Report on Elevate LAB DevOps Internship Projects

Prepared by: Debjyoti Shit

Introduction:

This report outlines the successful completion of four key projects as part of the Elevate Lab DevOps Internship. The projects were designed to provide hands-on experience with modern DevOps practices, covering the entire software development lifecycle from continuous integration and delivery to infrastructure as code and GitOps. The primary objective of this internship was to gain practical skills in automating, monitoring, and scaling applications using industry-standard tools and methodologies. This report details the objectives, tools, and steps involved in each project, demonstrating the practical application of DevOps principles.

Abstract:

The internship focused on four main areas of DevOps: CI/CD, cloud infrastructure, container orchestration, and GitOps. The projects involved setting up a complete CI/CD pipeline using GitHub Actions and Docker, deploying a scalable static website on AWS S3 with Cloudflare, implementing a canary deployment strategy on Kubernetes using K3s and Istio, and establishing a GitOps workflow with ArgoCD. These projects provided a comprehensive understanding of how to build and manage robust, scalable, and automated systems in a cloud-native environment. The successful completion of these projects has resulted in a strong foundational knowledge of DevOps practices and their real-world applications.

Tools Used:

The following tools and technologies were utilized across the four projects:

- **CI/CD:** GitHub Actions
- **Containerization:** Docker, Docker Hub
- **Orchestration:** Kubernetes (Minikube, K3s)

- **Cloud Services:** AWS S3
- **CDN & DNS:** Cloudflare
- **Service Mesh:** Istio
- **GitOps:** ArgoCD
- **Infrastructure as Code:** Docker Compose, YAML, Helm
- **Programming/Scripting:** HTML/CSS, Bash, Node.js/Python

Steps Involved in Building the Project:

Project 1: CI/CD Pipeline with GitHub Actions & Docker:

This project focused on creating a foundational CI/CD pipeline without relying on a cloud provider.

1. **Containerization:** A Dockerfile and docker-compose.yml were created to define the application environment and services.
2. **CI/CD Configuration:** GitHub Actions was configured to automate the following workflow upon code commits:
    a. Run automated tests to ensure code quality.
    b. Build a Docker image of the application.
    c. Push the built image to Docker Hub for storage and distribution.
3. **Local Deployment:** The Docker image was pulled from Docker Hub and deployed locally using Minikube (a local Kubernetes cluster) or a virtual machine to simulate a production environment.

Project 2: Scalable Static Website with S3 + Cloudflare + GitHub Actions

This project demonstrated how to host a highly available and performant static website using serverless technologies.

1. **Website Development:** A simple static website was created using HTML and CSS and pushed to a GitHub repository.
2. **Cloud Storage:** An AWS S3 bucket was configured to host the static website content.
3. **Automated Deployment:** GitHub Actions was set up to automatically synchronize the website content from the GitHub repository to the S3 bucket on every commit to the main branch.

4. **CDN and SSL:** Cloudflare was used to provide a global Content Delivery Network (CDN) for low-latency content delivery and to enable HTTPS for secure communication.
5. **Optimization:** Caching settings and versioning were implemented to improve website performance and ensure users receive the latest content.

Project 3: Kubernetes-Based Canary Deployment with K3s and Istio

This project explored advanced deployment strategies to minimize risk when releasing new features.

1. **Application Deployment:** Two versions of a sample application (stable and canary) were deployed to a lightweight Kubernetes cluster (K3s).
2. **Traffic Management:** Istio, a service mesh, was installed and configured to manage traffic routing between the two application versions.
3. **Canary Release:** Istio's traffic management rules were set to route 80% of the user traffic to the stable version and 20% to the new (canary) version.
4. **Monitoring and Promotion/Rollback:** The performance of the canary version was monitored. Based on the performance metrics, a decision was made to either gradually shift all traffic to the new version (promotion) or roll back to the stable version in case of issues.

Project 4: GitOps Workflow using ArgoCD on Kubernetes

This project implemented a GitOps approach to manage Kubernetes cluster configurations.

1. **ArgoCD Setup:** ArgoCD was deployed to a Kubernetes cluster (K3s or Minikube).
2. **Git Repository for Manifests:** A dedicated Git repository was created to store the Kubernetes deployment manifests for the application.
3. **Application Configuration:** ArgoCD was configured to monitor the Git repository for any changes to the manifests.
4. **Automated Synchronization:** When changes were pushed to the Git repository (e.g., updating the application version), ArgoCD automatically detected the changes and applied them to the Kubernetes cluster, ensuring the cluster state always matched the state defined in the Git repository.

Conclusion:

The completion of these four projects has provided invaluable experience in the core principles and practices of DevOps. Through hands-on application of industry-standard tools, a deep understanding of CI/CD, cloud infrastructure, container orchestration, and GitOps has been achieved. The skills acquired during this internship, from automating application delivery to managing scalable and resilient infrastructure, are directly applicable to modern software development and operations. This experience has built a strong foundation for a career in DevOps and cloud engineering.