GitHub Actions 실습 - 배포

주신영 bit1010@live.com

앞서 생성한 도커 컨테이너를 GitHub Action을 이용해서 Webpp에 배포하는 실습을 진행하 겠습니다. 이번 실습에서는 Webapp에 배포하지만 응용하면 Azure의 모든 컨테이너 관련 서비스(AKS,ACI 등)에 배포 가능합니다.

앞서 사용한 도커 관련 https://github.com/mr-smithers-excellent/docker-build-push 액션을 사용하였는데 Azure의 Container Registry(ACR)를 지원하지 않으므로 ACR과 도커허브를 지원하는 https://github.com/azure/docker-login를 사용할 수도 있습니다.

참고: GitHub Actions를 사용하여 App Service에 사용자 지정 컨테이너 배포

https://learn.microsoft.com/ko-kr/azure/app-service/deploy-container-github-action? tabs=publish-profile

Azure AppService 생성

리소스 만들기 → 웹앱

기본 탭

웹앱 만들기 ...

기본 Docker 네트워킹 모니터링 태그 검토 + 만들기

App Service Web Apps를 사용하면 모든 플랫폼에서 실행되는 엔터프라이즈급 웹, 모바일 및 API 앱을 신속하게 구축, 배포 및 확장할 수 있습니다. 엄격한 성능, 확장, 보안 및 컴플라이언스 요구 사항을 충족하는 동시에 완전 관리형 플랫폼을 사용하여 인프라 유지 관리를 수행하세요. 자세히 ♂

프로젝트 세부 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 폴더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

구독* ①	Microsoft Azure 스픈서쉽	~
리소스 그룹 * ①	Web Ann Cittle Action less group	
디조스 그룹 * ①	WebAppGitHubActionJsy_group 새로 만들기	
인스턴스 정보		
데이터베이스가 필요한가요? 새 웹 +	데이터베이스 환경을 사용해 보세요. ♂	
이름 *	WebAppGitHubActionJsy	~
		.azurewebsites.net
게시 *	○ 코드 ● Docker 컨테이너 ○ 정적 웹 앱	
운영 체제 *	Linux	

리소스 그룹 생성(또는 기존 그룹 선택)

이름: 웹앱의 고유 주소로 사용

게시: Docker 컨테이너

운영 체제: Linux

검토 + 만들기

만들기

WebApp에서 설정 추가

기본 도메인 링크로 접속

기본 페이지 확인

웹앱 개요 페이지 상단에 '게시 프로필 다운로드' 클릭 후 다운 받은 파일의 내용을 메모장에 복사(아래 코드에서 사용)



구성의 애플리케이션 설정에서

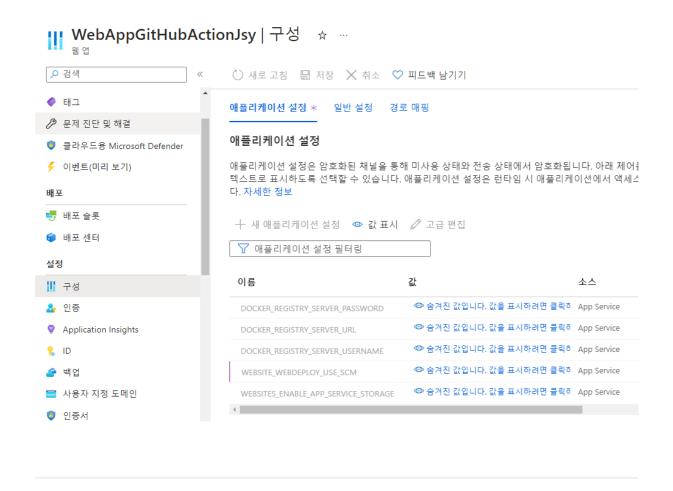
새 애플리케이션 설정 추가

이름: WEBSITE WEBDEPLOY USE SCM

값 : true



리눅스 웹앱에서는 위 설정을 추가해야 정상적으로 파일을 다운로드 할 수 있습니다.



GitHub코드에서 ci.yml 파일 수정

• 빌드할때 마다 고유한 태그값으로 도커 이미지를 계속 생성하기 위해 깃의 해시값(github.sha)을 사용하도록 코드 수정

tags: \${{ github.sha }

• azure/webapps-deploy Action 사용하여 WebApp에 컨테이너 배포

https://github.com/Azure/webapps-deploy

파일 아래에 코드 추가

- app-name에 생성한 웹앱 이름
- images에 도커허브에 추가한 저장소 추가 이미지이름은 앞에서 생성한 github-actions-app

 publish-profile 웹앱에 배포하기 위한 게시 프로필을 Secrets 설정에 추가 (위에서 메모장에 복사해둔 파일 내용)
 github - 레포지토리의 settings - Secrets and variables - New - Action New repository secret
 Name : AZURE_WEBAPP_PUBLISH_PROFILE Secret : 웹앱 게시 프로필

```
- name: 'Deploy to Azure Web App'
uses: azure/webapps-deploy@v2
with:
   app-name: '웹앱 이름'
   publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE
   images: '도커허브 계정/이미지이름:${{ github.sha }}'
```

ci.yml 전체 코드

```
name: Python application
on:
  push:
    branches: [ python-ci-workflow ]
  pull_request:
    branches: [ python-ci-workflow ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    uses: actions/checkout@v2
    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: "3.8"
    - name: Display Python version
```

```
run: python -c "import sys; print(sys.version)"
- name: Build & push Docker image
uses: mr-smithers-excellent/docker-build-push@v5
with:
    image: 도커허브 계정/github-actions-app
    tags: ${{ github.sha }}
    registry: docker.io
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}

- name: 'Deploy to Azure Web App'
uses: azure/webapps-deploy@v2
with:
    app-name: '웹앱 이름'
    publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROimages: '도커허브 계정/github-actions-app:${{ github.sha}}
```

수정내용 커밋

Start Commit

Commit changes

GitHub코드에서 Dockerfile 파일 수정

웹앱에서 80포트를 기본으로 사용하고 있으므로 컨테이너에서 80포트 사용하도록 설정

• 도커의 외부 노출 포트 추가

EXPOSE 80

• flask 실행시 80포트 사용

```
CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0", "--
```

Dockerfile 전체 코드

```
# syntax=docker/dockerfile:1
FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

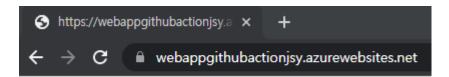
EXPOSE 80

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0", "---
```

수정내용 커밋

Commit changes

웹브라우저로 다시 접속 (배포되는데 시간이 걸릴 수 있습니다. 크롬의 시크릿 모드로 접속해보세요.)



Hello, GitHub Action!