Buổi 2 – Luyện tập nâng cao với Git

Hoạt động 1: Ôn tập nhiều trạng thái file trong Git

Yêu cầu chi tiết

- Tạo nhánh mới `practice-branch` từ `main`.
- Tạo 3 file: `practice1.txt`, `practice2.txt`, `practice3.txt`.
- Stage và commit `practice1.txt`.
- Stage `practice2.txt` nhưng không commit.
- Giữ `practice3.txt` chỉ ở working directory.
- Quan sát sự khác nhau giữa 3 trạng thái file.
- Thực hành đổi tên và xóa file bằng Git.
- Push nhánh lên GitHub và tạo Pull Request.

Các bước thực hiện

```
git checkout -b practice-branch

echo "File cho staging + commit" > practice1.txt
echo "File cho staging, chưa commit" > practice2.txt
echo "File ở working directory" > practice3.txt

git add practice1.txt
git commit -m "Add practice1"

git add practice2.txt
# practice3.txt để nguyên

git status
git diff --staged

git restore practice3.txt
git restore --staged practice2.txt

git mv practice1.txt exercise1.txt
git commit -m "Rename practice1 to exercise1"

git rm exercise1.txt
git commit -m "Remove exercise1"

git push origin practice-branch
```

Sản phẩm nộp bài

- Ảnh chụp `git status` và `git diff`.
- Ảnh chụp commit rename và remove.
- Link Pull Request từ `practice-branch` vào `main`.

Hoạt động 2: Sử dụng git stash

Yêu cầu chi tiết

- Chỉnh sửa `README.md` nhưng chưa commit.
- Lưu thay đổi bằng `stash`.
- Tạo commit khác trong file `note.md`.
- Khôi phục thay đổi từ `stash`.
- Thử stash nhiều lần, apply stash cụ thể, drop và clear stash.

Các bước thực hiện

```
echo "Thêm ghi chú buổi 2" >> README.md
git stash save "update readme"
git stash list

echo "Ghi chú" > note.md
git add note.md
git commit -m "Add note.md"
git push

git stash pop
git add README.md
git commit -m "Restore readme changes"
git push

# stash nhiều lần
echo "Stash 1" >> README.md
git stash
echo "Stash 2" >> README.md
git stash
git stash list

git stash apply stash@{1}
git stash drop stash@{1}
git stash clear
```

Sản phẩm nộp bài

- Ảnh chụp `git stash list`.
- Ảnh chụp nội dung `README.md` sau khi khôi phục.
- Ảnh chụp GitHub hiển thị commit đã được restore.
- Giải thích ý nghĩa stash

Hoạt động 3: Revert & Reset

Yêu cầu chi tiết

- Tạo file `history.txt`, thực hiện 3 commit liên tiếp.
- Dùng `git revert` để hủy commit cuối.
- Tạo thêm commit mới và thử `git reset` với 3 chế độ: soft, mixed, hard.
- Quan sát sự khác biệt bằng `git status` và `git log`.
- Dùng `git reflog` để khôi phục commit đã mất.

Các bước thực hiện

```
echo "Dòng 1" > history.txt
git add . && git commit -m "Add line 1"
echo "Dòng 2" >> history.txt
git commit -am "Add line 2"
echo "Dòng 3" >> history.txt
git commit -am "Add line 3"

git log --oneline
git revert HEAD

echo "Dòng 4" >> history.txt
git commit -am "Add line 4"
echo "Dòng 5" >> history.txt
git commit -am "Add line 5"

git reset --soft <commit-id>
git reset --mixed <commit-id>
git reset --hard <commit-id>

git reflog
git checkout <commit-id>
```

Sản phẩm nộp bài

- Ảnh chụp `git log --oneline` trước và sau revert.
- Ảnh chụp kết quả sau reset soft, mixed, hard.
- Ảnh chụp `git reflog` và commit được khôi phục.

Hoạt động 4: Tag và Release

Yêu cầu chi tiết

- Tạo file `version.txt` và commit với nội dung “Version 1.0”.
- Gắn lightweight tag `v1.0`.
- Cập nhật thành “Version 2.0”, commit và tạo annotated tag `v2.0`.
- Liệt kê, tạo thêm tag thử, xóa tag local/remote.
- Tạo nhánh mới từ tag `v2.0`.

Các bước thực hiện

```
echo "Version 1.0" > version.txt
git add .
git commit -m "Add version 1.0"
git tag v1.0
git push origin v1.0

echo "Version 2.0" > version.txt
git commit -am "Update version 2.0"
git tag -a v2.0 -m "Release version 2.0"
git push origin v2.0

git tag -l
git tag beta-1.0
git tag -d beta-1.0
git push origin --delete v1.0

git checkout -b branch-from-v2 v2.0
```

Sản phẩm nộp bài

- Ảnh chụp GitHub hiển thị tag `v1.0` và `v2.0`.
- Ảnh chụp `git tag -l` trong local.
- Ảnh chụp nhánh mới được tạo từ tag `v2.0`.

Hoạt động 5: Rebase thay vì merge

Yêu cầu chi tiết

- Tạo nhánh `feature-branch`, commit file `feature.md`.
- Quay lại `main`, commit file `main.md`.
- Thực hiện merge và quan sát lịch sử commit.
- Reset và thử lại bằng rebase, so sánh kết quả.
- Tạo conflict và giải quyết khi rebase.

Các bước thực hiện

```
git checkout -b feature-branch
echo "Nội dung tính năng" > feature.md
git add . && git commit -m "Add feature.md"
git push origin feature-branch

git checkout main
echo "Main file" > main.md
git add . && git commit -m "Add main.md"

git merge feature-branch
git log --graph --oneline --all

git reset --hard HEAD~1
git rebase feature-branch
```

```
git log --graph --oneline --all

# Tạo conflict
# Sửa cùng dòng trong cả main.md và feature.md
git add . && git commit -m "Conflict commit"
git rebase feature-branch

# Giải quyết conflict
git add <file>
git rebase --continue
```

Sản phẩm nộp bài

- Ảnh chụp `git log --graph` sau merge và sau rebase.
- Ảnh chụp file khi bị conflict và sau khi giải quyết.
- Giải thích ngắn gọn 3–5 câu: sự khác nhau giữa merge và rebase.

📌 Yêu cầu nộp sau Buổi 2

- File Word tổng hợp:
  - Kết quả ảnh chụp màn hình.
  - Lệnh đã chạy.
  - Giải thích theo yêu cầu.
- Link repository GitHub sau buổi 2 (có commit, nhánh, tag, rebase).