# 3.0 - GitHub Actions (poging 3)

## Introduction

GitHub contains a feature called GitHub actions. This is very useful to set up automation pipelines that get triggered either manually or automatically using a set of triggers.
As of a few years, it's also possible to run on-premises runners that execute commands in the order you need.

### Objective

In this assignment, we will explore GitHub Actions, to set up a powerful and versatile continuous integration and continuous deployment (CI/CD) pipeline. By employing custom GitHub Actions runners, we will learn how to optimize and tailor our workflows to suit our specific project requirements.

In sub-assignments, we will further enhance the capabilities of GitHub Actions by connecting it to a Kubernetes Cluster running on our host machine.

The final assignment will perform Azure CLI commands, in order to run AI pipelines.

### Knowledge

- YAML file structure
- Version Control Systems - Git / GitHub
  - Branching
- CI/CD pipelines

### Skills

- Setting up a CI/CD pipeline using GitHub Actions
- Automating a Docker Build using GitHub Actions
- Setting up a local runner on your host machine

### Necessities

- A Git Repository on GitHub
- An IDE such as Visual Studio Code
- A terminal

# Getting to know GitHub Actions

We will get started with a Basic GitHub Actions workflow now, so you know how everything works for the other coming courses.

## Create a new repository

We will need to create a new GitHub repository first, so we can add the GitHub Actions Runner to the repo.

- [x] ~~Create a new GitHub repository using your account~~
- [x] ~~Clone it to your PC~~
- [x] ~~Create a new branch on your repository called~~ ~~`01_DockerTest`~~

## Basic Docker application

GitHub Actions work best when a set of repeated actions have to be executed. This is why we will test it using a simple Dockerfile to create a very basic HTML website at first.

- Create a Dockerfile

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

- Create a basic HTML page `index.html`

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=
1.0">
    <title>Hello world</title>
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
```

💡 You can test this image by building it (I tagged it as `hello-world-nginx:v1`). Then you run it, with a port mapping to `chosen_port:80`. If you use VSCode, you can then `Open Browser` on the container, and it will tunnel the chosen port, and open it in the browser. You will see your **Hello world** content.

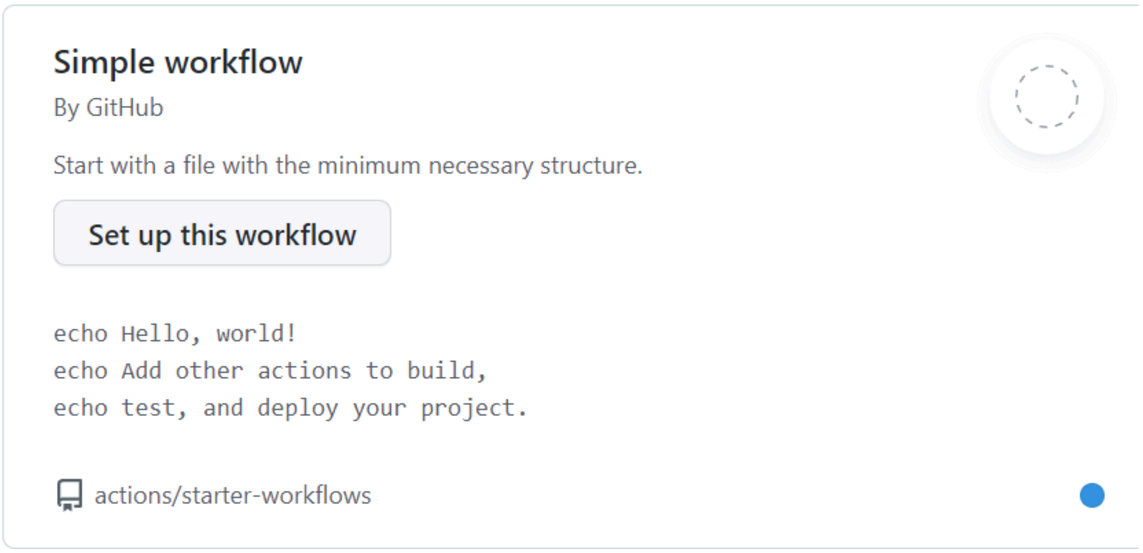
> ❓ **QUESTION 1 - Docker Run**
> Which command did you use to test this Docker image?

> 💻 **ANSWER 1 - Docker Run**
>
> ```
> # Bouw de Docker image
> docker build -t hello-world-nginx:v1 .
>
> # Draai de container met port mapping
> docker run -d -p 8080:80 --name test-nginx hello-world-nginx:v1
>
> # Na het testen opruimen:
> docker stop test-nginx
> docker rm test-nginx
> ```

- [x] ~~Push all of this content to a new branch called `01_DockerTest`. Your main branch stays clean and empty (apart from a Readme, if you want).~~

## Creating a GitHub Actions flow

- [x] ~~Go to the Actions tab on your GitHub Repository.~~
- [x] ~~Select the `Simple workflow`~~


Simple workflow
By GitHub
Start with a file with the minimum necessary structure.

```
echo Hello, world!
echo Add other actions to build,
echo test, and deploy your project.
```

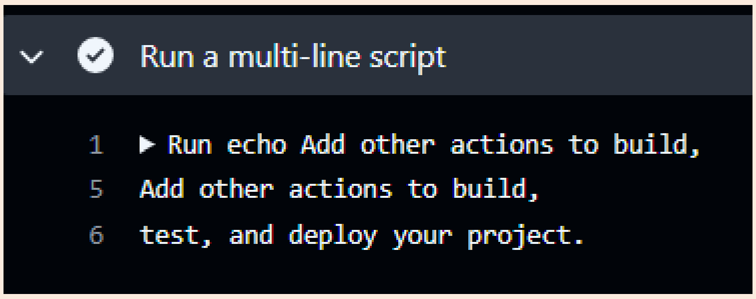
actions/starter-workflows

The GitHub interface to create a simple workflow through the browser

- [x] ~~Change the name of the Workflow to `My First Workflow`~~
- [x] ~~Commit this file to the Main branch~~
- [x] ~~Take a look at the output of your Job~~

❓ **QUESTION 2 - Multi-line script return**
What did the `multi-line script` return ?

**ANSWER 2 - Multi-line script return**


Run a multi-line script

```
▶ Run echo Add other actions to build,
Add other actions to build,
test, and deploy your project.
```

## YAML Syntax

Check out the documentation regarding the syntax for these YAML files.

Workflow syntax for GitHub Actions - GitHub Docs

A workflow is a configurable automated process made up of one or more jobs. You must create a YAML file to define your workflow configuration.

https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions

Following the syntax, it states that we have a few properties we need to fill in:

- `name: Starting Pipeline` — This is just the display name that we create for this pipeline
- `on: workflow_dispatch` — This states that we want to manually run this pipeline, in our case this is OK for now. We will gradually add more functionalities later.

To get a valid YAML file, we need at least one `job` . In this previous assignments, we will just experiment with one job to print something.

```
jobs:
  first-pipeline:
    # We will just use a machine that GitHub provides us
    runs-on: ubuntu-24.04 # Best to specify a fixed version, so it doesn't automatically change
    steps:
      # Make sure we can access our codebase and pull that into the runner
      - name: Check out code repository
        uses: actions/checkout@v4
```

## Changing the workflow file from your IDE

Open up any IDE → Visual Studio Code, Jetbrains Rider ...

To make sure your Visual Studio Code IDE checks the validity of the YAML file, you can install this extension and it will help you: https://marketplace.visualstudio.com/items?itemName=me-dutour-mathieu.vscode-github-actions

Open the repository that you cloned into your PC and search for the content. The file for the GitHub Actions workflow was added under `.github/workflows/blank.yml`

- [ ] Adapt the Workflow so you can build your Dockerfile, with the `hello-world-nginx:v1` flag. Try to make the Actions **start and run** the Docker container, map the right port using the right flags. You should be able to do that using simple Actions in GitHub.

❓ **QUESTION 3 - Changes to workflow**
What did you change ? Show the changes from your workflow file and explain it a little.

💻 **ANSWER 3 - Changes to workflow**

```
pull_request trigger toegevoegd
Docker build en run stappen toegevoegd
test of nginx draait met curl
stopt de container netjes op het einde
```

- [x] ~~Push your changes to the main branch.~~

❓ **QUESTION 4 - Workflow crash?**
Did your workflow fail? If so, why did it?

**ANSWER 4 - Workflow crash?**

> Ja, de workflow crasht, komt omdat als je de workflow op de mai
> n branch   test
> zonder dat de Dockerfile en index.html aanwezig zijn. De main br
> anch      bevat nog
> niet de Dockerfile en index.html die we op de 01_DockerTest bran
> ch hebben gemaakt.
>
> Dit is de fout:
> ERROR: failed to build: failed to solve: failed to read dockerfile: op
> en
> Dockerfile: no such file or directory
>
> Dit wordt opgelost door een pull request te maken van 01_Docker
> Test naar  main, zodat de bestanden worden samengevoegd.

▼ Click here for coming instructions, but they will give you the answer to the previous questions!

- [ ] Create a pull request from the `01_DockerTest` branch, into your `main` branch.
- [ ] Your pipeline will start already, as it also triggers on `pull_requests`

- [x] ~~If everything went well, you got the content of your nginx application inside of the GitHub Actions flow.~~
- [x] ~~You can now merge your Pull Request into your Main branch.~~

Try to experiment with the different options of the Docker build actions. There are a few community-available tasks you can experiment with.
This is one I like to use a lot:
https://github.com/marketplace/actions/build-and-push-docker-images

Using GitHub Actions, you can automate the tags to include the `branch` and the `commit-sha`.
Try that as well.

❓ **QUESTION 5 - Branch & Sha tags**
How did you automatically include the `branch` and `commit-sha` into the tags?

💻 **ANSWER 5 - Branch & Sha tags**

```
name: My First Workflow

on:
  workflow_dispatch:
  pull_request:
    branches:
      - main
  push:
    branches:
      - main

jobs:
  docker-build-and-push:
    runs-on: ubuntu-24.04

    steps:
      - name: Check out code repository
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: false  # Nog niet pushen, eerst lokaal testen
          tags: |
            hello-world-nginx:${{ github.ref_name }}
            hello-world-nginx:${{ github.sha }}
            hello-world-nginx:latest
          load: true
```

workflow_dispatch: Maakt het mogelijk om de workflow manueel t
e triggeren

```
pull_request: Triggert automatisch bij pull requests naar main
checkout@v4: Haalt de code op van de repository
docker build: Bouwt de Docker image met tag hello-world-nginx:v
1
docker run -d -p 8080:80: Draait de container in detached mode
met poort mapping
curl: Test of de webserver bereikbaar is
Cleanup: Stopt en verwijdert de container na het testen
```

---

## Running a Local runner on your host

Apart from using the GitHub Actions Runners that are hosted by GitHub themselves, we can get our own local runners as well. This is useful if you want to get more build time, or you require some bigger machines to perform your actions. We also need this if we have to access some things on our own network, such as the Kubernetes cluster that you could have running on your own host machine.

Documentation about GitHub Actions Runners can be found here

Follow these instructions to get started.

**Configure the runner for your OS**: https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/configuring-the-self-hosted-runner-application-as-a-service

### Using the self-hosted runner

- [x] ~~Adapt your Workflow to work on your self-hosted runner now.~~

```
# Use this YAML in your workflow file for each job
runs-on: self-hosted
```

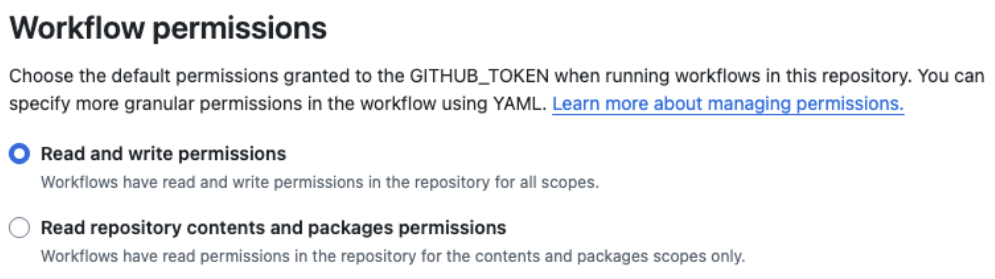
Try if everything still runs well.

## GitHub Container Registry

Now that we are working with Docker Images, we can also make use of the GitHub Container Registry. This Registry is a private one that is included in your GitHub subscription, and replaces Docker Hub. The useful thing is that we can configure access to the Docker Images (referred to as Packages in GitHub) next to our repository settings.

We can even connect the repository with the package, so that they are both in sync. This is not the only advantage. As we're working in the GitHub Repository, we don't need an extra account on Docker Hub anymore, and we can use the same Personal Access Tokens (PAT) with the necessary settings.

🚨 You'll need to configure this in your Repository settings so that the GitHub Actions runners have access to push to the GitHub Registry.
Settings > Actions > General



**Workflow permissions**

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. Learn more about managing permissions.

[● Read and write permissions] Workflows have read and write permissions in the repository for all scopes.

○ Read repository contents and packages permissions
Workflows have read permissions in the repository for the contents and packages scopes only.

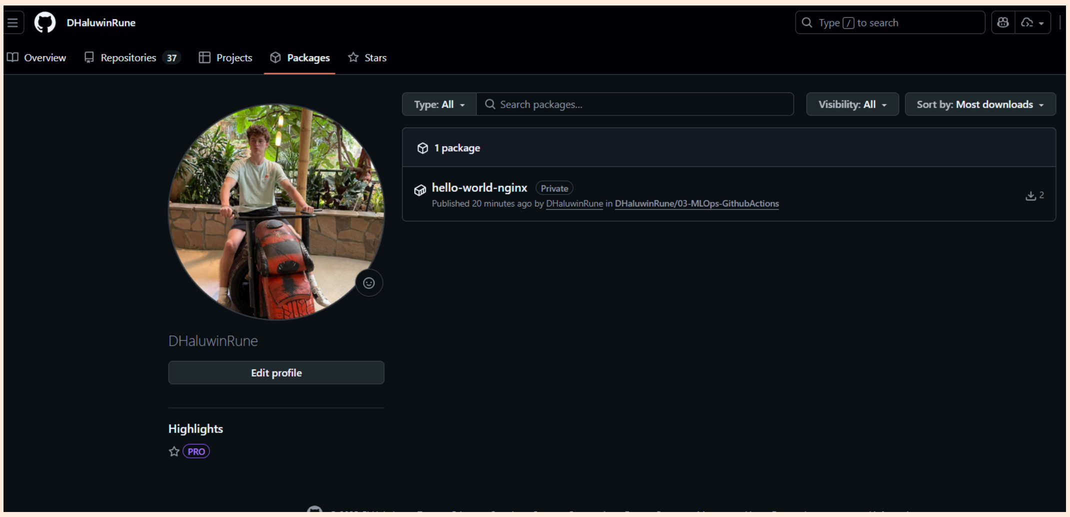
Allow the `GITHUB_TOKEN` write permissions to the packages.

Implement this by prefixing the Docker image using `ghcr.io/<your-github-username>`.

Use the Tagging and Pushing like in the previous chapter to allow the image to be added to the GitHub Packages.

❓ **QUESTION 6 - Screenshot of packages**
Show the result of the packages on GitHub.

## ANSWER 6 - Screenshot of packages



1 package

hello-world-nginx Private
Published 20 minutes ago by DHaluwinRune in DHaluwinRune/03-MLOps-GithubActions

2

## QUESTION 7 - Whole workflow

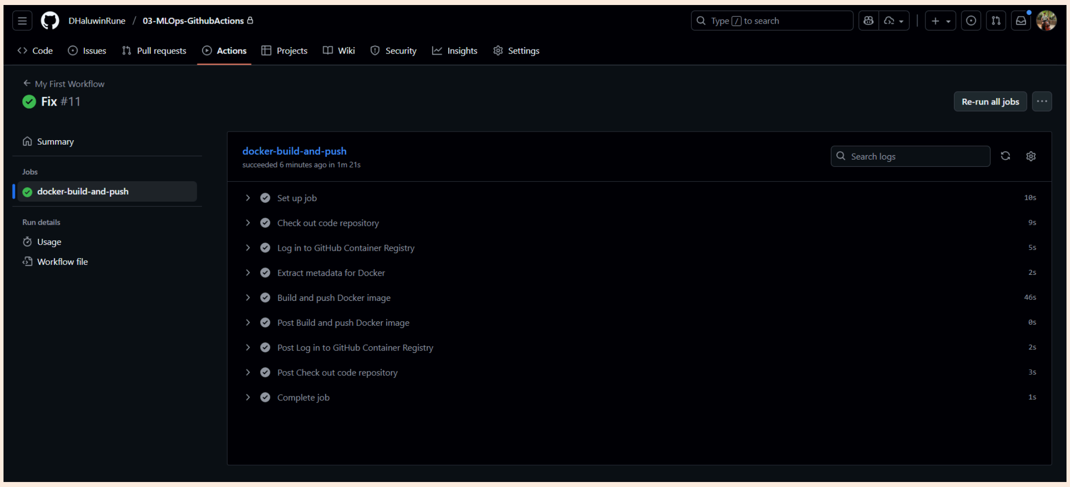
Paste your whole workflow in here as well as a screenshot.

💻 **ANSWER 7 - Whole workflow**

```
name: My First Workflow

on:
  workflow_dispatch:
  pull_request:
    branches:
      - main
  push:
    branches:
      - main

jobs:
  docker-build-and-push:
    runs-on: self-hosted

    permissions:
      contents: read
      packages: write

    steps:
      - name: Check out code repository
        uses: actions/checkout@v4

      - name: Log in to GitHub Container Registry
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${{ github.repository_owner }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract metadata for Docker
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ghcr.io/${{ github.repository_owner }}/hello-world-
nginx
```

```
          tags: |
            type=ref,event=branch
            type=sha,prefix={{branch}}-
            type=raw,value=latest

      - name: Build and push Docker image
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

Take a screenshot of the final Succeeded Workflow



My First Workflow
Fix #11

docker-build-and-push
succeeded 6 minutes ago in 1m 21s

```
Set up job                                10s
Check out code repository                  9s
Log in to GitHub Container Registry        5s
Extract metadata for Docker                2s
Build and push Docker image               46s
Post Build and push Docker image           0s
Post Log in to GitHub Container Registry   2s
Post Check out code repository             3s
Complete job                               1s
```

---

## What did you learn?

Fill in something that you learned during this lesson

- Hoe je GitHub Actions workflows configureert met YAML syntax
- Het verschil tussen GitHub-hosted en self-hosted runners
- Hoe je Docker images bouwt en test binnen een CI/CD pipeline
- Het gebruik van GitHub Container Registry als alternatief voor Docker Hub
- Hoe je automatisch tags genereert op basis van branch en commit SHA

- Het belang van permissions voor het pushen van packages
- Hoe je secrets en tokens veilig gebruikt in workflows
- Het principe van Infrastructure as Code toegepast op CI/CD

### Give three interesting exam questions

- Wat is het verschil tussen een GitHub-hosted runner en een self-hosted runner?
- Hoe kun je automatisch tags toevoegen aan Docker images in GitHub Actions?
- Waarom is het GitHub Container Registry handig in vergelijking met Docker Hub?

## Handing in this assignment

You can hand this in by duplicating this document on Notion, print this document as a `.pdf` and submit that document on Leho.

Also hand in the written Source Code in a `.zip` file please.

Checkboxes:

- [x] ~~I have duplicated this file~~
- [x] ~~I have filled in all the answers~~
- [x] ~~I have added something that I learned~~
- [x] ~~I have added three interesting exam questions~~
- [x] ~~I have zipped my project and uploaded it to Leho~~