Great novels are rarely written in one go and usually

endure several drafts before

the author is satisfied with the outcome.

Of course, there's always the possibility that

an idea from an earlier draft

will sound good again later.

But without a system of organization,

it can be difficult to find where this idea is located.

Programming is no different and

sometimes you need to revisit old code.

In this video, I'll guide you

through how to use git diff to

compare changes across your files branches and commits.

You probably know that the git status command

tells you which of your files have been changed.

The git diff command it goes a step

further and tells you what exactly these changes are.

When used together, you can

think of them as a file system.

Git status tells you the file names,

but to open the file and see the contents,

you'll need to use git diff.

To demonstrate let's say you have

a text file named CITIES.txt,

which contains the names of cities you visited.

You've been updating the list

during a tour of South America,

but upon returning home,

you've lost track of what you've

recorded. What can you do?

Well, this is a situation where

git diff makes itself useful.

Git diff will compare the previous version of

the file with your current one to find any differences.

It will then tell you specifically what content has been

removed as well as what

content has been added to the file.

In your CITIES.txt file,

git diff would tell you that you

removed one city that was in

version A and then added

a new one that appears in version B.

Now that you've had a basic explanation,

Let's go into a more detailed example.

In this video, I'll show you how you can use git diff.

Diff is used to make comparisons

against files on your local repository.

It can also be used against commits and against branches.

I'll start with a simple example.

When I go into my local repository,

I'll find a file called READ

ME that I'd like to update slightly.

You can do this with any editor such as VS code.

I can also do this by executing

the Vim command to enter the file for editing,

remove a few words,

and then save it.

If you would like to learn more about Vim,

there is a link to an additional reading

at the end of this lesson.

Next, I'm going to use

the git diff tool to compare

the updated file against the head.

Because we haven't yet completed a commit.

It's not available for a comparison

against another commit.

I'll input git diff passing the head as the first option,

and then finally the file name.

This then returns an output showing

the changes that occurred in each file.

Here, the line starting with

a minus symbol represents what it originally was.

While the line with a plus symbol shows what it is now.

My example tells me that the words

minor update have been removed.

In addition to individual files,

you can also make comparisons against previous commits.

I'll start by using

the git log command to display my history of commits.

I'll also use the pretty flag here

so that each one is shown in one line.

The pretty flag is used by

developers to make the output more readable.

Each commit has its own ID code.

So I'll perform a git diff command on the codes

from the most recent commit and from the very first one.

Git hub will go through all the files,

note all the changes that have occurred,

and return the differentiation between the two.

Finally, one more way of using git diff that I'll show

you is how to use it for

making comparisons against branches.

If I perform the command git branch,

it will display all the branches

that are available in the repository.

I can then use the git diff command

to pass in my main branch,

followed by my feature branch as the second option.

Once again, this will display

all the changes that have occurred between the two.

This shows that my feature branch

has the previous update,

while the main branch has the most recent one.

These are a few examples of how you can use git diff.

In this video, you've learned how to

use git diff command to keep

track of changes across

your files, branches and commits.

This tool can help you to stay on top of

updates and avoid mistakes or overlap.

See you next time.

You have returned to development of your hotel booking app after some time away, and want to see the last changes you made. At which levels can you compare the differences between versions using Git diff?

Choose all that apply:



Individual files

Correct

That's correct! Individual files, branches, and commits can all be compared with Git diff.



Commits



Branches

Correct

That's correct! Individual files, branches, and commits can all be compared with Git diff.



Author of changes

You didn’t select all the correct answers

3:56/4:22

** Transcript**

**Interactive Transcript - Enable basic transcript mode by pressing the escape key**

You may navigate through the transcript using tab. To save a note for a section of text press CTRL + S. To expand your selection you may use CTRL + arrow key. You may contract your selection using shift + CTRL + arrow key. For screen readers that are incompatible with using arrow keys for shortcuts, you can replace them with the H J K L keys. Some screen readers may require using CTRL in conjunction with the alt key

Great novels are rarely written in one go and usually

endure several drafts before

the author is satisfied with the outcome.

Of course, there's always the possibility that

an idea from an earlier draft

will sound good again later.

But without a system of organization,

it can be difficult to find where this idea is located.

Programming is no different and

sometimes you need to revisit old code.

In this video, I'll guide you

through how to use git diff to

compare changes across your files branches and commits.

You probably know that the git status command

tells you which of your files have been changed.

The git diff command it goes a step

further and tells you what exactly these changes are.

When used together, you can

think of them as a file system.

Git status tells you the file names,

but to open the file and see the contents,

you'll need to use git diff.

To demonstrate let's say you have

a text file named CITIES.txt,

which contains the names of cities you visited.

You've been updating the list

during a tour of South America,

but upon returning home,

you've lost track of what you've

recorded. What can you do?

Well, this is a situation where

git diff makes itself useful.

Git diff will compare the previous version of

the file with your current one to find any differences.

It will then tell you specifically what content has been

removed as well as what

content has been added to the file.

In your CITIES.txt file,

git diff would tell you that you

removed one city that was in

version A and then added

a new one that appears in version B.

Now that you've had a basic explanation,

Let's go into a more detailed example.

In this video, I'll show you how you can use git diff.

Diff is used to make comparisons

against files on your local repository.

It can also be used against commits and against branches.

I'll start with a simple example.

When I go into my local repository,

I'll find a file called READ

ME that I'd like to update slightly.

You can do this with any editor such as VS code.

I can also do this by executing

the Vim command to enter the file for editing,

remove a few words,

and then save it.

If you would like to learn more about Vim,

there is a link to an additional reading

at the end of this lesson.

Next, I'm going to use

the git diff tool to compare

the updated file against the head.

Because we haven't yet completed a commit.

It's not available for a comparison

against another commit.

I'll input git diff passing the head as the first option,

and then finally the file name.

This then returns an output showing

the changes that occurred in each file.

Here, the line starting with

a minus symbol represents what it originally was.

While the line with a plus symbol shows what it is now.

My example tells me that the words

minor update have been removed.

In addition to individual files,

you can also make comparisons against previous commits.

I'll start by using

the git log command to display my history of commits.

I'll also use the pretty flag here

so that each one is shown in one line.

The pretty flag is used by

developers to make the output more readable.

Each commit has its own ID code.

So I'll perform a git diff command on the codes

from the most recent commit and from the very first one.

Git hub will go through all the files,

note all the changes that have occurred,

and return the differentiation between the two.

Finally, one more way of using git diff that I'll show

you is how to use it for

making comparisons against branches.

If I perform the command git branch,

it will display all the branches

that are available in the repository.

I can then use the git diff command

to pass in my main branch,

followed by my feature branch as the second option.

Once again, this will display

all the changes that have occurred between the two.

This shows that my feature branch

has the previous update,

while the main branch has the most recent one.

These are a few examples of how you can use git diff.

In this video, you've learned how to

use git diff command to keep

track of changes across

your files, branches and commits.

This tool can help you to stay on top of

updates and avoid mistakes or overlap.

See you next time.