

Lumen Workflow and Submission

Fall 2026

What Lumen Is

This course uses **Lumen** for assignment management. When an assignment is released, Lumen creates a private GitHub repository for each of you from the instructor's template. There is no *Accept assignment* link. From that point on, the repository is a normal GitHub repository that you clone, edit, commit, and push to like any other.

Your repository starts with a single branch, `main`. You create a `development` branch and do all of your work there, as described in the Assignment Guidelines.

The Two Pull Requests

This workflow uses two pull requests that travel in opposite directions. They are easy to confuse, so this document names them and always uses those names. **Read the table before you click any green button on GitHub.**

	Direction	Who opens it	Who merges it
Update PR	instructor <code>main</code> → your <code>main</code>	Instructor	You
Student PR	your <code>development</code> → your <code>main</code>	You	Instructor

The short version: **you merge the Update PR; you never merge the Student PR.** Opening the **Student PR** is your submission. Merging it is part of grading, and it is the instructor's job.

Part 1: Handling an Update PR

An **Update PR** appears only when the instructor has to change an assignment after release, usually a correction or a clarification. This is **rare**. If you never see one for an assignment, that is normal and there is nothing to do.

When one does appear you will get a normal GitHub notification.

Step 1: Merge the Update PR into main

1. Go to your assignment repository on GitHub.
2. Click the **Pull requests** tab.
3. Open the PR from the instructor. Confirm it is merging *into* your `main` branch; that is what makes it the **Update PR**.
4. Click **Merge pull request**, then **Confirm merge**.

Your `main` branch now has the instructor's changes. Your `development` branch does not; branches do not update each other automatically. Continue to Step 2.

Step 2: Bring the Update into development

Run these commands from inside your repository directory:

```
git checkout main
git pull origin main

git checkout development
git merge main
git push origin development
```

Do not skip the last line. Without it the update exists only on your machine, and the instructor cannot see it.

Step 3: Verify

```
git status
git log --oneline -5
```

You should see the instructor's most recent commit in `git log`, and `git status` should report a clean working tree and that your branch is up to date with `origin/development`.

If You Get a Merge Conflict

A conflict means you and the instructor changed the same lines of the same file. This is uncommon but not an error, and it is most likely in a starter file you have been editing, which is exactly the file an update is most likely to touch.

Resolve it the way you would any conflict: open the marked files, keep the instructor's change while preserving your own work, then `git add` the resolved files and `git commit`. **If the conflict is in a file you did not write and do not understand, stop and ask the instructor** rather than guessing; deleting a piece of an updated assignment specification is worse than a late question.

Why not the GitHub website “Sync fork” button?

Lumen creates your repository as a fresh copy from a template, not as a fork. GitHub's **Sync fork** button will not appear on your assignment repositories, and **Contribute** does not apply. Ignore both.

Part 2: Submitting Your Work

Opening the Student PR is how you submit. There is no separate submit button, and pushing commits by itself does not submit anything. Until the **Student PR** exists, the assignment has not been handed in.

Step 1: Push your finished work

```
git checkout development
git status
git push origin development
```

`git status` must report a clean working tree. Anything uncommitted is not submitted.

Step 2: Open the Student PR

1. Go to your repository on GitHub and click the **Pull requests** tab, then **New pull request**.
2. Set **base** to `main` and **compare** to `development`. Check this carefully: backwards is the one mistake that is hard to undo.
3. Title it with the assignment name (for example, HW3 Submission).
4. Click **Create pull request**.

Step 3: Stop

Do not merge the Student PR, and do not delete your `development` branch. Both are part of grading.

Do not push further commits to `development` after opening the **Student PR**. Opening it tells the instructor the assignment is finished and ready to grade; work that appears afterward is not guaranteed to be read. If you realize you submitted too early, close the **Student PR**, finish the work, and open a new one; the new PR's timestamp is then your submission time, including for late-penalty purposes.

If you accidentally merge your own **Student PR**, do not try to unwind it. Tell the instructor.

What Happens After You Submit

You do not need to do anything during this. For reference, the instructor will:

1. Grade the work on your `development` branch, using the **Student PR** to review it.
2. Leave feedback as comments on the **Student PR**.
3. Merge the **Student PR** into `main`.
4. Commit your grade into `README.md` on `main`.
5. Delete the `development` branch.

When this is finished your repository has a single branch, `main`, holding your completed work and your grade. The **Student PR** page remains available with the full diff and all feedback comments even after the branch is deleted, so that is where to look for comments on your work.

Grades are also posted to the course LMS. The `README.md` entry is a convenience copy; the LMS gradebook is authoritative.

Deadlines and Late Work

Your submission time is **when the Student PR is opened**, not when you last pushed. A branch full of finished work with no **Student PR** has not been submitted.

The homework late policy in the syllabus applies to that timestamp:

- A **Student PR** opened after the deadline loses **10% per day late**, including weekends and university holidays.
- More than **three days** late receives a score of **0**.
- No **Student PR** at all is a score of **0**.

See the syllabus for the authoritative policy, including how paper reviews and project milestones differ.

Summary

- Work on development. Commit and push often.
- **Update PR** (rare, from the instructor): **you merge it** into `main`, then merge `main` into development and push.
- **Student PR** (from you, when finished): development into `main`. Opening it **is** your submission.
- **Never merge your own Student PR**. The instructor merges it as part of grading.
- Late clock runs on the **Student PR** timestamp: 10% per day, zero after three days.