DevOps - Module 4: Git For DevOps

DevOps

Module 4: Git For DevOps

Video: Developer GitHub Workflow

Whenever developer starts doing the work,
the developer work first it starts in his laptop.
So a developer who is going
to write the code in our scenario,
who is developer means technically right now,
my developer is this guy,
remember in the last session, and he is working on my
projects and some task was already given to him.
What was the task? To design a web
application page, isn't it?
This is what he's supposed to write.
Uh, now don't think from a DevOps perspective,
think from a developer perspective.
He's the one who's going to write the
code and how he's working.
Just I want you to observe step by step. Okay? Let's see.
Now what I'm going to do,
this is my developer laptop.
What the point is, so this mission is my developer laptop.
Right now what you're saying, you say a simple Windows
machine he purchased
and uh, I would say he got this mission from the
company to go and implement.
Uh, now in this particular mesh, he's going
to set up some tools because he's a developer, right?

Let's see what he's doing. Actually right now.
He's going to install GI Bash,
which is technically GI client.
I already said no reason for pushing you towards GI Bashes.
For this reason. All later on we'll be using Git operations
from here along with the command line.
And also we are going to install one ID
integrated development environment.
And I said, right, it'll provide what you said,
the programmers a comprehensive way to go and ride the boat.
Okay? An ID consists of code build automation tools
and also some DataWorks.
Uh, in this particular regard, we are working
with a software callers, visual Studio code or vs coding.
Simple terms. Uh, these two tools are required for a dev.
Uh, this might be common for everyone, any developer,
but this might change based upon the preference.
Uh, if you are a Python developer,
most probably you might use something called as.
Okay? Maybe if you are a Java developer,
you might use something like Intelligent
or something like Eclipse based upon your part.
The programming language you're working with, it defines
so here with respect to MON that we are going to work with.
So we are using something called
as Visual Studio Code, right?
It's developer choice. You don't need
to worry about for DevOps engineers.
Any editor is fine, no issues at all.
But in terms of Visual Studio Code, it's very light,
very lightweight actually for that reason we are concerned.
Okay? Right. So now let's see. What is developer doing?
Is a developer is going to install both those softwares.
One is Visual Studio Code, the other one is GI Bash.

You already know this thing.

We already installed it in the initial sessions,

but just showing you from the developer point of view today,

64 bit,

that's our GI Bash right now.

I just, the phone bit,

I'll just keep it as it.

This is where I'm going to perform some GI operations.

How we can confirm, I'll run the GI command.

I'll just verify if it is going to be present or not.

You can see all the commands coming up, right?

Meaning basically we can use Git here.

That's what the conclusion is.

Along with Git rate, I'm also going to install, uh,

one more software called as Visual Studio Code.

Let's say Visual Studio code. Also,

I'll say launch it,

this how it looks and maybe I'll just increase the font bit.

Okay,

five minutes.

Yes, I,

I think it's very useful there.

Okay? Yeah. Now as soon as the developer gets access

to Git Visual Studio Code,

he'll start writing the things, right?

So when he's writing the code, you need

to understand within his system there will be tracking

Because one developer, there can be 10 developers, right?

Each one of them will write different set

of files and codes.

So tracking should be enabled in every individual developer.

Meh. Uh, now this concept, you remember what was repository?

Actual storage of

what all your project files, isn't it?

And there versions.

Now GitHub repository contains overall

all the 10 developers work.

Every individual developer will have their own local

repository, terminology, local repository.

The one in GitHub is called us.

Technically remote repository.

Are you clear everyone? Local repository, remote repository.

Both are same, but one is overall. One is individual.

Are you clear everyone? So now

what already GitHub repositories created meaning what?

Remote repositories there. That remote repository.

Developers will get it into their laptop, right?

Then it is called as local repository, okay?

In this local repository, I'm gonna start writing the code.

Uh, now once they get the repository, within the system,

you'll be having three logical areas.

One is called as working area, staging area,

then local area.

Are you good? Everyone working staging local.

So for this, what I did, I have one image

to easily help you understand, okay?

I'll keep it in your document anyway.

Uh, first let's assume in this particular man,

it's clear it everyone, okay?

This draw it. You see this is GitHub

in which we created repository.

This is the one we created in the last session.

2 5 0 1 9 named, right? So assume this is the repository.

So in that repository, by default,

what branch will be there?

Main branch. That was clear, right?

Everyone according to the last class discussion.

Then our developer, we asked him to do what?

Some work and I said, I don't want him
to work on main branch because he might mess it up.
So we are not taking that, uh, what you say, uh,
a chance to make sure that something is corrected.
So for the reason what we protected the main branch
and we asked the developer
to create a new branch called as what?
Dev branch. So developer is the one who created dev branch.
From where? From the main branch,
which is there in the repository.
Uh, now you have two copies now main copy, dev copy.
Now the developer, what he does, this is his laptop.
Clear with this, the machine
that right now I showed you, right?
A Windows machine that is laptop. A developer will do this.
Step first. What is doing cloning?
Uh, generally when I say the word clone, what does it mean?
Making copies? Cloning. We just use this term, right?
So when I say clone, I'm making a copy
of the repository in my laptop.
So this copy right now is called local repository.
Uh, once I get a repository, you see this red box?
Red box you're seeing now this is repository. Now what?
He cannot work on the main branch rate.
He will work on which branch.
Second point you need to go, you clone the repository,
then you go to the dev branch.
In that, I said there will be three logical areas, working,
staging, local.
Are you good? Everyone working staging local.
So why did they give this partners?
I'll give you one simple explanation
first, then we'll come back.
So that, let's say you want to buy some products in Amazon,

you'll be having hundreds of products, isn't it?
So what is the first step
that you do whatever the products you like, what,
where do you keep them in which are maybe,
let's say cut whatever the products that you like, you go,
go ahead and just start moving them into car.
Now cart is technically what you say, a place
where you can take the call whether you want to buy it
or you want to move it to wishlist.
Agree on So meaning what?
It's an intermediate area to take the decision.
So whatever the products I don't like, I can move them back
to what you say.
My shelf, like Amazon bag. If I like them I can buy them.
So when the product is confirmed, when I buy, agree or not.
So when I buy the product, I'll get what an order id.
That order ID is used for me.
What to track where the product is.
Is it shipped in case any issue comes up?
If I want to make a written for all these things work,
the order ID is basically like a transaction.
I agree. Same parts.
In working area, you do all the work,
you have 10 different files.
For simplicity, I'm saying this 10 files,
you're waiting the changes.
Now do you think all the 10 works absolutely good? No.
You remember partial changes? Not at all.
Working changes, best working changes.
You have some code.
Something is partially working,
something is not at all working.
Something is absolutely working good.
Now the things which are not working are when you start the

work, it'll be always there in the working area.
Are you see these three things?
I kept here assume that there are three different files
I created where in the working goal, once I created it,
if I want to confirm them, I need to commit them.
There's a term use committing the changes.
If you want to commit the changes, you need
to be in the local area to commit them.
You can't directly move to local.
Whatever the changes you did, you stage them first.
But does it mean they're confirmed? No, they're not.
This staging area is like your car.
If you don't like in the future you can move that items back
to working area.
If you like, I'm a hundred percent confident
this is what I need.
Then what you do, you move it back to local.
When I put the things in local, we call it as committing.
If I put the changes over here, it's called stage.
When I commit, I'm going to get something called as
commit id,
which is a unique reference for what changes you made
when you made, who made the change.
Now declare everyone is so till here,
what still are changes are there in the laptop.
Then later on you do something,
call it pushing the chains, pulling the chain.
I'll discuss that later on.
But as of knowledge's, focus only in terms of
a developer laptop machine, WhatsApp.
Are you clear everyone now a developer is there,
is going to write the code.
He starts in the working area,
then he move into staging area,

then he will commit something.

Committing is like finalizing.

You're saying I am buying this product.

That's what committing means.

If you're not committing means that is not finalized.

Good, clear. Everyone is from online.

Any confusion regarding this part?

I hope everyone is

good, right?

Okay, perfect. Now whatever we just discussed,

there's one small thing you need to identify.

Now, there is one developer, so parallel,

there can be 10 other developers

who needs to do the same thing.

Agree or not. I just discussed in terms

of a single developer.

But literally there is the same thing is going

to happen across all other developers.

Uh, now who is committing?

Who is making, we need to keep a track of it.

I agree or not. How can we know developer called John is

working or developer, some other developer is working.

We need to have some track rate.

So what we do is whatever the GitHub account ID you got,

you will be registering in the gate

before you start doing the work.

We call it as setting the identity of the developer.

Before a developer starts the work

rate, he will set his identity.

What is identities?

GitHub id That's once he sets,

then he starts doing the work.

So we'll start beginning from that particular process.

Clear everyone what I'm trying to say right now. Okay?

Video: Git Config

Config Global, your name, which is nothing
but user id, then followed by email ID
of your GitHub account where I need to do it in the GI Bash.
Once I'm done with this, my work with GI Bash is done.
Are you clear everyone? Okay, now let me show you guys
how am I going to get the things done?
Yeah, okay. I'll just minimize it. This is my GI Bash.
Now I'm gonna set GI config
global user.me.
I need to accept the username of what might developer,
which is this one, right?
One second, I'll update the user.
I click and paste.
That's my developer user id.
Then followed by email, id,
alright, clear everyone.
So I have my GitHub account with username
and as well as email Id updated.

Video: Git Cloning Repository

I said we need to do the CLO to do cloning.
What you need means A URL.
I said every GitHub icon will have a URL.
Is this A URL? I have copied that. URL right now.
I'll go over here and you see there is one section called S
source code, source controller version control,
division control, et cetera.
You see you got two options.
Open folder, meaning if you already clone, you can open it

or you can go ahead and start cloning one osi.
I'll say clone paste the URL, what you copied.
Okay, that was a URL that I got for my login app.
Click the first option. Clone from URL, not from GitHub.
Okay. Clone from URL. Ah.
Now you said it cloning means what?
You're making a copy now it is asking
where do you want to save this copy?
I'm gonna keep it under my desktop.
Only clear what I'm trying. Ah.
Now once it's downloaded, it is called as local repository.
Are you clear everyone once it comes into your laptop?
It's called as what? Local repository.
Yes. Open this. Yes.
I trust the author.
Now you see this what? I'll close this welcome page.
No, it can you see here what we are seeing?
Read me file because that was the only file
that was there when we created the repository initially.
And you can see by default you'll always
get what branch means?
Main branch. Okay, clear everyone.
I hope you got the point.

Video: Git Branches

Start doing the work.
Now what we asked him to do,
develop one login page, isn't it?
Now he'll start writing the code.
Ah, he'll not directly start writing in the mail
where he needs to write the dev branch.
So now my developer will go ahead

and create his own branch called Dev branch.

From there he started the book. Are you clear everyone?

And Of course

any basically it's an I id You can use anything you want

to get same features every, okay, right?

Uh, now once I select it, I'm gonna start getting back

to my developer account.

You can see he's my developer. Uh, what was his work?

He's supposed to create a new branch rate.

Uh, see how he's creating a branch?

He'll go to the branch section.

You see Finder create a branch

are clear everyone.

So I said what? Create dev branch from me.

Are you clear? Everyone good? Now we have what?

A new branch. You see two.

This time there is one main branch and also dev branch.

Now my developer is going to work on the dev branch.

When I say work is going to the code, technically,

let's see how he does the work.

Now I'll go here

and you see if I click on this main,

I cannot see any dev branch, am I?

So there are two ways you can create the branch

from here and you can update.

Or you can update or you can create the branch from

GitHub and you can get it here.

Anything is fine. Okay, we created in GitHub.

Now I need to get that branch over here. Now pretty simple.

Now here I have zoomed out,

but you'll see one section called as Terminal.

Okay, click on terminal and say New Terminal.

All you need to do is here GI

Pull all.

So pull what? It basically does whatever is there on GitHub.

It'll update where in your laptop.

Because where I need to work, not on the main.

I need to work on what? Dev branch. So I'm saying pull.

Now let's see what happens when I run this instruction.

Okay. By default it's selected with still PowerShell.

So PowerShell does not support. Get out of the box.

So here you click on this box over here

like it dropdown is there, can you see there?

Open Git Bash.

You see what is the branch made?

Same command, GI pull, dash, dash.

Or you can do it in GI badge also.

But here itself integration is there. So I'm showing you.

That's why I closed it there.

Or if not in the GI badge, also you can execute.

So technically you're seeing GI Bash right now.

Isn't it the same color pattern right now

where we saw, see what it is saying.

New branch dev. Origin dev, simple.

This dev means local in your laptop.

This dev means remote GitHub.

I declare everyone there will be always two copies.

Don't get confused. One is local copy.

The other one is remote copy. Remote copy.

Generally we use the term called Origin

because from there we are getting the things directly.

Raise everyone. Now we are good.

I'll say close this off whenever you want to open.

Again, same thing. Go to Terminal New Terminal

and change to GI Bash and you can get the options.

Okay, now let's see. I will do this.

Uh, button once again. Click on me.

Can you see Dev now Earlier we didn't see it After doing

Pull, I'm seeing the dev branch.

Are you good? I'm gonna

say now.

Technically I'm not in the main branch.

I'm in the dev branch so far. Good.

Video: Git Workflow

Was the requirement given to us, uh,

create a login page, isn't it?

With this sort of code, uh, I already

prepared in such a way

that it should be practicable for everyone.

So what I did, I'll go back to my laptop

to browser simple.

All you need to do is do the Google search.

Like this will get the code copy paste

and you'll be able to understand it.

Just Google and say

create work page.

And that's it. Simple

in HT because that's what we said it index

html page should be created and all that requirement.

Select the first page.

How to create a login form from W three sports.

Just go there. You see how to create a login form

and you're seeing what all the code over here?

Okay, don't worry, just scroll down a bit here

and you'll find a button called try for yourself.

I'll show you.

Can you see here? Try it yourself.

It is about how to create a model login form.

Just click on try it yourself.

Did you get the point is I took this so that it is easy
for you to get this code anytime.
So the thing that you're seeing on the left is responsible
for creating this right side application page.
So obviously what I'm gonna do, control it,
copy all this piece of code,
I'm showing you how a developer is working.
Okay? Then go back to mine Vision studio code.
As of now, you see on the left side,
am I seeing any options or something like that?
Nothing. Right now you see slowly, I'll say right click
a new file and I'm gonna call it
s and index.
And you see as soon as I hit enter, you see
that like a timer button came
on the source control like a blue circle.
And you see what it is saying One,
what it is saying there, one pending change.
So it identified when the repository was there,
your state was different.
Now you're changing it.
I, I think. But what it is, tracking all those changes,
I created the file, right?
It was a change. I'll update the code.
That's a change. I'll save it.
Don't bother about what's code doing.
Just see what I'm doing as a developer. I wrote the code.
Ah, now I need to check it.
Whether it is really working as expected or not.
Where all this code is sitting. I mean like source.
Where is the physical source on the desktop?
Where did I clone it? Desktop, right?
I'll go to my file explorer right
now on desktop.

Did you get that login page?
I'm like, sorry, login repository 2 5 0 1.
I'm seeing the index. I'll open that. See desktop, login.
See the path, see users name desktop like that.
But is it working as expected? As is something missing?
What is missing? The image is missing, right?
Actually it is supposed to look like
this, but what is there?
Image is not there because images a separate file, right?
Code is separate. Image is separate.
So I'm gonna say right click this image.
You see there's an option called save images.
Uh, where should I save it now? Exactly.
In your project code, obviously in the repository,
all your project work will be there for sure in repository.
So I'll say go back to desktop login
and I'm gonna say save, save,
go back here and
refresh earlier.
How many pending changes were there?
One, which is your log code.
Uh, now you see what it is saying.
Why two basic, I understood
how this thing is working.
So everything you go ahead
and modify every change you do,
it'll be considered as a change.
All these changes where they will be there
not repository in the working area.
This is working area.
I said right there is working area
where you, you're doing the work.
This is technically right now working area.
It's not finalized yet. It is working area.

Do you like these changes? It's your call.

So when I say literally, okay,

it is showing me everything, right?

Yeah, right. Looks good.

Ah, maybe I'll say fine.

Yes, I like these particular changes.

I want to go ahead and update this code now

because it's working as expected.

Video: Git Stage - Commit

Now click on this button.

Source control. Now it says you see changes,

there's a column changes.

These are what changes, working

changes in your three places.

These are bottom place, which is called as working.

Now, let's assume that I like the index change.

Whatever the code I have written. You see there is a plus

button beside each file.

Can you observe? When I'm moving my mouse, it's showing up.

I hover the mouse on the plus

button and see what it is saying.

Me staging.

Basically it's saying St.

So when I click on this plus button, what happened?

Two different areas popped up.

One is changes, which is working on top of it.

Stage changes. That is like

card out of two.

One is there in the working.

The other one is there staging it,

but it is not finalized yet.

If you don't like later on,

what if you want to bring it back?

What are you saying? Like plus, minus,

minus means onstage.

Meaning you are literally saying, I don't want this changes.

Are you good guys? Everyone what I'm trying to say?

Is it clear what in the image that I discussed

working changes, staging changes.

Ah, if I don't like, I can say minus.

You see, it'll come

back clear everyone.

Ah, now what? But I'm pretty much sure I

want to stage the changes.

Now I staged it. Ah, now I want to finalize it.

Then we call it as what? Committing.

See, I'm committing right now.

Whenever you commit, there should be some description,

what work you did, agree or not.

If not, how The other developer,

when they read your history,

how would they know what happened?

So for every commit, it is always recommended to give

a proper description over that.

Hmm, what was the work I'm doing here right now?

Create login page and I'll say commit.

Once I click this button called commit.

I'm finalizing, meaning I I'm purchasing this product.

Now let's see what happens when I click on commit,

it's no longer even visible, right?

Because you only said I bought it.

Now every commit you make, you see

your history is getting updated.

Can you see the number one, 2D, D nine F 7 93

below the dev, that number you're seeing now

that is called Commit id.
And you see commit ID saying fires Restarted
committed this change on February 3rd.
At this point of time, uh, it's saying seven 20, don't worry
because this laptop is in some other time zone.
Are you clear right now?
One change I committed, uh,
but this change is still where it's in the working area.
If you like this, again, you need to go ahead and stage it.
Stage. I'll say
an avatar.
Uh, if I want, I can do multiple changes at a time
and I can commit, but I want to have, uh, what do you say,
more control in the future.
So I'm individually committing so that it'll also know like
how the changes are progressed.
I clear everyone now I'm gonna say what
I completed again, again, I want
to make the changes. The same
File. Same file, okay?
I'll show you where those options are.
If you want to change, you need to click on this file file.
If you click there again, you'll jump back
to your actual folder to make the changes.
Now you see I'm saying commit done.
Now you see there are no working changes
because you only said committed both.
And you see for every change I'm making, what's happening
again, a new commit Id got generated like that.
Moving forward, every change you do in the project
is going to be tracked.
Uh, you want to start doing the work again. Click on files.
Are you good everyone? Yes.
This is how literally developer works.

He goes, takes a requirement, starts writing the code.
He'll be using sort of tools like this.
Now, visuals is not tracking here it is tracking technically
using Visual Studio code we wrote, correct.
But as we install GitLab, it has automatic integration.
We don't need to do anything.
And what, what are the rules for the dev cloud dev
DevOps engineer in the future?
You need to write a script. Then how come you don't know
what is staging?
What is local? What is working? What is commit?
Where the versions are stored?
How to look at the id See technically speaking
as a developer, what they're writing a front end code.
Technically this is all html front end code.
In the future, you'll write scripts. What scripts?
Ya, L scripts, maybe Python scripts, bar scripts,
head sales scripts.
All that is code, but it does not look in this form.
Are you clear everyone? What I'm trying to say? Simple.
If not, I'll just show you a simple thing,
what you will write in the
future so that you'll get an idea.
Okay? This is what how I'm going to do the work
the same like 2, 5, 0, 1.
I had some other batch called 2 4, 3 5.
You see, I also have the same thing like main
branch and other branches.
You see there is a branch called, okay, it is Kubernetes.
In short it's folder.
You see this, this manuscripts you wrote? Mm-hmm.
When I open it, I want to know
how did I create something called like a
uh, HP?

I'm not sure how I created it. This was a file.
It is having a message. Uh, if I click on
this particular
HPAI
wrote two weeks ago, I can see all the history,
how it happened.
You can see there on January 17th,
I read the, I, sorry, I wrote this particular piece
of code as a DevOps.
I wrote along with your application code.
Your scripts will also be there in the
this get, uh, taken from Google or no, you need to write.
You can be writing. Not this.
Not only this, uh, people are going for later on Terraform.
I think for this batch, I did not complete the Terraform,
but yet maybe 4, 3, 3 branch.
I think I completed. Is it Terraform?
A branch. Now you've understood why
I'm creating different branches.
My DevOps engineers will keep their codes in their branches.
Within that. I'll click on it
is all Terraform code pretty lengthy?
You'll be writing at least two, 300 lines of code.
It looks complex and all of it, but eventually cap.
But all this, when I wrote how I wrote,
you see now I have entire history, isn't it?
What I'll do, I'll go back to my repository.
I'll click on something called commits, uh,
which is main branch.
Now our code is there in AWS everything.
What I wrote when I wrote everything is being
tracked as a developer.
What you did as a DevOps engineer,
literally do the same thing.

Only thing is you'll be writing different codes
because it's not your skillset, right?
Going and writing front end code is not your skillset.
Writing infrastructure script sets your skillset
and where do you write and manage all that code
and platforms like GI
and now clear how, how the things are going to be connected
with each other from online.
Does it make sense? Just a small glimpse
to help you understand how
a developer is working in the FU future.
You'll do exactly same. Literally same.
The only difference is requirement is different.
How I give the requirement
to create him a login application later on
as a DevOps engineer, you get a requirement for what?
Writing scripts deploy, creating pipelines,
all those inputs you'll get.
Now you'll write code here.
Then you need to know what's the repository.
You need to know what's the branch.
You need to know what is commit, what are uh, what is it?
Pull requests match everything a developer does, literally
as a DevOps engineer do.
But only the difference is context is different.
Am I clear radius?
Okay, so this is how a developer goes
and starts working with the text.
Now you got the clarity, everyone, ah,
still whatever the code we wrote it is where in
the lab, if I go back to my
developer account, which is here
and say Login 2, 5 0 1.
I'm in the main branch, I'm gonna switch to dev branch.

But am I seeing any changes here?

What was the commit one, commit when,

but now I did additionally, two more.

Commit. Commits, right? But where those commits are in my

workstation are basically in my laptop.

Agree or not. So those changes are there in

the local repository.

Now, that's why you see when I hover this mouse,

what it is saying, sync changes too.

Creating index pages, one change, updating the images.

One change. So you have two changes.

Now it is saying sync these two changes back where

to your remote reference entry like this.

You'll do the work in your local laptop

and then you'll be updating the work back to GitHub.