GitHub Tutorial for Beginners & Non-Techies

GitHub is a platform for hosting and collaborating on code. You can still use GitHub for version control, documentation, and teamwork even if you're not a developer. This tutorial will guide you through the basics.

Github, Bitbucket, Azure DevOps(**Repos**)

---

1. What is GitHub?

GitHub is an online platform that helps you store, manage, and collaborate on projects using Git, **a version control system**.
Think of it as **Google Drive for code**, where multiple people can work together without overwriting each other's work.

---

2. Setting Up GitHub

Step 1: Create a GitHub Account

1. Go to GitHub.com and **Sign Up**.
2. Choose a username, enter your email, and set a password.
3. Verify your email and log in.

---

3. Basic GitHub Features

Repositories (Repos)

A repository is like a folder where your project files are stored.

Branches [Develop, Release based]

Think of branches as different versions of your project. The main branch is usually called `main` or `master`.

Commits [Save the code in repo]

A commit is like saving a file but with a history of changes.

Pull Requests (PR) [Needs approval], PR comments

A pull request is when you ask to merge changes from one branch into another.
Link to the changes

Issues

Issues are like to-do lists or bug trackers where you can discuss problems and solutions.

---

4. Creating Your First Repository

Option 1: Using GitHub Website

1. Click on `+` (top-right corner) → `New repository`.
2. Give it a name (e.g., `MyFirstRepo`).
3. Choose **Public** (anyone can see) or **Private** (only you can see).
4. Click **Create repository**.

---

5. Uploading Files (Without Coding)

1. Open your repository.
2. Click **Add file** → **Upload files**.
3. Drag & drop or select a file.
4. Click **Commit changes** (like saving).

---

6. Editing Files in GitHub

1. Open a file in your repository.
2. Click the **pencil (edit) icon**.
3. Make changes.
4. Scroll down and click **Commit changes**.

---

7. Cloning a Repository (Copying to Your Computer)

If you want to edit files offline:

1. Install **Git** from git-scm.com.
2. Open **Terminal (Mac/Linux) or Git Bash (Windows)**.

Type:
```
git clone https://github.com/YourUsername/MyFirstRepo.git
```

3.
4. Press Enter.

---

8. Making Changes & Pushing Them to GitHub

If you edited files on your computer:

1. Open **Git Bash**.

Navigate to your folder:

```
cd MyFirstRepo
```

2.

Add changes:
```
git add .
```

3.

Commit changes:
t
```
git commit -m "Updated the file"
```

4.

Upload to GitHub:

```
git push origin main
```

5.

---

9. Forking & Pull Requests

Forking (Copying Someone's Repo)

1. Open a public repository.
2. Click **Fork** (top-right).
   This creates a copy in your GitHub account.

Pull Request (Suggesting Changes)

1. Go to your forked repository.
2. Click **New Pull Request**.
3. Add details and submit.

---

10. GitHub Desktop (No Coding Needed)

If you don't like using commands, use GitHub Desktop:

1. Install **GitHub Desktop**.
2. Log in to GitHub.
3. Clone, edit and commit with clicks.

---

11. Collaborating on GitHub

1. Click **Settings** → **Manage Access**. **[Repo Link]**
2. Invite team members.
3. They can push changes and contribute.

---

12. Deleting a Repository [Not Recommended]

1. Go to your repository.
2. Click **Settings** → Scroll to **Delete this repository**.
3. Confirm deletion.

---

13. Best Practices for Beginners

✅ Use clear names for files & reports.
✅ Write commit messages like "Fixed typo in README".
✅ Use issues to track tasks.
✅ Collaborate with pull requests instead of editing `main` directly.

Final Thoughts

GitHub is not just for coding—it's a powerful tool for **teamwork, documentation, and version control**. Even without coding skills, you can **upload files, collaborate, and track changes** easily.

Deep Dive into GitHub for Beginners & Non-Techies 🚀

If you're new to GitHub or not from a technical background, don't worry! **GitHub is more than just for coding.** You can use it to **store documents, collaborate on projects, track changes, and even manage teams.** Let's explore GitHub in more detail.

---

1. Why Use GitHub?

Even if you're not a programmer, you can use GitHub for:
✅ **Document version control** – Keep track of different versions of your files.
✅ **Team collaboration** – Work with others without overwriting their changes.
✅ **Project management** – Assign tasks and track progress using Issues and Milestones.
✅ **Portfolio building** – Store your work, blogs, or even non-code projects.

---

2. Understanding GitHub Terminology

Before using GitHub, it's important to understand these basic terms:

Term	Meaning
Repository (Repo)	A folder that contains your project files (like Google Drive).
Branch	A separate version of your project (like different document drafts).
Commit	A saved change in the project (like saving a new version of a file).
Pull Request (PR)	A request to merge one branch into another (like suggesting an edit).
Fork	A copy of someone else's repo in your account (like downloading a template).
Issue	A way to track tasks, suggestions, or bugs (like a to-do list).
Clone	Downloading a repo to your local computer for offline work.
Merge	Combining changes from different branches into one.

---

3. Getting Started with GitHub

Step 1: Create an Account

1. Visit GitHub.com and click **Sign Up**.
2. Choose a **Username**, **Email**, and **Password**.
3. Complete verification and click **Create Account**.

Step 2: Create Your First Repository

1. Click on the `+` sign (top-right) → **New Repository**.
2. Give it a name (e.g., `MyProject`).
3. Check **"Add a README file"** (this is like a project introduction).
4. Click **Create Repository**.

---

4. Uploading Files to GitHub (No Coding Required)

If you have a Word, Excel, or PDF file you want to store:

1. Open your repository.
2. Click **Add file** → **Upload files**.
3. Drag & drop your files or select them from your computer.
4. Click **Commit changes**.

📌 **Use Case**: You can use this to save documents, meeting notes, or project reports.

---

5. Editing Files on GitHub

1. Open a file inside your repository.
2. Click the **pencil icon** to edit.
3. Make changes and add a message (e.g., "Updated document title").
4. Click **Commit changes**.

📌 **Use Case**: If you're managing a content repository, you can edit articles, add meeting notes, or update documentation.

---

6. GitHub for Collaboration

If you're working with a team, GitHub makes it easy to collaborate.

Adding Team Members

1. Go to your repository → Click **Settings**.
2. Select **Manage Access**.
3. Click **Invite a Collaborator** and enter their GitHub username.

Now, they can **edit files, upload content, and contribute** to the project.

📌 **Use Case**: This is great for group projects, content teams, or research documentation.

---

7. Using Issues for Task Management

GitHub **Issues** help you track tasks, bugs, and feature requests.

Creating an Issue

1. Open your repo → Click **Issues**.
2. Click **New Issue**.
3. Add a **Title** (e.g., "Update Marketing Report").
4. Describe the task.
5. Assign it to a team member.
6. Click **Submit new issue**.

📌 **Use Case**: A **project manager** can track progress without writing any code.

---

8. Understanding Branches & Pull Requests

A **branch** is like a new version of your project where you can make changes without affecting the main version.

How to Create a Branch

1. Open your repo.
2. Click **Branch: main** → Type a new branch name (e.g., `feature-update`).
3. Click **Create Branch**.

Making a Pull Request (PR)

Once you've made changes in a branch:

1. Click **Pull Requests** → **New Pull Request**.
2. Compare your branch with `main`.
3. Click **Create Pull Request**.

4. Add a description and click **Submit**.

📌 **Use Case**: If you're writing a blog or a report, you can create drafts in a separate branch before publishing.

---

9. Forking a Repository (Copying Someone's Work)

If you find a public repository you want to use:

1. Click **Fork** (top-right corner).
2. This creates a copy in your GitHub account.
3. Edit it and make your own changes.

📌 **Use Case**: Useful for **downloading templates**, **collaborating on open-source projects**, or **customizing an existing project**.

---

10. Using GitHub Desktop (No Commands Needed)

If you don't like typing commands, use **GitHub Desktop**:

1. Download **GitHub Desktop** (GitHub Desktop).
2. Log in with your GitHub account.
3. Clone a repository (download it to your PC).
4. Edit files on your computer.
5. Click **Commit & Push** to upload changes.

📌 **Use Case**: This is great for **graphic designers, content creators, and business users** who prefer a visual interface.

---

11. Deleting a Repository

If you no longer need a repository:

1. Open the repository.
2. Click **Settings** (bottom left).
3. Scroll down and click **Delete this repository**.
4. Type the repo name and confirm.

---

12. GitHub for Non-Tech Professionals

You don’t need to write code to use GitHub! Here are some ways **non-techies** can use it:

Role	How GitHub Helps
Project Manager	Track tasks using Issues & Milestones.
Writer/Blogger	Store drafts, collaborate, and version control.
Designer	Share and track design files.
Data Analyst	Store CSV files and collaborate on reports.
HR & Recruiters	Store resumes, documents, and track candidates.

---

13. Best Practices for Beginners

✅ **Use meaningful commit messages** (e.g., "Updated Q1 Report" instead of "Changed file").
✅ **Use Issues to track tasks** instead of managing them in emails.
✅ **Use branches for major changes** instead of editing the main file.
✅ **Regularly push updates** to keep your repo up to date.
✅ **Keep repositories organized** with proper naming and descriptions.

---

14. Summary

✅ **GitHub is not just for coders!**
✅ **You can use it to manage documents, collaborate, and track work.**
✅ **No coding skills required—just use the web interface or GitHub Desktop.**
✅ **It’s great for teams, projects, and content management.**

---

15. Next Steps

📌 **Want hands-on practice?** Try creating a **GitHub repo for your notes or portfolio.**
📌 **Need help with a specific GitHub feature?** Ask me!

GitHub is like a supercharged Google Drive with history tracking—make the most of it! 🚀

GitHub & CI/CD Concepts for Beginners & Non-Techies 🚀

If you're new to **GitHub and CI/CD** (Continuous Integration & Continuous Deployment), don't worry! **You don't need to be a developer to understand how GitHub connects with CI/CD.**

CI/CD helps automate the process of testing, building, and deploying projects. Think of it as an **"Auto-Save + Auto-Publish"** system for your projects. Let's break it down in a simple way.

---

1. What is CI/CD? (In Simple Terms)

📌 Continuous Integration (CI)

- Every time a team member makes a change in a project, the system **automatically tests** whether everything is working fine.
- Example: If you update a document, CI ensures there are **no formatting errors** before saving.

📌 Continuous Deployment (CD)

- Once the changes are tested, the system **automatically updates** the live version.
- Example: When you upload a blog draft, CD **automatically publishes** it on the website.

✅ **CI/CD = Automated Workflow (Build → Test → Deploy Automatically!)**

---

2. How GitHub Connects to CI/CD

GitHub is used to **store and manage files**. But you can also **automate tasks using CI/CD pipelines**.

How It Works

1️⃣ **You upload or edit files** in a GitHub repository.
2️⃣ **A CI/CD tool checks for errors** (e.g., formatting, security issues).
3️⃣ **If everything is correct,** the changes are automatically applied to the live version (website, app, document, etc.).

✅ **This means NO MANUAL UPDATES are needed—GitHub & CI/CD do the work for you!**

---

3. Popular CI/CD Tools Used with GitHub

Here are some **popular CI/CD tools** that work with GitHub:

Tool	Use Case
GitHub Actions	Automates tasks inside GitHub itself.
Jenkins	Open-source automation tool for testing & deployment.
GitLab CI/CD	Built-in CI/CD for GitLab repositories.
CircleCI	Cloud-based CI/CD platform.
Travis CI	Simple automation for testing and deployment.

👉 **For beginners,** GitHub Actions is the easiest way to start!

---

4. Setting Up CI/CD with GitHub (No Coding Required)

Let's set up a **basic GitHub CI/CD workflow using GitHub Actions**.

Step 1: Create a GitHub Repository

1. Go to GitHub.com.
2. Click **New Repository** → Give it a name (e.g., `MyCICDProject`).
3. Check **Add a README** → Click **Create Repository**.

Step 2: Add a GitHub Actions Workflow

1. Go to your repository.
2. Click **Actions** (top menu).
3. Click **New Workflow** → **Set up a workflow yourself**.
4. Add the following YAML file:

```
name: Auto Check

on: [push]

jobs:
  check:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
      - name: Run a test
        run: echo "CI/CD is working!"
```

5. Click **Start Commit** → **Commit changes**.

🎯 **What happens?**

- Every time you update files, GitHub **automatically runs this script**.
- It prints **"CI/CD is working!"** to confirm automation is running.

✅ **You just created your first CI/CD automation!**

---

5. Real-World CI/CD Examples (For Non-Techies)

Here are some **practical, non-technical** use cases where GitHub + CI/CD can help:

Scenario	How CI/CD Helps
Updating a Blog	Automatically publishes the blog after writing in GitHub.
HR Resume Screening	Automatically checks resumes for specific keywords.
Project Documentation	Ensures no formatting errors before saving documents.
Graphic Design Review	Checks image file sizes before uploading to a website.
Website Management	Automatically deploys website updates after edits.

✅ **CI/CD is not just for developers—it helps in any workflow that needs automation!**

---

6. Key Benefits of Using GitHub & CI/CD

🔹 **Saves Time** – No need to manually check and update files.
🔹 **Reduces Errors** – Ensures changes are tested before being applied.
🔹 **Improves Collaboration** – Multiple people can work on a project without conflicts.
🔹 **Increases Efficiency** – Automates repetitive tasks like testing and publishing.

---

7. Next Steps

Now that you understand the basics, here's what you can do next:

✅ **Try GitHub Actions** – Set up a simple workflow in a test repository.
✅ **Explore CI/CD Tools** – Experiment with Jenkins, CircleCI, or Travis CI.
✅ **Automate Small Tasks** – Start with document formatting, then move to larger workflows.

---

8. Summary

📌 **GitHub is where you store projects.**
📌 **CI/CD automates testing and deployment.**
📌 **GitHub Actions is the easiest way to start.**
📌 **CI/CD isn't just for coders—anyone can use it for automation!**

Azure DevOps CI/CD for Beginners & Non-Techies 🚀

Azure DevOps is a **cloud-based platform** by Microsoft that helps automate software development and deployment. If you're new to **CI/CD (Continuous Integration & Continuous Deployment)**, don't worry! I'll explain how it works in **simple terms, even for non-techies**.

---

1. What is Azure DevOps?

Azure DevOps is a **one-stop solution** for managing projects, storing code, automating builds, and deploying applications. It includes:

✅ **Azure Repos** – A Git-based code storage (like GitHub).
✅ **Azure Pipelines** – Automates CI/CD (Build → Test → Deploy).
✅ **Azure Boards** – Task & project management (like Jira/Trello).
✅ **Azure Artifacts** – Stores dependencies & packages.
✅ **Azure Test Plans** – Manages test cases for quality assurance.

💡 **You don't need to be a developer** to use Azure DevOps! It helps in **project management, testing, automation, and documentation**.

---

2. What is CI/CD in Azure DevOps?

CI/CD = Continuous Integration (CI) + Continuous Deployment (CD)

Concept	Meaning (Simple Terms)
CI (Continuous Integration)	Automatically checks if new changes are correct.
CD (Continuous Deployment)	Automatically updates the live version after testing.

🔷 **Example:**

- **Before CI/CD** – A developer updates a website, and then manually uploads it.
- **With CI/CD** – The system automatically checks for errors and **deploys** the changes.

---

3. How Azure DevOps CI/CD Works?

1️⃣ **Developers or teams store project files** in **Azure Repos**.
2️⃣ **Azure Pipelines runs tests** to check for issues.
3️⃣ **If everything is correct**, the app is **automatically deployed**.
4️⃣ **The updated version goes live** with no manual work.

💡 **It's like an automatic content publishing system for software!**

---

4. Setting Up Azure DevOps CI/CD (Step-by-Step)

Let's create a **basic CI/CD pipeline in Azure DevOps**.

Step 1: Create an Azure DevOps Account

1. Go to Azure DevOps and sign in with a Microsoft account.
2. Click **Create New Organization** → Set up a project.
3. Choose **Public or Private** visibility.

Step 2: Create a Repository (Azure Repos)

1. Inside your project, go to **Repos**.
2. Click **Initialize Repository** → Choose Git.
3. Upload files or connect an existing GitHub repo.

Step 3: Create a CI/CD Pipeline (Azure Pipelines)

1. Go to **Pipelines** → Click **New Pipeline**.
2. Select **GitHub/Azure Repos** as the source.
3. Choose **Starter Pipeline** → Edit YAML if needed.

```
trigger:

- main

pool:

  vmImage: 'ubuntu-latest'

steps:

- script: echo "CI/CD is Running Successfully!"
```

```
  displayName: 'Print Message'
```

4. Click **Save & Run** → This runs a simple automation!

📌 **What Happens?**

- Every time you update your files, this pipeline runs and prints **"CI/CD is Running Successfully!"**
- You can add **tests, build steps, or deploy** to servers later!

---

5. Deploying to a Web App (Basic Example)

Let's say we want to **deploy a website** to Azure.

Step 1: Create an Azure Web App

1. Log in to Azure Portal.
2. Click **App Services** → **Create Web App**.
3. Choose **Resource Group, App Name & Runtime** (e.g., Node.js, Python).
4. Click **Review & Create**.

Step 2: Modify the Azure Pipeline

1. Open **Azure Pipelines**.
2. Modify your YAML file to deploy the web app:

```
trigger:

- main

pool:

  vmImage: 'ubuntu-latest'

steps:

- task: UseDotNet@2

  inputs:

    packageType: 'sdk'
```

```
    version: '6.x'
    installationPath: $(Agent.ToolsDirectory)/dotnet

- task: AzureWebApp@1
  inputs:
    azureSubscription: 'YourAzureSubscription'
    appName: 'YourAppName'
    package: '$(System.DefaultWorkingDirectory)/myproject'
```

3. Click **Save & Run** → Your web app is deployed automatically!

📌 **Now, every time you update the code, Azure DevOps will test & deploy it automatically!**

---

6. Non-Tech Use Cases for Azure DevOps CI/CD

Even if you're not a developer, you can use **CI/CD for automation!**

Scenario	How CI/CD Helps
Website Updates	Automatically deploys website content.
HR Resume Screening	Automatically sorts resumes using AI filters.
Document Formatting	Checks and fixes formatting errors.
Automated Reports	Generates & publishes weekly reports.

Graphic Design Approval	Auto-checks file formats & uploads designs.

✅ **CI/CD is useful beyond just software development!**

---

7. Key Benefits of Azure DevOps CI/CD

- 🔹 **Saves Time** – No manual testing or deployment.
- 🔹 **Reduces Errors** – Ensures no mistakes before deployment.
- 🔹 **Improves Collaboration** – Teams can work together seamlessly.
- 🔹 **Increases Efficiency** – Automates repetitive tasks.
- 🔹 **Scalable** – Works for small teams & enterprise-level projects.

---

8. Summary & Next Steps

- ✅ **Azure DevOps is a powerful automation tool.**
- ✅ **CI/CD helps teams work faster & smarter.**
- ✅ **Azure Pipelines automate testing & deployment.**
- ✅ **No manual intervention is needed—just push your updates!**
