Expert Cloud Consulting

Enhance Optimise & Scale

ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting”

SOP | Version Control to Deployment: GitHub for DevOps Teams

6 Jun 2025

—

Contributed by: Akshata Ujawane
Approved by : Akshay (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

Version Control to Deployment: GitHub for DevOps Teams

1.0: Table of Contents

2.0 General Information:

2.1: Document Purpose

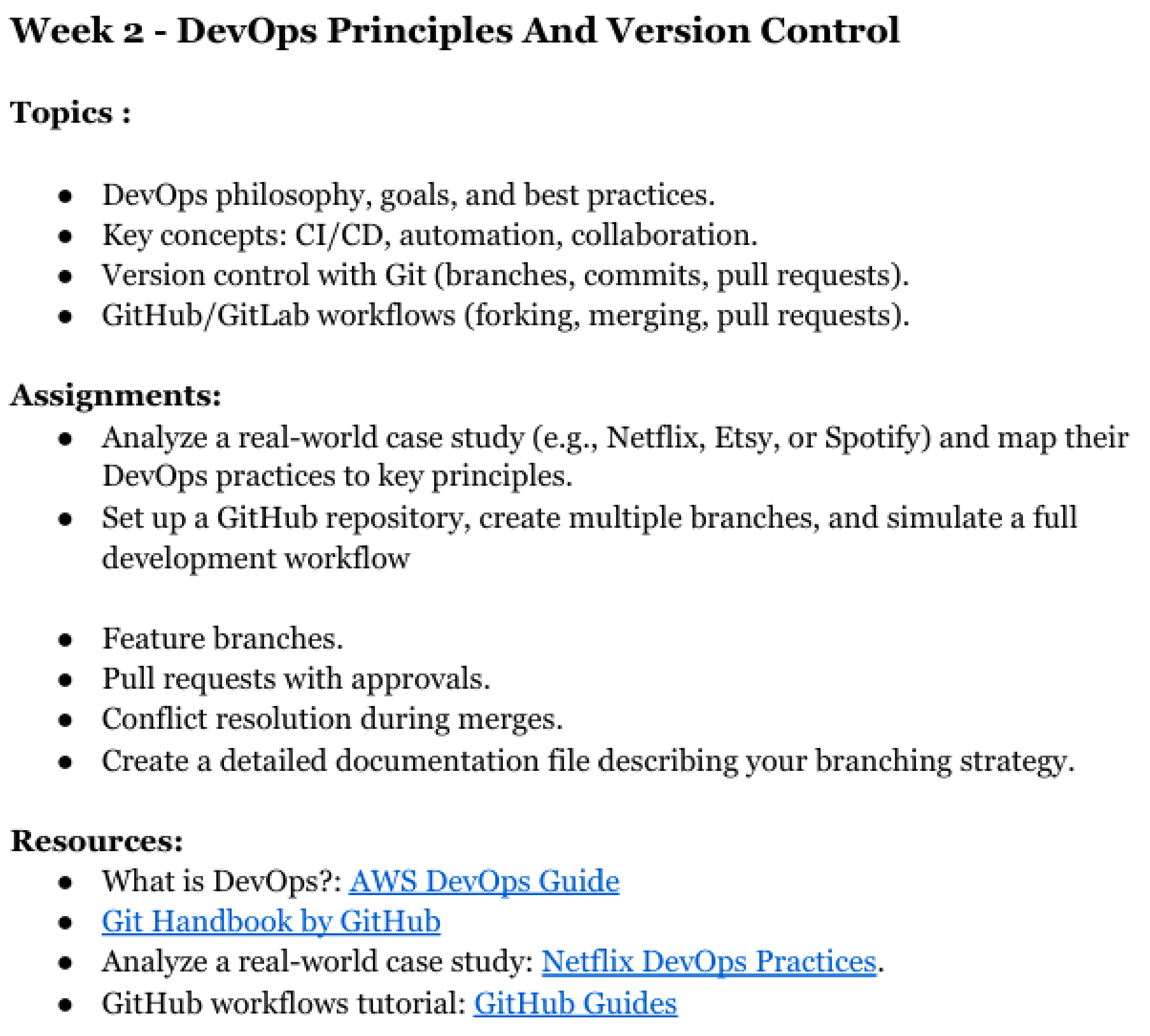
This document explores how GitHub became a cornerstone of modern DevOps workflows by enabling version control, seamless team collaboration, and automation. Through real-world workflows, branching strategies, pull requests, and conflict resolution methods, this document outlines the vital role GitHub plays in accelerating software delivery, maintaining quality, and enhancing collaboration across teams.

2.2 Document References

The following artifacts are referenced within this document. Please refer to the original documents for additional information.

Date	**Document**	**Filename / Url**
03.06.2025	Merge conflict	Git Branching Strategies for DevOps Engineers || 15+ Years Exp Tech Architect
04.06.2025	Branching Strategies	Introduction to Git - Branching and Merging

DocumentOverview**:**
GitHub is a web-based platform that facilitates version control and collaboration using Git. It enables developers to store and manage code repositories, track changes, handle branching, and collaborate through pull requests. GitHub also offers additional features such as issue tracking, project management tools, and integration with CI/CD pipelines to automate workflows. This makes it an essential tool for both open-source projects and professional software development.



Week 2 - DevOps Principles And Version Control

Topics :

- DevOps philosophy, goals, and best practices.
- Key concepts: CI/CD, automation, collaboration.
- Version control with Git (branches, commits, pull requests).
- GitHub/GitLab workflows (forking, merging, pull requests).

Assignments:

- Analyze a real-world case study (e.g., Netflix, Etsy, or Spotify) and map their DevOps practices to key principles.
- Set up a GitHub repository, create multiple branches, and simulate a full development workflow

- Feature branches.
- Pull requests with approvals.
- Conflict resolution during merges.
- Create a detailed documentation file describing your branching strategy.

Resources:

- What is DevOps?: AWS DevOps Guide
- Git Handbook by GitHub
- Analyze a real-world case study: Netflix DevOps Practices.
- GitHub workflows tutorial: GitHub Guides

1.0 Introduction

GitHub is a web-based platform built around Git, a distributed version control system widely used for source code management. It provides developers with a centralized location to store, track, and collaborate on code. GitHub simplifies tasks such as branching, merging, and managing pull requests, making it easier for individuals and teams to work together efficiently.

Beyond version control, GitHub offers a suite of tools including issue tracking, project boards, and integrated CI/CD features through GitHub Actions. Its intuitive interface, strong community support, and seamless integration with development tools have made it the preferred platform for both open-source contributors and professional software teams.

This case study explores how GitHub enhances code collaboration, improves transparency, and supports structured development workflows, making it an indispensable tool in modern software engineering.

Git in a DevOps Context

Netflix-style DevOps emphasizes automation, collaboration, and speed. Git supports this by enabling:

- **Version control** for all code changes
- **Collaboration** across globally distributed teams
- **Traceability** and rollback during deployments
- **CI/CD integrations** with tools like Jenkins, Spinnaker, or GitHub Actions

1.0: Practical Git Commands for DevOps Workflow

Set global username and email for Git (Locally).

```
git config --global user.name "<aksata Ujawane  >"
git config --global user.email "<akshataujawane19@gmail.com>"
```

Initialise an empty Git Repository

```
git init
```

Clone an existing Git Repository

```
git clone <repository URL>
```

Add file/stage to git

```
git add <filename>
```

Add all the files to git

```
git add .
```

Commit all the staged files to git

```
git commit -m "<your commit message>"
```

Restore the file from being modified to Tracked

```
git restore <filename>
git checkout <filename>
```

Show the status of your Git respository

```
git status
```

Show the branches of your git repository

```
git branch
```

Checkout to a new branch

```
git checkout -b <branch name>
```

Checkout to an existing branch

```
git checkout <branch name>
```

Remove a branch from Git

```
git branch -d <branch name>
```

Show remote origin URL

```
git remote -v
```

Add remote origin URL

```
git remote add origin <your remote git URL>
```

Remove remote origin URL

```
git remote remove origin
```

Fetch all the remote branches

```
git fetch
```

Push your local changes to remote branch
git push origin <branch name>

Pull your remote changes to local branch
git pull origin <branch name>

Check you git commits and logs
git log

Step 1: Sign In to GitHub

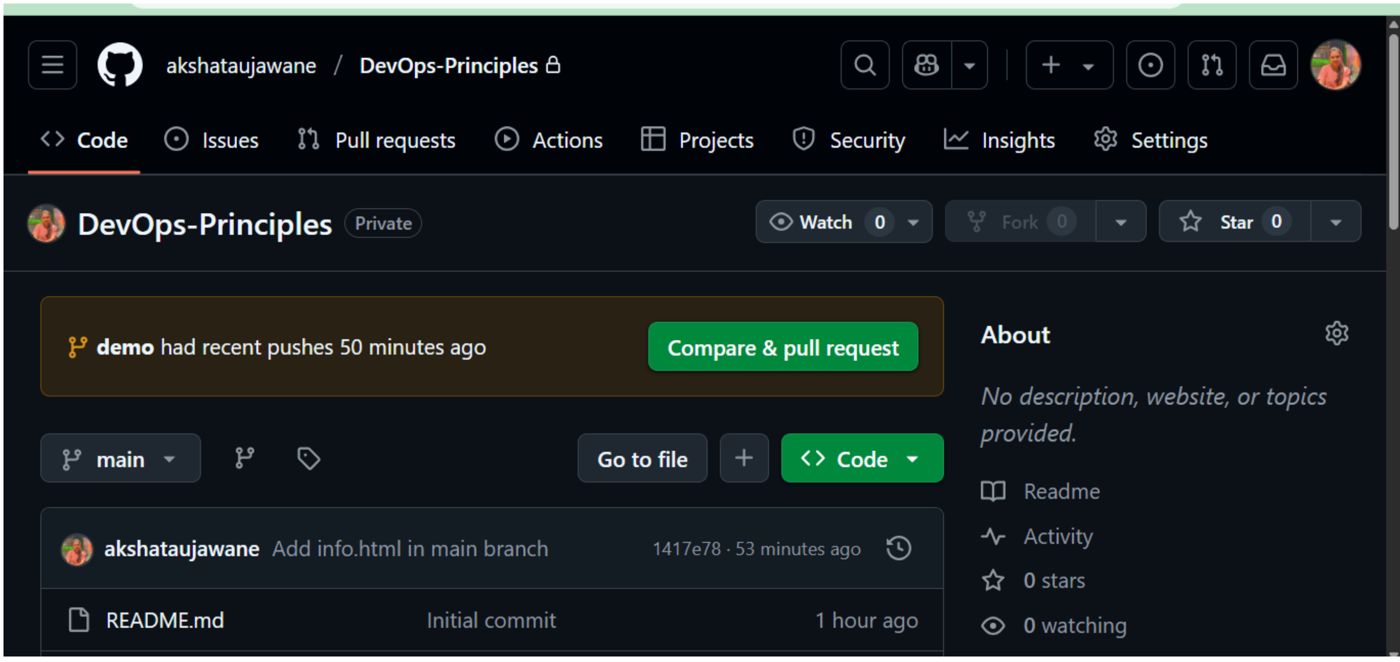
1. Go to [GitHub](https://github.com).
2. Sign in to your GitHub account using your credentials.

Step 2: Create a New Repository

1. Once logged in, click on the **+** icon on the top-right corner of the GitHub homepage.
2. Select **New repository** from the dropdown menu.

Step 3: Create Repository

Click the **Create repository** button at the bottom to finalize the creation of the repository.



akshataujawane Add info.html in main branch 1417e78 · 53 minutes ago
README.md Initial commit 1 hour ago

Step 4: Clone the Repository Locally

Once your repository is created, you can clone it to your local machine:

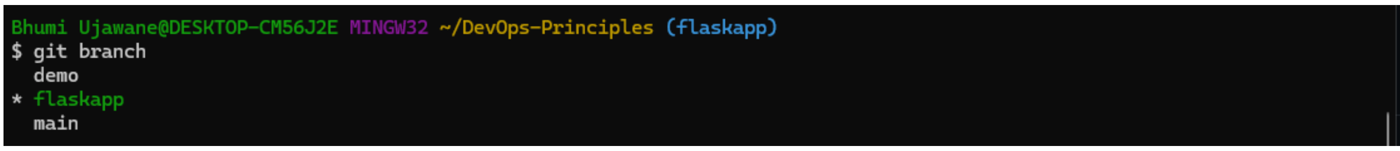
1. On your repository page, click the **Code** button.
2. Copy the URL under **HTTPS** (e.g., https://github.com/your-username/devops-project.git).
3. Open a terminal on your local machine and run the following command:

   git clone https://github.com/akshataujawane/DevOps-Principles.git

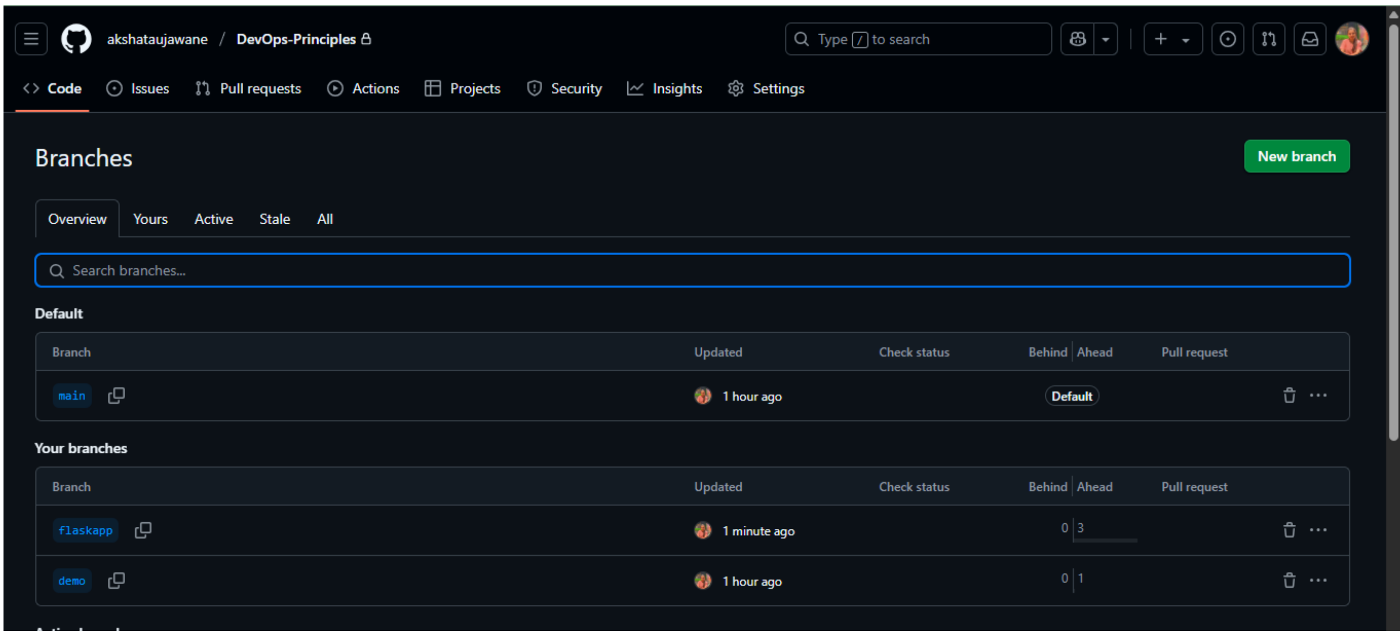
This will clone the repository to your local machine, and you'll be ready to start working on your DevOps project!

Step 5: Create New Branches (Using Git CLI)

git checkout -b <demo>
git branch



```
Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ git branch
  demo
* flaskapp
  main
```



Branches

Default

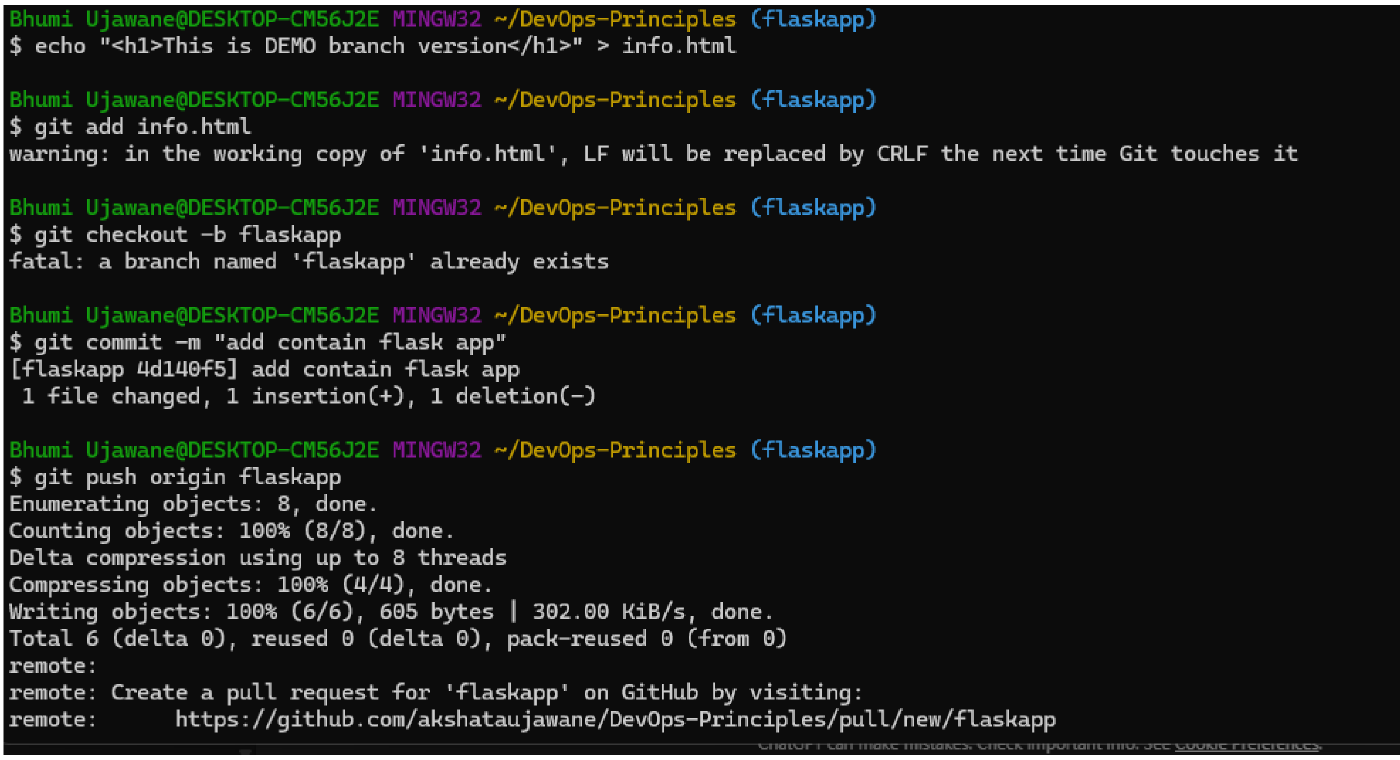
Branch	Updated	Check status	Behind	Ahead	Pull request
main	1 hour ago		Default		

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
flaskapp	1 minute ago		0	3	
demo	1 hour ago		0	1	

5: Stage, Commit & Push Changes

git add info.html
git commit -m "Added login page feature"
git push origin flskapp



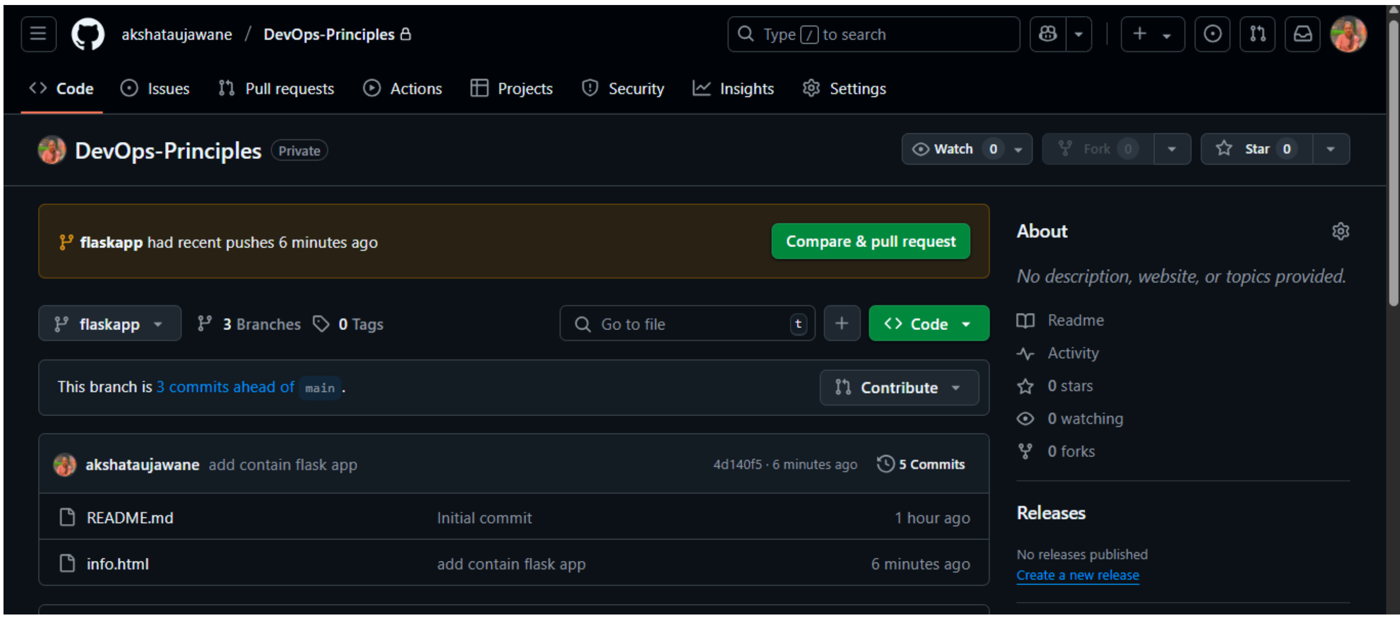
```
Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ echo "<h1>This is DEMO branch version</h1>" > info.html

Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ git add info.html
warning: in the working copy of 'info.html', LF will be replaced by CRLF the next time Git touches it

Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ git checkout -b flaskapp
fatal: a branch named 'flaskapp' already exists

Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ git commit -m "add contain flask app"
[flaskapp 4d140f5] add contain flask app
 1 file changed, 1 insertion(+), 1 deletion(-)

Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (flaskapp)
$ git push origin flaskapp
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 605 bytes | 302.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'flaskapp' on GitHub by visiting:
remote:      https://github.com/akshataujawane/DevOps-Principles/pull/new/flaskapp
```



flaskapp had recent pushes 6 minutes ago

This branch is 3 commits ahead of main.

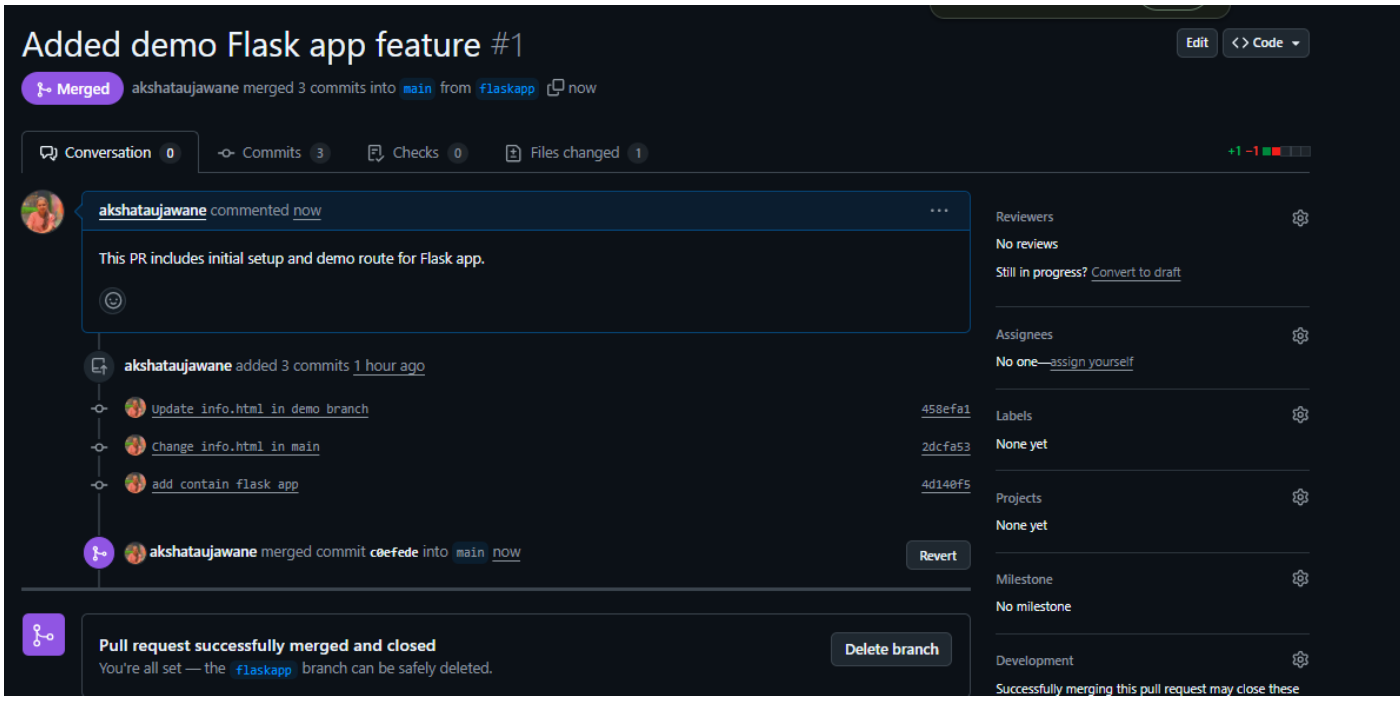
No description, website, or topics provided.

akshataujawane add contain flask app	4d140f5 · 6 minutes ago	5 Commits
README.md	Initial commit	1 hour ago
info.html	add contain flask app	6 minutes ago

No releases published
Create a new release

6. Open a Pull Request from flaskapp/demo

Go to your GitHub repository.
Navigate to the **"Pull requests"** tab.
Click on **"New pull request"**
Set the **base branch** to main and the **compare branch** to flaskapp/demo.
Review the changes and add a clear **title and description** explaining what the pull request is about.
Click on **"Create pull request"**.



Added demo Flask app feature #1

Merged akshataujawane merged 3 commits into main from flaskapp now

Conversation 0 Commits 3 Checks 0 Files changed 1 +1 −1

akshataujawane commented now

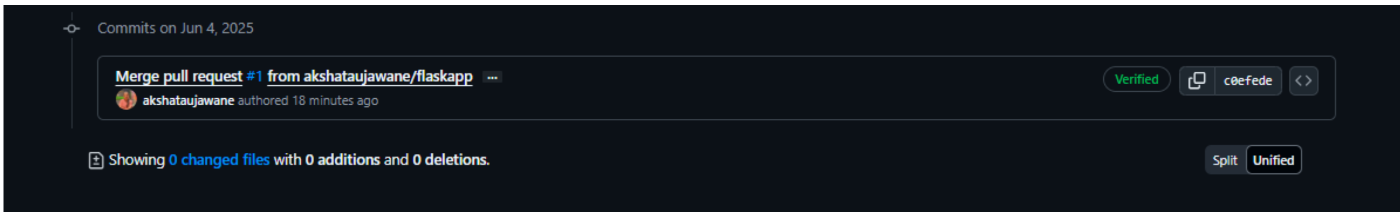
This PR includes initial setup and demo route for Flask app.

akshataujawane added 3 commits 1 hour ago

```
Update info.html in demo branch    458efa1
Change info.html in main           2dcfa53
add contain flask app              4d140f5
```

akshataujawane merged commit c0efede into main now

Pull request successfully merged and closed
You're all set — the flaskapp branch can be safely deleted.



Commits on Jun 4, 2025

Merge pull request #1 from akshataujawane/flaskapp
akshataujawane authored 18 minutes ago

Showing 0 changed files with 0 additions and 0 deletions.

7 : Merge Conflict Resolution

When two branches modify the same file/line, conflicts occur.

Make different changes in the info.html file in both branches

Step 1: On the `main` branch

```
git checkout main
echo "<h1>Main branch content</h1>" > info.html
git add info.html
git commit -m "Initial commit from main branch"
git push origin main
```

Step 2: On the demo branch

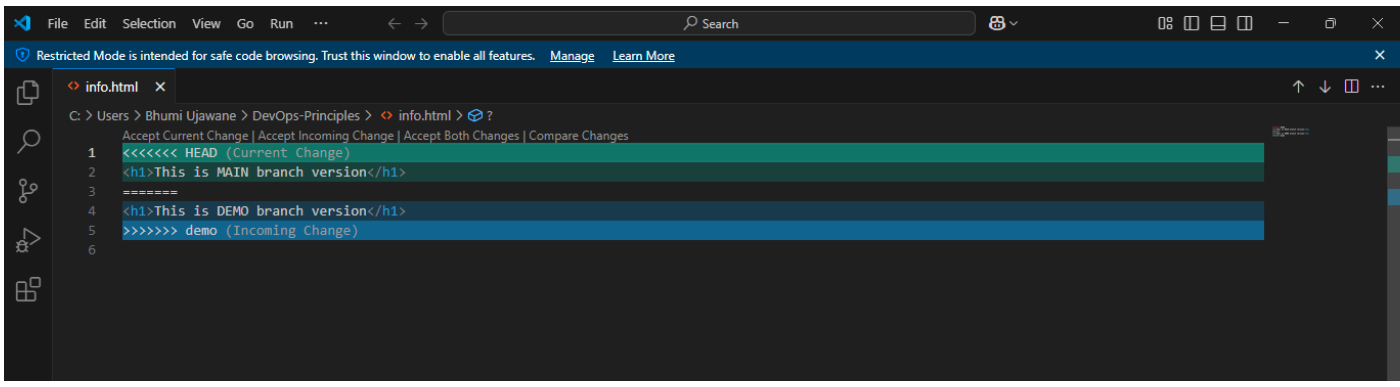
```
git checkout -b demo
echo "<h1>Demo branch content</h1>" > info.html
git add info.html
git commit -m "Edited info.html in demo branch"
git push origin demo
```

Step 3: Make conflicting changes in main

```
git checkout main
echo "<h1>Main branch updated content</h1>" > info.html
git add info.html
git commit -m "Updated info.html in main branch"
git push origin main
```

Step 4: Merge demo into main (conflict will occur)

```
git merge demo
```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

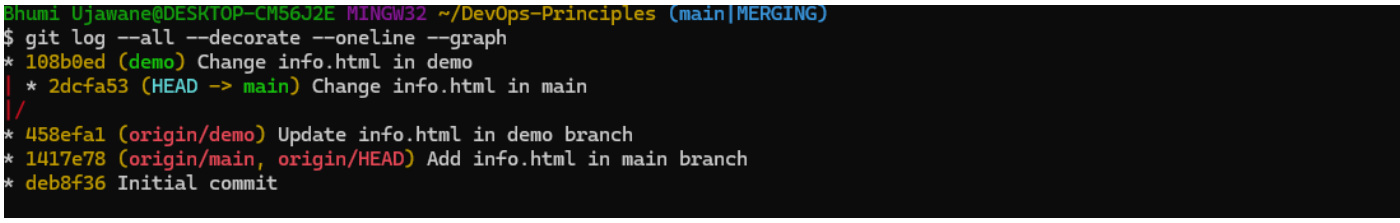
C: > Users > Bhumi Ujawane > DevOps-Principles > info.html > ?

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
<<<<<<< HEAD (Current Change)
<h1>This is MAIN branch version</h1>
=======
<h1>This is DEMO branch version</h1>
>>>>>>> demo (Incoming Change)
```

7.0 Viewing Commit History with Git Log

git log --all --decorate --oneline –graph



```
Bhumi Ujawane@DESKTOP-CM56J2E MINGW32 ~/DevOps-Principles (main|MERGING)
$ git log --all --decorate --oneline --graph
* 108b0ed (demo) Change info.html in demo
| * 2dcfa53 (HEAD -> main) Change info.html in main
|/
* 458efa1 (origin/demo) Update info.html in demo branch
* 1417e78 (origin/main, origin/HEAD) Add info.html in main branch
* deb8f36 Initial commit
```

Conclusion

In the modern DevOps landscape, GitHub plays a vital role in enabling collaboration, accelerating development, and ensuring high-quality software delivery. By offering powerful version control, streamlined branching workflows, and native CI/CD integrations like GitHub Actions, it empowers teams to build, test, and deploy with confidence.

This case study demonstrated how GitHub can be effectively used throughout the DevOps lifecycle—from code commits and pull requests to conflict resolution and automation. By adopting these practices, teams not only enhance productivity but also foster a culture of transparency and continuous improvement.

Whether you're a solo developer or part of a large enterprise team, mastering GitHub is essential for thriving in today's fast-paced software development environment.