DevOps: A How-To Guide

Introduction

In this article, we will cover the concept of DevOps and practices of development and deployment in it. As a simple explanation, DevOps is the philosophy of development and operations working together to make testing and deployment automated, repeatable and faster. The need for DevOps arose because of the large-scale adoption of Agile development over traditional software-development models like the Waterfall model.

Some common hurdles that come with scaling in organizations:

- Virtual Machine (VM) Provisioning and Management.
- The configuration of new hardware (infrastructure) and software on servers.
- Collecting and managing logs.
- Delivery and maintenance of software versions.
- Monitoring of hardware and software performance of machines and applications.
- Tackling and solving problems after product release.

In DevOps, the primary goals are:

- Easing the deployment, testing, and integration of software, introducing automation.
- Decreasing the failure rate of new releases.
- Ensuring reliability (web-servers or file-servers, development environments, etc.).
- Easing the scaling up for an organization with IT automation tools.
- Monitoring services and hardware/software and automatically making changes.
- Faster recovery in case of a new release crashing or causing problems.

DevOps: Brief History

With the rise of cloud-native technologies, deployment and management of software had to keep up. In 2009, the term “DevOps” made its appearance and gradually caught on, and now, many organizations have adopted the DevOps methodology. According to the CNCF 2018 survey, cloud-native projects in production grew more than 200% in 2018. The primary reason for this growth is the benefits of cloud-native technology (faster deployment, improved scalability, cloud portability). Cloud-native approaches are closely related to DevOps and microservice architectures, besides, newer products built around cloud-native architecture and support for DevOps processes by organizations are also on the rise.

Technical Information

Linux OS

The core of DevOps is the Linux OS. An engineer, or a team of engineers, with Linux OS expertise, is a necessity for adopting DevOps. Most organizations use Linux in product deployment.

Virtualization and VMs

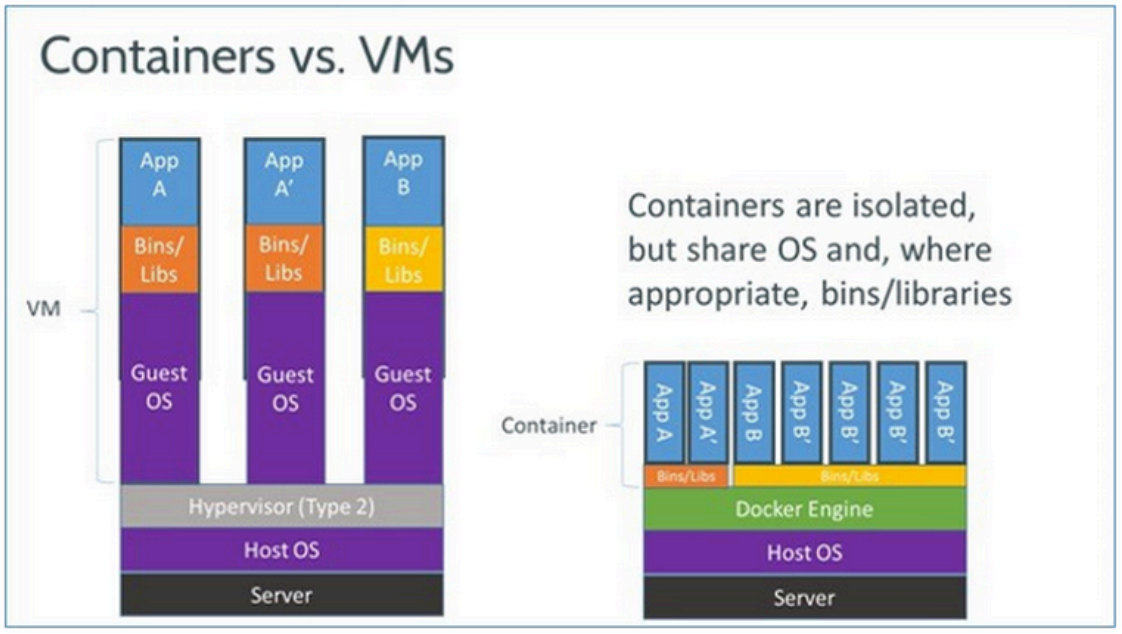
Virtualization is the process of creating a virtual replica of a physical machine, known as a Virtual Machine (VM). It can be done in two ways - Software-level, and Hardware-level. For instance, VMWare is a popular Windows software that allows virtualizing from ISO files. Alternatively, Citrix XenServer is hardware-level virtualization that is installed as a hard-disk partition, and allows creating, provisioning and managing VMs.

Version Control (git, mercury)

Version control, or Source control, is the process of keeping track of versions of content, especially useful during repairing updates that have broken something in the system. Git and Mercury are the most popular version control tools.

Software Containers and Docker

Containerization is a key element in DevOps. Docker is a software container platform designed for both developers and operations. A container is a light-weight, standalone package of software that includes everything needed to run it: Code, Libraries, System Tools and Settings. Containers can be thought of as a concept similar to VMs, but containers offer more isolation, resource efficiency, faster creation/management, and are limited compared to VMs. A variety of containers can be found in the Docker Hub.



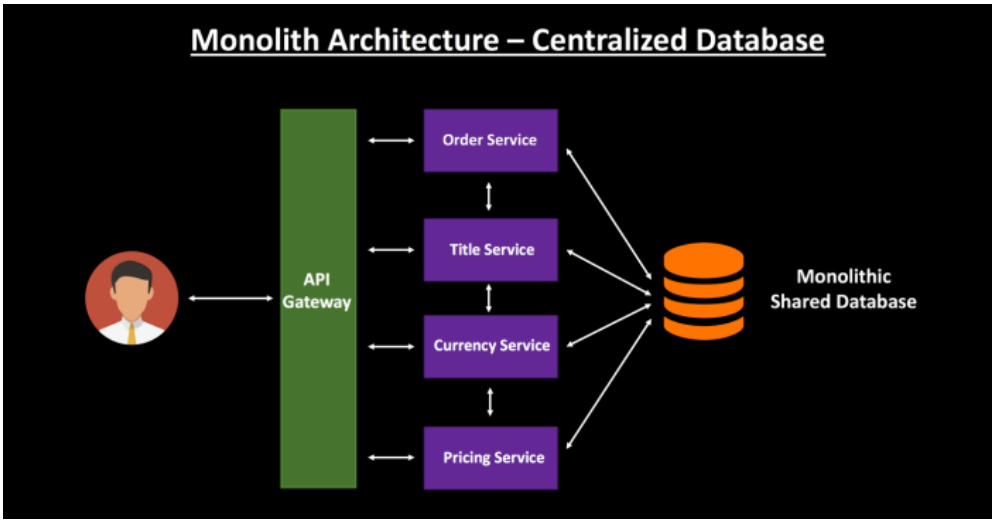
(Source:docker.com)

Infrastructure-as-code (IaC): Ansible/Terraform/Chef/Puppet

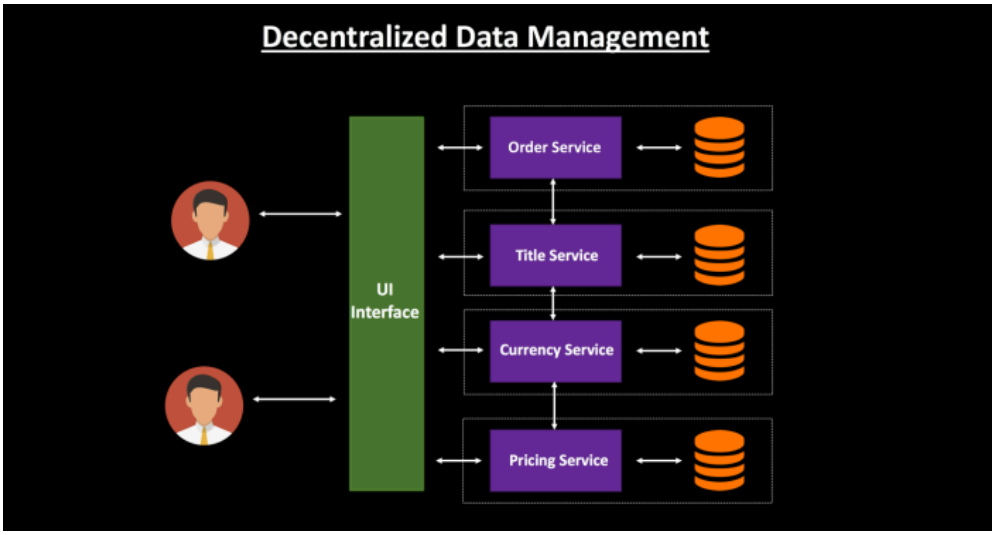
IaC is the process of managing and provisioning data centres through data files, instead of physical hardware configuration. Ansible, Terraform, and Chef are the most used IaC tools. IaC should not be confused with IaaS (Infrastructure-as-a-Service).

Microservices

Microservices are used to design modular and scalable architectures. They are not very different from a "module" that traditional software-development practice defines. With microservices, the cohesion is lower, as the architecture is decentralized such that no microservices are interdependent.



(a) Read/Write to a shared database



(b) Decentralizing the monolithic database.
(Source)

For instance, instead of reading/writing to one database as shown in (a) microservice architecture breaks down the database as shown in Figure (b) Microservices are a better alternative to modules.

Continuous Integration/Continuous Delivery (CI/CD) - Jenkins/Travis CI

CI/CD tools (Jenkins, Travis CI) improve the development process by automated testing, integration, and deployment, resulting in faster updating.

Agile development, Scrum

In Agile development, customer requirements and products evolve through the collaborative effort of the organization and the customer. Scrum is an Agile framework for development in teams of <10 people.

Serverless Computing and AWS Lambda

The cloud-provider acts as the server, managing allocation of resources. AWS Lambda has become popular for the same. Provided by Amazon, Lambda is event-driven and resources required by the event (code) are allocated automatically. In 2018, the majority (70%) of serverless technology users were using it. (Source:cncf.io)

DevOps Myths

Is DevOps a skill?

No, DevOps is a software development methodology. It is practised in teams, not individually.

DevOps is basically IT-Automation

Automation is a small part of DevOps. Various “automation tools” in DevOps make it simpler to update applications iteratively. However, DevOps is more about collaboration between development and IT to improve deployment processes, not just about automation.

DevOps is only using tools

DevOps requires a cultural shift in the development processes. The goal isn’t to simply implement toolsets, but to achieve collaboration between teams.

DevOps is about deploying updates frequently

With Agile and DevOps on the rise, the term “deployment” gets thrown around too often. DevOps is about improving the efficiency of deployment while reducing the pitfalls that arise when adding new code. Deployment should be based on improvements and requirements, not on frequency.

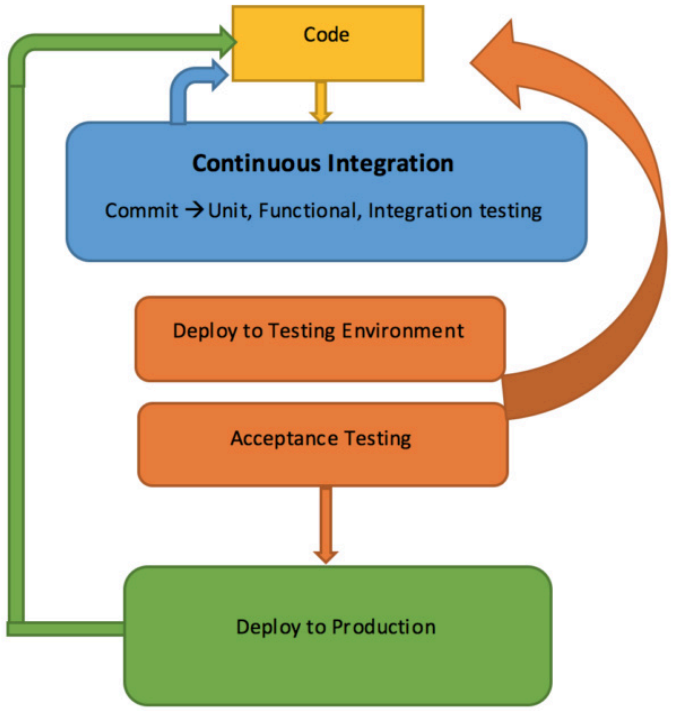
DevOps clashes with existing processes

DevOps easily integrates with modern software-development processes like Agile.

DevOps is for modern systems/smaller organizations

DevOps is a methodology that can be adapted for working with either modern or older systems. Similarly, the size of the organization does not matter - In larger organizations, DevOps is significantly beneficial.

Adopting DevOps



(Source:cio.com)

To reiterate, DevOps is not about tools; it is about people. To adopt the DevOps methodology:

1. Prepare for the process shift
   Development/Operations teams should be prepared first for any technological shift.
2. Develop a CI/CD pipeline with necessary tools.
3. Create a continuous testing environment and relevant metrics.
4. Establish the continuous deployment system
5. Use blue/green development
   B/G development reduces downtime by running two identical production environments. At any time, one is 'Live' and one is 'Idle'. During deployment, testing takes place in 'Idle' and when acceptance tests pass, 'Idle' and 'Live' environments are switched.
6. Frequently monitor and resolve performance metrics.

(Source)

Conclusion

In this article, we learned about concepts and processes related to DevOps. It is important to realize that DevOps is not a single skill but it is a methodology, and integrating it into your process will require investing time into automation and configuration but soon its benefits will become noticeable.