

Kursus Jurutera Cloud (The Cloud Engineer Course)

Daripada sifar kepada jurutera cloud yang benar-benar sedia bekerja — sebuah kursus B4LCILC

13 Ogos 2026

Kursus Jurutera Cloud (The Cloud Engineer Course)

Daripada sifar pengetahuan kepada benar-benar sedia bekerja sebagai jurutera cloud.

Bagaimana kursus ini berfungsi

Anda sedang berlatih untuk *menjadi* seorang jurutera cloud: orang yang membina server (pelayan) dengan kod, memastikan sistem terus hidup pada pukul 3 pagi, mengautomasikan kerja-kerja membosankan, dan mampu memandang sistem yang rosak lalu mencari puncanya dengan tenang. Itu adalah kerja tangan (hands-on trade), dan kursus ini melayannya sebegitu. Anda akan menghabiskan jauh lebih banyak masa menaip berbanding membaca — setiap modul ada makmal (lab), dan makmal itulah kursus ini. Membaca tentang cloud membina pengecaman; membina di dalam cloud membina kerjaya.

Kursus ini mempunyai lima peringkat. **Peringkat 0 (Minggu 1–4): Asas** — bagaimana komputer dan rangkaian sebenarnya berfungsi, dan Linux, sistem pengendalian yang menggerakkan cloud. **Peringkat 1 (Minggu 5–12): Teras cloud, secara hands-on** — lima perkhidmatan AWS yang muncul dalam setiap iklan jawatan, dipelajari sebagai makmal yang anda bina dan robohkan semula. **Peringkat 2 (Minggu 13–20): Automasi** — penskripan, Git, Terraform, CI/CD, dan container: kemahiran yang membezakan seorang jurutera daripada seorang pengklik console. **Peringkat 3 (Minggu 21–28): Operasi** — pemantauan, insiden, keselamatan, dan kos: kemahiran yang sebenarnya anda dibayar untuknya. **Peringkat 4 (Bulan 8–12): Kesediaan kerja** — tiga projek portfolio, tiga pensijilan, dan latihan temu duga.

Peraturan untuk keseluruhan kursus: belajar **1.5–2 jam sehari, enam hari seminggu** — konsistensi mengalahkan intensiti, dan ini adalah kemahiran yang dipelajari melalui ulangan harian, seperti alat muzik. Setiap modul berakhir dengan **jadual Perbendaharaan Kata** (perkataan yang mesti anda kuasai), **latih tubi Sebut Dengan Kuat** (ayat-ayat yang anda ulang sehingga menjadi semula jadi — temu duga kerja adalah lisan, bukan bertulis), **Latihan hands-on**, dan satu **Pencapaian (Milestone)** yang menjadi pintu kemajuan: jangan mara selagi anda belum mampu melakukannya, kerana setiap peringkat berdiri di atas peringkat sebelumnya. Mulai Minggu 1, simpan sebuah **Jurnal Kejuruteraan (Engineering Journal)**: setiap makmal, setiap mesej ralat, setiap pembetulan, satu perenggan yang jujur. Menjelang Bulan 10, jurnal itu menjadi bahan mentah portfolio anda dan kisah-kisah temu duga anda.

Satu janji sebagai balasan: tiada apa-apa dalam kursus ini yang menganggap anda sudah tahu sesuatu. Setiap istilah ditakrifkan pada kali pertama ia muncul. Jika anda boleh menggunakan

pelayar web dan sanggup menaip arahan yang terasa asing untuk dua minggu pertama, anda sudah memiliki semua prasyarat yang ada.

Deskripsi kerja yang anda latih untuk capai

Pada Ogos 2026 kami mengumpulkan deskripsi kerja Cloud Engineer sebenar daripada templat majikan dan panduan pengambilan (Arc.dev, Wiz, DevsData, XOPA, Betterteam — senarai penuh dalam Sumber). Buangkan nama-nama syarikat dan keperluan yang sama muncul berulang kali. Jadual ini ialah kontrak anda dengan kursus ini — setiap butiran yang diminta majikan dipetakan kepada peringkat yang mengajarnya:

Apa yang diminta oleh deskripsi kerja sebenar (hampir kata demi kata)	Di mana kursus ini mengajarnya
“Knowledge of Linux/Unix operating systems” (Pengetahuan sistem pengendalian Linux/Unix)	Peringkat 0, Modul 1
“Expertise in cloud networking including VPCs, subnets, load balancers, DNS” (Kepakaran rangkaian cloud termasuk VPC, subnet, load balancer, DNS)	Peringkat 0 Modul 2 + Peringkat 1 Makmal 3
“Demonstrated expertise in core AWS services, including EC2, S3, RDS, VPC, IAM” (Kepakaran terbukti dalam perkhidmatan teras AWS, termasuk EC2, S3, RDS, VPC, IAM)	Peringkat 1 (Makmal 1–5)
“Design, develop, and deploy cloud infrastructure using infrastructure as code tools such as Terraform, CloudFormation” (Mereka bentuk, membangunkan, dan men-deploy infrastruktur cloud menggunakan alat infrastructure as code seperti Terraform, CloudFormation)	Peringkat 2, Modul 7
“Proficiency in scripting languages such as Python, Bash, PowerShell, or Go” (Kemahiran bahasa penskripan seperti Python, Bash, PowerShell, atau Go)	Peringkat 2, Modul 6
“Build and maintain CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions” (Membina dan menyelenggara pipeline CI/CD menggunakan alat seperti Jenkins, GitLab CI, atau GitHub Actions)	Peringkat 2, Modul 8
“Experience with containerization technologies including Docker and Kubernetes” (Pengalaman dengan teknologi kontainerisasi termasuk Docker dan Kubernetes)	Peringkat 2, Modul 8

Apa yang diminta oleh deskripsi kerja sebenar (hampir kata demi kata)	Di mana kursus ini mengajarnya
“Monitor infrastructure health and performance using cloud-native monitoring tools” (Memantau kesihatan dan prestasi infrastruktur menggunakan alat pemantauan cloud-native)	Peringkat 3, Modul 9
“Participate in incident response, including log analysis”; “troubleshooting and analytical skills” (Menyertai tindak balas insiden, termasuk analisis log; kemahiran penyelesaian masalah dan analitikal)	Peringkat 3, Modul 10
“Implement and enforce security controls including encryption, identity and access management”; “least-privilege access” (Melaksana dan menguatkuasakan kawalan keselamatan termasuk penyulitan, pengurusan identiti dan capaian; capaian least-privilege)	Peringkat 1 Makmal 5 + Peringkat 3 Modul 10
“Manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging” (Mengurus kos cloud melalui rightsizing sumber, melaksanakan auto-scaling, tagging sumber)	Peringkat 3, Modul 10
“Maintaining, testing and implementing disaster recovery procedures” (Menyelenggara, menguji dan melaksanakan prosedur pemulihan bencana)	Peringkat 3 + Projek Portfolio 3
“AWS certifications preferred” (Pensijilan AWS diutamakan)	Jadual sijil Peringkat 4 (CCP – SAA – Terraform Associate)
“Provide technical guidance and documentation”; “good communication and collaboration skills” (Memberi panduan teknikal dan dokumentasi; kemahiran komunikasi dan kolaborasi yang baik)	Jurnal + runbook + persediaan temu duga Peringkat 4

Konteks gaji, supaya anda tahu apa yang sedang anda usahakan: gaji jurutera cloud di AS berkelompok sekitar **median hampir \$104,000**, dengan julat lazim **\$85K–\$140K** dan jawatan AWS senior/pakar diiklankan jauh melebihi itu. Jawatan peringkat permulaan wujud dengan pelbagai nama — cloud support associate, junior cloud engineer, cloud operations engineer — dan kursus ini menyasarkan tepat keperluan mereka.

PERINGKAT 0 — ASAS (Minggu 1-4)

Modul 1 (Minggu 1-2): Bagaimana komputer berfungsi, dan Linux — bahasa server

Idea besarnya: sebuah **server** hanyalah komputer yang tugasnya melayani komputer-komputer lain. Komputer riba di hadapan anda dan mesin-mesin yang menjalankan Netflix berbeza dari segi saiz dan kebolehpercayaan, bukan dari segi jenis: kedua-duanya ialah **CPU** (bahagian yang melakukan kerja), **memori/RAM** (ruang kerja jangka pendek yang pantas, dipadamkan apabila dimulakan semula), **disk** (cakera — storan jangka panjang yang perlahan tetapi terselamat selepas dimulakan semula), dan **kad rangkaian** (sambungan kepada segala yang lain), semuanya diselaraskan oleh sebuah **sistem pengendalian (OS)**. Komputer riba anda mungkin menjalankan Windows atau macOS. Server secara majoritinya menjalankan **Linux** — OS percuma dan sumber terbuka yang stabil, boleh diskripikan, dan dikawal sepenuhnya melalui arahan bertaip. Bahagian terakhir itulah intinya: anda tidak boleh mengautomasikan klik tetikus, tetapi anda boleh mengautomasikan arahan, dan kejuruteraan cloud *ialah* automasi. Maka kefasihan pertama anda ialah **terminal**.

Terminal, tanpa misteri. Terminal (atau “shell” — program di dalamnya biasanya **Bash**) ialah perbualan teks dengan komputer. Anda menaip satu arahan; ia menjawab. Itu sahaja. Tanda \$ yang akan anda lihat dalam contoh ialah **prompt** — shell berkata “giliran anda.” Segala-galanya dalam Linux ialah fail, fail hidup dalam satu pokok tunggal bermula di akar /, dan folder peribadi anda ialah /home/namaanda (nama panggilan: ~). Sebuah **path** (laluhan) ialah alamat sesebuah fail dalam pokok itu: /home/anna/notes.txt.

Dapatkan Linux untuk berlatih (pilih satu, sepuluh minit): di Windows, pasang **WSL** (Windows Subsystem for Linux — Ubuntu Linux sebenar di dalam Windows: <https://learn.microsoft.com/en-us/windows/wsl/install>); di Mac, aplikasi Terminal terbina dalam sudah cukup hampir untuk bermula (macOS ialah sepupu Unix); atau tunggu Minggu 5 apabila anda akan menyewa server Linux sebenar daripada AWS secara percuma. WSL ialah jawapan terbaik untuk kebanyakan orang.

25 arahan teratas — jadual ini ialah Minggu 1-2. Taip setiap satu daripadanya, berkali-kali:

Arahan	Apa fungsinya	Contoh
<code>pwd</code>	Print working directory — “saya di mana?”	<code>pwd</code>
<code>ls</code>	Senaraikan fail di sini (-l perincian panjang, -a termasuk yang tersembunyi)	<code>ls -la</code>
<code>cd</code>	Change directory — bergerak di dalam pokok fail	<code>cd /var/log</code>
<code>mkdir</code>	Mencipta sebuah direktori (folder)	<code>mkdir projects</code>
<code>touch</code>	Mencipta fail kosong	<code>touch notes.txt</code>

Arahan	Apa fungsinya	Contoh
cp	Menyalin fail (-r untuk folder)	cp a.txt backup.txt
mv	Memindah atau menamakan semula	mv old.txt new.txt
rm	Remove — memadam secara kekal, tiada tong sampah. Hormatinya.	rm notes.txt
cat	Memaparkan keseluruhan kandungan fail	cat notes.txt
less	Membaca fail panjang halaman demi halaman (q untuk keluar)	less /var/log/syslog
head / tail	Baris pertama / terakhir sesebuah fail. tail -f memerhati log secara langsung — klasik dunia ops	tail -f app.log
grep	Mencari corak dalam teks — arahan ops yang paling kerap digunakan	grep "ERROR" app.log
find	Mencari fail mengikut nama/saiz/umur	find / -name "*.conf"
echo	Memaparkan teks (sering ke dalam fail atau pemboleh ubah)	echo "hello"
nano	Penyunting teks dalam terminal yang mesra pengguna	nano notes.txt
man	Manual untuk mana-mana arahan (q untuk keluar)	man grep
sudo	Menjalankan satu arahan sebagai admin berkuasa penuh ("root"). Dengan penuh hormat.	sudo apt update
apt	Memasang/mengemas kini perisian (keluarga Ubuntu/Debian)	sudo apt install htop
chmod	Menukar kebenaran (permission) sesebuah fail	chmod 644 notes.txt
chown	Menukar pemilik sesebuah fail	sudo chown anna file
ps	Menyenaraikan proses yang	ps aux

Arahan	Apa fungsinya	Contoh
	sedang berjalan (ps aux untuk semua)	
top (atau htop)	Papan pemuka langsung CPU/memori/proses — “kenapa server ni perlahan?” bermula di sini	top
df -h / du -sh	Ruang cakera yang kosong / ruang yang digunakan sesebuah folder — “cakera dah penuh” bermula di sini	df -h
ssh	Log masuk ke terminal mesin lain melalui rangkaian — <i>inilah</i> pintu hadapan jurutera cloud	ssh anna@server-ip
curl	Membuat permintaan web dari terminal — “laman tu hidup tak?”	curl https://example.com

Kebenaran (permissions), versi 60 saat. Setiap fail ada pemilik dan mod seperti `rwxr-xr--`: tiga triplet — pemilik, kumpulan, semua orang — untuk **read** (baca), **write** (tuliskan), **execute** (laksana). Dalam nombor: `r=4, w=2, x=1`, jadi `chmod 755 script.sh` bermaksud “pemilik boleh buat segalanya ($7=4+2+1$); orang lain boleh membaca dan menjalankannya ($5=4+1$).” Apabila sesebuah program “doesn’t have permission,” itulah sistem berkata tidak — dan kini anda boleh membaca sebabnya.

SSH, versi 60 saat. `ssh` membuka terminal jauh yang selamat pada mesin lain — dari komputer riba anda masuk ke sebuah server di Virginia seolah-olah anda duduk di hadapannya. Berbanding kata laluan, para profesional menggunakan **key pair** (pasangan kunci): satu private key (fail rahsia di komputer riba anda, tidak sekali-kali dikongsi) dan satu public key (diletakkan di server). Kedua-duanya padan seperti kunci dan mangga. Setiap server AWS yang anda lancarkan di Peringkat 1 akan menyerahkan tepat perkara ini kepada anda.

Proses dan perkhidmatan: sebuah **proses** ialah program yang sedang berjalan; sebuah **service** (perkhidmatan, atau “daemon”) ialah proses yang berjalan selama-lamanya di latar belakang — web server, database. `systemctl status nginx` bertanya kepada Linux “adakah perkhidmatan nginx sihat?” — satu ayat yang akan anda taip secara profesional selama bertahun-tahun.

Perbendaharaan kata:

Istilah	Maksud
Server	Komputer yang tugasnya melayani komputer-komputer lain. Dalam cloud, yang anda sewa.
CPU / RAM / disk	Si pekerja, ruang kerja sementara yang

Istilah	Maksud
Sistem pengendalian (OS)	pantas (dipadamkan apabila dimulakan semula), dan storan kekal yang perlahan. Perisian yang menjalankan mesin dan menempatkan program-program. Server menjalankan Linux.
Linux / distribution	OS server percuma dan sumber terbuka. Sebuah “distro” (Ubuntu, Amazon Linux, Debian) ialah satu perisa terbungkusnya.
Terminal / shell / Bash	Antara muka teks kepada OS / program yang mentafsir arahan anda / nama shell piawai.
Prompt	Tanda \$ — shell menunggu arahan anda.
Direktori / path	Sebuah folder / alamat penuh sesebuah fail dalam pokok tunggal yang bermula di /.
Root (dua makna)	Puncak pokok fail (/) dan pengguna admin berkuasa penuh. Konteks memberitahu anda yang mana satu.
sudo	“Superuser do” — menjalankan satu arahan dengan kuasa admin.
Permissions (rwx)	Peraturan per-fail tentang siapa boleh membaca, menulis, melaksana — dipaparkan sebagai triplet untuk pemilik/kumpulan/semua orang.
Proses / service (daemon)	Program yang sedang berjalan / yang berjalan selama-lamanya di latar belakang (web server, database).
SSH / key pair	Log masuk jauh yang selamat ke terminal mesin lain / fail kunci private+public yang menggantikan kata laluan.
Log	Fail teks tempat perisian menulis apa yang berlaku — tempat pertama anda melihat apabila apa-apa sahaja rosak.
Package manager	Pemasang milik OS (apt, yum) — perisian melalui arahan, bukan melalui halaman muat turun.

Video untuk modul ini (pautan disahkan):

Video	Saluran	Tempoh	Pautan
Linux Operating System — Crash Course for Beginners	freeCodeCamp	~2 jam	https://www.youtube.com/watch?v=ROjZy1WbCIA

Video	Saluran	Tempoh	Pautan
WSL install guide (rujukan, bukan video)	Microsoft Learn	—	https:// learn.microsoft.com/ en-us/windows/ wsl/install

Sebut dengan kuat sehingga menjadi semula jadi: “Let me SSH in and check the logs.” (Biar saya SSH masuk dan semak log.) · “Grep the log for the error, then tail -f it while we retry.” (Grep log itu untuk cari ralat, kemudian tail -f sambil kita cuba semula.) · “It’s a permissions problem — who owns the file and what’s the mode?” (Ini masalah permission — siapa pemilik fail itu dan apakah modnya?) · “Check top — is it CPU, memory, or disk?” (Semak top — adakah CPU, memori, atau disk?)

Latihan: (1) Dalam terminal Linux anda, bina pokok projek kecil dengan `mkdir` dan `touch`, salin dan pindahkan beberapa perkara, kemudian padamkannya — sambil menceritakan setiap arahan dengan kuat. (2) Cipta `hello.sh` yang mengandungi `echo "hello from $(whoami)"`, jadikannya boleh dilaksanakan dengan `chmod +x`, jalankannya dengan `./hello.sh`. (3) Jalankan `tail -f` pada sebuah fail log (di Ubuntu: `sudo tail -f /var/log/syslog`) dan perhatikan baris-baris tiba. (4) Jurnal: terangkan kepada seorang rakan khayalan mengapa server menjalankan Linux, dalam tiga ayat.

Pencapaian: tanpa nota, anda boleh menavigasi ke mana-mana sahaja dalam pokok fail, mencipta/menyalin/memindah/memadam fail, menerangkan `chmod 755`, dan menggunakan `grep` untuk mencari sesuatu perkataan dalam fail. Jika mana-mana daripadanya masih memerlukan rujukan, luangkan dua hari lagi di sini. Modul ini menjadi penanggung beban untuk segala-galanya.

Modul 2 (Minggu 3–4): Asas rangkaian + akaun AWS anda

Idea besarnya: sebuah rangkaian ialah komputer-komputer yang menghantar sampul surat beralamat. Setiap mesin mendapat satu **IP address** (alamat IP — seperti `172.31.8.14`, sebuah alamat jalan berangka). Data dicincang menjadi **packet** (paket — sampul surat) dan dihalakan lompatan demi lompatan menuju alamat destinasi. Setibanya, satu nombor **port** menyatakan program mana sampul itu ditujukan — bangunan yang sama, beribu-ribu pintu bernombor: port **22** ialah SSH, **80** ialah web tidak tersulit (**HTTP**), **443** ialah web tersulit (**HTTPS**), **5432** ialah PostgreSQL. “Open port 443” bermaksud “benarkan sampul yang beralamat ke pintu 443.”

Private lawan public: rumah anda dan setiap rangkaian cloud menggunakan semula **julat IP private** (`10.x.x.x`, `172.16–31.x.x`, `192.168.x.x`) yang hanya berfungsi di dalam rangkaian setempat; sebuah **IP public** boleh dicapai daripada seluruh internet. Pemisahan inilah batu asas keselamatan cloud: perkara yang tidak perlu berhadapan dengan internet langsung tidak diberi alamat public.

DNS — buku telefon internet. Manusia menggunakan nama (`example.com`); paket memerlukan nombor. **DNS** menterjemah: mesin anda bertanya kepada server DNS “apakah IP untuk `example.com`?”, mendapat nombornya, kemudian menyambung. Separuh daripada

semua gangguan misteri melibatkan DNS; jenaka industri “it’s always DNS” wujud kerana ia sering benar. `nslookup example.com` melakukan carian itu secara manual.

HTTP — cara web berkomunikasi. Sebuah **client** (pelayar) menghantar permintaan — satu **method** (GET = ambil, POST = hantar) berserta laluan — dan **server** membalas dengan satu **status code** (kod status): 200 OK, 301 berpindah, 403 dilarang, 404 tidak ditemui, 500 ralat server, 502/503 “server di belakang saya rosak/terlebih beban.” Hafal enam itu; sebagai jurutera anda akan membacanya setiap hari. `curl -I https://example.com` menunjukkan kepada anda baris status dan header secara langsung.

Firewall: senarai peraturan yang menentukan paket mana boleh lalu, mengikut sumber, destinasi, dan port — “benarkan 443 dari mana-mana; benarkan 22 hanya dari pejabat; tolak selebihnya.” Dalam AWS, firewall per-server dipanggil **security group**, dan yang tersalah konfigurasi ialah lubang keselamatan #1 golongan pemula. **Latency** (kelewatan, dalam milisaat) dan **bandwidth** (kapasiti sesaat) melengkapkan perbendaharaan kata: jarak mewujudkan latency, sebab itulah cloud mempunyai region di seluruh dunia.

Sekarang, akaun AWS anda — separuh kedua modul ini. Pergi ke <https://aws.amazon.com/free> dan cipta akaun Free Tier (e-mel, telefon, kad kredit/debit untuk identiti — kad itu tidak dicaj secara bererti jika anda mengikuti disiplin teardown kursus ini). **Free Tier** (peringkat percuma) memberi elaun bulanan perkhidmatan asas (termasuk 750 jam/bulan sebuah server EC2 kecil pada tahun pertama anda) dan setiap makmal dalam kursus ini direka untuk muat di dalamnya. Kemudian, sebelum apa-apa lagi, tiga langkah keselamatan yang tidak boleh dirunding — melakukannya *ialah* latihan pertama kerjaya keselamatan anda:

1. **MFA pada pengguna root.** Pengguna root ialah kunci induk akaun. Tambah multi-factor authentication (aplikasi pengesah telefon) dalam IAM — Security credentials. Kemudian berhenti menggunakan root untuk kerja harian.
2. **Cipta seorang pengguna IAM admin** (Makmal 5 menerangkan IAM secara mendalam; buat masa ini: console → IAM → Users → cipta pengguna dengan capaian admin) dan log masuk sebagai pengguna itu mulai sekarang.
3. **Amaran bil (billing alert).** Console → Billing → Budgets → cipta sebuah **zero-spend budget** (templat AWS yang menge-mel anda sebaik sahaja apa-apa mula dibilkan). Hidupkan juga amaran penggunaan Free Tier. Jurutera yang tidak boleh mengawal kos ialah satu liabiliti; anda baru 20 minit memiliki akaun dan sudah mendahului ramai profesional. (Rujukan: <https://docs.aws.amazon.com/cost-management/latest/userguide/budgets-managing-costs.html>)

Perbendaharaan kata:

Istilah	Maksud
IP address	Alamat rangkaian berangka sesebuah mesin, cth. 172.31.8.14.
Packet	Satu sampul data beralamat; semua trafik ialah aliran paket-paket ini.

Istilah	Maksud
Port	Pintu bernombor pada sesebuah mesin yang mengenal pasti program mana trafik ditujukan: 22 SSH, 80 HTTP, 443 HTTPS.
IP private / public	Alamat yang sah hanya di dalam rangkaian setempat / yang boleh dicapai daripada seluruh internet.
DNS	Sistem yang menterjemahkan nama (example.com) kepada alamat IP. "It's always DNS."
HTTP / HTTPS	Protokol permintaan-balasan web / yang sama, disulitkan dengan TLS.
Status code	Ringkasan balasan server: 200 OK, 404 tidak ditemui, 500 ralat server, 503 terlebih beban.
Firewall	Senarai peraturan yang menentukan paket mana boleh lalu, mengikut sumber, destinasi, port.
Security group	Firewall per-server milik AWS. Tersalah konfigurasinya ialah lubang klasik golongan pemula.
Latency / bandwidth	Kelewatan (ms) / kapasiti (sesaat). Jarak mewujudkan latency.
Client / server (peranan)	Si penanya dan si penjawab dalam mana-mana perbualan rangkaian.
AWS Free Tier	Elaun percuma bulanan pada akaun AWS baharu — keseluruhan bajet kursus ini.
Pengguna root	Identiti induk akaun AWS. Pasangkan MFA, kemudian berhenti menggunakannya.
Billing alert / budget	E-mel automatik apabila perbelanjaan melepasi ambang. Milik anda ditetapkan pada \$0.

Video untuk modul ini:

Video	Saluran	Tempoh	Pautan
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=JbIYCM48to

Video	Saluran	Tempoh	Pautan
AWS Networking Basics — VPC & Subnets	KodeKloud	~30 min	https://www.youtube.com/watch?v=QM63dyA_4Pc

Sebut dengan kuat: “What’s the IP, and is it public or private?” (Apakah IP-nya, dan adakah ia public atau private?) · “Is port 443 open in the security group?” (Adakah port 443 terbuka dalam security group itu?) · “Curl it — what status code do you get?” (Curl sahaja — apa status code yang anda dapat?) · “Did DNS resolve? Check with nslookup before blaming the server.” (Adakah DNS berjaya resolve? Semak dengan nslookup sebelum menyalahkan server.)

Latihan: (1) ping google.com (catatkan latency-nya), nslookup google.com (perhatikan DNS memulangkan beberapa IP), curl -I https://aws.amazon.com (baca baris status dan tiga header, kemudian rujuk maksud setiap satu). (2) Cari IP private mesin anda (ip addr di Linux) dan IP public anda (cari “what is my IP”) — terangkan dalam jurnal anda mengapa kedua-duanya berbeza. (3) Lengkapkan persediaan akaun AWS: MFA, pengguna admin, zero-spend budget — tangkap layar bajet itu untuk jurnal anda. (4) Lukis daripada ingatan: komputer riba → carian DNS → permintaan HTTPS melalui port 443 → firewall → web server. Tiga kali, sehingga ia menjadi automatik.

Pencapaian — penghujung Peringkat 0: anda boleh menceritakan apa yang berlaku apabila anda menaip URL dan menekan Enter — DNS, IP, port, firewall, permintaan HTTP, status code — dalam masa kurang dua minit, menggunakan setiap istilah dengan betul; dan akaun AWS anda wujud dengan MFA dan amaran bajet \$0. Anda kini tahu lebih banyak tentang cara internet berfungsi berbanding kebanyakan orang yang menggunakannya untuk mencari rezeki. Peringkat 1 ialah tempat anda mula membina di atasnya.

PERINGKAT 1 — TERAS CLOUD, SECARA HANDS-ON (Minggu 5-12)

Bagaimana peringkat ini berfungsi. Setiap satu daripada lima perkhidmatan di bawah ialah sebuah **Makmal (Lab)**: satu matlamat, garis besar langkah-langkah (cukup terperinci untuk diikuti, cukup ringkas supaya anda terpaksa berfikir — pemikiran itulah pembelajarannya), apa yang anda pelajari, dan — sentiasa — **teardown** (merobohkan semula). Disiplin teardown penting atas dua sebab: ia memastikan anda kekal dalam Free Tier, dan “tidak meninggalkan apa-apa berjalan yang tidak diperlukan” ialah refleks profesional yang benar-benar disiasat oleh penemu duga. Bina semula setiap makmal sekurang-kurangnya dua kali: sekali mengikut garis besar, sekali daripada ingatan. Binaan kedua itulah saat ilmu berpindah ke tangan anda. Peruntukkan kira-kira satu minggu setengah bagi setiap makmal; gunakan lebih masa untuk kerosakan, kerana benda pasti akan rosak, dan menyahpepijatnya ialah pengajaran terbaik yang kursus ini tidak mampu skripkan.

Pertama, tiga idea yang membingkaikan segala yang akan anda bina. **Region dan Availability Zone:** sebuah Region ialah kelompok geografi pusat data AWS (pilih satu yang dekat dengan anda dan kekal di dalamnya — sumber dalam satu region tidak kelihatan dari region lain,

kekeliruan #1 “ke mana perginya server saya?”). Sebuah **Availability Zone (AZ)** ialah pusat data terasing di dalam region; sistem yang serius berjalan dalam dua supaya kegagalan satu bangunan tidak menumbangkannya. **Console lawan CLI:** console (<https://console.aws.amazon.com>) ialah panel kawalan web AWS — bagus untuk belajar dan melihat-lihat; **AWS CLI** (aws dalam terminal anda) melakukan segala yang console lakukan, secara boleh diskrikan. Anda akan bermula di console dan bergraduasi ke CLI, kerana Peringkat 2 mengautomasikan segala yang anda lakukan di sini secara manual.

Makmal 1 (Minggu 5-6): EC2 — server pertama anda

EC2 (Elastic Compute Cloud) menyewakan mesin maya, dipanggil **instance**. Inilah perbuatan asasi kejuruteraan cloud: sebuah server Linux sebenar, di internet, dalam 60 saat, secara percuma.

Matlamat: melancarkan sebuah server Linux, SSH masuk ke dalamnya, menjadikannya menghidangkan satu halaman web kepada dunia, kemudian memusnahkannya.

Garis besar langkah: 1. Console — EC2 — Launch instance. Namakannya. Pilih **Amazon Linux 2023** sebagai **AMI** (Amazon Machine Image — cakera templat yang server anda boot daripadanya) dan **t2.micro** atau **t3.micro** sebagai **instance type** (saiznya; ini termasuk Free Tier). 2. Cipta sebuah **key pair**; sebuah fail private-key .pem dimuat turun. Itulah kunci SSH anda dari Modul 1 — jagainya, `chmod 400-kannya`. 3. Dalam tetapan rangkaian, benarkan SSH (port 22) dari “My IP” sahaja — anda kini tahu tepat apa maksud peraturan security-group ini — dan benarkan HTTP (port 80) dari mana-mana. 4. Lancarkan, tunggu “running,” salin IP public, kemudian dari terminal anda: `ssh -i mykey.pem ec2-user@<public-ip>`. Tarik nafas: anda berada di dalam sebuah komputer di pusat data AWS. 5. Di server: `sudo dnf install -y nginx && sudo systemctl start nginx && sudo systemctl enable nginx`. Kemudian layari `http://<public-ip>` — itulah web server anda menjawab kepada dunia. 6. Gantikan halaman lalai: `echo "<h1>Built by me, on EC2</h1>" | sudo tee /usr/share/nginx/html/index.html`. Muat semula. Tangkap layar untuk jurnal. 7. Meninjau-ninjau seperti seorang jurutera: `top`, `df -h`, `sudo tail -f /var/log/nginx/access.log` sambil anda memuat semula halaman itu — saksikan lawatan anda sendiri tiba dalam log.

Apa yang anda pelajari: AMI, instance type, key pair, security group dalam situasi sebenar, SSH ke server sebenar, memasang dan menjalankan sebuah perkhidmatan Linux, membaca lognya — iaitu, gerak fizikal harian pekerjaan ini.

Teardown: EC2 — pilih instance — Instance state — **Terminate**. Sahkan ia menyatakan “terminated.” Free Tier memberi 750 jam/bulan untuk satu instance micro, jadi biarpun ditinggalkan hidup ia tidak akan dibilkan — tetapi robohkannya juga. Tabiatlah tujuannya.

Makmal 2 (Minggu 7): S3 — fail yang tidak pernah hilang

S3 (Simple Storage Service) ialah object storage (storan objek): baldi tanpa dasar dengan ketahanan sebelas-sembilan untuk fail. Ia jawapan lalai kepada “di mana kita letak fail?” dan, apabila tersalah konfigurasi, punca kebocoran data paling terkenal dalam sejarah — sebab itulah makmal ini separuh storan, separuh keselamatan.

Matlamat: mencipta sebuah bucket, mengendalikannya dari CLI, menghoskan laman web statik kecil, dan memahami dengan tepat apa maksud “public bucket.”

Garis besar langkah: 1. Console → S3 → Create bucket (nama adalah unik secara global — namaanda-lab-2026 boleh digunakan). Perhatikan **Block Public Access** dihidupkan secara lalai. Muat naik mana-mana fail melalui console; muat turunnya kembali. 2. Pasang AWS CLI secara tempatan dan jalankan `aws configure` dengan sebuah **access key** yang anda cipta untuk pengguna admin IAM anda (access key ialah nama pengguna+kata laluan programatik untuk API — layannya seperti kata laluan, jangan sekali-kali letakkannya dalam kod; anda akan menghayati peraturan ini di Makmal 5). 3. Dari terminal anda: `aws s3 ls · aws s3 cp notes.txt s3://yourname-lab-2026/ · aws s3 sync ./myfolder s3://yourname-lab-2026/backup/`. Rasakan perbezaannya: console itu melawat; CLI itu kejuruteraan. 4. Laman web statik: buat bucket kedua, hidupkan static website hosting, muat naik sebuah `index.html`, dan tambah **bucket policy** public-read yang didokumenkan (sebuah dokumen kebenaran JSON — baca baris demi baris: *siapa* boleh buat *apa* kepada *bucket mana*). Halaman anda kini berada di internet tanpa sebarang server langsung. 5. Terokai **storage class** (Standard — Infrequent Access — Glacier: lebih murah per GB, lebih perlahan/mahal untuk diambil semula) dan tetapkan satu **lifecycle rule** (“pindahkan objek ke IA selepas 30 hari”) — rasa pertama anda tentang pengurusan kos automatik.

Apa yang anda pelajari: object storage lawan cakera, CLI dan access key, bucket policy dan capaian public (secara sengaja, bukan tidak sengaja), tier storan sebagai tuas kos.

Teardown: kosongkan kedua-dua bucket, padamkan kedua-dua bucket, dan — penting — nyahaktifkan mana-mana access key yang tidak anda gunakan. Elaun percuma S3 kecil (5 GB) tetapi fail-fail ini bersaiz kilobait; disiplinnnya, sekali lagi, itulah tujuannya.

Makmal 3 (Minggu 8–9): VPC — rangkaian yang anda miliki

VPC (Virtual Private Cloud) ialah kepingan rangkaian AWS yang peribadi dan berpagar khas untuk anda. Sehingga kini anda menggunakan yang lalai tanpa memandangnya; jurutera membina milik mereka sendiri, kerana ayat “*the database sits in a private subnet with no internet route*” (database itu berada dalam subnet private tanpa laluan internet) ialah separuh daripada keselamatan cloud, dan anda bakal menjadikannya benar dengan tangan anda sendiri.

Matlamat: membina rangkaian dua peringkat — subnet public untuk web server, subnet private untuk database masa hadapan — dan membuktikan separuh yang private itu tidak boleh dicapai dari internet.

Garis besar langkah: 1. Console → VPC → Create VPC. Berikannya blok alamat `10.0.0.0/16` — **notasi CIDR**, di mana /16 bermaksud “16 bit pertama tetap, selebihnya milik saya”: 65,536 alamat private. 2. Cipta dua **subnet**: `10.0.1.0/24` (public, dalam AZ-a) dan `10.0.2.0/24` (private, dalam AZ-b). Subnet ialah blok lebih kecil di dalam VPC, hidup dalam tepat satu AZ. 3. Cipta sebuah **Internet Gateway** (pintu VPC ke internet) dan lampirkannya. Cipta sebuah **route table** dengan peraturan `0.0.0.0/0 → internet gateway` (“apa-apa yang bukan setempat pergi ke pintu internet”) dan kaitkannya dengan subnet *public* sahaja. Subnet private hanya menyimpan laluan setempat — ketiadaan laluan itulah *keselamatannya*. 4. Lancarkan satu instance EC2 micro dalam setiap subnet (yang public dengan IP public, yang private

tanpa). 5. Buktinya: SSH ke instance public — berjaya. Cuba alamat instance private dari komputer riba anda — tergantung selama-lamanya, dan kini anda boleh menyatakan dengan tepat sebabnya. Kemudian SSH *dari* instance public *ke* yang private (ia boleh dicapai dari dalam VPC): kotak public itu bertindak sebagai **bastion host**, satu corak piawai industri yang baru anda temui dengan membinanya. 6. Konsep bonus untuk dirujuk dan dijumpai: sebuah **NAT gateway** membolehkan mesin private mencapai *keluar* (untuk kemas kini) sambil kekal tidak boleh dicapai dari luar — tetapi ia dibilkan sejam, jadi bacalah tentangnya, jangan binanya.

Apa yang anda pelajari: CIDR, subnet, route table, internet gateway, pemisahan public/private, bastion host — tepat butiran rangkaian dalam setiap deskripsi kerja cloud, sebagai ingatan otot.

Teardown: tamatkan (terminate) kedua-dua instance dahulu, kemudian padamkan VPC (yang turut menyapu subnet, route table, dan gateway bersamanya). Sahkan dalam EC2 bahawa tiada apa-apa yang menyatakan “running.”

Makmal 4 (Minggu 10): RDS — database yang anda tidak perlu jagakan

RDS (Relational Database Service) ialah database terurus: AWS menjalankan enjin database (PostgreSQL, MySQL...) dan mengendalikan sandaran, tampalan (patching), dan failover, manakala anda memiliki data dan pertanyaan (query). “Managed service” (perkhidmatan terurus) ialah tawar-menawar teras cloud — tukarkan sedikit kawalan untuk mengurangkan banyak kerja tidak terbeza — dan makmal inilah tempat anda merasai tawar-menawar itu.

Matlamat: melancarkan sebuah database PostgreSQL dalam subnet private, menyambung kepadanya dari sebuah instance EC2, dan memahami sandaran serta multi-AZ — kemudian merobohkannya dengan segera, kerana RDS ialah makmal yang paling mudah tertinggal berjalan secara tidak sengaja.

Garis besar langkah: 1. Bina semula VPC Makmal 3 dengan pantas (binaan kedua daripada ingatan — ini ulangan berjarak yang disengajakan), sambil menambah satu lagi subnet *private* dalam AZ lain, kerana RDS memerlukan sebuah **subnet group** yang merentangi dua AZ. 2. Console — RDS — Create database — PostgreSQL — **Free tier template** (ini praupilih db.t3.micro/db.t4g.micro, single-AZ). Tetapkan kata laluan master. Letakkannya dalam VPC anda, **Public access: No**, dalam sebuah security group yang membenarkan port 5432 *hanya dari security group web server* — satu peraturan yang merujuk peraturan lain dan bukannya sebuah IP. Elegan, dan piawai. 3. Lancarkan sebuah EC2 micro dalam subnet public, pasang klien postgresql, dan sambung: `psql -h <rds-endpoint> -U postgres`. **Endpoint** itu ialah nama DNS, bukan IP — AWS mungkin memindahkan mesin di bawahnya, dan nama itu mengikutnya. (Modul 2 sudah pun membayar sewanya.) 4. Pada prompt `psql`: cipta sebuah jadual, masukkan tiga baris, pilih semula (select) kesemuanya. Anda tidak memerlukan kedalaman SQL hari ini; anda perlu pernah menyentuhnya. 5. Lawati, jangan hidupkan: tetapan **automated backups** (pemulihan point-in-time daripada snapshot malam + log), dan pilihan **Multi-AZ** (sebuah standby langsung dalam AZ lain dengan **failover** automatik — kira-kira dua kali ganda kos, sebab itulah prod berkata ya dan makmal berkata tidak). Ambil satu **snapshot** manual, carinya dalam console, fahami bahawa anda boleh memulihkan sebuah klon daripadanya. 6. Soalan jurnal yang berbaloi sepuluh minit: apa,

sebenarnya, yang AWS lakukan untuk anda di sini yang jika tidak akan anda lakukan sendiri pada pukul 2 pagi? (Tampalan, sandaran, failover, perkakasan.) Jawapan itulah jawapan temu duga tentang managed services.

Apa yang anda pelajari: database terurus, subnet group, peraturan security-group-ke-security-group, endpoint, snapshot, multi-AZ/failover — berserta satu demonstrasi langsung tentang membayar dengan kawalan untuk membeli kebolehpercayaan.

Teardown: padamkan instance RDS (tolak snapshot akhir untuk makmal; ambil perhatian bahawa prod akan mengambilnya), padamkan snapshot manual (snapshot dibilkan untuk storan!), tamatkan EC2, padamkan VPC. Semak papan pemuka bil anda keesokan harinya — membacanya setiap minggu ialah tabiat Peringkat 3 yang bermula sekarang.

Makmal 5 (Minggu 11-12): IAM — siapa boleh buat apa

IAM (Identity and Access Management) menentukan manusia mana dan program mana boleh buat apa kepada sumber mana. Ia tidak membilkan apa-apa, tidak membekalkan apa-apa — dan ia adalah perkhidmatan yang paling banyak diaudit, paling banyak disiasat dalam temu duga, dan paling banyak terlibat dalam pencerobohan dalam AWS. Butiran keselamatan dalam deskripsi kerja (“least-privilege access controls,” “IAM policies”) bermaksud makmal ini.

Matlamat: mencipta pengguna, kumpulan, policy, dan — yang paling penting — sebuah **role**, serta menghayati least privilege dengan merasai AWS menolak permintaan anda.

Garis besar langkah: 1. Konsep dahulu, lima minit: seorang **user** ialah identiti untuk manusia atau program; sebuah **group** menggabungkan pengguna; sebuah **policy** ialah dokumen JSON yang memberikan kebenaran (“benarkan `s3:GetObject` pada `arn:aws:s3:::my-bucket/*`”); sebuah **role** ialah identiti dengan policy tetapi *tanpa kata laluan* yang di-assume (diambil alih) buat sementara oleh pihak yang dipercayai — cara server dan perkhidmatan mendapat kebenaran tanpa sebarang rahsia tersimpan. 2. Cipta seorang pengguna `readonly-rita` dalam kumpulan dengan policy terurus `AWS ReadOnlyAccess`. Log masuk sebagai dia dalam tetingkap pelayar peribadi: dia boleh melihat segalanya, tetapi setiap butang cipta/padam gagal dengan penolakan yang jelas. Baca salah satu ralat itu sepenuhnya — belajar mentafsir “not authorized to perform X on Y” ialah kemahiran kerja harian. 3. Tulis custom policy pertama anda dalam penyunting JSON: benarkan `s3:ListBucket` dan `s3:GetObject` pada satu bucket tertentu. Lampirkannya pada pengguna baharu; sahkan dia boleh membaca bucket itu dan *tiada yang lain*. Anda kini telah melaksanakan least privilege, bukan sekadar mentakrifkannya. 4. Role itu, ganjarannya: cipta sebuah role untuk EC2 dengan `AmazonS3ReadOnlyAccess`, lancarkan sebuah instance micro dengan role itu dilampirkan, SSH masuk, dan jalankan `aws s3 ls` — ia berfungsi dengan **tiada access key langsung pada mesin itu**. Instance itu meng-assume role tersebut dan menerima kelayakan berhayat pendek secara automatik. Inilah corak keselamatan tunggal paling penting dalam AWS: *role untuk mesin, jangan sekali-kali kunci pada mesin*. Sebutkannya dua kali. 5. Audit akaun anda sendiri seperti seorang pro: adakah root ber-MFA (Minggu 3)? Ada access key melebihi 90 hari? Adakah mana-mana pengguna mempunyai lebih kebenaran daripada yang digunakannya? Jalankan **credential report** IAM dan bacanya. Ritual sepuluh minit ini, setiap bulan, ialah senarai semak kebersihan IAM Peringkat 3 yang sedang dilahirkan.

Apa yang anda pelajari: user/group/policy/role, membaca dan menulis JSON policy, least privilege yang dikuatkuasakan melalui eksperimen, role-bukan-kunci, dan audit keselamatan pertama anda.

Teardown: padamkan pengguna ujian dan kelayakan mereka, tamatkan instance itu, simpan ilmu tentang role selama-lamanya.

Perbendaharaan kata untuk Peringkat 1 (satu jadual, kelima-lima makmal):

Istilah	Maksud
Region / Availability Zone	Kelompok geografi pusat data AWS / satu pusat data terasing di dalamnya. Sistem serius merentangi dua AZ.
EC2 / instance	Perkhidmatan sewaan mesin maya / satu server yang disewa.
AMI	Amazon Machine Image — cakera templat yang sebuah instance boot daripadanya.
Instance type	Saiz/spesifikasi yang anda pilih (t3.micro = kecil, Free Tier).
Security group	Firewall per-sumber: port mana, dari sumber mana.
Key pair	Kunci SSH private/public untuk mencapai instance anda.
S3 / bucket / object	Object storage / satu bekas bernama / satu fail tersimpan.
Storage class / lifecycle rule	Tier harga-kelajuan untuk objek / peraturan automatik yang memindahkannya ke tier lebih murah mengikut usia.
Bucket policy	Dokumen JSON yang menyatakan siapa boleh buat apa kepada sesebuah bucket. Yang public menjadi tajuk berita.
AWS CLI / access key	Antara muka terminal kepada AWS / kelayakan programatik untuknya (layan seperti kata laluan).
VPC / subnet	Kepingan rangkaian private anda / blok lebih kecil daripadanya yang hidup dalam satu AZ, public atau private.
CIDR	Notasi blok alamat: $10.0.0.0/16$ = "16 bit pertama tetap, 65,536 alamat milik saya."
Internet gateway / route table	Pintu internet VPC / peraturan yang menentukan ke mana trafik dihantar. Tiada laluan = tidak boleh dicapai = selamat.
NAT gateway	Membolehkan mesin private memanggil keluar tanpa boleh dicapai masuk. Dibilkan

Istilah	Maksud
	sejam — kenalnya, jangan biarkannya melahu.
Bastion host	Mesin public yang diperkukuh, yang anda SSH melaluinya untuk mencapai mesin-mesin private.
RDS / endpoint	Perkhidmatan relational database terurus / nama DNS yang anda sambung kepadanya.
Snapshot / Multi-AZ / failover	Salinan pada satu titik masa / standby langsung dalam AZ kedua / pertukaran automatik kepadanya.
IAM user / group / policy / role	Identiti / gabungan identiti / pemberian kebenaran JSON / identiti boleh-assume tanpa kata laluan — cara mesin mendapat kebenaran.
Least privilege	Peraturan emas: kebenaran minimum yang menyelesaikan tugas, tidak lebih.
Managed service	AWS menjalankan kerja tidak terbeza (tampalan, sandaran, failover); anda menyimpan data dan keputusan.

Video untuk peringkat ini:

Video	Saluran	Tempoh	Pautan
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
AWS Networking Basics — VPC & Subnets	KodeKloud (tonton semula selepas Makmal 3 — kesannya berbeza sekarang)	~30 min	https://www.youtube.com/watch?v=QM63dyA_4Pc
The AWS Shared Responsibility Model	Digital Cloud Training	4 min	https://www.youtube.com/watch?v=ESPBBEK-cvo

Sebut dengan kuat: “It’s in a private subnet; there’s no route to the internet gateway.” (Ia berada dalam subnet private; tiada laluan ke internet gateway.) • “The security group only

allows 5432 from the web tier's security group.” (Security group itu hanya membenarkan 5432 dari security group tier web.) · “The instance uses a role — there are no keys on the box.” (Instance itu menggunakan role — tiada kunci pada mesin itu.) · “Terminate it when you're done; nothing idles in this account.” (Terminate apabila anda selesai; tiada apa-apa yang melahu dalam akaun ini.)

Pencapaian — penghujung Peringkat 1: binaan gauntlet. Dalam satu sesi, daripada ingatan: VPC dengan subnet public/private — web server EC2 (public, dengan role dilampirkan, menghadirkan sebuah halaman) — RDS PostgreSQL (private, hanya boleh dicapai dari web server) — sebuah bucket S3 yang boleh dibaca oleh instance itu melalui role-nya — kemudian robohkan kesemuanya dengan bersih. Bawah tiga jam bermakna anda bersedia untuk Peringkat 2. Juga: **mulakan persediaan CLF-C02 sekarang** (kursus penuh freeCodeCamp — <https://www.youtube.com/watch?v=7HKot-brXFE> — pada kelajuan 1.25× sebagai ulang kaji; edisi klasik 14 jam ialah <https://www.youtube.com/watch?v=NhDYbskXRgc>), dan duduki peperiksaan **AWS Cloud Practitioner** sekitar Minggu 14–16.

PERINGKAT 2 — AUTOMASI (Minggu 13–20)

Idea peringkat ini: segala yang anda klik di Peringkat 1, kini akan anda lakukan dengan kod. Ejekan lembut industri untuk mengklik-klik console ialah **ClickOps**; frasa deskripsi kerja untuk penggantinya ialah “*design, develop, and deploy cloud infrastructure using infrastructure as code*.” Peringkat inilah perbezaan antara seseorang yang pernah *menggunakan* AWS dan seseorang yang syarikat sanggup bayar untuk *menjalankannya*.

Modul 6 (Minggu 13–15): Penskripan — Bash dan Python untuk ops, berserta Git

Penskripan Bash ialah meletakkan arahan-arahan Modul 1 ke dalam sebuah fail supaya komputer mengulanginya dengan sempurna. Pelajari, mengikut urutan ini: **pemboleh ubah** (`NAME="web-1"`), **command substitution** (`TODAY=$(date +%F)`), **if** (`if [ -f "$FILE" ]; then ... fi`), **gelung** (`for f in *.log; do gzip "$f"; done`), **exit code** (setiap arahan memulangkan 0 untuk kejayaan, bukan-sifar untuk kegagalan; `&&` merantai atas kejayaan — skrip membuat keputusan dengan ini), dan membaca **argumen** (`$1, $2`). Itulah 90% daripada Bash ops sebenar. Tulis skrip pertama anda yang benar-benar berguna minggu ini: `backup.sh` — tar sebuah direktori, namakan arkib dengan tarikh hari ini, `aws s3 cp`-kannya ke sebuah bucket, padamkan arkib tempatan melebihi 7 hari, dan `echo` satu baris kejayaan/kegagalan ke sebuah fail `log`. Satu skrip itu ialah pemboleh ubah, substitution, bersyarat, exit code, dan CLI dalam satu artifak — dan satu butiran pada resume anda (“automated backups to S3”).

Python ialah abang kepada Bash: lebih baik untuk apa-apa yang melibatkan logik, data, atau berkomunikasi dengan API. Anda memerlukan Python-ops yang berfungsi, bukan kedalaman kejuruteraan perisian: pemboleh ubah dan jenis, senarai dan kamus (dictionary), `if/for`, fungsi, membaca/menulis fail, pengendalian ralat `try/except`, dan memasang pustaka (library) dengan `pip`. Kemudian berkenalanlah dengan **boto3**, pustaka AWS untuk Python:

`import boto3; ec2 = boto3.client("ec2"); ec2.describe_instances()` — tiba-tiba pengetahuan Peringkat 1 anda boleh diprogramkan. Tulis `audit.py`: senaraikan setiap instance EC2 dalam akaun dengan nama, jenis, keadaan, dan masa pelancarannya, dan cetak satu baris amaran untuk apa-apa yang berjalan melebihi 24 jam. Tahniah — anda telah menulis alat kawalan kos sebenar yang majikan sanggup bayar.

Git ialah sistem kawalan versi tempat semua kerja kejuruteraan hidup. Konsep: sebuah **repository** (folder yang keseluruhan sejarahnya direkodkan), sebuah **commit** (satu snapshot tersimpan berserta mesej), sebuah **branch** (satu garis kerja selari), sebuah **remote** (salinan di GitHub), dan sebuah **pull request** (meminta sebuah branch disemak dan digabungkan). Kitaran hariannya ialah enam arahan: `git init` / `git clone`, `git status`, `git add`, `git commit -m "message"`, `git push`, `git pull`. Cipta akaun GitHub hari ini, cipta sebuah repo bernama `cloud-journey`, dan mulai sekarang **setiap skrip dan konfigurasi yang anda tulis dalam kursus ini di-commit ke dalamnya**. Dalam sepuluh bulan, sejarah commit itu ialah bukti awam bertarih tentang segala yang kursus ini dakwa anda mampu lakukan.

Perbendaharaan kata:

Istilah	Maksud
Skrip	Fail berisi arahan yang dijalankan dari atas ke bawah — atom automasi.
Pemboleh ubah / argumen	Nilai bernama dalam skrip / nilai yang dihantar masuk semasa anda menjalankannya (\$1).
Exit code	Isyarat kejayaan (0) atau kegagalan (bukan-sifar) setiap arahan — cara skrip membuat keputusan.
Cron	Penjadual Linux: jalankan sebuah skrip setiap malam pukul 2 pagi, selama-lamanya.
Python / pip	Bahasa pengaturcaraan kegemaran dunia ops / pemasang pakejnya.
boto3	Pustaka Python untuk memandu AWS — console, sebagai kod.
try/except	“Cuba ini; jika gagal, buat itu pula” milik Python — cara skrip gagal dengan anggun.
Git / repository / commit	Kawalan versi / sejarah terekod satu projek / satu snapshot tersimpan berserta mesej.
Branch / merge	Satu garis kerja selari / membawanya kembali ke garis utama.
GitHub / remote / pull request	Laman pengehosan / salinan cloud repo anda / permintaan menggabungkan branch yang disemak dan dibincangkan.
README	Dokumen muka hadapan sebuah repo yang

Istilah	Maksud
	menerangkan apa ia dan cara menjalankannya. Perekrut membaca ini.

Video untuk modul ini:

Video	Saluran	Tempoh	Pautan
Learn Python — Full Course for Beginners	freeCodeCamp	~4.5 jam (tonton sepanjang 3 minggu itu)	https://www.youtube.com/watch?v=rfscVS0vtbw

Sebut dengan kuat: “I’ll script it — it’ll be wrong once, then never again.” (Saya akan skripkannya — ia akan silap sekali, kemudian tidak lagi selama-lamanya.) · “Check the exit code before the next step runs.” (Semak exit code sebelum langkah seterusnya berjalan.) · “Commit it with a message that says why, not what.” (Commit dengan mesej yang menyatakan mengapa, bukan apa.) · “It’s in the repo — pull the latest.” (Ia ada dalam repo — pull yang terkini.)

Latihan: (1) `backup.sh`, seperti dispesifikasikan di atas, dijadualkan setiap malam dengan cron pada sebuah instance EC2 makmal. (2) `audit.py`, seperti dispesifikasikan di atas. (3) Kedua-duanya di-commit ke `cloud-journey` dengan README yang menerangkan setiap satu. (4) Rosakkan skrip anda sendiri dengan sengaja (laluhan yang salah, bucket yang tiada) dan buatkan ia gagal *dengan lantang dan jelas* — pengendalian ralat ialah bahasa kasih sayang dunia ops.

Pencapaian: skrip sandaran anda terselamat apabila dijalankan dua kali berturut-turut dan dengan folder yang hilang (tiada ranap, mesej yang jelas); audit Python anda berjalan terhadap akaun sebenar anda; repo anda menunjukkan seminggu commit.

Modul 7 (Minggu 16–17): Terraform — infrastructure as code

Idea besarnya: berbanding mengklik sumber sehingga wujud, anda menulis fail teks yang *mengisytiharkan* apa yang sepatutnya wujud — “satu VPC, dua subnet, satu instance, tag-tag ini” — dan **Terraform** membuatkan AWS sepadan dengan fail itu. Deklaratif, bukan imperatif: anda menyatakan destinasi, bukan selekoh-selekoahnya. Mengapa setiap deskripsi kerja memintanya: fail-fail itu hidup dalam Git, jadi infrastruktur menjadi **disemak** (pull request sebelum perubahan), **boleh diulang** (fail yang sama membina dev, staging, dan prod secara serupa), **boleh diundur** (undur dengan mengembalikan commit), dan **boleh diaudit** (sejarah menyatakan siapa mengubah apa, bila, mengapa). Console ialah bengkel; Terraform ialah kilang.

Kitaran teras yang akan anda jalankan beratus-ratus kali: `terraform init` (muat turun **provider** AWS — plugin yang menterjemahkan fail anda kepada panggilan API) — `terraform plan` (larian percubaan yang mencetak dengan tepat apa yang akan dicipta/diubah/dimusnahkan — baca setiap baris; jurutera yang apply tanpa membaca plan menyebabkan gangguan) — `terraform apply` (jadikannya nyata) — `terraform destroy` (nyahjadikannya — teardown sebagai satu arahan, yang pada tahap ini akan anda dapati

indah). Terraform menjejaki apa yang dibinanya dalam sebuah **state file** — ingatannya tentang realiti; hilangkannya atau ubah realiti di belakangnya secara manual (“drift”) dan kesengsaraan menyusul. Sumber diisytiharkan dalam **HCL**, sebuah bahasa konfigurasi yang mudah dibaca:

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

Makmalnya (inilah modul ini): bina semula gauntlet Peringkat 1 dalam Terraform, secara berperingkat. (1) Init sebuah repo terraform-labs; tulis konfigurasi untuk sebuah bucket S3 bertag sahaja; plan, baca, apply, semak console — ia ada di sana; destroy. (2) Kembangkan konfigurasi: VPC, dua subnet, internet gateway, route table — binaan Makmal 3, kini ~60 baris HCL. (3) Tambah security group dan sebuah instance EC2 dengan `variables.tf` (region, saiz instance) dan `outputs.tf` (cetak IP public selepas apply). (4) Ubah jenis instance dalam fail dan apply semula — saksikan Terraform mengira perbezaannya dan mengubah *hanya itu*. Kelakuan berpandukan-diff inilah keseluruhan keajaibannya. (5) Musnahkan segalanya, sahkan console kosong, commit, dan tag repo itu v1.0. Dokumentasi rujukan berada di <https://developer.hashicorp.com/terraform> — tanda bukukannya; membaca dokumentasi provider ialah separuh daripada kerja Terraform sebenar.

Perbendaharaan kata:

Istilah	Maksud
IaC (Infrastructure as Code)	Mengisytiharkan infrastruktur dalam fail teks berversi yang diubah menjadi realiti oleh sebuah alat.
Terraform / HCL	Alat IaC multi-cloud yang dominan / bahasa konfigurasinya.
Provider	Plugin yang menterjemahkan fail anda kepada panggilan API satu cloud (di sini: AWS).
plan / apply / destroy	Diff larian percubaan / menjadikannya nyata / menyahjadikan segalanya. Baca setiap plan.
State file	Rekod Terraform tentang apa yang dibinanya — ingatannya tentang realiti. Lindunginya.
Drift	Realiti berubah di belakang Terraform (seseorang mengklik). Musuh.
Variable / output / module	Input konfigurasi / hasil yang dicetak (sebuah IP, sebuah URL) / cebisan konfigurasi terbungkus yang boleh diguna semula.
Deklaratif lawan imperatif	Menyatakan destinasi lawan menskriplan

Istilah	Maksud
CloudFormation	selekeh. Terraform bersifat deklaratif. Perkhidmatan IaC milik AWS sendiri — idea sama, AWS sahaja. Deskripsi kerja menerima kedua-duanya; pelajari Terraform dahulu.

Video untuk modul ini:

Video	Saluran	Tempoh	Pautan
Terraform explained in 15 mins	TechWorld with Nana	18 min	https://www.youtube.com/watch?v=l5k1ai_GBDE
Terraform Course — Automate your AWS cloud infrastructure	freeCodeCamp	~2.5 jam	https://www.youtube.com/watch?v=SLB_c_ayRMo

Sebut dengan kuat: “Nothing changes in prod except through Terraform.” (Tiada apa-apa berubah dalam prod kecuali melalui Terraform.) · “Show me the plan before you apply.” (Tunjukkan plan itu sebelum anda apply.) · “That was clicked in manually — that’s drift; let’s import it or remove it.” (Itu diklik masuk secara manual — itulah drift; mari kita import atau buangkannya.) · “It’s in the module; reuse it, don’t rewrite it.” (Ia ada dalam module; guna semula, jangan tulis semula.)

Latihan: makmal di atas, berserta: (1) cipta drift dengan sengaja — ubah tag instance itu secara manual dalam console, jalankan terraform plan, dan saksikan Terraform menyedarinya; jurnalkan apa yang dicadangkannya. (2) Tulis satu perenggan bahasa mudah dalam README repo itu: “mengapa ini mengalahkan mengklik.” Jika anda tidak boleh menulisnya, anda belum memilikinya.

Pencapaian: daripada direktori kosong, anda boleh menghidupkan stack VPC + instance dengan `init/plan/apply` dan membuangnya dengan `destroy`, sambil menerangkan dengan kuat apa yang dilakukan setiap arahan — tanpa nota.

Modul 8 (Minggu 18–20): CI/CD, Docker, dan pandangan pertama pada Kubernetes

CI/CD (Continuous Integration / Continuous Delivery) ialah tali sawat robot yang dipasangkan pada repo Git anda: pada setiap push, ia secara automatik menyemak, menguji, membina, dan — apabila dikonfigurasi — men-deploy perubahan anda. Tujuannya ialah keluaran kecil, kerap, dan membosankan berbanding yang jarang dan menakutkan. **GitHub Actions** ialah sistem CI/CD terbina dalam GitHub: sebuah **workflow** ialah fail YAML dalam `.github/workflows/` yang menyatakan “pada push, jalankan langkah-langkah ini pada sebuah **runner** baharu (VM sementara).” Makmal: tambah sebuah workflow pada `terraform-labs` yang menjalankan `terraform fmt -check` dan `terraform validate` pada setiap push — push satu fail yang sengaja dirosakkan dan saksikan ✖ merah, betulkannya

dan saksikan ✓ hijau. Anda kini mempunyai sebuah pipeline; sekecil mana pun, ia spesies yang sama dengan yang ada dalam setiap deskripsi kerja. (Langkah kedua, dalam Projek 1: sebuah workflow yang menjalankan terraform plan pada pull request, supaya perubahan infrastruktur yang dicadangkan menunjukkan diff-nya dalam semakan.)

Docker membungkus sebuah aplikasi berserta segala yang diperlukannya ke dalam sebuah **container** — bekas makan tengah hari tertutup yang berjalan secara serupa di komputer riba anda, di EC2, dan di mana-mana sahaja, menamatkan wabak purba “works on my machine.” Sebuah **image** ialah resipi beku (dibina daripada sebuah **Dockerfile**, fail teks ringkas berisi langkah binaan: mula FROM sebuah image asas, COPY masuk aplikasi anda, takrifkan arahan mula); sebuah **container** ialah satu instance image yang sedang berjalan; sebuah **registry** (Docker Hub, atau **ECR** milik AWS) ialah tempat image di-push dan di-pull. Container *bukan* VM: ia berkongsi kernel Linux hos, jadi ia bermula dalam kira-kira sesaat dan anda boleh menjalankan berpuluh-puluh di atas sebuah instance micro. Makmal: pada sebuah instance EC2, pasang Docker, docker run hello-world, kemudian docker run -d -p 80:8080 <a sample web image> dan layarnya; kemudian tulis Dockerfile lima baris yang menghidangkan halaman statik Makmal 2 anda daripada image asas nginx, binanya, jalankannya, dan push ke Docker Hub. Pelajari kata kerja harian: docker ps, docker logs, docker exec -it <id> bash (sebuah shell *di dalam* container), docker stop.

Kubernetes (K8s) — untuk kursus ini, kemahiran tahap bacaan, dilabelkan secara jujur. Apabila sebuah syarikat menjalankan ratusan container merentasi banyak mesin, sesuatu mesti menjadualkannya, memulakan semula yang ranap, menskalakan yang sibuk, dan menghalakan trafik antaranya: pengatur (orchestrator) itulah Kubernetes. Pelajari peta konsepnya sekarang — sebuah **cluster** berisi **node** menjalankan **pod** (unit terkecil boleh-deploy, biasanya satu container); sebuah **deployment** mengisytiharkan “pastikan 3 replika pod ini hidup” dan cluster itu menjadikannya benar secara berterusan (deklaratif lagi — Kubernetes ialah falsafah Terraform diterapkan pada perisian yang sedang berjalan); sebuah **service** memberi pod sebuah alamat yang stabil. Tawaran terurus AWS ialah **EKS**. Deskripsi kerja junior mahukan tepat literasi ini berserta kefasihan Docker; kemahiran operasi K8s sebenar (dan pensijilan CKA) ialah matlamat Tahun 2 yang kukuh, bukan Bulan 5. Kursus ini memberitahu anda secara jujur yang mana satu.

Perbendaharaan kata:

Istilah	Maksud
CI/CD	Pipeline automatik yang menyemak, membina, menguji, dan menghantar setiap perubahan.
GitHub Actions / workflow / runner	CI/CD terbina dalam GitHub / fail YAML yang mentakrifkan sebuah pipeline / VM sementara yang melaksanakannya.
YAML	Format konfigurasi berasaskan takuk (indentation) untuk pipeline dan Kubernetes. Takuk ialah makna — berhati-hatilah.
Docker / image / container	Kit alat container / resipi beku berlapis /

Istilah	Maksud
Dockerfile	satu instance-nya yang sedang berjalan. Fail teks ringkas berisi langkah-langkah yang membina sebuah image.
Registry / ECR	Tempat image di-push dan di-pull / registry milik AWS.
Kubernetes (K8s) / cluster / node	Pengatur container / kumpulan mesinnya / satu mesin di dalamnya.
Pod / deployment / service	Unit terkecil boleh-deploy / “pastikan N replika hidup,” dikuatkuasakan secara berterusan / alamat stabil di hadapan pod-pod.
EKS	Control plane Kubernetes terurus milik AWS.
Rollback	Kembali pantas kepada versi sebelumnya apabila sesuatu keluaran bermasalah — jaring keselamatan yang dijadikan rutin oleh CI/CD.

Video untuk modul ini:

Video	Saluran	Tempoh	Pautan
DevOps CI/CD Explained in 100 Seconds	Fireship	2 min	https://www.youtube.com/watch?v=scEDHsr3APg
What is DevOps? REALLY understand it	TechWorld with Nana	~15 min	https://www.youtube.com/watch?v=0yWAtQ6wYNM
Docker in 100 Seconds	Fireship	2 min	https://www.youtube.com/watch?v=Gjnup-PuquQ
Docker Tutorial for Beginners [FULL COURSE in 3 Hours]	TechWorld with Nana	3 jam	https://www.youtube.com/watch?v=3c-iBn73dDE
Kubernetes explained in 15 mins	TechWorld with Nana	~16 min	https://www.youtube.com/watch?v=VnvRFRk_51k

Sebut dengan kuat: “Don’t merge until the pipeline is green.” (Jangan merge sehingga pipeline hijau.) · “It’s containerized — same image in dev and prod.” (Ia sudah dikontainerkan)

— image yang sama dalam dev dan prod.) · “Exec into the container and check its logs.” (Exec masuk ke container itu dan semak log-nya.) · “K8s keeps three replicas up; kill one and watch it come back.” (K8s memastikan tiga replika hidup; matikan satu dan saksikan ia kembali.)

Latihan: dua makmal di atas, berserta: (1) jurnalkan jawapan satu perenggan untuk “VM lawan container lawan serverless — bila setiap satu?” (Serverless in 100 Seconds oleh Fireship — https://www.youtube.com/watch?v=W_VV2Fx32_Y — melengkapkan pilihan ketiga itu). (2) Tamatkan instance makmal Docker; sahkan image anda terus hidup dalam registry — perhatikan bahawa *artifak* kini hidup lebih lama daripada *server*, iaitu keseluruhan pandangan dunia deployment moden dalam satu pemerhatian.

Pencapaian — penghujung Peringkat 2: GitHub anda menunjukkan: sebuah repo berisi skrip ops Bash + Python yang berfungsi, sebuah repo Terraform yang membina dan memusnahkan stack sebenar, sebuah workflow Actions yang hijau, dan sebuah Dockerfile yang di-push ke registry — dan anda boleh menerangkan setiap fail dalam temu duga. Profil GitHub itu bukan lagi milik seorang pelajar. Ia milik seorang jurutera junior.

PERINGKAT 3 — OPERASI (Minggu 21-28)

Idea peringkat ini: membina sistem membuatkan anda diambil bekerja; *menjalankannya* ialah pekerjaan sebenar. Butiran deskripsi kerja yang dijawab peringkat ini ialah separuh operasi — “monitor infrastructure health,” “participate in incident response, including log analysis,” “identifying, analyzing, and resolving infrastructure vulnerabilities,” “manage cloud costs.” Minggu seorang jurutera cloud kebanyakannya ialah peringkat ini.

Modul 9 (Minggu 21-24): Pemantauan, pengelogan, dan insiden

Pemantauan — tahu sebelum pengguna tahu. CloudWatch ialah perkhidmatan kebolehcerahan (observability) terbina dalam AWS. Tiga primitif: **metrik** (nombor merentas masa — % CPU, % cakera, kiraan permintaan, kiraan ralat), **alarm** (peraturan yang memerhati sesuatu metrik: “jika CPU > 80% selama 5 minit — maklumkan”), dan **dashboard** (metrik disusun pada satu skrin). Notifikasi mengalir melalui **SNS** (Simple Notification Service — sebuah topic yang anda terbitkan kepadanya, pelanggannya menerima e-mel/panggilan pager). Kemahirannya ialah *apa* yang harus di-alarm-kan: panggil manusia hanya untuk perkara yang memerlukan manusia, atau orang akan belajar mengabaikan pager (**alert fatigue** — mod kegagalan yang mendahului banyak gangguan terkenal). Empat isyarat emas (golden signals) yang berbaloi dihafal: **latency, traffic, errors, saturation** — betapa perlahan, betapa sibuk, betapa rosak, betapa penuh.

Pengelogan — tahu mengapa. Metrik menyatakan ada sesuatu yang tidak kena; log menyatakan apa yang berlaku. CloudWatch Logs memusatkannya: sebuah **agent** pada setiap instance menghantar fail seperti `/var/log/nginx/access.log` ke **log group**, di mana **Logs Insights** membolehkan anda membuat pertanyaan merentasi mesin (“kira respons 5xx mengikut minit untuk sejam terakhir”). Pemusatan penting kerana server kini boleh dibuang

(anda membuktikannya di Modul 8) — log mesti hidup lebih lama daripada mesin yang menulisnya.

Makmal (Minggu 21–22): dirikan sebuah web server kecil yang dipantau, semuanya dalam Terraform (stack anda dari Modul 7, dilanjutkan): EC2 + nginx + agent CloudWatch; sebuah topic SNS yang menge-mel anda; satu alarm untuk CPU tinggi dan satu lagi untuk kegagalan semakan status instance. Kemudian serang diri sendiri: SSH masuk dan jalankan pembakar CPU (`yes > /dev/null &` beberapa kali); saksikan metrik memanjat, alarm menyala, e-mel tiba; matikan proses-proses itu; saksikan ia pulih. Kemudian buat pertanyaan terhadap access log anda sendiri dalam Logs Insights. Anda kini telah menyaksikan kitaran penuh kesan—maklum—diagnosis—selesai pada sistem yang anda bina sendiri.

Tindak balas insiden — separuh manusianya. Sebuah **insiden** ialah “sistem ini tidak okey” yang tidak dirancang; tahap **severity** (sev-1 = pelanggan terputus, semua turun padang) menetapkan skala tindak balas. Kitaran profesional: **kesan** (alarm itu, bukan e-mel pelanggan, idealnya) — **triage** (betapa teruk, siapa diperlukan) — **mitigasi** (hentikan pendarahan *dahulu* — undur, mula semula, failover; punca akar kemudian) — **selesaikan** — **post-mortem**: semakan bertulis *tanpa salah-menyalah* tentang apa yang berlaku, mengapa, dan apa yang akan mencegah ulangan. Tanpa salah-menyalah bukanlah kelembutan; ia kejuruteraan: orang yang dihukum menyembunyikan maklumat, dan maklumat tersembunyi menyebabkan gangguan berulang. **On-call** ialah giliran siapa yang membawa pager; deskripsi kerja menyebutnya, penemu duga bertanya tentangnya, dan jawapan jujur anda selepas modul ini ialah “saya telah mensimulasikannya dan saya tahu kitarannya.”

Runbook — butiran dokumentasi, dijadikan konkrit. Sebuah runbook ialah resipi langkah demi langkah untuk satu situasi operasi, ditulis supaya seorang yang tertekan pada pukul 3 pagi boleh mengikutinya: simptom — semakan (arahan yang tepat) — pembedaan (arahan yang tepat) — eskalasi (siapa harus dikejutkan jika tidak berjaya). **Makmal (Minggu 23–24):** tulis dua runbook dalam repo anda — “web server down” dan “disk filling up” — kemudian *latih* keduanya: rosakkan benda itu, ikuti dokumen anda sendiri secara harfiah, dan betulkan setiap langkah yang terbukti kabur. Kemudian jalankan satu **game day** penuh: minta seorang rakan (atau sebuah AI) merosakkan stack makmal anda secara rahsia; anda menerima panggilan, mendiagnosis daripada metrik dan log, memitigasi, dan menulis post-mortem dalam jurnal anda. Post-mortem itu ialah sebuah kisah temu duga, dan kisah yang baik.

Perbendaharaan kata:

Istilah	Maksud
CloudWatch	Perkhidmatan pemantauan AWS: metrik, alarm, dashboard, log.
Metrik / alarm / dashboard	Nombor merentas masa / peraturan yang menyala berdasarkannya / satu skrin berisi kesemuanya.
SNS	Perkhidmatan notifikasi yang alarm terbitkan kepadanya — e-mel, SMS, pager.
Golden signals	Latency, traffic, errors, saturation — empat nombor yang menggambarkan kesihatan

Istilah	Maksud
Alert fatigue	mana-mana perkhidmatan. Terlalu banyak panggilan tidak boleh ditindak → panggilan diabaikan → yang benar terlepas. Musuh reka bentuk alarm.
Log group / Logs Insights	Tempat log terpusat mendarat / bahasa pertanyaan ke atasnya.
Insiden / severity	Kemerosotan tidak dirancang / kedudukan keterukannya (sev-1 = paling teruk).
Triage / mitigasi / selesai	Nilai dengan pantas / hentikan pendarahan dahulu / betulkan sebenar-benarnya.
Post-mortem	Semakan bertulis tanpa salah-menyalah: apa, mengapa, apa yang mencegah ulangan.
Runbook	Resipi kalis-3-pagi untuk satu situasi: simptom, semakan, pembetulan, eskalasi.
On-call / game day	Giliran pager / insiden latihan yang disengajakan.
MTTR	Mean time to recovery — metrik ops yang dioptimumkan pasukan matang.

Sebut dengan kuat: “Did we find out from the alarm or from a customer?” (Adakah kita tahu daripada alarm atau daripada pelanggan?) · “Mitigate first — root cause after we’re stable.” (Mitigasi dahulu — punca akar selepas kita stabil.) · “Is there a runbook for this? There will be by tomorrow.” (Ada runbook untuk ini? Akan ada menjelang esok.) · “What did the post-mortem conclude, and what’s the prevention item?” (Apa kesimpulan post-mortem itu, dan apakah langkah pencegahannya?)

Pencapaian: post-mortem game day anda wujud, tanpa salah-menyalah, dan menamakan satu pencegahan konkrit yang kemudiannya benar-benar anda laksanakan (satu alarm ditambah, satu runbook dibetulkan).

Modul 10 (Minggu 25–28): Operasi keselamatan dan pengurusan kos

Operasi keselamatan — kebersihan, bukan kepahlawanan. Shared responsibility model (model tanggungjawab bersama) dahulu (tonton semula: <https://www.youtube.com/watch?v=ESPBBEK-cvo>): AWS melindungi cloud itu sendiri; *anda* melindungi apa yang anda letakkan di dalamnya — dan kebanyakan pencerobohan sebenar ialah salah konfigurasi pelanggan, bukan kegagalan AWS. Senarai semak keselamatan operasi anda, diamalkan sehingga membosankan:

1. **Kebersihan IAM** (bulanan): MFA di mana-mana; tiada access key berhayat panjang jika sebuah role sudah memadai; jalankan credential report; padamkan pengguna dan kunci yang tidak digunakan; soalkan setiap * dalam setiap policy. Anda membina refleksi ini di Makmal 5 — kini ia satu entri kalendar.

2. **Patching (tampalan)**: perisian tanpa tampalan ialah cara kebanyakan pencerobohan bermula. Pada instance anda: pasang kemas kini mengikut jadual (dan ketahuilah bahawa **SSM Patch Manager** mengautomasikannya untuk seluruh armada); dalam dunia container, patching sering bermaksud membina semula image daripada asas yang dikemas kini — hubungkan itu dengan Modul 8.
3. **Sandaran — yang telah diuji**: sandaran yang tidak diuji ialah harapan, bukan pelan. Snapshot RDS, **versioning** S3 (setiap penulisan ganti menyimpan versi lama — kawalan anti-tersilap dan anti-ransomware). Makmal: snapshot database makmal anda, *pulihkannya* ke sebuah instance baharu, sahkan baris-barisnya, robohkan. Kini barulah anda boleh berkata “telah diuji” dalam temu duga.
4. **Guardrail**: hidupkan **CloudTrail** (log audit setiap panggilan API dalam akaun — siapa buat apa, bila) dan imbas peristiwa semalam sekali; hidupkan percubaan percuma **GuardDuty** (pengesanan ancaman automatik) dan baca apa yang diperhatikannya. Kenali frasa **encryption at rest and in transit** (penyulitan semasa disimpan dan semasa bergerak) dan bahawa dalam AWS ia kebanyakannya sebuah kotak semak yang disokong oleh **KMS** — syarat asas, sentiasa hidup.

Pengurusan kos — kemahiran yang membuatkan junior kelihatan senior. Butiran deskripsi kerjanya kata demi kata: “*manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging.*” Bagaimana meter berjalan: compute dibilkan sesaat selagi ia *hidup* (melahu ≠ percuma — “kita terlupa matikannya” ialah pembaziran klasik), storan per GB-bulan, dan **egress** (data *keluar* dari AWS) per GB manakala data masuk percuma — kejutan bil yang terkenal. Tuas-tuasnya, mengikut urutan yang boleh ditarik seorang junior: **matikannya** (sistem dev pada waktu malam; tabiat teardown anda, diindustrikan) → **rightsize** (kebanyakan server terlebih saiz; semak sejarah CPU CloudWatch dan kecilkan) → **tag segalanya** (Project, Owner, Environment — perbelanjaan tanpa tag ialah perbelanjaan tanpa akauntabiliti; kuatkuasakan tag dalam Terraform anda) → **tier storan** (lifecycle rule dari Makmal 2) → ketahui bahawa **kapasiti reserved/Savings Plans** (komit 1–3 tahun, jimat 30–70%) dan **Spot** (sehingga 90% diskaun, boleh terganggu) wujud masing-masing untuk workload stabil dan batch. Alatan: **Cost Explorer** (perbelanjaan akaun, digrafkan — bacanya setiap minggu, selama-lamanya) dan **AWS Budgets** (amaran \$0 anda dari Minggu 3, kini difahami sebagai ahli terkecil sebuah keluarga yang serius). Disiplin yang lebih luas dipanggil **FinOps** — berbaloi dua minit: https://www.youtube.com/watch?v=Y-c_xw9bHFw.

Perbendaharaan kata:

Istilah	Maksud
Shared responsibility model	AWS melindungi cloud; anda melindungi apa yang anda letakkan di dalamnya. Kebanyakan pencerobohan ialah separuh yang kedua.
CloudTrail	Log audit akaun: setiap panggilan API, oleh siapa, bila.
GuardDuty	Pengesanan ancaman automatik AWS ke atas log dan trafik.

Istilah	Maksud
Patching / SSM	Memasang kemas kini keselamatan / Systems Manager milik AWS, yang mengautomasikannya untuk seluruh armada.
S3 versioning	Simpan setiap versi yang ditulis ganti — kawalan anti-tersilap, anti-ransomware.
KMS / encryption at rest & in transit	Perkhidmatan kunci AWS / data dikarau pada cakera dan pada wayar. Sentiasa hidup.
Egress	Data meninggalkan AWS — dibilkan per GB; yang masuk percuma. Kejutan bil klasik.
Rightsizing	Mengecilkan sumber terlebih saiz kepada keperluan terukur. Wang percuma.
Tagging	Melabelkan setiap sumber dengan pemilik/projek/persekitaran supaya semua perbelanjaan boleh dipertanggungjawabkan.
Savings Plans / Spot	Komitmen 1–3 tahun untuk 30–70% diskaun bagi beban stabil / kapasiti lebihan boleh dituntut semula sehingga 90% diskaun untuk batch.
Cost Explorer / Budgets	Graf perbelanjaan yang anda baca setiap minggu / amaran yang bermakna anda tidak sekali-kali mengetahuinya daripada invoices.
FinOps	Disiplin menjadikan perbelanjaan cloud kelihatan, diperuntukkan, dan dioptimumkan secara berterusan.

Sebut dengan kuat: “Who has access to prod, and when did we last review the list?” (Siapa yang mempunyai capaian ke prod, dan bilakah kali terakhir kita menyemak senarai itu?) · “When did we last *restore* a backup?” (Bilakah kali terakhir kita *memulihkan* sandaran?) · “What’s untagged, and what died but is still billing?” (Apa yang tiada tag, dan apa yang sudah mati tetapi masih dibilkan?) · “It’s over-provisioned — the CPU history says we can halve it.” (Ia diperuntukkan berlebihan — sejarah CPU menyatakan kita boleh separuhkannya.)

Latihan: (1) Makmal ujian-pemulihan di atas. (2) Sebuah fail senarai-semak-keselamatan-bulanan dalam repo anda, kemudian benar-benar jalankannya terhadap akaun anda. (3) Dalam Cost Explorer, cari benda paling mahal dalam sejarah akaun anda dan terangkannya dalam satu ayat jurnal. (4) Tambah tag lalai pada setiap sumber dalam repo Terraform anda.

Pencapaian — penghujung Peringkat 3: jalankan satu audit-kendiri penuh — senarai semak keselamatan, semakan kos, ujian alarm, pemulihan sandaran — dan tulis laporan satu halaman. Anda kini boleh melakukan *pekerjaan* itu, bukan sekadar binaannya. Yang tinggal ialah membuktikannya kepada orang asing: Peringkat 4.

PERINGKAT 4 — KESEDIAAN KERJA (Bulan 8–12)

Modul 11 (Bulan 8–10): Tiga projek portfolio

Pensijilan menyatakan anda telah belajar; projek membuktikan anda boleh membina. Tiga projek, setiap satu dalam repo GitHub-nya sendiri, setiap satu dengan gambar rajah seni bina, README yang ditulis untuk pengurus pengambilan (apa, mengapa, cara menjalankannya, berapa kosnya), dan sebuah skrip teardown. Bina — tangkap layar/rakam — *musnahkan* — repo itulah artifaknya, bukan bil yang terus berjalan.

Projek 1 — Aplikasi web tiga peringkat, diautomasikan sepenuhnya (mercu tanda).

Terraform membina segalanya: VPC dengan subnet public/private merentasi dua AZ; sebuah **Application Load Balancer** (peranti penyebar trafik — baharu bagi anda, dan langkah semula jadi seterusnya dari Makmal 3) di hadapan sebuah **Auto Scaling Group** berisi instance web (AWS menambah/membuang instance mengikut beban — rujuk, pasangkan); RDS dalam subnet private; S3 untuk aset statik; alarm CloudWatch dan sebuah runbook. GitHub Actions menjalankan terraform plan pada setiap pull request dan apply pada merge — sebuah pipeline infrastruktur sebenar. Projek tunggal ini mendemonstrasikan sepuluh daripada empat belas butiran deskripsi kerja dalam jadual pemetaan; jangkakan setiap temu duga akan menyusurnya, dan berlatihlah menceritakannya dalam lima minit.

Projek 2 — Pipeline data serverless (keluasan). Tiada server langsung: sebuah fail yang mendarat dalam bucket S3 mencetuskan sebuah **fungsi Lambda** (Python anda, dijalankan AWS per-peristiwa, dibilkan per-panggilan) yang memprosesnya — hurai sebuah CSV, ringkaskannya, tulis hasilnya ke bucket kedua atau sebuah jadual DynamoDB — dengan kegagalan ditangkap, dilogkan ke CloudWatch, dan di-alarm-kan ke peti masuk anda. Deploy-nya dengan Terraform. Ia menunjukkan Python, pemikiran dipacu-peristiwa, dan keluasan melangkaui EC2. Dua minit orientasi dahulu: https://www.youtube.com/watch?v=W_VV2Fx32_Y.

Projek 3 — Pameran ops gaya-production (pembeza). Ambil Projek 1 dan kendalikannya seolah-olah ia penting: sebuah dashboard CloudWatch untuk isyarat emas; alarm dengan polisi paging yang didokumenkan; tiga runbook; prosedur sandaran-dan-pemulihan yang teruji berserta buktinya; satu pusingan pengukuhan keselamatan (audit IAM, nota patching, CloudTrail hidup) yang ditulis lengkap; satu analisis kos (“stack ini berkos \$X/bulan; inilah tiga perubahan yang akan menyeparuhkannya”); dan satu post-mortem game-day. Hampir tiada calon junior yang memiliki ini. Ia menjawab satu-satunya soalan yang benar-benar ditanya oleh temu duga — “bolehkah orang ini dipercayai dengan production?” — dengan dokumen dan bukan kata sifat.

Modul 12 (Bulan 8–12, selari): Jadual pensijilan

Pensijilan membuka penapis perekrut dan menstruktur pembelajaran anda. Laluan piawai 2026 untuk peranan ini, dengan persediaan realistik pada kadar kursus ini — perhatikan bagaimana setiap peperiksaan tiba sejurus selepas peringkat kursus yang mengajar kandungannya, sebab itulah tempoh ini lebih singkat daripada yang dikatakan internet:

Urutan	Pensijilan	Apa yang dibuktikan	Masa persediaan dari kedudukan anda	Bila mendudukinya
1	AWS Certified Cloud Practitioner (CLF-C02)	Perbendaharaan kata cloud, bil, shared responsibility	3–4 minggu (panduan menyatakan yang sama untuk pemula; Peringkat 0–1 merangkumi kebanyakannya)	~ Bulan 4
2	AWS Solutions Architect Associate (SAA-C03)	Mereka bentuk seni bina AWS sebenar — kelayakan yang benar-benar ditapis oleh deskripsi kerja	6–8 minggu persediaan berfokus (anggaran piawai industri selepas CCP)	Bulan 8–9
3	HashiCorp Terraform Associate (003)	Kefasihan IaC, disahkan	2–3 minggu — selepas Modul 7 dan Projek 1 anda kebanyakannya mengulang kaji (persediaan rasmi: https://developer.hashicorp.com/terraform/tutorials/certification-003)	Bulan 10–11

Pilihan keempat, jika menyasarkan jawatan bertajuk ops: **AWS SysOps Administrator / CloudOps Associate** — kandungannya secara harfiah ialah Peringkat 3. Untuk CLF-C02, kursus penuh freeCodeCamp (<https://www.youtube.com/watch?v=7HKot-brXFE>, edisi klasik <https://www.youtube.com/watch?v=NhDYbskXRgc>) berserta peperiksaan latihan ialah laluan percuma yang lazim; untuk SAA-C03, gandingkan sebuah kursus dengan banyak ujian latihan bermasa — peperiksaannya berasaskan senario dan stamina itu penting.

Modul 13 (Bulan 11–12): Resume, temu duga, dan sepuluh soalan itu

Resume: satu halaman. Mulakan dengan kemahiran (AWS, Terraform, Python/Bash, Docker, CI/CD, CloudWatch — cerminkan perkataan deskripsi kerja itu sendiri; penapis automatik memadamkan kata kunci) dan tiga projek itu, setiap satu sebagai dua butiran berbentuk *melakukan X dengan Y mencapai Z*: “Deployed a three-tier web app on AWS with Terraform

and GitHub Actions; zero-downtime deploys via ALB + Auto Scaling.” Senaraikan pensijilan dengan tarikh. Pautkan GitHub — kemudian anggap mereka benar-benar akan membukanya, kerana yang bagus memang membukanya, dan milik anda kini menganjari lawatan itu.

Persediaan temu duga: tiga rasa untuk dilatih — trivia (jadual perbendaharaan kata Peringkat), senario (makmal yang benar-benar anda lakukan), dan tingkah laku (kisah-kisah jurnal anda, diceritakan dalam bentuk **STAR**: Situation, Task, Action, Result). Berlatih *dengan kuat*, setiap hari, selama dua minggu — idealnya dengan penemu duga AI, tanpa belas kasihan. Sepuluh soalan di bawah merangkumi yang klasik; jawapan contoh sengaja dipadatkan — kembangkan setiap satu dengan perincian makmal anda sendiri, kerana “...dan semasa saya membina ini, apa yang sebenarnya berlaku ialah...” ialah ayat yang memisahkan anda daripada calon yang hanya membaca.

Sepuluh soalan itu, dengan jawapan yang kuat:

1. **“Explain the difference between a public and a private subnet.” (Terangkan perbezaan antara subnet public dan private.)** *Route table subnet public mempunyai laluan ke internet gateway, jadi sumbernya boleh dicapai dari internet; subnet private tiada laluan sebegitu. Tier web pergi public, database pergi private, dan capaian admin mencapai mesin private melalui bastion atau SSM. Dalam makmal saya, saya membuktikan instance private tidak boleh dicapai dari komputer riba saya tetapi boleh dicapai dari tier public — laluan yang tiada itulah keselamatannya.*
2. **“What is IAM, and what is least privilege?” (Apakah IAM, dan apakah least privilege?)** *IAM mengawal identiti mana boleh melakukan tindakan mana ke atas sumber mana, melalui user, group, role, dan policy JSON. Least privilege bermaksud memberikan minimum yang menyelesaikan tugas — pengguna skrip audit saya mempunyai capaian baca-sahaja kepada tepat satu perkhidmatan, dan instance EC2 saya menggunakan role dan bukannya kunci tersimpan, jadi tiada kelayakan berhayat panjang untuk bocor.*
3. **“An EC2 web server stopped responding. Walk me through your troubleshooting.” (Sebuah web server EC2 berhenti membalas. Bimbing saya melalui penyelesaian masalah anda.)** *Skop dahulu: satu server atau semuanya — semak dashboard CloudWatch dan mana-mana alarm. Kemudian lapisan-lapisan mengikut urutan: semakan status instance; security group — adakah 80/443 benar-benar terbuka; SSH masuk — adakah nginx berjalan (`systemctl status`), adakah cakera penuh (`df -h`), adakah CPU terperap (`top`); kemudian log-log (`tail log` ralat). Mitigasi dahulu — mulakan semula perkhidmatan atau gantikan instance — dan buat punca akar selepas kita stabil. Itulah urutan yang saya latih dalam runbook saya.*
4. **“Why Terraform instead of clicking in the console?” (Mengapa Terraform dan bukannya mengklik dalam console?)** *Console tidak meninggalkan rekod yang boleh disemak dan tidak boleh membina semula apa-apa. Terraform bersifat deklaratif dan berversi: perubahan melalui pull request, konfigurasi yang sama membina persekitaran yang serupa, perubahan buruk diundur dengan mengembalikan sebuah commit, dan plan menunjukkan diff yang tepat sebelum apa-apa berlaku. Pipeline saya menjalankan plan pada setiap PR supaya diff itu sebahagian daripada semakan.*
5. **“What’s the difference between a container and a virtual machine?” (Apakah perbezaan antara container dan mesin maya?)** *Sebuah VM memvirtualkan*

perkakasan dan membawa OS penuh, mengambil beberapa minit untuk boot; sebuah container berkongsi kernel hos dan membungkus hanya aplikasi serta kebergantungannya, bermula dalam kira-kira sesaat. Container memberikan kelakuan serupa merentasi persekitaran — image yang sama berjalan di komputer riba saya dan di EC2. VM masih penting untuk pengasingan dan untuk menjalankan workload bukan-Linux.

6. **“How would you reduce our AWS bill?” (Bagaimana anda akan mengurangkan bil AWS kami?)** Mengikut urutan usaha: cari apa yang berjalan yang tidak sepatutnya — instance melahu, volume tidak terlampir, snapshot lama; Cost Explorer dan tag menjadikannya kelihatan. Kemudian rightsize berdasarkan sejarah CloudWatch. Kemudian jadualkan persekitaran dev mati di luar waktu kerja. Kemudian lifecycle-kan data S3 lama ke tier lebih murah. Kemudian komitkan workload stabil kepada Savings Plans dan letakkan batch boleh terganggu pada Spot. Dan perhatikan egress — data keluar ialah baris kejutan klasik.
7. **“What happens when you type a URL and press Enter?” (Apa berlaku apabila anda menaip URL dan menekan Enter?)** DNS me-resolve nama itu kepada sebuah IP; pelayar membuka sambungan TCP ke port 443 dan merundingkan TLS; ia menghantar sebuah HTTP GET; di hujung sana ia menemui sebuah load balancer, yang memajukannya ke instance sihat dalam subnet public; aplikasi mungkin membuat pertanyaan ke database dalam subnet private; respons kembali dengan sebuah status code — 200 jika sihat — dan pelayar memaparkannya. Saya boleh mendalami mana-mana lompitan, dan saya pernah membina setiap satunya.
8. **“A teammate’s change broke production. What happens next?” (Perubahan rakan sepasukan merosakkan production. Apa berlaku seterusnya?)** Mitigasi dahulu — undur melalui pipeline; pemulihan mengalahkan diagnosis pada saat itu. Kemudian post-mortem tanpa salah-menyalah: apa yang berlaku, mengapa sistem membenarkannya, dan guardrail mana yang tiada — satu ujian, satu semakan plan, satu alarm. Tanpa salah-menyalah penting secara praktikal: orang yang takut dihukum menyembunyikan maklumat, dan maklumat tersembunyi menyebabkan ulangan.
9. **“What is the shared responsibility model?” (Apakah shared responsibility model?)** AWS melindungi cloud — pusat data, perkakasan, hypervisor; pelanggan melindungi apa yang di dalamnya — data, IAM, peraturan rangkaian, patching, tetapan penyulitan. Kebanyakan pencerobohan cloud sebenar berada di pihak pelanggan, salah konfigurasi seperti bucket public atau kunci bocor. Jadi ‘AWS is secure’ dan ‘we are secure on AWS’ adalah ayat yang berbeza, dan yang kedua itulah tugas saya.
10. **“Tell me about a problem you debugged.” (Ceritakan tentang satu masalah yang anda nyahpeijat.)** (Milik anda, daripada jurnal, dalam bentuk STAR. Contoh rangka:) Semasa game day pemantauan saya, alarm saya menyala untuk ketepuan CPU (S). Saya perlu mencari dan menghentikan puncanya sebelum ‘pengguna’ perasan (T). Metrik menunjukkan permulaan lonjakan; Logs Insights mengaitkannya dengan proses-proses liar; saya SSH masuk, sahkan dengan top, matikannya, dan saksikan pemulihan pada dashboard (A). Selesai dalam sebelas minit, dan post-mortem itu menghasilkan sebuah alarm kiraan-proses yang kemudian saya laksanakan dalam Terraform (R).

Mekanik pencarian kerja: mohon dari Bulan 11 — selepas SAA, jangan tunggu “bersedia,” kerana temu duga ialah latihan. Tajuk sasaran: cloud engineer (junior/associate), cloud

support engineer, cloud operations engineer, junior DevOps engineer, AWS support associate. Setiap penolakan yang disertai temu duga ialah pelajaran percuma; jurnalkan apa yang mereka tanya.

Perbendaharaan kata:

Istilah	Maksud
Portfolio	Bukti awam terdokumen bahawa anda boleh membina — untuk bidang ini, repo GitHub dengan gambar rajah dan README.
Application Load Balancer (ALB)	Penyebar trafik AWS: mengagihkan permintaan merentasi instance sihat, menggugurkan yang sakit.
Auto Scaling Group	Memastikan N instance hidup dan melaraskan N mengikut beban — penyembuhan diri dan keanjalan dalam satu.
Lambda / serverless	Kod yang dijalankan AWS per peristiwa, dibilkan per panggilan — tiada server untuk diurus langsung.
DynamoDB	Jadual NoSQL serverless AWS — berpasangan secara semula jadi dengan Lambda.
STAR	Situation, Task, Action, Result — bentuk sebuah kisah temu duga yang baik.
ATS	Applicant tracking system — penapis kata kunci yang mesti dilepasi resume satu halaman anda.
CLF-C02 / SAA-C03 / Terraform Associate 003	Tiga peperiksaan anda: perbendaharaan kata, seni bina, IaC.
SysOps / CloudOps Associate	Associate AWS berfokus ops yang menjadi pilihan — Peringkat 3, dalam bentuk peperiksaan.

Pencapaian — penghujung kursus: tiga repo yang boleh difahami orang asing, tiga pensijilan dijadualkan atau lulus, sepuluh jawapan dilatih dengan kuat, permohonan sedang berjalan. Anda bukan lagi “berharap untuk masuk ke bidang cloud.” Anda seorang jurutera cloud junior yang berbekalkan bukti, sedang ditemu duga.

Silibus satu halaman

Bila	Fokus	Bukti hands-on	Bukti luaran
Minggu 1-2	Bagaimana komputer berfungsi; Linux,	Operasi fail berskrip;	—

Bila	Fokus	Bukti hands-on	Bukti luaran
	terminal, permission, SSH	skrip shell pertama	
Minggu 3–4	Rangkaian: IP, DNS, port, HTTP, firewall; akaun AWS	Akaun + MFA + bajet \$0	—
Minggu 5–6	Makmal 1: EC2	Web server di internet, dirobuhkan	—
Minggu 7	Makmal 2: S3 + CLI	Laman statik; kefasihan CLI	—
Minggu 8–9	Makmal 3: VPC	Rangkaian public/private, bukti bastion	—
Minggu 10	Makmal 4: RDS	Database private, snapshot, teardown	—
Minggu 11–12	Makmal 5: IAM; binaan semula gauntlet Peringkat 1	Stack penuh daripada ingatan, <3 jam	Jadualkan CCP
Minggu 13–15	Bash, Python/boto3, Git	backup.sh, audit.py, sejarah repo	Peperiksaan CCP (~Bulan 4)
Minggu 16–17	Terraform	Stack sebagai kod, plan/apply/destroy	—
Minggu 18–20	CI/CD, Docker, literasi K8s	Pipeline hijau; image dalam registry	—
Minggu 21–24	CloudWatch, log, insiden, runbook	Alarm menyala + post-mortem game-day	—
Minggu 25–28	Ops keselamatan; pengurusan kos	Laporan audit-kendiri; ujian pemulihan	—
Bulan 8–10	Projek Portfolio 1–3	Tiga repo terdokumen	SAA-C03 (Bulan 8–9)
Bulan 10–11	Persediaan Terraform Associate	—	Terraform Associate
Bulan 11–12	Resume, temu duga, permohonan	Sepuluh jawapan, dengan kuat	Tawaran pertama

Kata penutup daripada guru anda. Dua belas bulan itu singkat untuk pertukaran kerjaya dan panjang untuk tabiat harian, jadi inilah perjanjian jujur: orang yang menamatkan kursus ini bukanlah yang paling bijak — mereka ialah orang yang tetap menaip arahan pada hari-hari yang melelahkan juga. Setiap mesej ralat yang anda temui ialah kurikulum yang sedang berfungsi, bukan gagal; jurutera yang telah merosakkan dan membetulkan seratus perkara kecil ialah tepat apa yang dimaksudkan pengurus pengambilan dengan “pengalaman.”

Simpan jurnal itu, robohkan apa yang anda bina, jangan sekali-kali mendakwa dalam temu duga apa yang repo anda tidak dapat sokong — dan setahun dari sekarang, apabila pager berbunyi pada pukul 3 pagi, anda akan merasai sesuatu yang tidak dijangka di sebalik adrenalin itu: kecekapan. Pergilah membina.

Sumber

Kajian deskripsi kerja (Ogos 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](#). Laluan pensijilan dan tempoh persediaan: [StudyTech](#) — [AWS Certification Roadmap 2026](#) · [Cloud Evolvers](#) — [Cloud Engineer Roadmap 2026](#) · [HashiCorp](#) — [Terraform Associate 003 prep](#). Konteks gaji (median AS ≈ \$104K, julat \$85K–\$140K): [Wiz](#), di atas. Semua pautan YouTube disahkan melalui metadata YouTube semasa penulisan.

Sebuah kursus B4LCILC — jilid pendamping kepada “The Cloud Leader Course.”

PELAJARAN BAHARU — EDISI OGOS 2026

Pada Ogos 2026, pelajaran-pelajaran baharu telah ditambah kepada tiga kursus interaktif di [b4lcilc.org](#) — kursus *Pemimpin Cloud*, *Jurutera Cloud*, dan *Arkitek Cloud*. Addendum ini mengemas kini edisi bacaan dengan ringkasan setia bahan baharu itu, dalam suara yang sama seperti bahagian lain buku ini.

Sepatah kata tentang apa yang berada di tangan anda. Halaman-halaman ini ialah edisi bacaan: idea-ideanya, metafora-metaforanya, dan ayat-ayat yang berbaloi disebut dengan kuat. Versi interaktif penuh dalam talian membawa jauh lebih banyak — setiap pelajaran baharu disertakan kuiz yang menyemak kefahaman anda sebelum membenarkan anda mara, makmal amali (termasuk terminal latihan yang berkelakuan seperti yang sebenar tetapi tidak boleh menelan kos), video, latih tubi perbendaharaan kata, dan — untuk bahan capstone — latihan reka bentuk berpandu dengan senarai semak keperluan serta jawapan model yang lengkap. Baca ringkasan-ringkasan ini untuk memasukkan bentuk setiap idea ke dalam kepala anda; kemudian, apabila berkesempatan, duduki versi interaktifnya supaya ilmu itu melekat.

Penambahan ini terbahagi kepada tiga kumpulan. Untuk kursus **Pemimpin Cloud**: pengenalan lembut kepada terminal, undang-undang privasi Thailand sendiri, panduan pemimpin tentang cadangan AI dan bil AI, serta satu latihan capstone yang menghimpunkan keseluruhan kursus ke dalam satu halaman. Untuk kursus **Jurutera Cloud**: pengkomputeran serverless — corak yang menukarkan skrip operasi kecil kepada fungsi yang tidak menelan kos untuk dijalankan. Untuk kursus **Arkitek Cloud**: aritmetik janji ketersediaan, corak Retrieval-Augmented Generation yang kini ditanya pelanggan setiap minggu, dan satu latihan pengekosan penuh — menetapkan harga sebuah reka bentuk sebelum sesiapa membinanya.

Untuk kursus Jurutera Cloud

Apa sebenarnya serverless

Segala yang telah anda automatisasikan setakat ini masih menganggap ada sebuah mesin: cron job dalam makmal modul ini berjalan selama dua minit sehari pada sebuah virtual machine yang anda bayar sepanjang bulan, tampal (patch), pantau, dan suatu hari nanti nyahpepijat pada pukul 3 pagi. Serverless ialah jawapan cloud kepada ketidakpadanan itu. Server masih wujud — namanya hanyalah pemasaran — tetapi ia adalah masalah penyedia. Anda menyerahkan sebuah fungsi, penyedia menjalankannya apabila sesuatu mencetuskannya, dan anda membayar mengikut invocation (seruan), mengikut milisaat. Model ini dipanggil **FaaS — Function as a Service (Fungsi sebagai Perkhidmatan)**: **Lambda** di AWS, **Azure Functions** di Azure. Konsepnya berpindah nama demi nama.

Anatominya muat dalam satu genggaman. Sebuah **handler** ialah fungsi yang dipanggil oleh platform, dan ia menerima sebuah **event** — sebuah huraian JSON (JavaScript Object Notation) tentang apa sahaja yang berlaku: sebuah fail mendarat dalam bucket, sebuah permintaan HTTP tiba, sebuah jadual berbunyi. Duduk tegak pada yang terakhir itu: “sebuah jadual berbunyi” adalah tepat apa yang cron lakukan — kecuali sebuah Lambda pada jadual tidak memerlukan mesin yang dihidupkan antara larian. Menukar salah satu skrip operasi modul ini ialah projek serverless pertama yang klasik.

Ekonominya adalah sebab model ini wujud, dan ia memotong kedua-dua arah.

Peringkat percuma AWS merangkumi satu juta invocation Lambda sebulan, selama-lamanya — fungsi makmal anda boleh berjalan setiap minit sepanjang tahun dan tidak menelan kos apa-apa, menjadikan serverless rumah paling murah untuk gam operasi. Tetapi harga per-milisaat yang merupakan ralat pembundaran pada volum rendah menjadi wang sebenar pada volum tinggi; fungsi yang berjalan berterusan biasanya lebih murah sebagai container atau VM. Kemahiran profesionalnya ialah mengetahui di sebelah mana garisan itu sesebuah beban kerja berada.

Dua peringatan jujur. Pertama, **cold start**: fungsi yang tidak berjalan baru-baru ini mengambil beberapa ratus milisaat untuk bangun — tidak relevan bagi laporan malam, tetapi satu pertimbangan reka bentuk bagi API yang sensitif kependaman. Kedua, sebuah fungsi masih berjalan sebagai satu identiti: Lambda memikul peranan IAM (Identity and Access Management) sama seperti sebuah instance EC2, dan fungsi yang diberi kebenaran berlebihan adalah kesilapan keselamatan yang sama dalam bungkusan yang lebih kecil.
