DevOps Data

Introduction

A word about me

- Bachir Aït M'Barek
- Senior Data Engineer (Univalence)
- Worked on Data projects for more than 10 years as a consultant
- Experience with teaching

A word about the Course and the Exam

Composition:

- 60% Course + TPs
- 40% Group Project

Grading:

- 50% Project
- 50% Written Exam (2 hours)

In the early 2010s…

Agile software development

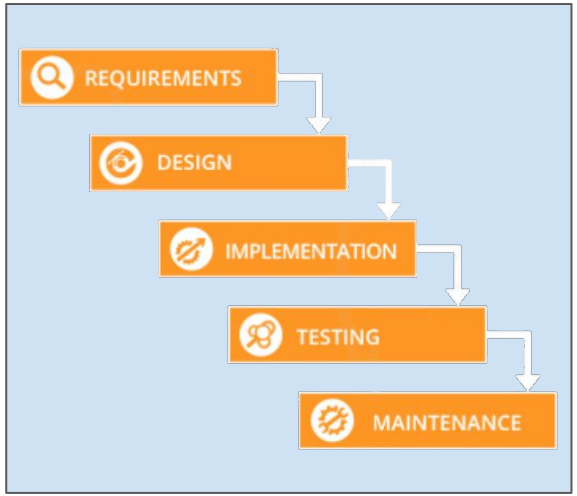
Agile Software development approach breaks with the legacy long cycle approach (waterfall).

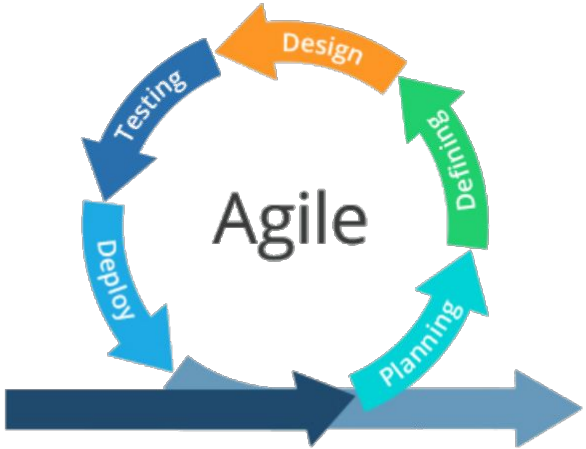
Tends to shorten the application / feature’s delivery by working on quick and regular iterations instead of batches.

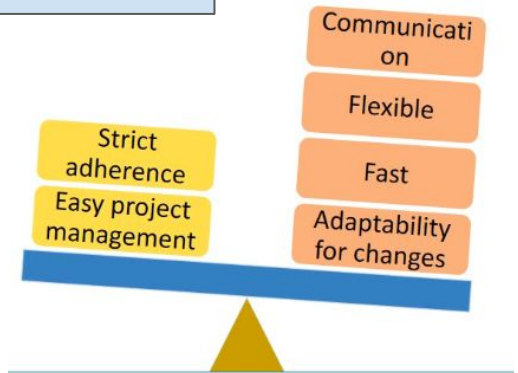
This way, the team has quicker feedbacks from end-users and a high adaptability to change.

The business also benefits from this approach, with a short time-to-market and the possibility of re-prioritizing the features.

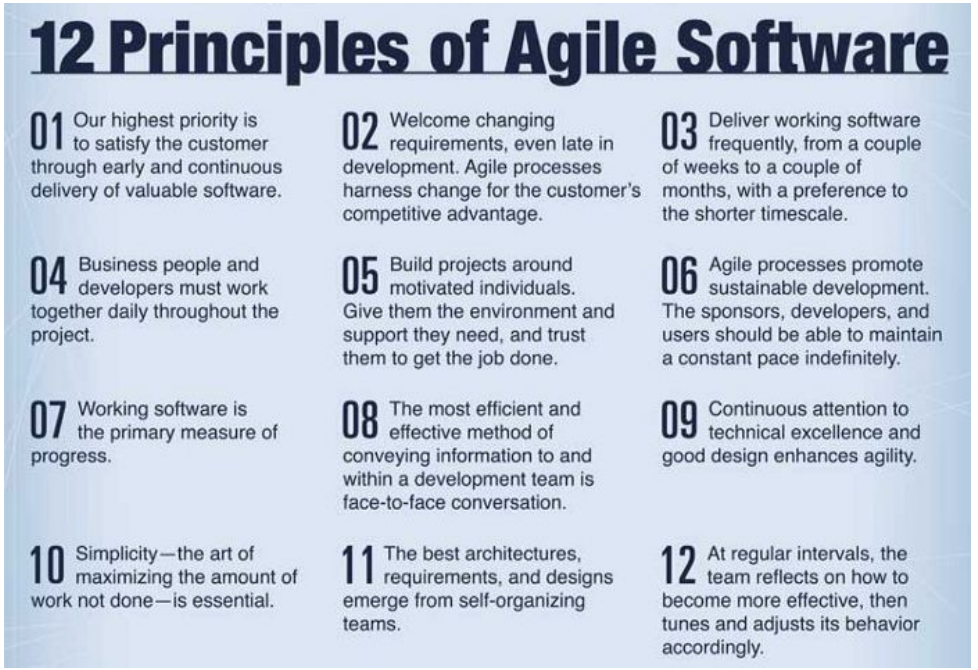
Agile software development



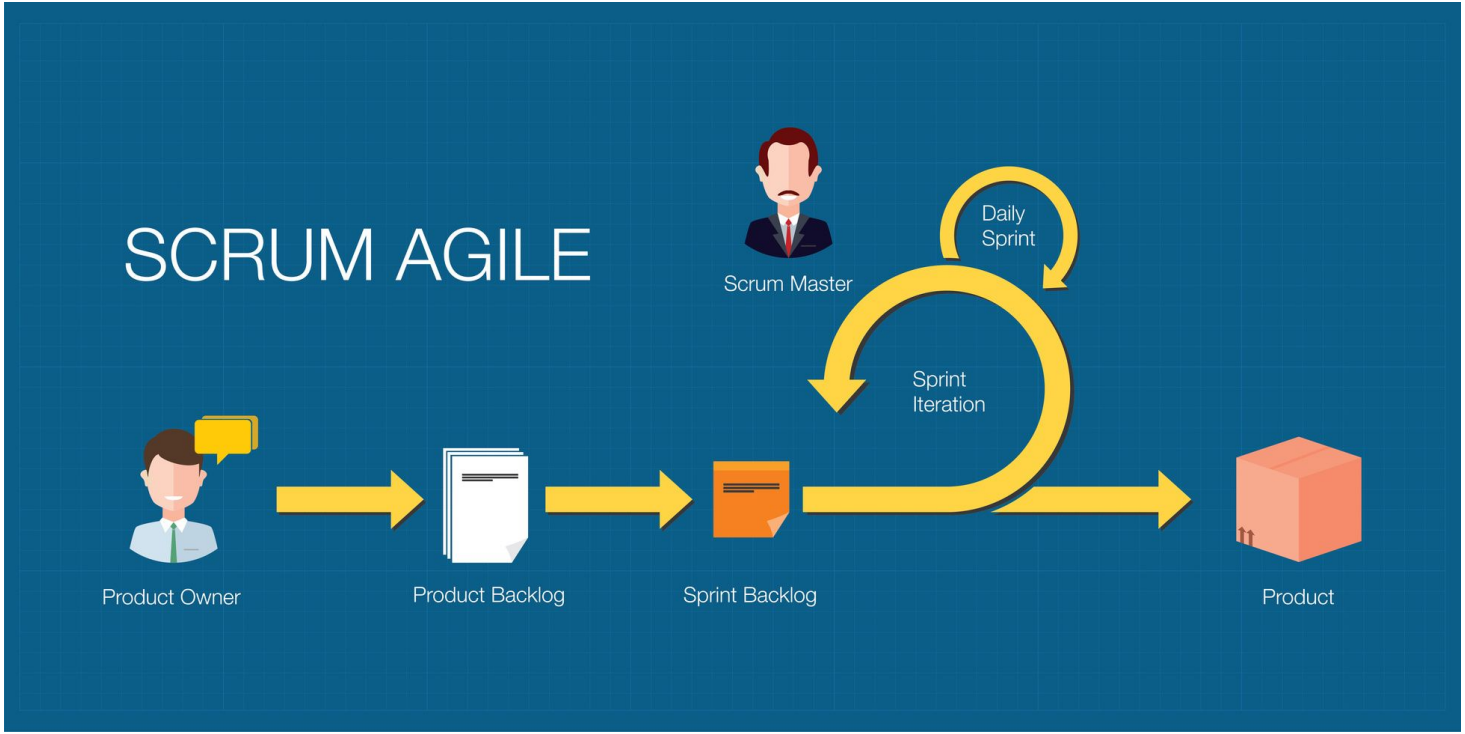




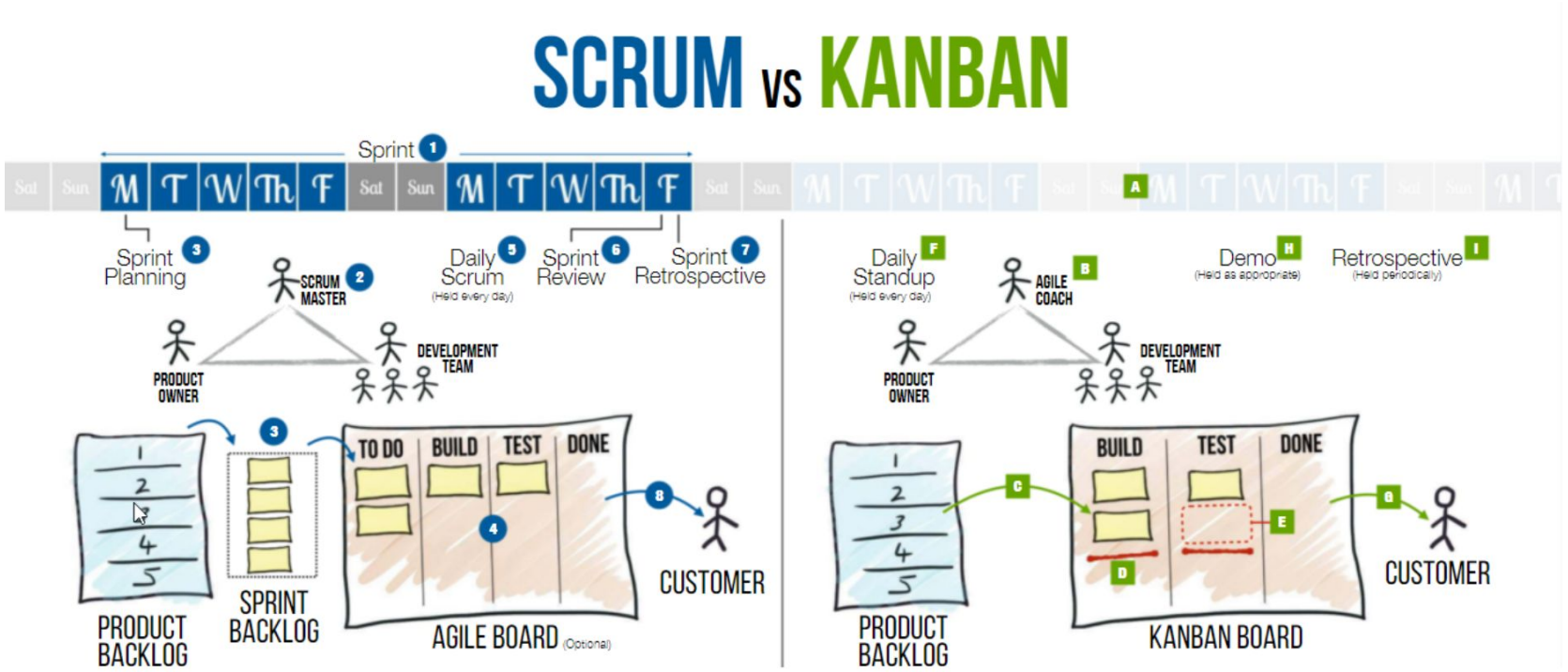
Agile software development



Agile software development



Agile software development



Agile software development

SCRUM vs KANBAN

- Work accomplished by increments. Iterative approach.
https://scrumguides.org/scrum-guide.html

Key concepts & characteristics:
- Rituals & Increments
- Self Managing Team / Autonomous Team
- Transparency, Inspection, Adaptation

When should one use it ?
- Ideal for long projects for which we need recurrent feedback
- Product Vision
- Tasks are clear

- Japanese expression for “visual signal”.
- Relies on notes we drag on a white dashboard.

Key concepts & characteristics:
- Visual Work
- Continuous Flow
- Limit work in Progress
- Transparency

When should one use it ?
- Work arrives in an unpredictable way
- Strive to deploy work as soon as it is ready

A bit of history

Developing and Operating an application doesn’t require the same knowledge / skills.

Mixing Dev and Ops tasks is **not efficient** and therefor costs increase.

Separated roles with **distinct responsibilities**:

- Developers focus on **Features**
- Operations focus on **Stability**

A bit of history

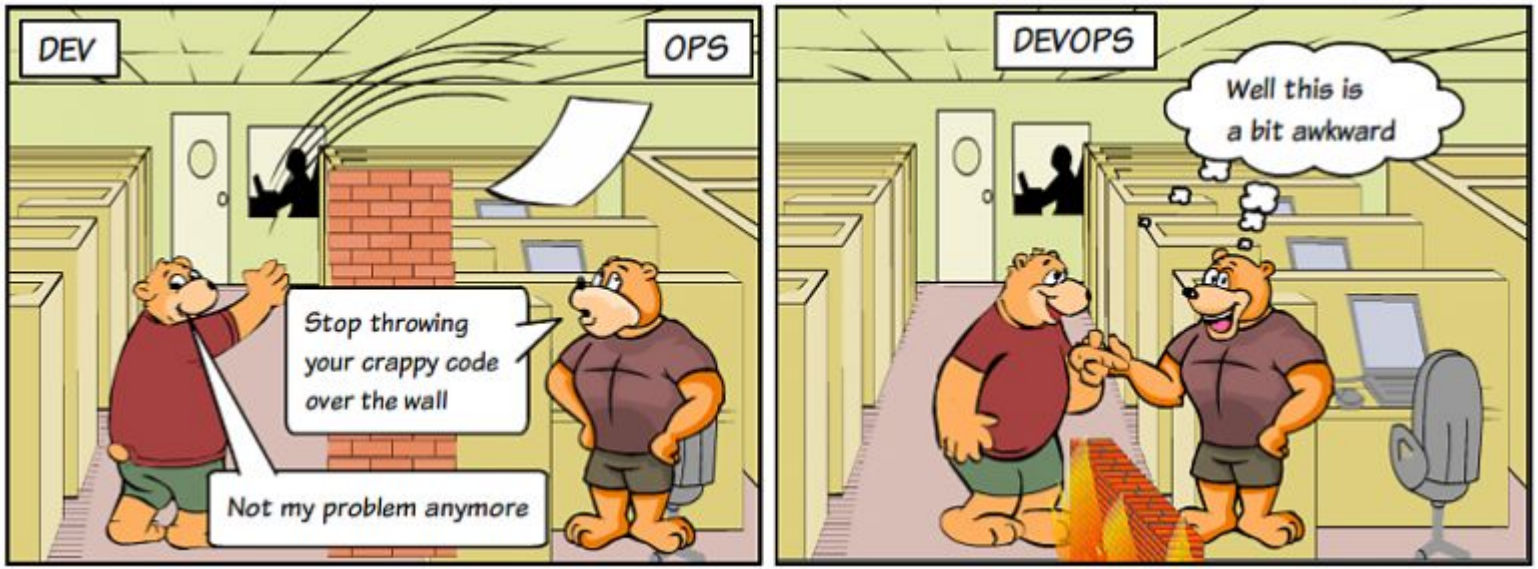
These two objectives almost always lead to **conflicts** with a lot of **confusion**.

Many side effects:

- Disagreements about deliverables : “It works on my computer”
- Some untested features are deployed directly in Production
- ...

The **DevOps culture** was introduced so both teams benefit from a **collaborative** approach.

A bit of history



We also want to avoid …

Fancy Deliveries

We also want to avoid …

Human intervention

We also want to avoid …

Delivering bugs & regression

Lessons learned

- Bugs & Regressions (often) released in Production.
- Impossible to reproduce behaviors.
- Each team accuses the other.
- Operating mode specific to each deployment. Manual errors become habits.
- Long delays to solve issues.

Consequences

- Operational issues are catastrophic for digital companies
  - Lost business : Customers may be tempted to go to the competitors
  - Bad message for customers / users (product quality, rigor, ...)

What is “DevOps” ?

DevOps - Definition

DevOps is a practice of bringing together two worlds closely: Software Development and Operations.

This helps to deliver software components faster and more reliably by defining and implementing processes.

It is a fundamental component for business agility.

DevOps - Objectives

- Deliver faster and more reliably (CI / CD)
- Infrastructure provisioning (Infrastructure As Code)
- Security compliance
- Monitoring and detect incidents (alerting systems)
- Automating recurrent transverse work (ie: delete obsolete logs / packages)

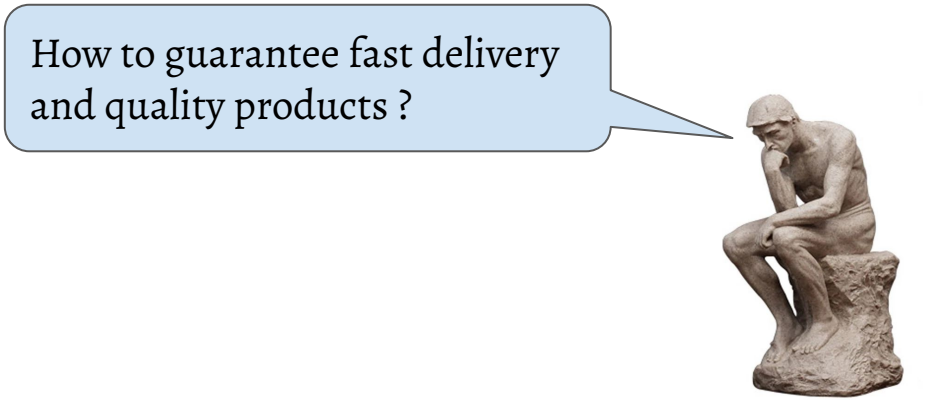
DevOps - Prerequisites

DevOps culture needs sound foundations to succeed.
The team needs to share **common values**:

- **Hiding** is considered harmful.
- We don’t need **heroes**, we need cohesion.
- Projects and team come before the individual.
- Being autonomous doesn’t mean working alone.
- Definition of “**done**”.
- **Healthy** social interaction.
  - Proactively share information
  - Create dialog, even in the event of disagreements

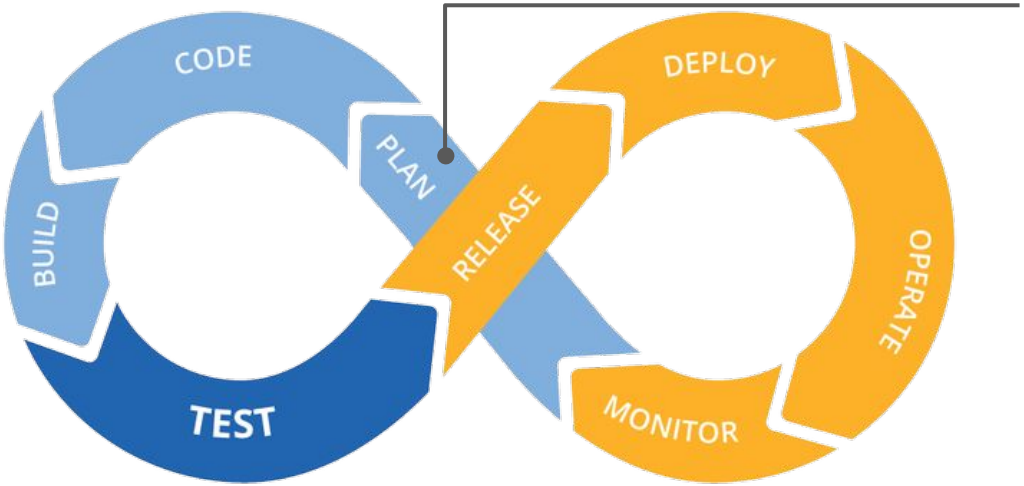
The DevOps Cycle

The DevOps culture in enterprise is materialized through the implementation of a **rigorous process** to make sure all the standards are taken into account before running the application in Production (ie: QA, Performance, Security, …).



This process should be specific to suite each enterprise / team's needs.

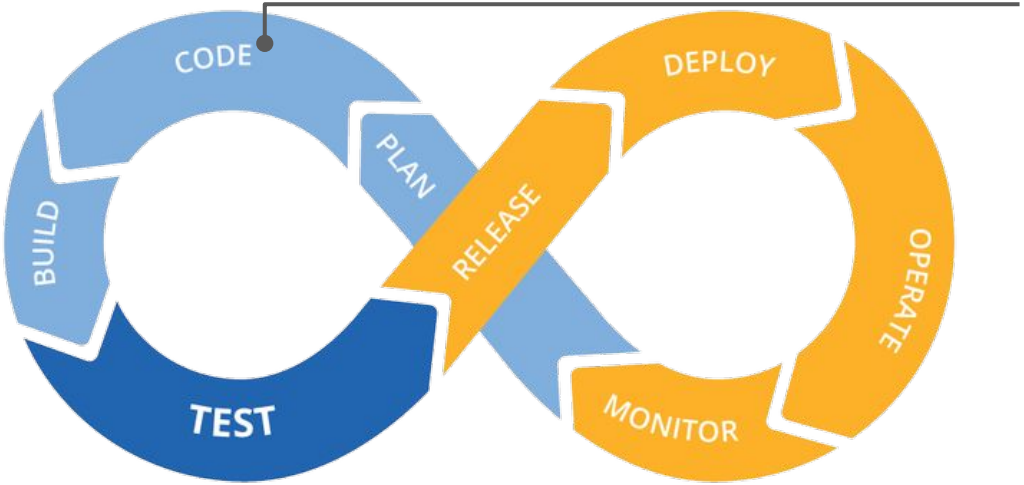
The DevOps Cycle



Plan : Refers to Agile Software Development Management.

Consists of creating and maintaining the Backlog, tracking bugs, prioritize tasks, etc...

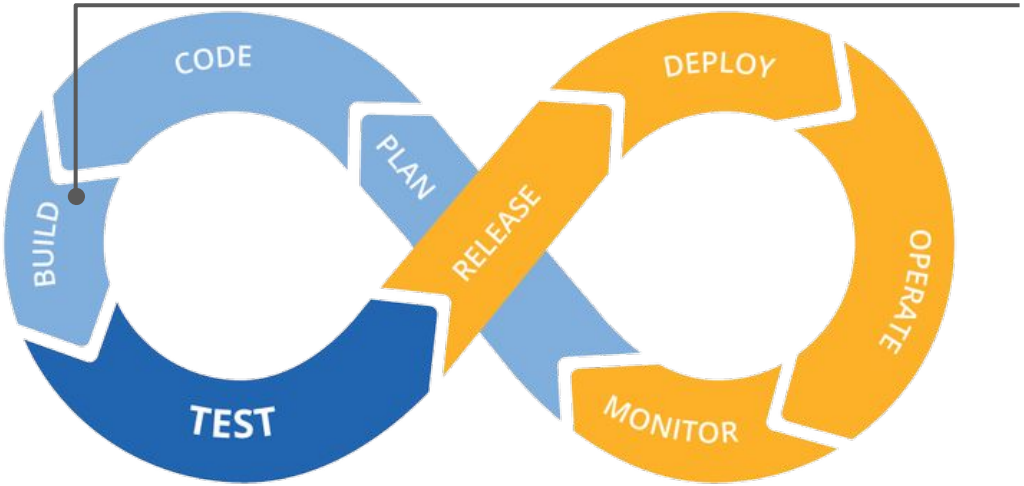
The DevOps Cycle



Code : Implementation.

Write the necessary modules / functions / instructions required by the User Story.

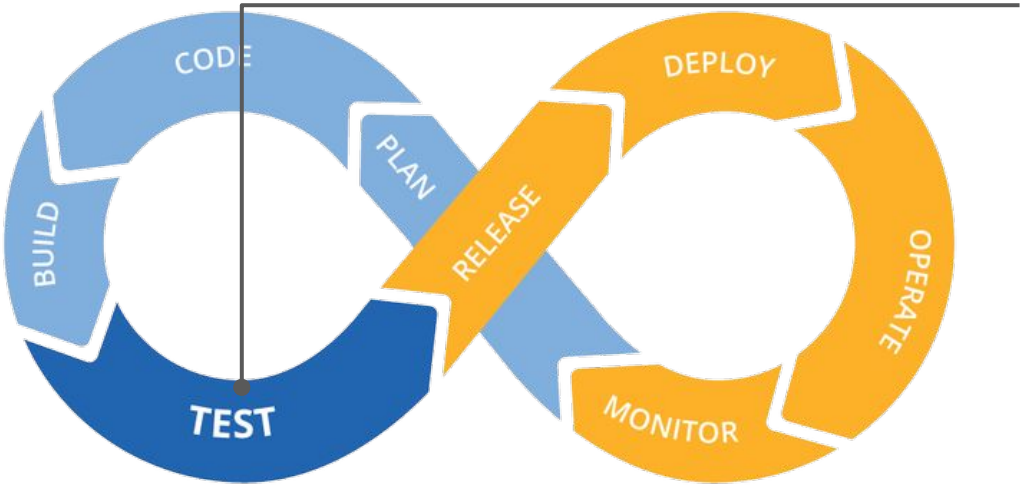
The DevOps Cycle



Build : Integration Process.

Compile / Package the application.

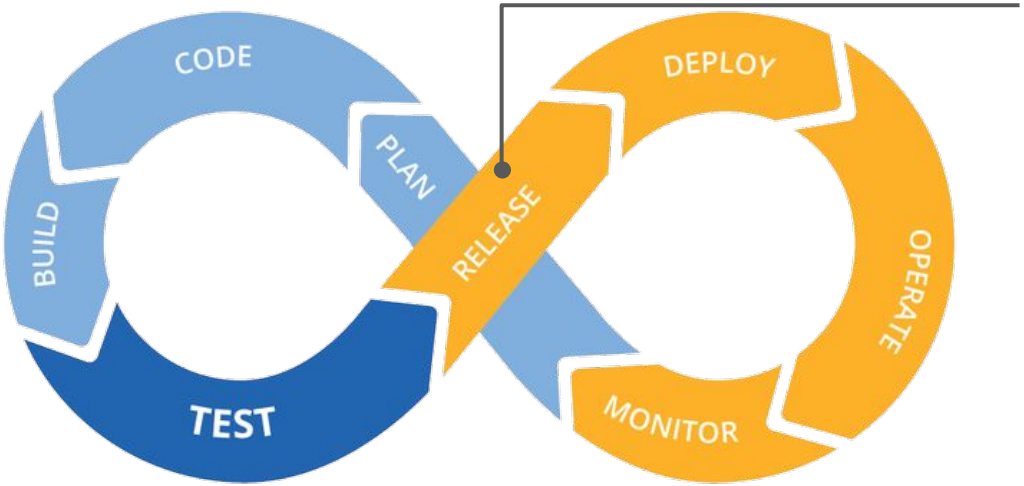
The DevOps Cycle



Test : Automatically run checks.

- Non-regression tests
- Unit tests
- Integration tests
- Load & Stress tests
- Static code analysis

The DevOps Cycle



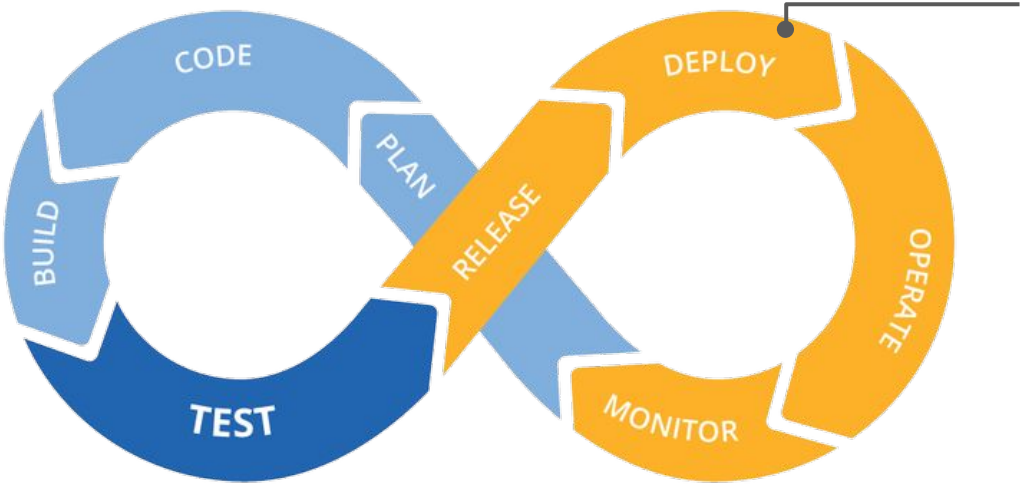
Release: Publish the artifacts on a repository.

Once the application is packaged and tested, it should be versioned and saved in a repository so it can be deployed when desired.

The static code analysis report can also be saved.

This phase must also include a Code Review step.

The DevOps Cycle

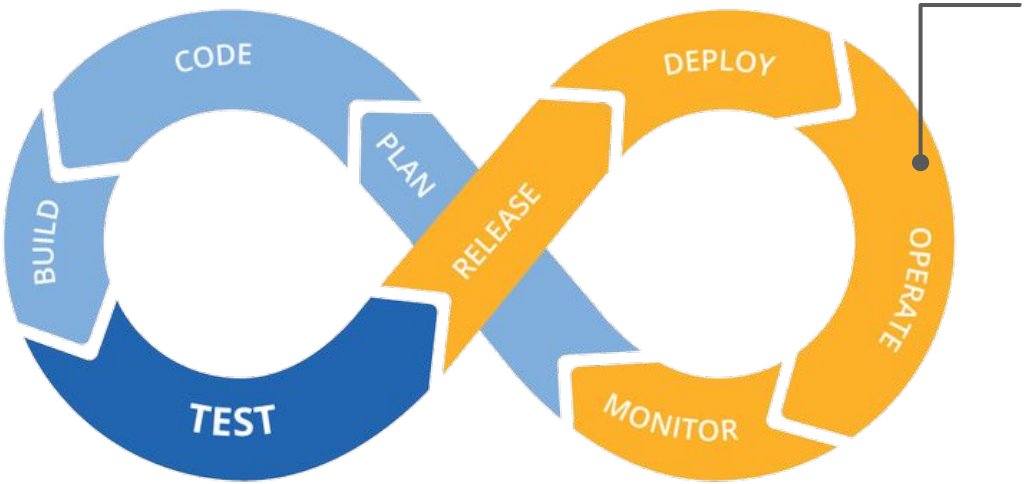


Deploy: Bring a new version of the application to the targeted environment.

Prepare the environment (provisioning), send the application's components to the expected hosts / directories, apply the required rights, ...

Should be triggered manually and explicitly.

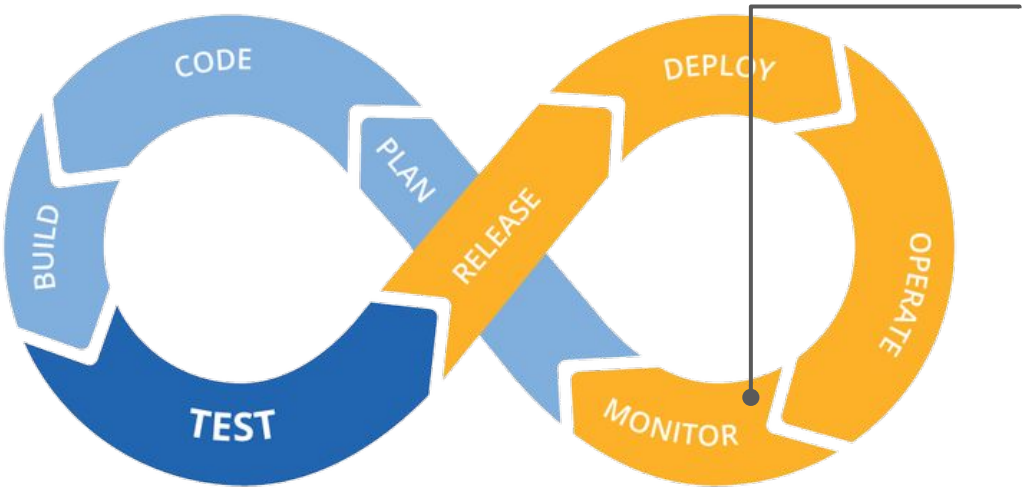
The DevOps Cycle



Operate: Run the application.

Schedule / Orchestrate the application and make it (or its results) available to third-party users.

The DevOps Cycle



Monitor: Make sure the application behaves as expected.

Collect logs & metrics.
Create Dashboards.
Anticipate / Detect incidents or performance issues.

Plan potential technical improvements.

Benefits

Collaboration and visibility across teams - Same end-to-end process - Same objectives - Fluent communication - Clear status	Shorter and reliable release cycles - Delivering increments instead of monoliths - Process matures iteration after iteration
Continuous improvement - Software faults (performance, bugs) are logged in the backlog	Unified Process - Each application adapts to the same process. - The process can be designed once and its components may be reused as many times as necessary - Reduce the risk of human error