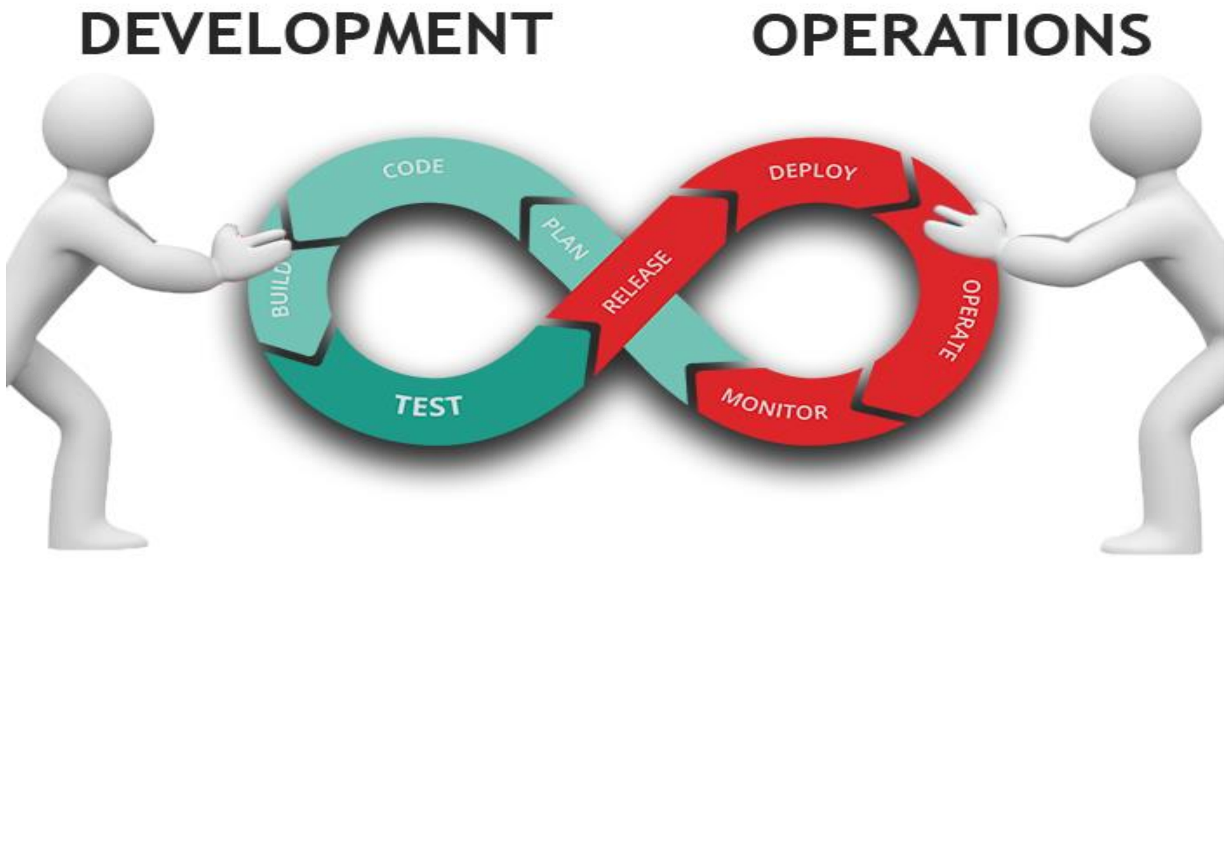
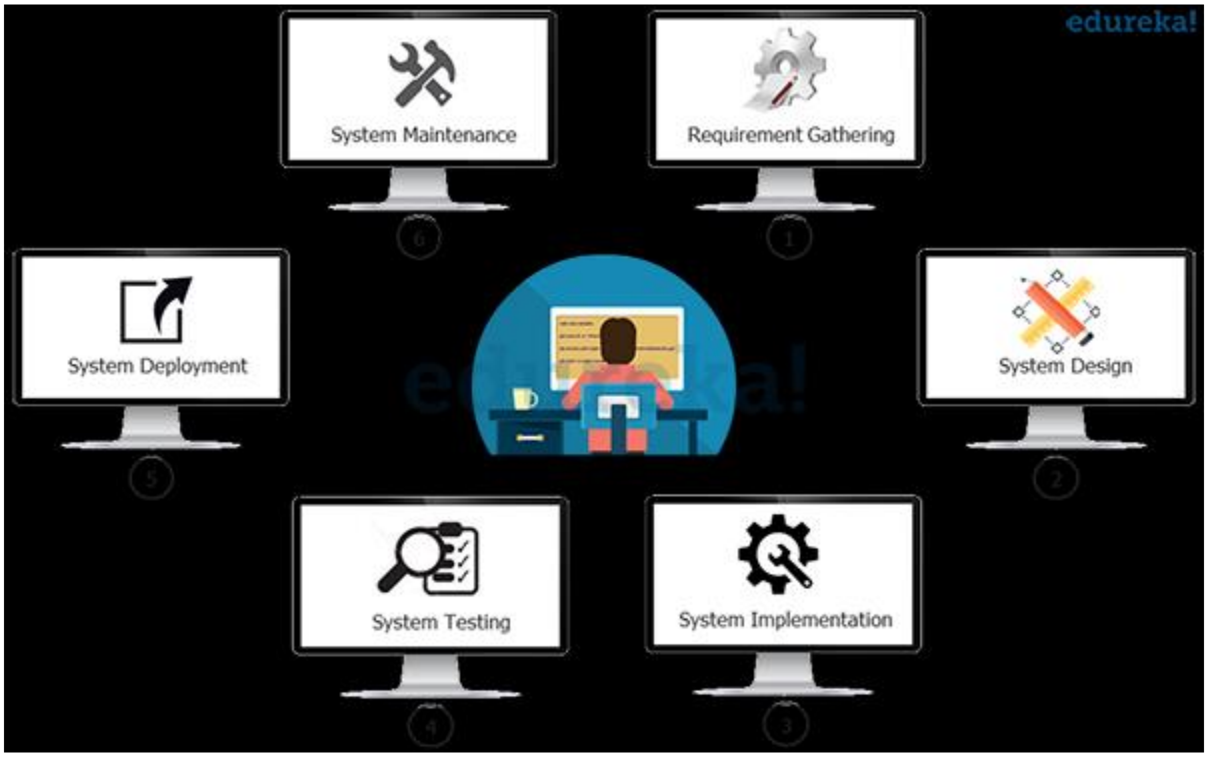
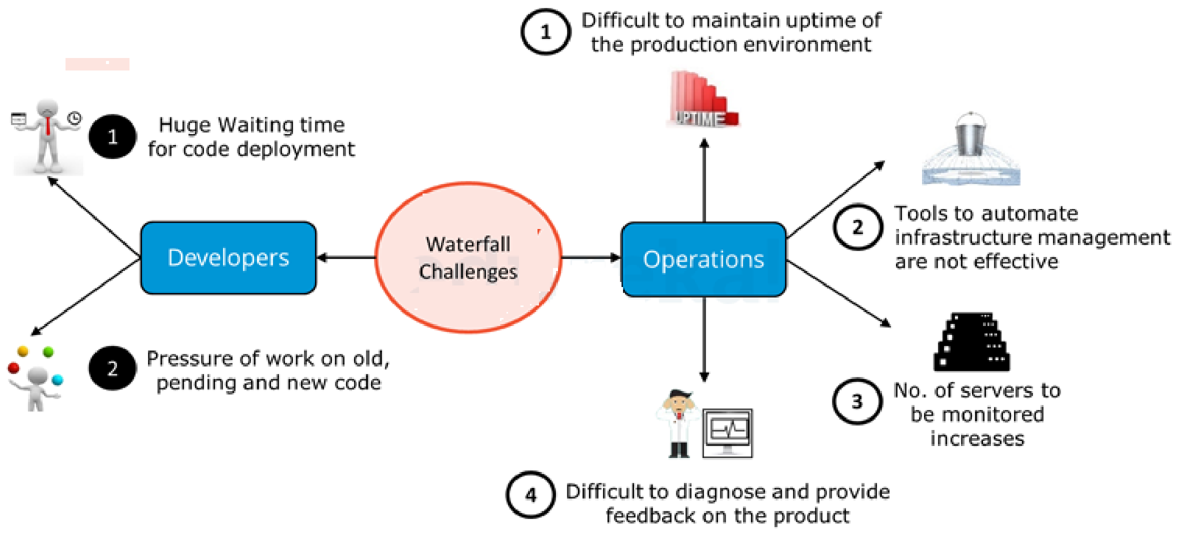

DEVELOPMENT
OPERATIONS
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

Waterfall Model:



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.


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

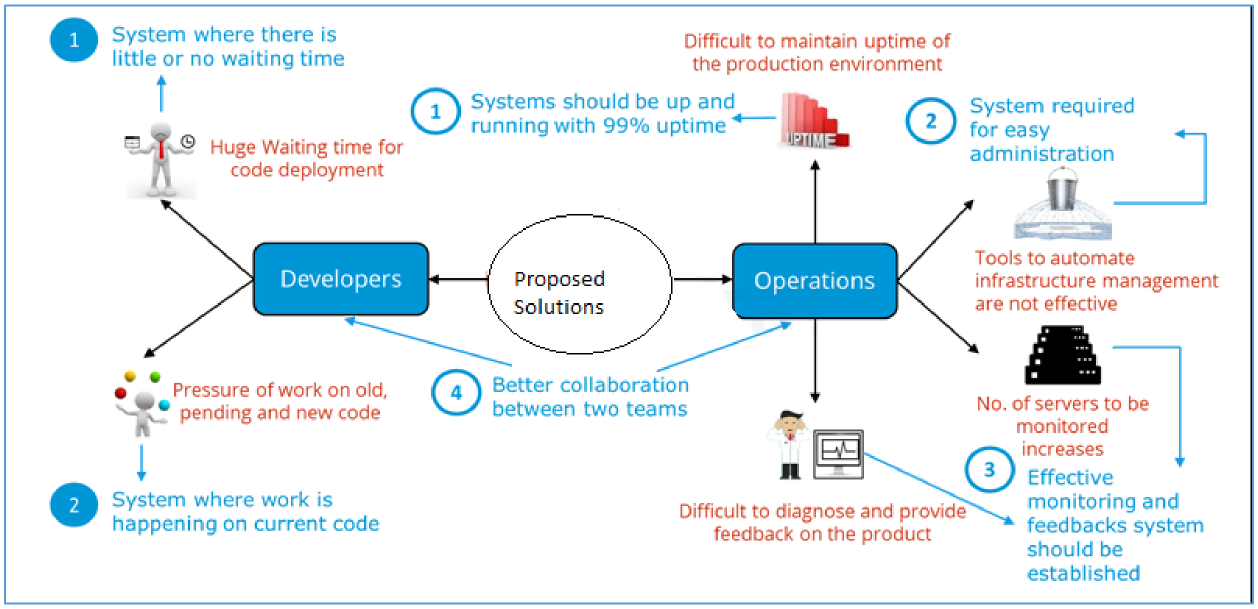
In the above diagram you can see that both Development and Operations had challenges in the Waterfall Model. From Developers point of view there were majorly two challenges:

- After Development, the code deployment time was huge.
- 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:

- It was difficult to maintain ~100% uptime of the production environment.
- Infrastructure Automation tools were not very affective.
- Number of severs to be monitored keeps on increasing with time and hence the complexity.
- 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.


1 System where there is little or no waiting time
Huge Waiting time for code deployment
Developers
Proposed Solutions
Operations
Pressure of work on old, pending and new code
2 System where work is happening on current code
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.

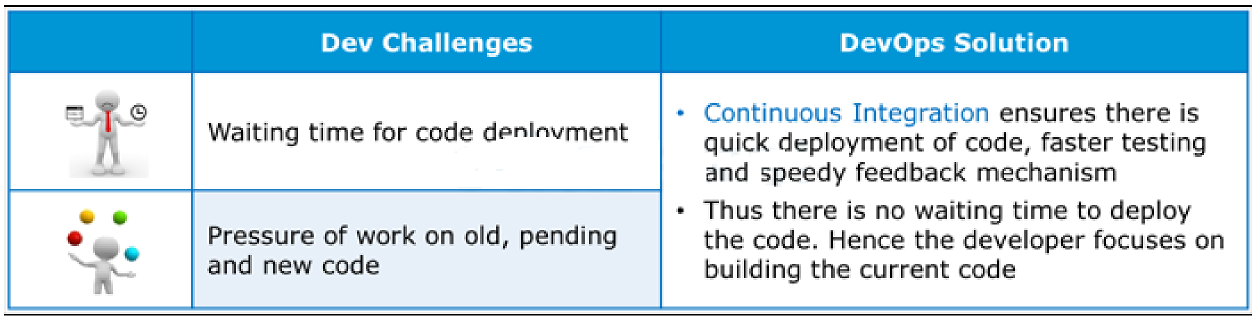
From Developers point of view:

- A system which enables code deployment without any delay or wait time.
- A system where work happens on the current code itself i.e. development sprints are short and well planned.

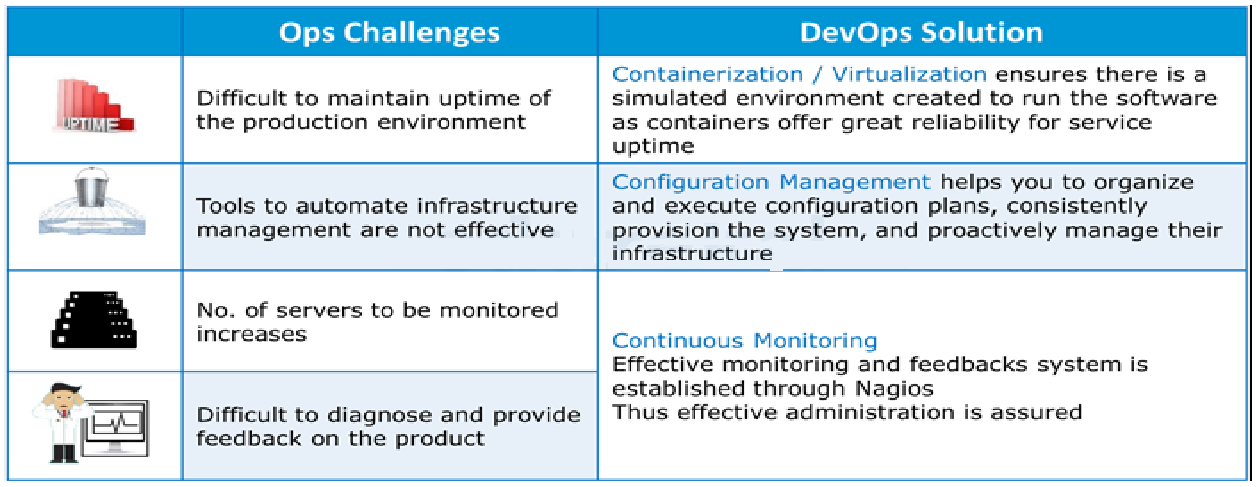
From Operations point of view:

- System should have at-least 99% uptime.
- Tools & systems are there in place for easy administration.
- Effective monitoring and feebacks system should be there.
- Better Collaboration between Development & Operations and is common requirement for Developers and Operations team.

Below table describes how DevOps addresses Ops Challenges.



	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	

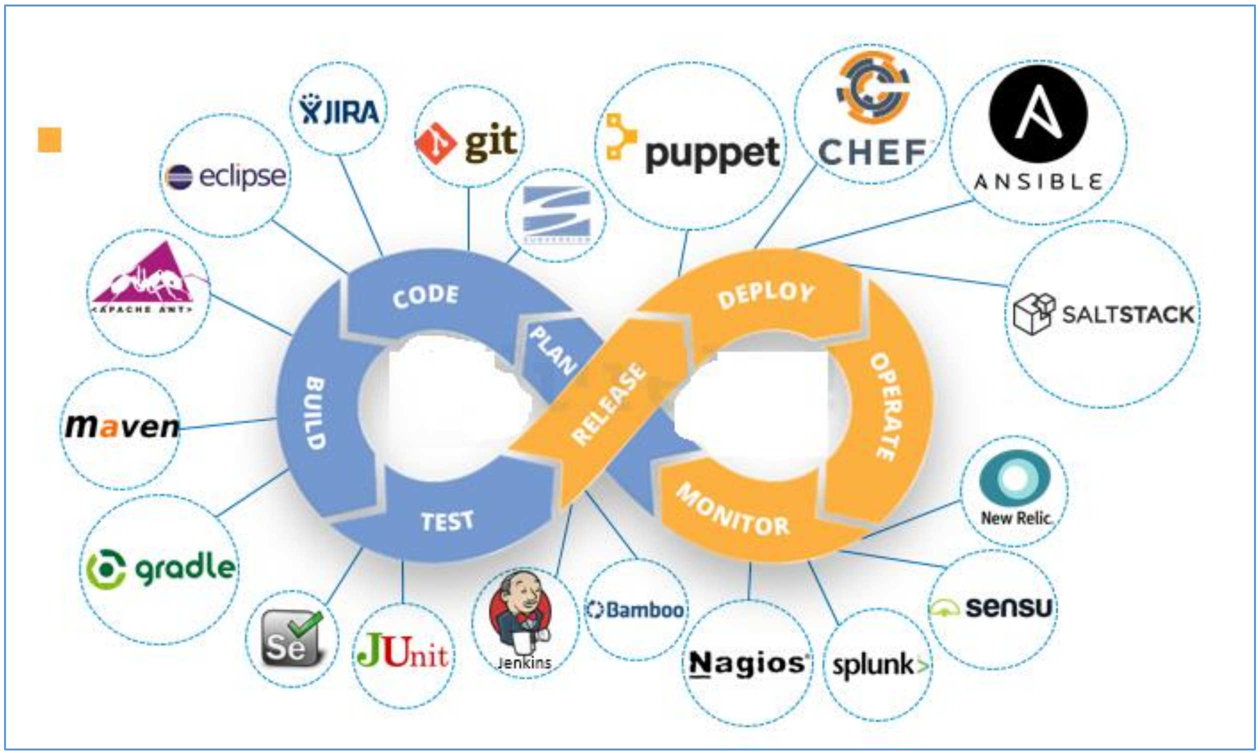


	Ops Challenges	DevOps Solution
	Difficult to maintain uptime of the production environment	Containerization / Virtualization ensures there is a simulated environment created to run the software as containers offer great reliability for service uptime
	Tools to automate infrastructure management are not 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	

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

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

Overall Picture of Devops tools:



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.

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. J
- enkins 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.