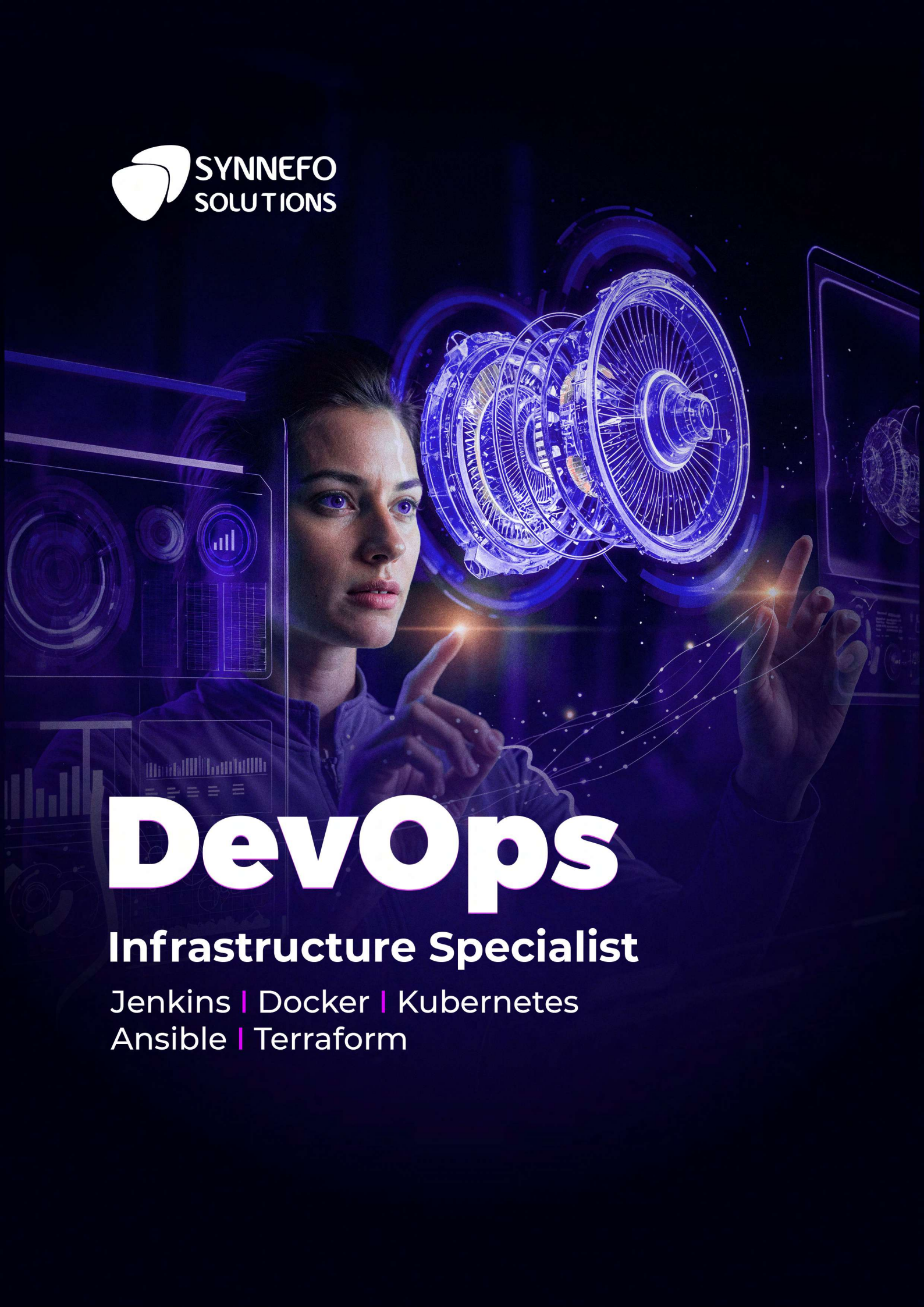
SYNNEFO SOLUTIONS



DevOps

Infrastructure Specialist

Jenkins | Docker | Kubernetes
Ansible | Terraform

The Advantages

Synnefo Solutions is an IT company–based academy delivering industry-focused training with real-world projects and expert mentorship

- 100% Job Guaranteed
- Synnefo SmartSpace: Kerala's First LaaS
- Intensive Internship
- Realtime Projects

Class Mode Details

Course Duration:

10 Months, Including a 4 month internship program

ONLINE OFFLINE HYBRID

About The Course

Course Overview

DevOps Engineering centers on leveraging modern automation and cloud infrastructure to develop, launch, and maintain highly scalable software systems

This program trains you to work like real DevOps engineers using:

- Linux & networking environments
- Cloud infrastructure (AWS)
- Containers & orchestration platforms
- Automation and CI/CD pipelines
- Monitoring & logging systems

What You Will Learn

You will gain hands-on experience with tools like Linux, Git, AWS, Docker, Kubernetes, Terraform, Ansible, Jenkins, Prometheus & ELK, following real industry workflows and DevOps best practices.

Suitable For

Freshers I Students I Software Developers
System Administrators I IT Support
Network Engineers I Career Switchers

Overview

Training Features

- Live projects & real-world case studies
- Hands-on cloud lab access (Synnefo SmartSpace)
- One-to-one mentoring support
- Recorded sessions (Online/Hybrid)
- Mock interviews & career guidance
- Industry-experienced trainers

Key Modules

- Linux Administration
- Cloud Computing Fundamentals
- AWS / Azure (Cloud Platforms)
- Git & GitHub (Version Control)
- Docker (Containerization)
- Kubernetes (Orchestration)
- Ansible (Configuration Management)
- Terraform (Infrastructure as Code)
- CI/CD with Jenkins & GitHub Actions
- ArgoCD (GitOps)
- Monitoring (Prometheus & Grafana)
- Logging (ELK Stack)

Careers

Career Opportunities

- DevOps Engineer
- Cloud Engineer
- Site Reliability Engineer
- Automation Engineer
- Release Manager
- Build Engineer
- Monitoring & Logging Engineer
- Infrastructure Engineer
- Platform Engineer
- Freelance Cloud Consultant

Certifications

About Us

Synnefo is an ISO-certified IT academy offering 100% job-guaranteed training with hands-on experience in real-time NOC, SOC, and Data Center labs. Having trained over 5,000 students, they provide industry-driven learning, NACTET-affiliated certifications, and lifetime placement support through 50+ recruitment partners

Start Your DevOps Journey With Guaranteed Job Support

ENQUIRE NOW

+91 773 601 3411

Near Maharajas
Metro Station, Kochi

www.synnefo.in


SYNNEFO
SOLUTIONS
DevOps
Syllabus

Certified DevOps Infrastructure Specialist (2026 Batch)

- Program Overview
  - **Learning Phase (6 Months):** Intensive technical deep-dives and daily labs.
  - **Internship Phase (4 Months):** Industrial projects and job placement prep.
  - **Daily Format:** 1.5 Hours Theory/Demo + 0.5 Hours Hands-on Lab.
  - **Focus:** Mandatory industry skills for freshers to secure high-paying DevOps roles.

PHASE 1: THE FOUNDATION

1》 Linux & Systems Mastery

- Hardware & Architecture:
  - Understanding CPU cycles, RAM volatile storage, and Disk I/O.
  - Kernel vs. User space; Monolithic vs. Microkernel concepts.
  - 64-bit vs 32-bit architecture and package compatibility.
- The Filesystem & Navigation:
  - Deep dive into FHS (Filesystem Hierarchy Standard): /etc, /var, /bin, /opt.
  - Mastering paths: Absolute vs. Relative; Hidden files and Dotfiles.
  - File operations: touch, cp -R, mv, rm -rf, mkdir -p.
  - Hard Links vs. Soft (Symbolic) Links and Inode structure.
- CLI Power Tools & Text Processing:
  - Standard Streams: stdin, stdout, stderr, and Redirection (>, >>, 2>).
  - Pipes (|) for command chaining.

- Searching: grep (Regex basics), find (by time, size, user), and locate.
- Stream Editors: sed (find/replace) and awk (columnar data processing).

- **User, Permissions & Security:**
  - User/Group management: useradd, usermod, groupadd.
  - Permission Triad: Read, Write, Execute (Numeric vs. Symbolic notation).
  - Ownership: chown and chgrp.
  - Special Permissions: SUID, SGID, and the Sticky Bit.
  - Sudoers configuration (/etc/sudoers) and privilege escalation.
- **System Administration & Monitoring:**
  - Process management: ps, top, htop, kill, nice, renice.
  - Background/Foreground jobs: &, bg, fg, jobs, nohup.
  - Service Management: systemctl (start, stop, enable, status) and Unit files.
  - Scheduling: crontab (syntax and automation) and at.
  - Log analysis: /var/log directory and journalctl for systemd logs.
- **Networking & Troubleshooting:**
  - OSI Model & TCP/IP suite (Layers 3, 4, and 7 focus).
  - IP Addressing: IPv4/IPv6, CIDR notation, and Subnetting.
  - Protocols: DNS (Records: A, CNAME, MX), HTTP/S, SSH, FTP, ICMP.
  - CLI Tools: ping, traceroute, dig, nslookup, netstat, ss, curl, wget.
  - SSH Deep Dive: Key-based auth, SSH-agent, config files, and Port Forwarding.

PHASE 2: AUTOMATION & CLOUD CORE

2 Version Control (Git)

- **Foundations & Local Workflow:**
  - Repository initialization (git init) and Configuration (git config).
  - The lifecycle of a file: Untracked → Staged → Committed.
  - Diffing: git diff (Working tree vs. Staging vs. Local).
- **Branching & Merging:**
  - Branch management: git branch, git checkout -b, git switch.
  - Merging: Fast-forward vs. 3-way Merge; Merge Commits.
  - Rebasing: git rebase (Interactive rebase for commit squashing).
  - Conflict Resolution: Identifying and fixing manual merge conflicts.
- **Remote Collaboration:**
  - Remotes: git remote add, git fetch, git pull, git push.
  - Upstream tracking and Pull Requests (PR) workflows.
  - Git Stash, Cherry-pick, and git revert vs. git reset.
- **Advanced Patterns:**
  - Gitflow vs. Trunk-based development.
  - Submodules and Git Hooks (Pre-commit, Pre-push).

3 Python for DevOps

- **Basics for Scripting:**
  - Data Types: Strings, Integers, Floats, Booleans.
  - Collections: Lists, Dictionaries (JSON-like), Tuples, and Sets.
  - Logic: If-Else, For/While loops, and List Comprehensions.

- **Functional Programming:**
  - Defining functions, Arguments (*args, **kwargs), and Return values.
  - Modules and Packages: Importing math, datetime, random.

- **Systems & Automation Libraries:**
  - os and sys: Directory navigation and CLI argument parsing.
  - Subprocess: Running shell commands from Python.
  - Shutil: High-level file operations (copying, archiving).
  - Requests: Consuming REST APIs (GET, POST) and handling JSON responses.

- **Error Handling & Packaging:**
  - Try-Except-Finally blocks and custom exceptions.
  - Virtual Environments (venv) and Dependency management (pip, requirements.txt).

4 Cloud Infrastructure (AWS) & Serverless

- **Identity & Security (IAM):**
  - Root account vs. IAM Users.
  - Policies (JSON structure), Roles (Service-linked), and Groups.
  - Multi-Factor Authentication (MFA) and CLI Access Keys.
- **Networking (VPC):**
  - Subnets (Public vs. Private) and Route Tables.
  - Internet Gateways (IGW) and NAT Gateways.
  - Security Groups (Stateful) vs. Network ACLs (Stateless).
  - VPC Peering and Endpoint Gateways.
- **Compute, Scaling & Serverless:**
  - EC2: Instance types, AMIs, Key Pairs, and User Data (Bootstrap scripts).
  - Auto Scaling Groups (ASG): Launch Templates and Scaling Policies.
  - Load Balancers: Application (Layer 7) vs. Network (Layer 4).
  - Serverless Computing: AWS Lambda (Event-driven logic), Trigger mechanisms
  - (S3, API Gateway), and AWS Fargate (Serverless Containers).
- **Storage & Databases:**
  - S3: Buckets, Versioning, Lifecycle Policies, and Static Website Hosting.
  - EBS: Volume types, Snapshots, and Multi-attach.
  - RDS: Multi-AZ for HA and Read Replicas for scaling.
  - DynamoDB: NoSQL basics, Partition keys, and Global tables.

- Monitoring & Routing:
  - CloudWatch: Metrics, Alarms, Logs, and Dashboards.
  - CloudTrail: Governance and Audit trails.
  - Route53: Domain registration, Hosted zones, and Routing policies (Latency, Weighted).

5》 Infrastructure as Code (Terraform)

- Language (HCL) & Core Workflow:
  - Providers (AWS, Azure, GCP) and Resources.
  - Variables (Input, Local, Output) and Data Sources.
  - Workflow: init → plan → apply → destroy.

- State Management:
  - terraform.tfstate file: Importance and security.
  - Remote Backends: S3 + DynamoDB for state locking in teams.
  - State commands: state list, state show, state rm.

- Modularity & Reusability:
  - Module structure: main.tf, variables.tf, outputs.tf.
  - Calling modules from Local paths vs. GitHub vs. Terraform Registry.

- Advanced Logic:
  - Functions (lookup, join, element).
  - Meta-arguments: count, for_each, depends_on, lifecycle.
  - Provisioners: local-exec and remote-exec.

PHASE 3: THE MODERN DEVOPS STACK

6 Configuration Management (Ansible)

- **Fundamentals:**
  - Control Node vs. Managed Nodes (SSH requirements).
  - Inventory management: Static (hosts file) and Dynamic inventories.
  - Ad-hoc commands for quick tasks.
- **Playbook Development:**
  - YAML syntax and structure.
  - Tasks, Modules (apt, yum, copy, service, lineinfile).
  - Handlers for service restarts.
- **Variables & Templates:**
  - Variable precedence and scope.
  - Jinja2 Templating: Using logic inside configuration files.
  - Registered variables and Conditionals (when).
- **Advanced Organization:**
  - Ansible Roles: tasks/, vars/, templates/, defaults/.
  - Ansible Vault: Encrypting passwords and SSH keys.
  - Ansible Galaxy: Using community-contributed roles.

7》 Containerization (Docker)

- **Foundations:**
  - Containers vs. Virtual Machines (Shared Kernel vs Hypervisor).
  - Docker Engine Architecture: Daemon, REST API, CLI.

- **Docker Images:**
  - Image layers and Read-only storage.
  - Writing Dockerfiles: FROM, RUN, CMD, ENTRYPOINT, COPY, ADD, ENV, WORKDIR.
  - Multi-stage builds for optimized, small image sizes.
  - Tagging and Image versioning.

- **Runtime & Storage:**
  - Container lifecycle: run, stop, start, pause, rm.
  - Executing into containers: docker exec -it.
  - Volumes (Named vs. Path) and Bind Mounts for persistence.

- **Networking & Compose:**
  - Network drivers: Bridge, Host, None, Overlay.
  - Port mapping (-p) and exposing ports.
  - Docker Compose: Defining multi-container apps with docker-compose.yml.

8》 CI/CD, GitOps & Database Automation

- **GitHub Actions:**
  - Workflows, Jobs, and Steps.
  - Events/Triggers: push, pull_request, workflow_dispatch.
  - Contexts, Secrets, and Environment Variables.
  - Self-hosted vs. GitHub-hosted Runners.

- **Jenkins (The Industry Standard):**
  - Installation and Plugin management.
  - Pipeline as Code: Declarative vs. Scripted Jenkinsfile.
  - Distributed builds: Setting up Agent nodes (Nodes & Clouds).
  - Multi-branch pipelines and Webhooks.

- **GitOps with ArgoCD:**
  - The GitOps principles: Declarative, Versioned, Pulled.
  - ArgoCD Architecture: Application Controller, Repo Server, API Server.
  - App-of-Apps pattern and Sync Policies (Auto-sync, Prune).

- **Database CI/CD:**
  - Database Migrations: Version control for schemas using tools like Flyway or Liquibase.
  - Automating schema updates within the deployment pipeline.

- **DevSecOps (Security Scanning):**
  - SAST: SonarQube for code quality.
  - SCA: Snyk for dependency vulnerabilities.
  - Image Scanning: Trivy or Anchore.

9 Kubernetes (K8s) Orchestration

- **Cluster Architecture:**
  - Control Plane: API Server (Gateway), Etcd (Key-value store), Scheduler (Placement), Controller Manager (Logic).
  - Worker Nodes: Kubelet (Agent), Kube-proxy (Networking), Container Runtime (Containerd).
  - Cluster Setup: Kubeadm (Standard), Minikube (Local), EKS (AWS Managed).

- **Workload & Core Objects:**
  - Foundational: Pods (Lifecycle, Multi-container patterns), Namespaces (Isolation), Labels & Selectors.
  - Controllers: Deployments (Rolling Updates/Rollbacks), StatefulSets (Sticky identity), DaemonSets (One pod per node).
  - Batch: Jobs and CronJobs.
  - Advanced Scheduling: Taints/Tolerations and Node Affinity/Anti-affinity.

- **Networking & Traffic:**
  - Services: ClusterIP (Internal), NodePort (Fixed port), LoadBalancer (Cloud integration).
  - Ingress: Ingress Controllers (Nginx), Ingress Rules (Path/Host based), and TLS/SSL termination.
  - Policy: CNI Plugins and Network Policies (Pod-to-Pod security).

- **Storage & Config:**
  - Persistence: Persistent Volumes (PV), Claims (PVC), and StorageClasses (Dynamic Provisioning).
  - Abstraction: ConfigMaps (Non-sensitive) and Secrets (Encoded data).
  - RBAC: Roles, ClusterRoles, RoleBindings, and ServiceAccounts.

- **Operations & Ecosystem:**
  - Helm: Charts, Templates, values.yaml, and Repository management.
  - Observability: Metrics Server, HPA (Scaling), and Liveness/Readiness/Startup probes.
  - Troubleshooting: kubectl describe, kubectl logs, kubectl get events, kubectl port-forward.