Product Analytics

DIANE WOODBRIDGE, PH.D

Contents

Course Objectives

Entrepreneur class for designing and developing a data science web application.

The course will cover overall techniques and collaboration tools.

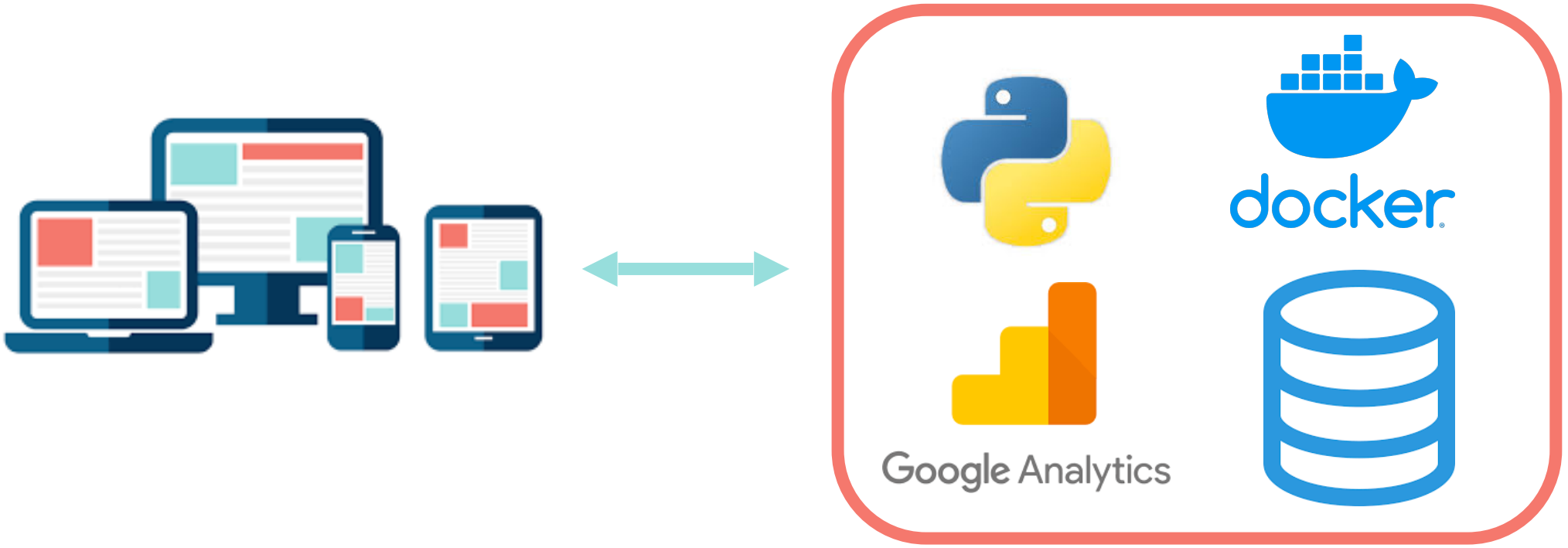
- Brush up on Python - Coding conventions, Documentation, Testing, etc.
- Learn basic front and back-end development.
- Apply web analytics tools for tracking website traffic and user activities.
- Design and develop a web application and deploy on AWS.

Students are encouraged to apply technical techniques and business strategies that they learned from previous and current courses.

This class also requires to present a business plan and progress regularly.

- This should meet the standards from Schaffer's class.

Course Objectives



Tentative Course Schedule

May 30 - Environment and Tool Setup for Collaboration

June 3 - Docker Container, Deployment (Elastic Beanstalk) and Backened

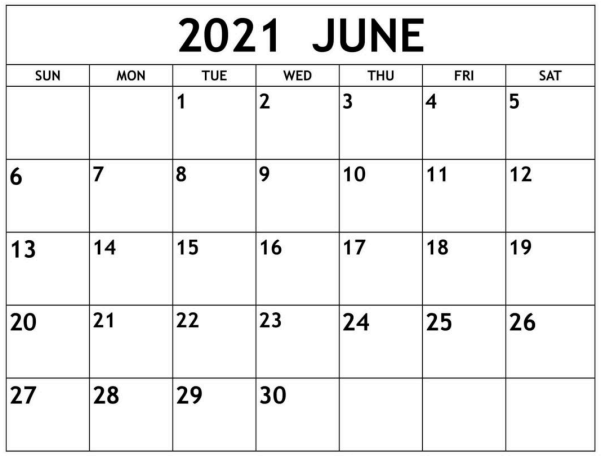
June 10 - Backened

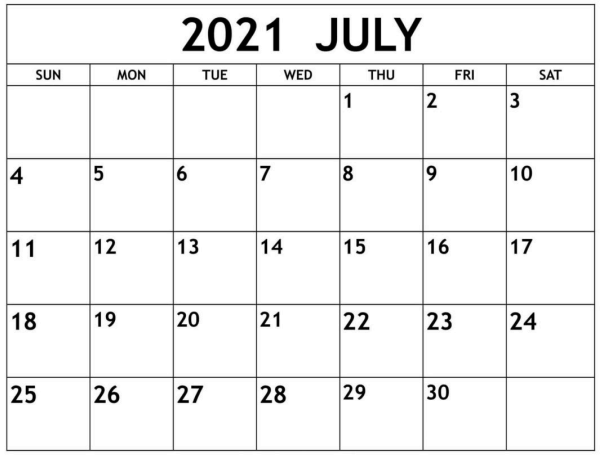
June 17 - Backened + Frontend Intro

June 22 - Frontend

June 24 - Web Analytics

June 29 - VC Presentation





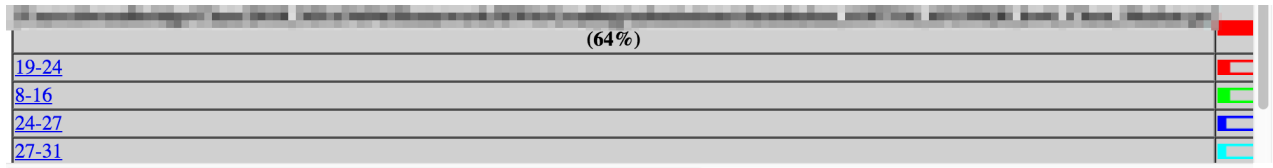
Grading

- Please see the Syllabus.
- **For the recorded lecture, you need to watch it before the discussion session.**
- **For the discussion session, attending the entire session is required.**
  - **No cellphones, No social media, No Slack, No naps!**

Grading

Please no plagiarism! – Zero tolerant.

Changed lines and tweaked logics



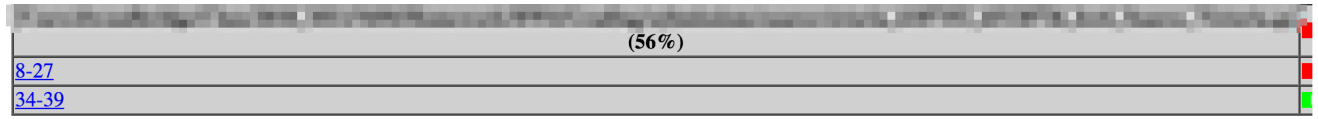

```
data = sc.textFile(input_file1)\
         .map(lambda x: x.split(","))\
         .filter(lambda x: len(x) == 5)

sensor_type = sc.textFile(input_file2)\
                .map(lambda x: x.split(","))\
                .map(lambda x: (int(x[0]), x[1]))

sensor_readings_types = data.groupBy(lambda x: int(x[1]))

sensor_readings = data.map(lambda x: (x[1]+","+x[0],
                        [float(x[2]), float(x[3]), float(x[4])]))\


                    .mapValues(lambda x: (1, x))\
                    .reduceByKey(lambda x, y: (x[0]+y[0],
                                               [x[1][0]+y[1][0],
                                                x[1][1]+y[1][1],
                                                x[1][2]+y[1][2]]))\
                    .mapValues(lambda x: (round(x[1][0]/x[0], 4),

                                          round(x[1][1]/x[0], 4),
                                          round(x[1][2]/x[0], 4)))\
                    .map(lambda x: (int(x[0].split(",")[0]),

                                    [float(x[0].split(",")[1]), x[1]]))

f = open(output_file, 'w')
for item in sensor_readings_types.leftOuterJoin(sensor_type)\


                            .sortByKey().collect():
    f.write(str(item[0])+" : "+str(item[1][1])+"\n")
    for prs in sensor_readings.groupByKey().collect():
```

```
data = sc.textFile(input_file1)\
         .map(lambda x : x.split(","))\
         .filter(lambda x : len(x) == 5)

sensor_type = sc.textFile(input_file2)\
                .map(lambda x : x.split(","))\
                .map(lambda x : (int(x[0]), x[1]))

sensor_type_in_readings = data.keyBy(lambda x : int(x[1]))\
                              .groupByKey()

#Transform data : timestamp, sensortype, x axis, y axis, z axis ==> sensortype:timestamp,

sensor_readings = data.map(lambda x : ((x[1]+ ":" + x[0]), [float(x[2]), float(x[3]), flo

#Calculate mean if there are multiple data with the same timestamp for each sensor.

preprocessed_sensor_readings = sensor_readings.mapValues(lambda x: (1,x))\
                                              .reduceByKey(lambda x, y : (x[0]+y[0],[x[1]
                                              .mapValues(lambda x : [round(x[1][0]/x[0],

                                              .map(lambda x : (int(x[0].split(":")[0]),[f


#Print sensor information and first and last n_element
f = open(output_file,"w")
for type in sensor_type_in_readings.leftOuterJoin(sensor_type).sortByKey().collect():
    #print readings
    for preprocessed_reading in preprocessed_sensor_readings.groupByKey().collect():

        if(type[0] == preprocessed_reading[0]):
            f.write(str(type[0]) + " : " + str(type[1][1]) + "\n")
            for i in sorted(list(preprocessed_reading[1]))[:n_element]:
```

Grading

Please no plagiarism! – Zero tolerant.

Changed lines and variable names




```
timeseries = sc.textFile(input_file1)
sensors = sc.textFile(input_file2)

timeseries = timeseries.filter(lambda x: len(x) > 0)  # remove empty lines
timeseries = timeseries.map(lambda x: x.split(','))
timeseries = timeseries.map(lambda x: [float(num) for num in x])
# ((timestamp, sensorID),  [x,y,z])
timeseries = timeseries.map(lambda x: ((x[0], x[1]), x[2:5]))

countbyTimeStamp = timeseries.countByKey()

timeseries = timeseries.reduceByKey(
    lambda x, y: [num_x + num_y for num_x, num_y in zip(x, y)])
timeseries = timeseries.map(
    lambda x: (x[0], [round(num/countbyTimeStamp[x[0]], 4) for num in x[1]]))

sensors = sensors.map(lambda x: x.split(','))
sensors = sensors.map(lambda x: (int(x[0]), x[1]))

timeseries = timeseries.map(
    lambda x: (x[0][1], [x[0][0], x[1]]))  # (sensorID,[timestamp, [x,y,z]])

sensor_name_join = timeseries.leftOuterJoin(sensors)
# ((sensor_ID, Name),[timestamp,[x,y,z]])
sensor_name_join = sensor_name_join.map(
    lambda x: ((x[0], x[1][1]), x[1][0]))

sensor_name_join = sensor_name_join.groupByKey()
sensor_name_join = sensor_name_join.mapValues(
    lambda x: sorted(x, key=lambda y: y[0]))

with open(output_file, 'w') as f:
    for sensorID in sensor_name_join.sortByKey().collect():
```

```
from user_definition import *
# DO NOT ADD OTHER LIBRARIES/PACKAGES!


conf = SparkConf().setAppName(app_name)
sc = SparkContext(conf=conf).getOrCreate()

ts = sc.textFile(input_file1)
sensor = sc.textFile(input_file2)

ts = ts.filter(lambda x: len(x) > 1)
ts = ts.map(lambda x: x.split(","))
ts = ts.map(lambda x: [float(num) for num in x])
ts = ts.map(lambda x: ((x[0], x[1]), x[2:5]))
count = ts.countByKey()
ts = ts.reduceByKey(lambda x, y: [(x+y) for x, y in zip(x, y)])
ts = ts.map(lambda x: (x[0], [round(num/count[x[0]], 4) for num in x[1]]))
sensor = sensor.map(lambda x: x.split(","))
sensor_num = sensor.map(lambda x: (int(x[0]), x[1]))
ts_group_sensor = ts.map(lambda x: [x[0][1], (x[0][0], x[1])]).groupByKey()
ts_group_sensor = ts_group_sensor.map(lambda x: [x[0], list(x[1])])
full = ts_group_sensor.leftOuterJoin(sensor_num)

full = full.map(lambda x: [(x[0], x[1][1]), x[1][0]]).sortByKey()
full = full.mapValues(lambda x: sorted(x, key=lambda y: y[0]))

with open(output_file, "w") as f:
    for sensor in full.collect():
        f.write(str(int(sensor[0][0])) + " : " + str(sensor[0][1]) + '\n')
        for value in sensor[1][:(n_element)]:
            f.write(str(list(value)) + '\n')
        f.write('...' + '\n')
        for value in sensor[1][-(n_element):]:
            f.write(str(list(value)) + '\n')
```

Collaboration

This class requires collaborations.

- Things to consider..
  - Code Standard
  - Environment Setup
  - Documentation
  - Unit Test
  - And many more!

Reference Materials

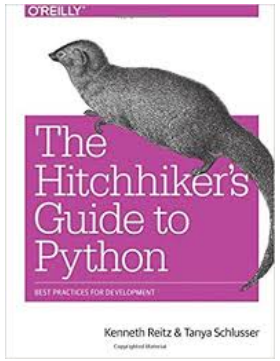
Kenneth Reitz. *The Hitchhiker's Guide to Python*. http://docs.python-guide.org/en/latest/

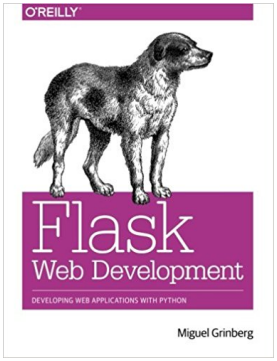
Grinberg, Miguel. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

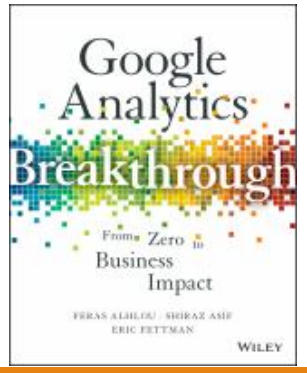
Google Analytics Academy (Beginner, Advanced). https://analytics.google.com/analytics/academy/

Alhlou, Feras, Shiraz Asif, and Eric Fettman. *Google Analytics Breakthrough: From Zero to Business Impact*. John Wiley & Sons, 2016.

Docker Documentation. https://docs.docker.com/







Office Hour

- Tuesday 9-10 am
- Please reserve a slot via Google Calendar

Others…

Example Data

- https://github.com/dianewoodbridge/2021-msds603-example.git

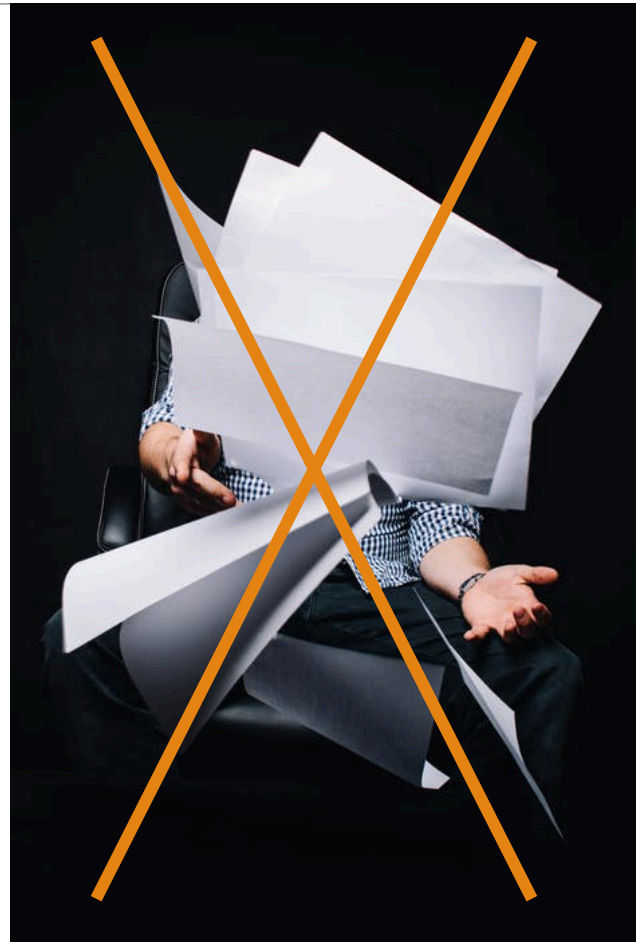
Poll

- https://pollev.com/msds

Project Requirements

Project presentation

- Need to upload a recorded presentation on Canvas.
- You need to watch and peer-review at least one Group Presentation session (All the recorded presentations).
  - Watch either on June 10th or June 17th
  - Everyone needs to watch on June 24th
- You need to be registered on Canvas “Group Presentation Peer Review Date” Group and must review that day.

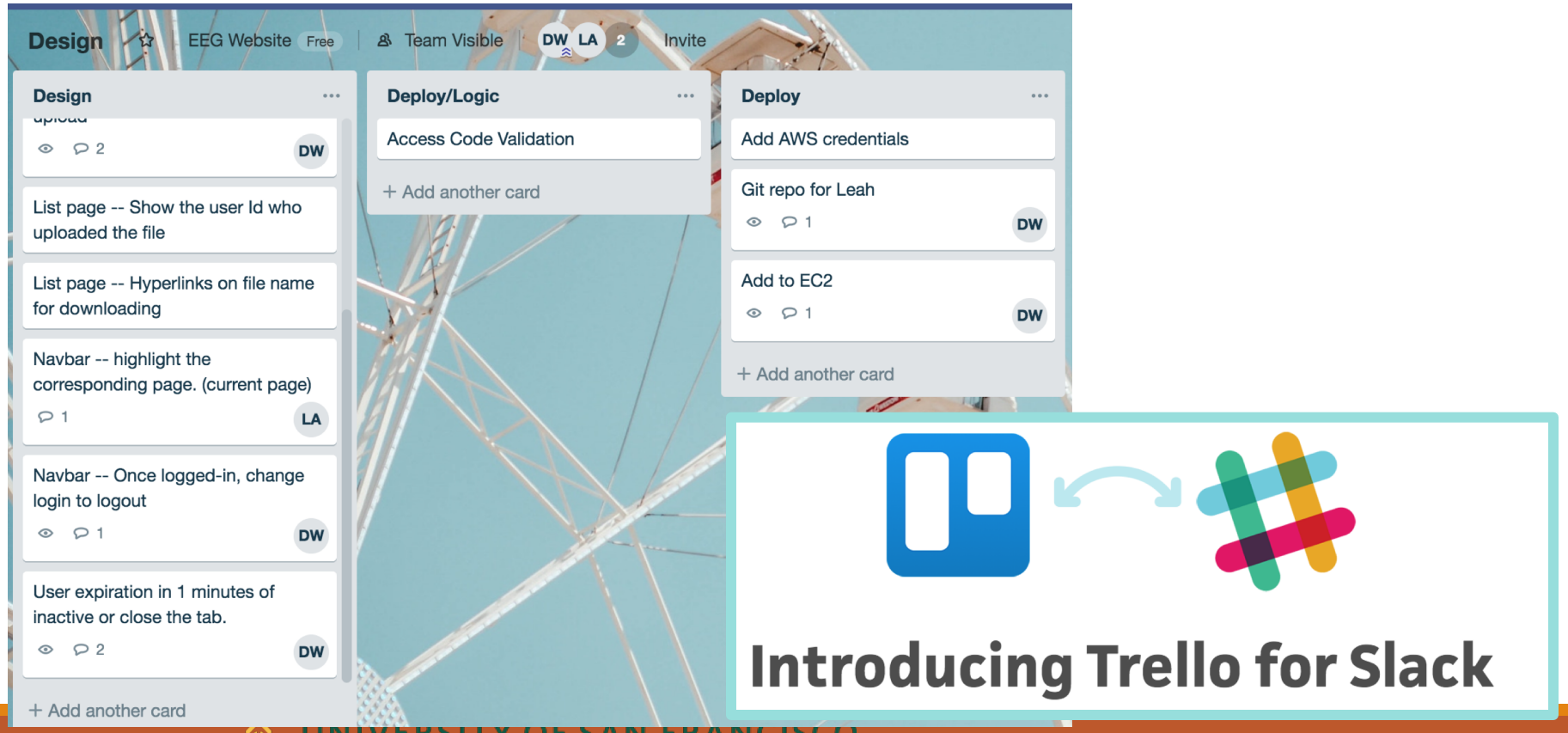


Any questions?

Contents

Trello

Project management tool



https://trello.com

Contents

Virtual Environment

```
import pyspark
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-2-49d7c4e178f8> in <module>
----> 1 import pyspark

ModuleNotFoundError: No module named 'pyspark'
```

Virtual Environment

A project team member might work on several projects requiring different versions of Python or other packages.

Environment

- Keep dependencies required by different project in separate places.
- Easily switch to a environment that requires different dependencies.

Virtual Environment using pip

*We are using pip based on the Docker/Elastic Beanstalk requirement.

Create an environment

- `$ python -m venv environment_name`
- This creates environment_name folder **in your working directory.**

```
ML-ITS-901885:driving_time_example dwoodbridge$ python -m venv ProductAnalytics
```

Activate the environment

- `$ source environment_name/bin/activate`
- You can install packages using pip, etc.

Dectivate the environment

- `$ deactivate`

Virtual Environment using pip

Export your environment

- `pip freeze > requirements.txt`

Creating an environment using requirements.txt file.

- `pip install -r requirements.txt`

Example Scenario

When should we go for a walk?

When it is sunny or cloudy and it is warm.

Example Scenario

Manual way

1. Check weather.com and see.
2. Call it every minute and write it down to a file.
3. See the trends and decide “when”.

Example Scenario

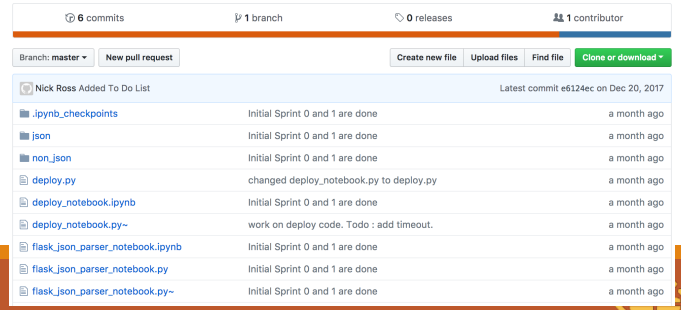
Automated way

1. Write a code to call OpenWeatherMap API to get duration.
2. Call it every minute on a server.
   - For having a bigger storage and setting a schedule.

Example Scenario

Automated way

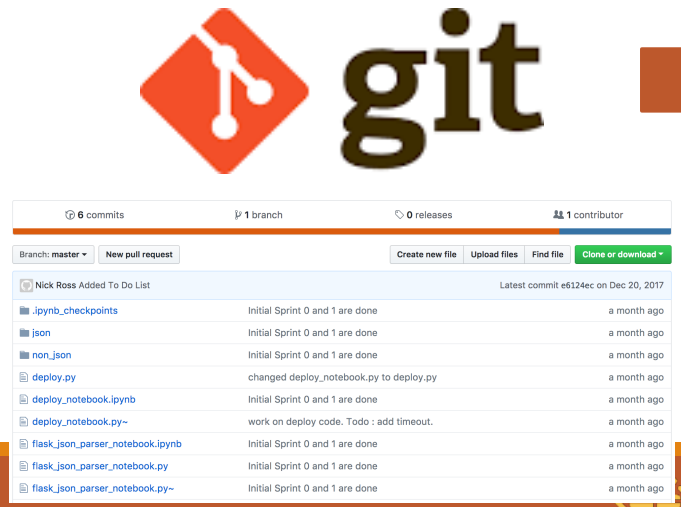
1. Write a code to call OpenWeatherMap API to get duration.
2. Call it every minute on a server.
   - For having a bigger storage and setting a schedule.



Example Scenario

Automated way

1. **Write a code to call OpenWeatherMap API to get duration.**
2. Call it every minute on a server.
   - For having a bigger storage and setting a schedule.



Example Scenario

Automated way

1. **Write a code to call OpenWeatherMap API to get duration.**
   - OpenWeatherMap API
     - Provide current weather data for any location (Over 200K cities).
     - You can get your API from https://openweathermap.org/api

Example Scenario

Automated way

1. **Write a code to call OpenWeatherMap API to get duration.**

```
@application.route('/calculate', methods=['GET', 'POST'])
def calculate():
    """ Read Google Distance API """
    api_key = os.environ['API_KEY']
    url = f"http://api.openweathermap.org/data/2.5/weather?appid={api_key}&zip={zip},us&units=imperial"
    response = requests.get(url)
    x = response.json()
    main = x['weather'][0]['main']
    temp = x['main']['temp']

    if (main in ['Clear', 'Clouds'] and 60 < temp < 85):
        msg = "Go for a walk"
    else:
        msg ="Stay home"
    prev_reading = read_s3_obj(bucket_name, output_file)
    print(prev_reading)

    body = "{}\t{}\t{}\t{}\t{}\n".format(msg,
                                        datetime.datetime.now(),
                                        main,
                                        temp,
                                        prev_reading)
    boto3.resource("s3").Bucket(bucket_name).put_object(Key=output_file, Body=body, ACL='public-read-write')

    return redirect("/index")
```

Example Scenario

Automated way

1. **Write a code to call OpenWeatherMap API to get duration.**

```python
@application.route('/', methods=['GET', 'POST'])
@application.route('/index', methods=['GET', 'POST'])
def index():
    """ index page -- shown on the beginning """
    body = read_s3_obj(bucket_name, output_file)

    return render_template('index.html', output=body)
```

index.html

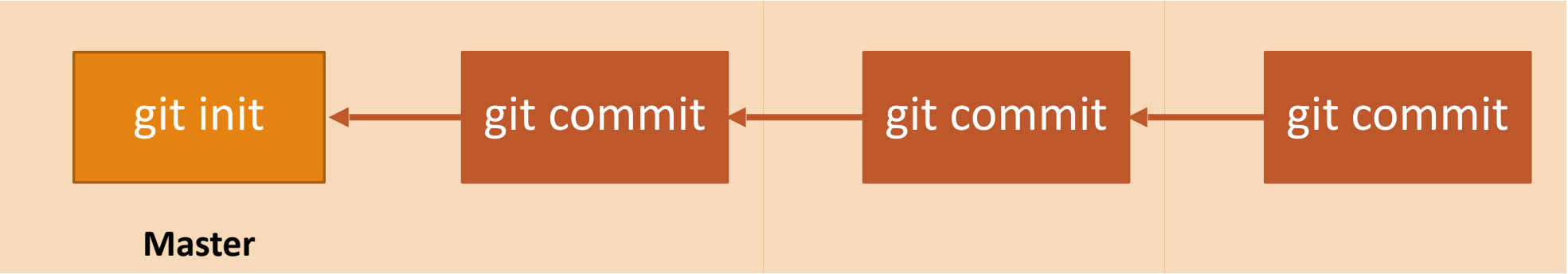
```
<html>
    <h1> OUTPUT </h1>
    {% if output|length > 0 %}
        {% for line in output.split("\n") %}
            {% for item in line.split("\t") %}
                {{ item }} </br>
            {% endfor %}
            </br>
        {% endfor %}
    {% else %}
        <p> Empty </p>
    {% endif %}
</html>
```

Contents

Git Branch

Diverge from the main line of development and continue to do work without messing with that main line.

- The default branch name : master.
- As you start making commits, you're given a master branch that points to the last commit you made.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Git Branch

Diverge from the main line of development and continue to do work without messing with that main line.

- To create a new branch, "`git branch branch_name`"
- To switch to a branch, "`git checkout branch_name`"
- The local your currently on is going to be called as "HEAD"

https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Git Branch

Diverge from the main line of development and continue to do work without messing with that main line.

- To visually check the changes in branches,
  - `git log --graph`
  - Try git GUI tools including Source Tree

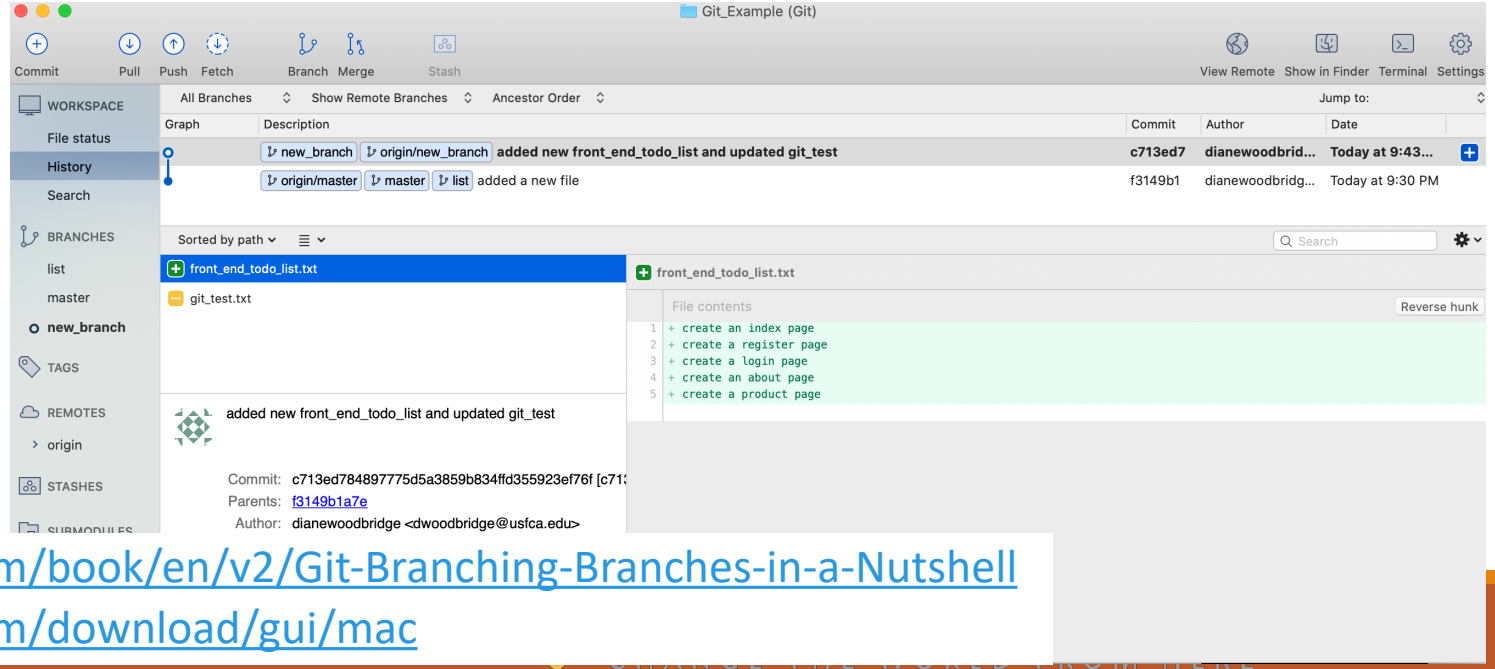
```
(ProductAnalytics_Env) (base) ML-ITS-901885:Git_Example dwoodbridge$ git log --graph
* commit c713ed784897775d5a3859b834ffd355923ef76f (HEAD -> new_branch, origin/new_branch)
| Author: dianewoodbridge <dwoodbridge@usfca.edu>
| Date:   Tue Mar 24 21:43:12 2020 -0700
|
|     added new front_end_todo_list and updated git_test
|
* commit f3149b1a7efe6f97ab8e1bec167bbccdab140870 (origin/master, master, list)
  Author: dianewoodbridge <dwoodbridge@usfca.edu>
  Date:   Tue Mar 24 21:30:04 2020 -0700

      added a new file
```

https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell
https://git-scm.com/download/gui/mac

Git Branch

Diverge from the main line of development and continue to do work without messing with that main line.

- To visually check the changes in branches,
  - `git log --graph`
  - Try git GUI tools including Source Tree



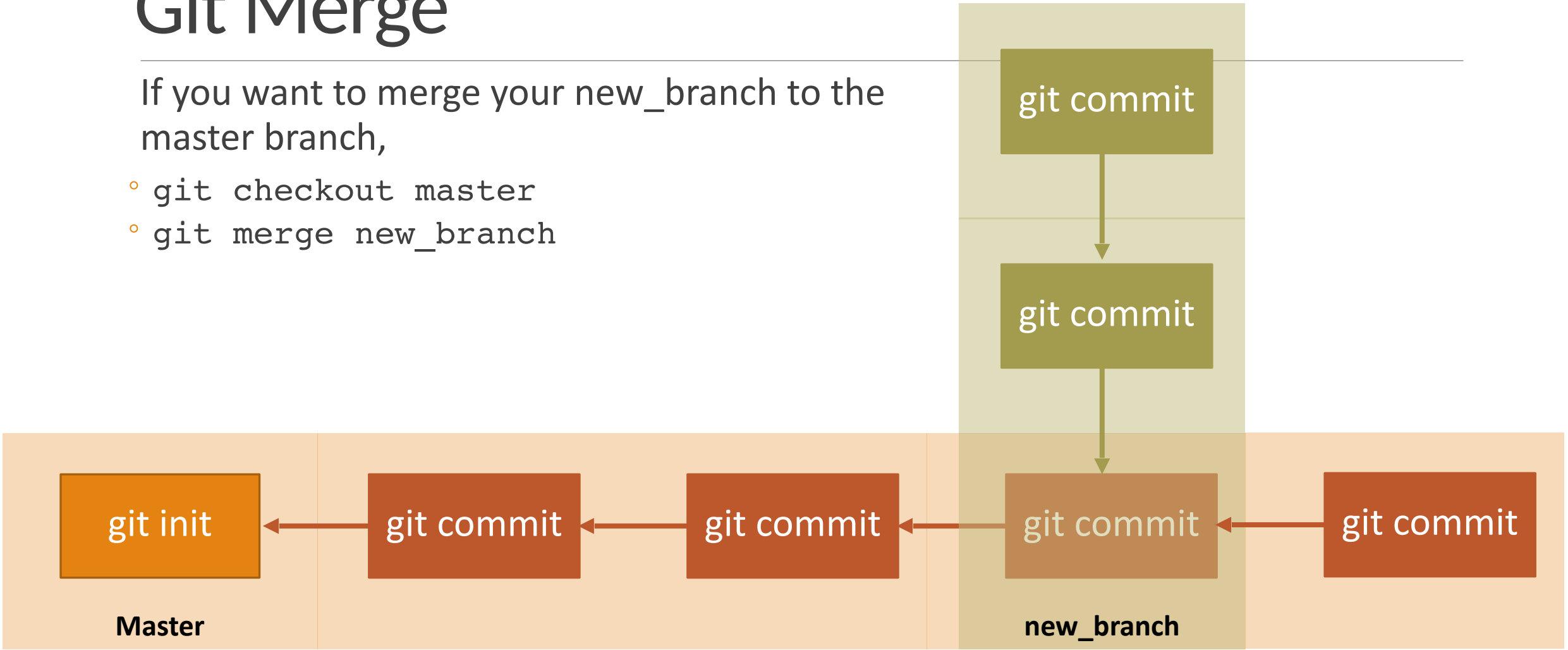
https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

https://git-scm.com/download/gui/mac

Git Merge

If you want to merge your new_branch to the master branch,

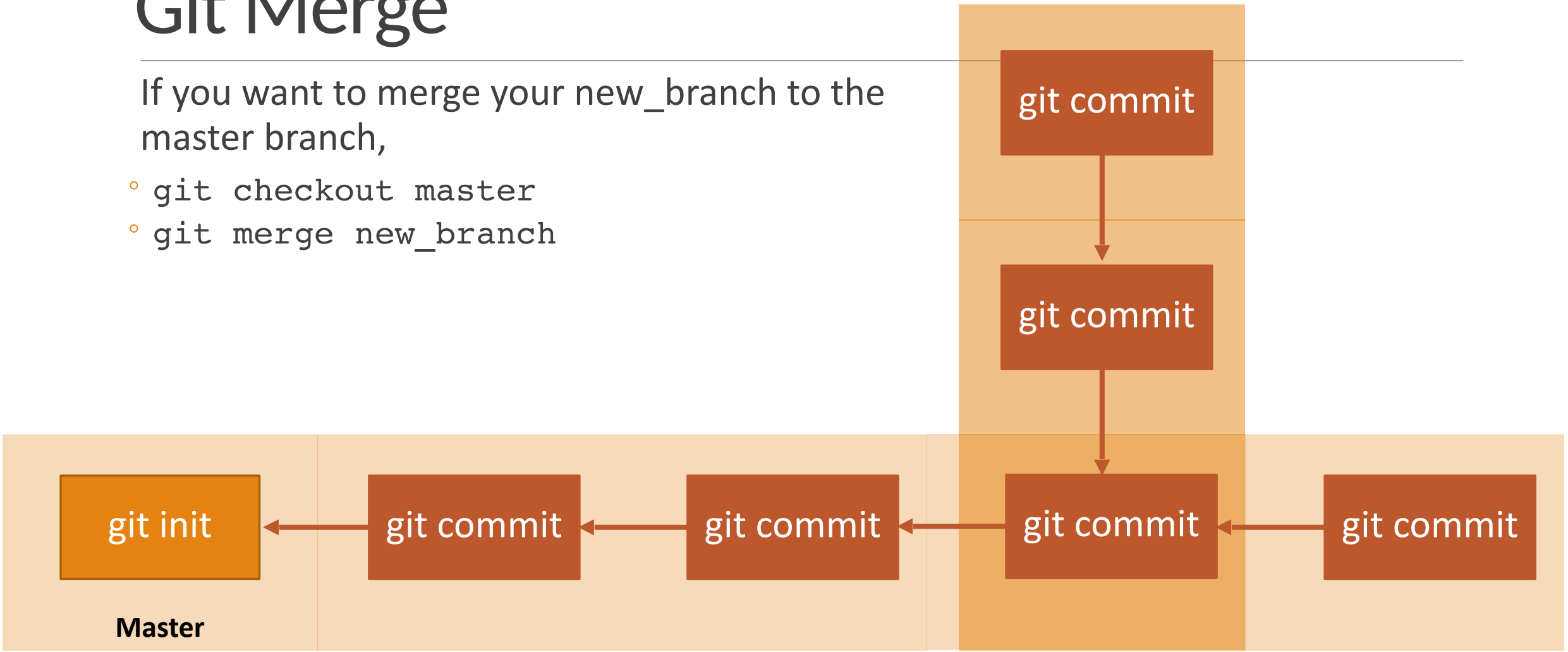
- `git checkout master`
- `git merge new_branch`



Git Merge

If you want to merge your new_branch to the master branch,

- `git checkout master`
- `git merge new_branch`



Contents

Documentation

README

- General information written in plain text to users and programmers of a project.
- Contents
  - A few lines explain the purpose of the project or library.
  - Contributors (Authors).
  - The URL of the main source.
  - Change logs.
  - Copyrights.

https://en.wikipedia.org/wiki/README

Documentation

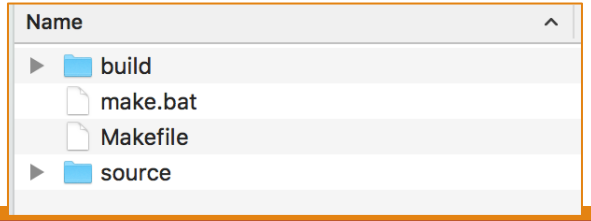
Project Publication

- Your project publication might include more detailed information.
  - Introduction and overview of the project
  - Contributors (Authors) and their roles.
  - Tutorial - detailed step-by-step instructions to setup and run the code.
  - API reference – API information (written in docstring) and actual code.
  - Developer guidance – code conventions and design strategy for other contributors.

Documentation

Sphinx

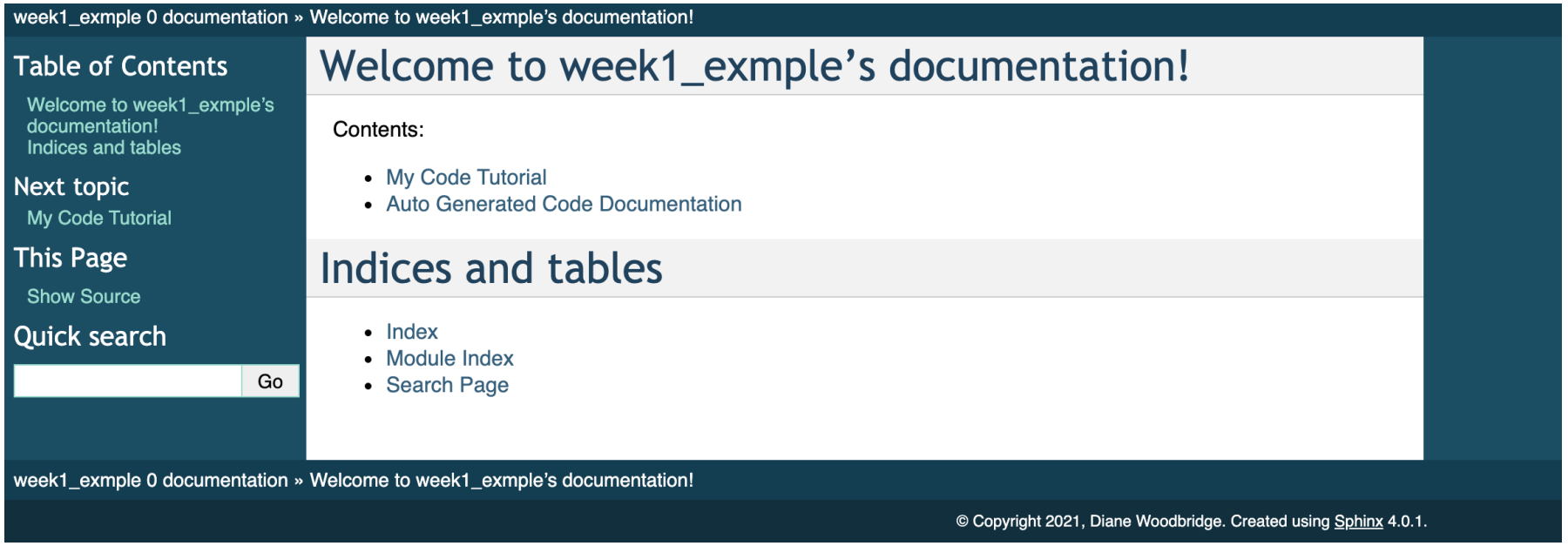
- Most popular Python documentation generation tool.
- Generate html, LaTex, txt, etc. Sphinx takes docstrings in your code and texts in reStructuredText (.rst) to create a code document.
- Installation
  - `$ pip install sphinx`
- Set up a document source directory
  - `$ sphinx-quickstart`
    - Creates a document source directory and conf.py with configuration options that you choose.



Documentation

Sphinx

- Generate and Publish automated document on Git.
  - Follow this tutorial.

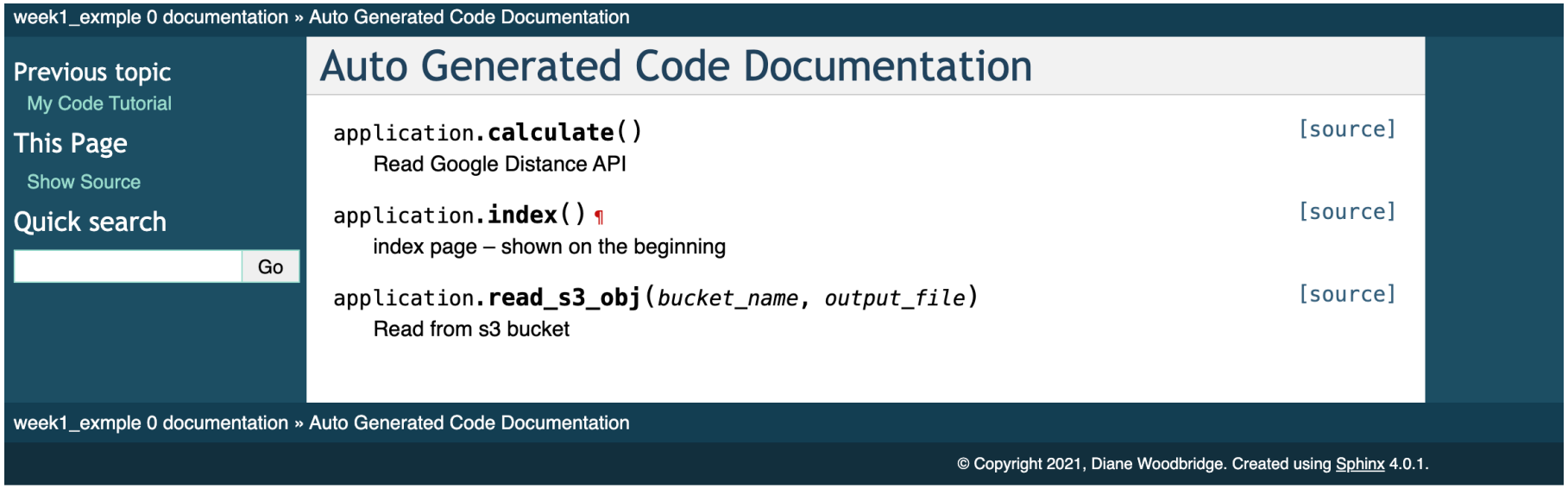


https://docs.google.com/document/u/2/d/1bEZl9Mm9FmuIvyEOPxknAmDi2ICml6wlbYcQKunaGdM/edit?usp=sharing

Documentation

Sphinx

- Generate and Publish automated document on Git.
  - Follow this tutorial.



https://docs.google.com/document/u/2/d/1bEZl9Mm9FmuIvyEOPxknAmDi2ICml6wlbYcQKunaGdM/edit?usp=sharing

Documentation

Sphinx

- Generate and Publish automated document on Git.
  - Follow this tutorial.

```
@application.route('/', methods=['GET', 'POST'])                    [docs]
@application.route('/index', methods=['GET', 'POST'])
def index():
    """ index page -- shown on the beginning """
    body = read_s3_obj(bucket_name, output_file)

    return render_template('index.html', output=body)



@application.route('/calculate', methods=['GET', 'POST'])           [docs]
def calculate():
    """ Read Google Distance API """
    api_key = os.environ['API_KEY']
    url = f"http://api.openweathermap.org/data/2.5/weather?appid={api_key}" \
          f"&zip={zip},us&units=imperial"
    response = requests.get(url)
    x = response.json()
    main = x['weather'][0]['main']
    temp = x['main']['temp']

    if (main in ['Clear', 'Clouds'] and 60 < temp < 85):
        msg = "Go for a walk"
    else:
        msg = "Stay home"
    prev_reading = read_s3_obj(bucket_name, output_file)
```

https://docs.google.com/document/u/2/d/1bEZl9Mm9FmuIvyEOPxknAmDi2ICml6wlbYcQKunaGdM/edit?usp=sharing

Contents

Unit Test

Automated tests that the section (class, module, function, etc.) of your code behaves as intended.

Good practices

- Use descriptive names for testing functions.
- A testing unit should focus on one tiny bit of functionality and prove it correct.
- Always run the full test suite before a coding session, and run it again after.

https://docs.python-guide.org/writing/tests/

Unit Test

pytest

- Python-based unit test framework.
  - Alternatives : robot, unittest, doctest, nose2, etc.
- Install
  - `pip install pytest`

```
(ProductAnalytics_Env) (base) ML-ITS-901885:Week3 dwoodbridge$ pip install pytest
Collecting pytest
  Downloading pytest-5.4.1-py3-none-any.whl (246 kB)
     |████████████████████████████████| 246 kB 534 kB/s
Collecting py>=1.5.0
  Downloading py-1.8.1-py2.py3-none-any.whl (83 kB)
     |████████████████████████████████| 83 kB 1.2 MB/s
Requirement already satisfied: wcwidth in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-
packages (from pytest) (0.1.8)
Requirement already satisfied: attrs>=17.4.0 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7
/site-
Collecting pluggy<1.0,>=0.12
  Downloading pluggy-0.13.1-py2.py3-none-any.whl (18 kB)
Collecting more-itertools>=4.0.0
```

https://docs.pytest.org/en/latest/

Unit Test

pytest

You can use the assert statement to verify test expectations.

- The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

```
from application import *

def test_api_call():
    assert retreive_web_data()[0] != None
    assert retreive_web_data()[1] != None
    assert len(read_s3_obj(bucket_name, output_file)) > 1
```

https://docs.pytest.org/en/latest/

Unit Test

Run pytest

To run multiple test files, pytest will run all files of the form test_*.py or *_test.py in the current directory and its subdirectories.

- If you don't have "test" in your .py file, you can directly call pytest your_code.py

```
(msds603) ML-ITS-901885:weather_checking dwoodbridge$ pytest
============================================================================= test session starts ==============
platform darwin -- Python 3.9.0, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /Users/dwoodbridge/Class/2021_MSDS603/2021-msds603-example/Week1/weather_checking
collected 1 item

test_application.py .

============================================================================== 1 passed in 1.35s ===============
```

Run pytest

To run multiple test files, pytest will run all files of the form test_*.py or *_test.py in the current directory and its subdirectories.

- If you don't have "test" in your .py file, you can directly call pytest your_code.py

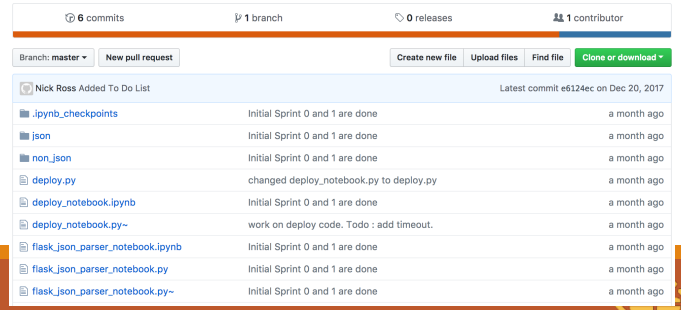
```
(msds603) ML-ITS-901885:weather_checking dwoodbridge$ pytest
============================================================================= test session starts ==============================================================================
platform darwin -- Python 3.9.0, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /Users/dwoodbridge/Class/2021_MSDS603/2021-msds603-example/Week1/weather_checking
collected 1 item

test_application.py F

================================================================================== FAILURES ====================================================================================
_____________________________________________________________________________ test_api_call _____________________________________________________________________________

    def test_api_call():
        assert retreive_web_data()[0] is not None
        assert retreive_web_data()[1] is not None
>       assert len(read_s3_obj(bucket_name, output_file)) > 1

test_application.py:7:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
application.py:24: in read_s3_obj
    body = obj.get()['Body'].read().decode('utf-8')
../../../example/Week1/weather_checking/msds603/lib/python3.9/site-packages/boto3/resources/factory.py:520: in do_action
    response = action(self, *args, **kwargs)
../../../example/Week1/weather_checking/msds603/lib/python3.9/site-packages/boto3/resources/action.py:83: in __call__
    response = getattr(parent.meta.client, operation_name)(*args, **params)
../../../example/Week1/weather_checking/msds603/lib/python3.9/site-packages/botocore/client.py:357: in _api_call
    return self._make_api_call(operation_name, kwargs)
```

Contents

Example Scenario

Automated way

1. Write a code to call OpenWeatherMap API to get duration.
2. **Call it every minute on a server.**
   - For having a bigger storage and setting a schedule.



Example Scenario

Automated way

2. **Call it every minute on a server.**
   - Execute `/calculate` every minute

Executing a Job at Designated Times

Crontab

- A system daemon used to execute desired tasks in the background at designated times.
- Especially useful for pulling data which doesn't send data to your application automatically.

https://help.ubuntu.com/community/CronHowto

Executing a Job at Designated Times

Crontab

- Format
  - 1st : Minute (0-59)
  - 2nd: Hour (0-23)
  - 3rd : Day (1-31)
  - 4th : Month(1-12)
  - 5th : Day of week (0-7. 0/7-Sun. 1-Mon, 2-Tue..)

```
* * * * * /command/to/be/executed
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

Crontab generator - https://crontab-generator.org/

Crontab on a Local Machine

`$ crontab filename`

- Load the crontab data from the specified file.
- The file should include the schedule and commands.

`$ crontab –e`

- Edit the current crontab.
- You need to add the schedule and commands.

`$ crontab –l`

- Display the current crontab.

Crontab on a Local Machine

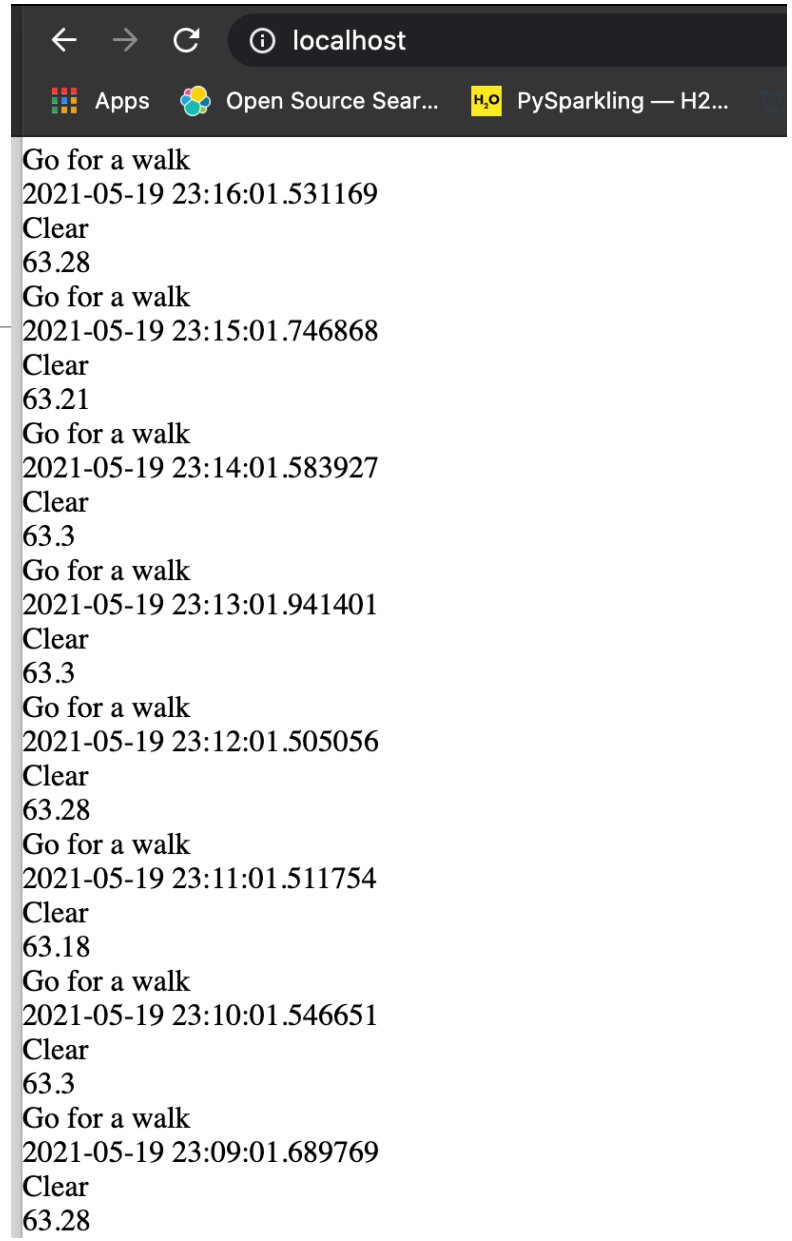
```
$crontab cronjobs
$crontab -l
```

```
(msds603) ML-ITS-901885:cron dwoodbridge$ crontab cronjobs
(msds603) ML-ITS-901885:cron dwoodbridge$ crontab -l
* * * * * curl localhost/calculate
```

Example Scenario

Automated way

1. Write a code to call OpenWeatherMap API to get duration.
2. Call it every minute on a server.



Contents

Course Overview

Trello

Virtual Environment

Git Branch

Auto Documentation (Sphinx)

Unit Test

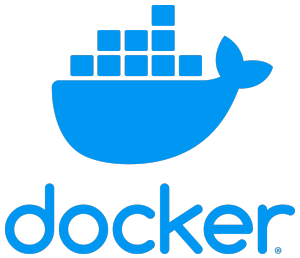
Crontab

Docker

Elastic Beanstalk

Create a script

Docker



Docker

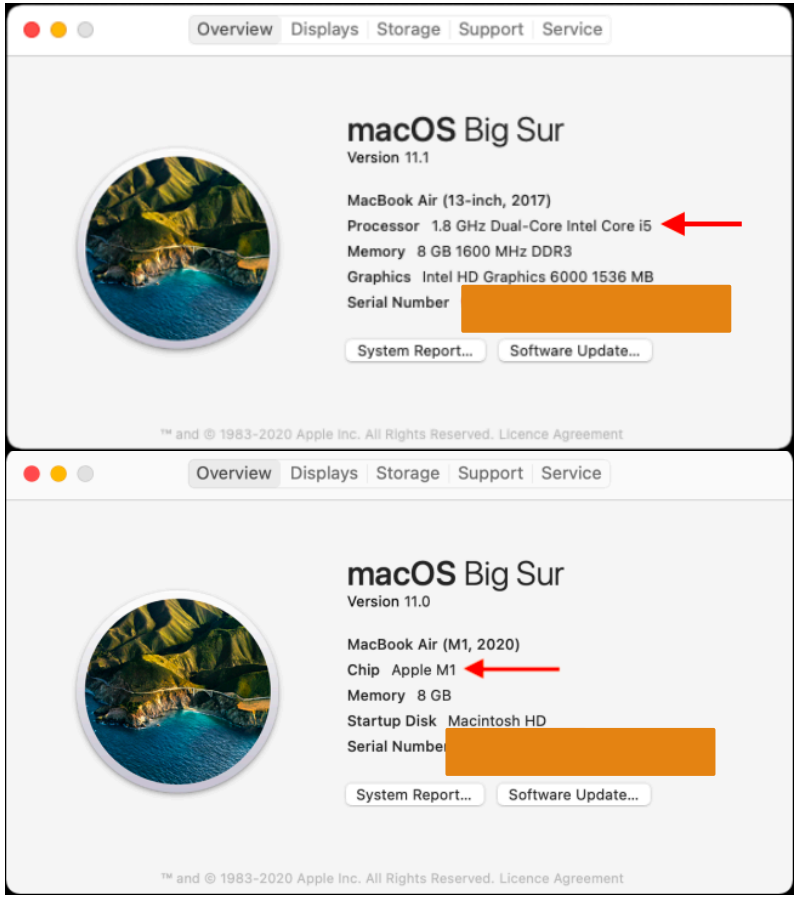
- Encapsulate the process of creating a distributable artifact for applications, deploying at scale into any environment, and streamlining the workflow.
  - Bundles OS file system, application software and all the requirement in an image format.
  - Uses the artifacts to test and deliver the exact same artifact to all systems/environments.

Matthias, K., and Kane S. Docker. "Up & Running: Shipping Reliable Containers in Production." (2018).

Docker

Install Docker Desktop on Mac

- Mac with Intel chip
- Mac with Apple chip



https://docs.docker.com/docker-for-mac/install/

Docker

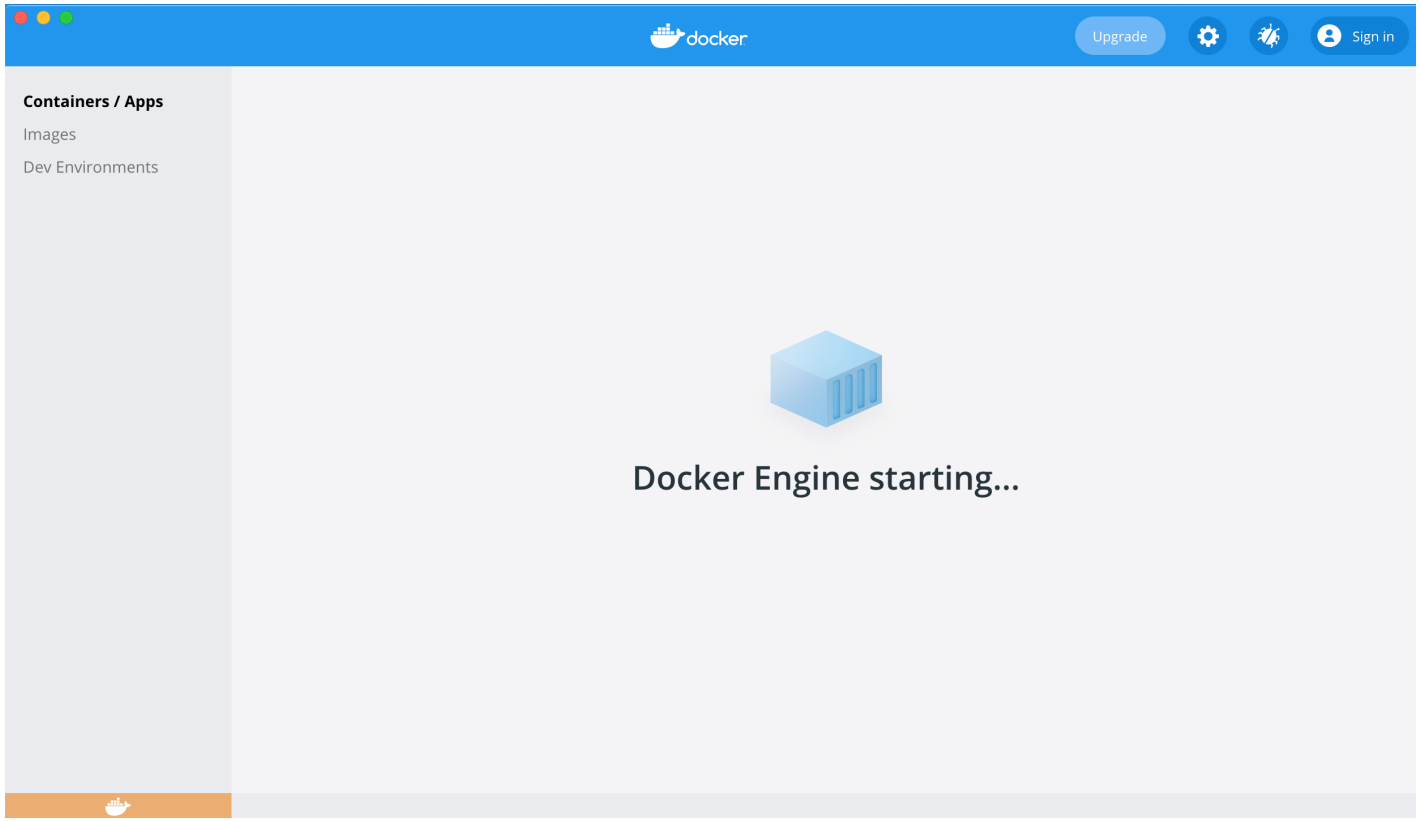
Install Docker Desktop on Mac

- Mac with Intel chip
- Mac with Apple chip

https://docs.docker.com/docker-for-mac/install/

Docker

Install Docker Desktop on Mac



https://docs.docker.com/docker-for-mac/install/

Docker

Image

- A read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
- To build your own image, you create a **Dockerfile** with a simple syntax for defining the steps needed to create the image and run it.
  - Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.

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

Docker

```
FROM python:3.7-alpine
RUN apk add git

RUN git clone https://github.com/dianewoodbridge/2021-msds603-example /2021-msds603-example
WORKDIR /2021-msds603-example/Week1/weather_checking

RUN pip install --upgrade pip
RUN pip install -r requirements.txt

EXPOSE 80
ENTRYPOINT [ "python", "application.py"]
```

A list of instructions in Dockerfile

FROM <image>	Initialize a new build stage and sets the Base Image.
RUN <command> or [“executable”, “param1”, ”param2”]	Execute any commands in a new layer on top of the current image
CMD [“executable","param1","param2"] or [“param1”,"param2"] or command param1 param2	The main purpose of a CMD is to provide defaults for an executing container. There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.
EXPOSE <port>	Informs the container listens on the specified network ports at runtime. To actually publish the port when running the container, use the `–p` flag on `docker run` to publish.
ENV <key>=<value> ...	Set the environment variable <key> to the value <value>.
ADD <src> <dest>	Copy new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>
COPY <src> <dest>	Copy new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.
WORKDIR /path/to/workdir	Set the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile
VOLUME ["/data"]	Create a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.

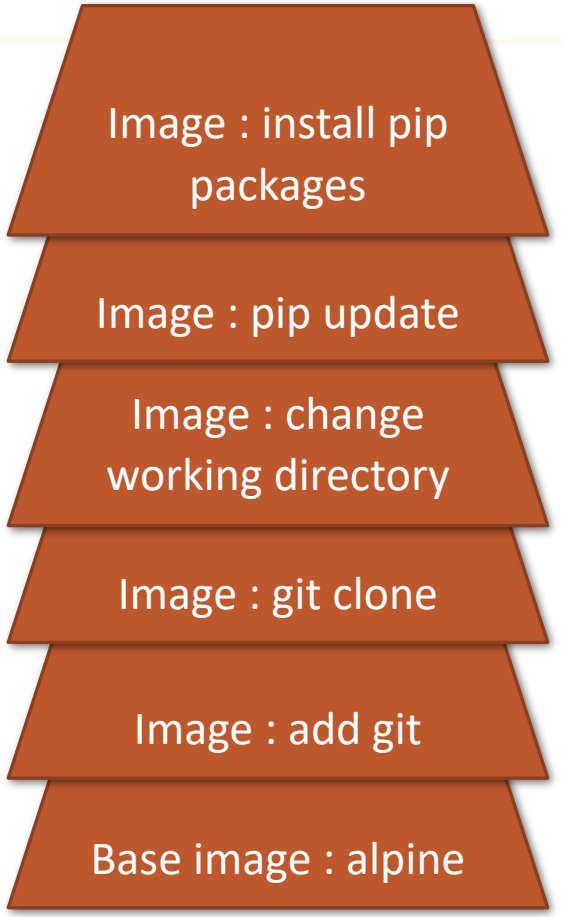
https://docs.docker.com/engine/reference/builder/

Docker

```
FROM python:3.7-alpine
RUN apk add git

RUN git clone https://github.com/dianewoodbridge/2021-msds603-example /2021-msds603-example
WORKDIR /2021-msds603-example/Week1/weather_checking

RUN pip install --upgrade pip
RUN pip install -r requirements.txt

EXPOSE 80
ENTRYPOINT [ "python", "application.py"]
```

Image

- A read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
- To build your own image, you create a **Dockerfile** with a simple syntax for defining the steps needed to create the image and run it.
  - Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.



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

Docker

Create an image

- `$ docker build [options] PATH`

```
(msds603) ML-ITS-901885:weather_chekcking dwoodbridge$ sudo docker build --tag weather-checking-app .
[+] Building 41.8s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 270B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/continuumio/miniconda3:latest                                            0.5s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 1.01kB                                                                                 0.0s
 => CACHED [stage-1 1/4] FROM docker.io/continuumio/miniconda3@sha256:7838d0ce65783b0d944c19d193e2e6232196bada9e5f3762dc7a9f07dc271179    0.0s
 => [stage-1 2/4] COPY . /app                                                                                       0.0s
 => [stage-1 3/4] WORKDIR /app                                                                                      0.0s
 => [stage-1 4/4] RUN pip install -r requirements.txt                                                              37.7s
 => exporting to image                                                                                              3.5s
 => => exporting layers                                                                                             3.5s
 => => writing image sha256:72fcaa30662233b3362c3ac37f16a894d49bbece427c9c465395578421742ce6                        0.0s
 => => naming to docker.io/library/weather-checking-app                                                             0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(msds603) ML-ITS-901885:weather_chekcking dwoodbridge$
```

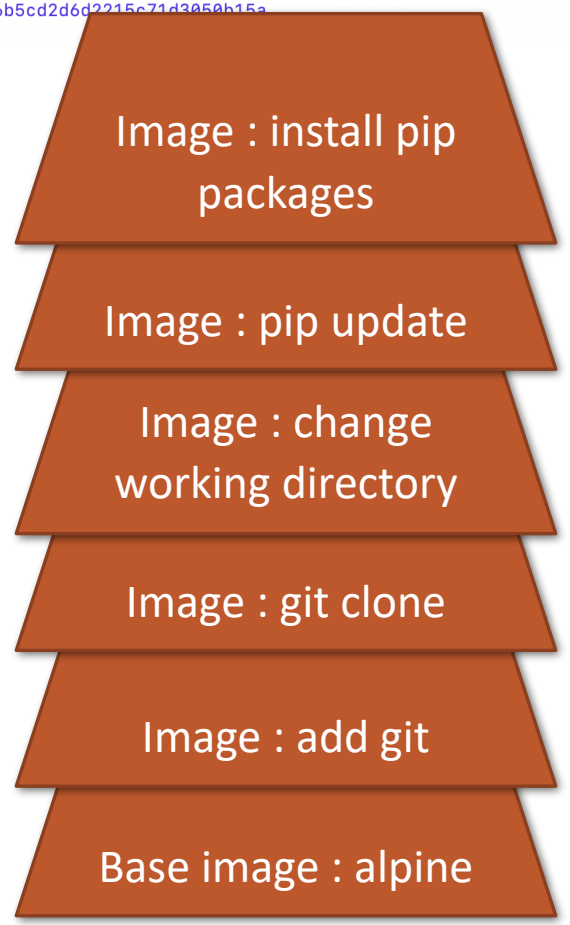
https://docs.docker.com/engine/reference/commandline/build/

Docker

```
(msds603) ML-ITS-901885:weather_checking dwoodbridge$ docker build . -t weather-app
[+] Building 1.7s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 354B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.7-alpine
 => [auth] library/python:pull token for registry-1.docker.io
 => [1/6] FROM docker.io/library/python:3.7-alpine@sha256:56cb9e22b1ab2264251c111144f22a2bc83d2404ca30bf4237178f49703ebbb8
 => CACHED [2/6] RUN apk add git
 => CACHED [3/6] RUN git clone https://github.com/dianewoodbridge/2021-msds603-example /2021-msds603-example
 => CACHED [4/6] WORKDIR /2021-msds603-example/Week1/weather_checking
 => CACHED [5/6] RUN pip install --upgrade pip
 => CACHED [6/6] RUN pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:89d717349780458a5df5c7115bb71b9fe1b88c26b5cd2d6d2215c71d3050b15a
 => => naming to docker.io/library/weather-app
```

Image

- A read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
- To build your own image, you create a **Dockerfile** with a simple syntax for defining the steps needed to create the image and run it.
  - Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.
  - Read-only

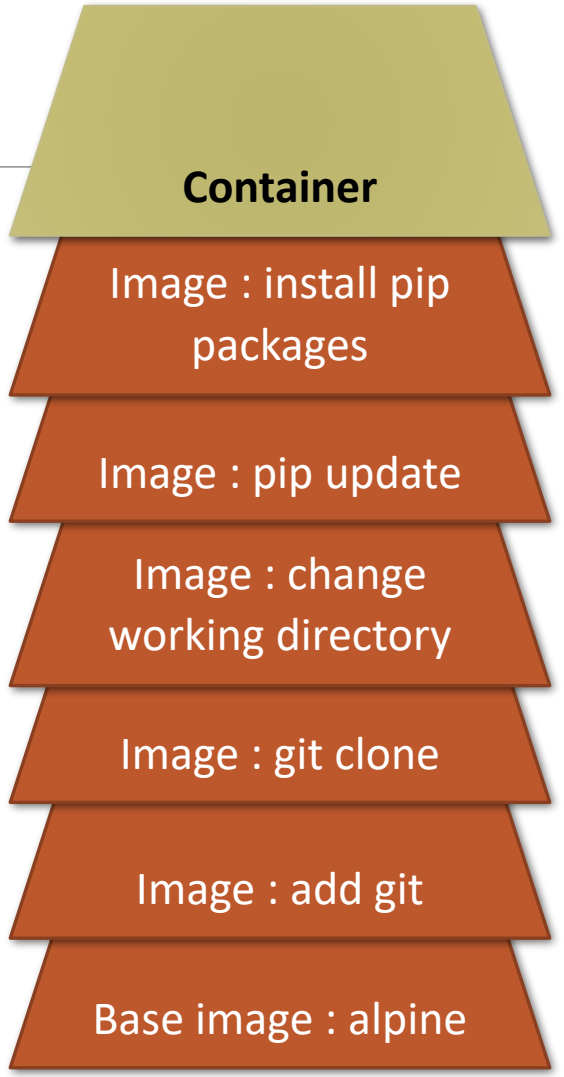


https://docs.docker.com/get-started/overview/

Docker

Container

- Defined by its image as well as any configuration options.
- A runnable instance of an image and a standardized unit of software that allows developers to isolate their app from its environment.
- You can create, start, stop, move, or delete a container using the Docker API or CLI.
- You can connect a container to one or more networks and attach storage to it.
  - Docker allocates a read-write filesystem to the container, as its final layer.



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

Docker

Create a container

- $ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

```
(msds603) ML-ITS-901885:weather_chekcking dwoodbridge$ docker run -e AWS_ACCESS_KEY_ID=[redacted]  -e AWS_SECRET_ACCESS_KEY=[redacted]
[redacted]  -e API_KEY=69[redacted]e -d -p 5000:5000 --name week1 weather-checking-app
b8dbf16f604e0a43ed460fd6f8ffdd1737312aea1d39c734a014bbad2811d822
(msds603) ML-ITS-901885:weather_chekcking dwoodbridge$
```

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

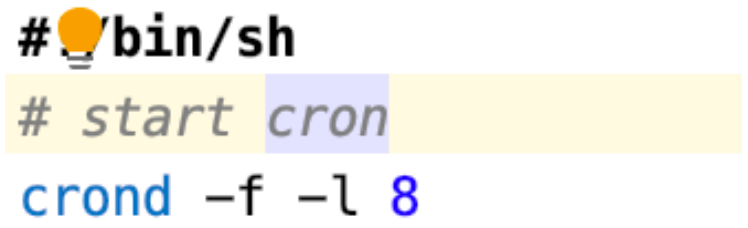
Example 1

Create images for weather-checking and cron and containers.

Example 1

Create images for weather-checking and cron and containers.

- Crontab is not working!!
  - `crond` : Executes cron jobs in the background.
    - `-f` : Foreground
    - `-b` : background
    - `-l N` : Set log level. (0: most verbose, 8: default)



https://linux.die.net/man/8/crond

https://gist.github.com/AntonFriberg/692eb1a95d61aa001dbb4ab5ce00d291

Example 1

Create images for weather-checking and cron and containers.

- Crontab is not working!!
  - Let's debug
    - Find a process running the cron container.
      - `$ docker ps`
    - Run sh.
      - `$ docker exec –it CONTAINER_ID /bin/sh`

```
ML-ITS-901885:cron dwoodbridge$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS                                   NAMES
03cc9c407a78   cron                   "/cron.sh"               About a minute ago   Up About a minute   0.0.0.0:57168->80/tcp                   cron-container
5a87a93ee387   weather-checking-app   "python application.…"   13 minutes ago       Up 13 minutes       0.0.0.0:80->80/tcp, :::80->80/tcp       week1
ML-ITS-901885:cron dwoodbridge$ docker exec -it 03cc9c407a78 /bin/sh
/2021-msds603-example/Week1/cron #
```

Example 1

Create images for weather-checking and cron and containers.

- Crontab is not working!!
  - Let's debug
    - Find a process running the cron container.
      - `$ docker ps`
    - Run sh.
      - `$ docker exec –it CONTAINER_ID /bin/sh`

```
/2021-msds603-example/Week1/cron # curl localhost/calculate
curl: (7) Failed to connect to localhost port 80: Connection refused
/2021-msds603-example/Week1/cron #
```

Doesn't understand localhost.

Example 1

Create images for weather-checking and cron and containers.

- Crontab is not working!!
  - Let's debug
    - Find a process running the cron container.
      - `$ docker ps`
    - Run sh.
      - `$ docker exec –it CONTAINER_ID /bin/sh`

```
/2021-msds603-example/Week1/cron # curl docker.for.mac.localhost/calculate
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: <a href="/index">/index</a>. If not click the link./2021-msds603-example/Week1/cron #
```

It makes work (in my Mac — won't work in other systems)
WAIT!! BUT, crontab still doesn't work!

Example 1

Create images for weather-checking and cron and containers.

- Crontab is not working!!
  - We fixed it by setting crontab to have
    * * * * * curl docker.for.mac.localhost/calculate
  - And run crond foreground.

This won't work on non-mac systems.
And we are going to run on Linux! :(

Example 1

Create images for weather-checking and cron and containers.

- Create a network in docker-compose.yml.

Docker

Docker Compose

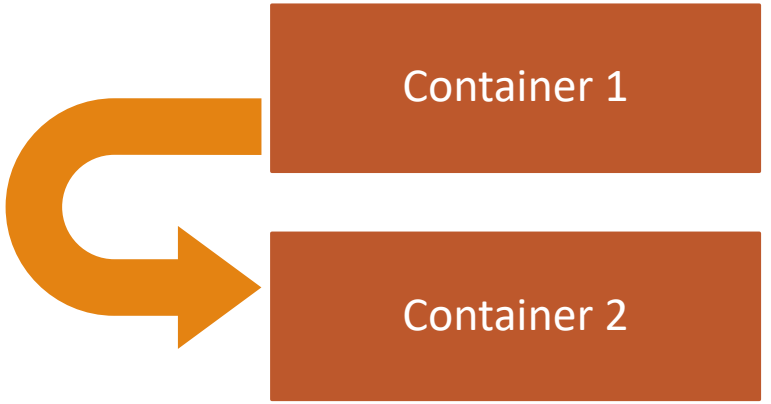
- Tool for defining and running multi-container Docker applications.
  - Configurations should be written in a YAML file called docker-compose.yml
    - Define services, networks and volumes.
    - YAML
      - Includes a list where members are at the same indentation level.
      - Conventional block format uses a **- (hyphen+space)** to begin a new item in list.
      - Has a list of dictionaries in a format of **key: value (colon+space)**

https://docs.docker.com/compose/
https://yaml.org/refcard.html

Docker

Docker Compose

- Tool for defining and running multi-container Docker applications.
  - Configurations should be written in a YAML file called docker-compose.yml
  - Define `services`, `networks` and `volumes`.
    - `services` : Started with a container name for a service along with configuration options including `build`, `context`, `dockerfile`, `args`, `environment`, `networks`, `container_name`, `depends_on`, `image`, `ports`, etc. that are applied at a build time.
    - `networks` : Started with a network name to be created with configuration options including `driver`, `internal`, `external`, `name`, etc.
    - `volumes` : Create named volumes that can be reused across multiple services with configuration options including `driver`, `external`, `name`, etc..



https://docs.docker.com/compose/compose-file/compose-file-v3/

Docker

Docker Compose

- Tool for defining and running multi-container Docker applications.
  - $docker-compose --env-file .env up

docker-compose.yml

```yaml
version: "3"
services:
  web:
    build:
      context: ../weather_checking
      dockerfile: Dockerfile
    ports:
      - "80:80"
    environment:
      - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
      - API_KEY=${API_KEY}
    container_name: web
    networks:
      - default
      - week1_network
  cron:
    build:
      context: ../cron
      dockerfile: Dockerfile
    container_name: cron
    depends_on:
      - web
    ports:
      - "80"
    networks:
      - default
      - week1_network

networks:
  week1_network:
```

https://docs.docker.com/compose/compose-file/compose-file-v3/

cronjobs

```
* * * * * curl web/calculate >> /var/log/cron.log 2>&1
```

Docker

Docker Compose

- Tool for defining and running multi-container Docker applications.
  - $docker-compose --env-file .env up

https://docs.docker.com/compose/compose-file/compose-file-v3/

Docker

Docker Compose

- Tool for defining and running multi-container Docker applications.
  - $docker-compose --env-file .env up

.env

```
AWS_ACCESS_KEY_ID:'[redacted]'
AWS_SECRET_ACCESS_KEY:'[redacted]
A[redacted]_KEY:'[redacted]'
```

https://docs.docker.com/compose/compose-file/compose-file-v3/

Docker

Docker Compose

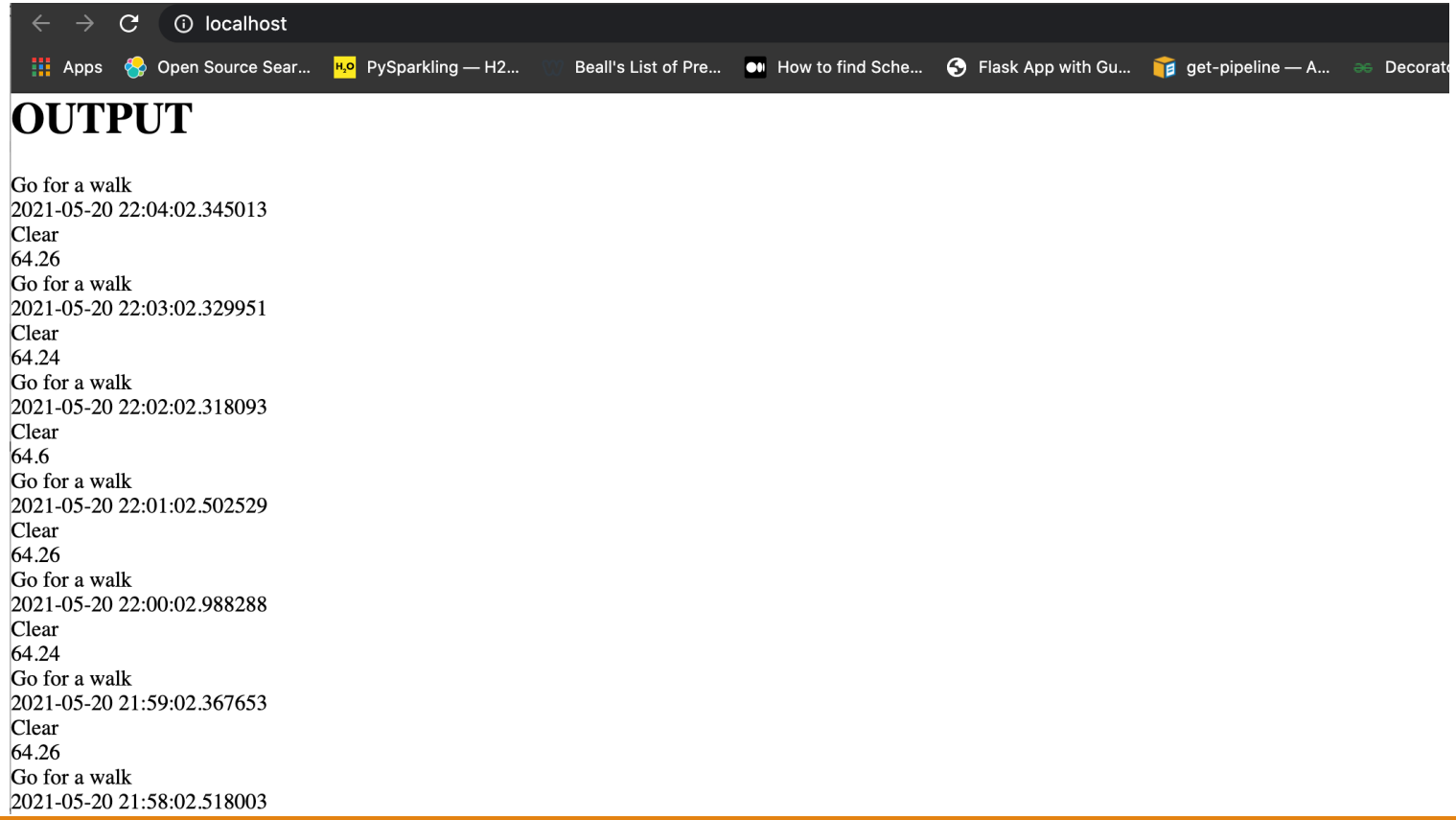
- Tool for defining and running multi-container Docker applications.
  - $docker-compose --env-file .env up

```
[   week1_network:(msds603) ML-ITS-901885:eb dwoodbridge$ docker-compose --env-file .env up
Building web
Step 1/8 : FROM python:3.7-alpine
 ---> ec8ed031b5be
Step 2/8 : RUN apk add git
 ---> Running in ce36c56ca4ca
fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/community/x86_64/APKINDEX.tar.gz
(1/5) Installing brotli-libs (1.0.9-r3)
(2/5) Installing nghttp2-libs (1.42.0-r1)
(3/5) Installing libcurl (7.76.1-r0)
(4/5) Installing pcre2 (10.36-r0)
(5/5) Installing git (2.30.2-r0)
Executing busybox-1.32.1-r6.trigger
OK: 23 MiB in 40 packages
Removing intermediate container ce36c56ca4ca
 ---> 984b5525dab9
Step 3/8 : RUN git clone https://github.com/dianewoodbridge/2021-msds603-example /2021-msds60
WARNING: Image for service cron was built because it did not already exist. To rebuild this i
Creating web ... done
Creating cron ... done
Attaching to web, cron
web      |  * Serving Flask app 'application' (lazy loading)
web      |  * Environment: production
web      |    WARNING: This is a development server. Do not use it in a production deployment.
web      |    Use a production WSGI server instead.
web      |  * Debug mode: on
web      |  * Running on all addresses.
web      |    WARNING: This is a development server. Do not use it in a production deployment.
web      |  * Running on http://172.30.0.2:80/ (Press CTRL+C to quit)
web      |  * Restarting with stat
web      |  * Debugger is active!
web      |  * Debugger PIN: 127-828-304
cron     |  * Serving Flask app 'application' (lazy loading)
cron     |  * Environment: production
cron     |    WARNING: This is a development server. Do not use it in a production deployment.
cron     |    Use a production WSGI server instead.
cron     |  * Debug mode: on
cron     |  * Running on all addresses.
cron     |    WARNING: This is a development server. Do not use it in a production deployment.
cron     |  * Running on http://172.30.0.3:80/ (Press CTRL+C to quit)
cron     |  * Restarting with stat
cron     |  * Debugger is active!
cron     |  * Debugger PIN: 182-705-019
```

https://docs.docker.com/compose/compose-file/compose-file-v3/

Example 1

Create images for weather-checking and cron and containers.



Docker

Docker Hub

- Repository of container images for storing and sharing images.

1. Create your Docker account : https://hub.docker.com/
2. Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
   - `$docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]`
3. Push the image to repository
   - `$docker push TARGET_IMAGE`

https://hub.docker.com/
https://docs.docker.com/docker-hub/

Docker

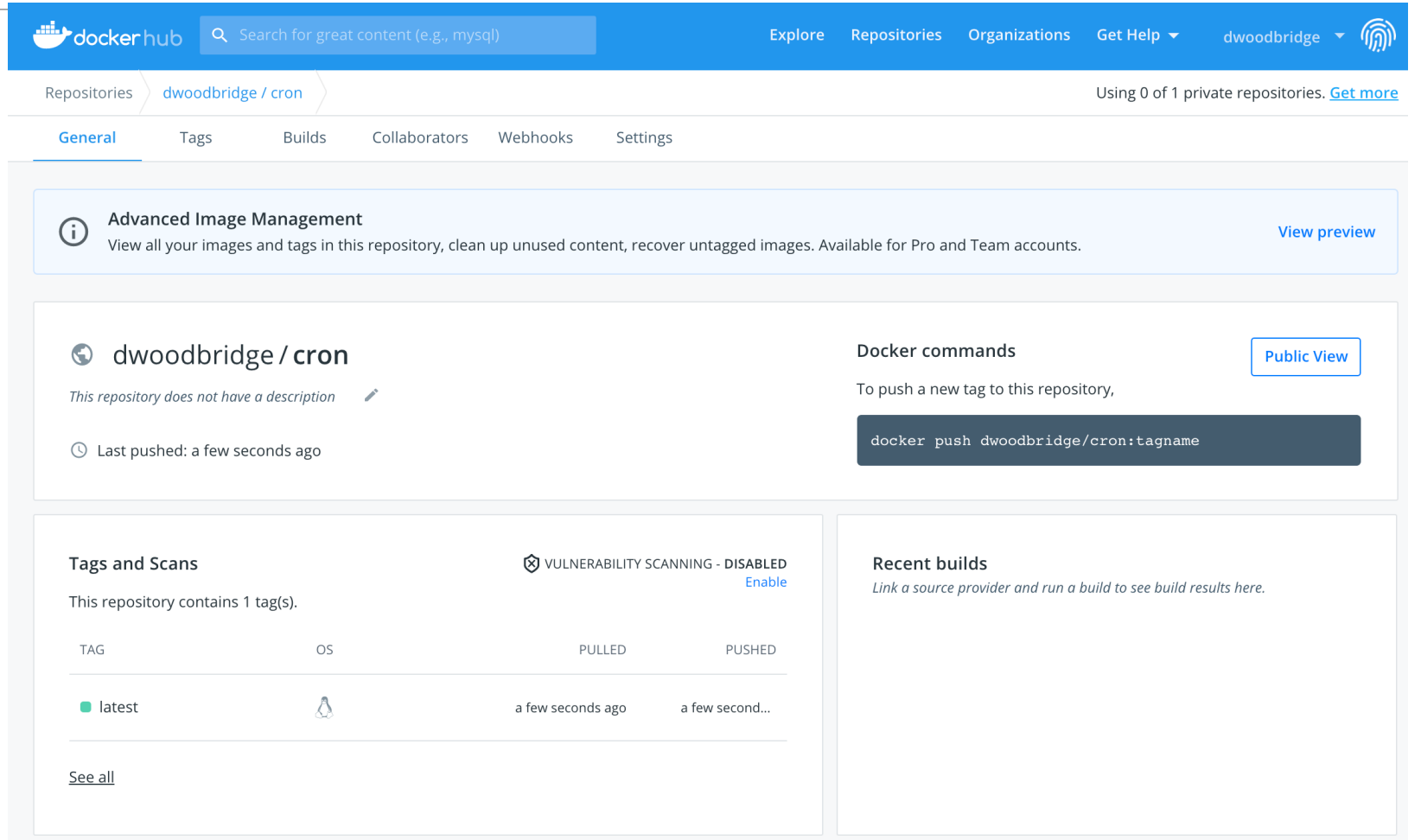
Docker Hub

```
(msds603) ML-ITS-901885:eb dwoodbridge$ docker tag 03_docker_compose_cron dwoodbridge/cron:latest
(msds603) ML-ITS-901885:eb dwoodbridge$ docker push dwoodbridge/cron
Using default tag: latest
The push refers to repository [docker.io/dwoodbridge/cron]
1952084b7130: Pushed
8b6a41edd4ee: Pushed
5e7fc301893e: Pushed
281a52bf9ecf: Pushed
d20ac3749e13: Pushed
17c67e0f5147: Pushed
9c48233bde85: Pushed
9624bf427ab5: Layer already exists
317c54183662: Layer already exists
4f007a2ceb29: Layer already exists
267a6a0302bf: Layer already exists
b2d5eeeaba3a: Layer already exists
latest: digest: sha256:25c3f57f77329c06612c1a6992ec81b11592035c5f169f579af76d464c36e819 size: 2824
```

https://hub.docker.com/
https://docs.docker.com/docker-hub/

Docker

Docker Hub



https://hub.docker.com/
https://docs.docker.com/docker-hub/

Example

Update docker-compose file to pull the image from dwoodbridge/weather_checking and dwoodbridge/cron.

Example

Update docker-compose file to pull the image from dwoodbridge/weather_checking and dwoodbridge/cron.

```yaml
version: "3"
services:
  web:
    image: dwoodbridge/weather_checking
    ports:
      - "80:80"
    environment:
      - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
      - API_KEY=${API_KEY}
    container_name: web
    networks:
      - default
      - week1_network
  cron:
    image: dwoodbridge/cron
    container_name: cron
    depends_on:
      - web
    ports:
      - "80"
    networks:
      - default
      - week1_network

networks:
   week1_network:
```

Contents

AWS Elastic Beanstalk

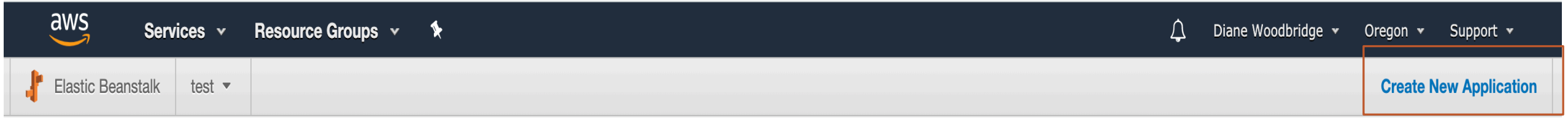
For easily deploying and scaling web applications.

- Utilizes EC2, S3, CloudWatch, Elastic Load Balancers, etc.
- Supports capacity provisioning, load balancing, auto-scaling, and application health monitoring.

https://aws.amazon.com/elasticbeanstalk/
https://en.wikipedia.org/wiki/AWS_Elastic_Beanstalk

Deploying a flask app to Elastic Beanstalk

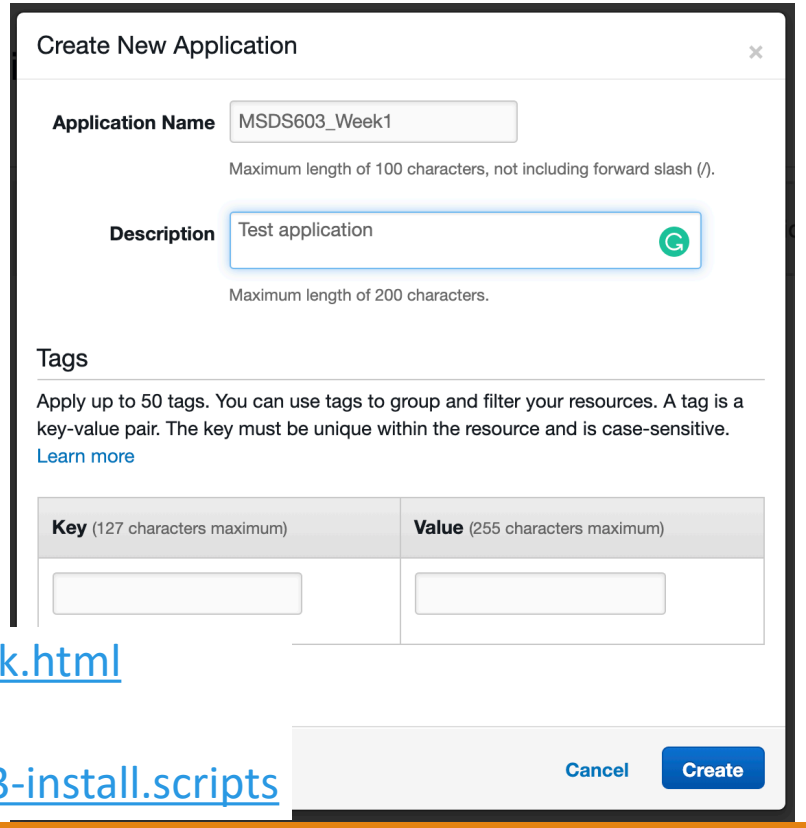
1. Go to AWS Console ➔ Elastic Beanstalk
2. Choose Create New Application



https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html#eb-cli3-install.scripts

Deploying a flask app to Elastic Beanstalk

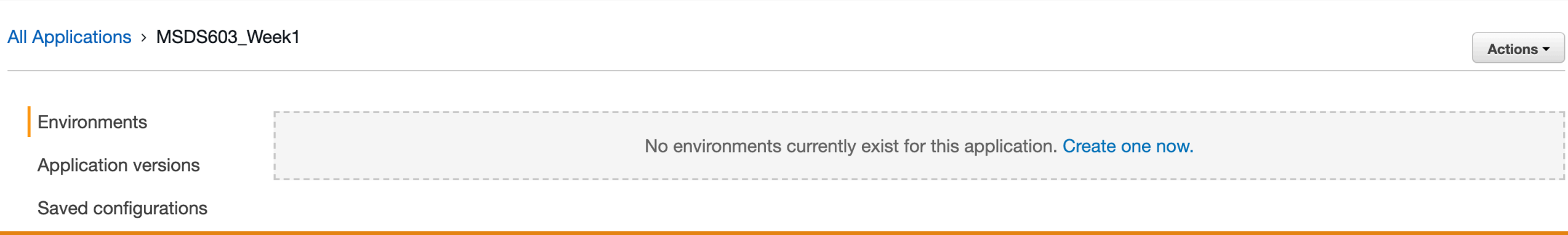
1. Go to AWS Console ➔ Elastic Beanstalk
2. Choose Create New Application
3. Enter application information



https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html#eb-cli3-install.scripts

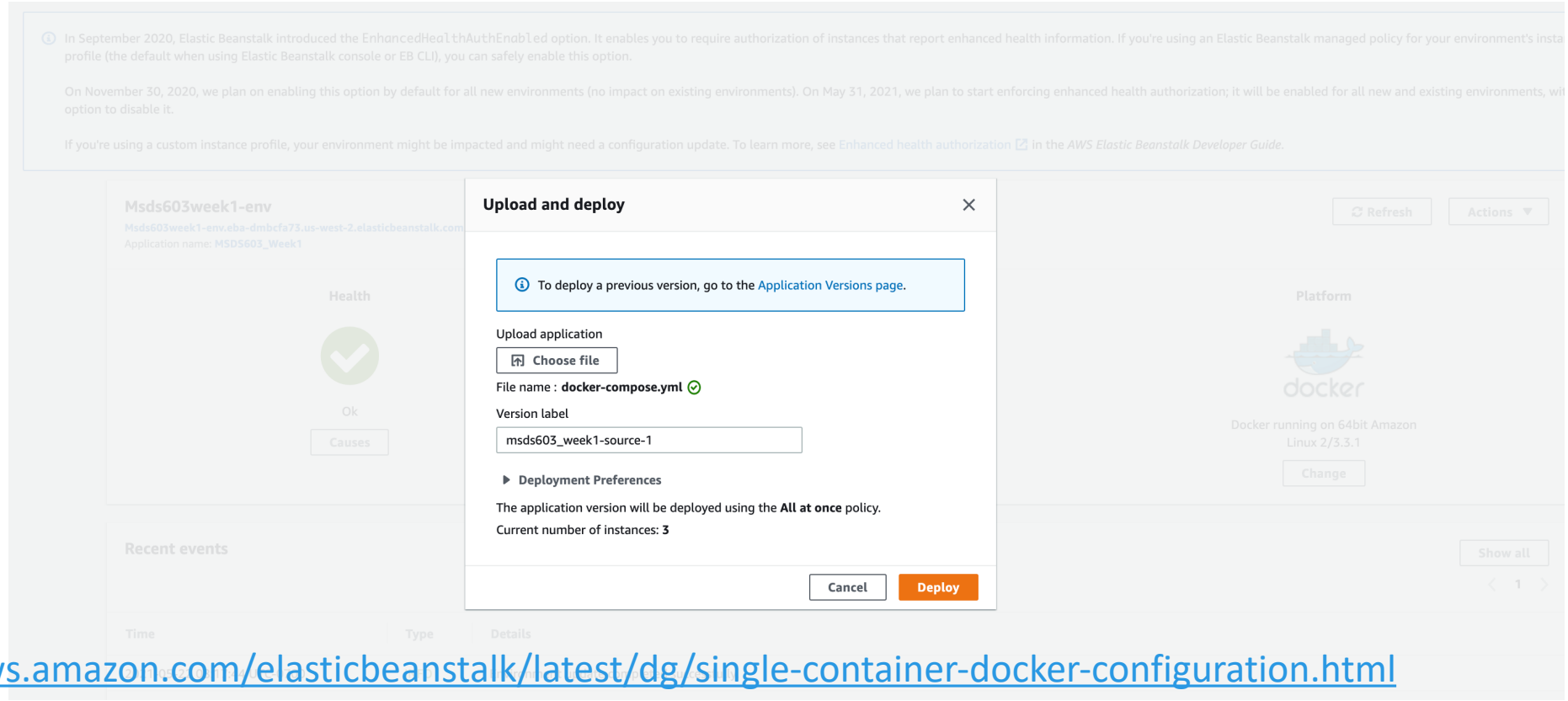
Deploying a flask app to Elastic Beanstalk

1. Go to AWS Console ➔ Elastic Beanstalk
2. Choose Create New Application
3. Enter application information
4. Create an Environment



Deploying a flask app to Elastic Beanstalk

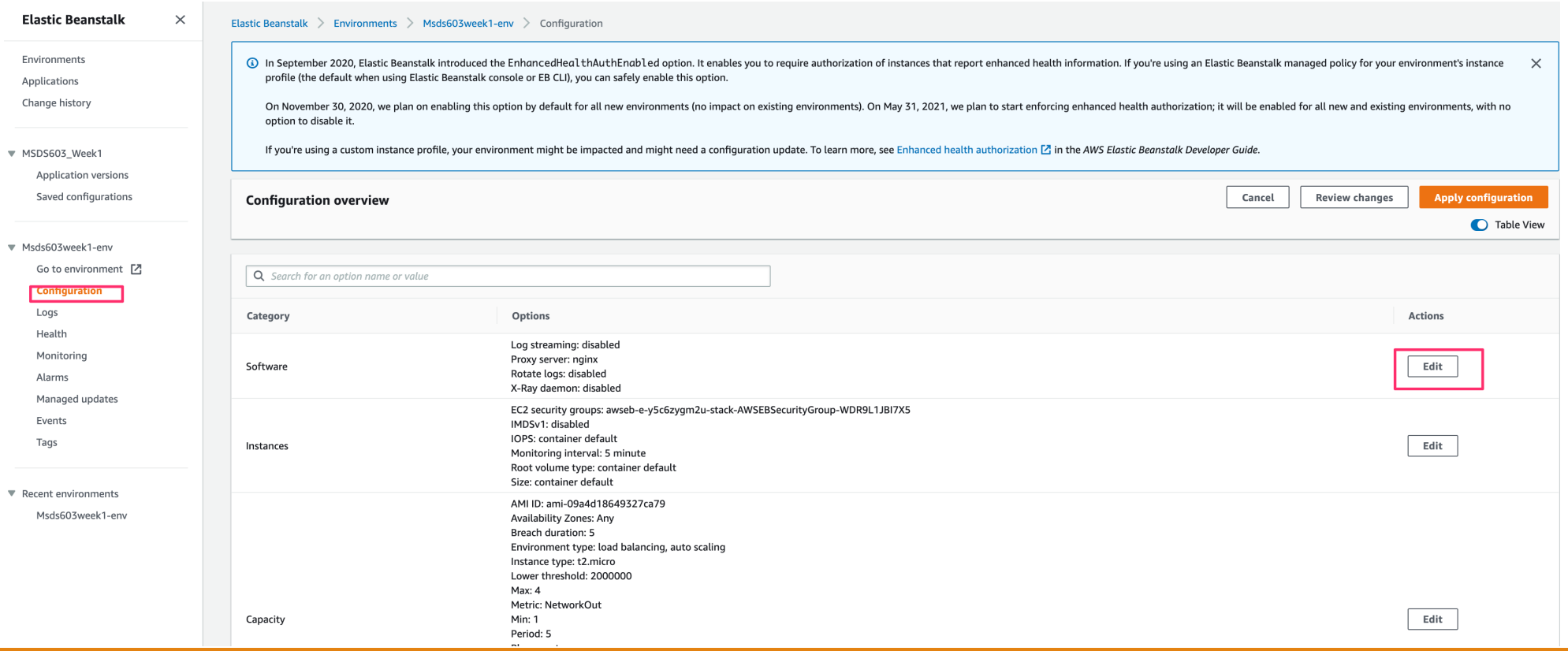
5. Upload the docker-compose.yml file.



https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/single-container-docker-configuration.html

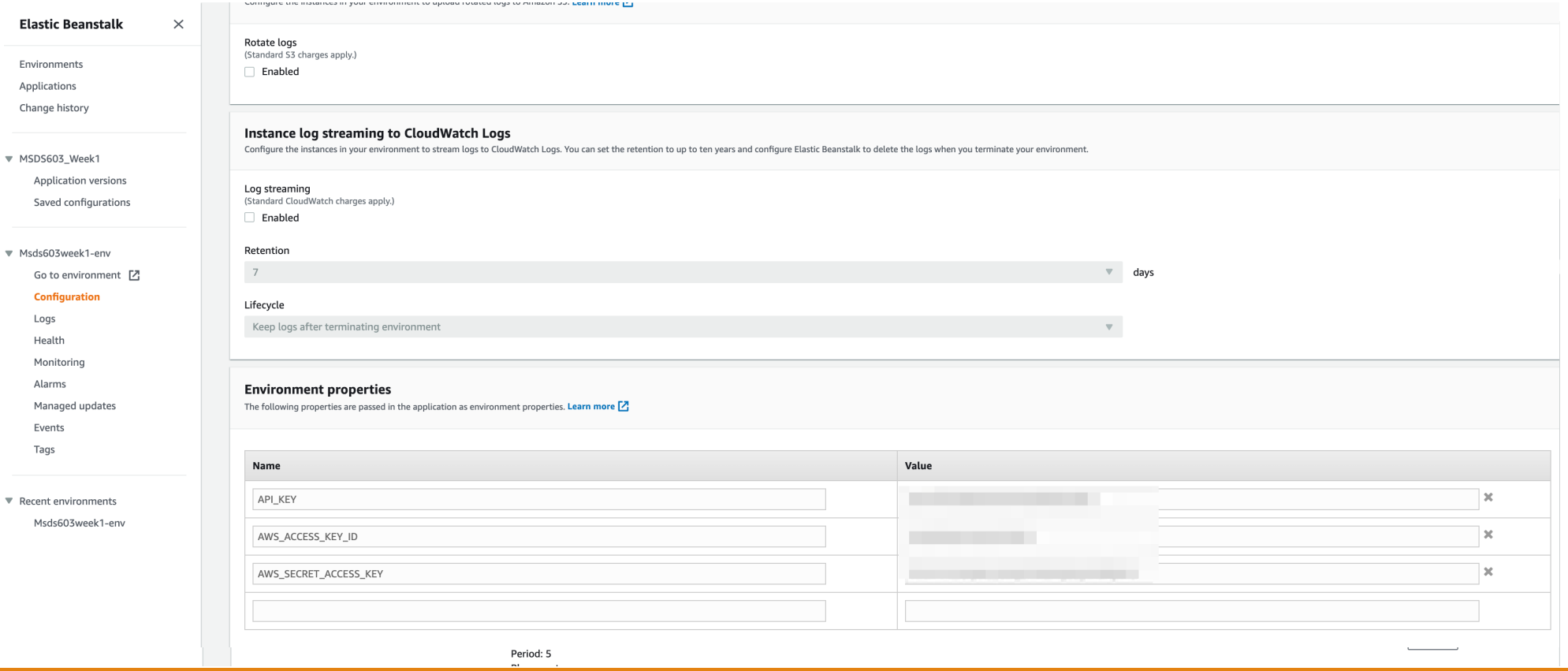
Deploying a flask app to Elastic Beanstalk

6. Update Environment Variable



Deploying a flask app to Elastic Beanstalk

6. Update Environment Variable



Deploying a flask app to Elastic Beanstalk

7. Done – You can open the URL to see the site and re-upload file using “Upload and Deploy”

Do it with Command Line Interface

1. Create an EB App
2. Create a environment under the app

Do it with Command Line Interface

Automate the process by developing a script.

1. Install the Elastic Beanstalk Command Line Interface (EB CLI).
   - `pip install awsebcli`

```
(MSDS603) ML-ITS-901885:driving_time_example dwoodbridge$ pip install awsebcli
Collecting awsebcli
Collecting six<1.12.0,>=1.11.0 (from awsebcli)
  Using cached https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bfa78ac9d43d8ad29a7ca43ea90a2d863fe3056e86a/six-1.11.0-py2.py3-none-any.whl
Collecting cement==2.8.2 (from awsebcli)
Collecting pathspec==0.5.9 (from awsebcli)
Collecting PyYAML<5.3,>=5.2 (from awsebcli)
Collecting botocore<1.15,>=1.14.0 (from awsebcli)
  Downloading https://files.pythonhosted.org/packages/f2/bc/788364aeb7d969a38c9d2295f95f1315a2ef49163f9b71d06de8c5ef0754/botocore-1.14.17-py2.py3-none-any.whl (5.9MB)
    100% |████████████████████████████████| 5.9MB 3.2MB/s
Collecting termcolor==1.1.0 (from awsebcli)
Collecting future<0.17.0,>=0.16.0 (from awsebcli)
Requirement already satisfied: wcwidth<0.2.0,>=0.1.7 in /Users/dwoodbridge/opt/anaconda3/envs/MSDS603/lib/python3.7/site-packages (from awsebcli) (0.1.7)
Collecting docker-compose<1.26.0,>=1.25.2 (from awsebcli)
  Using cached https://files.pythonhosted.org/packages/40/91/670b31a1c452cc5f3ed9e1ff905f6c4501a182551dfd8fa1981a8dca2d67/docker_compose-1.25.4-py2.py3-none-any.whl
```

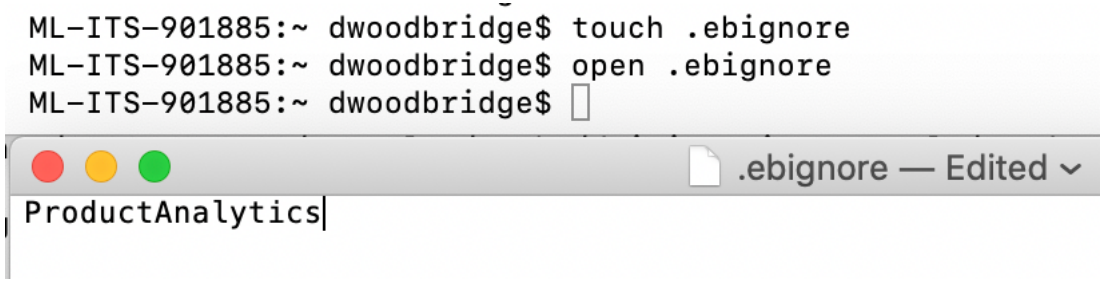
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html

https://github.com/aws/aws-elastic-beanstalk-cli-setup

1. Create an EB App

Automate the process by developing a script.

2. Create .ebignore file and add folder and files that don't need to be added to the application



https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html
https://github.com/aws/aws-elastic-beanstalk-cli-setup

1. Create an EB App

3. Initialize your EB CLI repository with

   `$ eb init -i`

```
(ProductAnalytics) (ProductAnalytics) ML-ITS-901885:driving_time_example dwoodbridge$ eb init

Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
15) ca-central-1 : Canada (Central)
16) eu-west-2 : EU (London)
19) ap-east-1 : Asia Pacific (Hong Kong)
```

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-init.html

Do it with Command Line Interface

1. Create an EB App
2. **Create an environment under the app**

2. Create two environments (Application and Worker) under the app

2. Create an environment and deploy your application to it with `$ eb create` (use `--envvars` for setting parameters)

```
(ProductAnalytics) (ProductAnalytics) ML-ITS-901885:driving_time_example dwoodbridge$ eb create ProductAnalytics-App
Creating application version archive "app-78d2-200311_145854".
Uploading ProductAnalytics2/app-78d2-200311_145854.zip to S3. This may take a while.
Upload Complete.
Environment details for: ProductAnalytics-App
  Application name: ProductAnalytics2
  Region: us-west-2
  Deployed Version: app-78d2-200311_145854
  Environment ID: e-tifuiz5jjh
  Platform: arn:aws:elasticbeanstalk:us-west-2::platform/Python 3.6 running on 64bit Amazon Linux/2.9.6
  Tier: WebServer-Standard-1.0
  CNAME: UNKNOWN
  Updated: 2020-03-11 21:59:00.006000+00:00
Printing Status:
2020-03-11 21:58:59    INFO    createEnvironment is starting.
2020-03-11 21:59:00    INFO    Using elasticbeanstalk-us-west-2-374226464461 as Amazon S3 storage bucket for environment data.
 -- Events -- (safe to Ctrl+C)
```

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-create.html

Contents

Shell Script

A file (.sh) includes a series of commands.

- You can execute it by calling `sh your_file.sh`

You can use various keywords and control flow.

- Commands – cd, ls, echo, pwd, touch etc.
- Keywords – if, else, break, etc.
- Control flow – if..then..else, case and for loops etc.

https://en.wikipedia.org/wiki/Shell_script

Shell Script

See what are the optional parameters that you can use.

```
(msds603) ML-ITS-901885:04_eb dwoodbridge$ eb init --help
usage: eb init <application_name> [options ...]

Initializes your directory with the EB CLI. Creates the application.

positional arguments:
  application_name      application name

optional arguments:
  -h, --help            show this help message and exit
  --debug               toggle debug output
  --quiet               suppress all output
  -v, --verbose         toggle verbose output
  --profile PROFILE     use a specific profile from your credential file
  -r REGION, --region REGION
                        use a specific region
  --no-verify-ssl       don't verify AWS SSL certificates
  -m [MODULES ...], --modules [MODULES ...]
                        module directory
  -p PLATFORM, --platform PLATFORM
                        default Platform
  -k KEYNAME, --keyname KEYNAME
                        default EC2 key name
```

Contents

Reference

Git - https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Sphinx - www.sphinx-doc.org/en/master/

Pytest - https://docs.pytest.org/en/latest/

Crontab generator - https://crontab-generator.org/

Docker - https://docs.docker.com/

AWS Elastic Beanstalk - https://docs.aws.amazon.com/elastic-beanstalk/index.html