Admissions & Support: support@mockattempt.com
Phone / WhatsApp: +91 95400 07917
Course Code: DOCK • **Updated:** 19 Nov 2026
https://mockattempt.com/courses/linux-docker-github-actions-cicd-cloudformation-cdk-course

DOCK • CLOUD & DEVOPS ENGINEERING **BEGINNER-INTERMEDIATE TO DEVOPS-READY**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation

Stop deploying manually: master Linux server ops, hardened multi-stage Dockerfiles, Docker Compose, Nginx TLS, secretless GitHub Actions OIDC CI/CD, Amazon ECS Fargate, Blue/Green rollouts, and AWS CloudFormation + AWS CDK IaC.

Duration: 7 Weeks • Weekend Live (Sat–Sun • 2h/Day) + IaC Labs (80 Hrs) • **Modules:** 7 • **Live Topics:** 54 • **Subtopics:** 432+ • **Labs & Tasks:** 108 • **Mocks / AI Vivas:** 54 / 54 • **Projects:** 7 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹2,999 (Net Tuition ₹2,542 + Included 18% GST ₹457) • **Split Option:** 100% Credit to Job Track • **Effective ROI:** ~₹37/Guided Hr (~₹56/Live Topic)

Faculty: 100% Active Working IT Professionals • **Weekend-Only Live Batch (7:00 AM – 8:00 PM IST Flexible Slots):** Weekend-Only Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday Only (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Weekend Slot Based on Working Professional Trainer Availability) • 10–12 Weeks Weekend Track)

Core Technologies & Engineering Stack: Linux CLI, systemd & Bash • Docker Engine & BuildKit • Non-Root Multi-Stage Dockerfiles • Docker Compose v2 & Volumes • Nginx Reverse Proxy & HTTPS • GitHub Actions & Secretless OIDC • Amazon ECR & ECS Fargate • Blue/Green & Canary Rollouts • AWS CloudFormation YAML • AWS CDK (TypeScript / Python)

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Administer Linux servers confidently (`systemd`, `journalctl`, permissions, SSH hardening, `ss`/`dig`/`curl`, and defensive Bash scripts).
2. Author minimal, vulnerability-scanned, non-root multi-stage `Dockerfile`s and multi-service `Docker Compose` + `Nginx` stacks.
3. Build automated GitHub Actions test/build/deploy pipelines authenticating to cloud roles via secretless OpenID Connect (`OIDC`) with zero stored keys.
4. Provision and tear down `ECS Fargate`, `ALB`, `S3`, and `VPC` stacks reproducibly using `AWS CloudFormation` and `AWS CDK` with Blue/Green safety.
5. Automated Testing, Debugging & Quality Gates (Linux CLI, systemd & Bash, Docker Engine & BuildKit, Non-Root Multi-Stage Dockerfiles, Docker Compose v2 & Volumes, Nginx Reverse Proxy & HTTPS) — Automated Linux CLI, systemd & Bash, Docker Engine & BuildKit, Non-Root Multi-Stage Dockerfiles, Docker Compose v2 & Volumes, Nginx Reverse Proxy & HTTPS test suites • Linting & static checks • Mentor Git pull-request reviews — ensuring your code never breaks silently when real users hit edge cases.
6. 100% Topic-Wise Timed Mock Tests, Spoken MockAttempt AI Viva & Live Client Production — Verify every single Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation topic through timed MCQ exams and spoken technical interviews before deploying your Live Client Project.
7. Translated live client functional and non-functional requirements into modular Linux CLI, systemd & Bash, Docker Engine & BuildKit, Non-Root Multi-Stage Dockerfiles, Docker Compose v2 & Volumes, Nginx Reverse Proxy & HTTPS production architecture.
8. Implemented core application modules, input validation, security hardening, and optimized data/state workflows.
9. Completed peer Git pull-request code reviews, automated verification suites, and edge-case debugging sprints.
10. Deployed the verified build to live client production with monitoring, documentation, and mentor sign-off.
11. Clear 100% topic-wise Timed MCQ Mock Tests and Spoken MockAttempt AI Technical Interviews across every module of Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation.
12. Deploy 'Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation — Enterprise Client Production Platform' to live HTTPS production and earn both a Barcode Graduation Certificate and Official Internship Letter.

Eligibility & Prerequisites: Basic familiarity with any programming language or web app (Java, Python, JS/TS, or SQL); Linux & Docker start from scratch in Weeks 1–2. • No prior Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend Weekend-Only live classes (Saturday & Sunday, 2 Hours/Day between 7:00 AM – 8:00 PM IST taught by Working IT Professionals), complete weekday real-repository checklists, and practice topic-wise

Mock Tests & AI Interviews. • Dedication to write original code in your GitHub portfolio and complete the mentor-guided Live Client Production Sprint.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (7 MODULES • 54 LIVE TOPICS • 432+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Month 1 – Stage 1: Production Linux, Multi-Stage Dockerfiles, Docker Compose, Nginx & GitHub OIDC CI/CD (Weeks 1–4) (42 Topics / 336 Subtopics) • Modules 5–7: Month 2 – Stage 2: ECS Fargate, Blue/Green Rollouts, AWS CloudFormation & AWS CDK IaC Capstone (Weeks 5–7) (12 Topics / 96 Subtopics)

MODULE 1 OF 7 • STAGE 1 • 11 LIVE + LAB HOURS • 12 TOPICS (96 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Production Linux CLI, Permissions, `systemd`, Networking & Bash Scripting

Master collaborative Git & GitHub pull-request workflows, essential Linux server commands/permissions/processes/SSH/logs, and core TCP/DNS/HTTP/HTTPS/REST/CORS networking — culminating in Month 1 Project 1: TypeScript Employee Management CLI.

Objectives: ✓ Collaborate cleanly using Git branches, rebasing, stashing, conflict resolution, and GitHub Pull Request reviews. • ✓ Navigate Linux filesystems, permissions (`chmod`/`chown`), processes/signals, environment variables, SSH, and log pipelines confidently. • ✓ Explain DNS, TCP/UDP, TLS/HTTPS, raw HTTP request/response headers, status codes, cookies/sessions, REST constraints, and CORS preflights. • ✓ Deliver **Month 1 Project (Project 1): TypeScript-Based Command-Line Employee Management System**.

1.1 Merge Conflict Resolution [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Reading `<<<<<<< HEAD` • `=======` • `>>>>>>> branch` conflict markers • Resolving semantic TypeScript/package-lock conflicts • Production Pitfall Guard (Merge Conflict Resolution) • Hands-On Verification (Merge Conflict Resolution) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Merge Conflict Resolution

Purpose: Resolving concurrent edits to `employee.service.ts` and regenerating `package-lock.json` cleanly via `npm install`.

What Topic Is Next → Next Topic 1.2: .gitignore Hygiene and Branching Strategies

1.2 .gitignore Hygiene and Branching Strategies [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Node/TS `.gitignore` (`node_modules/` • `dist/` • `coverage/` • `.env` • Production Pitfall Guard (.gitignore Hygiene and Branching Strategies) • Hands-On Verification (.gitignore Hygiene and Branching Strategies) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for .gitignore Hygiene

Purpose: Committing a clean `.env.example` template while guaranteeing real `.env` secrets can never be staged.

What Topic Is Next → Next Topic 1.3: Linux Filesystem Hierarchy and Navigation

1.3 Linux Filesystem Hierarchy and Navigation [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: FHS layout (`/etc` config • `/var/log` logs • `/opt` & `/srv` apps • `/tmp` • Production Pitfall Guard (Linux Filesystem Hierarchy and Navigation) • Hands-On Verification (Linux Filesystem Hierarchy and Navigation) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Filesystem Hierarchy

Purpose: Knowing immediately where Nginx configs (`/etc/nginx`), app logs (`/var/log`), and systemd units (`/etc/systemd/system`) live on Ubuntu/Debian.

What Topic Is Next → Next Topic 1.4: Linux File and Directory Operations

1.4 Linux File and Directory Operations [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `mkdir -p` • `cp -r` • `mv` • `rm` • Production Pitfall Guard (Linux File and Directory Operations) • Hands-On Verification (Linux File and Directory Operations) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux File

Purpose: Diagnosing a sudden `ENOSPC: no space left on device` production outage in 10 seconds with `df -h` and `du -sh /var/log/*`!

What Topic Is Next → Next Topic 1.5: Linux Permissions and Ownership (chmod & chown)

1.5 Linux Permissions and Ownership (chmod & chown) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: User/Group/Other bits (`r=4 • W=2 • X=1`) • `chmod 600 ~/.ssh/id_ed25519` • Production Pitfall Guard (Linux Permissions and Ownership (chmod & chown)) • Hands-On Verification (Linux Permissions and Ownership (chmod & chown)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Permissions

Purpose: Locking production `.env` permissions to `0600` owned strictly by the dedicated non-root `nodeapp` Linux user.

What Topic Is Next → Next Topic 1.6: Linux Process Management and Signals

1.6 Linux Process Management and Signals [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `ps aux` • `htop`/`top` • `ss -tulpn` / `lsof -i :3000` (finding what occupies port `3000`!) • `SIGTERM` (`15` • Production Pitfall Guard (Linux Process Management and Signals) • Hands-On Verification (Linux Process Management and Signals) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Process Management

Purpose: Resolving `Error: listen EADDRINUSE: address already in use :::3000` and handling `SIGTERM` cleanly in Node.js.

What Topic Is Next → Next Topic 1.7: Linux Environment Variables and Shell Configuration

1.7 Linux Environment Variables and Shell Configuration [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `export KEY=VAL` • Child-process inheritance • `env`/`printenv` • Systemd `EnvironmentFile=/etc/api/.env` • Production Pitfall Guard (Linux Environment Variables and Shell Configuration) • Hands-On Verification (Linux Environment Variables and Shell Configuration) • Idiomatic DOCK Implementation: multi-

stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Environment Variables

Purpose: Injecting 12-Factor configuration (`DATABASE_URL`, `REDIS_URL`, `PORT`) cleanly without hardcoding secrets in code.

What Topic Is Next → Next Topic 1.8: SSH Fundamentals and Key-Based Remote Access

1.8 SSH Fundamentals and Key-Based Remote Access [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `ssh-keygen -t ed25519` • `~/.ssh/authorized_keys` • `~/.ssh/config` host aliases • `scp`/`rsync` file transfer • Production Pitfall Guard (SSH Fundamentals and Key-Based Remote Access) • Hands-On Verification (SSH Fundamentals and Key-Based Remote Access) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for SSH Fundamentals

Purpose: Connecting securely to staging and production Linux VPS hosts via Ed25519 SSH keys.

What Topic Is Next → Next Topic 1.9: Logs, Piping and Linux Production Troubleshooting

1.9 Logs, Piping and Linux Production Troubleshooting [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `journalctl -u api -f` • `tail -f` • `grep -E` • `jq` JSON log filtering (`cat app.log | jq "select(.level >= 50)"`) • Production Pitfall Guard (Logs, Piping and Linux Production Troubleshooting) • Hands-On Verification (Logs, Piping and Linux Production Troubleshooting) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Logs

Purpose: Filtering structured Pino JSON logs with `jq` to isolate all `5xx` errors for a specific `requestId` in seconds!

What Topic Is Next → Next Topic 1.10: Client-Server Request-Response Architecture

1.10 Client-Server Request-Response Architecture [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Stateless HTTP clients (browsers • Mobile apps • Webhook callers • Microservices) -> Reverse Proxy (Nginx) -> Node.js Application -> Data/Cache Stores • Production Pitfall Guard (Client-Server Request-Response Architecture) • Hands-On Verification (Client-Server Request-Response Architecture) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Client-Server Request-Response Architecture

Purpose: Tracing end-to-end latency budgets across DNS + TCP + TLS + Nginx + Express + PostgreSQL + Response serialization.

What Topic Is Next → Next Topic 1.11: DNS and Domain Resolution

1.11 DNS and Domain Resolution [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Recursive resolver -> Root -> TLD -> Authoritative nameserver • `A` • `AAAA` • `CNAME` • Production Pitfall Guard (DNS and Domain Resolution) • Hands-On Verification (DNS and Domain Resolution) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Domain Resolution

Purpose: Pointing `api.yourdomain.com` (`A`/`AAAA`) to your Nginx reverse proxy and verifying TTL propagation with `dig +short`.

What Topic Is Next → Next Topic 1.12: IP Addresses, Loopback, Subnets and Ports

1.12 IP Addresses, Loopback, Subnets and Ports [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: IPv4 vs IPv6 • `127.0.0.1` (loopback ONLY!) vs `0.0.0.0` (all interfaces — required inside Docker containers!) • Ephemeral ports vs server ports (`80` • `443` • Production Pitfall Guard (IP Addresses, Loopback, Subnets and Ports) • Hands-On Verification (IP Addresses, Loopback, Subnets and Ports) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for IP Addresses

Purpose: Binding PostgreSQL/Redis strictly to `127.0.0.1` or private Docker bridge networks while binding containerized Express to `0.0.0.0:3000` behind Nginx.

What Topic Is Next → Module 1 Capstone (Week 1 Practical Build: Production Linux CLI, Permissions, `systemd`, Networking & Bash Scripting) → Next Topic 2.1: Linux Operating System Fundamentals

Module 1 Hands-On Capstone Project (100 Marks): Week 1 Practical Build: Production Linux CLI, Permissions, `systemd`, Networking & Bash Scripting — Implement and verify a clean, runnable DOCK repository covering Master collaborative Git & GitHub pull-request workflows, essential Linux server commands/permissions/processes/SSH/logs, and...

MODULE 2 OF 7 • STAGE 1 • 11 LIVE + LAB HOURS • 10 TOPICS (80 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Docker Engine Internals, Layer Caching & Non-Root Multi-Stage `Dockerfile`s

Containerize, automate, reverse-proxy, and deploy Python backends cleanly on Linux and cloud infrastructure: Linux OS & CLI (`pwd`/`ls`/`cd`/`mkdir`/`rm`/`cp`), `chmod`/`chown`, `ps`/`top`/`kill`, `curl`/`ping`/`ss`, env vars, SSH, server admin, Docker Engine architecture, images, containers, multi-stage non-root Python `Dockerfile`, volumes, networks, `docker-compose.yml` (App + PostgreSQL + Redis + Celery + Nginx), `.env`/secrets, Nginx reverse proxy, Uvicorn ASGI, Gunicorn + Uvicorn workers, GitHub Actions CI/CD pipelines, AWS (`EC2`, `S3`, `IAM`, `RDS`) fundamentals, and `/healthz` + log monitoring.

Objectives: ✓ Operate Ubuntu Linux servers confidently over SSH using filesystem, permission (`chmod`/`chown`), process (`ps`/`systemctl`), and network (`curl`/`ss`) commands. • ✓ Write lean, secure, non-root multi-stage `python:3.12-slim` `Dockerfile`s with `.dockerignore` and `PYTHONDONTWRITEBYTECODE=1` / `PYTHONUNBUFFERED=1`. • ✓ Orchestrate full-stack `docker-compose.yml` environments (FastAPI/Django + PostgreSQL + Redis + Celery Worker + Nginx) with healthchecks and persistent volumes. • ✓ Run Python in production behind Nginx reverse proxy using Gunicorn (`UvicornWorker` for FastAPI ASGI / WSGI workers for Django) with graceful shutdown. • ✓ Automate Lint -> Typecheck -> Pytest (with Postgres/Redis service containers) -> Docker Build -> Deploy in GitHub Actions and understand AWS EC2/S3/IAM/RDS.

2.1 Linux Operating System Fundamentals [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Kernel vs user space • FHS directory layout (`/etc` • `/var/log` • `/opt` • Production Pitfall Guard (Linux Operating System Fundamentals) • Hands-On Verification (Linux Operating System Fundamentals) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Operating System Fundamentals

Purpose: Navigating Ubuntu LTS servers and Debian-based `python:3.12-slim` Docker containers effortlessly.

What Topic Is Next → Next Topic 2.2: Essential Linux File Commands (pwd, ls, cd, mkdir, rm, cp)

2.2 Essential Linux File Commands (pwd, ls, cd, mkdir, rm, cp) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `pwd` • `ls -lah` • `cd` • `mkdir -p` • Production Pitfall Guard (Essential Linux File Commands (pwd, ls, cd, mkdir, rm, cp)) • Hands-On Verification (Essential Linux File Commands (pwd, ls, cd, mkdir, rm, cp)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Essential Linux File Commands (pwd, ls, cd, mkdir, r...

Purpose: Tailing live Gunicorn/Nginx logs (`tail -f /var/log/nginx/error.log`) and searching error timestamps with `grep`.

What Topic Is Next → Next Topic 2.3: Linux File Ownership and Permissions (chmod and chown)

2.3 Linux File Ownership and Permissions (chmod and chown) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Octal bits (`r=4 • W=2 • X=1`): `chmod 600` (`~/.ssh/id_ed25519` • `.env`) • Production Pitfall Guard (Linux File Ownership and Permissions (chmod and chown)) • Hands-On Verification (Linux File Ownership and Permissions (chmod and chown)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux File Ownership

Purpose: Locking `.env` secrets to `chmod 600` (owner read/write ONLY) and making `entrypoint.sh` executable (`chmod 755`).

What Topic Is Next → Next Topic 2.4: Process Management (ps, top, kill, systemctl)

2.4 Process Management (ps, top, kill, systemctl) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `ps aux` • `top`/`htop` • Graceful `SIGTERM` (`kill -15`) vs last-resort `SIGKILL` (`kill -9`) • Exit codes (`0` vs non-zero) • Production Pitfall Guard (Process Management (ps, top, kill, systemctl)) • Hands-On Verification (Process Management (ps, top, kill, systemctl)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Process Management (ps, top, kill, systemctl)

Purpose: Sending `SIGTERM` during rolling deploys so Gunicorn/Uvicorn finishes in-flight HTTP requests before exiting cleanly!

What Topic Is Next → Next Topic 2.5: Networking Diagnostics (curl, ping, ss/netstat)

2.5 Networking Diagnostics (curl, ping, ss/netstat) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `curl -i -f http://localhost:8000/healthz` • Inspecting headers (`curl -I`) • Listening sockets `ss -tulpn` • `127.0.0.1` loopback vs `0.0.0.0` all interfaces • Production Pitfall Guard (Networking Diagnostics (curl, ping, ss/netstat)) • Hands-On Verification (Networking Diagnostics (curl, ping, ss/netstat)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Networking Diagnostics (curl, ping, ss/netstat)

Purpose: Diagnosing why a Docker container isn't reachable (`uvicorn --host 127.0.0.1` binds ONLY inside container loopback — containers MUST bind `0.0.0.0`!).

What Topic Is Next → Next Topic 2.6: Linux Environment Variables and Shell Profiles

2.6 Linux Environment Variables and Shell Profiles [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `export KEY=VAL` • `env`/`printenv` • Subshell inheritance • `EnvironmentFile=` in `systemd` • Production Pitfall Guard (Linux Environment Variables and Shell Profiles) • Hands-On Verification (Linux Environment Variables and Shell Profiles) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Environment Variables

Purpose: Passing credentials via environment variables/secret mounts rather than `--password=...` CLI flags.

What Topic Is Next → Next Topic 2.7: Secure Remote Server Access with SSH

2.7 Secure Remote Server Access with SSH [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `ssh-keygen -t ed25519` • `~/.ssh/authorized_keys` • Disabling password auth & root login in `/etc/ssh/sshd_config` • `~/.ssh/config` • Production Pitfall Guard (Secure Remote Server Access with SSH) • Hands-On Verification (Secure Remote Server Access with SSH) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Secure Remote Server Access with SSH

Purpose: Connecting to AWS EC2 Ubuntu servers securely using Ed25519 SSH keys with password login disabled.

What Topic Is Next → Next Topic 2.8: Basic Linux Server Administration

2.8 Basic Linux Server Administration [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `ufw` firewall (`allow OpenSSH` • `allow 80 • 443/tcp` • Keep `5432`/`6379` private!) • Production Pitfall Guard (Basic Linux Server Administration) • Hands-On Verification (Basic Linux Server Administration) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Basic Linux Server Administration

Purpose: Ensuring PostgreSQL (`5432`) and Redis (`6379`) are NEVER exposed to `0.0.0.0` on the public internet!

What Topic Is Next → Next Topic 2.9: Containers vs Virtual Machines (Docker Introduction)

2.9 Containers vs Virtual Machines (Docker Introduction) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Shared Linux kernel namespaces (`pid` • `net` • `mnt` • `user`) + `cgroups` resource limits vs full guest-OS hypervisor VMs • Production Pitfall Guard (Containers vs Virtual Machines (Docker Introduction)) • Hands-On Verification (Containers vs Virtual Machines (Docker Introduction)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Containers vs Virtual Machines (Docker Introduction)

Purpose: Shipping the exact Python 3.12.x, `libpq`, and pinned wheel dependencies from laptop -> GitHub Actions -> production.

What Topic Is Next → Next Topic 2.10: Docker Engine Setup

2.10 Docker Engine Setup [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Docker Desktop (WSL 2 backend on Windows) & Docker Engine CE on Ubuntu Linux • Verifying `docker info` • BuildKit • Production Pitfall Guard (Docker Engine Setup) • Hands-On Verification (Docker Engine Setup) • Docker Engine Setup — Interview Readiness • Idiomatic DOCK Implementation: multi-stage

BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Docker Engine Setup

Purpose: Building Linux containers locally on Windows WSL 2 and deploying on Linux servers.

What Topic Is Next → Module 2 Capstone (Week 2 Practical Build: Docker Engine Internals, Layer Caching & Non-Root Multi-Stage `Dockerfile`s) → Next Topic 3.1: Server Logs, logrotate and Diagnostics

Module 2 Hands-On Capstone Project (100 Marks): Week 2 Practical Build: Docker Engine Internals, Layer Caching & Non-Root Multi-Stage `Dockerfile`s — Implement and verify a clean, runnable DOCK repository covering Containerize, automate, reverse-proxy, and deploy Python backends cleanly on Linux and cloud infrastructure: Linux OS & CLI (`...

MODULE 3 OF 7 • STAGE 1 • 11 LIVE + LAB HOURS • 11 TOPICS (88 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Docker Compose v2 Networking, Volumes, Healthchecks & Nginx Reverse Proxy + TLS

Package, harden, automate, and deploy production Node.js backends on Linux servers and containers — Linux sysadmin/SSH/ufw/systemd, multi-stage non-root Dockerfiles, multi-container Docker Compose (Node.js, PostgreSQL, MongoDB, Redis, MinIO), Nginx reverse proxy, PM2, DNS/Let's Encrypt HTTPS, GitHub Actions CI/CD, server monitoring, backups, and live troubleshooting.

Objectives: ✓ Administer and harden Linux backend servers (`users`, `chmod`, `systemd`, `ufw` firewall, SSH keys, `journalctl` logs). • ✓ Author optimized multi-stage non-root `Dockerfile`s and health-checked `Docker Compose` stacks (`Node.js`, `PostgreSQL`, `MongoDB`, `Redis`, `MinIO`, `Nginx`). • ✓ Configure `Nginx` reverse proxying, `PM2` zero-downtime reloads, Domain/DNS records, and automated Let's Encrypt TLS/HTTPS. • ✓ Automate `GitHub Actions` CI/CD pipelines, secrets management, Prometheus/health monitoring, automated backups, and production incident recovery.

3.1 Server Logs, logrotate and Diagnostics [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `journalctl -u backend-api --since "10 min ago"` • Configuring `/etc/logrotate.d/` (`daily` • `rotate 14` • `compress` • Production Pitfall Guard (Server Logs, logrotate and Diagnostics) • Hands-On Verification (Server Logs, logrotate and Diagnostics) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Server Logs

Purpose: Running `dmesg -T | grep -i "killed process"` to prove instantly whether Linux `OOMKiller` killed a crashed Node worker!

What Topic Is Next → Next Topic 3.2: Linux Server Tuning (File Descriptors & TCP Backlog)

3.2 Linux Server Tuning (File Descriptors & TCP Backlog) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Raising `ulimit -n` (`nofile = 65535` • Default `1024` crashes Node with `EMFILE: too many open files` at ~900 concurrent WebSockets!) • `somaxconn` • NTP time sync (critical for JWT `exp` & webhook timestamps!) • Production Pitfall Guard (Linux Server Tuning (File Descriptors & TCP Backlog)) • Hands-On Verification (Linux Server Tuning (File Descriptors & TCP Backlog)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Linux Server Tuning (File Descriptors & TCP Backlog)

Purpose: Supporting `10,000+` concurrent Socket.IO connections on one Linux server by raising `LimitNOFILE=65535` and `chrony` NTP sync.

What Topic Is Next → Next Topic 3.3: Containerization Concepts vs Virtual Machines

3.3 Containerization Concepts vs Virtual Machines [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Linux kernel `namespaces` (isolated PID • Net • Mount • User views) + `cgroups` (CPU/RAM caps) sharing the host kernel (`~20ms` boot • Production Pitfall Guard (Containerization Concepts vs Virtual Machines) • Hands-On Verification (Containerization Concepts vs Virtual Machines) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Containerization Concepts vs Virtual Machines

Purpose: Eliminating `"works on my Windows laptop, breaks on Ubuntu prod"` forever via immutable container artifacts.

What Topic Is Next → Next Topic 3.4: Docker Engine Architecture

3.4 Docker Engine Architecture [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Docker CLI -> `dockerd` Daemon -> `containerd` -> `runc` OCI runtime • Layered copy-on-write UnionFS image layers • OCI registries (`GHCR` / Docker Hub) • Production Pitfall Guard (Docker Engine Architecture) • Hands-On Verification (Docker Engine Architecture) • Docker Engine Architecture — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Docker Engine Architecture

Purpose: Ordering `Dockerfile` instructions from least-frequently changing (`COPY package*.json` -> `RUN npm ci`) to most-frequently changing (`COPY src ./src`) for `10x` faster c...

What Topic Is Next → Next Topic 3.5: Docker Images and Containers Lifecycle

3.5 Docker Images and Containers Lifecycle [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Immutable tagged Images (`ghcr.io/org/api:git-sha` • NEVER deploy mutable `:latest` tags in production!) vs ephemeral Containers (`docker run --rm` • `docker ps` • `docker logs -f` • Production Pitfall Guard (Docker Images and Containers Lifecycle) • Hands-On Verification (Docker Images and Containers Lifecycle) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Docker Images

Purpose: Tagging every CI build with its immutable Git commit SHA (`api:a1b2c3d`) so rolling back to yesterday's release takes 5 seconds!

What Topic Is Next → Next Topic 3.6: Authoring Production Dockerfiles and .dockerignore

3.6 Authoring Production Dockerfiles and .dockerignore [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `FROM node:22-bookworm-slim` (avoid `alpine` `musl` libc surprises with `prisma`/`argon2`/`sharp` native binaries unless tested — `bookworm-slim` uses standard `glibc`!) • `.dockerignore` (`node_modules` • `.git` • `.env*` • Production Pitfall Guard (Authoring Production Dockerfiles and .dockerignore) • Hands-On Verification (Authoring Production Dockerfiles and .dockerignore) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Authoring Production Dockerfiles

Purpose: Using **exec form** `CMD ["node", "dist/server.js"]` instead of `CMD ["npm", "start"]` (`npm` swallows `SIGTERM` and causes a 10-second `SIGKILL` timeout on every deploy!...

What Topic Is Next → Next Topic 3.7: Building and Running Hardened Containers

3.7 Building and Running Hardened Containers [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Running as built-in non-root `USER node` (UID `1000`) • `--read-only` root filesystem (`--tmpfs /tmp`) • `--cap-drop=ALL` • Resource limits (`mem_limit: 512m` • Production Pitfall Guard (Building and Running Hardened Containers) • Hands-On Verification (Building and Running Hardened Containers) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Building

Purpose: Ensuring a compromised dependency cannot write binaries to `/app` or consume all host RAM.

What Topic Is Next → Next Topic 3.8: Docker Volumes and Persistent Database Storage

3.8 Docker Volumes and Persistent Database Storage [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Named volumes (`pg_data:/var/lib/postgresql/data` • `mongo_data:/data/db` • `redis_data:/data` • `minio_data:/data`) vs bind mounts (dev only!) • Production Pitfall Guard (Docker Volumes and Persistent Database Storage) • Hands-On Verification (Docker Volumes and Persistent Database Storage) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Docker Volumes

Purpose: Persisting PostgreSQL, MongoDB, Redis AOF, and MinIO objects safely across container image upgrades.

What Topic Is Next → Next Topic 3.9: Container Networking and Internal DNS Isolation

3.9 Container Networking and Internal DNS Isolation [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Custom isolated Docker networks: `edge_net` (`nginx <-> api`) vs `backend_net` (`api + worker <-> postgres + mongo + redis + minio` with `internal: true` so DB containers have ZERO external exposure!) • Production Pitfall Guard (Container Networking and Internal DNS Isolation) • Hands-On Verification (Container Networking and Internal DNS Isolation) • Container Networking and Internal DNS Isolation — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Container Networking • Production Hardening & Security: non-root USER, read-only rootfs, dropped capabilities & Trivy CVE gates • Diagnostics, Profiling & Telemetry: docker buildx, docker stats, dive layer inspector & Compose logs

Purpose: Placing `postgres`, `mongo`, and `redis` on `backend_net` so `nginx` cannot even route TCP packets directly to databases.

What Topic Is Next → Next Topic 3.10: Docker Compose Orchestration

3.10 Docker Compose Orchestration [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Modern `compose.yaml` (`healthcheck` • `depends_on: { postgres: { condition: service_healthy } }` • `restart: unless-stopped` • `logging: { driver: "json-file" • Production Pitfall Guard (Docker Compose Orchestration) • Hands-On Verification (Docker Compose Orchestration) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Docker Compose Orchestration

Purpose: Waiting for `pg_isready` (`service_healthy`) BEFORE starting the `migrate` one-shot container and `api` service — eliminates boot-time `"Connection refused"` crashes!

What Topic Is Next → Next Topic 3.11: Multi-Container Backend Stack (Node + Postgres + Mongo + Redis + MinIO)

3.11 Multi-Container Backend Stack (Node + Postgres + Mongo + Redis + MinIO) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Assembling the complete polyglot backend stack (`nginx` • `migrate` • `api` • `worker` • Production Pitfall Guard (Multi-Container Backend Stack (Node + Postgres + Mongo + Redis + MinIO)) • Hands-On Verification (Multi-Container Backend Stack (Node + Postgres + Mongo + Redis + MinIO)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Multi-Container Backend Stack (Node + Postgres + Mon...

Purpose: Running the exact Final Capstone production topology locally and on any Linux server.

What Topic Is Next → Module 3 Capstone (Week 3 Practical Build: Docker Compose v2 Networking, Volumes, Healthchecks & Nginx Reverse Proxy + TLS) → Next Topic 4.1: git switch (Switching Branches)

Module 3 Hands-On Capstone Project (100 Marks): Week 3 Practical Build: Docker Compose v2 Networking, Volumes, Healthchecks & Nginx Reverse Proxy + TLS — Implement and verify a clean, runnable DOCK repository covering Package, harden, automate, and deploy production Node.js backends on Linux servers and containers — Linux sysadmin/SSH/ufw/sys...

MODULE 4 OF 7 • STAGE 1 • 11 LIVE + LAB HOURS • 9 TOPICS (72 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Git Workflows, GitHub Actions CI/CD Matrices & Secretless OIDC Federation

Automate lint, test, container build, and secretless OIDC cloud role assumption in GitHub Actions without storing static cloud keys.

Objectives: ✓ Automate lint, test, container build, and secretless OIDC cloud role assumption in GitHub Actions without storing static cloud keys. • ✓ Implement, test, and verify production-grade code and edge-case guardrails across all topics in Week 4.

4.1 git switch (Switching Branches) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Modern `git switch <branch>` • `git switch -c <new-branch>` vs `git restore` (replacing ambiguous overloaded `git checkout`) • Production Pitfall Guard (git switch (Switching Branches)) • Hands-On Verification (git switch (Switching Branches)) • git switch (Switching Branches) — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git switch (Switching Branches) • Production Hardening & Security: non-root USER, read-only rootfs, dropped capabilities & Trivy CVE gates

Purpose: Creating and switching to a hotfix branch cleanly via `git switch -c fix/negative-stock main`.

What Topic Is Next → Next Topic 4.2: git merge (Fast-Forward and 3-Way Merges)

4.2 git merge (Fast-Forward and 3-Way Merges) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Fast-forward pointer advance vs 3-way merge commit (`--no-ff`) vs Squash & Merge on Pull Requests • Production Pitfall Guard (git merge (Fast-Forward and 3-Way Merges)) • Hands-On Verification (git merge (Fast-Forward and 3-Way Merges)) • git merge (Fast-Forward and 3-Way Merges) — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git merge (Fast-Forward and 3-Way Merges) • Production Hardening & Security: non-root USER, read-only rootfs, dropped capabilities & Trivy CVE gates • Diagnostics, Profiling & Telemetry: docker buildx, docker stats, dive layer inspector & Compose logs

Purpose: Integrating reviewed feature branches into `main` while running CI tests on the merged result.

What Topic Is Next → Next Topic 4.3: Merge Conflict Resolution

4.3 Merge Conflict Resolution [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Reading `<<<<<<< HEAD` • `=======` • `>>>>>>> branch` markers • Resolving semantic intent • Production Pitfall Guard (Merge Conflict Resolution) • Hands-On Verification (Merge Conflict Resolution) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Merge Conflict Resolution

Purpose: Resolving concurrent edits to `urls.py`, Alembic heads, or `pyproject.toml` without silently deleting a teammate's code.

What Topic Is Next → Next Topic 4.4: git rebase Fundamentals

4.4 git rebase Fundamentals [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Replaying feature commits linearly atop latest `origin/main` (`git rebase main`) • `--continue`/`--abort` • The Golden Rule of Rebase • Production Pitfall Guard (git rebase Fundamentals) • Hands-On Verification (git rebase Fundamentals) • git rebase Fundamentals — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git rebase Fundamentals

Purpose: Keeping your open feature branch up-to-date with `main` cleanly without messy `"Merge branch main into feat"` commits.

What Topic Is Next → Next Topic 4.5: git stash (Saving Work in Progress)

4.5 git stash (Saving Work in Progress) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `git stash push -u -m "wip: ..."` • `git stash list` • `git stash pop` vs `apply` • `git stash drop` • Production Pitfall Guard (git stash (Saving Work in Progress)) • Hands-On Verification (git stash (Saving Work in Progress)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git stash (Saving Work in Progress)

Purpose: Shelving half-finished feature edits (including untracked `-u` files) to fix an urgent production bug on `main`.

What Topic Is Next → Next Topic 4.6: git reset (Soft, Mixed and Hard)

4.6 git reset (Soft, Mixed and Hard) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `--soft` (moves `HEAD` only) • `--mixed` (unstages too) • `--hard` (DESTROYS working changes!) • `git reflog` recovery • Production Pitfall Guard (git reset (Soft, Mixed and Hard)) • Hands-On Verification (git reset (Soft, Mixed and Hard)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git reset (Soft, Mixed and Hard)

Purpose: Squashing the last 2 local unpushed WIP commits cleanly via `git reset --soft HEAD~2` before pushing.

What Topic Is Next → Next Topic 4.7: git revert (Safe Public Rollbacks)

4.7 git revert (Safe Public Rollbacks) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Creating a new inverse commit via `git revert <sha>` to undo broken commits safely on shared `main` without rewriting history • Production Pitfall Guard (git revert (Safe Public Rollbacks)) • Hands-On Verification (git revert (Safe Public Rollbacks)) • git revert (Safe Public Rollbacks) — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for git revert (Safe Public Rollbacks) • Production Hardening & Security: non-root USER, read-only rootfs, dropped capabilities & Trivy CVE gates • Diagnostics, Profiling & Telemetry: docker buildx, docker stats, dive layer inspector & Compose logs

Purpose: Rolling back a faulty deployment commit on `main` in 10 seconds while keeping team Git history intact.

What Topic Is Next → Next Topic 4.8: Remote Repositories Configuration

4.8 Remote Repositories Configuration [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `git remote -v` • `origin` vs `upstream` • SSH (`git@github.com:...`) vs HTTPS PAT remotes • `git remote set-url` • Production Pitfall Guard (Remote Repositories Configuration) • Hands-On Verification (Remote Repositories Configuration) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Remote Repositories Configuration

Purpose: Linking local repositories to GitHub organization remotes over passwordless Ed25519 SSH keys.

What Topic Is Next → Next Topic 4.9: AWS CI/CD Suite: CodeBuild, CodeDeploy, CodePipeline & GitHub OIDC

4.9 AWS CI/CD Suite: CodeBuild, CodeDeploy, CodePipeline & GitHub OIDC [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Source (`GitHub` via CodeConnections or **GitHub Actions OIDC Federation -> IAM Role** with zero stored AWS keys!) • **AWS CodeBuild** (managed serverless build/test runner controlled by `buildspec.yml` • Place in VPC to run private DB integration tests!) • **AWS CodeDeploy** (deploys via `appspec.yml` to **EC2/On-Premises • Production Pitfall Guard (AWS CI/CD Suite: CodeBuild, CodeDeploy, CodePipeline & GitHub OIDC) • Hands-On Verification (AWS CI/CD Suite: CodeBuild, CodeDeploy, CodePipeline & GitHub OIDC) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for AWS CI/CD Suite

Purpose: Replacing static IAM user keys in GitHub Actions with an **AWS IAM OpenID Connect (`OIDC`) Identity Provider** (`token.actions.githubusercontent.com`) scoped strictly to...

What Topic Is Next → Module 4 Capstone (Week 4 Practical Build: Git Workflows, GitHub Actions CI/CD Matrices & Secretless OIDC Federation) → Next Topic 5.1: Containerization, Dockerfiles & Amazon ECR

Module 4 Hands-On Capstone Project (100 Marks): Week 4 Practical Build: Git Workflows, GitHub Actions CI/CD Matrices & Secretless OIDC Federation — Implement and verify a clean, runnable DOCK repository covering Automate lint, test, container build, and secretless OIDC cloud role assumption in GitHub Actions without storing static cloud...

MODULE 5 OF 7 • STAGE 2 • 11 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Private Amazon ECR, Serverless `ECS Fargate` & Blue/Green / Canary Rollouts

Deploy scanned container images onto serverless ECS Fargate (`Task Execution` vs `Task Role`) with CodeDeploy Canary & CloudWatch Alarm auto-rollback.

Objectives: ✓ Deploy scanned container images onto serverless ECS Fargate (`Task Execution` vs `Task Role`) with CodeDeploy Canary & CloudWatch Alarm auto-rollback. • ✓ Implement, test, and verify production-grade code and edge-case guardrails across all topics in Week 5.

5.1 Containerization, Dockerfiles & Amazon ECR [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Containers vs VMs • Multi-stage non-root `Dockerfile` • **Amazon Elastic Container Registry (`ECR`)** (Private vs Public gallery • IAM/KMS integration • Production Pitfall Guard (Containerization, Dockerfiles & Amazon ECR) • Hands-On Verification (Containerization, Dockerfiles & Amazon ECR) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Containerization

Purpose: Scanning every pushed container image automatically for OS & package CVEs in **Amazon ECR** and keeping only the latest 15 tagged production images via an **ECR Lifecycle...

What Topic Is Next → Next Topic 5.2: Amazon ECS Architecture: Clusters, Task Definitions, Roles & AWS Fargate

5.2 Amazon ECS Architecture: Clusters, Task Definitions, Roles & AWS Fargate [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: **ECS Task Definition** (JSON blueprint: image • CPU/RAM • `awsvpc` ENI network mode) • **ECS Task Execution Role** (used by the ECS/Fargate *agent* to pull image from `ECR` • Production Pitfall Guard (Amazon ECS Architecture: Clusters, Task Definitions, Roles & AWS Fargate) • Hands-On Verification (Amazon ECS Architecture: Clusters, Task Definitions, Roles & AWS Fargate) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Amazon ECS Architecture

Purpose: Running long-lived (`> 15 min`) containerized APIs across Multi-AZ with zero OS patching using **ECS on AWS Fargate** behind an **ALB (`IP` target type!)** with shared fi...

What Topic Is Next → Next Topic 5.3: Amazon EKS Overview, AWS App Runner & AWS Batch Selection Matrix

5.3 Amazon EKS Overview, AWS App Runner & AWS Batch Selection Matrix [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: **Amazon EKS** (managed Kubernetes control plane across AZs — choose when migrating existing on-premises or multi-cloud **Kubernetes `kubectl`/Helm** manifests onto EC2 or Fargate pods!) • **AWS App Runner** (fully managed PaaS: point at GitHub repo or ECR image -> auto-builds • Load-balances • TLS-terminates & auto-scales HTTPS web apps with **zero VPC/ALB/ECS setup**!) • Production Pitfall Guard (Amazon EKS Overview, AWS App Runner & AWS Batch Selection Matrix) • Hands-On Verification (Amazon EKS Overview, AWS App Runner & AWS Batch Selection Matrix) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Amazon EKS Overview

Purpose: Choosing **AWS App Runner** when a small team wants *"lowest operational overhead to deploy a containerized HTTPS web API with auto-scaling"* vs **AWS Batch on Spot** for...

What Topic Is Next → Next Topic 5.4: Deployment Strategies (`Rolling`, `Immutable`, `Blue/Green`, `Canary`) & Auto-Rollback

5.4 Deployment Strategies (`Rolling`, `Immutable`, `Blue/Green`, `Canary`) & Auto-Rollback [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Speed vs Cost vs Risk trade-offs: **All-at-Once** (downtime • Dev only) • **Rolling** (updates batch-by-batch • Temporarily reduced capacity • Production Pitfall Guard (Deployment Strategies (`Rolling`, `Immutable`, `Blue/Green`, `Canary`) & Auto-Rollback) • Hands-On Verification (Deployment Strategies (`Rolling`, `Immutable`, `Blue/Green`, `Canary`) & Auto-Rollback) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Deployment Strategies (`Rolling`, `Immutable`, `Blue...

Purpose: Deploying a risky checkout Lambda/ECS release safely using **CodeDeploy `Canary10Percent10Minutes` + CloudWatch Error Alarm Auto-Rollback** (or **RDS Blue/Green Deploymen...

What Topic Is Next → Module 5 Capstone (Week 11 Lab Build: Scanned ECR Repository & Serverless ECS Fargate Service Behind ALB) → Next Topic 6.1: CloudFormation Template Anatomy & Intrinsic Functions

Module 5 Hands-On Capstone Project (100 Marks): Week 11 Lab Build: Scanned ECR Repository & Serverless ECS Fargate Service Behind ALB — Build a non-root multi-stage Docker container, push to private Amazon ECR with `scanOnPush` + image lifecycle expiry, deploy a 2-task Serverless **ECS Fargate** service (`TargetType=ip`, dis...

MODULE 6 OF 7 • STAGE 2 • 11 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Declarative Infrastructure as Code with AWS CloudFormation (`Change Sets` & `Drift`)

Codify repeatable cloud stacks using CloudFormation YAML Parameters, Dynamic Secrets Manager References, `!Sub`/`!GetAtt`, `DeletionPolicy`, Change Sets, and Drift Detection.

Objectives: ✓ Codify repeatable cloud stacks using CloudFormation YAML Parameters, Dynamic Secrets Manager References, `!Sub`/`!GetAtt`, `DeletionPolicy`, Change Sets, and Drift Detection. • ✓ Implement, test, and verify production-grade code and edge-case guardrails across all topics in Week 6.

6.1 CloudFormation Template Anatomy & Intrinsic Functions [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: YAML sections (`AWSTemplateFormatVersion: "2010-09-09"` • `Parameters` with `NoEcho: true` or dynamic SSM/SecretsManager references! • `Mappings` Region->AMI lookup • `Conditions` `IsProd` • Production Pitfall Guard (CloudFormation Template Anatomy & Intrinsic Functions) • Hands-On Verification (CloudFormation Template Anatomy & Intrinsic Functions) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for CloudFormation Template Anatomy

Purpose: Referencing database credentials inside CloudFormation templates without ever putting plaintext secrets in Git or state using **`{{resolve:secretsmanager:prod/rds:SecretS...

What Topic Is Next → Next Topic 6.2: Change Sets, Drift Detection, Rollbacks, DeletionPolicy & StackSets

6.2 Change Sets, Drift Detection, Rollbacks, DeletionPolicy & StackSets [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: **Change Sets** (dry-run preview showing whether updating a property does an `In-Place Modify` vs destructive **`Replacement: True`**!) • **Drift Detection** (identifies out-of-band manual Console changes on deployed stacks!) • `CreationPolicy` + `cfn-signal` (waits for EC2 User Data app boot before marking stack `CREATE_COMPLETE`!) • **Stack Policy** (prevents accidental updates/replacements on production RDS during stack updates) • Production Pitfall Guard (Change Sets, Drift Detection, Rollbacks, DeletionPolicy & StackSets) • Hands-On Verification (Change Sets, Drift Detection, Rollbacks, DeletionPolicy & StackSets) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Change Sets

Purpose: Rolling out a mandatory IAM Security Audit Role and CloudTrail configuration automatically to **every existing and newly created member account across 5 Regions** using *...

What Topic Is Next → Next Topic 6.3: CloudFormation Intrinsic Functions (`!Ref`, `!GetAtt`, `!Sub`, `!FindInMap`, `!ImportValue`)

6.3 CloudFormation Intrinsic Functions (`!Ref`, `!GetAtt`, `!Sub`, `!FindInMap`, `!ImportValue`) [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Wiring VPC -> Subnet -> SecurityGroup -> ALB -> ECS resources declaratively using `!Ref` physical IDs • `!GetAtt` resource attributes • `!Sub` ARN interpolation • Region `Mappings` • Production Pitfall Guard (CloudFormation Intrinsic Functions (`!Ref`, `!GetAtt`, `!Sub`, `!FindInMap`, `!ImportValue`)) • Hands-On Verification (CloudFormation Intrinsic Functions (`!Ref`, `!GetAtt`, `!Sub`, `!FindInMap`, `!ImportValue`)) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for CloudFormation Intrinsic Functions (`!Ref`, `!GetAtt...

Purpose: Sharing foundational VPC & Subnet IDs across decoupled application stacks cleanly without hardcoding environment ARNs.

What Topic Is Next → Next Topic 6.4: CloudFormation `DeletionPolicy`, `UpdateReplacePolicy`, Rollbacks & `cfn-lint` CI Checks

6.4 CloudFormation `DeletionPolicy`, `UpdateReplacePolicy`, Rollbacks & `cfn-lint` CI Checks [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Protecting stateful RDS/S3 resources with `DeletionPolicy: Snapshot` / `Retain` • Understanding automatic stack rollback on failure • Diagnosing `CREATE_FAILED` events • Running `cfn-lint` + `cfn-nag` security checks in GitHub Actions PRs • Production Pitfall Guard (CloudFormation `DeletionPolicy`, `UpdateReplacePolicy`, Rollbacks & `cfn-lint` CI Checks) • Hands-On Verification (CloudFormation `DeletionPolicy`, `UpdateReplacePolicy`, Rollbacks & `cfn-lint` CI Checks) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for CloudFormation `DeletionPolicy`

Purpose: Catching invalid IAM actions, missing encryption flags, and circular resource dependencies in <5 seconds inside GitHub Pull Requests before touching AWS.

What Topic Is Next → Module 6 Capstone (Week 17 Lab Build: Parameterized Multi-AZ CloudFormation Stack, Change Sets & Drift Detection) → Next Topic 7.1: AWS CDK Architecture: App, Stacks & L1 / L2 / L3 Constructs

Module 6 Hands-On Capstone Project (100 Marks): Week 17 Lab Build: Parameterized Multi-AZ CloudFormation Stack, Change Sets & Drift Detection — Author modular CloudFormation YAML templates using `Parameters` (`ssm`/`secretsmanager` dynamic refs), `Mappings`, `Conditions`, `!GetAtt`/`!Sub`, `DeletionPolicy: Snapshot`, preview updates...

MODULE 7 OF 7 • STAGE 2 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Programmatic Cloud Automation with AWS CDK (`TypeScript`/`Python`), CLI & DevOps Capstone

Synthesize type-safe Multi-AZ VPC, S3, and ECS Fargate stacks in AWS CDK v2 with least-privilege `grant*` policies and clean zero-bill teardown.

Objectives: ✓ Synthesize type-safe Multi-AZ VPC, S3, and ECS Fargate stacks in AWS CDK v2 with least-privilege `grant*` policies and clean zero-bill teardown. • ✓ Implement, test, and verify production-grade code and edge-case guardrails across all topics in Week 7.

7.1 AWS CDK Architecture: App, Stacks & L1 / L2 / L3 Constructs [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: `App -> Stack -> Construct` tree compiling via **`cdk synth`** into standard **AWS CloudFormation templates** • `cdk bootstrap` (one-time per Account/Region S3 + ECR + IAM deployment roles) • **L1 Constructs** (`CfnBucket` raw 1:1 CloudFormation) • **L2 Constructs** (`s3.Bucket` • Production Pitfall Guard (AWS CDK Architecture: App, Stacks & L1 / L2 / L3 Constructs) • Hands-On Verification (AWS CDK Architecture: App, Stacks & L1 / L2 / L3 Constructs) • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for AWS CDK Architecture

Purpose: Defining organization-standard encrypted VPC + S3 + Lambda architectures in **TypeScript AWS CDK** with compile-time type checking and automated `aws-cdk-lib/assertions`...

What Topic Is Next → Next Topic 7.2: AWS CLI Mastery, Python Boto3 SDK Automation & Secret-Safe Config

7.2 AWS CLI Mastery, Python Boto3 SDK Automation & Secret-Safe Config [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: AWS CLI v2 (`aws configure sso` with IAM Identity Center • `--profile` • `--dry-run` • JMESPath `--query` filtering • Production Pitfall Guard (AWS CLI Mastery, Python Boto3 SDK Automation & Secret-Safe Config) • Hands-On Verification (AWS CLI Mastery, Python Boto3 SDK Automation & Secret-Safe Config) •

Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for AWS CLI Mastery

Purpose: Running a scheduled Python `Boto3` Lambda every evening at `20:00 IST` that finds all EC2/RDS resources tagged `Environment=Lab` and stops them automatically — saving 65%...

What Topic Is Next → Next Topic 7.3: AWS CDK L3 `ApplicationLoadBalancedFargateService` & Least-Privilege `grant*` Patterns

7.3 AWS CDK L3 `ApplicationLoadBalancedFargateService` & Least-Privilege `grant*` Patterns [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Deploying a complete Multi-AZ VPC + ALB + ECS Fargate Cluster + CloudWatch Log Group in ~25 lines of TypeScript AWS CDK (`aws-ecs-patterns`) • Granting scoped S3/DynamoDB/Secrets access via `bucket.grantRead(taskRole)` • Writing `aws-cdk-lib/assertions` unit tests • Production Pitfall Guard (AWS CDK L3 `ApplicationLoadBalancedFargateService` & Least-Privilege `grant*` Patterns) • Hands-On Verification (AWS CDK L3 `ApplicationLoadBalancedFargateService` & Least-Privilege `grant*` Patterns) • AWS CDK L3 `ApplicationLoadBalancedFargateService` & Least-Privilege `grant*` Patterns — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for AWS CDK L3 `ApplicationLoadBalancedFargateService`

Purpose: Replacing 700 lines of raw YAML boilerplate with type-checked, unit-tested TypeScript CDK constructs.

What Topic Is Next → Next Topic 7.4: Post-Deployment Smoke Checks, CloudWatch Alarms & Zero-Bill Teardown Automation

7.4 Post-Deployment Smoke Checks, CloudWatch Alarms & Zero-Bill Teardown Automation [50 mins • Assignment (100M) • Hands-On Lab (100M) • 1 Mock Sets • 1 AI Vivas]

Subtopics Covered: Automated `/healthz` HTTP smoke checks in GitHub Actions post-deploy • Wiring ALB `HTTPCode_Target_5XX_Count` alarms to automatic CodeDeploy rollback • Running automated AWS Tag Editor / Boto3 orphan-resource audits • Production Pitfall Guard (Post-Deployment Smoke Checks, CloudWatch Alarms & Zero-Bill Teardown Automation) • Hands-On Verification (Post-Deployment Smoke Checks, CloudWatch Alarms & Zero-Bill Teardown Automation) • Post-Deployment Smoke Checks, CloudWatch Alarms & Zero-Bill Teardown Automation — Interview Readiness • Idiomatic DOCK Implementation: multi-stage BuildKit Dockerfiles, distroless/Alpine images & Compose networks • Linux Namespaces, cgroups v2, OverlayFS Layers & containerd/runc OCI Execution & Memory Lifecycle for Post-Deployment Smoke Checks

Purpose: Ensuring broken container releases roll back automatically within 60 seconds and lab environments never incur surprise cloud bills.

What Topic Is Next → Module 7 Capstone → End-of-Course Live Client Project: Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation — Enterprise Client Production Platform

Module 7 Hands-On Capstone Project (100 Marks): Capstone: End-to-End Containerized Microservice Pipeline (Docker + GitHub OIDC + AWS CDK ECS Fargate) — ClickOps console provisioning causes configuration drift, undocumented snowflake servers, human copy-paste security bugs, and slow disaster recovery. Build a 100% repeatable, multi-environme...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation — Enterprise Client Production Platform • **Confirmed Role:** Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation Engineering Project Intern

Domain & Production Infra: Live Enterprise Cloud & DevOps Engineering Client Engagement (Linux CLI, systemd & Bash, Docker Engine & BuildKit, Non-Root Multi-Stage Dockerfiles, Docker Compose v2 & Volumes, Nginx Reverse Proxy & HTTPS • Git CI/CD + Automated Testing + Live HTTPS Cloud Production)

- Translated live client functional and non-functional requirements into modular Linux CLI, systemd & Bash, Docker Engine & BuildKit, Non-Root Multi-Stage Dockerfiles, Docker Compose v2 & Volumes, Nginx Reverse Proxy & HTTPS production architecture.
- Implemented core application modules, input validation, security hardening, and optimized data/state workflows.
- Completed peer Git pull-request code reviews, automated verification suites, and edge-case debugging sprints.
- Deployed the verified build to live client production with monitoring, documentation, and mentor sign-off.

Target Job Roles: Linux, Docker, GitHub Actions CI/CD & Cloud IaC Automation Developer, Associate Cloud & DevOps Engineering Engineer, Software Development Engineer (SDE-1)

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹4.5 LPA – ₹8.5 LPA **Mid-Level Product Engineer (1–3 Yrs):** ₹9.5 LPA – ₹19.0 LPA **Senior / Multi-Stack Specialist (3–6+ Yrs):** ₹20.0 LPA – ₹38.0+ LPA