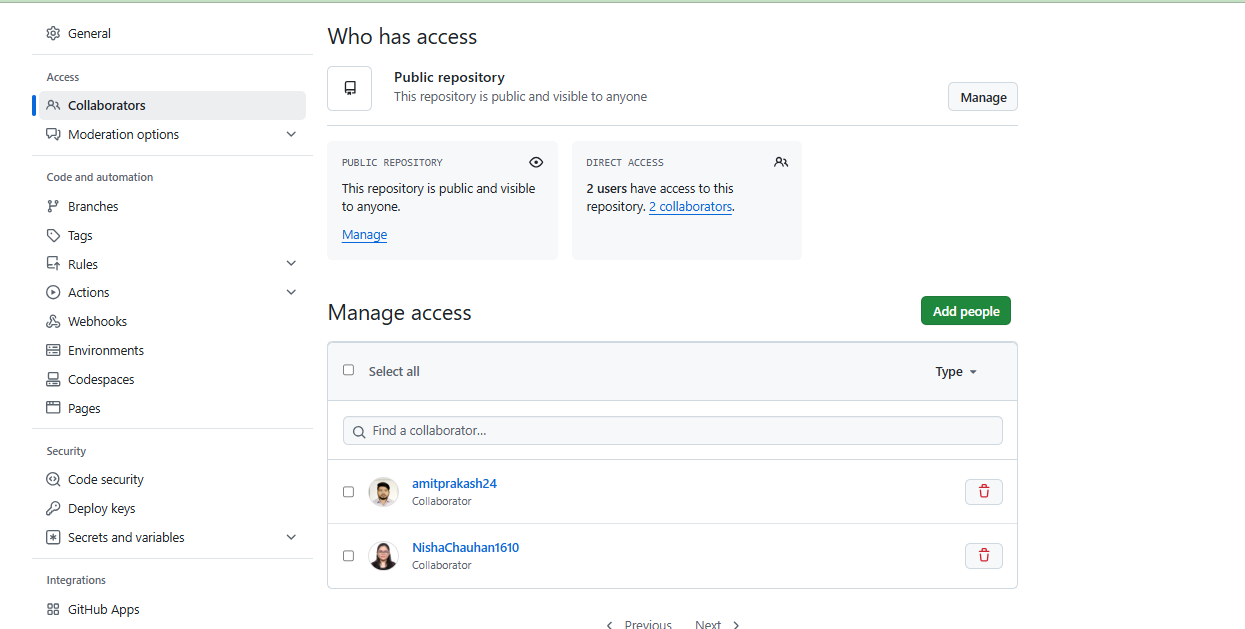
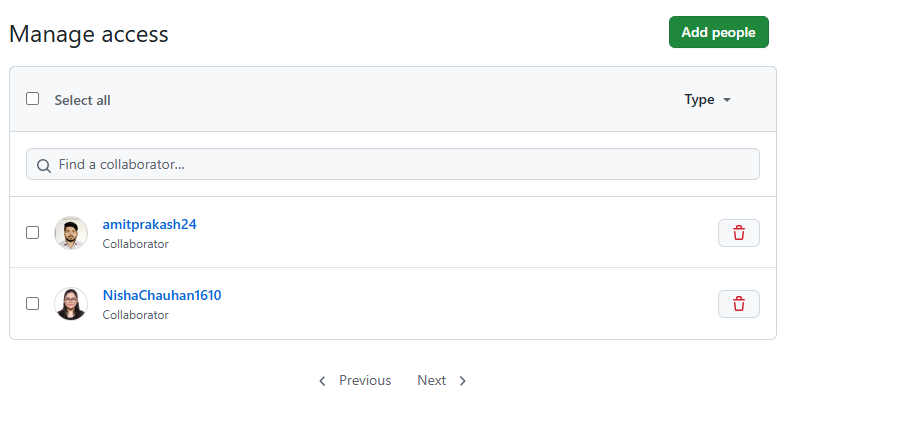
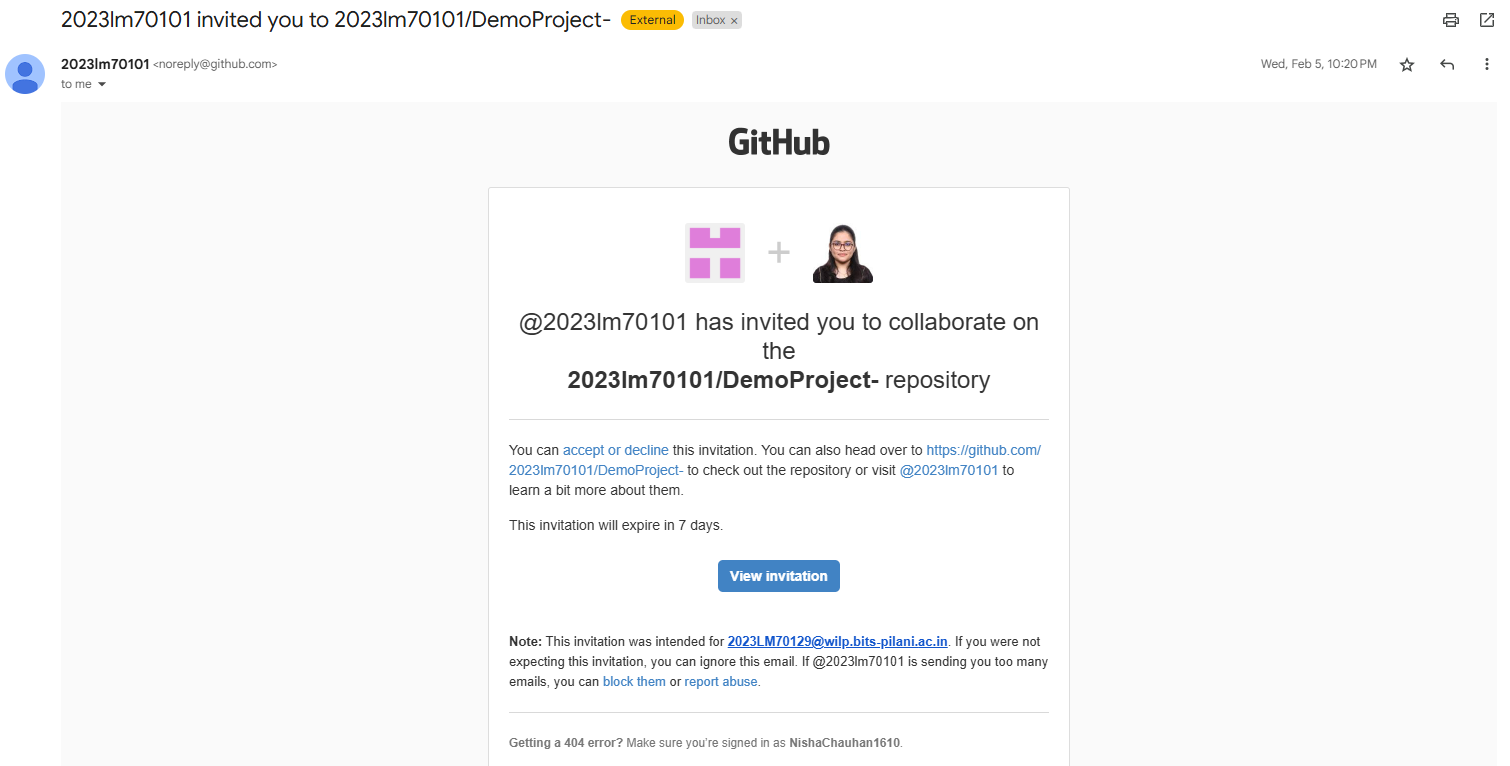
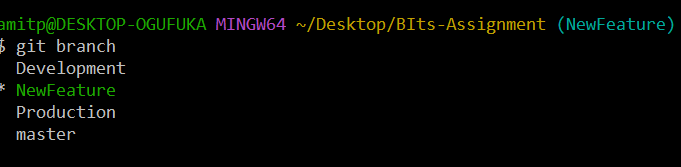
* **Add team members as collaborators and assign them appropriate rights (if you are performing individual then create one dummy account for the task**

****



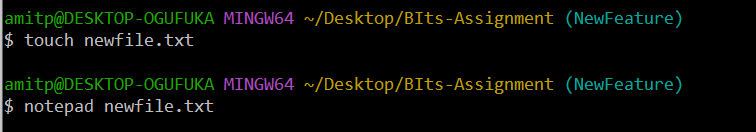
* Create a branch(development/production/feature)

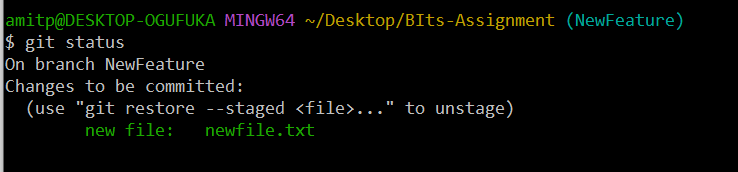


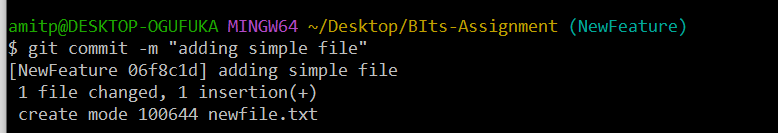


`

* **Edit files or create new files followed by commit**

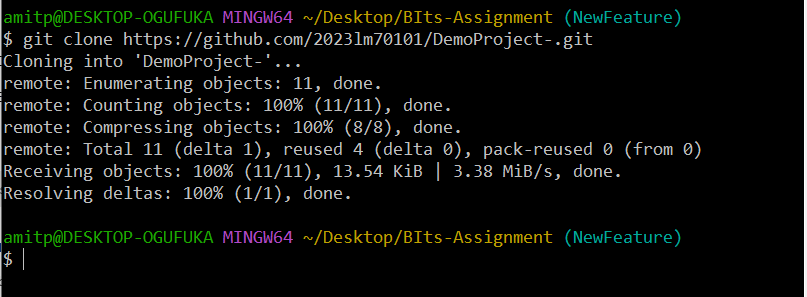
****

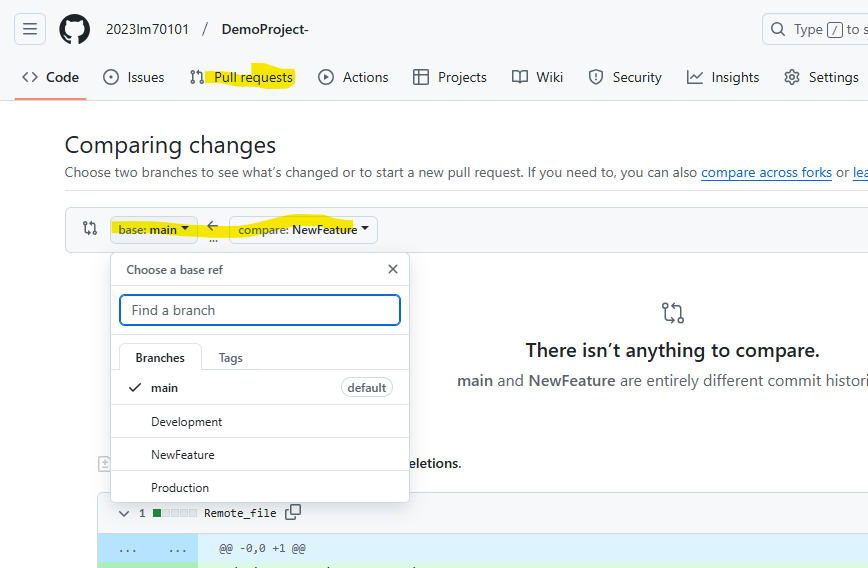


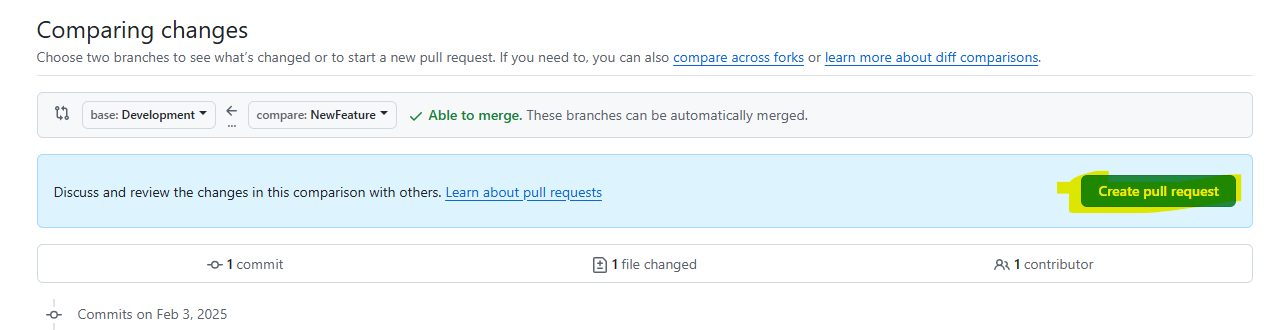
****

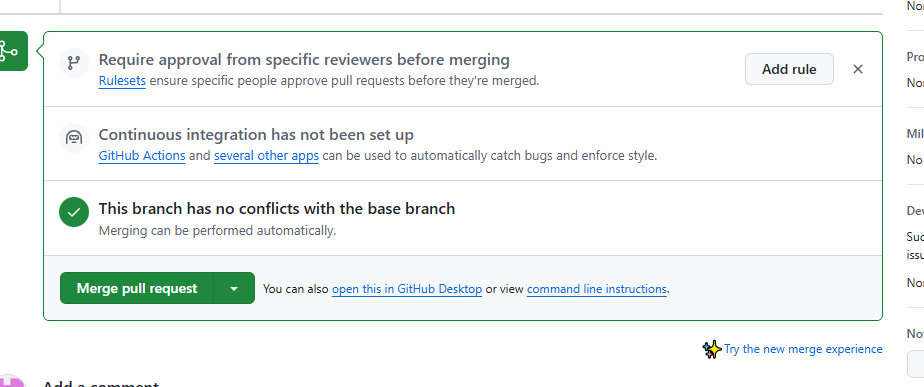
* Clone the repo and Create pull-request

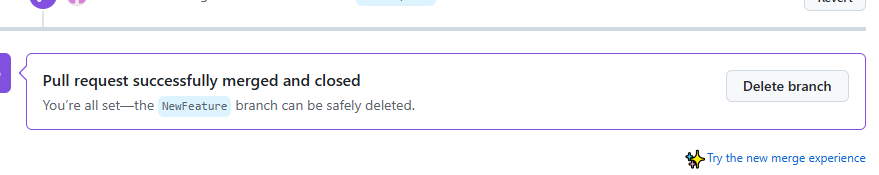
**git clone https://github.com/2023lm70101/DemoProject-.git**

****

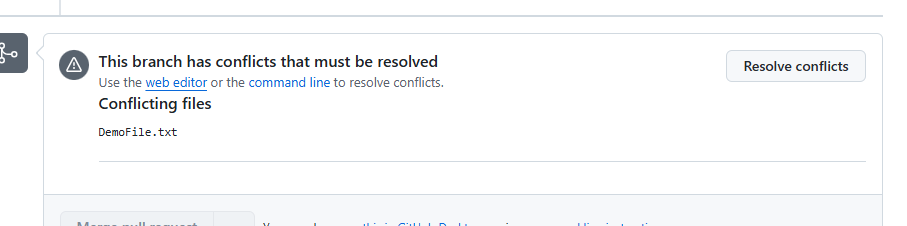


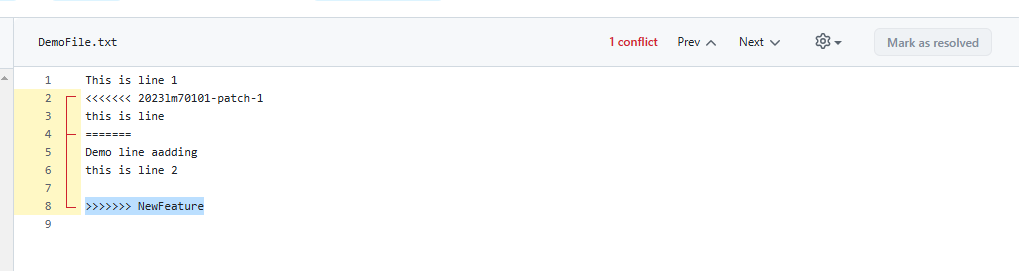


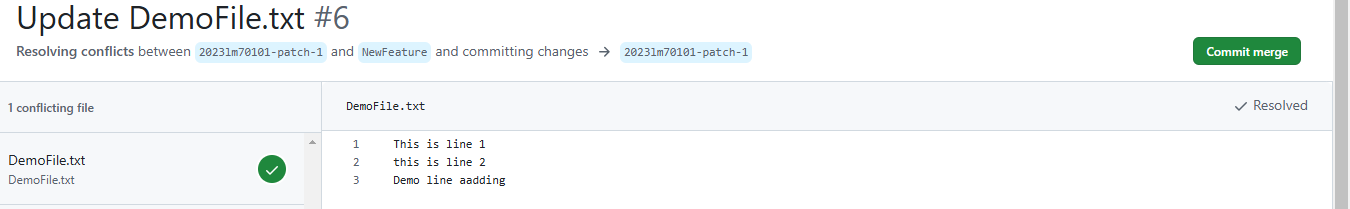


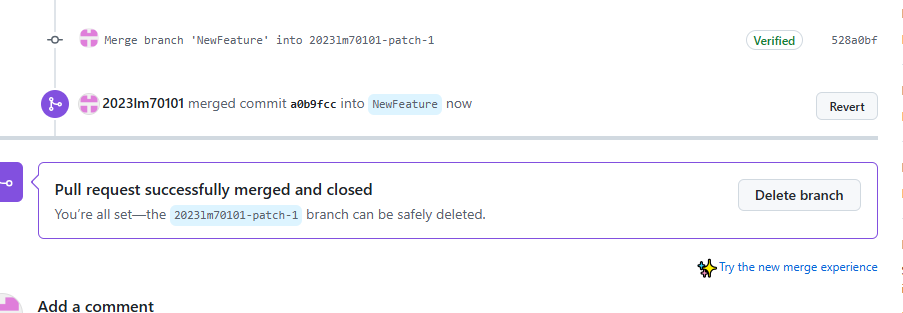


* While collaborating your work, showcase how conflicts are resolved

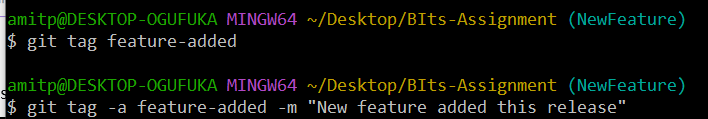


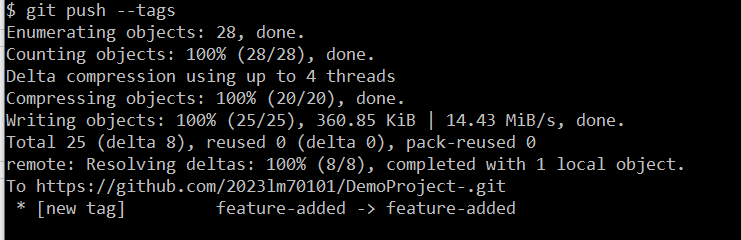




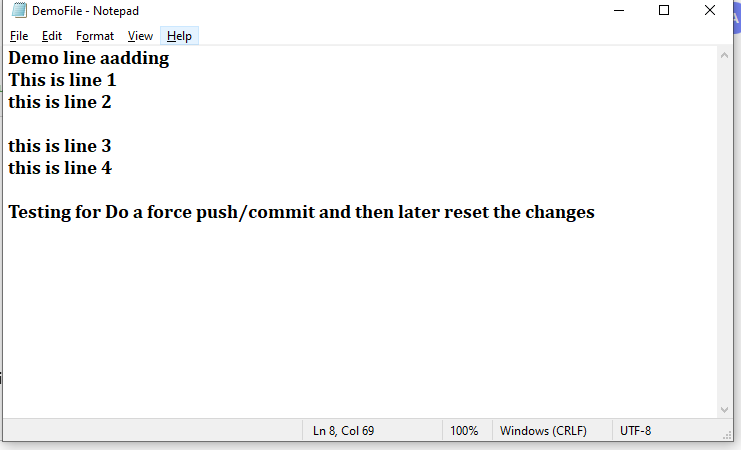


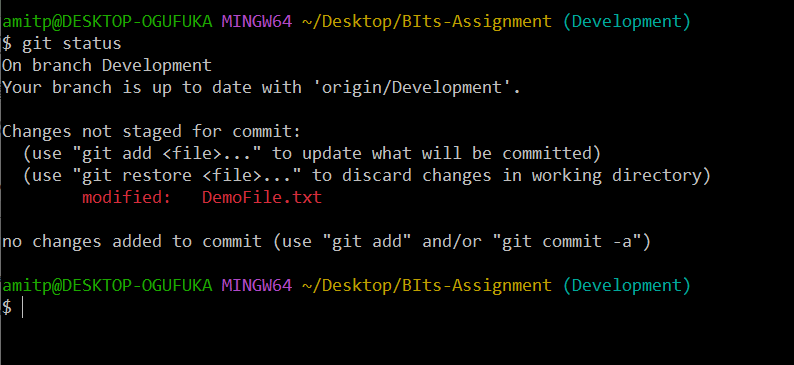
* Create tag such as open issue, or feature-added

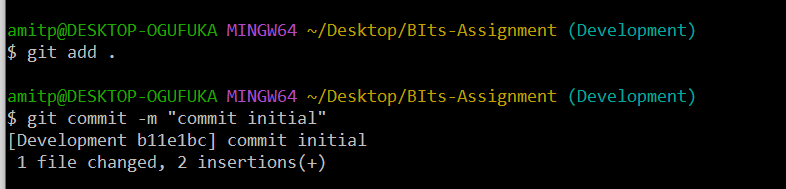


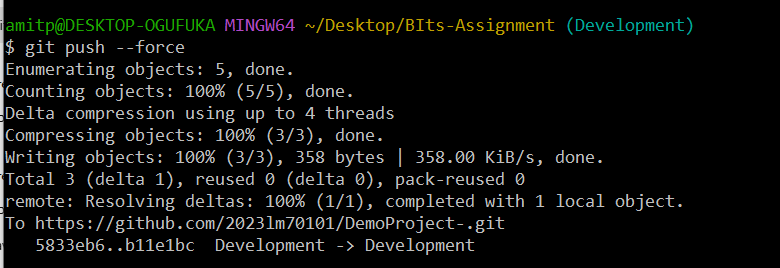


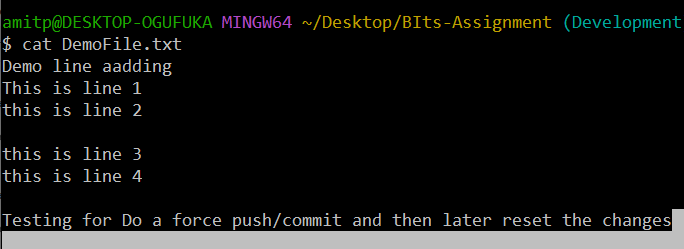
* Do a force push/commit and then later reset the changes

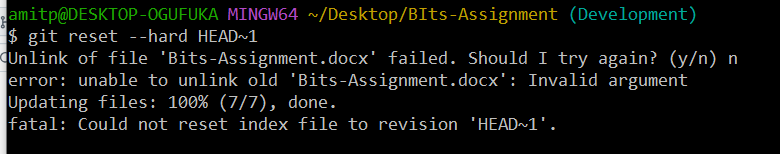






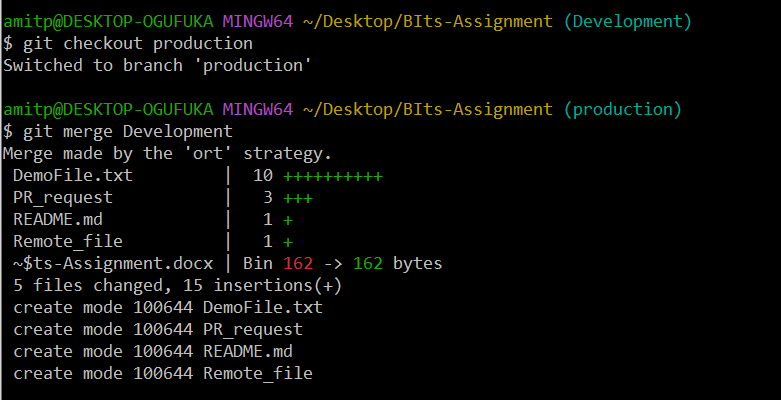


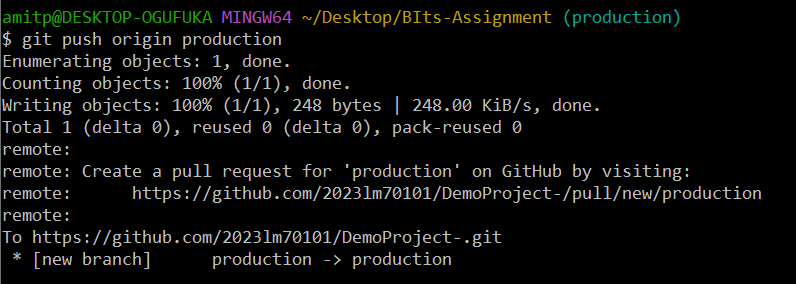




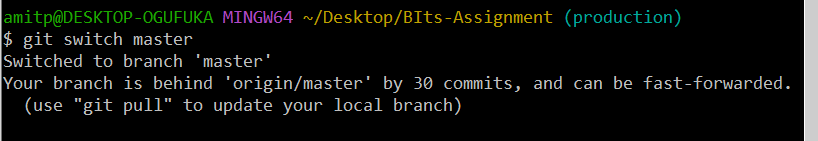


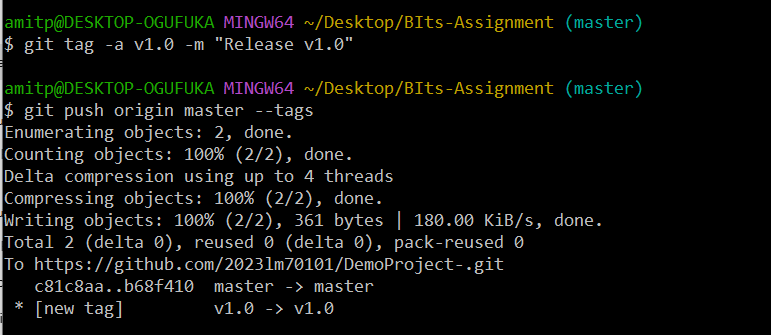
* Stage “development branch to production branch”

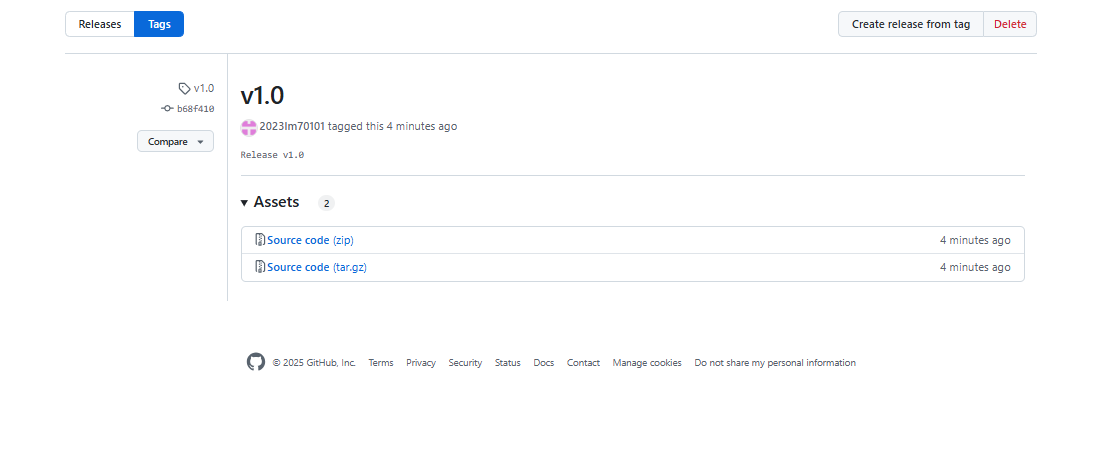




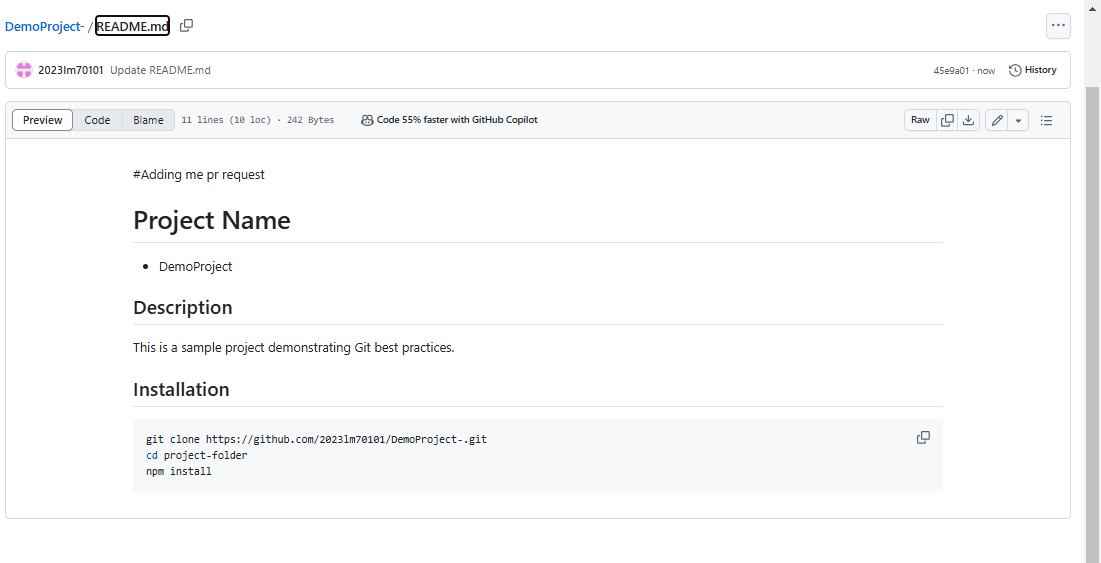
* Showcase how features are released in versions (merging production to master branch)







* Also, state importance of Readme and gitignore files and their usage while working in a distributed environment.



* Pull Requests linked with e-mail to the manager who finally approves the changes.

A screenshot of a computer

AI-generated content may be incorrect.

**A white rectangular object with a white stripe

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Assignment 2: Case Study**

**A) DevOps for Healthcare Management System**

The goal is to scale and accelerate the product delivery for a healthcare system dealing with large amounts of reports and stakeholders. DevOps can help in the following ways:

* **CI/CD (Continuous Integration / Continuous Deployment)**:
  + Implementing CI/CD pipelines will ensure automated testing and deployment, speeding up the release cycle. Every time code changes are made, the system will automatically build, test, and deploy the changes.
* **Load Balancing**:
  + Use load balancers to evenly distribute traffic across multiple servers. This will ensure the system can handle the 100 stakeholders and 1 lakh reports per month without downtime.
* **Parallel and Distributed Execution**:
  + With DevOps practices, you can implement containerization (using Docker) and orchestration (using Kubernetes) to manage workloads efficiently across multiple nodes, ensuring that data processing (like report generation) is done in parallel.
* **Error-Free System**:
  + Automated testing can be integrated into the CI/CD pipeline to catch bugs early. Also, implementing logging, monitoring, and alerting can help in error detection and resolution in real-time.
* **SaaS Deployment**:
  + Transitioning the product to a SaaS (Software as a Service) model requires scalability. Cloud platforms (AWS, Azure, GCP) provide the infrastructure to easily scale the service according to demand. Auto-scaling can adjust resources dynamically based on the workload.

**B) Product Development Challenges in a Global Team**

In the second case, the company is facing delays in delivering features due to a number of challenges:

1. **Nightly Builds**:
   * Nightly builds slow down the feedback loop. Implementing **Continuous Integration (CI)** will allow developers to commit code frequently and receive immediate feedback on whether the code works or not.
2. **Global Teams**:
   * A geographically distributed team requires tools and processes to stay aligned.
     + Use **Git** and collaboration platforms like **GitHub** to manage code effectively.
     + **Cloud-based CI/CD tools** (e.g., Jenkins, GitLab CI, Travis CI) can automate the building, testing, and deployment process for everyone, regardless of location.
     + Implement **Slack** or **Microsoft Teams** for real-time communication and daily stand-ups to maintain visibility on tasks.
3. **Agile Methodology**:
   * Adopt **Agile Scrum** or **Kanban** to manage development. Use platforms like **Jira** to track tasks, sprints, and backlogs.
4. **DevOps Culture**:
   * Shift towards a **DevOps culture**, where developers and operations teams collaborate closely, ensuring faster and more frequent delivery of product features.