

What is CI and CD?

Continuous Integration

Continuous Integration (CI) is a development practice where developers frequently integrate their code changes into a shared repository, preferably several times a day. Each integration is automatically verified by

1. Building the project and
2. Running automated tests.

This process allows teams to detect problems early, improve software quality, and reduce the time it takes to validate and release new software updates.

Continuous Deployment

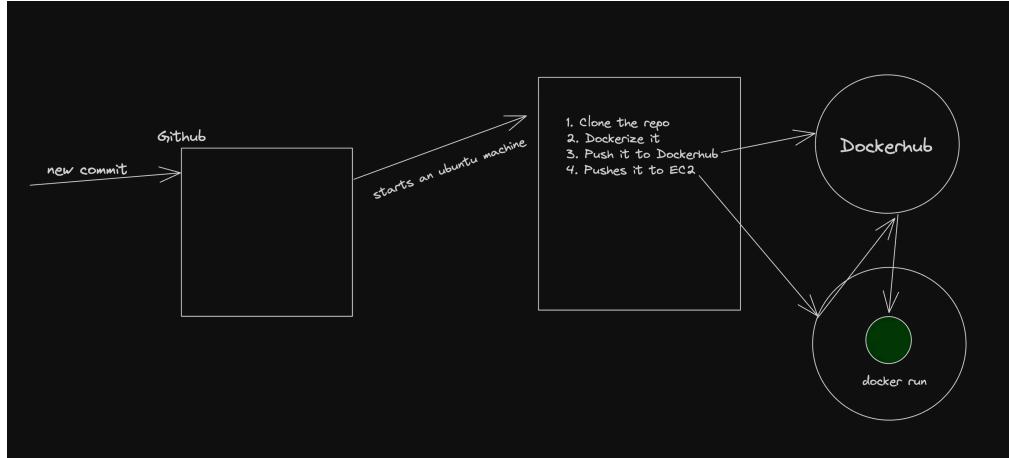
As the name suggests, deploying your code **continuously** to various environments (dev/stage/prod)

Continuous Deployment in Github

We'll be deploying a next.js app to EC2 servers via Docker

 You don't really need Docker here, since it's deploying on a simple EC2 server.
If you deploy to
1. GCP App runner
2. ECS
3. Kubernetes
then it makes more sense to deploy a **dockerised**

Architecture diagram



 Last step keeps changing based on where you're pushing your image

Monorepo we're dealing with today

<https://github.com/100xdevs-cohort-2/week-18-2-ci-cd>

This monorepo has 3 apps inside -

1. bank-webhook
2. merchant-app
3. user-app

We'll be deploying all three to the same ec2 instance

How to create a CI/CD pipeline?

For Github, you can add all your pipelines to `.github/workflows`

For eg -

<https://github.com/code100x/cms/blob/main/.github/workflows/lint.yml>

```

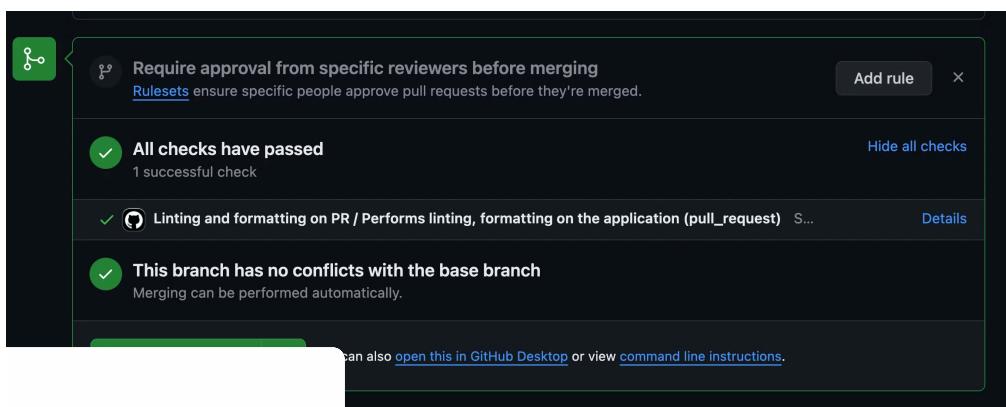
1   name: Linting and formatting on PR
2   on:
3     pull_request:
4       branches:
5         - '**'
6
7   jobs:
8     Continuous-Integration:
9       name: Performs linting, formatting on the application
10      runs-on: ubuntu-latest
11      steps:
12        - name: Checkout the Repository
13          uses: actions/checkout@v3
14        - name: Install Dependencies
15          run: npm install --legacy-peer-deps
16        - name: Run linting check
17          run: npm run lint:check
18        - name: Check formatting
19          run: npm run format:fix
20
21
22
23
24

```

The diagram shows a GitHub workflow YAML file with handwritten annotations explaining each section:

- on:** A box highlights the `pull_request` section under `on`. An arrow points from this box to the text: "Anytime a PR comes on any branch, run the following jobs".
- jobs:** A box highlights the `Continuous-Integration` job. An arrow points from this box to the text: "Name of the job".
- steps:** A box highlights the first step of the job. An arrow points from this box to the text: "Name of the job" and "Machine to run the job on".
- name:** An arrow points from the `name` key to the text: "Clone the repo on the machine".
- uses:** An arrow points from the `uses` key to the text: "Install dependencies".
- run:** An arrow points from the `run` key to the text: "Runs lint checks, if it fails, the workflow fails".

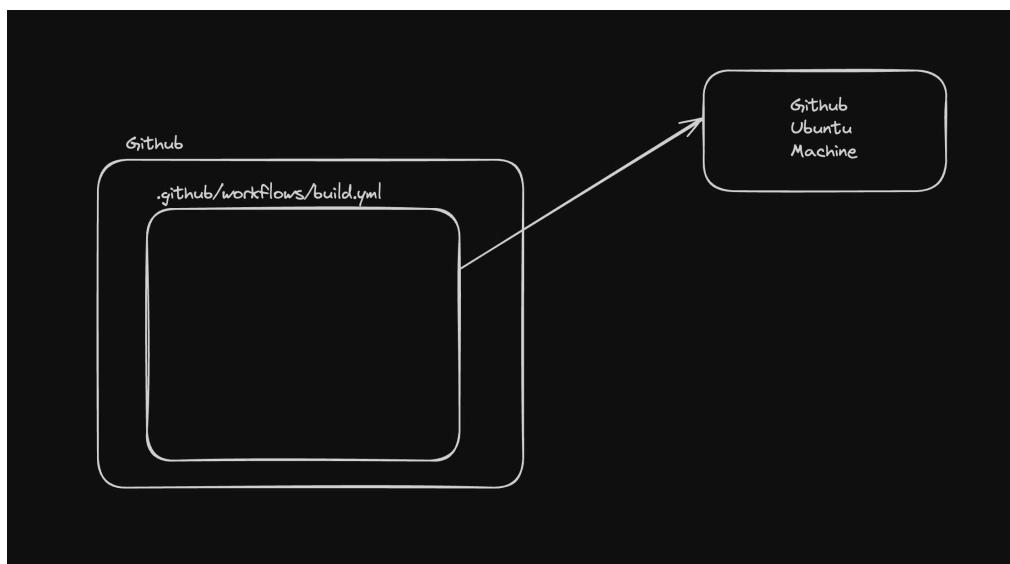
CD pipelines look like this finally -



- ☰ CI/CD 1 of 8 : - Use <https://onlineyamltools.com/convert-yaml-to-json> to see the pipeline in json

Step 1 – Create the CI pipeline

Make sure that whenever someone tries to create a PR, we build the project and make sure that it builds as expected



Lets add a build pipeline for our repo

Anytime a user creates a PR, we need to run `npm run build` and only if it succeeds should the workflow succeed

- Fork the main repo - <https://github.com/100xdevs-cohort-2/week-18-2-ci-cd>
- Add `.github/workflows/build.yml` in the root folder
- Create the workflow

```
name: Build on PR
```

```
on:
```

```
  pull_request:
```

```
    branches:
```

```
      - master
```

```
jobs:
```

```
  build:
```

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

```
    steps:
```

```
      - uses: actions/checkout@v3
```

```
      - name: Use Node.js
```

```
        uses: actions/setup-node@v3
```

```
        with:
```

```
          node-version: '20'
```

```
      - name: Install Dependencies
```

```
        run: npm install
```

```
      - name: Run Build
```

```
        run: npm run build
```

- Push this to master branch

- Create a new branch with some minimal changes and create a PR from

```
  .github/workflows/build.yml
```



Let's add a deploy step

- Create dockerfiles for the `apps` you have
 - Create `docker/Dockerfile.user`

```
FROM node:20.12.0-alpine3.19

WORKDIR /usr/src/app

COPY package.json package-lock.json turbo.json tsconfig.json .

COPY apps ./apps
COPY packages ./packages

# Install dependencies
RUN npm install
# Can you add a script to the global package.json that does this?
RUN cd packages/db && npx prisma generate && cd ..

# Can you filter the build down to just one app?
RUN npm run build

CMD ["npm", "run", "start-user-app"]
```

- Add `start-user-app` script to the root `package.json`

```
"start-user-app": "cd ./apps/user-app && npm run start"
```

💡 You don't really need to build every app for every dockerfile. Can you change the build command so that only a single app is built for each dockerfile?

```
> [5/8] 📁 / packages /> packages
=> [6/8] RUN npm install
=> [7/8] RUN npm run build
=> => # docs:build: - Environments: .env
=> => # docs:build:
=> => # web:build: ▲ Next.js 14.1.4
=> => # web:build:
=> => # web:build: Creating an optimized production build ...
=> => # docs:build: Creating an optimized production build ...
```

- Create the CI pipeline that

- Clones the repo
- Builds the docker image
- Pushes the docker image

`name: Build and Deploy to Docker Hub`

```
on:
  push:
    branches:
      - master

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Check Out Repo
        uses: actions/checkout@v2

      - name: Log in to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Build and Push Docker image
        uses: docker/build-push-action@v2
        with:
          context: .
          file: ./Dockerfile
          push: true
          tags: 100xdevs/web-app:latest # Replace with your Docker Hub username and repository name

      - name: Verify Pushed Image
        run: docker pull 100xdevs/web-app:latest # Replace with your Docker Hub username and repository name
```

- Make sure to add the `dockerhub` secrets to `github secrets` of the repo (`DOCKER_USERNAME`, `DOCKER_PASSWORD`)
- You should see a workflow running

CI/CD 1 of 8

Protect this branch from force pushing or deletion, or require status checks before merging. [View documentation.](#)

master ▾ **3 Branches** **0 Tags** [Go to file](#)

hkirat Added deploy step

.github/workflows **1 queued check** **Added deploy step**

https://getsubmiss... note in paris - Go... ad New tab

100xdevs-cohort-2 / week-18-2-final

Code Issues Pull requests 2 Actions Projects Wiki ⚙

← Build and Deploy to Docker Hub **Added deploy step #1**

Summary **build-and-push** Started 1m 10s ago

Check dockerhub to ensure the image has indeed reached there

100xdevs / Repositories / web-app / Tags

General Tags Builds Collaborators Webhooks Settings

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

Sort by Newest Filter Tags Delete

TAG	Digest	OS/ARCH	Last pull	Compressed Size
latest				

Last pushed a minute ago by 100xdevs

docker pull 100xdevs/web-app:latest [Copy](#)

Let's pull the docker image

Ref - <https://github.com/appleboy/ssh-action>

- Create an ec2 server
 - Download its keypair file
 - Allow http/https traffic
 - Ubuntu base image
- Download docker on the machine

<http://aws.amazon.com/amazon-linux-ami/> r.com/engine/install/ubuntu/

- sudo docker run hello-world

CI/CD 1 of 8

Update workflow to pull the latest image on the ec2 machine

name: Build and Deploy to Docker Hub

on:

push:

branches:

- master # Adjusted to trigger on pushes to master

jobs:

build-and-push:

runs-on: ubuntu-latest

steps:

- name: Check Out Repo

uses: actions/checkout@v2

- name: Prepare Dockerfile

run: cp ./docker/Dockerfile.user ./Dockerfile

- name: Log in to Docker Hub

uses: docker/login-action@v1

with:

username: \${{ secrets.DOCKER_USERNAME }}

password: \${{ secrets.DOCKER_PASSWORD }}

- name: Build and Push Docker image

uses: docker/build-push-action@v2

with:

context: .

file: ./Dockerfile

push: true

tags: 100xdevs/web-app:latest

- name: Verify Pushed Image

run: docker pull 100xdevs/web-app:latest

- name: Deploy to EC2

uses: appleboy/ssh-action@master

with:

host: \${{ secrets.SSH_HOST }}

username: \${{ secrets.SSH_USERNAME }}

key: \${{ secrets.SSH_KEY }}

script: |

sudo docker pull 100xdevs/web-app:latest

sudo docker stop web-app || true

sudo docker rm web-app || true

sudo docker run -d --name web-app -p 3005:3000 100xdevs/web-app:latest

- Point userapp.your_domain.com to the IP of the server
- Add nginx reverse proxy to forward requests from userapp.your_domain.com to port on which the app is running

```
server {
    server_name userapp.100xdevs.com;
```

```
location / {  
    proxy_pass http://localhost:3005;  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection 'upgrade';  
    proxy_set_header Host $host;  
    proxy_cache_bypass $http_upgrade;  
  
    # Basic Authentication  
    auth_basic "Restricted Content";  
    auth_basic_user_file /etc/nginx/.htpasswd;  
}  
  
listen 443 ssl; # managed by Certbot  
ssl_certificate /etc/letsencrypt/live/userapp.100xdevs.com/fullchain.pem; #  
ssl_certificate_key /etc/letsencrypt/live/userapp.100xdevs.com/privkey.pem  
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot  
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot  
  
}
```

- Install certbot and Refresh certificate

```
sudo certbot --nginx
```



Take home assignments

1. Get a DB on [neon.tech](#) / [RDS](#) / [Aeiven](#) and add a DB migration step to the DB
2. Pass in the DB credentials while starting the docker image
3. Start the docker image so that it restarts if it goes down (similar to pm2)