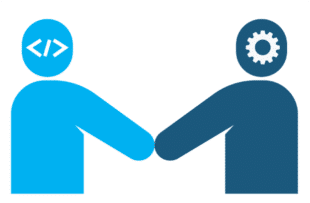
**What exactly is DevOps?**

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

[](https://intellipaat.com/mediaFiles/2017/04/d1-310x203.png)

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

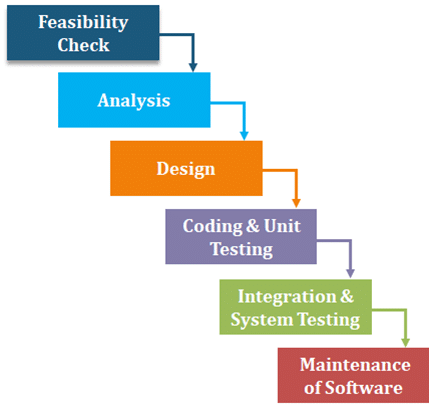
**Why DevOps?**

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](https://intellipaat.com/blog/what-is-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.

**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:[](https://intellipaat.com/mediaFiles/2018/12/d2.png)

* **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.
* **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.

**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.

**Agile Software Development**

[**Agile**](https://intellipaat.com/blog/what-is-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**.

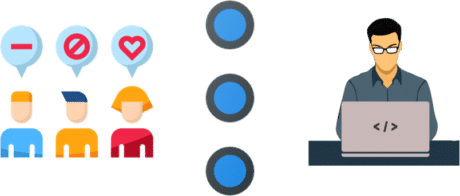
**Advantages**

* 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**

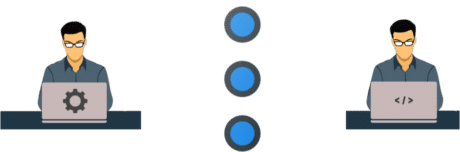
* There is less predictability in Agile
* It requires more time and commitment from every stakeholder. Testers, developers, and customers must interact with each other constantly and should agree to each other’s decisions in order to get the task done, and hence Agile is time-consuming
* Limited documentation 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

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[](https://intellipaat.com/mediaFiles/2018/12/d3.png)

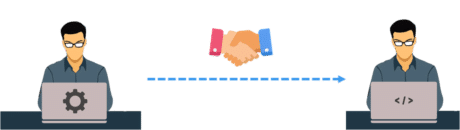
*Gaps Between Customers and Developers*

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

[](https://intellipaat.com/mediaFiles/2018/12/d4.png)*Gaps Between the Operations Team and Developers*

[](https://intellipaat.com/mediaFiles/2018/12/d5.png)

***How do you think it was overcome?***

It was in this scenario [**DevOps was introduced**](https://intellipaat.com/blog/tutorial/devops-tutorial/devops-introduction/) in order to overcome the gap between developers and the operations team.  
[](https://intellipaat.com/mediaFiles/2018/12/d6-1.png)

**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?

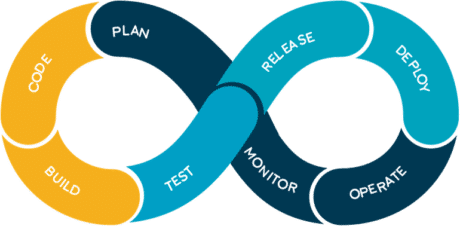
Let’s move to the next section in this DevOps tutorial and check out the DevOps life cycle. This way, we can understand how DevOps works.

**DevOps Lifecycle**

DevOps focuses on bringing all the development, operations, and [***IT infrastructure***](https://intellipaat.com/blog/serverless-computing-next-step-cloud-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, continuously monitoring, and fetching feedback from end-users. This process of developing, testing, deploying, and monitoring keeps on repeating for better results.

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

[](https://intellipaat.com/mediaFiles/2018/12/d7-1.png)

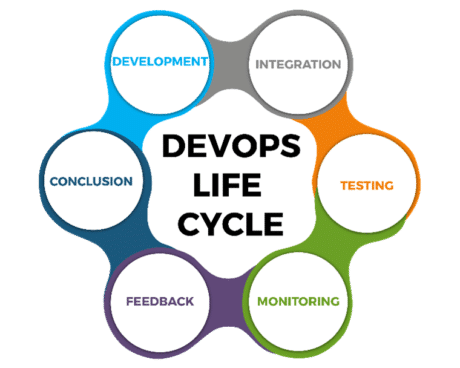
* **Code:**The first step in the DevOps life cycle is coding, where developers build the code on any platform
* **Build:**Developers build the version of their program in any extension depending upon the language they are using
* **Test:**For DevOps to be successful, the testing process must be automated using any [**DevOps automation tool**](https://intellipaat.com/blog/devops-automation-tools/) 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***](https://intellipaat.com/community/devops-and-agile)***!***

**Different Lifecycle Stages**

Now, let’s discuss the different stages in the [DevOps life cycle](https://intellipaat.com/blog/what-is-devops-lifecycle-and-process/) 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.  
[](https://intellipaat.com/mediaFiles/2018/12/d8-1.png)

**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**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-tutorial/)to maintain different versions of the code. To build/package the code into an executable file, we can use a reliable tool, namely, **Maven**.

**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**](https://intellipaat.com/blog/what-is-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**](https://intellipaat.com/blog/what-is-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**](https://intellipaat.com/blog/tutorial/splunk-tutorial/introduction-of-splunk/)**,**[**ELK Stack**](https://intellipaat.com/blog/what-is-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.

**DevOps Tools**

The most popular [DevOps tools](https://intellipaat.com/blog/top-devops-tools/) are discussed below.

[](https://intellipaat.com/mediaFiles/2018/12/d17.jpg)

* [**Puppet**](https://intellipaat.com/blog/tutorial/devops-tutorial/puppet-tutorial/): 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**](https://intellipaat.com/blog/tutorial/devops-tutorial/docker-tutorial/): 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**](https://intellipaat.com/blog/tutorial/devops-tutorial/jenkins-tutorial/): 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**](https://intellipaat.com/blog/what-is-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.

**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***](https://intellipaat.com/blog/benefits-of-devops/#:~:text=DevOps%20helps%20in%20introducing%20automation,and%20human%2Derror%2Dfree.) 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.

**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**.