

Fullstack Development

Preflight project - deployment

Github Repo

Local machine

Clear your dev environment

- Remove all containers
 - Check with `docker ps -a`
 - `docker rm -f CONTAINER`
- Remove volumes
 - `docker volume prune -a`
- Remove all image cache
 - `docker image prune -a`
- Remove unused networks
 - `docker network prune`

Setup

- `git clone https://github.com/fullstack-68/pf-deploy.git`
 - *Better yet, fork and clone this repo*
- `cd pf-deploy`
- Make `.env` from `.env.example` (Make necessary changes.)
- Take care of `./_entrypoint/init.sh`
 - Windows: Make sure that you save with LF option.
 - Mac/Linux: `chmod +x ./_entrypoint/init.sh`
- `docker compose up -d --force-recreate`
- Use `docker compose logs` to inspect.

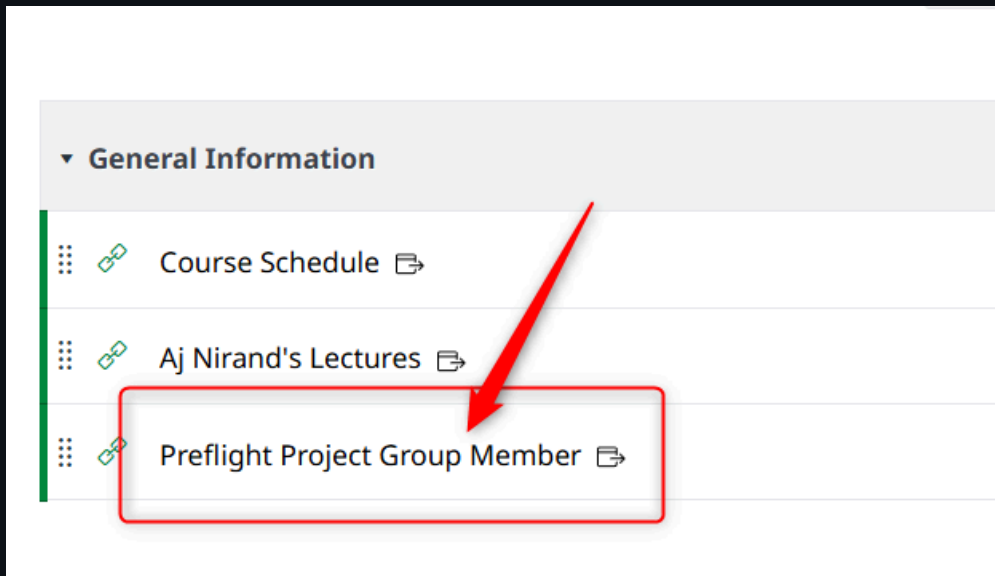
Remote server

Setup

- `ssh USERNAME@10.10.x.x`
- Clone the `pf-deploy` repo
 - **Make sure that the folder name is** `pf-deploy`.
- Repeat steps we just did on local machine
 - Use the assigned `FRONTEND_PORT` or else the public url will not work.
- Check your container
 - `docker ps | less -S`
- Visit your public `url`.

Setup

- Get server info from here.



How to cleanup

- If you have not deleted the folder.

- `docker compose exec -it backend sh`

- `cd logs`

- `rm *.*`

- `docker compose down`

- `docker volume prune -a`

- If you already delete the folder.

- `docker run -it --rm -v ./logs:/home/ubuntu ubuntu /bin/bash`

- `cd /home/ubuntu/logs`

- `rm *.*`

CI/CD

CI/CD

- Stands for *Continuous Integration and Continuous Delivery/Deployment*.
- Practice that aims to automate and streamline the process of `building`, `testing`, and `deploying` code.
- We will explore the automated *building and deploying* in this class.

Requirement

- Need a GitHub repository for your app. (Let's use the `pf-frontend` app.)
- To use my repo
 - `git clone https://github.com/fullstack-68/pf-frontend.git`
 - `remove-item .git, .github -Force`
 - `git init`
- Other commands
 - `git remote add origin [REPO_ADDR]`
 - `git remote set-url origin [REPO_ADDR]`
 - `git push --set-upstream origin [master/main]`

Setup

```
# Repository Secret
WEBHOOK_SECRET=
DISCORD_WEBHOOK=
DOCKERHUB_USERNAME=
DOCKERHUB_TOKEN=

# Repository Variable
PROJECT_GROUP=gxx
IMAGE_NAME=DOCKERHUB_ACCOUNT/preflight-frontend
```

Discord Webhook

Engagement

Boost Perks

EXPRESSION

Emoji

Stickers

Soundboard

PEOPLE

Members

Roles

Invites

Access

APPS

Integrations

App Directory



Integrations > Webhooks



ESC

Webhooks are a simple way to post messages from other apps and websites into Discord using internet magic.
[Learn more](#), or try [building one yourself](#).

New Webhook

Posting To #General



Spidey Bot


Created on Jun 30, 2025 by nnnpoo



Dockerhub Token

[Personal access tokens](#) / New access token

Create access token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#) 

Access token description

Github Action

Expiration date

None



Optional

Access permissions

Read & Write



Read & Write tokens allow you to push images to any repository managed by your account.

Cancel

Generate

Test `cpe_sever` Webhook

Send `POST` request

- <https://fs-webhook.iecmu.com/hooks/test>
- Body

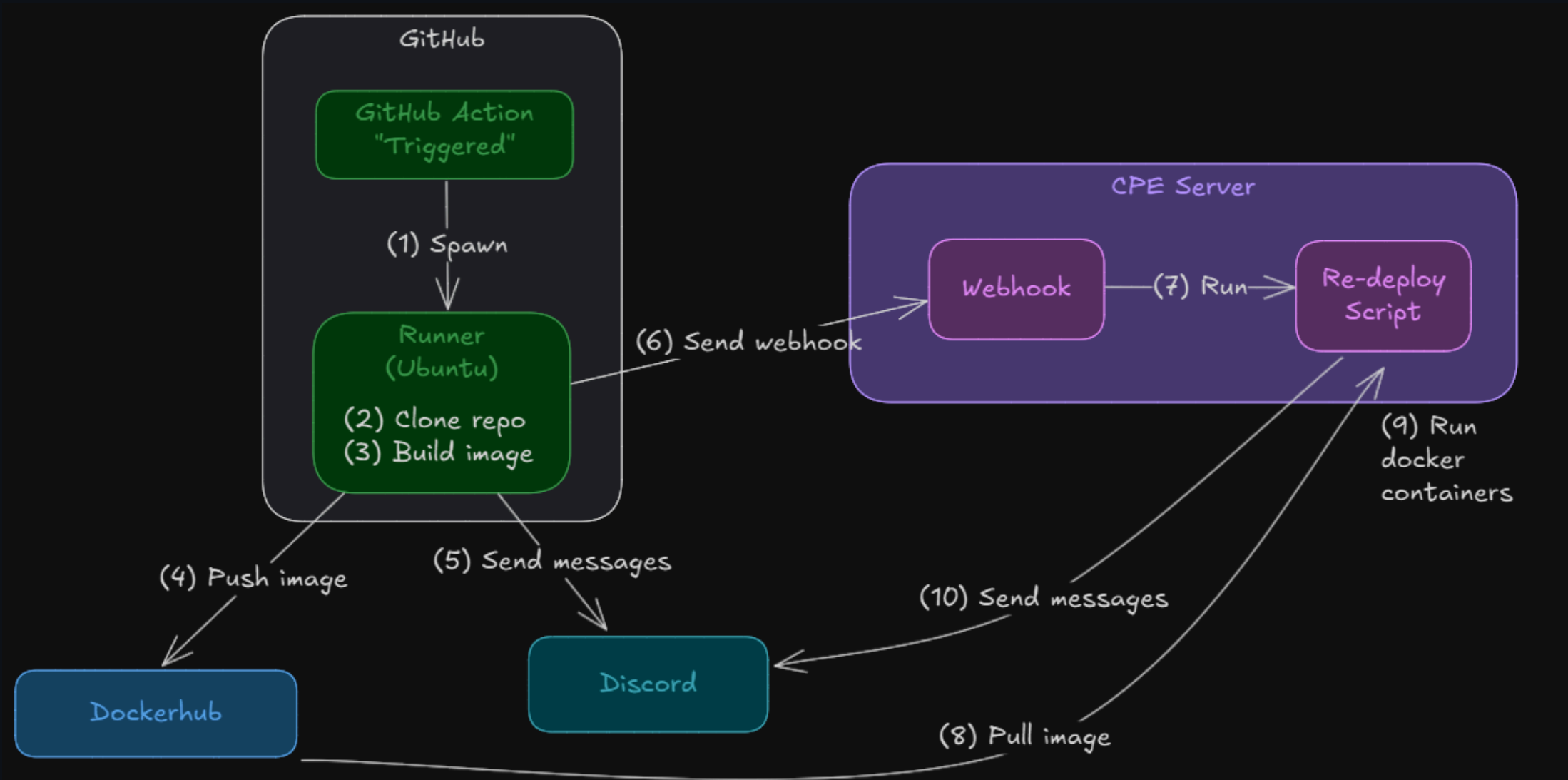
```
{
  "data": {
    "project_group": PROJECT_GROUP,
    "discord_webhook": DISCORD_WEBHOOK",
    "webhook_secret": WEBHOOK_SECRET
  }
}
```


Test GibHub Action

- Create secret and variables.
- Create test [workflow](#).
- Run test workflow.

Actual CI/CD

- Use this [workflow](#)



Behind the Server

- `Webhook` config
- `redploy` script

Recap

Topic	Stack
Language	TypeScript
DB	PostgreSQL / Drizzle ORM
Backend	Express
Frontend	Reac / Vite
Testing	Cypress
Deployment	Docker / Nginx
CI/CD	GiHub Action

Congratulations!

| Now go and make awesome apps!