BEST COURSE

Git & GitHub

اقوى دورة تعلم Git و GitHub

- افضل كورس تعلم جيت و جيت هب من الصفر حتى الاحتراف
- Git and GitHub for Beginners - Crash Course

- **ماهو Git و GitHub**
- الفرق بين GitHub و Git
- كيفية استخدام Git و GitHub
- أهم أوامر Git لجعلك محترفًا

شرح ماهو Git و GitHub والفرق بينهم

منصة يمكنك استخدامها لنشر مشاريعك

- GitHub هو موقع على الإنترنت يستضيف المشاريع التي تعتمد على Git
- يمكن للمطورين تحميل مشاريعهم إلى GitHub ومشاركتها مع الآخرين.
- يتيح ميزات إدارة المشروع مثل تتبع التغييرات وإدارة الأذونات وقراءة ومراجعة التعليقات

What is GitHub?

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world,
- In this tutorial, we will focus on using Git with GitHub.

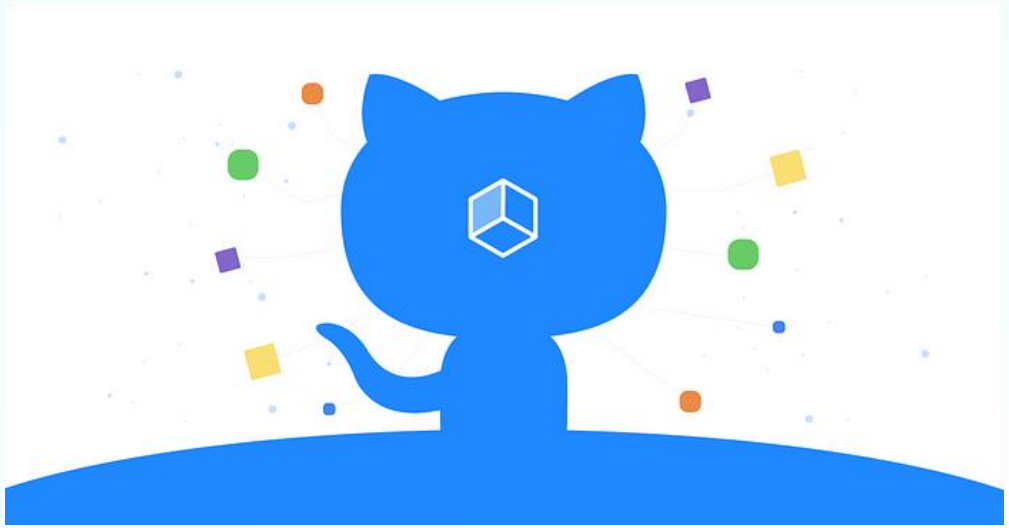
اداة لتسهيل إدارة وتتبع تغييرات الملفات

- يمكنك من خلالها بسهولة حفظ اي تعديل جديد على جهازك وGitHub او استرداد اي تعديلات يحدث على ملفات في GitHub
- يمكنك إنشاء نسخ مختلفة من مشاريعك ودمج تغييرات بسهولة
- يسهل على مطورين عمل على المشاريع البرمجية كا فريق بنفس مشروع

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

كيفية استخدام Git و GitHub



لاستخدام GitHub فيمكنك إنشاء حساب GitHub

لاستخدام Git فيمكنك تحميلها وتنصيبها على الكمبيوتر خاص بك

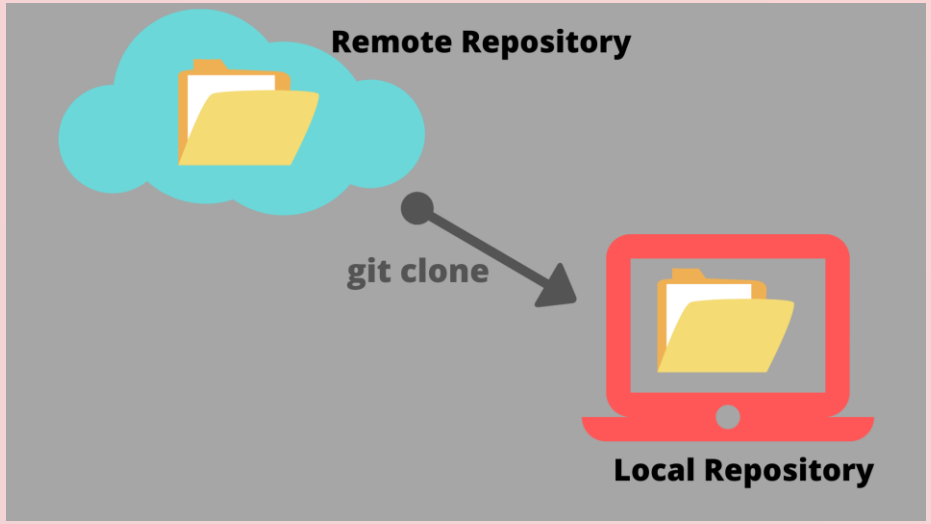
شرح كيفية إنشاء حساب GitHub

طريقة تحميل وتنصيب Git

- sudo apt-get install git
- https://git-scm.com/downloads

شرح كيفية استخدام GitHub

شرح كيفية استخدام Git



Git Clone

- يستخدم لحصول على نسخة كاملة من مستودع Git للعمل عليها محليًا

git config

git config --global user.name "Your GitHub Username"
git config --global user.email "Your GitHub Email"

git clone https://<username>:<access_token>@github.com/.....

```
git config --global user.name "اسم المستخدم الخاص بك"
git config --global user.email "عنوان البريد الإلكتروني الخاص بك"
```

- أمر git init يستخدم لإنشاء مستودع Git جديد على جهازك.

```
git init
```

- لربط لاحقا مستودك ب GitHub (على الأنترنت) :

1. قم بإنشاء مستودع جديد على GitHub
2. استخدم الأمر التالي لربط المستودع المحلي بمستودع GitHub

```
git remote add origin <remote-url>
```

حيث تحتاج إلى استبدال `<remote-url>` برابط URL الخاص بمستودع GitHub الذي أنشأته في الخطوة الأولى.

3. رفع التحديثات إلى مستودع GitHub:

بعد ربط المستودع المحلي بمستودع GitHub، يمكنك استخدام الأمر "git push" لرفع التحديثات المحلية إلى مستودع GitHub. استخدم الأمر التالي:

```
git push -u origin main
```

- أمر git init يستخدم لإنشاء مستودع Git جديد على جهازك.

```
git init
```

- طريقة أخرى لربط لاحقا مستودك ب GitHub (على الأنترنت) هو عند إنشاء Repository جديد
سيتم إعطائك اوامر لقيام بذلك - **مثال** :

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/shiyaracademy/test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/shiyaracademy/test.git
git branch -M main
git push -u origin main
```

- **git add ***

  تستخدم لإضافة تغييرات جميع الملفات إلى المنطقة الوسيط بين مجلد العمل (working directory) وسجل المستودع (repository)

- **git commit -m " "**

  لتأكيد وتسجيل التغييرات في سجل المستودع (repository)

- **git push**

  لإرسال تعديلات إلى GitHub لحفظها


git log
</>
>_
commit 2d358ee491.....
commit c2995fda.....
commit 258a5387124a....

git reset

1. إلغاء آخر commit والاحتفاظ بالتغييرات:

```
Coder Shiyar
git reset HEAD~
```

يلغي آخر commit ويعيد المستودع إلى حالة commit السابقة. التغييرات التي تمت في آخر commit ستظل في نطاق العمل المحلي.

2. إلغاء آخر commit وحذف التغييرات:

```
Coder Shiyar
git reset --hard HEAD~
```

يلغي آخر commit ويحذف جميع التغييرات التي تمت فيه. سيتم فقدان التغييرات نهائيًا.

3. إلغاء commit محدد والاحتفاظ بالتغييرات:

```
Coder Shiyar
git reset <commit-hash>
```

يلغي commit محددًا ويعيد المستودع إلى حالة commit السابقة. التغييرات التي تمت في الcommit الملغاة ستظل في نطاق العمل المحلي.

4. إلغاء commit محدد وحذف التغييرات:

```
Coder Shiyar
git reset --hard <commit-hash>
```

يلغي commit محددًا ويحذف جميع التغييرات التي تمت فيه. سيتم فقدان التغييرات نهائيًا.

git diff

1. عرض التغييرات بين الحالة الحالية للعمل المحلي والفرع الحالي:

```
git diff
```

يعرض التغييرات التي تم إجراؤها في الملفات المحلية ولم تتم تأكيدها بعد في commit.

2. عرض التغييرات بين commit معين والحالة الحالية:

```
git diff <commit>
```

يعرض التغييرات بين commit المحدد والحالة الحالية للعمل المحلي.

3. عرض التغييرات بين commitين مختلفين:

```
git diff <commit1> <commit2>
```

يعرض التغييرات بين commit1 وcommit2.

4. عرض التغييرات في ملف محدد:

```
git diff <file>
```

يعرض التغييرات التي تم إجراؤها في الملف المحدد.

git branch

أمر "git branch" يستخدم لإدارة الفروع في مستودع Git. يتيح لك عرض الفروع الموجودة، إنشاء فرع جديد، حذف فرع، وتغيير الفرع الحالي. إليك بعض الأمثلة لاستخدام "git branch":

1. عرض الفروع الموجودة:

```
git branch
```

يعرض قائمة بجميع الفروع الموجودة في المستودع، ويوضح الفرع الحالي بعلامة "*".

2. إنشاء فرع جديد:

```php
git branch <branch-name>
```

يقوم بإنشاء فرع جديد بالاسم المحدد، ويستند إلى الفرع الحالي. لا يؤدي هذا الأمر إلى التحويل إلى الفرع الجديد.

3. التحويل إلى فرع آخر:

```php
git checkout <branch-name>
```

يقوم بالتبديل إلى الفرع المحدد، ويقوم بتحديث الفهرس وملفات العمل لتكون وفقًا للفرع الجديد.

4. حذف فرع:

```php
git branch -d <branch-name>
```

يقوم بحذف الفرع المحدد. يجب عليك التأكيد على أنك لست في الفرع الذي ترغب في حذفه.

تذكر أن عمليات إنشاء وحذف الفروع تتطلب الحذر، حيث يجب التأكد من عدم فقدان التعديلات المهمة. تحتفظ Git بسجل كامل للتغييرات في كل فرع، ومن السهل الانتقال بين الفروع والعودة إلى حالات سابقة عبر سجل التاريخ.

git merge

Git merge هو أمر يستخدم لدمج تغييرات من فرع (branch) إلى آخر في Git. يُستخدم هذا الأمر عندما تكون لديك فرعان مستقلان وترغب في دمج التغييرات المُجراة في أحدهما إلى الفرع الآخر. هذا يُستخدم بشكل شائع عند العمل الجماعي في فرق التطوير.

لنوضح ذلك بمثال:

لنفترض أن لدينا فرعين في مستودع Git:

الفرع الأول يُدعى "feature-branch" وقد تم العمل عليه لتنفيذ ميزة جديدة.
الفرع الثاني هو الفرع الرئيسي "main-branch" الذي يحتوي على النسخة الأساسية للمشروع.

الآن، إذا كانت الميزة جاهزة للدمج، يمكن استخدام الأمر التالي:

Coder Shiyar Copy code

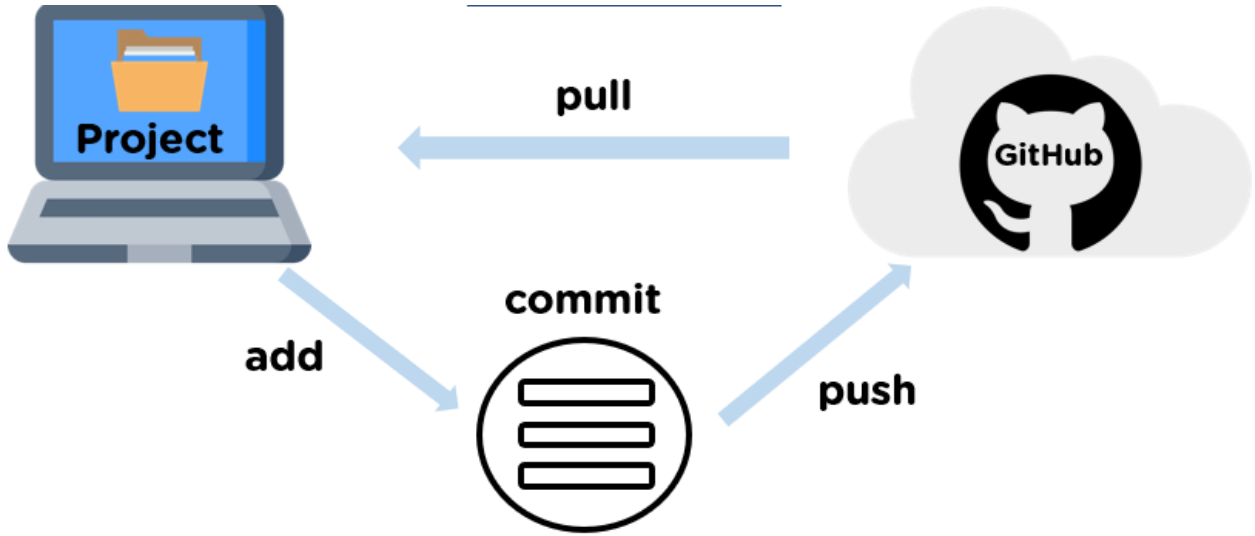
```
git checkout main-branch
git merge feature-branch
```

في هذا المثال، نقوم بالانتقال إلى الفرع الرئيسي "main-branch" ثم نستخدم أمر **`git merge`** لدمج التغييرات من "feature-branch" إلى "main-branch".

يمكن أن يتسبب دمج الفرعين في حالات مختلفة، بما في ذلك:

إذا لم تكن هناك تعارضات في التغييرات بين الفرعين، يتم الدمج بنجاح وتُدمج التغييرات بشكل آلي.
إذا كانت هناك تعارضات في التغييرات، Git يعلمك بذلك ويُطلب منك حل الصراعات يدويًا. يتعين عليك حل الصراعات وتحديد النسخة الصحيحة لكل تغيير قبل استكمال عملية الدمج.

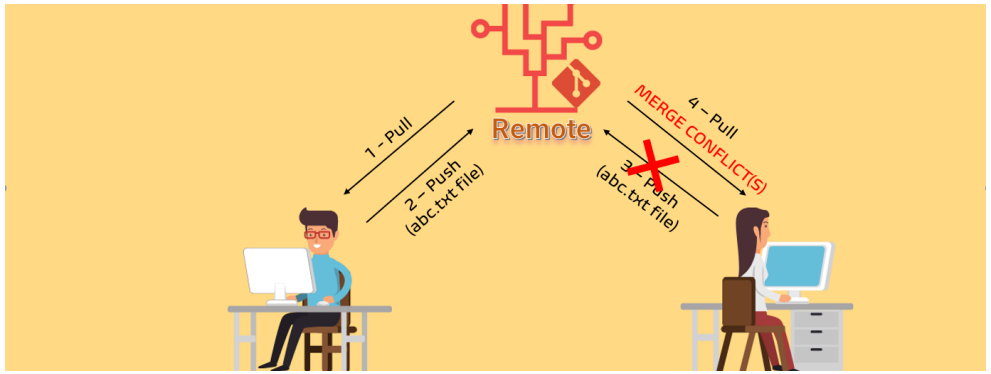
هذا مثال بسيط لكيفية استخدام Git merge لدمج التغييرات بين الفروع. يجب الإشارة إلى أنه يمكن تطبيق الأمر في سياقات أخرى وأكثر تعقيدًا بناءً على هيكل المشروع وطريقة عمل فريق التطوير.



git pull

- يستخدم لجلب التحديثات جديدة من مستودع على GitHub ودمجها مع الفرع المحلي الحالي.

Merge conflicts



تحدث التعارضات (conflicts) في Git عندما يحاول Git دمج التغييرات من فرعين مختلفين ويجد تعارضًا بينهما في نفس الملف أو نفس السطر. هذا يحدث عندما يقوم شخصين أو أكثر بتعديل نفس الجزء من الملف في فروع مختلفة بشكل متزامن.

لنقم بتوضيح ذلك من خلال أمثلة:

1. تعارض في نفس الملف:
   افترض أن لديك فرع "branchA" وفرع "branchB"، وكلاهما يقوم بتعديل نفس الملف "file.txt". على سبيل المثال، قام "branchA" بتغيير سطر محدد في الملف، بينما قام "branchB" بتغيير سطر مختلف في الملف. عند محاولة دمج "branchB" إلى "branchA"، ستظهر تعارضات في الملف "file.txt" ويحتاج المستخدم إلى حلها يدويًا عن طريق تحديد الإصدار الصحيح لكل جزء من التغييرات.
2. تعارض في نفس السطر:
   قد يحدث تعارض أيضًا عندما يقوم "branchA" و "branchB" بتعديل نفس السطر في نفس الملف. عند محاولة دمج الفرعين، ستظهر تعارضات في نفس السطر، ويتعين على المستخدم حل التعارض يدويًا عن طريق تحديد الإصدار الصحيح للسطر المعتمد على الحاجة والمنطق.

حل التعارضات ينطوي عادة على تحرير الملف المتضارب يدويًا باستخدام محرر النصوص وتحديد الإصدار الصحيح لكل جزء من التعارض. بعد حل التعارضات، يتم إضافة الملف المحلول والتعديلات المطبقة، ثم إكمال عملية الدمج باستخدام `git commit`.