

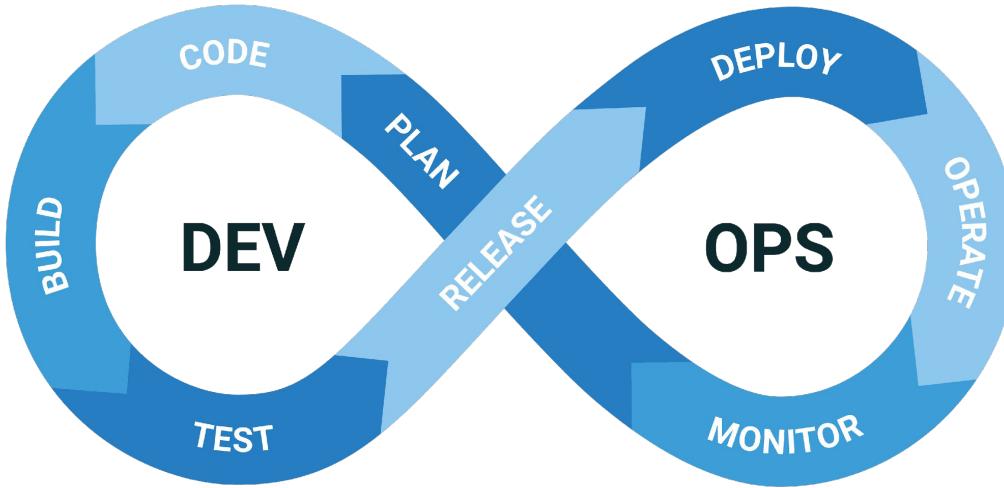
# Continuous integration (CI) / Continuous delivery (CD)

Mads Nyborg  
Business Academy Copenhagen

# Continuous Integration / Continuous Delivery pipeline

- **Code**
  - The developer writes code locally.
  - The code is stored in a version control system (locally: git - remotely: GitHub, GitLab, or Bitbucket).
- **Build**
  - The code is compiled or packaged if needed (e.g., Java to .class, TypeScript to JavaScript).
  - Dependencies are installed (e.g., npm install, mvn install).
- **Test**
  - Unit tests (eg. Jest), integration tests, or end-to-end tests are run automatically.
  - The goal is to catch bugs early.
- **Release**
  - Approved code is prepared for deployment.
  - This can mean creating a release build or generating a Docker image.
- **Deploy**
  - The code is rolled out to staging or production.
  - Can be automatic (CD) or require manual approval.
- **Operate**
  - The application runs in production.
  - Monitoring and logging detect issues.
- **Monitor**
  - Performance, stability, and user behavior are tracked.
  - Reports or alerts can trigger new backlog items.

# DevOps culture



**DevOps** is a cultural and organizational approach that fosters collaboration between development and operations teams

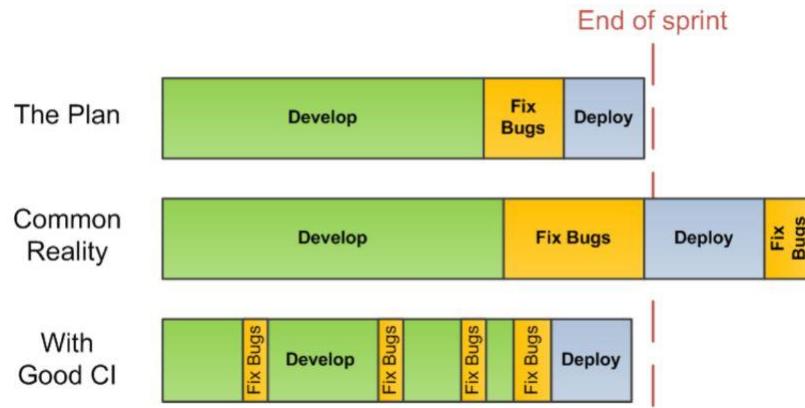
**CI/CD** (continuous integration and continuous delivery) is a set of practices for automating the integration and deployment of code changes

# Why CI/CD ?

- Frequent changes -> less integration problems
- Bugs are detected early -> saves money
- Avoid last minute chaos
- Transparency to all
- Testing on production-like environment
- Easy to rollback in case of any issue
- Enforce of automation culture
- Enforce DevOps culture
- Make the developers accountable and take ownership

# Integration early

Continuous integration involves integration early and often, in order to avoid “Integration hell”



# CI Principles



## Automate the build

(single command ), Automate all.  
Minimize human error



The entire build process (compiling, packaging, etc.) should be runnable with one command — no manual steps.



## Build is self-testable

Ensure code integrity  
Keep the branch stable



The build should automatically run tests to verify the integrity of the code.



## Baseline branch is open consistently

Keep the branch stable



The main branch (e.g., main or master) should always be in a deployable state.



## Every commit should be built

Detect issues early



Every commit triggers a build (and ideally tests).



## Build should be fast

Increase developer productivity



Feedback should be quick so developers know within minutes if something failed.



## Test Env is clone of Production

(as much as we can)  
Reduce deployment risks



The test environment should mimic production as closely as possible (OS, database, configurations).



## Everyone can see the status

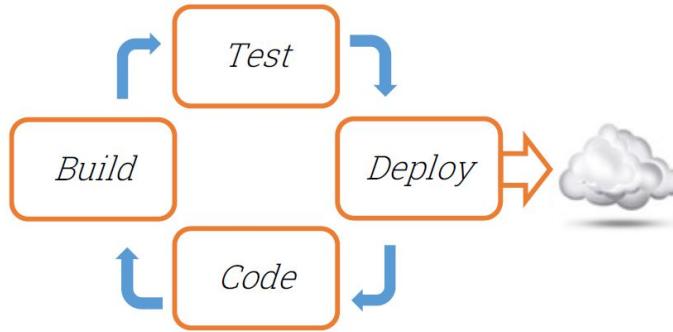
– Transparency  
Encourage shared responsibility



Build and test results should be visible to the entire team.

# Continuous Delivery & Deployment

Continuous Delivery (CD) is a software development discipline where you build software in such a way that the software can be released to production at any time.



Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

# CD is not only in Computer Software

Tesla Model S gets firmware updates on regular basis for both UI and major elements (suspension, acceleration and more)



# Github actions

# CI and GitHub Actions

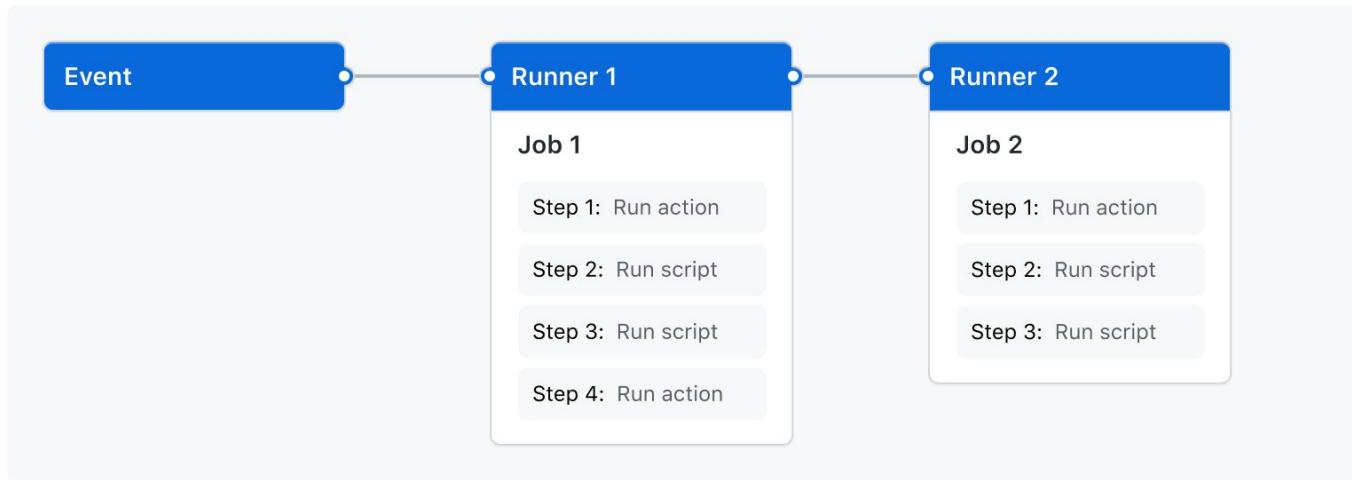
- Use GitHub Actions to create workflows that can **build the code** in your repository and **run your tests**.
- Workflows can run on **GitHub-hosted virtual machines**.
- Configure a CI workflow to run when a **GitHub event** occurs (e.g. when new code is pushed to your repository).
- GitHub runs your CI tests and provides the results of each test in the pull request, so you can see whether the change in your branch introduces an error.
- When all CI tests in a workflow pass, the changes you pushed are ready to be reviewed by a team member or merged.

# GitHub Actions - workflow

- [Understanding GitHub Actions](#)

specified in .yml file

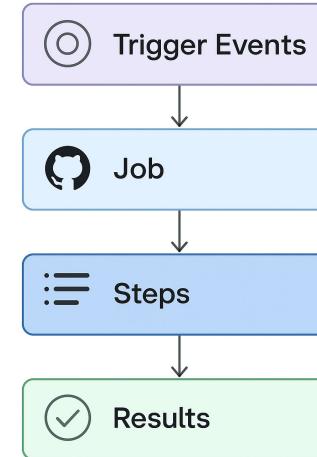
must be stored in a subfolder named /.github/workflows



# GitHub Actions - workflow .yml file

- Workflows

- configurable automated process that will run one or more jobs
- defined by a YAML file checked in to your repository
- run when triggered by an event in your repository e.g. push or manually, or at a defined schedule.



# GitHub Actions

- Job
  - set of steps in a workflow that is executed on the same runner.
  - Each step is either a shell script that will be executed, or an action that will be run.
  - Steps are executed in order and are dependent on each other.
  - Since each step is executed on the same runner, you can share data from one step to another. E.g., you can have a step that builds your application followed by a step that tests the application that was built.

# GitHub Actions

- Runner
  - a server that runs your workflows when they're triggered.
  - Each runner can run a single job at a time.
  - GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows.
  - Each workflow run executes in a fresh, newly-provisioned virtual machine.

# Workflow file - example

```
name: Hello and Goodbye Workflow

on:
  push:
    branches:
      - main          # Only run when pushing to the main branch
  workflow_dispatch: # Manual trigger from GitHub Actions UI

jobs:
  hello:
    runs-on: ubuntu-latest # spin up a virtual ubuntu server on GitHub
    steps:
      - name: Say Hello
        run: echo "Hello world"

  goodbye:
    runs-on: ubuntu-latest
    steps:
      - name: Say Goodbye
        run: echo "Goodbye"
```

# Now let's make the example ...

Make a repository and click "creating a file"

The screenshot shows the GitHub interface for creating a new repository. The repository name is 'nmit\_actions\_test'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the repository name, there are sections for 'Set up GitHub Copilot' and 'Add collaborators to this repository'. A prominent blue box at the bottom left provides instructions for quick setup, mentioning 'creating a new file' or 'uploading an existing file'. It also lists command-line steps for initializing a repository. The URL [https://github.com/mnyborg/nmit\\_actions\\_test.git](https://github.com/mnyborg/nmit_actions_test.git) is displayed.

Set up GitHub Copilot  
Use GitHub's AI pair programmer to autocomplete suggestions as you code.  
[Get started with GitHub Copilot](#)

Add collaborators to this repository  
Search for people using their GitHub username or email address.  
[Invite collaborators](#)

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH [https://github.com/mnyborg/nmit\\_actions\\_test.git](https://github.com/mnyborg/nmit_actions_test.git)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# nmit_actions_test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/mnyborg/nmit_actions_test.git
git push -u origin main
```

# Now let's make the example ...

Enter file name. `./.github/workflows/test.yml` and content. Commit changes

The screenshot shows a GitHub repository interface. The repository is named `mnyborg / nmit_actions_test`. The user is in the `Code` tab, viewing the `test.yml` file under the `.github / workflows` directory. The file content is highlighted with a red box:

```
1 name: Hello and Goodbye Workflow
2
3 on:
4   push:
5     branches:
6       - main          # Only run when pushing to the main branch
7     workflow_dispatch: # Manual trigger from GitHub Actions UI
8
9 jobs:
10   hello:
11     runs-on: ubuntu-latest # spin up a virtual ubuntu server on GitHub
12     steps:
13       - name: Say Hello
14         run: echo "Hello world"
15
16   goodbye:
17     runs-on: ubuntu-latest
18     steps:
19       - name: Say Goodbye
20         run: echo "Goodbye"
```

At the top right, there are two buttons: `Cancel changes` and `Commit changes...`, with the latter also highlighted by a red box.

To the right of the code editor is a sidebar titled `Getting started with a workflow`. It contains introductory text and a list of steps to customize workflow triggers:

- Set your workflow to run on push events to the `main` and `release/*` branches

```
on:
  push:
    branches:
      - main
      - release/*
```

# Now let's make the example ...

Observe workflow run

The screenshot shows the GitHub Actions interface for the repository 'mnyborg/nmit\_actions\_test'. The 'Actions' tab is selected, highlighted by a red box. On the left, a sidebar lists management options: Caches, Attestations, Runners, Usage metrics, and Performance metrics. A red box highlights the 'Hello and Goodbye Workflow' card, which is currently active. The main area displays the workflow details: 'Hello and Goodbye Workflow' (test.yml), 1 workflow run, and a note that it has a workflow\_dispatch event trigger. A red box highlights the first workflow run card, showing the event was triggered by a commit from 'mnyborg' on the 'main' branch, and the status is 'in progress'.

# Now let's make the example ...

Observe workflow completes

The screenshot shows the GitHub Actions interface for the repository 'nmyborg/nmit\_actions\_test'. The 'Actions' tab is selected. On the left, a sidebar lists management options: Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area displays the 'Hello and Goodbye Workflow' with its configuration file 'test.yml'. A message encourages users to help improve GitHub Actions. Below, it shows '1 workflow run' triggered by a 'workflow\_dispatch' event. The most recent run was created by 'mnyborg' on the 'main' branch, pushed 1 minute ago, and completed 10s ago. A red box highlights this specific run.

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Actions New workflow

All workflows

Hello and Goodbye Workflow

Management

Caches

Attestations

Runners

Usage metrics

Performance metrics

Help us improve GitHub Actions

Tell us how to make GitHub Actions work better for you with three quick questions. Give feedback

1 workflow run

Event Status Branch Actor

This workflow has a `workflow_dispatch` event trigger. Run workflow

Create test.yml

Hello and Goodbye Workflow #1: Commit [b92b7db](#) pushed by [mnyborg](#) main 1 minute ago 10s

# Now let's make the example ...

Observe workflow complete - detailed view

The screenshot shows a GitHub Actions workflow run details page for a repository named 'mnyborg / nmit\_actions\_test'. The workflow is named 'Create test.yml #1' and was triggered via push 3 minutes ago. The status is 'Success' with a total duration of '10s'. The workflow file is 'test.yml' and it runs on pushes. Two jobs are listed: 'hello' and 'goodbye'. The 'hello' job completed successfully in 3s, and the 'goodbye' job completed successfully in 2s. A red box highlights the 'hello' job.

Triggered via push 3 minutes ago

mnyborg pushed → b92b7db main

Status: Success | Total duration: 10s | Artifacts: -

test.yml  
on: push

hello 3s

goodbye 2s

# Now let's make the example ...

Observe workflow results

The screenshot shows the GitHub Actions interface for a repository named 'mnyborg / nmit\_actions\_test'. The 'Actions' tab is selected. A workflow named 'Hello and Goodbye Workflow' is shown, with a successful run labeled 'Create test.yml #1'. The 'hello' job is expanded, showing its steps:

- > Set up job (0s)
- < Say Hello (0s)
  - 1 ► Run echo "Hello world"
  - 4 Hello world
- > Complete job (0s)

A red box highlights the 'Say Hello' step, which contains the command 'Run echo "Hello world"' and its output 'Hello world'.

# Now let's make the example ...

- Make a change to the output from the echo command in the hello job:

steps:

- name: Say Hello  
run: echo "Hello world 2"

- Commit changes, commit message “Update test.yml”
- Observe the workflow runs again due to the push trigger

The screenshot shows a list of workflow runs. At the top, there are filters for Event, Status, Branch, and Actor. Below the filters, there are two entries:

- Update test.yml**: This entry has a green checkmark icon. It shows the commit message "Hello and Goodbye Workflow #2: Commit 74d8bbf pushed by mynborg" and a status of "main". The run was completed "now" (12s ago). There is also a three-dot menu icon.
- Create test.yml**: This entry has a green checkmark icon. It shows the commit message "Hello and Goodbye Workflow #1: Commit b92b7db pushed by mynborg" and a status of "main". The run was completed "1 hour ago" (10s ago). There is also a three-dot menu icon.

# We now extend the example ...

- We want to add two javascript files [hello.js](#) and [goodbye.js](#) to the repository that produce the same output as before
- We will need to use two premade action scripts from GitHub marketplace:
  - `actions/checkout` - checks out the content of the repo to the runner
  - `actions/setup-node` - installs node on the runner

[hello.js](#):

```
console.log("Hello from JavaScript!");
```

[goodbye.js](#):

```
console.log("Goodbye from JavaScript!");
```

# We now extend the example ...

[hello.js](#) and [goodbye.js](#) created in root folder of repository

The screenshot shows a GitHub repository page for 'nmit\_actions\_test'. The 'Code' tab is selected. Recent commits are listed:

- mnyborg Create goodbye.js (green checkmark) - b22be16 · 2 minutes ago · 6 Commits
- .github/workflows Update test.yml - 1 hour ago
- goodbye.js Create goodbye.js - 2 minutes ago
- hello.js Create hello.js - 3 minutes ago

A red box highlights the last two commits: 'goodbye.js' and 'hello.js'. The right sidebar shows repository details: 1 Branch, 0 Tags, Go to file search bar, + button, Code dropdown, About section (Demonstrate GitHub Actions at NMIT), Activity, 0 stars, 0 watching, 0 forks, and Releases (No releases published, Create a new release).

# We now extend the example ...

Before we define a new workflow we will disable the original workflow

The screenshot shows the GitHub Actions page for the repository 'mnyborg/nmit\_actions\_test'. The 'Actions' tab is selected. On the left sidebar, under 'Actions', the 'Hello and Goodbye Workflow' is highlighted. In the main content area, there is a card titled 'Hello and Goodbye Workflow' with the file name 'test.yml'. Below the card, there is a section for 'Help us improve GitHub Actions' and another section showing '4 workflow runs'. To the right of the card, there is a '...' button, which has a red box around it. A dropdown menu from this button contains the options 'Create status badge', 'Pin workflow', and 'Disable workflow', with 'Disable workflow' also having a red box around it.

# We now extend the example ...

Make a new workflow: test2.yml

```
name: Hello and Goodbye Workflow with javascript

on:
  push:
    branches:
      - main
  workflow_dispatch:

jobs:
  hello:
    runs-on: ubuntu-latest # Use the latest Ubuntu virtual machine
    steps:
      # Download the repository content to the runner
      - name: Check out repository
        uses: actions/checkout@v4

      # Set up a specific Node.js version
      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 20

      # Run the hello.js script from the repository
      - name: Run Hello script
        run: node hello.js
```

# We now extend the example ...

Make a new workflow

```
goodbye:
  # This job will only start after the "hello" job has finished successfully
  needs: hello
  runs-on: ubuntu-latest
  steps:
    # Download the repository content again
    - name: Check out repository
      uses: actions/checkout@v4

    # Set up Node.js
    - name: Set up Node.js
      uses: actions/setup-node@v3
      with:
        node-version: 20

    # Run the goodbye.js script from the repository
    - name: Run Goodbye script
      run: node goodbye.js
```

# We now extend the example ...

Observe the result. Since we have used “needs” the sequence of the jobs is guaranteed

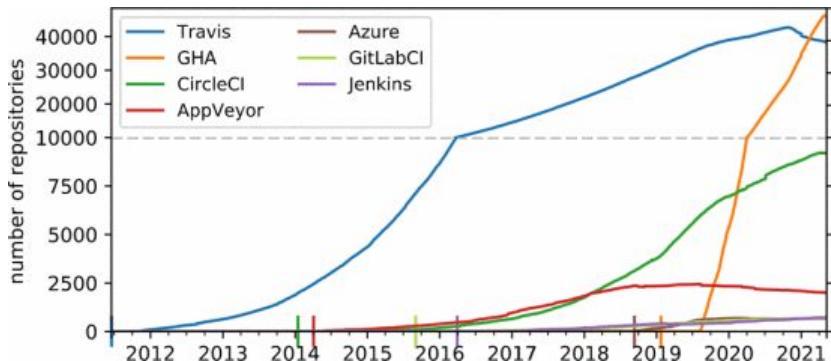
The screenshot shows a GitHub Actions workflow run for a repository named "mnyborg / nmit\_actions\_test". The workflow file is "test2.yml #1". The run was triggered via a push 8 minutes ago and is marked as "Success" with a total duration of 20s. The workflow contains two jobs: "hello" and "goodbye". The "hello" job ran for 8s and succeeded. The "goodbye" job ran for 4s and succeeded. A red box highlights the sequence of the jobs, showing the flow from the "hello" job to the "goodbye" job.

Job	Duration	Status
hello	8s	Success
goodbye	4s	Success

```
graph LR; hello[hello] -- "8s" --> goodbye[goodbye]; goodbye -- "4s" --> end[ ]
```

# CI platforms and popularity

Number of repositories using a specific CI.



Artikel: On the rise and fall of CI services in GitHub

<https://ieeexplore.ieee.org/document/9825792>

CI	URL	first observed on	repositories			usages	
			#	%	cum. %	%	cum. %
Travis	http://travis-ci.org	Jun 10, 2011	53,401	58.2%	58.2%	44.9%	44.9%
GHA	http://github.com/features/actions	Jan 23, 2019	46,416	50.6%	90.9%	39.0%	83.9%
CircleCI	http://circleci.com	Jan 15, 2014	11,431	12.4%	98.1%	9.6%	93.5%
AppVeyor	http://appveyor.com	Apr 04, 2014	3,553	3.9%	98.3%	3.0%	96.5%
Azure	http://azure.microsoft.com	Sep 11, 2018	1,045	1.1%	98.7%	0.9%	97.3%
GitLab CI	http://docs.gitlab.com/ee/ci	Sep 02, 2015	1,018	1.1%	99.1%	0.9%	98.2%
Jenkins	http://www.jenkins.io	Mar 30, 2016	1,008	1.1%	99.6%	0.8%	99.0%
Others	N/A	Oct 23, 2013	1,138	1.2%	100.0%	1.0%	100.0%

119,033 repositories in total containing configuration files,  
e.g. .yml files for GHA

Data obtained using libraries.io