CBD 3375

DevOps and Cloud Computing for Canadian Enterprises

Project Proposal

An AI-Powered Story Generation Application: Automating Cloud-Native Deployments with DevOps, IaC, and GitOps.

Group E

<u>Team Members</u>

Galgamuge Emmanuel Fernando - C0918066

Vihangi Kolamunna - C0903980

Chris Mary Bulatao - C0908671

William Binitie - C0906287

Yolimar Rios - C0904675

Lambton College | In Toronto

Table of Contents

Table of Figures

1. ABSTRACT

An AI-Powered Story Generation Application: Automating Cloud-Native Deployments with DevOps, IaC, and GitOps.

This project aims to implement a robust DevOps pipeline to automate deployments for an AI-powered short-story generating cloud-native web application. The project's primary objective is to demonstrate the use of DevOps practices such as Infrastructure-as-Code (IaC), Continuous Integration/Continuous Delivery (CI/CD), GitOps and Monitoring in the cloud to deliver secure, quality and efficient applications.

GitLab will be used as the primary Source Control Management (SCM) platform, and its DevOps features will be used to establish the automation of the CI/CD pipeline. The application will be hosted on Microsoft Azure and integrated with Azure OpenAI services based on GPT-3.5. Terraform will manage the provisioning of cloud resources and infrastructure, while GitOps will ensure that the infrastructure and application are efficiently version-controlled and managed.

With the incorporation of Grafana, Azure Monitor service tools will be integrated to track the system's health and performance in real-time, allowing it to detect and resolve anomalies.

In summary, the project's central focus is to showcase the capabilities of integrating different DevOps tools to create an efficient, reliable, highly scalable and automated pipeline for cloud-native applications, maintaining the quality of product delivery.

Keywords: *CI/CD, Cloud-native, DevOps, GitOps, GPT-3.5, Grafana, IaC, Microsoft Azure, OpenAI, SCM, Terraform.*

2. STATEMENT OF NEED

2.1. The Demand for DevOps Automation.

The growing demand for Automation has led to the need for more reliable and efficient application delivery. In contrast, traditional manual deployment, mostly based on on-premises servers, is often unreliable due to the slowness, human errors, high maintenance costs, and non-flexibility of scaling as per the demand (Faustino et al., 2022).

Introducing DevOps practices such as Continuous Integration and Continuous Delivery (CI/CD) for deployment automation reduces the time-to-market drastically while ensuring the application's stability (Buttar et al., 2023; Chauhan, 2024).

We aim to address the need for this DevOps automation by leveraging DevOps practices and tools such as GitLab, Terraform, Microsoft Azure and its services.

2.2. Migration to Cloud-Native

Managing cloud infrastructure manually and on-premises exposes the application to unnecessary complexities and risks, such as not being highly available or scalable based on demand and being exposed to threats. Managing on-premises infrastructure requires extra IT support, increased maintenance costs, high capital, a limit to scalability, and an increased risk of data (MOREFIELD, 2022).

Migration to the cloud will allow businesses to use built-in services (SaaS, IaaS, PaaS) within their applications while ensuring fault tolerance, security, high availability, pay for what you use and scalability (Boillat & Legner, 2014).

2.3. Reliability and Up-to-date AI-Driven Applications.

Applications based on Artificial Intelligence (AI) require constant updates and training to enhance the models. This requires businesses to gather datasets and train models in a timely manner, with reliable deployments without breaking the application. If this is not done, the models will be outdated and/or unreliable and will affect the business directly.

Building cloud-native applications will help us integrate already available services (PaaS, SaaS, IaaS) on the cloud in our application, thus reducing development time, costs and errors. By integrating DevOps practices, we can ensure that the code is tested and deployed continuously

automatically, reducing the risk of your application going down. Furthermore, in our specific scenario, building cloud-native AI-based applications, we can use pre-trained models in the cloud that are constantly updated, and businesses will not need to worry about training and building models, thus saving more costs, time and resources (Garbis & Chapman, 2021).

2.4. Who Benefits?

- **Developers** – Helps developers to focus on what they are specifically intended to do rather than worrying about the delivery. Automating tests will save time and allow the detection of errors and vulnerabilities before deployment.
- **Businesses** – The time-to-market will improve thus will be able to stay ahead of competitors and build trust. Cloud adoption allows businesses to budget costs ahead with low capital overall spending less cost spent on IT resources.
- **End Customers** – All users using the application will have exceptional user experience and will serve their needs on the go with high reliability and availability from anywhere, provided they have an internet connection.

2.5. Planning and Visualization

Maintenance of up-to-date documentation to fully understand the project, architectural diagrams will be introduced to specify each tool used for the project as well as the flow from the commit to the final deployment of the application. Additionally, Gantt Charts will be used to provide a detailed timeline for the project deliverables and milestones from start to end.

3. PROJECT ACTIVITY, METHODOLOGY & OUTCOMES

3.1. Project Activity

The following breakdown of activities gives us an overview of the project, which needs to be achieved to implement a successful DevOps pipeline to deploy an AI story generation cloud-native application.

Activity	Description
Development of an AI-powered Story Generator	• Integrate Azure OpenAI services to develop the backend for the application. • Create a simple User Interface for users to interact with the application and generate stories.
Terraform and Infrastructure Setup	• Terraform will be used to provision all cloud resources and configure networking.
CI/CD and GitOps Implementation	• CI/CD will be used to automate deployments and run the terraform scripts to provision resources.
Monitoring Setup	• Real-time monitoring will be done on the deployed application to detect health or performance anomalies via inbuilt Microsoft Azure services.

Figure 1: Project Activities

3.2. Tools and Technologies

- Microsoft Azure – the cloud platform used to host our application and infrastructure.
- Azure OpenAI Service – The web application uses OpenAI features to generate the stories for our web application.
- Azure Kubernetes Service (AKS) – used to orchestrate the containers.
- Terraform – used to automate cloud resource provisioning.
- Docker – used for containerization of the web application.
- GitLab – Used for source control management and GitOps operations.
- Azure Managed Grafana – used for monitoring of the application.

3.3. Project Methodology

3.3.1. Infrastructure setup (Waterfall)

Waterfall was chosen for this phase because infrastructure setup often has well-defined criteria and a straightforward, sequential process.

Deliverables:

Requirement gathering: Technical requirements (server specs, storage needs, networking, security protocols, architecture blueprint, high-level design diagrams.

Design: Detailed infrastructure design, infrastructure as code (IaC) templates.

Implementation: Set up physical or virtual servers, network configuration, storage allocation, and IaC code deployment (Terraform).

Testing: server and network testing, load testing, failover testing, and security tests.

Deployment: Deployed infrastructure in production.

3.3.2. Pipeline Development (Agile)

Agile is ideal for the iterative development of CI/CD pipelines with ongoing feedback from stakeholders.

Deliverables:

- Pipeline Design and Setup: Initial pipeline structure, Git setup (e.g., Git), containerization setup (Docker, Kubernetes).
- Continuous Integration (CI): Automated build scripts, unit testing integration, integration with version control system.
- Continuous Deployment (CD): Automated deployment scripts, deployment environment setup, staging environment configuration.

3.3.3. Application Development (Agile)

This phase follows the Agile methodology, allowing for iterative releases and continuous feedback. The application development will be broken into small, manageable chunks known as sprints.

Deliverables:

- Application Design: UI/UX mockups, user stories, database design, API design.
- Features Development: core features (database integration), unit test cases, code documentation.
- Feature Expansion & Iteration: Additional features (user roles), refined UI/UX, integration with CI/CD pipeline.
- Security & Testing: Application security, user acceptance testing (UAT), bug fixes,
- Final Application Delivery: Final build of the application, user documentation, and deployment to production.

3.3.4. Monitoring and Optimization

Monitoring and optimization activities will be integrated into regular sprints, with frequent reviews to ensure that performance, stability, and efficiency are continuously improved.

Deliverables:

- Monitoring Framework Setup: Grafana integration, dashboards, and alerts setup
- Application Performance Monitoring: Integration and KPI baselining
- Initial Optimization Based on Monitoring Data:
- Load Testing & Scaling: Initial performance tuning based on real-time data.
- Security Monitoring & Optimization: Load testing and scaling improvements.
- Continuous Feedback & Optimization: Continuous feedback, reporting, and optimization

3.4. Architectural Diagram

The architectural diagram below depicts the overall components of the project. It is a high-level illustration of the whole project.

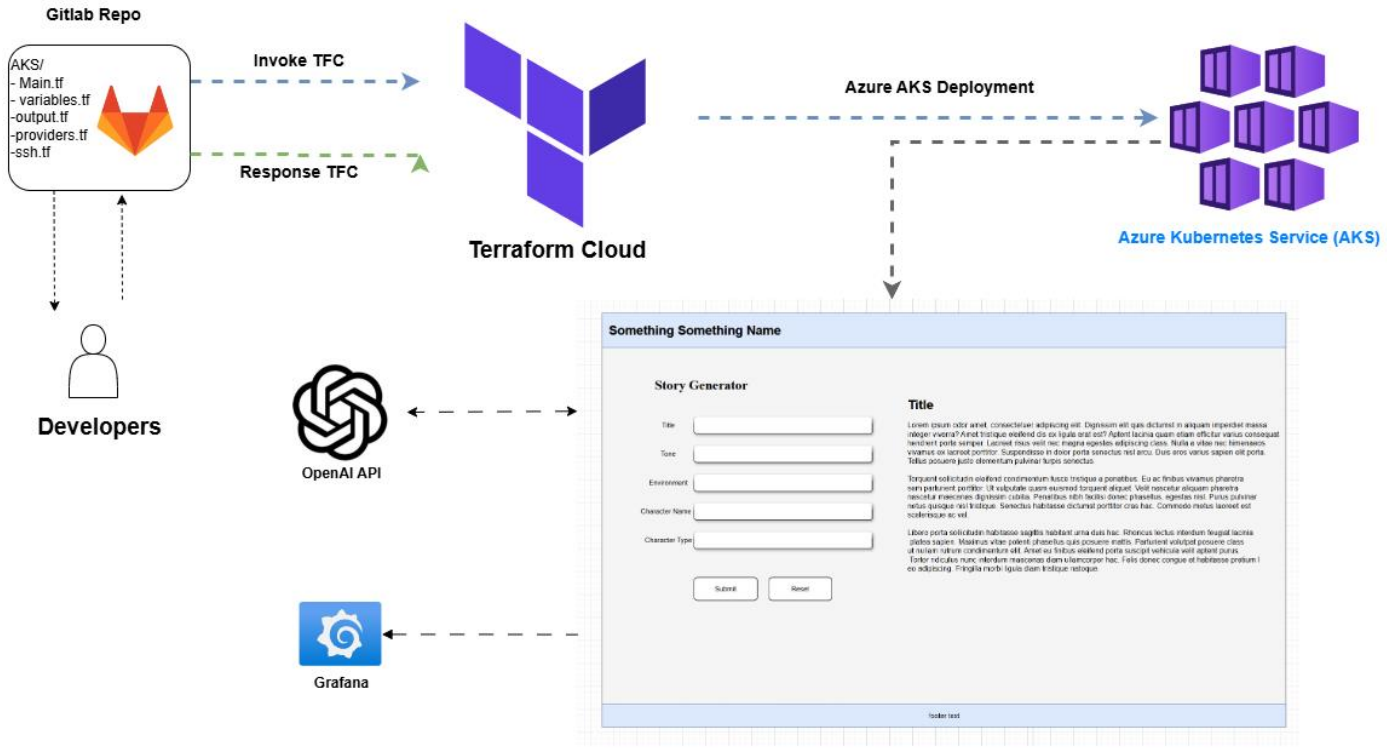

Figure 2: Architectural diagram.

3.5. Gantt Chart

The Gantt chart below shows the proposed timeline for the successful completion of the project. The project will span from the 4th of September 2024 till the 02nd of December 2024. The whole project is broken down into 7 main parts, where each part has its own subtasks.

ID	Task Mode	WBS	Task Name	Duration	Start	Finish	Assignee
1		1	**Requirements Gathering**	**6 days**	**Wed 24-09-04**	**Wed 24-09-11**	
2		1.1	Initial Meeting	1 day	Wed 24-09-04	Wed 24-09-04	All
3		1.2	Document Functional Requirements	4 days	Wed 24-09-04	Mon 24-09-09	Galgamuge
4		1.3	Finalize Non-Functional Requirements	6 days	Wed 24-09-04	Wed 24-09-11	William
5		2	**Story Generator Backend Development**	**6 days**	**Wed 24-09-11**	**Wed 24-09-18**	
6		2.1	Set Up API Endpoints	2 days	Wed 24-09-11	Thu 24-09-12	Galgamuge
7		2.2	Implement Core Story Logic	6 days	Wed 24-09-11	Wed 24-09-18	Galgamuge
8		3	**Front-end Development**	**10 days**	**Wed 24-09-18**	**Tue 24-10-01**	
9		3.1	Create User Interface Wireframes	5 days	Wed 24-09-18	Tue 24-09-24	Chris
10		3.2	Build Main UI Components	9 days	Thu 24-09-19	Tue 24-10-01	William
11		3.3	Integrate Backend API with Frontend	8 days	Sun 24-09-22	Tue 24-10-01	William
12		4	**Infrastructure Setup (Terraform)**	**21 days**	**Wed 24-10-02**	**Wed 24-10-30**	
13		4.1	Provision Resources (Compute, Storage)	14 days	Wed 24-10-02	Mon 24-10-21	Chris
14		4.2	Configure Networking and Security	20 days	Wed 24-10-02	Tue 24-10-29	Chris
15		4.3	Set up Monitoring and Alerts	4 days	Fri 24-10-25	Wed 24-10-30	Vihangi
16		5	**CI/CD Pipeline Configuration**	**16 days**	**Wed 24-10-23**	**Wed 24-11-13**	
17		5.1	Set Up GitOps CI	10 days	Wed 24-10-23	Tue 24-11-05	Yolimar
18		5.2	Configure Automated Build and Test Pipeline	16 days	Wed 24-10-23	Wed 24-11-13	Yolimar
19		6	**Monitoring Setup**	**6 days**	**Wed 24-11-13**	**Wed 24-11-20**	
20		6.1	Implement Monitoring: Grafana and azure monitor	6 days	Wed 24-11-13	Wed 24-11-20	Vihangi
21		7	**Final Deployment and Submission**	**13 days**	**Mon 24-11-25**	**Wed 24-12-11**	
22		7.1	Deploy to Production Environment	6 days	Mon 24-11-25	Mon 24-12-02	All
23		7.2	Validate Production Deployment	6 days	Mon 24-11-25	Mon 24-12-02	All
24		7.3	Submit Final Project Report	8 days	Mon 24-12-02	Wed 24-12-11	All

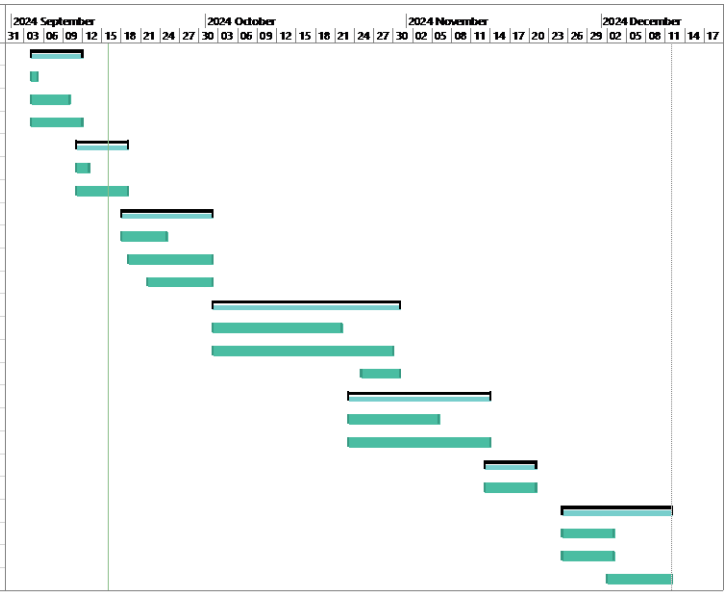

Figure 3: Gantt chart.

3.6. Project Outcomes

The following project outcomes are expected after the successful completion of the project.

- A fully functional AI story generation cloud-native web application.
- Automated application deployments and infrastructure provisioning.
- Real-time application status visibility (monitoring).
- A demonstration of the best DevOps practices for deploying cloud-native applications.

4. EVALUATION

As the application's users will be general public users, anyone with or without technical expertise will be able to use this application and evaluate it.

The project evaluation, from a technical point of view, will be based on the following criteria.

Criteria	Purpose of Evaluation
Functionality	To make sure the web application serves its intended purpose and serves as a proof of concept for the proposed project.
System design and architecture	To evaluate whether the architecture and design meet all the requirements stated.
Pipeline automation	This is to ensure the pipeline deploys the application without manual intervention.
Infrastructure provisioning	To ensure cloud infrastructure is deployed without manual intervention.
Monitoring	This is to ensure real-time metrics about the status of the application are provided.
Limitations and future work	To identify what could have been improved and what limitations are present in the prototype.

Figure 4: Evaluation criteria.

5. REFERENCE

Boillat, T., & Legner, C. (2014). Why Do Companies Migrate Towards Cloud Enterprise Systems? A Post-Implementation Perspective. *Conference on Business Informatics*, *1*, 102–109. https://doi.org/10.1109/CBI.2014.46

Buttar, A. M., Khalid, A., Alenezi, M., Akbar, M. A., Rafi, S., Gumaei, A. H., & Riaz, M. T. (2023). Optimization of DevOps Transformation for Cloud-Based Applications. *Electronics*, *12*(2). https://doi.org/10.3390/ELECTRONICS12020357

Chauhan, A. (2024). DevOps in the Cloud: Streamlining Software Delivery and Deployment. *International Journal of Advanced Research in Science, Communication and Technology*, 240–249. https://doi.org/10.48175/IJARSCT-17837

Faustino, J., Adriano, D., Amaro, R., Pereira, R., & da Silva, M. M. (2022). DevOps benefits: A systematic literature review. *Software: Practice and Experience*, *52*(9), 1905–1926. https://doi.org/10.1002/SPE.3096

Garbis, J., & Chapman, J. W. (2021). Infrastructure and Platform as a Service. *Zero Trust Security*, 173–183. https://doi.org/10.1007/978-1-4842-6702-8_14

MOREFIELD. (2022, May). *On-Premise vs. Cloud Pros and Cons | Which is Better?* https://morefield.com/blog/on-premises-vs-cloud/