SWPP Practice Session #10

Refactoring + Deployment pt.1

TOC

1. Refactoring
    - Learn principles of refactoring
    - Hands-on coding example
    - (HW) Refactor your previous HW 3 or 4 code

2. Deploying pt.1
    - Setup AWS EC2 instance
    - Deploy your app with Docker container

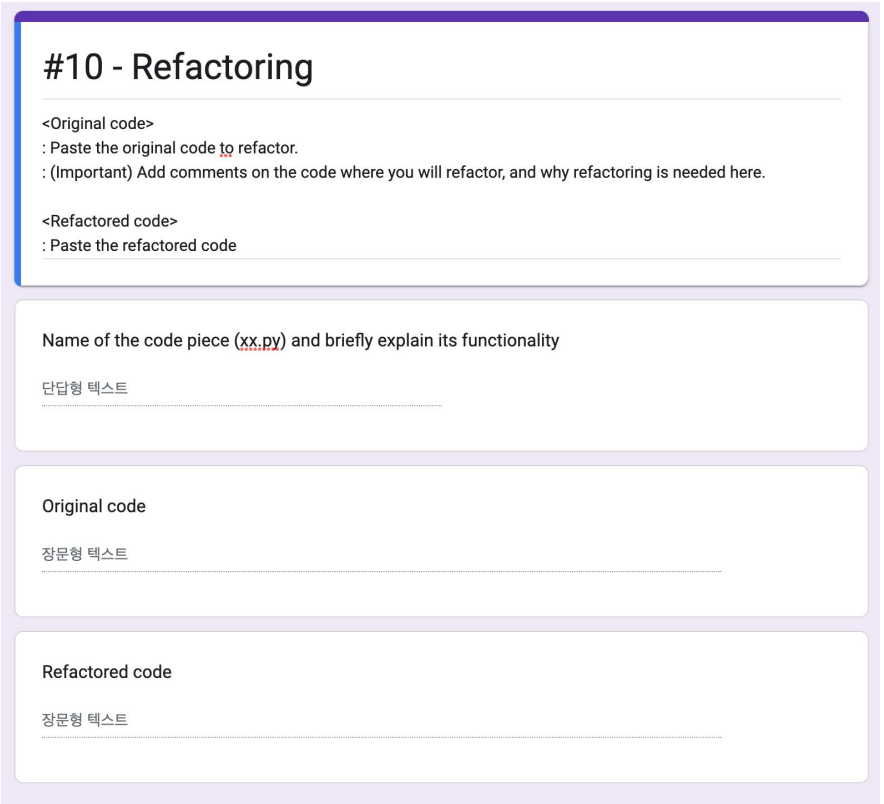
Code Refactoring

Prep: Fork the repository

https://github.com/swpp22fall-practice-sessions/swpp-p10-refactoring

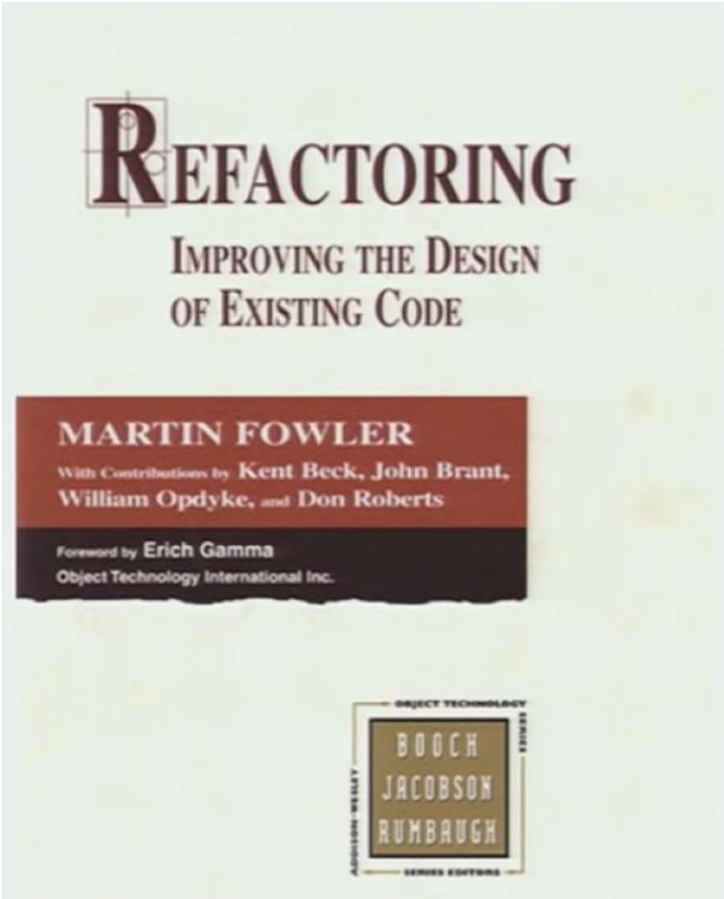
Announcement: Today's Task

- Q/A form
  - 1. Choose a piece of code example. We recommend you to pick a view function you wrote for HW4, or some React component you wrote for HW3. Any other code example is fine.
    2. Identify code smell and write briefly which solution you can apply.
    3. Refactor the code example.
  - link: https://forms.gle/CTcfYHgze4EdvBvx9
  - Due 11:59pm of 11/17 (Thur)



Goals of refactoring

- Readability
- Reusability
- Extendability
- Reduce possible causes of bugs & errors



Before dive into Refactoring

- Believe me, without automated tests you CAN’T do refactoring.
- Don’t do it too prematurely: After all, your primary job as programmer is to make your code work.

Symptoms or ‘Code Smell’

- Repetitive code
- Inappropriate naming
- Unnecessarily complicated code
- Overly large methods / classes
- Lack of useful comments

How to Solve Them? (Composing Function)

Name	Problem	Solution
Inline method	Code is unnecessarily extracted into too many methods.	Use one-liners when possible.
Extract method	Difficult to read, verbose. Code can be simplified	Turn the code performing a goal into a separate method.
Merge duplicated code	Hard to maintain and fix. Easy to make mistakes.	Merge them to a single place.
Replace tmp with query	difficult to keep track of the tmp variable.	replace tmp variables with method queries.
Substitute Algorithm	Code is overly complex.	Replace code with simpler, equivalent version.

Example:

refactor.py

swpp-p10-refactoring > refactor.py > ...

```python
# TODO: 1. Use one-liners
def count_asterisk_exceed_five(text):
    return text.count("*") > 5

if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    # TODO: 4. Consider general usage
    # TODO: 3. Substitute algorithm

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word1` from `text`
    tmp = text
    while word1 in tmp:
        tmp = tmp[:tmp.find(word1)] + "*" * len(word1) + tmp[tmp.find(word1) + len(word1):]
    text = tmp

    if count_asterisk_exceed_five(text):
        print("More than five *")

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word2` from `text`
    tmp = text
    while word2 in tmp:
        tmp = tmp[:tmp.find(word2)] + "*" * len(word2) + tmp[tmp.find(word2) + len(word2):]
    text = tmp

    if count_asterisk_exceed_five(text):
        print("More than five *")

    # Print result `text`
    print(text)
```

Example: Inline method

Example:
Inline method

sol1.py

```
if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    # TODO: 4. Consider general usage
    # TODO: 3. Substitute algorithm

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word1` from `text`
    tmp = text
    while word1 in tmp:
        tmp = tmp[:tmp.find(word1)] + "*" * len(word1) + tmp[tmp.find(word1) + len(word1):]
    text = tmp

    if text.count("*") > 5:
        print("More than five *")

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word2` from `text`
    tmp = text
    while word2 in tmp:
        tmp = tmp[:tmp.find(word2)] + "*" * len(word2) + tmp[tmp.find(word2) + len(word2):]
    text = tmp

    if text.count("*") > 5:
        print("More than five *")

    # Print result `text`
    print(text)
```

Example: Extract method, remove duplicated code

Example: Extract method, remove duplicated code

```
def censor(text, word):
    tmp = text
    while word in tmp:
        tmp = tmp[:tmp.find(word)] + "*" * len(word) + tmp[tmp.find(word) + len(word):]
    return tmp

if __name__ == "__main__":
    text = """ Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not our hero.
He is a silent guardian, a watchful protector... a dark knight. """
    word = "hero"
    word2 = "silent"

    text = censor(text, word)

    if text.count("*") > 5:
        print("More than five *")

    # censor the word2 from the text
    text = censor(text, word2)

    if text.count("*") > 5:
        print("More than five *")

    print(text)
```

Example:
Extract method, remove duplicated code

sol2.py

```
def censor(text, word):
    return text.replace(word, "*" * len(word))

def alert_text(text):
    if text.count("*") > 5:
        print("More than five *")

if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    # TODO: 4. Consider general usage
    # TODO: 3. Substitute algorithm

    text = censor(text, word1)

    alert_text(text)

    text = censor(text, word2)

    alert_text(text)

    # Print result `text`
    print(text)
```

Example: Get rid of the unnecessary tmp

Example: Get rid of the unnecessary tmp

```
def censor(text, word):
    while word in text:
        text = text[:text.find(word)] + "*" * len(word) + text[text.find(word) + len(word):]
    return text

def alert_text(text):
    if text.count("*") > 5:
        print("More than five *")

if __name__ == "__main__":
    text = """ Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not our hero.
He is a silent guardian, a watchful protector... a dark knight. """
    word = "hero"
    word2 = "silent"

    text = censor(text, word)

    alert_text(text)
    # censor the word2 from the text
    text = censor(text, word2)

    alert_text(text)

    print(text)
```

[Reminder]

Original Code

swpp-p10-refactoring > refactor.py > ...

```python
# TODO: 1. Use one-liners
def count_asterisk_exceed_five(text):
    return text.count("*") > 5

if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    # TODO: 4. Consider general usage
    # TODO: 3. Substitute algorithm

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word1` from `text`
    tmp = text
    while word1 in tmp:
        tmp = tmp[:tmp.find(word1)] + "*" * len(word1) + tmp[tmp.find(word1) + len(word1):]
    text = tmp

    if count_asterisk_exceed_five(text):
        print("More than five *")

    # TODO: 2. Extract method, remove duplicated code
    # Censor `word2` from `text`
    tmp = text
    while word2 in tmp:
        tmp = tmp[:tmp.find(word2)] + "*" * len(word2) + tmp[tmp.find(word2) + len(word2):]
    text = tmp

    if count_asterisk_exceed_five(text):
        print("More than five *")

    # Print result `text`
    print(text)
```

Example: Substitute algorithm

Example: Substitute algorithm

```
def censor(text, word):
    return text.replace(word, "*" * len(word))

def alert_text(text):
    if text.count("*") > 5:
        print("More than five *")
if __name__ == "__main__":
    text = """ Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not our hero.
He is a silent guardian, a watchful protector... a dark knight. """
    word = "hero"
    word2 = "silent"
    text = censor(text, word)

    alert_text(text)
    # censor the word2 from the text
    text = censor(text, word2)

    alert_text(text)

    print(text)
```

Example: Extract method, remove duplicated code

```
def censor(text, word):
    return text.replace(word, "*" * len(word))

def alert_text(text):
    if text.count("*") > 5:
        print("More than five *")

if __name__ == "__main__":
    text = """ Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not our hero.
He is a silent guardian, a watchful protector... a dark knight. """
    word = "hero"
    word2 = "silent"
    text = censor(text, word)

    alert_text(text)
    # censor the word2 from the text
    text = censor(text, word2)

    alert_text(text)

    print(text)
```

Example: Extract method, remove duplicated code

```python
def censor_and_alert(text, *words):
    for word in words:
        text = text.replace(word, "*" * len(word))

        if text.count("*") > 5:
            print("More than five *")

    return text

if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    # TODO: 4. Consider general usage

    text = censor_and_alert(text, word1, word2)

    # Print result `text`
    print(text)
```

Optional: Consider General Usage

```
def censor_and_alert(text, *words):
    for word in words:
        text = text.replace(word, "*" * len(word))
        if text.count("*") > 5:
            print("More than five *")
    return text

if __name__ == "__main__":
    text = """ Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not our hero.
He is a silent guardian, a watchful protector... a dark knight. """
    word = "hero"
    word2 = "silent"

    text = censor_and_alert(text, word, word2)

    print(text)
```

Example: Complete code

```
def censor_and_alert(text, *words, replace_pattern="*", alert_threshold=5):
    for word in words:
        text = text.replace(word, replace_pattern * len(word))

        if text.count("*") > alert_threshold:
            print(f"More than {alert_threshold} {replace_pattern}")

    return text

if __name__ == "__main__":
    text = """Because he's the hero Gotham deserves but not the one it needs right now.
So we will hunt him, because he can take it. Because he's not out hero.
He is a silent guardian, a watchful protector... a dark knight."""

    word1 = "hero"
    word2 = "silent"

    text = censor_and_alert(text, word1, word2)

    # Print result `text`
    print(text)
```

Real World Case Study (Django)

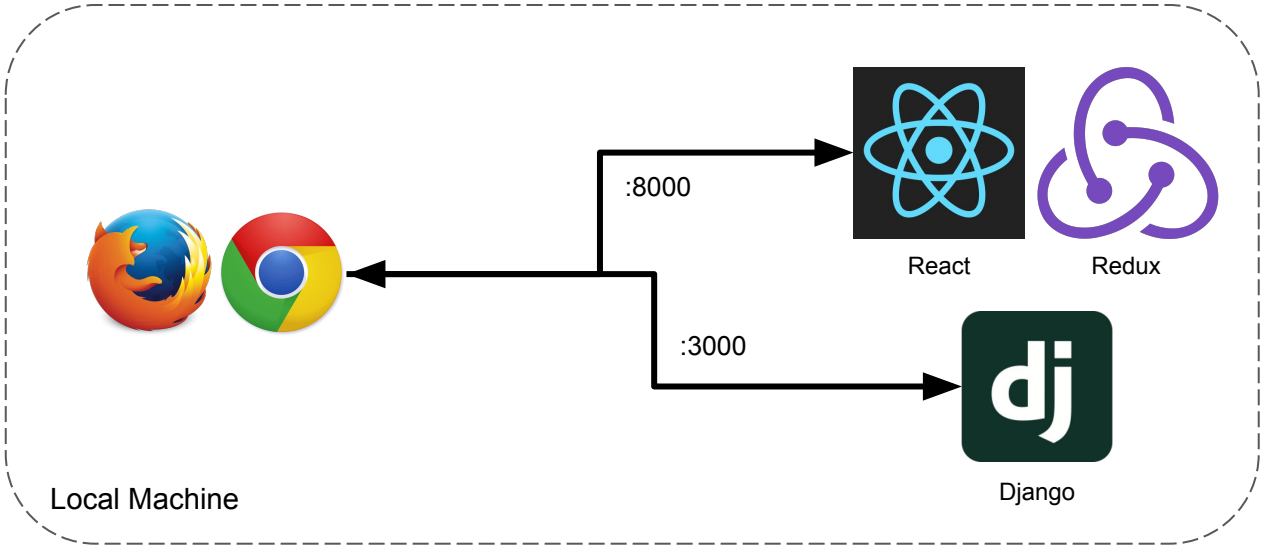
- We may have many view methods which need user authentication (reuse, reuse, reuse)
- Actually, it is much more related to design pattern, but we already mastered decorator :)

```
from functools import wraps

def check_logged_in(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if args and args[0].user.is_authenticated:
            return func(*args, **kwargs)
        return HttpResponse(status=401)
    return wrapper

@check_logged_in
def some_view_methods(request, ...):
    pass
```

Deployment pt.1

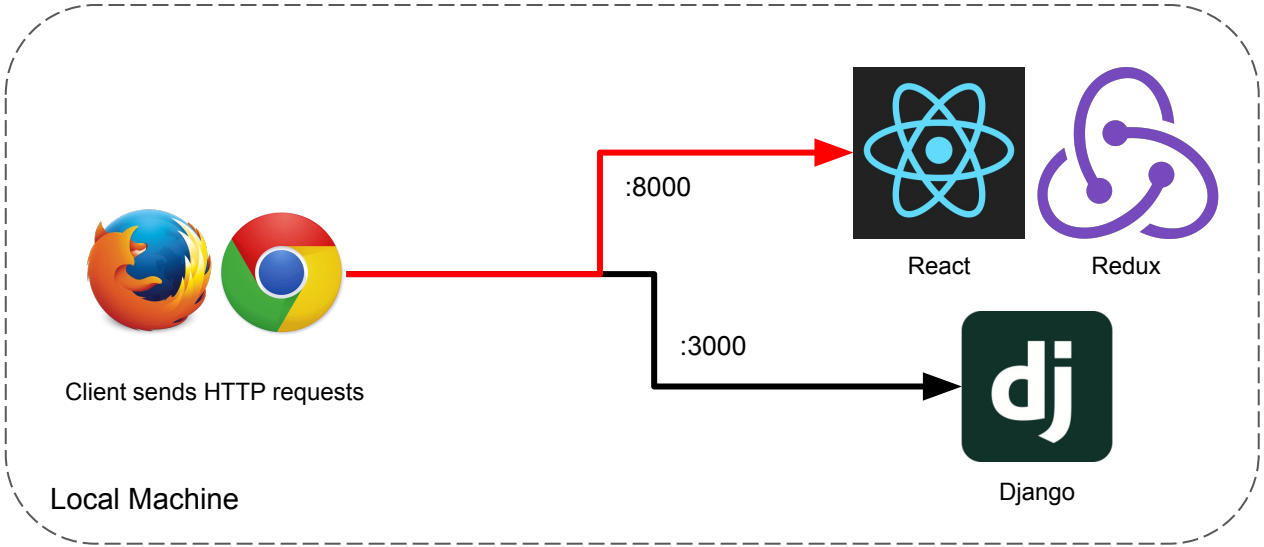
Until Now...

- You have deployed your web service on your local machine.
  - Frontend/Backend applications are hosted on the local machine.
  - The browser also runs on the local machine.



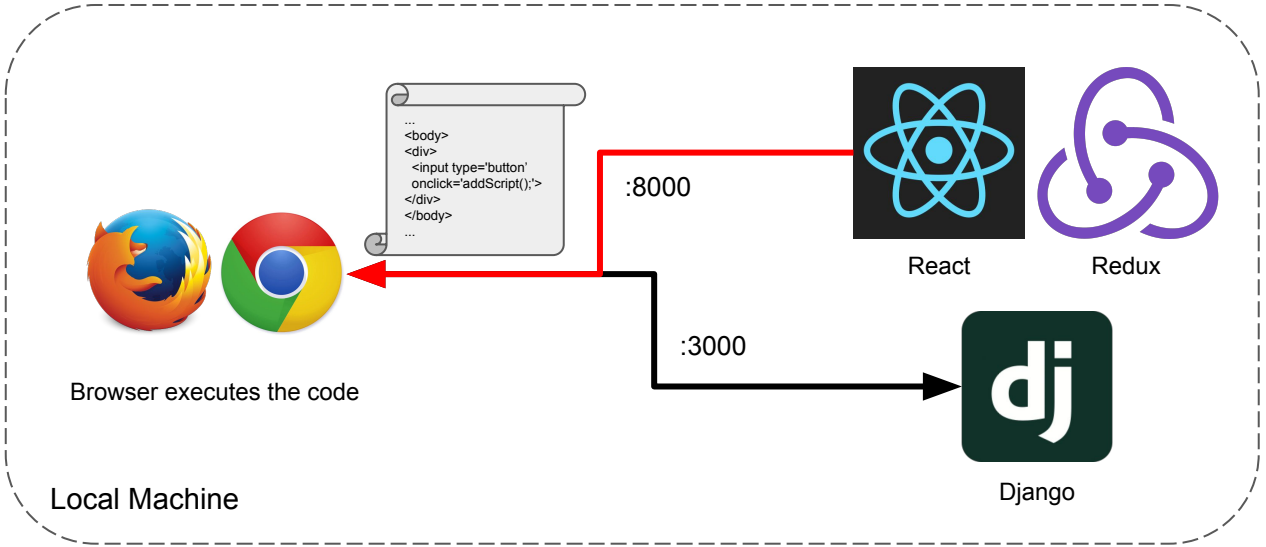
Until Now...

- You have deployed your web service on your local machine.
  - The browser sends requests and receives responses from the applications.



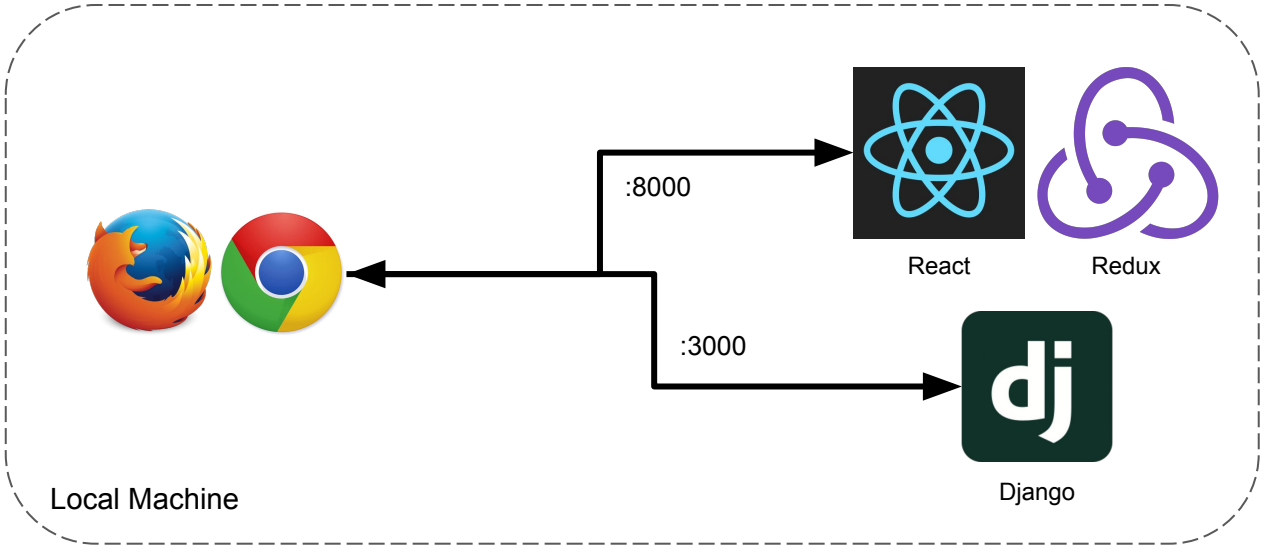
Until Now...

- You have deployed your web service on your local machine.
  - The browser sends requests and receives responses from the applications.
  - The retrieved codes are executed on the browser side.



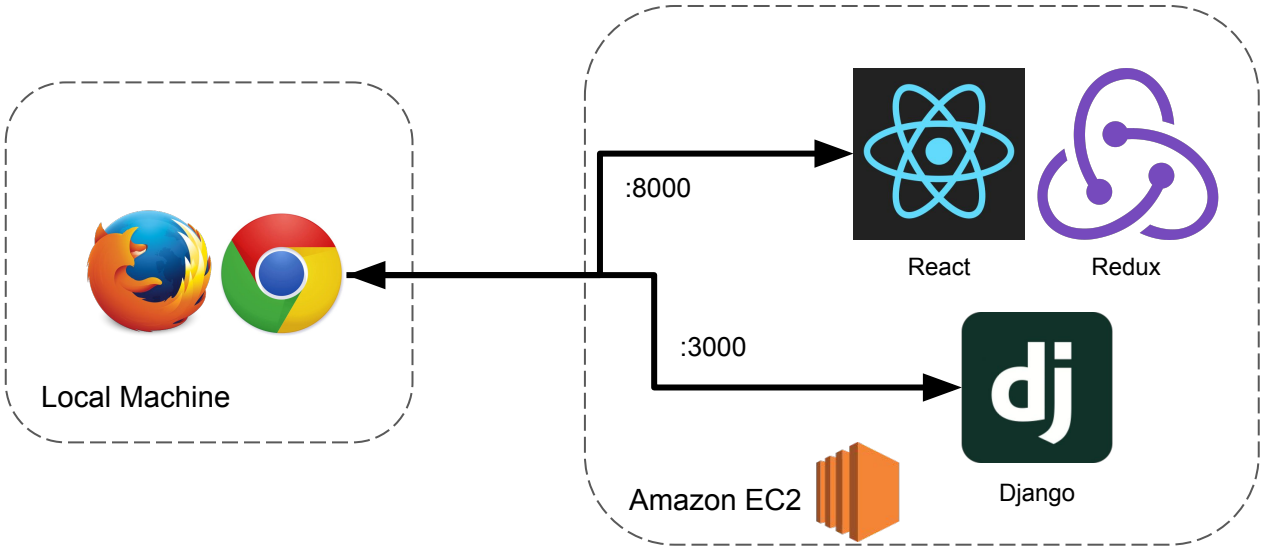
Until Now...

- But you cannot keep hosting your web service on your local machine.
  - We need stable HW/SW environment to host the web service.
  - We need to support secure communications between clients and servers, when the web services are hosted in remote servers.



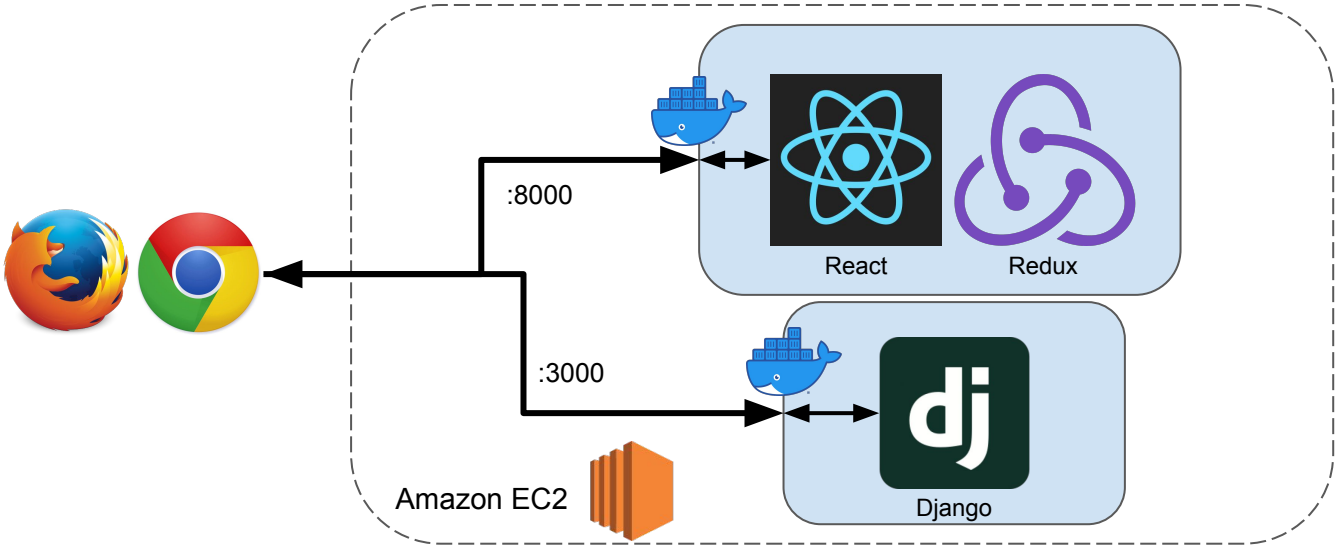
Objectives

1. Deploy the web service in the cloud environment (Amazon EC2).



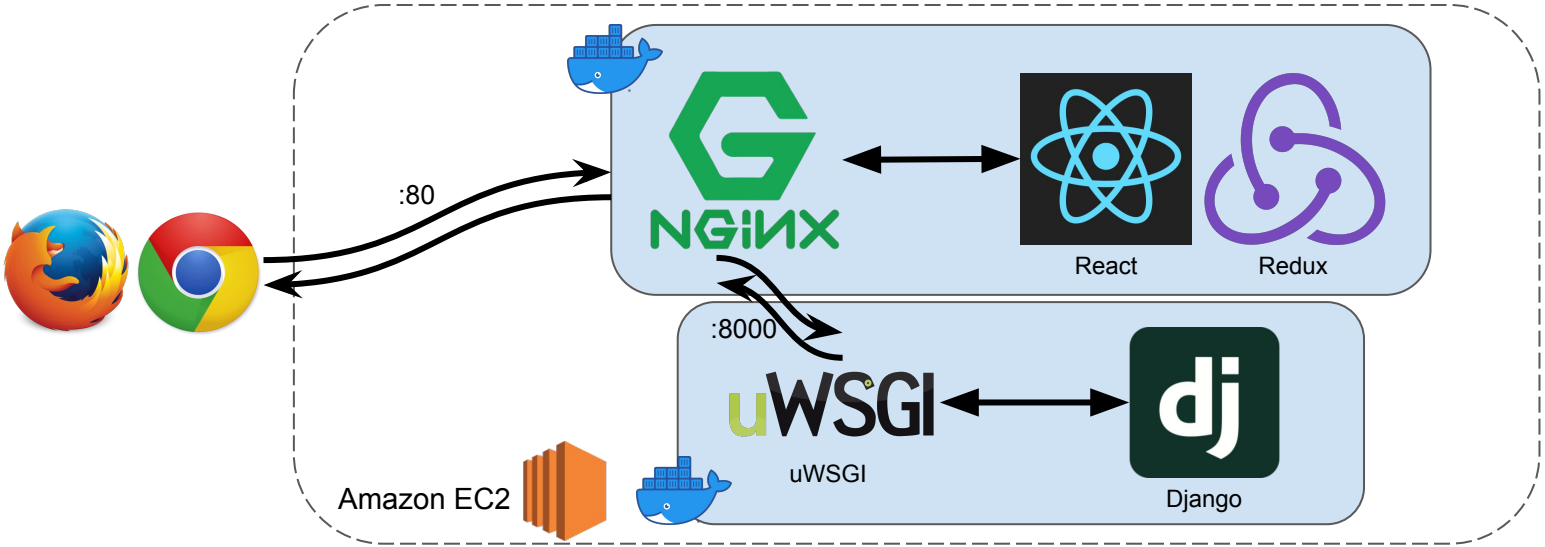
Objectives

1. Deploy the web service in the cloud environment (Amazon EC2).
2. Run your applications in Docker containers.
   a. A container provides an isolated software environment.
   b. You can save not only your codes but also your environments (e.g., python packages).
   c. Less likely to suffer from environment errors and easy to recover from any faults.



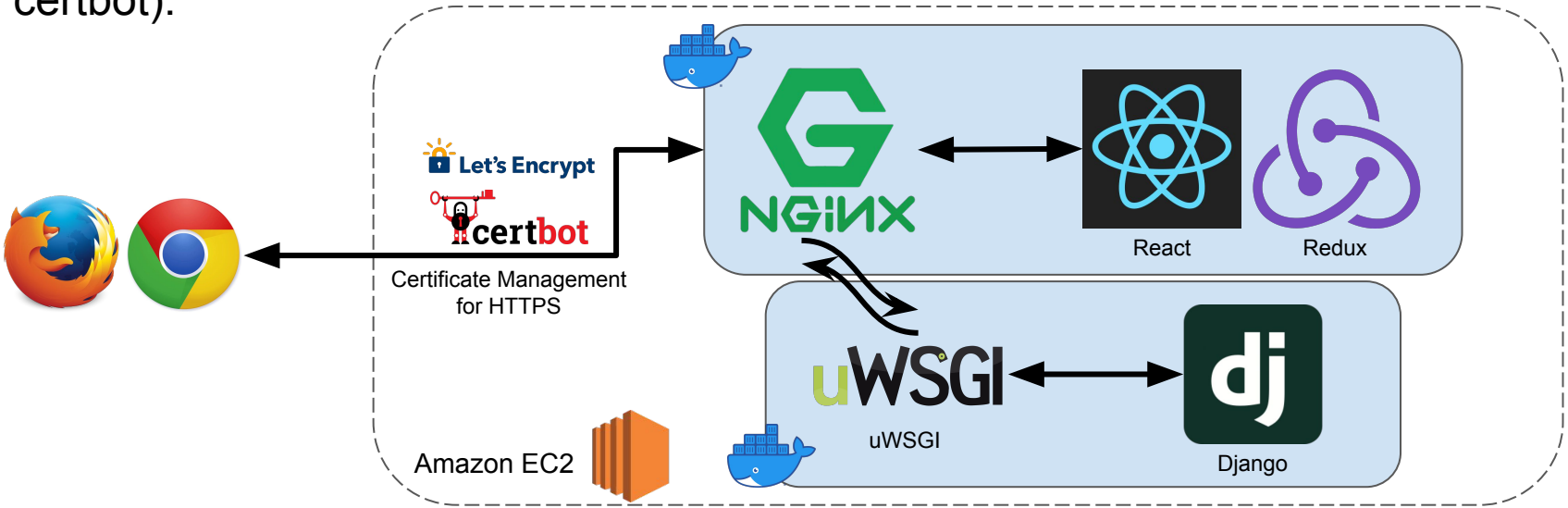
Objectives

1. Deploy the web service in the cloud environment (Amazon EC2).
2. Run your applications in Docker containers.
3. Setup a web server (Nginx, uWSGI).
   a. Nginx to run a web server.
   b. uWSGI to connect the web server and django application.



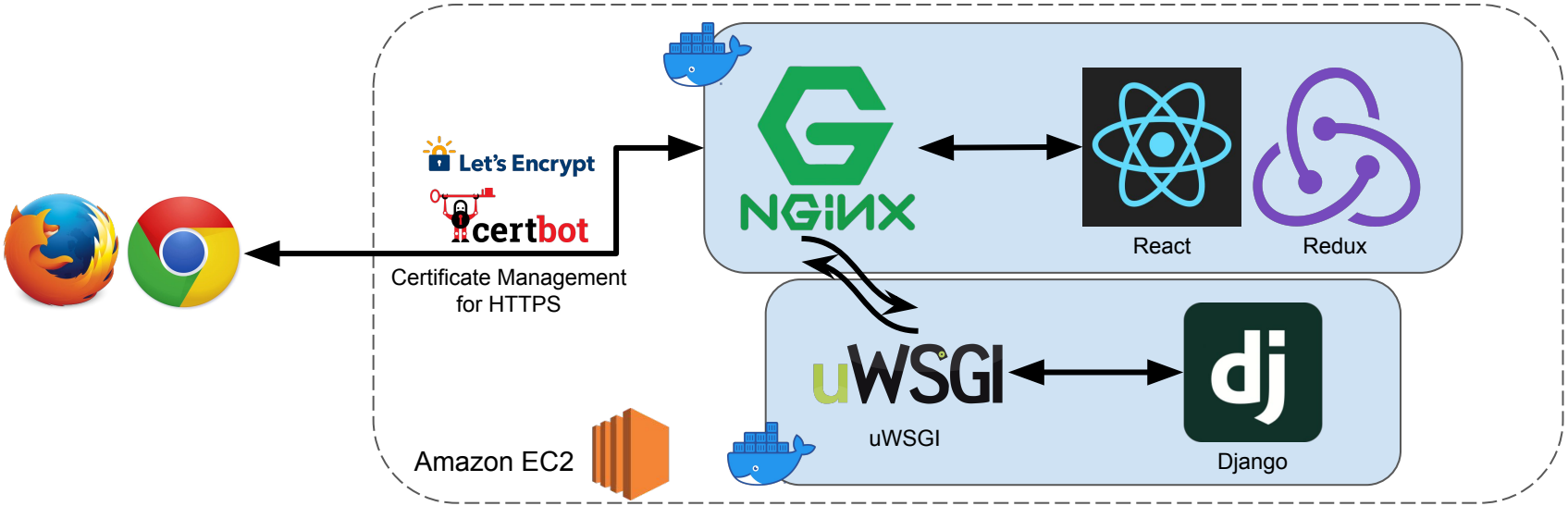
Objectives

1. Deploy the web service in the cloud environment (Amazon EC2).
2. Run your applications in Docker containers.
3. Setup a web server (Nginx, uWSGI).
4. Support HTTPS protocol with Certificate Management Tools (Let's Encrypt, certbot).



Today's Contents

1. AWS EC2
2. Docker
3. Web server setup (NginX, uWSGI)
4. HTTPS (SSL) & Certificate Management Tools



AWS EC2

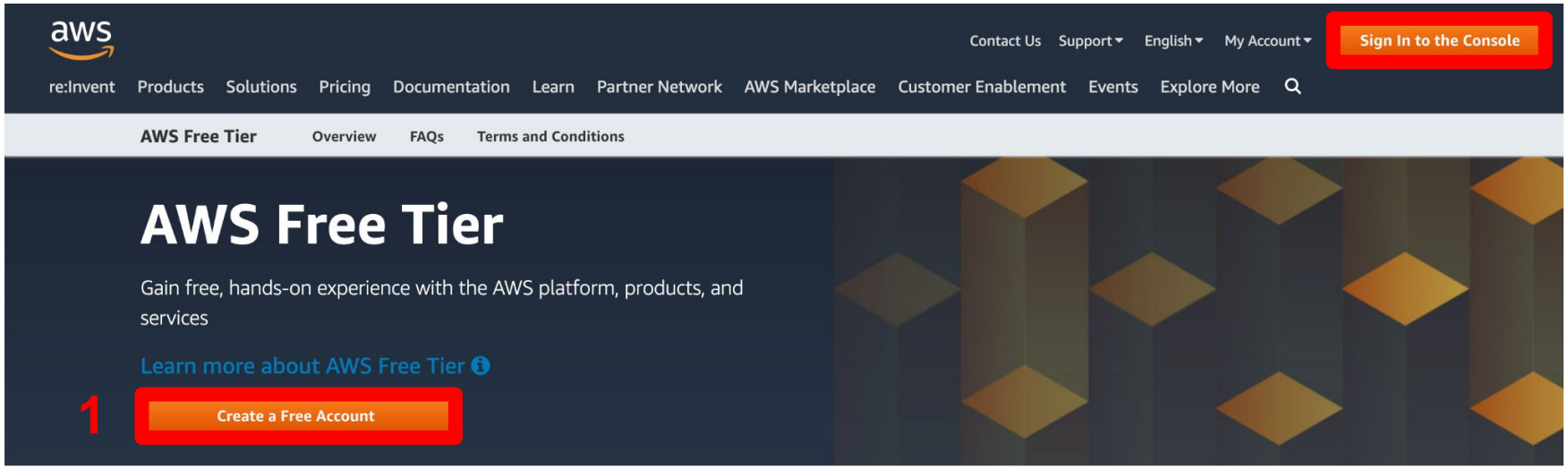
You must deploy your service but...

- Purchasing your own server is expensive
- Need some decent space for your server
- Need continuous and reliable power supply
- Should acquire a public IP address for your server
- ...

How to create an AWS instance

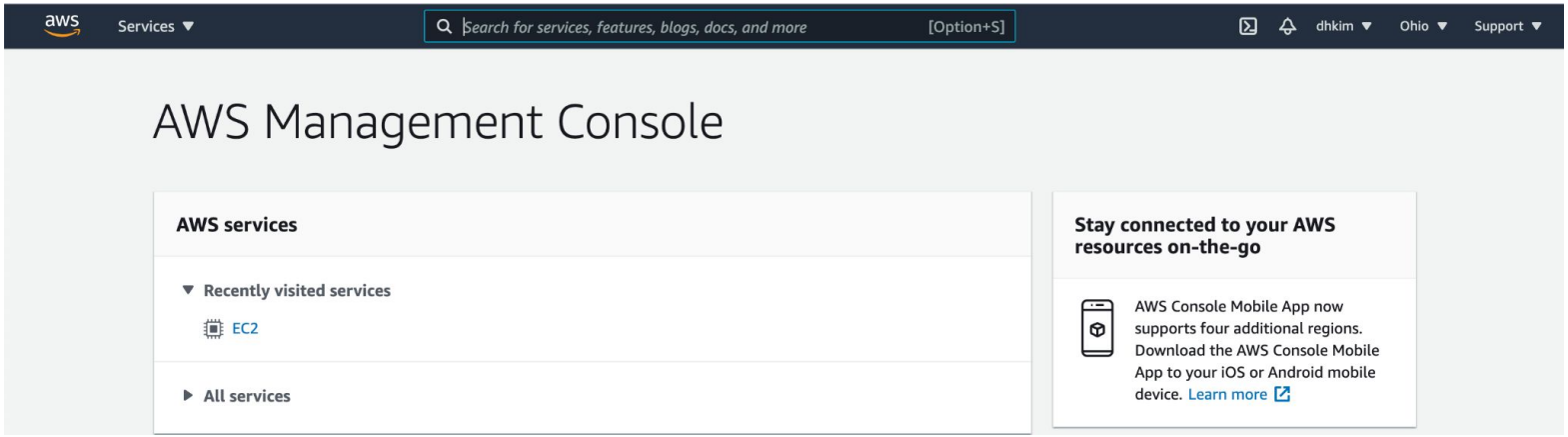
Create an AWS Free Tier Account (https://aws.amazon.com/free/)

2



How to create an AWS instance

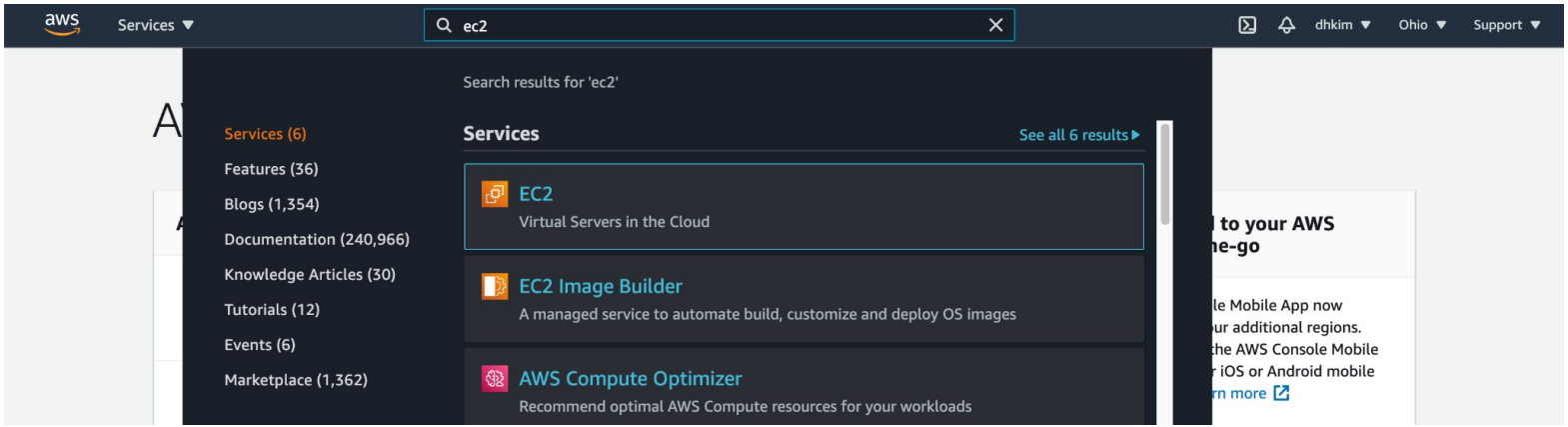
Get in AWS Management Console



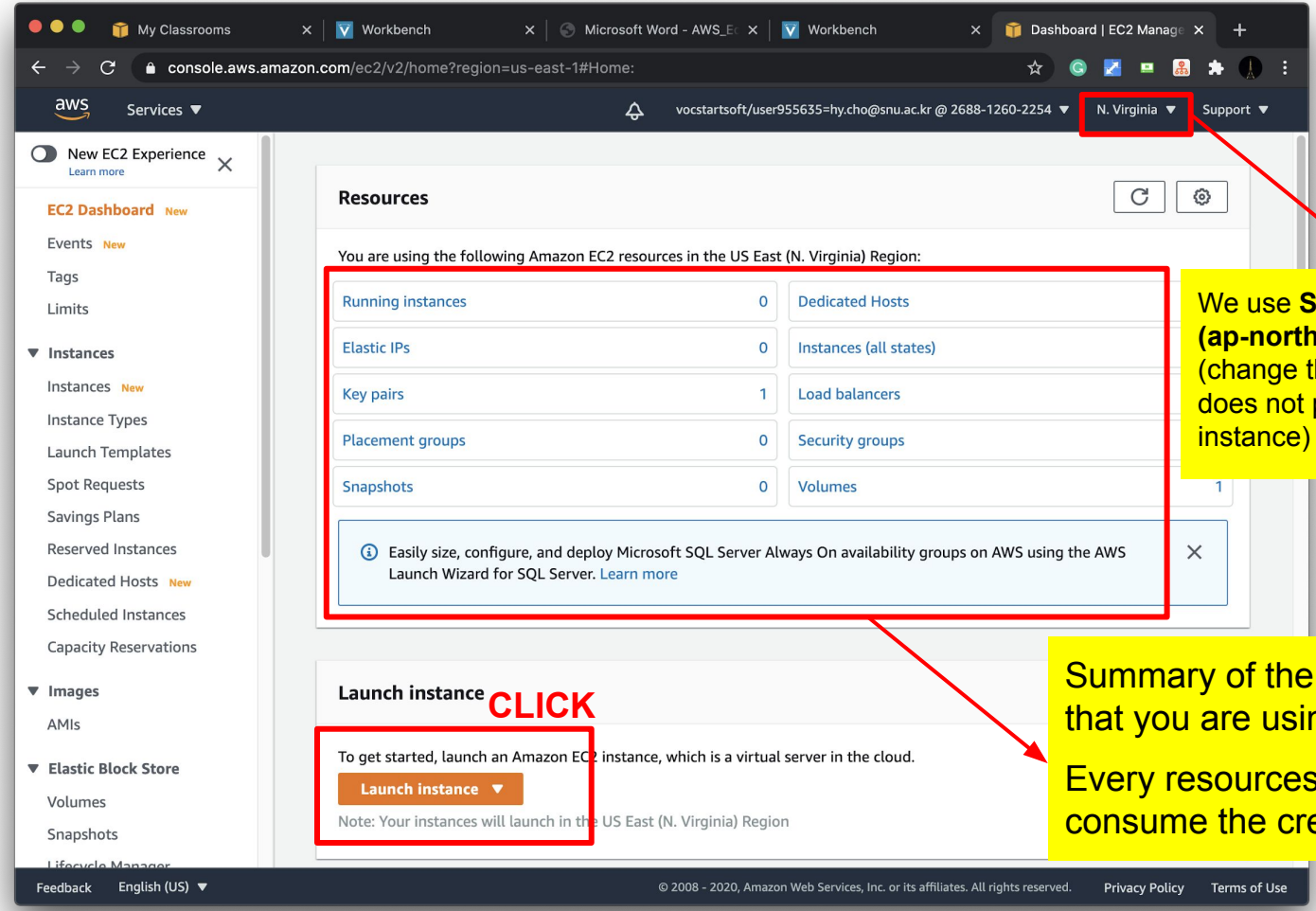
How to create an AWS instance

Get in AWS Management Console

- Find EC2 and click ec2 dashboard



AWS



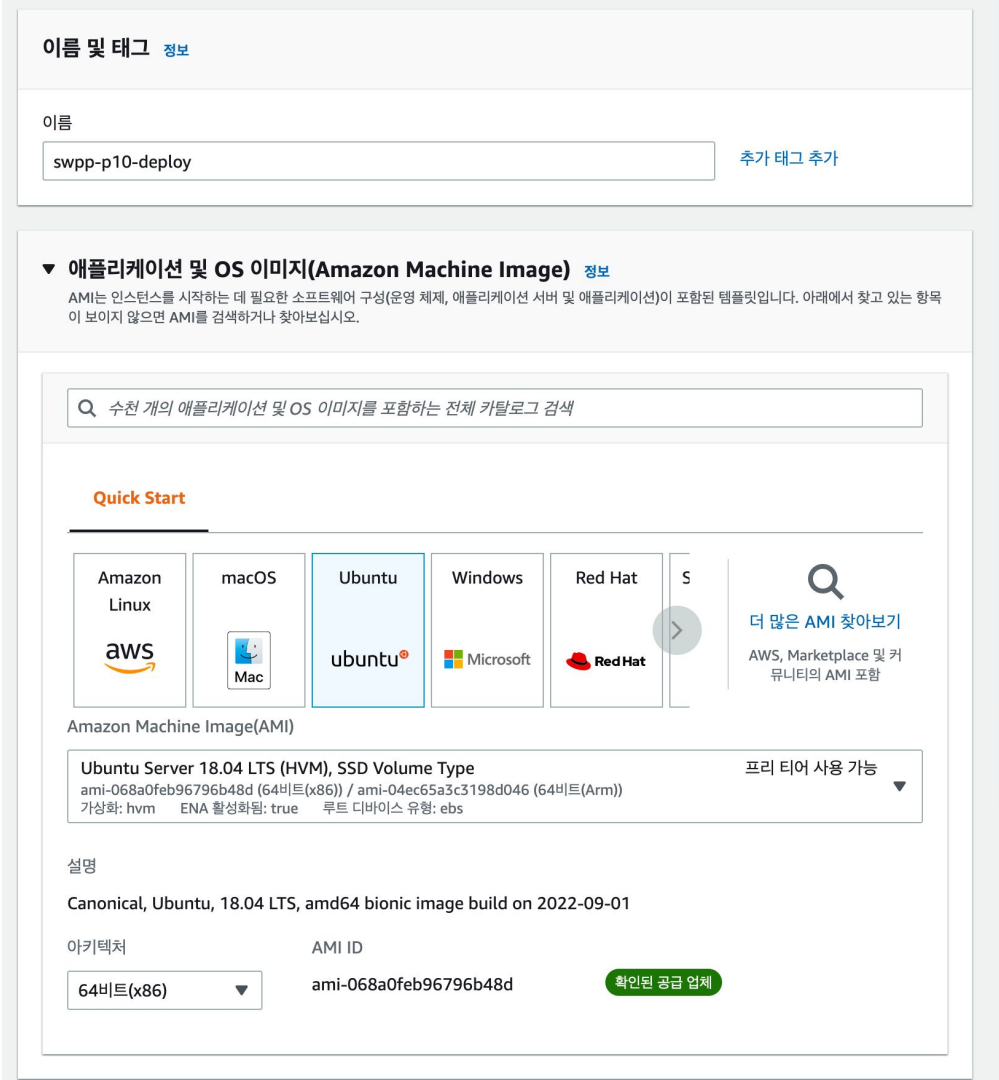
We use **Seoul (ap-northeast-2)** region (change the region if AWS does not provide you an instance)

Summary of the resources that you are using

Every resources you take will consume the credits

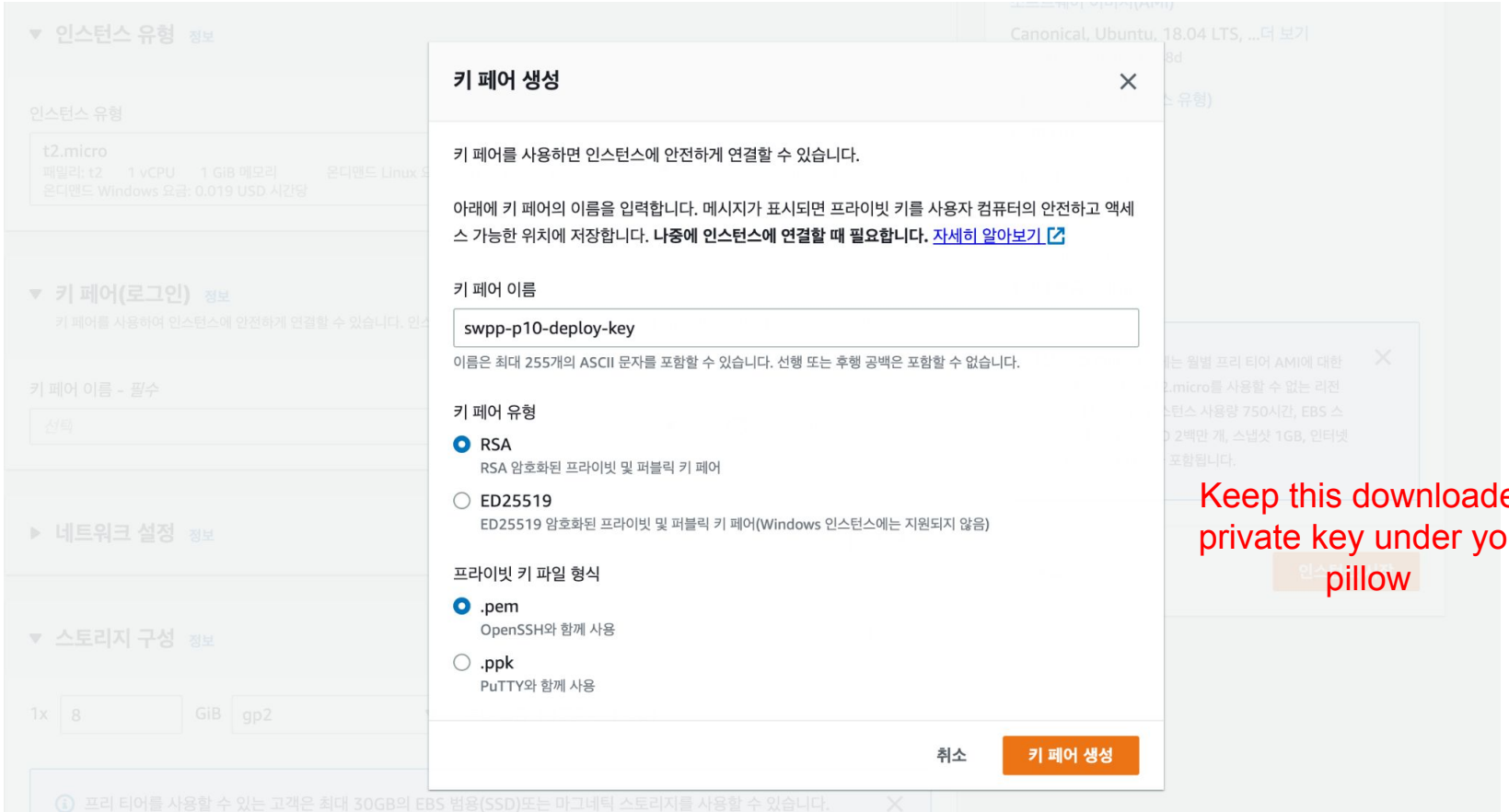
Create your instance

- **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type**



Create your instance

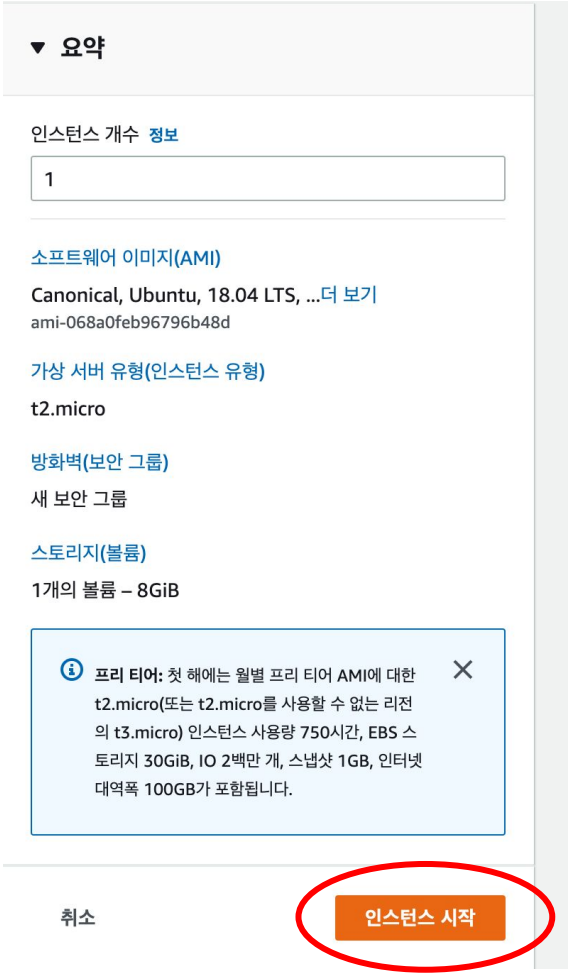
Create your instance



Keep this downloaded private key under your pillow

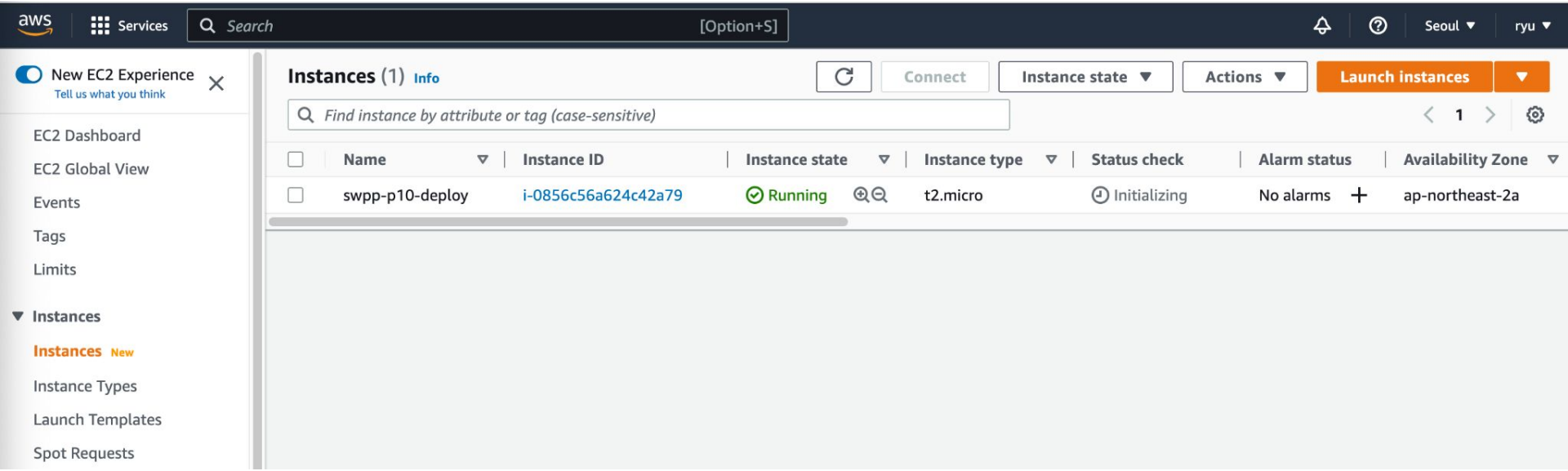
Create your instance

Create your instance



Create your instance

- Now, we've created a fresh linux instance!
- May take a while to initialize



Connecting EC2 Instance

- Click instance ID to see instance summary
- Use public DNS or IP to access your instance

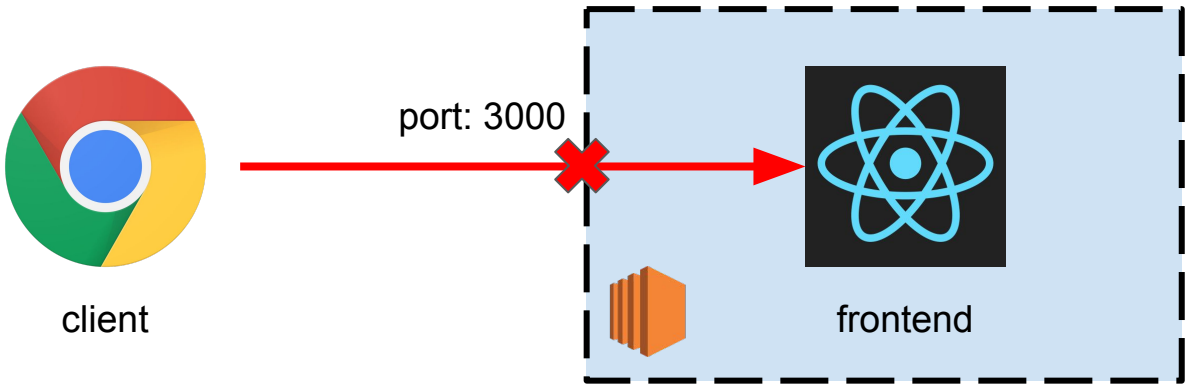
```
$ chmod 400 /path/to/your_key.pem
$ ssh -i /path/to/your_key.pem ubuntu@ip_address

OR

$ chmod 400 /path/to/your_key.pem
$ ssh -i /path/to/your_key.pem ubuntu@PUBLIC_DNS
```

Connected!

```
ryu@yujun-yeol-ui-MacBookPro swpp % chmod 400 swpp-p10-deploy-key.pem
ryu@yujun-yeol-ui-MacBookPro swpp % ssh -i swpp-p10-deploy-key.pem ubuntu@3.35.229.156
The authenticity of host '3.35.229.156 (3.35.229.156)' can't be established.
ED25519 key fingerprint is SHA256:hNGnfm9o5FOcDuSz83gbj1GtmJTPNAWZQsYIvOEKWW0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.35.229.156' (ED25519) to the list of known hosts.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1084-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Nov 13 16:29:34 UTC 2022

  System load:  0.01              Processes:           97
  Usage of /:   16.1% of 7.57GB   Users logged in:     0
  Memory usage: 19%               IP address for eth0: 172.31.6.126
  Swap usage:   0%

0 updates can be applied immediately.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-6-126:~$
```

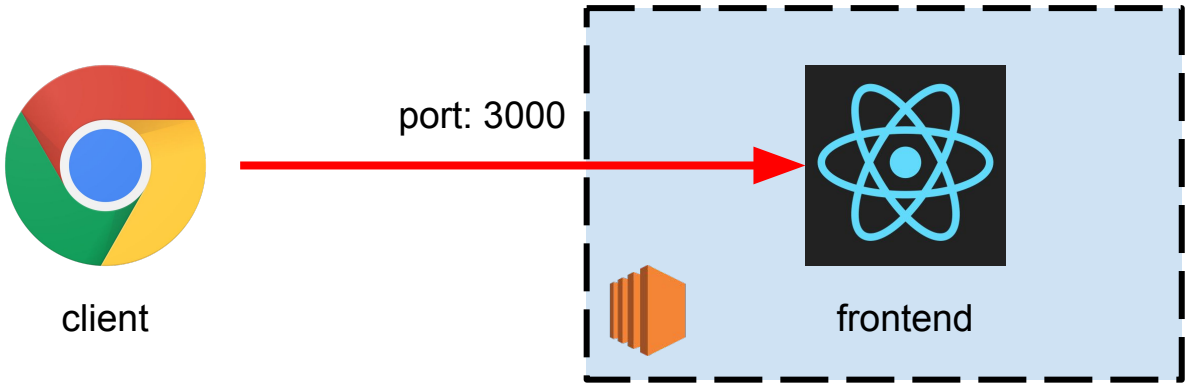
Allow Web Access

- Currently, no public access is allowed for the EC2 instance.
  - When you access port 3000, the connection will fail.



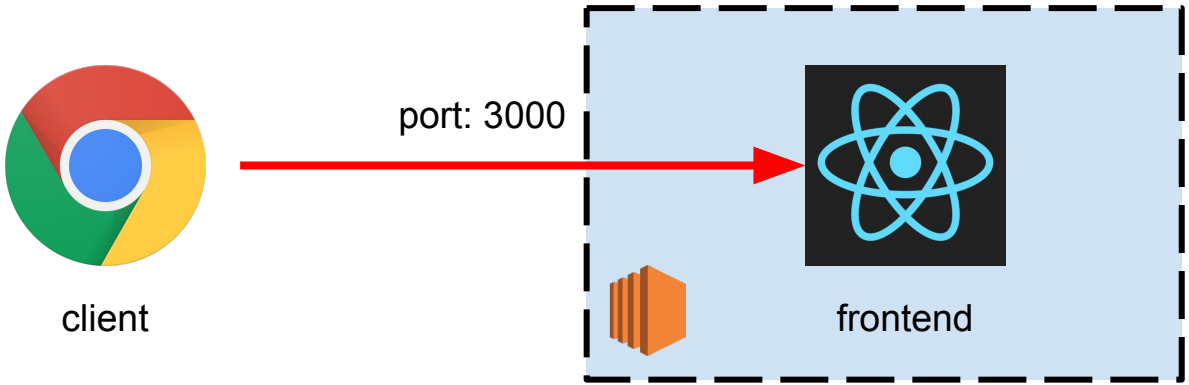
Allow Web Access

- Currently, no public access is allowed for the EC2 instance.
- Security group with new rule is required to allow inbound access (client => service).
  - We will allow *port 22 for SSH* and *port 3000, 8000 for HTTP* for now
  - If you need to allow more ports, you can add them as much as you want (Be careful for the security issues)

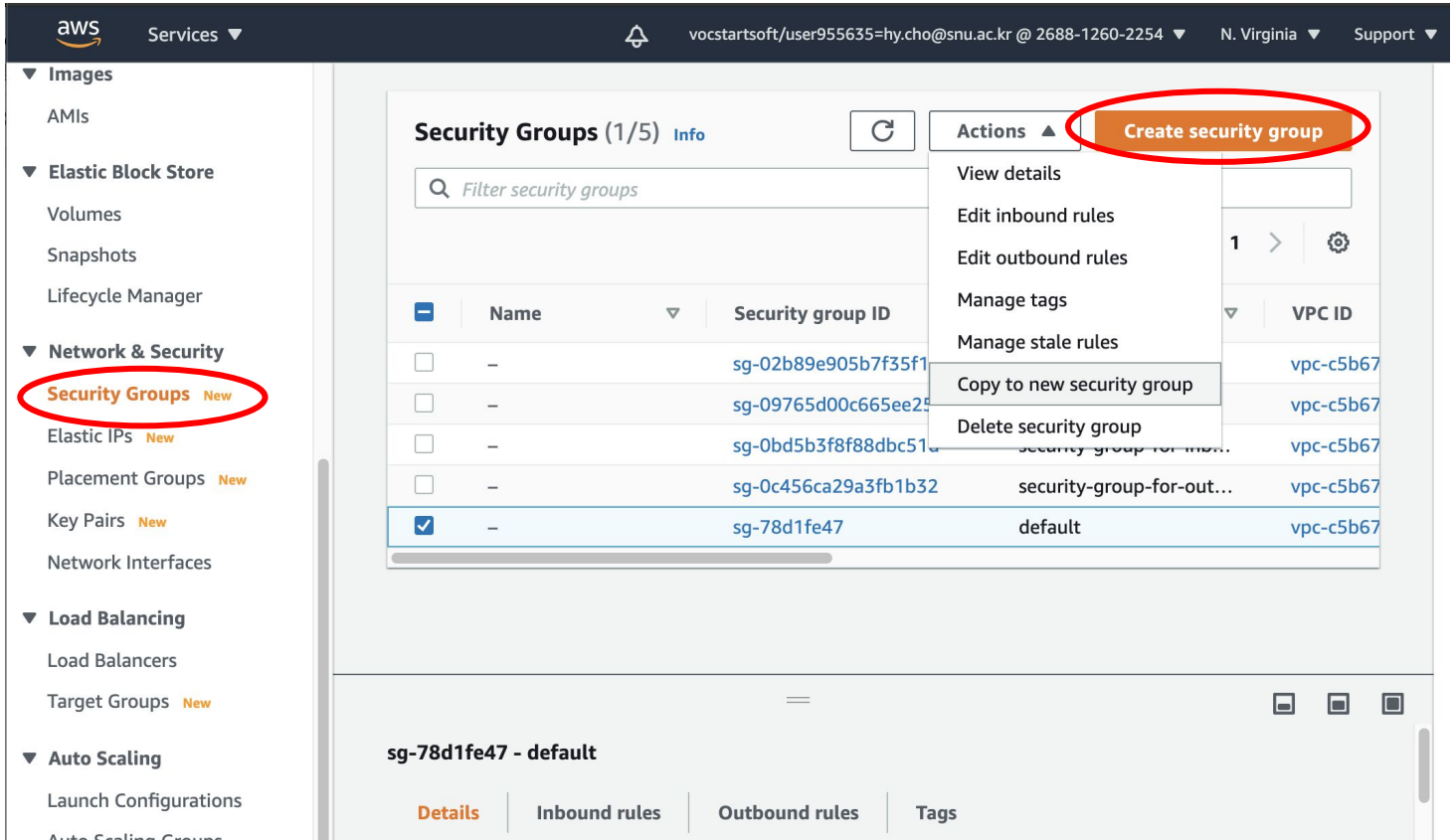


Allow Web Access

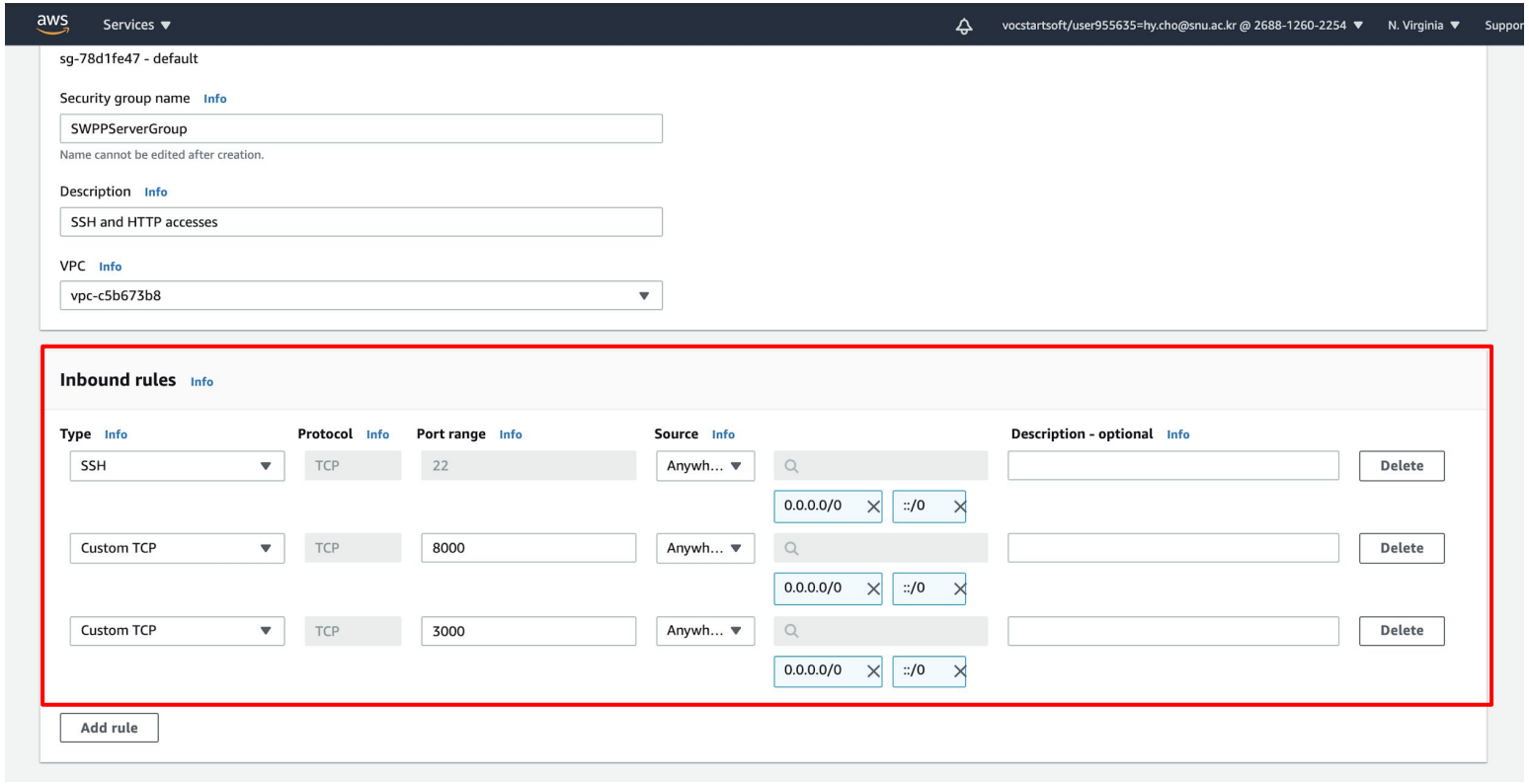
- Currently, no public access is allowed for the EC2 instance.
- Security group with new rule is required to allow inbound access (client => service).
- When the request gets into the instance, the request will be forwarded to the application.
    - Suppose the application is listening to the port.



Allow Web Access

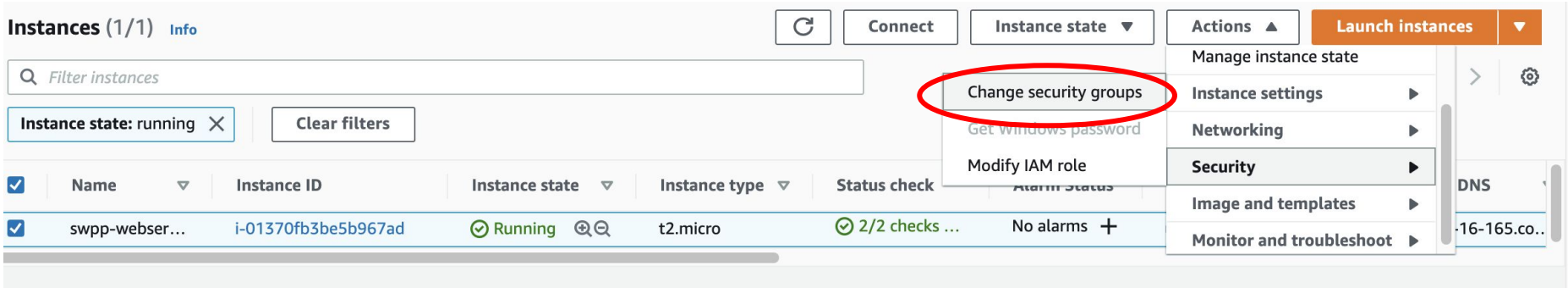


Allow Web Access

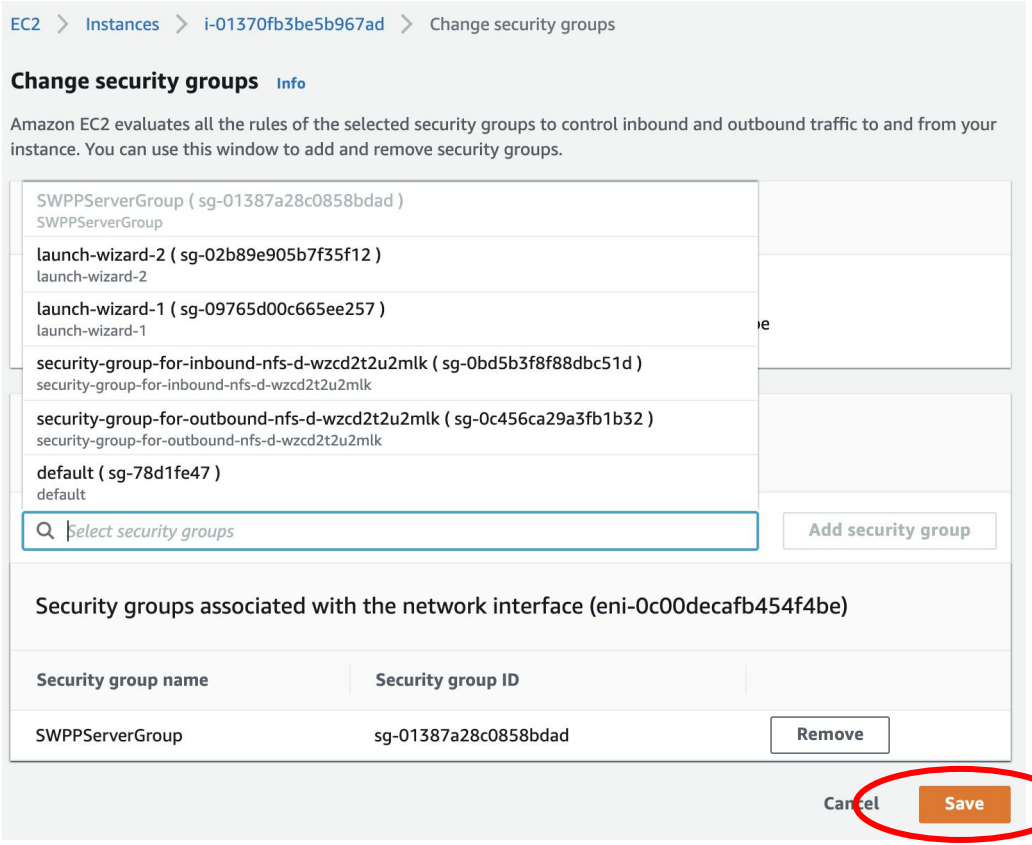


Allow Web Access

- Go to Instances



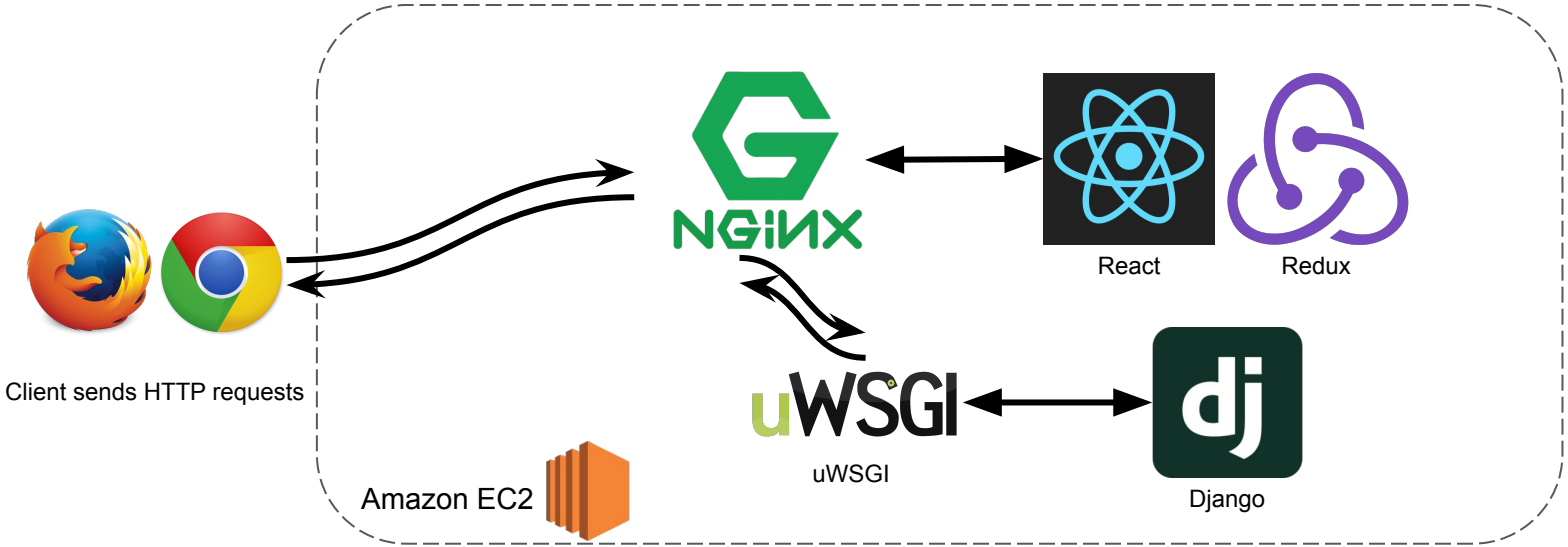
Allow Web Access



Docker

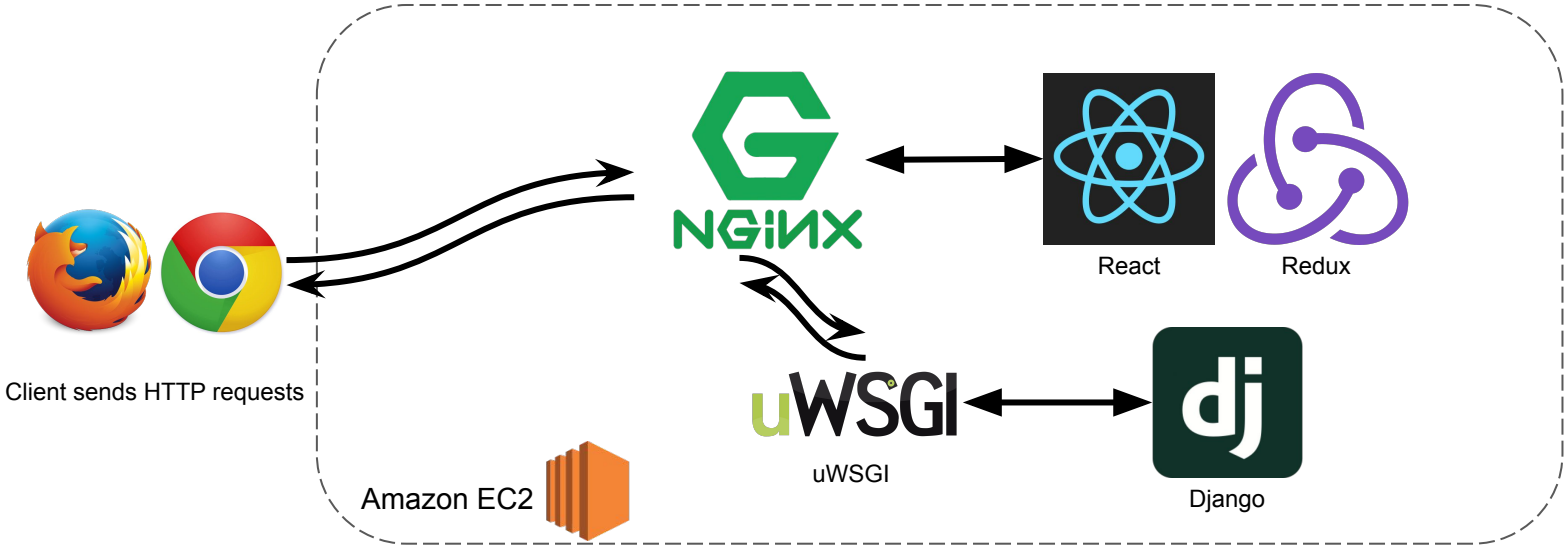
Why using Docker for deploying our services?

1. When deploying and maintaining a web service in a new EC2 instance
   - There is no way to check if the service works until you install all requirements and run apps.
   - It is hard to keep the instance environment consistent. (e.g. Changing the dependencies, version updates)



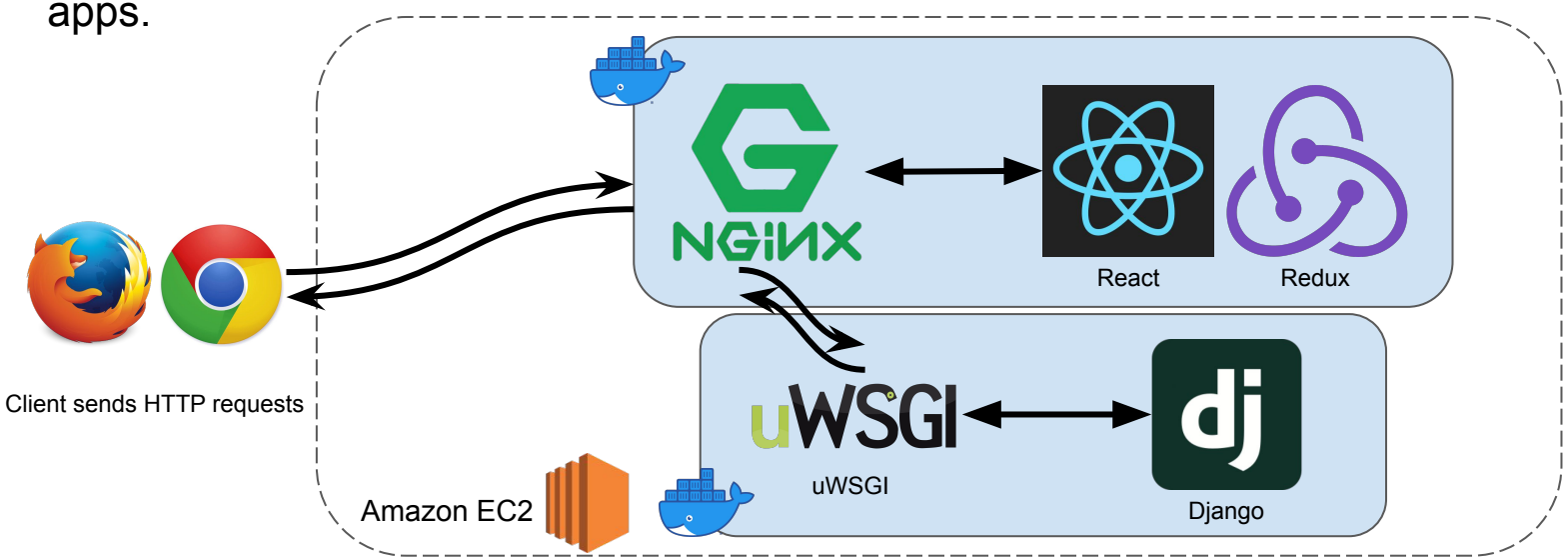
Why using Docker for deploying our services?

2. When restarting a new EC2 instance
   - You have to re-install the required dependencies.
   - If you have changed the instance type, you may need to change the requirements.



Why using Docker for deploying our services?

- With Docker Containers:
  - Even when you restart an EC2 instance, there is no need to manually install dependencies.
  - Also, it is easy to deploy a web server in a consistent manner.
- We will use Docker Containers to run a web server including both frontend and backend apps.



Inside your EC2 instance

Install Docker (Ubuntu)

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce
```

Recommended version for Docker Engine: 20.10.XX

Inside your EC2 instance

Install Docker

```
$ sudo docker version
```

```
# output:
Client: Docker Engine - Community
 Version:           20.10.05
 API version:       1.41
 Go version:        go1.13.15
 Git commit:        55c4c88
 Built:             Tue Mar  2 20:13:00 2021
 OS/Arch:           darwin/amd64
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.05
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  ...
```

Inside your EC2 instance

Setup integration repo

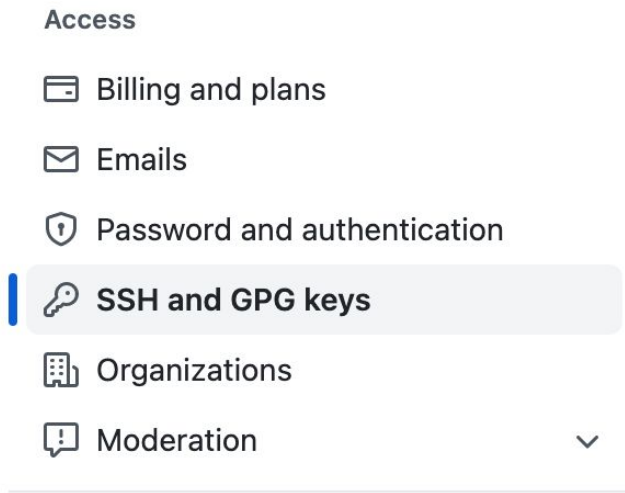
[Option 1]

Clone the FE + BE integrated repo from
https://github.com/swpp22fall-practice-sessions/swpp-p10-deploy

OR from your integration repo from session 8

→ You will need to add github key
& add to your github keys if you want to clone the repo

```
$ ssh-keygen -t rsa -b 4096 -C "your.email@domain.com"

$ cat $generated_key_path        # Register this to
```



Inside your EC2 instance

By now, your instance should have

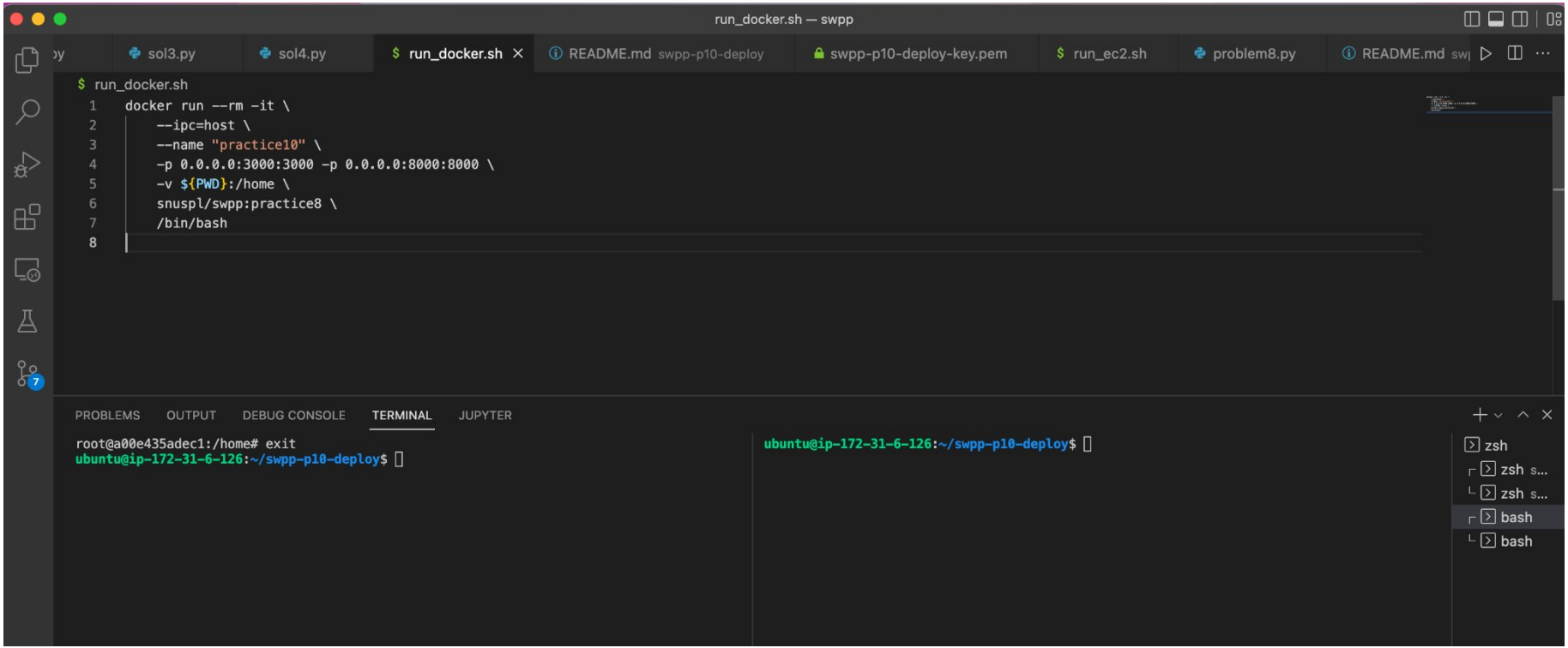
```
ubuntu@ip-172-31-6-126:~$ ls
swpp-p10-deploy   swpp-p10-deploy.zip
```

Inside your EC2 instance

Setup - docker container

- pull `snuspl/swpp:practice8` image
- move into integration repo
  - cd ${your_integration_repo}

Run two terminals for the instance (1 for FE, 1 for BE)



Inside your EC2 instance

Setup - docker container

- run docker container

```
$ sudo docker run --rm -it \
     --ipc=host \
     --name "practice10" \
     -p 0.0.0.0:3000:3000 -p 0.0.0.0:8000:8000 \
     -v ${PWD}:/home \
     snuspl/swpp:practice8
```

- Check docker container running

```
sudo docker ps
```

- Get inside running container

```
sudo docker exec -it practice10 bash
```

Inside your EC2 instance

Run your FE & BE

```
$ vi frontend/node-modules/react-scripts/config/webpackDevServer.config.js

# change `disableHostCheck` to `true`
```

How to use vim editor

Press “i” before start editing
Press “esc” when done editing
Press “:wq” to save and quit

We will revert this after “HTTPS”, so do not remove the original value but just comment it out!
This is just to check out our app working on EC2 instance DNS address

Inside your EC2 instance & Inside docker

Run your FE & BE

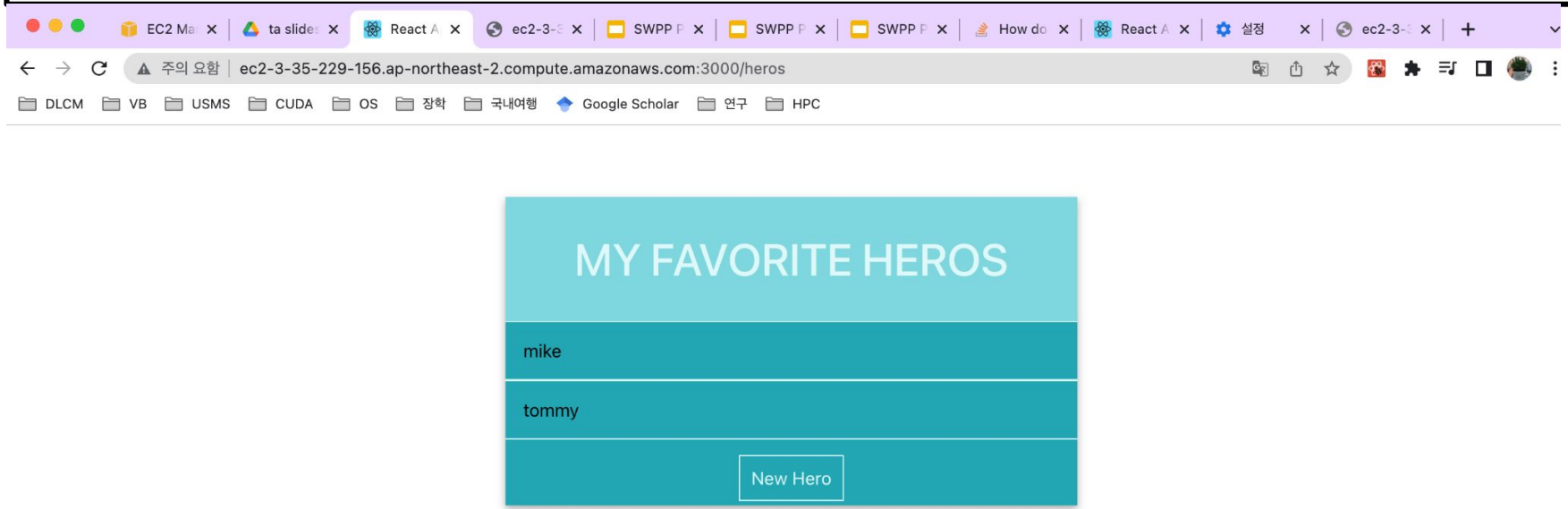
- FE

```
cd frontend
yarn
yarn start
```

- BE

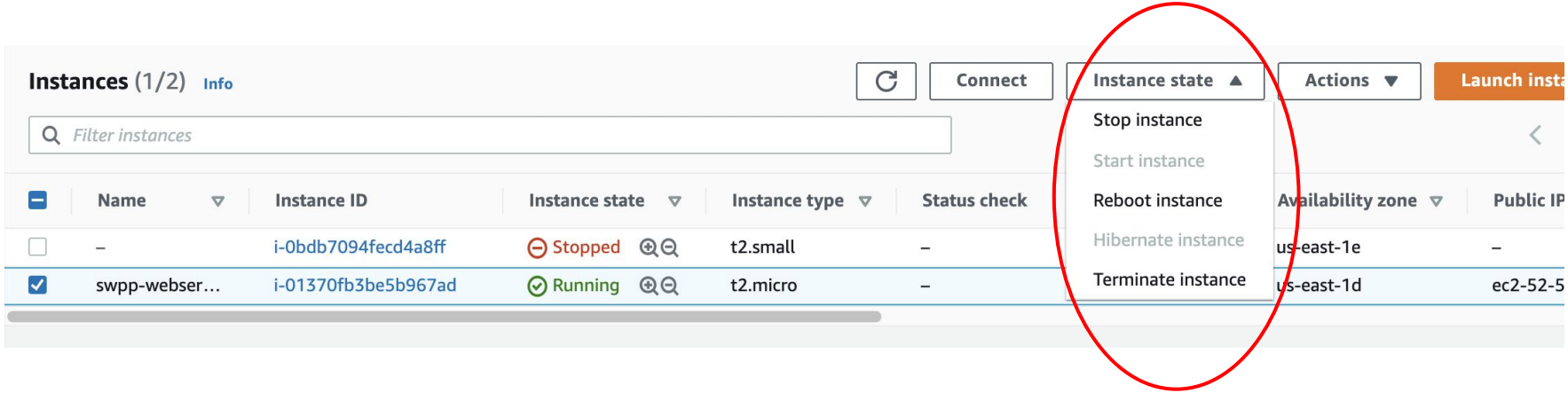
```
cd backend
python manage.py makemigrations
python manage.py migrate
python manage.py runserver 0.0.0.0:8000
```

TADA !


주의 요함
ec2-3-35-229-156.ap-northeast-2.compute.amazonaws.com:3000/heros
DLCM
VB
USMS
CUDA
OS
장학
국내여행
Google Scholar
연구
HPC
MY FAVORITE HEROS
mike
tommy
New Hero

Don’t forget to shutdown!

- If you do not want to use the machine anymore, you should stop the instance so that no credit will be wasted
- `Stop instance` will temporarily shutdown the instance
- `Terminate instance` will **delete** your instance (be careful)



Tip: Elastic IPs

- **If you shutdown the instance and start it again, your public IP and public DNS will change!**
- AWS provides Elastic IP service, where you can reserve an IP address and associate it with an EC2 instance so that it uses that IP address
- https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
- There are a lot services you can take advantage of in AWS!

What’s next?

1. Setup a web server
2. HTTPS
3. DB optimizing code

Don’t forget to submit your HW code refactoring!

Announcement: Today’s Task

- Q/A form
  - 1. Choose a piece of code example. We recommend you to pick a view function you wrote for HW4, or some React component you wrote for HW3. Any other code example is fine.
    2. Identify code smell and write briefly which solution you can apply.
    3. Refactor the code example.
  - link: https://forms.gle/CTcfYHgze4EdvBvx9
  - Due 11:59pm of 11/17 (Thur)