

Khóa học Kỹ sư Cloud (The Cloud Engineer Course)

Từ con số không đến một kỹ sư cloud thực sự sẵn sàng đi làm — một khóa học B4LCILC

Ngày 13 tháng 8 năm 2026

Khóa học Kỹ sư Cloud (The Cloud Engineer Course)

Từ chỗ chưa biết gì đến thực sự sẵn sàng đi làm với vai trò kỹ sư cloud.

Khóa học này vận hành như thế nào

Bạn đang rèn luyện để *trở thành* một kỹ sư cloud: người dựng server (máy chủ) bằng code, giữ cho hệ thống sống sót lúc 3 giờ sáng, tự động hóa những việc nhàm chán, và có thể nhìn vào một hệ thống đang hỏng rồi bình tĩnh tìm ra lý do vì sao. Đó là một nghề thủ công thực hành, và khóa học này đối xử với nó đúng như vậy. Bạn sẽ dành nhiều thời gian gõ phím hơn hẳn thời gian đọc — mỗi module đều có lab (bài thực hành), và các lab chính là khóa học. Đọc về cloud tạo ra sự nhận biết; xây dựng trên cloud tạo ra một sự nghiệp.

Khóa học gồm năm giai đoạn. **Giai đoạn 0 (Tuần 1–4): Nền tảng** — máy tính và mạng thực sự hoạt động ra sao, và Linux, hệ điều hành mà cloud vận hành trên đó. **Giai đoạn 1 (Tuần 5–12): Cloud cốt lõi, thực hành** — năm dịch vụ AWS xuất hiện trong mọi bản mô tả công việc, được học dưới dạng các lab mà bạn tự dựng lên rồi tháo dỡ. **Giai đoạn 2 (Tuần 13–20): Tự động hóa** — scripting (viết kịch bản lệnh), Git, Terraform, CI/CD và container: những kỹ năng phân biệt một kỹ sư với một người chỉ biết nhấp chuột trong console. **Giai đoạn 3 (Tuần 21–28): Vận hành** — giám sát, sự cố, bảo mật và chi phí: những kỹ năng mà người ta thực sự trả lương cho bạn. **Giai đoạn 4 (Tháng 8–12): Sẵn sàng đi làm** — ba dự án portfolio, ba chứng chỉ, và luyện phỏng vấn.

Quy tắc cho toàn khóa học: học **1,5–2 giờ mỗi ngày, sáu ngày mỗi tuần** — sự đều đặn thắng sự dôn dập, và đây là một nghề học bằng lặp lại hằng ngày, như một nhạc cụ. Mỗi module kết thúc bằng một **bảng Từ vựng** (những từ bạn phải làm chủ), một **bài luyện Nói thành tiếng** (những câu bạn lặp lại cho đến khi chúng trở nên tự nhiên — phỏng vấn xin việc là nói, không phải viết), các **Bài tập** thực hành, và một **Milestone** (cột mốc) đóng vai trò cổng kiểm soát tiên độ: đừng đi tiếp cho đến khi bạn làm được nó, vì mỗi giai đoạn đứng trên vai giai đoạn trước. Ngay từ Tuần 1, hãy duy trì một **Engineering Journal** (nhật ký kỹ thuật): mỗi lab, mỗi thông báo lỗi, mỗi cách sửa, một đoạn văn trung thực. Đến Tháng 10, cuốn nhật ký đó trở thành nguyên liệu thô cho portfolio của bạn và những câu chuyện phỏng vấn của bạn.

Một lời hứa để đáp lại: không có gì trong khóa học này giả định bạn đã biết trước điều gì. Mọi thuật ngữ đều được định nghĩa ngay lần đầu xuất hiện. Nếu bạn biết dùng trình duyệt web và sẵn lòng gõ những câu lệnh có vẻ xa lạ trong hai tuần đầu, bạn đã có đủ mọi điều kiện tiên quyết cần thiết.

Bản mô tả công việc mà bạn đang rèn luyện để đáp ứng

Vào tháng 8 năm 2026, chúng tôi thu thập các bản mô tả công việc Cloud Engineer thực tế từ các mẫu tuyển dụng và cẩm nang tuyển dụng (Arc.dev, Wiz, DevsData, XOPA, Betterteam — danh sách đầy đủ trong phần Nguồn tham khảo). Bỏ tên công ty đi thì cùng những yêu cầu ấy xuất hiện lặp đi lặp lại. Bảng này là bản hợp đồng của bạn với khóa học — mỗi gạch đầu dòng mà nhà tuyển dụng đòi hỏi đều ảnh xạ đến giai đoạn dạy nó:

Những gì bản mô tả công việc thực tế yêu cầu (gần như nguyên văn)	Nơi khóa học này dạy điều đó
“Knowledge of Linux/Unix operating systems” (Kiến thức về hệ điều hành Linux/Unix)	Giai đoạn 0, Module 1
“Expertise in cloud networking including VPCs, subnets, load balancers, DNS” (Chuyên môn về mạng cloud gồm VPC, subnet, load balancer, DNS)	Giai đoạn 0 Module 2 + Giai đoạn 1 Lab 3
“Demonstrated expertise in core AWS services, including EC2, S3, RDS, VPC, IAM” (Chuyên môn được chứng minh về các dịch vụ AWS cốt lõi, gồm EC2, S3, RDS, VPC, IAM)	Giai đoạn 1 (Lab 1–5)
“Design, develop, and deploy cloud infrastructure using infrastructure as code tools such as Terraform, CloudFormation” (Thiết kế, phát triển và triển khai hạ tầng cloud bằng các công cụ infrastructure as code như Terraform, CloudFormation)	Giai đoạn 2, Module 7
“Proficiency in scripting languages such as Python, Bash, PowerShell, or Go” (Thành thạo các ngôn ngữ scripting như Python, Bash, PowerShell hoặc Go)	Giai đoạn 2, Module 6
“Build and maintain CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions” (Xây dựng và duy trì các pipeline CI/CD bằng các công cụ như Jenkins, GitLab CI hoặc GitHub Actions)	Giai đoạn 2, Module 8
“Experience with containerization technologies including Docker and Kubernetes” (Kinh nghiệm với các công nghệ container hóa gồm Docker và Kubernetes)	Giai đoạn 2, Module 8
“Monitor infrastructure health and performance using cloud-native monitoring tools” (Giám sát sức khỏe và hiệu năng hạ tầng bằng các công cụ giám sát cloud-native)	Giai đoạn 3, Module 9

Những gì bản mô tả công việc thực tế yêu cầu (gần như nguyên văn)	Nơi khóa học này dạy điều đó
“Participate in incident response, including log analysis”; “troubleshooting and analytical skills” (Tham gia ứng phó sự cố, bao gồm phân tích log; kỹ năng xử lý sự cố và phân tích)	Giai đoạn 3, Module 10
“Implement and enforce security controls including encryption, identity and access management”; “least-privilege access” (Triển khai và thực thi các biện pháp kiểm soát bảo mật gồm mã hóa, quản lý danh tính và quyền truy cập; truy cập đặc quyền tối thiểu)	Giai đoạn 1 Lab 5 + Giai đoạn 3 Module 10
“Manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging” (Quản lý chi phí cloud thông qua điều chỉnh kích cỡ tài nguyên, triển khai auto-scaling, gắn thẻ tài nguyên)	Giai đoạn 3, Module 10
“Maintaining, testing and implementing disaster recovery procedures” (Duy trì, kiểm thử và triển khai các quy trình khôi phục thảm họa)	Giai đoạn 3 + Dự án Portfolio 3
“AWS certifications preferred” (Ưu tiên có chứng chỉ AWS)	Lịch thi chứng chỉ Giai đoạn 4 (CCP – SAA – Terraform Associate)
“Provide technical guidance and documentation”; “good communication and collaboration skills” (Cung cấp hướng dẫn kỹ thuật và tài liệu; kỹ năng giao tiếp và cộng tác tốt)	Nhật ký + runbook + luyện phỏng vấn Giai đoạn 4

Bối cảnh lương, để bạn biết mình đang hướng tới điều gì: lương kỹ sư cloud tại Mỹ tập trung quanh **mức trung vị gần \$104.000**, với khoảng phổ biến **\$85K–\$140K**, và các vị trí AWS senior/chuyên gia được đăng tuyển cao hơn hẳn mức đó. Các vị trí đầu vào tồn tại dưới nhiều tên gọi — cloud support associate, junior cloud engineer, cloud operations engineer — và khóa học này nhằm đúng vào các yêu cầu của chúng.

GIAI ĐOẠN 0 — NỀN TẢNG (Tuần 1-4)

Module 1 (Tuần 1-2): Máy tính hoạt động ra sao, và Linux — ngôn ngữ của server

Ý tưởng lớn: một **server** chỉ là một chiếc máy tính có nhiệm vụ phục vụ các máy tính khác. Chiếc laptop trước mặt bạn và những cỗ máy chạy Netflix khác nhau ở kích cỡ và độ tin cậy, chứ không khác về bản chất: cả hai đều là một **CPU** (bộ phận làm việc), **memory/RAM** (bộ nhớ — vùng làm việc ngắn hạn tốc độ cao, bị xóa khi khởi động lại), **disk** (ổ đĩa — kho lưu trữ dài hạn chậm hơn, sống sót qua các lần khởi động lại), và một **network card** (card mạng — kết nối đến mọi thứ khác), tất cả được điều phối bởi một **operating system (OS — hệ điều hành)**. Laptop của bạn có lẽ chạy Windows hoặc macOS. Server thì áp đảo chạy **Linux** — một hệ điều hành miễn phí, mã nguồn mở, ổn định, tự động hóa được bằng script, và được điều khiển hoàn toàn bằng các câu lệnh gõ tay. Phần cuối cùng đó chính là điểm mấu chốt: bạn không thể tự động hóa các cú nhấp chuột, nhưng bạn có thể tự động hóa các câu lệnh, và kỹ thuật cloud *chính là* tự động hóa. Vì vậy sự trôi chảy đầu tiên của bạn là **terminal** (cửa sổ dòng lệnh).

Terminal, được giải mã. Terminal (hay “shell” — chương trình bên trong nó thường là **Bash**) là một cuộc trò chuyện bằng văn bản với máy tính. Bạn gõ một câu lệnh; nó trả lời. Chỉ vậy thôi. Ký hiệu \$ bạn sẽ thấy trong các ví dụ là **prompt** (dấu nhắc lệnh) — shell đang nói “đến lượt bạn.” Mọi thứ trong Linux đều là file, các file sống trong một cây duy nhất bắt đầu từ gốc /, và thư mục cá nhân của bạn là /home/yourname (tên gọi tắt: ~). Một **path** (đường dẫn) là địa chỉ của một file trong cây đó: /home/anna/notes.txt.

Kiểm một Linux để luyện tập (chọn một, mười phút): trên Windows, cài **WSL** (Windows Subsystem for Linux — một Ubuntu Linux thật sự bên trong Windows:

<https://learn.microsoft.com/en-us/windows/wsl/install>); trên máy Mac, ứng dụng Terminal có sẵn đủ gần để bắt đầu (macOS là họ hàng của Unix); hoặc đợi đến Tuần 5 khi bạn sẽ thuê một server Linux thật từ AWS miễn phí. WSL là câu trả lời tốt nhất cho đa số mọi người.

25 câu lệnh hàng đầu — bảng này chính là Tuần 1-2. Hãy gõ từng lệnh một, nhiều lần:

Câu lệnh	Nó làm gì	Ví dụ
pwd	Print working directory — “tôi đang ở đâu?”	pwd
ls	Liệt kê các file ở đây (-l chi tiết dạng dài, -a gồm cả file ẩn)	ls -la
cd	Change directory — di chuyển trong cây thư mục	cd /var/log
mkdir	Tạo một directory (thư mục)	mkdir projects
touch	Tạo một file rỗng	touch notes.txt
cp	Sao chép một file (-r cho thư mục)	cp a.txt backup.txt

Câu lệnh	Nó làm gì	Ví dụ
mv	Di chuyển hoặc đổi tên	mv old.txt new.txt
rm	Remove — xóa vĩnh viễn, không có thùng rác. Hãy tôn trọng nó.	rm notes.txt
cat	In toàn bộ nội dung một file	cat notes.txt
less	Đọc một file dài theo từng trang (q để thoát)	less /var/log/syslog
head / tail	Những dòng đầu / cuối của một file. tail -f theo dõi log trực tiếp — một kinh điển của dân vận hành	tail -f app.log
grep	Tìm kiếm văn bản theo một mẫu — câu lệnh vận hành được dùng nhiều nhất	grep "ERROR" app.log
find	Tìm file theo tên/kích cỡ/tuổi	find / -name "*.conf"
echo	In văn bản (thường vào file hoặc biến)	echo "hello"
nano	Một trình soạn thảo văn bản thân thiện trong terminal	nano notes.txt
man	Sổ tay hướng dẫn cho bất kỳ lệnh nào (q để thoát)	man grep
sudo	Chạy một lệnh với tư cách quản trị viên toàn quyền ("root"). Một cách tôn trọng.	sudo apt update
apt	Cài đặt/cập nhật phần mềm (họ Ubuntu/Debian)	sudo apt install htop
chmod	Thay đổi quyền của một file	chmod 644 notes.txt
chown	Thay đổi chủ sở hữu của một file	sudo chown anna file
ps	Liệt kê các process (tiên trình) đang chạy (ps aux cho tất cả)	ps aux
top (hoặc htop)	Bảng điều khiển trực tiếp của CPU/bộ nhớ/tiên trình — "vì sao server chậm?" bắt đầu từ đây	top
df -h / du -sh	Dung lượng đĩa còn trống / dung lượng một thư mục chiếm — "đĩa đầy rồi" bắt đầu từ đây	df -h

Câu lệnh	Nó làm gì	Ví dụ
ssh	Đăng nhập vào terminal của một máy khác qua mạng — <i>chính là cửa trước của kỹ sư cloud</i>	ssh anna@server-ip
curl	Gửi một yêu cầu web từ terminal — “trang web có đang chạy không?”	curl https://example.com

Quyền truy cập (permissions), phiên bản 60 giây. Mỗi file có một chủ sở hữu và một mode như `rw-r--r--`: ba bộ ba — chủ sở hữu, nhóm, mọi người — của **read** (đọc), **write** (ghi), **execute** (thực thi). Theo số: `r=4, w=2, x=1`, nên `chmod 755 script.sh` nghĩa là “chủ sở hữu được làm mọi thứ ($7=4+2+1$); những người khác được đọc và chạy nó ($5=4+1$).” Khi một chương trình “không có quyền”, đó là hệ thống đang nói không — và giờ bạn có thể đọc được lý do vì sao.

SSH, phiên bản 60 giây. `ssh` mở một terminal từ xa an toàn trên một máy khác — từ laptop của bạn vào một server ở Virginia như thể bạn đang ngồi ngay trước nó. Thay vì mật khẩu, dân chuyên nghiệp dùng một **key pair** (cặp khóa): một private key (khóa riêng — một file bí mật trên laptop của bạn, không bao giờ chia sẻ) và một public key (khóa công khai — được đặt trên server). Chúng khớp nhau như chìa và ổ khóa. Mỗi server AWS bạn khởi chạy trong Giai đoạn 1 sẽ trao cho bạn chính xác thứ này.

Process và service: một **process** (tiến trình) là một chương trình đang chạy; một **service** (dịch vụ, hay “daemon”) là một tiến trình chạy mãi mãi ở chế độ nền — web server, cơ sở dữ liệu. `systemctl status nginx` hỏi Linux “dịch vụ nginx có khỏe không?” — một câu bạn sẽ gõ một cách chuyên nghiệp trong nhiều năm.

Từ vựng:

Thuật ngữ	Định nghĩa
Server	Một máy tính có nhiệm vụ phục vụ các máy tính khác. Trên cloud, là một chiếc bạn thuê.
CPU / RAM / disk	Người làm việc, vùng làm việc tạm thời tốc độ cao (bị xóa khi khởi động lại), và kho lưu trữ vĩnh viễn chậm hơn.
Operating system (OS)	Phần mềm vận hành cỗ máy và chứa các chương trình. Server chạy Linux.
Linux / distribution	Hệ điều hành server miễn phí, mã nguồn mở. Một “distro” (Ubuntu, Amazon Linux, Debian) là một phiên bản đóng gói của nó.
Terminal / shell / Bash	Giao diện văn bản với hệ điều hành / chương trình diễn giải các lệnh của bạn / tên của shell tiêu chuẩn.
Prompt	Ký hiệu \$ — shell đang chờ lệnh của bạn.

Thuật ngữ	Định nghĩa
Directory / path	Một thư mục / địa chỉ đầy đủ của một file trong cây duy nhất bắt đầu từ /.
Root (hai nghĩa)	Đỉnh của cây file (/) và người dùng quản trị toàn quyền. Ngữ cảnh cho bạn biết là nghĩa nào.
sudo	“Superuser do” — chạy một lệnh với quyền quản trị.
Permissions (rwx)	Quy tắc theo từng file về ai được đọc, ghi, thực thi — hiển thị dưới dạng bộ ba cho chủ sở hữu/nhóm/mọi người.
Process / service (daemon)	Một chương trình đang chạy / một chương trình chạy mãi mãi ở chế độ nền (web server, cơ sở dữ liệu).
SSH / key pair	Đăng nhập từ xa an toàn vào terminal của máy khác / cặp file khóa riêng+công khai thay thế mật khẩu.
Log	Một file văn bản nơi phần mềm ghi lại những gì đã xảy ra — nơi đầu tiên bạn nhìn vào khi bất cứ thứ gì hỏng.
Package manager	Trình cài đặt của hệ điều hành (apt, yum) — phần mềm bằng câu lệnh, không phải bằng trang tải về.

Video cho module này (liên kết đã kiểm chứng):

Video	Kênh	Thời lượng	Liên kết
Linux Operating System — Crash Course for Beginners	freeCodeCamp	~2 giờ	https://www.youtube.com/watch?v=ROjZy1WbCIA
WSL install guide (tài liệu tham khảo, không phải video)	Microsoft Learn	—	https://learn.microsoft.com/en-us/windows/wsl/install

Nói thành tiếng cho đến khi tự nhiên: “Let me SSH in and check the logs.” (Để tôi SSH vào và kiểm tra log.) · “Grep the log for the error, then tail -f it while we retry.” (Grep log để tìm lỗi, rồi tail -f nó trong khi chúng ta thử lại.) · “It’s a permissions problem — who owns the file and what’s the mode?” (Đây là vấn đề quyền truy cập — ai sở hữu file và mode là gì?) · “Check top — is it CPU, memory, or disk?” (Kiểm tra top — là CPU, bộ nhớ, hay đĩa?)

Bài tập: (1) Trong terminal Linux của bạn, dựng một cây dự án nhỏ bằng mkdir và touch, sao chép và di chuyển vài thứ, rồi xóa nó — vừa làm vừa thuật lại từng câu lệnh thành tiếng. (2) Tạo hello.sh chứa echo "hello from \$(whoami)", làm cho nó thực thi được bằng

`chmod +x`, chạy nó bằng `./hello.sh`. (3) Chạy `tail -f` trên một file log (trên Ubuntu: `sudo tail -f /var/log/syslog`) và xem các dòng xuất hiện. (4) Nhật ký: giải thích cho một người bạn tưởng tượng vì sao server chạy Linux, trong ba câu.

Milestone: không cần ghi chú, bạn có thể di chuyển đến bất kỳ đâu trong cây file, tạo/sao chép/di chuyển/xóa file, giải thích `chmod 755`, và dùng `grep` để tìm một từ trong một file. Nếu bất kỳ điều nào trong số đó đòi hỏi phải tra cứu, hãy dành thêm hai ngày ở đây. Module này là trụ đỡ cho mọi thứ về sau.

Module 2 (Tuần 3-4): Nền tảng mạng + tài khoản AWS của bạn

Ý tưởng lớn: một mạng là các máy tính chuyên cho nhau những phong bì có ghi địa chỉ. Mỗi máy nhận một **IP address** (địa chỉ IP — như `172.31.8.14` — một địa chỉ nhà dạng số). Dữ liệu được cắt thành các **packet** (gói tin — những phong bì) và được định tuyến từng chặng một về phía địa chỉ đích. Khi đến nơi, một số **port** (cổng) cho biết phong bì dành cho chương trình nào — cùng một tòa nhà, hàng nghìn cánh cửa được đánh số: port **22** là SSH, **80** là web không mã hóa (**HTTP**), **443** là web mã hóa (**HTTPS**), **5432** là PostgreSQL. “Mở port 443” nghĩa là “cho phép các phong bì gửi đến cửa số 443.”

Private vs public: nhà bạn và mọi mạng cloud đều tái sử dụng các **đải IP private** (`10.x.x.x`, `172.16–31.x.x`, `192.168.x.x`) chỉ hoạt động bên trong mạng cục bộ; một **public IP** thì truy cập được từ toàn bộ internet. Sự phân chia này là nền móng của bảo mật cloud: những thứ không cần đối mặt với internet thì hoàn toàn không nhận địa chỉ public nào.

DNS — danh bạ điện thoại của internet. Con người dùng tên (`example.com`); gói tin cần số. **DNS** phiên dịch: máy của bạn hỏi một DNS server “IP của `example.com` là gì?”, nhận được con số, rồi kết nối. Một nửa số sự cố bí ẩn có dính đến DNS; câu đùa trong ngành “it’s always DNS” (lúc nào cũng là DNS) tồn tại vì nó thường đúng. `nslookup example.com` thực hiện việc tra cứu bằng tay.

HTTP — cách web trò chuyện. Một **client** (trình duyệt) gửi một request (yêu cầu) — một **method** (`GET` = lấy về, `POST` = gửi đi) cộng với một đường dẫn — và **server** trả lời bằng một **status code** (mã trạng thái): 200 OK, 301 đã chuyển đi, 403 bị cấm, 404 không tìm thấy, 500 lỗi server, 502/503 “server phía sau tôi bị hỏng/quá tải.” Hãy học thuộc sáu mã đó; là một kỹ sư, bạn sẽ đọc chúng hằng ngày. `curl -I https://example.com` cho bạn xem một dòng trạng thái và các header trực tiếp.

Firewall (tường lửa): một danh sách quy tắc quyết định gói tin nào được đi qua, theo nguồn, đích và port — “cho phép 443 từ mọi nơi; cho phép 22 chỉ từ văn phòng; từ chối phần còn lại.” Trong AWS, tường lửa theo-từng-server được gọi là **security group**, và những security group bị cấu hình sai là lỗ hổng bảo mật số 1 của người mới. **Latency** (độ trễ, tính bằng mili giây) và **bandwidth** (băng thông — dung lượng mỗi giây) hoàn thiện bộ tứ vừng: khoảng cách tạo ra độ trễ, đó là lý do các cloud có region trên toàn thế giới.

Bây giờ, tài khoản AWS của bạn — nửa sau của module này. Truy cập <https://aws.amazon.com/free> và tạo một tài khoản Free Tier (email, số điện thoại, thẻ tín dụng/ghi nợ để xác minh danh tính — thẻ sẽ không bị trừ tiền đáng kể nếu bạn tuân theo kỷ luật tháo dỡ của khóa học này). **Free Tier** (gói miễn phí) cho hạn mức hằng tháng với các

dịch vụ cơ bản (bao gồm 750 giờ/tháng cho một server EC2 nhỏ trong năm đầu tiên) và mọi lab trong khóa học này được thiết kế để nằm gọn trong đó. Sau đó, trước bất cứ điều gì khác, ba bước an toàn không thể thương lượng — làm những điều này *chính là* bài tập đầu tiên trong sự nghiệp bảo mật của bạn:

1. **MFA trên root user.** Root user là chìa khóa vạn năng của tài khoản. Thêm multi-factor authentication (xác thực đa yếu tố — một ứng dụng xác thực trên điện thoại) trong IAM — Security credentials. Sau đó ngừng dùng root cho công việc hằng ngày.
2. **Tạo một IAM user quản trị** (Lab 5 giải thích IAM sâu hơn; hiện tại: console — IAM — Users — tạo user với quyền quản trị) và từ giờ đăng nhập bằng user đó.
3. **Một billing alert (cảnh báo chi phí).** Console — Billing — Budgets — tạo một **zero-spend budget** (ngân sách chi tiêu bằng 0 — mẫu có sẵn của AWS gửi email cho bạn ngay khoảnh khắc có bất kỳ khoản nào bị tính tiền). Đồng thời bật cảnh báo sử dụng Free Tier. Một kỹ sư không kiểm soát được chi phí là một mối nguy; bạn mới ở trong tài khoản được 20 phút mà đã đi trước nhiều người làm nghề. (Tham khảo: <https://docs.aws.amazon.com/cost-management/latest/userguide/budgets-managing-costs.html>)

Từ vựng:

Thuật ngữ	Định nghĩa
IP address	Địa chỉ mạng dạng số của một máy, ví dụ 172.31.8.14.
Packet	Một phong bì dữ liệu có ghi địa chỉ; mọi lưu lượng đều là các dòng phong bì như vậy.
Port	Một cánh cửa được đánh số trên một máy, xác định lưu lượng dành cho chương trình nào: 22 SSH, 80 HTTP, 443 HTTPS.
Private / public IP	Một địa chỉ chỉ có giá trị bên trong mạng cục bộ / một địa chỉ truy cập được từ toàn bộ internet.
DNS	Hệ thống dịch tên (example.com) thành địa chỉ IP. “It’s always DNS.”
HTTP / HTTPS	Giao thức yêu cầu-phản hồi của web / cùng giao thức đó, được mã hóa bằng TLS.
Status code	Bản tóm tắt câu trả lời của server: 200 OK, 404 không tìm thấy, 500 lỗi server, 503 quá tải.
Firewall	Danh sách quy tắc quyết định gói tin nào được qua, theo nguồn, đích, port.
Security group	Tường lửa theo-từng-server của AWS. Cấu hình sai một cái là lỗ hổng kinh điển của người mới.
Latency / bandwidth	Độ trễ (ms) / dung lượng (mỗi giây).

Thuật ngữ	Định nghĩa
	Khoảng cách tạo ra độ trễ.
Client / server (vai trò)	Người hỏi và người trả lời trong bất kỳ cuộc trò chuyện mạng nào.
AWS Free Tier	Hạn mức miễn phí hằng tháng trên một tài khoản AWS mới — toàn bộ ngân sách của khóa học này.
Root user	Danh tính chủ của tài khoản AWS. Bật MFA cho nó, rồi ngừng sử dụng nó.
Billing alert / budget	Email tự động khi chi tiêu vượt một ngưỡng. Ngưỡng của bạn được đặt ở mức \$0.

Video cho module này:

Video	Kênh	Thời lượng	Liên kết
What is AWS?	Amazon Web Services (official)	~2 phút	https://www.youtube.com/watch?v=a9_D53WsUs
Top 50+ AWS Services Explained in 10 Minutes	Fireship	~10 phút	https://www.youtube.com/watch?v=JlbiYCM48to
AWS Networking Basics — VPC & Subnets	KodeKloud	~30 phút	https://www.youtube.com/watch?v=QM63dyA_4Pc

Nói thành tiếng: “What’s the IP, and is it public or private?” (IP là gì, và nó là public hay private?) · “Is port 443 open in the security group?” (Port 443 có mở trong security group không?) · “Curl it — what status code do you get?” (Curl nó đi — bạn nhận được status code nào?) · “Did DNS resolve? Check with nslookup before blaming the server.” (DNS có phân giải không? Kiểm tra bằng nslookup trước khi đổ lỗi cho server.)

Bài tập: (1) ping google.com (ghi lại độ trễ), nslookup google.com (ghi lại việc DNS trả về nhiều IP), curl -I https://aws.amazon.com (đọc dòng trạng thái và ba header, rồi tra cứu từng cái). (2) Tìm IP private của máy bạn (ip addr trên Linux) và IP public của bạn (tìm kiếm “what is my IP”) — giải thích trong nhật ký vì sao chúng khác nhau. (3) Hoàn tất thiết lập tài khoản AWS: MFA, user quản trị, zero-spend budget — chụp màn hình budget cho nhật ký. (4) Vẽ từ trí nhớ: laptop — tra cứu DNS — yêu cầu HTTPS qua port 443 — firewall — web server. Ba lần, cho đến khi thành tự động.

Milestone — kết thúc Giai đoạn 0: bạn có thể thuật lại điều gì xảy ra khi bạn gõ một URL và nhấn Enter — DNS, IP, port, firewall, yêu cầu HTTP, status code — trong dưới hai phút, dùng đúng mọi thuật ngữ; và tài khoản AWS của bạn đã tồn tại với MFA và một cảnh báo

ngân sách \$0. Giờ đây bạn biết về cách internet hoạt động nhiều hơn phần lớn những người dùng nó để kiếm sống. Giai đoạn 1 là nơi bạn bắt đầu xây dựng trên nền đó.

GIẢI ĐOẠN 1 — CLOUD CỐT LÕI, THỰC HÀNH (Tuần 5–12)

Giai đoạn này vận hành như thế nào. Mỗi dịch vụ trong năm dịch vụ dưới đây là một **Lab**: một mục tiêu, một dàn ý các bước (đủ chi tiết để làm theo, đủ ngắn để bạn phải suy nghĩ — sự suy nghĩ chính là việc học), những gì bạn học được, và — luôn luôn — **teardown** (tháo dỡ). Kỷ luật tháo dỡ quan trọng gấp đôi: nó giữ bạn trong Free Tier, và “không để lại thứ gì đang chạy mà không cần thiết” là một phản xạ chuyên nghiệp mà người phỏng vấn thực sự dò hỏi. Hãy dừng lại mỗi lab ít nhất hai lần: một lần theo dàn ý, một lần từ trí nhớ. Lần dừng thứ hai là lúc kiến thức chuyển vào đôi tay bạn. Dự trữ khoảng một tuần rưỡi cho mỗi lab; dùng phần dư cho những lúc hổng hóc, vì mọi thứ sẽ hổng, và việc gỡ lỗi chúng là bài giảng hay nhất mà khóa học này không thể soạn kịch bản trước.

Trước tiên, ba ý tưởng đóng khung mọi thứ bạn sắp xây. **Region và Availability Zone**: một Region (vùng) là một cụm trung tâm dữ liệu AWS theo địa lý (chọn một cái gần bạn và ở yên trong đó — tài nguyên ở một region là vô hình từ region khác, nguồn cơn số 1 của sự bối rối “server của tôi biến đi đâu rồi?”). Một **Availability Zone (AZ — vùng khả dụng)** là một trung tâm dữ liệu tách biệt bên trong region; các hệ thống nghiêm túc chạy trên hai AZ để sự cố của một tòa nhà không kéo sập chúng. **Console vs CLI**: console (<https://console.aws.amazon.com>) là bảng điều khiển web của AWS — tuyệt vời để học và để nhìn; **AWS CLI** (aws trong terminal của bạn) làm mọi thứ console làm được, nhưng theo cách script hóa được. Bạn sẽ bắt đầu với console và tốt nghiệp lên CLI, vì Giai đoạn 2 sẽ tự động hóa mọi thứ bạn làm bằng tay ở đây.

Lab 1 (Tuần 5–6): EC2 — server đầu tiên của bạn

EC2 (Elastic Compute Cloud) cho thuê máy ảo, gọi là **instance**. Đây là hành động nguyên thủy của kỹ thuật cloud: một server Linux thật, trên internet, trong 60 giây, miễn phí.

Mục tiêu: khởi chạy một server Linux, SSH vào nó, khiến nó phục vụ một trang web cho cả thế giới, rồi hủy nó.

Dàn ý các bước: 1. Console — EC2 — Launch instance. Đặt tên cho nó. Chọn **Amazon Linux 2023** làm **AMI** (Amazon Machine Image — đĩa mẫu mà server của bạn khởi động từ đó) và **t2.micro** hoặc **t3.micro** làm **instance type** (kích cỡ; các loại này thuộc Free Tier). 2. Tạo một **key pair**; một file private key .pem được tải xuống. Đó chính là khóa SSH của bạn từ Module 1 — hãy bảo vệ nó, chmod 400 nó. 3. Trong phần cài đặt mạng, cho phép SSH (port 22) *chỉ từ “My IP”* — bây giờ bạn biết chính xác quy tắc security-group này nghĩa là gì — và cho phép HTTP (port 80) từ mọi nơi. 4. Khởi chạy, đợi trạng thái “running”, sao chép IP public, rồi từ terminal của bạn: `ssh -i mykey.pem ec2-user@<public-ip>`. Hít một hơi: bạn đang ở bên trong một máy tính trong một trung tâm dữ liệu của AWS. 5. Trên server: `sudo dnf install -y nginx && sudo systemctl start nginx && sudo systemctl enable nginx`. Rồi mở trình duyệt đến `http://<public-ip>` — đó là web server của bạn

đang trả lời thế giới. 6. Thay trang mặc định: `echo "<h1>Built by me, on EC2</h1>" | sudo tee /usr/share/nginx/html/index.html`. Làm mới trang. Chụp màn hình cho nhật ký. 7. Ngó nghiêng như một kỹ sư: `top, df -h, sudo tail -f /var/log/nginx/access.log` trong khi bạn làm mới trang — xem chính những lượt truy cập của bạn xuất hiện trong log.

Những gì bạn học được: AMI, instance type, key pair, security group trong thực chiến, SSH vào một server thật, cài đặt và chạy một dịch vụ Linux, đọc log của nó — tức là, những động tác vật lý hằng ngày của công việc.

Teardown: EC2 → chọn instance → Instance state → **Terminate**. Xác nhận nó hiển thị “terminated”. Free Tier cho 750 giờ/tháng của một micro instance, nên kể cả để nó chạy cũng chưa bị tính tiền — nhưng vẫn hãy tháo dỡ nó. Thói quen mới là điều cốt lõi.

Lab 2 (Tuần 7): S3 — những file không bao giờ biến mất

S3 (Simple Storage Service) là object storage (lưu trữ đối tượng): một chiếc xô không đáy, độ bền mười một số chín, dành cho file. Nó là câu trả lời mặc định cho “chúng ta để file ở đâu?” và, khi bị câu hình sai, là nguồn gốc của những vụ rò rỉ dữ liệu nổi tiếng nhất lịch sử — đó là lý do lab này một nửa về lưu trữ, một nửa về bảo mật.

Mục tiêu: tạo một bucket, thao tác với nó từ CLI, host một website tĩnh nho nhỏ, và hiểu chính xác “public bucket” nghĩa là gì.

Dàn ý các bước: 1. Console → S3 → Create bucket (tên bucket là duy nhất trên toàn cầu — yourname-lab-2026 là được). Lưu ý **Block Public Access** được bật mặc định. Tải lên một file bất kỳ qua console; tải nó về lại. 2. Cài AWS CLI trên máy bạn và chạy `aws configure` với một **access key** bạn tạo cho IAM user quản trị của mình (access key là một cặp tên người dùng+mật khẩu lập trình cho API — hãy đối xử với nó như một mật khẩu, không bao giờ đặt nó vào code; bạn sẽ tham quy tắc này trong Lab 5). 3. Từ terminal của bạn: `aws s3 ls ·aws s3 cp notes.txt s3://yourname-lab-2026/ ·aws s3 sync ./myfolder s3://yourname-lab-2026/backup/`. Cảm nhận sự khác biệt: console là đi thăm; CLI là làm kỹ thuật. 4. Website tĩnh: tạo một bucket thứ hai, bật static website hosting, tải lên một `index.html`, và thêm **bucket policy** public-read theo tài liệu (một văn bản quyền hạn dạng JSON — đọc từng dòng: *ai được làm gì với bucket nào*). Trang của bạn giờ đã ở trên internet mà không cần bất kỳ server nào. 5. Khám phá **storage class** (bậc lưu trữ: Standard → Infrequent Access → Glacier: rẻ hơn mỗi GB, chậm hơn/tốn hơn khi lấy lại) và đặt một **lifecycle rule** (quy tắc vòng đời — “chuyển các object sang IA sau 30 ngày”) — ném thử lần đầu vị quản lý chi phí tự động.

Những gì bạn học được: object storage vs ổ đĩa, CLI và access key, bucket policy và quyền truy cập public (một cách chủ đích, không phải vô tình), các bậc lưu trữ như một đòn bẩy chi phí.

Teardown: làm rỗng cả hai bucket, xóa cả hai bucket, và — quan trọng — vô hiệu hóa bất kỳ access key nào bạn không dùng. Hạn mức miễn phí của S3 nhỏ (5 GB) nhưng các file này chỉ vài kilobyte; kỷ luật, một lần nữa, mới là điều cốt lõi.

Lab 3 (Tuần 8-9): VPC — mạng lưới mà bạn sở hữu

VPC (Virtual Private Cloud) là lát cắt riêng, được rào chắn của bạn trong mạng của AWS. Cho đến giờ bạn dùng cái mặc định mà không để ý; kỹ sư thì tự xây cái của riêng mình, vì câu nói “*cơ sở dữ liệu nằm trong một private subnet không có đường ra internet*” là một nửa của bảo mật cloud, và bạn sắp biến nó thành sự thật bằng chính đôi tay mình.

Mục tiêu: xây một mạng hai tầng — public subnet cho một web server, private subnet cho một cơ sở dữ liệu tương lai — và chứng minh nửa private không thể chạm tới từ internet.

Dàn ý các bước: 1. Console — VPC — Create VPC. Cho nó khối địa chỉ 10.0.0.0/16 — **ký hiệu CIDR**, trong đó /16 nghĩa là “16 bit đầu cố định, phần còn lại là của tôi”: 65.536 địa chỉ private. 2. Tạo hai **subnet**: 10.0.1.0/24 (public, trong AZ-a) và 10.0.2.0/24 (private, trong AZ-b). Một subnet là một khối nhỏ hơn bên trong VPC, sống trong đúng một AZ. 3. Tạo một **Internet Gateway** (cánh cửa ra internet của VPC) và gắn nó vào. Tạo một **route table** (bảng định tuyến) với quy tắc 0.0.0.0/0 → internet gateway (“bất cứ thứ gì không phải nội bộ thì đi ra cửa internet”) và liên kết nó với *chỉ* public subnet. Private subnet chỉ giữ route nội bộ — chính sự vắng mặt của route đó là sự bảo mật. 4. Khởi chạy một micro EC2 instance trong mỗi subnet (cái public có IP public, cái private thì không). 5. Phân chứng minh: SSH vào instance public — hoạt động. Thử địa chỉ của instance private từ laptop của bạn — treo mãi mãi, và giờ bạn có thể nói chính xác vì sao. Sau đó SSH từ instance public vào instance private (nó truy cập được từ bên trong VPC): chiếc máy public đang đóng vai một **bastion host** (máy chủ bàn đạp), một mẫu hình chuẩn công nghiệp mà bạn vừa khám phá ra bằng cách tự xây nó. 6. Khái niệm bonus để tra cứu và ghi nhớ: một **NAT gateway** cho phép các máy private vươn ra ngoài (để cập nhật) trong khi vẫn không thể bị chạm tới từ bên ngoài — nhưng nó tính tiền theo giờ, nên hãy đọc về nó, đừng xây nó.

Những gì bạn học được: CIDR, subnet, route table, internet gateway, sự phân chia public/private, bastion host — chính xác cái gạch đầu dòng về mạng trong mọi JD cloud, dưới dạng trí nhớ cơ bắp.

Teardown: terminate cả hai instance trước, rồi xóa VPC (thao tác này quét sạch cả subnet, route table và gateway theo nó). Kiểm tra trong EC2 rằng không còn gì hiển thị “running”.

Lab 4 (Tuần 10): RDS — một cơ sở dữ liệu bạn không phải trông nom

RDS (Relational Database Service) là một cơ sở dữ liệu được quản lý: AWS chạy engine cơ sở dữ liệu (PostgreSQL, MySQL...) và lo việc sao lưu, vá lỗi và failover (chuyển đổi dự phòng), trong khi bạn sở hữu dữ liệu và các truy vấn. “Managed service” (dịch vụ được quản lý) là món hời cốt lõi của cloud — đánh đổi một phần quyền kiểm soát lấy rất nhiều lao động không tạo khác biệt — và lab này là nơi bạn cảm nhận món hời đó.

Mục tiêu: khởi chạy một cơ sở dữ liệu PostgreSQL trong một private subnet, kết nối đến nó từ một EC2 instance, và hiểu về backup và multi-AZ — rồi tháo dỡ nó ngay, vì RDS là lab dễ bị bỏ quên đang chạy nhất.

Dàn ý các bước: 1. Dừng lại nhanh VPC của Lab 3 (lần dựng thứ hai từ trí nhớ — đây là spaced repetition có chủ đích), thêm một private subnet thứ hai ở một AZ khác, vì RDS yêu cầu một **subnet group** trải trên hai AZ. 2. Console — RDS — Create database — PostgreSQL —

Free tier template (mẫu này chọn sẵn db.t3.micro/db.t4g.micro, single-AZ). Đặt mật khẩu master. Đặt nó trong VPC của bạn, **Public access: No**, trong một security group cho phép port 5432 *chỉ từ security group của web server* — một quy tắc tham chiếu đến một quy tắc khác thay vì một IP. Thanh lịch, và chuẩn mực. 3. Khởi chạy một micro EC2 trong public subnet, cài client postgresql, và kết nối: `psql -h <rds-endpoint> -U postgres`.

Endpoint là một tên DNS, không phải một IP — AWS có thể di dời cỗ máy bên dưới, và cái tên đi theo nó. (Module 2 đã bắt đầu sinh lời.) 4. Tại dấu nhắc `psql`: tạo một bảng, chèn ba hàng, truy vấn chúng ra lại. Hôm nay bạn không cần chiều sâu SQL; bạn cần đã từng chạm vào nó. 5. Tham quan, đừng bật: cài đặt **automated backups** (sao lưu tự động — khôi phục theo thời điểm từ các snapshot hằng đêm + log), và tùy chọn **Multi-AZ** (một bản standby sống ở AZ khác với **failover** tự động — chi phí gần gấp đôi, đó là lý do prod nói có còn lab nói không). Chụp một **snapshot** thủ công, tìm nó trong console, hiểu rằng bạn có thể khôi phục một bản sao từ nó. 6. Câu hỏi nhật ký đáng mười phút: chính xác thì AWS đang làm gì cho bạn ở đây mà nếu không bạn sẽ phải tự làm lúc 2 giờ sáng? (Vá lỗi, sao lưu, failover, phân cứng.) Câu trả lời đó chính là câu trả lời phỏng vấn về managed services.

Những gì bạn học được: cơ sở dữ liệu được quản lý, subnet group, quy tắc security-group đến security-group, endpoint, snapshot, multi-AZ/failover — cộng thêm một màn trình diễn sống động về việc trả giá bằng quyền kiểm soát để mua độ tin cậy.

Teardown: xóa RDS instance (từ chối snapshot cuối cùng với một lab; lưu ý rằng prod thì sẽ chụp một cái), xóa snapshot thủ công (snapshot tính tiền lưu trữ!), terminate EC2, xóa VPC. Kiểm tra bảng điều khiển billing của bạn vào ngày hôm sau — đọc nó hằng tuần là một thói quen của Giai đoạn 3 bắt đầu từ bây giờ.

Lab 5 (Tuần 11-12): IAM — ai được làm gì

IAM (Identity and Access Management — quản lý danh tính và quyền truy cập) quyết định người nào và chương trình nào được làm gì với tài nguyên nào. Nó không tính tiền, không cấp phát gì — và nó là dịch vụ bị kiểm toán nhiều nhất, bị phỏng vấn dò hỏi nhiều nhất, dính líu đến các vụ rò rỉ nhiều nhất trong AWS. Các gạch đầu dòng bảo mật trong JD (“least-privilege access controls”, “IAM policies”) nghĩa là lab này.

Mục tiêu: tạo user, group, policy, và — cái quan trọng nhất — một **role**, rồi thám nhuần least privilege (đặc quyền tối thiểu) bằng cách cảm nhận AWS từ chối bạn.

Dàn ý các bước: 1. Khái niệm trước, năm phút: một **user** là danh tính cho một con người hoặc chương trình; một **group** gom các user lại; một **policy** là một văn bản JSON cấp quyền (“cho phép `s3:GetObject` trên `arn:aws:s3:::my-bucket/*`”); một **role** là một danh tính có policy nhưng *không có mật khẩu* mà một bên được tin cậy tạm thời *đảm nhận* (*assume*) — cách server và dịch vụ nhận quyền mà không cần bất kỳ bí mật nào được lưu trữ. 2. Tạo một user `readonly-rita` trong một group với policy `ReadOnlyAccess` do AWS quản lý. Đăng nhập bằng cô ấy trong một cửa sổ trình duyệt riêng tư: cô ấy thấy được mọi thứ, nhưng mọi nút tạo/xóa đều thất bại với một lời từ chối rõ ràng. Đọc trọn vẹn một trong các lỗi đó — học cách phân tích “not authorized to perform X on Y” là một kỹ năng công việc hằng ngày. 3. Viết custom policy đầu tiên của bạn trong trình soạn JSON: cho phép `s3:ListBucket` và `s3:GetObject` trên một bucket cụ thể. Gắn nó cho một user mới; xác minh cô ấy đọc được bucket đó và *không gì khác*. Bạn vừa triển khai least privilege, chứ không chỉ định nghĩa nó. 4.

Role, phần thưởng chính: tạo một role cho EC2 với AmazonS3ReadOnlyAccess, khởi chạy một micro instance có gắn role đó, SSH vào, và chạy `aws s3 ls` — nó hoạt động mà **không có access key nào ở bất kỳ đâu trên máy**. Instance đang đảm nhận role và tự động nhận các thông tin xác thực ngắn hạn. Đây là mẫu hình bảo mật quan trọng nhất trong AWS: *role cho máy móc, không bao giờ để key trên máy*. Nói câu đó hai lần. 5. Kiểm toán tài khoản của chính bạn như một người chuyên nghiệp: root đã có MFA chưa (Tuần 3)? Có access key nào già hơn 90 ngày không? Có user nào có nhiều quyền hơn mức họ dùng không? Chạy **credential report** của IAM và đọc nó. Nghi thức mười phút này, mỗi tháng, chính là danh sách vệ sinh IAM của Giai đoạn 3 đang ra đời.

Những gì bạn học được: user/group/policy/role, đọc và viết policy JSON, least privilege được kiểm chứng bằng thí nghiệm, role-chức-không-phải-key, và cuộc kiểm toán bảo mật đầu tiên của bạn.

Teardown: xóa các user thử nghiệm và thông tin xác thực của họ, terminate instance, giữ kiến thức về role mãi mãi.

Từ vựng cho Giai đoạn 1 (một bảng, cả năm lab):

Thuật ngữ	Định nghĩa
Region / Availability Zone	Cụm trung tâm dữ liệu AWS theo địa lý / một trung tâm dữ liệu tách biệt bên trong nó. Các hệ thống nghiêm túc trải trên hai AZ.
EC2 / instance	Dịch vụ cho thuê máy ảo / một server đang thuê.
AMI	Amazon Machine Image — đĩa mẫu mà một instance khởi động từ đó.
Instance type	Kích cỡ/cấu hình bạn chọn (t3.micro = nhỏ xíu, Free Tier).
Security group	Tường lửa theo-từng-tài-nguyên: những port nào, từ những nguồn nào.
Key pair	Cặp khóa SSH riêng/công khai để tiếp cận instance của bạn.
S3 / bucket / object	Lưu trữ đối tượng / một thùng chứa có tên / một file được lưu.
Storage class / lifecycle rule	Bậc giá-tốc độ cho object / quy tắc tự động chuyển chúng sang bậc rẻ hơn theo tuổi.
Bucket policy	Văn bản JSON nêu rõ ai được làm gì với một bucket. Những cái public thì lên mặt báo.
AWS CLI / access key	Giao diện terminal với AWS / thông tin xác thực lập trình cho nó (đôi xử như mật khẩu).
VPC / subnet	Lát mạng riêng của bạn / một khối nhỏ hơn của nó sống trong một AZ, public hoặc private.

Thuật ngữ	Định nghĩa
CIDR	Ký hiệu khối địa chỉ: $10.0.0.0/16$ = “16 bit đầu cố định, 65.536 địa chỉ là của tôi.”
Internet gateway / route table	Cửa ra internet của VPC / các quy tắc quyết định lưu lượng được gửi đi đâu. Không có route = không thể chạm tới = an toàn.
NAT gateway	Cho phép máy private gọi ra ngoài mà không thể bị gọi vào. Tính tiền theo giờ — hãy biết nó, đừng để nó chạy không.
Bastion host	Chiếc máy public được gia cố mà bạn SSH xuyên qua để tới các máy private.
RDS / endpoint	Dịch vụ cơ sở dữ liệu quan hệ được quản lý / tên DNS mà bạn kết nối đến.
Snapshot / Multi-AZ / failover	Bản sao tại-một-thời-điểm / bản standby sống ở AZ thứ hai / cú chuyển tự động sang nó.
IAM user / group / policy / role	Danh tính / bó danh tính / văn bản JSON cấp quyền / danh tính đảm nhận được, không có mật khẩu — cách máy móc nhận quyền.
Least privilege	Quy tắc vàng: quyền tối thiểu đủ làm việc, không hơn.
Managed service	AWS gánh phân lao động không tạo khác biệt (và lỗi, sao lưu, failover); bạn giữ dữ liệu và các quyết định.

Video cho giai đoạn này:

Video	Kênh	Thời lượng	Liên kết
Getting Started with EC2	AWS Developers (official)	27 phút	https://www.youtube.com/watch?v=nJ-djerESW0
Introduction to Amazon S3	Amazon Web Services (official)	~5 phút	https://www.youtube.com/watch?v=ecv-19sYL3w
AWS Networking Basics — VPC & Subnets	KodeKloud (xem lại sau Lab 3 — giờ nó thâm theo cách khác)	~30 phút	https://www.youtube.com/watch?v=QM63dyA_4Pc
The AWS Shared Responsibility Model	Digital Cloud Training	4 phút	https://www.youtube.com/watch?v=ESPBBEK-

Video	Kênh	Thời lượng	Liên kết
			cvo

Nói thành tiếng: “It’s in a private subnet; there’s no route to the internet gateway.” (Nó nằm trong private subnet; không có route nào đến internet gateway.) · “The security group only allows 5432 from the web tier’s security group.” (Security group chỉ cho phép 5432 từ security group của tầng web.) · “The instance uses a role — there are no keys on the box.” (Instance dùng một role — không có key nào trên máy.) · “Terminate it when you’re done; nothing idles in this account.” (Terminate nó khi bạn xong việc; không thứ gì chạy không trong tài khoản này.)

Milestone — kết thúc Giai đoạn 1: màn dựng thử thách. Trong một lần ngồi, từ trí nhớ: VPC với public/private subnet — EC2 web server (public, có gắn role, phục vụ một trang) — RDS PostgreSQL (private, chỉ tiếp cận được từ web server) — một S3 bucket mà instance đọc được qua role của nó — rồi tháo dỡ toàn bộ cho sạch. Dưới ba giờ nghĩa là bạn sẵn sàng cho Giai đoạn 2. Ngoài ra: **bắt đầu ôn CLF-C02 ngay bây giờ** (khóa học đầy đủ của freeCodeCamp — <https://www.youtube.com/watch?v=7HKot-brXFE> — ở tốc độ 1.25× như một lượt ôn tập; bản kinh điển 14 giờ là <https://www.youtube.com/watch?v=NhDYbskXRgc>), và thi **AWS Cloud Practitioner** vào khoảng Tuần 14–16.

GAJI ĐOẠN 2 — TỰ ĐỘNG HÓA (Tuần 13–20)

Ý tưởng của giai đoạn này: mọi thứ bạn đã nhấp chuột trong Giai đoạn 1, giờ bạn sẽ làm bằng code. Lời châm biếm nhẹ của ngành dành cho việc nhấp chuột trong console là **ClickOps**; cụm từ trong bản mô tả công việc cho thứ thay thế nó là “*design, develop, and deploy cloud infrastructure using infrastructure as code*” (thiết kế, phát triển và triển khai hạ tầng cloud bằng hạ tầng dưới dạng code). Giai đoạn này là sự khác biệt giữa một người đã từng dùng AWS và một người mà công ty sẽ trả tiền để vận hành nó.

Module 6 (Tuần 13–15): Scripting — Bash và Python cho vận hành, cộng thêm Git

Bash scripting là việc đặt các câu lệnh của Module 1 vào một file để máy tính lặp lại chúng một cách hoàn hảo. Học, theo thứ tự này: **biến** (NAME="web-1"), **command substitution** (thay thế lệnh — TODAY=\$(date +%F)), **if** (if [-f "\$FILE"]; then ... fi), **vòng lặp** (for f in *.log; do gzip "\$f"; done), **exit code** (mã thoát — mọi lệnh trả về 0 khi thành công, khác 0 khi thất bại; && nối chuỗi khi thành công — script ra quyết định bằng thứ này), và đọc **đối số** (\$1, \$2). Đó là 90% Bash vận hành thực tế. Hãy viết script thực sự hữu ích đầu tiên của bạn trong tuần này: **backup.sh** — nén tar một thư mục, đặt tên file nén theo ngày hôm nay, **aws s3 cp** nó lên một bucket, xóa các file nén cục bộ cũ hơn 7 ngày, và **echo** một dòng thành công/thất bại vào một file log. Một script đó gói cả biến, thay thế lệnh, điều kiện, exit code và CLI trong một sản phẩm — và là một gạch đầu dòng trên résumé của bạn (“automated backups to S3” — tự động hóa sao lưu lên S3).

Python là người anh lớn của Bash: tốt hơn cho bất cứ thứ gì liên quan đến logic, dữ liệu, hoặc nói chuyện với API. Bạn cần Python vận hành đủ dùng, không phải chiều sâu kỹ sư phần mềm: biến và kiểu, list và dictionary, if/for, hàm, đọc/ghi file, xử lý lỗi try/except, và cài thư viện bằng pip. Rồi làm quen với **boto3**, thư viện AWS cho Python: `import boto3; ec2 = boto3.client("ec2"); ec2.describe_instances()` — đột nhiên kiến thức Giai đoạn 1 của bạn trở nên lập trình được. Viết `audit.py`: liệt kê mọi EC2 instance trong tài khoản với tên, loại, trạng thái và thời điểm khởi chạy, và in một dòng cảnh báo cho bất cứ thứ gì chạy lâu hơn 24 giờ. Chúc mừng — bạn vừa viết một công cụ kiểm soát chi phí thực thụ mà nhà tuyển dụng trả tiền cho nó.

Git là hệ thống quản lý phiên bản mà mọi công việc kỹ thuật sống trong đó. Khái niệm: một **repository** (một thư mục mà toàn bộ lịch sử của nó được ghi lại), một **commit** (một ảnh chụp được lưu kèm một thông điệp), một **branch** (một nhánh công việc song song), một **remote** (bản sao trên GitHub), và một **pull request** (yêu cầu để một branch được review và hợp nhất). Vòng lặp hằng ngày là sáu lệnh: `git init / git clone, git status, git add, git commit -m "message", git push, git pull`. Hãy tạo tài khoản GitHub ngay hôm nay, tạo một repo tên `cloud-journey`, và từ giờ trở đi **mọi script và config bạn viết trong khóa học này đều được commit vào đó**. Mười tháng nữa, lịch sử commit đó là bằng chứng công khai, có ngày tháng, cho mọi điều khóa học này tuyên bố bạn làm được.

Từ vựng:

Thuật ngữ	Định nghĩa
Script	Một file chứa các lệnh chạy từ trên xuống dưới — nguyên tử của tự động hóa.
Variable / argument	Một giá trị có tên trong script / một giá trị được truyền vào khi bạn chạy nó (\$1).
Exit code	Tín hiệu thành công (0) hoặc thất bại (khác 0) của mọi lệnh — cách script ra quyết định.
Cron	Bộ lập lịch của Linux: chạy một script mỗi đêm lúc 2 giờ sáng, mãi mãi.
Python / pip	Ngôn ngữ lập trình được giới vận hành ưà chuộng nhất / trình cài đặt gói của nó.
boto3	Thư viện Python để điều khiển AWS — cái console, dưới dạng code.
try/except	“Thử làm cái này; nếu thất bại, làm cái kia thay thế” của Python — cách script thất bại một cách nhẹ nhàng.
Git / repository / commit	Quản lý phiên bản / lịch sử được ghi lại của một dự án / một ảnh chụp được lưu kèm thông điệp.
Branch / merge	Một nhánh công việc song song / đưa nó trở lại dòng chính.
GitHub / remote / pull request	Trang hosting / bản sao cloud của repo bạn / một yêu cầu hợp nhất branch được

Thuật ngữ	Định nghĩa
README	review và thảo luận. Tài liệu trang đầu của một repo giải thích nó là gì và chạy nó ra sao. Nhà tuyển dụng đọc những thứ này.

Video cho module này:

Video	Kênh	Thời lượng	Liên kết
Learn Python — Full Course for Beginners	freeCodeCamp	~4,5 giờ (xem rải trong 3 tuần)	https://www.youtube.com/watch?v=rpscVS0vtbw

Nói thành tiếng: “I’ll script it — it’ll be wrong once, then never again.” (Tôi sẽ viết script cho nó — nó sẽ sai một lần, rồi không bao giờ nữa.) · “Check the exit code before the next step runs.” (Kiểm tra exit code trước khi bước tiếp theo chạy.) · “Commit it with a message that says why, not what.” (Commit nó với một thông điệp nói vì sao, không phải cái gì.) · “It’s in the repo — pull the latest.” (Nó ở trong repo — pull bản mới nhất về.)

Bài tập: (1) `backup.sh`, như mô tả ở trên, được lập lịch chạy hằng đêm bằng cron trên một EC2 instance lab. (2) `audit.py`, như mô tả ở trên. (3) Cả hai được commit vào `cloud-journey` với một README giải thích từng cái. (4) Có tình làm hỏng script của chính bạn (một đường dẫn sai, một bucket không tồn tại) và làm cho nó thất bại *một cách ôn ào và rõ ràng* — xử lý lỗi là một ngôn ngữ tình yêu của dân vận hành.

Milestone: script backup của bạn sống sót khi bị chạy hai lần liên tiếp và khi thiếu thư mục (không sập, thông điệp rõ ràng); audit Python của bạn chạy trên tài khoản thật của bạn; repo của bạn cho thấy một tuần commit.

Module 7 (Tuần 16-17): Terraform — hạ tầng dưới dạng code

Ý tưởng lớn: thay vì nhập chuột để các tài nguyên ra đời, bạn viết các file văn bản *khai báo* những gì cần tồn tại — “một VPC, hai subnet, một instance, những tag này” — và **Terraform** làm cho AWS khớp với file đó. Declarative (khai báo), không phải imperative (mệnh lệnh): bạn nêu đích đến, không phải các khúc rẽ. Vì sao mọi JD đòi hỏi nó: các file sống trong Git, nên hạ tầng trở nên **được review** (pull request trước khi thay đổi), **lập lại được** (cùng một file dựng dev, staging và prod giống hệt nhau), **đảo ngược được** (quay lui bằng cách revert commit), và **kiểm toán được** (lịch sử nói ai đã thay đổi gì, khi nào, vì sao). Console là một xưởng thủ công; Terraform là một nhà máy.

Vòng lặp cốt lõi bạn sẽ chạy hàng trăm lần: `terraform init` (tải về **provider** AWS — plugin dịch các file của bạn thành các lời gọi API) → `terraform plan` (một lượt chạy thử in ra chính xác những gì sẽ được tạo/thay đổi/hủy — đọc từng dòng; kỹ sư nào apply mà không đọc plan là người gây ra sự cố) → `terraform apply` (biến nó thành thật) → `terraform destroy` (gỡ bỏ nó — tháo dỡ bằng một lệnh duy nhất, mà đến giờ hẳn bạn sẽ thấy là đẹp đẽ). Terraform theo dõi những gì nó đã dựng trong một **state file** (file trạng thái) — trí nhớ của nó về thực tại; đánh mất nó hoặc tự tay sửa thực tại sau lưng nó (“drift” —

trôi dạt) là đau khổ theo sau. Tài nguyên được khai báo bằng **HCL**, một ngôn ngữ cấu hình dễ đọc:

```
resource "aws_instance" "web" {
  ami           = "ami-0abcdef1234567890"
  instance_type = "t3.micro"
  tags = { Name = "web-1", Project = "lab" }
}
```

Lab (đây chính là module): dừng lại màn thử thách của Giai đoạn 1 bằng Terraform, theo từng bước. (1) Khởi tạo một repo terraform-labs; viết config cho chỉ một S3 bucket có gắn tag; plan, đọc, apply, kiểm tra console — nó ở đó; destroy. (2) Mở rộng config: VPC, hai subnet, internet gateway, route table — bản dựng Lab 3, giờ là ~60 dòng HCL. (3) Thêm security group và một EC2 instance với `variables.tf` (region, kích cỡ instance) và `outputs.tf` (in ra IP public sau khi apply). (4) Đổi instance type trong file và apply lại — xem Terraform tính ra khác biệt và chỉ thay đổi *đúng chỗ đó*. Hành vi dựa-trên-diff này là toàn bộ phép màu. (5) Destroy mọi thứ, xác nhận console trống, commit, và gắn tag repo v1.0. Tài liệu tham khảo nằm tại <https://developer.hashicorp.com/terraform> — hãy bookmark nó; đọc tài liệu provider là một nửa công việc Terraform thực tế.

Từ vựng:

Thuật ngữ	Định nghĩa
IaC (Infrastructure as Code)	Khai báo hạ tầng trong các file văn bản có quản lý phiên bản mà một công cụ biên thành hiện thực.
Terraform / HCL	Công cụ IaC đa-cloud thông trị / ngôn ngữ cấu hình của nó.
Provider	Plugin dịch các file của bạn thành các lời gọi API của một cloud (ở đây: AWS).
plan / apply / destroy	Bản diff chạy thử / biên thành thật / gỡ bỏ tất cả. Đọc mọi plan.
State file	Bản ghi của Terraform về những gì nó đã dựng — trí nhớ của nó về thực tại. Bảo vệ nó.
Drift	Thực tại thay đổi sau lưng Terraform (ai đó đã nhập chuột). Kẻ thù.
Variable / output / module	Một đầu vào config / một kết quả được in ra (một IP, một URL) / một khối config đóng gói tái sử dụng được.
Declarative vs imperative	Nêu đích đến vs viết kịch bản các khúc rẽ. Terraform là declarative.
CloudFormation	Dịch vụ IaC riêng của AWS — cùng ý tưởng, chỉ cho AWS. JD chấp nhận cả hai; học Terraform trước.

Video cho module này:

Video	Kênh	Thời lượng	Liên kết
Terraform explained in 15 mins	TechWorld with Nana	18 phút	https://www.youtube.com/watch?v=l5klai_GBDE
Terraform Course — Automate your AWS cloud infrastructure	freeCodeCamp	~2,5 giờ	https://www.youtube.com/watch?v=SLB_c_ayRMo

Nói thành tiếng: “Nothing changes in prod except through Terraform.” (Không gì thay đổi trong prod ngoại trừ thông qua Terraform.) · “Show me the plan before you apply.” (Cho tôi xem plan trước khi bạn apply.) · “That was clicked in manually — that’s drift; let’s import it or remove it.” (Cái đó được nhập chuột tạo thủ công — đó là drift; hãy import nó hoặc gỡ nó đi.) · “It’s in the module; reuse it, don’t rewrite it.” (Nó ở trong module; tái sử dụng nó, đừng viết lại.)

Bài tập: lab ở trên, cộng thêm: (1) cô tình tạo drift — đổi tag của instance bằng tay trong console, chạy terraform plan, và xem Terraform nhận ra; ghi nhật ký những gì nó đề xuất. (2) Viết một đoạn văn tiếng Anh giản dị trong README của repo: “why this beats clicking” (vì sao cách này thắng nhập chuột). Nếu bạn không viết được, bạn chưa nắm được nó.

Milestone: từ một thư mục trống, bạn có thể dựng lên cả stack VPC + instance bằng init/plan/apply và gỡ nó bằng destroy, vừa làm vừa giải thích thành tiếng mỗi lệnh đang làm gì — không cần ghi chú.

Module 8 (Tuần 18–20): CI/CD, Docker, và cái nhìn đầu tiên về Kubernetes

CI/CD (Continuous Integration / Continuous Delivery — tích hợp liên tục / chuyển giao liên tục) là bằng chuyên robot gắn vào repo Git của bạn: với mỗi lần push, nó tự động kiểm tra, kiểm thử, build, và — khi được cấu hình — triển khai thay đổi của bạn. Mục đích là những bản phát hành nhỏ, thường xuyên, nhằm chần thay vì những bản hiếm hoi, đáng sợ. **GitHub Actions** là hệ thống CI/CD tích hợp sẵn trong GitHub: một **workflow** là một file YAML trong .github/workflows/ nói rằng “khi push, chạy các bước này trên một **runner** mới tinh (một VM tạm thời)”. Lab: thêm vào terraform-labs một workflow chạy terraform fmt -check và terraform validate với mỗi lần push — push một file có tình sai định dạng và xem dấu ổ đũa, sạ nó và xem dấu ở xanh. Bộn giở đã có một pipeline; dù nhẽ bé, nó cùng loài với những pipeline trong mọi JD. (Bước thứ hai, trong Dự án 1: một workflow chạy terraform plan trên các pull request, để những thay đổi hạ tầng được đề xuất hiển thị diff của chúng ngay trong phần review.)

Docker đóng gói một ứng dụng cùng mọi thứ nó cần vào một **container** — một hộp cơm niêm phong chạy giống hệt nhau trên laptop của bạn, trên EC2, và ở mọi nơi khác, chấm dứt đại dịch cổ xưa “works on my machine” (trên máy tôi chạy được mà). Một **image** là công thức đóng băng (được build từ một **Dockerfile**, một file văn bản ngắn gồm các bước build: bắt đầu FROM một image nền, COPY ứng dụng của bạn vào, định nghĩa lệnh khởi động); một

container là một thực thể đang chạy của một image; một **registry** (Docker Hub, hay **ECR** của AWS) là nơi image được push lên và pull về. Container *không phải* là VM: chúng chia sẻ Linux kernel của máy chủ, nên chúng khởi động trong khoảng một giây và bạn có thể chạy hàng chục cái trên một micro instance. Lab: trên một EC2 instance, cài Docker, `docker run hello-world`, rồi `docker run -d -p 80:8080 <a sample web image>` và mở trình duyệt đến nó; sau đó viết một Dockerfile nằm dòng phục vụ trang tĩnh Lab 2 của bạn từ một image nền nginx, build nó, chạy nó, và push nó lên Docker Hub. Học các động từ hằng ngày: `docker ps`, `docker logs`, `docker exec -it <id> bash` (một shell *bên trong* container), `docker stop`.

Kubernetes (K8s) — với khóa học này, là một kỹ năng ở mức đọc-hiểu, được dán nhãn trung thực. Khi một công ty chạy hàng trăm container trên nhiều máy, phải có thứ gì đó lập lịch cho chúng, khởi động lại cái bị sập, mở rộng cái đang bận, và định tuyến lưu lượng giữa chúng: bộ điều phối đó là Kubernetes. Học bản đồ khái niệm ngay bây giờ — một **cluster** gồm các **node** chạy các **pod** (đơn vị triển khai nhỏ nhất, thường là một container); một **deployment** khai báo “giữ 3 bản sao của pod này luôn sống” và cluster liên tục biến điều đó thành sự thật (lại là declarative — Kubernetes là triết lý của Terraform áp dụng vào phần mềm đang chạy); một **service** cho các pod một địa chỉ ổn định. Sản phẩm được quản lý của AWS là **EKS**. Các JD junior muốn chính xác mức độ hiểu biết này cộng với sự thành thạo Docker; kỹ năng vận hành K8s thực thụ (và chứng chỉ CKA) là một mục tiêu mạnh mẽ cho Năm 2, không phải Tháng 5. Khóa học này nói thẳng với bạn cái nào là cái nào.

Từ vựng:

Thuật ngữ	Định nghĩa
CI/CD	Pipeline tự động kiểm tra, build, kiểm thử và giao mọi thay đổi.
GitHub Actions / workflow / runner	CI/CD tích hợp sẵn của GitHub / file YAML định nghĩa một pipeline / VM tạm thời thực thi nó.
YAML	Định dạng cấu hình dựa trên thụt đầu dòng của pipeline và Kubernetes. Thụt dòng là ý nghĩa — hãy cẩn thận với nó.
Docker / image / container	Bộ công cụ container / công thức đóng băng, xếp lớp / một thực thể đang chạy của nó.
Dockerfile	File văn bản ngắn gồm các bước để build một image.
Registry / ECR	Nơi image được push và pull / registry của AWS.
Kubernetes (K8s) / cluster / node	Bộ điều phối container / nhóm máy của nó / một máy trong đó.
Pod / deployment / service	Đơn vị triển khai nhỏ nhất / “giữ N bản sao luôn sống,” được thực thi liên tục / một địa chỉ ổn định đứng trước các pod.
EKS	Control plane Kubernetes được quản lý của

Thuật ngữ	Định nghĩa
Rollback	AWS. Quay nhanh về phiên bản trước khi một bản phát hành trở chứng — tấm lưới an toàn mà CI/CD biến thành chuyện thường ngày.

Video cho module này:

Video	Kênh	Thời lượng	Liên kết
DevOps CI/CD Explained in 100 Seconds	Fireship	2 phút	https://www.youtube.com/watch?v=scEDHsr3APg
What is DevOps? REALLY understand it	TechWorld with Nana	~15 phút	https://www.youtube.com/watch?v=OyWAtQ6wYNM
Docker in 100 Seconds	Fireship	2 phút	https://www.youtube.com/watch?v=Gjnup-PuquQ
Docker Tutorial for Beginners [FULL COURSE in 3 Hours]	TechWorld with Nana	3 giờ	https://www.youtube.com/watch?v=3c-iBn73dDE
Kubernetes explained in 15 mins	TechWorld with Nana	~16 phút	https://www.youtube.com/watch?v=VnvRFRk_51k

Nói thành tiếng: “Don’t merge until the pipeline is green.” (Đừng merge cho đến khi pipeline xanh.) · “It’s containerized — same image in dev and prod.” (Nó đã được container hóa — cùng một image ở dev và prod.) · “Exec into the container and check its logs.” (Exec vào container và kiểm tra log của nó.) · “K8s keeps three replicas up; kill one and watch it come back.” (K8s giữ ba bản sao luôn chạy; giết một cái và xem nó quay trở lại.)

Bài tập: hai lab ở trên, cộng thêm: (1) ghi nhật ký một đoạn văn trả lời “VM vs container vs serverless — khi nào dùng cái nào?” (video Serverless in 100 Seconds của Fireship — https://www.youtube.com/watch?v=W_VV2Fx32_Y — bổ sung lựa chọn thứ ba). (2) Terminate instance của lab Docker; xác nhận các image của bạn vẫn sống trong registry — nhận ra rằng *sản phẩm* giờ đây sống lâu hơn *server*, đó chính là toàn bộ thế giới quan triển khai hiện đại gói trong một quan sát.

Milestone — kết thúc Giai đoạn 2: GitHub của bạn cho thấy: một repo các script vận hành Bash + Python hoạt động được, một repo Terraform dựng lên và hủy đi một stack thật, một workflow Actions xanh, và một Dockerfile đã được push lên registry — và bạn có thể giải

thích mọi file trong một buổi phỏng vấn. Hồ sơ GitHub đó không còn là của một sinh viên nữa. Nó là của một kỹ sư junior.

GIẢI ĐOẠN 3 — VẬN HÀNH (Tuần 21-28)

Ý tưởng của giai đoạn này: xây dựng hệ thống giúp bạn được tuyển; *vận hành* chúng mới là công việc thực sự. Các gạch đầu dòng JD mà giai đoạn này trả lời là nửa vận hành — “monitor infrastructure health” (giám sát sức khỏe hạ tầng), “participate in incident response, including log analysis” (tham gia ứng phó sự cố, gồm phân tích log), “identifying, analyzing, and resolving infrastructure vulnerabilities” (nhận diện, phân tích và xử lý các lỗ hổng hạ tầng), “manage cloud costs” (quản lý chi phí cloud). Một tuần của kỹ sư cloud phần lớn là giai đoạn này.

Module 9 (Tuần 21-24): Giám sát, log, và sự cố

Giám sát — biết trước cả người dùng. **CloudWatch** là dịch vụ observability (khả năng quan sát) tích hợp sẵn của AWS. Ba khái niệm nguyên thủy: **metric** (những con số theo thời gian — CPU %, đĩa %, số lượng request, số lượng lỗi), **alarm** (một quy tắc theo dõi một metric: “nếu CPU > 80% trong 5 phút — thông báo”), và **dashboard** (các metric xếp trên một màn hình). Thông báo chảy qua **SNS** (Simple Notification Service — một topic bạn publish vào, những người đăng ký được email/nhắn tin gọi). Cái tinh tế của nghề là alarm về cái gì: chỉ gọi con người cho những gì cần con người, nếu không mọi người sẽ học cách phớt lờ máy nhắn tin (**alert fatigue** — sự mệt mỏi vì báo động — chế độ thất bại đã đi trước nhiều vụ sập nổi tiếng). Bốn tín hiệu vàng đáng học thuộc: **latency, traffic, errors, saturation** (độ trễ, lưu lượng, lỗi, độ bão hòa) — chậm bao nhiêu, bận bao nhiêu, hỏng bao nhiêu, đầy bao nhiêu.

Log — biết vì sao. Metric nói có gì đó sai; log nói điều gì đã xảy ra. **CloudWatch Logs** tập trung chúng lại: một **agent** trên mỗi instance chuyển các file như `/var/log/nginx/access.log` về các **log group**, nơi **Logs Insights** cho phép bạn truy vấn xuyên qua các máy (“đếm các phản hồi 5xx theo từng phút trong một giờ qua”). Tập trung hóa quan trọng vì server giờ đây là thứ dùng xong bỏ đi (bạn đã chứng minh điều đó ở Module 8) — log phải sống lâu hơn những cỗ máy đã viết ra chúng.

Lab (Tuần 21-22): dựng một web server nhỏ có giám sát, tất cả bằng Terraform (stack của bạn từ Module 7, mở rộng thêm): EC2 + nginx + CloudWatch agent; một SNS topic gửi email cho bạn; một alarm về CPU cao và một cái khác về instance status-check thất bại. Rồi tự tấn công mình: SSH vào và chạy một trình đốt CPU (`yes > /dev/null` & vài lần); xem metric leo lên, alarm kích hoạt, email bay đến; giết các tiến trình; xem nó hồi phục. Sau đó truy vấn chính access log của bạn trong Logs Insights. Bạn vừa chứng kiến trọn vòng lặp phát hiện—thông báo—chẩn đoán—xử lý trên một hệ thống do chính bạn xây.

Ứng phó sự cố — nửa con người. Một **incident** (sự cố) là một trạng thái ngoài kế hoạch “hệ thống không ổn”; các mức **severity** (độ nghiêm trọng — sev-1 = khách hàng sập, tất cả vào cuộc) đặt quy mô ứng phó. Vòng lặp chuyên nghiệp: **phát hiện** (lý tưởng là từ alarm, không phải email của khách hàng) — **triage** (phân loại — tệ đến đâu, cần ai) — **mitigate**

(giảm nhẹ — cầm máu *trước*; quay lui, khởi động lại, failover; nguyên nhân gốc để sau) — **resolve** (xử lý dứt điểm) — **post-mortem**: một bản rà soát viết ra, *không đổ lỗi*, về điều gì đã xảy ra, vì sao, và điều gì sẽ ngăn lặp lại. Không đổ lỗi không phải là mềm yếu; đó là kỹ thuật: người bị trừng phạt sẽ giấu thông tin, và thông tin bị giấu gây ra sự cố lặp lại. **On-call** là vòng xoay ai mang máy nhắn tin; JD nhắc đến nó, người phỏng vấn hỏi về nó, và câu trả lời trung thực của bạn sau module này là “tôi đã mô phỏng nó và tôi biết vòng lặp.”

Runbook — gạch đầu dòng về tài liệu, được cụ thể hóa. Một runbook là một công thức từng bước cho một tình huống vận hành, viết sao cho một người căng thẳng lúc 3 giờ sáng có thể làm theo: triệu chứng — kiểm tra (lệnh chính xác) — cách sửa (lệnh chính xác) — leo thang (đánh thức ai nếu không được). **Lab (Tuần 23–24):** viết hai runbook trong repo của bạn — “web server down” và “disk filling up” — rồi *diễn tập* chúng: làm hỏng thứ đó, làm theo đúng nghĩa đen tài liệu của chính bạn, và sửa mọi bước tổ ra mơ hồ. Sau đó chạy một **game day** trọn vẹn: nhờ một người bạn (hoặc một AI) bí mật phá stack lab của bạn; bạn bị “gọi”, chẩn đoán từ metric và log, giảm nhẹ, và viết post-mortem vào nhật ký. Bản post-mortem đó là một câu chuyện phỏng vấn, và là một câu chuyện hay.

Từ vựng:

Thuật ngữ	Định nghĩa
CloudWatch	Dịch vụ giám sát của AWS: metric, alarm, dashboard, log.
Metric / alarm / dashboard	Một con số theo thời gian / một quy tắc kích hoạt trên nó / một màn hình các con số đó.
SNS	Dịch vụ thông báo mà alarm publish vào — email, SMS, máy nhắn tin.
Golden signals	Latency, traffic, errors, saturation — bốn con số mô tả sức khỏe của bất kỳ dịch vụ nào.
Alert fatigue	Quá nhiều báo động không cần hành động — báo động bị phớt lờ — bỏ lỡ báo động thật. Kẻ thù của thiết kế alarm.
Log group / Logs Insights	Nơi log tập trung đổ về / ngôn ngữ truy vấn trên chúng.
Incident / severity	Một sự suy giảm ngoài kế hoạch / mức độ tồi tệ được xếp hạng của nó (sev-1 = tệ nhất).
Triage / mitigate / resolve	Đánh giá nhanh / cầm máu trước / thực sự sửa.
Post-mortem	Bản rà soát viết ra, không đổ lỗi: cái gì, vì sao, cái gì ngăn lặp lại.
Runbook	Công thức chông-được-3-giờ-sáng cho một tình huống: triệu chứng, kiểm tra, cách sửa, leo thang.
On-call / game day	Vòng xoay mang máy nhắn tin / một sự cố

Thuật ngữ	Định nghĩa
MTTR	luyện tập có chủ đích. Mean time to recovery — thời gian trung bình để phục hồi; chỉ số vận hành mà các đội trưởng thành tối ưu.

Nói thành tiếng: “Did we find out from the alarm or from a customer?” (Chúng ta biết tin từ alarm hay từ một khách hàng?) · “Mitigate first — root cause after we’re stable.” (Giảm nhẹ trước — nguyên nhân gốc sau khi chúng ta ổn định.) · “Is there a runbook for this? There will be by tomorrow.” (Có runbook cho việc này không? Sẽ có trước ngày mai.) · “What did the post-mortem conclude, and what’s the prevention item?” (Post-mortem kết luận gì, và hạng mục phòng ngừa là gì?)

Milestone: bản post-mortem game-day của bạn tồn tại, không đổ lỗi, và nêu đích danh một biện pháp phòng ngừa cụ thể mà sau đó bạn đã thực sự triển khai (một alarm được thêm, một runbook được sửa).

Module 10 (Tuần 25–28): Vận hành bảo mật và quản lý chi phí

Vận hành bảo mật — vệ sinh, không phải anh hùng. Shared responsibility model (mô hình trách nhiệm chia sẻ) trước tiên (xem lại: <https://www.youtube.com/watch?v=ESPBBEK-cvo>): AWS bảo vệ bản thân cloud; *bạn* bảo vệ những gì bạn đặt vào trong nó — và đa số các vụ rò rỉ thực tế là do khách hàng cấu hình sai, không phải AWS thất bại. Danh sách bảo mật vận hành của bạn, luyện đến mức nhàm chán:

1. **Vệ sinh IAM** (hàng tháng): MFA ở khắp nơi; không có access key sống lâu ở nơi một role làm được việc; chạy credential report; xóa các user và key không dùng; chất vấn mọi dấu * trong mọi policy. Bạn đã xây phần xạ này trong Lab 5 — giờ nó là một mục trên lịch.
2. **Vá lỗi (patching)**: phần mềm chưa vá là cách đa số các vụ đột nhập bắt đầu. Trên các instance của bạn: áp dụng cập nhật theo lịch (và biết rằng **SSM Patch Manager** tự động hóa việc này trên cả hạm đội); trong thế giới container, vá lỗi thường nghĩa là build lại image từ một image nền đã cập nhật — hãy nói điều đó với Module 8.
3. **Backup — loại đã được kiểm chứng**: một bản backup chưa kiểm chứng là một niềm hy vọng, không phải một kế hoạch. RDS snapshot, S3 **versioning** (mỗi lần ghi đè giữ lại phiên bản cũ — biện pháp chống-lỡ-tay và chống-ransomware). Lab: chụp snapshot cơ sở dữ liệu lab của bạn, *khôi phục nó* sang một instance mới, xác minh các hàng dữ liệu, tháo dỡ. Giờ bạn được phép nói “đã kiểm chứng” trong các buổi phỏng vấn.
4. **Guardrails (lan can bảo vệ)**: bật **CloudTrail** (nhật ký kiểm toán của mọi lời gọi API trong tài khoản — ai làm gì, khi nào) và lướt qua các sự kiện của ngày hôm qua một lần; bật bản dùng thử miễn phí của **GuardDuty** (phát hiện mối đe dọa tự động) và đọc xem nó canh chừng những gì. Biết cụm từ **encryption at rest and in transit** (mã hóa khi lưu trữ và khi truyền) và rằng trong AWS nó chủ yếu là một ô đánh dấu được chống lưng bởi **KMS** — điều kiện tối thiểu, luôn bật.

Quản lý chi phí — kỹ năng khiến junior trông như senior. Gạch đầu dòng JD nguyên văn: “*manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging*” (quản lý chi phí cloud thông qua điều chỉnh kích cỡ tài nguyên, triển khai auto-scaling, gắn thẻ tài nguyên). Đồng hồ tính tiền chạy thế nào: compute tính tiền theo từng giây nó *bật* (nhàn rồi ≠ miễn phí — “chúng tôi để nó chạy quên” là kiểu lãng phí kinh điển), lưu trữ theo GB-tháng, và **egress** (dữ liệu *đi ra khỏi* AWS) theo từng GB trong khi dữ liệu đi vào miễn phí — cú bất ngờ hóa đơn nổi tiếng. Các đòn bẩy, theo thứ tự một junior có thể kéo: **tắt nó đi** (hệ thống dev vào ban đêm; thói quen tháo dỡ của bạn, được công nghiệp hóa) — **rightsized** (đa số server bị cấp thừa; kiểm tra lịch sử CPU trong CloudWatch và thu nhỏ) — **gắn tag mọi thứ** (Project, Owner, Environment — chỉ tiêu không tag là chỉ tiêu không ai chịu trách nhiệm; cưỡng chế tag trong Terraform của bạn) — **phân bậc lưu trữ** (lifecycle rule từ Lab 2) — biết rằng **reserved capacity/Savings Plans** (cam kết 1–3 năm, tiết kiệm 30–70%) và **Spot** (giảm đến 90%, có thể bị gián đoạn) tồn tại lần lượt cho workload ổn định và workload batch. Công cụ: **Cost Explorer** (chi tiêu của tài khoản, vẽ thành biểu đồ — đọc nó hằng tuần, mãi mãi) và **AWS Budgets** (cảnh báo \$0 của bạn từ Tuần 3, giờ được hiểu là thành viên nhỏ nhất của một gia đình nghiêm túc). Bộ môn rộng hơn được gọi là **FinOps** — đáng hai phút: https://www.youtube.com/watch?v=Y-c_xw9bHFW.

Từ vựng:

Thuật ngữ	Định nghĩa
Shared responsibility model	AWS bảo vệ cloud; bạn bảo vệ những gì bạn đặt vào đó. Đa số các vụ rò rỉ thuộc nửa thứ hai.
CloudTrail	Nhật ký kiểm toán của tài khoản: mọi lời gọi API, bởi ai, khi nào.
GuardDuty	Phát hiện môi đe dọa tự động của AWS trên log và lưu lượng.
Patching / SSM	Áp dụng các bản vá bảo mật / Systems Manager của AWS, thứ tự động hóa việc đó trên cả hạm đội.
S3 versioning	Giữ mọi phiên bản bị ghi đè — biện pháp chống-lỡ-tay, chống-ransomware.
KMS / encryption at rest & in transit	Dịch vụ khóa của AWS / dữ liệu được xáo trộn trên đĩa và trên đường truyền. Luôn bật.
Egress	Dữ liệu rời khỏi AWS — tính tiền theo GB; chiêu vào miễn phí. Cú bất ngờ hóa đơn kinh điển.
Rightsizing	Thu nhỏ tài nguyên bị cấp thừa về đúng nhu cầu đo được. Tiền nhặt được.
Tagging	Dán nhãn mọi tài nguyên với chủ sở hữu/dự án/môi trường để mọi khoản chi đều truy được trách nhiệm.

Thuật ngữ	Định nghĩa
Savings Plans / Spot	Cam kết 1–3 năm để giảm 30–70% cho tải ổn định / công suất dư có thể bị thu hồi, giảm đến 90% cho batch.
Cost Explorer / Budgets	Các biểu đồ chi tiêu bạn đọc hằng tuần / các cảnh báo giúp bạn không bao giờ phải học từ hóa đơn.
FinOps	Bộ môn làm cho chi tiêu cloud trở nên hữu hình, được phân bổ, và được tối ưu liên tục.

Nói thành tiếng: “Who has access to prod, and when did we last review the list?” (Ai có quyền truy cập prod, và lần cuối chúng ta rà soát danh sách là khi nào?) · “When did we last *restore* a backup?” (Lần cuối chúng ta thực sự *khôi phục* một bản backup là khi nào?) · “What’s untagged, and what died but is still billing?” (Cái gì chưa được gắn tag, và cái gì đã chết mà vẫn đang tính tiền?) · “It’s over-provisioned — the CPU history says we can halve it.” (Nó bị cấp thừa — lịch sử CPU nói chúng ta có thể giảm một nửa.)

Bài tập: (1) Lab kiểm chứng khôi phục ở trên. (2) Một file danh-sách-bảo-mật-hằng-tháng trong repo của bạn, rồi thực sự chạy nó trên tài khoản của bạn. (3) Trong Cost Explorer, tìm thứ đắt nhất trong lịch sử tài khoản của bạn và giải thích nó trong một câu nhật ký. (4) Thêm tag mặc định cho mọi tài nguyên trong repo Terraform của bạn.

Milestone — kết thúc Giai đoạn 3: chạy một cuộc tự kiểm toán trọn vẹn — danh sách bảo mật, rà soát chi phí, kiểm tra alarm, khôi phục backup — và viết bản báo cáo một trang. Giờ bạn có thể làm được *công việc*, không chỉ phân xây dựng. Điều còn lại là chứng minh nó với người lạ: Giai đoạn 4.

GIẢI ĐOẠN 4 — SẴN SÀNG ĐI LÀM (Tháng 8–12)

Module 11 (Tháng 8–10): Ba dự án portfolio

Chúng chỉ nói rằng bạn đã học; dự án chứng minh bạn xây được. Ba dự án, mỗi cái trong một repo GitHub riêng, mỗi cái có một sơ đồ kiến trúc, một README viết cho người quản lý tuyển dụng (cái gì, vì sao, chạy nó thế nào, nó tốn bao nhiêu), và một script tháo dỡ. Xây — chụp màn hình/quay lại — *hủy* — repo mới là sản phẩm, không phải một hóa đơn đang chạy.

Dự án 1 — Ứng dụng web ba tầng, tự động hóa hoàn toàn (tâm điểm). Terraform dựng mọi thứ: VPC với public/private subnet trải trên hai AZ; một **Application Load Balancer** (thiết bị phân tán lưu lượng — mới với bạn, và là bước tiếp theo tự nhiên từ Lab 3) đứng trước một **Auto Scaling Group** các instance web (AWS thêm/bớt instance theo tải — tra cứu nó, đầu nối nó); RDS trong các private subnet; S3 cho tài nguyên tĩnh; các alarm CloudWatch và một runbook. GitHub Actions chạy terraform plan trên mỗi pull request và apply khi merge — một pipeline hạ tầng thực thụ. Một mình dự án này chứng minh mười trong số

mười bốn gạch đầu dòng JD trong bảng ánh xạ; hãy dự liệu rằng mọi buổi phỏng vấn sẽ đi xuyên qua nó, và tập dượt thuật lại nó trong năm phút.

Dự án 2 — Pipeline dữ liệu serverless (độ rộng). Hoàn toàn không có server: một file đáp xuống một S3 bucket kích hoạt một **Lambda function** (Python của bạn, được AWS chạy theo từng sự kiện, tính tiền theo mỗi lần gọi) xử lý nó — phân tích một file CSV, tóm tắt nó, ghi kết quả vào một bucket thứ hai hoặc một bảng DynamoDB — với các thất bại được bắt lại, ghi log vào CloudWatch, và báo động về hộp thư của bạn. Triển khai nó bằng Terraform. Nó thể hiện Python, tư duy hướng sự kiện, và độ rộng vượt ra ngoài EC2. Hai phút định hướng trước: https://www.youtube.com/watch?v=W_VV2Fx32_Y.

Dự án 3 — Màn trình diễn vận hành kiểu production (điểm tạo khác biệt). Lấy Dự án 1 và vận hành nó như thể nó quan trọng thật: một dashboard CloudWatch của các tín hiệu vàng; các alarm với một chính sách gọi trực được ghi thành văn; ba runbook; một quy trình sao lưu-và-khôi phục đã kiểm chứng kèm bằng chứng; một lượt gia cố bảo mật (kiểm toán IAM, ghi chú vá lỗi, CloudTrail bật) được viết lại; một bản phân tích chi phí (“stack này tốn \$X/tháng; đây là ba thay đổi sẽ giảm một nửa”); và một bản post-mortem game-day. Gần như không ứng viên junior nào có cái này. Nó trả lời câu hỏi duy nhất mà các buổi phỏng vấn thực sự hỏi — “người này có đáng tin để giao production không?” — bằng tài liệu thay vì tính từ.

Module 12 (Tháng 8–12, song song): Lịch trình chứng chỉ

Chứng chỉ mở các bộ lọc của nhà tuyển dụng và cấu trúc hóa việc học của bạn. Lộ trình chuẩn năm 2026 cho vai trò này, với thời gian ôn thực tế theo nhịp của khóa học — chú ý mỗi kỳ thi rơi ngay sau các giai đoạn dạy nội dung của nó, đó là lý do các mốc thời gian này ngắn hơn con số trên internet:

Thứ tự	Chứng chỉ	Nó chứng minh điều gì	Thời gian ôn từ vị trí bạn sẽ đứng	Khi nào thi
1	AWS Certified Cloud Practitioner (CLF-C02)	Từ vựng cloud, billing, shared responsibility	3–4 tuần (các cẩm nang nói tương tự cho người mới; Giai đoạn 0–1 phủ phần lớn nội dung)	~Tháng 4
2	AWS Solutions Architect Associate (SAA-C03)	Thiết kế các kiến trúc AWS thực thụ — chứng chỉ mà JD thực sự dùng để lọc	6–8 tuần ôn tập trung (ước tính chuẩn ngành sau CCP)	Tháng 8–9
3	HashiCorp Terraform Associate (003)	Sự trôi chảy IaC, được xác nhận	2–3 tuần — sau Module 7 và Dự án 1, bạn chủ yếu ôn lại (tài liệu ôn chính)	Tháng 10–11

Thứ tự	Chứng chỉ	Nó chứng minh điều gì	Thời gian ôn từ vị trí bạn sẽ đứng	Khi nào thi
			thức: https://developer.hashicorp.com/terraform/tutorials/certification-003	

Lựa chọn thứ tự, nếu nhắm các vị trí mang chức danh vận hành: **AWS SysOps Administrator / CloudOps Associate** — nội dung của nó đúng nghĩa đen là Giai đoạn 3. Với CLF-C02, khóa học đầy đủ của freeCodeCamp (<https://www.youtube.com/watch?v=7HKot-brXFE>, bản kinh điển <https://www.youtube.com/watch?v=NhDYbskXRgc>) cộng các đề thi thử là con đường miễn phí đã được nhiều người đi; với SAA-C03, hãy ghép một khóa học với thật nhiều bài thi thử tính giờ — kỳ thi dựa trên tình huống và sức bền là quan trọng.

Module 13 (Tháng 11-12): Résumé, phỏng vấn, và mười câu hỏi

Résumé: một trang. Mở đầu bằng kỹ năng (AWS, Terraform, Python/Bash, Docker, CI/CD, CloudWatch — phần chiếu chính ngôn từ của JD; các bộ lọc tự động khớp từ khóa) và ba dự án, mỗi cái là hai gạch đầu dòng theo dạng *làm X bằng Y đạt được Z*: “Deployed a three-tier web app on AWS with Terraform and GitHub Actions; zero-downtime deploys via ALB + Auto Scaling.” Liệt kê chứng chỉ kèm ngày tháng. Gắn link GitHub — rồi hãy giả định họ sẽ thực sự mở nó, vì những người giỏi sẽ làm thế, và GitHub của bạn giờ đây xứng đáng với chuyên viêng thăm.

Chuẩn bị phỏng vấn: ba thể loại cần tập dượt — kiến thức (các bảng từ vựng của các Giai đoạn), tình huống (các lab bạn đã thực sự làm), và hành vi (những câu chuyện từ nhật ký của bạn, kể theo khuôn **STAR**: Situation, Task, Action, Result — Tình huống, Nhiệm vụ, Hành động, Kết quả). Luyện *thành tiếng*, mỗi ngày, trong hai tuần — lý tưởng là với một AI phỏng vấn viên, không khoan nhượng. Mười câu hỏi dưới đây phủ các câu kinh điển; các câu trả lời mẫu được cô ý viết cô đọng — hãy mở rộng từng câu bằng chi tiết lab của chính bạn, vì “...và khi tôi xây cái này, điều thực sự xảy ra là...” là câu nói tách bạn khỏi những ứng viên chỉ đọc.

Mười câu hỏi, với câu trả lời mạnh:

1. **“Explain the difference between a public and a private subnet.” (Giải thích sự khác biệt giữa public subnet và private subnet.)** *Route table của một public subnet có route đến một internet gateway, nên tài nguyên của nó có thể được tiếp cận từ internet; một private subnet không có route như vậy. Tầng web đi public, cơ sở dữ liệu đi private, và truy cập quản trị tới các máy private đi qua một bastion hoặc SSM. Trong các lab của tôi, tôi đã chứng minh instance private không thể chạm tới từ laptop của tôi nhưng chạm tới được từ tầng public — cái route bị thiếu chính là sự bảo mật.*
2. **“What is IAM, and what is least privilege?” (IAM là gì, và least privilege là gì?)** *IAM kiểm soát danh tính nào được thực hiện hành động nào trên tài nguyên nào, thông qua user, group, role và các policy JSON. Least privilege nghĩa là cấp mức tối thiểu đủ làm việc — user của script kiểm toán của tôi chỉ có quyền đọc trên đúng một dịch vụ,*

và các EC2 instance của tôi dùng role thay vì key lưu trữ, nên không có thông tin xác thực sống lâu nào để bị rò rỉ.

3. **“An EC2 web server stopped responding. Walk me through your troubleshooting.”** (Một web server EC2 ngừng phản hồi. Hãy dẫn tôi qua quy trình xử lý sự cố của bạn.) Trước hết là khoanh vùng: một server hay tất cả — kiểm tra dashboard CloudWatch và các alarm. Rồi đi qua các tầng theo thứ tự: instance status check; security group — 80/443 có thực sự mở không; SSH vào — nginx có đang chạy không (`systemctl status`), đĩa có đầy không (`df -h`), CPU có kẹt trần không (`top`); rồi đến log (`tail file error log`). Giảm nhẹ trước — khởi động lại dịch vụ hoặc thay instance — và tìm nguyên nhân gốc sau khi đã ổn định. Đó là thứ tự tôi diễn tập trong các runbook của mình.
4. **“Why Terraform instead of clicking in the console?”** (Vì sao dùng Terraform thay vì nhấp chuột trong console?) Console không để lại một bản ghi review được và không thể dựng lại bất cứ thứ gì. Terraform là declarative và có quản lý phiên bản: các thay đổi đi qua pull request, cùng một config dựng các môi trường giống hệt nhau, một thay đổi tôi được quay lui bằng cách revert một commit, và plan cho thấy chính xác bản diff trước khi bất cứ điều gì xảy ra. Pipeline của tôi chạy plan trên mỗi PR nên bản diff là một phần của khâu review.
5. **“What’s the difference between a container and a virtual machine?”** (Sự khác biệt giữa container và máy ảo là gì?) Một VM ảo hóa phần cứng và mang theo cả một hệ điều hành đầy đủ, mất vài phút để khởi động; một container chia sẻ kernel của máy chủ và chỉ đóng gói ứng dụng cùng các phụ thuộc của nó, khởi động trong khoảng một giây. Container mang lại hành vi giống hệt nhau giữa các môi trường — cùng một image chạy trên laptop của tôi và trên EC2. VM vẫn quan trọng cho sự cách ly và cho việc chạy các workload không phải Linux.
6. **“How would you reduce our AWS bill?”** (Bạn sẽ giảm hóa đơn AWS của chúng tôi như thế nào?) Theo thứ tự công sức: tìm những gì đang chạy mà không nên chạy — instance nhàn rỗi, volume không gắn kết, snapshot cũ; Cost Explorer và tag làm điều đó hữu hình. Rồi rightsize dựa trên lịch sử CloudWatch. Rồi lập lịch tắt các môi trường dev ngoài giờ làm việc. Rồi chuyển dữ liệu S3 cũ xuống các bậc rẻ hơn bằng lifecycle. Rồi cam kết các workload ổn định vào Savings Plans và đặt batch chịu được gián đoạn lên Spot. Và canh chừng egress — dữ liệu đi ra là dòng bất ngờ kinh điển trên hóa đơn.
7. **“What happens when you type a URL and press Enter?”** (Điều gì xảy ra khi bạn gõ một URL và nhấn Enter?) DNS phân giải cái tên thành một IP; trình duyệt mở một kết nối TCP đến port 443 và thương lượng TLS; nó gửi một HTTP GET; ở phía bên kia, yêu cầu chạm vào một load balancer, thứ chuyển tiếp đến một instance khỏe mạnh trong một public subnet; ứng dụng có thể truy vấn một cơ sở dữ liệu trong một private subnet; phản hồi quay về với một status code — 200 nếu khỏe mạnh — và trình duyệt hiển thị nó. Tôi có thể đào sâu vào bất kỳ chặng nào, và tôi đã tự xây từng chặng.
8. **“A teammate’s change broke production. What happens next?”** (Thay đổi của một đồng đội làm sập production. Điều gì xảy ra tiếp theo?) Giảm nhẹ trước — quay lui qua pipeline; phục hồi bằng cách đoán trong khoảng khắc đó. Rồi một post-mortem không đổ lỗi: điều gì đã xảy ra, vì sao hệ thống cho phép nó, và lan can bảo vệ nào đã thiếu — một bài kiểm thử, một lượt review plan, một alarm. Không đổ lỗi quan trọng một cách thực dụng: người sợ bị trừng phạt sẽ giấu thông tin, và thông tin bị giấu gây ra sự lặp lại.

9. **“What is the shared responsibility model?” (Shared responsibility model là gì?)**
AWS bảo vệ cloud — trung tâm dữ liệu, phần cứng, hypervisor; khách hàng bảo vệ những gì ở trong đó — dữ liệu, IAM, quy tắc mạng, vá lỗi, cài đặt mã hóa. Đa số các vụ rò rỉ cloud thực tế nằm ở phía khách hàng, những cấu hình sai như bucket public hay key bị lộ. Vậy nên ‘AWS an toàn’ và ‘chúng ta an toàn trên AWS’ là hai câu khác nhau, và câu thứ hai là công việc của tôi.
10. **“Tell me about a problem you debugged.” (Kể cho tôi nghe về một vấn đề bạn đã gỡ lỗi.)** *(Của bạn, từ nhật ký, theo khuôn STAR. Bộ khung ví dụ:) Trong game day giám sát của tôi, alarm của tôi kích hoạt vì bão hòa CPU (S). Tôi phải tìm và chặn nguyên nhân trước khi ‘người dùng’ nhận ra (T). Metric cho thấy thời điểm bắt đầu của cú tăng vọt; Logs Insights nói nó với các tiến trình chạy mất kiểm soát; tôi SSH vào, xác nhận bằng top, giết chúng, và xem sự phục hồi trên dashboard (A). Xử lý xong trong mười một phút, và bản post-mortem sinh ra một alarm để tiến trình mà sau đó tôi đã triển khai bằng Terraform (R).*

Cơ chế tìm việc: nộp hồ sơ từ Tháng 11 — sau SAA, đừng đợi đến khi “sẵn sàng”, vì phỏng vấn chính là luyện tập. Các chức danh mục tiêu: cloud engineer (junior/associate), cloud support engineer, cloud operations engineer, junior DevOps engineer, AWS support associate. Mỗi lần bị từ chối mà có phỏng vấn là một bài học miễn phí; ghi nhật ký những gì họ đã hỏi.

Từ vựng:

Thuật ngữ	Định nghĩa
Portfolio	Bảng chứng công khai, có tài liệu, rằng bạn xây được — với nghề này, là các repo GitHub kèm sơ đồ và README.
Application Load Balancer (ALB)	Bộ phân tán lưu lượng của AWS: phân phối request qua các instance khỏe mạnh, loại bỏ cái ốm yếu.
Auto Scaling Group	Giữ N instance luôn sống và điều chỉnh N theo tải — tự chữa lành và co giãn trong một.
Lambda / serverless	Code mà AWS chạy theo từng sự kiện, tính tiền theo mỗi lần gọi — hoàn toàn không có server nào phải quản lý.
DynamoDB	Bảng NoSQL serverless của AWS — ghép đôi tự nhiên với Lambda.
STAR	Situation, Task, Action, Result — khuôn hình của một câu chuyện phỏng vấn hay.
ATS	Applicant tracking system — bộ lọc từ khóa mà bản résumé một trang của bạn phải vượt qua.
CLF-CO2 / SAA-CO3 / Terraform Associate 003	Ba kỳ thi của bạn: từ vựng, kiến trúc, IaC.
SysOps / CloudOps Associate	Chứng chỉ associate hướng vận hành tùy

Thuật ngữ	Định nghĩa
	chọn của AWS — Giai đoạn 3, được đem ra thi.

Milestone — kết thúc khóa học: ba repo mà một người lạ có thể hiểu, ba chứng chỉ đã lên lịch hoặc đã đậu, mười câu trả lời đã tập dượt thành tiếng, hồ sơ ứng tuyển đang chạy. Bạn không còn “hy vọng bước vào ngành cloud”. Bạn là một kỹ sư cloud junior có bằng chứng, đang đi phỏng vấn.

Giáo trình một trang

Khi nào	Trọng tâm	Bằng chứng thực hành	Bằng chứng bên ngoài
Tuần 1–2	Máy tính hoạt động ra sao; Linux, terminal, permissions, SSH	Thao tác file bằng script; shell script đầu tiên	—
Tuần 3–4	Mạng: IP, DNS, port, HTTP, firewall; tài khoản AWS	Tài khoản + MFA + ngân sách \$0	—
Tuần 5–6	Lab 1: EC2	Web server trên internet, đã tháo dỡ	—
Tuần 7	Lab 2: S3 + CLI	Trang tĩnh; thành thạo CLI	—
Tuần 8–9	Lab 3: VPC	Mạng public/private, chứng minh bastion	—
Tuần 10	Lab 4: RDS	Cơ sở dữ liệu private, snapshot, tháo dỡ	—
Tuần 11–12	Lab 5: IAM; dựng lại màn thử thách Giai đoạn 1	Cả stack từ trí nhớ, <3 giờ	Đặt lịch thi CCP
Tuần 13–15	Bash, Python/boto3, Git	backup.sh, audit.py, lịch sử repo	Thi CCP (~Tháng 4)
Tuần 16–17	Terraform	Stack dưới dạng code, plan/apply/destroy	—
Tuần 18–20	CI/CD, Docker, hiểu biết K8s	Pipeline xanh; image trong registry	—
Tuần 21–24	CloudWatch, log, sự cố, runbook	Alarm đã kích hoạt + post-mortem game-	—

Khi nào	Trọng tâm	Bằng chứng thực hành	Bằng chứng bên ngoài
Tuần 25–28	Vận hành bảo mật; quản lý chi phí	đay Báo cáo tự kiểm toán; kiểm chứng khôi phục	—
Tháng 8–10	Dự án Portfolio 1–3	Ba repo có tài liệu	SAA-C03 (Tháng 8–9)
Tháng 10–11	Ôn Terraform Associate	—	Terraform Associate
Tháng 11–12	Résumé, phỏng vấn, nộp hồ sơ	Mười câu trả lời, thành tiếng	Những lời mời đầu tiên

Lời kết từ người thầy của bạn. Mười hai tháng là ngắn cho một cuộc đổi nghề và dài cho một thói quen hằng ngày, vậy nên đây là thỏa thuận trung thực: những người hoàn thành khóa học này không phải những người thông minh nhất — họ là những người vâng lời cả trong những ngày mệt mỏi. Mỗi thông báo lỗi bạn gặp là giáo trình đang hoạt động, không phải đang thất bại; người kỹ sư đã làm hỏng và sửa xong một trăm thứ nhỏ chính là điều mà một người quản lý tuyển dụng muốn nói khi dùng chữ “kinh nghiệm”. Giữ cuốn nhật ký, tháo dỡ những gì bạn xây, đừng bao giờ tuyên bố trong phỏng vấn điều mà các repo của bạn không chống lưng được — và một năm nữa, khi máy nhắn tin reo lúc 3 giờ sáng, bạn sẽ cảm thấy một điều bất ngờ bên dưới cơn adrenaline: năng lực. Đi mà xây thôi.

Nguồn tham khảo

Nghiên cứu bản mô tả công việc (tháng 8 năm 2026): [Arc.dev — AWS Cloud Engineer Job Description](#) · [Wiz — Cloud Engineer Job Description Guide](#) · [DevsData — AWS Cloud Engineer JD Template](#) · [XOPA — Cloud Engineer JD Template 2026](#) · [Betterteam — Cloud Engineer Job Description](#). Lộ trình chứng chỉ và thời gian ôn: [StudyTech — AWS Certification Roadmap 2026](#) · [Cloud Evolvers — Cloud Engineer Roadmap 2026](#) · [HashiCorp — Terraform Associate 003 prep](#). Bối cảnh lương (trung vị Mỹ ~ \$104K, khoảng \$85K–\$140K): Wiz, ở trên. Mọi liên kết YouTube đã được kiểm chứng qua metadata YouTube tại thời điểm viết.

Một khóa học B4LCILC — tập sách đồng hành của “The Cloud Leader Course”.

CÁC BÀI HỌC MỚI — ẨM BẢN THÁNG 8 NĂM 2026

Vào tháng 8 năm 2026, các bài học mới đã được bổ sung vào ba khóa học tương tác tại [b4lcilc.org](#) — khóa Cloud Leader (Lãnh đạo Cloud), Cloud Engineer (Kỹ sư Cloud) và Cloud Architect (Kiến trúc sư Cloud). Phần phụ lục này cập nhật ấn bản đọc cho theo kịp, với những bản cô đọng trung thành của tài liệu mới đó, cùng một giọng văn với phần còn lại của cuốn sách này.

Đôi lời về thứ bạn đang cầm trên tay. Những trang này là bản đọc: các ý tưởng, các phép ẩn dụ, và những câu văn đáng nói thành tiếng. Phiên bản tương tác đầy đủ trên mạng mang theo nhiều hơn đáng kể — mỗi bài học mới đều đi kèm các bài quiz kiểm tra mức độ hiểu trước khi cho bạn đi tiếp, các bài thực hành hands-on (bao gồm một terminal luyện tập hoạt động y như terminal thật nhưng không thể tốn tiền), video, bài luyện từ vựng, và — với phần tài liệu capstone — các bài tập thiết kế có hướng dẫn kèm danh sách kiểm tra yêu cầu và đáp án mẫu được giải chi tiết. Hãy đọc những bản cô đọng này để nắm được hình hài của từng ý tưởng trong đầu; rồi khi có điều kiện, hãy ngồi làm phiên bản tương tác để kiên thức bám rễ.

Các phần bổ sung chia thành ba nhóm. Cho khóa **Cloud Leader**: một phần giới thiệu nhẹ nhàng về terminal, luật riêng tư của chính Thái Lan, cẩm nang dành cho người lãnh đạo về các đề xuất AI và hóa đơn AI, và một bài tập capstone gom toàn bộ khóa học vào một trang giấy. Cho khóa **Cloud Engineer**: điện toán serverless — mẫu hình biến những script vận hành nhỏ thành các function chạy mà gần như không tốn đồng nào. Cho khóa **Cloud Architect**: phép tính số học của các lời hứa về độ sẵn sàng, mẫu hình Retrieval-Augmented Generation mà khách hàng giờ đây hỏi đến hằng tuần, và một bài tập tính chi phí trọn vẹn — định giá một bản thiết kế trước khi bất kỳ ai xây nó.

Cho khóa Cloud Engineer

Serverless thực chất là gì

Mọi thứ bạn đã tự động hóa đến giờ vẫn giả định có một cỗ máy: cron job trong bài thực hành của module này chạy hai phút mỗi ngày trên một máy ảo mà bạn trả tiền cả tháng, phải vá lỗi, giám sát, và một ngày nào đó gỡ lỗi lúc 3 giờ sáng. Serverless là câu trả lời của cloud cho sự lệch pha đó. Server vẫn tồn tại — cái tên chỉ là chiêu tiếp thị — nhưng chúng là vấn đề của nhà cung cấp. Bạn trao một function, nhà cung cấp chạy nó khi có thứ gì đó kích hoạt, và bạn trả tiền theo từng lần gọi, tính đến từng mili-giây. Mô hình này gọi là **FaaS — Function as a Service** (hàm như một dịch vụ): **Lambda** trên AWS, **Azure Functions** trên Azure. Các khái niệm chuyển đổi tương ứng tên-đối-tên.

Câu tạo của nó nằm gọn trong lòng bàn tay. Một **handler** là function mà nền tảng gọi đến, và nó nhận một **event** (sự kiện) — một bản mô tả dạng JSON (JavaScript Object Notation) về bất cứ điều gì vừa xảy ra: một file đáp xuống một bucket, một HTTP request đến nơi, một lịch hẹn điểm giờ. Hãy chú ý điều cuối cùng: “một lịch hẹn điểm giờ” chính xác là điều cron làm — chỉ khác là một Lambda chạy theo lịch không cần cỗ máy nào bật sẵn giữa các lần chạy. Chuyển đổi một trong các script vận hành của module này là một dự án serverless đầu tay kinh điển.

Bài toán kinh tế là lý do mô hình này tồn tại, và nó cắt theo cả hai chiều. Bậc miễn phí của AWS bao gồm một triệu lần gọi Lambda mỗi tháng, mãi mãi — function trong bài thực hành của bạn có thể chạy từng phút suốt cả năm mà không tốn đồng nào, biến serverless thành ngôi nhà rẻ nhất cho những mẫu keo dán vận hành. Nhưng mức giá theo mili-giây vốn chỉ là sai số làm tròn ở lưu lượng thấp sẽ thành tiền thật ở lưu lượng cao; một function chạy liên tục thường rẻ hơn nên là container hay VM. Kỹ năng chuyên nghiệp là biết một workload nằm ở phía nào của lần ranh đó.

Hai lời cảnh báo trung thực. Thứ nhất, **cold start** (khởi động nguội): một function lâu không chạy mất vài trăm mili-giây để thức dậy — không đáng kể với một báo cáo hằng đêm, nhưng là một cân nhắc thiết kế với một API nhạy cảm về độ trễ. Thứ hai, một function vẫn chạy dưới một danh tính: Lambda nhận một role IAM (Identity and Access Management — quản lý danh tính và quyền truy cập) đúng hệt như một EC2 instance, và một function được cấp quyền quá tay vẫn là cùng một sai lầm bảo mật, chỉ trong lớp vỏ nhỏ hơn.
