DEVOPS

DevOps - A Complete Guide

Phone/Whatsapp: +1 (515) 309-7846 (USA)

Email: info@zarantech.com

Website: www.zarantech.com

Contents

Introduction to DevOps?

DevOps represents a fundamental shift within organizations, extending beyond mere tools and practices. It emphasizes collaborative efforts, automation, and continuous delivery, effectively breaking down traditional barriers that separate development and operations teams. This approach promotes shared accountability for the complete software lifecycle.

Need of DevOps

In a hypothetical situation, envision a software development team investing substantial effort in creating innovative applications, only to encounter operational challenges during the launch phase. These difficulties result in delays, errors, and disruptions in the smooth transition from development to deployment, potentially impeding the company's competitiveness in the market.

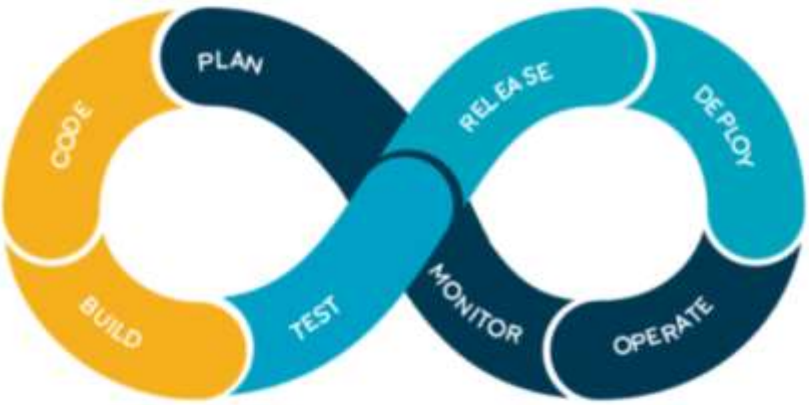
Why DevOps over other methodologies?

DevOps confronts these issues by fostering collaboration, communication, and integration between development and operations teams. It endeavors to dismantle the traditional barriers separating these two functions. Through instigating a cultural transformation and leveraging advanced tools and methodologies, DevOps is geared towards achieving pivotal objectives such as expediting delivery, bolstering reliability, and nurturing innovation.

DevOps LifeCycle

DevOps aims to unify all members involved in development, operations, and IT infrastructure, encompassing developers, testers, system administrators, and quality assurance professionals. Thus, collectively, these individuals are referred to as DevOps engineers.

These engineers collectively shoulder the comprehensive responsibility of information gathering, infrastructure setup, development, testing, deployment, continuous monitoring, and feedback collection from end-users.



Code: The initial phase of the DevOps life cycle involves coding, where developers build the code on any platform.

Build: Developers create the program's version in a suitable extension based on the language used.

Test: Automation of the testing process using a DevOps automation tool like Selenium is essential for successful DevOps implementation.

Release: This phase involves managing, planning, scheduling, and controlling the build across different environments after testing and before deployment.

Deploy: All artifacts and code files of the application are prepared and executed on the server during this phase.

Operate: After deployment, the application is operational, and clients use it in real-world scenarios.

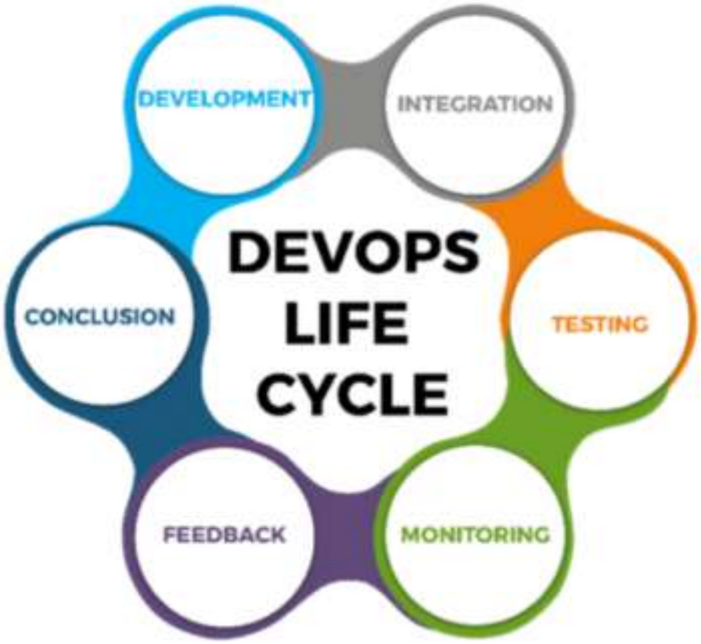
Monitor: This phase provides crucial information to ensure service uptime and optimal performance.

Plan: The planning stage gathers feedback from the monitoring phase and implements changes for better performance.

Different Lifecycle Stages

The various stages in the DevOps life cycle that contribute to the consistent software development life cycle (SDLC) are

- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Monitoring
- Virtualization and Containerization



Continuous Development

In the Waterfall model, the software product is divided into multiple pieces or sub-parts to shorten the development cycles. However, in this stage of DevOps, the software is developed continuously.

Tools employed: For coding and building in this stage, GIT is used to manage different versions of the code. Additionally, a reliable tool, Maven, is utilized to build and package the code into an executable file.

Continuous Integration

During this phase, when our code supports new functionality, it is continuously integrated with the existing code. As continuous development progresses, the existing code must be consistently integrated with the latest changes to ensure smooth functionality in the current environment.

Tools utilized: Jenkins serves as the tool for continuous integration. It enables pulling the latest code from the GIT repository, allowing the generation of the build and subsequent deployment on the test or production server.

Continuous Testing

During the continuous testing stage, our developed software undergoes continuous testing to detect bugs using various automation tools.

Tools utilized: For QA/Testing purposes, there are numerous automated tools available, with Selenium being widely used for automation testing. Selenium enables QAs to test the codes in parallel, ensuring that the software is free from errors, incompetencies, or flaws.

Continuous Monitoring

This stage is a vital component of the DevOps life cycle, providing essential information to maintain service uptime and optimize performance. The operations team relies on reliable monitoring tools to identify and address bugs/flaws within the application.

An array of tools, including Nagios, Splunk, ELK Stack, and Sensu, are employed to monitor the application, enabling close scrutiny of application and server health and activity. Any significant issues flagged by these tools are escalated to the development team for resolution in the continuous development phase.

DevOps Tools

The following showcases some of the most popular DevOps tools:

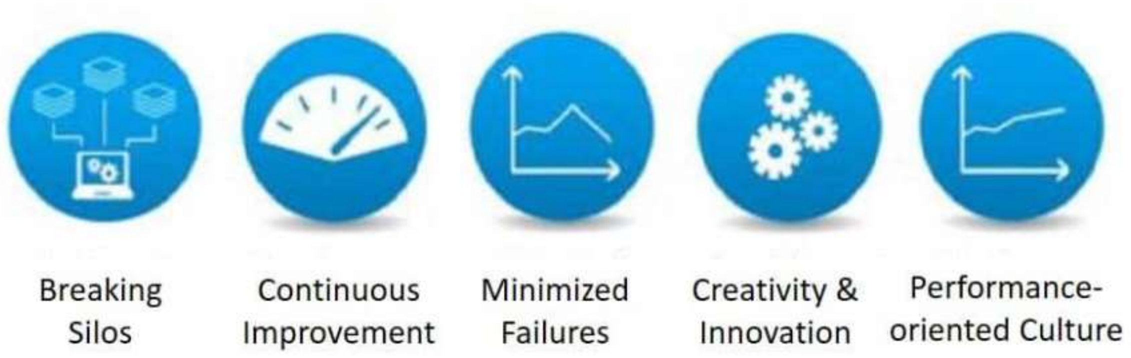
- **Puppet:** It facilitates the swift and frequent delivery and release of technological changes, featuring versioning, automated testing, and continuous delivery capabilities.
- **Docker:** This high-end tool is used for building, shipping, and running distributed applications across multiple systems, enabling rapid assembly of applications and container management.
- **Jenkins:** A widely utilized DevOps tool that monitors the execution of repetitive jobs while also facilitating seamless integration of changes and quick access to results.
- **Ansible:** This tool automates the entire life cycle of an application, managing complex deployments and enhancing productivity.
- **Nagios:** A DevOps tool designed for monitoring IT infrastructure, adept at identifying and rectifying errors through standard network, server, and log monitoring systems.

Advantages and Disadvantages of DevOps

Having proven successful in SDLC, DevOps is now recognized as essential for accelerating various cloud platforms. Its versatile performance has enticed individuals to pursue careers in this domain, emphasizing the importance of acquiring comprehensive knowledge.

DevOps represents a modern approach that empowers companies to leverage multiple benefits. Below are some of the primary DevOps advantages and constraints:

Advantages of DevOps



- **Faster Delivery:** DevOps practices enable rapid deployment, reducing time-to-market for software.
- **Improved Collaboration:** DevOps fosters better communication and collaboration among development, operations, and other teams.
- **Continuous Integration and Deployment:** CI/CD pipelines allow seamless code integration, testing, and deployment, ensuring a continuous delivery cycle.
- **Enhanced Quality and Stability:** DevOps emphasizes automated testing, leading to higher software quality and more stable releases.
- **Scalability and Flexibility:** DevOps supports scalability and flexibility, enabling systems to adapt and grow with evolving requirements.

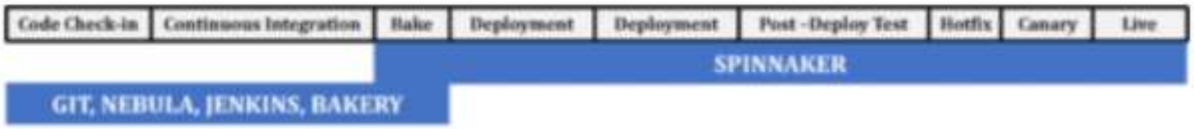
Disadvantages of DevOps

- **Complexity:** DevOps methodologies can be intricate to implement, necessitating a deep understanding and expertise.
- **Initial Set-Up Challenges:** Establishing an efficient DevOps environment may present initial hurdles related to integration and tooling.
- **Resistance to Change:** Teams accustomed to traditional methods may resist the implementation of DevOps principles, posing a challenge.
- **Security Concerns:** Frequent iterations and rapid deployments in DevOps can potentially elevate security vulnerabilities if not managed adequately.
- **Tool Dependency:** Relying on multiple tools and technologies can create challenges regarding tool compatibility and maintenance.

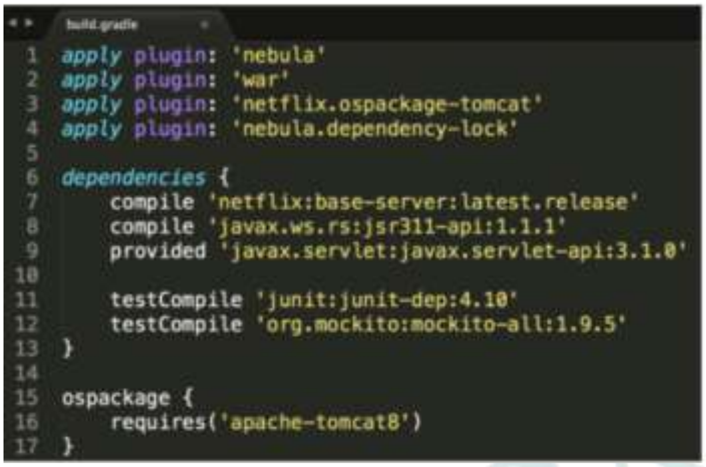
DevOps Use Case

Netflix, a prominent media-subscription provider, offers a wide array of TV shows and movies for streaming on popular smart devices, catering to over 75 million global customers and providing an exceptional viewing experience.

Netflix harnesses DevOps to ensure a seamless and superior video streaming experience for its customers. The company employs Spinnaker's continuous delivery platform for the uninterrupted deployment of its application. Prior to reaching Spinnaker, several preliminary steps are necessary, as illustrated in the diagram below:



Prior to deploying with Spinnaker, the code undergoes the build and testing phases. Netflix utilizes Nebula for the build process, considering it as the optimal build tool for Java applications. Nebula, comprising Gradle plugins, is tailored for Netflix engineers to streamline build logic and establish rational conventions.



```
apply plugin: 'nebula'
apply plugin: 'war'
apply plugin: 'netflix.ospackage-tomcat'
apply plugin: 'nebula.dependency-lock'

dependencies {
    compile 'netflix:base-server:latest.release'
    compile 'javax.ws.rs:jsr311-api:1.1.1'
    provided 'javax.servlet:javax.servlet-api:3.1.0'

    testCompile 'junit:junit-dep:4.10'
    testCompile 'org.mockito:mockito-all:1.9.5'
}

ospackage {
    requires('apache-tomcat8')
}
```

Additionally, the code is tested locally using Nebula, and any changes are committed to Netflix's central Git repository. Furthermore, Netflix transitions its monolithic application to cloud-based microservices in AWS, leveraging the microservice architecture to ensure loosely coupled teams can build and implement changes at their preferred pace.

Subsequently, a Jenkins job is established to execute Nebula, facilitating the building, testing, and packaging of the application for subsequent deployment. The resulting build is then transformed into an Amazon Machine Image (AMI). To generate AMIs from the source, Netflix employs a bakery that exposes an API, enabling the globally accessible creation of AMIs.

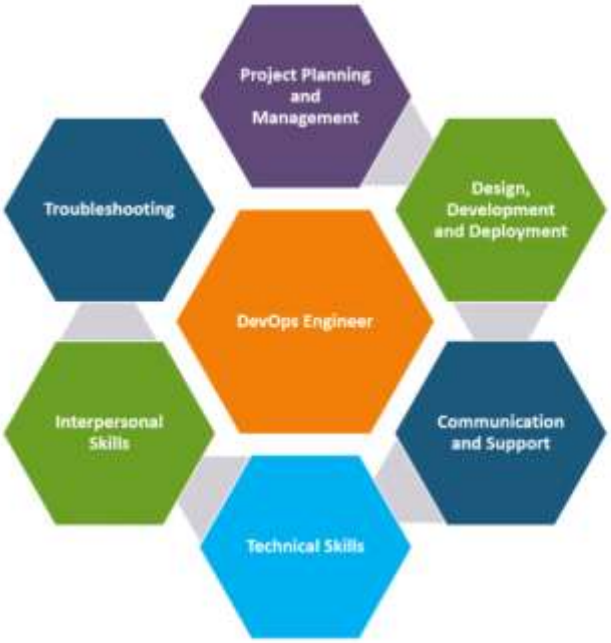
Once the baking process concludes, Spinnaker intervenes to facilitate deployment, making the resulting AMIs available for tens, hundreds, or even thousands of instances.

After this continuous integration, deployment, and ultimate availability, the application is set live.

Roles and Responsibilities of a DevOps Engineer

DevOps is increasingly ingrained as a culture embraced by major players in the IT industry. When implemented effectively, DevOps aids in bridging the divide between development and operations teams.

Key roles and responsibilities of DevOps Engineers encompass


Project Planning and Management
Troubleshooting
Design, Development and Deployment
DevOps Engineer
Interpersonal Skills
Communication and Support
Technical Skills

- **Project planning and management:** DevOps Engineers must monitor software, regulate and update tools, and demonstrate expertise in tracking costs, benefits, risks, and more in various DevOps projects.
- **Design, development, and deployment:** DevOps Engineers are responsible for designing, developing, and deploying automated modules to ensure seamless functioning within the production environment, utilizing risk-management techniques and tests.

- **Communication and support:** DevOps Engineers should possess exceptional communication skills for effective coordination with different departments and providing support.
- **Technical skills:** Basic technical experience and familiarity with configuration tools are essential.
- **Interpersonal skills:** DevOps Engineers should exhibit approachability, organization, foresight, and the ability to multitask effectively due to frequent interaction with other organizational departments.
- **Troubleshooting:** DevOps Engineers are tasked with troubleshooting and providing suitable solutions for various errors to enhance the firm's speed and efficiency.

Job Opportunities in DevOps

DevOps offers diverse and lucrative career prospects for skilled and certified professionals. The various profiles to consider include

- **DevOps Evangelist:** Focuses on identifying the benefits of DevOps and promoting its principles.
- **Code Release Manager:** Oversees the progress of projects using Agile methodology.
- **Automation Architect:** Designs and constructs automated tools and systems for continuous and seamless deployments.
- **Experience Assurance:** Works towards enhancing user experience by identifying bugs and ensuring inclusion of essential features in applications.
- **Software Developer/Tester:** Ensures that code aligns with original business requirements, while performing comprehensive testing and monitoring.

- **Security Engineer:** Integrates security measures into applications and products to safeguard the business.

Conclusion

After completing this comprehensive DevOps tutorial, we have gained a detailed understanding of DevOps, its life cycle, various tools, use cases, and the roles and responsibilities of DevOps Engineers.

While we have covered key aspects of DevOps, there is still much more to explore. For those seeking further knowledge, consider enrolling in a structured DevOps Certification Training provided by Intellipaat. This training offers case-based scenarios, topic-wise assignments, hands-on sessions, and industry-based projects to prepare you for opportunities in leading companies.

This course aims to provide a deep understanding of DevOps concepts, essential tools, and frameworks necessary for success as a productive DevOps team member in any organization.

We value your feedback, so please let us know in the comment section if this beginner's DevOps tutorial was helpful to you.

Elevate your skills and become a DevOps expert with our comprehensive DevOps Learning Progra

Zarantech
knowledge is Everything

THANK YOU

Corporate Training Course Catalog

https://lnkd.in/e4fJJRVZ

DevOps Learner Community

https://www.linkedin.com/showcase/devops-learner-community/

Get any DevOps Video Training

https://zarantech.teachable.com/courses/category/devops

Phone/Whatsapp: +1 (515) 309-7846

Email: info@zarantech.com

www.zarantech.com