

웹 서비스 구조와 AWS EC2 배포

Session 13

NEXT X LIKELION 이혁준

| 목차

1.웹서버와 웹 서비스 아키텍처

2.EC2 생성 및 기본적인 환경설정 후 장고 서버 배포

3.Nginx와 Gunicorn 연결

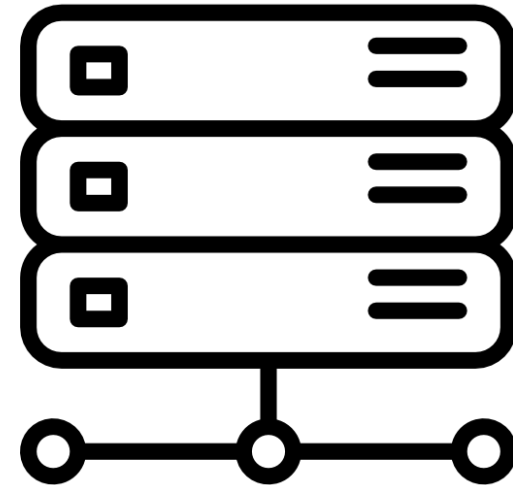
4.심화: Docker

서버란?

서버 = 컴퓨터, 클라이언트에게 네트워크를 통해 정보나 서비스를 제공하는 컴퓨터 시스템

즉, 당신의 핸드폰도 서버가 될 수 있음.

하지만, 썬 서버는 다른 핸드폰, 컴퓨터 등과 구분되는 특징이 있음 => 썸, 전기 덜먹음, 크기 작음

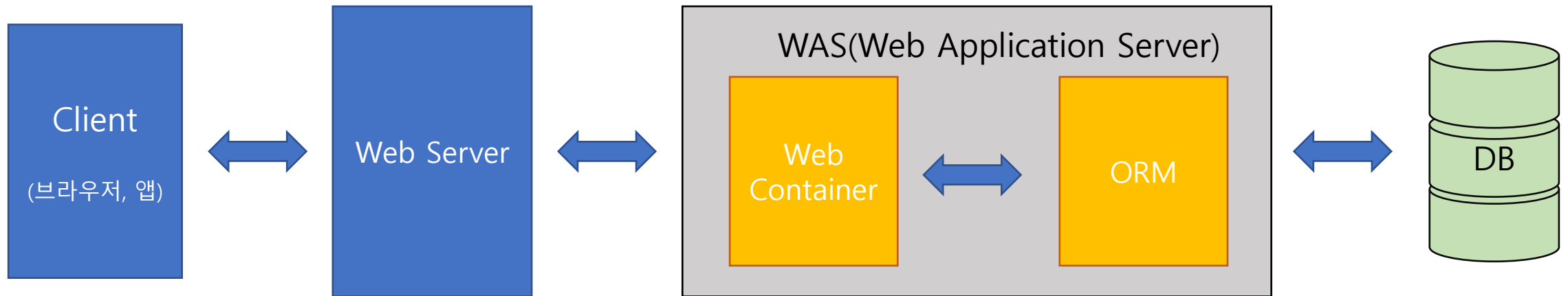


| 서버

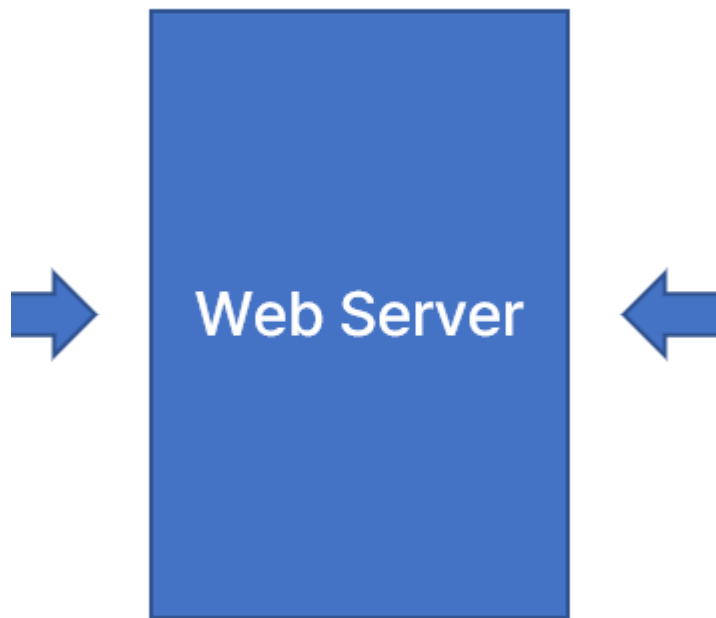


| 웹 서비스 아키텍처

일반적인 웹 서비스 아키텍처는 간략하게 표현하면 이렇게 생겼다. 하나씩 뜯어보자.



| WS(Web Server)



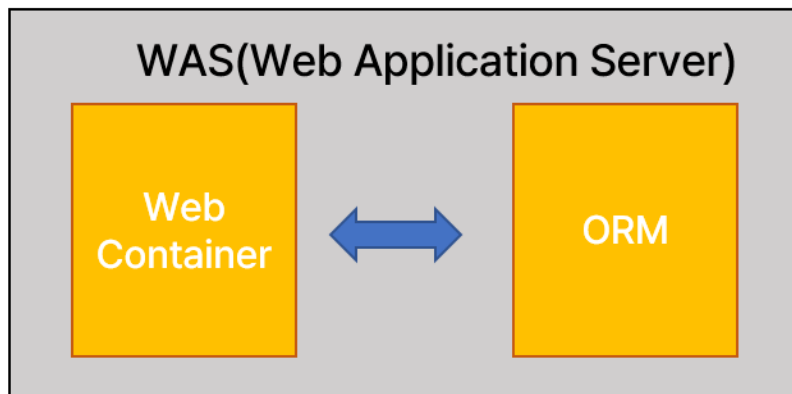
웹서버(WS, Web Server)는 사용자가 어떠한 페이지를 요청했을 때 정적(static) 콘텐츠를 제공하는 서버다.

*정적 콘텐츠: 단순 HTML, CSS, javascript 파일, 이미지, 파일 등 DB 또는 서비스 로직을 거치지 않고 즉시 응답 가능한 콘텐츠들.

-> 흔히 django에서는 static 폴더 안에 들어가는 파일들.



WS(Web Server)



웹 애플리케이션 서버(WAS): 웹서버 + 웹 컨테이너.
정적 파일 제공 + 동적 로직 처리 가능
(Django프로젝트가 웹 컨테이너에서 돌아감)

WAS는 자체적으로 웹서버를 가지고 있으므로 사실은 이렇게 생겼다.

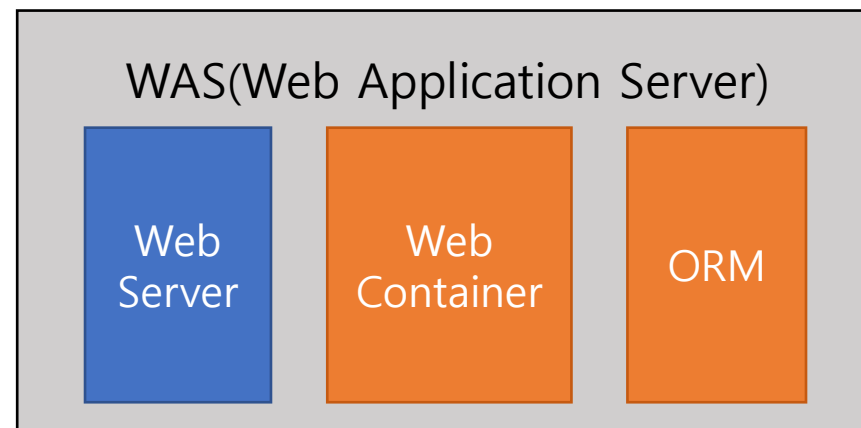
Django도 자체적으로 WAS를 탑재하고 있다.

``python manage.py runserver``

그래서 개발할 때 localhost에서 전체 서비스를 완벽히 볼 수 있는 것.

```
26 # SECURITY WARNING: don't  
27 DEBUG = False  
28 ☐ False
```

그러나 배포 환경에서는 보통 WAS가 가지고 있는 웹서버를 사용하지 않는다.



WAS만 쓰면 안되나요?

Django 공식 문서에서는 친절하게 자체 WAS에 대해 이렇게 말하고 있다.

DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests. (And that's how it's gonna stay. We're in the business of making web frameworks, not web servers, so improving this server to be able to handle a production environment is outside the scope of Django.)

PRODUCTION SETTING에서 이 서버를 사용하지 마십시오. 보안 감사나 성능 테스트를 거치지 않았습니다. (그리고 그렇게 될 것입니다. 우리는 웹 서버가 아닌 웹 프레임워크를 만드는 사업을 하고 있으므로 이 서버를 프로덕션 환경을 처리할 수 있도록 개선하는 것은 Django의 범위를 벗어납니다.)

WAS만 쓰면 안되나요?

WAS는 바쁘고, 서버컴퓨터는 생각보다 느리다

서버컴퓨터는 기본적으로 저전력이 제일 중요하다. 성능 높인다고 전기 많이 먹이면 그게 기상청에 있는 슈퍼컴퓨터다...

서버 컴퓨터에서 DB처리 로직을 담당하는 WAS에 정적 파일 처리까지 맡기면 조금만 부하가 걸려도 터진다. -> 대량 요청이나 동시 요청을 제대로 처리하지 못한다.

+

별도의 웹서버는 WAS가 제공하지 못하는 다양한 기능을 제공한다.
(도메인 인증, 로드 밸런싱, 프록시 패스 등등)

+

WAS는 일반 사용자에게 공개될 필요가 없다. 분리하는 것이 보안상 훨씬 더 안전하다.

심지어 django의 WAS는 https 프로토콜도 지원하지 않는다.

| WS & WAS

웹서버



웹 애플리케이션 서버
(WAS)

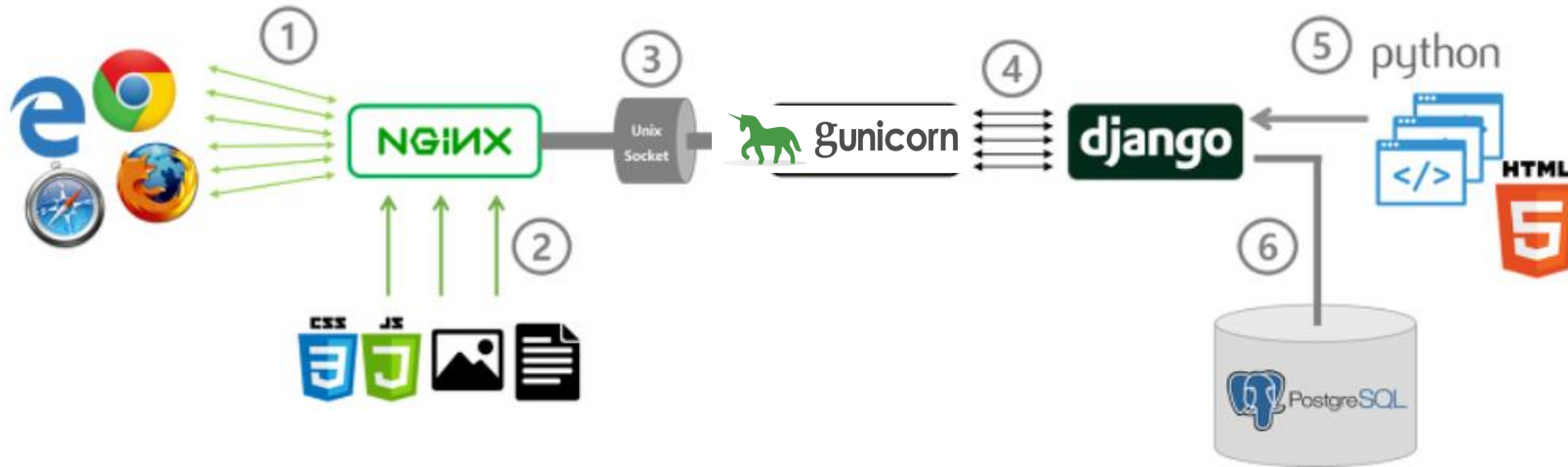


Apache Tomcat



웹 서비스 아키텍처 요약 (feat. Django)

클라이언트 -> 웹 서버 -> WSGI(=WAS) -> 웹 컨테이너 내부 프레임워크(ex. django) -> ORM -> DB



부록 - 왜 nginx인가 (이 서버는 1998년 C10K에서 시작되어...)



: 이벤트 기반,
하나의 프로세스가 여러 채널의 커넥션을 담당.
요청이 오면 큐 정렬 후 비동기로 처리



: 프로세스 기반,
하나의 프로세스가 하나의 커넥션을 담당.
프로세스가 많아지기에 메모리가 많이 소모되고, 요청 처리마다 프로세스를 바꿔가며 작업해야 하므로 부하가 쉽게 걸림



EC2 배포 실습

EC2
(Elastic Compute Cloud)



클라이언트



Web Server



WAS



gunicorn



실습

AWS EC2 콘솔 접속 – 리전 확인(ap-northeast-2)필수!! – مم바이나 버지니아로 등으로 설정될 때가 있는데 모르고 진행하면 나중에 머리아픉니다!!

<https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home>

The screenshot shows the AWS Management Console interface for the ap-northeast-2 region. The main content area is titled '리소스' (Resources) and displays a summary of EC2 resources. A notification banner at the bottom promotes the AWS Launch Wizard for SQL Server. On the right, a dropdown menu is open, showing a list of available AWS regions with 'ap-northeast-2' highlighted.

리소스	개수
인스턴스(실행 중)	2
로드 밸런서	1
배치 그룹	0
보안 그룹	5
볼륨	2
스냅샷	0
인스턴스	2
전용 호스트	0
키 페어	4
탄력적 IP	0

인스턴스 시작 서비스 상태

지역	리전
미국 동부 (버지니아 북부)	us-east-1
미국 동부 (오하이오)	us-east-2
미국 서부 (캘리포니아)	us-west-1
미국 서부 (오레곤)	us-west-2
아프리카 (케이프타운)	af-south-1
아시아 태평양 (홍콩)	ap-east-1
아시아 태평양 (자카르타)	ap-southeast-3
아시아 태평양 (مم바이)	ap-south-1
아시아 태평양 (오사카)	ap-northeast-3
아시아 태평양 (서울)	ap-northeast-2
아시아 태평양 (싱가포르)	ap-southeast-1

| 실습

인스턴스 탭에서 인스턴스 시작

New EC2 Experience
Tell us what you think

EC2 대시보드

EC2 글로벌 보기

이벤트

제한

▼ 인스턴스

인스턴스

인스턴스 유형

인스턴스 (1) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

< 1 > ⚙

<input type="checkbox"/>	Name ▼	인스턴스 ID	인스턴스 상태 ▼	인스턴스 유형 ▼	상태 검사	경보 상태	가용 영역 ▼	퍼블릭 IPv4 DI
<input type="checkbox"/>	gpt-cs-bot	i-0fa9d62c80a7d4dbb	🟢 실행 중 🔍	t2.micro	🟢 2/2개 검사 통과...	경보 없음 +	ap-northeast-2c	ec2-54-180-11

인스턴스 선택

만약 UI가 캡처된 이미지와 다른 경우 New EC2 Experience 활성화 시켜주세요!

Session 18

NEXT X LIKELION

실습

이름

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

최근 사용

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

Sl

더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-0c9c942bd7bf113a2 (64비트(x86)) / ami-00fdfe418c69b624a (64비트(Arm))

가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

선명

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16

아키텍처

AMI ID

ami-0c9c942bd7bf113a2

화인된 공급 업체

이름은 크게 중요하지 않아요. Ubuntu Server 22.04 LTS 선택하세요.

실습

t2.micro 선택 후 "새 키 페어 생성" 클릭(첫 AWS인
가)

▼ 인스턴스 유형 정보

인스턴스 유형

t2.micro	프리 티어 사용 가능		
패밀리: t2	1 vCPU	1 GiB 메모리	현재 세대: true
온디맨드 RHEL 요금: 0.0744 USD 시간당	온디맨드 Linux 요금: 0.0144 USD 시간당		
온디맨드 SUSE 요금: 0.0144 USD 시간당	온디맨드 Windows 요금: 0.019 USD 시간당		

모든 세대

인스턴스 유형 비교

▼ 키 페어(로그인) 정보

키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요.

키 페어 이름 - 필수

선택

새 키 페어 생성

키 페어 생성

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

deploy

이름에는 최대 255개의 ASCII 문자를 포함할 수 있습니다. 앞 또는 뒤에 공백을 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

⚠ When prompted, store the private key in a secure and accessible location on your computer. 나중에 인스턴스에 연결할 때 필요합니다. 자세히 알아보기

취소

키 페어 생성

키페어는 이렇게 세팅하고 생성하고 다운로드해주세요. 저는 보통 프로젝트폴더 바로 상위에 다운 받습니다.

주의: 키페어는 서버 접근권한을 부여하는 파일로, 생성 당시에만 다운로드가 가능하고 이후에는 불가능합니다.

잃어버리면 영영 그 서버에는 접속할 수 없고, 만일 github같은 곳에 실수로 올려 유출되면 AWS 본사에서 영어로 경고전화 걸려옵니다.

실습

네트워크 설정에서 편집을 클릭하고 적당한 이름을 정

▼ 네트워크 설정 정보

VPC - 필수 정보

vpc-0b427b61079678f91 (기본값) ↕

172.31.0.0/16

서브넷 정보

기본 설정 없음 ▼ ↕ 새 서브넷 생성

퍼블릭 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

보안 그룹 이름 - 필수

launch-wizard-1

이 보안 그룹은 모든 네트워크 인터페이스에 추가됩니다. 보안 그룹을 만든 후에는 이름을 편집할 수 없습니다. 최대 길이는 255자입니다. 유효한 문자는 소문자, 대문자, 숫자, 공백 및 .-:/()#,@!+=&{}~*입니다.

설명 - 필수 정보

launch-wizard-1 created 2023-05-28T21:25:02.459Z

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0)

제거

실습

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP 22 0.0.0.0/0) [제거]

유형 정보	프로토콜 정보	포트 범위 정보
ssh	TCP	22

소스 유형 정보: 위치 무관

원본 정보: 0.0.0.0/0

설명 - optional 정보: 예: 관리자 데스크톱용 SSH

▼ 보안 그룹 규칙 2 (TCP 80 0.0.0.0/0) [제거]

유형 정보	프로토콜 정보	포트 범위 정보
HTTP	TCP	80

소스 유형 정보: 사용자 지정

원본 정보: 0.0.0.0/0

설명 - optional 정보: 예: 관리자 데스크톱용 SSH

▼ 보안 그룹 규칙 3 (TCP 443 0.0.0.0/0) [제거]

유형 정보	프로토콜 정보	포트 범위 정보
HTTPS	TCP	443

소스 유형 정보: 사용자 지정

원본 정보: 0.0.0.0/0

설명 - optional 정보: 예: 관리자 데스크톱용 SSH

“보안 그룹 규칙 추가” 버튼을 누르고 인바운드 규칙을 추가합니다.

인바운드 규칙은 서버에 접속 가능한 포트를 제한합니다. ssh(22번), HTTP(80번), HTTPS(443번), 사용자 지정(8000번) 포트를 열어주세요. 일반적으로는 22, 80, 443 포트만 열지만 저희는 실습 중간에 테스트를 위해 8000번 포트를 함께 열겠습니다.

모든 소스는 0.0.0.0/0으로 설정해주세요. 전체 IP에 대해서 접근을 허용한다는 의미입니다.

실습

▼ 스토리지 구성 [정보](#)

[어드밴스드](#)

1x GiB 루트 볼륨 (암호화되지 않음)

❗

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

×

새 볼륨 추가

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](#)

스토리지 구성은 기본값 그대로 8GiB로 설정합니다. 최대 30GiB까지 가능하지만, 일반적으로 EC2 서버 용량이 그렇게까지 많이 필요하지는 않습니다.

Session 18

NEXT X LIKELION

실습

▼ 요약

인스턴스 개수 정보

1

소프트웨어 이미지(AMI)

Canonical, Ubuntu, 22.04 LTS, ...[더 보기](#)
ami-0c9c942bd7bf113a2

가상 서버 유형(인스턴스 유형)

t2.micro

방화벽(보안 그룹)

새 보안 그룹

스토리지(볼륨)

1개의 볼륨 – 8GiB

취소

인스턴스 시작


[명령 검토](#)

이 구성으로 "인스턴스 시작"을 클릭해주세요

실습

인스턴스가 시작되면 모든 인스턴스 보기 클릭

[EC2](#) > [인스턴스](#) > [인스턴스 시작](#)

 **성공**
인스턴스를 시작했습니다. (i-0aae06d55c822c674)

[▶ 로그 시작](#)

다음 단계

결제 및 프리 티어 사용 알림 생성

비용을 관리하고 높은 금액의 청구서를 방지하려면 결제 및 프리 티어 사용 임계값에 대한 이메일 알림을 설정합니다.

[결제 알림 생성](#)

인스턴스에 연결

인스턴스가 실행되면 로컬 컴퓨터에서 인스턴스에 로그인합니다.

[인스턴스에 연결](#)

[자세히 알아보기](#)

RDS 데이터베이스 연결

EC2 인스턴스와 데이터베이스 간의 트래픽 흐름을 허용하도록 연결을 구성합니다.

[RDS 데이터베이스 연결](#)

[새 RDS 데이터베이스 생성](#)

[자세히 알아보기](#)

[모든 인스턴스 보기](#)

Session 18

NEXT X LIKELION

실습

인스턴스 선택 후 연결 버튼을 클릭해서 SSH 클라이언트 탭을 참고해 다운받은 키페어가 있는 경로에서 git bash를 켭니다
chmod 400 <키페어 이름>
명령 실행 후 ssh -i로 시작하는 명령을 통해 EC2에 접속합니다.

인스턴스에 연결 정보
다음 옵션 중 하나를 사용하여 인스턴스 i-050673cac0c2e0ec4에 연결

EC2 인스턴스 연결

Session Manager

SSH 클라이언트

EC2 직렬 콘솔

인스턴스 ID
i-050673cac0c2e0ec4

- SSH 클라이언트를 엽니다.
- 프라이빗 키 파일을 찾습니다. 이 인스턴스를 시작하는 데 사용되는 키는 deploy_test.pem입니다.
- 필요한 경우 이 명령을 실행하여 키를 공개적으로 볼 수 없도록 합니다.
chmod 400 deploy_test.pem
- 퍼블릭 DNS을(를) 사용하여 인스턴스에 연결:
ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com

예:
ssh -i "deploy_test.pem" ubuntu@ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com

참고: 대부분의 경우 추정된 사용자 이름은 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하십시오.

```
this key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? |
```

만약 이 문구가 뜨면 yes를 입력합니다

실습: ec2

EC2에 연결되었습니다!

ubuntu에서 ~디렉토리는 home/ubuntu/ 를 지칭합니다(접속 유저에 따라 다름)

```
$ sudo apt-get update
```

```
$ sudo apt-get dist-upgrade
```

```
$ sudo apt-get install python3-pip
```

```
$ sudo apt-get install python3.12-venv
```

```
* Support:      https://ubuntu.com/advantage

System information as of Wed Aug 10 22:05:43 UTC 2022

System load:  0.0              Processes:      94
Usage of /:   16.1% of 7.58GB   Users logged in: 0
Memory usage: 18%              IP address for eth0: 172.31.10.134
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-172-31-10-134:~$ |
```


실습: ec2

srv 폴더의 소유자를 바꿉니다. srv 폴더는 우리의 프로젝트 파일들이 들어갈 자리입니다

```
$ sudo chown -R ubuntu:ubuntu /srv/ (R 대소문자 주의)
```

```
$ cd /
```

```
$ ls -al
```

```
ubuntu@ip-172-31-10-134:~$ sudo chown -R ubuntu:ubuntu /srv/
ubuntu@ip-172-31-10-134:~$ cd /
ubuntu@ip-172-31-10-134:/ $ ls -al
total 88
drwxr-xr-x 23 root root 4096 Aug 10 21:58 .
drwxr-xr-x 23 root root 4096 Aug 10 21:58 ..
drwxr-xr-x  2 root root 4096 Jun 10 17:39 bin
drwxr-xr-x  4 root root 4096 Jun 10 17:41 boot
drwxr-xr-x 15 root root 3160 Aug 10 21:58 dev
drwxr-xr-x 91 root root 4096 Aug 10 21:58 etc
drwxr-xr-x  3 root root 4096 Aug 10 21:58 home
lrwxrwxrwx  1 root root    30 Jun 10 17:41 initrd.img -> boot/initrd.img-5.
4.0-1078-aws
lrwxrwxrwx  1 root root    30 Jun 10 17:41 initrd.img.old -> boot/initrd.im
g-5.4.0-1078-aws
drwxr-xr-x 20 root root 4096 Jun 10 17:37 lib
drwxr-xr-x  2 root root 4096 Jun 10 17:36 lib64
drwx----- 2 root root 16384 Jun 10 17:38 lost+found
drwxr-xr-x  2 root root 4096 Jun 10 17:35 media
drwxr-xr-x  2 root root 4096 Jun 10 17:35 mnt
drwxr-xr-x  2 root root 4096 Jun 10 17:35 opt
dr-xr-xr-x 153 root root    0 Aug 10 21:58 proc
drwx----- 4 root root 4096 Aug 10 21:59 root
drwxr-xr-x 24 root root 860 Aug 10 22:05 run
drwxr-xr-x  2 root root 4096 Jun 10 17:38 sbin
drwxr-xr-x  6 root root 4096 Aug 10 21:59 snap
drwxr-xr-x  2 ubuntu ubuntu 4096 Jun 10 17:35 srv
dr-xr-xr-x 13 root root    0 Aug 10 21:58 sys
drwxrwxrwt  9 root root 4096 Aug 10 22:05 tmp
drwxr-xr-x 10 root root 4096 Jun 10 17:35 usr
drwxr-xr-x 13 root root 4096 Jun 10 17:38 var
lrwxrwxrwx  1 root root    27 Jun 10 17:41 vmlinuz -> boot/vmlinuz-5.4.0-10
78-aws
```

*왜 srv 폴더인가요?

우분투에서는 디렉토리 별로 사용하는 용도가 어느정도 정해져 있습니다.

괜히 다른 디렉토리에서 배포를 진행하고 소유자를 변경했다가 시스템까지 건드릴 수 있습니다.

srv/가 이러한 문제에서 가장 안전합니다.

| 실습: ec2

프로젝트 파일들을 ec2에 다운받습니다

```
$ cd /srv
```

```
$ git clone <레포지토리 주소>
```

<https://github.com/newxxson/next-session-11-hw.git>

```
ubuntu@ip-172-31-10-134:/$ cd /srv
ubuntu@ip-172-31-10-134:/srv$ git clone https://github.com/Kim-Jiseong/2022-Session19Deploy.git
Cloning into '2022-Session19Deploy'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 25 (delta 1), reused 25 (delta 1), pack-reused 0
Unpacking objects: 100% (25/25), done.
ubuntu@ip-172-31-10-134:/srv$ ls
2022-Session19Deploy
ubuntu@ip-172-31-10-134:/srv$ |
```

| 실습: ec2

Python용 가상 환경을 정의합니다.

```
$cd /srv/next-session-11-hw
```

```
$python3 -m venv next_13_venv
```

```
$source next_13_venv/bin/activate && which python
```

```
$pip install django
```

```
ubuntu@ip-172-31-10-134:/$ cd /srv
ubuntu@ip-172-31-10-134:/srv$ git clone https://github.com/Kim-Jiseong/2022-Session19Deploy.git
Cloning into '2022-Session19Deploy'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 25 (delta 1), reused 25 (delta 1), pack-reused 0
Unpacking objects: 100% (25/25), done.
ubuntu@ip-172-31-10-134:/srv$ ls
2022-Session19Deploy
ubuntu@ip-172-31-10-134:/srv$ |
```

실습: ec2

마이그레이션 하고 서버를 켜봅시다

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

```
$ python manage.py runserver 0:8000    0:8000 중요합니다.
```

```
[[A^C(myvenv) ubuntu@ip-172-31-10-134:/srv/2022-Session19Deploy/mysite$ python3
e.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

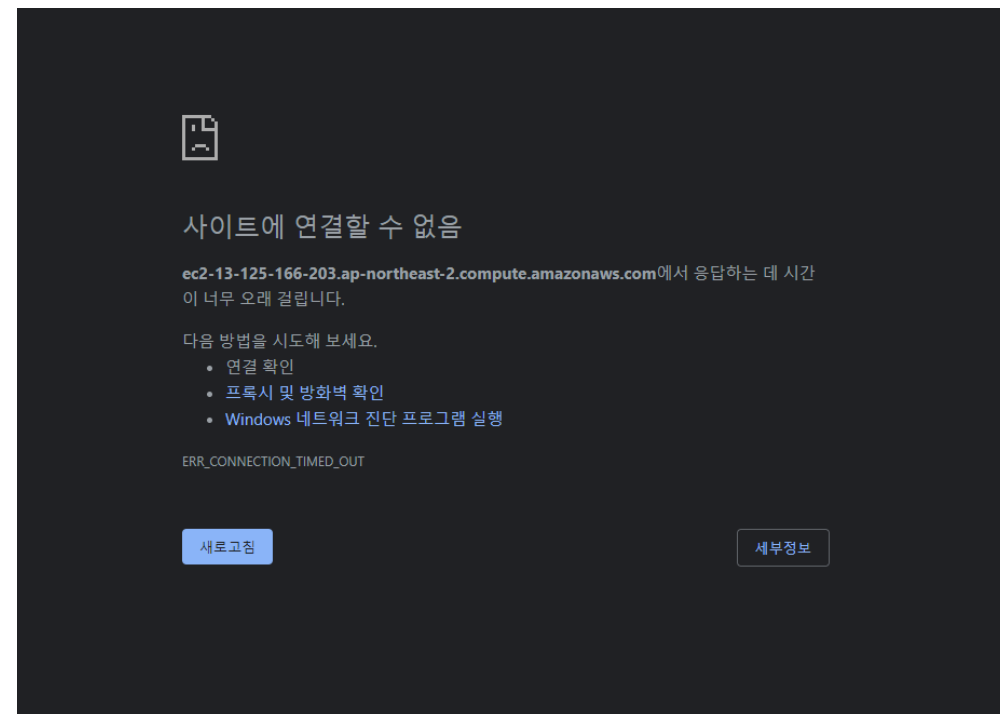
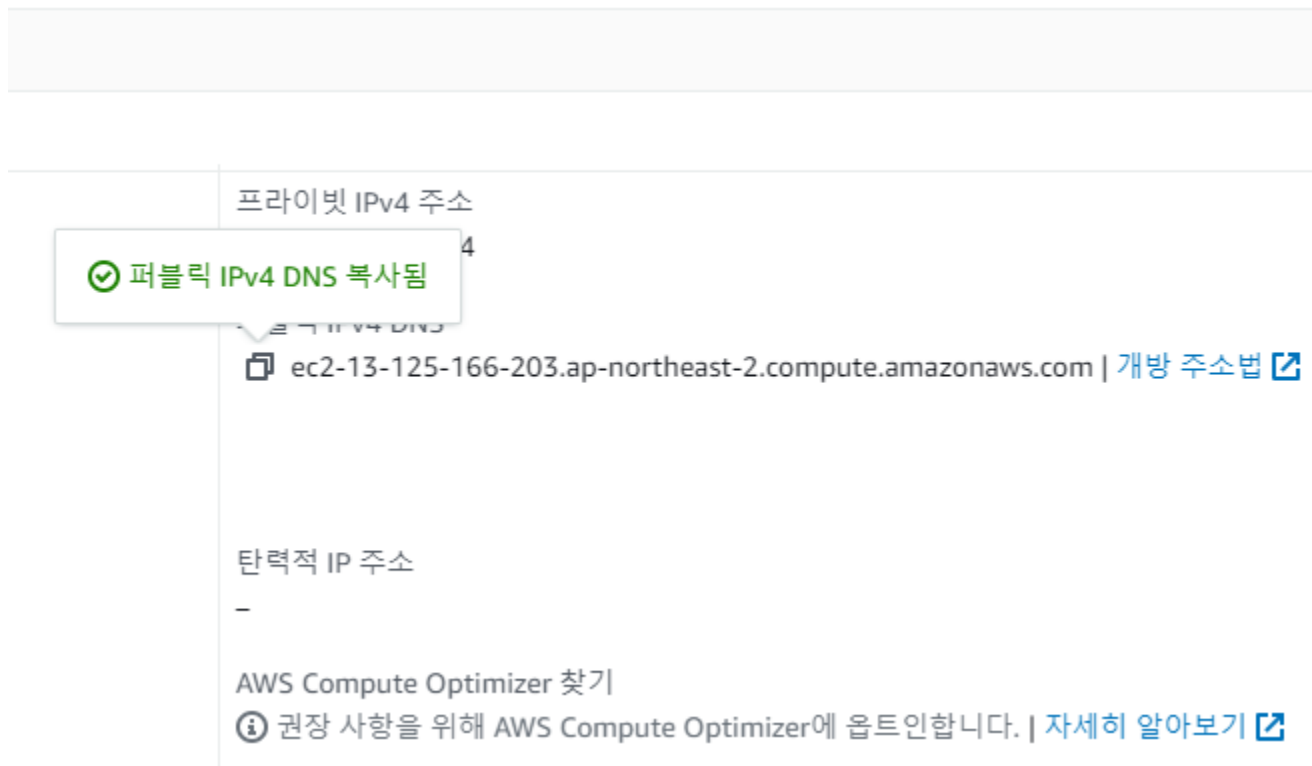
System check identified no issues (0 silenced).
August 11, 2022 - 07:46:51
Django version 3.2.15, using settings 'mysite.settings'
Starting development server at http://0:8000/
Quit the server with CONTROL-C.
```

실습: local

이제 접속해봅시다

EC2 콘솔에 나와있는 퍼블릭 DNS를 복사한 뒤 <퍼블릭 DNS>:8000으로 접속하면 됩니다

만약 아까 보안그룹을 만들 때 8000포트를 뚫어놓지 않았다면 아래처럼 접속이 안될겁니다.



실습: local-troubleshooting

ec2콘솔의 왼쪽을 잘 보면 보안그룹이라는 탭이 있습니다. 여기서 아까 인스턴스 만들 때 사용한 보안그룹을 체크하고 인바운드 규칙 편집을 클릭합니다

The screenshot shows the AWS Management Console interface. On the left sidebar, the 'Network & Security' section is expanded, and 'Security Groups' is selected. The main panel displays the 'Security Groups (1/6) 정보' page. A table lists several security groups, with 'session19' (ID: sg-08263149039440b7b) selected. Below the table, the 'Inbound Rules (2)' tab is active, showing a table of inbound rules for the selected security group.

Name	보안 그룹 ID	보안 그룹 이름	VPC ID	설명	소유
-	sg-0091c682a3f4b1644	PrayForGrade	vpc-0b427b61079678f91	PrayForGrade(launch-...	694!
-	sg-01417be1ce64ad2b7	next_saza_shop	vpc-0b427b61079678f91	launch-wizard-1 create...	694!
<input checked="" type="checkbox"/>	sg-08263149039440b7b	session19	vpc-0b427b61079678f91	launch-wizard-1 create...	694!
-	sg-0984528710263ed5a	default	vpc-0b427b61079678f91	default VPC security gr...	694!
-	sg-0b5e7a907a80dd7b2	CodokDeploy	vpc-0b427b61079678f91	CodokDeploy_created ...	694!
-	sg-0f9eb721db812bff8	SNUXKU_Deploy	vpc-0b427b61079678f91	launch-wizard-1 create...	694!

Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트
-	sgr-0f42e3fbe05099a01	IPv4	HTTP	TCP	80
-	sgr-0ad30d79d4f04ff18	IPv4	SSH	TCP	22

실습: local

애 뜯겁니다. 정상입니다.

```
DisallowedHost at /  
Invalid HTTP_HOST header: 'ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com:8000'. You may need to add 'ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com' to ALLOWED_HOSTS.  
  
Request Method: GET  
Request URL: http://ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com:8000/  
Django Version: 3.2.15  
Exception Type: DisallowedHost  
Exception Value: Host is IP address: 'ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com:8000'. You may need to add 'ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com' to ALLOWED_HOSTS.  
Exception Location: /home/ubuntu/myenv/lib/python3.8/site-packages/django/http/request.py, line 149, in get_host  
Python Executable: /home/ubuntu/myenv/bin/python3  
Python Version: 3.8.9  
Python Path: ['/usr/local/lib/python3.8/dist-packages', '/usr/local/lib/python3.8', '/usr/local/lib/python3.8/site-packages', '/home/ubuntu/myenv/lib/python3.8/site-packages']  
Server time: Thu, 11 Aug 2022 07:59:26 +0000
```

Traceback [Switch to copy-and-paste view](#)

```
Home/ubuntu/myenv/lib/python3.8/site-packages/django/core/handlers/exception.py, line 47, in inner  
47:         response = get_response(request)  
  
▶ Local vars  
  
Home/ubuntu/myenv/lib/python3.8/site-packages/django/utils/decorators.py, line 116, in _wrapper__  
116:         response = self.process_request(request)  
  
▶ Local vars  
  
Home/ubuntu/myenv/lib/python3.8/site-packages/django/middleware/common.py, line 48, in process_request  
48:         host = request.get_host()  
  
▶ Local vars  
  
Home/ubuntu/myenv/lib/python3.8/site-packages/django/http/request.py, line 149, in get_host  
149:     raise DisallowedHost(msg)
```

Request Information

USER [unable to retrieve the current user]

GET No GET data

POST No POST data

FILES No FILES data

COOKIES No cookie data

META

Variable	Value
CONTENT_LENGTH	
CONTENT_TYPE	'text/plain'
ORIG_SESSION_ID_ADDRESS	'unix:path=/run/user/1000/bus'
CWAVE_GETTEXT_MODULE	'python.sartips'
GATEWAY_INTERFACE	'CGI/1.1'
HOME	/home/ubuntu
HTTP_ACCEPT	'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8,application/javascript;q=0.9'
HTTP_ACCEPT_ENCODING	'gzip, deflate'
HTTP_ACCEPT_LANGUAGE	'en-US;q=0.9,en-GB;q=0.8,en;q=0.7,it;q=0.6'
HTTP_COOKIE_CONTROL	'no-cookies'
HTTP_CONNECTION	'keep-alive'
HTTP_HOST	'ec2-13-125-166-203.ap-northeast-2.compute.amazonaws.com:8000'

실습: local

로컬에서 settings.py에 ALLOWED_HOSTS에 .ap-northeast-2.compute.amazonaws.com 추가합니다.

접속하는 도메인이 .ap-northeast-2.compute.amazonaws.com으로 끝나는 접속은 허용한다는 뜻입니다.

다른 도메인을 붙이면 당연히 그 도메인을 적어주셔야 합니다.

귀찮으면 "*" 써도 되는데 이러면 아무나 접속할 수 있어서 보안상 매우 취약하니 개발할 때 말고는 쓰지 마세요

일반적인 경우라면 이때, local에서 수정을 한 뒤 git에 푸쉬하고 다시 서버에서 pull을 받아야합니다.

하지만 귀찮죠?

```
29 ALLOWED_HOSTS = [  
30     "localhost",  
31     "127.0.0.1",  
32     ".ap-northeast-2.compute.amazonaws.com",  
33 ]
```


실습: local

직접 수정을 합니다.

```
$ cd base_auth
```

```
$ sudo nano settings.py 하고 아래 추가하기 + ctrl x + y
```

```
29  ALLOWED_HOSTS = [  
30      "localhost",  
31      "127.0.0.1",  
32      ".ap-northeast-2.compute.amazonaws.com",  
33  ]
```

```
$ cd ..
```

```
$ python manage.py runserver 0:8000
```

예쁘긴 한데 아직 django WAS로 실행한 상태

NEXT-LIKELION

[New](#)

[Home](#)

[로그인](#)

[회원가입](#)

HOME!

제목: z

내용 : zzzzz

제목: testing

내용 : testing

제목: tesing my page

내용 : testing

| Gunicorn 추가하기

```
$ pip install gunicorn
```

//manage.py 랑 같은 층위에서

```
$ gunicorn -bind 0:8000 base_auth.wsgi:application
```

```
^C(next_13_venv) ubuntu@ip-172-31-13-173:/srv/next-session-11-hw/base_auth$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.5/53.5 kB 7.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
```

```
(next_13_venv) ubuntu@ip-172-31-13-173:/srv/next-session-11-hw/base_auth$ gunicorn --bind 0:8000 base_auth.wsgi:application
[2024-04-27 11:01:45 +0000] [2615] [INFO] Starting gunicorn 22.0.0
[2024-04-27 11:01:45 +0000] [2615] [INFO] Listening at: http://0.0.0.0:8000 (2615)
[2024-04-27 11:01:45 +0000] [2615] [INFO] Using worker: sync
[2024-04-27 11:01:45 +0000] [2616] [INFO] Booting worker with pid: 2616
```



NEXT-LIKELION

- [New](#)
- [Home](#)
- [로그인](#)
- [회원가입](#)

Home!

[제목: z](#)

[내용: zzzzz](#)

[제목: testing](#)

[내용: testing](#)

[제목: tesing my_page](#)

[내용: testing](#)

[글 쓰러 가기!](#)

더 이상 django가 자동으로 제공하고 관리해주는 WSGI에서 벗어나 Gunicorn이 제공하는 WSGI를 사용하므로 /static 자동 매핑을 지원받지 못해 css가 다 날아갑니다.

그리고, 아직 gunicorn을 손수 실행해줘야 하여 터미널을 닫게 된다면 서버도 같이 날아가게 됩니다.

Gunicorn이 백그라운드에서 계속 실행될 수 있도록 설정 파일을 추가합니다.

Gunicorn configuration

```
(next_13_venv) ubuntu@ip-172-31-13-173:/srv/next-session-11-hw/next_13_venv/bin$ cd /etc/systemd/system
(next_13_venv) ubuntu@ip-172-31-13-173:/etc/systemd/system$ sudo nano gunicorn.service
```

```
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/srv/next-session-11-hw/base_auth
ExecStart=/srv/next-session-11-hw/next_13_venv/bin/gunicorn \
    --workers 2 \
    --bind 0:8000 \
    base_auth.wsgi:application

[Install]
WantedBy=multi-user.target
```

백드라운드에서 시스템 설정으로 운영될 수 있도록 서비스 형식을 작성할 것입니다.

이름이 꼭 gunicorn일 필요는 없습니다. 하지만 인식하기 편하게 하기 위해 이름을 gunicorn.service로 지었습니다.

Gunicorn configuration

```
(next_13_venv) ubuntu@ip-172-31-13-173:/etc/systemd/system$ sudo systemctl start gunicorn.service
(next_13_venv) ubuntu@ip-172-31-13-173:/etc/systemd/system$ sudo systemctl status gunicorn.service
● gunicorn.service - gunicorn daemon
   Loaded: loaded (/etc/systemd/system/gunicorn.service; disabled; preset: enabled)
   Active: active (running) since Sat 2024-04-27 11:26:28 UTC; 4s ago
     Main PID: 2941 (gunicorn)
        Tasks: 3 (limit: 1130)
      Memory: 64.0M (peak: 64.1M)
         CPU: 573ms
    CGroup: /system.slice/gunicorn.service
            └─2941 /srv/next-session-11-hw/next_13_venv/bin/python3 /srv/next-session-11-hw/next_13_venv/bin/gunicorn --workers 2 --bind 0:8000 base_auth.wsgi:application
            └─2942 /srv/next-session-11-hw/next_13_venv/bin/python3 /srv/next-session-11-hw/next_13_venv/bin/gunicorn --workers 2 --bind 0:8000 base_auth.wsgi:application
            └─2943 /srv/next-session-11-hw/next_13_venv/bin/python3 /srv/next-session-11-hw/next_13_venv/bin/gunicorn --workers 2 --bind 0:8000 base_auth.wsgi:application
```

[NEXT-LIKELION](#)

- [New](#)
- [Home](#)
- [로그인](#)
- [회원가입](#)

Home!

제목: z

내용: zzzzz

제목: testing

내용: testing

제목: tesing my page

내용: testing

글 쓰러 가기!

CSS가 여전히...

그리고 포트도 여전히 8000포트. 일반적인 경우와 다른 접속 방식을 가지고 있습니다.

CSS와 포트 매핑을 하기 위해 Nginx가 나설 차례입니다.

| Gunicorn configuration

```
--bind unix:/tmp/gunicorn.sock \
```

Nginx를 다운 받습니다

```
$ sudo apt-get install nginx
```

Gunicorn.service 파일에서 bind를 오른쪽과 같이 수정해주세요.
서버 내부 통신에 용이한 규격으로 변경합니다.

그리고 gunicorn을 재시작 합니다.

```
$ sudo systemctl restart gunicorn.service
```

Nginx에 gunicorn을 바인딩할 설정 파일을 생성합니다. Next는 임의로 정한 파일입니다.

```
$ sudo nano /etc/nginx/sites-available/next
```

```
server {
    listen 80;
    server_name ec2-43-202-46-198.ap-northeast-2.compute.amazonaws.com;

    location / {
        proxy_pass http://unix:/tmp/gunicorn.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /static {
        alias /srv/next-session-11-hw/base_auth/blog/static;
    }
}
```

Gunicorn configuration

```
--bind unix:/tmp/gunicorn.sock \
```

Gunicorn.service 파일에서 bind를 오른쪽과 같이 수정해주세요.
서버 내부 통신에 용이한 규격으로 변경합니다.

이제 설정한 파일을 nginx에 등록합니다.

```
$ sudo ln -s /etc/nginx/sites-available/next /etc/nginx/sites-enabled
```

```
$ sudo systemctl start nginx
```

```
$ sudo systemctl status nginx
```

```
(next_13_venv) ubuntu@ip-172-31-13-173:/etc/nginx/sites-available$ sudo systemctl status nginx
```

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-04-27 12:17:57 UTC; 19min ago
     Docs: man:nginx(8)
  Process: 3989 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 3991 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 3992 (nginx)
    Tasks: 2 (limit: 1130)
   Memory: 1.8M (peak: 2.4M)
      CPU: 14ms
   CGroup: /system.slice/nginx.service
           └─3992 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─3994 "nginx: worker process"
```

```
Apr 27 12:17:57 ip-172-31-13-173 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
```

```
Apr 27 12:17:57 ip-172-31-13-173 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```


HOME!

제목: z

내용 : zzzzz

제목: testing

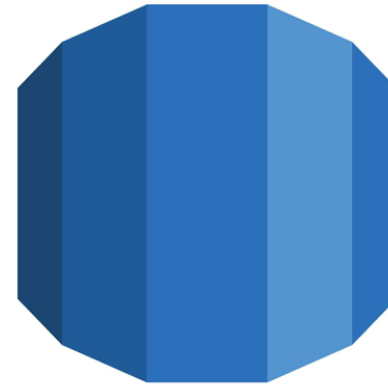
내용 : testing

제목: tesing my page

내용 : testing

| 완성..?

- https 연결 없음
- Fixed url domain 없음
- 서버 직접 설정해야함
- 업데이트도 직접 설정해야함
- 서버 한 대임
- 디비도 없음



Amazon **RDS**



GitHub Actions

| 알아보면 좋을 내용들

- Postgresql : RDB
- RDB 서버 : RDS
- 예쁜 url: Domain: Route53, 가비아
- 자동 배포 및 빌드 : CI/CD : genkins
- 서버 설정 간편화 : Docker