

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)

#10 - Refactoring

<Original code>
: Paste the original code to refactor.
: (Important) Add comments on the code where you will refactor, and why refactoring is needed here.

<Refactored code>
: Paste the refactored code

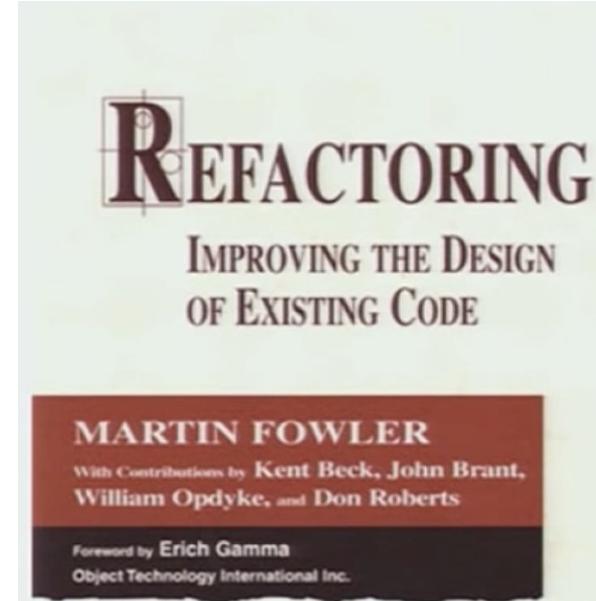
Name of the code piece (xx.py) and briefly explain its functionality
단답형 텍스트

Original code
장문형 텍스트

Refactored code
장문형 텍스트

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 > ...
1  # TODO: 1. Use one-liners
2  def count_asterisk_exceed_five(text):
3  |    return text.count("*") > 5
4
5  if __name__ == "__main__":
6  |    text = """Because he's the hero Gotham deserves but not the one it needs right now.
7  So we will hunt him, because he can take it. Because he's not out hero.
8  He is a silent guardian, a watchful protector... a dark knight."""
9
10 word1 = "hero"
11 word2 = "silent"
12
13 # TODO: 4. Consider general usage
14 # TODO: 3. Substitute algorithm
15
16 # TODO: 2. Extract method, remove duplicated code
17 # Censor `word1` from `text`
18 tmp = text
19 while word1 in tmp:
20 |    tmp = tmp[:tmp.find(word1)] + "*" * len(word1) + tmp[tmp.find(word1) + len(word1):]
21 text = tmp
22
23 if count_asterisk_exceed_five(text):
24 |    print("More than five *")
25
26 # TODO: 2. Extract method, remove duplicated code
27 # Censor `word2` from `text`
28 tmp = text
29 while word2 in tmp:
30 |    tmp = tmp[:tmp.find(word2)] + "*" * len(word2) + tmp[tmp.find(word2) + len(word2):]
31 text = tmp
32
33 if count_asterisk_exceed_five(text):
34 |    print("More than five *")
35
36 # Print result `text`
37 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 > ...
1  # TODO: 1. Use one-liners
2  def count_asterisk_exceed_five(text):
3  |   return text.count("*") > 5
4
5  if __name__ == "__main__":
6  |   text = """Because he's the hero Gotham deserves but not the one it needs right now.
7  So we will hunt him, because he can take it. Because he's not out hero.
8  He is a silent guardian, a watchful protector... a dark knight."""
9
10 word1 = "hero"
11 word2 = "silent"
12
13 # TODO: 4. Consider general usage
14 # TODO: 3. Substitute algorithm
15
16 # TODO: 2. Extract method, remove duplicated code
17 # Censor `word1` from `text`
18 tmp = text
19 while word1 in tmp:
20 |   tmp = tmp[:tmp.find(word1)] + "*" * len(word1) + tmp[tmp.find(word1) + len(word1):]
21 text = tmp
22
23 if count_asterisk_exceed_five(text):
24 |   print("More than five *")
25
26 # TODO: 2. Extract method, remove duplicated code
27 # Censor `word2` from `text`
28 tmp = text
29 while word2 in tmp:
30 |   tmp = tmp[:tmp.find(word2)] + "*" * len(word2) + tmp[tmp.find(word2) + len(word2):]
31 text = tmp
32
33 if count_asterisk_exceed_five(text):
34 |   print("More than five *")
35
36 # Print result `text`
37 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

```
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

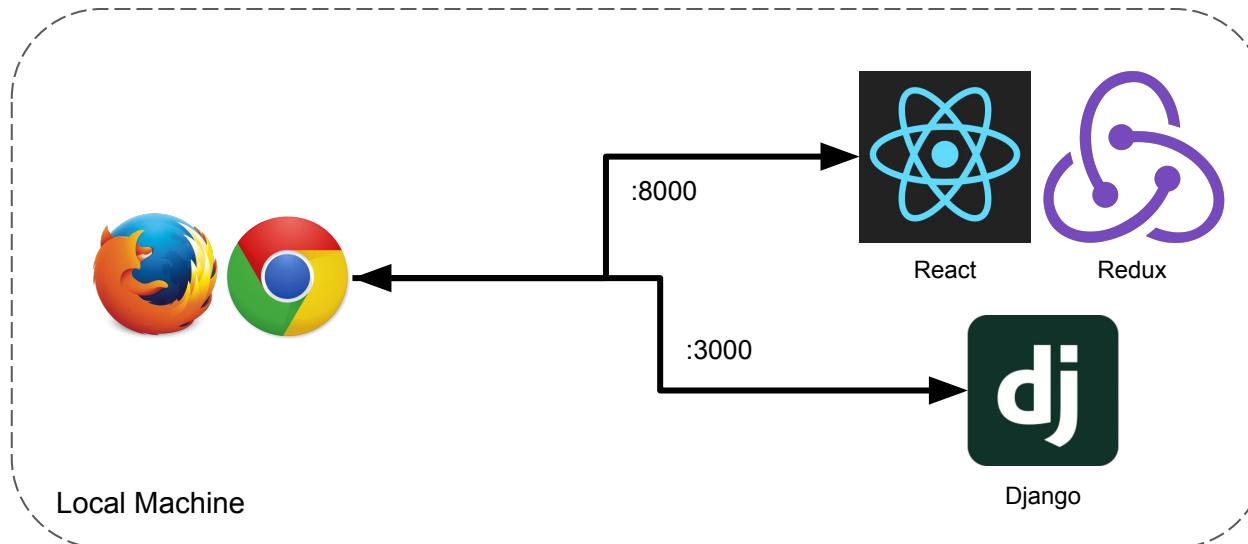
def check_logged_in(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if args and args[0].user.is_authenticated:
            return func(*args, **kwargs)
        return HttpResponseRedirect(status=401)
    return wrapper

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

Deployment pt.1

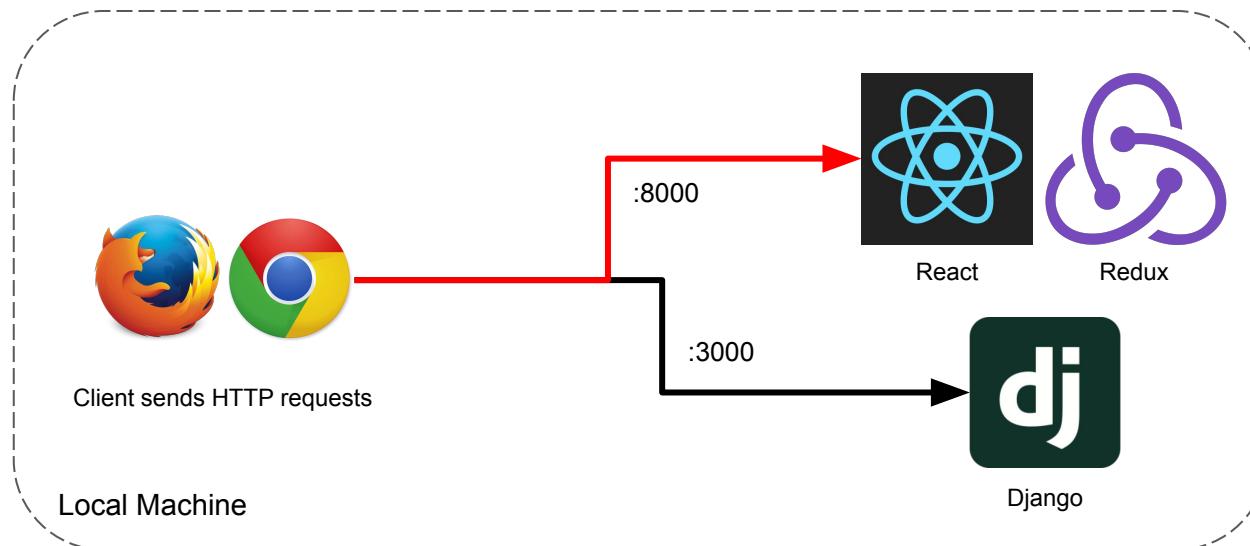
Until Now...

- 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.



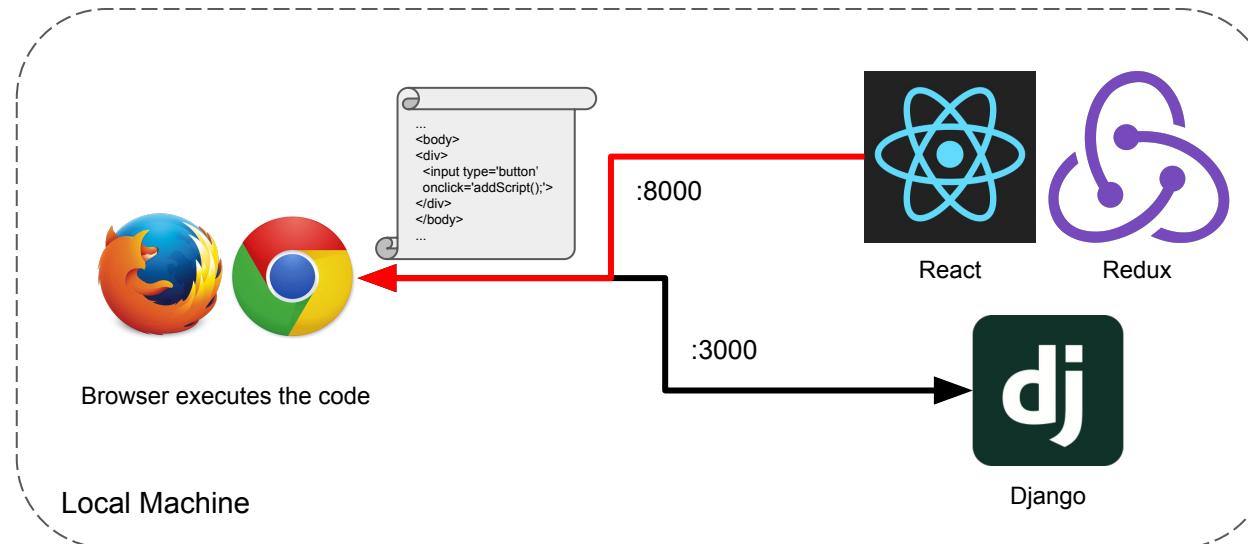
Until Now...

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



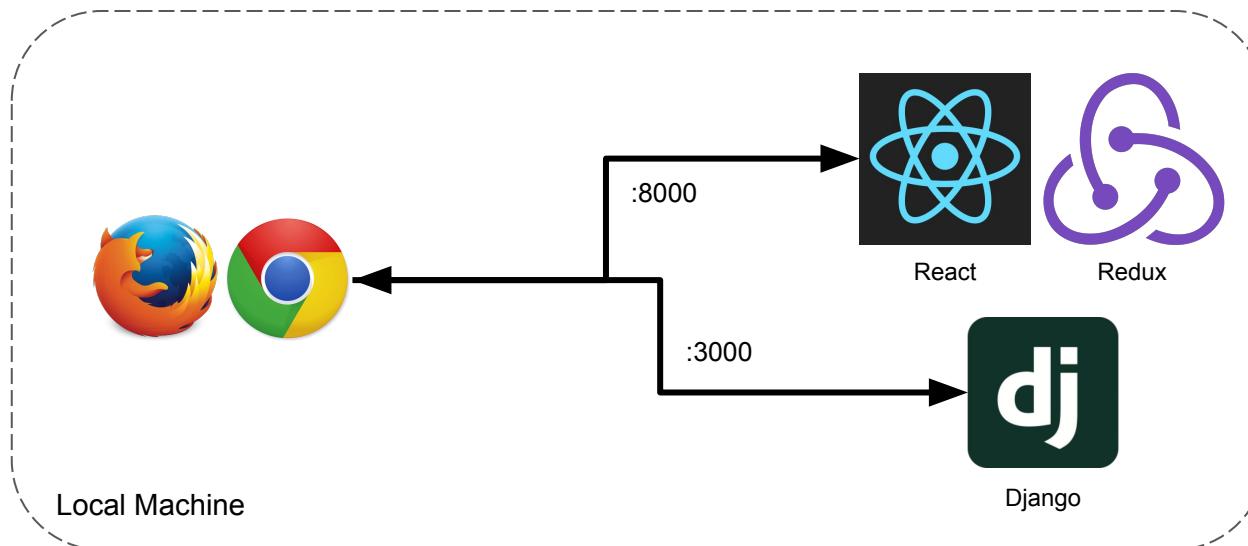
Until Now...

- 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.



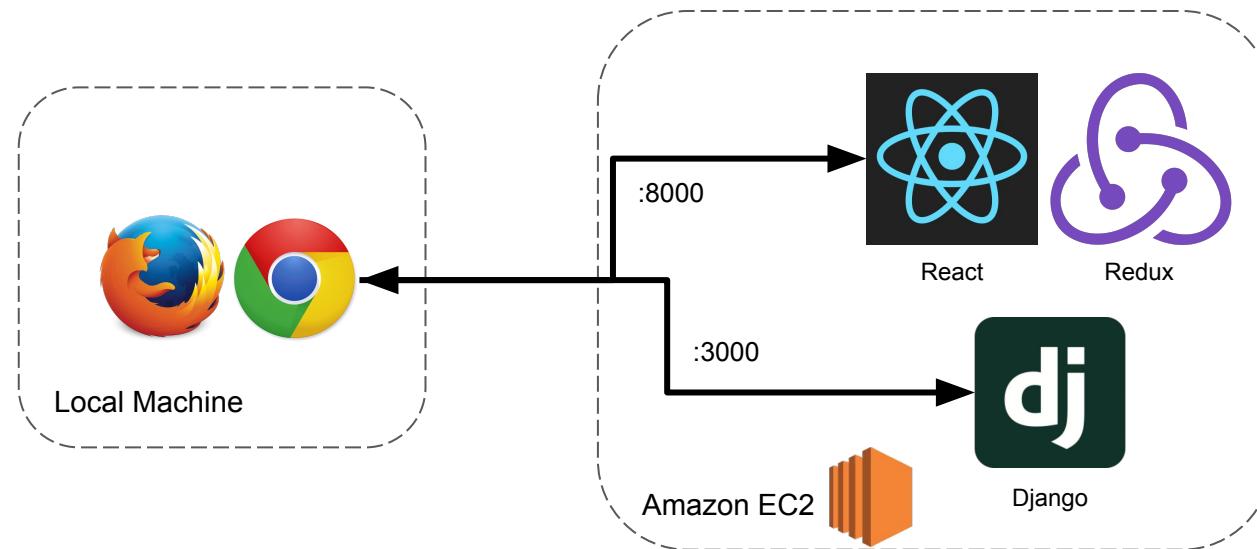
Until Now...

- 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.



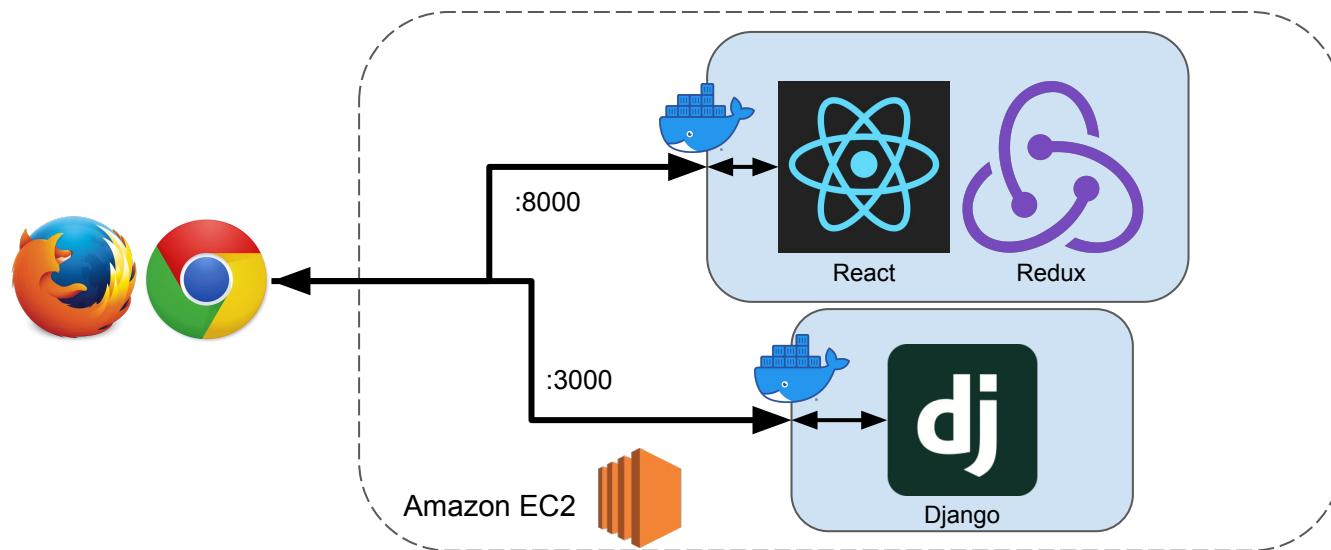
Objectives

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



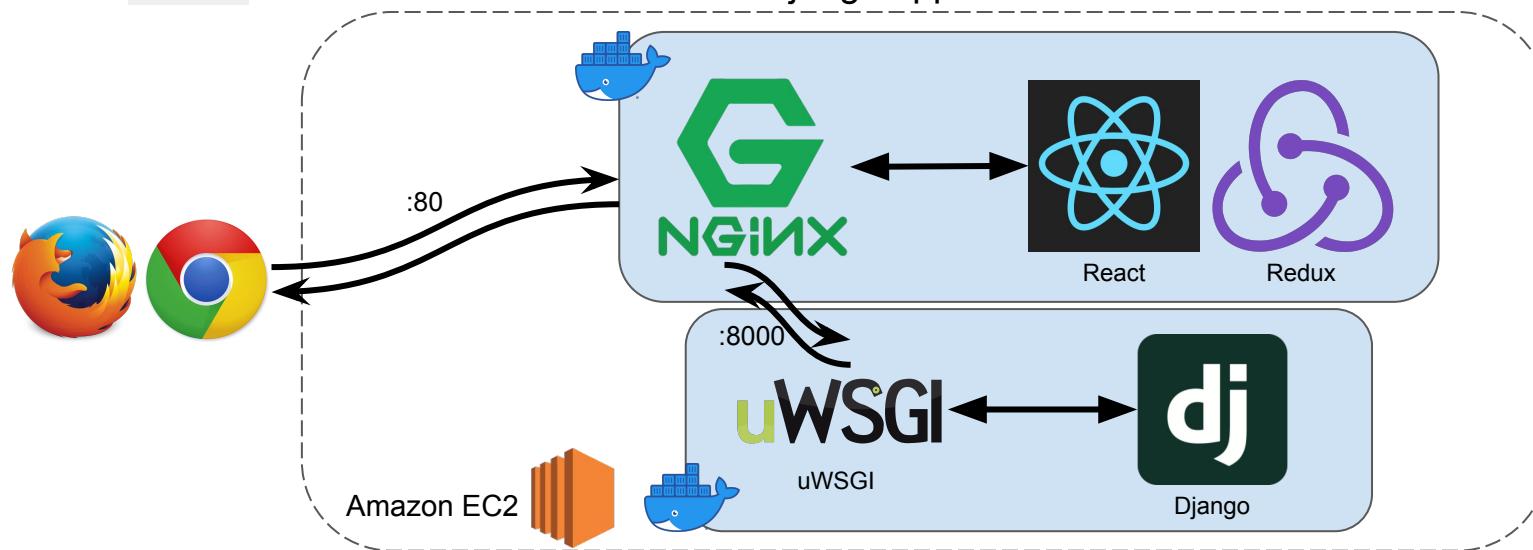
Objectives

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.



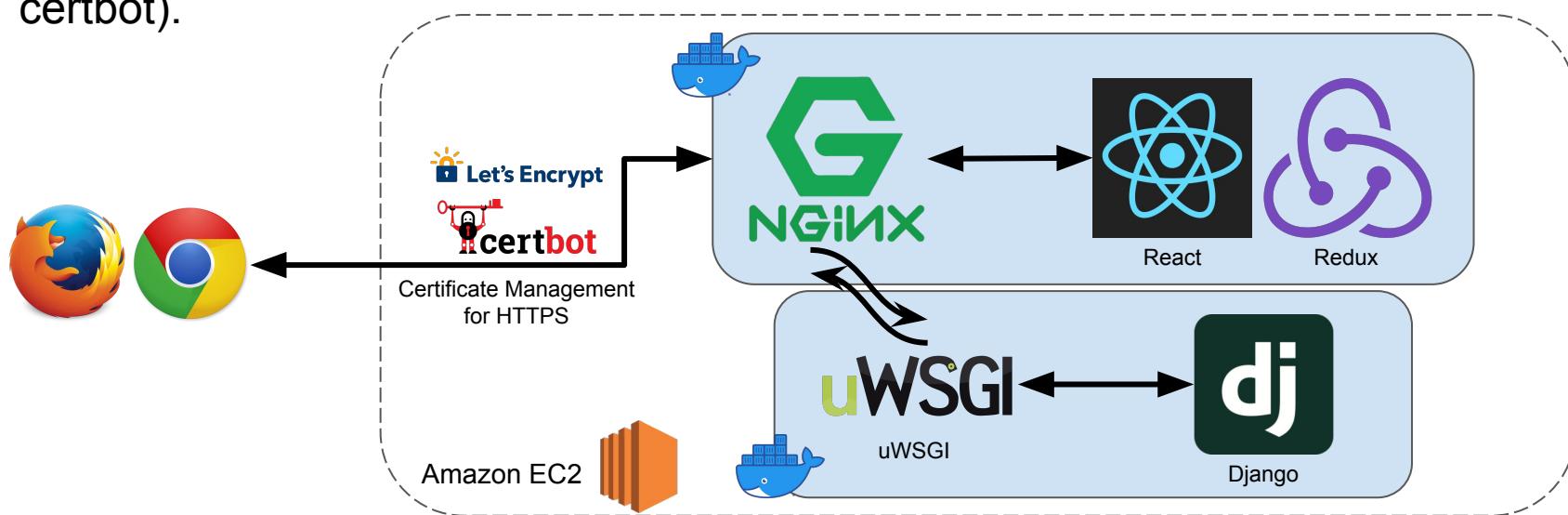
Objectives

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.



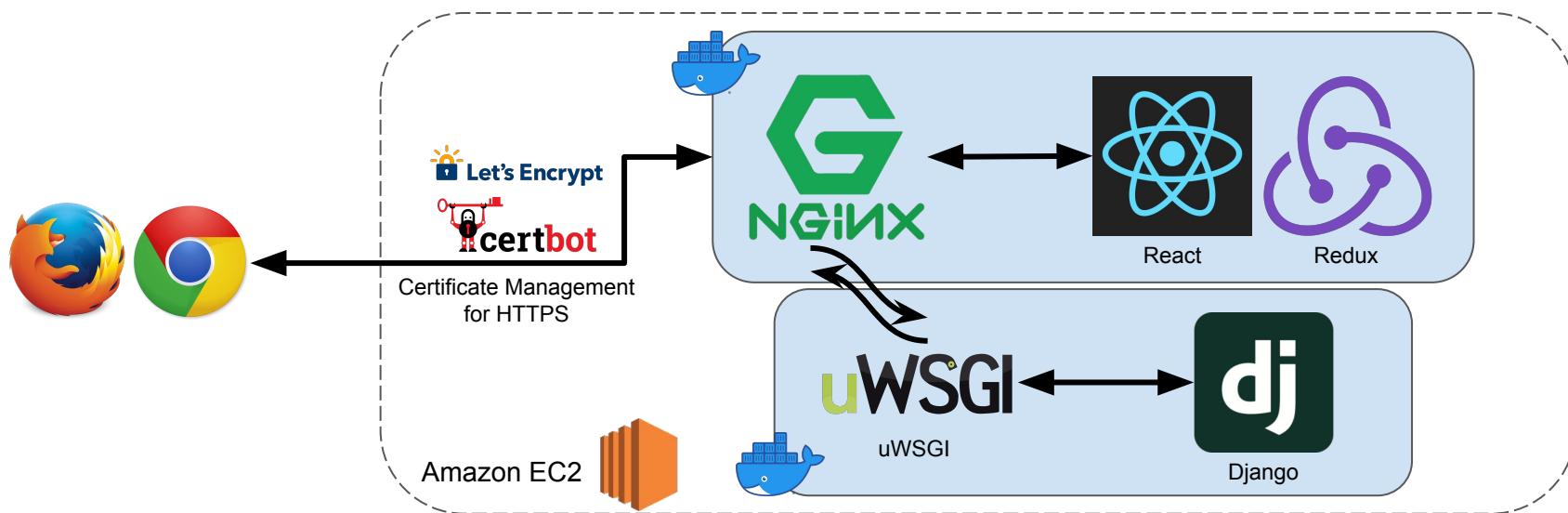
Objectives

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/>)

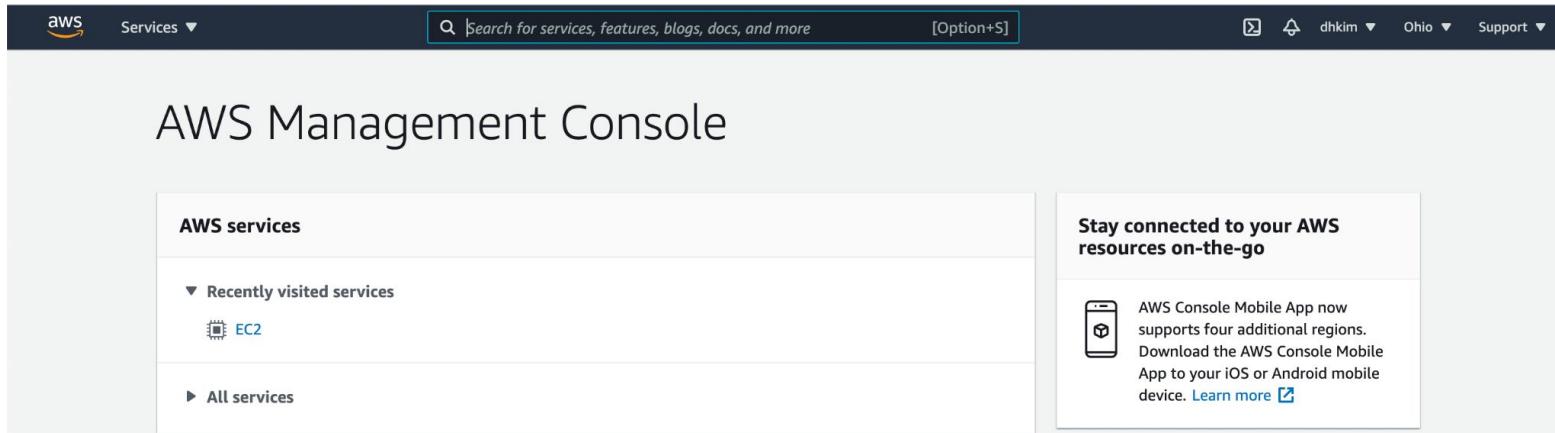
The screenshot shows the AWS Free Tier landing page. At the top, there's a dark header with the AWS logo, navigation links like 're:Invent', 'Products', 'Solutions', etc., and a 'Sign In to the Console' button. Below the header, a banner features the text 'AWS Free Tier' and a large orange 'Create a Free Account' button. The main content area has a dark background with a grid of 3D cubes and the heading 'AWS Free Tier'. A sub-headline says 'Gain free, hands-on experience with the AWS platform, products, and services' and a link 'Learn more about AWS Free Tier'. Step 1 is highlighted over the 'Create a Free Account' button, and step 2 is highlighted over the 'Sign In to the Console' button.

1 Create a Free Account

2 Sign In to the Console

How to create an AWS instance

Get in AWS Management Console



The screenshot shows the AWS Management Console homepage. At the top, there is a dark header bar with the AWS logo, a search bar containing "Search for services, features, blogs, docs, and more" and "[Option+S]", and user account information for "dhkim" and "Ohio". Below the header is a large, light-colored main area with the title "AWS Management Console". On the left, there is a sidebar titled "AWS services" with sections for "Recently visited services" (listing EC2) and "All services". On the right, there is a promotional box titled "Stay connected to your AWS resources on-the-go" featuring an icon of a smartphone and text about the AWS Console Mobile App.

AWS Management Console

AWS services

▼ Recently visited services

EC2

▶ All services

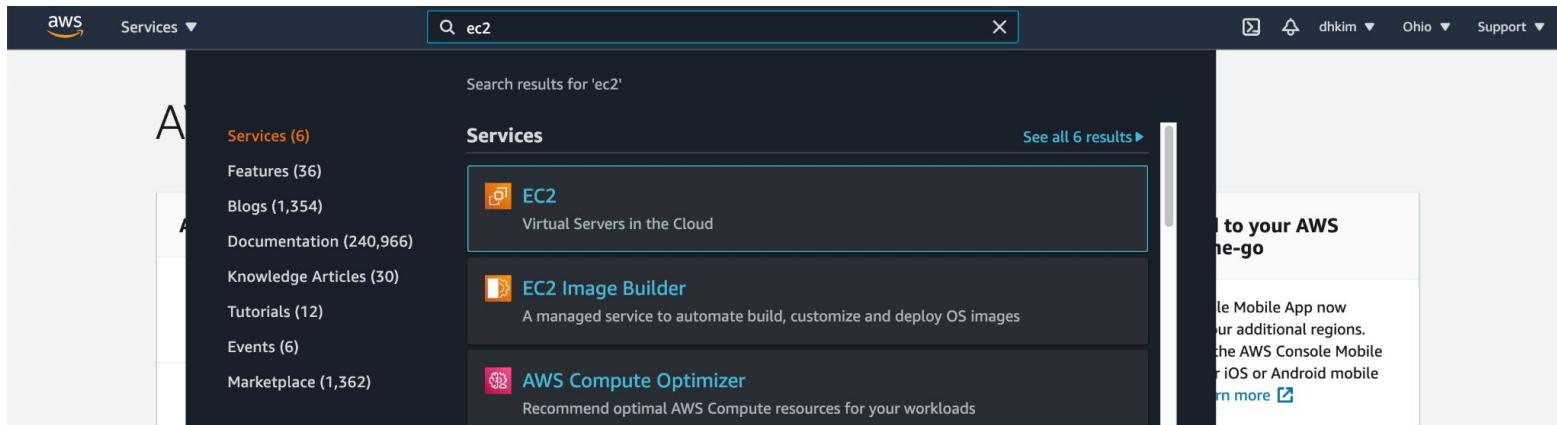
Stay connected to your AWS resources on-the-go

AWS Console Mobile App now supports four additional regions. Download the AWS Console Mobile App to your iOS or Android mobile device. [Learn more](#)

How to create an AWS instance

Get in AWS Management Console

- Find EC2 and click ec2 dashboard



AWS

The screenshot shows the AWS EC2 Dashboard. The top navigation bar includes tabs for 'My Classrooms', 'Workbench', 'Microsoft Word - AWS_Ec...', 'Workbench', and 'Dashboard | EC2 Manage'. The region dropdown is set to 'N. Virginia' (highlighted with a red box). The left sidebar lists categories like 'New EC2 Experience', 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances' (with sub-options like 'Instances', 'Instance Types', 'Launch Templates', etc.), 'Images', 'AMIs', 'Elastic Block Store' (with sub-options like 'Volumes', 'Snapshots', 'Lifecycle Manager'), and 'Feedback'.

The main content area displays 'Resources' with the following data:

Running instances	0	Dedicated Hosts
Elastic IPs	0	Instances (all states)
Key pairs	1	Load balancers
Placement groups	0	Security groups
Snapshots	0	Volumes

A tooltip message below the resources table reads: "Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more".

The bottom section features a large orange button labeled 'Launch instance CLICK' (with a red box around it). Below the button, text says 'To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.' and 'Launch instance ▾'. A note at the bottom states 'Note: Your instances will launch in the US East (N. Virginia) Region'.

Yellow callout boxes provide instructions:

- 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

Page footer: © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Create your instance

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

이름 및 태그 정보

이름 swpp-p10-deploy 추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat S 더 많은 AMI 찾아보기 AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
ami-068a0feb96796b48d (64비트(x86)) / ami-04ec65a3c3198d046 (64비트(Arm))
가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능 ▼

설명 Canonical, Ubuntu, 18.04 LTS, amd64 bionic image build on 2022-09-01

아키텍처 64비트(x86) AMI ID ami-068a0feb96796b48d 확인된 공급업체

Create your instance

▼ 인스턴스 유형 정보

인스턴스 유형

t2.micro

패밀리: t2 1 vCPU 1 GiB 메모리
온디맨드 Windows 요금: 0.019 USD 시간당

프리 티어 사용 가능

온디맨드 Linux 요금: 0.0144 USD 시간당



인스턴스 유형 비교

Create your instance

The screenshot shows the AWS Lambda 'Create your instance' wizard. The current step is 'Key Pair Creation'. A red box highlights the 'Private key' download link.

인스턴스 유형 Canonical, Ubuntu 18.04 LTS, ...더 보기
8d
인스턴스 유형
t2.micro
파일리: t2 1 vCPU 1 GiB 메모리
온디맨드 Linux 요금: 0.019 USD 시간당
온디맨드 Windows 요금: 0.019 USD 시간당

키 페어(로그인) 정보
키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스에 키 페어를 등록하면 SSH로 접속하거나 AWS Lambda와 같은 서비스에서 인스턴스에 자동으로 연결할 수 있습니다.

키 페어 이름 - 필수
선택

키 페어 이름
swpp-p10-deploy-key

이름은 최대 255개의 ASCII 문자를 포함할 수 있습니다. 실행 또는 후행 공백은 포함할 수 없습니다.

키 페어 유형
 RSA RSA 암호화된 프라이빗 및 퍼블릭 키 페어
 ED25519 ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식
 .pem OpenSSH와 함께 사용
 .ppk PuTTY와 함께 사용

스토리지 구성 정보
1x 8 GiB gp2

프리 티어를 사용할 수 있는 고객은 최대 30GB의 EBS 범용(SSD) 또는 마그네트릭 스토리지를 사용할 수 있습니다.

키 페어 생성

Keep this downloaded private key under your pillow

Create your instance

▼ 네트워크 설정 정보

네트워크 정보

vpc-07971557618b34b1e

서브넷 정보

기본 설정 없음(가용 영역의 기본 서브넷)

피블릭 IP 자동 할당 정보

활성화

방화벽(보안 그룹) 정보

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

보안 그룹 생성

기존 보안 그룹 선택

다음 규칙을 사용하여 'launch-wizard-1'(이)라는 새 보안 그룹을 생성합니다.

에서 SSH 트래픽 허용

인스턴스 연결에 도움

위치 무관
0.0.0.0/0

인터넷에서 HTTPS 트래픽 허용

예를 들어 웹 서버를 생성할 때 엔드포인트를 설정하려면

인터넷에서 HTTP 트래픽 허용

예를 들어 웹 서버를 생성할 때 엔드포인트를 설정하려면

어드밴스드

▼ 스토리지 구성 정보

1x GiB 루트 볼륨 (암호화되지 않음)

선택한 AMI에 인스턴스가 허용하는 것보다 많은 인스턴스 스토어 볼륨이 포함되어 있습니다. AMI에서 처음 0개의 인스턴스 스토어 볼륨에만 액세스할 수 있습니다.

0 x 파일 시스템

편집

Create your instance

▼ 요약

인스턴스 개수 [정보](#)
1

소프트웨어 이미지(AMI)
Canonical, Ubuntu, 18.04 LTS, ...[더 보기](#)
ami-068a0feb96796b48d

가상 서버 유형(인스턴스 유형)
t2.micro

방화벽(보안 그룹)
새 보안 그룹

스토리지(볼륨)
1개의 볼륨 – 8GiB

ⓘ **프리 티어:** 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다. **X**

취소 **인스턴스 시작**

Create your instance

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

The screenshot shows the AWS EC2 Instances page. At the top, there's a navigation bar with the AWS logo, a search bar, and various buttons like 'Connect' and 'Actions'. On the left, a sidebar titled 'Instances' is open, showing options like 'Instances New', 'Instance Types', 'Launch Templates', and 'Spot Requests'. The main content area displays a table for 'Instances (1)'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. The single instance listed is named 'swpp-p10-deploy' with Instance ID 'i-0856c56a624c42a79'. It is currently 'Running' (indicated by a green circle), has a 't2.micro' instance type, and is in the 'Initializing' status check phase. There are no alarms, and it is located in the 'ap-northeast-2a' availability zone.

Instances (1) Info						
<input type="text"/> Find instance by attribute or tag (case-sensitive)						
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	swpp-p10-deploy	i-0856c56a624c42a79	Running	t2.micro	Initializing	No alarms

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
```

Instance summary for i-069e75becce837ac8 [Info](#)
Updated less than a minute ago

Instance ID	i-069e75becce837ac8	Public IPv4 address	54.180.163.186 open address
IPv6 address	-	Instance state	Running
Private IPv4 DNS	ip-172-31-2-33.ap-northeast-2.compute.internal	Instance type	t2.micro
		Private IPv4 addresses	172.31.2.33
		Public IPv4 DNS	ec2-54-180-163-186.ap-northeast-2.compute.amazonaws.com open address
		Elastic IP addresses	-

[Edit](#) [Connect](#) [Instance state ▾](#) [Actions ▾](#)

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:hNGnfm9o5F0cDuSz83gbj1GtmJTPNAWZQsYIv0EKWW0.
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.

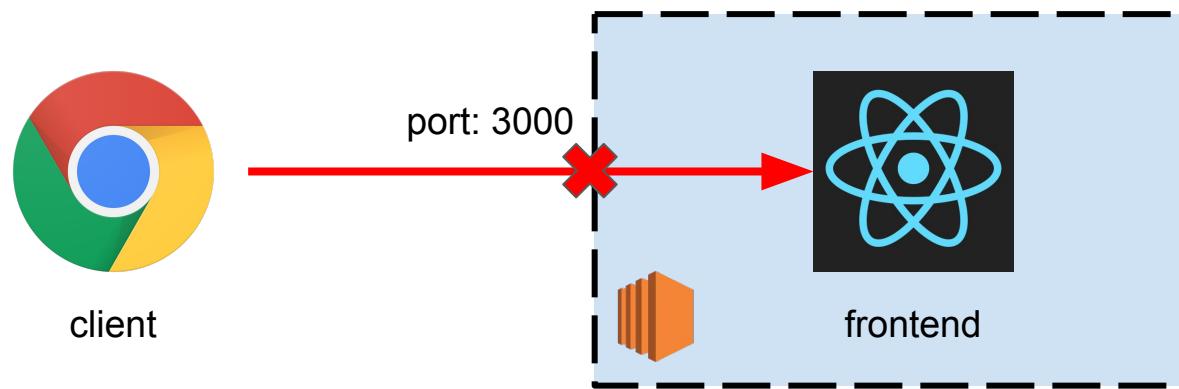
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

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

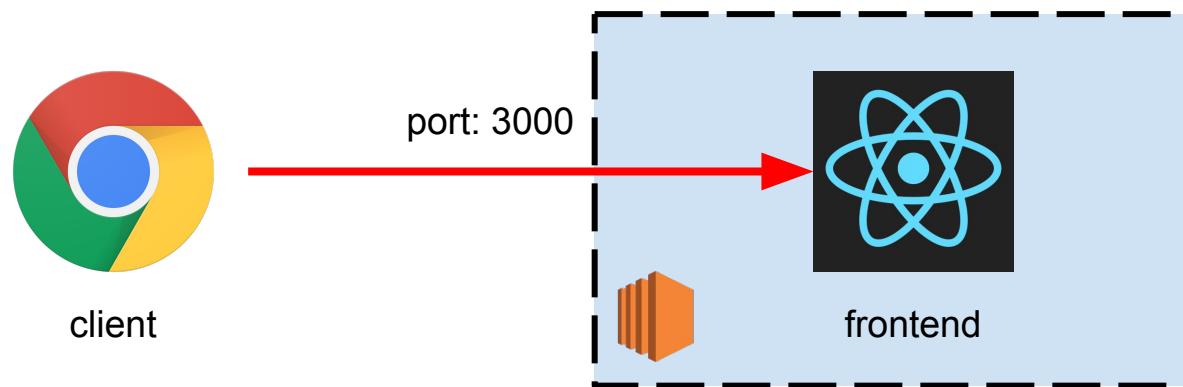
Allow Web Access

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



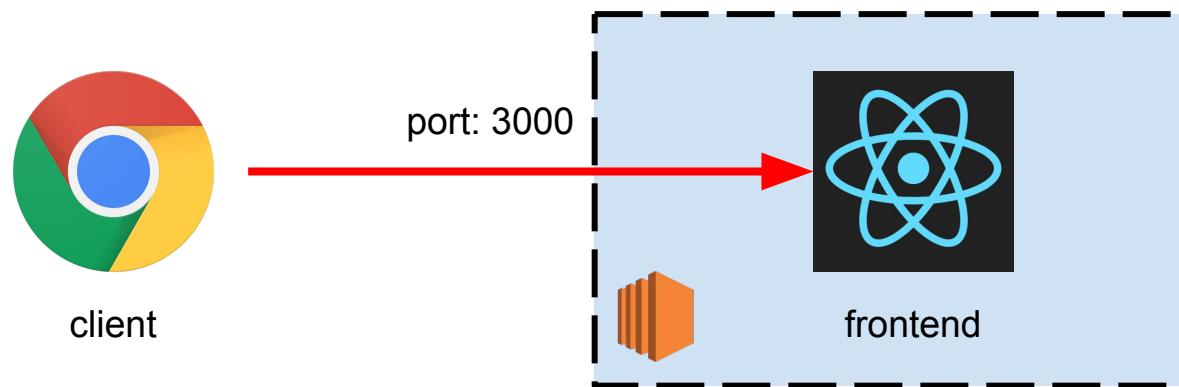
Allow Web Access

- 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

The screenshot shows the AWS Management Console interface for managing security groups. The left sidebar navigation includes:

- Images
- Elastic Block Store
- Network & Security
 - Security Groups** (highlighted with a red oval)
 - Elastic IPs
 - Placement Groups
 - Key Pairs
 - Network Interfaces
- Load Balancing
- Auto Scaling

The main content area displays the "Security Groups (1/5)" page. A search bar at the top says "Filter security groups". Below it is a table with columns: Name and Security group ID. The table lists five security groups, with the last one, "sg-78d1fe47 - default", selected (indicated by a checked checkbox). The Actions menu for this selected group includes options: View details, Edit inbound rules, Edit outbound rules, Manage tags, Manage stale rules, Copy to new security group (highlighted with a red oval), Delete security group, and a link to "security-group-for-out...". The "Create security group" button is also highlighted with a red oval.

Name	Security group ID	VPC ID
-	sg-02b89e905b7f35f1	vpc-c5b67
-	sg-09765d00c665ee25	vpc-c5b67
-	sg-0bd5b3f8f88dbc51	vpc-c5b67
-	sg-0c456ca29a3fb1b32	security-group-for-out...
<input checked="" type="checkbox"/>	sg-78d1fe47	default

At the bottom, there are tabs for "Details", "Inbound rules", "Outbound rules", and "Tags".

Allow Web Access

The screenshot shows the AWS Management Console interface for managing security groups. The top navigation bar includes the AWS logo, 'Services' dropdown, user information 'vocstartsoft/user955635=hy.cho@snu.ac.kr @ 2688-1260-2254', region 'N. Virginia', and 'Support' link.

The main section displays details for a security group named 'sg-78d1fe47 - default'. It includes fields for 'Security group name' (SWPPServerGroup), 'Description' (SSH and HTTP accesses), and 'VPC' (vpc-c5b673b8).

A red box highlights the 'Inbound rules' section, which lists three rules:

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Anywhere (0.0.0.0/0)	
Custom TCP	TCP	8000	Anywhere (0.0.0.0/0)	
Custom TCP	TCP	3000	Anywhere (0.0.0.0/0)	

At the bottom left of the highlighted area is a 'Add rule' button.

Allow Web Access

- Go to Instances

The screenshot shows the AWS Lambda Instances page. At the top left, it says "Instances (1/1) Info". Below that is a search bar with "Filter instances" and a button "Clear filters". To the right of the search bar are buttons for "Connect" and "Instance state". Further right is a "Actions" button with a dropdown menu. The "Actions" button is highlighted with a red circle. The dropdown menu includes options: "Change security groups" (which is also highlighted with a red circle), "Get windows password", "Modify IAM role", "Security" (with a sub-menu arrow), "Image and templates" (with a sub-menu arrow), and "Monitor and troubleshoot" (with a sub-menu arrow). On the far right, there are "Launch instances" and "DNS" buttons. The main table below the header shows one instance: "swpp-webser..." with Instance ID "i-01370fb3be5b967ad", status "Running", type "t2.micro", and 2/2 checks passing. There are also "Status check" and "No alarms" buttons.

Allow Web Access

EC2 > Instances > i-01370fb3be5b967ad > Change security groups

Change security groups Info

Amazon EC2 evaluates all the rules of the selected security groups to control inbound and outbound traffic to and from your instance. You can use this window to add and remove security groups.

SWPPServerGroup (sg-01387a28c0858bdad) SWPPServerGroup	
launch-wizard-2 (sg-02b89e905b7f35f12) launch-wizard-2	
launch-wizard-1 (sg-09765d00c665ee257) launch-wizard-1	
security-group-for-inbound-nfs-d-wzcd2t2u2mlk (sg-0bd5b3f8f88dbc51d) security-group-for-inbound-nfs-d-wzcd2t2u2mlk	
security-group-for-outbound-nfs-d-wzcd2t2u2mlk (sg-0c456ca29a3fb1b32) security-group-for-outbound-nfs-d-wzcd2t2u2mlk	
default (sg-78d1fe47) default	

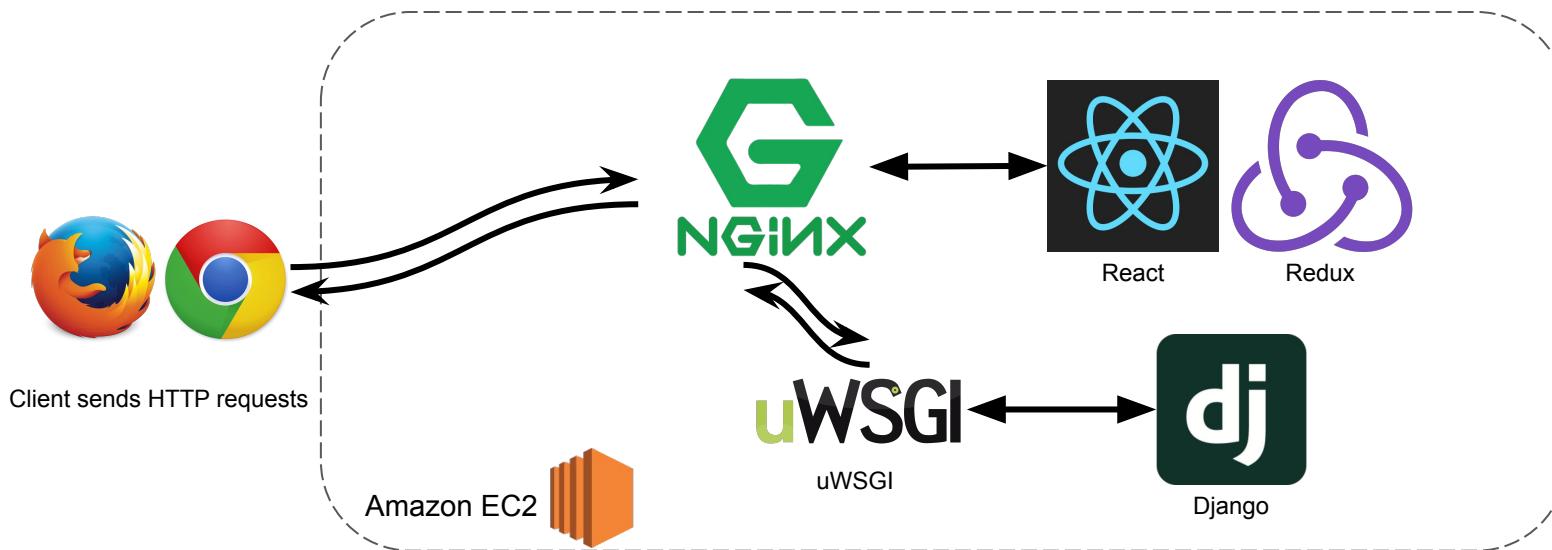
Security groups associated with the network interface (eni-0c00decaf8454f4be)

Security group name	Security group ID	
SWPPServerGroup	sg-01387a28c0858bdad	<input type="button" value="Remove"/>

Docker

Why using Docker for deploying our services?

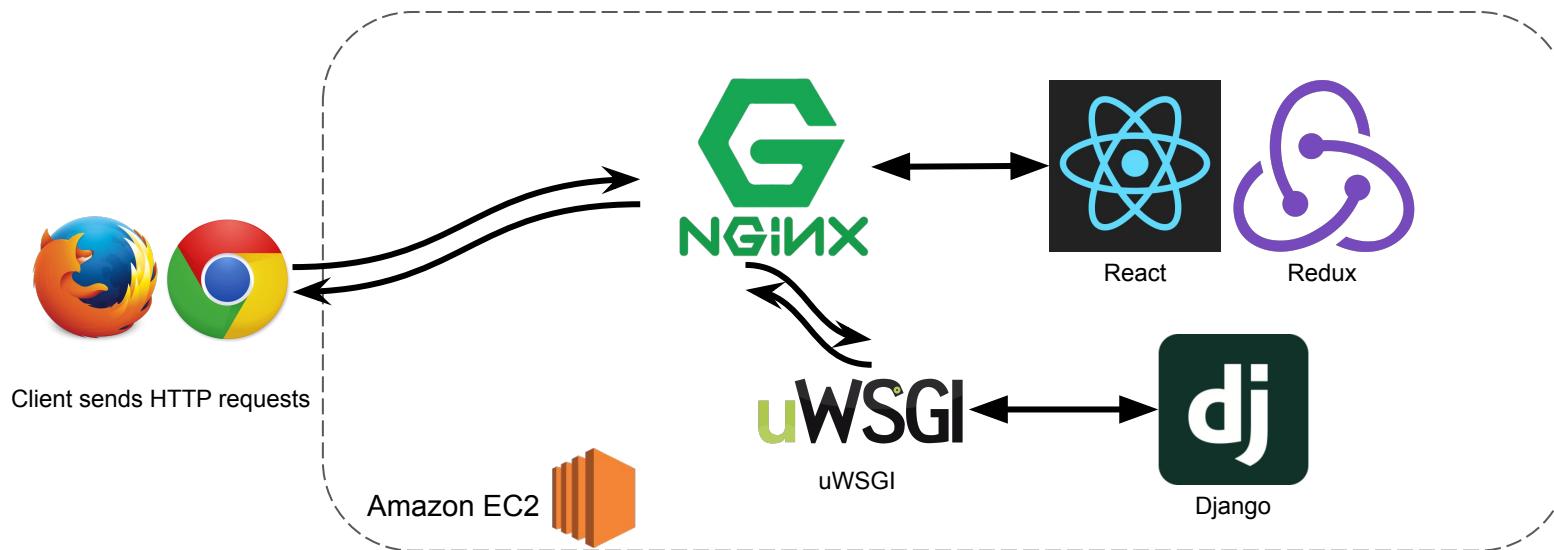
1. When deploying and maintaining a web service in a new EC2 instance
 - There is no way to check if the service works until you install all requirements and run apps.
 - It is hard to keep the instance environment consistent. (e.g. Changing the dependencies, version updates)



Why using Docker for deploying our services?

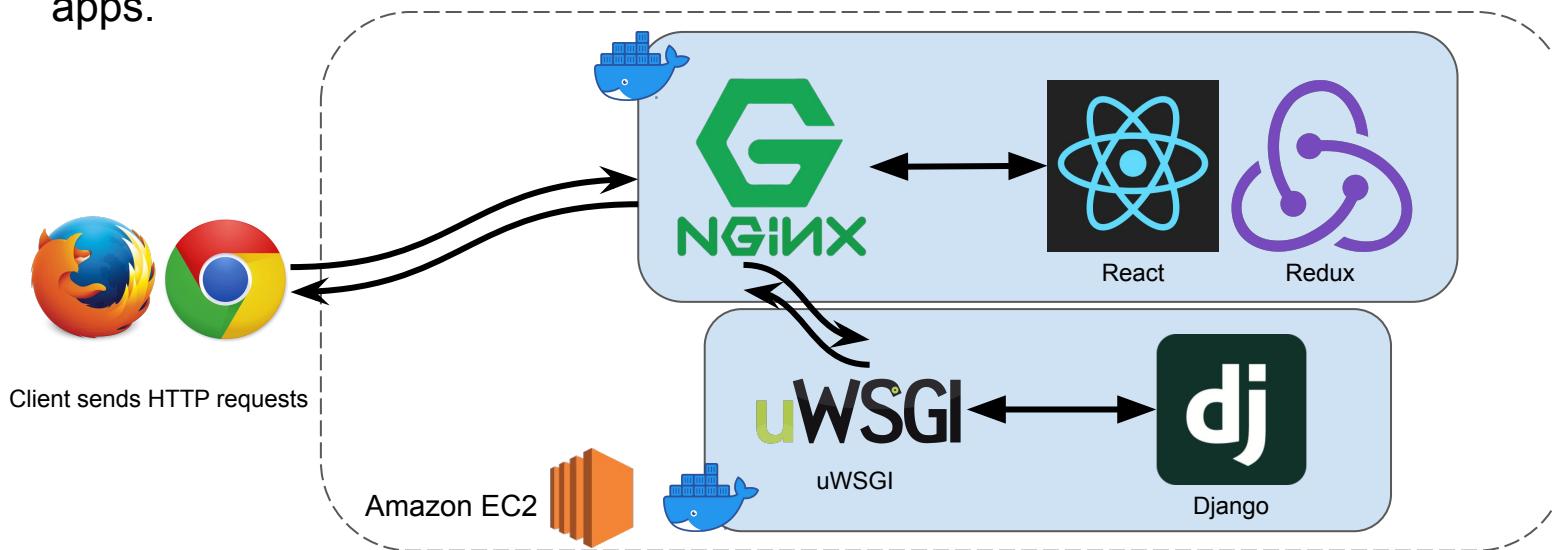
2. When restarting a new EC2 instance

- 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
```

Access

Billing and plans

Emails

Password and authentication

SSH and GPG keys

Organizations

Moderation

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)

The screenshot shows a dark-themed terminal interface within a code editor. The terminal window title is "run_docker.sh — swpp". The code editor has several tabs open, including "sol3.py", "sol4.py", "run_docker.sh", "README.md", "swpp-p10-deploy", "swpp-p10-deploy-key.pem", "run_ec2.sh", "problem8.py", and another "README.md". The "run_docker.sh" tab contains the following script:

```
$ run_docker.sh
1 docker run --rm -it \
2   --ipc=host \
3   --name "practice10" \
4   -p 0.0.0.0:3000 -p 0.0.0.0:8000:8000 \
5   -v ${PWD}:/home \
6   snuspl/swpp:practice8 \
7   /bin/bash
8
```

The terminal below shows a root shell on a Docker container:

```
root@a00e435adec1:/home# exit
ubuntu@ip-172-31-6-126:~/swpp-p10-deploy$
```

To the right, a file tree sidebar shows a directory structure with "zsh" and "bash" files.

Inside your EC2 instance

Setup - docker container

- run docker container
- Check docker container running

...

sudo docker ps

...

```
$ 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
```

- 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

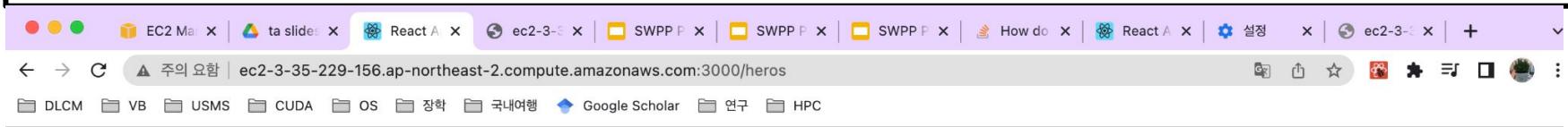
Run your FE & BE

- FE
- BE

```
cd frontend  
yarn  
yarn start
```

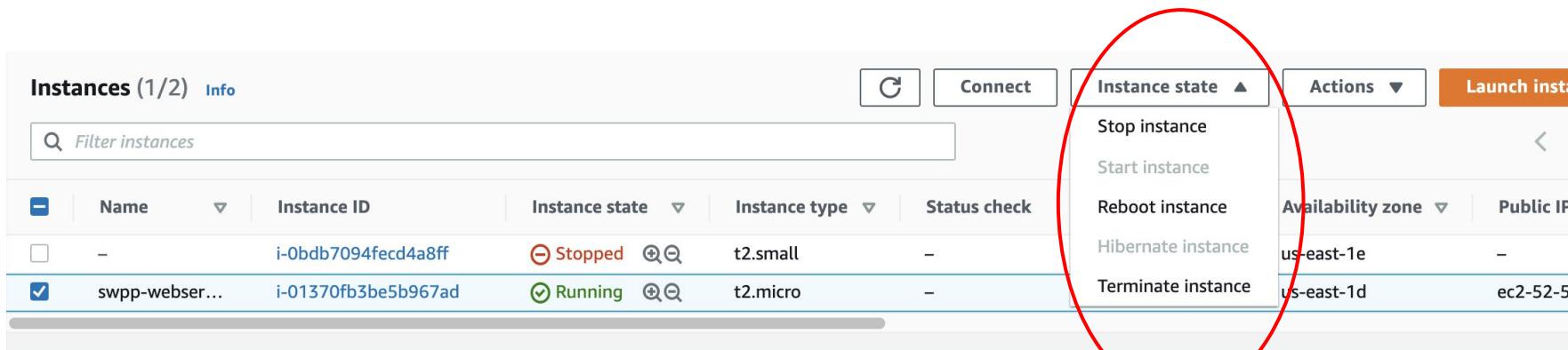
```
cd backend  
python manage.py makemigrations  
python manage.py migrate  
python manage.py runserver 0.0.0.0:8000
```

TADA !



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)

#10 - Refactoring

<Original code>
: Paste the original code to refactor.
: (Important) Add comments on the code where you will refactor, and why refactoring is needed here.

<Refactored code>
: Paste the refactored code

Name of the code piece (xx.py) and briefly explain its functionality
단답형 텍스트

Original code
장문형 텍스트

Refactored code
장문형 텍스트