The Development Environment

The Development Environment

- Development environment
- Development tools
- Cloud development
- Cloud environment at Griffith
- Unix commands

The Development Environments

- A key term for developers is productivity!
- We want to be as productive as possible
- Having the right tools and workflow can help us be more productive
- When developing a website we generally have two staging environments:
    - The **development** environment
    - The **production** environment
- We build the software and test it in the development environment before releasing it into the production environment.

Production Environment

- The production environment is where the software is 'live'
- This is where the end-users of the website will be using it
- If we were to make a modification of a webpage directly in the production environment, end users would see the changes immediately!
- Rarely is our development work ready for production.
- Some further attributes that the production environment may have:
    - Built for speed: debugging disabled, caching enabled, load balancing, etc.
    - High level of security enabled
    - May be closer to the end user

Development Environment

- The development environment is where code changes can be tested quickly without impacting the end user.
- Some key attributes of the development environment:
  - Fast turn around: quick to upload code, get results, find bugs
  - Debugging is enabled
  - Extensive logging
  - Convenience is more important than performance
  - Close to the developer
- Once code has been tested and verified in the development environment it can be transferred to the production environment.
- In practice an organisation may have additional environments for example there may be a testing environment which is identical to the production environment but not accessible by end users.
- The web development environment will consist of at least the following two tools:
  - A code editor
  - A web server
- In addition the following may also be used:
  - An integrated development environment (IDE)
  - A debugger
  - A file transfer tool to transfer files to the server
  - An application server, if the application doesn't run directly as part of the web server
  - Version control to keep track of different versions of the software

Development Tools

Code Editors

- Code editors are often a personal preference and can depend greatly on the language and frameworks being used.
- Most code is written as text and even a simple text editor like Microsoft NotePad or Apple TextEdit could be used.
- However, code editors provide additional features for writing code including:
  - Syntax highlighting
  - Automatic indentation
  - Code completion
  - And sometimes inline documentation
- Some popular code editors:
  - Sublime (Linux, Mac, Windows)
  - Atom (Linux, Mac, Windows)
  - WebStorm (Linux, Mac, Windows)
  - TextWrangler (Mac)
  - NotePad++ (Windows)
  - Brackets (Linux, Mac, Windows)
  - VS Code (Linux, Mac, Windows)

Web Server

- The development web server may be different to the production server
- Apache is the world's most popular web-facing server hosting over 30.3% of the world's active websites (Netcraft - April 2019).

- Apache is used both for development and production
- However there are faster production servers such as Nginx.
- Additionally some web frameworks come with there own servers such as Ruby on Rails' Webrick
- For this course we will be using Apache as our development server.
- These days it is rare that a company would host their own production server hardware (unless they are Google or Facebook).
- Most companies will use a 'cloud-based' server.
- The advantages of a cloud-based solution are:
  - Cheaper costs because of economies of scale
  - Easier to scale
  - Lower latency as cloud providers often have geographical gateways
  - Potentially better security especially for DDoS attacks.
- As a result many developers are also using cloud solutions for development purposes.

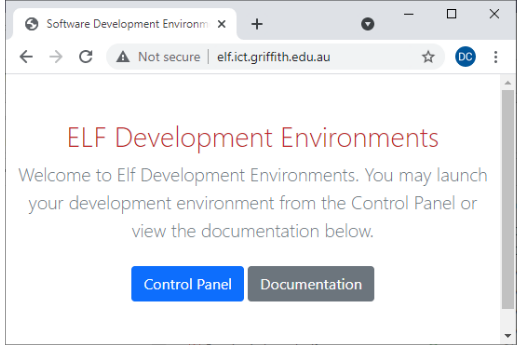
Cloud Development Environment

- A recent trend is cloud development.
- There are two aspects to cloud development:
  - A cloud-based server
  - A browser-based development environment
- Using **cloud-based servers** allow developers to do development using a machine (or virtual machine) with the same setup as the production machine.
- Having a **browser-based development environment** means:
  - No additional software installation is required, and
  - The code can be edited directly on the server without requiring the additional step to upload the software.
- Instead of having the virtual machine on the cloud, one could also host it locally using virtualization technologies such as Virtualbox.
- The biggest advantage of Cloud over local development environment is **ubiquity** – cloud can be access from anywhere and anytime with Internet.
- However, with cloud, one also needs to be more security conscious.

Cloud Development Environment at Griffith

- We (the university) have our own cloud development environment which we'll use for this course.
- This cloud environment, called **Elf** (https://elf.ict.griffith.edu.au/), allow users to create a virtual machines (containers) from images.
- A specific environment (container image on Linux), called **php-apache**, has been setup for this course, which contains:
  - Web server: Apache.
  - Programming language: PHP
  - Tools to run Laravel, e.g. composer.
  - Web-based IDE: Code-server, which provides Visual Studio Code IDE and consoles via a web interface.
- The tools we'll need in order to use Elf are:
  - A web browser
  - (Optional) A tool to download your files from Elf. E.g. WinSCP, or CyberDuck.
- The above tools are all available in the lab computers.
- If you are working **off campus** or **on Griffith Wireless** you need to VPN into Griffith in order to SSH into Elf.
  - VPN software: https://intranet.secure.griffith.edu.au/computing/remote-access/virtual-private-network

Using Elf


Software Development Environm
Not secure | elf.ict.griffith.edu.au
Control Panel
Documentation

ELF Development Environments

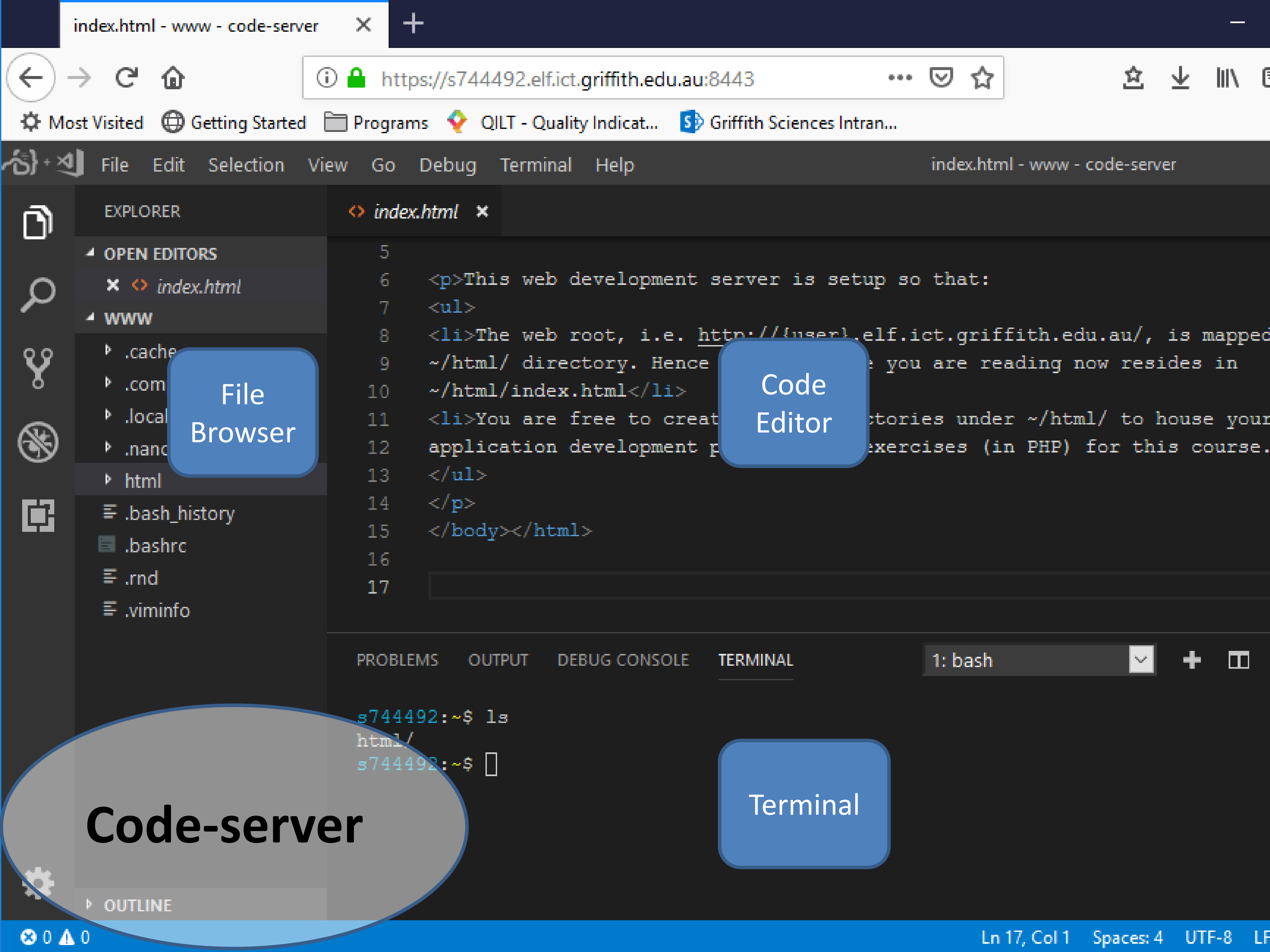
Welcome to Elf Development Environments. You may launch your development environment from the Control Panel or view the documentation below.

To use Elf:

1. Point your browser to: **https://elf.ict.griffith.edu.au/**
2. Click **Control Panel**.
3. Login (using your sNumber and Griffith password).
4. Change environment to **web-dev,** then Start Environment.
5. Copy the **password** (click on Copy)
6. Launch **Code Server**.
7. Paste the password and Submit.
8. Use **code-server** to develop your web application
9. View/run your web application in a browser.

Important:

- If you have previously used a different environment on Elf, it is highly recommended that you remove all your existing files on Elf (see the clean command) before you start the **web-dev** environment.
  - Before you “clean” make sure you back up all your files from Elf.



```
<p>This web development server is setup so that:
<ul>
<li>The web root, i.e. http://{user}.elf.ict.griffith.edu.au/, is mapped
~/html/ directory. Hence                    you are reading now resides in
~/html/index.html</li>
<li>You are free to creat                   ctories under ~/html/ to house your
application development p                   exercises (in PHP) for this course.
</ul>
</p>
</body></html>
```

```
s744492:~$ ls
html/
s744492:~$
```

Code-server

- Code-server is a web-based Integrated Development Environment (IDE) based on the popular **VS Code editor**.
  - https://github.com/cdr/code-server
- Code-server also provides a **terminal** so users can execute commands.
- Code-server is set to automatically run on all Elf VMs on this address: `https://s1234567.elf.ict.griffith.edu.au:8443`
- Note1: replace s1234567 with your s-number.
- Note2: Code-server runs on port 8443.

Files and Zip Backup

- Files in your home directory (i.e. /var/www) are **persistent**. You can start different VMs/environments, and these files would still be there.
- However, it is still your duty to **backup your files**.
- You can use the **zip** and **unzip** command to create a zip archive of your work or to extract a zip archive.
- To zip up a directory, use the command:

```
zip –r <zip file> <source directory>
```

- To unzip, simply:

```
unzip <zip file>
```

File download and Backup

Download your work regularly to your own storage to keep a backup copy. There are different ways to download files from Elf:

Through Code-server UI

- In the directory tree of code-server, right click on the file you want to download, then select download.

Use the Web Server

- Since Elf has a web server, we could also download the file via a browser. Simply place the file to download anywhere in the html directory. Then enter the URL to this file in your browser.
- E.g. The file to download is in ~/html/download.zip. Then simply put the following URL in a browser:

  `http://s1234567.elf.ict.griffith.edu.au/download.zip`

Backup with git

- For a more sophisticated backup solution, you can use **git** to backup your work to a Cloud repository such as **GitHub** or **BitBucket**. **git** is built into Linux.
    - To use git you'll need to learn git commands (which is covered in a different course).

Upload to Elf

- To upload files to Elf, you can simply drag and drop your file into code-server's directory tree.

Web Server and Configuration

- The web server (apache) has been preinstalled and is configured to run when you VM starts up.
- The web root of apache is set to your `~/html` directory. So that the URL `https://s1234567.elf.ict.griffith.edu.au` will load the file `~/html/index.html`
- The configuration file for apache is located in `/etc/apache2/sites-available`. However, on Elf, we do not have super user access to modify this file.

- Apache allows the use of **`.htaccess`** files to further configure its behavior.
- `.htaccess` files are placed in the directory where of the web pages are loaded from, hence we are able to create our own `.htaccess` files.
- Some of configuration we can do with `.htaccess` include:
  - Redirection – e.g. when a website has moved
  - Error page
  - Password protection
  - Show directory listing

Password Protection

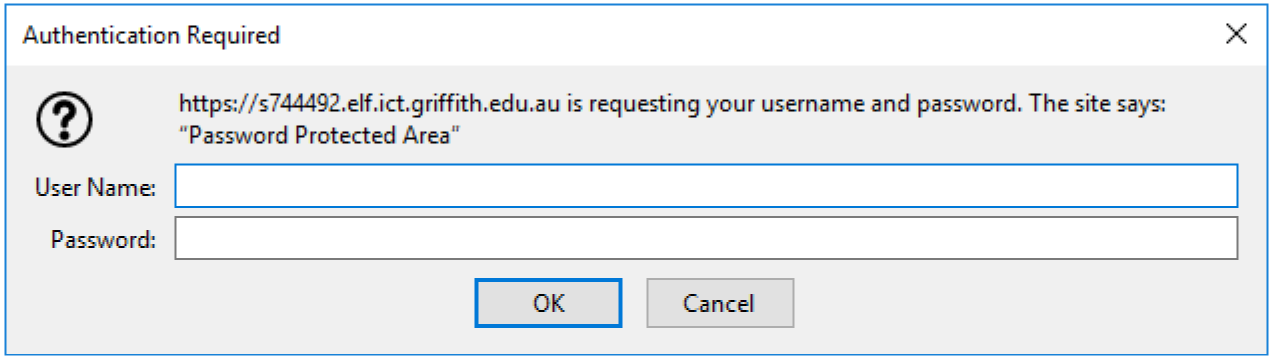
- Web front-end code are exposed to public. It's easy for anyone that have access to your webpage to see your HTML/CSS/Javascript code.
  - You should not embed any sensitive data on your front end code.
- For the purpose of this course, you can setup password protection for your website via *.htaccess* to prevent any unauthorized person from accessing your website.
- We'll setup password protection for the **html** directory, which will also apply to any directory under the html directory.
- **Step 1:** Create a password file with the command:

  ```
  htpasswd -c -b .htpasswd {username} {password}
  ```

  You can enter any username and password in the above command.
- **Step 2:** Update the .htaccess file in html directory by adding the following:

  ```
  AuthName "Password Protected Directory"
  AuthUserFile /path/to/passwd/file/.htpasswd
  AuthType Basic
  require valid-user
  ```
- If the .htpasswd file is also in the html directory, then the value for AuthUserFile would be: `/var/www/html/.htpasswd`

- Now when you try to access the website, you'll be ask to enter the username and password.


Authentication Required

https://s744492.elf.ict.griffith.edu.au is requesting your username and password. The site says: "Password Protected Area"

User Name:

Password:

- If you access your website via **http**, then your password will be sent to the server unencrypted.
- As you should never have password being sent unencrypted, hence, we highly recommend that you use **https** to access your website.

Exercise – Create a HTML file on Elf

- Create a html called **index.html** in the directory *webAppDev/week1/task1* under the html directory.
- This file should contain the following:

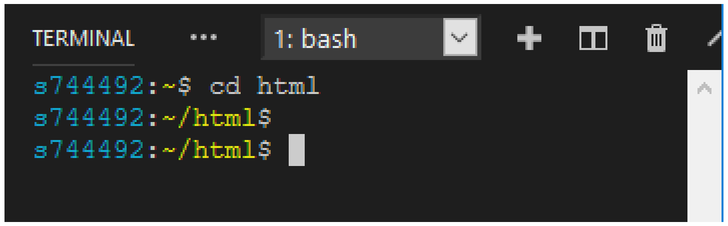
```
<!DOCTYPE html>
<html>
<head>
   <meta charset=utf-8>
   <title>Hello World!</title>
</head>
<body>
   <h1>Hello World!</h1>
</body>
</html>
```

- Test it to make sure you are able to display this page in a web browser.
- Upload an image to task1 directory and display this image in index.html.
- Zip up the *task1* project/directory and download it from Elf.
- Set a password to restrict access to all your websites on Elf via .htaccess file.

Unix Command Line

- Despite major advances in computing over recent decades the command-line is still prominent!
- The unix command-line, invented in the 1970s is still in use today!
- It would be nice to avoid it, however it is still a fundamental skill to have for software developers.
- Some tasks can be performed through a web interface, but inevitably you will need to access the command-line at some point so having some Unix skills is an advantage.

The Linux terminal

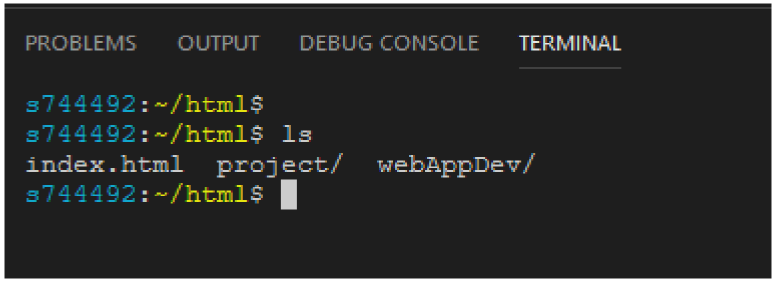


```
s744492:~$ cd html
s744492:~/html$
s744492:~/html$
```

- The Linux terminal may appear slightly differently depending on the interface used, but they all work the same way.
- The text before the white cursor is called the command prompt
- If you press enter, you will see that every line begins with the command prompt.
- It tells you two things:
  - Your username: s744492
  - Your current directory: ~/html
- Note that in Unix '~' indicates your home directory.

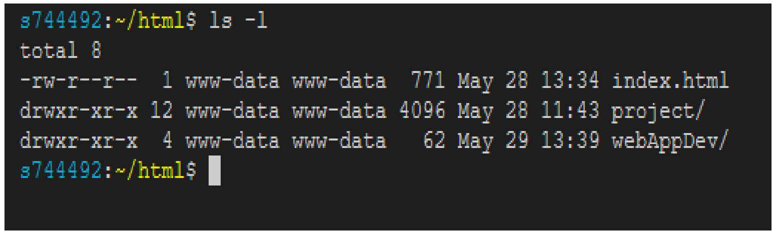
Some Unix Commands

- See what is in your current directory:



```
s744492:~/html$
s744492:~/html$ ls
index.html  project/  webAppDev/
s744492:~/html$
```

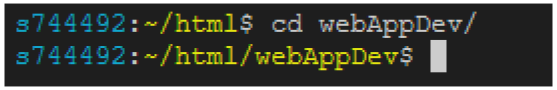
- The **ls** command provides a list of files and directories in the current directory.
- Adding the -l option provides a more detailed/long listing:



```
s744492:~/html$ ls -l
total 8
-rw-r--r--  1 www-data www-data  771 May 28 13:34 index.html
drwxr-xr-x 12 www-data www-data 4096 May 28 11:43 project/
drwxr-xr-x  4 www-data www-data   62 May 29 13:39 webAppDev/
s744492:~/html$
```

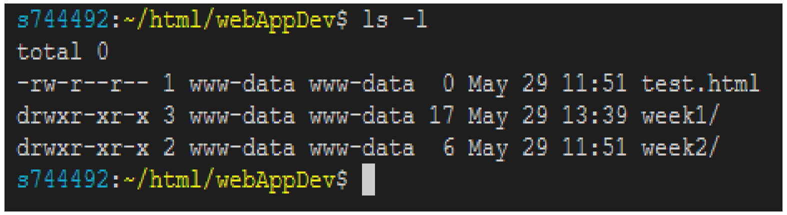
- We can see the last modified date/time as well as permission and the owners of the file.

- We can change into the *webAppDev* directory with the **cd** command (change directory):



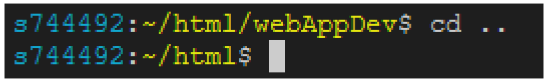
```
s744492:~/html$ cd webAppDev/
s744492:~/html/webAppDev$
```

- Notice how the prompt changes to reflect the current directory.
- Perform another ls -l to see what is in the *webAppDev* directory:



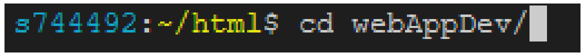
```
s744492:~/html/webAppDev$ ls -l
total 0
-rw-r--r-- 1 www-data www-data  0 May 29 11:51 test.html
drwxr-xr-x 3 www-data www-data 17 May 29 13:39 week1/
drwxr-xr-x 2 www-data www-data  6 May 29 11:51 week2/
s744492:~/html/webAppDev$
```

- **cd ..** Will take us back to the parent directory.



```
s744492:~/html/webAppDev$ cd ..
s744492:~/html$
```

- The **Tab key** will perform autocomplete of file/directory name in the current directory.
- So if you are sick of typing *webAppDev*, simply type **w<Tab>**:



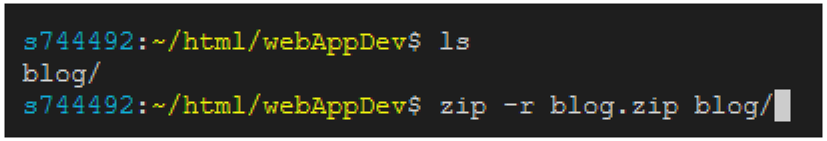
```
s744492:~/html$ cd webAppDev/
```

- Unix commands are case sensitive, hence WebAppDev ≠ webAppDev.

Zip and unzip

- To zip up a directory, use the command:

  `zip -r <zip file> <source directory>`



```
s744492:~/html/webAppDev$ ls
blog/
s744492:~/html/webAppDev$ zip -r blog.zip blog/
```

- To unzip, simply:

  `unzip <zip file>`

Other commands in brief

- **touch <filename> -** creates an empty file or sets the modified date/time for a file to the current time.
- **mkdir <directory name>** - creates a new directory.
- **rm <file or directory name>** - removes that file or directory.
- **cp <source> <destination> -** copies file or directory.
- **mv <source> <destination> -** moves file or directory.
- **sudo <command>** - performs the specified command as super user (not available in Elf).
- **man <command>** - displays the manual for that command.