**DevOps**

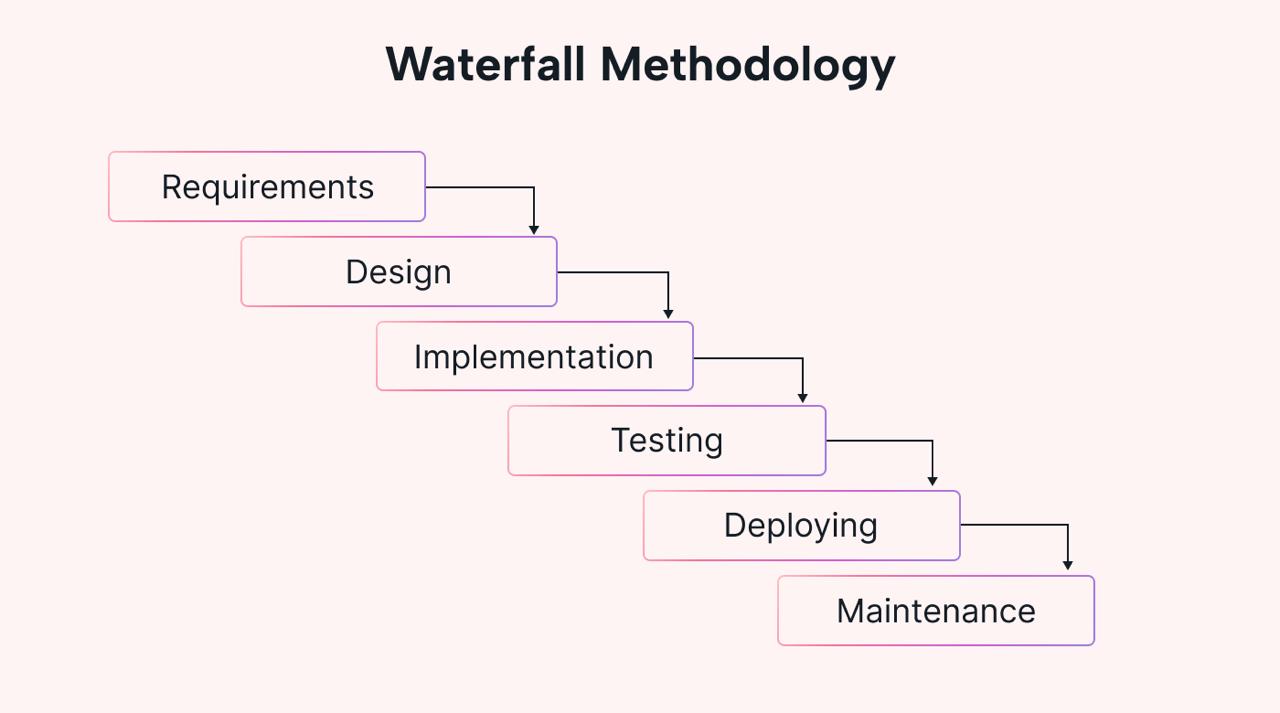
**Introduction to DevOps:-**

**SDLC :-** Software development Life Cycle.

* It is a process to build a package, software or applications.
* It has **2** models.
* Waterfall Model.
* Agile Model.

**WATERFALL MODEL:-**

Waterfall Model is a traditional and linear approach to software development , where progress is seen as flowing steadily downwards (like a waterfall) through distinct phases. It is one of the earliest models used in software engineering and follows a sequential design process**.**

****

**Limitations for Waterfall Model :-**

* IT is not suitable for complex projects . Where Changes are in high frequence.
* Time consuming.
* Until Unless Completion of the current stage . We cannot go for next stage at the same time we cannot go back.
* **In order to address these issues these (or) limitations we have next model is AGILE MODEL.**

**AGILE MODEL:-**

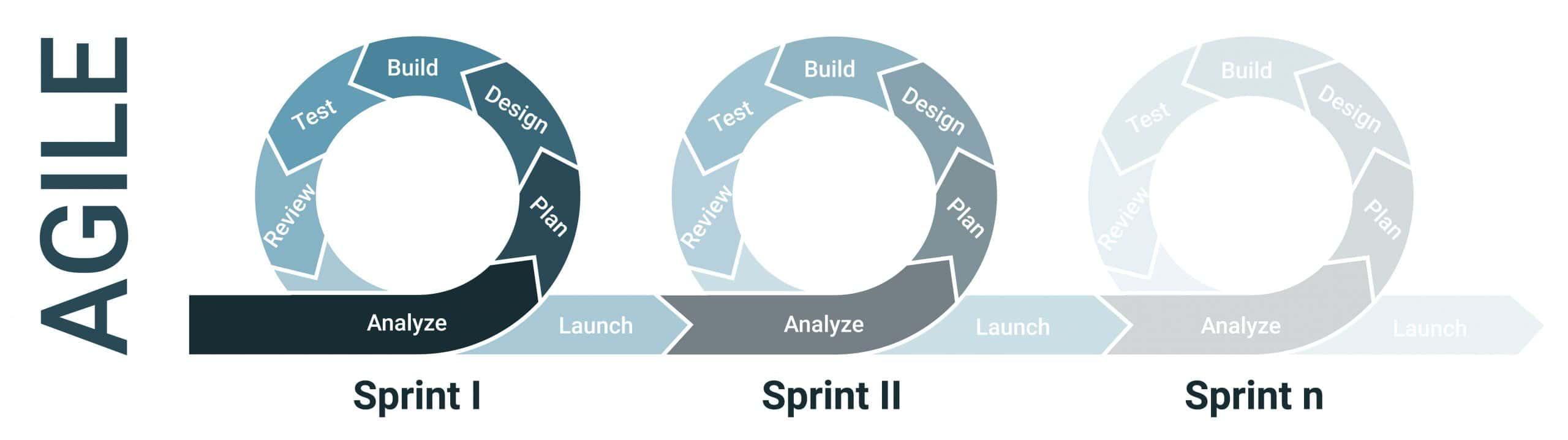
The Agile Model is an iterative and flexible approach to software development that emphasizes collaboration, customer feedback, and small, incremental improvements. Unlike the traditional Waterfall Model, which follows a linear and sequential process, the Agile Model focuses on delivering software in small, functional segments, with regular adjustments based on feedback and changing requirements.

**WHAT IS AGILE MODEL?**

In the Agile methodology each project is broken up into several ITERATION.

At the end of each iteration a working product should be delivered.

All the Iterations should be of the same time duration [Between 2 to 8 weeks].

****

**Limitations of AGILE MODEL: -**

DEVELOPMENT OPERATIONS

TEAM TEAM

**WANTS STABILITY**

**WANTS CHANGE**

**Example: -**

File – 1.0 version

Configuration

Java – 1.8 version

Tomcat – 9

Maven – 4

File – 2.0 version

To overcome these issues, we have a DevOps. Which is combination of Development team and operations team.

**Development without DevOps Culture: -**

* Release and deploy without mistakes.
* Unpredictable issues.
* Blame games.

**Development with DevOps culture: -**

* Stream line deliveries.
* Team works in Collaboration.
* Continuous Mentoring and Feedback.

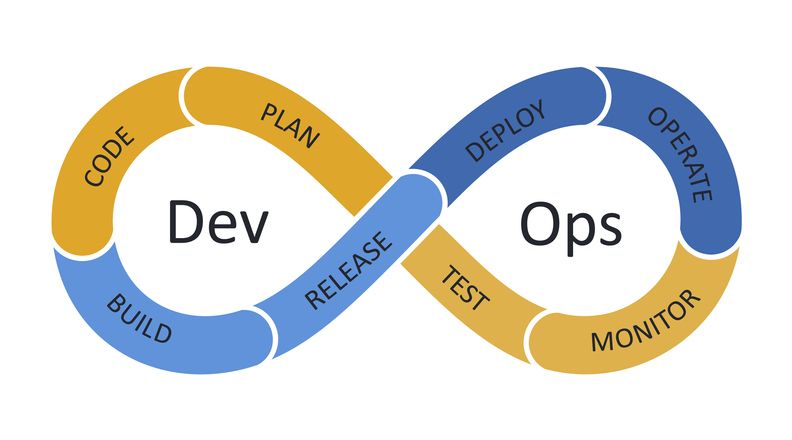
**Developers: -**

* Use System with little or no Waiting time.
* Use System with updated Code.

**Operations: -**

* Systems should be up and running most of the time.
* Systems Required for easy administrations.
* Effective Monitoring and Feedback systems should be Established.

**What Exactly is DevOps in Real Life?**

****

**Development Team: -**

* Plan
* Code
* Build
* Test

**Operations Team: -**

* Release
* Deploy
* Operations
* Monitor

**What DevOps is NOT!!**

* DevOps isn’t a Role, Person (or) Organization.
* DevOps isn’t a Separate team.
* DevOps isn’t a Product (or) tool.
* DevOps isn’t about just writing Scripts (or) implementing tools.

**What is DevOps?**

* DevOps is a Practice that allows a single team to manage the entire application development Lifecycle, that is Development, Testing , Deployment and Monitoring.

**CODE ------------------ > PRODUCTION**

**What Does DevOps Do?**

* Integrates developers and Operations teams.

Improves Collaboration and Productivity by: -

* Automating Infrastructure.
* Automating Workflows.
* Continuously measuring application Performance.

**SKILLS OF A DEVOPS ENGINEER: -**

* **TOOLS :-**
  + - Version control system  **GIT**
    - Continuous integration  **JENKINS**
    - Virtualization/configuration  **DOCKER**
    - Configuration management  **ANSIBLE**
    - Monitoring  **PROMRTHRUS AND GRAFANA**
* **Networking Skills :-**

General networking skills – Establishing connection between the containers, container orchestration.

* **Other skills :-**
  + - * People skills
      * Process skill
      * Customer skill and empathy
      * Cloud thing



**DEVOPS LIFECYCLE: -**

1. **Plan:**

“First stage of DevOps cycle’’ Where you plan, Track, visualise and summarized your project before working or starting it.

**Example: - JIRA, TRELLO, Tricenties**.

**2**.**Code**: -

Second stage of devops cycle, where the developers write the

code.

**Example: -Git, Git hub, Bit bucket, Git lab**

**3.Build: -**

Build is prerelease version and is defined by build number, rather than by release number.

**Example: - Apache maven**

**4.Test: -**

Process of executing automated test a part of the software delivery pipeline in order obtains 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 shade repository using

which you can detect and locate errors quickly and easily.

**Ex: - Bamboo, Gitlab, Travis CT**

**6.Deploy: -**

Manage and maintain development and deployment of software systems and servers in any computational (any cloud) Environment.

**Ex: -ansible, Aws, chef**

**7.Operations:** -

This phase is to keep the system upgraded with the latest update.

**Ex: - ansible, chef, Aws**

**8.Monitorizing:-**

It ensures that the application is performing as desired, and the environment is stable then it quickly determines when a service is unavailable and understands the underlying causes. (any issues)

**Ex: - prometheus, grafana, splunk, nagios, sensu**

* **VCS/SCM**
  + - * **VCS –** Version control system.
      * **SCM –** Source code management.
* **LVC –** Local version control system.
* **CVCS –** Centralized version control system.
* **DVCS –** Distribution version control system.

**WHY WE NEED VERSION CONTROL SYSTEM?**

* Collaboration
* Storing version
* Figuring out what happened
* Backup

**Issues without version control?**

* Once saved all the changes made in the files are permanent and cannot be reverted back.
* No record of what was done and by whom.
* Downtime that can occur because of a faulty update could cost million in losses.

**What is version control?**

**“**Version control is a system that documents changes made to a file or a set of files’’. It allows multiple users to manage multiple revisions of the same unit of information. It is a snap shot of your project over time.

**Version control types: -**

* **Local version control (LVC): -**
  + - The practice of having the version data base in the local computer.
    - Local data base keeps a record of the changes made to files in version data base.

**Local version control – Issue?**

**Issue: -** Multiple people parallelly working on the same

Project.

**Solutions: -** Centralized version control.

* **Centralized version control (CVC): -**
  + - Local version control’s issues are resolved by centralised version control.
    - In CVC, a central repository is maintained where all the versioned files are kept.
    - Now users can check out, and check in files from there different computers at any time.

**Centralized version control-issues?**

**\*Issue: -** In case of central sever failure whole system goes down.

**\*Solutions: -** Distributions version control.

**Distributed Version control:** -

\*Version Data base is stored at every user’s local system and at the

remote sever.

\*Users manipulate the local files and then upload the changes to the

remote users.

\*If any of the server die, a client server can be used to restore.