COMP4442 Service and Cloud Computing

Lab 6

Outline

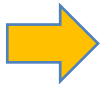
- Review of Lab5

- How to develop web app to visualize real-time data
  - Store data constantly to the database
  - Build a HTML chart with JavaScript
  - Create web app to visual real-time data with flask
  - Deploy code in AWS Beanstalk

Review of Lab5

- We learned how to use AWS Spark to count the frequency of words in large file.
- But how to visual the analytical results, e.g., in a webpage

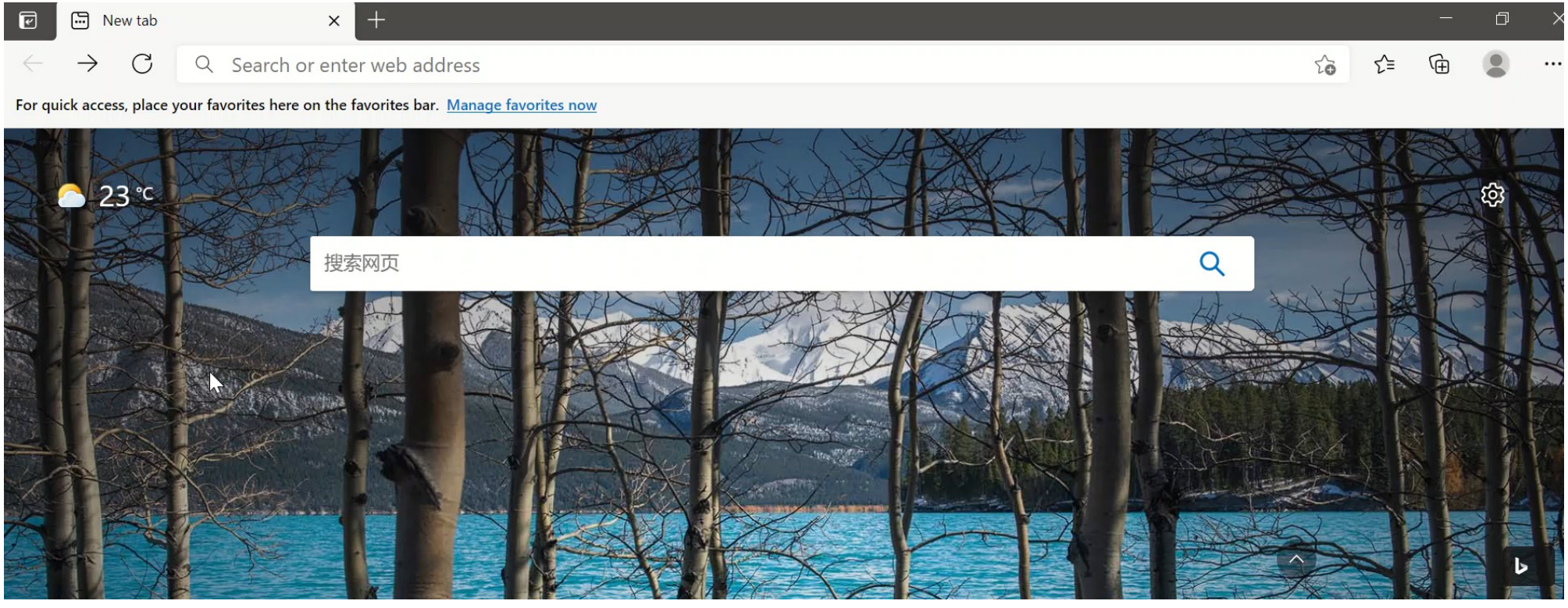
```
import os
import sys

from pyspark import SparkContext

args = sys.argv
inp = args[1]
out = args[2]

sc = SparkContext()

text_file = sc.textFile(inp)
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile(out)

sc.stop()
```



```
1  ('Gutenberg', 7)
2  ('Complete', 3)
3  ('Works', 3)
4  ('William', 64)
5  ('Shakespeare,', 1)
6  ('by', 2791)
7  ('This', 1101)
8  ('eBook', 2)
9  ('for', 5587)
10 ('the', 23104)
11 ('cost', 32)
12 ('and', 18173)
13 ('with', 6701)
14 ('almost', 135)
15 ('You', 1508)
16 ('copy', 14)
17 ('it', 4875)
18 ('or', 1717)
19 ('under', 196)
20 ('terms', 45)
21 ('License', 1)
22 ('included', 1)
23 ('**', 4)
24 ('a', 12472)
```

Visualize Real-time Data with Flask

- We will develop a web app to visualize real-time data
  - A database to store the constantly generated real-time data
  - A HTML chart to visualize and monitor the real-time data



Lab Preparation

- Configure the develop environment locally
  - Create an empty directory *lab6* and go to the directory `cd lab6`
  - Install the virtual environment `pip install virtualenv`
  - Create a virtual environment `virtualenv lab6`
  - Activate the virtual environment
    - `lab6\Scripts\activate` (windows)
    - `Source lab6/bin/activate` (macOS)
  - Install flask `pip install flask`
  - Install mysql-connector `pip install mysql-connector`

Store Data Constantly to Database

- Create a new table in a new database

```
drop database if exists lab6;
create database if not exists lab6;

use lab6;
CREATE TABLE IF NOT EXISTS Monitor
(
id int(11) unsigned NOT NULL AUTO_INCREMENT,
num int(11) DEFAULT NULL,
ctime bigint(11) DEFAULT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8;
```

Store Data Constantly to Database

- Create the *write.py* file to insert data to the database

```python
import time
import mysql.connector
import json
import random

def db_connection():
    mydb = mysql.connector.connect( host = 'comp4442-lab3.cicfnyxayefu.us-east-1.rds.amazonaws.com',
    user = 'admin',
    port = '3306',
    database = 'lab6',
    passwd = '12345678',
    autocommit = True)
    return mydb
mydb = db_connection()
cur = mydb.cursor()

def genData():
    number=15 + 5*random.randint(1,10)
    Time = int(time.time())
    data = {}
    data['time'] = Time
    data['number'] = number
    return data

def execute():
    data = genData()
    sql = "insert into Monitor(num,ctime) values ({0},{1})".format(data['number'],data['time'])
    ret = cur.execute(sql)

while True:
    execute()
    time.sleep(1)
```

→ Database connection

→ Generate real-time data

→ Insert data into database

Build a HTML chart with JavaScript

- Then, we create the HTML chart (Highchart)

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Real-time Monitoring</title>
  <script src="http://cdn.hcharts.cn/jquery/jquery-1.8.3.min.js"></script>
  <script src="http://cdn.hcharts.cn/highstock/highstock.js"></script>
  <script src="http://cdn.hcharts.cn/highcharts/modules/exporting.js"></script>
</head>

<body>
<div id="container" style="min-width:400px;height:400px"></div>
</body>

<script type="text/javascript">
$(function () {
  $.getJSON('/data', function (data) {
    // Create the chart
    $('#container').highcharts('StockChart', {
      chart: {
        events: {
          load: function () {
            var chart = $('#container').highcharts();
            var series = chart.series[0];
            //2 seconds interval
            setInterval(function () {
              $.getJSON("/data", function (res) {
                $.each(res, function (i, v) {
                  series.addPoint(v);
                });
              });
            }, 2000);
          }
        }
      },
      rangeSelector : {
        selected : 1
      },
      title : {
        text : 'Real-time Monitoring'
      },
      series : [{
        name : 'Real-time Number',
        data : data,
        tooltip: {
          valueDecimals: 2
        }
      }]
    });
  });
});
</script>
</html>
```

Claim the location of Highchart JavaScript

Set the interval as 2s, asynchronous update

Visual Real-time Data with Flask

- Bind the function `getdata()` to the url "`/data`"

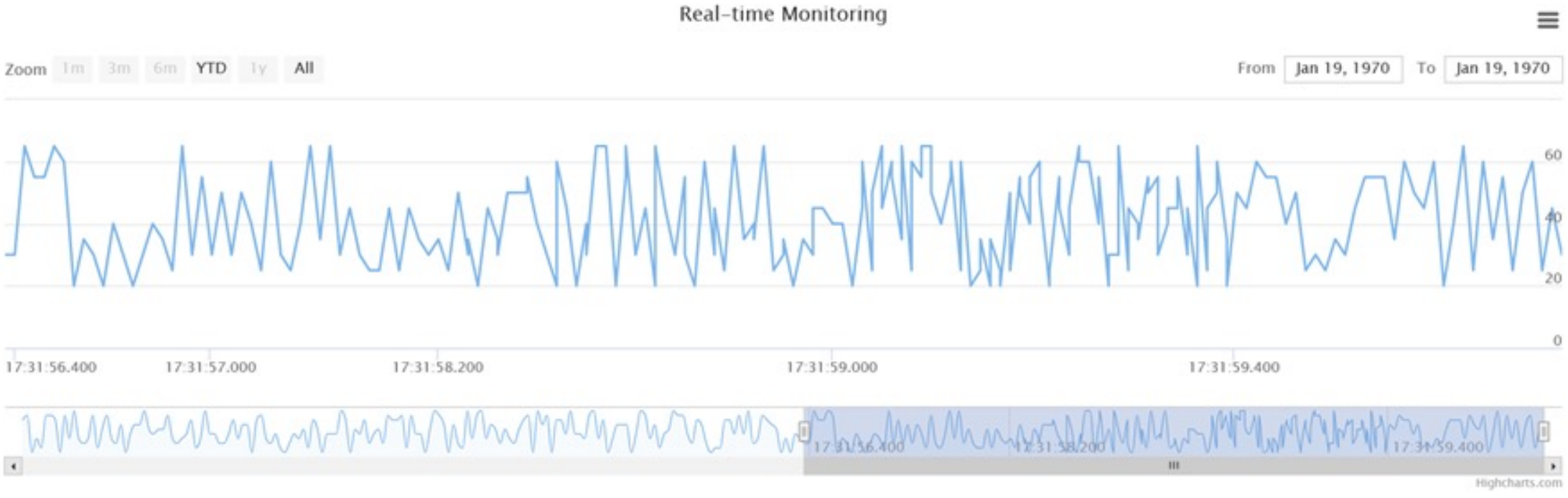
```python
from flask import Flask, request, render_template
import json
import mysql.connector

application = Flask(__name__)

def db_connection():
    mydb = mysql.connector.connect( host = 'comp4442-lab3.cicfnyxayefu.us-east-1.rds.amazonaws.com',
    user = 'admin',
    port = '3306',
    database = 'lab6',
    passwd = '12345678',
    autocommit = True)

    #print("successfully connect to the database")

    return mydb

mydb = db_connection()
cur = mydb.cursor()

@application.route("/")
def index():
 return render_template("monitor.html")

tmp_time = 0

@application.route("/data")
def getdata():
    global tmp_time
    if tmp_time > 0 :
        sql = "select ctime,num from Monitor where ctime >%s" %(tmp_time)
    else:
        sql = "select ctime,num from Monitor"

    cur.execute(sql)
    datas = []
    for i in cur.fetchall():
        datas.append([i[0], i[1]])

    if len(datas) > 0 :
        tmp_time = datas[-1][0]

    return json.dumps(datas)

if __name__ == "__main__":
    application.run(port=5000,debug=True)
```

→ Database connection

→ Query the data from the database and update the front-end webpage

Visual Real-time Data with Flask

- Check the chart to visualize data at "*http://127.0.0.1:5000/*".

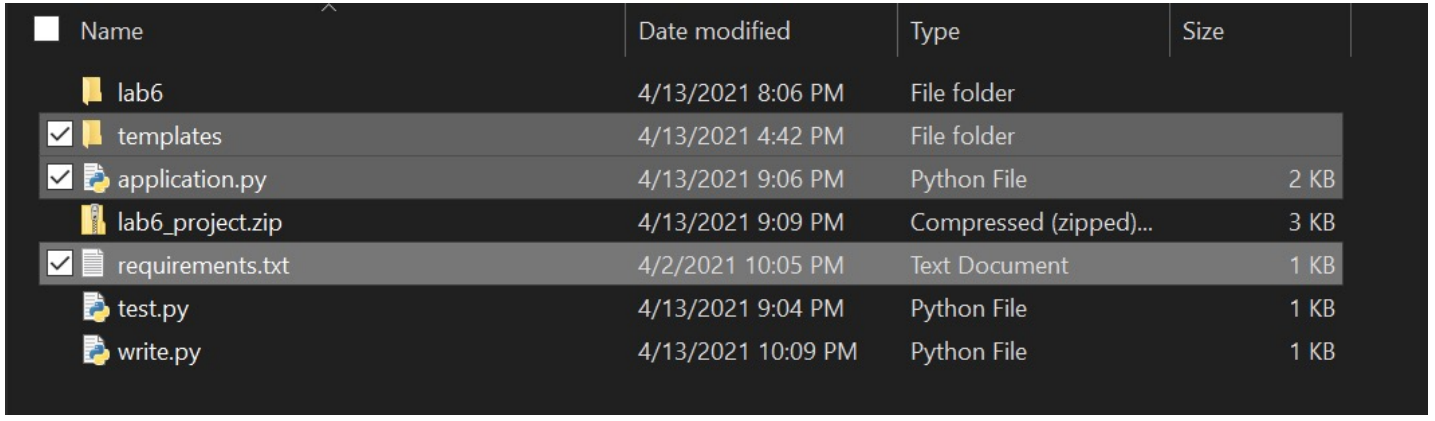


Deploy Code in AWS Beanstalk

- To deploy the code, we need to configure the requirements.txt.
  - `pip freeze > requirement.txt`
- Pack the following files and directories into a ZIP file.

requirements.txt

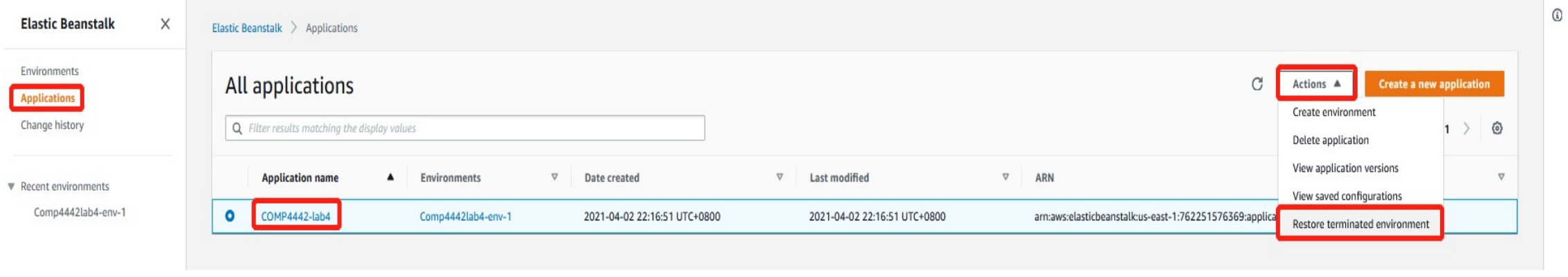
```
1  click==8.0.3
2  Flask==2.0.2
3  itsdangerous==2.0.1
4  Jinja2==3.0.3
5  MarkupSafe==2.0.1
6  mysql-connector==2.2.9
7  Werkzeug==2.0.3
8
```



Name	Date modified	Type	Size
lab6	4/13/2021 8:06 PM	File folder	
templates	4/13/2021 4:42 PM	File folder	
application.py	4/13/2021 9:06 PM	Python File	2 KB
lab6_project.zip	4/13/2021 9:09 PM	Compressed (zipped)...	3 KB
requirements.txt	4/2/2021 10:05 PM	Text Document	1 KB
test.py	4/13/2021 9:04 PM	Python File	1 KB
write.py	4/13/2021 10:09 PM	Python File	1 KB

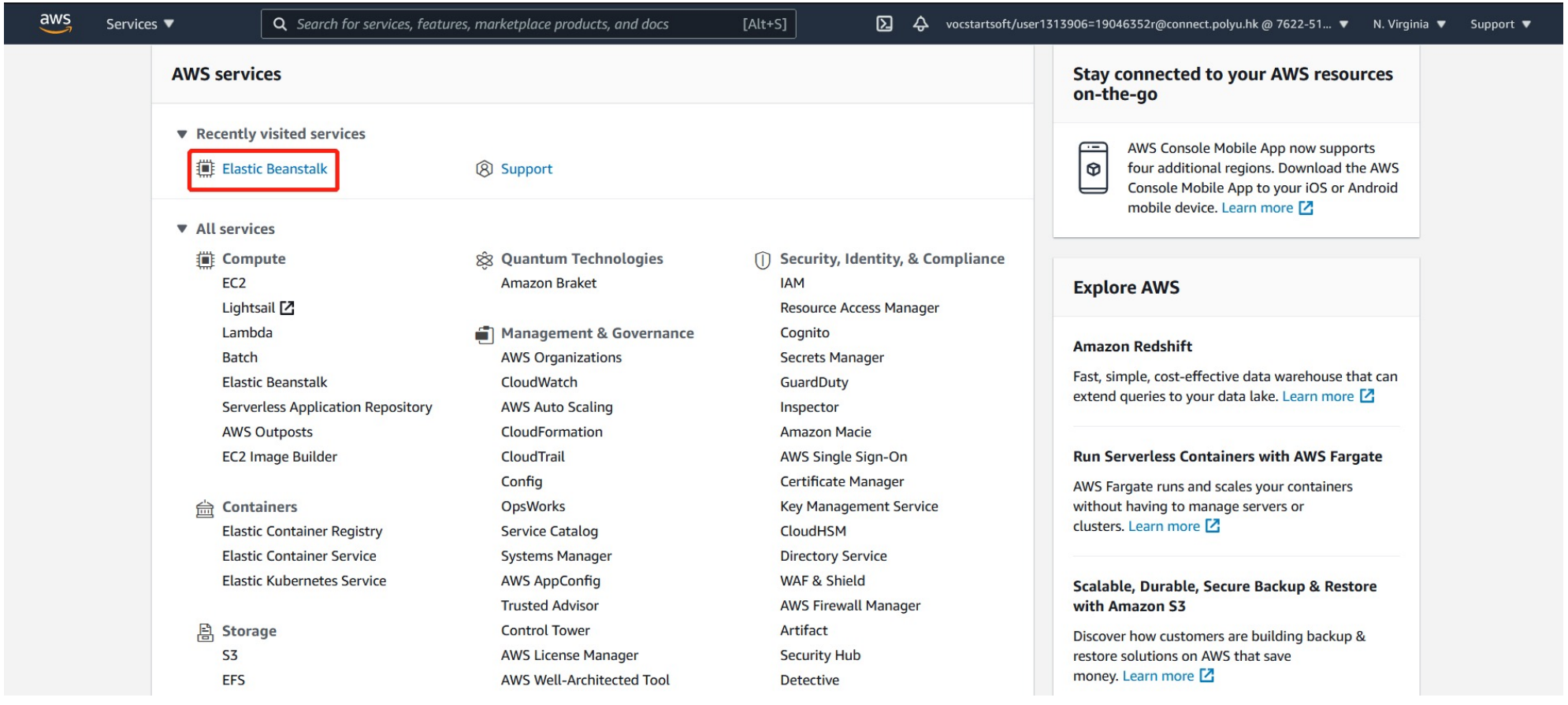
Deploy Code in AWS Beanstalk

- Restore Beanstalk environment and upload the ZIP file as we do in lab-01 and lab-04.
- Run the write.py locally



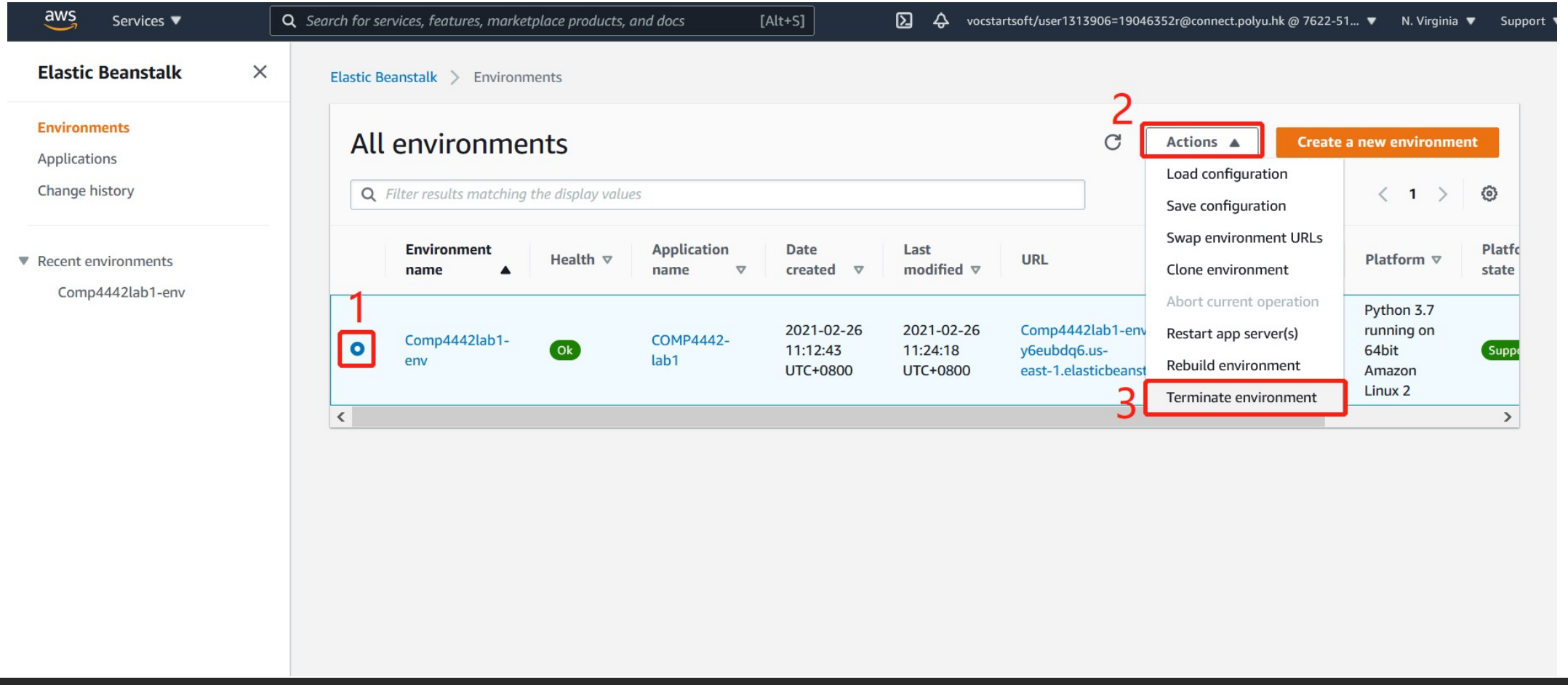
Stop Elastic Beanstalk

- Access the beanstalk



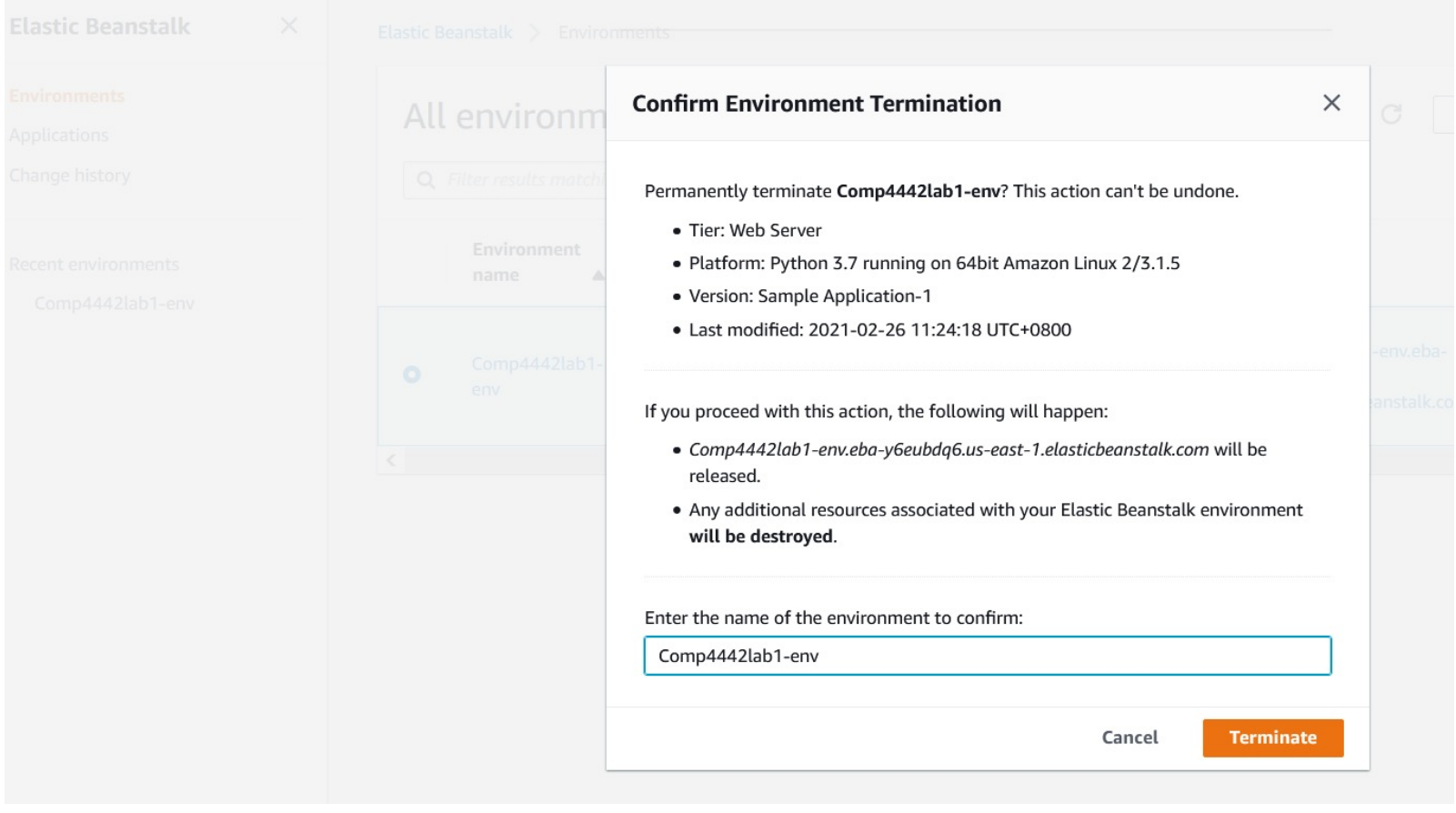
Stop Elastic Beanstalk

- Terminate the environment



Stop Elastic Beanstalk

- Confirm your operation



Lab Overview: What to Learn

- Introduction to Amazon Web Services (AWS)
- Data storage and management via AWS S3
  - Learn how to upload, access, and migrate data on AWS
- Create and manage databases via AWS RDS
  - Learn how to make use of database to support complex applications
- Building a dynamic web app via AWS Beanstalk
  - Learn how to use web server to deploy applications
- Big Data Analytics via AWS Spark
  - Learn how to use spark to analyze big data
- Visualize real-time data with dynamic web app

Q&A