AI CLASS · ASSIGNMENT 01

Get Yourself Online

Friday, August 14, 2026 · about 90 minutes · interactive version at `aiclass.oaksedtech.com`

By next week you can take a change from idea to live on the internet without looking anything up. That's the whole job this week — learn the machinery.

What you put live becomes your portfolio, and you'll add your work to it all year. That part comes later, and only once the machinery works.

> **Nothing here explains how. Finding out is the assignment.** Use AI, videos, docs, each other — there's no cheating on the how. You're judged on whether it works, whether you can explain it, and how you got yourself unstuck.

Hand in

1	Your GitHub profile
2	A pull request you opened and merged
3	Your live URL
4	What broke, and how you got unstuck

What you need

1 — A GitHub account

> **DONE WHEN**
> Student Pack application submitted, and your profile has your real name and a photo.

2 — The Git loop

> **DONE WHEN**
> You can go branch → commit → push → PR → merge → pull on a real repo, without notes.

3 — A Codespace

> **DONE WHEN**
> You've launched one, used its terminal, and stopped it when you were finished.

4 — Something live

Anything. A heading will do. This week it only has to exist at a URL.

> **DONE WHEN**
> A stranger can open it on their phone and see something you made.

Limits

- **GitHub Pages needs a public repo** on the free plan. Other hosts don't — worth knowing before you assume you're stuck.
- Codespaces bills **core-hours, not clock hours** — a 2-core machine burns 2 per real hour.
- Your profile needs to look like a real person for the Student Pack. Nothing beyond that belongs in a repo — not your address, not your schedule, not your phone number.

Secrets

Everything you commit is permanent, and on a public repo it's instantly public. Bots scrape new commits within seconds of a push. API keys, tokens, passwords, `.env` files: never.

Know what a `.gitignore` is and have one working **before** your first push. And find out what you're supposed to do if you commit a key by accident — the answer is not "delete it in the next commit," and understanding why not is the whole lesson.

When you're stuck

That's not the assignment going wrong. That's the assignment.

Before you ask a person: search the exact error text, read the real documentation, ask an AI to explain it three different ways until one lands, find a video. Note where the answer came from — hand-in 4 is that note.

PROMPT — BUILD YOUR OWN WAY IN

```
I need to learn X. Don't explain it yet.

Find me the three best resources — the official docs page, one video, one worked example.
Tell me what order to use them in and what I should be able to do after each. Then quiz me
on it and tell me what I still don't have.
```

Vocabulary — know these cold

Term	What it means	Like
Repository	One project's folder plus its entire history	A Doc where every version is saved
Commit	One saved snapshot, with a message explaining why	Hitting save and writing a note
Push	Upload your commits to GitHub	Syncing your photos to the cloud
Pull	Download commits other people made	Refreshing for everyone's latest
Clone	Make your own local copy of a repo	Downloading the whole album
Fork	Copy someone else's repo into your account	Making your own remix
Branch	A parallel timeline where you can experiment safely	A "what if" copy

Merge	Fold a branch's changes back into main	Accepting the "what if"
Pull Request	A formal request to merge, with room for review	"Look this over first"
main	The default branch. The real version	The master copy
Deploy	Put the code on a live public URL	Printing the book
Worktree	Two branches checked out in two folders at once	Two desks, one filing cabinet

Quiz yourself

Answer out loud before you check. Answers at the bottom.

1. Git vs GitHub?
2. Commit vs push?
3. Why branch instead of editing main?
4. What's a pull request for, if you could just merge?
5. Why isn't pushing the same as deploying?
6. You need a secret API key for a weather service. Where does it go, and why?

PROMPT — MAKE IT QUIZ YOU PROPERLY

```
Quiz me on: Git vs GitHub, the commit/push/pull/branch/merge/PR workflow, worktrees,
deploying, Codespaces core-hours, installing an AI coding agent, and the three layers of the
web stack.

One at a time, wait for my answer. Mix definitions, "what happens if...", and "here's an
error, what did I do wrong." Don't accept vague answers. Score me after 12 and tell me what
to review.
```

If you have time

Start the portfolio

That live page is going to become your portfolio. Nobody's handing you a template — working out what makes one good, and what makes yours worth looking at, is the assignment.

> **DONE WHEN**
> It's live, something of yours is on it, and you can say why it looks the way it does.

Get an agent in your terminal

Only if you already have a paid Claude or ChatGPT plan — both agents need one, and neither free tier includes them.

Get it running **inside your Codespace**, not on your laptop. Things to look up, in order:

- The official install command for Claude Code, or for the Codex CLI
- How to authenticate it from inside a container. This is the awkward part. The first link you're offered won't work, and working out why is the exercise
- How to get it to read your whole project before you ask it for anything

- How to stop your login vanishing every time you create a new Codespace

DONE WHEN
It has explained your own repo back to you in plain English, and you've had it change something that you then reviewed line by line.

Answers

1. Git is the program taking snapshots on your computer. GitHub hosts copies so others can collaborate. You can use Git with no GitHub at all.

2. A commit saves locally. A push uploads to GitHub. You can commit ten times and push once.

3. `main` should always work. A branch lets you break things without risking the live version — if it fails, delete it and nothing was ever at risk.

4. Review, and the record of why a change happened, kept with the code.

5. Pushing stores your code. Deploying serves it at a public URL.

6. On the server, behind an API layer. Anything shipped to the browser is public — the frontend can be read by anyone with Ctrl/Cmd+U, so a key there is a key given away.

The interactive version — Git sandbox, flashcards, self-checks — is at `aiclass.oaksedtech.com`. This sheet is the same assignment on paper.