-Activity

Penjelasan Git role
Penjelasan Alur git

1. Buat repo baru
2. Add readme file
3. Membuat branch di upstream ada master, develop, dan release (dibuat ketika mau dinaikan ke master/prod)
4. Maintainer add developer
5. Maintainer clone repository
6. Atur config > bikin ssh
7. Developer fork repo ke akunnya
8. Developer clone repo ke local
9. Developer bikin branch feature penamaan pake hyphen-case, berlaku utk feature, bugfix, hotfix
10. Untuk commit developer harus menggunakan backlog id
11. Delete branch yang sudah di merge ke develop, baik itu feature, bugfix, hotfix
12. Developer pull new update from upstream
13. Developer push develop ke origin
14. Developer merge request ke remote branch develop
15. Maintainer code review and approve MR
16. Maintainer pull from remote on branch development
17. Maintainer create branch release (dilakukan kalau belum ada branch release)
18. Maintainer create tag (git tag v1.0.0) -> use semantic versioning
19. Maintainer push tag ke remote
20. Maintainer push release ke remote
21. Maintainer merge latest tag ke master
22. Maintainer push master ke remote
23. Maintainer create change log file on branch release
24. Maintainer write tag version history (tag version, date time, and changes)
25. Maintainer push change log file ke remote branch release

Study case:
1. Develop dengan tidak ada bug
2. Ada bug
3. Ada hotfix

Note:
Case 1 hanya sampai branch development
Case 2 dan 3 dinaikkan ke production

Preparation:

1. Buat akun gitlab
2. Config ssh

Konfigurasi SSH

1. Open Terminal
2. Ketik Command berikut :

   ssh-keygen -t rsa -b 4096 -C "your_email@email.com"

3. Akan muncul pesan berikut :

   Enter file in which to save the key (/Users/<your_user>/.ssh/id_rsa

4. Jika diminta password, lewati saja dengan menekan **ENTER**
5. Set SSH hasil generate tadi di gitlab (Playcourt)
   5.1. Buka hasil generate SSH, masuk ke folder .ssh/id_rsa.pub
   5.2. Copy SSH (SSH diakhir dengan alamat email yang sebelumnya dimasukkan)
   5.3. Buka Gitlab Playcourt
   5.4. Masuk ke Profile Setting
   5.5. Pilih Menu SSH pada Sidebar
   5.6. Paste Text SSH hasil generate ke dalam settingan SSH di playcourt.

Aturan Penamaan Branch dan Commit

1. **Branch**

   Penamaan branch diawali dengan jenis branch dan menggunakan hypen-case untuk nama branch-nya, contoh :

   1. feature-home
   2. bugfix-register
   3. hotfix-login

2. **Commit Message**

   Penamaan commit message haruslah singkat padat dan jelas dan diakhir dengan kode backlog dengan simbol (#) dari setiap fitur yang dikerjakan maupun merupakan bugfix/hotfix, Contoh :

   1. "add feature login #STDGF-001"
   2. "bugfix feature register #STDGF-002"
   3. "hotfix feature login #STDGF-001"

Studi Kasus Gitflow

Anggap developer sudah fork repo upstream dan cloning repo origin dengan cara :

- Fork Repo upstream
- ***git clone <SSH_REPO_ORIGIN>***

Lakukan konfigurasi remote, hal ini dimaksud untuk memetakkan repo untuk dilakukan sebuah aksi nantinya dengan cara :

- ***git remote add upstream <SSH_REPO_UPSTREAM>***

Gitflow Pre-cases

Sebelum memulai kasus-kasus yang mungkin terjadi, hal yang harus dilakukan adalah membuat sebuah branch develop yang akan digunakan sebagai branch yang menampung semua aktivitas development di repo local dengan cara :

Maintainer

Install AutoChangelog

- ***npm install -g auto-changelog***

Developer

Buat branch develop

- ***git branch develop***

Pull terlebih dahulu dari develop upstream

- ***git pull upstream develop***

Terdapat 1 developer akan men-*develop* fitur login.
Hal yang dilakukan adalah :

1. Buat branch fitur login dengan aturan penamaan
   feature-<nama_fitur>
   contoh : **feature-login**
   - ***git branch feature-login***
2. Checkout ke branch **feature-login**
   - ***git checkout feature-login***
3. Lakukan aktivitas development *(coding)*

4. Pull terlebih dahulu dari **develop upstream**
    - ***git pull upstream develop***
5. Jika terjadi konflik, *resolve* terlebih dahulu
6. Setelah itu barulah lakukan commit hasil development fitur tersebut dengan cara :
    - ***git add -A***
    - ***git commit -m "add feature login #SDTGF-001"***
7. Setelah commit, barulah kita akan merge ke branch develop dengan cara
    a. Checkout ke branch develop
        - ***git checkout develop***
    b. Merge branch feature login
        - ***git merge feature-login --no-ff***
    c. Push ke branch develop
        - ***git push origin develop***
8. Setelah itu lakukan ***Merge Request*** dari **Develop Origin** ke **Develop Upstream**
9. Jangan lupa setelah itu branch feature dihapus dengan cara :
        - ***git branch -d feature-login***

Case 2 : Bugfix Development (There's a Bug in Development)

Terdapat 2 kasus yaitu :

1. Bugs terdeteksi setelah dilakukan Merge oleh Maintainer di repo Upstream
2. Bugs terdeteksi setelah dilakukan commit di repo Local

Case 2.1

Developer men-*develop* fitur login dan terdapat bugs setelah dilakukan merge di repo Upstream.

Hal yang dilakukan adalah :

1. Pastikan posisi pada branch develop dengan cara :
    - ***git checkout develop***
2. Pull terlebih dahulu dari develop upstream untuk update code terbaru
    - ***git pull upstream develop***
3. Buat branch bugfix dari fitur yang terdapat bug contoh :
    - ***git branch bugfix-login***
4. Checkout ke branch bugfix

- ***git checkout bugfix-login***

5. Lakukan pengerjaan bugfix *(coding)*
6. Setelah itu jangan lupa untuk pull dahulu dari develop upstream dengan cara :terlebih
    - ***git pull upstream develop***
7. Setelah selesai lakukan commit pada hasil development bug tersebut
    - ***git add -A***
    - ***git commit -m "bugfix login #SDTGF-001"***
8. Setelah commit, barulah kita akan merge ke branch develop dengan cara
    a) Checkout ke branch develop
        - ***git checkout develop***
    b) Merge branch feature login
        - ***git merge bugfix-login --no-ff***
    c) Push ke branch develop
        - ***git push origin develop***
9. Setelah itu lakukan ***Merge Request*** dari **Develop Origin** ke **Develop Upstream**
10. Jangan lupa setelah itu branch feature dihapus dengan cara :
    - ***git branch -d bugfix-login***

Case 2.2

Developer men-*develop* fitur register dan terdapat bugs setelah dilakukan commit di repo Local.

Hal yang dilakukan adalah :

1. Pastikan posisi pada branch develop dengan cara :
    - ***git checkout develop***
2. Pull terlebih dahulu dari develop upstream untuk update code terbaru
    - ***git pull upstream develop***
3. Misal developer buat branch feature register contoh :
    - ***git branch feature-register***
4. Checkout ke branch feature-register
    - ***git checkout feature-register***
5. Lakukan pengerjaan fitur register *(coding)*
6. Pull terlebih dahulu dari develop upstream
    - ***git pull upstream develop***
7. Lakukan commit hasil pengerjaan fitur register

- ***git add -A***
- ***git commit -m “add feature register #STDGF-002”***

8. Setelah commit nampaknya terdapat bugs, nah yang harus dilakukan adalah tetap lanjutkan merge ke develop terlebih dahulu dengan cara
   - ***git checkout develop***
   - ***git merge feature-register --no-ff***
9. Lalu setelah dilakukan merge, buatlah branch baru bernama bugfix dengan nama bugfix-register
   - ***git branch bugfix-register***
10. Checkout ke branch bugfix
    - ***git checkout bugfix-register***
11. Lakukan pengerjaan bugfix *(coding)*
12. Setelah itu jangan lupa untuk pull terlebih dahulu dari develop upstream dengan cara :
    - ***git pull upstream develop***
13. Setelah selesai lakukan commit pada hasil development bug tersebut
    - ***git add -A***
    - ***git commit -m “bugfix register #STDGF-002”***
14. Setelah commit, barulah kita akan merge ke branch develop dengan cara
    a. Checkout ke branch develop
       - ***git checkout develop***
    b. Merge branch feature login
       - ***git merge bugfix-register --no-ff***
    c. Push ke branch develop
       - ***git push origin develop***
15. Setelah itu lakukan ***Merge Request*** dari **Develop Origin** ke **Develop Upstream**
16. Jangan lupa setelah itu branch feature dihapus dengan cara :
    - ***git branch -d feature-register***
    - ***git branch -d bugfix-register***

Case 3 : Hotfix Development (There’s a Bug in Production)

Bagaimana kasus ini bisa terjadi? kasus ini terjadi apabila terjadi bugs setelah code naik ke production. Cara mengatasinya adalah kita melakukan yang namanya **Hotfix**. Contoh adalah terdapat bugs register di production.

1. Pastikan posisi ada di branch **Master**
    - ***git checkout master***
2. Lakukan Pull All Branch dari master (untuk dapatkan semua branch upstream)
    - ***git pull upstream***
3. Lakukan Pull terlebih dahulu dari Upstream Master
    - ***git pull upstream master***
4. Buat Branch Hotfix dengan nama fiturnya
    - ***git branch hotfix-register***
5. Checkout ke branch Hotfix
    - ***git checkout hotfix-register***
6. Lakukan pengerjaan aktivitas hotfix *(coding)*
7. Pull terlebih dahulu dari upstream master
    - ***git pull upstream master***
8. Commit hasil pengerjaan
    - ***git add -A***
    - ***git commit -m "hotfix register #STDGF-002"***
9. Checkout ke Branch release (Hasil tarik all branch dari repo Upstream, tidak dibuat sendiri)
    - ***git checkout release***
10. Lakukan Merge ke branch release
    - ***git merge hotfix-register --no-ff***
11. Push ke branch release
    - ***git push origin release***
12. Lakukan Merge Request dari **RELEASE** Origin ke **RELEASE** upstream
13. Jangan lupa untuk delete branch hotfix
    - ***git branch -d hotfix-register***

ROLE : MAINTAINER

Case 1 : Masih di develop (belum naik ke production)

Tugas Maintener:

1. Code review and merge MR

Case 2 (Dinaikkan ke production)

Tugas Maintainer:

1. Clone Repo
    - **git clone <SSH_REPO>**
2. Pindah ke direktori project
    - **cd <Direktori>**
3. Checkout ke branch release
    - ***git checkout release***
4. Merge release dengan branch develop
    - ***git merge develop***
5. Berikan tag
    - ***git tag v1.0.0***
6. Create Changelog file
    - ***auto-changelog***
7. Commit Changelog file
    - ***git add .***
    - ***git commit -m "<message>"***
8. Checkout ke branch master
    - ***git checkout master***
9. Merge branch dengan tag tersebut
    - ***git merge v1.0.0***
10. Push tag
    - ***git push origin v1.0.0***
11. Push ke branch master
    - ***git push origin master***
12. Checkout ke release
    - ***git checkout release***
13. Push ke release
    - ***git push origin release***

Case 3 : Terjadi Hotfix dan dinaikkan ke Production

Tugas Maintainer :

1. Checkout ke branch release
    - ***git checkout release***
2. Pull dari origin release

- ***git pull origin release***

3. Berikan tag
    - ***git tag v1.0.1***
4. Create Changelog file
    - ***auto-changelog***
5. Commit Changelog file
    - ***git add .***
    - ***git commit -m “<message>”***
6. Checkout ke master
    - ***git checkout master***
7. Merge tag dengan branch master
    - ***git merge v1.0.1***
8. Push tag
    - ***git push origin v1.0.1***
9. Push ke branch master
    - ***git push origin master***
10. Checkout ke release
    - ***git checkout release***
11. Push ke release
    - ***git push origin release***