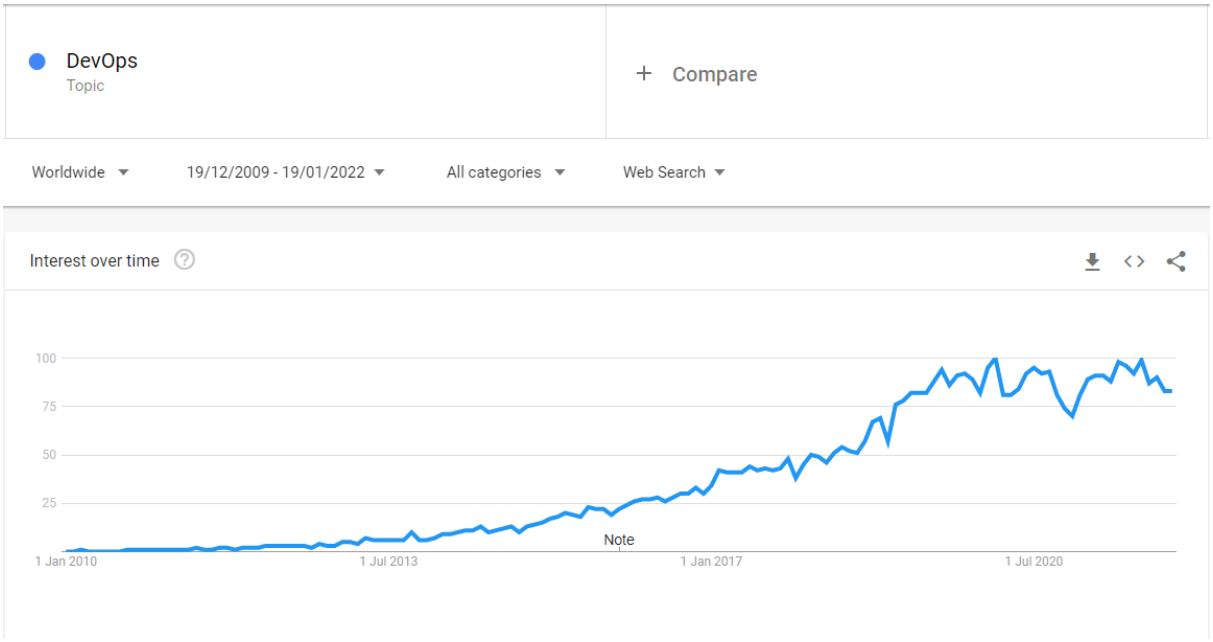
Introduction to DevOps

Be DeOps by NTT Data

DevOps core concepts

- DevOps is a movement based on a set of practices and tools that bring together *software development* and IT operations,
- Dev Team,
- Ops Team,

- **Developers** want to install new versions and **Operations** want stability in the environments,
- As a result, we have a conflict of individual interests that is ultimately reflected in the workflow and higher turn around times. This is known as a **wall of confusion.**
- **Developers:** They want to make changes, They want to deliver product
- **Operations:** They want stability, They want to provide good client service
- **Summarising:** DevOps removes the wall of confusion by creating synergy and common purpose as a team, between the areas of software development (Dev) and IT operations (Ops). Imagine a perfect world:
- Growth of DevOps topic:



DevOps accompanies agility so that your processes are not too manual and you can:

- Deliver incremental products with some frequency
- Get continuous feedback
- Eliminate impediments between teams

Cloud

- New opportunities and inititiaves are already cloud-oriented.

- Today, any company has the resources to be able to build cloud services, and initial investments have been reduced.

Trend

- Large companies today are facing small companies (Start-Ups), which are born being Cloud, Agile and DevOps oriented.
- Therefore, the adoption of DevOps practices and tools becomes the main lever of their continuous value delivery strategies.

DevOps Business Value

- Reduction of time-to-market

DevOps values

CALMS

- **CALMS** is a *framework* that assesses a company's ability to adopt **DevOps** processes, as well as a way to measure success during a **DevOps** transformation.
- The acronym was coined by Jez Humble, co-author of "*The DevOps Handbook*", and stands for: ***Culture***, ***Automation***, ***Lean***, ***Measurement*** and ***Sharing***, representing the DevOps values**.**

What DevOps is NOT?

It is very common to misunderstand the concept of **DevOps** and to attribute to it definitions that have nothing to do with it. Let's see what **DevOps is definitely NOT**.

- It is neither a title nor a role.
- It is not having separate teams.
- It's not just automating for the sake of it.
- It should not be the core of a business strategy.
- It is not simply a set of tools.

→ Automation is the consequence of making processes more efficient and reducing risks, but it is a tool, not the essence.

→ Strategy: It is a concept to generate products and services more efficiently, but it should not set the strategy of organisations; it is not the what, it is the how.

The Phoenix Project

"*The Phoenix Project*" by Gene Kim, Kevin Behr and George Spafford is a very interesting introductory book to **DevOps**. The book is written in a business parable style and follows Bill, an IT manager, as he tries to deal with different IT problems in his company.

Throughout the book, Bill relies heavily on a mysterious **DevOps** and *Lean* guru named Erik and their conversations help Bill learn the importance of **DevOps** and optimising workflow.

An important point from "*The Phoenix Project*", and more specifically from Bill and Erik's conversations, is the realisation that all **DevOps** practices can be reduced to three principles: **The Three Ways**. Let us look at these principles in more detail.

Ways of DevOps implementation

The 1st way: flow

The 2nd way: feedback

The second way **seeks continuous feedback early and in a timely manner** at each stage, to quickly identify what we can improve. This is achieved by:

1. Inspecting processes to validate whether they actually deliver value to the client.
2. Adjusting processes according to client feedback and their level of satisfaction.

The 3rd way: CE&L

Chaos engineering

"Chaos Engineering" is about experimenting with a software system by introducing faults to increase the level of resilience and robustness, building confidence in the system's ability to withstand unexpected conditions.

What is its origin

Netflix was the first to introduce the concept of ***Chaos monkey***, which was so named for creating destruction like a wild monkey and armed to provoke failure. Thus, ***Chaos Monkey*** gave birth to the new engineering practice ***Chaos Engineering***.

Chaos engineering principles:

1. Start by defining a **steady state** as a **measurable output** of a system that indicates the **normal behaviour**.
2. **Hypothesise** that this steady state will continue in both the **control group** and the **experimental group**.
3. Introduce **variables** that reflect **real world events**, such as servers that stop working, hard disks that malfunction, network connections that are down, etc.
4. Try to **refute the hypothesis** by looking for a difference in steady state between the control group and the experimental group.
5. Run the experiments in **production**.
6. Ensure that the **consequences** of the experiments are **minimised** and **contained**.
7. Automate experiments to run **continuously.**

Benefits:

At the business level: Chaos engineering helps stop large revenue losses by avoiding prolonged outages.

At the technical level: The team gains a **greater understanding** of system dependencies, allowing them to create a more robust system design.

At the client level: Improved **availability** and **durability** of the service are the two main client benefits of chaos engineering.

Overall: Enhances system recoveries, strengthens resilience and increases security.

Continuous lintegration

Continuous integration is really **a large set of practices** that focus specifically on **software construction**.

What is it?

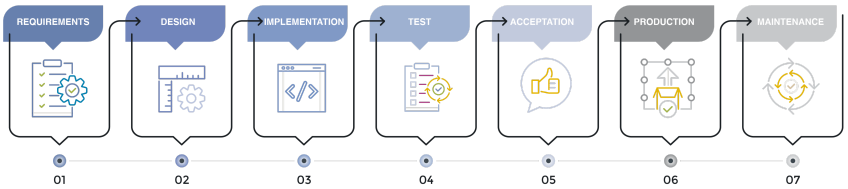
It is a practice based on **automated execution** of certain **processes** after each addition of changes to a software repository.

A clear example is that when a team member commits on version control (git for example), the project is compiled and all its unit tests and *code analysis* are executed. All this is done automatically.

Continuos testing

The process of running automated tests as part of software delivery to obtain *feedback* on the business risks associated with a software release candidate as quickly as possible.

Continuous Testing seeks to bring the development **quality assurance** team closer to the **development** team, doing testing cross-functionally rather than at the end of the development process.

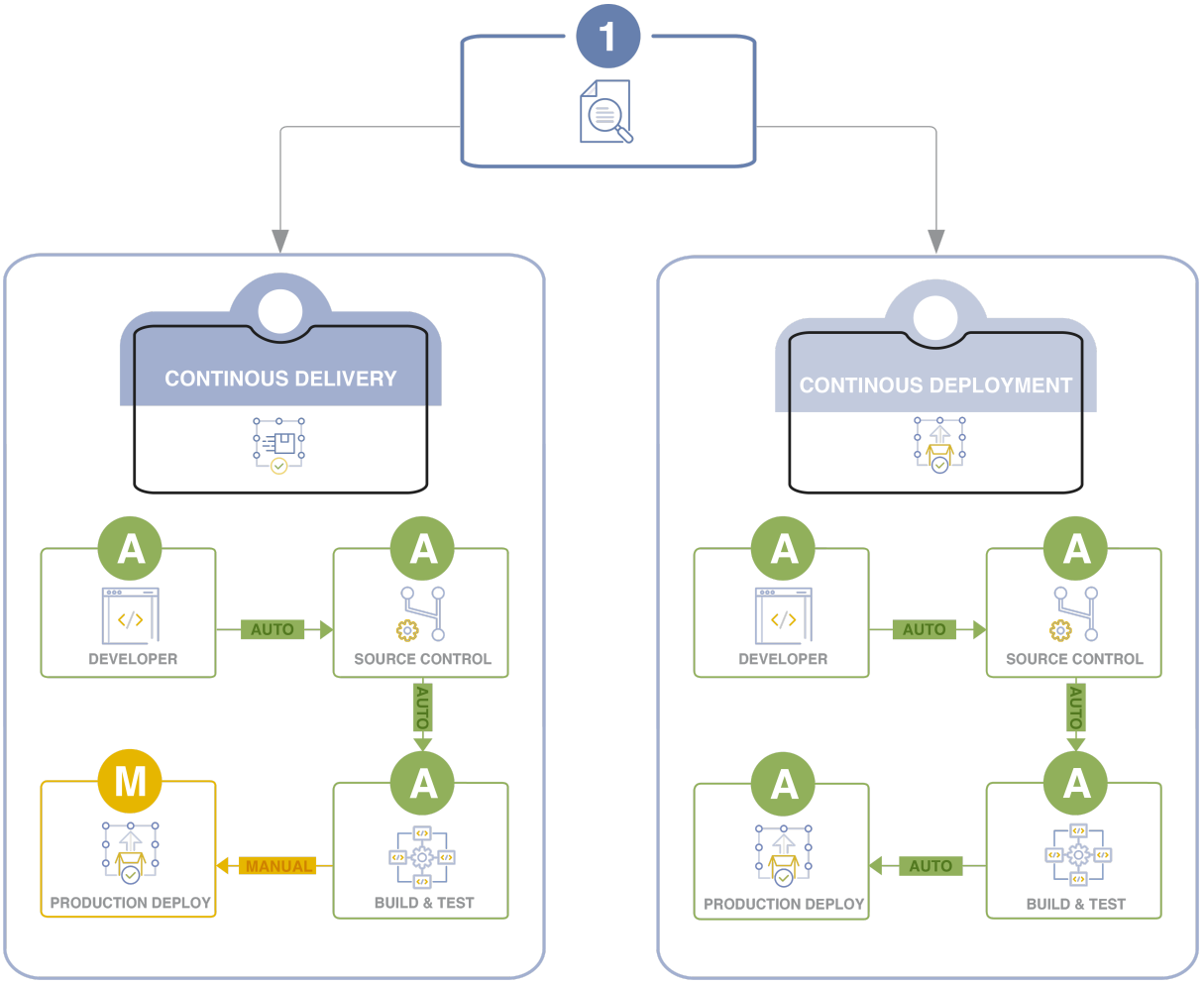


Some important concepts of Continous Testing:

- The fact of testing in early stages of software development is known as ***Shift left Testing***.
- The increase in frequency of testing in *continuous testing* is due to the implementation of **automation.**
- Types of Testing: **functional** and **non-functional**.

Continuous Delivery & Deployment

- They focus on creating secure and always potentially deliverable software throughout the entire lifecycle.
- Impacts: Artifact construction, quality assurance, publication in repository and manual or automatic deployment.
- Importance: Because it automates and enhances processes such as: quality control, security, detection of errors in early stages of development, the possibility of deploying any development as soon as possible and receiving continuous feedback from the client.
- You need to have Continuous Integration implemented, have a repository of artefacts for publication (Nexus, Artifactory) and an enabled strategy to automate deployments.



DevSecOps

- **Security** should be a **shared responsibility** and integrated throughout DevOps. Because it is such an important focus, it led to the term **DevSecOps**.

ChatOps

- **Chatbots** are Artificial Intelligence capabilities that allow holding a conversation via text or audio. If we put together client chats + chatbots, we will have chats related to **development**, **delivery** and **support**.
- **ChatOps** allow the collaboration tools we use in our day-to-day work to be complemented by tools we use for DevOps, so we can execute tasks like creating containers or running pipelines with just a click of a button.

- Advantages: **Integration** of multiple tools on the same interface, simplicity and SPOC (***Single Point Of Contact***) interaction with cognitive capabilities.

Agile

- **Agile** is the ability to create and respond to **change** using a number of *frameworks* and practices, such as *Scrum*, **SAFe**, **Kanban** and **Lean**, which are supported on some of its **values and principles**.

Scrum

- **Scrum** is a simple framework that promotes collaboration in teams to achieve the development of complex requirements and increase the ability to more frequently release the final product.
- On the surface it is simple, but difficult to master.

Kanban

- **Kanban** is a method of working based on demand, not forecasting. It was created at Toyota in the 1950s as a necessary improvement plan in car production.
- It promotes team collaboration to optimise workflow, and allows the speed of teams to be measured.
- Kanban's main features are: Reducing downtime, minimising waste in processes and visualisation of work in each state, especially in terms of progress.

Lean Management

→ **DevOps** is also complemented by the Lean framework, which focuses on building products the right way, reducing **waste** and poor quality.

→ ***Lean*** is an approach to translating the concept of **continuous improvement to the business world;** its objectives include:

- Reduce costs.
- Improve processes.
- Increase quality.
- Decrease delivery time

- **The most notable principle in Lean is eliminating waste;**
- **Only by eliminating waste is productivity improvement really noticeable.**

IT Frameworks

These frameworks allow IT services to be properly managed, taking into account the service lifecycle, and aligned to agile practices.

IT Service Management

What is it?

- It is the management of quality in IT services aligned with business needs.

Objective

- To establish guidelines and structures in processes such as: Change, configuration, release, incident and problem management.

Scope

- ITSM processes cover the entire lifecycle, from strategy, design, transition, operation, continuous improvement and value creation.

DevOps implementation

- DevOps draws on ITSM practices to achieve the goal of deploying faster without causing service disruption.

Agile Service Managment

Description

- Ensures that ITSM processes are aligned with agile values, and with resource efficiencies, meeting client needs.

Method

- Breaks down service management into small increments.

Objective

- To ensure that ITSM processes reflect all agile values through continuous service improvement.

Optimisation

- Increase efficiency in speed and satisfaction.

DevOps culture characteristics

- The culture must always be **a shared vision**, so that the overall corporate goals can be known, in order to perform good quality assurance of the product or service.
- The **DevOps culture** is based on **Agile** concepts such as: Two-way communication, collaboration, respect, transparency, no silos, trust, continuous inspection, adaptability and flexibility.
- It is always essential to **encourage continuous improvement**, based on experimentation, intelligent risk-taking, and learning, practising and sharing.

DORA State of DevOps

DORA is characterised by:

- Providing a **comprehensive view** of the DevOps landscape.
- Providing a **practical guide** for organisations of all sizes and in all industries to improve software delivery performance.
- Google acquired **DORA** in December 2018, and it is now **part of the Google Cloud** group.

> DORA-state-of-devops-2021.pdf

Questions

Which of the following terms **IS NOT** related to DevOps?

- CI/CD
- DevSecOps
- AC/DC ✅
- Cloud

Which of the following statements make DevOps a technique for software development and delivery?

- DevOps makes it easy to identify, correct and learn from mistakes.

- DevOps facilitates the creation of software with high quality and monitoring.
- DevOps gets the team organised and aligned with the business objective.

Which of the following practices **IS NOT** required in the DevOps lifecycle?

- Continuous rollback

Why is continuous testing important?

- To reduce the risk of possible errors.
- To increase the quality of our development.
- To enable the concept of Shift Left Testing.

Which **IS NOT** a **practice** on which DevOps is based?

- Waterfall

What is DevOps?

- DevOps is a movement based on a set of practices and tools, which eliminates silos between Dev and Ops.

What **IS NOT** a direct benefit of implementing DevOps?

- Reduction of automations

Which of the following is **NOT** a DevOps value?

- Planning

Which of the following statements is **incorrect**?

- DevOps is only feasible in start-up companies.
