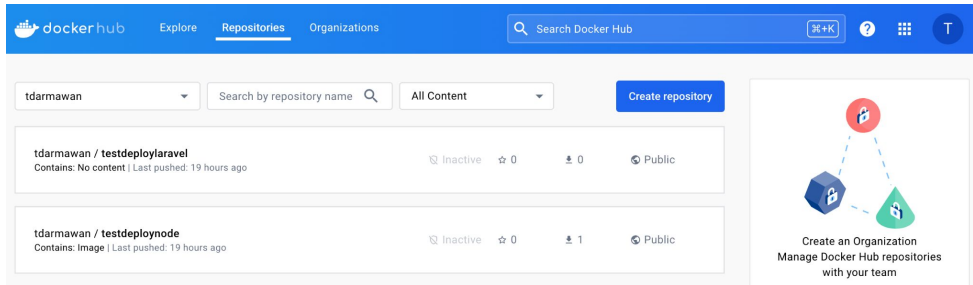


GitHub Actions

CI/CD with Github Action

Perhatikan proses workflow CI/CD yang akan kita buat dibawah ini :





[Repositories](#) / [Create](#)

Create repository

Namespace:

Repository Name *

Repository name is required



Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ Public

Appears in Docker Hub search results

☐ Private

Only visible to you

Cancel

Create

Kita perlu membuat repository dummy di Docker hub agar proses CI/CD Github dari aplikasi yang kita deploy dapat berjalan. Cara membuatnya seperti berikut :

1. Kunjungi <https://hub.docker.com/>
2. Lakukan Login jika belum punya bisa melakukan registrasi terlebih dahulu
3. Kunjungi menu repository
4. Buat repository baru dengan nama yang mudah dan cocok untuk aplikasi yang akan dibuat CI/CD

Berikutnya mari kita coba buat workflows untuk menjalankan CI/CD di github repository kita sehingga ketika ada git push baru akan melakukan proses deployment CI/CD.

1. Berikut adalah source dummy yang dapat digunakan :
https://drive.google.com/drive/folders/12MbUe0Rpi5X_XHGSYMPb1D4vVD5S8Gr-?usp=sharing
2. Pada projek aplikasi kita buat folder github → workflows
3. Buat file main.yml di dalam folder workflows
4. Lalu isikan file yml tersebut seperti contoh di slide berikut
5. ubah script `tdarmawan/testdeploynode` menyesuaikan dengan nama repository pada docker hub yang sudah dibuat

CI/CD with Github Action



```
name: Publish Docker image
```

```
on:
```

```
  push:
```

```
    branches: ['main']
```

```
jobs:
```

```
  push_to_registry:
```

```
    name: Push Docker image to Docker Hub
```

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

```
    steps:
```

```
      - name: Check out the repo
        uses: actions/checkout@v3
```

```
      - name: Log in to Docker Hub
        uses:
```

```
        docker/login-action@f054a8b539a109f9f41c372932f1ae047
        eff08c9
```

```
        with:
```

```
          username: ${ secrets.DOCKER_USERNAME }}
```

```
          password: ${ secrets.DOCKER_PASSWORD }}
```

```
      - name: Extract metadata (tags, labels) for
        Docker
```

```
        id: meta
```

```
        uses:
```

```
        docker/metadata-action@98669ae865ea3cfffcbcaa878cf5
        7c20bbf1c6c38
```

```
        with:
```

```
          images: tdarmawan/testdeploynode
```

```
      - name: Build and push Docker image
```

```
        uses:
```

```
        docker/build-push-action@ad44023a93711e3deb3375089
        80b4b5e9bcd5dc
```

```
        with:
```

```
          context: .
```

```
          push: true
```

```
          tags: ${ steps.meta.outputs.tags }
```

```
          labels: ${ steps.meta.outputs.labels }
```

Sebelumnya, kita telah membuat script workflows dari berkas yml. Selanjutnya, langkah kita adalah membuat repository baru di GitHub dan mengatur Secrets Key agar workflows dapat dijalankan dari GitHub menuju repositori Docker Hub.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *



tdarmawan

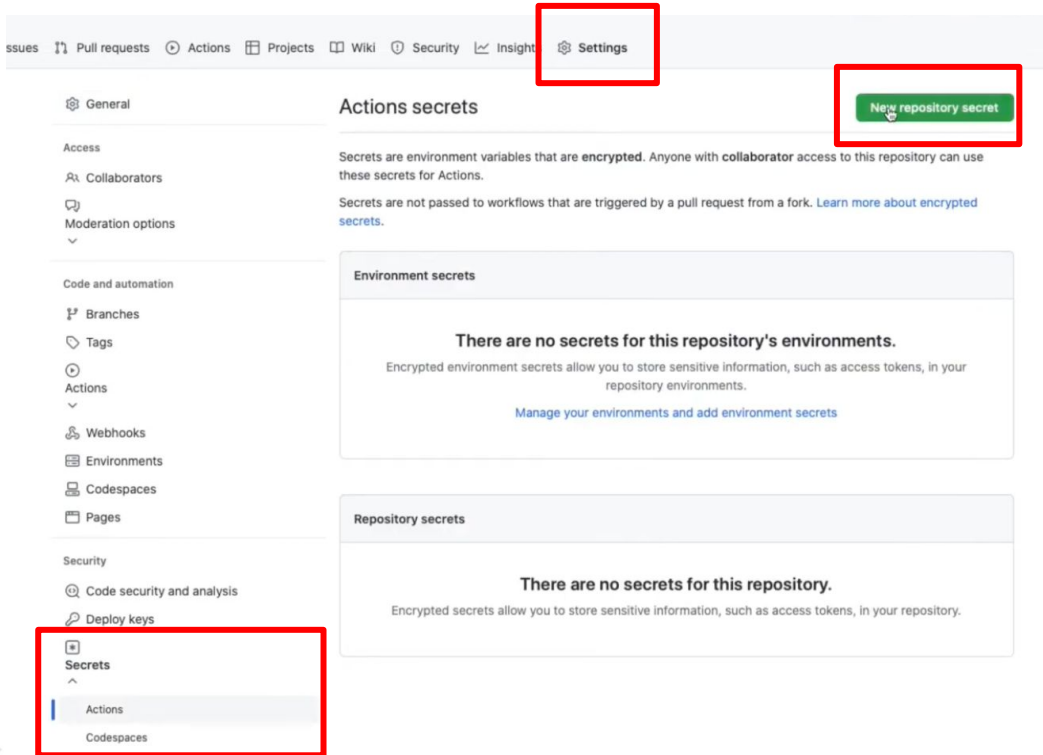


Repository name *

injs-cicd



CI/CD with Github Action



Berikutnya kita akan memasukan secrets key agar github kita dapat terhubung pada repository di docker hub.

- buka repository menu settings
→ secrets → menu
- tambahkan secret baru
DOCKER_USERNAME dan
DOCKER_PASSWORD value
nya isikan dari akun docker
hub username dan password.

CI/CD with Github Action

Actions secrets / New secret

Name *

YOUR_SECRET_NAME

Secret *

Add secret

Projects Wiki Security Insights Settings

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables








Environment secrets

This repository has no environment secrets.

Manage environment secrets

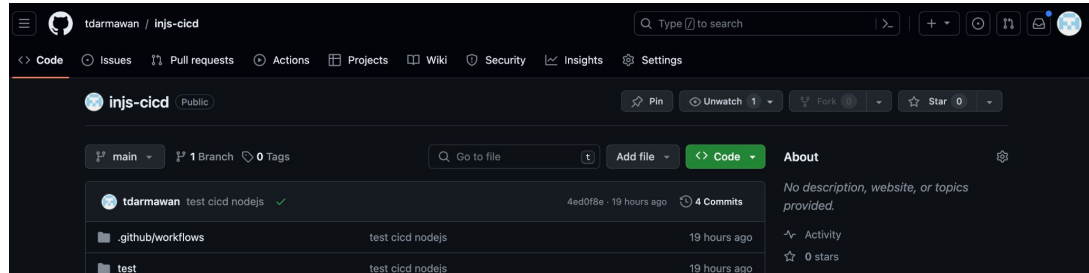
Repository secrets

New repository secret

Name 	Last updated
 DOCKER_PASSWORD	19 hours ago  
 DOCKER_USERNAME	19 hours ago  

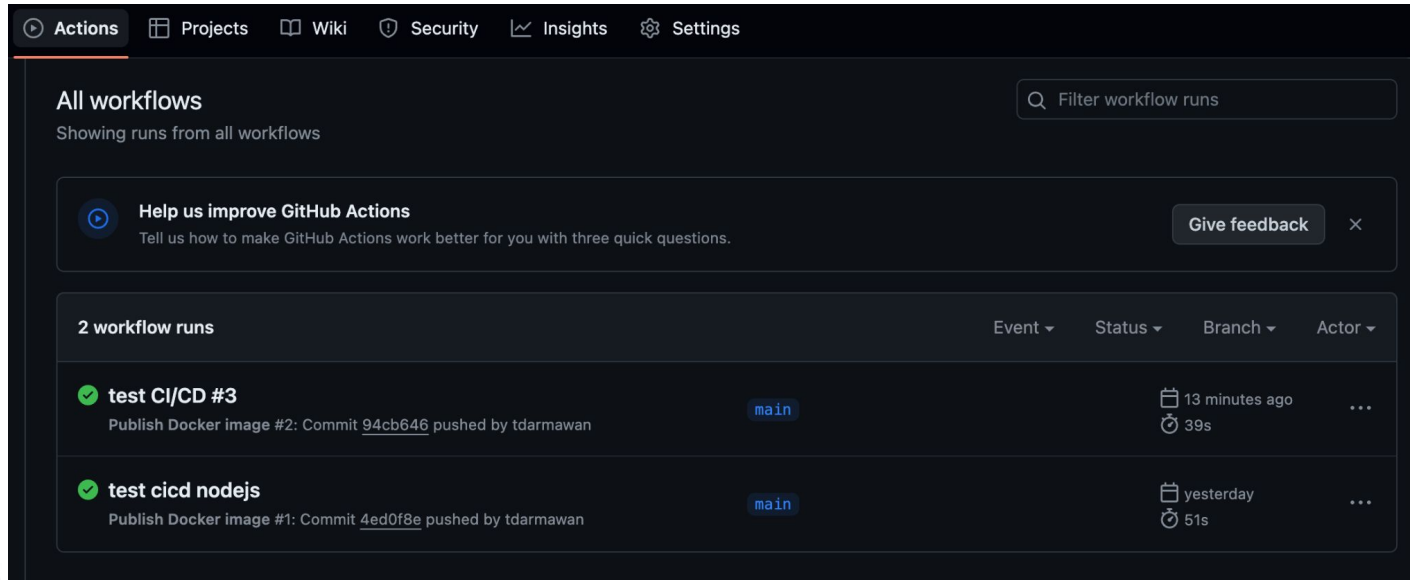
Setelah berhasil membuat repository baru di GitHub dan mengkonfigurasi Secrets Key, langkah selanjutnya adalah melakukan "git push" pada repository yang baru dibuat. Pastikan file aplikasi kita berhasil terunggah ke repository GitHub, dan periksa tab Actions untuk memastikan bahwa workflow yang telah kita buat berhasil berjalan.

```
● thomasdarmawan@Thomass-MacBook-Air inj-s-cicd % git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 815 bytes | 815.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/tdarmawan/injs-cicd.git
4ed0f8e..94cb646 main -> main
○ thomasdarmawan@Thomass-MacBook-Air inj-s-cicd %
```





CI/CD with Github Action

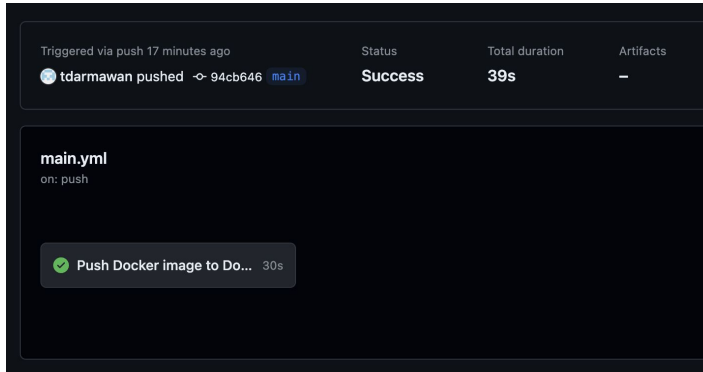
Lihat list workflows runs, setiap publish muncul ketika kita melakukan push pada github dan simbol checklist hijau mengartikan bahwa workflows CI/CD sudah berhasil. Untuk melihat lebih detail dari proses publish yang terjadi dapat dilakukan dengan cara memilih dan klik pada nama workflows yang ingin di lihat.




The screenshot shows the GitHub Actions interface. At the top, there are navigation tabs: Actions, Projects, Wiki, Security, Insights, and Settings. The 'Actions' tab is selected. Below the tabs, the heading 'All workflows' is displayed, followed by 'Showing runs from all workflows'. A search bar labeled 'Filter workflow runs' is on the right. A notification banner asks for feedback on GitHub Actions. Below this, a section titled '2 workflow runs' lists two runs. Each run has a green checkmark icon, a title, a description, the branch name 'main', and the time taken to complete.

	Event	Status	Branch	Actor
 test CI/CD #3 Publish Docker image #2: Commit 94cb646 pushed by tdarmawan			main	13 minutes ago 39s
 test cicd nodejs Publish Docker image #1: Commit 4ed0f8e pushed by tdarmawan			main	yesterday 51s


CI/CD with Github Action

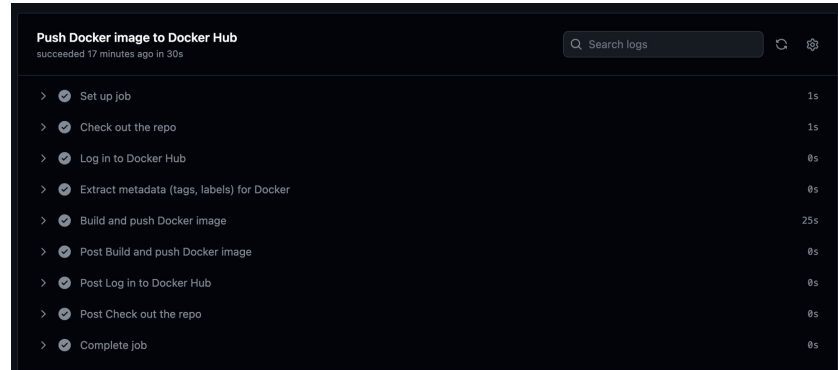


Triggered via push 17 minutes ago

	Status	Total duration	Artifacts
 tdarmawan pushed · 94cb646 · main	Success	39s	—










main.yml
on: push

 Push Docker image to Do... 30s



Push Docker image to Docker Hub
succeeded 17 minutes ago in 30s

Search logs

- >  Set up job 1s
- >  Check out the repo 1s
- >  Log in to Docker Hub 0s
- >  Extract metadata (tags, labels) for Docker 0s
- >  Build and push Docker image 25s
- >  Post Build and push Docker image 0s
- >  Post Log in to Docker Hub 0s
- >  Post Check out the repo 0s
- >  Complete job 0s

Gambar di atas menunjukkan status dan rincian proses dari workflow main.yml yang telah kita buat sebelumnya. Selanjutnya, kita dapat memastikan apakah aplikasi kita berhasil diunggah ke repository Docker di Docker Hub. Mari kita periksa Docker Hub seperti yang ditunjukkan pada slide berikutnya!

CI/CD with Github Action



tdarmawan / [Repositories](#) / [testdeploynode](#) / [General](#)

Using 0 of 1 private repositories. [Get more](#)

[General](#) [Tags](#) [Builds](#) [Collaborators](#) [Webhooks](#) [Settings](#)

Add a short description for this repository
The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.

[Update](#)

tdarmawan / testdeploynode

Description

This repository does not have a description

Last pushed: 20 minutes ago

Docker commands

To push a new tag to this repository:

[Public View](#)

```
docker push tdarmawan/testdeploynode:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
main		Image	20 minutes ago	20 minutes ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) .

[Upgrade](#)



Hands-On

silakan praktekan cara menerapkan CI/CD dari codingan kita menggunakan Github.