Version-Controlled DevOps Project with Git

Objective:

Manage a DevOps project using **Git best practices** for version control.

Tools Required:

- Git
- GitHub
- A properly structured Git repository
- Well-managed branches (main, dev, feature)
- Proper commits and pull requests (PRs)
- A README.md file for project documentation
- A .gitignore file to exclude unnecessary files
- Tags for versioning

✅ Step 1: Initialize a Git Repository and Push to GitHub

1. Create a New GitHub Repository

1. **Go to GitHub**
2. Click **"New Repository"**
3. Enter a repository name
4. Choose **Public** or **Private**
5. Check **"Add a README file"** (optional)
6. Click **"Create Repository"**

2. Clone the Repository Locally

git clone https://github.com/your-username/devops-project.git

cd devops-project

```
   git commit --amend --reset-author

 5 files changed, 202 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .terraform.lock.hcl
 create mode 100644 main.tf
 create mode 100644 terraform.tfstate
 create mode 100644 terraform.tfstate.backup
macbookair@MACBOOKs-MacBook-Air terraform % code .
macbookair@MACBOOKs-MacBook-Air terraform % cd
macbookair@MACBOOKs-MacBook-Air ~ % ls
Desktop                 Music                   dockerfile              my-jenkins-project      terraform-docker
Documents               Pictures                git                     my-node-app             whatspp
Downloads               Public                  hid                     sai.txt
Library                 crud-nodejs-mysql       index.html              stroge
Movies                  david                   k8s                     terraform
macbookair@MACBOOKs-MacBook-Air ~ % mkdir devops-project
macbookair@MACBOOKs-MacBook-Air ~ % cd devops-project
macbookair@MACBOOKs-MacBook-Air devops-project % ls
macbookair@MACBOOKs-MacBook-Air devops-project % git clone https://github.com/David-raj-11/devops-project.git
Cloning into 'devops-project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
macbookair@MACBOOKs-MacBook-Air devops-project % ls
devops-project
macbookair@MACBOOKs-MacBook-Air devops-project % cd devops-project

macbookair@MACBOOKs-MacBook-Air devops-project % ls
README.md
macbookair@MACBOOKs-MacBook-Air devops-project % git checkout -b dev
git push -u origin dev

Switched to a new branch 'dev'
Username for 'https://github.com': David-raj-11
Password for 'https://David-raj-11@github.com':
remote: Support for password authentication was removed on August 13, 2021.
```

✅ Step 2: Create Git Branches

1. Create a dev Branch

git checkout -b dev

git push -u origin dev

2. Create a feature-1 Branch

git checkout -b feature-1

git push -u origin feature-1

Verify branches on GitHub under **"Branches"**.

✅ Step 3: Add .gitignore

1. Create and Edit .gitignore

touch .gitignore

echo "node_modules/" >> .gitignore

echo ".terraform/" >> .gitignore

echo "*.log" >> .gitignore

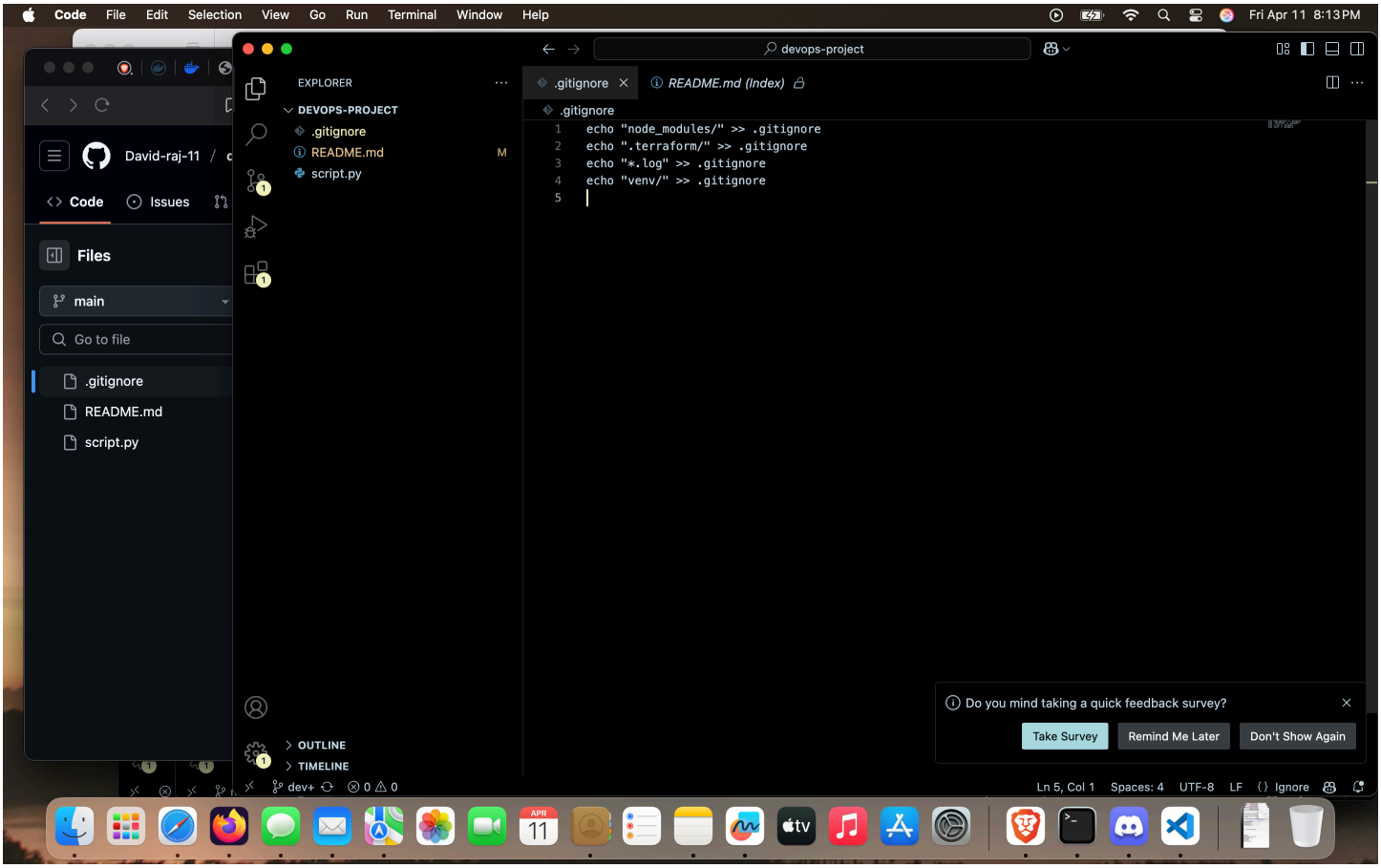
echo "venv/" >> .gitignore

2. Commit and Push

git add .gitignore

git commit -m "Added .gitignore"

git push origin dev



✅ Step 4: Add README.md

1. Create README.md

echo "# DevOps Project" > README.md

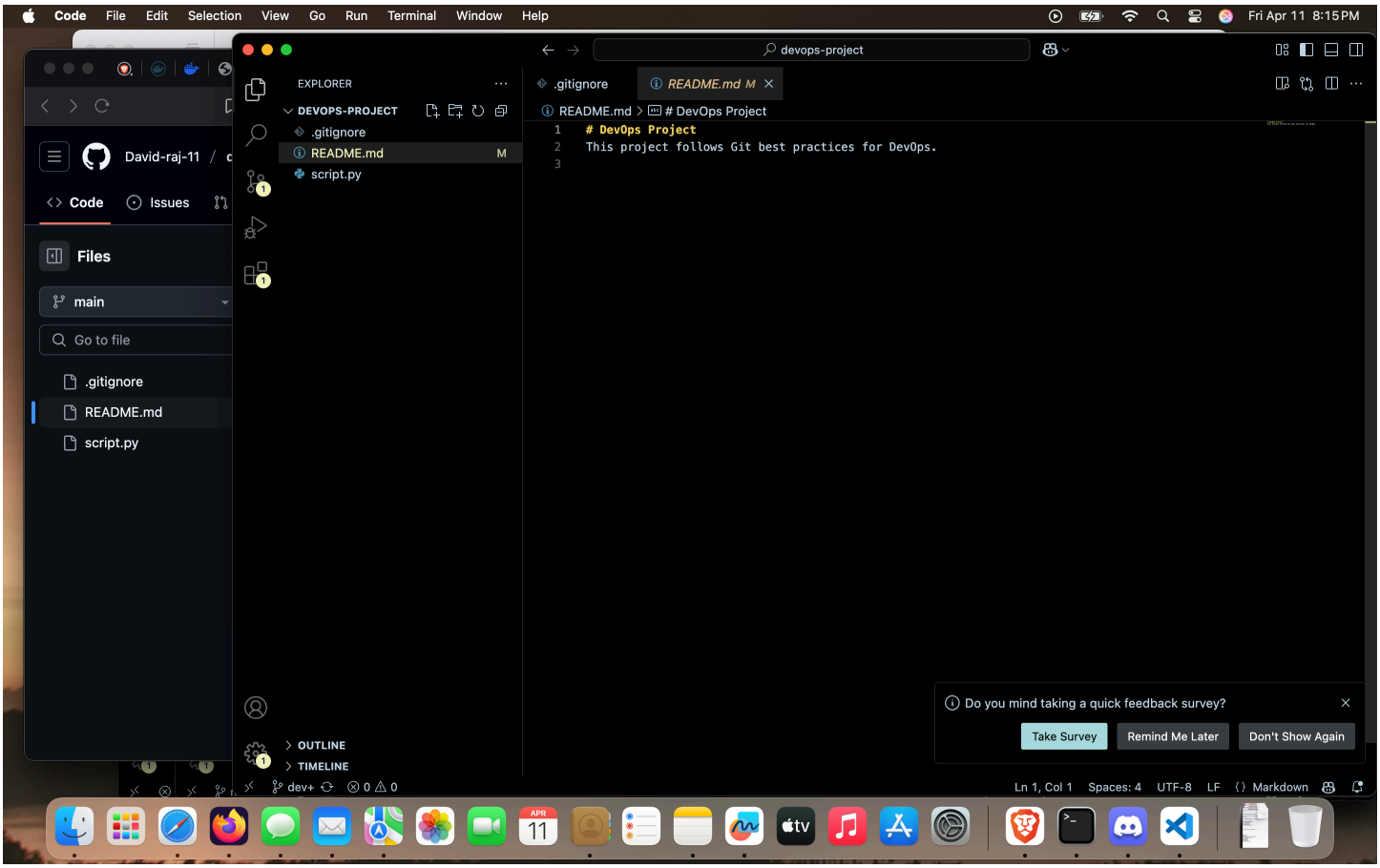
echo "This project follows Git best practices for DevOps." >> README.md

2. Commit and Push

git add README.md

git commit -m "Added README.md"

git push origin dev



✅ Step 5: Work on a Feature Branch

1. Switch to feature-1 Branch

git checkout feature-1

2. Create a New Script

echo "print('Hello from feature-1')" > script.py

3. Commit and Push

git add script.py

git commit -m "Added script.py in feature-1"

git push origin feature-1

✅ Step 6: Create a Pull Request (PR)

1. **Go to GitHub** → Open the repository.
2. Click **"Pull Requests"** → **"New Pull Request"**.
3. **Select feature-1 → dev**.
4. Click **"Create Pull Request"**.
5. Add a description and click **"Merge"**.

✅ Step 7: Merge dev into main

Once tested, merge dev into main.

git checkout main

git merge dev

git push origin main

2. Commit and Push

git add docs.md

git commit -m "Added documentation"

git push origin dev