

AZ-104. Challenge Lab 09

LAB 02. 서버리스 컨테이너 기반 환경 프로비저닝

이 문서는 Microsoft Technical Trainer팀에서 ESI 교육 참석자분들에게 제공해 드리는 문서입니다.

요약

이 내용들은 표시된 날짜에 Microsoft에서 검토된 내용을 바탕으로 하고 있습니다. 따라서, 표기된 날짜 이후에 시장의 요구사항에 따라 달라질 수 있습니다. 이 문서는 고객에 대한 표기된 날짜 이후에 변화가 없다는 것을 보증하지 않습니다.

이 문서는 정보 제공을 목적으로 하며 어떠한 보증을 하지는 않습니다.

저작권에 관련된 법률을 준수하는 것은 고객의 역할이며, 이 문서를 마이크로소프트의 사전 동의 없이 어떤 형태(전자 문서, 물리적인 형태 막론하고) 어떠한 목적으로 재 생산, 저장 및 다시 전달하는 것은 허용되지 않습니다.

마이크로소프트는 이 문서에 들어있는 특허권, 상표, 저작권, 지적 재산권을 가집니다. 문서를 통해 명시적으로 허가된 경우가 아니면, 어떠한 경우에도 특허권, 상표, 저작권 및 지적 재산권은 다른 사용자에게 허용되지 않습니다.

© 2023 Microsoft Corporation All right reserved.

Microsoft®는 미합중국 및 여러 나라에 등록된 상표입니다.

이 문서에 기재된 실제 회사 이름 및 제품 이름은 각 소유자의 상표일 수 있습니다.

문서 작성 연혁

날짜	버전	작성자	변경 내용
2023.08.30	1.0.0	우진환	LAB 02 내용 작성

목차

도전 과제	5
STEP 01. AZURE CONTAINER REGISTRY 프로비저닝	5
STEP 02. AZURE CONTAINER INSTANCE 프로비저닝	5
STEP 03. 컨테이너 기반 웹 앱 프로비저닝	6
TASK 01. AZURE CONTAINER REGISTRY 프로비저닝	7
TASK 02. AZURE CONTAINER INSTANCE 프로비저닝	10
TASK 03. 컨테이너 기반 웹 앱 프로비저닝	12

도전 과제

이 실습에서는 컨테이너 기반 환경을 프로비저닝합니다. 이 환경에는 컨테이너 레지스트리, 컨테이너 인스턴스 및 컨테이너 기반 웹 앱이 포함됩니다.

STEP 01. Azure Container Registry 프로비저닝

1. 다음 속성을 사용하여 컨테이너 레지스트리를 만듭니다.

속성	값
리소스 그룹	Archlod<xxxxxxxx>
레지스트리 이름	acr<xxxxxxxx>
위치	East US
SKU	기본

2. 새로 만든 컨테이너 레지스트리에 "관리 사용자"를 활성화합니다.
3. 다음 속성을 사용하여 [Cloud Shell]을 만듭니다.

속성	값
Cloud Shell 지역	미국 동부
리소스 그룹	Archlod<xxxxxxxx>
스토리지 계정	기존 계정 선택
파일 공유	cloud-shell

4. 자신의 컴퓨터에서 [터미널]을 열고 `dockerHost` 가상 머신에 SSH 세션으로 연결합니다.
5. `docker login` 명령을 사용하여 앞서 만든 컨테이너 레지스트리에 관리 사용자로 로그인합니다.
6. `docker pull` 명령을 사용하여 다음 Docker Hub 이미지를 풀링합니다.

Docker 이미지
notlods/exampleservice:1.0
notlods/examplewebapp:2.0

7. `docker tag` 명령을 사용하여 풀링한 이미지를 ACR의 이름으로 태그를 설정합니다.
8. `docker push` 명령을 사용하여 태그를 지정한 이미지를 ACR에 푸시합니다.

STEP 02. Azure Container Instance 프로비저닝

1. 다음 속성을 사용하여 컨테이너 인스턴스를 만듭니다.

속성	값
리소스 그룹	Archlod<xxxxxxxx>
컨테이너 이름	ci<xxxxxxxx>
이미지 원본	Azure Container Registry
레지스트리	acr<xxxxxxxx>
이미지	exampleservice
이미지 태그	1.0
크기	1 vcpu, 1.5GiB 메모리, 0 gpu
포트	8080

2. 새로 만든 컨테이너 인스턴스의 공용 IP를 메모장에 기록합니다.
3. 다음 명령을 실행하여 Python 테스트 스크립트를 다운로드하고 권한을 설정합니다.

```
curl https://lodschallenge.blob.core.windows.net/storagechallenges/Client.py
> ./Client.py

chmod +x ./Client.py
```

4. 다음 명령을 실행하여 컨테이너 서비스에 부하를 시뮬레이션합니다.

```
./Client.py -a 52.152.240.181 -p 8080 -c 2 -m 5
```

STEP 03. 컨테이너 기반 웹 앱 프로비저닝

1. [Cloud Shell]에서 다음 매개 변수와 함께 `az appservice plan create` 명령을 실행하여 App Service Plan을 만듭니다.

매개 변수	값
-g	Archlod<xxxxxxxx>
-n	cwaASP
--is-linux	
--sku	S1

2. [Cloud Shell]에서 다음 매개 변수와 함께 `az webapp create` 명령을 실행하여 웹 앱을 만듭니다.

매개 변수	값
-g	Archlod<xxxxxxxx>
-p	cwaASP
-n	cwa<xxxxxxxx>
-i	acr<xxxxxxxx>.azurecr.io/examplewebapp:2.0

3. 새로 만든 웹 앱에 다음 속성의 애플리케이션 설정을 추가합니다.

이름	값
Address	첫 번째로 배포했던 컨테이너 인스턴스의 공용 IP

4. 웹 앱에 다음과 같은 자동 크기 조정을 구성합니다.

설정	값
스케일 아웃	(평균) CPUPercentge > 70
스케일 인	(평균) CPUPercentge < 30
최대 인스턴스	5

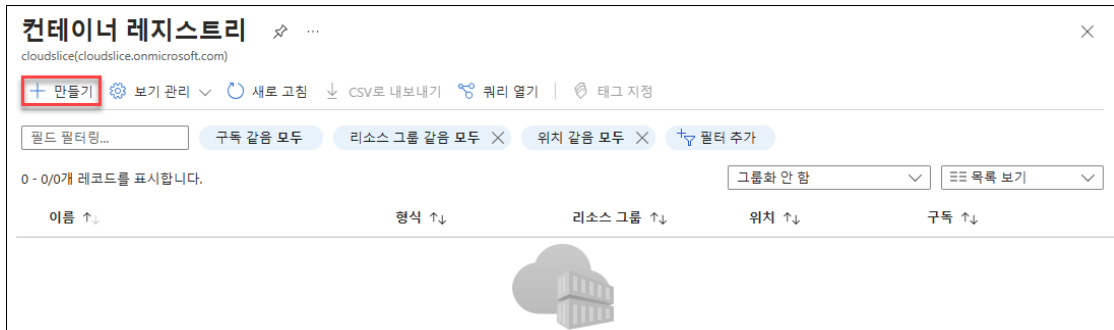
5. 웹 앱의 App Service 로그가 파일 시스템을 사용하도록 구성하고 보존 기간을 365일로 설정합니다.

6. 웹 앱을 열고 다음 속성으로 [Test API]를 클릭합니다. 성공 메시지가 반환되는 것을 확인합니다.

Address	Port	Command
빈 값을 유지	8080	Ping

TASK 01. Azure Container Registry 프로비저닝

1. Azure 포털의 검색창에서 "컨테이너 레지스트리"를 검색한 후 클릭합니다. [컨테이너 레지스트리] 블레이드에서 [만들기]를 클릭합니다.



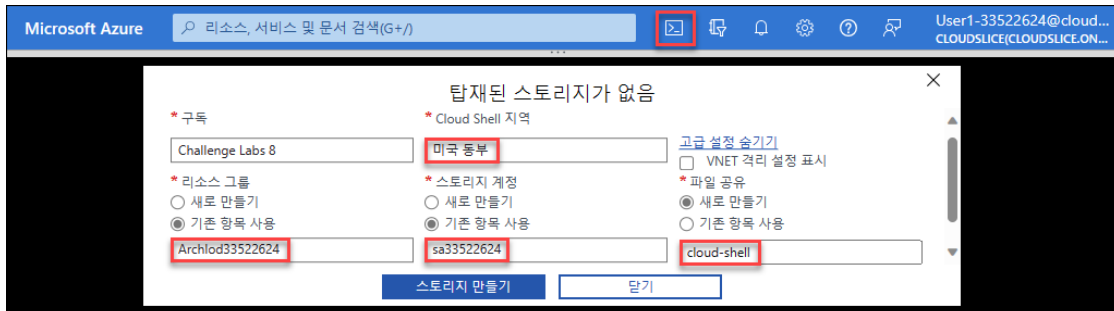
2. [컨테이너 레지스트리 만들기] 블레이드의 [기본 사항] 탭에서 아래와 같이 구성한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.

- [프로젝트 정보 - 리소스 그룹]: Archlod<xxxxxxxx>
- [인스턴스 정보 - 레지스트리 이름]: acr<xxxxxxxx>
- [인스턴스 정보 - 위치]: East US
- [인스턴스 정보 - SKU]: 기본

3. 새로 만든 [acr<xxxxxxxx> 컨테이너 레지스트리] 블레이드의 [설정 - 액세스 키]로 이동합니다. "관리 사용자" 옵션을 [사용]으로 설정한 후 "사용자 이름"과 "password" 값을 메모장에 복사합니다.

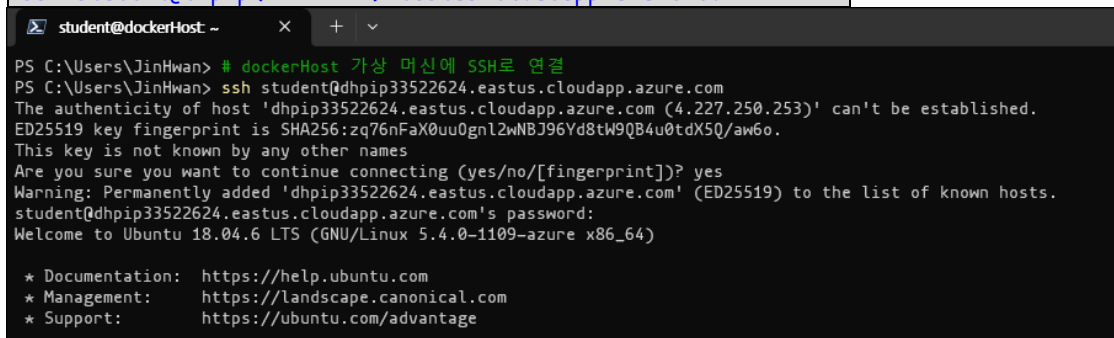
4. Azure 포털에서 [Cloud Shell] 아이콘을 클릭하고 "Bash"를 클릭합니다. [탑재된 스토리지가 없음] 창에서 "고급 설정 표시" 링크를 클릭합니다. [탑재된 스토리지가 없음] 페이지에서 아래와 같이 구성한 후 [스토리지 만들기]를 클릭합니다.

- Cloud Shell 지역: 미국 동부
- 리소스 그룹: "기존 항목 사용"을 선택하고 Archlod<xxxxxxxx> 리소스 그룹을 선택합니다.
- 스토리지 계정: "기존 항목 사용"을 선택하고 sa<xxxxxxxx> 스토리지 계정을 선택합니다.
- 파일 공유: "새로 만들기"를 선택하고 "cloud-shell" 이름을 입력합니다.



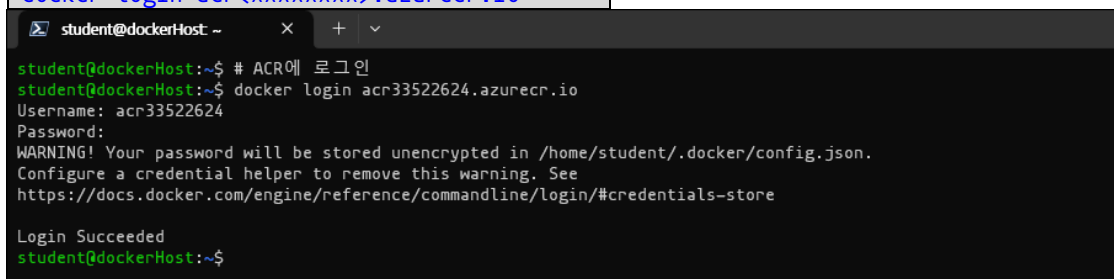
5. 자신의 컴퓨터에서 [터미널]을 열고 이미 배포되어 있는 dockerHost 가상 머신에 SSH 세션으로 연결합니다. 사용자 이름(student)과 암호(Azure!Bd1Dh0!#Be)를 사용합니다.

```
# dockerHost 가상 머신에 SSH로 연결
ssh student@dhpip<xxxxxxxx>.eastus.cloudapp.azure.com
```



6. [터미널]에서 다음 명령을 실행하여 Azure Container Registry에 로그인합니다. 앞서 ACR의 액세스 키에서 확인했던 "사용자 이름"과 "password"를 사용합니다.

```
# ACR에 로그인
docker login acr<xxxxxxxx>.azurecr.io
```



7. [터미널]에서 다음 명령을 실행하여 Docker Hub에서 두 개의 컨테이너 이미지를 풀링합니다.

```
# Docker Hub에서 이미지 풀링
docker pull notlods/exampleservice:1.0
docker pull notlods/examplewebapp:2.0
```

```

student@dockerHost: ~
student@dockerHost:~$ # Docker Hub에서 이미지 풀링
student@dockerHost:~$ docker pull notlods/exampleservice:1.0
1.0: Pulling from notlods/exampleservice
72c01b436656: Pull complete
65584f5f70ee: Pull complete
dc9874b52952: Pull complete
86656bbaa6fd: Pull complete
7fe6916ab382: Pull complete
572b3c520389: Pull complete
30a8297e20ea: Pull complete
d07ba5d228ac: Pull complete
8880a0d4f3dd: Pull complete
cc2c572ee38a: Pull complete
Digest: sha256:0e0d1ce5c7aa0c9c1766f9793c54b84d589bcd44782e062c5d71c83c9028b4a
Status: Downloaded newer image for notlods/exampleservice:1.0
docker.io/notlods/exampleservice:1.0
student@dockerHost:~$ docker pull notlods/examplewebapp:2.0
2.0: Pulling from notlods/examplewebapp
be8881be8156: Pull complete
5dfc1ce89b0d: Pull complete
a54a8a3ea138: Pull complete
c80594b69ec4: Pull complete
160fe1848092: Pull complete
2d3a006402f1: Pull complete
Digest: sha256:fed53843d98315331201bbbfcccb52e259192cce3a1b13881ef2ceeee19000bb
Status: Downloaded newer image for notlods/examplewebapp:2.0
docker.io/notlods/examplewebapp:2.0
student@dockerHost:~$

```

8. [터미널]에서 다음 명령을 실행하여 풀링한 컨테이너 이미지를 ACR의 리포지토리 이름과 일치하도록 태그를 지정합니다. 태그 지정 후 컨테이너 이미지를 확인합니다.

```

# 풀링한 컨테이너 이미지에 태그 지정
docker tag notlods/exampleservice:1.0
acr<xxxxxxxx>.azurecr.io/exampleservice:1.0

docker tag notlods/examplewebapp:2.0 acr<xxxxxxxx>.azurecr.io/examplewebapp:2.0

docker images

```

```

student@dockerHost: ~
student@dockerHost:~$ # 풀링한 컨테이너 이미지에 태그 지정
student@dockerHost:~$ docker tag notlods/exampleservice:1.0 acr33522624.azurecr.io/exampleservice:1.0
student@dockerHost:~$ docker tag notlods/examplewebapp:2.0 acr33522624.azurecr.io/examplewebapp:2.0
student@dockerHost:~$ docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
acr33522624.azurecr.io/examplewebapp      2.0      54175c99b496  4 years ago   258MB
notlods/examplewebapp                     2.0      54175c99b496  4 years ago   258MB
acr33522624.azurecr.io/exampleservice     1.0      6ba13d6aca67  4 years ago   235MB
notlods/exampleservice                     1.0      6ba13d6aca67  4 years ago   235MB
student@dockerHost:~$

```

9. [터미널]에서 다음 명령을 실행하여 로컬의 컨테이너 이미지를 ACR에 푸시합니다.

```

# ACR 에 이미지 푸시
docker push acr<xxxxxxxx>.azurecr.io/exampleservice:1.0
docker push acr<xxxxxxxx>.azurecr.io/examplewebapp:2.0

```

```

student@dockerHost: ~
student@dockerHost:~$ # ACR에 이미지 푸시
student@dockerHost:~$ docker push acr33522624.azurecr.io/exampleservice:1.0
The push refers to repository [acr33522624.azurecr.io/exampleservice]
76c403c21dc8: Pushed
92b25e94dd97: Pushed
90d741056c2b: Pushed
cf820b1b149d: Pushed
1d6a117905af: Pushed
1ff19b5310ed: Pushed
616fe7ac49b1: Pushed
52221a15d8f8: Pushed
6d734414ae85: Pushed
aa4e7d478f39: Pushed
1.0: digest: sha256:0e0d1ce5c7aa0c9c1766f9793c54b84d589bcd44782e062c5d71c83c9028b4a size: 2819
student@dockerHost:~$ docker push acr33522624.azurecr.io/examplewebapp:2.0
The push refers to repository [acr33522624.azurecr.io/examplewebapp]
c64c4433fa34: Pushed
4d577f75acc9: Pushed
393dd8f4a879: Pushed
0ad9ffac9ae9: Pushed
8ea427f58308: Pushed
cdb3f9544e4c: Pushed
2.0: digest: sha256:fed53843d98315331201bbbfcccb52e259192cce3a1b13881ef2ceae19000bb size: 1580
student@dockerHost:~$

```

TASK 02. Azure Container Instance 프로비저닝

1. Azure 포털의 검색창에서 "Container Instances"를 검색한 후 클릭합니다. [Container Instances] 블레이드에서 [만들기]를 클릭합니다.



2. [컨테이너 인스턴스 만들기] 블레이드의 [기본 사항] 탭에서 아래와 같이 구성한 후 [다음]을 클릭합니다.
 - [프로젝트 정보 - 리소스 그룹]: Archlod<xxxxxxxx>
 - [컨테이너 세부 정보 - 컨테이너 이름]: ci<xxxxxxxx>
 - [컨테이너 세부 정보 - 지역]: (US) East US
 - [컨테이너 세부 정보 - 가용성 영역]: None
 - [컨테이너 세부 정보 - SKU]: 표준
 - [컨테이너 세부 정보 - 이미지 원본]: Azure Container Registry
 - [컨테이너 세부 정보 - 레지스트리]: acr<xxxxxxxx>
 - [컨테이너 세부 정보 - 이미지]: exampleservice
 - [컨테이너 세부 정보 - 이미지 태그]: 1.0
 - [컨테이너 세부 정보 - 크기]: 1 vcpu, 1.5GiB 메모리, 0 gpu

컨테이너 인스턴스 만들기 ...

기본 사항 네트워크 고급 태그 검토 + 만들기

ACI(Azure Container Instances)를 통해 서버를 관리하거나 새 도구 사용 방법을 배울 필요 없이 Azure에서 컨테이너를 빠르게 쉽게 실행할 수 있습니다. ACI는 초당 요금 정가를 제공하여 클라우드에서 컨테이너 실행 비용을 최소화합니다.
[Azure Container Instances에 대한 자세한 정보](#)

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

구독 * ① Challenge Labs 8
 리소스 그룹 * ① Archlod33522624
[새로 만들기](#)

컨테이너 세부 정보

컨테이너 이름 * ① ci33522624 ✓
 지역 * ① (US) East US ✓
 가용성 영역(미리 보기) ① None ✓
 SKU 표준 ✓
 이미지 원본 * ① ☒ 빠른 시작 이미지 ☐ Azure Container Registry ☐ 기타 레지스트리
 Azure Spot 할인으로 실행 ① ☐
 레지스트리 * ① acr33522624 ✓
 이미지 * ① exampleservice ✓
 이미지 태그 * ① 1.0 ✓
 OS 유형 Linux
 크기 * ① 1 vcpu, 1.5GiB 메모리, 0 gpu
[크기 변경](#)

3. [네트워크] 탭에서 다른 설정은 기본값을 유지하고 포트는 "8080"을 입력한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.

컨테이너 인스턴스 만들기 ...

기본 사항 **네트워크** 고급 태그 검토 + 만들기

컨테이너 인스턴스의 네트워킹 옵션 세 개 중에서 선택합니다.

- '퍼블릭'을 선택하면 컨테이너 인스턴스의 공용 IP 주소가 만들어집니다.
- '프라이빗'을 사용하면 컨테이너 인스턴스의 새 가상 네트워크나 기존 가상 네트워크를 선택할 수 있습니다. Windows 컨테이너에는 아직 사용할 수 없습니다.
- '없음'을 선택하면 공용 IP나 가상 네트워크가 만들어지지 않습니다. 명령줄을 사용하여 컨테이너 로그에 계속 액세스할 수 있습니다.

네트워킹 유형 ☒ 퍼블릭 ☐ 프라이빗 ☐ 없음

DNS 이름 레이블 ①

DNS 이름 레이블 범위 재사용 ① 모든 다시 사용(안전하지 않음) ✓

포트 ①

포트 포트 프로토콜

8080 ✓ TCP ✓

4. 새로 만든 [ci<xxxxxxxx>] 컨테이너 인스턴스 블레이드의 [개요]로 이동한 후 "IP 주소(Public)"의 값을 메모장에 기록합니다.

ci33522624 ...

Container Instances

검색 << ▶ 시작 다시 시작 중지 삭제 새로 고침

개요

활동 로그
 액세스 제어(IAM)
 태그
 설정
 컨테이너

기본 정보

리소스 그룹 (이동) Archlod33522624
 상태 실행 중
 위치 East US
 구독 (이동) Challenge Labs 8

SKU Standard
 OS 유형 Linux
 IP 주소(Public) 20.241.190.173
 FQDN ---

[JSON 보기](#)

5. [Cloud Shell]의 Bash 세션에서 다음 명령을 실행하여 Python 테스트 스크립트를 [Cloud Shell]에 다운로드합니다.

```
# Python 테스트 스크립트 다운로드
curl https://lodschallenge.blob.core.windows.net/storagechallenges/Client.py
> ./Client.py
```

Bash

```
user1-33522624 [ ~ ]$ # Python 테스트 스크립트 다운로드
user1-33522624 [ ~ ]$ curl https://lodschallenge.blob.core.windows.net/storagechallenges/Client.py > ./Client.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 1903  100 1903    0     0  46345      0  --:--:-- --:--:-- --:--:-- 46414
user1-33522624 [ ~ ]$
```

6. [Cloud Shell]의 Bash 세션에서 다음 명령을 실행하여 다운로드한 Python 스크립트에 권한을 설정합니다.

```
# Python 스크립트 권한 설정
chmod +x ./Client.py
```

Bash

```
user1-33522624 [ ~ ]$ # Python 스크립트 권한 설정
user1-33522624 [ ~ ]$ chmod +x ./Client.py
user1-33522624 [ ~ ]$
```

7. [Cloud Shell]의 Bash 세션에서 다음 명령을 실행하여 Python 스크립트로 부하 시뮬레이션을 실행합니다.

```
# 컨테이너 인스턴스에 부하 시뮬레이션
./Client.py -a 20.241.190.173 -p 8080 -c 2 -m 5
```

Bash

```
user1-33522624 [ ~ ]$ # 컨테이너 인스턴스에 부하 시뮬레이션
user1-33522624 [ ~ ]$ ./Client.py -a 20.241.190.173 -p 8080 -c 2 -m 5
Received 2
Processed load for 5 minutes from 2023-08-30 15:19:18.115403 to 2023-08-30 15:24:18.115403
IPv4 Addresses: 192.168.0.23;
Computer Name: SandboxHost-638290052894871394
user1-33522624 [ ~ ]$
```

TASK 03. 컨테이너 기반 웹 앱 프로비저닝

1. [Cloud Shell]의 Bash 세션에서 다음 명령을 실행하여 Linux를 기반으로 하는 새 App Service Plan을 만듭니다.

```
# App Service Plan 만들기
az appservice plan create -g Archlod<xxxxxxxx> -n cwaASP --is-linux --sku S1
```

Bash

```
user1-33522624 [ ~ ]$ # App Service Plan 만들기
user1-33522624 [ ~ ]$ az appservice plan create -g Archlod33522624 -n cwaASP --is-linux --sku S1
{
  "elasticScaleEnabled": false,
  "extendedLocation": null,
  "freeOfferExpirationTime": null,
  "geoRegion": "East US",
  "hostingEnvironmentProfile": null,
  "hyperV": false,
  "id": "/subscriptions/6474fbd5-eeef-432f-94ec-861410d81d1d/resourceGroups/Archlod33522624/providers/Microsoft.Web/serverfarms/cwaASP",
  "isSpot": false,
  "isXenon": false,
  "kind": "linux",
  "kubeEnvironmentProfile": null,
  "location": "eastus",
  "maximumElasticWorkerCount": 1,
  "maximumNumberOfWorkers": 0,
}
```

2. [Cloud Shell]의 Bash 세션에서 다음 명령을 실행하여 ACR에 푸시한 컨테이너 이미지를 사용하여 새 웹 앱을 만듭니다.

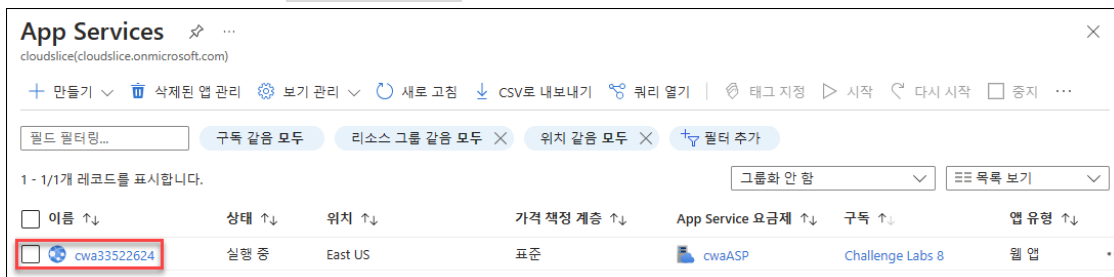
```
# 컨테이너 이미지를 사용하여 웹 앱 만들기
az webapp create -g Archlod<xxxxxxxx> -p cwaASP -n cwa<xxxxxxxx> \
-i acr<xxxxxxxx>.azurecr.io/examplewebapp:2.0
```

```

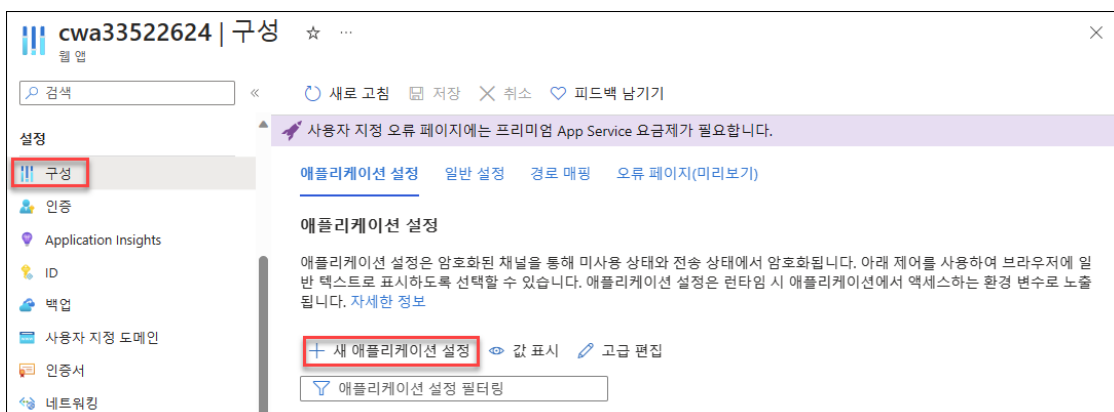
Bash
user1-33522624 [ ~ ]$ # 컨테이너 이미지를 사용하여 웹 앱 만들기
user1-33522624 [ ~ ]$ az webapp create --g Archlod33522624 --p cwaASP --n cwa33522624 \
-i acr33522624.azurecr.io/examplewebapp:2.0
No credential was provided to access Azure Container Registry. Trying to look up...
{
  "availabilityState": "Normal",
  "clientAffinityEnabled": true,
  "clientCertEnabled": false,
  "clientCertExclusionPaths": null,
  "clientCertMode": "Required",
  "cloningInfo": null,
  "containerSize": 0,
  "customDomainVerificationId": "84D590AB1F873984453E9A9F3588A7F4F4E6B2332FAB3F74EE0A4184F9C6C58E",
  "dailyMemoryTimeQuota": 0,

```

3. Azure 포털의 검색창에서 "App Services"를 검색한 후 클릭합니다. [App Services] 블레이드에서 Bash 세션에서 만들었던 `cwa<xxxxxxxx>` 웹 앱을 클릭합니다.

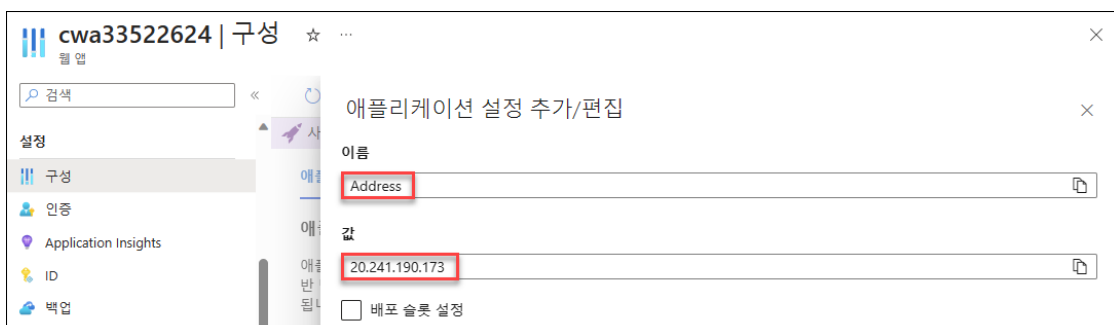


4. `[cwa<xxxxxxxx> 웹 앱]` 블레이드의 [설정 - 구성]으로 이동합니다. [애플리케이션 설정] 탭에서 [새 애플리케이션 설정]을 클릭합니다.



5. [애플리케이션 설정 추가/편집] 창에서 아래와 같이 입력한 후 [확인]을 클릭합니다.

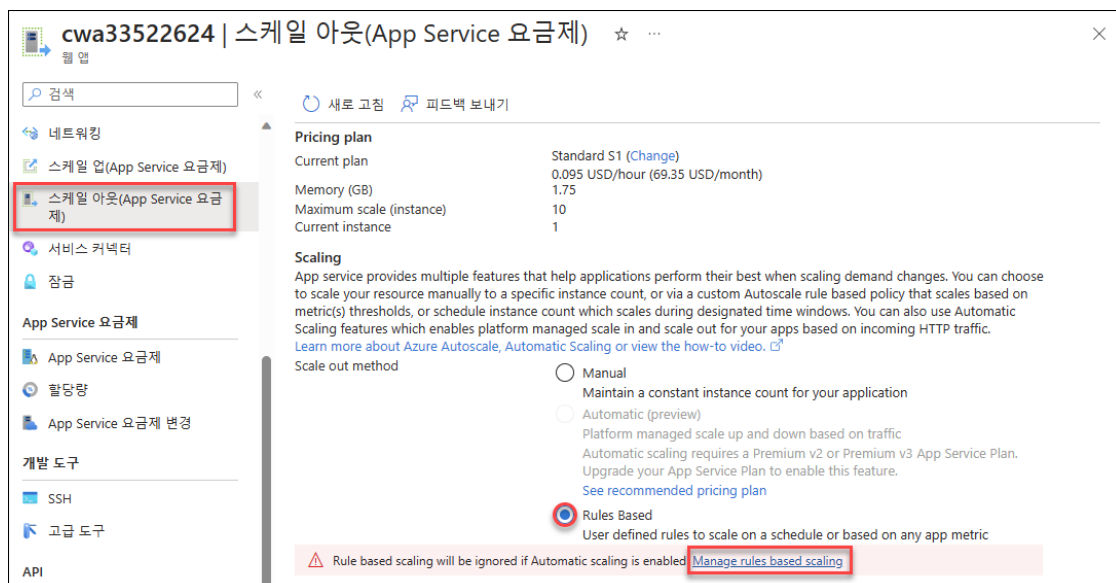
- 이름: Address
- 값: 복사했던 컨테이너 인스턴스의 공용 IP 주소를 입력합니다.



6. `[cwa<xxxxxxxx> | 구성]` 블레이드의 메뉴에서 [저장]을 클릭합니다. [변경 내용 저장] 창이 표시되면 [계속]을 클릭합니다.



7. [cwa<xxxxxxxx> 웹 앱] 블레이드의 [설정 - 스케일 아웃(App Service 요금제)]로 이동합니다. "Scale out method"를 "Rules Based"로 선택하고 "Manage rules based scaling" 링크를 클릭합니다.



8. [자동 크기 조정 설정] 블레이드의 [구성] 탭에서 아래와 같이 구성합니다.
- 리소스 크기를 조정하는 방법 선택: 사용자 지정 자동 크기 조정
 - 크기 조정 모드: 메트릭 기준 크기 조정
 - 인스턴스 제한: 최소값 1, 최대값 5, 기본값 1
 - "규칙 추가" 링크를 클릭합니다.

자동 크기 조정 설정 ...

cwaASP (App Service 요금제)

저장 취소 새로 고침 로그 피드백

구성 실행 기록 JSON 알리기 진단 설정

자동 스케일링은 수요가 변경될 때 애플리케이션이 최상의 성능을 유지할 수 있도록 하기 위해 기본 제공되는 기능입니다. 리소스 크기를 특정 인스턴스 수(으)로 수동으로 조정하거나, 메트릭 임계값에 따라 크기 조정되는 사용자 지정 자동 스케일링 정책을 통해 조정하거나, 지정된 기간 동안 스케일링되는 인스턴스 수를(를) 예약도록 선택할 수 있습니다. 자동 스케일링을 사용하면 요청에 따라 을(를) 추가하고 인스턴스 수를 제거하여 리소스의 성능을 향상하고 비용 효율적으로 유지할 수 있습니다. Azure 자동 스케일링에 대해 자세히 알아보거나 방법 동영상 확인하세요.

리소스 크기를 조정하는 방법 선택

수동 크기 조정
고정 인스턴스 수 유지

사용자 지정 자동 크기 조정 **1**
모든 메트릭을 기반으로 일정에 따라 크기 조정

사용자 지정 자동 크기 조정

자동 크기 조정 설정 이름 * cwaASP-자동 크기 조정-431

리소스 그룹 Archlod33522624

인스턴스 수 1

기본값 * 자동 생성된 기본 스케일링 조건

삭제 경고

크기 조정 모드

규칙

인스턴스 제한

일정

이 크기 조정 조건은 일치하는 다른 크기 조정 조건이 없을 때 실행됩니다.

+ 크기 조건 추가

9. [크기 조정 규칙] 창에서 아래와 같이 구성한 후 [추가]를 클릭합니다.

- 메트릭 이름: CPU Percentage
- 연산자: 보다 큼
- 크기 조정 작업을 트리거하는 메트릭 임계값: 70
- 작업: 다음을 기준으로 개수 늘이기
- 인스턴스 수: 1
- 다른 설정은 모두 기본값을 유지합니다.

크기 조정 규칙 ✕

메트릭 원본

현재 리소스(cwaASP)

리소스 종류 리소스

App Service 요금제 cwaASP

Criteria

메트릭 네임스페이스 *

표준 메트릭

메트릭 이름 CPU Percentage

자원 이름 연산자 크기 값 1분 시간 조정 추가

Instance = 모든 값 +

자원에 대해 여러 값을 선택하는 경우 자동 크기 조정이 각 값의 메트릭을 개별적으로 평가하지 않고, 선택한 값의 메트릭을 집계합니다.

CpuPercentage (평균)

10.83 %

인스턴스 개수로 메트릭 나누기 사용

연산자 * 크기 조정 작업을 트리거하는 메트릭 임계값 *

보다 큼 70 %

기간(분) * 시간 단위(분)

10 1

시간 조직 통계 * 시간 집계 *

평균 평균

Action

작업 * 휴지 기간(분) *

다음을 기준으로 개수 늘이기 5

인스턴스 수 *

1

10. 다시 "규칙 추가" 링크를 클릭합니다. [크기 조정 규칙] 창에서 아래와 같이 구성한 후 [추가]를 클릭합니다.

- 메트릭 이름: CPU Percentage

- 연산자: 보다 작음
- 크기 조정 작업을 트리거하는 메트릭 임계값: 30
- 작업: 다음을 기준으로 개수 줄이기
- 인스턴스 수: 1
- 다른 설정은 모두 기본값을 유지합니다.

크기 조정 규칙

메트릭 원본: 현재 리소스(cwaASP)

리소스 종류: App Service 요금제 | 리소스: cwaASP

☒ Criteria

메트릭 네임스페이스 *: 표준 메트릭 | 메트릭 이름: CPU Percentage

자원 이름: Instance | 연산자: = | 크기 값: 모든 값

작업: 다음을 기준으로 개수 줄이기

인스턴스 수: 1

시간 단위(분): 1 | 시간 집계: 평균

휴지 기간(분): 5

인스턴스 수: 1

11. [자동 크기 조정 설정] 블레이드의 메뉴에서 [저장]을 클릭합니다.

자동 크기 조정 설정

cwaASP (App Service 요금제)

저장 | 취소 | 새로 고침 | 로그 | 피드백

구성 | 실행 기록 | JSON | 알리기 | 진단 설정

자동 스케일링은 수요가 변경될 때 애플리케이션이 최상의 성능을 유지할 수 있도록 하기 위해 기본 제공되는 기능입니다. 리소스 크기를 특정 인스턴스 수(으)로 수동으로 조정하거나, 메트릭 임계값에 따라 크기 조정되는 사용자 지정 자동 스케일링 정책을 통해 조정하거나, 지정된 기간 동안 스케일링되는 인스턴스 수(들) 예약도록 선택할 수 있습니다. 자동 스케일링을 사용하면 요청에 따라 (들) 추가하고 인스턴스 수를 제거하여 리소스의 성능을 향상하고 비용 효율적으로 유지할 수 있습니다. Azure 자동 스케일링에 대해 자세히 알아보거나 방법 동영상을 확인하세요.

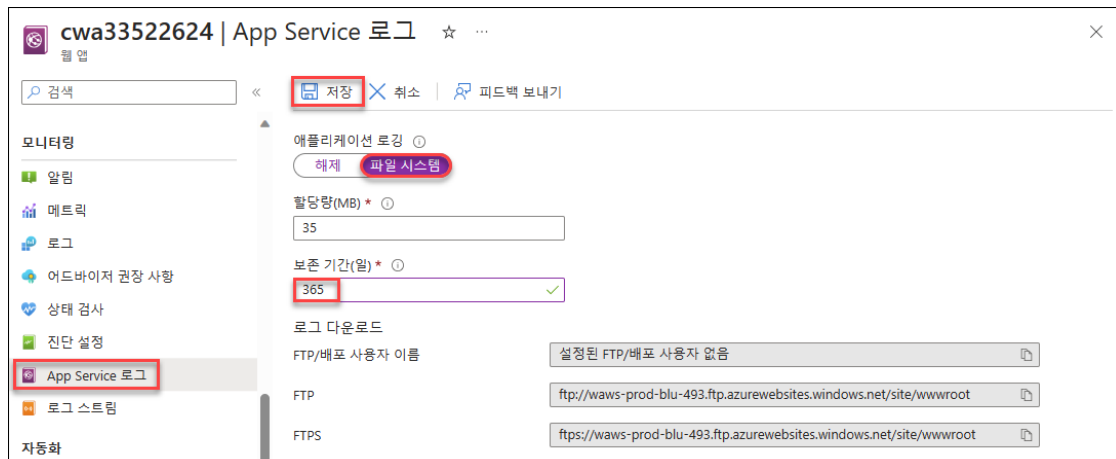
리소스 크기를 조정하는 방법 선택

☐ 수동 크기 조정
고정 인스턴스 수 유지

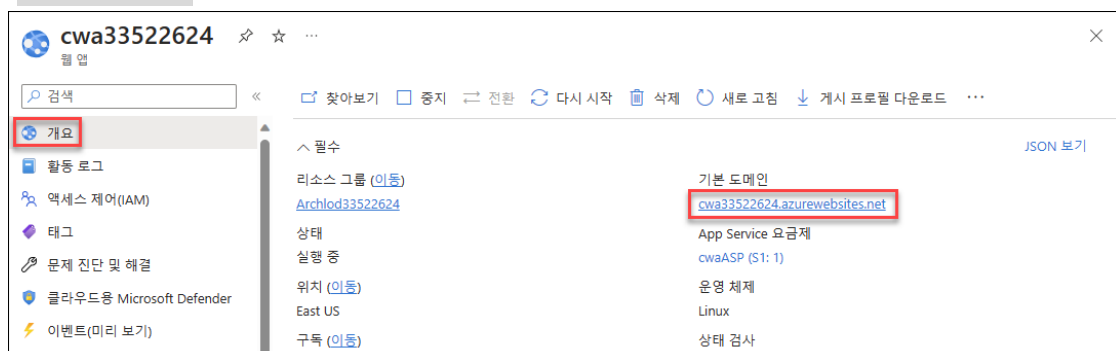
☒ 사용자 지정 자동 크기 조정
모든 메트릭을 기반으로 일정에 따라 크기 조정

12. [cwa<xxxxxxxx> 웹 앱] 블레이드의 [모니터링 - App Service 로그]로 이동합니다. 아래와 같이 구성한 후 [저장]을 클릭합니다.

- 애플리케이션 로깅: 파일 시스템
- 보존 기간(일): 365



13. [cwa<xxxxxxxx> 웹 앱] 블레이드의 [개요]로 이동한 후 기본 도메인 링크를 클릭합니다.



14. [Container-based Web App] 페이지에서 아래와 같이 구성한 후 [Test API]를 클릭합니다.

"Results"에 성공 메시지가 수신되었는지 확인합니다.

- Address: 아무런 값도 입력하지 않습니다.
- Port: 8080
- Command: Ping

