



PyTorch App을 Azure Container Apps로 한방에 배포하기

박정환 | PyTorch.KR



후원사 소개



Agenda

1. 워크샵 소개와 참석자 설문
2. PyTorch, torchvision 및 사전 학습 모델 소개
3. Sample App 소개
4. Container: 소개, 만들기, 실행
5. Azure Container Apps 및 Container Registry 소개
6. Azure Container Registry 사용 및 Container Apps 배포



워크샵 소개

01

준비한 것

02

기대하시는 것

03

안내할 것



준비한 것

-  PyTorch, 객체 탐지 작업 및 사전 학습 모델 소개
-  Sample App 소개: FastAPI를 사용한 간단한 API Server (+ Client)
-  Container 소개
-  Sample App을 Container Image로 만들기 & 실행하기
-  Azure Container Apps 소개
-  Azure Container Apps에 Sample App 배포하기



| 기대하시는 것

- 함께 하시는 분들이 궁금합니다 😊
- 예)
 - 어디서 오셨어요? & 얼마나 걸리셨어요? (& MBTI가 어떻게 되세요?)
 - 무슨 일 하시나요? & 얼마나 오래 하셨나요?
 - PyTorch를 사용해보신 적 있으신가요?
 - 사전 학습 모델을 사용하거나 배포해보신 적 있으신가요?
 - Docker / k8s를 들어보시거나 사용해보신 적 있으신가요?



안내할 것

- 자료/코드는 모두 공개되어 있으며, 교육 목적으로만 제작하였습니다.
- (Microsoft 포함) 어떠한 회사 또는 조직의 입장도 대변하지 않습니다.
- 궁금하신 것은 언제든 질문해주세요! (함께 나누고 성장해요!)
- 나중에라도 언제든 연락주세요!
 - E-mail: 9bow@pytorch.kr
 - GitHub: @9bow



Code/Slide



LinkedIn 

PyTorch 및 torchvision 소개

01

PyTorch 소개

02

Object Detection 소개

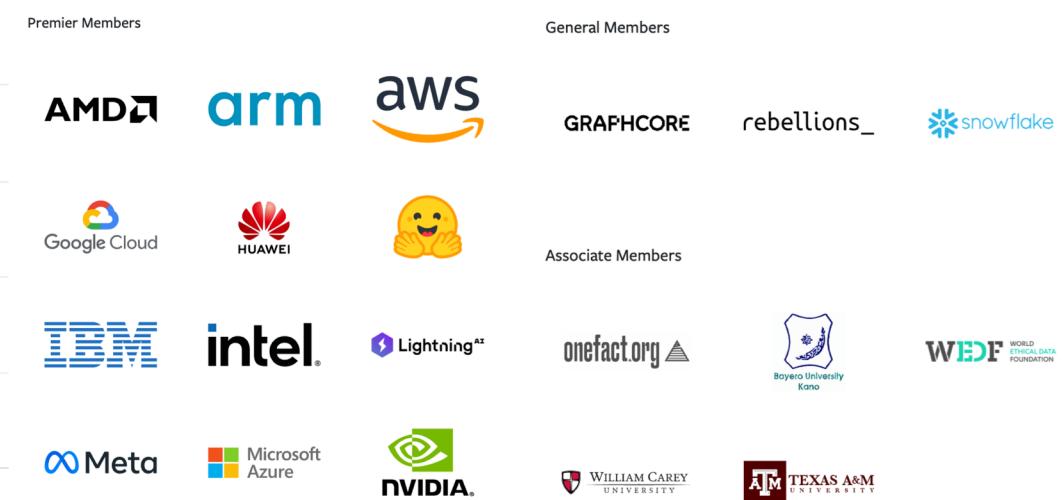
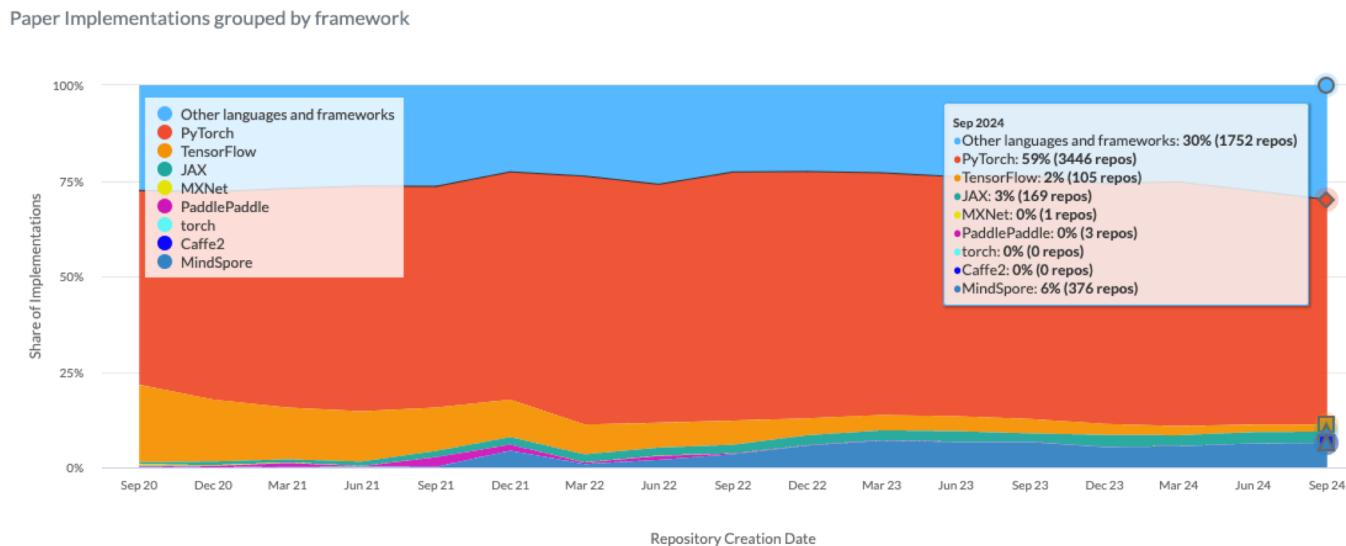
03

사전 학습 모델 소개



PyTorch 소개

- 가장 많이 사용하는 인공지능 Framework
- 초기 Meta AI(前FAIR, Facebook AI Search)에서 개발 주도
- 2022년 9월, Linux Foundation 산하 PyTorch Foundation으로 독립



torchvision (DAPI) 소개

- PyTorch Domain API 중 하나로, 이미지 처리와 컴퓨터 비전 특화
 - torchvision, torchaudio, torchtext, torchrl, torchtune, TorchRec, TorchData, ...

• 주요 구성

모듈	역할
<code>torchvision.datasets</code>	다양한 이미지 및 비디오 데이터셋을 제공하는 모듈로, 간편하게 데이터셋을 로드하고 사용할 수 있습니다. 유명한 데이터셋인 CIFAR, MNIST, ImageNet 등이 포함됩니다.
<code>torchvision.io</code>	이미지와 비디오 데이터를 읽고 쓰는 입출력 기능을 제공하는 모듈입니다. 데이터를 텐서로 변환하여 모델 학습에 사용하기 쉽게 해줍니다.
<code>torchvision.models</code>	미리 학습된 다양한 딥러닝 모델 아키텍처를 제공하는 모듈입니다. ResNet, AlexNet, VGG, EfficientNet 등 여러 CNN 모델을 쉽게 불러와 사용할 수 있습니다.
<code>torchvision.ops</code>	컴퓨터 비전 작업에 유용한 다양한 연산을 제공하는 모듈입니다. RoI Align, NMS (Non-Maximum Suppression) 등 특정 작업에 최적화된 연산 함수를 포함하고 있습니다.
<code>torchvision.transforms</code>	이미지 데이터에 대한 전처리와 변환 기능을 제공하는 모듈입니다. 데이터 증강 (augmentation) 및 정규화 등 다양한 변환을 손쉽게 적용할 수 있습니다.
<code>torchvision.utils</code>	이미지 데이터를 시각화하거나 텐서 데이터를 다루기 위한 다양한 유ти리티 함수를 제공하는 모듈입니다. 예를 들어, <code>make_grid</code> 함수는 여러 이미지를 격자 형태로 묶어 보여줄 수 있습니다.



Object Detection 소개

- CV(Computer Vision) Tasks 중 하나
 - 주요 Task들: Image Classification, Object Detection, Segmentation, ...
 - 하나의 이미지에 포함된 여러 객체를 탐지 결과 반환 (= 무엇이 & 어디에)
 - 객체 종류: Class ID(정수)로 반환
 - 객체 위치: 객체를 감싸는 박스(Bounding Box)의 좌표 반환
 - 신뢰 점수: 0.0 ~ 1.0 사이의 실수 값 (1.0에 가까울수록 높음)

객체 탐지 예시



Object Detection 예시



출처: <https://hands-on.pytorch.kr>, <https://unsplash.com/photos/a-group-of-people-walking-down-a-flight-of-stairs-jEEP-bzH3jl>

사전 학습 모델 소개

- 사전 학습(Pre-trained) 모델: 모델 구조 & 학습된 모델 가중치(Weights)
 - 모델 구조: 특정 작업(예. 객체 탐지, 분류 등)에 적합한 신경망 구조(Class 자료형)
 - 가중치: 특정 데이터셋에서 모델을 학습한 매개변수 값(torch.Tensor 자료형)
- torchvision이 제공하는 사전 학습 모델

The screenshot shows the PyTorch website's documentation page for 'Models and pre-trained weights'. The top navigation bar includes links for Learn, Ecosystem, Edge, Docs, Blogs & News, About, and Become a Member. The left sidebar has links for Package Reference, Transforming and augmenting images, TVTensors, Models and pre-trained weights (which is highlighted in red), Datasets, Utils, Operators, Decoding / Encoding images and videos, and Feature extraction for model inspection. The main content area has a breadcrumb trail 'Docs > Models and pre-trained weights'. The title 'Models and pre-trained weights' is bolded. Below it, a paragraph explains that the `torchvision.models` subpackage contains definitions of models for tasks like image classification, semantic segmentation, object detection, instance segmentation, person keypoint detection, video classification, and optical flow. A section titled 'General information on pre-trained weights' provides details about how TorchVision offers pre-trained weights for various architectures using the PyTorch `torch.hub`. To the right, there is a sidebar with a 'Shortcuts' button and a list of categories: 'Models and pre-trained weights', '+ General information on pre-trained weights', '+ Classification', '+ Semantic Segmentation', '+ Object Detection, Instance Segmentation and Keypoint Detection', '+ Video Classification', and 'Optical Flow'.

사전 학습 모델 사용 예시

- 사전 학습 가중치 불러오기

```
$ python
```

```
Python 3.10.13 (main, Nov 14 2023, 14:11:52) [Clang 15.0.0 (clang-1500.0.40.1)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> from torchvision.models.detection import RetinaNet_ResNet50_FPN_V2_Weights  
>>> weights = RetinaNet_ResNet50_FPN_V2_Weights.DEFAULT  
>>> print(weights.meta)          # 학습 과정에서 사용한 메타 데이터  
{'categories': ['_background_', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', ...  
...(중간 생략)...  
'_docs': 'These weights were produced using an enhanced training recipe to boost the model accuracy.'}  
>>> type(weights.transforms())    # 학습 과정에서 사용한 전처리 함수  
<class 'torchvision.transforms._presets.ObjectDetection'>
```



사전 학습 모델 사용 예시

- 사전 학습 모델 불러오기 (이어서)

```
>>> from torchvision.models.detection import retinanet_resnet50_fpn_v2  
>>> model = retinanet_resnet50_fpn_v2(weights=weights)
```

Downloading: "https://download.pytorch.org/models/retinanet_resnet50_fpn_v2_coco-5905b1c5.pth" to ...

```
>>> model.eval()
```

RetinaNet(
 (backbone): BackboneWithFPN(
 (body): IntermediateLayerGetter(
 (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
 (relu): ReLU(inplace=True)
 (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
 ...
 (transform): GeneralizedRCNNTransform(
 Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
 Resize(min_size=(800,), max_size=1333, mode='bilinear')
)
)

Q&A



예시 프로젝트 소개

01

예시 프로젝트 소개

02

사전 학습 모델 사용

03

API Server 소개



예시 코드

- <https://github.com/9bow/aca-sample>

- 프로젝트 구성 (Directory)

```
├── client          # API Server 동작 확인을 위한 Client 코드
│   └── data         # API Server 동작 확인 시 사용할 예시 이미지
├── docs            # 프로젝트 설명 문서
└── save-points    # ➡ 단계별 예제 코드
    └── src          # 예제 프로젝트 코드
```

- 저장소 복제

```
$ git clone https://github.com/9bow/aca-sample
```



예시 코드 실행: 이미지 준비

- 예제 이미지 준비: 다운로드 및 저장 (save-points/sample.jpg)



출처: <https://unsplash.com/photos/a-man-standing-on-a-set-of-stairs-using-a-cell-phone-VsDV9j30Sww>



예시 코드 실행: save-points/01

- 예제 스크립트 Python 실행

```
$ cd save-points/01
```

```
$ python example.py
```

- 결과 확인

```
> save-points/01/sample_output.jpg
```



예시 코드: save-points/01/example.py

```
import torchvision
from torchvision.models.detection import retinanet_resnet50_fpn_v2, RetinaNet_ResNet50_FPN_V2_Weights

# 사전 학습 모델 및 정보 불러오기
def load_model_and_preprocess():
    weights = RetinaNet_ResNet50_FPN_V2_Weights.DEFAULT
    model = retinanet_resnet50_fpn_v2(weights=weights)
    model.eval()

    return model, weights.transforms(), weights.meta

# sample.jpg 이미지를 불러와서 모델에 입력으로 전달하고, 예측 결과를 출력
if __name__ == '__main__':
    model, preprocess, meta = load_model_and_preprocess() # 모델 및 전처리 함수 등 불러오기

    img_tensor = torchvision.io.read_image('../sample.jpg') # 예제 이미지 불러오기

    input_batch = [preprocess(img_tensor)] # 입력이 하나뿐이므로, list로 감싸서 묶음(batch)으로 만듦(비추천). 또는,
    # input_batch = preprocess(img_tensor).unsqueeze(0) # unsqueeze()로 맨 앞에 차원을 추가해도 됨 (batch 차원)
```

예시 코드: save-points/01/example.py

```
preds = model(input_batch) # 입력을 모델에 전달하여 예측 결과 얻기
prediction = preds[0]      # 첫번째 입력에 해당하는 첫번째 결과 가져오기
print('예측 결과:', prediction)

# 객체 ID를 순서대로 가져와 객체 이름을 매핑
label_txts = [meta['categories'][class_id] for class_id in prediction['labels']]
print(prediction['labels'], label_txts) # Class ID 및 Label 확인

# draw_bounding_boxes() 함수를 사용하여 탐지된 객체들의 위치를 표시합니다.
tensor_with_boxes = torchvision.utils.draw_bounding_boxes(img_tensor, boxes=prediction['boxes'], labels=label_txts,
                                                          colors='red', width=2)

# draw_bounding_boxes()는 Tensor를 반환하므로, 시각화를 위해 PIL.Image로 변환합니다.
img_with_boxes = torchvision.transforms.v2.functional.to_pil_image(tensor_with_boxes)
img_with_boxes.save('sample_output.jpg') # 결과 이미지 저장
```



예시 코드 실행 결과: save-points/01



출처: <https://unsplash.com/photos/a-man-standing-on-a-set-of-stairs-using-a-cell-phone-VsDV9j30Sww>



Q&A



예시 코드 실행: save-points/02

- 예제 스크립트 Python 실행

```
$ cd save-points/02
```

```
$ python example.py
```

- 결과 확인

```
> save-points/02/sample_output.jpg
```



예시 코드: save-points/02/example.py

```
import torchvision
from torchvision.models.detection import retinanet_resnet50_fpn_v2, RetinaNet_ResNet50_FPN_V2_Weights

# 사전 학습 모델 및 정보 불러오기
def load_model_and_preprocess():
    weights = RetinaNet_ResNet50_FPN_V2_Weights.DEFAULT
    model = retinanet_resnet50_fpn_v2(weights=weights)
    model.eval()

    return model, weights.transforms(), weights.meta

# sample.jpg 이미지를 불러와서 모델에 입력으로 전달하고, 예측 결과를 출력
if __name__ == '__main__':
    model, preprocess, meta = load_model_and_preprocess() # 모델 및 전처리 함수 등 불러오기

    img_tensor = torchvision.io.read_image('../sample.jpg') # 예제 이미지 불러오기

    input_batch = [preprocess(img_tensor)] # 입력이 하나뿐이므로, list로 감싸서 묶음(batch)으로 만듦(비추천). 또는,
    # input_batch = preprocess(img_tensor).unsqueeze(0) # unsqueeze()로 맨 앞에 차원을 추가해도 됨 (batch 차원)
```

예시 코드: save-points/02/example.py

```
preds = model(input_batch) # 입력을 모델에 전달하여 예측 결과 얻기
prediction = preds[0] # 첫번째 입력에 해당하는 첫번째 결과 가져오기
prediction = {k: v[prediction['scores']] >= 0.5} for k, v in prediction.items() # 지정한 임계값(0.5) 미만의 결과를 제외
print('예측 결과:', prediction)
```

객체 ID를 순서대로 가져와 객체 이름을 매핑

```
# label_txts = [meta['categories'][class_id] for class_id in prediction['labels']]
label_txts = [f'{meta['categories'][class_id]} ({score:.2f})'
             for class_id, score in zip(prediction['labels'], prediction['scores'])]
```

```
print(prediction['labels'], label_txts) # Class ID 및 Label 확인
```

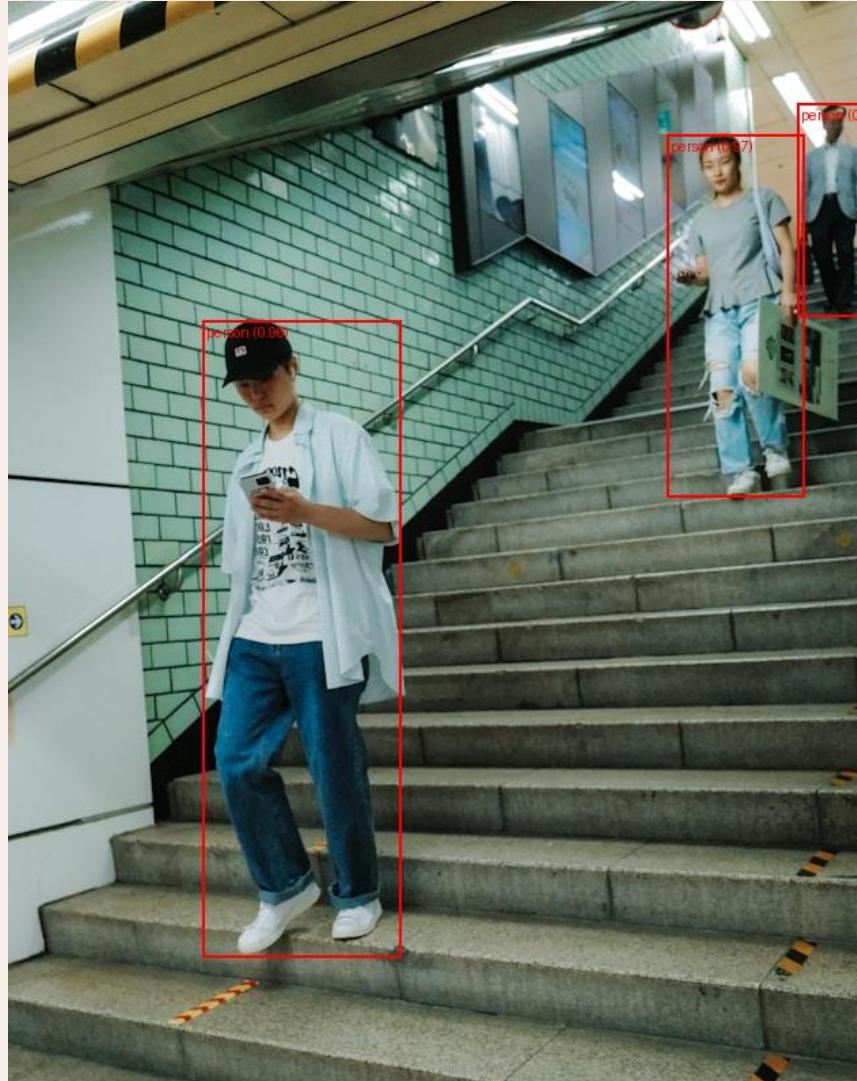
draw_bounding_boxes() 함수를 사용하여 탐지된 객체들의 위치를 표시합니다.

```
tensor_with_boxes = torchvision.utils.draw_bounding_boxes(img_tensor, boxes=prediction['boxes'], labels=label_txts,
                                                          colors='red', width=2)
```

draw_bounding_boxes()는 Tensor를 반환하므로, 시각화를 위해 PIL.Image로 변환합니다.

```
img_with_boxes = torchvision.transforms.v2.functional.to_pil_image(tensor_with_boxes)
img_with_boxes.save('sample_output.jpg') # 결과 이미지 저장
```

예시 코드 실행 결과: save-points/02



출처: <https://unsplash.com/photos/a-man-standing-on-a-set-of-stairs-using-a-cell-phone-VsDV9j30Sww>



Q&A



예제 API Server 소개

- 사진을 입력 받아, 검출한 객체 결과를 반환하는 REST API Server
- PyTorch, torchvision, FastAPI 사용(torch, torchvision, fastapi, python-multipart 등)
- torchvision이 제공하는 사전 학습(Pre-trained) 모델 사용 (RetinaNet)
 - 주요 기능
 - 상태 확인: REST API Server 동작 여부 확인
 - 모델 확인: 사용하는 모델 종류 확인
 - 객체 탐지: 사용자 제공 이미지의 객체 탐지 결과 반환

Object Detection, Instance Segmentation and Person Keypoint Detection

The pre-trained models for detection, instance segmentation and keypoint detection are initialized with the classification models in torchvision. The models expect a list of `Tensor[C, H, W]`. Check the constructor of the models for more information.

• WARNING

The detection module is in Beta stage, and backward compatibility is not guaranteed.

Object Detection

The following object detection models are available, with or without pre-trained weights:

- [Faster R-CNN](#)
- [FCOS](#)
- [RetinaNet](#)
- [SSD](#)
- [SSDlite](#)



예제 소개: API Endpoint 구성

- Endpoint 구성 (1/2)
 - 상태 확인: GET /liveness
 - 응답 코드: 200 OK, 500 Internal Error
 - 응답 예시: {"status": "ok"}
 - 모델 확인: GET /models
 - 응답 코드: 200 OK, 500 Internal Error
 - 응답 예시:

```
{"models": [  
    { "id": "MODEL_ID1", "name": "MODEL_NAME1"},  
    { "id": "MODEL_ID2", "name": "MODEL_NAME2"},  
]}
```



예제 소개: API Endpoint 구성

- Endpoint 구성 (2/2)
 - 객체 탐지: POST /image:detect
 - 응답 코드: 200 OK, 400 Bad Request, 415 Unsupported Media Type, 500 Internal Error
 - 요청 헤더: Content-Type: multipart/form-data
 - 요청 예시: { "image": @이미지파일, "threshold": 0.6 }
 - 응답 예시:

```
{ "objects": [  
    { "class": 1, "label": "person", "score": 0.85, "bbox": [50, 100, 200, 250] },  
    { "class": 1, "label": "person", "score": 0.65, "bbox": [150, 200, 300, 350] },  
    ...  
]
```



예제 코드

- <https://github.com/9bow/aca-sample>

- 프로젝트 구성 (Directory)

```
├── client          # API Server 동작 확인을 위한 Client 코드  
│   └── data        # API Server 동작 확인 시 사용할 예시 이미지  
├── docs            # 프로젝트 설명 문서  
└── save-points    # 단계별 프로젝트 코드(src/)  
    └── src          # ➡️ 예제 프로젝트 코드
```

- 저장소 복제

```
$ git clone https://github.com/9bow/aca-sample
```



예제 코드: src/ 구성

- 프로젝트 구성 (File)

- src/

```
    ├── Dockerfile          # 이미지 구성을 위한 Dockerfile
    ├── app.py              # FastAPI Application
    ├── model.py            # 사전 학습 모델
    ├── requirements.txt    # 의존성 정의
    └── utils.py            # 그 외 함수
```



예제 코드: src/requirements.txt

```
# PyTorch 및 torchvision  
torch  
torchvision  
  
# FastAPI  
fastapi  
uvicorn  
python-multipart
```

- 의존성 설치

\$ **pip install -r src/requirements.txt**



예제 코드: src/model.py

```
from torchvision.models.detection import retinanet_resnet50_fpn_v2, RetinaNet_ResNet50_FPN_V2_Weights

# 앞에서 가져온 가중치를 제공하여 사전 학습된 모델 가져오기
def load_model_and_preprocess(device='cpu'):
    weights = RetinaNet_ResNet50_FPN_V2_Weights.DEFAULT
    model = retinanet_resnet50_fpn_v2(weights=weights)
    model.to(device) # 지정된 장치로 모델 이동
    model.eval() # (학습이 아닌) 추론 모드로 설정

    return model, weights.transforms(), weights.meta

if __name__ == '__main__':
    model, preprocess, meta = load_model_and_preprocess()
    print(model)
    print(preprocess)
    print(meta)
```



예제 코드: src/utils.py

```
# 결과 필터링 함수
def filter_results(outputs, categories, threshold=0.5, klass=None):
    filtered_results = []

    for label, score, box in zip(outputs['labels'], outputs['scores'], outputs['boxes']):
        if score < threshold:
            continue

        if klass is not None and int(label) != klass:
            continue

        filtered_results.append({
            "class": int(label),
            "label": categories[int(label)],
            "score": float(score),
            "bbox": [float(coord) for coord in box]
        })

    return filtered_results
```



예제 코드: src/utils.py

```
# 결과 필터링 함수 동작 확인
if __name__ == "__main__":
    sample_outputs = {
        "labels": [1, 1, 2, 3, 4],
        "scores": [0.9, 0.8, 0.7, 0.6, 0.5],
        "boxes": [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16], [17, 18, 19, 20]]
    }
    sample_categories = {1: "cat", 2: "dog", 3: "bird", 4: "fish"}

    print(filter_results(sample_outputs, sample_categories, 0.75))
```

- 동작 확인

\$ **python src/utils.py**



예제 코드: src/app.py

```
import logging
from io import BytesIO

from fastapi import FastAPI, File, UploadFile
from fastapi.responses import JSONResponse
from PIL import Image
import torch
import torchvision

from model import load_model_and_preprocess
from utils import filter_results

logger = logging.getLogger() # Logger 설정
logger.setLevel(logging.DEBUG)

logger.debug('Creating FastAPI app...')
app = FastAPI()           # FastAPI 앱 생성
logger.debug('FastAPI app created.')
```



예제 코드: src/app.py

```
# 모델 불러오기
logger.debug('Initializing model...')
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model, preprocess, meta = load_model_and_preprocess(device)
logger.debug(f'Model initialized successfully with device: {device}')

# 모델 정보 관리
models = [
    'id': model.__class__.__module__,
    'name': type(model).__name__
]

# HTTP Health Probe
@app.get('/liveness')
def liveness():
    return JSONResponse(content={'status': 'ok'})
```



예제 코드: src/app.py

```
# 모델 정보 엔드포인트
@app.get('/models')
def get_models():
    logger.debug('Returning model information...')
    return JSONResponse(content={'models': models})

# 객체 탐지 엔드포인트
@app.post('/image:detect')
def detect_objects(image: UploadFile, threshold: float = 0.5, klass: int = None):
    logger.debug('Detecting objects in the uploaded image...')
    logger.debug(f'Threshold: {threshold}, Class ID: {klass}')

    try:
        # 이미지 파일이 아닌 경우 예외 발생
        if not image.headers['content-type'].startswith('image/'):
            raise ValueError('Uploaded file is not an image')
```



예제 코드: src/app.py

```
if klass is not None and klass < 0: # 클래스 ID가 주어진 경우, 유효한 클래스 ID인지 확인
    raise ValueError('Invalid class ID')

img_obj = Image.open(BytesIO(image.file.read())) # 업로드된 이미지 파일 열기 (PIL.Image 객체로 변환)

img_input = preprocess(img_obj).to(device) # 전처리
img_input = img_input.unsqueeze(0) # 단일 이미지이므로 배치(batch) 차원 추가

outputs = model(img_input)[0]          # 추론 후, (단일 이미지이므로) 첫번째 결과만 사용

results = filter_results(outputs, meta['categories'], threshold=threshold, klass=klass) # 결과 필터링

logger.debug(f'Objects detected: {len(results)}')
return JSONResponse(content={'objects': results})
except ValueError as e:
    return JSONResponse(content={'error': str(e)}, status_code=415)
except Exception as e:
    return JSONResponse(content={'error': str(e)}, status_code=500)
```



예제 코드: 실행

- API Server 실행

```
$ cd src/
```

```
$ uvicorn src/app:app --port 8080 --reload
```

- (새로운 창, 프로젝트 메인에서) 동작 확인

```
$ curl http://localhost:8080/
```

```
{"Hello": "World"}
```

```
$ curl http://localhost:8080/models
```

```
{"models": [{"id": "torchvision.models.detection.retinanet", "name": "RetinaNet"}]}
```



예제 코드: 실행

- 실행 확인

```
$ curl -X POST -F "image=@save-points/sample.jpg" http://localhost:8080/image:detect  
{"objects": [{"class": 1, "label": "person", "score": 0.9688001275062561, "bbox": [485.87969970703125, 98.0000228881836, 586.0333251953125, 365.99676513671875]}, {"class": 1, "label": "person", "score": 0.9597383141517639, "bbox": [143.8806915283203, 235.3536376953125, 289.22210693359375, 704.5020751953125]}, {"class": 1, "label": "person", "score": 0.9018632769584656, "bbox": [581.9451293945312, 75.80509948730469, 634.7716064453125, 232.50857543945312]}]}
```



예제 코드: Client

- <https://github.com/9bow/aca-sample>

- 프로젝트 구성 (Directory)

└── client	# API Server 동작 확인을 위한 Client 코드
└── data	# API Server 동작 확인 시 사용할 예시 이미지
└── docs	# 프로젝트 설명 문서
└── save-points	# 단계별 프로젝트 코드(src/)
└── src	# 예제 프로젝트 코드

- 저장소 복제

```
$ git clone https://github.com/9bow/aca-sample
```



예제 코드: client/client.py

```
import io
import json
import random
import collections

import requests
import torch
import torchvision

from PIL import Image

from torchvision.utils import draw_bounding_boxes
from torchvision.transforms.v2 import functional as F

import matplotlib.pyplot as plt

# 이미지 파일
fn_img_input = 'data/sample.jpg'
fn_img_output = 'data/sample_output.jpg'
```



예제 코드: client/client.py

```
url = 'http://localhost/image:detect?threshold=0.8&klass=1'
files = [('image', ('sample.jpg', open(fn_img_input, 'rb'), 'image/jpeg'))]

response = requests.post(url, files=files)
if response.status_code != 200:
    print(f'Error: {response.status_code}')
    print(response.text)

results = json.loads(response.text)
objects_types = [result['label'] for result in results['objects']]
objects_counter = collections.Counter(objects_types)
color_type = {result['class']:("#%06x" % random.randint(0, 0xFFFFFF)) for result in results['objects']}
box_coords = torch.stack([torch.tensor(result['bbox']) for result in results['objects']])
box_labels = [f'{result["label"]}{result["score"]:.2f}' for result in results['objects']]
box_colors = [color_type[result['class']] for result in results['objects']]

print(f'검출된 객체 수: {len(results["objects"])}')
print(f'검출된 객체 종류: {objects_counter}')
```



예제 코드: client/client.py

```
# 이미지에 검출 결과 그리기
tensor_with_boxes = draw_bounding_boxes(torchvision.io.read_image(fn_img_input), boxes=box_coords,
                                         labels=box_labels, colors=box_colors, font='Verdana', font_size=20, width=2,)

F.to_pil_image(tensor_with_boxes).save(fn_img_output)

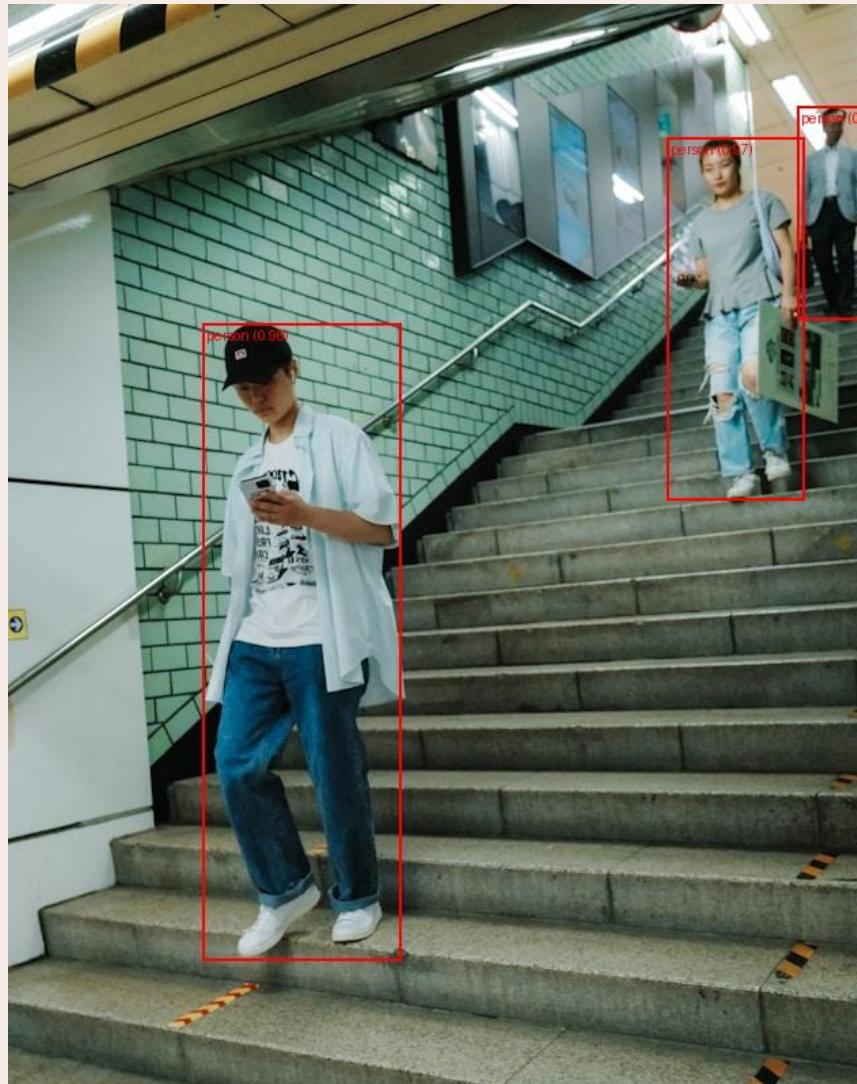
print(f'검출 결과 이미지 저장: {fn_img_output}')
```

- 동작 확인

\$ **python client/client.py**



예시 코드 실행 결과: client/data/sample_output.jpg



출처: <https://unsplash.com/photos/a-man-standing-on-a-set-of-stairs-using-a-cell-phone-VsDV9j30Sww>



Q&A



Container 소개

01

Container 소개

02

Image 만들기

03

Container 실행



Container 소개

- Container =
실행 파일 + 실행 환경

컨테이너 설명

해운 업계에서 실제 컨테이너를 사용하여 선박과 기차에서 운송할 다양한 화물을 분리하는 것처럼 소프트웨어 개발 기술에서도 컨테이너화라는 접근 방식을 점점 더 많이 사용합니다.

컨테이너라고 하는 표준 소프트웨어 패키지는 애플리케이션의 코드를 관련 구성 파일, 라이브러리 및 앱 실행에 필요한 종속성과 함께 번들로 제공합니다. 따라서 개발자와 IT 전문가는 여러 환경에서 애플리케이션을 원활하게 배포할 수 있습니다.

→ “동일한 실행 파일”을 “동일한 환경”에서 실행할 수 있도록 함



Container 소개

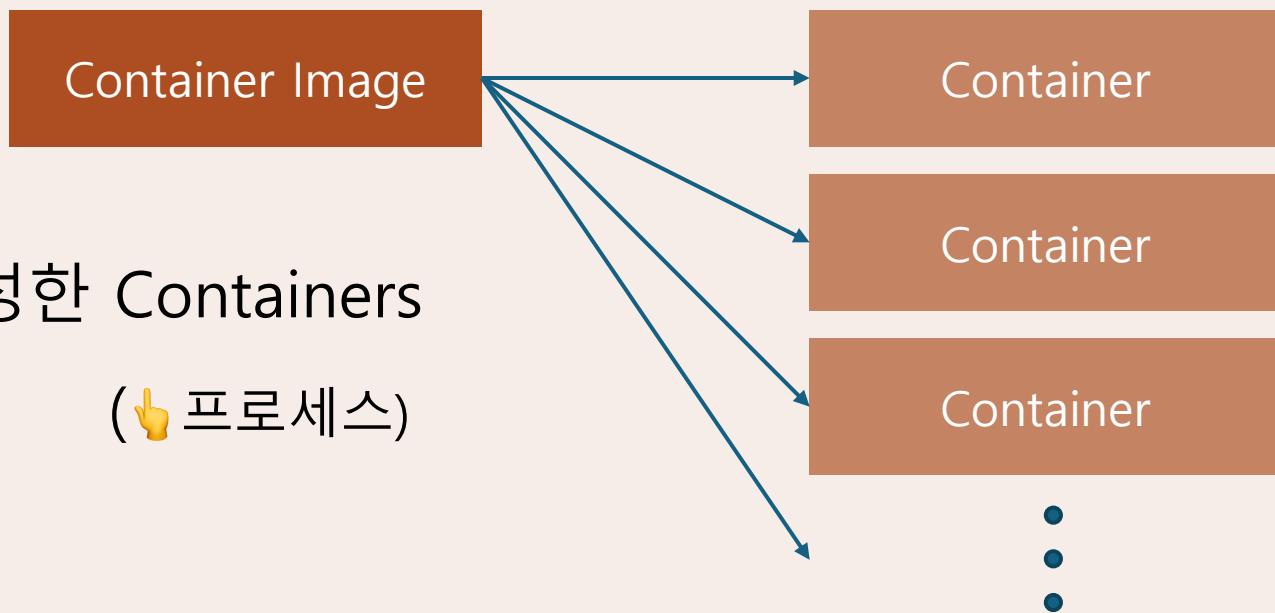
- 장점
 - 간단하고 일관된 배포 가능 → 개발 / 스테이징 / 상용 모두 동일 환경
 - VM(Virtual Machine) 대비 가볍고, 빠르게 확장이 가능 (붕어빵 틀 + 붕어빵)

- 구성

- Image와 이를 가지고 생성한 Containers

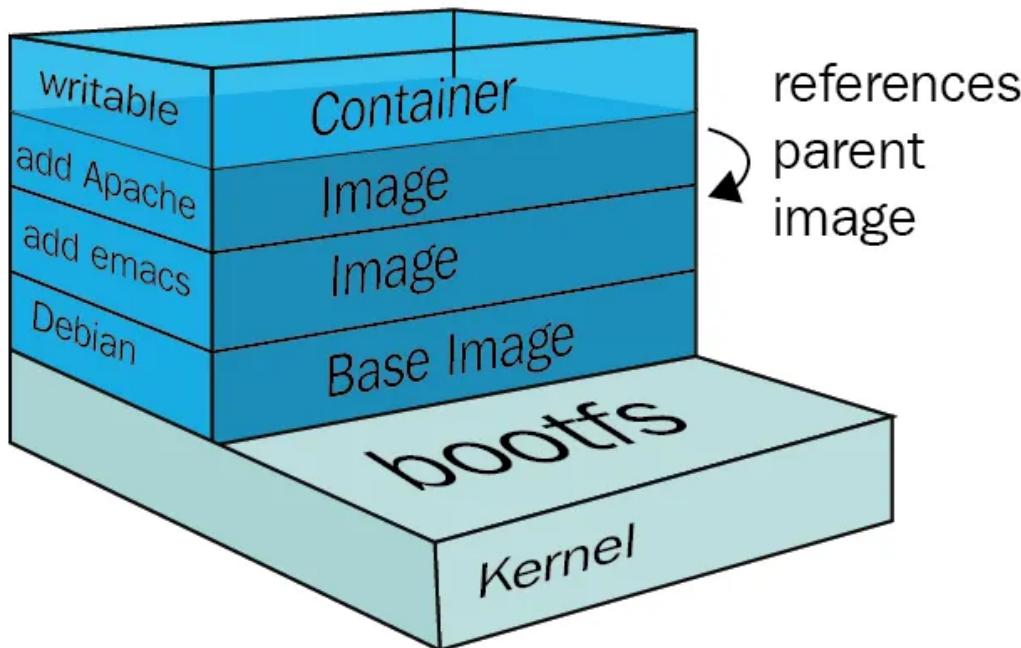
(👉 실행파일)

(👉 프로세스)



Container 소개

- 기존의 Container Image를 기반으로 새롭게 추가 가능



```
# syntax=docker/dockerfile:1
FROM ubuntu:22.04

# install app dependencies
RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip install flask==3.0.*

# install app
COPY hello.py /

# final configuration
ENV FLASK_APP=hello
EXPOSE 8000
CMD ["flask", "run", "--host", "0.0.0.0", "--port", "8000"]
```

Container 예시: src/Dockerfile

```
# Python:3.10 이미지를 가져와서 구성
FROM python:3.10
WORKDIR /app

# 의존성을 정의한 requirements.txt 파일 복사
COPY requirements.txt .
RUN pip3 install --no-cache-dir -r requirements.txt

# 나머지 파일 복사 (src/ 내의 모든 파일 복사)
COPY . .
RUN python3 app.py

# 그 외 실행 시 옵션
EXPOSE 80
CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "80"]
```



Container Image 만들기 (w/ Docker)

- (Dockerfile이 위치한 곳에서) 이미지 만들기 (= src/)

```
$ docker build . -t <image-name>(:<tag>)
```

- Apple Silicon(M1 ~ M4 등)과 같은 ARM 기반 CPU의 경우

```
$ docker build . -t <image-name> --platform=linux/amd64
```

Azure Container Apps는 다음을 지원합니다.

- Linux 기반 x86-64(linux/amd64) 컨테이너 이미지 (= ARM 지원 X)
- 공용 또는 프라이빗 컨테이너 레지스트리의 컨테이너
- 선택적 사이드카 및 init 컨테이너



Container Image 만들기 (w/ Docker)

- (Dockerfile이 위치한 곳에서) 이미지 만들기 (= src/)

```
$ docker build . -t sample-app:latest (:latest는 생략 가능)
```

(+ Apple Silicon의 경우 --platform=linux/amd64 추가 필요)

[+] Building 40.5s (10/11)

=> [internal] load build definition from Dockerfile	0.0s
=> => transferring dockerfile: 333B	0.0s
=> [internal] load metadata for docker.io/library/python:3.10	2.3s
=> [internal] load .dockerignore	0.0s
=> => transferring context: 2B	0.0s
=> [1/6] FROM docker.io/library/python:3.10@sha256:3ba2e48b...	11.3s
=> => resolve docker.io/library/python:3.10@sha256:3ba2e48b...	0.0s
... (중간 생략) ...	

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sample-app	latest	d8d70d43837c	2 minutes ago	2.55GB



Container 실행

- Host에서 여러 개의 Container 실행 가능
→ 단, Container는 격리된 환경으로, Host 자원 사용시 설정 필요
- 일반적인 실행 (-d는 Background 실행, :latest는 생략 가능)

```
$ docker run -d sample-app:latest
```

DEBUG: Creating FastAPI app...

DEBUG: FastAPI app created.

DEBUG: Initializing model...

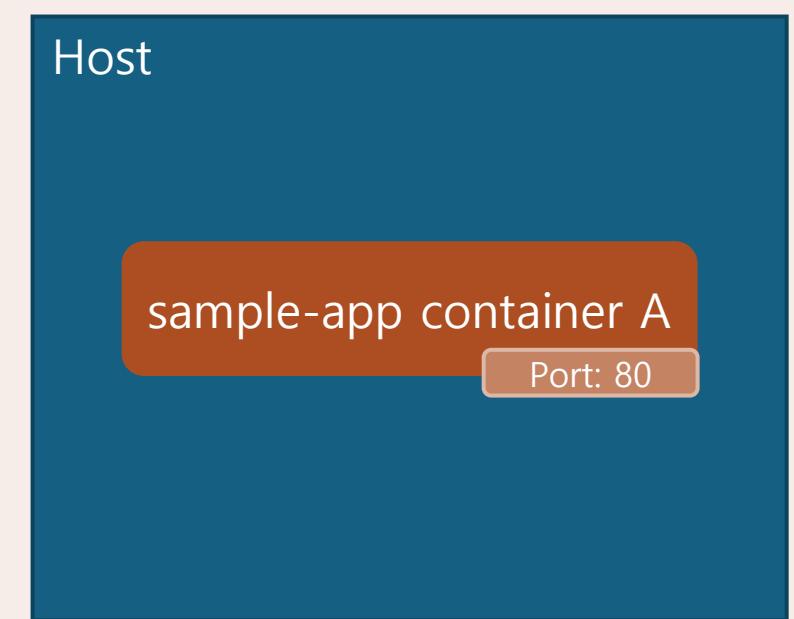
DEBUG: Model initialized successfully with device: cpu

INFO: Started server process [1]

INFO: Waiting for application startup.

INFO: Application startup complete.

INFO: Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)



Container 실행

- Host에서 여러 개의 Container 실행 가능
→ 단, Container는 격리된 환경으로, Host 자원 사용시 설정 필요
- Host와 Container의 Port Mapping 실행
\$ docker run -d -p 8080:80 sample-app

DEBUG: Creating FastAPI app...

DEBUG: FastAPI app created.

DEBUG: Initializing model...

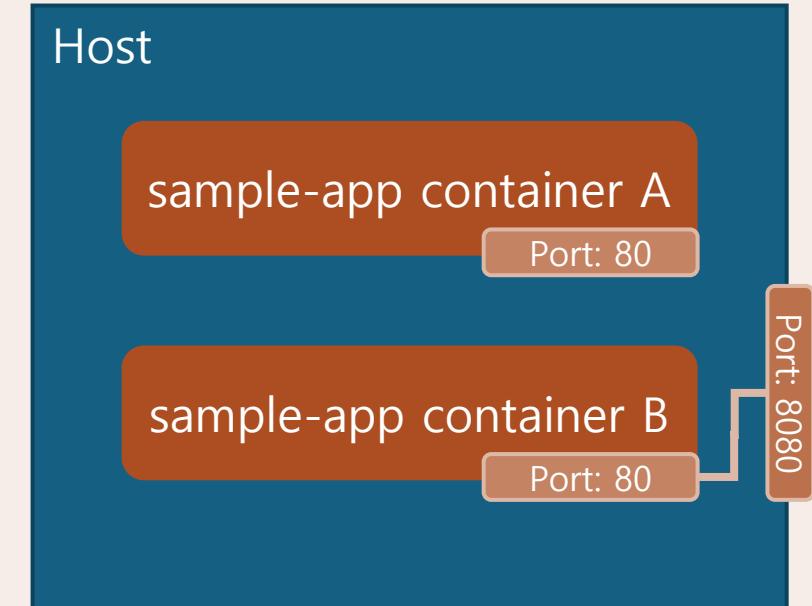
DEBUG: Model initialized successfully with device: cpu

INFO: Started server process [1]

INFO: Waiting for application startup.

INFO: Application startup complete.

INFO: Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)



Container 동작 확인

- 실행 중인 Container 목록 확인 → ID 또는 이름으로 식별 가능

```
$ docker ps
```

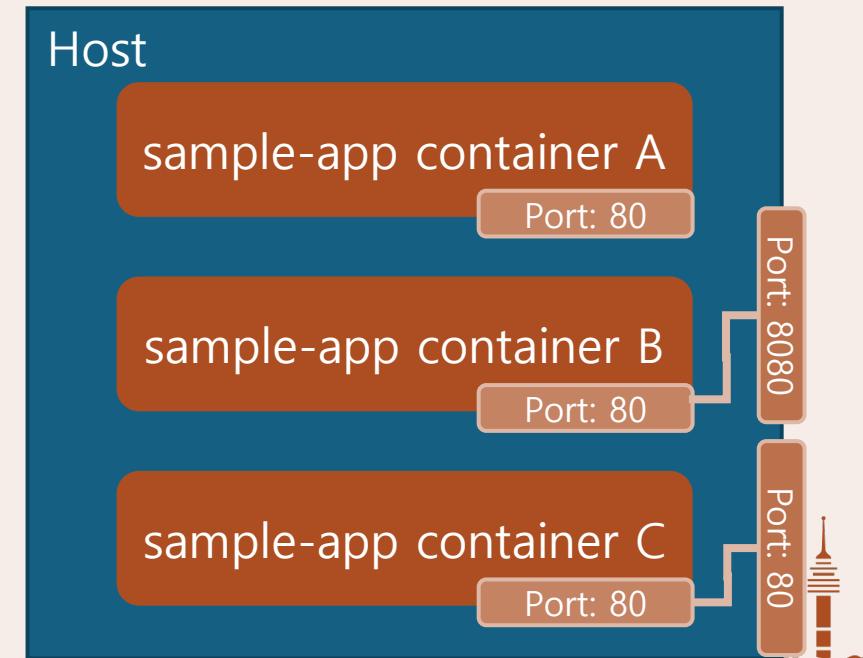
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
99eddc910309	sample-app	"uvicorn app:app --h..."	4 minutes ago	Up 4 minutes	0.0.0.0:8080->80/tcp	optimistic_kepler
20478a26f86e	sample-app	"uvicorn app:app --h..."	4 minutes ago	Up 4 minutes	80/tcp	bold_banach

- Host:80 → Container:80 매핑으로 추가 실행

```
$ docker run -d -p 80:80 sample-app
```

- Client를 실행하여 동작 확인

```
$ python client.py
```



Container 종료

- 식별 가능한 ID 또는 이름으로 종료 가능

\$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9f63072c51f2	sample-app	"uvicorn app:app --h..."	52 seconds ago	Up 51 seconds	0.0.0.0:80->80/tcp	frosty_bell
99eddc910309	sample-app	"uvicorn app:app --h..."	4 minutes ago	Up 4 minutes	0.0.0.0:8080->80/tcp	optimistic_kepler
20478a26f86e	sample-app	"uvicorn app:app --h..."	4 minutes ago	Up 4 minutes	80/tcp	bold_banach

\$ docker stop 9f 99 20

9f

99

20

\$ docker ps



Azure Container Apps 소개

01

Azure Container 서비스

02

Container Registry 소개

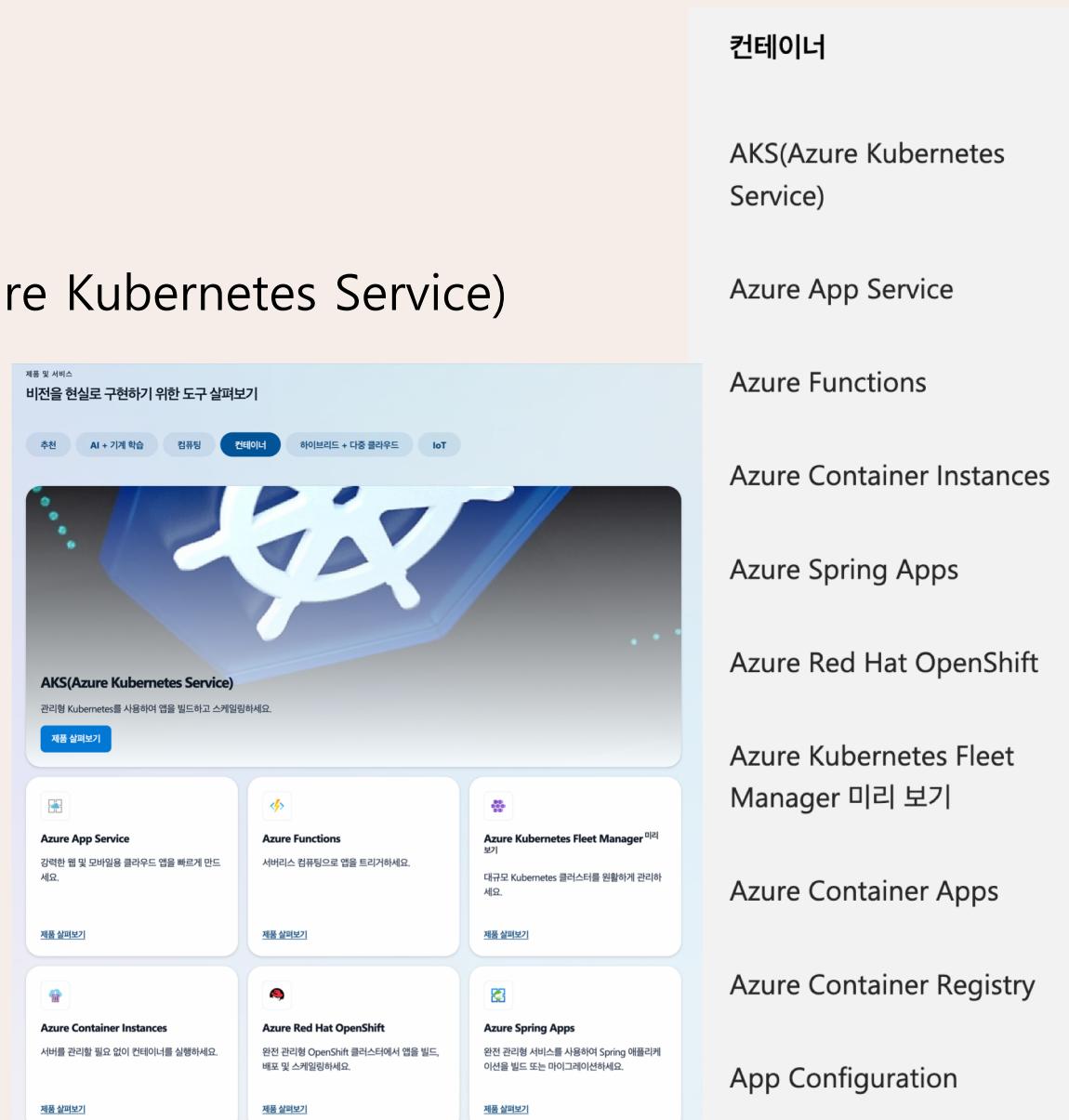
03

Container App 소개



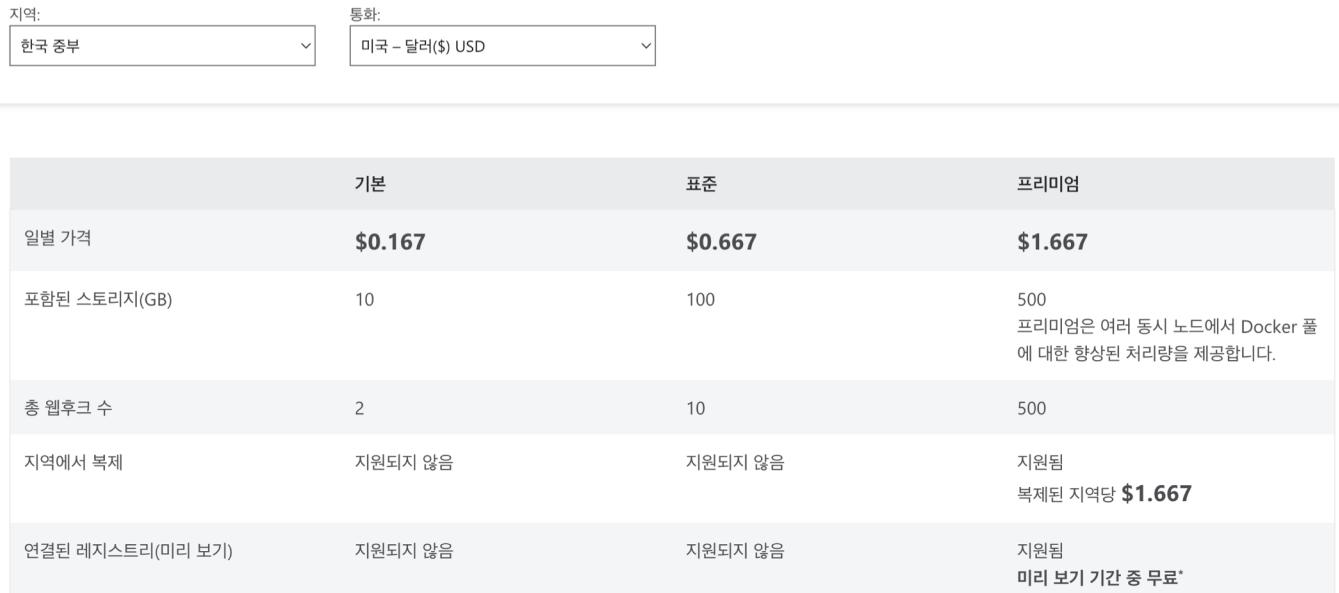
Azure Container 서비스

- 다양한 컨테이너 서비스 제공 중
- Azure Functions $\leftarrow \sim\sim\sim \rightarrow$ AKS (Azure Kubernetes Service)
- 상황에 따른 자유도와 편의성 선택
(=높은 자유도 & 편의성 \propto 가격)
- 적절한 서비스 선택이 중요
- 참고 문서 링크



Azure Container Registry 소개

- Container Image를 저장할 수 있는 별도의 공간 (예. Docker Hub)
- Private Registry 생성 및 이미지 저장 → 다른 서비스에서 활용
- 계층 및 용량에 따른 과금 



지역:	통화:	
한국 중부	미국 – 달러(\$) USD	
일별 가격	\$0.167	
포함된 스토리지(GB)	10	
총 웹후크 수	2	
지역에서 복제	지원되지 않음	
연결된 레지스트리(미리 보기)	지원되지 않음	
기본	표준	프리미엄
\$0.167	\$0.667	\$1.667
100	500	프리미엄은 여러 동시 노드에서 Docker 풀에 대한 향상된 처리량을 제공합니다.
10	500	
지원되지 않음	지원됨	
	복제된 지역당 \$1.667	
		지원됨
		미리 보기 기간 중 무료*

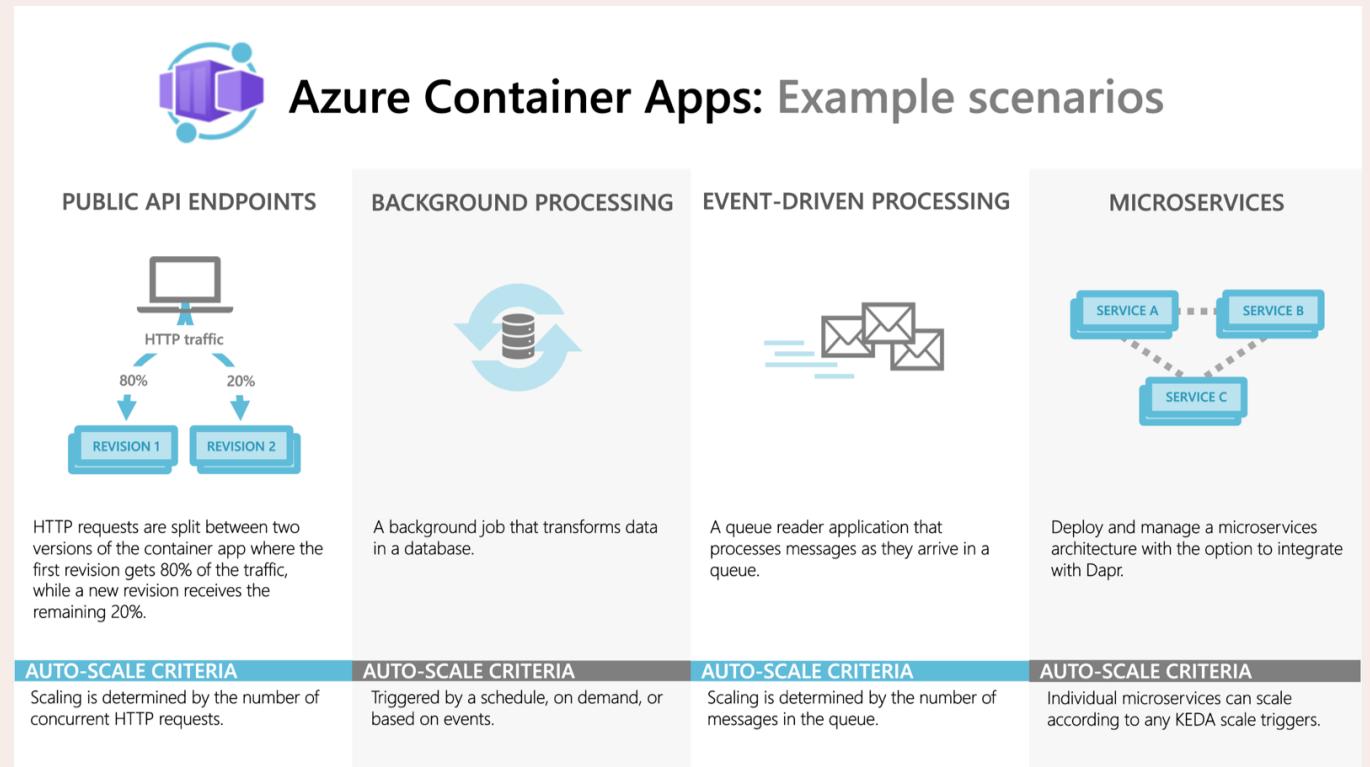
*연결된 레지스트리(미리 보기)는 정의된 동기화 일정 및 네트워크 연결에 따라 온-프레미스와 원격 위치 모두에서 로컬로 컨테이너화된 워크로드에 빠르게 액세스할 수 있도록 해 줍니다. 미리 보기 기간 중에는 이 기능에 대한 요금이 청구되지 않습니다.
연결된 레지스트리에 대해 자세히 알아보세요.

표준 네트워킹 요금이 적용됩니다.

자세한 내용: [레지스트리 계층 기능](#)

Azure Container Apps 소개

- Container Application 실행을 위한 서비스 플랫폼(Serverless Platform)
- 별도의 서버 구성이나 컨테이너 관리 등의 번거로움 없이 사용 가능
- 주로 다음의 용도로 사용👉
 - API Backend 배포
 - 후순위 처리 작업 호스팅
 - 이벤트 기반 처리 수행
 - 마이크로서비스 실행



Q&A



Azure Container Apps 배포

01

Azure CLI 설정

02

Container Registry 사용

03

Container App 배포



Azure CLI 설정

- Azure CLI 설치 및 설정: [참고](#)
- Azure CLI를 사용하여 Azure Login 및 Container Apps 확장 추가

```
$ az login      # az upgrade 실행 시 최신 버전 여부 확인 가능  
                # az extension add --name containerapp --upgrade
```

```
$ az group list --query "[?location='koreacentral']" # Resource Group 조회
```

```
$ az group create --name <이름> --location koreacentral
```

```
$ az group create --name DefaultRG --location koreacentral
```



Azure Container Registry 생성

- Azure CLI를 사용하여 Azure Container Registry 설정

```
$ az acr list          # 생성된 ACR(Azure Container Registry)이 존재하는지 확인
```

```
$ az acr create --name <고유한 이름> --resource-group <RG이름> --sku <서비스계층>
```

```
$ az acr create --name acrfordddseoulworkshop --resource-group DefaultRG --sku Basic
```

1

"adminUserEnabled": false

`"anonymousPullEnabled": false,`

...(중간 생략)...

`"loginServer": "acrfordddseoulworkshop.azurecr.io",`

...(*이하 생략*)...



Azure Container Registry 사용

- Azure CLI를 사용하여 Azure Container Registry 설정

```
$ az acr login -n acrfordddseoulworkshop -g DefaultRG # -n은 --name, -g는 --resource-group
```

Login Succeeded

```
$ docker images
```

<i>REPOSITORY</i>	<i>TAG</i>	<i>IMAGE ID</i>	<i>CREATED</i>	<i>SIZE</i>
<i>sample-app</i>	<i>latest</i>	<i>f38113fecb41</i>	<i>2 hours ago</i>	<i>12.9GB</i>

```
$ docker tag sample-app:latest acrfordddseoulworkshop.azurecr.io/sample-app:latest
```

```
$ docker images
```

<i>REPOSITORY</i>	<i>TAG</i>	<i>IMAGE ID</i>	<i>CREATED</i>	<i>SIZE</i>
<i>acrfordddseoulworkshop.azurecr.io/sample-app</i>	<i>latest</i>	<i>f38113fecb41</i>	<i>2 hours ago</i>	<i>12.9GB</i>
<i>sample-app</i>	<i>latest</i>	<i>f38113fecb41</i>	<i>2 hours ago</i>	<i>12.9GB</i>



Azure Container Registry 사용

- Azure Container Registry에 이미지 올리기(Push) 및 확인(Repository List)

```
$ docker push acrfordddseoulworkshop.azurecr.io/sample-app
```

Using default tag: latest

The push refers to repository [acrfordddseoulworkshop.azurecr.io/sample-app]

5bd71677db44: Pushed

68f3e6d9bfc5: Pushed

4ac722d9cf93: Pushed

...(이하 생략)...

```
$ az acr repository list --name acrfordddseoulworkshop # 해당 저장소의 이미지 목록 조회
```

[

"sample-app"

]



Azure Container App 배포: create

- Azure Container Registry에 올린 이미지로 Container App 생성

```
$ az containerapp env create -n ContainerAppEnv -g DefaultRG --location koreacentral
```

No Log Analytics workspace provided.

Generating a Log Analytics workspace with name "workspace-efaultSPAZ"

/ Running ..

(& 환경 구성 필요)

```
$ az containerapp create --name sample-container-app --resource-group DefaultRG --image acrfordddseoulworkshop.azurecr.io/sample-app:latest --target-port 80 --ingress external --query properties.configuration.ingress.fqdn --cpu 2 --memory 4Gi --environment ContainerAppEnv
```

- 또는, -yaml 옵션으로 yaml로 설정을 정의하고 생성시 사용 가능



Azure Container App 배포: up

- 기본 설정(vCPU: 0.5, Mem 1Gi)으로 신속하게 Container App 배포 가능

```
$ az containerapp up --name sample-container-app --resource-group DefaultRG --location koreacentral --image acrfordddseoulworkshop.azurecr.io/sample-app:latest --target-port 80 --ingress external --query properties.configuration.ingress.fqdn
```

Using resource group 'DefaultRG'

Creating ContainerAppEnvironment 'sample-container-app-env' in resource group DefaultRG

No Log Analytics workspace provided.

Generating a Log Analytics workspace with name "workspace-efault6opU"

/ Running ..

- **az containerapp update** 명령어로 배포 후 설정 변경 가능 (계속)



Azure Container App 변경: update

- 배포한 Container App 확인

```
$ az containerapp show -n sample-container-app -g DefaultRG
```

```
{
```

```
"id": "/subscriptions/...(생략).../resourceGroups/DefaultRG/...(생략).../sample-container-app",
"identity": {
...(중간 생략)...
"name": "sample-container-app",
...(중간 생략)...
"fqdn": "sample-container-app.ashymushroom-b307a8b4.koreacentral.azurecontainerapps.io",
...(0/하 생략)...
```



Azure Container App 변경: update

- 변경 가능한 설정 예시
 - Container App의 Spec 변경: vCPU / RAM, (내부) Storage 설정 등
 - Probe 설정: startup(시작), liveness(실행 중), readiness(replica 준비 완료)
 - 크기(replica) 조정 등

- Container Apps 설정을 확인하고 업데이트하기 (YAML)

```
$ az containerapp show -n sample-container-app -g DefaultRG -o yaml > app.yaml
```

- 설정 확인하기

```
$ code app.yaml
```



Azure Container App 변경: update

```
id: /subscriptions/(생략)/resourceGroups/DefaultRG/providers/Microsoft.App/containerapps/sample-container-app
identity:
  principalId: 8b3b9760-006c-48f4-9257-... (생략)
type: SystemAssigned
location: Korea Central
name: sample-container-app ➔...(중간 생략)...
properties:
  template:
    containers:
      image: acrfordddseoulworkshop.azurecr.io/sample-app:latest ➔...(0/하 생략)...
      imageType: ContainerImage
      name: sample-container-app ➔...(0/하 생략)...
    resources:
      cpu: 0.5 ➔...(0/하 생략)...
      ephemeralStorage: 2Gi ➔...(0/하 생략)...
      memory: 1Gi ➔...(0/하 생략)...
```

Azure Container App 변경: update

- 변경한 Container Apps 설정 반영하기

```
$ az containerapp update -n sample-container-app -g DefaultRG --yaml app.yaml
```

```
{
```

```
  "id": "/subscriptions/(생략)/resourceGroups/DefaultRG/(생략)/containerapps/sample-container-app",
```

```
  "identity": {
```

```
    "principalId": "31e7f460-6033-4c53-b292-(생략)",
```

```
    "tenantId": "9d22d751-cf84-480e-89f2-(생략)",
```

```
    "type": "SystemAssigned"
```

```
,
```

```
...(0/하 생략)...
```



Azure Container App 변경: update

- 변경한 Container Apps 설정 확인하기

```
$ az containerapp show -n sample-container-app -g DefaultRG -o yaml
```

...(생략)...

```
- image: acrfordddseoulworkshop.azurecr.io/sample-app:latest
```

```
  imageType: ContainerImage
```

```
  name: sample-container-app
```

```
  resources:
```

```
    cpu: 2.0
```

```
    ephemeralStorage: 8Gi
```

```
    memory: 4Gi
```

...(0/하 생략)...



Azure Container App 확인

- 배포한 Container App 동작 확인

```
$ curl https://sample-container-app.delightfulplant-4bdc5c56.koreacentral.azurecontainerapps.io/  
{"Hello": "World"}
```

```
$ curl -X POST -F "image=@save-points/sample.jpg" https://sample-container-app.delightfulplant-  
4bdc5c56.koreacentral.azurecontainerapps.io/image:detect  
{"objects": [{"class": 1, "label": "person", "score": 0.9688001275062561, "bbox": [485.87969970703125, 98.00002288  
81836, 586.0333251953125, 365.99676513671875]}, {"class": 1, "label": "person", "score": 0.9597384333610535, "b  
box": [143.88067626953125, 235.3536376953125, 289.2221374511719, 704.5020751953125]}, {"class": 1, "label": "  
person", "score": 0.9018634557723999, "bbox": [581.9451293945312, 75.80508422851562, 634.7716064453125, 2  
32.5085906982422]}]}
```



Azure Container App 배포

- 로그 확인

```
$ az containerapp logs show -n sample-container-app -g DefaultRG --follow
```

```
{"TimeStamp": "2024-12-05T15:08:53.99885", "Log": "Connecting to the container 'sample-container-app'..."}  
{"TimeStamp": "2024-12-05T15:08:54.08841", "Log": "Successfully Connected to container: 'sample-container-app' [Revision: 'sample-container-app--r0djh6a-5b7947f88c-qjxgv', Replica: 'sample-container-app--r0djh6a']"}  
{"TimeStamp": "2024-12-05T15:08:54.0887587Z", "Log": "DEBUG: Creating FastAPI app..."}  
{"TimeStamp": "2024-12-05T15:08:54.0888477Z", "Log": "DEBUG: FastAPI app created."}  
{"TimeStamp": "2024-12-05T15:08:54.0889089Z", "Log": "DEBUG: Initializing model..."}  
{"TimeStamp": "2024-12-05T15:08:54.0889589Z", "Log": "DEBUG: Model initialized successfully with device: cpu"}  
{"TimeStamp": "2024-12-05T15:08:54.0890092Z", "Log": "INFO: Started server process [1]"}  
출처: https://learn.microsoft.com/ko-kr/azure/container-app/logging/
```



사용한 Resource 확인 및 삭제

- 삭제할 Resource 확인 (⚠️ 삭제할 리소스가 맞는지 name & type 확인 필요 ⚠️)

```
$ az resource list
```

```
{  
    ...  
    "id": "/subscriptions/(생략)/resourceGroups/DefaultRG/(생략)/containerApps/sample-container-app",  
    ...  
    "type": "Microsoft.App/containerApps"  
}
```



- Resource 삭제

```
$ az resource delete --ids /subscriptions/... /subscriptions/...
```



Q&A



여러분의 의견을 나눠주세요!

이 세션에서 경험한 여러분의 **소중한 의견**을 들려주세요. 여러분께서 나눠주신 의견은 다음 행사를 준비할 때 더 나은 모습을 보여드리기 위해 사용합니다.

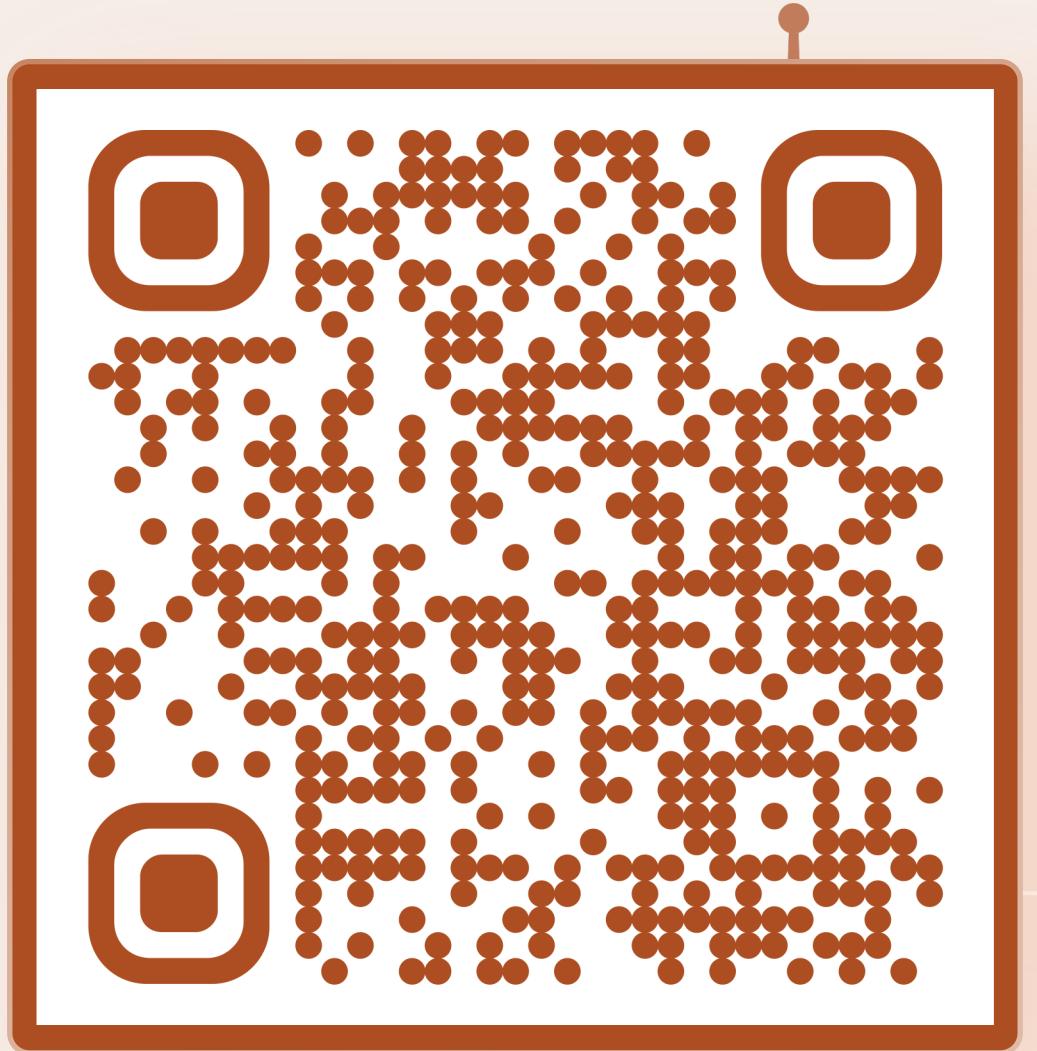
의견을 나눠주신 분을 대상으로 **경품추첨의 기회**도 있으니 놓치지 마세요!



DDD Seoul YouTube 채널

저희 DDD Seoul은 유튜브 채널도 함께 운영하고 있습니다.
오늘 참석해주신 여러분께서 다른 세션에 참석하지 못해 아
쉬움이 남았거나 내가 본 세션을 다시 찾아보길 희망할 경우
이곳 주소로 이동하시면 해당 세션을 다시보실 수 있습니다.

<https://www.youtube.com/@dddseoul>



감사합니다

