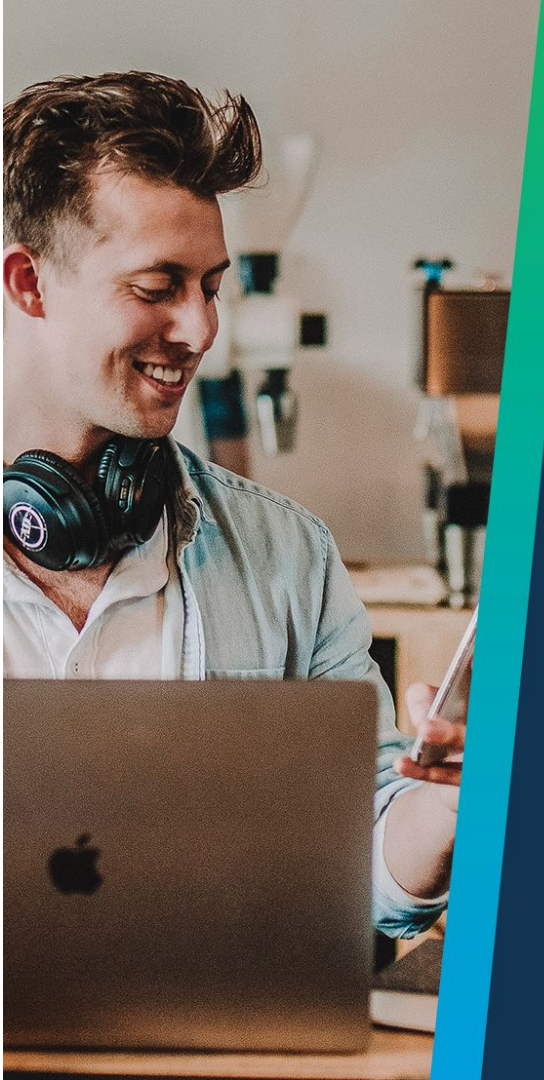


Software Engineering Bootcamp

Hyperiondev

Introduction to Git and Version Control

Lecture – Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. (FBV)
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- We would love your feedback on lectures:(FBV) https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Github Repository - Lecture Examples/Slides

https://github.com/HyperionDevBootcamps/C4_SE_lecture_examples

Github Download/Cheat sheet

https://git-scm.com/downloads

https://github.com/git-guides/install-git

https://education.github.com/git-cheat-sheet-education.pdf

Objectives

1. Version control
    a. What is version control?
    b. Why do we use it?
2. Git
    a. What is git?
    b. How do we use it?

What is Version control?

- Also referred to as source control
- It is a system that tracks and manages changes to software code.

Why Version Control?

- **Collaboration**
  - Multiple people working on the same file at the same time.
  - Hard to keep track of what changes happen when.
  - Certain changes can be accidentally overwritten.
- **Storing Versions**
  - Being able to rollback code becomes a great emergency tactic, when bugs become too difficult to handle.
- **Understanding What Happened**
  - Full history of who made what changes.

Some Terminology

- **Version**
  - Code at a particular state.
- **Repository**
  - The collection of all files at all versions.
- **History**
  - The list of all changes made to a set of files.
- **Commit**
  - Stores a set of changes to the repository.
- **Staging Area**
  - A file containing changes to be added to the next commit.

Introducing Git

- Most widely used version control system.
- Free and open-source. Designed to handle a large variety of systems.
- Distributed architecture:
  - When you download a repository, you download the full history of changes to your local computer.
- Everything is run from the command-line using the git application.

Repositories

- Two types: local and remote.
- All changes stored in a hidden file called “.git”.
- Two ways to get a repository:
  - Create a new one using **git init**.
  - Get a remote one using **git clone <repository-url>**.

Committing Code

- First, you need to add your files to the staging area.
  - **git add <file-name>**
- Once you have added all files to the staging area, then you can commit your code.
  - **git commit -m <commit-message>**
  - NB: Each commit has to have a message attached to it.
  - This just explains what what changed.

Viewing the Status of your Commit

- **git status**
- Shows all new files, changed files, and files added to the current commit.
- E.g:

```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   newFile.py
```

Viewing your Version History

- **git log**
- Shows the commit hash (a unique identifier for the commit), Author, Date and the commit message.
- E.g:

```
commit a9ca2c9f4e1e0061075aa47cbb97201a43b0f66f
Author: HyperionDev Student <hyperiondevstudent@gmail.com>
Date: Mon Sep 8 6:49:17 2017 +0200

Initial commit.
```

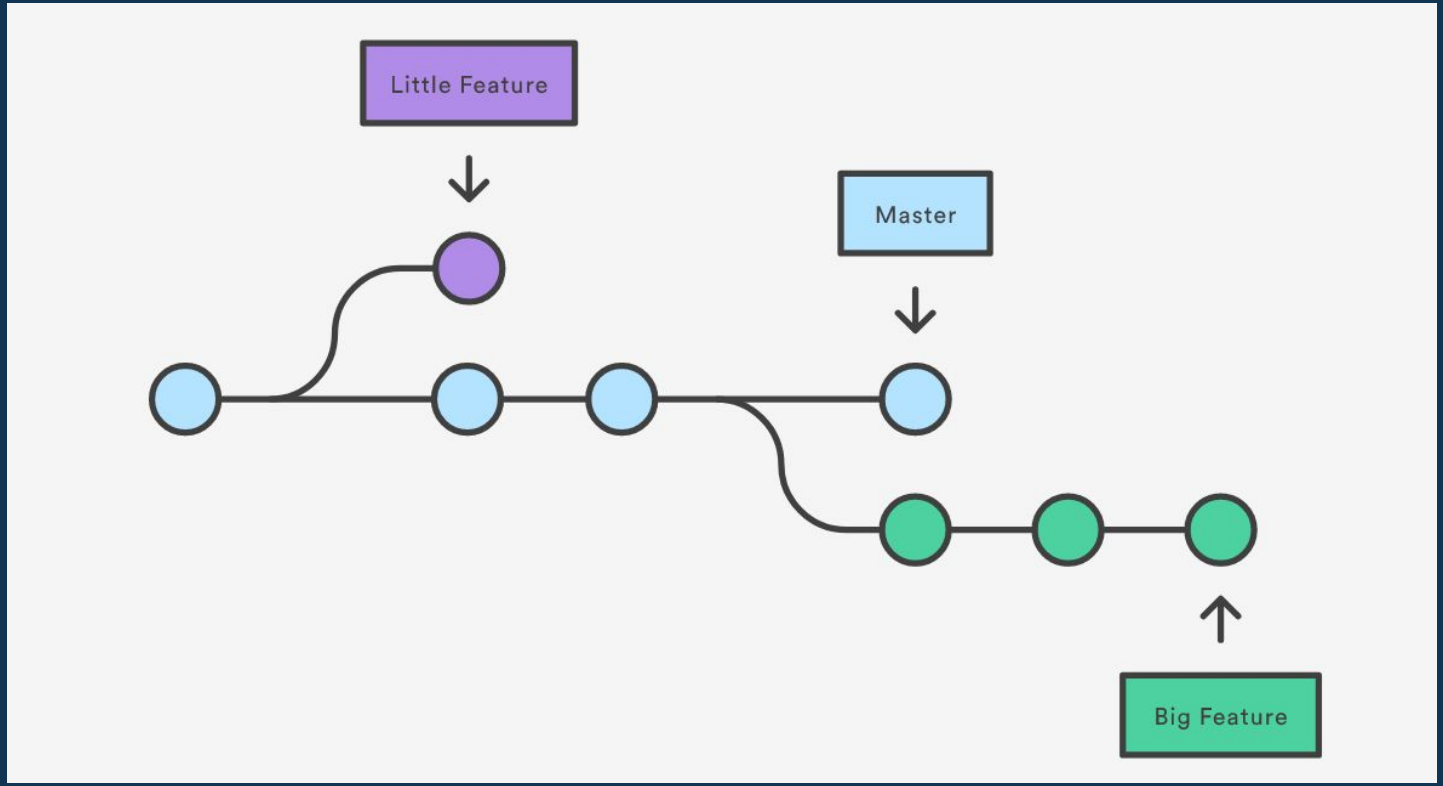
Branching

- Sometimes, a developer needs to work independently on the same code base.
- For example: adding a new feature.
- With other changes constantly being made, this can sometimes be difficult and cause many merge conflicts.
- Solution: branching.

Branching (cont.)

- **git branch <branch-name>**
- To switch branches:
  - **git checkout <branch-name>**
- By default, Git uses **master** as the name of the main branch.

Branching Visualised



Stashing Changes

- When switching branches, Git will throw up a fuss if you have uncommitted changes.
- However, sometimes your changes are not yet ready for a commit.
- You can use **git stash** to temporarily save your changes to a clipboard without committing.
- To get your changes back, **git stash pop** will get the latest stash on the clipboard.

Merging

- There is no use in branching code to make a new feature without being able to make it a part of the main branch.
- Merging allows you to take the changes that you have made in your branch and apply them to the main branch (or another branch of your choice).
- To merge bug-fix branch into master branch:
  - **git checkout master**
  - **git merge bug-fix**

Hyperiondev

Q & A Section

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

Hyperiondev

Thank you for joining us

Take regular breaks.
Stay hydrated.
Avoid prolonged screen time.
Remember to have fun :)