# Agile Methodology and Git: An Overview

## 1. Agile Methodology

### 1.1 Introduction to Agile

Agile methodology is a modern approach to software development and project management that emphasizes flexibility, collaboration, and iterative progress. It is designed to accommodate changing requirements while delivering incremental improvements in product development.

### 1.2 Core Principles of Agile

Agile is based on the Agile Manifesto, which includes four key values:

1. Individuals and interactions over processes and tools.

2. Working software over comprehensive documentation.

3. Customer collaboration over contract negotiation.

4. Responding to change over following a plan.

Additionally, Agile follows 12 principles that encourage continuous delivery, customer satisfaction, adaptability, and teamwork.

### 1.3 Agile Frameworks

Several frameworks operate under the Agile umbrella, including:

• Scrum: A structured framework with defined roles (Scrum Master, Product Owner, Development Team) and time-boxed iterations (sprints).

• Kanban: A visual workflow management system that optimizes efficiency by limiting work in progress.

• Extreme Programming (XP): Focuses on engineering practices like test-driven development (TDD) and pair programming.

• Lean Development: Emphasizes minimizing waste and maximizing value for the customer.

### 1.4 Benefits of Agile

• Faster delivery of high-quality software.

• Improved collaboration between stakeholders and teams.

• Greater adaptability to changing requirements.

• Increased customer satisfaction due to continuous feedback and improvements.

## 2. Git: Version Control System

### 2.1 Introduction to Git

Git is a distributed version control system (DVCS) that allows developers to track changes, collaborate efficiently, and maintain a history of their codebase. It was created by Linus Torvalds in 2005 to support Linux kernel development.

### 2.2 Core Concepts of Git

• Repository (Repo): A storage location containing all files and history of a project.

• Commit: A snapshot of changes recorded in the repository.

• Branching: A mechanism to create parallel versions of the codebase.

• Merging: Integrating changes from different branches into a single branch.

• Remote Repository: A shared repository hosted on platforms like GitHub, GitLab, or Bitbucket.

### 2.3 Common Git Commands

• git init – Initializes a new Git repository.

• git clone <repo\_url> – Clones an existing repository.

• git add <file> – Stages changes for the next commit.

• git commit -m "message" – Saves changes with a descriptive message.

• git pull origin <branch> – Fetches and merges changes from a remote branch.

• git push origin <branch> – Sends local commits to the remote repository.

• git checkout <branch> – Switches between branches.

• git merge <branch> – Merges a branch into the current branch.

### 2.4 Benefits of Using Git

• Efficient collaboration across teams.

• Provides a complete history of changes.

• Enables experimentation through branching and merging.

• Supports distributed workflows, allowing multiple developers to work on different features simultaneously.

### 2.5 Git and Agile Integration

Git complements Agile methodology by facilitating:

• Continuous integration (CI) and continuous deployment (CD).

• Feature branching for iterative development.

• Easy rollback of changes in case of errors.

• Improved team collaboration with version tracking.