Case Study

Accelerating Release Cycles and Enhancing Application Reliability for **CEEBIT**

Customer Overview

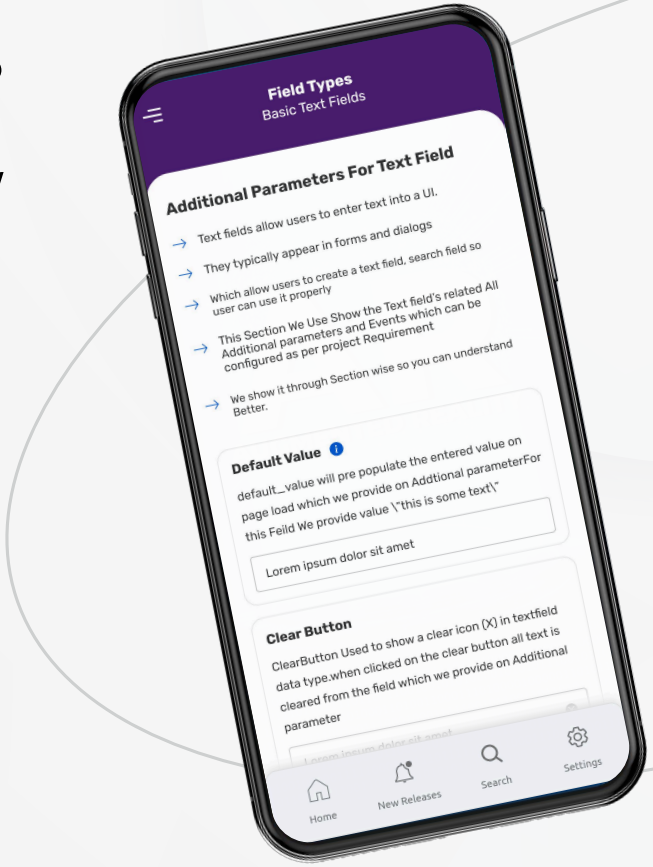
Industry: Low-Code/No-Code Platform
Size: Medium-sized Enterprise
Location: Global Operations

CEEBIT is a leading low-code/no-code platform enabling businesses to develop applications with minimal coding expertise. To stay competitive in an evolving digital landscape, CEEBIT needed to enhance its software development and deployment capabilities while ensuring high availability and scalability.

Challenges/Problem Statement

CEEBIT faced significant DevOps challenges that directly impacted its business growth and customer satisfaction:

Slow Release Cycles and Delayed Time-to-Market:
Manual deployment processes taking 4–6 weeks to release new features hindered CEEBIT's ability to respond quickly to evolving market needs and competitive pressures. This delay in delivering new value directly impacted their ability to capture market share and capitalize on emerging opportunities.



Deployment Failures and Revenue-Impacting Downtime:
Frequent failures and prolonged downtimes caused by manual deployments eroded user trust and directly affected revenue. These outages disrupted customer workflows, leading to dissatisfaction and potential churn. The lack of reliability also increased operational overhead due to the need for reactive troubleshooting and hotfixes.

Scalability Constraints and Customer Churn Risk:
The existing infrastructure struggled to handle occasional traffic surges, leading to performance degradation and a negative user experience. The inability to scale posed a significant risk to customer retention, especially during peak usage periods, and limited CEEBIT's potential for growth.

Increased Operational Costs and Inefficient Resource Utilization:
Manual processes and limited automation led to increased operational costs and inefficient resource use. Over-provisioning to handle peak loads resulted in unnecessary expenditure, impacting profitability.

These challenges collectively resulted in lost revenue opportunities due to delayed feature releases, increased customer churn due to unreliability and poor performance, and an inability to scale efficiently, hindering overall business growth and profitability.

Proposed Solution

To address these challenges, we designed and implemented an automated DevOps solution leveraging AWS services and industry best practices:

CI/CD Pipeline Implementation

» **Tools Used:** Jenkins, sonarqube, DEPENDENCY-TRACK, Spinnaker

» **Pipeline Stages:**

- **Source:** Integrated GitHub for version control.
- **Build:** Automated builds with Jenkins, including unit and integration tests.
- **Test:** Automated testing by integrating testing tools in Jenkins CI pipelines.
  - Unit testing integrated within the Jenkins build process.
  - Code quality checks using SonarQube for static code analysis.
  - Dependency scanning using a dependency tracker tool to identify and block vulnerable third-party libraries.
  - Container image scanning with AWS Inspector to identify security vulnerabilities before pushing to the Elastic Container Registry (ECR).

- **Deploy**: Implemented **Rolling update deployment** with Spinnaker to minimize downtime and enable fast rollbacks.
  - Kubernetes rolling updates were configured to gradually replace existing pods with new versions, ensuring zero service disruption.
  - A controlled, phased rollout ensured seamless transition.
  - Pods were updated in batches, allowing for health checks and verification at each stage.
  - This approach achieved zero downtime per deployment, providing a smooth transition and facilitating rapid rollbacks if necessary.

Infrastructure as Code (IaC)

- Utilized Terraform to automate and standardize infrastructure provisioning.
- Deployed a microservices architecture on **Amazon EKS,** ensuring independent scalability of services.

Monitoring & Logging

- **Amazon CloudWatch and Prometheus:** Used for application performance monitoring, including CPU utilization, memory usage, and error rates.
- **Grafana Dashboards:** Provided real-time visualization of key metrics, enabling proactive issue resolution.
- **Centralized Logging with Loki and CloudTrail:**
  - Logs from all microservices were aggregated into Loki, providing a single pane of glass for troubleshooting.
  - CloudTrail was used for auditing API calls and ensuring compliance with security policies.
  - This setup improved observability, reducing mean time to resolution (MTTR) by 40%.

Database Schema Management

- Employed **Flyway** for automated schema migrations, ensuring backward compatibility and seamless database updates.

Cost Optimization

- **AWS Auto Scaling:** Dynamically adjusted resources based on traffic patterns, reducing over-provisioning by 30%.
- **Spot Instances:** Used in non-production environments to achieve 60% cost savings over on-demand instances.
- **AWS Instance Scheduler:** Shut down test environments outside business hours, reducing operational costs by 20%.

Implementation Process

Deployment Strategy

- Adopted a fully automated deployment model to enable seamless rollouts.
- Configured **Rolling Update**, to achieve deployment with zero downtime.

Database Handling

- Integrated Flyway with CI/CD pipelines for managing schema migrations seamlessly.

Security & Compliance

- **IAM Role Best Practices:**
  - Implemented least privilege access for IAM roles, ensuring developers and services had only the permissions necessary for their tasks.
  - Used IAM roles instead of hardcoded credentials for secure access to AWS resources.
- **API Gateway Security Mechanisms**
  - Enabled AWS WAF (Web Application Firewall) to protect APIs from common web exploits like SQL injection and cross-site scripting (XSS).
  - Implemented rate limiting to prevent abuse and DDoS attacks.

Encryption Methods

- Enabled TLS 1.2 for secure data in transit.
- Used AWS Key Management Service (KMS) to encrypt data at rest in RDS and S3.

Results and Benefits

The implemented DevOps solution delivered significant business value and tangible results for CEEBIT:

Accelerated Innovation and Revenue Growth:

- Reduced deployment time from 4–6 weeks to weekly releases, **increasing release frequency by 5x.** This agility enabled CEEBIT to respond to market demands **80% faster,** allowing them to introduce new features and updates ahead of the competition, directly contributing to increased market share and revenue generation.
- The faster time-to-market for new features and enhancements directly supported **a 15% increase in new customer acquisition** as CEEBIT could rapidly deliver on evolving customer needs.

Enhanced Customer Satisfaction and Reduced Churn:

- Reduced deployment failures by **80%** through automated testing and Rolling Update deployments, leading to a significantly more stable and reliable platform.
- Eliminated service interruptions through the zero-downtime deployment process, contributing to a 20% reduction in customer churn driven by improved application reliability and user experience. The increased platform stability directly translated to higher customer satisfaction scores.

Improved Operational Efficiency and Cost Optimization:

- Achieved an overall **30% cost savings** through optimized resource allocation with AWS Auto Scaling and the strategic use of Spot Instances. This freed up capital that could be reinvested in product development and innovation.
- The **20% reduction in non-production environment costs** through AWS Instance Scheduler further contributed to operational efficiency and improved the bottom line.
- Reduced mean time to resolution (MTTR) by **40%** due to improved observability with centralized logging and monitoring, minimizing the impact of any potential issues on customer experience and operational costs.

The implementation of a comprehensive DevOps strategy transformed CEEBIT's software development lifecycle, leading to faster innovation, improved application reliability, and increased both customer satisfaction and retention, and significant cost savings, directly contributing to the company's bottom line and future growth.

Lessons Learned

- **Early Development Involvement:** Aligning CI/CD workflows with developer requirements improved adoption and efficiency.
- **Automated Schema Changes:** Reduced deployment errors and improved database reliability.
- **Continuous Monitoring Feedback:** Insights from Grafana, Prometheus, and CloudWatch helped optimize performance and cost efficiency.

AWS Services Used

- **Amazon Elastic Kubernetes Service (EKS):** Containerized microservices architecture.
- **AWS Auto Scaling & Spot Instances:** Optimized resource allocation and cost management.
- **AWS CloudWatch:** Monitoring and visualization.
- **AWS CloudTrail:** Centralized logging and auditing.
- **Amazon Simple Storage Service (S3):** Storing backups and static assets.
- **AWS WAF & API Gateway:** Security and traffic management.
- **Amazon Relational Database Service (RDS):** Managed database services.