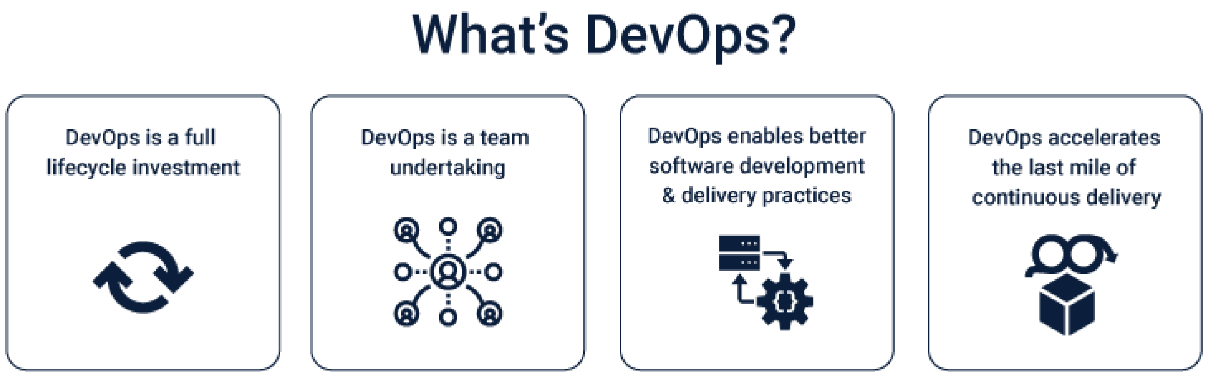
Introduction to DevOps Culture & Practices

What is DevOps?

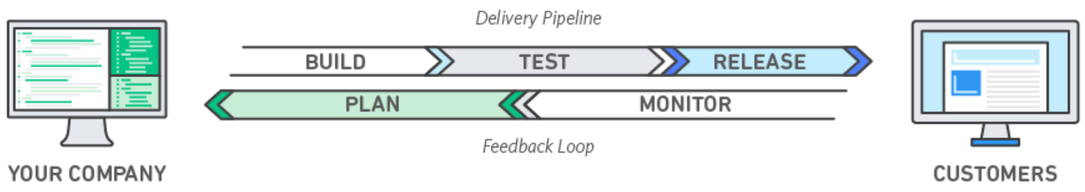
It's a combination of cultural philosophies, practices, and tools. It is a culture of shared understanding between developers and operations, and shared responsibility for the software that is built. The intent of this hypothesis is to increase transparency, communication, and collaboration across development, IT/operations, and Business.


What's DevOps?
DevOps is a full lifecycle investment
DevOps is a team undertaking
DevOps enables better software development & delivery practices
DevOps accelerates the last mile of continuous delivery

Why DevOps?

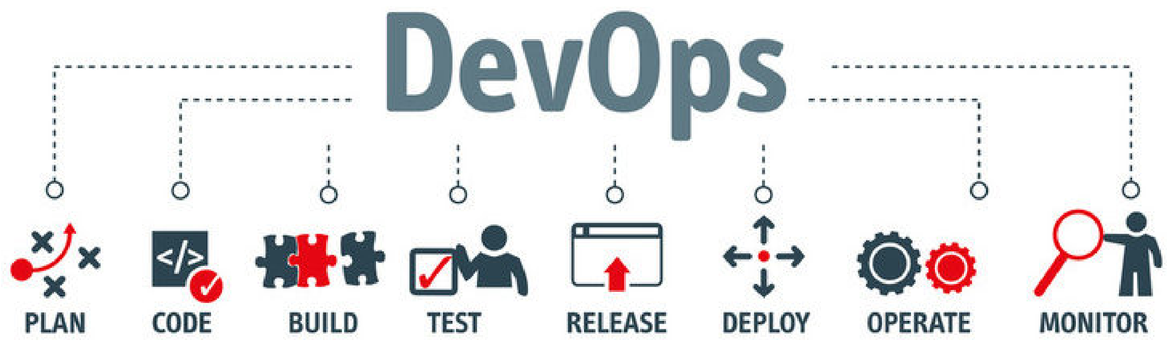
As you already understand, DevOps is a set of practices that combines software development and IT operations which aims to shorten the SDLC & provide continuous delivery (CD) with high software quality.

This set of practices increases the velocity of continuous software delivery of an organization. It decreases manual tasks tense to zero during deployment, test, and release for end users. Achieve greater customer satisfaction by delivering the final release product of the sprint at a faster time by early detection and correction of issues gained throughout this practice.



Phases of DevOps?

During writing new software, all processes can be divided into several steps or development phases of software from planning to getting end-user feedback and monitoring. DevOps combines every step in a single chain.



- ☐ **Plan:** Roughly equivalent to the requirements-gathering phase includes everything that happens before developers start writing code.
- ☐ **Code**: DevOps practitioners build in small modules, relying on automated tools to maintain version control, enforce consistent style standards and guard against security issues.
- ☐ **Build**: Build is a high-level plan of how to get from your code to an executable binary
- ☐ **Test**: Test is a process of software testing at every stage of the software development life cycle.
- ☐ **Release**: The product is prepared for actual release to the customer. Early form release called alpha version, beta version.
- ☐ **Deploy**: The final stage is "deploy," where once the production environment is created and is configured, the final version of the build is deployed.
- ☐ **Operate**: Once DevOps software has gone live in production, the operations team will rely on automated tools wherever possible for configuration management, scaling, and load balancing.
- ☐ **Monitor**: The goal of monitoring is detecting the problematic areas of a process and analyzing the feedback from the team and users to report existing inaccuracies and improve the product's functioning.

DevOps Practices

The following are DevOps best practices:

- ☐ **Continuous Integration:** Practice of automatic integration of code changes from multiple contributors into a single software project.
- ☐ **Continuous Delivery:** Continuous delivery is an extension of continuous integration. It automatically deploys all code changes to the testing and/or production environment after the build stage.
- ☐ **Microservices:** Microservices is an architectural style that structures an application as a collection of services also known as microservice architecture.
- ☐ **Infrastructure as Code:** A snippet of code that is used for managing and provisioning infrastructure instead of manual processes.

- ☐ **Monitoring and Logging:** Log monitoring refers to the set of practices involved in log management and analysis to help to keep eye on infrastructure and operations. Monitoring can be classified into different categories depending on the scope and methods used.
- ☐ **Communication and Collaboration**: Communication and collaboration can improve team bonding and help to reach the project goal. There must be a mutual understanding between Development & Operation where DevOps works like a bridge.

A DevOps Engineer: Role and Responsibilities

Here are some basic and widely-accepted responsibilities from my experience. But these are not only and differ from organization to organization & team to team:

- ☐ Understand packages & resource requirements for the application.
- ☐ Documentation for the server-side features and environment-wise configuration.
- ☐ Configure and manage Continuous deployment and continuous integration (**CI/CD**).
- ☐ Primary & backup Infrastructure provision & management
- ☐ Performance assessment and monitoring of application as well as Infrastructure.
- ☐ Write Infrastructure as Code (**IaC**) and various scripts as per requirements
- ☐ Configure & provide test automation infrastructure along with **QA** teams.
- ☐ Working with various log management & analysis tools

Additionally, a DevOps engineer can be responsible for IT infrastructure maintenance and management, which comprises hardware, software, network, storage, virtual and remote assets, and control over cloud data storage.

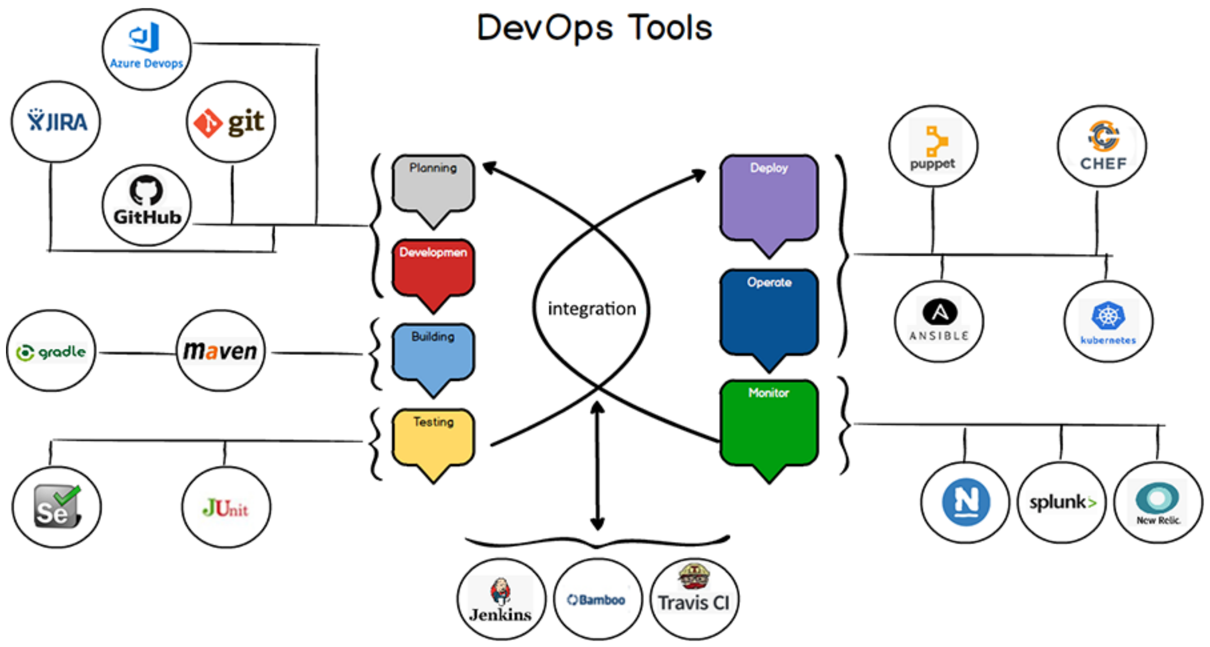
DevOps Engineer Require Skill Set (Minimum)

From my experience, to be a good DevOps engineer, you should maintain set skills as below -

- ☐ Git Repository / Version Control.
- ☐ Programming / Scripting Language: Python, Bash, Groovy.
- ☐ Linux / Unix Fundamental and Scripting.
- ☐ Networking fundamentals.
- ☐ Infrastructure Orchestration Tools.
- ☐ Understand Virtualization & Cloud.
- ☐ Container technology & Artifactory management
- ☐ Ability to manage & monitor logs.
- ☐ Testing & Performance monitoring Tools.

DevOps Tools

A set of applications that helps automate the software development and delivery process. It mainly focuses on communication and collaboration between product management, software development, Software Quality Assurance (SQA) and operations. DevOps tools also enable teams to automate most of the software development processes. Helps to reduce manual efforts for building, conflict management, dependency management, deployment, etc.



- ☐ **Build Tools**
    - ☐ Maven
    - ☐ Gradle
    - ☐ Makefile
- ☐ **Repository Management:**
    - ☐ Git
    - ☐ GitHub
- ☐ **CICD Tools**
    - ☐ Jenkins
    - ☐ Travis
    - ☐ TeamCity
    - ☐ Bamboo
- ☐ **Ticket Management Tools**
    - ☐ Jira
    - ☐ Youtrack
- ☐ **Configuration Management Tools**
    - ☐ Ansible
    - ☐ Puppet
    - ☐ Chef

- ☐ Saltstack

- ☐ **Container Frameworks Tools**
  - ☐ Docker Swarm
  - ☐ Kubernetes

- ☐ **Virtualization, and Containerization Tools**
  - ☐ AWS
  - ☐ OpenStack
  - ☐ Proxmox
  - ☐ Vagrant
  - ☐ Docker
- ☐ **Monitoring & Log Managements**
  - ☐ Nagios
  - ☐ Zabbix
  - ☐ Splunk
  - ☐ New Relic
  - ☐ Prometheus
  - ☐ Datadog
- ☐ **Secrets Management Tools**
  - ☐ Beyond Trust
  - ☐ AWS Secret Manager
  - ☐ Vault
  - ☐ Docker Secret
- ☐ **Reverse Proxy Server**
  - ☐ HAProxy
  - ☐ Nginx
  - ☐ Traefik
- ☐ **Testing Tools**
  - ☐ JMeter
  - ☐ Selenium
  - ☐ JUnit

Proclamation - All figures and images are used in this paper from the internet without modification.