DevOps

DevOps is a combination of practices, cultural philosophies, and tools that enhance an organization’s ability to provide services and applications in an expedited manner.

It also enables the improvement and evolution of products at a rapid pace than the organizations which employ traditional development software and infrastructure management processes.

Automation

Automation in DevOps makes the entire process smooth, and all conventional manual software development barriers become scalable and manageable. Furthermore, it helps to efficiently manage and run an application in the development environment with frequent updates. It also needs strict testing standards.

Benefits of Automation.

**Removes Manual Error**

**Shorter** **Development** **Cycle**

**Latency** **Removed**

**Improved** **Defect Detection**

**Principle of DevOps**

Certainly! Here are the principles of DevOps in an easy-to-understand way:

1. **Collaboration:** Imagine everyone in a big team, working together like a well-oiled machine. DevOps encourages teamwork and open communication between developers, operations, and other departments.
2. **Automation:** Picture tasks getting done automatically with the push of a button. DevOps aims to automate repetitive tasks like testing, deployment, and configurations, making things faster and less prone to errors.
3. **Continuous Improvement:** Think of always making things better. DevOps fosters a culture of continuously refining processes, tools, and methodologies to enhance efficiency and quality.
4. **Feedback Loop:** Like a game, you get feedback to play better. DevOps encourages constant monitoring and feedback to identify issues early, leading to faster problem-solving and improvements.
5. **Flexibility and Adaptability:** Think of being able to adapt to any situation. DevOps promotes flexibility in responding to changes, adjusting quickly to new demands and market shifts.
6. **Shared Responsibility:** Imagine everyone taking care of a project together. DevOps encourages shared ownership and responsibility for both successes and failures, fostering a sense of unity and collaboration.

**Container ?**

* + **“*It is a solution to the problem of how to get App/Software to run reliably when moved from one computing environment to another”***
  + ***Examples of discrete environments it can handle - Test Vs Prod; Phy vs Virtual Machines***
  + ***Container = (App + All its dependencies) in a box***
* A container in Docker is a lightweight, portable, and self-sufficient package that includes everything needed to run a software application: code, runtime, system tools, libraries, and settings. Containers are isolated from each other and share the same operating system kernel, allowing for consistent operation across different environments. Docker, a popular containerization platform, uses containers to streamline application development, deployment, and scaling.
* Top of Form
* ***Virtualization***− Virtualization is the technology that can simulate your physical hardware (such as CPU cores, memory, disk) and represent it as a separate machine. It has its own Guest OS, Kernel, process, drivers, etc. Therefore, it is hardware-level virtualization. Most common technology is "VMware" and "Virtual Box".
* Virtual Machine = (Apps + Run Time + OS) orchestrated by a Hypervisor
* ***Containerization***− Containerization is "OS-level virtualization". It doesn't simulate the entire physical machine. It just simulates the OS of your machine. Therefore, multiple applications can share the same OS kernel. Containers play similar roles as virtual machine but without hardware virtualization. Most common container technology is "Docker.
* Container = (Apps + Bins/Libs + Run Time) on a Shared OS managed through a Container Engine

Docker

Docker is a platform that simplifies the process of building, shipping, and running applications using containerization. It enables developers to create, deploy, and manage applications within containers—self-contained units that package all necessary software and dependencies. Docker provides a consistent environment across various systems, making software development, deployment, and scaling more efficient and portable.

* Benefits
  + Flexible: Even the most complex applications can be containerized.
  + Lightweight: Containers leverage and share the host kernel.
  + Interchangeable: You can deploy updates and upgrades on-the-fly.
  + Portable: You can build locally, deploy to the cloud, and run anywhere.
  + Scalable: You can increase and automatically distribute container replicas.
  + Stackable: You can stack services vertically and on-the-fly.
  + Risks -> Security is the big one
  + Subject of ongoing discussion..
    - Docker Engine (daemon itself), registry (.e.g. docker hub), intrinsic security of the kernel
    - Limited ability to reason effectively across abstraction boundaries

Volume

* Instead of binding your local directory, you can use Docker volumes.
* A Docker volume is a directory somewhere in your Docker storage directory and can be mounted to one or many containers.
* They are fully managed and do not depend on certain operating system specifics.

Docker Compose

**Docker Compose is a tool used to define and run multi-container Docker applications. It allows users to use a YAML file to configure the services, networks, and volumes needed for a multi-container application. With a simple command, Docker Compose can start and manage multiple containers as a single application, streamlining the setup and management of complex applications that require multiple interconnected containers.**