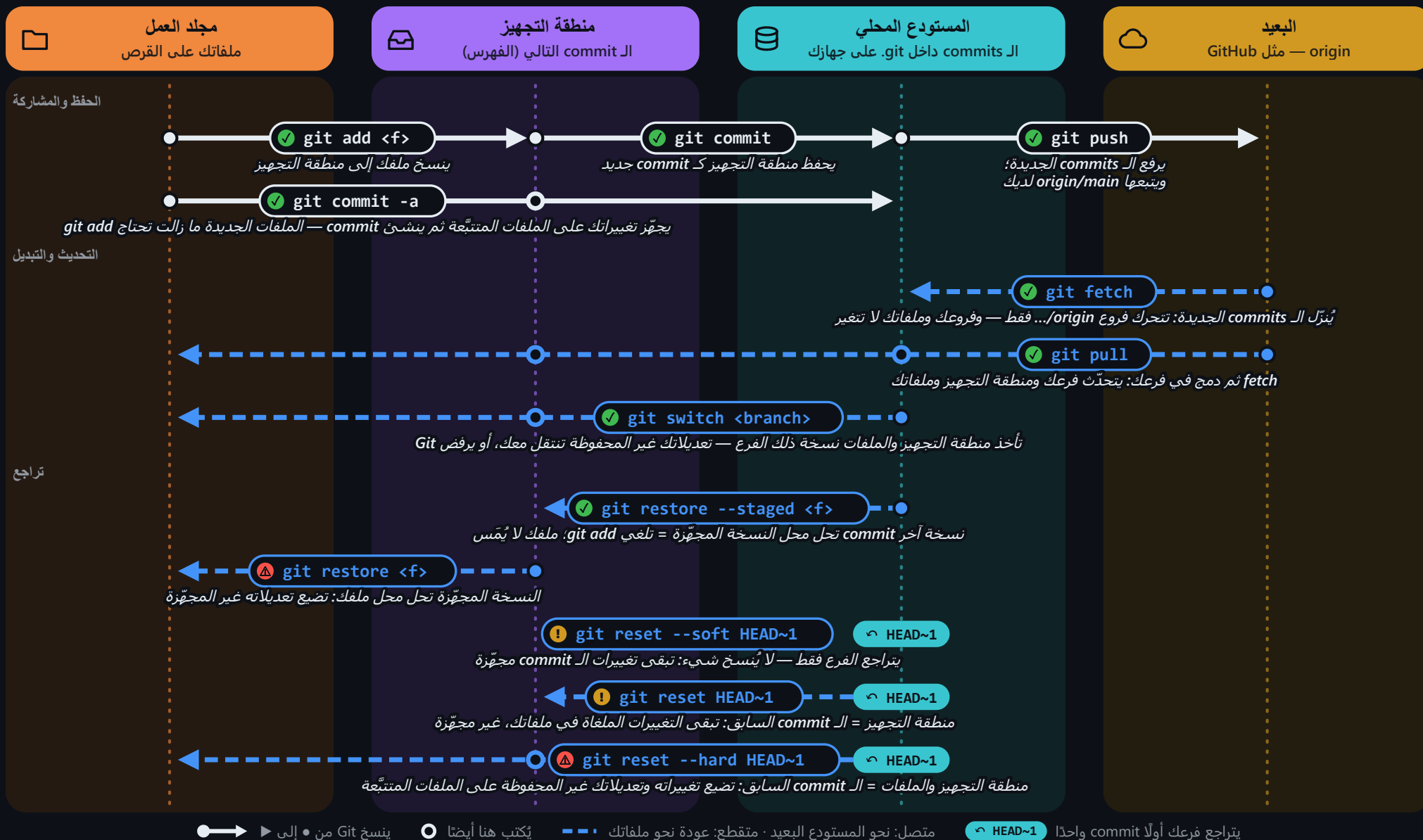




○ `<f>` مسار ملف: اكتب `app.js` دون `<>`

○ `HEAD~1` ال commit الذي يسبق الذي أنت عليه

○ إذا كان لديك وللمستودع البعيد commits جديدة، يتوقف Git حتى تختار: `git config pull.rebase false` (دمج) أو `git pull true` (دمج)

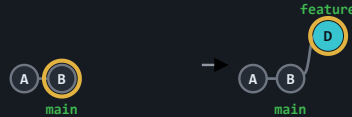
قبل ← بعد. الدوائر هي ال commits، والحلقة الصفراء هي موقعك (HEAD). ● جديد commit ○ لم يعد أي فرع يشير إليه

✓ git commit



commit جديد مما في التجهيز؛
ويتقدّم فرعك إليه.

✓ git switch -c feature + commit



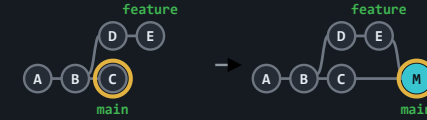
يبدأ فرع جديد من حيث أنت؛ تحرّكه
ال commits القادمة، ويبقى main.

✓ git merge feature (fast-forward)



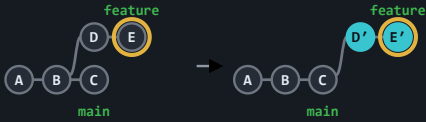
على main، عندما لا يحتوي main
على commits جديدة: يتقدّم main
ببساطة إلى E — دون commit دمج.

✓ git merge feature



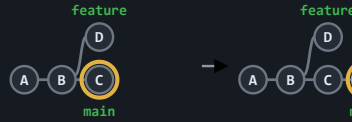
على main: يدمج feature بـ commit
دمج له أبوان.

! git rebase main



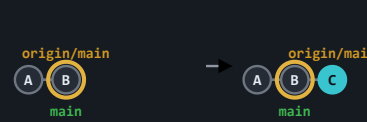
على feature: يعيد تطبيق D و E
فوق main كنسخ بـ hash جديد (D')
(E)؛ وتخرج الأصول من الفرع.

✓ git cherry-pick D



ينسخ commit واحدًا إلى فرعك (D)،
hash (جديد)؛ ويبقى الأصل.

✓ git fetch



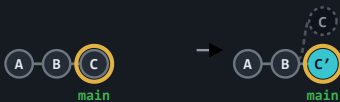
يُترّل ال commits الجديدة: تتحرك
تسميات /origin/main... فقط. فرعك
وملفاتك لا تتغير.

✓ git push



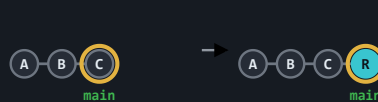
يرفع C: يتحرك main على المستودع
البعيد — و origin/main لديك —
إلى C. ملفاتك لا تتغير.

! git commit --amend



يستبدل آخر commit بآخر جديد (C')
hash (جديد)؛ ويخرج C من الفرع.
افعل ذلك قبل الدفع.

✓ git revert C



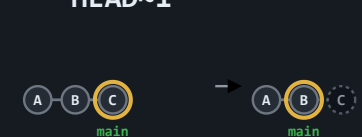
يضيف R commit يتراجع عن C.
يُحفظ السجل: آمن على الفروع
المشتركة.

! git reset --soft HEAD~1



يعيد فرعك إلى B؛ تبقى تغييرات C
مجهزة — أنشئ commit مجددًا
لإعادته.

! git reset --hard HEAD~1



يُرجع فرعك؛ تصبح منطقة التجهيز
والملفات B: تختفي تغييرات C (ما
زال git reflog يجد C).

البداية

- `git config --global user.name "Your Name"`
الاسم المسجل في ال commits (مرة واحدة لكل جهاز)
- `git config --global user.email "you@example.com"`
البريد الإلكتروني المسجل في ال commits
- `git init`
تحويل المجلد الحالي إلى مستودع
- `git clone <url>`
تنزيل مستودع مع تاريخه كاملاً

استكشاف

- `git status`
ما الذي تغيّر، وما المجهّز، وعلى أي فرع أنت
- `git diff`
تغييرات لم تُجهّز بعد (مجلد العمل مقابل منطقة التجهيز)
- `git diff --staged`
ما سيحتويه ال commit التالي
- `git log --oneline --graph --all`
تاريخ كل فرع، سطر لكل commit
- `git show <commit>`
واحد: الرسالة commit والتغييرات

حفظ عملك

- `git add <file>`
تجهيز ملف لل commit التالي
- `git add -p`
تجهيز أجزاء مختارة فقط (hunks)
- `git commit -m "message"`
تسجيل ما هو مجهّز ك commit
- `git commit --amend`
إعادة آخر commit (الرسالة أو المحتوى) ⚠️
- `git restore <file>`
تجاهل التعديلات غير المجهّزة في ملف ⚠️
- `git restore --staged <file>`
إلغاء التجهيز (يبقى تعديلك على القرص)

المشاركة

- `git fetch`
تنزيل commits جديدة؛ فروعك لا تتحرك
- `git pull`
ثم الدمج في الفرع الحالي
- `git pull --rebase`
fetch ثم إعادة تطبيق ال commits الخاصة بك فوقها (hash جديد) ⚠️
- `git push`
رفع ال commits إلى الفرع المقابل البعيد (upstream)
- `git push -u origin <branch>`
أول push لفرع جديد (يُضبط ال upstream)
- `git remote -v`
المستودعات البعيدة التي يتواصل معها هذا المستودع

- ال commit الذي أنت عليه (عادةً آخر commit في فرعك الحالي) HEAD
- الاسم الافتراضي للمستودع البعيد الذي نسخت منه origin
- مسار ملف: اكتب `app.js` دون `<` `>` `<file>`

الإنشاء والتبديل

- `git branch`
عرض الفروع المحلية (* = الحالي)
- `git branch -a`
عرض فروع التتبع البعيدة أيضًا
- `git switch <branch>`
الانتقال إلى فرع (الملفات تتبعه)
- `git switch -c <branch>`
إنشاء فرع هنا والانتقال إليه
- `git switch -`
العودة إلى الفرع السابق
- `git branch -m <new-name>`
إعادة تسمية الفرع الحالي

الجمع

- `git merge <branch>`
جلب فرع إلى الفرع الحالي
- `git merge --no-ff <branch>`
تسجيل commit دمج دائمًا
- `git rebase <branch>`
إعادة تطبيق ال commits الخاصة بك فوق <branch> (hashes جديدة) ⚠️
- `git rebase -i HEAD~3`
تعديل آخر 3 commits: إعادة ترتيب، ضمّ، إعادة صياغة، حذف ⚠️
- `git cherry-pick <commit>`
نسخ commit إلى الفرع الحالي

التعارضات

- `git status`
يسرد الملفات المتعارضة (مثل «both modified»)
عدّل الملف
احتفظ بما تريد واحذف أسطر >>>>>>
<<<<<< =====
- `git add <file>`
تعليم الملف كمحلول
- `git merge --continue`
الإنهاء (أو: git rebase --continue / cherry-pick --continue)
- `git merge --abort`
التراجع والعودة إلى ما قبل (rebase --abort, ...)

التنظيف

- `git branch -d <branch>`
حذف فرع مدموج (يرفض إن لم يكن مدموجًا)
- `git branch -D <branch>`
الحذف حتى إن لم يكن مدموجًا ⚠️
- `git push origin --delete <branch>`
يحذف الفرع على المستودع البعيد (وتذهب معه ال commits غير المدموجة) ⚠️
- `git fetch --prune`
يحذف أيضًا نسخ ./origin... للفروع المحذوفة على المستودع البعيد

الوسوم (TAGS)

- `git tag`
عرض الوسوم
- `git tag -a v1.0 -m "Release 1.0"`
وسم موصوف على ال commit الحالي
- `git push origin v1.0`
الوسوم لا تُدفع تلقائيًا
- `git push --follow-tags`
دفع ال commits مع وسومها الموصوفة

قراءة التاريخ

- `git log --oneline`
تاريخ مختصر للفرع الحالي
- `git log -p <file>`
كل تغيير أُجري على ملف واحد
- `git log main..feature`
ال commits الموجودة في feature وليست في main
- `git blame <file>`
من غيّر كل سطر آخر مرة

أريد أن...	تشغيل	من المفيد أن تعرف
تجاهل تعديلاتي على ملف	<code>git restore <file></code>	⚠️ التعديلات غير المجهّزة تضيع نهائيًا
إلغاء تجهيز ملف (مع الاحتفاظ بتعديلي)	<code>git restore --staged <file></code>	✅ تتغيّر منطقة التجهيز فقط
تصحیح رسالة آخر commit أو إضافة ملف منسي	<code>git add <file> && git commit --amend</code>	⚠️ جديد — افعلها قبل الدفع hash
التراجع عن آخر commit مع إبقاء التغييرات مجهّزة	<code>git reset --soft HEAD~1</code>	⚠️ يبقى عملك مجهّزًا وجاهزًا لـ commit جديد — ليس بعد الدفع
التراجع عن آخر commit مع إبقاء التغييرات في الملفات	<code>git reset HEAD~1</code>	⚠️ mixed-- افتراضيًا: غير مجهّز لكنه على القرص — ليس بعد الدفع
التراجع عن commit دُفع بالفعل	<code>git revert <commit></code>	✅ يضيف commit جديدًا بلغيه — آمن على الفروع المشتركة
وضع تغييراتي جانبًا مؤقتًا	<code>git stash push -m "wip"</code>	✅ استعدها بـ <code>git stash pop</code> — الملفات الجديدة غير المتتبّعة تحتاج -u
تجاهل آخر commit وتغييراته	<code>git reset --hard HEAD~1</code>	⚠️ تضيع تغييراته وتعديلاتك غير المحفوظة على الملفات المتتبّعة (ما زال git reflog يجد الـ commit)
العثور على commit «مفقود» (بعد reset أو rebase أو حذف فرع)	<code>git reflog</code>	✅ كل مكان مرّ به HEAD — ثم git <branch rescue <hash>
إيقاف merge / rebase / cherry-pick جارٍ	<code>git merge --abort</code>	✅ أو --cherry-pick --abort, rebase --abort
استبدال الفرع البعيد بعد إعادة كتابة التاريخ	<code>git push --force-with-lease</code>	⚠️ يرفض إذا تغيّر الفرع البعيد منذ آخر fetch قمت به

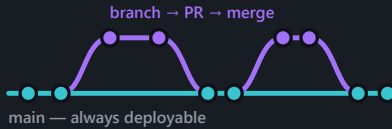
STASH

- `git stash`
وضع التغييرات غير المسجّلة جانبًا
- `git stash -u`
تضمين الملفات غير المتتبّعة (الجديدة)
- `git stash list`
كل الـ stashes، الأحدث = `stash@{0}`
- `git stash pop`
يعيد تطبيق الأحدث ويحذفه (يبقى إذا حدث تعارض)
- `git stash apply stash@{1}`
إعادة تطبيق واحد مع إبقائه في القائمة

ثلاث درجات — RESET

- `git reset --soft <commit>`
⚠️ ينقل الفرع؛ التجهيز والملفات دون تغيير
- `git reset <commit>`
⚠️ ينقل الفرع ويعيد ضبط التجهيز؛ الملفات دون تغيير
- `git reset --hard <commit>`
⚠️ ينقل الفرع والتجهيز والملفات أيضًا

GITHUB FLOW



```
git switch main &&
git pull
```

ابدأ من main محدّث

```
git switch -c
<feature>
```

فرع قصير الأمد لكل تغيير

```
git commit -am "..."
```

```
&& git push -u
origin <feature>
```

commit ثم push ثم فتح pull request

ادمج الـ pull request على GitHub

بعد المراجعة؛ ثم احذف الفرع

```
git switch main &&
git pull --prune
```

العودة إلى main؛ ويحذف أيضًا نسخ .../origin للفروع المحذوفة

GIT FLOW



```
git checkout -b
feature/x develop
```

الميزات تبدأ من develop

```
git checkout develop
&& git merge --no-ff
feature/x
```

إنهاء ميزة في develop

```
git checkout -b
release-1.1 develop
```

تجميد إصدار

```
git checkout main &&
git merge --no-ff
release-1.1
```

الإصدار إلى main...

```
git tag -a 1.1 -m
"Release 1.1"
```

وضع وسم عليه، ثم دمجه مرة أخرى في develop

TRUNK-BASED



```
git commit -am
"small change"
```

صغيرة ومكتملة commits على main

```
git pull --rebase
```

يعيد تطبيق ما لديك فوق ما لدى الآخرين (hash جديد)

```
git add <file> &&
git rebase
--continue
```

بعد إصلاح تعارض: علّمه كمحلول وتابع

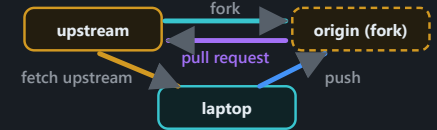
```
git push
```

ادمج مرة واحدة يوميًا على الأقل

أطلق العمل غير المكتمل وهو معطل

خلف flag feature، مثل NEW_CHECKOUT = false

FORKING



```
git remote add
upstream
```

```
<original-url>
```

مرة واحدة: المستودع الأصلي كمستودع يعيد ثان

```
git fetch upstream
&& git merge
upstream/main
```

زامن main الخاص بك مع الأصلي

```
git push
```

حدّث الـ fork الخاص بك (origin)

```
git switch -c <fix>
... git push -u origin
<fix>
```

اعمل على فرع في الـ fork الخاص بك

افتح pull request نحو الأصل

المشرفون يراجعون ويدمجون