PG6300
On Bash, Docker, Databases and Cloud Deployment

Dr. Andrea Arcuri

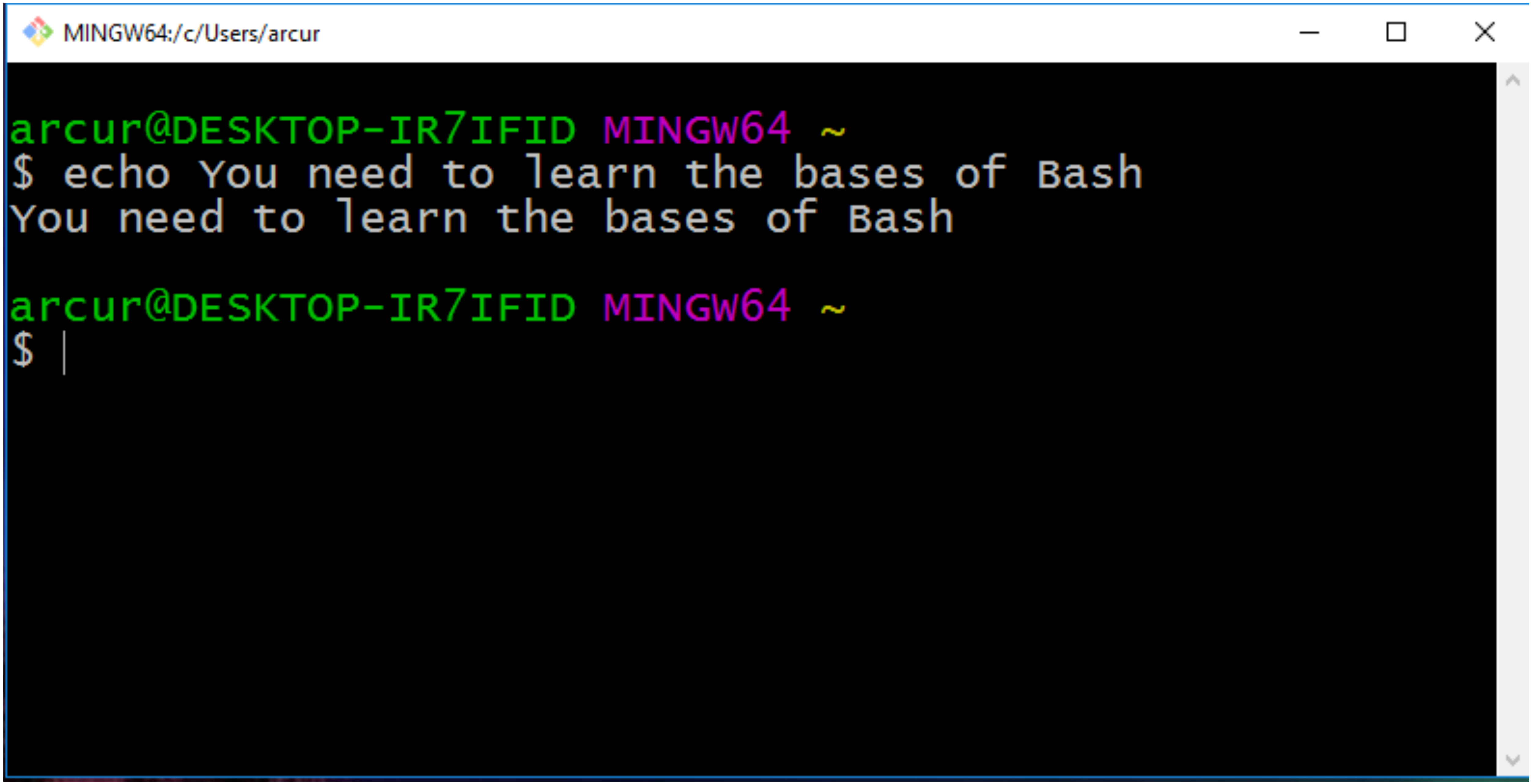
Goal

- Discuss some important related topics, although **NOT** strictly necessary for the exam
- Depending on your previous courses, could be totally new or just repetition

Bash

Bash

- Bash is a Linux/Mac/Unix shell and command language
- There are also other kinds of shells
  - eg, PowerShell in Windows
- A *shell* is also called: *terminal*, *console*, *command-line*, etc.
- Enable to *type* commands (eg programs), and execute them



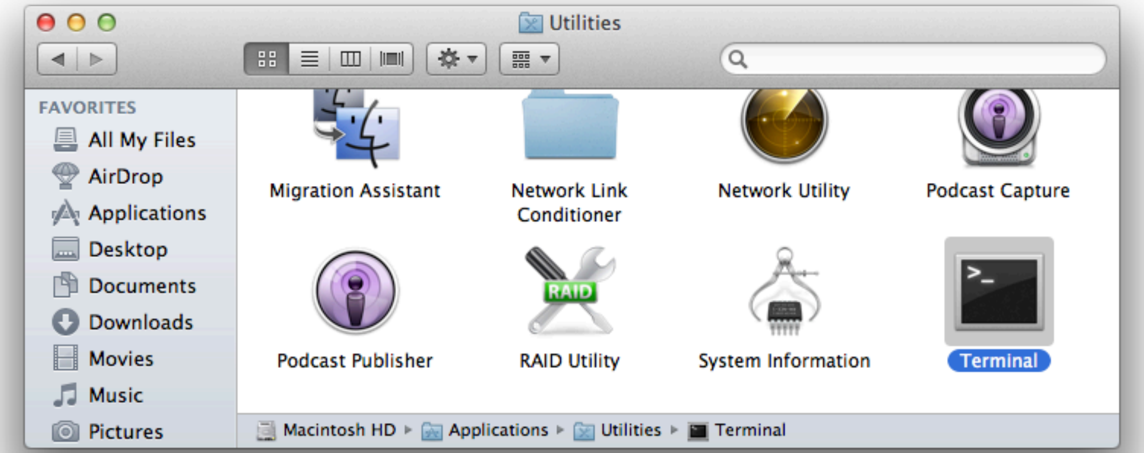
```
arcur@DESKTOP-IR7IFID MINGW64 ~
$ echo You need to learn the bases of Bash
You need to learn the bases of Bash

arcur@DESKTOP-IR7IFID MINGW64 ~
$ 
```

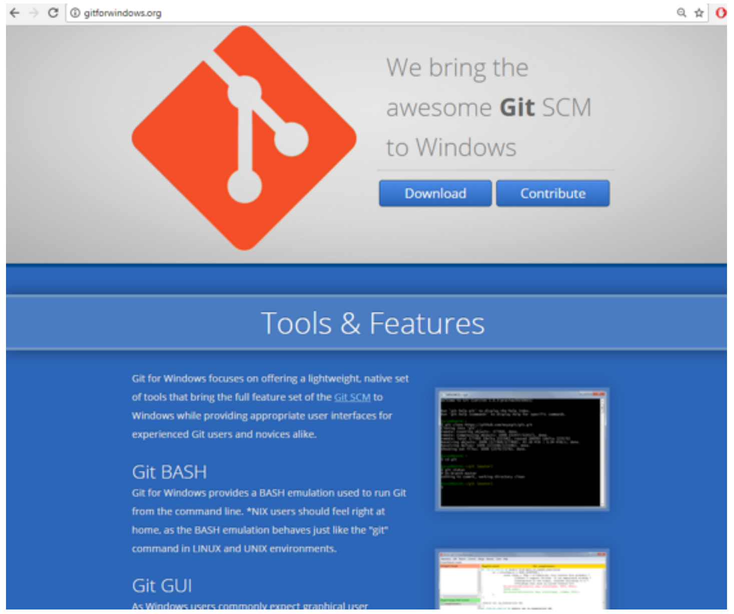
Why?

- Critical skill when you are a programmer
- Help automating several tasks
- When dealing with web/enterprise systems, many servers will NOT have a GUI…
  - … you will access them remotely via SSH using a terminal
  - … this also applies for embedded and IoT devices
- Helpful when commands with specific parameters (eg Git)
- You need to be able to do basic commands
- We will need *Bash* commands in *Docker*

Installing Bash

- If you are using Linux/Mac, it is already installed
    - Mac: Utilities -> Terminal
- If using Windows, strongly recommended to install GitBash
    - which is part of “Git for Windows” at http://gitforwindows.org/





We bring the awesome **Git** SCM to Windows

Tools & Features

Git for Windows focuses on offering a lightweight, native set of tools that bring the full feature set of the Git SCM to Windows while providing appropriate user interfaces for experienced Git users and novices alike.

Git BASH

Git for Windows provides a BASH emulation used to run Git from the command line. *NIX users should feel right at home, as the BASH emulation behaves just like the "git" command in LINUX and UNIX environments.

Git GUI

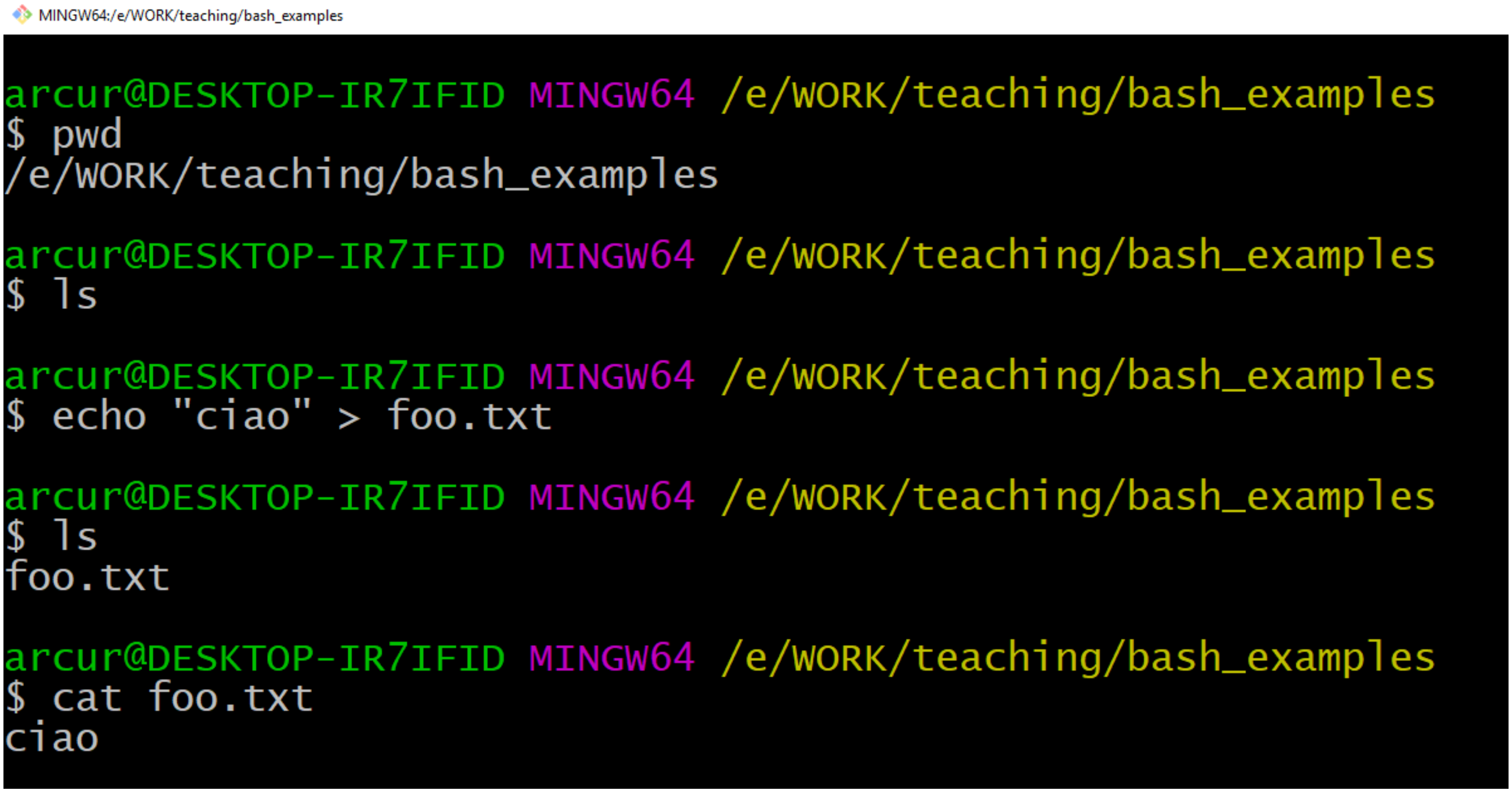
As Windows users commonly expect graphical user

Basic Commands

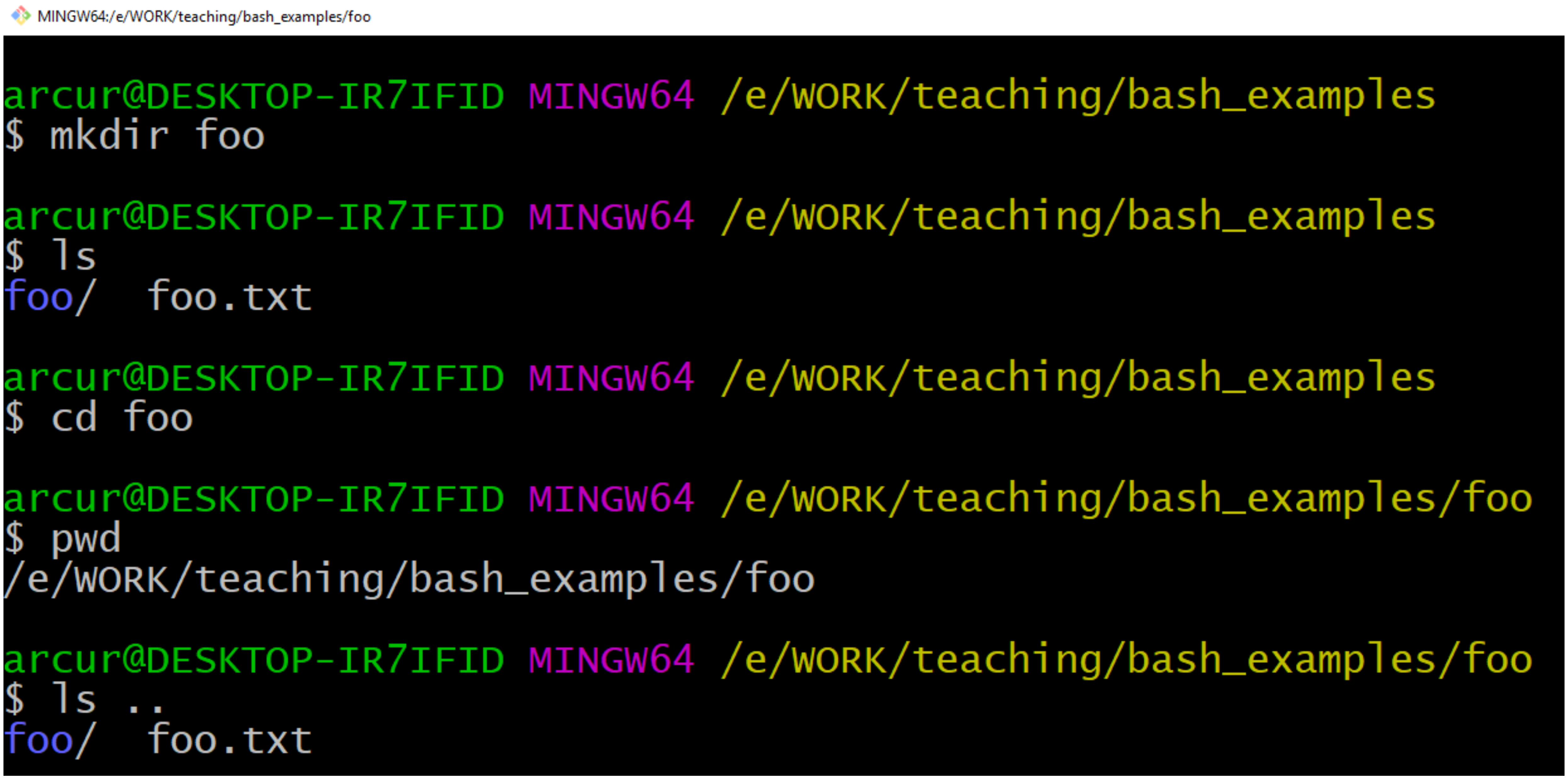
- "." the current directory
- ".." the parent directory
- "~" home directory
- "pwd" print working directory
- "cd" change directory
- "mkdir" make directory
- "ls" list directory content
- "cp" copy file
- "mv" move file
- "rm" remove ("-r" for recursive on directories)
- "man" manual for a specific command

Cont.

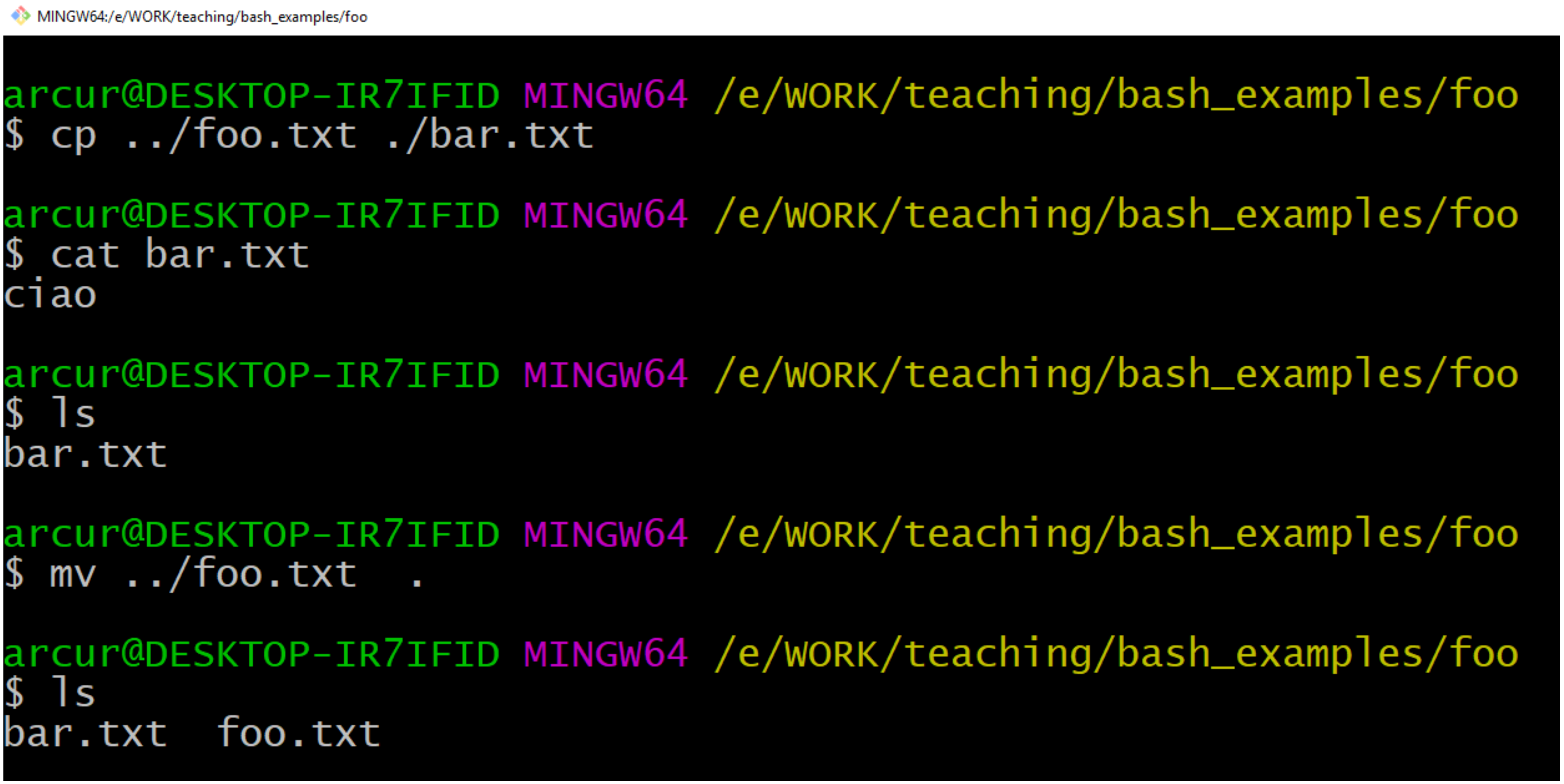
- “echo” print input text
- “cat” print content of file
- “less” scrollable print of file
- “>” redirect to
- “>>” append to
- “|” pipe commands
- “which” location of program
- “$” resolve variables
- “wc” word count
- “find” files
- “grep” extract based on regular expression
- “touch” modify access time of file, and create it if non-existent


MINGW64:/e/WORK/teaching/bash_examples

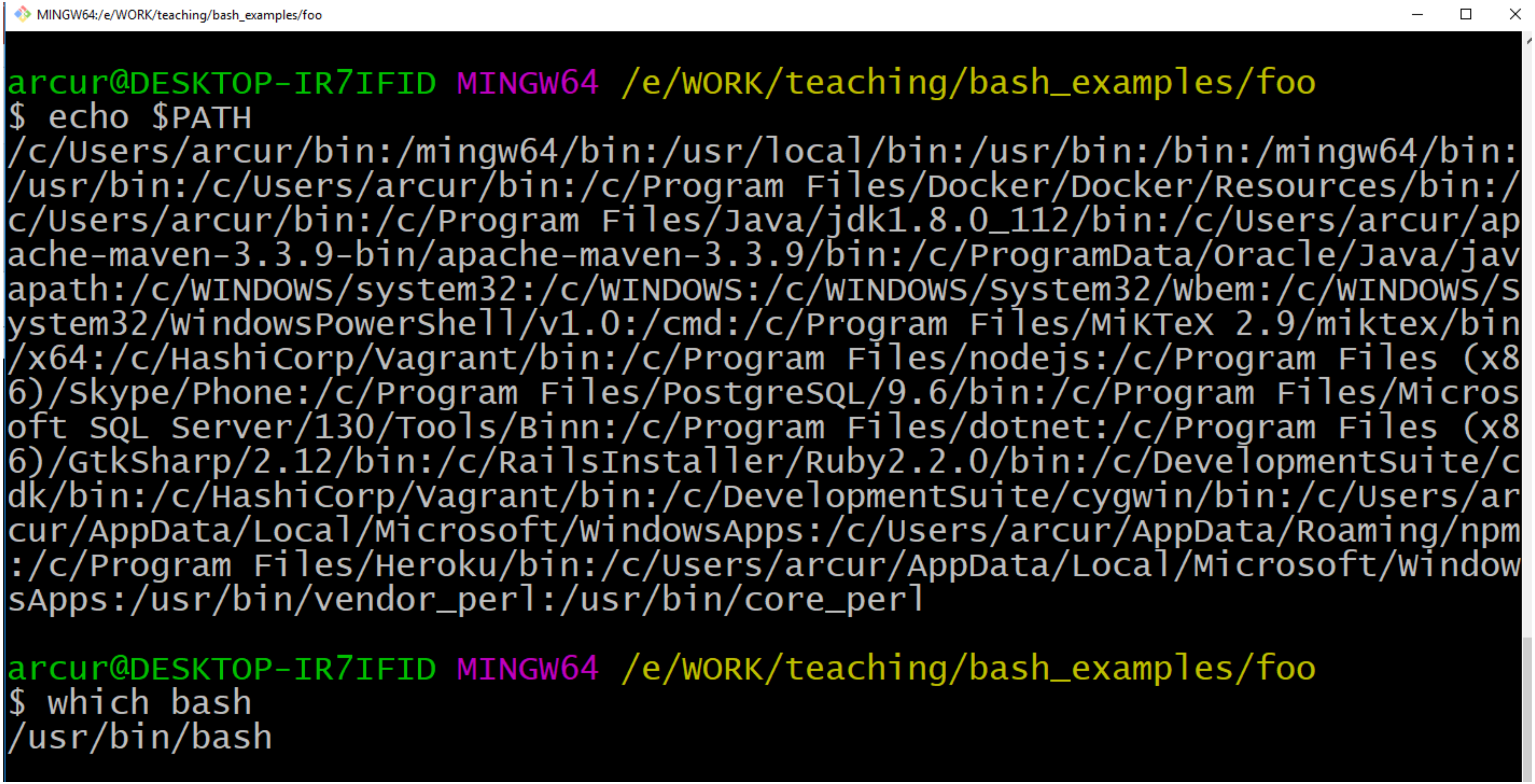
```
arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ pwd
/e/WORK/teaching/bash_examples

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ ls

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ echo "ciao" > foo.txt

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ ls
foo.txt

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ cat foo.txt
ciao
```


MINGW64:/e/WORK/teaching/bash_examples/foo

```
arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ mkdir foo

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ ls
foo/  foo.txt

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples
$ cd foo

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ pwd
/e/WORK/teaching/bash_examples/foo

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ ls ..
foo/  foo.txt
```



```
arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ cp ../foo.txt ./bar.txt

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ cat bar.txt
ciao

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ ls
bar.txt

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ mv ../foo.txt  .

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ ls
bar.txt  foo.txt
```

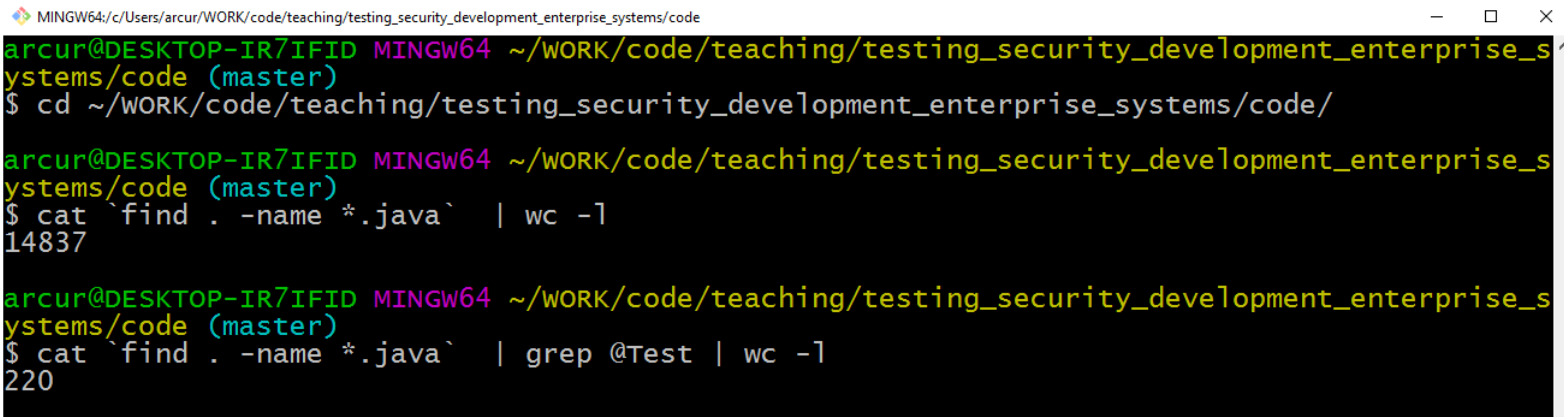


MINGW64:/e/WORK/teaching/bash_examples/foo

```
arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ echo $PATH
/c/Users/arcur/bin:/mingw64/bin:/usr/local/bin:/usr/bin:/bin:/mingw64/bin:
/usr/bin:/c/Users/arcur/bin:/c/Program Files/Docker/Docker/Resources/bin:/
c/Users/arcur/bin:/c/Program Files/Java/jdk1.8.0_112/bin:/c/Users/arcur/ap
ache-maven-3.3.9-bin/apache-maven-3.3.9/bin:/c/ProgramData/Oracle/Java/jav
apath:/c/WINDOWS/system32:/c/WINDOWS:/c/WINDOWS/System32/Wbem:/c/WINDOWS/S
ystem32/WindowsPowerShell/v1.0:/cmd:/c/Program Files/MiKTeX 2.9/miktex/bin
/x64:/c/HashiCorp/Vagrant/bin:/c/Program Files/nodejs:/c/Program Files (x8
6)/Skype/Phone:/c/Program Files/PostgreSQL/9.6/bin:/c/Program Files/Micros
oft SQL Server/130/Tools/Binn:/c/Program Files/dotnet:/c/Program Files (x8
6)/GtkSharp/2.12/bin:/c/RailsInstaller/Ruby2.2.0/bin:/c/DevelopmentSuite/c
dk/bin:/c/HashiCorp/Vagrant/bin:/c/DevelopmentSuite/cygwin/bin:/c/Users/ar
cur/AppData/Local/Microsoft/WindowsApps:/c/Users/arcur/AppData/Roaming/npm
:/c/Program Files/Heroku/bin:/c/Users/arcur/AppData/Local/Microsoft/Window
sApps:/usr/bin/vendor_perl:/usr/bin/core_perl

arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/teaching/bash_examples/foo
$ which bash
/usr/bin/bash
```

- What if you want to count the number of lines of your programs?

- Or the number of lines with a given word?
  - Eg “@Test” to count the number of tests in a program


MINGW64:/c/Users/arcur/WORK/code/teaching/testing_security_development_enterprise_systems/code

```
arcur@DESKTOP-IR7IFID MINGW64 ~/WORK/code/teaching/testing_security_development_enterprise_s
ystems/code (master)
$ cd ~/WORK/code/teaching/testing_security_development_enterprise_systems/code/

arcur@DESKTOP-IR7IFID MINGW64 ~/WORK/code/teaching/testing_security_development_enterprise_s
ystems/code (master)
$ cat `find . -name *.java`  | wc -l
14837

arcur@DESKTOP-IR7IFID MINGW64 ~/WORK/code/teaching/testing_security_development_enterprise_s
ystems/code (master)
$ cat `find . -name *.java`  | grep @Test | wc -l
220
```

- In this example, first recursively *find* in current directory "." all the files that ends with ".java"
- Then, such list of names is replaced inside ``, and so given as input to "cat", which prints those files line by line
- The output of *cat* is pipelined "|", and given as input to *grep*
- *grep* will output only the lines that contains the string "@Test", ie it acts as a filter
- the output of *grep* is then pipelined "|" to "*wc –l*", which counts the number of lines in input
- Therefore, that script checks content of all Java files and counts the number of lines in them having the text "@Test"

Useful Tips

- User arrows (up/down) to go through history of commands
- Use “tab” key to complete words, ie commands / file names
- Bash commands can be put in executable scripts
  - Can use “*.sh” as file extension, eg “foo.sh”
  - First lines needs to be “#!<pathToBash>”, eg “#!/usr/bin/bash”
  - Then it can be executed from terminal like any other program

Virtual Machines

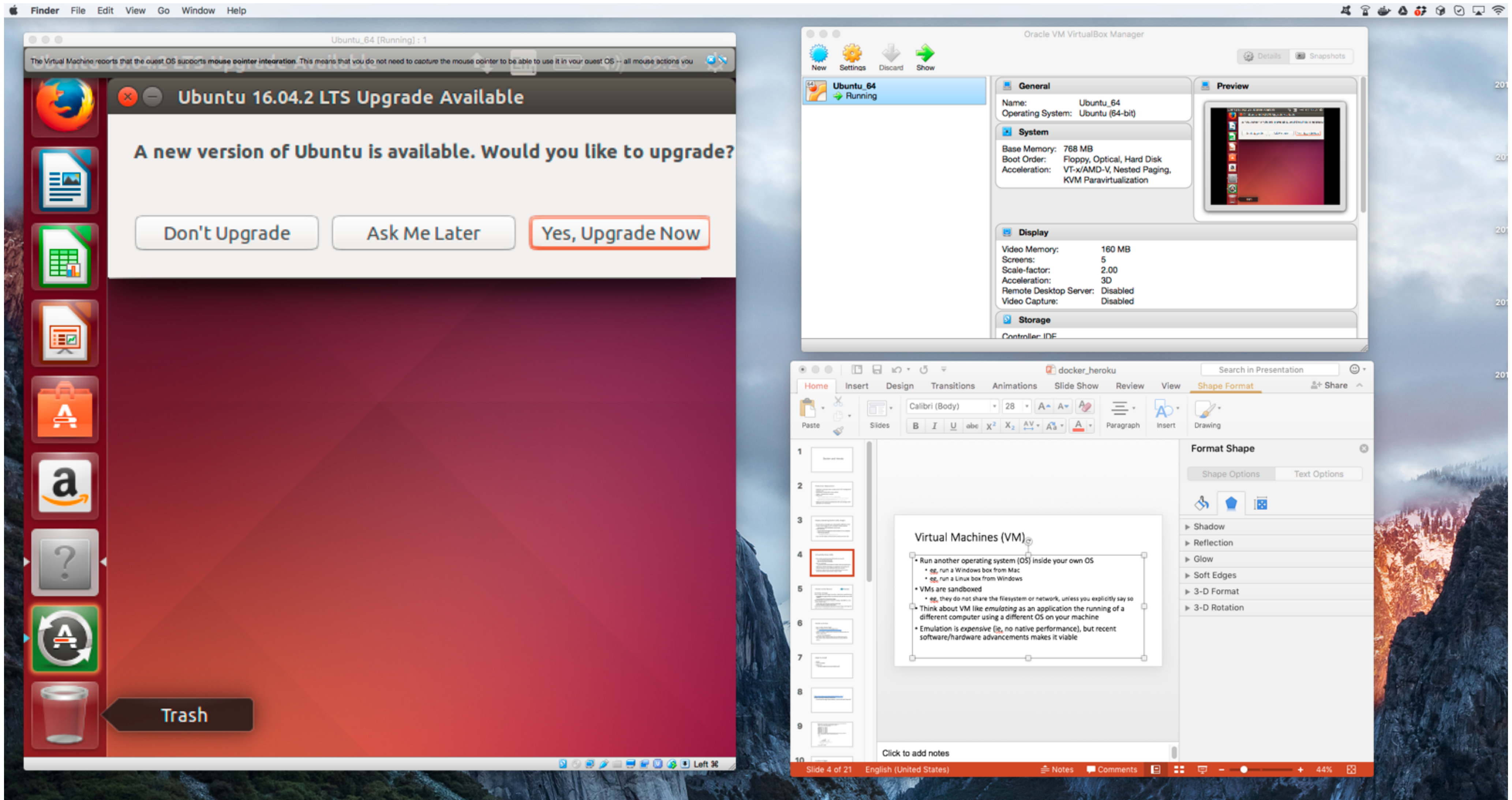
OS-specific Programs

- Ever tried to run a PS4 game on Windows or an XBox?
- Or a Windows application on a Mac?
- Why do they NOT work?
- Compiled code relies on *system calls* to interact with hardware (mouse, screen, etc), and those are OS dependent
  - system calls can be seen as APIs of the OS

Virtual Machines (VM)

- Run a OS (*guest*) inside another one (*host*)
- The *guest* OS runs in an *emulated* environment, controlled by a hypervisor / virtual machine monitor
- The VM with *guest* OS runs likes a program in the *host* OS
- The *host* OS can decide whether and how the *guest* OS interacts with the environment
- The *guest* OS will be less efficient than *host* one, but performance is getting much better nowadays

Linux running in Mac/Windows via VirtualBox



The Virtual Machine reports that the guest OS supports **mouse pointer integration**. This means that you do not need to *capture* the mouse pointer to be able to use it in your guest OS -- all mouse actions you

Ubuntu 16.04.2 LTS Upgrade Available

A new version of Ubuntu is available. Would you like to upgrade?

Don't Upgrade | Ask Me Later | Yes, Upgrade Now

Ubuntu_64
Running

General
Name: Ubuntu_64
Operating System: Ubuntu (64-bit)

System
Base Memory: 768 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Preview

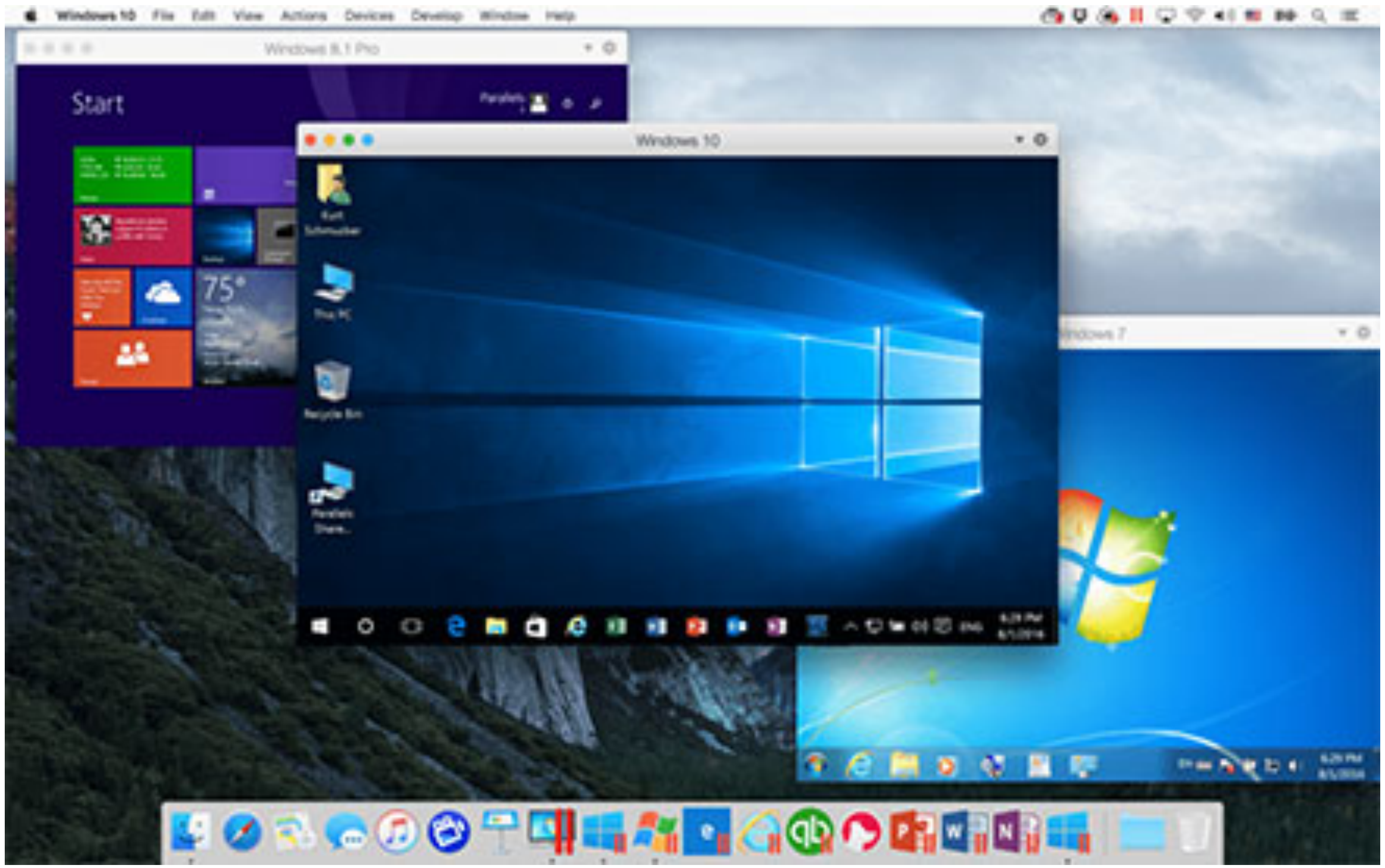
Display
Video Memory: 160 MB
Screens: 5
Scale-factor: 2.00
Acceleration: 3D
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE

Virtual Machines (VM)

- Run another operating system (OS) inside your own OS
  - eg, run a Windows box from Mac
  - eg, run a Linux box from Windows
- VMs are sandboxed
  - eg, they do not share the filesystem or network, unless you explicitly say so
- Think about VM like *emulating* as an application the running of a different computer using a different OS on your machine
- Emulation is *expensive* (ie, no native performance), but recent software/hardware advancements makes it viable

Windows running in Mac via Parallels



Why VMs?

- Can run programs compiled/written for other OSs
- Security: as *host* decides what *guest* can interact with, the guest is effectively run in a *sandbox*
    - Can also easily wipe out and re-run a comprised *guest* OS
- *Portability*: instead of shipping a program which needs to be installed and configured (and that might only work for a specific OS), do ship an entire OS *image* including the program and everything it needs
    - such image will run in the VM

Docker

Deploy OS Images

- When developing applications, not limit to just package your code
  - Java, NodeJS, PHP, etc.
- Create a whole image of an OS, including all needed software
  - Eg the version of JRE/.Net/Ruby/etc. that you need
- Particularly useful when developing web applications to install on a server
- Do not install the OS image on the server, but rather run it in a virtual machine
- Also, instead of installing a database, could just load a OS image with it
- How to automate all this?

Docker to the Rescue

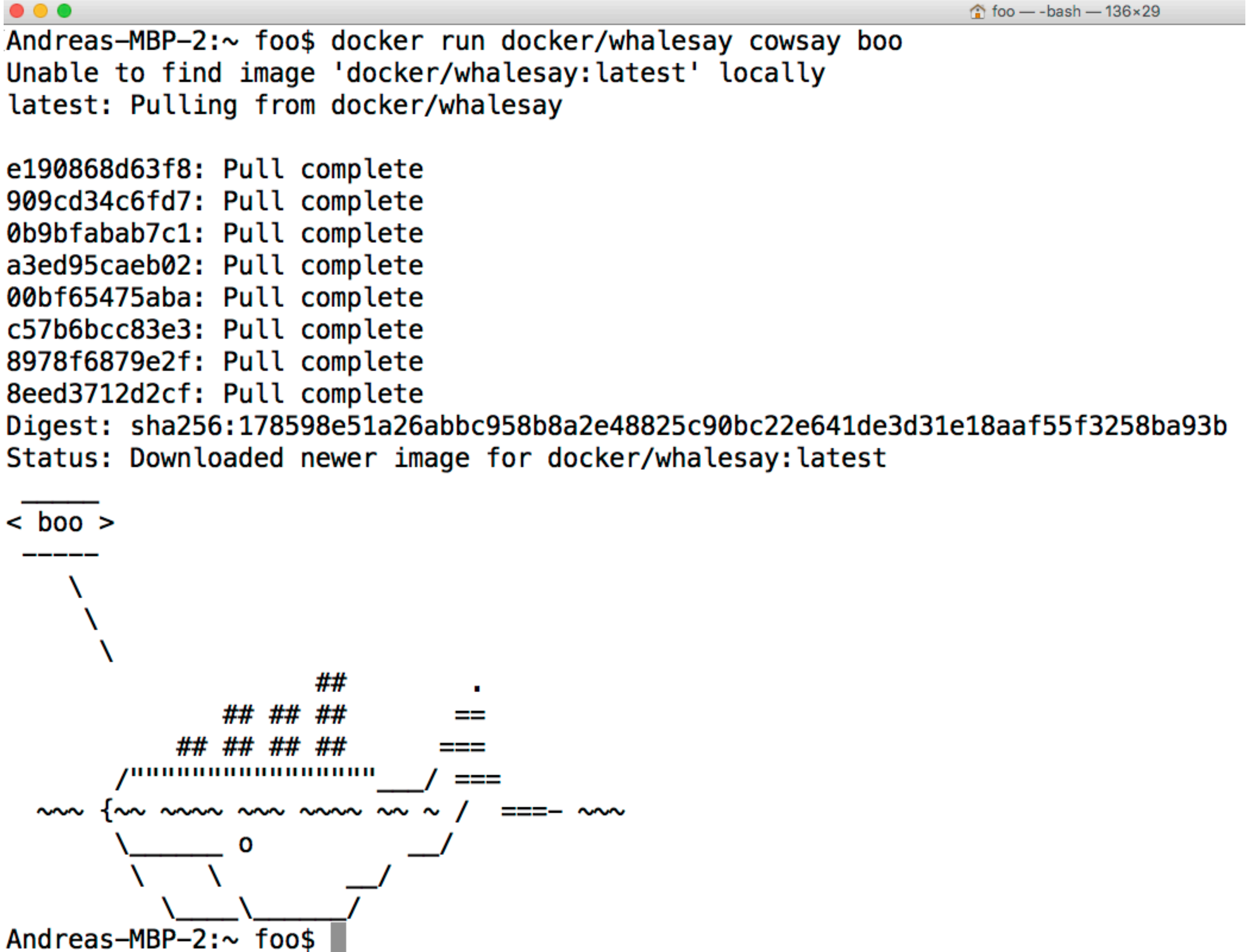
- Automate the deployment of application inside software containers
- Create whole OS images, based on predefined ones
- Eg, a Linux distribution with the latest version of the frameworks you need
  - NodeJS, PHP, JDK, etc.
- Large *online* catalog of existing base images at Docker Hub
- Your application, and any needed third-party library, will be part of the OS image
- Use Docker (and tools built on / using it) to deploy your OS images and start them locally or on remote servers

How to Use Docker?

- First you need to install it
  - https://store.docker.com/
  - Note: if you are using Windows, Home Edition might not be enough. You would need a better version, like the Educational one, which you should be able to freely get from school
- To run existing images, you just need to type commands from a shell terminal (eg, GitBash)
- When you are writing your own projects, you need to create configuration files
  - *Dockerfile*: specify how to build an OS image
  - *docker-compose.yml*: for handling multi-images
- Then, use *docker* and *docker-compose* commands from the command line

Docker Examples

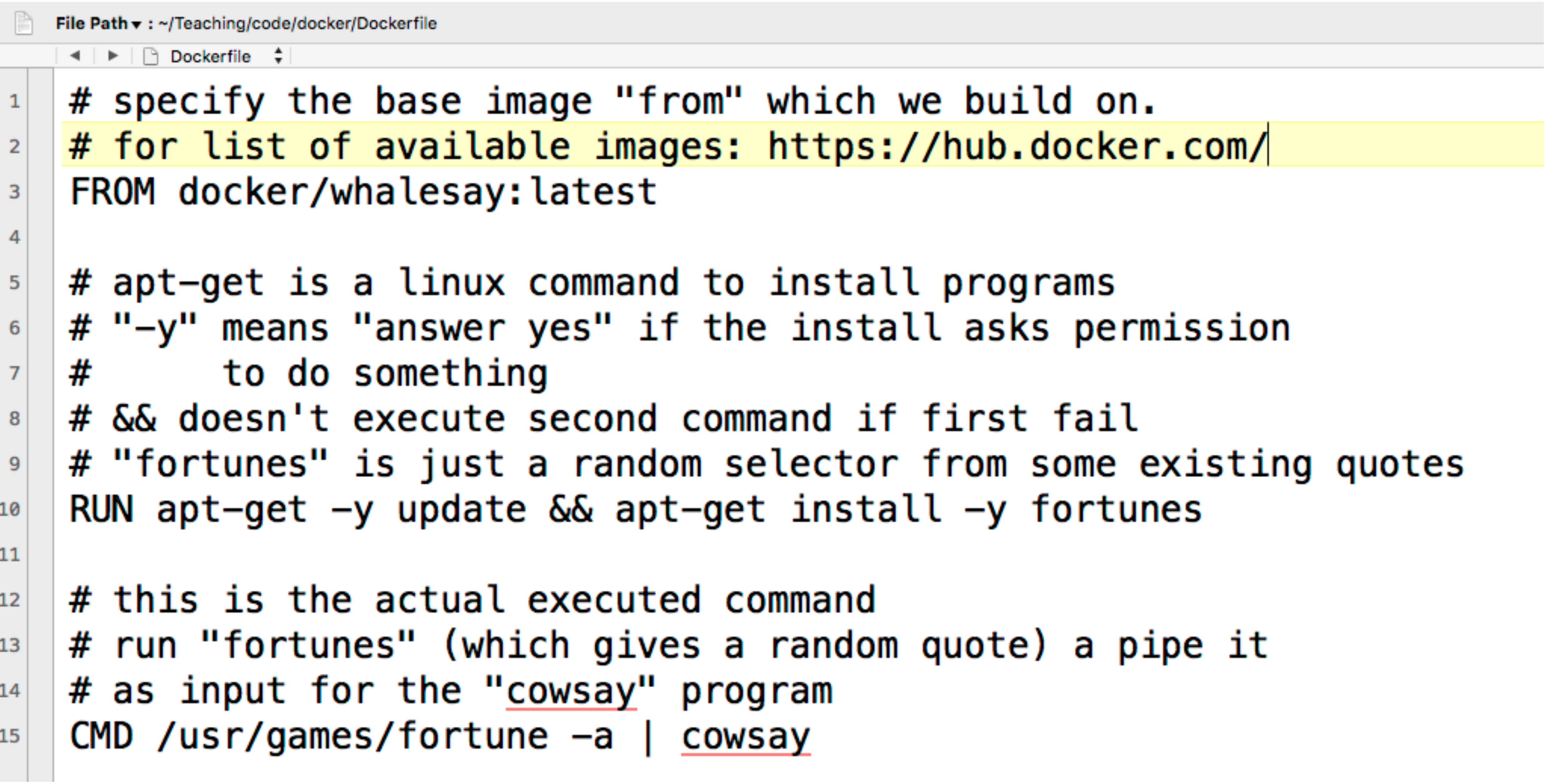
- https://docs.docker.com/get-started/
- https://hub.docker.com/r/docker/whalesay/
- ***docker run docker/whalesay cowsay boo***
  - This will install the image “docker/whalesay”, and run it with input “cowsay boo”
  - First time you run it, the “docker/whalesay” image will be downloaded



```
Andreas-MBP-2:~ foo$ docker run docker/whalesay cowsay boo
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay

e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 _____
< boo >
 -----
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
Andreas-MBP-2:~ foo$ 
```

Custom Images

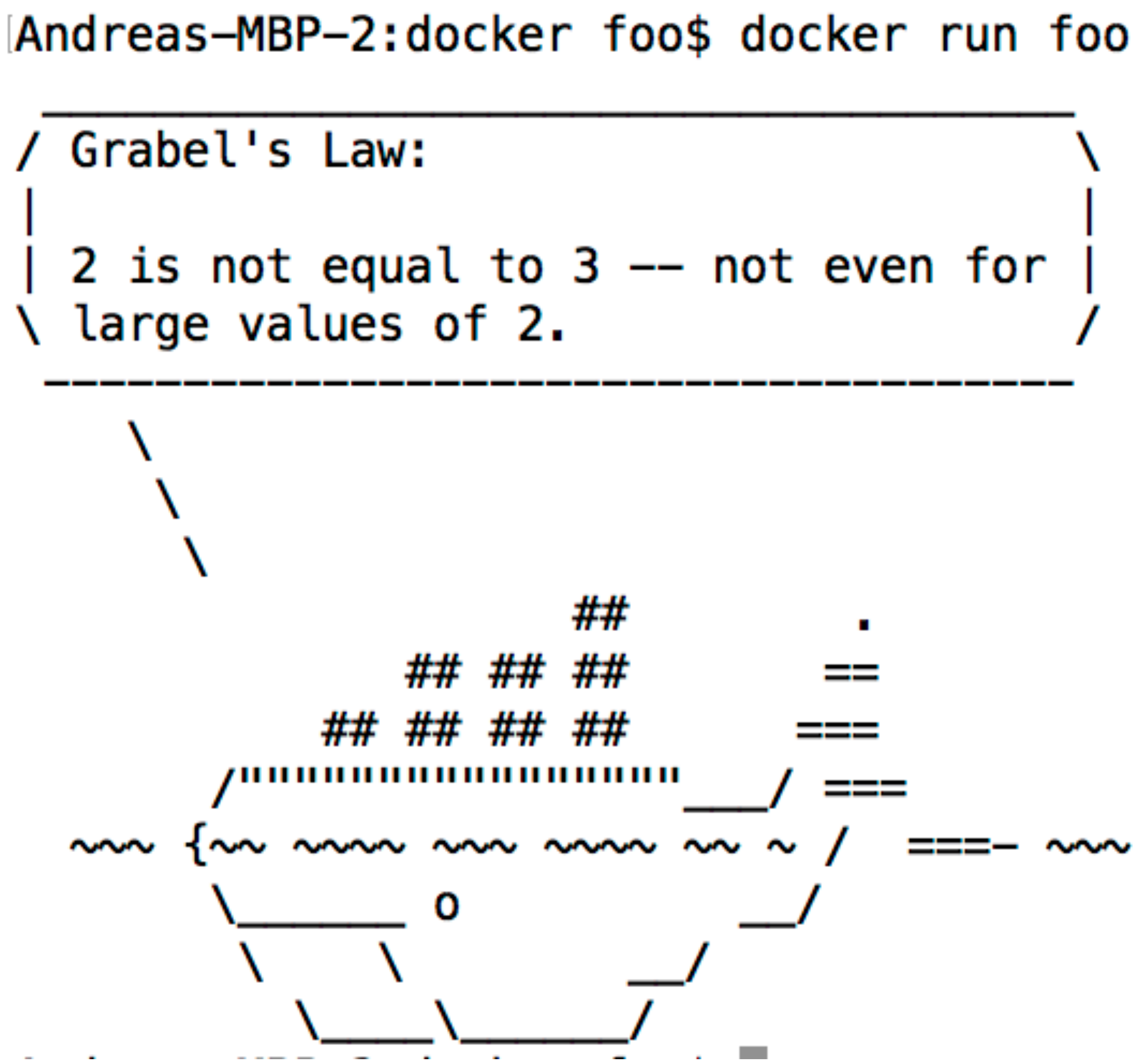
- Extend existing images to run the applications you develop
- Just need to create a text file called “Dockerfile”
- 3 “main” parts (there are more…)
  - **FROM:** specify the base OS image
  - **RUN**: execute commands in the virtual OS to set it up, like installing programs or create files/directories
  - **CMD**: the actual command for your application
  - **#** are comments
- ***docker build -t <name> .***
  - Create an image with name <name>, from the Dockerfile in the current “.” folder
- ***docker run <name>***
  - Run the give image
- **docker ps**
  - Show running images
- **docker stop <id>**
  - Stop the given running image. Note: an image can be run in several instances, with different ids



File Path ▾ : ~/Teaching/code/docker/Dockerfile

Dockerfile

```dockerfile
# specify the base image "from" which we build on.
# for list of available images: https://hub.docker.com/
FROM docker/whalesay:latest

# apt-get is a linux command to install programs
# "-y" means "answer yes" if the install asks permission
#       to do something
# && doesn't execute second command if first fail
# "fortunes" is just a random selector from some existing quotes
RUN apt-get -y update && apt-get install -y fortunes

# this is the actual executed command
# run "fortunes" (which gives a random quote) a pipe it
# as input for the "cowsay" program
CMD /usr/games/fortune -a | cowsay
```



```
Andreas-MBP-2:docker foo$ docker run foo
 _____________________________________
/ Grabel's Law:                       \
|                                     |
| 2 is not equal to 3 -- not even for |
\ large values of 2.                  /
 -------------------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
```

Some Further Commands

- **ENV**: define an environment variable
- **COPY**: take a file X from your hard-disk, and copy it over to the Docker image at the given path Y
  - When Docker image runs, it can access X at path Y, even when you deploy the image on a remote server
- **WORKDIR**: specify the working directory for the executed commands
  - Think about it like doing “cd” to that folder, so all commands/files are relative to that folder, and you do not need to specify full path

Networking

- When you run a server on your local host, it will open a TCP port, typically 80 or 8080
- A server running inside Docker will open the same kind of ports, but those will not be visible from the *host* OS
- You need to explicitly make a mapping from *host* to *guest* ports
- Ex.: **docker run –p 80:8080 foo**
  - When we do a connection on localhost on port 80, it will be redirected to 8080 inside Docker

CTRL-C

- When running Docker (eg "*docker run*") in a terminal, you can use CTRL-C to stop it
- On some terminals it *might* happen that the image still run in background
- Use "*docker ps*" to check if indeed the case
- Use "*docker stop <id>*" to stop an image manually
- If you have Docker images running in the background, that can be a problem if they use TCP ports

Example: Running Postgres

- Docker Hub: https://hub.docker.com/
- Most major tools are released as well in Docker images
- Postgres: https://hub.docker.com/_/postgres/
- How to download it and run it? Simple, just type:
- **docker run -p 5432:5432 postgres**
  - note, Postgres expect connections on port 5432, so we need to make it accessible

Databases

The 3 Rules of Choosing a Database

1. New project or unsure what to do? **Choose Postgres**
2. If you are already using *MySQL* and migration to *Postgres* would be too expensive, can stick with *MySQL*
3. If you have a long experience with databases, know exactly what you are doing, and can measure objectively the performance benefits of different tradeoffs compared to just using *Postgres*, then, *and only then*, choose best database for the *specific* problem you are facing

Example: *MySQL*

- Open source, but own (and mainly developed) by *Oracle*… and let’s not forget that one of its main commercial products is *Oracle Database*…
  - so, yes, in theory those 2 databases are competitors…
- For most use cases, *MySQL* is on par with *Postgres,* but usually slower in adding new advanced features
  - eg support for NoSQL features like JSON data type, or SQL compliance

Example: *MongoDB*

- Most famous *NoSQL* database
  - very, very popular in tutorials… especially in NodeJS
- Meant for *documents*, not for data with *relations*
  - Usually documents are in JSON format, where the only relations are hierarchical, eg nested objects
- Can be *fast* and *easy* to set up…
- … but you need to *sacrifice ACID* for it…
  - eg, when you “save” some data, can be just cached, and not actually saved…
  - ACID transactions added in v4.0, in 2018…
- *Postgres/MySQL* can save JSON fields, and be very fast at it
  - eg, in 2014, Postgres was actually faster than MongoDB in benchmarks at dealing with JSON

MongoDB Cont.

- Might start with JSON *documents*… but then one day you need to add relations between data: *you are screwed*
  - “screwed” meaning ending up implementing JOINs at application level, which is a nightmare and very inefficient… and/or duplicate data, which need to be kept always in sync…
- Or even worse, choosing *MongoDB* even when you deal with relational data, just because of *hype*…
- …or when you do not really deal with the amount of data of Google/Amazon/etc…

But… MongoDB is “Web Scale”!

- https://www.youtube.com/watch?v=b2F-DItXtZs
- http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/
- **echo "MongoDB is Web Scale!" > /dev/null**
- Note: video is from 2010. At that time MongoDB was “rubbish”. Today is better
    - eg ACID transactions added in 2018

Cloud Deployment

Cloud Deployment

- Different companies provide cloud hosting solutions for your applications, which frees you from hardware issues, but for a price
- *Amazon Web Services* (AWS) is perhaps the most famous/used one
  - eg, *Netflix* runs on AWS
- *Automated scaling*: if you need more load, automatically rent more nodes, and automatically scale down if less load
  - this is also good for applications targeting a specific country (eg Norway), in which you will not get much load during the night

Definition of "Cloud"

There is no cloud
it's just someone else's computer

Heroku

- One of the main cloud providers
- At the time of this writing, it provides *easy* to use *free* hosting
  - note, this might change at any time
- Supporting NodeJS applications
  - and many others

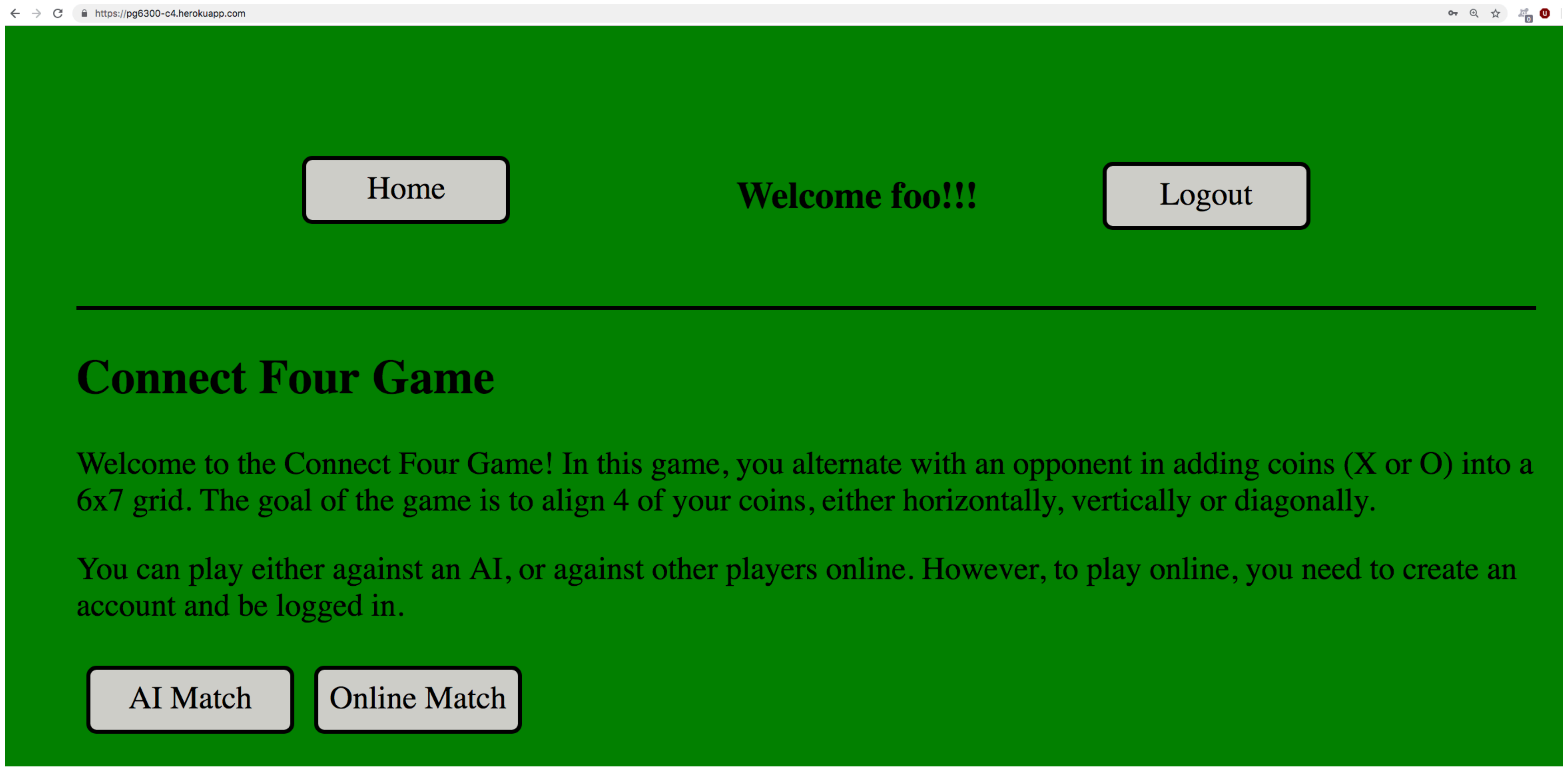
Using Heroku

- First you need to create an account at www.heroku.com
- Install *Heroku CLI*, which allows you to interact with Heroku from command line
- On the web interface, create an "app" with a name of your choice. In these slides, I will use "*pg6300-c4*"
  - as names are unique, you will need to choose a different name

From Command Line (CLI)

- **heroku plugins:install heroku-builds**
  - need to be run only once, to install the "*builds*" plugin
- **heroku login**
  - will setup credential for the other commands.
  - note: if using Windows, this might not work on GitBash, and need to do this command once from a regular Terminal
- **heroku builds:create -a pg6300-c4**
  - zip all your files in current folder, and deploy them in the app
  - note: use "*.gitignore*" to specify what to exclude

https://pg6300-c4.herokuapp.com/



Connect Four Game

Welcome to the Connect Four Game! In this game, you alternate with an opponent in adding coins (X or O) into a 6x7 grid. The goal of the game is to align 4 of your coins, either horizontally, vertically or diagonally.

You can play either against an AI, or against other players online. However, to play online, you need to create an account and be logged in.