

Cloud အင်ဂျင်နီယာ သင်တန်း (The Cloud Engineer Course)

အခြေခံဗဟုသုတ လုံးဝမရှိသည့်အခြေအနေမှ အလုပ်အတွက် အမှန်တကယ် အသင့်ဖြစ်သော cloud engineer တစ်ဦးအထိ —

B4LCILC သင်တန်း

၂၀၂၆ ခုနှစ်၊ ဩဂုတ်လ ၁၃ ရက်

Cloud အင်ဂျင်နီယာ သင်တန်း (The Cloud Engineer Course)

အခြေခံဗဟုသုတ လုံးဝမရှိသည့်အခြေအနေမှ cloud engineer အဖြစ် အလုပ်အတွက် အမှန်တကယ် အသင့်ဖြစ်သည်အထိ။

ဤသင်တန်း မည်သို့အလုပ်လုပ်သနည်း

သင်သည် cloud engineer ဖြစ်ရန် လေ့ကျင့်နေခြင်း ဖြစ်သည် — code ဖြင့် server များ တည်ဆောက်သူ၊ နံနက် ၃ နာရီတွင် စနစ်များ ဆက်လက်အသက်ဝင်နေအောင် ထိန်းသိမ်းသူ၊ ပျင်းစရာကောင်းသော အလုပ်များကို automation ဖြင့် ဖယ်ရှားပစ်သူ၊ ပျက်နေသော စနစ်တစ်ခုကို ကြည့်ပြီး ဘာကြောင့်ပျက်သည်ကို တည်ငြိမ်စွာ ရှာဖွေတွေ့ရှိနိုင်သူ ဖြစ်လာရန်ပင်။ ၎င်းသည် လက်ဖြင့် လုပ်ကိုင်ရသော အတတ်ပညာ (trade) တစ်ခုဖြစ်ပြီး ဤသင်တန်းကလည်း ထိုအတိုင်းပင် သဘောထားဆက်ဆံသည်။ စာဖတ်ချိန် ထက် စာရိုက်ချိန်က များစွာ ပိုမည် — module (သင်ခန်းစာအခန်း) တိုင်းတွင် lab (လက်တွေ့လေ့ကျင့်ခန်း) များ ပါရှိပြီး lab များသည်ပင် သင်တန်း၏ အနှစ်သာရ ဖြစ်သည်။ Cloud အကြောင်း ဖတ်ခြင်းက မှတ်မိနိုင်စွမ်းကိုသာ တည်ဆောက်ပေးပြီး cloud ထဲတွင် ကိုယ်တိုင်တည်ဆောက်ခြင်းကမူ အသက်မွေးဝမ်းကျောင်းကို တည်ဆောက်ပေးသည်။

သင်တန်းတွင် အဆင့်ငါးဆင့် ရှိသည်။ အဆင့် ၀ (အပတ်စဉ် ၁-၄): အခြေခံအုတ်မြစ်များ — ကွန်ပျူတာများနှင့် network များ အမှန်တကယ် အလုပ်လုပ်ပုံ၊ နှင့် cloud ကို လည်ပတ်စေသော operating system ဖြစ်သည့် Linux။ အဆင့် ၁ (အပတ်စဉ် ၅-၁၂): အဓိက cloud ကို လက်တွေ့လုပ်ခြင်း — အလုပ်ခေါ်စာတိုင်းတွင် ပါဝင်နေသော AWS ဝန်ဆောင်မှုငါးခုကို ကိုယ်တိုင် တည်ဆောက်ပြီး ပြန်ဖြိုချရသော lab များအဖြစ် သင်ယူခြင်း။ အဆင့် ၂ (အပတ်စဉ် ၁၃-၂၀): Automation — scripting၊ Giti Terraform၊ CI/CD နှင့် container များ — engineer တစ်ဦးနှင့် console ကိုသာ နှိပ်တတ်သူ (console-clicker) ကို ခွဲခြားပေးသော ကျွမ်းကျင်မှုများ။ အဆင့် ၃ (အပတ်စဉ် ၂၁-၂၈): Operations (လည်ပတ်ရေး) — monitoring၊ incident များ၊ security နှင့် ကုန်ကျစရိတ် — သင့်ကို အမှန်တကယ် လခပေးရသည့် ကျွမ်းကျင်မှုများ။ အဆင့် ၄ (လ ၈-၁၂): အလုပ် အတွက် အသင့်ဖြစ်ရေး — portfolio project သုံးခု၊ certification (အသိအမှတ်ပြုလက်မှတ်) သုံးခုနှင့် အင်တာဗျူး လေ့ကျင့် ရေး။

သင်တန်းတစ်ခုလုံးအတွက် စည်းကမ်းများ — တစ်ရက်လျှင် ၁.၅-၂ နာရီ၊ တစ်ပတ်လျှင် ခြောက်ရက် လေ့လာပါ။ ပြင်းထန်မှုထက် တစ်သမတ်တည်းရှိမှုက ပိုအရေးကြီးပြီး ဤအတတ်ပညာသည် တူရိယာတစ်လက်ကဲ့သို့ နေ့စဉ် ထပ်ခါတလဲလဲ လေ့ကျင့်မှုဖြင့် သင်ယူရသော အတတ်ပညာ ဖြစ်သည်။ **Module** တိုင်း၏ အဆုံးတွင် **ဝေါဟာရဇယား** (သင် ပိုင်ဆိုင်ရမည့် စကားလုံးများ)၊ **အသံထွက်ပြော လေ့ကျင့်ခန်း** (သဘာဝကျသည်အထိ ထပ်ခါပြောရမည့် စာကြောင်းများ — အလုပ်အင်တာဗျူးများသည် စာဖြင့်မဟုတ်၊ နှုတ်ဖြင့် ဖြေရသည်)၊ လက်တွေ့ **လေ့ကျင့်ခန်းများ** နှင့် ရှေ့ဆက်ခွင့်ကို ထိန်းချုပ်သည့် **Milestone** (ရည်မှန်းချက်မှတ်တိုင်) တစ်ခုစီ ပါရှိသည်။ — မလုပ်နိုင်သေးသရွေ့ ရှေ့မဆက်ပါနှင့်၊ အကြောင်းမှာ အဆင့်တိုင်းသည် ရှေ့အဆင့်ပေါ်တွင် ရပ်တည်နေသောကြောင့် ဖြစ်သည်။ အပတ်စဉ် ၁ မှစ၍ **Engineering Journal** (အင်ဂျင်နီယာ မှတ်တမ်းစာအုပ်) တစ်အုပ်ထားပါ။ — **lab** တိုင်း၊ **error message** တိုင်း၊ ပြင်ဆင်မှုတိုင်းကို ရိုးသားသော စာပိုဒ်တစ်ပိုဒ်စီဖြင့် မှတ်ပါ။ လ ၁၀ ရောက်သောအခါ ထိုမှတ်တမ်းသည် သင့် **portfolio** ၏ ကုန်ကြမ်းနှင့် သင့်အင်တာဗျူး ဇာတ်လမ်းများ ဖြစ်လာမည်။

အပြန်အလှန်အနေဖြင့် ကတိတစ်ခု — ဤသင်တန်းထဲရှိ မည်သည့်အရာကမှ သင်တစ်ခုခု သိပြီးသားဟု ယူဆမထားပါ။ ဝေါဟာရတိုင်းကို ပထမဆုံးအကြိမ် ပေါ်လာသည့်အချိန်၌ အဓိပ္ပာယ်ဖွင့်ဆိုပေးထားသည်။ **Web browser** တစ်ခုကို သုံးတတ်ပြီး ပထမနှစ်ပတ်အတွင်း ထူးဆန်းသလို ခံစားရမည့် **command** များကို ရိုက်ရန် ဆန္ဒရှိလျှင် လိုအပ်သော ကြိုတင်အရည်အချင်း အားလုံး ပြည့်စုံပြီးသား ဖြစ်သည်။

သင်လေ့ကျင့်နေသည့် အလုပ်ခေါ်စာ (job description)

၂၀၂၆ ခုနှစ် ဩဂုတ်လတွင် ကျွန်ုပ်တို့သည် အလုပ်ရှင်များ၏ **template** များနှင့် ခန့်အပ်ရေးလမ်းညွှန်များ (**Arc.dev**၊ **WiziDevsData**၊ **XOPA**၊ **Betterteam** — စာရင်းအပြည့်အစုံကို **Sources** တွင် ကြည့်ပါ) မှ **Cloud Engineer** အလုပ်ခေါ်စာအစစ်အမှန်များကို စုဆောင်းခဲ့သည်။ ကုမ္ပဏီနာမည်များကို ဖယ်လိုက်လျှင် တူညီသော လိုအပ်ချက်များသာ ထပ်ခါထပ်ခါ ပေါ်လာသည်။ ဤဇယားသည် သင်တန်းနှင့် သင့်အကြား စာချုပ်ပင် ဖြစ်သည်။ — အလုပ်ရှင်တောင်းသော အချက်တိုင်းသည် ၎င်းကို သင်ကြားပေးမည့် အဆင့်နှင့် ချိတ်ဆက်ထားသည်။

အလုပ်ခေါ်စာ အစစ်များ တောင်းဆိုသည့်အချက် (မူရင်းအတိုင်း နီးပါး)	ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ
“Knowledge of Linux/Unix operating systems” (Linux/Unix operating system များအကြောင်း ဗဟုသုတ)	အဆင့် ၀၊ Module 1
“Expertise in cloud networking including VPCs, subnets, load balancers, DNS” (VPC၊ subnet၊ load balancer၊ DNS အပါအဝင် cloud networking ကျွမ်းကျင်မှု)	အဆင့် ၀ Module 2 + အဆင့် ၁ Lab 3
“Demonstrated expertise in core AWS services, including EC2, S3, RDS, VPC, IAM” (EC2၊ S3၊ RDS၊ VPC၊ IAM အပါအဝင် အဓိက AWS ဝန်ဆောင်မှုများတွင် သက်သေပြနိုင်သော ကျွမ်းကျင်မှု)	အဆင့် ၁ (Labs 1–5)
“Design, develop, and deploy cloud infrastructure using infrastructure as code tools such as Terraform, CloudFormation” (Terraform၊	အဆင့် ၂၊ Module 7

အလုပ်ခေါ်စာ အစစ်များ တောင်းဆိုသည့်အချက် (မူရင်းအတိုင်း နှိပ်ပါ)

ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ

CloudFormation ကဲ့သို့ infrastructure as code ကိရိယာများဖြင့် cloud infrastructure ကို ဒီဇိုင်းဆွဲ၊ တည်ဆောက်၊ deploy လုပ်ခြင်း)

“Proficiency in scripting languages such as Python, Bash, PowerShell, or Go” (Python၊ Bash၊ PowerShell သို့မဟုတ် Go ကဲ့သို့ scripting ဘာသာစကားများ ကျွမ်းကျင်မှု)

အဆင့် ၂၊ Module 6

“Build and maintain CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions” (Jenkins၊ GitLab CI သို့မဟုတ် GitHub Actions ကဲ့သို့ ကိရိယာများဖြင့် CI/CD pipeline များ တည်ဆောက်ထိန်းသိမ်းခြင်း)

အဆင့် ၂၊ Module 8

“Experience with containerization technologies including Docker and Kubernetes” (Docker နှင့် Kubernetes အပါအဝင် containerization နည်းပညာများ အတွေ့အကြုံ)

အဆင့် ၂၊ Module 8

“Monitor infrastructure health and performance using cloud-native monitoring tools” (Cloud-native monitoring ကိရိယာများဖြင့် infrastructure ကျန်းမာရေးနှင့် စွမ်းဆောင်ရည်ကို စောင့်ကြည့်ခြင်း)

အဆင့် ၃၊ Module 9

“Participate in incident response, including log analysis”; “troubleshooting and analytical skills” (Log ခွဲခြမ်းစိတ်ဖြာမှုအပါအဝင် incident response တွင် ပါဝင်ခြင်း၊ ပြဿနာရှာဖွေဖြေရှင်းမှုနှင့် ခွဲခြမ်းစိတ်ဖြာမှု ကျွမ်းကျင်မှုများ)

အဆင့် ၃၊ Module 10

“Implement and enforce security controls including encryption, identity and access management”; “least-privilege access” (Encryption၊ identity and access management အပါအဝင် လုံခြုံရေးထိန်းချုပ်မှုများ အကောင်အထည်ဖော်ခြင်း၊ least-privilege access)

အဆင့် ၁ Lab 5 + အဆင့် ၃ Module 10

“Manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging” (Resource များကို အရွယ်အစားမှန်အောင်ချိန်ခြင်း၊ auto-scaling၊ resource tagging ဖြင့် cloud ကုန်ကျစရိတ် စီမံခြင်း)

အဆင့် ၃၊ Module 10

အလုပ်ခေါ်စာ အစစ်များ တောင်းဆိုသည့်အချက် (မူရင်းအတိုင်း နီးပါး)

ဤသင်တန်းတွင် သင်ကြားပေးသည့်နေရာ

“Maintaining, testing and implementing disaster recovery procedures” (Disaster recovery လုပ်ထုံးလုပ်နည်းများ ထိန်းသိမ်း၊ စမ်းသပ်၊ အကောင်အထည်ဖော်ခြင်း)

အဆင့် ၃ + Portfolio Project 3

“AWS certifications preferred” (AWS certification များ ဦးစားပေးမည်)

အဆင့် ၄ cert အချိန်ဇယား (CCP → SAA → Terraform Associate)

“Provide technical guidance and documentation”; “good communication and collaboration skills” (နည်းပညာလမ်းညွှန်မှုနှင့် documentation ပေးခြင်း၊ ဆက်သွယ်ရေးနှင့် ပူးပေါင်းဆောင်ရွက်မှု ကျွမ်းကျင်မှုကောင်းများ)

Journal + runbook များ + အဆင့် ၄ အင်တာဗျူး ပြင်ဆင်မှု

သင် ဘာအတွက် ကြိုးစားနေသည်ကို သိစေရန် လစာအခြေအနေ — အမေရိကန် cloud engineer လစာများသည် တစ်နှစ်လျှင် ပျမ်းမျှ (median) **\$104,000** ဝန်းကျင် တွင် စုပြုံနေပြီး ပုံမှန်အားဖြင့် **\$85K–\$140K** အတွင်း ရှိကာ senior/specialist AWS ရာထူးများကို ထိုထက် များစွာမြင့်သော လစာဖြင့် ကြော်ငြာကြသည်။ အစပြုအဆင့် (entry-level) ရာထူးများသည် နာမည်အမျိုးမျိုးဖြင့် ရှိသည် — cloud support associate၊ junior cloud engineer၊ cloud operations engineer — ဤသင်တန်းသည် ၎င်းတို့၏ လိုအပ်ချက်များကို တိတိကျကျ ရည်ရွယ်ထားသည်။

အဆင့် ၀ — အခြေခံအုတ်မြစ်များ (အပတ်စဉ် ၁-၄)

Module 1 (အပတ်စဉ် ၁-၂): ကွန်ပျူတာများ အလုပ်လုပ်ပုံနှင့် Linux — server များ၏

ဘာသာစကား

အဓိက အယူအဆကြီး: server (ဝန်ဆောင်မှုပေး ကွန်ပျူတာ) ဆိုသည်မှာ အခြားကွန်ပျူတာများကို ဝန်ဆောင်မှုပေးရန် တာဝန်ရှိသော ကွန်ပျူတာတစ်လုံးမျှသာ ဖြစ်သည်။ သင့်ရှေ့မှောက်ရှိ laptop နှင့် Netflix ကို လည်ပတ်စေနေသော စက်များသည် အရွယ်အစားနှင့် ယုံကြည်စိတ်ချရမှုသာ ကွာပြီး သဘောသဘာဝအရ မကွာပါ — နှစ်မျိုးစလုံးသည် **CPU** (အလုပ်လုပ်ပေးသော အစိတ်အပိုင်း)၊ **memory/RAM** (မြန်ဆန်သော ရေတိုအလုပ်ခုံ — restart လုပ်လျှင် ပျက်သွားသည်)၊ **disk** (restart ကို ခံနိုင်သော နှေးသည့် ရေရှည်သိုလှောင်ခန်း) နှင့် **network card** (ကျန်အရာအားလုံးနှင့် ချိတ်ဆက်ပေးသောအရာ) တို့ကို **operating system (OS)** (စက်ကို လည်ပတ်စေသော အခြေခံ software) တစ်ခုက ညှိနှိုင်းစီမံပေးထားခြင်း ဖြစ်သည်။ သင့် laptop သည် Windows သို့မဟုတ် macOS ဖြစ်နိုင်သည်။ Server များကမူ အလွန်အမင်း များပြားစွာ **Linux** ကို သုံးကြသည် — တည်ငြိမ်ပြီး script ရေး၍ရကာ ရိုက်ထည့်သော command များဖြင့် အပြည့်အဝ ထိန်းချုပ်နိုင်သော အခမဲ့ open-source OS ဖြစ်သည်။ နောက်ဆုံးအချက်ကပင် အဓိကအချက် ဖြစ်သည် — mouse click များကို automate လုပ်၍မရ

သော်လည်း **command** များကိုမူ **automate** လုပ်၍ရသည်။ ပြီးလျှင် **cloud engineering** ဆိုသည်မှာ **automation** ပင် ဖြစ်သည်။ ထို့ကြောင့် သင်၏ ပထမဆုံး ကျွမ်းကျင်စွာသုံးတတ်ရမည့်အရာမှာ **terminal** ဖြစ်သည်။

Terminal ကို ရိုးရိုးရှင်းရှင်း ရှင်းပြခြင်း။ **Terminal** (သို့မဟုတ် “**shell**” — ၎င်းအတွင်းရှိ **program** မှာ များသောအားဖြင့် **Bash**) ဆိုသည်မှာ ကွန်ပျူတာနှင့် စာသားဖြင့် စကားပြောခြင်း ဖြစ်သည်။ **Command** တစ်ခု ရိုက်လိုက်လျှင် အဖြေပြန် ပေးသည်။ ဒါပါပဲ။ ဥပမာများတွင် တွေ့ရမည့် **\$** သည် **prompt** ဖြစ်သည် — “သင့်အလှည့်” ဟု **shell** က ပြောနေခြင်း ဖြစ်သည်။ **Linux** တွင် အရာအားလုံးသည် ဖိုင်ဖြစ်ပြီး ဖိုင်များသည် **root /** မှ စတင်သော သစ်ပင်ပုံစံ တစ်ခုတည်းအတွင်း နေထိုင်ကာ သင့်ကိုယ်ပိုင် **folder** မှာ **/home/yourname** (အတိုကောက် **~**) ဖြစ်သည်။ **Path** ဆိုသည်မှာ ထိုသစ်ပင်အတွင်း ရှိ ဖိုင်တစ်ဖိုင်၏ လိပ်စာ ဖြစ်သည် — **/home/anna/notes.txt**။

လေ့ကျင့်ရန် **Linux** တစ်ခု ရယူပါ (တစ်ခုရွေးပါ၊ ဆယ်မိနစ်)။ **Windows** တွင် **WSL** (**Windows Subsystem for Linux** — **Windows** အတွင်း အစစ်အမှန် **Ubuntu Linux** တစ်ခု။ <https://learn.microsoft.com/en-us/windows/wsl/install>) ကို ထည့်သွင်းပါ။ **Mac** တွင် ပါပြီးသား **Terminal app** သည် စတင်ရန် လုံလောက်နီးပါး ဖြစ်သည် (**macOS** သည် **Unix** ၏ ဆွေမျိုးတစ်ဦး ဖြစ်သည်)။ သို့မဟုတ် အပတ်စဉ် ၅ ရောက်လျှင် **AWS** မှ **Linux server** အစစ်တစ်လုံးကို အခမဲ့ငှားရမည့်အချိန် ကို စောင့်နိုင်သည်။ လူအများစုအတွက် အကောင်းဆုံးအဖြေမှာ **WSL** ဖြစ်သည်။

ထိပ်တန်း **command** ၂၅ ခု — ဤဇယားသည် အပတ်စဉ် ၁-၂ ၏ သင်ခန်းစာ ဖြစ်သည်။ တစ်ခုချင်းစီကို အကြိမ်ကြိမ် ရိုက်ပါ။

Command	ဘာလုပ်ပေးသလဲ	ဥပမာ
<code>pwd</code>	Print working directory — “ကျွန်တော် ဘယ်မှာလဲ”	<code>pwd</code>
<code>ls</code>	ဤနေရာရှိ ဖိုင်များ စာရင်းပြ (<code>-l</code> အသေးစိတ်၊ <code>-a</code> ဖျောက်ထားသည်များပါ)	<code>ls -la</code>
<code>cd</code>	Change directory — သစ်ပင်အတွင်း လှည့်လည်သွားလာခြင်း	<code>cd /var/log</code>
<code>mkdir</code>	Directory (folder) တစ်ခု ဖန်တီးခြင်း	<code>mkdir projects</code>
<code>touch</code>	ဖိုင်အလွတ်တစ်ဖိုင် ဖန်တီးခြင်း	<code>touch notes.txt</code>
<code>cp</code>	ဖိုင်ကူးခြင်း (folder များအတွက် <code>-r</code>)	<code>cp a.txt backup.txt</code>
<code>mv</code>	ရွှေ့ခြင်း သို့မဟုတ် နာမည်ပြောင်းခြင်း	<code>mv old.txt new.txt</code>
<code>rm</code>	ဖျက်ခြင်း — အပြီးအပိုင်၊ အမှိုက်ပုံး မရှိပါ။ လေးစားစွာ သုံးပါ။	<code>rm notes.txt</code>
<code>cat</code>	ဖိုင်တစ်ဖိုင်၏ အကြောင်းအရာအားလုံး ပုံနှိပ်ပြခြင်း	<code>cat notes.txt</code>

Command	ဘာလုပ်ပေးသလဲ	ဥပမာ
less	ဖိုင်ရှည်ကြီးကို စာမျက်နှာလိုက် ဖတ်ခြင်း (ထွက်ရန် q)	less /var/log/syslog
head / tail	ဖိုင်တစ်ဖိုင်၏ ပထမ / နောက်ဆုံး စာကြောင်းများ။ tail -f သည် log တစ်ခုကို တိုက်ရိုက် စောင့်ကြည့်ပေးသည် — ops လောက၏ ဂန္ထဝင်နည်း	tail -f app.log
grep	စာသားထဲမှ ပုံစံ (pattern) တစ်ခုကို ရှာဖွေခြင်း — အသုံးအများဆုံး ops command	grep "ERROR" app.log
find	နာမည်/အရွယ်အစား/သက်တမ်းဖြင့် ဖိုင် ရှာခြင်း	find / -name "*.conf"
echo	စာသား ပုံနှိပ်ခြင်း (ဖိုင်များ သို့မဟုတ် variable များထဲသို့ မကြာခဏ)	echo "hello"
nano	Terminal အတွင်း သုံးရလွယ်သော text editor	nano notes.txt
man	မည်သည့် command အတွက်မဆို လက်စွဲစာအုပ် (ထွက်ရန် q)	man grep
sudo	Command တစ်ခုကို အာဏာအပြည့်ရှိ admin (“root”) အဖြစ် လည်ပတ်ခြင်း။ လေးစားစွာဖြင့်။	sudo apt update
apt	Software ထည့်သွင်း/update လုပ်ခြင်း (Ubuntu/Debian မိသားစု)	sudo apt install htop
chmod	ဖိုင်တစ်ဖိုင်၏ permission များ ပြောင်း ခြင်း	chmod 644 notes.txt
chown	ဖိုင်တစ်ဖိုင်၏ ပိုင်ရှင် ပြောင်းခြင်း	sudo chown anna file
ps	လည်ပတ်နေသော process များ စာရင်းပြ (အားလုံးအတွက် ps aux)	ps aux
top (သို့မဟုတ် htop)	CPU/memory/process များ၏ တိုက်ရိုက် dashboard — “server ဘာ လို့ နှေးနေတာလဲ” သည် ဤနေရာမှ စသည်	top

Command	ဘာလုပ်ပေးသလဲ	ဥပမာ
df -h / du -sh	လွတ်နေသော disk နေရာ / folder တစ်ခု သုံးထားသောနေရာ — “disk ပြည့်နေပြီ” သည် ဤနေရာမှ စသည်	df -h
ssh	Network မှတစ်ဆင့် အခြားစက် တစ်လုံး၏ terminal ထဲ ဝင်ရောက်ခြင်း — cloud engineer ၏ အဓိက အိမ်ရှေ့ တံခါး	ssh anna@server-ip
curl	Terminal မှ web request တစ်ခု ပြုလုပ်ခြင်း — “site က တက်နေရဲ့လား”	curl https://example.com

Permission များ — ဖိဝ စက္ကန့် ဗားရှင်း။ ဖိုင်တိုင်းတွင် ပိုင်ရှင်တစ်ဦးနှင့် `rwxr-xr--` ကဲ့သို့ mode တစ်ခု ရှိသည် — သုံးလုံးတွဲ သုံးတွဲ — ပိုင်ရှင် (owner)၊ အုပ်စု (group)၊ လူတိုင်း (everyone) အတွက် read (ဖတ်)၊ write (ရေး)၊ execute (လည်ပတ်) ခွင့်များ။ ဂဏန်းဖြင့်ဆိုလျှင် `r=4, w=2, x=1` ဖြစ်၍ `chmod 755 script.sh` ၏ အဓိပ္ပာယ်မှာ “ပိုင်ရှင်က အားလုံးလုပ်နိုင် ($7=4+2+1$)၊ ကျန်လူတိုင်းက ဖတ်ခြင်းနှင့် လည်ပတ်ခြင်း လုပ်နိုင် ($5=4+1$)” ဖြစ်သည်။ Program တစ်ခုက “doesn’t have permission” ဟုပြောလျှင် စနစ်က ငြင်းပယ်နေခြင်းဖြစ်ပြီး — ယခုအခါ ဘာကြောင့်ဆိုသည်ကို သင်ဖတ်တတ်ပြီ။

SSH — ဖိဝ စက္ကန့် ဗားရှင်း။ ssh သည် အခြားစက်တစ်လုံးပေါ်တွင် လုံခြုံသော အဝေးထိန်း terminal တစ်ခုကို ဖွင့်ပေးသည် — သင့် laptop မှ Virginia ရှိ server တစ်လုံးထဲသို့ ထိုစက်ရှေ့တွင် ထိုင်နေသကဲ့သို့ပင်။ Password များအစား ကျွမ်းကျင်သူများသည် **key pair** (သော့အတွဲ) ကို သုံးကြသည် — **private key** (သင့် laptop ပေါ်ရှိ လျှို့ဝှက်ဖိုင်၊ ဘယ်တော့မှ မမျှဝေရ) နှင့် **public key** (server ပေါ်တွင် ထားရှိသည်)။ ၎င်းတို့သည် သော့နှင့် သော့ခလောက်ကဲ့သို့ ကိုက်ညီသည်။ အဆင့် ၁ တွင် သင် launch လုပ်မည့် AWS server တိုင်းက ဤအရာကိုပင် ပေးအပ်မည်။

Process များနှင့် service များ: process ဆိုသည်မှာ လည်ပတ်နေသော program တစ်ခုဖြစ်ပြီး **service** (သို့မဟုတ် “daemon”) ဆိုသည်မှာ နောက်ကွယ်တွင် အမြဲလည်ပတ်နေသော process တစ်ခု ဖြစ်သည် — **web server** များ၊ **database** များ။ `systemctl status nginx` သည် Linux ကို “nginx service ကျန်းမာရဲ့လား” ဟု မေးခြင်း ဖြစ်သည် — နှစ်ပေါင်းများစွာ အလုပ်ထဲတွင် ရိုက်နေရမည့် စာကြောင်းတစ်ကြောင်း ဖြစ်သည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Server	အခြားကွန်ပျူတာများကို ဝန်ဆောင်မှုပေးရန် တာဝန်ရှိသော ကွန်ပျူတာ။ Cloud တွင် သင်ငှားရမ်းထားသော ကွန်ပျူတာ။
CPU / RAM / disk	အလုပ်သမား၊ မြန်ဆန်သော ယာယီအလုပ်ခုံ (restart လုပ်လျှင် ပျက်သည်)၊ နှင့် နှေးသော အမြဲတမ်းသိုလှောင်ခန်း။

ဝေါဟာရ	အဓိပ္ပာယ်
Operating system (OS)	စက်ကို လည်ပတ်စေပြီး program များကို လက်ခံထားပေးသော software ။ Server များသည် Linux ကို သုံးသည်။
Linux / distribution	အခမဲ့ open-source server OS ။ “ Distro ” (Ubuntu ၊ Amazon Linux ၊ Debian) ဆိုသည်မှာ ၎င်း၏ ထုပ်ပိုးထားသော အမျိုးအစားတစ်ခု။
Terminal / shell / Bash	OS ဆီသို့ စာသားဖြင့် ဆက်သွယ်ရာ မျက်နှာပြင် / သင့် command များကို အဓိပ္ပာယ်ဖော်ပေးသော program / စံ shell ၏ နာမည်။
Prompt	\$ သင်္ကေတ — shell က သင့် command ကို စောင့်နေခြင်း။
Directory / path	Folder တစ်ခု / / မှ စတင်သော သစ်ပင်တစ်ခုတည်းအတွင်းရှိ ဖိုင်တစ်ဖိုင်၏ လိပ်စာအပြည့်အစုံ။
Root (အဓိပ္ပာယ်နှစ်မျိုး)	ဖိုင်သစ်ပင်၏ ထိပ် (/) နှင့် အာဏာအပြည့်ရှိ admin user ။ စကားစပ်အရ ခွဲခြားနိုင်သည်။
sudo	“ Superuser do ” — command တစ်ခုကို admin အာဏာ ဖြင့် လည်ပတ်ခြင်း။
Permissions (rwx)	မည်သူ ဖတ်၊ ရေး၊ လည်ပတ်နိုင်သည်ကို ဖိုင်တစ်ဖိုင်ချင်း သတ်မှတ်သော စည်းကမ်းများ — ပိုင်ရှင်/အုပ်စု/လူတိုင်းအတွက် သုံးလုံးတွဲများဖြင့် ပြသည်။
Process / service (daemon)	လည်ပတ်နေသော program တစ်ခု / နောက်ကွယ်တွင် အမြဲ လည်ပတ်နေသော တစ်ခု (web server များ၊ database များ)။
SSH / key pair	အခြားစက်တစ်လုံး၏ terminal ထဲသို့ လုံခြုံစွာ အဝေးမှ ဝင် ရောက်ခြင်း / password များအစား သုံးသော private+public သော့ဖိုင်များ။
Log	Software က ဖြစ်ပျက်သမျှကို ရေးမှတ်ထားသော စာသားဖိုင် — တစ်ခုခု ပျက်လျှင် ပထမဆုံး ကြည့်ရမည့်နေရာ။
Package manager	OS ၏ software ထည့်သွင်းကိရိယာ (apt ၊ yum) — download စာမျက်နှာမှမဟုတ်၊ command ဖြင့် software ရယူခြင်း။

ဤ **module** အတွက် ဗီဒီယိုများ (**link** များ စစ်ဆေးပြီး):

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
Linux Operating System — Crash Course for Beginners	freeCodeCamp	~2 hr	https://www.youtube.com/watch?v=ROjZy1WbCIA
WSL install guide (ဗီဒီယိုမဟုတ်၊ ကိုးကားစာ)	Microsoft Learn	—	https://learn.microsoft.com/en-us/windows/wsl/install

အသံထွက်၍ သဘာဝကျသည်အထိ ပြောပါ: “Let me SSH in and check the logs.” (SSH ဝင်ပြီး log တွေ စစ်ကြည့် လိုက်မယ်။) · “Grep the log for the error, then tail -f it while we retry.” (Log ထဲက error ကို grep နဲ့ရှာပြီး ပြန်စမ်း နေစဉ် tail -f နဲ့ စောင့်ကြည့်ပါ။) · “It’s a permissions problem — who owns the file and what’s the mode?” (ဒါ permission ပြဿနာပဲ — ဖိုင်ကို ဘယ်သူပိုင်ပြီး mode က ဘာလဲ။) · “Check top — is it CPU, memory, or disk?” (top ကို စစ်ပါ — CPU လား၊ memory လား၊ disk လား။)

လေ့ကျင့်ခန်းများ: (၁) သင့် Linux terminal ထဲတွင် `mkdir` နှင့် `touch` ဖြင့် project သစ်ပင်ငယ်တစ်ခု တည်ဆောက်ပါ။ ဖိုင်များ ကူးပြီး ရွှေ့ပါ။ ထို့နောက် ဖျက်ပစ်ပါ — command တစ်ခုချင်းစီကို အသံထွက်၍ ရှင်းပြရင်း လုပ်ပါ။ (၂) `echo "hello from $(whoami)"` ပါဝင်သော `hello.sh` ကို ဖန်တီးပြီး `chmod +x` ဖြင့် executable ဖြစ်အောင်လုပ် ကာ `./hello.sh` ဖြင့် လည်ပတ်ပါ။ (၃) Log ဖိုင်တစ်ဖိုင်ပေါ်တွင် `tail -f` ကို လည်ပတ်ပြီး (Ubuntu တွင်: `sudo tail -f /var/log/syslog`) စာကြောင်းများ ရောက်လာသည်ကို ကြည့်ပါ။ (၄) Journal — server များ ဘာကြောင့် Linux သုံးကြသည်ကို စိတ်ကူးထဲက မိတ်ဆွေတစ်ဦးအား စာကြောင်းသုံးကြောင်းဖြင့် ရှင်းပြပါ။

Milestone: မှတ်စုမကြည့်ဘဲ ဖိုင်သစ်ပင်အတွင်း မည်သည့်နေရာသို့မဆို သွားနိုင်ခြင်း၊ ဖိုင်များ ဖန်တီး/ကူး/ရွှေ့/ဖျက်နိုင်ခြင်း၊ `chmod 755` ကို ရှင်းပြနိုင်ခြင်း၊ ဖိုင်တစ်ဖိုင်ထဲမှ စကားလုံးတစ်လုံးကို `grep` ဖြင့် ရှာနိုင်ခြင်း။ ဤအထဲမှ တစ်ခုခုအတွက် ရှာဖွေ ကြည့်ရှုရန် လိုနေသေးလျှင် ဤနေရာတွင် နောက်ထပ် နှစ်ရက် အချိန်ပေးပါ။ ဤ module သည် ကျန်အရာအားလုံး၏ အလေးချိန် ကို ထမ်းထားသော အုတ်မြစ် ဖြစ်သည်။

Module 2 (အပတ်စဉ် ၃–၄): Networking အခြေခံများ + သင့် AWS account

အဓိက အယူအဆကြီး: network ဆိုသည်မှာ လိပ်စာတပ်ထားသော စာအိတ်များ အပြန်အလှန် ပို့နေကြသော ကွန်ပျူတာများ ဖြစ်သည်။ စက်တိုင်းသည် IP address (ဂဏန်းဖြင့်ရေးထားသော လမ်းလိပ်စာကဲ့သို့ — ဥပမာ 172.31.8.14) တစ်ခုစီ ရရှိ သည်။ ဒေတာကို packet (စာအိတ်) များအဖြစ် အပိုင်းပိုင်းဖြတ်ပြီး ဦးတည်ရာလိပ်စာဆီသို့ တစ်ဆင့်ချင်း (hop by hop) လမ်းကြောင်းရွေး ပို့ဆောင်သည်။ ရောက်ရှိသောအခါ port (ဆိပ်ကမ်းနံပါတ်) က ထိုစာအိတ်သည် မည်သည့် program အတွက် ဖြစ်သည်ကို ပြောပြသည် — အဆောက်အအုံတစ်လုံးတည်း၊ နံပါတ်တပ်ထားသော တံခါးပေါက် ထောင်ချီ — port 22 သည် SSH၊ 80 သည် ကုန်မဝှက်ထားသော web (HTTP)၊ 443 သည် ကုန်ဝှက်ထားသော web (HTTPS)၊ 5432 သည် PostgreSQL။ “Open port 443” ၏ အဓိပ္ပာယ်မှာ “တံခါး 443 သို့ လိပ်စာတပ်ထားသော စာအိတ်များကို ခွင့်ပြုပါ” ဟူ၍ ဖြစ်သည်။

Private နှင့် public: သင့်အိမ်နှင့် cloud network တိုင်းသည် local network အတွင်း၌သာ အလုပ်လုပ်သော private IP အပိုင်းအခြားများ (10.x.x.x၊ 172.16–31.x.x၊ 192.168.x.x) ကို ပြန်လည်အသုံးပြုကြသည်။ Public IP

ဆိုသည်မှာ အင်တာနက်တစ်ခုလုံးမှ လက်လှမ်းမီနိုင်သော လိပ်စာ ဖြစ်သည်။ ဤခွဲခြားမှုသည် cloud လုံခြုံရေး၏ အခြေခံအုတ်မြစ် ဖြစ်သည် — အင်တာနက်နှင့် ရင်ဆိုင်စရာမလိုသောအရာများကို public လိပ်စာ လုံးဝ မပေးထားပါ။

DNS — အင်တာနက်၏ ဖုန်းစာအုပ်။ လူသားများက နာမည် (example.com) ကို သုံးကြပြီး packet များကမူ ဂဏန်းများ လိုအပ်သည်။ **DNS** က ဘာသာပြန်ပေးသည် — သင့်စက်က **DNS server** တစ်ခုအား “example.com ရဲ့ IP က ဘာလဲ” ဟု မေးပြီး ဂဏန်းရလျှင် ချိတ်ဆက်သည်။ လျှို့ဝှက်ဆန်းကြယ် ပျက်စီးမှု (outage) အားလုံး၏ တစ်ဝက်လောက်တွင် **DNS** ပါဝင် နေသည်။ “It’s always DNS” (DNS ကြောင့်ချည်းပဲ) ဟူသော လုပ်ငန်းနယ်ပယ် ပြက်လုံးရှိသည်မှာ မကြာခဏ မှန်နေတတ် သောကြောင့် ဖြစ်သည်။ `nslookup example.com` သည် ထိုရှာဖွေမှုကို လက်ဖြင့် လုပ်ကြည့်ခြင်း ဖြစ်သည်။

HTTP — web ၏ စကားပြောပုံ။ **Client** (browser) က request တစ်ခု ပို့သည် — **method** (GET = ယူ၊ POST = တင် သွင်း) တစ်ခုနှင့် **path** တစ်ခု — ပြီးလျှင် **server** က **status code** တစ်ခုဖြင့် ပြန်ဖြေသည် — 200 OK၊ 301 ရွှေ့ပြောင်း ပြီး၊ 403 တားမြစ်ထား၊ 404 ရှာမတွေ့၊ 500 server ချို့ယွင်းမှု၊ 502/503 “ကျွန်တော့်နောက်ကွယ်က server ပျက်နေ/ဝန်ပိနေ တယ်”။ ထိုခြောက်ခုကို အလွတ်ကျက်ပါ — **engineer** တစ်ဦးအနေဖြင့် ၎င်းတို့ကို နေ့စဉ် ဖတ်ရမည်။ `curl -I https://example.com` က status line နှင့် header များကို တိုက်ရိုက် ပြပေးသည်။

Firewall များ မည်သည့် packet များ ဖြတ်သန်းခွင့်ရှိသည်ကို source၊ destination နှင့် port အလိုက် ဆုံးဖြတ်သော စည်းကမ်းစာရင်း — “443 ကို ဘယ်နေရာကမဆို ခွင့်ပြု၊ 22 ကို ရုံးကလာမှသာ ခွင့်ပြု၊ ကျန်တာ ငြင်းပယ်”။ **AWS** တွင် server တစ်လုံးချင်း firewall ကို **security group** ဟုခေါ်ပြီး မှားယွင်းစွာ ပြင်ဆင်ထားသော security group များသည် အစပြု သူများ၏ နံပါတ်တစ် လုံခြုံရေးအပေါက် ဖြစ်သည်။ **Latency** (နှောင့်နှေးချိန်၊ millisecond ဖြင့်) နှင့် **bandwidth** (တစ်စက္ကန့် လျှင် သယ်ဆောင်နိုင်စွမ်း) တို့က ဝေါဟာရကို ပြည့်စုံစေသည် — အကွာအဝေးက latency ကို ဖြစ်စေသောကြောင့် cloud များ တွင် ကမ္ဘာအနှံ့ region များ ရှိရခြင်း ဖြစ်သည်။

ယခု၊ သင့် **AWS account** — ဤ **module** ၏ ဒုတိယတစ်ဝက်။ <https://aws.amazon.com/free> သို့သွား၍ Free Tier account တစ်ခု ဖွင့်ပါ (email၊ ဖုန်း၊ မည်သူမည်ဝါဖြစ်ကြောင်း အတည်ပြုရန် credit/debit ကတ် — ဤသင်တန်း၏ teardown (ပြန်ဖြိုချရေး) စည်းကမ်းကို လိုက်နာလျှင် ကတ်မှ ငွေ သိသိသာသာ ဖြတ်ခံရမည် မဟုတ်ပါ)။ **Free Tier** သည် အခြေခံဝန်ဆောင်မှုများကို လစဉ် အခမဲ့ခွင့်ပြုချက်ဖြင့် ပေးထားပြီး (ပထမနှစ်တွင် **EC2 server** အသေးတစ်လုံးကို တစ်လလျှင် နာရီ ၇၅၀ အပါအဝင်) ဤသင်တန်းရှိ lab တိုင်းကို ထိုအခမဲ့ဘောင်အတွင်း ဝင်အောင် ဒီဇိုင်းဆွဲထားသည်။ ထို့နောက် အခြားဘာမ လုပ်မီ ငြင်းဆို၍မရသော လုံခြုံရေးအဆင့် သုံးဆင့် — ဤအရာများကို လုပ်ခြင်းသည်ပင် သင့်လုံခြုံရေး အသက်မွေးဝမ်းကျောင်း၏ ပထမဆုံး လေ့ကျင့်ခန်း ဖြစ်သည် —

1. **Root user ပေါ်တွင် MFA။** Root user သည် account ၏ ပင်မသော့ချက် ဖြစ်သည်။ IAM → Security credentials တွင် multi-factor authentication (ဖုန်း authenticator app တစ်ခု) ကို ထည့်ပါ။ ထို့နောက် နေ့စဉ် အလုပ်အတွက် root ကို သုံးခြင်း ရပ်လိုက်ပါ။
2. **Admin IAM user တစ်ဦး ဖန်တီးပါ** (Lab 5 တွင် IAM ကို နက်နက်နဲနဲ ရှင်းပြမည်။ ယခုအတွက် — console → IAM → Users → admin access ဖြင့် user တစ်ဦး ဖန်တီးပါ) ပြီးလျှင် ယခုမှစ၍ ထို user ဖြင့်သာ login ဝင်ပါ။
3. **Billing alert (ကုန်ကျစရိတ် သတိပေးချက်) တစ်ခု။** Console → Billing → Budgets → **zero-spend budget** (တစ်ခုခု စတင်ကောက်ခံသည်နှင့် ချက်ချင်း email ပို့ပေးသော **AWS** ၏ template) တစ်ခု ဖန်တီးပါ။ **Free Tier usage alert** များကိုလည်း ဖွင့်ပါ။ ကုန်ကျစရိတ်ကို မထိန်းချုပ်နိုင်သော **engineer** သည် အန္တရာယ်တစ်ခုသာ ဖြစ်သည်။ Account ဖွင့်ပြီး မိနစ် ၂၀ အတွင်းမှာပင် သင်သည် ကျွမ်းကျင်သူအများစုထက် ရှေ့ရောက်နေပြီ။ (ကိုးကားစာ: <https://docs.aws.amazon.com/cost-management/latest/userguide/budgets-managing-costs.html>)

ဝေါဟာရများ

ဝေါဟာရ	အဓိပ္ပာယ်
IP address	စက်တစ်လုံး၏ ဂဏန်း network လိပ်စာ၊ ဥပမာ 172.31.8.14။
Packet	လိပ်စာတပ်ထားသော ဒေတာစာအိတ်တစ်စောင်။ Traffic အားလုံးသည် ၎င်းတို့၏ စီးကြောင်းများ ဖြစ်သည်။
Port	Traffic သည် မည်သည့် program အတွက်ဖြစ်သည်ကို ဖော်ပြသော စက်တစ်လုံးပေါ်ရှိ နံပါတ်တပ် တံခါးပေါက် — 22 SSH၊ 80 HTTP၊ 443 HTTPS။
Private / public IP	Local network အတွင်း၌သာ အကျုံးဝင်သော လိပ်စာ / အင်တာနက်တစ်ခုလုံးမှ လက်လှမ်းမီနိုင်သော လိပ်စာ။
DNS	နာမည်များ (example.com) ကို IP address များအဖြစ် ဘာသာပြန်ပေးသော စနစ်။ “It’s always DNS.”
HTTP / HTTPS	Web ၏ request-response ဆက်သွယ်ရေးနည်း (protocol) / အလားတူပင်၊ TLS ဖြင့် ကုဒ်ဝှက်ထားသော။
Status code	Server ၏ အဖြေအကျဉ်း — 200 OK၊ 404 ရှာမတွေ့၊ 500 server ချို့ယွင်း၊ 503 ဝန်ပိ။
Firewall	မည်သည့် packet များ ဖြတ်သန်းခွင့်ရသည်ကို source၊ destination၊ port အလိုက် ဆုံးဖြတ်သော စည်းကမ်းစာရင်း။
Security group	AWS ၏ server တစ်လုံးချင်း firewall။ မှားယွင်းစွာ ပြင်ဆင်ခြင်းသည် အစပြုသူတို့၏ ဂန္ထဝင်အပေါက်။
Latency / bandwidth	နှောင့်နှေးချိန် (ms) / သယ်ဆောင်နိုင်စွမ်း (တစ်စက္ကန့်လျှင်)။ အကွာအဝေးက latency ကို ဖြစ်စေသည်။
Client / server (အခန်းကဏ္ဍများ)	Network စကားဝိုင်းတိုင်းရှိ မေးသူနှင့် ဖြေသူ။
AWS Free Tier	AWS account အသစ်တစ်ခု၏ လစဉ် အခမဲ့ခွင့်ပြုချက် — ဤသင်တန်း၏ ဘတ်ဂျက်အားလုံး။
Root user	AWS account ၏ ပင်မ identity။ MFA တပ်ပြီး သုံးခြင်း ရပ်ပါ။

ဝေါဟာရ	အဓိပ္ပာယ်
Billing alert / budget	ကုန်ကျစရိတ် သတ်မှတ်ချက်ကျော်လျှင် အလိုအလျောက် ရောက်လာသော email။ သင့်ဟာကို \$0 တွင် သတ်မှတ်ထားသည်။

ဤ module အတွက် ဗီဒီယိုများ:

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
What is AWS?	Amazon Web Services (official)	~2 min	https://www.youtube.com/watch?v=a9__D53WsUs
Top 50+ AWS Services Explained in 10 Minutes	Fireship	~10 min	https://www.youtube.com/watch?v=JlblYCM48to
AWS Networking Basics — VPC & Subnets	KodeKloud	~30 min	https://www.youtube.com/watch?v=QM63dyA_4Pc

အသံထွက်၍ ပြောပါ: “What’s the IP, and is it public or private?” (IP က ဘာလဲ၊ public လား၊ private လား။) · “Is port 443 open in the security group?” (Security group ထဲမှာ port 443 ဖွင့်ထားရဲ့လား။) · “Curl it — what status code do you get?” (curl နဲ့ စမ်းကြည့် — ဘယ် status code ရလဲ။) · “Did DNS resolve? Check with nslookup before blaming the server.” (DNS resolve ဖြစ်ရဲ့လား။ Server ကို အပြစ်မတင်ခင် nslookup နဲ့ အရင်စစ်ပါ။)

လေ့ကျင့်ခန်းများ: (၁) ping google.com (latency ကို မှတ်ပါ)၊ nslookup google.com (DNS က IP အများကြီး ပြန်ပေးသည်ကို မှတ်ပါ)၊ curl -I https://aws.amazon.com (status line နှင့် header သုံးခုကို ဖတ်ပြီး တစ်ခုချင်း ရှာဖွေလေ့လာပါ)။ (၂) သင့်စက်၏ private IP (Linux တွင် ip addr) နှင့် public IP (“what is my IP” ဟု ရှာပါ) ကို ရှာပြီး — ဘာကြောင့်ကွာသည်ကို journal ထဲတွင် ရှင်းပြပါ။ (၃) AWS account ပြင်ဆင်မှုကို အပြီးသတ်ပါ — MFA၊ admin user၊ zero-spend budget — budget ကို journal အတွက် screenshot ရိုက်ထားပါ။ (၄) အလွတ်ဆွဲပါ — laptop → DNS lookup → port 443 မှတစ်ဆင့် HTTPS request → firewall → web server။ အလိုအလျောက် ဖြစ်လာသည်အထိ သုံးကြိမ်။

Milestone — အဆင့် ၀ အဆုံး: URL တစ်ခုရိုက်ပြီး Enter နှိပ်လိုက်သောအခါ ဖြစ်ပျက်သမျှကို — DNS၊ IP၊ port၊ firewall၊ HTTP request၊ status code — ဝေါဟာရတိုင်းကို မှန်ကန်စွာသုံးလျက် နှစ်မိနစ်အတွင်း ပြောပြနိုင်ရမည်။ ထို့ပြင် သင့် AWS account သည် MFA နှင့် \$0 budget alert တို့ဖြင့် တည်ရှိနေပြီ ဖြစ်ရမည်။ ယခုအခါ အင်တာနက်ကို အသက်မွေးဝမ်းကျောင်းအဖြစ် သုံးနေသော လူအများစုထက်ပင် အင်တာနက် အလုပ်လုပ်ပုံကို သင် ပိုသိနေပြီ။ အဆင့် ၁ သည် ထိုအပေါ်တွင် စတင်တည်ဆောက်ရမည့်နေရာ ဖြစ်သည်။

အဆင့် ၁ — အဓိက CLOUD ကို လက်တွေ့လုပ်ခြင်း (အပတ်စဉ် ၅-၁၂)

ဤအဆင့် အလုပ်လုပ်ပုံ။ အောက်ပါ ဝန်ဆောင်မှုငါးခုစီတိုင်းသည် **Lab** တစ်ခုစီ ဖြစ်သည် — ရည်မှန်းချက်တစ်ခု၊ လုပ်ဆောင်ရမည့် အဆင့်များ၏ အကြမ်းဖျင်း (လိုက်လုပ်နိုင်လောက်အောင် အသေးစိတ်ပြီး ကိုယ်တိုင်စဉ်းစားရလောက်အောင် တိုသည် — စဉ်းစားခြင်းသည်ပင် သင်ယူခြင်း ဖြစ်သည်)။ သင်ယူရရှိသည့်အရာများ၊ နှင့် — အမြဲတမ်း — **teardown** (ပြန်ဖို့ချခြင်း)။

Teardown စည်းကမ်းသည် နှစ်ထပ်ကွမ်း အရေးကြီးသည် — **Free Tier** အတွင်း သင့်ကို ထိန်းထားပေးပြီး “မလိုအပ်ဘဲ ဘာမှလည်ပတ်မထားခြင်း” သည် အင်တာဗျူးသူများ အမှန်တကယ် စမ်းစစ်တတ်သော ကျွမ်းကျင်သူအလေ့အထ ဖြစ်သည်။ **Lab** တစ်ခုချင်းစီကို အနည်းဆုံး နှစ်ကြိမ် ပြန်တည်ဆောက်ပါ — တစ်ကြိမ်က အကြမ်းဖျင်းအတိုင်း၊ တစ်ကြိမ်က အလွတ်။ ဒုတိယအကြိမ်တည်ဆောက်မှုသည် ဗဟုသုတကို သင့်လက်ထဲ ရွှေ့ပေးသည့်အချိန် ဖြစ်သည်။ **Lab** တစ်ခုလျှင် ခန့်မှန်းခြေ တစ်ပတ်ခွဲ အချိန်သတ်မှတ်ထားပြီး ပိုနေသောအချိန်ကို ပျက်စီးမှုများအတွက် သုံးပါ — အကြောင်းမှာ တစ်ခုခုတော့ ပျက်ကိုပျက်မည်ဖြစ်ပြီး ၎င်းတို့ကို **debug** လုပ်ခြင်းသည် ဤသင်တန်းက ကြိုရေးမပေးနိုင်သော အကောင်းဆုံး သင်ခန်းစာ ဖြစ်သောကြောင့်တည်း။

ပထမဦးစွာ သင်တည်ဆောက်တော့မည့်အရာအားလုံးကို ဘောင်ခတ်ပေးသော အယူအဆသုံးခု။ **Region များနှင့် Availability Zone များ**။ **Region** ဆိုသည်မှာ **AWS data center** များ၏ ပထဝီအလိုက် အစုအဝေးဖြစ်သည် (သင်နှင့်နီးသော တစ်ခုကို ရွေးပြီး ထိုထဲမှာပဲ နေပါ — **region** တစ်ခုရှိ **resource** များကို အခြား **region** မှ မမြင်ရ — “ကျွန်တော့် **server** ဘယ်ရောက်သွားလဲ” ဟူသော နံပါတ်တစ် အထင်အမြင်လွှဲမှု)။ **Availability Zone (AZ)** ဆိုသည်မှာ **region** အတွင်းရှိ သီးခြားခွဲထားသော **data center** တစ်ခုဖြစ်ပြီး အလေးအနက် စနစ်များသည် အဆောက်အအုံတစ်လုံး ပျက်လျှင် မပြုတ်ကျစေရန် နှစ်ခုတွင် လည်ပတ်ကြသည်။ **Console** နှင့် **CLI**။ **console** (<https://console.aws.amazon.com>) သည် **AWS** ၏ **web** ထိန်းချုပ်ခန်း ဖြစ်သည် — သင်ယူရန်နှင့် ကြည့်ရှုရန် ကောင်းသည်။ **AWS CLI** (သင့် **terminal** ထဲက **aws**) သည် **console** လုပ်နိုင်သမျှအားလုံးကို **script** ရေး၍ရအောင် လုပ်ပေးသည်။ **Console** မှ စတင်ပြီး **CLI** သို့ တက်လှမ်းသွားမည် — အကြောင်းမှာ အဆင့် ၂ တွင် ဤနေရာ၌ လက်ဖြင့်လုပ်သမျှအားလုံးကို **automate** လုပ်မည် ဖြစ်သောကြောင့်တည်း။

Lab 1 (အပတ်စဉ် ၅-၆): EC2 — သင့်ပထမဆုံး server

EC2 (Elastic Compute Cloud) သည် **instance** ဟုခေါ်သော **virtual machine** များကို ငှားရမ်းပေးသည်။ ဤသည်မှာ **cloud engineering** ၏ မူလဗီဇ လုပ်ရပ်ပင် — အင်တာနက်ပေါ်ရှိ **Linux server** အစစ်တစ်လုံး၊ ၆၀ စက္ကန့်အတွင်း၊ အခမဲ့။

ရည်မှန်းချက်။ **Linux server** တစ်လုံး **launch** လုပ်ပါ။ **SSH** ဝင်ပါ။ ကမ္ဘာကြီးဆီ **web page** တစ်ခု ဝန်ဆောင်မှုပေးအောင်လုပ်ပါ။ ထို့နောက် ဖျက်ဆီးပစ်ပါ။

လုပ်ဆောင်ရန် အဆင့်များ။ 1. **Console** → **EC2** → **Launch instance**။ နာမည်ပေးပါ။ **AMI (Amazon Machine Image)** — သင့် **server** စတင်လည်ပတ်ရာ **disk** ပုံစံခွက်) အဖြစ် **Amazon Linux 2023** နှင့် **instance type** (အရွယ်အစား၊ ဤအမျိုးအစားများသည် **Free Tier**) အဖြစ် **t2.micro** သို့မဟုတ် **t3.micro** ကို ရွေးပါ။ 2. **Key pair** တစ်ခု ဖန်တီးပါ။ . **pem private-key** ဖိုင်တစ်ဖိုင် **download** ဆင်းလာမည်။ ၎င်းသည် **Module 1** မှ သင့် **SSH** သော့ပင် — စောင့်ရှောက်ပါ။ **chmod 400** လုပ်ပါ။ 3. **Network settings** တွင် **SSH (port 22)** ကို “**My IP**” မှသာ ခွင့်ပြုပါ — ဤ **security-group** စည်းကမ်း၏ အဓိပ္ပာယ်ကို ယခု သင်တိတိကျကျ သိပြီ — ပြီးလျှင် **HTTP (port 80)** ကို မည်သည့်နေရာမှမ

ဆို ခွင့်ပြုပါ။ 4. Launch လုပ်ပြီး “running” ဖြစ်သည်အထိ စောင့်ပါ။ public IP ကို ကူးယူပြီး သင့် terminal မှာ `ssh -i mykey.pem ec2-user@<public-ip>`။ အသက်တစ်ချက် ရှုလိုက်ပါ — သင်သည် AWS data center တစ်ခု အတွင်းရှိ ကွန်ပျူတာတစ်လုံးထဲ ရောက်နေပြီ။ 5. Server ပေါ်တွင် `sudo dnf install -y nginx && sudo systemctl start nginx && sudo systemctl enable nginx`။ ထို့နောက် `http://<public-ip>` ကို browser ဖြင့်သွားပါ — ကမ္ဘာကြီးကို အဖြေပေးနေသည်မှာ သင့် web server ပင်။ 6. မူရင်း page ကို အစားထိုးပါ `echo "<h1>Built by me, on EC2</h1>" | sudo tee /usr/share/nginx/html/index.html`။ Refresh လုပ်ပါ။ Journal အတွက် screenshot ရိုက်ပါ။ 7. Engineer တစ်ဦးလို လှည့်လည်စူးစမ်းပါ `top`၊ `df -h`၊ page ကို refresh လုပ်နေစဉ် `sudo tail -f /var/log/nginx/access.log` — သင်ကိုယ်တိုင်၏ ဝင်ရောက်ကြည့်ရှုမှုများ log ထဲ ရောက်လာသည်ကို ကြည့်ပါ။

သင်ယူရရှိသည့်အရာ: AMI များ၊ instance type များ၊ key pair များ၊ security group များကို လက်တွေ့သုံးခြင်း၊ server အစစ်တစ်လုံးဆီ SSH ဝင်ခြင်း၊ Linux service တစ်ခု ထည့်သွင်းလည်ပတ်ခြင်း၊ ၎င်း၏ log များ ဖတ်ခြင်း — တစ်နည်းဆိုသော် ဤအလုပ်၏ နေ့စဉ် ကိုယ်လက်လှုပ်ရှားမှုများ။

Teardown: EC2 → instance ကိုရွေး → Instance state → **Terminate**။ “Terminated” ဟု ပြသည်ကို အတည်ပြုပါ။ Free Tier က micro instance တစ်လုံးကို တစ်လလျှင် နာရီ ၇၅၀ ပေးထားသဖြင့် ဖွင့်ထားခဲ့လည်း ငွေမကုန်နိုင်ပါ — သို့သော် ဘာပဲဖြစ်ဖြစ် ဖြိုချပါ။ အလေ့အထကပင် အဓိကရည်ရွယ်ချက် ဖြစ်သည်။

Lab 2 (အပတ်စဉ် ၇): S3 — ဘယ်တော့မှ မပျောက်သော ဖိုင်များ

S3 (Simple Storage Service) သည် object storage ဖြစ်သည် — ဖိုင်များအတွက် အောက်ခြေမရှိ၊ eleven-nines (ကိုးဂဏန်း ၁၁ လုံး) ခံနိုင်ရည်ရှိသော ခြင်းတောင်းကြီး။ “ဖိုင်တွေ ဘယ်မှာထားမလဲ” ၏ ပုံသေအဖြေဖြစ်ပြီး — မှားယွင်းစွာ ပြင်ဆင်မိလျှင် သမိုင်းတွင် နာမည်အကြီးဆုံး ဒေတာပေါက်ကြားမှုများ၏ ဇာစ်မြစ်လည်း ဖြစ်သည် — ထို့ကြောင့် ဤ lab သည် တစ်ဝက်က storage၊ တစ်ဝက်က security ဖြစ်သည်။

ရည်မှန်းချက်: bucket တစ်ခု ဖန်တီးပါ။ CLI မှ ကိုင်တွယ်ပါ။ static website သေးသေးလေးတစ်ခု host လုပ်ပါ။ ပြီးလျှင် “public bucket” ၏ အဓိပ္ပာယ်ကို တိတိကျကျ နားလည်ပါ။

လုပ်ဆောင်ရန် အဆင့်များ: 1. Console → S3 → Create bucket (နာမည်များသည် ကမ္ဘာလုံးအတိုင်းအတာဖြင့် ထပ်၍မရ — `yourname-lab-2026` ဆိုလျှင် ရသည်)။ **Block Public Access** သည် မူလအတိုင်း ဖွင့်ထားသည်ကို သတိပြုပါ။ Console မှတစ်ဆင့် ဖိုင်တစ်ဖိုင် upload တင်ပြီး ပြန် download ဆွဲပါ။ 2. AWS CLI ကို သင့်စက်တွင် ထည့်သွင်းပြီး သင့် IAM admin user အတွက် ဖန်တီးထားသော **access key** တစ်ခုဖြင့် `aws configure` ကို လည်ပတ်ပါ (access key ဆိုသည်မှာ API အတွက် program သုံး username+password ဖြစ်သည် — password တစ်ခုလို သဘောထားပါ။ code ထဲ ဘယ်တော့မှ မထည့်ပါနှင့်။ ဤစည်းကမ်းကို Lab 5 တွင် အသည်းစွဲ နားလည်သွားမည်)။ 3. သင့် terminal မှာ `aws s3 ls`၊ `aws s3 cp notes.txt s3://yourname-lab-2026/`၊ `aws s3 sync ./myfolder s3://yourname-lab-2026/backup/`။ ကွာခြားချက်ကို ခံစားကြည့်ပါ — console သည် အလည်လာခြင်း၊ CLI သည် engineering လုပ်ခြင်း။ 4. Static website: ဒုတိယ bucket တစ်ခုလုပ်ပါ။ static website hosting ကို ဖွင့်ပါ။ `index.html` တစ်ဖိုင် upload တင်ပါ။ ပြီးလျှင် စာရွက်စာတမ်းများပါ **public-read bucket policy** (JSON permission စာတမ်း — တစ်ကြောင်းချင်း ဖတ်ပါ — မည်သူ့က မည်သည့် bucket ကို ဘာ လုပ်ခွင့်ရသနည်း) ကို ထည့်ပါ။ သင့် page သည် server တစ်လုံးမှမပါဘဲ အင်တာနက်ပေါ် ရောက်နေပြီ။ 5. **Storage class** များ (Standard →

Infrequent Access → Glacier: GB လျှင် ဈေးသက်သာလာပြီး ပြန်ထုတ်ရန် နှေး/ဈေးကြီးလာသည်) ကို လေ့လာပြီး **lifecycle rule** တစ်ခု (“ရက် ၃၀ ကြာလျှင် object များကို IA သို့ ရွှေ့ပါ”) သတ်မှတ်ပါ — အလိုအလျောက် ကုန်ကျစရိတ်စီမံခန့်ခွဲမှု၏ ပထမဆုံး အရသာ။

သင်ယူရရှိသည့်အရာ: object storage နှင့် disk များ ကွာခြားပုံ၊ CLI နှင့် access key များ၊ bucket policy များနှင့် public access (မတော်တဆမဟုတ်ဘဲ တမင်တကာ)၊ ကုန်ကျစရိတ်ထိန်းကိရိယာအဖြစ် storage tier များ။

Teardown: bucket နှစ်ခုလုံးကို အလွတ်ဖြစ်အောင်လုပ်ပြီး ဖျက်ပါ။ ပြီးလျှင် — အရေးကြီးသည် — မသုံးတော့သော access key ဟူသမျှကို ပိတ်ပါ။ S3 ၏ အခမဲ့ခွင့်ပြုချက်သည် သေးသည် (5 GB) — သို့သော် ဤဖိုင်များသည် kilobyte သာသာမျှသာ။ ထပ်ပြောရလျှင် စည်းကမ်းကပင် ရည်ရွယ်ချက် ဖြစ်သည်။

Lab 3 (အပတ်စဉ် ၈-၉): VPC — သင်ပိုင်ဆိုင်သော network

VPC (Virtual Private Cloud) သည် AWS ၏ network အတွင်း သင့်ကိုယ်ပိုင် သီးသန့်ခြခတ်ထားသော အပိုင်း ဖြစ်သည်။ ယခုအချိန်ထိ သင်သည် မူလ (default) VPC ကို မကြည့်ဘဲ သုံးနေခဲ့သည်။ Engineer များကမူ ကိုယ်ပိုင်တည်ဆောက်ကြသည် — အကြောင်းမှာ “*the database sits in a private subnet with no internet route*” (database က အင်တာနက်လမ်းကြောင်းမရှိတဲ့ private subnet ထဲမှာ ရှိတယ်) ဟူသော စာကြောင်းသည် cloud လုံခြုံရေး၏ တစ်ဝက်ဖြစ်ပြီး ယခု သင်ကိုယ်တိုင် လက်ဖြင့် အမှန်ဖြစ်အောင် လုပ်တော့မည် ဖြစ်သောကြောင့်တည်း။

ရည်မှန်းချက်: two-tier network တစ်ခု တည်ဆောက်ပါ — web server အတွက် public subnet၊ နောင်လာမည့် database အတွက် private subnet — ပြီးလျှင် private အပိုင်းကို အင်တာနက်မှ လက်လှမ်းမမီနိုင်ကြောင်း သက်သေပြပါ။

လုပ်ဆောင်ရန် အဆင့်များ: 1. Console → VPC → Create VPC။ လိပ်စာအုပ်စု 10.0.0.0/16 ပေးပါ — **CIDR notation** ဖြစ်ပြီး /16 ၏ အဓိပ္ပာယ်မှာ “ပထမ 16 bit သေချာနေပြီး ကျန်တာ ကျွန်တော့်ဟာ” — private လိပ်စာ ၆၅,၅၃၆ ခု။ 2. **Subnet** နှစ်ခု ဖန်တီးပါ: 10.0.1.0/24 (public၊ AZ-a ထဲတွင်) နှင့် 10.0.2.0/24 (private၊ AZ-b ထဲတွင်)။ Subnet ဆိုသည်မှာ VPC အတွင်းရှိ ပိုသေးသော အုပ်စုဖြစ်ပြီး AZ တစ်ခုတည်းတွင်သာ နေထိုင်သည်။ 3. **Internet Gateway** (VPC ၏ အင်တာနက်တံခါး) တစ်ခု ဖန်တီးပြီး ချိတ်တွဲပါ။ 0.0.0.0/0 → internet gateway (“local မဟုတ်တာမှန်သမျှ အင်တာနက်တံခါးဆီသွား”) ဟူသော စည်းကမ်းပါ **route table** တစ်ခု ဖန်တီးပြီး public subnet နှင့်သာ ချိတ်ဆက်ပါ။ Private subnet မှာ local route သာ ကျန်သည် — ထို route မရှိခြင်းသည်ပင် လုံခြုံရေး ဖြစ်သည်။ 4. Subnet တစ်ခုစီတွင် micro EC2 instance တစ်လုံးစီ launch လုပ်ပါ (public ကို public IP ဖြင့်၊ private ကို မပါဘဲ)။ 5. သက်သေ — public instance ဆီ SSH ဝင်ပါ — ရသည်။ Private instance ၏ လိပ်စာကို သင့် laptop မှ စမ်းပါ — ထာဝရ hang နေမည်။ ပြီးလျှင် ဘာကြောင့်ဆိုသည်ကို ယခု သင်တိတိကျကျ ပြောနိုင်ပြီ။ ထို့နောက် public instance မှ တစ်ဆင့် private instance ဆီ SSH ဝင်ပါ (VPC အတွင်းမှ လက်လှမ်းမီသည်) — public စက်သည် **bastion host** အဖြစ် ဆောင်ရွက်နေခြင်းဖြစ်ပြီး ၎င်းသည် သင်ကိုယ်တိုင် တည်ဆောက်ရင်း ရှာဖွေတွေ့ရှိလိုက်သော လုပ်ငန်းနယ်ပယ်စံ ပုံစံ (pattern) တစ်ခု ဖြစ်သည်။ 6. ရှာဖွေဖတ်ရှုပြီး journal ထဲမှတ်ရန် အပိုအယူအဆ — **NAT gateway** သည် private စက်များကို အပြင်မှ လက်လှမ်းမမီစေဘဲ အပြင်သို့ ထွက် ဆက်သွယ်နိုင်စေသည် (update များအတွက်) — သို့သော် နာရီအလိုက် ငွေကောက်သောကြောင့် ဖတ်ရုံသာဖတ်ပါ။ မတည်ဆောက်ပါနှင့်။

သင်ယူရရှိသည့်အရာ: CIDR၊ subnet များ၊ route table များ၊ internet gateway များ၊ public/private ခွဲခြားမှု၊ bastion host များ — cloud အလုပ်ခေါ်စာတိုင်းပါ networking အချက်ကို ကြွက်သားမှတ်ဉာဏ် (muscle memory) အဖြစ်။

Teardown: instance နှစ်လုံးကို အရင် terminate လုပ်ပါ။ ထို့နောက် VPC ကို ဖျက်ပါ (subnet၊ route table နှင့် gateway များပါ လိုက်ပါရှင်းသွားသည်။)။ EC2 တွင် “running” ဟုပြနေသောအရာ မရှိကြောင်း စစ်ဆေးပါ။

Lab 4 (အပတ်စဉ် ၁၀): RDS — ချောမြူထိန်းသိမ်းစရာမလိုသော database

RDS (Relational Database Service) သည် managed database (စီမံပေးပြီးသား database) ဖြစ်သည် — AWS က database engine (PostgreSQL၊ MySQL...) ကို လည်ပတ်ပေးပြီး backup၊ patch နှင့် failover များကို တာဝန်ယူကာ သင်ကမူ ဒေတာနှင့် query များကို ပိုင်ဆိုင်သည်။ “Managed service” ဆိုသည်မှာ cloud ၏ အဓိက အပေးအယူပင် — ထိန်းချုပ်မှုအချို့ကို စွန့်လွှတ်ပြီး ထူးခြားမှုမရှိသော လုပ်အားများစွာကို လွှဲပြောင်းခြင်း — ဤ lab သည် ထိုအပေးအယူကို ကိုယ်တိုင်ခံစားရမည့်နေရာ ဖြစ်သည်။

ရည်မှန်းချက်: PostgreSQL database တစ်ခုကို private subnet ထဲတွင် launch လုပ်ပါ။ EC2 instance တစ်လုံးမှ ချိတ်ဆက်ပါ။ backup နှင့် multi-AZ ကို နားလည်ပါ — ထို့နောက် အမြန် ပြန်ဖြူချပါ။ အကြောင်းမှာ RDS သည် မတော်တဆ ဖွင့်ထားခဲ့မိရန် အလွယ်ဆုံး lab ဖြစ်သောကြောင့်တည်း။

လုပ်ဆောင်ရန် အဆင့်များ: 1. Lab 3 ၏ VPC ကို အမြန်ပြန်တည်ဆောက်ပါ (အလွတ်ပြန်ဆောက်သည့် ဒုတိယအကြိမ် — ၎င်းသည် တမင်ရည်ရွယ်ထားသော spaced repetition ဖြစ်သည်)။ ထို့ပြင် အခြား AZ တစ်ခုတွင် ဒုတိယ private subnet တစ်ခု ထပ်ထည့်ပါ — အကြောင်းမှာ RDS သည် AZ နှစ်ခုကို လွှမ်းမိုးသော **subnet group** တစ်ခု လိုအပ်သောကြောင့်တည်း။ 2.

Console → RDS → Create database → PostgreSQL → **Free tier template** (၎င်းက

db.t3.micro/db.t4g.micro၊ single-AZ ကို ကြိုရွေးပေးသည်။)။ Master password သတ်မှတ်ပါ။ သင့် VPC ထဲတွင် ထားပါ။ **Public access: No**၊ port 5432 ကို web server ၏ security group မှသာ ခွင့်ပြုသော security group တစ်ခုထဲတွင် — IP တစ်ခုကို မဟုတ်ဘဲ အခြားစည်းကမ်းတစ်ခုကို ရည်ညွှန်းသော စည်းကမ်း။ လှပပြီး စံနည်းလည်း ဖြစ်သည်။ 3.

Public subnet ထဲတွင် micro EC2 တစ်လုံး launch လုပ်ပါ။ postgresql client ကို ထည့်သွင်းပြီး ချိတ်ဆက်ပါ: `psql -h <rds-endpoint> -U postgres`။ **Endpoint** သည် IP မဟုတ်၊ DNS နာမည်ဖြစ်သည် — AWS က

အောက်ခံစက်ကို ရွှေ့နိုင်ပြီး နာမည်က လိုက်ပါသွားသည်။ (Module 2 က အကျိုးပြန်ပေးနေပြီ။) 4. psql prompt တွင်: ဇယားတစ်ခု ဖန်တီးပါ။ အတန်းသုံးတန်း ထည့်ပါ။ ပြန်ဆွဲထုတ်ကြည့်ပါ။ ယနေ့အတွက် SQL အနက်ရှိုင်းစရာ မလိုပါ — ထိတွေ့ဖူးရန်သာ လိုသည်။ 5. လှည့်ကြည့်ရုံသာ၊ မဖွင့်ပါနှင့် — **automated backups setting** (ညစဉ် snapshot + log များမှ အချိန်တစ်ခု

အလိုက် ပြန်ယူနိုင်ခြင်း) နှင့် **Multi-AZ option** (အခြား AZ တစ်ခုတွင် တိုက်ရိုက် standby တစ်ခုနှင့် အလိုအလျောက်

failover — ကုန်ကျစရိတ် နှစ်ဆနီးပါး၊ ထို့ကြောင့် production က yes၊ lab က no)။ **Snapshot** တစ်ခု ကိုယ်တိုင်ရိုက်ပါ။

console ထဲမှာ ရှာတွေ့ပါ။ ၎င်းမှ မိတ္တူတစ်ခု ပြန်တည်ဆောက်နိုင်ကြောင်း နားလည်ပါ။ 6. ဆယ်မိနစ်တန်သော journal မေးခွန်း

— ဤနေရာတွင် AWS သည် သင့်အတွက် ဘာတွေ တိတိကျကျ လုပ်ပေးနေသနည်း — မဟုတ်လျှင် ညသန်းခေါင် ၂ နာရီတွင် သင်ကိုယ်တိုင် လုပ်ရမည့်အရာများ။ (Patching၊ backup များ၊ failover၊ hardware)။ ထိုအဖြေသည်ပင် managed-

services အင်တာဗျူးအဖြေ ဖြစ်သည်။

သင်ယူရရှိသည့်အရာ: managed database များ၊ subnet group များ၊ security-group ချင်း ရည်ညွှန်းသော စည်းကမ်းများ၊ endpoint များ၊ snapshot များ၊ multi-AZ/failover — ထို့ပြင် ယုံကြည်စိတ်ချရမှုကို ထိန်းချုပ်မှုဖြင့် ဝယ်ယူခြင်း၏ တိုက်ရိုက်သရုပ်ပြမှုတစ်ခု။

Teardown: RDS instance ကို ဖျက်ပါ (lab အတွက် နောက်ဆုံး snapshot ကို ငြင်းပါ — production ဆိုလျှင်တော့ ရိုက်ထားမည်ကို မှတ်သားပါ)။ ကိုယ်တိုင်ရိုက်ထားသော snapshot ကို ဖျက်ပါ (snapshot များသည် သိုလှောင်ခအတွက် ငွေ

ကောက်သည်!)။ EC2 ကို terminate လုပ်ပါ။ VPC ကို ဖျက်ပါ။ နောက်တစ်နေ့တွင် billing dashboard ကို စစ်ပါ — ၎င်းကို အပတ်စဉ်ဖတ်ခြင်းသည် ယခုမှစတင်နေပြီဖြစ်သော အဆင့် ၃ အလေ့အထတစ်ခု ဖြစ်သည်။

Lab 5 (အပတ်စဉ် ၁၁-၁၂): IAM — မည်သူက ဘာလုပ်ခွင့်ရှိသနည်း

IAM (Identity and Access Management) သည် မည်သည့်လူများနှင့် မည်သည့် program များက မည်သည့် resource များကို ဘာလုပ်ခွင့်ရှိသည်ကို ဆုံးဖြတ်သည်။ ၎င်းသည် ဘာမှ ငွေမကောက်၊ ဘာမှ မတည်ဆောက်ပေး — သို့သော် AWS တွင် အစစ်ဆေးအခံရဆုံး၊ အင်တာဗျူးတွင် အမေးအခံရဆုံး၊ လုံခြုံရေးပေါက်ကြားမှုများတွင် အပါဝင်ဆုံး ဝန်ဆောင်မှု ဖြစ်သည်။ အလုပ်ခေါ်စာများရှိ လုံခြုံရေးအချက်များ (“least-privilege access controls”၊ “IAM policies”) ဆိုသည်မှာ ဤ lab ကို ဆိုလိုခြင်း ဖြစ်သည်။

ရည်မှန်းချက်: user များ၊ group များ၊ policy များနှင့် — အရေးအကြီးဆုံးဖြစ်သော — role တစ်ခု ဖန်တီးပြီး AWS က သင့် ကို ငြင်းပယ်သည်ကို ကိုယ်တိုင်ခံစားခြင်းဖြင့် least privilege ကို အသုံးပြုစွဲအောင် သင်ယူပါ။

လုပ်ဆောင်ရန် အဆင့်များ: 1. အယူအဆများ အရင်၊ ငါးမိနစ် — **user** ဆိုသည်မှာ လူသား သို့မဟုတ် program တစ်ခုအတွက် identity တစ်ခု။ **Group** သည် user များကို စုစည်းသည်။ **Policy** ဆိုသည်မှာ permission များ ပေးအပ်သော JSON စာတမ်း (“allow s3:GetObject on arn:aws:s3:::my-bucket/*”)။ **Role** ဆိုသည်မှာ policy များပါ သော်လည်း password မရှိသော identity တစ်ခုဖြစ်ပြီး ယုံကြည်ရသော အဖွဲ့အစည်းတစ်ခုက ယာယီ assume (ဆောင်းယူ) လုပ်သည် — server များနှင့် service များသည် သိမ်းဆည်းထားသော လျှို့ဝှက်ချက် တစ်ခုမှမပါဘဲ permission ရယူပုံ ဖြစ်သည်။ 2. AWS-managed ReadOnlyAccess policy ပါသော group တစ်ခုထဲတွင် readonly-rita ဟူသော user တစ်ဦး ဖန်တီးပါ။ Private browser window တစ်ခုတွင် သူမအဖြစ် login ဝင်ပါ — အားလုံးကို မြင်နိုင်သော်လည်း create/delete ခလုတ်တိုင်း အတိအလင်း ငြင်းပယ်ချက်ဖြင့် ကျရှုံးမည်။ ထို error တစ်ခုကို အပြည့်ဖတ်ပါ — “not authorized to perform X on Y” ကို ဖတ်ရှုနားလည်တတ်ခြင်းသည် နေ့စဉ်အလုပ် ကျွမ်းကျင်မှုတစ်ခု ဖြစ်သည်။ 3. JSON editor ထဲတွင် သင့်ပထမဆုံး custom policy ကို ရေးပါ — bucket တစ်ခုတည်းပေါ်တွင် s3:ListBucket နှင့် s3:GetObject ကို ခွင့်ပြုပါ။ User အသစ်တစ်ဦးနှင့် ချိတ်တွဲပါ။ ထို bucket ကို ဖတ်နိုင်ပြီး အခြားဘာမှ မလုပ်နိုင်ကြောင်း စစ်ဆေးပါ။ ယခုအခါ least privilege ကို အဓိပ္ပာယ်ဖွင့်ရုံသာမက အကောင်အထည်ဖော်ပြီးပြီ။ 4. Role — အသီးအပွင့်အပိုင်း — EC2 အတွက် AmazonS3ReadOnlyAccess ပါသော role တစ်ခု ဖန်တီးပါ။ ထို role ချိတ်တွဲထားသော micro instance တစ်လုံး launch လုပ်ပါ။ SSH ဝင်ပြီး aws s3 ls ကို လည်ပတ်ပါ — **စက်ပေါ်တွင် access key တစ်ခုမှ မရှိဘဲ** အလုပ်ဖြစ်သည်။ Instance က role ကို assume လုပ်ပြီး သက်တမ်းတို credential များကို အလိုအလျောက် ရယူနေခြင်း ဖြစ်သည်။ ဤသည်မှာ AWS တွင် အရေးအကြီးဆုံး လုံခြုံရေးပုံစံ ဖြစ်သည် — စက်များအတွက် role၊ စက်များပေါ်တွင် key ဘာယ်တော့မှ မထား၊ နှစ်ကြိမ် ထပ်ပြောပါ။ 5. သင့်ကိုယ်ပိုင် account ကို ကျွမ်းကျင်သူတစ်ဦးလို စစ်ဆေးပါ — root မှာ MFA ရှိရဲ့လား (အပတ်စဉ် ၃)။ ရက် ၉၀ ထက် အိုဟောင်းသော access key ရှိသလား။ User တစ်ဦးဦးမှာ သုံးသည်ထက် ပိုသော permission ရှိသလား။ IAM credential report ကို ထုတ်ပြီး ဖတ်ပါ။ ဤဆယ်မိနစ် ဓလေ့ကို လစဉ်လုပ်ခြင်းသည် အဆင့် ၃ ၏ IAM ကျန်းမာရေး checklist မွေးဖွားလာခြင်း ဖြစ်သည်။

သင်ယူရရှိသည့်အရာ: user/group/policy/role များ၊ policy JSON ဖတ်ခြင်းနှင့် ရေးခြင်း၊ စမ်းသပ်မှုဖြင့် သက်သေပြထားသော least privilege၊ key မဟုတ် role ဟူသောမူ၊ နှင့် သင့်ပထမဆုံး လုံခြုံရေးစစ်ဆေးမှု။

Teardown: စမ်းသပ် user များနှင့် ၎င်းတို့၏ credential များကို ဖျက်ပါ။ instance ကို terminate လုပ်ပါ။ role အကြောင်း ဗဟုသုတကို ထပ်မံ သိမ်းထားပါ။

အဆင့် ၁ အတွက် ဝေါဟာရများ (ဇယားတစ်ခုတည်း၊ lab ငါးခုလုံးအတွက်):

ဝေါဟာရ	အဓိပ္ပာယ်
Region / Availability Zone	AWS data center များ၏ ပထဝီအလိုက် အစုအဝေး / ၎င်းအတွင်းရှိ သီးခြားခွဲထားသော data center တစ်ခု။ အလေးအနက် စနစ်များသည် AZ နှစ်ခုကို လွှမ်းခြုံသည်။
EC2 / instance	Virtual machine ငှားရမ်းရေး ဝန်ဆောင်မှု / ငှားထားသော server တစ်လုံး။
AMI	Amazon Machine Image — instance တစ်လုံး စတင်လည်ပတ်ရာ disk ပုံစံခွက်။
Instance type	သင်ရွေးထားသော အရွယ်အစား/spec (t3.micro = အသေးစား၊ Free Tier)။
Security group	Resource တစ်ခုချင်း firewall — မည်သည့် port များ၊ မည်သည့် source များမှ။
Key pair	သင့် instance ဆီရောက်ရန် private/public SSH သော့များ။
S3 / bucket / object	Object storage / အမည်ပေးထားသော ကွန်တိန်နာတစ်ခု / သိမ်းထားသော ဖိုင်တစ်ဖိုင်။
Storage class / lifecycle rule	Object များအတွက် ဈေးနှုန်း-အမြန်နှုန်း အဆင့် / သက်တမ်းအလိုက် ပိုဈေးသက်သာသော အဆင့်များသို့ အလိုအလျောက် ရွှေ့ပေးသော စည်းကမ်း။
Bucket policy	Bucket တစ်ခုကို မည်သူက ဘာလုပ်ခွင့်ရှိသည်ကို ဖော်ပြသော JSON စာတမ်း။ Public ဖြစ်နေသောများက သတင်းစာမျက်နှာတက်တတ်သည်။
AWS CLI / access key	AWS ဆီသို့ terminal မျက်နှာပြင် / ၎င်းအတွက် program သုံး credential များ (password တစ်ခုလို သဘောထားပါ)။
VPC / subnet	သင့်ကိုယ်ပိုင် network အပိုင်း / ၎င်း၏ ပိုသေးသောအုပ်စု — AZ တစ်ခုတည်းတွင် နေပြီး public သို့မဟုတ် private။
CIDR	လိပ်စာအုပ်စု သင်္ကေတ: 10.0.0.0/16 = “ပထမ 16 bit သေချာနေပြီး လိပ်စာ ၆၅,၅၃၆ ခု ကျွန်တော့်ဟာ”။
Internet gateway / route table	VPC ၏ အင်တာနက်တံခါး / traffic ကို ဘယ်ပိုမည်ကို ဆုံးဖြတ်သော စည်းကမ်းများ။ Route မရှိ = လက်လှမ်းမမီ = လုံခြုံ။

ဝေါဟာရ	အဓိပ္ပာယ်
NAT gateway	Private စက်များကို အပြင်မှ လက်လှမ်းမမီစေဘဲ အပြင်ထွက်ခေါ်ဆိုနိုင်စေသည်။ နာရီအလိုက် ငွေကောက်သည် — သိထားပါ။ အလကား ဖွင့်မထားပါနှင့်။
Bastion host	Private စက်များဆီရောက်ရန် SSH ဖြင့် ဖြတ်သန်းရသော ခိုင်ခံ့အောင်လုပ်ထားသည့် public စက်။
RDS / endpoint	Managed relational database ဝန်ဆောင်မှု / သင်ချိတ်ဆက်ရသော DNS နာမည်။
Snapshot / Multi-AZ / failover	အချိန်တစ်ခုအလိုက် မိတ္တူ / ဒုတိယ AZ ရှိ တိုက်ရိုက် standby / ၎င်းဆီသို့ အလိုအလျောက် ပြောင်းခြင်း။
IAM user / group / policy / role	Identity / identity အစုအဝေး / JSON permission ပေးအပ်မှု / password မရှိသော assume လုပ်နိုင်သည့် identity — စက်များ permission ရယူပုံ။
Least privilege	ရွှေစည်းကမ်း — အလုပ်ဖြစ်ရုံ အနည်းဆုံး permission သာ၊ ထို့ထက်ပိုပြီး ဘာမှမပေး။
Managed service	AWS က ထူးခြားမှုမရှိသော လုပ်အား (patching၊ backup၊ failover) ကို လည်ပတ်ပေးသည်။ သင်က ဒေတာနှင့် ဆုံးဖြတ်ချက်များကို ဆက်ပိုင်သည်။

ဤအဆင့်အတွက် ဗီဒီယိုများ:

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
Getting Started with EC2	AWS Developers (official)	27 min	https://www.youtube.com/watch?v=nJ-djerESW0
Introduction to Amazon S3	Amazon Web Services (official)	~5 min	https://www.youtube.com/watch?v=ecv-19sYL3w
AWS Networking Basics — VPC & Subnets	KodeKloud (Lab 3 ပြီးမှ ပြန်ကြည့်ပါ — ယခုအခါခံစားမှု ကွာသွားမည်)	~30 min	https://www.youtube.com/watch?v=QM63dyA_4Pc
The AWS Shared Responsibility Model	Digital Cloud Training	4 min	https://www.youtube.com/watch?v=ESPBBEK-cvo

အသံထွက်၍ ပြောပါ: “It’s in a private subnet; there’s no route to the internet gateway.” (private subnet ထဲမှာ ရှိတယ်၊ internet gateway ဆီ route မရှိဘူး။) · “The security group only allows 5432 from the web tier’s security group.” (Security group က 5432 ကို web tier ရဲ့ security group ကနေပဲ ခွင့်ပြုတယ်။) · “The instance uses a role — there are no keys on the box.” (Instance က role သုံးတယ် — စက်ပေါ်မှာ key မရှိဘူး။) · “Terminate it when you’re done; nothing idles in this account.” (ပြီးရင် terminate လုပ်ပါ — ဒီ account ထဲမှာ ဘာမှ အလကား လည်မနေရဘူး။)

Milestone — အဆင့် ၁ အဆုံး: စိန်ခေါ်မှုကြီး (gauntlet) တည်ဆောက်ခြင်း။ တစ်ထိုင်တည်း၊ အလွတ် — public/private subnet ပါ VPC → EC2 web server (public၊ role ချိတ်ထား၊ page တစ်ခု ဝန်ဆောင်နေ) → RDS PostgreSQL (private၊ web server မှသာ လက်လှမ်းမီ) → instance က ၎င်း၏ role ဖြင့် ဖတ်နိုင်သော S3 bucket တစ်ခု — ထို့နောက် အားလုံးကို သန့်သန့်ရှင်းရှင်း ပြန်ဖြူချပါ။ သုံးနာရီအောက်ဖြင့် ပြီးလျှင် အဆင့် ၂ အတွက် အသင့်ဖြစ်ပြီ။ ထို့ပြင် — **CLF-C02** ပြင်ဆင်မှုကို ယခုစတင်ပါ (freeCodeCamp သင်တန်းအပြည့်အစုံ — <https://www.youtube.com/watch?v=7HKot-brXFE> — ကို ပြန်လှန်လေ့လာမှုအဖြစ် 1.25x နှုန်းဖြင့်။ ဂန္ထဝင် ၁၄ နာရီ edition မှာ <https://www.youtube.com/watch?v=NhDYbskXRgc>) ပြီးလျှင် **AWS Cloud Practitioner** စာမေးပွဲကို အပတ်စဉ် ၁၄-၁၆ ဝန်းကျင်တွင် ဖြေပါ။

အဆင့် ၂ — AUTOMATION (အပတ်စဉ် ၁၃-၂၀)

ဤအဆင့်၏ အယူအဆ: အဆင့် ၁ တွင် click နှိပ်ခဲ့သမျှအားလုံးကို ယခု code ဖြင့် လုပ်တော့မည်။ Console ကို click နှိပ်ခြင်းအတွက် လုပ်ငန်းနယ်ပယ်၏ နူးညံ့သော လှောင်ပြောင်စကားမှာ **ClickOps** ဖြစ်ပြီး ၎င်းကို အစားထိုးမည့်အရာအတွက် အလုပ်ခေါ်စာစကားစုမှာ “*design, develop, and deploy cloud infrastructure using infrastructure as code*” ဖြစ်သည်။ ဤအဆင့်သည် AWS ကို သုံးဖူးသူ တစ်ဦးနှင့် ၎င်းကို လည်ပတ်ရန် ကုမ္ပဏီက လခပေးမည့်သူတစ်ဦး၏ ကွာခြားချက် ဖြစ်သည်။

Module 6 (အပတ်စဉ် ၁၃-၁၅): Scripting — ops အတွက် Bash နှင့် Python၊ ထို့ပြင် Git

Bash scripting ဆိုသည်မှာ Module 1 command များကို ဖိုင်တစ်ဖိုင်ထဲ ထည့်ပြီး ကွန်ပျူတာက အမှားအယွင်းမရှိ ထပ်ခါ လုပ်ပေးအောင် လုပ်ခြင်း ဖြစ်သည်။ ဤအစီအစဉ်အတိုင်း သင်ယူပါ — **variable** များ (NAME="web-1")၊ **command substitution** (TODAY=\$(date +%F))၊ **if** (if [-f "\$FILE"]; then ... fi)၊ **loop** များ (for f in *.log; do gzip "\$f"; done)၊ **exit code** များ (command တိုင်းသည် အောင်မြင်လျှင် 0၊ ကျရှုံးလျှင် သုည မဟုတ်သောဂဏန်း ပြန်ပေးသည်။ && သည် အောင်မြင်မှသာ ဆက်ချိတ်သည် — script များသည် ၎င်းဖြင့် ဆုံးဖြတ်ချက်ချသည်)၊ နှင့် **argument** များ ဖတ်ခြင်း (\$1၊ \$2)။ ဤသည်တို့သည် လက်တွေ့ ops Bash ၏ ၉၀% ဖြစ်သည်။ ဤအပတ်တွင် သင့်ပထမဆုံး အမှန်တကယ် အသုံးဝင်သော script ကို ရေးပါ — **backup.sh** — **directory** တစ်ခုကို **tar** ဖြင့်ထုပ်ပါ။ **archive** ကို ယနေ့ရက်စွဲဖြင့် နာမည်ပေးပါ။ **aws s3 cp** ဖြင့် **bucket** တစ်ခုသို့ တင်ပါ။ ၇ ရက်ထက် အိုဟောင်းသော **local archive** များကို ဖျက်ပါ။ ပြီးလျှင် အောင်မြင်/ကျရှုံး စာကြောင်းတစ်ကြောင်းကို **log** ဖိုင်တစ်ဖိုင်ထဲ **echo** လုပ်ပါ။ ထို script တစ်

ခုတည်းသည် **variable** များ၊ **substitution**၊ **conditional** များ၊ **exit code** များနှင့် **CLI** တို့ကို အနုပညာလက်ရာတစ်ခုတည်းတွင် ပေါင်းစည်းထားပြီး — သင့် **résumé** ပေါ်တွင်လည်း အချက်တစ်ချက် ဖြစ်လာသည် (“**automated backups to S3**”)။

Python သည် **Bash** ၏ အစ်ကိုကြီး ဖြစ်သည် — **logic**၊ ဒေတာ သို့မဟုတ် **API** များနှင့် စကားပြောခြင်းပါဝင်သော မည်သည့်အရာအတွက်မဆို ပိုကောင်းသည်။ သင်လိုအပ်သည်မှာ **software-engineering** အနက်မဟုတ်၊ အလုပ်ဖြစ်သော **ops-Python** ဖြစ်သည် — **variable** များနှင့် **type** များ၊ **list** များနှင့် **dictionary** များ၊ **if/for**၊ **function** များ၊ ဖိုင်ဖတ်ခြင်း/ရေးခြင်း၊ **try/except error** ကိုင်တွယ်ခြင်း၊ နှင့် **pip** ဖြင့် **library** များ ထည့်သွင်းခြင်း။ ထို့နောက် **Python** အတွက် **AWS library** ဖြစ်သော **boto3** နှင့် မိတ်ဆက်ပါ: `import boto3; ec2 = boto3.client("ec2");`
`ec2.describe_instances()` — သင့်အဆင့် ၁ ဗဟုသုတသည် ရုတ်တရက် **program** ရေး၍ရသောအရာ ဖြစ်သွားသည်။ **audit.py** ကို ရေးပါ — **account** ထဲရှိ **EC2 instance** တိုင်းကို ၎င်း၏ နာမည်၊ **type**၊ **state** နှင့် **launch** လုပ်သည့်အချိန်တို့ဖြင့် စာရင်းပြပြီး ၂၄ နာရီထက်ကြာအောင် လည်နေသော မည်သည့်အရာအတွက်မဆို သတိပေးစာကြောင်း ပုံနှိပ်ပါ။ ဂုဏ်ယူပါ — အလုပ်ရှင်များ လခပေးဝယ်သော **cost-control** ကိရိယာအစစ်တစ်ခုကို သင်ရေးပြီးပြီ။

Git သည် **engineering** အလုပ်အားလုံး နေထိုင်ရာ **version-control** စနစ် ဖြစ်သည်။ အယူအဆများ — **repository** (သမိုင်းတစ်ခုလုံး မှတ်တမ်းတင်ထားသော **folder** တစ်ခု)၊ **commit** (**message** တစ်ခုပါသော သိမ်းဆည်းထားသည့် **snapshot** တစ်ခု)၊ **branch** (အပြိုင်အလုပ်လမ်းကြောင်းတစ်ခု)၊ **remote** (**GitHub** ပေါ်ရှိ မိတ္တူ)၊ နှင့် **pull request** (**branch** တစ်ခုကို ပြန်လည်သုံးသပ်ပြီး ပေါင်းစည်း (**merge**) ပေးရန် တောင်းဆိုခြင်း)။ နေ့စဉ်စက်ဝန်းမှာ **command** ခြောက်ခု — `git init`၊ `git clone`၊ `git status`၊ `git add`၊ `git commit -m "message"`၊ `git push`၊ `git pull`။ ယနေ့ပင် **GitHub account** တစ်ခု ဖွင့်ပါ။ **cloud-journey** ဟူသော **repo** တစ်ခု ဖန်တီးပါ။ ပြီးလျှင် ယခုမှစ၍ ဤသင်တန်းတွင် သင်ရေးသမျှ **script** နှင့် **config** တိုင်းကို ၎င်းထဲ **commit** လုပ်ပါ။ ဆယ်လအကြာတွင် ထို **commit** သမိုင်းသည် ဤသင်တန်းက သင်လုပ်နိုင်သည်ဟု ဆိုထားသမျှ၏ ရက်စွဲပါ။ လူသိရှင်ကြား သက်သေ ဖြစ်လာမည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Script	အပေါ်မှအောက် အစဉ်လိုက်လည်ပတ်သော command ဖိုင်တစ်ဖိုင် — automation ၏ အက်တမ်။
Variable / argument	Script ထဲရှိ အမည်ပေးထားသောတန်ဖိုး / လည်ပတ်ချိန်တွင် ထည့်ပေးလိုက်သောတန်ဖိုး (\$1)။
Exit code	Command တိုင်း၏ အောင်မြင် (0) သို့မဟုတ် ကျရှုံး (သုညမဟုတ်) အချက်ပြမှု — script များ ဆုံးဖြတ်ချက်ချပုံ။
Cron	Linux ၏ အချိန်ဇယားကိရိယာ — script တစ်ခုကို ညတိုင်း ၂ နာရီတွင်၊ ထာဝရ လည်ပတ်စေခြင်း။
Python / pip	Ops လောက၏ အကြိုက်ဆုံး programming ဘာသာစကား / ၎င်း၏ package ထည့်သွင်းကိရိယာ။

ဝေါဟာရ	အဓိပ္ပာယ်
boto3	AWS ကို မောင်းနှင်ရန် Python library — console ကို code အဖြစ်။
try/except	Python ၏ “ဒါကို ကြိုးစားကြည့်ပါ။ ကျရှုံးလျှင် ဒီဟာကို လုပ်ပါ” — script များ သိက္ခာရှိရှိ ကျရှုံးတတ်ပုံ။
Git / repository / commit	Version control / project တစ်ခု၏ မှတ်တမ်းတင်ထားသော သမိုင်း / message ပါသော သိမ်းဆည်းထားသည့် snapshot တစ်ခု။
Branch / merge	အပြိုင်အလုပ်လမ်းကြောင်းတစ်ခု / ၎င်းကို ပင်မလမ်းကြောင်းထဲ ပြန်ပေါင်းထည့်ခြင်း။
GitHub / remote / pull request	Hosting site / သင့် repo ၏ cloud မိတ္တူ / ပြန်လည်သုံးသပ် ဆွေးနွေးပြီး branch ကို merge လုပ်ရန် တောင်းဆိုမှု။
README	Repo ၏ မျက်နှာစာ စာတမ်း — ၎င်းသည် ဘာလဲ၊ ဘယ်လို လည်ပတ်ရမလဲ။ Recruiter များ ဤစာများကို ဖတ်ကြသည်။

ဤ module အတွက် ဗီဒီယိုများ:

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
Learn Python — Full Course for Beginners	freeCodeCamp	~4.5 hr (သုံးပတ်တာ အတွင်း ခွဲကြည့်ပါ)	https://www.youtube.com/watch?v=rfscVS0vtbw

အသံထွက်၍ ပြောပါ: “I’ll script it — it’ll be wrong once, then never again.” (Script ရေးလိုက်မယ် — တစ်ခါပဲ မှားမယ်၊ နောက်ဘယ်တော့မှ မမှားတော့ဘူး။) · “Check the exit code before the next step runs.” (နောက်တစ်ဆင့် မလည်ခင် exit code ကို စစ်ပါ။) · “Commit it with a message that says why, not what.” (ဘာလုပ်တယ်မဟုတ်ဘဲ ဘာကြောင့်လုပ်တယ်ဆိုတဲ့ message နဲ့ commit လုပ်ပါ။) · “It’s in the repo — pull the latest.” (Repo ထဲမှာ ရှိတယ် — နောက်ဆုံးဗားရှင်းကို pull လုပ်ပါ။)

လေ့ကျင့်ခန်းများ: (၁) အထက်ပါအတိုင်း `backup.sh` — lab EC2 instance တစ်လုံးပေါ်တွင် cron ဖြင့် ညစဉ် အချိန်ဇယားဆွဲထားပါ။ (၂) အထက်ပါအတိုင်း `audit.py`။ (၃) နှစ်ခုလုံးကို တစ်ခုချင်းရှင်းပြသော README နှင့်အတူ `cloud-journey` ထဲ commit လုပ်ပါ။ (၄) သင့်ကိုယ်ပိုင် script ကို တမင်ဖျက်ဆီးကြည့်ပါ (path မှားတစ်ခု၊ bucket ပျောက်တစ်ခု) ပြီးလျှင် ကျယ်ကျယ်လောင်လောင် ရှင်းရှင်းလင်းလင်း ကျရှုံးအောင် လုပ်ပါ — error ကိုင်တွယ်မှုသည် ops လောက၏ ချစ်ခြင်းမေတ္တာ ဘာသာစကားတစ်ခု ဖြစ်သည်။

Milestone: သင့် backup script သည် နှစ်ကြိမ်ဆက်တိုက် လည်ပတ်ခြင်းနှင့် folder ပျောက်နေခြင်းတို့ကို ခံနိုင်သည် (crash မဖြစ်၊ ရှင်းလင်းသော message)။ သင့် Python audit သည် သင့် account အစစ်ပေါ်တွင် လည်ပတ်သည်။ သင့်

repo တွင် တစ်ပတ်စာ commit များ ပြုနေသည်။

Module 7 (အပတ်စဉ် ၁၆-၁၇): Terraform — infrastructure as code

အဓိက အယူအဆကြီး: resource များကို click နှိပ်ပြီး ဖန်တီးမည့်အစား ဘာတွေရှိသင့်သည်ကို ကြေညာထားသော (declare) စာသားဖိုင်များ ရေးသည် — “VPC တစ်ခု၊ subnet နှစ်ခု၊ instance တစ်လုံး၊ ဤ tag များ” — ပြီးလျှင် Terraform က AWS ကို ထိုဖိုင်နှင့် ကိုက်ညီအောင် လုပ်ပေးသည်။ Imperative မဟုတ်၊ declarative ဖြစ်သည် — လမ်းချိုးအလှည့်များကို မဟုတ်ဘဲ ဦးတည်ရာကိုသာ ဖော်ပြခြင်း။ အလုပ်ခေါ်စာတိုင်း ၎င်းကို တောင်းရသည့်အကြောင်းရင်း — ဖိုင်များသည် Git ထဲ နေထိုင်သောကြောင့် infrastructure သည် ပြန်လည်သုံးသပ်ခံရသည် (ပြောင်းလဲမှုမတိုင်မီ pull request များ)၊ ထပ်ခါလုပ် နိုင်သည် (ဖိုင်တစ်ဖိုင်တည်းက dev၊ staging နှင့် prod ကို တစ်ပုံစံတည်း တည်ဆောက်သည်)၊ ပြန်ပြင်နိုင်သည် (commit ကို revert လုပ်ခြင်းဖြင့် ပြန်ဆုတ်နိုင်သည်)၊ နှင့် စစ်ဆေးနိုင်သည် (မည်သူ ဘာကို ဘယ်တုန်းက ဘာကြောင့် ပြောင်းခဲ့သည်ကို သမိုင်းက ပြောပြသည်)။ Console သည် လက်မှုအလုပ်ရုံ၊ Terraform သည် စက်ရုံ။

ရာနှင့်ချီ လည်ပတ်ရမည့် အဓိကစက်ဝန်း: terraform init (AWS provider — သင့်ဖိုင်များကို API ခေါ်ဆိုမှုများ အဖြစ် ဘာသာပြန်ပေးသော plugin — ကို download လုပ်ခြင်း) → terraform plan (ဘာတွေ ဖန်တီး/ပြောင်း/ဖျက် မည်ကို အတိအကျ ပုံနှိပ်ပြသော အစမ်းလည်ပတ်မှု — စာကြောင်းတိုင်း ဖတ်ပါ။ Plan မဖတ်ဘဲ apply လုပ်သော engineer များသည် outage ဖြစ်စေတတ်သည်) → terraform apply (အစစ်ဖြစ်အောင်လုပ်ခြင်း) → terraform destroy (ပြန်ဖျက်ခြင်း — teardown ကို command တစ်ခုတည်းဖြင့် — ဤအချိန်ရောက်လျှင် ၎င်းကို လှပသည်ဟု သင် ခံစားမိမည်)။ Terraform သည် တည်ဆောက်ခဲ့သမျှကို state file — လက်တွေ့ကမ္ဘာအပေါ် ၎င်း၏ မှတ်ဉာဏ် — ထဲတွင် မှတ်သားထားသည်။ ၎င်းကို ပျောက်စေခြင်း သို့မဟုတ် ၎င်း၏ကျော့နောက်ကွယ်တွင် လက်တွေ့ကမ္ဘာကို လက်ဖြင့်ပြင်ခြင်း (“drift”) လုပ်လျှင် ဒုက္ခ လိုက်လာသည်။ Resource များကို ဖတ်ရလွယ်သော config ဘာသာစကား HCL ဖြင့် ကြေညာသည်

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

Lab (ဤသည်ပင် module ဖြစ်သည်): အဆင့် ၁ ၏ စိန်ခေါ်မှုကြီးကို Terraform ဖြင့် တစ်ဆင့်ချင်း ပြန်တည်ဆောက်ပါ။ (၁) terraform-labs ဟူသော repo တစ်ခု init လုပ်ပါ။ Tag တပ်ထားသော S3 bucket တစ်ခုတည်းအတွက် config ရေးပါ။ Plan လုပ်၊ ဖတ်၊ apply လုပ်၊ console ထဲစစ် — ရှိနေပြီ။ Destroy လုပ်ပါ။ (၂) Config ကို ချဲ့ပါ — VPC၊ subnet နှစ်ခု၊ internet gateway၊ route table — Lab 3 ၏ တည်ဆောက်မှု၊ ယခုအခါ HCL ~၆၀ ကြောင်း။ (၃) Security group နှင့် EC2 instance တစ်လုံးကို variables.tf (region၊ instance အရွယ်အစား) နှင့် outputs.tf (apply ပြီးလျှင် public IP ကို ပုံနှိပ်ပြ) တို့ဖြင့် ထပ်ထည့်ပါ။ (၄) ဖိုင်ထဲရှိ instance type ကို ပြောင်းပြီး ပြန် apply လုပ်ပါ — Terraform က ကွာခြားချက်ကို တွက်ပြီး ထိုတစ်ခုတည်းကိုသာ ပြောင်းသည်ကို ကြည့်ပါ။ ဤ diff-driven အပြုအမူသည်ပင် မှော်ဆန်မှုတစ်ခုလုံး ဖြစ်သည်။ (၅) အားလုံး destroy လုပ်ပါ။ console ဗလာဖြစ်ကြောင်း အတည်ပြုပါ။ commit လုပ်ပြီး repo ကို v1.0 ဟု tag တပ်ပါ။ ကိုးကားစာတမ်းများသည် <https://developer.hashicorp.com/terraform> တွင် ရှိသည် — bookmark လုပ်ထားပါ။ Provider doc များ ဖတ်ခြင်းသည် လက်တွေ့ Terraform အလုပ်၏ တစ်ဝက် ဖြစ်သည်။

ဝေါဟာရများ

ဝေါဟာရ	အဓိပ္ပာယ်
IaC (Infrastructure as Code)	Infrastructure ကို version ထိန်းထားသော စာသားဖိုင်များဖြင့် ကြေညာပြီး ကိရိယာတစ်ခုက လက်တွေ့ဖြစ်အောင် လုပ်ပေးခြင်း။
Terraform / HCL	လွှမ်းမိုးထားသော multi-cloud IaC ကိရိယာ / ၎င်း၏ configuration ဘာသာစကား။
Provider	သင့်ဖိုင်များကို cloud တစ်ခု၏ API ခေါ်ဆိုမှုများအဖြစ် ဘာသာပြန်ပေးသော plugin (ဤနေရာတွင်: AWS)။
plan / apply / destroy	အစမ်း diff / အစစ်ဖြစ်အောင်လုပ်ခြင်း / အားလုံးပြန်ဖျက်ခြင်း။ Plan တိုင်းကို ဖတ်ပါ။
State file	Terraform တည်ဆောက်ခဲ့သမျှ၏ မှတ်တမ်း — လက်တွေ့ကမ္ဘာအပေါ် ၎င်း၏ မှတ်ဉာဏ်။ ကာကွယ်ပါ။
Drift	Terraform ၏ ကျော့နောက်ကွယ်တွင် လက်တွေ့ကမ္ဘာ ပြောင်းသွားခြင်း (တစ်ယောက်ယောက် click နှိပ်ခဲ့သည်)။ ရန်သူ။
Variable / output / module	Config ၏ input တစ်ခု / ပုံနှိပ်ပြသောရလဒ် (IP တစ်ခု၊ URL တစ်ခု) / ပြန်သုံးနိုင်အောင် ထုပ်ပိုးထားသော config အပိုင်း။
Declarative vs imperative	ဦးတည်ရာကို ဖော်ပြခြင်းနှင့် လမ်းချိုးအလှည့်များကို script ရေးခြင်း။ Terraform သည် declarative။
CloudFormation	AWS ၏ ကိုယ်ပိုင် IaC ဝန်ဆောင်မှု — အယူအဆတူ၊ AWS သီးသန့်။ အလုပ်ခေါ်စာများက နှစ်ခုလုံးကို လက်ခံသည်။ Terraform ကို အရင်သင်ပါ။

ဤ module အတွက် ဗီဒီယိုများ

ဗီဒီယို	ချန်နယ်	ကြာချိန်	လင့်ခ်
Terraform explained in 15 mins	TechWorld with Nana	18 min	https://www.youtube.com/watch?v=I5k1ai_GBDE
Terraform Course — Automate your AWS cloud infrastructure	freeCodeCamp	~2.5 hr	https://www.youtube.com/watch?v=SLB_c_ayRMO

အသံထွက်၍ ပြောပါ: “Nothing changes in prod except through Terraform.” (Prod မှာ Terraform ကလွဲရင် ဘာမှ မပြောင်းရဘူး။) · “Show me the plan before you apply.” (Apply မလုပ်ခင် plan ကို ပြပါ။) · “That was clicked in manually — that’s drift; let’s import it or remove it.” (အဲဒါ လက်နဲ့ click နှိပ်ထည့်ထားတာ — drift ပဲ။ Import လုပ် မလား ဖယ်မလား။) · “It’s in the module; reuse it, don’t rewrite it.” (Module ထဲမှာ ရှိပြီးသား — ပြန်သုံးပါ။ ထပ်မရေး ပါနဲ့။)

လေ့ကျင့်ခန်းများ: အထက်ပါ lab အပြင် — (၁) drift ကို တမင်ဖန်တီးပါ — instance ၏ tag ကို console ထဲ လက်ဖြင့် ပြောင်းပါ။ terraform plan ကို လည်ပတ်ပြီး Terraform က သတိထားမိသည်ကို ကြည့်ပါ။ ဘာလုပ်ရန် အဆိုပြုသည်ကို journal ထဲ မှတ်ပါ။ (၂) Repo ၏ README ထဲတွင် “ဘာကြောင့် ဒါက click နှိပ်တာထက် ကောင်းသလဲ” ဟူသော ရိုးရိုး စကားပြေ စာပိုဒ်တစ်ပိုဒ် ရေးပါ။ မရေးနိုင်သေးလျှင် နားမလည်သေးခြင်း ဖြစ်သည်။

Milestone: Directory အလွတ်တစ်ခုမှစ၍ VPC + instance stack ကို init/plan/apply ဖြင့် ဖော်ဆောင်ပြီး destroy ဖြင့် ဖယ်ရှားနိုင်ရမည် — command တစ်ခုချင်းစီ ဘာလုပ်နေသည်ကို မှတ်စုမကြည့်ဘဲ အသံထွက် ရှင်းပြရင်း။

Module 8 (အပတ်စဉ် ၁၈-၂၀): CI/CD, Docker နှင့် Kubernetes ကို ပထမဆုံးအကြိမ် လှမ်း ကြည့်ခြင်း

CI/CD (Continuous Integration / Continuous Delivery) ဆိုသည်မှာ သင့် Git repo နှင့် ချိတ်တွဲထားသော စက်ရုပ် စက်ခါးပတ်လမ်း (conveyor belt) ဖြစ်သည် — push တိုင်းတွင် သင့်ပြောင်းလဲမှုကို အလိုအလျောက် စစ်ဆေး၊ စမ်းသပ်၊ တည်ဆောက်ပြီး — ပြင်ဆင်ထားလျှင် — deploy လုပ်ပေးသည်။ ရည်ရွယ်ချက်မှာ ရှားပါးပြီး ကြောက်စရာကောင်းသော release များအစား သေးငယ်၊ မကြာခဏ၊ ငြိမ့်စရာကောင်းသော release များ ဖြစ်သည်။ **GitHub Actions** သည် GitHub ထဲ ပါပြီးသား CI/CD စနစ်ဖြစ်သည် — **workflow** ဆိုသည်မှာ .github/workflows/ ထဲရှိ YAML ဖိုင်တစ် ဖိုင်ဖြစ်ပြီး “push လုပ်လျှင် **runner** (ယာယီ VM တစ်လုံး) အသစ်ပေါ်တွင် ဤအဆင့်များကို လည်ပတ်ပါ” ဟု ဆိုထားသည်။ Lab: terraform-labs ထဲသို့ push တိုင်းတွင် terraform fmt -check နှင့် terraform validate ကို လည်ပတ်သော workflow တစ်ခု ထည့်ပါ — တမင်ပုံစံမမှန်သော ဖိုင်တစ်ဖိုင် push လုပ်ပြီး အနီရောင် ✖ ကို ကြည့်ပါ။ ပြင်ပြီး အစိမ်းရောင် ✓ ကို ကြည့်ပါ။ ယခုအခါ pipeline တစ်ခု သင့်မှာရှိပြီ — သေးငယ်သော်လည်း အလုပ်ခေါ်စာတိုင်းပါ pipeline များနှင့် မျိုးတူပင်။ (ဒုတိယအဆင့်ကို Project 1 တွင် — pull request များပေါ်တွင် terraform plan ကို လည်ပတ် သော workflow — ထိုအခါ အဆိုပြုထားသော infrastructure ပြောင်းလဲမှုများ၏ diff ကို review ထဲတွင် မြင်ရသည်။)

Docker သည် application တစ်ခုကို ၎င်းလိုအပ်သမျှအားလုံးနှင့်အတူ **container** (ချိပ်ပိတ်ထားသော ထမင်းချိုင့်) တစ်ခု အဖြစ် ထုပ်ပိုးပေးသည် — သင့် laptop၊ EC2 နှင့် အခြားနေရာတိုင်းတွင် တစ်ပုံစံတည်း လည်ပတ်ကာ “works on my machine” (ကျွန်တော့်စက်မှာတော့ ရတယ်) ဟူသော ရှေးရိုးကပ်ဘေးကြီးကို အဆုံးသတ်ပေးသည်။ **Image** ဆိုသည်မှာ အေးခဲ ထားသော ဟင်းချက်နည်း (Dockerfile — build အဆင့်များပါ စာသားဖိုင်တို — မှ တည်ဆောက်သည်: base image တစ်ခု FROM ဖြင့် စ၊ သင့် app ကို COPY ဖြင့် ထည့်၊ start command ကို သတ်မှတ်)၊ **container** ဆိုသည်မှာ image တစ်ခု၏ လည်ပတ်နေသော instance တစ်ခု၊ **registry** (Docker Hub သို့မဟုတ် AWS ၏ ECR) ဆိုသည်မှာ image များကို push နှင့် pull လုပ်ရာနေရာ။ Container များသည် VM များ မဟုတ် — host ၏ Linux kernel ကို မျှဝေသုံးသောကြောင့် တစ် စက္ကန့်ခန့်ဖြင့် စတင်နိုင်ပြီး micro instance တစ်လုံးပေါ်တွင် ဒါဇင်နှင့်ချီ လည်ပတ်နိုင်သည်။ Lab: EC2 instance တစ်လုံး ပေါ်တွင် Docker ကို ထည့်သွင်းပါ။ docker run hello-world၊ ထို့နောက် docker run -d -p 80:8080 <a sample web image> လုပ်ပြီး browser ဖြင့်သွားကြည့်ပါ။ ထို့နောက် သင့် Lab 2 static page ကို nginx base

image မှ ဝန်ဆောင်ပေးသော ငါးကြောင်းတန်း Dockerfile တစ်ခု ရေးပါ။ build လုပ်ပါ။ run လုပ်ပါ။ Docker Hub သို့ push လုပ်ပါ။ နေ့စဉ်သုံး ကြိယများကို သင်ယူပါ: docker ps၊ docker logs၊ docker exec -it <id> bash (container အတွင်း shell တစ်ခု)၊ docker stop။

Kubernetes (K8s) — ဤသင်တန်းအတွက်မူ ရိုးသားစွာ တံဆိပ်ကပ်ထားသည့် ဖတ်ရှုနားလည်ရုံအဆင့် ကျွမ်းကျင်မှု။ ကုမ္ပဏီတစ်ခုက container ရာနှင့်ချီကို စက်များစွာပေါ်တွင် လည်ပတ်သောအခါ တစ်ခုခုက ၎င်းတို့ကို နေရာချ (schedule) ပေးရမည်၊ crash ဖြစ်သည်များကို ပြန်စတင်ပေးရမည်၊ ဝန်ပိသည်များကို ချဲ့ပေးရမည်၊ ၎င်းတို့အကြား traffic လမ်းကြောင်းချပေးရမည် — ထို orchestrator (စနစ်တကျ ညွှန်ကြားစီမံသူ) သည် Kubernetes ဖြစ်သည်။ အယူအဆမြေပုံကို ယခုသင်ယူပါ — **node** များပါသော **cluster** တစ်ခုသည် **pod** များ (deploy လုပ်နိုင်သော အသေးဆုံးယူနစ်၊ များသောအားဖြင့် container တစ်ခု) ကို လည်ပတ်စေသည်။ **Deployment** တစ်ခုက “ဤ pod ၏ replica ၃ ခု အသက်ရှင်နေစေ” ဟု ကြေညာပြီး cluster က ၎င်းကို အဆက်မပြတ် အမှန်ဖြစ်အောင် လုပ်သည် (ထပ်မံ၍ declarative — Kubernetes သည် Terraform ၏ ဒဿနကို လည်ပတ်နေသော software အပေါ် အသုံးချထားခြင်း)။ **Service** တစ်ခုက pod များကို တည်ငြိမ်သောလိပ်စာတစ်ခု ပေးသည်။ AWS ၏ managed ထုတ်ကုန်မှာ **EKS**။ Junior အလုပ်ခေါ်စာများ လိုချင်သည်မှာ ဤအဆင့် literacy နှင့် Docker ကျွမ်းကျင်မှု အတိအကျပင်။ လက်တွေ့ K8s လည်ပတ်နိုင်စွမ်း (နှင့် CKA certification) သည် Year 2 ရည်မှန်းချက်ကောင်းတစ်ခုဖြစ်ပြီး Month 5 ရည်မှန်းချက် မဟုတ်ပါ။ ဤသင်တန်းက မည်သည့်အရာက မည်သည့်အဆင့်ဖြစ်သည်ကို ရိုးသားစွာ ပြောပြသည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
CI/CD	ပြောင်းလဲမှုတိုင်းကို စစ်ဆေး၊ တည်ဆောက်၊ စမ်းသပ်၊ ပို့ဆောင်ပေးသော အလိုအလျောက် pipeline။
GitHub Actions / workflow / runner	GitHub ၏ built-in CI/CD / pipeline တစ်ခုကို သတ်မှတ်သော YAML ဖိုင် / ၎င်းကို လည်ပတ်ပေးသော ယာယီ VM။
YAML	Pipeline များနှင့် Kubernetes တို့၏ indentation အခြေခံ config format။ Indentation သည် အဓိပ္ပာယ် — ဂရုစိုက်ပါ။
Docker / image / container	Container ကိရိယာစုံ / အေးခဲထားသော အလွှာလိုက် ဟင်းချက်နည်း / ၎င်း၏ လည်ပတ်နေသော instance တစ်ခု။
Dockerfile	Image တစ်ခုကို တည်ဆောက်ပေးသော အဆင့်များပါ စာသားဖိုင်တို။
Registry / ECR	Image များကို push နှင့် pull လုပ်ရာနေရာ / AWS ၏ registry။
Kubernetes (K8s) / cluster / node	Container orchestrator / ၎င်း၏ စက်အုပ်စု / ထဲရှိ စက်တစ်လုံး။
Pod / deployment / service	Deploy လုပ်နိုင်သော အသေးဆုံးယူနစ် / “replica N ခု အသက်ရှင်နေစေ” — အဆက်မပြတ် အာမခံထား / pod များရှေ့ရှိ

ဝေါဟာရ	အဓိပ္ပာယ်
	တည်ငြိမ်သောလိပ်စာ။
EKS	AWS ၏ managed Kubernetes control plane။
Rollback	Release တစ်ခု ပြဿနာတက်လျှင် ယခင်ဗားရှင်းသို့ အမြန်ပြန်ပြောင်းခြင်း — CI/CD က ပုံမှန်ဖြစ်အောင် လုပ်ပေးသော ဘေးကင်းရေးကွန်ရက်။

ဤ module အတွက် ဗီဒီယိုများ:

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

အသံထွက်၍ ပြောပါ: “Don’t merge until the pipeline is green.” (Pipeline အစိမ်းမဖြစ်မချင်း merge မလုပ်ပါနဲ့။) · “It’s containerized — same image in dev and prod.” (Container ထဲထည့်ထားတယ် — dev နဲ့ prod မှာ image တစ်ခုတည်း။) · “Exec into the container and check its logs.” (Container ထဲ exec ဝင်ပြီး log တွေ စစ်ပါ။) · “K8s keeps three replicas up; kill one and watch it come back.” (K8s က replica သုံးခု ထိန်းထားတယ် — တစ်ခုသတ်ကြည့်ပြီး ပြန်တက်လာတာ ကြည့်ပါ။)

လေ့ကျင့်ခန်းများ: အထက်ပါ lab နှစ်ခုအပြင် — (၁) “VM၊ container၊ serverless — ဘယ်အချိန် ဘယ်ဟာသုံးမလဲ” ကို စာပိုဒ်တစ်ပိုဒ်ဖြင့် journal ထဲဖြေပါ (Fireship ၏ Serverless in 100 Seconds — https://www.youtube.com/watch?v=W_VV2Fx32_Y — က တတိယရွေးချယ်စရာကို ဖြည့်စွက်ပေးသည်။) (၂) Docker lab instance ကို terminate လုပ်ပါ။ သင့် image များ registry ထဲတွင် ဆက်ရှင်သန်နေကြောင်း အတည်ပြုပါ — artifact က server ထက် အသက်ပိုရှည်သွားပြီဖြစ်ကြောင်း သတိပြုပါ — ၎င်းသည် ခေတ်သစ် deployment လောကအမြင် တစ်ခုလုံးကို တွေ့ရှိမှုတစ်ခုတည်းဖြင့် ချုပ်ထားခြင်း ဖြစ်သည်။

Milestone — အဆင့် ၂ အဆုံး: သင့် GitHub တွင် — အလုပ်ဖြစ်သော Bash + Python ops script repo တစ်ခု၊ stack အစစ်တစ်ခုကို တည်ဆောက်ဖျက်သိမ်းနိုင်သော Terraform repo တစ်ခု၊ အစိမ်းရောင် Actions workflow တစ်ခု၊ registry ထဲ push လုပ်ထားသော Dockerfile တစ်ခု — ပြီးလျှင် ဖိုင်တိုင်းကို အင်တာဗျူးတစ်ခုတွင် ရှင်းပြနိုင်သည်။ ထို GitHub profile သည် ကျောင်းသားတစ်ဦး၏ profile မဟုတ်တော့ပါ။ Junior engineer တစ်ဦး၏ profile ဖြစ်နေပြီ။

အဆင့် ၃ — OPERATIONS (အပတ်စဉ် ၂၁–၂၈)

ဤအဆင့်၏ အယူအဆ: စနစ်များ တည်ဆောက်တတ်ခြင်းက အလုပ်ရစေသည်။ ၎င်းတို့ကို လည်ပတ် တတ်ခြင်းကမူ အလုပ်အစစ် ဖြစ်သည်။ ဤအဆင့်က ဖြေကြားပေးသော အလုပ်ခေါ်စာအချက်များမှာ လည်ပတ်ရေးတစ်ဝက် ဖြစ်သည် — “monitor infrastructure health”၊ “participate in incident response, including log analysis”၊ “identifying, analyzing, and resolving infrastructure vulnerabilities”၊ “manage cloud costs”။ Cloud engineer တစ်ဦး၏ တစ်ပတ်တာသည် အများအားဖြင့် ဤအဆင့်ပင် ဖြစ်သည်။

Module 9 (အပတ်စဉ် ၂၁–၂၄): Monitoring, logging နှင့် incident များ

Monitoring — အသုံးပြုသူများ မသိခင် ကြိုသိခြင်း။ CloudWatch သည် AWS ၏ built-in observability (မြင်နိုင်စွမ်း) ဝန်ဆောင်မှု ဖြစ်သည်။ အခြေခံသုံးခု — **metric** များ (အချိန်နှင့်အမျှ ဂဏန်းများ — CPU %၊ disk %၊ request အရေအတွက်၊ error အရေအတွက်)၊ **alarm** များ (metric တစ်ခုကို စောင့်ကြည့်သော စည်းကမ်း — “CPU > 80% ၅ မိနစ် ကြာလျှင် → အကြောင်းကြား”)၊ နှင့် **dashboard** များ (metric များကို မျက်နှာပြင်တစ်ခုတည်းပေါ် စီစဉ်ထားခြင်း)။ အကြောင်းကြားချက်များသည် **SNS** (Simple Notification Service — publish လုပ်ရသော topic တစ်ခု၊ subscriber များက email/page ရသည်) မှတစ်ဆင့် စီးဆင်းသည်။ အနုပညာက ဘာကို alarm သတ်မှတ်မလဲ ဖြစ်သည် — လူသားလိုအပ်သောအရာအတွက်သာ လူသားကို page လုပ်ပါ။ မဟုတ်လျှင် လူများက pager ကို လျစ်လျူရှုတတ်လာသည် (**alert fatigue** — နာမည်ကြီး outage များစွာ မတိုင်မီ ဖြစ်ပွားခဲ့သော ကျရှုံးမှုပုံစံ)။ အလွတ်ကျက်ထိုက်သော ရွှေအချက်ပြလေးပါး (four golden signals) — **latency, traffic, errors, saturation** — ဘယ်လောက်နွေး၊ ဘယ်လောက်များ၊ ဘယ်လောက်ပျက်၊ ဘယ်လောက်ပြည့်။

Logging — ဘာကြောင့် ဆိုသည်ကို သိခြင်း။ Metric များက တစ်ခုခုမှားနေကြောင်း ပြောသည်။ Log များကမူ ဘာဖြစ်ခဲ့သည်ကို ပြောသည်။ **CloudWatch Logs** က ၎င်းတို့ကို စုစည်းပေးသည် — instance တစ်လုံးစီပေါ်ရှိ **agent** တစ်ခုက `/var/log/nginx/access.log` ကဲ့သို့ ဖိုင်များကို **log group** များဆီ ပို့ပေးပြီး **Logs Insights** ဖြင့် စက်များစွာ အနံ့ query လုပ်နိုင်သည် (“နောက်ဆုံးတစ်နာရီအတွင်း 5xx response များကို မိနစ်အလိုက် ရေတွက်ပါ”)။ စုစည်းခြင်း အရေးကြီးရသည့်အကြောင်းမှာ **server** များသည် ယခုအခါ စွန့်ပစ်နိုင်သောအရာများ ဖြစ်နေပြီ (Module 8 တွင် သင်ကိုယ်တိုင် သက်သေပြခဲ့သည်) — log များသည် ၎င်းတို့ကို ရေးခဲ့သော စက်များထက် အသက်ရှည်ရမည်။

Lab (အပတ်စဉ် ၂၁–၂၂): စောင့်ကြည့်ခံ web server သေးသေးတစ်လုံးကို Terraform ဖြင့် အကုန်တည်ဆောက်ပါ (Module 7 မှ သင့် stack ကို ချဲ့ထွင်၍) — EC2 + nginx + CloudWatch agent။ သင့်ဆီ email ပို့သော SNS topic တစ်ခု၊ CPU မြင့်လျှင် alarm တစ်ခုနှင့် instance status-check ကျရှုံးလျှင် နောက်တစ်ခု။ ထို့နောက် ကိုယ့်ကိုယ်ကို တိုက်ခိုက်ပါ — SSH

ဝင်ပြီး CPU လောင်စာစက် (yes > /dev/null & အကြိမ်အနည်းငယ်) ကို လည်ပတ်ပါ။ Metric တက်လာသည်၊ alarm မြည်သည်၊ email ရောက်လာသည်ကို ကြည့်ပါ။ Process များကို သတ်ပါ။ ပြန်ကောင်းလာသည်ကို ကြည့်ပါ။ ထို့နောက် သင့် ကိုယ်ပိုင် access log များကို Logs Insights ထဲတွင် query လုပ်ပါ။ ယခုအခါ ကိုယ်တိုင်တည်ဆောက်ထားသော စနစ်တစ်ခု ပေါ်တွင် detect → notify → diagnose → resolve စက်ဝန်းအပြည့်ကို မျက်မြင်ကိုယ်တွေ့ ကြုံပြီးပါ။

Incident response — လူသားတစ်စက်။ Incident ဆိုသည်မှာ မမျှော်လင့်ထားသော “စနစ်က အဆင်မပြေဘူး” ဖြစ်ပြီး **severity** (ပြင်းထန်မှု) အဆင့်များ (sev-1 = ဖောက်သည်တွေ ကျသွားပြီ၊ လူတိုင်းဝင်) က တုံ့ပြန်မှုအတိုင်းအတာကို သတ်မှတ်သည်။ ကျွမ်းကျင်သူစက်ဝန်း — **detect** (ဖောက်သည် email မဟုတ်ဘဲ alarm က ဖြစ်စေချင်သည်) → **triage** (ဘယ်လောက်ဆိုးလဲ၊ ဘယ်သူတွေလိုလဲ) → **mitigate** (သွေးထွက်တာကို အရင် တားပါ — roll back၊ restart၊ failover။ **Root cause** က နောက်မှ) → **resolve** → **post-mortem** — ဘာဖြစ်ခဲ့သည်၊ ဘာကြောင့်၊ ထပ်မံဖြစ်အောင် ဘာကတားမည်ကို အပြစ်မတင်ဘဲ (blameless) ရေးသားထားသော ပြန်လည်သုံးသပ်ချက်။ **Blameless** သည် ပျော့ညံ့ခြင်းမဟုတ်၊ **engineering** ဖြစ်သည် — အပြစ်ပေးခံရသောလူများသည် အချက်အလက်ကို ဖုံးကွယ်တတ်ပြီး ဖုံးကွယ်ထားသော အချက်အလက်က ထပ်ခါ outage ဖြစ်စေသည်။ **On-call** ဆိုသည်မှာ **pager** ကိုင်ရသူ အလှည့်ကျစနစ် ဖြစ်သည်။ အလုပ်ခေါ်စာများက ဖော်ပြသည်၊ အင်တာဗျူးသူများက မေးသည်၊ ဤ **module** ပြီးလျှင် သင့်ရိုးသားသောအဖြေမှာ “I’ve simulated it and I know the loop.” (simulate လုပ်ဖူးပြီး စက်ဝန်းကို သိပါတယ်) ဖြစ်သည်။

Runbook များ — documentation အချက်ကို လက်တွေ့ဖြစ်စေခြင်း။ Runbook ဆိုသည်မှာ လည်ပတ်ရေးအခြေအနေ တစ်ခုအတွက် အဆင့်လိုက် ဟင်းချက်နည်းဖြစ်ပြီး နံနက် ၃ နာရီတွင် စိတ်ဖိစီးနေသောလူတစ်ဦး လိုက်လုပ်နိုင်အောင် ရေးထားသည် — လက္ခဏာများ → စစ်ဆေးမှုများ (command အတိအကျ) → ပြင်ဆင်မှုများ (command အတိအကျ) → **escalation** (မရလျှင် ဘယ်သူ့ကို နှိုးမလဲ)။ **Lab (အပတ်စဉ် ၂၃-၂၄)** သင့် repo ထဲတွင် runbook နှစ်ခု ရေးပါ — “web server down” နှင့် “disk filling up” — ထို့နောက် ၎င်းတို့ကို **drill** လုပ်ပါ — ပစ္စည်းကို ဖျက်ပါ။ ကိုယ့်စာတမ်းကို စာသားအတိုင်း လိုက်လုပ်ပါ။ မရှင်းလင်းကြောင်း သက်သေပြသော အဆင့်တိုင်းကို ပြင်ပါ။ ထို့နောက် **game day** အပြည့်တစ်ကြိမ် လုပ်ပါ — မိတ်ဆွေတစ်ဦး (သို့မဟုတ် AI တစ်ခု) ကို သင့် lab stack ကို လျှို့ဝှက်ဖျက်ဆီးခိုင်းပါ။ သင် page ခံရမည်၊ metric များနှင့် log များမှ ရောဂါရှာမည်၊ mitigate လုပ်မည်၊ ပြီးလျှင် post-mortem ကို journal ထဲ ရေးမည်။ ထို post-mortem သည် အင်တာဗျူးဇာတ်လမ်းတစ်ပုဒ်ဖြစ်ပြီး ကောင်းသောတစ်ပုဒ်လည်း ဖြစ်သည်။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
CloudWatch	AWS ၏ monitoring ဝန်ဆောင်မှု — metric များ၊ alarm များ၊ dashboard များ၊ log များ။
Metric / alarm / dashboard	အချိန်နှင့်အမျှ ဂဏန်းတစ်ခု / ၎င်းပေါ်တွင် အလုပ်လုပ်သော စည်းကမ်းတစ်ခု / ၎င်းတို့၏ မျက်နှာပြင်တစ်ခု။
SNS	Alarm များ publish လုပ်ရာ အကြောင်းကြားရေးဝန်ဆောင်မှု — email၊ SMS၊ pager။
Golden signals	Latency၊ traffic၊ errors၊ saturation — မည်သည့် ဝန်ဆောင်မှု၏ ကျန်းမာရေးကိုမဆို ဖော်ပြသော ဂဏန်းလေးလုံး။

ဝေါဟာရ	အဓိပ္ပာယ်
Alert fatigue	အရေးမယူနိုင်သော page များများလာခြင်း → လျစ်လျူရှုခံ page များ → လွတ်သွားသော အစစ်များ။ Alarm ဒီဇိုင်း၏ ရန်သူ။
Log group / Logs Insights	စုစည်းထားသော log များ ရောက်ရှိရာ / ၎င်းတို့အပေါ် query ဘာသာစကား။
Incident / severity	မမျှော်လင့်ထားသော ယိုယွင်းမှုတစ်ခု / ၎င်း၏ အဆင့်သတ်မှတ်ထားသော ဆိုးရွားမှု (sev-1 = အဆိုးဆုံး)။
Triage / mitigate / resolve	အမြန်အကဲဖြတ် / သွေးထွက်တာ အရင်တား / အမှန်တကယ် ပြင်ဆင်။
Post-mortem	အပြစ်မတင်သော ရေးသားထားသည့် ပြန်လည်သုံးသပ်ချက် — ဘာ၊ ဘာကြောင့်၊ ထပ်မဖြစ်အောင် ဘာကတားမည်။
Runbook	အခြေအနေတစ်ခုအတွက် နံနက်-၃-နာရီ-ခံ ဟင်းချက်နည်း — လက္ခဏာ၊ စစ်ဆေးမှု၊ ပြင်ဆင်မှု၊ escalation။
On-call / game day	Pager အလှည့်ကျစနစ် / တမင်လေ့ကျင့်သော incident တစ်ခု။
MTTR	Mean time to recovery — ရင့်ကျက်သောအဖွဲ့များ အကောင်းဆုံးဖြစ်အောင်လုပ်သော ops တိုင်းတာချက်။

အသံထွက်၍ ပြောပါ: “Did we find out from the alarm or from a customer?” (Alarm ကနေ သိခဲ့တာလား၊ ဖောက်သည်ကနေလား။) · “Mitigate first — root cause after we’re stable.” (အရင် mitigate — တည်ငြိမ်မှ root cause။) · “Is there a runbook for this? There will be by tomorrow.” (ဒါအတွက် runbook ရှိလား။ မနက်ဖြန်ဆို ရှိလိမ့်မယ်။) · “What did the post-mortem conclude, and what’s the prevention item?” (Post-mortem က ဘာ ကောက်ချက်ချလဲ၊ ကာကွယ်ရေးအချက်က ဘာလဲ။)

Milestone: သင့် game-day post-mortem သည် တည်ရှိသည်၊ blameless ဖြစ်သည်၊ ပြီးလျှင် သင်အမှန်တကယ် အကောင်အထည်ဖော်ခဲ့သော ခိုင်မာသည့် ကာကွယ်ရေးတစ်ခု (ထပ်ထည့်လိုက်သော alarm တစ်ခု၊ ပြင်လိုက်သော runbook တစ်ခု) ကို ဖော်ပြထားသည်။

Module 10 (အပတ်စဉ် ၂၅–၂၈): လုံခြုံရေးလည်ပတ်မှုနှင့် ကုန်ကျစရိတ်စီမံခန့်ခွဲမှု

Security operations — သူရဲကောင်းဆန်မှုမဟုတ်၊ ကျန်းမာရေးအလေ့အထ။ ပထမဦးစွာ **shared responsibility model** (တာဝန်ခွဲဝေမှုပုံစံ) ကို ပြန်ကြည့်ပါ (<https://www.youtube.com/watch?v=ESPBBEK-cvo>) — AWS က cloud ကိုယ်တိုင်ကို လုံခြုံအောင်လုပ်သည်။ ၎င်းထဲ သင်ထည့်သမျှကိုမူ သင် က လုံခြုံအောင်လုပ်ရသည် — ပြီးလျှင် လက်တွေ့

ပေါက်ကြားမှုအများစုမှာ AWS ၏ချို့ယွင်းမှုများမဟုတ်၊ ဖောက်သည်များ၏ မှားယွင်းသောပြင်ဆင်မှုများ ဖြစ်သည်။ ပြီးငွေသည် အထိ လေ့ကျင့်ရမည့် သင်၏ လည်ပတ်ရေးလုံခြုံရေး checklist —

1. **IAM ကျန်းမာရေး (လစဉ်):** နေရာတိုင်း MFA။ Role ဖြင့်ရနိုင်လျှင် သက်တမ်းရှည် access key မထား။ Credential report ကို ထုတ်ဖတ်ပါ။ မသုံးသော user များနှင့် key များ ဖျက်ပါ။ Policy တိုင်းရှိ * တိုင်းကို မေးခွန်းထုတ်ပါ။ ဤ တို့ပြန်လက်ခံနေအလေ့ကို Lab 5 တွင် တည်ဆောက်ခဲ့ပြီးပြီ — ယခုအခါ ပြက္ခဒိန်ထဲက အချက်တစ်ခု ဖြစ်သွားပြီ။
2. **Patching:** patch မလုပ်ထားသော software သည် ဖောက်ထွင်းမှုအများစု စတင်ရာနေရာ ဖြစ်သည်။ သင့် instance များပေါ်တွင် — update များကို အချိန်ဇယားအလိုက် လုပ်ပါ (ပြီးလျှင် **SSM Patch Manager** က ၎င်းကို fleet တစ် ခုလုံးအတွက် အလိုအလျောက်လုပ်ပေးကြောင်း သိထားပါ)။ Container လောကတွင် patching ဆိုသည်မှာ update လုပ်ထားသော base image မှ image ကို ပြန် build လုပ်ခြင်း ဖြစ်တတ်သည် — ၎င်းကို Module 8 နှင့် ချိတ်ဆက်ပါ။
3. **Backup များ — စမ်းသပ်ပြီးသားများ:** မစမ်းသပ်ရသေးသော backup သည် အစီအစဉ်မဟုတ်၊ မျှော်လင့်ချက်သာ ဖြစ်သည်။ RDS snapshot များ၊ S3 versioning (overwrite တိုင်းတွင် ဗားရှင်းဟောင်း ကျန်သည် — မှားရေးမိမှု နှင့် ransomware တို့ကို တန်ပြန်သော ထိန်းချုပ်မှု)။ Lab: သင့် lab database ကို snapshot ရိုက်ပါ။ instance အသစ်တစ်ခုဆီ restore လုပ်ပါ။ အတန်းများကို စစ်ဆေးပါ။ ပြန်ဖြုတ်ပါ။ ယခုမှသာ အင်တာဗျူးများတွင် “tested” ဟု ပြော ခွင့်ရှိပြီ။
4. **Guardrail များ:** CloudTrail (account ထဲက API ခေါ်ဆိုမှုတိုင်း၏ စစ်ဆေးမှတ်တမ်း — မည်သူ ဘာကို ဘယ်တုန်းက လုပ်ခဲ့သနည်း) ကို ဖွင့်ပြီး မနေ့က ဖြစ်ရပ်များကို တစ်ကြိမ် ဖတ်ကြည့်ပါ။ GuardDuty (အလိုအလျောက် ခြိမ်းခြောက်မှုရှာဖွေရေး) ၏ အခမဲ့စမ်းသုံးချိန်ကို ဖွင့်ပြီး ဘာတွေကို စောင့်ကြည့်သည်ကို ဖတ်ပါ။ Encryption at rest and in transit (သိမ်းထားချိန်နှင့် ပို့ဆောင်ချိန် ကုဒ်ပုဒ်ခြင်း) ဟူသောစကားစုကို သိထားပါ။ AWS တွင် ၎င်းသည် KMS အားပြုထားသော checkbox တစ်ခုသာဖြစ်ကြောင်းလည်း — အခြေခံလိုအပ်ချက်၊ အမြဲဖွင့်။

Cost management — junior များကို senior ပုံပေါက်စေသော ကျွမ်းကျင်မှု။ အလုပ်ခေါ်စာအချက်မှာ မူရင်းအတိုင်း — “manage cloud costs through rightsizing resources, implementing auto-scaling, resource tagging”။ မီတာ လည်ပုံ — compute သည် ဖွင့်ထားသော စက္ကန့်အလိုက် ငွေကောက်သည် (အလကားနေ ≠ အခမဲ့ — “ဖွင့်ထားခဲ့မိတယ်” သည် ဂန္ထဝင်ဖြုန်းတီးမှု)၊ storage သည် GB-လအလိုက်၊ ပြီးလျှင် egress (AWS မှ ထွက်သော ဒေတာ) သည် GB အလိုက် — ဝင် လာသောဒေတာက အခမဲ့ — နာမည်ကြီး ဘေလ်အံ့အားသင့်မှု။ Junior တစ်ဦး ဆွဲနိုင်သောအစီအစဉ်အတိုင်း ကုန်ကျစရိတ်ချုပ် တင်းရေး ကိရိယာတံများ — ပိတ်ပါ (dev စနစ်များကို ညအချိန်၌။ သင့် teardown အလေ့အထကို စက်မှုအဆင့်တင်ခြင်း) → **rightsize** (server အများစုသည် အရွယ်ကြီးလွန်းသည် — CloudWatch CPU သမိုင်းကို စစ်ပြီး ချုံ့ပါ) → **အားလုံးကို tag** တပ်ပါ (Project၊ Owner၊ Environment — tag မတပ်ထားသောသုံးစွဲမှုသည် တာဝန်ခံမဲ့သုံးစွဲမှု။ သင့် Terraform ထဲ တွင် tag များကို မဖြစ်မနေထည့်စေပါ) → **storage ကို tier ခွဲပါ** (Lab 2 မှ lifecycle rule များ) → **reserved capacity/Savings Plans** (၁-၃ နှစ် ကတိပြု၊ ၃၀-၇၀% သက်သာ) နှင့် **Spot** (၉၀% အထိလျှော့၊ ကြားဖြတ်ခံရနိုင်) တို့သည် တည်ငြိမ်သော workload နှင့် batch workload များအတွက် အသီးသီးရှိကြောင်း သိထားပါ။ ကိရိယာများ — **Cost Explorer** (account ၏ သုံးစွဲမှုကို graph ဆွဲပြထားခြင်း — အပတ်စဉ်ဖတ်ပါ။ ထာဝရ) နှင့် **AWS Budgets** (အပတ်စဉ် ၃ မှ သင့် \$0 alert — ယခုအခါ လေးနက်သော မိသားစုတစ်ခု၏ အသေးဆုံးအဖွဲ့ဝင်အဖြစ် နားလည်သွားပြီ)။ ပိုကျယ်သောပညာရပ် ကို **FinOps** ဟုခေါ်သည် — နှစ်မိနစ်တန်သည်: https://www.youtube.com/watch?v=Y-c_xw9bHFw။

ဝေါဟာရများ:

ဝေါဟာရ	အဓိပ္ပာယ်
Shared responsibility model	AWS က cloud ကို လုံခြုံစေသည်။ ထဲထည့်သမျှကို သင်က လုံခြုံစေရသည်။ ပေါက်ကြားမှုအများစုသည် ဒုတိယတစ်ဝက်တွင် ဖြစ်သည်။
CloudTrail	Account ၏ စစ်ဆေးမှတ်တမ်း — API ခေါ်ဆိုမှုတိုင်း၊ မည်သူက၊ ဘယ်တုန်းက။
GuardDuty	Log များနှင့် traffic အပေါ် AWS ၏ အလိုအလျောက် ခြိမ်းခြောက်မှုရှာဖွေရေး။
Patching / SSM	လုံခြုံရေး update များ လုပ်ခြင်း / ၎င်းကို fleet တစ်ခုလုံးအတွက် အလိုအလျောက်လုပ်ပေးသော AWS ၏ Systems Manager။
S3 versioning	Overwrite ခံရသောစားရှင်းတိုင်း သိမ်းထားခြင်း — မှားရေးမိမှု နှင့် ransomware တန်ပြန်ထိန်းချုပ်မှု။
KMS / encryption at rest & in transit	AWS ၏ သော့ဝန်ဆောင်မှု / disk ပေါ်နှင့် ကြိုးပေါ်တွင် ဒေတာကို ကုဒ်ဝှက်ခြင်း။ အမြဲဖွင့်။
Egress	AWS မှ ထွက်သောဒေတာ — GB အလိုက် ငွေကောက်။ ဝင်လာတာ အခမဲ့။ ဂန္ထဝင် ဘေလ်အံ့အားသင့်မှု။
Rightsizing	အရွယ်ကြီးလွန်းသော resource များကို တိုင်းတာထားသော လိုအပ်ချက်အထိ ချုံ့ခြင်း။ အခမဲ့ရသောပိုက်ဆံ။
Tagging	Resource တိုင်းကို owner/project/environment ဖြင့် တံဆိပ်တပ်ခြင်း — သုံးစွဲမှုအားလုံး တာဝန်ခံနိုင်စေရန်။
Savings Plans / Spot	တည်ငြိမ်ဝန်အတွက် ၁-၃ နှစ် ကတိပြုပြီး ၃၀-၇၀% လျှော့ / batch အတွက် ပြန်သိမ်းခံရနိုင်သော ပိုလျှံစွမ်းအား ၉၀% အထိ လျှော့။
Cost Explorer / Budgets	အပတ်စဉ်ဖတ်ရသော သုံးစွဲမှု graph များ / invoice မှ သင်ခန်းစာမယူရအောင် ကြိုသိစေသော alert များ။
FinOps	Cloud သုံးစွဲမှုကို မြင်သာအောင်၊ ခွဲဝေတာဝန်ခံအောင်၊ အဆက်မပြတ် ပိုကောင်းအောင်လုပ်သော ပညာရပ်။

အသံထွက်၍ ပြောပါ: “Who has access to prod, and when did we last review the list?” (Prod ကို ဘယ်သူတွေ ဝင်ခွင့်ရှိလဲ၊ စာရင်းကို နောက်ဆုံး ဘယ်တုန်းက ပြန်စစ်ခဲ့လဲ။) · “When did we last restore a backup?” (Backup ကို နောက်ဆုံး ဘယ်တုန်းက restore လုပ်ခဲ့လဲ။) · “What’s untagged, and what died but is still billing?” (ဘာက tag မ

တပ်ရသေးလဲ၊ ဘာကတော့ သေပြီးလည်း ငွေကောက်နေတုန်းလဲ။) · “It’s over-provisioned — the CPU history says we can halve it.” (အရွယ်ကြီးလွန်းနေတယ် — CPU သမိုင်းအရ တစ်ဝက်ချလို့ရတယ်။)

လေ့ကျင့်ခန်းများ: (၁) အထက်ပါ restore-test lab။ (၂) သင့် repo ထဲတွင် လစဉ်လုံခြုံရေး checklist ဖိုင်တစ်ဖိုင် — ပြီးလျှင် သင့် account အပေါ် အမှန်တကယ် လိုက်စစ်ပါ။ (၃) Cost Explorer ထဲတွင် သင့် account သမိုင်း၏ အမျိုးမျိုးဆုံးအရာကို ရှာပြီး journal စာကြောင်းတစ်ကြောင်းဖြင့် ရှင်းပြပါ။ (၄) သင့် Terraform repo ထဲရှိ resource တိုင်းတွင် default tag များ ထည့်ပါ။

Milestone — အဆင့် ၃ အဆုံး: ကိုယ်ပိုင်စစ်ဆေးမှုအပြည့်တစ်ကြိမ် လုပ်ပါ — လုံခြုံရေး checklist၊ ကုန်ကျစရိတ်သုံးသပ်မှု၊ alarm စမ်းသပ်မှု၊ backup restore — ပြီးလျှင် တစ်မျက်နှာအစီရင်ခံစာ ရေးပါ။ ယခုအခါ သင်သည် တည်ဆောက်ရုံသာမက အလုပ် ကိုပါ လုပ်နိုင်ပြီ။ ကျန်သည်မှာ သူစိမ်းများအား သက်သေပြခြင်းသာ — အဆင့် ၄။

အဆင့် ၄ — အလုပ်အတွက် အသင့်ဖြစ်ရေး (လ ၈-၁၂)

Module 11 (လ ၈-၁၀): Portfolio project သုံးခု

Certification များက သင်လေ့လာခဲ့ကြောင်း ပြောသည်။ Project များကမူ သင်တည်ဆောက်နိုင်ကြောင်း သက်သေပြသည်။ သုံးခု — တစ်ခုစီ ကိုယ်ပိုင် GitHub repo ထဲတွင်၊ တစ်ခုစီတွင် architecture diagram တစ်ခု၊ hiring manager အတွက် ရေးထားသော README (ဘာလဲ၊ ဘာကြောင့်လဲ၊ ဘယ်လိုလည်ပတ်ရမလဲ၊ ဘယ်လောက်ကုန်လဲ)၊ နှင့် teardown script တစ်ခု။ တည်ဆောက် → screenshot/record → ဖျက်ဆီး — အနုပညာလက်ရာသည် repo ဖြစ်သည်၊ လည်နေသောဘေလ် မဟုတ်ပါ။

Project 1 — Three-tier web app၊ အပြည့်အဝ အလိုအလျောက် (ဗဟိုချက်မ)။ Terraform က အားလုံးကို တည်ဆောက်သည် — AZ နှစ်ခုအနွဲ့ public/private subnet များပါ VPC။ Web instance များ၏ **Auto Scaling Group** (ဝန်အလိုက် AWS က instance များ ထည့်/ဖယ်ပေးသည် — ရှာဖွေလေ့လာပြီး ချိတ်ဆက်ပါ) ရှေ့တွင် **Application Load Balancer** (traffic ဖြန့်ဝေကိရိယာ — သင့်အတွက် အသစ်ဖြစ်ပြီး Lab 3 ၏ သဘာဝကျသော နောက်တစ်ဆင့်)။ Private subnet များထဲတွင် RDS။ Static asset များအတွက် S3။ CloudWatch alarm များနှင့် runbook တစ်ခု။ GitHub Actions က pull request တိုင်းတွင် terraform plan နှင့် merge တွင် apply ကို လည်ပတ်သည် — infrastructure pipeline အစစ်တစ်ခု။ ဤ project တစ်ခုတည်းက ချိတ်ဆက်ဇယားထဲရှိ အလုပ်ခေါ်စာအချက် ၁၄ ချက်အနက် ၁၀ ချက်ကို သရုပ်ပြသည်။ အင်တာဗျူးတိုင်း ၎င်းကို လျှောက်မေးမည်ဟု မျှော်လင့်ထားပြီး ငါးမိနစ်အတွင်း ပြောပြနိုင်အောင် ဇာတ်တိုက်ထားပါ။

Project 2 — Serverless data pipeline (နယ်ပယ်ကျယ်ကြောင်းပြခြင်း)။ Server လုံးဝမပါ — S3 bucket တစ်ခုထဲ ဖိုင်တစ်ဖိုင် ရောက်လာလျှင် **Lambda function** (သင့် Python ကို AWS က event အလိုက် လည်ပတ်ပေးပြီး ခေါ်ဆိုမှု အလိုက် ငွေကောက်သည်) တစ်ခုကို trigger လုပ်ပြီး ၎င်းက ဒေတာကို process လုပ်သည် — CSV တစ်ဖိုင်ကို parse လုပ်၊ အကျဉ်းချုပ်၊ ရလဒ်များကို ဒုတိယ bucket သို့မဟုတ် DynamoDB ဇယားတစ်ခုထဲ ရေး — ကျရှုံးမှုများကို ဖမ်း၊ CloudWatch

ထဲ log လုပ်၊ သင့် inbox ဆီ alarm ပို့။ Terraform ဖြင့် deploy လုပ်ပါ။ ၎င်းက Python၊ event-driven တွေးခေါ်မှုနှင့် EC2 ကျော်လွန်သော နယ်ပယ်ကျယ်မှုကို ပြသသည်။ အရင်ဆုံး နှစ်မိနစ် လမ်းညွှန်ချက်: https://www.youtube.com/watch?v=W_VV2Fx32_Y။

Project 3 — Production ပုံစံ ops ပြခန်း (ကွာသွားစေသောအချက်)။ Project 1 ကို ယူပြီး အရေးကြီးသလိုမျိုး လည်ပတ်ပါ — golden signal များ၏ CloudWatch dashboard တစ်ခု။ Paging policy စာတမ်းပါ alarm များ။ Runbook သုံးခု။ သက်သေအထောက်အထားပါ စမ်းသပ်ပြီးသား backup-and-restore လုပ်ထုံး။ ရေးသားတင်ပြထားသော security-hardening တစ်ကြိမ် (IAM audit၊ patching မှတ်စုများ၊ CloudTrail ဖွင့်ထား)။ ကုန်ကျစရိတ်ခွဲခြမ်းစိတ်ဖြာမှုတစ်ခု (“ဤ stack သည် တစ်လ \$X ကုန်သည်။ တစ်ဝက်လျှော့ချပေးမည့် ပြောင်းလဲမှုသုံးခုမှာ ဤသည်တို့”) နှင့် game-day post-mortem တစ်ပုဒ်။ Junior လျှောက်ထားသူ မည်သူမှလိုလို ဤအရာ မရှိကြပါ။ အင်တာဗျူးများ တကယ်မေးနေသော တစ်ခုတည်းသောမေးခွန်း — “ဒီလူကို production နဲ့ ယုံစိတ်ချလို့ရမလား” — ကို နာမဝိသေသနများဖြင့်မဟုတ်ဘဲ စာတမ်းများဖြင့်ဖြေပေးသည်။

Module 12 (လ ၈-၁၂၊ အပြိုင်): Certification အချိန်ဇယား

Certification များက recruiter စစ်ထုတ်မှုများကို ဖွင့်ပေးပြီး သင့်လေ့လာမှုကို ပုံစံချပေးသည်။ ဤရာထူးအတွက် ၂၀၂၆ ခုနှစ် စံလမ်းကြောင်းကို ဤသင်တန်း၏ အရှိန်နှုန်းအလိုက် လက်တွေ့ကျသော ပြင်ဆင်ချိန်များနှင့်အတူ — စာမေးပွဲတစ်ခုစီသည် ၎င်း၏အကြောင်းအရာကို သင်ပေးသော သင်တန်းအဆင့်များပြီးချိန်နှင့် အံဝင်ကွင်းကျဖြစ်ကြောင်း သတိပြုပါ — ထို့ကြောင့် ဤအချိန်များသည် အင်တာနက်ကပြောသည်ထက် တိုနေခြင်း ဖြစ်သည်။ —

အစဉ်	Certification	ဘာကို သက်သေပြသလဲ	သင်ရပ်နေမည့်နေရာမှ ပြင်ဆင်ချိန်	ဘယ်တုန်း ဖြေမလဲ
1	AWS Certified Cloud Practitioner (CLF-C02)	Cloud ဝေါဟာရ၊ billing၊ shared responsibility	၃-၄ ပတ် (လမ်းညွှန်များကလည်း အစပြုသူများအတွက် ထိုအတိုင်းဆိုသည်။ အဆင့် ၀-၁ က အများစုကို ခြုံပြီးပြီ)	~လ ၄

အစဉ်	Certification	ဘာကို သက်သေပြ သလဲ	သင်ရပ်နေမည့်နေရာမှ ပြင်ဆင်ချိန်	ဘယ် တုန်း ဖြစ် လဲ
2	AWS Solutions Architect Associate (SAA-C03)	AWS architecture အစစ်များ ဒီဇိုင်း ဆွဲခြင်း — အလုပ်ခေါ်စာများ အမှန်တကယ် စစ်ထုတ်သော အသိအမှတ်ပြုလက်မှတ်	အာရုံစိုက်ပြင်ဆင်မှု ၆-၈ ပတ် (CCP ပြီးနောက် လုပ်ငန်းနယ်ပယ်စံ ခန့်မှန်းချက်)	လ ၈-၉
3	HashiCorp Terraform Associate (003)	IaC ကျွမ်းကျင်မှု၊ အတည်ပြုပြီး	၂-၃ ပတ် — Module 7 နှင့် Project 1 ပြီးလျှင် အများစုမှာ ပြန်လှန်ရုံသာ (တရားဝင်ပြင်ဆင်မှု: https://developer.hashicorp.com/terraform/tutorials/certification-003)	လ ၁၀- ၁၁

Ops ခေါင်းစဉ်ပါ ရာထူးများကို ရည်မှန်းလျှင် ရွေးချယ်နိုင်သော စတုတ္ထတစ်ခု — **AWS SysOps Administrator / CloudOps Associate** — ၎င်း၏အကြောင်းအရာသည် အဆင့် ၃ အတိုင်းပင်။ CLF-C02 အတွက် freeCodeCamp သင်တန်းအပြည့်အစုံ (<https://www.youtube.com/watch?v=7HKot-brXFE>၊ ဂန္ထဝင် edition <https://www.youtube.com/watch?v=NhDYbskXRgc>) နှင့် practice exam များသည် လူသွားများသော အခမဲ့လမ်းကြောင်း ဖြစ်သည်။ SAA-C03 အတွက် သင်တန်းတစ်ခုကို အချိန်ကိုက် practice test များစွာနှင့် တွဲပါ — စာမေးပွဲသည် scenario အခြေခံဖြစ်ပြီး ခံနိုင်ရည်က အရေးပါသည်။

Module 13 (လ ၁၁-၁၂): Résumé၊ အင်တာဗျူးများနှင့် မေးခွန်းဆယ်ခု

Résumé: တစ်မျက်နှာတည်း။ ကျွမ်းကျင်မှုများဖြင့် အစပြုပါ (AWS၊ Terraform၊ Python/Bash၊ Docker၊ CI/CD၊ CloudWatch — အလုပ်ခေါ်စာ၏ ကိုယ်ပိုင်စကားလုံးများကို ပြန်ဟပ်ပါ။ အလိုအလျောက် စစ်ထုတ်စနစ်များသည် keyword များကို တိုက်ဆိုင်ကြည့်သည်) ပြီးလျှင် project သုံးခုကို *did X with Y achieving Z* ပုံစံ အချက်နှစ်ချက်စီဖြင့် — “Deployed a three-tier web app on AWS with Terraform and GitHub Actions; zero-downtime deploys via ALB + Auto Scaling.”။ Certification များကို ရက်စွဲများနှင့် ဖော်ပြပါ။ GitHub ကို link ချိတ်ပါ — ပြီးလျှင် သူတို့ အမှန်တကယ် ဖွင့်ကြည့်မည်ဟု ယူဆထားပါ။ ကောင်းသောသူများက ဖွင့်ကြည့်ကြသည်။ ပြီးလျှင် သင့် repo များက ယခုအခါ ထိုအလည်ကို အကျိုးရှိစေသည်။

အင်တာဗျူး ပြင်ဆင်မှု: ဇာတ်တိုက်ရမည့် အရသာသုံးမျိုး — trivia (အဆင့်များ၏ ဝေါဟာရဇယားများ)၊ scenario များ (သင်ကိုယ်တိုင် လုပ်ခဲ့ဖူးသော lab များ)၊ နှင့် behavioral (သင့် journal ၏ ဇာတ်လမ်းများ — **STAR** ပုံစံဖြင့်: Situation၊ Task၊

Action၊ Result)။ နှစ်ပတ်လုံး နေ့စဉ် အသံထွက် လေ့ကျင့်ပါ — ဖြစ်နိုင်လျှင် AI interviewer တစ်ဦးနှင့် ရက်ရက်စက်စက်။ အောက်ပါမေးခွန်းဆယ်ခုသည် ဂန္ထဝင်များကို ခြုံငုံသည်။ နမူနာအဖြေများကို တမင်ကျစ်ကျစ်လျစ်လျစ် ရေးထားသည် — တစ်ခုချင်းစီကို သင့်ကိုယ်ပိုင် lab အသေးစိတ်များဖြင့် ချဲ့ပါ။ အကြောင်းမှာ “...and when I built this, what actually happened was...” (ဒါကို ကျွန်တော် တည်ဆောက်တုန်းက တကယ်ဖြစ်ခဲ့တာက...) ဟူသောစာကြောင်းသည် ဖတ်ရုံသာဖတ်ထားသော လျှောက်ထားသူများနှင့် သင့်ကို ခွဲခြားပေးသောစာကြောင်း ဖြစ်သောကြောင့်တည်း။

မေးခွန်းဆယ်ခုနှင့် ခိုင်မာသောအဖြေများ:

1. **“Explain the difference between a public and a private subnet.” (Public subnet နှင့် private subnet ကွာခြားချက်ကို ရှင်းပြပါ။)** Public subnet ၏ route table တွင် internet gateway ဆီ route ရှိသောကြောင့် ၎င်း၏ resource များကို အင်တာနက်မှ လက်လှမ်းမီနိုင်သည်။ Private subnet တွင် ထို route မရှိပါ။ Web tier များက public၊ database များက private သွားကြပြီး admin ဝင်ရောက်မှုသည် bastion သို့မဟုတ် SSM မှတစ်ဆင့် private စက်များဆီ ရောက်သည်။ ကျွန်တော့် lab များတွင် private instance ကို laptop မှ လက်လှမ်းမမီသော်လည်း public tier မှ မီကြောင်း သက်သေပြခဲ့သည် — ပျောက်နေသော route သည်ပင် လုံခြုံရေး ဖြစ်သည်။
2. **“What is IAM, and what is least privilege?” (IAM ဆိုတာ ဘာလဲ၊ least privilege ဆိုတာ ဘာလဲ။)** IAM သည် မည်သည့် identity များက မည်သည့် resource များပေါ်တွင် မည်သည့် action များ လုပ်နိုင်သည်ကို user၊ group၊ role နှင့် JSON policy များမှတစ်ဆင့် ထိန်းချုပ်သည်။ Least privilege ဆိုသည်မှာ အလုပ်ဖြစ်ရုံ အနည်းဆုံးကိုသာ ပေးအပ်ခြင်း — ကျွန်တော့် audit script ၏ user တွင် ဝန်ဆောင်မှုတစ်ခုတည်းအပေါ် read-only ခွင့်သာရှိပြီး ကျွန်တော့် EC2 instance များသည် သိမ်းထားသော key များအစား role များကို သုံးသောကြောင့် ပေါက်ကြားစရာ သက်တမ်းရှည် credential မရှိပါ။
3. **“An EC2 web server stopped responding. Walk me through your troubleshooting.” (EC2 web server တစ်လုံး တုံ့ပြန်မှုရပ်သွားပြီ။ သင့် troubleshooting ကို လျှောက်ပြောပြပါ။)** အရင်ဆုံး အတိုင်းအတာ — server တစ်လုံးလား အားလုံးလား — CloudWatch dashboard နှင့် alarm များကို စစ်သည်။ ထို့နောက် အလွှာများကို အစဉ်လိုက် — instance status check များ။ Security group — 80/443 တကယ်ဖွင့်ထားရဲ့လား။ SSH ဝင် — nginx လည်နေရဲ့လား (systemctl status)၊ disk ပြည့်နေလား (df -h)၊ CPU ထိပ်ရောက်နေလား (top)။ ထို့နောက် log များ (error log ကို tail)။ အရင် mitigate — service ကို restart သို့မဟုတ် instance ကို အစားထိုး — တည်ငြိမ်မှု root cause လုပ်သည်။ ဒါက ကျွန်တော့် runbook များထဲ drill လုပ်ထားတဲ့ အစဉ်ပါပဲ။
4. **“Why Terraform instead of clicking in the console?” (Console ထဲ click နှိပ်မယ့်အစား ဘာကြောင့် Terraform လဲ။)** Console သည် ပြန်သုံးသပ်နိုင်သော မှတ်တမ်း မကျန်စေသလို ဘာကိုမှ ပြန်တည်ဆောက်မပေးနိုင်ပါ။ Terraform သည် declarative ဖြစ်ပြီး version ထိန်းထားသည် — ပြောင်းလဲမှုများသည် pull request မှတစ်ဆင့် သွားသည်၊ တူညီသော config က တူညီသော environment များ တည်ဆောက်သည်၊ မကောင်းသောပြောင်းလဲမှုကို commit revert ဖြင့် ပြန်ဆုတ်နိုင်သည်၊ ပြီးလျှင် plan က မဖြစ်ခင် diff အတိအကျ ပြသည်။ ကျွန်တော့် pipeline က PR တိုင်းတွင် plan ကို လည်ပတ်သောကြောင့် diff သည် review ၏ အစိတ်အပိုင်း ဖြစ်နေသည်။
5. **“What’s the difference between a container and a virtual machine?” (Container နှင့် virtual machine ကွာခြားချက်က ဘာလဲ။)** VM သည် hardware ကို virtualize လုပ်ပြီး OS အပြည့် သယ်ဆောင်ကာ boot တက်ရန် မိနစ်ပိုင်း ကြာသည်။ Container သည် host ၏ kernel ကို မျှဝေသုံးပြီး app နှင့် ၎င်း၏ dependency များကိုသာ ထုပ်ပိုးကာ တစ်စက္ကန့်ခန့်ဖြင့် စတင်သည်။ Container များက environment များအနှံ့ တူညီသောအပြုအမူပေးသည် — image တစ်ခုတည်းက ကျွန်တော့် laptop နှင့် EC2 ပေါ်တွင် လည်သည်။ Isolation နှင့် Linux မဟုတ်သော workload များအတွက် VM များ ဆက်အရေးပါနေဆဲ။

6. **“How would you reduce our AWS bill?”** (ကျွန်တော်တို့ AWS ဘောင်ကို ဘယ်လိုလျှော့မလဲ။) ခက်ခဲမှု အစဉ်လိုက် — မလည်သင့်ဘဲ လည်နေသည်များကို ရှာပါ။ အလကားနေ instance များ၊ ချိတ်မထားသော volume များ၊ snapshot ဟောင်းများ။ Cost Explorer နှင့် tag များက ၎င်းကို မြင်သာစေသည်။ ထို့နောက် CloudWatch သမိုင်းနှင့်ယှဉ်ပြီး rightsize။ ထို့နောက် dev environment များကို အလုပ်ချိန်ပြင်ပတွင် ပိတ်ရန် အချိန်ဇယားဆွဲ။ ထို့နောက် S3 ဒေတာဟောင်းများကို ပိုဈေးသက်သာသော tier များဆီ lifecycle။ ထို့နောက် တည်ငြိမ် workload များကို Savings Plans ဖြင့် ကတိပြုပြီး ကြားဖြတ်ခံနိုင်သော batch ကို Spot ပေါ်တင်။ ပြီးလျှင် egress ကို စောင့်ကြည့်ပါ။ — ဒေတာအထွက်သည် ဂန္ထဝင် အံ့အားသင့်စရာလိုင်း ဖြစ်သည်။
7. **“What happens when you type a URL and press Enter?”** (URL ရိုက်ပြီး Enter နှိပ်လိုက်ရင် ဘာဖြစ် သလဲ။) DNS က နာမည်ကို IP အဖြစ် resolve လုပ်သည်။ Browser က port 443 ဆီ TCP connection ဖွင့်ပြီး TLS ညှိနှိုင်းသည်။ HTTP GET တစ်ခု ပို့သည်။ တစ်ဖက်တွင် load balancer တစ်ခုကို ထိပြီး ၎င်းက public subnet ထဲရှိ ကျန်းမာသော instance တစ်လုံးဆီ ဆက်ပို့သည်။ App က private subnet ထဲရှိ database တစ်ခုကို query လုပ်နိုင်သည်။ Response သည် status code — ကျန်းမာလျှင် 200 — ဖြင့် ပြန်လာပြီး browser က ပြသသည်။ Hop တိုင်းကို အသေးစိတ်ဆင်းနိုင်ပြီး တစ်ခုချင်းစီကို ကိုယ်တိုင် တည်ဆောက်ဖူးပါတယ်။
8. **“A teammate’s change broke production. What happens next?”** (အဖွဲ့ဝင်တစ်ဦးရဲ့ ပြောင်းလဲမှုက production ကို ဖျက်လိုက်ပြီ။ နောက်ဘာဆက်ဖြစ်မလဲ။) အရင် mitigate — pipeline မှတစ်ဆင့် roll back။ ထို အခိုက်တွင် ပြန်လည်ကောင်းမွန်ရေးက ရောဂါရှာဖွေရေးထက် အရေးကြီးသည်။ ထို့နောက် blameless post-mortem — ဘာဖြစ်ခဲ့လဲ၊ စနစ်က ဘာကြောင့် ခွင့်ပြုခဲ့လဲ၊ ဘယ် guardrail ပျောက်နေခဲ့လဲ — test တစ်ခု၊ plan review တစ်ခု၊ alarm တစ်ခု။ Blameless သည် လက်တွေ့အရ အရေးကြီးသည် — အပြစ်ပေးခံရမည်ကို ကြောက်သောလူများသည် အချက်အလက် ဖုံးကွယ်တတ်ပြီး ဖုံးကွယ်ထားသော အချက်အလက်က ထပ်ဖြစ်စေသည်။
9. **“What is the shared responsibility model?”** (Shared responsibility model ဆိုတာ ဘာလဲ။) AWS က cloud ကို လုံခြုံစေသည်။ — data center များ၊ hardware၊ hypervisor။ ဖောက်သည်က ၎င်းထဲရှိအရာများကို လုံခြုံစေရသည်။ — ဒေတာ၊ IAM၊ network စည်းကမ်းများ၊ patching၊ encryption setting များ။ လက်တွေ့ cloud ပေါက်ကြားမှုအများစုသည် ဖောက်သည်ဘက်တွင် ဖြစ်သည်။ — public bucket များ၊ ပေါက်ကြားသော key များကဲ့သို့ မှားယွင်းသောပြင်ဆင်မှုများ။ ထို့ကြောင့် ‘AWS is secure’ နှင့် ‘we are secure on AWS’ သည် မတူညီသော စာကြောင်းနှစ်ကြောင်းဖြစ်ပြီး ဒုတိယတစ်ကြောင်းက ကျွန်တော့်အလုပ် ဖြစ်ပါတယ်။
10. **“Tell me about a problem you debugged.”** (သင် debug လုပ်ခဲ့ဖူးတဲ့ ပြဿနာတစ်ခုအကြောင်း ပြောပြပါ။) (သင့်ဟာ — journal ထဲက၊ STAR ပုံစံဖြင့်။ နမူနာအရိုး) ကျွန်တော့် monitoring game day အတွင်း CPU saturation အတွက် alarm မြည်ခဲ့သည် (S)။ ‘အသုံးပြုသူများ’ မသိခင် အကြောင်းရင်းကို ရှာပြီး ရပ်တန့်ရမည် (T)။ Metric များက spike ချိတ်ဆက်ပြသည်။ Logs Insights က ထိန်းမရသော process များနှင့် ချိတ်ဆက်ပြသည်။ SSH ဝင်ပြီး top ဖြင့် အတည်ပြု၊ ၎င်းတို့ကို သတ်ပြီး dashboard ပေါ်တွင် ပြန်ကောင်းလာသည်ကို ကြည့်ခဲ့သည် (A)။ ၁၁ မိနစ်အတွင်း ဖြေရှင်းပြီး post-mortem မှ process-count alarm တစ်ခု ထွက်လာကာ ၎င်းကို Terraform ထဲ တွင် အကောင်အထည်ဖော်ခဲ့သည် (R)။

အလုပ်ရှာရေး လက်တွေ့နည်းလမ်းများ။ လ ၁၁ မှစ၍ လျှောက်ပါ။ — SAA ပြီးလျှင်၊ “အသင့်ဖြစ်မှ” ကို မစောင့်ပါနှင့်၊ အကြောင်း မှာ အင်တာဗျူးများသည်ပင် လေ့ကျင့်မှုဖြစ်သောကြောင့်တည်း။ ရည်မှန်းရာထူးများ — cloud engineer (junior/associate)၊ cloud support engineer၊ cloud operations engineer၊ junior DevOps engineer၊ AWS support associate။ အင်တာဗျူးပါသော ငြင်းပယ်မှုတိုင်းသည် အခမဲ့သင်ခန်းစာ ဖြစ်သည်။ ဘာမေးသည်ကို journal ထဲ မှတ်ပါ။

ဝေါဟာရများ။

ဝေါဟာရ	အဓိပ္ပာယ်
Portfolio	သင်တည်ဆောက်နိုင်ကြောင်း လူသိရှင်ကြား၊ စာတမ်းပြုထားသော သက်သေ — ဤအတတ်ပညာအတွက် diagram များနှင့် README များပါသော GitHub repo များ။
Application Load Balancer (ALB)	AWS ၏ traffic ဖြန့်ဝေကိရိယာ — request များကို ကျန်းမာသော instance များအနံ့ ခွဲဝေပြီး မကျန်းမာသည်များကို ဖယ်သည်။
Auto Scaling Group	Instance N လုံးကို အသက်ရှင်စေပြီး ဝန်အလိုက် N ကို ချိန်ညှိသည် — self-healing နှင့် elasticity တစ်ပေါင်းတည်း။
Lambda / serverless	AWS က event အလိုက် လည်ပတ်ပေးသော code — ခေါ်ဆိုမှုအလိုက် ငွေကောက် — စီမံရမည့် server လုံးဝမရှိ။
DynamoDB	AWS ၏ serverless NoSQL ဇယား — Lambda နှင့် သဘာဝကျကျ တွဲဖက်သည်။
STAR	Situation ၊ Task ၊ Action ၊ Result — ကောင်းသော အင်တာဗျူးဇာတ်လမ်း၏ ပုံစံ။
ATS	Applicant tracking system — သင့်တစ်မျက်နှာ résumé ကျော်ဖြတ်ရမည့် keyword စစ်ထုတ်စနစ်။
CLF-C02 / SAA-C03 / Terraform Associate 003	သင့်စာမေးပွဲသုံးခု — ဝေါဟာရ၊ architecture ၊ IaC ။
SysOps / CloudOps Associate	ရွေးချယ်နိုင်သော ops အာရုံစိုက် AWS associate — အဆင့် ၃ ကို စာမေးပွဲစစ်ထားခြင်း။

Milestone — သင်တန်းအဆုံး: သူစိမ်းတစ်ဦး နားလည်နိုင်သော **repo** သုံးခု၊ ရက်ချိန်းယူပြီး သို့မဟုတ် အောင်ပြီးသော **certification** သုံးခု၊ အသံထွက်ဇာတ်တိုက်ပြီးသော အဖြေဆယ်ခု၊ လျှောက်လွှာများ တင်ပြီး။ သင်သည် “**cloud** ထဲဝင်ဖို့ မျှော်လင့်နေသူ” မဟုတ်တော့ပါ။ သက်သေအထောက်အထားပါသော၊ အင်တာဗျူးဖြေနေသော **junior cloud engineer** တစ်ဦး ဖြစ်နေပြီ။

တစ်မျက်နှာ သင်ရိုးညွှန်းတမ်း

ဘယ်တုန်း	အာရုံစိုက်ရာ	လက်တွေ့ သက်သေ	ပြင်ပ သက်သေ
အပတ်စဉ် ၁-၂	ကွန်ပျူတာများ အလုပ်လုပ်ပုံ။ Linux၊ terminal၊ permission များ၊ SSH	Script ဖြင့် ဖိုင်ကိုင်တွယ်မှု များ။ ပထမဆုံး shell script	—
အပတ်စဉ် ၃-၄	Networking: IP၊ DNS၊ port များ၊ HTTP၊ firewall များ။ AWS account	Account + MFA + \$0 budget	—
အပတ်စဉ် ၅-၆	Lab 1: EC2	အင်တာနက်ပေါ်ရှိ web server၊ ဖြိုချပြီး	—
အပတ်စဉ် ၇	Lab 2: S3 + CLI	Static site။ CLI ကျွမ်းကျင် မှု	—
အပတ်စဉ် ၈-၉	Lab 3: VPC	Public/private network၊ bastion သက်သေ	—
အပတ်စဉ် ၁၀	Lab 4: RDS	Private database၊ snapshot၊ teardown	—
အပတ်စဉ် ၁၁-၁၂	Lab 5: IAM။ အဆင့်-၁ စိန်ခေါ်မှု ပြန်တည်ဆောက် ခြင်း	Stack အပြည့်ကို အလွတ်၊ ၁၃ နာရီ	CCP ရက်ချိန်းယူ
အပတ်စဉ် ၁၃-၁၅	Bash၊ Python/boto3၊ Git	backup.sh၊ audit.py၊ repo သမိုင်း	CCP စာမေးပွဲ (လ ၄)
အပတ်စဉ် ၁၆-၁၇	Terraform	Stack ကို code အဖြစ်၊ plan/apply/destroy	—
အပတ်စဉ် ၁၈-၂၀	CI/CD၊ Docker၊ K8s literacy	အစိမ်းရောင် pipeline။ Registry ထဲ image	—
အပတ်စဉ် ၂၁-၂၄	CloudWatch၊ log များ၊ incident များ၊ runbook များ	Alarm မြည်ပြီး + game- day post-mortem	—
အပတ်စဉ် ၂၅-၂၈	Security ops။ ကုန်ကျစရိတ် စီမံခန့်ခွဲမှု	ကိုယ်ပိုင်စစ်ဆေးမှု အစီရင်ခံစာ။ Restore စမ်းသပ်မှု	—
လ ၈-၁၀	Portfolio Project 1-3	စာတမ်းပြုထားသော repo သုံး ခု	SAA-C03 (လ ၈-၉)

ဘယ်တုန်း	အာရုံစိုက်ရာ	လက်တွေ့ သက်သေ	ပြင်ပ သက်သေ
လ ၁၀-၁၁	Terraform Associate ပြင်ဆင်မှု	—	Terraform Associate
လ ၁၁-၁၂	Résumé၊ အင်တာဗျူးများ၊ လျှောက်လွှာများ	အဖြေဆယ်ခု၊ အသံထွက်	ပထမဆုံး ကမ်းလှမ်းချက်များ

သင့်ဆရာထံမှ နိဂုံးချုပ်စကား။ ဆယ့်နှစ်လသည် အသက်မွေးဝမ်းကျောင်း ပြောင်းလဲရေးအတွက် တိုပြီး နေ့စဉ်အလေ့အထတစ်ခု အတွက် ရှည်သည်။ ထို့ကြောင့် ရိုးသားသော သဘောတူညီချက်မှာ — ဤသင်တန်းကို ပြီးမြောက်သောလူများသည် အတော်ဆုံး လူများ မဟုတ်ကြပါ။ ပင်ပန်းသောနေ့များတွင်လည်း **command** များကို ရိုက်ခဲ့သောလူများသာ ဖြစ်ကြသည်။ သင်ကြိုရသော **error message** တိုင်းသည် သင်ရိုးက ကျရှုံးနေခြင်းမဟုတ်။ အလုပ်လုပ်နေခြင်း ဖြစ်သည်။ အသေးအမွှားရာချီကို ဖျက်ဖူးပြင်ဖူး သော **engineer** သည်ပင် **hiring manager** က “**experience**” ဟုဆိုရာတွင် တိတိကျကျ ဆိုလိုသောသူ ဖြစ်သည်။ **Journal** ကို ဆက်ရေးပါ။ တည်ဆောက်သမျှ ဖြိုချပါ။ သင့် **repo** များက မထောက်ခံနိုင်သောအရာကို အင်တာဗျူးတွင် ဘယ်တော့မှ မပြော ပါနှင့်။ — ထို့နောက် တစ်နှစ်အကြာ၊ နံနက် ၃ နာရီတွင် **pager** မြည်လာသောအခါ **adrenaline** အောက်တွင် မမျှော်လင့်ထား သောအရာတစ်ခုကို ခံစားရမည်။ — ကျွမ်းကျင်ပိုင်နိုင်မှု။ သွား၊ တည်ဆောက်လိုက်ပါ။

Sources

Job-description research (August 2026): [Arc.dev — AWS Cloud Engineer Job Description](#) · [Wiz — Cloud Engineer Job Description Guide](#) · [DevsData — AWS Cloud Engineer JD Template](#) · [XOPA — Cloud Engineer JD Template 2026](#) · [Betterteam — Cloud Engineer Job Description](#). Certification path and prep times: [StudyTech — AWS Certification Roadmap 2026](#) · [Cloud Evolvers — Cloud Engineer Roadmap 2026](#) · [HashiCorp — Terraform Associate 003 prep](#). Salary context (US median ≈ \$104K, range \$85K–\$140K): Wiz, above. All YouTube links verified via YouTube metadata at time of writing.

B4LCILC သင်တန်းတစ်ခု — “The Cloud Leader Course” ၏ တွဲဖက်ကျမ်း။

သင်ခန်းစာအသစ်များ — ၂၀၂၆ ခုနှစ် ဩဂုတ်လ ထုတ်ဝေမှု

၂၀၂၆ ခုနှစ် ဩဂုတ်လတွင် [b4lcilc.org](#) ၏ *interactive* သင်တန်းသုံးခု — **Cloud ခေါင်းဆောင် (Cloud Leader)**၊ **Cloud Engineer** နှင့် **Cloud Architect** သင်တန်းများ — ထဲသို့ သင်ခန်းစာအသစ်များ ထပ်မံဖြည့်သွင်းခဲ့သည်။ ဤနောက်ဆက်တွဲ သည် ထိုသင်ခန်းစာအသစ်များကို ဤစာအုပ်၏ ကျန်အပိုင်းများနှင့် လေသံတစ်မျိုးတည်းဖြင့် သစ္စာရှိရှိ ချုံ့ရေးသားပေးထားကာ ဖတ်ရှုရန်ထုတ်ဝေမှု (*reading edition*) ကို နောက်ဆုံးအခြေအနေနှင့် ကိုက်ညီအောင် လုပ်ဆောင်ပေးသည်။

သင်လက်ထဲတွင် ကိုင်ထားသည့်အရာအကြောင်း စကားတစ်ခွန်း။ ဤစာမျက်နှာများသည် ဖတ်ရှုရန်ထုတ်ဝေမှု ဖြစ်သည် — အယူအဆများ၊ ဥပမာတင်စားချက်များနှင့် အသံထွက်၍ ပြောထိုက်သော စာကြောင်းများ။ အွန်လိုင်းရှိ **interactive** ဗားရှင်းအပြည့်အစုံများတွင် 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 ဤချို့ရားသားချက်များကို 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 လုံးဝမရှိသလောက် ဖြစ်သော **function** များအဖြစ် ပြောင်းလဲပေးသည့် ပုံစံ။ **Cloud Architect** သင်တန်းအတွက် — ရရှိနိုင်မှု (**availability**) ကတိကဝတ်များ၏ ဂဏန်းသင်္ချာ၊ ယခုအခါ ဖောက်သည်များ အပတ်စဉ်လိုလို မေးမြန်းလာကြသော **Retrieval-Augmented Generation** ပုံစံ၊ နှင့် ကုန်ကျစရိတ် တွက်ချက်ခြင်း လေ့ကျင့်ခန်းအပြည့်အစုံ — မည်သူမျှ မတည်ဆောက်မီ **design** တစ်ခုကို ဈေးနှုန်းတွက်ခြင်း။

Cloud Engineer သင်တန်းအတွက်

Serverless ဆိုတာ တကယ်တမ်း ဘာလဲ

ယခုအထိ သင် **automate** လုပ်ခဲ့သမျှ အားလုံးက စက်တစ်လုံး ရှိသည်ဟု ယူဆထားခဲ့ ဖြစ်သည်။ ဤ **module** ၏ **lab** ထဲက **cron job** သည် တစ်ရက်လျှင် နှစ်မိနစ်သာ **run** သော်လည်း သင်ကမူ တစ်လလုံးစာ ငွေပေးရသော၊ **patch** လုပ်ရသော၊ စောင့်ကြည့်ရသော၊ တစ်နေ့နေ့တွင် နံနက် ၃ နာရီ၌ **debug** လုပ်ရမည့် **virtual machine** တစ်လုံးပေါ်တွင် **run** နေခြင်း ဖြစ်သည်။ **Serverless** သည် ထိုမညီမျှမှုအတွက် **cloud** ၏ အဖြေ ဖြစ်သည်။ **Server** များ ရှိနေဆဲပင် — နာမည်က **marketing** သာ — သို့သော် ၎င်းတို့သည် ဝန်ဆောင်မှုပေးသူ၏ ပြဿနာ ဖြစ်သွားသည်။ သင်က **function** တစ်ခု လွှဲအပ်လိုက်သည်၊ တစ်ခုခုက **trigger** လုပ်သောအခါ ဝန်ဆောင်မှုပေးသူက **run** ပေးသည်၊ သင်ကမူ **invocation** (ခေါ်သုံးမှု) တစ်ကြိမ်လျှင်၊ **millisecond** လိုက် ငွေပေးရသည်။ ဤမော်ဒယ်ကို **FaaS — Function as a Service** ဟုခေါ်သည်။ **AWS** ပေါ်တွင် **Lambda**၊ **Azure** ပေါ်တွင် **Azure Functions**။ သဘောတရားများသည် နာမည်တစ်လုံးချင်း ကူးပြောင်း၍ရသည်။

ခန္ဓာဗေဒက လက်တစ်ဖဝါးထဲ ဆုံသည်။ **Handler** ဆိုသည်မှာ **platform** က ခေါ်ပေးသော **function** ဖြစ်ပြီး ၎င်းက **event** တစ်ခု လက်ခံရရှိသည် — ဖြစ်ပျက်ခဲ့သမျှ၏ **JSON (JavaScript Object Notation)** ဖော်ပြချက်၊ **file** တစ်ခု **bucket** ထဲ ရောက်လာသည်၊ **HTTP request** တစ်ခု ရောက်လာသည်၊ **schedule** (အချိန်ဇယား) တစ်ခု အချိန်ကျ၍ မြည်သည်။ နောက်ဆုံးတစ်ခုတွင် မတ်မတ်ထိုင်လိုက်ပါ။ “**schedule** တစ်ခု မြည်သည်” ဆိုသည်မှာ **cron** လုပ်သည့်အတိုင်း အတိအကျပင် — သို့သော် **schedule** ပေါ်ရှိ **Lambda** တစ်ခုက **run** ချိန်နှစ်ခုကြားတွင် ဖွင့်ထားရမည့် စက်တစ်လုံးမျှ မလိုပါ။ ဤ **module** ၏ လည်ပတ်ရေး **script** တစ်ခုကို ပြောင်းလဲခြင်းသည် ဂန္ထဝင်မြောက် ပထမဆုံး **serverless** ပရောဂျက်တစ်ခု ဖြစ်သည်။

ဤမော်ဒယ် တည်ရှိခြင်း အကြောင်းရင်းမှာ စီးပွားရေးသဘောတရားများပင် ဖြစ်ပြီး ၎င်းတို့သည် ဘက်နှစ်ဖက်လုံး ဓားသွားရှိသည်။ AWS ၏ free tier ထဲတွင် Lambda invocation တစ်လလျှင် တစ်သန်း — ထာဝရ — ပါဝင်သည်။ သင့် lab function သည် တစ်နှစ်လုံး မိနစ်တိုင်း run နေလျှင်ပင် ဘာမျှ မကုန်ကျနိုင် — serverless ကို လည်ပတ်ရေး ကော်ကလေးများ (operations glue) အတွက် ဈေးအသက်သာဆုံး နေအိမ် ဖြစ်စေသည်။ သို့သော် volume နည်းချိန်တွင် ဂဏန်းအကြွင်းလောက်သာရှိသော millisecond-လိုက် ဈေးနှုန်းသည် volume များလာသောအခါ ငွေအစစ် ဖြစ်လာသည်။ အဆက်မပြတ် run နေသော function တစ်ခုသည် container သို့မဟုတ် VM အဖြစ်ထားလျှင် များသောအားဖြင့် ပိုသက်သာသည်။ Workload တစ်ခုသည် ထိုမျှ၏ ဘယ်ဘက်ခြမ်းတွင် ရှိနေသည်ကို သိခြင်းသည် ပရော်ဖက်ရှင်နယ် ကျွမ်းကျင်မှုပင် ဖြစ်သည်။

ရိုးသားသော သတိပေးချက်နှစ်ခု။ ပထမ၊ **cold start**။ မကြာသေးမီက run မထားသော function တစ်ခုသည် နိမ့်ထရန် millisecond ရာဂဏန်းအချို့ကြာသည် — ညစဉ်အစီရင်ခံစာတစ်ခုအတွက် အရေးမပါသော်လည်း latency (နှောင့်နှေးချိန်) အရေးကြီးသော API တစ်ခုအတွက်မူ design အဆင့်မှာပင် ထည့်စဉ်းစားရမည့်အချက် ဖြစ်သည်။ ဒုတိယ၊ function တစ်ခုသည် identity တစ်ခုအနေနှင့် run နေဆဲ ဖြစ်သည်။ EC2 instance တစ်လုံး လုပ်သည့်အတိုင်းပင် Lambda သည် IAM (Identity and Access Management) role တစ်ခုကို ယူဆောင်သည်။ ခွင့်ပြုချက် ပိုလွန်းသော function တစ်ခုသည် အထုပ်အပိုးသေးသွားရုံသာရှိသော လုံခြုံရေးအမှား တစ်မျိုးတည်းပင် ဖြစ်သည်။
