HyperionDev

Welcome to this session:
Introduction to Git and Version Control

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators answering questions.

HyperionDev

What is Safeguarding?

Safeguarding refers to actions and measures aimed at protecting the human rights of adults, particularly vulnerable individuals, from abuse, neglect, and harm.

To report a safeguarding concern reach out to us via email:

safeguarding@hyperiondev.com

HyperionDev

Live Lecture Housekeeping:

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - ask them!
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- To report a safeguarding concern reach out to us via email: safeguarding@hyperiondev.com
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

HyperionDev

Lecture Overview

Learning Outcomes

- **Identify** the basic concepts of version control and Git.
- **Explain** the purpose and benefits of version control systems.
- **Describe** the basic commands and operations in Git.
- **Initialise** a Git repository.
- **Stage and commit** changes to a repository.
- **Assess** the impact of version control on collaboration.

Version Control: The "Time Machine" for Code

HyperionDev

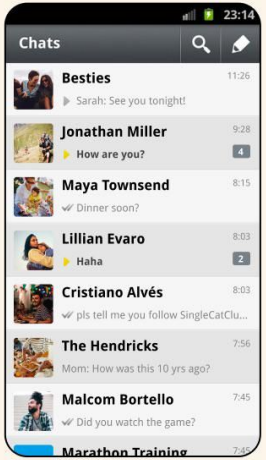
Relevance

- Ever wondered how apps like Instagram and WhatsApp keep getting better without breaking?
- How do developers manage to add new features and fix bugs without chaos?
- The answer lies in a powerful tool called **Version Control**.

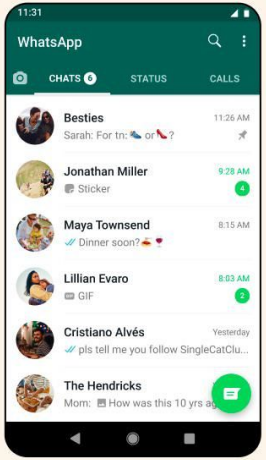
Relevance

Design over the years

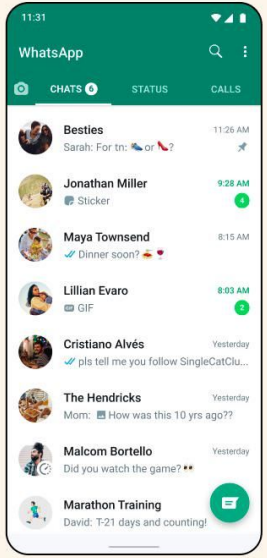
2011



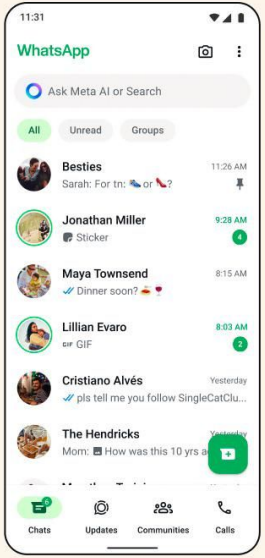
2012



2020



2021



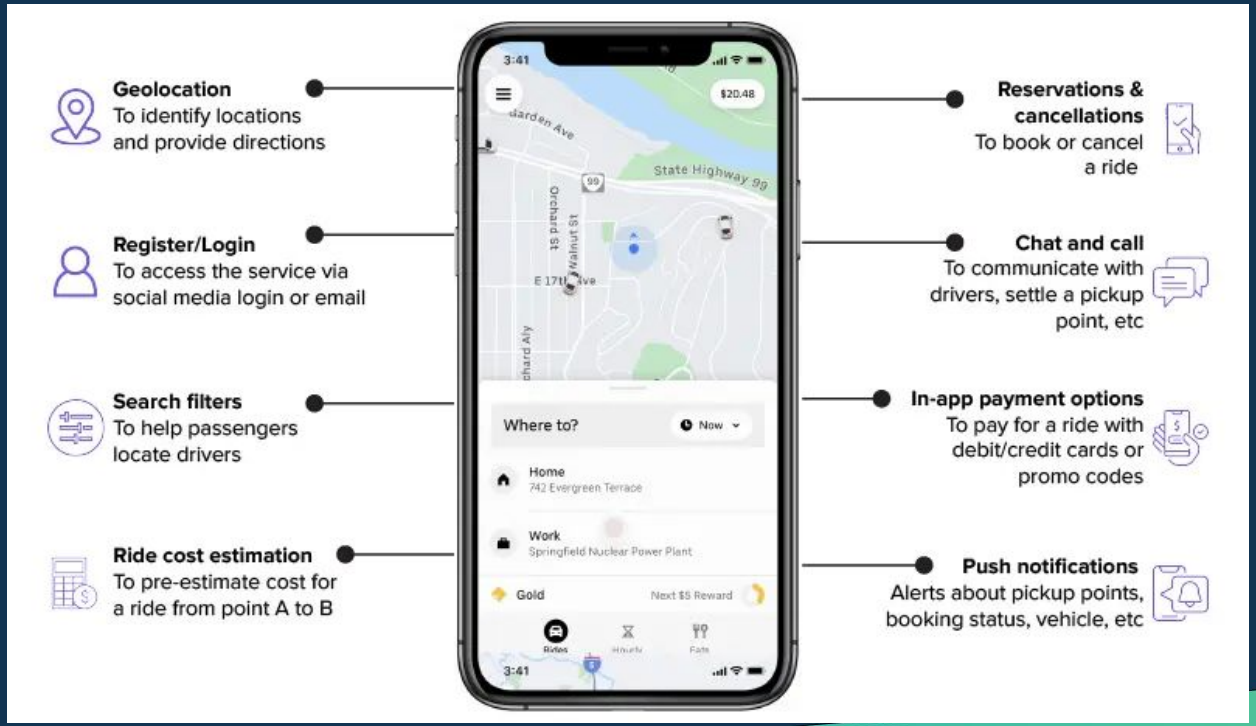
2024

HyperionDev

Real-World Example

- Each update you see has a team of developers behind it, each contributing their part. But how do they track who changed what? And if something goes wrong, how do they roll back to a previous version?

Relevance



HyperionDev

Introduction to Version Control

Just as you can track changes in a Google Doc, developers use Version Control to manage and track code changes.

The Essentials: Version Control 101

HyperionDev

Meet the Squad: Git Lingo 101!

- **Repository:** The central storage for your project.
- **Commit:** A snapshot of your project at a specific point in time.
- **Branch:** A parallel version of your project.
- **Merge:** Combining changes from different branches.
- **Clone:** Copying an existing repository to your local machine.
- **Working Directory:** Where you make changes to your files.
- **Staging Area:** A temporary holding area for changes before committing.

Theoretical Demo

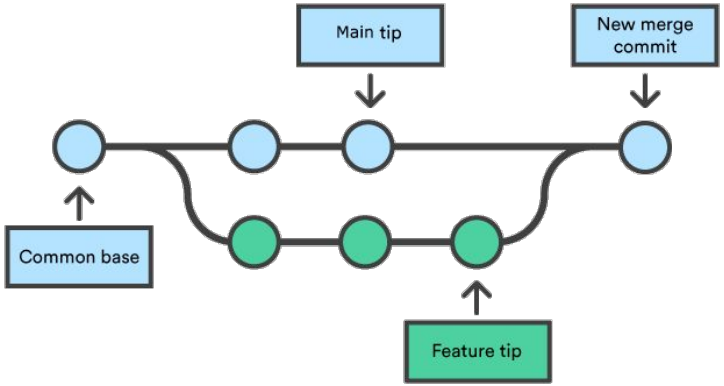
HyperionDev

The Version Control Journey

- **Modified:** Changes made to files in the working directory.
- **Staged:** Files are added to the staging area, preparing them for the next commit.
- **Committed:** Changes are saved in the repository as a new version snapshot.

Branching Out & Coming Back

- **Branching:** Experiment without changing the main code.
- **Merging:** Combine your changes back to the main project.



HyperionDev

BREAK

HyperionDev

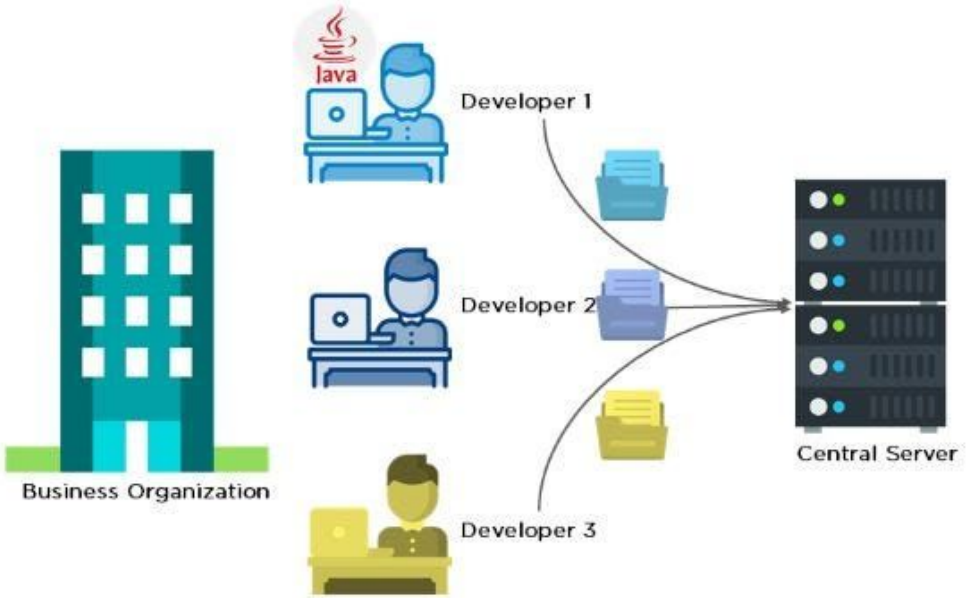
Introduction to Git

HyperionDev

What is Git? Disclaimer

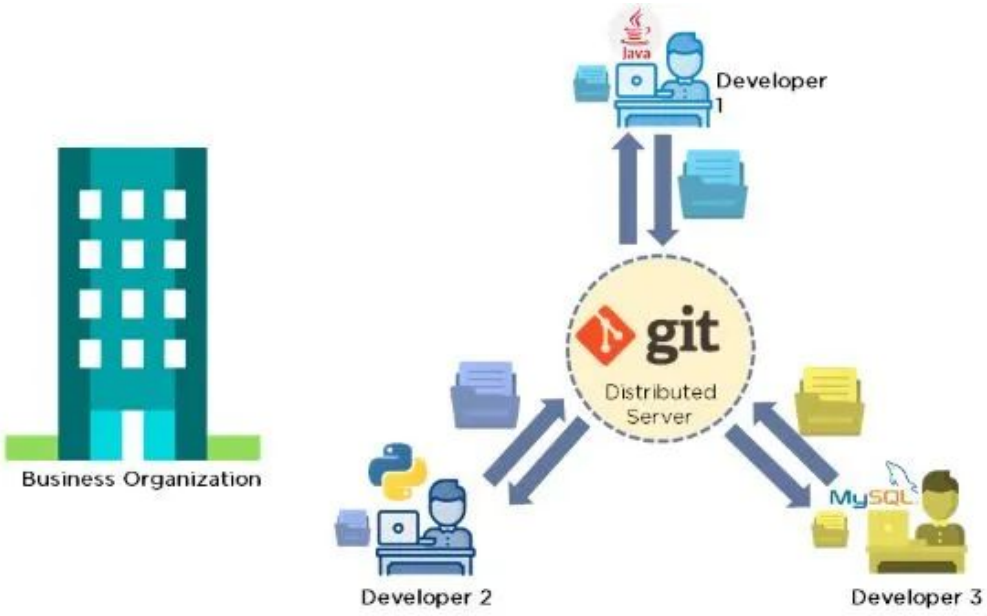
HyperionDev

What is Git?



HyperionDev

What is Git?



HyperionDev

What is Git?

- A powerful version control system
- Tracks changes to your code over time
- Enables collaboration with other developers
- It’s a distributed system, so every developer has a full copy.
- **Why Git?**
  - **Distributed nature:** Work offline and sync later
  - **Branching and merging:** Experiment without risk
  - Strong community and support

Your Git Toolkit

- **Repository Operations:**
  - **git init:** Create a new repository
  - **git clone:** Clone an existing repository

Your Git Toolkit

- **Working with Changes:**
  - **git status:** Check the status of your files
  - **git diff:** View changes between commits
  - **git add:** Stage changes for commit
  - **git commit:** Commit changes to the repository
  - **git push:** Push changes to a remote repository
  - **git pull:** Pull changes from a remote repository

Real-world Context and Conclusion

Real-World Git: From Code to Collaboration!

- **Efficient Collaboration:** Teams work together seamlessly on shared codebases.
- **Risk Mitigation:** Backups, version history, and easy rollback.
- **Continuous Integration/Continuous Delivery (CI/CD):** Automated testing and deployment.
- **Open Source Development:** Fostering community and collaboration.

HyperionDev

The Big Three: Where the Magic Happens!

GitHub

GitLab

Bitbucket

HyperionDev

Best Practices: Keep Calm and Commit On!

- **Commit Frequently:** Small, focused commits.
- **Write Clear Commit Messages:** Describe the changes made.
- **Use Branches Effectively:** Isolate features and bug fixes.
- **Review Code Regularly:** Improve code quality and collaboration.
- **Utilize Pull Requests:** A structured review process.
- **Automate Your Workflow:** Use CI/CD pipelines.

Lesson Conclusion and Recap

- **Defining Version Control**: Explain version control purpose and analyse its necessity in software development.
- **Applying Git Terminology**: Define essential Git concepts and illustrate terms like repository, commit, branch, merge, working directory, and staging area. Classify each team's role in version control workflow.
- **Executing Basic Git Commands**: Identify and describe fundamental Git commands for version control. List and differentiate between commands like git init, clone, status, diff, add, commit, push, and pull, demonstrating their purpose and basic usage.

HyperionDev

Practical: Basic Git and GitHub Setup and Workflow

1. **Objective:** Create a repository, make changes, and understand how to push these changes to GitHub. This exercise will help to have a basic understanding of how to initialise a repository, make commits, and work with GitHub as a remote.
2. **Steps to Implement:**
    - Set Up and Initialize Git
    - Create a Local Repository
    - Create and Add Files to the Repository
    - Commit Changes
    - Push Changes to GitHub
    - Verify and View Changes on GitHub

HyperionDev

Resources

- **Additional Resources**
  - 1.5 Getting Started - Installing Git
  - Pro Git book

HyperionDev

Poll

1. **Which of the following is a benefit of using Git for version control?**
   a. It allows multiple developers to work on the same project simultaneously.
   b. It automatically deploys code to production servers.
   c. It replaces the need for a Terminal.

Poll

2. **What is the correct sequence of steps to push code to a remote Git repository?**

   a. Stage changes → Commit changes → Push to remote repository.

   b. Push to remote repository → Stage changes → Commit changes.

   c. Commit changes → Push to remote repository → Stage changes.

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.


Thank you
for attending
HyperionDev