Critique Report

DevOps Practices we followed:

- Version Control for All Production Artifacts
- Continuous Integration and Deployment
- Automated Acceptance Testing
- Peer Review of Production Changes
- High-Trust Culture
- Proactive Monitoring of the Production Environment
- Win-Win Relationship (and Outcomes) Between Dev and Ops

Agile project management: Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Agile teams focus on delivering work in smaller increments, instead of waiting for a single massive release date. Requirements, plans, and results are evaluated continuously, allowing teams to respond to feedback and pivot as necessary.

Source Code Management: Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version control.

Continuous Build Management: Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information in the "About Maven" section of the navigation.

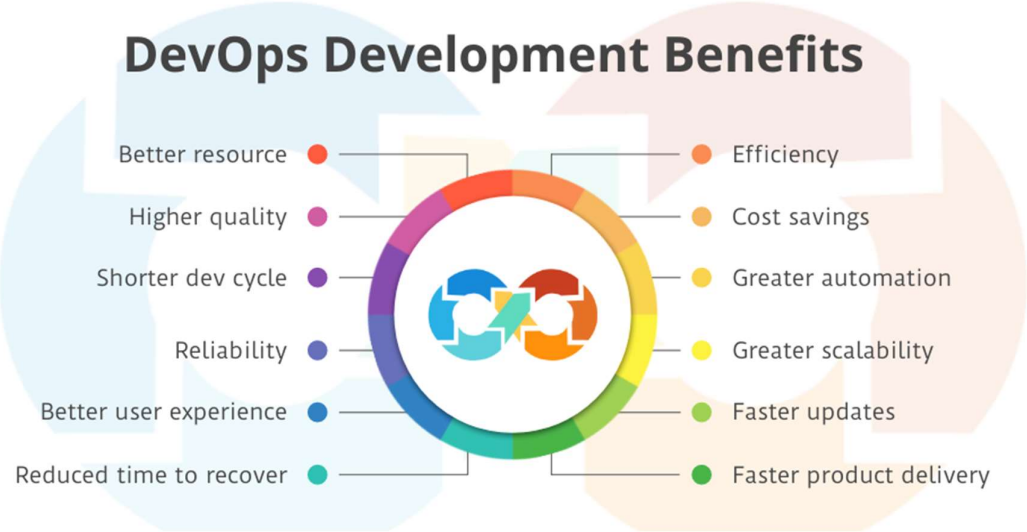
Continuous Integration and Continuous Delivery: CD is a software development methodology where the release process is automated. Every software change is automatically built, tested, and deployed to production. Before the final push to production, a person, an automated test, or a business rule decides when the final push should occur. Although every successful software change can be immediately released to production with continuous delivery, not all changes need to be released right away.

CI is a software development practice where members of a team use a version control system and frequently integrate their work to the same location, such as a main branch. Each change is built and verified to detect integration errors as quickly as possible. Continuous integration is focused on automatically building and testing code, as compared to *continuous delivery*, which automates the entire software release process up to production.

Configuration Management: Continuous performance management is a modern, human-centered approach to promoting, evaluating, and improving employee performance. It enables your organization to create a trusted environment in which employees feel empowered to take control of their own development.

Continuous Deployment: Continuous Deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment.

Benefits of these Practices: Teams that practice DevOps ship better work faster, streamline incident responses, and improve collaboration and communication across teams.



Enhancements in current practices:

1. Think agile, stay agile
2. Innovate everyday
3. Use the right tools
4. Stop the line not only when the build breaks but also when something breaks.
5. Use peer reviews for better quality; leverage your team's familiarity, shared goals, and mutual accountability, as opposed to external change approval.
6. Monitor and communicate across the teams so everyone can see, understand, and affect end results and customer utilization.
7. Win-Win Relationship (and Outcomes) Between Dev and Ops approach counters the learned behaviour that deployments hurt.
8. Monitor DevOps pipelines and applications.
9. The goal of product management is get the best product to market as quickly as possible.
10. High Trust Culture: This is both a practice and an outcome result from a single source of truth, peer reviews, and shared goals.
11. Organizations should make sure they version everything.
12. Using containers to drive microservices
13. Automating configuring and provisioning with infrastructure as code (IAC)

Challenges / Learnings:

• First, we thought of going through centralized workflow of git, but we were getting lot of conflicts as we directly commit and push to the central repository.

• So, we have started the Feature branch workflow and leveraged the feature of Pull Requests which made a developer new changes transparent to all the developers and helped in reducing the conflicts.

• Had explored SonarQube for code quality and later we have explored Jenkins and made couple of pipelines for our CI/CD. So, these all are of great learnings for us.

To Conclude in short:

• We have created instances of Jenkins, sonar, and tomcat server in Google Cloud Platform.

• We have two pipeline jobs one for backend and other for UI.

• We have used GitHub hook to trigger our pipeline jobs automatically whenever there is a
commit, push, or merge of code into our central repository.

• Backend pipeline job first clones the source code, performs build to create WAR file, performs the sonar check using the sonar instance deployed in our GCP then at last deploys this WAR file.

• UI pipeline job first clones the source code, performs the build, and then deploys the UI source code.