Welcome to this CoGrammar Tutorial:

Version Control Practice

The session will start shortly...

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

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** throughout this session, should you wish to ask any follow-up questions.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Enhancing Accessibility: Activate Browser Captions

Why Enable Browser Captions?

- Captions provide **real-time text for spoken content**, ensuring inclusivity.
- Ideal for individuals in noisy or quiet environments or for those with **hearing impairments**.

How to Activate Captions:

1. **YouTube or Video Players:**
   - Look for the CC (Closed Captions) icon and click to enable.
2. **Browser Settings:**
   - Google Chrome: Go to *Settings* > *Accessibility* > *Live Captions* and toggle ON.
   - Edge: Enable captions in *Settings* > *Accessibility*.

CoGrammar

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

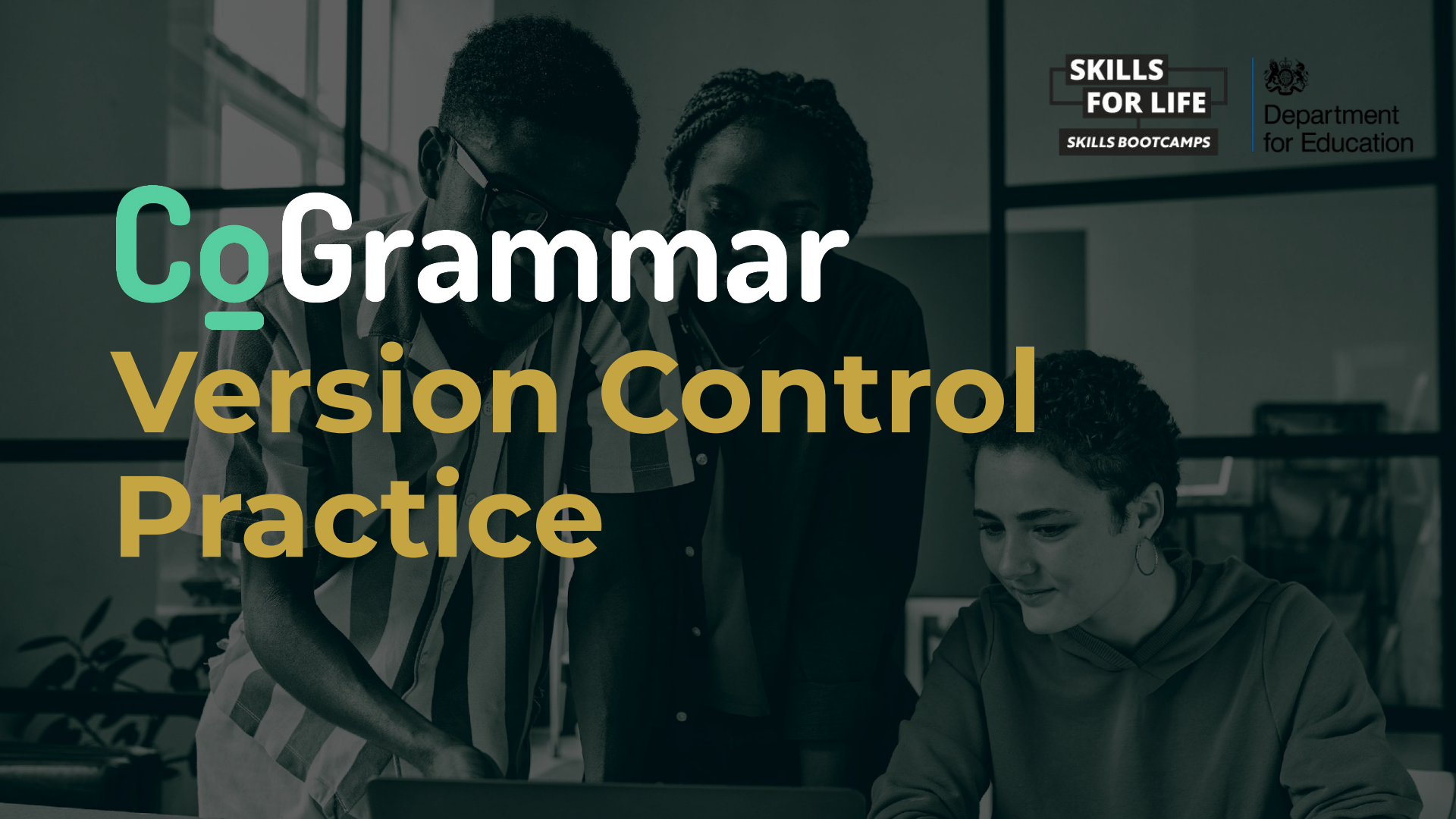
Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev


SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education
CoGrammar
Version Control Practice

Learning Objectives & Outcomes

- **Explain the purpose and benefits of version control systems** in the context of individual and team-based software development.
- **Perform basic Git operations**, including repository initialization, staging, committing, and viewing commit histories.
- **Implement branching workflows**, such as feature branching, and merge changes back into the main branch while handling conflicts.
- **Collaborate effectively using Git and remote repositories**, including cloning, pushing, pulling, and submitting/merging pull requests.
- **Apply best practices for maintaining a clean project history**, such as writing meaningful commit messages, and using Git commands to resolve issues.

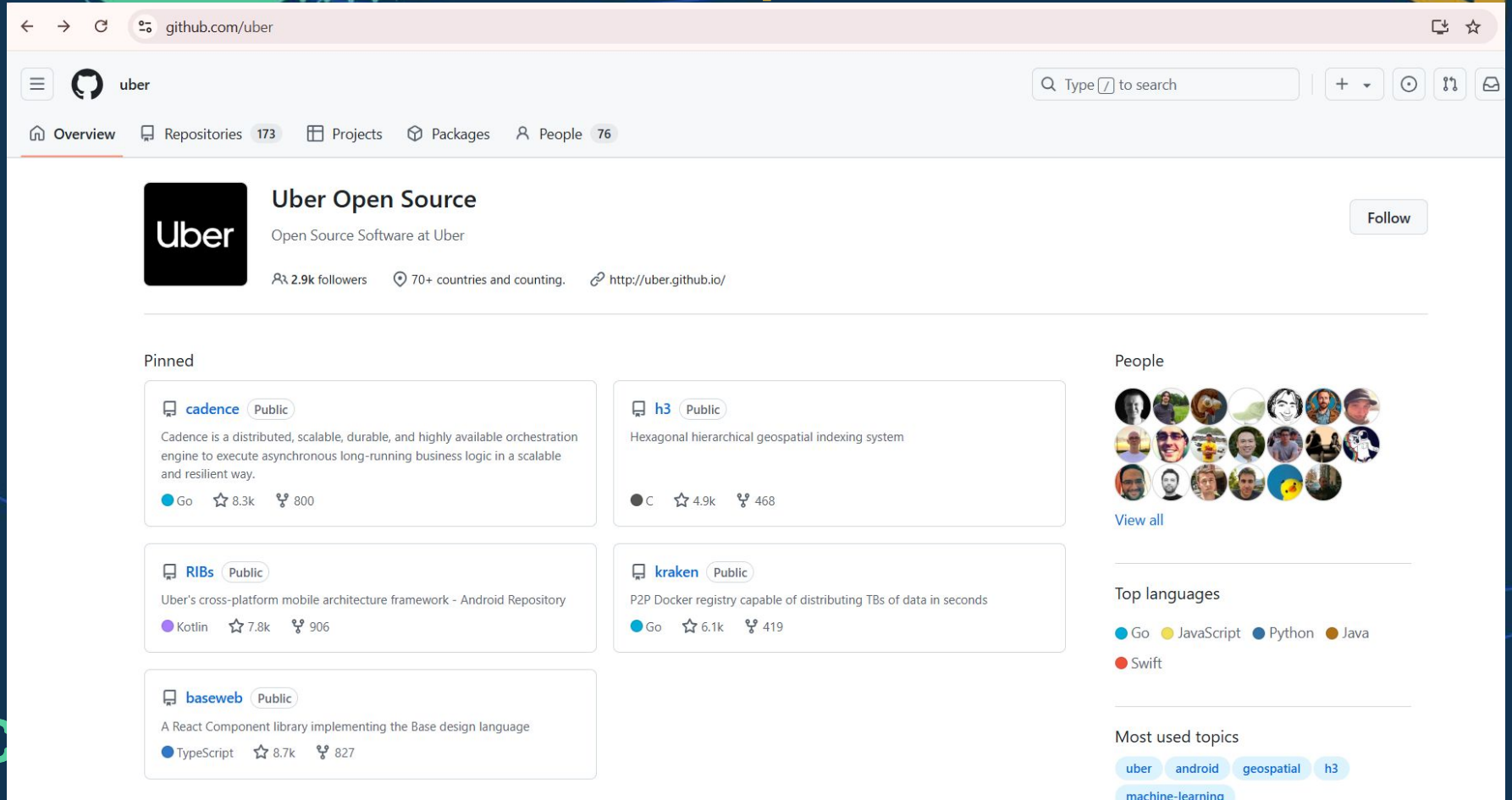
Git in High-Scale Software Development

CoGrammar

Git in High-Scale Software Development

Companies like **Netflix** and **Uber** handle **thousands of code updates daily** across vast, distributed teams. With **hundreds of developers** contributing to their systems, pull requests are reviewed and merged continuously, sometimes reaching **hundreds per day**. Teams rely on Git to manage frequent updates, resolve merge conflicts efficiently, and ensure code stability in real-time. Netflix, for example, deploys new code to production **hundreds of times per day**, leveraging Git and CI/CD pipelines for smooth rollouts. Similarly, Uber's microservices architecture involves **thousands of repositories**, where Git ensures synchronisation and conflict resolution across all services.

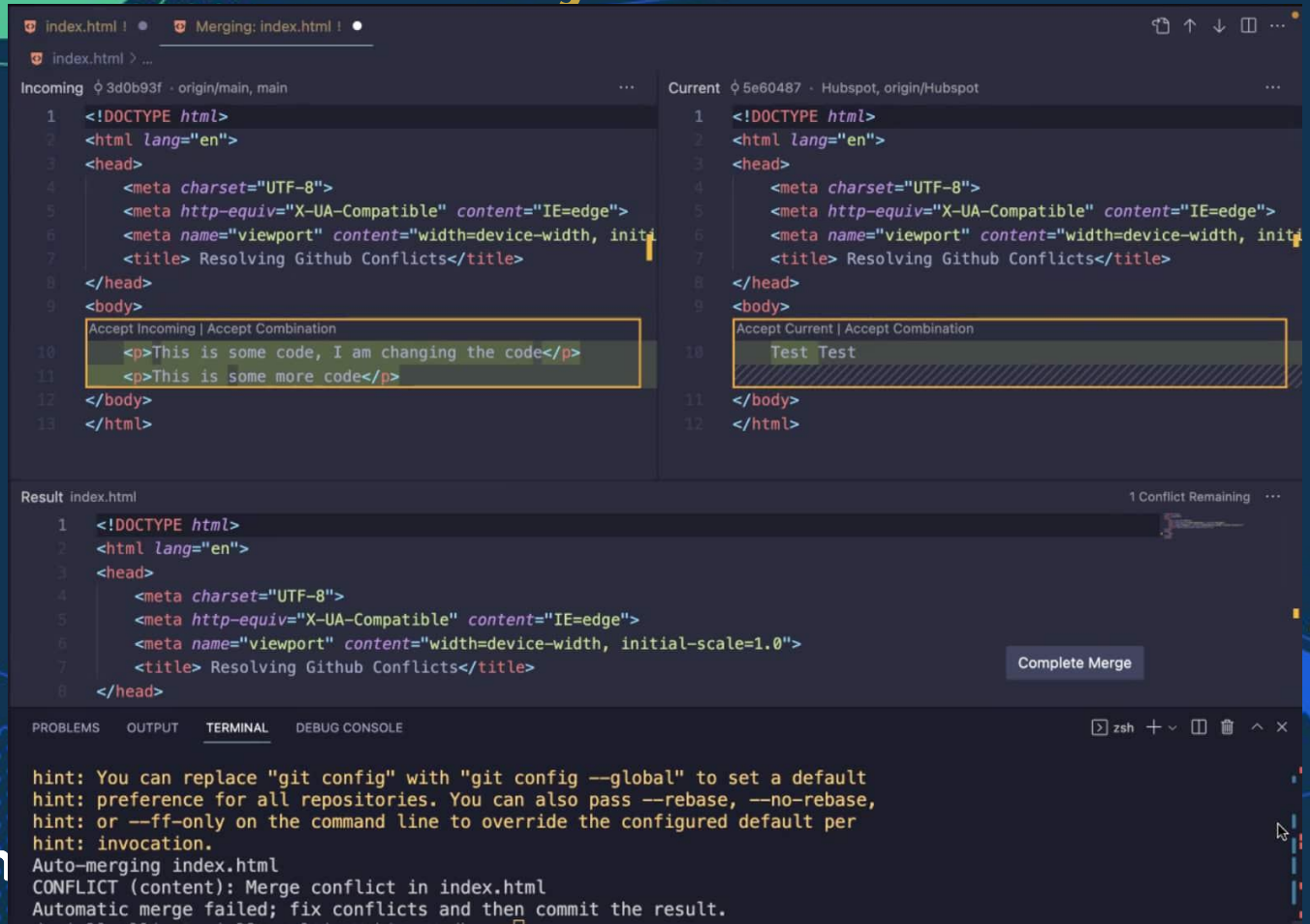
Git in High-Scale Software Development



Introduction to Git and Version Control

CoGrammar

Track, Manage, and Collaborate Effectively on Codebases



CoGramm

What is Version Control?

- **Track Changes**
  - Like a time machine for your code
  - See who changed what and when
- **Collaborate**
  - Multiple developers, one codebase
  - No more "*final_final_v2.py*"

Why is Git the Most Popular?

- Distributed version control: Work locally and sync with remote repositories.
- Widely adopted by platforms like GitHub, GitLab, and Bitbucket.
- Robust branching and merging capabilities.

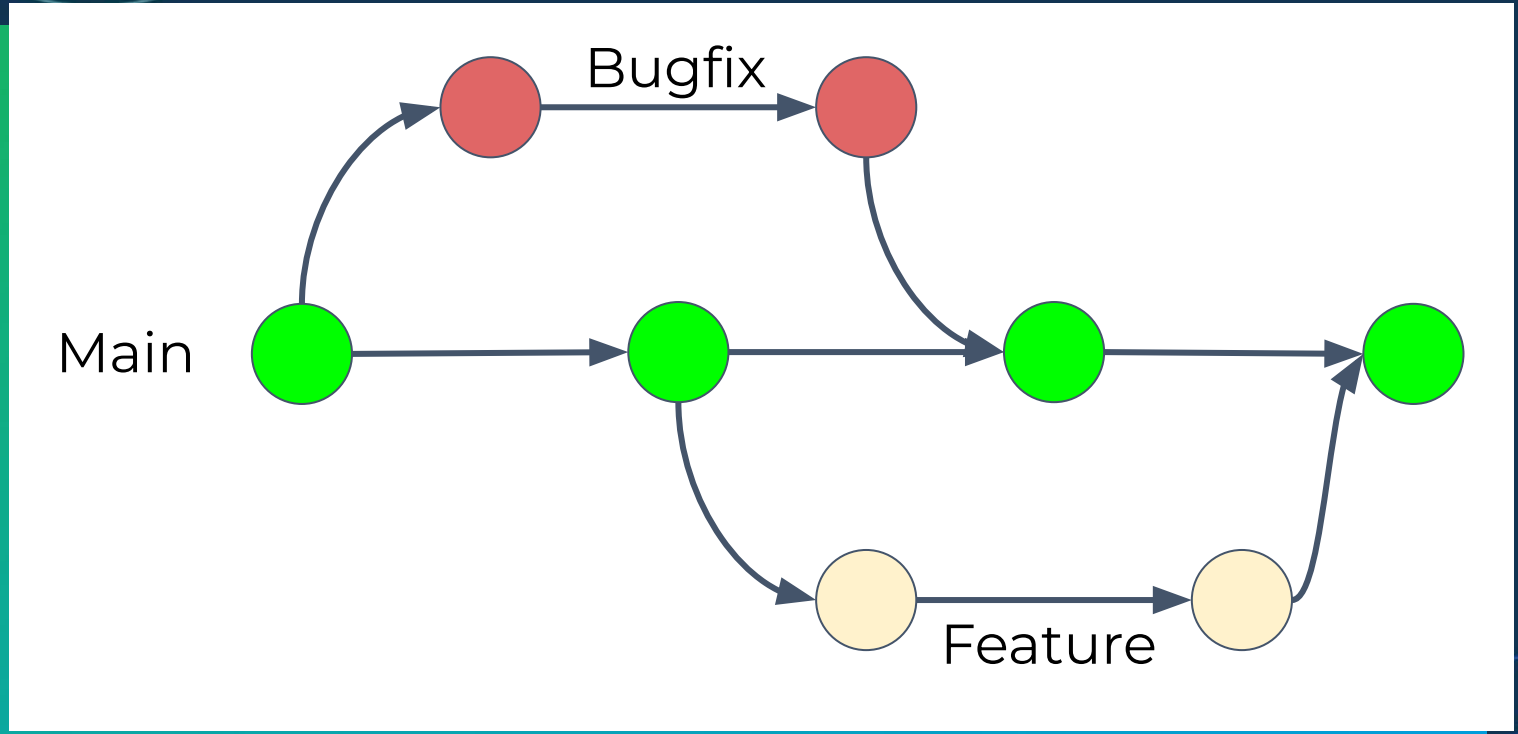
Key Commands and Concepts

CoGrammar

Key Git Concepts

- **Repository**: A storage for your project.
- **Commit**: A snapshot of changes.
- **Branch**: An independent line of development.
- **Merge**: Combining branches.

Branching Strategies: A Visual Guide



Essential Git Commands 💻

```
# Initialize a repository
git init

# Stage changes
git add filename
git add .  # all files

# Commit changes
git commit -m "Your message"

# View history
git log --oneline
```

Branching: Your Development Pipeline 🌳

```
# Create & switch to new branch
git checkout -b feature-name

# List branches
git branch

# Switch branches
git checkout branch-name
```

Essential Git Commands

- Initialize a Repository: `git init`
- Track Changes: `git add`, `git commit`
- View History: `git log`
- Branching and Merging: `git branch`, `git merge`
- Remote Collaboration: `git push`, `git pull`, `git clone`

The Role of .gitignore

- Exclude unnecessary or sensitive files from version control.
- Examples: `node_modules/`, `*.log`, `.env`.

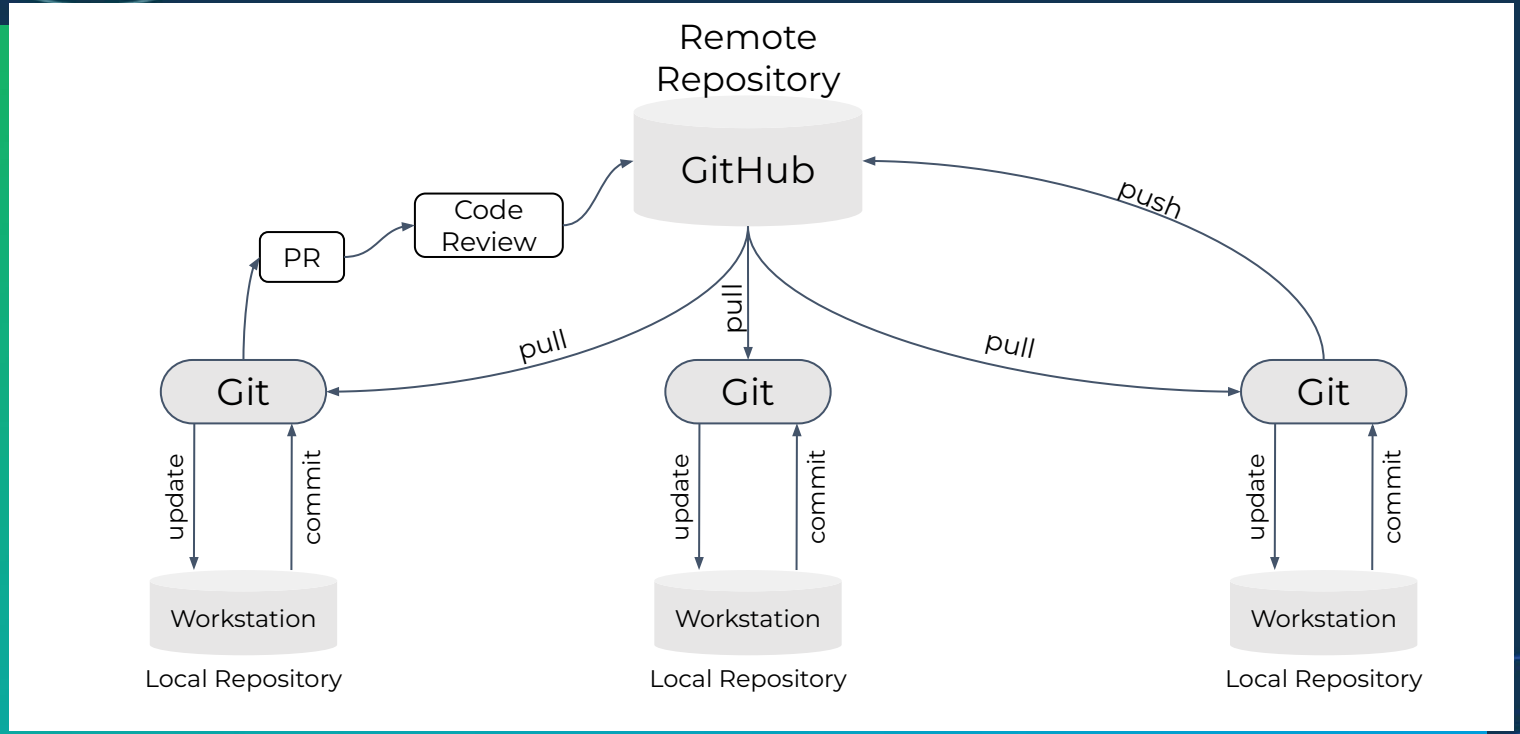
GitHub Overview

CoGrammar

GitHub Workflow

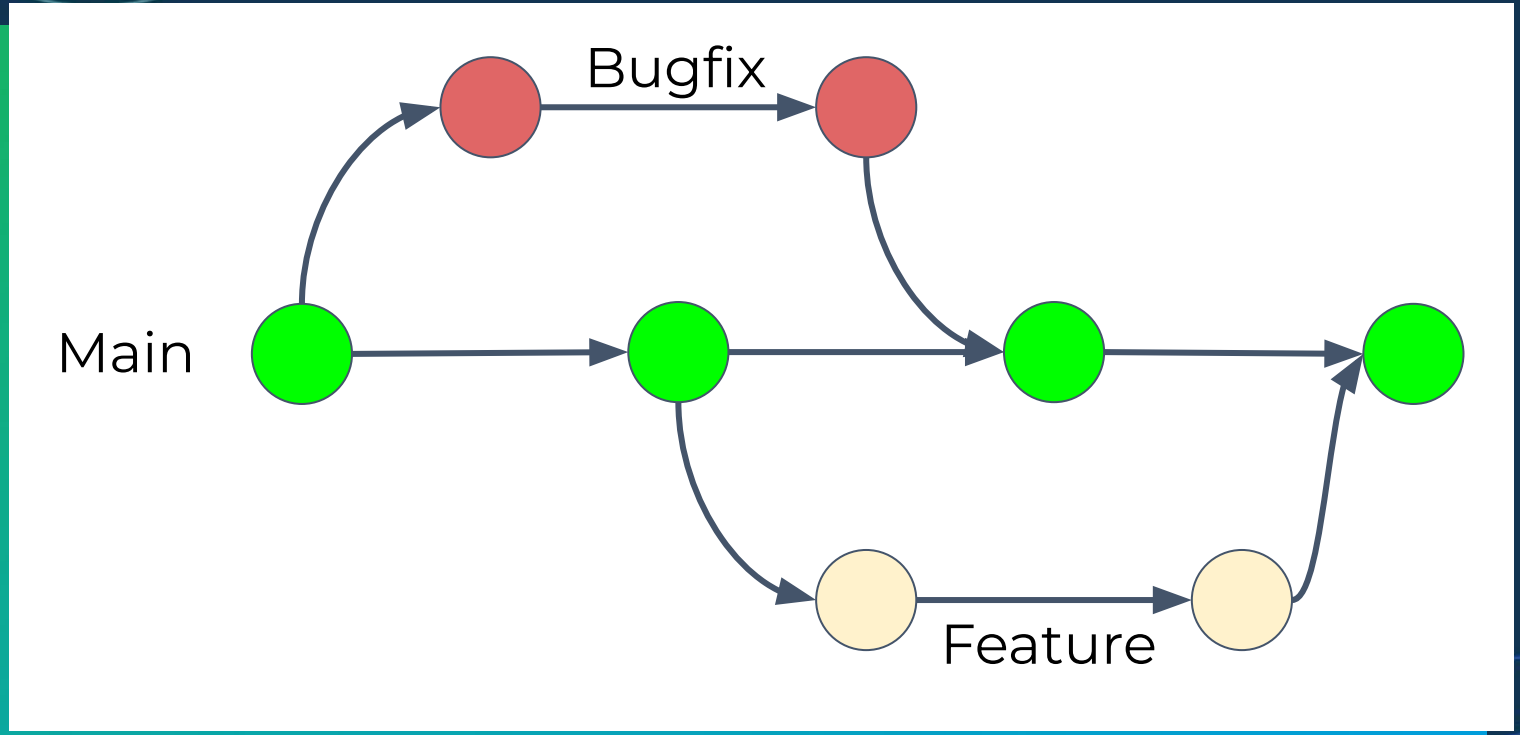
1. Fork repository
2. Clone locally
3. Create branch
4. Make changes & commit
5. Push to your fork
6. Create Pull Request

Pull Requests and Code Reviews

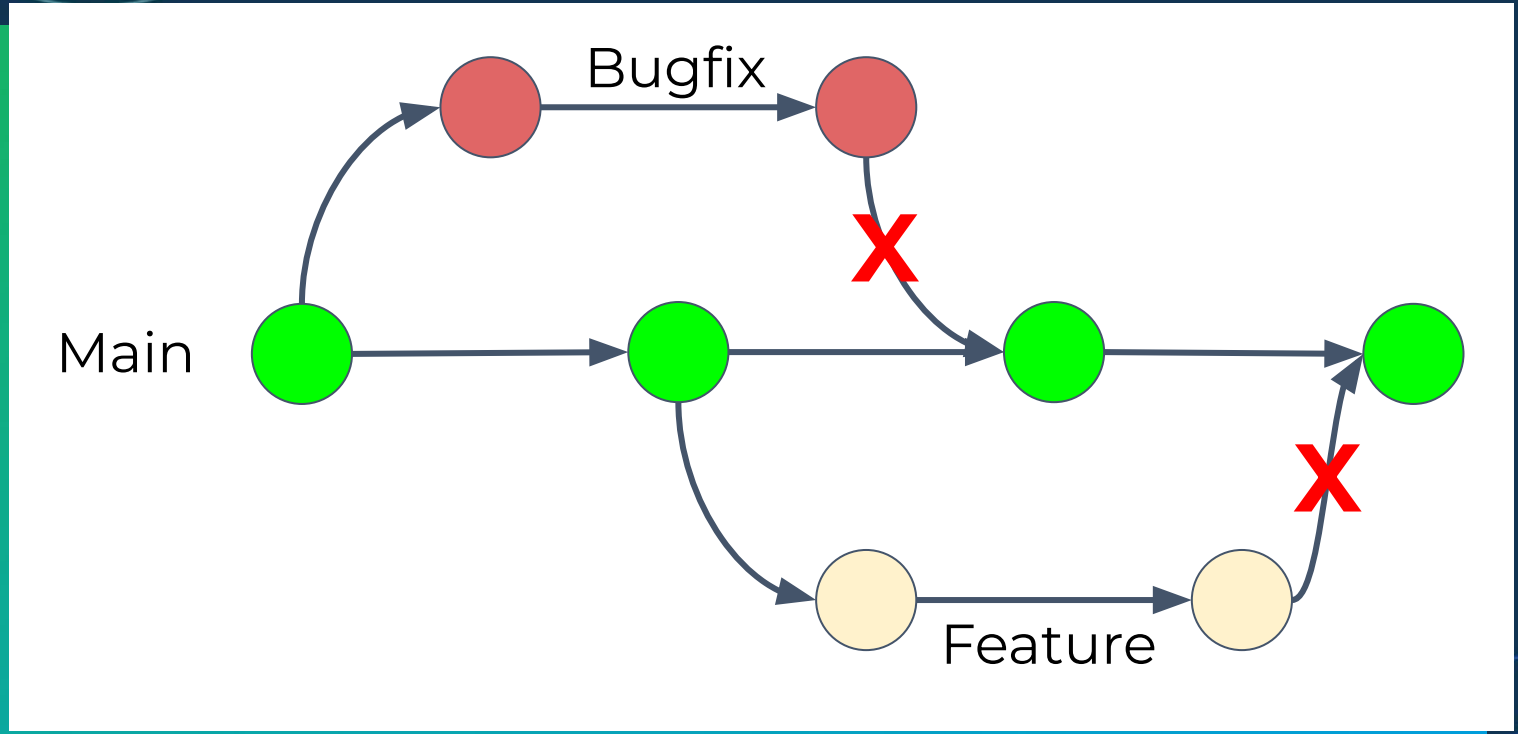


CoGrammar

What Are Branches in Git?



Handling Merge Conflicts



Handling Merge Conflicts

- **Why Do Merge Conflicts Happen?**
  - Two branches modify the same line in a file.
  - Changes are made to the same file in ways Git cannot automatically combine.
- **How to Resolve Conflicts:**
  1. Identify the conflict (Git will mark files with conflicts).
  2. Edit the file to keep the desired changes.
  3. Mark the conflict as resolved (`git add`).
  4. Complete the merge (`git commit`).
- **Tools to Simplify Conflict Resolution:**
  - VS Code Git integration.
  - GitHub's conflict resolution editor.

Best Practices & Tools

CoGrammar

Managing What Git Tracks with .gitignore

- **What is .gitignore?**
  - A file to tell Git which files or directories to ignore (not track).
  - Keeps sensitive or irrelevant files out of the repository.
- **Common Use Cases:**
  - Ignoring files like logs, temporary files, and environment variables.
  - Excluding OS or editor-specific files (e.g., `.DS_Store`, `*.swp`).
- **How to Use .gitignore:**
  - Add a `.gitignore` file at the root of your project.
  - Use patterns to specify ignored files (e.g., `*.log`, `/node_modules/`).
- **Best Practice:**
  - Always add `.gitignore` when initializing a project to avoid tracking unnecessary files.

Guidelines for Effective Code Reviews

- **What is Code Review?**
  - A systematic examination of code by peers to improve quality and ensure adherence to team standards.
- **Review Guidelines:**
  - Focus on the code, not the person.
  - Check for functionality, readability, and adherence to standards.
  - Ensure the code is well-tested.
- **Providing Constructive Feedback:**
  - Be specific: "Consider renaming this variable to make it clearer."
  - Be polite: "What if we refactor this function for better readability?"
  - Avoid negative or personal comments.

Using GitHub's Review Features Effectively

- **GitHub Review Features:**
  - Leave inline comments on specific lines of code.
  - Approve or request changes on pull requests.
  - Use suggestions for quick fixes.
- **Best Practices for Reviewers:**
  - Understand the feature's purpose before reviewing.
  - Test locally if necessary.
  - Avoid nitpicking minor issues unless they impact functionality.
- **Best Practices for Submitters:**
  - Write clear commit messages.
  - Use meaningful PR descriptions (what/why/how).
  - Address feedback promptly and update your PR.

Practical: Version Control for Collaborative Web Development

CoGrammar

Lesson Conclusion and Recap

Recap the key concepts and techniques covered during the lesson.

- **Branching Strategies:** Key strategies like feature branches and main branch usage help organise work and streamline collaboration.
- **Merging and Conflict Resolution:** Merging branches and handling merge conflicts ensure smooth integration of changes from different contributors.
- **Using .gitignore:** The .gitignore file helps manage which files to track or ignore, keeping the repository clean and focused.
- **Pull Requests (PRs) and Code Reviews:** PRs and code reviews support collaborative development, allowing team members to review, discuss, and improve code before merging.
- **Remote Repositories and GitHub Collaboration:** Leveraging platforms like GitHub enhances teamwork, making it easier to share, collaborate, and track project progress.

Resources

Resources

- **Software:**
  - Git - Downloading Package
  - Download GitHub Desktop
- **Additional Resources**
  - Hello World - GitHub Docs
  - Get started with GitHub documentation
- **Books:**
  - Pro Git book

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar