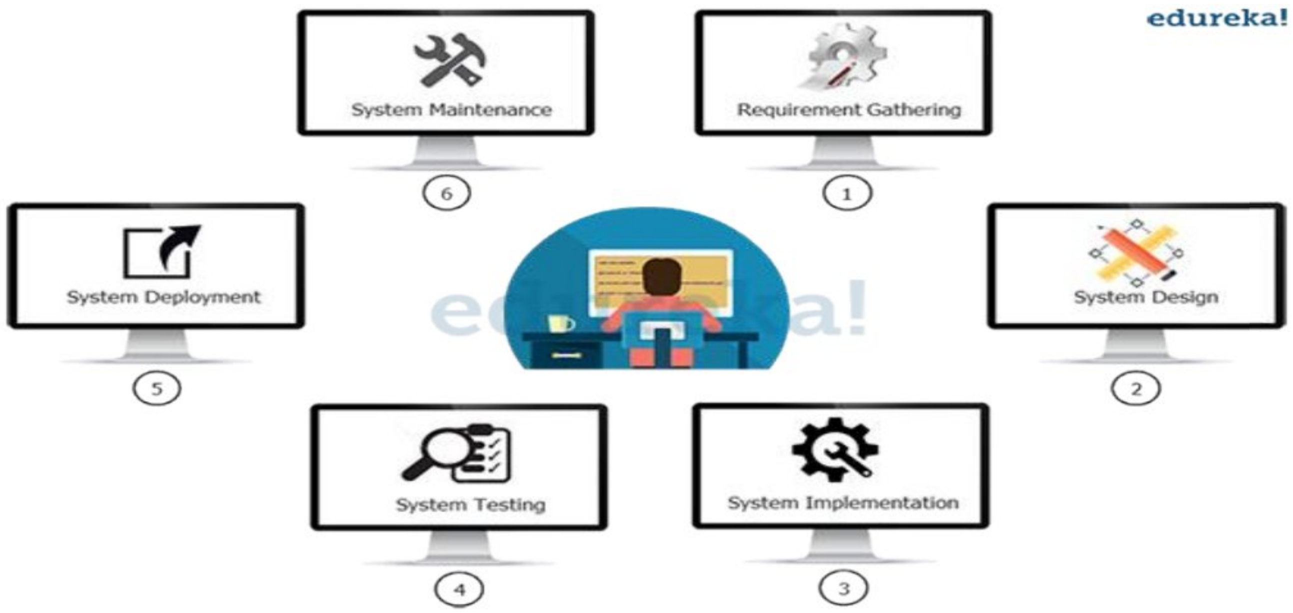
DEVOPS

The DevOps Tutorial

Waterfall Model

Let’s consider developing software in a traditional way using a Waterfall Model.

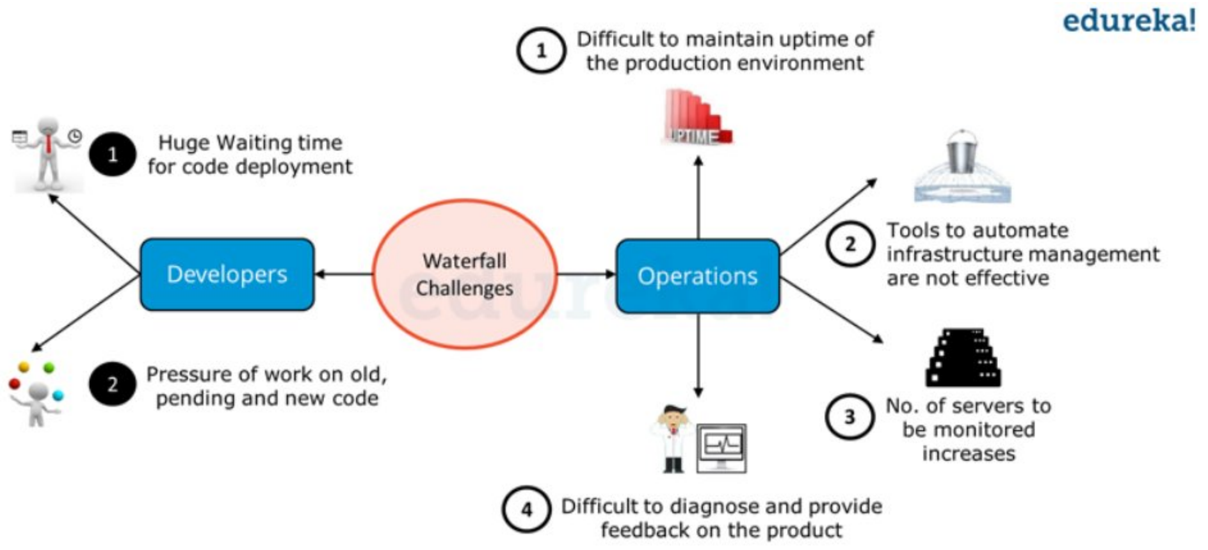


In the above diagram you will see the phases it will involve:

- In phase 1 – Complete Requirement is gathered and SRS is developed
- In phase 2 – This System is Planned and Designed using the SRS
- In phase 3 – Implementation of the System takes place
- In phase 4 – System is tested and its quality is assured
- In phase 5 – System is deployed to the end users
- In phase 6 – Regular Maintenance of the system is done

Waterfall Model Challenges

The Water-fall model worked fine and served well for many years however it had some challenges. In the following diagram the challenges of Waterfall Model are highlighted.


edureka!
1 Difficult to maintain uptime of the production environment
UPTIME
1 Huge Waiting time for code deployment
Developers
Waterfall Challenges
Operations
2 Tools to automate infrastructure management are not effective
2 Pressure of work on old, pending and new code
3 No. of servers to be monitored increases
4 Difficult to diagnose and provide feedback on the product

In the above diagram you can see that both Development and Operations had challenges in the Waterfall Model. From Developers point of view there were majorly two challenges:

1 After Development, the code deployment time was huge.

2 Pressure of work on old, pending and new code was high because development and deployment time was high.

On the other hand, Operations was also not completely satisfied. There were four major challenges they faced as per the above diagram:

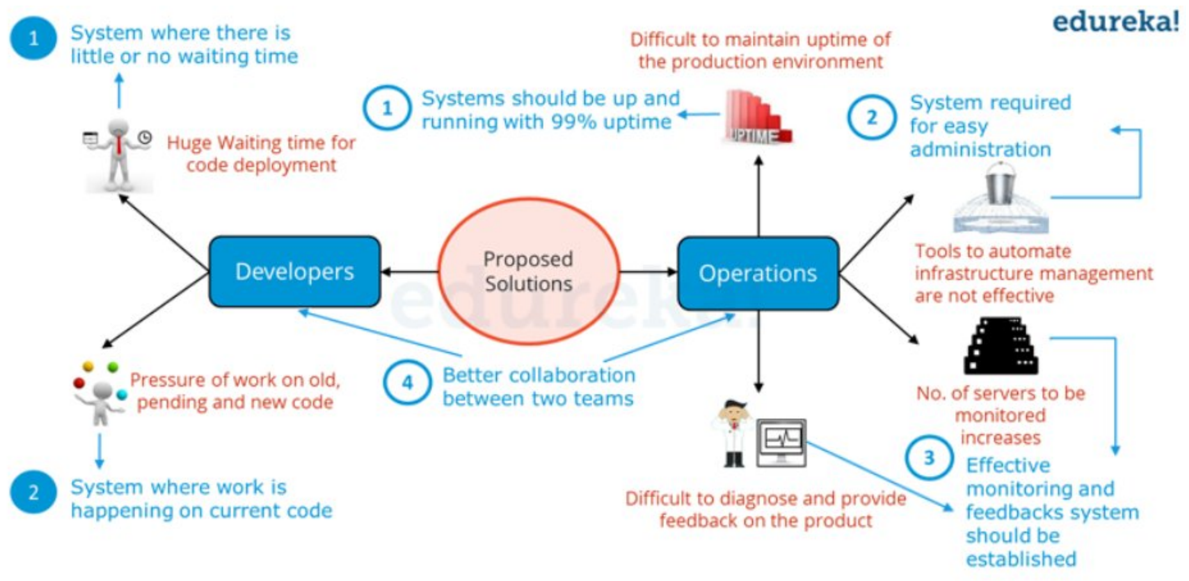
1 It was difficult to maintain ~100% uptime of the production environment.

2 Infrastructure Automation tools were not very affective.

3 Number of severs to be monitored keeps on increasing with time and hence the complexity.

4 It was very difficult to provide feedback and diagnose issue in the product.

In the following diagram proposed solution to the challenges of Waterfall Model are highlighted.


edureka!
1 System where there is little or no waiting time
Huge Waiting time for code deployment
Pressure of work on old, pending and new code
2 System where work is happening on current code
Developers
Proposed Solutions
Operations
4 Better collaboration between two teams
Difficult to maintain uptime of the production environment
1 Systems should be up and running with 99% uptime
UPTIME
2 System required for easy administration
Tools to automate infrastructure management are not effective
No. of servers to be monitored increases
3 Effective monitoring and feedbacks system should be established
Difficult to diagnose and provide feedback on the product

In the above diagram, Probable Solutions for the issues faced by Developers and Operations are highlighted in blue. This sets the guidelines for an Ideal Software Development strategy.

From Developers point of view:

1 A system which enables code deployment without any delay or wait time.

2 A system where work happens on the current code itself i.e. development sprints are short and well planned.

From Operations point of view:

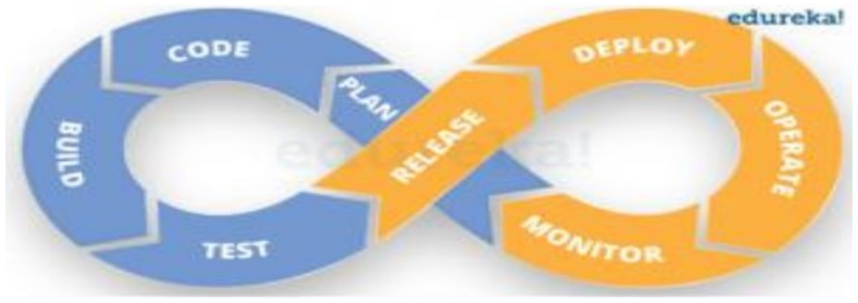
1 System should have at-least 99% uptime.

2 Tools & systems are there in place for easy administration.

3 Effective monitoring and feedbacks system should be there.

4 Better Collaboration between Development & Operations and is common requirement for Developers and Operations team.

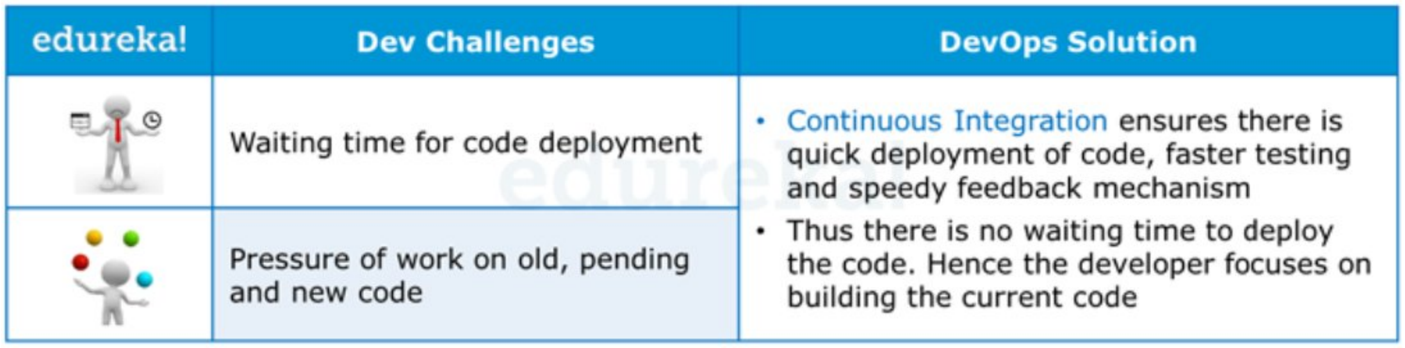
DevOps integrates developers and operations team to improve collaboration and productivity.



According to the DevOps culture, a single group of Engineers (developers, system admins, QA's. Testers etc turned into DevOps Engineers) has end to end responsibility of the Application (Software) right from gathering the requirement to development, to testing, to infrastructure deployment, to application deployment and finally monitoring & gathering feedback from the end users, then again implementing the changes.

This is a never ending cycle and the logo of DevOps makes perfect sense to me. Just look at the above diagram – What could have been a better symbol than infinity to symbolize DevOps?

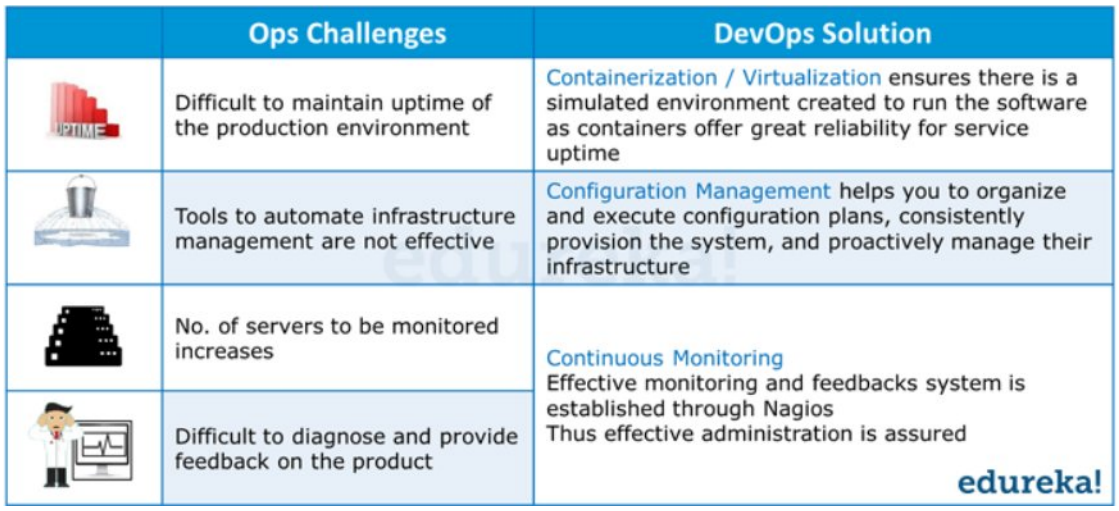
Now let us see how DevOps takes care of the challenges faced by Development and Operations. Below table describes how DevOps addresses Dev Challenges.



edureka!	Dev Challenges	DevOps Solution
	Waiting time for code deployment	• Continuous Integration ensures there is quick deployment of code, faster testing and speedy feedback mechanism • Thus there is no waiting time to deploy the code. Hence the developer focuses on building the current code
	Pressure of work on old, pending and new code	

DevOps Tutorial Table 1 – Above table states how DevOps solves Dev Challenges

Going further, below table describes how DevOps addresses Ops Challenges.

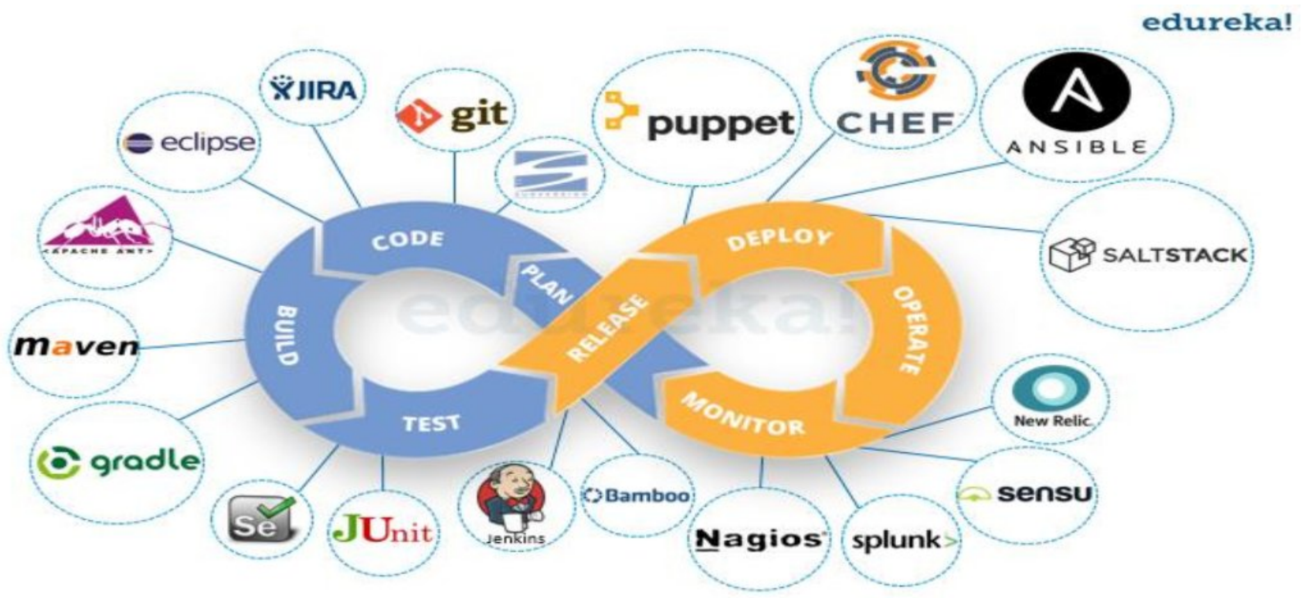


	Ops Challenges	DevOps Solution
	Difficult to maintain uptime of the production environment	Containerization / Virtualization ensures there is a simulated environment created to run the software as containers offer great reliability for service uptime
	Tools to automate infrastructure management are not effective	Configuration Management helps you to organize and execute configuration plans, consistently provision the system, and proactively manage their infrastructure
	No. of servers to be monitored increases	Continuous Monitoring Effective monitoring and feedbacks system is established through Nagios Thus effective administration is assured
	Difficult to diagnose and provide feedback on the product	

edureka!

However, you would still be wondering, how to implement DevOps. To expedite and actualize DevOps process apart from culturally accepting it, one also needs various DevOps tools like Puppet, Jenkins, GIT, Chef, Docker, Selenium, AWS etc to achieve automation at various stages which helps in achieving Continuous Development, Continuous Integration, Continuous Testing, Continuous Deployment, Continuous Monitoring to deliver a quality software to the customer at a very fast pace.

Now take a look at the below DevOps diagram with various DevOps Tools closely and try to decode it.



These tools has been categorized into various stages of DevOps. Hence it is important that I first tell you about DevOps stages and then talk more about DevOps Tools.

DevOps Lifecycle can be broadly broken down into the below DevOps Stages:

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

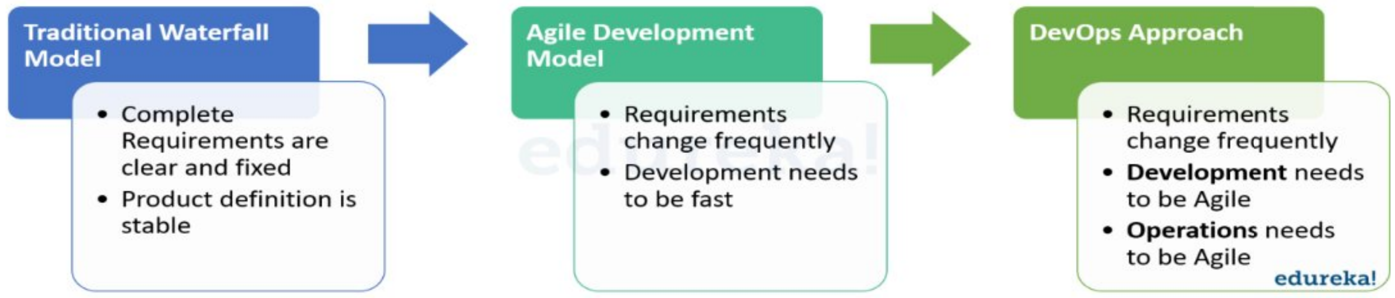
These stages are the building blocks to achieve DevOps as a whole.

The intention of DevOps is to create better-quality software more quickly and with more reliability while causing greater communication and collaboration between teams. This is the key to understanding what is DevOps!

Leading organizations across the world have adopted DevOps methodologies to overhaul their performance, security and team dynamics. With more and more companies jumping on to the DevOps bandwagon, it has emerged as a hot skill to master in 2016. In this blog, let us find out what is DevOps and why is it such a big deal! We will do this by first tracing the evolution of software development methodologies leading to DevOps, then exploring what is DevOps and its life cycle, and finish up by evaluating how top companies such as Facebook areusing DevOps to their benefit.

Evolution of Software Development

DevOps evolved from existing software development strategies/methodologies over the years in response to business needs. Let us briefly look at how these models evolved and in which scenarios they would work best.

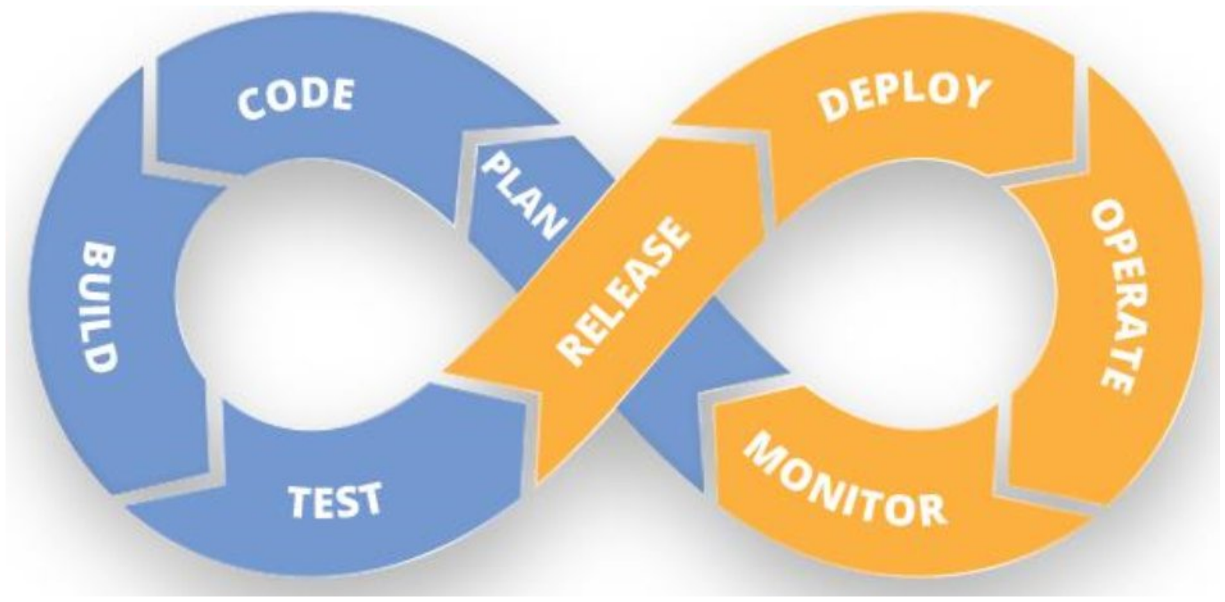

Traditional Waterfall Model
• Complete Requirements are clear and fixed
• Product definition is stable
Agile Development Model
• Requirements change frequently
• Development needs to be fast
DevOps Approach
• Requirements change frequently
• Development needs to be Agile
• Operations needs to be Agile
edureka!

The slow and cumbersome Waterfall model evolved into Agile which saw development teams working on the software in short sprints lasting not more than two weeks. Having such a short release cycle helped the development team work on client feedback and incorporate it along with bug fixes in the next release. While this Agile SCRUM approach brought agility to development, it was lost on Operations which did not come up to speed with Agile practices. Lack of collaboration between Developers and Operations Engineers still slowed down the development process and releases. DevOps methodology was born out of this need for better collaboration and faster delivery. DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems.

What is DevOps?

DevOps is a Software Development approach which involves Continuous Development, Continuous Testing, Continuous Integration, Continuous Deployment and Continuous Monitoring of the software throughout its development life cycle. These activities are possible only in DevOps, not Agile or waterfall, and this is why Facebook and other top companies have chosen DevOps as the way forward for their business goals. DevOps is the preferred approach to develop high quality software in shorter development cycles which results in greater customer satisfaction. Check out the below video on What is DevOps before you go ahead.

Your understanding of what is DevOps is incomplete without learning about its life cycle. Let us now look at the DevOps life cycle and explore how they are related to the Software Development stages depicted in the diagram below.



Continuous Development:

This is the stage in the DevOps life cycle where the Software is developed continuously. Unlike the Waterfall model the software deliverables are broken down into multiple sprints of short development cycles, developed and then delivered in a very short time. This stage involves the Coding and Building phases and makes use of tools such as Git and SVN for maintaining the different versions of the code, and tools like Ant, Maven, Gradle for building / packaging the code into an executable file that can be forwarded to the QAs for testing.

Continuous Testing:

It is the stage where the developed software is continuously tested for bugs. For Continuous testing testing automation tools like Selenium, JUnit etc are used. These tools enables the QA's for testing multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality. In this phase use of Docker containers for simulating testing environment on the fly, is also a preferred choice. Once the code is tested, it is continuously integrated with the existing code.

Continuous Integration:

This is the stage where the code supporting new functionality is integrated with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end users. The changed code, should also ensure that there are no errors in the runtime environment, allowing us to test the changes and check how it reacts with other changes. Jenkins is a very popular tool used for Continuous Integration. Using Jenkins one can pull the latest code revision from GIT repository and produce a build which can finally be deployed to test or production server. It can be set to trigger a new build automatically as soon as there is change in the GIT repository or can be triggered manually on click of a button.

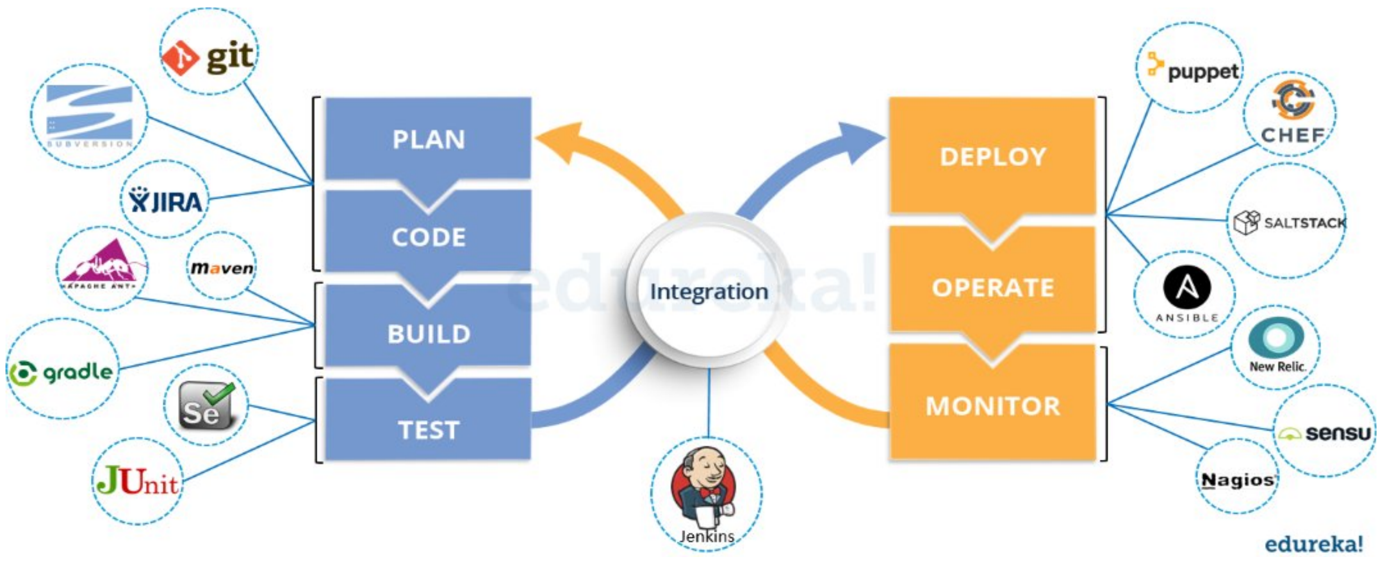
Continuous Deployment:

It is the stage where the code is deployed to the production environment. Here we ensure that the code is correctly deployed on all the servers. If there is any addition of functionality or a new feature is introduced then one should be ready to welcome greater website traffic. So it is also the responsibility of the SysAdmin to scale up the servers to host more users. Since the new code is deployed on a continuous basis, automation tools play an important role for executing tasks quickly and frequently. Puppet, Chef, SaltStack and Ansible are some popular tools that are used in this stage.

Continuous Monitoring:

This is a very crucial stage in the DevOps life cycle which is aimed at improving the quality of the software by monitoring its performance. This practice involves the participation of the Operations team who will monitor the user activity for bugs / any improper behavior of the system. This can also be achieved by making use of dedicated monitoring tools which will continuously monitor the application performance and highlight issues. Some popular tools used are Nagios, NewRelic and Sensu. These tools help you monitor the application and the servers closely to check the health of the system proactively. They can also improve productivity and increase the reliability of the systems, reducing IT support costs. Any major issues found could be reported to the Development team so that it can be fixed in the continuous development phase.

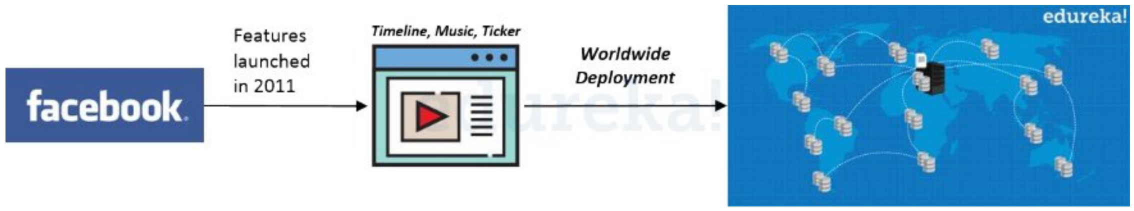
These DevOps stages are carried out on loop continuously until the desired product quality is achieved. The diagram given below will show you which tools can be used in which stage of the DevOps life cycle.



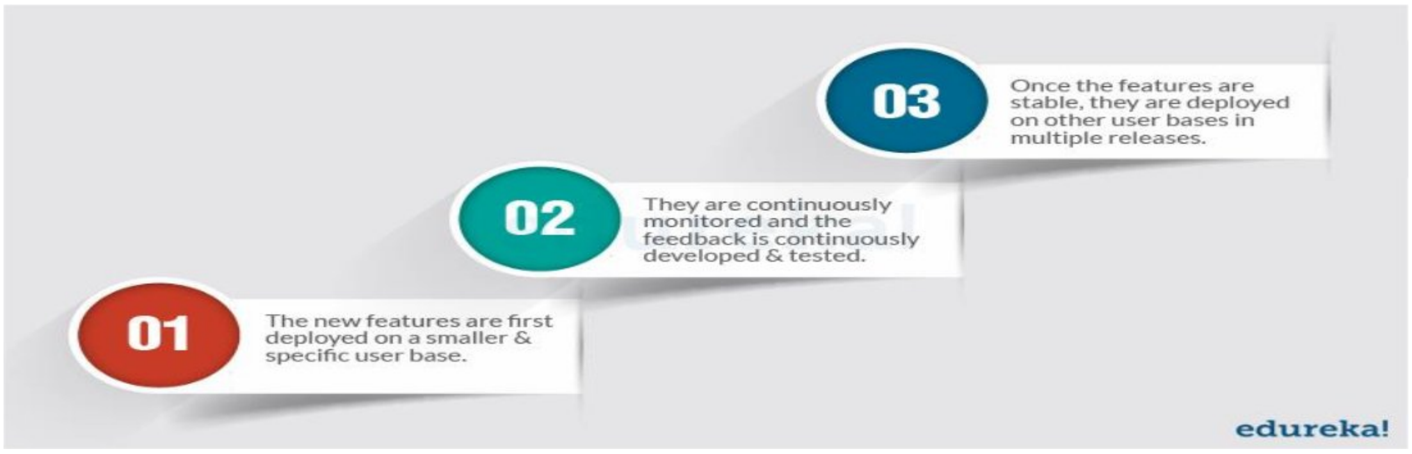
Now that we have established the significance of DevOps and learnt about its different stages along with the DevOps tools involved, let us now look at a Facebook case study and understand why they moved from Agile to DevOps. We will do this by taking up the use case of Facebook's 2011 roll-out of new features that resulted in them reassessing their product delivery and taking on the DevOps approach. But before go through the below video on DevOps Tools.

DevOps Case Study: Facebook

In 2011, Facebook rolled out a slew of new features – timeline, ticker and music functionalities – to its 500 million users spread across the globe. The huge traffic that was generated on Facebook following the release led to a server meltdown. The features that were rolled out garnered mixed response from users which led to inconclusive results of the effectiveness of the new features, leaving them with no actionable insights.

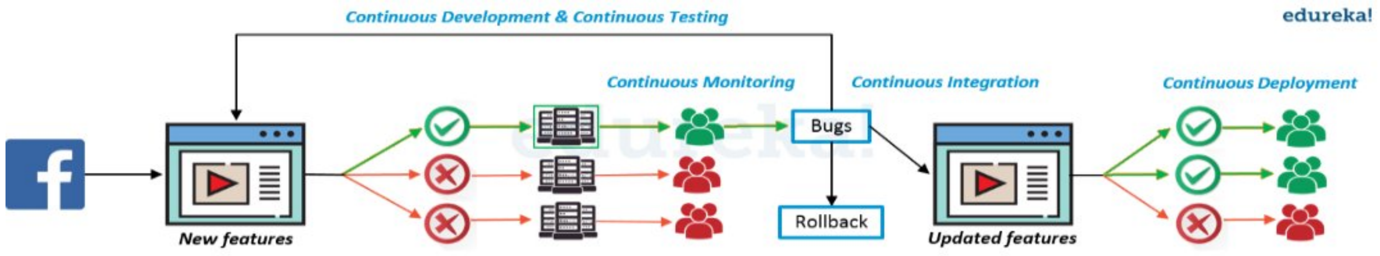


This led to an evaluation and reassessment of strategies, resulting in Facebook coming up with the Dark Launching Technique. Using the DevOps principles, Facebook created the following methodology for the launch of its new releases.


01
The new features are first deployed on a smaller & specific user base.
02
They are continuously monitored and the feedback is continuously developed & tested.
03
Once the features are stable, they are deployed on other user bases in multiple releases.
edureka!

Facebook Dark Launching Technique

Dark launching is the process of gradually rolling out production-ready features to a select set of users before a full release. This allows development teams to get user feedback early on, test bugs, and also stress test infrastructure performance. A direct result of continuous delivery, this method of release helps in faster, more iterative releases that ensure that application performance does not get affected and that the release is well received by customers.



In the Dark Launching technique, features are released to a small user base through a dedicated deployment pipeline. In the below given diagram of Facebook Dark Launch, you can see that that only one deployment pipeline is turned on to deploy the new features to a select set of users. The remaining hundreds of pipelines are all turned off at this point. The specific user base on which the features have been deployed are continuously monitored to collect feedback and identify bugs. These bugs and feedback will be incorporated in development, tested and deployed on the same user base until the features becomes stable. Once stability is achieved, the features will be gradually deployed on other user bases by turning on other deployment pipelines.

Facebook does this by wrapping code in a feature flag or feature toggle which is used to control who gets to see the new feature and when. This exposes pain points and areas of the application's infrastructure that needs attention prior to the full-fledged launch while still simulating the full effect of launching the code to users. Once the features are stable, they are deployed to the rest of the users over multiple releases.

This way Facebook has a controlled or stable mechanism for developing new functionality to its massive user base. On the contrary if the feature does not get a good response they have an option to rollback on their deployments altogether. This also helps them to prepare their servers for deployment as they can predict the user activity on their website and they can scale up their servers accordingly. The diagram given above depicts how a dark launch takes place at Facebook.

Facebook, Amazon, Netflix and Google, along with many leading tech giants, use dark launches to gradually release and test new features to a small set of their users before releasing to everyone.

The intention of DevOps is to create better-quality software more quickly and with more reliability while inviting greater communication and collaboration between teams. It is also an automation process that allows quick, safe and high quality software development and release while keeping all the stakeholders in the loop. This is the real reason why DevOps is seeing an all-time high adoption leading to increasing career opportunities in DevOps.