Git + Github

إعداد : عمار قصاب

1

intro

Git : و هو عبارة عن نظام يتحكم بإصدارات البرمجية خاصتيك التي تطورها مع الوقت .

Github : و هو موقع يتم رفع عليه المشاريع من قبل الشركات و الأفراد مثله مثل Gitlab / bitbucket حيث أن Github يسهل استخدام Git و يسهل التحكم و التعديل على الأصدارات البرمجهة المرمزة .

يتم استخدم Git ل :

١. تتبع التغييرات رمز البرمجي
٢. تتبع من قام بالتغييرات
٣. التعاون بالترميز بين الفريق

لماذا يجب أن أتعلم Git :

١. تساعد المبرمجين على الحصول على التعديلات الخاصة بالمشروع التي قام أفراد الفريق
٢. امكانية معرفة من قام بتعديل المشروع وتعديلها أو إلغائها
٣. التعاون بين المبرمجين لحل المشكلة البرمجية في الرمز
٤. إمكانية إضافة مكتبات جديدة و الوصول إلها من قبل الفريق
٥. المقارنة بين التعديلات و أخذ التعديل الصحيح
٦. عند رفع التعديلات يمكن إضافة معها وصف للإصدار

ماذا يفعل ال git :

١. Repositories (المستودعات) : إدارة المشاريع مع المستودعات حيث كل مشروع له مستودع خاص فيه (repo)
٢. Branch and Merge (فرع و الدمج) : حيث الفرع branch هو فرع من المستودع الرئيسي حيث الفرع والدمج تستخدم للسماح بالعمل على أجزاء وإصدارات مختلفة من المشروع
٣. Local repo : و هو المستودع الذي تعمل عليه في الوقت الحاضر
٤. Remote repo : و هو مستودع الشركة سواء كان على Github / Gitlab / bitbucket / server
٥. Staging and Committing (التدريج والالتزام) : تحكم في التغييرات وتتبعها باستخدام التدريج والالتزام حيث هو تعديل محلي لم يتم رفعه بعد للمستودع
٦. Clone (استنساخ) : استنساخ مشروع للعمل على نسخة محلية
٧. Push (دفع) : بعد إنهاء من التعديل على المشروع يتم دفع التعديلات للمستودع الرئيسي
٨. Pull (سحب) : اسحب أحدث إصدار من المشروع إلى نسخة محلية
٩. Pull request : وهي دفع التعديلات للأدمن ليتحقق منها وبعدها يدفها للمستودع الرئيسي

ملاحظات :

١. يجب أن يكون لكل مشروع مستودع خاص فيه repo من أجل التنظيم
٢. يجب خلق فرع Branch لكل تعديل وبعد إنهاء التعديل و التأكد منه يتم دفعه لمستودع الرئيسي وبعدها تحذف الفرع اذا أردت
٣. أنت لست بحاجة لتكون متصل بالمستودع الرئيسي خلال العمل بل يمكنك الأتصال به بعد إنهاء العمل ودفع التعديلات
٤. أي شخص يمكنه الدفع و السحب أذا كان له الصلاحيات من الأدمن أو أذا أراد الأدمن التشيك على الترميز أولا

2

التنصيب

١. نذهب إلى موقع https://git-scm.com ونقوم بتحميل ال git وتنصيبه
٢. نفتح على سطح المكتب ثم نضغط بالزر الأيمن للماوس ثم نفتح Git bash وهي كومند لاين خاص بال git
٣. ثم نكتب git –version ونضغط انتر فيظهر لدينا أصدار ال git وهكذا أصبح لدينا git على الحاسب
٤. الأن نسجيل في github
٥. بعد التسجيل سنقوم بإنشاء مستودع لمشروع جديد
٦. لذلك ندخل على repositories ثم نضغط new ثم ندخل أسم المشروع ثم ندخل الوصف للمشروع ثم تختار هل هو عام أو خاص
٧. ثم نعلم على Add a README file التي تضيف تعليمات عن المشروع
٨. ثم نعلم على Add .gitignore اختر الملفات التي لا تريد ترفعها من الحاسب
٩. ثم نعلم على Choose a license يخبر الترخيص الآخرين بما يمكنهم وما لا يمكنهم فعله باستخدام التعليمات البرمجية الخاصة بك
١٠. هكذا تم انشاء مستودع للمشروع جديد

لإضافة تعليمات README للمشروع نكتب التعليمات بسينتاكس md لتعرف عليها Basic writing and formatting syntax

عبر الرابط

```
https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax
```

الأن نأخذ clone للمشروع لتعديل عليه لذلك ندخل على المشروع من الأعلى ثم نضغط على زر code ثم ننسخ رابط clone وهو رابط المستودع

الأن ننشأ مجلد للمشروع ثم ندخل عليه ثم نفتح git bash ونكتب git clone وبعدها الرابط

```
git clone https://link.com
```

هكذا تم تحميل المشروع من github وهو داخل المجلد الخاص فيه و هو جاهز لتعديل لكن قبل التعديل

يجب وضع معلوماتي في git :

وذلك من خلال أرسال أسم المستخدم و الإيميل

```
git config --global user.name "w3schools-test"
git config --global user.email “test@w3schools.com”

لعرض القيمة المحفوظة

git config --global user.name
git config --global user.email

لحذف القيم نكتب

git config –global --unset user.name
git config –global --unset user.email

لمعرفة المعلومات التي يمكنني إدخالها أكتب الأمر

git config –list
git help config

للخروج من list نكتب

q
```

هكذا تحفظ المعلومات في git من أجل إذا رفعنا تعديل أو مشروع جديد لل github تنحفظ بأني أنا قمت بالتعديل

لإختصار الأوامر أو صنع أسم جديد للأمر :

```
git config --global alias.newName oldName

مثال

git config --global alias.st status
>> git st

مثال لأمر من كلمتين

git config --global alias.cm “commit -m”
>> git cm
```

ملاحظة : لا يتم إلغاء الأمر القديم

بعد الدخول إلى المشروع نلاحظ وجود ملف التعليمات README قد تم تحميله

الأن ننشأ بجانبه ملفات المشروع index.html / js / css ونقوم بفتح git bash في المجلد وبعدها نكتب بداخله . code من أجل فتح الفجول استديو وبعد فتح الفجول استديو نلاحظ تلون الملفات بالأخضر أي هي ملفات جديدة أو معدل عليها في المشروع

لنتعرف على الأمر git status الذي يقوم بالأتصال بالمستودع و المقارنة بين الملفات المحلية و ملفات المستودع ليظهر الملفات المعدل عليها أو الجديدة

```
git status
```

ويظهر لدينا من خلال الأمر السابق حالتين :

١. Tracked : الملفات الموجودة محليا تمت إضافتها لمستودع
٢. Untracked : الملفات الموجودة محليا لم يتم إضافتها لمستودع ويحدد ما هي

4

Staging Environment

مراحل رفع المشروع وتطويره :

١. Local : working directory : منطقة العمل المحلية التي يتم تطوير المشروع فيها ومنها يتم رفع المشروع على staging area من خلال الأمر git add
٢. Local : staging area : وهي منطقة التجهيز المشروع لرفعه على local repo ويتم رفعه من خلال الأمر git commit
٣. Local : local repo : و هو مستودع محلي على الجهاز منه يتم رفع المشروع على remote repo من خلال الأمر git push
٤. كل مما سبق كان محلى فقط أي أن المشروع ما يزال على الحاسب
٥. Remote : remote repo : وهو المستودع الخاص بالمشروع على github
٦. يتم إرجاع المشروع من remote repo إلى local repo من خلال الأمر git fetch
٧. يتم إرجاع المشروع من local repo إلى working directory من خلال الأمر git checkout أو git merge

الأمر git add :

يرفع الملفات من منطقة العمل المحلية working directory إلى بيئة التجهيز المحلية staging area

```
git add file : لرفع ملف أو أكثر من الملفات الجديدة أو المعدلة
git add --all : لرفع كل الملفات الجديدة أو المعدلة مرة واحدة
git add . : لرفع كل الملفات الجديدة أو المعدلة مرة واحدة
```

نلاحظ أذا كتبنا الأمر git status أن الملفات اختفت من منطقة working directory و أصبحت ضمن staging area

الأمر git reset head file :

يقوم بإرجاع الملفات من staging area إلى working directory

ملاحظة : يتم إرجاع الملف بكتابة اسمه بدل file ويمكن كتابه بجانبه أسم ملف أخر لإرجاعه

```
git reset head file1 file2
```

5

local repo

الأمر git commit -m “des” :

يرفع الملفات من بيئة التجهيز المحلية staging area إلى المستودع المحلي local repo

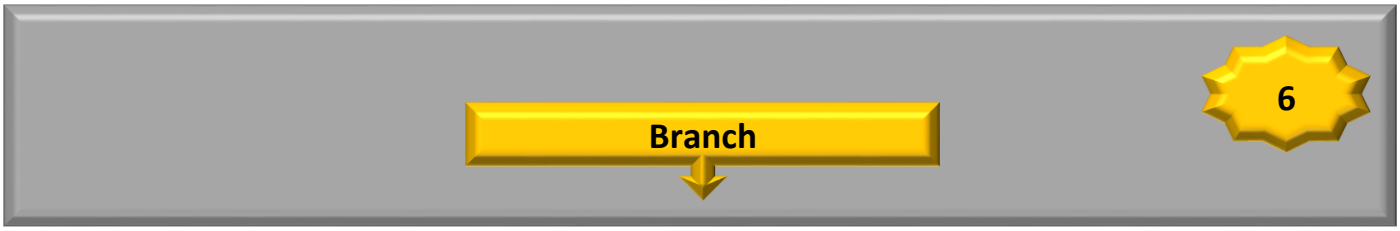
حيث نكتب وصف التعديلات الجديدة بين علامتي التنصيص

```
git commit -m “new project”
```

لعرض حالة المستودع المحلي و الرئيسي ولمعرفة commit المعمولة :

حيث كل commit له id

```
git log
```



Branch : هو فرع من فروع المشروع في local repo حيث يمكن أن يكون للمشروع الرئيسي في remote repo عدة فروع قيد التطوير في local repo حيث عند اكتمال العمل يمكن دمج فرع مع المشروع الرئيسي ويكون لكل فرع أسم خاص به لتميز بين الأفرع

لمعرفة أسم الفرع أو الجزء :

```
git branch
```

هكذا يتم معرفة أسم الفرع

لإنشاء فرع جديد :

```
git branch namebranch
```

لنقوم بالقلب إلى الفرع الجديد من أجل أن يأخذ التعديلات الجديدة التي سنقوم بها حيث نكتب اسم الفرع الجديد :

```
git checkout namebranch
```

هكذا عندما نقوم بإجراء التعديلات نقوم بأتباع الخطوات السابقة لرفع الفرع

للإرجاع إلى الفرع الأصلي :

```
git checkout oldnamebranch
```

لإنشاء فرع طوارئ :

يستخدم لإنشاء فرع طوارئ لإصلاح المشاكل أو لتعديل على المشروع ولكن بعيد عن الفروع الأخرى وبعد الأصلاح نقوم بدمج فرع الطوارئ مع الفرع الأصلي الذي سنرفعه على github

```
git checkout -b namebranchfix
```

هكذا تم إنشاء فرع طوارئ جديد وتغيرنا إليه

الأن نقوم بالتعديل ورفع التعديلات على local repo بعد التعديل نقلب على الفرع الرئيسي من خلال الأمر git chechout branch

و ذلك من أجل دمج فرع الطوارئ مع الفرع الرئيسي

دمج الفرع الطوارئ مع الفرع الرئيسي :

نقلب على الفرع الرئيسي من خلال الأمر git chechout branch و نستخدم الأمر مع كتابة أسم فرع الطوارئ من أجل الدمج

```
git merge namebranchfix
```

بعد الدمج لم نعد بحاجة لفرع الطوارئ لذلك نقوم بحذفه

لحذف فرع الطوارئ أو أي فرع نريد :

```
لا يحذف الأمر الفرع غير مرفوع
git branch -d namebranch
يحذف حتى لو لم يتم رفع الفرع
git branch -D namebranch
```

لإعادة تسمية أي فرع :

ندخل عليه ثم نضغط الأمر

```
git branch -m newName
```

7

remote repo

لرفع الفرع المختار (نختاره من خلال git checkout name) من local repo إلى (github) remote repo

لرفع الفرع إلى (github) remote repo :

```
git push origin namebranch
```

حيث بعد كتابة الأمر يطلب تسجيل الدخول إلى github لأول مرة فقط وهكذا تم رفع الفرع أو المشروع على github أذهب ومتع عيناك

لاحظ الرسالة التي بجنب كل ملف قمت برفعه هي نفسها وصف التعديل في "git commit -m "new project

العمل على تطوير المستودعات :

أدخل إلى المستودع الذي أريد حيث أنا مو بالفريق الذي أنشئ المستودع ولكن سوف أعدل على المستودع و أبعت التعديلات أما أن يوافقو عليها أو يرفضوها حيث أضغط على add file ثم create new file هكذا أصبح المستودع في مستودعاتي

حيث الشخص الذي أنشئ المستودع يمكنه أن يشاهد من أخذ المستودع لتطويره أو لخزنه من خلال دخوله لمستودع ثم ضغطه لل fork

العمل كفريق :

العمل مع فريق حيث يجب أولا على الشخص الذي أنشئ المستودع أن يضمني للفريق وذلك من خلال

يدخل مسؤول الذي أنشئ المستودع إلى المستودع ثم يضغط على settings ثم الضغط على collaborators ثم يضغط على addpeople ثم يكتب أسمه في البحث ثم يضغط على add name to this repository هكذا تم إرسال طلب إنضمام لشخص للفريق عندما يوافق الشخص من خلال دخوله على الإيميل والموافقة سيصبح في الفريق

بعد الأنضمام إلى الفريق لتعديل على المشروع أدخل إلى المستودع الذي أصبحت جزءا من فريقه ثم

أضغط على add file ثم create new file ثم أكتب أسم المستودع الذي سوف أعدله newrepo

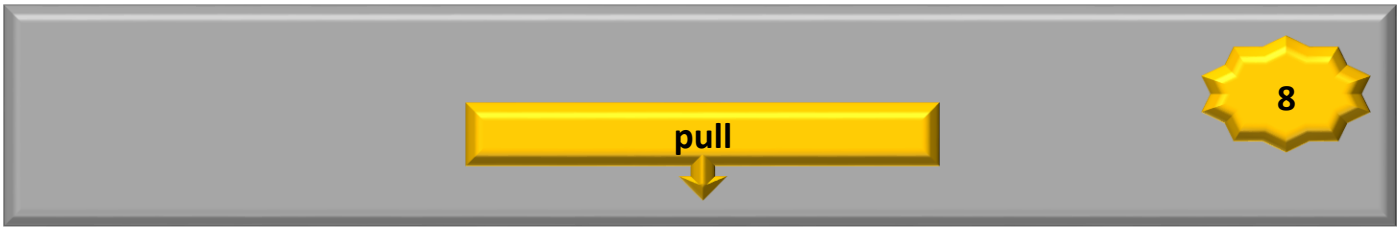
ثم ادخل في commit و أدخل create newrepo ثم لدي حالتين أما أن أعدل على البرناش الرئيسي أو أنشأ برانش فرعي و أطلب pull requests من أجل أن يوافق منشأ المشروع على التعديلات أو لا وعدم التعديل على المشروع مباشرة

في هذه الحالة سوف أعدل على البرانش الرئيسي اضغط على commit new file هكذا تم إنشاء ملف جديد

لسحب المشروع على الحاسب لتعديل عليه انسخ رابط clone ثم أفتح git bash ثم اضغط git clone https://

هكذا أصبح لدي المشروع على الحاسب ويمكنني تعديله

سنشرح النوع الثاني البرانش الفرعي لاحقا



عندما يكون لدينا مستودع خاص بمشروع على github remote repo و تم تعديله من قبل أحد أعضاء الفريق والمشروع موجود لدي على local repo وعندما نضغط على الأمر git status نلاحظ أن المشروع لم يحصل له تغيير لذلك يجب علينا سحب التعديلات إلى local repo من خلال الأمر

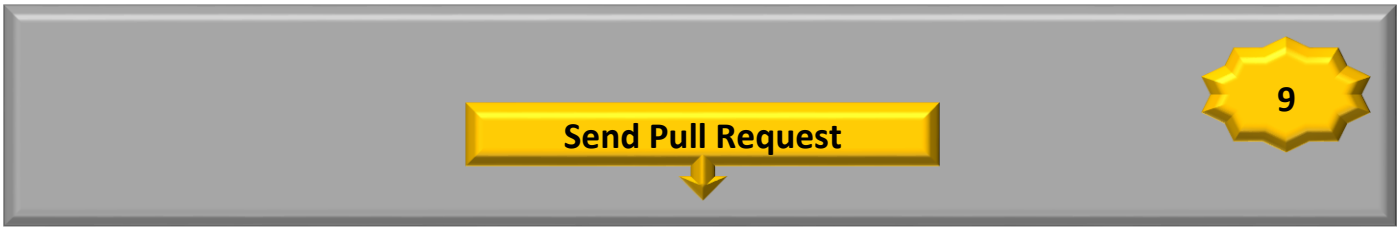
```
git pull origin
```

هكذا أصبحت التعديلات موجودة لدينا

ملاحظة : الأمر git pull origin يفعل أمريين التالين

```
git fetch origin // يقوم برفع الملفات الجديدة

git merge origin // يقوم بدمج الملفات الجديدة مع القديمة
```



العمل على تطوير المستودعات :

أدخل إلى المستودع الذي أريد حيث أنا مو بالفريق الذي أنشئ المستودع ولكن سوف أعدل على المستودع و أبعت التعديلات أما أن يوافقو عليها أو يرفضوها حيث أضغط fork فورا أو على add file ثم create new file هكذا أصبح المستودع في مستودعاتي

حيث الشخص الذي أنشئ المستودع يمكنه أن يشاهد من أخذ المستودع لتطويره أو لخزنه من خلال دخوله لمستودع ثم ضغطه لل fork

ثم أعدل على المشروع مثل سحبه على على الحاسب من خلال clone وإجراء التعديلات أو نعدل على README.md

الأن لنرسل التعديلات أما أن يوافقوا عليها أو يرفضوها ندخل إلى مستودعنا الذي قمنا بالتعديل عليه ثم نضغط على تالت خيار وهو pull request ثم نضغط على new pull request ثم create pull request ثم نكتب أسم التعديل و وصفه ثم نضغط على create pull request الأن نظهر لدينا الحالة open ويمكننا وضع comment ليراها مسؤول المستودع

الأن يدخل مسؤول المستودع إلى المستودع ثم يضغط على ثالث خيار pull request فيظهر لديه التعديل الذي قمنا فيه اذا وافق على التعديل يضغط على merge pull request ثم بضغط على confirm merge فتتغير لدينا الحالة من open إلى merged وهكذا أصبحت التعديلات في المستودع الرئيسي عندها يظهر لدينا أشعار انو التعديل أصبح merged يمكننا الأن حذف مستودعنا

ملاحظة : كنا نعدل على الفرع الرئيسي دوما يمكننا إنشاء فرع ثانوي ونعدل عليه ثم ندمجه مع الفرع الرئيسي لدينا عن طريق pull request لنفسنا و نوافق عليه ليصبح merged و من ثم نرسل pull request للمسؤول

10

Security SSH kay

ssh وهو برتوكول شبكة الأمنة التي تستخدم للوصول للحساب github عن بعد من حاسب أخر

أولا يجب إنشاء kay من ال git وبعدها نحفظه في github

لإنشاء SSH Kay :

نكتب الأمر مع الايميل

```
ssh-keygen -t rsa -b 4096 -C "test@w3schools.com"
```

بعد كتابة الأمر يعرض لدينا مسار ويطلب كتابة أسم الملف الذي سوف يحفظ فيه kay نضغط انتر ولا نكتب شيء ثم نكتب كلمة سر للحفاظ على المفتاح ونعيد كتابتها مرة ثانية وهكذا تم إنشاء kay و يعطينا مسار حفظ kay ويكون أخره .pub ننسخه من أجل الوصول للمفتاح حيث ننسخه من بعد اسم الحاسب

لأستخراج kay :

نكتب الأمر مع مسار الملف الذي نسخناه من بعد أسم الحاسب حيث الأمر cat يعطينا محتوى أي ملف

```
cat ~/.ssh/id_rsa.pub
```

عندها يظهرkay نقوم بنسخة من أجل لصقة في github ندخل لحساب github من أجل لصق المفتاح من أجل الوصول للحساب لذلك

ندخل إلى settings ثم SSH ثم نضغط على new ssh key ثم ندخل title ثم نلصق المفتاح ونضغط على add new key

هكذا تم إنشاء المفتاح للحساب ويمكن إنشاء أكثر من مفتاح

لدخول للحساب من حاسب بعيد :

نكتب الأمر

```
ssh -T git@github.com
```

بعدها ندخل yes لإستمرار الأجراء ثم ندخل كلمة السر التي كتبناها عند إنشاء المفتاح وهكذا تم الأتصال مع الحساب

11

create repo pc

كنا قد تعلمنا كيف ننشأ مستودع من داخل github وبعدها نأخذ clone ومن ثم نعدل المشروع ونرفع التعديلات الأن سنتعلم إنشاء مستودع من داخل الحاسب ونقوم برفعه على github حيث

أولا ننشئ مجلد للمشروع وندخل إلى داخله ونفتح git bash

ثانية لإنشاء المستودع نضغط الأمر

```
git init
```

هكذا تم إنشاء مستودع

للنشئ أي ملف جديد مثل index.html ونقوم برفعه على local repo من خلال الأوامر التي تعلمناها سابقا

بعدها ندخل إلى github و ننشئ مستودع جديد بنفس أسم المجلد الموجود على الحاسب ولا نضيف شيء للمستودع ولا حتى وصف ونضغط create repo ثم نضغط على SSH ثم ننسخ الأمر الذي أوله git remote وبعدها نلصقه في git مثل

```
git remote add origin git@github.com:ammarqassab/project_1.git
```

ثم ننسخ و نضغط الأمر الأخير في github مع ملاحظة أن أسم الفرع يجب أن يكون مطابق في نهاية الأمر

```
git push -u origin master
```

بعدها نكتب كلمة سر SSH

وظيفة الأمر هو رفع الملفات من local repo حيث قمنا بتجهيزها لرفع إلى remote repo مع العلم أن u- ترمز أنه أولا يعمل pull في حال كان هناك تغيرات من الفريق ثم يدمجها ويعمل push

12

stash

وهو مخزن محلي في git يتم رفع اليه الملفات المعدلة حيث أولا نرفعهم على منطقة staging area من خلال الأمر git add

ثم نضغط الأمر مع كتابة وصف الملف

لإخفاء الملفات :

```
git stash save “des”
```

هكذا أصبحوا في المخزن

بمعرفة الملفات المخزنة :

```
git stash list
```

لإخراج أخر الملفات التي تمت إضافتها إلى المخزن :

```
git stash pop
```

لإخراج الملفات التي أريد من المخزن نضع رقم الملف :

حيث نحصل على رقم الملف المخزن من خلال الأمر git stash list

```
git stash pop stash@{0}
```

لإخراج أخر الملفات التي تمت إضافتها إلى المخزن مع الأحتفاظ فيها في المخزن أي اخذ نسخة :

```
git stash apply
```

لإخراج الملفات التي نريد مع الأحتفاظ فيها في المخزن أي اخذ نسخة نضع رقم الملف :

حيث نحصل على رقم الملف المخزن من خلال الأمر git stash list

```
git stash apply stash@{0}
```

لحذف أخر ملف تمت إضافته للمستودع :

```
git stash drop
```

لحذف الملف الذي نريد من الملفات التي تمت إضافته للمستودع نكتب رقم الملف:

حيث نحصل على رقم الملف المخزن من خلال الأمر git stash list

```
git stash drop stash@{0]
```

لعرض محتوى المخزن من الملفات الذي تم إضافته أخر شيء :

```
git stash show
```

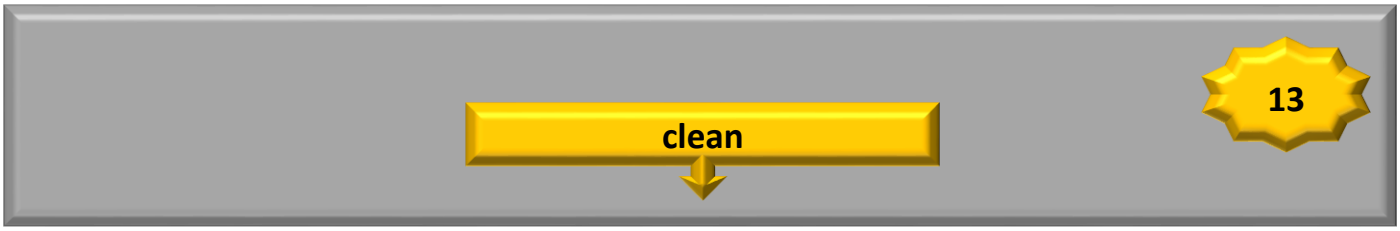
لعرض محتوى أي مخزن من الملفات نضع رقم المخزن :

حيث نحصل على رقم الملف المخزن من خلال الأمر git stash list

```
git stash show stash@{0]
```

لحذف كل المخازن دفعة واحدة ونولع فيها بانزين :

```
git stash clear
```



working directory في منطقة العمل هناك ملفات لم ترفع بعد إلى staging area ونرغب بحذف هذه الملفات

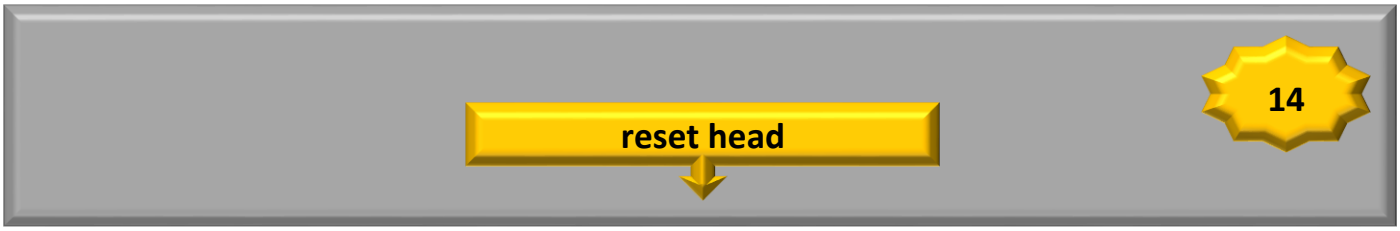
لمعرفة الملفات التي سوف يتم حذفها دون ان تحذف :

```
git clean -n
```

بعد معرفتها يمكننا حذفها نهائيا لذلك نكتب الأمر :

```
git clean -f
```

ملاحظة : تحذف الملفات الموجودة فقط في منطقة العمل working directory فقط



ملاحظة : في حال أردنا أن نحذف شيء قديم بهذه الطريقة فسوف يتم حذف كل التحديثات يلي قبل commit الذي أخترناه لذلك أحزر عند استعمال هذه الطريقة

عند رفع على github remote repo ملفات ونرغب بحذفها من المستودع على github نتبع الخطوات التالية :

أولا نضغط الأمر التالي لمعرفة id ال commit يلي قبل ال commit الذي نريد حذفه وننسخه :

```
git log
```

ثانيا نكتب الأمر مع لصق id ال commit الذي قبل الذي نريد حذفه و نلصقة أخر الأمر :

```
git reset --hard id
```

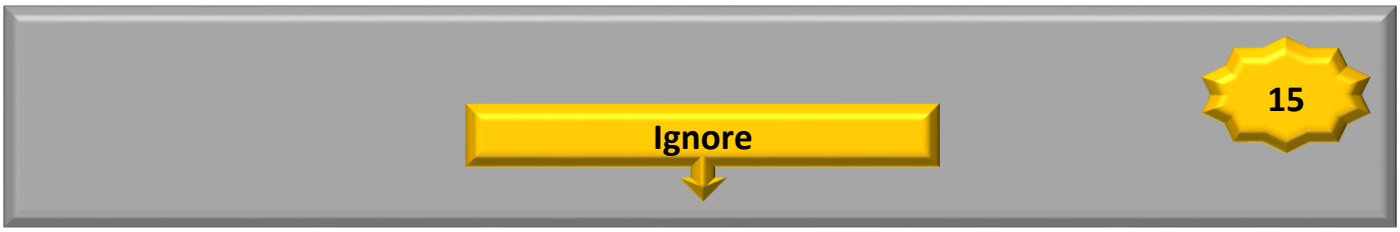
هكذا تم حذف ال commit وجعل ال commit قبل الذي نريد حذفه في الأعلى

ثالثا نحدث مستودع على github من خلال الأمر :

```
git push origin main --force
```

هكذا تم تحديث المستودع وحذف كلشي commit قبل يلي أخرناه

ملاحظة : في حال أردنا أن نحذف شيء قديم بهذه الطريقة فسوف يتم حذف كل التحديثات يلي قبل commit الذي أخترناه لذلك أحزر عند استعمال هذه الطريقة



وهي ميزة يمكن استخدامها لمنع الملفات من الرفع على staging area وبتالي منعها من الرفع على remote repo

لإنشاء ملف .gitignore لحفظ فيه أسماء الملفات التي لا نرغب برفعها :

```
touch .gitignore
```

ثم ندخل اليه من خلال vs code :

```
code .gitignore
```

ونكتب داخله أسماء الملفات :

```
جمع الملفات التي إمتدادها // *.log
من عدا الملف التالي // !vol.log
لمنع مجلد //  node-modil/
لمنع ملف معين // text.txt
```

لإضافة الملفات رغم .gitignore :

```
git add -f file
```

لتحكم بإصدارات المشروع حيث يمكن تحمل ورفع كل إصدار على حدا

أولا نقوم برفع المشروع على remote repo على فرع جديد ثم ننشئ ال tag مع التسمية v1.0 وبتالي

أنشاء tag الأصدار الأول للمشروع :

```
git tag v1.0
```

لرفع ال tag على remote repo :

```
git push origin v1.0
```

هكذا تم رفع الأصدار

قمنا بتحديث المشروع ونرغب برفع إصدار أخر

لرفع إصدار ثاني للمشروع :

أولا نقوم برفع المشروع على remote repo على فرع جديد ثم ننشئ ال tag مع التسمية v2.0 وبتالي

```
git tag -a v2.0 -m “des”
```

لرفع ال tag على remote repo :

```
git push origin v2.0
```

للبحث أو لمعرفة عدد الأصدارات :

```
git tag -l “v1.*”
```

لإنشاء releasing لكل تاغ :

ندخل على ال github ثم نضغط على خيارات ال tag ثم نضغط على create relese ثم نكتب الأسم ثم نكتب الوصف مع التنسيقات ثم نضغط على publish

لحذف tag أو releasing :

أما ندخل عليها في github ونحذفها مع العلم لا يتم حذف ال tag على الحاسب

أو من ال git حيث نحذفها على الحاسب من خلال الأمر

```
git tag -d v1.0
```

لحذف ال tag من على github remote repo :

أولا نحذف ال tag من على الحاسب مثل الأمر السابق ثم نحذف من على github من خلال الأمر

```
git push origin --delete v1.0
```

لمعرفة عدد ال tag يلي موجودة على الحاسب :

```
git tag
```

ملاحظة نهائية : أوامر الرفع يمكن التي كنا نقوم بها من ال git يمكن عملها من vs code

اسم الوظيفة	الرمز

ملاحظة: