git config --global user.email "elkannas9@gmail.com"
git config --global user.name "Ahmed Saif"
git config --global init.defaultBranch main
git config --global --list (بيتأكد من الإعدادات اللي فوق)
git --version
git update-git-for-windows
mkdir project
cd path | cd / (root of all)
cd .. (parent directory) (~ means in Home UserFolder)
rm -rf FolderName
alias hi='echo Hello World' (create command hi)
unalias hi (remove command)
git init
initialize an empty git repository to store changes timeline in the project folder. (.git hidden)
rm -rf .git (Untracked folder again)
touch names.txt
git add names.txt
git add .
بطلع ملف محدد أو كل الملفات على stage المسرح عشان تكون جاهزة يتاخد لها snapshots (staged) يعني يتعملها commit تثبيت للتعديلات اللي حصلت فيها
git status
بيعرض حالة التغييرات staged or unstaged commited or not
git commit -m "message"
git show commit-hash (أول سبع حروف من الهاش)
git log
بيعرض سجل ال commits من الأحدث للأقدم وكل commit مبني على اللي قبله.
Head بيشاور على ال branch اللي هو تسلسل من ال commits وبدوره بيشاور على آخر commit يعني pointer
Head => Branch (main)الفرع الرئيسي
vi names.txt
press i for insert mode

press esq for commands mode
press :wq to write and quit
ls -a (all)
show content of directory include hidden files
لو عملت تعديلات خطأ بس لسه معملتش commit تقدر تسترجع عن طريق:
git restore style.css (بيشيل آخر تعديل ع الملف وملوش علاقة بستيجنج)
unstage file but keep changes:
git restore --staged style.css
لو عملت تعديلات خطأ وللأسف عملت commit ممكن تسيب التعديلات زي ما هي
وتحذف ال commit الأخير من السجل والتعديلات هتفضل في stage area عن طريق:
بتستخدمه لما تحب تعدل في الرسالة فقط
HEAD~1 معناها الكومت قبل الأخير
git reset --soft HEAD~1
نفس اللي فوق بس التعديلات هتنزل من ال stage وهتضطر تعمل add and commit عن طريق: ودة الوضع الافتراضي --mixed
git reset HEAD~1
تحذف ال commit الأخير وكمان تحذف التعديلات بتاعته بشكل نهائي عن طريق:
git reset --hard HEAD~1
إعادة مؤشر الفرع الحالي إلى commit معين فالتعديلات هيفضل أثرها في ال directory ولكن ال commits بتاعتها هتحذف وتصبح التغييرات unstaged عن طريق:
git reset commit-hash
في حال إنك عملت push على GitHub اعمل commit جديد بعكس تعديلات ال commit القديم برسالة افتراضية توضح كدا عن طريق:
git revert a1b2c3d (old commit-hash)

---

stash مخبأ بنفس طريقة عمل ستاك آخر شيء تخفيه أول شيء تستعيده وهو تخزين مؤقت للتغييرات اللي مش جاهز تعملها commit دلوقت.
كل التغييرات سواء staged or not بتروح فيه عن طريق الأمر:

والمشروع بيرجع لحالة نظيفة عند آخر commit قبل التعديلات
git stash
ترجع التغييرات على البروجكت وبعدين تعملها commit لو عايز عن طريق:
آخر تغيير اتعمل هو أول تغيير بيرجع
git stash pop
git stash clear (changes are gone)
git update-ref -d HEAD
لما يكون عندي commit واحد بس مفيش قبله هعمل تحديث للمرجع d stands for delete HEAD
هيحذف الكومت وهتحتاج تعمل add and commit من جديد

Connecting remote repository to local repository

الطريقة الأولى: البدء من مشروع موجود على جهازك

هذه الطريقة تستخدمها عندما يكون لديك مشروع قائم بالفعل على جهازك وتريد أن تبدأ في تتبعه باستخدام Git ورفعه على GitHub

1. يكون لديك مجلد المشروع على جهازك
2. تدخل إلى المجلد وتنفذ git init لتخبر git بتتبع هذا المجلد
3. تذهب إلى GitHub وتنشئ مستودعًا فارغًا
4. تستخدم الأمر git remote add origin repo_URL لتربط المشروع المحلي بالمستودع الفارغ
5. تقوم بعمل add, commit & push لرفع الملفات

الطريقة الثانية: البدء من مستودع جديد على GitHub

هذه الطريقة تستخدمها عندما تبدأ مشروعًا جديدًا من الصفر أو عندما تريد العمل على مشروع موجود بالفعل على GitHub

1. تذهب إلى GitHub وتنشئ مستودع جديد به ملف README.md عادي
2. تنسخ ال URL الخاص بالمستودع
3. تستخدم الأمر git clone repo_URL
4. الأمر دة بيعمل كل شيء تلقائي بينشئ مجلد المشروع على الجهاز git init, git remote add & pull
5. pull يعني بيسحب الملفات الموجودة

بعد كدا تقدر تعمل تعديلاتك في الفولدر الجديد وتعمل commit & push عادي

git remote -v

لينكات ال repos المرتبطة بالفولدر دة

Repos that exist in your own account called origin

git push origin main | git push -u origin main

-u --set-upstream (Git remembers where to push and pull in the future git push / pull)

git pull origin main

git log --oneline (commits in one line)

Branches

branch مسار جديد كأن معك نسخة طبق الأصل من المشروع في لحظة معينة يستخدم لاختبار ميزة جديدة دون التأثير على ال main وإذا نجحت الميزة ننتقل لل main ونعمل merge

git branch (show all branches)
git branch feature

ينشئ branch فقط

git checkout -b feature
-b branch

ينشئ branch وينتقل له مباشرة.

git checkout feature (ينتقل فقط)
HEAD moves to point to the last commit in feature branch
(HEAD => feature*)
git checkout main

git merge feature

دمج للميزة مع branch main

git branch -d feature

حذف للنسخة التجريبية لم نعد بحاجة لها

merge commit كوميت له أبوان آخر two commits وهو مرجع لعملية الدمج.

git log --oneline –merges

فقط الكوميتس الخاصة بعملية الدمج اللي هي عناوين الميزات المضافة.

git show a1b2c3d (Merge Commit Hash)

مش هيكتفي بعرض معلومات كوميت الدمج بل هيعرض إجمالي كل التغييرات التي جاءت من الفرع المدموج.

```
git diff main feature
```

بيعرض كل التغييرات في فرع الميزة الجديدة مقارنة بالفرع الرئيسي (ميزة لم تدمج بعد).

```
git log main a1b2c3d
```

بيعرض كل الكوميتس من أول كوميت الدمج لحد آخر كوميت في الفرع بدون الكوميتس اللي كانت موجودة أصلاً في ال main

---

Fork يعني نسخة متفرعة بتاخد نسخة كاملة من مشروع شخص آخر على Github في حسابك الخاص وتعمل clone لل repo على جهازك تنفذ تعديلاتك وتعمل commit و push وتقوم بفتح طلب سحب pull request لمالك المشروع الأصلي.

upstream يعني لينك الريبو اللي أنت عملتله fork عندك

```
git remote add upstream URL
```

```
git remote -v
```

```
git fetch upstream
```

تجلب التحديثات من upstream للفرع اللي واقف فيه main مثلاً ويخزنها محليًا داخل .get تحت مسمى upstream/main وبعدين تشوف الفرق بين الفرع والفرع اللي عندك ولو عايز تدمجه.

```
git diff main upstream/main
```

```
git merge upstream/main
```

لو عايز تحدث نسختك على :GitHub

```
git push origin main
```

---

pull request يعني بتشتغل على ميزة في branch جديد اعمل تعديلاتك وال commits بتاعتك واعمل push على GitHub ولما تروح لصفحة المستودع هيطلب منك تعمل pull request فتملى البيانات والوصف هيروح

لأصحاب ال upstream يراجعوه وبعدها يعملوا merge في ال main

كل فرع (branch) يقابله طلب سحب (pull request) واحد فقط وإذا طلب المراجعون تعديلات تقوم بتعديلها وتقديمها ضمن نفس الطلب.
أي كوميت جديد تقوم بدفعه (push) إلى نفس الفرع سيظهر تلقائيًا في طلب السحب المفتوح.

لو سجل الكوميتس على جهازك المحلي مختلف عن الريبو لأنك عدلت فيه أو مسحت كوميتس وعايز تجبر GitHub يمسح السجل اللي عنده ويستخدم نسخة طبق الأصل من السجل الموجود على جهازك لنفس الفرع يبقى تستخدم الأمر:

```
git push --force
```

push على فرع هو لحفظ التقدم الشخصي بينما pull request لدمج ميزة فرعية (الفرع) بعد اكتمالها في الفرع الرئيسي main على GitHub سواء مشروع شخصي أو مشروع شخص آخر upstream.

دمج سلسلة من الكوميتس الصغيرة المتتالية في كوميت واحد نظيف وتغييرات الكود تظل كما هي.

```
git rebase -i commitHash
```

```
git rebase -i HEAD~5
```

rebase إعادة تطبيق ال commits في base commit
-i interactive- تفاعلي
HEAD~1 مرجع للكوميت السابق وهكذا... HEAD~5 مرجع للكوميت الخامس
HEAD~5 بداية النطاق من ال commit الأخير اللي ال HEAD واقف عندها لحد قبل ال commit الخامس.
commitHash بداية النطاق من بعد ال commit هذا وصولًا لل Head
في المحرر:
pick معناها احتفظ بهذا الكوميت كما هو
s squash اسحق / ادمج هذا الكوميت في الكوميت الذي قبله
الكوميتس المحددة ب s تدمج في أول pick فوقها

---

https://learngitbranching.js.org
إذا كان عندك فرع وفشلت الميزة الخاصة به فبمجرد تنفيذ الأمر
git checkout main
تعود للفرع main ويعود الكود كما كان عليه في الأصل

```
git branch -d feature
```

تحذف الفرع والكوميتس الخاصة به

Merge Conflicts

سيلاحظ Git أن نفس السطر الذي انطلقتما منه في ال main الأصل المشترك قد تم تعديله بطريقتين مختلفتين في الفرعين وعند الدمج لن يتمكن من اتخاذ قرار تلقائي، لذلك سيوقف العملية ويخبرك بوجود تعارض.