رواد مصر الرقمية

وزارة الاتصالات
وتكنولوجيا المعلومات

# DevOps Command Hub

Graduation Project – Proposal

رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

# Graduation Project Journey

- Team Consist Of 5 Persons , Diverse Background
- Project Idea is Self Proposed , Not yet Reviewed By instructor !

# Our Team

### Ahmed Alhusainy

You can simply impress your audience and add a unique zing.

### Mohamed Sherif

**Director**

You can simply impress your audience and add a unique zing.

### Lamiaa Abdallah

**Developer**

You can simply impress your audience and add a unique zing.

### Mina Mamdouh

**Programmer**

You can simply impress your audience and add a unique zing.

### Mohammed Reda

**Designer**

You can simply impress your audience and add a unique zing.

# Project Planning And Management

01 Proposal Crafting

02 Milestone Establishment

03 Timeline Creation

04 Task Assignment & Roles

# Proposal

At this stage, each team is required to prepare and submit a **Project Proposal Document**. This proposal acts as the foundation of the Graduation Project and must include the following sections:

- **Problem Statement** – a clear definition of the problem the project aims to solve.
- **Objectives** – the goals the team plans to achieve through the project.
- **Methodology / Approach** – the steps the team will follow, along with the **methods** to be applied.
- **Expected Outcomes** – the results or deliverables the project is expected to produce.
- **Tools & Technologies** – programming languages, platforms, frameworks, or tools that will be used.

# Problem Statement

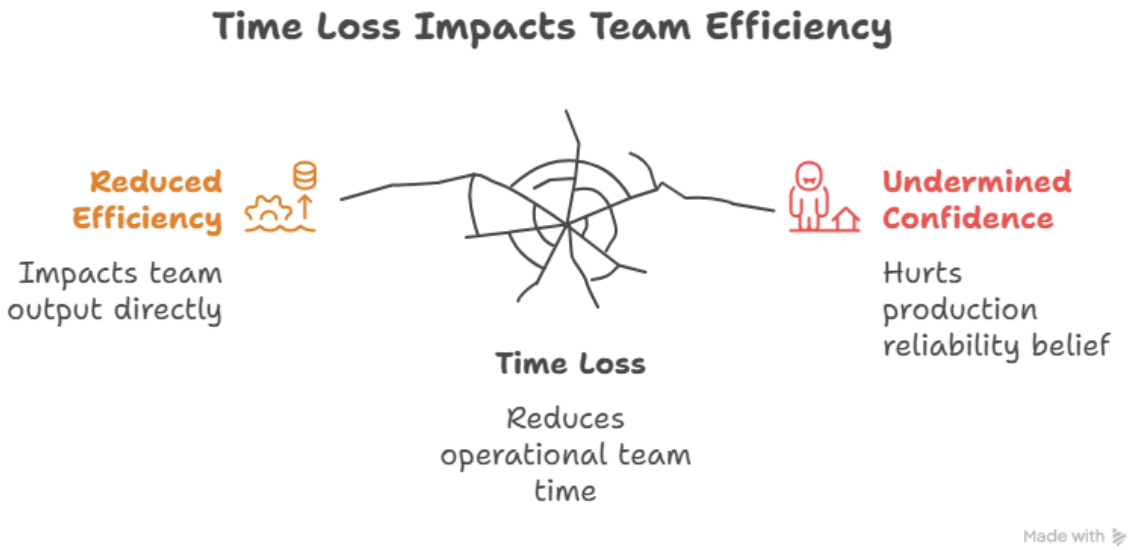
In modern software development, DevOps teams struggle to maintain visibility, stability, and security across continuous delivery pipelines. While multiple tools exist for monitoring, logging, incident management, and security they are often scattered, complex to integrate, and hard to manage in real time.

This fragmentation leads to:

- **Slow detection and response to incidents.**
- **Manual investigation of logs and failed deployments.**
- **Limited automation in recovery or security validation.**
- **Lack of unified dashboards combining DevOps + DevSecOps insights.**

# Problem Statement

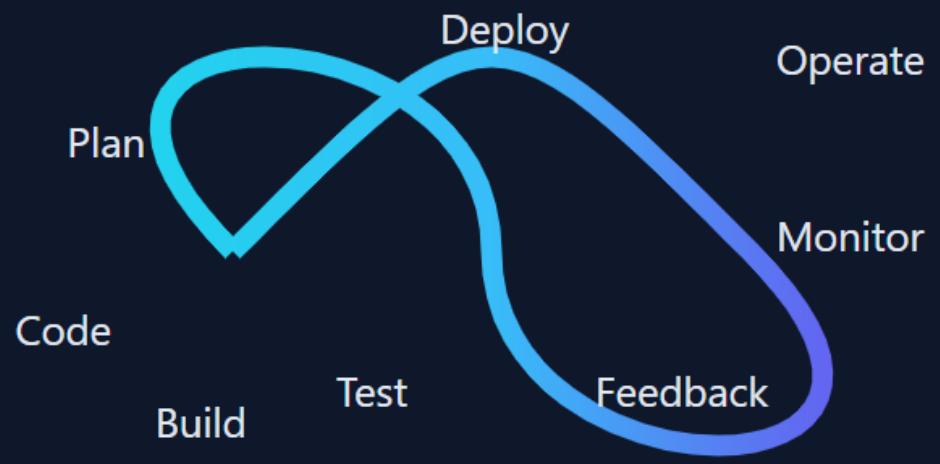
There is no simple, unified, and intelligent platform that helps teams observe system health, detect incidents, analyze logs with AI, and ensure security compliance — all from one hub.



# Objectives

## About the Project

A unified DevOps hub that covers the full infinity loop from planning and coding to monitoring and feedback. Our platform integrates the power of DevOps, the intelligence of AI, and the security of DevSecOps into a single, cohesive command center.



# Objectives

The **DevOps Command Hub** aims to create a **lightweight, unified platform** that helps teams observe, analyze, and improve their DevOps operations with simplicity and intelligence. Through this project, the team plans to achieve the following objectives:

- **Integrate DevOps observability in one dashboard** — combining health metrics, logs, and incidents into a single view.
- **Enable AI-powered log and incident analysis** — provide natural-language explanations and quick insights for developers and operators.
- **Implement basic self-healing automation** — automatically detect and resolve simple infrastructure or service issues.
- **Embed DevSecOps practices** — integrate security scans and highlight vulnerabilities within the CI/CD pipeline.
- **Design a user-friendly web interface** — structured around the DevOps infinity cycle (Deploy, Operate, Observe, Incidents, Logs, Settings).
- **Demonstrate real use cases** — simulate a working pipeline, alerts, and AI responses using open-source tools and sample workloads.

# Methodology / Approach

1. Code/Fetch Online: Simple Full Stack Website
2. Code Repo Setup
   - Add templates: README, CONTRIBUTING, SECURITY, issue/PR templates.

3. Define high level System Architecture – AWS Cloud
   - AWS Architecture Composer
   - Design Project Network Topology (Design VPC: 2 public + 2 private subnets, 1 NAT Gateway, route tables, Internet Gateway)

3. IaC & Configuration - Environment Setup
   - Terraform: VPC, subnets, SGs, IAM roles, S3 bucket, EC2/EKS, ALB.
   - Ansible: baseline config for EC2 nodes (packages, users, hardening, agents).
   - Bastion (not “Bosition”): one small EC2 in a public subnet for controlled SSH.
   - Optional Proxy: Nginx/Squid if you need outbound egress control.

# Methodology / Approach

5. Build Golden AMI
6. Containers & Platform
   - Build Docker images for some the app and utility services.
   - Build Kubernates For the rest app components
   - Add **ALB** + Auto Scaling Group (EC2) or HPA (EKS).
7. Use AWS RDS – With Best Practice
8. CI/CD (simple first)
   - Build & Deploy – CI/CD
   - Push Code to GitHub
   - Add Some Modifications as team to ( Merge , ..etc.)
   - Build & scan image (Trivy)
   - Enable PR checks and protected branches; team collaborates (merge, reviews).

# Methodology / Approach

8. **Observability & Logs (start with managed)**
   - CloudWatch metrics & logs (app/OS).
   - CloudWatch Alarms → SNS → Slack (via AWS Chatbot) or email.
   - Stretch: add Prometheus + Grafana (Grafana Cloud free tier) and Loki or OpenSearch.

9. **DevSecOps (minimum viable)**
   - Image scanning: Trivy in CI.
   - IaC scanning: Checkov (Terraform), kube-score (if K8s).
   - AWS services: Inspector, GuardDuty, Security Hub (enable defaults).
   - Compliance (if time): OpenSCAP on AMI or nodes; basic CIS checks.
   - Encryption: enforce S3 SSE, EBS encryption, HTTPS via ACM.

# Methodology / Approach

10. AI Log Analyzer (small & practical)
    - Start with a simple endpoint that takes a log snippet and returns an explanation:
        - OpenAI API.
        - Try Using Managed Or Self Managed Services
    - Wire it into a “Logs” page: paste filter → “Explain with AI”.

**11. Backups & Events**

- S3 backups for app/config using a Bash script + cron or AWS Backup.
- Event-driven actions: Event Bridge (e.g., nightly snapshots, rotation jobs).

**12. Optionally – Develop**

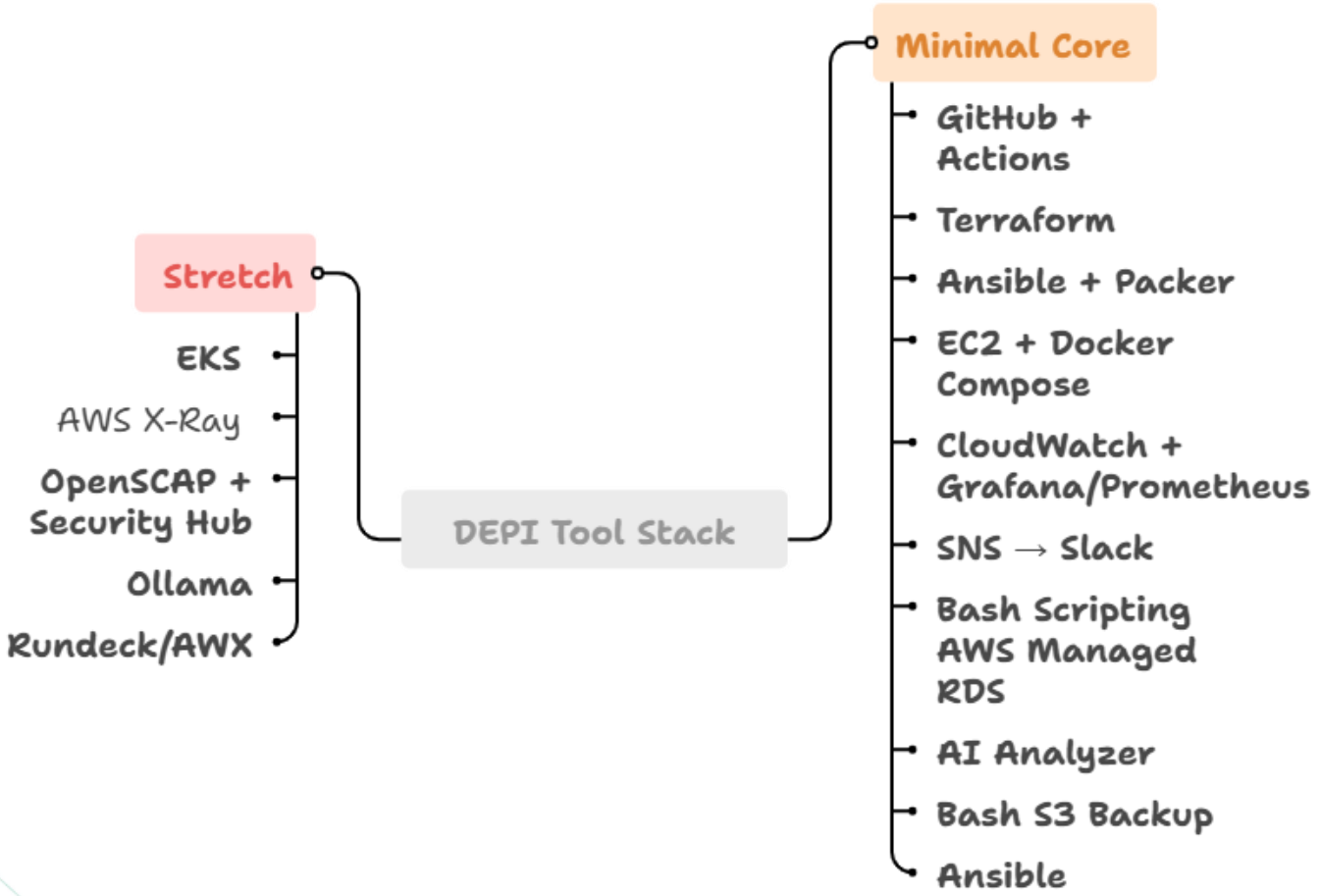
- Present live demo of the Command Hub in operation.

# Methodology / Approach – Notes

- **Secrets management:** use **AWS Secrets Manager**
- **State management:** remote **Terraform state** (S3 + DynamoDB lock).
- **Access & RBAC:** IAM least-privilege roles; restrict SSH to Bastion; MFA for AWS users.
- **Cost guardrails:** budgets + alerts, small instance sizes, stop non-prod at night.
- **Data retention:** CloudWatch log retention (e.g., 14–30 days) to control cost.
- **Tagging & naming:** consistent tags (Project=DEPI-Ninjas, Env=Dev/Prod) for every resource.
- **Security reviews:** basic threat model + checklist before “final demo”.

# Tools & Technologies

- ReactJS
- NodeJS
- Python
- MongoDB/Postgress/mySQL
- GitHub
- GitHub Actions/Jenkins
- EC2 Image Builder
- CloudWatch Dashboards
- AWS X-Ray
- EBS Snapshots via EventBridge
- Docker
- K8s
- YARA/Sigma rules, Trivy, OpenSCAP
- Ansible
- Prometheus/Grafana, loki/ELK
- Ollama for AI
- Terraform / Cloud Formation
- Bash Scripting
- NingX



# Expected Outcomes

1. Apply concepts learned across the curriculum to a working system.
2. Provision AWS infrastructure with **Terraform** (optionally compare with **CloudFormation**).
3. Build and harden **Linux** hosts (users, SSH, logs) using **Ansible** and **Packer** (Golden AMI).
4. Containerize services with **Docker**; run on **EC2 (Compose)** or **EKS**.
5. Implement **CI/CD** in GitHub Actions (build, scan, deploy, rollback).
6. Monitor and analyze with **CloudWatch** (and optionally Prometheus/Grafana).
7. Centralize logs (CloudWatch Logs; optionally Loki/OpenSearch).
8. Add **AI log explanations** (API or local model).
9. Integrate **DevSecOps** (Trivy, Checkov; enable Inspector/GuardDuty/Security Hub).
10. Automate **backups to S3** (script or AWS Backup).
11. Document runbooks, and deliver a **live demo** of the Command Hub.

# Thank You