



포팅메뉴얼

1. 개발 환경

- [1.1. Frontend/AI](#)
- [1.2. Backend/AI](#)
- [1.3. Server](#)
- [1.4. IDE](#)
- [1.5. 형상/이슈관리](#)
- [1.6. 기타 툴](#)
- [1.7. 앱 설치 및 실행 명령어](#)

2. EC2 세팅

- [2.1. 기본 세팅](#)
- [2.2. Jenkins파이프라인](#)
- [2.3. EC2 Port](#)
- [2.4. 방화벽\(UFW\) 설정](#)

1. 개발 환경

1.1. Frontend/AI

기술스택	버전	설명
Visual Studio Code	1.97.2	개발 환경 (IDE)
React Native	expo	모바일 앱 개발 프레임워크
Fast TFLite	1.6.1	로컬 TFLite 추론 엔
Frame Processor		프레임 단위로 추론 호출 (60FPS 대응)
EfficientDet-lite0		사과 탐지용 TFLite 객체 인식 모델 (모바일 최적화)
Lottie	7.1.0	애니메이션 로딩
Skia	1.5.0	2D 그래픽 렌더링
Vision Camera	4.6.4	카메라 입력처리
Expo	~52.0.11	Expo 앱 빌드 및 실행도구
Expo SlapshScreen	~0.29.24	앱 로딩 화면 제어

1.2. Backend/AI

항목	스택 / 프레임워크	버전	설명
언어 및 환경	Python	3.10	백엔드 및 AI 모델 전반에 사용된 언어
백엔드 프레임워크	FastAPI	0.115.12	비동기 REST API 백엔드 구현
서버 실행기	Uvicorn	0.34.2	ASGI 서버로 FastAPI 실행
데이터 처리	NumPy / Pandas / Scikit-Learn	2.1.3 / 2.2.3 / 1.6.1	수치 연산 및 전처리
시각화 도구	Matplotlib / Seaborn	3.10.3 / 0.13.2	디버깅 및 결과 분석 시 시각화
모델 추론용 프레임워크	LightGBM / XGBoost	4.6.0 / 3.0.1	당도 예측 모델 (트리 기반)
딥러닝 프레임워크	PyTorch / TensorFlow	2.7.0 / 2.19.0	CNN 추출 및 TFLite 모델 사용
융합 모델 구조	CNN + 수작업 특징 + MLP		이미지 CNN 임베딩 (EfficientNetB0)과 수작업 feature(색상, 텍스처 등)를 결합한 회귀 모델 구조
이미지 처리	OpenCV / Kornia / scikit-image	4.11.0 / 0.8.1 / 0.25.2	이미지 전처리 및 변환
YOLO 모델	Ultralytics YOLOv8	8.3.131	사과 인식 모델 (PyTorch 및 TFLite 지원)
API 유틸리티	python-multipart / Pydantic	0.0.20 / 2.11.4	이미지 업로드 및 스키마 검증
로깅 및 디버깅	Rich / tqdm	14.0.0 / 4.67.1	터미널 시각화 및 진행률 표시
기타 도구	TensorBoard / Netron	2.19.0 / 8.3.3	학습 로그 시각화 및 모델 구조 확인

1.3. Server

소프트웨어	버전	설명
Ubuntu	22.04.05 LTS x86_64	서버 운영체제
Nginx	1.18.0	Reverse Proxy 및 Static 파일 서빙
Docker	28.1.1	컨테이너 가상화 플랫폼
Jenkins	2.504.1	CI/CD 자동화 도구

1.4. IDE

- Visual Studio Code (1.97.2)

1.5. 형상/이슈관리

- GitLab
- Jira

1.6. 기타 툴

- Postman

1.7. 앱 설치 및 실행 명령어

- FE/dalldidan
 - 필요 패키지 설치

```
npm install
```

- 개발 모드로 빌드(emulator 활용 혹은 안드로이드 폰 연결 후 사용)

SDK 설치 및 환경변수 PATH 설정 필요(C:\...\...
\AppData\Local\Android\Sdk)

```
adb devices(폰 연결 했을 경우 연결 확인용)
```

```
npx expo start --dev-client
```

- 빌드 후 QR 코드와 접속할 서버 주소가 뜸
(expo server 주소 입력하는 곳에

`http://{IP주소}:8081` 입력)

```
$ npx expo start --dev-client
Starting project at C:\Users\SSAFY\Desktop\S12P31E206\FE\dalldidan
Starting Metro Bundler

> Metro waiting on
exp+dalldidan-dev://expo-development-client/?url=http%3A%2F%2F192.168.30.129%3A8081
> Scan the QR code above to open the project in a development build.
```

- 개발 완료 후 production build 하기(배포)
(expo 계정 필요)

```
eas build --platform android --profile production
```

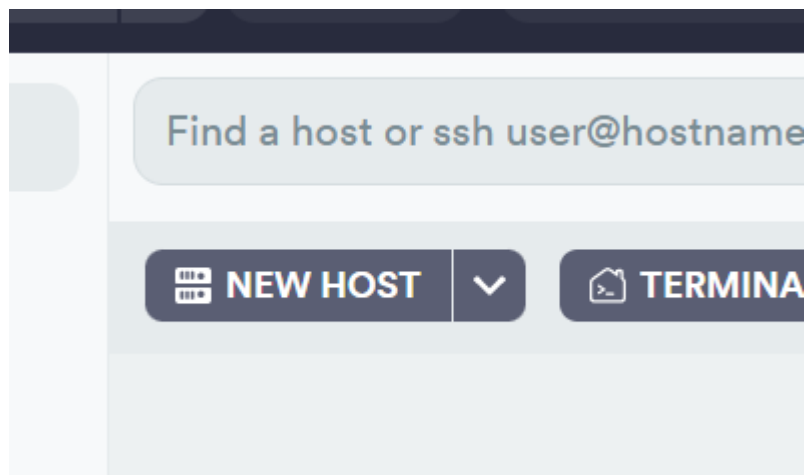
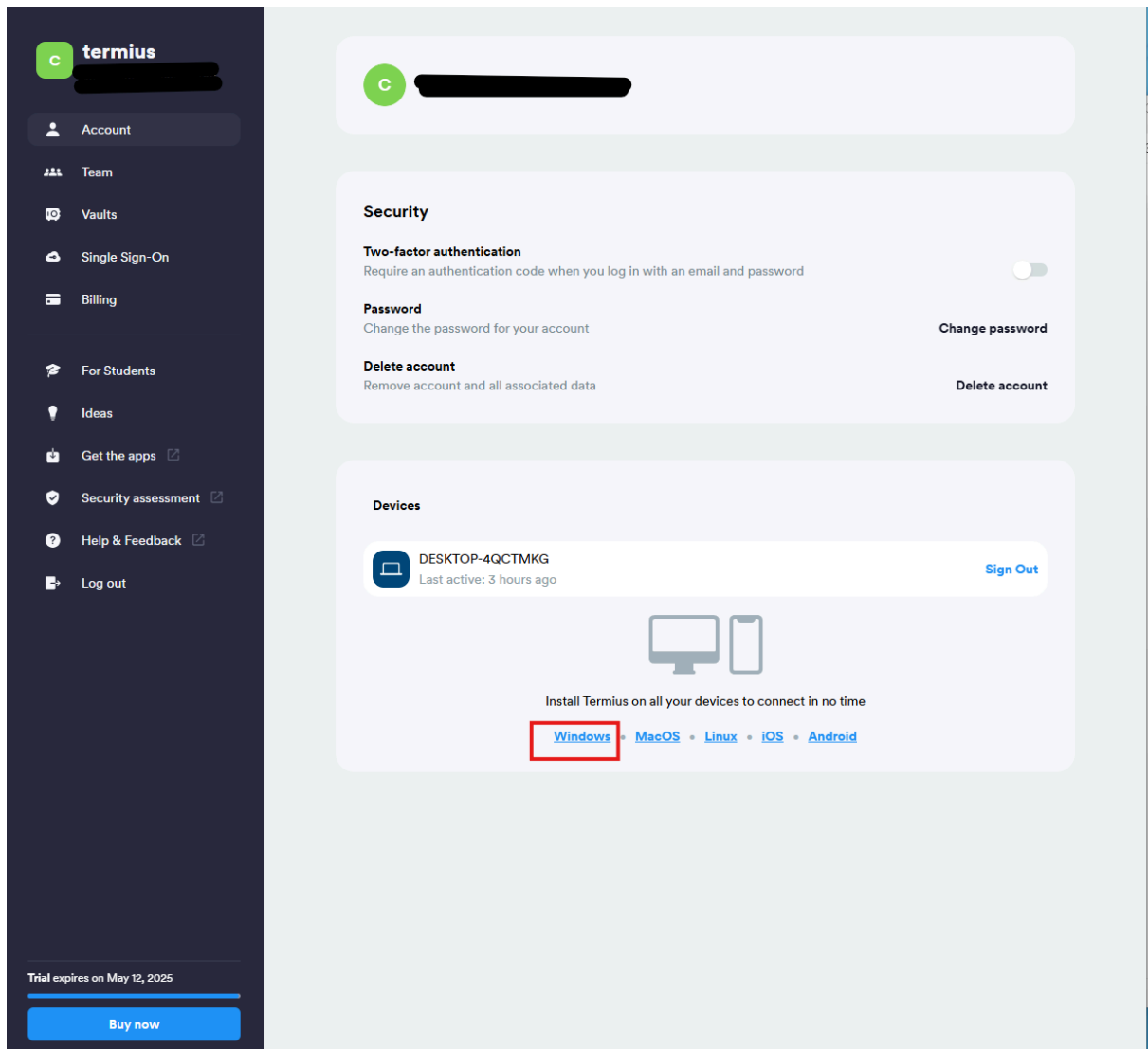
2. EC2 세팅

2.1. 기본 세팅

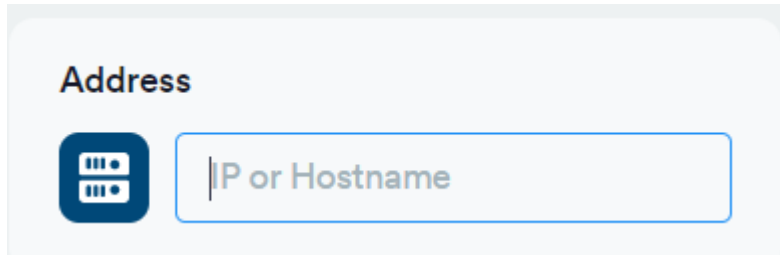
1. termius 설정

terminus를 구글 계정으로 가입한다. 비밀번호를 12자 정도로 해서 만든다.

terminus를 Windows설치한다

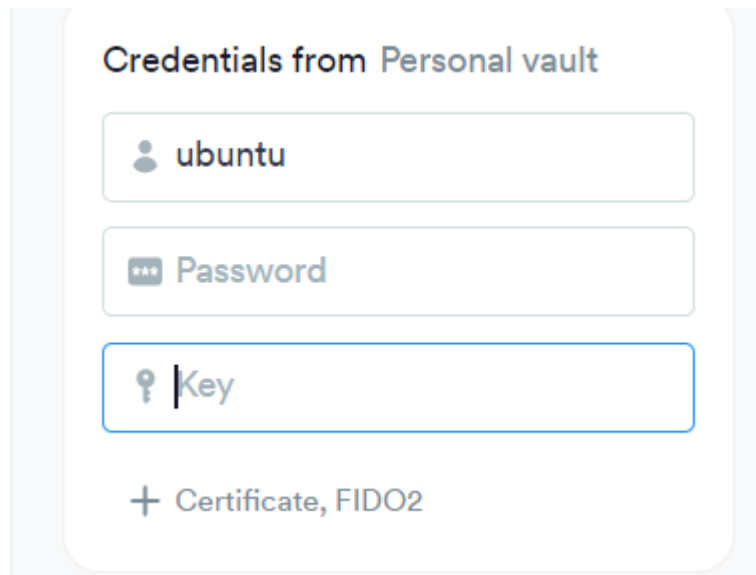


New Host를 누르면 오른쪽에 입력창이 새로로 열린다
그러면



The screenshot shows a light gray rounded rectangle with the title 'Address' in bold black text. Below the title is a dark blue square icon with a white terminal window symbol. To the right of the icon is a white rectangular input field with a blue border. Inside the field, the text 'IP or Hostname' is displayed in a light gray font, preceded by a vertical cursor line.

Address에 도메인 주소를 입력한다.



The screenshot shows a light gray rounded rectangle with the title 'Credentials from Personal vault' in bold black text. Below the title are three stacked white rectangular input fields with blue borders. The first field contains a gray user icon and the text 'ubuntu'. The second field contains a gray password icon (three dots) and the text 'Password'. The third field contains a gray key icon and the text 'Key', with a vertical cursor line positioned after the 'y'. Below these fields is a plus sign icon followed by the text 'Certificate, FIDO2'.

Keys를 추가한다: Edit Key에서
pem파일을 넣으면된다.

2. termius에서 작업을 해보자

```

Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

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

System information as of Mon Apr 28 11:34:11 UTC 2025

System load:  0.0          Processes:      132
Usage of /:   1.6% of 309.95GB Users logged in:  0
Memory usage: 4%          IPv4 address for eth0: 
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

92 updates can be applied immediately.
15 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Apr 18 04:20:08 2025 from
ubuntu@ip-172-26-2-54:~$ # Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

```

<https://docs.docker.com/engine/install/ubuntu/> 도커 공식문서...

dockerdocs

Get started

Guides

Manuals

Reference

Manuals

OPEN SOURCE

Docker Engine

Install

Ubuntu

Debian

RHEL

Fedora

Raspberry Pi OS (32-bit)

CentOS

SLES (s390x)

Binaries

Post-installation steps

Storage

Networking

Containers

CLI

Daemon

Manage resources

Logs and metrics

Security

Swarm mode

Deprecated features

Docker Engine plugins

Release notes

Docker Build

Docker Compose

Install using the `apt` repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker `apt` repository. Afterward, you can install and update Docker from the repository.

- Set up Docker's `apt` repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable"
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```
- Install the Docker packages.

Latest

Specific version

To install the latest version, run:

```
docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```
- Verify that the installation is successful by running the `hello-world` image:

```
$ sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine

Edit this page

Request changes

Table of contents

Prerequisites

Firewall limitations

OS requirements

Uninstall old versions

Installation methods

Install using the `apt` repository

Upgrade Docker Engine

Install from a package

Upgrade Docker Engine

Install using the convenience script

Install pre-releases

Upgrade Docker after using the convenience scr...

Uninstall Docker Engine

Next steps

#도커 설치 명령어

Add Docker's official GPG key:

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/ke
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Add the repository to Apt sources:

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker
```

```
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
```

3.

일반 사용자 권한 설정

```
sudo usermod -aG docker $USER
```

등록 && 시작

```
sudo systemctl enable docker
```

```
sudo systemctl start docker
```

```
sudo systemctl status docker
```

#status로 아래 사진의 결과값을 보여줄 수 있다.

나가기는 q 누르자


```
ubuntu@ip-172-26-2-54:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-04-28 11:46:32 UTC; 3min 46s ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 4454 (dockerd)
      Tasks: 10
     Memory: 20.8M
        CPU: 331ms
    CGroup: /system.slice/docker.service
            └─4454 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

apt == advanced package tool

4. 엔지닉스

https://nginx.org/en/linux_packages.html#Ubuntu 엔지닉스 공식문서...

Ubuntu

Install the prerequisites:

```
sudo apt install curl gnupg2 ca-certificates lsb-release ubuntu-keyring
```

Import an official nginx signing key so apt could verify the packages authenticity. Fetch the key:

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor &
| sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

Verify that the downloaded file contains the proper key:

```
gpg --dry-run --quiet --no-keyring --import --import-options import-show /usr/share/keyrings/nginx-archive-keyring.gpg
```

The output should contain the full fingerprint `573BFD6B3D8FBC641079A6A8ABF5B0827BD9BF62` as follows:

```
pub   rsa2048 2011-08-19 [SC] [expires: 2027-05-24]
       573BFD6B3D8FBC641079A6A8ABF5B0827BD9BF62
uid
       nginx signing key <signing-key@nginx.com>
```

Note that the output can contain other keys used to sign the packages.

To set up the apt repository for stable nginx packages, run the following command:

```
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] &
http://nginx.org/packages/ubuntu `lsb_release -cs` nginx" &
| sudo tee /etc/apt/sources.list.d/nginx.list
```

If you would like to use mainline nginx packages, run the following command instead:

```
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] &
http://nginx.org/packages/mainline/ubuntu `lsb_release -cs` nginx" &
| sudo tee /etc/apt/sources.list.d/nginx.list
```

Set up repository pinning to prefer our packages over distribution-provided ones:

```
echo -e "Package: *Pin: origin nginx.orgPin: release o=nginxPin-Priority: 900Pin" &
| sudo tee /etc/apt/preferences.d/99nginx
```

To install nginx, run the following commands:

```
sudo apt update
sudo apt install nginx
```

SLES

Install the prerequisites:

```
sudo zypper install curl ca-certificates gpg2
```

To set up the zypper repository for stable nginx packages, run the following command:

```
sudo zypper addrepo --gpgcheck --type yum --refresh --check &
'http://nginx.org/packages/sles/$releasever/major' nginx-stable
```

```
sudo apt install curl gnupg2 ca-certificates lsb-release ubuntu-keyring
sudo apt install nginx
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
```

```
ubuntu@ip-172-26-2-54:~$ sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-04-28 11:54:32 UTC; 1min 37s ago
     Docs: man:nginx(8)
```

5. 방화벽 설정

```
sudo ufw allow 22
```

한줄씩

```
sudo ufw allow 80
```

```
sudo ufw allow 8080
```

```
sudo ufw allow 443
```

```
sudo ufw enable
```

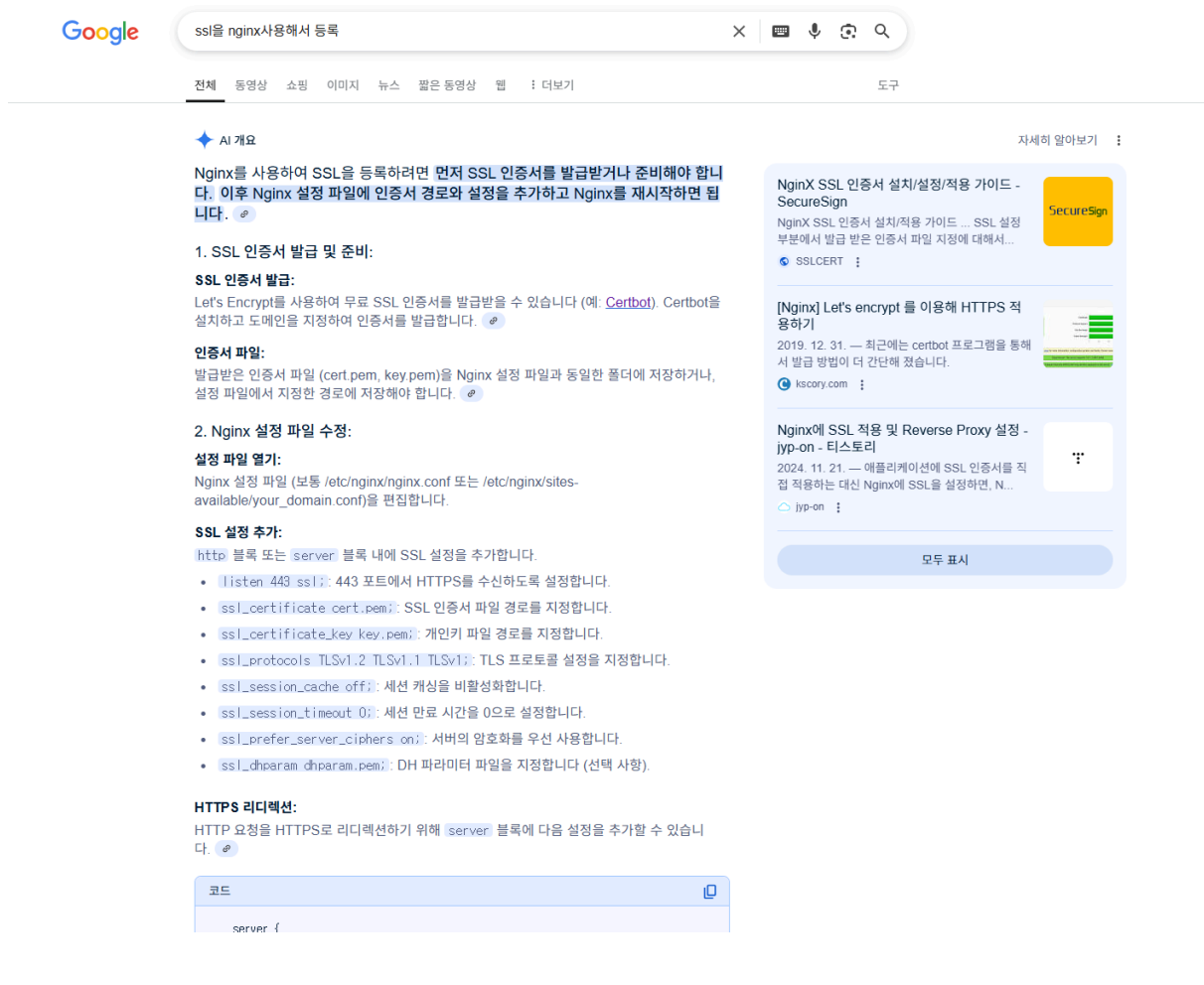
```
sudo ufw status numbered
```

```
ubuntu@ip-172-26-2-54:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
ubuntu@ip-172-26-2-54:~$ sudo ufw status numbered
Status: active
```

	To	Action	From
--	-----	-----	----
[1]	22	ALLOW IN	Anywhere
[2]	80	ALLOW IN	Anywhere
[3]	44	ALLOW IN	Anywhere
[4]	8989	ALLOW IN	Anywhere
[5]	8080	ALLOW IN	Anywhere
[6]	443	ALLOW IN	Anywhere
[7]	22 (v6)	ALLOW IN	Anywhere (v6)
[8]	80 (v6)	ALLOW IN	Anywhere (v6)
[9]	44 (v6)	ALLOW IN	Anywhere (v6)
[10]	8989 (v6)	ALLOW IN	Anywhere (v6)
[11]	8080 (v6)	ALLOW IN	Anywhere (v6)
[12]	443 (v6)	ALLOW IN	Anywhere (v6)

6. HTTPS관련 설정인 ssl인증서 발급

구글에 검색해보면,,,



Google

ssl을 nginx사용해서 등록

전체 동영상 쇼핑 이미지 뉴스 짧은 동영상 웹 더보기

도구

AI 개요

자세히 알아보기

Nginx를 사용하여 SSL을 등록하려면 먼저 SSL 인증서를 발급받거나 준비해야 합니다. 이후 Nginx 설정 파일에 인증서 경로와 설정을 추가하고 Nginx를 재시작하면 됩니다.

1. SSL 인증서 발급 및 준비:

SSL 인증서 발급:
Let's Encrypt를 사용하여 무료 SSL 인증서를 발급받을 수 있습니다 (예: [Certbot](#)). Certbot을 설치하고 도메인을 지정하여 인증서를 발급합니다.

인증서 파일:
발급받은 인증서 파일 (cert.pem, key.pem)을 Nginx 설정 파일과 동일한 폴더에 저장하거나, 설정 파일에서 지정한 경로에 저장해야 합니다.

2. Nginx 설정 파일 수정:

설정 파일 열기:
Nginx 설정 파일 (보통 /etc/nginx/nginx.conf 또는 /etc/nginx/sites-available/your_domain.conf)을 편집합니다.

SSL 설정 추가:
http 블록 또는 server 블록 내에 SSL 설정을 추가합니다.

- listen 443 ssl: 443 포트에서 HTTPS를 수신하도록 설정합니다.
- ssl_certificate cert.pem: SSL 인증서 파일 경로를 지정합니다.
- ssl_certificate_key key.pem: 개인키 파일 경로를 지정합니다.
- ssl_protocols TLSv1.2 TLSv1.1 TLSv1: TLS 프로토콜 설정을 지정합니다.
- ssl_session_cache off: 세션 캐싱을 비활성화합니다.
- ssl_session_timeout 0: 세션 만료 시간을 0으로 설정합니다.
- ssl_prefer_server_ciphers on: 서버의 암호화를 우선 사용합니다.
- ssl_dhparam dhparam.pem: DH 파라미터 파일을 지정합니다 (선택 사항).

HTTPS 리디렉션:
HTTP 요청을 HTTPS로 리디렉션하기 위해 server 블록에 다음 설정을 추가할 수 있습니다.

코드

```
server {
```

<https://letsencrypt.org/ko/getting-started/>

위에 나와있는대로 하면 되지만(<https://certbot.eff.org/>),

그러기보다는 그냥 아래 명령어를 입력함

```
sudo apt-get update
```

```
sudo apt-get install -y certbot python3-certbot-nginx
```

적용할 도메인 주소와 이메일 입력해주기

```
sudo certbot certonly --nginx -d {도메인주소}
```

→ 에서 나오는 내용을----if위의 내용을 복사해서 gpt에게 물어보자!!!!

질문내용 "이거에 맞게 ssl적용되는 nginx.conf를 만들어줘"

nginx.conf를 아래 명령어 위치로 이동해서 안의 내용을 바꾼다.

```
sudo nano /etc/nginx/nginx.conf
```

모두 선택(ctrl+k)후 다 지우고 붙여넣고 저장하고 창을닫는다.

```
sudo nano /etc/nginx/sites-available/default
```

```
sudo nginx -t  
#ok가 뜨는 것을 확인한다.
```

```
sudo systemctl restart nginx
```

7.젠킨스를 설치하려면,,, Java부터 설치해야하는데,
백엔드를 FastAPI로 쓸예정이지만, 예전플젝을 참고하여 springboot가 추가될지도 모르는
상황에 대비하기위해,
자바 버전을 예전플젝버전을 사용할 예정으로 생각하고, 버전을 알아오니, 17이었다

```
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(17)  
    }  
}
```

이 버전의 자바를 깔아보자...

```
sudo apt-get update  
sudo apt-get install openjdk-17-jdk
```

설치 끝나면 확인

```
java -version
```

8. 환경변수 설정

```
echo $JAVA_HOME
```

```
sudo nano /etc/environment
```

#environment를 써야 젠킨스 가능. 그냥 bashsr?? 은 젠킨스가 무시할 수 있음.

#안의 내용을 이렇게 추가한다.

```
PATH="~~~~:/usr/lib/jvm/java-17-openjdk-amd64/bin"
```

```
JAVA_HOME="/usr/lib/jvm/java-17-openjdk-amd64"
```

```
sudo update-alternatives --config java
```

#이 명령어를 해서, 자바가 설치된 위치를 확인하고, #안의 내용을 이렇게 추가한다.이부

```
source /etc/environment
```

9. 젠킨스 설치

<https://www.jenkins.io/download/> 젠킨스 공식문서...

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
```

```
/etc/apt/sources.list.d/jenkins.list > /dev/null  
sudo apt-get update  
sudo apt-get install jenkins
```

```
sudo ls -l /var/lib/jenkins_home
```

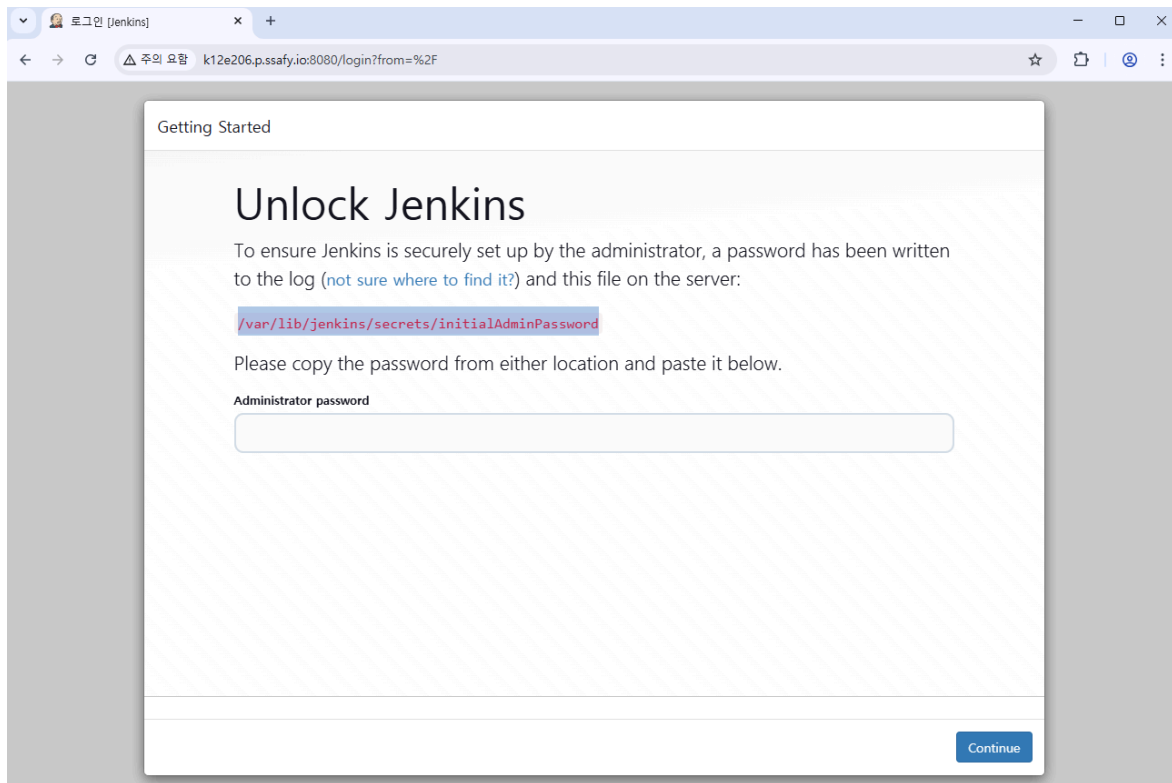
```
sudo chown -R jenkins:jenkins /var/lib/jenkins  
sudo chown -R jenkins:jenkins /var/cache/jenkins  
sudo chown -R jenkins:jenkins /var/log/jenkins
```

젠킨스 상태를 보자

```
sudo systemctl enable jenkins  
sudo systemctl start jenkins  
sudo systemctl status jenkins  
→ 결과: active(running)
```

<http://k12e206.p.ssafy.io:8080/>로 들어가자

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



```
server {  
    listen 80;  
    server_name k12e206.p.ssafy.io;  
  
    # HTTP → HTTPS 리다이렉트  
    return 301 https://$host$request_uri/;  
}  
  
server {  
    listen 443 ssl;  
    server_name k12e206.p.ssafy.io;  
  
    ssl_certificate /etc/letsencrypt/live/k12e206.p.ssafy.io/fullchain.pem;  
    ssl_certificate_key /etc/letsencrypt/live/k12e206.p.ssafy.io/privkey.pem;  
  
    ssl_protocols TLSv1.2 TLSv1.3;
```



```

ssl_ciphers HIGH:!aNULL:!MD5;

# 여기에 실제 서비스 설정 추가
location / {
    proxy_pass http://localhost:3000/; # 예시로, 내부의 3000번 포트 앱에 연결
    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 /jenkins/ {
    proxy_pass http://localhost:8080/; # 젠킨스 접속
    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;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;

    proxy_redirect http://localhost:8080/jenkins/ /jenkins/;
    client_max_body_size 10m;
    proxy_read_timeout 90;
    proxy_request_buffering off;
}
}

```

sudo systemctl edit jenkins

```

[Service]
Environment="JENKINS_PREFIX=/jenkins"

```

sudo systemctl daemon-reload

```

sudo nano /etc/nginx/sites-available/default

```

```
sudo nginx -t
```

→ 결과

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

- 명령어 정리

```
# Nginx 웹서버를 재시작
sudo systemctl restart nginx

# Jenkins 재시작
sudo systemctl edit jenkins

# Jenkins 서비스의 시스템 설정 오버라이드 파일을 편집.
sudo systemctl edit jenkins

# Nginx의 기본 사이트 설정 파일을 편집.
sudo nano /etc/nginx/sites-available/default

# Nginx 설정이 올바른지 확인하는 명령어
sudo nginx -t
```

2.2. Jenkins파이프라인

```
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "mybackend:latest"
    DOCKER_CONTAINER = "mybackend-container"
  }
  stages {
```

```

stage('Fetch or Clone Repository') {
    steps {
        git credentialsId: 'gitlab-token', branch: 'develop', url: 'https://lab.ssa
    }
}
stage('Build Docker Image') {
    steps {
        sh '''
        docker image rm -f ${DOCKER_IMAGE} || true
        cd repo/be
        docker build -t ${DOCKER_IMAGE} .
        '''
    }
}
stage('Stop Old Container') {
    steps {
        sh '''
        docker stop ${DOCKER_CONTAINER} || true
        docker rm ${DOCKER_CONTAINER} || true
        '''
    }
}
stage('Run New Container') {
    steps {
        sh '''
        docker run -d --name ${DOCKER_CONTAINER} -p 8000:8000 ${DO
        '''
    }
}
}
}
}
}

```

- 와이어 가드 설치후 nginx 설치 파일 수정

```

server {
    listen 443 ssl;
    server_name k12e206.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k12e206.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k12e206.p.ssafy.io/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    # /predict 요청은 로컬 노트북(WireGuard ????)로 전달
    location /predict {
        proxy_pass http://로컬노트북/predict;
        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;
    }

    # 나머지 모든 요청은 EC2 내부 8000번으로
    location / {
        proxy_pass http://localhost:8000;
        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;
    }
}

```

2.3. EC2 Port

Port 번호	내용
8080	Jenkins

Port 번호	내용
80	http
443	https

2.4. 방화벽(UFW) 설정

인바운드 허용 포트 목록 (IPv4 & IPv6)

22	ALLOW	Anywhere	# SSH 접속
80	ALLOW	Anywhere	# HTTP
44	ALLOW	Anywhere	# 사용자 정의 포트
8989	ALLOW	Anywhere	# Apache 또는 커스텀 웹서버
8080	ALLOW	Anywhere	# Jenkins 등 웹 앱
443	ALLOW	Anywhere	# HTTPS
51820/udp	ALLOW	Anywhere	# WireGuard VPN
9000/tcp	ALLOW	Anywhere	# FastAPI 또는 AI 서비스

동일한 IPv6 버전 포트 허용

22 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
44 (v6)	ALLOW	Anywhere (v6)
8989 (v6)	ALLOW	Anywhere (v6)
8080 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
51820/udp (v6)	ALLOW	Anywhere (v6)
9000/tcp (v6)	ALLOW	Anywhere (v6)

☒ 인터페이스 간 트래픽 포워딩 규칙 (VPN용 WireGuard 연결)

Anywhere on eth0	ALLOW FWD	Anywhere on wg0
Anywhere on wg0	ALLOW FWD	Anywhere on eth0
Anywhere (v6) on eth0	ALLOW FWD	Anywhere (v6) on wg0
Anywhere (v6) on wg0	ALLOW FWD	Anywhere (v6) on eth0