**Case Study**

Assignment 4: CSDO 1020 – DevOps CICD Pipeline Modernization  
  
Submitted July 09, 2023

Members:

1. Yael Karmani (220760047)
2. Gary (Lyndon) Reid (220546388)
3. Vasudeo Rane (220778213)
4. Sopuruchi Owen (220143962)
5. Sotola Olusegun (220275715 )

# 

[**Introduction - Gary Lyndon Reid 3**](#_rrs1ygant965)

[**Next-Gen DevOps can improve our innovation, speed, agility and accelerate digital transformation - Yael Karmani 4**](#_6xirqj4tby02)

[**How SRE will help maintain application reliability - Gary Lyndon Reid 6**](#_py2tter1yo3k)

[**DevOps impact on business Productivity - Gary Lyndon Reid 7**](#_vcn5znnozx3a)

[**Explain and apply DevOps best practices in relation to cloud-native, serverless and Kubernetes - Vasudeo 8**](#_9ltkiy3wxq9d)

[**DevOps Impact On Financial Results - Sotola Olusegun 11**](#_v4wir6l2swtt)

[**Pipeline Design Diagram - Sopuruchi Owen 12**](#_ksi0gpkcdrmg)

[**Conclusion 17**](#_mofek9gci4pi)

# 

# 

# Introduction - Gary Lyndon Reid

This document outline many critical aspects DevOps. It aims to achieve three goals:

1. **Educate**

Our hope is that you and your teams will better understand the benefits of embracing DevOps, Cloud, FinOps, and SRE practices. We’ve outlined what these concepts mean and how they can accelerate software delivery, minimize cost, and increase our revenue.

1. **Support**

We are seeking your support to continue our journey of building and fostering a world-class multi-cloud DevOps organization. We are looking for developers to embrace the cloud and DevOps culture, we need product owners to understand the benefits, we need finance to approve investment, and we need leadership support.

1. **Design**

Finally, we will share our CICD pipeline design. The design should help you understand the patterns that will be embraced to increase the speed to ship, decrease the MTTR, and ultimately delight our customers.

Please reach out to our team with any questions at all. And if you’d like to get really involved, we invite you to join our DevOps Jam Sessions.

We host them every Friday from 11:00am to 1:00pm in Room 16C in Building A. We broadcast them via the DevOps Rocks Microsoft Teams channel for remote participation, but we ask that you attend in person if you are located in the GTA.

# 

# Next-Gen DevOps can improve our innovation, speed, agility and accelerate digital transformation - Yael Karmani

Next-Generation DevOps can greatly enhance innovation, speed, agility, and accelerate digital transformation within organizations. By adopting these advanced technologies, tool with efficient DevOps practices, businesses can revolutionize their software development and deployment processes, resulting in faster delivery of updates and new features while promoting collaboration and efficiency.

key aspects and example tools that leverage machine learning and AI to achieve these benefits:

**Better Collaboration and Communication:** Next-Gen DevOps encourages teams to work together and leverage AI-powered collaboration tools such as Slack, Microsoft Teams, and Jira. These tools facilitate real-time communication, knowledge sharing, and efficient decision-making, leading to improved teamwork, faster progress, and innovative problem-solving.

**Agile Development:** Next-Gen DevOps embraces AI and machine learning algorithms, including tools like TensorFlow and PyTorch, to enhance Agile development practices. These tools enable organizations to analyze data, user feedback, and market trends, providing valuable insights for continuous improvement and enabling them to respond quickly to changing requirements.

**Automated Infrastructure**: Next-Gen DevOps utilizes AI-driven tools like Terraform and Ansible for infrastructure management and optimization. These tools leverage AI algorithms to automate provisioning, configuration, and management of infrastructure resources, resulting in more efficient infrastructure utilization and reduced downtime.

**Scalability and Reliability**: Next-Gen DevOps leverages machine learning techniques and tools like Kubernetes and Docker to ensure scalable and reliable applications. These tools enable organizations to automate containerization, orchestration, and scaling processes, allowing systems to handle increased workloads, maintain availability, and adapt to changing resource requirements.

**Continuous Learning and Improvement**: Next-Gen DevOps promotes a culture of continuous learning and improvement using AI-driven analytics tools such as Splunk and ELK Stack. These tools can analyze vast amounts of data collected from various sources, identify patterns, detect anomalies, and provide actionable insights for optimizing processes and enhancing performance.

**Monitoring and Visibility:** Next-Gen DevOps relies on AI-powered monitoring and observability tools like Prometheus and Grafana. These tools use machine learning algorithms to analyze system metrics, log data, and user behaviour, enabling proactive detection of issues, rapid troubleshooting, and ensuring optimal application performance.

**AI-Enabled Testing and Deployment:** Next-Gen DevOps incorporates AI in testing and deployment processes with tools like Jenkins X and GitLab CI/CD. These tools leverage AI algorithms to automate testing, identify potential defects, and optimize release strategies, ensuring high-quality software and faster time-to-market.

With Next-Generation DevOps organisations can work more efficiently as teams, deliver products faster, and continuously improve their processes. This not only helps businesses stay competitive in the digital world but also enables them to leverage the power of AI and machine learning through the use of mentioned tools.

# 

# 

# How SRE will help maintain application reliability - Gary Lyndon Reid

Our SRE can help us maintain the reliability of our application by leveraging many best practices. A core responsibility of our Site Reliability Engineers is to work with stakeholders to define Service Level Objectives, Service Level Agreements, and Sevice Level Indicators. Once these metrics are defined, we will have quantifiable targets. Our SRE will then instrument our application so that they can monitor the health of the application and notify the operations team in the event that performance is not as expected.

In addition to the above, our SREs will assist with all aspects of site reliability, and that includes items such as:

* Load, performance, and other nonfunctional testing
* Assisting with the design of highly available\fault tolerant applications and infrastructure
* Disaster recovery design and testing
* Embracing automation to reduce manual tasks (DevOps)

Ultimately, our investment in Site Reliability Engineering will position us to achieve our reliability targets. Our customers will be thrilled, and we will reduce the time spent troubleshooting issues and increase our time shipping features.

# 

# 

# DevOps impact on business Productivity - Gary Lyndon Reid

Establishing DevOps practices will have a positive impact on business productivity. As outlined in detail in the document, we will achieve several key benefits. You may be familiar with the project management that says you can choose 2 of 3 - fast, cheap, quality. With DevOps, we can have all three - we can ship code faster, ensure that we are shipping high-quality code, and do it cheaply. Best of all, we will spend less time firefighting, allowing us to invest more time and money into shipping features for our customers.

So how will we achieve all of this? We will build a fully automated CICD pipeline to provision infrastructure and data and ship our code for testing. We will write fully automated tests to quickly confirm that our features are high quality. We will deploy our code via automation once the tests are passed. We will instrument our application and work with SREs to ensure the reliability of our sites. Our pipeline will include functional tests, regression tests, static security scanning, dynamic security testing, static code analysis, and more. Our infrastructure, or rather our containers that operate in AWS, will allow for easy scaling.

So read on to learn more about the specific tools, techniques and processes that we have implemented or plan on implementing to achieve our vision of increased business productivity, increased customer satisfaction, and increased employee satisfaction.

# 

# Explain and apply DevOps best practices in relation to cloud-native, serverless and Kubernetes - Vasudeo

DevOps best practices are :-

**Cultural Change** :- DevOps requires collaboration, transparency, trust, and empathy. Teams has to overcome working in silos and encourage teams to have cross-team communication or collaboration.

Cloud Native: Cloud-native environments promote collaboration by providing shared platforms and tools that facilitate communication and knowledge sharing among development, operations, and other teams. The use of shared services, APIs, and infrastructure-as-code enables teams to work together more effectively.

Serverless: Serverless architectures encourage small, focused teams that work closely together to develop and deploy individual functions or microservices. This collaborative approach fosters cross-functional communication and a sense of shared responsibility for delivering working software.

Kubernetes: Kubernetes enables the orchestration and management of containerized applications, which promotes collaboration between developers and operations teams. Kubernetes provides a common platform that encourages joint ownership of the deployment and monitoring of applications, bridging the gap between development and operations.

**Delivering fast using agile project management -** Agile methodology is a project management approach that prioritizes cross-functional collaboration and continuous improvement. It divides projects into smaller phases and guides teams through cycles of planning, execution, and evaluation.

Cloud Native, Serverless, and Kubernetes can significantly support Agile project management by providing the necessary infrastructure, scalability, and flexibility to deliver software products iteratively and respond quickly to changing requirements. These technologies enable teams to respond to customer feedback and changing requirements swiftly, fostering an iterative and customer-centric approach to software development.

**Shift left with CI/CD -** Shift left is an approach that moves testing to earlier in the software development lifecycle. Continuous integration (CI) is a process that improves code quality through deployment pipelines and support shift left strategy.

Cloud-native environments allow teams to replicate production-like environments easily, thus enables to identify and address issues related to scalability, performance, and integration early in the development process, reducing the chances of encountering such problems during later stages.

With serverless architectures, individual functions or microservices can be tested independently, enabling more targeted and granular testing. This approach facilitates early detection of issues and enables developers to fix them before they impact the overall system.

Kubernetes enables the containerization of applications, which provides a consistent and isolated environment for testing. Containers can be created and destroyed quickly, allowing developers to perform various types of testing, such as unit testing, integration testing, and end-to-end testing, in parallel

**Tool Strategy**:- A DevOps toolchain requires the right tools for each phase of the DevOps lifecycle, with key capabilities to improve software quality and speed of delivery. Cloud Native, Serverless, and Kubernetes provide the platform/tools which help to achieve the goals.

**Automation:-** Using automation the delivery of the software was done early. There is less manual intervention and hence there are fewer chances of any human error involved. Automation in build, implementation, testing, and deployment is the key aspect in DevOps implementation.

Cloud-native platforms provide tools and services for automated testing, such as integration testing, load testing, and security testing. By integrating these tests into the CI/CD pipeline, teams can identify and resolve issues early, ensuring higher quality and reliability of the application.

Serverless platforms provide fast execution and deployment of functions or microservices. This allows developers to quickly receive feedback on code changes, ensuring that issues are caught early in the development process and reducing the time required for debugging and resolution.

Kubernetes helps in automating the deployment, scaling, and management of containerized applications.

**Observability:-** For the successful implementation of DevOps monitoring play a critical role. The early an issue is identified helps it to fix early respecting in less effort and time as compared to identifying the issue later.

Cloud Monitoring Services provide very useful tools for monitoring services, and gathering logs which helps to gauge the health of cloud-native applications.

Serverless platforms like AWS Lambda and Azure Functions provide built-in monitoring capabilities that capture metrics at the function level. You can monitor execution duration, invocation counts, error rates, and more. This helps identify performance issues and optimize the function's performance.

Kubernetes offers built-in monitoring components to collect various metrics, tools like Prometheus and Grafana can scrape metrics exposed by the pods and provide insights into resource consumption, response times, and other performance indicators.

# DevOps Impact On Financial Results - Sotola Olusegun

Integrating DevOps with Financial Services can bring to your company multiple benefits. And the best part is that you can improve your processes within your business and while having happy clients. In the end,By implementing DevOps, financial companies can automate key business processes, accelerate the deployment of new features, and enhance the reliability of their systems.DevOps in financial services has brought in the much-needed revolution in the Financial Services industry and helped shed its age-old, traditional, and siloed practices and replace them with modern, agile, and integrated systems that helped the finance industry immensely.But maybe you are wondering What is the relation between Financial Services and DevOps?

Well, since Financial Services companies are all about working with money, they need to be sure to count on quick delivery of their services in order to have continuous money flow, and as a result, a higher customer experience, which translates to higher engagement. And the only way to achieve these is through DevOps practices.

Highlighted below are various ways Devops have impacted Positively in the Financial services sector..

1. Reliability

Reliability has two main fields. On one side, applying DevOps for Financial Services ensures that your applications are up and running all the time without any interruptions, or at least, just a few of them. Certainly, if this happens, your customers are going to be happy with your services because they are permanently available without any kind of downtime.

On the other hand, reliability also means that you have a team you can rely on. A nearshore team would be an excellent idea. Why? Because you can be sure that you are working at the right time with experts in the field. As a curious fact, there is a 62% productivity increase in people who work from home.

2. Security Compliance

Financial Services are always dealing with investments and transactions, in consequence, they need to have all their data protected. That’s why security is a key aspect of your company and a well-protected app needs to be regulated by the best security standards and regulations in the Cloud. One of the most important aspects of security compliance is DevSecOps which is a practice to implement security during all the Software Development Lifecycle (SDLC) of DevOps practices.

3. Automation

DevOps can help Financial Services to automate their processes, a really valuable benefit because sometimes it can take a lot of time. For example, if you deal constantly with repetitive tasks, you can automate them and simplify the processes that are too complicated. This automation can help to create even from reports to big data.

In addition, automation minimize human errors and as a result, increases accuracy and safety measures. With automation, Fintech companies can benefit from reducing costs and increasing the speed of their services.

4. Cost savings

Migrating to the cloud can help your company reduce your costs by 20% or 30%. And if after that, you adopt the DevOps practices you can reduce your costs another 20% or 30%. Outsourcing a DevOps company in the USA will cost you around 90 – 120 USD; with a nearshore company, it’ll be around 45 – 60 USD. In the end, while outsourcing Remote DevOps you’ll have a cost reduction on DevOps for a considerable 50%..

5. Compliance

With new laws being enforced and customers concerned about the optimization and security, compliance is such a great benefit. With Soc2, customers will be confident of not getting their data compromised due to hacking or attacks. What is Soc2 and what is it required to get compliant? Soc2 is an international standard that establishes the basic security criteria to fully protect your application interface and your customers’ data. Getting compliant will increase security protection and faster releases to the market while decreasing

6. Time-to-market

Speeding up the time to market is an excellent benefit because, in a matter of days or even seconds, you can release new issues instead of waiting a long time. In order to fulfill these objectives, companies need to adopt the best DevOps practices to reduce their time to market.

Nowadays, most of the business work with applications with the objective of complete sales and operate in an easier way. In consequence, they need to bring those applications to the market very quickly so they can’t miss available opportunities.

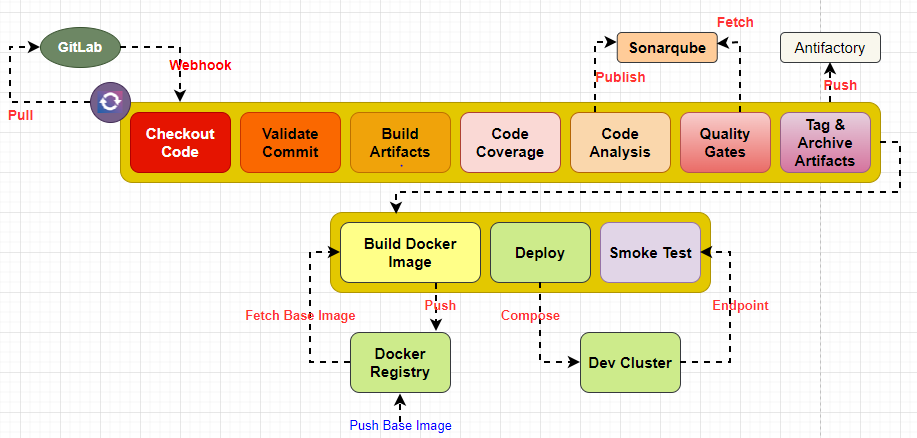
# Pipeline Design Diagram - Sopuruchi Owen

**NEXT-GEN DEVOPS CI-CD PIPELINE**

**Stages of Build Pipeline:**

- Trigger[Webhook(On-commit/merge), Scheduled, On-demand], Checkout, Build, Unit Test, Code Coverage, Store Artifacts, Product Build Image, Smoke Test, Push Image to Docker Registry.

Each stage represents a distinct step in the CI/CD pipeline, where the output of one stage becomes the input for the subsequent stage. The trigger determines when the pipeline starts, and each stage performs specific actions to ensure code validation, building, testing, artifact management, and eventually the creation and deployment of a Docker image.



**Trigger:**

* Webhook (on commit/merge): Automatically triggers the pipeline when a commit or merge event occurs in the repository.
* Scheduled: Runs the pipeline on a predefined schedule (e.g., nightly) to ensure regular builds and tests.
* On-demand: Allows manual triggering of the pipeline as needed, such as for specific debugging or testing purposes.

Connection: The trigger determines when the pipeline is initiated, based on events (webhook), time (scheduled), or user interaction (on-demand).

**Checkout:**

* Retrieves the source code from the repository, making it available for subsequent stages.
* Ensures that the pipeline has access to the latest version of the codebase for further processing.

**Validate code:**

* Conducts static code analysis, linting, or other checks to ensure code quality and adherence to coding standards using Sonaqube.
* Helps catch common issues early in the pipeline, reducing the likelihood of errors later on.

**Build Artifacts:**

* Compiles the source code (Human readable form) and generates the executable or deployable artifacts (.Object Files or Class file for Jar or War)
* Transforms the codebase into a format that can be executed or deployed to a target environment.

**Unit Test:**

* Executes a suite of tests designed to verify the functionality of individual units (e.g., methods, functions, classes) within the codebase.
* Helps identify bugs, errors, or regressions at the component level.

**Code Coverage:**

* Measures the extent to which the codebase is exercised by the unit tests.
* Provides insights into the effectiveness and thoroughness of the testing process.

**Artifactory**:

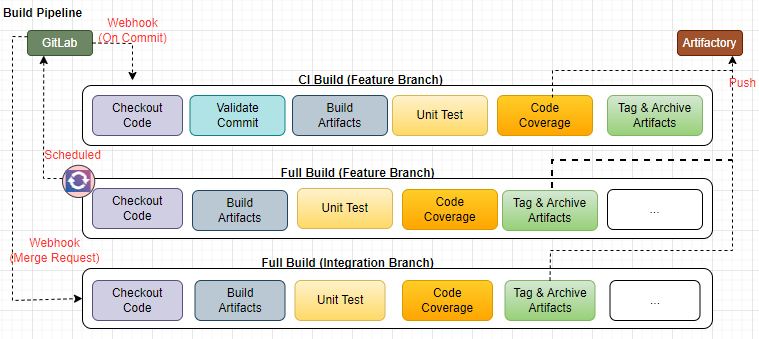
* Archives and stores the build artifacts (e.g., compiled binaries, JAR files) generated during the build stage.
* Allows for easy retrieval and traceability of the artifacts for future use or deployment.

**Product Build Image:**

* Builds a Docker image for the product, containing the necessary dependencies and configurations.
* Enables packaging the application as a containerized image for easier deployment and scalability.

**Smoke Test:**

* Executes a set of high-level tests to ensure that critical functionalities of the product work as expected.
* Verifies that the application is able to perform basic operations and provide a satisfactory user experience.
* Push Image to Docker registry:



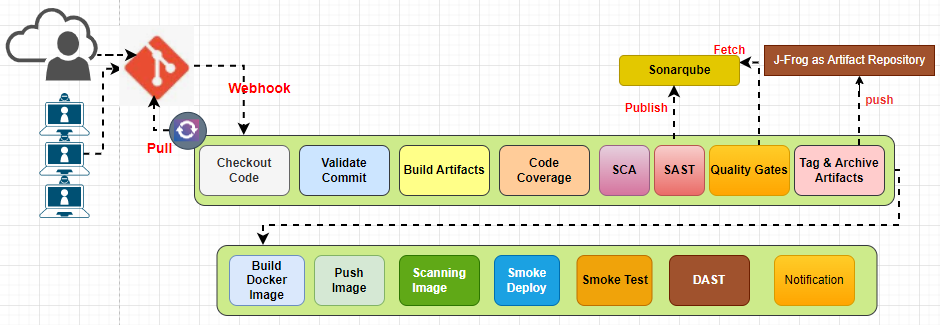
NB: This is a single Jenkins Pipeline but used the diagram to give a pictorial representation of CI Build, Full Build(Feature Branch) and Full Build(Integration Branch)

**NEXT-GEN DEVSECOPS CI-CD PIPELINE**

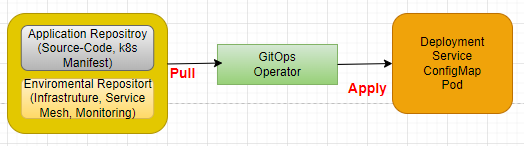
The Diagram below represents the DevSecops based CICD pipeline for SAST, DAST, Vulnerabilities, GitOps for Next-Gen DevOps

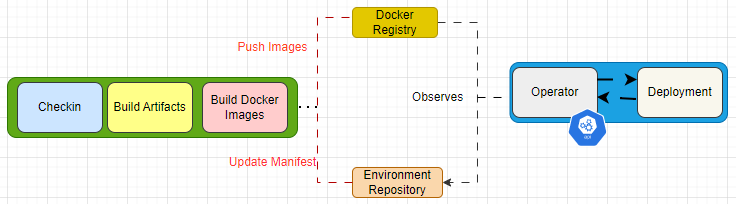
**Stages of DevSecops Build Pipeline**:

- Trigger [ Webhook(On-commit/merge), Scheduled, On-demand ], Checkout, Validate code, Build, Unit Test, Code Coverage, SCA, SAST, Quality Gates, Store Artifacts, Product Build Image, Scan Image, DAST, Smoke test, Push Image to Docker registry



**DEVOPS CONTINUOUS DELIVERY (GITOPS)-ARGOCD**





The Diagram above represents DevOps/DevSecOps implementation of CICD practice using Jenkins and Argo CD, a popular GitOps tool, using Kubernetes.

**VERSION CONTROL SYSTEM:**

Set up Git as a repository and GitFlow as a branching model for our Branching strategy to enable a well structured and formal release process for our code to aid our team of Developers and operators to collaborate effectively and for hotfixes and long-term support for multiple versions. The ‘**master**’ branch represents the stable production-ready code while our ‘**develop**’ branch serves the **integration branch** for our ongoing developments. The feature branches are created off the ‘**develop**’ branch for new features or **enhancement**.Here once a feature is created, it is merged back into the develop branch through a pull request. Our **Release branches** are created from the ‘**develop**’ branch to prepare for **production release**. We also have a hotfix, which we use to address critical issues in the production codebase and merge back into the ‘master’ and ‘develop’.

**CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY (CI/CD): GITLAB CI/CD WITH JENKINS AS CI SERVER USING AMAZON EC2 INSTANCE**

**Set up Jenkins:**

* Install and configure Jenkins on a server (EC2 Instance).
* Install the necessary plugins for Git integration and pipeline support.

**Configure Jenkins to Monitor GitLab:**

Jenkins is configured with GitLab url to point to GitLab Repository with necessary access credentials to monitor Gitlab repositories. We agreed to use Periodic Polling at specified intervals to check for changes in GitLab. Jenkins will trigger the pipeline if it detects any update.

**Define the Jenkins Pipeline: Declarative**

* Create Jenkinsfile in GitLab repository for pipeline.
* Jenkinsfile defines stages, steps, and commands.
* Define stages: checkout, build Artifacts, test, deploy.
* Each stage and specify steps: compiling code, running tests, generating artifacts

**GitLab CI/CD Configuration:**

From the GitLab repository:

* create a .gitlab-ci.yml file to configure the CI/CD pipeline with Jenkins instance as the executor.
* Stage and Step definition for the pipeline corresponding to the stages defined in the Jenkinsfile.
* Configure GitLab CICD pipeline to trigger the Jenkins job when changes are pushed to the repository.

**Jenkins Pipeline Configuration:**

* In Jenkins, configure the Jenkins pipeline to execute the Jenkinsfile in response to GitLab CI/CD triggers.
* Set up Jenkins credentials to access GitLab repositories securely.
* Define environment variables and parameters for the pipeline, if needed.
* Configure build agents or slaves to distribute the workload, if required.

**Build Process:** within the Jenkins pipeline, the build process includes the necessary steps:

* Check out the source code from the GitLab repository.
* Compile the code using appropriate build tools and configurations.
* Run unit tests and generate test reports.
* Perform code analysis, dependency management, or packaging.
* Publish build artifacts to Jfrog Artifactory .

**Integration with GitLab:**

* Jenkins is configured to report the build status and test results back to GitLab.
* Use Jenkins plugins or GitLab integrations to publish build and test reports in GitLab.
* Set up notifications or webhooks to alert team members about build failures or important events using Slack or email.

# Conclusion

In summary, adopting DevOps tools and practices and embracing a DevOps culture will allow us to achieve our corporate goals. The investment will pay returns in increased productivity and increased revenue. The productivity gains will be achieved through reduction of friction in the software developement life cycle and revenue gains will be realized through increased speed to market and increased customer satisfaction.