Git Branching

التعريف

الـ **Branches** في Git بتساعدك تشتغل على نسخ مختلفة من الكود بنفس الوقت، يعني تقدر تجرب ميزات جديدة أو تصلح أخطاء **بدون ما تأثر على الكود الأساسي**.

كيف بيشتغل الـ Branching؟

- لما تعمل **Branch جديد**، هو بأخذ نسخة من الكود الحالي وبتشتغل عليه بشكل مستقل.
- بعد ما تخلص، بتقدر **تدمجه** مع الفرع الرئيسي (**main branch**).
- هيك بتضمن إن الشغل منظم وما تخرب الكود الأساسي.

المزايا

ليش بنستخدم الـ Branches؟

- **Parallel Development** ← كل واحد يقدر يشتغل على ميزة أو تعديل لحاله بدون ما يزعج باقي الفريق.
- **Isolation** ← أي تعديل أو تجربة جديدة بتكون في فرع لحاله، وما بتأثر على الكود الرئيسي إلا لما تقرر تدمجه.
- **Efficient Collaboration** ← الفريق كله بقدر يشتغل بشكل مرتب وكل واحد عنده فرعه الخاص.
- **Version Control** ← لو صار أي خطأ، بتقدر بسهولة ترجع للكود القديم بدون مشاكل.

حالات الاستخدام

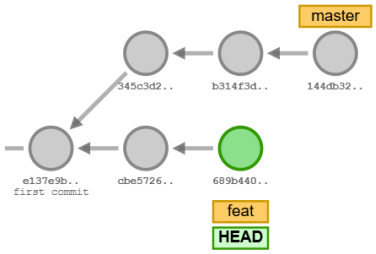
متى بنستخدم الـ Branches؟

- **Feature Development** ← لما بدك تضيف ميزة جديدة، بتعمل فرع خاص فيها، بتشتغل عليها، وبترجع تدمجها لما تخلص.
- **Bug Fixes** ← إذا طلع عندك **Bug**، بتفتح فرع لحاله، بتصلحه، وبعدين بترجعه للكود الأساسي.
- **Hotfixes** ← لو فيه **مشكلة مستعجلة** بالكود، بتفتح فرع سريع، بتصلحها، وبتدمجها عالسريع.
- **Experimentation** ← بدك **تجرب فكرة جديدة؟** افتح فرع خاص فيها، ولو عجبتك بترجع تدمجها، ولو ما زبطت بتكنسلها.
- **Code Reviews** ← لما بدك حد يراجع الكود تبعك قبل ما تضيفه، بتشتغل بفرع خاص، وبعد المراجعة بتدمجه.

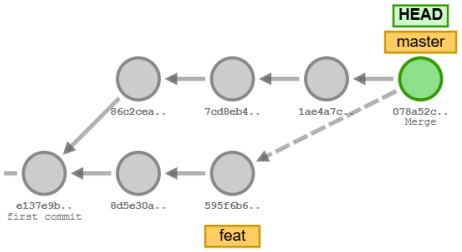
.1

Basic Git Branch Commands

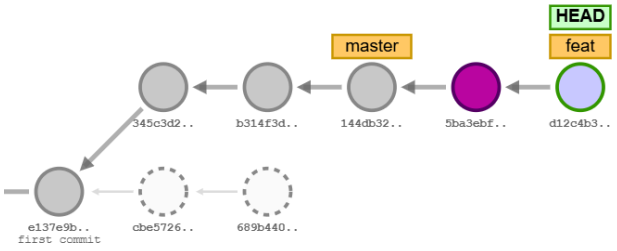
- `git branch` – List all branches
- `git branch <branch-name>` – Create a new branch
- `git branch -d <branch-name>` – Delete a branch
- `git branch -D <branch-name>` – Force delete a branch
- `git branch -M <branch-name>` – Rename the current branch
- `git checkout <branch-name>` – Switch to a branch
- `git checkout -b <branch-name>` – Create and switch to a new branch
- `git merge <branch-name>` – Merge changes from another branch
- `git rebase <branch-name>` – Rebase changes onto another branch



1. **Merge**: بدمج التغييرات من **Branch** واحد لـ **Branch** تاني عن طريق إنشاء **Commit** جديد.



2. **Rebase**: بنقل التغييرات من **Branch** واحد لـ **Branch** تاني عن طريق وضع الـ **Commits** فوق الـ **Branch** المستهدف.

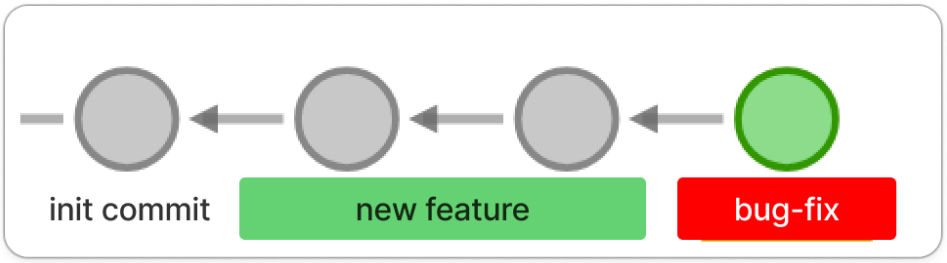


مثال على workflow

تخيل إنك شغال على ميزة كبيرة بتتطلب عدد من **Commits**. فجأة، مديرك يطلب منك إصلاح **Bug**. طبعاً ما بدك تخلط الكود غير المكتمل لميزتك مع إصلاح الـ **Bug**. هنا الـ **Branching** بحل المشكلة لأنو بخليك:

- تسوي **Branch** جديد لميزتك.
- ترجع للـ **main Branch** عشان تطبق إصلاح الـ **Bug**.
- تكمل شغلك على ميزتك بدون ما يحصل أي تداخل.

الشغل بدون ما نستخدم branch رح يكون كله بنفس الخط ورح يصير تداخل



بنستخدم الbranches عشان نشتغل على كل اشي بشكل معزول عن غيره