Experiment 02

**Aim:**

To perform various Git operations on local and remote repositories using a Git cheat sheet and explore core version control features such as initialization, committing, branching, merging, pushing, pulling, and conflict management.

**Theory:**

**Git** is a distributed version control system (DVCS) used for tracking changes in source code during software development. It allows multiple developers to work simultaneously on a project, efficiently managing code changes, historical versions, and collaboration. Git forms the backbone of many modern **DevOps workflows**, enabling CI/CD (Continuous Integration/Continuous Deployment), automation, and versioned deployments.

**Importance of Git in DevOps:**

* **Version Control**: Every file change is tracked. You can revert to a previous state, compare changes, or recover lost data.
* **Team Collaboration**: Git enables branching so multiple developers can work independently and merge changes later.
* **Code Integrity**: Changes can be reviewed, tested, and only then merged—ensuring quality and reliability.
* **Deployment Ready**: Git integrates easily with deployment platforms like **Netlify**, **Vercel**, **GitHub Actions**, etc.

**Key Git Concepts & Commands:**

1. **Configuration**:
   * git config is used to set user-specific configurations like name and email for commits.
2. **Repository Setup**:
   * git init initializes a local repository.
   * git clone <url> clones a remote repository to the local machine.
3. **Staging & Committing**:
   * git add stages changes.
   * git commit -m "message" saves changes with a message.
4. **Branching**:
   * git branch lists branches.
   * git checkout -b <name> creates and switches to a new branch.
5. **Merging**:
   * git merge <branch> integrates changes from one branch into another.
6. **Remote Sync**:
   * git push uploads local commits to the remote repo.
   * git pull fetches and merges changes from the remote.
7. **History & Undo**:
   * git log, git diff, git reset, and git revert allow inspection and rollback of changes.
8. **Conflict Resolution**:
   * If changes in two branches overlap, Git flags a **merge conflict**, which must be manually resolved before continuing.

Git plays a **critical role** in DevOps pipelines where **automation, reproducibility, and rollback capabilities** are essential. Learning Git ensures developers and operations teams can collaborate effectively, resolve issues quickly, and deliver software reliably.

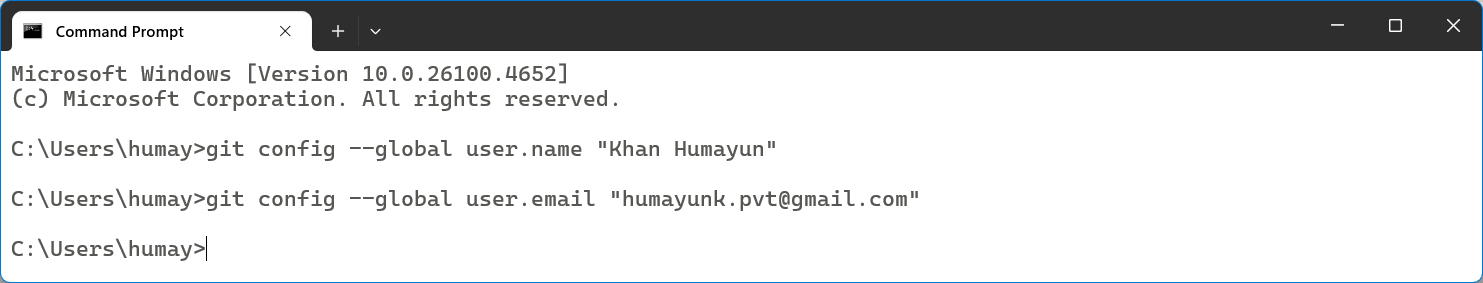
**Procedure:**

* 1. **Initial Git Setup**

Set up user identity globally:

>> git config --global user.name "Khan Humayun"

>> git config --global user.email "humayunk.pvt@gmail.com"

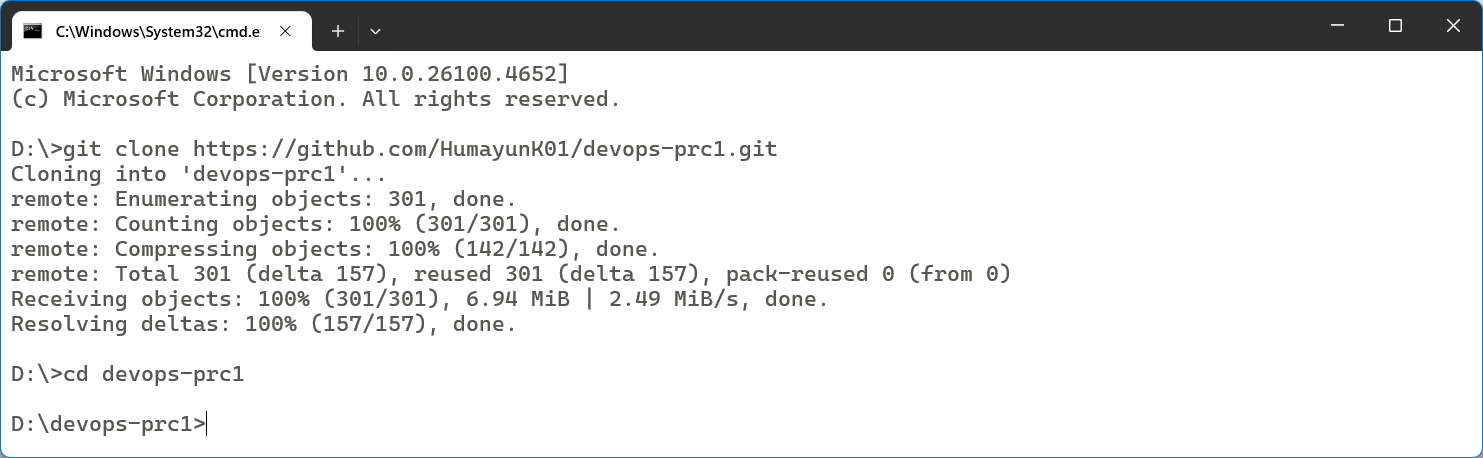


* 1. **Clone Existing Repository**

Use the repo created in Experiment 1 and clone it locally:

>> git clone https://github.com/HumayunK01/devops-prc1.git

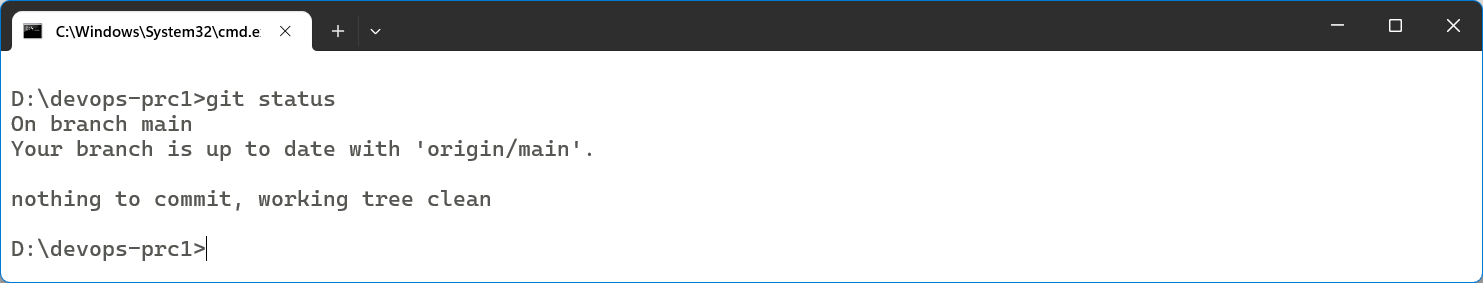
>> cd devops-prc1



* 1. **Check Repository Status**

Check the current status of files:

>> git status

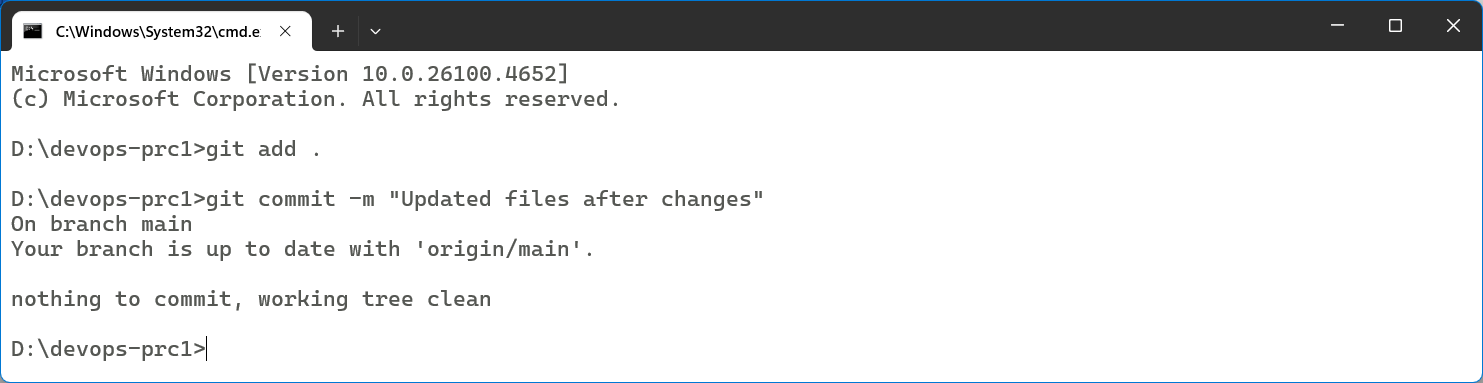


* 1. **Track and Commit Changes**

Stage all files and commit with a message:

>> git add .

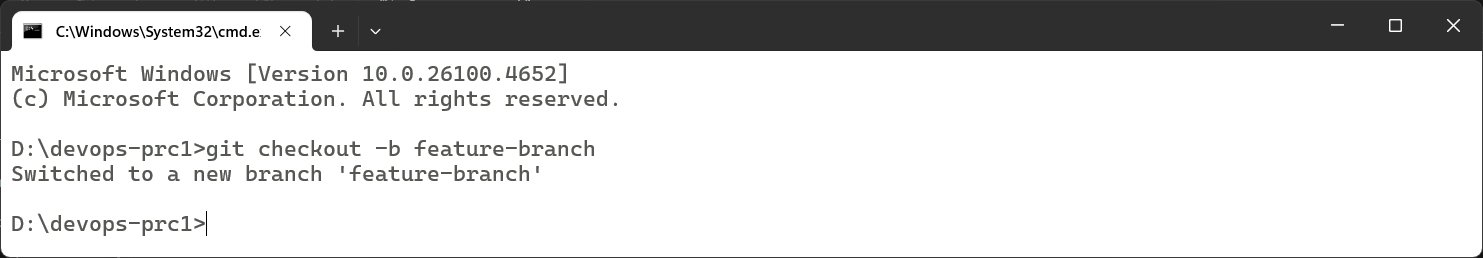
>> git commit -m "Updated files after changes"



* 1. **Create & Switch to a New Branch**

Create a new branch and switch to it:

>> git checkout -b feature-branch



* 1. **Make Edits and Merge Branch**

Modify a file, commit it, then merge the branch:

>> git add .

>> git commit -m "Changes in feature branch"

>> git checkout main

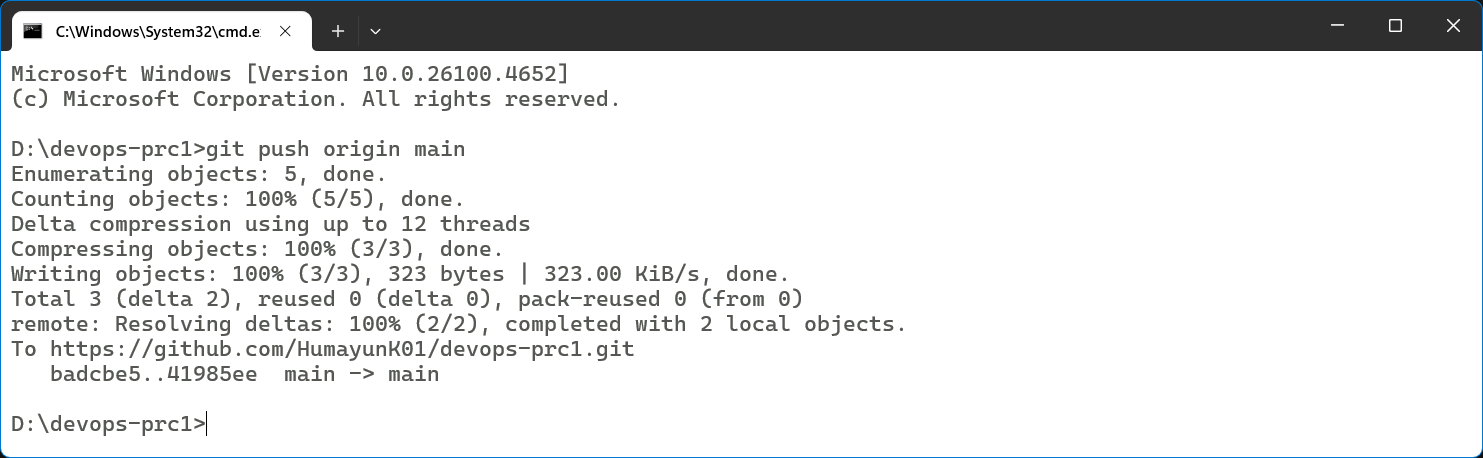
>> git merge feature-branch



* 1. **Push Changes to Remote Repo**

Push the local changes to the online repository:

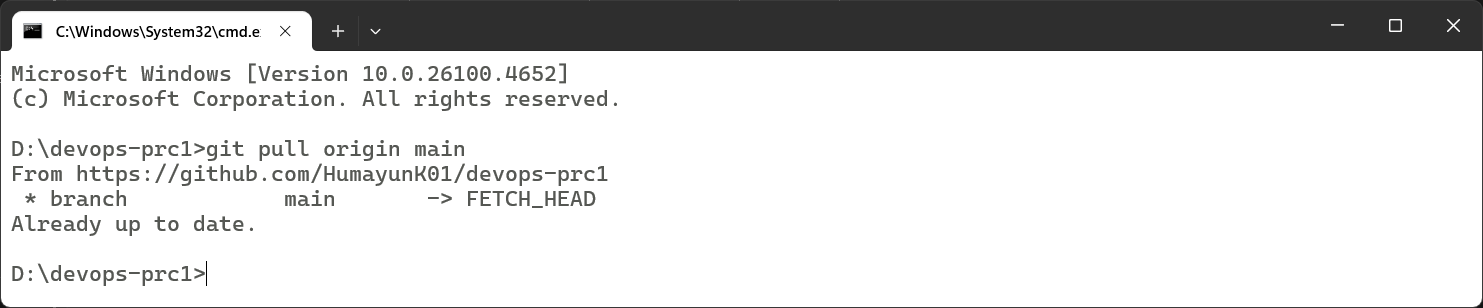
>> git push origin main



* 1. **Pull Remote Changes**

Fetch the latest changes from the remote repo:

>> git pull origin main

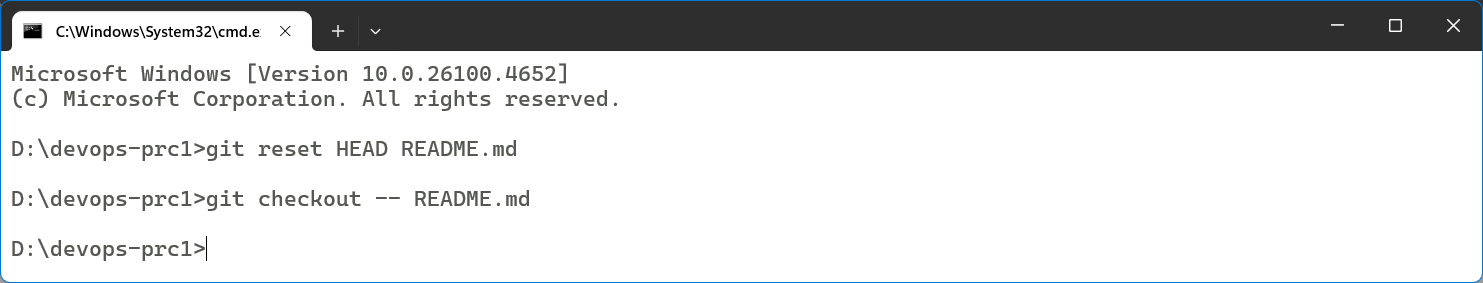


* 1. **Undoing Changes (if required)**

Unstage a file and discard local modifications:

>> git reset HEAD <filename>

>> git checkout -- <filename>



**Output:**

All commands were successfully executed. The cloned repository was modified, committed, merged, and pushed to the remote GitHub repository.

**Conclusion:**

By performing this experiment, we have developed a deeper understanding of using Git for real-world development workflows. This hands-on experience with Git commands has shown how version control systems enable teamwork, code reliability, and smooth project delivery in DevOps environments.