Python + Docker for Big Data

EECS 4415: Big Data System (Fall 2018)

Agenda

1. Recap: Python
2. Introducing Docker
3. Installing Docker
4. Using Docker
5. Our Docker Environment
6. Examples
7. Python Code Snippets
8. Conclusion

Examples

0. Basic Docker
1. Basic Python and CSVs
2. Word Frequency
3. Great Lakes Water Quality

YORK UNIVERSITÉ UNIVERSITY

Recap: Python

- Dynamically-typed scripting language
- Very easy to learn
- Interactive front-end for C/C++ code
- Object-oriented
- Lots of libraries
  - Including tools for data analysis
- Powerful, scalable
  - Supporting tools that handle very large datasets

Introducing Docker

Introducing Docker

- Docker is a platform for developers and sysadmins to **develop, deploy, and run** applications with containers. The use of Linux containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.
  - **Containerization** is increasingly popular because containers are:
  - **Flexible**: Even the most complex applications can be containerized.
  - **Lightweight**: Containers leverage and share the host kernel.
  - **Interchangeable**: You can deploy updates and upgrades on-the-fly.
  - **Portable**: You can build locally, deploy to the cloud, and run anywhere.
  - **Scalable**: You can increase and automatically distribute container replicas.
  - **Stackable**: You can stack services vertically and on-the-fly.

Docker Basics

Images and Containers

- A container is launched by running an image. An image is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.
- A container is a runtime instance of an image--what the image becomes in memory when executed (that is, an image with state, or a user process). You can see a list of your running containers with the command, **docker ps**, just as you would in Linux.

Containers vs Virtual Machines

- A container runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.
- By contrast, a virtual machine (VM) runs a full-blown “guest” operating system with virtual access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.

Docker Basics: Images + Containers

- A Docker Image is the **template** (application plus required binaries and libraries) needed to build a running Docker Container (the running instance of that image). As templates, images are what can be used to share a containerized applications. Collections of images are stored/shared in registries like Docker Hub.

Why Docker?

- A lightweight approach that allows us to simulate an environment that has parallels to how one might interact with a cloud-based VM or container, without having the overhead and cost of setting up AWS or Azure instances.

- FYI: If you're intrigued:
  - https://aws.amazon.com/docker/
  - https://docs.docker.com/docker-for-azure/
  - https://azure.microsoft.com/en-us/services/kubernetes-service/

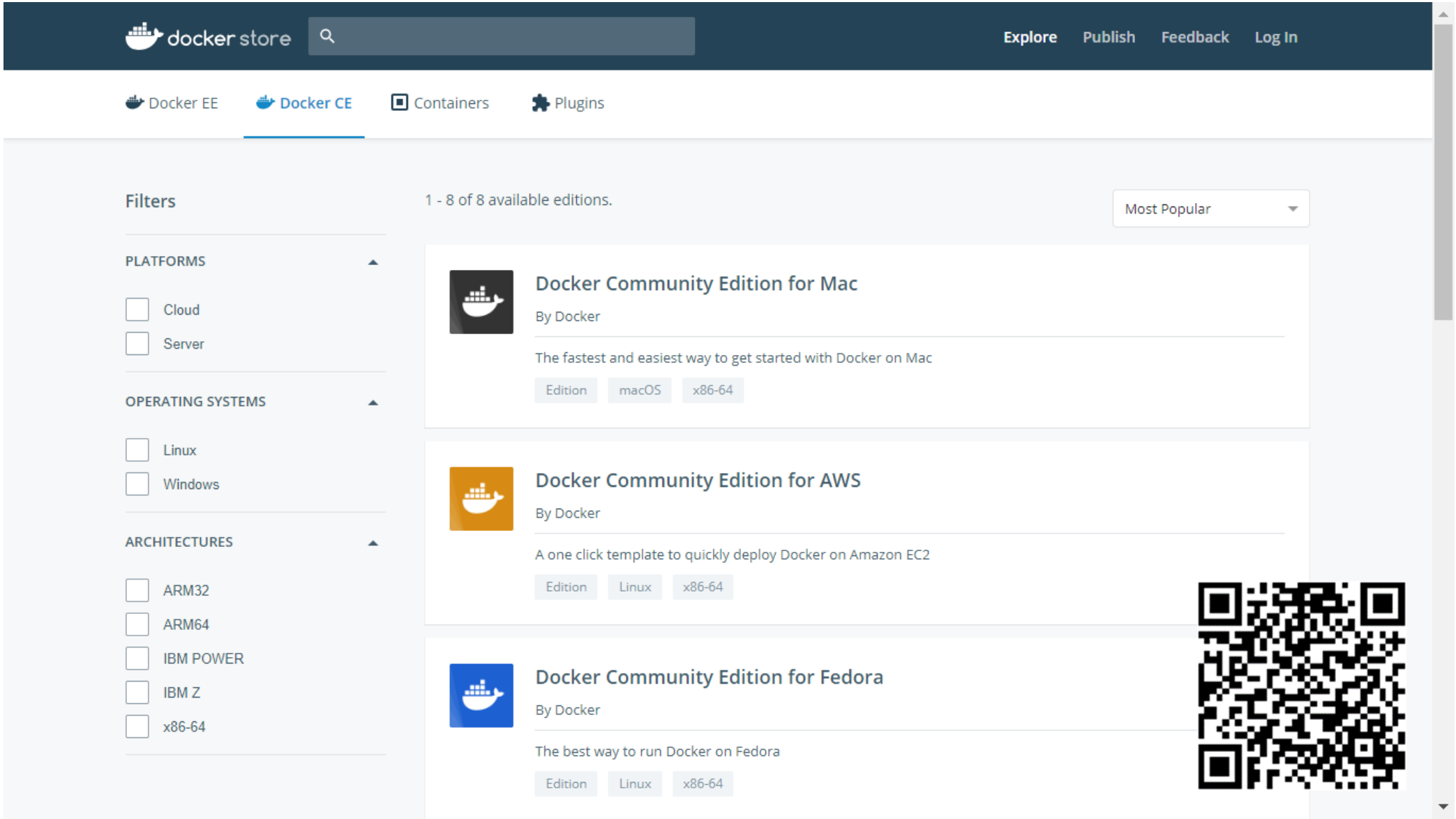
YORK UNIVERSITÉ UNIVERSITY

Installing Docker

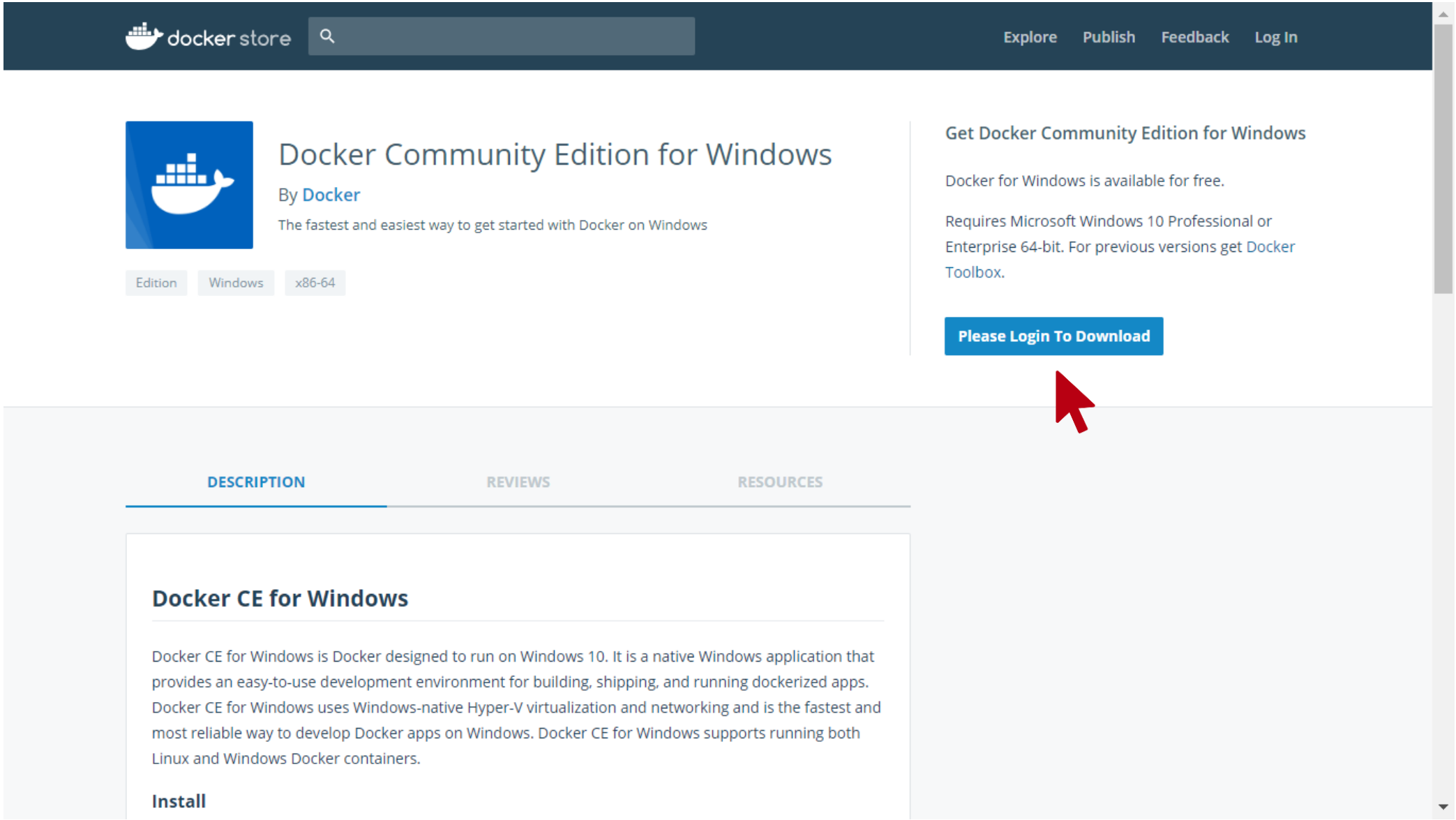
For Windows, Mac & Linux

YORK UNIVERSITÉ UNIVERSITY

Go to https://store.docker.com/search?type=edition&offering=community

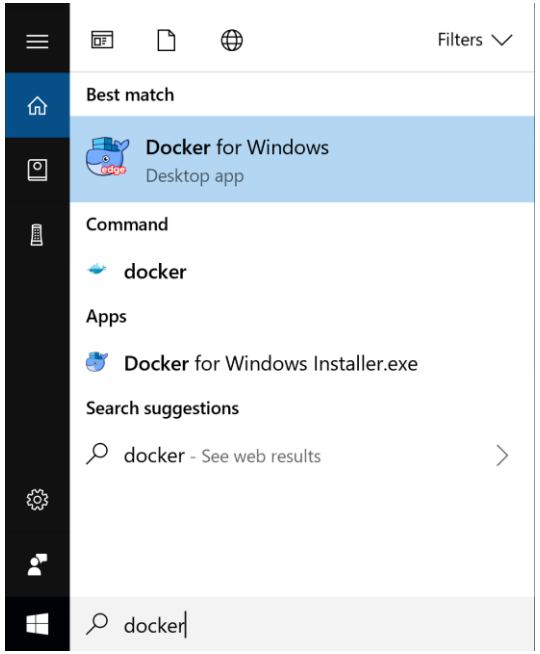


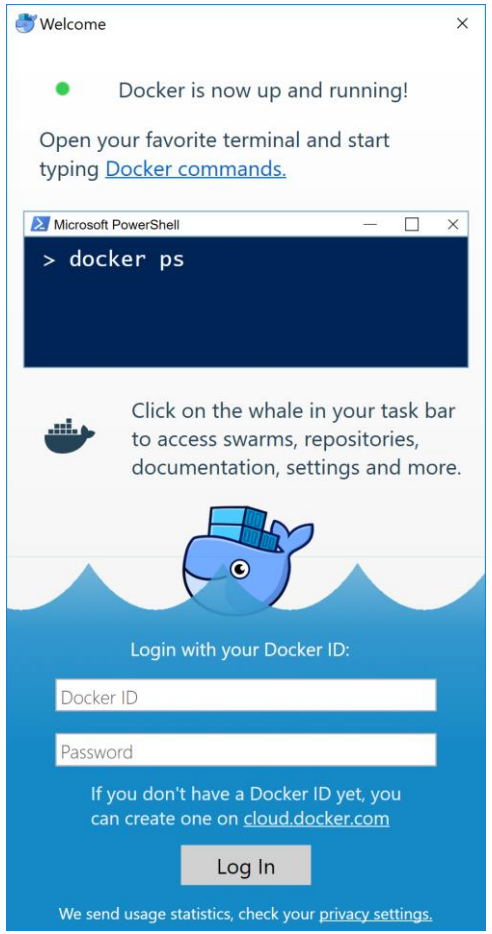
Select the Edition that matches your computer's OS.
Then click **Get Docker**. You may need to login / create a Docker account first.



Installing Docker: For Windows

1. Enable Hyper-V:
   - https://blogs.technet.microsoft.com/canitpro/2015/09/08/step-by-step-enabling-hyper-v-for-use-on-windows-10/
   - Steps 1 through 2(6)
2. Run downloaded EXE as Administrator
3. Start the Docker client by opening **Docker for Windows** via the Start Menu
4. Open Docker controls via the notification area (system tray).

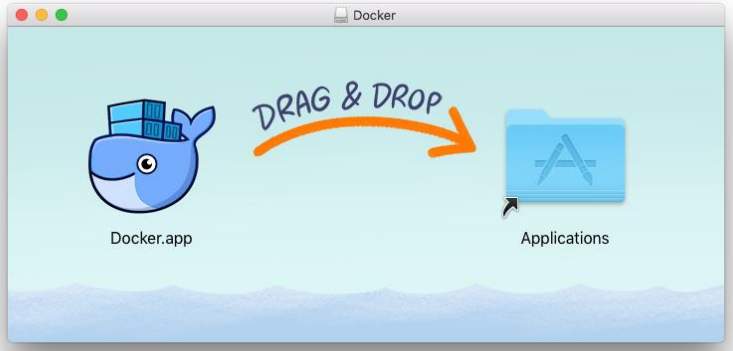


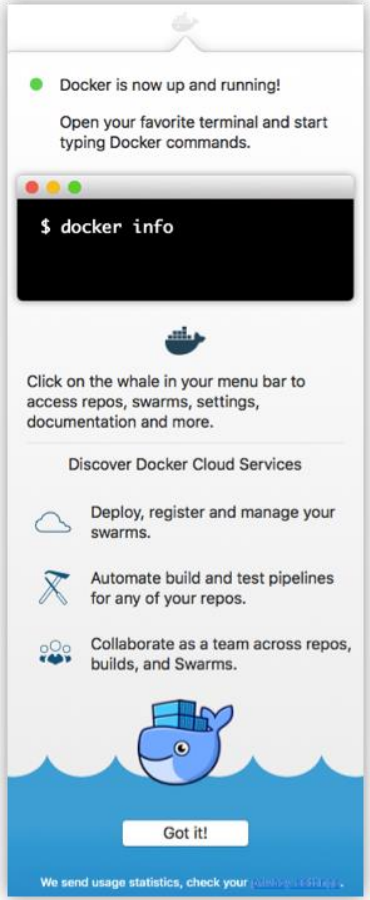


Refer to: https://docs.docker.com/install/

Installing Docker: For Mac

1. Run downloaded DMG as Super user
2. Start the Docker client by opening **Docker for Mac** via the Launchpad
3. Open Docker controls via the menu bar.





Refer to: https://docs.docker.com/install/

YORK UNIVERSITÉ UNIVERSITY

For non Windows 10 Pro/Edu users:

- If you have Windows 10 Home:
  - Upgrade to Windows 10 Pro via https://webapp.eecs.yorku.ca/imagine/
  - These editions work:
    - Windows 10 Education, Version 1803 32/64-bit
    - Windows 10 (Multiple Editions), Version 1803 32/64-bit

- If you have Windows 7 or 8.1:
  - Um… Upgrade, already?
  - But if you must insist:
    - https://docs.docker.com/toolbox/toolbox_install_windows/
    - Legacy desktop solution. **Docker Toolbox** is for older Mac and Windows systems that do not meet the requirements of Docker for Mac and Docker for Windows. We recommend updating to the newer applications, if possible.

Installation: Notes for VM Users

- For Windows users that used VirtualBox or VMware
  - Hyper-V is need for **Docker for Windows**
  - Hyper-V cannot be used alongside other hypervisors
  - Two options:
    1. Use: **Docker Toolbox** https://docs.docker.com/toolbox/toolbox_install_windows/
    2. Toggle on/off hypervisorlaunchtype at startup

       INVOLVES EDIT BOOT MENU

       PROCEED AT YOUR OWN RISK! YOU BETTER KNOW WHAT YOU'RE DOING!

       https://www.hanselman.com/blog/SwitchEasilyBetweenVirtualBoxAndHyperVWithABCDEditBootEntryInWindows81.aspx

YORK UNIVERSITÉ UNIVERSITY

Installation: Alternatives

- For those comfortable using command-lines
- For those who don’t want to create a Docker account
  - Package Manager:
    - Chocolatey: Windows
    - Homebrew: macOS
    - Snap: Ubuntu / Linux

Alternative for Windows:

1. Enable Hyper-V:
   - https://blogs.technet.microsoft.com/canitpro/2015/09/08/step-by-step-enabling-hyper-v-for-use-on-windows-10/
   - Steps 1 through 2(6)
2. Install chocolatey first:
   - https://chocolatey.org/install
3. In PowerShell as Administrator:
   - `PS> choco install docker-for-windows`
4. Start the Docker client by opening **Docker for Windows** via the Start Menu
5. Open Docker controls via the notification area (system tray).

YORK UNIVERSITÉ UNIVERSITY

Alternative for Mac:

1. Install Homebrew:
   - https://brew.sh/
   - https://docs.brew.sh/Installation
2. In the Terminal as Administrator, run:
   - `$ brew install docker`
3. Start the Docker client by opening **Docker for Mac** via the Launchpad
4. Open Docker controls via the menu bar.

YORK UNIVERSITÉ UNIVERSITY

Alternative for Linux:

- In the terminal, run:
  - `$ sudo snap install docker`
  - https://github.com/docker/docker-snap/blob/master/README.md
  - https://snapcraft.io/docker

- If you use other methods or can’t install **snap**…
  - https://docs.docker.com/install/#supported-platforms
  - Install **Docker CE**

Alternative for Linux (Ubuntu):

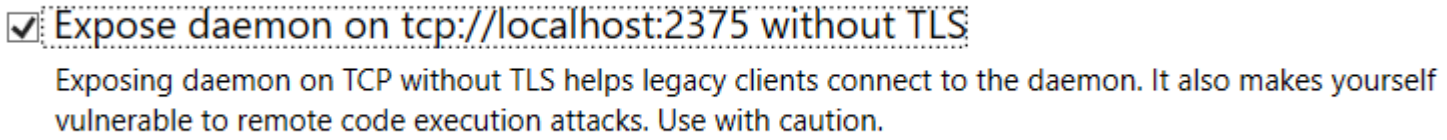
PROCEED AT YOUR OWN RISK!

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo usermod -aG docker $USER

export DOCKER_COMPOSE_VERSION=1.22.0
sudo curl -L https://github.com/docker/compose/releases/download/${DOCKER_COMPOSE_VERSION}/docker-compose-$(uname –s)-$(uname –m) -o /usr/local/bin/docker-compose
sudo chmod a+rx /usr/local/bin/docker-compose

sudo -i << 'EOF'
curl -L https://raw.githubusercontent.com/docker/docker-ce/master/components/cli/contrib/completion/bash/docker > /etc/bash_completion.d/docker
curl -L https://raw.githubusercontent.com/docker/compose/$(docker-compose version --short)/contrib/completion/bash/docker-compose > /etc/bash_completion.d/docker-compose
EOF

# Uninstall:
# sudo apt-get remove docker-ce
# sudo apt autoremove
# sudo rm /usr/local/bin/docker-compose
# sudo rm /etc/bash_completion.d/docker
# sudo rm /etc/bash_completion.d/docker-compose
```

YORK UNIVERSITÉ UNIVERSITY

Alternative for Windows (WSL):

- For **Windows Subsystem for Linux**, install Docker CE with instructions for **Alternative for Linux (Ubuntu)**, but requires the below setting be enabled in the Docker Settings. There are more secure, but very involved workarounds.



https://medium.com/@sebagomez/installing-the-docker-client-on-ubuntus-windows-subsystem-for-linux-612b392a44c4
https://nickjanetakis.com/blog/setting-up-docker-for-windows-and-wsl-to-work-flawlessly
https://blogs.msdn.microsoft.com/commandline/2017/12/08/cross-post-wsl-interoperability-with-docker/
https://raesene.github.io/blog/2018/03/29/WSL-And-Docker/
https://docs.docker.com/engine/security/https/#create-a-ca-server-and-client-keys-with-openssl
https://blogs.technet.microsoft.com/stefan_stranger/2018/04/02/access-my-docker-for-windows-kubernetes-cluster-from-debian-wsl/

YORK UNIVERSITÉ UNIVERSITY

Additional Notes

- Recommend installing and using Git
- For Windows, Git comes with a Bash terminal
  - The Git-Bash / MinGW terminal works with Docker with some caveats requiring some workarounds.
  - PowerShell and CMD should work will Docker
  - Windows Subsystem for Linux can install Docker CE, but can only be used as a client not the engine. See here.

Using Docker

Docker commands

YORK UNIVERSITÉ UNIVERSITY

Using Docker: The Basic Commands

- docker images — List images
- docker pull — Pull an image or a repository from a registry
- docker ps — List containers
- docker run — Run a command in a new container
- docker rm — Remove one or more containers
- docker help — Help about the command

https://docs.docker.com/get-started/

https://docs.docker.com/engine/reference/commandline/docker/

YORK UNIVERSITÉ UNIVERSITY

docker images

- The default docker images will show all top level images, their repository and tags, and their size. The docker images command takes an optional [REPOSITORY[:TAG]] argument that restricts the list to images that match the argument. If you specify REPOSITORY but no TAG, the docker images command lists all images in the given repository.

```
$ docker images

REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
<none>                      <none>              77af4d6b9913        19 hours ago        1.089 GB
committ                     latest              b6fa739cedf5        19 hours ago        1.089 GB
<none>                      <none>              78a85c484f71        19 hours ago        1.089 GB
docker                      latest              30557a29d5ab        20 hours ago        1.089 GB
<none>                      <none>              5ed6274db6ce        24 hours ago        1.089 GB
postgres                    9                   746b819f315e        4 days ago          213.4 MB
postgres                    9.3                 746b819f315e        4 days ago          213.4 MB
postgres                    9.3.5               746b819f315e        4 days ago          213.4 MB
postgres                    latest              746b819f315e        4 days ago          213.4 MB
```

docker pull

- Most of your images will be created on top of a base image from the **Docker Hub** registry. Docker Hub contains many pre-built images that you can pull and try without needing to define and configure your own. To download a particular image, or set of images (i.e., a repository), use docker pull.

- Find images & documentation on: https://hub.docker.com/

docker pull

- If no tag is provided, Docker Engine uses the :latest tag as a default. This command pulls the debian:latest image:

```
$ docker pull debian

Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

docker pull

docker pull python

- Downloads the latest python image
- Same as: docker pull python:latest

docker pull python:3.7

- Downloads the python image for version 3.7

docker pull eecsyorku/eecs4415

- Downloads the latest version of the class's image
- Same as: docker pull eecsyorku/eecs4415:latest

docker ps

- List containers

```
$ docker ps

CONTAINER ID        IMAGE                        COMMAND                  CREATED             STATUS              PORTS               NAMES
4c01db0b339c        ubuntu:12.04                 bash                     17 seconds ago      Up 16 seconds       3300-3310/tcp       webapp
d7886598dbe2        crosbymichael/redis:latest   /redis-server --dir      33 minutes ago      Up 33 minutes       6379/tcp            redis,webapp/db
```

- The docker ps command only shows running containers by default. To see all containers, use the -a (or --all) flag:

docker run

- The docker run command first creates a writeable container layer over the specified image, and then starts it using the specified command.
- A stopped container can be restarted with all its previous changes intact using docker start. See docker ps -a to view a list of all containers.

docker run

docker run -it -v $PWD:/app -w /app python:3.7 bash

- Start a new python:3.7 container
- Run the bash command within the container
- -v $PWD:/app same as --volume $PWD:/app
  - Mount a volume with the current working directory to the /app path in container, so you can access files within the container from /app directory.
  - If -v /doesnt/exist:/foo, when the host directory of a bind-mounted volume doesn't exist, Docker will automatically create this directory on the host for you. In the example above, Docker will create the /doesnt/exist folder before starting your container.

See: https://docs.docker.com/engine/reference/commandline/run/

YORK UNIVERSITÉ UNIVERSITY

docker run

docker run -it -v $PWD:/app -w /app python:3.7 bash

- -w /app lets the command being executed inside directory given, here /app. If the path does not exist it is created inside the container.
- -it instructs Docker to allocate a pseudo-TTY connected to the container's stdin; creating an interactive bash shell in the container.
  - -t, --tty Allocate a pseudo-TTY
  - -i, --interactive Keep STDIN open even if not attached
- --rm automatically remove the container when it exits

See: https://docs.docker.com/engine/reference/commandline/run/

YORK UNIVERSITÉ UNIVERSITY

docker rm

docker rm $(docker ps -a -q)

- This command will delete all stopped containers. The command docker ps -a -q will return all existing container IDs and pass them to the rm command which will delete them. Any running containers will not be deleted.

Other Common Commands

- docker start Start one or more stopped containers
- docker build Build an image from a **Dockerfile**
- docker cp Copy files/folders between a container and the local filesystem
- docker exec Run a command in a running container
- docker kill Kill one or more running containers
- docker pause Pause all processes within one or more containers
- docker stop Stop one or more running containers

Usage Notes

- For Windows Users using Git-Bash or MinGW
  - You may need to prefix docker run with winpty
  - When setting volumes or working directories absolute paths must be prefixed with an extra /.
  - For instance: docker run -it -v $PWD:/app –w /app ubuntu bash
  - Becomes: winpty docker run -it -v /$PWD:/app –w //app ubuntu bash

YORK UNIVERSITÉ UNIVERSITY

Our Docker Environment

eecs4415 Image

Our class's Docker image is now available at: **eecsyorku/eecs4415**

See docs: https://hub.docker.com/r/eecsyorku/eecs4415/

Download:	docker pull eecsyorku/eecs4415
Run:	docker run –it –v $PWD:/app eecsyorku/eecs4415 bash
Python Shell:	docker run –it eecsyorku/eecs4415 python3
Python Script:	docker run –v $PWD:/app eecsyorku/eecs4415 python3 /app/main.py

YORK UNIVERSITÉ UNIVERSITY

Examples

Demos

0. Basic Docker

```
$ docker pull hello-world
$ docker images hello-world
$ docker run hello-world
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

0. Basic Docker

```
$ docker run -it ubuntu bash
$ uname -a
Linux VC003 4.4.0-17134-Microsoft #285-Microsoft Thu Aug 30 17:31:00 PST 2018 x86_64
x86_64 x86_64 GNU/Linux

$ echo 'uname -a' > script.sh
$ docker run -it --rm -v $PWD:/home/ubuntu -w /home/ubuntu ubuntu bash script.sh
Linux 0ba427dd6d8d 4.9.93-linuxkit-aufs #1 SMP Wed Jun 6 16:55:56 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux

$ docker run –it --rm python python
Python 3.7.0 (default, Sep  5 2018, 03:25:31)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

1. Basic Python and CSVs

- **Demonstrates:**
  - Running Python scripts in Docker
  - Reading and parsing STDIN into words
  - Reading and writing CSV files
  - Printing output to the terminal

1. Basic Python and CSVs

- **Usage:**
  - Start Python Docker container with volume mounted
  - Run each of the python scripts in the directory to observe the output

```
$ docker run -it –v $PWD:/usr/src/app –w /usr/src/app python bash
root:/usr/src/app# ls -la
root:/usr/src/app# python readstdin.py < inputs/input.txt
root:/usr/src/app# python readstdin.py < inputs/input.csv
root:/usr/src/app# python readtxt.py
root:/usr/src/app# python readcsv.py
```

2. Word Frequency

- A Python program (use **Python 3**) to find the top ten words in an input stream by number of occurrences and to make a bar-chart plot and CSV output of them.
- Based on: https://www.eecs.yorku.ca/course_archive/2017-18/F/4415/project/zipf/
- Reads book.txt as input (downloaded from Dracula by Bram Stoker on The Project Gutenberg website).
- Eliminate **stopwords** — the very common words in English — and words just one character long as not being interesting. When tokenizing, split with “[\W+_]” (This splits on whitespace and underscores “_”). We won't worry about preserving words with apostrophes for now (e.g., “won't”). If we were to extend our program to be more robust and useful later, we surely would improve on our tokenizer, or find a good library for it.
- Use the file stopwords.txt for stopwords. Our program can read in the file and make a *stopword* dictionary to use to check against to eliminate the stopwords as we are parsing the input stream.

YORK UNIVERSITÉ UNIVERSITY

2. Word Frequency

- **Demonstrates**:
  - Reading and parsing text files into words
  - Multiple Python files
  - Plotting bar graphs with matplotlib
  - Writing to a CSV file

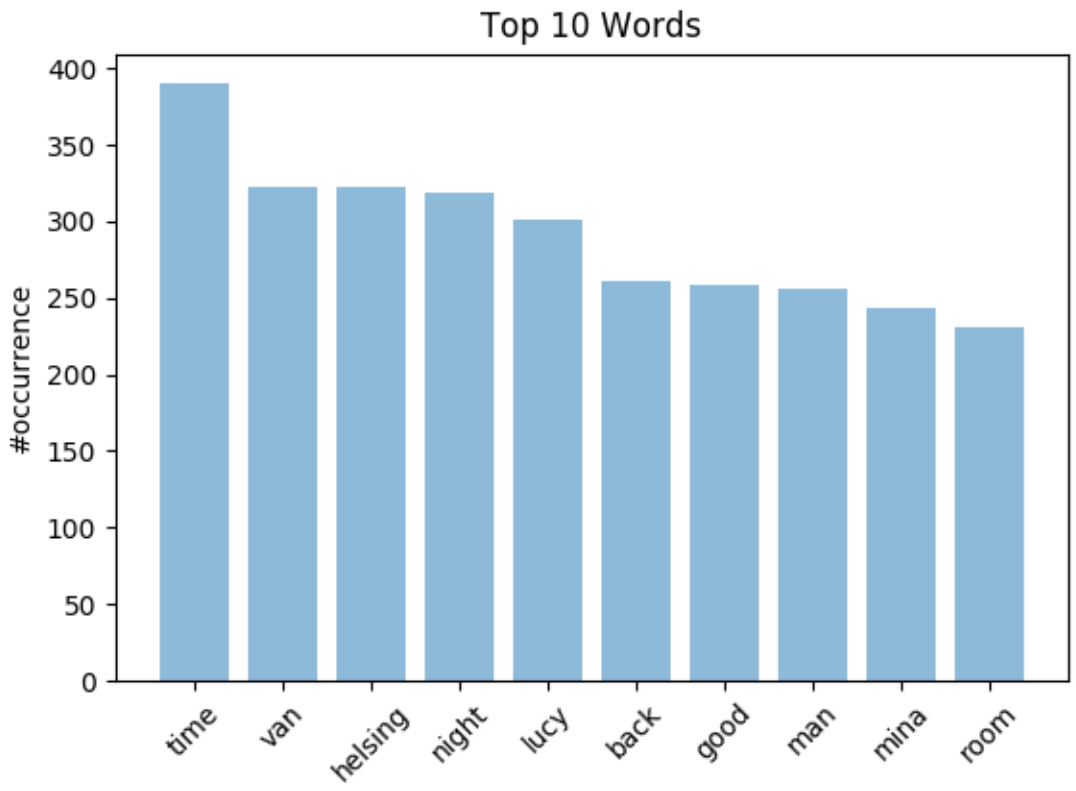
2. Word Frequency

- **Usage:**
  - Start Python Docker container with volume mounted
  - Run **./start.sh** to install the python library dependencies
  - Run **python src/main.py**
  - Output graph and CSV should be found in the outputs/ directory.

```
$ docker run -it –v $PWD:/usr/src/app –w /usr/src/app python bash
root:/usr/src/app# ls -la
root:/usr/src/app# ./start.sh
root:/usr/src/app# python src/main.py
root:/usr/src/app# ls -la outputs/
```

2. Word Frequency

- The Bar Graph



- The CSV

	A	B
1	Word	Frequency
2	time	390
3	van	323
4	helsing	323
5	night	319
6	lucy	301
7	back	261
8	good	258
9	man	256
10	mina	244
11	room	231

3. Great Lakes Water Quality

- **Great Lakes Water Quality Monitoring and Surveillance Data**
  - Water quality and ecosystem health data collected in the Great Lakes and priority tributaries to determine baseline water quality status, long term trends and spatial distributions, the effectiveness of management actions, determine compliance with water quality objectives and identify emerging issues are included in this dataset.
  - http://data.ec.gc.ca/data/substances/monitor/great-lakes-water-quality-monitoring-and-aquatic-ecosystem-health-data/great-lakes-water-quality-monitoring-and-surveillance-data/
  - 5 Datasets: Lake Ontario, Lake Erie, Lake Huron, Lake Superior, & Georgian Bay
  - Contains 2106 different measurement methods (codes) for assessing water quality
  - Format: CSV
- **Goal: Select a few methods, and output a line graph of the daily averages of the measurement over time. Visualize the change in the measurements.**

YORK UNIVERSITÉ UNIVERSITY

3. Great Lakes Water Quality

- **Demonstrates:**
  - Handling command line arguments
  - Working with multiple dataset files
  - Reading and parsing multiple CSV files
  - Multiple classes
  - Plotting line graphs with matplotlib

3. Great Lakes Water Quality

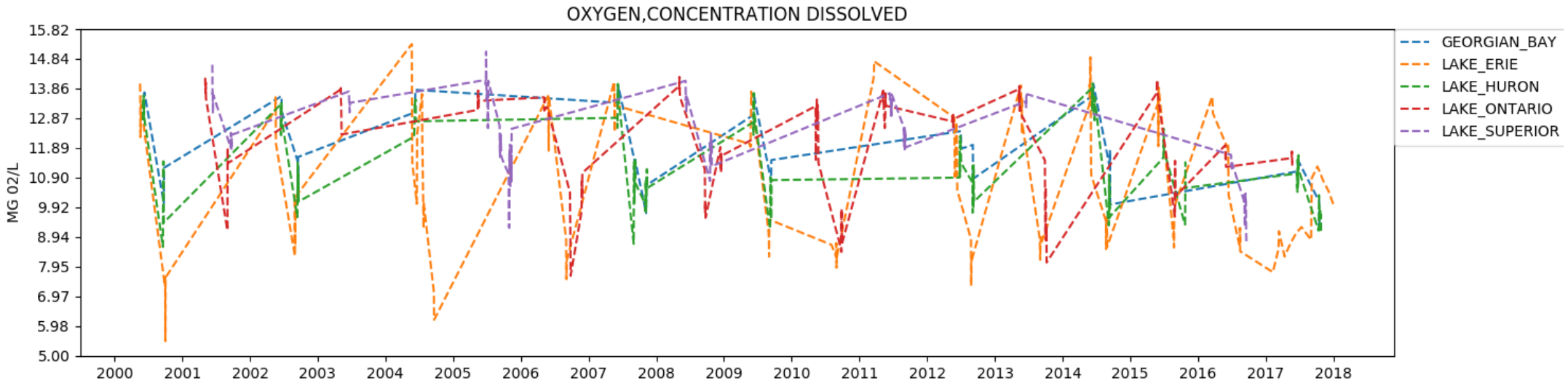
- **Usage:**
  - Start Python Docker container with volume mounted
  - Run **./start.sh** to download the datasets and install the python library dependencies
  - Run **python src/main.py** with method codes as arguments. For instance:
    - 245 -- OXYGEN,CONCENTRATION DISSOLVED
    - 247 -- OXYGEN,% SAT. DISSOLVED
    - 270 -- AMMONIA NITROGEN,SOLUBLE
  - Output graphs should be found in the outputs/ directory.

```
$ docker run -it –v $PWD:/usr/src/app –w /usr/src/app python bash
root:/usr/src/app# ls -la
root:/usr/src/app# ./start.sh
root:/usr/src/app# python src/main.py 245 247 270
root:/usr/src/app# ls -la outputs/
```

YORK UNIVERSITÉ UNIVERSITY

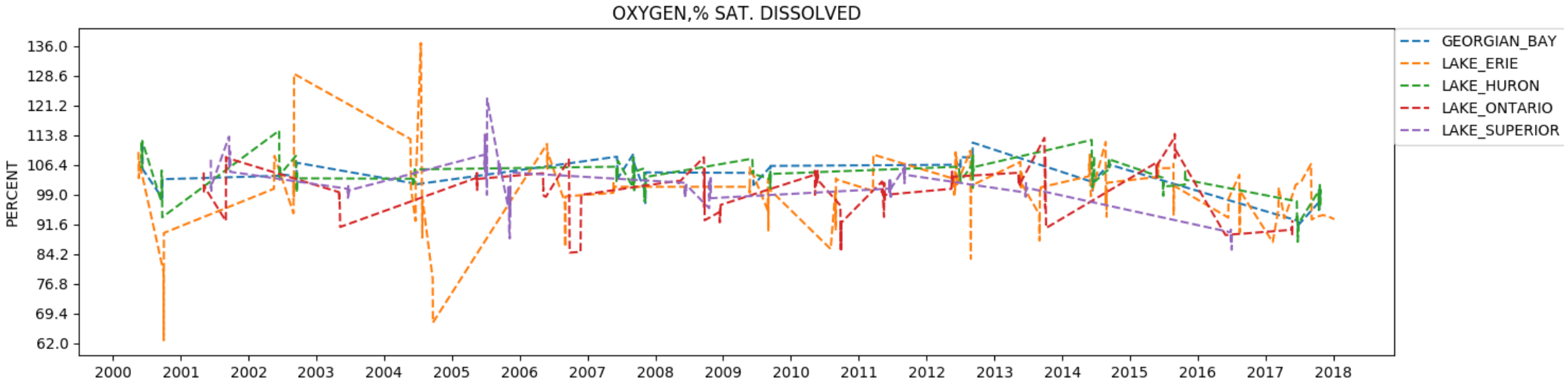
3. Great Lakes Water Quality

- 245 -- OXYGEN,CONCENTRATION DISSOLVED



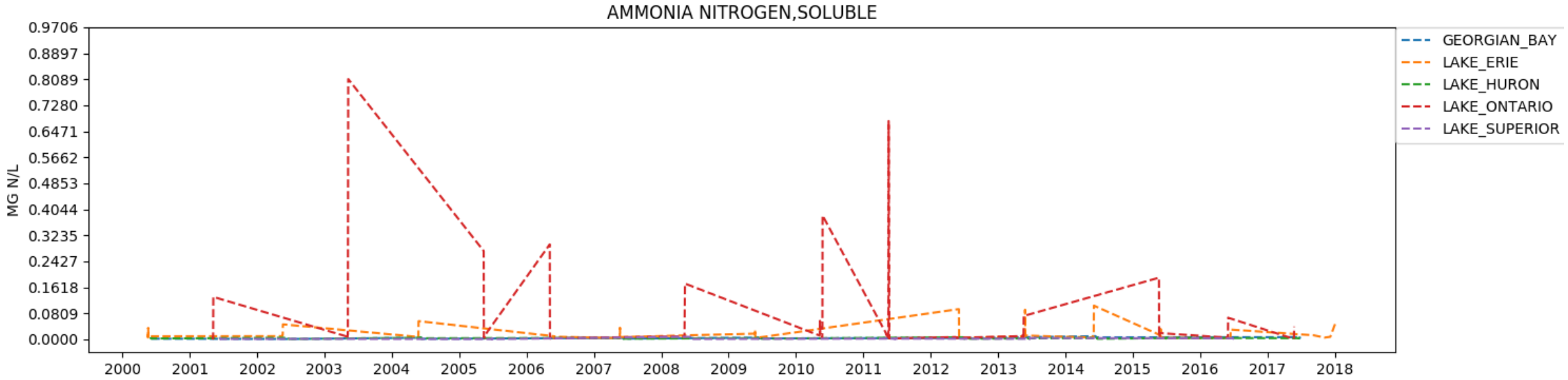
3. Great Lakes Water Quality

- 247 -- OXYGEN,% SAT. DISSOLVED



3. Great Lakes Water Quality

- 270 -- AMMONIA NITROGEN,SOLUBLE



YORK UNIVERSITÉ UNIVERSITY

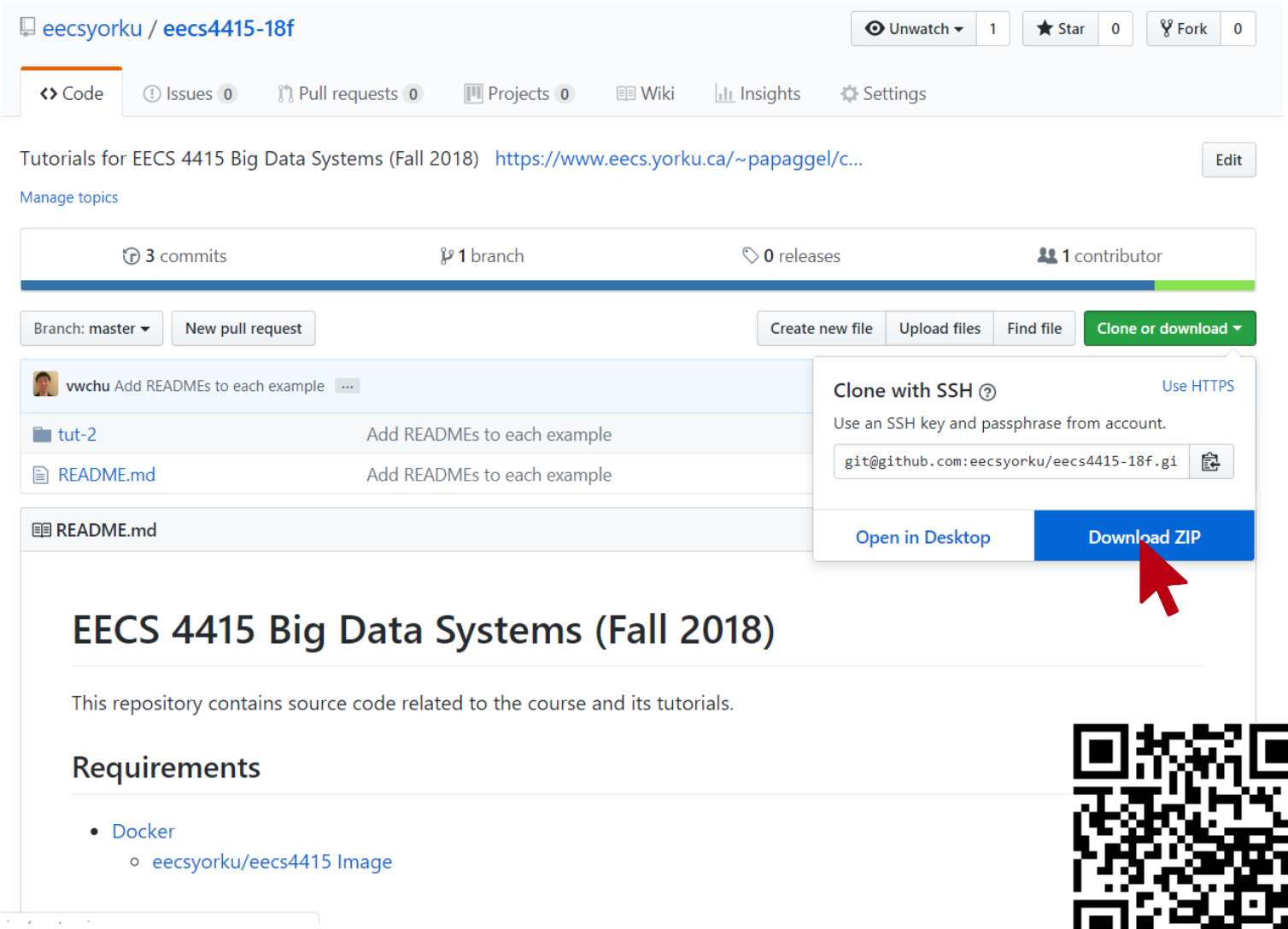
Download Examples

https://github.com/eecsyorku/eecs4415-18f

Download the ZIP

Or if you know how to use Git:

- git clone the project onto your computer and you should be able to pull new changes as we have more tutorial sessions and update the code.



Python Code Snippets

Main Code

```
if __name__ == "__main__":
  # Your main code goes here, or
  # call your main method here.
  main()
```

Arguments / STDIN / STDOUT

```
import sys

arguments = sys.argv[1:]
print(sys.stdin)
sys.stdout.write('Text to print.\n')
```

Classes:

```
class MyNameClass:
  """Documentation for my MyNameClass class"""

  def __init__(self, name):
    """
      Initializes the object. (constructor)
      The keyword 'self', refers object (like 'this' in Java)
    """
    self.name = name

  def __str__(self):
    """
      Returns a string representation of the object
      Equivalent to ‘toString()' in Java
    """
    return self.name
```

Note:

- Methods prefixed by ‘__’ are usually reserved for system defined methods like **__init__** or **__str__**.
- Methods prefixed by underscore ‘_’ are private methods within class.
- All other methods (not prefixed by underscore ‘_’) are public methods and accessible to calls outside of the class.

YORK UNIVERSITÉ UNIVERSITY

Reading a File / STDIN

```
import sys

with open(filepath, 'r') as f:
  for line in iter(f):
    print(line)

for line in sys.stdin:
  print(line)
```

Reading & Parsing Words in Plain Text File

```
import sys
import re

with open('essay.txt', 'r') as f:
  for line in iter(f):
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words, by whitespace
    words = filter(None, re.split('\W+', line))
    # increase counters
    for word in words:
      # write the results to STDOUT (standard output), in all lowercase
      print(word.lower())
```

YORK UNIVERSITÉ UNIVERSITY

Reading Words with Iterators

```
import sys
import re

def iterate_words(textfile):
  with open(textfile, 'r') as f:
    for line in iter(f):
      line = line.strip()
      words = filter(None, re.split('\W+', line))
      for word in words:
        # Use the yield keyword to specify the next iteration item
        yield word.lower()

def process_words(textfile):
  for word in iterate_words(textfile):
    print(word)
```

YORK UNIVERSITÉ UNIVERSITY

Reading & Parsing CSV

```
import csv

with open('names.csv', 'r') as csvfile:
  reader = csv.DictReader(csvfile)
  for row in reader:
    print(row['first_name'], row['last_name’])
```

See: https://docs.python.org/3/library/csv.html

Writing Files

```
with open(output, 'w') as f:
  f.write('Text to write.\n')
```

Writing CSV

```
import csv

with open('names.csv', 'w') as csvfile:
  fieldnames = ['first_name', ‘last_name']
  writer = csv.DictWriter(csvfile, fieldnames = fieldnames)
  writer.writeheader()
  writer.writerow({'first_name': 'Baked', 'last_name': 'Beans'})
  writer.writerow({'first_name': 'Lovely', 'last_name': 'Spam'})
  writer.writerow({'first_name': 'Wonderful', 'last_name': 'Spam’})
```

See: https://docs.python.org/3/library/csv.html

YORK UNIVERSITÉ UNIVERSITY

Find CSVs in a Directories

```
import os
import mimetypes
from os import path

__dirpath__ = path.dirname(__file__) # directory containing this script
datapath = path.join(__dirpath__, '../data')
for f in os.listdir(datapath):
  f = path.join(datapath, f)
  if path.isfile(f) and mimetypes.guess_type(f)[0] == 'text/csv':
    print(f)                                  # Full file path
    print(path.basename(f))                   # Just the filename
    print(path.splitext(path.basename(f))[0]) # Filename without .csv
```

YORK UNIVERSITÉ UNIVERSITY

Plot Bar Graph & Save to PNG

See: https://matplotlib.org/

```
import matplotlib; matplotlib.use('Agg')
import numpy as np
import matplotlib.pyplot as plot

def plot_figure(output, keys, values, ylabel, title):
  """Plot bar chart with the given values and output to the given file."""
  ypos = np.arange(len(keys))
  plot.figure()
  plot.bar(ypos, values, align = 'center', alpha = 0.5)
  plot.xticks(ypos, keys, rotation = 45)
  plot.ylabel(ylabel)
  plot.title(title)
  plot.savefig(output)
```

Plot Line Graph & Show

See: https://matplotlib.org/

```
import matplotlib; matplotlib.use('Agg')
import numpy as np
import matplotlib.pyplot as plot

def plot_figure(x, y, label, ylabel, title):
  """Plot line chart with the given values and show plot in a new window."""
  plot.figure()
  plot.plot(x, y, '--', label = label)
  plot.legend(bbox_to_anchor = (1, 1), loc = 'upper left', borderaxespad = 0.)
  plot.xticks(np.arange(min(x), max(x)))
  plot.yticks(np.arange(min(y), max(y)))
  plot.ylabel(ylabel)
  plot.title(title)
  plot.show()
```

Installing External Python Libraries

- Use Pip: https://pypi.org/project/pip/
- For instances:

```
pip install matplotlib
pip install numpy
pip install scipy
pip freeze > requirements.txt
```

- To reinstall:

```
pip install --no-cache-dir -r ./requirements.txt
```

YORK UNIVERSITÉ UNIVERSITY

Conclusions

Conclusions

- Install Docker on your computer
- Try out Docker and the Getting Started
- Try the Examples
- Learn Python 3

YORK UNIVERSITÉ UNIVERSITY

Questions or Issues?

- Post questions and issues to https://piazza.com/class/jlo569j7clw246
- Anyone having issues installing or using Docker on their computer should submit their questions to the Piazza
- The TA's will do our best to provide assistance and help resolve any issues.