

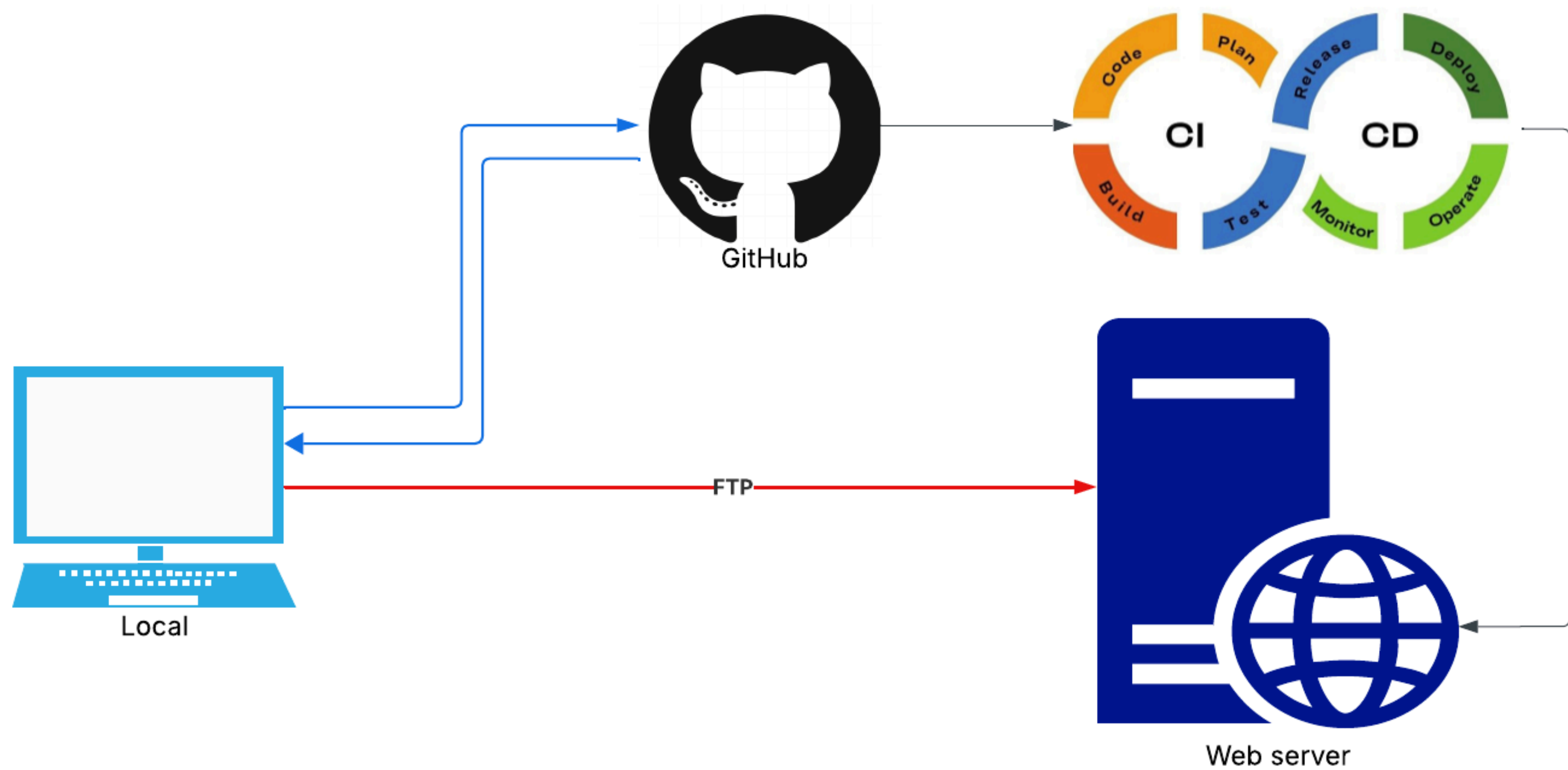


# CI/CD

Abd Mizwar A.Rahim, M.Kom



# CI/CD



# CI/CD

- **Continuous Integration (CI)**

Continuous Integration adalah proses otomatisasi pengujian kode setiap kali ada perubahan yang dilakukan oleh pengembang. Tujuannya adalah untuk mengidentifikasi dan mengatasi bug lebih awal dalam siklus pengembangan.

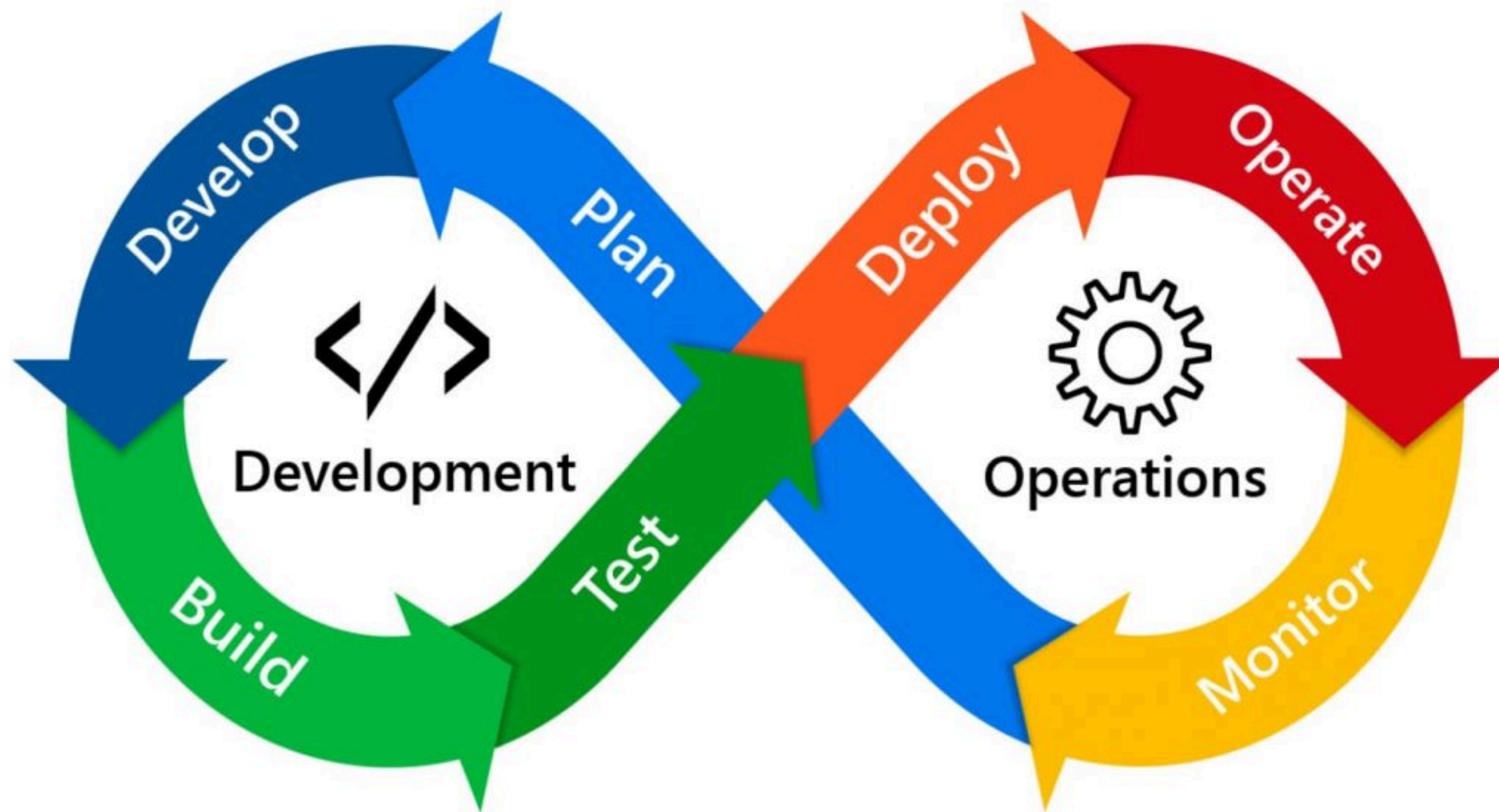
- **Continuous Delivery/Continuous Deployment (CD)**

Continuous Delivery dan Continuous Deployment adalah proses lanjutan dari CI. Keduanya melibatkan pengiriman kode ke lingkungan produksi (server )

# CI/CD

- **Continuous Delivery:** Setelah kode lulus semua tes otomatis, kode siap untuk dirilis ke produksi kapan saja. Namun, rilis ke produksi masih memerlukan persetujuan manual.
- **Continuous Deployment:** Setelah kode lulus semua tes otomatis, kode secara otomatis dirilis ke produksi tanpa memerlukan persetujuan manual.

# Cycle Ci/Cd



# CYCLE CI

- Plan (Perencanaan)

Mendefinisikan fitur baru, perbaikan bug, dan peningkatan lainnya berdasarkan kebutuhan bisnis dan feedback pengguna.

- Develop (Pengembangan)

Menulis kode dan unit test untuk fitur baru atau perbaikan. Menggunakan alat version control seperti Git untuk mengelola perubahan kode.

- Build (Pembangunan)

Menggabungkan kode dari berbagai pengembang dan membangun aplikasi. Ini termasuk mengkompilasi kode (jika diperlukan) dan mengelola dependensi.

- Test (Pengujian)

Menjalankan serangkaian tes otomatis seperti unit tests, integration tests, dan functional tests untuk memastikan bahwa kode berfungsi dengan benar.

# CYCLE CD

- **Deploy (Penerapan)**

Menerapkan aplikasi ke lingkungan staging dan akhirnya ke produksi. Proses ini bisa otomatis sepenuhnya (Continuous Deployment) atau memerlukan persetujuan manual (Continuous Delivery).

- **Operate (Operasi)**


Menjalankan dan memelihara aplikasi di lingkungan produksi. Ini termasuk manajemen server, database, jaringan, dan layanan lain yang diperlukan.




- **Monitor (Pemantauan)**






Memantau aplikasi untuk mendeteksi masalah, performa, dan kegunaan.



# CI ILUSTRASI

C--ci-cd / .github / workflows / c-cpp.yml 

 **abdulmizwar** Update c-cpp.yml  a6ce617 · 8 hours ago  History

**Code** Blame 26 lines (20 loc) · 608 Bytes  Code 55% faster with GitHub Copilot Raw    

```
1  name: C++ CI
2
3  on:
4    push:
5      branches: [ master ]
6    pull_request:
7      branches: [ master ]
8
9  jobs:
10   build:
11     runs-on: ubuntu-latest
12
13     steps:
14       - name: Checkout repository
15         uses: actions/checkout@v2
16
17       - name: Install dependencies
18         run: sudo apt-get install -y g++
19
20       - name: Compile C++ code
21         run: g++ -o main stack.cpp
22
23       - name: Run the program
24         run: ./main
25       # Note: If your program requires user input, you might want to handle this differently in a CI environment.
26       # For automated testing, consider using a testing framework or redirecting input from a file.
```

**Nama dari workflow CI ini**

**Workflow akan berjalan setiap kali ada push ke branch 'master'**  
**Workflow juga akan berjalan setiap kali ada pull request yang ditujukan ke branch 'master'**

**Workflow juga akan berjalan setiap kali ada pull request yang ditujukan ke branch 'master'**

**Penginstalan depedensi : menginstal gcc melalui Homebrew, paket manager untuk macOS**

**Menjalankan perintah g++ untuk mengkompilasi file stack.cpp dan menghasilkan executable bernama 'main'**

**Menjalankan program yang sudah dikompilasi  
Menjalankan executable 'main'**



# CD ILUSTRASI

## Skema Dari CD

- Done Skema Ci
- Ketika Code Merge
- Trigger Ssh action
- Connect Server
- Pull Codingan terbaru
- Restart

```
name: C++ CD

on:
  push:
    branches:
      - master

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build Docker image
        run: docker build -t stack_program .

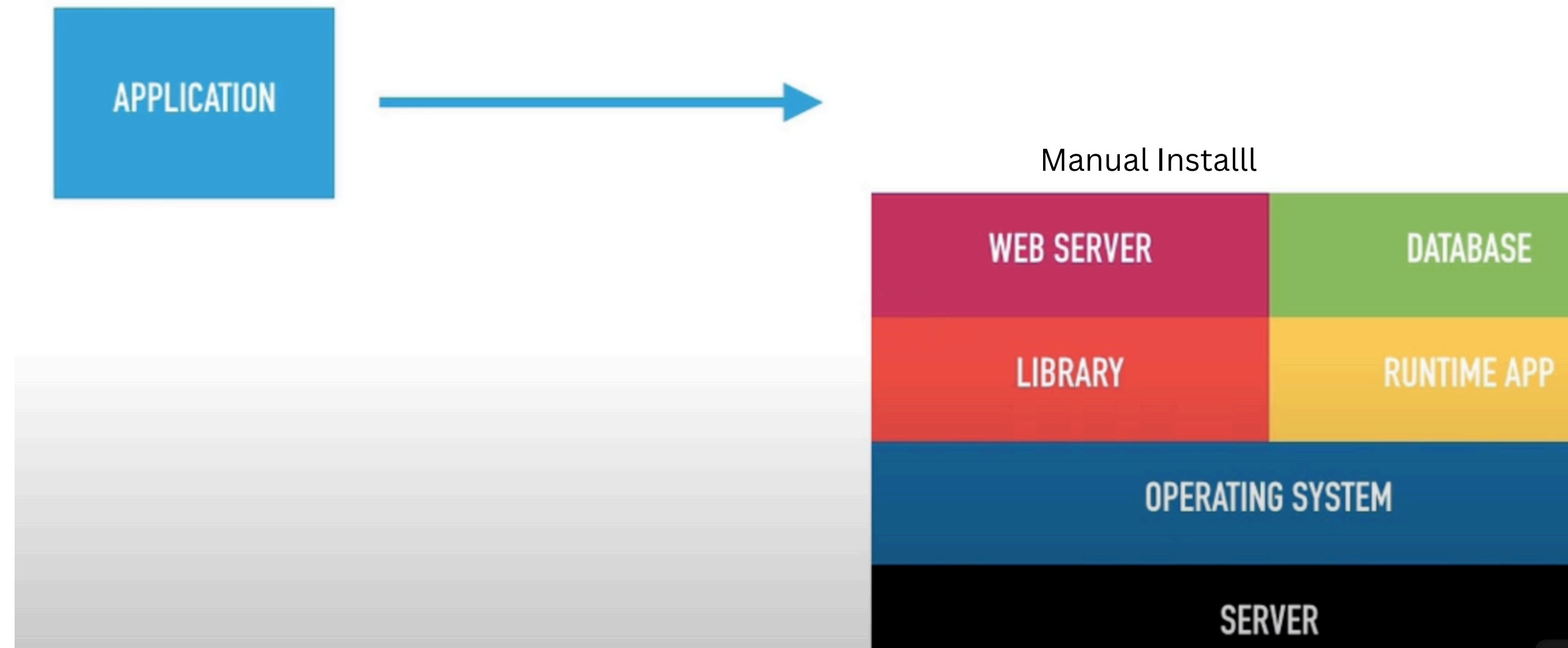
      - name: Log in to Heroku Container Registry
        env:
          HEROKU_API_KEY: ${ secrets.HEROKU_API_KEY }
        run: |
          echo "$HEROKU_API_KEY" | docker login --username=_ --password-stdin registry.heroku.com

      - name: Push Docker image to Heroku
        run: |
          docker tag stack_program registry.heroku.com/<HEROKU_APP_NAME>/web
          docker push registry.heroku.com/<HEROKU_APP_NAME>/web

      - name: Release Heroku app
        env:
          HEROKU_API_KEY: ${ secrets.HEROKU_API_KEY }
        run: |
          heroku container:release web --app <HEROKU_APP_NAME>
```

# CD ILUSTRASI

## DEPLOYING APPLICATION



### Proyek PHP

- SERVER

Menyewa server (misalnya: VPS dari DigitalOcean, atau shared hosting dari Niagahoster).

- OPERATING SYSTEM

Biasanya menggunakan Ubuntu atau CentOS.

- RUNTIME APP

PHP engine (misalnya PHP 8.1)

- LIBRARY

Jika menggunakan Composer

- WEB SERVER

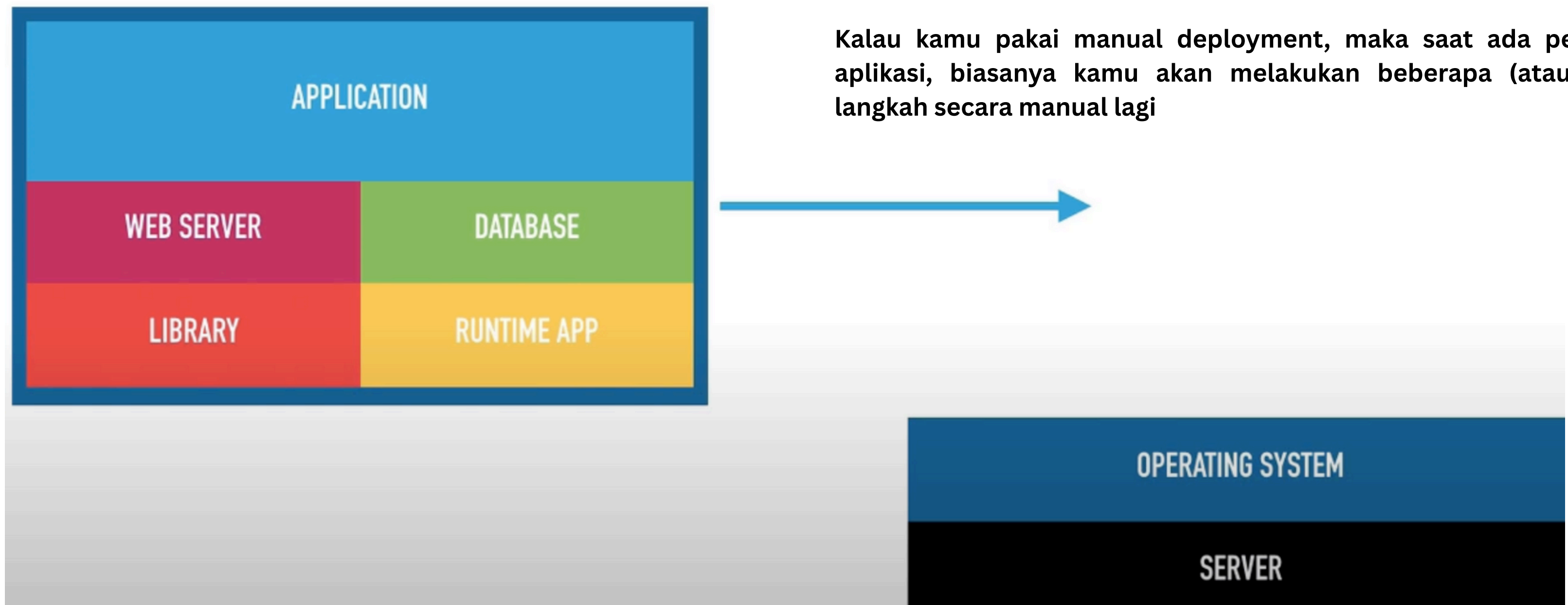
Instal Apache atau Nginx

- DATABASE

Instal dan setup MySQL

# CD ILUSTRASI

## DEPLOYING APPLICATION WITH DOCKER



# TASK

## PROBLEM

- Konflik pada Git : Anggota tim mengalami kegagalan saat melakukan push ke branch utama (main branch) karena adanya konflik dengan perubahan yang telah diterima sebelumnya. Konflik ini terjadi ketika dua pengembang atau lebih mengubah bagian yang sama dari file yang sama dalam branch yang berbeda.
- Pipeline CI Rusak : Selain itu, pipeline CI mengalami kegagalan karena file konfigurasi YAML rusak. File YAML adalah konfigurasi yang mengatur alur dan langkah-langkah dalam pipeline (misalnya, build, dan test) sehingga kerusakan pada file ini dapat menyebabkan kegagalan seluruh pipeline.

- Buatkan Repo di github (Nama Repo = CI-CD)
- Lakukan push file proyek anda (**proyek nya terserah**) dari repo local ke repo github (**menggunakan git bash**)
- Tambahkan Pipeline Continuous Integration (CI)

### Skema CI nya :

- Pipeline ini otomatis berjalan saat event push/pull dilakukan
- Mengambil seluruh isi repository (source code) ke dalam runner GitHub agar bisa diproses.
- Install Dependencies
- Compile Code
- Run the Program
- **implementasi** kedua Problem tersebut, dan lakukan **penanganannya**.

**THANK YOU**

