

Introduction to branches

INTERMEDIATE GIT



George Boorman

Curriculum Manager, DataCamp

What we will cover

- Branches
- Remotes
- Conflicts



What you should know

- How Git stores data
- How to create repos
- How to make commits
- How to compare versions
- How to revert versions

The screenshot shows the DataCamp learning platform. At the top, there's a navigation bar with links for Home, Learn (which is currently selected), Certification, DataLab, and a search bar containing the text 'git'. Below the navigation is a sidebar with various options: Progress, My Library, Leaderboard, Assignments, Tracks, Courses, Practice, Assessments, Tutorials, Real World Projects, Code Alongs, and Competitions. The main content area displays an 'INTERACTIVE COURSE' titled 'Foundations of Git'. It features a 'Continue' button and a 'Bookmark' button. Below the title, it says 'Beginner', '2 hours', '8 videos', '22 exercises', '657 participants', and '1650 XP'. A 'Description' section follows, which includes a detailed text about the course's purpose and what participants will learn.

INTERACTIVE COURSE
Foundations of Git

Continue Bookmark

Beginner 2 hours 8 videos 22 exercises 657 participants 1650 XP

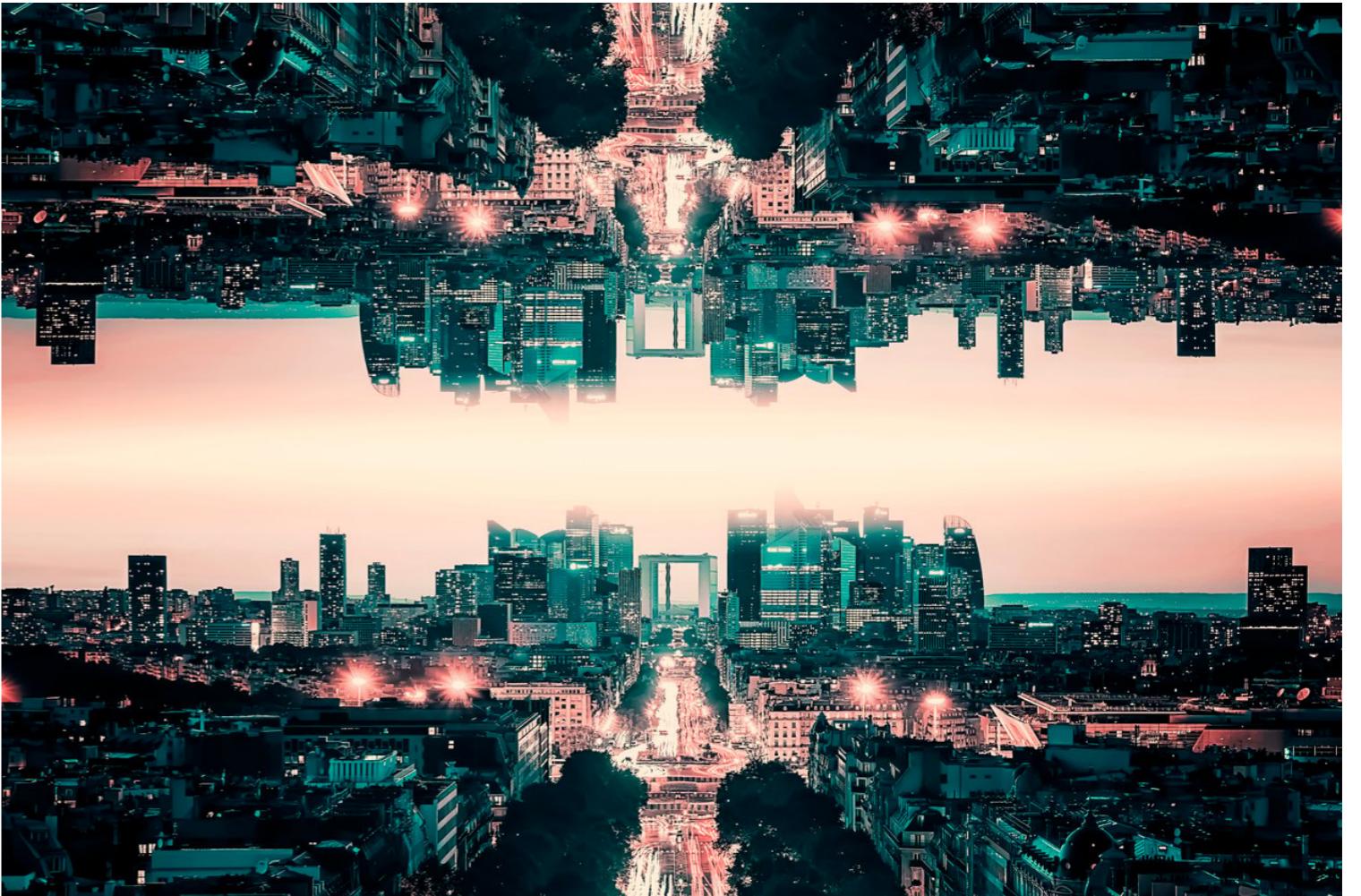
Description

This course introduces the concept of version control and highlights its benefits for software and data projects. You'll learn about Git, the popular software for implementing version control in projects, and use it to create repositories and track files.

Discover how Git stores data through blobs, trees, and commits. Use this information to compare the state of your project at different points in time, understanding what changes have been made, by who, and when. Learn valuable tips and tricks to customize your view of a repository's history and how to undo changes to revert files!

Branches

- Branch = an individual **version** of a repo
- Git uses **branches** to systematically track multiple versions of files
- In each branch:
 - Some files might be the same
 - Others might be different
 - Some may not exist at all



Why use branches?

Live system

Feature development

- Works as expected
- Default branch = `main`
- Might encounter issues during development and testing
- Doesn't affect the live system

Why use branches?

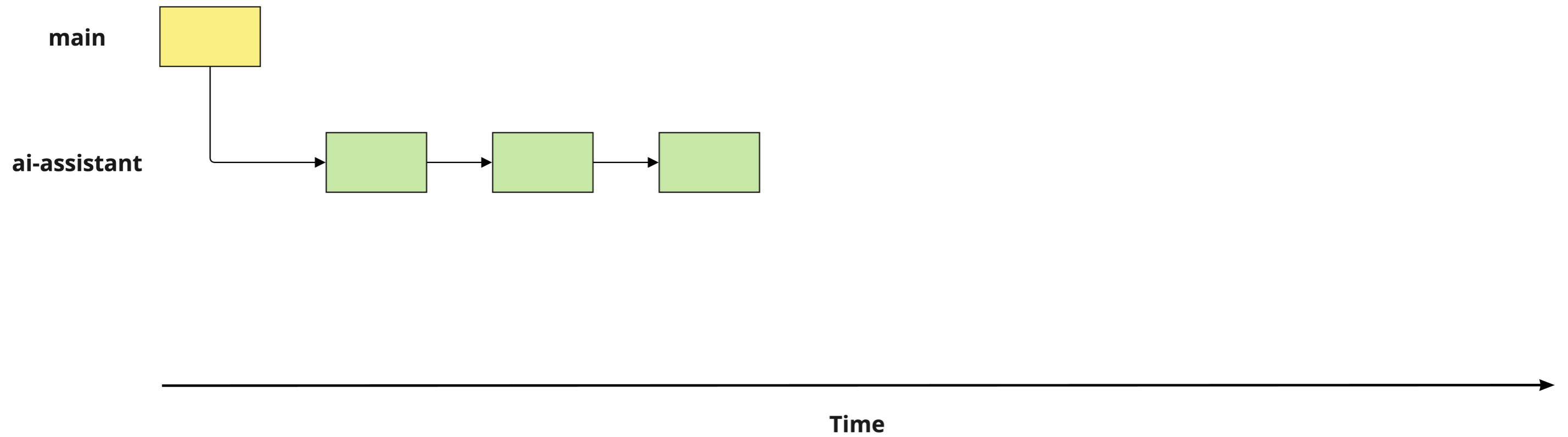
- Multiple developers can work on a project **simultaneously**
- **Compare the state** of a repo **between branches**
- **Combine contents**, pushing new features to a live system
- Each branch should have a **specific purpose**

Visualizing branches

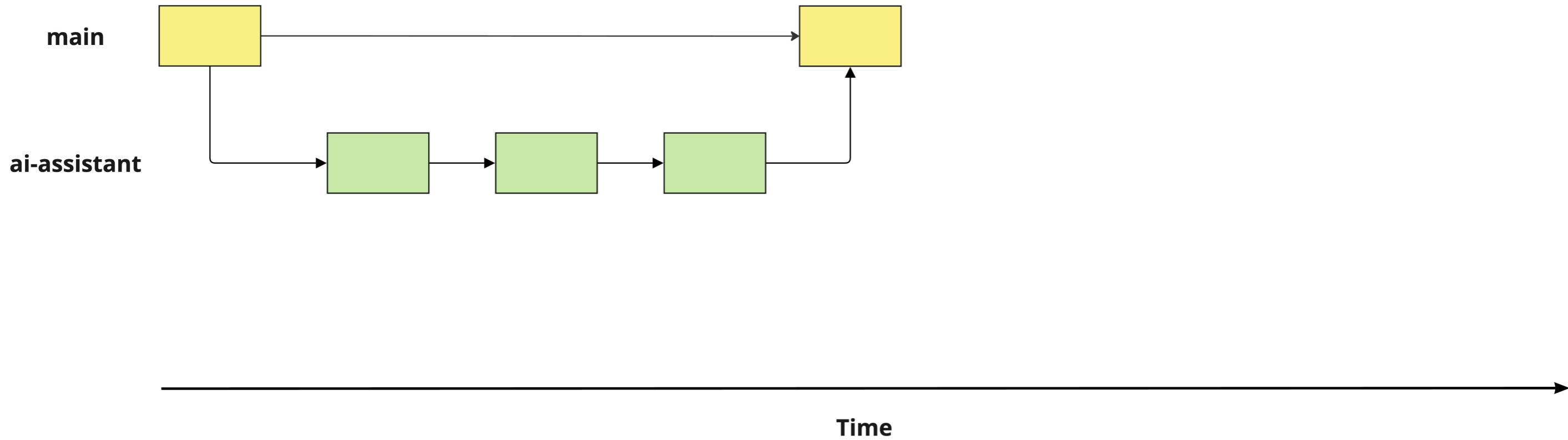
main



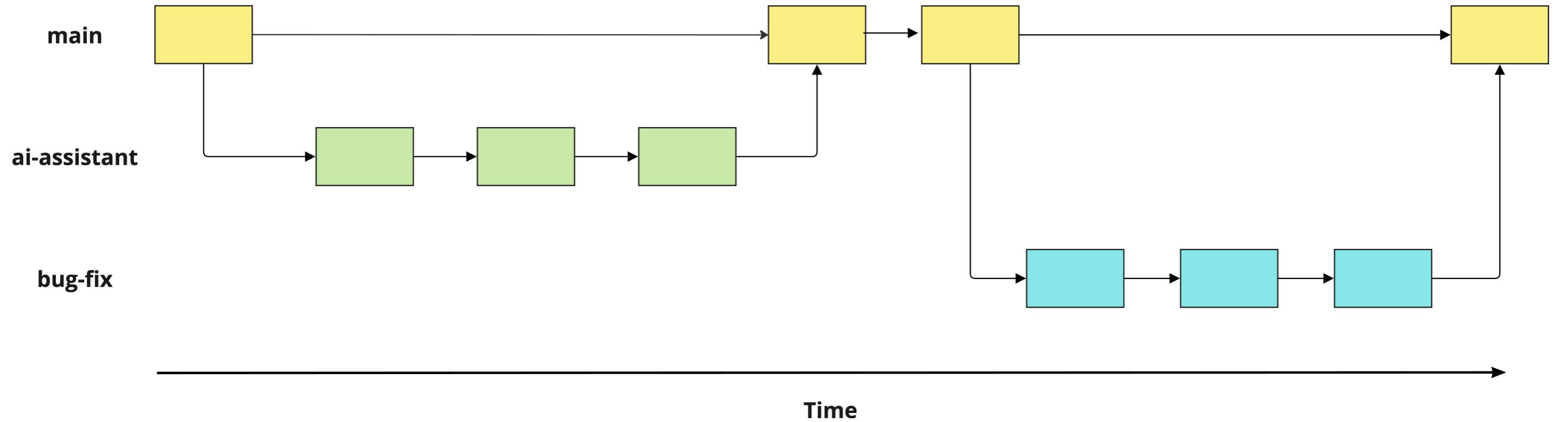
Branching off



Merging back into main



Fixing a bug



Identifying branches

- Listing all branches

```
git branch
```

```
main  
* ai-assistant
```

- * = current branch

Switching between branches

```
git switch main
```

```
Switched to branch 'main'
```

Creating a new branch

- Create a new branch called speed-test

```
git branch speed-test
```

- Move to the speed-test branch

```
git switch speed-test
```

```
Switched to branch 'speed-test'
```

- Create a new branch called speed-test and switch to it

```
git switch -c speed-test
```

```
Switched to a new branch 'speed-test'
```

Terminology

- Creating a new branch = "branching off"
- Creating `speed-test` from `main` = "branching off `main`"

Let's practice!

INTERMEDIATE GIT

Modifying and comparing branches

INTERMEDIATE GIT



George Boorman

Curriculum Manager, DataCamp

Diff recap

Command	Function
git diff	Show changes between all unstaged files and the latest commit
git diff report.md	Show changes between an unstaged file and the latest commit
git diff --staged	Show changes between all staged files and the latest commit
git diff --staged report.md	Show changes between a staged file and the latest commit
git diff 35f4b4d 186398f	Show changes between two commits using hashes
git diff HEAD~1 HEAD~2	Show changes between two commits using <code>HEAD</code> instead of commit hashes

Comparing branches

```
git diff main summary-statistics
```

git diff output

```
diff --git a/bin/summary b/bin/summary
new file mode 100755
index 000000..9d6e2fa
--- /dev/null
+++ b/bin/summary
@@ -0,0 +1,44 @@
+Summary statistics
+
+Age:
+Yes: 25
+No: 24
+
+treatment:
+Yes: 31
+No: 18
+
+work_interfere:
+Sometimes: 17
```

git diff output

```
+  
+benefits:  
+Don't know: 17  
+Yes: 17  
+No: 15  
+  
+mental_health_interview:  
+No: 41  
+Maybe: 7  
+No: 1  
+  
+mental_vs_physical:  
+Don't know: 24  
+Yes: 15  
+No: 10  
diff --git a/results/summary.txt b/results/summary.txt  
new file mode 100644  
index 0000000..e69de29
```

Navigating large git outputs

- Can produce large outputs!
- Press space to progress through and q to exit

Modifying branches

```
git branch
```

```
main  
* feature_dev
```

- **feature_dev**
- Need another branch for a second new feature being developed
- Solution - rename **feature_dev**
- Renaming a branch

```
git branch -m
```

Renaming a branch

```
git branch
```

```
main  
* feature_dev
```

- **feature_dev**
- Need another branch for a second new feature being developed
- Solution - rename **feature_dev**
- Renaming a branch

```
git branch -m feature_dev chatbot
```

Checking our branches

```
git branch
```

```
main  
* chatbot
```

Deleting a branch

- Large projects can have many branches
- Delete branches once we are finished with them
- Delete the `chatbot` branch with `-d` flag

```
git branch -d chatbot
```

```
Deleted branch chatbot (was 3edb989).
```

Deleting a branch that hasn't been merged

- If `chatbot` hasn't been merged to `main`, `git branch -d chatbot` will produce an error

```
error: The branch 'chatbot' is not fully merged.
```

```
If you are sure you want to delete it, run 'git branch -D chatbot'.
```

- Delete with `-D` flag

```
git branch -D chatbot
```

```
Deleted branch chatbot (was 3edb989).
```

- Difficult, but not impossible, to recover deleted branches
- Be sure we don't need the branch any more before deleting!

Summary

Command	Function
<code>git diff main chatbot</code>	Compare the state of the <code>main</code> and <code>chatbot</code> branches
<code>git branch</code>	List all branches
<code>git branch -m old_name new_name</code>	Rename branch called <code>old_name</code> to <code>new_name</code>
<code>git branch -d chatbot</code>	Delete <code>chatbot</code> branch, which has been merged
<code>git branch -D chatbot</code>	Delete <code>chatbot</code> branch, which has not been merged

Let's practice!

INTERMEDIATE GIT

Merging branches

INTERMEDIATE GIT



George Boorman

Curriculum Manager, DataCamp

The purpose of branches

- Each branch should have a particular purpose
 - Developing a new feature
 - Debugging an error
- Once the task is complete, we incorporate the changes into production
 - Typically the `main` branch - "ground truth"

Source and destination

- When merging two branches:
 - the last commits from each branch are called **parent commits**
 - **source** —the branch we want to merge from
 - **destination** —the branch we want to merge into
- When merging **ai-assistant** into **main** :
 - **ai-assistant** = **source**
 - **main** = **destination**

Merging branches

- Move to the destination branch:

```
git switch main
```

- `git merge source`
- From `main`, to merge `ai-assistant` into `main`:

```
git merge ai-assistant
```

- From another branch: `git merge source destination`

```
git merge ai-assistant main
```

Git merge output

```
Updating 7964fe1..d7b2310
Fast-forward
 source/main.py | 11 ++++++++
 1 file changed, 11 insertions(+)
 create mode 100644 source/main.py
```

Git merge output

Commit hashes



```
Updating 7964fe1..d7b2310
Fast-forward
 source/main.py | 11 ++++++++
 1 file changed, 11 insertions(+)
 create mode 100644 source/main.py
```

- Parent commits

Git merge output

Type of merge

```
Updating 7964fe1..d7b2310
Fast-forward
  source/main.py | 11 ++++++++
  1 file changed, 11 insertions(+)
  create mode 100644 source/main.py
```

- Linear commit history: branched off `main` to create `ai-assistant`
- Fast-forward: point `main` to the last commit in the `ai-assistant` branch

Git merge output

Updating 7964fe1..d7b2310
Fast-forward
 source/main.py | 11 +++++++
 1 file changed, 11 insertions(+)
create mode 100644 source/main.py

Number of lines changed



Git merge output

Files modified

```
Updating 7964fe1..d7b2310
Fast-forward
  source/main.py | 11 ++++++++
  1 file changed, 11 insertions(+)
create mode 100644 source/main.py
```

Let's practice!

INTERMEDIATE GIT