



Version Controls

Git and Github



Presented By:
G. Chandana

Overview

1. Git & Github
2. Difference B/W Git & Github
3. Deployment
4. Difference B/W Google drive & Github
5. Jeera
6. Agile Methodology
7. Waterfall Methodology
8. SDLC



Git

Git and GitHub are two related but distinct tools commonly used in software development for version control and collaboration. Here's an explanation of each:

- Git is a distributed version control system for tracking changes in source code during software development. It allows for coordinating work among programmers, and it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Key features of Git include:

- **Version Control:** Git tracks changes to files over time, allowing developers to revert to previous versions, compare changes, and merge code changes from multiple contributors.
- **Branching and Merging:** Git enables developers to create separate branches for new features or bug fixes, work on them independently, and later merge them back into the main codebase.
- **Distributed Architecture:** Each developer has a complete copy of the project's repository, including the entire history of changes. This distributed model allows developers to work offline and synchronize changes with remote repositories.
- **Lightweight and Fast:** Git is designed to be lightweight and efficient, with fast performance for common operations such as committing changes, branching, and merging.

Github

GitHub, on the other hand, is a web-based Git repository hosting service that offers all of the distributed version control and source code management (SCM) functionality of Git, as well as additional features such as project management, support ticket management, and bug tracking. It provides a cloud-based platform for developers to store and manage their code, and it allows for collaboration and interaction with other developers' code.

GitHub repositories are open to the public, making it a networking site for web professionals. It is commonly used for storing and sharing code, and it allows for the creation of remote copies of repositories to serve as backups.

Git is a software tool that is installed and used locally on a developer's machine, while GitHub is a cloud-based service that is accessed through the web. It is worth noting that Git can be used without GitHub, but GitHub cannot be used without Git.

Key features of GitHub include:

- **Repository Hosting:** GitHub hosts Git repositories in the cloud, allowing developers to access and collaborate on projects from anywhere with an internet connection.
- **Pull Requests:** GitHub's pull request feature enables developers to propose changes to a repository, request code reviews from peers, and discuss and collaborate on changes before merging them into the main codebase.
- **Issue Tracking:** GitHub provides tools for tracking bugs, feature requests, and other tasks using a built-in issue tracker. Developers can create, assign, and prioritize issues, as well as link them to specific code changes.
- **Collaboration Tools:** GitHub offers features such as wikis, project boards, and discussions to facilitate collaboration and communication among team members.
- **Integration Ecosystem:** GitHub integrates with a wide range of third-party tools and services for continuous integration (CI), deployment, code quality analysis, and project management, enhancing the development workflow.

Difference B/W Git & Github

Git

Git is a software.

Git is a Command-line tool.

Git is installed locally on the system

Git is maintained by linux

Github

GitHub is a service.

GitHub is a graphical user interface.

GitHub is a hosted on the web

GitHub is maintained by Microsoft



Git is focused on version control and code sharing.

Git is a version control system to manage source code history

Git was first released in 2005.

Git has no user management feature.

Git is open-source licensed.

Git has minimal external tool configuration.

GitHub is focused on centralized source code hosting.

GitHub is a hosting service for Git repositories.

GitHub was launched in 2008.

GitHub has a built-in-user management feature.

GitHub includes a free-tier and pay-for-use tier.

GitHub has an active marketplace for tool integration.



Git provides a Desktop interface named Git Gui.

GitHub provides a Desktop interface named GitHub Desktop

Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc.


GitHub competes with GitLab, Bit Bucket, AWS Code Commit, etc.

Deployment

Deployment in the context of software development refers to the process of making a software application available for use by end-users or customers. It involves transferring the application from a development environment to a production environment where it can be accessed and utilized by its intended audience.

Here's a breakdown of the deployment process:

1. **Development:** The software application is developed, tested, and refined in a controlled environment, often referred to as a development or staging environment. Developers work on coding, debugging, and testing the application's features and functionalities during this phase.
2. **Version Control:** The application's source code, configuration files, and other necessary assets are managed using version control systems like Git. Version control allows developers to track changes, collaborate on code, and maintain a history of revisions.
3. **Build Process:** The application code is compiled, bundled, and prepared for deployment. This may involve tasks such as compiling code into executable files, minifying and optimizing assets (e.g., JavaScript, CSS), and generating documentation.





4. **Testing:** Before deployment, the application undergoes various testing phases to ensure its quality, stability, and functionality. This includes unit testing, integration testing, regression testing, and user acceptance testing (UAT). Testing helps identify and resolve any issues or bugs before the application is deployed to production.

5. **Deployment Pipeline:** A deployment pipeline is a series of automated steps that automate the deployment process, from code changes to production release. Continuous Integration (CI) and Continuous Deployment (CD) practices are often used to automate the build, test, and deployment phases, enabling faster and more reliable deployments.

6. **Deployment Environment:** The application is deployed to a production environment, which is typically a server or cloud infrastructure where end-users can access the application. Depending on the application's architecture, deployment may involve deploying to physical servers, virtual machines, containers (e.g., Docker), or serverless platforms (e.g., AWS Lambda).


7. **Configuration and Setup:** Configuration files, environment variables, and other settings are configured for the production environment. This includes database connections, API endpoints, security settings, and other environment-specific configurations.






8. **Monitoring and Maintenance:** After deployment, the application is monitored for performance, availability, and security issues. Monitoring tools and alerts help detect and respond to issues in real-time, ensuring the application remains operational and performs optimally. Regular maintenance, updates, and patches are applied to keep the application secure and up-to-date.

9. **Rollback and Recovery:** In the event of deployment failures or issues, rollback procedures are in place to revert to a previous known-good state. This minimizes downtime and disruptions to users while issues are investigated and resolved.

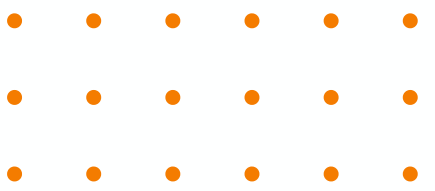


Difference B/W Google Drive & GitHub

Features	Google Drive	GitHub
Primary Function	File storage and synchronization	Version control and collaboration on software development projects.
File Types	All file types, including document, images and videos	Text-based files, such as code and documentation.
Version Control	Manual versioning	Built-in version control using Git
Collaboration	Real-time collaboration on documents	Collaboration and Project management on code repositories



Access	Accessible through web and mobile apps	Accessible through web and command-line interface
Users	Personal and commercial use	Primarily used by developers and organization for software development
Open source	Limited support for open source	Large repository of open-source projects
Security	Encryption in transit and at rest	Encryption in transit, option for at-rest encryption
Pricing	Free storage up to 15GB, paid plans for additional storage	Free and paid plans for private and public repositories



Jira

JIRA is a software development tool used for project management and issue tracking. It is a popular tool among software development teams to plan, track, and release software projects. JIRA provides a centralized platform for managing tasks, bugs, and other types of issues and it helps teams to organize and prioritize their work. The tool integrates with other software development tools and has a variety of customizable features and workflows that allow teams to adapt it to their specific needs. Additionally, JIRA also provides various reporting and dashboard features that help teams stay on top of their work and make data-driven decisions. JIRA supports multiple languages including English, French, etc. It is a platform-independent tool.

Agile Methodology

Agile methodology is an iterative approach to software development that emphasizes flexibility, collaboration, and customer feedback. It focuses on delivering small, incremental releases of software in short cycles, known as iterations or sprints, rather than trying to deliver the entire project at once. The Agile methodology prioritizes customer satisfaction, adaptability to change, and continuous improvement throughout the development process.



Waterfall Methodology


The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.





The image depicts a person's legs and feet in tan trousers and brown shoes, ascending a staircase. The staircase is composed of six horizontal steps that decrease in width from top to bottom, creating a descending staircase effect. The steps are colored in a gradient of blue, with the top step being the darkest and the bottom step being the lightest. The background features light blue abstract shapes and a large light blue circle. Orange decorative elements are visible in the corners of the overall image.

Requirements

Analysis

Design

Implementation

Testing

Maintenance

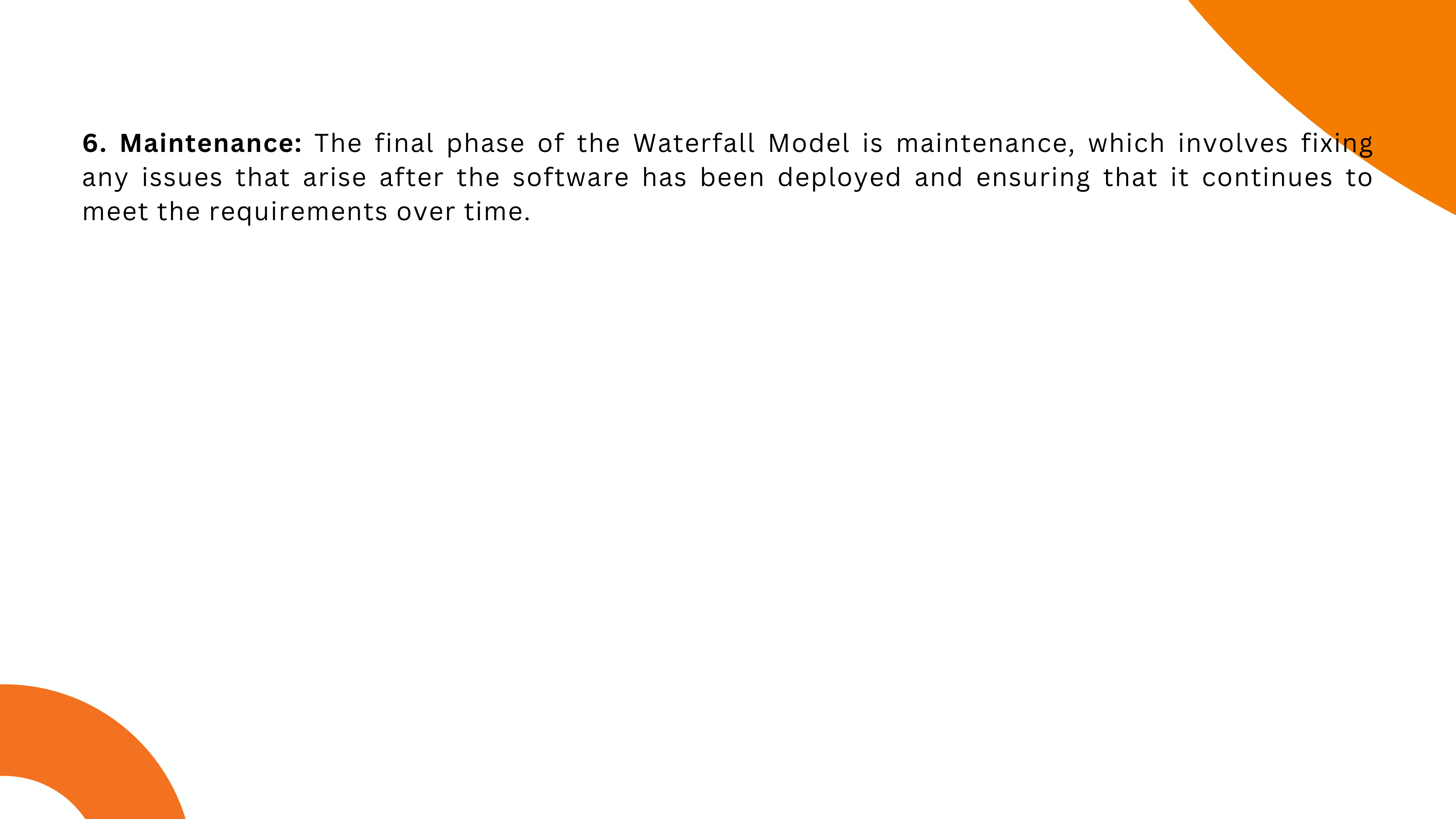
1. **Requirements:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

2. **Analysis:** The analysis stage in the Waterfall model is the first stage of the development process, where the project team collects and analyzes the system requirements. This stage involves gathering information from various stakeholders, including end users, system owners, and other project team members. The requirements are then documented in a detailed requirement specification document, which serves as a blueprint for the development team.

3. **Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

4. **Implementation:** With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

5. **Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.



6. Maintenance: The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

SDLC

SDLC stands for Software Development Life Cycle. It refers to the process of planning, creating, testing, and deploying software applications or systems. It is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.

Here, are prime reasons why SDLC is important for developing a software system.

- It offers a basis for project planning, scheduling, and estimating
- Provides a framework for a standard set of activities and deliverables
- It is a mechanism for project tracking and control
- Increases visibility of project planning to all involved stakeholders of the development process
- Increased and enhance development speed
- Improved client relations
- Helps you to decrease project risk and project management plan overhead

