DEPLOY ML MODEL ON CLOUD VM

Shilan Jin

WORKFLOW

1. Wrap your ML model into a flask application (tech: flask)
2. Call your web service from local computer
3. Use docker to containerize the flask application (tech: docker)
4. Host the docker container on an AWS ec2 instance and consume the web-service (tech: ec2)

STEP 1: WRAP ML MODEL INTO A FLASK WEB SERVICE

STEP 2: CALL YOUR WEBSERVER FROM LOCAL MACHINE

START YOUR WEBSERVER

Turn on your webserver by

- Going to Terminal, your work directory (where your app.py file is located), your development environment
- Input "python app.py"
- You should see

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
* Restarting with fsevents reloader
* Debugger is active!
* Debugger PIN: 136-446-349
```

TEST YOUR WEBSERVER (1/2)

Create a "request.py" to call your webserver

```
import requests
url = 'http://127.0.0.1:80/surprise_planner'

# User's inputs
r = requests.post(url,json={'prime-delay':1.80, 'objective':0.71,'method':'Shannon'})
print(r.json())
```

TEST YOUR WEBSERVER (2/2)

- Open another Terminal (production environment)
- Go to directory where "request.py" is located
- Input "python request.py"
- You should see on your production environment:

```
(base) celinejin@Celines-MacBook-Air AFRL % python request.py
{'experiment_suggested': 1.8, 'predicted_objective': 0.5824373302204151, 'search_method': 'exploitation',
'surprise': -0.3940799532472003, 'surprise_threshold': -0.5015761434299529}
```

- On your development environment

```
127.0.0.1 - - [12/Mar/2022 17:14:18] "POST /surprise_planner HTTP/1.1" 200 -
```

So far, your webserver is working locally!!

STEP 3: CONTAINERIZE YOUR MODEL

DOCKER

Install Docker:

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

CREATE A FOLDER FOR DOCKER CONTAINER

Example: \AFLR on my user root

All files needed should be stored under \AFLR.

Files include:

- A Dockerfile (details following)
- App.py
- Request.py
- Your ML model python files and data files

DOCKERFILE EXAMPLE

```
FROM python:3.8

RUN apt-get update -y

RUN apt-get install -y vim

RUN apt-get install -y gcc g++ gfortran subversion patch wget git
make

# Install python packages

WORKDIR /plan/

RUN pip install numpy

RUN pip install json5

RUN pip install matplotlib

RUN pip install scikit-learn

RUN pip install scipy

RUN pip install pyDOE

RUN pip install Flask

# Copy files

COPY ./* /plan/

EXPOSE 80

ENTRYPOINT ["python3", "app.py"]
```

INSTALL DOCKER AND BUILD

After you install the docker app, test your containerized model on your local computer within the terminal:

cd \AFRL (change to work directory)

docker build -t app-surprise . (build docker image)

docker run -p 80:80 app-surprise . (run docker image)

Your engine is running, when you see your terminal shows things like

```
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.3:80/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 131-732-918
```

TEST CONTAINERIZED MODEL

- Open another Terminal
- Go to the directory where "request.py" is located
- Input "python request.py"
- You should see

```
(base) celinejin@Celines-MacBook-Air AFRL % python request.py
{'experiment_suggested': 1.8, 'predicted_objective': 0.5858696829997289, 'search_method': 'exploitation',
'surprise': -0.4098200505324897, 'surprise_threshold': -0.5121842105145367}
```

Now, you can containerize your model.

STEP 4: HOLD YOUR MODEL ON CLOUD

AWS EC2

1. Create an AWS account
2. Create a key pair
3. Launch and edit an ec2 instance
4. Equip your ec2 instance with docker

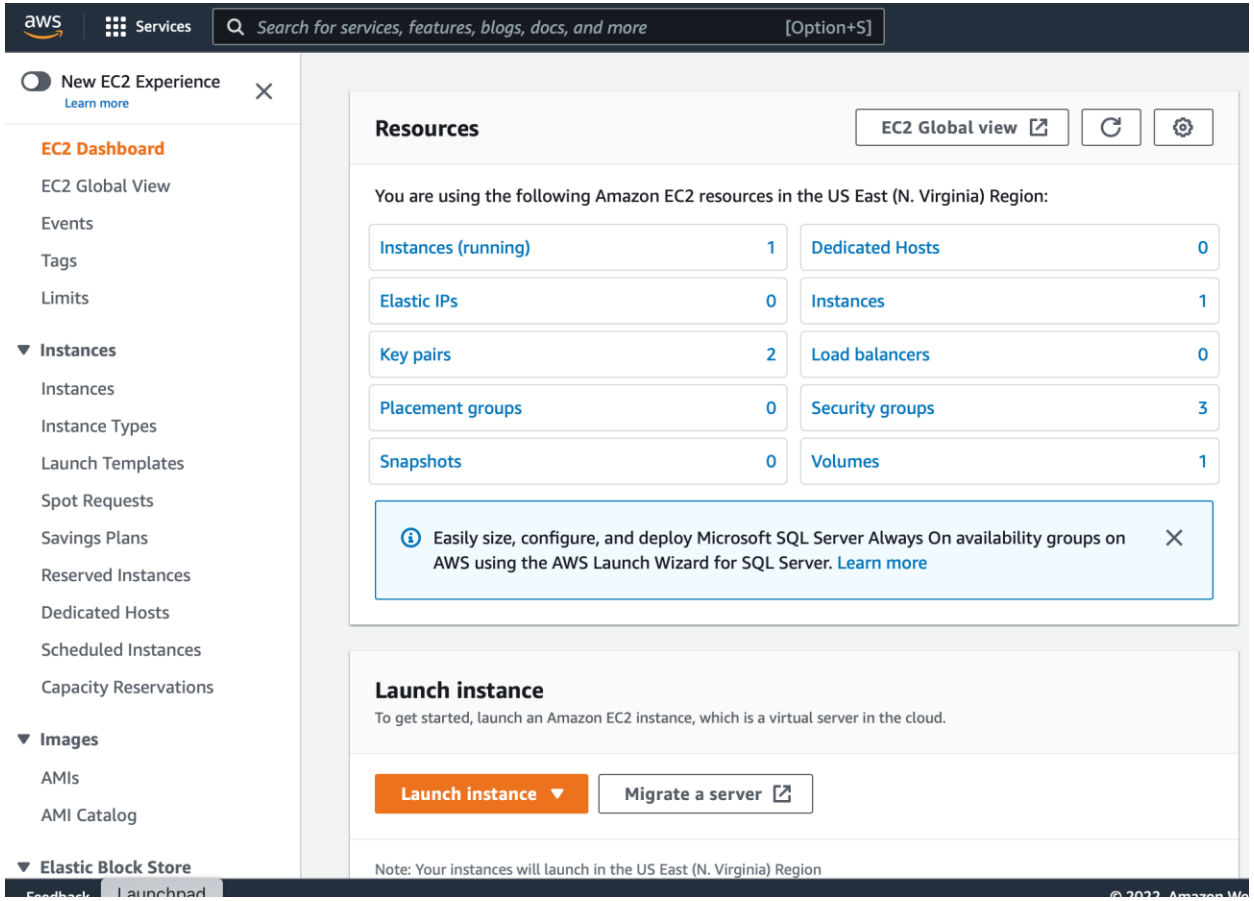
AWS EC2 ACCOUNT

- Go to

https://aws.amazon.com/ec2/

to sign up and sign in to the ec2 console.

- The ec2 dashboard looks like



CREATE A KEY PAIR

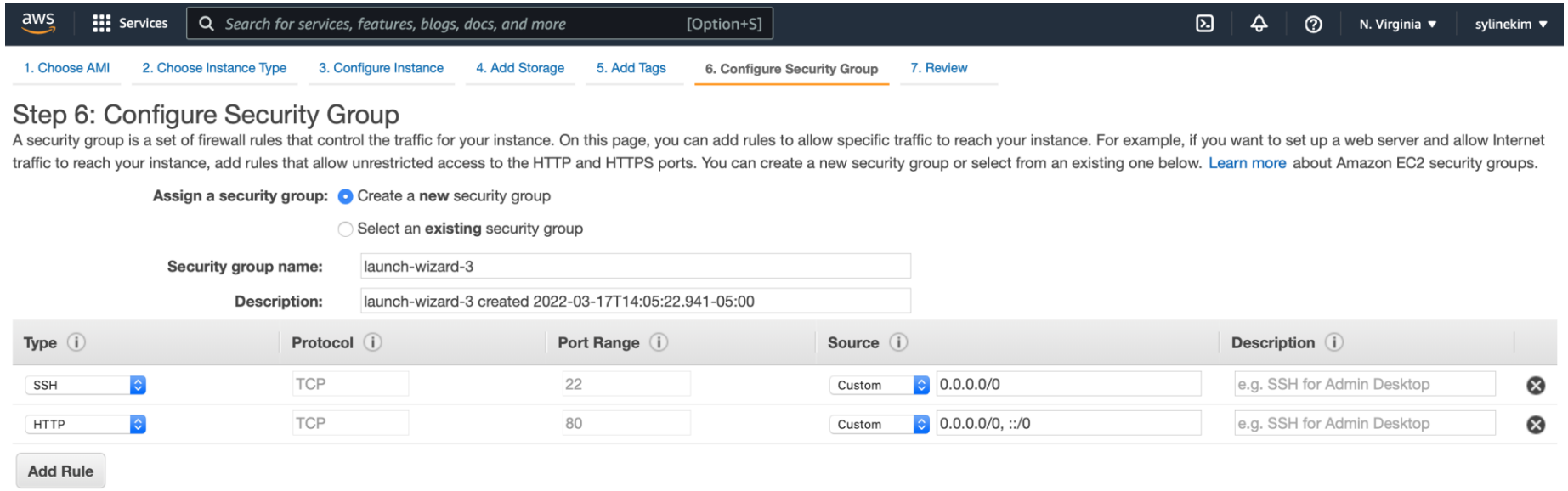
- On ec2 dashboard, click on "key pairs".
- Create a key pair. This will download a '.pem' file. Save it safely and remember the directory.
- Change the permission on your key pair file to private, by
  1. On terminal, navigate to the .pem file directory
  2. Input "chmod 400 [key-file-name].pem"

LAUNCH EC2 INSTANCE

- On ec2 dashboard, click "launch instance"
- Choose an Amazon Machine Image
  - ⑩ Choose the default one for free tier eligible (e.g., Amazon Linus 2 AMI (HVM), SSD Volume Type)
- Choose an Instance Type
  - Choose the default one for free tier eligible (i.e., t2.micro)
- Click "Review and Launch"

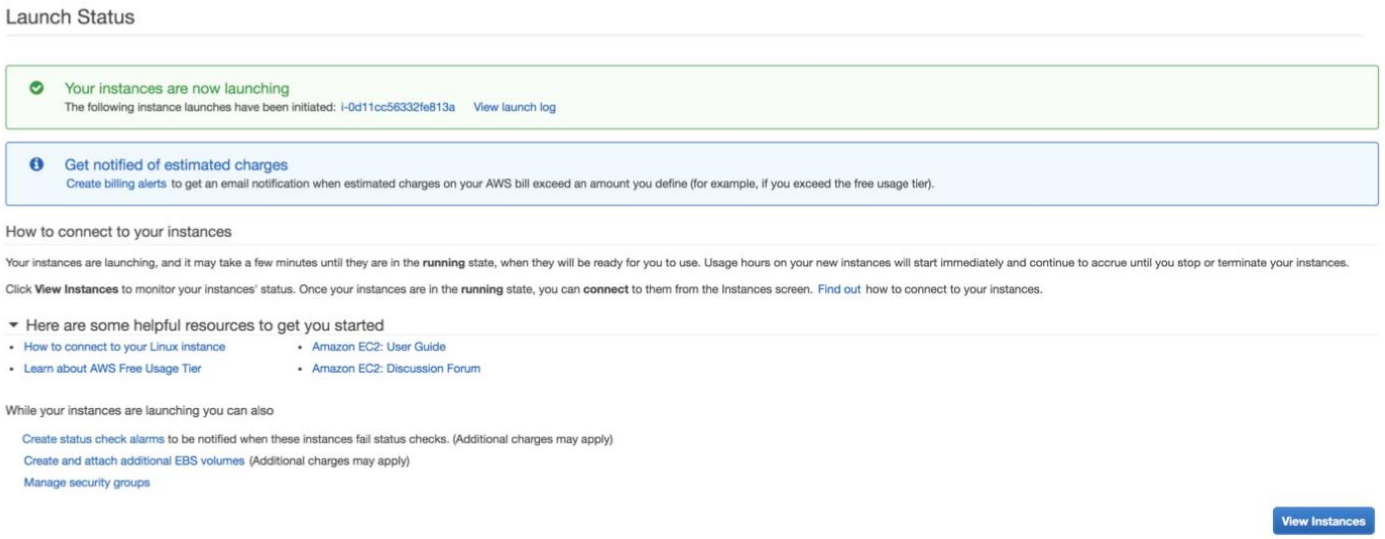
EDIT EC2 INSTANCE

- Edit Security Groups. "Add rule" to allow HTTP traffic on port 80



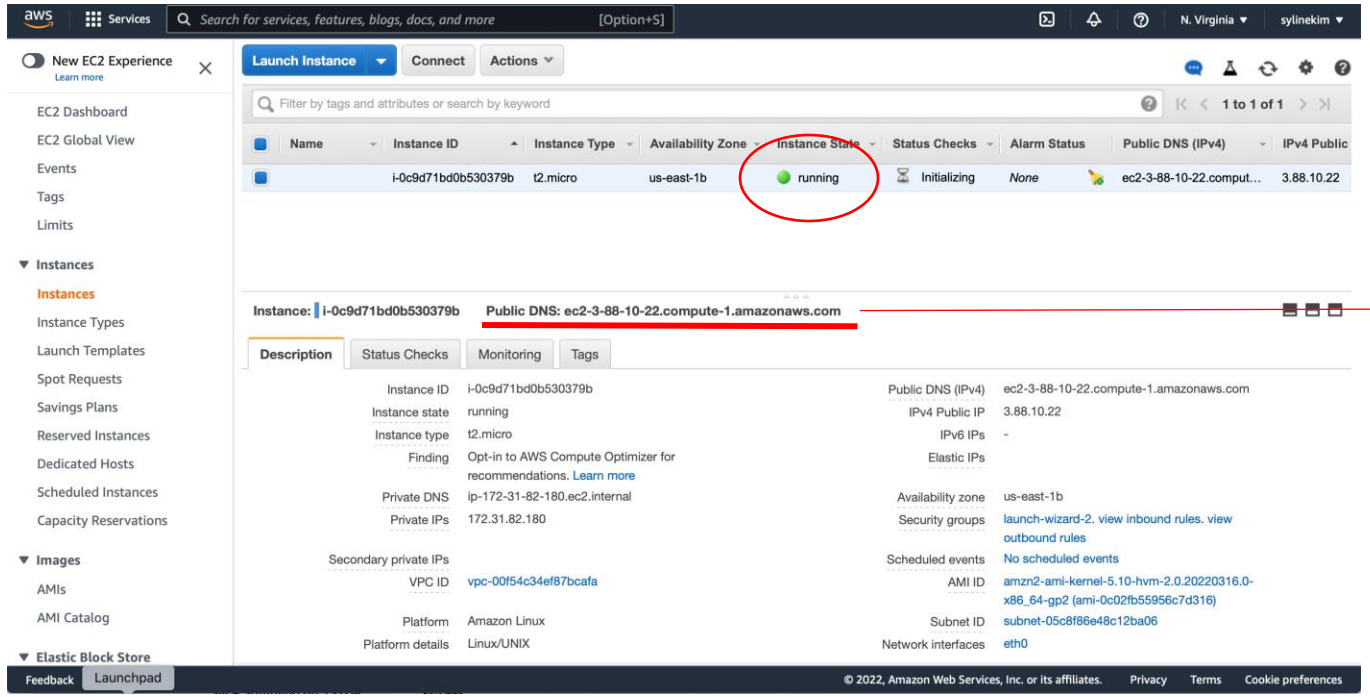
EDIT EC2 INSTANCE

- Click "Review and Launch" and "Launch"
- It prompts the request of choosing your downloaded key pair file
- Click "Launch Instances"



YOUR EC2 INFO

Now the ec2 should be "running".



The address will be used.

ACCESS EC2 INSTANCE

Open a terminal. Input:

```
ssh -i /[path]/[my-key-pair].pem ec2-user@[public-dns-name]
```

- [path]: the directory where you locate your key pair file .pem
- [my-key-pair]: the name of your key pair file
- [public-dns-name]: the public dns (the address info showed on last slide)

INSTALL DOCKER ON EC2

- Now, you are on the running ec2.
- Install docker on there, by inputting (do not need to change anything)

```
sudo amazon-linux-extras install docker

sudo yum install docker

sudo service docker start

sudo usermod –a –G docker ec2-user
```

REFRESH THE SETTING

- Log out of the ec2 instance using 'exit'
- Log back in again using

  'ssh -i /[path]/[my-key-pair].pem ec2-user@[public-dns-name]'
- Check if docker works using 'docker info'

COPY FILES TO DOCKER

- Open another terminal and input

'scp -i /[path]/[my-key-pair].pem [file-to-copy] ec2-user@[public-dns-name]:/home/ec2-user'

  - [file-to-copy] includes app.py, your model python files, and Docker file

- Go back to ec2 terminal and input 'ls' to check if all the files are copied there

RUN DOCKER

The same way to build and run docker on ec2:

```
docker build –t app-surprise .
docker run –p 80:80 app-surprise .
```

TEST EC2

- Input 'http://[public-dns-name]' on your internet browser and you should be able to see 'Hello World!'
- Open your python compiler or a terminal
- Navigate to the python working directory
- Run 'request.py' file