Ask Company

Docker 기반으로 스트레스없이 PaaS 인프라에 배포하기

Overview

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

배포 스트레스 … ☹

서버 인프라를 운영하기 위해서는 알아야할 것들이 참으로 많습니다 …

On Promise, 클라우드, IaaS, PaaS, Docker, 모니터링, Scale Up/Out

© All rights reserved by Ask Company

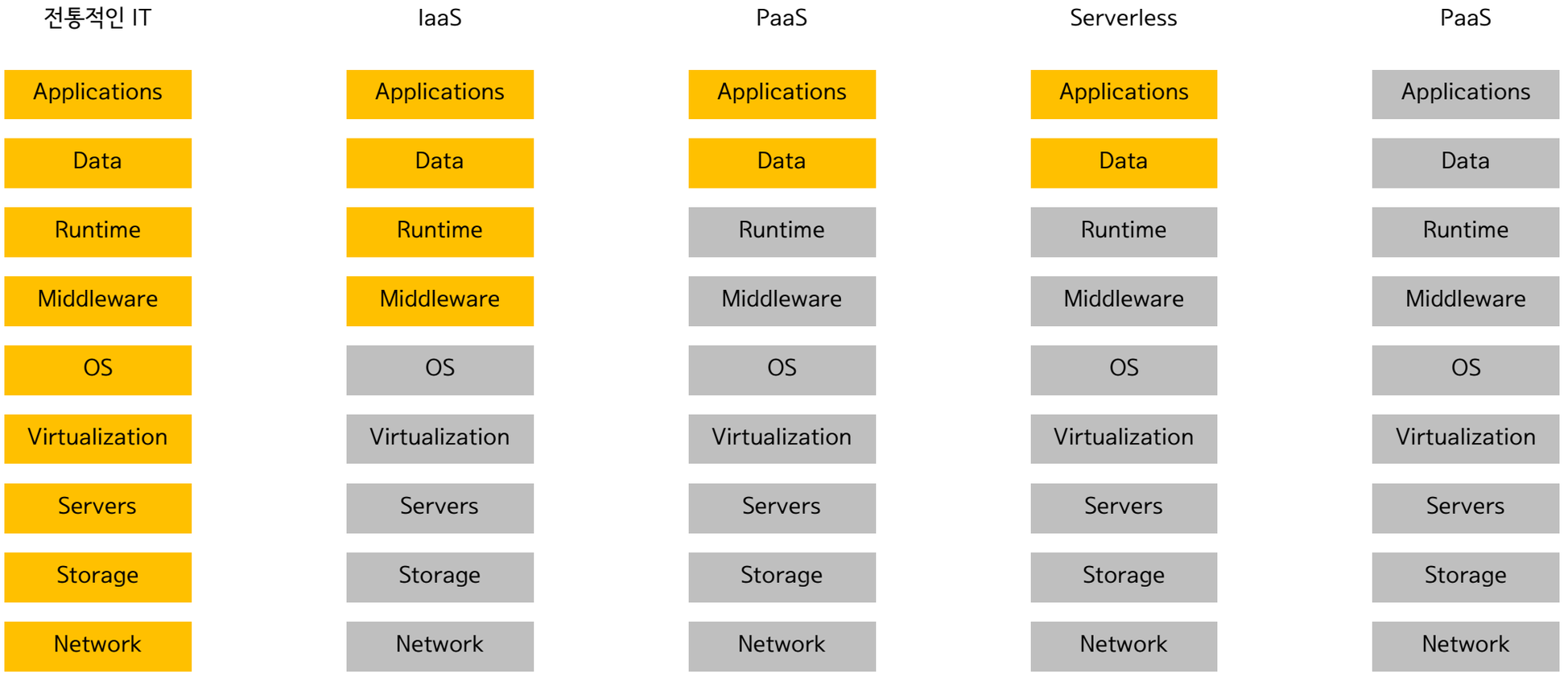
서버란?

서버는 클라이언트가 접속하기를 계속 기다리는 프로그램

1. 서버를 어디에 두느냐? 우리집? 사무실? IDC? 클라우드?
2. 서버 증설을 사람이 하느냐, 프로그램이 하느냐? 서버 1대, 100대, 1000대 …
3. 서버 하드웨어 장애가 나면? 소프트웨어 장애가 나면?

© All rights reserved by Ask Company

다양한 형태의 클라우드 서비스 유형



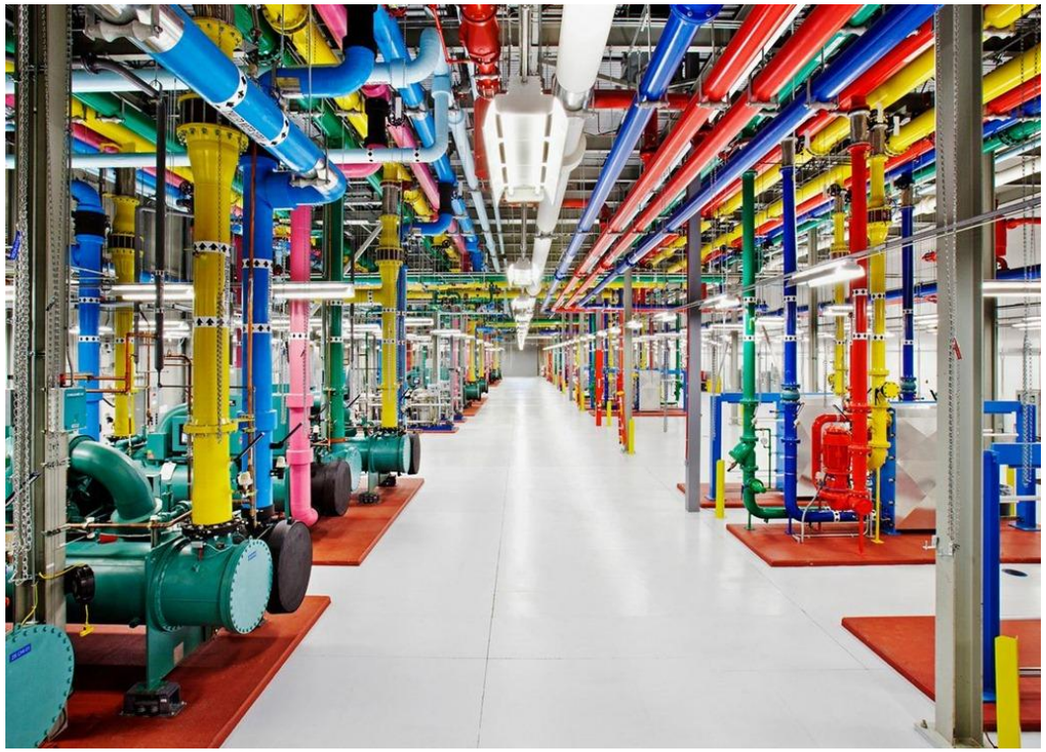
https://specify.io/concepts/serverless-baas-faas

© All rights reserved by Ask Company

클라우드란?

기본적으로는 자동화된 인프라 서비스.

!= 웹하드 서비스



© All rights reserved by Ask Company

클라우드 벤더에서 인프라/OS를 운영해준다면?

"클라우드 인프라/OS를 하나 하나 세팅해서 운영하고 싶은 마음"

➔ cf. 수동차를 몰고 다니고, 하나 하나 직접 정비해서 쓰기

선택과 집중. 본질에 집중하기. 서비스 !!!

우리의 시간은 소중합니다. 시간이 가장 큰 비용.

© All rights reserved by Ask Company

PaaS/Serverless가 좋은 것은 알겠는 데 …

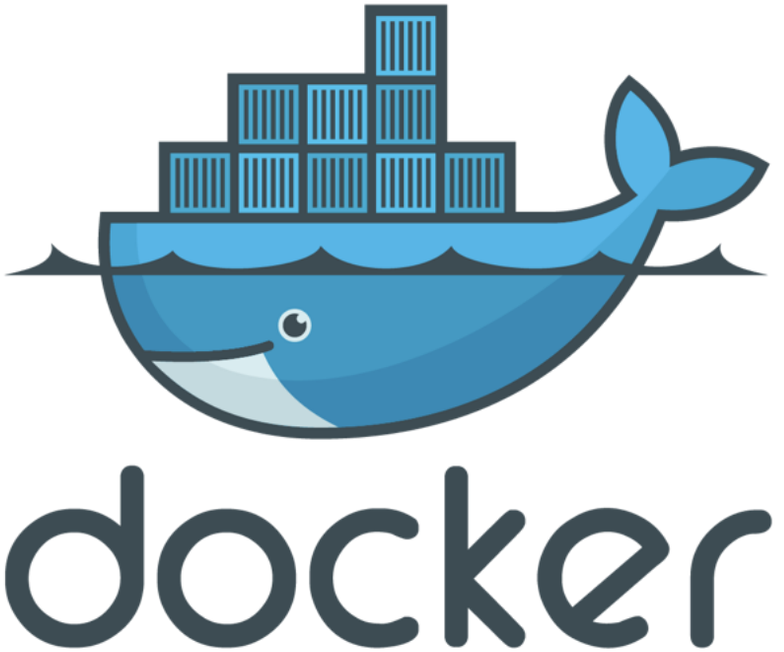
클라우드 벤더마다 기능도 다르고 적용법도 다르고 … ☹

PaaS 쓰면, 특정 클라우드 벤더에 Lock-in 되는 거 아냐???

예전에는 그랬지만, 컨테이너 기술의 등장으로 Don't mind.

© All rights reserved by Ask Company

Docker



소프트웨어 팩키징 방법

빠르고 가벼운 소프트웨어 팩키징 방법

애플리케이션과 그 실행환경/OS를 모두 포함한 소프트웨어 팩키지 => Docker Image

플랫폼에 상관없이 실행될 수 있는 애플리케이션 **컨테이너를 만드는 기술**

윈도우에서 exe 파일을 바로 실행하듯이,

Docker를 지원하는 플랫폼에서 Docker Image를 받아서 즉시 실행할 수 있습니다. (추가적인 OS/라이브러리 설치 No)

Docker Image는 Container의 형태로 Docker Engine이 있는 어디에서나 실행가능

대상: 로컬 머신 (윈도우/맥/리눅스), Azure, AWS, Digital Ocean 등

하나의 Docker Image를 통해 다수의 Container (인스턴스) 를 생성할 수 있습니다.

© All rights reserved by Ask Company

머신에 Docker만 설치되어있다면 !!!

어떤 추가적인 소프트웨어 설치 없이 ~

Docker Image만 당겨와서(pull), 바로 실행(run)

```
docker  run \
    --env MYSQL_ROOT_PASSWORD=my_root_password \
    --detach \
    --publish 3306:3306
    mysql:latest
```

© All rights reserved by Ask Company

Docker Container는 VM이 아닙니다.

Docker Container != Virtual Machine

일반적인 컴퓨터에서나 VM에서는 다수의 프로세스가 구동됩니다. 심지어 백그라운드/서비스 프로세스도 많죠.

도커 컨테이너는 하나의 Foreground 프로세스를 구동하는 것이 원칙.

VM과 비슷해보이지만, 혹은 비슷하게 쓸 수도 있지만, 그렇게 쓰질 않습니다.

생성된 Docker Container는 바로 쓰고 버리는 것

Immutable Infrastructure 패러다임을 충실히 지킬 수 있도록 도와줍니다.

비교) 예전 VM은 한 번 생성하면 애지중지. 관리 및 업데이트

Docker Container는 격리

해킹되더라도 Docker Engine이 구동되는 원래의 서버에는 영향을 끼치지 않습니다.

© All rights reserved by Ask Company

리눅스 기반의 기술

리눅스의 chroot 기반의 기술 → 리눅스가 꼭 필요.

참고) 윈도우 만의 컨테이너 기술도 별도로 있습니다.

윈도우에서는

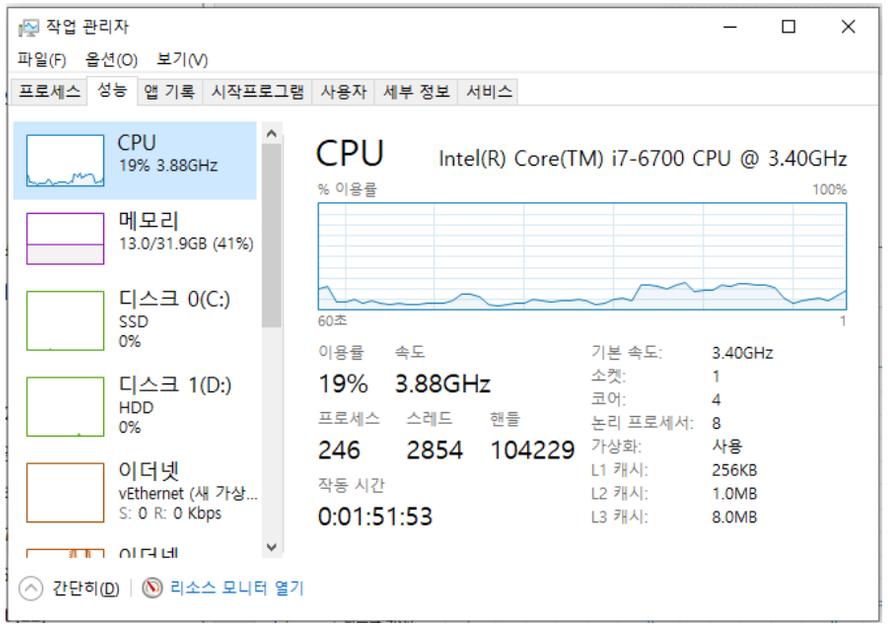
Hyper-V에서 리눅스를 구동하거나 → 메인보드에서 가상화 지원, 윈도우 10 Pro 이상에서 Hyper-V 지원.

Virtual Box를 통한 리눅스 구동이 필요. → 윈도우 10 Pro 미만

WSL v2에서는 윈도우에 리눅스가 통합 (2004버전부터)

맥도 과거에는

Virtual Box를 썼었더랬죠.



© All rights reserved by Ask Company

Docker 만의 특징/유의사항

Docker 내에서 어떤 프로세스가 도는 지 명확히 하기 위해서.

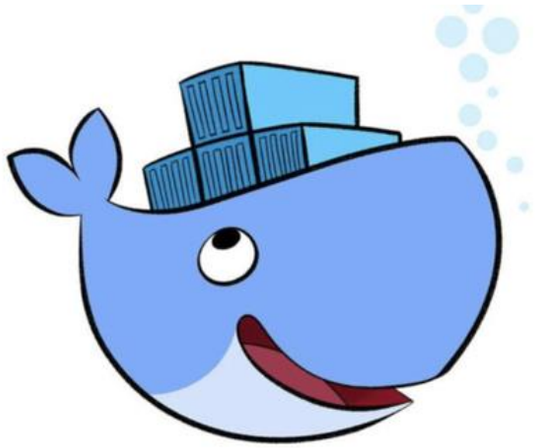
하나의 Docker 내에서 다양한 프로세스가 구동되는 것을 지양

하나의 프로세스 만을 구동하는 것을 지향

하나의 Docker 내에서 프로세스를 Background (Daemon 형태) 로 구동하는 것을 지양

프로세스를 Foreground로 구동하는 것을 지향

실행 로그도 표준출력(stdout)으로 출력



© All rights reserved by Ask Company

Docker Registry

“Docker 이미지 저장소”를 뜻하는 말

공식 저장소로서 Docker Hub (Docker계의 GitHub)

컨테이너 생성 시에 지정 이름의 이미지가 로컬에 없다면, Docker Registry로부터 이미지를 즉시 다운받습니다.

혹은 다운받는 명령 (docker pull)으로 미리 받아두어도 OK.

이 외에 각 클라우드 벤더에서 저장소 지원

Azure Container Registry

AWS Elastic Container Registry

Google Container Registry

혹은 사설 저장소 : Nexus 3, GitLab을 활용하거나, 도커 서비스로도 가능.

© All rights reserved by Ask Company

다양한 도커 이미지들

nginx

nginx:1.17.9

nginx:latest

ubuntu:18.04

alpine:3.11.3

python:3.9.0a4-buster

node:13.10.1-stretch

...

© All rights reserved by Ask Company

docker image name

python:3.7

repository:tag 형식

repository hostname/accountname/name 구조

저장소 서버의 주소를 포함한 이미지의 이름

name 으로 쓸 경우 : Docker Hub에 공개된 공식 이미지 → nginx

accountname/name 으로 쓸 경우 : Docker Hub에 공개된 일반 유저의 이미지 → askcompany/nginx

hostname/accountname/name : 다른 Docker Registry에 공개된 이미지 → myregistry.azurecr.io/samples/nginx

tag

대개 도커 이미지의 버전을 지정하는 용도로 사용합니다.

문자열 타입. 숫자로서 쓰더라도 정렬의 개념은 존재하지 않습니다.

tag를 생략하면 latest → 단순히 디폴트 이름일 뿐, 최근 버전을 자동으로 지정하는 기능이 아닙니다.

© All rights reserved by Ask Company

서버 패턴

Immutable Infrastructure 패러다임 (1)

이미지 기반 애플리케이션 배포 시나리오

인프라가 만들어지고 (거의) 변경하지 않는 상태의 인프라

다수의 서버를 동적으로 관리하는 클라우드를 기반으로 어떻게하면 유연하게 배포할 수 있을까에 대한 고민에서 나온 패러다임

© All rights reserved by Ask Company

Immutable Infrastructure 패러다임 (2)

기존의 서버를 관리한다? (X)

➔ 어떻게 하면 서버를 잘 쓰고 버리는 지에 포커스 !!!

1) 개발 단에서 만들어진 인프라를
2) 개발 단계에서 Dev Test를 거치고
3) 스테이징 단계에서 테스트를 거치고
4) 이를 프로덕션에 적용
5) 문제가 생길 경우, 현재 인프라를 수정하지 않고 새로운 버전 배포
6) 원하는 버전을 지정한 재배포 지원

© All rights reserved by Ask Company

Snowflake (눈송이) 서버 패턴

서버를 한 번 셋업하고 나서, 설정을 변경하고, 패치를 적용하는 등의 업데이트를 지속적으로 적용/운영하는 서버

새로운 서버를 세팅하고자 할 때, 동일한 환경을 구성하기 어렵고, 누락된 설정이나 패치 등에 의해서 장애가 발생하는 경우가 많다.

한 번 설정을 하고 나면 다시 설정이 불가능한 “마치 눈처럼 녹아버리는” 서버 형태

© All rights reserved by Ask Company

Phoenix (피닉스) 서버 패턴

불 속에서 다시 태어나는 (re-born) 피닉스

한 번 생성된 서버는 (거의) 수정해서 쓰지 않습니다.

새로운 서버를 세팅할 때마다, 처음 OS 설치에서부터, 소프트웨어 인스톨, 설정 변경까지 모두 반복

매번 전체 설치를 반복할 경우, 긴 시간 소요

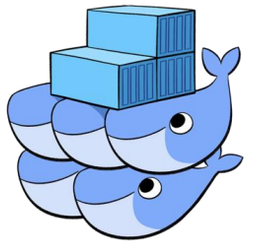
- 보통, 베이스 이미지를 만들어놓고, 차이가 나는 부분만 재설정
- Docker, Chef, Puppet, Vagrant, Packer, Serf와 같은 도구들을 활용

© All rights reserved by Ask Company

Container Orchestration

다수의 컨테이너들의 관리를 자동화해야하는 필요성

Container Orchestration



컨테이너 관리 툴의 필요성

툴이 다수의 서버에 컨테이너를 배포하고 운영하면서 Service Discovery 등의 기능을 통해, 서비스간 연결을 쉽게 해주기

사람이 일일이 서버마다 이름을 붙여주고, 하나하나 접속하여 관리하는 개념이 X

컨테이너 자동 배치 및 복제, 컨테이너 그룹에 대한 로드 밸런싱, 컨테이너 장애 복구, 클러스터 외부에 서비스 노출, 컨테이너 추가 또는 제거로 확장 및 축소, 컨테이너 서비스 간의 인터페이스를 통한 연결 및 네트워크 포트 노출 제어

패러다임의 변화

물리머신, VM 기반 ➔ 컨테이너 기반

애지중지 OS 설치, 라이브러리 업데이트, 소스 업데이트 ➔ 컨테이너 단위로 배포

https://devopedia.org/container-orchestration

© All rights reserved by Ask Company

Kubernetes → 줄여서 k8s

이미 왕좌에 오른 대세 툴.

기존에 이미 다양한 Orchestration 툴들이 있었지만,

Docker Swarm, ECS, Nomad, Marathon 등 → 대동소이

구글의 강력한 인프라 노하우로 개발된 오픈소스

일주일에 수십억개의 컨테이너들을 운영한 원칙에 따라 디자인

Microsoft, Amazon, Redhat, IBM 등의 수많은 기업들의 참여

관리형 Kubernetes 쓰세요. 직접 구축은 … No … ☹

© All rights reserved by Ask Company

Azure 에서의 Docker Container Orchestation

Azure Container Registry : Docker Registry 관리형 서비스

Azure Container Instances : VM을 직접 프로비저닝하지 않고, VM 기반의 컨테이너

Azure Kubernetes Service : k8s 관리형 서비스

Azure Web App for Containers : Docker도 지원하는 PaaS 플랫폼

Azure Functions : Serverless 플랫폼. Docker를 통한 운영도 지원.

학습자료 : https://docs.microsoft.com/ko-kr/learn/paths/administer-containers-in-azure/

© All rights reserved by Ask Company

Google 에서의 Docker Container Orchestation

Google Container Registry : Docker Registry 관리형 서비스

Google Compute Engine : 서비스 차원에서 VM으로의 Docker 지원

Google Kubernetes Engine : 본진의 k8s 관리형 서비스

Google AppEngine Flexible : PaaS 플랫폼. Docker를 통한 운영도 지원.

https://cloud.google.com/cloud-build/docs/quickstart-docker?hl=ko

© All rights reserved by Ask Company

AWS 에서의 Docker Container Orchestation

Elastic Container Registry : Docker Registry 관리형 서비스

Fargate : ECS/EKS에서 구동. 리소스 제어권을 AWS에 위임

Elastic Container Service : AWS만의 EC2 컨테이너 서비스

Elastic Kubernetes Service : k8s 관리형 서비스

Elastic Beanstalk : PaaS 플랫폼이라고 이야기하지만, 저는 동의 X. 자동화된 IaaS.

AWS Lambda : Serverless 플랫폼. AWS에서 직접 운영하는 VM에서 구동. Docker 지원 X.

https://aws.amazon.com/ko/docker/

© All rights reserved by Ask Company

앞선 내용들은 컴퓨팅 자원에 대한 것들.

이 외에도 서비스 아키텍처에 따라 다양한 서비스가 필요할 수 있습니다.

정적 스토리지 : Azure Storage, Amazon S3, Google Cloud Storage 등

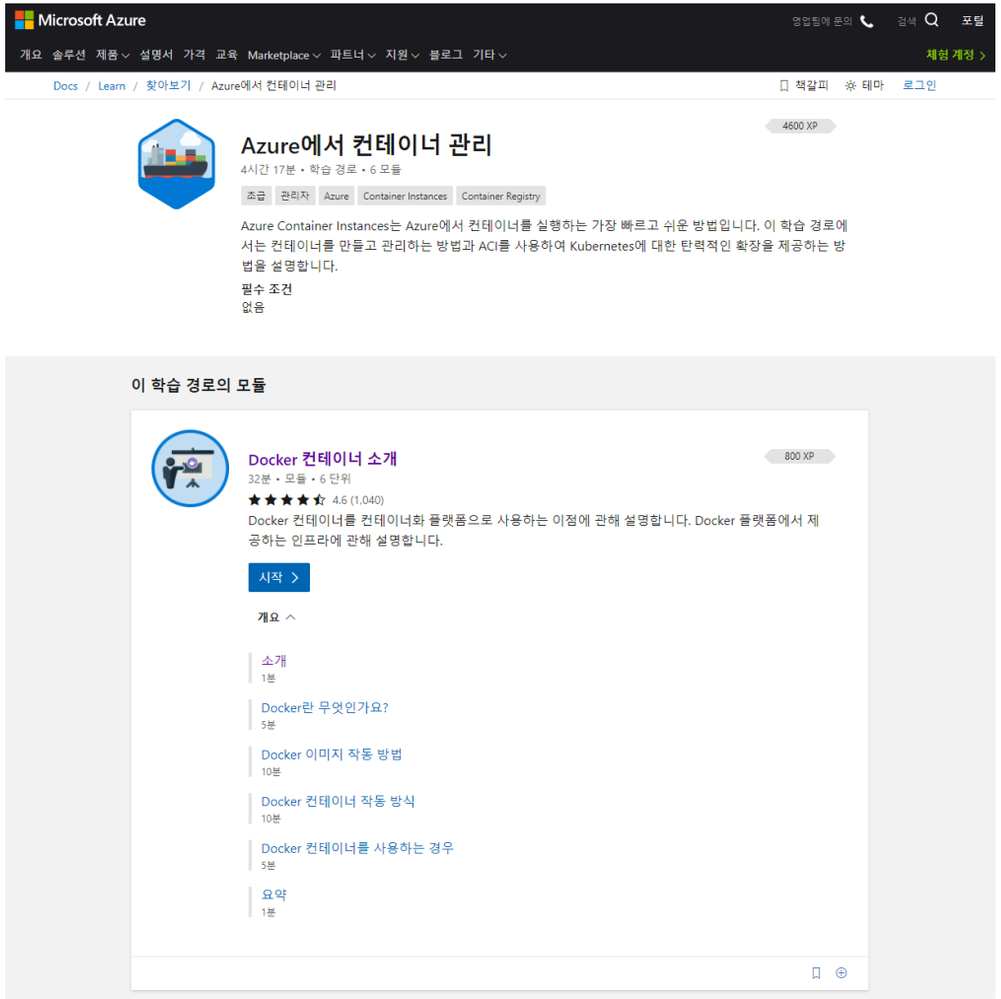
데이터베이스 : Azure Database for PostgreSQL, Amazon Aurora Serverless, Google Cloud SQL 등

캐시/Redis서버 : Azure Cache for Redis, Amazon ElasticCache, Google Cloud Memorystore 등

기타 등등 … 수많은 서비스, 인프라들 ~~~

© All rights reserved by Ask Company

(추천 자료) Microsoft Learn



https://docs.microsoft.com/ko-kr/learn/paths/administer-containers-in-azure/

© All rights reserved by Ask Company

Life is short.
You need Python and Django.

I will be your pacemaker.

© Ask Company