WELCOME

TechPro

[Pavlos Xouplidis]

[Git and code Editor Setup]

[07/10/2024]

In this Course

1. Understand the need and use of Git
2. Install and Configure Git
3. Run most common used git commands
4. Understand the need and use of a centralized development environment
5. Install and be familiar with IDEs
6. You use VS Code as your development environment
7. The powerful extension market of VS Code
8. Connect VS code with git and start programming
9. Jupyter notebooks the need, the use and Markdown syntax

TechPro

Survey

{Programming Experience}

https://forms.office.com/e/KZJZWcUpPT?origin=lprLink

TechPro

Discussion

{What is Git? What is an IDE?}

TechPro

Version Control System? Why would you need one?

Collaboration: In any team, multiple people are working on the same project. A VCS allows each team member to work on the same codebase without overwriting each other's changes. It manages merging of code and conflict resolution.

Tracking changes: A VCS tracks every change made to the project files over time. It keeps a history of what was modified, who made the changes, and when they were made. This makes it easy to understand the project's evolution and revert to earlier versions if needed.

Backup and restore: In case something goes wrong, like bugs introduced or files accidentally deleted, you can easily roll back to a previous state of the project.

TechPro

Version Control System? Why would you need one?

Versioning: With VCS, each change has a unique identifier (commit hash in Git), allowing the development team to refer to specific versions or releases of the project. This is crucial for managing production releases and debugging specific issues.

Branching and merging: VCS allows developers to work on different features or experiments in isolation (on branches) without affecting the main project. Once features are complete, the changes can be merged back into the main project safely.

Audit trail: VCS provides a detailed record of who made which changes, which is useful for accountability, code reviews, and improving the quality of the code.

TechPro

Introduction to Git

Git is a powerful version control system that tracks changes in files over time.

It's a crucial tool for developers, allowing them to collaborate effectively, manage different versions of their projects, and revert to previous states if needed.

TechPro

Life before git?

Manual versioning: Developers would often make copies of files and append the date or version number in the filename, like project_v1_final_final_v2.doc, **final_of_the_I swear_this_time_is_the_final_pre_final_version_of_this_file**. This was error-prone and confusing as the project grew or if multiple developers worked on the same file simultaneously.

TechPro

Life before git?

Centralized VCS: Before Git, centralized VCS systems like CVS (Concurrent Versions System) and Subversion (SVN) were popular. These systems had a central repository, and developers had to check out files, make changes, and commit them back to the central server. This method was slower and less flexible:
Developers had to be online to make commits.

- Merging was often more complex because only one person could "lock" a file at a time in some systems, and conflicts could become hard to manage.
- History was tied to the central server, which made offline development challenging.

TechPro

Life before git?

Backup and merge challenges: Without a proper VCS, if multiple people were working on the same files, it was difficult to track and merge their changes. People often worked in silos, leading to issues when trying to integrate different parts of the project. You had to manually compare files, leading to mistakes and lost work.

The story of Git

Linus Torvalds, known for his programming prowess and uncompromising standards, decided to write his own version control system.

He drew inspiration from various existing systems, but also introduced several innovative features.

Key aspects of Git's design

Distributed nature: Git operates on a distributed model. Each developer has a complete copy of the repository, allowing for offline work and greater flexibility.

Speed and efficiency: Git is renowned for its speed and efficiency, especially when dealing with large repositories. This is due to its data structure and algorithms.

TechPro

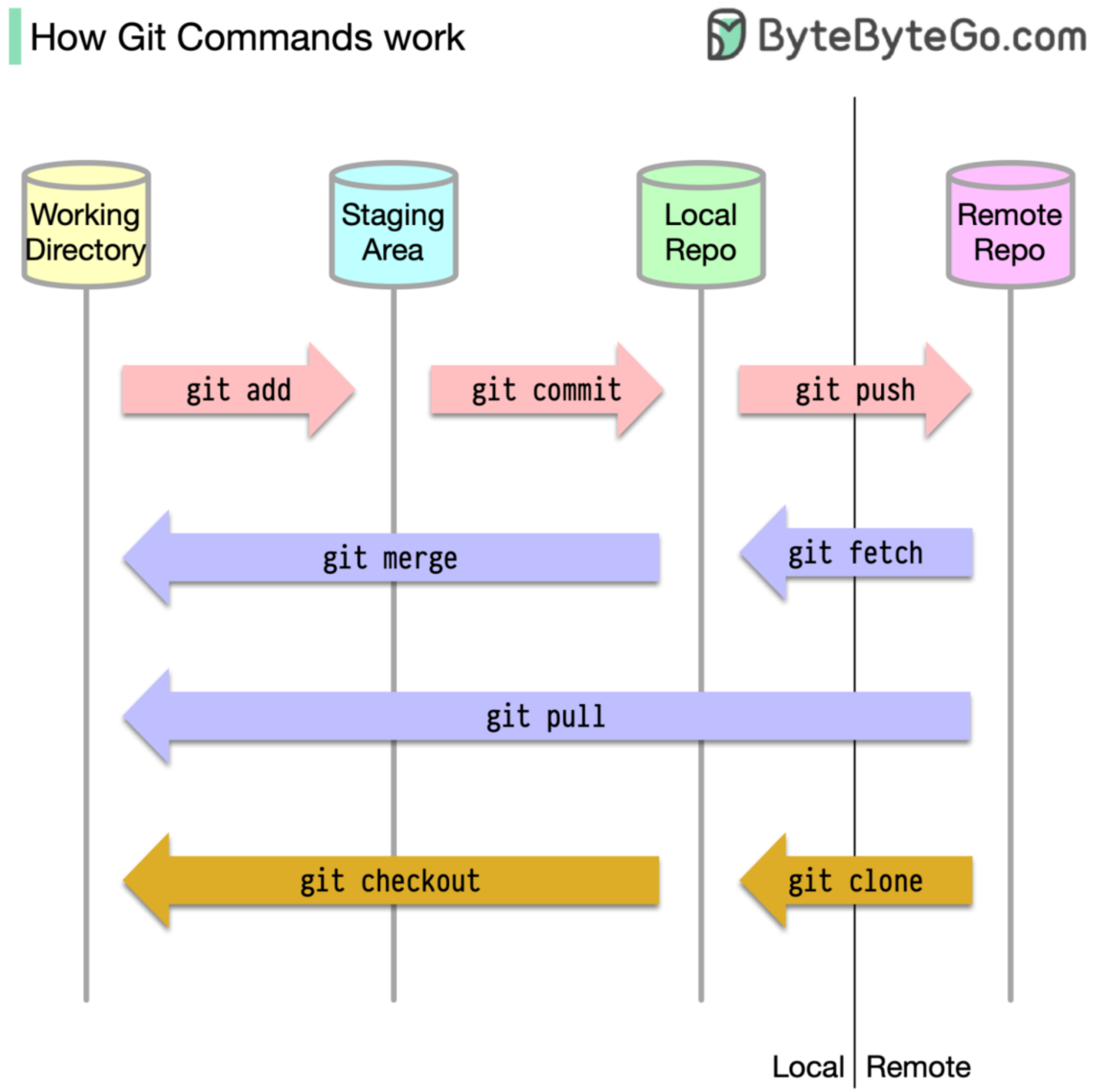
Key aspects of Git's design

Branching and merging: Git's branching model is powerful and encourages experimentation. Creating and merging branches is fast and easy, making it ideal for parallel development.

Content-based storage: Git stores files based on their content rather than their filenames, allowing for efficient tracking of changes.

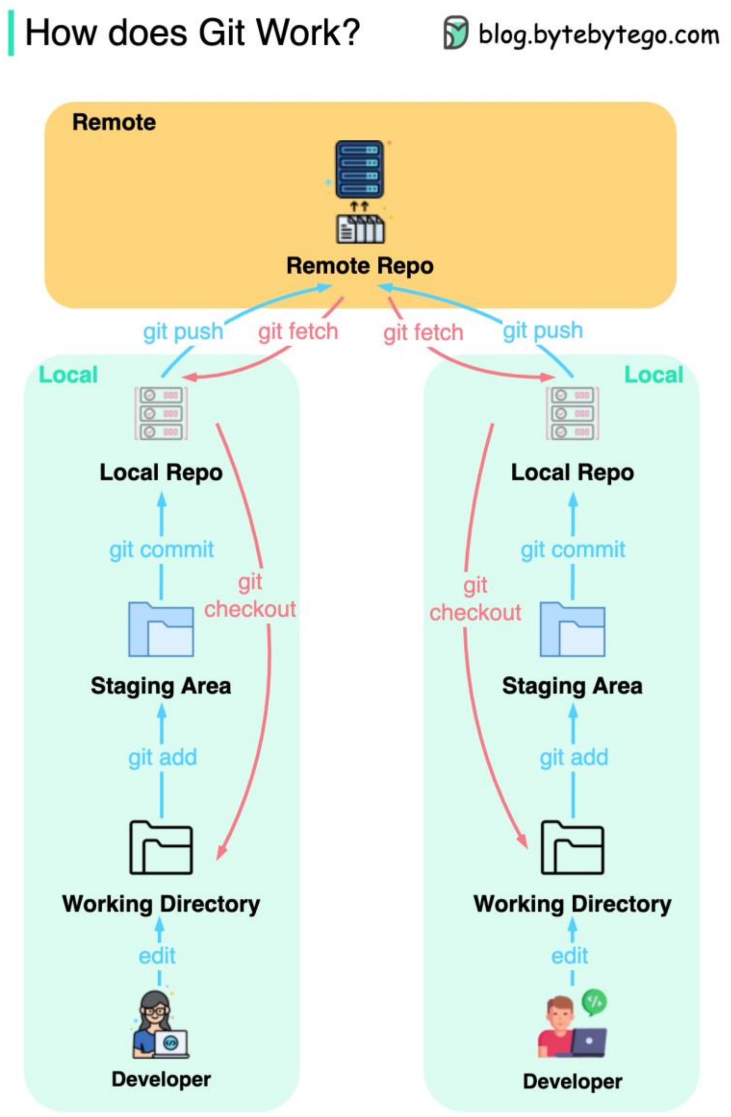
TechPro

Git Architecture Diagram



TechPro

Git Architecture Workflow



TechPro

Git Architecture Diagram

https://www.youtube.com/watch?v=e9lnsKot_SQ

TechPro

Benefits of Using Git

Version Control: Track changes over time and revert to previous states.

Collaboration: Work effectively with teams on the same project.

TechPro

Benefits of Using Git

Branching and merging: Experiment with different features without affecting the main codebase.

Distributed version control: Each developer has a local copy of the repository.

TechPro

Install and Configure Git

https://git-scm.com/downloads/win

Common Git Commands

git init: Create a new Git repository.

git add <filename>: Add files to the staging area.

git status: Show the current status of the working directory.

git clone <url>: Clone an existing repository.

TechPro

Common Git Commands

git commit -m "<message>": Commit changes to the repository.

git push <remote> <branch>: Push changes to a remote repository.

git pull <remote> <branch>: Pull changes from a remote repository.

TechPro

Common Git Commands

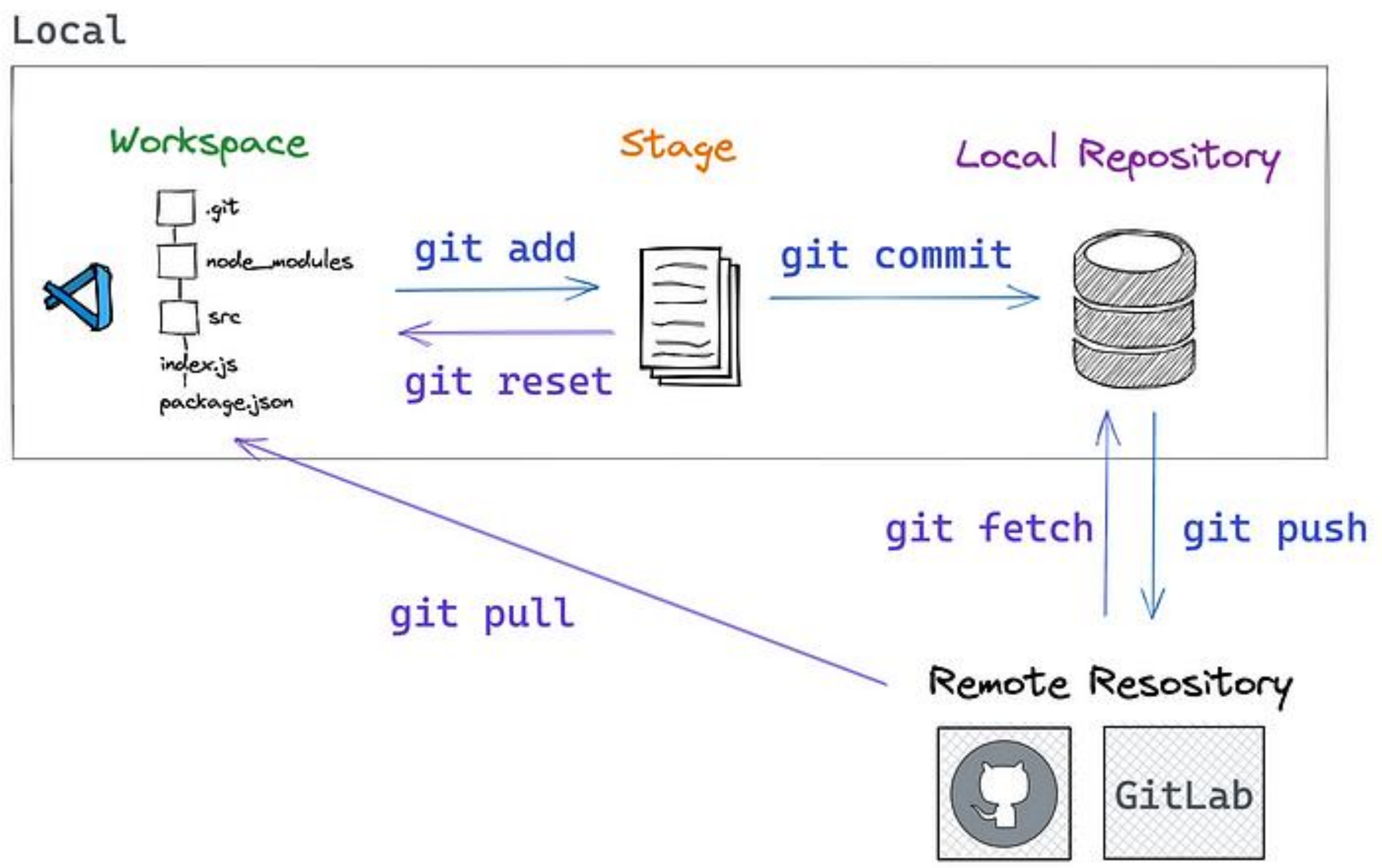
git branch: List branches.

git checkout <branch>: Switch to a different branch.

git merge <branch>: Merge changes from one branch into another.

TechPro

Visually!



TechPro

Summing up!

Command	Use
git init	Initialises a Git repository in that directory
git add .	Adds all changes to the staging area to be committed
git add file_name	Adds changes to the specified file to the staging area to be committed
git commit	Commits staged changes and allows you to write a commit message
git checkout SHA	Checks out a past commit with the given SHA
git checkout SHA -- file_name	Checks out the past version of a file from the commit with the given SHA
git checkout -b branch_name	Creates and switches to a new branch
git checkout branch_name	Switches to the specified branch
git merge branch_name	Merges the branch you are on into the specified branch
git log	Outputs a log of past commits with their commit messages
git status	Outputs status, including what branch you are on and what changes are staged
git diff	Outputs the differences between the working directory and most recent commit
git diff thing_a thing_b	Outputs the differences between two things, such as commits and branches
git clone URL	Makes a clone of the repository at the specified URL
git remote add origin URL	Links a local repository and an online repository at the specified URL
git push origin branch_name	Pushes local changes to the specified branch of the online repository
git pull origin branch_name	Pull changes from the online repository into local repository

TechPro

Development Environment

A developer's environment refers to the setup of hardware, software, and tools that a developer uses to create, test, and maintain software applications.

TechPro

Development Environment

Hardware: This can range from a simple laptop or desktop computer to a powerful workstation.

Operating system: The software that controls the computer's hardware and resources. Popular options include Windows, macOS, and Linux.

TechPro

Development Environment

Development Tools: These include text editors, compilers, debuggers, version control systems, and other software that assists in the development process.

Programming Languages: The languages that developers use to write code. Examples include Python, Java, C++, and JavaScript.

TechPro

Development Environment

Libraries and frameworks: These are pre-written code modules that can be used to simplify development tasks.

Integrated Development Environment (IDE): A software application that combines many of the tools mentioned above into a single interface.

Version Control System (VCS): A tool that tracks changes to code over time and allows developers to collaborate effectively.

TechPro

Integrated Development Environment

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities for software development. It typically consists of at least a source code editor, build automation tools, and a debugger.

TechPro

Key features of an IDE

Source code editor: A text editor that can assist in writing software code with features like syntax highlighting, code completion, and error checking.

Build automation: Tools that automate the process of compiling, linking, and packaging software.

TechPro

Key features of an IDE

Debugger: A tool that helps developers find and fix errors in their code.

Version control integration: Integration with version control systems like Git to manage code changes.

Project management: Tools to organize and manage software projects.

TechPro

Popular IDEs

Visual Studio Code: A powerful IDE, supporting almost all languages with a massive extension market

IntelliJ IDEA: A commercial IDE for Java, Kotlin, and other languages, known for its advanced features.

TechPro

Popular IDEs

Eclipse: A popular open-source IDE for Java development, also supporting other languages.

Xcode: The official IDE for macOS and iOS development, supporting Swift and Objective-C.

Visual Studio Code

VS Code stands for Visual Studio Code. It's a popular and free integrated development environment (IDE) developed by Microsoft.
It's designed to make coding more efficient and enjoyable, especially for web development and programming with languages like JavaScript, TypeScript, Python, C++, and more

TechPro

Install and Configure Visual Studio Code

https://code.visualstudio.com/docs/setup/windows

TechPro

Jupyter Notebooks

Jupyter Notebooks are an open-source web-based tool that allows users to create and share documents containing live code, equations, visualizations, and narrative text.

Why Jupyter Notebooks?

Jupyter Notebooks provide a flexible, interactive environment that simplifies coding, data analysis, and documentation.

TechPro

Key Features

- Code Execution
- Notebook Structure
- Markdown Support
- Interactive Outputs
- Visualizations
- Exporting

TechPro

Common Use Cases

- ➢ Data Analysis and Visualization
- ➢ Prototyping and Experimentation
- ➢ Research
- ➢ Education

TechPro

Vs Code Jupyter VS Code Extension Tutorial

https://code.visualstudio.com/docs/datascience/jupyter-notebooks

TechPro

By the End of this Course

1. Understand the need and use of Git
2. Install and Configure Git
3. Run most common used git commands
4. Understand the need and use of a centralized development environment
5. Install and be familiar with IDEs
6. You use VS Code as your development environment
7. The powerful extension market of VS Code
8. Connect VS code with git and start programming
9. Jupyter notebooks the need, the use and Markdown syntax

TechPro

Homework

Take the Quiz
https://www.w3schools.com/git/git_quiz.asp

Find:
The most common VS code extension for opinionated code format!

What is the Github repository of the extension?

All links in → https://linktr.ee/xpaulos

Feedback Discussion

TechPro

Thank you

TechPro