

Django

단계1: 장고 프로젝트 생성

```
# 장고 프로젝트로 이동  
cd django-server  
# 장고 프로젝트 생성  
django-admin startproject config .
```

단계2: 장고 앱 생성

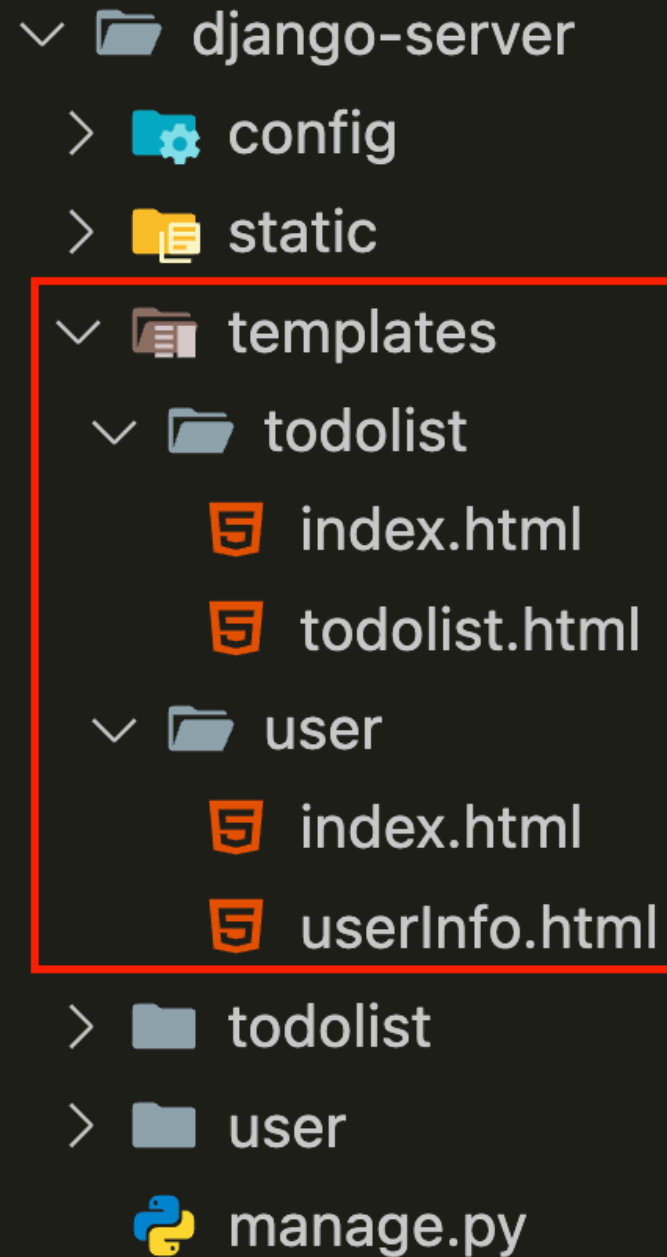
```
# todolist app
python manage.py startapp todolist
# user app
python manage.py startapp user
```

- config/settings.py에 앱 설정 추가

```
INSTALLED_APPS = [
    ...
    "todolist",
    "user"
]
```

단계3: templates 설정

- templates 폴더 및 html 파일 추가



```

  ∨  django-server
    >  config
    >  static
    ∨  templates
      ∨  todolist
        index.html
        todolist.html
      ∨  user
        index.html
        userInfo.html
    >  todolist
    >  user
    manage.py

```

The image shows a file explorer view of a Django project structure. The 'templates' directory and its contents are highlighted with a red box. The structure is as follows:

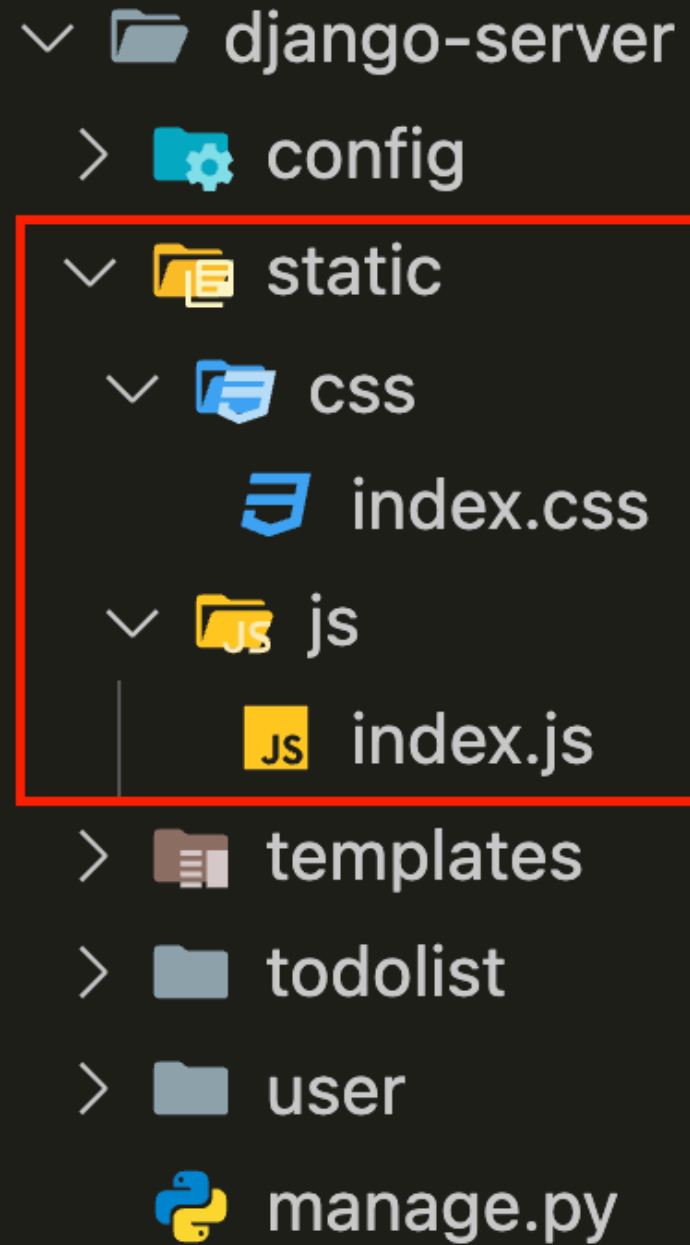
- django-server
 - config
 - static
 - templates
 - todolist
 - index.html
 - todolist.html
 - user
 - index.html
 - userInfo.html
 - todolist
 - user
 - manage.py

- config/settings.py에 templates path 설정 추가

```
TEMPLATES = [  
    {  
        "BACKEND": "django.template.backends.django.DjangoTemplates",  
        "DIRS": ["templates"],  
        "APP_DIRS": True,  
        "OPTIONS": {  
            ...  
        },  
    },  
]
```

단계4: static 설정

- static 폴더 및 css & js 파일 추가



- config/settings.py에 static path 설정 추가

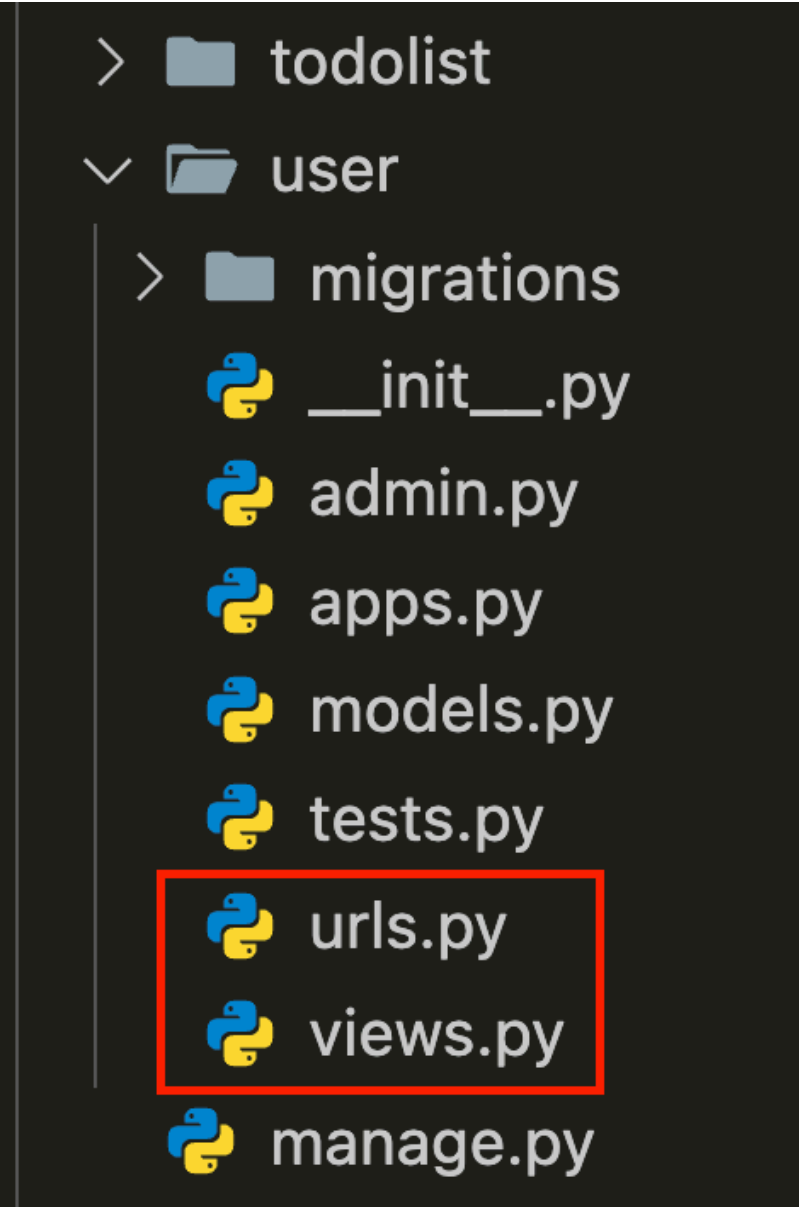
```
import os

STATIC_ROOT = '/static/'
STATIC_URL = "static/"
STATIC_PATH = os.path.join(
    BASE_DIR, "static"
)

STATICFILES_DIRS = (STATIC_PATH,)
```

단계5: user app

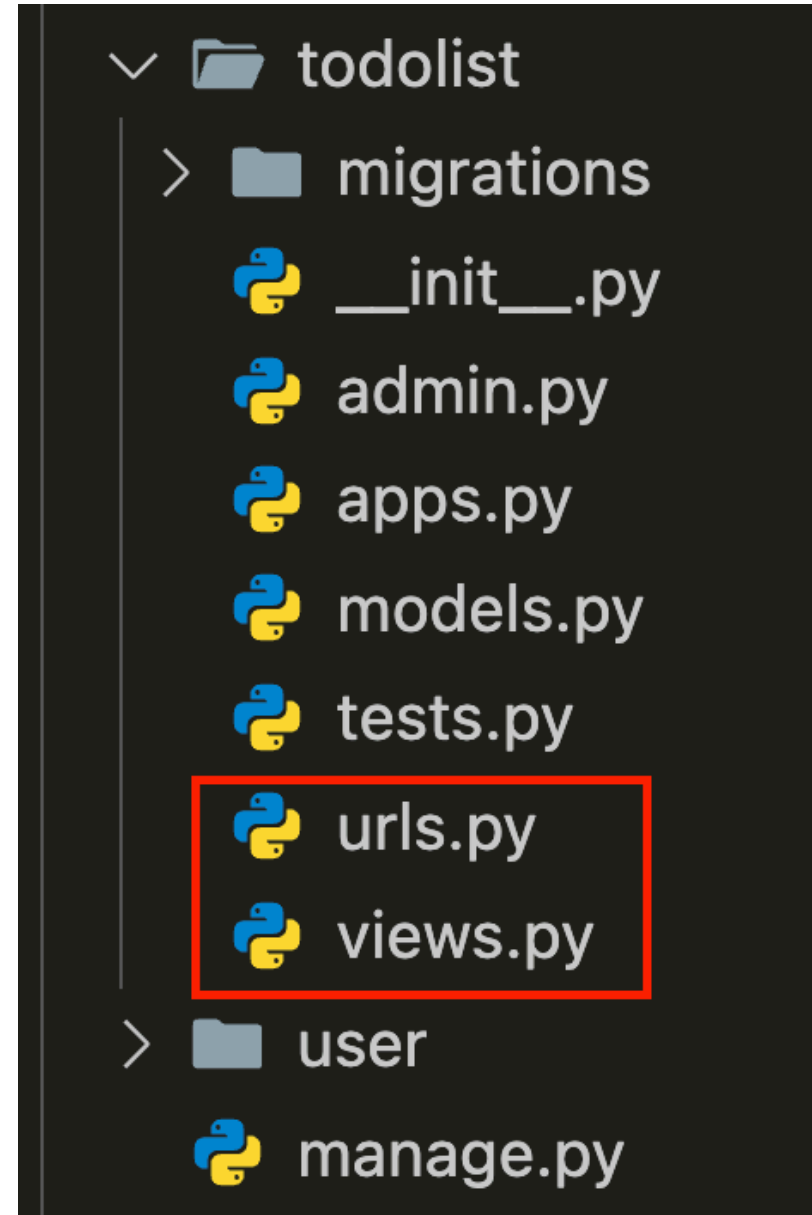
- views.py
- urls.py



```
> folder todolist
✓ folder user
  > folder migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  urls.py
  views.py
  manage.py
```


단계6: todolist app

- views.py
- urls.py



단계7: config.urls.py

```
from django.urls import path, include

urlpatterns = [
    path("", include("todolist.urls")),
    path("user/", include("user.urls"))
]
```

Bash Shell

- run.sh

```
#!/bin/sh

# Django ORM 적용
python manage.py makemigrations
python manage.py migrate --no-input
# static( css & js ) 적용
python manage.py collectstatic --no-input
# gunicorn을 이용한 django server 구동
gunicorn config.wsgi:application --bind 0.0.0.0:8000 &

# nginx 설정 적용
unlink /etc/nginx/sites-enabled/default
# nginx 실행
nginx -g 'daemon off;'
```

Dockerfile

```
# python:[버전]-slim: 해당 이미지는 alpine linux를 기반으로 제작된 이미지인데 이 alpine 은 os 자체 용량이 매우 작음
# https://jadehan.tistory.com/58
FROM python:3.12-slim

# nginx 설치
RUN apt update && apt install nginx -y

# 파이썬 모듈 설치
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

...
```

...

Nginx 설정

COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf

Django server

COPY ./django-server /app

WORKDIR /app

환경변수 적용

ENV SECRET_KEY 'django-insecure-yq9quc!!j!@^p-(ez^o%sb&-jxnq@z@)@di-unm14wp8smed9h '

실행

COPY run.sh .

RUN **chmod** +x run.sh

CMD ["/run.sh"]

Nginx 설정

```
# server: 어떤 서버가 http request 를 처리할지 설정한다.
server {
    # listen: port 를 설정한다.
    listen 80;
    # server_name 디렉티브는 하나의 IP 주소에 대해 여러개의 도메인(domains)을 사용할 수 있게 한다.
    server_name 0.0.0.0;

    # root: request 의 root address 를 설정한다.
    root /app/static;
    ...
}
```

```
# server: 어떤 서버가 http request 를 처리할지 설정한다.
server {
    ...
    # location: request 의 URI 별로 request 를 처리하는 설정을 한다.
    location /static/ {
        alias /app/static/;
    }

    # location: request 의 URI 별로 request 를 처리하는 설정을 한다.
    location / {
        proxy_set_header Host $host;
        # proxy_pass: 해당 location request 를 proxy_pass server 와 매핑하여 해당 서버로 request 를 전달한다.
        proxy_pass http://0.0.0.0:8000;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    # access_log: logging 파일을 설정한다. Logging 형식은 log_format 에서 정의한 main 형식으로 저장한다.
    access_log /var/log/nginx/access.log;
    # error_log: nginx logging 설정. log file 과 logging level 을 설정한다.
    error_log /var/log/nginx/error.log;
}
```

Run Docker Container

단계1: Make docker image

- 명령어: `docker build --platform linux/amd64 -t [dockerhub 아이디]/[이미지명]`
 - Mac M1에서는 꼭 `--platform linux/amd64` 추가해야함

```
docker build --platform linux/amd64 -t goodwon593/django-image .
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↶ dev ±
● ▶ docker build -t goodwon593/django-image .
[+] Building 2.5s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.12-slim
=> [auth] library/python:pull token for registry-1.docker.io
```

- 결과 확인

```
docker images
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  ...  zsh - 4-1. ec2 - django
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↶ dev ±
```

```
➤ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
goodwon593/django-image	latest	01fab73f3a86	27 hours ago	234MB
mysql	latest	b068ea59c677	6 months ago	638MB

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↶ dev ±
```

```
➤ █
```

단계2: Create container

- 명령어: `docker run --name [컨테이너명] -d -p 80:80 [dockerhub 아이디] / [이미지명]`

```
# Create container
docker run --name django-container -d -p 80:80 goodwon593/django-image
# 결과 확인
docker ps
```

```
(web-venv) good593 ♥ ~ /dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
▶ docker run --name django-container -d -p 80:80 goodwon593/django-image
1ef4305543427d6d94ea188a3b659ec418af818779f702e5c175d1f2f0f95e25
(web-venv) good593 ♥ ~ /dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
▶ docker ps
```

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
1ef430554342	goodwon593/django-image django-container	"./run.sh"	5 seconds ago	Up 4 seconds	0.0.0.0:80->80/tcp
a261a60a369e	mysql mysql-stock-db-1	"docker-entrypoint.s..."	2 weeks ago	Up 2 weeks	0.0.0.0:3306->3306/tcp,

```
(web-venv) good593 ♥ ~ /dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
▶
```

단계3: Django 접속 테스트

- `http://localhost:80/`

todolist 초기화면입니다.

DevTools is now available in Korean! [Always match Chrome's language](#) [Switch DevTools to Korean](#)

Elements Console Sources **Network** Performance Memory Application

☐ Preserve log ☐ Disable cache No throttling ☐ Filter ☐ Invert ☐ Hide data URLs ☐ Hide extension URLs **All** [Fetch/XHR](#)

☐ Blocked response cookies ☐ Blocked requests ☐ 3rd-party requests

Name	Status	Type
localhost	200	document
index.css	200	stylesheet
index.js	200	script

단계4: 컨테이너 삭제

```
# 컨테이너 삭제
docker rm -f django-container
# 결과 확인
docker ps -a
```

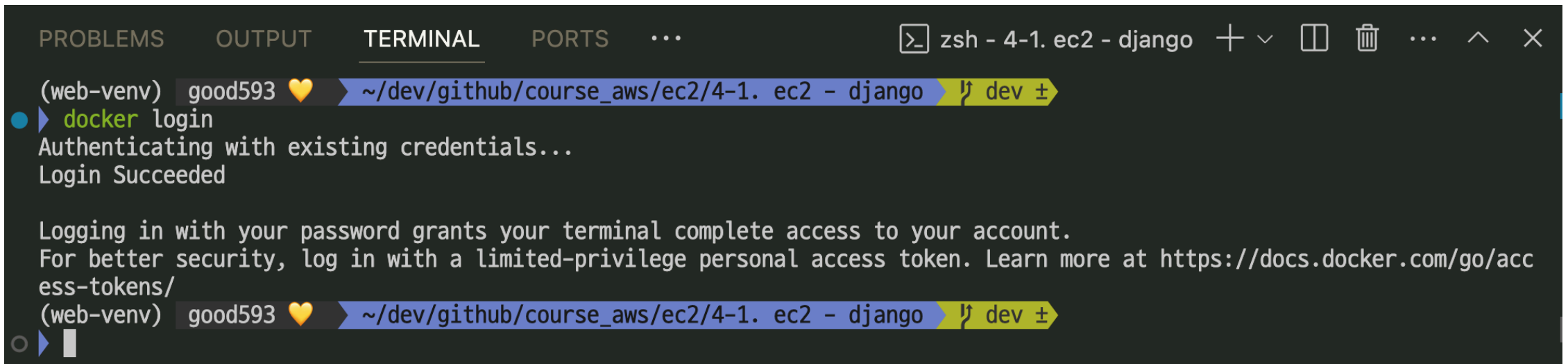
```
(web-venv) good593 ♥ ~/dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
● ▶ docker rm -f django-container
django-container
(web-venv) good593 ♥ ~/dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
● ▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a261a60a369e	mysql	"docker-entrypoint.s..."	2 weeks ago	Up 2 weeks	0.0.0.0:3306->3306/tcp, 33060/tcp	mysql-s
tock-db-1						

```
(web-venv) good593 ♥ ~/dev/github/course_aws/ec2/4-1. ec2 - django ↗ dev ±
○ ▶ █
```

단계5: 도커 로그인

```
docker login
```

A terminal window with a dark background and light text. The title bar at the top shows 'zsh - 4-1. ec2 - django' and several window control icons. The terminal content shows a user prompt '(web-venv) good593' followed by a command 'docker login'. The output of the command is 'Authenticating with existing credentials...' and 'Login Succeeded'. Below this, a message from Docker explains that logging in with a password grants complete access and suggests using a personal access token for better security, with a link to the Docker documentation. The prompt '(web-venv) good593' appears again at the bottom, followed by a cursor.

```
PROBLEMS OUTPUT TERMINAL PORTS ... zsh - 4-1. ec2 - django + - X
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ dev ±
```

```
➤ docker login
```

```
Authenticating with existing credentials...
```

```
Login Succeeded
```

```
Logging in with your password grants your terminal complete access to your account.
```

```
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ dev ±
```

```
➤
```

단계6: 도커허브로 이미지 업로드

```
docker push goodwon593/django-image
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↶ dev ±
● ▶ docker push goodwon593/django-image
Using default tag: latest
The push refers to repository [docker.io/goodwon593/django-image]
dadae49b3dc6: Pushed
96dab329f72d: Pushed
5f70bf18a086: Mounted from armswdev/tensorflow-arm-neoverse
9a676eb60cc9: Pushed
3042c178bc0e: Pushed
a30946f2ac95: Pushed
1762239e252f: Pushed
```

단계7: 도커허브에서 나의 이미지 조회

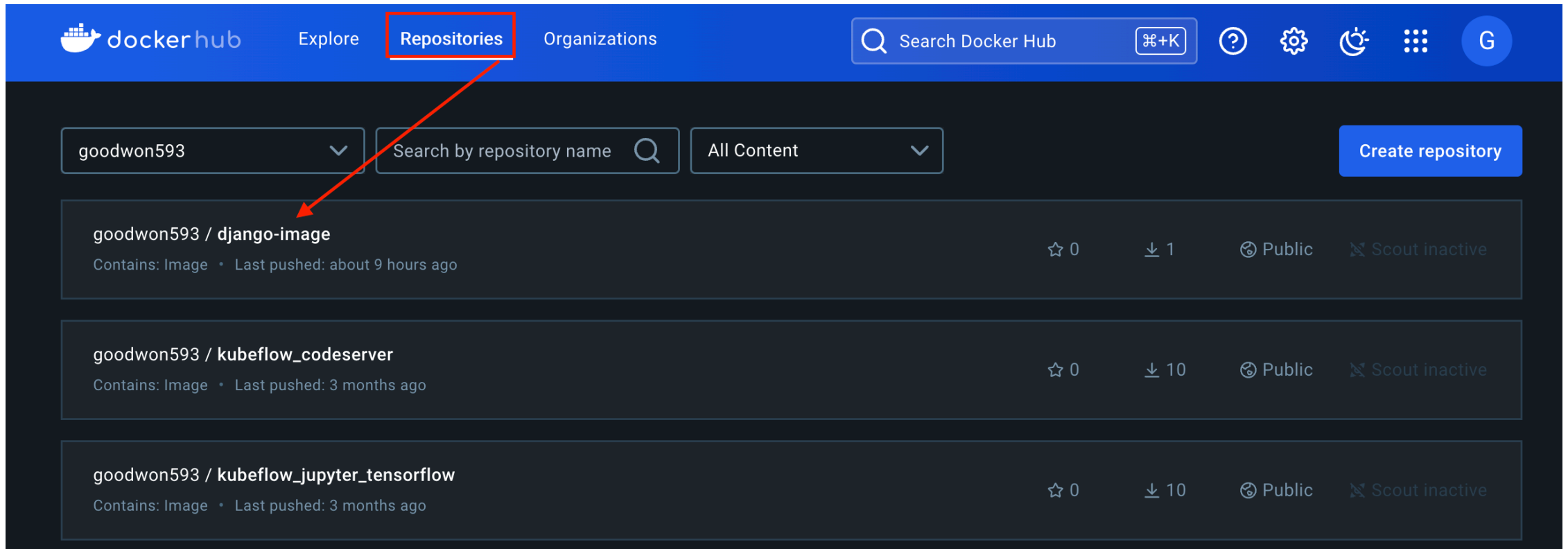
```
docker search goodwon593
```

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↗ dev ±
● ▶ docker search goodwon593
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
goodwon593/kubeflow_jupyter		0		
goodwon593/kubeflow_jupyter_tensorflow		0		
goodwon593/kubeflow_transformer		0		
goodwon593/kubeflow_codeserver		0		
goodwon593/helloapache		0		
goodwon593/kubeflow_tensorflow		0		
goodwon593/hellois-image		0		
goodwon593/django-image		0		

```
(web-venv) good593 ❤️ ➤ ~/dev/github/course_aws/ec2/4-1. ec2 - django ➤ ↗ dev ±
○ ▶ █
```


단계8: 도커허브



The screenshot shows the Docker Hub interface. The top navigation bar includes the Docker Hub logo, 'Explore', 'Repositories' (highlighted with a red box), and 'Organizations'. A search bar is present with the text 'Search Docker Hub'. Below the navigation bar, there's a section for the user 'goodwon593' with a dropdown menu, a search bar labeled 'Search by repository name', and a filter dropdown set to 'All Content'. A blue button 'Create repository' is on the right. The main content area lists three repositories:

Repository Name	Stars	Downloads	Visibility	Scout Status
goodwon593 / django-image Contains: Image • Last pushed: about 9 hours ago	☆ 0	↓ 1	Public	Scout inactive
goodwon593 / kubeflow_codeserver Contains: Image • Last pushed: 3 months ago	☆ 0	↓ 10	Public	Scout inactive
goodwon593 / kubeflow_jupyter_tensorflow Contains: Image • Last pushed: 3 months ago	☆ 0	↓ 10	Public	Scout inactive

참고 문서

- https://www.youtube.com/watch?v=oCu3_JKsYYc
- https://github.com/dotja/django_CD_github_actions