CALIFORNIA STATE UNIVERSITY
FULLERTON™

Department of Computer Science

CPSC 597 / 598 PROJECT / THESIS DEFINITION

To the graduate student:
1. Complete a project proposal, following the department guidelines.
2. Have this form signed by your advisor and reviewer / committee.
3. Submit it with the proposal attached, to your advisor.

[X] Project

[ ] Thesis

Please print or type.

Student Name: Alejandro S Pinto Student ID: 821088580

Address: 3320 Quartz Lane D22, (Street) Fullerton California, (City) 928312 (Zip Code)

Home Phone: 7076018042 Work Phone:

E-Mail: alexpinto130@gmail.com Units: 3 Semester: Spring SM25

Are you a Classified graduate student?

[X] Yes

[ ] No

Is this a group project?

[ ] Yes

[X] No

Proposal Date: 05/27/2025

Tentative Date for Demo: 12/10/2025

Completion Deadline: 12/20/2025

Tentative Title: **Automated Kubernetes-Based DevOps Pipeline for Microservices Deployment**

We recommend that this proposal be approved:

	Printed name	Signature	Date
Faculty Advisor	Christopher Ryu, Ph.D.		
Faculty Reviewer	Doina Bein, Ph.D.		
Faculty Reviewer			

Abstract:

Microservices architecture significantly enhances scalability, flexibility, and resilience compared to monolithic applications. However, it introduces complexity in deployment, management, and operation, particularly when utilizing cloud environments. This complexity arises from the intricate integration required among diverse DevOps tools and processes. Motivated by the demand for streamlined operations and robust application management, this project focuses on developing advanced automated DevOps practices.

The primary problems addressed by this project include complex tool integration, automated security and compliance practices, advanced observability, and ensuring high availability and scalability. These challenges are critical, as failing to effectively address them can lead to significant operational inefficiencies, security vulnerabilities, and system unreliability, negatively impacting business operations.

The project goals encompass creating a fully automated CI/CD pipeline, comprehensive security automation, enhanced observability, and automated scalability infrastructure. Expected outcomes include an integrated DevOps solution capable of significantly reducing deployment time, improving system uptime, automating vulnerability detection, and efficiently managing variable workloads. Final deliverables will include a detailed project report, fully operational infrastructure-as-code scripts, automated CI/CD pipeline implementations, and comprehensive monitoring and security solutions.

The proposed solutions involve integrating and automating various industry-standard tools and methodologies such as Kubernetes for orchestration, Terraform for infrastructure-as-code, Prometheus and Grafana for monitoring, ELK stack for logging, Jenkins/GitHub Actions for CI/CD automation, and security tools like SonarQube and OWASP ZAP. This integration is non-trivial due to the complexity of these tools and the precision required for seamless interactions. Successfully implementing these advanced integrations will significantly benefit organizations by improving operational efficiency, security, and reliability.

This proposal details the background and importance of DevOps practices in managing microservices, clearly states the critical problems this project will address, outlines measurable project goals with tasks, and explains the significant benefits of solving these complex challenges. Subsequent sections will elaborate on detailed methodologies, implementation strategies, evaluation metrics, and anticipated project outcomes.

1. Introduction

This project addresses the growing complexity and management challenges in deploying and maintaining modern microservices applications in cloud environments. Microservices architecture, consisting of independently deployable, loosely coupled services, offers substantial benefits in scalability, flexibility, and resilience compared to monolithic applications. However, efficiently managing and deploying such applications presents significant operational challenges, including configuration management, infrastructure provisioning, security compliance, automated scaling, and comprehensive monitoring and observability.

To handle these challenges, DevOps practices have emerged as essential. DevOps integrates software development and IT operations, emphasizing automation, continuous integration and continuous deployment (CI/CD), infrastructure as code (IaC), security automation, and comprehensive monitoring. Kubernetes, an open-source container orchestration tool, has become the industry standard for managing microservices due to its robustness, scalability, and community support.

State-of-the-art DevOps pipelines currently leverage tools such as Terraform for infrastructure provisioning, Docker for containerization, Kubernetes for orchestration, Jenkins or GitHub Actions for CI/CD, and advanced monitoring stacks (Prometheus, Grafana, ELK). Despite these tools, real-world implementation still poses considerable complexity due to the intricate integration required among various services and tools.

Problem Statements:
The primary problems this project addresses are:

1. Complex Integration: Seamlessly integrating multiple DevOps tools and processes into a cohesive pipeline that automates deployment, scalability, and management.
2. Security and Compliance Automation: Implementing continuous and automated security checks and compliance practices across the DevOps lifecycle to minimize vulnerabilities.
3. Advanced Observability and Reliability: Achieving comprehensive, real-time observability and reliability to rapidly diagnose and resolve issues, ensuring minimal downtime and optimal application performance.

4. High Availability and Scalability: Ensuring robust infrastructure capable of automatic scaling and recovery in response to varying workloads and failure scenarios.

2. Project Goals:

Goal 1: Develop a fully automated CI/CD pipeline.

- Task 1.1: Configure automated build and testing processes using Jenkins/GitHub Actions.
- Task 1.2: Integrate automated container deployments with Kubernetes.
- Task 1.3: Validate pipeline through functional and integration tests.

Goal 2: Implement comprehensive security automation.

- Task 2.1: Integrate static and dynamic vulnerability scanning tools (SonarQube, OWASP ZAP).
- Task 2.2: Automate security alerts and remediation workflows.
- Task 2.3: Establish secrets management using HashiCorp Vault.

Goal 3: Enhance system observability and reliability.

- Task 3.1: Configure Prometheus and Grafana dashboards for real-time metrics.
- Task 3.2: Implement ELK stack for log aggregation and analysis.
- Task 3.3: Set up automated alerts and incident response mechanisms.

Goal 4: Achieve automated high availability and scalability.

- Task 4.1: Develop infrastructure as code scripts using Terraform.
- Task 4.2: Configure Kubernetes autoscaling and failover mechanisms.
- Task 4.3: Conduct load testing and system optimization for performance improvements.

These problems are non-trivial because each involves intricate interactions among multiple sophisticated technologies and best practices, requiring detailed technical knowledge and practical skills. Solving these issues is significantly beneficial because it reduces operational complexity, enhances reliability, improves security posture, and supports agile, rapid deployments—benefiting organizations by dramatically increasing efficiency and reducing downtime. This project directly showcases my mastery of DevOps tools and practices gained during my graduate studies, demonstrating my readiness to manage real-world, complex deployment scenarios effectively.

3. Related Work:

DevOps practices have evolved significantly over the past decade, driven by the necessity to rapidly deploy software, reduce operational overhead, and improve reliability and security. Existing DevOps methodologies typically focus on automation of software build and deployment pipelines using Continuous Integration and Continuous Deployment (CI/CD) tools such as Jenkins, GitHub Actions, and GitLab CI. Tools like Terraform and CloudFormation have popularized Infrastructure as Code (IaC), enabling teams to define and provision infrastructure using code, improving consistency and reducing manual errors.

Microservices architecture is a design approach in which software is composed of small, independent services that communicate over well-defined APIs. Each service typically focuses on a specific functionality, making them easier to develop, test, and maintain independently. This architectural pattern significantly improves scalability, resilience, and flexibility but introduces complexities in managing numerous services, their interactions, and their deployments.

Containerization addresses some of these complexities by encapsulating applications and their dependencies into containers. Docker is the predominant tool used for containerization, allowing developers to create lightweight, portable, and isolated environments. Containers streamline deployment across different stages of the software development lifecycle, ensuring consistency from development to production.

Kubernetes has become the industry-standard platform for container orchestration, offering robust capabilities for deploying, scaling, and managing containerized applications. Its extensive ecosystem, scalability, and support for complex microservices architectures make it an essential tool in modern DevOps practices. Despite its advantages, integrating Kubernetes effectively with CI/CD pipelines and managing its associated security and observability concerns remain challenging.

Monitoring and observability tools such as Prometheus, Grafana, and the ELK stack (Elasticsearch, Logstash, Kibana) are widely used to ensure system reliability and facilitate rapid issue diagnosis. However, effectively integrating these tools to provide comprehensive, real-time insights remains challenging for many organizations.

Security automation through tools like SonarQube for static code analysis, OWASP ZAP for dynamic security testing, and HashiCorp Vault for secrets management is increasingly critical.

Despite their individual strengths, organizations often struggle to integrate these tools into a cohesive DevOps lifecycle seamlessly.

This project addresses these integration gaps and operational challenges, proposing advanced integration strategies and automated workflows that streamline deployment processes, enhance reliability, security, and observability beyond current common practices.

4. Proposed Approaches:

This project proposes the following detailed strategies for addressing identified challenges:

CI/CD Pipeline: Utilize Jenkins/GitHub Actions for automating CI/CD processes, leveraging robust scripting capabilities to build, test, and deploy microservices rapidly and reliably. The selection ensures flexibility and extensive community support, allowing easy integration with existing workflows and minimizing overhead.

Infrastructure as Code (IaC): Employ Terraform to script infrastructure provisioning and management, enabling consistent, repeatable, and version-controlled infrastructure setup. Terraform's wide provider support and declarative nature streamline cloud resource management, reducing complexity and human error.

Monitoring and Observability: Integrate Prometheus and Grafana for metric visualization and alerting, along with the ELK stack for centralized log management. These tools provide real-time insights and analytics, significantly enhancing troubleshooting efficiency and system reliability.

Security Automation: Implement SonarQube and OWASP ZAP for static and dynamic security scanning, automating vulnerability detection and alerting. Additionally, HashiCorp Vault will manage secure secrets handling, ensuring comprehensive security posture across the DevOps pipeline.

Scalability and Reliability: Leverage Kubernetes' autoscaling and self-healing features combined with Terraform to ensure high availability and resilience, handling variable workloads dynamically and efficiently. Automated load testing will validate these approaches, ensuring optimal performance and stability under diverse conditions.

5. Required Environment and Resources:

Successfully executing this project demands a comprehensive and robust technical environment encompassing several industry-leading technologies and tools. Kubernetes clusters, particularly AWS Elastic Kubernetes Service (EKS), will form the core of our deployment environment, providing powerful orchestration capabilities essential for managing multiple microservices. Kubernetes was chosen for its extensive scalability, reliability, and strong community support, which ensures ongoing updates and best practices adoption. Terraform will be utilized extensively as our infrastructure-as-code solution to automate the provisioning, configuration, and management of cloud resources. Terraform's declarative configuration language simplifies infrastructure setup, ensures consistency across environments, and reduces potential configuration errors.

Docker containers will encapsulate each microservice, providing a standardized environment across various stages of development, testing, and deployment. Docker's wide adoption and ease of use significantly streamline the software delivery lifecycle. Continuous Integration and Continuous Deployment (CI/CD) processes will be managed using Jenkins or GitHub Actions, chosen for their robust capabilities, ease of integration, and widespread use within the industry, ensuring rapid build, testing, and deployment cycles.

Monitoring tools such as Prometheus and Grafana are critical for tracking application performance metrics and visualizing real-time data, enabling quick responses to potential issues. The ELK stack (Elasticsearch, Logstash, Kibana) will provide centralized logging and analysis capabilities, facilitating rapid identification and troubleshooting of problems. Security is paramount, thus we will incorporate SonarQube for static code analysis and OWASP ZAP for dynamic vulnerability scanning, ensuring comprehensive and continuous security assessments throughout the development lifecycle. HashiCorp Vault will securely manage sensitive data, including API keys and credentials, further enhancing the security posture. Finally, AWS services including EC2 for compute resources, IAM for access control, and CloudWatch for supplementary monitoring will be leveraged to complete a fully integrated and efficient cloud environment.

6. Expected Activities:

The activities planned for this project align closely with a comprehensive software development process model that will ensure organized progression and thorough validation of all project components. Initially, activities will focus on Requirements Gathering and Analysis, involving detailed stakeholder interviews, document reviews, and gap analysis to establish a solid foundation and clarify the project scope.

The subsequent Design phase will entail creating detailed system architecture diagrams, workflow charts, and comprehensive documentation to specify the interactions between components clearly. During this phase, technology selections and strategies for integrating various DevOps tools will be finalized.

Following the design phase, the project will move into Development, starting with Infrastructure as Code (IaC) scripts. Terraform scripts will be developed and tested to automate the deployment and configuration of Kubernetes clusters and AWS resources. The CI/CD pipeline development will occur concurrently, utilizing Jenkins or GitHub Actions to create automated build, test, and deployment workflows.

The next major phase is Integration and Testing, where components such as Kubernetes deployments, CI/CD pipelines, security tools (SonarQube, OWASP ZAP, HashiCorp Vault), and monitoring solutions (Prometheus, Grafana, ELK) will be methodically integrated. Rigorous testing will include functional testing, integration testing, security assessments, performance, and scalability evaluations.

Finally, the Deployment and Documentation phase will include deploying the fully integrated solution into a production-like environment, monitoring system performance, and gathering data for benchmarking. Detailed documentation will be produced to capture system configurations, operational guidelines, troubleshooting procedures, and performance results, culminating in the final project report and demonstration.

7. Project Outcomes:

Upon completion, this project will deliver several well-defined and substantial outcomes. A comprehensive and detailed project report will be provided, thoroughly documenting all phases of the project lifecycle, including methodologies, architectural choices, integration techniques, testing strategies, and final results. This documentation will serve as both a valuable reference for future projects and a testament to the rigorous approach undertaken.

Operational infrastructure-as-code scripts developed with Terraform will be one of the core deliverables, enabling reliable, repeatable, and efficient deployment of microservices infrastructure. These scripts will significantly simplify infrastructure provisioning tasks and reduce potential errors caused by manual configurations. Additionally, a robust and fully automated CI/CD pipeline will be implemented, demonstrating streamlined software delivery processes capable of significantly accelerating deployment times and improving overall productivity.

Comprehensive security automation configurations, integrated with SonarQube, OWASP ZAP, and HashiCorp Vault, will deliver continuous vulnerability assessment and secure management of sensitive data, substantially improving the project's security posture. Enhanced monitoring and observability solutions utilizing Prometheus, Grafana, and the ELK stack will provide in-depth, real-time insights into application performance, reliability, and user experience, facilitating proactive issue resolution.

Performance benchmarking results will form another crucial outcome, clearly demonstrating improvements in scalability, reliability, and system responsiveness under various load scenarios. These outcomes collectively represent a robust, integrated DevOps solution, significantly enhancing operational efficiency, system reliability, security standards, and overall manageability of microservices-based architectures.

8. Schedule:

The project schedule outlined below details the specific tasks, milestones, and deadlines from initial planning through to final delivery and evaluation, ensuring completion by December 2025. Work begins in July 2025 and continues through December 2025, spanning approximately 24 weeks.

July (Weeks 1-4): The project will start with initial planning and requirement gathering, allocating approximately 40 hours to thoroughly review existing documentation, confirm resource availability, and finalize project scope and detailed architecture designs. Milestone: Completion of project scope documentation and architectural diagrams.

August (Weeks 5-8): The next phase involves infrastructure preparation, where approximately 60 hours will be spent developing Terraform scripts for AWS resource provisioning, establishing Kubernetes clusters, and setting up initial environment configurations. Milestone: Fully provisioned and tested Kubernetes infrastructure.

September (Weeks 9-12): Approximately 70 hours will be dedicated to developing automated CI/CD pipelines using Jenkins/GitHub Actions, setting up Docker containers, and integrating them into the Kubernetes environment. Comprehensive integration and initial testing will also occur. Milestone: Operational CI/CD pipeline and initial integration test results.

October (Weeks 13-16): Security and compliance automation will be the focus, involving approximately 50 hours to integrate and configure SonarQube, OWASP ZAP, and HashiCorp Vault. Automated vulnerability scanning and remediation protocols will be thoroughly validated. Milestone: Security automation integrated and validated.

November (Weeks 17-20): This month will focus heavily on monitoring, observability, and performance optimization, spending approximately 60 hours configuring Prometheus, Grafana, and the ELK stack. Extensive load testing and system optimization activities will be conducted to ensure performance requirements are met. Milestone: Comprehensive observability system in place and optimized performance benchmarks established.

December (Weeks 21-24): The final month will consist of approximately 50 hours spent on preparing comprehensive project documentation, conducting final system validations, generating benchmarking reports, and refining the overall implementation. The final report preparation and project demonstration are major tasks during this period. Milestone: Final project presentation and submission of complete documentation.

Month (2025)	**Task**	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**Total Hours**
July	Requirements Gathering & Analysis	10	10	10	10	40
August	Infrastructure Setup (Terraform, Kubernetes)	15	15	15	15	60
September	CI/CD Pipeline Development & Testing	18	18	17	17	70
October	Security Integration & Testing	12	12	13	13	50
November	Monitoring, Observability & Optimization	15	15	15	15	60
December	Documentation, Reporting, Final Demonstration	20	20	10		50
Total Hours						330

9. References:

[1] Kubernetes Official Documentation, "Kubernetes Basics," 2025. [Online]. Available: https://kubernetes.io/docs/home/

[2] Terraform Official Documentation, "Introduction to Terraform," 2025. [Online]. Available: https://www.terraform.io/docs/index.html

[3] Docker Official Documentation, "Get Started with Docker," 2025. [Online]. Available: https://docs.docker.com/get-started/

[4] Jenkins User Documentation, "Getting Started with Jenkins," 2025. [Online]. Available: https://www.jenkins.io/doc/

[5] GitHub Actions Documentation, "Introduction to GitHub Actions," 2025. [Online]. Available: https://docs.github.com/actions

[6] Prometheus Documentation, "Introduction to Prometheus," 2025. [Online]. Available: https://prometheus.io/docs/introduction/overview/

[7] Grafana Labs Documentation, "Grafana Overview," 2025. [Online]. Available: https://grafana.com/docs/grafana/latest/

[8] Elastic Stack Documentation, "Getting Started with Elastic Stack," 2025. [Online]. Available: https://www.elastic.co/guide/index.html

[9] SonarQube Documentation, "Analyzing Source Code," 2025. [Online]. Available: https://docs.sonarqube.org/latest/

[10] OWASP ZAP Documentation, "ZAP User Guide," 2025. [Online]. Available: https://www.zaproxy.org/docs/

[11] HashiCorp Vault Documentation, "Vault Concepts," 2025. [Online]. Available: https://www.vaultproject.io/docs/concepts

[12] AWS Documentation, "Getting Started with AWS," 2025. [Online]. Available: https://docs.aws.amazon.com/index.html