

Temporary chat  
tokens ٢٥,١١٤

User

Microsoft Windows [Version 10.0.22000.2538]

(c) Microsoft Corporation. جميع الحقوق محفوظة.

C:\xampp\htdocs\my\_blog2>git branch

api

catgory

master

C:\xampp\htdocs\my\_blog2>git merge api

.fatal: You have not concluded your merge (MERGE\_HEAD exists)

.Please, commit your changes before you merge

C:\xampp\htdocs\my\_blog2>git merge api

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

خطأ في Git: لم تكمل الدمج (MERGE\_HEAD موجود)

واجهت رسالة الخطأ "fatal: You have not concluded your merge (MERGE\_HEAD exists)" لأنك

حاولت بدء عملية دمج جديدة بينما لا تزال هناك عملية دمج سابقة لم تكتمل. يحدث هذا عادةً عندما يؤدي الدمج إلى تعارض في الملفات،

ويترك Git المستودع في حالة دمج معلق حتى يتم حل التعارض.

عندما تبدأ عملية دمج في Git، فإنه يقوم بإنشاء ملف خاص يسمى MERGE\_HEAD لتتبع فرع الدمج. [١] [٢] إذا لم تكتمل عملية الدمج

بنجاح، يبقى هذا الملف، مما يمنع عمليات الدمج الجديدة. [٢]

لحل هذه المشكلة، لديك خياران رئيسيان: إكمال الدمج الحالي أو إلغاؤه.

الخيار الأول: إكمال عملية الدمج

إذا كنت ترغب في متابعة الدمج، اتبع الخطوات التالية:

تحقق من حالة الدمج: استخدم الأمر git status لعرض حالة المستودع الحالية. سيعرض لك هذا الملفات التي بها تعارضات.

code

Shell

git status

حل التعارضات: افتح الملفات التي بها تعارضات في محرر النصوص. ستري علامات مثل >>>>>> و ===== و <<<<<< التي

تحدد الأجزاء المتعارضة من التعليمات البرمجية. [٢] [٣] قم بتحرير هذه الملفات يدويًا لإزالة العلامات وحل الاختلافات بالطريقة التي

تريدها.

إضافة الملفات التي تم حلها: بعد حل التعارضات في ملف، استخدم الأمر git add لإضافته إلى منطقة التجهيز، مما يشير إلى أن

التعارض قد تم حله.

code

Shell

git add <اسم\_الملف>

تنفيذ الدمج: بمجرد حل جميع التعارضات وإضافة جميع الملفات التي تم حلها، أكمل عملية الدمج عن طريق تنفيذ التغييرات.

code

Shell

git commit

سيفتح هذا محرر النصوص الافتراضي لكتابة رسالة تنفيذ للدمج. بعد الحفظ والإغلاق، سيتم إكمال الدمج.

الخيار الثاني: إلغاء عملية الدمج

إذا كنت لا ترغب في المتابعة في الدمج الحالي وتريد العودة إلى الحالة التي كان عليها المستودع قبل بدء الدمج، يمكنك إلغاء عملية

الدمج. [١] [٤]

code

Shell

git merge --abort

هذا الأمر سيعيد مستودعك إلى الحالة التي كان عليها قبل محاولة الدمج، وسيزيل ملف MERGE\_HEAD. بعد إلغاء الدمج، يمكنك محاولة

دمج فرع api مرة أخرى إذا كنت ترغب في ذلك.  
باختصار:

إذا كنت تريد إنهاء الدمج الذي بدأته، فقم بحل أي تعارضات، ثم استخدم `git add` و `git commit`.  
إذا كنت تريد إلغاء الدمج والبدء من جديد، فاستخدم `git merge --abort`.

Sources

help

geeksforgeeks.org

tecadmin.net

betterstack.com

bobbyhadz.com

Google Search Suggestions

Display of Search Suggestions is required when using Grounding with Google  
Search. Learn more

Google logo

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

ملاحظة! هذا الناتج من `git status` يوضح أنك قمت بحل التعارضات بنجاح وأضفت جميع الملفات الضرورية إلى منطقة التجهيز (Staging Area).

الآن، أنت في الخطوة الأخيرة لإكمال عملية الدمج.

ماذا تفعل الآن؟

بما أن جميع التغييرات جاهزة للتنفيذ ("Changes to be committed")، كل ما عليك فعله هو إنشاء "commit" لإنهاء عملية الدمج.

نقد الأمر التالي في الطرفية (Terminal):

code

Shell

git commit

ماذا سيحدث بعد تنفيذ هذا الأمر؟

سيفتح Git محرر النصوص الافتراضي لديك (مثل Vim أو Nano).

سترى رسالة commit افتراضية تم إنشاؤها تلقائيًا، والتي تبدو عادةً كالتالي:

code

Code

'Merge branch 'api

,Please enter a commit message to explain why this merge is necessary #

.especially if it merges an updated upstream into a topic branch #

#

Lines starting with '#' will be ignored, and an empty message aborts #

.the commit #

يمكنك ترك الرسالة الافتراضية كما هي، أو يمكنك إضافة ملاحظاتك الخاصة إذا أردت.

احفظ الملف وأغلق المحرر.

في Vim: اضغط على Esc، ثم اكتب wq واضغط Enter.

في Nano: اضغط على Ctrl + X، ثم Y، ثم Enter.

بمجرد إغلاق المحرر، سيتم إكمال عملية الدمج بنجاح، وسيختفي ملف MERGE\_HEAD. سيعود المستودع الخاص بك إلى حالته الطبيعية،

مع دمج جميع التغييرات من فرع api إلى فرع master الخاص بك.

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

هذه الرسالة (E325: ATTENTION) لا تأتي من Git، بل من محرر النصوص Vim الذي يستخدمه Git لإنشاء رسالة الـ commit.

ماذا يعني هذا الخطأ؟

هذا الخطأ يعني أن محرر Vim قد تم إغلاقه بشكل غير صحيح في المرة السابقة التي حاولت فيها عمل commit (على الأرجح تم إغلاق نافذة الطرفية أو تعطلت). عندما يحدث هذا، يترك Vim خلفه ملفًا مؤقتًا يسمى "swap file" (بامتداد .swp) لحفظ أي تغييرات غير محفوظة ومنع تحرير نفس الملف في مكانين مختلفين في نفس الوقت.

عندما حاولت عمل git commit مرة أخرى، وجد Vim هذا الملف المؤقت القديم وحذرك من وجوده.

الحل الأسهل والأكثر أمانًا

بما أنك تقوم فقط بإكمال عملية دمج (merge commit)، فأنت لست بحاجة إلى استعادة أي نص من المحاولة السابقة. الحل الأبسط هو حذف ملف الـ swap المؤقت والمحاولة مرة أخرى.

الخطوة ١: الخروج من المحرر

أولاً، أخرج من المحرر الحالي بالضغط على Esc ثم كتابة :q! والضغط على Enter.

code

Code

!q:

(هذا الأمر يعني "أخرج فورًا دون حفظ").

الخطوة ٢: حذف ملف الـ Swap

بعد العودة إلى سطر الأوامر، قم بتنفيذ الأمر التالي لحذف الملف المؤقت المذكور في رسالة الخطأ:

code

Shell

```
rm /c/xampp/htdocs/my_blog2/.git/.COMMIT_EDITMSG.swp
```

الخطوة ٣: المحاولة مرة أخرى

الآن بعد أن تم حذف الملف المؤقت، قم بتنفيذ أمر الـ commit مرة أخرى. هذه المرة سيفتح المحرر بشكل نظيف بدون أي تحذيرات.

code

Shell

git commit

عندما يفتح المحرر:

اترك رسالة الدمج الافتراضية كما هي.

احفظ وأخرج (اضغط Esc ثم اكتب :wq واضغط Enter).

طريقة بديلة أسرع (لتجنب المحرر)

إذا كنت لا تريد التعامل مع محرر Vim على الإطلاق، يمكنك إلغاء ما تفعله حاليًا (أخرج من المحرر بـ :q!) ثم قم بتنفيذ الـ commit مع

رسالة مباشرة من سطر الأوامر باستخدام الخيار -m.

code

Shell

```
"git commit -m "Merge branch 'api' into master"
```

هذا الأمر سيقوم بإنشاء الـ commit مباشرة بالرسالة التي قدمتها، دون فتح محرر النصوص، وهو ما يحل المشكلة ويتجنبها في المستقبل.

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

عذرًا، لقد أخطأت. الأمر rm يعمل على أنظمة Linux أو في طرفية Bash Git، ولكنه ليس أمرًا متاحًا في موجه الأوامر (Command Prompt) الخاص بنظام Windows.

الأمر الصحيح لحذف ملف في Windows هو del.

الحل الصحيح

استخدم الأمر التالي في موجه الأوامر الخاص بك:

code

Shell

```
del .git\COMMIT_EDITMSG.swp
```

ملاحظة: لقد استخدمت مسارًا نسبيًا هنا (.git\COMMIT\_EDITMSG.swp) لأنك موجود بالفعل داخل المجلد

C:\xampp\htdocs\my\_blog2. هذا أسهل من كتابة المسار الكامل.

الخطوات التالية

نَقِّدْ أمر الحذف:

code

Shell

del .git\COMMIT\_EDITMSG.swp

بعد حذف الملف بنجاح، حاول إكمال عملية الدمج مرة أخرى عن طريق تنفيذ:

code

Shell

git commit

سيفتح محرر Vim مرة أخرى. هذه المرة، لن تظهر رسالة التحذير. ببساطة احفظ الملف وأغلقه (اضغط Esc ثم اكتب wq واضغط Enter) لإكمال عملية الدمج.

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

لا تقلق، هذا خطأ بسيط جداً وشائع.

ظهور الخطأ E492: Not an editor command: qw يعني أنك كتبت أمراً خاطئاً في محرر Vim.

لقد كتبت qw عن طريق الخطأ، بينما الأمر الصحيح للحفظ والخروج هو wq (الحرفان معكوسان).

الحل

أنت ما زلت داخل المحرر. كل ما عليك فعله هو كتابة الأمر الصحيح.

اضغط على مفتاح Esc مرة أو مرتين للتأكد من أنك في الوضع العادي (Normal Mode).

اكتب الأمر الصحيح:

code

Code

wq:

اضغط Enter.

بمجرد فعل ذلك، سيتم حفظ التغييرات، إغلاق المحرر، وإكمال عملية الدمج بنجاح.

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

تهانينا! لقد نجحت.

هذا الناتج [master ddb7a1c] Merge branch 'api' هو رسالة تأكيد من Git بأن عملية الدمج قد اكتملت بنجاح.

شرح ما حدث:

[master ddb7a1c]: هذا يعني أنه تم إنشاء commit جديد على فرع master. الرمز ddb7a1c هو المعرّف الفريد (hash)

لهذا ال commit الجديد.

'Merge branch 'api': هذه هي رسالة ال commit التي قمت بحفظها للتو، والتي توثق أن هذا ال commit هو نتيجة دمج فرع

.api

لقد قمت بحل جميع المشاكل السابقة بنجاح، بدءاً من التعارضات، مروراً بمشكلة ملف ال swap في محرر Vim، وانتهاءً بكتابة أمر الحفظ الخاطئ.

الوضع الحالي لمستودعك:

الدمج اكتمل: جميع التغييرات والملفات الجديدة التي كانت في فرع api أصبحت الآن جزءًا من فرع master. المستودع نظيف: لم تعد في حالة "دمج معلق". يمكنك الآن تنفيذ أوامر Git الأخرى بشكل طبيعي (مثل git pull, git push, git branch).

الخطوات التالية الموصى بها: (اختياري) حذف الفرع المدموج: بما أنك قمت بدمج فرع api بنجاح في master، فغالبًا لن تحتاج إليه بعد الآن. من الممارسات الجيدة حذف الفروع التي تم الانتهاء من العمل عليها للحفاظ على نظافة المستودع.

code

Shell

```
git branch -d api
```

\*\*\* ملاحظة: لن يسمح لك Git بحذف الفرع إلا إذا تم دمج جميع تغييراته بالكامل، وهو ما قمت به للتو.\*

دفع التغييرات إلى المستودع البعيد: إذا كنت تعمل مع مستودع بعيد (مثل GitHub أو GitLab)، فالخطوة التالية هي دفع فرع master المحدث إلى الخادم.

code

Shell

```
git push origin master
```

عمل رائع في حل المشكلة خطوة بخطوة

User

```
C:\xampp\htdocs\my_blog2>git log
commit ddb7a1cde02fd9fd62cb296a401275478bfd7f4d (HEAD -> master, origin/master)
```

```
Merge: db09873 262de4b
```

```
Author: zakaria zeakoalslmy@gmail.com
```

```
Date: Fri Oct 17 00:44:04 2025 +0300
```

code

Code

```
'Merge branch 'api
```

```
commit 262de4b375b1fb2c936962fb5e95431c00eef20b (api)
```

```
Author: zakaria zeakoalslmy@gmail.com
```

```
Date: Thu Oct 16 23:34:26 2025 +0300
```

code

Code

```
CheakLangApi
```

```
commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c
```

```
Author: zakaria zeakoalslmy@gmail.com
```

```
Date: Thu Oct 16 00:42:40 2025 +0300
```

code

Code

```
feat: Initial API setup with authentication and core resources
```

```
.Create API v1 routes for settings, categories, and authentication -
```

```
Implement login functionality using Laravel Sanctum for secure API -
```

```
.access
```

```
.Add SettingController and CategoriesController to expose site data -
```

```
,Introduce API Resources (SettingResource, CatgoryResource -
```

```
.PostResource) to format JSON responses
```

```
Implement a middleware (CheakLangApi) to handle API language -
```

```
.localization based on headers
```

```
commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b
```

```
Author: zakaria zeakoalslmy@gmail.com
```

```
Date: Tue Sep 23 00:26:02 2025 +0300
```

code

Code

```
feat: Implement CRUD for categories, posts, and pages
```

```
.Add functionality to create, read, update, and delete categories -
```

```

        .Add functionality to create, read, update, and delete posts -
            .Create the main index page for the website -
commit 910a78e001198eef81b7cd611b34b97d5336ae1b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Mon Jun 16 17:08:12 2025 +0300
        code
        Code
            feat: Manage department users

Implement full CRUD (Create, Read, Update, Delete) functionality for
    users within the department section
Implement full CRUD (Create, Read, Update, Delete) functionality for
    users within the department section
commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Thu Oct 16 00:42:40 2025 +0300
        code
        Code
            feat: Initial API setup with authentication and core resources

        .Create API v1 routes for settings, categories, and authentication -
        Implement login functionality using Laravel Sanctum for secure API -
            .access
    .Add SettingController and CategoriesController to expose site data -
        ,Introduce API Resources (SettingResource, CatgoryResource -
            .PostResource) to format JSON responses
        Implement a middleware (CheakLangApi) to handle API language -
            .localization based on headers
commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Tue Sep 23 00:26:02 2025 +0300
        code
        Code
            feat: Implement CRUD for categories, posts, and pages

    .Add functionality to create, read, update, and delete categories -
    .Add functionality to create, read, update, and delete posts -
        .Create the main index page for the website -
commit 910a78e001198eef81b7cd611b34b97d5336ae1b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Mon Jun 16 17:08:12 2025 +0300
        code
        Code
            feat: Manage department users

Implement full CRUD (Create, Read, Update, Delete) functionality for
    users within the department section
commit 5adff9bc89c6118ec6ab38d7d17da1f108b81fa1
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Sun Mar 9 08:34:20 2025 +0300
        code
        Code
        chore: Initialize database schema for my_blog

```

"Create all the necessary tables for the initial setup of the "my\_blog project.s

commit db0987378ea2b8aa04981a92521f0d82624a8e77 (catgory)

Author: zakaria zeakoalslmy@gmail.com

Date: Tue Sep 23 00:26:02 2025 +0300

code

Code

i am crud created catgories and post and page index web site

commit ae7191ca128812a40b0e5fed51d4786f72b9f58b

Author: zakaria zeakoalslmy@gmail.com

Date: Mon Jun 16 17:08:12 2025 +0300

code

Code

today i compleated department users add user and edite and delete

commit 7c0402f5913dd3c08569bf5ad1cb4ec984fcbec3

Author: zakaria zeakoalslmy@gmail.com

Date: Sun Mar 9 08:34:20 2025 +0300

code

Code

project my\_blog today i created all tables

(END)

warning

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

بكل سرور، سأشرح لك ناتج الأمر git log بطريقة بسيطة ومفهومة.

ببساطة، git log هو سجل التاريخي الكامل للمشروع. [١] [٢] هو يريك كل التغييرات (commits) التي حدثت، من أحدث تغيير إلى أقدم تغيير. [٣] [٤] كل "فقرة" في هذا السجل تمثل تغييرًا واحدًا أو عملية حفظ واحدة (commit).

دعنا نحلل السجل الذي ظهر لك، بدءًا من الأعلى (الأحدث):

١. Commit الأول (الأحدث): عملية الدمج

code

Code

commit ddb7a1cde02fd9fd62cb296a401275478bfd7f4d (HEAD -> master, origin/master)

Merge: db09873 262de4b

<Author: zakaria <zeakoalslmy@gmail.com

Date: Fri Oct 17 00:44:04 2025 +0300

'Merge branch 'api

commit ddb7a1c...: هذا هو الرقم التعريفي الفريد (يسمى hash) لهذا التغيير. يمكنك اعتباره "رقم الفاتورة" الخاص بهذه العملية. (HEAD -> master, origin/master): هذا الجزء مهم جدًا ويخبرك بحالة المشروع الحالية:

HEAD -> master: يعني أنك حاليًا تقف على فرع master، وهذا ال commit هو آخر وأحدث تغيير في هذا الفرع على جهازك المحلي. [٥] [٦] [٧]

origin/master: يعني أن هذا ال commit موجود أيضًا في النسخة البعيدة (الريموت) من المشروع (غالبًا على GitHub) في فرع [5] [master]. [8] هذا يعني أن نسختك المحلية متطابقة مع النسخة البعيدة.

Merge: db09873 262de4b: هذا السطر يظهر فقط في عمليات الدمج (Merge commits). هو يخبرك أن هذا ال commit نتج عن دمج تغييرين آخرين (له "أبوين"):

db09873 (الذي كان آخر commit في master قبل الدمج).

٢٦٢de4b (الذي كان آخر commit في فرع api).

Author: zakaria... الشخص الذي قام بهذا التغيير.

Date: ... تاريخ ووقت التغيير.

Merge branch 'api': الرسالة التي كتبها بنفسك لوصف هذا التغيير.  
الخلاصة: هذا أول commit في السجل هو عملية الدمج التي قمت بها للتو، والتي أخذت كل التغييرات من فرع api وأضافتها إلى فرع master.

٢. ال Commit الثاني: آخر تغيير في فرع api

code

Code

commit 262de4b375b1fb2c936962fb5e95431c00eef20b (api)

<Author: zakaria <zeakoalslmy@gmail.com

Date: Thu Oct 16 23:34:26 2025 +0300

CheakLangApi

commit 262de4b...: الرقم التعريفي لهذا التغيير. لاحظ أن هذا نفس الرقم الذي رأيته في سطر Merge في ال commit السابق.

(api): هذا المؤشر يخبرك أن هذا ال commit كان في الأصل رأس (آخر تغيير) لفرع api قبل أن تقوم بدمجه.

CheakLangApi: الرسالة التي وصفت بها هذا التغيير.

الخلاصة: هذا هو آخر عمل قمت به في فرع api قبل أن تقرر دمجها في master.

٣. ال Commits الباقية: تاريخ المشروع

code

Code

commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c

<Author: zakaria <zeakoalslmy@gmail.com

Date: Thu Oct 16 00:42:40 2025 +0300

feat: Initial API setup with authentication and core resources

...

commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b

<Author: zakaria <zeakoalslmy@gmail.com

Date: Tue Sep 23 00:26:02 2025 +0300

feat: Implement CRUD for categories, posts, and pages

...

كل الفقرات التالية تتبع نفس النمط. هي عبارة عن سجل لجميع الإضافات والميزات التي عملت عليها في الماضي، مرتبة من الأحدث إلى الأقدم. يمكنك أن تقرأ رسائل ال commit مثل:

"Initial API setup" (إعداد ال API الأولي).

"Implement CRUD for categories" (تنفيذ عمليات CRUD للأقسام).

"Manage department users" (إدارة مستخدمي القسم).

ستلاحظ أيضًا وجود فروع أخرى مثل (catgory) بجانب بعض ال commits، وهذا يعني أن هذا ال commit كان في وقت ما هو آخر تغيير في ذلك الفرع.

ملخص بسيط جداً

تخيل أنك تكتب كتابًا. كل commit هو فصل جديد أو تعديل تقوم بحفظه. والأمر git log يسمح لك بتصفح فهرس الكتاب بالترتيب العكسي، لترى كل فصل كتبته، ومن كتبه، ومتى. عملية الدمج التي قمت بها تشبه أخذ فصل من مسودة (فرع api) وإضافته بشكل رسمي إلى الكتاب الرئيسي (فرع master).

Sources

help

careerkarma.com

freecodecamp.org

git-scm.com

tencentcloud.com

deployhq.com

stackoverflow.com

github.com

reddit.com

Google Search Suggestions

Display of Search Suggestions is required when using Grounding with Google



Search. Learn more

Google logo

User

commit 262de4b375b1fb2c936962fb5e95431c00eef20b (api) هل يمكن حذف هذا

Author: zakaria zeakoalslmy@gmail.com

Date: Thu Oct 16 23:34:26 2025 +0300

code

Code

CheakLangApi

commit 262de4b...: الرقم التعريفي لهذا التغيير. لاحظ أن هذا نفس الرقم الذي رأيته في سطر Merge في ال commit السابق.

(api): هذا المؤشر يخبرك أن هذا ال commit كان في الأصل رأس (آخر تغيير) لفرع api قبل أن تقوم بدمجه.

CheakLangApi: الرسالة التي وصفت بها هذا التغيير.

warning

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

نعم، يمكن حذفه تقريباً، ولكن لا يُنصح بذلك على الإطلاق في حالتك، وهو يعتبر ممارسة سيئة وخطيرة.

دعني أشرح لك السبب، ثم أعطيك البديل الصحيح والأمن لما تريد فعله.

لماذا حذف هذا ال Commit فكرة سيئة جداً؟

لقد تم دمجه بالفعل: هذا ال (262de4b) commit لم يعد مجرد تغيير في فرع api. لقد أصبح جزءاً أساسياً من تاريخ فرع master

من خلال عملية الدمج (Merge commit ddb7a1c). تخيل أنك بنيت جداراً من الطوب (فرع master)، وهذا ال commit هو

إحدى الطوب في الأساس. إذا حاولت سحب هذه الطوبة الآن، فإن الجدار بأكمله سيتضرر.

التاريخ مشترك مع الآخرين (Pushed History): بما أن سجل git log يظهر origin/master بجانب ال commit

الخاص بالدمج، فهذا يعني أنك على الأغلب قمت بدفع (push) هذه التغييرات إلى مستودع بعيد (مثل GitHub). بمجرد دفع التغييرات،

يصبح التاريخ "تاريخاً عاماً". تغييره أو حذفه يشبه محاولة استرجاع كل نسخة من كتاب تم نشره لتغيير صفحة فيه. سيسبب هذا مشاكل

وفوضى كبيرة لأي شخص آخر يعمل على المشروع أو قام بسحب (pull) آخر التحديثات.

البديل الصحيح والأمن: git revert

إذا كنت تريد "إلغاء" التغييرات التي حدثت في هذا ال commit (لأنها مثلاً سببت خطأ)، الطريقة الصحيحة ليست الحذف، بل "العكس".

الأمر git revert يقوم بإنشاء commit جديد يقوم بعكس كل التغييرات التي حدثت في ال commit الذي تريد إلغاؤه.

فكر فيها مثل سجل حسابات بنكي:

الحذف (الطريقة الخطأ): كأنك تمحو معاملة خاطئة من السجل، وهذا يجعل السجل غير موثوق.

العكس (الطريقة الصحيحة): كأنك تضيف معاملة جديدة تعكس المعاملة الخاطئة (إذا سحبت ١٠٠ بالخطأ، تقوم بإيداع ١٠٠ لتصحيح

الرصيد). السجل يظل كاملاً وواضحاً.

كيف تستخدم git revert؟

لتطبيق هذا على حالتك، قم بتنفيذ الأمر التالي:

code

Shell

git revert 262de4b375b1fb2c936962fb5e95431c00eef20b

ماذا سيحدث بعد تنفيذ الأمر؟

سيقوم Git بتحديد كل التغييرات التي تمت في ال commit 262de4b ويعكسها.

سيفتح لك محرر النصوص لكي تكتب رسالة لل commit الجديد. الرسالة الافتراضية ستكون شيئاً مثل:

code

Code

"Revert "CheakLangApi

.This reverts commit 262de4b375b1fb2c936962fb5e95431c00eef20b

احفظ الرسالة وأغلق المحرر (بنفس الطريقة: Esc ثم wq).

النتيجة:

سيظل ال commit القديم CheakLangApi موجوداً في تاريخك (للشفافية).

سيتم إنشاء commit جديد فوقه يلغي تأثيره تماماً، وكأنه لم يكن موجوداً من ناحية الكود.

سيبقى تاريخ مشروعك نظيفاً ومنطقياً وآمناً للجميع.  
خلاصة:

الخاصية الحذف (خطير - git reset أو rebase) العكس (git revert - آمن)  
ماذا يفعل؟ يغير ويمحو التاريخ بشكل دائم. ينشئ تاريخاً جديداً يلغي التاريخ القديم.  
متى يُستخدم؟ فقط على commits محلية لم يتم دفعها (push) أبداً. دائماً لإلغاء commits تم دفعها أو مشاركتها.  
النتيجة في حالتك سيؤدي إلى تدمير تاريخ الفرع ويسبب مشاكل لك وللآخرين. سيحل المشكلة بأمان ويحافظ على سلامة التاريخ.  
نصيحتي النهائية: استخدم git revert دائماً لإلغاء التغييرات في فرع مشترك أو تم دفعه.

```
User
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. جميع الحقوق محفوظة.
C:\xampp\htdocs\my_blog2>git branch
api
category
master
C:\xampp\htdocs\my_blog2>git merge api
.fatal: You have not concluded your merge (MERGE_HEAD exists)
.Please, commit your changes before you merge
C:\xampp\htdocs\my_blog2>git status
On branch master
.'Your branch is up to date with 'origin/master
.All conflicts fixed but you are still merging
(use "git commit" to conclude merge)
:Changes to be committed
new file:   app/Http/Controllers/api/dashboard/CategoriesController.php
new file:   app/Http/Controllers/api/dashboard/SettingController.php
new file:   app/Http/Controllers/api/loginController.php
new file:   app/Http/Controllers/loginController.php
new file:   app/Http/Middleware/CheakLangApi.php
new file:   app/Http/Requests/categories/StoreCategoryRequest.php
new file:   app/Http/Resources/CatgoryCollection.php
new file:   app/Http/Resources/CatgoryResource.php
new file:   app/Http/Resources/PostResource.php
new file:   app/Http/Resources/SettingResource.php
modified:   app/Models/User.php
modified:   routes/api.php
C:\xampp\htdocs\my_blog2>git commit
hint: Waiting for your editor to close the file... error: There was a problem
.'with the editor 'vi
.Please supply the message using either -m or -F option
C:\xampp\htdocs\my_blog2>w
,w' is not recognized as an internal or external command
.operable program or batch file
C:\xampp\htdocs\my_blog2>rm /c/xampp/htdocs/my_blog2/.git/.COMMIT_EDITMSG.swp
,rm' is not recognized as an internal or external command
.operable program or batch file
C:\xampp\htdocs\my_blog2>del .git.COMMIT_EDITMSG.swp
C:\xampp\htdocs\my_blog2>git commit
'[master ddb7a1c] Merge branch 'api
C:\xampp\htdocs\my_blog2>git push origin master
.Enumerating objects: 159, done
.Counting objects: 100% (159/159), done
Delta compression using up to 4 threads
.Compressing objects: 100% (106/106), done
```

```

        .Writing objects: 100% (113/113), 18.55 KiB | 211.00 KiB/s, done
        Total 113 (delta 58), reused 0 (delta 0), pack-reused 0
    .remote: Resolving deltas: 100% (58/58), completed with 31 local objects
        To https://github.com/zakariaalslmy2/blog.git
            db09873..ddb7a1c  master -> master
    C:\xampp\htdocs\my_blog2>git log
commit ddb7a1cde02fd9fd62cb296a401275478bfd7f4d (HEAD -> master, origin/master)
    Merge: db09873 262de4b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Fri Oct 17 00:44:04 2025 +0300
        code
        Code
            'Merge branch 'api
    commit 262de4b375b1fb2c936962fb5e95431c00eef20b (api)
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Thu Oct 16 23:34:26 2025 +0300
        code
        Code
            CheakLangApi
    commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Thu Oct 16 00:42:40 2025 +0300
        code
        Code
        feat: Initial API setup with authentication and core resources

    .Create API v1 routes for settings, categories, and authentication -
    Implement login functionality using Laravel Sanctum for secure API -
        .access
    .Add SettingController and CategoriesController to expose site data -
        ,Introduce API Resources (SettingResource, CatgoryResource -
            .PostResource) to format JSON responses
    Implement a middleware (CheakLangApi) to handle API language -
        .localization based on headers
    commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Tue Sep 23 00:26:02 2025 +0300
        code
        Code
        feat: Implement CRUD for categories, posts, and pages

    .Add functionality to create, read, update, and delete categories -
    .Add functionality to create, read, update, and delete posts -
        .Create the main index page for the website -
    commit 910a78e001198eef81b7cd611b34b97d5336ae1b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Mon Jun 16 17:08:12 2025 +0300
        code
        Code
        feat: Manage department users

    Implement full CRUD (Create, Read, Update, Delete) functionality for
        users within the department section
    Implement full CRUD (Create, Read, Update, Delete) functionality for

```

```

        users within the department section
commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Thu Oct 16 00:42:40 2025 +0300
        code
        Code
    feat: Initial API setup with authentication and core resources

    .Create API v1 routes for settings, categories, and authentication -
    Implement login functionality using Laravel Sanctum for secure API -
        .access
    .Add SettingController and CategoriesController to expose site data -
        ,Introduce API Resources (SettingResource, CatgoryResource -
        .PostResource) to format JSON responses
    Implement a middleware (CheakLangApi) to handle API language -
        .localization based on headers
commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Tue Sep 23 00:26:02 2025 +0300
        code
        Code
    feat: Implement CRUD for categories, posts, and pages

    .Add functionality to create, read, update, and delete categories -
    .Add functionality to create, read, update, and delete posts -
    .Create the main index page for the website -
commit 910a78e001198eef81b7cd611b34b97d5336ae1b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Mon Jun 16 17:08:12 2025 +0300
        code
        Code
        feat: Manage department users

    Implement full CRUD (Create, Read, Update, Delete) functionality for
        users within the department section
commit 5adff9bc89c6118ec6ab38d7d17da1f108b81fa1
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Sun Mar 9 08:34:20 2025 +0300
        code
        Code
    chore: Initialize database schema for my_blog

    "Create all the necessary tables for the initial setup of the "my_blog
        project.s
commit db0987378ea2b8aa04981a92521f0d82624a8e77 (catgory)
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Tue Sep 23 00:26:02 2025 +0300
        code
        Code
    i am crud created catgories and post and page index web site
commit ae7191ca128812a40b0e5fed51d4786f72b9f58b
    Author: zakaria zeakoalslmy@gmail.com
    Date:   Mon Jun 16 17:08:12 2025 +0300
        code

```

```

Code
today i compleated department users add user and edite and delete
commit 7c0402f5913dd3c08569bf5ad1cb4ec984fcbec3
Author: zakaria zeakoalslmy@gmail.com
Date: Sun Mar 9 08:34:20 2025 +0300
code
Code
project my_blog today i created all tables
(END)
لماذا تظهر هاذي الرسائل project my_blog today i created all tables وانا قد قمتو بتعديلها اقصد هل
هذا يعتبر مشكلة او هاذا هو نمط الجيت لقد قمتو بتعديلها الي
chore: Initialize database schema for my_blog
code
Code
"Create all the necessary tables for the initial setup of the "my_blog
project.s
warning
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts
chevron_right
أهلاً بك، هذا سؤال ممتاز جداً ويكشف نقطة مهمة وأساسية في طريقة عمل Git.
الجواب المختصر: لا، هذه ليست مشكلة. هذا هو السلوك الطبيعي لـ Git عندما تقوم بـ "تعديل" تاريخك، وما تراه هو نتيجة وجود فرعين
(master و catgory) يعتمدان على تاريخين مختلفين قليلاً.
دعنا نشرح بالتفصيل لماذا ترى الرسائلتين.
Git لا "يعدل" بل "يستبدل"
عندما تستخدم أمرًا لتعديل commit قديم (مثل git commit --amend أو git rebase)، فإن Git في الحقيقة لا يرجع
للماضي ويمسح الـ commit القديم. بدلاً من ذلك، هو يقوم بالآتي:
يأخذ التغييرات من الـ commit القديم.
ينشئ commit جديد تمامًا بالرسالة الجديدة التي كنتيتها.
يجعل الفرع الحالي (master في حالتك) يشير إلى هذا الـ commit الجديد.
يترك الـ commit القديم الأصلي مهجورًا في الخلفية.
تحليل ما حدث في مشروعك
بناءً على سجل git log الذي عرضته، هذا هو ما حدث على الأغلب:
في البداية، قمت بإنشاء الـ commit ذو الرقم Vc0402f مع الرسالة القديمة: "project my_blog today i created
all tables".
بعد ذلك، قررت أن الرسالة غير احترافية وأردت تعديلها. فقامت باستخدام أمر مثل git commit --amend.
قام Git بإنشاء commit جديد وهو 0adff9b بنفس التغييرات ولكن بالرسالة الجديدة والمحسنة: "chore: Initialize
database schema for my_blog".
الفرع الرئيسي master أصبح يتبع هذا التاريخ الجديد والمُحسن.
إذًا، لماذا ما زلت ترى الـ commit القديم (Vc0402f)؟
السبب هو أن لديك فرعًا آخر، وهو catgory، لا يزال يعتمد على التاريخ القديم.
لاحظ في سجل git log هذين السطرين:
code
Code
// الـ commit الجديد في فرع master
commit 5adff9bc89c6118ec6ab38d7d17da1f108b81fa1
...
chore: Initialize database schema for my_blog
// الـ commit القديم الذي يعتمد عليه فرع catgory

```

```

commit db0987378ea2b8aa04981a92521f0d82624a8e77 (catgory)
...
i am crud created catgories and post and page index web site
أمر git log (بدون تحديد فرع) يعرض لك تاريخ كل الفروع في مشروعك. لذلك، هو يعرض لك مسار التاريخ الصحيح لفرع
master، ويعرض لك أيضًا مسار التاريخ القديم الذي ما زال فرع catgory يشير إليه.
يمكنك تخيل تاريخك كشجرة بها تفرعات:
code
Code
(التاريخ الجديد والمُحسّن)
9١٠a -- 6bab -- 5adf -- ... -- ddb7a1c (HEAD -> master) -- ...
\
\ (التاريخ القديم)
ae71 -- 7c04 -- db09 (catgory) --
فرع master سلك الطريق العلوي (المُعدل)، بينما فرع catgory ما زال على الطريق السفلي (القديم).
هل هذا يسبب مشكلة؟ وكيف تصلحه؟
هو لا يسبب مشكلة الآن، لكنه قد يسبب تعارضات مربكة في المستقبل إذا حاولت دمج فرع catgory في master، لأنه سيحاول إعادة
إدخال الـ commits القديمة مرة أخرى.
الحل الصحيح هو تحديث فرع catgory ليعتمد على نفس تاريخ master الجديد. ويتم ذلك باستخدام git rebase.
نفذ الأوامر التالية:
الانتقال إلى فرع catgory:
code
Shell
git checkout catgory
إعادة بناء فرع catgory فوق master:
code
Shell
git rebase master
هذا الأمر يعني: "يا Git، خذ كل التغييرات التي قمت بها في فرع catgory، وانقلها لتصبح فوق آخر تحديث موجود في فرع
master". سيؤدي هذا إلى التخلص من الاعتماد على التاريخ القديم.
بعد تنفيذ هذه الأوامر، إذا قمت بكتابة git log مرة أخرى، ستجد أن الـ commits القديمة ذات الرسائل السيئة قد اختفت تمامًا من
السجل لأن لا يوجد أي فرع يشير إليها بعد الآن.
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts

chevron_right
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts

chevron_right
إجابة مباشرة:
لا يعتبر عيبًا خطيرًا يعطل عملك حاليًا، ولكنه يعتبر ممارسة غير جيدة وتاريخًا للمشروع غير نظيف، ويمكن أن يسبب مشاكل مربكة في
المستقبل.
دعنا نفصل الإجابة:
لماذا لا يعتبر عيبًا خطيرًا الآن؟
فرعك الرئيسي (master) نظيف ويحتوي على التاريخ الصحيح والمُحسّن.
يمكنك الاستمرار في العمل على فرع master وإضافة commits جديدة عليه دون أي مشاكل.
المشروع نفسه يعمل والكود سليم.
لماذا يعتبر ممارسة غير جيدة (عيبًا مستقبليًا)؟

```

هنا تكمن المشكلة الحقيقية: مشاكل عند الدمج المستقبلي: هذه هي أكبر مشكلة. تخيل أنك بعد شهر من الآن، قمت بتطوير ميزة مهمة في فرع category وتريد دمجها في master. عندما تحاول الدمج، سيحاول Git دمج فرع يعتمد على تاريخ قديم (category) مع فرع يعتمد على تاريخ جديد (master). هذا قد يؤدي إلى:

تعارضات (Conflicts) غير متوقعة وصعبة الحل. قد يحاول Git إعادة إدخال الـ commit القديم مرة أخرى إلى master، مما يسبب فوضى في تاريخ المشروع. سيصبح شكل تاريخ المشروع معقدًا وغير مفهوم (يُعرف بـ "spaghetti history"). إرباك لك وللآخرين: أي شخص (بما في ذلك أنت في المستقبل) ينظر إلى سجل المشروع الكامل سيرى نسختين من نفس التغيير برسالتين مختلفتين. هذا يسبب إرباكًا ويجعل من الصعب تتبع التغييرات وفهم تاريخ المشروع بشكل صحيح. تاريخ غير نظيف: الهدف من Git هو الحفاظ على سجل تاريخي نظيف وواضح ودقيق. وجود "commits مهجورة" مثل هذه يعتبر ضد هذه الممارسة.

خلاصة وتوصية هل هو عيب؟ نعم، يمكن اعتباره عيبًا في "نظافة" و "صحة" تاريخ المستودع الخاص بك. هل يجب أن تصلحه؟ نعم، يُفضل بشدة أن تصلحه الآن. الأمر بسيط جدًا ولن يستغرق أكثر من دقيقة، وسيوفر عليك الكثير من الصداق في المستقبل. كلما تركت هذا الوضع لفترة أطول وأضفت المزيد من التغييرات، أصبح إصلاحه أكثر تعقيدًا. الإصلاح بسيط جدًا كما ذكرنا سابقًا:

```
code
Shell
# ١. انتقل إلى الفرع الذي يعتمد على التاريخ القديم
git checkout category
```

```
# ٢. أعد بناء هذا الفرع ليعتمد على التاريخ الجديد والصحيح في master
git rebase master
بعد تنفيذ هذين الأمرين، سيصبح تاريخ مشروعك نظيفًا ومنطقيًا تمامًا، وسيختفي الـ commit القديم.
```

```
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts
```

```
chevron_right
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts
```

```
chevron_right
User
عفو لم أفهم انا ما اقصد هو ال comit الذي Author: zakaria zeakoalslmy@gmail.com
Date: Sun Mar 9 08:34:20 2025 +0300
```

```
code
Code
project my_blog today i created all tables
عند التعديل هل يبقى تاريخ الكوميت القديم او يستبدل بجديد
warning
```

```
Model
ThinkingThoughts
(experimental)
Auto
Expand to view model thoughts
```

```
chevron_right
فهمت سؤالك الآن بشكل دقيق جدًا. شكرًا للتوضيح. هذا هو جوهر الموضوع.
```

الجواب المباشر:

في العادة، عندما تعدل رسالة commit (باستخدام `git commit --amend`)، فإن Git ينشئ commit جديد بتاريخ جديد (تاريخ ووقت التعديل).

ولكن، بناءً على سجل `git log` الذي أرسلته، هذا لم يحدث معك!

دعنا نرى الدليل من سجلك:

الكوميت المُعدل (الجديد والجيد):

code

Code

```
commit 5adfff9bc89c6118ec6ab38d7d17da1f108b81fa1
```

```
Date: Sun Mar 9 08:34:20 2025 +0300
```

```
...chore: Initialize database schema for my_blog
```

الكوميت الأصلي (القديم والسيء):

code

Code

```
commit 7c0402f5913dd3c08569bf5ad1cb4ec984fcbec3
```

```
Date: Sun Mar 9 08:34:20 2025 +0300
```

```
project my_blog today i created all tables
```

لاحظ أن التاريخ والوقت متطابقان تمامًا في الاثنين!

ماذا يعني هذا؟

هذا يعني أن الطريقة التي استخدمتها لتعديل الرسالة كانت طريقة "ذكية" جعلت Git يحافظ على التاريخ الأصلي للعمل. هذا غالبًا ما يحدث عند استخدام أوامر متقدمة مثل `git rebase -i` واختيار `reword` (إعادة صياغة)، حيث يفهم Git أنك تغير "الوصف" فقط وليس "العمل" نفسه، فيحتفظ بتاريخ المؤلف الأصلي.

إذًا، للإجابة على سؤالك بوضوح:

عند التعديل هل يبقى تاريخ الكوميت القديم أو يستبدله بجديد؟

في حالتك أنت، بقي تاريخ الكوميت القديم ولم يتم استبداله. وهذا أفضل سيناريو ممكن، لأنه يحافظ على مصداقية تاريخ مشروعك. الكوميت الجديد والمُحسن لا يزال يُظهر أنه تم إنشاؤه في ٩ مارس ٢٠٢٥.

الخلاصة النهائية لمشكلتك:

هل فقدت تاريخك القديم؟ لا، الكوميت المُعدل يحمل نفس التاريخ الصحيح.

إذًا ما هي المشكلة المتبقية؟ المشكلة هي فقط وجود نسخة مكررة من هذا الكوميت (النسخة ذات الرسالة القديمة) والتي لا يزال فرع

catgory يتذكرها.

ماذا سيفعل `git rebase`؟ كل ما سيفعله هو جعل فرع catgory ينسب النسخة المكررة القديمة، ويتذكر فقط النسخة الجيدة (التي

تحمل تاريخك الصحيح والقديم).

يمكنك المتابعة بثقة تامة. أنت لن تخسر أي شيء من تاريخك، بل ستنظفه فقط.

Model

ThinkingThoughts

(experimental)

Auto

Expand to view model thoughts

chevron\_right

بالتأكيد. سأرشدك خطوة بخطوة لتنفيذ هذا الأمر بأمان. العملية بسيطة جدًا.

تذكر أن ما سنفعله هو كأننا نقول لفرع catgory: "انتقل من الطريق القديم الذي يحتوي على الكوميت المكرر، وقفز إلى نهاية الطريق

الجديد والمُحسن الذي يسير عليه فرع master".

إليك الخطوات بالتفصيل. قم بتنفيذها أمرًا تلو الآخر في الطرفية (Command Prompt):

الخطوة ١: انتقل إلى الفرع الذي تريد إصلاحه

أولاً، يجب أن تنتقل إلى فرع catgory لأن هو الفرع الذي يعتمد على التاريخ القديم.

code

Shell

```
git checkout catgory
```

ماذا سيحدث؟ ستخبرك الطرفية أنها انتقلت بنجاح إلى الفرع:

```
'Switched to branch 'catgory'
```

الخطوة ٢: قم بتنفيذ عملية الـ Rebase

هذا هو الأمر الرئيسي. معناه الحرفي هو: "أعد بناء هذا الفرع (catgory) ليكون أساسه (base) هو فرع master".

code



Shell

```
git rebase master
```

ماذا سيحدث؟

سيقوم Git تلقائيًا بأخذ كل الـ commits الفريدة في فرع category (التي لا توجد في master) ويضعها فوق آخر تحديث في master. بما أن category يعتمد على تاريخ قديم، فإن هذه العملية ستجعله يتجاهل هذا التاريخ القديم ويعتمد على التاريخ الجديد النظيف الموجود في master.

النتائج المتوقعة سيكون شيئًا مثل:

```
.Successfully rebased and updated refs/heads/category
```

الخطوة ٣: تحقق من النتيجة

الآن، دعنا نتأكد من أن كل شيء أصبح نظيفًا. اعرض سجل التاريخ مرة أخرى:

code

Shell

```
git log
```

ماذا ستري الآن؟

يجب أن تلاحظ أن الـ commit القديم ذو الرسالة السيئة قد اختفى تمامًا من السجل. لم يعد له وجود لأن لا يوجد أي فرع يشير إليه الآن. ستجد فقط النسخة الجيدة من الـ commit بتاريخه الصحيح (9 مارس).

الخطوة ٤: ارجع إلى فرعك الرئيسي (اختياري ولكن موصى به)

لقد انتهينا من إصلاح category. من الجيد دائمًا العودة إلى فرع master لمواصلة عملك الرئيسي.

code

Shell

```
git checkout master
```

ماذا سيحدث؟ ستعود إلى فرع master:

```
'Switched to branch 'master'
```

ملخص الأوامر للتنفيذ:

code

Shell

```
git checkout category
```

```
git rebase master
```

```
git checkout master
```

ماذا لو حدث تعارض (Conflict)؟

في حالتك، هذا غير محتمل، ولكن إذا حدث ورأيت رسالة "Conflict"، فلا تقلق. هذا يعني فقط أن Git لم يستطع تطبيق تغييراتك تلقائيًا. إذا لم تكن متأكدًا مما يجب فعله، يمكنك دائمًا إلغاء كل شيء والعودة إلى ما كنت عليه بأمان باستخدام الأمر:

code

Shell

```
git rebase --abort
```

هذا الأمر هو "زر الأمان" الخاص بك. سيلغي عملية الـ rebase ويعيد كل شيء كما كان قبل أن تبدأ.

User

```
C:\xampp\htdocs\my_blog2>git checkout category
```

```
'Switched to branch 'category'
```

```
C:\xampp\htdocs\my_blog2>git rebase master
```

```
.Current branch category is up to date
```

```
C:\xampp\htdocs\my_blog2>git checkout master
```

```
'Switched to branch 'master'
```

```
.'Your branch is up to date with 'origin/master'
```

```
C:\xampp\htdocs\my_blog2>git log
```

```
,commit ddb7a1cde02fd9fd62cb296a401275478bfd7f4d (HEAD -> master, origin/master
```

```
(category
```

```
Merge: db09873 262de4b
```

```
Author: zakaria zeakoalslmy@gmail.com
```

```
Date: Fri Oct 17 00:44:04 2025 +0300
```

code

Code

```
'Merge branch 'api'
```

```
commit 262de4b375b1fb2c936962fb5e95431c00eef20b (api)
```

Author: zakaria zeakoalslmy@gmail.com  
Date: Thu Oct 16 23:34:26 2025 +0300  
code  
Code

CheakLangApi  
commit 328b9ca1e993abb4bf25825df50c8c3598b35d5c  
Author: zakaria zeakoalslmy@gmail.com  
Date: Thu Oct 16 00:42:40 2025 +0300  
code  
Code

feat: Initial API setup with authentication and core resources

.Create API v1 routes for settings, categories, and authentication -  
Implement login functionality using Laravel Sanctum for secure API -  
.access

.Add SettingController and CategoriesController to expose site data -  
,Introduce API Resources (SettingResource, CatgoryResource -  
.PostResource) to format JSON responses

Implement a middleware (CheakLangApi) to handle API language -  
.localization based on headers

commit 6bab0592462d51646fc37c7aa6f6e3c49f026d5b  
Author: zakaria zeakoalslmy@gmail.com  
Date: Tue Sep 23 00:26:02 2025 +0300  
code  
Code

feat: Implement CRUD for categories, posts, and pages

.Add functionality to create, read, update, and delete categories -  
.Add functionality to create, read, update, and delete posts -  
.Create the main index page for the website -

commit 910a78e001198eef81b7cd611b34b97d5336ae1b  
Author: zakaria zeakoalslmy@gmail.com  
Date: Mon Jun 16 17:08:12 2025 +0300  
code  
Code

feat: Manage department users

Implement full CRUD (Create, Read, Update, Delete) functionality for  
users within the department section

commit 5adff9bc89c6118ec6ab38d7d17da1f108b81fa1  
Author: zakaria zeakoalslmy@gmail.com  
Date: Sun Mar 9 08:34:20 2025 +0300  
code  
Code

chore: Initialize database schema for my\_blog

"Create all the necessary tables for the initial setup of the "my\_blog  
project.s

commit db0987378ea2b8aa04981a92521f0d82624a8e77  
Author: zakaria zeakoalslmy@gmail.com  
Date: Tue Sep 23 00:26:02 2025 +0300  
code  
Code

i am crud created catgories and post and page index web site

commit ae7191ca128812a40b0e5fed51d4786f72b9f58b  
Author: zakaria zeakoalslmy@gmail.com  
Date: Mon Jun 16 17:08:12 2025 +0300  
code  
Code

today i compleated department users add user and edite and delete  
commit 7c0402f5913dd3c08569bf5ad1cb4ec984fcbec3  
Author: zakaria zeakoalslmy@gmail.com  
Date: Sun Mar 9 08:34:20 2025 +0300  
code  
Code

project my\_blog today i created all tables  
(END)  
warning  
Model

ThinkingThoughts  
(experimental)

Auto  
Expand to view model thoughts

chevron\_right