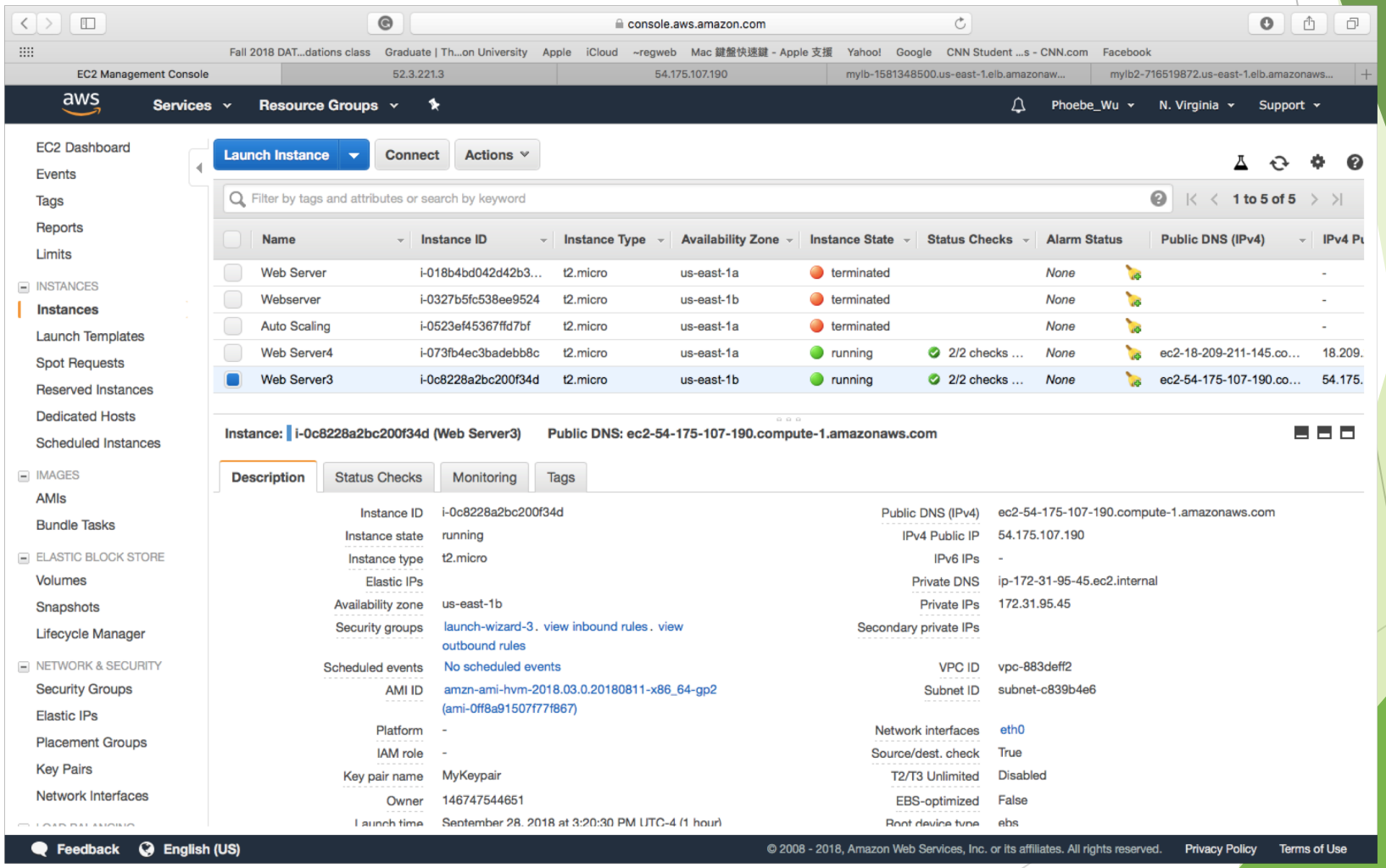
Cloud Computing
Lab1

Phoebe Wu

Part-I:
Compute Instances

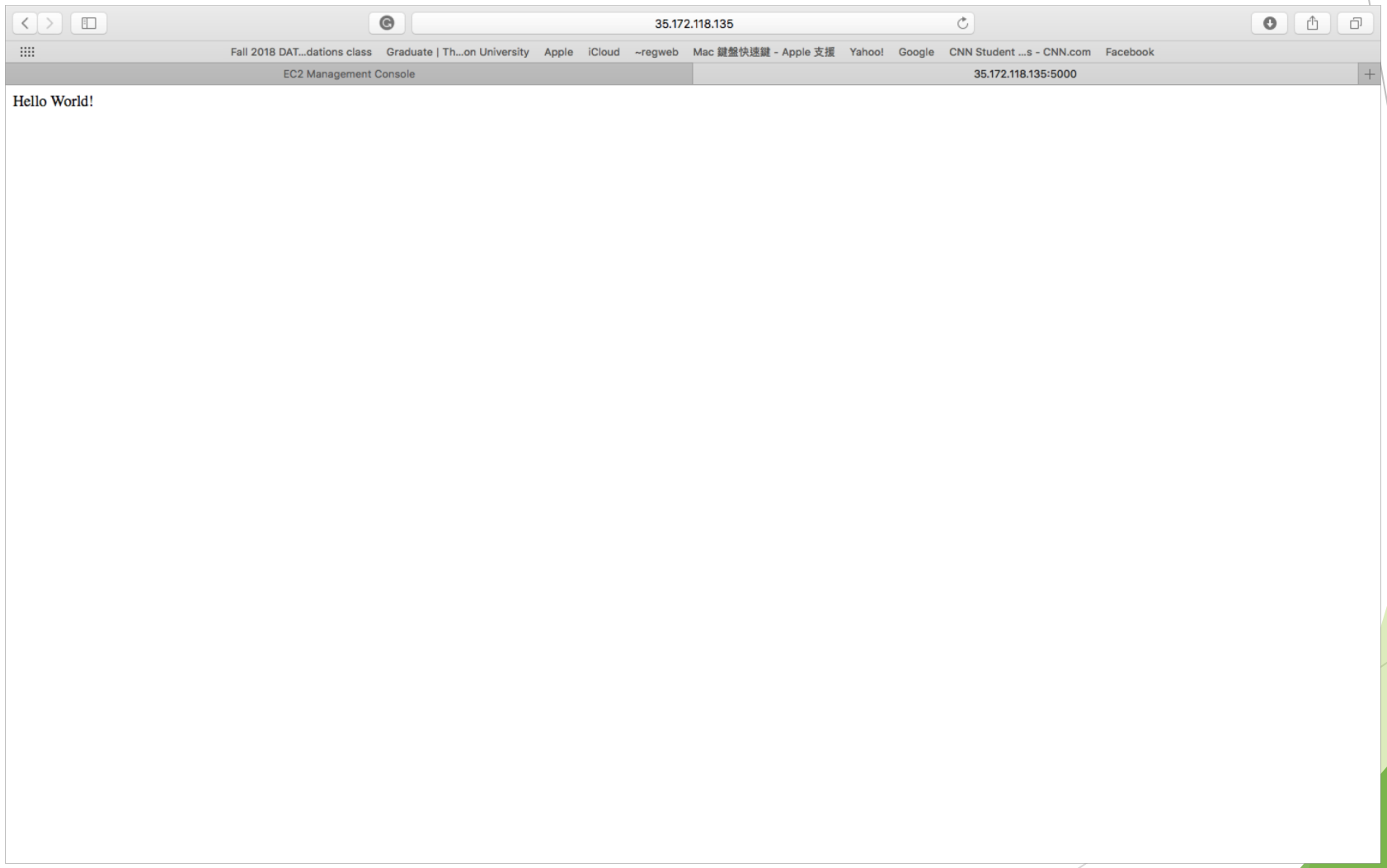
Launch EC2 instance



Virtualization Type of EC2

- I think Amazon EC2 is a kind of Hardware virtualization.

Exercise-2(Setup Flask and run hello.py):

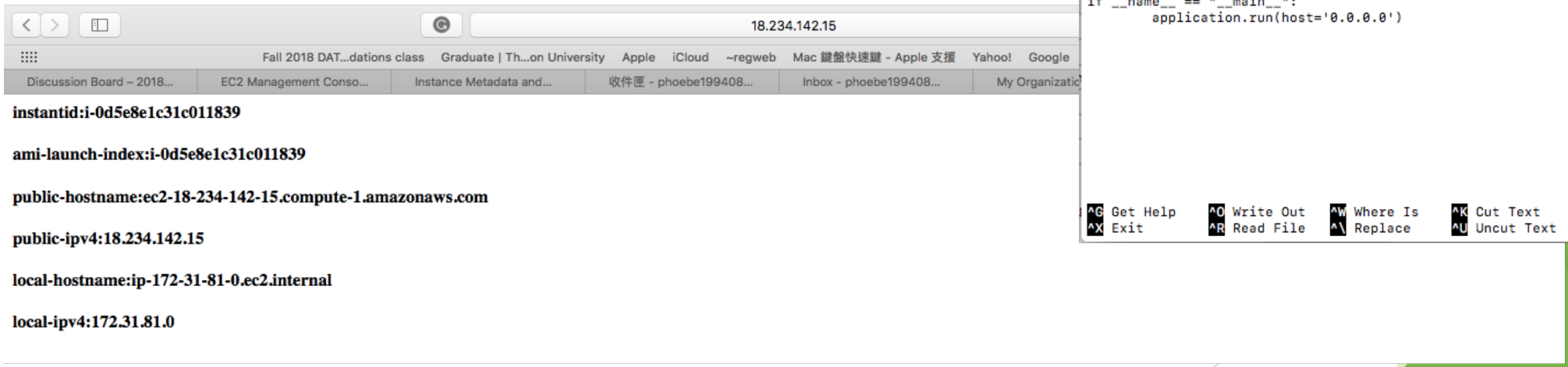


Exercise-3(Meta-data flask application):

First, run the commands:
curl http://169.254.169.254/latest/meta-data/instance-id > a.txt
curl http://169.254.169.254/latest/meta-data/instance-id > b.txt
curl http://169.254.169.254/latest/meta-data/public-hostname > c.txt
curl http://169.254.169.254/latest/meta-data/public-ipv4 > d.txt
curl http://169.254.169.254/latest/meta-data/local-hostname > e.txt
curl http://169.254.169.254/latest/meta-data/local-ipv4 > f.txt

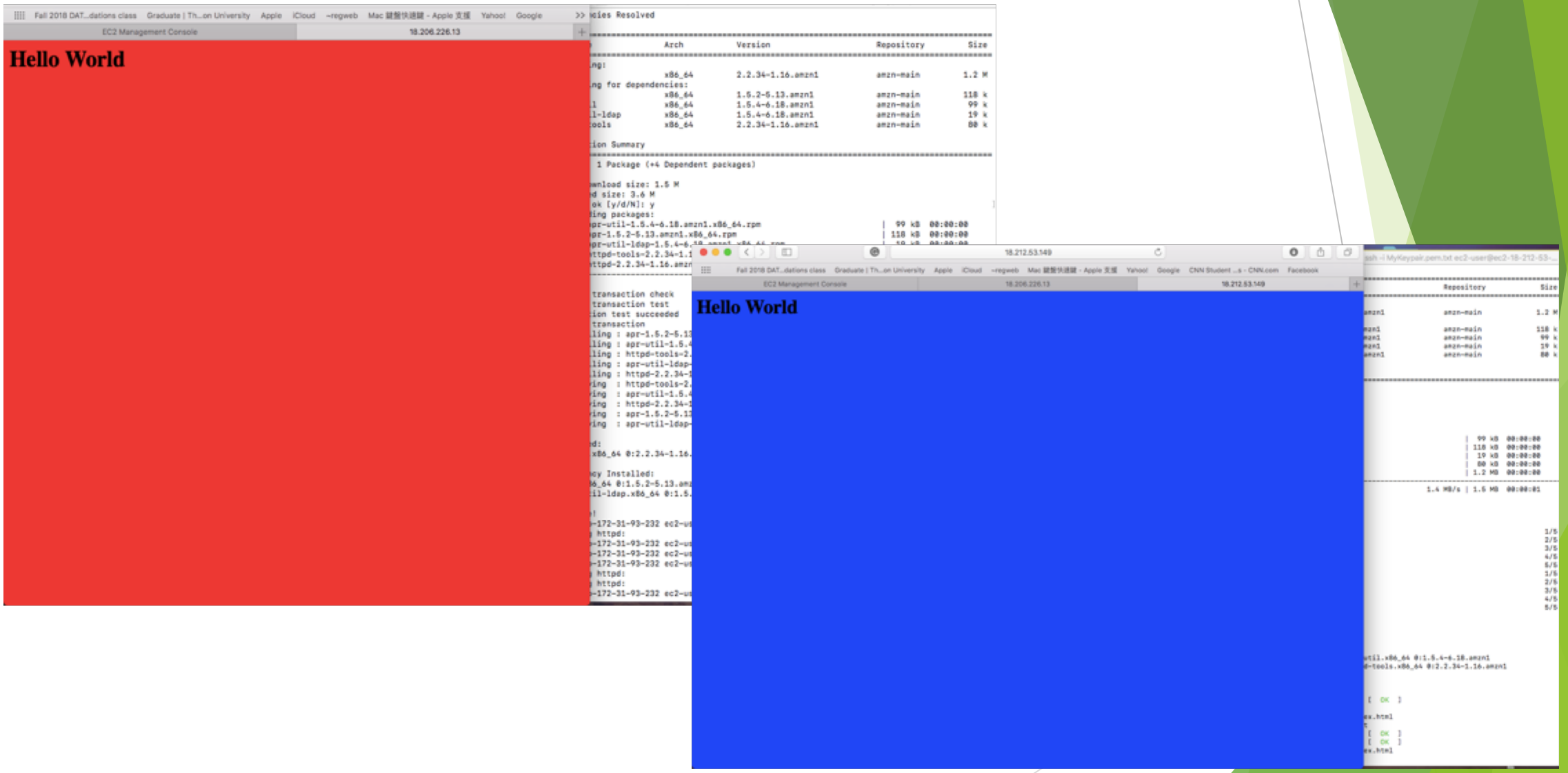
```
 GNU nano 2.5.3                    File: ex3_metadata

from flask import Flask
application = Flask(__name__)
@application.route("/")
def hello():
        content=" "
        with open("a.txt",'r') as rd1:
                instantid = rd1.read()
        with open("b.txt",'r') as rd2:
                index = rd2.read()
        with open("c.txt",'r') as rd3:
                phostname = rd3.read()
        with open("d.txt",'r') as rd4:
                pip = rd4.read()
        with open("e.txt",'r') as rd5:
                lhostname = rd5.read()
        with open("f.txt",'r') as rd6:
                lip = rd6.read()
                content +="<h4>instantid:"+instantid+"</h4>"
                content +="<h4>ami-launch-index:"+index+"</h4>"
                content +="<h4>public-hostname:"+phostname+"</h4>"
                content +="<h4>public-ipv4:"+pip+"</h4>"
                content +="<h4>local-hostname:"+lhostname+"</h4>"
                content +="<h4>local-ipv4:"+lip+"</h4>"
        return content
if __name__ == "__main__":
        application.run(host='0.0.0.0')

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

The output:

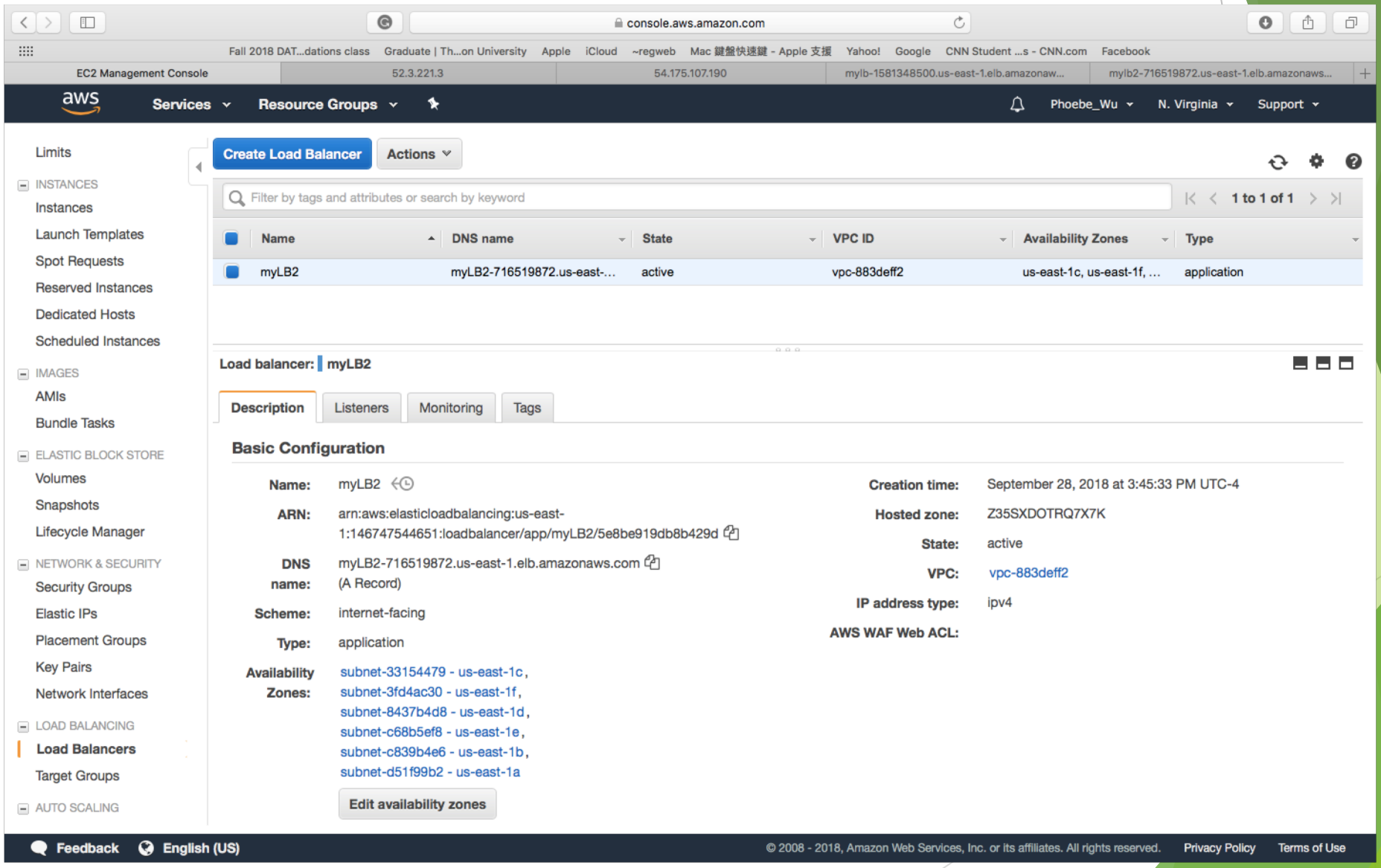


Part-II:
Load Balancing

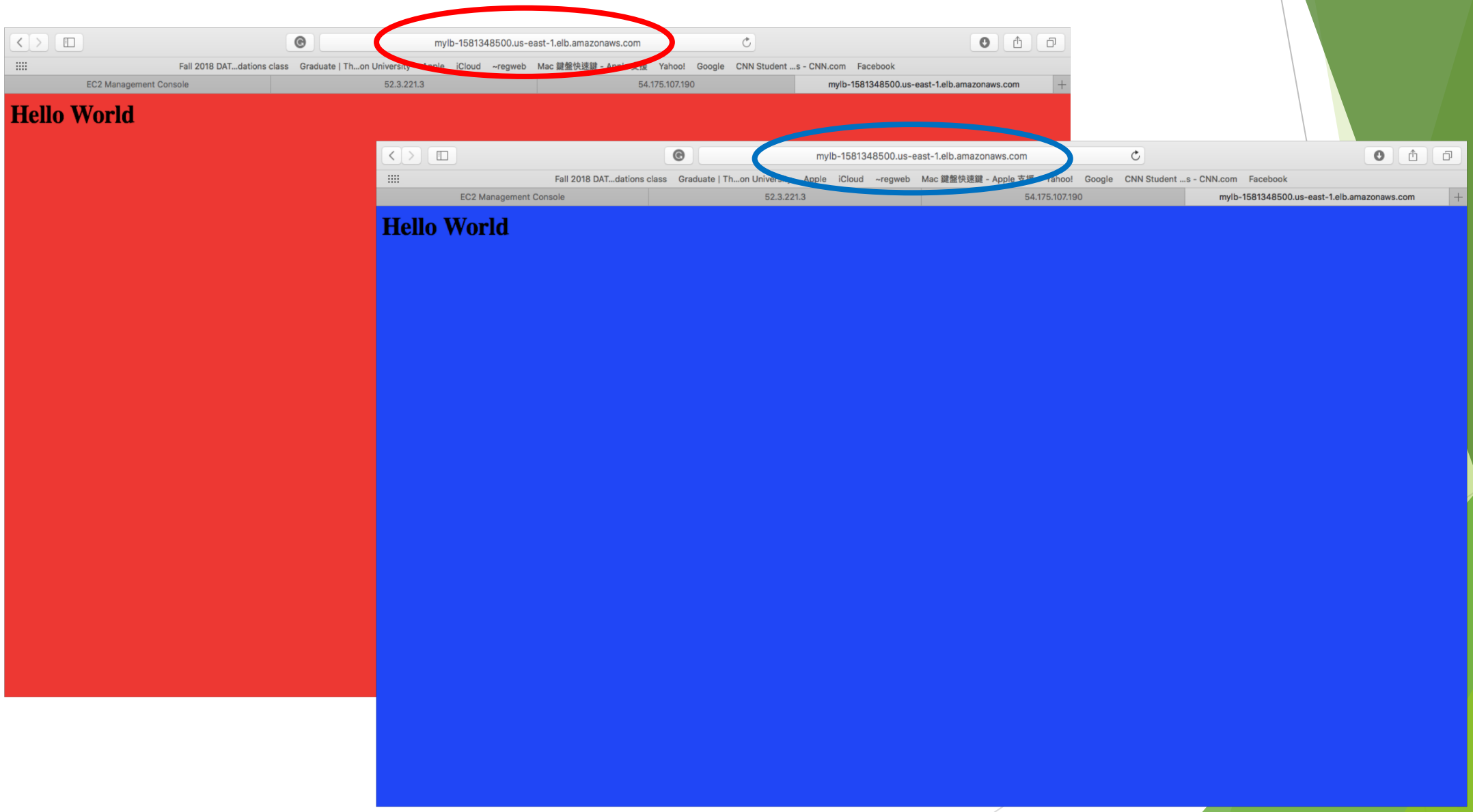
Create two instances and run the hello.py



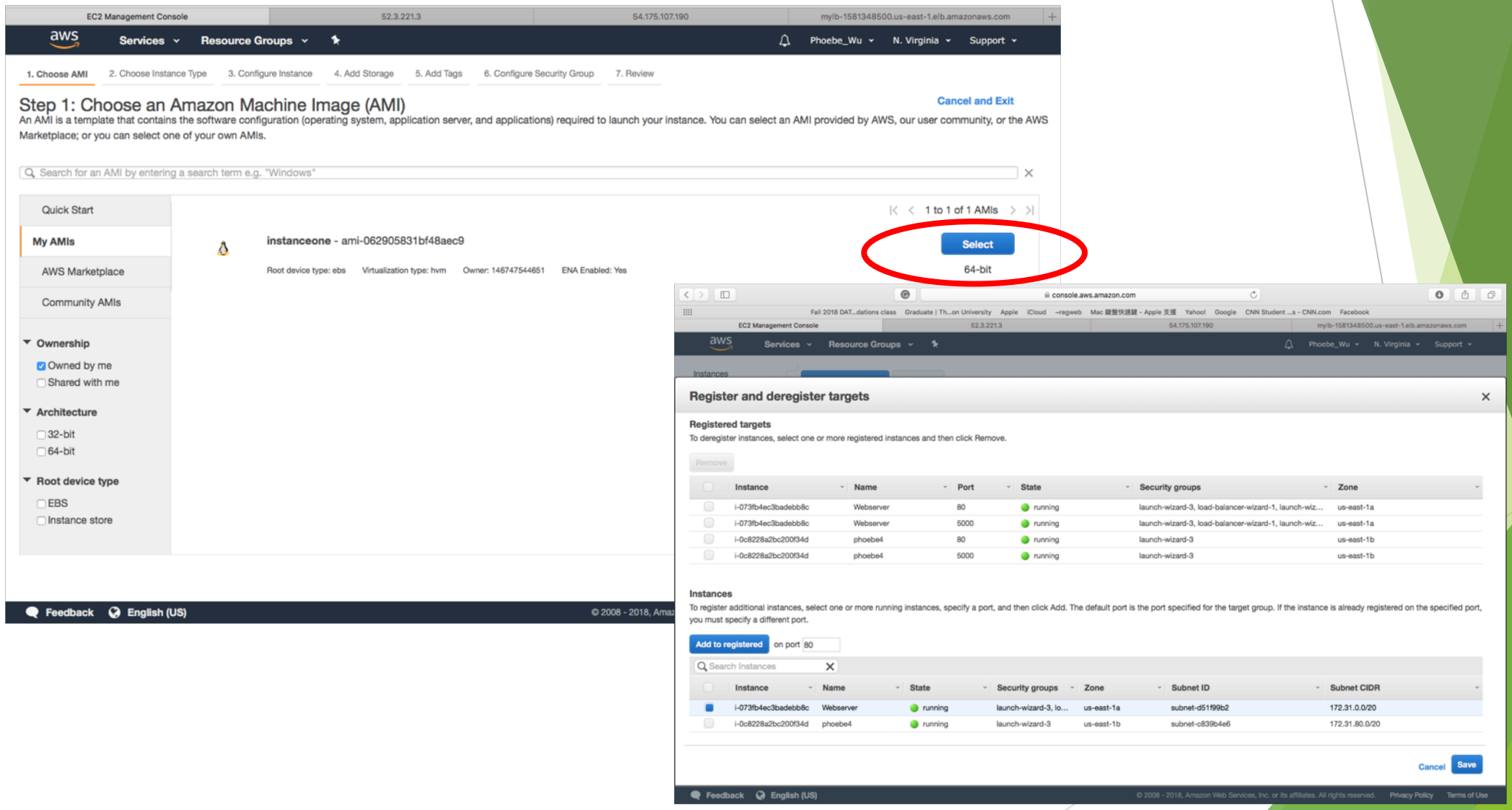
ELB details and web application



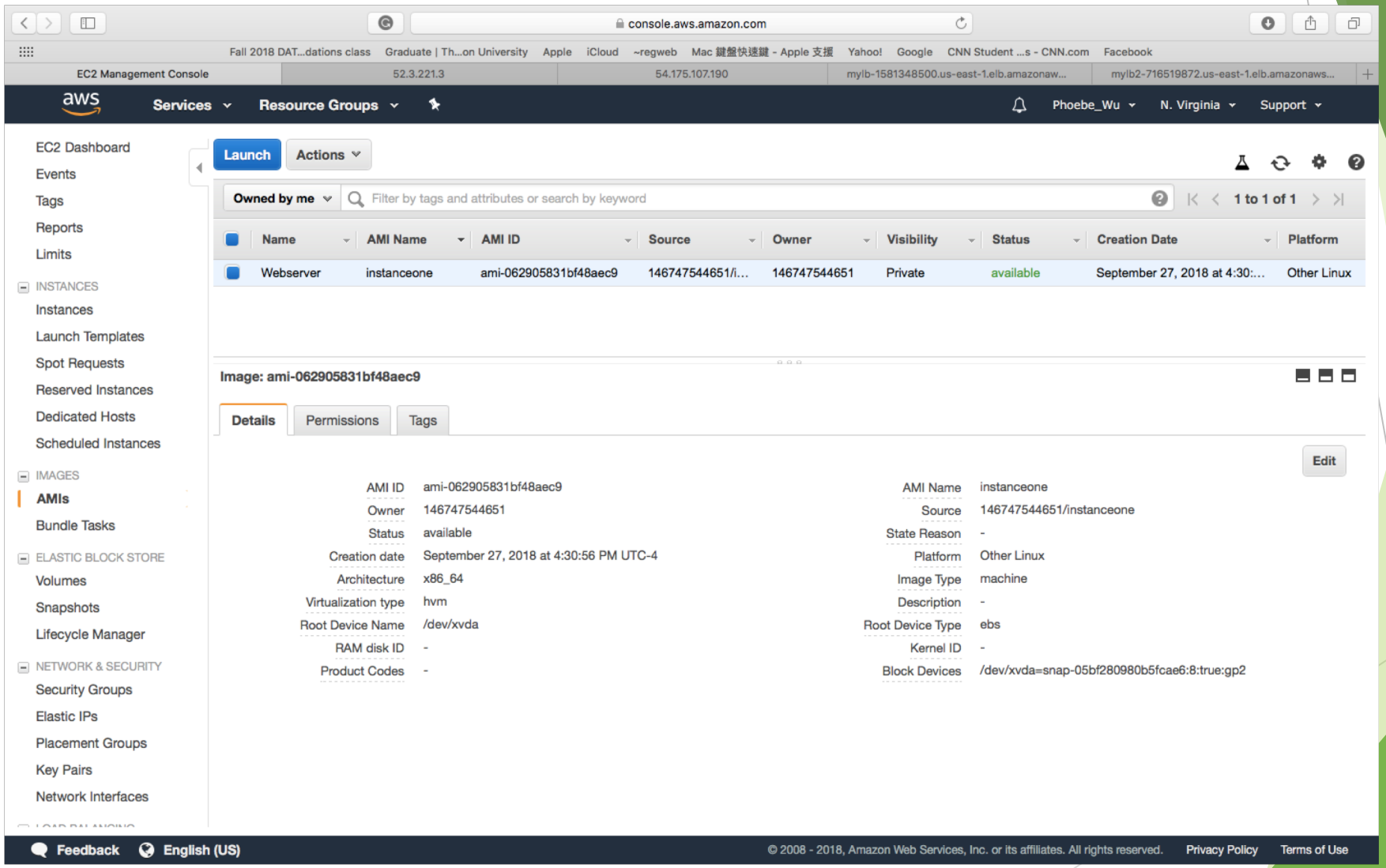
Exercise-1(My load balancer link):



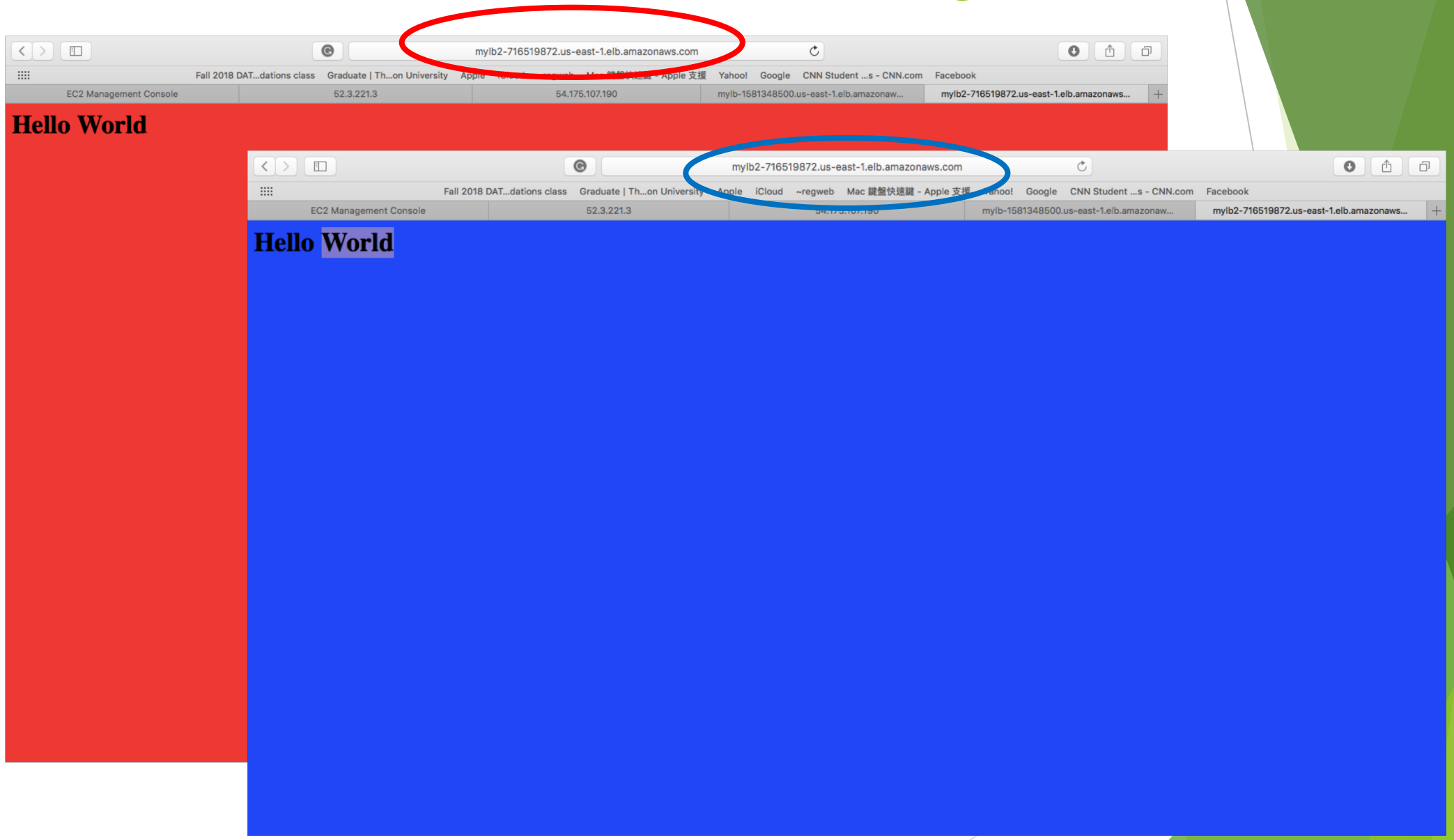
Create custom AMI



My AMI

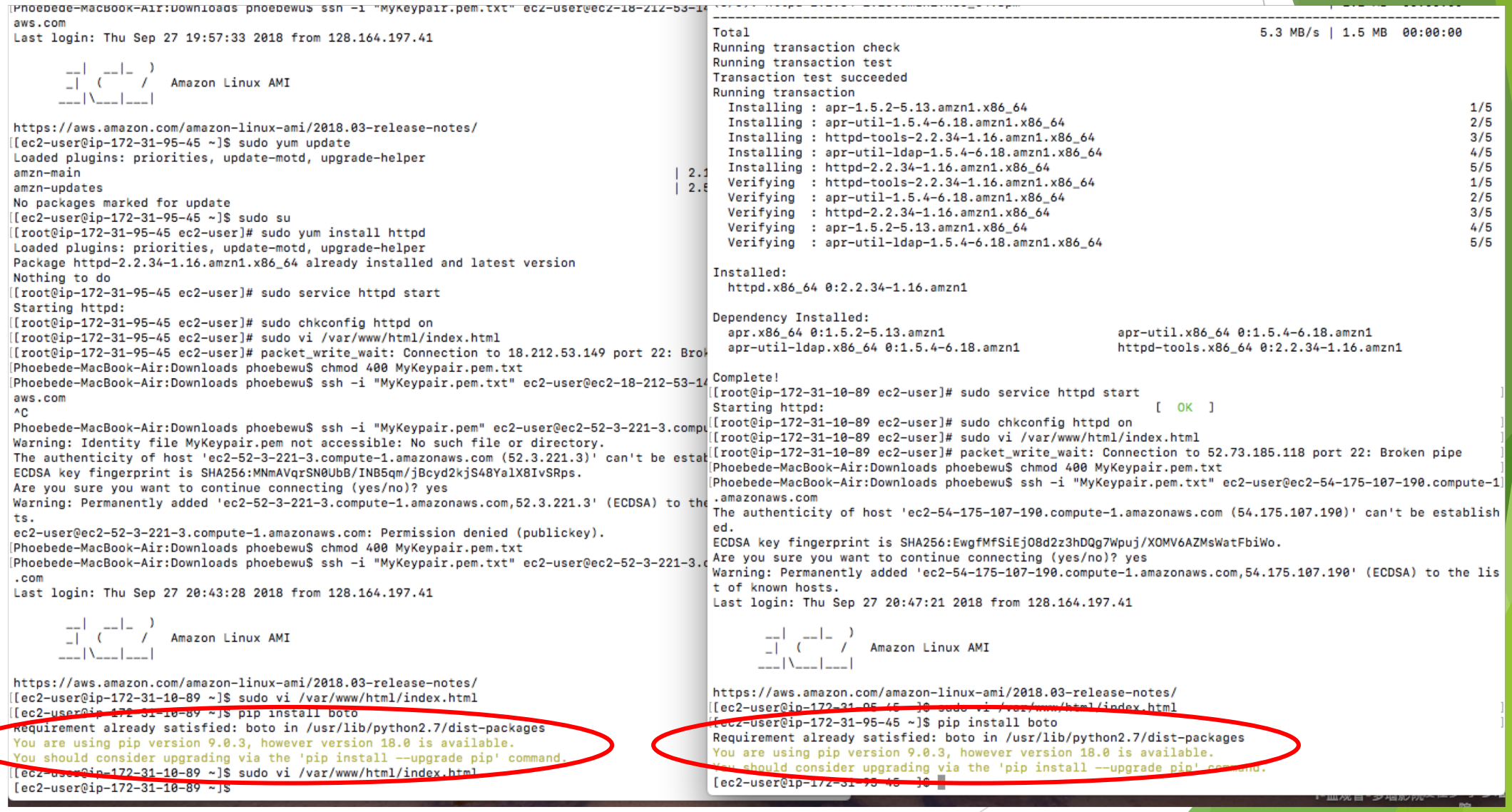

console.aws.amazon.com
EC2 Management Console
aws
Services
Resource Groups
Phoebe_Wu
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
Lifecycle Manager
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Launch
Actions
Owned by me
Filter by tags and attributes or search by keyword
1 to 1 of 1
Name
AMI Name
AMI ID
Source
Owner
Visibility
Status
Creation Date
Platform
Webserver
instanceone
ami-062905831bf48aec9
146747544651/i...
146747544651
Private
available
September 27, 2018 at 4:30:...
Other Linux
Image: ami-062905831bf48aec9
Details
Permissions
Tags
Edit
AMI ID ami-062905831bf48aec9
Owner 146747544651
Status available
Creation date September 27, 2018 at 4:30:56 PM UTC-4
Architecture x86_64
Virtualization type hvm
Root Device Name /dev/xvda
RAM disk ID -
Product Codes -
AMI Name instanceone
Source 146747544651/instanceone
State Reason -
Platform Other Linux
Image Type machine
Description -
Root Device Type ebs
Kernel ID -
Block Devices /dev/xvda=snap-05bf280980b5fcae6:8:true:gp2
Feedback
English (US)
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

New Load Balance after creating AMI

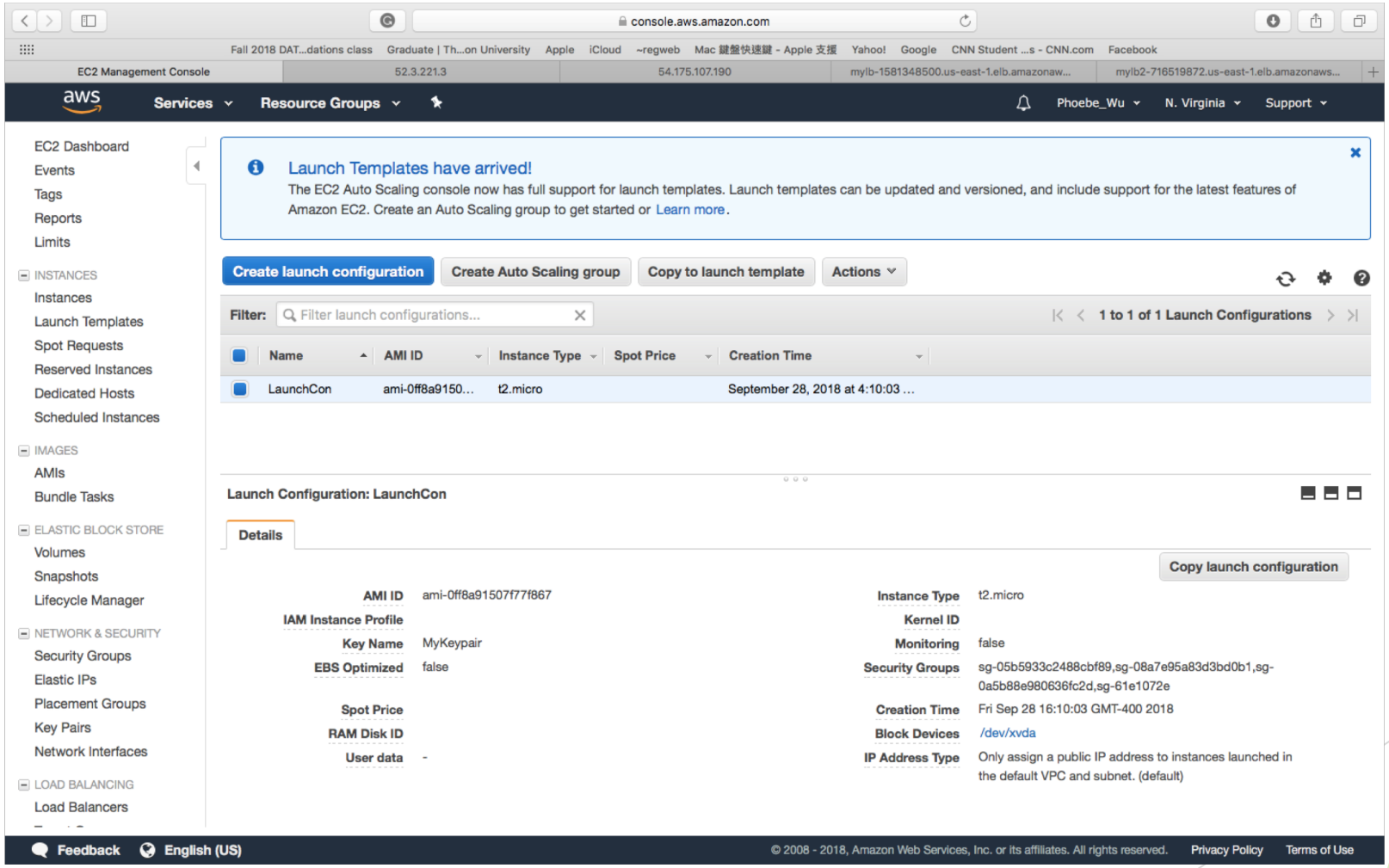


Part-III:
Auto Scaling

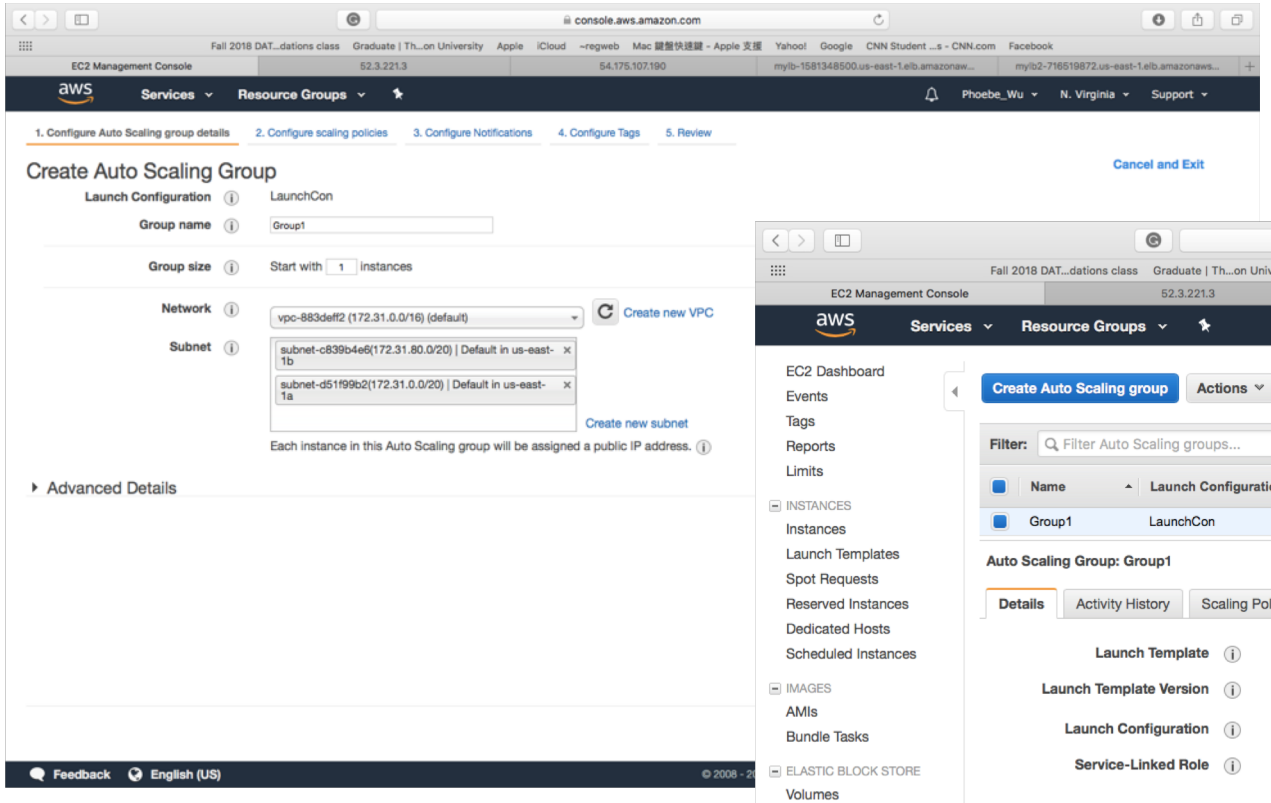
Python boto setup

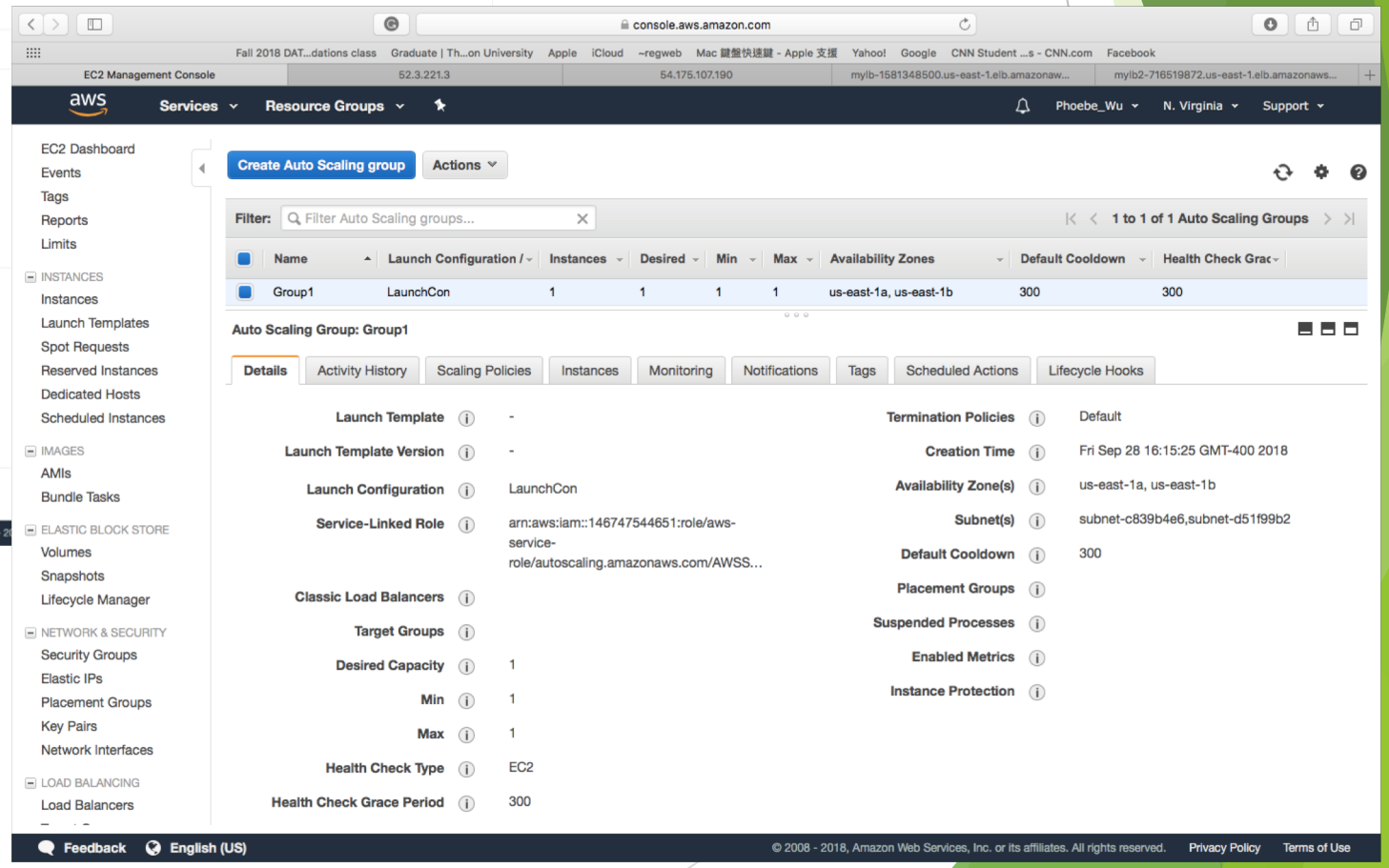


Create Launch Configuration

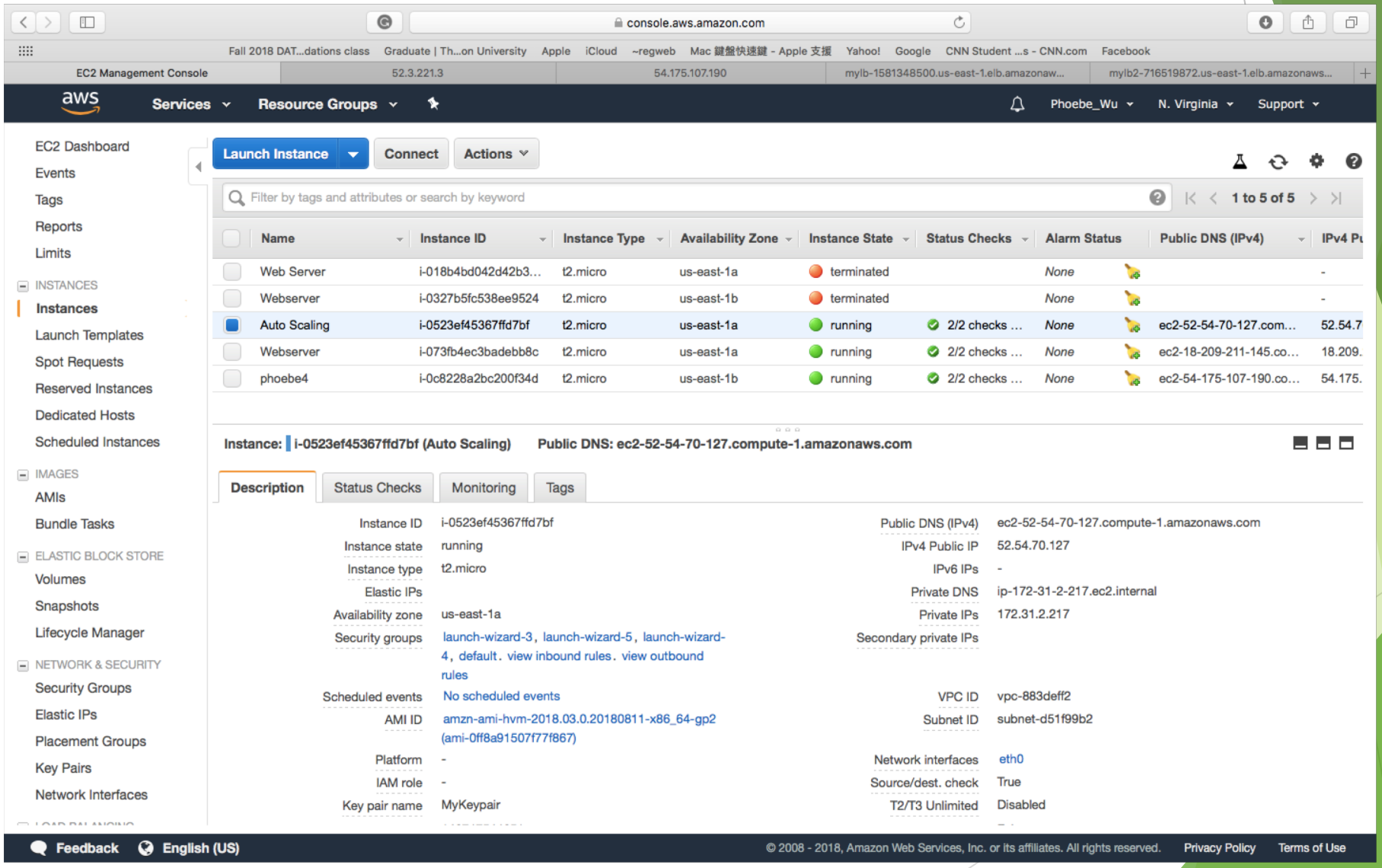


Create Auto Scaling group

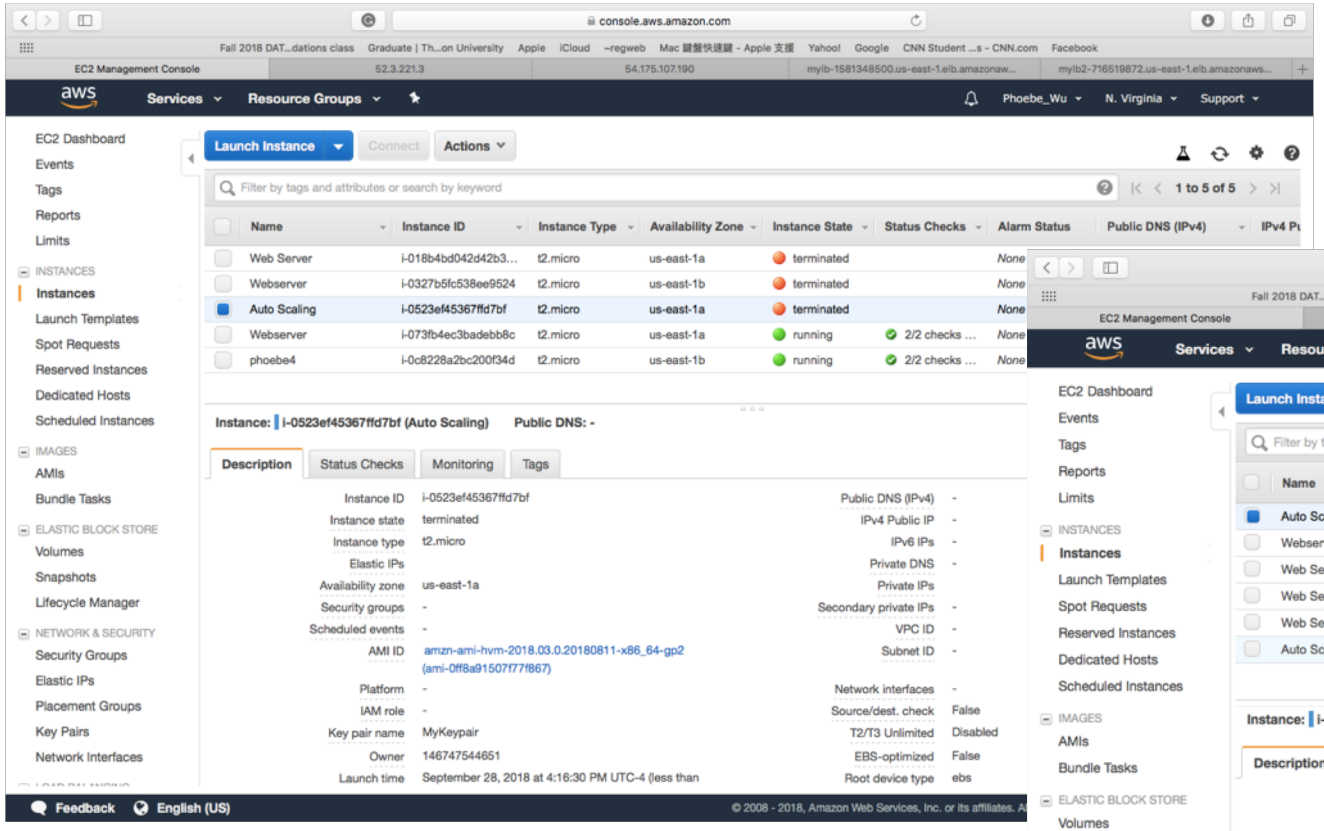


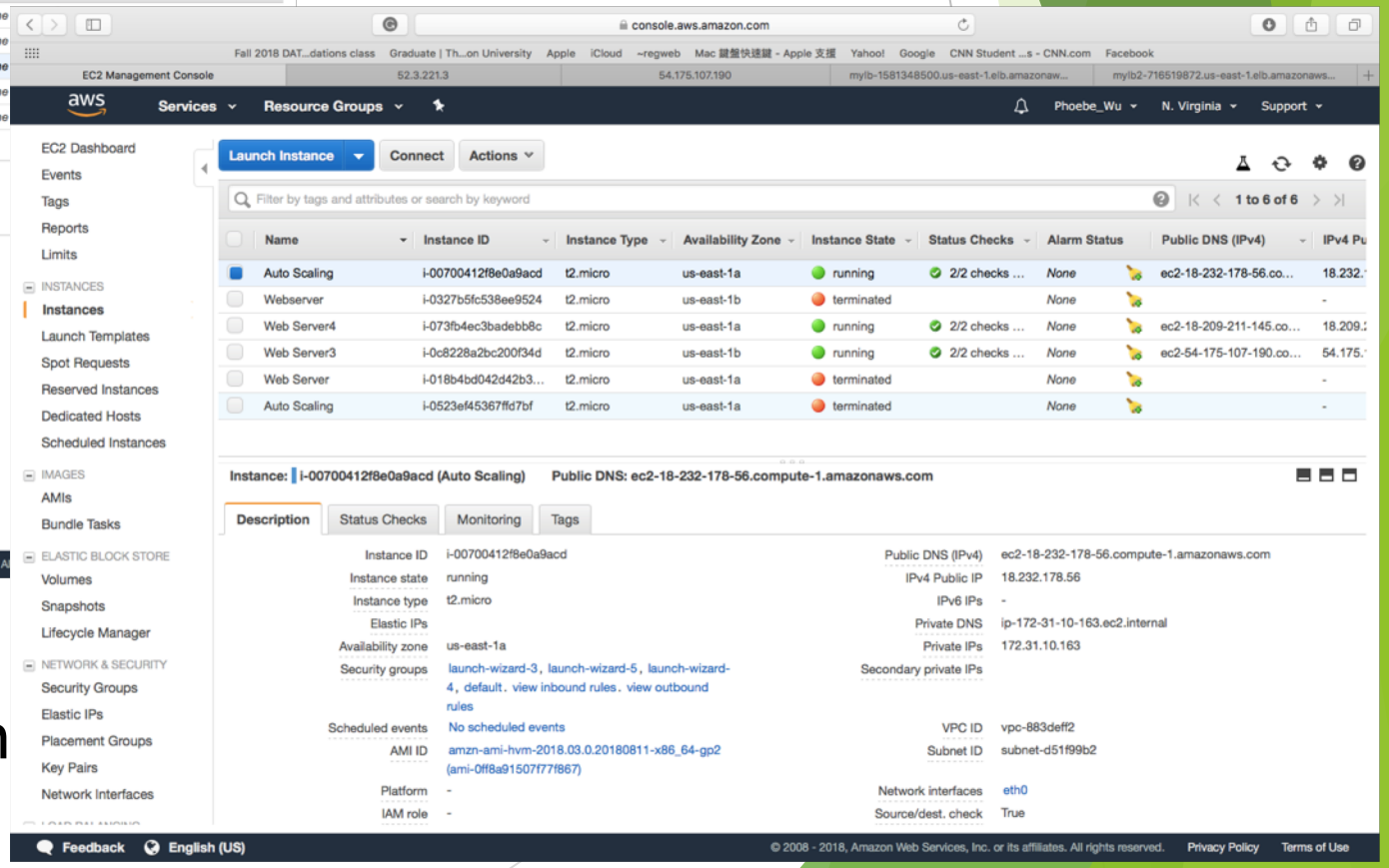


My Autoscaling Instance



Test Auto Scaling by Terminate the Auto Scaling Instance





The Auto Scaling Instance automatically provisioned within 5 minutes.

Auto Scaling Group History