



DevOps: Git & GitHub Crash Course

What is DevOps?

 **DevOps = Developers + Operations**

DevOps is like a superhero team where:

- **Developers** build the software (the *brains* )
- **Operations** make sure it works perfectly for everyone, everywhere (the *muscle* )

When they work **together**, magic happens! ✨

Why Do We Need DevOps?

Imagine you made a video game, but it works only on your computer.





Now your friends want to play too — on *their* devices. You need DevOps to:

- ✓ Share updates fast
- ✓ Fix bugs quickly
- ✓ Make sure it works for *everyone*
- ✓ Keep the game running 24/7

What Tools Do DevOps Use?

TOOL	WHAT IT DOES
GIT & GITHUB	Tracks and shares code changes
DOCKER	Packs the app like a lunchbox so it runs anywhere
CI/CD	Robots that test and deliver your app automatically
CLOUD (LIKE AWS)	Computers in the sky that run your app for people everywhere

DevOps is Like a Science Lab

1. **Plan**  – Decide what you want to build
2. **Build**  – Create it with code
3. **Test**  – Check for problems
4. **Release**  – Share it with the world

5. **Monitor** 🗓️ – Make sure it keeps working

Then repeat! That's called the **DevOps Cycle** 🔄

💡 Why It's Cool

- You can fix things fast
- You learn to work with others
- You get to **build real stuff** that helps people
- It's like **engineering + teamwork + coding + fun**

🚩 Final Thought

DevOps is not just tools. It's a **team spirit** to build better things, faster and together!

Ready to become a DevOps superhero? 🦹♀️🦹♂️

🔖 Page 1: What is Git & GitHub?

✂ **Git** is a tool that helps you **track changes** in your code or files. Like a **magic notebook** that remembers everything you did!

🌐 **GitHub** is a **website** where you can keep your Git work safe online, share it, or work with friends.

Why it's cool:

If something breaks, you can go back. If you want to try something new, you can test it without messing up the original!

🔖 Page 2: Setting Up Git on Your Computer

1. Install Git:

Go to git-scm.com and download it for your computer.

2. Set your name and email:

bash

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

🔖 Git needs to know who is doing the work so it can keep track of your changes!

📁 Page 3: Start Your First Git Project

Imagine you're writing a story.

bash

mkdir my-story

cd my-story

git init

🔊 Boom! Git is now watching your folder.

🔖 Page 4: Making Your First Commit

1. Create a file:

bash

echo "Once upon a time..." > story.txt

2. Tell Git to track the file:

bash

git add story.txt

3. Save your work with a message:

bash

git commit -m "Started my story"

🔖 Like saving a game with a note: "Saved before dragon fight."

🔍 Page 5: Checking What's Happening

- See what's changed:

bash

git status

- See your saved versions:

bash

git log

👤 Git shows you the who, what, and when of your changes.

🌐 Page 6: What is GitHub and Why Use It?

Git is local (on your computer).
GitHub is remote (on the internet).

You push your work to GitHub like this:

1. Create a free account at github.com
2. Make a new repository
3. Link your Git folder to GitHub:

bash

git remote add origin https://github.com/yourusername/my-story.git

git push -u origin main

☁ Now your story is in the cloud! You can share it, and work on it from anywhere.

Page 7: Downloading Projects from GitHub

If your friend has a cool story, you can copy it like this:

bash

git clone https://github.com/friend/story.git

📖 It's like packing their whole notebook into your backpack!

Page 8: Trying New Ideas with Branches

Let's say you want to try a plot twist.

bash

git branch twist-ending

git checkout twist-ending

Now you're writing in a new notebook! When you're done:

bash

git checkout main

git merge twist-ending

📖 Branches are awesome for experimenting without messing up your main work.


Page 9: Collaborating with Others

You can work with friends using:

- **Pull Requests** (suggest changes)
- **Forks** (make a copy of someone's repo)
- **Issues** (report bugs or ideas)

Typical teamwork flow:

1. Pull updates: git pull
2. Make changes
3. Push: git push

 Teamwork makes the code work!

Page 10: Summary & DevOps Connection

Git Command	What it Does
<code>git init</code>	Start tracking changes
<code>git add</code>	Select files to save
<code>git commit -m</code>	Save changes with a message
<code>git status</code>	Check what's changed
<code>git push</code>	Upload to GitHub
<code>git pull</code>	Download updates
<code>git branch</code>	Make a new version line
<code>git merge</code>	Combine version lines

DevOps Tip:

Git + GitHub is how real DevOps engineers manage code safely, fix bugs fast, and work in teams. You've just learned the foundation of professional software development!

Bonus Activities

- Create a personal website repo on GitHub
- Practice version control by making silly story edits
- Make a game mod and track versions
- Explore GitHub Projects for team planning