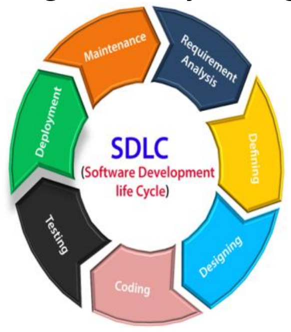
UNIT 1

Introduction: Introduction, Agile development model, DevOps, and ITIL. DevOps process and Continuous Delivery, Release management, Scrum, Kanban, delivery pipeline, bottlenecks, examples.

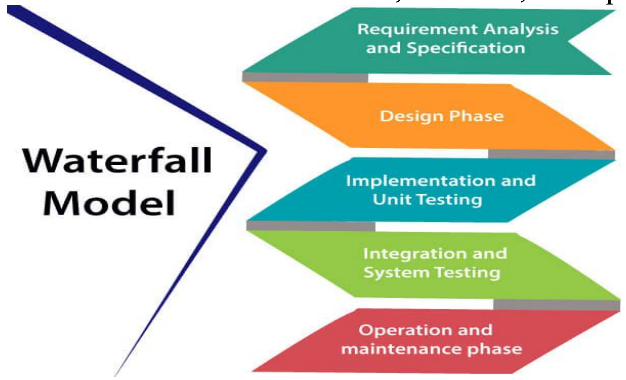
Introduction:
Software Development Life Cycle (SDLC)
A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.



Stage 1: Planning and requirement analysis
Requirement Analysis is the most important and necessary stage in SDLC.
The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.
Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.
Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.
For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.
Once the required function is done, an analysis is complete with auditing the feasibility of the growth of a product. In case of any ambiguity, a signal is set up for further discussion.Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.
Stage 2: Defining Requirements
Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders. This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.
Stage 3: Designing the Software
The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.
Stage 4: Developing the project
In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.
Stage 5: Testing
After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage. During this stage, unit testing, integration testing, system testing, acceptance testing are done.

Stage 6: Deployment
Once the software is certified, and no bugs or errors are stated, then it is deployed.
Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.
Stage 7: Maintenance
Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

Waterfall model
Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.
1. Requirements analysis and specification phase: The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.
2. Design Phase: This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).
3. Implementation and unit testing: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.
During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.
4. Integration and System Testing: This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.
5. Operation and maintenance phase: Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.


Waterfall Model
Requirement Analysis and Specification
Design Phase
Implementation and Unit Testing
Integration and System Testing
Operation and maintenance phase

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.

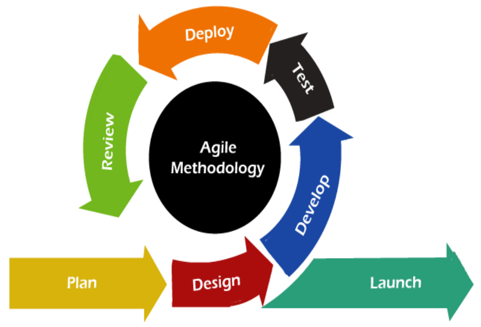
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

Agile Development Model

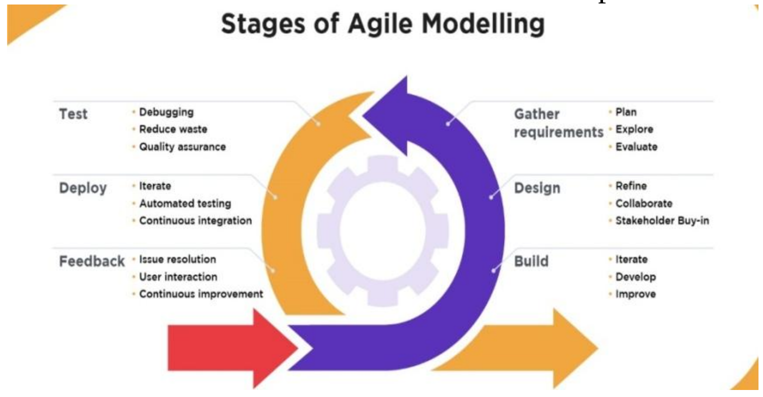
An agile methodology is an iterative & incremental approach to software development. This model breaks the product into small pieces & integrates them for final testing. The Agile development model can lead to increased employee input, faster solutions, and better risk management.



It can be implemented in many ways such as Kanban, XP, Scrum etc. Each Iteration is considered as a short time frame in agile model process. The division of entire project is into smaller parts that helps to minimize the project risk & to reduce the overall delivery time requirement

Agile Development Model focus on 4 core values, such as:

- Working software over comprehensive documentation
- Responded to change over following a plan
- Customer collaboration over contract negotiation
- Individual & team interaction over the process & tools



The Agile development model can lead to increased employee input, faster solutions, and better risk management. Agile Software Development Methodology enhances collaboration, customer feedback, and quick iterations. Unlike other software development methods, Agile does not offer a solution; instead, it offers various frameworks customized to specific industries, project types, and team sizes.

There are several different cycles in Agile development starting from the portfolio-level planning, then moving down to individual team cycles like Scrum and Kanban, and finally reaching the granular level of Continuous Integration (CI) where code changes are frequently integrated and tested, representing a complete cycle within the development process.

Scrum: A popular agile framework where teams work in short "sprints" with defined timeframes, focusing on delivering working increments of software during each sprint. Scrum cycle can be 2 to 4 weeks & are often used by development team using scrum agile process.
Kanban: Agile methodology that emphasizes visualizing work flow and limiting work in progress using a board with different columns representing stages of development. Kanban emphasize 24 hr cycle, which is popular in operation teams. It is used for early delivery by visualizing the tasks.

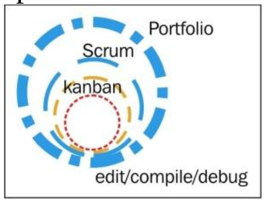


FIGURE: The Agile wheel of wheels

When to use agile model:

- when frequent changes required.
- when highly qualified & experienced teams is available
- when customer is ready to have a meeting with a software team all the time.
- when project size is small.

Pros

- frequent delivery
- face to face communication with clients
- efficient design
- any time change are acceptable
- it reduces total deployment time

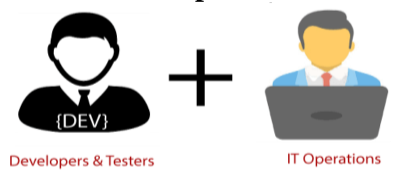
Cons

- due to shortage of formal documentation it create confusion & crucial decision
- once project complete another project is allotted, maintenance of project is difficult

Introduction to DEVOPS

DevOps is set of practice that combines Software Development (Dev) & IT Operations (Ops) to improve the speed, quality & reliability of software. DevOps is not a tool or application it is a methodology or practice.
The word DevOps is combination of two words software development & operation. This allows a single team to handle the entire application life cycle from development to testing, development & operations.
DevOps in a practice where the collaboration between different disciplines of software development is encouraged. It helps you to reduce the disconnection between software developers; Quality assurance (QA) engineers system administrators.
It promotes collaboration between development & operation team to deploy code to production faster in an automated & repeatable way. It helps to increase organization speed to deliver applications & services. It also allows organization to serve their customer better & compete more strongly in the market.
It can also be defined as a sequence of development & IT operations with better communication & collaboration.
It has become one of the most valuable business disciplines for enterprises or organization, with the help of DevOps quality & speed of application delivery has improved to gereat extent. Some of DevOps tool are Git, Github, Docker, Jenkins, Kubernets, selenium, puppet etc.

What is DevOps?



Need of DevOps:

- The operation and development team worked in complete isolation.
- After the design-build, the testing and deployment are performed respectively. That's why they consumed more time than actual build cycles.
- Without the use of DevOps, the team members are spending a large amount of time on designing, testing, and deploying instead of building the project.
- Manual code deployment leads to human errors in production.
- Coding and operation teams have their separate timelines and are not in synch, causing further delays.

DevOps History

- In 2009, the first conference named **DevOps days** was held in Ghent Belgium. Belgian consultant and Patrick Debois founded the conference.
- In 2012, the state of DevOps report was launched and conceived by Alanna Brown at Puppet.
- In 2014, the annual State of DevOps report was published by Nicole Forsgren, Jez Humble, Gene Kim, and others. They found DevOps adoption was accelerating in 2014 also.
- In 2015, Nicole Forsgren, Gene Kim, and Jez Humble founded DORA (DevOps Research and Assignment).
- In 2017, Nicole Forsgren, Gene Kim, and Jez Humble published "Accelerate: Building and Scaling High Performing Technology Organizations".

DevOps advantages:

- It an excellent approach for quick development & deployment of application. It responds faster to the market changes to improve business growth.
- It improves customer experience & their satisfaction. It also escalate business profit by decreasing software delivery time & transportation cost.
- DevOps means collective responsibility which leads to better team engagement & productivity.

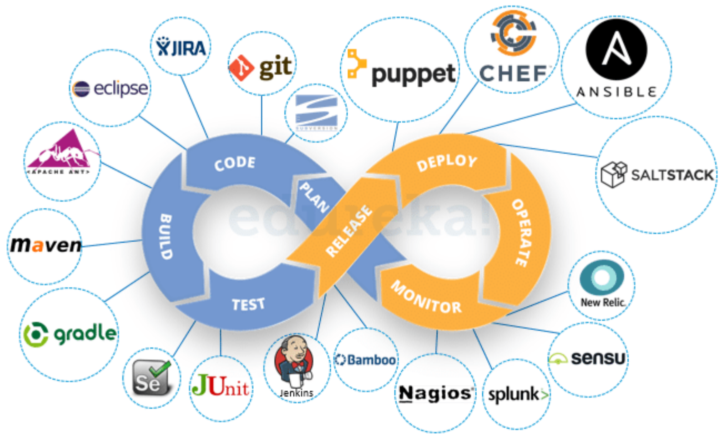
DevOps Disadvantages:

- Professional or expert developers are less available.
- Developing with DevOps is expensive.
- Adopting new technology into the industries is hard to manage in short time. Lack of DevOps knowledge.

DevOps Architecture:

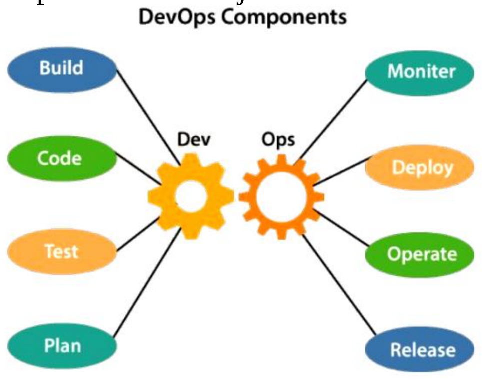
Development & operation both play essential role in order to deliver application. The development comprises of analyzing requirement, designing, developing & testing of software component. The operation consist of the administrative processes, services & support of the software i.e.; maintenance. When both the development & operation combined with collaborating then the DevOps architecture is a solution to fix the gap between development team and operational team to accelerate delivery through fast feedback, automation, maintainability, predictability & efficiency of operational process. DevOps architecture is used for the application hosted on the cloud platform & large distributed applications. Ex: Netflix, Amazon etc.

Agile development is used in DevOps architecture so that integration & delivery can be contiguous. When the development & operation team works separately from each other, then it is time consuming to design, test & deploy & if the terms are not in sync with each other, then it may cause a delay in delivery, so DevOps enables the teams to change the short coming & increases productivity.



Components of DevOps Architecture:

- Plan: In this phase DevOps use agile methodology to plan the development with operation & development team in sync, it helps in organizing the work to plan accordingly to increase the productivity. During initial phase, the scope & objective of software project is defined. This includes gathering requirement, setting goals & creating roadmap for development.
- Code: In this phase developers write the actual source code for the software based on the requirements & plans establishing in the previous phase. This is where the core functionality of software is implemented.
- Build: In this phase the source code is compiled or built into executable files or artifacts. Tools like compiler, build scripts are used to create a deployable package from the codebase.
- Test: It is critical phase, where software is subjected to various type of testing, which includes unit testing, system testing & integration testing. The goal is to identify bugs of fix the bugs to ensure functionality works as expected.
- Deploy: In this phase deployment involves the process of releasing software to the target environment. Deployment may be manual or automated, depending on DevOps practices.
- Operate: After deployment, the software is actively used by end users. During the phase, ongoing operations such as monitoring, maintenance & support are carried out to ensure the software remains operational & perform well.
- Monitor: Continuous monitoring is essential to track the performance, availability & security of software in the environment. Monitoring tools & practices helps to identify issues, & ensure the software meet the performance objective.



Difference between DevOps & Agile:

DevOps

- It is a practice to bring development & operation team together.
- Purpose is to manage end to end engineering process.
- Focuses on constant testing & delivery.
- Large team size.
- End to end business solution & fast delivery target.
- DevOps divide & spreads the skill set between the development & operation team.

Agile
- Refers to continuous iterative approach which focus on collaboration & customer feedback.
- Purpose is to manage complex projects.
- Focuses on constant changes.
- Small team size.
- Software development is main target.
- Agile development emphasize training all team members to have wide variety of similar skills.

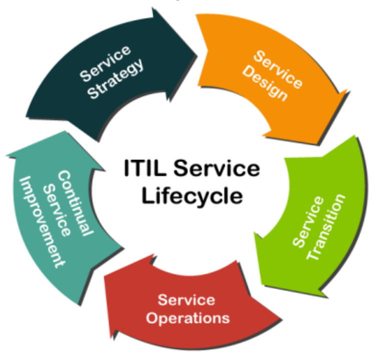
ITIL

ITIL is an abbreviation of **Information Technology Infrastructure Library**.

It actually focuses to define & document IT process procedure with a large focus on planning. It is used to guide the team to improve the value of service by focusing on solving the business issues & creating business values. It is set of principles for IT service management and highly structured model built to boost the productivity which improves the predictability & IT service delivery efficiency.

ITIL has been adopted by thousand of organizations worldwide like NASA, HSBC, Microsoft. Current version which is used in market is ITIL 4. It provides a practical & flexible basis to support service management that focus on customer experience & value. It is a framework for service management & involves 5 key stages:
- Service strategy: All managers follow instruction to develop a service strategy that ensure companies can manage all associated costs & risks.
- Service operation: It include technical support teams & application management that respond when an issue has impact on business.
- Service design: It involves the services architecture is developed & the business needs are translated into technical requirements.
- Service transition: In this stage, all assets are controlled to deliver a complete service for testing & integration.
- Continuous Service improvement: It is reflective approach that involves 4 stages to check the services are always in line with the demand of business.

Service Lifecycle in ITIL



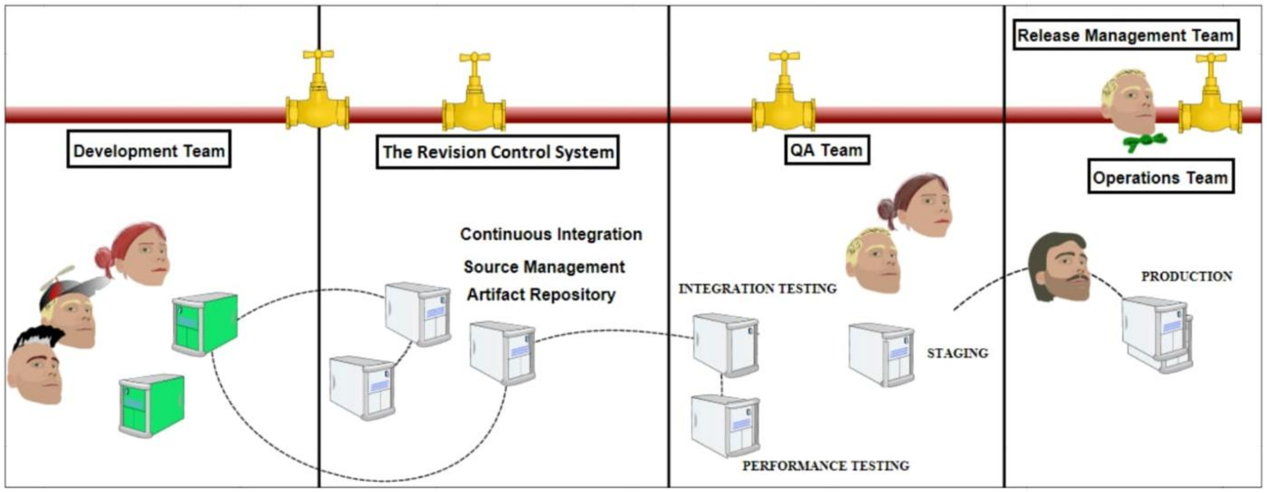
Misconceptions about DevOps vs. ITIL:

1. DevOps can replace ITIL. DevOps = continuous development, integration, and automated delivery. ITIL = framework for service management.
2. ITIL/ITSM is always documentation (own guidelines but decision made by IT teams)
3. ITIL/ITSM is only for large companies (can also be used for small business)

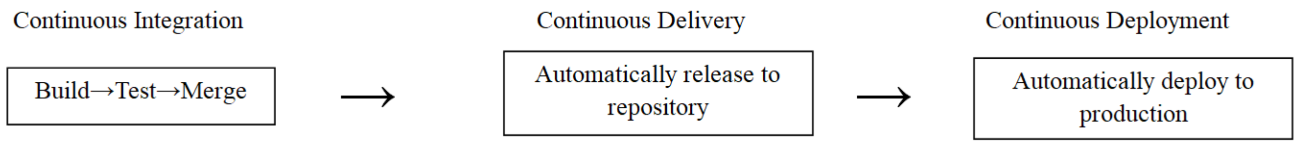
Difference between DevOps & ITIL:

It is combination of developer & operation.	Means information technology infrastructure Library.
Refers to effective collaboration between development & operation team.	Refers to set of guidelines for effective & efficient management.
Focus on concept that has dynamic body of knowledge.	Focus on development and has static body of knowledge.
Continuous integration, continuous delivery are backbone of DevOps philosophy.	Aims to increase the delivery process

DevOps Process & Continuous Delivery
It is set of practices, tools that automate & integrate the processes between software development & IT teams. It focuses on closing the loops between development and operation & driving product thigh continuous development, integration, testing, monitoring, feedback, delivery & deployment.
Continuous Delivery (CD) is the process of automating build, test, configuration & deployment from a build to production environment. CD value has become mandatory requirement for organization to deliver value to end users.
A release pipeline can create multiple testing or staging environment to automate infrastructure creation & deploy new builds. Example of CD pipeline in large organization is introduced is the following diagram:



The Developers: Developers work on their workstation. They develop code & need many tools to be efficient. Developer would have product like environment available to work locally on their workstation or laptops.
Depending on the type of software that is being developed, this might actually be possible, but it’s more common to simulate or rather mock, the parts of the product environment that are hard to replicate.
CD is a lean practice with a goal to keep product fresh with fastest path from new code or component availability to deployment. CD makes up a part of CI/CD a method to frequently deliver software by automating some of stages of application development.
CI is where new code changes to an application that are regularly built tested & merged into shared repository.



Revision control system: The revision control system is often the heart of development environment. The code that forms the organization software product is stored here. It is also common to store the configuration that forms the infrastructure here. If you are working with hardware development, the designs might also be stored in RCS. If you have great deal with independent component, you may decide to use a separate repository for each of them.
QA Team / Test Environment: After the build server has stored the artifacts in binary repository, they can be installed into test environment. Staging/production environment are the last line of environment, they are interchangeable with product environment. You can install new releases in the staging servers, and then check that everything works well & then swap out older production sever & replace that with staging server which well then becomes the new production server.
Not all organization has the resource to maintain production quality, staging servers, but when it is possible, it is safe to handle upgrade.
Operation Team: Operation team is like the final aspect of CD pipeline. It will maintain the deployed production by the development team.

Release Management

Release management refers to the process of planning, coordinating and deploying software releases to production environments. The goal of release management is to ensure that new features, bug fixes, and enhancements are delivered to end-users in a reliable, efficient, and timely manner.

Release management is a critical component of DevOps, as it helps to ensure that software is delivered to end-users in a timely and reliable manner, while minimizing the risk of errors and downtime

It is process of managing; planning, scheduling & controlling software build through different stages & environment including testing & deploying software release. DevOps emphasize the collaboration & communication of software developer & IT professional.

Release management in software development and IT operations is a system for managing the entire software delivery lifecycle — from planning to building to testing to deployment. Release & deployment management also ensure handover to service operation takes place & that suitable training & documentation exists to ensure ongoing support of new service. Deployment system usually has a way to support how to describe which software version to use is different environment.

Advantages of Release Management for DevOps

- Aligning business & IT goals
- Minimizes organizational risk
- Direct accelerating change

Release management best practices

As DevOps increases and changes accelerate, it is critical to have best practices in place to ensure that it moves as quickly as possible. Well-refined processes enable DevOps teams to more effectively and efficiently. Some best practices to improve your processes include:

- Define clear criteria for success

Well-defined requirements in releases and testing will create more dependable releases. Everyone should clearly understand when things are actually ready to ship. Release managers, quality supervisors, product vendors, and product owners must all have an agreed-upon set of criteria
before starting a project.

- Minimize downtime

DevOps is about creating an ideal customer experience. Likewise, the goal of release management is to minimize the amount of disruption that customers feel with updates. A good release manager will be able to identify any problems before the customer. The team can resolve incidents quickly and experience a successful release when proactive efforts are combined with a collaborative response plan.

- Optimize your staging environment

The staging environment requires constant upkeep. Identifying problems in staging before deploying to production is only possible with the right staging environment. Maintaining a staging environment that is as close as possible to production will enable DevOps teams to confirm that all releases will meet acceptance criteria more quickly.

- Strive for immutable

Whenever possible, aim to create new updates as opposed to modifying new ones. Immutable programming drives teams to build entirely new configurations instead of changing existing structures. These new updates reduce the risk of bugs and errors that typically happen when modifying current configurations.

- Keep detailed records

Good records management on any release/deployment artifacts is critical. From release notes to binaries to compilation of known errors, records are vital for reproducing entire sets of assets. In most cases, tacit knowledge is required.

- Focus on the team

Well-defined and implemented DevOps procedures will usually create a more effective release management structure. They enable best practices for testing and cooperation during the complete delivery lifecycle.

Although automation is a critical aspect of DevOps and release management, it aims to enhance team productivity. The more that release management and DevOps focus on decreasing human error and improving operational efficiency, the more they'll start to quickly release dependable services.

Scrum

Scrum is a framework used by teams to manage work & solve complex problem collaboratively in short cycles while delivery valuable product. Scrum implements the principle of agile as a concrete set of artifacts, practices & roles. They are several different cycles in agile development model. The entire lifecycle is completed in fixed time periods called sprints. A sprint is typically one-to-four weeks long. Scrum focuses on sprint cycles which occur biweekly or monthly.

Scrum Roles: There are three main key roles:

1. Product Owner: Product owner is responsible for what the team builds & why they build it.
2. Scrum Master: Scrum master ensure the scrum process is followed by the team, it looks how the team can improve, resolve & other blocking issues.
3. Scrum Teams: The members of scrum team actually build the product; the team owns the engineering of the product & the quality.

Scrum defines a practice called a *daily Scrum*, often called the *daily standup*. The daily Scrum is a daily meeting limited to fifteen minutes. Team members often stand during the meeting to ensure it stays brief. Each team member briefly reports their progress since yesterday, the plans for today, and anything impeding their progress.

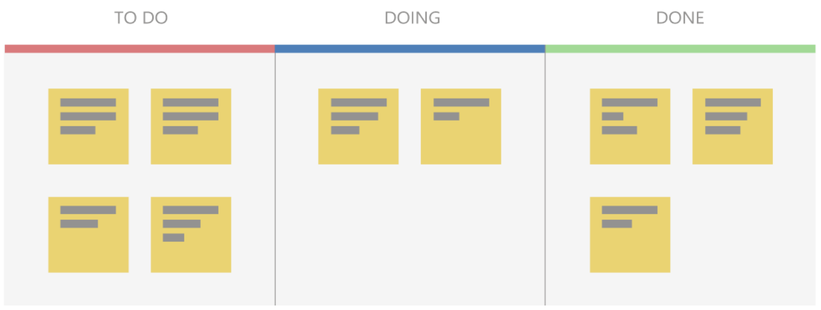
Scrum is very popular because it provides just enough frameworks to guide teams while giving them flexibility in how they execute. Its concepts are simple and easy to learn. Teams can get started quickly and learn as they go. All of this makes Scrum a great choice for teams just starting to implement agile principles.

Benefits of Scrum: Timely production, higher software quality, flexible to change, easily scalable.

Kanban

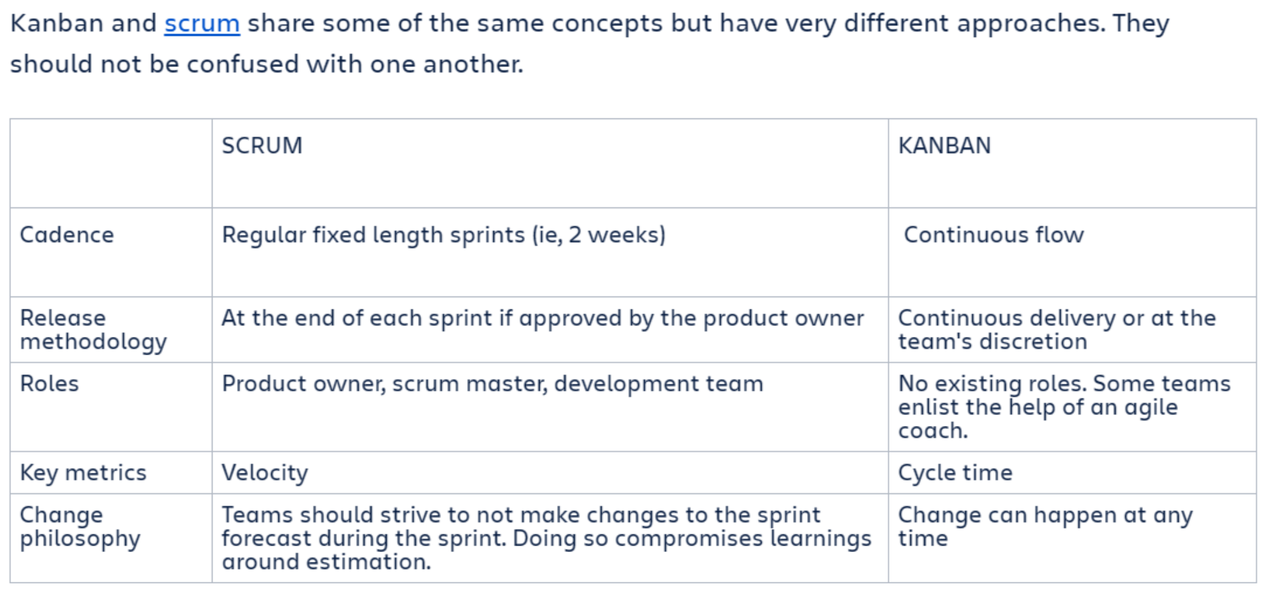
Kanban is a Japanese term that means signboard or billboard. Although Kanban was created for manufacturing, software development shares many of the same goals, such as increasing flow and throughput. Software development teams can improve their efficiency and deliver value to users faster by using Kanban guiding principles and methods.

Kanban is popular framework used to implement agile & DevOps software development. It requires real time communication & full transparency of work. It focuses more on shorter cycle which can occur daily. The *Kanban board* is one of the tools teams use to implement Kanban practices Work item are represented visually on kanban board, allowing team members to see the state of every piece of work at any time.



Benefits of Kanban: planning flexibility, shortened time cycles, fever bottlenecks: multitasking kill efficiency, visual metrics, continuous delivery.

Scrum VS Kanban



Kanban and scrum share some of the same concepts but have very different approaches. They should not be confused with one another.

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Delivery Pipeline:

A delivery pipeline is a process that drives software development through a path of building, testing & deploying the software & also known as continuous integration, continuous delivery/deployment (CI/CD) & Continuous Delivery Pipeline (CDP).

- Delivery pipeline focuses on streamlines the process of building, testing, and deploying software.
- Components used are automated builds, automated testing, static code analysis, frequent commits, and a well-defined promotion process
- The goal is to enables developers to quickly deliver high-quality software to customers.

By automating the process, the objective is to maintain a consistent process for how software is released by minimizing human error. CI/CD is the backbone of DevOps methodology, bringing developer & IT operation team together to deploy software delivery pipeline that manage the following scenarios:

The build server supports the generation of objective code quality metric that we need in order to make decisions. These can be made automatically or based on mutual decisions.

The deployment pipeline can also be handled manually. This can be handled with an issue management system via configuration code commits or both.

DevOps perspective, it doesn’t really matter if we use scrum scaled agile model framework, Kanban or other method with in lean or agile frameworks. Even a traditional waterfall model can be successfully managed -- DevOps serves all.

Identifying Bottlenecks

A bottleneck is a part of congestion in a production system (such as assembly line or computer network) that stops or severely slow down the system. A bottleneck in DevOps is a factor that limits the efficiency or capacity of a process or system. It can occur at any stage of the DevOps pipeline, such as testing, integration, delivery, deployment, or code development.

Two main types of bottleneck are: Long term and Short term.

Short term bottleneck is temporary & typically caused by temporary condition such as employees on vacation or on sick leave.

Long term bottleneck are built into the manufacturing protocol & often related to inefficient equipment or processes.

Bottlenecking, the process that create bottlenecks, can have a significant impact on the flow of manufacturing & can sharply increases the time & expense of production.

Bottlenecks have a negative effect on practical production capacity, keeping it further below theoretical capacity than normal. Eliminating bottleneck is key to increasing production efficiency.

Operation management is concerned with controlling the production process, identifying potential bottleneck before they occur & finding efficient solution.
Bottlenecks can occur in any project methodology because all methodologies involve some form of progression or sequence that can be slowed.
The process of tracking down bottlenecks is called performance analysis. Specialized tools such as profilers or performance analyzers can help identify bottlenecks.
The goal is to improve the bottleneck component so that the entire system can achieve the desired performance.
According to the theory of constraints, improving the bottleneck constraint's hot-spot point can improve the software's overall processing speed.