Overview of DevOps

A study by Grand View Research says that the ***DevOps*** market size across the globe is estimated to reach US$12.85 billion by 2025. According to a report from SD Times, 'Indeed's job postings show that the role of a DevOps Engineer has seen a 225 percent jump. DevOps Engineer is at the second position on Glassdoor's rankings for the 50 best jobs in America. Glassdoor says that the average annual salary of a DevOps Engineer ranges from US$135,000 to US$180,000.

What is DevOps?

Reportlinker also predicts that the DevOps market size would grow up to US$10.3 billion by 2023, at a compound annual rise rate of 24.70 percent during the forecast period. Being in the technology field, you must be aware of this buzzword, **DevOps,** taking up the IT world by storm. This tutorial will take you through the main concepts of DevOps, along with its important tools and use cases.

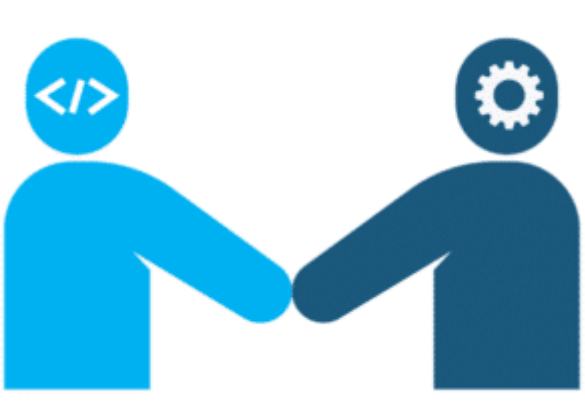
Here, we have the list of topics if you want to jump right into a specific one:

- What exactly is DevOps?
- Why DevOps?
  - The Workflow of the Waterfall Method
  - Agile Software Development
- How does DevOps work?
- DevOps Tools
- DevOps Benefits
- DevOps Use Case in Netflix

- Roles and Responsibilities of a DevOps Engineer
- Conclusion

What exactly is DevOps?

Don't get scared with this term 'DevOps.' It is nothing but the practice or methodology of making 'Developers' and the 'Operations' team work together.



Now, how exactly is this achieved? We will discuss this as we move further.

Why DevOps?

Before understanding the concepts and methodology of DevOps, we need to understand why do we even need DevOps?

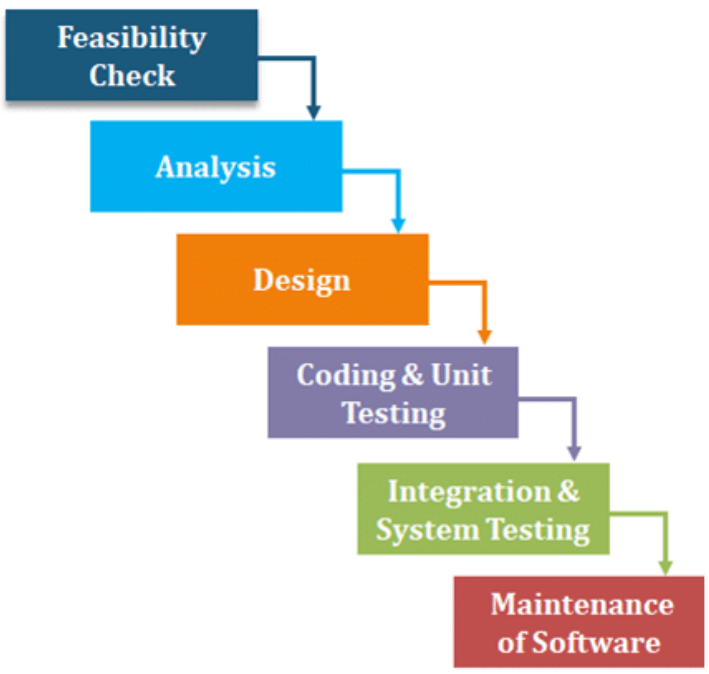
Why DevOps? Why not other methods?

Before DevOps came into the picture, the Waterfall model was the earliest SDLC approach that was used for software development. This method, which was used for illustrating SDLC in a sequential flow, was considered to be reliable at first.

Prepare yourself for the industry by going through these Top DevOps Interview Questions and Answers!

The Workflow of the Waterfall Method

Let's consider, we're developing software using the Waterfall method. Below are the steps that will be included in the SDLC if we're using this method:



- **Feasibility check:** The feasibility phase is used for determining whether a particular approach/technique will be feasible enough for developing the software.
- **Analysis of the requirements:** In this phase, we need to analyze all system and software requirements from customers' points of view and gather information about these requirements. The requirements will then be captured in the software requirement specification (SRS) document to avoid the incompleteness of the product.
- **Design:** The goal of this phase is to transform the requirements listed in the SRS document into an ordered structure that is appropriate for their implementation in programming.

Learn the Azure DevOps interview Questions to have a better aspect of the industry-level questions.

- **Coding and unit testing:** The design that is created in the previous stage is supposed to get converted into the source code in this stage, and then every design module is coded and checked individually.
- **Integration and system testing:** After the design of each module has been coded, the integration of these modules is carried out appropriately. Then, these integrated modules are tested individually. After this, **acceptance testing** is carried out in which the product is delivered to and tested by the customer for checking whether to accept it or reject it.
- **Maintenance of the software:** Maintenance is that phase of the software development life cycle where 60 percent of the entire effort is spent. Several maintenance operations are performed in this phase such as **corrective maintenance, perfective maintenance**, and **adaptive maintenance**, where error corrections and functionality enhancement, along with trying the software on new environments and operating systems, are done.

Become a master of DevOps by going through this online DevOps Training in London!

Watch this video on DevOps Tutorial for Beginners:

DevOps Training | DevOps Course | DevOps Tutorial for Beginners | Intellipaat

Now, let's discuss the **advantages and disadvantages** of this model.

Advantages

- This method is easy and simple to use
- Easy to manage due to its rigidity
- Each phase has a review process making it less vulnerable to errors
- The one-at-a-time process phases do not overlap each other
- Reliable for small projects

Disadvantages

- While the application is in the testing stage, it is really difficult to go back and make changes relating to any issue that happened in the previous steps due to miscommunication or lack of knowledge
- It is a risky process as it is difficult to diagnose and to provide feedback
- Its main focus is to help internal teams work efficiently. It excludes end-users/clients, due to which the majority of people do not trust this methodology
- There will be delays in the testing process because this method insists teams to wait until the process reaches its 4th or 6th stage

Because of all these disadvantages, organizations wanted a much efficient model to carry out SDLC. Hence came the **Agile** method that changed the scenario.

Enroll today in the Best DevOps Course in New York to get a clear understanding of DevOps!

Agile Software Development

Agile involves an incremental approach like the Waterfall model but with an iterative perspective, along with focusing on customer feedback, incorporating small rapid changes, and speeding up releases. It basically breaks the product into smaller divisions and finally integrates them for the testing process.

Now, let's take a look at its **advantages and disadvantages**.

Advantages

- The Agile methodology considers customer feedback throughout the project, which gives enough time to the team for making decisions
- It welcomes making changes but at great expense
- It has the ability to scale
- There is continuous attention to technical excellence and good designs
- This method prioritizes and schedules the most valuable features for implementation, decreasing the risk of having unusable resources
- Small and dedicated teams are involved with a high degree of involvement and coordination

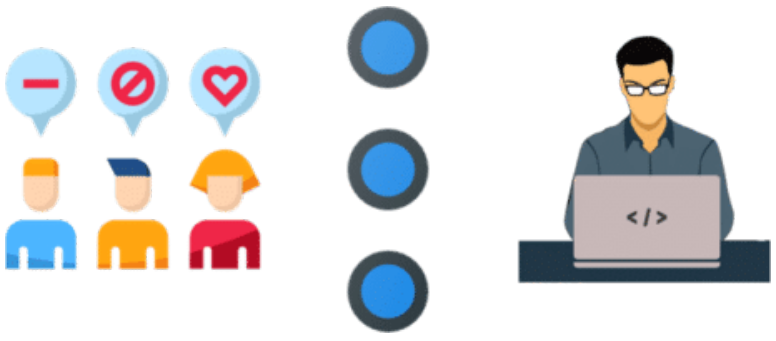
Disadvantages

- There is less predictability in Agile
- It requires more time and commitment from every stakeholder. Testers, developers, and customers must interact with each other constantly and should agree to each other's decisions in order to get the task done, and hence Agile is time-consuming
- Limited documentation often comes as a problem. In the case of fallbacks, there are very less detailed documents so as to cross-check
- The Agile model requires minimal planning at the beginning that makes it easier to develop the project quickly, but there is never a finite end. Due to unexpected functionalities, a clear vision of the project is not available, and mostly the stakeholders are not sure of what their final product would look like

Have you looked into the AWS DevOps Interview Questions yet? Read more.

In a nutshell, when the Waterfall model failed to deliver consistency in the result, the Agile methodology came into existence. However, as discussed above, there were many disadvantages to the Agile model as well:

- In the case of the **Waterfall model**, there was a gap between customers' software requirements and the developers, which was overcome by Agile



Gaps Between Customers and Developers

- While in the case of the **Agile** method, there was still a gap between the development and operations folks

Gaps Between the Operations Team and Developers

How do you think it was overcome?

It was in this scenario **DevOps was introduced** in order to overcome the gap between developers and the operations team.

Look into the Monitoring DevOps Tools blog provided by Intellipaat.

Moving to the next section in this DevOps tutorial, let's check out the major differences between Agile and DevOps.

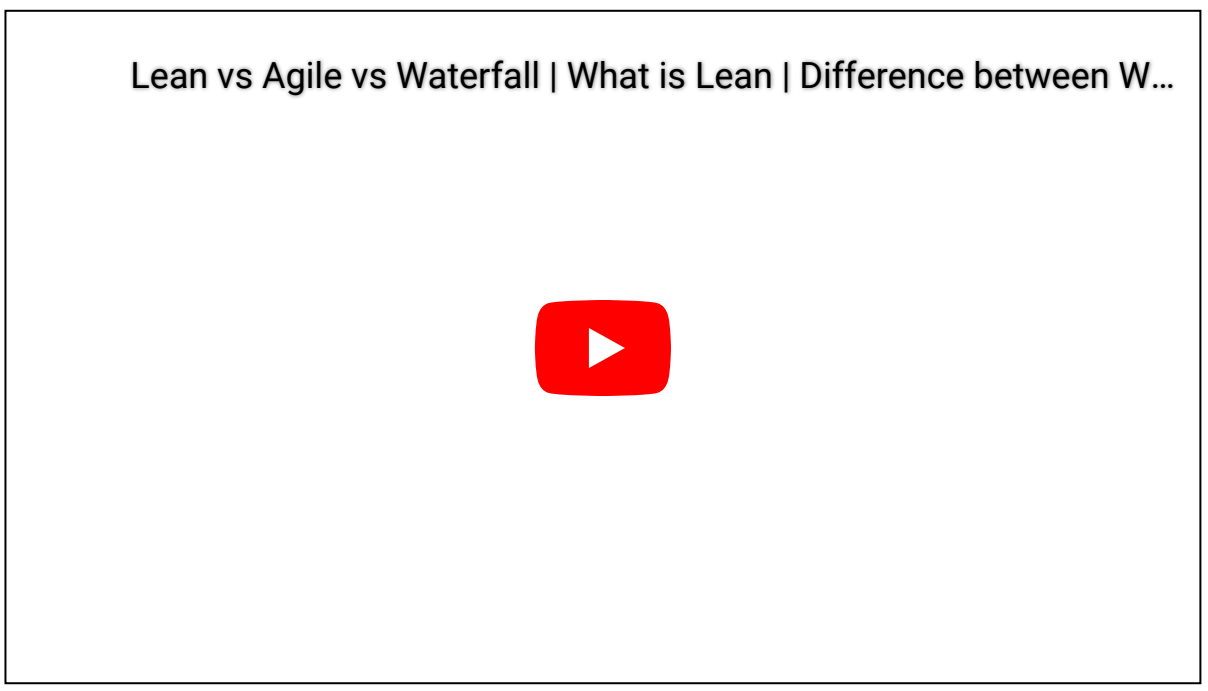
Differences Between Agile and DevOps

Agile	DevOps
Agile majorly focuses on collaboration, customer feedback, and small rapid changes	DevOps brings development and operations teams together
It does not focus on automation	It focuses majorly on automation to increase efficiency while deployment
The development process is inherent for Agile, making it less focused on testing and implementation processes	DevOps focuses on all development, testing, and implementation phases with equal importance
It overcomes the gap between customers and developers	It overcomes the gap between the development and operations folks

How exactly does DevOps work?

Let's move to the next section in this DevOps tutorial and check out the DevOps life cycle. This way, we can understand how DevOps works.

Watch this video on Lean vs Agile vs Waterfall:



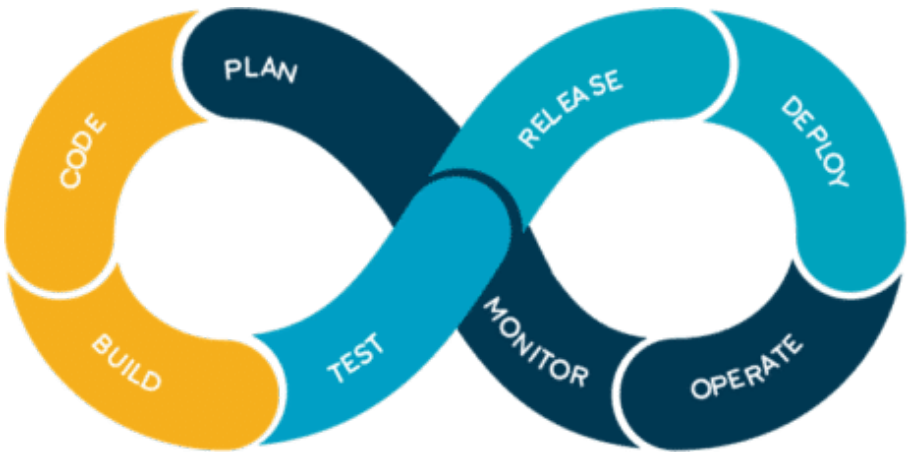
DevOps Lifecycle

DevOps focuses on bringing all the development, operations, and ***IT infrastructure*** guys, including Developers, Testers, System Admins, and QAs, under one roof. Hence, all these people together are called ***DevOps Engineers***.

DevOps Engineers share the end-to-end responsibility of gathering information, setting up the infrastructure, developing, testing, deploying, continuous monitoring, and fetching feedback from end-users. This process of developing, testing, deploying, and monitoring keeps on repeating for better results.

Learn more about DevOps from this DevOps Training in Sydney to get ahead in your career!

You can actually figure it all out from the DevOps diagram illustrated below:



- **Code:** The first step in the DevOps life cycle is coding, where developers build the code on any platform
- **Build:** Developers build the version of their program in any extension depending upon the language they are using
- **Test:** For DevOps to be successful, the testing process must be automated using any automation tool like Selenium
- **Release:** A process for managing, planning, scheduling, and controlling the build in different environments after testing and before deployment
- **Deploy:** This phase gets all artifacts/code files of the application ready and deploys/executes them on the server
- **Operate:** The application is run after its deployment, where clients use it in real-world scenarios.
- **Monitor:** This phase helps in providing crucial information that basically helps ensure service uptime and optimal performance
- **Plan:** The planning stage gathers information from the monitoring stage and, as per feedback, implements the changes for better performance

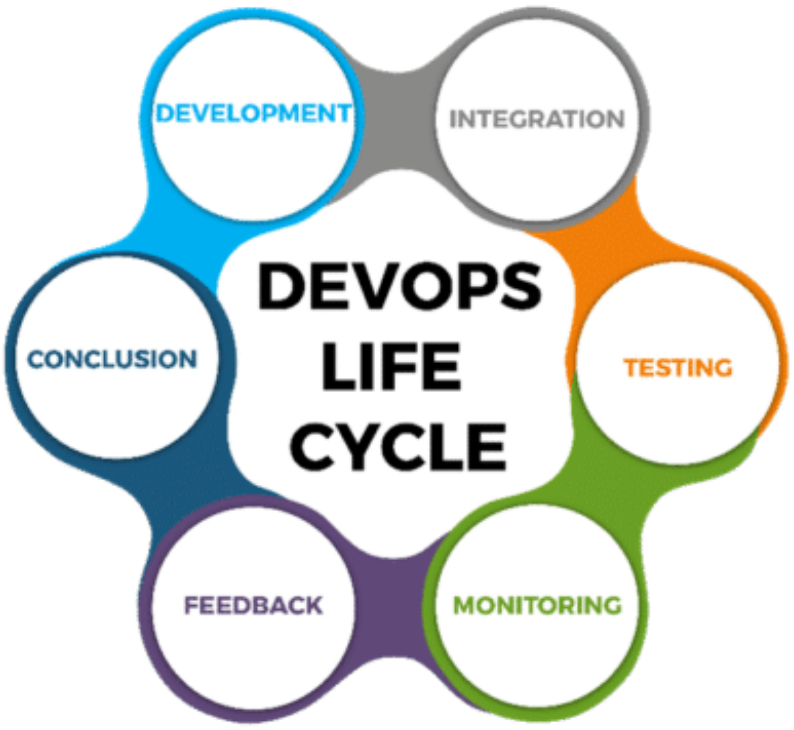
If you have doubts or queries related to DevOps, get them clarified from DevOps experts on our DevOps Community!

Different Lifecycle Stages

Now, let's discuss the different stages in the DevOps life cycle that contributes to the consistent software development life cycle (SDLC):

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

These stages are basically the aspects of achieving the DevOps goal.



Now, let's discuss each of them in detail.

Career Transition

Continuous Development

In the Waterfall model, our software product gets broken into multiple pieces or sub-parts for making the development cycles shorter, but in this stage of DevOps, the software is getting developed continuously.

- **Tools used**: As we code and build in this stage, we can use **GIT** to maintain different versions of the code. To build/package the code into an executable file, we can use a reliable tool, namely, **Maven**.

***GIT Cheat Sheet** by Intellipaat is one of the most visited and downloaded. Check out.*

Continuous Integration

In this stage, if our code is supporting new functionality, it is integrated with the existing code continuously. As the continuous development keeps on, the existing code needs to be integrated with the latest one '**continuously**,' and the changed code should ensure that there are no errors in the current environment for it to work smoothly.

- **Tools used**: **Jenkins** is the tool that is used for continuous integration. Here, we can pull the latest code from the GIT repository, of which we can produce the build and deploy it on the test or the production server.

Continuous Testing

In the **continuous testing** stage, our developed software is getting tested continuously to detect bugs using several automation tools.

- **Tools used:** For the **QA/Testing** purpose, we can use many automated tools, and the tool used widely for **automation testing** is Selenium as it lets QAs test the codes in parallel to ensure that there is no error, incompetencies, or flaws in the software.

Continuous Monitoring

It is a very crucial part of the DevOps life cycle where it provides important information that helps us ensure service uptime and optimal performance. The operations team gets results from reliable monitoring tools to detect and fix the bugs/flaws in the application.

- **Tools used**: Several tools such as **Nagios, Splunk**, **ELK Stack**, and **Sensu** are used for monitoring the application. They help us monitor our applications and servers closely to check their health and whether they are operating actively. Any major issue detected by these tools is forwarded to the development team to fix in the continuous development phase.

Let's now talk about some of the major DevOps tools in the next section of this DevOps tutorial.

Interested in becoming a DevOps expert? Go for our DevOps Course in Toronto!

DevOps Tools

The most popular DevOps tools are discussed below.

- **Puppet**: Puppet is one of the widely-used DevOps tools. It allows delivering and releasing technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery.
- **Docker**: Docker is a high-end DevOps tool that allows building, shipping, and running distributed applications on multiple systems. It helps assemble the applications quickly and is typically suitable for container management.
- **Jenkins**: Jenkins is one of the most popular DevOps tools that allow monitoring of the execution of repeated jobs. Apart from this, Jenkins lets us integrate the changes and access the results easily and quickly.
- **Ansible**: This tool helps automate the entire life cycle of an application, and manages complicated deployments, and enhances productivity.
- **Nagios**: This DevOps tool helps monitor the IT infrastructure. It is capable of determining errors and rectifying them with the help of the standard network, server, and log monitoring systems.

Have you gone through the Ansible Cheat Sheet yet? Visit Now!

DevOps Benefits

After being successfully implemented in SDLC, now DevOps is considered the key to speeding up various cloud platforms. Its all-rounder performance has attracted aspirants to build a career in this domain, and hence having sound knowledge is imperative for them.

DevOps is a contemporary approach that lets companies utilize numerous benefits. Some of the major ***DevOps benefits*** are as follows:

Breaking Silos

DevOps breaks down the conventional style of departmentation where each task is designated to a certain team and, in effect, it used to be siloed. This, in turn, reduced flexibility and responsiveness. Going beyond the lines of organizational hierarchy, DevOps promoted mutual cooperation and communication.

Continuous Improvement

DevOps stresses continuous improvement by aligning business with IT. It strives to reduce the feedback cycle and delivery loops which, in turn, increases customer satisfaction.

Minimized Failures

When organizations integrate DevOps with fault detection techniques, it leads to minimizing failures significantly. Since DevOps is usually implemented on top of the Agile model, it promotes collaboration, modular programming, etc., making fault detection an easy task.

Also, look into this Docker Cheat Sheet that might come in handy.

Creativity and Innovation

In DevOps, teams build a culture of trust and cooperation that encourages them to improve the organizational products and services by continuously working on creativity and innovation. These attempts allow organizations to better understand and address their customer needs.

Performance-oriented Culture

With DevOps, organizations become more performance-based than power-based. This makes the workforce more creative and productive while reducing turnover and improving retention.

As we have learned about DevOps, its life cycle, and its major tools, along with their functionalities, now let's move forward and discuss a well-popular use case of DevOps in **Netflix**.

Interested in getting an industry-recognized certification in DevOps? Enroll in Intellipaat's DevOps Course in Bangalore now!

Courses you may like

DevOps Use Case in Netflix

NETFLIX We all know about Netflix, the world's leading media-subscription provider that streams various TV shows and movies on our favorite smart devices, delivering the best experience anywhere at any time to more than 75 million global customers.

Let's understand how Netflix uses DevOps to provide its customers with the best and smooth video streaming experience. Netflix uses **Spinnaker's** continuous delivery platform for the continuous delivery of its application. Before reaching Spinnaker, there are a number of steps that are supposed to take place.

Let's understand this from the diagram below:

Before going with Spinnaker for deployment, the code has to be first built and then tested.

Netflix uses **Nebula** for the build; it considers Nebula as the best build tool for Java applications. It is a collection of Gradle plugins meant for Netflix engineers to eliminate boilerplate build logic and provide sane conventions.

```
build.gradle
1  apply plugin: 'nebula'
2  apply plugin: 'war'
3  apply plugin: 'netflix.ospackage-tomcat'
4  apply plugin: 'nebula.dependency-lock'
5
6  dependencies {
7      compile 'netflix:base-server:latest.release'
8      compile 'javax.ws.rs:jsr311-api:1.1.1'
9      provided 'javax.servlet:javax.servlet-api:3.1.0'
10
11     testCompile 'junit:junit-dep:4.10'
12     testCompile 'org.mockito:mockito-all:1.9.5'
13 }
14
15 ospackage {
16     requires('apache-tomcat8')
17 }
```

A Simple Java Application Build.gradle File

Further, the code is tested locally using Nebula. The changes, if any, are committed to their central Git repository. Also, Netflix migrates its monolith application to the cloud-based microservices in AWS. The microservice architecture allows teams at Netflix to be loosely coupled, building and pushing changes at a speed they are comfortable with.

A **Jenkins** job is created that helps execute Nebula, which further builds, tests, and packages the application for further deployment. The build is further 'baked' into an Amazon Machine Image (AMI). For generating AMIs from the source, Netflix creates a bakery, and the bakery exposes an API that facilitates the creation of AMIs globally.

Once the baking is complete, **Spinnaker** comes into the picture that helps in deployment by making the resultant AMIs available for the tens, hundreds, and thousands of instances.

After this continuous integration, deployment, and final availability, the application goes live.

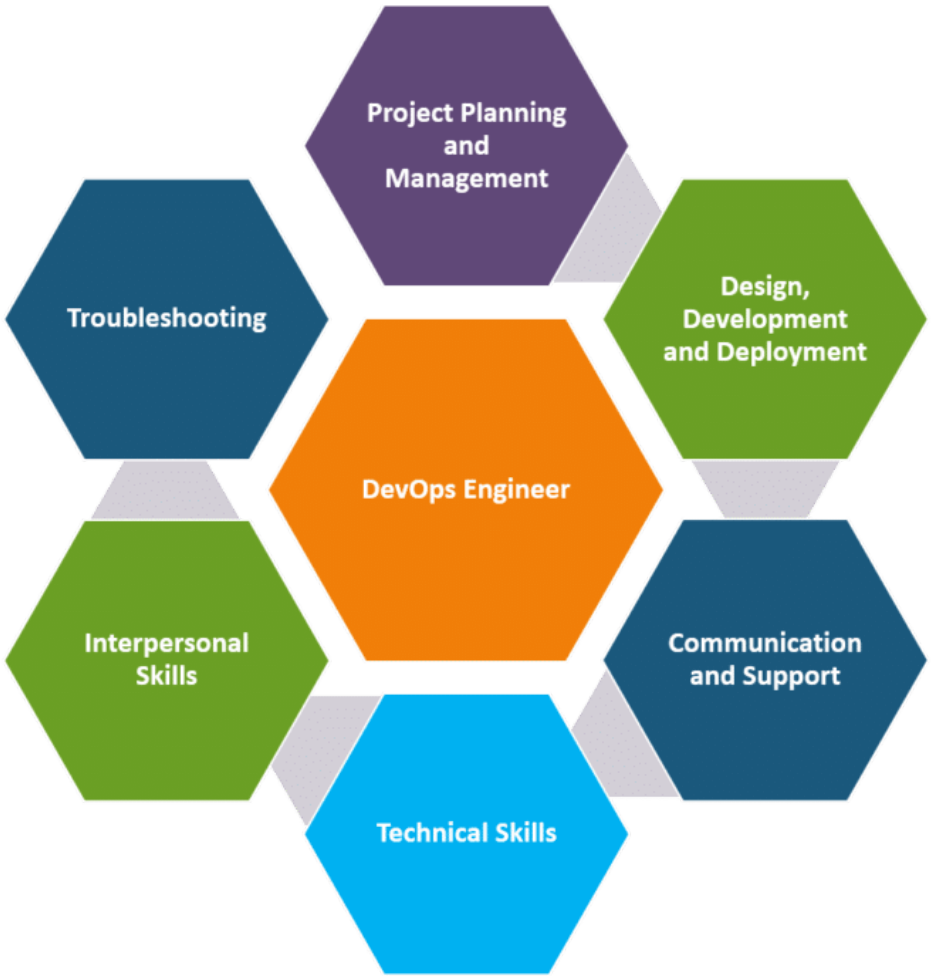
DevOps Engineers are among the highest paid professionals in the technology domain! Join DevOps Training in Hyderabad!

Since we thoroughly understand what DevOps is and how it works, now, we need to know the various DevOps Engineer skills one should possess.

Roles and Responsibilities of a DevOps Engineer

DevOps is more of a **culture** that is being incorporated by the giants in the IT world currently. DevOps, when practiced the right way keeping certain roles and responsibilities in mind, helps overcome the gap between development and operations teams.

The roles and responsibilities of DevOps Engineers are as follows:



- **Project planning and management:** In addition to monitoring software and regulating and updating tools, DevOps Engineers need to have expertise in keeping track of the costs, benefits, risks, and much more of various projects.
- **Design, development, and deployment:** DevOps Engineers are required to design, develop, and deploy automated modules for smooth functioning within the production environment, by utilizing risk-management techniques, tests, etc.
- **Communication and support**: DevOps Engineers should have exceptional communication skills that come in handy when they need to work and coordinate with different departments and provide support.
- **Technical skills**: Some basic technical experience and familiarity with configuration tools are a must.
- **Interpersonal skills**: Since DevOps Engineers are in constant interaction with other departments in the organization, they should be approachable, organized, and foresighted team players with an ability to multitask.
- **Troubleshooting**: Last but not least, one of the major responsibilities of DevOps Engineers is to troubleshoot and come up with apt solutions for various errors to benefit the firm with speed and efficiency.

The main aim behind adopting all these roles and responsibilities is to be able to perform coding, scripting, and process re-engineering.

What are career opportunities awaiting skillful DevOps Engineers? Let’s dig out.

Job Opportunities in DevOps

DevOps opens up a huge world of career options. If we are skillful enough and certified, we can go for any of the below-mentioned profiles and bag high salaries.

- **DevOps Evangelist**: Identifies the benefits coming from DevOps and thus aids in the promotion of DevOps
- **Code Release Manager**: Understands the Agile methodology and supervises the overall progress
- **Automation Architect**: Designs and builds automated tools and systems to implement continuous and smooth deployments
- **Experience Assurance**: Enhances user experience by finding bugs and including all the essential features in the applications
- **Software Developer/Tester**: Makes sure that the code meets all the original business requirements, along with performs testing and monitoring
- **Security Engineer**: Integrates security into the applications and products to keep the business safe

Conclusion

From this extensive DevOps tutorial, we acquired a detailed understanding of DevOps, its life cycle, various DevOps tools, its use cases, and most importantly the roles and responsibilities of DevOps Engineers.

While here we covered quite a bit of the core functionality of DevOps, there is still a lot more to know. If you're looking forward to learning more about it, then you must go for a structured **DevOps Certification Training** provided by Intellipaat, where you will work on various case-based scenarios, along with exhaustive topic-wise assignments, hands-on sessions, and various industrial-based projects, which would prepare you for getting placed in top-notch companies.

This course will help you understand DevOps concepts, the most important tools and frameworks that you must learn to use to become a successful and productive DevOps team member at your workplace.

Do let us know in the comment section if this DevOps tools tutorial for beginners was helpful to you.