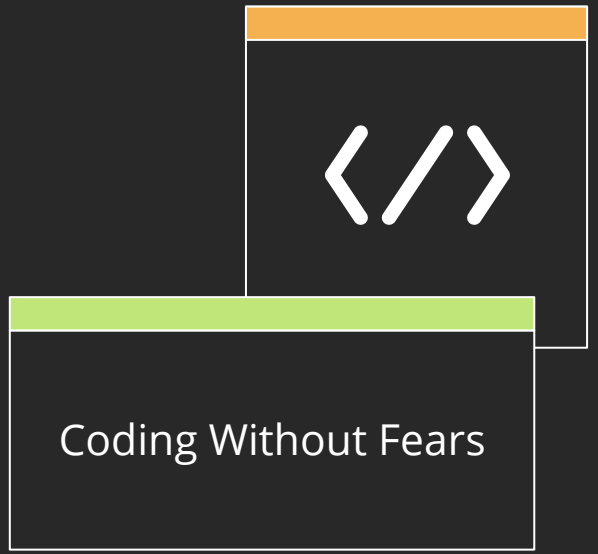
Version Control System VCS



By and Supervision

احمد عاطف احمد عبد الغنى عطيوى	16
اسلام مجدي عبد الحكيم الفرماوي	13
صلاح احمد صلاح الدين عشاوى	54
علاء على عبد السلام عبد السلام	76
محمد عبد الفتاح مصطفى محمد عبد الفتاح	115

Under Supervision: Prof. Abdulrahman Nasr

Main Points

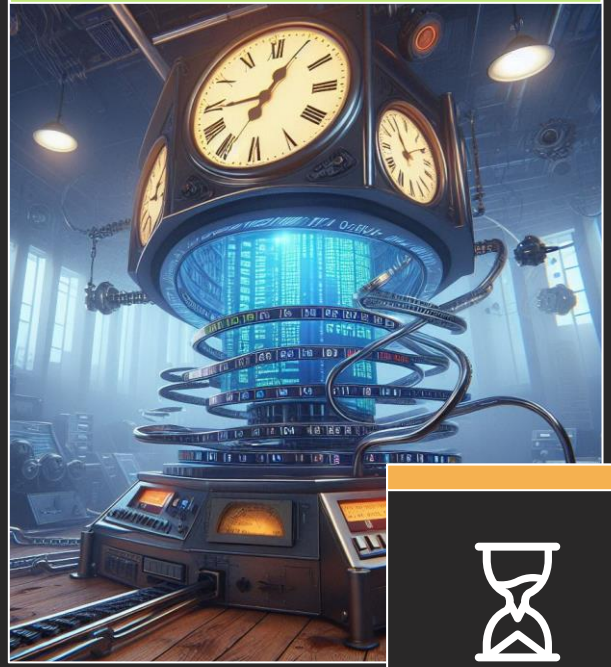
1010101011100011110

1100101010101011110

- 00 Introduction
- 01 About VCS
- 02 Types of VCS
- 03 VCS Tools
- 04 About Git
- 05 Git Commands
- 06 About GitHub
- 07 Git and VSCode
- 08 Live Demo

INTRODUCTION

Just imagine the VCS as a Time Machine...



1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

VCS As A Time Machine

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

"Ladies and gentlemen, fasten your seatbelts and prepare to embark on a journey through time. But this is no ordinary time machine; it's the most powerful tool in the arsenal of a software developer – a Version Control System.

Imagine being able to travel back to any point in your project's history, revisit decisions, and explore alternate realities where you took a different coding path. With a VCS, you're the master of time, able to undo mistakes, branch off into parallel universes, and merge timelines.

In the world of software development, Git is our DeLorean, and each commit is a timestamp on our journey. Today, we'll learn how to harness this power, navigate through our project's timeline, and collaborate with fellow time travelers across the globe.

So, let's start the engine, punch in the coordinates, and see how Git can make us all lords of time in our coding adventures."

1
0
1
0
0
1
0
1
1
0
1
1
1

01

About VCS

About VCS

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

Definition of VCS:

Version Control Systems (VCS) are software tools designed to help software teams manage changes to source code over time. They track every modification to the code in a special kind of database, allowing developers to recall earlier versions if needed and integrate work done simultaneously by different team members.

Importance of VCS:

1. **Change Tracking**: VCS allows for detailed tracking of code changes, making it possible to see who changed what and when, which is crucial for understanding the evolution of a project.
2. **Collaboration**: It enables multiple people to work on the same project concurrently, providing tools to merge changes and resolve conflicts.
3. **Historical Access**: Developers can access historical versions of a project, which is invaluable for troubleshooting and understanding past decisions.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

About VCS

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

Importance of VCS:

4. Code Quality: Systematic code reviews are facilitated by VCS, promoting higher code quality and consistency across the project.

5. Release Management: VCS supports streamlined release management, maintaining different versions of software releases and aligning them with the release roadmap.

6. Conflict Prevention: By maintaining separate branches for different features or releases, VCS minimizes the chance of overlapping changes that cause conflicts.

7. Backup and Recovery: In case of a catastrophe or accidental changes, VCS serves as a backup from which the codebase can be restored to a previous state.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

02

Types of VCS

Types VCS

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

Version Control Systems (VCS) are categorized into three main types, each with its unique approach to version management:

1. **Local Version Control Systems (LVCS)**:
    - A system that stores changes to files in a simple local database to keep track of modifications.
2. **Centralized Version Control Systems (CVCS)**:
    - A single server stores all versions of a project's files, and developers check out files to work on them.
3. **Distributed Version Control Systems (DVCS)**:
    - Every developer has a local copy of the entire repository, allowing for independent commits, branches, and merges.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

03

VCS Tools

VCS Common Tools

1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0

Git

Distributed Nature

SVN

Centralized Approach

Mercurial

Simplicity and efficiency

And way more...

VCS Common Tools

The three most commonly used Version Control Systems (VCS) and their distinctive advantages are:

1. **Git**: Known for its **distributed nature**, Git allows every developer to have a full copy of the entire repository, making it possible to work offline and enabling robust collaboration.
2. **SVN (Subversion)**: SVN is appreciated for its **centralized approach**, which is simpler for non-technical contributors to understand and use, and it supports better management of binary files.
3. **Mercurial**: It stands out for its **simplicity and efficiency**, offering an intuitive interface that's easier for newcomers to version control, making the initial learning curve less steep.

These systems are widely adopted in the industry due to these significant advantages that cater to different project needs and organizational workflows.

04 About Git

About Git

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

Git is a powerful and widely-used Version Control System (VCS) that has revolutionized how developers track and manage changes to their code. Here's a brief overview:

What is Git? Git is a free and open-source distributed VCS designed to handle projects of all sizes with speed and efficiency. It allows every developer to have a complete history of the codebase, enabling offline work and easy collaboration.

Key Features of Git:

- Efficient Performance: Git is known for its fast performance, handling large projects with ease.
- Branching and Merging: Git's branching model allows for multiple independent lines of development, which can be easily merged back into the main codebase.
- Distributed Development: With Git, every developer has a full copy of the repository, including its history, which enhances collaboration and provides a backup of the code.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

About Git

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

Data Integrity: Git ensures the integrity of the codebase by using a cryptographic hash function (SHA-1) to track and manage changes.

- Staging Area: Git provides a staging area or “index” where changes can be formatted and reviewed before committing them to the repository.

Why to use Git?

- Flexibility: Git supports various workflows, from solo development to large, distributed teams.
- Resilience: The distributed nature of Git means there’s no single point of failure.
- Community: Git has a large and active community, providing a wealth of resources and support.

Git’s impact on software development is profound, offering a robust set of tools for managing complex projects and fostering collaboration among developers worldwide.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

05

Git Basic Commands

Git Basic Commands

1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 1 0

git init

git clone

git add

git commit

git status

git push

git pull

git branch

git checkout

git merge

1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

Git Basic Commands

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Here are some basic Git commands that are essential for any developer working with this Version Control System:

- git init: Initializes a new Git repository and begins tracking an existing directory.
- git clone [url]: Copies a Git repository from a remote source, including all the files, branches, and commits.
- git add [file]: Adds a file to the staging area, preparing it for inclusion in the next commit.
- git commit -m "[message]": Records file snapshots permanently in the version history along with a descriptive message.
- git status: Displays the status of changes as untracked, modified, or staged.
- git push [alias] [branch]: Uploads all local branch commits to the remote repository.
- git pull [alias] [branch]: Fetches and merges any commits from the tracking remote branch.
- git branch: Lists all the branches in your repo, and also tells you what branch you're currently in.
- git checkout [branch-name]: Switches to the specified branch and updates the working directory.
- git merge [branch]: Combines the specified branch's history into the current branch.

These commands form the foundation of most workflows in Git, allowing you to track, manage, and share your code efficiently.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

06

About GitHub

About GitHub

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

GitHub is a web-based platform that hosts projects utilizing Git, a Distributed Version Control System (DVCS). It enhances Git's capabilities by providing a graphical interface and collaboration features such as pull requests, issue tracking, and code reviews. GitHub allows developers to save different versions of files, track changes, and collaborate on projects with others. It's particularly known for fostering open-source projects and community-driven development

Key Features of GitHub:

- **Collaboration**: GitHub's pull request system streamlines the process of discussing and reviewing code changes before they are merged into the main project.
- **Project Management**: Issue tracking and project boards help teams organize tasks, milestones, and progress.
- **Code Hosting**: It provides a central place to store repositories, making it easier to share and access code.
- **Documentation**: With GitHub, you can also host project documentation alongside your code, ensuring that everything stays in sync.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

07

Git and VSCode

Git and VSCode

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Git and Visual Studio Code (VSCode) integrate seamlessly to provide a powerful and efficient development environment. Here's how they work together:

- **Built-in Git Support**: VSCode has built-in support for Git, allowing you to perform common Git commands directly within the editor.
- **Source Control Tab**: The Source Control tab in VSCode tracks all changes and lets you stage, commit, push, and pull changes without leaving the editor.
- **GitHub Integration**: With the GitHub Pull Requests and Issues extension, you can collaborate on code, review pull requests, and manage issues directly from VSCode.
- **User-Friendly Interface**: Even beginners find it easy to manage their source code and collaborate with others using the intuitive interface provided by VSCode.
- **Efficient Workflow**: The integration allows developers to stay in their editor while managing Git repositories, streamlining their workflow and saving time.

This integration makes VSCode a popular choice for developers who want to leverage Git's capabilities while enjoying the rich features of a modern code editor.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1


08
Live Demo
1010101101110001110
1001010101011110

1
0
1
0
1
0
1
1
1
0
0
0
1
1
1
0

<THANKS />

1
1
0
0
1
0
1
0
1
0
1
0
1
1
1
0