빠르게 살펴보는
Azure PaaS를 활용한
인프라 스트레스없는 웹서비스 배포

Ask Company 시즌2 오픈 기념

2018.07.29

모두를 위한 개발
모두를 위한 파이썬
모두에 의한 웹서비스

ASK

COMPANY

배포 스트레스 ...

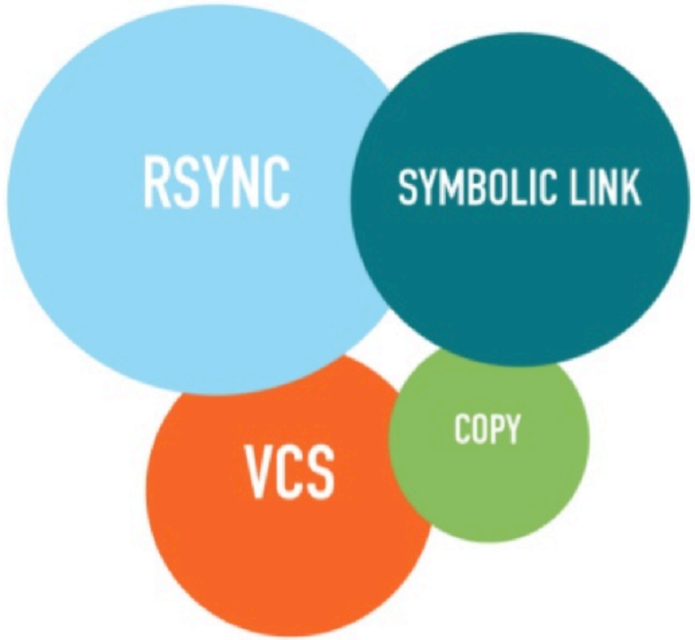
- 언어/백엔드/프론트엔드 개발 등을 익혀서 개발을 하고 나니 ...

- 다시 인프라 공부 ... ;(
  - 클라우드 플랫폼마다 기술이 다르고, 새로 나오는 기술들은 어찌나 많은 지 ...
  - 다들 AWS 많이 쓴다길래, AWS 삽질부터 많이들 시작하십니다. :)

나는 배포가 하고 싶은 데 ...

- [XECon2016] A-4 조정현 GitHub + Jenkins + Docker로 자동배포 시스템 구축하기

기존배포방식



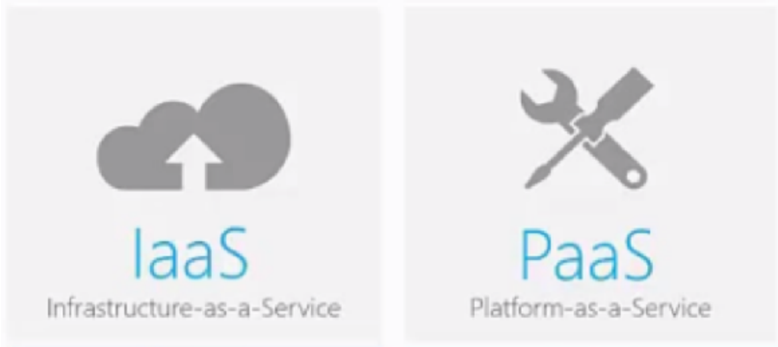
IaaS 강자. AWS

- IaaS를 가장 잘 하는 클라우드 벤더는 AWS

  - 넷플릭스를 포함한 수많은 국내외 회사들이 IaaS 베이스로 서비스

    - 왜냐하면 제대로된 PaaS가 없었고, 각 회사들은 커스텀하기 좋아하기 때문.

- 반면, **PaaS가 가장 약한 플랫폼**은 AWS



쓸만한 PaaS가 있다면 PaaS를 쓰셔야죠.

- 그 Platform을 이해하고 그에 맞춰 개발해서 올리면 바로 돌아갑니다. :D
- “클라우드 인프라 하나 세팅 못 한다면, 어찌 웹개발자라 할 수 있단 말이오~~~”
  - 응 ??? “수동차 운전을 못 한다면, 어찌 Driver라 할 수 있단 말이오 ~~~”
  - 요즘은 Auto 기어 세상. 심지어 후방 카메라도 필수 시대.
- **우리의 시간은 소중합니다. 시간이 가장 큰 비용**.

각 클라우드 벤더의 PaaS 플랫폼

- Google App Engine
  - 클라우드 벤더 중 가장 처음으로 PaaS 지원. 최신 버전은 Docker 지원

  - 참고) 왜 레진코믹스는 구글 앱엔진을 선택했나?
- AWS Elastic Beanstalk : PaaS라기보다 배포 자동화된 IaaS
- Azure
  - Azure WebApp 플랫폼을 시작으로 지원 (언어/프레임워크 제한적 지원)
    - 윈도우 IIS 웹서버에 asp.net, php, nodejs, python, java 등이 모두 설정되어있음.
  - Azure Containers for Web App 플랫폼에서 **Docker 지원** (2017년도 말)

Azure Web App

- Microsoft 기술을 가장 잘 지원하는 PaaS
- 윈도우 IIS 기반의 웹서비스 인프라
- Python도 지원하긴 하지만, 윈도우라서 제한적인 사용을 지원

Azure Containers for Web App

- 공식 사이트 : https://azure.microsoft.com/ko-kr/services/app-service/containers/
- Docker 기반의 웹서비스 인프라
- Docker를 지원하기에 지원 플랫폼의 제한이 없음.
  - 어떤 플랫폼이든 Docker Image로 팩키징만 된다면, 올려서 서비스 OK.
  - **Docker에 대해서 몇 가지 개념과 명령어를 익히시면 됩니다.**
- Azure Console **UI** 및 Azure-CLI 를 통한 관리 지원
- Docker Image 저장소에 이미지가 push되면, Azure Containers for Web App 으로의 자동 배포 지원 (연속적인 배포, Continuous Delivery)

서버 패턴

Immutable Infrastructure 패러다임

- 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
- 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

Snowflake (눈송이) 서버 패턴

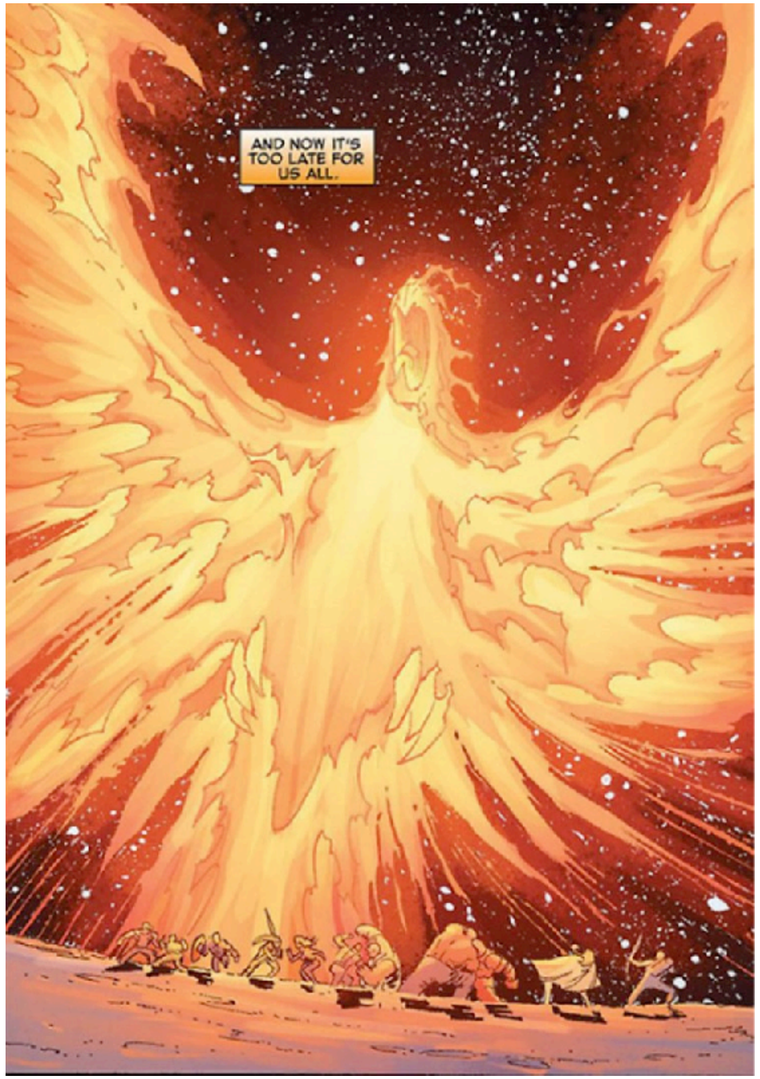
- 서버를 한 번 셋업하고 나서, 설정을 변경하고, 패치를 적용하는 등의 업데이트를 지속적으로 적용/운영하는 서버

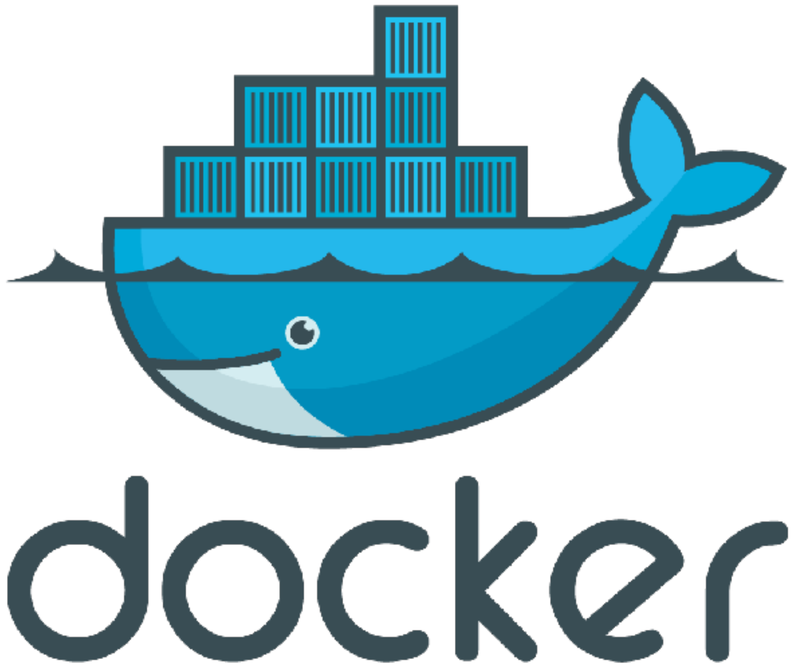
- 새로운 서버를 세팅하고자 할 때, 동일한 환경을 구성하기 어렵고, 누락된 설정이나 패치 등에 의해서 장애가 발생하는 경우가 많다.

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

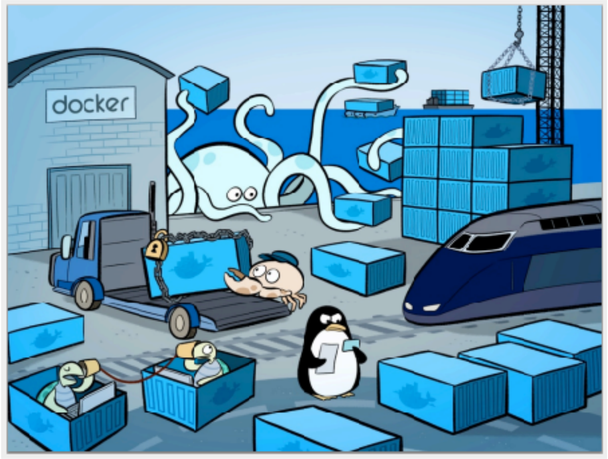
Phoenix (피닉스) 서버 패턴

- 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





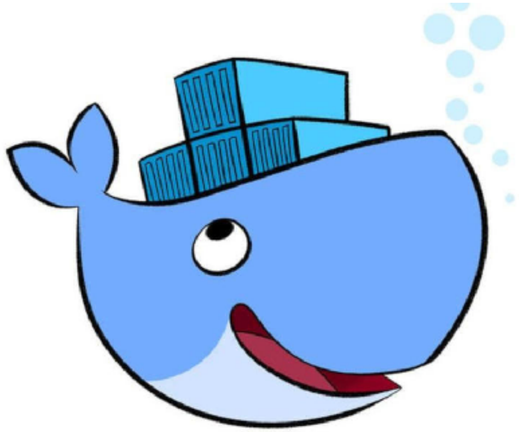
Docker ???



- 빠르고 가벼운 가상화 솔루션
- 애플리케이션과 그 실행환경/OS를 모두 포함한 소프트웨어 팩키지 => Docker Image

- 플랫폼에 상관없이 실행될 수 있는 애플리케이션 **컨테이너를 만드는 기술**
- 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 생성할 수 있습니다.
- 생성된 Docker Container는 바로 쓰고 버리는 것 (Immutable Infrastructure 패러다임)
  - 비교) 예전 VM은 한 번 생성하면 애지중지. 관리 및 업데이트
- Docker Container는 격리되어있어서, 해킹되더라도 Docker Engine이 구동되는 원래의 서버에는 영향을 끼치지 않음.

Docker 만의 특징/유의사항



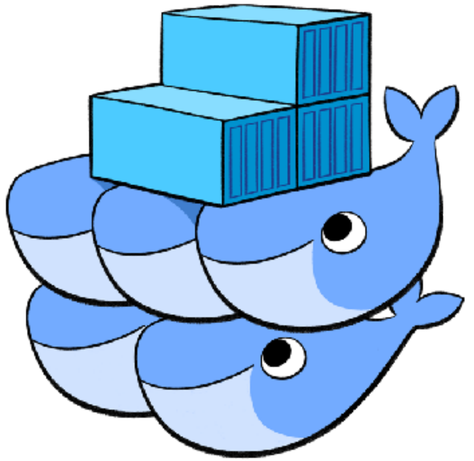
- Docker 내에서 어떤 프로세스가 도는 지 명확히 하기 위해서.

  - 하나의 Docker 내에서 다양한 프로세스가 구동되는 것을 지양
    - **한 종류의 프로세스 만을 구동하는 것을 지향**
  - 하나의 Docker 내에서 프로세스를 Background (Daemon 형태) 로 구동하는 것을 지양
    - **프로세스를 Foreground로 구동하는 것을 지향** nginx 예시: nginx -g daemon off;
    - **실행 로그도 표준출력(stdout)으로 출력**

Container Orchestration

- 컨테이너 관리 툴의 필요성
  - 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
- 주요 도구
  - 구글의 Kubernetes : Azure/AWS/Google 에서도 지원
  - Docker Swarm
  - Apache Mesos
- Azure Containers for Web App은 **웹서비스 전용**, Orchestration 서비스

kubernetes



Docker Registry

https://www.docker.com/sites/default/files/oyster-registry-3.png

- “Docker 이미지 저장소”를 뜻하는 말

- 공식 저장소로서 Docker Hub : https://hub.docker.com/ (Docker계의 GitHub)
- 이 외에 각 클라우드 벤더에서 저장소 지원
    - Azure Container Registry : https://azure.microsoft.com/ko-kr/services/container-registry/
    - AWS Elastic Container Registry : https://aws.amazon.com/ko/ecr/
- Azure Containers for Web App에서는 지정 Docker Registry로부터 이미지를 읽어들여, Docker Container를 적재합니다.

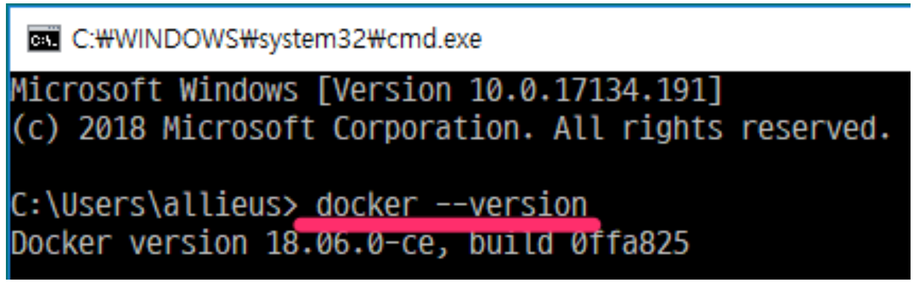
Dockerfile

- Docker 이미지를 만들 때, 수행할 명령과 설정들을 시간 순으로 기술한 파일


```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y python3-pip python3-dev && apt-get clean

RUN mkdir /code
WORKDIR /code
ADD requirements.txt /code/
RUN pip3 install -r requirements.txt
ADD . /code/

EXPOSE 8000
CMD ["python3", "/code/manage.py", "runserver", "0.0.0.0:8000"]
```

개발환경 설치하기

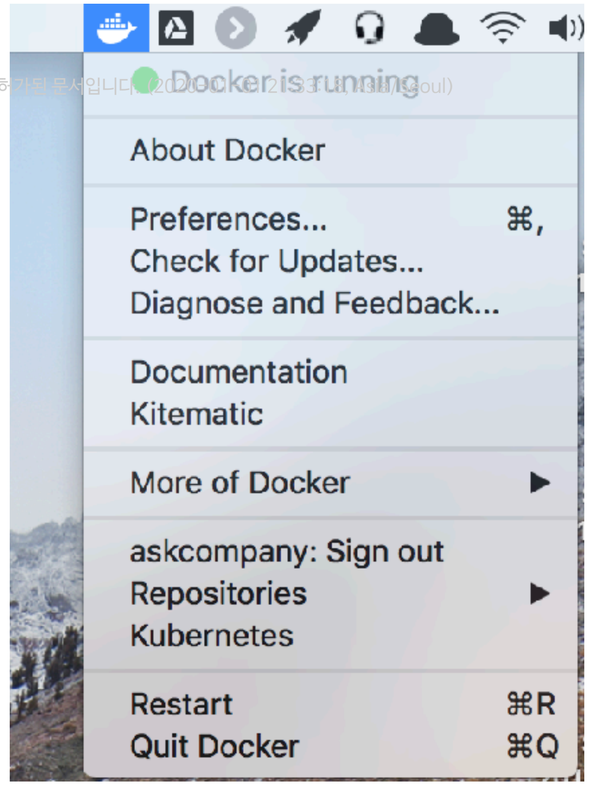
Docker 딱 하나. 설치



- 각 OS에 맞게 Docker를 설치하시고, 버전을 확인해보세요.
  - “Error response from daemon: Bad response from Docker engine” 와 같은 메세지가 출력될 경우, docker daemon이 실행 중이 아니거나, root 권한이 필요하실 수 있습니다. linux/mac 이라면 sudo docker —version 명령을 입력해보세요.

- 다운받을 때 docker.com 사이트에서 인증을 요구하니, 회원가입/로그인을 미리 해주세요.
- **윈도우 10 Pro 이상** : https://docs.docker.com/docker-for-windows/
- 윈도우7 이상, 윈도우 10 Home 이하 버전 : Docker ToolBox
  - https://docs.docker.com/toolbox/toolbox_install_windows/
  - VirtualBox와 Git이 같이 설치되기에, VirtualBox가 이미 설치되어있으시면 미리 제거하셔야 합니다.
- 맥에서 설치하기 : https://docs.docker.com/docker-for-mac/
- 리눅스에서 설치하기 : `curl -fsSL https://get.docker.com/ | sudo sh`

Docker Daemon 실행을 확인해주세요.

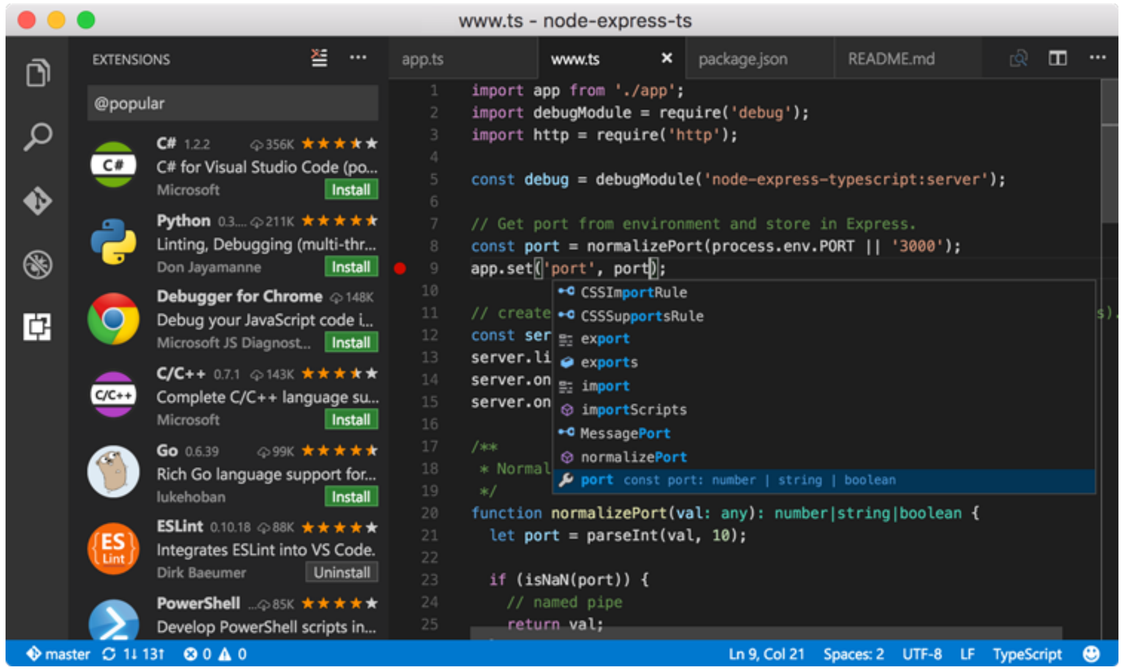


리눅스에서는 OS에 맞춰, 실행여부를 확인해주세요.
Ubuntu Linux에서는 sudo service docker start 와 같은 명령으로 서비스를 시작해주셔야할 수도 있습니다.

Visual Studio Code 설치

- https://code.visualstudio.com/

- 간단한 명령어 몇 줄 정도만 입력할 거 예요. 겁먹지 마세요. ㅎㅎㅎ



Docker 놀이터

파이썬 새 버전 써보기


```
→ docker run --rm -it python:3.7-stretch
Python 3.7.0 (default, Jul 17 2018, 11:04:33)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.version
'3.7.0 (default, Jul 17 2018, 11:04:33) \n[GCC 6.3.0 20170516]'
>>> exit()
```

nginx 웹서버 띄우기


```
# nginx 이미지를 통해 mynginx Container 적재 및 실행
# nginx 이미지는 80포트로 listen으로 세팅되어있음.
# host의 8080포트와 container의 80포트를 연결
docker run --detach --publish 8080:80 --name mynginx nginx

# 웹브라우저로 http://localhost:8080 주소로 접속해보세요.

# mynginx Container 정지
docker stop mynginx

# mygninx Container 삭제
docker rm mynginx
```

호스트 머신측 컨텐츠로 html 서빙하기

- 현재 디렉토리내 **html/index.html** 생성

  - 내용: **hello world**

```
# 맥/리눅스) --volume 옵션 추가
docker run -d -p 8080:80 --volume `pwd`/html:/usr/share/nginx/html --name mynginx nginx

# 윈도우, 명령프롬프트) --volume 옵션 추가
docker run -d -p 8080:80 --volume %cd%\html:/usr/share/nginx/html --name mynginx nginx

# 윈도우, 파워쉘) 웹브라우저로 http://localhost:8080/ 접속 테스트
docker run -d -p 8080:80 --volume ${PWD}/html:/usr/share/nginx/html --name mynginx nginx

# mynginx Container를 정지 및 제거
docker stop mynginx
docker rm mynginx
```

Tip: docker run 시에 --rm 옵션을 붙이면, stop시에 자동 remove 됩니다.
Tip: docker rm -f mynginx 명령은 stop하지 않고도 강제로 remove할 수 있습니다.

샘플 Django 프로젝트로 놀아봅시다.

우리가 쓸 서비스들

- Django 서버 by **Azure Containers for Web App**
  - Docker Registry 서비스 by **Docker Hub** (또는 Azure Container Registry)
- 정적 파일 저장소 (static/media) by **Azure Storage Accounts**
- 관계형 데이터베이스 by **Azure Database for PostgreSQL**
- 카카오톡 플러스 친구 API

샘플 Django 프로젝트

- Github 소스코드 저장소 Clone
  - github.com/allieus/deploy-with-azure-paas-get-started
  - 단축 URL : bit.ly/2OqQVPg (알파벳 대소문자 구별해주세요.)
- Azure 리소스 생성
  - Resource Group 생성
  - Storage Accounts 생성하고, Storage account name과 Key 조사
  - Azure Database for PostgreSQL 생성하고, 접속정보 조사
  - Azure Containers for Web App 은 이후에 생성

Docker 이미지 빌드 및 DB Migrate

```
호스트머신> docker build -t 이미지명 .
호스트머신> docker run \
    -e AZURE_ACCOUNT_NAME="----------" \
    -e AZURE_ACCOUNT_KEY="----------" \
    -e DB_HOST="----------" \
    -e DB_NAME="postgres" \
    -e DB_USER="----------" \
    -e DB_PASSWORD="----------" \
    --rm -it \
    이미지명 sh
```


Docker 이미지 내에 파일을 생성한 것은 없습니다.
단지 본 Docker Container를 활용해서 Azure 서비스에 static 파일을 올리고
데이터베이스 마이그레이션을 수행했습니다.

```
/code # python3 manage.py collectstatic --no-input
/code # python3 manage.py showmigrations
/code # python3 manage.py migrate
/code # python3 manage.py createsuperuser
/code # exit

호스트머신>
```

복사할 코드는 bit.ly/2LQtvo9 페이지에서 업데이트됩니다.

Azure 리소스 기반으로 서버 실행

Docker Container는 80포트를 listen 중
Host 머신의 8888포트와 80포트를 연결

```
호스트머신> docker build -t 이미지명 .
호스트머신> docker run \
    -p 8888:80 \
    -e AZURE_ACCOUNT_NAME="----------" \
    -e AZURE_ACCOUNT_KEY="----------" \
    -e DB_HOST="----------" \
    -e DB_NAME="postgres" \
    -e DB_USER="----------" \
    -e DB_PASSWORD="----------" \
    --rm -it \
    이미지명

# 브라우저로 http://localhost:8888/ 로 접속해보세요.
```

Docker Hub 가입

- https://hub.docker.com/ 에서 먼저 회원가입 진행하고, 필히 확인메일 확인


```
쉘> docker login
Login with your Docker ID to push and pull images from Docker Hub. If
you don't have a Docker ID, head over to https://hub.docker.com to
create one.
Username: 여러분의아이디
Password: 여러분의암호
Login Succeeded
```

Docker Hub용으로 빌드 및 push

```
호스트머신> docker build -t askcompany/deploy-with-azure-pass-get-started:1.0 .
호스트머신> docker push askcompany/deploy-with-azure-pass-get-started:1.0
```

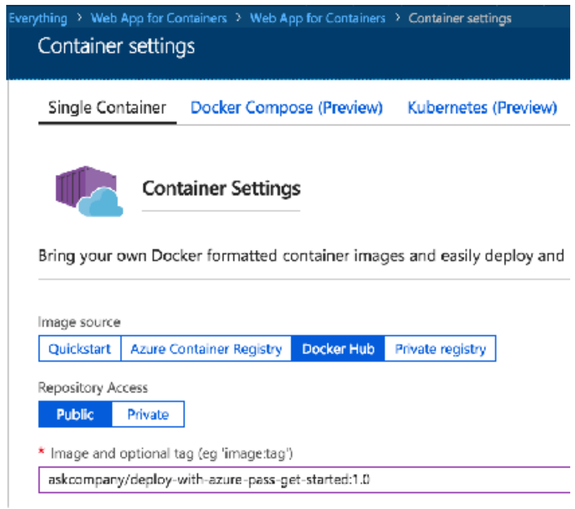

업로드된 저장소 주소
https://hub.docker.com/r/askcompany/deploy-with-azure-pass-get-started/tags/

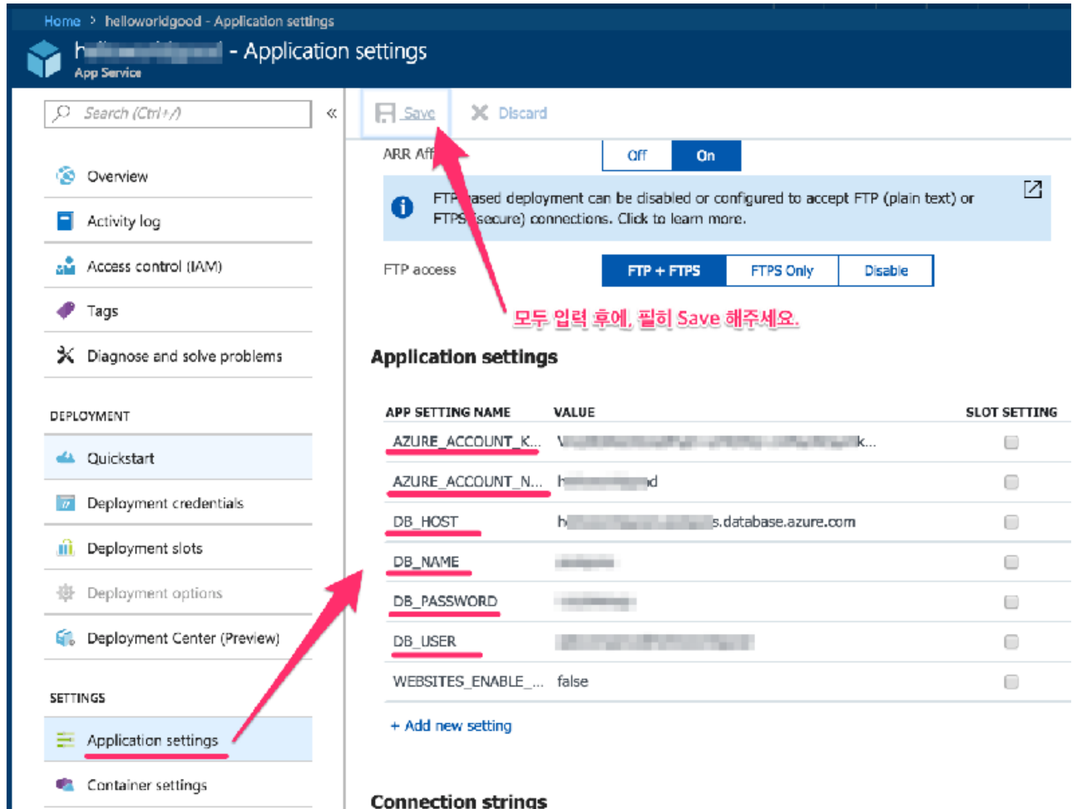
- 디폴트로 public 저장소로 생성
  - Free Plan에서는 1개의 private 저장소가 제공

Azure Containers for Web App 생성 고고.

- Azure Portal 사이트에서
  - 1) Docker Image 지정하고

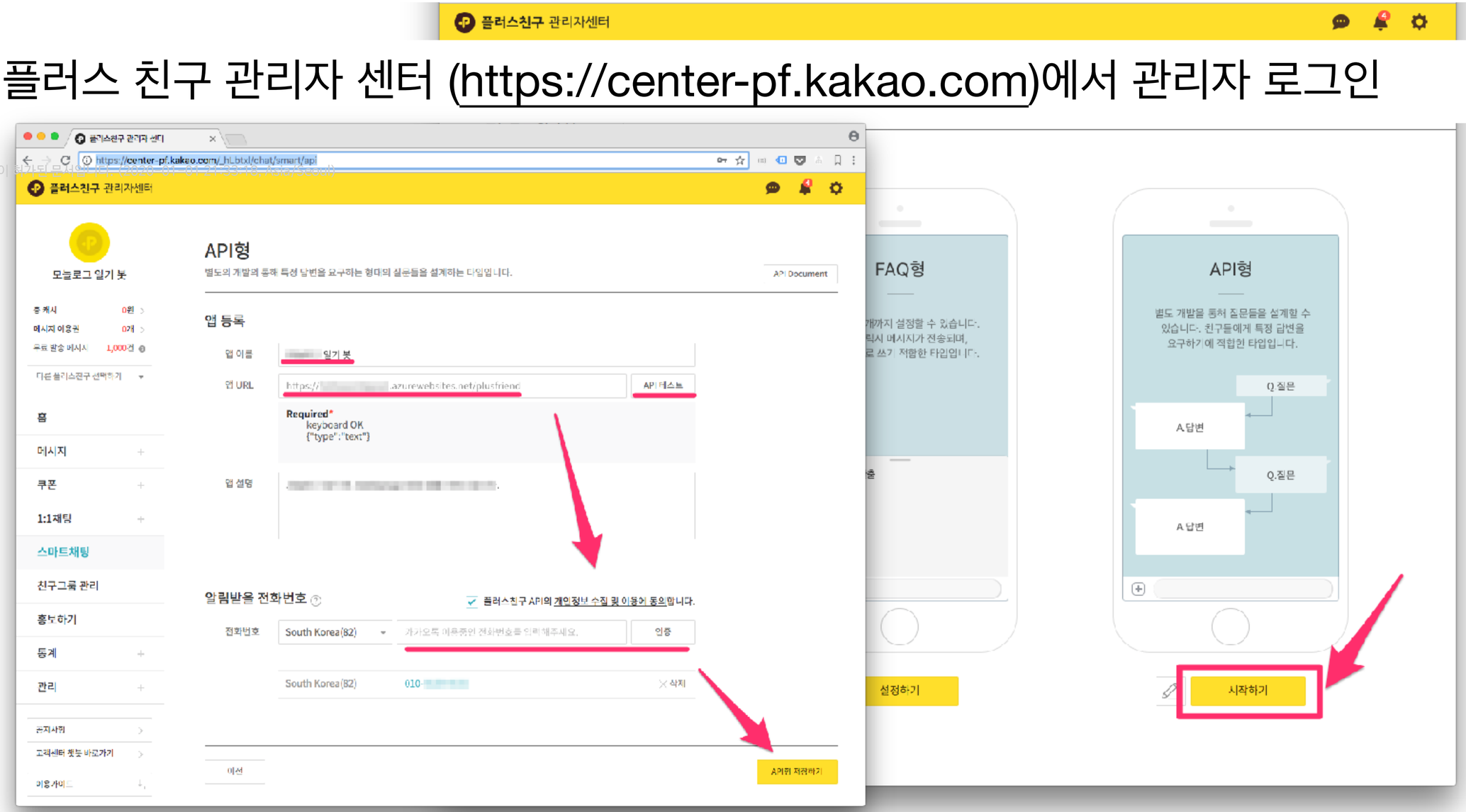
  - 2) 필요한 환경변수 지정하고, 재시작 (settings => Application settings)
  - 3) Always On 옵션 켜기 (Standard Plan 이상에서 지원)
  - 3) Out Bound 아이피 조사





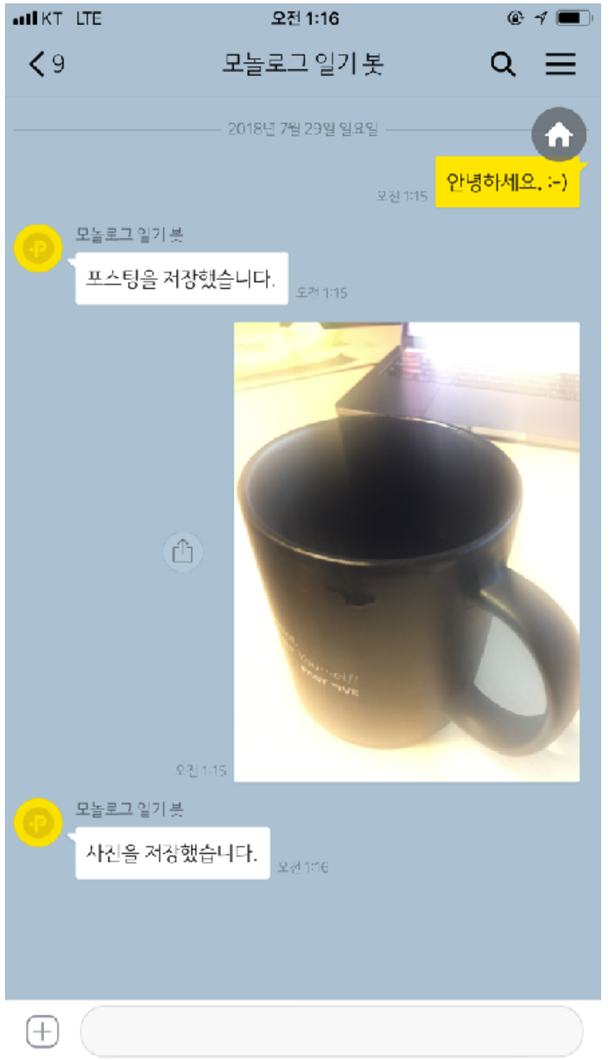
카카오톡 플러스 친구에 등록

- 플러스 친구 관리자 센터 (https://center-pf.kakao.com)에서 관리자 로그인



카카오톡 플러스 친구 테스트

- 카카오톡에서 만드신 플러스 친구를 검색해서, 말을 걸어 보세요.



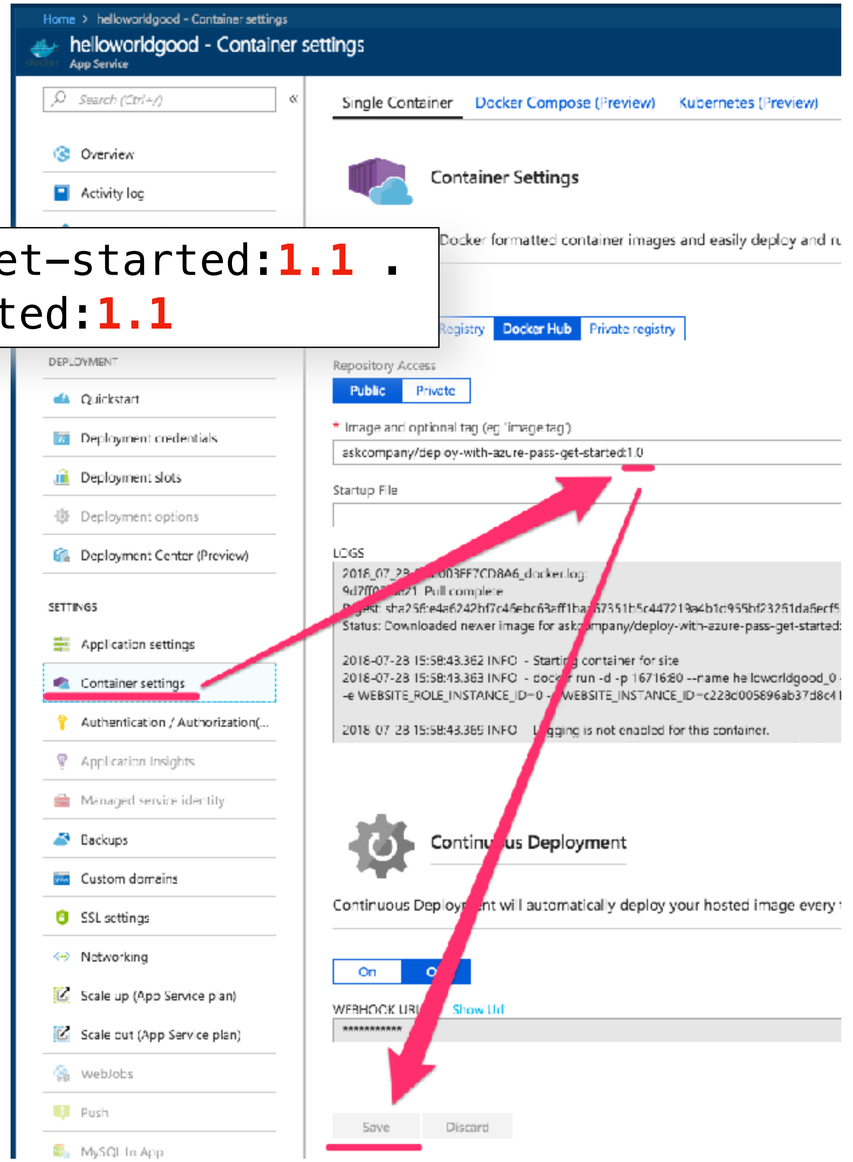
프로젝트가 업데이트되면

- **태그**를 변경하고, 프로젝트 다시 빌드 및 docker hub로 push

```
호스트머신> docker build --tag askcompany/deploy-with-azure-pass-get-started:1.1 .
호스트머신> docker push askcompany/deploy-with-azure-pass-get-started:1.1
```

- 운영 중인 Azure Containers for Web App의 Containers settings에서 태그 변경 후에 "Save"

- 수 분 후에 업데이트



Azure 리소스 제거

- 처음 생성한 Resource Group만 날리면, 모두 한 방에 제거 !!!

실서비스에서는 필히 DEBUG=False

- 현재 코드에서는 settings.DEBUG 옵션을 켜뒀지만,
- 실서비스에서는 필히 옵션을 꺼주세요.
  - 유저에게 불필요하게 서버 정보를 노출하지 않습니다.
  - 서버측에서 실행한 쿼리내역이 메모리에 계속 쌓여, 메모리가 부족해질 수 있습니다.

Ask Company VOD에서 앞으로 다룰 내용들

- Docker 차근차근 살펴보기

- 전략적 배포 기법 : Canary release, Blue/Green deployment, A/B Test
- Azure 플랫폼을 활용한 ~
  - 연속적인 통합 (Continuous Integration)
  - 연속적인 배포 (Continuous Delivery)
  - 모니터링 등등.
- Azure 플랫폼을 어느 정도 정리 후에, AWS를 다룰 예정

“여러분의 파이썬/장고 페이스메이커가 되겠습니다.”

– *Ask Company* 이진석

https://www.askcompany.kr
me@askcompany.kr