Introduction to DevOps

By
ABHISHK K L
Assistant Professor
Dept Of MCA
RIT, Bengaluru

INTRODUCTION

What is DevOps?

DevOps is the combination of **cultural philosophies, practices, and tools** which **collaborates Development** and **IT Operations** to make software production and deployment in an **automated & repeatable** way to **increase** the **organization's speed** to **deliver software applications and services**.

Why is DevOps is Needed?

- Before DevOps, the development and operation team worked in complete isolation.
- Testing and Deployment were isolated activities done after design-build. Hence they consumed more time than actual build cycles.
- Without using DevOps, team members are spending a large amount of their time in testing, deploying, and designing instead of building the project.
- Manual code deployment leads to human errors in production
- Coding & operation teams have their separate timelines and are not in synch causing further delays.

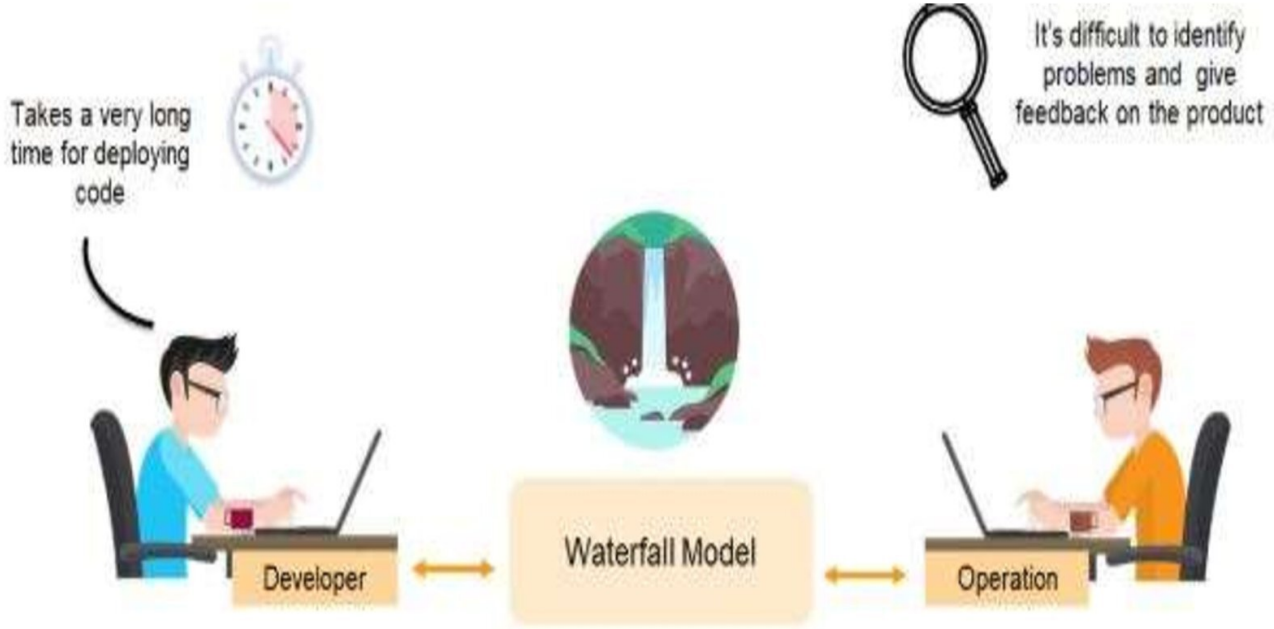
To bride these gaps DevOps is needed

Note: There is a demand to increase the rate of software delivery by business stakeholders. As per Forrester Consulting Study, Only 17% of teams can use delivery software fast enough. This proves the pain point.

Why is DevOps is Needed?

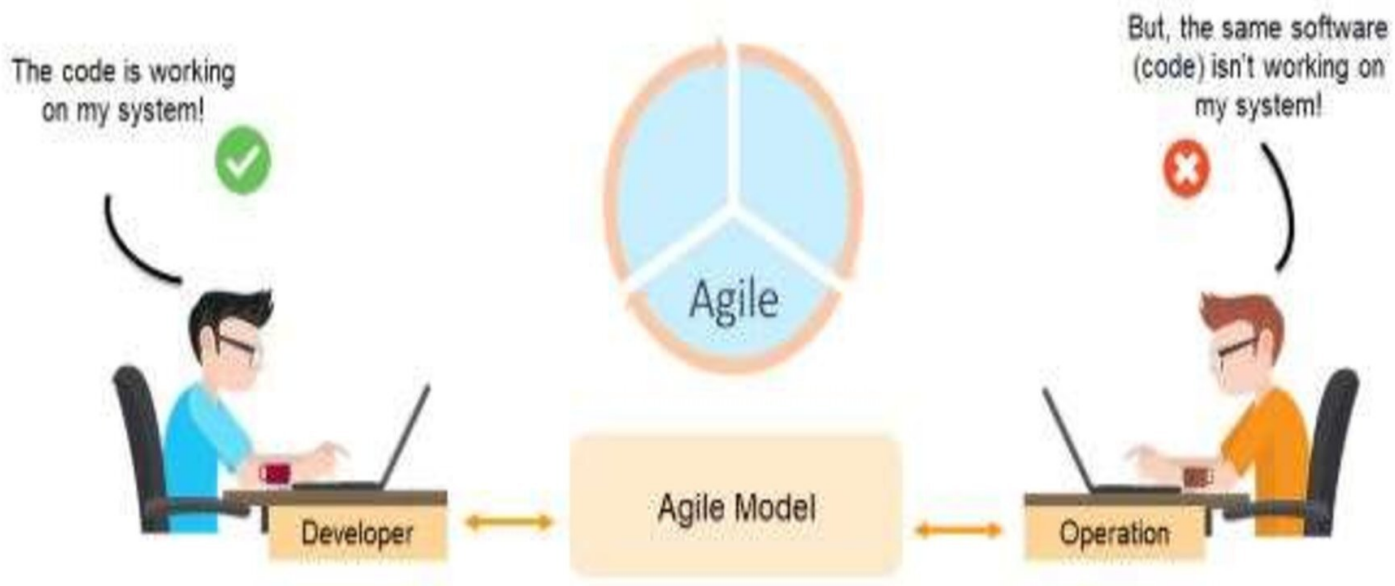
From the figure below, we can see the issues with the waterfall model:

- **The developer took a very long time to deploy code**
- **On the operations side, the tester found it challenging to identify problems and give useful feedback**



Why is DevOps is Needed?

Using the agile method, the code that works for the developer may not work for the operations team.

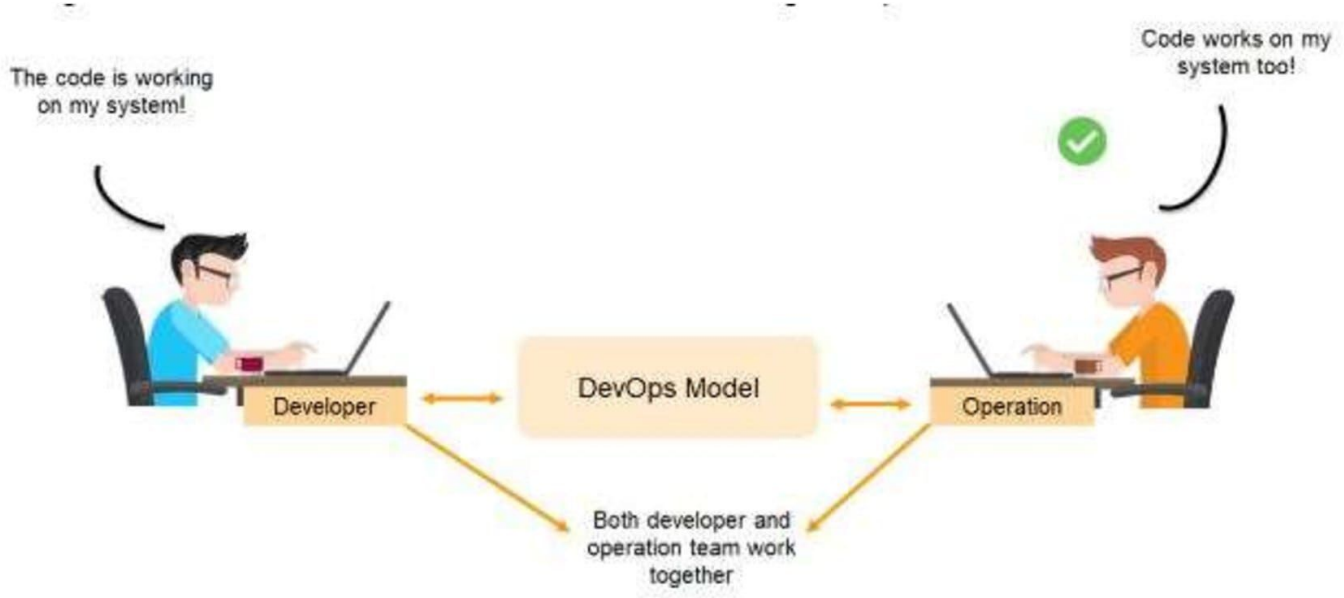


So how can this issue be solved?

Why is DevOps is Needed?

With DevOps, there is continuous integration between deployment of code and the testing of it. Near real-time monitoring and immediate feedback through a DevOps continuous monitoring tool enables both the developer and operations team work together.

The figure below shows how well the software is handled using DevOps.



When to adopt DevOps?

DevOps should be used for large distributed applications such as eCommerce sites or applications hosted on a cloud platform.

When not to adopt DevOps?

It should not be used in a mission-critical application like bank, power and other sensitive data sites. Such applications need strict access controls on the production environment, a detailed change management policy, access control policy to the data centers.

How is DevOps different from traditional IT?

let's compare traditional software waterfall model with DevOps to understand the changes DevOps bring.

We assume the application is scheduled to go live in 2 weeks and coding is 80% done. We assume the application is a fresh launch and the process of buying servers to ship the code has just begun-

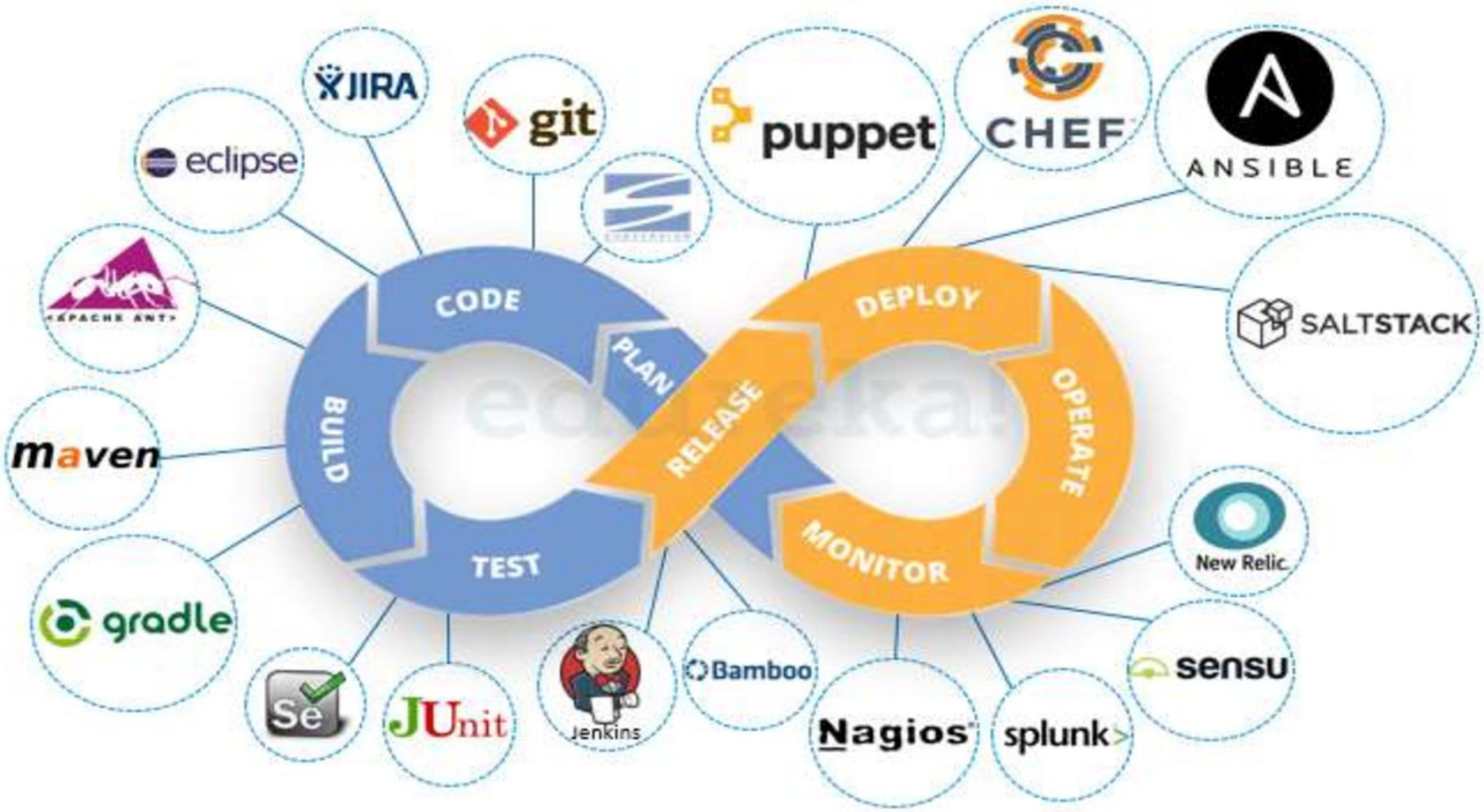
How is DevOps different from traditional IT?

Old Process	DevOps
After placing an order for new servers, the Development team works on testing. The Operations team works on extensive paperwork as required in enterprises to deploy the infrastructure.	After placing an order for new servers Development and Operations team work together on the paperwork to set-up the new servers. This results in better visibility of infrastructure requirement.
Projection about failover, redundancy, data center locations, and storage requirements are skewed as no inputs are available from developers who have deep knowledge of the application.	Projection about failover, redundancy, disaster recovery, data center locations, and storage requirements are pretty accurate due to the inputs from the developers.
Operations team has no clue on the progress of the Development team. Operations team develop a monitoring plan as per their understanding.	In DevOps, the Operations team is completely aware of the progress the developers are making. Operations team interact with developers and jointly develop a monitoring plan that caters to the IT and business needs. They also use advance Application Performance Monitoring (APM) Tools
Before go-live, the load testing crashes the application. The release is delayed.	Before go-live, the load testing makes the application a bit slow. The development team quickly fixes the bottlenecks. The application is released on time.

Why is DevOps used?

1. **Predictability:** DevOps offers significantly lower failure rate of new releases.
2. **Reproducibility:** Version everything so that earlier version can be restored.
3. **Maintainability:** Effortless process of recovery in the event of a new release crashing or disabling the current system.
4. **Time to market:** DevOps reduces the time to market up to 50% through streamlined software delivery. This is particularly the case for digital and mobile applications
5. **Greater Quality:** DevOps helps the team to provide improved quality of application development as it incorporates infrastructure issues.
6. **Reduced Risk:** DevOps incorporates security aspects in the software delivery lifecycle. It helps in reduction of defects across the lifecycle.
7. **Cost Efficiency:** DevOps offers cost efficiency in the software development process which is always an aspiration of IT companies' management.

DevOps Lifecycle



1. Continuous Development

Tools Used: Git, SVN, Mercurial, CVS

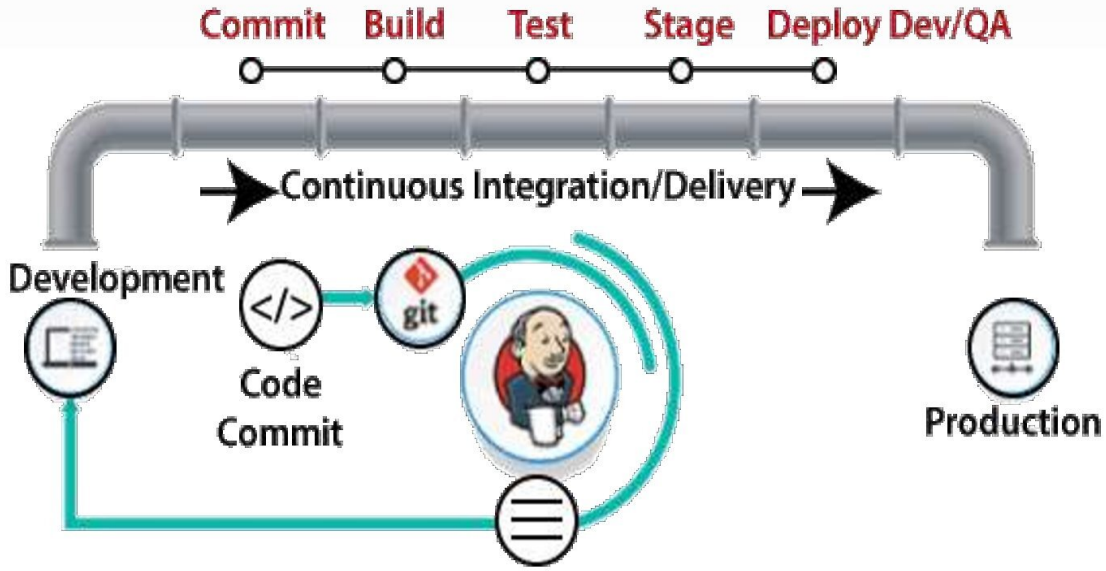
- This is the phase that involves **`planning `**and **`coding '**of the software.
- This is the phase where the team members sit down and **visualize the outcome** or, in other words, **how the software application will turn out to be**.
- Once the vision is in place, the developers start **writing and maintaining** the **code.**
- JavaScript, **C/C++, Ruby, and Python** are prominently used for coding applications in DevOps.
- No DevOps tools required for planning.
- Many version control tools are used to maintain code.

2. Continuous Integration

Tools: Jenkins, TeamCity, Travis

- Continuous Integration is a development practice where developers integrate code into a shared repository frequently where each integration is verified by an automated build and automated tests.
- This stage is the heart of the entire DevOps lifecycle.
- developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis.
- Then every commit is built, and this allows early detection of problems if they are present
- The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software.
- Jenkins is a popular tool used in this phase.
- Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an **executable file in the form of war or jar.**

2. Continuous Integration



3. Continuous Testing

Tools: Jenkins, Selenium TestNG, Junit

- This phase, where the developed software is continuously testing for bugs, defects and flaws.
- This is also the phase where the usability of the software is tested using the set of best practices
- Determine whether the software meets the specifications defined by the client.
- Continuous testing is carried out using automation testing tools, which can be open source tools like **Selenium** or advanced test management tools like **TestNG, JUnit, Selenium.**
- **Selenium** does the **automation testing**, and **TestNG** generates the **reports**. This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.
- Automation testing saves a lot of time and effort for executing the tests instead of doing this manually.
- Also, we can schedule the execution of the test cases at predefined times. **After testing, the code is continuously integrated with the existing code.**

4. Continuous Deployment

Tools Used: Configuration Management – Chef, Puppet, Ansible
Containerization – Docker, Vagrant

- In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.
- In this phase **Chef, Puppet and Ansible** are used as **Configuration Management tools**.
- Configuration Management is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
- **Containerization tools** are also playing an essential role in the deployment phase. **Vagrant** and **Docker** are popular tools that are used for this purpose.
- These tools help to produce consistency across development, staging, testing, and production environment.
- There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment.

5. Continuous Monitoring

Tools Used: Splunk, ELK Stack, Nagios, New Relic

- Continuous monitoring is an operational phase where the objective is to enhance the overall efficiency of the software application
- Different system errors such as **'server not reachable', 'low memory',** etc., are resolved
- **Network issues** and other problems are automatically fixed during this phase at the time of their detection.
- As a result, during continuous monitoring, **developers** can proactively **check the overall health of the system**. Proactive checking **improves the reliability and productivity** of the system and also reduces maintenance costs.
- Security issues get resolved and problems are automatically detected and fixed.
- Compared to the software development teams, the IT operations teams are more involved in this phase. Their role is pivotal in supervising user activity, checking the system for unusual behavior, and tracing the presence of bugs.