Index

Title Page

Assignment 4
Date: June 11, 2023
Members:
Octavio Colantonio, ID#: **220792784**
Gabriel Adebowale Ogundimu, ID#:220746541
Kieran Loewen, ID#: 220411153
Justin Le, ID#: 208837221

Phase 1: Version control - Octavio

Phase 2: Infrastructure as code - Octavio

Phase 3: Configuration management - Kieran

Phase 4: Containerization and orchestration - Kieran

Phase 5: Automated builds and testing - Justin

Phase 6: Security - Justin

Phase 7: Monitoring, Observability, FinOps, and DataOps - Justin

Phase 8 : MLOps and AIOps - Gabriel

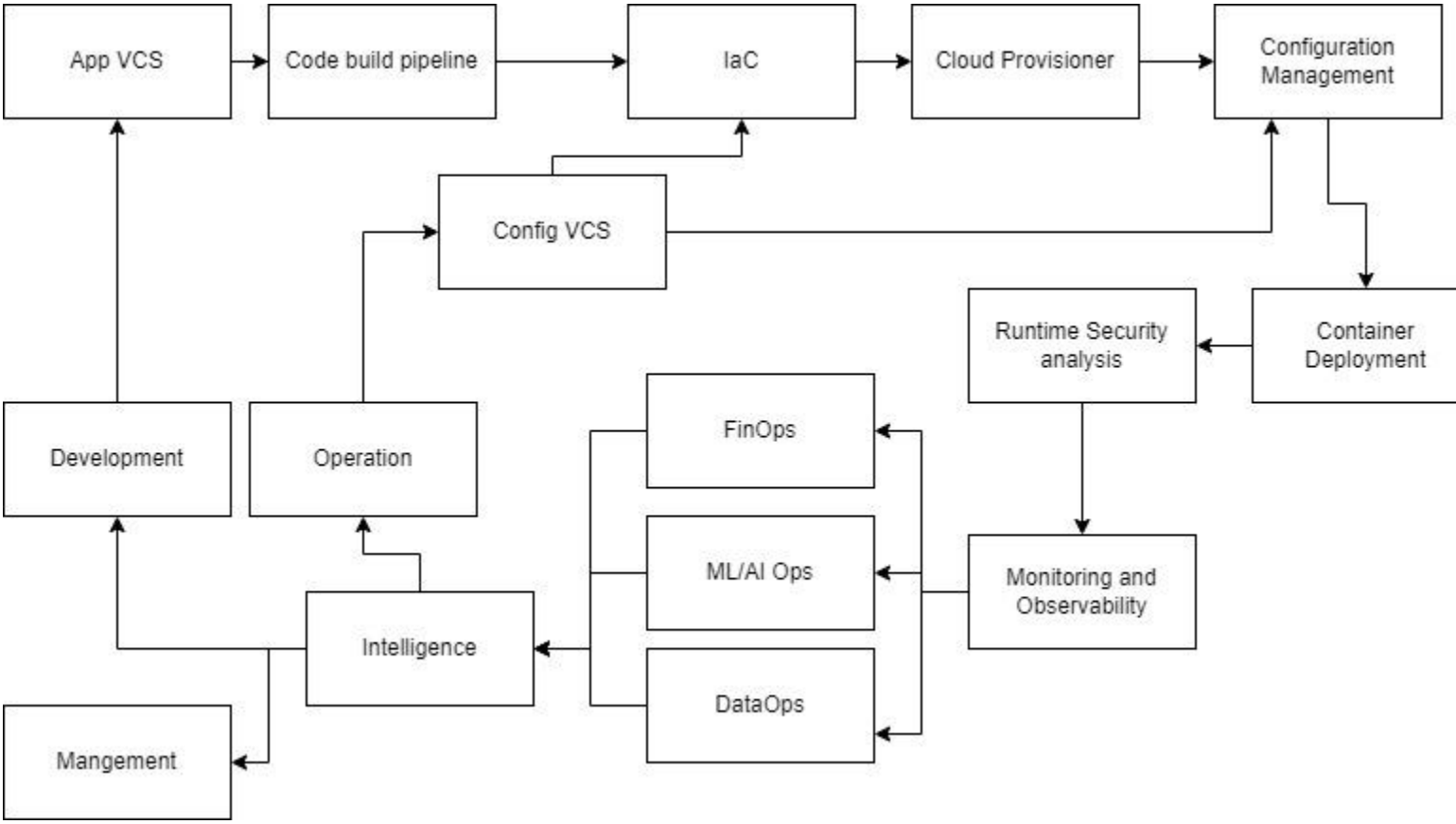
Introduction

Big Bang is a global e-commerce web-based application that helps its customers to purchase goods. Currently, Big-Bang's software system needs an upgrade from its legacy state allowing for scaling their global footprint and rolling out new continuous deployment capabilities to update their applications quickly. With an organization as large as Big Bang, modernizing the legacy system can be quite a challenge. We propose 8 phases that allow the company to incrementally make the transition to a modern system.

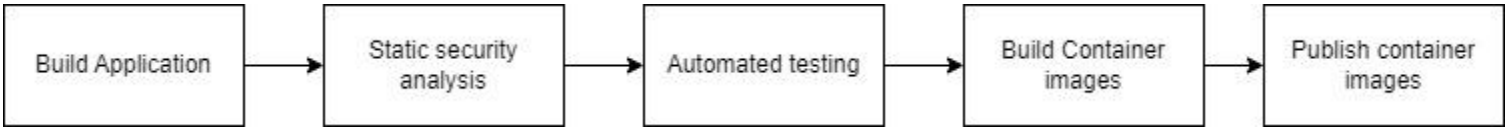
Design Overview

By using cloud native principals, and DevOps practices as our guidelines, we would like to map out a design strategy to implement a modern CI/CD pipeline into Big Bang. The goal is to provide a high level overview of the functional flows of different phases in the transitioning of the system. Further details of each phase will be explained below in the designated sections.

CI/CD main flow



Code build pipeline flow



The above diagrams explain the transitional sequences that can be converted into actionable work items. Using version control system as the main source truth to store our Application Codes as well as our Infrastructure configuration codes. In essence, everything will be code based. This will give Big Bang the flexibility to restore and migrate their platform to various environments (cloud providers, on-prem). The infrastructure will be provisioned based on the infrastructure and configure management codes (this doesn't happen often in a real environment). Once infrastructure has been provisioned, everytime an application code is committed by the development team, this will trigger the code build pipeline. Once the application container images are successfully built, they will get deployed into the infrastructure (a managed Kubernetes cluster). Further runtime security analysis can further increase the security of the application. Operations will monitor and observe the state of the system, and provide necessary data to FinOps, DataOps, ML/AIOps engineers that transform the observable data into business intelligence which can be used by the Development,

Operation, Management to further optimize their business process and provide greater values to their customers.

Phase 1 Version Control (Octavio)

Version control tools play a crucial role in enabling a CI/CD pipeline by providing a centralized repository to manage and track changes to source code and other project assets. Here are some popular version control tools that can be used for a CI/CD pipeline:

When replacing legacy system tools with Git and AWS CodeCommit, you can achieve significant benefits in terms of version control, collaboration, and automation. Here's how this transition addresses technical requirements and drives business outcomes:

1. Tools for Implementation:
   - Git: An open-source distributed version control system that offers powerful branching, merging, and collaboration capabilities.
   - AWS CodeCommit: A managed Git repository service provided by AWS, allowing you to host private Git repositories in the cloud.

2. Solving Technical Requirements:
   - Improved Version Control: Git provides a more advanced and flexible version control system compared to legacy tools. It enables efficient branching and merging, simplifies code collaboration, and allows developers to work independently on different features or bug fixes.
   - Enhanced Collaboration: Git and CodeCommit enable seamless collaboration among team members. They provide a centralized repository that supports concurrent development, simplifies code reviews, and facilitates better communication and coordination.
   - Simplified Code Management: Git's distributed nature and local repository copies make it easier to manage and synchronize code changes. It allows for offline work and reduces the risks associated with centralized repositories.
   - Secure and Scalable Hosting: AWS CodeCommit ensures secure and scalable hosting of Git repositories, with features like access control, encryption, and automated backups. It provides a reliable and robust infrastructure for storing and managing code.

3. Impact on Business Requirements:
   - Accelerated Time-to-Market: By migrating from legacy tools to Git and CodeCommit, teams can work independently on different code changes, leading to faster development cycles and shorter time-to-market for new features and bug fixes.
   - Improved Collaboration and Efficiency: Git and CodeCommit foster collaboration, knowledge sharing, and code reusability among team members. This enhances productivity and efficiency, enabling teams to deliver higher-quality software at a faster pace.
   - Streamlined CI/CD Pipelines: Integrating Git and CodeCommit into a CI/CD pipeline allows for automated builds, testing, and deployments. This automation reduces manual effort, ensures consistent releases, and improves the overall stability and reliability of the software.
   - Scalable and Resilient Infrastructure: Leveraging AWS CodeCommit provides access to AWS's scalable and resilient infrastructure. This allows for reliable hosting of repositories, even with increased team sizes and growing codebases.

By adopting Git and AWS CodeCommit as replacements for legacy system tools, organizations can unlock the benefits of modern version control, collaboration, and automation. This transition promotes faster development,

better code management, improved collaboration, and increased efficiency, ultimately driving business growth and competitive advantage.

Phase 2 Infrastructure As Code (Octavio)

Infrastructure as code plays a vital role in a CI/CD pipeline by automating the provisioning and management of infrastructure resources.

Adopting AWS CloudFormation to replace legacy system tools, organizations benefit from Infrastructure as Code, streamlined resource provisioning, improved scalability, and reduced manual effort. This results in faster deployments, improved operational efficiency, better compliance, and cost optimization, ultimately driving business agility and competitive advantage.

To replace legacy system tools with AWS CloudFormation, here's how it addresses technical requirements and drives business outcomes:

1. Tools for Implementation:
   - AWS CloudFormation: A fully managed service provided by AWS for defining and provisioning infrastructure resources as code.

2. Solving Technical Requirements:
   - Infrastructure as Code: CloudFormation allows you to define your infrastructure as code using declarative templates. This replaces manual, error-prone processes and provides a consistent and reproducible way to provision and manage infrastructure resources.
   - Resource Provisioning: CloudFormation automates the provisioning and management of resources, handling tasks such as resource creation, updates, and deletion. It ensures that the infrastructure is in the desired state and reduces the risk of manual errors.
   - Dependency Management: CloudFormation manages dependencies between resources, determining the correct order of creation and handling interdependencies automatically. This simplifies the management of complex infrastructures.
   - Scalability and Flexibility: CloudFormation templates support parameters and conditions, allowing for scalable and adaptable infrastructure configurations. You can define different configurations for different environments and easily customize deployments as needed.

3. Impact on Business Requirements:
   - Faster Time-to-Market: With CloudFormation, infrastructure provisioning is automated and repeatable. This reduces the time required for manual setup, configuration, and troubleshooting, enabling faster deployment of applications and features to the market.
   - Improved Efficiency and Reliability: CloudFormation's infrastructure automation eliminates manual effort and human error, ensuring consistent and reliable deployments. This leads to higher efficiency, improved resource utilization, and reduced risk of misconfigurations.
   - Standardization and Compliance: CloudFormation enables standardization of infrastructure configurations across environments, ensuring consistency and compliance with organizational policies and industry regulations.
   - Cost Optimization: CloudFormation provides visibility into the cost impact of infrastructure changes through template-defined resources. This helps in optimizing resource allocation, avoiding unnecessary expenses, and ensuring cost-effective infrastructure management.

Phase 3 Configuration Management (Kieran)

Configuration management is an essential DevOps tool and practice since we are always looking for ways to automate tasks and find ways to increase the stability of the application.

Ansible is one of the most common open source tools that aids in configuration management by providing a method to configure systems after they have been provisioned. Infrastructure as code such as Terraform is great for defining the specifications of a server, but is not as helpful when it comes to configuration that is done after a system is up and running. Ansible is a tool that does just this, and it is built with “Playbooks”, which are small repeatable and reusable modules that can be combined to create the complete system configuration.

Strong configuration management prevents snowflake servers and configuration drift by having a predefined set of requirements and actions that will be applied to a system. With the reusability of Ansible Playbooks it makes it very easy to create identical, or similar configurations by creating a playbook that for example installs a specific version of some software library. If you have multiple different configurations that both need that same library, just reuse the playbook which ensures that any future changes in that playbook are propagated across all the places it is needed.

Big Bang’s current infrastructure is all self hosted which itself is not necessarily bad, but it can make it harder to know what’s currently running and how it got there. Moving to the cloud is the right time to solidify a good configuration management system since they will be setting up new clusters anyway. Their current infrastructure involves a significant amount of manual work to maintain which makes it slow to react to changing needs as well as expensive.

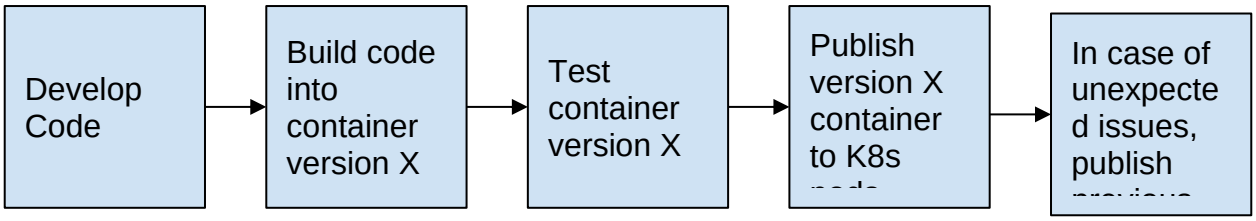
Using Ansible as a CM tool allows Big Bang to meet a number of their technical requirements:

- Use open source tools
    - Ansible is entirely Open source so it will generally work well with other open source build, CI, and deployment tools
- Increase Automation
    - Currently server setup is manually configured, so this will save a significant amount of time when new services are spun up
- Reduce operating costs
    - Automation server configuration will save many hours of manual work
    - Simpler deployment process will allow for easier scaling up or down so that only what’s actually needed is provisioned
- High uptime
    - Using Ansible with a git repository allows for review, historical changes, as well as an easy way to undo changes in the event that something does go wrong which reduces mean time to recovery

Phase 4 Containerization and Orchestration (Kieran)

Containers have revolutionized the software industry by providing an easy way to create, share, and deploy an entire bundle of an application in a portable manner. The most common open source tool used for container definition is Docker, which on its own is very useful but at a larger scale additional tools are needed to help manage the containers, which is where Kubernetes and Helm come in. Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications, and Helm is a tool that automates creation, packaging, configuration, and deployment of Kubernetes applications.

Big Bang currently has a large number of web apps and microservice-based APIs that are currently not containerized, so this is a big opportunity to switch these services to be built with Docker containers and to be managed with Kubernetes. Each web app and microservice should be built as a containerized image of everything that application needs, so that the container can be deployed easily. Containerizing an application like this makes it very easy to spin up new production environments or development/QA environments since you can create a container with a specific version of all the required dependencies that you know will always be the same no matter where and when it is deployed. This makes it easy to create a pipeline of developing, testing, and releasing a new version of an application, as well as easy reverting incase of an issue:



Utilizing Kubernetes to manage deployment of containers via K8s pods makes this as simple as updating a version number and waiting for the pods to be created with the new container version.

Combining Docker and Kubernetes will provide many benefits that will meet their desired state technical requirements:

- Use open source tools
    - Docker, Kubernetes and Helm are Open source ensuring the target is met
    - Open source projects give room to contribute back to the original project incase there is a custom need built that isn't originally supported by the tool
- Increase Automation
    - Kubernetes can automatically scale applications based on current needs
    - Helm provides tools to manage some of the otherwise boilerplate code needed to define K8s clusters
- Reduce operating costs
    - Flexibility of container deployment allows for easier scaling up or down of resources so that only what's actually needed is provisioned
- High uptime
    - Versioned containers provided repeatable deployments that are easily reverted
    - Easy scaling in existing areas or to new areas ensures that during high traffic scenarios, the application can be expanded to handle the extra load
    - Kubernetes can handle re-creating instances if they run into issues

Phase 5 CI-Automated builds and testing (Justin)

One of the main requirements is to implement a CI system in replace of the Hudson legacy tool.
We would like to perform automated builds and at the same time perform continuous testing throughout the software development life cycle by integrating a brand new CI pipeline system.

Automated builds

In the context of software development, build refers to the process that converts files and other assets under the developers' responsibility into a software product in its final or consumable form. The build may include:

- compiling source files
- packaging compiled files into compressed formats (such as jar, zip)

- producing installers
- creating or updating of database schema or data

The build is automated when these steps are repeatable, require no direct human intervention, and can be performed at any time with no information other than what is stored in the source code control repository.

Why do automated builds

One major benefit is eliminating a source of variation and thus defects; a manual build process containing a large number of necessary steps offers as many opportunities to make mistakes.

Tools

Gradle is an open-source build automation tool flexible enough to build almost any type of software. Gradle makes few assumptions about what you're trying to build or how to build it. This makes Gradle particularly flexible.

Continuous testing

Continuous testing is the process of incorporating automated feedback at different stages of the software development life cycle (SDLC) in support of better speed and efficiency when managing deployments.

Why do continuous testing

One of the fundamental principles in developing a practical DevOps approach is to bridge the gap between rapid software delivery and reliable user experiences. However, the conventional way of manually gaining feedback at each software development stage (i.e., project design, coding, testing, deployment, and maintenance) has led to an insufficient and ineffective use of organizational resources and, ultimately, longer integration cycles and delayed product updates. Continuous testing addresses these inefficiencies by helping DevOps teams "shift left," providing them with valuable feedback early in the SDLC while automating manual testing processes and minimizing human error.

Tools

Testing tools and frameworks are numerous depending on the software stack project. In terms of functional end-user test, Selenium web driver has been the goto open source tool. WebDriver is an API and protocol that defines a language-neutral interface for controlling the behavior of web browsers. As for unit, tests jest and mocha is very common for javascript, and j-unit is popular among java community

Continuous Integration

Continuous integration combines automated build process and execution automated with a CI pipeline prior to delivery/deployment.

Tools

There are a number of open source tools that can perform continuous integration. The tool we're most

interested in is Jenkins that allows Big-Bang to accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage and deployment. Jenkins works off an eco system of plugins. Each of the above mentioned tools for automated builds and testing (gradles and selenium) both have their designated Jenkins plugins.

Implementation steps

We would like to construct a Jenkins CI pipeline with each of the existing microservices. A Jenkins Pipeline can be created through the classic UI, In SCM, or through Blue Ocean UI. Through the classic UI, you can enter a basic pipeline directly in Jenkins. In Source Control Management (SCM), you can write a Jenkinsfile manually and commit it to your project's source control repository. Through Blue Ocean UI, after setting up a pipeline project in Blue Ocean, the Blue Ocean UI helps you write your Pipeline's Jenkinsfile and commit it to source control. Using Git as single source truth for each microservices source code repository, each repository will have a Jenkins build file that will execute the pipeline. The process in the pipeline will be running automated builds and verifying the correctness of the code via automated tests. Based on push based implementation, the pipeline will be triggered every time new code is committed into the repository.

Phase 6 Security (Justin)

We would like to enhance Security practices of Big Bang by introducing DevSecOp into our continuous pipeline.

DevSecOp

DevOps was born from merging the practices of development and operations, removing the silos, aligning the focus, and improving efficiency and performance of both the teams and the product. A new synergy was formed, with DevOps focused on building products and services that are easy to maintain and that automate typical operations functions.

Security is a common silo in many organizations. Security’s core focus is protecting the organization, and sometimes this means creating barriers or policies that slow down the execution of new services or products to ensure that everything is well understood and done safely and that nothing introduces unnecessary risk to the organization.

DevSecOps looks at merging the security discipline within DevOps. By enhancing or building security into the developer and/or operational role, or including a security role within the product engineering team, security naturally finds itself in the product by design.

How it will benefit Big Bang

In modern CI/CD pipeline security is paramount, with the advent of DevOps and the breaking of all silos, there is a need to incorporate security practices within the application development process. One mantra of DevOps has been ‘shift left’ everything.One place where DevOps Security lacks is in the tackling of security vulnerabilities. Typically, most security vulnerabilities are discovered at the end of a software development life

cycle and DevSecOps seeks to change that. We will incorporate DevSecOp practices into our pipeline by securing Big Bang's applications early as possible in the development cycle. This will lead to faster development and delivery time for new features.

How to build security into our pipeline

You can implement automated security checks along with manual code reviews to ensure security vulnerabilities do not make it past the reviewing stage of code production. With a continuous security philosophy, your business is less likely to run into security concerns. Code auditors should make use of static code analysis methods as well as unit tests for running checks. They do not need to execute your code. The cost of a security vulnerability is not much during the testing phase but if it goes through in production, it can lead to serious consequences. Static analyzers are recommended for security testing as they are cost-effective and are easy to implement.

Dynamic analyzer can be integrated at staging prior going to production, notably after a container images have been successfully built. These tools will scan and detect vulnerabilities at run-time.
Visualization and alerting systems can be integrated to provide quick feedback to the team if any vulnerabilities are detected during the build process.

Tools to use

1. For Visualization: Kibana and Grafana
2. For Automation: StackForm
3. For threat detection: Stackstorm
4. For testing: Gauntlt, Snyk, Chef Inspec, Kakiri, Infer, Owasp Zap, and Lynis
5. Alerting tools: Elasalert, Alerta, and 411
6. Threat intelligence tools: OpenTPX, Critical Stack, and Passive total.

Phase 7 Monitoring, Observability, FinOps and DataOps (Justin)

We would like to implement monitoring and observability tools in our pipeline. Using the data collected, we can integrate FinOps and DataOp practices that could further improve business efficiency and increase throughput on deliverables.

Monitoring

Cloud monitoring comprises a series of strategies and practices for analyzing, tracking, and managing cloud-based services and applications.

Observability

In general, observability is the extent to which you can understand the internal state or condition of a complex system based only on knowledge of its external outputs.

FinOps

FinOps is a cultural practice that brings financial accountability to variable cloud spend. It's the way for teams to manage their cloud costs, where everyone takes ownership of their cloud usage supported by a central best-practices group.

DataOps

DataOps combines Agile development, DevOps and statistical process controls and applies them to data

analytics.

How it will Benefit Big Bang

As businesses scale their infrastructure and digital footprint, it becomes vitally important for IT administrators to maintain visibility into the performance of their digital assets. Cloud monitoring provides an efficient way to achieve this visibility while providing an enterprise with actionable insights to improve availability and user experiences.

Observability tools allow the Operation team to quickly identify performance problems to its root cause, without additional testing or coding.

FinOps practices allow cross-functional teams in Engineering, Finance, Product, etc work together to enable faster product delivery, while at the same time gaining more financial control and predictability.

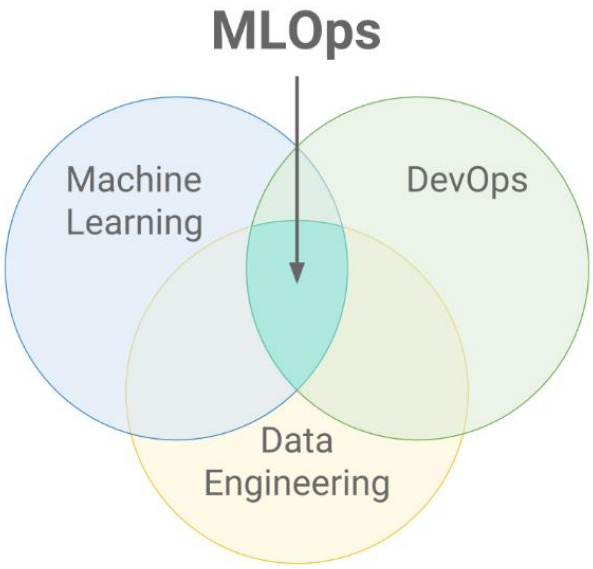
DataOps focuses on minimization of waste within the system without sacrificing productivity along with quick feedback cycle and fast feature delivery from agile development, and Devops practices.

How to implement Monitoring, Observability, FinOps and DataOps into our pipeline

Monitoring and Observability tools can be integrated/installed after the infrastructure has been provisioned with a managed container orchestration cluster, notably a Kubernetes cluster. Monitoring and Observability tools like Prometheus and Grafana can be installed directly on the Kubernetes cluster. AWS cloudwatch can be used to monitor and observe the cloud instances. Using the data collected by our tools, FinOps practice will be used to track instance usage and have each of the teams responsible for their own cloud resources. Finally, DataOps statistical models can be applied to the data collected to provide better business intelligence and values to the customers.

Phase 8 MLOps and AIOps (Gabriel)

MLOps (Machine Learning Operations): This is a core function of Machine Learning engineering, focused on streamlining the process of taking machine learning models to production, and then maintaining and monitoring them. MLOps is a collaborative function, often comprising data scientists, devops engineers, and IT.The machine learning lifecycle consists of many complex components such as data ingest, data prep, model training, model tuning, model deployment, model monitoring, explainability, and much more. It also requires collaboration and hand-offs across teams, from Data Engineering to Data Science to ML Engineering. Naturally, it requires stringent operational rigor to keep all these processes synchronous and working in tandem. MLOps encompasses the experimentation, iteration, and continuous improvement of the machine learning lifecycle.



Here are some key benefits **Big Bang** will benefit from adopting MLOps practices:

- Scalability and Efficiency: MLOps provides a systematic and scalable approach to managing machine learning models. It enables organizations to streamline the model development process, automate repetitive tasks, and leverage infrastructure and tools for efficient model training and deployment. MLOps practices ensure that models can handle larger data volumes, adapt to changing business needs, and scale to meet growing demand.

- Reproducibility and Version Control: MLOps emphasizes version control for models, code, and data. It enables organizations to track and manage changes made to models, ensure reproducibility of results, and maintain a history of model versions. This helps in auditing, collaboration among data scientists, and maintaining a clear lineage of models for compliance and regulatory purposes.
- Faster Time to Market: MLOps facilitates faster model deployment cycles. By automating the end-to-end process of model training, testing, deployment, and monitoring, organizations can accelerate the time to market for new models or model updates. This agility allows organizations to respond quickly to changing business requirements, gain a competitive edge, and deliver value more rapidly.

- Improved Model Performance and Stability: MLOps incorporates monitoring and performance tracking of deployed models. It helps organizations identify performance degradation, model drift, or data quality issues in real-time. By continuously monitoring model performance, organizations can proactively address issues, optimize models, and maintain high-quality predictions and outcomes.

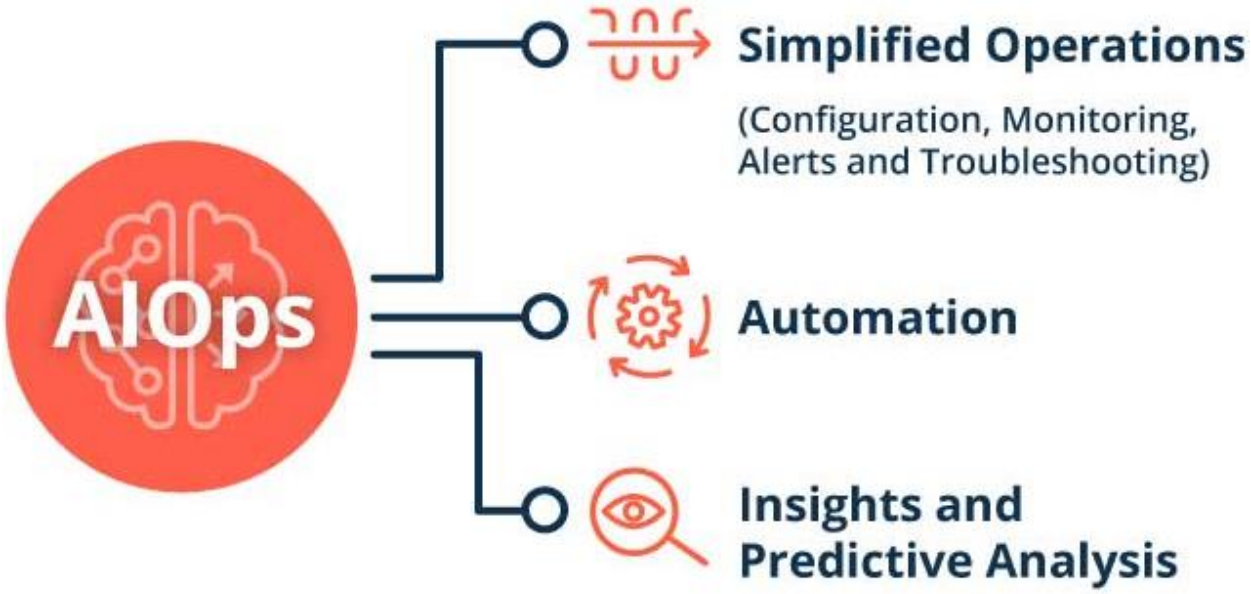
- Collaboration and Cross-functional Alignment: MLOps encourages collaboration among data scientists, developers, and operations teams. It establishes processes and tools that promote communication, knowledge sharing, and alignment between different stakeholders involved in model development and deployment. This collaboration ensures that models are developed with operational considerations in mind, leading to more successful and effective deployments.

- Governance and Compliance: MLOps practices help organizations enforce governance and compliance standards for machine learning models. With clear documentation, version control, and reproducibility, organizations can adhere to regulatory requirements, perform audits, and ensure the responsible and ethical use of models.

- Cost Optimization: MLOps allows organizations to optimize resource utilization and costs associated with machine learning models. By automating processes, monitoring resource consumption, and identifying performance bottlenecks, organizations can make informed decisions about resource allocation and optimize the efficiency of their machine learning infrastructure.

- By adopting MLOps practices, organizations can overcome challenges in managing machine learning models in production environments and unlock the full potential of their ML initiatives. MLOps promotes efficiency, scalability, collaboration, and governance, ultimately leading to more successful and impactful deployments of machine learning models.

AIOps (Artificial Intelligence for IT Operations): AIOps combines artificial intelligence and machine learning techniques with IT operations to enhance the efficiency and effectiveness of managing and monitoring IT infrastructure and services. AIOps leverages ML algorithms to analyze and interpret vast amounts of operational data, such as log files, metrics, events, and alerts, to detect anomalies, identify patterns, and provide actionable insights.



Here are some key benefits **Big Bang** will benefit from adopting AIOps practices:

- **Faster mean time to resolution (MTTR)**: By cutting through IT operations noise and correlating operations data from multiple IT environments, AIOps is able to identify root causes and propose solutions faster and more accurately than humanly possible. This enables organizations to set and achieve previously unthinkable MTTR goals. For example, IBM IT infrastructure reduced the mean time to repair (MTTR) for the company's app by 66%, from three days to one day or less.
- Lower operational costs: Automatic identification of operational issues and re-programmed response scripts will reduce operational costs, allowing for better resource allocation. This also frees up staffing resources to work on more innovative and complex work, leading to an improved employee experience. Through optimization, Providence saved more than USD 2 million while assuring app performance during peaks.

- More observability and better collaboration: Available integrations within AIOps monitoring tools facilitate more effective cross-team collaboration across DevOps, ITOps, governance and security functions. Better visibility, communication, and transparency allows these teams to improve decision-making and respond to issues more quickly. As an example, Fisher-price brought more observability to their container-based architecture, which improved app performance during the pandemic and reduced delivery latency by 98%.

How to integrate ML/AI ops into our pipeline
Open source ML toolings can be installed directly into the orchestration cluster (Kubernetes). Kubeflow builds on Kubernetes as a system for deploying, scaling, and managing complex ML systems. Kubeflow contains applications and scaffolding specifically for machine learning which ML engineers can leverage to generate machine learning models. Data collected from the overall system can be fed into the machine learning models to further optimize the state of the system.

Conclusion

As with many legacy software systems like Big-Bang, the challenge has always been migrating existing systems to modern DevOps practices. The pipeline design is segregated into implementation phases and together will construct a functional CI/CD pipeline system that allows Big-Bang to close the gaps between various silos, meeting their business goals, and most important of all providing values to the customers.