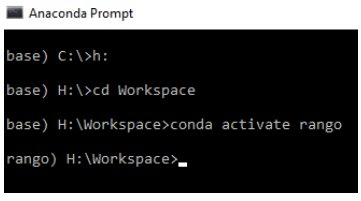
University of Glasgow

Development Environment

Web Application Development 2

Command-Line Interfaces

- Control software or an Operating System by issuing text-based commands
- Alternative to Graphical User Interface (GUI)
  - An OS might have both, and you can choose which to use
- Can be used to perform many common tasks
  - Navigate around directory (folder) structure
  - Create/edit files
  - Run applications
  - Lots more!



Command-Line Interfaces

- We'll be using Command Line a lot in labs
  - Anaconda Prompt
- Common commands:
  - **dir** list files in current directory (**ls** on UNIX-based OS)
  - **mkdir <name>** create new directory called 'name'
  - **cd <name>** change directory/navigate to named destination
  - **cd ..** move 'up' one level to current directory's parent
- Many Django/Anaconda/Git-specific commands
  - see later

Setting Up

- It is good Software Engineering practice to:
  - Use a Version Control System
    - e.g. Git
  - Use a Package Manager
    - e.g. pip
  - Use a Virtual Environment
    - e.g. Anaconda
  - Use an Integrated Development Environment
    - e.g. PyCharm, IDLE

Version Control

- There have been many VC systems, i.e.
  - CVS, Subversion (SVN), Mercurial, Git, etc
- Maintain a history of a software project
  - Often remotely-stored
  - Multiple users can contribute to code
  - It is common practice in industry and open source projects to use version control
  - Essential for teams but also useful for individual projects

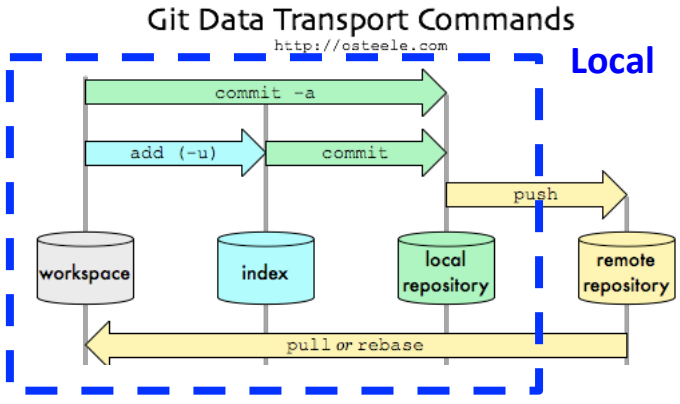
Why use Version Control?

- Access to older (working) versions of your code
- Keeps track of different versions and releases
- Greatly simplifies concurrent work
- Compare/understand changes made by others
- Enables changes to be merged (easily)
- Safeguards your code against disaster
  - (especially if the repo is in the cloud)
- Enables exploratory work (branching)

Git

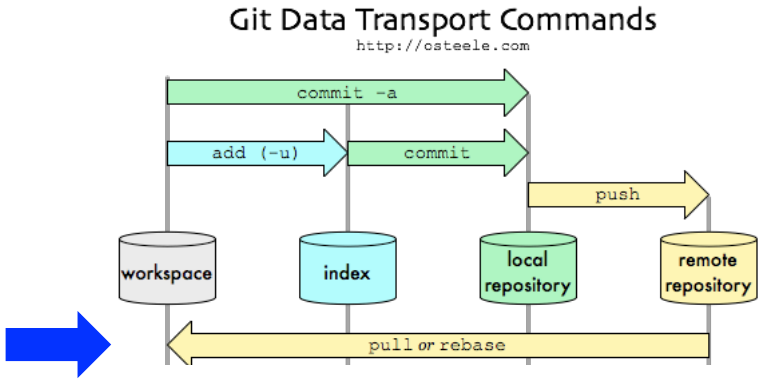
- Originally developed by Linus Torvalds in 2005
- Git is one of the newer VC systems which has several benefits over older ones (CVS/SVN)
  - Efficient, flexible controls
  - 'Extra step' of local repo – easier branching, encourages more commits
- Download Git from:
  - http://git-scm.com/downloads
- How to install:
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Git – How it Works



- For WAD2, your remote repository will be on GitHub: https://github.com/username/repo_name

Git – How it Works



- (**clone** for initially creating a local copy of a project from remote link; **pull** for subsequent updates)

First Steps with Git

- **Either**:
  - Clone a repo that already exists on a remote host
    - e.g. a project on GitHub that you want to work on
    - git clone https://github.com/wad2gla/demo.git
  - Start using git with your existing code project
    - In main project directory:
    - git init
    - git add *
    - git commit −m “first commit”
    - git remote add origin <url you’ve setup on e.g. GitHub>
    - git push −u origin main

Common Git Commands

- git clone <remote_repository>
  - Make a copy of the repository (done once)
- git pull
  - Get the latest remote changes into your local repo
  - Merge with your code files
- git status
  - Find out state of index, changes in workspace, etc
- git add <filename>
  - Add the files you want to commit to the index
- git commit –m “what bugs you fixed”
  - Add the changes to the local repo
- git push
  - Uploads your changes to the remote repository

Git Tips

- Always Pull to make sure you are working on the latest version
- Commit early, Commit often, and then Push your changes frequently − required in Rango assessment!
- The biggest hassle is dealing with merge conflicts
  - If the remote repo has changed, it is your responsibility to merge the versions
  - So communicate with your team
- You will be submitting your Rango app and project code via GitHub!
- Recommend you read the Appendix chapter ‘A Git Crash Course’ in *Tango With Django* course text

Package Managers

- PMs are software tools that automate the process of installing, upgrading and configuring software libraries.
- It tracks the packages installed and their dependencies
  - If pre-requisite packages are not installed it will install them too
- They help to overcome the nightmare of managing libraries, setting up software, replicating an environment
  - Defined: the list of packages is defined
  - Repeatable: easy to install the same set of libraries and versions
  - Managed: stored in the package manager and exportable

Pip: Python Package Manager

- PyPI: Python Package Index is a repository of software for Python
- Pip is used to install and manage packages from PyPI
  - Pip is a recursive acronym: "*Pip Installs Packages*"
- Using pip reduces development set up hassles
  - No need to mess around with path issues
  - No need to worry about what version of the library is used (it is recorded)
  - Easy to export and share the "requirements" i.e. the set of libraries used
  - Easy to install the same set of libraries on another machine
  - Works in conjunction with Virtual Environments

Pip commands

- pip install django==2.2.26
- pip list
  - Show all installed packages
- pip freeze > requirements.txt

Virtual Environments

- A virtual environment instance is a local environment that is configured to provide access to libraries, settings, hardware
  - They keep the dependencies required by different projects in separate places
  - They don't interfere with each other, or the system
- Virtual environment software refers to an application that implements, manages and controls multiple virtual environment instances

Anaconda Virtual Environment

- Anaconda is a tool to keep the dependencies required by different projects in separate places
  - It isolates the different environments and lets you switch between them easily
- Main Advantages
  - Separation of package installation – you can use different package sets for each project
  - Separation of Python versions – you can use different Python versions for each project
  - Virtual environments can be created/switched between easily using the Anaconda command prompt

Anaconda Virtual Environment

- Commands
  - conda create −n <ENVNAME> python=3.8.5
    - Create a new environment, named however we want − replace <ENVNAME>. We can specify Python version
  - conda activate <ENVNAME>
    - Enter the environment
    - Name of active env shown before prompt.
      e.g. (rango) H:\Workspace
  - conda deactivate
    - Leave the environment
  - conda env list
    - List all my environments
  - conda env remove -n <ENVNAME>
    - Delete an environment

QUICK INTRO TO HTML

What is HTML?

- HTML stands for **HyperText Markup Language**
- It's the language web browsers use to interpret what gets displayed when you view a web page
- A mark-up language is a set of tags which describe document content
- Hyperlinks are connections between documents
- HTML documents (web pages) contain HTML mark-up tags and plain text

Basic HTML Example

```
<html>
<head>
<title> The title </title>
</head>
<body>

<h1>WAD2</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Begin HTML now

End HTML now

Tags:

- Keywords (tag names) surrounded by <>
- Normally have opening and closing tags

Plain text:

- Between tags
- The content displayed in the browser

Basic HTML Example

```
<html>
  <head>
    <title> The title</title>
  </head>
  <body>

    <h1>WAD2</h1>

    <p>My first paragraph.</p>

  </body>
</html>
```

HTML Document Structure:

- Nested tags
- Starts with <html> tag
- <head> tag contains information about the document such as title and other things
- <body> tag contains the html to be displayed

Basic HTML Example

```
<html>
    <head>
      <title>The title</title>
    </head>
    <body>

      <h1>Things to do</h1>

      <p>My first paragraph.</p>

    </body>
</html>
```

Elements

- From an opening tag to a closing tag is called an element
- The plain text between opening and closing tags is called the element content

Basic HTML Example

<!DOCTYPE html>

```
<html>
    <head>
      <title>The title</title>
    </head>
    <body>
      <h1>Things to Do</h1>
       Make an HTML page<br>
       Add a paragraph
      <p>My first paragraph.</p>
    </body>
</html>
```

Empty Elements

- There are some tags which have no content
- They also have no end tag
- E.g.
 which forces a line break

Basic HTML Example

<!DOCTYPE html>

```
<html>
    <head>
      <title> The title</title>
    </head>
    <body>

      <h1>Things to To</h1>

       <p>My first paragraph.</p>

    </body>
</html>
```

Two Basic Tags:

- <h1> - "header 1"
  - Used just once
  - Defines the most important heading
  - Search engines use H1 to determine the content of your web pages
  - There are h1,…,h6 headers. H1 being the most important

Basic HTML Example

```
<html>
    <head>
      <title>The title</title>
    </head>
    <body>

      <h1>WAD2</h1>

      <p>My first paragraph.</p>

    </body>
</html>
```

Two Basic Tags:

- <p> is the paragraph tag
- Browsers add space (margin) before and after each <p> element
- They ignore your own formatting – collapse whitespace

Other useful HTML tags (1)

- Anchor tags – provide HTML hyperlinks
  Syntax:
  link text
  Example:
  Visit the WAD2 Moodle page

- Unordered list / list items

   List item one
   List item two

Visit the WAD2 Moodle page

- List item one
- List item two

Other useful HTML tags (2)

- Div elements let you create sections to divide up the page in different ways when coupled with CSS

- Span elements are used to group inline-elements in a document, again when coupled with CSS
  Example: I have
  blue eyes.

I have blue eyes.