

Git-Arabic-Cheat-Sheet

الفهرس

- [الإنشاء](#)
- [التعديلات المحلية](#)
- [المؤشر](#)
- [للتراجع](#)
- [للحذف](#)
- [لنقل الملفات](#)
- [التفاصيل التاريخية](#)
- [مستودع الشفرة](#)
- [للإختصارات والأسماء مستعارة](#)
- [للتنظيف](#)
- [##الإنشاء](#)

- لإنشاء مستودع فارغ:

- `$ git init`

- ##التعديلات المحلية

- لإضافة ملف:

- `$ git add <file_name>`

- **مثال على ذلك:**

- `$ git add home.php`
- `$ git add contact.php`
- `$ git add admin.php`

- لإضافة العديد من الملفات:

- `$ git add .`

- للتخزين الفعلي للتعديلات و حفظها:

- `$ git commit -m 'reason here..'`

- للتراجع عن العمليات و التعديلات التي تقوم بها

- `$ git commit --amend`

مثال يوضح كيفية إضافة ملف بعد عمل **commit** :

- `$ git commit -m 'initial commit'`
- `$ git add file.cpp`
- `$ git commit --amend`

- لعرض تفاصيل عن حالة الملفات

- `$ git status`

للحصول على تقرير مختصر عن حالة الملفات :

```
$ git status --short
```

للحصول على تقرير مختصر حول حالة المشروع والتعديلات الحالية :

```
$ git status -s
```

إلغاء كل التعديلات والعودة للنسخة التي كنت عليها قبل البدء في التعديل :

```
$ git checkout -- <file_name>
```

مثال على ذلك:

```
$ git checkout -- file.java
```

لحفظ حالة التفرع على ما هي عليه حتى تعود إليها مرة أخرى وتكمل العمل دون أن تحفظ أي Commit :

```
$ git stash
```

لمعرفة قائمة الحالات التي قمت بتخزينها لكي تساعدك في الرجوع للحالة التي تريدها :

```
$ git stash list
```

لعرض قائمة بالحالات التي قمت بتخزينها من قبل و بإمكانك الرجوع لأي منها، أي عمل Reapply :

```
$ git stash apply
```

للعودة لأحد الحالات المخزنة مسبقاً، فبإمكانك استخدام الاسم الذي يظهر مع تلك الحالة عند القيام بتنفيذ الأمر :

```
$ git stash apply stash@{2}
```

المؤشر##

لعرض قائمة Tags :

```
$ git tag
```

للبحث عن Tags :

```
$ git tag -l <صيغة معينة>
```

مثال على ذلك :

```
$ git tag -l "v1.7*"
```

لإنشاء Annotated Tag :

```
$ git tag -a v1.8.0 -m 'version 1.8' # للتوضيح Tag name is: v1.8.0, After -m you just write a message that will be saved with the tag.
```

لإنشاء Lightweight Tag :

```
$ git tag v1.8.0
```

لرؤية تفاصيل أكثر عن Tag :

```
$ git show v1.8.0
```

للتراجع##

للتراجع ولجعل الملف بحالة Unstage :

```
$ git reset HEAD <file_name>
```

مثال على ذلك :

```
$ git reset HEAD myCode.c
```

للحذف##

لحذف ملف وإلغاء متابعته :

```
$ git rm <file_name>
$ git commit -m 'reason here..'
```

مثال على ذلك :

```
$ git rm myFile.py
$ git commit -m 'Delete myFile.py 🚫'
```

مثال يوضح كيفية حذف جميع ملفات txt in settings folder :

```
$ git rm settings/*.txt
$ git commit -m 'Delete all .txt files in settings folder'
```

لحذف المتابعة مع بقاء الملف نفسه :

```
$ git rm --cached <file_name>
```

مثال على ذلك :

```
$ git rm --cached myFile.py
```

لنقل الملفات ##

لنقل الملف من جلد إلى مجلد :

```
$ git mv <source> <destination>
```

مثال يوضح نقل base.rb → lib folder :

```
$ git mv base.rb lib/base.rb
```

ويمكنك استخدام الأمر لإعادة تسميه ملف :

```
$ git mv <old_file_name> <new_file_name>
```

مثال على ذلك:

```
$ git mv core.java base.java
```

التفاصيل التاريخية ##

لرؤية التفاصيل السابقة للمستودع الذي تعمل عليه :

```
$ git log
```

لرؤية التفاصيل السابقة للمستودع الذي تعمل عليه ولتحديد عدد commits:

```
$ git log -n # n عدد ! للتوضيح مجرد عدد
```

مثال على ذلك:

```
$ git log -2
```

لمعرفة تفاصيل أكثر عن commits:

```
$ git log -p
```

لرؤية عدد من الإحصائيات بشكل مختصر :

```
$ git log -stat
```

لعرض المعلومات بطريقة مبسطة وبسطر واحد :

```
$ git log --pretty=oneline
```



لتحديد طريقة العرض التي تريدها و المعلومات التي تريد وضعها :

```
$ git log --pretty=format:<طريقة العرض التي تريدها>
```

مثال على ذلك :

```
$ git log --pretty=format:"%h - %an, %ar"
```

شرح لبعض أهم الرموز المتاحة :

الرمز	يعني
%H	هو الرقم الذي يأتي مع commit  commit hash
%h	نفس السابق ولكن يعرض بطريقة مختصرة أي عدد محدد من الأرقام
%an	من قام بعمل التعديلات  Author Name
%ae	بريد من قام بالتعديلات  Author Email
%ar	تاريخ إضافة التعديلات  Author Date
%s	الرسالة أو النص الذي يوضح سبب التعديلات

لتحديد المخرجات زمنياً :

```
$ git log --since=<المدة الزمنية التي تريدها>
```

مثال يوضح المدة الزمنية قبل أسبوعين:

```
$ git log --since=2.weeks
```

commits التي في تعديلاتها نص معين :

```
$ git log -S <النص الذي تريده>
```

مثال يوضح البحث عن myFunction:

```
$ git log -S myFunction
```

أهم (وليس كل) الخيارات التي تساعدك على تحديد المخرجات وفق المعايير التي تريدها:

الرمز	يعني
-n	عرض عدد محدد من المخرجات
--since, --after	التعديلات بعد تاريخ معين
--until, --before	التعديلات قبل تاريخ معين
--author	جلب المخرجات التي تطابق المؤلف

مستودع الشيفرة##

لإضافة Remote Repository

```
$ git remote add [remote_name] [remote_URL]
```

مثال على ذلك:

```
$ git remote add calc https://github.com/algorithms/calc
```

لمعرفة المستودعات التي نتعامل معها عن بعد :

```
$ git remote -v
```

للحصول على قائمة بالأسماء المستعارة أو المؤشرات التي تشير لتلك المستودعات بدون التفاصيل الأخرى :
التي ترافقها

```
$ git remote
```

لنسخ مستودع شيفرة و جلبه إلى Working Directory :

```
$ git clone [repository_URL]
```

مثال على ذلك :

```
$ git clone https://github.com/algorithms/my.git
```

تحديد اسم خاص بالمجلد إذا لم تكن تريد الاسم الافتراضي :

```
$ git clone [repository_URL] [new-name]
```

مثال على ذلك :

```
$ git clone https://github.com/algorithms/my.git proj
```

لجلب البيانات الموجودة في Remote Repository:

```
$ git fetch [remote-name]
```

مثال على ذلك :

```
$ git fetch origin
```

رفع البيانات أو التعديلات الجديدة التي قام بها المطور إلى مستودع الشيفرة الموجود على السيرفر :

```
$ git push [remote-name] [branch-name]
```

مثال على ذلك :

```
$ git push origin master
```

لمعرفة تفاصيل أكثر حول Remote Repository :

```
$ git remote show [remote-name]
```

مثال على ذلك :

```
$ git remote show origin
```

لإعادة تسمية الإسم المختصر الذي قمت بإضافته لمستودع شيفرة موجود على Server :

```
$ git remote rename [old-remote-name] [new-remote-name]
```

مثال على ذلك :

```
$ git remote rename dev devrepo
```

لحذف المستودع :

```
$ git remote rm [remote-name]
```

مثال على ذلك :

```
$ git remote rm devrepo
```

للإختصارات والأسماء مستعارة##

لوضع أسماء مستعارة أو مختصرة لأوامر كاملة أو إختصار لجزء معين من الأمر :

```
$ git config --global alias.<إختصار الذي تريده> <الأمر الذي تود إختصاره>
```

مثال على ذلك :

```
$ git config --global alias.st status
```

للتنظيف##

لتنظيف و إزالة الملفات أو المجلدات الزائدة أو التي لا تحتاج إليها :

```
$ git clean -f -d
```

لتنظيف و إزالة الملفات أو المجلدات الزائدة مع تزويدك بصورة عن ما سيتم حذفه فعلياً قبل حذفه بشكل :
فعلي

```
$ git clean -n -d
```

لحذف الملفات و المجلدات الموجودة أيضاً في gitignore .:

```
$ git clean -f -d -x
```

للتحقق مما سيتم حذفه قبل حذفه بشكل فعلي :

```
$ git clean -n -d -x
```

للتنظيف و الحذف من خلال الأسلوب التفاعلي :

```
$ git clean -x -i
```