BeCode Charleroi

20 août 2024
par Lyne Kapongo

From SDLC

to Deployment : Strategies, Tools, and Best Practices"

↘

+ trailer about the workshop Maryam and I are conducting on CI/CD (or not)

summary

of this tech talk

Introduction

01 classic, it’s an introduction.

phases of the waterfall model

02 (there’s 6 phases to be exact)

deployment strategies

03 why and when to use them

best deployment practices

04 for most of them, you already are familiar to it

CI/CD pipelines

05 trailer about de upcoming workshop

intro

SDLC

The Software Development Life Cycle (SDLC) is a systematic process used by software developers to design, develop, test, and deploy software applications. It provides a structured approach to software development, ensuring that the software meets the specified requirements, is of high quality, and is delivered within time and budget constraints.

phases

of the waterfall model

to understand what the software needs to achieve.

1

requirements

This phase concentrates on communicating with the users/end users to gather the requirements and to capture information regarding a user's needs. The product manager, at this stage, defines and documents the scope of the project in a document called a business case.

phases

of the waterfall model

to plan the architechture of the project.

2

design

A business analyst evaluates the business case and starts the logical design of the software by using the requirements and information collected by the product manager. Based on the high-level design created by the business analyst, a system analyst translates the high-level design to a detailed low-level design that considers software and hardware technology.

phases

of the waterfall model

to write the code that forms the application.

3

implementation

Here, the actual code of the software system is written. Software developers create the system according to the instruction and requirements recorded, written, and prepared in the design and requirement phases. The output of this phase is the actual product.

phases

of the waterfall model

To ensure the software is free of bugs and meets the requirements.

4

testing

This stage gets the input from the implementation stage. Software testers draft test plans based on the functional specification documented in the low-level design document (LLDD). On the other hand, software developers prepare testing plans in the form of a checklist to examine if every function is executable as expected.

phases

of the waterfall model

to move the software to a production environment where users can interact with it.

5

deployment

After passing all processes of the testing phase, the product is ready to release. The software system is either released for users to install on their own machine or deployed to production servers.

phases

of the waterfall model

Regular updates and fixes to ensure the software continues to function properly.

6

maintenance

This phase focuses on enhancements, delivering changes, or fixing any defects and issues that may arise.

strategies

of deployment

1

Recreate Deployment:

Recreate Deployment involves completely replacing the old version of the software with the new one all at once. This method is simple but causes downtime since the system is offline while the update is happening. It works well if a short service interruption is acceptable.

2

Rolling Deployment

Rolling Deployment updates the software gradually by swapping out the old version with the new one in stages. This approach keeps the system running with minimal downtime, but it requires careful monitoring to make sure everything stays stable throughout the update process.

3

Blue-Green Deployment

Blue-Green Deployment uses two identical environments: one (Blue) runs the current version, while the other (Green) is used for the new version. Once the new version is tested and ready, traffic is switched from Blue to Green. This method offers zero downtime and makes it easy to roll back if something goes wrong.

4

Canary Deployment

Canary Deployment releases the new version to a small group of users first before rolling it out to everyone. This way, you can catch any issues early on with just a few users before the full release. It’s a good way to test and ensure the new version works well on a smaller scale.

best practices

Aa automate to avoid human error and for better consistency

Bb version control for traceabillity and an easy rollback if necessary

Cc a+b=c catch issues early and implement strategies and tools so the recovery can be quick