Project Canario

candidate number
124

12 December 2024

Oslo Metropolitan University

Contents

1 Introduction

Canario Project is focused on improving Canario's operations, a company known for designing and hosting websites for high-profile clients such as celebrities. Although Canario excels at creating unique and visually appealing sites, it faces several technical and operational challenges behind the scenes. These challenges include manual testing, a disorganized and risky release process, limited automation, and poor version control. These problems put the company's reputation and long-term competitiveness at risk.

The goal of this project is to design a modernized technical infrastructure that addresses these key problems. By introducing automation, version control, and a streamlined release process, the project aims to make Canario's workflows more efficient, reliable, and scalable. A gradual transition plan is proposed to ensure that these improvements can be implemented without disrupting the company's ongoing operations.

In addition to the technical changes, the report also includes recommendations for adopting professional practices that will help Canario achieve operational excellence. Although the focus is on technical solutions, these procedural improvements are equally important for long-term success.

This report suggests a prototype to tackle Canario's most urgent operational problems. Focuses on three key areas such as automation, revision control, and improving the release process.The solution offers a strong starting point to solve these issues. However, it does not aim to address all the challenges Canario is facing.The report acknowledges that some changes will need to be introduced gradually. In addition, further adjustments to the company's structure and workflows will be necessary to fully benefit from the new system. These additional adjustments are not covered in this report.

2 Case Analysis

To better understand Canario's current operational challenges, let's first examine their existing workflow. Figure 1 illustrates the typical process for developing and deploying a website at Canario. This figure highlights the manual nature of their operations. It also shows the points where inefficiencies and risks are most prominent.

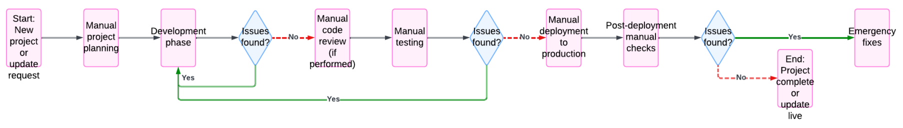

Start: New project or update request
Manual project planning
Development phase
Issues found?
No
Yes
Manual code review (if performed)
Manual testing
Issues found?
No
Yes
Manual deployment to production
Post-deployment manual checks
Issues found?
Yes
No
End: Project complete or update live
Emergency fixes

Figure 1: Overview of canarios current workflow

Looking at the workflow diagram, we can see that Canario's process involves a lot of manual work, doesn't follow consistent standards, and uses risky ways to update websites.

Canario's current work processes are causing major headaches, a problem shared by many in the web development world. While they're known for creating eye catching and unique websites, their old-school approach is now more of a burden than a benefit.These old fashioned practices are making it hard for Canario to work efficiently, expand their business, and keep up with rival companies. If they don't address these problems soon, Canario risks losing clients.In this section of the report, we'll take a close look at these challenges and suggest practical solutions to overcome them.

To start, I will group Canario's problems into categories and see how they affect the company. The table below shows these problems clearly. This will help us understand what Canario needs to do to stay successful and keep up with other companies.

Category	**Challenges**	**Impact on Canario**
Technical	- Manual testing and manual release processes	- Increased risk of human error, slow and unreliable website updates
	- Lack of automation (testing, deployment)	- Slow response to client needs and high chance of downtime
	- No version control (Git)	- Difficult to track changes and developers are unable to collaborate efficiently
Procedural	- Lack of standardized governance and decision-making	- Inconsistent project management and ad-hoc decisions lead to operational inefficiencies
	- Poor resource allocation (human and technical)	- Bottlenecks in workflows where developers lack the right tools or expertise at the right time
Organizational	- No performance metrics or KPIs	- Difficult to measure progress or identify areas for improvement
	- Lack of empowerment among employees (no tools or training for independent decision-making)	- Developers feel disempowered, reducing innovation and adaptability

Table 1: Challenges breakdown for Canario

2.1 Technical Challenges

Canario's technical challenges primarily revolve around the lack of automation and structured processes, which limits the efficiency and reliability of their operations.

- **Manual Testing and Release Process:** At present, testing is done manually, and pushing updates live involves manual changes to live websites. This creates a high risk of human error and often results in embarrassing public mistakes, as seen in the recent incident with Eminem's tour announcement. Without automation, updates are slow, and quality is not consistently guaranteed.

- **Lack of Automation:** Canario does not currently utilize automated testing or deployment pipelines. In the fast paced world of web development, automated processes are essential for ensuring that changes can be made quickly and accurately without introducing new issues into the live environment.

- **Absence of Version Control (Git):** The lack of version control tools like Git means that there's no structured way to track changes made to the codebase. Without this, developers find it difficult to collaborate, track changes, or revert to previous versions if something goes wrong, further compounding the risk of errors.

2.2 Procedural Challenges

Beyond technical deficiencies, Canario suffers from weak governance and resource management, which have led to inefficiencies across its operations.

- **Lack of Standardized Governance:** Currently, there are no clear guidelines or standardized procedures for managing projects at Canario. Decisions are made on an ad-hoc basis, often without proper oversight. This leads to inconsistent results and makes it hard to maintain quality and efficiency across projects.

- **Poor Resource Allocation:** Human and technical resources are not assigned to projects in a balanced way. Some developers have too much work, while others have too little, which slows down progress for everyone. On top of that, the team doesn't have the right tools and technology to work efficiently, making the problem even worse.

These procedural issues prevent Canario from operating at full capacity. Without a clear governance structure, it's impossible to manage resources effectively or ensure consistency across projects.

2.3 Organizational Challenges

Finally, Canario's lack of internal metrics and empowerment culture limits its ability to evolve and improve.

- **No Performance Metrics (KPIs):** Canario does not track key performance indicators (KPIs) or other metrics that would allow them to measure progress and identify areas for improvement. Without a clear sense of how well their processes are working, the company is unable to make data driven decisions to improve efficiency or quality.

- **Lack of Empowerment Among Employees:** Canario's developers are not given the right tools, training, or authority to make informed decisions. This lack of empowerment creates a culture where employees feel disempowered, reducing innovation and making it difficult for the company to adapt to new challenges.

Canario's challenges are a reflection of broader trends seen across the web development industry. Many companies with outdated infrastructures face similar issues, such as the inability to scale due to manual processes and the absence of clear governance structures. This is especially true for Canario, where the absence of automation, version control, and effective resource management makes it hard for them to stay competitive.

These organizational issues extend beyond Canario, reflecting common problems faced by many companies in the industry. Businesses often struggle to create workplaces that promote innovation and accountability. When companies fail to empower their teams, they risk falling behind as employees lose motivation and become disconnected from the company's goals.

To address these challenges and foster continuous growth and improvement, companies need to provide their employees with the necessary tools, training, and authority. This approach helps build a work culture where teams can adapt to challenges and consistently find better ways to work.

Therefore, Canario and companies in similar situations must adopt solutions that goes beyond technical fixes. While implementing automation pipelines, version control, and cloud infrastructure will certainly help improve scalability and reduce human error, these technical improvements must be complemented by organizational changes. By addressing both technical and organizational aspects, companies can create a more resilient and adaptive work environment, better equipped to thrive in the rapidly evolving web development industry.

3 Vision and Design of the Solution

The goal of this project is to create a streamlined, automated, and scalable technical infrastructure for Canario that addresses its current limitations and allows it to grow more efficiently. This vision extends beyond just fixing the immediate technical problems such as lack of automation, manual processes but to transforming the company's overall approach to web development and hosting.

The future infrastructure will focus on achieving operational excellence through the following key objectives:

1. Automation: Automating key tasks such as testing, deployment, and monitoring will reduce human error, speed up releases, and ensure consistent quality.

2. Scalability: Leveraging cloud infrastructure and containerization technologies will allow Canario to scale its operations efficiently, handle more clients, and respond quickly to new business needs.

3. Reliability: Implementing version control, continuous integration, and feature flags will create more reliable and safer development workflows, minimizing risks during updates and reducing downtime.

4. Operational Transparency: Introducing standardized processes, governance policies, and performance metrics will allow Canario's management to monitor progress, identify bottlenecks, and make data-driven decisions.

5. Empowerment and Accountability: By providing the right tools and processes, developers will be empowered to make independent decisions while ensuring accountability through proper documentation, version control, and automated testing.

3.1 Technical Architecture and Key Components

The proposed technical architecture is designed to address Canario's immediate pain points while providing a scalable, flexible foundation for future growth. The design consists of several integrated components, which work together to automate workflows, improve collaboration, and reduce risks during development and deployment.

Here's an overview of the key components:

Component	Purpose	Benefits
Version Control (Git)	Implement version control to track changes, enable collaboration, and create an audit trail of all code modifications.	- Ensures accountability - Simplifies collaboration across teams - Makes rollbacks and code recovery easier
CI/CD Pipeline (GitLab)	Use GitLab's CI/CD tools to automate testing and deployment processes whenever new code is committed to the repository.	- Automates testing and deployment - Reduces human error - Ensures faster, safer releases
Feature Flags	Implement feature flags to control the release of new features without affecting the entire system.	- Reduces risk of new feature releases - Enables safe, incremental rollouts
Cloud Hosting	Use cloud infrastructure (e.g., AWS, Azure, GCP) to host websites, enabling easy scalability, flexibility, and disaster recovery.	- Provides scalability - Increases flexibility - Reduces reliance on single servers
Test Automation	Introduce automated testing to ensure all code is thoroughly tested before going live, using tools like Selenium or JUnit.	- Reduces the likelihood of bugs reaching production - Speeds up testing process - Ensures consistent quality
Monitoring and Alerts	Set up monitoring tools (e.g., Prometheus, Grafana) to track system health and performance, with alerts for critical issues.	- Detects issues early - Reduces downtime - Helps maintain system reliability
Containerization (Docker)	Use containers to ensure consistency between development, testing, and production environments.	- Simplifies deployment - Ensures consistent environments across teams and servers

Table 2: Overview of Key Components for Canario's Future Infrastructure

3.2 Design of the prototype

The prototype developed as part of this project focuses on solving key operational issues at Canario by automating workflows, improving version control, and enabling more reliable deployments. Here's a more detailed explanation of the design and how it fits into the broader vision for Canario's future infrastructure:

3.2.1 Version Control (Git) and GitLab Integration

Canario currently struggles with keeping track of code changes, working as a team, and fixing mistakes. Version control with Git helps solve these problems. It acts like a safety net for our code. If something breaks, we can easily go back to a working version.

This is especially important for Canario because we need to deliver high-quality websites for big clients without errors. With Git, developers can work on different parts of a project at the same time without stepping on each other's toes. It also makes sure we always know who changed what and why. By adding this tool, we're making Canario's development process much more organized and reliable.

3.2.2 Continuous Integration and Continuous Deployment (CI/CD)

Imagine if every time we updated a website, it automatically checked for mistakes, fixed itself if something was wrong, and safely uploaded the new version. That's what a CI/CD pipeline does!

For Canario, this means no more stressful manual updates or downtime. The pipeline automatically tests every change before it goes live and deploys only when it's safe. This makes updates faster and reduces errors. It's like having a robot assistant for our development process, ensuring Canario's websites always work smoothly, even for big clients like celebrities.

3.2.3 Feature Flags for Incremental Feature Rollout

Feature flags are like on-off switches for parts of a website. They let us try out new features safely without affecting the whole system. For Canario, this means we can roll out changes to small parts of a website, test them, and turn them off instantly if something goes wrong.

This is really helpful for big clients like celebrities, where even small mistakes can create big problems. For example, if a new VIP ticketing option has an issue, we can turn it off without disrupting the rest of the website. With feature flags, Canario can keep improving its websites without risking downtime or errors.

3.2.4 Cloud Hosting for Scalability

To support Canario's need for scalability, flexibility, and high availability, the application for Eminem's ticketing system has been deployed using OpenStack, an open-source cloud computing platform. OpenStack provides Canario with a robust infrastructure-as-a-service (IaaS) solution, enabling the company to scale its resources dynamically based on traffic demands, ensuring that the website remains responsive and available even during high-demand events, such as ticket sales for major concerts.

3.3 Extending Beyond the Prototype

While this prototype demonstrates key features like version control, automation, and CI/CD, the full vision for Canario's infrastructure goes beyond what has been implemented in this initial phase. To fully realize the benefits, the company should:

- **Scale the CI/CD pipeline**: Extend the pipeline to handle more complex projects and multi-stage deployments. Implement automated performance testing, security scans, and code quality checks to ensure robust and reliable releases.
- **Implement Canary Releases**: Introduce a canary release strategy as part of the deployment process. This approach allows new features or updates to be gradually rolled out to a small subset of users before full deployment. Canary releases complement the existing feature flag system, providing an additional layer of risk mitigation. By testing changes in a limited real-world environment, Canario can detect and address potential issues early, minimizing the impact on the overall user base. This strategy aligns with the company's need for reliable, low-risk deployment methods, especially for high-profile clients.
- **Enhance Monitoring and Alert Systems**: Implement comprehensive monitoring tools like Prometheus or Grafana to track system health, performance metrics, and user behavior patterns. Set up real-time alerts for critical issues and create dashboards for easy visualization of key performance indicators (KPIs).
- **Advance Containerization and Orchestration**: Expand the use of Docker and introduce Kubernetes for orchestration. This will create more robust, scalable, and easily manageable containerized environments across development, testing, and production workflows, facilitating seamless scaling and deployment.
- Re**fine Governance and Documentation**: Establish company-wide documentation standards, project management practices, and policies for change management. Implement a knowledge base system to ensure consistency, transparency, and efficient knowledge sharing across all teams and projects. This will support better decision-making and maintain operational excellence as the company grows.

These steps will help Canario build on what we've started, making the company more efficient and ready for future challenges.

3.4 Transition Plan for Implementation

Given Canario's need to continue working on ongoing projects, the implementation of this new infrastructure must be gradual. Here's a proposed phased approach

- **Phase 1:Establish Version Control and CI/CD Pipelines**

  In the first phase, Canario will introduce version control and automated pipelines. Git will be implemented as the primary version control tool. The team will receive training on Git basics, including branching and merging. GitLab will serve as the platform for managing repositories. This phase will also involve setting up CI/CD pipelines to automate testing, building, and deployment processes. The pipelines will first be tested on small, low-risk projects. This will help identify issues early. Once refined, the system will be rolled out to all new projects. These changes will improve collaboration and reduce the risks of errors in deployments.

- **Phase 2: Migrate Current Projects to Cloud Infrastructure**

  The second phase will focus on moving existing projects to the cloud. An audit will be conducted to determine the order of migrations, starting with smaller, less complex applications. Docker will be used to containerize these applications. This approach will ensure consistency between development, testing, and production environments. The containerized applications will then be deployed to a cloud platform, such as OpenStack. Testing will be performed to verify performance and stability under simulated traffic loads. This phase will provide Canario with a more scalable and reliable hosting solution.

- **Phase 3: Implement Feature Flags and Automated Testing**

  The third phase will introduce feature flags and automated testing. Feature flags will allow Canario to control the rollout of new features without affecting the entire system. Developers will receive training on how to use feature flags effectively. These flags will be managed dynamically using tools like Unleash. Automated testing tools, such as Selenium or JUnit, will also be integrated into the CI/CD pipelines. This will ensure all code changes are thoroughly tested before deployment. This phase will improve the safety and quality of updates while enabling more flexible feature releases.

- **Phase 4:Expand Monitoring and Introduce Advanced Tools**

  The fourth phase will focus on improving visibility, reliability, and scalability in Canario's infrastructure. Monitoring tools such as Prometheus and Grafana will be deployed to track critical system metrics, including performance, uptime, and resource usage. Real-time alerts will be configured to notify the team of potential issues, enabling quick responses to prevent downtime.

For container orchestration, Docker Swarm will be implemented initially to manage and scale containerized applications. Docker Swarm provides a simpler setup and integration with the current Docker-based infrastructure, making it an ideal starting point. However, as Canario's operational needs grow, the company will evaluate transitioning to Kubernetes. Kubernetes offers advanced features like automated scaling, self-healing, and robust orchestration for large-scale deployments. This dual approach ensures that Canario can adapt its orchestration strategy as its projects and workloads evolve.

This phase will ensure that Canario's system remains stable under varying loads, scales effectively with demand, and provides the necessary tools to monitor and optimize performance continuously.

This phased approach will help Canario modernize its operations while minimizing risks. Each phase builds on the previous one, ensuring a smooth transition to a more efficient, reliable, and scalable system.

3.5 Achieving Operational Excellence

In this subsection, we're going to look at how we're going to achieve operational excellence with the tools and plans we've introduced. We'll explore how each part of our solution helps Canario work better and smarter, following key principles that make operations run smoothly.

First, we're setting up clear rules for everyone to follow. By using tools like Git for tracking changes, and pipelines for automatic testing and deployment, we're making sure everyone works in the same, organized way. This helps reduce mistakes and makes it easier for team members to work together.

Next, we're making sure Canario uses its resources wisely. With our new cloud setup, Canario can easily adjust how much computing power it uses for each project. This means they'll always have what they need, without wasting money on unused resources.

We're also giving Canario's team members more power to make decisions. With new tools for testing and controlled feature releases, developers can work more independently. This helps create new ideas and makes everyone feel more responsible for their work.

Automation is a big part of our plan. We're setting up systems that will do many tasks automatically, like testing new code and putting it live on websites. This saves time and reduces human error.

We're not just automating one thing, though. We're bringing in tools to automatically watch how the systems are running, alert the team if there are problems, and even adjust resources as needed. This creates a smart, self-managing system.

Lastly, we're setting up ways to measure how well everything is working. Using tools like Prometheus and Grafana, Canario can keep track of important numbers that show how healthy and efficient their systems are. This helps them spot problems early and keep improving over time.

By putting all these ideas into action, we're helping Canario become a stronger, more efficient company. They'll be able to create better websites more quickly, respond faster to what their clients need, and stay ahead in the competitive web development world. It's like giving Canario a superpower to do great work and keep getting better every day!

4 From Concept to Execution

In this section, we explain how the technical solution for Project Canario is being built. To provide a clear overview of our proposed solution, we'll start by examining the overall architecture, as illustrated in Figure.

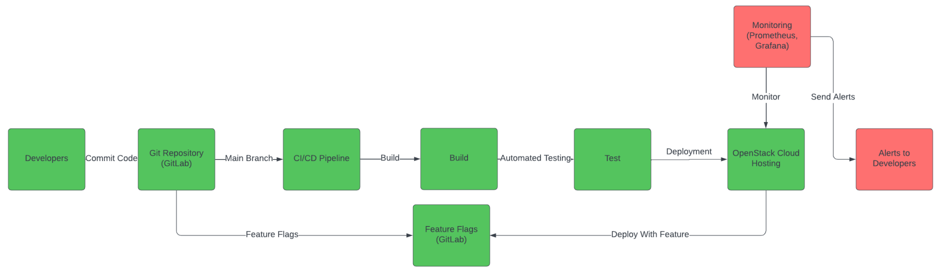


Figure 2: Streamlined Development and Deployment Architecture

This diagram showcases how all the components of our solution interact, from code development to deployment and monitoring. It provides a visual roadmap of the improvements we're introducing to Canario's workflow.

Each part of the process is then broken down into simple steps, showing how all the pieces fit together. This approach makes it easy to see how the solution works and why each step is important.

So far, the prototype has been developed up to the point of deploying applications on OpenStack cloud hosting, as shown in the right side of the diagram. The next steps, like adding monitoring tools and alert systems (represented in the top right of the diagram), are planned for future work. By focusing on one stage at a time, we ensure the project progresses smoothly and effectively.

In the following subsections, we'll delve into the details of each component, explaining how they address Canario's current challenges and contribute to a more efficient, reliable, and scalable system.

4.1 Version Control

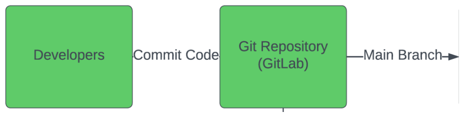


Figure 3: Version Control Workflow

The implementation of Git-based version control, integrated with GitLab, is a foundational improvement that directly addresses Canario's operational challenges. Previously, the absence of structured version control resulted in difficulties tracking changes, resolving conflicts, and ensuring code consistency issues that slowed development and increased the risk of errors.
Git resolves these issues by:

- **Centralizing Change Tracking**: Every modification to the codebase is logged, providing clear accountability and a comprehensive change history.
- **Enabling Seamless Collaboration**: Branching and merging workflows allow developers to work on features independently while maintaining overall code integrity.
- **Facilitating Quick Rollbacks**: In the event of issues, reverting to previous stable versions is straightforward, reducing downtime and ensuring reliable updates.

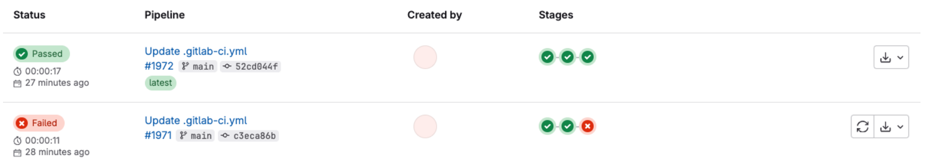
Version control doesn't operate in isolation, however. To fully realize its potential, it must work hand-in-hand with tools that ensure consistency across environments. This is where the Dockerfile becomes essential. While Git manages the what of code changes, the Dockerfile manages the how and where those changes are deployed.
By incorporating Docker into the workflow, the application benefits from several key advantages such as:

- **Environment Standardization**: The Dockerfile ensures the application behaves consistently across development, testing, and production environments, regardless of system differences.
- **Seamless Deployment**: The Dockerfile defines the containerized setup, including dependencies and configurations, creating a reproducible environment for every update pushed through Git.
- **Scalability**: Containerized applications with Docker can be easily scaled and deployed on cloud infrastructure without conflicts or manual intervention.

For example, in Canario's prototype, the Dockerfile starts with an official Nginx image, integrates version-controlled website content, and sets up the application for deployment. This integration between Git and Docker means that every commit to the repository can be deployed predictably and efficiently using the same standardized containerized environment.

Together, Git and Docker transform Canario's development process from a fragmented and error-prone system into a cohesive, reliable, and scalable workflow. This combination directly addresses the challenges Canario faces with manual processes and inconsistent environments, laying the groundwork for the robust infrastructure outlined in this project.

4.2 CI/CD Pipeline



Status	Pipeline	Created by	Stages
Passed 00:00:17 27 minutes ago	Update .gitlab-ci.yml #1972 main 52cd044f latest		✓ ✓ ✓
Failed 00:00:11 28 minutes ago	Update .gitlab-ci.yml #1971 main c3eca86b		✓ ✓ ✗

Figure 4: CI/CD Pipeline, Successful and failed pipeline runs showing the status of build, test, and deploy stages in GitLab

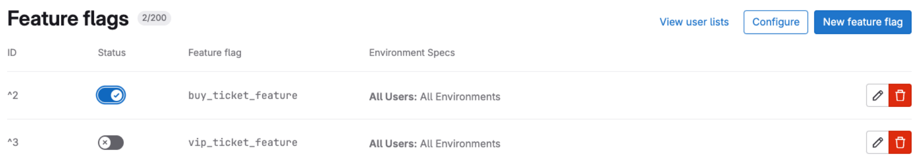
The implementation of a CI/CD pipeline using GitLab addresses several operational challenges Canario faces, particularly those related to manual, error-prone updates. The pipeline automates the process of testing, validating, and deploying changes, creating a faster and more reliable workflow. This transformation ensures that Canario can handle updates with the speed and precision required to meet client demands.

The CI/CD pipeline integrates seamlessly with version control (Git) and Docker, ensuring consistency across development, testing, and production environments. When new code is committed to the repository, the pipeline automatically initiates a sequence of steps. First, it builds a Docker container to encapsulate the code and its dependencies, ensuring that the application behaves the same across all environments. Next, it runs automated tests to identify and resolve any issues before the code reaches production. Once validated, the pipeline deploys the updated container to the cloud using a rolling update strategy, ensuring uninterrupted service during deployment. Figure 4 illustrates this process, showing successful and failed pipeline runs with the status of build, test, and deploy stages in GitLab.

This automated workflow eliminates the risks associated with manual testing and deployment, such as downtime or human error. For Canario's high-profile clients, such as those managing major event ticket sales, the ability to deploy safe, tested updates quickly is critical. Additionally, the pipeline includes rollback functionality, allowing Canario to revert to a previous stable version if an issue arises, further enhancing reliability.

By automating these processes, the CI/CD pipeline not only improves operational efficiency but also ensures that Canario can meet its goals of delivering scalable, reliable, and high-quality services to its clients.

4.3 Feature Flags



Feature flags 2/200

ID	Status	Feature flag	Environment Specs
^2	✓	`buy_ticket_feature`	**All Users:** All Environments
^3	✗	`vip_ticket_feature`	**All Users:** All Environments

Figure 5: Feature Flag Configuration – Example of feature flags used to manage the availability of regular and VIP ticket options dynamically in the application.

Feature flags are an important addition to Canario's workflow. They allow the team to enable or disable specific features without redeploying the entire application. This adds flexibility and reduces risks during updates.

In the prototype, feature flags were used to demonstrate how this technology can improve operations. For instance, as shown in Figure 4, the "buy_ticket_feature" flag controls the standard ticket option, while the "vip_ticket_feature" flag manages VIP tickets. These flags allow teams to test features safely, monitor their performance, and turn them off quickly if issues arise. This ensures the rest of the system remains stable and unaffected.

For Canario's high-profile clients, feature flags provide a safer way to roll out updates. Features can be gradually introduced to smaller user groups, helping to identify and fix issues early. This minimizes disruptions and protects the user experience.

Feature flags are managed using Unleash, a specialized tool. The application fetches the current status of each flag dynamically from the Unleash API. This ensures the website always operates with the latest settings. Real-time control like this allows Canario to adapt quickly to client needs or unexpected challenges.

By adopting feature flags, Canario has made its update process safer and more flexible. This innovation complements other improvements, such as CI/CD pipelines and version control, enabling Canario to deliver reliable and scalable services. The prototype demonstrates how this approach can enhance operations and meet the needs of demanding clients.

5 Technologies and Alternatives

The proposed technologies in the template are not necessarily the only ones available. Alternatives were considered for each component, but the chosen tools were selected based on Canario's specific needs for scalability, efficiency, and reliability. For version control, Git was chosen over SVN. Git's decentralized model supports better collaboration and version tracking, while SVN, though simpler, is centralized and less flexible for Canario's workflows.

For CI/CD, GitLab CI/CD was selected instead of GitHub Actions. GitLab's seamless integration with version control simplifies automation and deployment. While GitHub Actions is flexible, it requires more customization, which could complicate implementation and slow progress at Canario.

For containerization, Docker was preferred over virtual machines (VMs). Docker is lightweight and faster to deploy, ensuring consistency across development and production environments. VMs, while offering stronger isolation, are resource-intensive and slower, making them less practical for agile workflows.

These choices reflect Canario's operational priorities, focusing on tools that address immediate challenges while supporting future growth. While alternatives like SVN, GitHub Actions, and VMs have their own strengths, the chosen tools better align with Canario's goals of achieving operational excellence, reducing inefficiencies, and building a scalable, reliable infrastructure.

6 Conclusion

This project marks an important milestone for Canario, addressing key challenges and setting the stage for growth and improvement. By adopting automation, streamlined workflows, and scalable technologies, the company can enhance reliability, speed up project delivery, and provide better experiences for its clients.

The proposed solutions, such as Git for version control, GitLab CI/CD pipelines, feature flags with Unleash, and a robust cloud infrastructure, directly tackle Canario's current problems. These tools reduce errors, make collaboration easier, and ensure consistency across all stages of development. With automation and containerization, Canario is well-equipped to handle the demanding needs of high-profile clients with confidence and efficiency.

This project does more than fix immediate problems—it prepares Canario for long-term success. The gradual implementation plan ensures that improvements are introduced without disrupting ongoing work. This thoughtful approach demonstrates how innovation can be incorporated effectively and sustainably.

The changes extend beyond technology, fostering organizational growth as well. Employees are empowered with better tools and clearer guidelines, while governance practices and performance metrics ensure processes are more organized and measurable. These updates create a stronger, more efficient workplace where teams can perform at their best and contribute to Canario's success.

For lasting impact, Canario must continue to build on this foundation. Future steps like introducing advanced monitoring systems, expanding CI/CD pipelines, and increasing automation will further strengthen operations. These improvements will help Canario adapt to new challenges, stay competitive, and consistently deliver excellent results to its clients.

This project is not just about upgrading tools but about building a better future for Canario. By focusing on learning, innovation, and continuous improvement, the company can position itself as a leader in its field and ensure long-term growth and success.

7 appendix

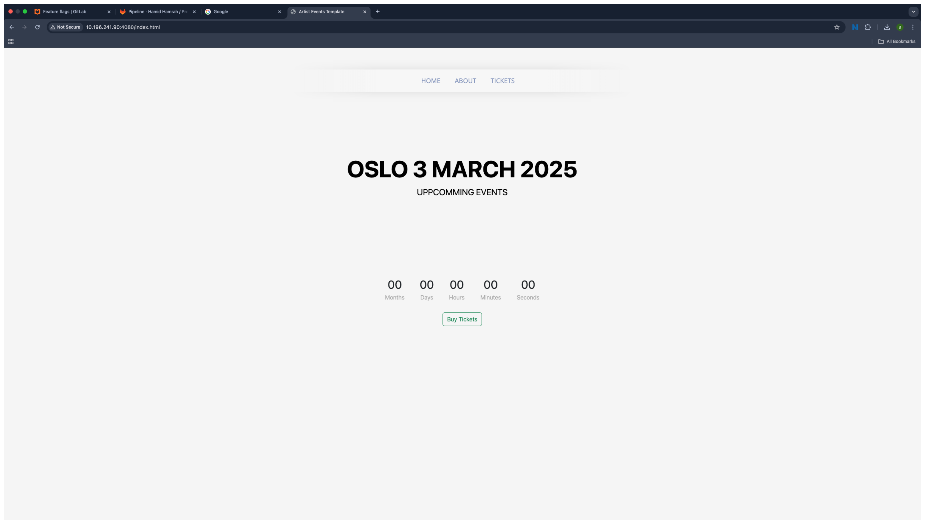
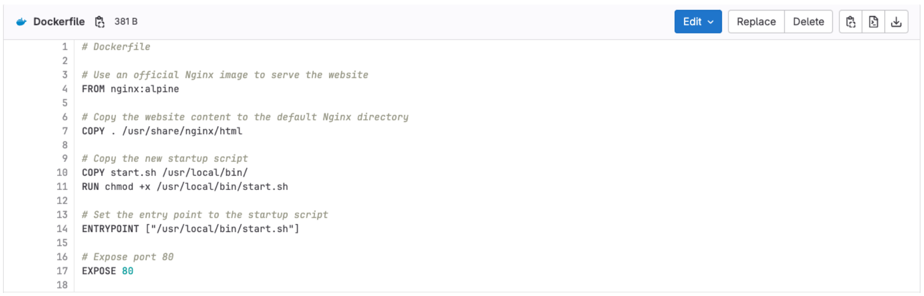
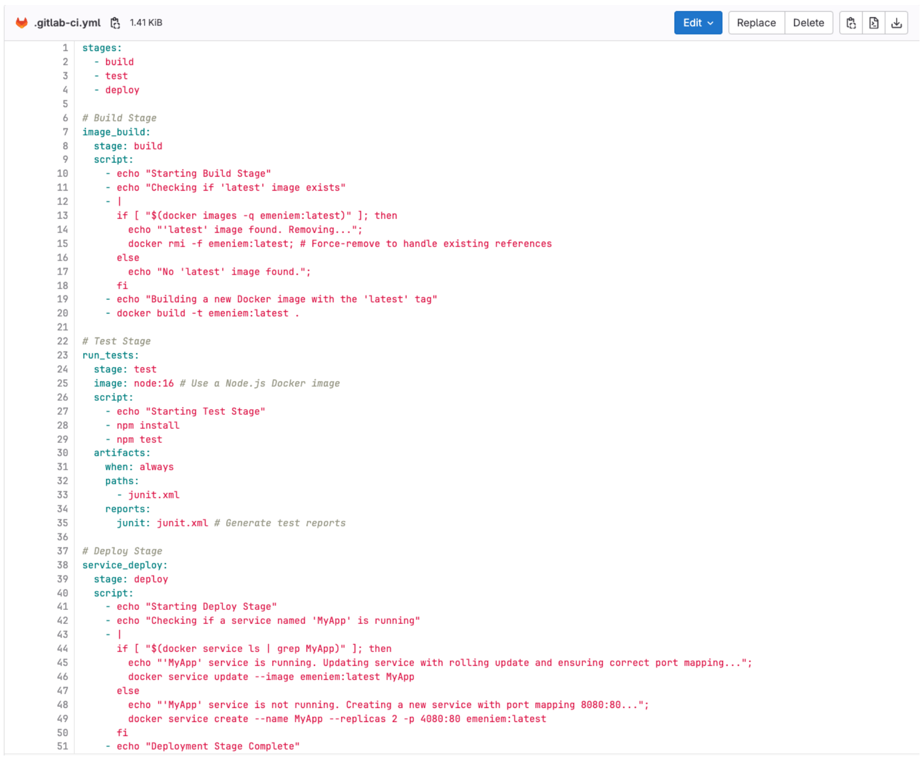


Figure 6: Feature Flag in Action – The 'Buy Tickets' button is visible based on the enabled status of the 'buy-ticket-feature' flag, demonstrating dynamic control of user interface elements.



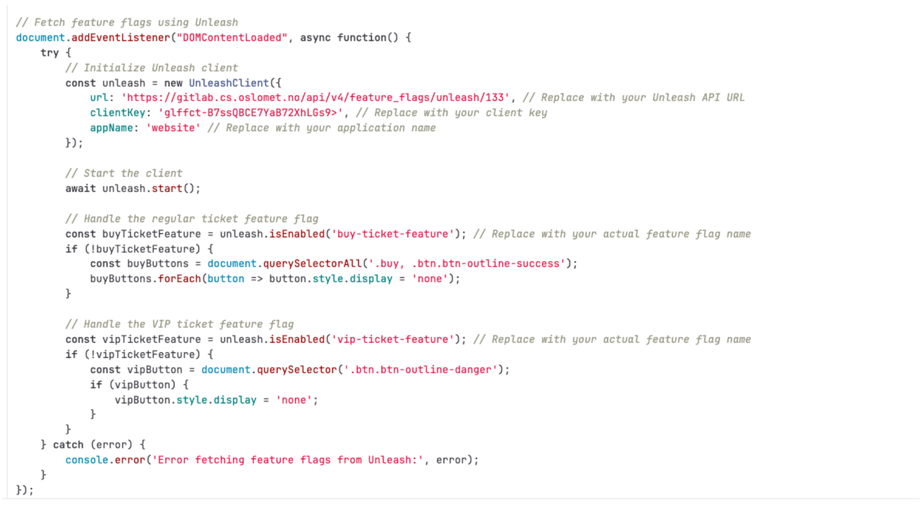
```
# Dockerfile

# Use an official Nginx image to serve the website
FROM nginx:alpine

# Copy the website content to the default Nginx directory
COPY . /usr/share/nginx/html

# Copy the new startup script
COPY start.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/start.sh

# Set the entry point to the startup script
ENTRYPOINT ["/usr/local/bin/start.sh"]

# Expose port 80
EXPOSE 80

```

Figure 7: Dockerfile setup for deploying Canario's application using an Nginx server. This file ensures consistency across environments during the deployment process.



```yaml
stages:
  - build
  - test
  - deploy

# Build Stage
image_build:
  stage: build
  script:
    - echo "Starting Build Stage"
    - echo "Checking if 'latest' image exists"
    - |
      if [ "$(docker images -q emeniem:latest)" ]; then
        echo "'latest' image found. Removing...";
        docker rmi -f emeniem:latest; # Force-remove to handle existing references
      else
        echo "No 'latest' image found.";
      fi
    - echo "Building a new Docker image with the 'latest' tag"
    - docker build -t emeniem:latest .

# Test Stage
run_tests:
  stage: test
  image: node:16 # Use a Node.js Docker image
  script:
    - echo "Starting Test Stage"
    - npm install
    - npm test
  artifacts:
    when: always
    paths:
      - junit.xml
    reports:
      junit: junit.xml # Generate test reports

# Deploy Stage
service_deploy:
  stage: deploy
  script:
    - echo "Starting Deploy Stage"
    - echo "Checking if a service named 'MyApp' is running"
    - |
      if [ "$(docker service ls | grep MyApp)" ]; then
        echo "'MyApp' service is running. Updating service with rolling update and ensuring correct port mapping...";
        docker service update --image emeniem:latest MyApp
      else
        echo "'MyApp' service is not running. Creating a new service with port mapping 8080:80...";
        docker service create --name MyApp --replicas 2 -p 4080:80 emeniem:latest
      fi
    - echo "Deployment Stage Complete"
```

Figure 8: GitLab CI/CD configuration showcasing the build, test, and deploy stages. This setup automates deployment workflows to improve reliability and efficiency.



```javascript
// Fetch feature flags using Unleash
document.addEventListener("DOMContentLoaded", async function() {
    try {
        // Initialize Unleash client
        const unleash = new UnleashClient({
            url: 'https://gitlab.cs.oslomet.no/api/v4/feature_flags/unleash/133', // Replace with your Unleash API URL
            clientKey: 'glffct-B7ssQBCE7YaB72XhLGs9>', // Replace with your client key
            appName: 'website' // Replace with your application name
        });

        // Start the client
        await unleash.start();

        // Handle the regular ticket feature flag
        const buyTicketFeature = unleash.isEnabled('buy-ticket-feature'); // Replace with your actual feature flag name
        if (!buyTicketFeature) {
            const buyButtons = document.querySelectorAll('.buy, .btn.btn-outline-success');
            buyButtons.forEach(button => button.style.display = 'none');
        }

        // Handle the VIP ticket feature flag
        const vipTicketFeature = unleash.isEnabled('vip-ticket-feature'); // Replace with your actual feature flag name
        if (!vipTicketFeature) {
            const vipButton = document.querySelector('.btn.btn-outline-danger');
            if (vipButton) {
                vipButton.style.display = 'none';
            }
        }
    } catch (error) {
        console.error('Error fetching feature flags from Unleash:', error);
    }
});
```

Figure 9: JavaScript implementation for fetching and applying feature flags using Unleash. This enables dynamic control over website features based on runtime configurations.