Acara 16

Pokok Bahasan : Github Workflow

Acara Praktikum/Praktik : Minggu 14/1-4

Tempat : Politeknik Negeri Jember

Alokasi Waktu : 100 menit

a. Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa mampu melakukan alur kerja GIT (Git Workflows)

b. Indikator

Mahasiswa mampu mempraktikan pengelolaan git secara efisien.

c. Dasar Teori

Git adalah sistem kontrol versi terdistribusi untuk melacak perubahan kode sumber selama pengembangan perangkat lunak. Git dirancang untuk mengoordinasikan pekerjaan di antara pemrogram, tetapi dapat digunakan untuk melacak perubahan dalam kumpulan file apa pun. Sasarannya mencakup kecepatan, integritas data, dan dukungan untuk alur kerja non-linier terdistribusi.

Git membantu organisasi membantu memudahkan proses pengembangan dan mengelola repositori. Ini membantu meningkatkan kinerja dan mengubah cara tim pengembangan Anda membuat perangkat lunak. Katakanlah jika perusahaan Anda bergantung pada aplikasi perangkat lunak penting yang mengubah aliran pengembangan Anda berdampak pada seluruh bisnis Anda.

Parralel Developing

Git Workflow membuat pengembangan paralel menjadi sangat mudah. Ini membantu mengisolasi pengembangan dari pekerjaan yang sudah selesai. Ini memberikan kebebasan bagi pengembang untuk melanjutkan pengembangan atau perbaikan baru pada cabang fitur dan bergabung ke badan kode utama ketika pengembang puas dengan kode tersebut.

Git Workflow

Menggunakan alur kerja git flow yang efisien akan membantu anggota tim untuk mengerjakan berbagai fitur secara paralel. Ini akan membantu menyelesaikan konflik penggabungan saat menggabungkan fitur yang sedang dikerjakan anggota Anda dalam sprint.

d. Alat dan Bahan

1. Laptop dan ponsel
2. Internet
3. Browser

e. Prosedur Kerja

Membuat Branch

1. Pertama-tama Anda mengkloning repositori Anda dengan menggunakan HTTPs atau SSH

```
git clone https://github.com/anjulapaulus/sample-project.git
```

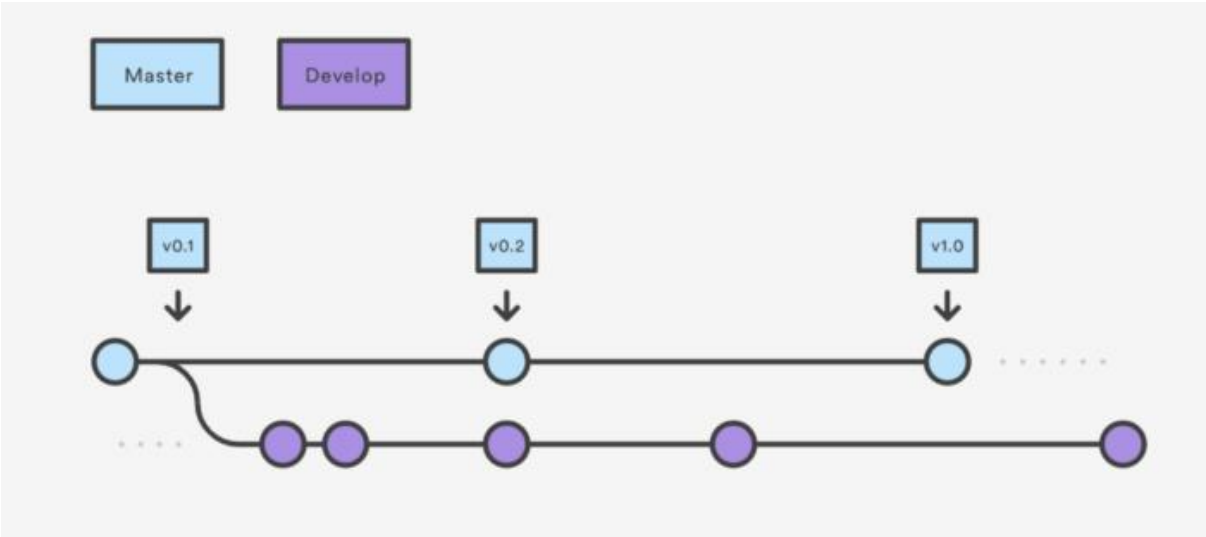
2. Buat cabang pengembangan untuk implementasi awal Anda

```
git checkout -b kembangkan
```

3. Buat perubahan awal Anda dan kode didorong ke cabang pengembangan dan digabungkan ke cabang master.

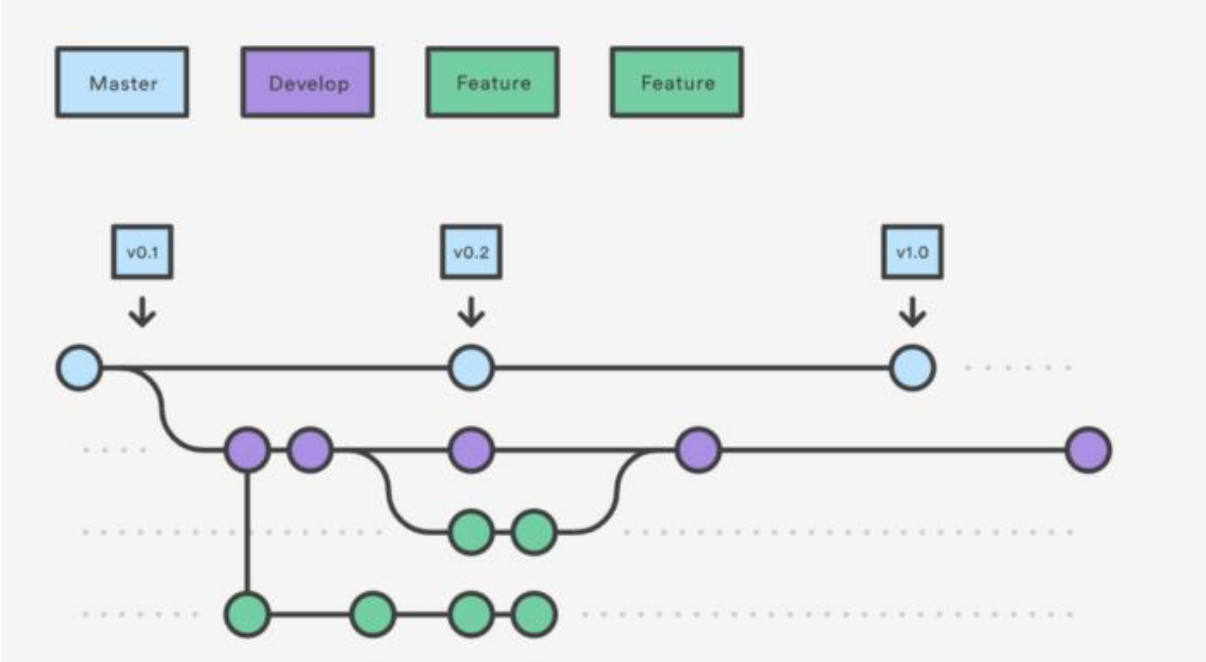
```
git Push -u asal kembangkan
```

4. Jadi cabang pengembangan akan menyimpan riwayat proyek dan menguasai versi singkatnya.



Menggabungkan Branch

5. Setiap fitur harus memiliki cabangnya sendiri. Cabang ini harus bercabang dari cabang develop dan digabung menjadi develop setelah fitur diimplementasikan.



6. Jadi mari kita lihat cara membuat cabang fitur.

```
git checkout kembangkan
git checkout -b fitur/nama
```

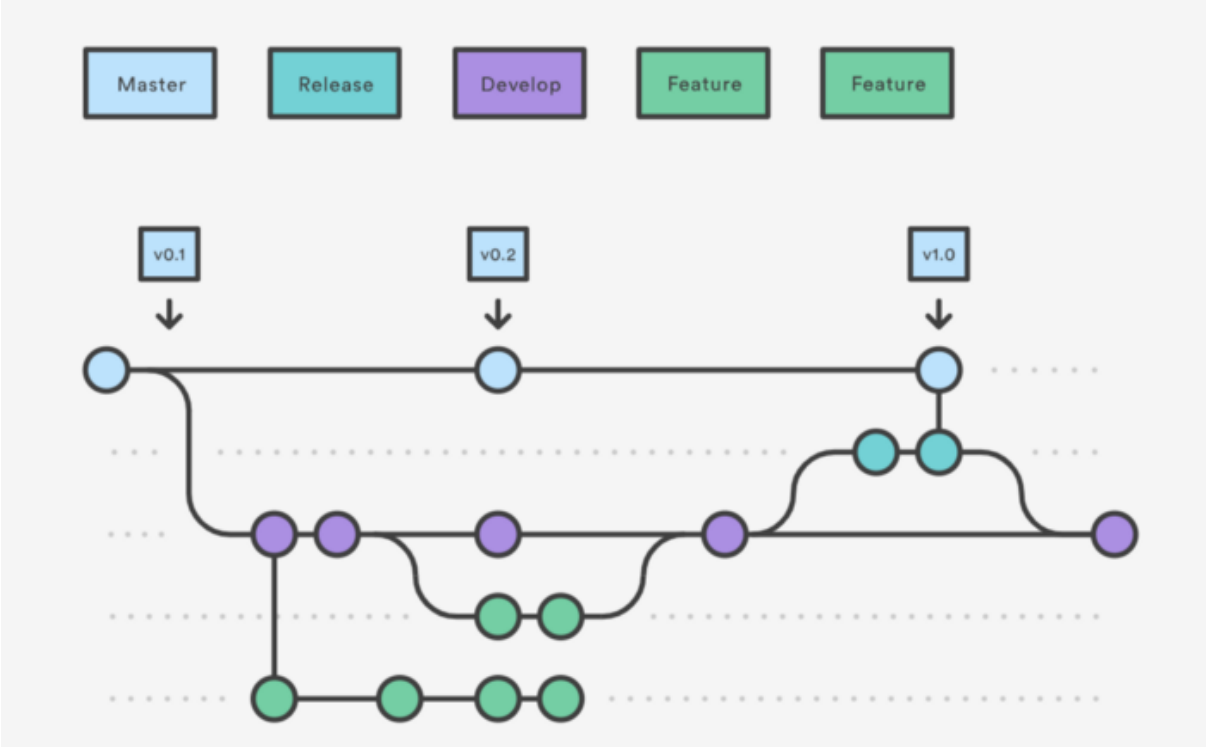
7. Berikan nama yang bermakna untuk cabang fitur Anda sehingga terlihat fitur apa yang diterapkan di cabang fitur yang relevan.
8. Setelah semua pekerjaan selesai pada cabang fitur, kami menggabungkannya untuk dikembangkan.

```
git checkout kembangkan
fitur/nama git merge
```

9. Setelah ini cabang develop dapat digabungkan menjadi cabang master.

Release Branch

10. Setelah fitur siap untuk dirilis, kami dapat melakukan fork cabang rilis dari pengembangan. Membuat cabang ini memulai siklus rilis berikutnya, fitur baru tidak dapat ditambahkan ke rilis berikut selain perbaikan bug atau perubahan dalam dokumentasi. Setelah kami siap untuk menyebarkan, kami menggabungkan cabang rilis untuk menguasai dan menandai versi rilis menggunakan versi semantik. Gunakan cabang khusus untuk setiap rilis.



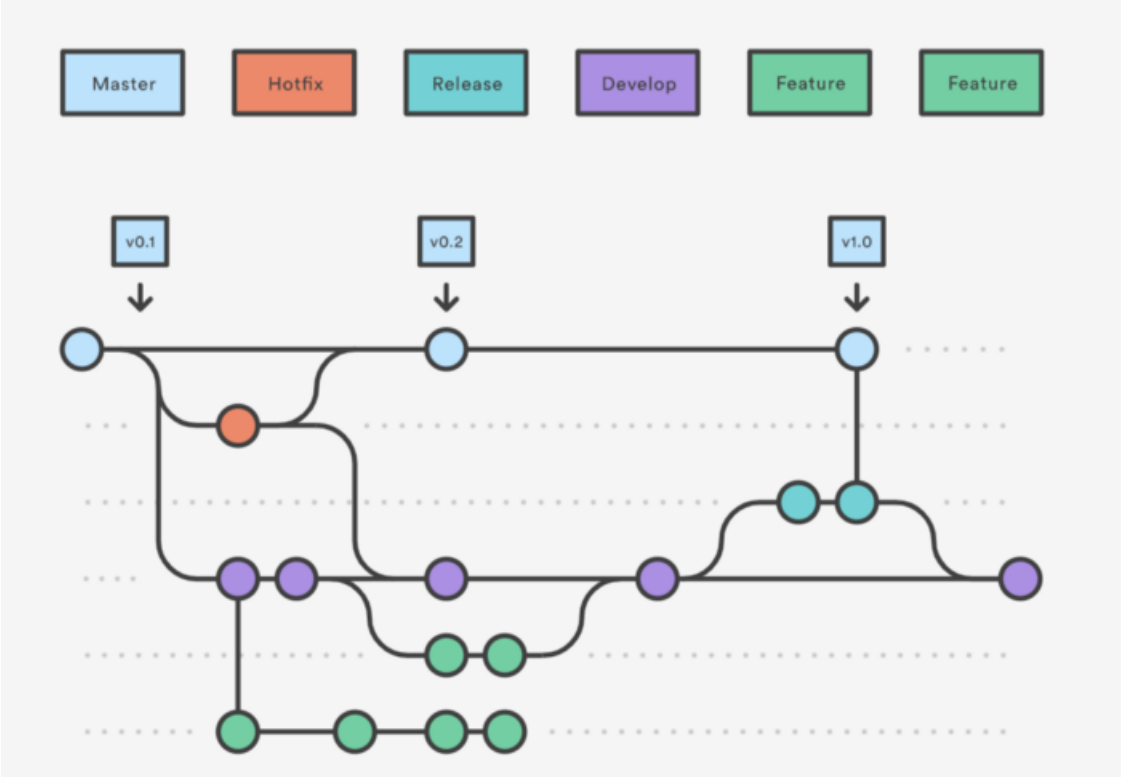
11. Mari kita lihat bagaimana kita bisa melakukannya.

```
git checkout kembangkan
git checkout -b release/v1.0.0
```

12. Sekarang gabungkan cabang rilis Anda untuk menguasai dan mengembangkan cabang dan cabang rilis dapat dihapus.

Hotfix Branch

13. Cabang hotfix dibuat untuk perbaikan bug segera untuk menambal rilis produksi. Tidak seperti cabang rilis atau cabang fitur yang bercabang dari pengembangan, cabang-cabang ini bercabang dari master. Setelah perbaikan selesai, cabang ini harus digabungkan menjadi master dan pengembangan atau cabang rilis yang relevan dan master harus ditandai dengan nomor versi yang diperbarui.



14. Mari kita lihat cara membuat cabang hotfix.

```
git checkout master
git checkout -b hotfix/nama
```

Contoh Alur Kerja

Fitur Aliran Cabang

Dengan asumsi bahwa repositori baru dibuat.

```
git checkout master
git checkout -b develop
git checkout -b fitur/nama-fitur#Setelah pengembangan fitur selesai
git checkout kembangkan
git merge fitur/nama-fitur
git checkout master
git merge fitur/nama-fitur
git branch -d fitur/nama-fitur
```

Release Aliran Cabang

```
git checkout kembangkan
git checkout -b rilis/versigit checkout kembangkan
git merge rilis/versi
git checkout master
rilis git merge/versi
git branch -d rilis/versi
```

Aliran Cabang Perbaikan Terbaru

```
git checkout master
git merge hotfix/fix-name
git checkout develop
git merge hotfix/fix-name
git branch -D hotfix/fix-name
```

Menandai

Saat Anda membuat tag, Anda memiliki dua jenis tag.

1. Tag QA
2. Rilis Tag

Sebuah tag QA dapat dibuat sebagai berikut.

```
git checkout rilis/v1.4.0

git tag -a v1.4.0-rc.1 -m "versi pengujian saya untuk 1.4"
```

Setelah pengujian selesai, Anda dapat menghapus tag. Tag ini dibuat dari cabang rilis yang relevan.

Rilis tag

Kami membuat tag ini dari cabang master untuk rilis produksi yang relevan.

```
git checkout master

git tag -a v1.4.0 -m "versi untuk 1.4.0"
```

f. Hasil dan Pembahasan

Jadi perlu diingat bahwa cabang master dan cabang pengembangan Anda harus memiliki kode yang sama sebelum pengembangan. Alasan kami mempertahankan dua cabang adalah untuk melacak sejarah proyek. Cabang master menyimpan riwayat rilis dan cabang pengembangan untuk integrasi fitur. Setelah fitur terintegrasi, kami menggabungkan cabang pengembangan ke cabang master kami. Pengembang lain harus mengkloning repositori pusat dan membuat cabang pelacakan untuk dikembangkan.

g. Kesimpulan

Mahasiswa menjelaskan mengenai hasil tugas praktik e-commerce serta mampu membuat sebuah konten yang menarik sehingga dipresentasi yang baik dan benar.

h. Rubrik Penilaian

No	Indikator	Skor*			
1	Ketepatan waktu dan ketepatan dalam menjelaskan dari tugas ditunjang dengan bukti referensi	1	2	3	(4)
2	Ketepatan waktu dan ketepatan dalam menjelaskan dari tugas	1	2	(3)	4
3	Ketepatan waktu akan tetapi kurang tepat dalam menjelaskan tugas	1	(2)	3	4
4	Keterlambatan pengumpulan tugas dan ketidaktepatan dalam menjelaskan tugas	(1)	2	3	4
Jumlah skor					