WEEK 2: PROJECT PLANNING, SWE PRACTICES, & DEVOPS

UML CLOUD COMPUTING CLUB

UMASS LOWELL

ANNOUNCEMENTS

- New UniPath.io Project Repository!
- Meeting Restructure: Every meeting will be from 6:30pm-9:30pm.
  - Presentation Section (6:30pm-8:00pm): This portion is reserved for presentations, demos, and sometimes special guest speakers. It is more structured and follows the weekly topics closely.
  - Hands-On Section (8:00pm-9:30pm): This time is allocated for working on the semester project. It is more interactive, free-flowing, and unplanned, allowing for hands-on experience and collaboration.
- We are in the process of getting better rooms
- Guest speakers to be announced soon!

AGENDA & OBJECTIVES

- Introduction
- Project Planning
- Software Engineering Practices
- Introduction to DevOps
- Hands-On Activity

- Understand the fundamentals and importance of Project Planning
- Gain insights into Software Engineering Practices
- Learn the basics and significance of DevOps

PROJECT PLANNING

- Introduction to Agile methodologies like Scrum and Kanban
- Importance of effective project planning in software development
- Role of user stories and epics in defining project scope
- Brainstorm Project Requirements and create a sample backlog

UMASS LOWELL

PROJECT PLANNING: INTRODUCTION TO AGILE METHODOLOGIES

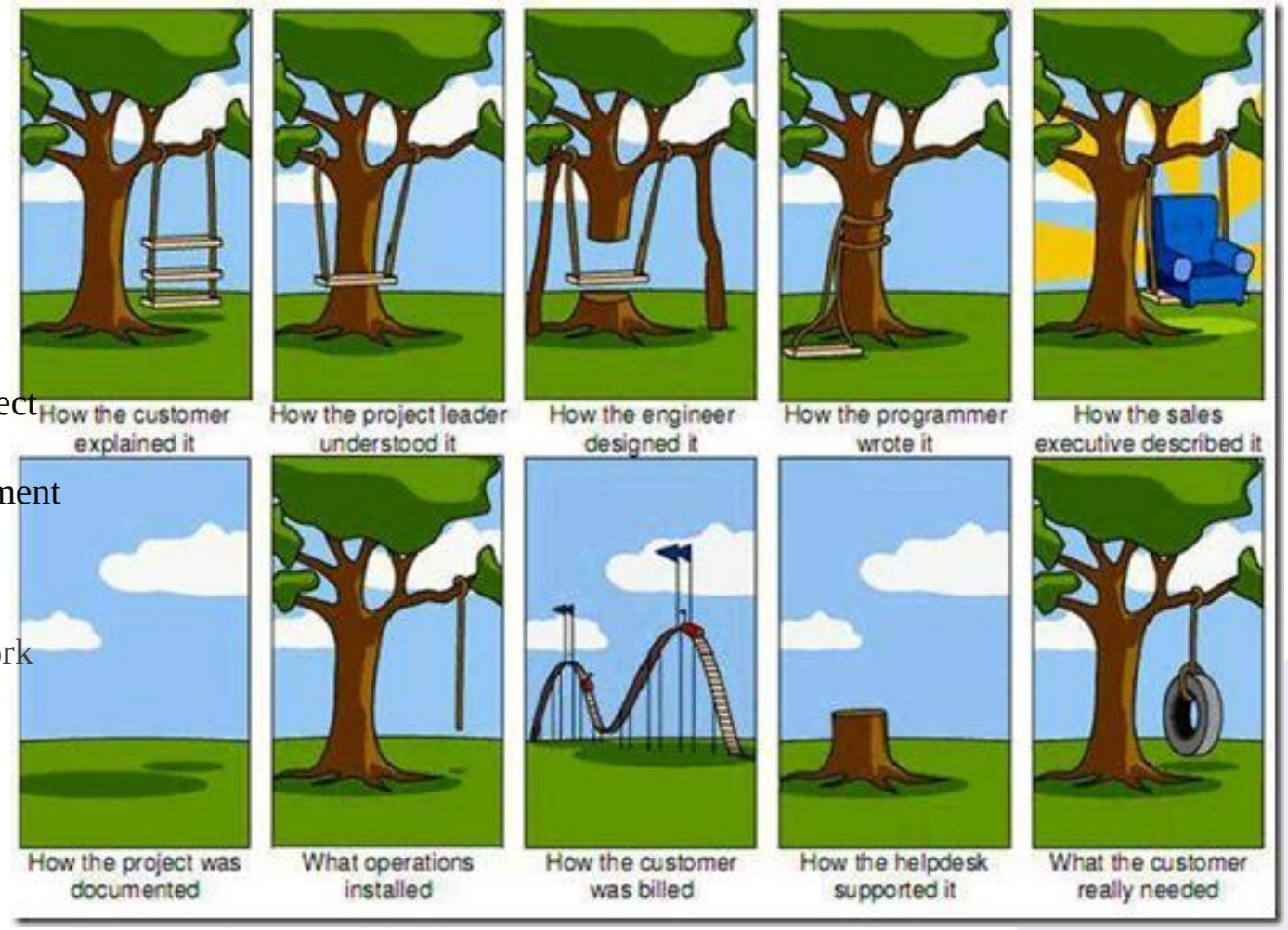
HTTPS://WWW.YOUTUBE.COM/WATCH?V=8EVXTYIZ1HS

PROJECT PLANNING: INTRODUCTION TO AGILE METHODOLOGIES

HTTPS://WWW.YOUTUBE.COM/WATCH?V=BCID33TGQ8A

PROJECT PLANNING: IMPORTANCE OF EFFECTIVE PROJECT PLANNING IN SOFTWARE DEVELOPMENT

- **Risk Mitigation**
  - Early identification of potential issues
  - Allows for contingency planning
- **Resource Allocation**
  - Ensures optimal use of time, money, and manpower
  - Helps avoid bottlenecks and delays
- **Clear Objectives**
  - Sets measurable goals and KPIs
  - Provides a roadmap for the team
- **Stakeholder Alignment**
  - Ensures all stakeholders have a clear understanding of project scope and objectives
  - Facilitates better communication and expectations management
- **Quality Assurance**
  - Allows time for adequate testing and quality checks
  - Reduces the likelihood of costly late-stage changes or rework
- **Timeline Management**
  - Helps in setting realistic deadlines
  - Allows for tracking progress and making adjustments as needed
- **Cost Efficiency**
  - Better planning can lead to cost savings through optimized resource usage
  - Helps avoid scope creep that can inflate budgets



PROJECT PLANNING: ROLE OF USER STORIES AND EPICS IN DEFINING PROJECT SCOPE

- What are **User Stories**?
  - Short, simple descriptions of a feature from the perspective of the user
  - Format: "**As a [user type], I want [an action] so that [benefit/value].“**
    - “As a **student**, I want **to be able to create a long-term degree plan** so that **I can see what classes I need to take each semester**”
- What are **Epics**?
  - Large bodies of work that can be broken down into smaller tasks or stories
  - Often span multiple sprints or even releases
- Defining **Scope**
  - User stories and epics help in outlining the scope of the project
  - Provide a structured way to capture requirements

PROJECT PLANNING: ROLE OF USER STORIES AND EPICS IN DEFINING PROJECT SCOPE

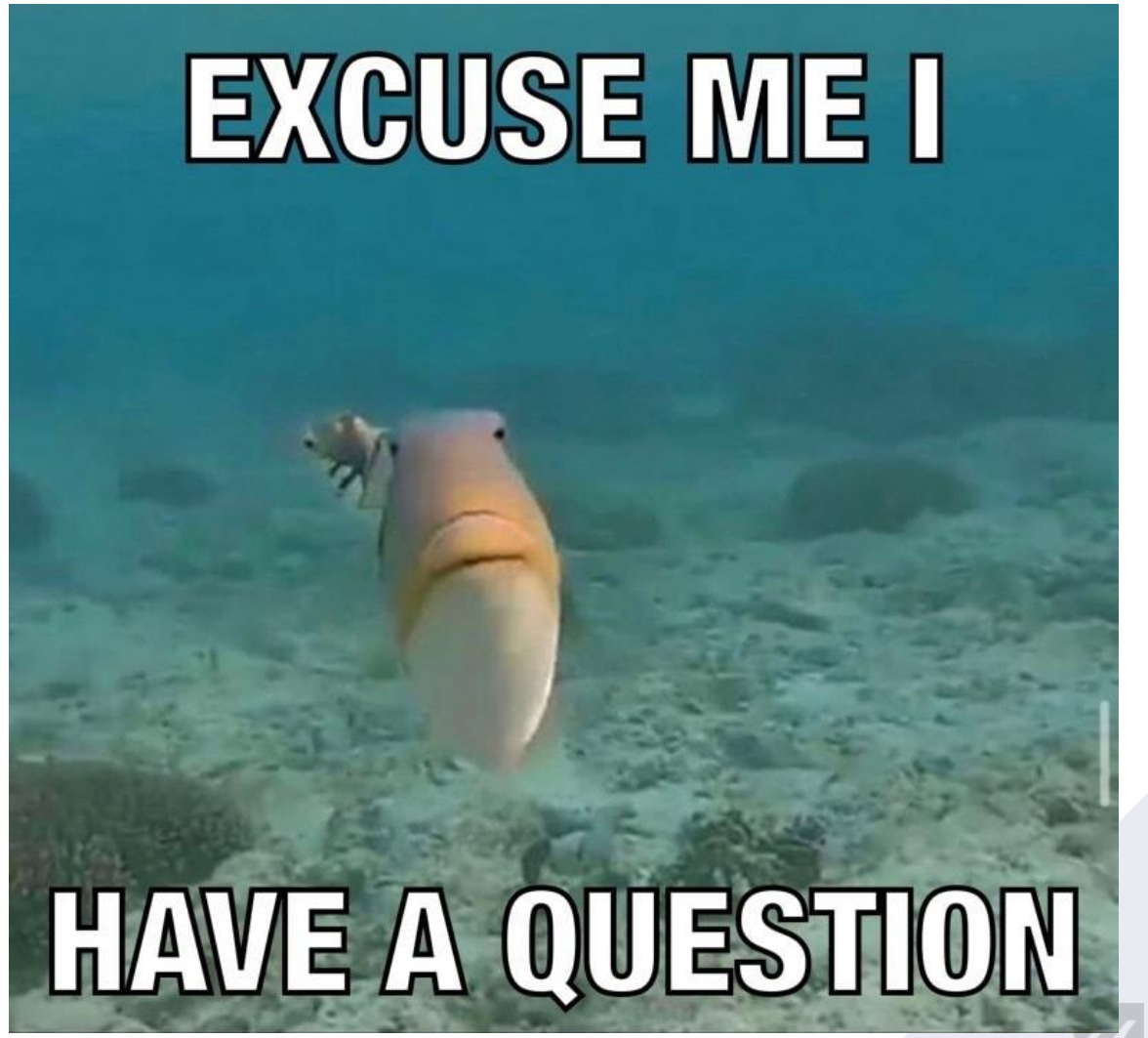
- Prioritization
  - Helps in ranking what features or tasks are more important or urgent
  - Allows for better sprint planning
- Traceability
  - Easy to track progress at both micro (user story) and macro (epic) levels
  - Helps in accountability and performance metrics
- Flexibility
  - User stories can be easily updated or reprioritized
  - Allows for agile response to changes in project requirements or market conditions

PROJECT PLANNING: BRAINSTORM PROJECT REQUIREMENTS AND CREATE A SAMPLE BACKLOG

- Epic: Degree Planning
  - As a student, I want to be able to create a long-term degree plan so that I can see what classes I need to take each semester.
  - As a student, I want to be able to edit my degree plan so that I can make adjustments as needed.
- Epic: User Interface
  - As a user, I want to have a responsive design so that I can use the app on any device.
  - As a user, I want to be able to easily navigate through the app so that I can find what I'm looking for quickly.
- Epic: Schedule Sharing
  - As a student, I want to be able to share my class schedule with friends so that we can coordinate our classes.
  - As a student, I want to be able to export my schedule so that I can print it or save it for later.

PROJECT PLANNING: QUESTIONS?

- Questions?



SOFTWARE ENGINEERING PRACTICES

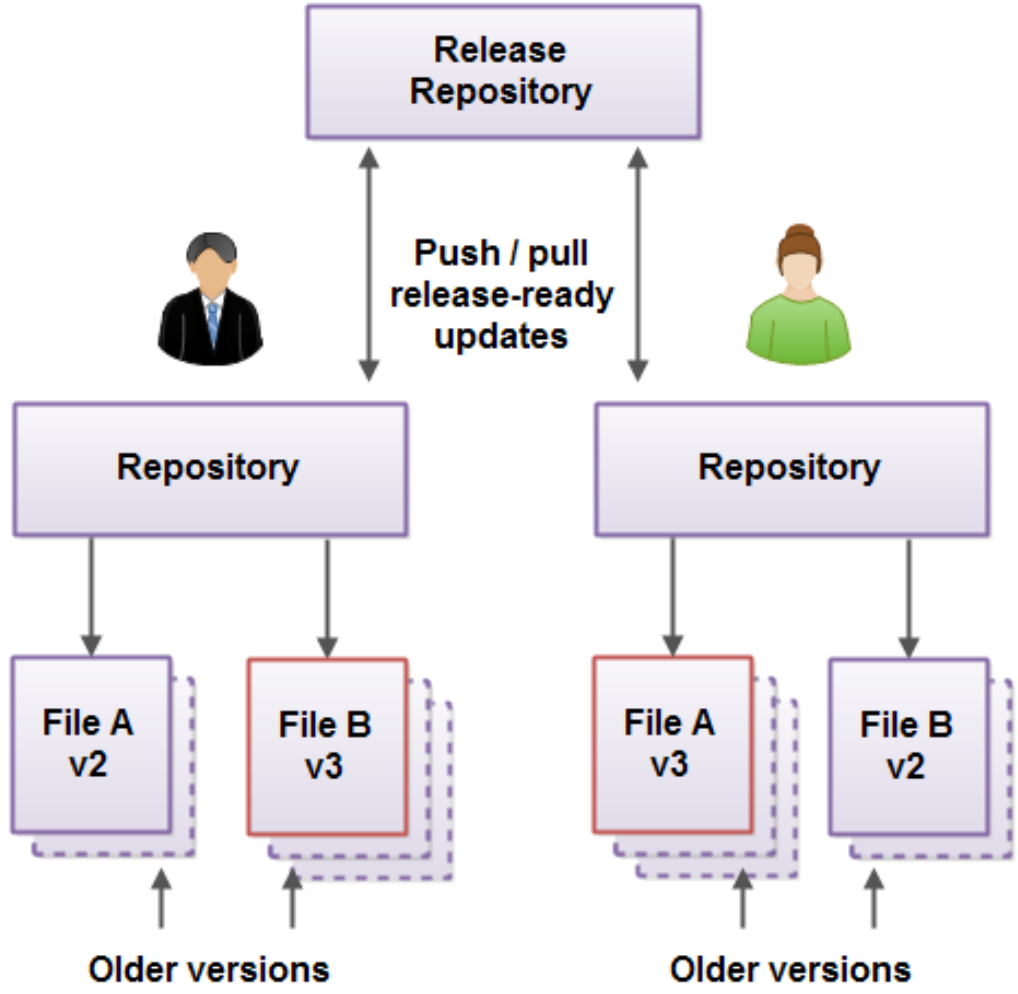
- Importance of code quality and readability
- Introduction to Git as a version control system
- Basic Git commands: clone, add, commit, push, pull
- GitHub for collaborative coding: Repositories, Issues, Pull Requests
- Discuss coding standards, linting, and code documentation
- Role of testing: Unit tests, Integration tests, and End-to-End tests

SOFTWARE ENGINEERING PRACTICES: IMPORTANCE OF CODE QUALITY AND READABILITY

- Why is Code Quality Important?
  - **Reliability**: Well-written code is less prone to errors and bugs.
  - **Maintainability**: High-quality code is easier to update and maintain.
  - **Scalability**: Quality code can be easily expanded for future requirements.
- Why is Code Readability Important?
  - **Collaboration**: Readable code is easier for team members to understand and work on.
  - **Debugging**: Easier to identify issues and implement fixes.
  - **Onboarding**: New team members can quickly understand and contribute to the project.
- Best Practices for Code Quality
  - Code Reviews
  - Unit Testing
  - Continuous Integration
- Best Practices for Code Readability
  - Proper Naming Conventions
  - Commenting and Documentation
  - Consistent Indentation and Formatting

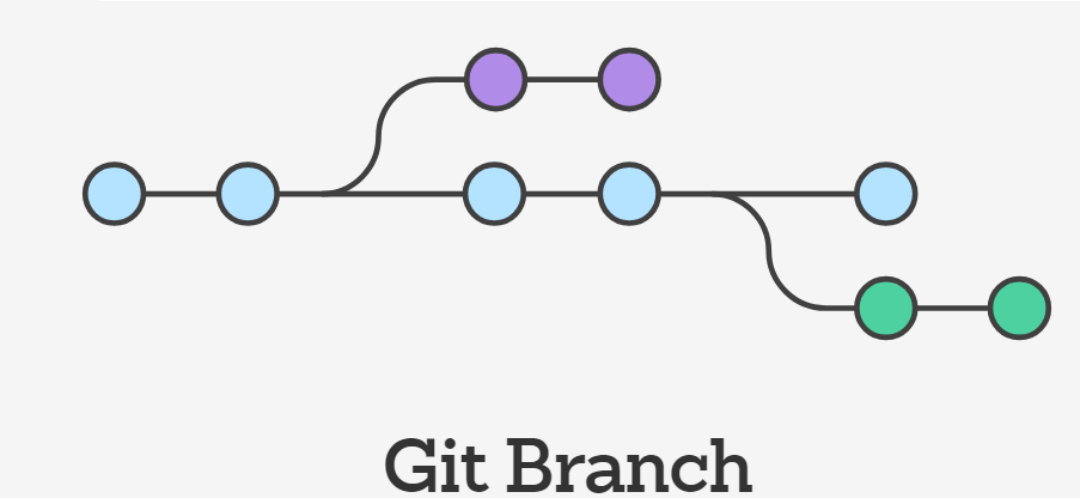
SOFTWARE ENGINEERING PRACTICES: INTRODUCTION TO GIT AS A VERSION CONTROL SYSTEM

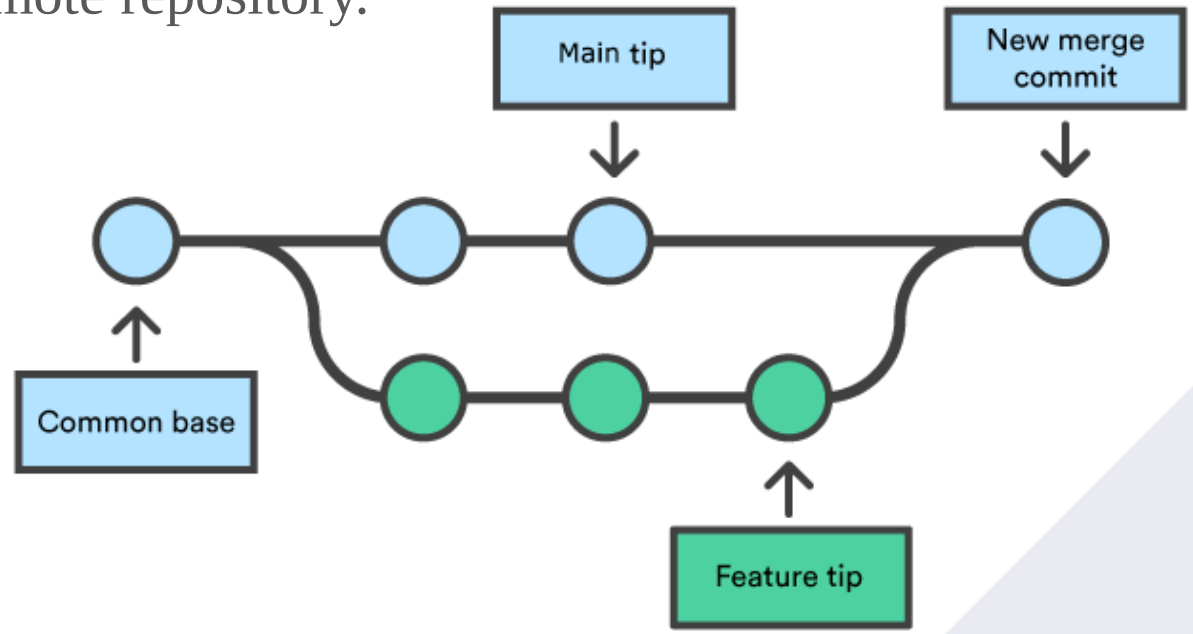
- What is Git?
  - A distributed version control system that helps track changes in source code during software development.
- Why Use Git?
  - **Collaboration**: Allows multiple developers to work on the same project.
  - **Versioning**: Keeps a history of all changes, making it easy to revert to previous versions.
  - **Branching**: Enables working on different features or bugs simultaneously without affecting the main codebase.



SOFTWARE ENGINEERING PRACTICES: INTRODUCTION TO GIT AS A VERSION CONTROL SYSTEM

- Core Concepts in Git
  - **Repository**: The container for the source code and its history.
  - **Commit**: A snapshot of changes in the code.
  - **Branch**: A separate line of development within the same project.
  - **Merge**: Combining changes from different branches.
  - **Clone**: Creating a local copy of a remote repository.





SOFTWARE ENGINEERING PRACTICES: GITHUB FOR COLLABORATIVE CODING: REPOSITORIES, ISSUES, PULL REQUESTS

- What is GitHub?
  - A web-based platform that uses Git for version control, enabling multiple people to collaborate on projects.
- Key Features:
  - **Repositories**
    - Online storage space for your project.
    - Can be public or private.
    - Includes code, documentation, and other project files.
  - **Issues**
    - Used for tracking bugs, enhancements, tasks, and other project-related concerns.
    - Anyone can create an issue, and project collaborators can comment and assign tasks.
  - **Pull Requests**
    - The process of submitting changes for review.
    - Allows for code review and discussion before merging.
    - Can be linked to issues to track the completion of tasks.

SOFTWARE ENGINEERING PRACTICES: GITHUB FOR COLLABORATIVE CODING: REPOSITORIES, ISSUES, PULL REQUESTS

HTTPS://WWW.YOUTUBE.COM/WATCH?V=D3N2YEAALKC

SOFTWARE ENGINEERING PRACTICES: DISCUSS CODING STANDARDS, LINTING, AND CODE DOCUMENTATION

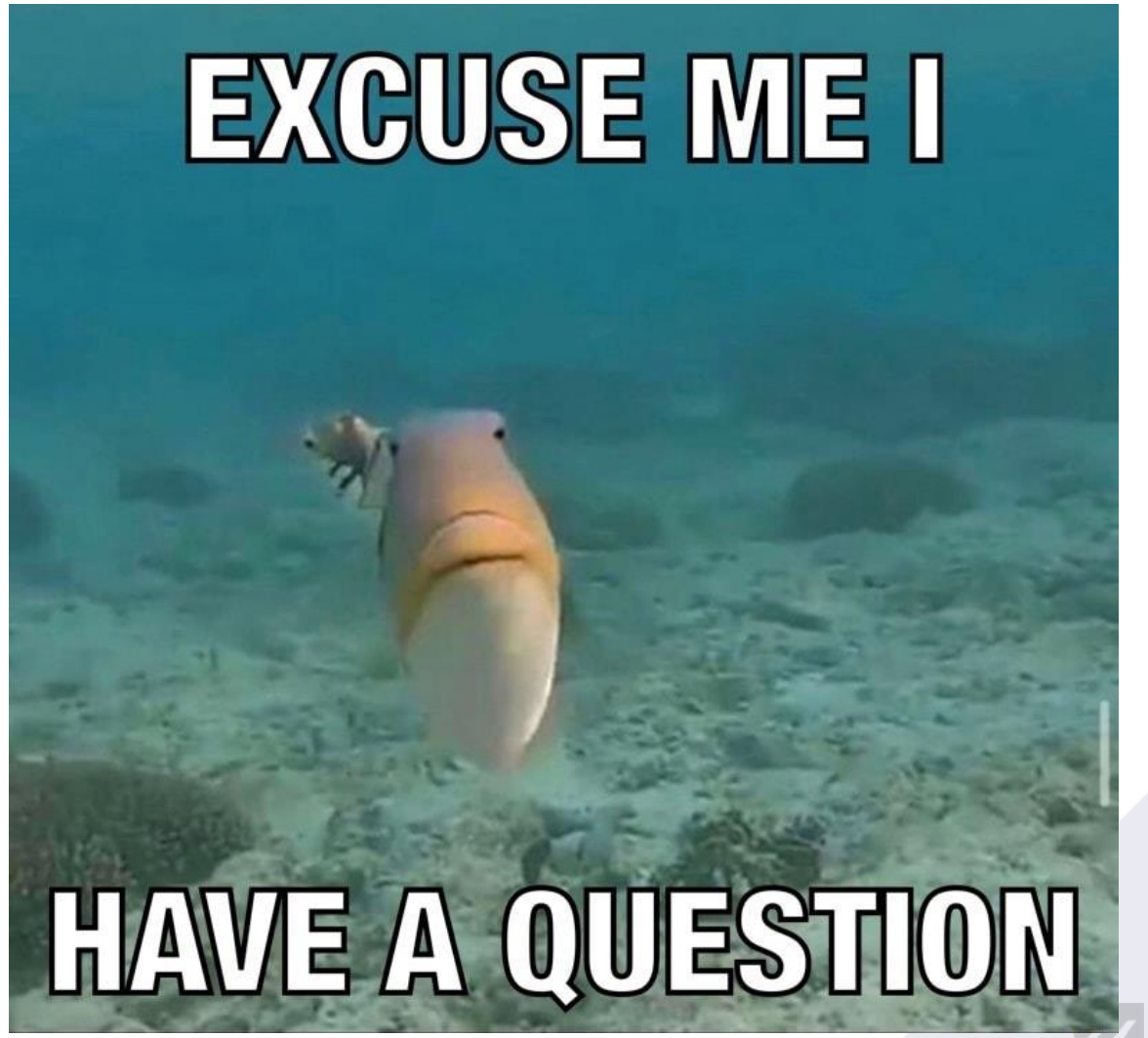
- Coding Standards
  - Set of guidelines for writing code.
  - Ensures consistency and readability across the project.
- Linting
  - Automated code analysis tool.
  - Helps to enforce coding standards and catch errors early.
- Code Documentation
  - Comments and documentation within the code.
  - Makes it easier to understand the code's purpose and functionality.
- Why It Matters
  - Ensures code quality and maintainability.
  - Facilitates collaboration by making code easier to read and understand.

SOFTWARE ENGINEERING PRACTICES: ROLE OF TESTING: UNIT TESTS, INTEGRATION TESTS, AND END-TO-END TESTS

- Unit Tests
  - Test individual components or functions.
  - Quick to run, easy to debug.
- Integration Tests
  - Test the interaction between multiple components.
  - Validates that different parts of the system work together.
- End-to-End Tests
  - Test the system as a whole, often using automated UI tests.
  - Validates the entire process from start to finish.
- Why It Matters
  - Ensures reliability and robustness of the software.
  - Catches issues early, reducing the cost of fixing bugs.

SOFTWARE ENGINEERING PRACTICES: QUESTIONS?

- Questions?



INTRODUCTION TO DEVOPS

- Definition and significance of DevOps in modern software development
- Importance of Continuous Integration (CI) and Continuous Deployment (CD)
- How Git and GitHub Actions fit into CI/CD pipelines
- Brief overview of DevOps tools like Jenkins, Docker, and Kubernetes
- Discuss the DevOps lifecycle: Plan, Code, Build, Test, Release, Deploy, Operate, Monitor

INTRODUCTION TO DEVOPS: DEFINITION AND SIGNIFICANCE OF DEVOPS IN MODERN SOFTWARE DEVELOPMENT

- What is DevOps?
  - DevOps is a set of practices that combines software development (Dev) and IT operations (Ops).
  - Aims to shorten the system development life cycle and provide continuous delivery.
- Significance in Modern Software Development
  - Accelerates the development process.
  - Enhances collaboration between development and operations teams.
  - Improves efficiency through automation.
  - Increases the reliability and quality of the software.
- Key Components
  - **Continuous Integration (CI)**
  - **Continuous Deployment (CD)**
  - **Infrastructure as Code (IaC)**
- Why It Matters
  - Enables rapid innovation and faster time-to-market.
  - Reduces the cost and risk of software changes.

Learning with Purpose

INTRODUCTION TO DEVOPS: IMPORTANCE OF CONTINUOUS INTEGRATION (CI) AND CONTINUOUS DEPLOYMENT (CD)

- What is Continuous Integration (CI)?
  - CI is the practice of automatically integrating code changes from multiple contributors into a single software project.
  - Involves automated testing to detect errors early.
- What is Continuous Deployment (CD)?
  - CD is the automated process of deploying software changes to a staging or production environment.
  - Ensures that you can release new changes to your customers quickly in a sustainable way.
- Why Are CI/CD Important?
  - **Speed**: Accelerate release cycles, enabling faster innovation.
  - **Quality**: Automated tests improve software quality, reducing the cost of defects.
  - **Reliability**: Consistent, automated deployment process improves reliability.
  - **Collaboration**: Encourages better collaboration between developers, testers, and operations.

INTRODUCTION TO DEVOPS: HOW GIT AND GITHUB ACTIONS FIT INTO CI/CD PIPELINES

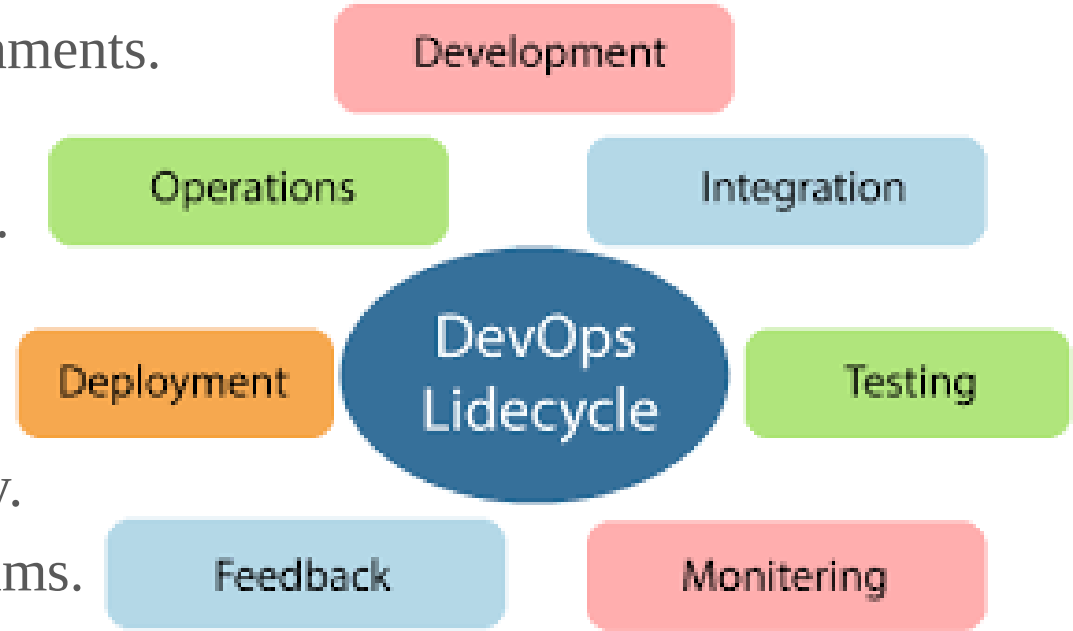
- Git
  - **What is it?:** A distributed version control system.
  - **Role in CI/CD**: Source code management, versioning, and branching.
  - **Why Use Git?:** Enables collaboration and serves as the starting point for CI/CD pipelines.
- GitHub Actions
  - **What is it?:** A CI/CD service by GitHub.
  - **Role in CI/CD:** Automates workflows, including build, test, and deploy tasks.
  - **Why Use GitHub Actions?:** Seamlessly integrates with GitHub repositories, customizable with YAML files.
- Integration Points
  - **Source Code:** Git repositories serve as the source of truth.
  - **Automation**: GitHub Actions reads from the Git repository to execute CI/CD tasks.
  - **Feedback Loop**: Both can notify developers of build statuses, errors, or successful deployments.

INTRODUCTION TO DEVOPS: BRIEF OVERVIEW OF DEVOPS TOOLS LIKE JENKINS, DOCKER, AND KUBERNETES

- Jenkins
  - **What is it?:** An open-source automation server.
  - **Key Features**: Build automation, testing, and deployment.
  - **Why Use Jenkins**?: Streamlines the CI/CD pipeline, integrates with a multitude of technologies.
- Docker
  - **What is it?:** A platform for containerization.
  - **Key Features**: Package, distribute, and manage applications within containers.
  - **Why Use Docker**?: Ensures consistency across multiple development and release cycles.
- Kubernetes
  - **What is it?:** An open-source container orchestration platform.
  - **Key Features**: Automates deployment, scaling, and management of containerized applications.
  - **Why Use Kubernetes**?: Manages and scales complex applications with ease.

INTRODUCTION TO DEVOPS: DISCUSS THE DEVOPS LIFECYCLE: PLAN, CODE, BUILD, TEST, RELEASE, DEPLOY, OPERATE, MONITOR

- The DevOps Lifecycle
  - **Plan**: Define objectives, requirements, and KPIs.
  - **Code**: Develop the application, write scripts, and store in version control.
  - **Build**: Compile, assemble, and package the code.
  - **Test**: Validate functionality, performance, and security.
  - **Release**: Prepare the code for deployment, versioning, and rollback plans.
  - **Deploy**: Move the code to production or other environments.
  - **Operate**: Manage and scale the application.
  - **Monitor**: Track performance, errors, and user activity.
- Why It Matters
  - Ensures a seamless transition between Dev and Ops.
  - Facilitates continuous improvement and rapid delivery.
  - Enhances collaboration and communication across teams.



Learning with Examples

INTRODUCTION TO DEVOPS

HTTPS://WWW.YOUTUBE.COM/WATCH?V=SCEDHSR3APG

INTRODUCTION TO DEVOPS: QUESTIONS?

- Questions?