Devops on AWS for beginner

Instructor – Linh Nguyen

Engineering Consultant, AWS Cloud Solution Architect

Thao tác cơ bản với Git

“Không có việc gì khó, chỉ sợ không biết làm!”

Target

- Hiểu được cách hoạt động của Git
- Cài đặt và sử dụng Git ở máy local
- Các thao tác cơ bản khi sử dụng Git trong dự án.
- Chiến lược quản lý branch trên Git hiệu quả.

Giới thiệu Git

Git là một hệ thống quản lý phiên bản (Version Control System - VCS) phổ biến được sử dụng để theo dõi các thay đổi trong **source code** trong quá trình phát triển phần mềm. Git được phát triển bởi **Linus Torvalds** vào năm 2005 và nhanh chóng trở thành một công cụ quan trọng cho các nhà phát triển.

Git giúp đội ngũ phát triển làm việc cùng nhau trên cùng một dự án mà không gặp vấn đề xung đột giữa các phiên bản khác nhau của source code. Nó giúp theo dõi lịch sử thay đổi, quản lý nhánh (branching) để phát triển đồng thời nhiều tính năng hoặc sửa lỗi, và có khả năng hồi phục lại các phiên bản trước đó của dự án..

Đặc trưng của Git

Open-source.

Kiến trúc của Git: Hoạt động theo mô hình phân tán (Distributed), mỗi developer sẽ có 1 bản copy của code base và có thể change/sync với nhau. Kiến trúc distributed cũng cho phép developer có thể làm việc ngoại tuyến (no-internet).

Hoạt động hiệu quả và nhanh chóng ngay cả với những repository có codebase lớn.

Sử dụng hệ thống content-addressable file system để lưu tất cả version của các file trong repository giúp access nhanh chóng và lấy lại version cũ một cách dễ dàng.

Thay đổi trên Git thông thường sẽ là add mới nội dung. Ngay cả khi bạn revert một thay đổi thì nó sẽ tạo thêm 1 commit mang nghĩa **revert** chứ không xoá hoàn toàn.

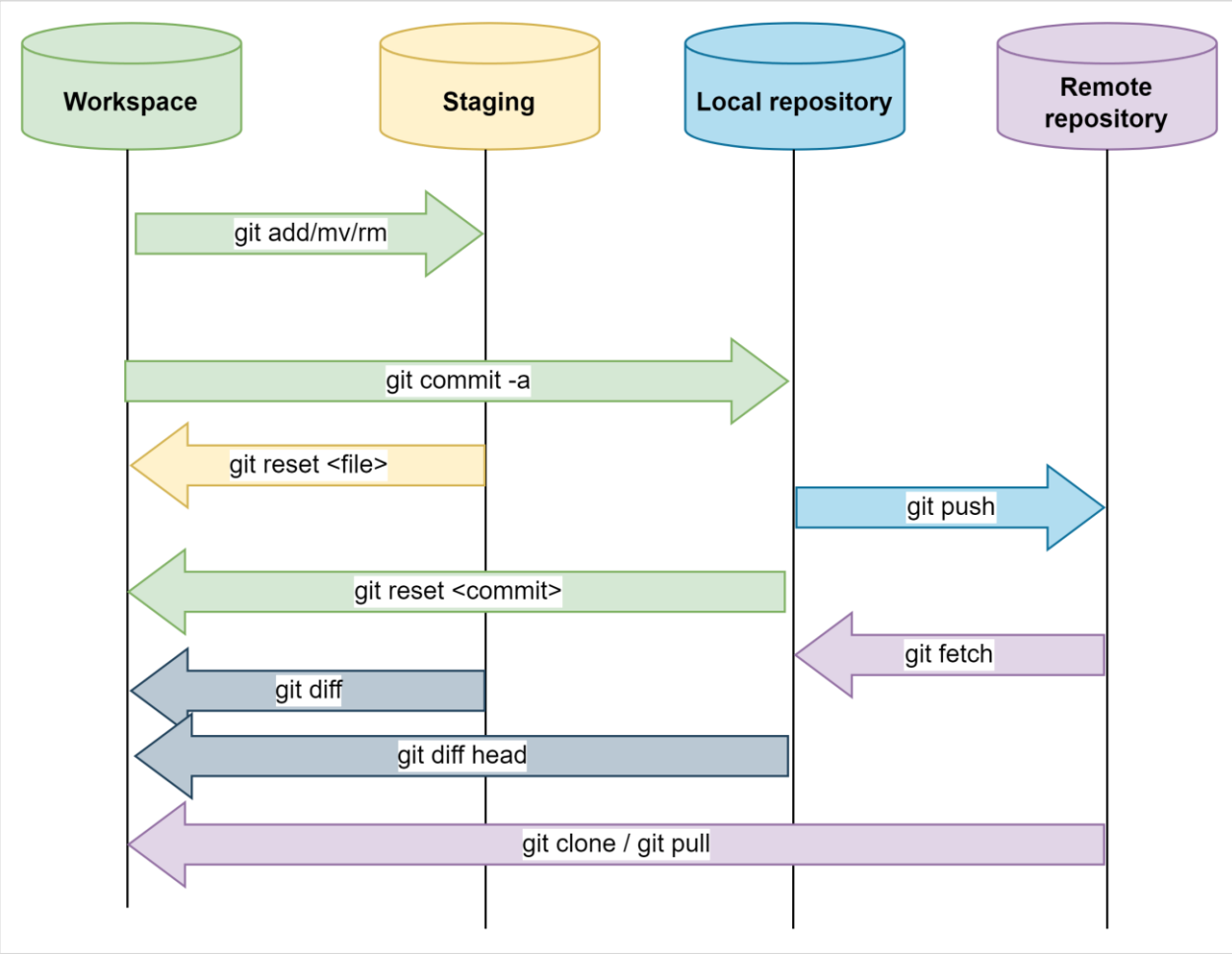
Kiến trúc của Git(1)

GitRepository: nơi lưu trữ toàn bộ file code, thư mục, project history. Repository có thể nằm ở máy local, server hoặc trên Cloud.

Commit: ảnh chụp (snapshot) của codebase tại 1 thời điểm cụ thể.

Branch: một line riêng biệt tạo từ 1 commit của codebase.

Merge: kết hợp 1 branch vào branch khác.



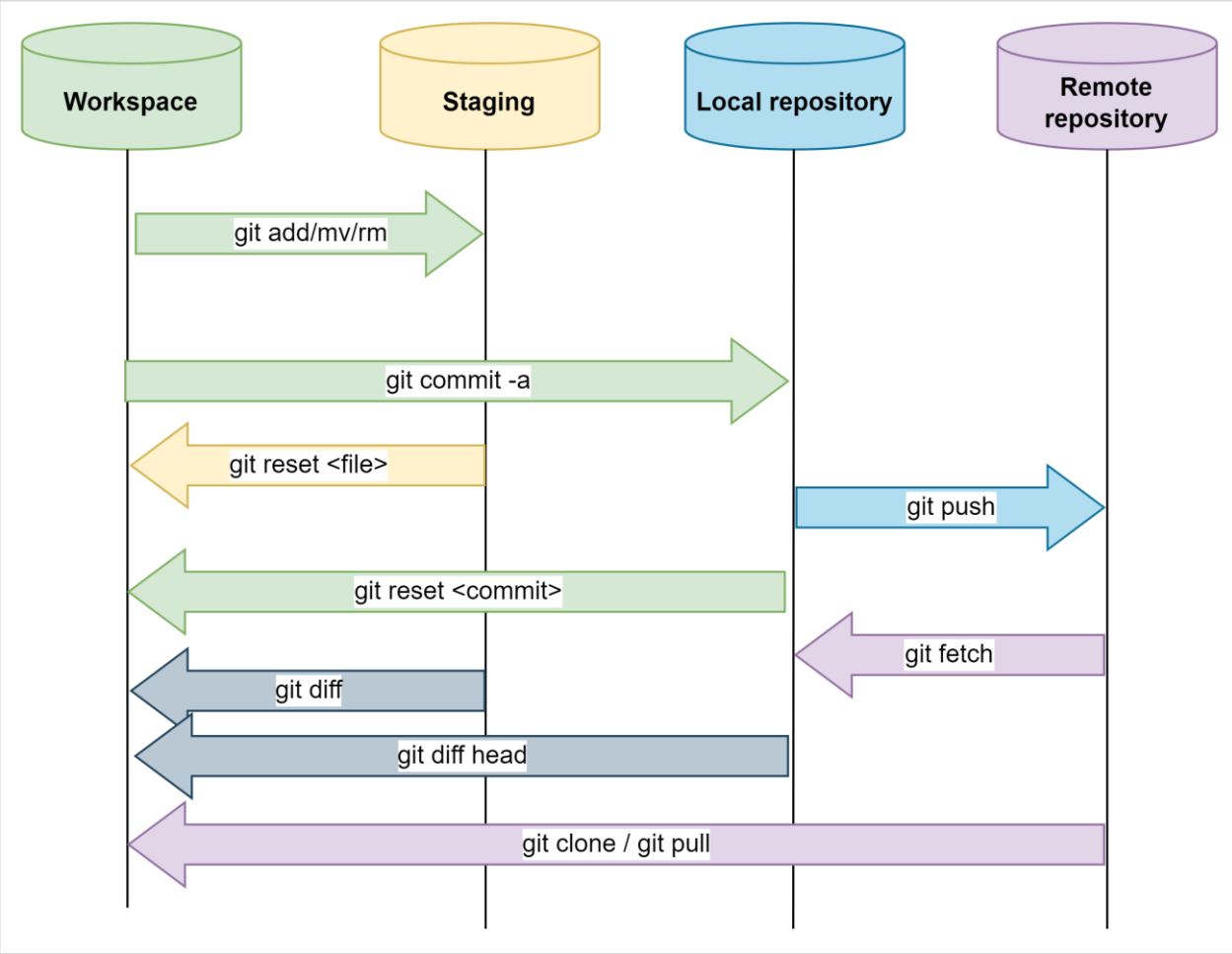
Kiến trúc của Git(2)

Remote: 1 bản copy của repository nằm trên máy khác hoặc server. Remotes có thể được sử dụng để cộng tác với các nhà phát triển khác và đồng bộ hóa các thay đổi giữa các bản sao khác nhau của repository

Clone: 1 bản copy của repository.

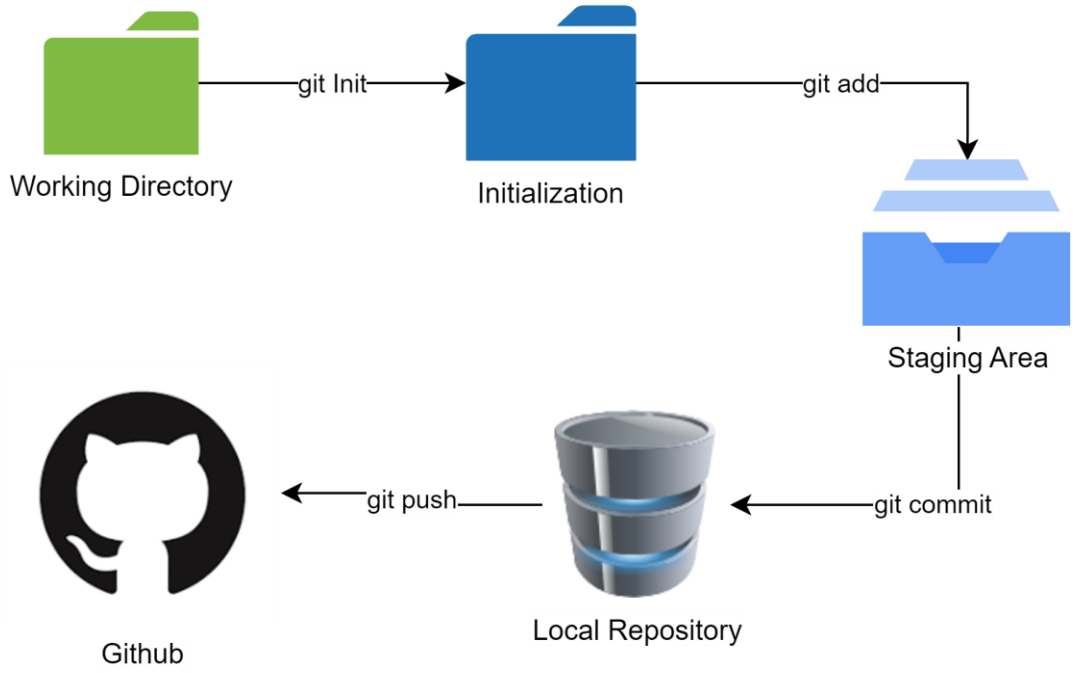
Pull: thao tác download thay đổi từ một remote repository và merge chúng vào local repository.

Push: thao tác upload thay đổi từ local repository lên remote repository



Git life cycle

1. **Create**: Tạo mới repository trên máy local hoặc trên remote server
2. **Modify**: add thêm các file code sử dụng các IDE. Git sẽ tự động detect các thay đổi.
3. **Stage**: sử dụng git add command để chuẩn bị các thay đổi sẽ được commit vào repository.
4. **Commit**: Apply các thay đổi vào repository.
5. **Push**: đẩy những thay đổi từ local repository lên một repository khác (vd Github, Git server công ty.

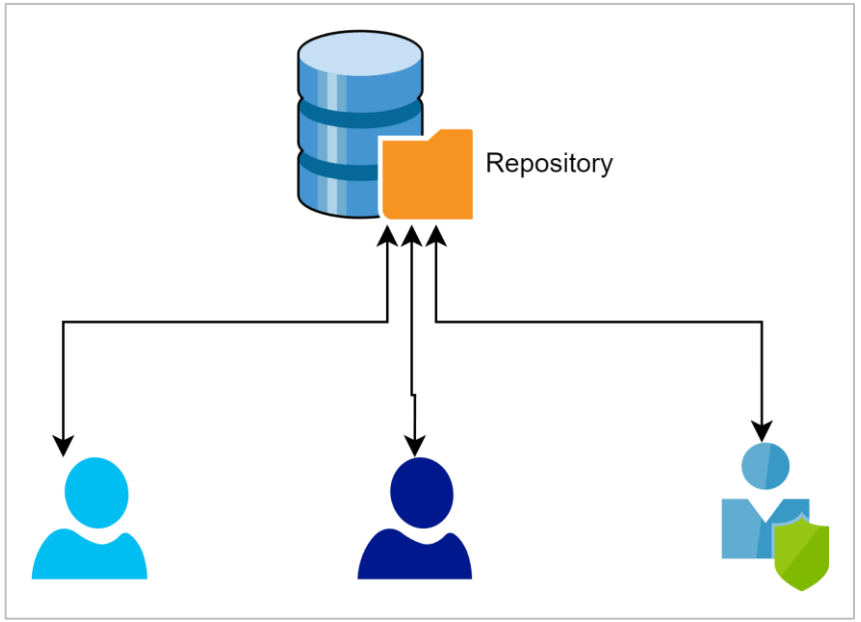


Một số mô hình Workflow

Centralized workflow: Phù hợp cho những dự án nhỏ, codebase đơn giản.

Developer clone repository về máy local, make thay đổi sau đó push ngược trở lại central repository.

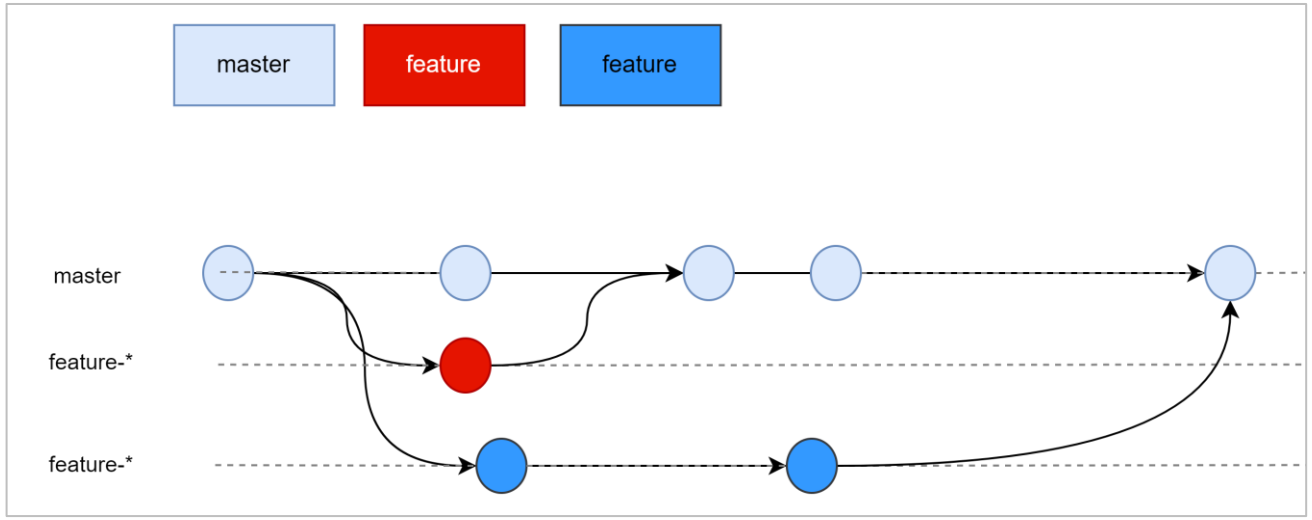
*Nhược điểm: dễ gây conflict khi nhiều người sửa chung 1 module or file.



Một số mô hình Workflow

Feature Branch workflow.

Mỗi khi developer phát triển chức năng mới, họ sẽ tạo một nhánh riêng từ nhánh chính. Sau khi kết thúc công việc sẽ tạo pull-request để merge ngược trở lại.



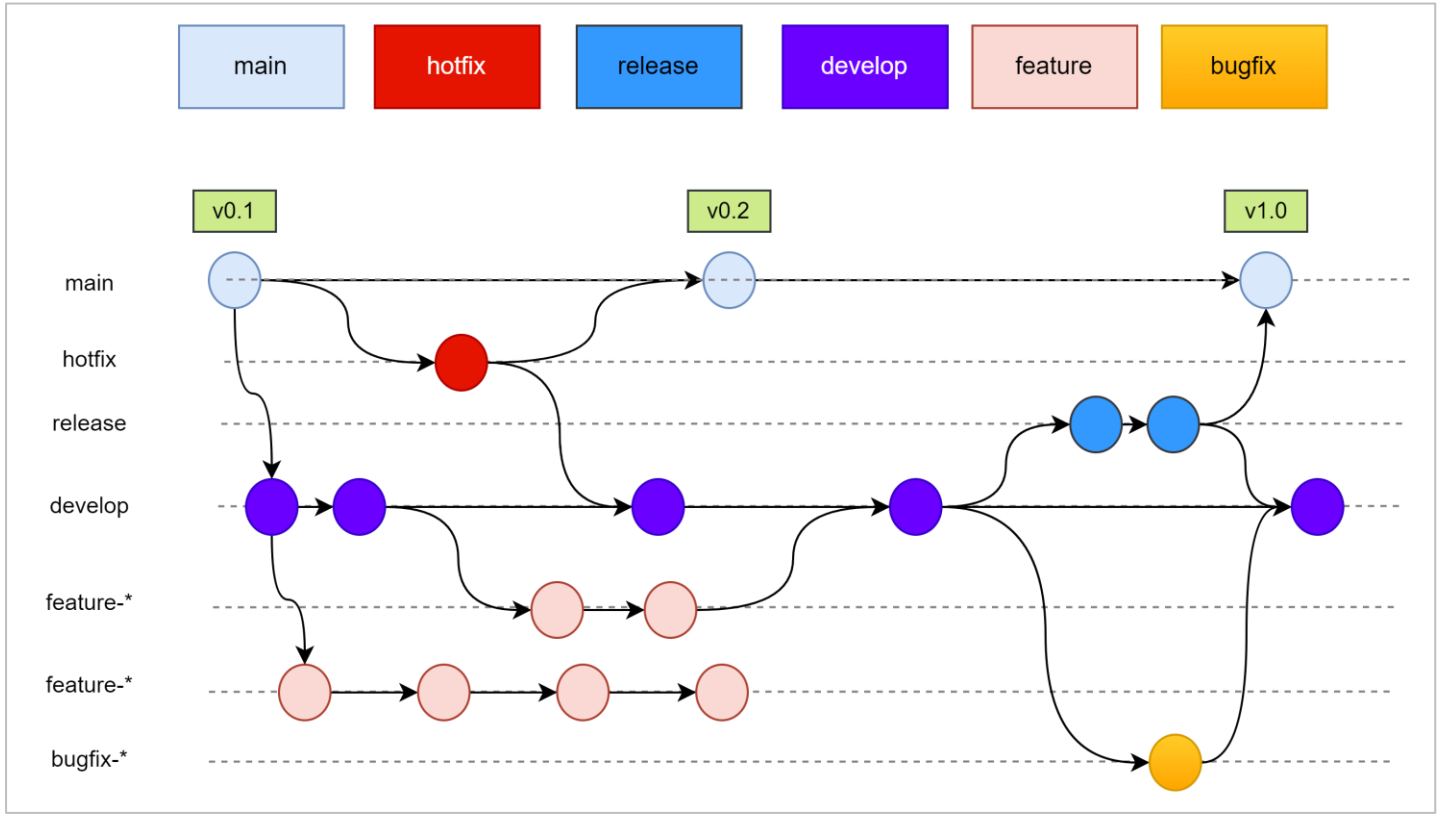
Một số mô hình Workflow

Gitflow workflow.

Phù hợp cho dự án lớn, codebase phức tạp, có thời gian phát triển dài.

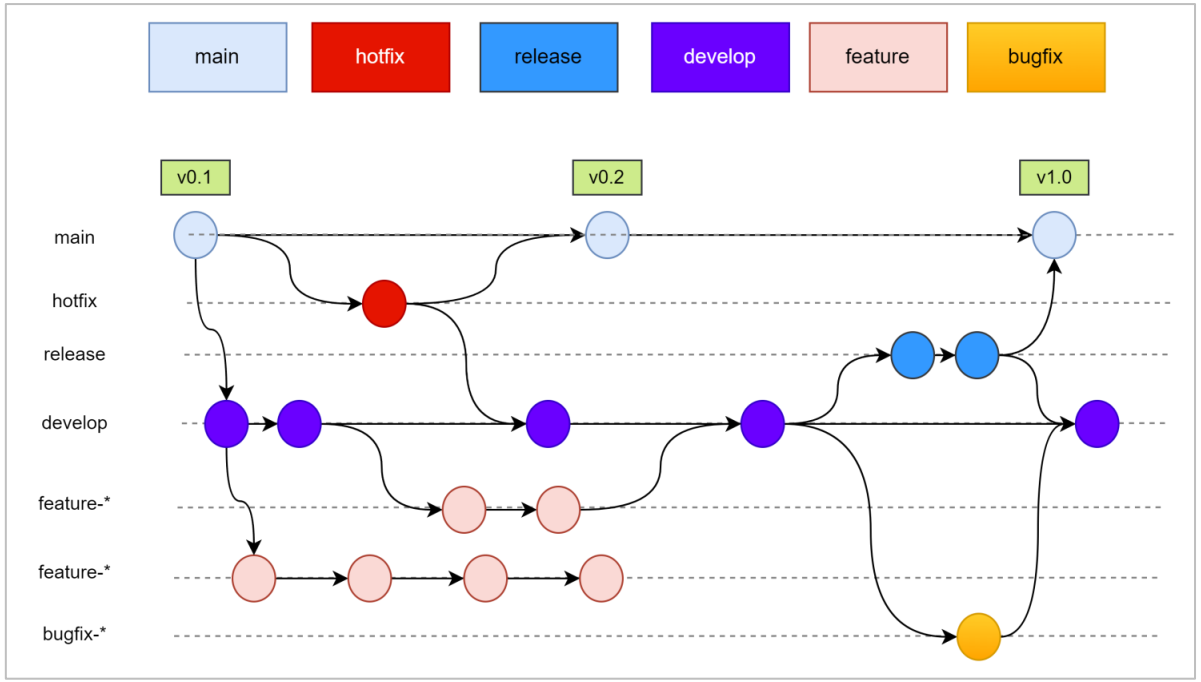
Mô hình này chia branch thành các branch có nhiệm vụ khác nhau như:

- main
- develop
- feature-*
- bugfix-*
- release-*
- hotfix-*



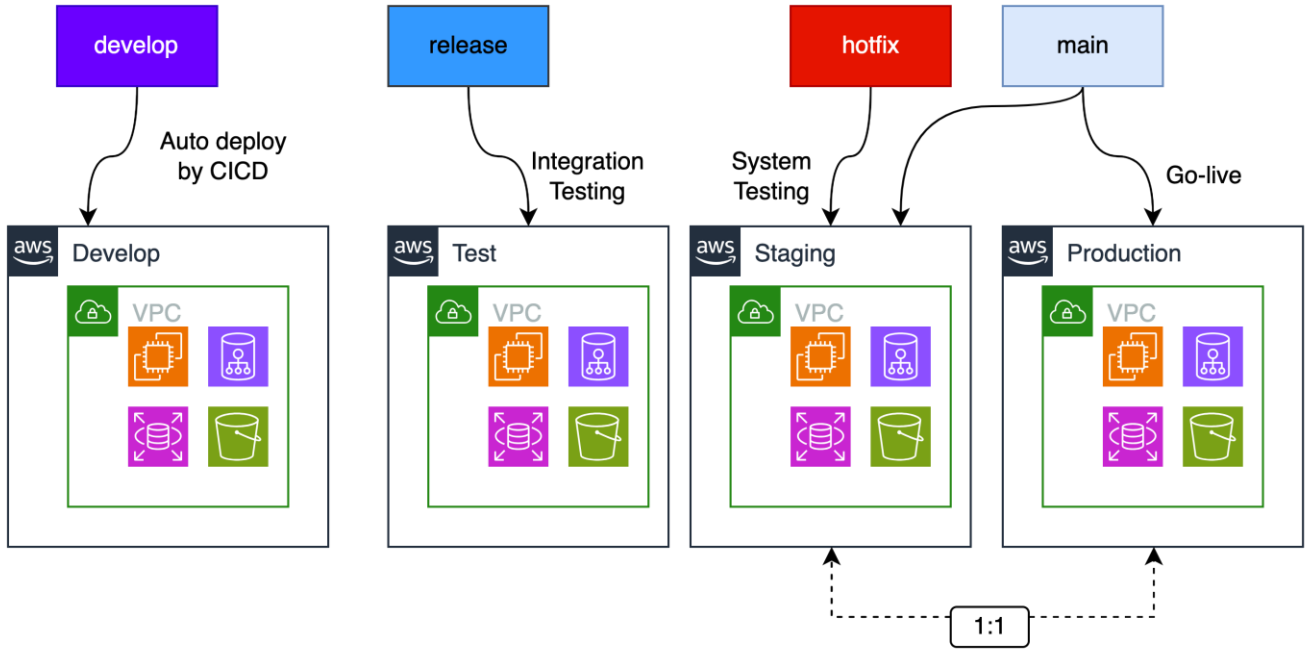
Gitflow workflow – Detail

- main: nhánh chính duy trì xuyên suốt vòng đời dự án. Chỉ được merge code vào khi có release lớn hoặc hot-fix. Được đánh tag theo release version.
- develop: nhánh dùng để các developer phát triển feature/fix-bug. Code được checkout từ đây và merge trở lại khi hoàn thành task.
- feature-*: naming rule tuỳ theo dự án. vd feature-<ticket-id>. Checkout từ nhánh develop.
- bugfix-*: tương tự nhánh feature, dùng để fix bug.
- release: được tạo ra trước mỗi đợt release version mới, code thường được deploy lên môi trường test/staging, có thể thực hiện chỉnh sửa nhỏ(optional).



Gitflow workflow – Mapping giữa branch và môi trường deploy.

- Môi trường Dev: thường được deploy auto ngay khi có pull request được merge vào nhánh develop (CICD).
- Môi trường Test: sử dụng cho Integration Testing, thường được release theo phiên bản từ nhánh release.
- Môi trường Staging: sử dụng cho System Testing & hoạt động UAT trước khi release cho end-user. Về cơ bản mt Staging phải giống mt Production.
- Khi apply Hotfix, nên test trước với mt Staging trước khi apply merge vào main và deploy lên Production.



*Lưu ý: các quy tắc mapping trên đây mang tính tham khảo, hoàn toàn có thể tuỳ biến tuỳ theo quy mô dự án, yêu cầu từ khách hàng. Tuy nhiên, việc phá vỡ flow deployment chuẩn, apply hot-fix quá nhiều, release không kèm testing sẽ khiến chất lượng dự án đi xuống, tốn nhiều effort để sửa chữa.

Có bao nhiêu nhà cung cấp Git?

Một số nhà cung cấp dịch vụ Git

Lab1 - Thao tác cơ bản với Git – Cài đặt Git trên máy local.

- Cài đặt Git trên máy local.
- Cài đặt tool Source Tree trên máy local.
- Khởi động Gitbash
- Checkout 1 public repository trên Github.
- Mở repository bằng tool VSCode.
- Sử dụng SourceTree để xem các branch của repository.

Lab2 - Thao tác cơ bản với Git – Tạo repo trên Github, cấu hình SSH key

- Tạo một repo trên Github (private repository).
- Tạo một SSH key theo guide của Github.
  *Google keyword: *“github create ssh key”*
- Cấu hình private key tại máy local.
- Add public key lên Github account setting.
- Thử checkout một repo sử dụng SSH key.

Lab3 – Check out nhánh develop, thay đổi, commit, push, tạo Pull Request(PR).

- Tạo nhánh develop từ nhánh master của repository mới checkout.
- Tạo một vài files vd: **index.html, README.md**
- Push nhánh develop lên repository Github.

Lab4 – Tạo nhánh feature/bugfix từ nhánh develop.

- Thiết lập protect branch cho nhánh **master** và nhánh **develop** (chặn push trực tiếp).
- Tạo nhánh feature từ nhánh develop. Vd: feature/F001
- Modify một vài file code.
- Push nhánh **feature/F001** lên remote repository.
- Tạo Pull Request (PR) từ **feature/F001** vào nhánh **develop**.
- Tiến hành review và approve Pull Request.

Lab5 – Tạo nhánh release từ nhánh develop.

- Tạo nhánh **release/v0.1**.
- Merge nhánh **release/v0.1** vào master.
- Đánh tag cho nhánh master, vd **v0.1**
- [Optional] merge nhánh **release/v0.1** ->**develop** nếu có thay đổi

Lab6 – Git Revert

Git revert giống như việc "undo" một commit trước đó. Về bản chất Git sẽ không xoá commit trước đó mà sẽ tạo thêm 1 commit với ý nghĩa "revert".

Cú pháp:

git revert <commit-hash>

Thực hành: tiến hành commit 1 thay đổi sau đó tiến hành revert.

Lab7 – Xử lý xung đột trên Git.

Giả lập tình huống có 2 developer Dev-A và Dev-B cùng chỉnh sửa 1 chức năng dẫn đến người tạo Pull Request sau bị conflict.

- Repo có branch **develop** chứa file test.txt.
- Developer A checkout nhánh mới **feature/devA** và sửa test.txt. Sau đó add file test.txt và commit.
- Developer B checkout nhánh mới **feature/devB** và sửa test.txt. Sau đó add file a.txt và commit. Và cuối cùng merge vào **develop** không conflict.
- Người A lúc này merge vào **develop** nhưng **develop** lúc này đã chứa nội dung của devB chứ không còn là nội dung của develop lúc checkout -> conflict.
- Tiến hành resolve conflict sau đó tạo lại pull request.

Lab7 – Xử lý xung đột trên Git.

*Hạn chế conflict trong quá trình làm việc chung:

- Hạn chế code chung module.
- Luôn pull latest code trước khi thay đổi.

Lab8 – Sử dụng AWS CodeCommit

- Tạo một repo trên CodeCommit.
- Add SSH key cho IAM User.
- Checkout source code.
- Checkout branch
- Modify, Commit, Push, tạo Pull Request.
- Review & merge pull request trên Console.

Các thao tác khác

- Git log
- Git show <commit hash>
- Git stash: Lưu tạm công việc dang dở lên một khu vực để switch sang branch khác.

  git stash : cmd này chỉ include những tracked file.

  git stash --include-untracked : cmd này include cả những file untracked.

  git stash apply : lấy những thay đổi đã stash xuống branch hiện tại.
- Git reset: Reset branch hiện tại.

  git reset --hard <commit> :di chuyển branch về commit chỉ định và discard toàn bộ thay đổi sau commit đó.

  git reset --mixed <commit> : di chuyển branch về commit chỉ định và cho các file đã được change thành **unstaged**. Bạn có thể review & add lại file cần thiết.

  git reset --soft <commit> : di chuyển branch về commit chỉ định và keep các file đã được change thành **staged**. Bạn có thể chạy lại lệnh git commit để commit lại.

Phân biệt Git Merge v.s Rebase vs Squash commit

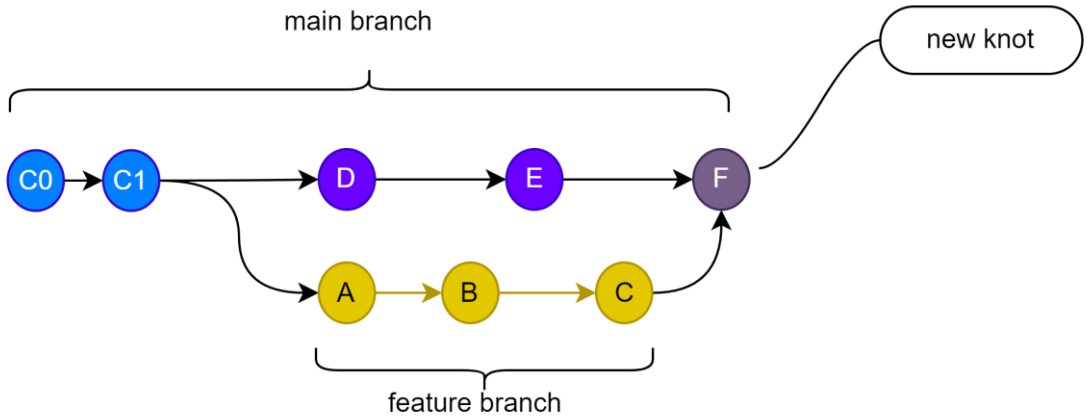
Khi combine code giữa nhánh feature với nhánh chính (vd develop), có 2 phương án thường được sử dụng:

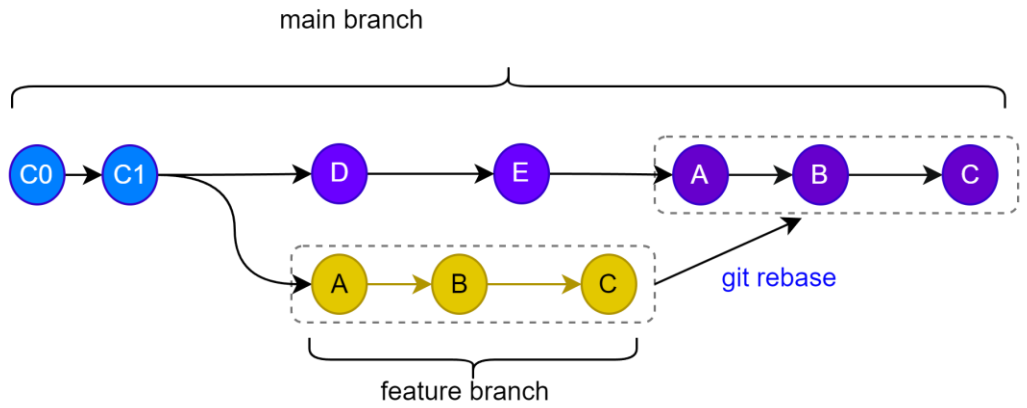
Merge: tạo ra một nút (knot) mới chung giữa hai nhánh.

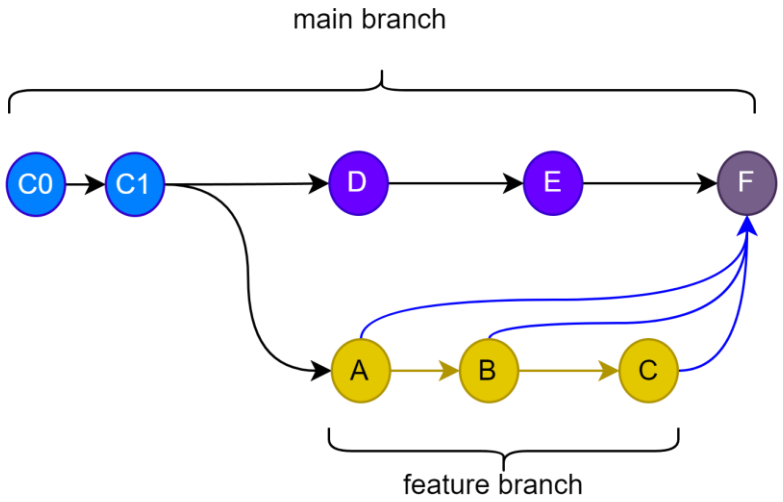
Rebase: Đem toàn bộ commit của nhánh source đẩy lên đầu nhánh target (theo thứ tự).

Squash commit: đem toàn bộ commit của nhánh source gom thành một commit duy nhất và đẩy lên đầu nhánh target.

Phân biệt Git Merge v.s Rebase vs Squash commit







Best practice for Git using

Use Branches: Sử dụng branch mỗi khi phát triển feature hoặc fix bug. Giúp developer làm việc mà không ảnh hưởng lẫn nhau, ngoài ra cũng dễ dàng review các thay đổi.

Commit thường xuyên: Giúp theo dõi tiến độ, cô lập các thay đổi, dễ dàng revert nếu cần thiết. Mỗi commit nên bao gồm logic và thay đổi nhỏ (atomic).

Viết commit message rõ ràng, dễ hiểu: giúp các member khác dễ dàng review code.

Pull Before Push: Luôn pull code latest từ remote repository trước khi push thay đổi. Giúp giảm thiểu conflict và đảm bảo đang làm việc với code base mới nhất.

Review Code: Phát hiện bug sớm, cải thiện chất lượng code, share knowledge giữa team member.

Resolve Conflicts một cách phù hợp: Dành thời gian để hiểu rõ nội dung code trước khi resolve conflict.

Use Git Tags and Releases: dùng git Tag cho các milestone quan trọng. Giúp theo dõi release, quản lý stable version, đơn giản hoá quy trình deploy.

Backup and Remote Repositories: Thường xuyên backup remote repository để tránh data loss. Cân nhắc sử dụng Gitlab hoặc Github (hoặc cloud provider).

Tổng kết & Clear resources

- Các resource như **Github** Repository, **CodeCommit** repository có thể xoá nếu không có nhu cầu hoặc giữ lại tham khảo trong tương lai.

Thanks you and see you in the next chapter!