Introduction to DevOps

DevOps is a combination of Development and Operations.

Source Code Management (SCM)	→	Git, GitHub
Build	→	Apache Maven
Web Server	→	Tomcat, JBoss, WildFly
CI-CD	→	Jenkins (Admin Level)
Mini Project	→	Project to be deployed in Windows (10,11) environment.
Linux Project		
Containerization	→	Docker
K8s	→	Container Orchestration
Config Management	→	Ansible
Iaac	→	Terraform
Monitoring	→	Prometheus, Grafana

DRAWBACK OF AGILE METHODOLOGY:

1. Platform Dependency.
2. Environment Dependency.

Software Development Life Cycle (SDLC): It is a process to build any package, software or any application.

SDLC has two models: Waterfall model

Agile model

1. WATERFALL MODEL:

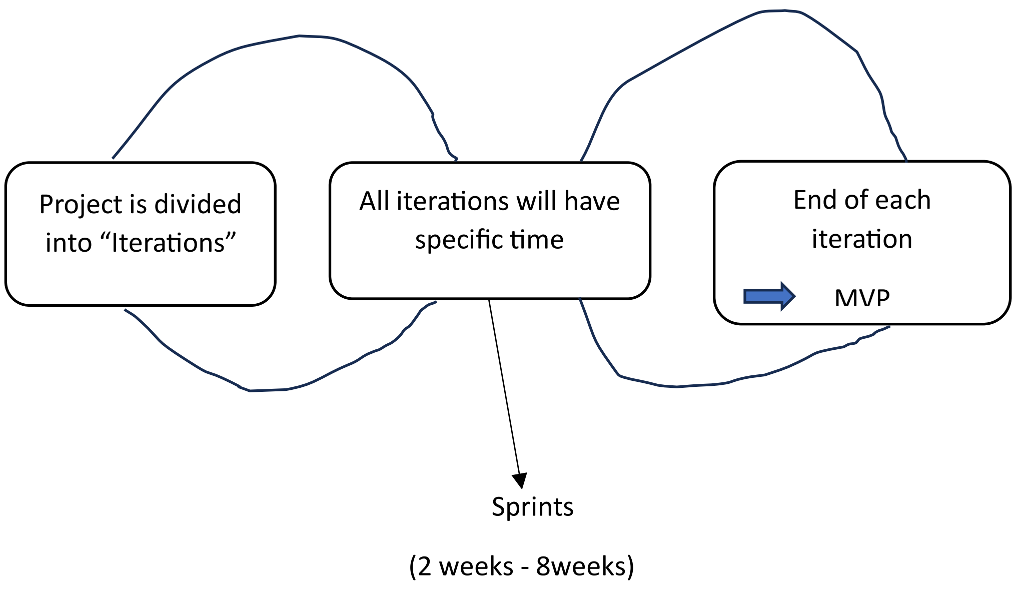
- Requirement/Gathering & Analysis
- Design/Planning/Architect
- Implementation/Coding/Development
- Testing
- Release/Delivery & Deployment
- Maintenance/Monitoring & Feedback

Drawbacks of Waterfall Model

1. It is not suitable for complex projects where changes are in high frequency.
2. Time consuming
3. Until and unless, one stage is not complete we can't go to the next stage and at the same time we can't go to back.

In order to address these issues/limitations, we have another model called “Agile Methodology”.

Agile Methodology:


Project is divided into “Iterations”
All iterations will have specific time
End of each iteration
MVP
Sprints
(2 weeks - 8weeks)

MVP: Minimum Viable Product

Limitations of Agile Methodology:

Dev Team

Wants changes

Ex: file 1.0 version

Config:

Java – 1.8 version

Tomcat – 9

Maven – 4

File 2.0 version

Ops Team

Wants stability

Server

To overcome the limitations of Agile methodology we go for DevOps method, which is a combination of Development and Operations.

<u>Development without DevOps culture:</u>

1. Release & deploy mismatches.
2. Unpredictable issues.
3. Blame games

<u>Development with DevOps culture:</u>

1. Streamline deliveries.
2. Teamwork in collaboration.
3. Continuous Monitoring & Feedback.

<u>DevOps Life Cycle:</u>

1. Plan
2. Code
3. Build
4. Test

Dev Team

1. Release
2. Deploy
3. Operations
4. Monitor

Ops Team

What is not the "DevOps"?

1. DevOps is not a role, person or an organization.
2. DevOps is not a separate team.
3. DevOps is not a product or tool.
4. DevOps is not about just writing the scripts or implementing the tools.

What is DevOps?

- **DevOps** is a practice that allows a single team to manage the entire development life cycle that is: Development, Testing, Deployment & Monitoring

$\boxed{\text{Code} \rightarrow \text{Production}}$

What does DevOps do?

1. Integrates developers and operations team.
2. Improves collaboration and productivity by:
    - Automating the infrastructure

- Automating the workflows
- Continuously measuring application performances

SKILLS OF A DEVOPS ENGINEER

Tools	**1. Version Control System – Git** **2. Continuous integration – Jenkins** **3. Containerization/Virtualization – Docker** **4. Configuration Management – Ansible** **5. Monitoring – Prometheus & Grafana**
Networking Skills	• General networking sills – Establishing connection between the containers, container orchestration
Other Skills	➢ People Skills ➢ Process Skills ➢ Customer skills and Empathy ➢ Cloud Awareness

DEVOPS LIFECYCLE

1. **Plan:** First stage of DevOps cycle where you can plan, track, visualize and summarize your project before working or starting it.

   Ex: JIRA, Trello

2. **Code:** Second stage of DevOps cycle where the developers can write their code.

   Ex: Git, GitHub, BitBucket, GitLab, AWS Codecommit, Azure Repo

3. **Build:** It is a pre-release version and is defined by build number, rather than by release number.

   Ex: Apache Maven, Apache Ant, Gradle, Jenkins

4. **Test:** Process of executing automated test as a part of the software delivery pipeline in order to obtain feedback on the business risks associated with a software release as rapidly as possible.

   Ex: JMeter, Selenium, Junit

5. **Release:** This phase helps to integrate code into a shared repository using which you can detect and locate errors quickly and easily.

   Ex: Bamboo, GitLab, Jenkins

6. **Deploy:** Manage and maintain development and deployment of software systems and servers in any computational (any cloud) environment.

   Ex: Ansible

7. **Operate:** This phase is to keep the system upgraded with the latest code.

   Ex: Ansible, Chef, Puppet

8. **Monitor:** It ensures that application is performing as desired and the environment is stable. It quickly determines when a service is unavailable and understanding the underlying causes/issues.

   Ex: Prometheus, Grafana, Nagios, Splunk, Sensu