Group Assignment

COMP 3104 – DevOps **Total: 16% (100 Points)**

Submission Deadline: Sunday, 20th Oct 2025, 11:59 pm (Week06)

Completing a group assignment involving GitHub collaboration, branching, and continuous integration (CI) can be streamlined by following a structured approach. Below is a step-by-step guide to help your group successfully complete the **COMP3104 DevOps** assignment:

1. Forming Your Group

- **Group Size:** Ensure your group has **3 to 4 members**.
- **Communication:** Decide on a primary communication channel (e.g., Slack, email) to coordinate tasks and share updates.

2. Creating the GitHub Repository

- **Repository Name Format:** COMP3104_Group#_Assignment (e.g., COMP3104_Group1_Assignment).
- **Repository Type:** Public
- **Repository Owner:** Only one member (Leader) will create the repository.

Reference Steps:

1. **Leader Creates Repository:**
   - Log in to GitHub.
   - Click on "**New**" to create a new repository.
   - Enter the repository name following the specified format.
   - Set the repository to **Public**.
   - Initialize with a **README.md** if desired.
2. **Add Collaborators:**
   - Go to the repository's **Settings**.
   - Navigate to "**Manage access**".
   - Click "**Invite a collaborator**" and add the GitHub usernames of the other group members.
   - Assign appropriate permissions (e.g., **Write** access).

3. Assigning Roles

The leader will add the remaining group members as collaborators with the necessary permissions to push and manage branches.

- **Leader/Owner:** Manages the repository and adds collaborators.
- **Collaborators:** Clone the repository and contribute via their branches.

4. Branch Management

Each member will create a branch named with their STUDENTID-Name

- **Branch Naming Convention:** STUDENTID-Name (e.g., 1023756-Pritesh).

Reference Steps for Each Member:

1. **Clone the Repository:**

   git clone https://github.com/YourUsername/COMP3104_Group#_Assignment.git

2. **Navigate to the Repository:**

   cd COMP3104_Group#_Assignment

3. **Create and Switch to Your Branch:**

   git checkout -b STUDENTID-Name

4. **Push the Branch to GitHub:**

   git push -u origin STUDENTID-Name

5. Setting Up Continuous Integration (CI)

One member will integrate the repository with GitHub Actions or another CI tool to automate the build and testing processes.

- **CI Tools:** GitHub Actions (recommended) or any free CI tool of your choice.
- **Responsibility:** Only the Leader sets up the CI pipeline.

Using GitHub Actions:

1. **Navigate to the Repository on GitHub.**
2. **Go to the "Actions" Tab.**
3. **Set Up a Workflow:**
   - Choose a template or create a new workflow.
   - For example, a basic workflow can be set to run on push events.
4. **Commit the Workflow File:**
   - This file will be located in .github/workflows/.

Sample GitHub Actions Workflow (ci.yml): Modify the workflow according to your project's requirements.

```
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
```

```
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
      - run: echo "🐧 This job is now running on a ${{ runner.os }} server hosted by GitHub!"
      - run: echo "🔎 The name of your branch is ${{ github.ref }} and your repository is ${{
github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "💡 The ${{ github.repository }} repository has been cloned to the runner."
      - run: echo "🖥️ The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

6. Making Commits and Creating Files

Each Member Must:

- **Create at Least 10 Commits:**
  - Regularly commit changes with meaningful messages.
- **Create 3 Specific Files:** **(Think of your own content)**
  - **Format:** Studentid_filename.txt (e.g., 1023756_gb.txt)
  - **Contents:**
    1. **Studentid_gb.txt:** Information about George Brown College.
    2. **Studentid_devops.txt:** Information about the COMP 3104 DevOps course.
    3. **Studentid_sdlc.txt:** Importance of learning DevOps in the software development cycle.

Reference Steps:

1. **Create Files:** **(You can use any file editor or IDE)**

   touch 1023756_gb.txt 1023756_devops.txt 1023756_sdlc.txt

2. **Add Content:**
   - Open each file in a text editor and add the required information.
3. **Stage and Commit:** **(Put your own commit message)**

   git add 1023756_gb.txt 1023756_devops.txt 1023756_sdlc.txt
   git commit -m "Add gb, devops, and sdlc information"

4. **Repeat:** Ensure you make at least 10 commits, each with clear and descriptive messages.

Merge their branch into the main branch.

7. Merging Branches and Resolving Conflicts

- Regularly pull updates from the remote repository to your local branches.
- Resolve any merge conflicts that arise.

8. Updating README.md

Responsibilities:

- **All Members:** Update the README.md to include repository details and group information.

Suggested Sections:

1. **Project Title**
2. **Group Members:**
   - List each member with their Student ID and GitHub username.
3. **Project Description:**
   - Brief overview of what the assignment entails.
4. **Instructions:**
   - How to set up the project locally.
5. **CI/CD Details:**
   - Information about the CI pipeline setup.
6. **Branching Strategy:**
   - Explain the branching conventions used.

Reference Example README.md: **(Put your own details)**

 # COMP3104_Group1_Assignment

Group Members
*- **Leader:** Pritesh Patel (1023756) - [GitHub](https://github.com/priteshpatel)*
*- **Member 2:** Jane Doe (1023456) - [GitHub](https://github.com/janedoe)*
*- **Member 3:** John Smith (1027890) - [GitHub](https://github.com/johnsmith)*

Project Description
This repository hosts the group assignment for COMP3104 DevOps course, focusing on collaborative Git workflows, branching strategies, and CI/CD integration.

Setup Instructions
1. Clone the repository.
2. Switch to your branch using `git checkout STUDENTID-Name`.
3. Install any dependencies as listed.

CI/CD Pipeline
The project utilizes GitHub Actions for continuous integration. The workflow is defined in `.github/workflows/ci.yml`.

Branching Strategy

Each member has their own branch named `STUDENTID-Name`. All changes are merged into the `main` branch via Pull Requests.

9. Creating a Pull Request Template

- **Responsibility:** Only the Leader creates the PR template.

Reference Steps:

1. **Create a `.github` Directory:**

   mkdir .github

2. **Create the PR Template File:**

   touch .github/PULL_REQUEST_TEMPLATE.md

3. **Add the Following Content to `PULL_REQUEST_TEMPLATE.md`:**

   ## Description

   Please include a summary of the changes and the related issue.

   ## Type of Change

   - [ ] Bug fix
   - [ ] New feature
   - [ ] Documentation update

   ## Checklist

   - [ ] My code follows the style guidelines of this project
   - [ ] I have performed a self-review of my code
   - [ ] I have commented my code, particularly in hard-to-understand areas
   - [ ] I have made corresponding changes to the documentation
   - [ ] My changes generate no new warnings
   - [ ] I have added tests that prove my fix is effective or that my feature works
   - [ ] New and existing unit tests pass locally with my changes

4. **Commit and Push the Template:**

   git add .github/PULL_REQUEST_TEMPLATE.md
   git commit -m "Add Pull Request template"
   git push origin main

   Reference: Creating a pull request template

10. Final Submission

Create separate PDF/DOCX file for each member and submit all file by one member. If any member had not contributed, please mention it while submitting your work.

- **Repository Link:** Ensure your GitHub repository is public and copy the link.
- **Screenshots:**
    - Capture screenshots showing:
        - Your local repository setup.
        - Git commit history.
        - Branch structure.
        - CI pipeline status.
- **Submission Folder:** Upload the **GitHub repository link** and **screenshots** to the assignment submission folder on D2L before the deadline.

Additional Tips

- **Commit Often:** Regular commits make it easier to track changes and resolve conflicts.
- **Clear Commit Messages:** Use descriptive messages to explain the purpose of each commit.
- **Branch Naming:** Stick to the naming convention to avoid confusion.
- **Regular Syncing:** Frequently pull changes from `main` to keep your branch updated.
- **Review PRs:** Before merging, review changes to ensure code quality and consistency.
- **Documentation:** Keep the `README.md` updated to reflect the current state of the project.
- **Late Submission:** Late submissions will not be accepted.

Support and Communication

- **Contact for Assistance:** Reach out to pritesh.patel2@georgebrown.ca or use the designated Slack channel for any questions or issues.
- **Collaboration:** Utilize collaborative tools like GitHub Issues or Slack to manage tasks and resolve queries within the group.

By following this guide, your group should be well-equipped to handle each component of the assignment effectively. Good luck and ensure consistent communication and collaboration within your group to achieve the best results!

Keep In mind

- ALL Commands given in the assignment documents are for reference. User your knowledge and expertise to use the commands.
- No commits or file changes/upload are allowed directly on GitHub website. All commits must be done on local machine of each member should merge their branch to main/master branch
- You can create PR/Merge code on GitHub website
- Each member will award marks based on their contributions to the assignment after verification on GitHub commits logs
- Late submission will be awarded ZERO

Good luck with your assignment!

COMP 3104 Group Assignment Rubric

Criteria	**Points**	**Level 1 (0-40%)**	**Level 2 (41-60%)**	**Level 3 (61-80%)**	**Level 4 (81-100%)**
Group Formation	**05**	Group not formed according to guidelines (less than 3 or more than 4 members).	Group formed but required assistance or clarification.	Group correctly formed but with minor errors.	Group correctly formed with 3-4 members as per guidelines.
GitHub Repository Creation	**05**	Repository not created, or multiple repositories created by the group.	Repository created but with errors in naming or visibility.	Correct repository created with minor errors in configuration.	Correct repository created with appropriate naming, public visibility, and proper configuration.
Adding Collaborators	**05**	Collaborators not added or incorrectly added with	Collaborators added but with incorrect permissions.	Collaborators added correctly with minor errors in	All collaborators correctly added with

Criteria	Points	Level 1 (0-40%)	Level 2 (41-60%)	Level 3 (61-80%)	Level 4 (81-100%)
		improper permissions.		permission settings.	appropriate permissions.
Branch Naming and Creation	**05**	Branches not created or named incorrectly.	Branches created but with significant errors in naming conventions.	Branches correctly created but with minor errors in naming conventions.	All branches correctly created following the STUDENTID-Name format.
CI/CD Integration	**10**	CI pipeline not set up or incorrectly configured.	CI pipeline set up but with major issues.	CI pipeline set up correctly but with minor issues or incorrect triggers.	CI pipeline set up correctly with appropriate triggers and configurations (e.g., GitHub Actions).
Commits and File Creation	**40**	Less than 10 commits, incorrect file naming, or missing files.	10 commits made but with generic messages; files created but with minor errors.	10 commits made with mostly appropriate messages; files correctly named and containing appropriate content.	10+ commits made with clear, meaningful messages; all files correctly named and containing relevant, well-written content.
Merge and Conflict Resolution	**10**	Merge not performed or unresolved conflicts remain in the repository.	Merge performed but with unresolved conflicts or improper conflict resolution.	Merge performed with minor issues in conflict resolution.	All branches successfully merged into main, with conflicts resolved correctly and documented if necessary.
Readme.md Update	**10**	README.md not	README.md updated	README.md updated	README.md fully updated

Criteria	Points	Level 1 (0-40%)	Level 2 (41-60%)	Level 3 (61-80%)	Level 4 (81-100%)
		updated or lacks essential information.	but missing key details or clarity.	with most required information but lacks detail or accuracy in some areas.	with group details, project description, setup instructions, CI/CD information, and branching strategy clearly documented.
Pull Request Template	**05**	Pull request template not created or incorrectly implemented.	Pull request template created but with significant omissions or errors.	Pull request template created with minor issues or missing details.	Pull request template correctly implemented with all necessary sections and instructions.
Final Submission (GitHub Link & Screenshots)	**05**	Submission not provided or incomplete (missing screenshots, broken links, etc.).	Submission provided but with missing or incorrect details (e.g., incorrect link, incomplete screenshots).	Submission provided with most required elements but missing minor details.	Correct GitHub link and complete set of screenshots provided, demonstrating the entire workflow.

Total Points: 100

Project Description:

This is a group assignment effort will be lead by Team Leader wherein you will be exploring the concepts behind version control, specifically with git and GitHub. In addition to this, you will also be implementing CI/CD (Continuous

Itegration/Continuous Deployment) principles by way of GitHub action or Travis CI; This will be so we can monitor the health of your GitHub deployment and ensure it stays online an operation as intended.

After making several (10) commits, you must then merge all member branch into the main branch and continually pull updates to our local branches. Lastly, each group member will update the communal README.md file (like so), and then your team lead will create the pull request template prior to submission.

Such an assignment will do, and has done, well to solidify the concepts and methods behind GitHub and version control at large but within a simulated and relatively safe environment. You will get the exposure to merge conflicts, file structure navigation, and implementing branching strategy to name a few.

Instructions:

Step 1: Clone the repository to your local machine.
command: ***git clone <Your Group Assignment Repo link>***

Step 2: Create/switch to your own branch.
command: git checkout -b your-new-branch-name

Step 3: Push your new branch to the remote repository.
command: git push -u origin 'your-new-branch-name'

CI/CD Details:

This project is utilizing the Travis CI tool for continuous integration. All branches of this repository have also been set up successfully through Travis. An example of the files/packages needed to do this can be found in the ci-files directory.

Branching Strategy:

The purpose of a branching strategy is threefold: - Protect the main branch (the deployed product) from errors, vulnerabilities or code that has not been quality checked and peer reviewer - Enable developers to work on / introduce seperate features or aspects into a project without having to frequently update and / or pull everything everyone is working on at the same time which would introduce breaking bugs throughout the entire process - Enable peer review, QA checks, and safety checks through pull requests.

Your branching stategy was to use a 'per-developer' branch as opposed to a 'per feature'. This means that every developer involve had their own branch to commit their own code.

At this point, we left out branches open, although they can be closed if needed. We determined that there was no sense in closing branches that could potentially be used again, and would therefore be recreated, wasting development time.