Homework Sesi 22 Bootcamp DevOps Engineer Digital Skola

Buatlah sebuah pipeline CI/CD menggunakan GitLab untuk sebuah aplikasi web sederhana, katakanlah nginx.

Pipeline ini harus mencakup langkah-langkah berikut:

1. Build aplikasi untuk memastikan bahwa kode dapat dibangun tanpa error.

2. Deploy aplikasi ke server

Berikut adalah pembuatan pipeline Continuous Integration/Continuous Deployment (CI/CD) di GitLab yang memanfaatkan Docker Hub. Pipeline ini akan secara otomatis:

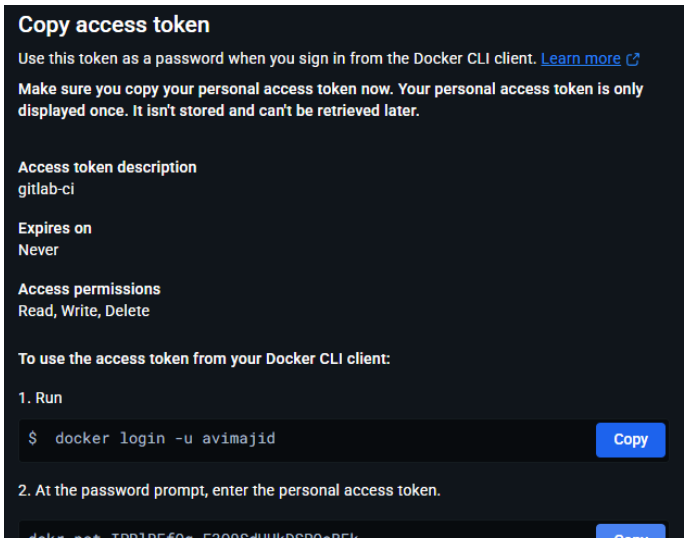
1. Membangun (build) image Docker aplikasi Nginx dan mendorong (push) image tersebut ke Docker Hub.
2. Menyebarkan (deploy) aplikasi dengan menarik (pull) image dari Docker Hub dan menjalankannya di server tujuan.

Prasyarat

Sebelum memulai, pastikan hal-hal berikut telah disiapkan:

1. **Akun GitLab:** Instance GitLab self-hosted.
2. **Repositori GitLab:** Sebuah proyek (repositori) baru di GitLab bernama **my-awesome-project**.
3. **Docker:** Terpasang di server tujuan deployment.
4. **Server Tujuan (Deployment Target):** VM di GCP (bernama **server1**)
5. **GitLab Runner:** Sebuah GitLab Runner yang aktif, terkonfigurasi untuk proyek, dan memiliki kemampuan menjalankan Docker (disarankan menggunakan Docker executor atau Shell executor dengan Docker terpasang) serta melakukan SSH. Runner perlu ditandai (tagged).
6. **Akun Docker Hub:** Diperlukan akun di hub.docker.com.
7. **Repositori Docker Hub:** Buat sebuah repositori baru di akun Docker Hub (avimajid/aplikasi-nginx-simple).
8. **Docker Hub Access Token:** Buat *Access Token* di Docker Hub untuk login dari CI/CD. Ini lebih aman daripada menggunakan kata sandi akun.
    - Login ke Docker Hub.
    - Buka Account Settings -> Personal access token.

- Klik New Access Token, beri nama (misal: gitlab-ci), pilih izin (Read, Write, Delete direkomendasikan), dan klik Generate.



Langkah 1: Penyiapan Aplikasi Sederhana

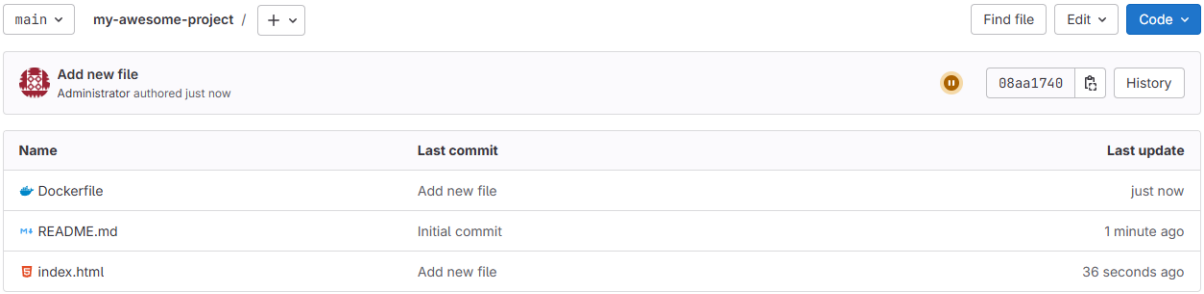
1. File HTML sederhana sebagai konten web bernama index.html dengan isi berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>Selamat Datang di Nginx!</title>
</head>
<body>
    <h1>Halo dari Nginx via GitLab CI/CD!</h1>
    <p>Aplikasi ini dideploy secara otomatis.</p>
</body>
</html>
```

2. File bernama Dockerfile (tanpa ekstensi) untuk mendefinisikan bagaimana image Docker aplikasi akan dibangun:

```
# Gunakan base image Nginx resmi yang ringan
FROM nginx:alpine

# Salin file index.html ke direktori default Nginx
COPY index.html /usr/share/nginx/html/index.html

# Ekspos port 80
EXPOSE 80

# Perintah default untuk menjalankan Nginx saat container dimulai
CMD ["nginx", "-g", "daemon off;"]
```

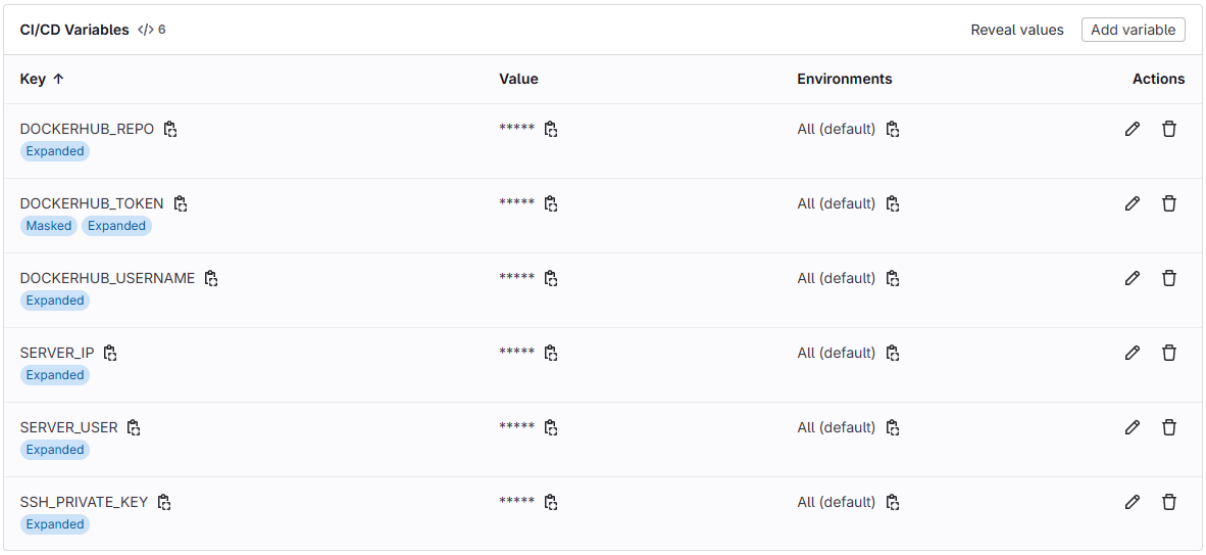
Kedua file tersebut dimasukkan ke dalam repositori di Gitlab.



Langkah 2: Konfigurasi Variabel CI/CD di GitLab

Tambahkan kredensial dan konfigurasi yang diperlukan sebagai variabel rahasia di GitLab:

1. Buka proyek GitLab -> Settings -> CI/CD.
2. Expand bagian Variables.
3. Klik Add variable dan tambahkan variabel berikut:
    - **DOCKERHUB_USERNAME**: Username akun Docker Hub.
    - **DOCKERHUB_TOKEN**: *Access Token* yang dibuat di Docker Hub tadi.
    - **DOCKERHUB_REPO**: Nama repositori lengkap di Docker Hub (avimajid/aplikasi-nginx-simple).
    - **SERVER_USER**: Username SSH untuk server tujuan.
    - **SERVER_IP**: Alamat IP atau domain server tujuan.
    - **SSH_PRIVATE_KEY**: Kunci privat SSH untuk login ke server tujuan.



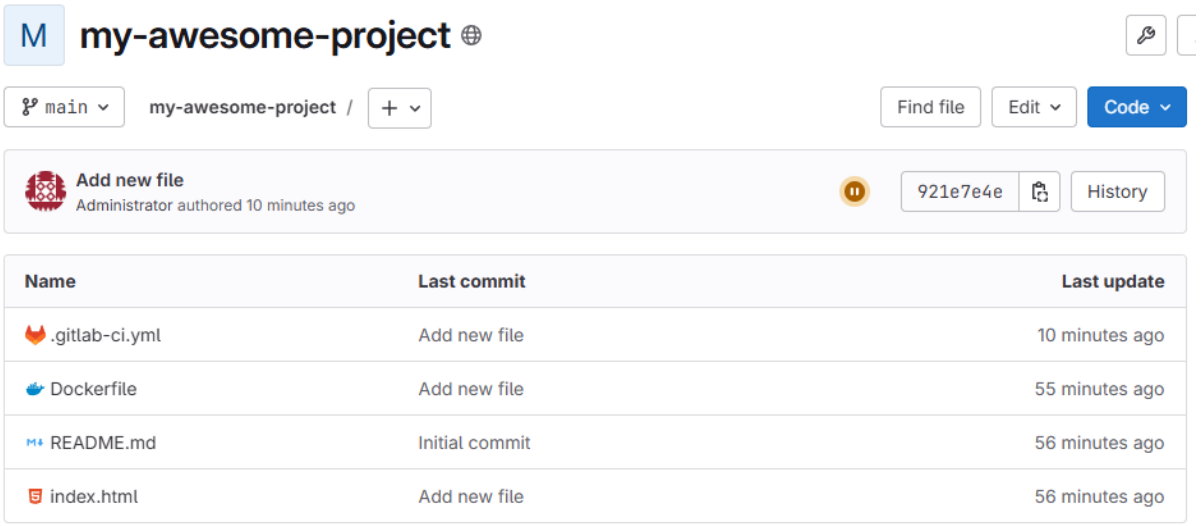
Langkah 3: Membuat File Konfigurasi Pipeline (.gitlab-ci.yml)

Buat file .gitlab-ci.yml di root proyek dengan isi berikut:

```
# Mendefinisikan tahapan pipeline
stages:
  - build
  - deploy

# Variabel global untuk nama image agar konsisten
variables:
  # Menggunakan variabel DOCKERHUB_REPO yang sudah diset di GitLab CI/CD
Variables
  # Menambahkan tag unik berdasarkan ID commit singkat
  IMAGE_TAG: $CI_COMMIT_SHORT_SHA
  FULL_IMAGE_NAME: $DOCKERHUB_REPO:$IMAGE_TAG

# Job 1: Build dan Push Docker Image ke Docker Hub
build_and_push_to_hub:
  stage: build
  image: docker:stable
  services:
    - docker:stable-dind # Menjalankan Docker-in-Docker service
  before_script:
    # Login ke Docker Hub menggunakan username dan token dari variabel CI/CD
    - echo "Logging in to Docker Hub..."
    # - docker login -u "$DOCKERHUB_USERNAME" -p "$DOCKERHUB_TOKEN"
    - echo "$DOCKERHUB_TOKEN" | docker login -u "$DOCKERHUB_USERNAME" --
password-stdin
    - docker info
  script:
    - echo "Building Docker image $FULL_IMAGE_NAME"
    # Build image dengan tag yang telah ditentukan
    - docker build -t $FULL_IMAGE_NAME .
    - echo "Pushing image to Docker Hub..."
    # Push image ke Docker Hub
    - docker push $FULL_IMAGE_NAME
    - echo "Build and push complete."
  tags:
    - docker

# Job 2: Deploy ke server target (VM GCP)
deploy_to_gcp:
  stage: deploy
  image: alpine:latest
  before_script:
    # Install SSH client dan tools yang diperlukan
    - apk add --no-cache openssh-client bash
    # Setup SSH credentials
```

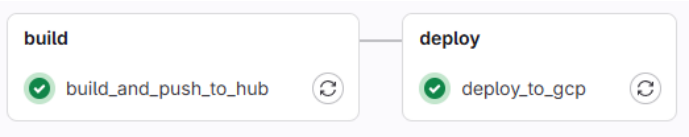
```
    - mkdir -p ~/.ssh
    # - cat "$SSH_PRIVATE_KEY"
    - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
    - chmod 600 ~/.ssh/id_rsa
    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
  script:
    - echo "Deploying image $FULL_IMAGE_NAME to server $SERVER_IP..."
    # Perintah SSH untuk dijalankan di server tujuan
    # - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker pull
$DOCKER_HUB_USERNAME/$DOCKER_IMAGE"
    - |
      ssh -i ~/.ssh/id_rsa $SERVER_USER@$SERVER_IP "
        # echo 'Logging in to Docker Hub on server (if needed for private
repo)...';
        # Jika repositori Docker Hub bersifat PRIVATE, login diperlukan di
server.
        # Jika PUBLIC, baris docker login di bawah ini TIDAK diperlukan.
        # docker login -u \"$DOCKERHUB_USERNAME\" -p \"$DOCKERHUB_TOKEN\";

        echo 'Pulling latest image from Docker Hub...';
        # Tarik image spesifik yang baru saja di-push dari Docker Hub
        docker pull $FULL_IMAGE_NAME;

        echo 'Stopping and removing old container...';
        # Hentikan dan hapus container lama jika ada
        docker stop my-simple-nginx-hub || true;
        docker rm my-simple-nginx-hub || true;

        echo 'Running new container...';
        # Jalankan container baru dari image yang ditarik dari Docker Hub
        # Memetakan port 80 container ke port 8080 host server
        docker run -d --name my-simple-nginx-hub -p 8080:80
$FULL_IMAGE_NAME;

        echo 'Deployment complete. Aplikasi berjalan di
http://$SERVER_IP:8080';
      "
  environment: # Mendefinisikan lingkungan deployment (opsional)
    name: production-hub
    url: http://$SERVER_IP:8080
  rules:
    # Hanya jalankan job deploy ini jika ada commit ke branch 'main'
    - if: '$CI_COMMIT_BRANCH == "main"'
  tags:
    - docker
```

Sekarang isi repo sudah lengkap:



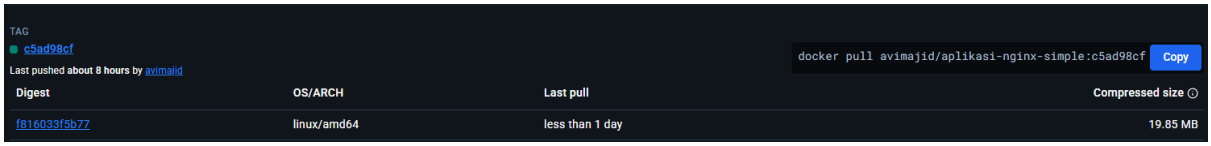
Langkah 4: Memicu Pipeline

Buka **Build > Pipelines** di GitLab. Pipeline baru akan berjalan. Amati tahap build (mencakup build image dan push ke Docker Hub) dan tahap deploy (menarik image dari Docker Hub).



Langkah 5: Verifikasi Deployment

1. **Periksa Docker Hub:** Setelah tahap build berhasil, buka repositori di Docker Hub. Seharusnya terlihat tag image baru yang sesuai dengan ID commit singkat dari GitLab.



2. **Periksa Aplikasi:** Setelah tahap deploy berhasil, akses http://<SERVER_IP>:8080 di browser. Aplikasi Nginx seharusnya berjalan, disajikan dari container yang imagenya ditarik dari Docker Hub.

Temuan kendala:

Sempat menemukan error yang mana untuk mencari solusinya cukup sulit untuk ditemukan. Setelah mencari dari forum-forum ternyata error ini terjadi dikarenakan Gitlab Runner gagal connect ke docker daemon untuk menjalankan image dind (docker in docker). Solusinya adalah mengatur secara manual configurasi gitlab runner yang mengerjakan CICD.

```
ERROR: Cannot connect to the Docker daemon at tcp://docker:2375. Is the docker daemon running?
errors pretty printing info
Cleaning up project directory and file based variables
ERROR: Job failed: exit code 1
```

Buka file konfigurasi gitlab runner dengan editor **nano /etc/gitlab-runner/config.toml** lalu atur menjadi:

- image = "docker:stable"
- privileged = true
- volumes = ["/var/run/docker.sock:/var/run/docker.sock", "/cache"]

Dengan ini gitlab-runner akan berjalan dengan normal.

```
root@gitlab:~# nano /etc/gitlab-runner/config.toml
```

```
[[runners]]
  name = "docker-runner-nih"
  url = "http://[redacted]9"
  id = 39
  token = "[redacted]Koa"
  token_obtained_at = 2025-04-09T13:24:48Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "docker"
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "docker:stable"
    privileged = true
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/var/run/docker.sock:/var/run/docker.sock", "/cache"]
    shm_size = 0
    network_mtu = 0
```