DevOps Training

19/10/2024

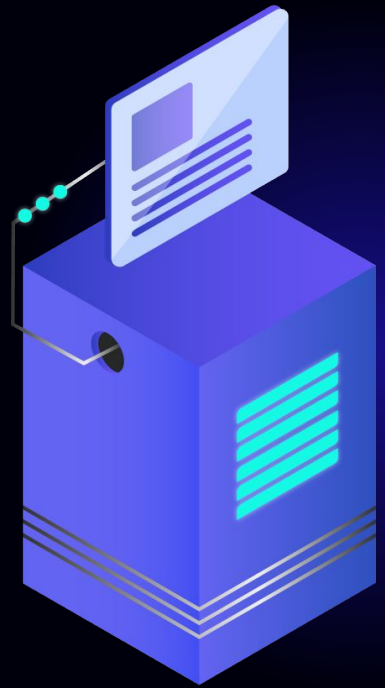
Presented by: Besma Guesmi

TABLE OF CONTENTS

01

ABOUT ME

Besma Guesmi, Lead Data Scientist @ Ubotica Technologies with over 3 years of experience in Data Science, Computer Vision and Edge AI.

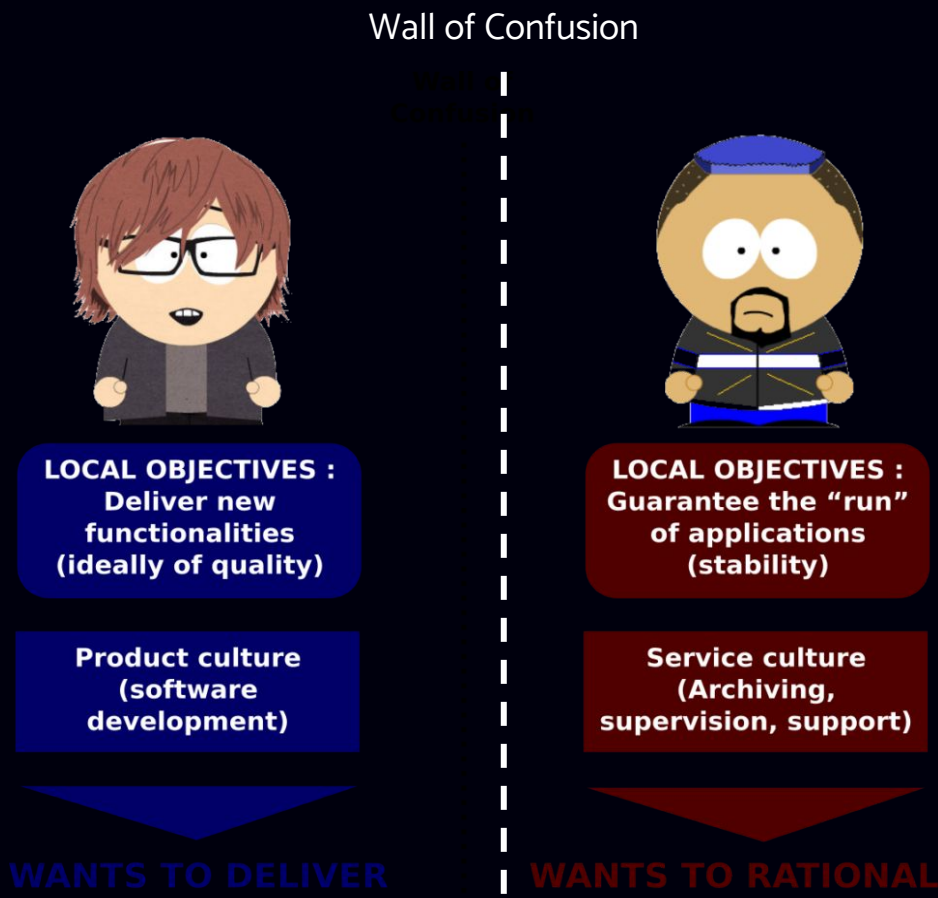


INTRODUCTION

Session 1 covers the fundamentals of DevOps, its importance, core principles, and how it integrates Agile practices. It explores the DevOps lifecycle stages and includes setting up a GitHub repository for practical use throughout the course.

Unite Dev and Ops: Breaking the Silos

- **Developers** focus on writing new code, releasing features quickly, and meeting business demands.
- **Operations** prioritise system stability, uptime, and minimising risks from new changes

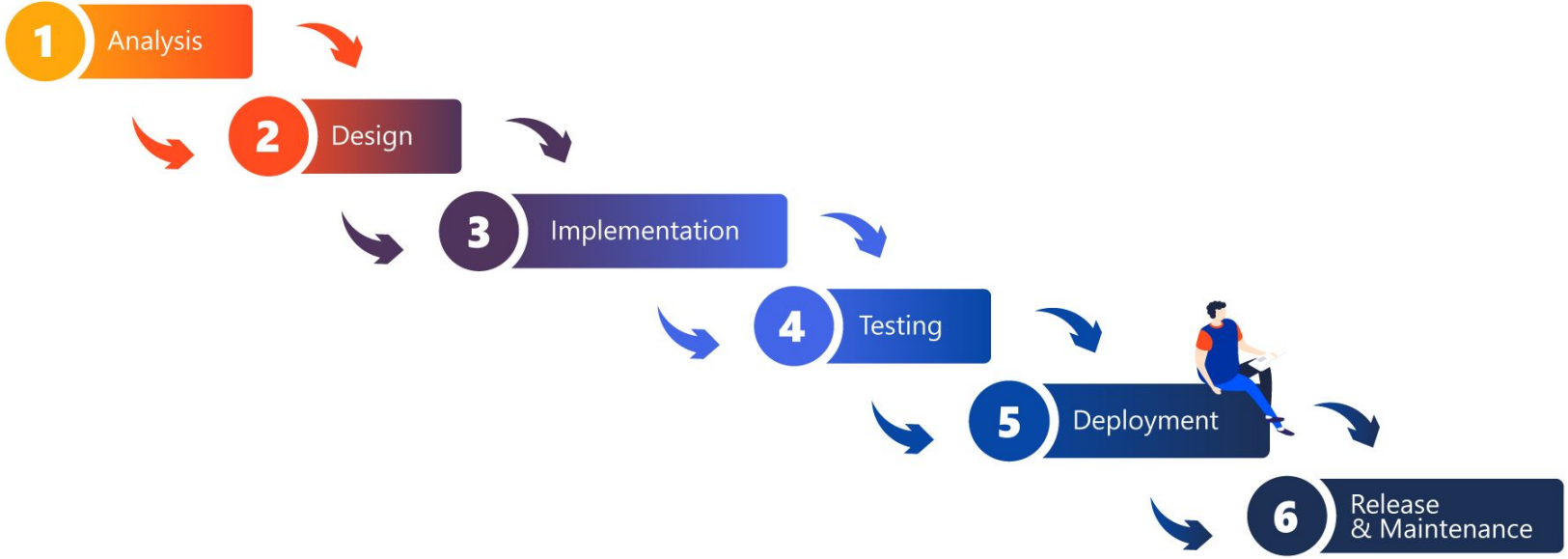


How we got to DevOps

- Before 2000, most software was developed and updated by using the waterfall methodology, a linear approach to large-scale development projects.
- Development teams spent months developing large bodies of new code.
- Changes impacted most or all of the application lifecycle.
- Extensive changes required several additional months for integration into the codebase.
- QA, security, and operations teams spent months testing the code, resulted in long delays between software releases.
- Often included several significant patches or bug fixes between releases.
- Characterised by complex and risky deployment strategies.
- Hard-to-schedule interlocks with upstream and downstream systems.
- Relied on the hope that business requirements had not changed drastically during development.

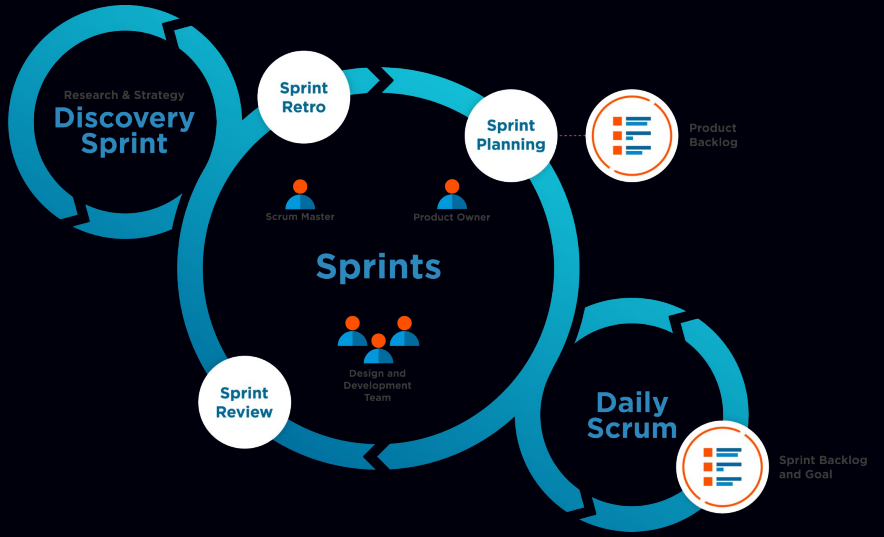
Waterfall

approach



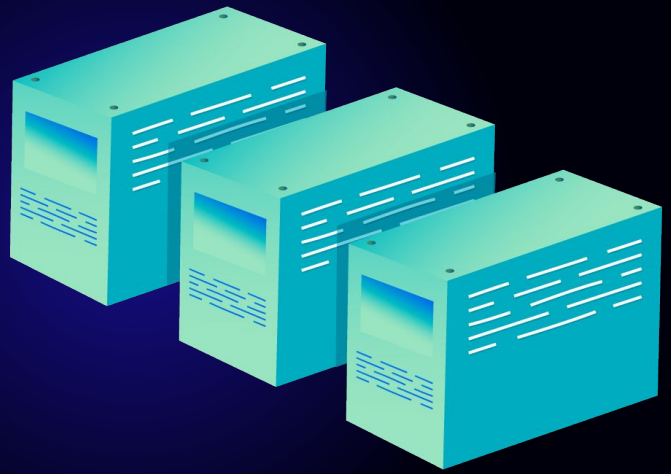
AGILE Methodology

Agile methodology is a software development approach that emphasises iterative development, collaboration, and flexibility. It focuses on delivering smaller, incremental updates, allowing teams to respond quickly to changing requirements and stakeholder feedback. By promoting cross-functional teams and utilising practices like Scrum and Kanban, Agile enhances product quality and customer satisfaction through continuous improvement.



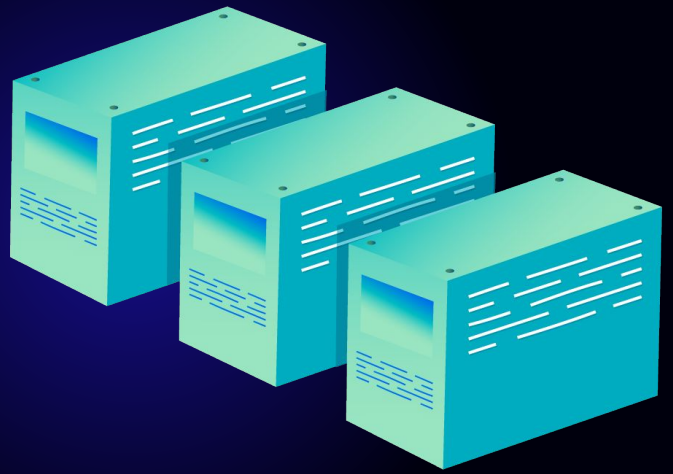
What is DevOps

DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams. It emphasises team empowerment, cross-team communication and collaboration, and technology automation.



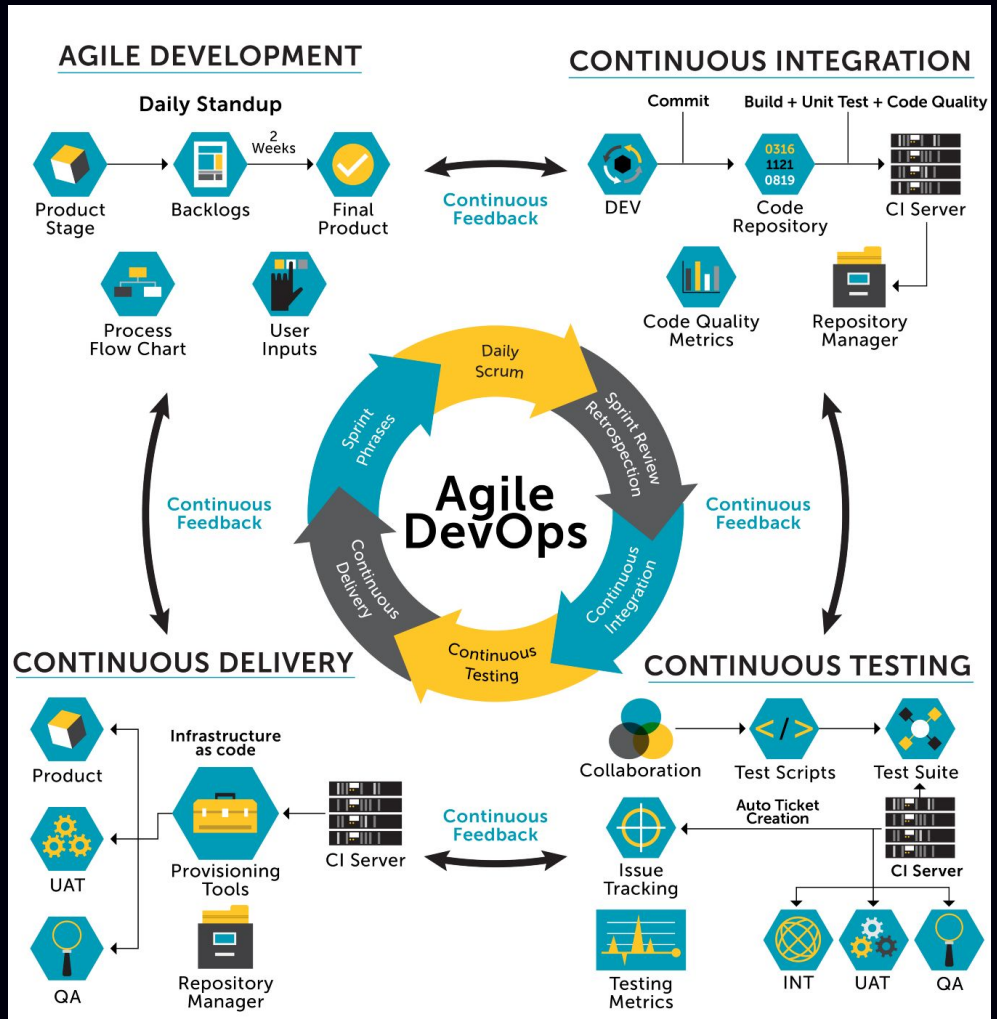
What is DevOps

- **DevOps** is a movement that emerged around 2007, addressing concerns about the traditional separation between software development and IT operations.
- **DevOps** integrates development (Dev) and operations (Ops) into a continuous process, fostering collaboration and improving software delivery.

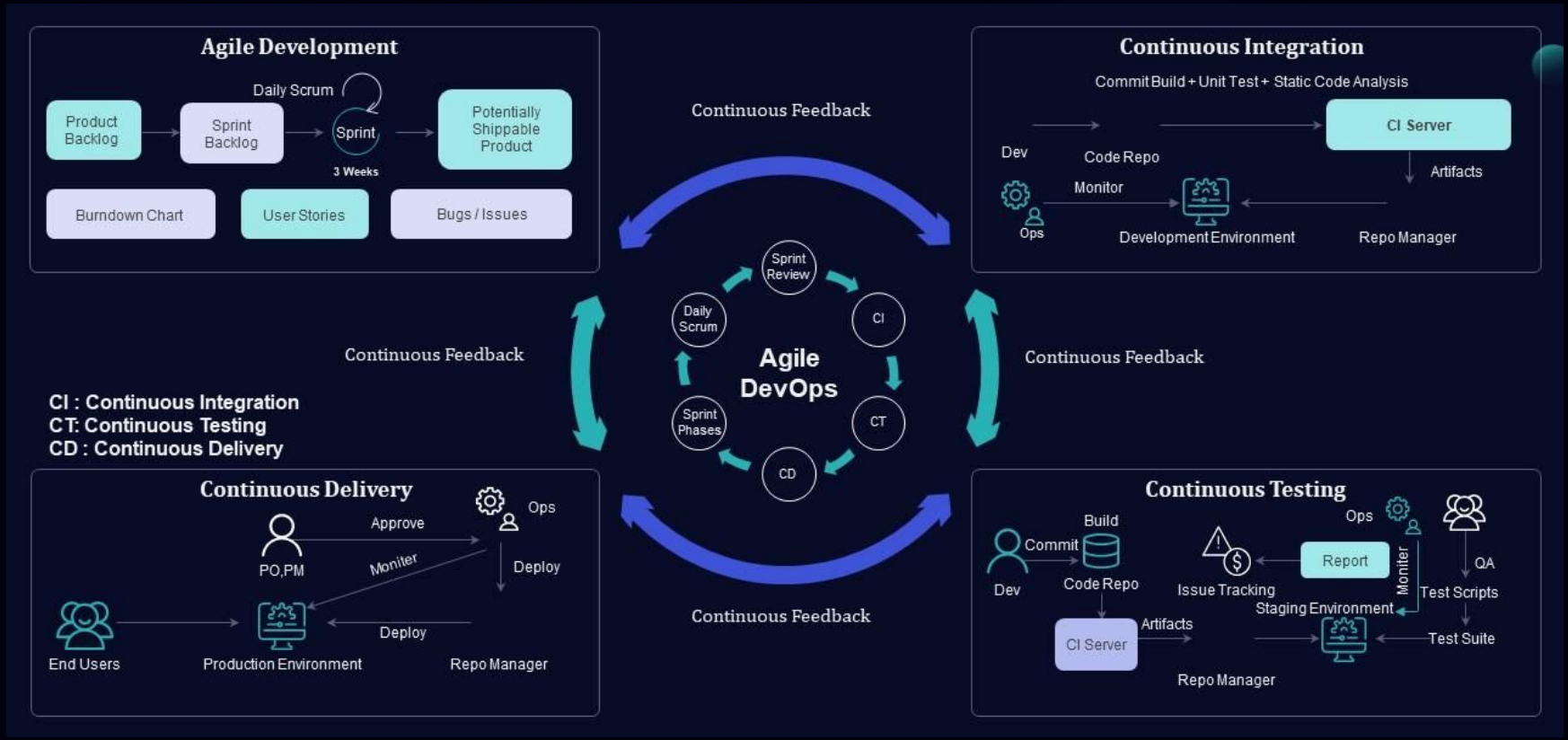


What's the difference?

A technical practice to achieve end to end solution	It is more about processes like scrum methodology and sprints
Operations as deployment is just a small part of ops. Other being configuration management etc.	It is the link between Design Thinking and DevOps
Aligns Ops with stability, such as scripting deployment and standardizing app builds	Aligns development with business priorities, thus focusing on speeding up the development process
DevOps helps to bridge the line between several cross functional teams (Development, QA, Monitoring, Deployment, Infrastructure etc.) by simplifying the process	In agile, the gap between these teams are bridged in daily scrum calls where all roadblocks are discussed and resolved with agile scrum methodologies.
Aims at faster deliveries through automation via various applications and tools like Docker, Chef, Jira, Configuration Management and Splunk. These tools are used for automating deployment and tracking bugs in real-time for quick fix.	Aims at streamlined delivery via tools like Kanban Board, Scrum, Extreme Programming (XP), Test Driven Development (TDD), Feature Driven Development (FDD), and Lean. These practices help in project management, planning and collaboration.



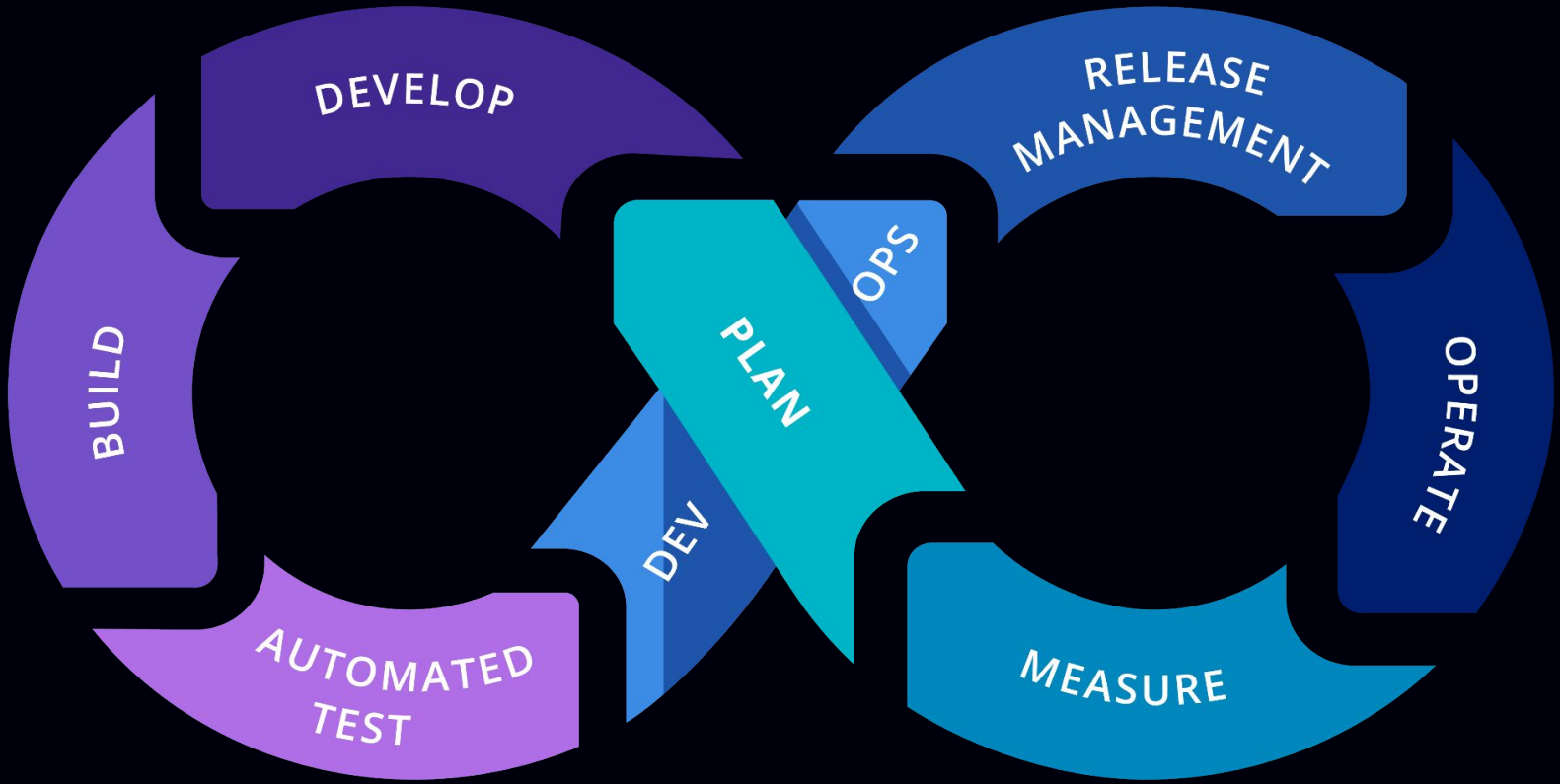
AGILE DevOps Process



How Does DevOps Works?

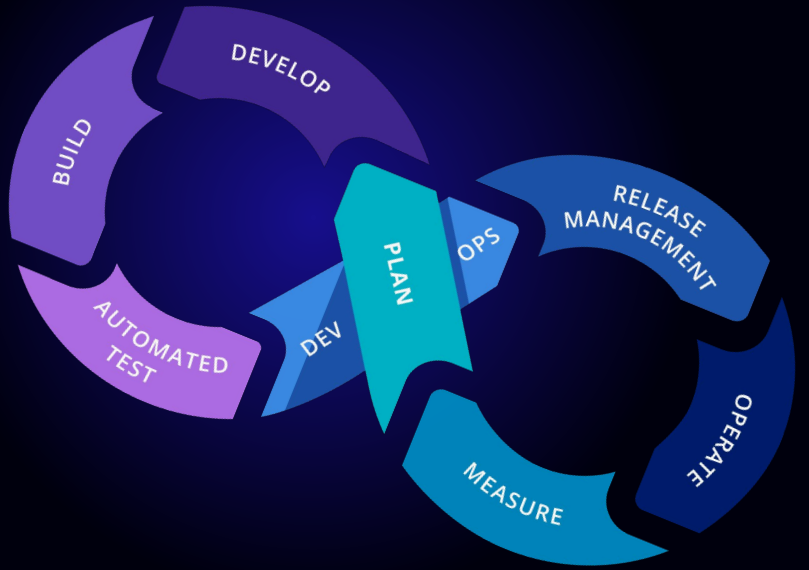
- **Collaborative Teams:** DevOps teams consist of developers and IT operations working together throughout the product lifecycle to enhance speed and quality of software deployment.
- **Cultural Shift:** This approach represents a significant cultural change, breaking down silos between development and operations, sometimes merging them into a single multidisciplinary team.
- **Cross-Functional Skills:** Engineers in DevOps teams possess a diverse skill set that allows them to engage across the entire application lifecycle—from development and testing to deployment and operations.
- **Automation and Toolchain:** DevOps teams leverage tools to automate processes, increasing reliability and efficiency. Key components include continuous integration, continuous delivery, and collaboration.
- **DevSecOps:** The DevOps model extends to include security teams, integrating security practices into the development process, ensuring security is a fundamental aspect of the lifecycle.

The DevOps LifeCycle



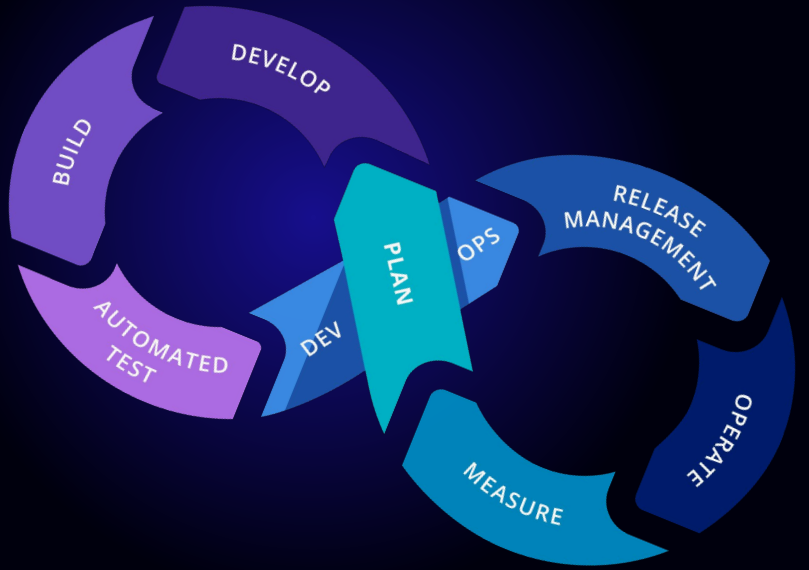
PLAN

- In the Plan stage involves defining project goals, gathering requirements, and planning the features to be developed.
- Collaborating with stakeholders to understand needs.
- Creating user stories and defining acceptance criteria.
- Prioritising tasks and planning sprints (if using Agile methodologies).



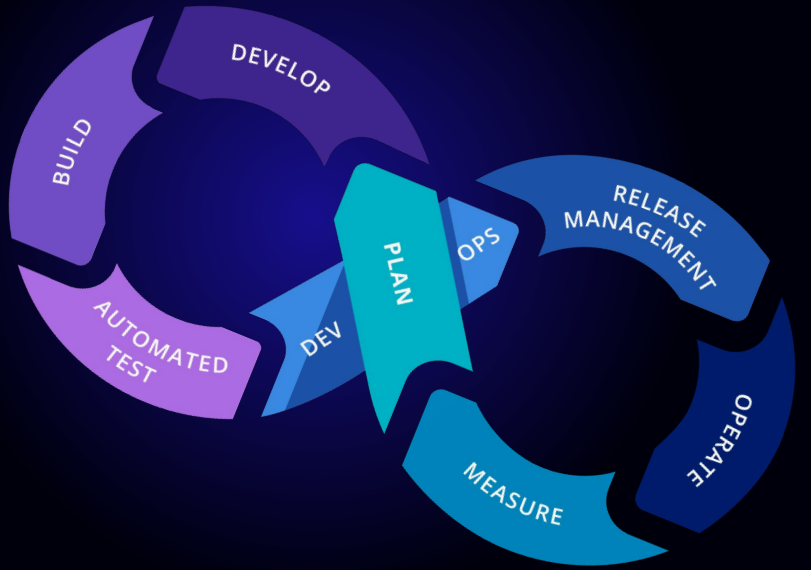
DEVELOP

- In Develop phase, developers write and review code based on the requirements outlined in the planning stage.
- Coding and implementing features.
- Conducting code reviews and utilising version control systems (e.g., Git).
- Maintaining coding standards and best practices.



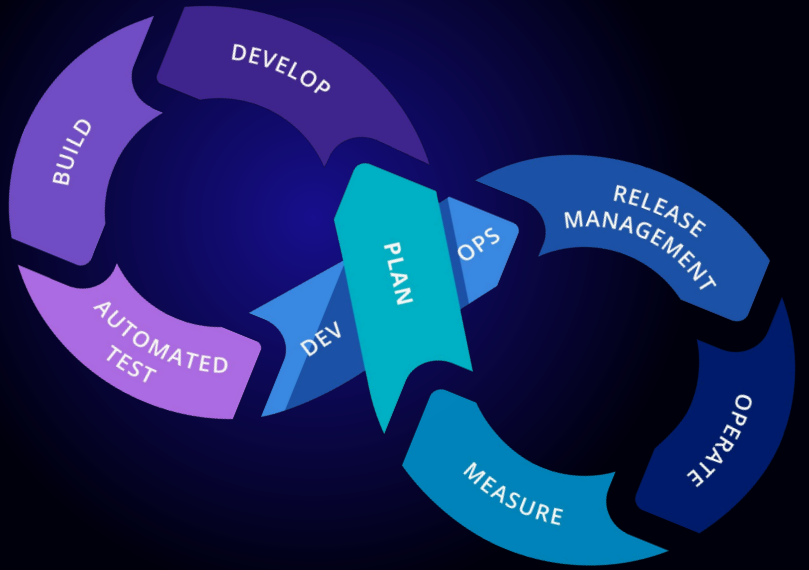
BUILD

- The Build step involves compiling the code and creating a deployable artifact (e.g., binaries, container images).
- Automated builds triggered by code commits.
- Continuous integration processes that compile and package code.
- Running initial tests to verify that the build is functional.



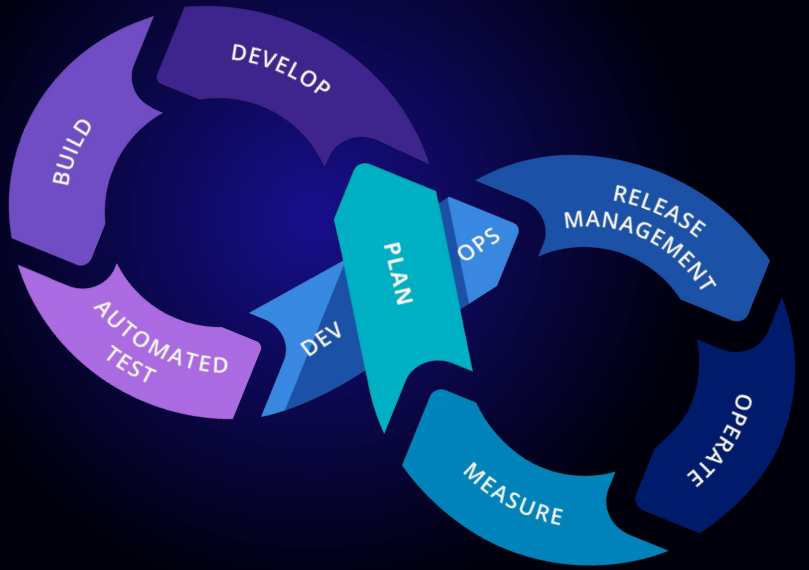
TEST

- In the Test phase, automated and manual testing is conducted to identify bugs and ensure that the software meets quality standards.
- Running unit tests, integration tests, and end-to-end tests.
- Performing regression testing to ensure new changes do not break existing functionality.
- Using test automation tools to streamline the testing process



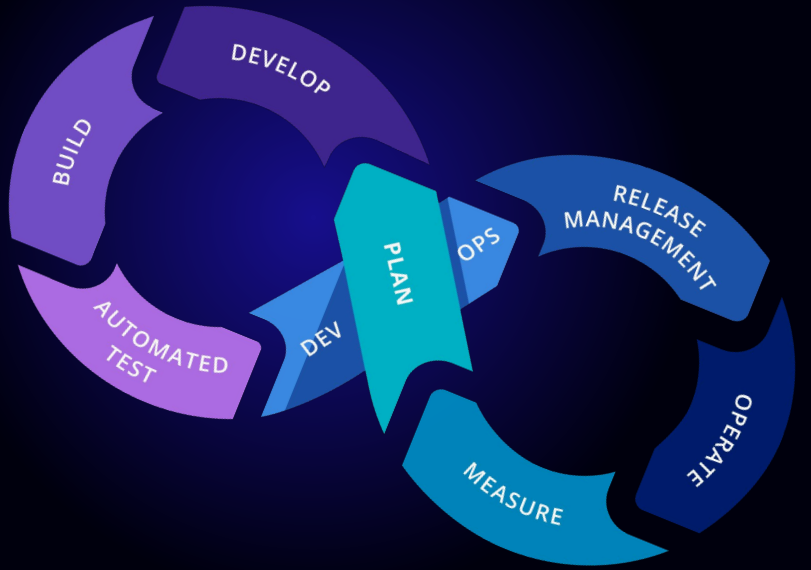
RELEASE

- The Release stage prepares the software for deployment, ensuring that it is ready to be released to production.
- Packaging the build artifacts for deployment.
- Conducting release readiness reviews.
- Managing release notes and documentation.



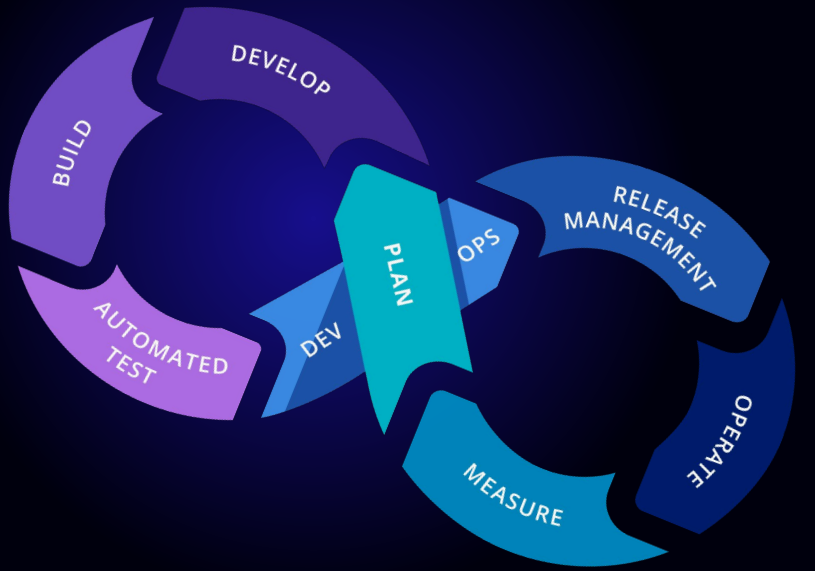
DEPLOY

- The software is deployed to production or a staging environment, making it available to users.
- Automating deployment processes using CI/CD tools (e.g., Jenkins, GitHub Actions).
- Monitoring the deployment for any issues. Rolling back deployments if necessary.



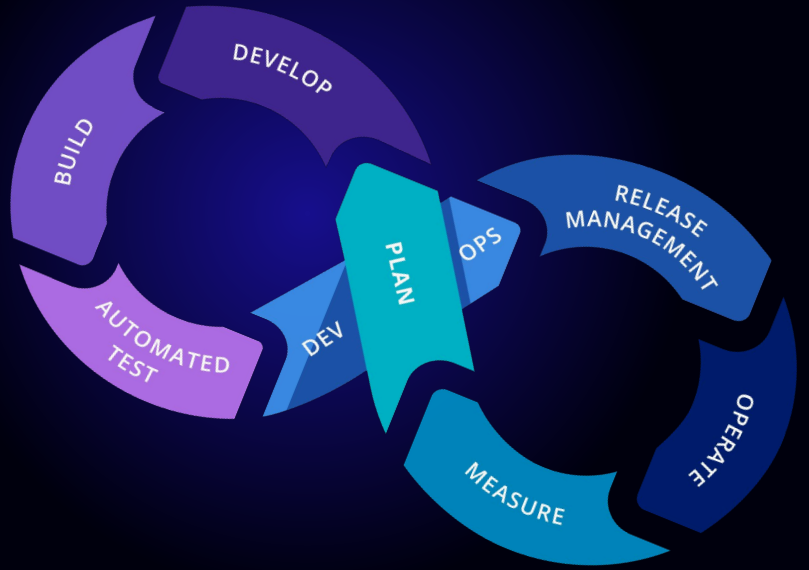
OPERATE

- In the OPERATE (MONITOR) phase, the deployed application is monitored and maintained to ensure optimal performance and reliability
- Monitoring application performance, availability, and infrastructure health.
- Managing incident responses and resolving issues in real-time.
- Implementing updates and patches as needed.



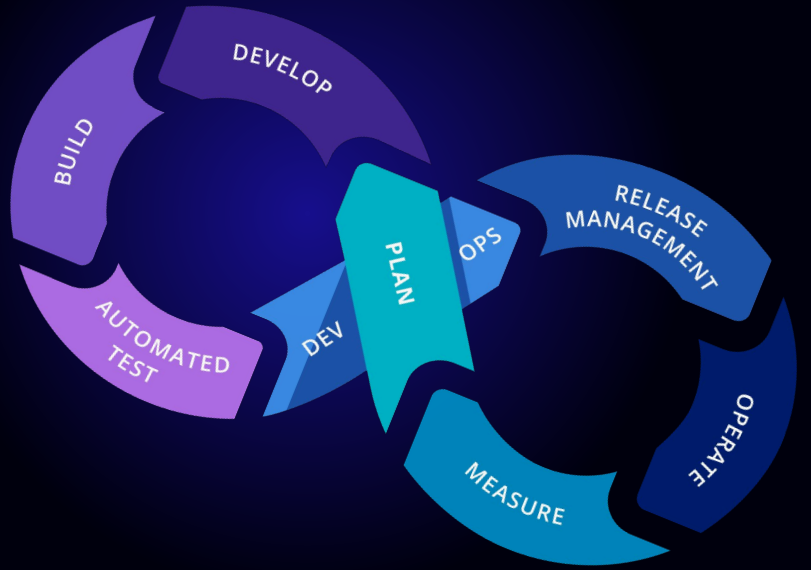
MEASURE

- The Measure (Monitor) phase focuses on gathering metrics and feedback from the application and its users to inform future improvements.
- Using monitoring tools (e.g., Prometheus, Grafana) to track key performance indicators (KPIs).
- Analysing logs and user feedback for insights.
- Conducting post-mortems and retrospectives to identify areas for improvement.



FEEDBACK

- The Feedback step emphasises continuous improvement based on the insights gained from monitoring and user feedback.
- Gathering feedback from users and stakeholders.
- Iterating on features and processes based on the feedback received.
- Planning for future development cycles.



THANKS!

Do you have any questions?

bessmagsm@gmail.com