

Kursus Arkitek Cloud (The Cloud Architect Course)

Daripada sifar kepada mereka bentuk sistem cloud — sebuah kursus B4LCILC

13 Ogos 2026

Kursus Arkitek Cloud (The Cloud Architect Course)

Daripada sifar pengetahuan kepada mereka bentuk sistem cloud yang syarikat sanggup pertaruhkannya.

Bagaimana kursus ini berfungsi

Anda bukan berlatih untuk *mengendalikan* sistem cloud. Anda berlatih untuk **mereka bentuknya**: mengambil masalah perniagaan yang berselirat (“kami perlu menjual kepada sejuta pelanggan dan tidak sekali-kali kehilangan pesanan”), mengubahnya menjadi pelan teknikal yang dilukis, dikoskan, dan boleh dipertahankan, dan kemudian meyakinkan kedua-dua jurutera yang akan membinanya dan eksekutif yang akan membayarnya. Itulah pekerjaan seorang **arkitek cloud** (juga diiklankan sebagai *solutions architect*, *cloud solutions architect*, atau *cloud domain architect*), dan ia adalah kemahiran yang boleh dipelajari dengan silibus yang jelas — inilah dia.

Kursus ini mempunyai lima peringkat. **Peringkat 1 (Minggu 1–6): Blok-blok binaan, sebagai keputusan.** Setiap komponen cloud, diajar semula bukan sebagai fakta tetapi sebagai *pilihan dengan tolak ansur (trade-off)* — kerana unit kerja seorang arkitek ialah keputusan. **Peringkat 2 (Minggu 7–14): Domain reka bentuk.** Enam tunjang Well-Architected Framework — kebolehpercayaan, keselamatan, prestasi, kos, operasi, kelestarian — setiap satu diajar secara mendalam dengan corak piawainya dan satu reka bentuk contoh. **Peringkat 3 (Minggu 15–20): Katalog corak.** Sedozen seni bina yang menyelesaikan 95% masalah sebenar, dan — lebih penting lagi — bila *tidak* patut menggunakan setiap satu. **Peringkat 4 (Minggu 21–26): Migrasi dan dunia sebenar.** Memindahkan syarikat sedia ada ke cloud, dan kekangan (undang-undang, legasi, lock-in) yang menjadikan seni bina sebenar lebih sukar daripada seni bina papan putih. **Peringkat 5 (Bulan 7–12): Kemahiran kraf.** Dokumen, gambar rajah, semakan, pembentangan, dan pensijilan yang menjadikan anda *boleh diambil bekerja* sebagai arkitek, dimahkotai tiga projek reka bentuk bertaraf portfolio.

Untuk siapa kursus ini. Ia menganggap sifar pengetahuan awal: segala yang bergantung kepadanya diajar semula secara ringkas sebelum digunakan. Jika anda telah menamatkan **Kursus Jurutera Cloud (The Cloud Engineer Course)** (atau anda bekerja secara hands-on dalam cloud hari ini), Peringkat 1 akan terasa biasa — *layari sepintas lalu, tetapi tetap lakukan latihannya*, kerana ia membingkaikan semula perkara yang anda kenali sebagai fakta menjadi perkara yang mesti anda timbangkan sebagai keputusan, dan pementakan semula itulah keseluruhan anjakan daripada jurutera kepada arkitek.

Peraturan untuk keseluruhan kursus: belajar **satu jam sehari, enam hari seminggu** — konsistensi mengalahkan intensiti. Setiap modul berakhir dengan **jadual Perbendaharaan Kata** (perkataan yang mesti anda kuasai), **Latihan** (kerja reka bentuk, bukan bacaan — seni bina dipelajari dengan melukis dan memutuskan), dan satu **Pencapaian (Milestone)** (bukti anda bersedia untuk mara; jangan langkau pencapaian yang belum dipenuhi). Dan mulai Minggu 1, simpan sebuah **Jurnal Reka Bentuk (Design Journal)**: setiap seni bina yang anda lukis, setiap tolak ansur yang anda namakan, setiap ramalan yang anda buat tentang kos atau kegagalan. Sepuluh bulan dari sekarang ia menjadi portfolio temu duga anda; tiada apa yang lebih mengagumkan panel pengambilan daripada sebuah buku nota bertarikh berisi empat puluh reka bentuk dengan nota jujur tentang apa yang anda silap.

Apa sebenarnya yang dilakukan seorang arkitek cloud

Bayangkan seorang arkitek bangunan. Dia tidak menyusun bata — tetapi dia mesti tahu dengan tepat apa yang bata boleh dan tidak boleh lakukan, berapa kosnya, dan bagaimana bangunan gagal. Dia mendengar seorang klien (“rumah keluarga, cerah, di bawah bajet ini, di atas tapak yang janggal ini”), dan mengubah kehendak serta kekangan menjadi *lukisan dan spesifikasi* yang cukup tepat sehingga pembina boleh membina daripadanya dan klien boleh meluluskannya. Kemudian dia kekal sepanjang pembinaan, menjawab soalan dan melaraskan pelan apabila tanah didapati lebih lembut daripada yang dikatakan tinjauan.

Tukarkan bata dengan server (pelayan) dan itulah pekerjaannya. Seorang arkitek cloud menghabiskan minggunya melakukan gabungan: **mendengar** pemegang taruh perniagaan dan mengekstrak keperluan sebenar daripada hajat yang kabur; **mereka bentuk** — memilih komponen, melukis gambar rajah, menulis keputusan dan sebabnya; **menyemak** reka bentuk orang lain terhadap enam tunjang; **menganggar** berapa kos sesebuah reka bentuk sebulan dan mempertahankan nombor itu di hadapan pihak kewangan; **merancang migrasi** sistem lama ke dalam cloud; **membentang** — reka bentuk yang sama diterangkan satu cara kepada jurutera dan cara yang berbeza sama sekali kepada eksekutif; dan **membimbing (mentoring)** jurutera supaya piawaian terus hidup walau berdepan tarikh akhir. Dalam banyak pasukan ada juga **pra-jualan (pre-sales)**: duduk di sebelah jurujual di hadapan bakal pelanggan, melakarkan penyelesaian yang memenangi urusan itu.

Apa yang seorang arkitek *bukan*: pengekod terbaik dalam bilik itu (biasanya bukan), orang yang paling banyak menaip (paling sedikit menaip), atau genius bersendirian yang menurunkan pelan tindakan (cara terpantas untuk diabaikan). Hasil sebenar seorang arkitek ialah **keputusan yang baik, ditulis, yang orang lain rela bina**.

Deskripsi kerja yang anda latih untuk capai

Pada Ogos 2026 kami mengumpulkan iklan jawatan dan templat Cloud/Solutions Architect sebenar — templat cloud-architect seorang perekrut (KORE1), deskripsi kerja sebuah firma pengambilan (4 Corner Resources), takrifan Microsoft sendiri tentang peranan arkitek dalam Azure Well-Architected Framework, sebuah iklan Solutions Architect pra-jualan (Arpio, pemulihan bencana AWS), sebuah iklan Solution Architect perusahaan (Intel, melalui Built In), dan sebuah iklan Cloud Domain Architect perusahaan (Halliburton, Azure diutamakan). Buangkan perisa syarikat dan dua belas keperluan yang sama muncul berulang kali. Kursus ini dibina mengikut senarai itu — inilah dengan tepat di mana setiap satu diajar:

Apa yang diminta oleh deskripsi kerja (kata-kata mereka, diparafrasakan)	Di mana kursus ini mengajarnya
“Design scalable and secure cloud architectures tailored to business and technical requirements” (Mereka bentuk seni bina cloud berskala dan selamat mengikut keperluan perniagaan dan teknikal) — reka bentuk penyelesaian hujung ke hujung	Peringkat 1–3, kemuncak di Peringkat 5
“Deep platform expertise” (Kepakaran platform mendalam) dalam AWS dan/atau Azure — compute, storage, rangkaian, IAM, struktur akaun	Peringkat 1 + trek pensijilan di Peringkat 5
“Run Well-Architected reviews” (Menjalankan semakan Well-Architected) terhadap enam tunjang	Peringkat 2 (tunjang-tunjang), Peringkat 5 Modul 12 (menjalankan semakan)
“Lead migration/modernization initiatives — which workloads move as-is, get rearchitected, or retire” (Memimpin inisiatif migrasi/pemodenan — workload mana berpindah seadanya, direka bentuk semula, atau dipersarakan)	Peringkat 4 Modul 10 (7 R, perancangan gelombang)
“Design landing zones, account structure, guardrails, and reference patterns teams deploy within” (Mereka bentuk landing zone, struktur akaun, guardrail, dan corak rujukan tempat pasukan men-deploy)	Peringkat 4 Modul 10
“Integrate security and compliance into the design rather than bolting it on” (Menyepadukan keselamatan dan pematuhan ke dalam reka bentuk dan bukannya menampalnya kemudian) — zero trust, identiti	Peringkat 2 Modul 5, Peringkat 4 Modul 11
Ketersediaan tinggi dan pemulihan bencana — “RTO/RPO gaps, downtime costs, ransomware exposure” (jurang RTO/RPO, kos downtime, pendedahan ransomware)	Peringkat 2 Modul 4, Peringkat 3 Modul 9 (multi-region)
“Own the cloud cost model — tagging, showback, reserved capacity, rightsizing” (Memiliki model kos cloud — tagging, showback, kapasiti reserved, rightsizing); menganggar kos penyelesaian	Peringkat 2 Modul 6, Peringkat 5 Modul 13 (pengekosan cadangan)
Reka bentuk seni bina rangkaian — VPC, hub-and-spoke, sambungan hibrid	Peringkat 1 Modul 2, Peringkat 3 Modul 9, Peringkat 4

Apa yang diminta oleh deskripsi kerja (kata-kata mereka, diparafrasakan)	Di mana kursus ini mengajarnya
“Create architectural documentation, diagrams, and standards”; “maintain Architecture Decision Records” (Mencipta dokumentasi seni bina, gambar rajah, dan piawaian; menyelenggara Architecture Decision Records)	Peringkat 1 Modul 3 (gambar rajah), Peringkat 5 Modul 12 (ADR, set gambar rajah)
Komunikasi pemegang taruh — “present technical concepts to C-level and technical audiences”; “defend architectural decisions to security, finance, and engineering” (Membentang konsep teknikal kepada khalayak C-level dan teknikal; mempertahankan keputusan seni bina kepada keselamatan, kewangan, dan kejuruteraan)	Peringkat 5 Modul 13
“Mentor the cloud and platform engineers who build against your standards” (Membimbing jurutera cloud dan platform yang membina mengikut piawaian anda); sokongan pra-jualan — demo, POC, RFP	Peringkat 5 Modul 13

Pensijilan, disahkan terhadap pasaran yang sama: tangga piawai ialah **AWS Certified Solutions Architect – Associate (SAA-C03)** — kelayakan tunggal paling banyak diminta, lazimnya **2–3 bulan** persediaan — diikuti **AWS Certified Solutions Architect – Professional (SAP-C02)**, lazimnya **4–8 bulan** lagi; syarikat berat-Azure meminta **AZ-305 (Azure Solutions Architect Expert)**; perusahaan besar kadangkala menambah **TOGAF** untuk kaedah seni bina perusahaan. Pelan penuhnya di Peringkat 5, Modul 14.

PERINGKAT 1 — BLOK-BLOK BINAAN, SEBAGAI KEPUTUSAN (Minggu 1–6)

Graduan Kursus Jurutera Cloud: layari penerangannya sepintas lalu, tetapi lakukan setiap latihan. Faktanya sama; soalnya baharu.

Modul 1 (Minggu 1–2): Apa itu cloud, dan bagaimana arkitek berfikir

Cloud dalam satu perenggan (penyegar sifar-pengetahuan). Cloud ialah *komputer orang lain, disewa mengikut jam, diurus oleh perisian*. AWS, Microsoft Azure, dan Google Cloud menjalankan gudang-gudang berisi server (**data center**, iaitu pusat data) yang dikelompokkan ke dalam **Region** geografi (cth., “Asia Pacific (Bangkok)”); setiap region mengandungi beberapa **Availability Zone (AZ)** terasing — bangunan berasingan dengan bekalan kuasa

tersendiri, cukup dekat untuk sambungan pantas, cukup jauh supaya satu banjir atau kebakaran tidak boleh melumpuhkan dua. Anda menyewa kepingan-kepingan semua ini sesaat: mesin mentah (**IaaS** — anda mengurus segala di atasnya), platform terurus (**PaaS** — anda hanya membawa aplikasi anda), atau perisian siap (**SaaS** — anda hanya menggunakannya). Itulah keseluruhan substratnya. Segala yang direka bentuk seorang arkitek disusun di atasnya.

Sekarang langkah arkitek: ubah setiap fakta menjadi soalan. Seorang jurutera belajar “AZ ialah pusat data terasing.” Seorang arkitek terus bertanya: “*Berapa banyak AZ yang layak diterima workload ini?*” — kerana dua AZ lebih mahal daripada satu, tiga lebih mahal daripada dua, dan sebuah laman risalah pemasaran tidak layak menerima apa yang layak diterima sebuah sistem pembayaran. Inilah idea pertama dan terpenting kursus ini:

Seni bina ialah disiplin menjadikan tolak ansur eksplisit. Hampir tiada komponen yang salah — hanya komponen yang salah *untuk workload ini, bajet ini, pasukan ini, tarikh akhir ini.*

Segi tiga tolak ansur. Setiap reka bentuk berunding antara tiga penjuru: **pantas** (prestasi — pantas untuk pengguna, pantas untuk dibina), **murah** (bil bulanan rendah, usaha kejuruteraan rendah), dan **berdaya tahan** (terselamat daripada kegagalan, berskala, kekal selamat). Anda boleh menolak ke arah mana-mana dua penjuru; penjuru ketiga membayar harganya. Prototaip sebuah syarikat pemula patut pantas dan murah — daya tahan boleh menunggu. Lejar teras sebuah bank mesti berdaya tahan dan pantas — ia tidak akan murah. Apabila seorang pemegang taruh berkata “kami mahu ketiga-tiganya,” tugas anda ialah tersenyum dan bertanya yang mana satu mereka mahu *paling*, kerana reka bentuk tidak boleh bermula selagi mereka tidak menjawab. Lukis segi tiga ini di bahagian atas setiap reka bentuk yang anda buat dalam kursus ini, dan tandakan di mana workload itu berada. Ia akan menyelamatkan anda daripada seribu pertengkaran.

Keperluan: bahan mentah reka bentuk. Arkitek memisahkan **keperluan berfungsi (functional requirements)** (apa yang sistem lakukan — “pelanggan boleh memesan makan tengah hari”) daripada **keperluan bukan berfungsi (non-functional requirements, NFR)** (sebaik mana ia mesti melakukannya — “bawah 2 saat, untuk 10,000 pelajar serentak, 99.9% masa, dalam lingkungan PDPA”). Pemula taasub dengan senarai pertama; arkitek memperoleh gaji mereka pada yang kedua, kerana NFR-lah yang sebenarnya menentukan seni bina. Dua soalan ajaib mengekstrak NFR daripada pemegang taruh yang tidak sedar mereka memilikinya: “*What happens to the business if this is down for an hour?*” (Apa berlaku kepada perniagaan jika ini terhenti selama sejam?) dan “*What does success look like at ten times today’s size?*” (Bagaimana rupa kejayaan pada sepuluh kali ganda saiz hari ini?)

Perbendaharaan kata:

Istilah	Maksud
Region / Availability Zone (AZ)	Region: kelompok geografi pusat data yang anda pilih untuk beroperasi di dalamnya. AZ: pusat data terasing (atau kumpulan kecil) di dalamnya — unit “satu bangunan boleh gagal.”

Istilah	Maksud
IaaS / PaaS / SaaS	Menyewa mesin mentah / platform terurus / perisian siap. Penggelongsor daripada kawalan terbanyak kepada penyelenggaraan tersedikit.
Workload	Sebarang aplikasi atau sistem, dianggap sebagai satu unit reka bentuk — “workload pembayaran.”
Keperluan berfungsi	Apa yang sistem mesti lakukan (“pelanggan boleh memesan”).
Keperluan bukan berfungsi (NFR)	Sebaik mana ia mesti melakukannya — kelajuan, skala, uptime, keselamatan, pematuhan, kos. NFR memacu seni bina.
Trade-off (tolak ansur)	Apa yang anda lepaskan untuk mendapat apa yang anda pilih. Setiap keputusan reka bentuk memilikinya; tugas arkitek ialah menamakannya dengan lantang.
Kekangan (constraint)	Sempadan yang tidak boleh dirunding: bajet, tarikh akhir, undang-undang, sistem sedia ada, kemahiran pasukan. Kekangan bukan halangan kepada reka bentuk — ia <i>ialah</i> ringkasan tugas reka bentuk itu.
Pemegang taruh (stakeholder)	Sesiapa yang berkepentingan dalam sistem: pengguna, jurutera, kewangan, keselamatan, eksekutif, pengawal selia. Pemegang taruh berbeza, bahasa berbeza — anda bertutur kesemuanya.
Greenfield / brownfield	Sistem serba baharu tanpa sejarah (greenfield) lawan yang terjalin dengan sistem sedia ada (brownfield — kebanyakan kerja sebenar).
Managed service	Komponen yang dikendalikan penyedia untuk anda (sandaran, tampalan, failover termasuk). Pilihan lalai arkitek, kecuali ada sebab bertulis sebaliknya.

Video untuk modul ini (pautan disahkan):

Video	Saluran	Tempoh	Pautan
Top 50+ AWS Services Explained in 10 Minutes	Fireship	~10 min	https://www.youtube.com/watch?v=JlbIYCM48to

Tonton lawatan Fireship itu dua kali: sekali sekarang untuk petanya, sekali di penghujung Peringkat 1 — anda akan terkejut betapa banyak perkhidmatan yang kini boleh anda *tempatkan* dalam sebuah reka bentuk.

Latihan: (1) Pilih tiga aplikasi yang anda guna setiap hari (aplikasi bank, aplikasi penghantaran makanan, aplikasi video) dan, untuk setiap satu, tulis tiga NFR teratasnya dan tandakannya pada segi tiga tolak ansur. (2) Temu bual seorang rakan tentang satu idea perniagaan selama sepuluh minit dan ekstrak lima keperluan berfungsi dan lima bukan berfungsi — perhatikan bagaimana NFR hanya keluar apabila anda bertanya dua soalan ajaib itu. (3) Mulakan Jurnal Reka Bentuk anda dengan entri #1: segi tiga itu, dan satu perenggan tentang mengapa “penjuru mana yang kita korbankan?” ialah soalan perniagaan, bukan soalan teknikal.

Pencapaian: diberikan mana-mana penerangan sistem satu ayat, anda boleh menghasilkan NFR-nya yang berkemungkinan dan kedudukannya pada segi tiga itu dalam lima minit, dengan kuat, tanpa nota.

Modul 2 (Minggu 3–4): Compute, storage, database, rangkaian — empat keputusan itu

Setiap reka bentuk yang akan anda lukis ialah empat pilihan ini, dibuat dengan sengaja. Inilah setiap satu diajar sebagai keputusan.

Keputusan 1 — Compute: VM, container, atau serverless? Sebuah **virtual machine (VM)** (mesin maya) ialah kepingan server yang disewa yang berkelakuan seperti komputer penuh — kawalan maksimum, penyelenggaraan maksimum (anda menampalnya, anda menskalakannya, anda membayar semasa ia melahu). Sebuah **container** (bekas perisian) membungkus satu aplikasi berserta segala keperluannya supaya ia berjalan secara serupa di mana-mana; sebuah pengatur (**Kubernetes**) menjalankan dan memulihkan armadanya — kepadatan dan kemudahalihan yang hebat, tetapi anda telah menerima pakai sebuah platform kompleks yang memerlukan orang mahir. **Serverless** (AWS Lambda, Azure Functions) menjalankan kod anda hanya apabila dicetuskan dan membilkan setiap panggilan — penyelenggaraan hampir sifar dan sempurna untuk trafik memuncak, tetapi dengan had (had masa pelaksanaan, cold start, dan perkahwinan lebih mendalam dengan satu penyedia). Jadual keputusan yang patut anda mampu hasilkan semula daripada ingatan:

Pilih	Bila	Awasi
VM	Perisian legasi, pelesenan khas, kawalan OS penuh diperlukan, beban stabil boleh diramal	Anda menanggung tampalan, penskalaan, dan pukul 3 pagi; masa melahu dibilkan penuh
Container + Kubernetes	Banyak servis, pasukan sudah punya kemahiran, kemudahalihan penting	Kerumitan platform — K8s ialah kerja sepenuh masa; berlebihan untuk pasukan kecil
Serverless	Trafik memuncak atau tidak dapat diramal, perekat	Had runtime, cold start, ramalan kos lebih sukar pada

Pilih	Bila	Awasi
	dipacu-peristiwa, pasukan kecil, masa-ke-pasaran pantas	skala stabil yang besar, lock-in

Wawasan senior: **ini pilihan per-workload, bukan agama syarikat**. Estet sebenar menjalankan ketiga-tiganya bersebelahan, dengan betul.

Keputusan 2 — Storage: objek, blok, atau fail — dan sepanas mana? Object storage (Amazon S3) ialah baldi tanpa dasar untuk fail — murah, tahan lasak secara luar biasa (“eleven nines”), jawapan lalai kepada “ke mana fail pergi?” **Block storage** (EBS) ialah cakera maya yang dipasang pada sebuah VM. **File storage** (EFS) ialah pemacu kongsi yang dipasang banyak mesin serentak. Dimensi tambahan arkitek ialah **suhu**: data panas (dicapai sentiasa, dihargakan untuk kelajuan) lawan tier sejuk/arkib (Glacier — sangat murah, tetapi mengambil minit hingga jam untuk diambil semula). Mereka bentuk lifecycle rule yang menghanyutkan data lama ke tier sejuk ialah kemenangan kos termurah dalam cloud; terlupa melakukannya ialah yang paling lazim.

Keputusan 3 — Database: SQL atau NoSQL (dan perisa terurus yang mana)? Database **relational/SQL** (PostgreSQL, MySQL; diurus sebagai RDS/Aurora) menyimpan data dalam jadual ketat dengan konsistensi terjamin — pilihan lalai untuk apa sahaja yang ketepatannya suci: wang, pesanan, inventori, pengguna. Database **NoSQL** (DynamoDB, MongoDB) menukarkan struktur ketat dengan fleksibiliti dan skala mendatar hampir tanpa had — pilihan lalai untuk sesi, katalog, suapan, telemetri. Heuristik keputusannya: *mulakan dengan SQL kecuali anda boleh menamakan sebab khusus ia tidak akan berjaya* (skala melampau, skema fleksibel, bacaan global dalam satu-digit milisaat). Dan dalam cloud, “database” hampir selalu patut bermaksud “database **terurus**” — penyedia mengendalikan sandaran, tampalan, dan failover; pasukan yang menjalankan database sendiri di atas VM patut mempunyai sebab bertulis. Tambahkan pakar-pakar ke perbendaharaan kata anda: **cache** (Redis — data panas dalam memori, bacaan mikrosaas), **warehouse** (analitik berskala — Peringkat 3), **queue** (bukan database, tetapi sering bahagian yang hilang — Peringkat 3).

Keputusan 4 — Rangkaian: bentuk dunia peribadi itu. Sebuah **VPC (Virtual Private Cloud)** ialah bahagian rangkaian penyedia yang dipagar khas untuk anda. Di dalamnya, **subnet public** menempatkan perkara yang boleh dicapai internet (load balancer), dan **subnet private** menempatkan segala yang lain — server aplikasi dan, sentiasa, database. “*The database sits in a private subnet*” (database itu berada dalam subnet private) ialah ayat paling kerap diulang dalam semakan seni bina; alasannya — tiada apa boleh menyerang apa yang tiada laluan dari internet — ialah separuh daripada keselamatan rangkaian. Di sekeliling VPC: sebuah **load balancer** menyebarkan trafik merentasi server dan mengelak yang sakit; **DNS** (Route 53) mengubah nama menjadi alamat; sebuah **CDN** (CloudFront) menyimpan cache kandungan di ratusan bandar supaya ia pantas di mana-mana; sebuah **API gateway** ialah kaunter hadapan terurus untuk API anda (pengesahan, had kadar, pengelogan). Menghubungkan dunia lama: **VPN** (terowong tersulit melalui internet) atau **Direct Connect** (talian fizikal peribadi) — tali pusat setiap reka bentuk hibrid di Peringkat 4.

Perbendaharaan kata:

Istilah	Maksud
Instance / instance type	Satu VM yang disewa / saiznya (CPU + RAM), yang menetapkan harga sejamnya.
Container / Docker / Kubernetes (K8s)	Bungkusan mudah alih untuk satu aplikasi / alat yang membina dan menjalankannya / pengatur yang menjalankan dan memulihkan armadanya.
Serverless / Lambda	Kod yang berjalan hanya apabila dicetuskan, dibilkan setiap larian, tiada server untuk diurus. Lambda ialah versi AWS.
Cold start	Kelewatan tambahan apabila sebuah fungsi serverless berjalan selepas melahu — tolak ansur klasik serverless.
S3 / bucket / durability	Object storage AWS / satu bekas fail bernama / kebarangkalian data terselamat — 99.999999999% milik S3 bermakna kehilangan pada dasarnya tidak pernah berlaku.
Storage tier / lifecycle policy	Kelas harga-kelajuan (panas — sejuk — arkib) / peraturan automatik yang memindahkan data menua ke tier lebih murah.
RDS / Aurora / DynamoDB	Perkhidmatan SQL terurus AWS / SQL berprestasi tinggi cloud-native miliknya / NoSQL terurus perdananya.
Konsistensi	Jaminan bahawa semua yang membaca data melihat kebenaran yang sama pada masa yang sama — kuasa hebat SQL, dan apa yang dilonggarkan NoSQL demi skala.
Cache / Redis	Salinan berkelajuan-memori data panas yang diletakkan di hadapan sebuah database / alat piawai untuknya.
VPC / subnet (public, private)	Bahagian rangkaian peribadi anda / subbahagiannya — public menghadap internet, private tidak. Database tinggal di private. Sentiasa.
Load balancer / health check	Pengarah trafik merentasi server / ujian denyut nadi yang digunakannya untuk berhenti menghantar trafik kepada yang mati.
CDN / edge	Cache peringkat bandar untuk kandungan anda di seluruh dunia / “the edge” = dekat dengan pengguna.

Istilah	Maksud
API / API gateway	Pintu terkawal yang ditawarkan satu perisian kepada perisian lain / kaunter hadapan terurus untuk pintu-pintu itu.
VPN / Direct Connect	Terowong tersulit melalui internet / talian fizikal peribadi ke cloud — dua cara on-prem bertemu cloud.

Video untuk modul ini (pautan disahkan):

Video	Saluran	Tempoh	Pautan
Kubernetes explained in 15 mins	TechWorld with Nana	~16 min	https://www.youtube.com/watch?v=VnvRFRk_51k
Serverless Computing in 100 Seconds	Fireship	~2 min	https://www.youtube.com/watch?v=W_VV2Fx32_Y
AWS Networking Basics (VPC & Subnets)	KodeKloud	~30 min	https://www.youtube.com/watch?v=QM63dyA_4Pc

Latihan: (1) Untuk setiap satu daripada lima workload ini, pilih compute, database, dan storage, dan tulis satu ayat justifikasi setiap satu: laman pemesanan makan tengah hari sekolah; lejar transaksi sebuah bank; aplikasi perkongsian foto; penjana laporan malam yang berjalan 20 minit; aplikasi sembang untuk 5 juta pengguna. (2) Ambil salah satu pilihan anda dan hujahkan pilihan yang *bertentangan* sebijak mungkin — arkitek yang tidak boleh memperkukuh hujah pihak lawan (steel-man) belum memahami tolak ansurnya. (3) Jurnal: jadual keputusan compute anda, daripada ingatan.

Pencapaian: diberikan mana-mana workload dalam satu ayat, anda boleh menamakan empat keputusannya dengan justifikasi dalam masa kurang tiga minit — dan untuk sekurang-kurangnya satu keputusan, menamakan apa yang akan mengubah fikiran anda.

Modul 3 (Minggu 5–6): Membaca dan melukis gambar rajah seni bina

Arkitek yang tidak boleh melukis ialah perunding yang hanya boleh bercakap. Gambar rajah ialah bahasa kerja anda: modul ini menjadikan anda fasih membacanya dan cekap melukisnya.

Gambar rajah kanonik — pelajari yang ini dahulu. Seni bina tiga peringkat (three-tier) ialah “struktur ayat” gambar rajah cloud; kebanyakan reka bentuk ialah variasinya:

Users → DNS → CDN → Load Balancer → [Web/App servers, ×N, multi-AZ, auto-scaled]
→ [Cache]

→ [Database: primary +
standby, private subnets]

Peringkat 1 (**persembahan**): apa yang disentuh pengguna — kandungan statik daripada CDN, permintaan melalui load balancer. Peringkat 2 (**aplikasi**): armada server boleh saling ganti yang menjalankan logik anda, dalam subnet private, diskalakan secara automatik. Peringkat 3 (**data**): database, paling dalam dan paling terlindung, dengan sebuah standby dalam AZ kedua. Trafik mengalir satu hala: pengguna tidak sekali-kali menyentuh peringkat aplikasi secara terus, peringkat aplikasi sahaja yang bercakap dengan peringkat data. Berlatih melukis ini sehingga tangan anda melakukannya tanpa otak anda — ia pembuka temu duga papan putih di mana-mana sahaja di muka bumi.

Konvensyen notasi yang membuatkan anda kelihatan profesional (kerana ia membuatkan anda *berfikir* secara profesional): kotak ialah komponen — labelkan setiap satu dengan apa ia *dan* perkhidmatan mana (“App servers — EC2, auto-scaling group”). Anak panah menunjukkan arah aliran sesebuah *permintaan*, dilabelkan dengan protokol apabila ia penting. Bingkai putus-putus menunjukkan sempadan — VPC, setiap subnet, setiap AZ (lukis sempadan AZ dan multi-AZ menjadi kelihatan dan bukan sekadar dakwaan). Seorang pengguna/pelakon berdiri di luar sistem. Nombor pada anak panah (1, 2, 3...) membolehkan anda menceritakan perjalanan sesebuah permintaan. Dan setiap gambar rajah membawa tajuk, tarikh, dan legenda. Peraturan lebih dalam: **satu gambar rajah, satu khalayak, satu soalan**. Gambar rajah yang menunjukkan segalanya tidak menunjukkan apa-apa; anda akan mempelajari set aras zum piawai (context – container – deployment) di Peringkat 5.

Membaca gambar rajah orang lain — x-ray si arkitek. Apabila dihulurkan sebuah gambar rajah, jalankan imbasan ini dengan kuat: Di mana internet menyentuh sistem ini (setiap titik sentuhan ialah permukaan serangan)? Di mana datanya, dan adakah ia dalam subnet private? Apa yang berganda (berdaya tahan) dan apa yang menjadi **single point of failure** — kotak tanpa kembar? Di mana akan sakit pada trafik 10×? Berapa kos setiap kotak sebulan? Lima soalan, tiga puluh saat, dan anda telah membaca gambar rajah itu sebagaimana seorang doktor membaca x-ray. Layari AWS Architecture Center (<https://aws.amazon.com/architecture/>) dan jalankan imbasan itu pada tiga seni bina rujukan yang diterbitkan.

Perbendaharaan kata:

Istilah	Maksud
Seni bina tiga peringkat	Pemisahan klasik: persembahan (pintu masuk pengguna) – aplikasi (logik) – data (database). Bentuk lalai sistem web.
Tier / lapisan	Kepingan mendatar sistem dengan satu tanggungjawab, hanya bercakap dengan jirannya.
Single point of failure (SPOF)	Mana-mana komponen yang kematiannya bersendirian menumbangkan sistem. Perkara pertama untuk diburu dalam mana-mana gambar rajah.
Multi-AZ	Menjalankan pendua merentasi sekurang-

Istilah	Maksud
	kurangnya dua Availability Zone supaya satu pusat data boleh gagal tanpa disedari.
Auto-scaling (group)	Mesin ditambah dan dibuang secara automatik mengikut permintaan — kapasiti yang bernafas.
Stateless / stateful	Server yang tidak memegang data unik (mana-mana kembar boleh menggantikannya, jadi ia berskala bebas) lawan yang memegang data yang tidak boleh hilang. Matlamat reka bentuk: peringkat aplikasi stateless, keadaan (state) ditolak ke bawah kepada database dan cache.
Seni bina rujukan	Contoh reka bentuk yang diterbitkan dan direstui penyedia untuk masalah lazim — arkitek memasang daripadanya sebelum mencipta.
Context diagram	Aras zum tertinggi: sistem anda sebagai satu kotak, berserta pengguna dan sistem luaran yang disentuhnya.
Permukaan serangan (attack surface)	Setiap titik di mana dunia luar boleh menyentuh sistem. Lebih kecil lebih selamat.
Trafik utara-selatan / timur-barat	Trafik masuk/keluar sistem lawan trafik antara komponen di dalamnya.

Latihan: (1) Lukis gambar rajah tiga peringkat itu daripada ingatan, lima hari berturut-turut, sehingga ia mengambil masa kurang empat minit dengan semua sempadan (VPC, subnet, dua AZ) dilukis. (2) Ambil lima workload Modul 2 dan lukis setiap satu — dua puluh minit bagi setiap gambar rajah, peraturan notasi dikuatkuasakan. (3) Cari mana-mana gambar rajah seni bina sebenar dalam talian (AWS Architecture Center ada ratusan), dan tulis imbasan x-ray lima soalnya dalam jurnal anda. (4) Latih tubi penceritaan: gambar rajah di hadapan anda, ceritakan satu klik seorang pengguna dari pelayar ke database dan kembali, dengan kuat, sambil menomborkan anak panah.

Pencapaian — penghujung Peringkat 1: di papan putih (atau kertas, difotograf untuk jurnal anda), anda boleh melukis reka bentuk tiga peringkat yang betul dan bernotasi wajar untuk sebuah workload satu-ayat yang baharu dalam masa kurang lima belas minit, menceritakan sebuah permintaan melaluinya, dan menjawab “apa gagal jika kotak ini mati?” untuk setiap kotak. Lukisan-berserta-soal-siasat inilah tepat separuh pertama temu duga arkitek sebenar — dari sini, segalanya ialah kedalaman.

PERINGKAT 2 — DOMAIN REKA BENTUK (Minggu 7–14)

Peta untuk peringkat ini ialah AWS Well-Architected Framework — senarai semak bersama industri tentang maksud “direka bentuk dengan betul,” disusun ke dalam enam tunjang: **kecemerlangan operasi, keselamatan, kebolehpercayaan, kecekapan prestasi, pengoptimuman kos, kelestarian**. (Azure mempunyai kerangka yang hampir serupa; pelajari satu secara mendalam dan anda telah mempelajari kedua-duanya.) Deskripsi kerja meminta arkitek yang boleh “run Well-Architected reviews” dengan namanya, jadi kita mengambil tunjang-tunjang itu satu demi satu, dan untuk setiap satu anda mempelajari tiga perkara: **soalan-soalan utama** tunjang itu, **corak piawainya** (jawapan membosankan yang terbukti — arkitek memasang sebelum mencipta), dan satu **contoh bekerja** pada sebuah senario berterusan.

Senario berterusan untuk seluruh Peringkat 2: *ThaiTicket*, sebuah platform tiket acara fiksiyen di Bangkok. Beban biasa: 2,000 pengunjung/jam. Tetapi apabila tiket seorang artis terkenal dilepaskan pada 10:00 pagi, ia menerima 400,000 pengunjung dalam sepuluh minit, pembayaran tidak boleh menjual kerusi dua kali, dan data peribadi pelanggan Thailand tertakluk kepada PDPA. Pantas, murah, berdaya tahan — ThaiTicket memerlukan ketiga-tiganya dan tidak boleh memilikinya, dan itulah yang menjadikannya pesakit latihan yang sempurna.

Tonton sebelum Modul 4 (pautan disahkan):

Video	Saluran	Tempoh	Pautan
The Five Pillars of the AWS Well-Architected Framework	Amazon Web Services (official; tunjang keenam, Sustainability, ditambah kemudian)	~4 min	https://www.youtube.com/watch?v=KvEDbPmha6o
What is the AWS Well-Architected Framework?	Tech With Lucy	~10 min	https://www.youtube.com/watch?v=MpDJ6TCWKjk

Dan tanda bukukan kerangka itu sendiri — <https://aws.amazon.com/architecture/well-architected/> dan dokumen penuh di <https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html> — anda akan hidup di dalamnya selama lapan minggu.

Modul 4 (Minggu 7–8): Kebolehpercayaan — mereka bentuk untuk hari benda rosak

Pegangan tunjang ini: segalanya gagal, sepanjang masa. Cakera mati, AZ dibanjiri, deploy tersilap, dan sebuah sijil di suatu tempat sentiasa hampir luput. Kebolehpercayaan bukan ketiadaan kegagalan — ia *ketidakrelevanan* kegagalan: mereka bentuk supaya apabila (bukan jika) sesebuah komponen mati, pengguna tidak pernah perasan.

Soalan-soalan utama (tanyakan ini kepada setiap reka bentuk, selama-lamanya): Apa berlaku apabila setiap komponen gagal — adakah sentiasa ada “kemudian apa”? Bagaimana sistem menangani beban 10×? Bagaimana kita *tahu* sesuatu gagal sebelum pelanggan memberitahu kita? Apakah **RTO** dan **RPO** kita — dan siapa dalam perniagaan yang meluluskannya? Bilakah kali terakhir kita *menguji* sebuah pemulihan?

RTO dan RPO — dua nombor yang menjadi perbualan bencana itu. Recovery Time

Objective: berapa lama kita boleh terhenti? **Recovery Point Objective:** berapa banyak data kita boleh hilang (iaitu, berapa lama usia sandaran baik yang terakhir)? RTO 4 jam dan RPO 1 jam bermaksud: kembali dalam empat jam, dengan kehilangan paling banyak satu jam data. Ini adalah keputusan *perniagaan* dengan tanda harga eksponen — RPO 24 jam ialah sandaran malam (murah); RPO ~sifar ialah replikasi berterusan (mahal); RTO beberapa minit bermakna infrastruktur suam melahu sedia menunggu (sangat mahal). Tugas arkitek ialah membuatkan perniagaan memilih nombor-nombor itu *secara sedar* — dan mereka bentuk tepat kepadanya, bukan secara romantik melebihinya.

Corak-corak piawai: **redundancy** (lewahan — dua bagi setiap yang penting — N+1) · **multi-AZ** (pendua merentasi pusat data; load balancer dan failover database terurus menjadikannya automatik) · **auto-scaling** (kapasiti mengikut permintaan) · **health check + penyembuhan diri** (instance mati dikesan dan digantikan oleh jentera, bukan manusia) · **sandaran, yang diuji** (sandaran yang tidak diuji ialah harapan, bukan pelan — jadualkan latihan pemulihan) · **graceful degradation** (kemerosotan beradab — apabila terlebih beban, gugurkan ciri paling tidak penting dahulu: ThaiTicket boleh menggugurkan pratonton peta kerusi dan mengekalkan checkout) · **queue sebagai penyerap hentakan** (Peringkat 3) · **mengelakkan kegagalan melata** (timeout, cubaan semula dengan backoff, circuit breaker — supaya satu kebergantungan yang perlahan tidak menenggelamkan seluruh armada).

Contoh bekerja — reka bentuk kebolehpercayaan ThaiTicket. Dua AZ dalam region Bangkok. Peringkat aplikasi stateless dalam sebuah auto-scaling group di belakang load balancer, dipanaskan awal mengikut jadual sebelum waktu jualan yang diumumkan (auto-scaling bertindak balas dalam minit; pelepasan 10:00 memerlukan kapasiti pada 09:45). Database Aurora, primary di AZ-a, standby segerak di AZ-b, failover automatik ≈ bawah seminit. Sebuah queue antara klik “beli” dan pemprosesan pembayaran supaya kelembapan penyedia pembayaran membariskan pesanan dan bukannya meranapkan laman. Sandaran: berterusan, pemulihan point-in-time, latihan pemulihan bulanan. Nombor yang dipersetujui dengan perniagaan: RTO 15 minit, RPO ~sifar untuk pesanan (ia wang), RPO 24 jam untuk data analitik (ia bukan). Latihan jurnal: apa yang ini merugikan kita dari penjuru “murah” segi tiga itu?

Perbendaharaan kata:

Istilah	Maksud
RTO / RPO	Recovery Time Objective: berapa lama anda boleh terhenti. Recovery Point Objective: berapa banyak data anda boleh hilang. Dua nombor yang mentakrifkan setiap perbualan DR — dan ia keputusan perniagaan.

Istilah	Maksud
High availability (HA)	Mereka bentuk supaya kegagalan rutin (sebuah instance, sebuah AZ) tidak menyebabkan gangguan yang kelihatan kepada pengguna.
Disaster recovery (DR)	Pelan untuk kegagalan <i>besar</i> — seluruh region, sebuah peristiwa ransomware — dengan sasaran RTO/RPO dan runbook yang diuji.
Redundancy / N+1	Kapasiti ganti supaya mana-mana satu komponen boleh mati: N diperlukan, N+1 berjalan.
Failover	Pertukaran automatik kepada standby apabila yang primary mati.
Health check	Denyut nadi automatik yang mengesan komponen mati supaya jentera boleh mengelakkannya.
Graceful degradation	Menggugurkan ciri kurang penting semasa tekanan supaya teras terselamat.
Timeout / retry with backoff	Berputus asa dengan panggilan perlahan selepas satu had / mencuba semula dengan jeda yang membesar — adab yang mencegah kegagalan melata.
Circuit breaker	Komponen yang berhenti memanggil kebergantungan yang gagal buat seketika supaya ia boleh pulih — seperti fius di rumah.
Chaos engineering	Menyuntik kegagalan dengan sengaja (gaya Netflix) untuk membuktikan dakwaan daya tahan sebelum realiti mengujinya untuk anda.

Latihan: (1) Ambil reka bentuk pemesanan makan tengah hari Peringkat 1 anda dan naik tarafkannya untuk terselamat: kematian sebuah instance, kehilangan sebuah AZ, kegagalan database, dan kesibukan makan tengah hari 10× — lukis sebelum dan selepas. (2) Tulis perbualan RTO/RPO untuk tiga perniagaan (sebuah blog, sebuah laman e-dagang, sebuah sistem rekod hospital) sebagai dialog pendek antara arkitek dan pemilik — berlatih menjadikan kos kelihatan dalam dialog itu. (3) Jurnal: senaraikan setiap SPOF dalam reka bentuk ThaiTicket di atas. Ada sekurang-kurangnya satu yang sengaja ditinggalkan. (Petunjuk: berapa banyak region?)

Pencapaian: anda boleh menjalankan soal siasat “apa berlaku apabila ini gagal?” ke atas mana-mana gambarajah selama sepuluh minit berturut-turut tanpa kehabisan, dan anda boleh menerangkan RTO lawan RPO kepada pemilik bukan teknikal menggunakan analogi kedai-dilanda-banjir dalam sembilan puluh saat.

Modul 5 (Minggu 9-10): Keselamatan — mereka bentuk sistem yang sukar disakiti

Bingkainya: shared responsibility model (model tanggungjawab bersama). Penyedia melindungi *cloud itu sendiri* — bangunan, perkakasan, hypervisor. Anda melindungi segala yang anda *letakkan di dalamnya* — data, identiti, konfigurasi, kod. Hampir setiap pencerobohan cloud terkenal ialah salah konfigurasi di pihak pelanggan (bucket public, kunci bocor, role terlebih kebenaran), sebab itulah keselamatan ialah masalah *seni bina* sebelum ia masalah peralatan: deskripsi kerja menyebut “integrate security into the design rather than bolting it on,” dan modul inilah caranya.

Soalan-soalan utama: Siapa dan apa boleh mencapai setiap komponen, dan adakah setiap kebenaran itu *minimum* yang diperlukan? Di mana data disulitkan — semasa disimpan (at rest), semasa bergerak (in transit), dan siapa memegang kuncinya? Di mana sempadan rangkaian, dan apa yang melintasinya? Bagaimana kita akan *mengesan* pencerobohan — dan mampukah kita menceritakan kisahnya kemudian daripada log? Undang-undang apa terpakai kepada data ini, dan di mana data itu berada secara fizikal?

Corak-corak piawai: least privilege (setiap manusia dan program mendapat capaian minimum yang diperlukan tugasnya — peraturan emas yang menjadi kayu ukur setiap policy IAM) · **MFA di mana-mana, root disimpan berkunci** · **penyulitan at rest dan in transit, sentiasa hidup** (ia kotak semak dalam cloud; tiada alasan) · **segmentasi rangkaian** (subnet public/private, security group sebagai firewall per-server; blast radius sesebuah pencerobohan ditakrifkan oleh dinding yang anda lukis lebih awal) · **rahsia dalam vault, tidak sekali-kali dalam kod** · **zero trust** (sahkan setiap permintaan secara eksplisit — identiti, peranti, konteks — jangan percayai apa-apa semata-mata kerana berada “di dalam rangkaian”; deskripsi kerja menamakannya, anda juga mesti) · **defense in depth** (pertahanan berlapis, supaya satu kawalan yang gagal bukan tamat permainan) · **pengelogan audit** (rekod gaya-CloudTrail tentang siapa buat apa — tidak boleh diubah, dipantau) · **keselamatan sebagai guardrail, bukan pintu sekatan** (kodkan peraturan sebagai polisi automatik yang menjadikan laluan selamat itu laluan mudah, dan bukannya mesyuarat semakan yang menjadikan keselamatan musuh penghantaran).

Contoh bekerja — reka bentuk keselamatan ThaiTicket. Data pelanggan (nama, e-mel, rujukan pembayaran) dikelaskan sebagai data peribadi di bawah PDPA — disimpan tersulit dalam Aurora di region Bangkok (data residency), kunci dalam KMS. Rangkaian: hanya load balancer yang public; peringkat aplikasi private; subnet database menerima sambungan *hanya* daripada security group peringkat aplikasi. Manusia: SSO + MFA; jurutera mendapat capaian production baca-sahaja secara lalai dan capaian tinggi berhad-masa atas permintaan (least privilege dengan jejak audit). Pengendalian kad pembayaran diserahkan kepada penyedia pembayaran bertauliah supaya nombor kad mentah tidak pernah menyentuh sistem kita — sebuah keputusan *reka bentuk* yang menghapuskan seluruh beban pematuhan, iaitu seni bina keselamatan pada tahap terbaiknya. CloudTrail hidup, amaran atas capaian luar biasa. Soalan pencerobohan dijawab lebih awal: PDPA mempunyai kewajipan notifikasi 72 jam — log dan runbook mesti membolehkan kita menceritakan kisahnya dalam masa kurang daripada itu.

Perbendaharaan kata:

Istilah	Maksud
Shared responsibility model	Penyedia melindungi cloud; anda melindungi apa yang di dalamnya. Kebanyakan pencerobohan berada di pihak pelanggan pada garisan itu.
IAM / role / policy	Identity and Access Management — siapa boleh buat apa. Policy ialah senarai kebenaran; role ialah gabungan boleh-pakai kesemuanya.
Least privilege	Peraturan emas: capaian minimum yang diperlukan, tidak lebih, disemak secara berkala.
MFA	Bukti identiti kedua melangkaui kata laluan. Tidak boleh dirunding untuk manusia.
Penyulitan at rest / in transit / KMS	Data dikarau pada cakera / pada wayar / perkhidmatan terurus yang memegang kunci.
Security group	Senarai peraturan firewall per-server — “trafik web masuk daripada load balancer sahaja.”
Segmentasi rangkaian / blast radius	Membahagikan rangkaian kepada zon berding / sejauh mana penyerang boleh sampai selepas satu pencerobohan. Dinding yang dilukis lebih awal mentakrifkannya.
Zero trust	Sahkan setiap permintaan secara eksplisit; jangan percayai apa-apa kerana berada di “dalam.” Postur lalai moden.
Defense in depth	Berbilang kawalan bertindih supaya satu kegagalan tidak membawa maut.
Pengurusan rahsia (secrets)	Kata laluan, kunci, dan token hidup dalam perkhidmatan vault — tidak sekali-kali dalam kod, tidak sekali-kali dalam hamparan.
Log audit / CloudTrail	Rekod tidak boleh diubah tentang siapa buat apa, bila — cara anda mengesan masalah dan membina semula kisahnya kemudian.
Pengelasan data	Melabelkan data mengikut sensitiviti (public / dalaman / peribadi / terkawal selia) supaya kawalan sepadan dengan label, bukan tekaan.
Data residency	Menyimpan data secara fizikal di dalam sempadan sesebuah negara — keperluan undang-undang dalam banyak industri,

Istilah	Maksud
PDPA / GDPR	input seni bina sentiasa. Undang-undang data peribadi Thailand dan Eropah: keizinan, notifikasi pelanggaran (72 jam), peraturan pemindahan merentas sempadan. Peringkat 4 mendalaminya.

Video untuk modul ini (pautan disahkan): The AWS Shared Responsibility Model — Digital Cloud Training — ~4 min — <https://www.youtube.com/watch?v=ESPBBEK-cvo>

Latihan: (1) Lukis gambar rajah ThaiTicket dan tindakan keselamatannya: tandakan setiap sempadan kepercayaan, setiap titik penyulitan, setiap tempat kelayakan hidup. Tabiat tindakan ini — gambar rajah sama, kanta keselamatan — ialah tepat apa yang diminta sebuah semakan daripada anda. (2) Baca satu post-mortem awam tentang pencerobohan salah konfigurasi cloud (Capital One 2019 ialah klasik pengajaran) dan tulis, dalam jurnal anda, *corak yang mana di atas* yang akan mencegahnya. (3) Main peranan dengan sebuah AI: ia berlakon sebagai pengasas syarikat pemula yang berkata “kita tambah keselamatan kemudian” — pujuk mereka keluar daripadanya dengan aritmetik kos-pencerobohan, secara baik hati.

Pencapaian: anda boleh mengambil mana-mana gambar rajah Peringkat 1 anda dan menghasilkan tindakan keselamatannya dalam dua puluh minit, dan menerangkan least privilege, zero trust, dan shared responsibility model kepada pemegang taruh bukan teknikal tanpa jargon.

Modul 6 (Minggu 11-12): Prestasi dan kos — kelajuan dan wang, tunjang berkembar

Dua tunjang ini diajar bersama kerana ia adalah tuas yang sama ditolak ke arah bertentangan, dan arkitek ialah orang yang tangannya berada pada tuas itu.

Kecekapan prestasi — soalan utama: Di mana pengguna akan merasai kelembapan dahulu? Apa yang dikira berulang kali yang boleh dikira sekali dan di-cache? Adakah setiap komponen *alat yang betul* (SQL melakukan kerja enjin carian ialah pepijat prestasi seni bina itu, bukan kodnya)? Bagaimana prestasi berubah pada 10× — dan di mana bottleneck pertama?

Corak prestasi: lapisan caching — corak prestasi bertuas tertinggi: cache pelayar — **CDN** di edge (kandungan statik dihidangkan dari bandar berdekatan pengguna; boleh menyerap sebahagian besar trafik bacaan) → cache aplikasi (**Redis** di hadapan database untuk bacaan panas: peta kerusi, sesi, halaman produk) → cache pertanyaan database. Setiap lapisan menjawab permintaan sebelum ia sampai ke teras yang mahal; soalan reka bentuknya sentiasa *apa yang boleh di-cache, untuk berapa lama, dan bagaimana ia dibatalkan (invalidate) apabila kebenaran berubah*. Kemudian: **read replica** (salinan database yang melayani bacaan supaya yang primary terus menulis) · **pemprosesan asinkron** (jangan buat pengguna menunggu kerja yang boleh berlaku kemudian — resit e-mel, thumbnail, laporan) · **alat yang betul untuk setiap kerja** (carian → enjin carian; analitik → warehouse; carian panas → kedai kunci-nilai) · dan **ukur dahulu** (kerja prestasi tanpa pengukuran ialah kepercayaan karut).

Pengoptimuman kos — soalan utama: Berapa kos reka bentuk ini sebulan pada beban hari ini — dan per *unit* (setiap pesanan, setiap pelanggan)? Apa yang berjalan pada pukul 3 pagi yang tidak perlu? Penggunaan stabil apa yang boleh dikomitkan pada harga diskaun? Apa yang akan dikatakan pihak kewangan tentang arah alirannya?

Corak kos: rightsizing (kebanyakan armada diperuntukkan berlebihan secara senyap; mengecil kepada keperluan terukur ialah wang percuma) · **strategi reservation** — menu harga: **on-demand** (harga penuh, kebebasan penuh) untuk beban memuncak/tidak diketahui, **reserved instances / savings plans** (komitmen 1–3 tahun, 30–70% diskaun) untuk garis dasar stabil, **spot** (sehingga 90% diskaun, boleh dituntut semula dengan notis singkat) untuk kerja batch boleh terganggu; langkah arkitek ialah melapiskannya: reserve lantainya, auto-scale bahagian tengah secara on-demand, spot untuk batch · **lifecycle storan** (tiering Modul 2, diautomasikan) · **matikan benda** (persekitaran bukan production yang tidur malam dan hujung minggu boleh memotong kosnya dua pertiga) · **perhatikan egress** (data *keluar* dari cloud dibilkan; reka bentuk merentas region yang banyak berbual dan muat turun awam besar mengejutkan semua orang sekali) · **tagging dan showback** (setiap sumber dilabelkan dengan pemilik/projek supaya setiap baht boleh dipertanggungjawabkan) · dan metrik mahkota, **unit economics** (ekonomi seunit): bukan “bilnya \$800k/bulan” tetapi “kos setiap tiket terjual ialah \$1.90 dan menurun.” Bil yang membesar tidak mengapa; *kos seunit* yang membesar ialah bau busuk seni bina. Disiplin ini ada nama — **FinOps** — dan arkitek duduk di tengah-tengahnya.

Contoh bekerja — ThaiTicket, kedua-dua kanta. Prestasi: CloudFront menghidangkan halaman artis dan imej peta kerusi (400,000 penonton itu kebanyakannya tidak pernah menyentuh server); Redis meng-cache ketersediaan kerusi dengan TTL 2 saat — cukup basi untuk murah, cukup segar sehingga checkout (yang menyemak semula terhadap Aurora, sumber kebenaran) mencegah jualan berganda; resit dan tiket dijana secara asinkron selepas pembayaran. Kos: garis dasar stabil 2,000-pengunjung/jam berjalan atas savings plan (~40% diskaun); lonjakan jualan berjalan on-demand untuk jam ia wujud; kerja analitik berjalan atas spot pada waktu malam; staging tidur di luar waktu pejabat; setiap sumber ditag `project:thaiticket`. Cadangan kepada CFO berbunyi: “\$62,000/bulan garis dasar, ~\$9,000 setiap acara jualan besar, kos setiap tiket ~\$1.90 menurun dengan volum” — dan *ayat itulah* bunyi seorang arkitek yang fasih kos.

Perbendaharaan kata:

Istilah	Maksud
Latency / throughput	Berapa lama satu permintaan mengambil masa / berapa banyak permintaan sesaat yang ditanggung sistem. Berkaitan, bukan sama.
Bottleneck	Titik tersempit yang menetapkan rentak keseluruhan sistem. Mengoptimumkan di tempat lain hanyalah hiasan.
Cache / TTL / invalidation	Salinan pantas data yang mahal untuk diambil / berapa lama ia boleh dipercayai / masalah sukar menyegarkannya apabila kebenaran berubah.

Istilah	Maksud
CDN	Cache paling luar — kandungan anda di ratusan bandar. Jawapan pertama kepada “jadikan ia pantas secara global.”
Read replica	Salinan database yang melayani bacaan, supaya yang primary menyimpan kekuatannya untuk penulisan.
Pemprosesan asinkron	Melakukan kerja tidak mendesak selepas membalas kepada pengguna, biasanya melalui queue.
On-demand / reserved / savings plan / spot	Harga penuh fleksibel / komitmen 1–3 tahun pada 30–70% diskaun / idea sama, lebih fleksibel / sehingga 90% diskaun tetapi boleh dituntut semula. Arkitek melapiskan keempat-empatnya.
Rightsizing	Mengecilkan sumber terlebih peruntukan kepada keperluan terukur. Wang percuma dalam hampir setiap akaun.
Egress	Data meninggalkan cloud — dibilkan per GB. Kejutan bil yang terkenal; reka bentuk aliran data dengan mengambil kiranya.
Tagging / showback	Melabelkan setiap sumber dengan pemilik dan projek / menunjukkan kepada setiap pasukan bilnya sendiri. Akauntabiliti mengubah tingkah laku.
Unit economics	Kos setiap unit perniagaan (setiap pesanan, setiap pengguna). Metrik yang menjadikan sebuah bil bermakna — dan ayat kegemaran arkitek di hadapan seorang CFO.
TCO	Total cost of ownership — harga label berserta manusia, lesen, migrasi, dan kos keluar sepanjang hayat sesuatu pilihan.
FinOps	Disiplin menjadikan perbelanjaan cloud kelihatan, diperuntukkan, dan dioptimumkan secara berterusan.

Video untuk modul ini (pautan disahkan): What is FinOps? — FinOps Foundation (official) — ~2 min — https://www.youtube.com/watch?v=Y-c_xw9bHFw

Latihan: (1) Hargakan garis dasar ThaiTicket dalam kalkulator harga AWS (cari “AWS pricing calculator” — pengalaman memperincikan sebuah reka bentuk sebenar sekali menghilangkan misteri setiap perbualan kos masa depan); bandingkan jumlah anda dengan \$62,000 di atas dan terangkan sebarang jurang. (2) Tambahkan tindanan caching pada dua gambar rajah Peringkat 1 anda: tandakan setiap cache, TTL-nya, dan kisah invalidation-nya. (3) Sebuah AI

berlakon sebagai jurutera yang mahukan segalanya on-demand “supaya mudah” — rundingkan strategi reservation, dengan nombor.

Pencapaian: untuk mana-mana reka bentuk yang telah anda lukis, anda boleh menyatakan tiga baris kos terbesarnya dan bottleneck pertamanya yang berkemungkinan — dan mencadangkan satu perubahan yang menambah baik kedua-duanya serentak (hampir selalu ada satu; caching biasanya jawabannya).

Modul 7 (Minggu 13–14): Kecemerlangan operasi, kelestarian — dan menjalankan keenam-enam tunjang bersama

Kecemerlangan operasi ialah tunjang yang bertanya: *bolehkah manusia benar-benar menjalankan benda ini, dengan tenang?* Soalan utama: Bagaimana sesebuah perubahan sampai ke production — melalui pipeline automatik yang diuji, atau melalui seorang manusia heroik? Bagaimana kita tahu sistem itu sihat *sekarang*? Apabila ia rosak pada pukul 3 pagi, apa sebenarnya yang dilakukan jurutera on-call? Corak: **infrastructure as code** (persekitaran ditakrifkan dalam teks yang disemak dan dikawal versi — Terraform/CloudFormation — supaya ia boleh dihasilkan semula dan diaudit; seni bina yang wujud hanya sebagai klik-klik console ialah cerita rakyat, bukan kejuruteraan) · **pipeline CI/CD** dengan strategi deployment selamat (**blue-green**: dirikan versi baharu di sebelah yang lama dan tukar; **canary**: beri versi baharu 5% trafik dan perhatikan) · **observability** (kebolehcerapan — log, metrik, trace, dan amaran yang diikat kepada simptom *yang kelihatan kepada pengguna*, bukan perkara remeh mesin) · **runbook** (panduan bertulis apa-nak-buat untuk setiap kegagalan yang diketahui) · **post-mortem tanpa salah-menyalah** (selepas insiden: apa yang gagal, mengapa, apa yang mencegah ulangan — tiada penjahat, atau orang berhenti berkata benar). Bahagian arkitek: *mereka bentuk* untuk kebolehkendalian — sistem yang dua kali lebih bijak dan separuh boleh dicerap ialah sistem yang lebih buruk.

Kelestarian, tunjang keenam, meminta anda kurang membazirkan dunia fizikal: rightsize (CPU melahu membakar elektrik sebenar), skala mengikut permintaan dan bukannya memperuntukkan untuk puncak, gunakan perkhidmatan terurus dan serverless (infrastruktur kongsi ialah infrastruktur yang lebih penuh), tier storan, dan padamkan yang sudah mati. Kebetulan pula, pilihan lestari dan pilihan murah biasanya pilihan yang sama — sebutkan kedua-duanya dalam semakan dan anda akan memenangi bilik itu dua kali.

Menjalankan tunjang-tunjang bersama — semakan Well-Architected pertama anda.

Dalam semakan sebenar tunjang-tunjang itu *berkonflik*: kebolehpercayaan multi-region berlawanan dengan kos; semakan keselamatan ketat berlawanan dengan kelajuan operasi; caching prestasi berlawanan dengan ketepatan kesegaran data. Kerangka itu tidak menyelesaikan konflik untuk anda — ia memaksanya ke tempat terbuka, di mana perniagaan boleh memilih. Itulah keseluruhan kebijaksanaannya, dan milik anda untuk digunakan. Kaedah semakan itu sendiri (set soalan, mengutamakan penemuan, laporan) diajar sepenuhnya di Peringkat 5, Modul 12; dua minggu ini anda melatih kemahiran mentahnya: menyoal siasat satu reka bentuk dari enam arah dalam satu sesi.

Perbendaharaan kata:

Istilah	Maksud
Infrastructure as Code (IaC) / Terraform	Infrastruktur diisytiharkan dalam fail teks terkawal versi yang diubah alat menjadi realiti / alat paling popular sebegitu.
Pipeline CI/CD	Tali sawat automatik yang membina, menguji, dan men-deploy setiap perubahan.
Deployment blue-green / canary	Dua corak keluaran selamat: tukar trafik antara salinan lama dan baharu / titiskan trafik ke versi baharu dan perhatikan.
Observability (log, metrik, trace)	Keupayaan sistem menerangkan dirinya: rekod peristiwa, nombor merentas masa, dan peta perjalanan per-permintaan.
Alert / on-call / runbook	Panggilan automatik apabila ambang dilanggar / giliran manusia yang menjawabnya / skrip bertulis yang mereka ikuti.
Post-mortem tanpa salah-menyalah	Semakan jujur tanpa penjahat selepas sebuah insiden yang menghasilkan pencegahan, bukan hukuman.
Drift	Realiti menyimpang daripada takrifan IaC kerana seseorang mengklik sesuatu. Musuh kebolehulangan.
Toil	Kerja operasi manual berulang yang sepatutnya telah diserap automasi. Arkitek mereka bentuknya keluar.
Semakan Well-Architected	Soal siasat berstruktur sesebuah workload terhadap keenam-enam tunjang, menghasilkan penemuan yang diutamakan. Peringkat 5 mengajar anda menjalankannya.

Latihan: (1) Latih tubi enam kanta: ambil gambar rajah terbaik anda dan luangkan sepuluh minit bagi setiap tunjang menulis penemuan — enam puluh minit, satu reka bentuk, enam kanta; latih tubi ini *ialah* Peringkat 2 dalam bentuk miniatur, ulangnya setiap minggu mulai sekarang. (2) Baca dua laporan pasca-insiden AWS (diterbitkan di halaman status mereka selepas gangguan besar) dan kenal pasti corak tunjang mana yang gagal. (3) Jurnal: untuk ThaiTicket, tulis tiga konflik-tunjang yang akan anda ketengahkan kepada perniagaan, setiap satu sebagai pilihan satu ayat (“Kita boleh dapat X atau Y pada bajet ini — yang mana?”).

Pencapaian — penghujung Peringkat 2: semakan olok-olok: sebuah AI menjana seni bina yang cacat untuk sebuah syarikat pemula pinjaman dalam talian Thailand; anda mesti menghasilkan penemuan bertulis merentasi keenam-enam tunjang, termasuk sekurang-kurangnya: satu jurang kebolehpercayaan (database single-AZ mereka), satu jurang keselamatan (IAM terlalu luas), satu jurang kos (semuanya on-demand), satu jurang operasi (tiada IaC), dan perbualan RTO/RPO yang mereka tidak pernah adakan — setiap penemuan

diungkapkan sebagai soalan yang boleh didengar seorang rakan sekerja tanpa tersentak. Apabila senarai penemuan anda berbunyi seperti rakan sekerja senior yang membantu dan bukannya seorang juruaudit, Peringkat 2 selesai.

PERINGKAT 3 — KATALOG CORAK (Minggu 15–20)

Arkitek tidak mencipta; mereka **memilih**. Peringkat ini ialah katalog anda tentang seni bina piawai — untuk setiap satu: apa ia, bila menggunakannya, bila *tidak*, dan profil kos/kerumitannya. Lajur “bila tidak” ialah muatan sebenar peringkat ini: mana-mana kursus boleh memberitahu anda apa itu microservices; mengetahui bila ia akan memusnahkan sebuah syarikat ialah apa yang anda dibayar untuknya. Sambil belajar, teruskan melayari seni bina rujukan sebenar di <https://aws.amazon.com/architecture/> — mengesan corak di alam liar ialah latihan yang membuatkan katalog ini melekat.

Modul 8 (Minggu 15–17): Corak aplikasi

Monolith lawan microservices — pertenggaran paling bising industri, diselesaikan dengan tenang. Sebuah **monolith** ialah satu aplikasi boleh-deploy yang mengandungi semua ciri; **microservices** memecahkan sistem kepada banyak servis kecil yang di-deploy secara bebas, setiap satu memiliki datanya sendiri, berkomunikasi melalui API dan peristiwa. Jadual jujurnya:

Corak	Apa ia	Guna bila	JANGAN guna bila	Kos/kerumitan
Monolith	Satu aplikasi, satu deployment, satu database	Pasukan kecil, produk muda, sempadan domain belum jelas — iaitu, kebanyakan sistem baharu	Berbilang pasukan saling menyekat; bahagian-bahagian perlu berskala dengan sangat berbeza	Kerumitan rendah, kos rendah; berskala lebih jauh daripada yang diakui fesyen — “membosankan” ialah satu kelebihan
Microservices	Banyak servis kecil, dibina, di-deploy, dan diskalakan secara bebas	Banyak pasukan memerlukan irama keluaran bebas; penskalaan sangat berbeza bagi setiap bahagian; sempadan domain terbukti	Pasukan kecil (“distributed monolith ialah monolith dengan kegagalan rangkaian ditambah”); domain masih beranjak	Kerumitan tinggi: setiap panggilan fungsi menjadi panggilan rangkaian yang boleh gagal; memerlukan CI/CD matang, observability,

Corak	Apa ia	Guna bila dan stabil	JANGAN guna bila	Kos/kerumitan on-call
-------	--------	-------------------------	---------------------	--------------------------

Keputusan arkitek: **mulakan secara monolitik, bermodul di dalam; ekstrak servis apabila — dan hanya apabila — kesakitan penskalaan-pasukan atau penskalaan-beban benar-benar tiba.** Menyebut ini dalam temu duga, dengan sebab, menandakan anda senior; ideologi ke mana-mana arah menandakan anda junior.

Seni bina dipacu-peristiwa — queue dan topic, penyerap hentakan. Berbanding komponen saling memanggil dan *menunggu* (segerak), komponen menerbitkan **peristiwa (event)** (“OrderPlaced”) ke sebuah **queue** (SQS — satu pengguna mengambil setiap mesej, mengikut rentaknya sendiri) atau sebuah **topic** (SNS — setiap pelanggan mendapat satu salinan; fan-out). Penerbit tidak tahu dan tidak peduli siapa yang mendengar. Guna bila: komponen patut terselamat daripada gangguan satu sama lain (queue memegang mesej semasa pengguna terhenti — daya tahan *dan* penyahgandingan dalam satu pembelian); beban memuncak (queue menyerap lonjakan; pekerja mengalirkannya secara stabil); satu kejadian mencetuskan banyak reaksi (pesanan dibuat — caj, e-mel, stok, analitik — empat pelanggan, sifar gandingan). JANGAN guna bila: pengguna memerlukan jawapan *sekarang* (checkout tidak boleh mengesahkan “akhirnya nanti”); atau pasukan itu kecil dan panggilan segerak mudah sudah memadai — setiap queue menambah semantik penghantaran (at-least-once bermakna pengguna mesti **idempotent** — selamat dijalankan dua kali), pengendalian dead-letter, dan beban pemantauan. Kos/kerumitan: komponennya murah, penyahpepijatan lebih mahal — kisah sesebuah permintaan kini bertaburan merentasi servis dan queue, sebab itulah observability (Modul 7) berhenti menjadi pilihan di sini.

Tiga peringkat: sudah milik anda (Modul 3). Guna untuk: golongan tengah luas aplikasi web — ia corak yang menjadi kayu ukur corak-corak lain. Bukan untuk: workload sangat memuncak dengan lembah melahu (serverless lebih murah) atau produk berbilang pasukan yang benar-benar besar (lihat di atas). Profil: difahami baik di mana-mana, mudah mengambil pekerja, kos sederhana.

Serverless-first: gubah kepingan terurus — API Gateway — fungsi Lambda — DynamoDB, S3, dan queue — tanpa memiliki server langsung. Guna bila: trafik memuncak atau rendah (skala-ke-sifar bermakna kos melahu ~tiada), pasukan kecil, masa-ke-pasaran pantas, perekat dipacu-peristiwa. JANGAN guna bila: pengiraan panjang atau khusus (had runtime), laluan ultra-sensitif-latency (cold start), beban *stabil* yang besar (harga per-panggilan boleh melebihi server reserved — buat aritmetikanya pada skala), atau apabila kemudahan antara cloud ialah keperluan tulen (ini lock-in paling dalam antara semua corak — sering berbaloi, tetapi sebutkannya dengan lantang). Profil: beban operasi terendah dalam katalog; kos hebat pada skala rendah/memuncak, perlu disemak pada skala stabil tinggi.

Perbendaharaan kata:

Istilah	Maksud
Monolith / modular monolith	Satu aplikasi boleh-deploy dengan segalanya di dalamnya / versi berdisiplin: satu boleh-deploy, sempadan modul dalaman yang

Istilah	Maksud
Microservices	bersih — lalai terbaik untuk sistem baharu. Banyak servis kecil yang di-deploy secara bebas, setiap satu memiliki datanya sendiri. Alat penskalaan-pasukan yang mengenakan cukai sistem-teragih.
Coupling / decoupling	Sejauh mana komponen bergantung pada ketersediaan dan perincian satu sama lain. Seni bina sebahagian besarnya ialah seni membeli jumlah penyahgandingan yang betul.
Segerak / asinkron	Panggil-dan-tunggu lawan hantar-dan-teruskan. Pilihan asasi pada setiap anak panah yang anda lukis.
Event / dipacu-peristiwa	Satu fakta yang diumumkan kepada sesiapa yang mendengar (“OrderPlaced”) / seni bina yang dibina daripada pengumuman sebegitu.
Queue (SQS)	Barisan mesej; setiap satu diambil oleh satu pengguna mengikut rentaknya sendiri. Penyerap hentakan dan penyahganding.
Topic / pub-sub (SNS)	Saluran siaran; setiap pelanggan mendapat setiap mesej. Alat fan-out.
Dead-letter queue	Tempat mesej yang berulang kali gagal diproses diketepikan untuk manusia — jaring keselamatan corak ini.
Idempotent	Selamat diproses dua kali dengan hasil sama — diwajibkan ke atas pengguna, kerana queue mungkin menghantar sesuatu mesej lebih daripada sekali.
Serverless-first	Menggubah kepingan terurus skala-ke-sifar (fungsi, DB terurus, queue) dan bukannya menjalankan server.
Lock-in	Kebergantungan pada perkhidmatan proprietari satu penyedia, dihargakan sebagai kos penukaran. Bukan dosa — istilah ekonomi untuk ditimbang secara terbuka (Peringkat 4).

Latihan: (1) Reka bentuk aliran pesanan ThaiTicket dua kali — tiga peringkat segerak, kemudian dipacu-peristiwa dengan SQS/SNS — dan tulis satu perenggan tentang apa yang dilakukan setiap versi semasa gangguan penyedia pembayaran; perenggan itulah keseluruhan hujah untuk peristiwa. (2) Untuk empat syarikat (syarikat pemula 3 orang, syarikat berkembang 50 jurutera, sebuah bank, aplikasi undian TV yang digunakan 4 malam/tahun), pilih satu corak dan pertahankannya — kemudian namakan pencetus yang akan membuatkan

setiap syarikat menukar corak. (3) Cari satu kisah blog kejuruteraan sebenar “kami berpindah ke microservices dan menyesalinya” dan satu kisah kejayaan; jurnalkan perbezaannya (ia hampir selalu saiz pasukan dan kematangan domain).

Pencapaian: diberikan penerangan sesebuah syarikat, anda boleh mengesyorkan satu corak *berserta laluan keluarnya* — “mula di sini; apabila X berlaku, berevolusi ke Y” — dalam lima minit. Laluan evolusi, bukan hukuman muktamad, ialah cara arkitek sebenarnya bercakap.

Modul 9 (Minggu 18–20): Corak data, global, dan hibrid

Seni bina data — lake lawan warehouse. Database transaksi (Peringkat 1) menjalankan perniagaan; analitik mahu *bertanya soalan* kepadanya tanpa melambatkannya. Sebuah **data warehouse** (Redshift, Snowflake, BigQuery) menyimpan data *berstruktur dan dibersihkan* yang dioptimumkan untuk analitik SQL pantas — guna apabila soalan-soalannya diketahui dan dashboard mesti pantas; lebih mahal per TB, memerlukan disiplin pemodelan di awal. Sebuah **data lake** (S3 + katalog + enjin pertanyaan seperti Athena) menyimpan *segalanya, mentah, dengan murah* — log, clickstream, imej — dan menerapkan struktur pada masa bacaan; guna apabila anda mahu menyimpan segalanya sekarang dan memutuskan soalan kemudian; tetapi tanpa tadbir urus ia merosot menjadi jenaka industri, **data swamp** (paya data). Konsensus moden ialah kedua-duanya, berlapis (“lakehouse”): kebenaran mentah dalam lake, mart terkurasi dalam warehouse, disuap oleh **pipeline ETL/ELT**. Peraturan arkitek: analitik tidak sekali-kali membuat pertanyaan ke database production (replika atau pipeline menyuapnya), dan setiap set data ada pemilik, entri katalog, dan polisi penyimpanan (retention) — garisan tadbir urus itu ialah satu ayat dalam reka bentuk anda dan setahun kesengsaraan jika anda meninggalkannya.

Multi-region: active-passive lawan active-active. Apabila satu region tidak mencukupi — kerana perniagaan menuntut DR daripada bencana berskala region, atau pengguna merentangi benua — anda memilih:

Corak	Apa ia	Guna bila	JANGAN guna bila	Kos/kerumitan
Active-passive	Satu region melayani; sebuah region standby memegang data direplikasi, daripada “sandaran sahaja” (sejuk) hingga “salinan kecil sedang berjalan” (suam), dinaikkan taraf semasa bencana	RTO/RPO yang dinyatakan perniagaan mewajarkannya; pematuhan menuntut kisah DR	Tiada sesiapa telah menandatangani RTO/RPO yang mewajarkan perbelanjaan itu (keperluan jujur kebanyakan syarikat ialah multi-AZ yang baik + sandaran merentas region)	Kos 1.1×–1.7× mengikut kesuaman standby; kerumitan sederhana — keperluan pembunuhnya ialah <i>failover yang benar-benar anda latih</i> , jika tidak standby itu hanyalah teater

Corak	Apa ia	Guna bila	JANGAN guna bila	Kos/kerumitan
Active-active	Dua+ region melayani serentak; pengguna dihalakan ke yang terdekat; data direplikasi dua hala	Pangkalan pengguna global mahukan latency tempatan; RTO hampir-sifar benar-benar diperlukan (rangkaian pembayaran, dagangan, SaaS besar)	Hampir semua orang lain — <i>konflik penulisan</i> merentas region ialah salah satu masalah yang benar-benar sukar dalam pengkomputeran	Kos 2×+, kerumitan tertinggi dalam kursus ini; memerlukan kedai data penyelesaian-konflik dan pasukan senior

Ayat bertaraf temu duga: “*Multi-AZ is table stakes; multi-region is a business case.*” (Multi-AZ ialah syarat asas; multi-region ialah kes perniagaan.) Buatlah perniagaan menyatakan RTO/RPO, hargakan pilihan-pilihannya, dan biarkan nombor yang memilih.

Hybrid cloud. Sebahagian on-prem, sebahagian cloud, disambungkan oleh VPN atau Direct Connect — bagi kebanyakan perusahaan mapan bukan satu corak tetapi *realiti sepanjang dekad*: mainframe yang tidak boleh berpindah, sistem kilang terikat-latency, data terkawal selia terpaku on-premises, dan sebuah migrasi (Peringkat 4) sedang melaluinya. Guna sebagai jambatan sengaja dengan arah perjalanan. JANGAN terima: “hybrid” sebagai eufemisme untuk “kami tidak pernah memutuskan.” Nota reka bentuk: identiti mesti disatukan dahulu (satu log masuk merentasi kedua-dua dunia — perkara yang dimaksudkan deskripsi kerja perusahaan dengan “hybrid identity integration”); namakan sistem mana sumber kebenaran; perhatikan kos egress-merentasi-wayar; dan jangkakan pautan rangkaian itu *menjadi* SPOF kecuali digandakan. Kerumitan: segala-galanya untuk dua estet — sebab jujur arkitek menolak untuk mengecilkan sisi on-prem secara berterusan.

Perbendaharaan kata:

Istilah	Maksud
OLTP / OLAP	Pemprosesan transaksi (banyak bacaan/penulisan kecil pantas — menjalankan perniagaan) lawan pemprosesan analitik (imbasan besar — mengkaji perniagaan). Asingkan keduanya.
Data warehouse	Storan berstruktur dan terkurasi yang dioptimumkan untuk analitik SQL pantas (Redshift, Snowflake, BigQuery).
Data lake / data swamp	Storan murah untuk segalanya, mentah, distrukturkan pada masa bacaan (S3 + Athena) / versi tanpa tadbir urusnya, tempat data pergi untuk hilang.

Istilah	Maksud
ETL / ELT	Pipeline yang memindahkan data daripada sistem sumber ke lake/warehouse (Extract, Transform, Load — urutan berbeza-beza).
Tadbir urus data / katalog / retention	Peraturan pemilikan, dokumentasi, dan jangka hayat untuk setiap set data — satu ayat reka bentuk yang menyelamatkan setahun kesengsaraan.
Replikasi (segerak / tak segerak)	Menyalin data secara berterusan ke database atau region lain — konsisten-serta-merta-tetapi-terhad-jarak lawan sedikit-ketinggalan-tetapi-di-mana-mana. Kelewatan tak segerak ialah asal usul RPO.
Active-passive / latihan failover	DR region-standby / raptai berjadual yang membuktikan ia berfungsi. Failover tanpa latihan ialah harapan.
Active-active	Berbilang region melayani serentak dengan replikasi dua hala. Berkuasa; benar-benar sukar; biasanya tidak perlu.
Konflik penulisan	Dua region mengubah data yang sama serentak — sebab teknikal active-active ialah wilayah pakar.
Hybrid cloud / hybrid identity	On-prem + cloud disambungkan sebagai satu estet / satu log masuk merentasi kedua-duanya — perkara pertama untuk disatukan.
Direct Connect	Talian fizikal peribadi yang menyambungkan pusat data ke cloud — tali pusat hibrid, digandakan jika ia penting.

Latihan: (1) ThaiTicket menjadi serantau: reka bentuk pengembangan Singapura dua kali — active-passive (Bangkok sebagai primary) dan active-active — dengan gandaan kos dan RTO/RPO untuk setiap satu; tulis syor satu halaman dan *buat keputusan*. (2) Sebuah peruncit mempunyai 15 tahun jualan dalam sebuah database SQL production dan mahukan “analitik sedia-AI”; lakarkan reka bentuk lake + warehouse dan tulis tiga ayat tadbir urusnya. (3) Kesan corak: pilih tiga seni bina rujukan dari <https://aws.amazon.com/architecture/> dan namakan setiap corak katalog yang hadir dalam setiap satu.

Pencapaian — penghujung Peringkat 3: gauntlet katalog: sebuah AI memberi anda enam senario pantas; untuk setiap satu anda menamakan coraknya, sebabnya, amaran bila-JANGAN, dan profil kos/kerumitan — bawah lima minit setiap satu. Pertukaran yang tepat ini, pada kelajuan yang tepat ini, ialah pertiga tengah temu duga arkitek sebenar.

PERINGKAT 4 — MIGRASI DAN DUNIA SEBENAR (Minggu 21–26)

Reka bentuk greenfield ialah minoriti pekerjaan ini. Kebanyakan seni bina berlaku dalam syarikat yang sudah wujud — dengan bilik server, perisian purba yang perniagaan bergantung padanya, kontrak, dan undang-undang. Peringkat ini ialah dunia itu, dan di sinilah “lead migration and modernization initiatives” — satu butiran dalam hampir setiap deskripsi kerja yang kami kaji — diajar.

Modul 10 (Minggu 21–23): Migrasi — memindahkan sebuah syarikat ke cloud

7 R itu — perbendaharaan kata bersama setiap perbualan migrasi. Untuk setiap workload dalam estet, anda memilih satu:

R	Maksud	Bila	Usaha / pulangan
Retire	Matikannya — tiada sesiapa sebenarnya menggunakannya	Setiap estet mempunyai 10–20% daripada ini; carinya dahulu	Remeh / penjimatan segera — R yang terbaik
Retain	Kekalkan on-prem, buat masa ini	Sistem terikat-latency, terpaku-pematuhan, atau bakal-tamat-hayat	Tiada / menanggung kos — “belum lagi” yang jujur
Rehost (“lift and shift”)	Pindah ke VM cloud seadanya	Kelajuan penting, aplikasi stabil, kemahiran nipis	Rendah / keluar pantas dari pusat data, tetapi manfaat cloud masih sedikit — langkah pertama, bukan destinasi
Relocate	Pindah pada aras hypervisor (cth., armada VMware ke VMware-di-cloud)	Estet tervirtual besar dengan tarikh akhir	Rendah / pemindahan pukal terpantas; sebuah persinggahan
Repurchase (“drop and shop”)	Gantikan dengan SaaS	Aplikasi tidak terbeza — e-mel, HR, CRM	Rendah-sederhana / seluruh kategori keluar dari estet anda
Replatform (“lift, tinker, shift”)	Naik taraf kecil semasa berpindah — DB urus-sendiri — RDS, aplikasi — container	Golongan tengah pragmatik: manfaat sebenar, risiko terkawal	Sederhana / R pekerja keras kebanyakan migrasi
Refactor / re-architect	Tulis semula untuk cloud-native (corak Peringkat 3)	Sistem pembeza teras yang hadnya menyakiti perniagaan	Tinggi / pulangan tertinggi — belanjakannya pada segelintir workload

R	Maksud	Bila	Usaha / pulangan yang layak
---	--------	------	--------------------------------

Kaedah migrasi: assess – mobilize – migrate. Assess (nilai): inventori segalanya (alat penemuan berserta arkeologi bertanya kepada orang), dan skorkan setiap workload atas nilai perniagaan dan kesukaran migrasi — hasilnya: sebuah inventori aplikasi dengan satu R setiap baris, dan sebuah **kes perniagaan** (TCO kekal lawan berpindah; jujurhah bahawa bil *meningkat* semasa tempoh pertindihan apabila kedua-dua estet berjalan — pemimpin yang tidak diberi amaran tentang gelembung migrasi menjadi pemimpin yang membatalkan migrasi di tengah jalan). **Mobilize (gerakkan):** bina **landing zone** — asas cloud pratersedia dan bertadbir *sebelum* workload pertama mendarat: struktur berbilang akaun (akaun berasingan bagi setiap persekitaran dan pasukan, supaya blast radius kekal kecil), identiti dan pengelogan terpusat, hab rangkaian (VPC hub-and-spoke, Direct Connect kembali ke on-prem), dan **guardrail** — polisi automatik yang menjadikan laluan selamat, bertag, dan patuh sebagai lalai (AWS Control Tower ialah kit permulaan terurusnya). Deskripsi kerja menyebut “design the landing zone every workload inherits” — inilah ia. Melangkauinya untuk “terus mula migrasi” mencipta semula pusat data yang berselerak itu di dalam cloud pada kos lebih tinggi; ia tanda tangan klasik migrasi yang gagal. **Migrate dalam gelombang:** kumpulkan workload ke dalam gelombang beberapa unit, disusun mudah-dahulu: Gelombang 1 sengaja bertaruhan-rendah (pelajari jenteranya di tempat kesilapan itu murah), gelombang kemudian mengambil permata mahkota dengan cutover yang dirapatakan, setiap satu dengan pelan undur dan tempoh **hypercare** pengawasan dipertingkat. Untuk setiap pemindahan database, dua soalan yang penting ialah nombor-nombor Modul 4 dalam samaran: berapa lama downtime yang boleh diambil cutover itu (RTO) — dan replikasi berterusan dengan pertukaran singkat wujud apabila jawapannya “hampir tiada.”

Perbendaharaan kata:

Istilah	Maksud
7 R	Retire, retain, rehost, relocate, repurchase, replatform, refactor — menu migrasi per-workload. (Anda juga akan mendengar “6 R” — senarai lama tanpa relocate.)
Discovery / inventori aplikasi	Mengetahui apa yang sebenarnya berjalan (alat + temu bual) / senarai terhasil, satu baris setiap workload, dengan pemilik, kebergantungan, dan R-nya.
Pemetaan kebergantungan	Mencartakan apa bercakap dengan apa — sebab workload berhijrah berkumpulan, dan migrasi tanpanya gagal pada hari pertama.
Kes perniagaan / gelembung migrasi	Hujah TCO kekal-lawan-pindah / bonggol kos sementara semasa kedua-dua estet berjalan. Beri amaran tentangnya atau diserang hendap olehnya.
Landing zone	Asas cloud bertadbir urus dan pratersedia —

Istilah	Maksud
	akaun, identiti, rangkaian, pengelogan, guardrail — yang diwarisi setiap workload. Dibina <i>sebelum</i> migrasi.
Strategi berbilang akaun	Akaun cloud berasingan bagi setiap persekitaran/pasukan, supaya bil boleh dipertanggungjawabkan dan blast radius terkawal.
Guardrail	Polisi automatik yang mencegah atau menandai tindakan tidak patuh — tadbir urus sebagai jentera, bukan memo.
Control Tower	Perkhidmatan terurus AWS untuk mendirikan landing zone berbilang akaun dengan guardrail.
Pelan gelombang	Jadual migrasi dalam kumpulan kecil, mudah-dahulu, kebergantungan bersama-sama.
Cutover / rollback / hypercare	Saat bertukar kepada salinan cloud / undur yang diraptaikan / tetingkap sokongan dipertingkat selepasnya.

Latihan: (1) Migrasi atas kertas: sebuah AI menjana inventori syarikat fiksyen 40 server (anda akan menemuinya lagi sebagai Capstone 2); tetapkan satu R kepada setiap baris dan pertahankan sepuluh keputusan paling sukar. (2) Lakarkan sebuah landing zone: gambar rajah akaun, aliran identiti, hub-and-spoke rangkaian, lima guardrail yang akan anda kuatkuasakan dari hari pertama. (3) Tulis amaran “gelembung migrasi” dua perenggan kepada seorang CFO — berlatih berita-buruk-awal ialah kardio arkitek.

Pencapaian: diberikan inventori ~20 workload dengan penerangan, anda menghasilkan jadual R-per-workload yang boleh dipertahankan, pelan tiga gelombang dengan alasan, dan lakaran landing zone — dalam satu sesi.

Modul 11 (Minggu 24–26): Kekangan — undang-undang, legasi, dan lock-in

Data residency dan undang-undang privasi sebagai input seni bina. Undang-undang data peribadi — **PDPA** Thailand, **GDPR** Eropah, dan sepupu-sepupunya di seluruh dunia — berkongsi satu bentuk yang mesti diketahui seorang arkitek luar kepala: data peribadi memerlukan asas yang sah (sering keizinan); individu mempunyai hak (capaian, pembetulan, pemadaman — reka bentuk anda mesti mampu *mencari dan memadam data seorang individu*, yang sukar jika anda telah menaburkannya merentasi salinan tanpa tadbir urus); pelanggaran membawa kewajipan **notifikasi 72 jam** (pengelogan anda mesti membolehkan anda menceritakan kisahnya dalam masa kurang); dan **peraturan pemindahan merentas sempadan** mengekang di mana data boleh berada secara fizikal — iaitu kekangan *pemilihan region*, kekangan *reka bentuk replikasi* (salinan analitik di Singapura itu mungkin satu peristiwa undang-undang), dan sebab region dalam-negara wujud. Kaedah arkitek, mengikut urutan: kelaskan data (apa yang peribadi?), petakan perjalanannya (setiap kedai, setiap

salinan, setiap sempadan yang dilintasi — aliran yang tiada sesiapa melukisnya ialah tempat pelanggaran hidup), kemudian reka bentuk kawalan: region patuh-residency, penyulitan, jadual penyimpanan, dan laluan pemadaman yang benar-benar mencapai tempoh luput sandaran. Sebut “data protection by design” — ia frasa yang digunakan kedua-dua undang-undang itu, dan ia secara harfiah jawatan anda dalam satu ayat.

Integrasi legasi. Sistem bil mainframe itu tidak akan berpindah tahun ini, dan aplikasi cloud yang berkilat mesti bercakap dengannya. Corak-coraknya: sebuah **anti-corruption layer** — servis penterjemahan antara baharu dan lama, supaya bentuk data ganjil sistem legasi tidak meresap masuk dan merosakkan reka bentuk baharu anda; **strangler fig** (pokok ara pencekik) — halakan trafik melalui sebuah fasad, kemudian kupas fungsi daripada sistem legasi satu demi satu sehingga, bertahun-tahun kemudian, ia boleh dimatikan (dinamakan sempena pokok ara yang perlahan-lahan menyelubungi pokok perumahannya; ia mengalahkan penulisan semula big-bang, yang gagal pada kadar legenda); jambatan batch dan sadapan peristiwa untuk data yang mesti mengalir antara dua dunia; dan hormat terhadap fizik latency panggilan cloud-ke-on-prem yang banyak berbual (Direct Connect membantu; lebih baik lagi, reka bentukkan perbualan itu keluar). Peraturannya: **kurung legasi itu, jangan dijangkitinya** — setiap komponen baharu patut dibina seolah-olah sistem legasi itu sudah tiada.

Ekonomi vendor lock-in. Setiap perkhidmatan terurus yang mudah mendalamkan perkahwinan anda dengan satu penyedia; kemudahalihan (abstraksi multi-cloud, urus-sendiri segalanya) ialah pilihan sebenar yang menelan wang sebenar dalam kerumitan dan kelajuan yang dilepaskan. Kedua-duanya bukan dosa — mod kegagalannya bukan *memilih* lock-in, tetapi tidak *menyedarinya*. Alat arkitek ialah anggaran kos-keluar yang ditulis ke dalam reka bentuk: “Menggunakan DynamoDB menjimatkan ≈ 2 tahun-jurutera sekarang; bertukar kemudian $\approx$ penulisan semula 6 bulan lapisan data — kami menerimanya, dan inilah sempadan antara muka yang akan mengecilkan penulisan semula itu.” Tiga ayat, dihargakan dengan jujur, keputusan direkodkan (ADR Peringkat 5) — itulah pengurusan lock-in yang matang. Multi-cloud *sebagai strategi* (menjalankan workload yang sama secara mudah alih di dua cloud) biasanya jawapan paling mahal yang mungkin dan dibeli oleh tepat organisasi yang skala atau pengawal selianya menuntutnya; multi-cloud *sebagai fakta* (workload berbeza di cloud berbeza kerana sejarah atau kesesuaian) ialah kehidupan biasa.

Perbendaharaan kata:

Istilah	Maksud
PDPA / GDPR	Undang-undang perlindungan data peribadi Thailand dan Eropah — pasangan templat untuk kekangan undang-undang privasi di seluruh dunia.
Data protection by design	Membina kawalan privasi ke dalam seni bina dari lakaran pertama — frasa undang-undangnya dan kewajipan arkitek.
Asas sah / keizinan	Justifikasi undang-undang yang diperlukan untuk memproses data peribadi.
Hak pemadaman (right to erasure)	Hak seseorang individu untuk datanya

Istilah	Maksud
	dipadamkan — yang mesti dijadikan <i> mungkin</i> oleh reka bentuk anda.
Pemetaan aliran data	Mencartakan setiap simpanan, salinan, dan lintasan sempadan bagi satu kategori data. Di mana aliran yang tidak dilukis berada, di situlah pelanggaran hidup.
Pemindahan merentas sempadan	Data peribadi meninggalkan negara — dikawal selia; menjadikan reka bentuk replikasi satu soalan undang-undang.
Anti-corruption layer	Komponen penterjemahan yang menghalang keanehan sistem legasi daripada meresap ke dalam reka bentuk baharu.
Strangler fig	Memodenkan dengan menghalakan melalui fasad dan menggantikan sistem legasi kepingan demi kepingan sehingga ia boleh dimatikan.
Penulisan semula big-bang	Menggantikan sistem sekali gus, bertukar pada satu hari. Kadar kegagalan legenda; strangler fig wujud kerananya.
Kos keluar / kos penukaran	Kos meninggalkan penyedia atau perkhidmatan yang dihargakan secara realistik — nombor yang mengubah lock-in daripada ketakutan kepada ekonomi.
Multi-cloud (strategi lawan fakta)	Sengaja berjalan secara mudah alih di berbilang cloud (mahal, jarang wajar) lawan sekadar mempunyai workload di beberapa (biasa).

Latihan: (1) Petakan aliran data pelanggan ThaiTicket: setiap simpanan, setiap salinan (jangan lupa log, sandaran, pipeline analitik, eksport pasukan sokongan), setiap sempadan; kemudian tulis laluan pemadamannya. Rasakan bagaimana peta itu menemui masalah yang disembunyikan gambar rajah. (2) Reka bentuk pelan strangler-fig untuk sebuah sistem inventori on-prem berusia 20 tahun, tiga kupasan pertama dinamakan. (3) Tulis perenggan lock-in gaya-DynamoDB (manfaat sekarang, kos keluar kemudian, diterima atau dimitigasi) untuk tiga perkhidmatan yang benar-benar akan anda gunakan.

Pencapaian — penghujung Peringkat 4: gauntlet kekangan: satu ringkasan reka bentuk yang sarat dengan ketiga-tiganya (“penanggung insurans Thailand, data pelanggan, sistem polisi mainframe, lembaga pengarah gugup tentang kebergantungan AWS”) — anda menghasilkan pendekatan satu halaman yang menyentuh residency, integrasi, dan lock-in, setiap satu dengan corak bernama dan kos yang jujur. Apabila kekangan lebih mengujakan anda daripada greenfield — kerana kekanganlah tempat arkitek mengatasi pendapatan pelukis-gambar-rajah — Peringkat 4 selesai.

PERINGKAT 5 — KRAF SEORANG ARKITEK (Bulan 7–12)

Segala setakat ini menjadikan anda mampu *mereka bentuk*. Peringkat ini menjadikan anda mampu *bekerja sebagai arkitek* — dokumen, semakan, pembentangan, dan bukti yang sebenarnya membentuk peranan ini, berserta pensijilan yang melepasi anda daripada HR, dan tiga capstone yang menjadi portfolio anda.

Modul 12 (Bulan 7–8): Hasil kerja — ADR, set gambar rajah, dan menjalankan semakan

Architecture Decision Records (ADR). Sebuah ADR ialah dokumen satu halaman yang merakamkan satu keputusan penting: apa yang kami pilih, apa yang tidak, dan mengapa — ditulis *ketika keputusan itu dibuat*, dinomborkan, dan disimpan dalam repositori projek selama-lamanya. Mengapa deskripsi kerja menamakannya: dalam dua tahun, seseorang akan bertanya “kenapa pula ini DynamoDB?” dan ADR itu menjawab dalam tiga puluh saat — dengan konteks, kekangan, dan alternatif yang dipertimbangkan dengan jujur. Pasukan dengan ADR tidak membicarakan semula apa-apa; pasukan tanpanya bertengkar dalam bulatan setiap tahun. Templatnya — hafalkannya:

```
# ADR-014: Use DynamoDB for the session store
Status: Accepted          Date: 2026-08-13
Context:    What situation forced a decision? (Load, constraints,
deadlines – the facts.)
Decision:    What we chose, in one sentence, active voice: "We will..."
Options considered: 2–3 real alternatives, each with honest pros/cons
– including the one you rejected reluctantly.
Consequences: What becomes easier; what becomes harder; the risks we
accept; the exit cost.
```

Tulis satu ADR bagi setiap keputusan penting dalam setiap capstone. Portfolio temu duga yang mengandungi ADR sebenar itu langka dan sangat mematkan keberkesanannya.

Set gambar rajah — satu sistem, tiga aras zum. Dokumentasi seni bina sebenar ialah satu *set* (model C4 mempopularkan disiplin ini): **context diagram** — sistem sebagai satu kotak, dengan pengguna dan sistem luaran yang disentuhnya; pandangan eksekutif/orang baharu, dan yang akan anda bentangkan kepada pimpinan. **Container diagram** — zum masuk: kepingan berjalan utama (aplikasi web, API, database, queue, cache) dan bagaimana ia berkomunikasi; pandangan kejuruteraan harian, lebih kurang lukisan Peringkat 1–3 anda. **Deployment diagram** — di mana semuanya berjalan secara fizikal: region, AZ, VPC, subnet, kumpulan penskalaan; pandangan untuk semakan, keselamatan, dan operasi. Satu sistem, tiga khalayak, tiga gambar rajah — sentiasa dikemas kini, bertarikh, dalam kawalan versi di sebelah ADR. Gambar rajah yang tidak boleh ditemui atau dipercayai ialah gambar rajah yang tidak wujud.

Menjalankan semakan Well-Architected. Anda tahu enam tunjang itu (Peringkat 2); inilah mesyuaratnya sendiri. **Sebelum:** pilih workload dan skop, pastikan set gambar rajah terkini,

jemput orang yang *mengendalikan* benda itu (bukan hanya pereka bentuknya), dan tetapkan nada — ini pemeriksaan kesihatan untuk manfaat pasukan, bukan audit untuk fail sesiapa.

Semasa (separuh hari): susuri tunjang-tunjang dengan set soalan kerangka itu (AWS menerbitkannya, dengan Well-Architected Tool percuma dalam console yang menstruktur keseluruhan latihan —

<https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html>); untuk setiap jawapan, catatkan penemuan, dan *utamakan sambil berjalan* — segenggam isu berisiko tinggi, bukan seratus celaan kecil. Nada penyoalan anda ialah semakan itu: “help me understand what happens when the payment provider times out” (bantu saya faham apa berlaku apabila penyedia pembayaran tamat masa) membuka pintu yang dihempas oleh “you don’t handle timeouts?” (anda tidak mengendalikan timeout?). **Selepas:** laporan ringkas — penemuan teratas, setiap satu dengan risiko, usaha, dan syor — dan item penambahbaikan mendarat dalam backlog sebenar pasukan dengan pemilik, atau semakan itu hanyalah teater. Latihan: jalankan semakan penuh terhadap reka bentuk Capstone 1 anda sendiri; menemui kecacatan sendiri secara berstruktur ialah kemahiran yang paling pantas berganda dalam kursus ini.

Latihan: (1) Isi semula ADR untuk lima keputusan besar dalam reka bentuk jurnal Peringkat 2–3 anda — anda akan menemui sekurang-kurangnya satu yang anda tidak lagi mampu justifikasikan, dan itulah pengajarannya. (2) Hasilkan set tiga-gambar-raja penuh untuk ThaiTicket. (3) Jalankan semakan olok-olok di atas, hasilkan laporan penemuan satu halaman.

Pencapaian: seorang asing (atau sebuah AI yang berlakon sebagai seorang) boleh mengambil pek ThaiTicket anda — tiga gambar raja + lima ADR — dan menjawab dengan betul “apa ini, bagaimana ia berjalan, dan mengapa ia dibina begini?” tanpa anda di dalam bilik. Pek itu *ialah* hasil kerja pekerjaan seorang arkitek.

Modul 13 (Bulan 8–9): Kraf manusia — membentang, mengekos, menangani bantahan, dan membimbing

Membentang kepada eksekutif lawan jurutera — satu reka bentuk, dua bahasa.

Eksekutif membeli hasil; jurutera membeli mekanisme. Kepada eksekutif: mulakan dengan keputusan dan nombornya (“Reka bentuk ini menyokong sasaran sejuta pelanggan pada \$1.90 setiap pesanan, dan inilah dua keputusan yang saya perlukan daripada anda”); satu slaid, gambar rajah *context*, risiko dibingkaikan sebagai risiko perniagaan (hasil, pematuhan, reputasi), dan jangan sekali-kali menyebut “Kubernetes” apabila “platform itu” sudah memadai. Bersedia untuk tiga soalan yang eksekutif sentiasa tanya: *berapa kosnya, apa yang boleh silap, mengapa bukan pilihan yang lebih murah?* Kepada jurutera: mulakan dengan masalah dan kekangan *sebelum* penyelesaian (jurutera yang merasai masalah menerima penyelesaian; jurutera yang dihulurkan hukuman muktamad memburu kecacatan atas prinsip), tunjukkan gambar rajah container dan deployment, namakan sendiri tolak ansur dan pilihan yang ditolak (kredibiliti datang daripada apa yang anda akui), dan tinggalkan ruang sebenar untuk mengubah reka bentuk — soalan-soalan dalam bilik itu ialah semakan percuma. Tabiat arkitek paling buruk ialah satu dek untuk kedua-dua khalayak; tabiat terbaik ialah menulis ringkasan eksekutif *dahulu*, kerana jika reka bentuk itu tidak dapat bertahan dimampatkan kepada lima ayat, ia belum siap.

Menganggar kos untuk sebuah cadangan. Kaedahnya: uraikan reka bentuk kepada ~10 komponen penanggung kosnya; hargakan setiap satu pada beban jangkaan dalam kalkulator harga; **nyatakan andaian anda secara bertulis** (permintaan/hari, pertumbuhan data, egress — andaian itulah anggarannya; apabila ia berubah, nombor itu bergerak dengan hati yang tenang); terapkan strategi reservation pada bahagian yang stabil; tambah senario pertumbuhan (hari ini, 2×, 10× — eksekutif mengingati nombor 10× itu); dan bentangkan sebagai *julat dengan pemacunya* (“\$55–70k/bulan, dipacu terutamanya oleh egress — inilah tuasnya”), jangan sekali-kali satu nombor tunggal berketepatan palsu. Sertakan gelembung migrasi jika ada. Menganggar rendah untuk memenangi kelulusan ialah dosa klasik arkitek junior; projek itu mengingatnya.

Menangani bantahan (pushback). Anda akan ditolak atas kos (“terlalu mahal” — kembali kepada segi tiga: “kita boleh potong \$30k dengan menerima single-AZ — inilah matematik gangguannya; keputusan anda, dan saya akan menuliskannya” — menjadikan pertukaran itu eksplisit dan *direkodkan* menukarkan kebanyakan bantahan kepada sama ada persetujuan atau risiko diterima secara termaklum, kedua-duanya baik); atas cita rasa (“jurutera lebih suka stack berbeza” — perkukuhkan hujahnya dengan lantang, kemudian bawanya kepada kriteria, bukan keutamaan: kesesuaian dengan NFR, kemahiran pasukan, ekosistem, kos keluar — dan apabila ia benar-benar hampir seri, biarkan keutamaan jurutera pelaksana menang, membeli komitmen dengan murah); dan atas autoriti (“kawan CTO kata guna X” — jangan sekali-kali melawan pendapat dengan pendapat; minta kriterianya, jalankan X melalui penilaian awam yang sama seperti segala yang lain, biarkan matriks itu menjawab dengan sopan). Dan apabila anda silap — anda akan mereka bentuk sesuatu yang gagal; setiap arkitek begitu — langkahnya ialah post-mortem tanpa salah-menyalah diterapkan pada diri sendiri, di khalayak, dengan ADR dikemas kini. Tiada apa yang membina kredibiliti sepanjang dekad lebih pantas; tiada apa yang memusnahkannya lebih pantas daripada mempertahankan mayat.

Membimbing jurutera. Frasa deskripsi kerjanya ialah “mentor the engineers who build against your standards,” dan bentuk kerjanya: **waktu pejabat semakan reka bentuk** (pintu yang sentiasa terbuka, supaya bimbingan berlaku sebelum kod, bukan selepas); **menyemak reka bentuk mereka dengan baik hati** — soalan sebelum hukuman, dan sentiasa *sebab* di sebalik piawaian (guardrail yang diterangkan merekrut sekutu; guardrail yang dipaksakan merekrut jalan pintas); **mendelegasikan keputusan sebenar** dengan jaring keselamatan (“anda memiliki reka bentuk cache itu; inilah kekangannya; saya akan semak, dan saya akan menyokong keputusan anda”); dan **mengajar di khalayak** — setiap ADR, setiap tulisan semakan, setiap sesi brown-bag menskalakan anda melangkaui jam anda sendiri. Kebenaran kerjaya yang terus terang: arkitek genius bersendirian mencapai siling; arkitek yang telah membesarkan lima jurutera menjadi rakan sekerja berkeupayaan-reka-bentuk menjalankan seluruh amalan itu.

Latihan: (1) Bentangkan ThaiTicket dua kali — versi eksekutif 5 minit dan versi kejuruteraan 20 minit — rakam kedua-duanya, dan dengar jargon yang meresap ke dalam yang pertama. (2) Hasilkan cadangan kos bertulis penuh (andaian, julat, pemacu, senario 2×/10×). (3) Teater bantahan dengan sebuah AI: tiga pusingan — serangan kos daripada seorang CFO, keutamaan stack daripada seorang jurutera senior, permainan autoriti kawan-CTO — jurnalkan apa yang berkesan. (4) Semak reka bentuk seorang junior (AI boleh menjana yang cacat) secara bertulis, soalan-dahulu, dan minta AI menggeddakan nada anda.

Pencapaian: acara berganda: sampaikan pembentangan eksekutif dan bertahan dua puluh minit soal jawab bercampur yang keras-tetapi-adil (panel AI: seorang CFO, seorang jurutera skeptikal) tanpa silang-jargon, sikap defensif, atau satu pun tolak ansur yang tidak ditulis.

Modul 14 (Bulan 9–12): Pensijilan dan tiga capstone

Laluan pensijilan, dengan garis masa yang jujur. Pensijilan tidak menjadikan anda arkitek — tiga belas modul sebelumnya yang menjadikannya — tetapi ia mendapatkan anda temu duga, dan bersedia untuknya mendawaikan perkhidmatan penyedia itu ke dalam jari anda. Tangganya, pada kadar satu-jam-sehari: **AWS Certified Solutions Architect – Associate (SAA-C03)** — 2–3 bulan; kelayakan arkitek paling diminta dalam iklan kerja, dan selepas Peringkat 1–3 kebanyakannya akan terasa seperti ulang kaji dengan nama perkhidmatan dilekatkan; ambil ini dahulu. **AWS Certified Solutions Architect – Professional (SAP-C02)** — 4–8 bulan selepas Associate; berasaskan senario, benar-benar sukar, dan baris resume tunggal terkuat dalam bidang ini; bahan migrasi dan berbilang akaun Peringkat 4 ialah separuh silibusnya. **Azure AZ-305 (Azure Solutions Architect Expert)** — tambah jika pasaran anda berat-Microsoft (kebanyakan pasaran perusahaan sekurang-kurangnya dwibahasa); jangkakan 2–3 bulan dengan pengetahuan AWS berpindah pada diskaun curam (mula di <https://learn.microsoft.com/en-us/credentials/certifications/azure-solutions-architect/>). **TOGAF Foundation** — pilihan, 2–4 minggu; sesetengah perusahaan besar memintanya; ia memperakui *kaedah* dan perbendaharaan kata seni bina perusahaan dan bukannya kemahiran cloud. Urutan untuk kursus ini: pengajian SAA seiring Bulan 7–8, peperiksaan pada ~Bulan 8; kemudian sama ada SA Pro (laluan teknikal-mendalam) atau AZ-305 (laluan keluasan) hingga Bulan 12, dengan SA Pro diselesaikan menjelang Bulan 14–16 pada kadar yang jujur.

Tiga capstone itu. Setiap satu ialah pek bertaraf portfolio yang lengkap: set gambar rajah tiga aras, 5+ ADR, anggaran kos dengan andaian, dan semakan Well-Architected yang dijalankan sendiri berserta penemuan. Luangkan 3–4 minggu setiap satu. Tiga artifak ini, berserta jurnal anda, *ialah* bukti temu duga anda tentang “end-to-end solution design.”

Capstone 1 — greenfield: sebuah platform e-dagang Thailand untuk 1J pengguna.

Ringkasan: pasaran (marketplace), 1 juta pengguna berdaftar, 50k serentak pada puncak jualan kilat, mudah-alih-dahulu, integrasi pembayaran gaya-PromptPay, patuh PDPA, ketersediaan 99.9%, berjimat bajet peringkat benih. Mesti merangkumi: pilihan corak dengan laluan evolusi, strategi caching, perbualan RTO/RPO ditulis sebagai dialog, anggaran unit-economics, peta aliran data untuk data peribadi.

Capstone 2 — brownfield: memigrasi sebuah syarikat on-prem 40 server. Ringkasan: sebuah firma logistik Thailand; 40 server (ERP atas Oracle, aplikasi pengurusan gudang berusia 12 tahun, perkongsian fail, AD, pelbagai aplikasi jabatan — jana inventori penuh dengan sebuah AI dan bekukannya); pajakan pusat data tamat dalam 14 bulan; lembaga pengarah mahu keluar. Mesti merangkumi: jadual inventori 7-R penuh, pelan tiga gelombang dengan alasan kebergantungan, reka bentuk landing zone, garis masa kos gelembung migrasi, dan memo CFO.

Capstone 3 — yang sukar: DR multi-region untuk sebuah fintech. Ringkasan: sebuah syarikat pembayaran dikawal selia kepada $RTO \leq 15$ minit / $RPO \approx 0$ untuk lejar, dengan

kekangan data-residency ke atas data pelanggan dan ujian failover tahunan yang disaksikan pengawal selia. Mesti merangkumi: analisis active-passive lawan active-active dengan kos, reka bentuk replikasi dengan peta residency, runbook failover, dan pelan latihan.

Rubrik penilaian sendiri (terapkan pada setiap capstone, dengan jujur, dalam jurnal anda):

Dimensi	1 — Belum lagi	3 — Kukuh	5 — Ambil orang ini bekerja
Keperluan	Terus melompat ke penyelesaian	NFR dinyatakan dan dijejaki kepada pilihan reka bentuk	Segi tiga tolak ansur diposisikan, konflik diketengahkan kepada “perniagaan,” keputusan diekstrak
Kualiti reka bentuk	SPOF masih ada; corak tidak sepadan	Corak betul, multi-AZ, kegagalan ditangani	Alasan bila-JANGAN ditunjukkan; laluan evolusi; reka bentuk paling ringkas yang memenuhi keperluan
Enam tunjang	Tunjang diabaikan	Setiap tunjang jelas ditangani	Samakan sendiri menemui kecacatan sebenar — dan reka bentuk disemak semula sebagai tindak balas
Kos	Tiada nombor	Anggaran terperinci, andaian dinyatakan	Julat dengan pemacu, unit economics, senario 10×, strategi reservation
Dokumentasi	Gambar rajah sahaja	Set tiga aras + ADR, notasi betul	Seorang asing boleh menjawab “apa/bagaimana/mengapa” daripada pek itu sahaja
Komunikasi	Satu artifak untuk semua khalayak	Versi eksekutif dan jurutera wujud	Ringkasan lima ayat yang bertahan; bantahan dijawab lebih awal secara bertulis

Skorkan setiap dimensi 4+ pada ketiga-tiga capstone — menyemak semula sehingga anda berjaya — dan anda telah menamatkan kursus ini.

Sepuluh soalan temu duga arkitek (dengan jawapan yang kuat)

Berlatih ini dengan kuat; kekuatannya berada dalam *struktur* setiap jawapan, yang kini anda miliki.

1. “Design a URL shortener / ticketing site / photo app for a million users.” (Reka bentuk pemendek URL / laman tiket / aplikasi foto untuk sejuta pengguna.) Bentuk yang kuat: keperluan dahulu, dengan kuat (“Bacaan mendominasi penulisan? Sasaran ketersediaan? Bajet?”) — kedudukan pada segi tiga — rangka tiga-peringkat atau serverless-first dengan multi-AZ, cache, CDN — namakan tolak ansur tanpa diminta — tamatkan dengan tertib magnitud kos dan laluan evolusi 10×. Penemu duga meluluskan *soalan yang anda tanya*, bukan kotak yang anda lukis.

2. “When would you choose NoSQL over a relational database?” (Bilakah anda akan memilih NoSQL berbanding relational database?) “Saya lalaikan kepada SQL — jaminan ketepatan dan kemahiran sejagat — sehingga satu sebab bernama mengatasinya: skala mendatar melampau, skema fleksibel, atau capaian kunci-nilai satu-digit-milisaat. Kemudian saya pilih perisa NoSQL yang sesuai dengan corak capaian, dan saya tulis ADR-nya termasuk apa yang kami lepaskan: join, dan sebahagian semantik konsistensi.”

3. “Explain RTO and RPO, and how they drive design.” (Terangkan RTO dan RPO, dan bagaimana ia memacu reka bentuk.) Takrifkan kedua-duanya dengan tajam — “ia keputusan perniagaan dengan tanda harga eksponen” — tangganya: sandaran malam — replikasi berterusan — standby suam — active-active, dengan kos meningkat pada setiap anak tangga — “tugas saya ialah membuatkan perniagaan memilih secara sadar, kemudian mereka bentuk tepat kepada nombor itu — dan melatihnya, kerana failover yang tidak diuji ialah harapan.”

4. “Monolith or microservices?” (Monolith atau microservices?) Keputusan Modul 8, kata demi kata: mulakan dengan modular-monolith; ekstrak apabila kesakitan penskalaan-pasukan atau penskalaan-beban benar-benar tiba; microservices ialah alat penskalaan-pasukan yang mengenakan cukai sistem-teragih. Mata bonus untuk “saya lebih rela menjalankan monolith yang baik daripada sistem teragih yang buruk.”

5. “How do you handle a large cloud bill / cost optimization?” (Bagaimana anda menangani bil cloud yang besar / pengoptimuman kos?) “Keterlihatan dahulu — tagging dan showback; kemudian tuaian mengikut urutan usaha: matikan sumber mati, rightsize daripada ukuran, tier storan, jadualkan tidur bukan-prod, kemudian reserve garis dasar stabil. Dan saya laporkan unit economics, bukan jumlah — bil membesar dengan kos-setiap-pesanan menurun ialah kejayaan, bukan masalah.”

6. “How would you migrate a legacy on-prem application?” (Bagaimana anda akan memigrasi sebuah aplikasi legasi on-prem?) “Nilai sebelum berpindah — inventori, kebergantungan, dan satu R setiap workload daripada 7 R; bina landing zone sebelum gelombang pertama; gelombang mudah-dahulu dengan cutover dirapatakan dan pelan undur; dan untuk database permata mahkota, cutover berasaskan replikasi disaizkan mengikut downtime yang ditandatangani perniagaan. Juga: amaran gelembung-migrasi yang jujur dari awal.”

7. “How do you secure a cloud architecture?” (Bagaimana anda melindungi sebuah seni bina cloud?) Susuri lapisan-lapisan: identiti (least privilege, MFA, tiada root harian) — rangkaian (subnet private, segmentasi, permukaan serangan minimum) — data (penyulitan at rest/in transit, pengelasan, residency) — pengesanan (log audit, amaran) — “dan secara reka bentuk, bukan ditampal kemudian — kawalan keselamatan termurah ialah keputusan seni bina yang menghapuskan risiko itu sepenuhnya, seperti tidak sekali-kali menyentuh data kad mentah.”

8. “Tell me about a design decision you got wrong.” (Ceritakan tentang satu keputusan reka bentuk yang anda silap.) Mereka menguji ego, bukan sejarah. Bentuknya: contoh sebenar (capstone) — apa yang anda percayai — apa yang dikatakan realiti — post-mortem itu, ADR yang dikemas kini, corak yang kini anda periksa. Arkitek yang tidak dapat menghasilkan jawapan ini ialah arkitek yang belum pernah disemak.

9. “How do you explain a complex technical decision to a non-technical executive?” (Bagaimana anda menerangkan keputusan teknikal kompleks kepada eksekutif bukan teknikal?) “Keputusan dan nombor perniagaan dahulu, mekanisme hanya atas permintaan; context diagram, bukan container; risiko dalam istilah hasil-pematuhan-reputasi; dan saya bawa dua keputusan yang saya perlukan daripada mereka, ditulis sebagai pilihan dengan tanda harga — eksekutif memutuskan antara pilihan; mereka tidak meluluskan misteri.”

10. “An engineer strongly disagrees with your design. What do you do?” (Seorang jurutera sangat tidak bersetuju dengan reka bentuk anda. Apa anda buat?) “Mula-mula saya perkukuhkan hujah mereka dengan lantang — mereka mungkin betul, dan semakan yang mengubah reka bentuk saya ialah semakan yang berfungsi. Jika ia benar-benar hampir seri, kriteria yang memutuskan (kesesuaian NFR, kemahiran pasukan, kos keluar), bukan senioriti — dan saya biarkan keutamaan pelaksana memenangi seri, kerana komitmen itu murah pada harga itu. Apa pun, keputusan dan pilihan yang ditolak masuk ke dalam ADR, supaya kami tidak sekali-kali bertengkar tentangnya dua kali.”

Silibus satu halaman

Bila	Modul	Fokus	Bukti luaran
Minggu 1–2	1	Penyegar cloud; segi tiga tolak ansur; NFR	Jurnal Reka Bentuk dimulakan
Minggu 3–4	2	Compute/storage/database/rangkaian sebagai keputusan	—
Minggu 5–6	3	Membaca & melukis gambar rajah; kefasihan 3-tier	Pencapaian papan putih 15 minit
Minggu 7–8	4	Kebolehpercayaan: multi-AZ, auto-	—

Bila	Modul	Fokus	Bukti luaran
Minggu 9–10	5	scaling, RTO/RPO Keselamatan: least privilege, zero trust, segmentasi	—
Minggu 11–12	6	Prestasi & kos: caching, CDN, reservation, unit economics	Reka bentuk dihängakan dalam kalkulator
Minggu 13–14	7	Kecemerlangan ops & kelestarian; latihan tubi enam tunjang	Semakan enam tunjang olok-olok
Minggu 15–17	8	Corak: monolith/microservices, dipacu-peristiwa, serverless	—
Minggu 18–20	9	Corak: data lake/warehouse, multi-region, hibrid	Gauntlet katalog
Minggu 21–23	10	Migrasi: 7 R, gelombang, landing zone	Migrasi atas kertas
Minggu 24–26	11	Kekangan: PDPA/GDPR, legasi, ekonomi lock-in	Gauntlet kekangan
Bulan 7–8	12	ADR, set gambar rajah, menjalankan semakan Well-Architected	Pek ThaiTicket; peperiksaan SAA-C03 (persediaan 2–3 bulan)
Bulan 8–9	13	Membentang, mengekos cadangan, bantahan, membimbing	Acara berganda dirakam
Bulan 9–12	14	Capstone 1–3	Portfolio lengkap; SA Pro (4–8 bulan) atau AZ-305 sedang berjalan

Kata penutup daripada guru anda. Dua puluh enam minggu berlalu, anda sudah mampu mereka bentuk; dua belas bulan berlalu, anda mampu *mengamalkannya* — ada perbezaannya, dan itulah perbezaan yang kursus ini dibina untuk merapatkannya. Tabiat-tabiat itu kini kerjaya anda: lukis sebelum anda berhujah, tulis ADR pada hari anda memutuskan, hargakan apa yang anda cadangkan, namakan tolak ansur sebelum sesiapa bertanya, dan besarkan jurutera-jurutera di sekeliling anda sehingga piawaian anda hidup lebih lama daripada

kehadiran anda di dalam bilik. Seni bina ialah pekerjaan teknikal yang jarang-jarang menjadi *lebih baik* apabila anda menua di dalamnya, kerana bahan mentahnya ialah pertimbangan, dan pertimbangan itu berganda. Terus kan jurnal itu. Empat puluh reka bentuk dari sekarang, anda tidak akan memerlukan kursus ini lagi — anda yang akan membetulkannya.

Sumber

Deskripsi kerja dan takrifan peranan yang digunakan untuk pemetaan keperluan (diperoleh Ogos 2026): [templat deskripsi kerja Cloud Architect KORE1](#) · [deskripsi kerja Cloud Architect 4 Corner Resources](#) · [tanggungjawab Solution Architect Microsoft dalam Azure Well-Architected Framework](#) · [iklan Pre-Sales Solutions Architect – AWS milik Arpio](#) · [iklan Solution Architect – Enterprise milik Intel \(melalui Built In\)](#) · [iklan Cloud Domain Architect milik Halliburton](#). Anggaran masa persediaan pensijilan: [tinjauan masa-belajar SAA-C03 CBT Nuggets](#) dan [panduan persediaan SAP-C02 Whizlabs](#); takrifan pensijilan daripada [Microsoft Learn \(AZ-305\)](#) dan [AWS Well-Architected Framework](#). Sebuah jilid pendamping kepada *The Cloud Leader Course* dalam siri B4LCILC.

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 Arkitek Cloud

Aritmetik angka sembilan — SLA, SLO dan bajet ralat

Ketersediaan disebut dalam angka sembilan (nines), dan perkara pertama yang dilakukan seorang arkitek dengan satu angka sembilan ialah menukarnya kepada minit — kerana minit adalah apa yang boleh dirasakan oleh perniagaan. Bulan 30 hari mengandungi 43,200 minit:

Ketersediaan	Masa henti dibenarkan sebulan
99.9% (“tiga sembilan”)	kira-kira 43 minit
99.99% (“empat sembilan”)	kira-kira 4.3 minit
99.999% (“lima sembilan”)	kira-kira 26 saat

Lima sembilan — angka yang gemar disebut kuat-kuat oleh para eksekutif — berada di luar kemampuan sebarang pemulihan kendalian manusia; failover mesti automatik atau janji itu hanyalah dongeng. Setiap tambahan satu angka sembilan menelan kos kira-kira sepuluh kali ganda untuk direkayasa; menukar kepada minit ialah cara anda membuat pihak berkepentingan memilih dengan sedar.

Tiga akronim membawa perbualan ini. Service-Level Indicator (SLI) ialah satu pengukuran. **Service-Level Objective (SLO)** ialah sasaran dalaman pada pengukuran itu. **Service-Level Agreement (SLA)** ialah kontrak luaran, dengan wang terikat pada kegagalan mencapainya. Disiplinnya: SLO sentiasa lebih ketat daripada SLA — anda menyasar di sebelah dalam garisan yang anda dibayar untuk mempertahankannya — dan SLI mengukur apa yang dialami pengguna: sebuah server yang hidup sambil memulangkan ralat adalah mati dalam setiap erti kata yang penting.

Bajet ralat menjadikan SLO operasional: jurang antara kesempurnaan dan objektif, dilayan sebagai sumber untuk dibelanjakan. SLO 99.9% memberikan 43 minit kegagalan sebulan; deployment dan eksperimen berbelanja daripadanya; apabila ia habis, pasukan berhenti menghantar dan bekerja pada kebolehpercayaan — menamatkan pertengkaran paling tua dalam perisian, hantar lawan kestabilan, dengan menggantikan pendapat dengan aritmetik.

Kini perangkapnya: ketersediaan mendarab sepanjang satu rantai. Tiga perkhidmatan bersiri, masing-masing dengan jujur pada 99.9%, tergubah menjadi $0.999 \times 0.999 \times 0.999 \approx 99.7\%$ — lebih kurang 130 minit sebulan, tiga kali ganda daripada apa yang diakui oleh mana-mana satu perkhidmatan. Redundansi bekerja ke arah sebaliknya: dua instance 99.9% yang bebas secara selari gagal bersama-sama hanya $0.1\% \times 0.1\%$ daripada masa — sebabnya multi-zone begitu berbaloi. Ayat bertaraf temu duga: *anda tidak boleh menjanjikan ketersediaan lebih daripada hasil darab kebergantungan segerak anda, maka sama ada pendekkan rantai itu atau rendahkan janji itu.*

Seni bina dalam zaman AI — RAG dan kerabatnya

Menjelang 2026 katalog corak mempunyai satu entri baharu yang ditanya pelanggan setiap minggu: apakah ia, bila ia berbaloi, dan bila ia akan memalukan anda. Sebuah API **Large Language Model (LLM)** — Amazon Bedrock, Azure OpenAI Service — dari segi seni

bina, ialah *sebuah kebergantungan yang mahal, perlahan, dan tanpa keadaan (stateless)*. Ia dimeter mengikut **token** — serpihan perkataan lebih kurang empat aksara. Ia menjawab dalam saat, bukan milisaat, tidak mengingati apa-apa antara panggilan, dan sekali-sekala salah dengan penuh yakin — sebuah komponen dengan mod kegagalan yang luar biasa, yang layak menerima timeout, retry, circuit breaker, dan penggera bajet yang sama seperti mana-mana kebergantungan, kerana gelung panggilan yang terlepas kawalan membakar wang sebenar mengikut jam.

Model itu tidak mengenali dokumen anda; jawapan standardnya ialah **Retrieval-Augmented Generation (RAG)**: bukannya mengajar model itu data anda, anda *mencari* fakta yang relevan pada waktu soalan. Dokumen dipotong menjadi cebisan (chunks), setiap satu ditukar kepada sebuah **embedding** — senarai nombor yang meletakkan maknanya dalam ruang — disimpan dalam sebuah **vector database** (pgvector di AWS, AI Search di Azure), yang memulangkan cebisan-cebisan terdekat. Cebisan itu ditampal ke dalam prompt: model itu bukan merujuk data anda; ia meringkaskan petikan-petikan yang anda ambilkan untuknya.

Di papan putih, RAG ialah dua saluran paip: sebuah laluan pengingesan (ingestion path) yang berjalan lebih awal (dokumen dicebis, di-embed, disimpan) dan sebuah **laluan pertanyaan (query path)** yang berjalan bagi setiap permintaan — dua corak katalog sedia ada, sebabnya para arkitek menguasainya dalam satu petang.

Timbal balasnya ialah tiga yang standard, ditajamkan: kos (setiap pertanyaan menelan wang sebenar — cache yang berulang, halakan soalan mudah kepada model lebih murah), kependaman (saat, bukan milisaat — strimkan jawapan itu), dan privasi (dokumen anda keluar dari sempadan anda; jika vector database itu mengabaikan kawalan akses peringkat dokumen, chatbot itu akan dengan senang hati meringkaskan hamparan gaji untuk sesiapa sahaja yang bertanya). Ketahui bila *tidak* patut membinanya: banyak permintaan “kami perlukan chatbot” ialah kotak carian yang memakai fesyen, dan jawapan tepat yang boleh diaudit — jumlah, baki, lebar — adalah kerja yang salah untuk sistem yang betul 97% daripada masa. Ayat bertaraf temu duga: *RAG ialah masalah carian yang dikokot pada sebuah API peringkasan.*

Mengekos sebuah reka bentuk sebelum sesiapa membinanya

Modul Tonggak Prestasi dan Kos kursus ini kini ditutup dengan satu latihan reka bentuk penuh: penganggaran di bawah tekanan. Taklimatnya: pihak kepimpinan mahukan anggaran kos bulanan untuk sebuah produk pengurusan dokumen sebelum meluluskan pembinaannya. Beban kerjanya — 200,000 pengguna berdaftar, 40,000 aktif setiap hari, masing-masing membuat kira-kira 25 permintaan API sehari dan memuat naik secara purata 2 megabait dokumen sebulan, disimpan selama tujuh tahun. Reka bentuknya: satu lapisan API berimbang beban merentasi dua zon, sebuah database PostgreSQL terurus multi-zone, storan objek, eksport 50-gigabait setiap malam kepada rakan kongsi, dan persekitaran development serta staging yang berasingan. Tugasnya: anggaran bulanan satu halaman dalam angka bulat, dengan andaian dinyatakan secara bertulis sebelum sebarang harga muncul, aritmetik kapasiti ditunjukkan (pengguna — permintaan sesaat — bilangan instance), dan senario hari ini / $2\times$ / $10\times$.

Senarai semak latihan ini mengajar disiplinnya: hargakan pengkomputeran daripada beban puncak, bukan purata; kira standby database itu; modelkan storan sebagai pertumbuhan sepanjang tujuh tahun dengan peringkat kitaran hayat (lifecycle tiers); hitung egress dan bukannya menekanya — baris yang paling kerap dilupakan oleh anggaran dan yang paling kerap mendominasi; ingat gateway NAT (Network Address Translation), persekitaran bukan production, backup, pemantauan, dan pelan sokongan; kenakan diskaun komitmen hanya pada garis dasar yang stabil; dan akhiri dengan satu julat berserta pemacu yang dinamakan serta semakan kewarasan ekonomi unit — kos bagi setiap pengguna aktif sebulan. Satu angka tunggal yang persis adalah salah pada hari ia disebut.