AWS SERVER SETUP WITH DOCKER CONTAINER

MUNTABIR HASAN CHOUDHURY | MCHOU001@ODU.EDU

CS 431/531 – WEB SERVER DESIGN

OLD DOMINION UNIVERSITY
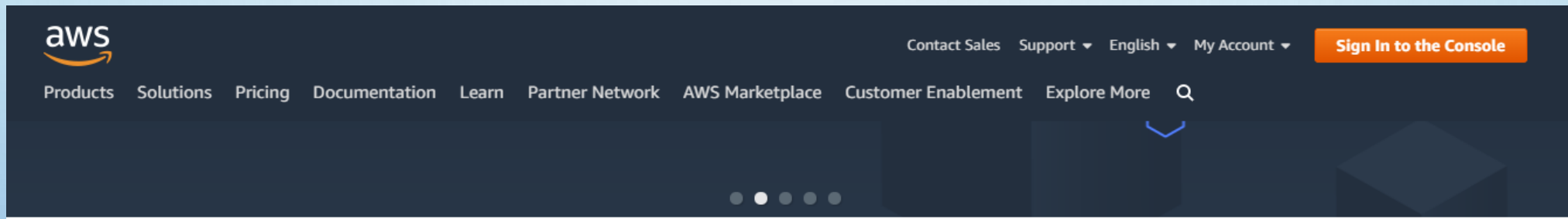
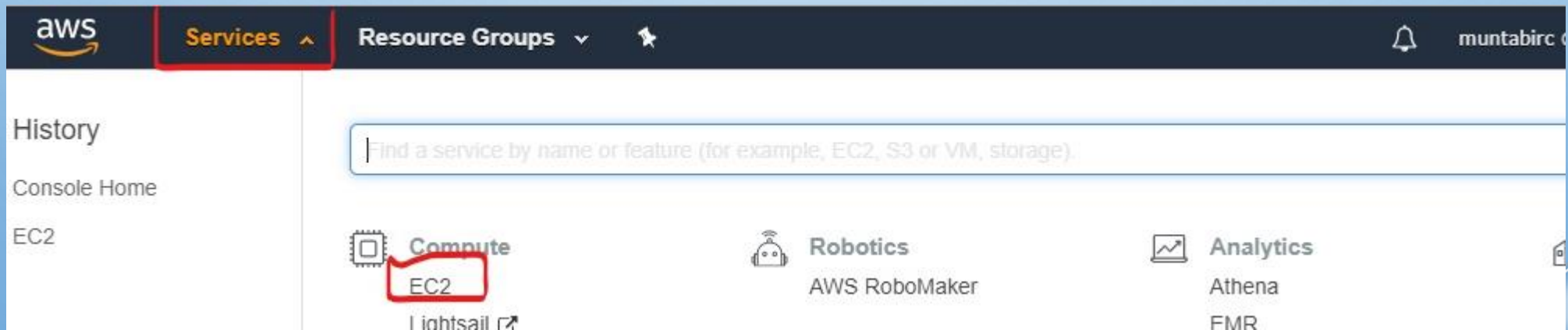DEPARTMENT OF COMPUTER SCIENCE

ACCOUNT SET UP

- OPEN A BROWSER AND GO TO AWS (HTTPS://AWS.AMAZON.COM/)

- OPEN A FREE TIER ACCOUNT FOR 12 MONTHS

- AWS WILL ASK YOU TO PUT YOUR CREDIT CARD INFORMATION BUT DO NOT WORRY SINCE IT IS A FREE TIER ACCOUNT

- BEFORE 12 MONTHS, YOU CAN CANCEL YOUR SUBSCRIPTION

EC2 INSTANCE

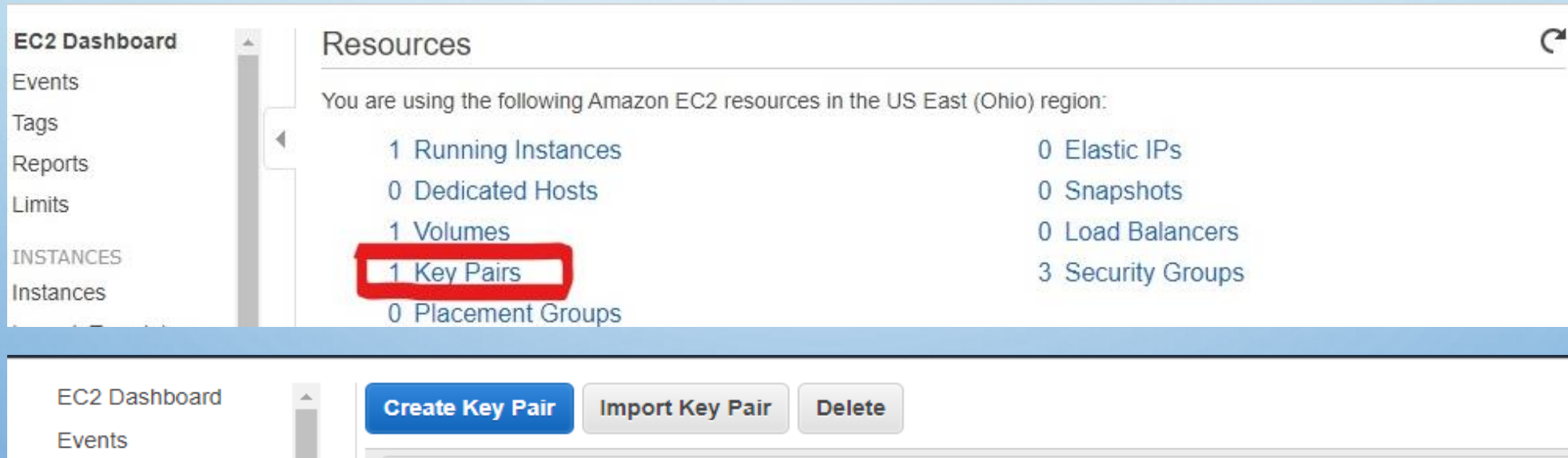- SIGN INTO AWS MANAGEMENT CONSOLE

- CLICK ON SERVICES AND SELECT 'EC2'

EC2 INSTANCE DASHBOARD

- CLICK ON KEY PAIRS AND CREATE A NEW KEY PAIR. IT WILL SAVED BE AS <YOUR_SELECTED_NAME>.PEM FILE IN YOUR COMPUTER

KEY PAIRS CONVERSION

- IF YOU ARE A MAC USER, YOU WILL NOT NEED TO CONVERT THE .PEM FILE (KEY PAIRS)

- IF YOU ARE A WINDOWS USER, YOU WILL NEED TO CONVERT THE .PEM FILE TO .PPK FILE USING 'PUTTY GEN'

- CLICK ON THE LINK BELOW FOR A COMPLETE GUIDE ON HOW TO CONVERT THE .PEM FILE TO .PPK FILE AND PERFORM SSH

- HTTPS://STACKOVERFLOW.COM/QUESTIONS/3190667/CONVERT-PEM-TO-PPK-FILE-FORMAT

CREATE AN INSTANCE

- CLICK ON LAUNCH INSTANCE AND SELECT THE FREE TIER ON THE LEFT NAVIGATION PANE

LAUNCH INSTANCE

- CHOOSE YOUR FAVORITE MACHINE IMAGE BASED ON THE OS (64 BIT, 32BIT) YOUR MACHINE IS RUNNING.

- PREFERABLY SELECT UBUNTU 18.04 64BIT IMAGE.

Amazon Linux
Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0c64dd618a49aeee8

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Select

64-bit (x86)

Red Hat
Free tier eligible

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0520e698dd500b1d1 (64-bit x86) / ami-0099847d600887c9f (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Select

◉ 64-bit (x86)
○ 64-bit (Arm)

SUSE Linux
Free tier eligible

SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type - ami-0e0bae59dc35fe89a (64-bit x86) / ami-0b49a8f443e46ff20 (64-bit Arm)

SUSE Linux Enterprise Server 15 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Select

◉ 64-bit (x86)
○ 64-bit (Arm)

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-05c1fa8df71875112 (64-bit x86) / ami-0606a0d9f566249d3 (64-bit Arm)

Free tier eligible

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Select

◉ 64-bit (x86)
○ 64-bit (Arm)

LAUNCH INSTANCE CONT..

- CLICK ON REVIEW AND LAUNCH

CONNECT YOUR INSTANCE

- ONCE YOU LAUNCHED YOUR INSTANCE,

- FOR WINDOWS USER –
  - CONNECT THE INSTANCE USING 'PUTTY'
  - A COMPLETE GUIDE (HTTPS://STACKOVERFLOW.COM/QUESTIONS/3190667/CONVERT-PEM-TO-PPK-FILE-FORMAT) CAN BE FOUND ON HOW TO CONNECT
  - HOWEVER, WHEN CONNECTING, USE THE "PUBLIC DNS / PUBLIC IP" FROM AWS AS A 'HOST NAME' ON PUTTY (DEMONSTRATED IN NEXT SLIDE)
  - KEEP THE PORT AS DEFAULT
  - CLICK ON OPEN
  - LOGIN AS "UBUNTU"

CONNECT YOUR INSTANCE CONT..

Connect To Your Instance ✕

I would like to connect with
- ⦿ A standalone SSH client ⓘ
- ◯ EC2 Instance Connect (browser-based SSH connection) ⓘ
- ◯ A Java SSH Client directly from my browser (Java required) ⓘ

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)

2. Locate your private key file (docker.pem). The wizard automatically detects the key you used to launch the instance.

3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

   `chmod 400 docker.pem`

4. Connect to your instance using its Public DNS:

   `ec2-18-222-188-28.us-east-2.compute.amazonaws.com`

Example:

`ssh -i "docker.pem" ubuntu@ec2-18-222-188-28.us-east-2.compute.amazonaws.com`

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Close

CONNECT YOUR INSTANCE

- FOR MAC USER,
  - GO TO THE TERMINAL
  - CHANGE THE MODE OF THE .PEM FILE
  - THEN CONNECT TO THE INSTANCE BY FOLLOWING THE "CONNECT TO YOUR INSTANCE"

COPY YOUR DOCKERFILE AND ECHO FILE

- ONCE YOU LOGGED INTO THE INSTANCE,
  - IF YOU ARE A WINDOWS USER, YOU CAN USE WINSCP TO TRANSFER YOUR FILE FROM LOCAL TO REMOTE
  - DOWNLOADABLE LINK FOR WINSCP: HTTPS://WINSCP.NET/ENG/DOWNLOAD.PHP
  - IF YOU ARE A MAC USER, YOU CAN USE "SCP" COMMAND TO TRANSFER YOUR FILE FROM LOCAL TO REMOTE (HTTPS://LINUXIZE.COM/POST/HOW-TO-USE-SCP-COMMAND-TO-SECURELY-TRANSFER-FILES/)
  - EXAMPLE: SCP FILE.TXT REMOTE_USERNAME@10.10.0.2:/REMOTE/DIRECTORY
- UPON TRANSFERRING THE FILES FROM LOCAL TO REMOTE -
  - IN THE EC2 INSTANCE, SIMPLY CHANGE THE DIRECTORY TO THE FOLDER WHICH CONTAINS THE DOCKER FILE
  - BUILD THE DOCKER IMAGE
  - RUN YOUR ECHO SERVER

```
ubuntu@ip-172-31-39-60:~/Assignment$ docker build -t docker-python-echoserver-app .
Sending build context to Docker daemon  7.168kB
Step 1/7 : FROM python:3.6
 ---> 1c515a624542
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> d6f0dd07e6e2
Step 3/7 : COPY src/requirements.txt ./
 ---> Using cache
 ---> 06da4a9991aa
Step 4/7 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> eeb8a1c932e2
Step 5/7 : COPY src /app
 ---> 4c8474023e1a
Step 6/7 : EXPOSE 80
 ---> Running in 143d8b7e9956
Removing intermediate container 143d8b7e9956
 ---> 9e926c5d5a08
Step 7/7 : CMD ["python", "echo-server.py"]
 ---> Running in 2ff748b540b4
Removing intermediate container 2ff748b540b4
 ---> dc146d069f6c
Successfully built dc146d069f6c
Successfully tagged docker-python-echoserver-app:latest
ubuntu@ip-172-31-39-60:~/Assignment$ docker run --rm -it -p 80:80 docker-python-echoserver-app
Listening on 0.0.0.0:80 for HTTP connection
```

RUN ECHO SERVER IN DOCKER

PING AND TELNET YOUR IP

```
ubuntu@ip-172-31-39-60:~$ ping 3.19.143.89
PING 3.19.143.89 (3.19.143.89) 56(84) bytes of data.
64 bytes from 3.19.143.89: icmp_seq=1 ttl=63 time=0.282 ms
64 bytes from 3.19.143.89: icmp_seq=2 ttl=63 time=0.395 ms
64 bytes from 3.19.143.89: icmp_seq=3 ttl=63 time=0.379 ms
64 bytes from 3.19.143.89: icmp_seq=4 ttl=63 time=0.414 ms
64 bytes from 3.19.143.89: icmp_seq=5 ttl=63 time=0.553 ms
64 bytes from 3.19.143.89: icmp_seq=6 ttl=63 time=0.348 ms
64 bytes from 3.19.143.89: icmp_seq=7 ttl=63 time=0.339 ms
^C
--- 3.19.143.89 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6151ms
rtt min/avg/max/mdev = 0.282/0.387/0.553/0.079 ms
ubuntu@ip-172-31-39-60:~$ telnet 3.19.143.89 80
Trying 3.19.143.89...
Connected to 3.19.143.89.
Escape character is '^]'.
```

- PING <YOUR_IP_ADDRESS>

- TELNET <YOUR_IP_ADDRESS> <PORT>

- YOU SHOULD BE ABLE TO SEE A MESSAGE THAT IS: "CONNECTED <YOUR_IP_ADDRESS>"

- SO, THE ECHO-SERVER CAN BE ACCESSIBLE FROM ELSEWHERE

ECHOED BACK FROM THE SERVER AFTER TELNET