Big Bang Case Study

Assignment #1: CSDO 1020 – DevOps CICD Pipeline Modernization

Submitted June 11, 2023

Members:

1. Yael Karmani (220760047)
2. Gary (Lyndon) Reid (220546388)
3. Sotola Olusegun (220275715)
4. Vasudeo Rane
5. Sopuruchi Owen (220143962)

Table of Contents

Big Bang Background

Big Bang is a global e-commerce web-based application that has experienced significant growth and success. It started as a small venture in the founder's garage in Metro Vancouver, with only a few servers and applications. Over time, it has expanded into a company with several hundred servers and applications hosted in a co-located data centre.

The company's success can be attributed to its focus on providing excellent customer service and minimizing downtimes. However, with the exponential growth and unexpected popularity, Big Bang has faced challenges in automating and rolling out new continuous deployment capabilities to update applications quickly.

Company Goals:

- Adopt Agile, DevOps, and SRE principles and practices.
- Increase autonomy in development and operations.
- Modernise legacy systems.
- Enable data-driven decision-making using Machine Learning and Artificial Intelligence.
- Provide best-in-class services to customers.
- Support analytics capabilities.
- Improve speed and reliability of development and operations.
- Achieve cost management and reduction.

Assumptions

- **Cloud Adoption:** Big Bang is in the process of moving some of its on-premises systems to the public cloud. The exact cloud provider and the extent of their cloud adoption are not specified. They are transitioning certain workloads to the cloud and the DevOps CI/CD pipeline should support hybrid/multi-cloud deployments.

- **Technology Stack:** The existing technical environment at Big Bang includes Linux distributions, Apache Subversion, and the Hudson legacy tool. However, specific details about the versions and specific technologies used are not provided. They would use popular open-source technologies commonly used in web application development.

- **Development and Testing Environments:** The case study mentions separate development and testing environments but doesn't provide details about the specific tools and technologies used. They will follow common development and testing practices, including IDEs, unit testing frameworks, and integration testing methodologies.

- **Microservices Architecture:** Big Bang has numerous non-containerized web applications and microservices-based APIs running on various application servers. However, the case study lacks details about the architecture patterns, communication protocols, and service discovery mechanisms. They follow a microservices architecture using RESTful APIs, potentially utilising service mesh technology for service-to-service communication.

- **Security**: Big Bang requires secure keys and secrets management. Although specific security requirements are not mentioned, We assumed that they follow industry best practices for security, including secure storage and transmission of sensitive information.

Improvement Opportunities

- Manual infrastructure provisioning and patch management: Implement automated infrastructure provisioning and patch management using Infrastructure as Code (IaC) and configuration management tools.

- Limited scalability and agility: Adopt a cloud-native approach and leverage containerization to achieve better scalability and agility in handling increased demand and rapid feature deployment.

- Reactive monitoring and alerting: Improve monitoring practices by implementing proactive monitoring and alerting systems with real-time visibility and centralised logging for faster issue detection and resolution.

- Lack of version control and collaboration: Transition to a Git-based repository and embrace GitOps practices to enhance version control, facilitate collaboration, and enable better tracking of changes across infrastructure and application code.

- Inefficient execution of batch jobs: Optimise batch job execution by leveraging cloud-native services like serverless computing and event-driven architectures to reduce costs and enable dynamic triggering based on customer activities.

Business and Technical Goals:

Tactical Objectives:

- Automation: Automate manual processes and tasks to improve efficiency and reduce errors.
- Continuous Integration (CI): Implement CI practices for frequent code integration and early bug detection.
- Continuous Delivery (CD): Establish CD practices for frequent and reliable software releases.
- Infrastructure as Code (IaC): Use IaC principles to programmatically provision and manage infrastructure.
- Configuration Management: Implement tools and practices to ensure consistency and manage changes in different environments.

Strategic Objectives:

- Focus on the customer: Maintain a competitive advantage by focusing on the customer with best-in-class customer service and minimal downtimes.
- Analytics and real-time projections: Real-time business insights will allow Big Bang to serve the customers and pivot to market demand.
- Cost-management: Manage technology spending and minimise expenses
- Speed to Market: Increase the velocity of features delivered by adopting DevOps practices

Performance Objectives

- Time-to-Market: Reduce the time it takes to deliver new features and updates to customers.
- Customer Satisfaction: Improve customer satisfaction by delivering high-quality products with minimal defects and downtime.
- Cost Optimization: Optimise infrastructure, operations, and software development costs.
- Business Agility: Increase the ability to respond quickly and adapt to changing market demands and customer needs.
- Revenue Growth: Drive revenue growth by attracting and retaining customers through improved product offerings and timely updates.

Technical Goals and Requirements:

- Deployment Frequency: Increase the frequency of software deployments to enable faster delivery cycles.
- Lead Time: Reduce the lead time from code commit to production deployment to shorten the development lifecycle.
- Mean Time to Recovery (MTTR): Minimise the time it takes to recover from failures and restore service availability.
- Scalability: Ensure the ability to scale the infrastructure and applications to handle growing user demands.
- Reliability: Improve the reliability and stability of the system to minimise downtime and disruptions.

Priorities

Business and Technical Goals:	Business Requirements:	Performance Objectives:
Revenue Growth	Adopt Agile, DevOps, and SRE principles	Deployment Frequency
Customer Satisfaction	Decrease infrastructure costs	Lead Time
Time-to-Market	Achieve cost reduction targets	Mean Time to Recovery (MTTR)
Cost Optimization	Ensure high availability for customer-facing systems	Scalability
Business Agility	Provide financial management for containers	Reliability
	Increase automation and minimise complexity	
	Improve development workflow speed and reliability	
	Enhance visibility and proactive system performance management	
	Enable analytics for cross-selling trends and predictions	

Agile, DevOps, and Site Reliability Engineering (SRE)

Agile, DevOps, and Site Reliability will principles and practices will enable Big Bang to accelerate the rate at which we release high-quality software to the marketplace. Our investment in Site Reliability Engineering will ensure that our products are stable and meet our Service Level Agreements of 99.99%.

Agile:

At Big Bang, we will take an agile, iterative approach to software development. This will ensure we are getting feedback early and often from our customers. Our emphasis on early feedback will ensure that we deliver features that matter to our customers and ultimately result in better organisational performance.

DevOps:

Our focus on DevOps practices, tools, and cultural philosophies will unlock our ability to deliver quickly. Developers, operations, and security will work together as ONE team.

DevOps combines cultural philosophies, practices, and tools that increase an organization's ability to deliver applications and services at high velocity.

Site Reliability Engineering:

We will apply engineering principles to manage operations and infrastructure. Our investment in SRE training and tools will ensure that we are positioned to support SLAs that target high availability and performance.

The culmination of these practices and the culture at Big Ban will ultimately enable us to deliver world-class functionality that our customers love and is highly available.

Expected Outcomes

Reduce risk
Finding and fixing bugs late in the development process is expensive and time-consuming. This is especially true when there are issues with features that have already been released to production.

With a CI/CD pipeline, you can test and deploy code more frequently, allowing testers to detect issues as soon as they occur and fix them immediately. You are essentially mitigating risks in real-time.

Deliver faster
Companies are moving toward releasing features multiple times a day. This is not an easy task; only a handful of companies like Netflix, Amazon, and Facebook have been able to achieve this goal. But, with a seamless CI/CD pipeline, multiple daily releases can be made a reality.

Teams can build, test and deploy features automatically with almost no manual intervention. This is accomplished using various tools, frameworks, and systems like Travis CI, Docker, Kubernetes, and LaunchDarkly.

Expend less manual effort
To align with the shift-left paradigm, we need automation right from the start. This is also a vital component of having a successful CI/CD implementation. Once you build features and check in code, tests should be automatically triggered to make sure that the new code does not break existing features and that the new features are working correctly.

After the tests run, the code gets deployed to different environments, including QA, staging and production. Throughout this process, you will be getting constant notifications through different channels, giving you plenty of information about the build, test and deploy cycles.

Generate extensive logs
Observability is one of the biggest aspects of DevOps and CI/CD integration. If something is wrong, you need to understand why. You need a mechanism to study the system in production over time and identify key performance metrics. Observability is a technical solution that helps in this effort.

One key aspect of observability is logging information. Logs are a rich source of information to understand what is happening beneath the UI and study application behaviour.

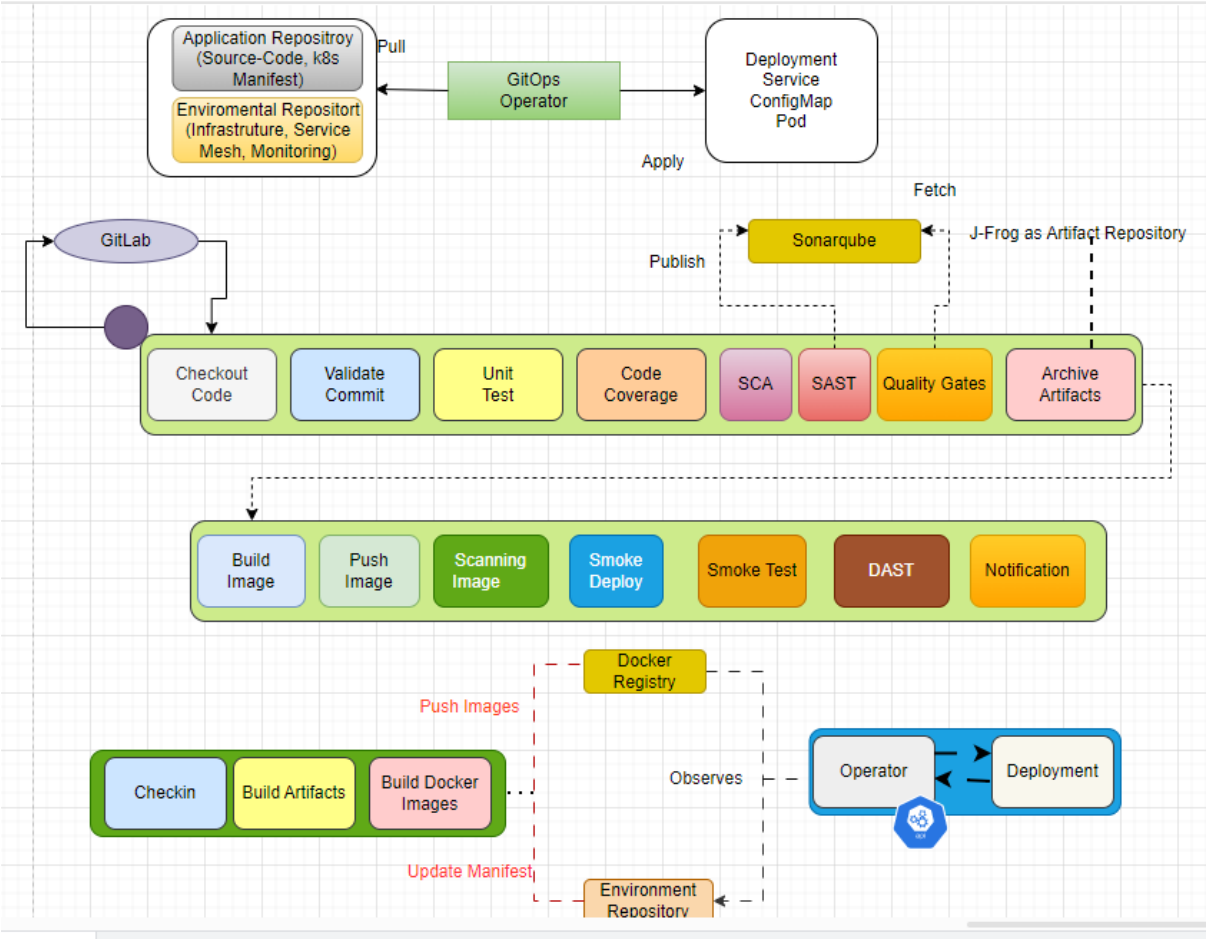
With a CI/CD pipeline, extensive logging information is generated in each stage of the development process. There are various tools available to analyse these logs effectively and get immediate feedback about the system.

Make easier rollbacks

One of the biggest advantages of a CI/CD pipeline is you can roll back changes quickly. If any new code changes break the production application, you can immediately return the application to its previous state. Usually, the last successful build gets immediately deployed to prevent production outages.

DevOps

CI/CD pipeline Modernization Solution Overview



Big Bang will adopt OSS tools to optimise our CICD pipeline. A number of new tools will be implemented. Of course, tooling is essential, but adopting the DevOps culture and ways of working is as important to our success.

An overview of the proposed tools and a brief overview of the purpose follows

Source Control Management:

A git-based source control tool such as gitlab will be used. This will ensure our code is always versioned and will ensure we are using widely supported tools.

Orchestration

Jenkins X will be used as our orchestration tool. This will allow us to adopt a multi-cloud implementation when we are ready.

Build

Maven or Gradle will be used to build software into compiled code and create the artifacts.

Quality Gates

A number of tools will be used to ensure code quality including:.

- **Sonar Qube** will be used for Static Code Analysis to ensure the quality of code being released meets our standards.
- **ZAP** will be used for security scanning to ensure no new vulnerabilities are introduced with features.

Testing

A number of open-source tools will be used to conduct testing. Including:

- **JUnit** - Developers will use Test Driven Development and write unit tests
- **Postman** - Postman will be used to test API Endpoints
- **Selenium** - Selenium will be used for Automated UI Tests
- **Jmeter** - Jmeter will be used for non-functional performance tests

Artifact Storage

Nexus Repository Manager stores artifacts and containers.

Release Orchestration

Ansible will be used for configuration management, and Docker and Kubernetes will be used to containerize applications and ensure that we can deploy to a multi-cloud environment when our organisation is ready.

Monitoring & Logging

While not technically part of the CICD pipeline, we want to ensure it's understood that applications will complement Prometheus and Grafana for monitoring, alerting, and visualisation.