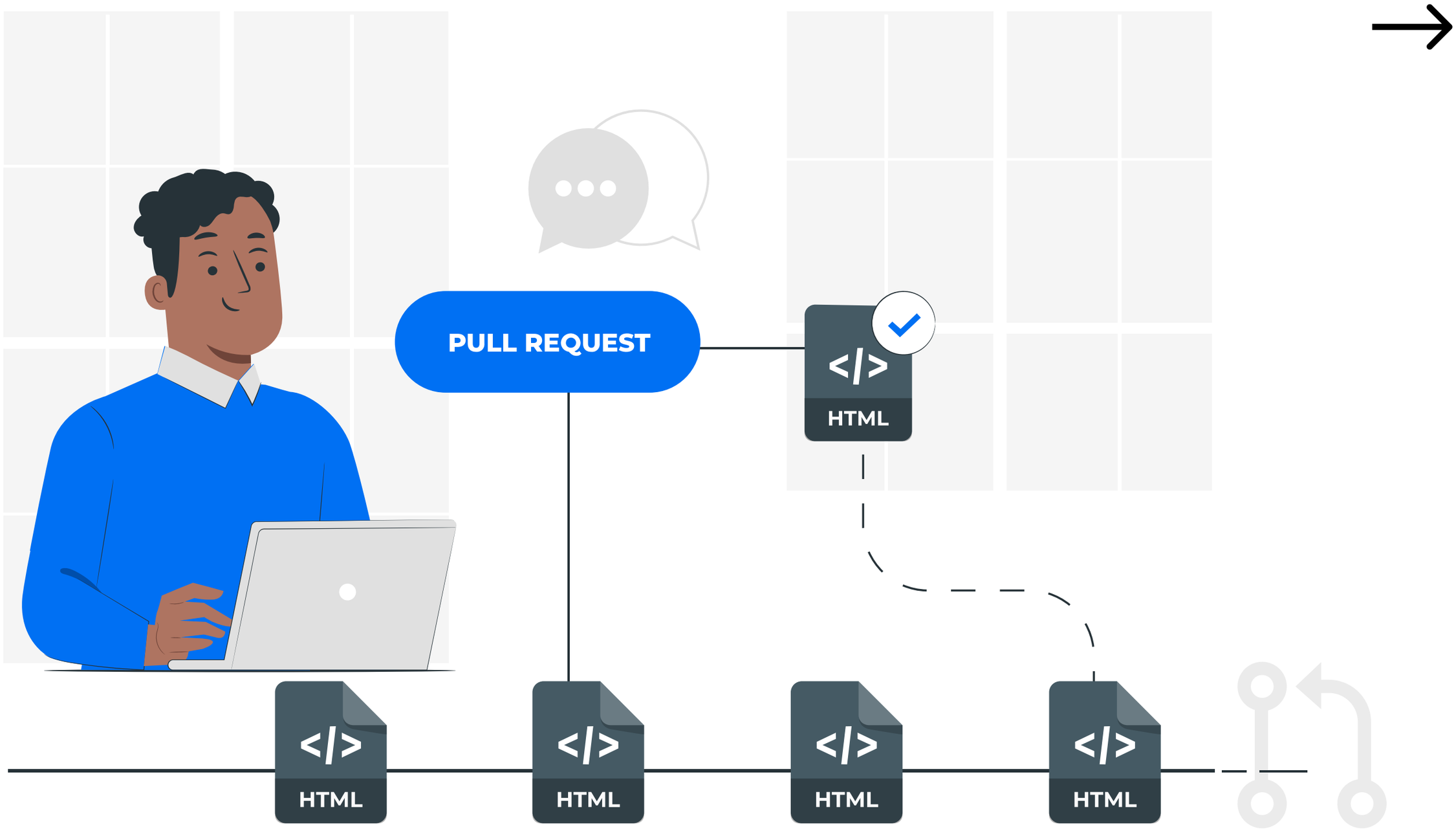
ط

تعلم أساسيات Git وأهم الأوامر

ما فائدة الـ Git وكيف نتعامل معه بسلاسة ونفهم بنيته التحتية وما أهم أوامره



Ahmed El-Tabarani

Back-End Developer

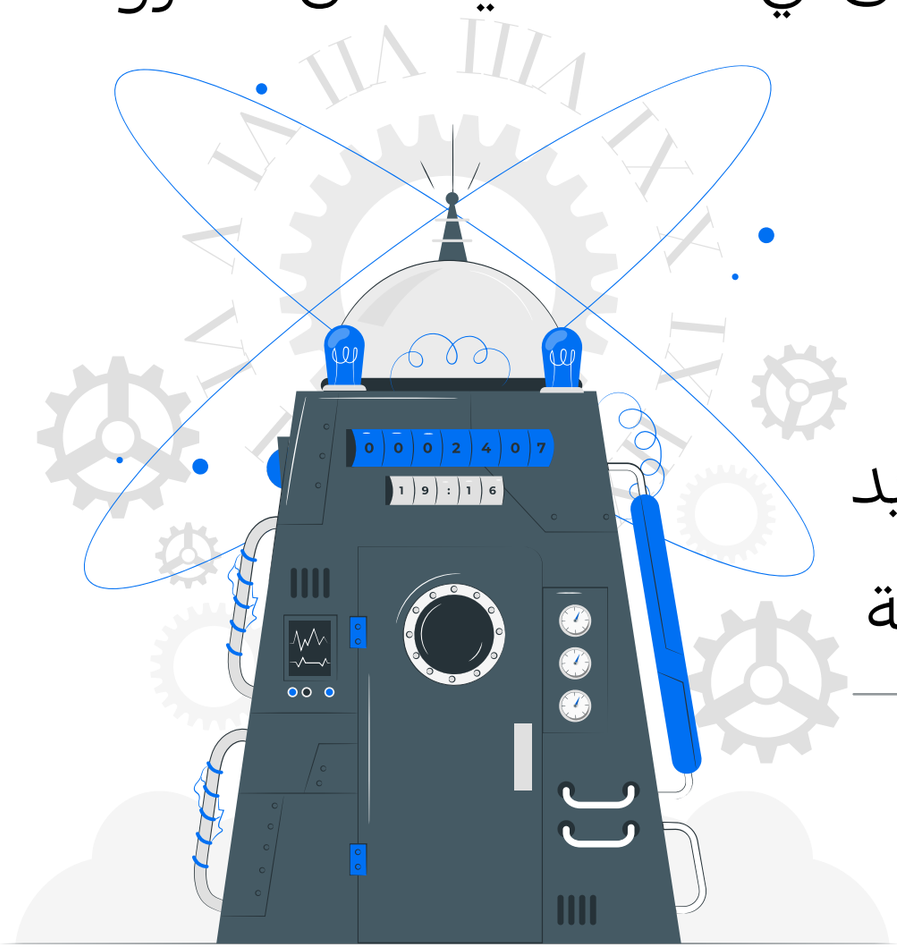
ما هو الـ Git ؟

الـ Git أشهر Version Control System

بمعنى أنه نظام متكامل يسجل كل الإضافات والتعديلات التي حدثت لمشروع على مر الزمن

ومن مميزاته:

- يتابع تاريخ المشروع بالكامل على مر الزمن، على سبيل المثال الإضافات والتعديلات والحذف مع الإشارة إلى من قام بالتغيير ومتى
- يمكنك العودة إلى الماضي في أي وقت إلى أي نسخة قديمة من مشروعك ورؤية كيف كانت حالته في هذا الوقت ومقارنة التغيرات التي حدثت له
- يمكنك إنشاء عدة نسخ من المشروع وكل نسخة تطور فيها وتعدل وتفعل ما تريد دون أن تأثر على نسخة المشروع الأساسية



الرجوع للماضي؟ نُسخ مختلفة من المشروع؟
يبدو لي كأنه جهاز للسفر عبر الزمن والأبعاد

Ahmed El-Tabarani
Back-End Developer

كيف يتعامل الـ Git مع مشروعك ؟

Working Directory

ويطلق عليه الـ Working Tree
وهو المجلد الأساسي الذي يحتوي على ملفات المشروع على جهازك

Staging Area

وتسمى أيضًا بالـ Index أو Cache
وهي مرحلة وسيطة تكون ما بين الـ Working Directory والـ Local Repository

يمكنك اعتبارها مرحلة تأكيدية ولمراجعة كل شيء قبل وضع الختم النهائي عليها، وهذا مفيد لتجنب التعديل والتغير المباشر ما بين الـ Working Directory و الـ Local Repository

وأيضًا تحتفظ بالملفات الـ Tracked ليتابعها ويقارنها مع النسخة التي في الـ Working Directory ليتأكد هل تم تعديلها أم لا

Stash

هو مكان يستخدم لحفظ التعديلات التي قمت بها بشكل مؤقت، لتتمكن من العمل على شيء آخر أو أن تنتقل إلى branch آخر
ثم بعد انتهائك يمكنك أن تعود وتستحضر هذه التعديلات المحفوظة التي قمت بها من الـ Stash من أي وقت

Repository

هو مكان يستخدمه الـ Git ليحتفظ ويتابع التعديلات التي تحدث، ويمكن أن نقسمهما إلى:

- **Local Repository**

يطلق عليه عدة مسميات مثل
Git Repository, Git Directory, Git History

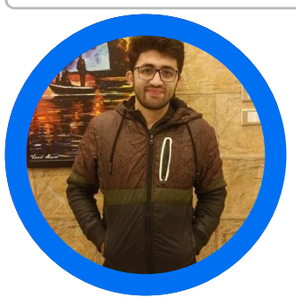
وهو الذي يقوم Git باستخدامه ليحتفظ ويتابع كل التعديلات الموجودة
في الـ Working Directory الخاص بك على جهازك

- **Remote Repository**

ويسمى Upstream Repository
هو كالـ Local Repository لكنه نسخة رئيسية من المشروع يتم استضافتها في مكان ما على الإنترنت مثل GitHub أو GitLab أو غيرهما

بالتالي يصبح لدينا نسخة مركزية من المشروع مرفوعة ومرئية للجميع
تسمى الـ Remote Repository

وكل شخص يستطيع أن يتابع أخر التعديلات التي قام بها الأشخاص الآخرون، ويرفع هو تعديلاته للجميع



Ahmed El-Tabarani
Back-End Developer

كيف يقسم الـ Git مشروعك ؟

(لمحجي الرسوم التوضيحية)

إبدأ القراءة من هنا

نسخة مركزية من المشروع مرفوعة على GitLab GitHub

Remote Repository

أعضاء الفريق

يقومون بسحب ورفع التعديلات

أنت

تقوم برفع وسحب التعديلات

المجلد الذي فيه ملفات المشروع على جهازك الشخصي

Working Directory

التراجع عن بعض التعديلات

إضافة تعديلات جديدة

Staging Area

التراجع عن بعض التعديلات

إضافة تعديلات جديدة

Local Repository

المكان الذي يحتوي على كل أسرار الـ Git ويستخدمه ليتابع ويخزن ويحتفظ بكل التعديلات وهو مجلد يسمى .git

المكان الذي يقوم بمراقبة ومتابعة الملفات والتحقق منها وللتأكد من العمليات مثل هل تريد حقًا تعديل هذا الملف؟ او حذفه او التراجع عنها او .. او .. إلخ

استرجاع التعديلات التي قمنا بتخزينها مؤقتًا

تخزين التعديلات مؤقتًا

هو مخزن يمكنك استخدامه للاحتفاظ بالتعديلات التي تقوم بها في الـ Working Directory ولا تريد أن تقوم بنقلها إلى الـ Staging بعد

Stash

Ahmed El-Tabarani
Back-End Developer

كيف يتعامل Git مع ملفات المشروع ؟

Tracked Files

هي الملفات التي أصبح Git يدرك وجودها
ويتابع أي تغير فيها
ويملك نسخة منها في الـ Staging ليتابعها

Untracked Files

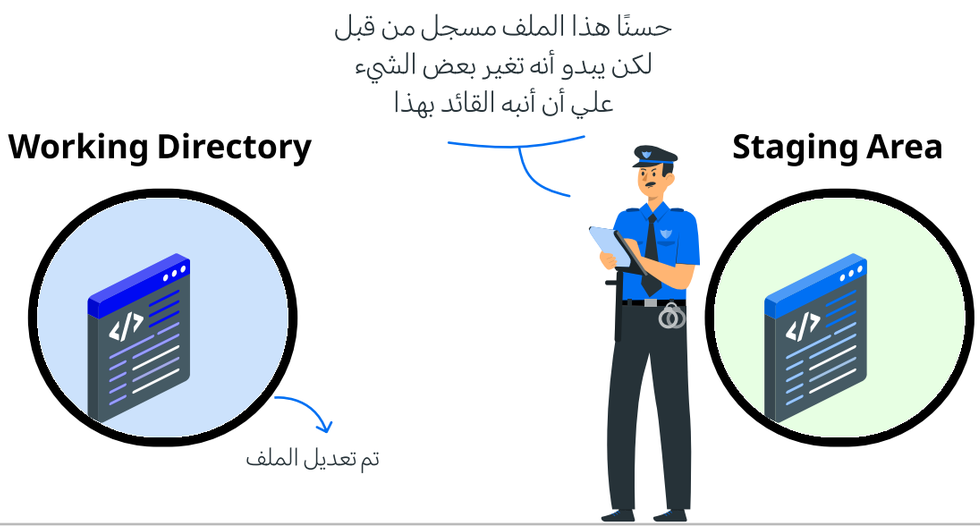
هي ملفات موجودة موجودة بالفعل في
الـ Working Directory ولكن Git لا يعرف شيئًا عنها
ولم يتخذ أي إجراء تجاهها من قبل ولا يتابعها

أي أنها لم تُنقل إلى الـ Staging من قبل أو تم حذفها من
الـ Staging والـ Git لم يعد يتابعها

Modified Files

هي الملفات التي كان Git يتابعها
أي أنها Tracked بالفعل لكن حصل لها تعديل
ولم تُنقل إلى مرحلة الـ Staging بعد

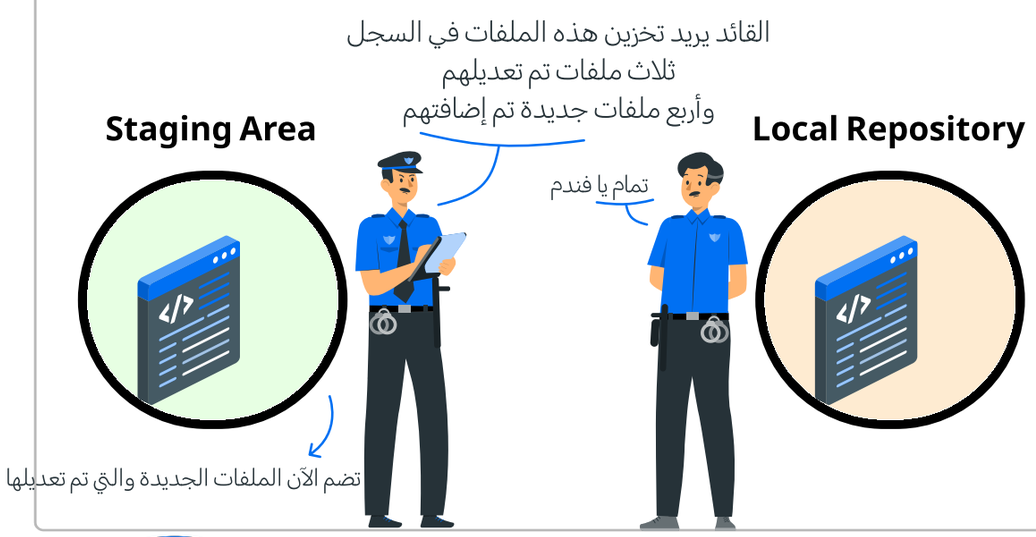
وأصبحت نسختها التي في Working Directory
مختلفة عن نسختها التي يتم متابعتها في الـ Staging

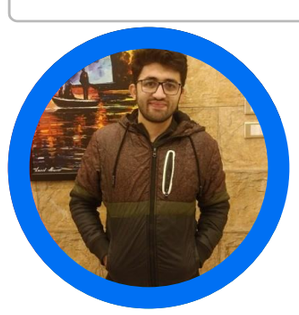


Staged Files

هي الملفات التي كان Git يتابعها
أي أنها Tracked بالفعل وحصل لها تعديل
ثم نُقلت إلى مرحلة الـ Staging

أي أن نسختها في الـ Staging
مختلفة عن نسختها في الـ Local Repository





Ahmed El-Tabarani
Back-End Developer

ط

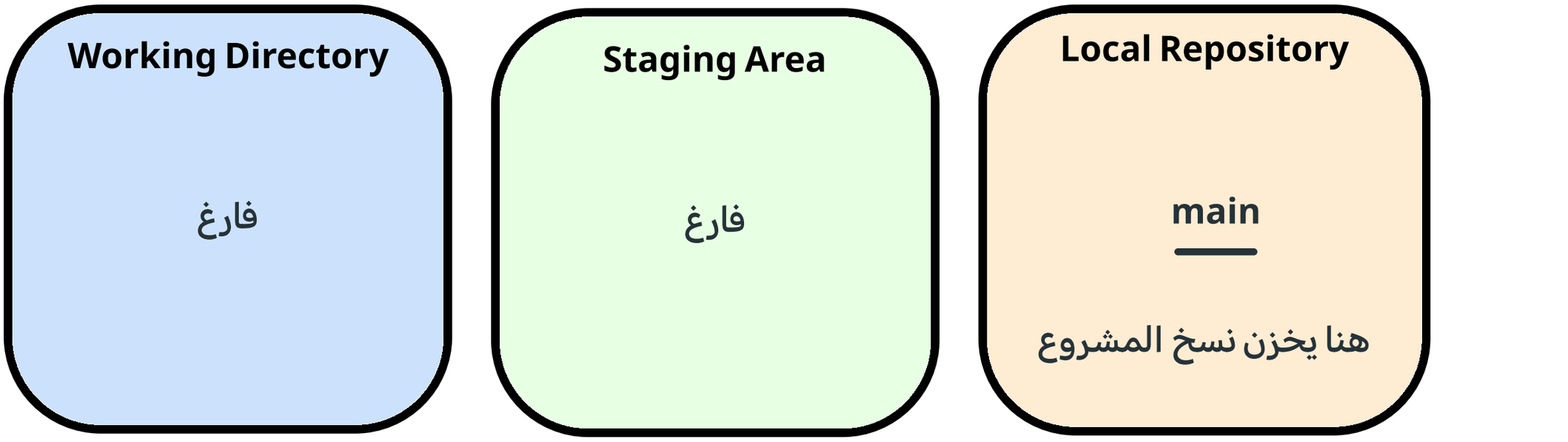
أوامر الـ Git الأساسية:

git init

نستخدمه لمرة واحدة فقط داخل المجلد الذي فيه المشروع أي الـ Working Directory لنخبر Git بإنشاء الـ Local Repository فيه ويبدأ في مراقبته ومتابعة أي تعديلات فيه

```
> git init
Initialized empty Git repository in project/.git/
```

- تم إنشاء Local Repository داخل الـ Working Directory
- تم إنشاء branch فارغ يدعى main لا يحتوي على أي commit
- الـ commit هي نسخة من المشروع، وسنتطرق لها لاحقًا



Ahmed El-Tabarani
Back-End Developer

git status

نستخدمه لمعرفة حالة المشروع والملفات هل تم تعديل الملفات ؟ هل لدينا ملفات جديدة ؟ هل هناك ملفات نُقلت للـ Staging أم لا ... إلخ

```
> git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

لاحظ أنه يخبرنا ببعض المعلومات مثل:

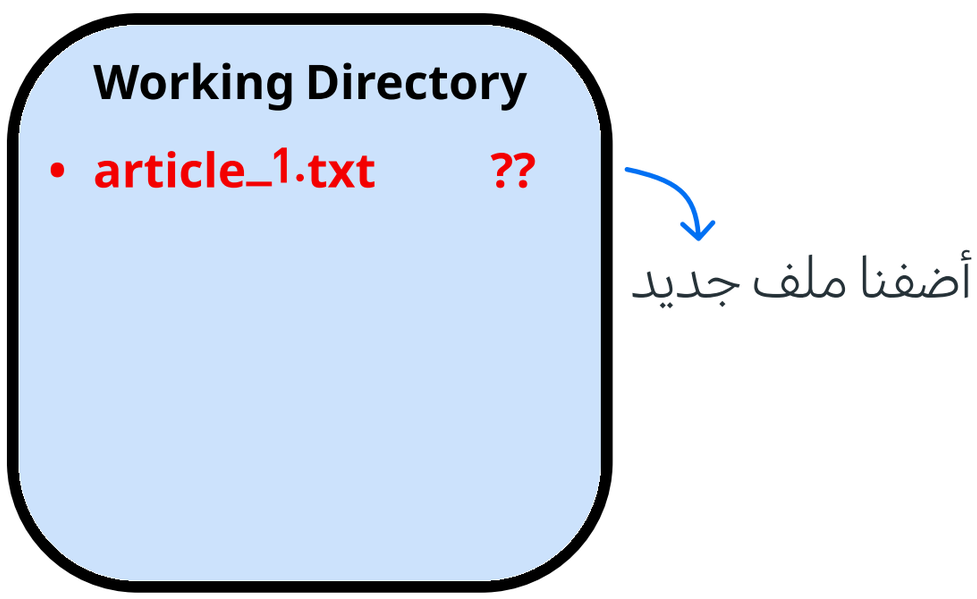
- أننا في الـ branch الأساسي main الذي ينشئه Git لنا ليضم كل الـ commit الخاصة بمشروعنا
- أننا لا نملك أي commit بعد وهذا منطقي لأننا لم نفعل شيء بعد

يفضل أن تستخدم git status قبل أن تقوم بأي شيء
لأنه يخبرك تمامًا ما يحصل ودائمًا ما يساعدك على اقتراح لك بعض الأوامر
التي يمكنك أن تقوم بها في الحالة التي أنت فيها

Ahmed El-Tabarani
Back-End Developer

لنقم بإنشاء ملف يدعى article_1.txt

ثم نقوم بتنفيذ git status




```
> git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        article_1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

لاحظ أنه قام بمعرفة التغير الذي حدث وقال لنا:

- أن هناك ملف في الـ Working Directory يسمى article_1.txt من ضمن الـ Untracked Files و Git لا يعرف عنه أي شيء
- يساعدنا ويقول أن نستخدم git add لتحويل الملف إلى Tracked من قبل Git وأيضًا لنقل الملف إلى مرحلة الـ Staging

---

الآن سنطبق نصيحته ونقوم بتنفيذ الأمر git add

ثم ننفذ git status مجددًا لنرى ما الذي سيقوله لنا

Ahmed El-Tabarani
Back-End Developer

git add

نستخدمه لنقل التعديلات من الـ Working Directory إلى مرحلة الـ Staging ويحول الملفات من حالة Untracked إلى Tracked

```
> git add article_1.txt
```

بعد إضافة الملف article_1.txt إلى مرحلة الـ Staging

يمكننا استخدام git status لنرى حالة الملفات الآن

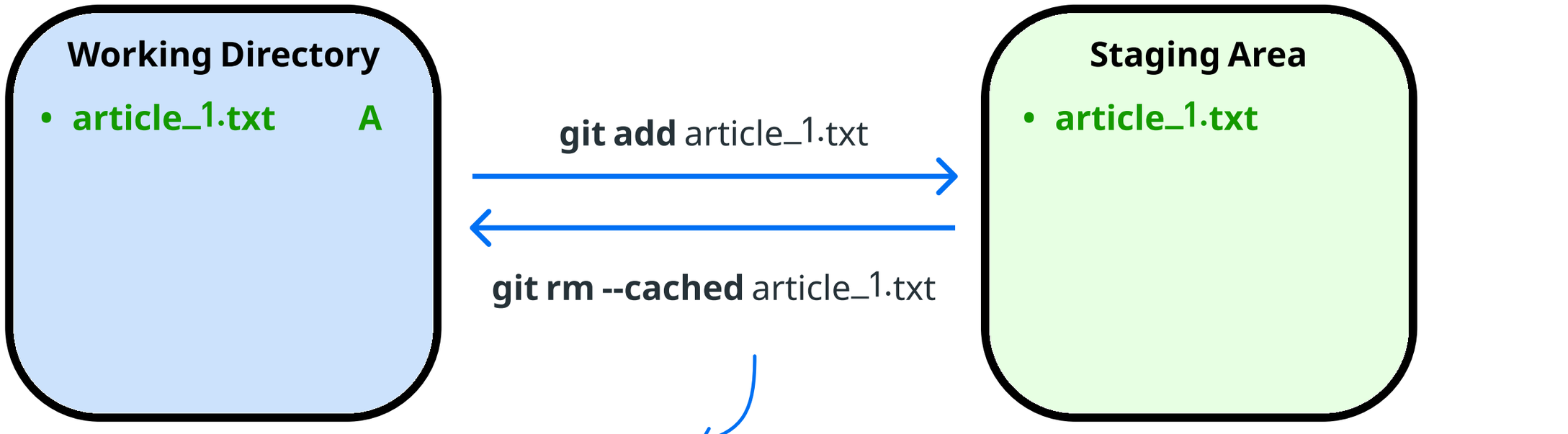
```
> git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   article_1.txt
```

ستلاحظ أنه يخبرنا أن الملف الآن

في مرحلة الـ Staging

وأنه اصبح جاهزًا لكي نقوم بعمل commit

لننقله إلى الـ Local Repository



يقوم بحذف الملف من الـ Staging

فبالتبعية سيتم تحويل حالة الملف من Tracking إلى Untracking

Ahmed El-Tabarani

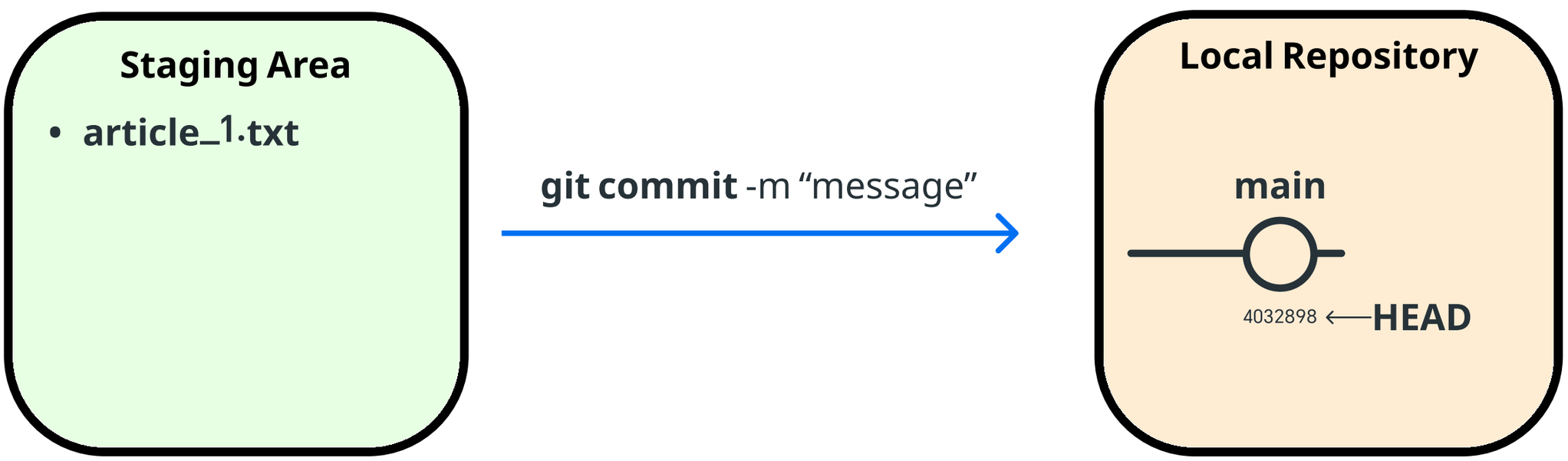
Back-End Developer

git commit

نستخدمه لنقل التعديلات من مرحلة الـ Staging إلى الـ Local Repository

```
> git commit -m "add new article to the blog"
[main (root-commit) 4032898] add new article to the blog
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 article_1.txt
```

نستخدم -m لكتابة رسالة توضح التغييرات التي تمت في هذا الـ commit



لاحظ الآتي:

- تم إنشاء أول commit داخل الـ Local Repository
- تم إنشاء مؤشر يدعى HEAD يشاور على أول commit في الـ main
- الـ commit هو يمثل نسخة جديدة من المشروع والتعديلات التي حصلت فيه
- تم نقل كل ما هو موجود في مرحلة الـ Staging كحزمة واحدة إلى الـ Local Repository في commit واحد فقط
- مكان الـ HEAD يعكس المحتوى الذي نراه في الـ Working Directory

Ahmed El-Tabarani
Back-End Developer

git restore

نستخدمه لنتراجع عن تعديل لملف
في الـ Working Directory

لنقم بتعديل الملف article_1.txt ثم نرى الـ git status

```
> git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   article_1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

يقوم الـ Git بتوضيح كل شيء ويساعدك على اتخاذ القرارات، فيقول لك:

- أنه لا يوجد ملفات حالتها Untracked لكن يوجد ملف Tracked تم التعديل عليه
  بمعنى أن نسخته في الـ Working Directory مختلفة عن نسخته في الـ Staging
- ويمكنك أن تضيف هذا الملف إلى مرحلة الـ Staging عن طريق git add
- ويمكنك التراجع عن التعديل وجعل الملف يرجع لحالته الأصلية عن طريق git restore

```
> git restore article_1.txt
```

ما يقوم به git restore حقًا
هو أنه يأخذ نسخة الملف التي في الـ Staging
ويستبدلها مع الملف الذي في الـ Working Directory

Ahmed El-Tabarani
Back-End Developer

git restore --staged

نستخدمه لنتراجع عن تعديل لملف دخل مرحلة الـ Staging (نخرجه فقط منها)

لنقم بتعديل الملف article_1.txt مجددًا ونقوم بعمل git add هذه المرة ثم نرى الـ git status

```
> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   article_1.txt
```

الـ Git يوضح الحالة، ويقول لك:

- أن هناك ملف تم تعديله ونُقله إلى مرحلة الـ Staging
- نستطيع أن نقوم بعمل commit ليتم نقله إلى الـ Local Repository عن طريق git commit
- أو نتراجع عن هذا التعديل وإخراج الملف فقط من مرحلة الـ Staging

نستعمل git restore --staged

```
> git restore --staged article_1.txt
```

Ahmed El-Tabarani
Back-End Developer

git rm

نستخدمه لحذف ملف من المشروع كليًا أي من الـ Working Directory ومن الـ Staging

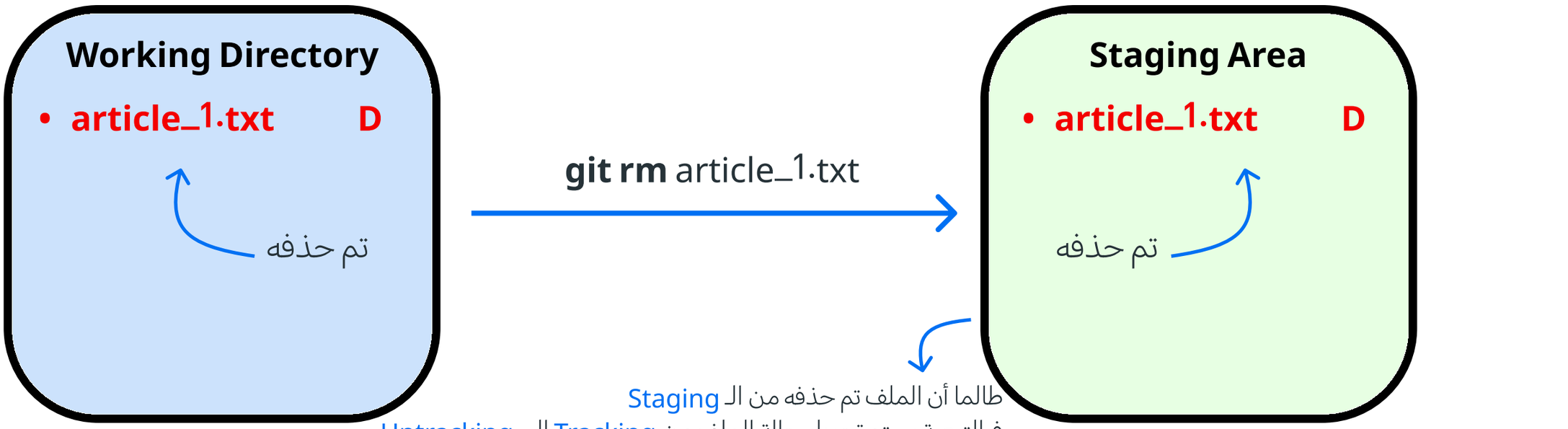
```
> git rm article_1.txt
rm 'article_1.txt'
```

قمنا بحذف article_1.txt من الـ Working Directory والـ Staging، لنرى الـ git status ونتأكد

```
> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    article_1.txt
```

لاحظ الآتي:

- تم وضع الملف إلى الـ Staging تلقائيًا مع للإشارة بأنه حُذف ولتجهيزه للحذف في الـ commit التالي
- يتم الإشارة إلى الملف بأنه محذوف بمعنى انه سيتم حذفه في النسخة الجديدة التي سينشئها الـ commit
- للتراجع عن عملية الحذف سنستعمل git restore --staged لإرجاع نسخة الملف من الـ Local Repository إلى الـ Staging ثم سنستعمل git restore لإرجاع الملف من الـ Staging إلى الـ Working Directory



Ahmed El-Tabarani
Back-End Developer

git rm --cached

نستخدمه لحذف ملف من الـ Staging فقط!

```
> git rm --cached article_1.txt
rm 'article_1.txt'
```

قمنا بحذف article_1.txt من الـ Staging، وأبقيناه في الـ Working Directory كما هو ونستطيع أن نتأكد عن طريق الـ git status

```
> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    article_1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        article_1.txt
```

لاحظ الآتي:

- تم وضع الملف إلى الـ Staging تلقائيًا مع للإشارة بأنه حُذف ولتجهيزه للحذف في الـ commit التالي
- نفس الملف مازال موجودًا في الـ Working Directory لكنه Untracked

بسبب أن الأمر git rm --cached لا يحذف الملف من الـ Working Directory بل يحذفه فقط من الـ Staging وبالتالي فسيتم تحويل حالة الملف من Tracking إلى Untracking

Ahmed El-Tabarani
Back-End Developer

لنقم بعمل commit جديد

قبل أن نكمل دعونا نُنشيء commit جديد ونرى ما الذي سيحدث

لنُنشيء أولًا مقالة جديدة article_xyz.txt

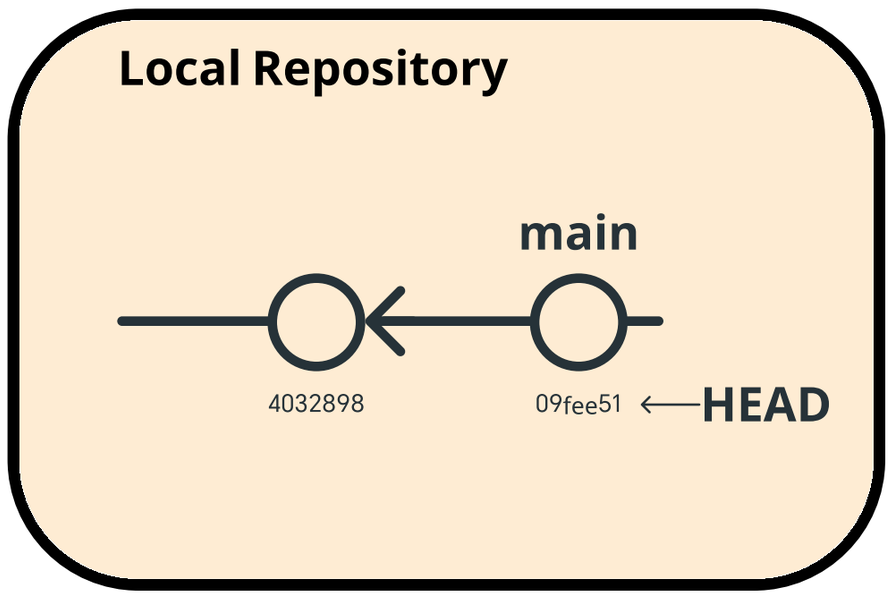
الآن نقوم بتنفيذ git add لنضعها في مرحلة الـ Staging

```
> git add article_xyz.txt
```

ثم نقوم بتنفيذ git commit لتسجيلها في الـ Local Repository

```
> git commit -m "add xyz article"
[main 09fee51] add xyz article
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 article_xyz.txt
```

الآن في الـ Local Repository تم إنشاء commit جديد يحتوي نسخة جديدة من المشروع يضم الملف الجديد



ولاحظ أن الـ HEAD تحرك من مكانه ليشاور على الـ commit الجديد

Ahmed El-Tabarani
Back-End Developer

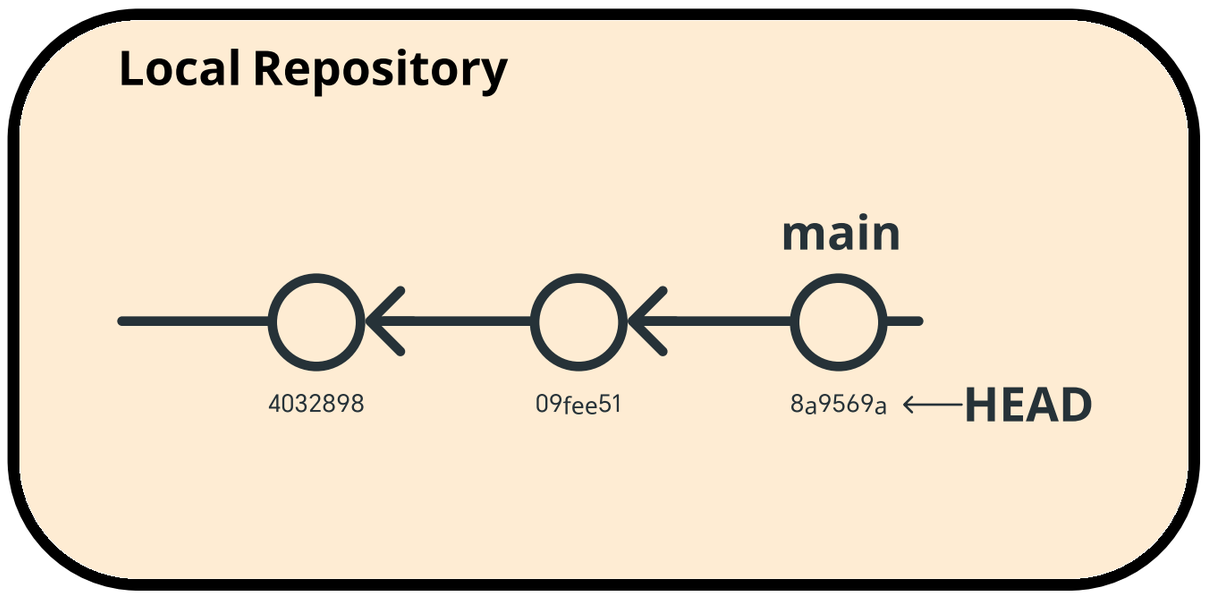
git revert

نستخدمه لنتراجع عن commit معين عن طريق إنشاء commit جديد يعكس التعديلات التي في الـ commit المعين الذي نريد التراجع عنه

وgit revert تحتاج لـ hash خاصة بالـ commit الذي نريد التراجع عنه أو يمكننا استخدام الـ HEAD لأنه بطبيعة الحال مؤشر يشير إلى commit

```
> git revert HEAD
[main 8a9569a] Revert "add xyz article"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 article_xyz.txt
```

الآن في الـ Local Repository تم إنشاء commit يتراجع عن الـ commit السابق، وأيضًا الـ HEAD تحرك من مكانه ليشاور على الـ commit الجديد



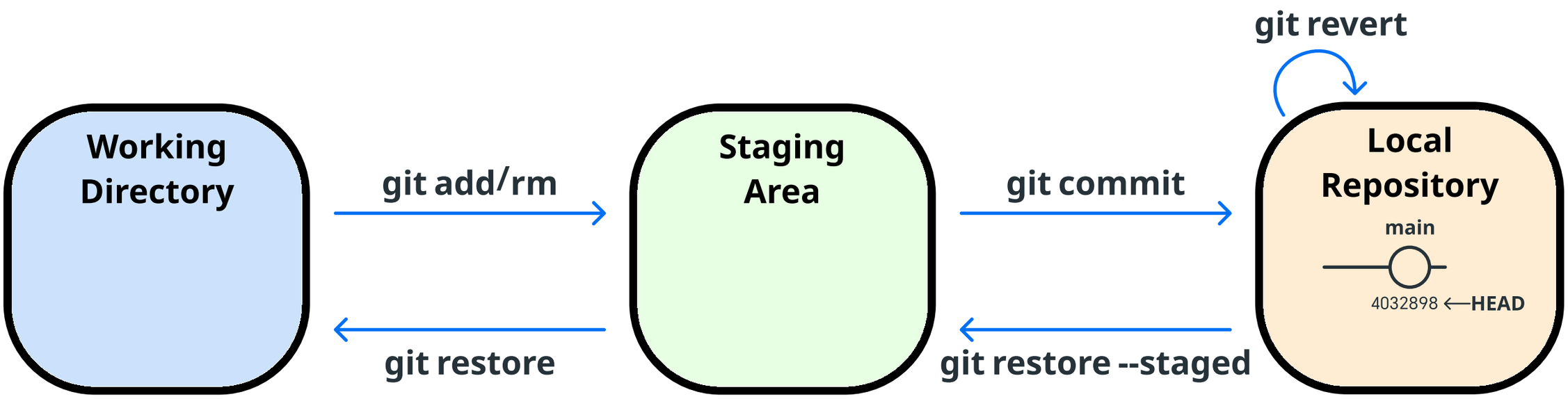
ملحوظة هامة:

مكان الـ HEAD يمثل ما هو موجود في الـ Working Directory

فإذا جعلناه يشاور على commit قديم فأن محتويات الـ Working Directory ستكون النسخة التي كان عليها في هذا الـ commit

Ahmed El-Tabarani
Back-End Developer

ملخص بسيط لما سبق (لمحبي الرسوم التوضيحية)



سنكتفي بهذا القدر البسيط في هذا الجزء
ونستكمل باقي الشرح في أجزاء أخرى بإذن الله

ويمكنك قراءة مقالة أساسيات الـ Git كاملة
من مدونتي عن طريق كتابة الرابط التالي:
tabarani.tk/articles/git-basics

المقالة تحتوى على شرح لأوامر وأمور وتفاصيل قد لا تجددها في هذه الملخصات البسيطة
وفي المقالة ستفهم لما أقول أن Git أشبه بجهاز للسفر عبر الزمن والأبعاد
لأنني لم أتطرق لأهم مزايا الـ Git في هذا الملخص

Ahmed El-Tabarani
Back-End Developer

أنا أحمد الطبراني، مهندس برمجيات SWE 😁

مبرمج متخصص في عالم الـ Backend ⚙️💻

أحب دائمًا أن أُشارك معرفتي المتواضعة مع الآخرين لعله عسى أن يستفيد شخص ما بما أكتبه 🤗
أكتب بعض المقالات عن أشياء مختلفة في عالم البرمجة، أرجوا أن تستفيدوا وتستمتعوا 😉

تابعني علي لينكدإن او علي مدونتي الشخصية