Unit-1

Introduction:

DevOps is a combination of software development (dev) and IT operations (ops) teams.

It is defined as a software engineering methodology which aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibility.

The development team is responsible for developing, designing, and building the application.

The operations team deals with the testing and deployment of the application. If there are problems with the application, the operation team also provides feedback to the development team.

The reason for combining these two teams is that, when the development team continuously push the changes to the code at random time, sometimes the operations team may not be available to validate the changes, due to which the whole project delays.

Hence, by collaboration of these two teams, the delivery of the product will accelerate by the implementation of various strategies such as automation, fast feedback system and iterative improvement.

Features of DevOps:

1. **Automation of the software development lifecycle.** This includes automating testing, builds, releases, the provisioning of development environments, and other manual tasks that can slow down or introduce human error into the software delivery process.
2. **Collaboration and communication.** A good DevOps team has automation, but a great DevOps team also has effective collaboration and communication.
3. **Continuous improvement and minimization of waste.** From automating repetitive tasks to watching performance metrics for ways

to reduce release times or mean-time-to-recovery, high performing DevOps teams are regularly looking for areas that could be improved.

4. **Hyperfocus on user needs with short feedback loops.** Through automation, improved communication and collaboration, and continuous improvement, DevOps teams can take a moment and focus on what real users really want, and how to give it to them.

Life Cycle of DevOps: (Architecture of DevOps)

- **Plan:** Determining the commercial needs and gathering the opinions of end-user by professionals in this level of the DevOps lifecycle.
- **Code:** At this level, the code for the same is developed and in order to simplify the design, the team of developers uses tools and extensions that take care of security problems.
- **Build:** After the coding part, programmers use various tools for the submission of the code to the common code source.
- **Test:** This level is very important to assure software integrity. Various sorts of tests are done such as user acceptability testing, safety testing, speed testing, and many more.
- **Release:** At this level, everything is ready to be deployed in the operational environment.
- **Deploy:** In this level, Infrastructure-as-Code assists in creating the operational infrastructure and subsequently publishes the build using various DevOps lifecycle tools.
- **Operate:** At this level, the available version is ready for users to use. Here, the department looks after the server configuration and deployment.
- **Monitor:** The observation is done at this level that depends on the data which is gathered from consumer behavior, the efficiency of applications, and from various other sources.

Advantages of DevOps**:**

1. **Faster Delivery**: DevOps enables organizations to release new products and updates faster and more frequently, which can lead to a competitive advantage.

2. **Improved Collaboration**: DevOps promotes collaboration between development and operations teams, resulting in better communication, increased efficiency, and reduced friction.
3. **Improved Quality**: DevOps emphasizes automated testing and continuous integration, which helps to catch bugs early in the development process and improve the overall quality of software.
4. **Increased Automation**: DevOps enables organizations to automate many manual processes, freeing up time for more strategic work and reducing the risk of human error.
5. **Better Scalability**: DevOps enables organizations to quickly and efficiently scale their infrastructure to meet changing demands, improving the ability to respond to business needs.
6. **Increased Customer Satisfaction**: DevOps helps organizations to deliver new features and updates more quickly, which can result in increased customer satisfaction and loyalty.
7. **Improved Security**: DevOps promotes security best practices, such as continuous testing and monitoring, which can help to reduce the risk of security breaches and improve the overall security of an organization's systems.
8. **Better Resource Utilization**: DevOps enables organizations to optimize their use of resources, including hardware, software, and personnel, which can result in cost savings and improved efficiency.

Agile Development Model: [agile => "move quickly and easily"]

Agile Model is a combination of iterative and Incremental process models.

Agile Model divides the features and functionalities into parts known as increments. Each increment is delivered to the customer when completed, for feedback.

The output of each iteration of Agile Model is a fully functioning product with some flaws, incomplete features, and errors. These flaws and errors are identified and fixed in the next Iteration with some more added features.

Each iteration is intended to be small and easily manageable and can be completed within a couple of weeks only.

At a time one iteration is planned, developed, and deployed to the customers. Long-term plans are not made.

Hence, the main aim of Agile Model is to build the product quickly and easily and provide regular upgrades to the product over time (with customer feedback, etc).

Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project. Also, anything that is a waste of time and effort is avoided.

Features of Agile Model:

- Iterative Development: Work is divided into small iterations or sprints, each of which results in a potentially shippable product increment. This iterative approach allows for frequent inspection and adaptation.
- Customer Collaboration: Agile teams engage with customers and stakeholders to gather feedback, prioritize features, and ensure the product meets their needs.
- Cross-Functional Teams: Agile teams are typically cross-functional, including members with diverse skills such as developers, testers, designers, and product owners. This promotes collaboration and reduces handoffs.
- Continuous Integration: Code changes are integrated and tested frequently to maintain a working and reliable code base.
- Daily Stand-up Meetings: Teams hold daily stand-up meetings to discuss progress, identify obstacles, and plan the day's work.
- User Stories: Requirements are often expressed as user stories, which are short descriptions of functionality from the user's perspective.
- Test-Driven Development (TDD): Developers write tests before writing code, ensuring that the code functions correctly and is maintainable.
- Product Backlog: A prioritized list of features, user stories, and tasks that guide development efforts.
- Less Documentation: The detailed part of the documentation is avoided at the preset to be able to build the product first.

Phases of Agile Model

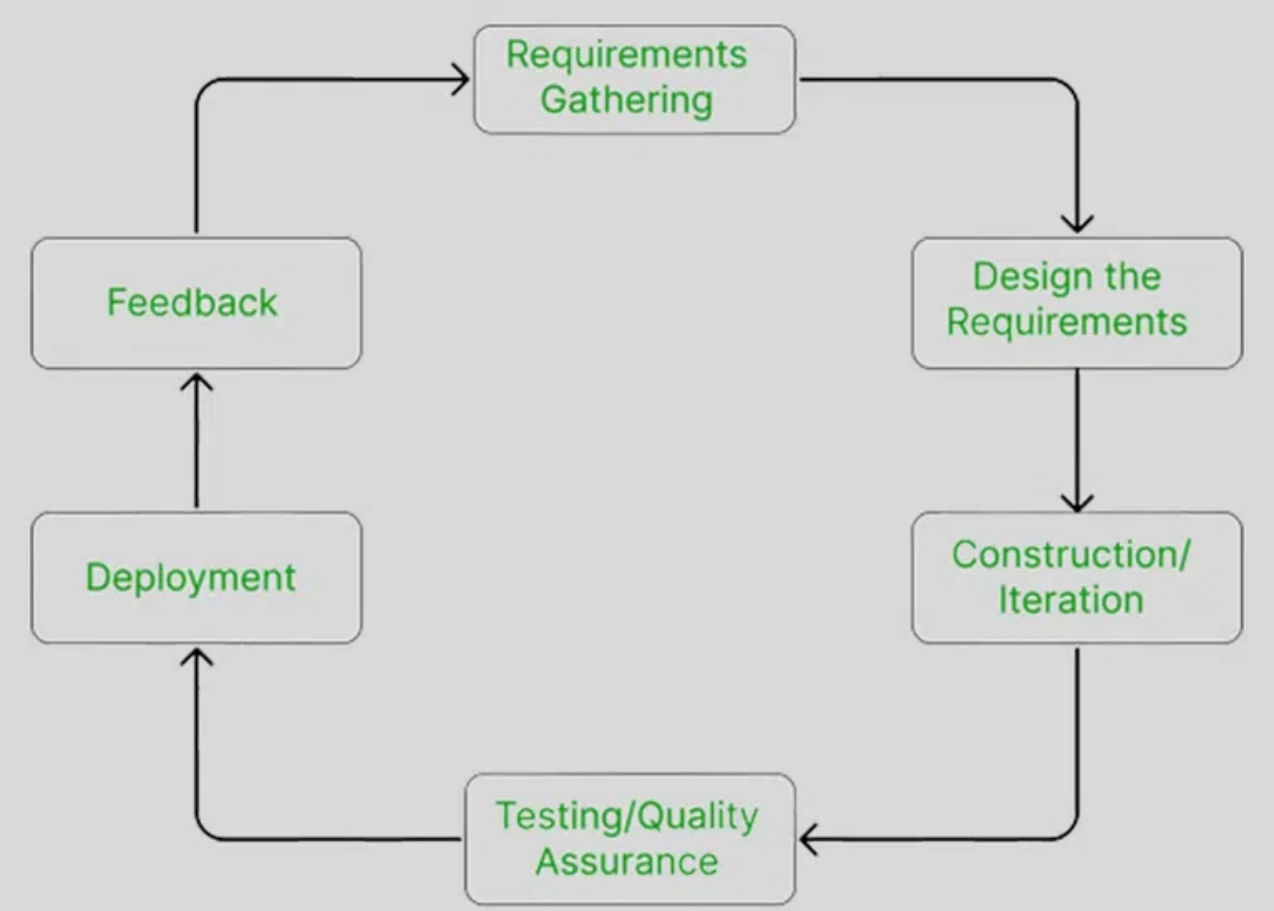


Fig:- Steps in Agile SDLC Model

1. Requirement Gathering:- In this phase, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.

2. Design the Requirements:- In this phase, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.

3. Construction / Iteration:- In this phase, development team members start working on their project, which aims to deploy a working product.

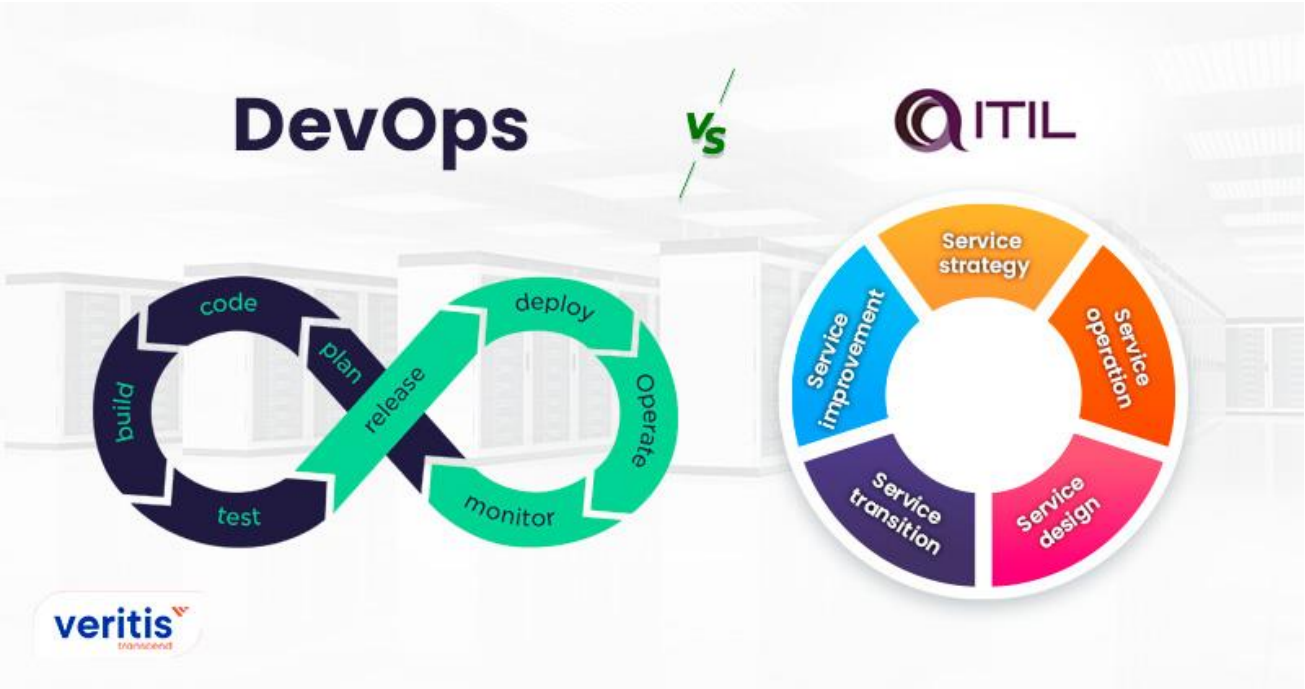
4. Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:

- **Unit Testing:-** Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.
- **Integration Testing:-** Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
- **System Testing:-** Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. Deployment:- In this phase, the development team will deploy the working project to end users.

6. Feedback:- This is the last phase of the **Agile Model.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

DevOps VS ITIL:



Basis	DevOps	ITIL

Definition	DevOps aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibility to accelerate the delivery process through automation, collaboration, fast feedback, and iterative improvement.	ITIL refers to a set of detailed guidelines for effective and efficient management of an organization's IT services to accelerate delivery process.
Focus & Scope	DevOps focuses on collaboration and communication between development (Dev) and operations (Ops) teams and implement Agility in the development process.	ITIL focuses on managing IT services and preparing the processes, documentation, and a structured approach to service delivery process.
Culture:	DevOps has a more collaborative and agile culture. It seeks to break down barriers between development and operations teams, fostering a culture of	ITIL has a more traditional and process-driven culture. It places importance on following established procedures, documentation, and roles within the IT

	shared responsibility and rapid iteration.	service management structure.
Speed vs. Stability	DevOps focuses is on delivering value to customers quickly, even if it means accepting a certain level of risk	ITIL often prioritizes stability and reliability of IT services. Changes are carefully planned and implemented to minimize the risk of disruptions.
Tools	DevOps places a strong emphasis on automation, and there is a wide range of tools specifically designed for various stages of the development and operations lifecycle, such as Jenkins, Docker, Kubernetes, etc.	ITIL doesn't prescribe specific tools but provides guidance on processes. Organizations implementing ITIL may use a variety of tools for service management, incident tracking, and other related activities.

DevOps Process & Continuous Delivery:

DevOps (Development and Operations) is a set of practices that aim to automate and improve the collaboration between software development and IT operations teams.

The goal of DevOps is to shorten the software development lifecycle, deliver high-quality software continuously, and enhance the overall efficiency of the development and operations processes.

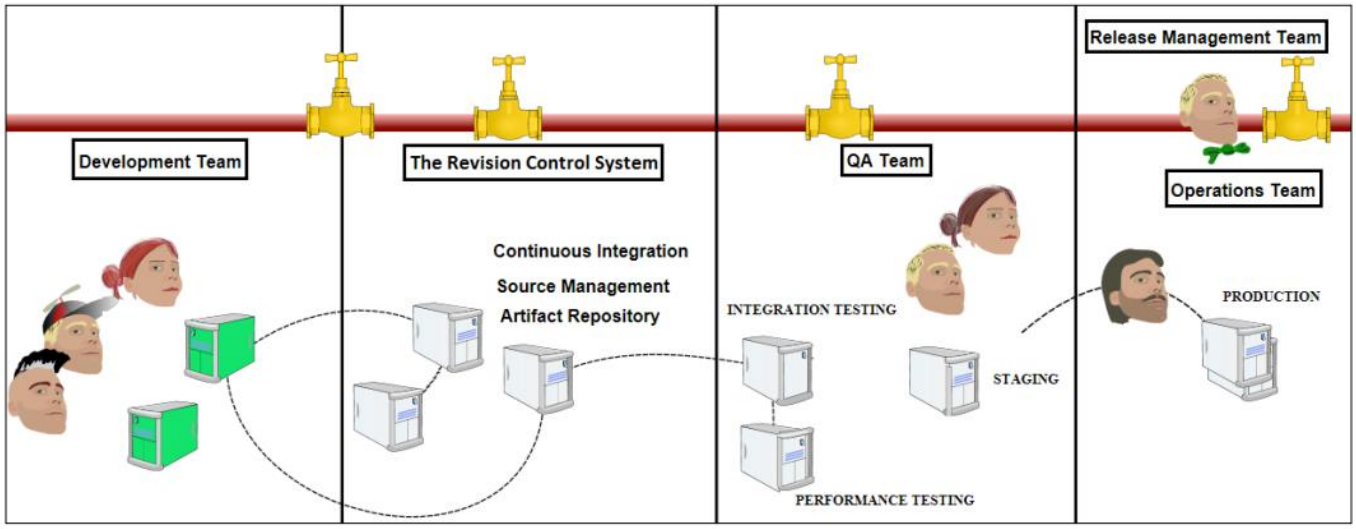
Continuous Delivery (CD) is a key concept within the DevOps methodology which refers to the practice of continuously delivering software updates to production, making the release process more efficient, reliable, and faster.

Continuous Delivery aims to automate the entire software release process, from code development to testing to deployment.

Continuous Delivery (CD) Pipeline:

Continuous Delivery (CD) pipeline is a set of automated processes and tools that facilitate the continuous delivery of software updates from development to production.

The goal of a CD pipeline is to automate the building, testing, and deployment of applications, ensuring that code changes are delivered to users in a consistent, reliable, and efficient manner.



The CD process is divided into (5) parts and assigned to separate teams, viz.:

(1) Development Team
(2) Revision Control System
(3) QA Team
(4) Operations Team
(5) Release Management Team

(1) Development Team:

This is a team of developers who write the code to build the software and provide features and functionalities.

Developers work on their respective local workstations and may use many packages, dependencies and tools.

To run the code in other workstations for testing, etc., the same packages, dependencies and tools must be installed, which is a tedious process.

Hence, to make the other teams easily run the code in their environments without having to install the dependencies and tools, the developers build a virtual machine.

A virtual machine is a digital copy of a physical machine.

One of the software that is used to build a virtual machine is “Docker”, which stores the packages and dependencies in containers.

(2) Revision Control System:

This System refers to the local repository where the committed changes are stored and managed.

Later it was Transcribed as Version Control System (Global Information Tracker (GIT)).

This System has (3) Responsibilities, viz.:

- Storage Management
- Continuous Integration
- Artifact Repository

Storage Management refers to storing the directory of the code base (files and folders) into the local repository and managing the changes made to the code and other files.

Continuous Integration is an automated process of listening to the local repository to build, test and merge the latest changes and updates into the central production repository, known as a Build Server. “Jenkins” is one of the software that offers the Build Server for CI and CD.

Artifact Repository is a local/central repository where the versioned artifacts (source files, packages, libraries, environment specs, & tools) are stored.

(3) QA (Quality Assurance) Team:

QA Team is responsible for Analyzing the quality of the build.

The quality of a build is analyzed by performing various types of tests after integrating the build changes into the original build, such as: Integration Testing, Performance Testing, etc.

After testing, the files are committed into the Staging Area.

Staging Area is a location where the only files that are changed/updated are present.

(4) Operations Team:

The production codebase management members in the main branch are called the members of the operational team.

They are the administrative members who track and manage the main/central repository.

The committed code in the Staging Area is tested and validated (Validation Testing) by the Operational team and pushed into the main branch of the production codebase.

(5) Release Management Team:

This team is responsible for deploying and publishing the latest release to allow users to start using the new version of the platform.

This work is mostly automated by using some Auto-Deployment tools.

In case of any new issues with the latest release, the Operations team can roll back the previous version of the platform at any time.

Agile Methodologies: [the frameworks used for faster development processes]

Agile (move quickly) Methodologies are those methods/frameworks built with Agile Development Model that can be implemented in the product development and management lifecycle for faster and efficient processes.

Agile Methodologies prioritize flexibility, collaboration, and customer satisfaction.

The (2) most used agile methodologies are:

(1) Scrum
(2) Kanban

An organization usually uses both frameworks.

(1) Scrum:

An Iterative Agile Framework which aims to collaborate the development teams and implementing sprints to expect the quick and qualitative product at the end.

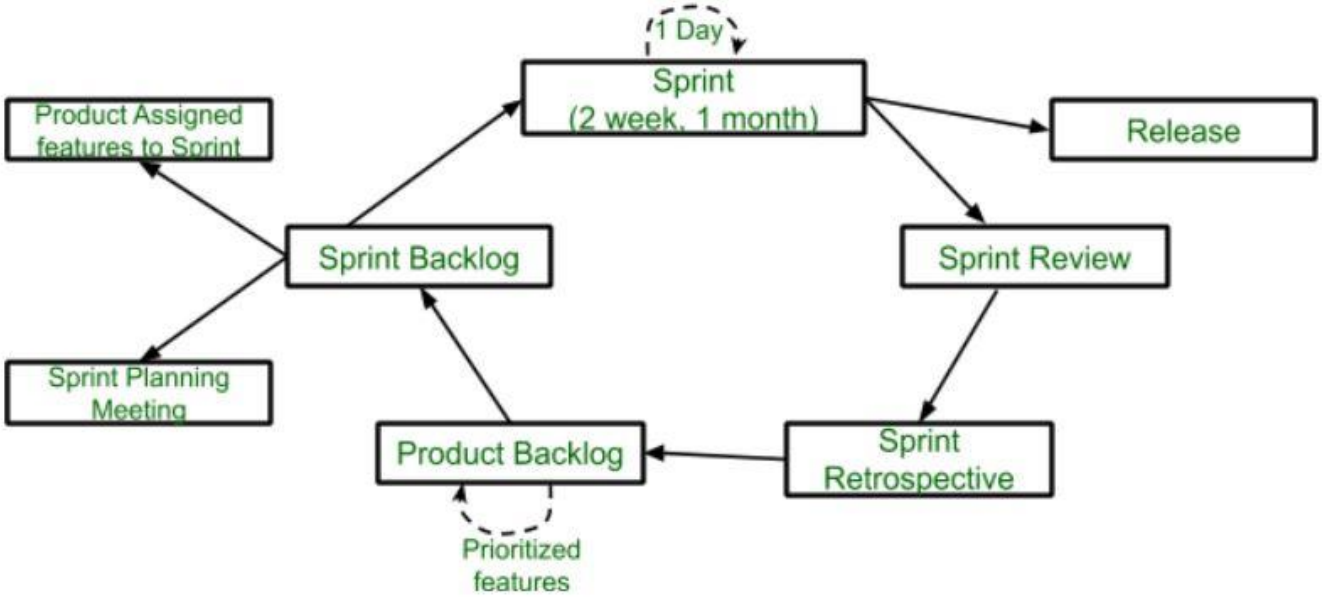
This framework focuses on reducing complexity of development process by diving it into assignable tasks (increments).

The large tasks such as major features and functionalities of product, are assigned to the developers, in these sprints.

Roles in Scrum: (entities participating in scrum)

(a) Product Owner: Stakeholders of the project who define and prioritize the product backlog and ensure that team delivers value aligned with business goals.
(b) Scrum Master: Facilitates the Scrum process (creating and assigning stories) and removes obstacles, Manages and helps the team stick to the Scrum practices.
(c) Development team: Developers/coders who are supposed to complete the assigned stories by the Scrum Master and deliver the output at the end of each sprint.

Scrum Events:



i) Sprint: A time-boxed period under which the developers are supposed to complete the specified tasks (writing code, etc.) they are assigned with.

A sprint is created by assigning a fixed no. of tasks to the developers known as “Stories”.

Each sprint cycle is 2 weeks/1 month long.

The new Sprint starts immediately after completion of the previous sprint.

ii) Sprint Planning: The product owner prioritizes the product backlog, defines the sprint backlog (and) decides the starting time and end time of the sprint. Scrum Master creates stories by collaborating with developers.

iii) Daily Scrum meetings: All scrum team members gather in a meeting for 15min. They Specify current progress, report and roadblocks.

iv) Sprint Review: Once all the developers complete their stories, the pushed code is checked for bugs and errors and various tests are performed to ensure the quality of the code, before deploying it into central repository.

v) Sprint Retrospective: A meeting at the end of each sprint for the team members to take feedback to improve the future sprints.

Scrum Artifacts:

i) Product Backlog: An ordered list of all features, enhancements, and fixes needed to build and maintain the product, which is prioritized by the product owner.

ii) Sprint Backlog: A subset of product backlog selected by Product Owner for a specific sprint.

Advantages:

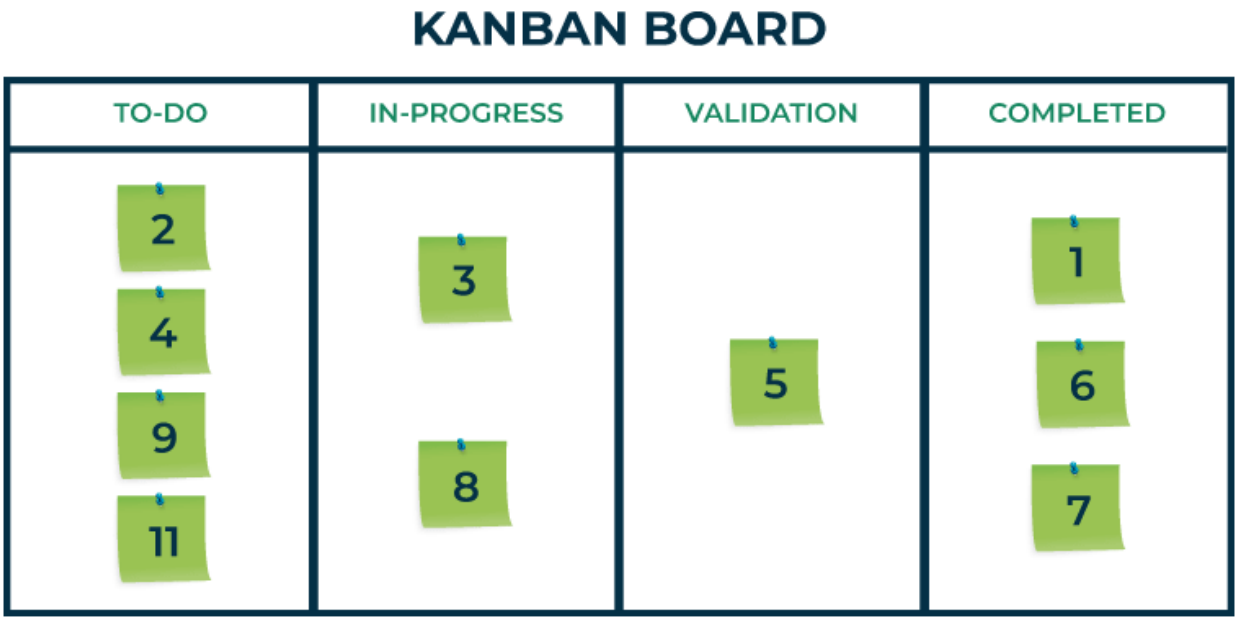
- Fast moving and money efficient.
- Works by dividing the large product into small sub-products. It’s like a divide and conquer strategy.
- Customer satisfaction is prioritized.
- Relies on constant feedback therefore the quality of product increases in less amount of time.

Disadvantages:

- Scrum framework does not allow changes into their sprint.
- Scrum framework is not fully described model. If you want to adopt it you need to fill in the framework with your own details like Extreme Programming (XP), Kanban, DSDM.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources, such as time for making reports and attending meetings, etc.

(2) Kanban:

A non-iterative agile framework that aims to visualize the work, limit work-in-progress (avoiding having the work from leaving incomplete) and optimize the workflow.
The framework uses a Kanban Board to visualize the workflow of the entire project.
Kanban Board:



->It is a work board that is divided into 4 columns. The individual columns represent the workflow phases of the project.
->The workflow phases are To-Do, In-Progress, Validation, and completed. A Workflow phase represents the status of the work item.
->The tasks of project are written on sticky notes, known as “user stories”. (work items)
-> Initially these stories are placed in the To-Do column. As the work progress, the story keeps on moving through In-Progress, Validation columns. If the story is completed, then it is placed in completed column.

Bottlenecks:

In DevOps, a bottleneck refers to a point in the software development and delivery process where the flow of work is impeded, causing delays and reducing overall efficiency. Identifying and addressing bottlenecks is crucial for optimizing the entire DevOps pipeline and ensuring a smooth, continuous delivery of software. Here are common areas where bottlenecks may occur in DevOps:

1. Code Development:
   - Bottleneck: Slow development cycles or delays in completing code.
   - Causes: Complex coding tasks, lack of collaboration, insufficient code reviews.
   - Solution: Encourage collaboration, implement efficient code review processes, and provide adequate training.

2. Build and Compilation:
   - Bottleneck: Slow build times or compilation processes.
   - Causes: Large codebases, complex dependencies, inadequate build infrastructure.

- Solution: Optimize build scripts, parallelize builds, use caching mechanisms, and invest in scalable build infrastructure.

3. Testing:
   - Bottleneck: Delays in testing processes, including unit, integration, and regression testing.
   - Causes: Insufficient test automation, lack of parallel testing, long feedback loops.
   - Solution: Increase test automation, parallelize testing, adopt shift-left testing practices, and optimize testing environments.

4. Deployment and Release:
   - Bottleneck: Slow or error-prone deployment processes.
   - Causes: Manual deployment processes, lack of automation, complex release procedures.
   - Solution: Implement continuous delivery practices, automate deployment pipelines, and use infrastructure as code for reproducible environments.

5. Infrastructure Provisioning:
   - Bottleneck: Delays in provisioning or configuring infrastructure.
   - Causes: Manual provisioning, lack of automation, resource constraints.
   - Solution: Embrace infrastructure as code, automate provisioning processes, and use cloud services for scalability.

6. Monitoring and Feedback:
   - Bottleneck: Inadequate monitoring or delayed feedback on application performance.
   - Causes: Limited monitoring tools, lack of real-time feedback, incomplete metrics.
   - Solution: Implement comprehensive monitoring solutions, establish feedback loops, and continuously analyze performance metrics.

7. Collaboration and Communication:
   - Bottleneck: Poor communication and collaboration between development and operations teams.
   - Causes: Siloed teams, lack of shared objectives, communication gaps.
   - Solution: Foster a DevOps culture, encourage cross-functional collaboration, and use collaborative tools for communication.

8. Manual Processes:
   - Bottleneck: Reliance on manual processes throughout the DevOps pipeline.
   - Causes: Resistance to automation, outdated workflows, manual approvals.
   - Solution: Automate repetitive tasks, introduce continuous integration and continuous delivery (CI/CD), and minimize manual interventions.

9. Resource Constraints:
   - Bottleneck: Insufficient resources, either human or infrastructure.
   - Causes: Limited budget, understaffed teams, inadequate hardware.
   - Solution: Allocate sufficient resources, invest in training, and consider cloud-based solutions for scalability.

Identifying and addressing bottlenecks require a combination of technical solutions, process improvements, and cultural changes. Regularly reviewing and optimizing the DevOps pipeline is essential for maintaining a high level of efficiency and delivering value to end-users consistently.