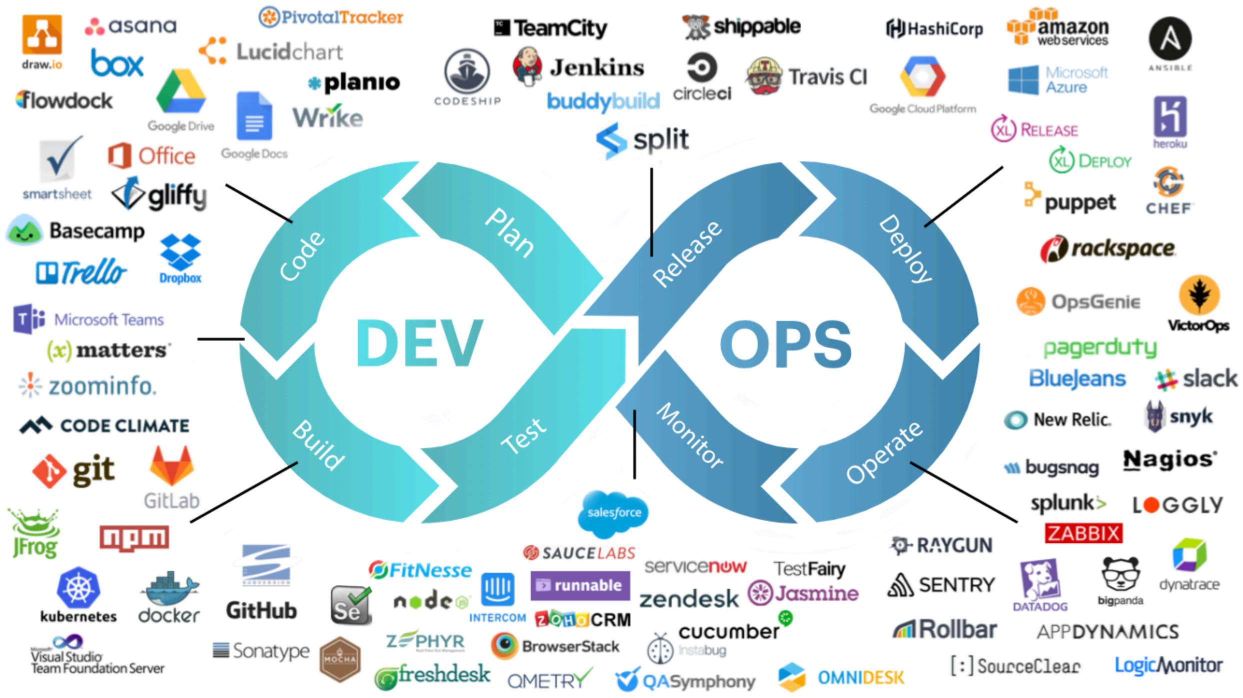
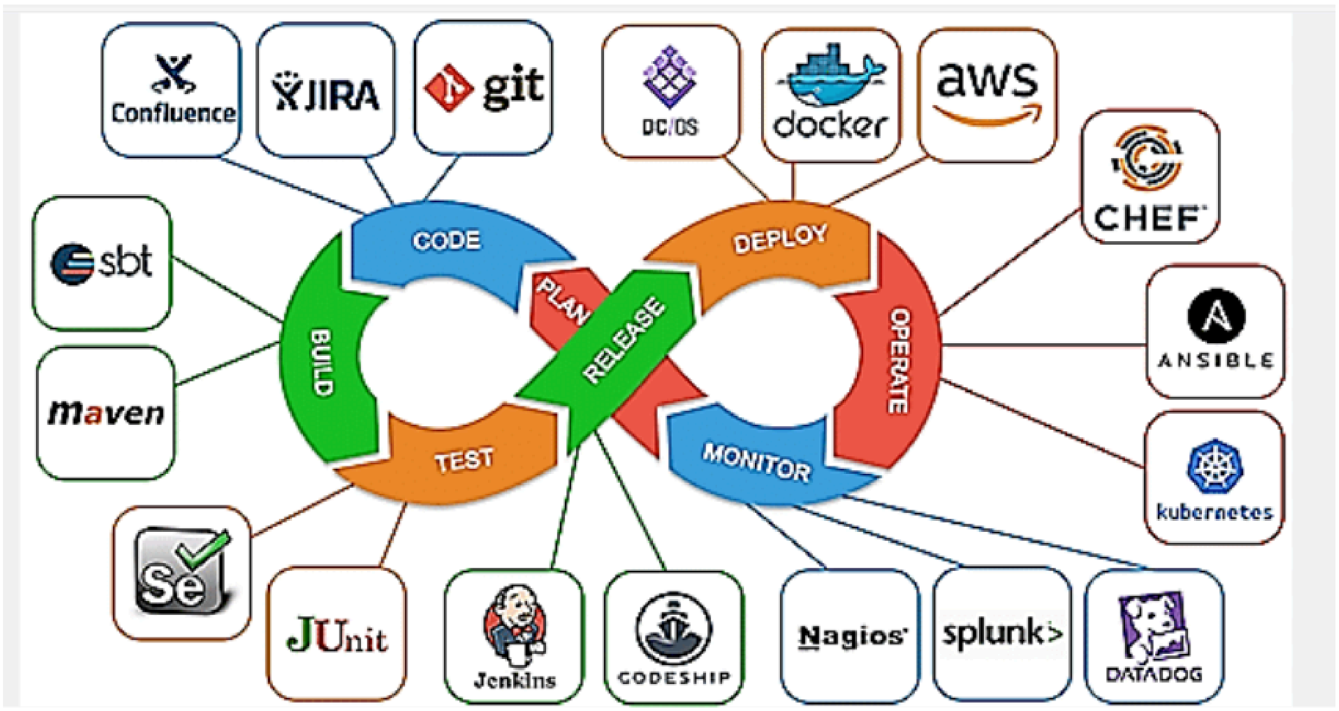
DEVOPS INTRO:

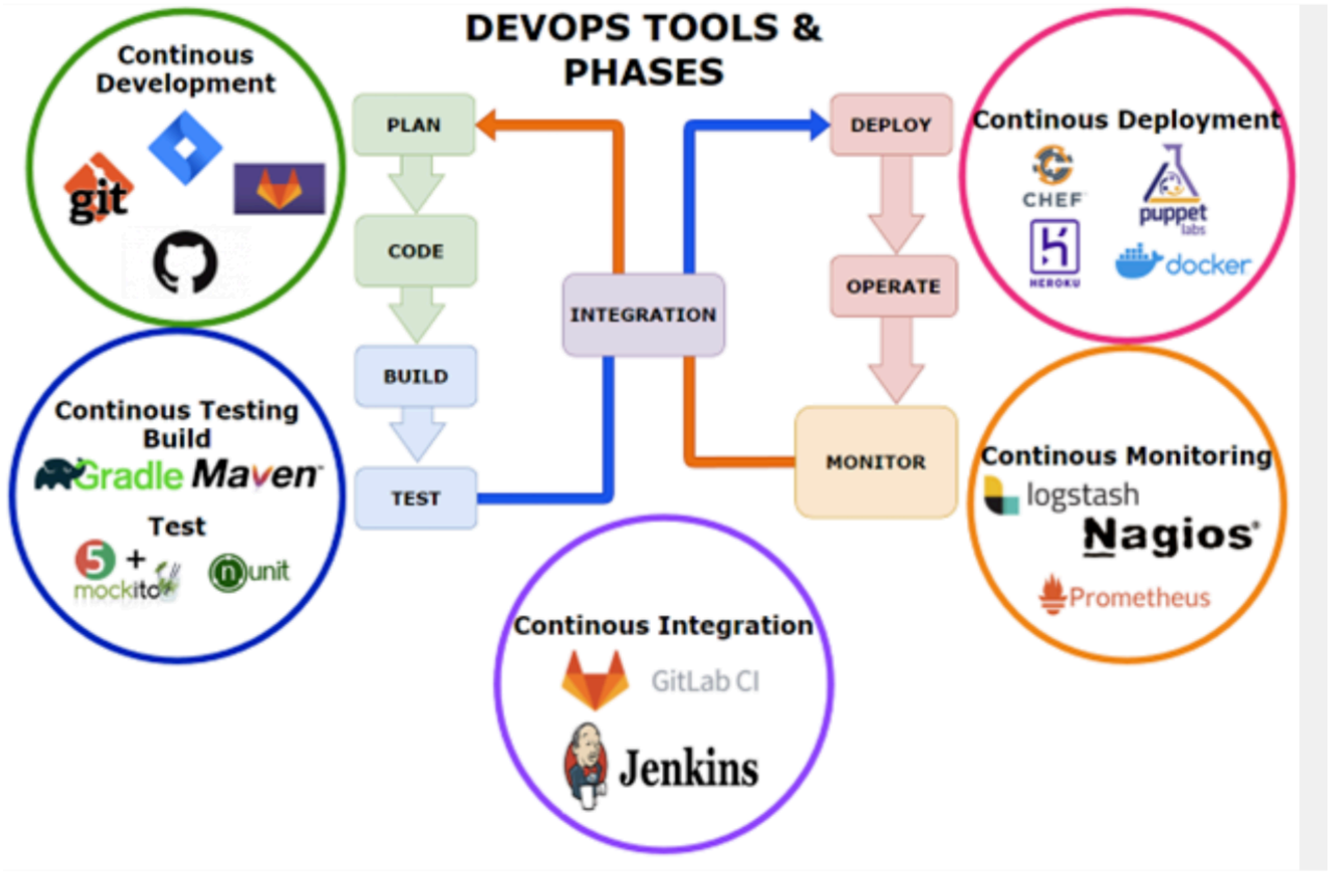
How is DevOps different from agile methodology?

A: DevOps is a culture that allows the development and the operations team to work together. This results in continuous development, testing, integration, deployment, and monitoring of the software throughout the lifecycle.

Agile is a software development methodology that focuses on iterative, incremental, small, and rapid releases of software, along with customer feedback. It addresses gaps and conflicts between the customer and developers.







Which are some of the most popular DevOps tools?
The most popular DevOps tools include:
1. Git/GItHub, Source code Management

2. Selenium
2. Puppet
3. Chef
4. Git
5. Jenkins
6. Ansible
7. Docker

What are the different phases in DevOps?
PC-BT-R-DOM
A: The various phases of the DevOps lifecycle are as follows:

Plan - Initially, there should be a plan for the type of application that needs to be developed. Getting a rough picture of the development process is always a good idea.
Code - The application is coded as per the end-user requirements.
Build - Build the application by integrating various codes formed in the previous steps.
Test - This is the most crucial step of the application development. Test the application and rebuild, if necessary.
Integrate - Multiple codes from different programmers are integrated into one.
Deploy - Code is deployed into a cloud environment for further usage. It is ensured that any new changes do not affect the functioning of a high traffic website.
Operate - Operations are performed on the code if required.
Monitor - Application performance is monitored. Changes are made to meet the end-user requirements.

Mention some of the core benefits of DevOps.
The core benefits of DevOps are as follows:

Technical benefits
- Continuous software deliver
- Less complex problems to manage
- Early detection and faster correction of defects

Business benefits
- Faster delivery of features
- Stable operating environments
- Improved communication and collaboration between the teams

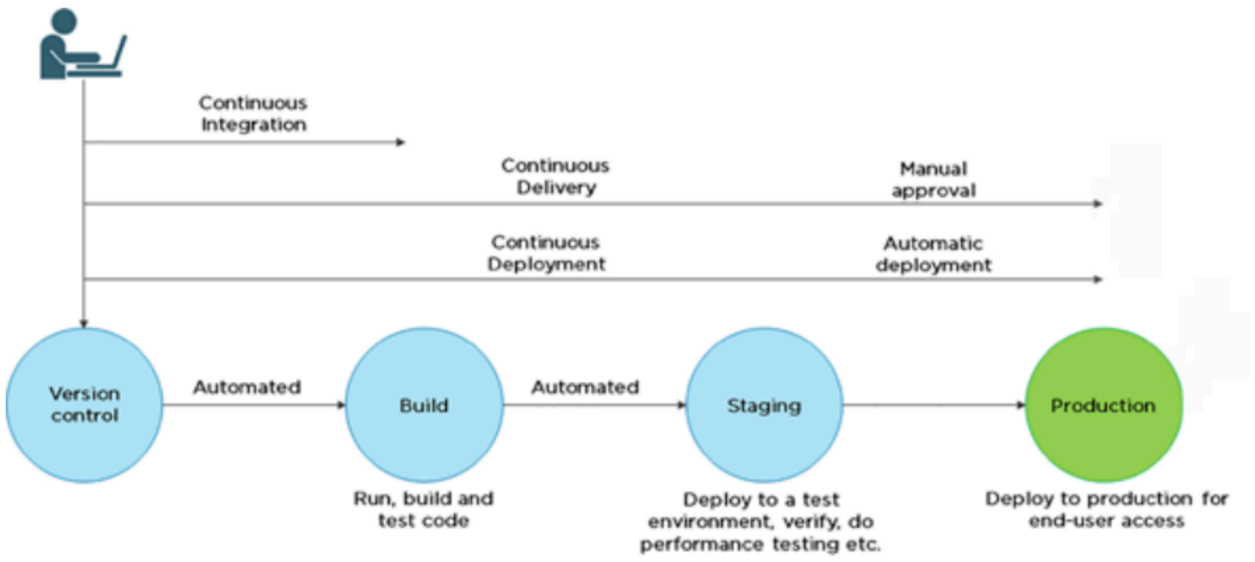
What is the difference between continuous delivery and continuous deployment?

Continuous Delivery

- Ensures code can be safely deployed
  on to production
- Ensures business applications and
  services function as expected
- Delivers every change to a production-like
  environment through rigorous automated testing

Continuous Deployment

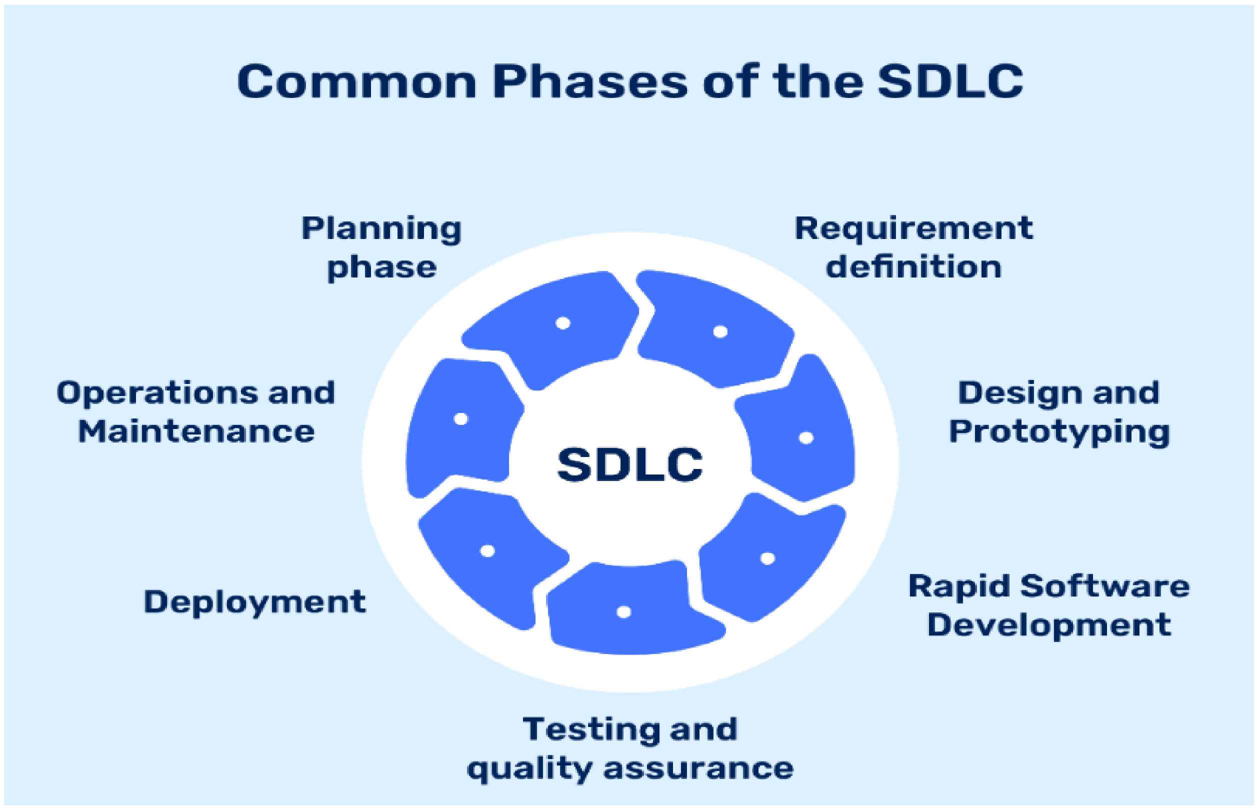
- Every change that passes the automated tests is deployed to production automatically
- Makes software development and the release process faster and more robust
- There is no explicit approval from a developer and requires a developed culture of monitoring



SDLC(Software Development Life-Cycle)

Entity1: Customer
Entity2: Service Provider(Development Team)

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time.

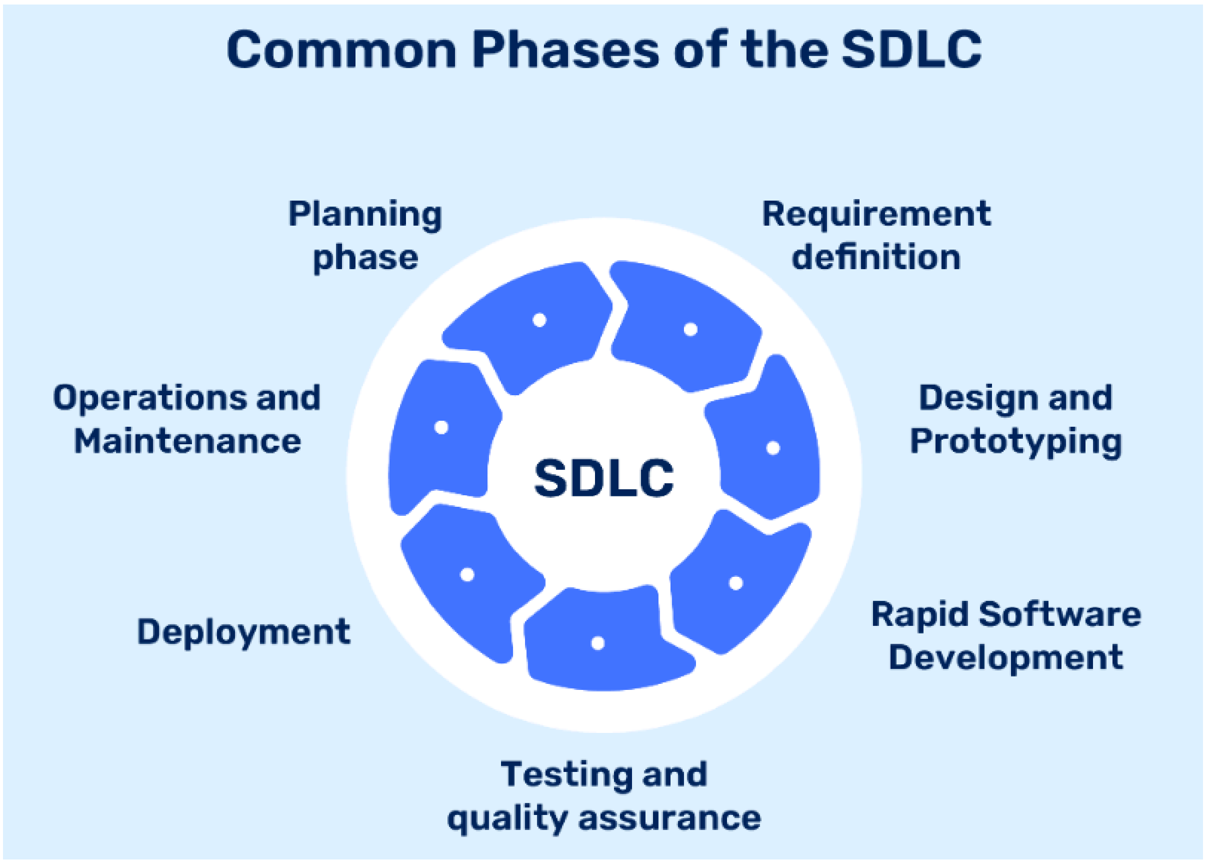

Common Phases of the SDLC
Planning phase
Requirement definition
Operations and Maintenance
SDLC
Design and Prototyping
Deployment
Rapid Software Development
Testing and quality assurance

The SDLC defines and outlines a detailed plan with stages, or phases, that each encompass their own process and deliverables. Adoption to the SDLC **enhances development speed and minimises project risks and costs associated with alternative methods of production.**

The primary advantages of pursuing a secure SDLC approach include

- More secure software as security is a *continuous* concern
- Awareness of security considerations by stakeholders
- Early detection of flaws in the system
- Cost reduction as a result of early detection and resolution of issues
- Overall reduction of intrinsic business risks for the organisation

The 7 Phases Of SDLC (Software Development Life Cycle)


Common Phases of the SDLC
Planning phase
Requirement definition
Design and Prototyping
Rapid Software Development
Testing and quality assurance
Deployment
Operations and Maintenance
SDLC

Scenario: We want to develop a new web-application for ecommerce: like: Signup, Login, Product Page, Add to Cart, Order, Payment Processing, Shipping, etc…

Stage 1: Project Planning

- The purpose of the application
- The details about the end-user of the Products
- Key elements like formats and attributes of the application
- Overall user interfaces design of the software

The first stage of SDLC is all about “What do we want?” Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software. Everything regarding cost, initial financial analysis, time, initail approach,

Stage 2: Gathering Requirements & Analysis

- Information about each element to design
- Validating the installation
- Calibrating the security protocols and risk analysis
- SRS (Software Requirement Specification Document)

The second step of SDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer. The development team will then analyze the requirements keeping the design and code of the software in mind. Further, investigating the validity and possibility of incorporating these requirements into the software system. The main goal of this stage is that everyone understands even the minute detail of the requirement. Hardware, operating systems, programming, and security are to name the few requirements. SRS- Software requirements specification

Stage 3: Designing and Prototyping

- Design the system
- Check feasibility with requirements
- **Design Document Specification (DDS Document)** - Software Architecture-shared with stakeholder

In the design phase (3rd step of SDLC), the program developer scrutinizes whether the prepared software suffices all the requirements of the end-user. Additionally, if the project is feasible for the customer **technologically, practically, and financially**. Once the developer decides on the best design approach, he then selects the program languages like Oracle, Java, etc., that will suit the software.

Once the design specification is prepared, all the stakeholders will review this plan and provide their feedback and suggestions. It is absolutely mandatory to collect and incorporate stakeholder's input in the document, as a small mistake can lead to cost overrun.

UML Unified Modelling Language, Flow-charts, E-R diagram in DBMS.

Stage 4: Coding or Implementation

- Developers start writing the code using the Languages
- Actual Implementation of the Software Product

- Development tools

Time to code! It means translating the design to a computer-legible language. In this fourth stage of SDLC, the tasks are divided into modules or units and assigned to various developers. The developers will then start building the entire system by writing code using the programming languages they chose. This stage is considered to be one of the longest in SDLC. The developers need certain predefined coding guidelines, and programming tools like interpreters, compilers, debugger to implement the code.

The developers can show the work done to the business analysts in case if any modifications or enhancements required.

Stage 5: Testing

Once the developers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements.

After testing, the QA and Testing team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system. Unit Testing, Integration Testing,

Stage 6: Deployment

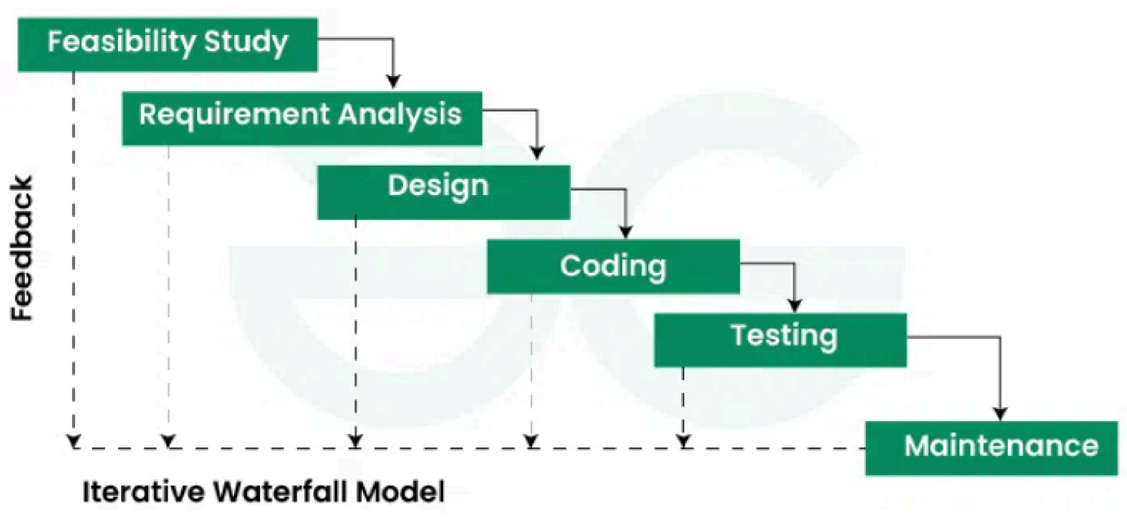
The sixth phase of SDLC: Once the testing is done, and the product is ready for Deployment., it is released for customers to use. The size of the project determines the complexity of the deployment. The users are then provided with the training or documentation that will help them to operate the software. Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

Stage 7: Maintenance

The actual problem starts when the customer actually starts using the developed system and those needs to be solved from time to time. Maintenance is the seventh phase of SDLC where the developed product is taken care of. According to the changing user end environment or technology, the software is updated timely.

Predominant Models of SDLC

Waterfall Model and Iterative Waterfall Model (feedback added):



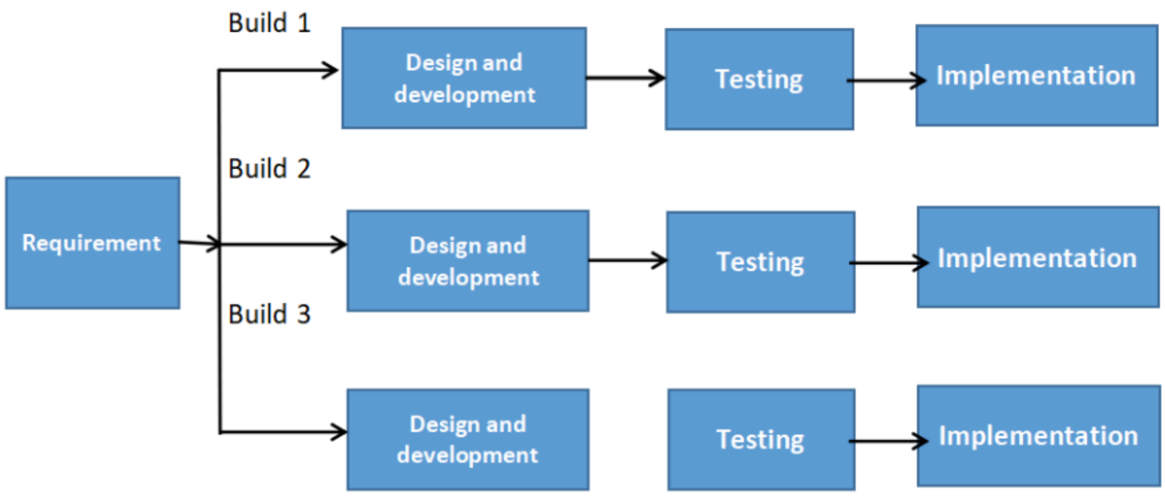
Phases of Iterative Waterfall Model

This SDLC model is considered to be the **oldest, simple, classical model** and most forthright (clear vision).

It is also called as a **linear sequential model.**

We finish with one phase and then start with the next, with the help of this methodology. Why the name waterfall? Because each of the phases in this model has its own mini-plan and each stage waterfalls into the next. A drawback that holds back this model is that even the small details left incomplete can hold an entire process.

Iterative Model:



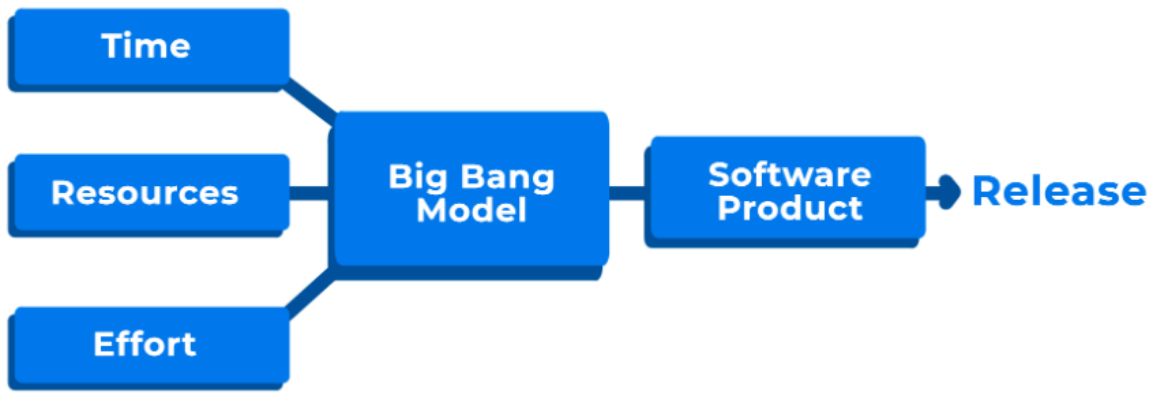
This SDLC model stresses on repetition. Developers create a **version rapidly** for relatively less cost, then test and improve it through successive versions. One big disadvantage of this model is that if left unchecked, it can eat up resources fast.

V-Shaped Model:

It is also known as a **Verification and Validation model.**

This model can be considered as an extension of the waterfall model, as it includes **tests at each stage of development**. Just like the case with waterfall, this process can run into obstructions. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a **highly-disciplined model** and the **next phase starts only after completion of the previous phase.** Validation Testing includes..**(Unit Testing, Integration Testing, System and UAT)**

Big Bang Model:



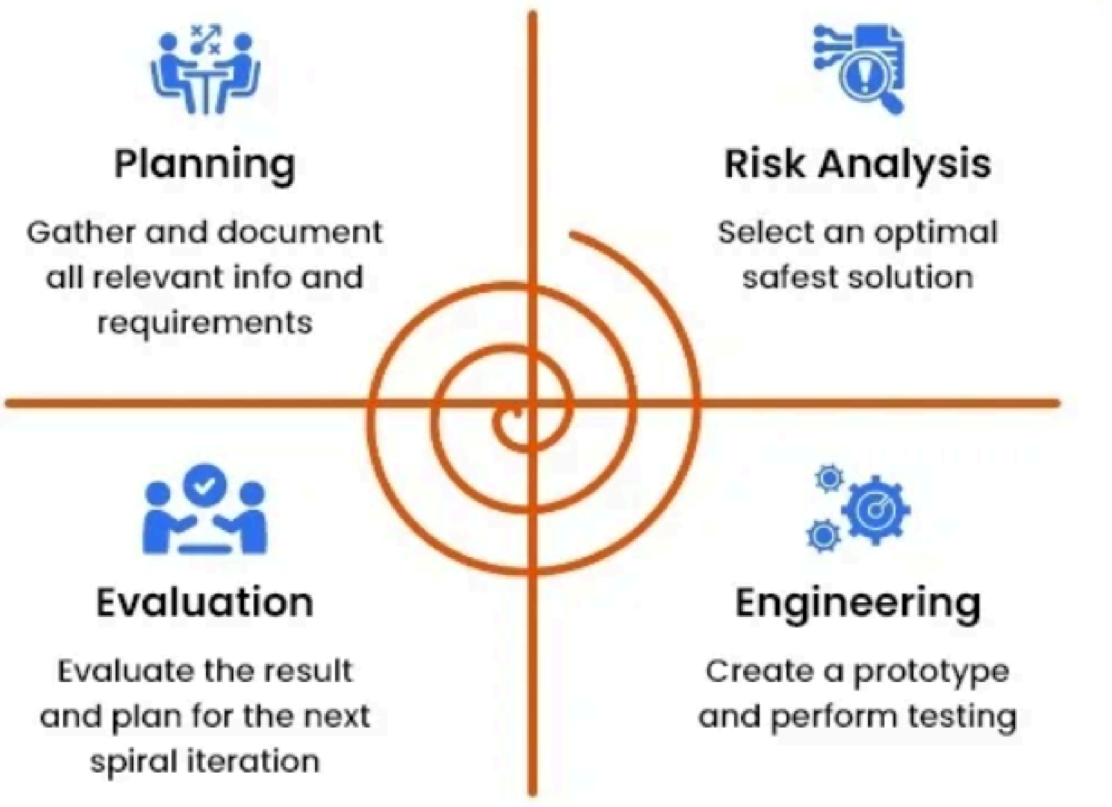
This SDLC model is considered best for small projects, **experimental projects** as it throws most of its resources at development. It **lacks the detailed requirements** definition stage when compared to the other methods.

This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the **customer is not sure** about what exactly he wants and the requirements are implemented **on the fly without much analysis.**

The major disadvantages of the Big Bang Model are:

- Very **High risk and uncertainty**.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Can turn out to be very expensive if requirements are misunderstood.

Spiral Model:

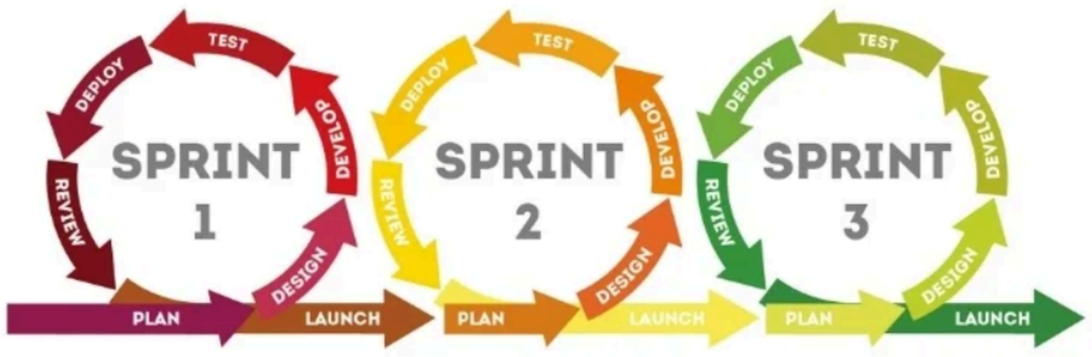

Planning
Gather and document all relevant info and requirements
Risk Analysis
Select an optimal safest solution
Evaluation
Evaluate the result and plan for the next spiral iteration
Engineering
Create a prototype and perform testing

One of the **most flexible of the SDLC models** is the spiral model. It resembles the iterative model in its emphasis on repetition. Even this model goes through the planning, design, build and test phases again and again, **with gradual improvements at each stage.**

The following pointers explain the typical uses of a Spiral Model –

- When there is a **budget constraint and risk evaluation** is important.
- For **medium to high-risk projects.**
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customers are **not sure of their requirements** which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Agile Model:

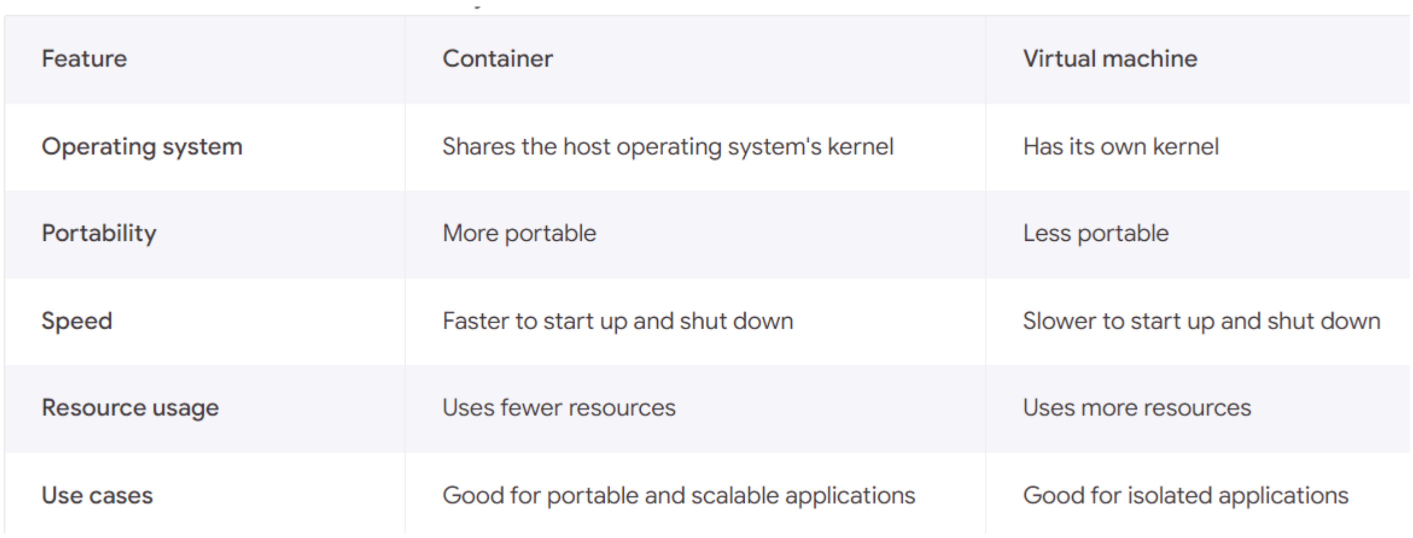


Agile means to move quickly and easily. It is one of the most utilised models, as it approaches software development in **incremental but rapid cycles**, commonly referred to as **“sprints”**. With new changes in scope and direction being implemented in each sprint, the project can be completed quickly with higher flexibility. Agile means spending less time in the planning phases, and a project can diverge from original specifications.

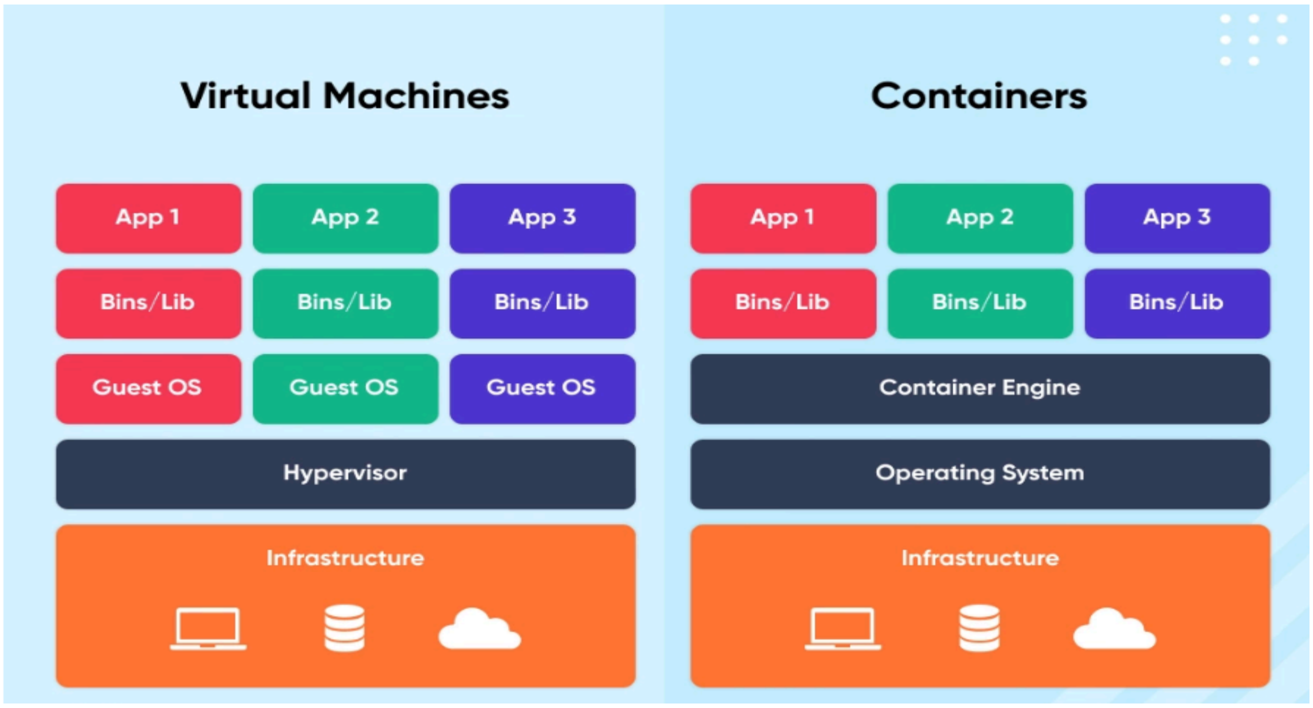
What is Virtualization?

Virtualization is achieved using a hypervisor, which splits CPU, RAM, and storage resources between multiple virtual machines (VMs). Each user on the hypervisor gets their own operating system environment.

It should be noted that none of the individual VMs interact with one another, but they all benefit from the same hardware. This means cloud platforms like AWS can maximize resource utilization per server with multiple tenants, enabling lower prices for enterprises through economies of scale



Feature	Container	Virtual machine
Operating system	Shares the host operating system's kernel	Has its own kernel
Portability	More portable	Less portable
Speed	Faster to start up and shut down	Slower to start up and shut down
Resource usage	Uses fewer resources	Uses more resources
Use cases	Good for portable and scalable applications	Good for isolated applications


Virtual Machines
Containers
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure