애저 서버리스에서 파이썬/장고 웹서비스 쉽게 배포/운영하기

애저펑션과 파이썬 구동 환경 소개

여러분의 파이썬/장고/리액트 페이스메이커가 되겠습니다.

세션 목차

- EP01. 애저펑션과 파이썬 구동 환경 소개
- EP02. 애저펑션 개발환경 구축 및 기본 프로젝트 생성
- EP03. 장고 애플리케이션 소개 및 애저펑션 연동
- EP04. Azure SQL Database와 장고 연동
- EP05. Azure Blob Storages와 장고 연동
- EP06. 애저펑션 인프라로 배포하기
- EP07. 리액트 웹 애플리케이션 배포와 애저 장고 API 연동
- EP08. 마무리

Agenda

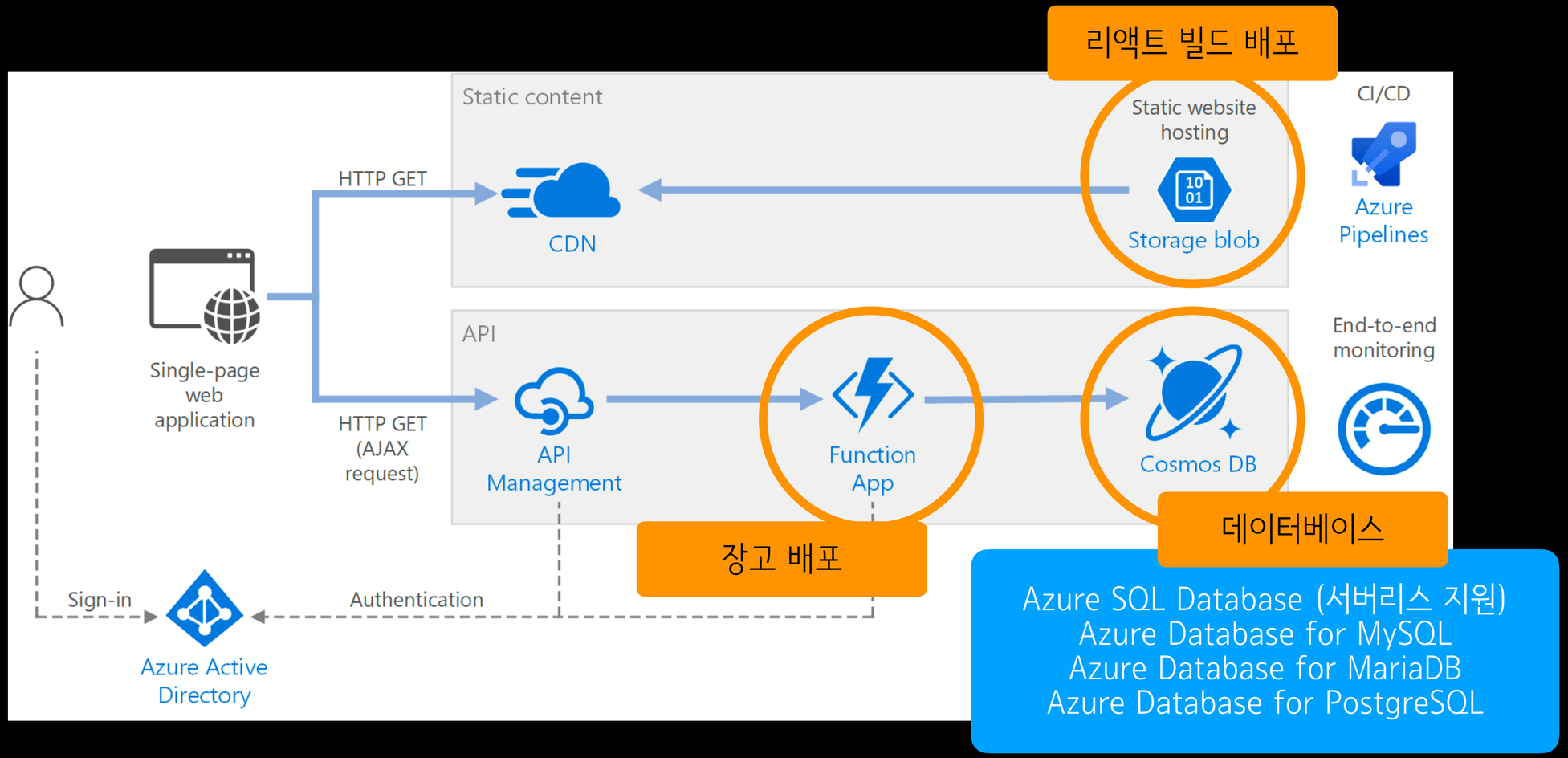
- 서버리스 웹 애플리케이션 아키텍처
- 애저 펑션 소개 (입/출력 바인딩)
- Blob Storage Trigger 예시 및 Http Trigger 예시
- 애저 펑션의 소비 계획 (Consumption Plan) 요금제 소개 및 요금 계산 예

서버리스 웹 애플리케이션

- 서버 관리는 클라우드 벤더에서 해주고, 고가용성을 가집니다.
  - Azure Functions, AWS Lambda, Google Cloud Functions 등
- 유휴 용량에 대한 비용을 지불할 필요가 없습니다.
  - 접속자가 없을 때에는 비용이 지불되지 않습니다. 단 Cold start가 존재.

=> 비즈니스 로직에 집중할 수 있습니다.

애저 서버리스 웹 애플리케이션 아키텍처



[아키텍처] Azure의 서버리스 웹 애플리케이션

애저 펑션 소개 (1/2)

- Azure의 서버리스 컴퓨팅 서비스 (1백만/월 실행까지 무료제공)
- 다양한 언어 지원 (C#, Java, JavaScript, Python, F#, Powershell)를 지원하며, Custom Handler를 통해 Rust, Go, R, Cobol 등의 언어를 사용할 수도 있습니다.
- 다양한 트리거에 의해 호출이 되는 함수(모듈)들을 호스팅
  - Http, Timer, Cosmos DB, Blob Storage, 다양한 큐/메세지 (Queue storage, Event Grid, Event Hub, Service Bus Queue, Service Bus Topic) 등
- Azure Functions 확장으로서 Orchestration 지원 (Duration Functions)

출처 : (공식문서) Azure Functions 소개

애저 펑션 소개 (2/2)

- 통합 인증 지원 : Azure Active Directory, Facebook, Google, Microsoft 등
- Azure Application Insights를 통한 로깅 및 모니터링 내장 지원
- Headless Chromium 지원 (Linux 소비계획, puppeteer/playwright)
- 오픈소스이기에, AWS나 별도 서버에서도 구동 가능

출처 : (공식문서) Azure Functions 소개

입/출력 바인딩

- Azure Functions에서 기본 지원하는 바인딩(Binding)을 통해, 별도 라이브러리나 세팅없이도 관련 리소스에 대한 I/O 수행 가능
  - ex) Http, Cosmos DB, Queue, Blob Storage, Microsoft Graph Excel 테이블, Microsoft Graph OneDrive 파일, Microsoft Graph Outlook 이메일, Notification Hubs, SendGrid, SignalR, Table Storage, Twilio 등

Blob Storage Trigger 예시

- Azure Blob Storage 라이브러리 설치 불필요

함수/funtion.json

```
{
  "scriptFile": "__init__.py",
  "disabled": false,
  "bindings": [
    {
      "name": "myblob",
      "type": "blobTrigger",
      "direction": "in",
      "path": "samples-workitems/{name}",
      "connection": "MyStorageAccountAppSetting"
    }
  ]
}
```

함수/__init__.py

```
import logging
import azure.functions as func


def main(myblob: func.InputStream):
    logging.info('Python Blob trigger function processed %s', myblob.name)
```

Azure Functions 활용 예

- Http Trigger : 웹 API, 웹 서비스, 챗봇 WebHook (텔레그램 등)
- Timer Trigger : 주기적으로 크롤링한 데이터를 Table Storage (NoSQL) 및 Blob Storage에 저장
- Blob Storage 파일 변경에 대한 자동 처리 => 이메일, 문자, 푸쉬로 요약 알림
- Excel/OneDrive 변경에 대한 자동 처리

Trigger: Http, Output Binding: Http

myfuntion/function.json

```
{
    "scriptFile": "__init__.py",
    "bindings": [
        {
            "authLevel": "function",
            "type": "httpTrigger",
            "direction": "in",
            "name": "req",
            "methods": [
                "get",
                "post"
            ]
        },
        {
            "type": "http",
            "direction": "out",
            "name": "$return"
        }
    ]
}
```

myfuntion/__init__.py

```
import azure.functions


def main(req: azure.functions.HttpRequest) -> str:
    user = req.params.get('user')
    return f'Hello, {user}!'
```

WSGI 빌트인 지원

cf. AWS Lambda에서는 써드파티 zappa 라이브러리 활용

- 현 장고 프로젝트를 그대로 가져와서 사용할 수 있습니다.
  - OS : Debian Linux 10 혹은 윈도우 지원. Docker 지원. (AWS Lambda는 Amazon Linux)
- requirements.txt 자동 원격 설치 지원
  - 클라이언트 머신이 윈도우라도 OK (zappa에서는 Docker를 통한 빌드)

```python
import azure.functions


def main(req):
    user = req.params.get('user')
    return f'Hello, {user}!'
```

```python
import sys
sys.path.insert(0, 'dj_proj')

import azure.functions as func
from dj_proj.wsgi import application

wsgi = func.WsgiMiddleware(app=application)

def main(req: func.HttpRequest, context: func.Context) -> func.HttpResponse:
    return wsgi.main(req, context)
```

빌트인 지원하는 파이썬 버전

- 파이썬 3.6, 3.7, 3.8을 지원하며, 함수앱(프로젝트) 생성 시에 버전을 지정
- Linux OS (Debian 10 기반)
  - 공식 Dockerfile
  - mysqlclient, SQL Server 드라이버, OpenCV, OpenMP 등이 기본 설치

요금제

다양한 요금제

- 소비 계획 (Consumption plan)
  - 실행된 시간에 대한 과금. Auto Scale.
  - 20여분 동안 접속이 없으면 유휴 상태로 진입.
    - 유휴상태에서 깨어나는 Cold start 있음 (약 40여초)
    - 즉각적인 응답을 요하는 웹서비스에서는 이슈가 될 수 있음.
- 프리미엄 계획 (Premium plan) : 전용 VM을 미리 준비. VM 과금. 향상된 성능 등
- 앱서비스 계획 (App Service plan)
  - 다른 App Service의 ASP를 공유하여, 추가 비용 X (ASP 단위 VM 과금)

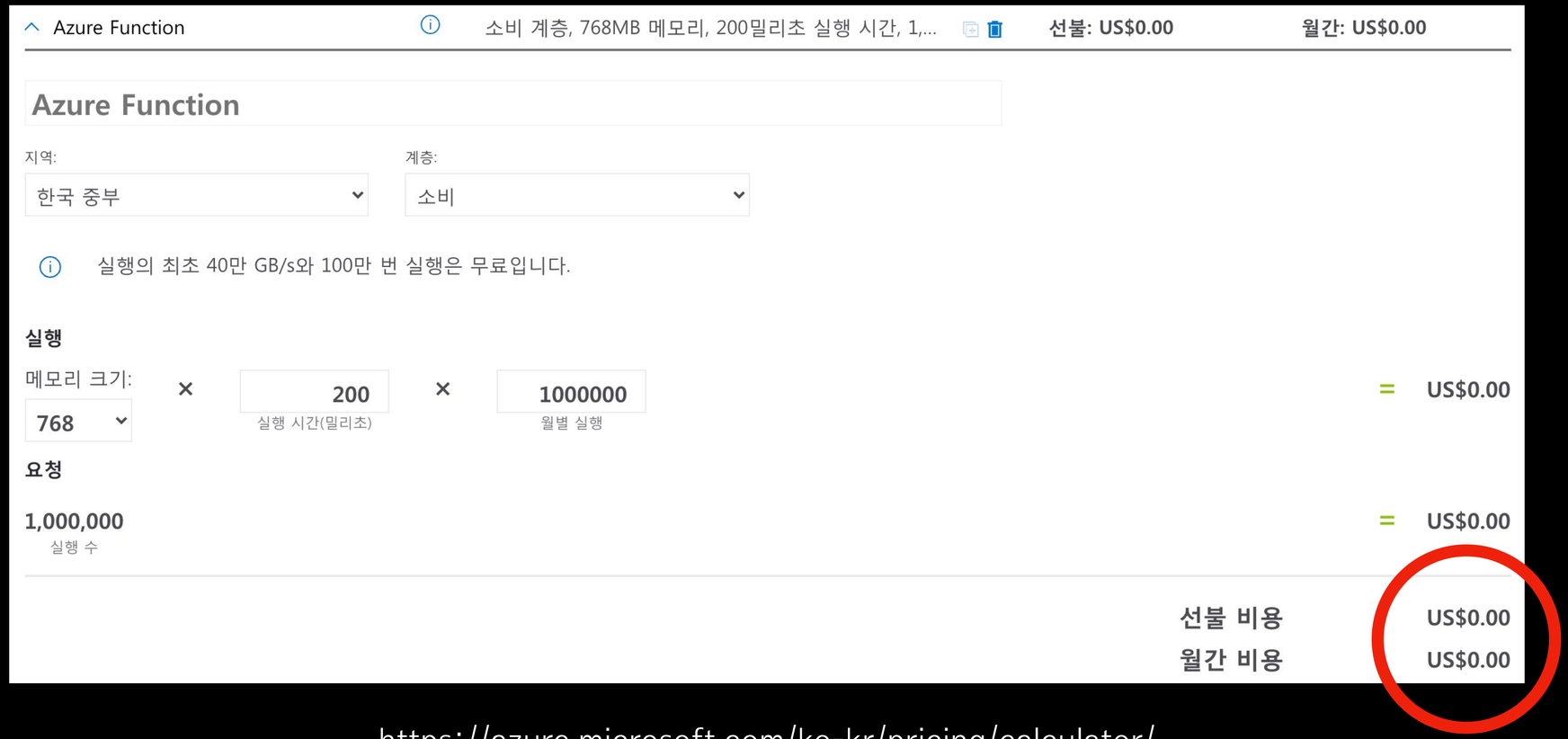
리소스 사용량 계산

- 리소스 사용량 = 평균 메모리 크기 (GB) * 실행시간 (ms)

미터	가격	무료 제공 (월)
실행 시간	0.017995원/GB-s	40만 GB-s
총 실행 수	백만번 실행당 224.930원	1백만번 실행

Consumption plan에서의 기본 제공 범위

- 매월 최소 40만 GB/s와 100만번 실행까지 무료



https://azure.microsoft.com/ko-kr/pricing/calculator/

Consumption Plan 가격 계산 예 - 리소스 사용량

- 메모리 사용량이 0.5GB 인 함수가 1달에 300만번 실행되고, 매 1회 실행에 1초 소요.
- 리소스 사용량 (초) : 300만 실행 * 1초 ➡ 300만 초
- 리소스 사용량 (GB-s) : 0.5 GB * 300만 초 ➡ 150만 GB-s
- 청구 가능한 리소스 사용량 = 150만 - 40만(무료제공) = 110만 GB-s
- 월간 리소스 사용량 비용 : 110만 GB-s * 0.017995원 = 약 19,794원

Consumption Plan 가격 계산 예 - 실행횟수 청구 계산

- 메모리 사용량이 0.5GB 인 함수가 1달에 300만번 실행되고, 매 1회 실행에 1초 소요.
- 청구 가능한 실행 수 : 300만번 - 100만번(무료제공) = 200만번
- 월간 실행 비용 : 200만 / 100만 * 224.930원(백만당요금) = 449.86원
- 합계 => 월간 리소스 사용량 비용 + 월간 실행 비용 : 약 20,243원

이 외에, Storage account, Application Insights, Network Traffic 등의 비용이 발생합니다.

데이터센터 위치별, 시기별로 가격이 상이할 수 있습니다.
Azure 계산기 서비스로 꼭 계산해보세요. ;-)

소비 계획 (Consumption Plan) 제약사항 (1/2)

- Trigger되는 이벤트 수에 따라, 호스트의 인스턴스를 최대 200개까지 추가
  - 각 인스턴스는 1개의 CPU와 1.5GB의 메모리
  - Http Trigger의 경우 새 인스턴스는 초당 1번만 할당되며, 그 외 Trigger는 30초당 한 번씩.
  - properties.functionAppScaleLimit 설정을 통해, 최대 인스턴스 개수 지정 가능.
- 수행 제한 시간 : 디폴트 5분, 최대 10분. 그리고 Cold start 있음.
- 최대 요청 크기 : 100MB
- Linux OS를 지원하고, Windows OS는 미지원.
  - 코드 실행만 지원할 뿐, Linux Docker Container는 미지원.

[공식문서] Azure Functions 크기 조정 및 호스팅

소비 계획 (Consumption Plan) 제약사항 (2/2)

- Outbound IP 제한 불가. 가상 네트워크 (VNet) 불가.
  - Database 방화벽에 허용 IP Rule을 지정하기가 어렵습니다.
  - 비공개 네트워크에서만 서비스에 접근토록 할 수 있습니다. (Azure Virtual Network)
- Inbound IP 제한은 가능.

다음 에피소드에서는 ...

EP02. 애저펑션 개발환경 구축 및 기본 프로젝트 생성

- 파이썬/nodejs 및 Visual Studio Code 설치 안내
- Azure Functions Core Tools 설치
- 애저펑션 프로젝트 생성 및 구동
- Visual Studio Code를 활용한 디버깅

Life is short.
You need Python and Django.

I will be your pacemaker.