DEVOPS

B.Tech III Year I-Sem

III-I: CSE								
Course Code	**Category**	**Hours/Weak**			**Credits**	**Max Marks**		
		L	**T**	**P**	**C**	**CIE**	**SEE**	**Total**
23CS503	**Professional Core**	**3**	**0**	**0**	**3**	**40**	**60**	**100**
Contact Classes: 48	**Tutorial classes: Nil**	**Practical classes: Nil**				**Total Classes: 48**		

Pre-requisites:

1. Software Engineering
2. Software Project Management

Course Objectives

- Describe the agile relationship between development and IT operations.
- Understand the skill sets and high-functioning teams involved in DevOps and related methods to reach a continuous delivery capability.
- Implement automated system update and DevOps lifecycle.

Course Outcomes

- Understand the various components of DevOps environment
- Identify Software development models and architectures of DevOps
- Use different project management and integration tools
- Select an appropriate testing tool and deployment model for project

UNIT – I

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

UNIT – II

Software development models and DevOps: DevOps Lifecycle for Business Agility, DevOps, and Continuous Testing.

DevOps influence on Architecture: Introducing software architecture, The monolithic scenario, Architecture rules of thumb, The separation of concerns, Handling database migrations, Microservices, and the data tier, DevOps, architecture, and resilience.

UNIT - III

Introduction to project management: The need for source code control, The history of source code management, Roles and code, source code management system and migrations, Shared authentication, Hosted Git servers, Different Git server implementations, Docker intermission, Gerrit, The pull request model, GitLab.

UNIT - IV

Integrating the system: Build systems, Jenkins build server, Managing build dependencies, Jenkins plugins, and file system layout, The host server, Build slaves, Software on the host, Triggers, Job chaining and build pipelines, Build servers and infrastructure as code, Building by dependency order, Build phases, Alternative build servers, Collating quality measures.

UNIT - V

Testing Tools and Deployment: Various types of testing, Automation of testing Pros and cons, Selenium - Introduction, Selenium features, JavaScript testing, Testing backend integration points, Test-driven development, REPL-driven development

Deployment of the system: Deployment systems, Virtualization stacks, code execution at the client, Puppet master and agents, Ansible, Deployment tools: Chef, Salt Stack and Docker.

TEXT BOOKS:

1. Joakim Verona. Practical Devops, Second Edition. Ingram short title; 2nd edition (2018). ISBN-10: 1788392574

REFERENCE BOOKS:

1. Deepak Gaikwad, Viral Thakkar. DevOps Tools from Practitioner's Viewpoint. Wiley publications.
2. Len Bass, Ingo Weber, Liming Zhu. DevOps: A Software Architect's Perspective. Addison Wesley

DevOps UNIT-1

1. Software Development Life Cycle (SDLC)
2. Agile Model (or) Agile Methodology (or) Agile process, Advantages, Disadvantages, 12 principles of Agile Software development, Difference between Agile and waterfall model.
3. Scrum and Kanban methodologies, Scrum Vs Kanban
4. DevOps, DevOps lifecycle, Best principles and practices of DevOps that improve Collaboration and efficiency in IT Operations.
5. Difference between Agile and DevOps
6. Continuous Integration (CI), Continuous delivery, Continuous Deployment (CD)
7. Common Bottle necks in DevOps CI/CD process & how to overcome them.
8. Role of automation in DevOps, & examples of popular automation tools
9. Artifacts & Artifacts Repository
10. Monitoring and feedback Stage in DevOps
11. Release management in DevOps and its Benefits and challenges
12. Relationship between DevOps and ITIL

Software Development Life Cycle (SDLC)

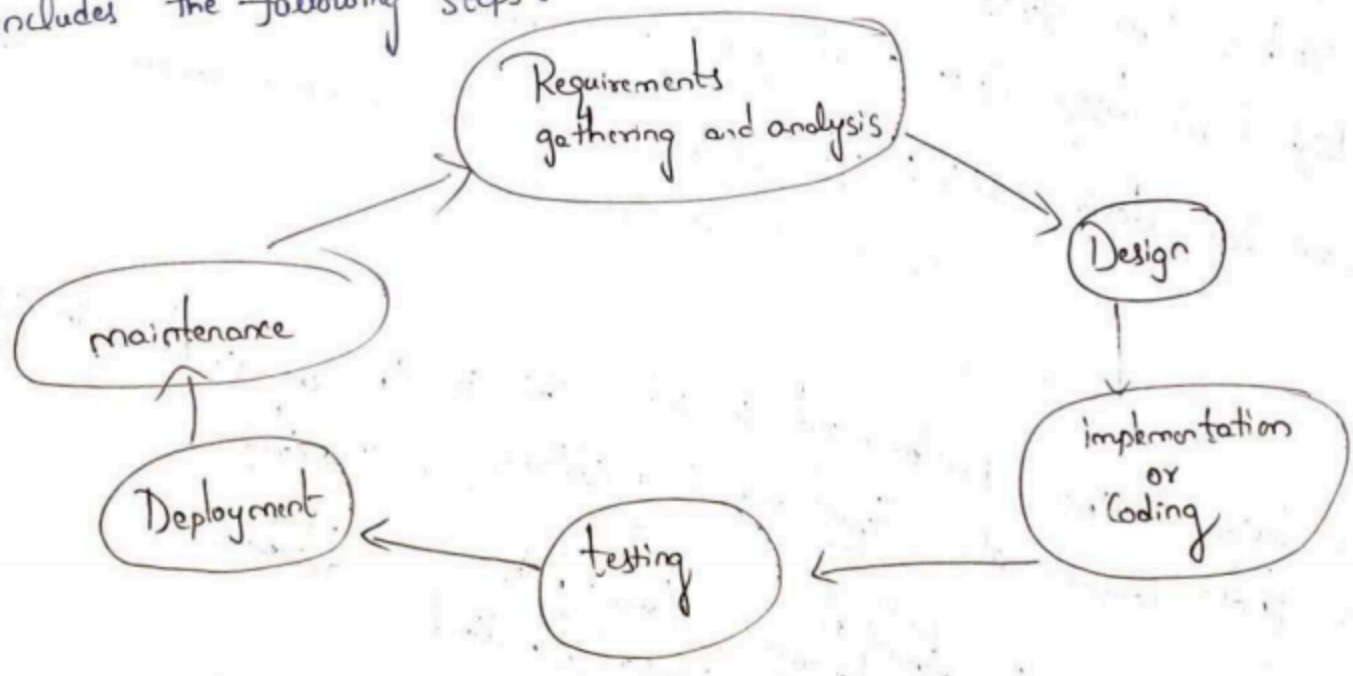
Software Development Life Cycle (SDLC) is a process used by the Software industry to design, develop and test high Quality Softwares. The SDLC aims to produce a high-quality Software that meets or exceeds Customer expectations, reaches Completion within times and Cost estimates.

⇒ It is also called as Software Development process

⇒ without using an exact life Cycle model, the development of Software product would not be in a Systematic and disciplined manner. when a team is developing a Software product, there must be a clear Understanding about what to do and when to do. Otherwise, it would point to project failure. A Software life Cycle model describes entry and exit Criteria for each phase. So without a Software life Cycle model, the entry and exit Criteria for a Stage, Cannot be recognized. without Software life Cycle models, it becomes tough for Software project managers to monitor the progress of the project.

SDLC Cycle :-

SDLC Cycle represents the process of developing Software. SDLC framework includes the following Steps :



(1) Requirements gathering and analysis :-

this phase involves gathering information about the Software requirements from Customers or clients or Stakeholders. Business Analyst or project manager will interact with clients and Collect all Requrements from client and note it in a document Called Business Requirement Specification (BRS).

Once business analyst and his team finalize requrements, they will Create Software Requrement Specification (SRS) document. this SRS document Contains Complete information about project like. Software and hardware Requrements, time requred to Complete this project, Cost of project, how many developers requred, etc.

(2) design :- the primary goal of the design phase is to Create a detailed blueprint or plan for how the Software will be implemented the design documents Created during this phase Serves as a guide for the development team during implementation, helping to ensure that the final product meets the requrements and Objectives of project.

It has two Steps:

High level design (HLD) :- It gives the architecture of Software products

Low Level design (LLD) :- It describes how each and every feature in the Product should work

(3) Implementation or Coding :

the design is then implemented in code, this is the longest phase in SDLC model. this phase Consists of Front end + Middleware + Backend

Front end → development of Coding using various language

Middleware → Connect both frontend & backend

Backend → database

(4) Testing:- the Software is thoroughly tested to ensure that it meet the requirements and works correctly.

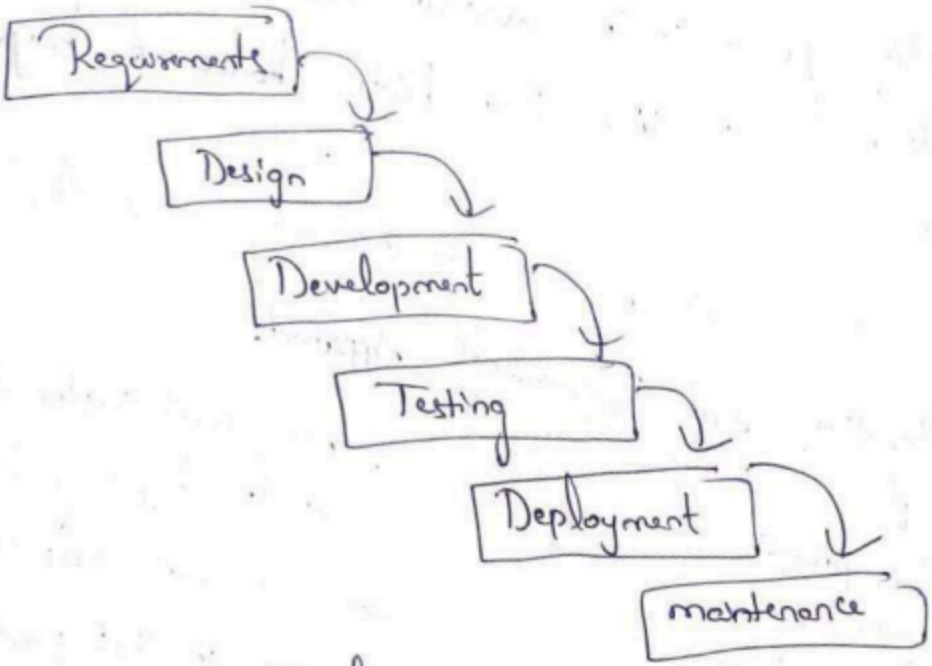
(5) Deployment:- After Successful testing, the Software is deployed to a production environment and made available to end-users.

(6) maintenance:- this phase includes Ongoing Support, bug fixes, updates to the Software.

Difference between Agile and waterfall model :-

waterfall model:-

It is one of the easiest and traditional model to manage. the waterfall model works well in Smaller Size projects where requirements are easily understandable.



waterfall model

The waterfall model is Universally accepted SDLC model. In this method, the whole process of Software development is divided into Various phases. The development in the waterfall model is Seen as flowing Steadily downwards like a waterfall as it is a Continuous Software development model.

Some important points related to the waterfall model are listed as follows -

⇒ waterfall model is not Suitable to develop a large Scale projects.

⇒ The requirements in the waterfall model Should be clear cut at the beginning time; otherwise it may lead to a less effective Software development.

⇒ In the waterfall model, it is hard to move back in order to make changes in the previous phase.

⇒ The testing process in the waterfall model Starts after the Completion of development So, there is a high chance of bugs to be found later. ~~in the~~

Agile Model :- (or) Agile methodology (or) Agile process

It is an iterative and Incremental Approach.

↓

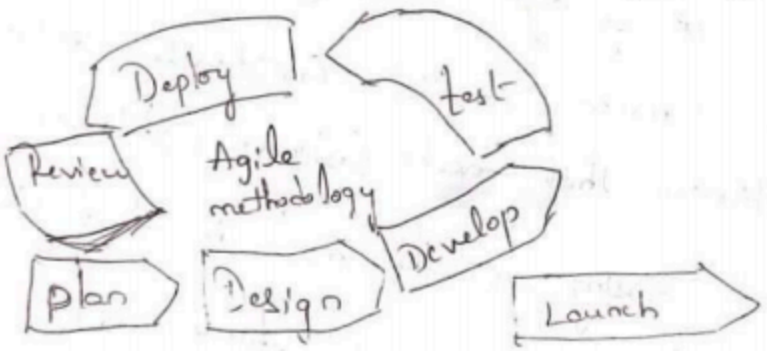
Repeating Some process again and again ie,

Repeating Requirements, design, Coding, testing again and again.

→ we will implement Some features at the beginning in the Software. and we will keep on adding new features on that particular Software.

Agile is an Iterative and Incremental process. In agile process, we will deliver piece of Software with few functionalities. and later on we will keep on adding new functionalities to that same Software. So there is no need of Customer to wait long time until entire Software is developed.

Instead of in-depth planning for project at beginning. Agile methodologies are open to changing requirements over time progresses and expects continuous feedback from the end users.



⇒ Agile is a process that follows Concepts of the Agile manifesto held in February 2001

⇒ the Agile manifesto Contains 12 principles of Agile Software development. they are

(1) highest priority is to Satisfy the Customer through early and Continuous delivery of Valuable Software

(2) Allows Customers to change requirements, even late in development.

(3) Deliver working Software frequently, with in a short period of time,

(4) Business people, developers & testers must work together daily throughout the project.

(5) this Agile principle is about having faith in your team and helping them do their best work. Its like giving them the tools and encourging them to Success, instead of Constantly watching over their shoulders.

(6) face to face Conversation within a team

(7) working Software is the primary measure

(8) Agile processes promote Sustainable development.

(9) technical excellence i.e, quality of the work produced by the development team. It means ~~Constist~~ Consistently delivering high Quality Software that meets Customer expectations.

(10) Simplicity- it means focusing on the most important task and avoiding unnecessary Complexity. So we can achieve goals faster

(11) Self-organizing team:- Instead of having a manager tell everyone exactly what to do, team members decide among themselves how to handle tasks and Solve problem.

(12) Adjusts :- " responding to change over following plan "

Its about being open to adjusting your plans and Strategies based on new information, feedback or changes in the project's requirements.

Advantages :-

⇒ Customer Satisfaction

⇒ Customers, developers, testers Constantly interact with each other

⇒ working Software is delivered frequently (weeks rather than months)

⇒ Face to Face Conversation is the best form of Communication

⇒ close, daily Cooperation between business people and developers.

⇒ Continuous attention to technical excellence and good design

⇒ Even late changes in requirements are welcomed.

Disadvantages :-

⇒ Less Documentation

⇒ Team must be Knowledgeable

⇒ the project can easily get taken off track if Customer is not Satisfied.

Methodologies that are Used to implement Agile :-

1. Extreme programming (XP)
2. Feature - driven development (FDD)
3. Kanban
4. Scrum.

AGILE SCRUM Methodology

Agile Vs Scrum:-

⇒ Agile is basically a process model. whereas Scrum is a kind of framework through which we will develop, test and release Software

⇒ Agile defines principle ie, how process must be. and Scrum is a framework that follows Agile principles

Scrum:-

Scrum is a framework through which we build Software product by following Agile principles.

before Starting, you need to know what is Epic and Story ?

Epic:-

Epic is Description of what client wants or client needs with less functional / no functional details.

Example: client need College website → this is Epic

⇒ Epic in Agile is a big part of work which Can be divided into Smaller User Stories.

Story:- User Stories are Simple and Short, Simple descriptions of a feature told by User / Customer for his product.

Example: User need College website → Epic

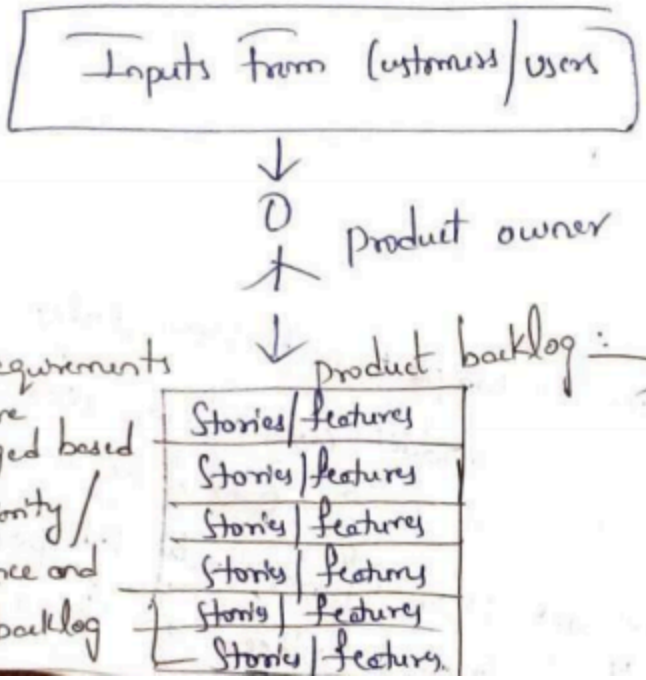
User stories { College website must Contain Student login → Story
faculty login → Story
attendance of Student & faculty → Story

Scrum includes group of people called as Scrum team. Normally Contains 5-9 members

Scrum Team:- 1) product Owner
2) Scrum master
3) Developer
4) Tester

(1) Product Owner :- ~~The~~

Product Owner is the person who always Contact to Customer and Collect features of product from Customers and ~~are~~ he will arrange all those features based on priority or importance and finally he may Accept or reject work results. if product owner is Satisfied with product then product owner will accept work done and if Customer is not Satisfied then he will reject work done by developer and testers.



(2) Scrum master:

⇒ the main role of Scrum master is to handle entire agile process. He will take Care of entire process from Starting to till delivery of project. He will look at Flow of development, managing Coordination between teams and Remove blockers.

⇒ all meetings are Organized by Scrum master.

⇒ Scrum meeting → what did you do yesterday & what will you do today? is Conducted by Scrum master every day, ~~in which~~ in this meeting team members (developers & testers) make Commitments to each other and to the Scrum master. It is a good way for a Scrum master to track the progress of the team.

(3) development team :- they will develop Software

(4) QA team :- they will test Software.

Scrum Terminology

(1) Epic

(2) Story

(3) product backlog.

(4) Sprint :- period of time to Complete each User Story, decided by the product owner and team, Usually 2 to 4 weeks of time.

(5) ~~Sprint meeting (or)~~ Sprint planning meeting & Scrum meeting

(6) Sprint backlog :- among all User Stories, only the particular Selected Story

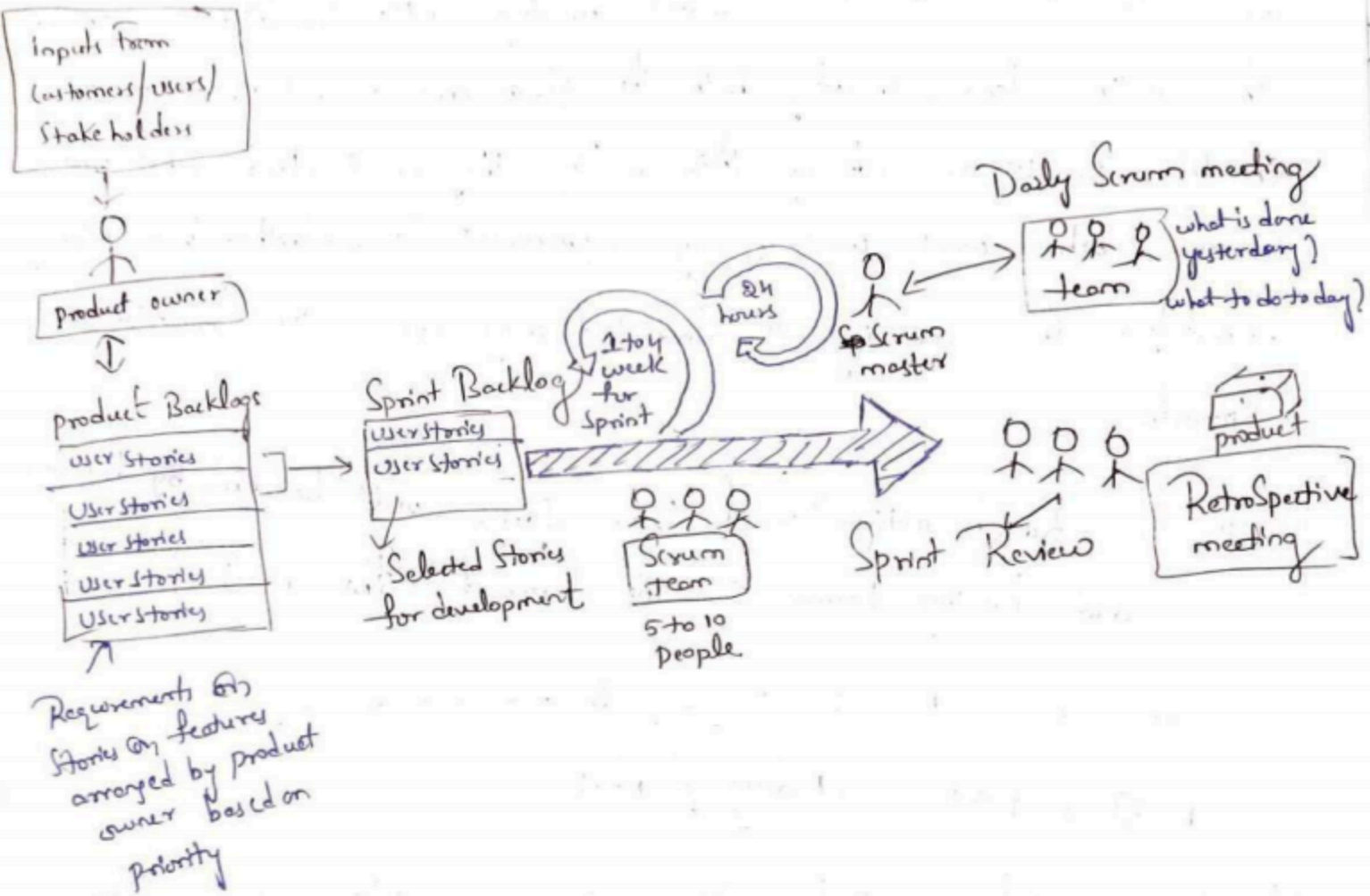
Sprint RetroSpective meeting: Conducts meeting after Completion of Sprint. the entire team, including both the Scrum master and product owner should participate during this meeting they will decide what went wrong? what went well? any improvement to do further? in upcoming Sprint? this meeting is Conducted only once after Completion of Sprint.

Story print:- Rough estimation of User Stories, will be given by development and testing teams in the form of fibonacci Series

0 1 1 2 3 5 8 . . . . . .

1 Story print = 1 day / 1 hrs

Ex: 1 Story print → ~~1 day~~ Dev – 5 (team) testing – 3 (team) → 8 hrs


Inputs from Customers/ users/ Stakeholders
Product owner
Product Backlogs
User Stories
User Stories
User Stories
User Stories
User Stories
User Stories
Requirements or Stories or features arranged by product owner based on priority
Sprint Backlog
User Stories
User Stories
1 to 4 week for Sprint
Selected Stories for development
Scrum team
5 to 10 people
24 hours
Scrum master
Daily Scrum meeting
team
what is done yesterday?
what to do to day?
Sprint
Review
product
RetroSpective meeting

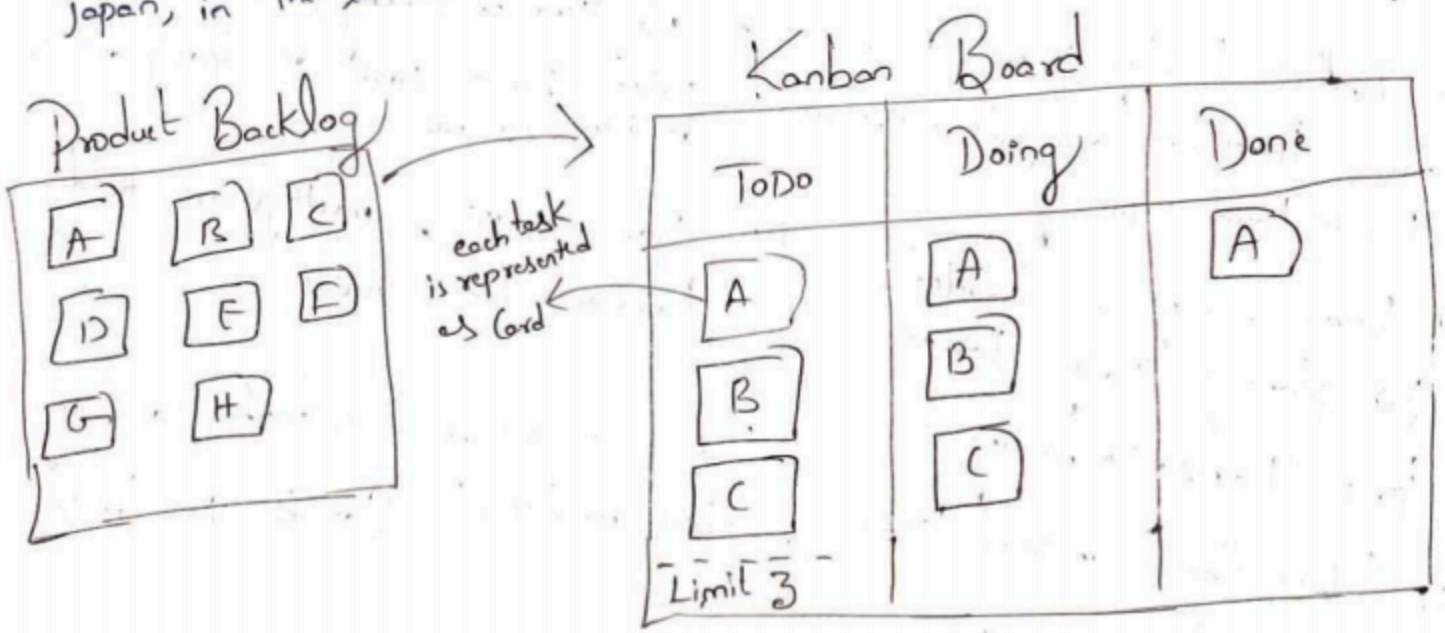
Kanban Methodology :-

⇒ Kanban is a well known Agile methodology. To use the Kanban framework, your team will implement a philosophy of "Continuous improvement", where work items are "pulled" from a product backlog into a steady flow of work. the Kanban framework is applied using Kanban boards.

⇒ this Kanban is next most popular methodology after Scrum methodology

⇒ Kanban boards is a form of "visual project management". In a Kanban board, tasks are represented as Cards and move through Stages of work, represented as Columns. So that your team Can See where work is in real time.

⇒ Kanban was developed by Taiichi ohno, a toyota engineer from japan, in the late 1940's. To improve toyota production System.



Kanban methodology Principles:

(1) Visualize the workflow :- Using Kanban board

(2) Limit work in progress :- only works on few Cards

(3) manage work flow :- focus on Continuous flow of work, identify errors, and optimize process

(4) implement Feedback loops - Conduct feedback & meeting to identify issues and Solve problems

(5) improvement: Accept Continuous Improvement

Scrum	Kanban
(1) work is Organized into fixed length iterations called "sprints"	(1) work is Visualized on a Kanban Board (No fixed iterations)
(2) planning Occurs at the Start of each Sprint, with fixed time	(2) No fixed time, Continuous flow
(3) Product backlog is maintained and Stories are arranged in product backlog based on priority	(3) No fixed backlog; work items are pulled from a pool as Capacity allows ie; items are not
(4) Frequent meeting are Conducted Sprint planning, Daily Standups, Sprint Retrospective meetings.	(4) Rare meetups, meetings are held based on team needs.
(5) Contains Cross-functional teams where members can assume different roles	(5) Roles are more flexible, allowing team members to Specialize in Specific areas.
(6) Contains product manager,	(6) No formal roles

DevOps:

DevOps is a combination of Software development (dev) and operations (Ops). It is a Software development process which aims to integrate the work of development teams and Operations teams by facilitating a Culture of Collaboration and shared responsibility.

Software Development (Dev) :-

this is where the developers write code to create amazing applications.

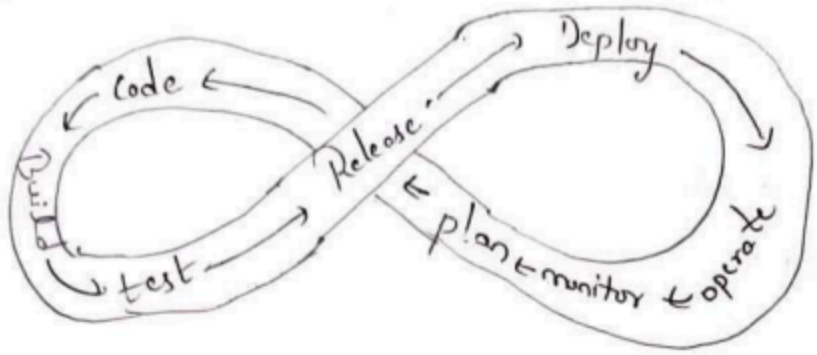
operation (Ops) :-

these employees are like the protectors of the digital world. they take care of deploying, managing and keeping applications running Smoothly in the real world.

⇒ DevOps means people working together to create, build, and deliver Software quickly and Securly. It's all about making the process faster and better by using automation, teamwork, quick feedback and learning from mistakes.

⇒ DevOps is also based on the Agile approach. that means we're looking for ways to improve and do things more efficiently. we learn from each other, work together and take responsibility for making Sure the Software is the best it can be. It's like teamwork in action, making Software better and faster then ever before!.

DevOps lifecycle:



⇒ In the DevOps process, different Stages of the Software development lifecycle (SDLC) can be Categorized into development (Dev) and Operations (Ops) based on their primary focus.

① **plan**:- this Stage primarily falls under development (Dev). It involves gathering requirements, defining features and planning the development process. Developers, project managers, other Stakeholders collaborate to outline the goals and Scope of the project.

Tool:- Jira

② **Example**:- for example, if you want to Create CollegeApplication, you can use 'jira' to Create User Stories and epics for features like Student enrollment, Course management, and you can assign tasks to developers and track progress throughout the development cycle.

(9)

(8)

(2) Code:

the Coding Stage obviously belongs to development (Dev). Developers write and review Code to implement the planned features and functionalities. they work on Creating the Software according to the Specifications defined during the planning phase.

tools:- git, github, gitlab, eclipse

Git:- Git is a Version Control System (it is a Software) works based on Commands, it is a place where our Code is Stored, and everyone in our project Can access it. whenever Someone wants to make a change, they make the changes directly to the Code Stored in Git. Git keeps track of all the changes, including who made each change and when it was made. this way, everyone Knows who made which change, and if Something goes wrong, you Can always look back at the history of changes in Git to See what happened.

GitHub:- it is a web-based platform and Service that host Git repositories in the cloud. Github provides a graphical User interface (GUI) for interacting with Git repositories, making it easier for User to manage their projects.

(3) Build :-

Building the application from Source Code is also a development (Dev) activity. Developers Compile Source Code into executable binaries or artifacts that Can be deployed and Yun. Continuous integration (CI) tools aatomate this process, ensuring that Code changes are Consistently build and tested

tools: jenkins, CircleCI, TravisCI

Example: jenkins is Setup to automate the build process. whenever developers push Code changes to the Grit hub Yepository, jenkins triggers a build job. It Compiles the Source Code, Yuns unit tests, and generates artifacts Such as executable binaries or deployment packages.

(4) Test :-

Testing is primarily a development (Dev) activity, but it Can also involve Collaboration with Operations (Ops) teams. Developers write Unit tests, integration tests, and possibly end-to-end tests to Verify the functionality and quality of the Code.

tools: JUnit, Selenium, postman

Example: Developers write Unit tests Using JUnit to ensure individual Components of ~~College~~ application function Correctly. Testers Use Selenium for automated

browser testing and postman for API testing.

(5) Release:-

Releasing the application to a production environment involves coordination between development (Dev) and Operations (Ops) teams. Developers prepare the application for release, ensuring that its properly packaged and documented. Operations teams may handle tasks like configuring servers and deploying application.

Tool:- Ansible

Example:- Operations teams Use Ansible to automate the deployment process, they defines the necessary configurations and execute application on production Servers.

(6) Deploy:- Deployment is primarily an Operations (Ops) activity. Operations teams are responsible for deploying the application to production Servers or cloud environments. they ensure that the deployment process is smooth, minimize downtime, and follows best practices for reliability and Scalability.

Tools:- Docker, Kubernetes

Example:- For example, my college application is containerized using Docker. operations teams deploy containers to a Kubernetes cluster. ~~Fo~~ which can manages the applications lifecycle. For example, if we deploy new feature it is

Stored as in a Separate Containers on Kubernetes.

(7) Operate:-

Operating the application in production is a core Operations (ops) responsibility, Operations teams monitor the application performance, handle incidents and outages.

Tool:- Grafana

(8) monitor:-

monitoring the applications performance and health is primarily an Operations (Ops) activity. Operations teams set up monitoring tools to track metrics such as Server CPU Usage, memory Usage, response time, error rates. they use this data to detect and respond to issues proactively, ensuring high availability and performance.

Tool:- Nagios

Best principles and practices of DevOps that improve Collaboration and efficiency in IT Operations :-

(1) Automation :-

DevOps teams use tools to do repetitive tasks like testing and building Software, instead of doing them manually. This makes things faster and reduces mistakes

(2) Collaboration and Communication :-

Good DevOps teams work well together and talk a lot. they share ideas and help each other out to get things done Smoothly

(3) Continuous improvement and minimizing waste :-

DevOps teams are always looking for way to make things better. they automate things that are repetitive, and they pay attention to how long it takes to release Software or fix problems So they can make it faster.

(4) Hyperfocus on User needs with short feedback loops

DevOps teams always care a lot about what users want. they use automation and Communication to quickly figure out if user like what they're doing and make changes if needed

(5) Infrastructure as Code (IaC):-

Instead of Setting up Servers and other infrastructure manually, Use Code to do it automatically this makes Sure that all your infrastructure is Set up the Same way every time, which makes things faster and more reliable.

(6) Continuous Integration (CI): whenever Someone makes a change to the Code, put it in a Shared place right away. this makes Sure that everyone's changes are tested together, which helps find and fix problems early.

(7) Continuous Delivery (CD):-

Once the Code changes are tested and ready, automatically Send them to where they need to go, like the production Servers. this means that new features or fixes Can get to Users quickly and Safely.

(8) Security by Design: Make Sure Security is a part of everything we do when making Software. this means putting in Safety measures from the very Start, checking for any Security problems regularly and Using tools to automatically check for Security issues.

(9) Scalability and Resilience:

when we make Software, we design it to handle changes in how many people are ~~Using~~ Using it and to keep working even if Something goes wrong. this means the Software Can grow Smoothly if more people Start Using it, and it won't break down easily if Something Unexpected happens. It's like building a bridge that Can handle heavy traffic and Still Stay Standing even if there's a Strong wind.

(10) monitoring and Logging:

Set up tools that keep an eye on how well our Software and Systems are working. these tools track things like how fast our Software is running and if any parts of it stop working. when Something goes wrong, they Create a record of what happened. this help us find and fix problems Quickly So our Software Stays up and running Smoothly. It's like having a watchful guardian that alerts us if anything goes away, allowing us to fix it before it Causes any trouble.

Difference between Agile and DevOps

DevOps and Agile are the two Software development processes, with Similar aims, getting the end product as quickly and efficiently as possible. while many organizations are hoping to employ these practices, there is Some Confusion between both methodologies.

the key difference between Agile Versus DevOps is that Agile is a philosophy that Contains principles about how to develop and deliver Software, while DevOps describes how to Continuously deploy Code through the Use of modern tools and automated processes.

Agile	DevOps
① Agile was introduced in 2001	① DevOps was introduced in 2001
② In Agile, feedback is coming from the Customer	② in DevOps, feedback Comes from internal team.
③ in Agile. Developing Software is important	③ in DevOps developing, testing, implementation all are equally important.
④ Agile does not focus on automation	④ Automation is the primary goal of DevOps
⑤ it focuses on Constant changes	⑤ it focuses on Constant testing and delivery
⑥ Agile refers to Continuous iterative approach which focuses on Collaboration, Customer feedback, Small and rapid releases	⑥ DevOps is a practice of bringing development and Operation teams together.
⑦ Agile development is managed in Units of Sprints. So this time is much less than a month for each Sprint	⑦ the ideal goal is to deliver the Code to production daily or every few hours
⑧ Used by Small teams of 10 or less	⑧ implemented as Company wide Strategy

Continuous Integration, Continuous delivery, Continuous deployment (CI/CD) :-

① Continuous Integration :-(CI) :-

Developers regularly merge their code changes into a shared repository, triggering automated builds and tests.

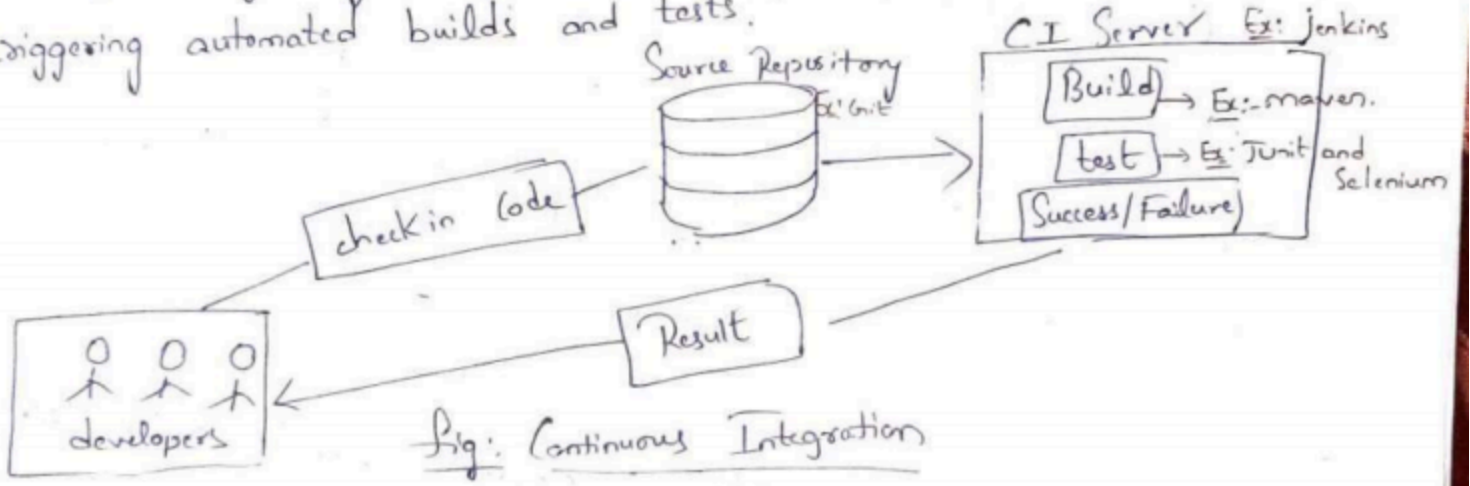


fig: Continuous Integration

Example:-

⇒ Developers write code and push it to a Version Control System like Git. jenkins then monitors these repositories for changes. when changes are detected, jenkins pulls the code, typically using a build tool like maven or Gradle, and then builds the project.

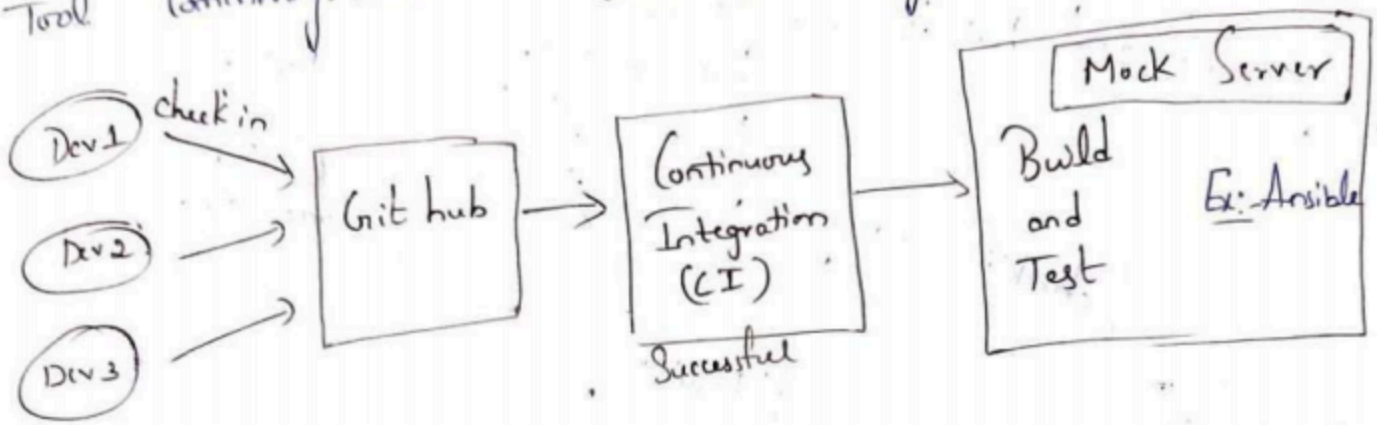
⇒ After building, jenkins may run unit tests using tools like JUnit and integration tests using tools like Selenium.

⇒ the results of the tests are typically reported back to jenkins, which determines whether the build is Successful or failed.

⇒ if the build is Successful, jenkins may proceed to package the application into an artifact (e.g., a JAR, WAR, or Docker image)

(2) Continuous Delivery (CD):-

In Continuous Delivery, the packaged application is automatically prepared for release to production. this can involve tasks like further testing, environment configuration, and final validations before deployment. Tool commonly used in Continuous Delivery is Ansible.



Example:

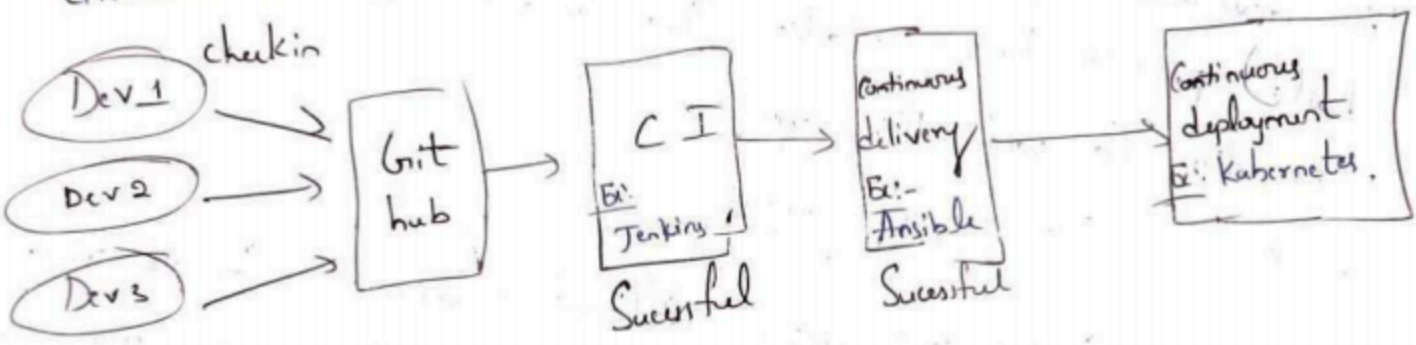
In Continuous Delivery, Ansible can perform automate tasks like configuring the production environment and deploying the application to a Sample Server for further testing and Validation. Once everything is Validated on the Sample Server, the application is then ready for release to the actual production Server. this process ensures that the application undergoes through testing and Validation before being deployed to production Server, reducing the risks of Issues and ensuring a Smoother release process.

(3) Continuous Deployment (CD):

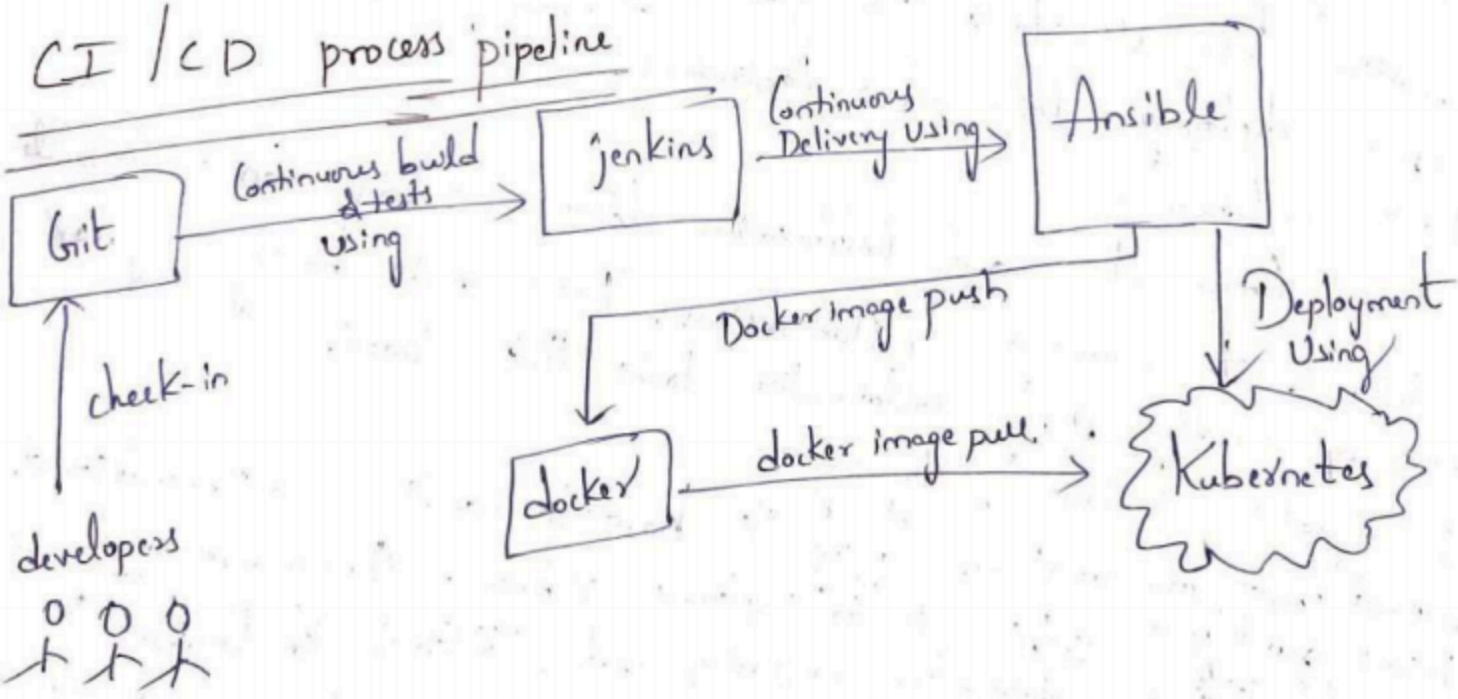
⇒ In Continuous Deployment, every Successful build automatically goes into production without human intervention.

⇒ Tool like Kubernetes is used for automatically deploying and managing application in production Server

Example: After jenkins builds and packages the application, Kubernetes is used to automatically deploy the Docker image to production environments.



CI / CD process pipeline



Common Bottlenecks in DevOps CI/CD & how to overcome them

(1) **Limited Automation:-** If your CI/CD pipeline completely depends on manual steps, then work will be very slow. So Solution is to automate repetitive tasks. For example, instead of manually testing code, use automated testing frameworks like Selenium, tool.

(2) **Slow Feedback loops:-** waiting too long for feedback can delay the development process. So Solution is to optimize tests and build processes to run faster. For instance, parallelizing test execution can speed things up.

(3) **Complex Configuration management:**

Managing configurations across different environments can be tricky. Solution is to use tools like Kubernetes ~~Ansible~~ for configuration management and deploying. Containerized Applications.

(4) **poor Collaboration:**

Lack of communication and collaboration between teams can lead to bottlenecks. Solution is to encourage collaborations. For example, having regular stand up meetings using collaboration tools like slack.

(5) **Inadequate Monitoring & Logging:**

without proper monitoring and logging, its hard to identify and troubleshoot issues quickly. Solution is to implement robust monitoring and logging solutions. Use tools like prometheus for monitoring and ELK Stack for logging.

(6) **Manual Release processes:** manual release processes can introduce errors and delays. Solution is to Automate the release process using CI/CD tools like jenkins, Ansible,

Role of automation in DevOps, & examples of popular automation tools :

the role of automation in DevOps is to automate repetitive tasks through the Software Development life cycle. improve efficiency and ensure consistency

Jenkins :- Automates building, testing, deployment Software

maven :- Automates the build process for java based projects

JUnit :- Automates Unit testing for java based applications

Selenium :- Automates web browser testing

Ansible :- Automates Configuration management, application deployment and task automation

Docker :- Automates the deployment of application inside Containers for Consistent and reliable environments.

Kubernetes :- Automates the deployment & management of Containerized application

Gitlab :- Automates Continuous integration and Continuous deployment (CI/CD) pipeline for Software development

chef :- Automates infrastructure Configuration & management Using Code → Softwares, os, databases, Storage devices, Servers etc.

puppet :- Automates IT infrastructure management

Terraform :- Automates Cloud infra Structure management

Artifacts & Artifacts Repository

In DevOps, artifacts refer to the output or result of software build process. these artifacts can include compiled code, executable files, documentation or any other files generated during build process. Artifacts are typically created as part of CI or CD pipeline and are used for deploying and releasing software.

An artifact repository, also known as a binary repository or artifact management system, is a centralized location where these artifacts are stored and managed. It serves as a secure and reliable storage solution for storing and versioning artifacts throughout the software development life cycle.

Popular Artifact repositories used in DevOps include JFrog Artifactory, & Docker Registry.

Monitoring & Feedback Stage in DevOps:

⇒ the monitoring and Feedback Stage in DevOps is an important step in the Software development process that helps to track the performance and reliability of Software System.

⇒ the use of monitoring tools and logging Systems allows to track the System performance, User behaviour and understand the System behaviour.

⇒ the alerting Systems help to quickly identify and fix issues.

⇒ the data Collected and analysed helps to Continuously improve the Software and the feedback loop allows for a Smooth Collaboration between development and Operations teams. this includes Several Key Activities Such as

⇒

(1) Setting up monitoring tools:- Such as prometheus, Grafana to track System performance, application performance, User behaviour. this includes tracking matrices Such as CPU, memory, response time, error rates.

(2) Logging:-

Setting up logging Systems Such as ELK, Splunk, Loggly to Collect, Store and analyse log data this allows to track the events and Understand the System behaviour.

(3) Alerting :-

Setting up alerting Systems that notify developers when a metric exceeds a Specified threshold, this helps to Quickly identify and fix issues

(4) Analysing data :-

Analysing the data Collected from monitoring and logging tools to identify patterns and trends, this helps to identify and fix issues and improve the Overall performance of the Software.

(5) Feedback loop :-

Creating a feedback loop between development and Operations teams this allows the development teams to quickly address issue and make improvement based on the data and feedback Collected from monitoring and logging tools.

(6) Continuous improvement: Continuously monitor the feedback and improve the Software based on the data Collected, this helps to identify and fix issues and improve Overall performance of Software.

tools for monitoring infrastructure, application Services → Grafana, Nagios, Zabbix, DATADOG, Prometheus

tools for Collecting, processing, Visualizing log data → ELK Stack, ~~Grafana~~, New Relic

Release Management in DevOps

⇒ Release management in DevOps involves Scheduling, planning and Controlling Software Delivery process across development, testing, pre-production and production environments.

⇒ this Release management typically involves Versim Control, Continuous integration / Continuous deployment (CI/CD) pipelines, automated testing, ~~develop~~ deployment automation and monitoring., we use Various tools and processes to automate this process. this help us release updates faster and with fewer mistakes, which is good for the business and the Users.

⇒ In simple terms, release management in DevOps is all about making Sure that when we update or release new Software, everything goes Smoothly. this is done using automation tools and processes.

Here in this Software Development process

Release manager :- Scheduling, planning & Controlling Software Delivery process

Product Owner :- Define requirement for Release

Test / QA manager :- Bug-Free Software

Security Team :- ~~Enfo~~ Ensure No Security issues

DevOps :- Make Deployment Smooth with DevOps tools. i.e most of the things must be automated and that is done by devops team

Release management:

① Coordination: make sure that all the people working on the release, like developers, testers, managers, understand what need to be done and when.

② Complexity:- Dealing with different environments, like on your computer versus on a website and making sure everything fit together smoothly.

③ Risk management:- making sure that the software being released is as free from error or issue as possible, so it doesn't cause problem for users.

④ Time pressure:- Balancing speed and Quality of software

Benefits of Release management:

① Consistency: making sure that everytime we release new software, we follow same steps and rules, which helps to avoid mistakes and confusion.

② Efficiency: making the release process smoother and faster so that dont waste time on unnecessary steps or tasks.

③ Quality Assurance: checking for problems in the software before it's released, helps us fix them early and avoid bigger issues later.

④ Customer Satisfaction: making users happy by releasing good and Quality software in short time.

Realworld Examples:

Netflix
Amazon
Spotify
} automated release management process without disturbing service

Relationship between DevOps and ITIL :-

ITIL Stands for "information Technology Infrastructure Library"

Guidebook for IT :- ITIL is like a handbook that helps manage IT Services. It covers things like fixing problems when they come up, making changes to computer systems, and handling user requests.

Life Cycle :- It's a set of steps that shows how IT Services are planned, built, delivered Overtime. It is like a roadmap for managing IT Services.

DevOps :-

Teamwork for Software :-

DevOps is all about people working together to make Software better and faster. It's like a team sport where everyone has a role to play.

Automation :-

Using tools to make tasks easier and faster. For example, instead of manually doing something, you use a tool to do it automatically.

Keeping Everyone in Sync :-

Making Sure everyone involved in making Software is on the same page and knows what they're doing

How they work together :-

① Similar Goals :- Both ITIL and DevOps want to make sure that technology helps a company reach its goals

② Improving process :-

ITIL gives you a plan for managing IT tasks, and DevOps helps you do them faster and better Overtime.

③ Teamwork and Communication :

DevOps encourages people from different parts of a company to work closely together, which fits with ITIL's idea of everyone being on the same team.

④ making things Easier with Automation :

DevOps focuses on automating repetitive tasks, which can make ITIL's processes smoother and less prone to mistakes.

So, even though they have different focuses, ITIL and DevOps can team up to make sure companies run smoothly and keep their customers happy.