Week 1 session By Fadilulahi Olasunkanmi
Objectives
Understand Devops history and necessity
Learning core DevOps principles

Day 1
Introduction into DevOps & DevOps culture and practices

The Evolution of DevOps: A Historical Journey

DevOps is more than just a buzzword; it's a cultural movement that has transformed how software is developed, delivered, and operated. To appreciate its significance, let's take a journey through time:

1. **Origins and Dysfunction (2007-2008)**:
    - Around 2007-2008, the IT operations and software development communities began to voice their concerns about the industry's dysfunction.
    - Traditional software development models kept developers and IT professionals apart, leading to siloed teams with separate objectives, leadership, and performance metrics.
    - The result? Long hours, botched releases, and unhappy customers.
2. **The Birth of DevOps**:
    - DevOps emerged as a solution to bridge the gap between development and operations.
    - It aimed to break down the barriers that hindered collaboration.
    - Developers and IT/Ops professionals started working together, sharing responsibilities, and aligning their goals.
3. **Key Principles and Practices**:
    - **Collaboration**: DevOps emphasizes teamwork, communication, and shared ownership.
    - **Automation**: Automation streamlines processes, reduces manual errors, and accelerates delivery.
    - **Continuous Integration and Continuous Deployment (CI/CD)**: DevOps promotes frequent, small releases.
    - **Monitoring and Feedback**: Real-time monitoring ensures quick feedback loops for improvements.
    - **Culture Shift**: DevOps fosters a culture of learning, experimentation, and continuous improvement.
4. **Why DevOps?**:
    - **Speed**: DevOps enables faster delivery cycles, reducing time-to-market.
    - **Quality**: Continuous testing and monitoring enhance software quality.
    - **Reliability**: Automation minimizes human error and improves system reliability.
    - **Scalability**: DevOps practices scale seamlessly as organizations grow.
    - **Customer Satisfaction**: Happy customers result from reliable, feature-rich software.
5. **Necessity in a Complex World**:
    - As websites and applications became more complex, the need for DevOps grew organically.
    - Consumer demands, technology advancements, and competitive pressures pushed organizations to adopt DevOps practices.

    - Today, DevOps is not just a methodology; it’s a mindset that drives innovation and agility.

In summary, DevOps is a journey marked by collaboration, innovation, and efficiency

Certainly! Let’s dive into the world of **DevOps** and explore its culture and practices.

DevOps: Bridging the Gap

DevOps is a collaborative approach that unites software development (Dev) and IT operations (Ops). Its primary goal is to streamline the software delivery process, enhance quality, and foster a culture of continuous improvement. Here are the key aspects of DevOps:

1. **Collaboration and Communication**:
    - DevOps breaks down silos between development and operations teams.
    - Developers, system administrators, and testers work together seamlessly.
    - Effective communication ensures shared goals and faster problem-solving.
2. **Automation**:
    - Automation is at the heart of DevOps.
    - It eliminates manual, repetitive tasks, reducing errors and speeding up processes.
    - Continuous integration (CI) and continuous deployment (CD) pipelines automate code builds, testing, and deployment.
3. **Continuous Integration and Continuous Deployment (CI/CD)**:
    - CI ensures that code changes are integrated frequently.
    - Automated tests validate code quality.
    - CD automates the deployment process, allowing rapid and reliable releases.
4. **Infrastructure as Code (IaC)**:
    - IaC treats infrastructure (servers, networks, etc.) as code.
    - Configuration management tools (like Ansible, Puppet, or Terraform) enable consistent, repeatable infrastructure setups.
5. **Monitoring and Feedback**:
    - Real-time monitoring provides insights into system performance.
    - Feedback loops help teams identify issues early and make informed decisions.
6. **Culture Shift**:
    - DevOps is not just about tools; it’s a mindset.
    - Teams embrace a culture of learning, experimentation, and continuous improvement.
    - Failure is seen as an opportunity to learn and evolve.

Why DevOps Matters

1. **Speed and Agility**:
    - DevOps accelerates software delivery.
    - Frequent releases allow organizations to respond swiftly to market demands.
2. **Quality Assurance**:
    - Automated testing ensures consistent quality.
    - Bugs are caught early, reducing post-release issues.
3. **Reliability and Stability**:

    - Automation minimizes human error.
    - Reliable systems lead to better user experiences.
4. **Scalability**:
    - DevOps practices scale effortlessly as applications grow.
    - Cloud technologies facilitate dynamic resource allocation.
5. **Customer Satisfaction**:
    - Happy customers result from reliable, feature-rich software.
    - DevOps aligns development with business goals.

In summary, DevOps is a powerful force that transforms how software is developed, deployed, and maintained. By fostering collaboration, embracing automation, and nurturing a learning culture, organizations can thrive in the ever-evolving tech landscape. 🚀🔧👩‍💻

Day 2
Version control system and Introduction to automated testing

Version Control System (VCS)

Version control, also known as **source control**, is a practice used in software engineering to track and manage changes to code over time. It is a software tool that helps manage changes to source code over time. It allows multiple developers to collaborate on a project, keeping track of every modification to the codebase. Some popular version control systems include Git, Subversion (SVN), and Mercurial.

Key Concepts:

Here are the key points about VCS:

1. **Purpose**:
    - VCS helps software teams manage changes to source code efficiently.
    - It maintains a comprehensive history of modifications, allowing developers to recall specific versions when needed.
2. **Benefits**:
    - **Protection**: Source code is a valuable asset, and VCS safeguards it from mistakes, human errors, and unintended consequences.
    - **Collaboration**: Multiple developers can work on a project simultaneously without conflicts.
    - **Traceability**: Every change is tracked, making it easier to identify issues and revert if necessary.
3. **Types of VCS**:
    - **Distributed VCS (DVCS)**: Git is a popular DVCS that allows developers to work offline, create branches, and merge changes seamlessly.
    - **Centralized VCS (CVCS)**: Examples include Subversion (SVN) and Perforce. These systems rely on a central server for version control.

4. **Workflow Flexibility**:
    - VCS tools should support developers' preferred workflows without imposing restrictions.
    - Great VCS systems facilitate smooth collaboration and avoid file locking.
5. **Application**:
    - VCS is essential for managing software projects, tracking changes, and ensuring code integrity.

Things to note when using Verion control System

1. **Repository**:
   This is where all the project files and their revision history are stored. Repositories can be either centralized (like SVN) or distributed (like Git).
2. **Commit**:
   A commit is a snapshot of changes made to the codebase at a particular time. It typically includes a commit message explaining the changes.
3. **Branching and Merging:**
   Branching allows developers to create separate lines of development. Merging integrates changes from one branch into another.
4. **Pull Requests (PR):**
   In Git-based systems, a pull request is a way to propose changes to a repository. It allows others to review the changes before they are merged into the main branch.
5. **Conflict Resolution:**
   When multiple developers make changes to the same file, conflicts may arise during merging. Version control systems provide tools to resolve these conflicts.

Automated Testing

Automated testing involves using tools and scripts to execute test cases automatically.It involves using software tools to run tests on the codebase automatically. These tests verify whether the software behaves as expected, helps catch bugs early in the development process, and ensures that changes don't introduce regressions.

Here's what you need to know:

1. **Purpose**:
    - Manual testing is repetitive and time-consuming. Automation speeds up the process.
    - It's practical, cost-effective, and provides quicker results.
2. **Benefits**:
    - **Cost Reduction**: Automated testing can reduce hourly fees significantly.
    - **Speed and Accuracy**: Automation provides faster and more accurate results than manual testing.
    - **Increased Test Coverage**: Detect bugs and vulnerabilities early.
3. **Criteria for Automation**:

    - Automate tests that are repetitive, time-consuming, or vulnerable to human error.
    - High-risk functions, cross-browser tests, and tests on multiple configurations should be automated.
4. **Automation Tools**:
    - **Record/Playback**: Tools like UFT or Selenium allow recording user actions and playing them back automatically.
    - **Browser Automation**: Selenium-based tools (e.g., Sauce Labs, BrowserStack) run tests on installed browsers remotely.

Types of Automated tests

1. **Unit Tests:**

These test individual components or units of code in isolation. They are typically small, fast, and focus on a specific function or method.

2. **Integration Tests:**

These test how different components work together. They ensure that interactions between various parts of the system function correctly.

3. **Functional Tests:**

These test the functionality of the software as a whole, simulating user interactions with the system.

4. **Regression Tests:**

These tests verify that changes to the codebase haven't introduced new bugs or broken existing functionality.

Other Benefits of Automated Testing:

- Faster Feedback:
  Automated tests provide rapid feedback to developers, enabling them to catch and fix issues early in the development cycle.
- Consistency:
  Automated tests ensure that the same tests are run consistently across different environments, reducing the likelihood of human error.
- Regression Detection:
  Automated tests help detect regressions caused by new changes, preventing the reintroduction of previously fixed bugs.
- Facilitates Continuous Integration:
  Automated testing is a crucial component of continuous integration (CI) pipelines, where changes are automatically tested and integrated into the codebase.

By combining version control systems with automated testing practices, development teams can collaborate more effectively, maintain code quality, and deliver higher-quality software more efficiently.

Day 3

Hands on workshop on Git CI pipeline ad Github action

For the hands on lab on GIT CI the repository on GIThub will be used

https://github.com/TechFems-projects-and-resources/github-actions-workshop