DEVOPS

Objectives of this Course:

- The Main objectives of this DevOps Course Include

→ Describe the Agile Relationship B/w Development & IT operations.

→ Understand the Skillset & High-Functioning Teams Involved in DevOps & Related Methods to reach a Continuous Delivery Capability.

→ Implement Automated System Update & DevOps lifecycle.

Outcomes of this Course:

On Successful Completion of this Course, Students will be able to:

→ Identify Components of DevOps Environment.

→ Describe S/w Development Models & Architectures of DevOps.

→ Apply different Project Mgmt, Integration, Testing & Code Deployment Tool.

→ Investigate different DevOps S/w Dev. Models.

→ Assess Various DevOps Practices.

→ Collaborate & Adopt DevOps in Real-Time Projects.

UNIT-I : INTRODUCTION.

- Introduction
- Agile Development Model
- DevOps
- ITIL
- DevOps Process & Continuous Delivery
- Release Management.
- Scrum
- kanban
- Delivery Pipeline
- Bottlenecks

* Introduction to DevOps :→

→ DevOps is Basically a Combination of Two words. Development & Opeeations.

→ DevOps is a Culture that Implements the Technology in ordee to promote Collaboeation b/w the developee team & the opeeation's team to deploy code to production faster in an automated & repeatable way.

Why DevOps?

→ The Goal of DevOps is to Increase an Organization's speed when it Comes to delivering applications & Seevices.

→ Many Companies have Successfully Implemented DevOps to enhance their User Expeeience Including Amazon, Netflix, etc.

→ Facebook's mobile App which is updated Every Two weeks effectively tells users... You Can Have What You Want & You Can Have It.

→ Now, Ever wondered how facebook was able to do Social Smoothing?

It's the DevOps philosophy that helps Facebook ensures that apps aren't outdated & that users get the Best Experience on Facebook.

→ Facebook accomplishes this True Code ownership model that makes its developers responsible that includes Testing & Supporting throughs Production & Delivery for Each Kernel of Code.

→ They write and update their true policies like this but Facebook has developed a DevOps culture & has successfully accelerated its development lifecycle.

→ Industries have started to gear up for digital transformation by shifting their means to weeks & months instead of years while maintaining High Quality as a Result.

→ The solution to All this is .... DevOps.

How DevOps is Different from Traditional IT?

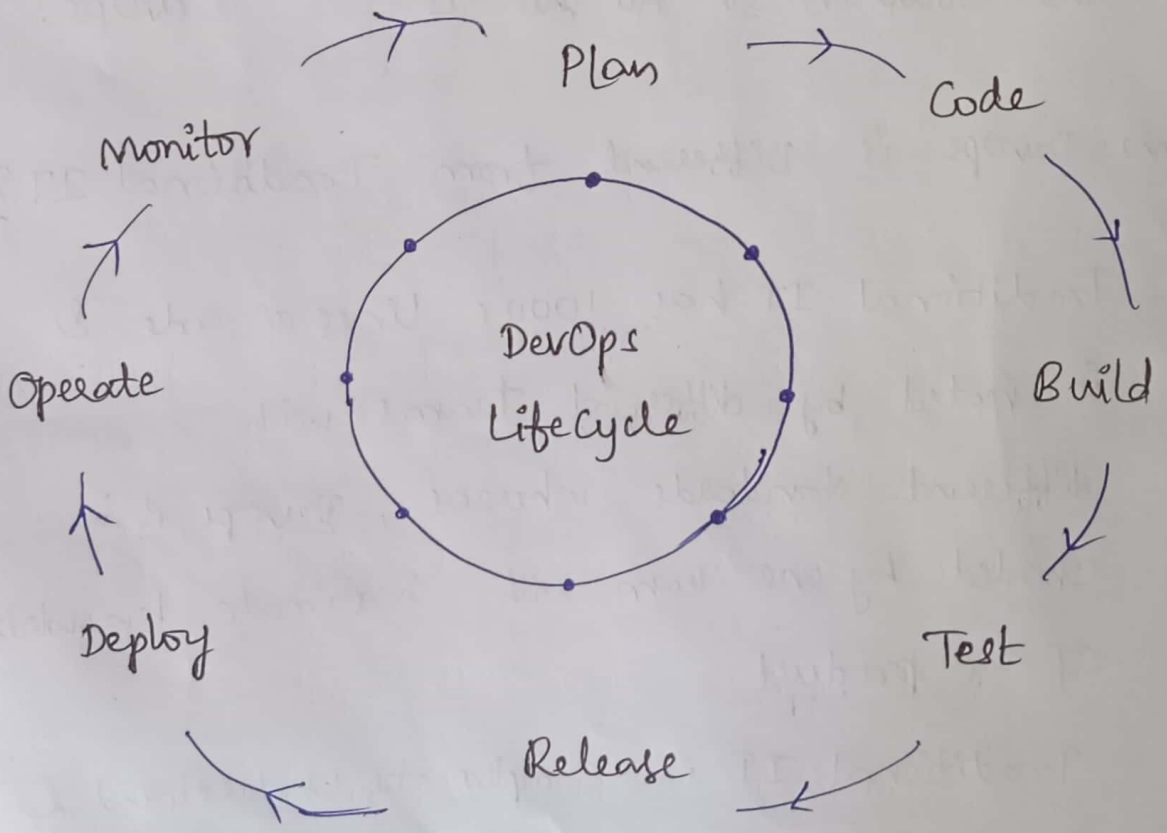
→ Traditional IT has 1000's lines of code & is created by different teams with different standards whereas, DevOps is created by one Team with intimate knowledge of the product.

→ Traditional IT is Complex to understand & DevOps is Easily Understandable.

* DevOps Lifecycle :

→ DevOps Lifecycle is the Methodology where Professional Development Teams Come together to bring products to market more efficiently & quickly.

→ The Structure of the DevOps Lifecycle Consists of PLAN, CODE, BUILDING, TEST, RELEASING, DEPLOYING, OPERATING, & MONITORING.



PLAN: Determining the Commercial Needs & 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 & In order to simplify the Design, The Team of Developers uses Tools & 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, & 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 & subsequently publishes the build using various DevOps lifecycle Tools.

OPERATE : At this Level, The Available version is ready for User's to use.

Here, The Department looks after the server Configuration & Deployment.

MONITOR : The Observation is done at this level that depends on the data which is gathered from Customer Behaviours, The Efficiency of Applications, & from various other Sources.

* Best Practices to Follow:

→ Implement Automated Dashboard.

→ Keep the Entire Team Together

→ Allow DevOps to be a cultural change.

→ Be patient with the Developers

→ Maintain a centralized Unit.

→ Build a Flexible Infrastructure.

* Advantages & DisAdvantages of DevOps:

*Advantages:

1. Faster Delivery:

DevOps enables organizations to release New Products & Updates faster & More frequently, which can lead to a Competitive Advantage.

2. Improved Collaboration:

DevOps Promotes Collaboration B/w Development & Operations Teams, Resulting in better Communication, Increased Efficiency, & Reduced Friction.

3. Improved Quality:

DevOps Emphasizes Automated Testing & Continuous Integration, which helps to catch bugs early in the development process & Improve the overall quality of s/w.

4. Increased Automation:

DevOps enables organizations to automate many manual processes, freeing up time for more Strategic work & reducing the risk of Human Error.

5. Better Scalability:

DevOps enables Organizations to quickly & 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 & Loyalty.

7. Improved Security:

DevOps promotes security Best Practices, such as Continuous Testing & Monitoring, which can help to reduce the risk of security breaches & Improves the overall security of an Organization's System's.

8. Better Resource Utilization:

DevOps enables Organizations to optimize their use of resources, including Hardware, Software, & Personnel, which can result in Cost Savings & Improved Efficiency.

* DisAdvantages :

1. High Initial Investment :

Implementing DevOps can be a complex & costly process.

It requires significant Investment in Technology, Infrastructure & personnel.

2. Skills shortage :

Finding Qualified DevOps Professionals can be a challenge, & Organizations may need to Invest in Training & Development programs to build the necessary skills within their Teams.

3. Resistance to change:

Some Employees may resist the cultural & Organizational changes required for Successful DevOps adoption, which can result in resistance, resistance to collaboration, & reduced efficiency.

4. Lack of Standardization:

DevOps is still a relatively new field & there is a lack of standardization in terms of methodologies, Tools, & Processes.

This can make it difficult for organizations to determine the best approach for their specific needs.

5. Increased Complexity:

DevOps can increase the complexity of S/W Delivery, Requiring organizations to manage a larger no. of moving parts & integrate multiple systems & Tools.

6. Dependency on Technology:

DevOps relies heavily on Technology, & organizations may need to invest in a variety of Tools & platforms to support the DevOps process.

7. Need for Continuous Improvement:

DevOps requires Ongoing Improvement & Adaptation, As new Technologies & Best Practices emerge.

Organizations must be prepared to Continuously adapt & evolve their DevOps Practices to remain Competitive.

* Agile Development Model:

The Agile Methodology is a Project Mgmt Approach that involves breaking the project into phases & Emphasizes Continuous Collaboration & Improvement.

Teams follow a cycle of planning, Executing, & Evaluating.

* What are the 3 C's in Agile?

the 3 C's of Agile Methodology -

Collaboration, Communication & Coordination

These are the Essential Elements that helps to ensure the success of Any Agile Project.

* How Many Pillars are in Agile?

All Agile Methodologies are based on FOUR Fundamental values, The So-called four Pillars:

→ Individuals
→ Interactions
→ Working software
→ customer Collaboration.

* What are KPI's in Agile?

→ KPI stands for Key Performance Indicator.
→ It is a means of measuring a Team's Performance to ensure they are on Track to hit their Project Objectives.
→ KPI's are used in Many Departments, Including Finance, Customer Success, & Marketing.

* what is an EPIC in Agile?

An EPIC is a large body of work that can be broken down into a number of smaller stories, or sometimes called "Issues" in Jira.

→ EPICS often encompass multiple Teams, on multiple Projects, and can even be tracked on multiple Boards.

→ EPICS are almost always delivered over a set of SPRINT's.

* What is the Difference Between EPIC & STORY in Agile?

On an Agile Team, Stories are something the Team can commit to Finish within a one / Two week SPRINT.

Oftentimes, Developer's would work on Dozen's of Stories a month.

EPICS, in Contrast, are few in number & take longer to Complete.

Teams often have Two or Three Epics they work to Complete Each Quarter.

* What is the Cycle Time in Agile?

Cycle Time is the Amount of Time that a work Item spends in the Active stage of the work flow, from the Moment it is picked up by a Team Member until it is marked as done.

For Example, If a User Story is moved from the Backlog to The In-Process Column on Monday & it is completed on Friday, It's Cycle Time is Five Days.

* Who writes User stories in Agile?

Generally, A story is written by The Product Owner, Product Manager, or Program Manager & submitted for Review.

During a SPRINT or Iteration Planning Meeting, The Team Decides what stories they'll tackle that SPRINT.

* What is a Bug in Agile?

In software, A malfunction of the System, An Error, Flaw, or a Default in the System, that causes an Incorrect Result.

A Bug is when the System doesn't behave as Intended.

* Which Agile Framework is Best?

→ Scrum ... Scrum is one of the most well-known Agile Frameworks for Teams.

→ The Scrum Guide defines Scrum as,..

"A Framework within which people can address complex adaptive problems, while productively & creatively delivering products of the highest possible value."

* what is the difference b/w Agile & Scrum?

→ the Primary difference b/w Agile & Scrum is that Agile is a project Mgmt philosophy that Employs a Fundamental set of values / Principles.

→ whereas, Scrum is a Precise Agile Methodology utilized to facilitate a Project.

⊛ What are Agile Tools ?

Agile Tools provide a Digital workspace where teams can collaborate, plan & execute their projects using Agile Principles.

There are many Popular Agile Tools available such as JIRA for Agile Management.

Many companies offer JIRA Training to help Teams maximize their potential.

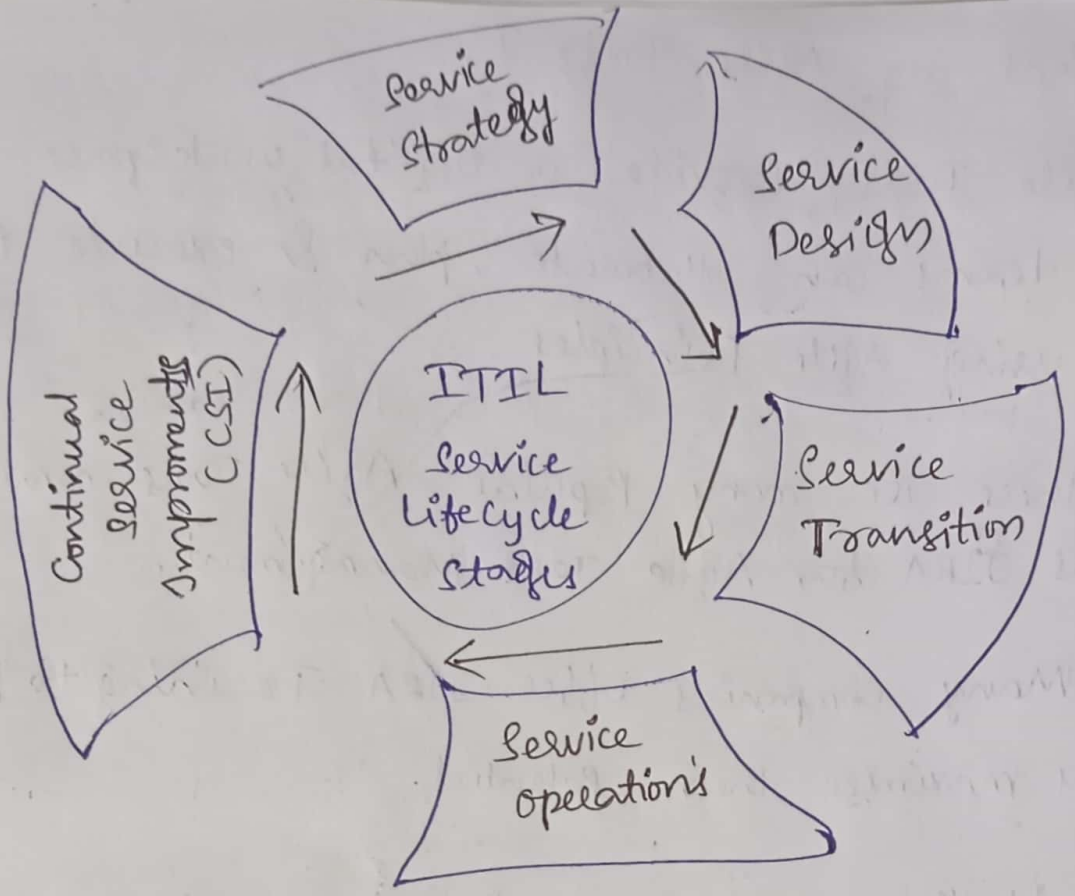
⊛ ITIL:

The Information Technology Infrastructure Library.

It is a set of practices and framework for IT Activities such as IT Service Management (ITSM) & IT Asset Mgmt (ITAM) that focuses on aligning IT services with the needs of the business.

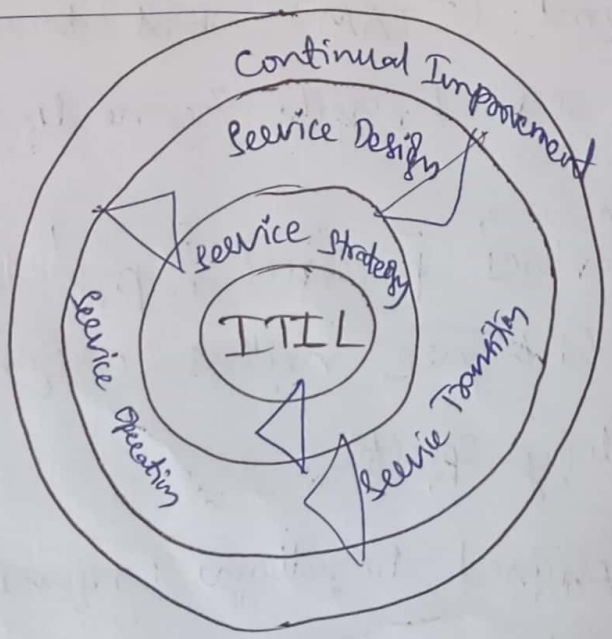
ITIL describes processes, procedures, tasks & checklists which are neither organization specific nor Technology-Specific.

It was designed to allow organizations to establish a baseline.



* ITIL Processes List & ITIL Service Lifecycle stages.

⊛ ITIL Framework:



Service Strategy:

- → Generation
- → Portfolio
- → Financial
- → Demand

Service Design:

- → Catalogue
- → Service Level
- → Availability
- → Capacity
- → Continuity
- → Security
- → Supplier

Service Transition:

- → Planning
- → Change
- → Configuration
- → Release
- → Validation
- → Evaluation
- → Knowledge

Service Operation:

- → Event
- → Incident
- → Request
- → Problem
- → Access

Continual Improvement:

- → Measurement
- → Reporting
- → Improvement.

⊛ DevOps Processes Explained in this way.

* Continuous Integration :

With Continuous Integration, Application developers frequently check in their changes to the source Environment & use an automated build process to verify these changes automatically.

Following are the Typical Activities that occur during continuous Integration.

- Code or Change Review
- Unit Test Execution.
- Code or Change Analysis.
- Smoke Test Execution.

Key Benefits from practicing this process are,

⊛ Smooth Integration

* Developers commit Code frequently.

⊛ Detect Problems early:

- Early & Frequent Integration of code detects the following:
- Integration Issues with other developer's code.
- Unit Test Issues.
- Code & Security Analysis based on the Tool leveraged, a wide spectrum of Issues are detected.
- Test failures - Leveraging Scenario, UI, & Integration Test Functional Integration Issues.

⊛ Solve Problems Early:

- Helps reduce the cost of Issue Resolution.
- Allows for focused Integration Testing later in the delivery life cycle as there is a confidence that the core units of the Application are validated.

Continuous Integration means,

Developers must check in frequently to integrate the changes with other's & get an early health of their changes.

⊛ Continuous Delivery:

With Continuous Delivery, Application changes Run through Automated Regression Testing & are Deployed to a staging Environment for further Testing.

This process ensures that the Application is ready for deployment on the production System.

The following items are the Typical Activities that occur during Continuous Delivery.

* Deployment to a staging or Acceptance Environment with Production, such as Configuration.

* End-to-End Functional Regression Testing.
* Performance & load testing against Production, such as Datasets.
* Security Compliance checks.
* Localization Testing
* Stakeholder signoff & Acceptance Testing.

Key outcomes from Adopting this process are:

* Repeatability through Automation.
  - → Focus on Automating Build package, deployment, testing, & Environment provisioning.
  - → Automation Allows the process to be repeated as often as necessary, daily, or even multiple times a day.
* Repeatable Automation reduces the chance of Human Errors.

* Lower Risk through Faster feedback:

→ Automated Testing provides early feedback on potential bugs & performance issues.

→ Emphasis on Testing in production, Environments provide reliable Indicators of the experience & performance in production.

* Issues that are caught are immediately reported to the developers to address as quickly as possible.

* Lower cost without Sacrificing Quality:

→ Automation significantly reduces the cost of delivery due to its repeatability.

→ Automated releases enable continuous Testing, ensuring that issues are caught quickly & early in the release cycle when it is easier & cheaper to fix.

⊛ Release Management:

In a DevOps Environment, Release Management refers to the process of planning, coordinating & Deploying s/w releases to production Environments.

The Goal of Release Management is to ensure that new features, bug fixes, & enhancements are ~~divided~~ delivered to end-users in a reliable, efficient, & timely manner.

Release Management is a Critical Component of DevOps, As It helps to ensure that Software is delivered to end-users in timely & reliable manner, while Minimizing the risk of Errors & downtime.

What is Release Management?

RM in s/w Development & IT Operations is a System for managing the Entire s/w delivery lifecycle — from planning to building to Testing to Deployment.

For Both ITIL & DevOps, this is the general process.

However, DevOps encourages more collaboration & visibility throughout the entire delivery process, shortening feedback loops & encouraging simpler, faster release management.

While the general concept of RM doesn't really change across ITIL & DevOps, the approaches different in two key ways.

→ ITIL is a framework specific to IT Service Mgmt. while DevOps looks at the collaboration across s/w Development & IT operations.

Let's look at how RM manifests itself for ITIL & DevOps Teams.

ITIL Release Management:

The Process for RM in ITIL 4 is to schedule & maintain the Integrity of new deployments, all the way from planning to Release.

In ITIL, The IT operations team will receive code from the s/w Developers & decide when and how to deliver the service while maintaining uptime for existing services.

DevOps Release Management:

In DevOps, RM is also about planning, scheduling & controlling the s/w Development & Delivery process. But, In DevOps, Both Developers & IT operations collaborate from the beginings of the process to the end allowing for fewer, shorter feedback loops & faster releases.

DevOps teams share accountability for the services they deliver, own their code & take on-call responsibilities.

With s/w Developers & IT professionals involved in the entire delivery lifecycle & on-call, Incidents are detected & resolved faster - both during the release process & after.

Stages in DevOps Release Management:

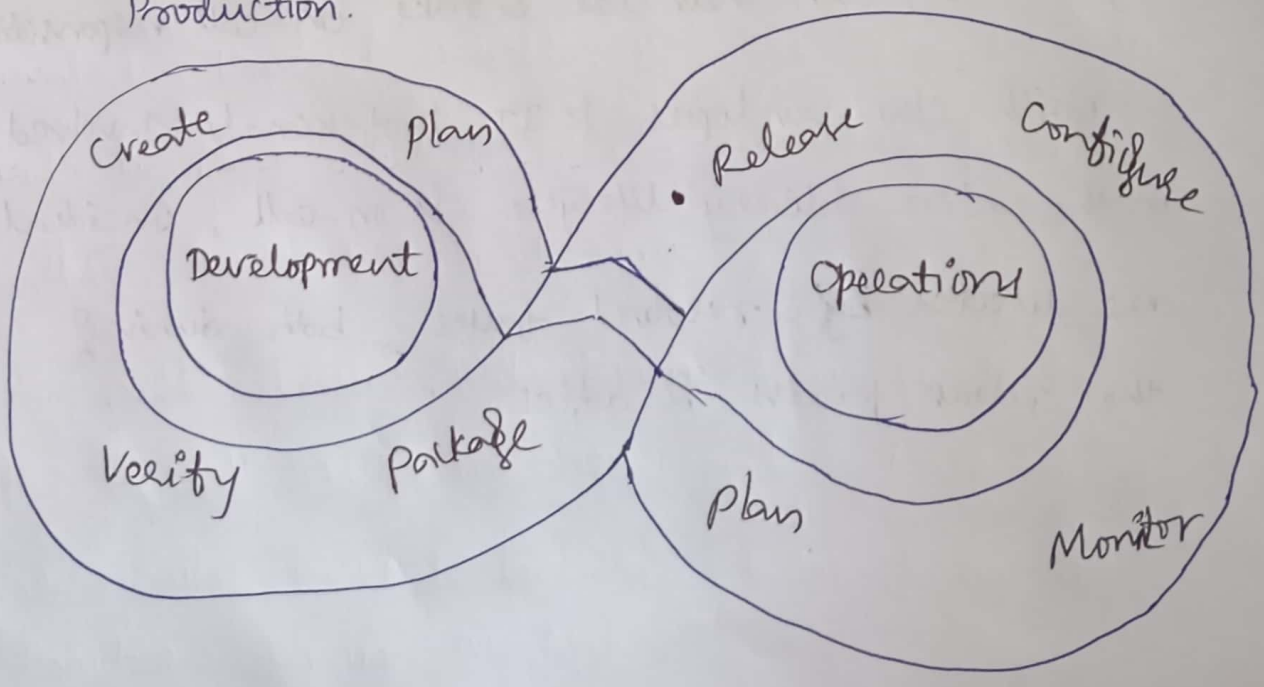
Release Management involves several stages, including Planning, Testing, Deployment & Monitoring.

The Processes are typically broken into 3 phases:

① The Development Team creates New code or features.

② Next, They test the new code/features in a staging environment.

③ Following thorough Testing & Approval, that code/New feature is then released to Production.



* Constantly strive for Minimal User Impact:

The Best Release Managers will Constantly work to reduce two significant Events:

- Downtime
- Impact on the customers.

Proactive Testing, Active Monitoring & Real-time collaborative alerting can help us in Identifying the issues during a release - many times before a customer will even notice.

Coupled with a "Collaborative" Incident response plan", the team can quickly resolve Incidents & continue along towards a successful release.

Scrum:

Scrum is a framework used by teams to manage work & solve problems collaboratively in short cycles.

Scrum Implements the principles of Agile as a Concrete set of artifacts, practices, & roles.

Scrum is a Management framework that teams use to self-organize & work towards a common goal.

It describes a set of Goals, Meetings, Tools, & Roles for efficient project delivery.

Much like a sports team participating for a Big Match, Scrum practices allow teams to self-manage, learn from experience, & adapt to change.

what is Scrum Meeting in DevOps?

Scrum Meetings are when the Scrum Master, Product owner, & Development Team meet to plan work, discuss work in progress, Gather feedback, and more.

Not Every Agile scrum Team needs to practice all scrum meetings, and a team doesn't necessarily have to be a Scrum Team to Practice Scrum Meetings.

What is the Role of Scrum Master in DevOps?

The Scrum Master serves to facilitate Scrum to the Larger team by ensuring the Scrum framework is followed.

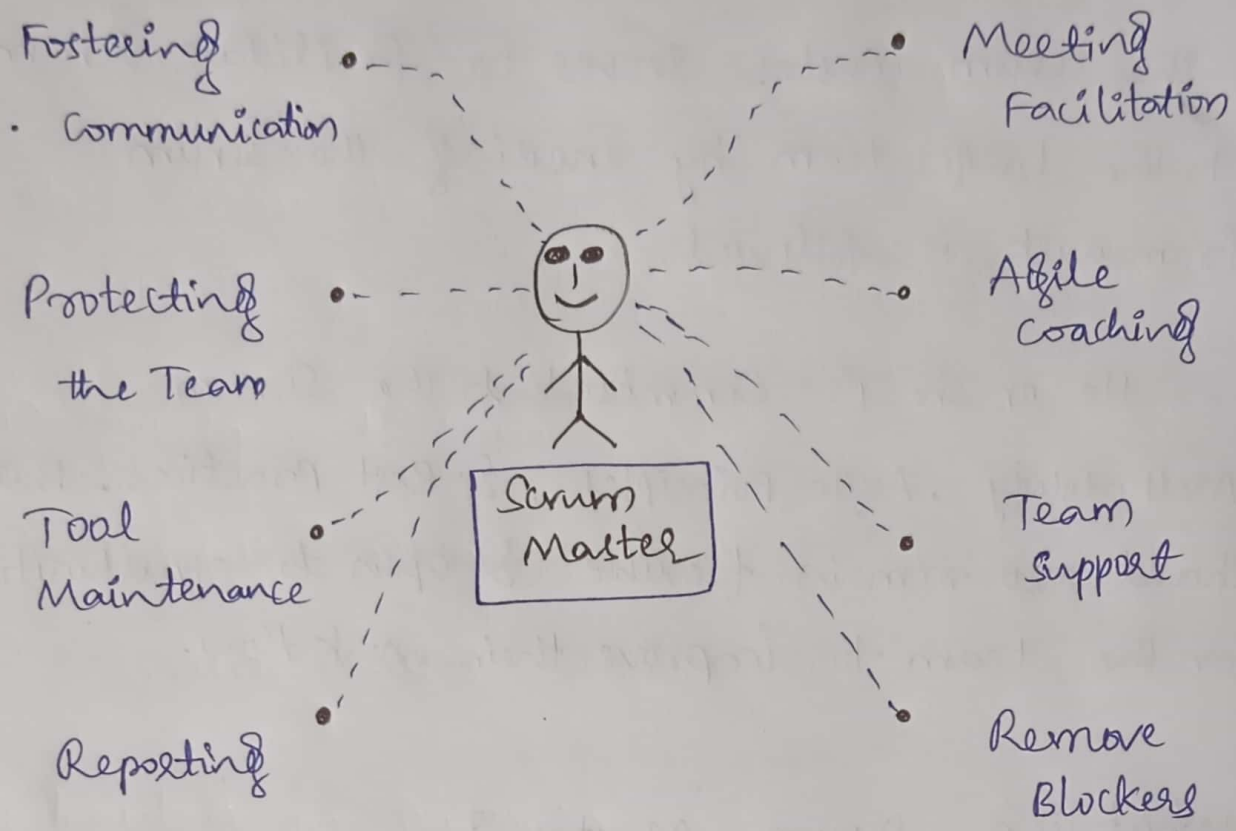
He or she is Committed to the Scrum Methodology, Agile Principles, & Best Practices - But should also remain flexible & open to opportunities for the team to improve their work flow.

What is a Scrum Master?

A Scrum Master is the facilitator of Scrum, A lightweight Agile framework focusing on time boxed Iterations called Sprints.

Scrum Master Acts as Coaches to the rest of the Team, or Servant leaders, as the Scrum Guide puts it.

Good Scrum Masters are Committed to the foundational elements of Scrum but remain flexible & open to opportunities for the team to Improve their workflows.



* Scrum Master Responsibilities :->

Although the Scrum Guide lists how Scrum Masters can serve other Scrum Team Roles, it doesn't provide an exhaustive list of Potential responsibilities.

Scrum Masters often perform many of the following duties.

1. StandUps
2. Steeation / Sprint Planning Meetings.
3. Sprint Reviews
4. Retrospectives
5. Board Administration
6. 1 on 1's :
7. Internal Consulting
8. Reporting
9. Blocker's
10. Busy Work

* Scrum Master skills:

1. Leadership
2. Effective Communication
3. Empathy
4. Problem Solving
5. Adaptability
6. Facilitation skills
7. Coaching & Mentoring
8. Technical Knowledge
9. Time Mgmt.

✱ Scrum Team :→

The Scrum Team is a fundamental aspects of the scrum framework.

It typically consists of the following roles :

① Development Team

② Product Owner

③ Scrum Master

✱ Kanban in DevOps :→

Kanban is a popular framework used to Implement Agile & DevOps s/w Development.

It requires real-time communication of capacity & full transparency of work.

✱ " Work items are represented visually on a kanban board ", allowing team members to see the state of every piece of work at any time.

Optimizing S/w Development with Kanban flow :→

Kanban flow, A cornerstone of Agile & DevOps methodologies, Drives efficiency by orchestating seamless task progression through visualized workflows.

Kanban flow mirrors the streamlined Inventory Mgmt of supermarkets, ensuring tasks move through development processes precisely when needed.

In S/w Development, kanban flow fosters dynamic task mgmt, acceleeates delivery cycles, & enhances customer satisfaction through focused, uninterrupted work.

Structuring the Kanban Flow :→

Establishing a structured kanban flow within the S/w development team is essential to implementing kanban effectively.

This ensures smooth task progression & optimized workflow Mgmt.

Here's how we can structure our kanban flow:

* Visualize workflow
* Standardize workflow
* Identify blockers & dependencies
* Set work-in-progress (WIP) Limits.
* Encourage Collaboration
* Utilize kanban cards

By structuring our kanban flow in this manner, we can streamline our s/w development processes, enhance Team Collaboration, & Maximize Efficiency in task Mgmt.

* What are the 5 Elements of kanban?

David Anderson Established that kanban boards can be broken down into five components.

→ Visual Signals
→ Columns
→ Work-in-progress limits
→ A Commitment Point
→ A Delivery point.

⊛ Delivery Pipeline in DevOps:

A DevOps Pipeline is the set of Automated Processes & tools that the development & operations teams use to Compile, Construct, Test & Deploy s/w code faster & Easier.

However, The term "Pipeline" isn't an exact fit, It's more like an assembly.

What are the steps in an Application Delivery Pipeline?

Application Delivery Pipeline that drives the DevOps Process flow.

One way to represent the pipeline is to break it down into six distinct stages —

Commit, Build, Test, Release, Deploy, & Operate.

What are the Components of Delivery Pipeline?

The Safe Continuous delivery pipeline Contains Four Aspects.

* Continuous Exploration
* Continuous Integration
* Continuous Deployment
* Release on demand.

The CDP enables organizations to map their current pipeline into a new structure & then use relentless improvement to deliver value to customers.

* What are the 3 Methods of Application Delivery?

. Application ~~Methods~~ Delivery Methods:

→ N/w Not Required:

After Initial Installation, the Application does not need N/w Access.

→ Application Exists Locally:

The Application files are stored on a drive that is Connected to the Computer.

→ Files Saved Locally:

The Application is designed to save files to local storage.

⊛ What is a Deployment Pipeline in DevOps?

In S/w Development, A Deployment Pipeline is a System of Automated Processes designed to quickly & Accurately move new code additions & updates from version control to Production.

⊛ what are the 6 phases of the DevOps Pipeline?

6 Stages of DevOps CI/CD Pipeline

→ Code Integration (CI)

→ Automated Testing

→ Continuous Integration Server

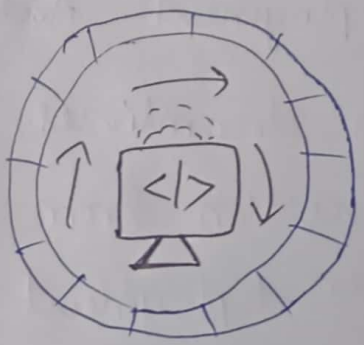
→ Artifact Management

→ Deployment Automation

→ Continuous Monitoring.

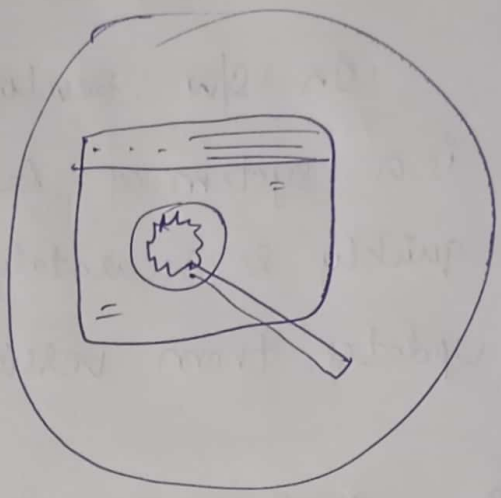
* Code Integration:

DevOps CI/CD Pipeline relies on Seamless Code Integration (CI).



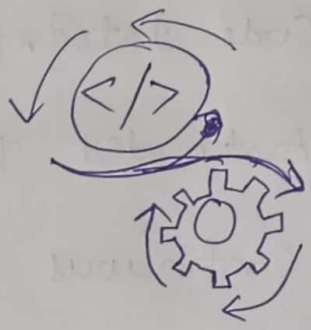
* Automated Testing:

Incorporate Automated Tests throughout SDLC for bug-free, efficient development.



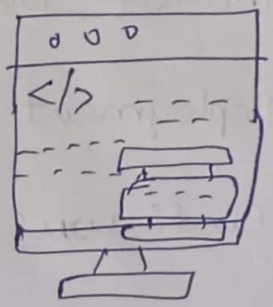
* Continuous Integration Server:

The Continuous Integration Server Orchestrates the DevOps CI/CD Pipeline.



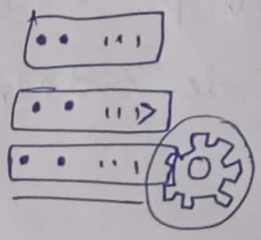
* Artifact Management:

Artifact Mgmt Efficiently handles build outcomes, including code, libraries, & dependencies.



* Deployment Automation:

DA Streamlines Application Transition from development to deployment



* Continuous Monitoring :→

CM is Essential for optimal production performance; set up & Integrate monitoring tools in your pipeline.

* Bottlenecks in DevOps:

"Inefficient Automation" Can act as a Bottleneck in the DevOps Infrastructure by slowing down or obstructing the flow of work.

Automation Issues Can Appear in Several ways

Complex Automatin processes require a lot of Manual Intervention or are difficult to understand & maintain.

Building Effective DevOps Infrastructure (+ How to Avoid them).

The Conventional way of developing & managing S/w across development Organizations has changed significantly.

As a result, Building, Testing, & Deploying S/w is Increasingly flexible. One of these Tenets is DevOps.

DevOps First coined in the Early 2010's, DevOps describes the growing need to unite an organization's development & operations teams to enhance s/w speed, quality, reliability, & release time.

With these qualities, organizations can serve their customers effectively & increase their competitive advantage.

To create an Effective DevOps Infrastructure, we must Quickly Identify & Remediate Bottlenecks.

DevOps Infrastructure refers to Systems & Tools that support s/w Applications development & Operation.

These develop, Test, Deploy, & Observe s/w Applications in the DevOps Environment.

Main Bottlenecks we may Encounter in DevOps Infrastructure & how to Avoid/Overcome those,

1. Unchecked Technical Debt.
2. Inefficient Automation.

3. Complex Automation Processes:

CAP's require a lot of Manual Intervention or are difficult to understand & maintain.

These CAP's slow the workflow, Causing delays in delivering new features or updates to customers.

4. A Lack of Standardization:

In Automation can lead to Inconsistency & Errors, which can cause delays & disruptions.

5. Poorly Designed Automation:

Processes can be challenging to scale & maintain. It leads to Increased downtime & decreased efficiency.

6. A lack of Visibility | Transparency:

In how Automation processes function make it challenging to Identify & fix problems as they arise.

This leads to delays & disruptions.

Q: what is a Bottleneck in Agile?

A Bottleneck is any work stage within a project that stalls & holds up subsequent tasks & dependencies.

Bottlenecks in PM reduce the pace & capacity of the project / workflow.

* what Harm can Bottlenecks Ultimately cause?

Bottlenecks in projects Interrupt the flow of work & hinder progress. They also cause.

- Reduced Efficiency
- Backlogged work
- Long wait times
- High Stress levels
- Reduced Team morale
- Dissatisfied clients
- Loss of Revenue
- Productive Time wasted.

✱ Types of Bottlenecks:

1. Systems - Based Bottlenecks

2. Performer - Based Bottlenecks.

✱ How do we do a Bottleneck Analysis?

Conducting a Successful Bottelneck Analysis requires that we should take a close look at Each step in our project plan or workflow to Identify where the work piles up.

To Achieve this, we should

① Map and Analyze our processes

② Identify the Bottlenecks & their causes.

③ Work out Solutions.

④ Implement Solutions & Evaluate Performance