

Cloud ဗိသုကာ သင်တန်း (The Cloud Architect Course)

အခြေခံဗဟုသုတ လုံးဝမရှိသည့်အခြေအနေမှ cloud စနစ်များ ဒီဇိုင်းဆွဲနိုင်သည်အထိ — B4LCILC သင်တန်း
၂၀၂၆ ခုနှစ်၊ ဩဂုတ်လ ၁၃ ရက်

Cloud ဗိသုကာ သင်တန်း (The Cloud Architect Course)

အခြေခံဗဟုသုတ လုံးဝမရှိသည့်အခြေအနေမှ ကုမ္ပဏီတစ်ခုက ၎င်း၏လုပ်ငန်းကို အလောင်းအစားထားပုံသော cloud စနစ်များ ဒီဇိုင်းဆွဲနိုင်သည်အထိ။

ဤသင်တန်း မည်သို့အလုပ်လုပ်သနည်း

သင်သည် cloud စနစ်များကို လည်ပတ်ရန် လေ့ကျင့်နေခြင်း မဟုတ်ပါ။ ၎င်းတို့ကို ဒီဇိုင်းဆွဲရန် လေ့ကျင့်နေခြင်း ဖြစ်သည်။ — ရှုပ်ထွေးသော လုပ်ငန်းပြဿနာတစ်ခု (“ဖောက်သည်တစ်သန်းကို ရောင်းချရမယ်၊ အော်ဒါတစ်ခုမှ မပျောက်ရဘူး”) ကို ယူ၍ ပုံဆွဲထား၊ ကုန်ကျစရိတ်တွက်ထား၊ ခုခံကာကွယ်နိုင်သော နည်းပညာအစီအစဉ်တစ်ခုအဖြစ် ပြောင်းလဲကာ ထို့နောက် ၎င်းကို တည်ဆောက်မည့် engineer များနှင့် ငွေထုတ်ပေးမည့် အုပ်ချုပ်ရေးမှူးများ နှစ်ဦးစလုံးကို ယုံကြည်လက်ခံအောင် ဆွဲဆောင်ရန် ဖြစ်သည်။ ၎င်းသည် **cloud architect** (cloud ဗိသုကာပညာရှင် — *solutions architect*၊ *cloud solutions architect* သို့မဟုတ် *cloud domain architect* ဟူ၍လည်း ကြော်ငြာကြသည်) ၏ အလုပ်ဖြစ်ပြီး ရှင်းလင်းသော သင်ရိုးညွှန်းတမ်းဖြင့် သင်ယူနိုင်သော အတတ်ပညာတစ်ခု ဖြစ်သည်။ — ဤသင်ရိုးပင်တည်း။

သင်တန်းတွင် အဆင့်ငါးဆင့် ရှိသည်။ အဆင့် ၁ (အပတ်စဉ် ၁-၆) အခြေခံအုတ်မြစ်များကို ဆုံးဖြတ်ချက်များအဖြစ်။ Cloud အစိတ်အပိုင်းတိုင်းကို အချက်အလက်တစ်ခုအနေနှင့်မဟုတ်ဘဲ အပေးအယူများပါသော ရွေးချယ်မှုတစ်ခု အဖြစ် ပြန်လည်သင်ကြားခြင်း — အကြောင်းမှာ architect တစ်ဦး၏ အလုပ်ယူနစ်သည် ဆုံးဖြတ်ချက် ဖြစ်သောကြောင့်တည်း။ အဆင့် ၂ (အပတ်စဉ် ၇-၁၄) ဒီဇိုင်းနယ်ပယ်များ။ Well-Architected Framework ၏ မဏ္ဍိုင်ခြောက်ရပ် — reliability (ယုံကြည်စိတ်ချရမှု)၊ security (လုံခြုံရေး)၊ performance (စွမ်းဆောင်ရည်)၊ cost (ကုန်ကျစရိတ်)၊ operations (လည်ပတ်ရေး)၊ sustainability (ရေရှည်တည်တံ့မှု) — တစ်ခုချင်းစီကို ၎င်း၏ စံပုံစံများနှင့် လက်တွေ့ဒီဇိုင်းနမူနာတစ်ခုဖြင့် နက်နက်နဲနဲ သင်ကြားခြင်း။ အဆင့် ၃ (အပတ်စဉ် ၁၅-၂၀) ပုံစံစာရင်း (pattern catalog)။ လက်တွေ့ပြဿနာများ၏ ၉၅% ကို ဖြေရှင်းပေးသော architecture ဆယ့်နှစ်ခုနှင့် — ပို၍အရေးကြီးသည်မှာ — တစ်ခုချင်းစီကို ဘယ်အချိန်

တွင် မသုံးသင့် သနည်း။ **အဆင့် ၄ (အပတ်စဉ် ၂၁-၂၆): Migration နှင့် လက်တွေ့ကမ္ဘာ။** ရှိပြီးသားကုမ္ပဏီများကို cloud ထဲ ရွှေ့ပြောင်းခြင်းနှင့် လက်တွေ့ architecture ကို whiteboard architecture ထက် ခက်ခဲစေသော ကန့်သတ်ချက်များ (ဥပဒေ၊ legacy၊ lock-in)။ **အဆင့် ၅ (လ ၇-၁၂): အတတ်ပညာ။** Architect တစ်ဦးအဖြစ် အလုပ်ခန့်ခံရနိုင်စေမည့် စာတမ်းများ၊ diagram များ၊ review များ၊ တင်ဆက်မှုများနှင့် certification များ — ပြီးလျှင် portfolio အဆင့် ဒီဇိုင်း project သုံးခုဖြင့် အထွတ်အထိပ်တင်ခြင်း။

ဤသင်တန်းက ဘယ်သူ့အတွက်လဲ။ ကြိုတင်ဗဟုသုတ လုံးဝမလိုဟု ယူဆထားသည် — မှီခိုရသမျှအားလုံးကို မသုံးမီ အကျဉ်းချုံး ပြန်သင်ပေးထားသည်။ **The Cloud Engineer Course** ကို ပြီးမြောက်ပြီးသူ (သို့မဟုတ် ယနေ့ cloud တွင် လက်တွေ့အလုပ်လုပ်နေသူ) ဖြစ်လျှင် အဆင့် ၁ သည် ရင်းနှီးပြီးသား ဖြစ်နေမည် — အပေါ်ယံဖတ်ပါ။ သို့သော် လေ့ကျင့်ခန်းများကိုတော့ မဖြစ်မနေလုပ်ပါ။ — အကြောင်းမှာ ၎င်းက အချက်အလက်အဖြစ် သိထားသည်များကို ဆုံးဖြတ်ချက်အဖြစ် ချိန်ဆရမည့်အရာများအဖြစ် ပြန်လည်ဘောင်ခတ်ပေးပြီး ထိုဘောင်ပြောင်းခြင်းသည်ပင် engineer မှ architect သို့ ကူးပြောင်းမှုတစ်ခုလုံး ဖြစ်သောကြောင့်တည်း။

သင်တန်းတစ်ခုလုံးအတွက် စည်းကမ်းများ — **တစ်ရက်လျှင် တစ်နာရီ၊ တစ်ပတ်လျှင် ခြောက်ရက်** လေ့လာပါ။ — ပြင်းထန်မှု ထက် တစ်သမတ်တည်းရှိမှုက ပိုအရေးကြီးသည်။ **Module** တိုင်း၏ အဆုံးတွင် **ဝေါဟာရဇယား** (သင် ပိုင်ဆိုင်ရမည့် စကားလုံးများ)၊ **လေ့ကျင့်ခန်းများ** (စာဖတ်ခြင်းမဟုတ်၊ ဒီဇိုင်းအလုပ် — architecture ကို ပုံဆွဲခြင်းနှင့် ဆုံးဖြတ်ခြင်းဖြင့် သင်ယူရသည်) နှင့် **Milestone** (ရှေ့ဆက်ရန် အသင့်ဖြစ်ကြောင်း သက်သေ။ မပြည့်မီသေးသော milestone ကို ကျော်မသွားပါနှင့်) တစ်ခုစီ ပါရှိသည်။ ထို့ပြင် အပတ်စဉ် ၁ မှစ၍ **Design Journal** (ဒီဇိုင်း မှတ်တမ်းစာအုပ်) တစ်အုပ် ထားပါ။ — သင်ဆွဲသမျှ architecture တိုင်း၊ သင်ခေါ်သမျှ trade-off တိုင်း၊ ကုန်ကျစရိတ် သို့မဟုတ် ကျရုံးမှုအကြောင်း သင် ခန့်မှန်းသမျှတိုင်း။ ဆယ်လအကြာတွင် ၎င်းသည် သင့်အင်တာဗျူး portfolio ဖြစ်လာမည် — မှားခဲ့သည်များအကြောင်း ရိုးသားသော မှတ်စုများပါ။ ရက်စွဲထိုးထားသော ဒီဇိုင်းလေးဆယ်ပါ မှတ်စုစာအုပ်တစ်အုပ်လောက် hiring panel တစ်ခုကို သဘောကျစေသောအရာ မရှိပါ။

Cloud architect တစ်ဦး အမှန်တကယ် ဘာလုပ်သနည်း

အဆောက်အအုံ ဗိသုကာပညာရှင်တစ်ဦးကို မြင်ကြည့်ပါ။ သူမသည် အုတ်များ မစီပါ။ — သို့သော် အုတ်များ ဘာလုပ်နိုင်၍ ဘာမလုပ်နိုင်သည်၊ ဘယ်လောက်ကုန်ကျသည်၊ အဆောက်အအုံများ ဘယ်လိုပြုကျတတ်သည်တို့ကို တိတိကျကျ သိရမည်။ သူမသည် client တစ်ဦး၏ စကားကို နားထောင်ပြီး (“မိသားစုအိမ်၊ နေရောင်ကောင်းကောင်း၊ ဒီဘတ်ဂျက်အောက်၊ ဒီ မြေကွက်ခက်ခက်ပေါ်မှာ”) ဆန္ဒများနှင့် ကန့်သတ်ချက်များကို ဆောက်လုပ်သူများ ဆောက်နိုင်လောက်အောင်၊ client က လက်မှတ်ထိုးအတည်ပြုနိုင်လောက်အောင် တိကျသော ပုံစံများနှင့် သတ်မှတ်ချက်များ အဖြစ် ပြောင်းလဲပေးသည်။ ထို့နောက် ဆောက်လုပ်ရေးကာလတစ်လျှောက် ရှိနေကာ မေးခွန်းများကို ဖြေပြီး မြေကြီးက စစ်တမ်းပြောသည်ထက် ပျော့နေကြောင်း တွေ့ရသောအခါ အစီအစဉ်ကို ချိန်ညှိပေးသည်။

အုတ်များနေရာတွင် server များ အစားထိုးလိုက်လျှင် ထိုအလုပ်ပင် ဖြစ်သည်။ Cloud architect တစ်ဦး၏ တစ်ပတ် တာသည် ဤအရာများ ရောနှောပါဝင်သည်။ — လုပ်ငန်း stakeholder များ၏ စကားကို နားထောင်ပြီး ဝေဝါးသောဆန္ဒများ ထဲမှ လိုအပ်ချက်အစစ်များကို ထုတ်ယူခြင်း။ **ဒီဇိုင်းဆွဲခြင်း** — အစိတ်အပိုင်းများ ရွေးချယ်ခြင်း၊ diagram များ ဆွဲခြင်း၊ ဆုံးဖြတ်ချက်များနှင့် အကြောင်းရင်းများကို ရေးမှတ်ခြင်း။ အခြားသူများ၏ ဒီဇိုင်းများကို မဏ္ဍိုင်ခြောက်ရပ်နှင့် ယှဉ်၍ ပြန်လည်သုံးသပ်ခြင်း။ ဒီဇိုင်းတစ်ခု တစ်လလျှင် ဘယ်လောက်ကုန်မည်ကို ခန့်မှန်းပြီး ထိုဂဏန်းကို ဘဏ္ဍာရေးဌာနရှေ့တွင်

ခုခံကာကွယ်ခြင်း။ စနစ်ဟောင်းများကို cloud ထဲ ရွှေ့ပြောင်းရန် **migration စီစဉ်ခြင်း**။ တင်ဆက်ခြင်း — ဒီဇိုင်းတစ်ခုတည်းကို **engineer** များအတွက် တစ်နည်း၊ အုပ်ချုပ်ရေးမှူးများအတွက် လုံးဝကွဲပြားသော အခြားတစ်နည်းဖြင့် ရှင်းပြခြင်း။ ပြီးလျှင် စံသတ်မှတ်ချက်များ **deadline** များနှင့် တိုက်မိသောအခါ ရှင်သန်ကျန်ရစ်စေရန် **engineer** များကို **လမ်းညွှန်ပြုစုခြင်း (mentoring)**။ အဖွဲ့များစွာတွင် **pre-sales** လည်း ပါသည် — အရောင်းဝန်ထမ်းဘေးတွင်ထိုင်ပြီး ဖောက်သည်လောင်းရှေ့မှောက်၌ စာချုပ်ရအောင် ဆွဲဆောင်ပေးမည့် **solution** ကို ချက်ချင်းပုံကြမ်းဆွဲပြခြင်း။

Architect တစ်ဦး မဟုတ်သည့်အရာ — အခန်းထဲက **code** အတော်ဆုံးလူ (များသောအားဖြင့် မဟုတ်ပါ)၊ စာအရိုက်အများဆုံးလူ (အနည်းဆုံး ရိုက်သည်)၊ သို့မဟုတ် **blueprint** များကို အထက်စီးမှ ချပေးသော လူရည်ချွန်တစ်ကိုယ်တော် (လျစ်လျူရှုခံရရန် အမြန်ဆုံးနည်းလမ်း)။ **Architect** ၏ ထွက်ကုန်အစစ်မှာ **ရေးမှတ်ထားသော၊ အခြားသူများက စိတ်လိုလက်ရ တည်ဆောက်ပေးသော ဆုံးဖြတ်ချက်ကောင်းများ** ဖြစ်သည်။

သင်လေ့ကျင့်နေသည့် အလုပ်ခေါ်စာ (job description)

၂၀၂၆ ခုနှစ် ဩဂုတ်လတွင် ကျွန်ုပ်တို့သည် Cloud/Solutions Architect အလုပ်ခေါ်စာနှင့် **template** အစစ်အမှန်များကို စုဆောင်းခဲ့သည် — **recruiter** တစ်ဦး၏ **cloud-architect template (KORE1)**၊ **staffing** ကုမ္ပဏီတစ်ခု၏ အလုပ်ခေါ်စာ (**4 Corner Resources**)၊ **Azure Well-Architected Framework** ထဲရှိ **Microsoft** ကိုယ်တိုင်၏ **architect** အခန်းကဏ္ဍ အဓိပ္ပာယ်ဖွင့်ဆိုချက်၊ **pre-sales Solutions Architect** ခေါ်စာတစ်ခု (**Arpio**၊ **AWS disaster-recovery**)၊ **enterprise Solution Architect** ခေါ်စာတစ်ခု (**Intel**၊ **Built In** မှတစ်ဆင့်) နှင့် **enterprise Cloud Domain Architect** ခေါ်စာတစ်ခု (**Halliburton**၊ **Azure** ဦးစားပေး)။ ကုမ္ပဏီအရသာများကို ဖယ်လိုက်လျှင် တူညီသော လိုအပ်ချက် ဆယ့်နှစ်ချက်သာ ထပ်ခါထပ်ခါ ပေါ်လာသည်။ ဤသင်တန်းကို ထိုစာရင်းအတိုင်း တည်ဆောက်ထားသည် — တစ်ခုချင်းစီကို ဘယ်နေရာတွင် သင်ကြားသည်ကို ဤတွင် တိတိကျကျ ဖော်ပြထားသည်။ —

အလုပ်ခေါ်စာများ တောင်းဆိုသည့်အချက် (သူတို့စကားလုံးများ၊ ပြန်ချိုးထား)	ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ
“Design scalable and secure cloud architectures tailored to business and technical requirements” (လုပ်ငန်းနှင့် နည်းပညာလိုအပ်ချက်များနှင့် ကိုက်ညီအောင် ချဲ့ထွင်နိုင်၊ လုံခြုံသော cloud architecture များ ဒီဇိုင်းဆွဲခြင်း) — အစအဆုံး solution ဒီဇိုင်း	အဆင့် ၁-၃၊ အဆင့် ၅ တွင် capstone များ
AWS နှင့်/သို့မဟုတ် Azure တွင် “deep platform expertise” (နက်ရှိုင်းသော platform ကျွမ်းကျင်မှု) — compute၊ storage၊ networking၊ IAM၊ account ဖွဲ့စည်းပုံ	အဆင့် ၁ + အဆင့် ၅ ရှိ certification လမ်းကြောင်း
မဏ္ဍိုင်ခြောက်ရပ်နှင့် ယှဉ်၍ “run Well-Architected reviews” (Well-Architected review များ ဦးဆောင်ကျင်းပခြင်း)	အဆင့် ၂ (မဏ္ဍိုင်များ)၊ အဆင့် ၅ Module 12 (review ကျင်းပခြင်း)

အလုပ်ခေါ်စာများ တောင်းဆိုသည့်အချက် (သူတို့စကားလုံးများ၊ ပြန်ချိုးထား)

ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ

“Lead migration/modernization initiatives — which workloads move as-is, get rearchitected, or retire” (မည်သည့် workload များ အတိုင်းရွှေ့၊ ပြန်ဒီဇိုင်းဆွဲ၊ အနားပေးမည်ကို ဆုံးဖြတ်သော migration/ခေတ်မီအောင် ပြုပြင်ရေး ဦးဆောင်ခြင်း)

အဆင့် ၄ Module 10 (7 R များ၊ wave စီစဉ်ခြင်း)

“Design landing zones, account structure, guardrails, and reference patterns teams deploy within” (အဖွဲ့များ deploy လုပ်ရာ landing zone၊ account ဖွဲ့စည်းပုံ၊ guardrail နှင့် reference pattern များ ဒီဇိုင်းဆွဲခြင်း)

အဆင့် ၄ Module 10

“Integrate security and compliance into the design rather than bolting it on” (လုံခြုံရေးနှင့် စည်းကမ်းလိုက်နာမှုကို နောက်မှတပ်ဆင်ခြင်းမဟုတ်ဘဲ ဒီဇိုင်းထဲ ပေါင်းစည်းခြင်း) — zero trust၊ identity

အဆင့် ၂ Module 5၊ အဆင့် ၄ Module 11

High availability နှင့် disaster recovery — “RTO/RPO gaps, downtime costs, ransomware exposure” (RTO/RPO ကွာဟချက်များ၊ ရပ်တန့်ချိန် ကုန်ကျစရိတ်များ၊ ransomware အန္တရာယ်)

အဆင့် ၂ Module 4၊ အဆင့် ၃ Module 9 (multi-region)

“Own the cloud cost model — tagging, showback, reserved capacity, rightsizing” (cloud ကုန်ကျစရိတ်ပုံစံကို ပိုင်ဆိုင်ခြင်း — tagging၊ showback၊ reserved capacity၊ rightsizing)၊ solution ကုန်ကျစရိတ်များ ခန့်မှန်းခြင်း

အဆင့် ၂ Module 6၊ အဆင့် ၅ Module 13 (proposal ကုန်ကျစရိတ်တွက်ခြင်း)

Network architecture ဒီဇိုင်း — VPC များ၊ hub-and-spoke၊ hybrid ချိတ်ဆက်မှု

အဆင့် ၁ Module 2၊ အဆင့် ၃ Module 9၊ အဆင့် ၄

“Create architectural documentation, diagrams, and standards”; “maintain Architecture Decision Records” (architecture စာတမ်း၊ diagram နှင့် စံများ ဖန်တီးခြင်း၊ Architecture Decision Record များ ထိန်းသိမ်းခြင်း)

အဆင့် ၁ Module 3 (diagram များ)၊ အဆင့် ၅ Module 12 (ADR များ၊ diagram အစုံများ)

Stakeholder ဆက်သွယ်ရေး — “present technical concepts to C-level and technical audiences”;

အဆင့် ၅ Module 13

အလုပ်ခေါ်စာများ တောင်းဆိုသည့်အချက် (သူတို့စကားလုံးများ၊ ပြန်ချိုးထား)

ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ

“defend architectural decisions to security, finance, and engineering” (နည်းပညာအယူအဆများကို C-level နှင့် နည်းပညာပရိသတ်များအား တင်ဆက်ခြင်း၊ architecture ဆုံးဖြတ်ချက်များကို security, finance နှင့် engineering ရှေ့တွင် ခုခံကာကွယ်ခြင်း)

“Mentor the cloud and platform engineers who build against your standards” (သင့်စံများအတိုင်း တည်ဆောက်သော cloud နှင့် platform engineer များကို လမ်းညွှန်ပြုစုခြင်း)၊ pre-sales ပံ့ပိုးမှု — demo များ၊ POC များ၊ RFP များ

အဆင့် ၅ Module 13

Certification များ — တူညီသောဈေးကွက်နှင့် တိုက်ဆိုင်စစ်ဆေးပြီး စံလေ့ကားမှာ **AWS Certified Solutions Architect – Associate (SAA-C03)** — အတောင်းဆိုအခံရဆုံး တစ်ခုတည်းသော အသိအမှတ်ပြုလက်မှတ်၊ ပုံမှန်အားဖြင့် ပြင်ဆင်ချိန် ၂-၃ လ — ထို့နောက် **AWS Certified Solutions Architect – Professional (SAP-C02)**၊ ပုံမှန်အားဖြင့် နောက်ထပ် ၄-၈ လ။ Azure အလေးသာသော ကုမ္ပဏီများက **AZ-305 (Azure Solutions Architect Expert)** ကို တောင်းသည်။ Enterprise ကြီးများက enterprise-architecture နည်းစနစ်အတွက် **TOGAF** ကို တစ်ခါတစ်ရံ ထပ်ထည့်သည်။ အစီအစဉ်အပြည့်အစုံမှာ အဆင့် ၅၊ Module 14 တွင် ရှိသည်။

အဆင့် ၁ — အခြေခံအုတ်မြစ်များကို ဆုံးဖြတ်ချက်များအဖြစ် (အပတ်စဉ် ၁-၆)

Cloud Engineer Course ဘွဲ့ရများ — ရှင်းလင်းချက်များကို အပေါ်ယံဖတ်ပါ။ သို့သော် လေ့ကျင့်ခန်းတိုင်းကို လုပ်ပါ။ အချက်အလက်များက အတူတူပင်။ မေးခွန်းများကမူ အသစ်။

Module 1 (အပတ်စဉ် ၁-၂): Cloud ဆိုသည်မှာ ဘာလဲ၊ architect များ ဘယ်လိုတွေးသလဲ

Cloud ကို စာပိုဒ်တစ်ပိုဒ်တည်းဖြင့် (အခြေခံသုည ပြန်လှန်သင်ခန်းစာ)။ Cloud ဆိုသည်မှာ သူတစ်ပါးပိုင် ကွန်ပျူတာများကို နာရီအလိုက် ငှားရမ်းပြီး software ဖြင့် စီမံခန့်ခွဲထားခြင်း ဖြစ်သည်။ AWS၊ Microsoft Azure နှင့် Google Cloud တို့သည် server ဂိုဒေါင်ကြီးများ (data center များ) ကို ပထဝီအလိုက် **Region** များ (ဥပမာ “Asia Pacific (Bangkok)”) အဖြစ် စုစည်းလည်ပတ်သည်။ Region တစ်ခုစီတွင် သီးခြားခွဲထားသော **Availability Zone (AZ)**

အများအပြား ပါဝင်သည် — လျှပ်စစ်သီးခြားစီရှိသော အဆောက်အအုံခွဲများ၊ မြန်ဆန်သောချိတ်ဆက်မှုအတွက် လုံလောက်စွာနီးပြီး ရေကြီးမှု သို့မဟုတ် မီးလောင်မှုတစ်ခုခုက နှစ်ခုစလုံးကို မထိနိုင်လောက်အောင် ဝေးသည်။ ဤအားလုံးကို စက္ကန့်အလိုက် အပိုင်းလိုက် ငှားရမ်းနိုင်သည် — စက်အကြမ်းများ (**IaaS** — ပေါ်ရှိအရာအားလုံးကို သင်စီမံရသည်)၊ စီမံပြီးသား platform များ (**PaaS** — သင့် application တစ်ခုတည်းသာ ယူလာရမည်)၊ သို့မဟုတ် အပြီးသတ် software (**SaaS** — သုံးရုံသာ)။ ဒါက အခြေခံအလွှာတစ်ခုလုံးပင်။ **Architect** တစ်ဦး ဒီဇိုင်းဆွဲသမျှအားလုံးကို ၎င်း၏အပေါ်တွင် စီစဉ်ထားခြင်း ဖြစ်သည်။

ယခု architect ၏ လှုပ်ရှားချက် — အချက်အလက်တိုင်းကို မေးခွန်းအဖြစ် ပြောင်းပါ။ Engineer တစ်ဦးက “AZ ဆိုတာ သီးခြားခွဲထားတဲ့ data center တစ်ခု” ဟု သင်ယူသည်။ Architect တစ်ဦးကမူ ချက်ချင်း မေးသည် — “ဒီ workload က AZ ဘယ်နှစ်ခုနဲ့ ထိုက်တန်သလဲ။” — အကြောင်းမှာ AZ နှစ်ခုသည် တစ်ခုထက် ပိုကုန်ပြီး သုံးခုက နှစ်ခုထက် ပိုကုန်ကာ ကြော်ငြာလက်ကမ်းစာစောင် website တစ်ခုသည် ငွေပေးချေမှုစနစ်တစ်ခုနှင့် ထိုက်တန်သောအရာကို မထိုက်တန်သောကြောင့် ဖြစ်သည်။ ဤသည်မှာ သင်တန်း၏ ပထမဆုံးနှင့် အရေးအကြီးဆုံး အယူအဆ —

Architecture ဆိုသည်မှာ trade-off (အပေးအယူ) များကို ထင်ရှားအောင်ဖော်ထုတ်သည့် ပညာရပ် ဖြစ်သည်။ မှားသော အစိတ်အပိုင်းဟူ၍ မရှိသလောက်ပင် — ဤ workload၊ ဤဘတ်ဂျက်၊ ဤအဖွဲ့၊ ဤ deadline အတွက် မှားနေသော အစိတ်အပိုင်းများသာ ရှိသည်။

Trade-off တြိဂံ။ ဒီဇိုင်းတိုင်းသည် ထောင့်သုံးထောင့်အကြား ညှိနှိုင်းရသည် — **fast** (မြန်ခြင်း — performance၊ အသုံးပြုသူများအတွက် မြန်၊ တည်ဆောက်ရ မြန်)၊ **cheap** (ဈေးသက်သာခြင်း — လစဉ်ဘောလ်နည်း၊ engineering အားစိုက်မှုနည်း)၊ နှင့် **resilient** (ခံနိုင်ရည်ရှိခြင်း — ကျရှုံးမှုများကို ကျော်ဖြတ်နိုင်၊ ချဲ့ထွင်နိုင်၊ လုံခြုံနေ)။ မည်သည့်ထောင့် နှစ်ခုဆီသို့မဆို တွန်းနိုင်သည် — တတိယထောင့်က ၎င်း၏တန်ဖိုးကို ပေးဆပ်ရသည်။ **Startup** တစ်ခု၏ **prototype** သည် **fast** နှင့် **cheap** ဖြစ်သင့်သည် — **resilience** က စောင့်နိုင်သည်။ ဘဏ်တစ်ခု၏ ပင်မစာရင်း (**ledger**) သည် **resilient** နှင့် **fast** ဖြစ်ရမည် — **cheap** တော့ ဖြစ်မည်မဟုတ်ပါ။ **Stakeholder** တစ်ဦးက “သုံးခုလုံး လိုချင်တယ်” ဟုပြောလျှင် သင့်အလုပ်မှာ ပြီး၍ ဘယ်ဟာကို အလိုချင်ဆုံး လဲဟု မေးရန်ဖြစ်သည်။ — အကြောင်းမှာ သူတို့ မဖြေမချင်း ဒီဇိုင်း စတင်၍မရသောကြောင့်တည်း။ ဤသင်တန်းတွင် သင်လုပ်သမျှ ဒီဇိုင်းတိုင်း၏ ထိပ်တွင် ဤတြိဂံကို ဆွဲပြီး **workload** ဘယ်နေရာတွင် ရှိသည်ကို မှတ်သားပါ။ ၎င်းက ငြင်းခုံမှုတစ်ထောင်ကို ကယ်တင်ပေးလိမ့်မည်။

Requirement များ — ဒီဇိုင်း၏ ကုန်ကြမ်း။ Architect များသည် **functional requirement** များ (စနစ်က ဘာလုပ်သနည်း — “ကျောင်းသားတွေ နေ့လယ်စာ မှာနိုင်တယ်”) နှင့် **non-functional requirement (NFR)** များ (ဘယ်လောက်ကောင်းကောင်း လုပ်ရမည်နည်း — “၂ စက္ကန့်အောက်၊ တစ်ပြိုင်နက် ကျောင်းသား ၁၀,၀၀၀ အတွက်၊ အချိန်၏ 99.9%၊ PDPA ဘောင်အတွင်း”) ကို ခွဲခြားကြသည်။ အစပြုသူများသည် ပထမစာရင်းကို အာရုံစိုက်လွန်းသည်။ Architect များကမူ ဒုတိယစာရင်းဖြင့် လစာရအောင် လုပ်ကြသည်။ — အကြောင်းမှာ **architecture** ကို အမှန်တကယ် ဆုံးဖြတ်ပေးသည်မှာ **NFR** များ ဖြစ်သောကြောင့်တည်း။ မိမိတွင် **NFR** ရှိမှန်းမသိသော **stakeholder** များထံမှ ၎င်းတို့ကို ထုတ်ယူပေးသော မှော်မေးခွန်းနှစ်ခု — “*What happens to the business if this is down for an hour?*” (ဒါ တစ်နာရီ ရပ်သွားရင် လုပ်ငန်းက ဘာဖြစ်မလဲ။) နှင့် “*What does success look like at ten times today's size?*” (ဒီနေ့ အရွယ်အစားရဲ့ ဆယ်ဆမှာ အောင်မြင်မှုက ဘယ်လိုပုံစံလဲ။)

ဝေါဟာရများ

ဝေါဟာရ	အဓိပ္ပာယ်
Region / Availability Zone (AZ)	Region: သင်ရွေးချယ်လည်ပတ်ရာ ပထဝီအလိုက် data center အစုအဝေး။ AZ: ၎င်းအတွင်းရှိ သီးခြားခွဲထားသော data center (သို့မဟုတ် အုပ်စုငယ်) — “အဆောက်အအုံ တစ်လုံး ပျက်နိုင်သည်” ၏ ယူနစ်။
IaaS / PaaS / SaaS	စက်အကြမ်းများ ငှားခြင်း / စီမံပြီးသား platform / အပြီးသတ် software။ ထိန်းချုပ်မှုအများဆုံးမှ ပြုပြင်ထိန်းသိမ်းမှု အနည်းဆုံးသို့ ရွေ့သော slider။
Workload	ဒီဇိုင်းယူနစ်တစ်ခုအဖြစ် သဘောထားသော မည်သည့် application သို့မဟုတ် စနစ်မဆို — “the payments workload”။
Functional requirement	စနစ်က မဖြစ်မနေ လုပ်ရမည့်အရာ (“ဖောက်သည်တွေ မှာယူ နိုင်တယ်”)။
Non-functional requirement (NFR)	ဘယ်လောက်ကောင်းကောင်း လုပ်ရမည်နည်း — အမြန်နှုန်း၊ အတိုင်းအတာ၊ uptime၊ လုံခြုံရေး၊ စည်းကမ်းလိုက်နာမှု၊ ကုန်ကျစရိတ်။ NFR များက architecture ကို မောင်းနှင်သည်။
Trade-off	ရွေးချယ်လိုက်သည့်အရာ ရရန် စွန့်လွှတ်လိုက်ရသည့်အရာ။ ဒီဇိုင်းဆုံးဖြတ်ချက်တိုင်းတွင် တစ်ခုစီ ရှိသည်။ Architect ၏ အလုပ်မှာ ၎င်းကို အသံထွက် နာမည်တပ်ပေးရန် ဖြစ်သည်။
Constraint	ညှိနှိုင်း၍မရသော နယ်နိမိတ် — ဘတ်ဂျက်၊ deadline၊ ဥပဒေ၊ ရှိပြီးသားစနစ်များ၊ အဖွဲ့၏ကျွမ်းကျင်မှုများ။ Constraint များသည် ဒီဇိုင်း၏ အတားအဆီးများ မဟုတ် — ၎င်းတို့သည်ပင် ဒီဇိုင်းအမှာစာ ဖြစ်သည်။
Stakeholder	စနစ်တွင် အကျိုးစီးပွားပါဝင်သူ မည်သူမဆို — အသုံးပြုသူများ၊ engineer များ၊ ဘဏ္ဍာရေး၊ လုံခြုံရေး၊ အုပ်ချုပ်ရေးမှူးများ၊ ထိန်းချုပ်စည်းကမ်းချသူများ။ Stakeholder မတူလျှင် ဘာသာစကား မတူ — သင်က အားလုံးကို ပြောတတ်ရမည်။
Greenfield / brownfield	သမိုင်းမရှိသော လုံးဝအသစ်စနစ် (greenfield) နှင့် ရှိပြီးသား စနစ်များနှင့် ရှုပ်ထွေးချိတ်ဆက်နေသောစနစ် (brownfield — လက်တွေ့အလုပ်အများစု)။

ဝေါဟာရ	အဓိပ္ပာယ်
Managed service	Provider က သင့်အတွက် လည်ပတ်ပေးသော အစိတ်အပိုင်း (backup, patching, failover အပါအဝင်)။ ရေးမှတ်ထားသော အကြောင်းပြချက် မရှိသရွေ့ architect ၏ ပုံသေ ရွေးချယ်မှု။

ဤ module အတွက် ဗီဒီယိုများ (link များ စစ်ဆေးပြီး):

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
Top 50+ AWS Services Explained in 10 Minutes	Fireship	~10 min	https://www.youtube.com/watch?v=JlBIYCM48to

Fireship ခရီးစဉ်ကို နှစ်ကြိမ်ကြည့်ပါ — မြေပုံရရန် ယခုတစ်ကြိမ်၊ အဆင့် ၁ အဆုံးတွင် နောက်တစ်ကြိမ် — ဒီဇိုင်းတစ်ခုထဲတွင် ဝန်ဆောင်မှုဘယ်လောက်များများကို နေရာချ တတ်လာပြီဖြစ်ကြောင်း အံ့သြသွားလိမ့်မည်။

လေ့ကျင့်ခန်းများ: (၁) နေ့စဉ်သုံးနေသော app သုံးခု (ဘဏ် app၊ အစားအစာပို့ app၊ ဗီဒီယို app) ကို ရွေးပြီး တစ်ခုချင်းစီအတွက် ထိပ်တန်း NFR သုံးခုကို ရေးကာ trade-off တြိဂံပေါ်တွင် နေရာချပါ။ (၂) မိတ်ဆွေတစ်ဦးကို လုပ်ငန်းအိုင်ဒီယာ တစ်ခုအကြောင်း ဆယ်မိနစ် အင်တာဗျူးလုပ်ပြီး functional ငါးခုနှင့် non-functional requirement ငါးခု ထုတ်ယူပါ — မှော်မေးခွန်းနှစ်ခုကို မေးမှသာ NFR များ ထွက်လာသည်ကို သတိပြုပါ။ (၃) သင့် Design Journal ကို မှတ်တမ်း #1 ဖြင့် စတင်ပါ — တြိဂံနှင့် “ဘယ်ထောင့်ကို စွန့်လွှတ်မလဲ” သည် နည်းပညာမေးခွန်းမဟုတ်ဘဲ လုပ်ငန်းမေးခွန်း ဖြစ်ရသည့် အကြောင်း စာပိုဒ်တစ်ပိုဒ်။

Milestone: မည်သည့် စာကြောင်းတစ်ကြောင်းတည်း စနစ်ဖော်ပြချက်ကိုမဆို ပေးလိုက်လျှင် ၎င်း၏ ဖြစ်နိုင်ခြေရှိသော NFR များနှင့် တြိဂံပေါ်ရှိနေရာကို ငါးမိနစ်အတွင်း၊ အသံထွက်၍၊ မှတ်စုမကြည့်ဘဲ ထုတ်ပြနိုင်ရမည်။

Module 2 (အပတ်စဉ် ၃-၄): Compute, storage, database, network — ဆုံးဖြတ်ချက်လေးရပ်

သင် တစ်သက်တာဆွဲမည့် ဒီဇိုင်းတိုင်းသည် ဤရွေးချယ်မှုလေးခုကို ရည်ရွယ်ချက်ရှိရှိ ချမှတ်ခြင်းသာ ဖြစ်သည်။ တစ်ခုချင်းစီကို ဆုံးဖြတ်ချက်တစ်ခုအဖြစ် သင်ကြားထားသည်။

ဆုံးဖြတ်ချက် ၁ — Compute: VM လား၊ container လား၊ serverless လား။ **Virtual machine (VM)** ဆိုသည်မှာ ကွန်ပျူတာတစ်လုံးလုံးလို အလုပ်လုပ်သော ငှားထားသည့် server အပိုင်းတစ်ပိုင်း ဖြစ်သည် — ထိန်းချုပ်မှု အများဆုံး၊ ပြုပြင်ထိန်းသိမ်းမှုလည်း အများဆုံး (patch ကိုယ်လုပ်၊ scale ကိုယ်လုပ်၊ အလကားနေချိန်လည်း ပိုက်ဆံကိုယ်ပေး)။ **Container** သည် application တစ်ခုကို ၎င်းလိုအပ်သမျှနှင့်အတူ ထုပ်ပိုးပြီး ဘယ်နေရာတွင်မဆို တစ်ပုံစံတည်း

လည်ပတ်စေသည်။ **Orchestrator (Kubernetes)** က ၎င်းတို့၏ အုပ်စုကြီးများကို လည်ပတ်ကုစားပေးသည် — သိပ်သည်းမှုနှင့် ရွှေ့ပြောင်းနိုင်မှု ကောင်းသော်လည်း ကျွမ်းကျင်လူများ လိုအပ်သော ရှုပ်ထွေးသည့် **platform** တစ်ခုကို လက်ခံလိုက်ရသည်။ **Serverless (AWS Lambda, Azure Functions)** သည် သင့် **code** ကို **trigger** ခံရမှသာ လည်ပတ်ပြီး ခေါ်ဆိုမှုအလိုက် ငွေကောက်သည် — ပြုပြင်ထိန်းသိမ်းမှု သုညနီးပါးဖြစ်ပြီး ရုတ်တရက်တက်သော **traffic** အတွက် အကောင်းဆုံး — သို့သော် ကန့်သတ်ချက်များ ရှိသည် (**execution** အချိန်ကန့်သတ်ချက်၊ **cold start** များ၊ **provider** တစ်ခုတည်းနှင့် ပိုနက်သော ထိမ်းမြားမှု)။ အလွတ်ပြန်ဆွဲနိုင်ရမည့် ဆုံးဖြတ်ချက်ဇယား —

ရွေးချယ်ရန်	ဘယ်အခါ	သတိထားရန်
VM	Legacy software၊ အထူး licensing၊ OS အပြည့်ထိန်းချုပ်မှု လိုအပ်ခြင်း၊ တည်ငြိမ်ခန့်မှန်းနိုင်သော ဝန်	Patching၊ scaling နှင့် နံနက် ၃ နာရီ ကို သင်ပိုင်သည်။ အလကားနေချိန် လည်း အပြည့်ကောက်သည်
Container + Kubernetes	ဝန်ဆောင်မှုများစွာ၊ အဖွဲ့တွင် ကျွမ်းကျင်မှုရှိပြီးသား၊ ရွှေ့ပြောင်းနိုင်မှု အရေးကြီးခြင်း	Platform ရှုပ်ထွေးမှု — K8s သည် အချိန်ပြည့်အလုပ်တစ်ခု။ အဖွဲ့ငယ်များ အတွက် လိုသည်ထက်ပို
Serverless	ရုတ်တရက်တက် သို့မဟုတ် ခန့်မှန်းမရသော traffic၊ event-driven ချိတ်ဆက်မှုများ၊ အဖွဲ့ငယ်များ၊ ဈေးကွက်အမြန်ရောက်ရေး	Runtime ကန့်သတ်ချက်များ၊ cold start များ၊ ကြီးမားတည်ငြိမ်သော ဝန်တွင် ကုန်ကျစရိတ် ခန့်မှန်းရခက်ခြင်း၊ lock-in

Senior အမြင် — ဤသည်မှာ **workload** တစ်ခုချင်း ရွေးချယ်မှုဖြစ်သည်။ ကုမ္ပဏီဘာသာရေး မဟုတ်ပါ။ လက်တွေ့စနစ် ပိုင်ဆိုင်မှုကြီးများသည် သုံးမျိုးလုံးကို ဘေးချင်းကပ်၍ မှန်မှန်ကန်ကန် လည်ပတ်ကြသည်။

ဆုံးဖြတ်ချက် ၂ — **Storage: object လား၊ block လား၊ file လား** — ပြီးလျှင် ဘယ်လောက် **hot** လဲ။ **Object storage (Amazon S3)** သည် ဖိုင်များအတွက် အောက်ခြေမရှိသော ခြင်းတောင်း — ဈေးပေါ့၊ မယုံနိုင်လောက်အောင် ခံနိုင်ရည်ရှိ (“eleven nines”)၊ “ဖိုင်တွေ ဘယ်သွားထားမလဲ” ၏ ပုံသေအဖြေ။ **Block storage (EBS)** သည် VM တစ်လုံးတွင် တပ်ဆင်ထားသော virtual disk။ **File storage (EFS)** သည် စက်များစွာ တစ်ပြိုင်နက် mount လုပ်နိုင်သော မျှဝေ drive။ Architect ၏ အပိုအတိုင်းအတာမှာ **အပူချိန်** ဖြစ်သည် — **hot** ဒေတာ (အမြဲသုံး၊ အမြန်နှုန်းအတွက် ဈေးသတ်မှတ်) နှင့် **cold/archive tier** များ (Glacier — ပြားဂဏန်းသာသာ၊ သို့သော် ပြန်ထုတ်ရန် မိနစ်ပိုင်းမှ နာရီပိုင်း)။ ဒေတာဟောင်းများကို **cold tier** ဆီ တဖြည်းဖြည်းရွှေ့ပေးသော **lifecycle rule** များ ဒီဇိုင်းဆွဲခြင်းသည် **cloud** တွင် အဈာန်အသက်သာဆုံး ကုန်ကျစရိတ်အနိုင်ရမှု ဖြစ်သည်။ မေ့လျော့ခြင်းကမူ အဖြစ်အများဆုံး။

ဆုံးဖြတ်ချက် ၃ — **Database: SQL လား၊ NoSQL လား (ပြီးလျှင် ဘယ် managed အမျိုးအစားလဲ)**။ **Relational/SQL database** များ (PostgreSQL၊ MySQL — RDS/Aurora အဖြစ် managed) သည် ဒေတာကို အာမခံထားသော **consistency** ဖြင့် တင်းကျပ်သောဇယားများထဲ ထိန်းသိမ်းသည် — မှန်ကန်မှုက အသက်တမျှအရေးကြီးသော အရာအားလုံးအတွက် ပုံသေရွေးချယ်မှု ငွေကြေး၊ အော်ဒါများ၊ ကုန်ပစ္စည်းစာရင်း၊ **user** များ။ **NoSQL database** များ (DynamoDB၊ MongoDB) သည် တင်းကျပ်သောဖွဲ့စည်းပုံကို စွန့်ပြီး ပြောင်းလွယ်မှုနှင့် အကန့်အသတ်နီးပါးမရှိ

အလျားလိုက်ချဲ့ထွင်နိုင်မှုကို ရယူသည် — session များ၊ catalog များ၊ feed များ၊ telemetry အတွက် ပုံသေရွေးချယ်မှု။ ဆုံးဖြတ်ရေး နည်းဥပဒေ — SQL အလုပ်မဖြစ်နိုင်သည့် တိကျသောအကြောင်းရင်းကို နာမည်တပ်နိုင်မှသာ SQL မှ ခွာပါ (အလွန်အမင်း အတိုင်းအတာ၊ ပြောင်းလွယ်သော schema၊ တစ်ဂဏန်း-millisecond ကမ္ဘာလုံးဆိုင်ရာ ဖတ်ရှုမှုများ)။ ပြီးလျှင် cloud တွင် “database” ဟုဆိုလျှင် အမြဲလိုလို “managed database” ကို ဆိုလိုသင့်သည် — provider က backup၊ patch နှင့် failover ကို တာဝန်ယူသည်။ VM များပေါ်တွင် ကိုယ်ပိုင် database လည်ပတ်နေသော အဖွဲ့တစ်ဖွဲ့တွင် ရေးမှတ်ထားသော အကြောင်းပြချက် ရှိသင့်သည်။ အထူးပြုများကို ဝေါဟာရထဲ ထည့်ပါ — cache (Redis — memory ထဲက hot ဒေတာ၊ microsecond ဖတ်ရှုမှုများ)၊ warehouse (အတိုင်းအတာကြီး analytics — အဆင့် ၃)၊ queue (database မဟုတ်သော်လည်း မကြာခဏ ပျောက်နေသည့် အစိတ်အပိုင်း — အဆင့် ၃)။

ဆုံးဖြတ်ချက် ၄ — **Network: သီးသန့်ကမ္ဘာ၏ ပုံသဏ္ဌာန်။ VPC (Virtual Private Cloud)** ဆိုသည်မှာ provider ၏ network ထဲမှ သင့်အတွက် ခြံခတ်ခွဲထားသောအပိုင်း ဖြစ်သည်။ ၎င်းအတွင်း **public subnet** များက အင်တာနက် လက်လှမ်းမီနိုင်သောအရာများ (load balancer များ) ကို ထားရှိပြီး **private subnet** များက ကျန်အားလုံး — app server များနှင့်၊ အမြဲတမ်း၊ database များ — ကို ထားရှိသည်။ “The database sits in a private subnet” (database က private subnet ထဲမှာ ရှိတယ်) သည် architecture review များတွင် အထပ်ခါဆုံး စာကြောင်း ဖြစ်သည်။ အကြောင်းရင်း — အင်တာနက်မှ လမ်းကြောင်းမရှိသောအရာကို ဘာကမှ မတိုက်ခိုက်နိုင် — သည် network လုံခြုံရေး၏ တစ်ဝက် ဖြစ်သည်။ VPC ပတ်လည်တွင် — **load balancer** က traffic ကို server များအနှံ့ ဖြန့်ပြီး မကျန်းမာသည်များကို ရှောင်ကွင်းပေးသည်။ **DNS (Route 53)** က နာမည်များကို လိပ်စာအဖြစ် ပြောင်းပေးသည်။ **CDN (CloudFront)** က content ကို မြို့ရာနှင့်ချီတွင် cache လုပ်ထားသဖြင့် နေရာတိုင်းတွင် မြန်သည်။ **API gateway** သည် သင့် API များအတွက် managed ဧည့်ကြိုစားပွဲ (authentication၊ rate limit များ၊ logging)။ ကမ္ဘာဟောင်းနှင့် ချိတ်ဆက်ခြင်း — **VPN** (အင်တာနက်ပေါ်မှ ကုဒ်ဂှက်လမ်းကြောင်း) သို့မဟုတ် **Direct Connect** (သီးသန့် ရုပ်ပိုင်း ဆိုင်ရာလှိုင်း) — အဆင့် ၄ ရှိ hybrid ဒီဇိုင်းတိုင်း၏ ချက်ကြီးများ။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Instance / instance type	ငှားထားသော VM တစ်လုံး / ၎င်း၏ အရွယ်အစား (CPU + RAM) — နာရီဈေးကို သတ်မှတ်ပေးသည်။
Container / Docker / Kubernetes (K8s)	App တစ်ခုအတွက် ရွှေ့ပြောင်းနိုင်သော ပက်ကေ့ဂျ် / ၎င်းတို့ကို တည်ဆောက်လည်ပတ်ပေးသောကိရိယာ / ၎င်းတို့၏အုပ်စုကြီးများကို လည်ပတ်ကုစားပေးသော orchestrator။
Serverless / Lambda	Trigger ခံရမှသာ လည်ပတ်ပြီး လည်ပတ်မှုအလိုက် ငွေကောက်ကာ စီမံရမည့် server မရှိသော code။ Lambda သည် AWS ၏ ဗားရှင်း။

ဝေါဟာရ	အဓိပ္ပာယ်
Cold start	Serverless function တစ်ခု အလကားနေပြီးမှ လည်ပတ်ရသည့်အခါ ထပ်ဆောင်းနှောင့်နှေးမှု — ဂန္ထဝင် serverless trade-off။
S3 / bucket / durability	AWS object storage / အမည်ပေးထားသော ဖိုင်ကွန်တိန်နာတစ်ခု / ဒေတာ ရှင်သန်ကျန်ရှိနိုင်ခြေ — S3 ၏ 99.999999999% ဆိုသည်မှာ ဆုံးရှုံးမှုသည် လက်တွေ့တွင် ဘယ်တော့မှ မဖြစ်သလောက်။
Storage tier / lifecycle policy	ဈေးနှုန်း-အမြန်နှုန်း အတန်းအစား (hot → cold → archive) / အိုဟောင်းလာသောဒေတာကို ပိုဈေးသက်သာသော tier များဆီ ရွှေ့ပေးသော အလိုအလျောက်စည်းကမ်း။
RDS / Aurora / DynamoDB	AWS ၏ managed SQL ဝန်ဆောင်မှုများ / ၎င်း၏ cloud-native စွမ်းဆောင်ရည်မြင့် SQL / ၎င်း၏ အဓိက managed NoSQL။
Consistency	ဒေတာဖတ်သူတိုင်း တစ်ချိန်တည်းတွင် တူညီသောအမှန်တရားကို မြင်ရမည်ဟူသော အာမခံချက် — SQL ၏ စူပါပါဝါ၊ NoSQL က ချဲ့ထွင်နိုင်မှုရရန် လျှော့ပေးလိုက်သောအရာ။
Cache / Redis	Database တစ်ခုရှေ့တွင် ထားသော hot ဒေတာ၏ memory-အမြန်နှုန်း မိတ္တူ / ၎င်းအတွက် စံကိရိယာ။
VPC / subnet (public, private)	သင့်သီးသန့် network အပိုင်း / ၎င်း၏ အခွဲများ — public က အင်တာနက်ကို မျက်နှာမူသည်၊ private က မမူပါ။ Database များသည် private ထဲနေသည်။ အမြဲတမ်း။
Load balancer / health check	Server များအနှံ့ traffic လမ်းညွှန် / သေနေသည်များဆီ traffic ပို့ ခြင်းရပ်ရန် ၎င်းသုံးသော နှလုံးခုန်စစ်ဆေးမှု။
CDN / edge	ကမ္ဘာအနှံ့ မြို့အဆင့် သင့် content ၏ cache များ / “the edge” = အသုံးပြုသူများနှင့် နီးသောနေရာ။
API / API gateway	Software တစ်ခုက အခြားတစ်ခုအား ပေးထားသော ထိန်းချုပ်ထားသည့် တံခါးပေါက် / ထိုတံခါးပေါက်များအတွက် managed ဧည့်ကြိုစားပွဲ။
VPN / Direct Connect	အင်တာနက်ပေါ်မှ ကုန်သွယ်လမ်းကြောင်း / cloud ဆီသို့ သီးသန့် ရုပ်ပိုင်းဆိုင်ရာလိုင်း — on-prem နှင့် cloud တွေ့ဆုံရာ နည်းလမ်းနှစ်သွယ်။

ဤ module အတွက် ဗီဒီယိုများ (link များ စစ်ဆေးပြီး):

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
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

လေ့ကျင့်ခန်းများ: (၁) အောက်ပါ workload ငါးခုစီအတွက် compute၊ database နှင့် storage ကို ရွေးပြီး တစ်ခုစီအတွက် အကြောင်းပြချက် စာကြောင်းတစ်ကြောင်းစီ ရေးပါ — ကျောင်းနေလယ်စာ မှာယူရေး site တစ်ခု။ ဘဏ်တစ်ခု၏ ငွေလွှဲစာရင်း (transaction ledger)။ ဓာတ်ပုံမျှဝေ app တစ်ခု။ ညစဉ် မိနစ် ၂၀ လည်သော report ထုတ်စက်တစ်ခု။ အသုံးပြုသူ ၅ သန်းအတွက် chat app တစ်ခု။ (၂) သင့်ရွေးချယ်မှုတစ်ခုကို ယူပြီး ဆန့်ကျင်ဘက် ရွေးချယ်မှုကို တတ်နိုင်သမျှ ယုံကြည်လက်ခံစရာကောင်းအောင် ငြင်းခုံကြည့်ပါ — အခြားရွေးချယ်စရာကို steel-man မလုပ်နိုင်သော architect များသည် trade-off ကို နားမလည်သေးခြင်း ဖြစ်သည်။ (၃) Journal — သင့် compute ဆုံးဖြတ်ချက်ဇယားကို အလွတ်။

Milestone: မည်သည့် workload ကိုမဆို စာကြောင်းတစ်ကြောင်းဖြင့် ပေးလိုက်လျှင် ၎င်း၏ ဆုံးဖြတ်ချက်လေးရပ်ကို အကြောင်းပြချက်များနှင့်တကွ သုံးမိနစ်အောက်အတွင်း ပြောနိုင်ရမည် — ပြီးလျှင် အနည်းဆုံး ဆုံးဖြတ်ချက်တစ်ခုအတွက် သင့်စိတ်ပြောင်းစေမည့်အရာကို နာမည်တပ်နိုင်ရမည်။

Module 3 (အပတ်စဉ် ၅-၆): Architecture diagram များ ဖတ်ခြင်းနှင့် ဆွဲခြင်း

ပုံမဆွဲတတ်သော architect သည် စကားသာပြောတတ်သော consultant တစ်ဦးမျှသာ ဖြစ်သည်။ Diagram များသည် သင့်အလုပ်သုံးဘာသာစကား ဖြစ်သည် — ဤ module က ၎င်းတို့ကို ဖတ်ရာတွင် ကျွမ်းကျင်စေပြီး ဆွဲရာတွင် အရည်အချင်းပြည့်မီစေသည်။

စံနမူနာ diagram — ဤတစ်ခုကို အရင်သင်ပါ။ **Three-tier architecture** (အလွှာသုံးလွှာ တည်ဆောက်ပုံ) သည် cloud diagram များ၏ “ဝါကျဖွဲ့စည်းပုံ” ဖြစ်သည်။ ဒီဇိုင်းအများစုသည် ၎င်း၏ ကွဲပြားမှုပုံစံများသာ ဖြစ်သည် —

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

→ [Cache]

→ [Database: primary +

standby, private subnets]

Tier 1 (**presentation**): အသုံးပြုသူများ ထိတွေ့ရာ — CDN မှ static content၊ load balancer မှတစ်ဆင့် request များ။ Tier 2 (**application**): သင့် logic ကို လည်ပတ်ပေးသော အပြန်အလှန်အစားထိုးနိုင်သည့် server အုပ်စု — private subnet များထဲတွင်၊ အလိုအလျောက် ချဲ့ထွင်ခံ။ Tier 3 (**data**): အနက်ဆုံး၊ အကာအကွယ်အများဆုံး database — ဒုတိယ AZ တစ်ခုတွင် standby တစ်ခုနှင့်အတူ။ Traffic သည် တစ်လမ်းသွားစီးဆင်းသည် — အသုံးပြုသူများသည် app tier ကို ဘယ်တော့မှ တိုက်ရိုက်မထိ၊ app tier တစ်ခုတည်းသာ data tier နှင့် စကားပြောသည်။ ဦးနှောက်မပါဘဲ လက်ကရောက်ဆွဲနိုင်သည်အထိ လေ့ကျင့်ပါ — ၎င်းသည် ကမ္ဘာပေါ် နေရာတိုင်းရှိ whiteboard အင်တာဗျူး၏ အဖွင့်မေးခွန်း ဖြစ်သည်။

ကျွမ်းကျင်သူပုံပေါက်စေသော notation ထုံးစံများ (အကြောင်းမှာ ၎င်းတို့က သင့်ကို ကျွမ်းကျင်သူလို တွေး စေသောကြောင့်): လေးထောင့်ကွက်များသည် အစိတ်အပိုင်းများ — တစ်ခုစီကို ၎င်းဘာလဲ နှင့် ဘယ်ဝန်ဆောင်မှုလဲ (“App servers — EC2, auto-scaling group”) ဟု တံဆိပ်တပ်ပါ။ မြားများက request တစ်ခု စီးဆင်းသောဦးတည်ရာကို ပြပြီး အရေးကြီးလျှင် protocol ကို တံဆိပ်တပ်ပါ။ မျဉ်းပြတ်ဘောင်များက နယ်နိမိတ်များကို ပြသည် — VPC၊ subnet တစ်ခုစီ၊ AZ တစ်ခုစီ (AZ နယ်နိမိတ်များကို ဆွဲလိုက်လျှင် multi-AZ သည် ကြွေးကြော်ချက်မဟုတ်တော့ဘဲ မြင်သာလာသည်)။ User/actor သည် စနစ်၏အပြင်ဘက်တွင် ရပ်သည်။ မြားများပေါ်ရှိ နံပါတ်များ (1၊ 2၊ 3...) က request တစ်ခု၏ ခရီးစဉ်ကို ပြောပြနိုင်စေသည်။ ပြီးလျှင် diagram တိုင်းတွင် ခေါင်းစဉ်၊ ရက်စွဲနှင့် legend ပါရသည်။ ပိုနက်သော စည်းကမ်း — **diagram တစ်ခု၊ ပရိသတ်တစ်မျိုး၊ မေးခွန်းတစ်ခု**။ အားလုံးပြသော diagram သည် ဘာမှမပြပါ။ Zoom အဆင့်စံအစုံ (context → container → deployment) ကို အဆင့် ၅ တွင် သင်ယူမည်။

သူတစ်ပါး၏ diagram များ ဖတ်ခြင်း — **architect ၏ ဓာတ်မှန်**။ Diagram တစ်ခု လက်ခံရလျှင် ဤစစ်ဆေးမှုကို အသံထွက်လုပ်ပါ — အင်တာနက်က ဤစနစ်ကို ဘယ်နေရာတွင် ထိသနည်း (ထိတွေ့ရာတိုင်းသည် attack surface)။ ဒေတာက ဘယ်မှာလဲ၊ private subnet ထဲမှာလား။ ဘာက ပုံတူပွားထား (ခံနိုင်ရည်ရှိ) ပြီး ဘာက **single point of failure** — အမွှာမပါသော လေးထောင့်ကွက် — လဲ။ Traffic ၁၀ ဆတွင် ဘယ်နေရာက နာမလဲ။ ကွက်တစ်ကွက်စီက တစ်လ ဘယ်လောက်ကုန်မလဲ။ မေးခွန်းငါးခု၊ စက္ကန့်သုံးဆယ် — ဆရာဝန်တစ်ဦး ဓာတ်မှန်ဖတ်သလို diagram ကို ဖတ်ပြီး ပြီ။ AWS Architecture Center (<https://aws.amazon.com/architecture/>) ကို လှန်လှောကြည့်ပြီး ထုတ်ဝေထားသော reference architecture သုံးခုပေါ်တွင် ဤစစ်ဆေးမှုကို လုပ်ပါ။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Three-tier architecture	ဂန္ထဝင် ခွဲခြားမှု — presentation (အသုံးပြုသူဝင်ပေါက်) → application (logic) → data (database)။ Web စနစ်များ၏ ပုံသေပုံသဏ္ဌာန်။
Tier / layer	တာဝန်တစ်ခုတည်းရှိပြီး အိမ်နီးချင်းများနှင့်သာ စကားပြောသော စနစ်၏ အလျားလိုက်အလွှာတစ်လွှာ။

ဝေါဟာရ	အဓိပ္ပာယ်
Single point of failure (SPOF)	၎င်းတစ်ခုတည်း သေရုံဖြင့် စနစ်တစ်ခုလုံး ကျသွားစေသော မည်သည့်အစိတ်အပိုင်းမဆို။ Diagram တိုင်းတွင် ပထမဆုံး လိုက်ရှာရမည့်အရာ။
Multi-AZ	Data center တစ်ခု မသိမသာ ပျက်သွားနိုင်စေရန် အနည်းဆုံး Availability Zone နှစ်ခုအနွံ ပုံတူများ လည်ပတ်ခြင်း။
Auto-scaling (group)	ဝယ်လိုအားအလိုက် စက်များ အလိုအလျောက် ထည့်ခြင်းဖယ်ခြင်း — အသက်ရှူနေသော စွမ်းရည်။
Stateless / stateful	ထူးခြားသောဒေတာ မကိုင်ထားသော server (မည်သည့်အမှုှာ မဆို အစားထိုးနိုင်၍ လွတ်လပ်စွာ ချဲ့နိုင်သည်) နှင့် မပျောက်ရမည့်ဒေတာ ကိုင်ထားသော server ။ ဒီဇိုင်းရည်မှန်းချက် — stateless app tier ၊ state ကို database နှင့် cache ဆီ အောက်ချ။
Reference architecture	ဖြစ်လေ့ရှိသောပြဿနာတစ်ခုအတွက် provider ထုတ်ဝေထားသော ထောက်ခံပြီးသား နမူနာဒီဇိုင်း — architect များသည် တီထွင်မည့်အစား ဤသည်တို့မှ စုစည်းတပ်ဆင်ကြသည်။
Context diagram	အမြင့်ဆုံး zoom အဆင့် — သင့်စနစ်ကို ကွက်တစ်ကွက်တည်းအဖြစ်၊ ၎င်းထိတွေ့သော အသုံးပြုသူများနှင့် ပြင်ပစနစ်များနှင့်အတူ။
Attack surface	ပြင်ပကမ္ဘာက စနစ်ကို ထိနိုင်သောနေရာတိုင်း။ သေးလေ လုံခြုံလေ။
North–south / east–west traffic	စနစ်ထဲ ဝင်/ထွက်သော traffic နှင့် စနစ်အတွင်းရှိ အစိတ်အပိုင်းများအကြား traffic ။

လေ့ကျင့်ခန်းများ: (၁) **Three-tier diagram** ကို အလွတ်ဆွဲပါ — ငါးရက်ဆက်တိုက်၊ နယ်နိမိတ်အားလုံး (**VPC**၊ **subnet** များ၊ **AZ** နှစ်ခု) ပါလျက် လေးမိနစ်အောက် ရောက်သည်အထိ။ (၂) **Module 2** ၏ **workload** ငါးခုကို ယူပြီး တစ်ခုချင်း ဆွဲပါ — **diagram** တစ်ခုလျှင် မိနစ်နှစ်ဆယ်၊ **notation** စည်းကမ်းများ တင်းကျပ်စွာ။ (၃) အွန်လိုင်းတွင် **architecture diagram** အစစ်တစ်ခု ရှာပါ (**AWS Architecture Center** တွင် ရာနှင့်ချီရှိသည်) ပြီးလျှင် ၎င်း၏ မေးခွန်းငါးခု ဓာတ်မှန်စစ်ဆေးမှုကို **journal** ထဲ ရေးပါ။ (၄) ပြောပြလေ့ကျင့်ခန်း — **diagram** ကို ရှေ့တွင်ထား၊ **user** တစ်ဦး၏ **click** တစ်ချက်ကို **browser** မှ **database** အထိ အသွားအပြန်၊ အသံထွက်၍၊ မြားများကို နံပါတ်တပ်ရင်း ပြောပြပါ။

Milestone — အဆင့် ၁ အဆုံး: whiteboard (သို့မဟုတ် journal အတွက် ဓာတ်ပုံရိုက်ထားသော စက္ကူ) ပေါ်တွင် စာကြောင်းတစ်ကြောင်းတည်းပါ workload အသစ်တစ်ခုအတွက် မှန်ကန်ပြီး notation စနစ်ကျသော three-tier ဒီဇိုင်းကို ဆယ့်ငါးမိနစ်အောက်ဖြင့် ဆွဲနိုင်ရမည်။ request တစ်ခုကို ၎င်းအထဲဖြတ်၍ ပြောပြနိုင်ရမည်။ ပြီးလျှင် ကွက်တိုင်းအတွက် “ဒီ ကွက် သေသွားရင် ဘာပျက်မလဲ” ကို ဖြေနိုင်ရမည်။ ဤပုံဆွဲခြင်းနှင့် စစ်ဆေးမေးမြန်းခြင်းသည်ပင် architect အင်တာဗျူး အစစ်၏ ပထမတစ်ဝက် အတိအကျ ဖြစ်သည် — ဤနေရာမှစ၍ ကျန်အားလုံးသည် အနက်သာ ဖြစ်သည်။

အဆင့် ၂ — ဒီဇိုင်းနယ်ပယ်များ (အပတ်စဉ် ၇-၁၄)

ဤအဆင့်၏ မြေပုံမှာ **AWS Well-Architected Framework** ဖြစ်သည် — “စနစ်တကျ ဒီဇိုင်းဆွဲထားသည်” ၏ အဓိပ္ပာယ်ကို လုပ်ငန်းနယ်ပယ်တစ်ခုလုံး မျှဝေထားသော checklist ဖြစ်ပြီး မဏ္ဍိုင်ခြောက်ရပ်ဖြင့် ဖွဲ့စည်းထားသည် — **operational excellence, security, reliability, performance efficiency, cost optimization, sustainability**။ (Azure တွင် နီးပါးတူညီသော framework ရှိသည်။ တစ်ခုကို နက်နက်သင်လျှင် နှစ်ခုလုံး သင်ပြီးသား ဖြစ်သည်။) အလုပ်ခေါ်စာများက “run Well-Architected reviews” ကို နာမည်တပ်တောင်းသောကြောင့် မဏ္ဍိုင်များကို တစ်ခုချင်းယူပြီး တစ်ခုစီအတွက် အရာသုံးခု သင်ယူမည် — မဏ္ဍိုင်၏ **အဓိကမေးခွန်းများ**၊ ၎င်း၏ **စံပုံစံများ** (ငြီးငွေ့ဖွယ် ကောင်းသော်လည်း သက်သေပြပြီးသား အဖြေများ — architect များသည် တီထွင်မီ စုစည်းတပ်ဆင်ကြသည်)၊ နှင့် ဆက်တိုက်သုံး **scenario** တစ်ခုပေါ်တွင် **လက်တွေ့နမူနာ** တစ်ခု။

အဆင့် ၂ တစ်ခုလုံးအတွက် ဆက်တိုက်သုံး scenario: ThaiTicket — ဘန်ကောက်ရှိ စိတ်ကူးယဉ် ပွဲလက်မှတ်ရောင်း platform တစ်ခု။ ပုံမှန်ဝန် — တစ်နာရီလျှင် ဧည့်သည် ၂,၀၀၀။ သို့သော် နာမည်ကြီး အနုပညာရှင်တစ်ဦး၏ လက်မှတ်များ နံနက် ၁၀:၀၀ တွင် စတင်ရောင်းလျှင် ဆယ်မိနစ်အတွင်း ဧည့်သည် ၄၀၀,၀၀၀ ဝင်လာသည်။ ငွေပေးချေမှုများသည် ထိုင်ခုံ တစ်ခုကို နှစ်ခါမရောင်းမီရ၊ ပြီးလျှင် ထိုင်းဖောက်သည်များ၏ ကိုယ်ရေးကိုယ်တာဒေတာသည် PDPA အောက်တွင် ရှိသည်။ **Fast, cheap, resilient** — ThaiTicket သည် သုံးခုလုံး လိုအပ်ပြီး သုံးခုလုံး မရနိုင် — ထို့ကြောင့်ပင် ၎င်းသည် အကောင်းဆုံး လေ့ကျင့်ရေးလူနာ ဖြစ်နေခြင်း ဖြစ်သည်။

Module 4 မတိုင်မီ ကြည့်ရန် (link များ စစ်ဆေးပြီး):

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
The Five Pillars of the AWS Well-Architected Framework	Amazon Web Services (official; ဆဋ္ဌမမဏ္ဍိုင် Sustainability ကို နောက်မှ ထည့်ခဲ့သည်)	~4 min	https://www.youtube.com/watch?v=KvEDbPmha6o

မီဒီယံ	ချန်နယ်	ကြာချိန်	လင့်ခ်
What is the AWS Well-Architected Framework?	Tech With Lucy	~10 min	https://www.youtube.com/watch?v=MpDJ6TCWKjk

Framework ကိုယ်တိုင်ကိုလည်း bookmark လုပ်ပါ — <https://aws.amazon.com/architecture/well-architected/> နှင့် စာတမ်းအပြည့်အစုံမှာ

<https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html> — ရှစ်ပတ်လုံး ၎င်းထဲတွင် နေထိုင်ရမည်။

Module 4 (အပတ်စဉ် ၇-၈): Reliability — အရာများပျက်သောနေ့အတွက် ဒီဇိုင်းဆွဲခြင်း

မဏ္ဍိုင်၏ ယုံကြည်ချက်: အရာအားလုံးသည် အချိန်တိုင်း ပျက်နေကြသည်။ Disk များ သေသည်။ AZ များ ရေကြီးသည်။ deploy များ မှားသည်။ ပြီးလျှင် တစ်နေရာရာက certificate တစ်ခုသည် သက်တမ်းကုန်တော့မည်သာ။ Reliability ဆိုသည်မှာ ကျရှုံးမှု မရှိခြင်း မဟုတ်ပါ။ ကျရှုံးမှု၏ အရေးမပါခြင်း ဖြစ်သည်။ အစိတ်အပိုင်းတစ်ခု သေသောအခါ (ဖြစ်လျှင်မဟုတ်၊ ဖြစ်သောအခါ) အသုံးပြုသူများ ဘယ်တော့မှ မသိအောင် ဒီဇိုင်းဆွဲခြင်း။

အဓိကမေးခွန်းများ (ဒီဇိုင်းတိုင်းကို ထာဝရ မေးရမည့်): အစိတ်အပိုင်းတစ်ခုချင်း ပျက်လျှင် ဘာဖြစ်မည်နည်း။ “ပြီးရင် ဘာလုပ်မလဲ” တစ်ခု အမြဲရှိရဲ့လား။ ဝန် ၁၀ ဆကို စနစ်က ဘယ်လိုကိုင်တွယ်မည်နည်း။ ဖောက်သည်များ မပြောခင် တစ်ခုခုပျက်ကြောင်း ဘယ်လို သိ မည်နည်း။ ကျွန်ုပ်တို့၏ RTO နှင့် RPO က ဘာလဲ။ ပြီးလျှင် လုပ်ငန်းထဲက ဘယ်သူက ၎င်းတို့ကို လက်မှတ်ထိုးအတည်ပြုခဲ့သနည်း။ Recovery တစ်ခုကို နောက်ဆုံး ဘယ်တုန်းက စမ်းသပ် ခဲ့သနည်း။

RTO နှင့် RPO — disaster စကားဝိုင်းဖြစ်သော ဂဏန်းနှစ်လုံး။ Recovery Time Objective: ဘယ်လောက်ကြာကြာ ရပ်နေခွင့်ရှိသနည်း။ Recovery Point Objective: ဒေတာ ဘယ်လောက် ဆုံးရှုံးခွင့်ရှိသနည်း (တစ်နည်း — နောက်ဆုံး backup ကောင်းသည် ဘယ်လောက် အိုဟောင်းခွင့်ရှိသနည်း)။ RTO ၄ နာရီနှင့် RPO ၁ နာရီ ဆိုသည်မှာ — လေးနာရီအတွင်း ပြန်တက်မည်။ အများဆုံး တစ်နာရီစာ ဒေတာ ဆုံးရှုံးလျက်။ ၎င်းတို့သည် exponential ဈေးနှုန်းတံဆိပ်ပါသော လုပ်ငန်း ဆုံးဖြတ်ချက်များ ဖြစ်သည်။ RPO ၂၄ နာရီ ဆိုသည်မှာ ညစဉ် backup (ဈေးပေါ့)၊ RPO ~သုည ဆိုသည်မှာ အဆက်မပြတ် replication (ဈေးကြီး)၊ RTO မိနစ်ပိုင်း ဆိုသည်မှာ standby ပေါ်တွင် အသင့်စောင့်နေသော warm infrastructure (အလွန်ဈေးကြီး)။ Architect ၏ အလုပ်မှာ လုပ်ငန်းကို ဂဏန်းများ သိသိသာသာ ရွေးချယ်စေရန် — ပြီးလျှင် ထိုဂဏန်းများအတိုင်း တိတိကျကျ ဒီဇိုင်းဆွဲရန်၊ ၎င်းတို့ကိုကျော်၍ စိတ်ကူးယဉ်ဆန်ဆန် မဟုတ်ရန်။

စံပုံစံများ: redundancy (အရေးကြီးသမျှ နှစ်ခုစီ — N+1) · multi-AZ (data center များအနှံ့ ပုံတူများ။ Load balancer နှင့် managed database failover က ၎င်းကို အလိုအလျောက်ဖြစ်စေသည်) · auto-scaling (စွမ်းရည်က ဝယ်လိုအားနောက် လိုက်သည်) · health check + self-healing (သေနေသော instance များကို လူသားမဟုတ်၊ စက်ယန္တရားက ရှာဖွေအစားထိုးသည်) · backup များ — စမ်းသပ်ပြီးသား (မစမ်းသပ်ရသေးသော backup သည် အစီအစဉ်မဟုတ်၊ မျှော်လင့်ချက်သာ — restore drill များကို အချိန်ဇယားဆွဲပါ) · graceful degradation (ဝန်ပိလျှင် အရေးအနည်းဆုံး feature များကို အရင်ချ — ThaiTicket သည် ထိုင်ခုံမြေပုံ preview များကို ချပြီး checkout ကို

ဆက်ထားနိုင်သည်) · **shock absorber** အဖြစ် **queue** များ (အဆင့် ၃) · **cascading failure** ရှောင်ခြင်း (timeout များ၊ **backoff** ပါသော **retry** များ၊ **circuit breaker** များ — နှေးနေသော **dependency** တစ်ခုက အုပ်စုတစ်ခုလုံးကို မနစ်မြုပ်စေရန်)။

လက်တွေ့မှုနာ — ThaiTicket reliability ဒီဇိုင်း။ ဘန်ကောက် **region** တွင် **AZ** နှစ်ခု။ **Load balancer** နောက်တွင် **auto-scaling group** ထဲက **stateless app tier** — ကြေညာထားသော ရောင်းချချိန်များမတိုင်မီ အချိန်ဇယားဖြင့် ကြိုနှေးထား (**auto-scaling** သည် မိနစ်ပိုင်းအတွင်း တုံ့ ပြန်သည်။ ၁၀:၀၀ ရောင်းချမှုအတွက် ၀၉:၄၅ တွင် စွမ်းရည် လိုသည်)။ **Aurora database** — **primary** ကို **AZ-a** တွင်၊ **synchronous standby** ကို **AZ-b** တွင်၊ အလိုအလျောက် **failover** ≈ တစ်မိနစ်အောက်။ “ဝယ်မည်” **click** များနှင့် ငွေပေးချေမှုလုပ်ငန်းစဉ်အကြား **queue** တစ်ခု — **payment-provider** နှေးလျှင် **site** မပျက်ဘဲ အော်ဒါများ တန်းစီစောင့်သည်။ **Backup** များ — အဆက်မပြတ်၊ **point-in-time restore**၊ လစဉ် **restore drill**။ လုပ်ငန်းနှင့် သဘောတူထားသောဂဏန်းများ — **RTO** ၁၅ မိနစ်၊ အော်ဒါများအတွက် **RPO** ~သုည (ပိုက်ဆံကိစ္စမို့)၊ **analytics** ဒေတာအတွက် **RPO** ၂၄ နာရီ (မဟုတ်သောကြောင့်)။ **Journal** လေ့ကျင့်ခန်း — ဤအရာက တြိဂံ၏ “cheap” ထောင့်မှ ဘာတွေ ကုန်ကျစေခဲ့သနည်း။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
RTO / RPO	Recovery Time Objective: ဘယ်လောက်ကြာကြာ ရပ်ခွင့်ရှိသနည်း။ Recovery Point Objective: ဒေတာဘယ်လောက် ဆုံးရှုံးခွင့်ရှိသနည်း။ DR စကားပိုင်းတိုင်းကို သတ်မှတ်ပေးသော ဂဏန်းနှစ်လုံး — ၎င်းတို့သည် လုပ်ငန်းဆုံးဖြတ်ချက်များ ဖြစ်သည်။
High availability (HA)	ပုံမှန်ကျရှုံးမှုများ (instance တစ်လုံး၊ AZ တစ်ခု) က အသုံးပြုသူမြင်သော ပြတ်တောက်မှု မဖြစ်စေအောင် ဒီဇိုင်းဆွဲခြင်း။
Disaster recovery (DR)	ကြီးမားသော ကျရှုံးမှုများ — region တစ်ခုလုံး၊ ransomware ဖြစ်ရပ် — အတွက် RTO/RPO ရည်မှန်းချက်များနှင့် စမ်းသပ်ပြီးသား runbook ပါသော အစီအစဉ်။
Redundancy / N+1	မည်သည့်အစိတ်အပိုင်းတစ်ခုမဆို သေနိုင်စေရန် အပိုစွမ်းရည် — N လိုအပ်၊ N+1 လည်ပတ်။
Failover	Primary သေလျှင် standby ဆီ အလိုအလျောက် ပြောင်းခြင်း။
Health check	သေနေသောအစိတ်အပိုင်းများကို ရှာဖွေပေးသဖြင့် စက်ယန္တရားက ရှောင်ကွင်းလမ်းကြောင်းချနိုင်စေသော အလိုအလျောက် နှလုံးခုန်စစ်ဆေးမှု။

ဝေါဟာရ	အဓိပ္ပာယ်
Graceful degradation	ဖိအားအောက်တွင် အရေးအနည်းဆုံး feature များကို ချထားခြင်းဖြင့် အဓိကအရာ ရှင်သန်စေခြင်း။
Timeout / retry with backoff	နူးနေသောခေါ်ဆိုမှုကို ကန့်သတ်ချိန်ကျော်လျှင် စွန့်လွှတ်ခြင်း / ကြိုးလာသောအငြိမ်ချိန်များဖြင့် ပြန်ကြိုးစားခြင်း — cascade failure များကို တားပေးသော ယဉ်ကျေးမှုများ။
Circuit breaker	ကျရှုံးနေသော dependency တစ်ခုကို ခဏရပ်၍ မခေါ်တော့ဘဲ ပြန်ကောင်းခွင့်ပေးသော အစိတ်အပိုင်း — အိမ်ထဲက ဓာတ်ကြိုးပြတ်ခလုတ် (fuse) ကဲ့သို့။
Chaos engineering	ခံနိုင်ရည်ကြွေးကြော်ချက်များကို လက်တွေ့ကမ္ဘာက မစမ်းသပ်ခင် ကိုယ်တိုင်သက်သေပြရန် ကျရှုံးမှုကို တမင်ထိုးသွင်းခြင်း (Netflix ပုံစံ)။

လေ့ကျင့်ခန်းများ: (၁) သင့်အဆင့် ၁ နေ့လယ်စာမှာယူရေး ဒီဇိုင်းကို ယူပြီး ဤအရာများကို ခံနိုင်အောင် အဆင့်မြှင့်ပါ — **instance** တစ်လုံးသေခြင်း၊ **AZ** တစ်ခုဆုံးရှုံးခြင်း၊ **database** ကျရှုံးခြင်း၊ နေ့လယ်စာအချိန် ၁၀ ဆတိုးခြင်း — မတိုင်မီနှင့် ပြီးနောက်ကို ဆွဲပါ။ (၂) လုပ်ငန်းသုံးခု (**blog** တစ်ခု၊ **e-commerce site** တစ်ခု၊ ဆေးရုံမှတ်တမ်းစနစ်တစ်ခု) အတွက် **RTO/RPO** စကားဝိုင်းကို **architect** နှင့် ပိုင်ရှင်အကြား စကားပြောဆိုမှုအတိုအဖြစ် ရေးပါ — စကားဝိုင်းထဲတွင် ကုန်ကျစရိတ်ကို မြင်သာအောင်ပြရန် လေ့ကျင့်ပါ။ (၃) **Journal** — အထက်ပါ **ThaiTicket** ဒီဇိုင်းထဲရှိ **SPOF** တိုင်းကို စာရင်းပြုစုပါ။ တမင်ချန်ထားသည့် တစ်ခု အနည်းဆုံးရှိသည်။ (အရိပ်အမြွက် — **region** ဘယ်နှစ်ခုလဲ။)

Milestone: မည်သည့် **diagram** ကိုမဆို “ဒါပျက်ရင် ဘာဖြစ်မလဲ” စစ်ဆေးမေးမြန်းမှုကို ဆယ်မိနစ်ဆက်တိုက် မကုန်ခန်းဘဲ လုပ်နိုင်ရမည်။ ပြီးလျှင် နည်းပညာမကျွမ်းကျင်သော ပိုင်ရှင်တစ်ဦးအား **RTO** နှင့် **RPO** ကို ဆိုင်ရေကြီးမှ ဥပမာဖြင့် ကိုးဆယ်စက္ကန့်အတွင်း ရှင်းပြနိုင်ရမည်။

Module 5 (အပတ်စဉ် ၉-၁၀): Security — နာကျင်စေရန်ခက်သော စနစ်များ ဒီဇိုင်းဆွဲခြင်း

ဘောင်ခတ်ချက်: shared responsibility model (တာဝန်ခွဲဝေမှုပုံစံ)။ **Provider** က **cloud** ကိုယ်တိုင် — အဆောက်အအုံများ၊ **hardware**၊ **hypervisor** — ကို လုံခြုံစေသည်။ သင်ကမူ ၎င်းထဲ သင်ထည့်သမျှအားလုံး — ဒေတာ၊ **identity** များ၊ **configuration** များ၊ **code** — ကို လုံခြုံစေရသည်။ နာမည်ကြီး **cloud** ပေါက်ကြားမှုနီးပါးတိုင်းသည် ဖောက်သည်ဘက်က မှားယွင်းသောပြင်ဆင်မှု (**public bucket** တစ်ခု၊ ပေါက်ကြားသော **key** တစ်ခု၊ ခွင့်ပြုချက်လွန်ကဲသော **role** တစ်ခု) ဖြစ်သည် — ထို့ကြောင့် လုံခြုံရေးသည် ကိရိယာပြဿနာ မဖြစ်ခင် **architecture** ပြဿနာ ဖြစ်နေခြင်း ဖြစ်သည်။ အလုပ်ခေါ်စာများက “integrate security into the design rather than bolting it on” ဟုဆိုပြီး ဤ **module** သည် ထိုနည်းလမ်းပင်။

အဓိကမေးခွန်းများ:: အစိတ်အပိုင်းတစ်ခုချင်းစီကို ဘယ်သူနှင့် ဘာက ဝင်ရောက်နိုင်ပြီး **permission** တိုင်းသည် လိုအပ်သော အနည်းဆုံး ဖြစ်ရဲ့လား။ ဒေတာကို ဘယ်နေရာတွင် ကုဒ်ဝှက်ထားသနည်း — **at rest, in transit**၊ ပြီးလျှင် သော့များကို ဘယ်သူကိုင်သနည်း။ **Network** နယ်နိမိတ်များက ဘယ်မှာလဲ၊ ဘာတွေက ၎င်းတို့ကို ဖြတ်ကျော်သနည်း။ ပေါက်ကြားမှုတစ်ခုကို ဘယ်လို ရှာဖွေတွေ့ရှိ မည်နည်း — ပြီးလျှင် နောက်ပိုင်း **log** များမှ ဇာတ်လမ်းကို ပြန်ပြောနိုင်မလား။ ဤဒေတာအပေါ် ဘယ်ဥပဒေ သက်ရောက်ပြီး ဒေတာက ရုပ်ပိုင်းအရ ဘယ်မှာ နေထိုင်သနည်း။

စံပုံစံများ:: **least privilege** (လူသားနှင့် **program** တိုင်း ၎င်း၏အလုပ်လိုအပ်သော အနည်းဆုံးဝင်ရောက်ခွင့်သာ ရသည် — **IAM policy** တိုင်းကို ဆုံးဖြတ်ရာ ရွှေ့စည်းကမ်း) · **နေရာတိုင်း MFA**၊ **root** ကို သော့ခတ်သိမ်း · **encryption at rest and in transit**၊ **အမြဲဖွင့်** (cloud တွင် **checkbox** တစ်ခုသာ။ ဆင်ခြေမရှိ) · **network segmentation** (public/private subnet များ၊ **server** တစ်လုံးချင်း **firewall** အဖြစ် **security group** များ။ ပေါက်ကြားမှုတစ်ခု၏ **blast radius** ကို သင်ကြိုတင်ဆွဲထားသော နံရံများက သတ်မှတ်သည်) · **secret များကို vault** ထဲ၊ **code** ထဲ ဘယ်တော့မှ မထား · **zero trust** (request တိုင်းကို အတိအလင်း စစ်ဆေးပါ — **identity, device, context** — “network အတွင်းမှာ ရှိရုံ” ဖြင့် ဘာကိုမှ မယုံပါနှင့်။ အလုပ်ခေါ်စာများက နာမည်တပ်သည်၊ သင်လည်း တပ်ရမည်) · **defense in depth** (အလွှာများ — ထိန်းချုပ်မှုတစ်ခု ကျရှုံးရုံဖြင့် ပွဲမပြီးစေရန်) · **audit logging** (ဘယ်သူ ဘာလုပ်ခဲ့သည်၏ **CloudTrail** ပုံစံ မှတ်တမ်းများ — ပြင်မရ၊ စောင့်ကြည့်ခံ) · **gate မဟုတ်၊ guardrail အဖြစ် security** (စည်းကမ်းများကို အလိုအလျောက် **policy** အဖြစ် ရေးသွင်းပြီး လုံခြုံသောလမ်းကြောင်းကို လွယ်ကူသောလမ်းကြောင်း ဖြစ်စေခြင်း — **security** ကို ပေးပို့မှု၏ရန်သူ ဖြစ်စေသော **review** အစည်းအဝေးမျိုး မဟုတ်)။

လက်တွေ့မှုနမူနာ — ThaiTicket security ဒီဇိုင်း။ ဖောက်သည်ဒေတာ (နာမည်များ၊ **email** များ၊ ငွေပေးချေမှု ကိုးကားချက်များ) ကို **PDPA** အောက် ကိုယ်ရေးကိုယ်တာဒေတာအဖြစ် သတ်မှတ် → ဘန်ကောက် **region** ရှိ **Aurora** ထဲတွင် ကုဒ်ဝှက်သိမ်း (**data residency**)၊ သော့များကို **KMS** ထဲ။ **Network** — **load balancer** တစ်ခုတည်းသာ **public**။ **App tier** က **private**။ **Database subnet** သည် **app tier** ၏ **security group** မှ ချိတ်ဆက်မှုများကို သာ လက်ခံသည်။ လူသားများ — **SSO + MFA**။ **Engineer** များသည် ပုံသေအားဖြင့် **production** ကို **read-only** ရပြီး တောင်းဆိုမှသာ အချိန်ကန့်သတ် အဆင့်မြှင့်ဝင်ရောက်ခွင့် ရသည် (**audit trail** ပါသော **least privilege**)။ ငွေပေးချေကတ်ကိုင်တွယ်မှုကို အသိအမှတ်ပြု **payment provider** တစ်ခုသို့ လွှဲအပ်ထားသဖြင့် ကတ်နံပါတ်အစစ်များ ကျွန်ုပ်တို့စနစ်များကို ဘယ်တော့မှ မထိ — **compliance** ဝန်ထုပ်တစ်ခုလုံးကို ဖယ်ရှားပေးသော ဒီဇိုင်း ဆုံးဖြတ်ချက် — ၎င်းသည် **security architecture** ၏ အကောင်းဆုံးပုံစံပင်။ **CloudTrail** ဖွင့်ထား၊ ပုံမှန်မဟုတ်သော ဝင်ရောက်မှုများအတွက် **alert** များ။ ပေါက်ကြားမှုမေးခွန်းကို ကြိုဖြေထား — **PDPA** တွင် ၇၂ နာရီ အကြောင်းကြားရန်တာဝန် ရှိသည် — **log** များနှင့် **runbook** က ထိုထက်တိုသောအချိန်အတွင်း ဇာတ်လမ်းပြောနိုင်ရမည်။

ဝေါဟာရများ::

ဝေါဟာရ	အဓိပ္ပာယ်
Shared responsibility model	Provider က cloud ကို လုံခြုံစေသည်။ ထဲရှိအရာများကို သင်က လုံခြုံစေရသည်။ ပေါက်ကြားမှုအများစုသည် မျဉ်း၏ ဖောက်သည်ဘက်တွင်။

ဝေါဟာရ	အဓိပ္ပာယ်
IAM / role / policy	Identity and Access Management — ဘယ်သူ ဘာ လုပ်ခွင့်ရှိသနည်း။ Policy သည် permission စာရင်း။ Role သည် ဝတ်ဆင်နိုင်သော ၎င်းတို့၏အစုအဝေး။
Least privilege	ရွှေစည်းကမ်း — လိုအပ်သော အနည်းဆုံးဝင်ရောက်ခွင့်၊ ထို့ထက်ပို မပေး၊ ပုံမှန် ပြန်စစ်။
MFA	Password အပြင် identity ၏ ဒုတိယသက်သေ။ လူသား များအတွက် ညှိနှိုင်း၍မရ။
Encryption at rest / in transit / KMS	Disk ပေါ်တွင် / ကြိုးပေါ်တွင် ကုဒ်ဝှက်ထားသောဒေတာ / သော့များကို ကိုင်ထားသော managed ဝန်ဆောင်မှု။
Security group	Server တစ်လုံးချင်း firewall စည်းကမ်းစာရင်း — “web traffic ကို load balancer ကနေသာ ဝင်ခွင့်”။
Network segmentation / blast radius	Network ကို နံရံခတ်ဇုန်များအဖြစ် ခွဲခြင်း / ပေါက်ကြားမှုတစ် ခုပြီးနောက် တိုက်ခိုက်သူ ဘယ်လောက်ဝေးဝေး ရောက်နိုင် သနည်း။ ကြိုတင်ဆွဲထားသောနံရံများက သတ်မှတ်သည်။
Zero trust	Request တိုင်းကို အတိအလင်း စစ်ဆေးပါ။ “အတွင်းမှာရှိရုံ” ဖြင့် ဘာကိုမှ မယုံပါနှင့်။ ခေတ်သစ် ပုံသေရပ်တည်ချက်။
Defense in depth	တစ်ခုကျရှုံးလည်း အသက်မသေစေရန် ထပ်နေသော ထိန်းချုပ် မှုအလွှာများစွာ။
Secrets management	Password များ၊ key များနှင့် token များသည် vault ဝန်ဆောင်မှုထဲ နေထိုင်သည် — code ထဲ ဘယ်တွေ့မှမနေ၊ spreadsheet ထဲလည်း ဘယ်တွေ့မှမနေ။
Audit log / CloudTrail	ဘယ်သူ ဘာကို ဘယ်တုန်းကလုပ်ခဲ့သည်၏ ပြင်မရသော မှတ်တမ်း — ဒုက္ခကို ရှာဖွေတွေ့ရှိပြီး နောက်ပိုင်း ပြန် တည်ဆောက်ပုံ။
Data classification	ဒေတာကို အထိခိုက်မခံနိုင်မှုအလိုက် တံဆိပ်တပ်ခြင်း (public / internal / personal / regulated) — ထိန်းချုပ်မှု များသည် ခန့်မှန်းချက်မဟုတ်၊ တံဆိပ်နှင့် ကိုက်ညီစေရန်။
Data residency	ဒေတာကို နိုင်ငံနယ်နိမိတ်အတွင်း ရုပ်ပိုင်းအရ ထိန်းသိမ်းထားခြင်း — လုပ်ငန်းနယ်ပယ်များစွာတွင် ဥပဒေ လိုအပ်ချက်၊ architecture input အမြဲတမ်း။

ဝေါဟာရ	အဓိပ္ပာယ်
PDPA / GDPR	ထိုင်းနှင့် ဥရောပ၏ ကိုယ်ရေးကိုယ်တာဒေတာ ဥပဒေများ — သဘောတူညီချက်၊ ပေါက်ကြားမှုအကြောင်းကြားခြင်း (၇၂ နာရီ)၊ နယ်စပ်ဖြတ်ကျော် လွှဲပြောင်းမှုစည်းကမ်းများ။ အဆင့် ၄ တွင် ပိုနက်နက်သွားမည်။

ဤ **module** အတွက် ဗီဒီယို (**link စစ်ဆေးပြီး**): The AWS Shared Responsibility Model — Digital Cloud Training — ~4 min — <https://www.youtube.com/watch?v=ESPBBEK-cvo>

လေ့ကျင့်ခန်းများ (၁) ThaiTicket ၏ diagram ကို ဆွဲပြီး ၎င်း၏ security ကို ထပ်ဆင့်ပြပါ — trust boundary တိုင်း၊ encryption point တိုင်း၊ credential နေထိုင်ရာနေရာတိုင်းကို မှတ်သားပါ။ ဤ overlay အလေ့အထ — diagram တစ်ခုတည်း၊ security မှန်ဘီလူး — သည် review တစ်ခုက သင့်ထံ တောင်းဆိုသည့်အရာ အတိအကျပင်။ (၂) Cloud မှားယွင်းပြင်ဆင်မှု ပေါက်ကြားမှုတစ်ခု၏ လူသိရှင်ကြား post-mortem တစ်ပုဒ် ဖတ်ပါ (Capital One 2019 သည် သင်ကြားရေး ဂန္ထဝင်) ပြီးလျှင် အထက်ပါ ပုံစံများထဲက ဘယ်ပုံစံ က ၎င်းကို တားနိုင်ခဲ့မည်ကို journal ထဲ ရေးပါ။ (၃) AI တစ်ခုနှင့် roleplay — ၎င်းက “security ကို နောက်မှထည့်မယ်” ဟုပြောသော startup founder အဖြစ် သရုပ်ဆောင်သည် — ပေါက်ကြားမှုကုန်ကျစရိတ် ဂဏန်းသိန်းဖြင့်၊ ကြင်နာစွာ၊ စိတ်ပြောင်းအောင် ပြောပါ။

Milestone: သင့်အဆင့် ၁ diagram တစ်ခုခုကို ယူပြီး ၎င်း၏ security overlay ကို မိနစ်နှစ်ဆယ်အတွင်း ထုတ်လုပ် နိုင်ရမည်။ ပြီးလျှင် least privilege၊ zero trust နှင့် shared responsibility model ကို နည်းပညာဝေါဟာရမပါဘဲ နည်းပညာမကျွမ်းကျင်သော stakeholder တစ်ဦးအား ရှင်းပြနိုင်ရမည်။

Module 6 (အပတ်စဉ် ၁၁-၁၂): Performance နှင့် cost — အမြန်နှုန်းနှင့် ငွေ၊ တွဲလျက် မဏ္ဍိုင်နှစ်ရပ်

ဤမဏ္ဍိုင်နှစ်ရပ်ကို အတူတွဲသင်ရသည့်အကြောင်းမှာ ၎င်းတို့သည် ဆန့်ကျင်ဘက်ဦးတည်ရာနှစ်ခုဆီ တွန်းထားသော ကိရိယာတံတစ်ချောင်းတည်း ဖြစ်ပြီး architect သည် ထိုကိရိယာတံကို လက်တင်ထားသောသူ ဖြစ်သောကြောင့်တည်း။

Performance efficiency — အဓိကမေးခွန်းများ အသုံးပြုသူများသည် နှေးကွေးမှုကို ဘယ်နေရာတွင် အရင်ခံစားရ မည်နည်း။ ထပ်ခါထပ်ခါ တွက်နေသော်လည်း တစ်ကြိမ်တည်းတွက်ပြီး cache လုပ်ထားနိုင်သည့်အရာက ဘာလဲ။ အစိတ်အပိုင်းတစ်ခုချင်းစီသည် မှန်ကန်သောကိရိယာ ဖြစ်ရဲ့လား (search engine အလုပ်ကို SQL လုပ်နေခြင်းသည် code ၏မဟုတ်၊ architecture ၏ performance bug ဖြစ်သည်။) ၁၀ ဆတွင် performance ဘယ်လိုပြောင်းမည် နည်း — ပထမဆုံး bottleneck က ဘယ်မှာလဲ။

Performance ပုံစံများ caching အလွှာများ — leverage အမြင့်ဆုံး performance ပုံစံတစ်ခုတည်း: browser cache → edge မှာ CDN (static content ကို အသုံးပြုသူအနီး မြို့တစ်မြို့မှ ဝန်ဆောင်ပေးသည်။ ဖတ်ရှုမှု traffic အများစုကြီးကို စုပ်ယူနိုင်သည်) → application cache (hot ဖတ်ရှုမှုများအတွက် database ရှေ့တွင် Redis — ထိုင်ခုံမြေပုံများ၊ session များ၊ ကုန်ပစ္စည်းစာမျက်နှာများ) → database query cache။ အလွှာတစ်ခုစီက request

များကို ဈေးကြီးသောအူတိုင် မရောက်ခင် ဖြေပေးသည်။ ဒီဇိုင်းမေးခွန်းများမှာ အမြဲတမ်း — ဘာကို **cache** လုပ်ခွင့်ရှိသနည်း၊ ဘယ်လောက်ကြာကြာ၊ ပြီးလျှင် အမှန်တရားပြောင်းသောအခါ ဘယ်လို **invalidate** လုပ်မည်နည်း။ ထို့နောက် — **read replica** များ (ဖတ်ရှုမှုများကို ဝန်ဆောင်ပေးသော database မိတ္တူများ — **primary** က ရေးသားမှုကို ဆက်လုပ်နိုင်စေရန်) · **asynchronous processing** (နောက်မှလုပ်လို့ရသောအလုပ်အတွက် အသုံးပြုသူကို မစောင့်ခိုင်းပါနှင့် — email ပြေစာများ၊ thumbnail များ၊ report များ) · **အလုပ်တစ်ခုလျှင် မှန်ကန်သောကိရိယာ** (search → search engine။ Analytics → warehouse။ Hot lookup များ → key-value store) · ပြီးလျှင် **အရင်တိုင်းတာပါ** (တိုင်းတာမှုပါသော performance အလုပ်သည် အယူသီးမှုသာ)။

Cost optimization — အဓိကမေးခွန်းများ: ဤဒီဇိုင်းသည် ယနေ့ဝန်တွင် တစ်လ ဘယ်လောက်ကုန်သနည်း — ပြီးလျှင် ယူနစ် တစ်ခုလျှင် (အော်ဒါတစ်ခုလျှင်၊ ဖောက်သည်တစ်ဦးလျှင်)။ နံနက် ၃ နာရီတွင် မလိုအပ်ဘဲ ဘာလည်နေသနည်း။ ဘယ်တည်ငြိမ်သောသုံးစွဲမှုကို လျှော့ဈေးဖြင့် ကတိပြုနိုင်မည်နည်း။ **Finance** က လမ်းကြောင်း (trend) ကို ဘာပြောမည်နည်း။

Cost ပုံစံများ: rightsizing (အုပ်စုအများစုသည် တိတ်တဆိတ် အရွယ်ကြီးလွန်းနေသည်။ တိုင်းတာထားသော လိုအပ်ချက်အထိ ချုံ့ခြင်းသည် အခမဲ့ရသောပိုက်ဆံ) · **reservation နည်းဗျူဟာ** — ဈေးနှုန်း menu: **on-demand** (ဈေးအပြည့်၊ လွတ်လပ်မှုအပြည့်) — ရုတ်တရက်တက်/မသိသောဝန်အတွက်၊ **reserved instance / savings plan များ** (၁-၃ နှစ် ကတိပြု၊ ၃၀-၇၀% လျှော့) — တည်ငြိမ်သော **baseline** အတွက်၊ **spot** (၉၀% အထိလျှော့၊ အသိပေးချိန်တိုဖြင့် ပြန်သိမ်းခံရနိုင်) — ကြားဖြတ်ခံနိုင်သော **batch** အလုပ်အတွက်။ **Architect** ၏ လှုပ်ရှားချက်မှာ ၎င်းတို့ကို အလွှာလိုက်စီခြင်း — ကြမ်းခင်းကို **reserve**၊ အလယ်ကို **on-demand** ဖြင့် **auto-scale**၊ **batch** ကို **spot** · **storage lifecycle** (Module 2 ၏ tier ခွဲခြင်း — အလိုအလျောက်) · **ပိတ်ထားပါ** (ညများနှင့် စနေတနင်္ဂနွေများတွင် အိပ်နေသော non-production environment များသည် ၎င်းတို့၏ကုန်ကျစရိတ်ကို သုံးပုံနှစ်ပုံ လျှော့နိုင်သည်) · **egress ကို စောင့်ကြည့်ပါ** (cloud မှ ထွက်သော ဒေတာကို ငွေကောက်သည်။ စကားပြောများသော **cross-region** ဒီဇိုင်းများနှင့် ကြီးမားသော **public download** များသည် တစ်ခါတော့ လူတိုင်းကို အံ့အားသင့်စေသည်) · **tagging နှင့် showback** (resource တိုင်းကို owner/project ဖြင့် တံဆိပ်တပ်ထားသဖြင့် ဘတ်တိုင်း တာဝန်ခံနိုင်သည်) · ပြီးလျှင် သရဖူတိုင်းတာချက် — **unit economics**: “ဘေလ်က တစ်လ \$800k” မဟုတ်ဘဲ “ရောင်းရသောလက်မှတ်တစ်စောင်လျှင် ကုန်ကျစရိတ်က \$1.90 ဖြစ်ပြီး ကျဆင်းနေသည်”။ ကြီးလာသောဘေလ်များက ပြဿနာမဟုတ်ပါ။ ကြီးလာသော ယူနစ်ကုန်ကျစရိတ်များကမူ **architecture** အနံ့အသက်တစ်ခု။ ဤပညာရပ်တွင် နာမည်ရှိသည် — **FinOps** — ပြီးလျှင် **architect** များသည် ၎င်း၏အလယ်တွင် ထိုင်ကြသည်။

လက်တွေ့မှုနာမူ — ThaiTicket ကို မှန်ဘီလူးနှစ်ခုလုံးဖြင့်။ Performance — CloudFront က အနုပညာရှင်စာမျက်နှာများနှင့် ထိုင်ခုံမြေပုံများကို ဝန်ဆောင်ပေးသည် (ကြည့်ရှုသူ ၄၀၀,၀၀၀ အများစုသည် **server** များကို ဘယ်တော့မှ မထိ)။ **Redis** က ထိုင်ခုံရရှိနိုင်မှုကို ၂ စက္ကန့် **TTL** ဖြင့် **cache** လုပ်သည် — ဈေးသက်သာလောက်အောင် ဟောင်းပြီး **checkout** (အမှန်တရား၏ဇာစ်မြစ် **Aurora** နှင့် ထပ်စစ်သော) က နှစ်ခါရောင်းခြင်းကို တားနိုင်လောက်အောင် လတ်ဆတ်သည်။ ပြေစာများနှင့် လက်မှတ်များကို ငွေပေးချေပြီးနောက် **asynchronous** ထုတ်လုပ်သည်။ **Cost** — တည်ငြိမ်သော တစ်နာရီ ဧည့်သည် ၂,၀၀၀ **baseline** သည် **savings plan** ပေါ်တွင် လည်သည် (≈40% လျှော့)။ ရောင်းချချိန် တက်ခေတ်များသည် ၎င်းတို့တည်ရှိသော တစ်နာရီအတွက် **on-demand** ဖြင့်။ **Analytics** အလုပ်များသည် ညအချိန် **spot** ပေါ်တွင်။ **Staging** သည် ရုံးချိန်ပြင်ပ အိပ်သည်။ **Resource** တိုင်းကို **project:thaiticket** ဖြင့် **tag** တပ်ထား။ **CFO** ထံ **proposal** က ဤသို့ဖတ်ရသည် — “တစ်လ **baseline** \$62,000၊ အဓိကရောင်းချပွဲတစ်ခုလျှင် ≈\$9,000၊ လက်မှတ်တစ်

စောင်လျှင် ကုန်ကျစရိတ် $\approx$ **1.90** — ပမာဏတိုးလျှင် ကျဆင်း” — ပြီးလျှင် ထိုစာကြောင်း သည်ပင် **cost** ကျွမ်းကျင်သော **architect** တစ်ဦး၏ အသံပုံစံ ဖြစ်သည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Latency / throughput	Request တစ်ခု ဘယ်လောက်ကြာသနည်း / စနစ်က တစ်စက္ကန့်လျှင် request ဘယ်နှစ်ခု ခံနိုင်သနည်း။ ဆက်စပ်သော်လည်း မတူပါ။
Bottleneck	စနစ်တစ်ခုလုံး၏ အရှိန်ကို သတ်မှတ်ပေးသော အကျဉ်းဆုံးနေရာ။ အခြားနေရာတွင် optimize လုပ်ခြင်းသည် အလှဆင်ရုံသာ။
Cache / TTL / invalidation	ရယူရန်ဈေးကြီးသောဒေတာ၏ အမြန်မိတ္တူ / ၎င်းကို ဘယ်လောက်ကြာကြာ ယုံနိုင်သနည်း / အမှန်တရားပြောင်းသောအခါ ၎င်းကို ပြန်လတ်ဆတ်စေရေး ပြဿနာခက်။
CDN	အပြင်ဆုံး cache — မြို့ရာနှင့်ချီရှိ သင့် content ။ “ကမ္ဘာအနှံ့ မြန်အောင်လုပ်ပါ” ၏ ပထမအဖြေ။
Read replica	ဖတ်ရှုမှုများကို ဝန်ဆောင်ပေးသော database မိတ္တူ — primary ၏ ခွန်အားကို ရေးသားမှုအတွက် ချန်ထားစေရန်။
Asynchronous processing	အရေးမကြီးသောအလုပ်ကို အသုံးပြုသူအား ပြန်ဖြေပြီးမှ လုပ်ခြင်း — များသောအားဖြင့် queue မှတစ်ဆင့်။
On-demand / reserved / savings plan / spot	ဈေးအပြည့် ပြောင်းလွယ် / ၁-၃ နှစ်ကတိ ၃၀-၇၀% လျှော့ / အယူအဆတူ ပိုပြောင်းလွယ် / ၉၀% အထိလျှော့ သို့သော် ပြန်သိမ်းခံရနိုင်။ Architect က လေးမျိုးလုံးကို အလွှာလိုက် စီသည်။
Rightsizing	အရွယ်ကြီးလွန်းသော resource များကို တိုင်းတာထားသော လိုအပ်ချက်အထိ ချုံ့ခြင်း။ Account နီးပါးတိုင်းတွင် အခမဲ့ရသောပိုက်ဆံ။
Egress	Cloud မှ ထွက်သောဒေတာ — GB အလိုက် ငွေကောက်။ နာမည်ကြီး ဘောလ်အံ့အားသင့်မှု။ ဒေတာစီးကြောင်းများကို ၎င်းကို ထည့်တွက်၍ ဒီဇိုင်းဆွဲပါ။
Tagging / showback	Resource တိုင်းကို owner နှင့် project ဖြင့် တံဆိပ်တပ်ခြင်း / အဖွဲ့တစ်ဖွဲ့ချင်းစီအား ၎င်း၏ကိုယ်ပိုင်ဘောလ်ကို ပြခြင်း။

ဝေါဟာရ	အဓိပ္ပာယ်
	တာဝန်ခံမှုက အပြုအမူကို ပြောင်းစေသည်။
Unit economics	လုပ်ငန်းယူနစ်တစ်ခုလျှင် ကုန်ကျစရိတ် (အော်ဒါတစ်ခုလျှင်၊ user တစ်ဦးလျှင်)။ ဘေလ်တစ်ခုကို အဓိပ္ပာယ်ရှိစေသော တိုင်းတာချက် — CFO ရှေ့မှောက်တွင် architect ၏ အကြိုက်ဆုံးစာကြောင်း။
TCO	Total cost of ownership — ကပ်ထားသောဈေးအပြင် လူ၊ license များ၊ migration နှင့် ထွက်ခွာစရိတ်များ — ရွေးချယ်မှုတစ်ခု၏ သက်တမ်းတစ်လျှောက်လုံး။
FinOps	Cloud သုံးစွဲမှုကို မြင်သာအောင်၊ ခွဲဝေတာဝန်ခံအောင်၊ အဆက်မပြတ် ပိုကောင်းအောင်လုပ်သော ပညာရပ်။

ဤ module အတွက် ဗီဒီယို (link စစ်ဆေးပြီး): What is FinOps? — FinOps Foundation (official) — ~2 min — https://www.youtube.com/watch?v=Y-c_xw9bHFw

လေ့ကျင့်ခန်းများ: (၁) ThaiTicket ၏ baseline ကို AWS pricing calculator ထဲတွင် ဈေးတွက်ပါ (“AWS pricing calculator” ဟု ရှာပါ — ဒီဇိုင်းအစစ်တစ်ခုကို တစ်ခုချင်းစာရင်းလုပ်သော လေ့ကျင့်ခန်းသည် နောင်လာမည့် ကုန်ကျစရိတ် စကားဝိုင်းတိုင်း၏ လျှို့ဝှက်ဆန်းကြယ်မှုကို ဖယ်ရှားပေးသည်။ သင့်စုစုပေါင်းကို အထက်ပါ \$62,000 နှင့် နှိုင်းယှဉ်ပြီး ကွာဟချက်ကို ရှင်းပြပါ။ (၂) သင့်အဆင့် ၁ diagram နှစ်ခုတွင် caching overlay ထည့်ပါ — cache တိုင်း၊ ၎င်း၏ TTL နှင့် ၎င်း၏ invalidation ဇာတ်လမ်းကို မှတ်သားပါ။ (၃) AI တစ်ခုက “ရိုးရှင်းအောင်” အားလုံးကို on-demand ထားလိုသော engineer အဖြစ် သရုပ်ဆောင်သည် — reservation နည်းဗျူဟာကို ဂဏန်းများဖြင့် ညှိနှိုင်းပါ။

Milestone: သင်ဆွဲထားသော မည်သည့်ဒီဇိုင်းအတွက်မဆို ၎င်း၏ အကြီးဆုံး ကုန်ကျစရိတ်လိုင်းသုံးခုနှင့် ဖြစ်နိုင်ခြေရှိသော ပထမ bottleneck ကို ပြောနိုင်ရမည် — ပြီးလျှင် နှစ်ခုလုံးကို တစ်ပြိုင်နက် တိုးတက်စေမည့် ပြောင်းလဲမှုတစ်ခု အဆိုပြုနိုင်ရမည် (အမြဲလိုလို တစ်ခုရှိတတ်သည်။ များသောအားဖြင့် caching ပင်)။

Module 7 (အပတ်စဉ် ၁၃–၁၄): Operational excellence၊ sustainability — နှင့် မဏ္ဍိုင်ခြောက်ရပ်လုံးကို အတူလည်ပတ်ခြင်း

Operational excellence သည် ဤသို့မေးသော မဏ္ဍိုင်ဖြစ်သည် — လူသားများက ဒီအရာကို တည်တည်ငြိမ်ငြိမ် အမှန်တကယ် လည်ပတ်နိုင်ရဲ့လား။ အဓိကမေးခွန်းများ — ပြောင်းလဲမှုတစ်ခုသည် production ကို ဘယ်လိုရောက်သနည်း — အလိုအလျောက်၊ စမ်းသပ်ပြီးသား pipeline မှလား၊ သူရဲကောင်းဆန်သော လူသားတစ်ဦးမှလား။ စနစ်သည် ယခုအချိန် ကျန်းမာကြောင်း ဘယ်လိုသိသနည်း။ နံနက် ၃ နာရီတွင် ပျက်လျှင် on-call engineer က တကယ် ဘာလုပ်သနည်း။ ပုံစံများ — **infrastructure as code** (environment ကို review လုပ်ထား၊ version ထိန်းထားသော စာသားဖြင့် သတ်မှတ်ခြင်း — Terraform/CloudFormation — ထို့ကြောင့် ထပ်ခါတည်ဆောက်နိုင်ပြီး စစ်ဆေးနိုင်သည်။ Console

click များအဖြစ်သာ တည်ရှိသော architecture သည် engineering မဟုတ်၊ ရိုးရာပုံပြင်သာ) · လုံခြုံသော deployment နည်းဗျူဟာများပါ **CI/CD pipeline** များ (blue-green: ဗားရှင်းအသစ်ကို အဟောင်းဘေးတွင် ထောင်ပြီး ပြောင်းလိုက်ခြင်း။ **canary**: ဗားရှင်းအသစ်ကို traffic ၏ 5% ပေးပြီး စောင့်ကြည့်ခြင်း) · **observability** (log များ၊ metric များ၊ trace များနှင့် — စက်၏အသေးအမွှားမဟုတ်ဘဲ အသုံးပြုသူမြင်သော လက္ခဏာများနှင့် ချိတ်ထားသော alert များ) · **runbook** များ (သိပြီးသားကျရှုံးမှုတစ်ခုချင်းအတွက် ရေးထားသော ဘာလုပ်ရမည်နည်း) · **blameless post-mortem** များ (incident များပြီးနောက် — ဘာကျရှုံးခဲ့လဲ၊ ဘာကြောင့်လဲ၊ ထပ်မဖြစ်အောင် ဘာကတားမလဲ — လူဆိုးမရှိ၊ မဟုတ်လျှင် လူများသည် အမှန်တရားပြောခြင်း ရပ်သွားမည်)။ **Architect** ၏ အခန်းကဏ္ဍ — လည်ပတ်နိုင်မှုအတွက် ဒီဇိုင်းဆွဲပါ — နှစ်ဆပိုတော်ပြီး တစ်ဝက်သာ **observable** ဖြစ်သောစနစ်သည် ပိုဆိုးသောစနစ် ဖြစ်သည်။

ဆဋ္ဌမမဏ္ဍိုင် **sustainability** က ရုပ်ပိုင်းကမ္ဘာကို လျော့ဖြုန်းရန် တောင်းဆိုသည် — **rightsize** လုပ်ပါ (အလကားနေ CPU များသည် လျှပ်စစ်အစစ်ကို လောင်ကျွမ်းသည်)။ အမြင့်ဆုံးအတွက် ကြိုပြင်ဆင်မည့်အစား ဝယ်လိုအားအလိုက် **scale** လုပ်ပါ။ **managed** နှင့် **serverless** ဝန်ဆောင်မှုများကို သုံးပါ (မျှဝေထားသော **infrastructure** သည် ပိုပြည့်သော **infrastructure**)၊ **storage** ကို **tier** ခွဲပါ။ သေပြီးသားကို ဖျက်ပါ။ အဆင်ပြေစွာပင် — **sustainable** ရွေးချယ်မှုနှင့် **cheap** ရွေးချယ်မှုသည် များသောအားဖြင့် တစ်ခုတည်း ဖြစ်နေတတ်သည် — **review** များတွင် နှစ်ခုလုံးပြောလျှင် အခန်းကို နှစ်ကြိမ်အနိုင်ရမည်။

မဏ္ဍိုင်များကို အတူလည်ပတ်ခြင်း — သင့်ပထမဆုံး **Well-Architected review**။ **Review** အစစ်များတွင် မဏ္ဍိုင်များသည် ပဋိပက္ခဖြစ်ကြသည် — **multi-region reliability** သည် **cost** နှင့် ရန်ဖြစ်သည်။ တင်းကျပ်သော **security review** သည် **operational** အရှိန်နှင့် ရန်ဖြစ်သည်။ **Performance caching** သည် ဒေတာလတ်ဆတ်မှု မှန်ကန်ရေးနှင့် ရန်ဖြစ်သည်။ **Framework** က ပဋိပက္ခများကို သင့်အတွက် မဖြေရှင်းပေးပါ — ၎င်းတို့ကို လင်းလင်းချင်းချင်း ပေါ်လာအောင် ဖိအားပေးပြီး လုပ်ငန်းက ရွေးချယ်နိုင်စေသည်။ ၎င်းသည်ပင် **framework** ၏ ပါရမီတစ်ခုလုံးဖြစ်ပြီး သင်ကိုင်တွယ်ရမည့်အရာလည်း ဖြစ်သည်။ **Review** နည်းလမ်းကိုယ်တိုင် (မေးခွန်းအစုံများ၊ တွေ့ရှိချက်များကို ဦးစားပေးစီခြင်း၊ အစီရင်ခံစာ) ကို အဆင့် ၅၊ **Module 12** တွင် အပြည့်အစုံ သင်ကြားသည်။ ဤနှစ်ပတ်တွင် ကျွမ်းကျင်မှုအကြမ်းကို ဇာတ်တိုက်သည် — ဒီဇိုင်းတစ်ခုတည်းကို ဦးတည်ရာခြောက်ခုမှ တစ်ထိုင်တည်း စစ်ဆေးမေးမြန်းခြင်း။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Infrastructure as Code (IaC) / Terraform	Infrastructure ကို version ထိန်းထားသော စာသားဖိုင်များဖြင့် ကြေညာပြီး ကိရိယာများက လက်တွေ့ဖြစ်စေခြင်း / အရေပန်းအစားဆုံး ထိုကိရိယာ။
CI/CD pipeline	ပြောင်းလဲမှုတိုင်းကို တည်ဆောက်၊ စမ်းသပ်၊ deploy လုပ်ပေးသော အလိုအလျောက် စက်ခါးပတ်လမ်း။
Blue-green / canary deployment	လုံခြုံသော release ပုံစံနှစ်မျိုး — အဟောင်းနှင့်အသစ် မိတ္တူ၊ နှစ်ခုအကြား traffic ပြောင်းခြင်း / အသစ်ဆီ traffic အနည်းငယ်စီးစေပြီး စောင့်ကြည့်ခြင်း။

ဝေါဟာရ	အဓိပ္ပာယ်
Observability (log များ၊ metric များ၊ trace များ)	စနစ်၏ မိမိကိုယ်ကို ရှင်းပြနိုင်စွမ်း — ဖြစ်ရပ်မှတ်တမ်းများ၊ အချိန်နှင့်အမျှဂဏန်းများ၊ request တစ်ခုချင်း ခရီးစဉ်မြေပုံများ။
Alert / on-call / runbook	သတ်မှတ်ချက်ကျော်လျှင် အလိုအလျောက် အသိပေးချက် / ၎င်းကို ဖြေကြားသော လူသားအလှည့်ကျစနစ် / သူတို့လိုက်နာသော ရေးထားသည့် script။
Blameless post-mortem	Incident တစ်ခုပြီးနောက် ရိုးသား၊ လူဆိုးမရှိသော ပြန်လည်သုံးသပ်မှု — အပြစ်ပေးမှုမဟုတ်၊ ကာကွယ်မှုကို ထုတ်လုပ်သည်။
Drift	တစ်ယောက်ယောက် click နှိပ်မိသောကြောင့် လက်တွေ့ကမ္ဘာသည် IaC သတ်မှတ်ချက်မှ သွေဖည်သွားခြင်း။ ထပ်ခါတည်ဆောက်နိုင်မှု၏ ရန်သူ။
Toil	Automation က စုပ်ယူထားသင့်ပြီဖြစ်သော ထပ်ခါလုပ်နေရသည့် လက်လုပ်လည်ပတ်ရေးအလုပ်။ Architect များက ဒီဇိုင်းဖြင့် ဖယ်ထုတ်သည်။
Well-Architected review	Workload တစ်ခုကို မဏ္ဍိုင်ခြောက်ရပ်လုံးနှင့်ယှဉ်၍ စနစ်တကျစစ်ဆေးမေးမြန်းပြီး ဦးစားပေးစီထားသောတွေ့ရှိချက်များ ထုတ်လုပ်ခြင်း။ အဆင့် ၅ တွင် ကျင်းပနည်းကို သင်ပေးမည်။

လေ့ကျင့်ခန်းများ: (၁) မှန်ဘီလူးခြောက်ခု လေ့ကျင့်ခန်း — သင့်အကောင်းဆုံး diagram ကို ယူပြီး မဏ္ဍိုင်တစ်ခုလျှင် ဆယ် မိနစ်စီ တွေ့ရှိချက်များ ရေးပါ — မိနစ်ခြောက်ဆယ်၊ ဒီဇိုင်းတစ်ခု၊ မှန်ဘီလူးခြောက်ခု။ ဤလေ့ကျင့်ခန်းသည်ပင် အဆင့် ၂ ကို အသေးချဲ့ထားခြင်း — ယခုမှစ၍ အပတ်စဉ် ထပ်လုပ်ပါ။ (၂) AWS ၏ incident ပြီးနောက် အစီရင်ခံစာနှစ်စောင် (ကြီးမားသော outage များပြီးနောက် ၎င်းတို့၏ status page များတွင် ထုတ်ဝေသည်) ကို ဖတ်ပြီး ဘယ်မဏ္ဍိုင်၏ပုံစံ ကျ ရှုံးခဲ့သည်ကို ဖော်ထုတ်ပါ။ (၃) Journal — ThaiTicket အတွက် လုပ်ငန်းထံ တင်ပြမည့် မဏ္ဍိုင်ပဋိပက္ခသုံးခုကို တစ်ခုစီ စာကြောင်းတစ်ကြောင်း ရွေးချယ်မှုအဖြစ် ရေးပါ (“ဒီဘတ်ဂျက်နဲ့ X ဒါမှမဟုတ် Y ရနိုင်တယ် — ဘယ်ဟာလဲ။”)

Milestone — အဆင့် ၂ အဆုံး: အတုပြု review — AI တစ်ခုက ထိုင်း အွန်လိုင်းချေးငွေ startup တစ်ခုအတွက် ချို့ယွင်းချက်ရှိသော architecture တစ်ခု ထုတ်ပေးသည်။ သင်သည် မဏ္ဍိုင်ခြောက်ရပ်လုံးအနွဲ့ ရေးသားထားသော တွေ့ရှိချက်များ ထုတ်လုပ်ရမည် — အနည်းဆုံး: reliability ကွာဟချက်တစ်ခု (သူတို့၏ single-AZ database)၊ security ကွာဟချက်တစ်ခု (ကျယ်လွန်းသော IAM)၊ cost ကွာဟချက်တစ်ခု (အားလုံး on-demand)၊ operational ကွာဟချက်တစ်ခု (IaC မရှိ)၊ နှင့် သူတို့ ဘယ်တော့မှ မလုပ်ခဲ့သော RTO/RPO စကားဝိုင်း — တွေ့ရှိချက်တစ်ခုချင်းစီကို

လုပ်ဖော်ကိုင်ဖက်တစ်ဦး မတုန်လှုပ်ဘဲ နားထောင်နိုင်သော မေးခွန်းအဖြစ် ရေးသားထားရမည်။ သင့်တွေ့ရှိချက်စာရင်းသည် စာရင်းစစ်တစ်ဦးလိုမဟုတ်ဘဲ ကူညီတတ်သော **senior** လုပ်ဖော်ကိုင်ဖက်တစ်ဦးလို ဖတ်ရသောအခါ အဆင့် ၂ ပြီးပြီ။

အဆင့် ၃ — ပုံစံစာရင်း (PATTERN CATALOG) (အပတ်စဉ် ၁၅-၂၀)

Architect များသည် တီထွင်ခြင်း မပြုကြပါ။ **ရွေးချယ်** ကြသည်။ ဤအဆင့်သည် စံ **architecture** များ၏ သင့်ကိုယ်ပိုင် စာရင်း ဖြစ်သည် — တစ်ခုချင်းစီအတွက်: ၎င်းက ဘာလဲ၊ ဘယ်အခါသုံးရမလဲ၊ ဘယ်အခါ မသုံးရမလဲ၊ ပြီးလျှင် ၎င်း၏ **cost/complexity** ပရိုဖိုင်။ “ဘယ်အခါ မသုံးရ” ကော်လံများသည်ပင် ဤအဆင့်၏ ဝန်စည်အစစ် ဖြစ်သည် — **microservices** ဆိုတာဘာလဲကို မည်သည့်သင်တန်းမဆို ပြောနိုင်သည်။ ၎င်းတို့က ကုမ္ပဏီတစ်ခုကို ဘယ်အချိန် ဖျက်ဆီးမည်ကို သိခြင်းကမူ သင်လခရသည့်အရာ ဖြစ်သည်။ လေ့လာနေစဉ် <https://aws.amazon.com/architecture/> တွင် **reference architecture** အစစ်များကို ဆက်လှန်လျော့ကြည့်ပါ — တောရိုင်းထဲ ပုံစံရှာဖွေခြင်းသည်ပင် စာရင်းကို စွဲမြဲစေသော လေ့ကျင့်ခန်း ဖြစ်သည်။

Module 8 (အပတ်စဉ် ၁၅-၁၇): Application ပုံစံများ

Monolith နှင့် microservices — လုပ်ငန်းနယ်ပယ်၏ အဆူညံဆုံးငြင်းခုံမှုကို တည်တည်ငြိမ်ငြိမ် ဖြေရှင်းခြင်း။

Monolith ဆိုသည်မှာ **feature** အားလုံးပါဝင်သော **deploy** လုပ်နိုင်သည့် **application** တစ်ခုတည်း ဖြစ်သည်။

Microservices ကမူ စနစ်ကို သီးခြားစီ **deploy** လုပ်သော ဝန်ဆောင်မှုငယ်များစွာအဖြစ် ခွဲထုတ်ပြီး တစ်ခုစီက ၎င်း၏ဒေတာကို ပိုင်ဆိုင်ကာ **API** များနှင့် **event** များမှတစ်ဆင့် စကားပြောကြသည်။ ရိုးသားသောဇယား —

ပုံစံ	ဘာလဲ	ဘယ်အခါသုံး	ဘယ်အခါ မသုံးရ	Cost/complexity
Monolith	App တစ်ခု၊ deployment တစ်ခု၊ database တစ်ခု	အဖွဲ့ငယ်၊ ထုတ်ကုန် နယ်၊ domain နယ်နိမိတ်များ မ ရှင်းလင်းသေး — တစ်နည်း စနစ်အသစ် အများစု	အဖွဲ့များစွာ တစ်ဖွဲ့ကို တစ်ဖွဲ့ စောင့်နေရ ခြင်း။ အစိတ်အပိုင်း များ အလွန်ကွဲပြားစွာ scale လုပ်ရန်လိုခြင်း	ရှုပ်ထွေးမှုနည်း၊ ဈေး သက်သာ။ ခေတ် ရေစီးကြောင်း ဝန်ခံ သည်ထက် ပိုဝေးဝေး scale ရသည် — “ငြီးငွေ့ဖွယ်” သည် feature တစ်ခု
Microservices	သီးခြားစီ တည်ဆောက် deploy scale လုပ်	သီးခြား release အရှိန်လိုသော အဖွဲ့ များစွာ။ အစိတ်အပိုင်း	အဖွဲ့ငယ် ("distributed monolith ဆိုတာ	ရှုပ်ထွေးမှုမြင့် — function ခေါ်ဆိုမှု တိုင်း ကျရှုံးနိုင်သော

ပုံစံ	ဘာလဲ	ဘယ်အခါသုံး	ဘယ်အခါ မသုံးရ	Cost/complexity
	သော ဝန်ဆောင်မှုငယ်များစွာ	အလိုက် အလွန်ကွဲပြားသော scaling။ သက်သေပြပြီး တည်ငြိမ်သော domain နယ်နိမိတ်များ	network ကျရှုံးမှုများ ထပ်ထည့်ထားတဲ့ monolith ပဲ”)။ Domain က ရွေးနေဆဲ	network ခေါ်ဆိုမှု ဖြစ်သွားသည်။ ရင့်ကျက်သော CI/CD၊ observability၊ on-call လိုသည်

Architect ၏ ဆုံးဖြတ်ချက် — အတွင်း **module** ခွဲထားသော **monolith** ဖြင့် စတင်ပါ။ **Team-scaling** သို့မဟုတ် **load-scaling** နာကျင်မှု အမှန်တကယ် ရောက်လာမှသာ — ရောက်လာမှသာလျှင် — **service** များကို ခွဲထုတ်ပါ။ ဤစကားကို အင်တာဗျူးများတွင် အကြောင်းပြချက်များနှင့် ပြောခြင်းက သင့်ကို **senior** အဖြစ် မှတ်သားစေသည်။ ဘယ်ဘက်ကိုမဆို အယူဝါဒဆန်ခြင်းက **junior** အဖြစ် မှတ်သားစေသည်။

Event-driven architecture — **queue** များနှင့် **topic** များ၊ **shock absorber** များ။ အစိတ်အပိုင်းများ တစ်ခုကို တစ်ခု ခေါ်ပြီး စောင့် နေမည့်အစား (**synchronous**)၊ အစိတ်အပိုင်းများက **event** များ (“**OrderPlaced**”) ကို **queue** တစ်ခု (**SQS** — **message** တစ်ခုစီကို **consumer** တစ်ဦးက ၎င်း၏ကိုယ်ပိုင်အရှိန်ဖြင့် ယူသည်) သို့မဟုတ် **topic** တစ်ခု (**SNS** — **subscriber** တိုင်း မိတ္တူတစ်စောင်စီရသည်။ **fan-out**) ဆီ **publish** လုပ်ကြသည်။ **Producer** က ဘယ်သူနားထောင်နေမှန်း မသိ၊ ဂရုလည်းမစိုက်။ သုံးရမည့်အခါ — အစိတ်အပိုင်းများသည် တစ်ခုချင်း၏ ပြတ်တောက်မှုများကို ခံနိုင်သင့်ခြင်း (**consumer** တစ်ဦး ကျနေစဉ် **queue** က **message** များ ကိုင်ထားပေးသည် — ဝယ်ယူမှုတစ်ခုတည်းဖြင့် **resilience** နှင့် **decoupling**)။ ဝန် ရုတ်တရက်တက်ခြင်း (**queue** က **spike** ကို စုပ်ယူပြီး **worker** များက မှန်မှန် ဖောက်ထုတ်သည်)။ ဖြစ်ရပ်တစ်ခုက တုံ့ပြန်မှုများစွာ ဖြစ်စေခြင်း (အော်ဒါတင်ပြီး → ငွေဖြတ်၊ **email**၊ ကုန်စာရင်း၊ **analytics** — **subscriber** လေးဦး၊ **coupling** သုည)။ မသုံးရမည့်အခါ — အသုံးပြုသူက အဖြေကို ချက်ချင်း လိုခြင်း (**checkout** က “တဖြည်းဖြည်း” အတည်ပြုလို့မရ)။ သို့မဟုတ် အဖွဲ့ငယ်ပြီး ရိုးရိုး **synchronous** ခေါ်ဆိုမှုနှင့် လုံလောက်ခြင်း — **queue** တိုင်းက **delivery semantics** (**at-least-once** ဆိုသည်မှာ **consumer** များသည် **idempotent** — နှစ်ကြိမ်လည်ပတ်လည်း ဘေးကင်း — ဖြစ်ရမည်)၊ **dead-letter** ကိုင်တွယ်မှုနှင့် **monitoring** ဝန်ထုပ်တစ်ခု ထပ်ထည့်သည်။ **Cost/complexity** — အစိတ်အပိုင်းများ ဈေးပေါ၊ **debug** လုပ်ရ ပိုဈေးကြီး — **request** တစ်ခု၏ဇာတ်လမ်းသည် ယခုအခါ **service** များနှင့် **queue** များအနွှံ့ ပြန့်ကျဲနေပြီ — ထို့ကြောင့်ပင် ဤနေရာမှစ၍ **observability** (**Module 7**) သည် ရွေးချယ်စရာ မဟုတ်တော့ခြင်း ဖြစ်သည်။

Three-tier: သင်ပိုင်ပြီးသား (**Module 3**)။ သုံးရန် — **web application** များ၏ ကျယ်ပြန့်သော အလယ်ပိုင်းအတွက် — ၎င်းသည် အခြားပုံစံများကို တိုင်းတာရာ စံပုံစံ ဖြစ်သည်။ မသုံးရန် — အလကားနေချိန် နက်သော အလွန်ရုတ်တရက်တက် **workload** များ (**serverless** က ပိုဈေးသက်သာသည်) သို့မဟုတ် အမှန်တကယ်ကြီးမားသော **multi-team** ထုတ်ကုန်များ (အထက်တွင်ကြည့်ပါ)။ ပရီဖိုင် — နေရာတိုင်းတွင် ကောင်းစွာနားလည်ထား၊ လူခန့်ရလွယ်၊ ကုန်ကျစရိတ် အလယ်အလတ်။

Serverless-first: **managed** အစိတ်အပိုင်းများကို စုစည်းတပ်ဆင်ပါ — **API Gateway** → **Lambda function** များ → **DynamoDB**၊ **S3** နှင့် **queue** များ — **server** တစ်လုံးမှ မပိုင်ဘဲ။ သုံးရမည့်အခါ — ရုတ်တရက်တက် သို့မဟုတ် နည်းသော **traffic** (**scale-to-zero** ဆိုသည်မှာ အလကားနေချိန် ကုန်ကျစရိတ် ~သုည)၊ အဖွဲ့ငယ်များ၊ ဈေး

ကွက်အမြန်ရောက်ရေး၊ **event-driven** ချိတ်ဆက်မှုများ။ မသုံးရမည့်အခါ — ကြာရှည်လည်ပတ်သော သို့မဟုတ် အထူးပြု **compute (runtime ကန့်သတ်ချက်များ)**၊ **latency** အလွန်စိုးရိမ်ရသော လမ်းကြောင်းများ (**cold start** များ)၊ ကြီးမားသော တည်ငြိမ် ဝန် (ခေါ်ဆိုမှုအလိုက်ဈေးက **reserved server** များထက် ကျော်နိုင်သည် — အတိုင်းအတာကြီးတွင် ဂဏန်းတွက်ပါ)၊ သို့မဟုတ် **cloud** များအကြား ရွှေ့ပြောင်းနိုင်မှုက အမှန်တကယ် လိုအပ်ချက်ဖြစ်နေချိန် (ဤသည်မှာ စာရင်းထဲက ပုံစံအားလုံးတွင် အနက်ဆုံး **lock-in** — များသောအားဖြင့် တန်သော်လည်း အသံထွက်ပြောပါ)။ ပရိုပိုင် — စာရင်းထဲတွင် လည်ပတ်ရေးဝန်ထုပ် အနည်းဆုံး။ နည်း/ရုတ်တရက် အတိုင်းအတာတွင် ကုန်ကျစရိတ် အထူးကောင်း၊ မြင့်တင်ငြိမ် အတိုင်းအတာတွင် စစ်ဆေးရန်လို။

ဝေါဟာရများ::

ဝေါဟာရ	အဓိပ္ပာယ်
Monolith / modular monolith	အားလုံးပါဝင်သော deploy လုပ်နိုင်သည့် app တစ်ခုတည်း / စည်းကမ်းရှိသောဗားရှင်း — deployable တစ်ခုတည်း၊ သန့်ရှင်းသော အတွင်း module နယ်နိမိတ်များ — စနစ်အသစ်များအတွက် အကောင်းဆုံး ပုံသေရွေးချယ်မှု။
Microservices	သီးခြားစီ deploy လုပ်သော ဝန်ဆောင်မှုငယ်များစွာ — တစ်ခုစီက ကိုယ်ပိုင်ဒေတာကို ပိုင်သည်။ Distributed-systems အခွန်ကောက်သော team-scaling ကိရိယာ။
Coupling / decoupling	အစိတ်အပိုင်းများသည် တစ်ခုချင်း၏ ရရှိနိုင်မှုနှင့် အသေးစိတ်များအပေါ် ဘယ်လောက် မှီခိုသနည်း။ Architecture ဆိုသည်မှာ decoupling အသင့်အတင့်ပမာဏကို ဝယ်ယူသည့်အနုပညာ အများစုပင်။
Synchronous / asynchronous	ခေါ်ပြီးစောင့်ခြင်းနှင့် ပို့ပြီးဆက်သွားခြင်း။ သင်ဆွဲသောမြှားတိုင်းပေါ်ရှိ အခြေခံရွေးချယ်မှု။
Event / event-driven	နားထောင်သူမည်သူ့ကိုမဆို ကြေညာလိုက်သော အချက်အလက်တစ်ခု (“ OrderPlaced ”) / ထိုကြေညာချက်များဖြင့် တည်ဆောက်ထားသော architecture ။
Queue (SQS)	Message တန်းစီကြောင်းတစ်ခု။ တစ်ခုစီကို consumer တစ်ဦးက ၎င်း၏ကိုယ်ပိုင်အရှိန်ဖြင့် ယူသည်။ Shock absorber နှင့် decoupler ။
Topic / pub-sub (SNS)	ထုတ်လွှင့်ချန်နယ်တစ်ခု။ Subscriber တိုင်း message တစ်ခုစီ ရသည်။ Fan-out ကိရိယာ။
Dead-letter queue	ထပ်ခါထပ်ခါ process မအောင်မြင်သော message များကို လူသားများအတွက် ဖယ်ထားရာနေရာ — ပုံစံ၏ ဘေးကင်းရေး

ဝေါဟာရ	အဓိပ္ပာယ်
	ကွန်ရက်။
Idempotent	နှစ်ကြိမ် process လုပ်လည်း ရလဒ်တူ၍ ဘေးကင်းခြင်း — consumer များတွင် လိုအပ်သည်။ အကြောင်းမှာ queue များသည် message တစ်ခုကို တစ်ကြိမ်ထက်ပို၍ ပို့ဆောင် နိုင်သောကြောင့်တည်း။
Serverless-first	Server များ လည်ပတ်မည့်အစား managed၊ scale-to-zero အစိတ်အပိုင်းများ (function များ၊ managed DB များ၊ queue များ) ကို စုစည်းတပ်ဆင်ခြင်း။
Lock-in	Provider တစ်ခု၏ ကိုယ်ပိုင်ဝန်ဆောင်မှုများအပေါ် မှီခိုမှု — ပြောင်းရွှေ့ရန်ဖြင့် ဈေးသတ်မှတ်ထား။ အပြစ်မဟုတ် — လင်းလင်းချင်းချင်း ချိန်ဆရမည့် စီးပွားရေးဝေါဟာရ (အဆင့် ၄)။

လေ့ကျင့်ခန်းများ: (၁) ThaiTicket ၏ အော်ဒါစီးကြောင်းကို နှစ်ကြိမ် ဒီဇိုင်းဆွဲပါ — synchronous three-tier၊ ထို့နောက် SQS/SNS ဖြင့် event-driven — ပြီးလျှင် payment-provider ပြတ်တောက်မှုတစ်ခုအတွင်း ဗားရှင်းတစ်ခုချင်း ဘာလုပ်သည်ကို စာပိုဒ်တစ်ပိုဒ် ရေးပါ။ ထိုစာပိုဒ်သည်ပင် event များအတွက် ငြင်းခုံမှုတစ်ခုလုံး ဖြစ်သည်။ (၂) ကုမ္ပဏီလေးခု (၃ ယောက် startup၊ engineer ၅၀ scale-up၊ ဘဏ်တစ်ခု၊ တစ်နှစ်လျှင် ၄ ညသားသုံးသော TV မဲပေး app) အတွက် ပုံစံတစ်ခုစီ ရွေးပြီး ခုခံကာကွယ်ပါ — ထို့နောက် ကုမ္ပဏီတစ်ခုချင်းစီကို ပုံစံပြောင်းစေမည့် အချက်ပြ (trigger) ကို နာမည်တပ်ပါ။ (၃) “Microservices ဆီ ပြောင်းပြီး နောင်တရခဲ့တယ်” engineering-blog ဇာတ်လမ်းအစစ်တစ်ပုဒ်နှင့် အောင်မြင်မှုဇာတ်လမ်းတစ်ပုဒ် ရှာပါ။ ကွာခြားချက်ကို journal ထဲရေးပါ (အမြဲလိုလို အဖွဲ့အရွယ်အစားနှင့် domain ရင့်ကျက်မှုပင်)။

Milestone: ကုမ္ပဏီဖော်ပြချက်တစ်ခု ပေးလိုက်လျှင် ပုံစံတစ်ခုကို ၎င်း၏ထွက်ပေါက်လမ်းနှင့်တကွ — “ဒီကစပါ။ X ဖြစ်လာရင် Y ဆီ ဆင့်ကဲပြောင်းပါ” — ငါးမိနစ်အတွင်း အကြံပြုနိုင်ရမည်။ Architect များ အမှန်တကယ် စကားပြောပုံမှာ စီရင်ချက်များမဟုတ်၊ ဆင့်ကဲပြောင်းလမ်းကြောင်းများ ဖြစ်သည်။

Module 9 (အပတ်စဉ် ၁၈–၂၀): Data၊ global နှင့် hybrid ပုံစံများ

Data architecture များ — lake နှင့် warehouse။ Transactional database များ (အဆင့် ၁) က လုပ်ငန်းကို လည်ပတ်စေသည်။ Analytics ကမူ ၎င်းကို မနေ့စေ့ မေးခွန်းများမေး လိုသည်။ **Data warehouse** (Redshift၊ Snowflake၊ BigQuery) သည် မြန်ဆန်သော SQL analytics အတွက် အကောင်းဆုံးပြင်ဆင်ထားသည့် ဖွဲ့စည်းပြီး၊ သန့်စင်ပြီး ဒေတာကို သိမ်းသည် — မေးခွန်းများကို သိပြီးသားဖြစ်ပြီး dashboard များ မြန်ရမည့်အခါ သုံးပါ။ TB လျှင် ပိုကုန်ကျပြီး ကြိုတင် modeling စည်းကမ်း လိုသည်။ **Data lake** (S3 + catalog + Athena ကဲ့သို့ query engine များ) သည် အားလုံးကို၊ အကြမ်းအတိုင်း၊ ဈေးပေါ်ပေါ် သိမ်းသည် — log များ၊ clickstream များ၊ ပုံများ — ဖတ်ချိန်မှသာ

ဖွဲ့စည်းပုံ သတ်မှတ်သည်။ အားလုံးကို ယခုသိမ်းထားပြီး မေးခွန်းများကို နောက်မှဆုံးဖြတ်လိုသောအခါ သုံးပါ။ သို့သော် စီမံအုပ်ချုပ်မှုမရှိလျှင် လုပ်ငန်းနယ်ပယ်ပြက်လုံးဖြစ်သော **data swamp** အဖြစ် ယိုယွင်းသွားသည်။ ခေတ်သစ်သဘောတူညီချက်မှာ နှစ်ခုလုံး၊ အလွှာလိုက် (“lakehouse”) — အမှန်တရားအကြမ်းကို lake ထဲ၊ ပြုပြင်ထားသော mart များကို warehouse ထဲ — **ETL/ELT pipeline များ** က ကျွေးမွေးသည်။ Architect ၏ စည်းကမ်းများ — **analytics** သည် **production database** ကို ဘယ်တော့မှ query မလုပ်ရ (replica များ သို့မဟုတ် pipeline များက ကျွေးသည်)၊ ပြီးလျှင် **dataset** တိုင်းတွင် ပိုင်ရှင်တစ်ဦး၊ **catalog** မှတ်တမ်းတစ်ခုနှင့် **retention policy** တစ်ခု ရှိရမည် — ထို **governance** စာကြောင်းသည် သင့်ဒီဇိုင်းထဲတွင် စာကြောင်းတစ်ကြောင်းဖြစ်ပြီး ချန်လှပ်လျှင် တစ်နှစ်စာ နာကျင်မှု ဖြစ်သည်။

Multi-region: active-passive နှင့် active-active။ Region တစ်ခုတည်း မလုံလောက်သောအခါ — လုပ်ငန်းက region အဆင့် ဘေးအန္တရာယ်များမှ DR တောင်းသောကြောင့် သို့မဟုတ် အသုံးပြုသူများ တိုက်ကြီးများအနံ့ ရှိသောကြောင့် — သင်ရွေးရသည် —

ပုံစံ	ဘာလဲ	ဘယ်အခါသုံး	ဘယ်အခါ မသုံးရ	Cost/complexity
Active-passive	Region တစ်ခုက ဝန်ဆောင်သည်။ Standby region က replicate လုပ်ထားသောဒေတာကို “backup များသာ” (cold) မှ “ချုံထားသောမိတ္တူ လည်နေသည်” (warm) အထိ ကိုင်ထားပြီး ဘေးအန္တရာယ်ကျမှ တင်မြှောက်သည်	လုပ်ငန်းဖော်ပြထားသော RTO/RPO က ထိုက်တန်စေခြင်း။ Compliance က DR ဇာတ်လမ်းတောင်းခြင်း	သုံးစွဲမှုကို ထိုက်တန်စေသော RTO/RPO ကို ဘယ်သူမှ လက်မှတ်မထိုးရသေးခြင်း (ကုမ္ပဏီအများစု၏ ရိုးသားသော လိုအပ်ချက်မှာ ကောင်းသော multi-AZ + cross-region backup များ)	Standby နွေးထွေးမှု အလိုက် ကုန်ကျစရိတ် 1.1×–1.7×။ ရှုပ်ထွေးမှု အလယ်အလတ် — သေချာလိုအပ်ချက်မှာ အမှန်တကယ် drill လုပ်သော failover၊ မဟုတ်လျှင် standby သည် ပြဇာတ်သာ
Active-active	Region နှစ်ခု+ တစ်ပြိုင်နက် ဝန်ဆောင်သည်။ အသုံးပြုသူများကို အနီးဆုံးဆီ လမ်းကြောင်းချသည်။ ဒေတာကို နှစ်ဘက်လုံး replicate လုပ်သည်	Local latency လိုချင်သော ကမ္ဘာလုံးဆိုင်ရာ အသုံးပြုသူအခြေခံ။ သုညနီးပါး RTO အမှန်တကယ် လိုအပ်ခြင်း (ငွေပေးချေမှုကွန်ရက်များ၊ trading၊ SaaS ကြီးများ)	ကျန်လူအားလုံးနီးပါး — cross-region write conflict များသည် ကွန်ပျူတာပညာ၏ အမှန်တကယ်ခက်ခဲသော ပြဿနာများထဲက တစ်ခု	ကုန်ကျစရိတ် 2×+၊ ဤသင်တန်းထဲက အမြင့်ဆုံးရှုပ်ထွေးမှု။ Conflict ဖြေရှင်းပေးသော data store များနှင့် senior အဖွဲ့များ လိုသည်

အင်တာဗျူးအဆင့် စာကြောင်း — “Multi-AZ is table stakes; multi-region is a business case.” (Multi-AZ က အခြေခံလိုအပ်ချက်၊ multi-region ကတော့ လုပ်ငန်းအကြောင်းပြချက် လိုတယ်။) လုပ်ငန်းကို RTO/RPO ဖော်ပြခိုင်းပါ။ ရွေးချယ်စရာများကို ဈေးတွက်ပါ။ ဂဏန်းများကို ရွေးချယ်ခွင့်ပေးပါ။

Hybrid cloud။ တစ်ပိုင်း on-prem၊ တစ်ပိုင်း cloud — VPN သို့မဟုတ် Direct Connect ဖြင့် ဆက်ထား — တည်ထောင်ပြီးသား enterprise အများစုအတွက် ပုံစံတစ်ခုမဟုတ်ဘဲ ဆယ်စုနှစ်ကြာ လက်တွေ့ဘဝ — မရွေးနိုင်သော mainframe များ၊ latency ချည်နှောင်ထားသော စက်ရုံစနစ်များ၊ on-premises တွင် ချည်ထားသည့် ထိန်းချုပ်ခံဒေတာ၊ နှင့် ဖြတ်သန်းနေဆဲ migration (အဆင့် ၄) တစ်ခု။ သုံးရမည့်ပုံစံ — ဦးတည်ရာလမ်းကြောင်းပါသော ရည်ရွယ်ချက်ရှိရှိ တံတား။ လက်မခံရမည့်အရာ — “hybrid” ကို “ကျွန်တော်တို့ ဘယ်တော့မှ မဆုံးဖြတ်ခဲ့ဘူး” ၏ ယဉ်ကျေးသောအစားထိုး စကားအဖြစ်။ ဒီဇိုင်းမှတ်စုများ — identity ကို အရင်ဆုံး ပေါင်းစည်းရမည် (ကမ္ဘာနှစ်ခုလုံးအနှံ့ login တစ်ခုတည်း — enterprise အလုပ်ခေါ်စာများက “hybrid identity integration” ဟုဆိုရာတွင် ဆိုလိုသောအရာ)။ ဘယ်စနစ်များက အမှန်တရား၏ဇာစ်မြစ် (source of truth) ဖြစ်သည်ကို နာမည်တပ်ပါ။ ကြိုးတစ်လျှောက် egress ကုန်ကျစရိတ်များကို စောင့်ကြည့်ပါ။ ပြီးလျှင် network link သည် နှစ်ဆမလုပ်ထားလျှင် SPOF ဖြစ်နေမည် ကို မျှော်လင့်ထားပါ။ ရှုပ်ထွေးမှု — အရာအားလုံး နှစ်စုံစာ — architect များက on-prem ဘက်ကို တဖြည်းဖြည်း ချုံ့ရန် တွန်းအားပေးရသည့် ရိုးသားသော အကြောင်းရင်း။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
OLTP / OLAP	Transaction processing (သေးငယ်မြန်ဆန်သော ဖတ်/ရေးများစွာ — လုပ်ငန်းကို လည်ပတ်စေသည်) နှင့် analytics processing (ကြီးမားသော scan များ — လုပ်ငန်းကို လေ့လာသည်)။ ခွဲထားပါ။
Data warehouse	မြန်ဆန်သော SQL analytics အတွက် အကောင်းဆုံး ပြင်ဆင်ထားသည့် ဖွဲ့စည်းပြီး၊ ပြုပြင်ပြီး သိုလှောင်မှု (Redshift၊ Snowflake၊ BigQuery)။
Data lake / data swamp	အားလုံးကို အကြမ်းအတိုင်း ဈေးပေါပေါသိမ်းပြီး ဖတ်ချိန်မှ ဖွဲ့စည်းပုံသတ်မှတ်ခြင်း (S3 + Athena) / စီမံအုပ်ချုပ်မှုမရှိသောဗားရှင်း — ဒေတာများ ပျောက်ဆုံးရန် သွားရာနေရာ။
ETL / ELT	ဒေတာကို ဇာစ်မြစ်စနစ်များမှ lake/warehouse ထဲ ရွှေ့ပေးသော pipeline များ (Extract၊ Transform၊ Load — အစဉ် ကွဲနိုင်သည်)။
Data governance / catalog / retention	Dataset တိုင်းအတွက် ပိုင်ဆိုင်မှု၊ စာတမ်းပြုမှုနှင့် သက်တမ်းစည်းကမ်းများ — တစ်နှစ်စာနာကျင်မှုကို ကယ်တင်ပေးသော ဒီဇိုင်းစာကြောင်းတစ်ကြောင်း။

ဝေါဟာရ	အဓိပ္ပာယ်
Replication (sync / async)	ဒေတာကို အခြား database သို့မဟုတ် region ဆီ အဆက်မပြတ် ကူးခြင်း — ချက်ချင်းကိုက်ညီသော်လည်း အကွာအဝေးကန့်သတ်ခံ နှင့် အနည်းငယ်နောက်ကျသော်လည်း ဘယ်နေရာမဆိုရ။ Async နောက်ကျချိန်သည် RPO ၏ ဇာစ်မြစ်။
Active-passive / failover drill	Standby-region DR / ၎င်းအလုပ်ဖြစ်ကြောင်း သက်သေပြသော အချိန်ဇယားကျ ဇာတ်တိုက်မှု။ Drill မလုပ်ရသေးသော failover သည် မျှော်လင့်ချက်သာ။
Active-active	Region များစွာ တစ်ပြိုင်နက် ဝန်ဆောင်ပြီး နှစ်လမ်းသွား replication။ အစွမ်းထက်သည်။ အမှန်တကယ် ခက်သည်။ များသောအားဖြင့် မလိုအပ်ပါ။
Write conflict	Region နှစ်ခုက တူညီသောဒေတာကို တစ်ပြိုင်နက် ပြောင်းခြင်း — active-active သည် ကျွမ်းကျင်သူနယ်မြေဖြစ်ရသည့် နည်းပညာအကြောင်းရင်း။
Hybrid cloud / hybrid identity	On-prem + cloud ကို ပိုင်ဆိုင်မှုတစ်ခုတည်းအဖြစ် ဆက်ထားခြင်း / နှစ်ခုလုံးအနွံ sign-on တစ်ခုတည်း — ပထမဆုံး ပေါင်းစည်းရမည့်အရာ။
Direct Connect	Data center နှင့် cloud ကို ဆက်ပေးသော သီးသန့် ရုပ်ပိုင်းဆိုင်ရာလိုင်း — hybrid ချက်ကြိုး၊ အရေးကြီးလျှင် နှစ်ဆလုပ်ထား။

လေ့ကျင့်ခန်းများ: (၁) ThaiTicket ဒေသတွင်း ချဲ့ထွင်သည် — စင်္ကာပူ ချဲ့ထွင်မှုကို နှစ်ကြိမ် ဒီဇိုင်းဆွဲပါ — active-passive (ဘန်ကောက် primary) နှင့် active-active — တစ်ခုစီအတွက် ကုန်ကျစရိတ်အဆများနှင့် RTO/RPO ပါလျက်။ တစ်မျက်နှာ ထောက်ခံချက် ရေးပြီး ဆုံးဖြတ်ချက်ချပါ။ (၂) လက်လီရောင်းချသူတစ်ဦးတွင် production SQL database ထဲ ၁၅ နှစ်စာ အရောင်းမှတ်တမ်းရှိပြီး “AI-ready analytics” လိုချင်သည်။ Lake + warehouse ဒီဇိုင်းကို ပုံကြမ်းဆွဲပြီး governance စာကြောင်းသုံးကြောင်း ရေးပါ။ (၃) ပုံစံရှာဖွေခြင်း — <https://aws.amazon.com/architecture/> မှ reference architecture သုံးခု ရွေးပြီး တစ်ခုစီတွင် ပါဝင်သော စာရင်းထဲကပုံစံတိုင်းကို နာမည်တပ်ပါ။

Milestone — အဆင့် ၃ အဆုံး: စာရင်း စိန်ခေါ်မှုကြီး — AI တစ်ခုက အမြန် scenario ခြောက်ခု ပေးသည်။ တစ်ခုစီအတွက် ပုံစံ၊ အကြောင်းရင်း၊ ဘယ်အခါ-မသုံးရ သတိပေးချက်နှင့် cost/complexity ပရိုဖိုင်ကို ပြောရမည် — တစ်ခုလျှင် ငါးမိနစ်အောက်။ ဤအပြန်အလှန် အတိအကျ၊ ဤအရှိန် အတိအကျသည်ပင် architect အင်တာဗျူးအစစ်၏ အလယ်သုံးပုံတစ်ပုံ ဖြစ်သည်။

အဆင့် ၄ — MIGRATION နှင့် လက်တွေ့ကမ္ဘာ (အပတ်စဉ် ၂၁-၂၆)

Greenfield ဒီဇိုင်းသည် အလုပ်၏ လူနည်းစုသာ ဖြစ်သည်။ Architecture အများစုသည် ရှိပြီးသားကုမ္ပဏီများတွင် ဖြစ်ပွားသည် — server ခန်းများ၊ လုပ်ငန်းက မှီခိုနေရသော ရှေးဟောင်း software များ၊ စာချုပ်များနှင့် ဥပဒေများနှင့် အတူ။ ဤအဆင့်သည် ထိုကမ္ဘာဖြစ်ပြီး ကျွန်ုပ်တို့လေ့လာခဲ့သော အလုပ်ခေါ်စာနီးပါးတိုင်းပါ အချက်ဖြစ်သည့် “lead migration and modernization initiatives” ကို သင်ကြားပေးရာနေရာ ဖြစ်သည်။

Module 10 (အပတ်စဉ် ၂၁-၂၃): Migration — ကုမ္ပဏီတစ်ခုကို cloud ထဲ ရွှေ့ခြင်း

7 R များ — migration စကားပိုင်းတိုင်း၏ မျှဝေဝေါဟာရ။ ပိုင်ဆိုင်မှုထဲရှိ workload တစ်ခုချင်းစီအတွက် တစ်ခု ရွေးရ သည် —

R	အဓိပ္ပာယ်	ဘယ်အခါ	အားစိုက်မှု / အကျိုးရလဒ်
Retire	ပိတ်လိုက်ပါ — ဘယ်သူမှ တကယ်မသုံးပါ	ပိုင်ဆိုင်မှုတိုင်းတွင် ၁၀-၂၀% ရှိသည်။ ဒါတွေကို အရင်ရှာပါ	အလွန်လွယ် / ချက်ချင်း ချွေတာမှု — အကောင်းဆုံး R
Retain	On-prem မှာ ထားပါ။ လောလောဆယ်	Latency ချည်နှောင်ခံ၊ compliance ချည်ထားခံ သို့မဟုတ် မကြာခင် သက်တမ်းကုန်မည့် စနစ်များ	မရှိ / ကုန်ကျစရိတ် ရွှေ့ဆိုင်း — ရိုးသားသော “မဟုတ်သေးဘူး”
Rehost (“lift and shift”)	Cloud VM များဆီ အတိုင်း ရွှေ့	အမြန်လိုခြင်း၊ app များ တည်ငြိမ်ခြင်း၊ ကျွမ်းကျင်မှု နည်းခြင်း	နည်း / data center မှ အမြန်ထွက် — သို့သော် cloud အကျိုး နည်းသေး — ပထမခြေလှမ်း၊ ဦးတည်ရာ မဟုတ်
Relocate	Hypervisor အဆင့် ရွှေ့ (ဥပမာ VMware အုပ်စုကို VMware-on-cloud ဆီ)	Deadline ရှိသော ကြီးမားသည့် virtualized ပိုင်ဆိုင်မှု များ	နည်း / အမြန်ဆုံး အစုလိုက် ရွှေ့ခြင်း။ လမ်းခုလတ် မှတ်တိုင်
Repurchase (“drop and shop”)	SaaS ဖြင့် အစားထိုး	ထူးခြားမှုမရှိသော app များ — email၊ HR၊ CRM	နည်း-အလယ်အလတ် / အမျိုးအစားတစ်ခုလုံးများ သင့်ပိုင်ဆိုင်မှုမှ ထွက်သွား

R	အဓိပ္ပာယ်	ဘယ်အခါ	အားစိုက်မှု / အကျိုးရလဒ်
Replatform (“lift, tinker, shift”)	ရွှေ့ရင်း အဆင့်မြှင့်မှုငယ်များ — ကိုယ်တိုင်စီမံ DB → RDS၊ app → container များ	လက်တွေ့ကျသော အလယ် ပိုင်း — အကျိုးအစစ်၊ ဘောင်ခတ်ထားသော အန္တရာယ်	အလယ်အလတ် / migration အများစု၏ အလုပ်သန် R
Refactor / re-architect	Cloud-native အတွက် ပြန်ရေး (အဆင့် ၃ ပုံစံများ)	ကန့်သတ်ချက်များက လုပ်ငန်းကို နာကျင်စေသော အဓိကထူးခြားစနစ်များ	မြင့် / အကျိုးရလဒ်အမြင့်ဆုံး — ထိုက်တန်သော workload အနည်းငယ် အတွက်သာ သုံးပါ

Migration နည်းလမ်း: assess → mobilize → migrate။ Assess (အကဲဖြတ်ခြင်း): အားလုံးကို စာရင်းကောက်ပါ (discovery ကိရိယာများ + လူများကို မေးမြန်းသော ရှေးဟောင်းသုတေသန) ပြီးလျှင် workload တစ်ခုချင်းကို လုပ်ငန်းတန်ဖိုးနှင့် migration ခက်ခဲမှုအလိုက် အမှတ်ပေးပါ — ထွက်ကုန်များ: တစ်တန်းလျှင် R တစ်ခုပါသော application inventory တစ်ခုနှင့် **business case** တစ်ခု (နေမြဲနေခြင်းနှင့် ရွှေ့ခြင်း၏ TCO။ ပိုင်ဆိုင်မှုနှစ်ခုလုံး လည်နေသော ထပ်နေကာလအတွင်း ဘေလ် တက်သည် ကို ရိုးသားစွာပြောပါ — migration bubble အကြောင်း သတိမပေးခံရသော ခေါင်းဆောင်များသည် migration ကို လမ်းတစ်ဝက်တွင် ဖျက်သိမ်းသောခေါင်းဆောင်များ ဖြစ်လာတတ်သည်)။ **Mobilize (ပြင်ဆင်ခြင်း):** ပထမဆုံး workload မဆိုက်မီ **landing zone** — ကြိုတည်ဆောက်ထား၊ စီမံအုပ်ချုပ်ထားသော cloud အခြေခံအုတ်မြစ် — ကို တည်ဆောက်ပါ: multi-account ဖွဲ့စည်းပုံ (environment နှင့် အဖွဲ့တစ်ခုစီအတွက် သီးခြား account များ — blast radius များ သေးနေစေရန်)၊ ဗဟိုချုပ်ကိုင် identity နှင့် logging၊ network hub (hub-and-spoke VPC များ၊ on-prem ဆီပြန်သော Direct Connect)၊ နှင့် **guardrail** များ — လုံခြုံ၊ tag တပ်ပြီး၊ စည်းကမ်းညီသောလမ်းကြောင်းကို ပုံသေဖြစ်စေသော အလိုအလျောက် policy များ (AWS Control Tower သည် managed အစပြုကိရိယာစုံ)။ အလုပ်ခေါ်စာများက “design the landing zone every workload inherits” ဟုဆိုသည် — ဒါက အဲဒါပဲ။ “စပြီး ရွှေ့လိုက်ရအောင်” ဟု ၎င်းကိုကျော်လျှင် မသပ်ရပ်သော data center ကို cloud ထဲတွင် ပိုမြင့်သောဈေးဖြင့် ပြန်ဖန်တီးမိသည် — ၎င်းသည် ကျရှုံးသော migration ၏ ဂန္ထဝင်လက္ခဏာ ဖြစ်သည်။ **Wave များဖြင့် migrate လုပ်ပါ:** workload များကို အနည်းငယ်စီပါသော wave များအဖြစ် အုပ်စုဖွဲ့ပြီး လွယ်သည်များအရင် အစဉ်စီပါ — Wave 1 သည် တမင် အန္တရာယ်နည်းအောင် ထား (အမှားများ ဈေးပေါသောနေရာတွင် စက်ယန္တရားကို သင်ယူပါ)။ နောက်ပိုင်း wave များက အဖိုးတန်ရတနာများကို ဇာတ်တိုက်ပြီးသား cutover များဖြင့် ယူသည် — တစ်ခုစီတွင် rollback အစီအစဉ်နှင့် အထူးစောင့်ကြည့်ကာလ **hypercare** ပါလျက်။ Database ရွှေ့မှုတစ်ခုချင်းစီအတွက် အရေးကြီးသောမေးခွန်းနှစ်ခုမှာ Module 4 ၏ ဂဏန်းများ ရုပ်ဖျက်ထားခြင်းသာ — cutover က ရပ်တန့်ချိန် ဘယ်လောက်ယူခွင့်ရှိသနည်း (RTO) — ပြီးလျှင် အဖြေက “မရှိသလောက်” ဆိုလျှင် ပြောင်းချိန်တို အဆက်မပြတ် replication ရှိသည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
7 Rs	Retire၊ retain၊ rehost၊ relocate၊ repurchase၊ replatform၊ refactor — workload တစ်ခုချင်း migration menu။ (“6 Rs” — relocate မပါသော စာရင်းအဟောင်း — ကိုလည်း ကြားရမည်။)
Discovery / application inventory	ဘာတွေ တကယ်လည်နေမှန်း ရှာဖွေခြင်း (ကိရိယာများ + အင်တာဗျူးများ) / ရလာသောစာရင်း — workload တစ်ခုလျှင် တစ်တန်း၊ ပိုင်ရှင်၊ dependency များနှင့် ၎င်း၏ R ပါလျက်။
Dependency mapping	ဘာက ဘာနှင့် စကားပြောသည်ကို ဇယားဆွဲခြင်း — workload များ အုပ်စုလိုက် migrate ရသည့်အကြောင်းရင်း၊ ၎င်းမပါသော migration များ ပထမနေ့တွင် ကျရှုံးသည်။
Business case / migration bubble	နေမြဲ-နှင့်-ရွှေ့ TCO ငြင်းခုံချက် / ပိုင်ဆိုင်မှုနှစ်ခုလုံး လည်နေစဉ် ယာယီကုန်ကျစရိတ်ကုန်း။ သတိပေးပါ၊ မဟုတ်လျှင် ၎င်း၏ ခြုံခိုတိုက်ခိုက်မှုခံရမည်။
Landing zone	Workload တိုင်း အမွေဆက်ခံသော စီမံအုပ်ချုပ်ထား၊ ကြိုတည်ဆောက်ထားသည့် cloud အခြေခံအုတ်မြစ် — account များ၊ identity၊ network၊ logging၊ guardrail များ။ Migration မတိုင်မီ တည်ဆောက်သည်။
Multi-account strategy	Environment/အဖွဲ့တစ်ခုစီအတွက် သီးခြား cloud account များ — billing တာဝန်ခံနိုင်ပြီး blast radius ထိန်းထားနိုင်စေရန်။
Guardrail	စည်းကမ်းမညီသောလုပ်ရပ်များကို တားဆီး သို့မဟုတ် အလံတူပေးသော အလိုအလျောက် policy — မှတ်ချက်စာများမဟုတ်၊ စက်ယန္တရားအဖြစ် governance။
Control Tower	Guardrail များပါသော multi-account landing zone ထောင်ရန် AWS ၏ managed ဝန်ဆောင်မှု။
Wave plan	အုပ်စုငယ်များဖြင့် migration အချိန်ဇယား — လွယ်သည်များအရင်၊ dependency များ အတူတူ။
Cutover / rollback / hypercare	Cloud မိတ္တူဆီ ပြောင်းသည့်အခိုက်အတန့် / ဇာတ်တိုက်ပြီးသား ပြန်ဖျက်နည်း / ပြီးနောက် အထူးပံ့ပိုးစောင့်ကြည့်ကာလ။

လေ့ကျင့်ခန်းများ: (၁) စက္ကူပေါ် migration — AI တစ်ခုက စိတ်ကူးယဉ် server ၄၀ လုံး ကုမ္ပဏီ inventory တစ်ခု ထုတ်ပေးသည် (Capstone 2 တွင် ပြန်တွေ့ရမည်)။ တန်းတိုင်းကို R တစ်ခုစီ သတ်မှတ်ပြီး အခက်ဆုံး ဆုံးဖြတ်ချက်ဆယ် ခုကို ခုခံကာကွယ်ပါ။ (၂) Landing zone တစ်ခု ပုံကြမ်းဆွဲပါ — account diagram၊ identity စီးကြောင်း၊ network hub-and-spoke၊ ပထမနေ့မှစ၍ တင်းကျပ်မည့် guardrail ငါးခု။ (၃) CFO ထံ “migration bubble” သတိပေးချက် စာပိုဒ်နှစ်ပိုဒ် ရေးပါ — သတင်းဆိုးကို စောစောပြောခြင်းကို လေ့ကျင့်ခြင်းသည် architect ၏ ကိုယ်ကာယလေ့ကျင့်ခန်း။

Milestone: ဖော်ပြချက်များပါ workload ~၂၀ ခု inventory တစ်ခု ပေးလိုက်လျှင် ခုခံနိုင်သော workload-တစ်ခုချင်း- R ဇယား၊ အကြောင်းပြချက်ပါ သုံး-wave အစီအစဉ်နှင့် landing-zone ပုံကြမ်း — တစ်ထိုင်တည်း ထုတ်လုပ်နိုင်ရမည်။

Module 11 (အပတ်စဉ် ၂၄-၂၆): ကန့်သတ်ချက်များ — ဥပဒေ၊ legacy နှင့် lock-in

Architecture input များအဖြစ် data residency နှင့် privacy ဥပဒေ။ ကိုယ်ရေးကိုယ်တာဒေတာ ဥပဒေများ — ထိုင်း၏ PDPA၊ ဥရောပ၏ GDPR နှင့် ကမ္ဘာအနှံ့ ၎င်းတို့၏ဆွေမျိုးများ — တွင် architect တစ်ဦး ရေရေလည်လည် သိရမည့် ပုံသဏ္ဌာန်တစ်ခု တူညီစွာရှိသည်။ ကိုယ်ရေးကိုယ်တာဒေတာသည် တရားဝင်အခြေခံ (များသောအားဖြင့် သဘောတူညီချက်) လိုသည်။ လူပုဂ္ဂိုလ်တစ်ဦးချင်းတွင် အခွင့်အရေးများရှိသည် (ကြည့်ရှုခွင့်၊ ပြင်ဆင်ခွင့်၊ ဖျက်ခိုင်းခွင့် — သင့်ဒီဇိုင်းသည် လူတစ်ဦး၏ဒေတာကို ရှာပြီး ဖျက်နိုင်ရမည် — စီမံအုပ်ချုပ်မှုမရှိသော မိတ္တူများအနံ့ ကြွေးလျှင် ခက်သည်)။ ပေါက်ကြားမှုများတွင် ၇၂ နာရီ အကြောင်းကြားရန် တာဝန်ရှိသည် (သင့် logging က ထိုထက်တိုသောအချိန်အတွင်း ဇာတ်လမ်းပြောနိုင်ရမည်)။ ပြီးလျှင် cross-border transfer စည်းကမ်းများ က ဒေတာ ရပ်ပိုင်းအရ ဘယ်နေရာနေထိုင်ခွင့်ရှိသည်ကို ကန့်သတ်သည်။ ၎င်းသည် region ရွေးချယ်ရေး ကန့်သတ်ချက်၊ replication ဒီဇိုင်း ကန့်သတ်ချက် (စင်ကာပူရှိ ထို analytics မိတ္တူသည် ဥပဒေဖြစ်ရပ်တစ်ခု ဖြစ်နိုင်သည်)၊ နှင့် နိုင်ငံတွင်း region များ တည်ရှိရသည့် အကြောင်းရင်းတစ်ခု ဖြစ်သည်။ Architect ၏ နည်းလမ်း၊ အစဉ်လိုက် — ဒေတာကို classify လုပ်ပါ (ဘာက ကိုယ်ရေးကိုယ်တာလဲ)၊ ၎င်း၏ခရီးစဉ်များကို မြေပုံဆွဲပါ (store တိုင်း၊ မိတ္တူတိုင်း၊ ဖြတ်ကျော်သောနယ်စပ်တိုင်း — ဘယ်သူမှ မဆွဲထားသောစီးကြောင်းများသည် ချိုးဖောက်မှုများ နေထိုင်ရာ)၊ ထို့နောက် ထိန်းချုပ်မှုများ ဒီဇိုင်းဆွဲပါ — residency ကိုက်ညီသော region များ၊ encryption၊ retention အချိန်ဇယားများ၊ နှင့် backup များ၏ သက်တမ်းကုန်ဆုံးမှုအထိ အမှန်တကယ် ရောက်သော ဖျက်သိမ်းရေးလမ်းကြောင်း။ “Data protection by design” ဟု ပြောပါ။ ၎င်းသည် ဥပဒေနှစ်ခုလုံး သုံးသောစကားစုဖြစ်ပြီး စာကြောင်းတစ်ကြောင်းတည်းဖြင့် သင့်ရာထူးအမည်ပင် ဖြစ်သည်။

Legacy ပေါင်းစည်းခြင်း။ Mainframe billing စနစ်သည် ဒီနှစ် မရွေ့ပါ။ ပြီးလျှင် တောက်ပြောင်သော cloud app က ၎င်းနှင့် စကားပြောရမည်။ ပုံစံများ — anti-corruption layer — အသစ်နှင့်အဟောင်းအကြား ဘာသာပြန်ဝန်ဆောင်မှု တစ်ခု — legacy စနစ်၏ ထူးဆန်းသောဒေတာပုံသဏ္ဌာန်များ သင့်ဒီဇိုင်းအသစ်ထဲ ယိုစိမ့်ပျက်စီးမသွားစေရန်။ **Strangler fig** — traffic ကို façade တစ်ခုမှတစ်ဆင့် လမ်းကြောင်းချပြီး legacy စနစ်မှ function များကို တစ်ခုချင်း ခွာယူကာ နှစ်များကြာပြီးနောက် ၎င်းကို ပိတ်နိုင်သည်အထိ (အိမ်ရှင်သစ်ပင်ကို ဖြည်းဖြည်းချင်း ရစ်ပတ်ဖုံးလွှမ်းသော ညောင်ပင်ကို အစွဲပြု နာမည်ပေးထား။ ဒဏ္ဍာရီဆန်သောနှုန်းဖြင့် ကျရှုံးတတ်သည့် big-bang ပြန်ရေးမှုများထက် သာသည်)။ ကမ္ဘာနှစ်ခု အကြား စီးရမည့်ဒေတာအတွက် batch တံတားများနှင့် event ခွဲယူမှုများ။ ပြီးလျှင် စကားပြောများသော cloud-to-on-prem ခေါ်ဆိုမှုများ၏ latency ရှုပေဒကို လေးစားခြင်း (Direct Connect က ကူသည်။ ပိုကောင်းသည်မှာ စကားပြောများမှုကို ဒီဇိုင်းဖြင့် ဖယ်ထုတ်ခြင်း)။ စည်းကမ်း — legacy ကို ထိန်းချုပ်ပါ။ ကူးစက်မခံပါနှင့်။ အစိတ်အပိုင်းအသစ်တိုင်းကို legacy စနစ် မရှိတော့ပြီးသားဆိုသလို တည်ဆောက်သင့်သည်။

Vendor lock-in စီးပွားရေး။ အဆင်ပြေသော managed ဝန်ဆောင်မှုတိုင်းသည် provider တစ်ခုနှင့် သင့်ထိမ်းမြားမှုကို ပိုနက်စေသည်။ Portability (multi-cloud abstraction များ၊ အားလုံးကိုယ်တိုင်စီမံခြင်း) သည် ရှုပ်ထွေးမှုနှင့် ဆုံးရှုံးသွားသောအရှိန်တို့ဖြင့် ငွေအစစ်ကုန်သော ရွေးချယ်စရာအစစ် ဖြစ်သည်။ နှစ်ခုလုံး အပြစ်မဟုတ် — ကျရှုံးမှုပုံစံမှာ lock-in ကို ရွေးချယ်ခြင်း မဟုတ်ဘဲ မသတိထားမိခြင်း ဖြစ်သည်။ Architect ၏ ကိရိယာမှာ ဒီဇိုင်းထဲ ရေးသွင်းထားသော exit-cost ခန့်မှန်းချက် — “DynamoDB သုံးခြင်းက ယခု engineer-နှစ် ≈2 ချွေတာသည်။ နောင်ပြောင်းလျှင် data layer ၏ ≈6 လ ပြန်ရေးမှု — ကျွန်ုပ်တို့ လက်ခံသည်၊ ပြီးလျှင် ပြန်ရေးမှုကို ချုံပေးမည့် interface နယ်နိမိတ်မှာ ဤသည်။” စာကြောင်းသုံးကြောင်း၊ ရိုးသားစွာ ဈေးတွက်ထား၊ ဆုံးဖြတ်ချက် မှတ်တမ်းတင်ထား (အဆင့် ၅ ၏ ADR များ) — ၎င်းသည် ရင့်ကျက်သော lock-in စီမံခန့်ခွဲမှု ဖြစ်သည်။ နည်းဗျူဟာအဖြစ် multi-cloud (workload တစ်ခုတည်းကို cloud နှစ်ခုပေါ်တွင် ရွှေ့ပြောင်းနိုင်အောင် လည်ပတ်ခြင်း) သည် များသောအားဖြင့် အဈာန်အကြီးဆုံး ဖြစ်နိုင်သောအဖြေဖြစ်ပြီး ၎င်း၏အတိုင်းအတာ သို့မဟုတ် ထိန်းချုပ်သူက တောင်းဆိုသော အဖွဲ့အစည်းများကသာ အတိအကျ ဝယ်ယူကြသည်။ အဖြစ်မှန်အဖြစ် multi-cloud (သမိုင်း သို့မဟုတ် ကိုက်ညီမှုကြောင့် workload မတူများ cloud မတူများပေါ်ရှိခြင်း) သည် ပုံမှန်ဘဝသာ။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
PDPA / GDPR	ထိုင်းနှင့် ဥရောပ၏ ကိုယ်ရေးကိုယ်တာဒေတာ ကာကွယ်ရေး ဥပဒေများ — ကမ္ဘာအနှံ့ privacy-ဥပဒေ ကန့်သတ်ချက်များ အတွက် ပုံစံခွက်အတွဲ။
Data protection by design	ပထမဆုံးပုံကြမ်းမှစ၍ privacy ထိန်းချုပ်မှုများကို architecture ထဲ တည်ဆောက်ထည့်သွင်းခြင်း — ဥပဒေ စကားစုနှင့် architect ၏တာဝန်။
Lawful basis / consent	ကိုယ်ရေးကိုယ်တာဒေတာ ကိုင်တွယ်ရန် လိုအပ်သော ဥပဒေ အရ တရားဝင်မှု။
Right to erasure	မိမိဒေတာကို ဖျက်ခိုင်းနိုင်သော လူတစ်ဦးချင်း၏အခွင့်အရေး — သင့်ဒီဇိုင်းက ဖြစ်နိုင်အောင် လုပ်ပေးရမည်။
Data flow mapping	ဒေတာအမျိုးအစားတစ်ခု၏ store တိုင်း၊ မိတ္တူတိုင်းနှင့် နယ်စပ်ဖြတ်ကျော်မှုတိုင်းကို ဇယားဆွဲခြင်း။ မဆွဲထားသောစီးကြောင်းများရှိရာတွင် ချိုးဖောက်မှုများ နေထိုင်သည်။
Cross-border transfer	ကိုယ်ရေးကိုယ်တာဒေတာ နိုင်ငံမှထွက်ခြင်း — ထိန်းချုပ်ခံ။ Replication ဒီဇိုင်းကို ဥပဒေမေးခွန်း ဖြစ်စေသည်။
Anti-corruption layer	Legacy စနစ်တစ်ခု၏ ထူးဆန်းချက်များ ဒီဇိုင်းအသစ်ထဲ မယိုစိမ့်စေရန် ကာကွယ်ပေးသော ဘာသာပြန်အစိတ်အပိုင်း။

ဝေါဟာရ	အဓိပ္ပာယ်
Strangler fig	Façade တစ်ခုမှတစ်ဆင့် လမ်းကြောင်းချပြီး legacy စနစ်ကို ပိတ်နိုင်သည်အထိ တစ်ပိုင်းချင်း အစားထိုးခြင်းဖြင့် ခေတ်မီအောင်ပြုပြင်ခြင်း။
Big-bang rewrite	စနစ်တစ်ခုကို တစ်ပြိုင်နက်လုံး အစားထိုးပြီး တစ်နေ့တည်း cutover လုပ်ခြင်း။ ဒဏ္ဍာရီဆန်သော ကျရှုံးနှုန်း။ Strangler fig သည် ၎င်းကြောင့် ပေါ်လာခြင်း။
Exit cost / switching cost	Provider သို့မဟုတ် ဝန်ဆောင်မှုတစ်ခုမှ ထွက်ခွာခြင်း၏ လက်တွေ့ကျကျ ဈေးတွက်ထားသောကုန်ကျစရိတ် — lock-in ကို ကြောက်စိတ်မှ စီးပွားရေးအဖြစ် ပြောင်းပေးသော ဂဏန်း။
Multi-cloud (နည်းဗျူဟာနှင့် အဖြစ်မှန်)	Cloud များစွာပေါ်တွင် ရွှေ့ပြောင်းနိုင်အောင် တမင်လည်ပတ်ခြင်း (ဈေးကြီး၊ အကြောင်းပြချက်ရှား) နှင့် workload များ cloud အမျိုးမျိုးပေါ် ရှိရှိခြင်း (ပုံမှန်)။

လေ့ကျင့်ခန်းများ: (၁) ThaiTicket ၏ ဖောက်သည်ဒေတာကို data-flow-map ဆွဲပါ — store တိုင်း၊ မိတ္တူတိုင်း (log များ၊ backup များ၊ analytics pipeline၊ support အဖွဲ့၏ export များကို မမေ့ပါနှင့်)၊ နယ်စပ်တိုင်း၊ ထို့နောက် ဖျက်သိမ်းရေးလမ်းကြောင်း (erasure path) ကို ရေးပါ။ Diagram က ဖုံးကွယ်ထားသောပြဿနာများကို မြေပုံက ရှာတွေ့ပုံကို ခံစားကြည့်ပါ။ (၂) သက်တမ်း ၂၀ နှစ်ရှိ on-prem ကုန်ပစ္စည်းစာရင်းစနစ်တစ်ခုအတွက် strangler-fig အစီအစဉ်ကို ဒီဇိုင်းဆွဲပါ — ပထမဆုံး အခွာသုံးခုကို နာမည်တပ်လျက်။ (၃) သင်အမှန်တကယ်သုံးမည့် ဝန်ဆောင်မှုသုံးခုအတွက် DynamoDB ပုံစံ lock-in စာပိုဒ် (ယခုအကျိုး၊ နောင် exit cost၊ လက်ခံ သို့မဟုတ် လျော့ပါးအောင်လုပ်) ကို ရေးပါ။

Milestone — အဆင့် ၄ အဆုံး: ကန့်သတ်ချက် စိန်ခေါ်မှုကြီး — သုံးမျိုးလုံးဖြင့် ရက်ထားသော ဒီဇိုင်းအမှာစာတစ်စောင် (“ထိုင်း အာမခံကုမ္ပဏီ၊ ဖောက်သည်ဒေတာ၊ mainframe policy စနစ်၊ AWS မှီခိုမှုကို ဘုတ်အဖွဲ့က စိုးရိမ်”) — residency၊ integration နှင့် lock-in တစ်ခုစီကို နာမည်တပ်ထားသောပုံစံနှင့် ရိုးသားသောကုန်ကျစရိတ်ဖြင့် ထိစပ်သော တစ်မျက်နှာချဉ်းကပ်နည်း ထုတ်လုပ်ရမည်။ Greenfield ထက် ကန့်သတ်ချက်များက သင့်ကို ပိုစိတ်လှုပ်ရှားစေသောအခါ — အကြောင်းမှာ ကန့်သတ်ချက်များသည် architect များက diagram-ဆွဲသူများထက် ပိုဝင်ငွေရသောနေရာ ဖြစ်သောကြောင့် — အဆင့် ၄ ပြီးပြီ။

အဆင့် ၅ — ARCHITECT ၏ အတတ်ပညာ (လ ၇-၁၂)

ယခုအထိအားလုံးက သင့်ကို ဒီဇိုင်းဆွဲနိုင်စေခဲ့သည်။ ဤအဆင့်က သင့်ကို **architect** တစ်ဦးအဖြစ် အလုပ်လုပ်နိုင်စေမည် — ရာထူးကို အမှန်တကယ် ဖွဲ့စည်းထားသော စာတမ်းများ၊ **review** များ၊ တင်ဆက်မှုများနှင့် သက်သေများ၊ ထို့ပြင် **HR** ကို ကျော်စေမည့် **certification** များနှင့် သင့် **portfolio** ဖြစ်လာမည့် **capstone** သုံးခု။

Module 12 (လ ၇-၈): ပေးအပ်ရမည့်အရာများ — **ADR** များ၊ **diagram** အစုံများနှင့် **review** ကျင်းပခြင်း

Architecture Decision Record (ADR) များ။ ADR ဆိုသည်မှာ သိသာထင်ရှားသော ဆုံးဖြတ်ချက်တစ်ခုကို ဖမ်းယူထားသော တစ်မျက်နှာစာတမ်း — ဘာရွေးခဲ့လဲ၊ ဘာမရွေးခဲ့လဲ၊ ဘာကြောင့်လဲ — ဆုံးဖြတ်ချက်ချသည့်အချိန်တွင် ရေးသည်။ နံပါတ်တပ်သည်၊ **project** ၏ **repository** ထဲတွင် ထာဝရ သိမ်းထားသည်။ အလုပ်ခေါ်စာများက ၎င်းတို့ကို နာမည်တပ်ရသည့်အကြောင်း — နှစ်နှစ်အကြာတွင် တစ်ယောက်ယောက်က “ဒါ ဘာလို့များ **DynamoDB** လဲ” ဟု မေးမည် ဖြစ်ပြီး ADR က စက္ကန့်သုံးဆယ်အတွင်း ဖြေသည် — **context**၊ ကန့်သတ်ချက်များနှင့် ရိုးသားစွာ စဉ်းစားခဲ့သော အခြား ရွေးချယ်စရာများနှင့်တကွ။ ADR ရှိသော အဖွဲ့များသည် ဘာကိုမှ ပြန်မငြင်းကြ။ မရှိသောအဖွဲ့များကမူ နှစ်စဉ် စက်ဝိုင်းလည် ငြင်းခုံကြသည်။ ပုံစံခွက် — အလွတ်ကျက်ပါ —

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.

Capstone တိုင်းတွင် သိသာထင်ရှားသော ဆုံးဖြတ်ချက်တစ်ခုလျှင် **ADR** တစ်စောင်စီ ရေးပါ။ ADR အစစ်များပါသော အင်တာဗျူး **portfolio** သည် ရှားပါးပြီး ကြောက်ခမန်းလိလိ ထိရောက်သည်။

Diagram အစုံ — စနစ်တစ်ခု၊ zoom အဆင့်သုံးဆင့်။ Architecture စာတမ်းပြုမှုအစစ်သည် အစုံ တစ်ခုဖြစ်သည် (**C4 model** က ဤစည်းကမ်းကို လူသိများစေခဲ့သည်) — **context diagram** — စနစ်ကို ကွက်တစ်ကွက်တည်းအဖြစ်၊ ၎င်း၏အသုံးပြုသူများနှင့် ထိတွေ့သောပြင်ပစနစ်များနှင့်အတူ။ အုပ်ချုပ်ရေးမှူး/အသစ်ဝင်လာသူ အမြင် — ခေါင်းဆောင်ပိုင်း အား တင်ဆက်မည့်ပုံ။ **Container diagram** — zoom ဝင်ပါ — အဓိကလည်ပတ်နေသော အစိတ်အပိုင်းများ (**web app**၊ **API**၊ **database**၊ **queue**၊ **cache**) နှင့် ၎င်းတို့ စကားပြောပုံ။ နေ့စဉ် **engineering** အမြင် — သင့်အဆင့် ၁–၃ ပုံများနှင့် အကြမ်းအားဖြင့်တူ။ **Deployment diagram** — အားလုံး ရုပ်ပိုင်းအရ လည်ပတ်ရာနေရာ — **region**၊ **AZ** များ၊ **VPC**၊ **subnet** များ၊ **scaling group** များ။ **Review** များ၊ **security** နှင့် **operations** အတွက် အမြင်။ စနစ်တစ်ခု၊ ပရိသတ်သုံးမျိုး၊ **diagram** သုံးခု — လက်ရှိအတိုင်း ထိန်းသိမ်း၊ ရက်စွဲထိုး၊ ADR များဘေး **version control** ထဲ။ ရှာမတွေ့ သို့မဟုတ် ယုံကြည်မရသော **diagram** သည် မရှိသော **diagram** ပင်။

Well-Architected review ကျင်းပခြင်း။ မဏ္ဍိုင်ခြောက်ရပ်ကို သင်သိပြီ (အဆင့် ၂)။ ဤသည်မှာ အစည်းအဝေးကိုယ်တိုင်။ **မတိုင်မီ:** workload နှင့် scope ကို ရွေးပါ။ diagram အစုံကို လက်ရှိဖြစ်အောင်လုပ်ပါ။ အရာကို လည်ပတ်နေသော လူများ (ဒီဇိုင်းဆွဲသူများသာမက) ကို ဖိတ်ပါ။ ပြီးလျှင် လေသံကို ချမှတ်ပါ — ဤသည်မှာ အဖွဲ့၏အကျိုးအတွက် ကျန်းမာရေးစစ်ဆေးမှုဖြစ်သည်။ ဘယ်သူ့ဖိုင်အတွက်မှ audit မဟုတ်။ **အတွင်း (နေ့တစ်ဝက်):** framework ၏ မေးခွန်းအစုံများဖြင့် မဏ္ဍိုင်များကို လျှောက်သွားပါ (AWS က ထုတ်ဝေထားပြီး console ထဲတွင် လေ့ကျင့်ခန်းတစ်ခုလုံးကို ပုံစံချပေးသော အခမဲ့ Well-Architected Tool လည်း ရှိသည် —

<https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html>)။ အဖြေတစ်ခုစီအတွက် တွေ့ရှိချက်ကို မှတ်ပြီး သွားရင်း ဦးစားပေးစီပါ — nitpick တစ်ရာမဟုတ်၊ အန္တရာယ်မြင့် ကိစ္စလက်တစ်ဆုပ်စာ။ သင့်မေးမြန်းသောလေသံသည်ပင် review ဖြစ်သည် — “payment provider က timeout ဖြစ်ရင် ဘာဖြစ်လဲ နားလည်အောင် ကူပြပါဦး” က တံခါးများဖွင့်ပေးပြီး “timeout တွေ မကိုင်တွယ်ထားဘူးလား” က ဆောင့်ပိတ်သည်။ **ပြီးနောက်:** အစီရင်ခံစာအတို — ထိပ်တန်းတွေ့ရှိချက်များ၊ တစ်ခုစီတွင် အန္တရာယ်၊ အားစိုက်မှုနှင့် ထောက်ခံချက်ပါလျက် — ပြီးလျှင် တိုးတက်ရေးအချက်များသည် ပိုင်ရှင်များပါသော အဖွဲ့၏ backlog အစစ်ထဲ ဆိုက်ရောက်ရမည်။ မဟုတ်လျှင် review သည် ပြဇာတ်သာ ဖြစ်ခဲ့သည်။ လေ့ကျင့်မှု — သင့်ကိုယ်ပိုင် Capstone 1 ဒီဇိုင်းအပေါ် review အပြည့်တစ်ကြိမ် ကျင်းပပါ။ ကိုယ့်ချို့ယွင်းချက်များကို စနစ်တကျ ရှာတွေ့ခြင်းသည် ဤသင်တန်းထဲက အမြန်ဆုံး အတိုးပွားသော ကျွမ်းကျင်မှု ဖြစ်သည်။

လေ့ကျင့်ခန်းများ: (၁) သင့်အဆင့် ၂-၃ journal ဒီဇိုင်းများထဲက ဆုံးဖြတ်ချက်ကြီးငါးခုအတွက် ADR များကို နောက်ကြောင်းပြန်ဖြည့်ပါ — အနည်းဆုံးတစ်ခုကို ယခုအခါ အကြောင်းပြု၍မရတော့ကြောင်း တွေ့ရမည် — ၎င်းသည်ပင် သင်ခန်းစာ။ (၂) ThaiTicket အတွက် diagram သုံးခုအစုံအပြည့် ထုတ်လုပ်ပါ။ (၃) အထက်ပါ အတုပြု review ကို ကျင်းပပြီး တစ်မျက်နှာ တွေ့ရှိချက်အစီရင်ခံစာ ထုတ်ပါ။

Milestone: သူစိမ်းတစ်ဦး (သို့မဟုတ် သူစိမ်းအဖြစ်သရုပ်ဆောင်သော AI) သည် သင့် ThaiTicket အထုပ် — diagram သုံးခု + ADR ငါးစောင် — ကို ကောက်ယူပြီး “ဒါ ဘာလဲ၊ ဘယ်လိုလည်လဲ၊ ဘာလို့ ဒီလိုတည်ဆောက်ထားလဲ” ကို သင်အခန်းထဲမရှိဘဲ မှန်ကန်စွာ ဖြေနိုင်ရမည်။ ထိုအထုပ်သည်ပင် architect အလုပ်၏ ပေးအပ်ရမည့်အရာ ဖြစ်သည်။

Module 13 (လ ၈-၉): လူသားအတတ်ပညာ — တင်ဆက်ခြင်း၊ ကုန်ကျစရိတ်တွက်ခြင်း၊ ကန့်ကွက်မှုကိုင်တွယ်ခြင်းနှင့် လမ်းညွှန်ပြုစုခြင်း

အုပ်ချုပ်ရေးမှူးများနှင့် engineer များအား တင်ဆက်ခြင်း — ဒီဇိုင်းတစ်ခု၊ ဘာသာစကားနှစ်မျိုး။ အုပ်ချုပ်ရေးမှူးများသည် ရလဒ်များကို ဝယ်သည်။ Engineer များသည် ယန္တရားများကို ဝယ်သည်။ အုပ်ချုပ်ရေးမှူးများအား — ဆုံးဖြတ်ချက်နှင့် ဂဏန်းဖြင့် ဦးဆောင်ပါ (“ဒီဒီဇိုင်းက ဖောက်သည်တစ်သန်း ရည်မှန်းချက်ကို အော်ဒါတစ်ခုလျှင် \$1.90 ဖြင့် ထောက်ပံ့နိုင်ပြီး ခင်ဗျားတို့ဆီက လိုအပ်တဲ့ ဆုံးဖြတ်ချက်နှစ်ခုက ဒါတွေပါ”)။ Slide တစ်ချပ်၊ context diagram၊ အန္တရာယ်များကို လုပ်ငန်းအန္တရာယ်များအဖြစ် (ဝင်ငွေ၊ compliance၊ ဂုဏ်သိက္ခာ) ဘောင်ခတ်ပါ။ ပြီးလျှင် “the platform” ဟုပြော၍ရလျှင် “Kubernetes” ဟု ဘယ်တော့မှ မပြောပါနှင့်။ အုပ်ချုပ်ရေးမှူးများ အမြဲမေးသောမေးခွန်းသုံးခုအတွက် ပြင်ဆင်ထားပါ — ဘယ်လောက်ကုန်မလဲ၊ ဘာမှားနိုင်လဲ၊ ပိုဈေးသက်သာတဲ့ရွေးချယ်စရာက ဘာလို့မရလဲ။ Engineer များအား — solution မတိုင်မီ ပြဿနာနှင့် ကန့်သတ်ချက်များဖြင့် ဦးဆောင်ပါ (ပြဿနာကို ခံစားရသော engineer များသည် solution ကို လက်ခံသည်။ စီရင်ချက်ကမ်းခံရသော engineer များကမူ မူအရ ချို့ယွင်းချက် လိုက်ရှာကြသည်)၊ container နှင့်

deployment diagram များကို ပြပါ။ trade-off များနှင့် ပယ်ချခဲ့သောရွေးချယ်စရာများကို ကိုယ်တိုင် နာမည်တပ်ပါ (ယုံကြည်စိတ်ချရမှုသည် သင်ဝန်ခံသည့်အရာမှ လာသည်)။ ပြီးလျှင် ဒီဇိုင်းပြောင်းရန် နေရာအစစ် ချန်ထားပါ — အခန်းထဲက မေးခွန်းများသည် အခမဲ့ review ဖြစ်သည်။ Architect ၏ အဆိုးဆုံးအလေ့အထမှာ ပရီသတ်နှစ်မျိုးလုံးအတွက် deck တစ်ခုတည်း။ အကောင်းဆုံးအလေ့အထမှာ executive summary ကို အရင် ရေးခြင်း — အကြောင်းမှာ ဒီဇိုင်းသည် စာကြောင်းငါးကြောင်းအထိ ဖိသိပ်ခြင်းကို မခံနိုင်လျှင် မပြီးသေးသောကြောင့်တည်း။

Proposal အတွက် ကုန်ကျစရိတ် ခန့်မှန်းခြင်း။ နည်းလမ်း — ဒီဇိုင်းကို ကုန်ကျစရိတ်ရှိသော အစိတ်အပိုင်း ~၁၀ ခုအဖြစ် ခွဲပါ။ တစ်ခုစီကို မျှော်မှန်းဝန်ဖြင့် pricing calculator ထဲ ဈေးတွက်ပါ။ သင့်ယူဆချက်များကို စာဖြင့် ဖော်ပြပါ (တစ်ရက် request များ၊ ဒေတာကြီးထွားမှု၊ egress — ယူဆချက်များသည်ပင် ခန့်မှန်းချက် ဖြစ်သည်။ ၎င်းတို့ပြောင်းလျှင် ဂဏန်းက သိက္ခာရှိရှိ လိုက်ပြောင်းသည်။)။ တည်ငြိမ်သောအပိုင်းများတွင် reservation နည်းဗျူဟာကို အသုံးပြုပါ။ ကြီးထွားမှု scenario များ ထည့်ပါ (ယနေ့ 2x၊ 10x — အုပ်ချုပ်ရေးမှူးများသည် 10x ဂဏန်းကို မှတ်မိကြသည်။)။ ပြီးလျှင် မောင်းနှင်အားများပါသော အပိုင်းအခြား အဖြစ် တင်ပြပါ (“တစ်လ \$55–70k၊ အဓိကမောင်းနှင်အားက egress — ကိရိယာတံက ဒီမှာ”) — တိကျဟန်ဆောင်သော ဂဏန်းတစ်လုံးတည်း ဘယ်တော့မှ မတင်ပါနှင့်။ Migration bubble ရှိလျှင် ထည့်ပါ။ အတည်ပြုချက်ရရန် လျှော့ခန့်မှန်းခြင်းသည် ဂန္ထဝင် junior-architect အပြစ် — project က မှတ်ထားတတ်သည်။

ကန့်ကွက်မှု ကိုင်တွယ်ခြင်း။ ကုန်ကျစရိတ်အတွက် ဖိအားပေးခံရမည် (“ဈေးကြီးလွန်းတယ်” → တြိဂံဆီ ပြန်သွားပါ — “single-AZ ကို လက်ခံရင် \$30k လျှော့နိုင်တယ် — outage ဂဏန်းသင်္ချာက ဒီမှာ။ ခင်ဗျားဆုံးဖြတ်ချက်ပါ။ ကျွန်တော် ရေးမှတ်ထားမယ်” — အပေးအယူကို ထင်ရှား၊ မှတ်တမ်းတင် လုပ်ခြင်းက ကန့်ကွက်မှုအများစုကို သဘောတူညီမှု သို့မဟုတ် သိရှိလက်ခံထားသောအန္တရာယ် နှစ်မျိုးထဲတစ်မျိုးအဖြစ် ပြောင်းပေးသည် — နှစ်ခုလုံး အဆင်ပြေသည်။)။ အကြိုက်အတွက် (“engineer က stack တစ်မျိုး ပိုကြိုက်တယ်” → ၎င်းကို အသံထွက် steel-man လုပ်ပါ။ ထို့နောက် အကြိုက်မဟုတ်၊ စံနှုန်းများဆီ ယူသွားပါ — NFR ကိုက်ညီမှု၊ အဖွဲ့ကျွမ်းကျင်မှုများ၊ ecosystem၊ exit cost — ပြီးလျှင် အမှန်တကယ် ကပ်နေလျှင် အကောင်အထည်ဖော်မည့် engineer ၏အကြိုက်ကို အနိုင်ပေးပါ — commitment ကို ဈေးပေါပေါ ဝယ်လိုက်ခြင်း။)။ အာဏာအတွက် (“CTO ရဲ့ မိတ်ဆွေက X သုံးပါတဲ့” → ထင်မြင်ချက်ကို ထင်မြင်ချက်နှင့် ဘယ်တော့မှ မတိုက်ပါနှင့်။ စံနှုန်းများကို တောင်းပါ။ X ကို ကျန်အားလုံးနှင့် တူညီသော လူသိရှင်ကြားအကဲဖြတ်မှုထဲ ဖြတ်စေပါ။ matrix ကို ယဉ်ယဉ် ကျေးကျေး ဖြေခွင့်ပေးပါ။)။ ပြီးလျှင် သင်မှားသောအခါ — ကျရှုံးမည့်အရာတစ်ခုကို သင်ဒီဇိုင်းဆွဲမိမည်၊ architect တိုင်း ဆွဲမိသည် — လှုပ်ရှားချက်မှာ blameless post-mortem ကို ကိုယ့်ကိုယ်ကို လူသိရှင်ကြား အသုံးပြုပြီး ADR ကို update လုပ်ခြင်း။ ဆယ်စုနှစ်ကြာ ယုံကြည်စိတ်ချမှုကို ၎င်းထက်မြန်အောင် တည်ဆောက်ပေးသောအရာ မရှိပါ။ အလောင်းကောင်ကို ခုခံကာကွယ်ခြင်းထက် မြန်အောင် ဖျက်ဆီးသောအရာလည်း မရှိပါ။

Engineer များကို လမ်းညွှန်ပြုစုခြင်း။ အလုပ်ခေါ်စာစကားစုမှာ “mentor the engineers who build against your standards” ဖြစ်ပြီး လက်တွေ့ပုံစံများမှာ — design review office hours (အမြဲဖွင့်ထားသောတံခါး — လမ်းညွှန်မှုသည် code ပြီးမှမဟုတ်၊ မတိုင်မီ ဖြစ်စေရန်)။ သူတို့၏ဒီဇိုင်းများကို ကြင်နာစွာ review လုပ်ခြင်း — စီရင်ချက်များမတိုင်မီ မေးခွန်းများ၊ ပြီးလျှင် စံနာမည်ကွယ်က ဘာကြောင့် ကို အမြဲပြောပါ (ရှင်းပြထားသော guardrail သည် မဟာမိတ်တစ်ဦးကို စုဆောင်းသည်။ အတင်းချမှတ်သော guardrail သည် ရှောင်လမ်းတစ်ခုကို စုဆောင်းသည်။)။ ဆုံးဖြတ်ချက်အစစ်များကို လွှဲအပ်ခြင်း — ဘေးကင်းရေးကွန်ရက်ပါလျက် (“cache ဒီဇိုင်းကို မင်းပိုင်တယ်။ ကန့်သတ်ချက်တွေက ဒါတွေ။ ငါ review လုပ်မယ်၊ မင်းဆုံးဖြတ်ချက်ကို ငါထောက်ခံမယ်”)။ ပြီးလျှင် လူသိရှင်ကြား သင်ကြားခြင်း — ADR တိုင်း၊ review ရေးသားချက်တိုင်း၊ brown-bag ဆွေးနွေးပွဲတိုင်းက သင့်ကို သင့်ကိုယ်ပိုင်နာရီများထက်ကျော်၍ ချဲ့ထွင်ပေးသည်။ ရိုးသားသော အသက်မွေးဝမ်းကျောင်းအမှန်တရား — တစ်ကိုယ်တော် ပါရမီရှင် architect သည် အထိုက်အလျောက်တွင် ရပ်သွား

သည်။ Engineer ငါးဦးကို ဒီဇိုင်းဆွဲနိုင်သော လုပ်ဖော်ကိုင်ဖက်များအဖြစ် ပြုစုပျိုးထောင်ပေးပြီးသော architect ကမူ practice တစ်ခုလုံးကို ဦးဆောင်သည်။

လေ့ကျင့်ခန်းများ: (၁) ThaiTicket ကို နှစ်ကြိမ် တင်ဆက်ပါ — ၅ မိနစ် executive ဗားရှင်းနှင့် ၂၀ မိနစ် engineering ဗားရှင်း — နှစ်ခုလုံး အသံသွင်းပြီး ပထမတစ်ခုထဲ ဝေါဟာရများ ယိုစိမ့်နေခြင်းကို နားထောင်ပါ။ (၂) ရေးသားထားသော ကုန်ကျစရိတ် proposal အပြည့်အစုံ (ယူဆချက်များ၊ အပိုင်းအခြား၊ မောင်းနှင်အားများ၊ 2x/10x scenario များ) ထုတ်လုပ်ပါ။ (၃) AI နှင့် ကန့်ကွက်မှုပြဇာတ် — သုံးပွဲ — CFO ထံမှ ကုန်ကျစရိတ်တိုက်ခိုက်မှု၊ senior engineer ထံမှ stack အကြိုက်၊ CTO-မိတ်ဆွေ အာဏာကစားကွက် — ဘာအလုပ်ဖြစ်ခဲ့သည်ကို journal ထဲမှတ်ပါ။ (၄) Junior တစ်ဦး၏ဒီဇိုင်းကို စာဖြင့် review လုပ်ပါ (AI က ချို့ယွင်းချက်ရှိတစ်ခု ထုတ်ပေးနိုင်သည်) — မေးခွန်းများအရင်း၊ ပြီးလျှင် AI ကို သင့်လေသံအမှတ်ပေးခိုင်းပါ။

Milestone: နှစ်ပွဲဆက် — executive pitch ကို တင်ဆက်ပြီး ရန်လိုသော်လည်း တရားမျှတသော ရောနှော Q&A မိနစ် နှစ်ဆယ် (AI panel: CFO တစ်ဦး၊ သံသယရှိသော engineer တစ်ဦး) ကို ဝေါဟာရလွဲမှား၊ ခုခံစိတ် သို့မဟုတ် မရေးမှတ် ထားသော trade-off တစ်ခုမှမပါဘဲ ကျော်ဖြတ်ပါ။

Module 14 (လ ၉-၁၂): Certification များနှင့် capstone သုံးခု

Certification လမ်းကြောင်း — ရိုးသားသောအချိန်ဇယားများနှင့်။ Certification များက သင့်ကို architect ဖြစ်စေသည် မဟုတ် — ရှေ့က module ၁၃ ခုက ဖြစ်စေသည် — သို့သော် ၎င်းတို့က သင့်ကို အင်တာဗျူးများဆီ ရောက်စေပြီး ပြင်ဆင်ရင်း provider ၏ဝန်ဆောင်မှုများကို သင့်လက်ချောင်းထဲ ကြိုးသွယ်ပေးသည်။ လေ့ကားထစ်များ၊ တစ်နေ့တစ်နာရီ အရှိန်ဖြင့် — **AWS Certified Solutions Architect – Associate (SAA-C03)** — ၂-၃ လ; အလုပ်ခေါ်စာများတွင် အတောင်းဆိုအခံရဆုံး architect အသိအမှတ်ပြုလက်မှတ်ဖြစ်ပြီး အဆင့် ၁-၃ ပြီးနောက် ၎င်း၏အများစုမှာ ဝန်ဆောင်မှုအမည်များ တွဲကပ်ထားသော ပြန်လှန်လေ့လာမှုသဖွယ် ခံစားရမည်; ဤခုကို ပထမဆုံး ဖြေပါ။ **AWS Certified Solutions Architect – Professional (SAP-C02)** — Associate ပြီးနောက် ၄-၈ လ; scenario အခြေပြု၊ တကယ်ခက်ခဲပြီး ဤနယ်ပယ်တွင် resume စာကြောင်း အားအကောင်းဆုံး တစ်ကြောင်းတည်း; အဆင့် ၄ ၏ migration နှင့် multi-account အကြောင်းအရာသည် ၎င်း၏သင်ရိုး တစ်ဝက်ဖြစ်သည်။ **Azure AZ-305 (Azure Solutions Architect Expert)** — သင့်ဈေးကွက်သည် Microsoft အလေးသာလျှင် ထပ်ထည့်ပါ (enterprise ဈေးကွက်အများစုမှာ အနည်းဆုံး နှစ်ဘာသာသုံး ဖြစ်သည်); AWS အသိပညာ များစွာ ကူးပြောင်းအသုံးဝင်သဖြင့် ၂-၃ လ မျှော်လင့်ပါ (<https://learn.microsoft.com/en-us/credentials/certifications/azure-solutions-architect/> မှ စတင်ပါ)။ **TOGAF Foundation** — ရွေးချယ်နိုင်၊ ၂-၄ ပတ်; enterprise ကြီးအချို့က တောင်းသည်; ၎င်းသည် cloud ကျွမ်းကျင်မှုထက် enterprise-architecture နည်းစနစ် နှင့် ဝေါဟာရကို အသိအမှတ်ပြုပေးခြင်း ဖြစ်သည်။ ဤသင်တန်းအတွက် အစဉ်လိုက် — လ ၇-၈ နှင့်အပြိုင် SAA လေ့လာပါ။ စာမေးပွဲကို ~လ ၈ တွင် ဖြေပါ; ထို့နောက် လ ၁၂ အထိ SA Pro (နည်းပညာနက်ရှိုင်းသောလမ်း) သို့မဟုတ် AZ-305 (ကျယ်ပြန့်မှုလမ်း) တစ်ခုခုကို ဆက်လုပ်ပြီး SA Pro ကို ရိုးသားသောအရှိန်ဖြင့် လ ၁၄-၁၆ တွင် အပြီးသတ်ပါ။

Capstone သုံးခု။ တစ်ခုစီသည် ပြည့်စုံသော portfolio-အဆင့် အထုပ်တစ်ခု ဖြစ်သည် — အဆင့်သုံးဆင့် diagram အစုံ၊ ADR ၅ စောင်အထက်၊ ယူဆချက်များပါသော ကုန်ကျစရိတ်ခန့်မှန်းချက်၊ နှင့် တွေ့ရှိချက်များပါသော ကိုယ်တိုင်ကျင်းပ

သည့် Well-Architected review။ တစ်ခုလျှင် ၃-၄ ပတ် သုံးပါ။ ဤအရာသုံးခုနှင့် သင့် journal သည်ပင် “end-to-end solution design” အတွက် သင့်အင်တာဗျူး သက်သေ ဖြစ်သည်။

Capstone 1 — greenfield: အသုံးပြုသူ ၁ သန်းအတွက် ထိုင်း e-commerce platform။ အမှာစာ — marketplace၊ စာရင်းသွင်းအသုံးပြုသူ ၁ သန်း၊ flash-sale အမြင့်ဆုံးချိန်တွင် တစ်ပြိုင်နက် ၅ သောင်း၊ mobile-first၊ PromptPay ပုံစံ ငွေပေးချေမှု ပေါင်းစည်းမှု၊ PDPA နှင့်ကိုက်ညီရမည်၊ ရရှိနိုင်မှု 99.9%၊ seed-stage ဘတ်ဂျက်သတိရှိရမည်။ မဖြစ်မနေပါရမည် — ဆင့်ကဲပြောင်းလဲရေးလမ်းကြောင်းပါသော ပုံစံရွေးချယ်မှု၊ caching နည်းဗျူဟာ၊ စကားပြောခန်းအဖြစ် ရေးထားသော RTO/RPO စကားဝိုင်း၊ unit-economics ခန့်မှန်းချက်၊ ကိုယ်ရေးကိုယ်တာဒေတာအတွက် data-flow မြေပုံ။

Capstone 2 — brownfield: on-prem server ၄၀ လုံးရှိ ကုမ္ပဏီတစ်ခုကို migrate လုပ်ခြင်း။ အမှာစာ — ထိုင်း logistics ကုမ္ပဏီတစ်ခု၊ server ၄၀ (Oracle ပေါ်က ERP၊ ၁၂ နှစ်သား၊ ဂိုဒေါင်စီမံခန့်ခွဲမှု app၊ file share များ၊ AD၊ ဌာနအလိုက် app အမျိုးမျိုး — inventory အပြည့်ကို AI ဖြင့် ထုတ်ပြီး ခိုင်မြဲအောင် သတ်မှတ်ထားပါ)၊ data center ငှားရမ်းစာချုပ် ၁၄ လအတွင်း ကုန်ဆုံးမည်၊ ဘုတ်အဖွဲ့က ထွက်ချင်နေသည်။ မဖြစ်မနေပါရမည် — 7-R inventory ဇယားအပြည့်၊ dependency အကြောင်းပြချက်ပါသော သုံး-wave အစီအစဉ်၊ landing-zone ဒီဇိုင်း၊ migration-bubble ကုန်ကျစရိတ် အချိန်ဇယား၊ နှင့် CFO အမှာစာ။

Capstone 3 — ခက်ခဲသောခု: fintech တစ်ခုအတွက် multi-region DR။ အမှာစာ — ledger အတွက် RTO ≤ ၁၅ မိနစ် / RPO ≈ 0 ဟု ထိန်းချုပ်သတ်မှတ်ခံထားသော ငွေပေးချေမှုကုမ္ပဏီတစ်ခု၊ ဖောက်သည်ဒေတာအပေါ် data-residency ကန့်သတ်ချက်များနှင့် နှစ်စဉ် ထိန်းချုပ်သူ မျက်မြင်တွေ့ failover စမ်းသပ်မှုတစ်ခုပါ။ မဖြစ်မနေပါရမည် — ကုန်ကျစရိတ်များပါလျက် active-passive နှင့် active-active နှိုင်းယှဉ်ဆန်းစစ်ချက်၊ residency မြေပုံပါသော replication ဒီဇိုင်း၊ failover runbook၊ နှင့် ဇာတ်တိုက်အစီအစဉ်။

ကိုယ်တိုင်အကဲဖြတ်ရန် rubric (capstone တစ်ခုစီအပေါ် သင့် journal ထဲတွင် ရိုးသားစွာ အသုံးပြုပါ)။

ကဏ္ဍ	၁ — မရသေး	၂ — ခိုင်မာ	၅ — ဒီလူကို ခန့်ပါ
လိုအပ်ချက်များ	Solution ဆီ ခုန်ကျော်သွားသည်	NFR များကို ဖော်ပြပြီး ဒီဇိုင်းရွေးချယ်မှုများနှင့် ချိတ်ဆက်ပြထားသည်	အပေးအယူကြံဂံပေါ် နေရာချထား၊ ပဋိပက္ခများကို “လုပ်ငန်း” ဆီ တင်ပြ၊ ဆုံးဖြတ်ချက်များ ထုတ်ယူပြီး
ဒီဇိုင်းအရည်အသွေး	SPOF များ ကျန်နေ၊ ပုံစံ မကိုက်ညီ	ပုံစံမှန်၊ multi-AZ၊ ကျရှုံးမှုများ ကိုင်တွယ်ထား	ဘယ်အခါ-မသုံးရ၊ အကြောင်းပြချက် ပြထား၊ ဆင့်ကဲလမ်းကြောင်း၊ လိုအပ်ချက်ပြည့်မီသော အရိုးရှင်းဆုံးဒီဇိုင်း
မဏ္ဍိုင်ခြောက်ရပ်	မဏ္ဍိုင်များကို လျစ်လျူရှု	မဏ္ဍိုင်တစ်ခုစီကို မြင်သာအောင် ကိုင်တွယ်ထား	ကိုယ်တိုင်ကျင်းပသော review က ချို့ယွင်းချက်

ကဏ္ဍ	၁ — မရသေး	၃ — ခိုင်မာ	၅ — ဒီလူကို ခန့်ပါ
			အစစ်များ တွေ့ရှိ — ပြီးလျှင် ဒီဇိုင်းကို တုံ့ပြန်ပြင်ဆင်ပြီး
ကုန်ကျစရိတ်	ဂဏန်းမရှိ	အချက်အလိုက်ခွဲထားသော ခန့်မှန်းချက်၊ ယူဆချက်များ ဖော်ပြထား	မောင်းနှင်အားများပါသော အပိုင်းအခြား၊ unit economics၊ 10× scenario၊ reservation နည်းဗျူဟာ
စာတမ်းပြုစုမှု	Diagram သက်သက်	အဆင့်သုံးဆင့်အစုံ + ADR များ၊ မှတ်သားပုံမှန်ကန်	သူစိမ်းတစ်ဦးက အထုပ်တစ်ခုတည်းမှ “ဘာလဲ/ဘယ်လိုလဲ/ဘာကြောင့်လဲ” ကို ဖြေနိုင်
ဆက်သွယ်ပြောဆိုမှု	ပရိသတ်အားလုံးအတွက် အရာတစ်ခုတည်း	Executive နှင့် engineer ဗားရှင်း နှစ်ခုစလုံး ရှိ	ခံနိုင်ရည်ရှိသော စာကြောင်း၊ ငါးကြောင်း အကျဉ်းချုပ်၊ ကန့်ကွက်မှုများကို စာဖြင့် ကြိုဖြေထား

Capstone သုံးခုလုံးတွင် ကဏ္ဍတိုင်းကို ၄ အထက် ရအောင် — ရသည်အထိ ပြန်ပြင်ရေးလျက် — လုပ်နိုင်လျှင် သင်သည် ဤသင်တန်းကို ပြီးမြောက်ပြီ ဖြစ်သည်။

Architect အင်တာဗျူး မေးခွန်း ဆယ်ခု (အားကောင်းသောအဖြေများ နှင့်)

ဤမေးခွန်းများကို အသံထွက် လေ့ကျင့်ပါ၊ အားသာချက်သည် အဖြေတစ်ခုစီ၏ ဖွဲ့စည်းပုံ ထဲတွင် ရှိပြီး ထိုဖွဲ့စည်းပုံကို သင် ယခုပိုင်ဆိုင်ပြီ ဖြစ်သည်။

၁။ “Design a URL shortener / ticketing site / photo app for a million users.” (အသုံးပြုသူတစ်သန်း အတွက် URL shortener / လက်မှတ်ရောင်းဆိုက် / photo app တစ်ခု ဒီဇိုင်းဆွဲပါ။) အားကောင်းသောပုံသဏ္ဌာန် — လိုအပ်ချက်များကို အသံထွက် အရင်မေးပါ (“Reads က writes ထက် လွှမ်းမိုးသလား။ ရရှိနိုင်မှု ရည်မှန်းချက်က ဘယ်လောက်လဲ။ ဘတ်ဂျက်က။”) → တြိဂံပေါ် နေရာချပါ → multi-AZ၊ cache၊ CDN ပါသော three-tier သို့မဟုတ် serverless-first ဘောင် → မမေးခင် အပေးအယူများကို ကိုယ်တိုင်နာမည်တပ်ပါ → ကုန်ကျစရိတ် အဆက်အသွယ်အဆင့်နှင့်

10× ဆင့်ကဲလမ်းကြောင်းဖြင့် အဆုံးသတ်ပါ။ အင်တာဗျူးသူများသည် သင်ဆွဲသော ကွက်များကို မဟုတ်ဘဲ သင်မေးသော မေးခွန်းများ ကို အောင်မှတ်ပေးသည်။

၂။ “When would you choose NoSQL over a relational database?” (Relational database အစား NoSQL ကို ဘယ်အခါ ရွေးမလဲ။) “ကျွန်တော် SQL ကို ပုံသေထားပါတယ် — မှန်ကန်မှုအာမခံချက်များနှင့် အနှံ့အပြားရှိ သောကျွမ်းကျင်မှုကြောင့် — နာမည်တပ်နိုင်သောအကြောင်းရင်းတစ်ခုက ကျော်လွှားသည်အထိ — အလွန်အမင်း အလျားလိုက်ခဲ့သွင်းမှု၊ ပြောင်းလွယ်သော schema၊ သို့မဟုတ် တစ်လုံးဂဏန်း မီလီစက္ကန့် key-value ရယူမှု။ ထိုအခါ access pattern နှင့်ကိုက်အောင် NoSQL အမျိုးအစားကို ရွေးပြီး ကျွန်တော်တို့ စွန့်လွှတ်လိုက်ရသည်များ — join များ နှင့် consistency semantics အချို့ — ကိုပါ ထည့်ရေးထားသော ADR ကို ရေးပါတယ်။”

၃။ “Explain RTO and RPO, and how they drive design.” (RTO နှင့် RPO ကို ရှင်းပြပါ။ ဒီဇိုင်းကို ဘယ်လို မောင်းနှင်သလဲ။) နှစ်ခုလုံးကို ပြတ်ပြတ်သားသား အဓိပ္ပာယ်ဖွင့်ပါ → “၎င်းတို့သည် ကိန်းဂဏန်းပွားများသော ဈေးတံဆိပ်များ ပါသည့် လုပ်ငန်းဆုံးဖြတ်ချက်များ ဖြစ်ပါတယ်” → လေ့ကျင့်ထမ်းများ — ညစဉ် backup → စဉ်ဆက်မပြတ် replication → warm standby → active-active၊ ထစ်တိုင်းတွင် ကုန်ကျစရိတ် တက်လျက် → “ကျွန်တော့်အလုပ်က လုပ်ငန်းကို သိသိသာသာ ရွေးချယ်စေပြီး ထိုဂဏန်းအတိုင်း အတိအကျ ဒီဇိုင်းဆွဲကာ ဇာတ်တိုက်ရန်ပါပဲ — မစမ်းသပ်ရသေးသော failover ဟာ မျှော်လင့်ချက်သာ ဖြစ်လို့ပါ။”

၄။ “Monolith or microservices?” (Monolith လား၊ microservices လား။) Module 8 ၏ ဆုံးဖြတ်ချက် အတိအကျ — modular-monolith ဖြင့် စပါ; အဖွဲ့ချဲ့ထွင်မှု သို့မဟုတ် ဝန်ဆောင်မှု နာကျင်မှု အမှန်တကယ်ရောက်လာမှ ခွဲထုတ်ပါ; microservices သည် distributed-systems အခွန်ကောက်ခံသော အဖွဲ့ချဲ့ထွင်ရေးကိရိယာ ဖြစ်သည်။ “ညံ့သော distributed system တစ်ခုထက် ကောင်းသော monolith တစ်ခုကို လည်ပတ်ရတာ ကျွန်တော် ပိုသဘောကျပါတယ်” ဆိုလျှင် အမှတ်ပို။

၅။ “How do you handle a large cloud bill / cost optimization?” (Cloud ဘောလ်ကြီးတစ်ခု / ကုန်ကျစရိတ် အကောင်းဆုံးဖြစ်အောင်လုပ်ခြင်းကို ဘယ်လိုကိုင်တွယ်မလဲ။) “မြင်သာမှု အရင်း — tagging နှင့် showback; ထို့နောက် အားစိုက်မှုအလိုက် အစဉ်လိုက် ရိတ်သိမ်းခြင်း — သေနေသော resource များ ဖျက်၊ တိုင်းတာ ချက်များအရ rightsize လုပ်၊ storage ကို အလွှာခွဲ၊ non-prod များကို အချိန်ဇယားဖြင့် အိပ်စေ၊ ထို့နောက် တည်ငြိမ် သော baseline ကို reserve လုပ်။ ပြီးတော့ ကျွန်တော် စုစုပေါင်းမဟုတ်ဘဲ unit economics ကို အစီရင်ခံပါတယ် — အော်ဒါတစ်ခုလျှင် ကုန်ကျစရိတ် ကျဆင်းနေစဉ် ဘောလ်ကြီးလာခြင်းဟာ အောင်မြင်မှုဖြစ်ပြီး ပြဿနာ မဟုတ်ပါဘူး။”

၆။ “How would you migrate a legacy on-prem application?” (Legacy on-prem application တစ်ခု ကို ဘယ်လို migrate လုပ်မလဲ။) “မရွှေ့ခင် အကဲဖြတ်ပါ — inventory၊ dependency များနှင့် 7 R များထဲမှ workload တစ်ခုလျှင် R တစ်ခု; wave တစ်ခုတိုင်မီ landing zone ကို တည်ဆောက်ပါ; wave များကို လွယ်သည်အရင်၊ ဇာတ်တိုက်ပြီးသား cutover နှင့် rollback အစီအစဉ်များနှင့်အတူ; ပြီးတော့ အဖိုးတန်ရတနာ database အတွက် လုပ်ငန်းက လက်မှတ်ထိုးထားသော ရပ်တန့်ချိန်နှင့် ကိုက်အောင် အရွယ်ချိန်ထားသော replication အခြေပြု cutover။ ထို့ပြင် — migration-bubble သတိပေးချက်ကို ရိုးရိုးသားသား ကြိုပြောပါ။”

၇။ “How do you secure a cloud architecture?” (Cloud architecture တစ်ခုကို ဘယ်လို လုံခြုံအောင်လုပ်မ လဲ။) အလွှာများကို လျှောက်သွားပါ — identity (least privilege၊ MFA၊ နေ့စဉ် root မသုံး) → network (private

subnet များ၊ segmentation၊ တိုက်ခိုက်ခံရနိုင်သောမျက်နှာပြင် အနည်းဆုံး) → data (နားနေချိန်/သွားလာချိန် encryption၊ classification၊ residency) → စောင့်ကြည့်ရှာဖွေခြင်း (audit log များ၊ သတိပေးချက်များ) → “ပြီးတော့ နောက်မှတပ်ဆင်ခြင်းမဟုတ်ဘဲ ဒီဇိုင်းအားဖြင့် — အဈောအပေါ်ဆုံး လုံခြုံရေးထိန်းချုပ်မှုက အန္တရာယ်ကို လုံးဝဖယ်ရှားပေးသော architecture ဆုံးဖြတ်ချက်ပါပဲ — ကတ်ဒေတာအကြမ်းကို လုံးဝမကိုင်တာမျိုးပေါ့။”

၈။ “Tell me about a design decision you got wrong.” (မှားခဲ့သော ဒီဇိုင်းဆုံးဖြတ်ချက်တစ်ခုအကြောင်း ပြောပြပါ။) သူတို့စမ်းသပ်နေသည်မှာ သမိုင်းမဟုတ်၊ တစ်ကိုယ်ကောင်းစိတ်။ ပုံသဏ္ဌာန် — တကယ့်ဥပမာ (capstone) → သင်ဘာယုံကြည်ခဲ့သည် → အဖြစ်မှန်က ဘာပြောခဲ့သည် → post-mortem၊ update လုပ်ပြီးသော ADR၊ ယခုအခါ သင် စစ်ဆေးလေ့ရှိသောပုံစံ။ ဤအဖြေကို မထုတ်နိုင်သော architect သည် review မခံဖူးသေးသော architect ဖြစ်သည်။

၉။ “How do you explain a complex technical decision to a non-technical executive?” (ရှုပ်ထွေးသော နည်းပညာဆုံးဖြတ်ချက်တစ်ခုကို နည်းပညာမဲ့ အုပ်ချုပ်ရေးမှူးတစ်ဦးအား ဘယ်လိုရှင်းပြမလဲ။) “ဆုံးဖြတ်ချက် နှင့် လုပ်ငန်းဂဏန်း အရင်၊ ယန္တရားကို တောင်းမှသာ၊ container မဟုတ်ဘဲ context diagram၊ အန္တရာယ်များကို ဝင်ငွေ-compliance-ဂုဏ်သိက္ခာ အသုံးအနှုန်းများဖြင့်၊ ပြီးတော့ သူတို့ဆီက ကျွန်တော်လိုအပ်တဲ့ ဆုံးဖြတ်ချက်နှစ်ခုကို ဈေးတံဆိပ် ပါ ရွေးချယ်စရာများအဖြစ် ရေးထားပြီး ယူသွားပါတယ် — အုပ်ချုပ်ရေးမှူးများဟာ ရွေးချယ်စရာများကြားက ဆုံးဖြတ်ကြပါ တယ်၊ လျှို့ဝှက်ဆန်းကြယ်မှုတွေကို အတည်ပြုပေးကြတာ မဟုတ်ပါဘူး။”

၁၀။ “An engineer strongly disagrees with your design. What do you do?” (Engineer တစ်ဦးက သင့်ဒီဇိုင်းကို ပြင်းပြင်းထန်ထန် သဘောမတူဘူး။ ဘာလုပ်မလဲ။) “ပထမဆုံး သူ့အငြင်းအခုံကို ကျွန်တော်ကိုယ်တိုင် အသံထွက် steel-man လုပ်ပါတယ် — သူမှားချင်မှမှားမှား၊ ကျွန်တော့်ဒီဇိုင်းကို ပြောင်းစေတဲ့ review ဟာ review က အလုပ်လုပ်နေတာပါ။ တကယ် ကပ်နေရင် သက်ကြီးဝါကြီးမှုမဟုတ်ဘဲ စံနှုန်းများ (NFR ကိုက်ညီမှု၊ အဖွဲ့ကျွမ်းကျင်မှု၊ exit cost) က ဆုံးဖြတ်ပါတယ် — ပြီးတော့ သရေကျရင် အကောင်အထည်ဖော်မည့်သူ၏ အကြိုက်ကို ကျွန်တော် အနိုင်ပေးပါ တယ်၊ အဲဒီဈေးနှုန်းနဲ့ဆို commitment က ပေါပါတယ်။ ဘယ်လိုပဲဖြစ်ဖြစ် ဆုံးဖြတ်ချက်ရော ပယ်ချခဲ့သောရွေးချယ်စရာပါ ADR ထဲ ဝင်သွားလို့ ကျွန်တော်တို့ ဘယ်တော့မှ နှစ်ခါ မငြင်းခုံကြပါဘူး။”

တစ်မျက်နှာ သင်ရိုးညွှန်းတမ်း

အချိန်	Module	အာရုံစိုက်ရန်	ပြင်ပသက်သေ
အပတ်စဉ် ၁-၂	1	Cloud ပြန်လှန်လေ့လာမှု၊ အပေးအယူကြိဂံ၊ NFR များ	Design Journal စတင်
အပတ်စဉ် ၃-၄	2	Compute/storage/database/network ကို ဆုံးဖြတ်ချက်များအဖြစ်	—

အချိန်	Module	အာရုံစိုက်ရန်	ပြင်ပသက်သေ
အပတ်စဉ် ၅-၆	3	Diagram ဖတ်ခြင်း၊ ဆွဲခြင်း; three-tier ကျွမ်းကျင်မှု	၁၅ မိနစ် whiteboard milestone
အပတ်စဉ် ၇-၈	4	Reliability: multi-AZ၊ auto-scaling၊ RTO/RPO	—
အပတ်စဉ် ၉-၁၀	5	Security: least privilege၊ zero trust၊ segmentation	—
အပတ်စဉ် ၁၁-၁၂	6	Performance နှင့် cost: caching၊ CDN၊ reservation များ၊ unit economics	Calculator ဖြင့် ဈေးတွက်ထားသောဒီဇိုင်း
အပတ်စဉ် ၁၃-၁၄	7	Ops excellence နှင့် sustainability; မဏ္ဍိုင်ခြောက်ရပ် လေ့ကျင့်ခန်း	စမ်းသပ် မဏ္ဍိုင်ခြောက်ရပ် review
အပတ်စဉ် ၁၅-၁၇	8	ပုံစံများ: monolith/microservices၊ event-driven၊ serverless	—
အပတ်စဉ် ၁၈-၂၀	9	ပုံစံများ: data lake/warehouse၊ multi-region၊ hybrid	စာရင်း စိန်ခေါ်မှုကြီး
အပတ်စဉ် ၂၁-၂၃	10	Migration: 7 R များ၊ wave များ၊ landing zone များ	စက္ကူပေါ် migration
အပတ်စဉ် ၂၄-၂၆	11	ကန့်သတ်ချက်များ: PDPA/GDPR၊ legacy၊ lock-in စီးပွားရေး	ကန့်သတ်ချက် စိန်ခေါ်မှုကြီး
လ ၇-၈	12	ADR များ၊ diagram အစုံများ၊ Well-Architected review ကျင်းပခြင်း	ThaiTicket အထုပ်; SAA-C03 စာမေးပွဲ (ပြင်ဆင်ချိန် ၂-၃ လ)
လ ၈-၉	13	တင်ဆက်ခြင်း၊ proposal ကုန်ကျစရိတ် တွက်ခြင်း၊ ကန့်ကွက်မှု၊ လမ်းညွှန်ပြုစုခြင်း	အသံသွင်းထားသော နှစ်ပွဲဆက်
လ ၉-၁၂	14	Capstone ၁-၃	Portfolio ပြည့်စုံ; SA Pro (၄-၈ လ) သို့မဟုတ် AZ-305 လုပ်ဆောင်ဆဲ

သင့်ဆရာထံမှ နိဂုံးချုပ်စကား။ ၂၆ ပတ်အကြာတွင် သင် ဒီဇိုင်းဆွဲ နိုင်ခဲ့ပြီ; ၁၂ လအကြာတွင် သင် အတတ်ပညာကို လက်တွေ့ကျင့်သုံး နိုင်ပြီ — ကွာခြားချက်ရှိပြီး ထိုကွာဟချက်ကို ပိတ်ရန်ပင် ဤသင်တန်းကို တည်ဆောက်ခဲ့ခြင်း ဖြစ်သည်။ ဓလေ့များသည် ယခုအခါ အသက်မွေးဝမ်းကျောင်းလမ်းပင် ဖြစ်လာပြီ — မငြင်းခုံမီ ပုံဆွဲပါ။ ဆုံးဖြတ်သောနေ့တွင်ပင် ADR ကို ရေးပါ။ သင်အဆိုပြုသည်များကို ဈေးတွက်ပါ။ ဘယ်သူမှမမေးမီ အပေးအယူကို ကိုယ်တိုင်နာမည်တပ်ပါ။ ပြီးလျှင် သင့် စံနှုန်းများသည် သင်အခန်းထဲရှိနေခြင်းထက် ပိုရှည်ကြာ အသက်ရှင်သည်အထိ သင့်ပတ်ဝန်းကျင်ရှိ engineer များကို ပြုစု ပျိုးထောင်ပါ။ Architecture သည် အသက်ကြီးလာသည်နှင့်အမျှ ပိုကောင်း လာသော ရှားပါးသည့် နည်းပညာအလုပ် ဖြစ်သည် — အကြောင်းမှာ ၎င်း၏ကုန်ကြမ်းသည် အကဲဖြတ်ဉာဏ်ဖြစ်ပြီး အကဲဖြတ်ဉာဏ်သည် အတိုးထပ်တတ် သောကြောင့်တည်း။ Journal ကို ဆက်ရေးပါ။ ယခုမှ ဒီဇိုင်းလေးဆယ်အကြာတွင် သင်သည် ဤသင်တန်းကို မလိုတော့ဘဲ — ၎င်းကို ပြင်ဆင်ပေးနေမည့်သူ ဖြစ်နေပါလိမ့်မည်။

ရင်းမြစ်များ

လိုအပ်ချက်များ ဆက်စပ်မြေပုံဆွဲရာတွင် အသုံးပြုခဲ့သော အလုပ်ခေါ်စာများနှင့် ရာထူးအဓိပ္ပာယ်ဖွင့်ဆိုချက်များ (၂၀၂၆ ခုနှစ် ဩဂုတ်လတွင် ရယူ)၊ [KORE1 Cloud Architect job description template](#) · [4 Corner Resources Cloud Architect job description](#) · Microsoft ၏ [Solution Architect responsibilities in the Azure Well-Architected Framework](#) · Arpio ၏ [Pre-Sales Solutions Architect – AWS posting](#) · Intel ၏ [Solution Architect – Enterprise posting \(via Built In\)](#) · Halliburton ၏ [Cloud Domain Architect posting](#)။ Certification ပြင်ဆင်ချိန် ခန့်မှန်းချက်များ၊ [CBT Nuggets' SAA-C03 study-time survey](#) နှင့် [Whizlabs' SAP-C02 preparation guidance](#); certification အဓိပ္ပာယ်ဖွင့်ဆိုချက်များကို [Microsoft Learn \(AZ-305\)](#) နှင့် [AWS Well-Architected Framework](#) မှ ရယူထားသည်။ B4LCILC စီးရီးရှိ [The Cloud Leader Course](#) ၏ တွဲဖက် စာစောင် ဖြစ်သည်။

သင်ခန်းစာအသစ်များ — ၂၀၂၆ ခုနှစ် ဩဂုတ်လ ထုတ်ဝေမှု

၂၀၂၆ ခုနှစ် ဩဂုတ်လတွင် [b4lcilc.org](#) ရှိ *interactive* သင်တန်းသုံးခု — *Cloud ခေါင်းဆောင် (Cloud Leader)*၊ *Cloud Engineer* နှင့် *Cloud Architect* သင်တန်းများ — ထဲသို့ သင်ခန်းစာအသစ်များ ထပ်မံဖြည့်သွင်းခဲ့သည်။ ဤ နောက်ဆက်တွဲသည် ထိုသင်ခန်းစာအသစ်များကို ဤစာအုပ်၏ ကျန်အပိုင်းများနှင့် လေသံတစ်မျိုးတည်းဖြင့် သစ္စာရှိရှိ ချုံ့ ရေးသားပေးထားကာ ဖတ်ရှုရန်ထုတ်ဝေမှု (*reading edition*) ကို နောက်ဆုံးအခြေအနေနှင့် ကိုက်ညီအောင် လုပ်ဆောင် ပေးသည်။

သင်လက်ထဲတွင် ကိုင်ထားသည့်အရာအကြောင်း စကားတစ်ခွန်း။ ဤစာမျက်နှာများသည် ဖတ်ရှုရန်ထုတ်ဝေမှု ဖြစ်သည် — အယူအဆများ၊ ဥပမာတင်စားချက်များနှင့် အသံထွက်၍ ပြောထိုက်သော စာကြောင်းများ။ အွန်လိုင်းရှိ *interactive* ဗားရှင်း

အပြည့်အစုံများတွင် များစွာ ပိုပါဝင်သည်။ သင်ခန်းစာအသစ်တိုင်းတွင် ရှေ့ဆက်ခွင့်မပြုမီ သင်၏နားလည်မှုကို စစ်ဆေးပေးသော **quiz** များ၊ လက်တွေ့လုပ်ဆောင်ရသော **lab** များ (အစစ်အတိုင်း အလုပ်လုပ်သော်လည်း ငွေကြေး လုံးဝမကုန်ကျနိုင်သည့် လေ့ကျင့်ရေး **terminal** အပါအဝင်)၊ ဗီဒီယိုများ၊ ဝေါဟာရ လေ့ကျင့်ခန်းများနှင့် — **capstone** (နိဂုံးချုပ် စုစည်းလေ့ကျင့်ခန်း) အကြောင်းအရာများအတွက် — လိုအပ်ချက် **checklist** များနှင့် နမူနာဖြေဆိုချက်များပါသော လမ်းညွှန် **design** လေ့ကျင့်ခန်းများ ပါဝင်သည်။ အယူအဆတစ်ခုစီ၏ ပုံသဏ္ဌာန်ကို ဦးနှောက်ထဲ ရောက်အောင် ဤချို့ချေးသားချက်များကို ဖတ်ပါ။ ထို့နောက် အခွင့်သာသည့်အခါ ဗဟုသုတ စွဲမြဲသွားစေရန် **interactive** ဗားရှင်းများကို ထိုင်ဖြေပါ။

ထပ်ဖြည့်သင်ခန်းစာများကို သုံးစွဲခဲ့ထားသည်။ **Cloud ခေါင်းဆောင်** သင်တန်းအတွက် — **terminal** ကို ညင်သာစွာ မိတ်ဆက်ပေးခြင်း၊ ထိုင်းနိုင်ငံ၏ ကိုယ်ပိုင် **privacy** ဥပဒေ၊ **AI** အဆိုပြုချက်များနှင့် **AI** ငွေတောင်းခံလွှာများအတွက် ခေါင်းဆောင်တစ်ဦး၏ လမ်းညွှန်၊ နှင့် သင်တန်းတစ်ခုလုံးကို စာမျက်နှာတစ်မျက်နှာပေါ် စုစည်းပေးသော **capstone** လေ့ကျင့်ခန်း။ **Cloud Engineer** သင်တန်းအတွက် — **serverless computing** — လည်ပတ်ရေး **script** ကလေးများကို **run** ရသည့်စရိတ် လုံးဝမရှိသလောက်ဖြစ်သော **function** များအဖြစ် ပြောင်းလဲပေးသည့် ပုံစံ။ **Cloud Architect** သင်တန်းအတွက် — ရရှိနိုင်မှု (**availability**) ကတိကဝတ်များ၏ ဂဏန်းသင်္ချာ၊ ယခုအခါ ဖောက်သည်များ အပတ်စဉ်လိုလို မေးမြန်းလာကြသော **Retrieval-Augmented Generation** ပုံစံ၊ နှင့် ကုန်ကျစရိတ် တွက်ချက်ခြင်း လေ့ကျင့်ခန်း အပြည့်အစုံ — မည်သူမျှ မတည်ဆောက်မီ **design** တစ်ခုကို ဈေးနှုန်းတွက်ခြင်း။

Cloud Architect သင်တန်းအတွက်

Nine များ၏ ဂဏန်းသင်္ချာ — **SLA**၊ **SLO** နှင့် **error budget** များ

Availability (ရရှိနိုင်မှု) ကို **nine** များဖြင့် ဖော်ပြကြပြီး **architect** တစ်ဦးက **nine** တစ်လုံးကို ပထမဆုံး လုပ်သည့် အရာမှာ မိနစ်များအဖြစ် ပြောင်းခြင်းဖြစ်သည်။ အကြောင်းမှာ မိနစ်များသာ စီးပွားရေးလုပ်ငန်းက ခံစားသိရှိနိုင်သောကြောင့်တည်း။ ရက် ၃၀ ရှိသောလတစ်လတွင် မိနစ် 43,200 ရှိသည်။

Availability	တစ်လလျှင် ခွင့်ပြုထားသော downtime
99.9% (“nine သုံးလုံး”)	မိနစ် ၄၃ ခန့်
99.99% (“nine လေးလုံး”)	၄.၃ မိနစ်ခန့်
99.999% (“nine ငါးလုံး”)	၂၆ စက္ကန့်ခန့်

Nine ငါးလုံး — အမှုဆောင်အရာရှိများ အသံထွက်ပြောရ ကြိုက်တတ်သည့် နံပါတ် — သည် လူကိုင်တွယ်သော ပြန်လည်ကုစားမှု မည်သည့်တစ်ခုထက်မဆို ကျော်လွန်နေသည်။ **failover** သည် အလိုအလျောက် ဖြစ်ရမည် သို့မဟုတ်ပါက ထိုကတိသည် စိတ်ကူးယဉ်သာ ဖြစ်သည်။ **Nine** တစ်လုံး ထပ်တိုးတိုင်း **engineer** လုပ်ရသည့်စရိတ် ဆယ်ဆခန့် တက်သည်။ မိနစ်များအဖြစ် ပြောင်းပြခြင်းသည် **stakeholder** တစ်ဦးကို သိသိကြီးနှင့် ရွေးချယ်စေနိုင်သော နည်းလမ်း ဖြစ်သည်။

Acronym သုံးလုံးက စကားဝိုင်းကို သယ်ဆောင်သည်။ **Service-Level Indicator (SLI)** ဆိုသည်မှာ တိုင်းတာမှုတစ်ခု ဖြစ်သည်။ **Service-Level Objective (SLO)** ဆိုသည်မှာ ထိုတိုင်းတာမှုအပေါ် ချမှတ်ထားသော အတွင်းပိုင်း ပစ်မှတ် ဖြစ်သည်။ **Service-Level Agreement (SLA)** ဆိုသည်မှာ ပြင်ပ ပဋိညာဉ်စာချုပ်ဖြစ်ပြီး လွှဲချော်လျှင် ငွေကြေးတာဝန် တွဲပါသည်။ စည်းကမ်းက: SLO သည် SLA ထက် အမြဲ ပိုတင်းကျပ်ရမည် — ငွေယူပြီး ထိန်းပေးထားရသော မျဉ်း၏ အတွင်းဘက်ကို ချိန်ရွယ်ရသည် — ထို့ပြင် SLI သည် အသုံးပြုသူများ တကယ်ခံစားရသည့်အရာကို တိုင်းရမည်: ဖွင့်ထားလျက် error များ ပြန်ပေးနေသော server တစ်လုံးသည် အရေးပါသော အဓိပ္ပာယ်တိုင်းအရ ကျနေခြင်း (down) ဖြစ်သည်။

Error budget က SLO များကို လက်တွေ့လည်ပတ်နိုင်စေသည်: ပြီးပြည့်စုံမှုနှင့် objective ကြားက ကွာဟချက်ကို သုံးစွဲရမည့် အရင်းအမြစ်တစ်ခုအဖြစ် သဘောထားခြင်း။ 99.9% SLO တစ်ခုက တစ်လလျှင် ချို့ယွင်းချိန် မိနစ် ၄၃ ခွင့် ပြုသည်။ deployment များနှင့် စမ်းသပ်မှုများက ၎င်းထဲမှ သုံးစွဲကြသည်။ ကုန်သွားသောအခါ အဖွဲ့သည် ship လုပ်ခြင်းကို ရပ်ကာ ယုံကြည်စိတ်ချရမှုအပေါ် အလုပ်လုပ်သည် — software လောက၏ အသက်အရင့်ဆုံး ငြင်းခုံမှုဖြစ်သော ship လုပ်မလား တည်ငြိမ်မှုလား ဆိုသည်ကို ထင်မြင်ချက်နေရာတွင် ဂဏန်းသင်္ချာ အစားထိုးခြင်းဖြင့် အဆုံးသတ်ပေးလိုက်သည်။

ယခု ထောင်ချောက်: availability သည် ကွင်းဆက်တစ်လျှောက် မြှောက်ယူရသည်။ Service သုံးခု အစဉ်လိုက် (series) ဆက်ထားလျှင်၊ တစ်ခုစီက ရိုးသားစွာ 99.9% ရှိသော်လည်း ပေါင်းလိုက်သောအခါ $0.999 \times 0.999 \times 0.999 \approx 99.7\%$ ဖြစ်သွားသည် — တစ်လလျှင် မိနစ် ၁၃၀ ခန့်၊ service တစ်ခုချင်းစီ ဝန်ခံထားသည်ထက် သုံးဆ။

Redundancy (အရံထားခြင်း) ကမူ ပြောင်းပြန် အလုပ်လုပ်သည်: သီးခြားလွတ်လပ်သော 99.9% instance နှစ်လုံးကို ပြိုင်တူ (parallel) ထားလျှင် နှစ်လုံးလုံး တစ်ပြိုင်နက် ကျမည့်အချိန်မှာ $0.1\% \times 0.1\%$ သာဖြစ်သည် — multi-zone က ဤမျှ တန်ဖိုးရှိရခြင်း အကြောင်းရင်းပင်။ Interview အဆင့်မီ စာကြောင်း: သင်၏ *synchronous dependency* များ၏ မြှောက်လဒ်ထက် ပိုသော *availability* ကို ကတိမပေးနိုင်ပါ။ ထို့ကြောင့် ကွင်းဆက်ကို တိုအောင်လုပ်ပါ သို့မဟုတ် ကတိကို လျှော့ချပါ။

AI ခေတ်ထဲက Architecture — RAG နှင့် ၎င်း၏ဆွေမျိုးများ

၂၀၂၆ ခုနှစ်ရောက်သောအခါ **pattern catalog** ထဲသို့ ဖောက်သည်များ အပတ်စဉ်လိုလို မေးမြန်းလာကြသော အသစ်တစ်ခု ဝင်ရောက်လာပြီး ၎င်းက ဘာလဲ၊ ဘယ်အခါ တန်ဖိုးရှိလဲ၊ ဘယ်အခါ သင့်ကို အရှက်ရစေမလဲ။ **Large Language Model (LLM — ဘာသာစကား မော်ဒယ်ကြီး)** API တစ်ခု — Amazon Bedrock၊ Azure OpenAI Service — သည် architecture ရှုထောင့်မှကြည့်လျှင် ဈေးကြီးသော၊ နှေးသော၊ *stateless* (မှတ်ဉာဏ်မဲ့) *dependency* တစ်ခု ဖြစ်သည်။ ၎င်းကို **token** လိုက် — token ဆိုသည်မှာ စာလုံးလေးလုံးခန့်ရှိသော စကားလုံးအပိုင်းအစ — မီတာတပ် တွက်ချက်သည်။ ၎င်းသည် millisecond နှင့်မဟုတ်ဘဲ စက္ကန့်နှင့်ချီ၍ ဖြေသည်။ ခေါ်သုံးမှုနှုန်းကြိမ်ကြားတွင် ဘာမျှ မမှတ်မိ၊ ထို့ပြင် တစ်ခါတစ်ရံ ယုံကြည်မှုအပြည့်နှင့် မှားတတ်သည် — ထူးခြားသော ချို့ယွင်းပုံစံများရှိသည့် component တစ်ခု ဖြစ်၍ အခြား dependency များနည်းတူ timeout များ၊ retry များ၊ circuit breaker များနှင့် ဘတ်ဂျက် alarm များ ထိုက်တန်စွာ ရသင့်သည် — အကြောင်းမှာ ထိန်းမရသော ခေါ်သုံးမှုသံသရာ (runaway loop) တစ်ခုသည် နာရီပိုင်း အတွင်း ငွေအစစ်ကို လောင်ကျွမ်းစေသောကြောင့်တည်း။

မော်ဒယ်သည် သင့် document များကို မသိပါ။ စံအဖြေမှာ **Retrieval-Augmented Generation (RAG — ရှာဖွေ ထုတ်ယူမှုဖြင့် အားဖြည့်ထားသော ထုတ်လုပ်ဖန်တီးမှု)** ဖြစ်သည်: မော်ဒယ်ကို သင့် data သင်ပေးမည့်အစား သက်ဆိုင်ရာ

အချက်အလက်များကို မေးခွန်းမေးချိန်တွင် ရှာဖွေကြည့်ရှု လိုက်ခြင်း။ Document များကို အပိုင်းအစ (chunk) များအဖြစ် ဖြတ်၍ တစ်ခုစီကို **embedding** — ၎င်း၏အဓိပ္ပာယ်ကို အာကာသထဲ တည်နေရာချပေးသော ဂဏန်းစာရင်းတစ်ခု — အဖြစ် ပြောင်းကာ **vector database** (AWS ပေါ်တွင် pgvector၊ Azure ပေါ်တွင် AI Search) ထဲ သိမ်းဆည်းထားပြီး ၎င်းက အနီးဆုံး chunk များကို ပြန်ပေးသည်။ ထို chunk များကို **prompt** ထဲ ကူးထည့်လိုက်သည်။ မော်ဒယ်သည် သင့် data ကို ကိုးကားနေခြင်း မဟုတ်ဘဲ သင်ရှာယူပေးလိုက်သော ကောက်နုတ်ချက်များကို အနှစ်ချုပ်ပေးနေခြင်းသာ ဖြစ်သည်။

Whiteboard ပေါ်တွင် **RAG** သည် **pipeline** နှစ်ခုသာ ဖြစ်သည်။ ကြိုတင် run ထားသော **ingestion path** (document များကို ဖြတ်၊ embed လုပ်၊ သိမ်း) နှင့် request တစ်ခုချင်းစီအတွက် run သော **query path** — catalog ထဲ ရှိပြီးသား pattern နှစ်ခုပင်ဖြစ်၍ architect များ ဤအရာကို နေ့လယ်ခင်းတစ်ခုအတွင်း တတ်မြောက်သွားကြခြင်း ဖြစ်သည်။

Trade-off များမှာ ထုံးစံအတိုင်း သုံးခုပင်၊ ပို၍ ထက်နေရုံသာ။ ကုန်ကျစရိတ် (query တိုင်း ငွေအစစ် ကုန်သည် — ထပ်ခါ မေးသည်များကို **cache** လုပ်ပါ၊ လွယ်သောမေးခွန်းများကို ဈေးသက်သာသော မော်ဒယ်ဆီ လမ်းကြောင်းလွှဲပါ)၊ **latency** (millisecond မဟုတ်၊ စက္ကန့်နှင့်ချီ — အဖြေကို stream လုပ်ပြပါ)၊ နှင့် **privacy** (သင့် document များ သင့်နယ်နိမိတ်ကို ကျော်ထွက်သွားသည်၊ **vector database** က document အဆင့် **access control** ကို လျစ်လျူရှုထားလျှင် **chatbot** သည် လစာစာရင်း **spreadsheet** ကို မေးသမျှလူတိုင်းအတွက် ပျော်ပျော်ကြီး အနှစ်ချုပ်ပေးလိမ့်မည်)။ ဘယ်အခါ မတည်ဆောက်သင့် သည်ကိုလည်း သိပါ။ “**chatbot** လိုတယ်” ဆိုသော တောင်းဆိုမှုများစွာသည် ဖက်ရှင်ဝတ်ထားသော **search box** တစ်ခုသာဖြစ်ပြီး၊ တိကျ၍ စာရင်းစစ်နိုင်ရမည့် အဖြေများ — ပေါင်းလဒ်များ၊ လက်ကျန်ငွေများ၊ စာရင်းများ — သည် 97% သာ မှန်တတ်သော စနစ်တစ်ခုအတွက် မှားယွင်းသော အလုပ်ဖြစ်သည်။ **Interview** အဆင့်မီ စာကြောင်း၊ **RAG** ဆိုသည်မှာ အနှစ်ချုပ်ပေးသော **API** တစ်ခုနှင့် ချုပ်တွဲထားသော **search** ပြဿနာတစ်ခုသာ ဖြစ်သည်။

မည်သူမျှ မတည်ဆောက်မီ **design** တစ်ခုကို ကုန်ကျစရိတ် တွက်ခြင်း

သင်တန်း၏ **Pillars of Performance and Cost module** သည် ယခုအခါ **design** လေ့ကျင့်ခန်း အပြည့်အစုံတစ်ခုဖြင့် ပိတ်သည်။ ဖိအားအောက်မှ ခန့်မှန်းတွက်ချက်ခြင်း။ တာဝန်ပေးချက်၊ ခေါင်းဆောင်ပိုင်းက document-management ထုတ်ကုန်တစ်ခု တည်ဆောက်ခွင့် မပြုမီ လစဉ်ကုန်ကျစရိတ် ခန့်မှန်းချက်ကို လိုချင်သည်။ **Workload** — မှတ်ပုံတင်ထားသော အသုံးပြုသူ 200,000၊ နေ့စဉ်သုံးသူ 40,000၊ တစ်ဦးလျှင် တစ်ရက် API request ၂၅ ခုခန့် ပြုလုပ်ပြီး တစ်လလျှင် ပျမ်းမျှ document ၂ megabyte upload လုပ်ကာ ခုနစ်နှစ် သိမ်းဆည်းထားရမည်။ **Design** — zone နှစ်ခုပေါ် ဖြန့်ထားသော load-balanced API tier တစ်ခု၊ managed multi-zone PostgreSQL database တစ်ခု၊ object storage၊ partner တစ်ဦးထံ ညစဉ် 50-gigabyte export တစ်ခု၊ နှင့် သီးခြား development နှင့် staging ပတ်ဝန်းကျင်များ။ လုပ်ငန်းတာဝန်၊ ဂဏန်းအကြမ်းများဖြင့် တစ်မျက်နှာတည်း လစဉ်ခန့်မှန်းချက် — ဈေးနှုန်း တစ်ခုမျှ မပေါ်မီ ယူဆချက်များကို စာဖြင့် ကြိုဖော်ပြထားလျက်၊ capacity ဂဏန်းသင်္ချာ (အသုံးပြုသူများ → တစ်စက္ကန့်လျှင် request → instance အရေအတွက်) ကို ပြသလျက်၊ ယနေ့ / ၂ ဆ / ၁၀ ဆ ဇာတ်လမ်းတွဲများနှင့်အတူ။

လေ့ကျင့်ခန်း၏ **checklist** က ဤစည်းကမ်းများကို သင်ပေးသည်။ **compute** ကို ပျမ်းမျှမဟုတ်ဘဲ **peak** (အမြင့်ဆုံး) load မှ ဈေးတွက်ပါ။ **database standby** ကို ထည့်ရေတွက်ပါ။ **storage** ကို **lifecycle tier** များနှင့်အတူ ခုနစ်နှစ်တာကြီးထွားမှုအဖြစ် **model** လုပ်ပါ။ **egress** (အထွက် data စရိတ်) ကို ခန့်မှန်းရုံမဟုတ်ဘဲ တွက်ချက်ပါ — ခန့်မှန်းချက်

အများစု မေ့တတ်ဆုံး စာကြောင်းဖြစ်ပြီး အကြီးမားဆုံး ဖြစ်တတ်ဆုံး စာကြောင်းလည်း ဖြစ်သည် — NAT (Network Address Translation) gateway များ၊ production မဟုတ်သော ပတ်ဝန်းကျင်များ၊ backup များ၊ monitoring နှင့် support plan ကို မမေ့ပါနှင့်၊ commitment discount များကို တည်ငြိမ်သော baseline အပေါ်တွင်သာ အသုံးပြုပါ။
ထို့ပြင် အဓိကမောင်းနှင်အားများ အမည်တပ်ထားသော range (အပိုင်းအခြား) တစ်ခုနှင့် unit-economics ကျိုးကြောင်း စစ်ဆေးချက် — နေ့စဉ်သုံးသူတစ်ဦးလျှင် တစ်လကုန်ကျစရိတ် — ဖြင့် အဆုံးသတ်ပါ။ တစ်ခုတည်းသော တိကျလွန်းသည့် ဂဏန်းသည် ပြောလိုက်သည့်နေ့မှာပင် မှားနေပြီးသား ဖြစ်သည်။