Lab 2: Collaborative Development with Git & CI/CD Workflows

COMP 395-2 – Deep Learning

Week 2

Duration: 50 minutes (may be completed asynchronously)
Team Size: 2–3 students
Points: 100

Learning Objectives

By the end of this lab, you will be able to:

1. Clone a shared repository and set up a Python environment using `conda`
2. Create and manage feature branches following standard Git workflows
3. Submit pull requests with meaningful descriptions
4. Conduct peer code reviews and provide constructive feedback
5. Merge approved changes and maintain a clean repository history
6. *(Extension)* Convert a Jupyter notebook to a CLI script with automated tests

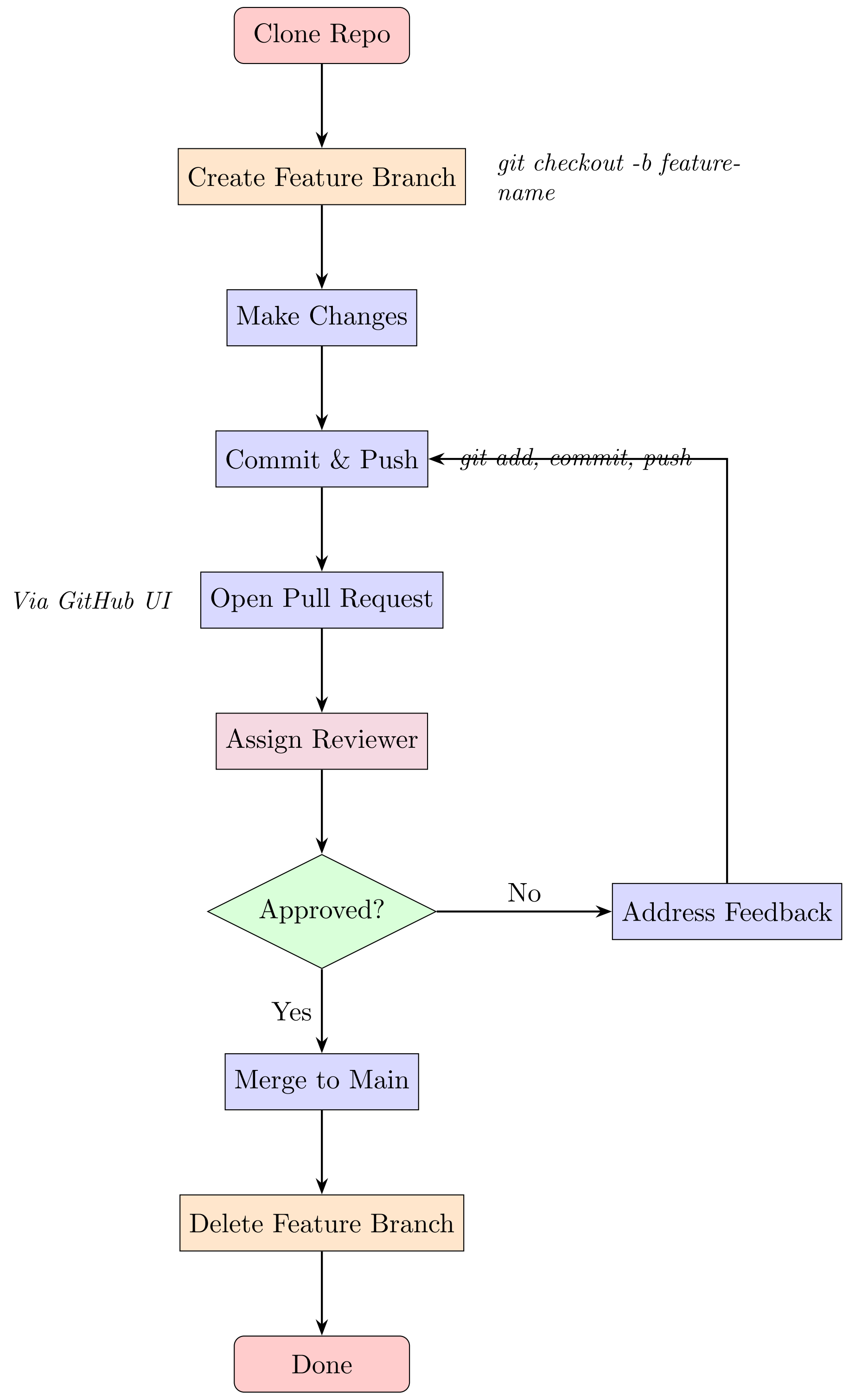
Context

Professional software development rarely happens in isolation. Teams use **version control** (Git) and **CI/CD pipelines** (Continuous Integration/Continuous Deployment) to collaborate effectively, catch bugs early, and deploy reliably. This lab introduces the *feature branch workflow*—a pattern used across industry and open-source projects, including most deep learning research repositories.

You will take your existing GPU benchmarking notebook from Lab 1 and collaborate with your team to add features, practicing the pull request and code review cycle that you'll use throughout this course and your careers.

CI/CD Feature Branch Workflow

The diagram below illustrates the workflow each team member will follow. Study this before beginning.



Repository Structure

Your team will work with a template repository structured as follows:

```
repo-name/
    gpu_benchmark.ipynb          # Main notebook (team edits this)
    other-benchmarks/            # Individual team member notebooks
        alice_benchmark.ipynb
        bob_benchmark.ipynb
        ...
    requirements.txt             # Dependencies (conda export)
    README.md
```

Instructions

Phase 1: Repository Setup (10 min) – *One team member only*

1. **Accept the GitHub Classroom assignment** at: https://classroom.github.com/a/9RrEenyz
2. **Generate requirements.txt** from your conda environment:

   ```
   conda list --export > requirements.txt
   ```

3. **Commit and push** the initial setup:

   ```
   git add requirements.txt
   git commit -m "Add conda requirements"
   git push origin main
   ```

4. **Share the repository URL** with your teammates.

Phase 2: Clone and Environment Setup (5 min) – *All team members*

1. **Clone the repository**:

   ```
   git clone <repository-url>
   cd <repo-name>
   ```

2. **Create/activate your conda environment** and install dependencies:

   ```
   conda create --name comp395-dl --file requirements.txt
   conda activate comp395-dl
   ```

   Note: If you encounter issues, you may need to install packages manually with pip.

Phase 3: Feature Development Cycle (25 min) – *Each team member*

Each team member must complete one feature. Coordinate to avoid merge conflicts (work on different cells or different sections of the notebook). Example features:

- Add GPU memory usage tracking using `torch.cuda.memory_allocated()`
- Compare `float16` vs `float32` performance (mixed precision)
- Add statistical analysis: run each size 5 times, report mean $\pm$ std

- Benchmark element-wise operations (add, multiply) vs matrix multiply
- Add a heatmap visualization of speedup across sizes
- Test batch matrix multiplication (`torch.bmm`)
- Add CPU/GPU temperature or utilization monitoring

For each feature, follow these steps:

1. **Create a feature branch** (use a descriptive name):

   ```
   git checkout -b feature/add-memory-tracking-yourname
   ```

2. **Implement your feature** in the notebook. Test that it runs correctly.
3. **Stage, commit, and push** your changes:

   ```
   git add gpu_benchmark.ipynb
   git commit -m "Add GPU memory usage tracking per matrix size"
   git push -u origin feature/add-memory-tracking-yourname
   ```

4. **Open a Pull Request** on GitHub:
   - Navigate to your repository on GitHub
   - Click "Compare & pull request"
   - Write a meaningful title and description explaining your changes
   - Assign a teammate as a reviewer
5. **Review a teammate's PR**:
   - Go to the "Pull requests" tab
   - Review the code changes (Files changed tab)
   - Leave at least one constructive comment
   - Approve the PR (or request changes if needed)
6. **Merge and clean up** (after approval):
   - Click "Merge pull request" then "Confirm merge"
   - Delete the feature branch (GitHub will prompt you)
7. **Update your local repository**:

   ```
   git checkout main
   git pull origin main
   ```

Phase 4: Verification (5 min)

Before finishing, verify:

Each team member has at least one merged PR

All feature branches have been deleted

The main branch contains all features and runs without errors

Each team member has reviewed at least one PR

Extension Task (If Time Permits)

If your team completes the main lab early, tackle this extension:

Convert to CLI Script with Tests

1. **Create a Python script** that runs from the command line:

```
# Target usage:
python benchmark_cli.py --sizes 512 1024 2048 --output results.json
```

2. **Create a test directory** with at least one pytest test:

```
repo-name/
    tests/
        __init__.py
        test_benchmark.py
    benchmark_cli.py
```

3. **Example test** (tests/test_benchmark.py):

```
import pytest
import torch
from benchmark_cli import create_matrices, benchmark_matmul

def test_create_matrices_shape():
    A, B = create_matrices(256, torch.device("cpu"))
    assert A.shape == (256, 256)
    assert B.shape == (256, 256)

def test_create_matrices_dtype():
    A, B = create_matrices(128, torch.device("cpu"))
    assert A.dtype == torch.float32
    assert B.dtype == torch.float32

def test_benchmark_returns_positive_time():
    A, B = create_matrices(128, torch.device("cpu"))
    time_ms = benchmark_matmul(A, B)
    assert time_ms > 0
```

4. **Run tests**:

```
pytest tests/ -v
```

5. Follow the same PR workflow to merge the CLI script and tests.

Grading Rubric

Criterion	**Points**	**Earned**
Repository Setup (Team)		
Repository created with correct structure	5	
`requirements.txt` generated and committed	5	
Individual Contributions (Per Team Member)		
Feature branch created with descriptive name	10	
Meaningful feature implemented and working	20	
Clear commit message(s)	5	
Pull request with descriptive title and body	10	
Assigned reviewer to PR	5	
Provided constructive code review on teammate's PR	15	
Successfully merged PR to main	10	
Deleted feature branch after merge	5	
Final Verification		
Main branch runs without errors	10	
Total	**100**	

Extension Bonus (up to 15 points):

- CLI script with argument parsing: +5 pts
- Test directory with passing pytest: +5 pts
- Tests merged via PR workflow: +5 pts

Submission

No separate submission required. Your instructor will verify completion by reviewing:

1. Your team's GitHub repository
2. Merged pull requests and commit history
3. Deleted feature branches
4. Code review comments

Ensure all work is pushed to GitHub before the end of the lab period (or by the asynchronous deadline if finishing at home).

Troubleshooting Tips

Merge conflicts: If you encounter a merge conflict, coordinate with your teammate. You can resolve conflicts locally:

```
git checkout main
git pull origin main
git checkout your-feature-branch
git merge main
# Resolve conflicts in your editor, then:
git add .
git commit -m "Resolve merge conflicts"
git push
```

Can't push: Make sure you're on your feature branch, not main:

```
git branch  # Check current branch
git checkout feature/your-feature  # Switch if needed
```

Conda issues: If `requirements.txt` doesn't work, try:

```
pip install torch numpy matplotlib jupyter
```

GPU not detected: Ensure you have CUDA installed (for NVIDIA) or are using MPS (for Apple Silicon). Check with:

```
import torch
print(torch.cuda.is_available())  # NVIDIA
print(torch.backends.mps.is_available())  # Apple Silicon
```

Remember: The goal is to practice the workflow, not to create perfect code.
Ask for help if you get stuck!