

The Cloud Leader Course

From zero knowledge to leading a team of cloud engineers — prepared for Fedly Bonneau

August 24, 2026

The Cloud Leader Course

From zero knowledge to fluently leading a team of cloud engineers.

How this course works

You are not training to *be* a cloud engineer. You are training to **lead** cloud engineers: to follow every conversation in the room, ask the questions that matter, make good decisions about money, risk, and people, and earn the respect of experts without pretending to be one. That is a different, entirely learnable skill — and it is the skill this course teaches.

The course has three stages. **Stage 1 (Weeks 1–8): Speak the language.** You learn what the cloud actually is and the vocabulary of every core building block, so meetings stop sounding like noise. **Stage 2 (Weeks 9–16): Think like the room.** Architecture, security, and money — the three conversations every cloud team has every week, and the three where a leader either adds value or gets ignored. **Stage 3 (Months 5–24): Lead.** Certifications, hiring, running decision meetings, and the honest two-year path to holding your own with senior engineers.

Rules for the whole course: study **one hour a day, six days a week** — consistency beats intensity. Every module ends with a **Fluency Drill** (things you say out loud until they're natural) and a **Milestone** (proof you're ready to move on). Do them; reading alone builds recognition, not fluency. And from Week 1, keep a **Decision Journal**: every time you learn a concept, write one sentence on how it affects money, risk, or people — because that translation is your entire job.

STAGE 0 — THE GROUND FLOOR (start here if technology is new to you)

These two modules are for the reader who has never worked in technology at all. They are the same as Modules 1 and 2 of the interactive course at b4lcilc.org. If you already know what a web server and a database are, skim them; if you do not, they are the most important pages in this book — everything after stands on them.

Module A: How the web works — what happens when you type an address

The big idea: A web page is not a picture beamed to your phone. It is a set of **written instructions** sent from a distant computer, which your own browser follows to build the page on your screen. Learn the six steps between typing an address and seeing a page, and every technology meeting for the rest of your career becomes easier to follow.

The journey, in six steps you should be able to say out loud:

1. You type a name — `b4lcilc.org` — into a **browser** (Chrome, Safari, Edge, Firefox).
2. **DNS**, the internet's phone book, turns that name into an **IP address** — the numeric address of a computer, like `142.250.66.14`.
3. Your browser — the **client**, the one who asks — sends an **HTTP request** to the computer at that address: "GET me this page."
4. The request travels as small **packets** under the rules of **TCP/IP**, the internet's plumbing, which numbers every packet, confirms delivery, and re-sends anything lost.
5. The distant computer — the **server**, the one who answers — sends back a **response**: the page's instructions, or a numbered excuse like **404** ("no such page").
6. Your browser follows those instructions and builds the page. The instructions come in three languages: **HTML** (structure — headings, paragraphs, buttons), **CSS** (appearance — colors, sizes, layout), and **JavaScript** (behavior — what happens when you click).

Client-side and server-side — the frame that organizes everything: work done on your device, in the browser, is *client-side* (the **front end**); work done on the distant computer is *server-side* (the **back end**). The rule of thumb worth memorising: **anything involving trust or memory belongs on the server; anything involving speed of feel belongs on the client**. Checking a password, saving a record, charging a card: server. Drawing the page, instant feedback as you type: client. Between the two often stands a **proxy** — a server in the middle that filters, shields, or spreads traffic; the "load balancers" and "CDNs" of later modules are proxies with specialities.

Why DNS matters to a leader: DNS fails in a special way — when it breaks, every site "disappears" at once even though every server is still running. The phone book is down, not the phones. When an engineer says "it might be DNS," that is what they mean, and some of the internet's most famous outages were exactly this.

And the padlock: HTTPS is HTTP with encryption — the whole conversation scrambled against anyone in between, which is what the padlock in the address bar certifies. The modern rule: every site, always.

Vocabulary:

Term	Definition
Web browser	The program that fetches pages and draws them: Chrome, Safari, Edge, Firefox.
URL	The full address you type: protocol + computer name + which document you want.

Term	Definition
IP address	A computer's numeric address on a network, like 142.250.66.14.
DNS	The internet's phone book: turns names (google.com) into IP addresses.
HTTP / HTTPS	The request-and-response rules browsers and servers speak; the S adds encryption (the padlock).
TCP/IP	The internet's plumbing: moves numbered packets reliably between addresses.
Client / server	The asker and the answerer. A relationship, not a type of machine.
Client-side (front end)	Work done in the user's browser: drawing, instant feedback.
Server-side (back end)	Work done on the distant machine: trust, rules, permanent records.
Proxy	A middleman server that handles traffic on one side's behalf.
HTML / CSS / JavaScript	The three page languages: structure / appearance / behavior.

Fluency drill: say the six-step journey out loud, alone, until it takes you under thirty seconds and you never reach for a word. Then say it to another person.

Milestone: you can answer, at dinner, with no preparation: "What actually happens when I type a website's name into my phone?" — as a story, not a list of jargon.

Module B: How websites are built and hosted — from one computer to the cloud

The big idea: Every real website is built in **three layers** — a front end people touch, a middle layer where the organization's rules run, and a data layer that never forgets — and for twenty-five years almost all of it has been assembled from one of two famous families of tools. Learn the three layers and the two families, and vendor meetings switch from a foreign language to a menu.

The restaurant, which you will use for the rest of this course: the **front end (UI)** is the dining room — everything the customer sees, built in HTML, CSS and JavaScript. The **middle tier (business logic)** is the kitchen — code on the server applying the house rules: "a student may not enroll twice," "orders over the limit need approval." The **data layer** is the pantry and the recipe book — the **database**, where everything that must not be forgotten is kept. The industry calls this shape **N-tier architecture**. Why layers? So each can be renovated without rebuilding the others — and when an engineer says "the bottleneck is in the data layer," they are handing you this exact map with a circle on it.

The machine and the program: a **server** is a real computer — a screenless metal box racked with hundreds of others in a **data center**, built to run every second of the year. On it runs a *program* also called a web server, which conducts the HTTP conversation. Know the two most famous by name: **IIS** (Microsoft's, runs on Windows) and **Apache** (free, runs mostly on Linux). Keep the words straight: *a browser asks; a web server answers*. Apache is a web **server** — the answering side — not a browser. **Web hosting** is renting space on someone else's server so their machine answers under your name.

The database and SQL, gently: a **database** is a program that stores records and answers questions about them — a fanatically organized filing clerk who never sleeps. **SQL** (“sequel”) is simply **the language for asking a database questions**. A *query* is a question. Read this line — you already can: `SELECT name FROM students WHERE village = 'Mae La'` — *from the students table, give me the names of everyone from Mae La*. Two brands dominate: **SQL Server** (Microsoft's, licensed) and **MySQL** (free, open source). Both speak SQL. The leadership point: **the database is where the organization actually lives** — pages can be rebuilt, but lose the enrollment records and there is no school. That is why databases get the most protective engineering in the building.

The two families:

Job in the machine	Microsoft family	Open-source LAMP family
Operating system	Windows Server	Linux
Web server (answers HTTP)	IIS	Apache
Business-logic language	C# on .NET	PHP
Database	SQL Server	MySQL
Cost model	License fees, one vendor	Free and open source; you pay for people
Classic home turf	Banks, governments, enterprise	The open web — WordPress alone runs a third of all websites

Open source means the software's recipe is public — anyone may inspect, improve and use it without license fees. It is why a scrappy organization could always build a world-class website for the price of one rented machine.

The road to the cloud — the sentence this whole book stands on: for decades all of this ran **on-premises**: in a server room the organization owned, bought for the busiest imaginable day, idle every other day, kept cold at the organization's expense. Then Amazon and Microsoft offered to rent capacity from their own warehouses by the hour. The stacks did not change — the same Linux, Apache, IIS, SQL Server run in the cloud today. **What changed is where the machines live and who owns them**. Hold that sentence: Module 1 of Stage 1 opens exactly there.

Vocabulary:

Term	Definition
Server / data center	The always-on machine, and the building full

Term	Definition
	of racks of them.
Web server (program)	Software that answers HTTP requests: IIS (Microsoft), Apache (open source).
Web hosting	Renting space on someone else's server for your website.
N-tier architecture	Building in layers: front end (UI), middle tier (logic), data layer.
Database	The program that stores records and answers questions — where the organization lives.
SQL / query	The near-English language for asking a database questions; a query is one question.
SQL Server / MySQL	Microsoft's licensed database / the open-source favorite. Both speak SQL.
Microsoft stack	Windows Server + IIS + C#/.NET + SQL Server.
LAMP stack	Linux + Apache + MySQL + PHP — all open source.
Open source	Public recipe: free to inspect, improve, and use.
On-premises	Machines you own, in a room you own — the world before the cloud.

Fluency drill: unpack LAMP on four fingers, out loud, three days running. Then explain the restaurant — dining room, kitchen, pantry — to someone who has never heard of a server.

Milestone: a contractor says: “You’re on an old on-prem Microsoft setup — IIS and SQL Server. We’re a LAMP shop, and we’d move you to the cloud.” You can translate every word for your board — and you know the risky part to probe is the database migration.

STAGE 1 — SPEAK THE LANGUAGE (Weeks 1-8)

Module 1 (Weeks 1-2): What the cloud actually is

The big idea: The cloud is *someone else's computers, rented by the hour, managed by software*. Before cloud, a company bought physical servers, put them in a room, and paid people to maintain them — slow, expensive, inflexible. AWS, Microsoft Azure, and Google Cloud built warehouses of millions of computers (data centers) and let anyone rent slices of them per second, from anywhere, through a webpage or a line of code. That's it. Everything else in this course is detail on top of that one idea.

The big three providers, defined:

Provider	What it is
AWS (Amazon Web Services)	Amazon's cloud division and the world's largest cloud provider (~30% of the global market). Started in 2006 when Amazon began renting out the computing systems it had built for its own store. Today it offers 200+ services — servers, storage, databases, AI, and more — rented by the second. When this course says “the cloud,” AWS is the default example, and it opened its own Thailand Region (Bangkok) in January 2025.
Microsoft Azure	Microsoft's cloud, #2 worldwide and the strongest inside large enterprises because it connects naturally to the Microsoft tools companies already use (Windows, Office, Active Directory). Currently building its first Thailand datacenter region.
Google Cloud (GCP)	Google's cloud, #3 — strongest in data analytics and AI tooling. Committed \$1 billion to a Thai data center and Bangkok cloud region.

Your free AWS account — set this up in Week 1. Go to <https://aws.amazon.com/free> and click “Create a Free Account” (the direct sign-up page is https://signin.aws.amazon.com/signup?request_type=register). You'll need an email address, a phone number, and a credit/debit card for identity verification — the **Free Tier** gives you a monthly allowance of the basic services free (including 750 hours/month of a small EC2 server your first year), and this course's exercises stay inside it. Two safety habits from day one: turn on MFA (Module 6 explains it) and set a **billing alert** at \$5 (Module 7 explains how) so you can never be surprised. After sign-up you log in at <https://console.aws.amazon.com> — the “console” is simply AWS's control webpage.

Why companies use the cloud: speed (a new server in 60 seconds instead of 6 weeks), elasticity (rent 100 servers for the big sale day, give back 90 the day after), no upfront cost (operating expense instead of capital expense), and global reach (put your app near your customers in Bangkok, Tokyo, and Frankfurt without building anything).

The three service layers — the most useful mental model in cloud:

Layer	What you rent	Kitchen analogy	Example
IaaS (Infrastructure as a Service)	Raw computers, storage, networks — you manage everything on them	Renting an empty kitchen: you bring chefs, recipes, ingredients	Amazon EC2, Azure Virtual Machines
PaaS (Platform as a Service)	A managed platform — you bring only	Renting a staffed kitchen: you bring	AWS Elastic Beanstalk, Azure App

Layer	What you rent	Kitchen analogy	Example
	your application	the recipe	Service
SaaS (Software as a Service)	Finished software	Ordering from a restaurant	Gmail, Salesforce, Canva

Regions and availability zones: a **Region** is a geographic cluster of data centers (Thailand has its own AWS Region as of January 2025, in and around Bangkok). An **Availability Zone (AZ)** is one or more isolated data centers inside a region. Apps that matter run in at least two AZs, so one building's failure doesn't take them down. When engineers say "we're multi-AZ," they mean "one data center can burn down and we stay up."

Vocabulary:

Term	Definition
Cloud provider / hyperscaler	A company that owns huge data centers and rents computing over the internet (AWS, Azure, Google Cloud). "Hyperscaler" = the biggest few, built to near-unlimited scale.
Data center	A secured warehouse full of thousands of servers with industrial power and cooling — the physical place "the cloud" actually lives.
Region	A geographic cluster of data centers offered as one location choice, e.g., "Asia Pacific (Bangkok)." You choose the region your systems run in.
Availability Zone (AZ)	One or more isolated data centers inside a region, with independent power and networking. Running in two AZs means one building can fail and you stay online.
On-premises ("on-prem")	The old way: servers your company owns, in your own building. The opposite of cloud.
Migration	The project of moving systems from on-prem into the cloud.
Workload	Any application or system that runs — "the payroll workload," "the website workload." Handy word: it covers anything.
Provision	To create/set up a cloud resource (a server, a database). "Provision a server" = bring one into existence.
Scale up / scale out	Handle more demand by making a machine bigger (up) or adding more machines (out). Cloud favors <i>out</i> .
Latency	Delay before data arrives, measured in milliseconds. Distance creates latency — the

Term	Definition
	reason a Bangkok region matters to Thai users.
Console	The provider's web control panel where you see and manage everything you're renting.

Videos for this module (all free, links verified):

Video	Channel	Length	Link
What is AWS?	Amazon Web Services (official)	~2 min	https://www.youtube.com/watch?v=a9__D53WsUs
Top 50+ AWS Services Explained in 10 Minutes	Fireship	~10 min	https://www.youtube.com/watch?v=JlBYCM48to
What is Microsoft Azure? An Introduction	Eye on Tech	~3 min	https://www.youtube.com/watch?v=l9JkLhvaKA8
รู้จัก AWS Cloud คืออะไร (Thai-language intro)	Aware Corporation	short	https://www.youtube.com/watch?v=nrSpZKGxXd0

Fluency drill — say these until natural: “Is that workload on-prem or in the cloud?” · “Which region are we in — and are we multi-AZ?” · “Is this an IaaS approach or is there a managed service that removes the maintenance?”

Exercises: (1) Create your free AWS account at <https://aws.amazon.com/free> — the sign-up flow itself teaches you more vocabulary than a chapter of reading. (2) Watch the four videos in the table above (under 20 minutes total). (3) In your journal: write the elevator explanation of cloud you'd give a Thai school director in one breath.

Milestone: you can explain IaaS vs PaaS vs SaaS with your own analogy, and explain why AWS opening a Bangkok region mattered to Thai banks (answer: latency + data residency — see Module 6).

Module 2 (Weeks 3–4): The building blocks — compute, storage, and databases

Compute — the engines that run code. A **virtual machine (VM/instance)** is a slice of a physical server that behaves like a whole computer; you pick a size (CPU/RAM) and pay per second it runs. A **container** is a lighter, faster way to package one application so it runs identically anywhere; **Docker** packages them and **Kubernetes** (K8s) orchestrates fleets of them — when you hear “K8s,” think “the system that runs and heals our hundreds of

containers automatically.” **Serverless** (AWS Lambda) means you upload only a function of code; the cloud runs it when triggered and you pay per invocation — no servers to manage at all. The pattern to notice: VM → container → serverless is a slider from “more control, more maintenance” to “less control, near-zero maintenance.” Good teams pick per workload, not by fashion.

Storage — three shapes. Object storage (Amazon S3): a bottomless bucket for files — images, videos, backups, datasets; cheap, eleven-nines durable, the default answer to “where do we put files?” **Block storage** (EBS): the virtual hard disk attached to a VM. **File storage** (EFS): a shared drive many machines mount at once. Then **tiers**: hot (frequent access, costly) vs cold/archive (Glacier — cheap, slow) — moving old data to cold tiers is one of the easiest cost wins any team can ship.

Databases — two families. Relational/SQL (MySQL, PostgreSQL; managed as Amazon RDS/Aurora): data in tables with strict structure; the default for money, orders, users — anything where correctness is sacred. **NoSQL** (DynamoDB, MongoDB): flexible, massively scalable; the default for huge, fast, simpler-shaped data (sessions, catalogs, feeds). “Managed database” means the provider handles backups, patches, and failover — teams should have a strong reason to run their own.

Vocabulary:

Term	Definition
Instance	One rented virtual machine (VM). “Spin up an instance” = start a server.
Instance type / size	The spec you chose for it — how many CPUs, how much memory. Bigger type = higher hourly price.
Container	A lightweight package holding one application plus everything it needs, so it runs identically on any machine. Faster and cheaper than a full VM.
Docker	The standard tool for building and running containers.
Kubernetes (K8s)	The orchestrator that runs and heals fleets of containers automatically — restarts crashed ones, adds more under load. “K8s” is the industry nickname.
Cluster / node	A cluster is a group of machines working as one system; each machine in it is a node.
Serverless	Running code without managing any server at all — the cloud runs your function when triggered and bills per run.
Lambda / function	AWS’s serverless product; a “function” is the small piece of code it runs.

Term	Definition
S3 / bucket	AWS's object storage for files; a bucket is one named container of files. The default answer to "where do we put files?"
EBS volume	The virtual hard disk attached to an instance.
Durability	The chance stored data survives. S3's "eleven nines" (99.999999999%) means loss is essentially never.
Storage tier	Price/speed class for data: hot (instant, costly) – cold/archive (Glacier — cheap, minutes-to-hours to retrieve).
RDS	AWS's managed relational-database service — AWS handles backups, patches, and failover for you.
SQL vs NoSQL	SQL: strict tables, perfect for money and orders. NoSQL: flexible and massively scalable, for sessions, catalogs, feeds.
Backup / snapshot	A saved copy of data (a snapshot is a point-in-time copy of a disk or database) you can restore from.
Failover	Automatic switch to a standby copy when the primary fails — why managed databases survive bad nights.

Videos for this module:

Video	Channel	Length	Link
Getting Started with EC2	AWS Developers (official)	27 min	https://www.youtube.com/watch?v=nJ-djerESW0
Introduction to Amazon S3	Amazon Web Services (official)	~5 min	https://www.youtube.com/watch?v=ecv-19sYL3w
Docker in 100 Seconds	Fireship	2 min	https://www.youtube.com/watch?v=Gjnup-PuquQ
Kubernetes explained in 15 mins	TechWorld with Nana	~16 min	https://www.youtube.com/watch?v=VnvRFRk_5lk

Video	Channel	Length	Link
Serverless Computing in 100 Seconds	Fireship	~2 min	https://www.youtube.com/watch?v=W_VV2Fx32_Y

Fluency drill: “Should this run on VMs, containers, or serverless — and what’s the operational cost of each choice?” · “Is that data hot or can it go to a cheaper tier?” · “Why are we self-managing that database instead of using RDS?”

Exercises: (1) In your free account: launch the smallest EC2 instance, then terminate it. Upload a file to S3. You have now personally used IaaS. (2) Ask an AI assistant to quiz you: “Give me 10 scenarios; I’ll answer VM, container, or serverless, and you grade me.”

Milestone: given any simple app described in one sentence (“a website where students order lunch”), you can name its compute, storage, and database pieces out loud in 60 seconds.

Module 3 (Weeks 5–6): Networks, and how the pieces talk

The VPC — your private neighborhood. A **Virtual Private Cloud** is your own fenced-off section of the provider’s network. Inside it are **subnets** — public ones (reachable from the internet, e.g., web servers) and private ones (unreachable, e.g., databases). The single most common security sentence you’ll hear: “*the database sits in a private subnet.*” If you understand why — attackers can’t touch what has no path from the internet — you understand half of network security.

The traffic layer: a **load balancer** spreads incoming traffic across multiple servers (and quietly removes the sick ones); **DNS** (Route 53) translates names like yourcompany.com into addresses; a **CDN** (CloudFront) caches your content in hundreds of cities so it loads fast everywhere; an **API** is the doorway one piece of software offers another — when engineers say “we’ll expose an API,” they mean “we’ll give other software a controlled way to use ours”; an **API gateway** is the front desk that manages those doorways.

Connecting worlds: **VPN** (encrypted tunnel over the internet) or **Direct Connect** (a private physical line) links a company’s offices/on-prem systems to its cloud. **Hybrid cloud** = running both on-prem and cloud (most Thai enterprises); **multi-cloud** = using more than one provider.

Vocabulary:

Term	Definition
VPC (Virtual Private Cloud)	Your own private, fenced-off section of the provider’s network where your systems live.
Subnet (public / private)	A subdivision of the VPC. Public subnets are reachable from the internet (web servers); private subnets are not (databases).
IP address	The numeric address of a machine on a network — how computers find each other.
Firewall / security group	The rules list saying which traffic may reach

Term	Definition
Load balancer	a machine (“web traffic in, nothing else”). A security group is AWS’s per-server firewall. The traffic director that spreads incoming requests across multiple servers and stops sending to unhealthy ones.
DNS	The internet’s phone book — translates yourcompany.com into an IP address. AWS’s DNS service is Route 53.
CDN (Content Delivery Network)	Copies of your content cached in hundreds of cities so it loads fast everywhere. AWS’s is CloudFront.
Edge location	One of those city-level CDN points — “the edge” means close to users.
API	The doorway one piece of software offers another. “We’ll expose an API” = we’ll give other software a controlled way to use ours.
API gateway	The front desk managing those doorways — authentication, rate limits, logging.
VPN	An encrypted tunnel over the public internet linking two networks (e.g., your office to your VPC).
Direct Connect	A private physical line to the cloud — faster and steadier than VPN, at a price.
Hybrid / multi-cloud	Hybrid: running on-prem and cloud together (most Thai enterprises). Multi-cloud: using more than one provider.
Ingress / egress	Traffic entering / leaving the cloud. Egress costs money — remember this word for Module 7.

Video for this module: AWS Networking Basics — VPC & Subnets for Beginners — KodeKloud — https://www.youtube.com/watch?v=QM63dyA_4Pc (watch the first half now; return for the rest after Module 5).

Fluency drill: “Is the database in a private subnet?” · “What happens when one web server dies — is the load balancer health-checking?” · “Are we exposing that as an API or is it internal only?”

Exercises: (1) Have an AI walk you through drawing, on paper, the network of a food-delivery app: users → CDN → load balancer → web servers (public subnet) → database (private subnet). Draw it three times until you can do it from memory. (2) Find your company’s or any sample architecture diagram and circle every box you can now name.

Milestone: you can sketch that standard three-tier diagram on a whiteboard and narrate the path of one customer's click through it.

Module 4 (Weeks 7–8): How cloud teams work — DevOps, IaC, and reading the room

DevOps is the culture of merging “the people who write software” (Dev) and “the people who run it” (Ops) into one team that ships small changes frequently and safely, with automation doing the heavy lifting. Its heartbeat is the **CI/CD pipeline**: every code change is automatically built, tested, and deployed (Continuous Integration / Continuous Delivery). When a team says “it’s in the pipeline,” they mean the robot is testing and shipping it.

Infrastructure as Code (IaC) — the idea that changed everything: instead of clicking around a console to create servers, engineers write *text files* that declare the infrastructure (“two servers, one load balancer, one database, these firewall rules”), and a tool (**Terraform**, CloudFormation) makes reality match the file. Why leaders care: the files live in **Git** (version control), so every infrastructure change is reviewed, reversible, and auditable — the difference between a workshop and a factory.

The rituals you’ll sit in: **stand-up** (daily 15 minutes: what’s done, what’s next, what’s blocked — listen for blockers; removing them is your job), **sprint** (a 1–2 week unit of planned work), **retro** (what to improve), **post-mortem/incident review** (after an outage: *blameless* analysis of what failed and what will prevent it — the health of a team shows in whether these are honest), **on-call** (the rotation of who gets woken at 3 a.m.; if on-call is miserable, your best people will leave — ask about it monthly).

The health metrics that matter: **uptime/availability** (“three nines” = 99.9% ≈ 8.8 hours down/year; each extra nine multiplies cost), **SLA** (the promise, with penalties), **SLO** (the internal target), **MTTR** (how fast you recover — mature teams optimize recovery, not the fantasy of zero failures), **deployment frequency** (healthy teams ship small changes often; fear of deploying is a smell).

Vocabulary:

Term	Definition
DevOps	The culture of one team building <i>and</i> running its software, shipping small changes frequently with automation.
CI/CD pipeline	The automated conveyor belt that builds, tests, and deploys every code change (Continuous Integration / Continuous Delivery).
Deploy / rollback	Deploy: release a change to the live system. Rollback: undo it fast when it misbehaves.
Git / repository / pull request	Git: the version-control system recording every change. Repository (“repo”): one project’s code home. Pull request (PR): a

Term	Definition
	proposed change another engineer reviews before it merges.
IaC (Infrastructure as Code)	Declaring infrastructure in text files a tool turns into reality — reviewed, reversible, auditable.
Terraform	The most popular IaC tool (works on every cloud). AWS's own is CloudFormation.
Staging vs production (“prod”)	Staging: the rehearsal copy of the system. Prod: the real one customers touch. “Broke prod” = the bad day.
Stand-up	Daily 15-minute sync: done / next / blocked. Listen for blockers — removing them is your job.
Sprint / backlog	Sprint: a 1–2 week unit of planned work. Backlog: the ordered to-do list feeding sprints.
Retro	End-of-sprint meeting on how to work better next time.
Post-mortem	The blameless review after an outage: what failed, why, what prevents a repeat.
On-call	The rotation of who answers the 3 a.m. alarm. If on-call is miserable, your best people leave.
Incident / sev-1	An unplanned disruption; “sev-1” (severity one) = the worst kind, all-hands.
SLA / SLO	SLA: the customer promise with penalties (e.g., 99.9% uptime). SLO: the stricter internal target that protects the SLA.
MTTR	Mean Time To Recovery — how fast you’re back up after failure. Mature teams optimize this, not the fantasy of never failing.
Monitoring / observability	Watching live health via logs (event records), metrics (numbers over time), and alerts (automatic pages when thresholds break).

Videos for this module:

Video	Channel	Length	Link
What is DevOps? REALLY understand it	TechWorld with Nana	~15 min	https://www.youtube.com/watch?v=0yWAtQ6wYNM

Video	Channel	Length	Link
DevOps CI/CD Explained in 100 Seconds	Fireship	2 min	https://www.youtube.com/watch?v=scEDHsr3APg
Terraform explained in 15 mins	TechWorld with Nana	18 min	https://www.youtube.com/watch?v=l5k1ai_GBDE

Fluency drill: “Is that change through the pipeline or was it manual?” · “What did the post-mortem conclude, and what’s the prevention item?” · “What’s our MTTR trending like?” · “Is this in Terraform, or did someone click it into existence?” (the latter is called “ClickOps,” said with a frown).

Exercises: (1) Watch one real incident post-mortem write-up being discussed (many are public — Cloudflare’s and AWS’s outage reports are famous) and summarize it in your journal in five sentences. (2) Sit in (or watch a recording of) any stand-up and write down the three blockers you heard.

Milestone — end of Stage 1: you can sit through a 30-minute technical planning meeting and follow ≥80% of it, and your journal proves it: notes from one real or simulated meeting with every acronym correctly expanded. **This is also when you should schedule the AWS Cloud Practitioner exam (CLF-C02)** — 2–4 focused weeks of prep on top of these modules is the published norm, and passing it is your first external proof.

STAGE 2 — THINK LIKE THE ROOM (Weeks 9–16)

Module 5 (Weeks 9–10): Architecture — judging designs without drawing them

Your role in a design review is not to design — it is to interrogate. The framework the whole industry uses is AWS’s **Well-Architected Framework**, six pillars every design should answer to: operational excellence, **security**, **reliability**, performance efficiency, **cost optimization**, sustainability. Learn the bold three deeply; they’re where the money and risk live.

Reliability, in plain words: everything fails eventually, so good systems assume it.

Redundancy (no single point of failure — two of everything important), **multi-AZ** (survive a data-center loss), **auto-scaling** (machines added/removed automatically with demand), **backups that are tested** (an untested backup is a hope, not a plan), **RTO/RPO** (Recovery Time Objective: how long can we be down; Recovery Point Objective: how much data can we lose — these two numbers *are* the disaster-recovery conversation, and they are business decisions, i.e., **yours**).

The trade-off triangle: fast, cheap, resilient — pick two. Every architecture argument you will ever referee reduces to where on that triangle the business needs to sit for *this* workload. A payment system and a marketing site do not deserve the same answer.

The leader's seven questions — memorize these; they make you dangerous in any design review:

1. “What happens when this component fails?” (there is always a “when”)
2. “What are the RTO and RPO, and who signed off on them?”
3. “Where does this run — single AZ, multi-AZ, multi-region — and why?”
4. “What will this cost per month at 10× today’s load?”
5. “What’s the simplest version that meets the requirement?” (over-engineering is the junior disease; simplicity is the senior virtue)
6. “What are we locked into, and what would leaving cost?” (vendor lock-in is a price, sometimes worth paying — knowingly)
7. “Who else has built this before — are we inventing or assembling?” (prefer boring, proven patterns)

Fluency drill: practice delivering questions 1, 4, and 5 in a warm tone — they land as wisdom or as attack depending entirely on delivery. “Help me understand what happens if the cache goes down” beats “did you think about failure?”

Exercises: (1) Take three sample architectures (ask an AI to generate: an e-commerce site, a mobile-app backend, a data-analytics platform) and run all seven questions against each, writing the answers you’d expect. (2) Read the one-page summary of the Well-Architected pillars at <https://aws.amazon.com/architecture/well-architected/>.

Videos for this module:

Video	Channel	Length	Link
The Five Pillars of the AWS Well-Architected Framework	Amazon Web Services (official; a sixth pillar, Sustainability, was added later)	~4 min	https://www.youtube.com/watch?v=KvEDbPmha6o
What is the AWS Well-Architected Framework?	Tech With Lucy (ex-AWS)	~10 min	https://www.youtube.com/watch?v=MpDJ6TCWKjk

Milestone: in a mock design review (do it with an AI playing the engineer), you ask five substantive questions and correctly summarize the design’s weakest point at the end.

Module 6 (Weeks 11–12): Security and compliance — the conversation that can end companies

The shared responsibility model — the first thing to understand: the provider secures the cloud itself (buildings, hardware, hypervisors); **you** secure what you put *in* it (your data, access rules, configurations). Most breaches are customer misconfigurations — an S3 bucket left public, a leaked access key — not provider failures. So “AWS is secure” and “we are secure on AWS” are different sentences.

The vocabulary of defense: **IAM** (Identity and Access Management — who may do what; the single most audited thing in cloud), **least privilege** (everyone gets the minimum access needed — the golden rule), **MFA** (second factor on every human login — non-negotiable), **encryption at rest and in transit** (data scrambled on disk and on the wire — table stakes, always on), **root account** (the master key — locked away, never used daily), **secrets management** (passwords/keys in a vault, never in code), **zero trust** (verify everything, trust no network location by default), **penetration test** (hired attackers proving your defenses), **ransomware** (why tested, offline backups are a security control, not just an ops control).

PDPA — Thailand’s data law, and your market advantage. The Personal Data Protection Act is Thailand’s GDPR: consent required for collecting personal data, a **72-hour breach notification** duty, Data Protection Officers for large-scale processors, and rules on sending data across borders. Enforcement became real in 2025 — over THB 21.5M in first fines, including a hospital fined for poorly supervising a *vendor*. Two consequences for you: (1) every Thai enterprise now has a compliance problem your company can be paid to manage; (2) **data residency** — keeping Thai data on Thai soil — is a real driver pushing workloads into the Bangkok regions. This paragraph is half of your sales pitch in Thailand; know it cold.

The leader’s security questions: “Who has access to production, and when did we last review the list?” · “Are we alerted on unusual access, or would we find out from the news?” · “When did we last *restore* a backup?” (not “do we have backups”) · “If we lost this dataset, is it a PDPA-notifiable breach — and could we notify within 72 hours?” · “What did the last pen test find, and what’s still open?”

Vocabulary:

Term	Definition
IAM (Identity and Access Management)	The system controlling who (people and software) may do what in your cloud — the most audited thing in it.
Role / policy	A policy is a written permission set; a role is a wearable bundle of policies a person or program assumes.
Least privilege	The golden rule: everyone gets the minimum access their job needs, nothing more.
MFA (multi-factor authentication)	A second proof (phone code, hardware key) on top of a password. Non-negotiable on every human login.

Term	Definition
Encryption at rest / in transit	Data scrambled while stored / while traveling the network. Both always on; table stakes.
KMS (key management)	The service that holds and rotates the encryption keys.
Secrets	Passwords, API keys, tokens — stored in a vault, never written into code.
Root account	The master key to the whole cloud account. Locked away with MFA, never used for daily work.
Zero trust	Security posture that verifies every request and trusts no network location by default.
Vulnerability / patching	A known weakness in software / applying the fix. Unpatched systems are how most break-ins start.
Pen test (penetration test)	Hired ethical attackers proving where your defenses fail before real ones do.
Ransomware	Attack that encrypts your data for ransom — the reason tested, offline backups are a <i>security</i> control.
SOC 2 / ISO 27001	The independent security-audit badges enterprise customers demand before signing.
PDPA	Thailand's Personal Data Protection Act — consent, 72-hour breach notification, cross-border transfer rules. Enforced with real fines since 2025.
Data residency	Keeping data physically inside a country's borders — a key reason Thai workloads move to the Bangkok regions.
DPO / breach notification	Data Protection Officer (required for large-scale processors) / the 72-hour duty to report a personal-data breach.

Video for this module: The AWS Shared Responsibility Model — Digital Cloud Training — 4 min — <https://www.youtube.com/watch?v=ESPBBEK-cvo>

Fluency drill: “Is that bucket public? Why?” · “Least privilege — does the intern really need prod access?” · “Where does the PDPA line sit in this design — what’s personal data here and where does it physically live?”

Exercises: (1) Read one famous cloud-misconfiguration breach story (Capital One 2019 is the classic) and write the three-sentence version for your journal. (2) Have an AI simulate a

customer asking “why should I trust you with my data?” and practice the answer using shared-responsibility + PDPA language.

Milestone: you can explain the shared responsibility model and PDPA’s practical meaning to a non-technical Thai business owner in under three minutes — because that explanation is the MSP sales meeting.

Module 7 (Weeks 13–14): The money — cloud economics and FinOps

This is where a non-engineer leader adds value fastest, because most engineers were never taught it and the waste is enormous — surveys consistently find roughly a third of cloud spend is wasted.

How the meter runs: compute is billed per second it’s *on* (not per use — an idle server bills fully; “we left it running” is the classic waste), storage per GB-month, and — the famous trap — **egress:** data flowing *out* of the cloud costs money while data in is free. Large egress bills surprise everyone once; after this module, not you.

The pricing menu: **on-demand** (full price, full flexibility) · **reserved instances / savings plans** (commit 1–3 years for 30–70% off — the single biggest lever on a steady workload) · **spot instances** (up to 90% off for interruptible work like batch jobs) · **rightsizing** (most servers are over-provisioned; shrinking them is free money) · storage tiering (Module 2) · **auto-scaling as a cost tool** (why pay for midnight capacity at noon prices?).

FinOps is the practice of making cloud spend visible, allocated, and optimized continuously: **tagging** every resource with its owner and project (untagged spend is unaccountable spend), **showback/chargeback** (showing each team its bill — behavior changes instantly), budgets with **alerts** (never discover overspend from the invoice), and **unit economics** — the leader’s metric: not “our bill is ฿800k/month” but “our cost *per customer transaction* is falling.” The FinOps Certified Practitioner credential takes about two weeks and is, for a business-side leader, the highest-credibility-per-hour certificate in the entire cloud world. Get it.

The leader’s money questions: “What’s our cost per [customer/transaction/tenant], and which direction is it moving?” · “What percentage of our steady workload is on reservations?” · “What’s untagged?” · “What died but is still billing?” (orphaned disks and idle IPs — every account has them) · “What would this bill look like at 10× growth — does our architecture get cheaper or more expensive per unit?”

Vocabulary:

Term	Definition
On-demand	Pay-as-you-go pricing: full price, cancel anytime. The default, and the most expensive way to run steady workloads.
Reserved / savings plan	A 1–3 year commitment to steady usage in exchange for 30–70% off — the single biggest cost lever.
Spot	Spare capacity at up to 90% off that the

Term	Definition
	provider can reclaim with minutes' notice — perfect for interruptible batch work.
Rightsizing	Shrinking over-provisioned servers to what they actually use. Free money in almost every account.
Egress	Data leaving the cloud — billed per GB while inbound data is free. The famous surprise on invoices.
Tagging	Labeling every resource with owner/project/environment so every baht of spend is attributable. Untagged spend = unaccountable spend.
Showback / chargeback	Showing each team its own cloud bill (showback) or actually billing it internally (chargeback). Behavior changes instantly.
Budget alert	An automatic warning at a spend threshold — so you never learn about overspend from the invoice.
Unit economics	Cost per business unit — per customer, per transaction — the leader's metric, more meaningful than the total bill.
TCO (total cost of ownership)	The full cost of a choice over its life: licenses, people, power, migration — not just the sticker price.
FinOps	The discipline (and team culture) of making cloud spend visible, allocated, and continuously optimized.

Video for this module: What is FinOps? — FinOps Foundation (official) — 2 min — https://www.youtube.com/watch?v=Y-c_xw9bHFW · then the free “Introduction to FinOps” course at <https://learn.finops.org>.

Exercises: (1) Open the AWS pricing calculator and price a small real system (two servers, a database, 500GB storage, 1TB egress) — the exercise of doing it once demystifies every future cost conversation. (2) Ask an AI to roleplay an engineer defending an oversized server; practice negotiating the rightsizing kindly.

Milestone: take (or schedule) the FinOps Practitioner exam, and in a mock review, find four cost problems in a sample bill an AI generates for you.

Module 8 (Weeks 15–16): Fluency bootcamp — putting the language together

This fortnight is pure integration — the difference between knowing words and *being fluent in meetings*.

Daily drill (30 min): have an AI generate a realistic meeting transcript (design review, incident retro, or cost review), with two deliberate technical errors buried in it. Your job: summarize the meeting in five sentences, catch the errors, and write the three questions you'd have asked. Alternate meeting types daily.

The translation muscle (15 min): take one technical statement per day and translate it for three audiences — the CFO (money), a customer (risk/benefit), and a new junior engineer (teaching). Example: “We’re moving the session store from the database to Redis” — CFO: “cuts database load so we defer a \$2M upgrade” — customer: “pages load faster at peak” — junior: “Redis keeps hot data in memory, so we stop hammering the database for every click.”

Reading practice (15 min): one real engineering blog post per day (AWS Architecture Blog, or the engineering blogs of Netflix/Grab — Grab is especially relevant: Southeast Asian scale, Thai-adjacent market). You will now understand 70–80% of them. Look up the rest.

Your exam-prep companion for this stage: the famous free full course — *AWS Certified Cloud Practitioner (CLF-C02) 2026* by Andrew Brown on freeCodeCamp — <https://www.youtube.com/watch?v=7HKot-brXFE> (the earlier 14-hour edition, also valid for the same exam code, is at <https://www.youtube.com/watch?v=NhDYbskXRgc>). Watch it at 1.25× across Weeks 13–16; after Modules 1–7 most of it will feel like revision, which is exactly the sign you’re ready.

Milestone — end of Stage 2, your graduation exam: (1) Pass the **AWS Cloud Practitioner** if you haven’t yet. (2) The mock gauntlet: an AI plays a senior engineer walking you through a flawed architecture for a Thai e-commerce client; you must find the single-AZ database, the public S3 bucket, the missing egress estimate, and the absent PDPA consideration — and deliver the feedback in a tone that makes the “engineer” feel helped, not caught. When you can do that, you are conversationally dangerous, sixteen weeks in.

STAGE 3 — LEAD (Months 5–24)

Module 9 (Months 5–8): The deep certification and the technical floor

Work through **AWS Solutions Architect Associate (SAA-C03)** material — 2–3 months at your one-hour pace. You may or may not sit the exam (as a leader, the *material* is the value; the badge is optional theater), but this is where regions, VPCs, IAM, and pricing stop being vocabulary and become a connected system in your head. In parallel, keep the Module 8 daily drills at half dose. This is also the window to add **Azure fundamentals (AZ-900 level)** — Thailand is a Microsoft-heavy enterprise market, and bilingual (AWS+Azure) literacy widens your customer conversations.

Module 10 (Months 5–12, running continuously): Hiring and managing engineers

Hiring when you can’t fully judge the skill: structure beats instinct. Use a consistent loop: a screening conversation you lead (motivation, communication, how they explain a past project

to a non-engineer — if they can't, they'll fail your customers too) + a technical interview conducted by your **bar-raiser** (your technical #2 or a paid senior contractor) + a reference call where you ask exactly one question that matters: “would you hire this person again for this role?” Watch for the two failure archetypes: the fluent talker with no depth (your bar-raiser catches them) and the deep-but-silent expert (often gold in Thai teams, where modesty is cultural — don't let interview polish outweigh evidence of real work).

The technical #2 — the most important decision of the venture: hire them first, pay with meaningful equity (10–20% if a true co-founder), and define the deal explicitly: they hold the technical bar and own architecture decisions; you own customers, money, priorities, and people; disagreements between you happen in private and resolve before the team sees them.

The rituals that make engineers stay: weekly or biweekly **1:1s** that are about *them* (career, friction, energy — not status updates); a written growth path per person (in Thailand's 70k/year talent-shortage market, growth and certification budgets retain better than salary alone — pay for every cert, with a 12-month vesting bonus on completion); public credit, private correction; and ruthless protection of their focus time — a leader who cancels a sprint-mid meeting is a hero.

Decision forums — how you make technical calls you can't fully evaluate: for any big decision, require a one-page **decision doc** from the proposing engineer: the problem, 2–3 options, costs, risks, recommendation. Then run the meeting with your seven questions from Module 5. You are not the smartest architect in the room and never need to be — you are the person who makes the *best-argued* option win, on schedule, with the money and risk trade-offs made explicit. Engineers deeply respect this when it's done honestly; write down who predicted what (your Decision Journal again) and review predictions quarterly — it calibrates both you and them.

Module 11 (Months 6–24): The credibility curve, honestly

Published experience from engineering-leadership literature converges on this timeline, and pretending otherwise is how non-technical founders fail: **~90 days** to run the business rhythm competently · **~6 months** to baseline effectiveness (meetings fluent, decisions structured, team stable) · **12–18 months** before your technical instincts are independently worth much · **~2 years** before you genuinely hold your own with senior engineers on architecture trade-offs. The mitigations while the curve climbs: borrow credibility (your #2 presents the technical half of sales meetings — customers need to see the bench anyway), never bluff (the fastest credibility-killer; “I don't know — walk me through it” is a leadership sentence), and let your questions do the talking: a leader asking “what's our RPO and who signed off on it?” sounds like thirty years of scars, and after this course, you'll mean it.

Module 12 (Ongoing): The Thai-market layer

Fold in the specifics from TSI Part 5 as operating knowledge: **PDPA** as both compliance duty and product (\$Module 6); the **partner ladders** (AWS Select needs a handful of certified staff + 3 launched deals; Microsoft Solutions Partner needs a 70/100 capability score — your team's certifications are literally sales assets, which is another reason to fund them); **BOI promotion** for 100% foreign ownership of the company; Bangkok salary bands (junior ฿50–75k —

architect ฿180–280k/month) so you price bids and offers correctly; and the sales math of the moment — \$27B+ of approved data-center investment, a government cloud-first mandate, and a 70,000/year skills gap that your certified bench exists to fill.

The one-page syllabus

When	Focus	External proof
Weeks 1–2	What cloud is; IaaS/PaaS/SaaS; regions/AZs	—
Weeks 3–4	Compute, storage, databases	—
Weeks 5–6	Networking; reading architecture diagrams	—
Weeks 7–8	DevOps, IaC, team rituals, ops metrics	Schedule Cloud Practitioner
Weeks 9–10	Architecture judgment; the seven questions	—
Weeks 11–12	Security; PDPA; shared responsibility	AWS Cloud Practitioner exam
Weeks 13–14	Cloud economics; FinOps	FinOps Practitioner (≈2 weeks prep)
Weeks 15–16	Fluency bootcamp; mock gauntlet	Graduation: the gauntlet
Months 5–8	SAA-C03 material; Azure AZ- 900	SAA exam (optional)
Months 5–12	Hiring; technical #2; decision forums	First hires made well
Months 6–24	Credibility curve; Thai market layer	Partner tier; first retainers

A closing word from your teacher. Sixteen weeks from now you will follow every conversation in the room. That is not the finish line — it's the license to start. The two-year curve to real technical judgment is not a wall; it is a moat: every week of it you complete is a week your competitors' non-technical founders didn't. Study daily, journal every decision, never bluff, and hire people better than you and make them glad they came. That is the whole job.

Companion to “The Thailand Strategic Investment (TSI),” Part 5. Certification prep-time estimates and leadership timelines are drawn from the sources cited there (CBT Nuggets, StudyTech, FinOps Foundation, First Round Review, The Pragmatic Engineer).

NEW LESSONS — AUGUST 2026 EDITION

In August 2026, new lessons were added to the three interactive courses at b4lcilc.org — the Cloud Leader, Cloud Engineer, and Cloud Architect courses. This addendum brings the reading edition up to date with faithful condensations of that new material, in the same voice as the rest of this book.

A word on what you are holding. These pages are the reading edition: the ideas, the metaphors, and the sentences worth saying out loud. The full interactive versions online carry considerably more — each new lesson ships with quizzes that check your understanding before letting you move on, hands-on labs (including a practice terminal that behaves like the real one but cannot cost money), videos, vocabulary drills, and — for the capstone material — guided design exercises with requirement checklists and worked model answers. Read these condensations to get the shape of each idea into your head; then, when you can, sit the interactive versions to make the knowledge stick.

*The additions fall into three groups. For the **Cloud Leader** course: a gentle introduction to the terminal, Thailand's own privacy law, a leader's guide to AI proposals and AI bills, and a capstone exercise that assembles the entire course into one page. For the **Cloud Engineer** course: serverless computing — the pattern that turns small operational scripts into functions that cost nothing to run. For the **Cloud Architect** course: the arithmetic of availability promises, the Retrieval-Augmented Generation pattern that customers now ask about weekly, and a full costing exercise — pricing a design before anyone builds it.*

For the Cloud Leader course

The black window — how to speak to a computer directly

Before your first laboratory, meet the tool it runs in. A **terminal** (also called a *command line* or *console*) is a program that looks like a plain black window with blinking text. It is the oldest way of using a computer, and still how engineers do serious work — because a typed instruction is precise, repeatable, and can be written down and shared, in a way clicking through menus never can be.

The terminal shows a **prompt** — usually a short line ending in \$ — which means exactly one thing: *I am listening*. A **command** is a sentence with a grammar: the first word is the *tool* you are calling (like `git` or `aws`), the next words say *what to do*, and pieces starting with a dash, called **flags** (like `--output table`), adjust *how* to do it.

Two customs surprise every newcomer. First: **silence usually means success.** Many commands, when they work perfectly, print nothing at all — the machine considers doing the job to be answer enough. Second: the terminal does not guess. Where a search engine forgives spelling, the terminal forgives nothing — one wrong letter and it politely tells you the command does not exist. That strictness feels harsh for a day and then becomes a comfort: the machine never does *roughly* what you asked.

One more piece of grammar: computers organize everything into **files** living inside **folders** (also called *directories*) — so `mkdir decisions` means “*make directory* called decisions.” The course’s labs run in a **simulator**: a practice terminal that cannot create, delete, or cost anything — but the commands are the real ones, letter for letter. Master them here, and the day you sit at a real terminal, your hands will already know the way.

Thailand’s PDPA — the privacy law closest to home

The rest of the compliance module is global; this is the law whose regulator can actually knock on the door. Thailand’s PDPA — the **Personal Data Protection Act, B.E. 2562 (2019)** — came fully into force in 2022 and applies to any organization in Thailand handling personal data. A Mae Sot charity holding student records is squarely inside it.

The good news is genuinely good: the PDPA is a near-copy of GDPR — the European General Data Protection Regulation. Thailand’s drafters took the European regulation as their template, so GDPR has already taught you roughly ninety percent of this law. Personal data belongs to the person it describes; you need a lawful basis — most often **consent**, freely given and revocable — to collect it; and **purpose limitation** forbids using data gathered for one purpose for another: records collected to enroll a student may not quietly become a fundraising mailing list.

The regulator is the **PDPC — the Personal Data Protection Committee** — whose penalties reach criminal liability, not only fines. The breach clock is familiar too: a breach likely to pose a risk to individuals must be reported to the PDPC **within 72 hours**.

The leadership point deserves stating plainly. An organization holding records on students — names, families, and in this region sometimes documentation status, about as sensitive as data gets — **answers to this law first**: the answer must satisfy Bangkok before it worries about Brussels. The questions to ask your team: what personal data do we hold, and did we state the purpose when we collected it? Could we delete one student’s records completely if asked? If breached tonight, could we tell the PDPC within 72 hours what was taken and whose? None requires a lawyer to ask. All require a leader to insist on.

Leading through the AI wave

Sooner or later — and lately it is sooner — a design will arrive on your desk with AI in it. Your job does not change: you interrogate, you do not design.

The engine behind most AI features is an **LLM — a Large Language Model** — and the crucial commercial fact: you almost never own one; you rent access through an **AI API** (Application Programming Interface), billed **by the token** — a word-piece of roughly three-quarters of an English word — so a feature costs more whenever questions or answers grow longer. Ask to see the token maths, per user, per month; if nobody has done it, the feature has no price yet, only a hope.

The alternative — running your own model — means renting **GPUs (Graphics Processing Units)**, the specialized chips AI requires: scarce, priced many times above ordinary servers, and billing for every hour switched on whether thinking or idle. The first rule of the AI line: **an idle GPU is the most expensive idleness in the building**. A forgotten test server wastes coffee

money; a forgotten GPU instance wastes a salary. And when someone says *let's fine-tune our own model*, remember the honest sequence: careful prompting on a rented API first, then giving the model your documents at question time, and only when both fall short, fine-tuning.

Three questions carry you through any AI proposal. One: what does a wrong answer cost? LLMs are confidently wrong a small, irreducible fraction of the time; the acceptable error rate is a business decision — which means it is yours. **Two: what data leaves us?** Every question sent to an AI API travels to the provider's servers; if it contains a student's name or refugee status, the compliance module applies in full. **Three: what is the per-user monthly cost at ten times the users?** Token pricing scales linearly — a comfortable \$30 pilot becomes \$300 when it succeeds, with no volume discount arriving to save you. The same three pillars as ever — reliability, security, cost — wearing new vocabulary. The technology is novel; the interrogation is not.

The capstone — one exercise that assembles the whole course

The course now ends with a single design exercise that uses everything in it. The brief: you have just been appointed cloud leader of a border-region education charity in Mae Sot. Its estate — a public website, a database of about 2,000 student records (including, for some, documentation status), three learning centers, and one talented, exhausted engineer holding the only production access and the only on-call phone. Last month's cloud bill arrived at nearly double the usual amount and nobody can say why. The board's one technology item: "Can we use AI for tutoring?" The task: your first 90 days on one page, with the sequence justified.

The interactive version grades the plan against a checklist: inventory before opinions; the three-list map of machines, labels, and money; one quick win; the security sweep and backup-restore test with dates attached; a budget tripwire; an honest position on the one-engineer risk; the PDPA named as the frame for the student records; the AI question answered with the three AI questions; and a day-90 board review, reported in unit terms.
