Modul Praktikum 2

Version Control dengan Git

1. Maksud

Modul ini dirancang untuk memberikan pemahaman mendalam tentang konsep Version Control System (VCS) dan penggunaan Git sebagai alat utama dalam pengembangan perangkat lunak. Modul ini juga bertujuan untuk memperkenalkan praktik-praktik dasar Git yang esensial dalam alur kerja DevOps.

2. Tujuan

a) Memahami konsep dasar Version Control System (VCS) dan pentingnya dalam DevOps.

b) Menguasai perintah-perintah dasar Git seperti clone, commit, push, pull, branch, dan merge.

c) Mampu menerapkan Git dalam proyek pengembangan perangkat lunak secara kolaboratif.

d) Meningkatkan efisiensi dan produktivitas dalam pengembangan perangkat lunak dengan menggunakan Git.

3. Dasar Teori

3.1 Konsep Version Control dan Pentingnya dalam DevOps

Version Control System (VCS) adalah sistem yang mencatat perubahan pada file atau sekumpulan file dari waktu ke waktu sehingga Anda dapat mengingat versi tertentu di masa mendatang. Dalam konteks pengembangan perangkat lunak, VCS memungkinkan tim untuk melacak perubahan kode sumber, berkolaborasi secara efektif, dan mengelola berbagai versi dari proyek yang sama.

DevOps adalah budaya dan praktik yang bertujuan untuk menyatukan pengembangan (Development) dan operasi (Operations) untuk meningkatkan kolaborasi dan produktivitas. Version Control, khususnya Git, memainkan peran penting dalam DevOps karena:

a) **Kolaborasi**: Memungkinkan banyak developer bekerja pada proyek yang sama tanpa mengganggu satu sama lain.

b) **Pelacakan Perubahan**: Setiap perubahan dicatat, sehingga mudah untuk melacak bug atau memahami evolusi kode.

c) **Integrasi Berkelanjutan**: Memfasilitasi integrasi kode yang sering dan otomatis, yang merupakan inti dari DevOps.

d) **Rollback**: Memungkinkan pengembalian ke versi sebelumnya jika terjadi kesalahan.

3.2 Perintah Dasar Git

a) **Clone**: Mengunduh salinan repositori Git dari remote repository ke lokal.

git clone <repository_url>

b) **Commit**: Menyimpan perubahan yang dilakukan pada file ke dalam repositori lokal.

git commit -m "Pesan commit"

c) **Push**: Mengunggah perubahan dari repositori lokal ke remote repository.

git push origin <branch_name>

d) **Pull**: Mengambil perubahan dari remote repository ke repositori lokal.

git pull origin <branch_name>

e) **Branch**: Membuat cabang baru dari repositori untuk pengembangan fitur atau perbaikan bug.

git branch <branch_name>

f) **Merge**: Menggabungkan perubahan dari satu cabang ke cabang lain.

git merge <branch_name>

4. Prosedur Praktik

4.1 Persiapan

a) **Instalasi Git**: Pastikan Git sudah terinstal di sistem Anda. Jika belum, unduh dan instal dari git-scm.com.

b) **Konfigurasi Git**: Setel nama dan email Anda di Git.

git config --global user.name "Nama Anda"

git config --global user.email "email@anda.com"

4.2 Langkah-langkah Praktik

1. **Clone Repositori**:
    a) Buat repositori baru di GitHub atau platform Git lainnya.
    b) Clone repositori tersebut ke lokal.
    git clone <repository_url>
2. **Buat dan Commit Perubahan**:
    a) Buat file baru atau ubah file yang sudah ada.
    b) Tambahkan perubahan ke staging area.
    git add <file_name>
    c) Commit perubahan.
    git commit -m "Pesan commit"
3. **Push Perubahan**:
    a) Unggah perubahan ke remote repository.
    git push origin <branch_name>
4. **Buat dan Switch Branch**:

a) Buat branch baru.

git branch <branch_name>

b) Switch ke branch tersebut.

git checkout <branch_name>

5. **Merge Branch**:

a) Kembali ke branch utama (misalnya, main atau master).

git checkout main

b) Gabungkan branch yang baru dibuat ke branch utama.

git merge <branch_name>

5. Latihan

Berikut adalah contoh studi kasus konkret beserta penyelesaiannya untuk membantu Anda memahami penggunaan Git dalam skenario pengembangan perangkat lunak.

Studi Kasus: Pengembangan Fitur Login pada Aplikasi Web

Deskripsi Kasus

Anda adalah bagian dari tim pengembang yang sedang mengerjakan aplikasi web. Tim memutuskan untuk mengimplementasikan fitur login. Fitur ini akan dikembangkan secara terpisah dari kode utama agar tidak mengganggu pengembangan fitur lainnya. Setelah selesai, fitur login akan digabungkan ke dalam kode utama.

Langkah Penyelesaian

Langkah 1: Clone Repositori

1. Clone repositori proyek dari GitHub ke lokal Anda.

   git clone https://github.com/username/nama-repositori.git

2. Masuk ke direktori repositori.

   cd nama-repositori

Langkah 2: Buat Branch Baru untuk Fitur Login

1. Buat branch baru bernama feature-login.

   git branch feature-login

2. Pindah ke branch feature-login.

   git checkout feature-login

Langkah 3: Implementasi Fitur Login

1. Buat file baru bernama *login.html* dan tambahkan kode berikut:

   <!-- login.html -->
   <!DOCTYPE html>

```
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h1>Login Page</h1>
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password"><br><br>
    <button type="submit">Login</button>
  </form>
</body>
</html>
```

2. Tambahkan file login.html ke staging area.

   git add login.html

3. Commit perubahan dengan pesan yang deskriptif.

   git commit -m "Menambahkan halaman login"

Langkah 4: Push Branch ke Remote Repository

1. Push branch feature-login ke remote repository.

   git push origin feature-login

Langkah 5: Merge Fitur Login ke Branch Utama

1. Kembali ke branch utama (main atau master).

   git checkout main

2. Gabungkan branch feature-login ke branch utama.

   git merge feature-login

3. Push perubahan yang telah digabungkan ke remote repository.

   git push origin main

Studi Kasus Tambahan: Menangani Konflik Merge

Deskripsi Kasus

Selama pengembangan fitur login, salah satu anggota tim mengubah file *index.html* di branch utama (main). Ketika Anda mencoba menggabungkan branch *feature-login* ke main, terjadi konflik karena kedua branch mengubah file yang sama.

Langkah Penyelesaian

Langkah 1: Identifikasi Konflik

1. Saat menjalankan perintah git merge feature-login, Git akan memberi tahu adanya konflik.

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.

Langkah 2: Perbaiki Konflik

1. Buka file index.html yang bermasalah. Anda akan melihat tanda konflik seperti ini:

   <<<<<<< HEAD

   * Selamat datang di halaman utama! *

   =======

   * Silakan login untuk melanjutkan. *

   >>>>>>> feature-login

2. Perbaiki konflik dengan memilih perubahan yang diinginkan atau menggabungkan keduanya. Misalnya:

   * Selamat datang di halaman utama! Silakan login untuk melanjutkan. *

Langkah 3: Commit Perbaikan Konflik

1. Tambahkan file yang telah diperbaiki ke staging area.

   git add index.html

2. Commit perubahan.

   git commit -m "Memperbaiki konflik merge di index.html"

Langkah 4: Selesaikan Merge

1. Push perubahan yang telah digabungkan ke remote repository.

   git push origin main

6. Tugas

1. **Tugas 1**: Buat repositori baru di GitHub, clone ke lokal, dan buat setidaknya 3 commit dengan pesan yang deskriptif.
2. **Tugas 2**: Buat dua branch berbeda, lakukan perubahan pada masing-masing branch, lalu merge kedua branch tersebut ke branch utama.
3. **Tugas 3**: Jelaskan dalam bentuk laporan singkat bagaimana Git membantu dalam kolaborasi tim dan integrasi berkelanjutan dalam DevOps.