Master Course on Cloud-Native DevOps with Multiple Projects

Module 1: Advanced Infrastructure as Code (IaC)

Overview

This module takes a comprehensive dive into Infrastructure as Code (IaC) practices, focusing on Terraform and AWS CloudFormation as pivotal tools for managing complex cloud infrastructures. Students will explore the nuances of creating reproducible and scalable cloud environments, emphasizing disaster recovery, high availability, and data replication strategies. This deep dive equips students with the expertise to design and manage cloud infrastructure efficiently, with a strong emphasis on automation and best practices in cloud architecture.

Topics:

- Foundations of IaC: Introduction to the principles of IaC, comparing imperative vs. declarative approaches, and understanding the significance of idempotency in infrastructure management.
- Mastering Terraform: In-depth exploration of Terraform, including state management, modular design patterns, and working with Terraform Cloud for team collaboration and governance.
- Advanced CloudFormation: Techniques for leveraging AWS CloudFormation to its full potential, focusing on nested stacks, custom resources, and integrating CloudFormation with AWS Service Catalog for organizational governance.
- Disaster Recovery and High Availability: Strategies for designing disaster recovery plans using IaC, including cross-region backups, multi-region deployments, and automating failover processes.
- Data Replication Techniques: Implementing data replication and continuity strategies across cloud environments to ensure data integrity and availability.

Module 2: Continuous Integration and Continuous Deployment (CI/CD)

Overview

This module explores the intricacies of setting up advanced CI/CD pipelines using Jenkins and Harness, two powerful tools that enable automation, comprehensive testing, and strategic deployments in software development workflows. Focusing on microservices architecture, which is prevalent in modern FinTech applications, this module guides students through the creation of pipelines that incorporate automated testing, security scanning, and deployment strategies such as blue/green deployments to minimize disruptions during updates.

Topics:

- CI/CD Fundamentals with Jenkins and Harness: Introduction to Jenkins for continuous integration and Harness for continuous deployment, including setup, configuration, and integration between both tools.
- Automated Testing in CI/CD Pipelines: Strategies for integrating various types of automated tests (unit, integration, end-to-end) within pipelines to ensure code quality and reliability.
- Security Scanning: Implementing automated security scanning within CI/CD workflows to identify and remediate vulnerabilities early in the development cycle.

- Deployment Strategies for Microservices: Exploring deployment techniques, with an emphasis on blue/green deployments, to achieve zero downtime during application updates in a microservices architecture.
- Monitoring and Feedback for CI/CD Pipelines: Leveraging monitoring tools to gather insights from deployment processes and creating feedback loops to continuously improve pipeline efficiency and reliability.

Module 3: Containerization and Orchestration

Topics:

- Introduction to Containerization with Docker: Understanding the concept of containerization, creating Docker images, container management, and Docker Compose for multi-container applications.
- Kubernetes Essentials: A comprehensive overview of Kubernetes, including its architecture, core components (Pods, Deployments, Services, etc.), and the kubectl command-line tool.
- Application Deployment with Kubernetes: Practical instructions on deploying applications on Kubernetes, focusing on manifest files, deployment strategies, and health checks.
- Scaling and Load Balancing: Exploring Kubernetes' capabilities for auto-scaling applications based on traffic or other metrics, and implementing load balancing to distribute traffic evenly across multiple instances.
- Self-Healing Applications: Leveraging Kubernetes' self-healing mechanisms to automatically replace failed containers, ensuring high availability and resilience of applications.
- Advanced Kubernetes Features: Introduction to advanced topics such as StatefulSets for stateful applications, managing secrets and configurations with ConfigMaps and Secrets, and securing cluster access with Role-Based Access Control (RBAC).

Module 4: Monitoring, Logging, and Observability

Topics:

- Introduction to Monitoring and Logging: Key concepts in monitoring, logging, and observability. Understanding the role of each in maintaining healthy and performant applications and infrastructure.
- Getting Started with Prometheus and Grafana: Setting up Prometheus for metric collection and Grafana for data visualization. Constructing dashboards that offer actionable insights into application performance and system health.
- Leveraging the ELK Stack for Logging: Implementing the ELK Stack for centralized logging. Techniques for log aggregation, processing, and visualization with Elasticsearch, Logstash, and Kibana.
- Alerting and Incident Management: Configuring alerting rules in Prometheus and integrating with notification systems. Best practices for incident management and response based on real-time data.

- Advanced Observability Techniques: Tracing application workflows with distributed tracing tools. Correlating logs, metrics, and traces to gain a holistic view of system behavior and performance bottlenecks.

Module 5: Security and Compliance in DevOps

Topics:

- DevOps Security Fundamentals: Introduction to the principles of DevSecOps and the importance of incorporating security practices throughout the DevOps lifecycle.
- Managing Secrets with HashiCorp Vault: In-depth exploration of HashiCorp Vault for secrets management, including setup, secure access, dynamic secrets, and best practices for integration with DevOps pipelines.
- Automated Vulnerability Assessment: Techniques and tools for conducting automated vulnerability assessments within CI/CD pipelines, including static and dynamic analysis, container scanning, and dependency checking.
- Compliance in DevOps: Understanding key compliance standards relevant to software development and deployment, such as PCI DSS, HIPAA, and GDPR, and strategies for ensuring compliance through automated checks and balances.
- Implementing Security Policies and Governance: Developing and enforcing security policies across DevOps workflows, utilizing tools and practices for role-based access control, audit trails, and compliance reporting.

Module 6: Cloud-Native Development Practices

Topics:

- Cloud-Native Foundations: Introduction to cloud-native concepts, benefits of cloud-native architectures, and the distinction between monolithic and microservices architectures.
- Microservices Architecture: Designing and developing applications using microservices, focusing on domain-driven design, service discovery, and communication patterns.
- Serverless Computing: Leveraging serverless computing for cost-efficient and scalable application development, including functions as a service (FaaS) platforms like AWS Lambda, Azure Functions, and Google Cloud Functions.
- Data Security in Cloud-Native Applications: Implementing robust data security measures, encryption practices, and secure access patterns to protect sensitive information.
- Compliance with Healthcare Regulations: Understanding healthcare industry regulations such as HIPAA in the US, ensuring applications comply with data protection and privacy standards.

Module 7: Performance Tuning and Optimization

Topics:

- Performance Analysis Techniques: Understanding how to use monitoring tools to collect and analyze performance data, identifying bottlenecks in application and database layers.
- Application Optimization Strategies: Techniques for optimizing application code and architecture, including profiling, caching, and asynchronous processing, to improve response times and resource utilization.
- Database Performance Tuning: Best practices for tuning database queries and schemas, indexing, and leveraging cloud database services for scalability and reliability.
- Cloud Resource Optimization: Strategies for optimizing cloud resource allocation, including auto-scaling, load balancing, and choosing the right compute and storage options to balance performance and cost.
- User Experience and Performance: Approaches to measure and enhance the end-user experience, incorporating frontend optimizations, content delivery networks (CDN), and web performance best practices.
- Project: Performance Optimization Audit for STU Education's Online Learning Platform
- Problem Statement:

Deliverables:

- Comprehensive lecture notes and slides for each module.
- Step-by-step guides for hands-on projects.
- Access to a forum for Q&A and discussions.
- Final assessment test to evaluate understanding and practical skills.

Certificate of completion.

This course structure is designed to provide a thorough understanding of DevOps, from basic concepts to advanced practices, with a strong focus on real-world applications and projects that prepare students for implementing DevOps in their Production environments effectively.

Hands-on Projects

Project 1: Multi-Region, High Availability Cloud Infrastructure for a Web Application

Problem Statement:

As an architect at ABC Company, you are tasked with designing a cloud infrastructure that supports a web application with stringent requirements for high availability and disaster recovery. The current infrastructure does not adequately support these requirements, leading to potential risks in data loss and service downtime. Your challenge is to use Terraform and AWS CloudFormation to create a robust infrastructure that spans multiple regions, incorporates disaster recovery strategies, and ensures data is replicated securely and efficiently.

Objective:

Design and implement a multi-region, highly available cloud infrastructure for a critical web application, incorporating disaster recovery and data replication strategies using Terraform and AWS CloudFormation.

Task Breakdown

Task 1: Designing the Infrastructure

Objective: Create a detailed infrastructure design that meets the high availability and disaster recovery requirements.

Activities:

- Define the architecture, including the selection of AWS regions, VPC setup, subnets, and the distribution of resources to ensure high availability.
- Plan for disaster recovery by designing a multi-region deployment strategy that includes data replication and automated failover mechanisms.

Task 2: Implementing the Infrastructure with Terraform and CloudFormation

Objective: Develop the infrastructure as code using Terraform and AWS CloudFormation.

Activities:

- Use Terraform to provision the foundational cloud resources, emphasizing modularity and reusability of code.
- Supplement with AWS CloudFormation for AWS-specific resources and services, ensuring seamless integration and management.

Task 3: Automating Disaster Recovery and Data Replication

Objective: Implement automated disaster recovery and data replication strategies.

Activities:

- Configure cross-region Amazon RDS instances or DynamoDB tables for automatic data replication.
- Set up automated failover processes using Route 53 health checks and DNS failover policies.

Task 4: Validation and Testing

Objective: Ensure the infrastructure meets all requirements for high availability, disaster recovery, and data replication.

Activities:

- Conduct load testing to validate the high availability setup under simulated traffic spikes.
- Test disaster recovery procedures to verify automatic failover and data integrity in secondary regions.

Deliverables:

- A comprehensive infrastructure codebase using Terraform and AWS CloudFormation, with documentation on design choices and configurations.
- Detailed implementation plans for disaster recovery and high availability setups, including data replication strategies across multiple regions.
- Test reports validating the infrastructure's resilience, scalability, and compliance with the initial requirements, ensuring the web application remains robust and reliable under various scenarios.
- This project within Module 1 provides a hands-on, real-world scenario for students to apply advanced IaC concepts, preparing them to tackle complex infrastructure challenges in production environments.

==========

Project 2: CI/CD Pipeline for a Microservices-Based Application at XYZ FinTech

Problem Statement:

XYZ FinTech is rapidly evolving, requiring a robust CI/CD pipeline to manage the deployment of its microservices-based application. The current deployment process is manual, time-consuming, and prone to errors, leading to downtime during updates that affect customer satisfaction. As the lead DevOps engineer, your challenge is to implement a CI/CD pipeline using Jenkins and Harness that integrates automated testing, security scans, and adopts a blue/green deployment strategy to ensure the application remains available and secure at all times.

Objective:

Design and deploy an advanced CI/CD pipeline for XYZ FinTech's microservices-based application, incorporating automated testing, security scanning, and blue/green deployment to achieve zero downtime during updates.

Task Breakdown

Task 1: Setting Up Jenkins and Harness

Objective: Establish the foundational CI/CD tools for the project.

Activities:

- Install and configure Jenkins for continuous integration, setting up build jobs for each microservice.
- Integrate Jenkins with Harness for continuous deployment, ensuring a seamless flow from code commit to deployment.

Task 2: Integrating Automated Testing

Objective: Ensure code quality and reliability through automated testing.

Activities:

- Implement unit and integration testing within Jenkins pipelines, running tests automatically on every commit.
- Configure end-to-end testing in Harness deployment pipelines to validate functionality in a pre-production environment.

Task 3: Implementing Security Scans

Objective: Incorporate security scanning within the CI/CD process.

Activities:

- Integrate automated security scanning tools (like SonarQube or Snyk) with Jenkins to scan code and dependencies for vulnerabilities.
- Analyze scan results and automate the triage process to address critical vulnerabilities before deployment.

Task 4: Deploying with Blue/Green Strategy

Objective: Minimize downtime and risk during deployments.

Activities:

- Set up blue/green deployment pipelines in Harness, defining criteria for traffic switching to the new version.

- Implement automated rollbacks in case of deployment failure or degraded performance in the green environment.

Task 5: Monitoring and Optimization

Objective: Establish monitoring for continuous improvement of the CI/CD process.

Activities:

- Integrate monitoring tools to track the performance and health of both the CI/CD pipeline and the deployed application.
- Use feedback from monitoring tools to continuously refine and optimize the CI/CD pipeline for efficiency, reliability, and performance.

Deliverables:

- A fully functional CI/CD pipeline configured with Jenkins and Harness, tailored for a microservices architecture.
- Documentation detailing the setup process, integration points, testing protocols, and security scanning procedures.
- A report on the implementation of the blue/green deployment strategy, including metrics on downtime reduction and deployment success rates.

Project 3: Video Processing Application Deployment at DEF Media

Problem Statement :

DEF Media is enhancing its video processing capabilities to meet increasing demand and improve service reliability. As the lead DevOps engineer, your mission is to deploy a new video processing application on Kubernetes, ensuring it can dynamically scale to handle variable workloads, distribute tasks efficiently through load balancing, and maintain service continuity through Kubernetes' self-healing features.

Objective:

To deploy a robust video processing application on Kubernetes for DEF Media that embodies auto-scaling, effective load balancing, and self-healing to guarantee optimal performance and reliability.

Task Breakdown

Task 1: Application Containerization

Objective: Containerize the video processing application for Kubernetes deployment.

Activities:

- Create Dockerfiles for the application's services, ensuring optimization and security practices are adhered to.

- Build and push the Docker images to a registry accessible by Kubernetes.

Task 2: Kubernetes Deployment Configuration

Objective: Configure Kubernetes resources for deploying the application.

Activities:

- Write Kubernetes YAML manifest files for the application's deployments, services, and any necessary ingress controllers for external access.
- Utilize Helm charts for templating and managing Kubernetes resources, streamlining the deployment process.

Task 3: Implementing Auto-Scaling and Load Balancing

Objective: Ensure the application can automatically scale and distribute load.

Activities:

- Configure Horizontal Pod Autoscaler (HPA) for the application, setting appropriate metrics for scaling.
- Implement a LoadBalancer service or an Ingress controller to manage incoming traffic and ensure efficient load distribution.

Task 4: Ensuring Application Resilience

Objective: Leverage Kubernetes' self-healing features for application resilience.

Activities:

- Define readiness and liveness probes for each service, facilitating Kubernetes in managing the application's health.
- Simulate failure scenarios to test Kubernetes' ability to automatically replace failed containers and maintain application availability.

Deliverables:

- Dockerfiles and Kubernetes manifest files or Helm charts for the video processing application, along with detailed documentation on their setup and management.
- A deployment strategy report that outlines the implementation of auto-scaling, load balancing, and self-healing mechanisms, including metrics for monitoring application performance and scalability.
- An evaluation of the deployment's success, highlighting how Kubernetes features have been leveraged to enhance application reliability and performance, along with recommendations for future optimizations.

==========

Project 4: Centralized Observability Platform for GHI Retail's E-Commerce Platform

Overview:

In Module 4, students will master the art and science of implementing sophisticated monitoring, logging, and observability solutions for modern applications and infrastructure. Utilizing industry-standard tools like Prometheus, Grafana, and the ELK Stack (Elasticsearch, Logstash, Kibana), the module covers the end-to-end process of capturing, analyzing, and visualizing operational data. This knowledge enables DevOps professionals to proactively manage system performance, troubleshoot issues efficiently, and enhance user experience through real-time insights.

Problem Statement :

GHI Retail is experiencing rapid growth in its e-commerce platform but faces challenges in maintaining optimal performance and ensuring a seamless user experience. With the complexity of the platform's microservices architecture, the current observability practices are inadequate for detecting and resolving issues promptly. As the lead DevOps engineer, your objective is to set up a centralized observability platform that integrates monitoring, logging, and alerting capabilities to enable real-time insights into the platform's operations.

Objective:

Implement a comprehensive observability platform for GHI Retail's e-commerce platform, utilizing Prometheus, Grafana, and the ELK Stack to facilitate real-time monitoring, alerting, and log analysis.

Task Breakdown

Task 1: Implementing Monitoring with Prometheus and Grafana

Objective:

Set up Prometheus for metric collection and Grafana for visualization to monitor the e-commerce platform's performance.

Activities:

- Install and configure Prometheus to scrape metrics from the e-commerce platform's services.
- Create Grafana dashboards that visualize key performance indicators (KPIs) critical for the platform's operations and user experience.

Task 2: Centralized Logging with the ELK Stack

Objective:

Deploy the ELK Stack for centralized log management, enabling efficient log analysis and troubleshooting.

Activities:

- Configure Logstash to collect logs from various services within the e-commerce platform.
- Use Elasticsearch for log storage and Kibana for creating insightful log visualizations.

Task 3: Configuring Alerting and Incident Management

Objective:

Establish alerting mechanisms for operational anomalies and integrate with incident management workflows.

Activities:

- Define alerting rules in Prometheus based on critical thresholds and anomalies.
- Integrate alert notifications with communication channels and incident management tools used by the DevOps and support teams.

Task 4: Advanced Observability and Optimization

Objective:

Enhance the observability platform with advanced techniques and optimize it for scalability and reliability.

Activities:

- Implement distributed tracing for deeper insight into the e-commerce platform's transactions and workflows.
- Conduct performance tuning of the observability tools to handle increased data volume and ensure scalability.

Deliverables:

- A fully functional observability platform comprising Prometheus, Grafana, and the ELK Stack, tailored for GHI Retail's e-commerce platform, complete with installation and configuration guides.
- Customized Grafana dashboards and Kibana visualizations that provide real-time insights into system performance and user experience.
- Documentation on alerting configurations, incident management procedures, and insights gained from the observability platform that have led to performance optimizations.

==========

Project 5: Securing JKL Bank's Application Deployment Pipeline

Overview:

Module 5 is meticulously crafted to address the critical intersection of security, compliance, and DevOps, offering students a deep dive into the practices, tools, and strategies essential for securing DevOps workflows. By focusing on vulnerability assessments, secrets management, and adherence to compliance standards, this module equips students with the knowledge to implement a security-first approach in any DevOps pipeline. Emphasizing hands-on experience with leading security tools like HashiCorp Vault and automated security scanning solutions, students will learn to navigate the complexities of securing application deployment pipelines while ensuring compliance with industry standards such as PCI DSS.

Problem Statement :

JKL Bank is transforming its software delivery process to adopt DevOps practices but faces significant challenges in meeting stringent security requirements and compliance with PCI DSS standards. As the security architect, you are tasked with redesigning the bank's application deployment pipeline to incorporate a security-first approach, integrating HashiCorp Vault for secrets management, and automating security assessments to ensure the protection of sensitive data and compliance with regulatory standards.

Objective:

Design and implement a secure, compliant application deployment pipeline for JKL Bank, leveraging HashiCorp Vault for secrets management and integrating automated security assessments to meet PCI DSS compliance requirements.

Task Breakdown

Task 1: Integrating HashiCorp Vault for Secrets Management

Objective: Securely manage secrets and sensitive configurations using HashiCorp Vault.

Activities:

- Set up HashiCorp Vault within the bank's infrastructure, ensuring secure storage and access to API keys, database credentials, and other secrets.
- Integrate Vault with the CI/CD pipeline to dynamically inject secrets at runtime, eliminating hard-coded or exposed credentials.

Task 2: Automating Vulnerability Assessments

Objective: Incorporate automated security scanning and vulnerability assessments in the CI/CD pipeline.

Activities:

- Select and integrate automated security scanning tools that support static analysis, container scanning, and dependency checks.
- Configure scanning triggers within the CI/CD pipeline to ensure every build and deployment is assessed for vulnerabilities, with fail-safe mechanisms for critical findings.

Task 3: Ensuring PCI DSS Compliance

Objective: Implement compliance checks and reporting mechanisms to adhere to PCI DSS standards.

Activities:

- Develop a compliance checklist based on PCI DSS requirements relevant to the application and data handling processes.
- Automate compliance checks within the deployment pipeline, including data encryption validation, access controls, and audit logging.

Task 4: Security Policies and Governance

Objective: Establish security policies and governance practices within DevOps workflows..

Activities:

- Define security policies for code reviews, access to production environments, and incident response.
- Implement role-based access controls and audit trails to monitor and govern access to deployment tools, environments, and sensitive operations.

Deliverables:

- A comprehensive security-first deployment pipeline configuration, integrating HashiCorp Vault for secrets management and automated security assessments.
- Documentation detailing the setup, configuration, and integration of security tools and practices within the pipeline, including guidelines for Vault usage and security scanning.
- A PCI DSS compliance report showcasing the compliance checks, findings, and remediation actions taken, affirming the pipeline's adherence to regulatory standards.

Project 6: Serverless Patient Data Processing Application for MNO Healthcare

Overview:

This Module offers an in-depth exploration into the principles and practices of cloud-native development, a pivotal approach for creating applications that leverage the full potential of cloud computing. This module equips students with the knowledge and skills to design, build, and deploy resilient and scalable applications using cloud-native architectures, including microservices and serverless computing. Emphasis is placed on architectural patterns, data security, and regulatory compliance, preparing students to tackle development challenges in highly regulated industries such as healthcare.

Problem Statement:

MNO Healthcare is seeking to modernize its patient data processing system to improve scalability, security, and compliance with healthcare industry regulations. As the solution architect, you are tasked with developing a serverless application that efficiently processes and analyzes patient data, ensuring

the system is capable of handling fluctuating workloads while maintaining stringent data security and regulatory compliance.

Objective:

Design and implement a scalable, secure serverless application for patient data processing at MNO Healthcare, adhering to cloud-native development principles and compliance with healthcare industry regulations.

Task Breakdown

Task 1: Designing the Serverless Architecture

Objective: Architect a serverless solution for processing patient data.

Activities:

- Map out the serverless architecture, identifying key components such as data ingestion, processing functions, data storage, and user interface.
- Select appropriate cloud services for each component of the architecture, ensuring integration and scalability.

Task 2: Implementing Data Security Measures

Objective: Ensure the application adheres to best practices for data security.

Activities:

- Implement encryption in transit and at rest for all patient data handled by the application.
- Design secure access mechanisms, including authentication and authorization for users and services interacting with the application.

Task 3: Developing and Deploying the Application

Objective: Build and deploy the serverless application.

Activities:

- Develop serverless functions for data ingestion, processing, and analysis, using cloud-native development practices.
- Deploy the application components to the chosen cloud platform, configuring autoscaling and monitoring as necessary.

Task 4: Ensuring Regulatory Compliance

Objective: Validate compliance with healthcare industry regulations.

Activities:

- Conduct a compliance audit of the application, reviewing data handling practices, security measures, and access controls against healthcare regulations.
- Document compliance measures and prepare a report detailing how the application meets regulatory standards.

Deliverables:

- A detailed design document outlining the serverless architecture and rationale for component selection.
- Source code for the serverless application, including deployment scripts and configuration files.
- A security and compliance report, demonstrating the application's adherence to data security best practices and healthcare regulations.

==========

Project 7: Performance Optimization Audit for STU Education's Online Learning Platform

Overview:

This module is meticulously curated to arm students with advanced techniques for optimizing application and database performance in cloud environments. Given the dynamic and scalable nature of cloud computing, this module emphasizes practical strategies for identifying performance bottlenecks and implementing optimizations to enhance efficiency, reduce costs, and improve user experiences. Through a blend of theoretical insights and real-world applications, students will become adept at diagnosing and resolving performance issues in cloud-deployed applications, ensuring they are well-prepared to manage and optimize modern software systems.

Problem Statement :

STU Education's online learning platform has experienced significant growth in user traffic, leading to performance issues that negatively impact the user experience, including slow page loads and occasional downtime. As the performance optimization specialist, you are tasked with conducting a comprehensive audit of the platform to identify performance bottlenecks and implement necessary enhancements to ensure the platform can efficiently handle increased traffic and meet user satisfaction benchmarks.

Objective:

Perform a detailed performance optimization audit for STU Education's online learning platform, identifying key bottlenecks and executing optimizations to improve scalability, efficiency, and user satisfaction.

Task Breakdown

Task 1: Conducting the Performance Audit

Objective: Analyze the current performance of the online learning platform.

Activities:

- Utilize monitoring and profiling tools to collect performance data across the application and database layers.
- Identify critical performance bottlenecks affecting user experience, response times, and system scalability.

Task 2: Application and Frontend Optimizations

Objective: Implement optimizations in the application code and frontend.

Activities:

- Refactor inefficient code paths and implement caching strategies to reduce server load and improve response times.
- Optimize frontend assets, implement lazy loading, and leverage CDNs to enhance page load speeds.

Task 3: Database Performance Tuning

Objective: Optimize database interactions to improve data retrieval and storage efficiency.

Activities:

- Analyze and optimize slow database queries, introduce appropriate indexing, and adjust database configurations for optimal performance.
- Consider scaling options provided by the cloud provider, such as read replicas or sharding, to handle increased load.

Task 4: Cloud Resource Optimization

Objective: Fine-tune cloud resource allocation and configurations to meet performance goals efficiently.

Activities:

- Assess and adjust compute and storage resources, implementing auto-scaling policies to dynamically adapt to traffic patterns.
- Review and optimize cloud service costs, ensuring that the platform uses the most cost-effective resources to deliver the desired performance.

Deliverables:

- A comprehensive performance optimization audit report detailing identified bottlenecks, proposed optimizations, and expected outcomes.
- Documentation of implemented application, database, and frontend optimizations, including before-and-after performance metrics.
- A cloud resource optimization strategy, including auto-scaling configurations and cost management plans.

==========

EACH PROJECT IS DESIGNED TO SIMULATE SCENARIOS THAT PROFESSIONALS MIGHT ENCOUNTER IN REAL-WORLD SETTINGS, ALLOWING STUDENTS TO APPLY THEIR KNOWLEDGE OF HARNESS AND GITOPS PRINCIPLES TO SOLVE PRACTICAL PROBLEMS.