**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
4. Vasudeo Rane
5. Sopuruchi Owen

# Table of Contents

[Table of Contents 2](#_d881bi52dk6m)

[Big Bang Background 3](#_fnylvuj38n4e)

[Assumptions 4](#_rkex2asgy00m)

[Improvement Opportunities 5](#_5q31oawznswz)

[Business and Technical Goals: 6](#_aiiyv7g9db80)

[Performance Objectives 7](#_15zdlbwfyv7f)

[Priorities 8](#_1k316jax8z3m)

[Agile, DevOps, and Site Reliability Engineering (SRE) 9](#_l6dp8n9329x1)

[DevOps: 10](#_4k0pu1xpltlf)

[Expected Outcomes 11](#_7hb3o1eemd3x)

[CI/CD pipeline Modernization Solution Overview 13](#_nh37emf1luyy)

[Source Control Management: 13](#_7el2bn5w25wg)

[Orchestration 13](#_4yf5btvsyq8z)

[Build 13](#_a5tqop4ue4d4)

[Quality Gates 14](#_jopysvmj2426)

[Testing 14](#_jjiz57hi5e3o)

[Artifact Storage 14](#_3h47rm51zx5n)

[Release Orchestration 14](#_z587qhz195q)

[Monitoring & Logging 14](#_dtdk2wdxfhyh)

# 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 the Metro Vancouver area, 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 minimising 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:

1. Adopt Agile, DevOps, and SRE principles and practices.

2. Increase autonomy in development and operations.

3. Modernise legacy systems.

4. Enable data-driven decision-making using Machine Learning and Artificial Intelligence.

5. Provide best-in-class services to customers.

6. Support analytics capabilities.

7. Improve speed and reliability of development and operations.

8. Achieve cost management and reduction.

# Assumptions

1. Cloud Adoption: Big Bang is in the process of moving some of their 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 that the DevOps CI/CD pipeline should support hybrid/multi-cloud deployments.
2. 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.

1. 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.

1. 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.

1. 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

1. Manual infrastructure provisioning and patch management: Implement automated infrastructure provisioning and patch management using Infrastructure as Code (IaC) and configuration management tools.
2. 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.
3. 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.
4. 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.
5. 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:**

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

**Strategic Objectives:**

1. Focus on the customer: Maintain competitive advantage by focusing on the customer with best in class customer service and minimal downtimes.
2. Analytics and real-time projections: Real-time business insights will allow Big Bang to serve the customers and pivot to market demand.
3. Cost-management: Manage technology spend and minimise expense
4. 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 costs associated with infrastructure, operations, and software development.
* 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 Engineering (SRE) are the building block for all modern software development and operations. These methodologies improve efficiency, collaboration, and quality of software systems.

Agile:

Agile is an iterative approach to project management and software development. It emphasises flexibility, adaptability, and customer collaboration. There are four core principles of Agile as per its manifesto.

1. Individuals and interactions over processes and tools - Close coordination and communication is the key for Agile development.Agile ceremonies play a key role to break the silos.
2. Working software over comprehensive documentation - Each phase of agile delivery is supported by show and tale hence emphasis is on a workable product, document is also critical but it is limited to what is necessary.
3. Customer collaboration over contract negotiation - Product owner and development team interact day to day, this was to provide the product as per expectation and further it is enhanced to satisfy the customer need rather than following a strict contract.
4. Responding to change over following a plan - The core concept of Agile is being able to accept stories as per product priority.

The two common practices followed in Agile are :-:

· Scrum: A popular Agile framework that defines roles, events, and artifacts in a project. It includes daily stand-up meetings, sprint planning, and sprint reviews.

· Kanban: A visual workflow management system that helps teams manage and optimize the flow of work, limiting work in progress (WIP).

# DevOps:

DevOps is the combination of cultural philosophies, practices, and tools that increases an organisation’s ability to deliver applications and services at high velocity.

DevOps best practices are :-

1. Agile project management:- Agile méthodologies can help ease the implementation of devOps best practices, Requirements, plans, and results are evaluated continuously, allowing teams to respond to feedback and pivot as necessary.

2. ​​​​​​​Shift left with CI/CD – The early the testing is done along the code delivery reduces the cost of fixing them at the later stage. Feedbacks are received early in the development cycle.

​​​​3. ​​​​​​​Implement automation – Automation is the core for DevOps implementation. Automated build, test and integration helps the team to deliver the product faster.

4. Monitor the DevOps pipeline and applications – It is important to monitor production applications in order to identify failures or performance deficiencies.

5. ​​​​​​​Observability – Logs, traces and metrics need to be captured and aggregated to make discoveries and predictions about the functioning of the application.

6. ​​​​​​​Gather continuous feedback - Continuous feedback ensures team members have all the information needed to do their jobs on a timely basis

7. ​​​​​​​Change the culture – DevOps is all about developing the culture, it requires collaboration, transparency, trust, and empathy.

SRE (Site Reliability Engineering):

Site reliability engineering (SRE) is the practice of applying software engineering principles to operations and infrastructure processes to help organisations create highly reliable and scalable software systems.

. Key principles of SRE include:

* Embrace Risk - Improving system reliability and customer satisfaction comes at a risk and it needs to be properly weighted. Embrace risk allows to make provision for cost management.
* Service-level objectives (SLOs): SLO helps to manage risk and error budget for error. SLO should be monitored . SLI needs to be determined and it helps determine the customer pain point.
* Eliminate Toil - SRE advocates that the amount of repetitive work needs to be reduced. This helps building the team morale.
* Error budgets: Allocating a margin of error within the SLOs to balance reliability and innovation. This enables teams to focus on system improvements without compromising stability.
* Automation: Automation reduces the manual intervention and can be applied in different area of development like testing, deployment, incident response and communication.
* Simplicity - SRE advocates to develop the least complex system and thus increase the reliability of the system.

Key practices within SRE include:

* Monitoring and Alerting ensure the availability, performance, and reliability of software systems by providing visibility into their behaviour and promptly notifying teams about any anomalies or potential issues
* Incident Management: Implementing well-defined processes for handling incidents, including incident response, communication, and postmortem analysis.

# 

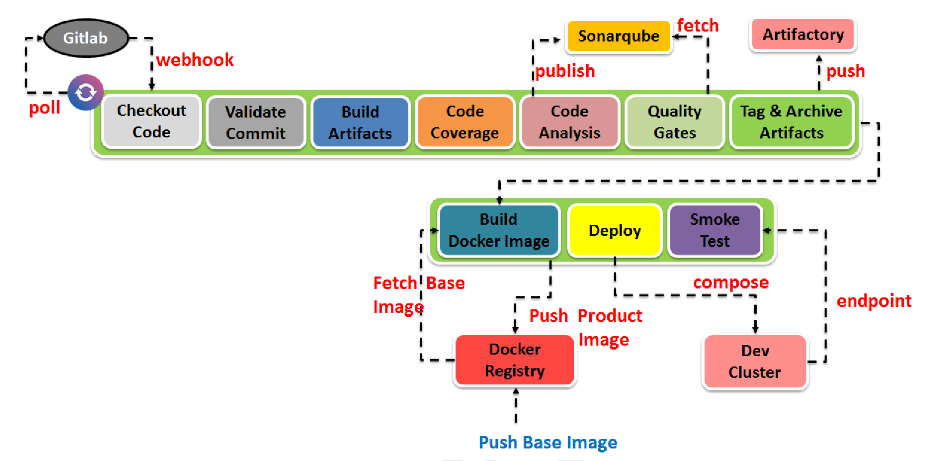
# Expected Outcomes

Adopting DevOps, Site Reliability Engineering (SRE) practices, and a robust CI/CD pipeline can bring several anticipated outcomes that support the business case. Here are some of the key anticipated outcomes:

1. Faster Time-to-Market: By implementing CI/CD practices, the development and deployment cycles are automated and streamlined. This results in faster delivery of new features, bug fixes, and enhancements to customers, reducing time-to-market and increasing competitiveness in the market.
2. Improved Software Quality: Continuous integration ensures that code changes are regularly integrated and tested, catching issues early in the development process. Automated testing in the pipeline, such as unit tests and integration tests, helps identify bugs and regressions, leading to higher software quality and fewer production issues.
3. Increased Efficiency and Productivity: DevOps practices encourage collaboration and communication between development, operations, and other teams involved in the software development lifecycle. This collaboration, combined with the automation and standardization provided by CI/CD, leads to improved efficiency and productivity across the entire development process.
4. Enhanced Reliability and Stability: Site Reliability Engineering (SRE) practices focus on building resilient and stable systems. By implementing SRE principles, such as monitoring, alerting, incident response, and capacity planning, the reliability of applications and infrastructure is improved. This reduces downtime, improves system availability, and enhances the overall customer experience.
5. Rapid and Safe Deployments: The CI/CD pipeline allows for automated deployments, reducing the risk of human error and providing a repeatable and consistent process for deploying applications. This enables faster, more frequent deployments with minimal disruption to users, ensuring that new features and fixes can be delivered to production quickly and safely.
6. Scalability and Flexibility: The automation and infrastructure-as-code principles inherent in CI/CD and DevOps practices allow for scalability and flexibility in infrastructure management. This means that applications can easily scale up or down based on demand, and infrastructure changes can be managed programmatically, reducing manual effort and increasing adaptability to changing business needs.
7. Continuous Improvement: DevOps and SRE practices promote a culture of continuous improvement. By continuously monitoring and analyzing system performance, collecting feedback, and fostering a blameless culture, teams can identify areas for improvement and implement iterative changes to enhance processes, technologies, and user experiences.

# 

# 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 very important, 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 for building software into compiled code and creating the artefacts.

### 

### 

### Quality Gates

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

* Sonar Qube Sonar Qube will be used for Static Code Analysis which will 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 test API Endpoints
* Selenium - Selenium will be used for Automated UI Tests
* Jmeter - Jmeter will be used for non-functional performance tests

### ArtifactStorage

Nexus Repository Manager store 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 that it’s understood that applications will complement Prometheus and Grafana for monitoring, alerting, and visualization.