**DevOps Overview**

**Technology used**

**AWS**

Amazon Web Services is a leading cloud platform offering over 200 services like computing, storage, databases, and machine learning. It enables scalable, secure, and cost-effective solutions for individuals, businesses, and organizations worldwide.

**Docker**

Docker is a platform that enables developers to build, package, and run applications in lightweight, portable containers. These containers include all necessary dependencies, ensuring consistent performance across different environments.

**Kubernetes**

Kubernetes is an open-source platform designed for automating the deployment, scaling, and management of containerized applications. It provides a robust and scalable architecture for managing containerized workloads and services, ensuring high availability and efficient resource utilization. Kubernetes supports various container runtimes, like Docker and containerd, and can run on any cloud provider or on-premises infrastructure.

We use Kubernetes to manage and deploy applications in a highly available and scalable manner, ensuring optimal performance. Kubernetes' extensive ecosystem, including tools like Helm for package management, makes it easier to deploy and manage complex applications.

Alerts from Kubernetes clusters, integrated with monitoring tools like Grafana and Loki, notify teams of any critical issues, ensuring a rapid response to application and infrastructure problems.

**ArgoCD**

ArgoCD is a declarative, GitOps-based continuous delivery tool for Kubernetes. It automates application deployment and lifecycle management by syncing Kubernetes clusters with configurations stored in Git repositories.

**Helm charts**

Helm charts are packages that define, configure, and deploy applications on Kubernetes. They simplify application management by bundling YAML templates and values for repeatable, scalable deployments.

**GitLab**

**GitLab** is a web-based DevOps platform that provides version control, CI/CD, and project management tools. It enables teams to collaborate efficiently, automate workflows, and manage code repositories securely.

**Branching Strategy in Repositories**

Our GitLab repositories use a structured three-branch strategy to manage application deployment across different environments. This ensures a smooth and controlled workflow for development, testing, and production releases.

**1. Staging Branch**

Used for deploying and testing the beta version of the application.

**2. Preprod Branch**

Acts as a staging area for the production-ready version of the application.

This branch mirrors the production environment as closely as possible to validate application behavior

**3. Prod Branch**

The final branch for deploying the production application.

Only thoroughly tested and approved changes are merged here.

**GitLab Auto Deployments**

GitLab Auto Deployments streamline the process of deploying applications by integrating Continuous Integration/Continuous Deployment (CI/CD) pipelines. This approach automates the deployment process, ensuring consistency, reducing manual intervention, and accelerating delivery cycles.

**Key Features of GitLab Auto Deployments**

**Automation:** Eliminates manual deployment steps by automating the build, test, and deployment processes.

**Consistency:** Ensures identical deployments across environments using pipeline templates and configurations.

**Rollback:** Supports easy rollback to previous versions in case of deployment issues.

**Integration:** Works seamlessly with Kubernetes, Docker, and other deployment platforms.

**Monitoring:** Provides built-in tools to monitor deployment status and logs for debugging.

**Slack Alerts Integration for Staging Deployments**

Slack alerts integration in GitLab enables teams to stay informed about the status of staging deployments by sending automated notifications to a designated Slack channel. This provides real-time updates on build statuses, deployment progress, and failures, ensuring all team members are promptly notified of staging deployments and their outcomes, thereby enabling faster response times to deployment issues.

**SonarQube**

SonarQube is an open-source platform for continuous inspection of code quality. It provides static analysis for various programming languages, helping teams identify bugs, vulnerabilities, and code smells. SonarQube integrates seamlessly into CI/CD pipelines, enabling automated code reviews and promoting a culture of code quality.

In our setup, SonarQube is used to analyze code quality across different environments, providing real-time feedback to developers.SonarQube’s integration with CI/CD pipelines ensures that code quality checks are automatically performed with every commit or pull request

SonarQube’s alerting and reporting features provide valuable insights into potential issues within the code. These insights help teams prioritize and address issues before they impact production.

**Prometheus**

Prometheus is an open-source monitoring and alerting toolkit designed for collecting and querying metrics from applications and infrastructure. It uses a powerful query language and is widely used with Grafana for visualization.

**Grafana**

Grafana is an open-source platform for real-time monitoring and visualization of metrics, logs, and traces. It supports various data sources like Prometheus and Elasticsearch, allowing teams to create interactive and customizable dashboards for centralized monitoring across environments.

In our setup, we use separate dashboards for each application to ensure focused and application-specific monitoring. This approach helps teams track performance, identify issues, and analyze trends effectively. Additionally, Grafana's alerting capabilities provide timely notifications for critical issues specific to each application.

**Loki, integrated with Grafana**, is a log aggregation system that stores and queries logs efficiently. Unlike traditional log management tools, Loki indexes only metadata, making it a lightweight and cost-effective solution. It works seamlessly with Grafana, enabling teams to correlate logs with metrics and traces, providing deeper insights into application behavior.

In our setup, Loki is used alongside Grafana for centralized logging, ensuring efficient log management and enabling teams to troubleshoot and resolve issues faster. Alerts from Loki, integrated with Grafana, help notify teams of any critical log events, ensuring a timely response.

**DataBases**

**MySQL**

MySQL is an open-source relational database management system (RDBMS) known for its speed and reliability. It supports SQL-based querying and is widely used in web applications.

**PostgreSQL**

PostgreSQL is a powerful, open-source object-relational database system that supports advanced data types and SQL features. It's known for its extensibility, standards compliance, and performance.

**Metabase**

Metabase is an open-source business intelligence (BI) tool that allows users to create dashboards, visualizations, and reports without writing complex queries. It connects to various databases for easy data exploration.

**Clickhouse**

Clickhouse is a fast, open-source columnar database optimized for real-time analytics and large-scale data processing. It supports high-speed querying and is ideal for big data workloads.

**SyllaDB**

ScyllaDB is a high-performance NoSQL database compatible with Apache Cassandra. It offers low-latency and high-throughput operations, making it suitable for large-scale distributed systems.