Devops[Development and Operations]:
CIAEC59
Unit-I

Course Outline

Unit I

- **Introduction: DevOps Overview and Foundations-** What is DevOps, Agile Infrastructure, DevOps Culture, DevOps Engineer, DevOps Lifecycle, DevOps Pipeline, Cloud Computing, Virtualization, Cloud Providers.

Course Outline

Unit II

- **Build Tools: Git-** Git Basics, Repository, Branching / Merging, Git Workflow. **Maven, -** Maven, POM (Project Object Model, **Gradle -**Gradle, Groovy / Kotlin DSL, Build Lifecycle, Dependency Management, Build Automation

Course Outline

Unit III

- **Continuous Integration with Jenkins-** Introduction to Jenkins, Continuous Integration (CI) with Jenkins Pipeline, Freestyle Job ,Source Code Management, Jenkins Plugins, Build Triggers, Post-build Actions, Slave Nodes

Course Outline

Unit IV

- **Configuration Management using Ansible-** Why Configuration Management? Ansible Architecture & Requirements, YAML, Playbooks, Modules, Variables, Handlers, Roles and Reusability, Ansible Galaxy.

Course Outline

Unit V

- **Kubernetes -** Overview,Pods, Replica Sets, Deployments, Namespaces,Services and Networking, Persistent Volumes (PV) and Persistent Volume Claims (PVC), Labels, Selectors, Jobs, and Schedulers, Capstone: Deploy and Manage a Sample App on Kubernetes Cluster **Scaling**: Understanding Scaling, Teams & Tools in DevOps.

Textbooks

1. Algorithm Design, Jon Kleinberg and Eva Tardos, Pearson, 1st Edition 2013.
2. Introduction to the Design & Analysis of Algorithms, Anany Levitin, 3rd Edition, 2012, Pearson education.

Text Books

1. Jennifer Davis and Katherine Daniels, “Effective DevOps”, 1st Edition, Shroff / O’Reilly Publications, 2021. (ISBN-13: 978-9352133765)

2. The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise 1st Edition, by Sanjeev Sharma.

Course Outcomes

At the end of the course, the student will be able to:

1.Understand DevOps principles, culture, and cloud infrastructure (PO 1,2,3,4,5,,6,7,8,12 PSO 1,2,3)

2. Build automation using Git, Maven, and Gradle (PO 1,2,3,4,5,,6,7,8,12 PSO 1,2,3)

3. Configure and manage continuous integration pipelines using Jenkins for automated builds and tests.(PO 1,2,3,4,5,,6,7,8,12 PSO 1,2,3)

4. Automate infrastructure using Ansible playbooks and roles.(PO 1,2,3,4,5,,6,7,8,12 PSO 1,2,3)

5. Deploy and manage applications on Kubernetes with scaling and orchestration. (PO 1,2,3,4,5,,6,7,8,12 PSO 1,2,3)

.

RAMAIAH Institute Of Technology

Course Assessment

DEVOTION TO ENLIGHTENMENT

- CIE- 30 Marks

- Case Study=20 Marks
- Total Marks=CIE+ Assignments=50

Devops

- **DevOps** is a modern way of working in software development in which the **development team** (who writes the code and builds the software) and the **operations team** (which sets up, runs, and manages the software) work together as a single team.
- Before DevOps, the development and operations teams worked separately. This caused:
- Delays in launching software
- Miscommunication between teams
- Slow fixing of problems

Devops

Devops

Why DevOps?

DevOps was created to resolve these issues by making both Development and Operations teams work together in entire software development lifecycle. The following are some reasons why DevOps was needed:

1. Faster Delivery of Software

In traditional development, it takes a long time to move from writing code to delivering it to users. There are many steps, and they are often done manually. DevOps makes this faster by automating tasks like testing and deployment. This means new features, updates, and bug fixes can reach users quickly sometimes even several times a day.

2. Better Teamwork and Communication

In the past, developers (who build the software) and operations teams (who manage it) worked separately. This lead to confusion and delays. DevOps encourages teamwork both teams work together, share knowledge, and take responsibility for the software from start to finish. This reduces mistakes and improves problem-solving.

3. More Reliable Software with Fewer Errors

When software is tested manually and updated rarely, it's easier to miss problems. DevOps uses automated testing and monitoring, which helps catch bugs early before they reach the users. This makes the software more stable, secure, and reliable.

4. Automation Saves Time and Reduces Errors

Manual work takes time and can lead to human mistakes. For example, a small typo during a software update might crash a system. DevOps uses tools that automate tasks like:

Testing code

Releasing updates

Monitoring the system

This saves time, improves accuracy, and makes the whole process smoother.

Devops

5. Helps Businesses Be More Flexible and Competitive

In today’s fast-moving world, businesses need to release features quickly and respond to customer feedback. DevOps supports this by allowing faster changes and quick adjustments, helping companies stay ahead of competitors and adapt to changes easily.

6. Better Experience for Customers

When updates are fast, bugs are rare, and systems are stable, customers are happier. DevOps helps deliver software that works well, gets updated often, and solves user problems quickly, leading to a better overall experience.

Devops

How DevOps Works?

Here is a basic understanding of DevOps working flow:

Code is developed collaboratively by Dev and Ops teams.

Changes are integrated continuously using automated builds and tests (CI).

Applications are deployed automatically through Continuous Delivery (CD).

Infrastructure is managed as code for consistency and repeatability (IaC).

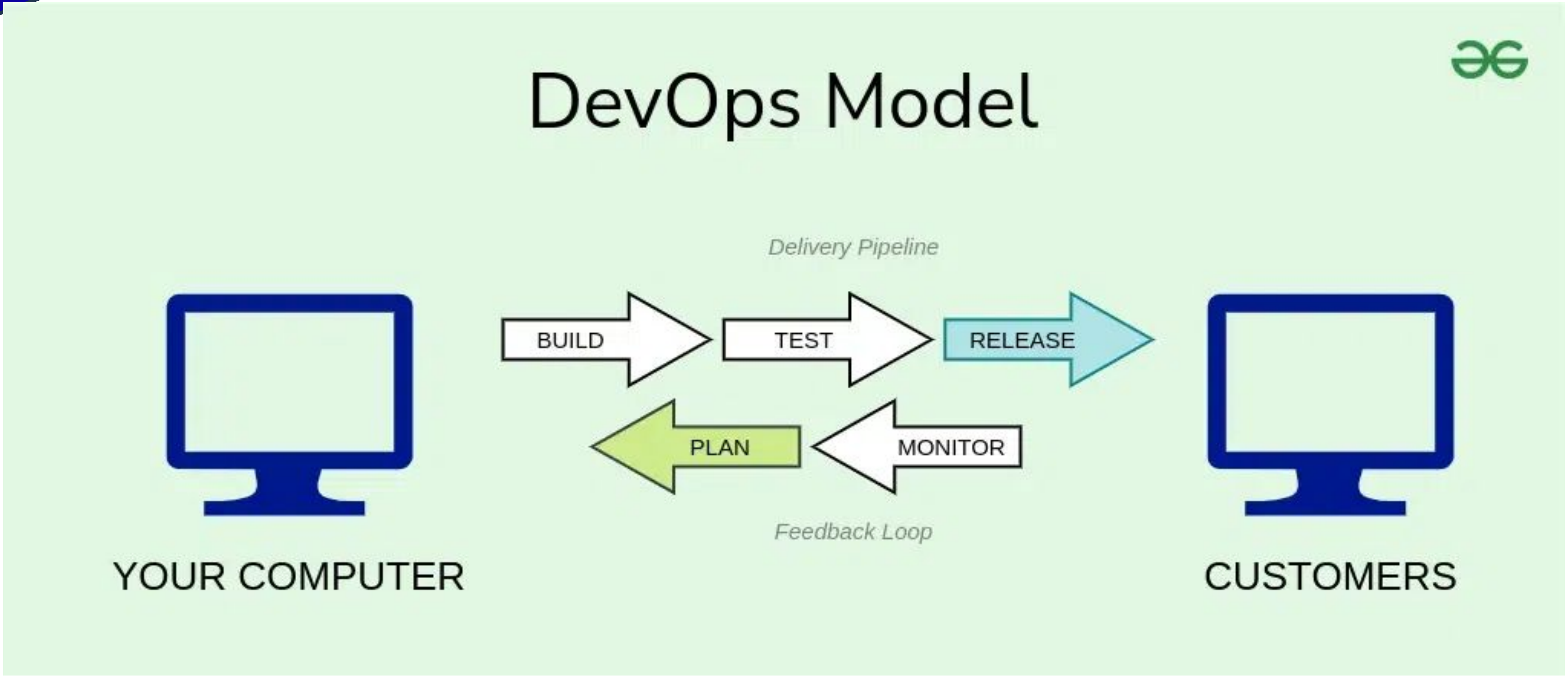
Systems are monitored continuously to gather feedback and improve future releases.

DevOps Model Defined

DevOps is a software development approach that emphasizes collaboration and communication between development (Dev) and operations (Ops) teams. It aims to shorten the software development lifecycle and improve the quality and reliability of software releases.

-

Devops Model



Devops Model

Delivery Pipeline

The pipeline represents the different stages that software goes through before it is released to production. These stages might typically include:

Build Stage

1. Developers **write and organize code**, using version control tools like Git to track changes.

2. The system **automatically compiles and packages the code** into a deployable format.

3. Dependencies (external libraries and tools) are included to ensure smooth operation.

4. Common Tools: Git, Jenkins, GitLab CI/CD, Gradle, Maven.

Test Stage

1. The software undergoes **thorough testing** to catch bugs and security risks before release.

2. Different testing methods include:

Unit Testing: Checks individual pieces of code.

Integration Testing: Ensures different parts of the system work together.

Performance Testing: Measures speed and scalability.

Security Testing: Identifies potential vulnerabilities.

Devops

3. Automated tests help ensure the software is stable before moving forward.

4. **Common Tools:** Selenium, JUnit, TestNG, SonarQube.

Release Stage

1. The software is deployed in a **staging environment** to simulate real-world conditions.

2. If everything checks out, the software is **rolled out to production** using deployment strategies like:

Blue-Green Deployment: Two identical environments switch traffic for a seamless update.

Canary Deployment: A small percentage of users get the new version first, ensuring safety.

Rolling Updates: The update is gradually pushed out to all users.

3. **Common Tools:** Docker, Kubernetes, Ansible, Helm, ArgoCD.

Continuous Feedback Loop

A key aspect of DevOps is **learning from real-world performance** and using that feedback to improve future releases.

1. **Monitoring & Logging:** Track system performance and detect errors.

2. **User Feedback:** Gather insights from customers to enhance features.

3. **Incident Response:** Alert systems notify teams of failures for quick fixes.

4. **Process Improvement:** Teams analyze past releases to optimize automation and workflow.

5. **Common Tools:** Prometheus, Grafana, ELK Stack, Datadog, New Relic.

How to Adopt a DevOps Model?

How to Adopt a DevOps Model?

To adopt a DevOps model, ensure the following points:

Assess Current Workflow: Evaluate your existing development and operations processes to identify gaps, inefficiencies, and areas for automation.

Set Clear DevOps Goals: Define measurable objectives such as faster deployment cycles, better collaboration, or improved system stability.

Build a Collaborative Culture: Break silos between development, operations, QA, and security teams by encouraging communication and shared responsibilities.

Automate Infrastructure and Testing: Use tools like Jenkins, Docker, and Ansible to automate code integration, deployment, testing, and infrastructure provisioning.

Implement CI/CD Pipelines: Establish Continuous Integration and Continuous Deployment pipelines for faster, error-free code delivery.

Monitor and Optimize Continuously: Use real-time monitoring and feedback loops (via tools like Prometheus, Grafana) to track performance and improve systems iteratively.

RAMAIAH Institute Of Technology

How DevOps Can Benefit from AI and ML?

DEVOTION TO ENLIGHTENMENT

How DevOps Can Benefit from AI and ML?

Even though Artificial Intelligence (AI) and Machine Learning (ML) are still growing in DevOps, they are already making a big difference.

Handling Big Data: DevOps tools generate a huge amount of data from testing, deployment, and monitoring. AI and ML are great at reading all this data quickly, finding useful insights, and helping teams make faster and smarter decisions.

Saving Time with Smart Suggestions: AI can learn how developers and operations teams work, then suggest better ways to do tasks or automatically set up the needed tools and servers, reducing manual work.

Spotting Bugs Early: AI and ML can look at code and test results to find problems (like bugs) early. They can detect unusual patterns that may cause issues later and warn the DevOps team before users are affected.

Improving Security: These technologies can scan security logs and alerts to find threats, such as hacking attempts or breaches. Once something risky is found, they can even respond automatically. For example, by blocking access or sending alerts.

DevOps Engineer Job Description

A DevOps Engineer combines software development and IT operations to improve how software is built and deployed. This role involves creating and managing systems that help teams work together more efficiently, ensuring that updates and new features are released quickly and reliably.

Key Responsibilities

Build and Maintain Tools: Create and manage tools that automate software development and deployment processes.

Collaborate with Teams: Work closely with software developers and IT staff to ensure smooth and fast delivery of applications.

Monitor Systems: Keep an eye on system performance and fix any issues that arise to ensure everything runs smoothly.

Improve Processes: Continuously look for ways to make the software development and deployment processes more efficient.

Ensure Security: Implement practices to keep systems secure from potential threats.

Benefits of DevOps

The following are some benefits of DevOps:

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

Improved Collaboration: DevOps promotes collaboration between development and operations teams, resulting in better communication, increased efficiency, and reduced friction.

Improved Quality: DevOps emphasizes automated testing and continuous integration, which helps to catch bugs early in the development process and improve the overall quality of software.

Increased Automation: DevOps enables organizations to automate many manual processes, freeing up time for more strategic work and reducing the risk of human error.

Better Scalability: DevOps enables organizations to quickly and efficiently scale their infrastructure to meet changing demands, improving the ability to respond to business needs.

Increased Customer Satisfaction: DevOps helps organizations to deliver new features and updates more quickly, which can result in increased customer satisfaction and loyalty.

Improved Security: DevOps promotes security best practices, such as continuous testing and monitoring, which can help to reduce the risk of security breaches and improve the overall security of an organization's systems.

Better Resource Utilization: DevOps enables organizations to optimize their use of resources, including hardware, software, and personnel, which can result in cost savings and improved efficiency.

RAMAIAH Institute Of Technology

Challenges While Adopting DevOps

DEVOTION TO ENLIGHTENMENT

The following are some challenges you will face while adopting devops:

High Initial Investment: Implementing DevOps can be a complex and costly process, requiring significant investment in technology, infrastructure, and personnel.

Skills Shortage: Finding qualified DevOps professionals can be a challenge, and organizations may need to invest in training and development programs to build the necessary skills within their teams.

Resistance to Change: Some employees may resist the cultural and organizational changes required for successful DevOps adoption, which can result in resistance, resistance to collaboration, and reduced efficiency.

Lack of Standardization: DevOps is still a relatively new field, and there is a lack of standardization in terms of methodologies, tools, and processes. This can make it difficult for organizations to determine the best approach for their specific needs.

Increased Complexity: DevOps can increase the complexity of software delivery, requiring organizations to manage a larger number of moving parts and integrate multiple systems and tools.

Dependency on Technology: DevOps relies heavily on technology, and organizations may need to invest in a variety of tools and platforms to support the DevOps process.

Need for Continuous Improvement: DevOps requires ongoing improvement and adaptation, as new technologies and best practices emerge. Organizations must be prepared to continuously adapt and evolve their DevOps practices to remain competitive.

•

Devops Life Cycle

The **DevOps lifecycle** is a structured approach that integrates development (Dev) and operations (Ops) teams to streamline software delivery. It focuses on collaboration, automation, and continuous feedback across key phases planning, coding, building, testing, releasing, deploying, operating, and monitoring executed in a continuous loop.

Phases of DevOps Lifecycle

Plan: This phase focuses on understanding the business needs and gathering feedback from end-users. Teams create a plan that aligns the project with business goals and ensures the right results are delivered.

Code: In this phase, developers write the actual code for the software. Tools like Git help manage the code, making sure that the code is well-organized and free from security issues or bad coding practices.

Build: Once the code is written, it is submitted to a central system using tools like Jenkins. This step ensures the code is compiled, and all components are integrated together smoothly.

Test: The software is then tested to ensure it works properly. This includes different types of tests like security, performance, and user acceptance. Tools like JUnit and Selenium are used to automate these tests and verify the software's integrity.

Release: After testing, the software is ready to be released to production. The DevOps team ensures that all checks are passed and then sends the latest version to the production environment.

Deploy: Using Infrastructure-as-Code (IaC) tools like Terraform, the necessary infrastructure (servers, networks, etc.) is automatically created. Once the infrastructure is set up, the code is deployed to various environments in an automated and repeatable way.

Operate: Once deployed, the software is available for users. Tools like Chef help manage the configuration and ongoing deployment of the system to ensure it operates smoothly.

Monitor: This phase involves observing how the software is performing in the real world. Data about user behaviour and application performance is collected to identify any issues or bottlenecks. By monitoring the system, the team can quickly spot and fix problems that may affect performance.

RAMAIAH Institute Of Technology

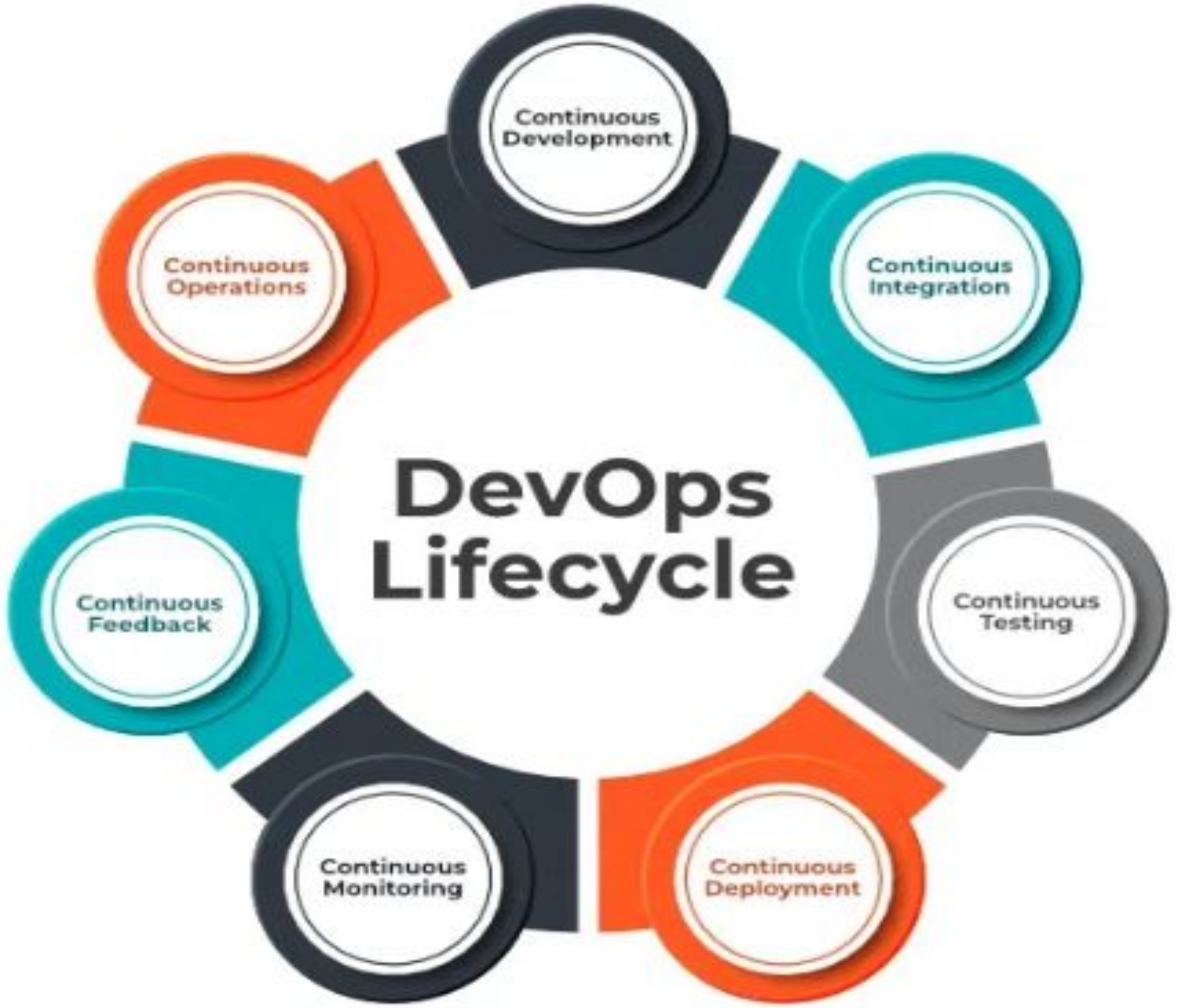
Devops Life Cycle

DEVOTION TO ENLIGHTENMENT

7 Cs of DevOps

The 7 Cs of DevOps are core principles that help make DevOps successful. They guide how teams work together, build, test, and deliver software faster and more reliably. Each of these Cs contributes to a workflow

- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Deployment
- Continuous Monitoring
- Continuous Feedback
- Continuous Operations

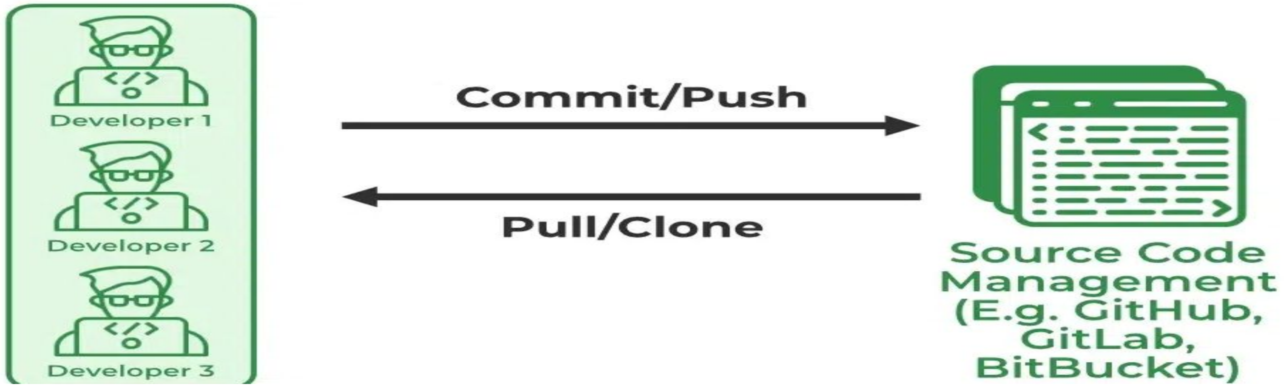


RAMAIAH Institute Of Technology

Devops Life Cycle

DEVOTION TO ENLIGHTENMENT

- **1. Continuous Development**
- Continuous Development involves the iterative and incremental approach to software creation, where development teams plan, code, and prepare software features in small, manageable units. This methodology enables rapid feedback, early detection of issues, and swift delivery of value to end-users. It integrates closely with version control systems and automation tools to streamline the development process.
- **Example:**
- Imagine a team building a food delivery app. Instead of waiting to finish the whole app and testing it later, the team adds features one by one:
- On Monday, they add a "Login" featur
- On Tuesday, they add the "Search for
- Each feature is checked and added to " part, they can fix it right away without affectin



RAMAIAH Institute Of Technology

DEVOTION TO ENLIGHTENMENT

Devops Life Cycle

2. Continuous Integration

Continuous Integration (CI) in DevOps ensures that code changes made by developers are automatically built, tested, and integrated into the main codebase. This process typically involves four key stages:

1. Source Code Management (SCM): Developers push their code from local machines to a remote repository such as GitHub. This allows teams to collaborate, review, and manage code versions easily.

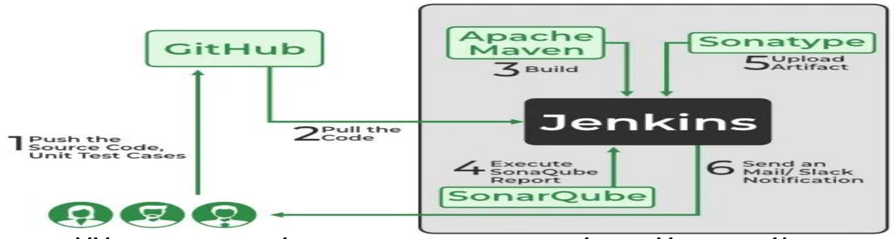
2. Build Process: The source code is then compiled using tools like Maven, which packages the application into artifacts such as .jar, .war, or .ear files.

3. Code Quality Check: Tools like SonarQube analyze the code for bugs, code smells, and security issues. It generates detailed reports (HTML or PDF) to maintain code quality standards.

4. Artifact Repository: The generated build artifacts are stored in a repository manager like Nexus, which serves as a central storage for future deployment.

All these steps are automated using Jenkins, a popular CI tool that orchestrates the complete flow, from fetching code to storing the final build artifact.
Example:
Let's say your team adds a new feature: "Track Delivery Person on Map."

Developer Meena writes code for the tracking feature and pushes it to GitHub.
As soon as the code is pushed, Jenkins picks it up and uses Maven to build the app and
test it using JUnit.The code goes through SonarQube, which finds a few duplicate lines a
final app version (with the new feature) is saved in Nexus as a .jar file.This way, every small change is tested, verified, and saved automatically without manual effort.

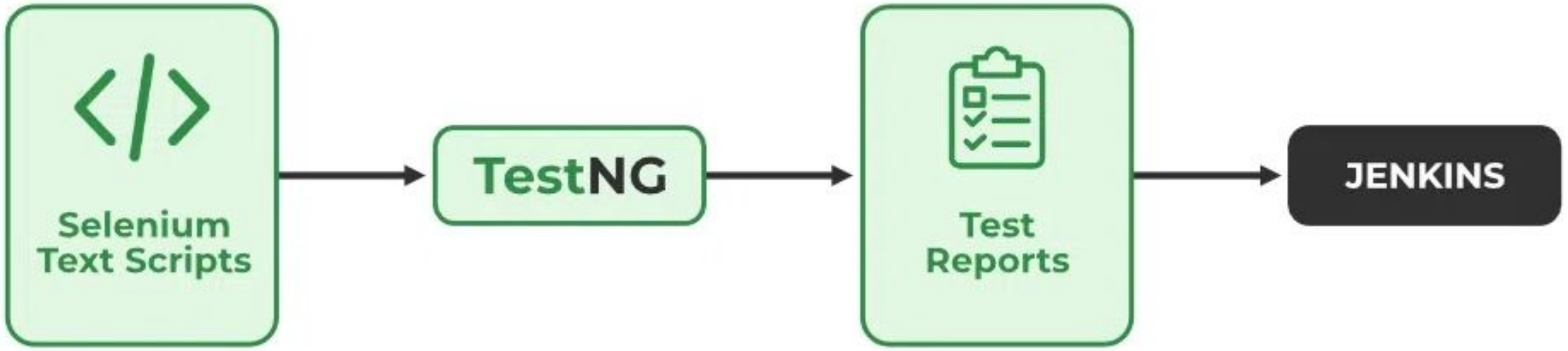

GitHub
Apache Maven
Sonatype
Jenkins
1 Push the Source Code, Unit Test Cases
2 Pull the Code
3 Build
4 Execute SonaQube Report
5 Upload Artifact
6 Send an Mail/ Slack Notification
SonarQube

RAMAIAH Institute Of Technology

Devops Life Cycle

DEVOTION TO ENLIGHTENMENT

- **3. Continuous Testing**
- **Continuous Testing** means testing the code automatically every time there is a change. This helps catch bugs early before the app goes live. With DevOps and Agile methods, companies can use tools like [Selenium](), **Testsigma**, or **LambdaTest** to test their applications automatically. These tools run tests faster and smarter than manual testing.
- Using a tool like **Jenkins**, we can set up the entire testing process to run **automatically** after every code change. This saves time and reduces human errors.
- **Example:**
- Let's say we want to add a new feature: **"Apply Coupon at Checkout"**



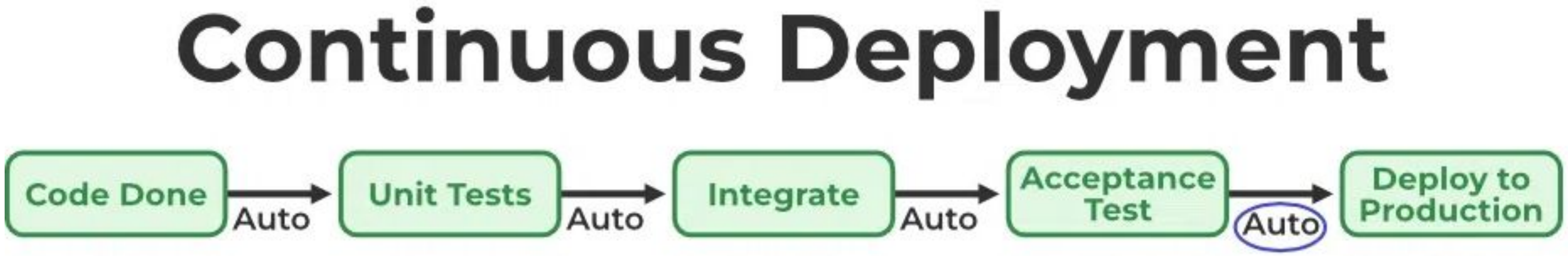
:

- If a problem is found, for example, the app crashes when a wrong coupon is entered the test will **fail**, and the developer will be notified immediately to fix it. This way,

Devops Life Cycle

4. Continuous Deployment/ Continuous Delivery

- **Continuous Deployment:** Continuous Deployment is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.



RAMAIAH Institute Of Technology

Devops Life Cycle

DEVOTION TO ENLIGHTENMENT

Continuous Delivery: Continuous Delivery is the process of deploying an application into production servers manually when it has completed testing and the build stages. Here, we will automate the continuous integration processes, however, manual involvement is still required for deploying it to the production environment.

Example:

Suppose the team adds a “Refer & Earn” feature.

The code is developed, tested, and marked as ready to go live.

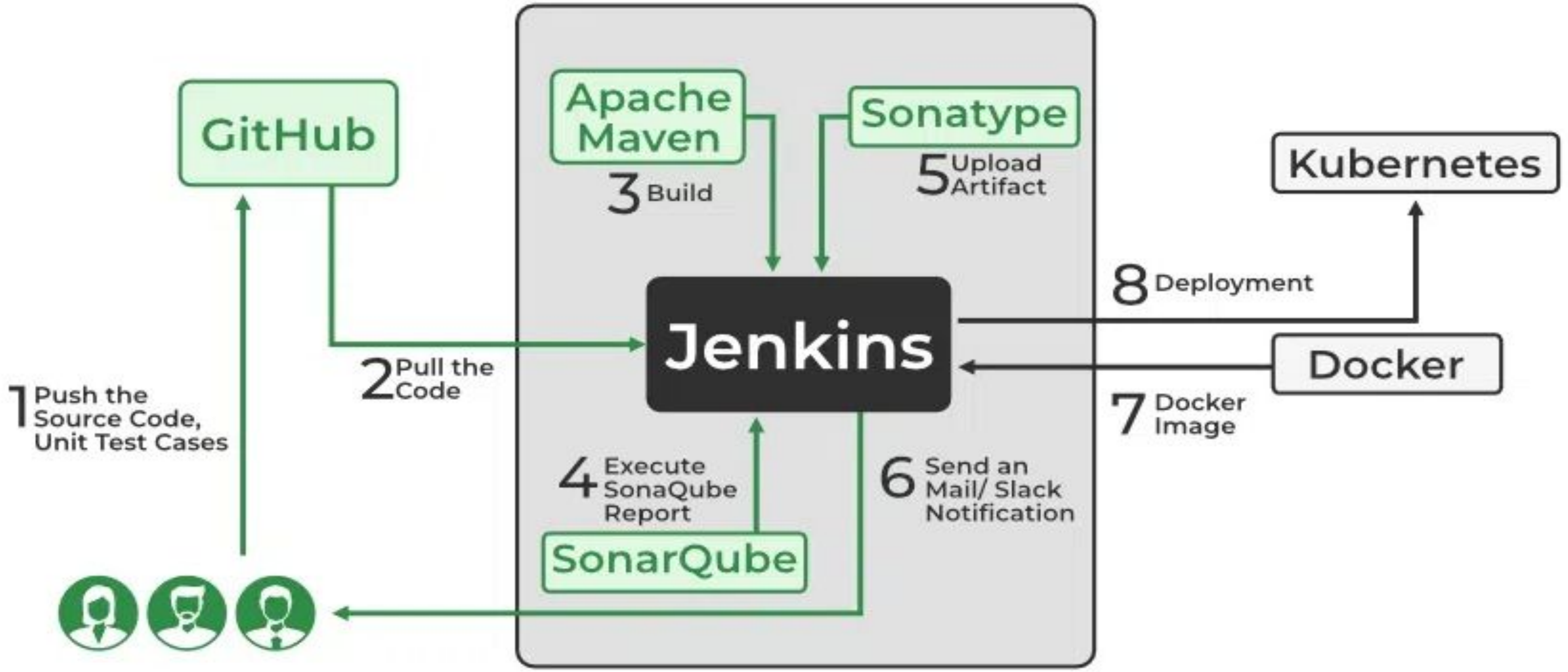
However, the product team decides to launch it during a weekend campaign.

Until then, the feature stays on standby in the staging area.

Once approved, the code is manually deployed to production using a single click.

So, Continuous Delivery ensures every update is deployable anytime, but the actual release can be controlled.

Devops Life Cycle



Devops Life Cycle

5. Continuous Monitoring

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and memory utilization.

Example:

Let's say your app suddenly takes longer to load the "Order History" page.

Prometheus tracks this slow response time and sends an alert to the team.

Grafana shows a graph that spikes during dinner hours when traffic is high.

The team uses this data to adjust the server settings or optimize the code.

This prevents crashes and keeps the app smooth for all users, especially during peak hours like dinner time.

Devops Life Cycle

6. Continuous Feedback

Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

Example:

Suppose users complain that the live delivery tracking is not updating fast enough.

The feedback is collected via app reviews, customer support, or feedback forms.

The DevOps team analyzes the issue and works with developers to improve the tracking speed.

The next update includes a fix, and the user experience improves.

By responding to feedback, the app becomes more reliable and enjoyable to use.

Devops Life Cycle

7. Continuous Operations

We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and better quality control are benefits of continuous operations.

Example:

Let’s say you need to update the payment system.

Instead of shutting the app down, Continuous Operations ensures the update happens in the background.

Users continue ordering food while the change is made without even noticing.

This keeps customers happy and business running 24/7, especially during peak times like lunch and dinner.

RAMAIAH Institute Of Technology

DevOps Pipeline

DEVOTION TO ENLIGHTENMENT

- **DevOps Pipeline**
- **Definition**:
  A DevOps Pipeline is an **automated workflow** that takes source code from development and delivers it into production in a reliable and repeatable way.
- **Stages**:
  - **Source** → Code written & version-controlled (Git).
  - **Build** → Code compiled, packaged (Maven, Gradle).
  - **Test** → Automated testing (JUnit, Selenium, TestNG).
  - **Release** → Prepare artifacts for deployment.
  - **Deploy** → Continuous delivery/deployment to environments (Docker, Kubernetes, Ansible).
  - **Monitor** → Track logs, performance, and user feedback (Prometheus, Grafana).
- **Benefits**:
  ✅ Faster delivery

RAMAIAH Institute Of Technology

Cloud Computing

DEVOTION TO ENLIGHTENMENT

- **Cloud Computing**
- **Definition**:
  Cloud Computing is the **on-demand delivery of IT resources** (servers, storage, databases, networking, software) over the internet (or private cloud).
- **Service Models**:
  - **IaaS (Infrastructure as a Service)** → Virtual machines, storage, networks (e.g., AWS EC2, Azure VMs).
  - **PaaS (Platform as a Service)** → Ready-made platforms for app development (e.g., Google App Engine, Heroku).
  - **SaaS (Software as a Service)** → Ready-to-use software delivered via browser (e.g., Gmail, Office 365).
- **Benefits**:
  ✅ Scalability
  ✅ Cost efficiency (pay as you use)
  ✅ Flexibility and accessibility

RAMAIAH Institute Of Technology

Virtualization

DEVOTION TO ENLIGHTENMENT

- **Virtualization**
- **Definition**:
  Virtualization is the technology that allows **multiple virtual machines (VMs)** to run on a single physical server, sharing its resources.
- **Types**:
  - **Server Virtualization** → Split one physical server into multiple VMs.
  - **Storage Virtualization** → Combine storage devices into a single pool.
  - **Network Virtualization** → Abstract physical network into virtual networks.
- **Hypervisors**:
  Software layer that creates and manages VMs.
  - Type 1: Bare-metal (VMware ESXi, Microsoft Hyper-V).
  - Type 2: Hosted (VirtualBox, VMware Workstation).
- **Relation to Containers**:

RAMAIAH
Institute Of Technology

Cloud Providers

DEVOTION TO ENLIGHTENMENT

- **Cloud Providers**
- **Major Providers**:
  - **Amazon Web Services (AWS)** → Leading cloud provider with services like EC2, S3, Lambda.
  - **Microsoft Azure** → Popular for enterprise solutions, integrates with Microsoft ecosystem.
  - **Google Cloud Platform (GCP)** → Strong in AI, ML, Big Data services.
  - **IBM Cloud, Oracle Cloud, DigitalOcean** → Other notable providers.
- **Selection Factors**:
  - Cost model (pay-as-you-go, reserved instances).
  - Region availability & compliance.
  - Services offered (AI/ML, databases, DevOps tools).

Agile Infrastructure

- **1.Agile Infrastructure**
- **Definition:**
  Agile Infrastructure refers to designing IT infrastructure (servers, networks, storage, cloud environments) in a way that it can **adapt quickly to changing business needs**, similar to how Agile software development adapts to changing requirements.
- **Key Characteristics:**
- **Flexibility:** Infrastructure can scale up or down quickly.
- **Automation:** Uses tools like Ansible, Terraform, or Puppet to provision and manage resources automatically.
- **Rapid Deployment:** Enables faster rollout of applications and services.
- **Resilience:** Quickly recovers from failures or changes in the environment.
- **Collaboration:** Operations teams work closely with development teams.

DevOps Culture

2. DevOps Culture

Definition:
DevOps Culture is the **set of values, practices, and mindsets** that encourages collaboration between development (Dev) and operations (Ops) teams to deliver software faster and more reliably.

Core Principles:

Collaboration: Breaking silos between Dev and Ops.

Automation: Automating repetitive tasks like builds, tests, deployments.

Continuous Improvement: Regularly monitoring, learning, and improving processes.

Shared Responsibility: Both Dev and Ops are responsible for product quality and uptime.

Customer-Centric: Focused on delivering value to end-users quickly.

Benefits:

Faster release cycles

Higher software quality

Reduced downtime

More predictable deployments

Example:
Using **CI/CD pipelines** (Continuous Integration/Continuous Deployment) to automatically test and deploy code whenever it's updated.

DevOps Engineer

3. DevOps Engineer

Definition:
A DevOps Engineer is a professional who **bridges the gap between development and operations** by implementing practices, tools, and infrastructure to support DevOps culture.

Key Responsibilities:

Automate infrastructure provisioning and software deployment.

Implement CI/CD pipelines.

Monitor application performance and uptime.

Collaborate with developers to ensure reliable software delivery.

Maintain cloud infrastructure and configuration management.

Skills Required:

Technical: Scripting, cloud platforms (AWS, Azure, GCP), CI/CD tools (Jenkins, GitLab CI), containerization (Docker, Kubernetes).

Soft Skills: Communication, collaboration, problem-solving, continuous learning.

Example:
A DevOps engineer sets up an automated deployment pipeline using **Jenkins + Docker + Kubernetes**, so new code is automatically tested and deployed to production without manual intervention.

Thank
You