

The B4LCILC Capstone

Build one real thing. Show it to the world.

August 2026

The B4LCILC Capstone

Finish this and you stop being someone who studied cloud. You become someone who has built on it — with a link to prove it.

Certificates say you sat through something. A working system with a public address says you can do the job. Every hiring manager knows the difference, and this is the thing they ask to see.

How this works — read this part carefully

This document gives you objectives, not instructions. Each step tells you *what* must be true when you're done. It does not tell you *how* to do it. That is deliberate, and it is the whole point.

Being stuck, searching, reading documentation, trying something that fails, and working out why — that *is* the job. An engineer who has only ever followed step-by-step instructions has practised following instructions. An engineer who has fought through an error message at 11pm has practised engineering. This capstone is designed to make you do the second one.

So when you get stuck — and you will, repeatedly — that is not the capstone going wrong. That is the capstone working.

Rules:

- Use **either AWS or Azure**. Both is better, but one is enough. The steps are written to work on both.
- Everything must stay inside a free tier. Set a spend alarm *before* you start. If you don't have a cloud account, read our *Free Real Cloud Practice* guide — there are ways to do this with no credit card.
- Take a screenshot when something breaks. You will want it for step 12.
- There is no model answer. Nobody is going to hand you the solution, because in the job nobody will either.

Roughly how long: two to six weeks of evenings for most people. Faster is not better here.

Part One — Put something real on the internet

Step 1 — Own your identity

Register a domain name that is yours. Your name is ideal. It will cost roughly \$10–15 a year and it is the only part of this capstone that costs money — if that is not possible for you, use the free subdomain your hosting provides and note it in your write-up.

Done when: typing your domain into a browser reaches something you control.

Step 2 — Build the site

A single page is fine. It should be your CV: who you are, what you can do, what you have built. Write the HTML yourself — no site builders, no templates you don't understand. You must be able to explain every line.

Done when: the page opens correctly on both a phone and a laptop.

Step 3 — Host it on object storage

Your site must be served from cloud object storage — **S3** on AWS, **Blob Storage** on Azure — configured for static website hosting. Not a web server. Not a hosting company. Object storage.

Done when: the site loads from the storage endpoint URL.

Step 4 — Put it behind a CDN, over HTTPS

Serve it through a content delivery network — **CloudFront** or **Azure Front Door** — with a valid TLS certificate, so it loads over `https://` with no browser warning, and your custom domain points at it.

Done when: `https://yourdomain.com` loads with a padlock, and you can explain what the CDN is actually doing for you.

Checkpoint. You now have a real production website on real cloud infrastructure. Most people who “learn cloud” never get this far.

Part Two — Make it do something

Step 5 — Add a visitor counter

Your page must display how many times it has been viewed. The count must survive a refresh and be the same for everyone.

Done when: the number goes up when someone else visits, not just you.

Step 6 — Store the count in a database

That number lives in a cloud database — **DynamoDB** or **Cosmos DB**. Not in a file, not in the browser.

Done when: you can see the record in the database console and watch it change.

Step 7 — Never let the browser touch the database

The webpage must not talk to the database directly. It calls an **API**, and the API talks to the database. Use a serverless function — **Lambda** or **Azure Functions** — behind **API Gateway** or an HTTP trigger.

Done when: the site works, and there are no database credentials anywhere in your webpage's code.

Why this step exists: putting credentials in front-end code is one of the most common serious security mistakes in the industry. Building it correctly once means you will never be tempted.

Step 8 — Lock the permissions down

The function must have permission to do exactly one thing — read and update that one counter — and nothing else. Not “full access.” Not an administrator role.

Done when: you can state, in one sentence, precisely what your function is permitted to do, and you have deleted every permission beyond that.

Part Three — Work like a professional

Step 9 — Write a test

At least one automated test that checks your API returns what it should. It must fail if you break the code.

Done when: you deliberately break the function, the test fails, you fix it, the test passes.

Step 10 — Define your infrastructure as code

Every cloud resource must be defined in code — **Terraform** or **OpenTofu** (both clouds), or **CloudFormation** / **Bicep**. Then prove it: destroy everything and rebuild it from your code alone.

Done when: you have deleted your entire stack and brought it back with one command. **This is the single most important step in the capstone.** Anyone can click buttons in a console. Being able to destroy and rebuild is what separates an engineer from a user.

Step 11 — Automate deployment

Set up a **CI/CD pipeline** — GitHub Actions is free and works with both clouds. When you push a change to your repository, the site updates by itself. No manual uploads, ever again.

Done when: you change one word, push it, and the live site updates without you touching the cloud console.

Part Four — Make it count

Step 12 — Write it up

A blog post, on your own site, explaining what you built and how. It must include:

- An architecture diagram you drew yourself
- **At least one thing that went badly wrong**, what the error actually said, and how you worked it out
- What you would do differently next time
- What this cost you per month

That failure section is not filler. It is the part hiring managers read most carefully, because it is the only part that proves you can debug rather than just follow.

Step 13 — Publish everything

Your code goes in a public **GitHub** repository with a README explaining how it works. Your write-up goes online. Both links go on your CV.

Done when: a stranger can find your work, read it, and understand what you did without asking you.

If you want to go further

Optional, and each one is a genuine talking point in an interview:

- **Build it on the other cloud too**, and write about which was easier and why. Very few candidates can speak credibly about both.
 - **Add monitoring and an alarm** — CloudWatch or Azure Monitor — that emails you if the API starts failing.
 - **Put a WAF in front of it** and write about what it blocks.
 - **Add a second region** and explain your failover approach honestly, including what your RTO and RPO actually are.
 - **Do a cost teardown:** what each component costs, what it would cost at 10,000 visitors a day, and where the first bottleneck appears.
-

Judging your own work

Before you call it finished, answer these out loud. If you cannot, you have found your next week of study:

1. What happens if the database is unavailable? What does the visitor see?
2. Where exactly does your TLS certificate come from, and what happens when it expires?
3. If someone found your GitHub repository, what could they do with it? What is protected, and by what?

4. Which single component, if it failed, takes your whole site down?
5. What does this cost you per month, and which line item would grow fastest under load?
6. If you had to hand this to another engineer tomorrow, what would they struggle with?

Ready to show an employer

Your finished capstone gives you three links:

- **the live site** — proof it works
- **the repository** — proof you can write and organise code
- **the write-up** — proof you can think, debug, and explain

Put all three at the top of your CV, above your education. Then, in an interview, when they ask “tell me about something you’ve built,” you will not be describing a course you took. You will be opening a browser.

Part of the free B4LCILC cloud courses. This capstone is inspired by the community-built project-first tradition of cloud learning, especially Forrest Brazeal’s Cloud Resume Challenge — with a dual-cloud, AWS-and-Azure structure of our own. Build it, publish it, and tell someone.