OVER 40 YEARS OF ACADEMIC WISDOM

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

CSE3150 – Front-end Full Stack Development

Department of Computer Science Engineering

School of Engineering

Module I - Syllabus

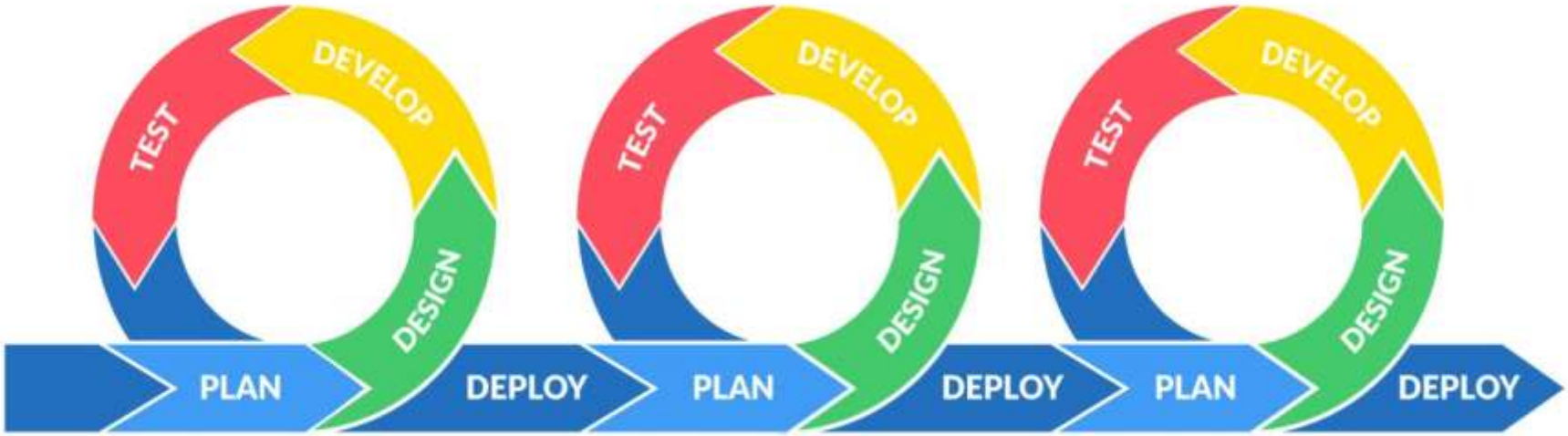
[Lecture-5Hrs, Practical-4Hrs,Knowledge]

Introduction to Agile Methodology; Scrum Fundamentals; Scrum Roles, Artifacts and Rituals; DevOps – Architecture, Lifecycle, Workflow & Principles; DevOps Tools Overview – Jenkins, Docker, Kubernetes. Review of GIT source control.

HTML5 – Syntax, Attributes, Events, Web Forms 2.0, Web Storage, Canvas, Web Sockets; CSS3 – Colors, Gradients, Text, Transform

Introduction to Agile Methodology:

- Agile is a project management approach that emphasizes collaboration, flexibility, and customer satisfaction.
- It values delivering a working product incrementally



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Principles behind the Agile Manifesto

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Pros of Agile Methodology

- Flexibility
- Collaboration
- Customer satisfaction
- Continuous improvement
- Faster delivery

Cons of Agile Methodology

- Implementation challenges
- Lack of structure
- Difficulty with longer-term planning
- Requires a dedicated team
- Can lead to scope creep

Practices Agile

- Daily Standup
- Sprints
- Splitting user stories
- Pair Programming
- Test-driven development

Scrum

1. Scrum is an Agile framework for managing and completing complex projects.

2. It involves a self-organizing and cross-functional team that works to deliver a usable product incrementally.

"Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time."
- Mike Cohn, Mountain Goat Software

Scrum FUNDAMENTALS

1. **Empirical Process Control:** Scrum is based on the principle of empirical process control, which means that it relies on experience and experimentation to continuously improve the process.
2. **Cross-functional Teams:** Scrum teams are cross-functional, meaning they consist of individuals with a range of skills and expertise.
3. **Sprint:** A sprint is a time-boxed period, usually one to four weeks, during which the team works on completing a set of tasks.
4. **Incremental Delivery:** Scrum allows for the delivery of a usable product increment at the end of each sprint.
5. **Transparency:** Scrum emphasizes transparency in all aspects of the process, from the work being done by the team to the progress being made.

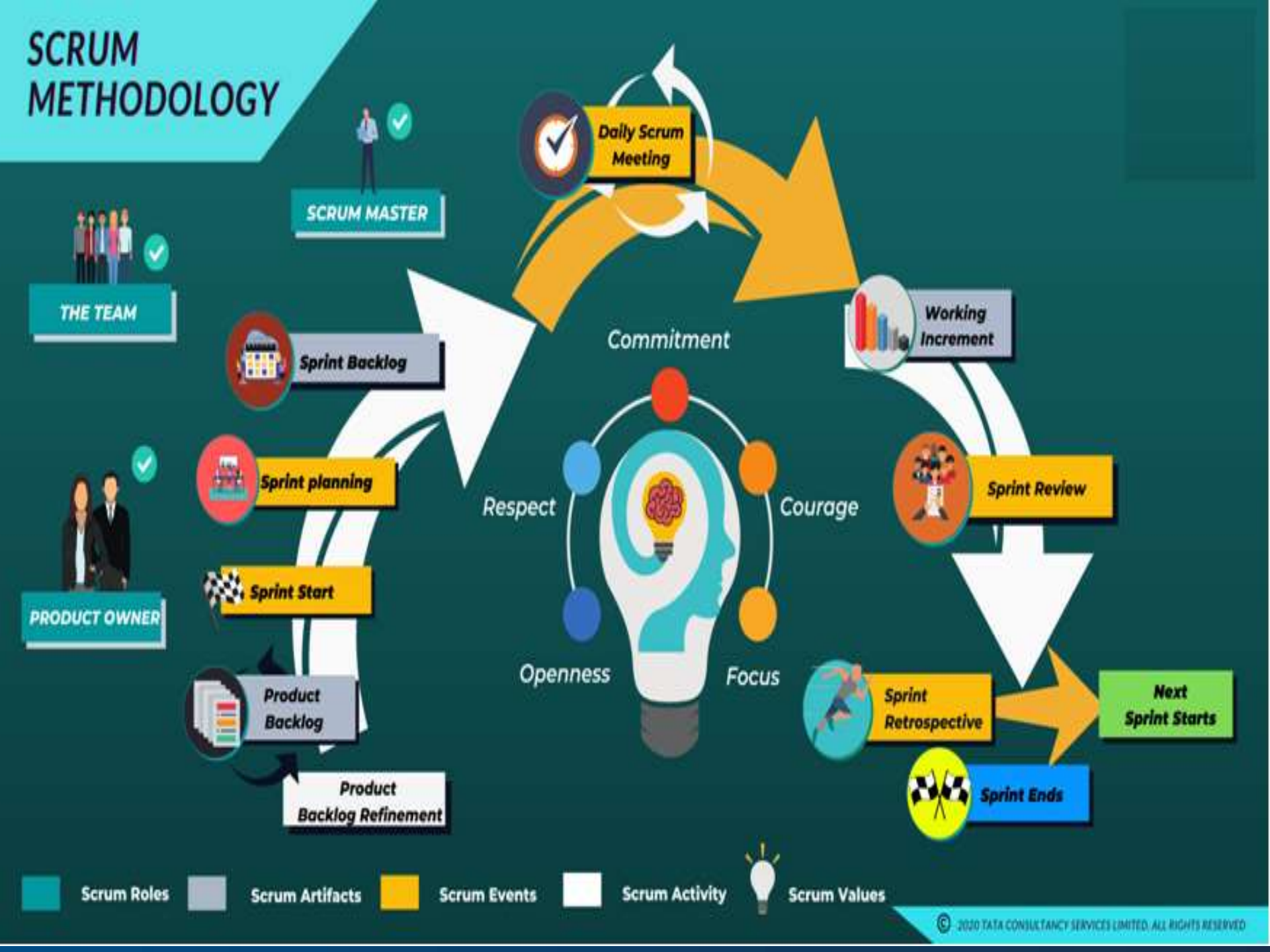
Scrum Methodology

Scrum is simple light weighted agile project management methodology that enables product teams to build products incrementally in an iterative fashion through effective team collaboration.

Scrum forms the base for many of the other frameworks and hence it is important for an agile practitioner to understand this methodology.

The Scrum Methodology is defined by:

- **Scrum Roles**
- **Scrum Events**
- **Scrum Artifacts**


SCRUM METHODOLOGY
SCRUM MASTER
THE TEAM
PRODUCT OWNER
Daily Scrum Meeting
Working Increment
Sprint Review
Sprint Retrospective
Next Sprint Starts
Sprint Ends
Product Backlog Refinement
Product Backlog
Sprint Backlog
Sprint planning
Sprint Start
Commitment
Respect
Courage
Openness
Focus
Scrum Roles
Scrum Artifacts
Scrum Events
Scrum Activity
Scrum Values
© 2020 TATA CONSULTANCY SERVICES LIMITED. ALL RIGHTS RESERVED

What are Scrum Roles?

The scrum team is made up of three roles:

1. A Product Owner
2. A Scrum Master
3. The Development Team.

What are Scrum Roles?

1. Product Owner:

A Product Owner in a scrum team decides what needs to be built.

Unlike traditional delivery, this person is a part of the team that delivers the product.

Following are the key responsibilities of the Product Owner:

- Creates the vision
- Represents business, and is responsible for the ROI
- Cascades the vision to the teams
- Owns the backlog of features
- Prioritizes features by market value
- Is empowered to take decisions
- Negotiates with the team and business to deliver the right product at the right time

What are Scrum Roles?

2. Scrum Master:

The Scrum Master's major responsibility is to ensure that scrum is understood and practiced by every team member in the true spirit.

Following are the key responsibilities of the Scrum Master :

- Is a servant leader
- Helps remove obstacles/impediments
- Facilitates collaboration
- Teaches scrum to the team.
- Protects the teams from external disruptions such as changes to stories in the current sprint.
- Is a change agent in growing the organization to deliver early and often, and removing waste.

What are Scrum Roles?

3. The Development team:

The Development team in scrum is the team that has all the skills necessary to execute the backlog items.

The following are the special characteristics of 'The Development team':

- Self-organizing
- Empowered
- Cross-functional
- Small-sized
- Co-located
- Committed
- Dedicated

What are Scrum Artifacts?

Scrum is an Agile framework for managing software development projects.

It involves a set of practices and principles designed to help teams deliver high-quality products in a fast and efficient manner.

The following are the three main artifacts in Scrum:

1. **Product Backlog**
2. **Sprint Backlog**
3. **Increment**

Scrum Artifacts

1.Product backlog

A product backlog is a dynamic list of functionalities the product might include, such that it provides value to users.

These are a few unique characteristics of a product backlog:

- It is dynamic in nature as it evolves based on changing market needs
- Lists all the features and capabilities that will be taken up in iteration and delivered as a product increment
- It is refined on a continuous basis. The Product Owner and Development team collaborate and update the details, estimate, and prioritize based on business value and size

2. Sprint backlog:

Sprint backlog is a subset of the entire product backlog that the scrum team plans to implement in one iteration or sprint.

Sprint backlog has:

- Subset of product backlog items that the teams commit to implement in one sprint
- Items broken into smaller pieces of work as tasks
- A focus on 'HOW' the team does the work and delivers the value in one sprint
- A story or task board that is used by the teams to view backlog and what the individuals sign up for work after backlog prioritization
- Provision for the Development teams to track the sprint progress and check their alignment with sprint goals

Scrum Artifacts

3. Increment:

- An increment is the work delivered at the end of every sprint. Typically, after every iteration there will be a Product Increment (PI) that delivers value and the final product will be a working software.

- This increment is a sum of all the capabilities that were delivered in the previous sprints as a part of the PI.

- At the end of every sprint, the Product Owner decides whether to release the working product increment or wait until the next release.

Scrum events and rituals

Events: Events are time-boxed activities with specific agendas and outcomes that must be achieved within the time frame.

There are four main events in Scrum:

1. Sprint,
2. Sprint Planning,
3. Daily Scrum, and
4. Sprint Review.

Rituals: Rituals are repeated activities that are performed regularly to reinforce the principles and values of Scrum.

Rituals are less formal and less time-boxed than events and do not have specific agendas or outcomes.

Examples of Scrum rituals include the product backlog refinement, sprint retrospectives, and product demos.

SRUM Rituals

Scrum rituals are regular activities that help reinforce the principles and values of Scrum and improve the development process.

Some of the main Scrum rituals:

Sprint Planning: The team meets to plan the upcoming sprint, review the product backlog, and determine which items can be completed during the sprint. This is a collaborative effort between the development team and the product owner.

Daily Scrum: Also known as the "stand-up," this is a daily meeting where the development team members share updates on their progress and discuss any obstacles they may be facing. The purpose of the Daily Scrum is to increase transparency and accountability within the team.

SRUM Rituals

Sprint Review: The team meets to review the work completed during the sprint and demonstrate the product increments to stakeholders. This is an opportunity for the team to receive feedback and make improvements for the next sprint.

Sprint Retrospective: The team meets to reflect on the past sprint and identify ways to improve the development process for the next sprint. The Sprint Retrospective is an opportunity for the team to continuously improve and adapt to changing circumstances.

Product Backlog Refinement: The product backlog is regularly reviewed and updated to ensure that it accurately reflects the priorities and goals of the project. This helps to ensure that the team is focused on delivering the most valuable features and functionality to the customer.

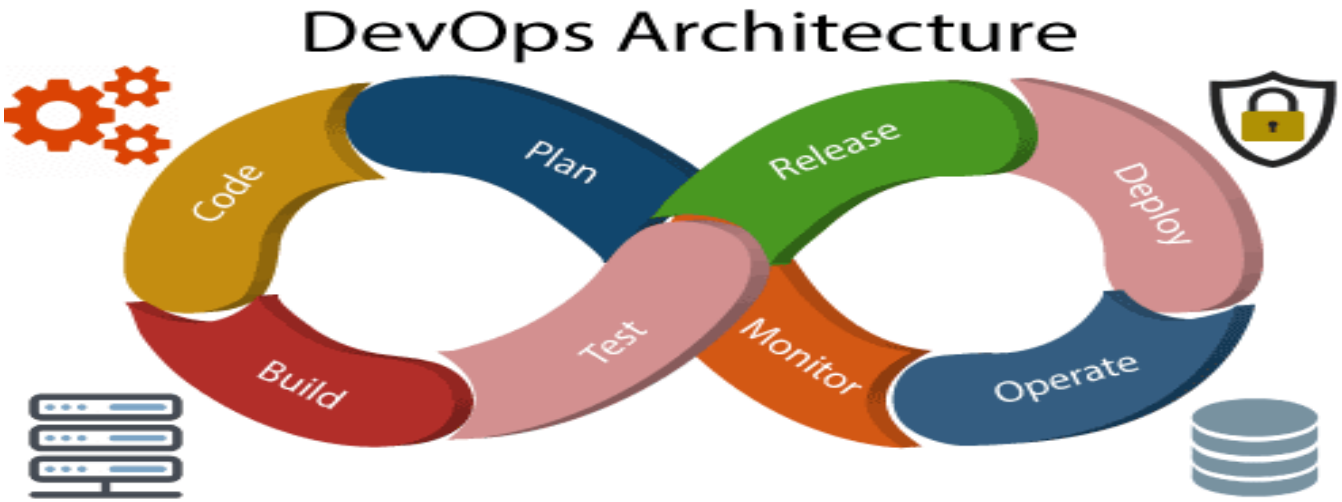
What is DevOps?

The DevOps is a combination of two words, one is **software Development**, and second is **Operations.** This allows a single team to handle the entire application lifecycle, from development to **testing, deployment**, and **operations**. DevOps helps you to reduce the disconnection between software developers, quality assurance (QA) engineers and system administrators.

Various DevOps tools such as Git, Ansible, Docker, Puppet, Jenkins, Chef, Nagios, and Kubernetes.

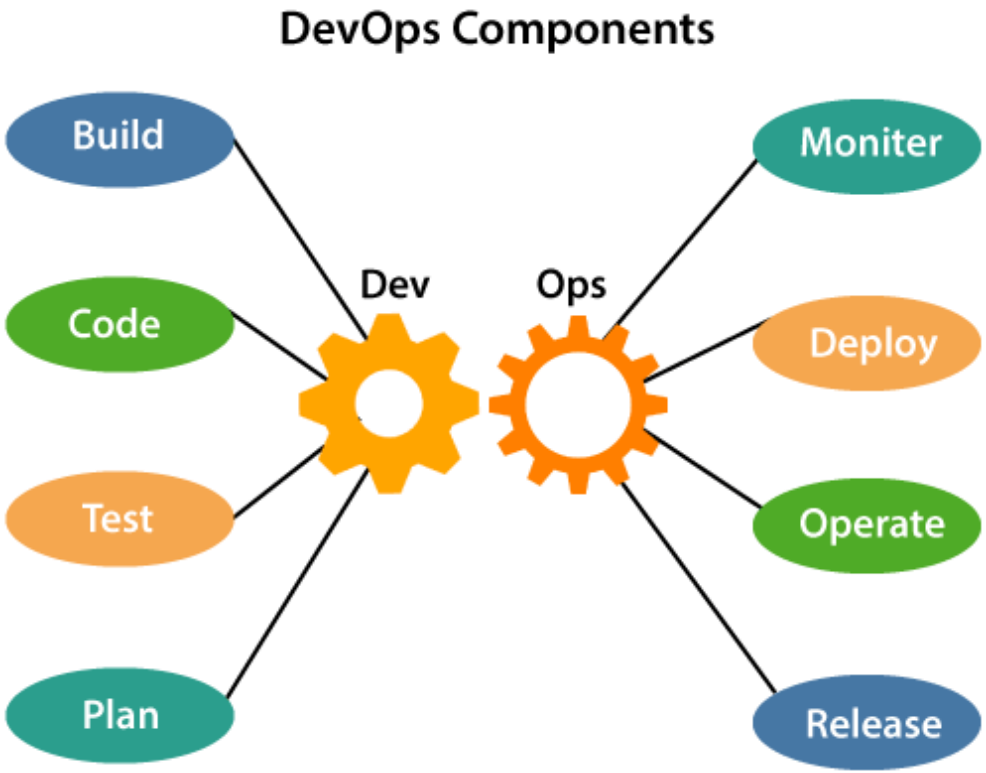
DevOps Architecture

The operation consists of the administrative processes, services, and support for the software. When both the development and operations are combined with collaborating, then the DevOps architecture is the solution to fix the gap between deployment and operation terms; therefore, delivery can be faster.



DevOps Architecture

The various components that are used in the DevOps architecture:



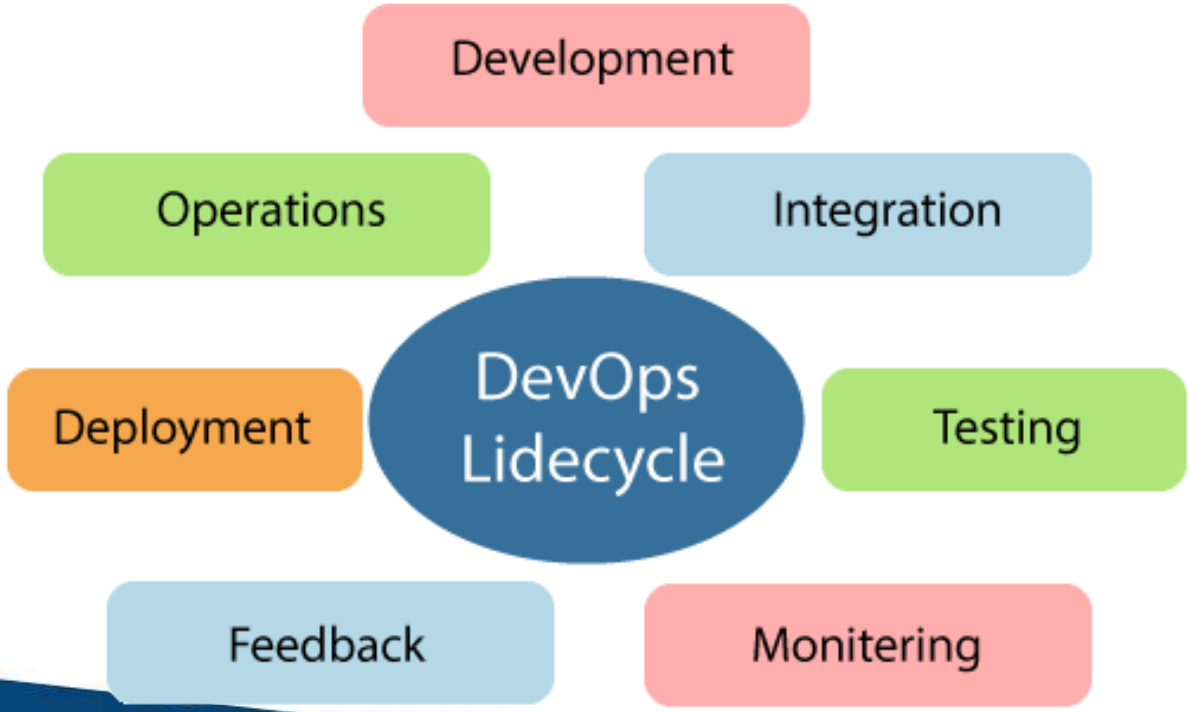
DevOps Architecture

The various **components** that are used in the DevOps architecture:

- **Build:** Cloud and shared resources are used to control resource usage.
- **Code:** Good practices like Git make code use, tracking, and reusability easier.
- **Test:** Automated testing saves time and prepares the app for production.
- **Plan:** Agile methodology helps plan development and improve productivity.
- **Monitor:** Continuous monitoring reduces risk of failure and tracks app health.
- **Deploy:** Automated deployment captures insights and optimizes performance.
- **Operate:** DevOps collaborates throughout the service lifecycle.
- **Release:** Deployment to production is done manually to minimize impact.

DevOps Lifecycle

DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product



DevOps Lifecycle

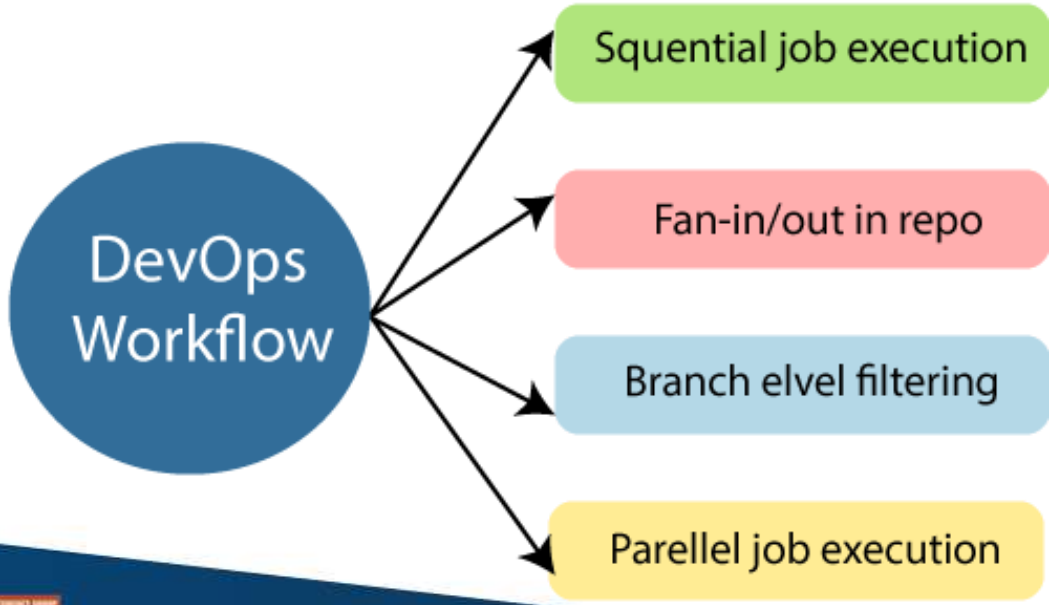
- The DevOps lifecycle consists of **seven phases:** Continuous Development, Continuous Integration, Continuous Testing, Continuous Monitoring, Continuous Feedback, Continuous Deployment, and Continuous Operations.
- Each phase plays a crucial role in the software development process and contributes to the overall efficiency and effectiveness of the DevOps approach.
- **Continuous Development** involves planning and coding the software, where the vision of the project is decided and the developers begin coding the application.
- **Continuous Integration** is the heart of the DevOps lifecycle and involves committing changes to the source code frequently and building the code, including unit testing, integration testing, code review, and packaging.
- **Continuous Testing** involves constantly testing the developed software for bugs using automation testing tools such as TestNG, JUnit, and Selenium.

DevOps Lifecycle contd..

- **Continuous Monitoring** involves monitoring the operational factors of the DevOps process and recording important information about the use of the software.
- **Continuous Feedback** involves analyzing the results from the operations of the software to continuously improve the application development. Continuous Deployment involves deploying the code to the production servers and ensuring that the code is correctly used on all servers. Configuration management tools play a crucial role in executing tasks frequently and quickly in this phase.
- **Continuous Operations** are based on continuity with complete automation of the release process and allow organizations to accelerate the overall time to market. With DevOps, the software product becomes more efficient and increases the overall count of interested customers.

DevOps Workflow

- DevOps workflow provides a visual overview of the sequence in which input is provided. Also, it tells about which one action is performed, and output is generated for an operations process
- DevOps workflow allows the ability to separate and arrange the jobs which are top requested by the users. Also, it gives the ability to mirror their ideal process in the configuration jobs.



DevOps Principles

The main principles of DevOps are Continuous delivery, automation, and fast reaction to the feedback.

- **End to End Responsibility**
- **Continuous Improvement**
- **Automate Everything**
- **Custom Centric Action**
- **Monitor and test everything**
- **Work as one team**

DevOps Tools

Docker

- Docker is a high-end DevOps tool that **allows building, ship, and run distributed applications on multiple systems.** It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

Features

- It configures the system more comfortable and faster.
- It increases productivity.
- It provides containers that are used to run the application in an isolated environment.
- It routes the incoming request for published ports on available nodes to an active container. This feature enables the connection even if there is no task running on the node.
- It allows saving secrets into the swarm itself.

DevOps Tools

Jenkins

Jenkins is a DevOps tool for **monitoring the execution of repeated tasks.** Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly

Features

- Jenkins increases the scale of automation.
- It can easily set up and configure via a web interface.
- It can distribute the tasks across multiple machines, thereby increasing concurrency.
- It supports continuous integration and continuous delivery.
- It offers 400 plugins to support the building and testing any project virtually.
- It requires little maintenance and has a built-in GUI tool for easy updates.

DevOps Tools

Kubernetes

Kubernetes is a powerful and feature-rich platform for **managing containerized applications**

Features

- **Container orchestration:** Kubernetes provides a platform for deploying, scaling, and managing containers, making it easier to manage and scale applications.
- **Self-healing:** Kubernetes has built-in mechanisms for automatically replacing failed containers, ensuring that applications remain available and running even in the face of failures.
- **Automatic scaling:** Kubernetes can automatically scale the number of containers running based on demand, providing a way to ensure that applications can handle increased traffic and load.
- **Load balancing:** Kubernetes provides built-in load balancing capabilities, making it easy to distribute incoming traffic across multiple containers.

DevOps Tools

- **Configuration management:** Kubernetes provides a way to manage the configuration of containers, making it easier to deploy and manage applications at scale.
- **Secret and configuration management:** Kubernetes provides a secure way to manage secrets and configuration data, helping to ensure that sensitive information is kept secure.
- **Resource management:** Kubernetes provides a way to manage the resources required by containers, including CPU, memory, and storage. This helps to ensure that applications get the resources they need to run effectively.
- **Rollouts and rollbacks:** Kubernetes provides a way to easily perform rolling updates and rollbacks of applications, making it easier to manage the deployment of updates.
- **Batch processing:** Kubernetes provides a platform for running batch processing jobs, making it a great choice for organizations that need to process large amounts of data.
- **Service discovery:** Kubernetes provides a way to discover and access services running within the cluster, making it easier to build distributed applications.

What Is Source Control?

Source control, also known as version control, is a system for managing changes to a codebase or other collection of files over time.

It provides a way to track the history of changes made to the code, making it easier to manage and collaborate on software development projects.

Source Control vs. Version Control

These two terms are used interchangeably. However, source control is specific to source code.

Version control also covers large binary files and digital assets.

Source Control Features

Features

- Versioning
- Branching and Merging
- Collaboration
- Conflict resolution
- Continuous integration and delivery
- Auditing
- Security

What Is Source Control Management?

- Source control management (SCM) refers to tools that help you keep track of your code with a complete history of changes.

Source control tools include:

- Git -Git is a popular open-source version control system that allows multiple users to collaborate and manage changes to software projects efficiently. It enables teams to track changes and work together in real-time, making it a crucial tool for DevOps.
- Helix Core
- Subversion
- ClearCase
- Team Foundation Server
- Mercurial

Review of GIT source control.

- **Speed:** Git is designed to be fast, even when working with large projects or repositories. It uses a unique data structure called a "packfile" to efficiently store and retrieve version control data.
- **Easy Collaboration:** Git allows multiple developers to work on the same project simultaneously.
- **Security:** Git provides robust security features, including cryptographic hashing and digital signatures to secure version control data.
- **Large Community:** Git has a large and active community of developers who contribute to the project and provide support.
- Git is a widely used open-source distributed version control system that helps track changes made to software source code. It was created **by Linus Torvalds in 2005** and has since become one of the most popular version control systems in use today.
- Git offers a number of benefits over other version control systems, including:

1. Distributed Architecture.
2. Branching and Merging.


THANK YOU