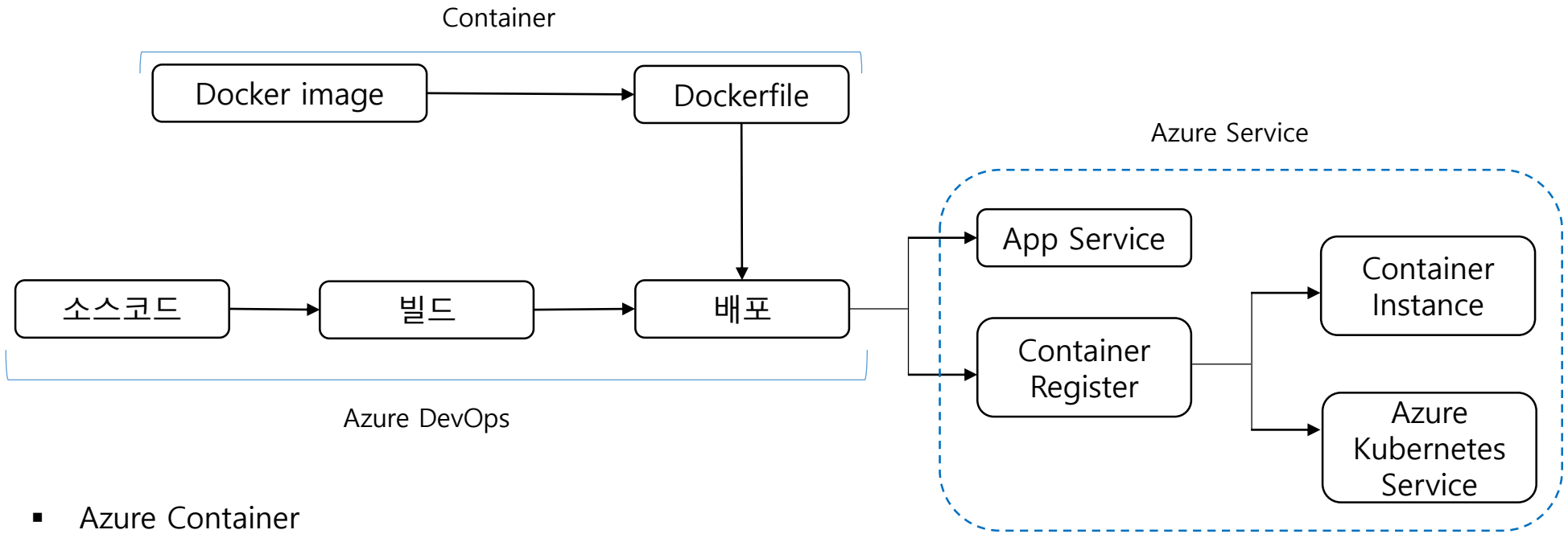
Docker기반 Azure Container

Azure Container Register & Azure Container Instance

SPTek

❖ Azure Service Release Process



- Azure Container
  - Docker의 Container를 ACR에 등록하고 ACI를 기반으로 서비스
- Azure DevOps & Azure App Service
  - DevOps의 build-pipeline과 release-pipeline를 통해 app service 기반으로 서비스
- Azure DevOps & Azure App Service for Container
  - DevOps의 build-pipeline과 release-pipeline를 통해 ACI를 기반으로 서비스
- Azure DevOps & Azure Kubernetes Service
  - DevOps의 build-pipeline과 release-pipeline를 통해 Azure Kubernetes Service를 기반으로 서비스

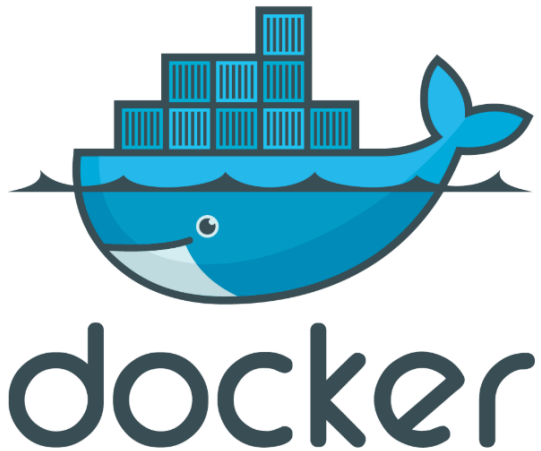
Contents

SPTek

1. Docker의 개요

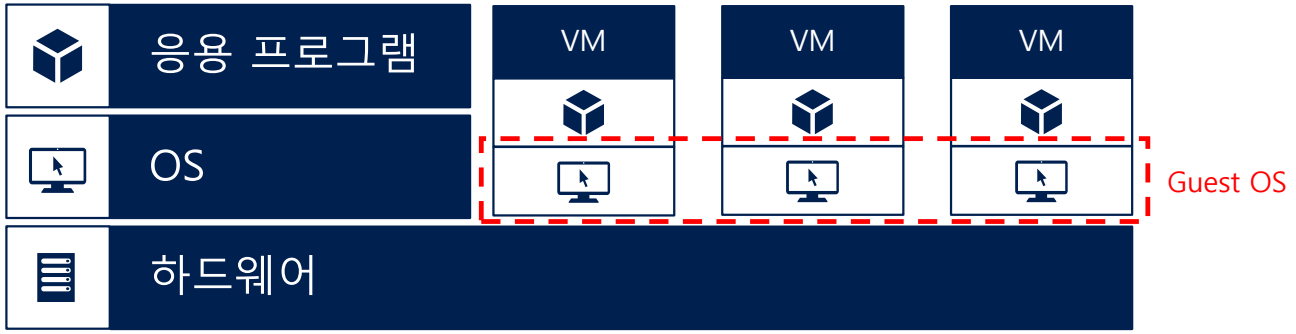
1.1.1 Docker(도커)란 무엇인가?

- 도커는 2013년 3월 산타클라라에서 열린 Pycon Conference에서 dotCloud의 창업자인 Solomon Hykes가 The future of Linux Containers 라는 세션을 발표하면서 처음 세상에 알려졌습니다.
- Docker Inc.에 의해 개발 된 Go 언어로 작성된 소프트웨어입니다. 원래는 컨테이너를 실현하기 위해 LXC(Linux Container)을 사용하고있었지만, Docker 0.9 버전 부터는 기본으로 직접 만든 자체 컨테이너를 사용하고 있습니다.
- Docker는 애플리케이션을 신속하게 구축, 테스트 및 배포할 수 있는 소프트웨어 플랫폼입니다.
- Docker는 소프트웨어를 컨테이너라는 표준화된 유닛으로 패키징 됩니다.
- 컨테이너에는 라이브러리, 시스템 도구, 코드, 런타임 등 소프트웨어를 실행하는 데 필요한 모든 것이 포함
- Docker를 사용하면 환경에 구애받지 않고 애플리케이션을 신속하게 배포 및 확장할 수 있으며 코드가 문제없이 실행 수 있는 환경을 제공



1.1.2 Docker의 구조

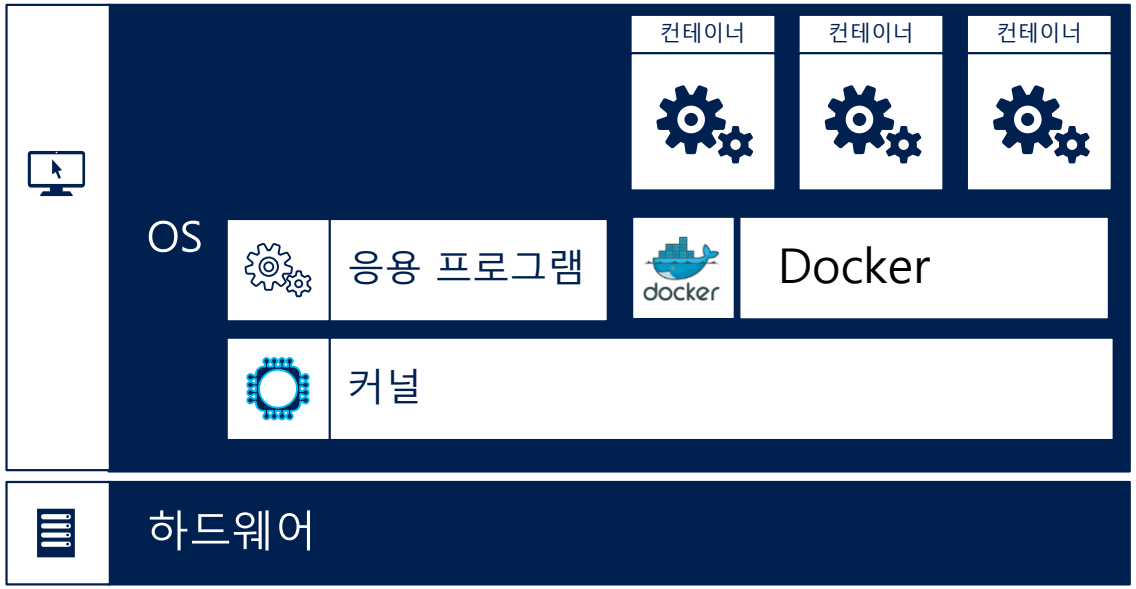
➢ 전통적인 Virtural Machine ①



- 우리에게 익숙한 VMware나 VirtualBox같은 가상머신은 호스트 OS위에 게스트 OS 전체를 가상화하여 사용하는 방식입니다.
- 이 방식은 여러가지 OS를 가상화(리눅스에서 윈도우를 돌린다던가) 할 수 있고 비교적 사용법이 간단하지만 무겁고 느려서 운영환경에선 문제가 발생함

1.1.2 Docker의 구조

➢ 컨테이너(Container) ②

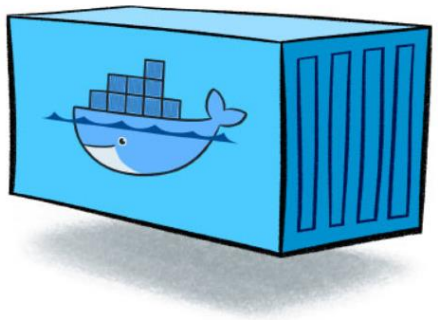


- 실행 가능한 Docker 이미지를 의미한다.
- 각 Container들은 Host 및 다른 Container등과 완전히 격리된 공간을 구성하며 Image를 Base로 한 환경에서 격리된 공간의 리소스에 접근 할 수 있게 구성되어 있다.
- 하이퍼바이져가 OS와 커널 전체를 가상화하는 반면, Container는 FileSystem의 가상화만 이루어짐
- Container는 Host PC의 커널을 공유하고, Host PC의 자원을 격리 된 상태로 이용하기 때문에 성능상 이점이 있음

1.1.2 Docker의 구조

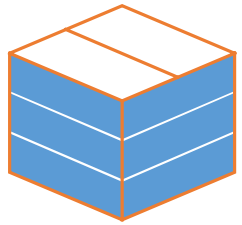
➢ 이미지(Image) ③

- Docker 컨테이너를 만들기 위해 Read Only Layer
- 각 Image들은 Docker엔진 위에서 다른 Image들을 Base로 하는 Image Layer를 구성하기 때문에 여러 이미지의 재사용을 통해 새로운 이미지를 빌드하는 것이 가능함

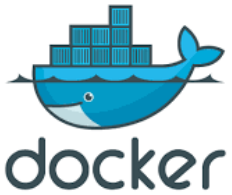


1.1.3 Docker의 운영

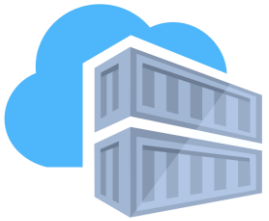
➢ Docker 기반 운영환경 ①



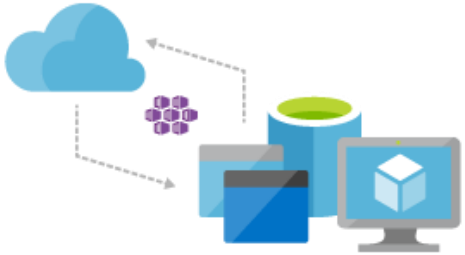
컨테이너 이미지에서
컨테이너가 생성됨



컨테이너는 Docker로 관리됨



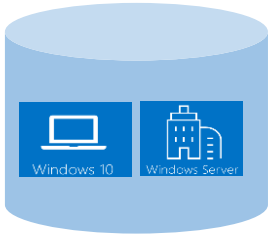
컨테이너 이미지는 컨테이너
리포지토리에 저장됨



컨테이너 호스트에서 오케스트레이터로 배포되고 업데이트됨

(Kubernetes 혹은 Docker Swarm)

관리



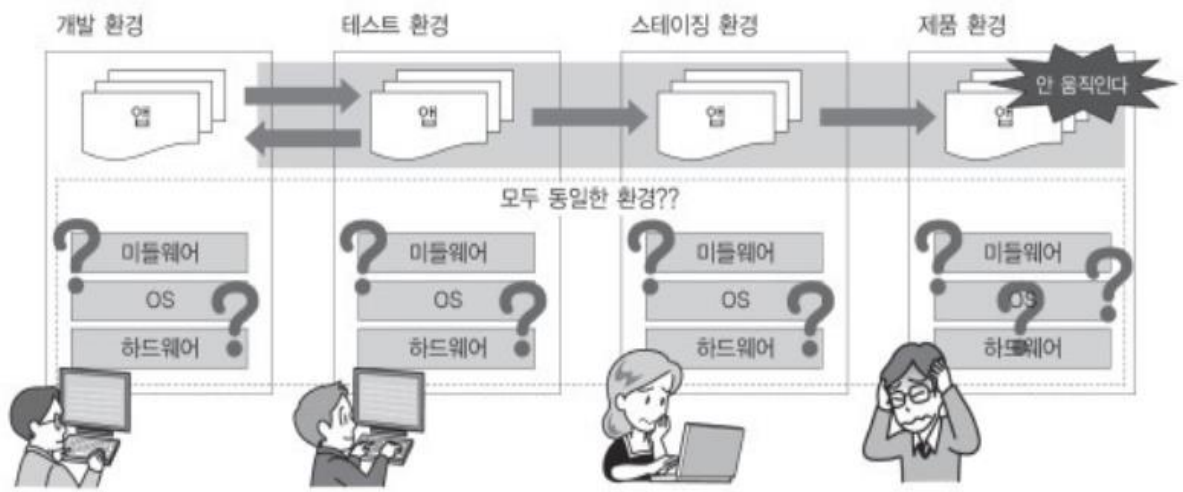
컨테이너는
컨테이너 호스트에서 호스팅

SPTek

1.1.3 Docker의 운영

➢ 전통적인 운영환경 ②

- 폭포수형 모델



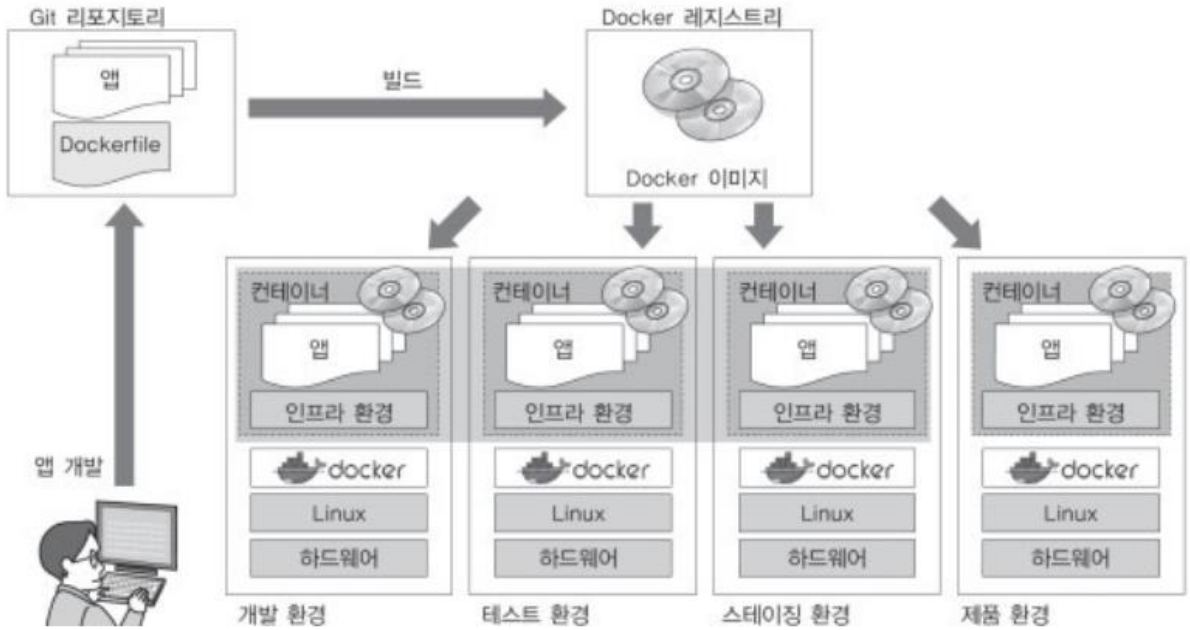
[이미지 참조 : 정보문화사]

- 개발 환경이나 테스트 환경에서는 올바르게 작동해도 스테이징 환경에서나 제품 환경으로 전개하면 정상적으로 작동하지 않는 경우도 있습니다.

1.1.3 Docker의 운영

➢ 컨테이너 운영환경 ③

- 어플리케이션



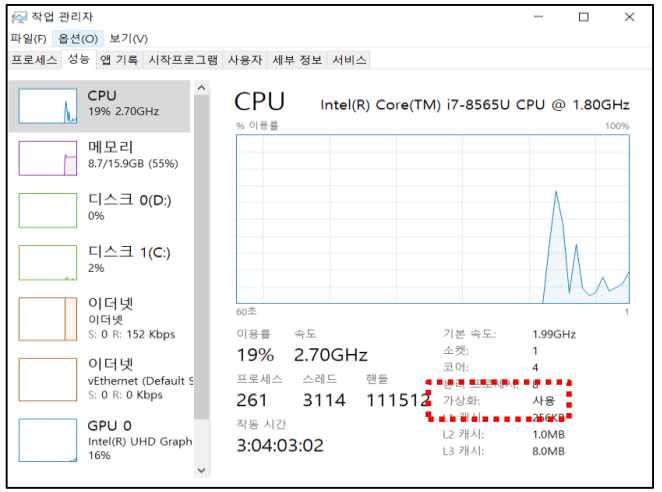
[이미지 참조 : 정보문화사]

- Docker에서는 애플리케이션의 실행에 필요한 모든 파일 및 디렉토리들을 컨테이너로서 구성함
- Docker 이미지를 Docker Hub와 같은 리포지토리(repository)에서 공유합니다.

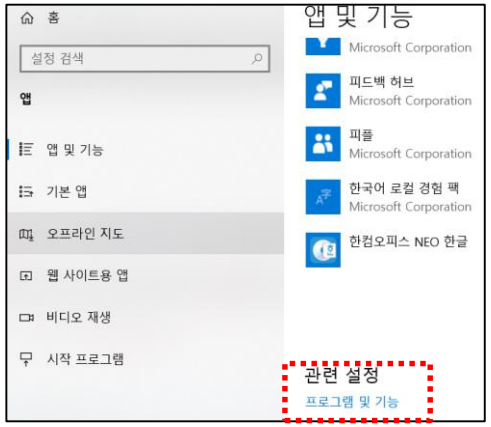
1.2.1 Docker의 다운로드

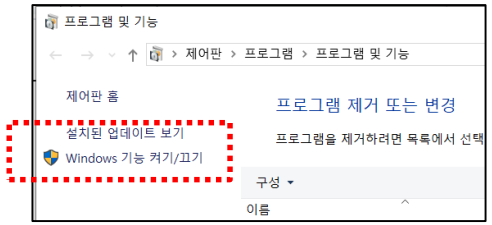
➢ 윈도우 docker 설치 준비 ①

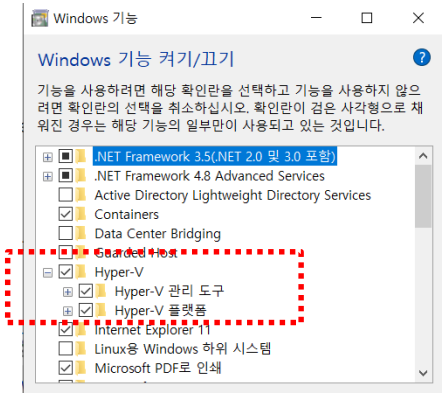
- Windows 10 x64 Pro, Enterprise 또는 Education (빌드 15063 이상)이 필요.
- CPU 가상화 기능이 활성화
  '작업관리자'에서 '가상화'가 '사용'으로 되어 있는지 확인



- 'Microsoft Hyper-V' 옵션의 실행
  윈도우 10 에서 'Windows 기능 켜기/끄기' 를 실행한 다음, Hyper-V 옵션을 활성화



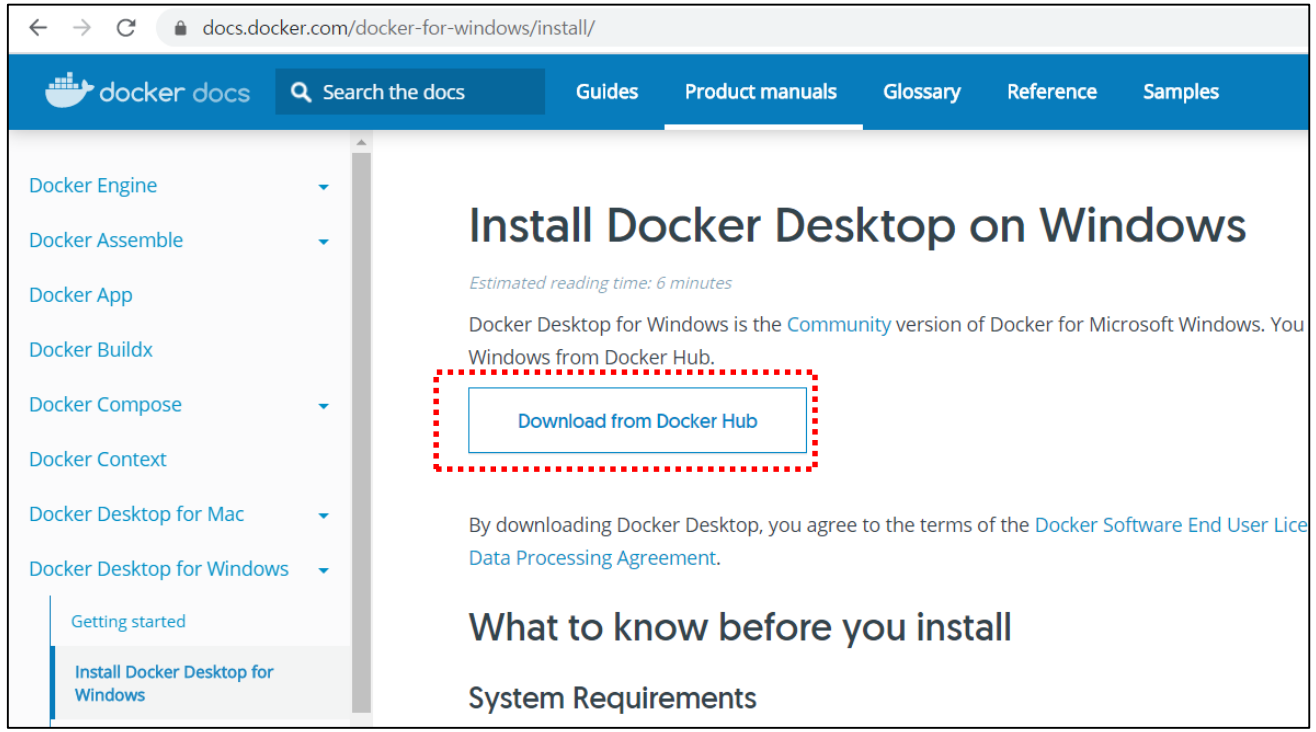




1.2.1 Docker의 다운로드

➢ 윈도우 docker 다운로드 경로 ②

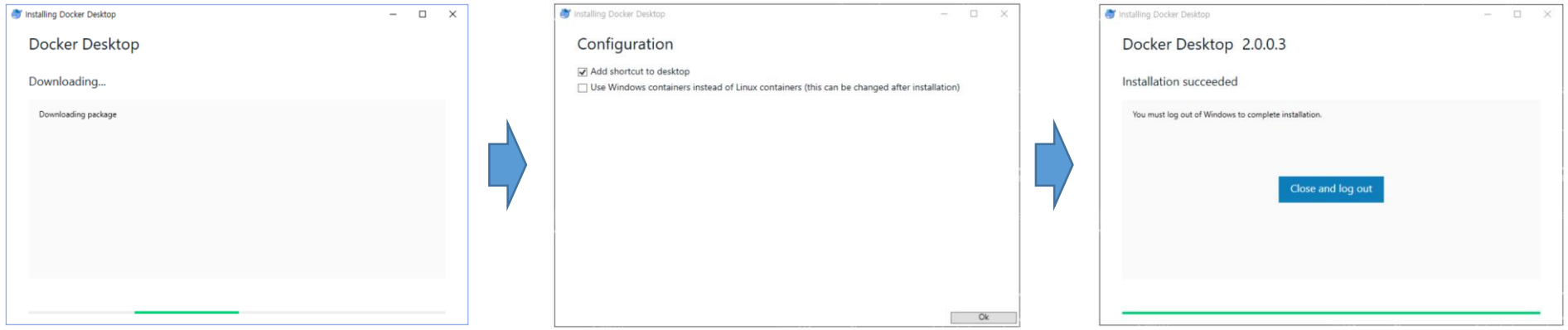
- https://docs.docker.com/docker-for-windows/install/



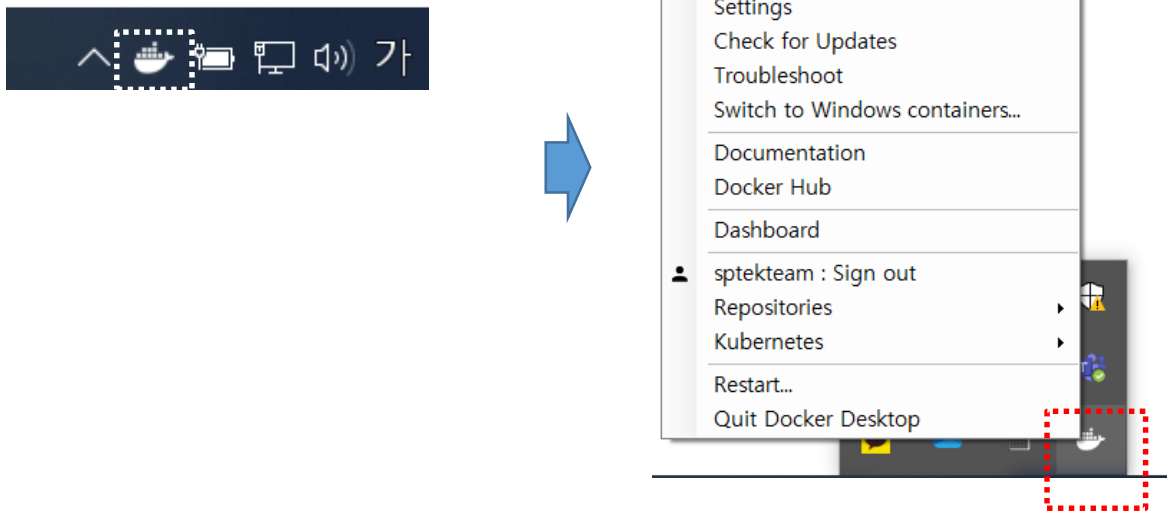
1.2.2 Docker의 설치

➢ 윈도우 docker 설치

- Docker 포털에서 받은 'Docker Desktop Installer.exe' 의 실행



- 설치완료



1.2.3 Docker Hub

- Docker는 각종 공식 어플리케이션들의 공식 이미지와 사용자 개인의 이미지까지 올리고 받을 수 있는 공식 Repository인 Docker Hub를 제공
- Docker Hub - https://hub.docker.com/
- Docker Hub는 GitHub 등의 소스 형상 관리 툴과 연계되어 코드를 build해서 자동으로 이미지화 하는 기능을 제공한다.
- 또한 Azure, AWS 등의 PaaS서비스와 연계해서 어플리케이션을 Deploy하는 기능까지 제공한다.

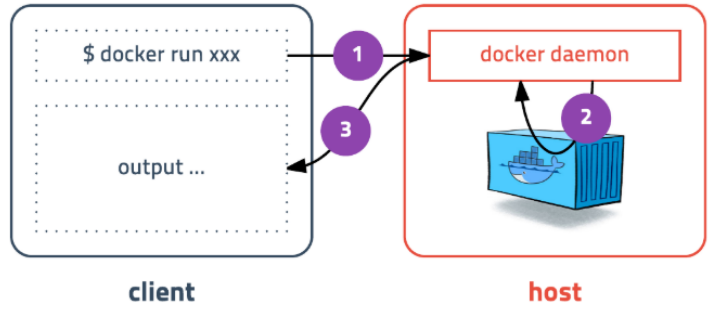
1.3.1 Docker 설치 확인

➢ 설치 버전 확인

- 명령어 : **docker version**
- Output :

```
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:22:37 2019
 OS/Arch:           windows/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea
  Built:            Wed Nov 13 07:29:19 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v1.2.10
  GitCommit:
b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:
3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```



docker client-host

[이미지 참조 : subicura.com]

- 도커는 하나의 실행파일이나 클라이언트와 서버역할을 수행
- 커맨드를 입력하면 도커 클라이언트가 도커 서버로 명령을 전송
- 결과를 받아 터미널에 출력함

SPTek

1.3.2 Docker 이미지 확인

➢ 이미지 확인

- 명령어 : **docker images**
- Output :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sptek-shopping	v01	867a401712ef	6 days ago	520MB
sptek-member	v3	df557bbe96b3	12 days ago	520MB
mymbrcontainreg.azurecr.io/sptek-member	v3	df557bbe96b3	12 days ago	520MB
tomcat	8.5.46-jdk8-openjdk	8973f493aa0a	4 months ago	508MB

1.3.3 Docker 이미지 검색

➢ 이미지 검색

- 2가지 방법으로 검색 가능
  - ➢ Docker Hub Web에 접속해 tomcat을 검색한다
  - ➢ docker search 명령어를 사용한다.

- 명령어 : **docker search [옵션] <검색어>**

  ※ 옵션

  --automated=false : Automated Build만 표시

  --no-trunc=false : 모든 결과를 다 표시

  -s[--stars=n] : star 수가 n개 이상인 결과만 표시

- 예제 : **docker search tomcat**

- Output :

```
NAME                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
tomcat              Apache Tomcat is an open source implementati...  2636      [OK]
tomee               Apache TomEE is an all-Apache Java EE certif...  74        [OK]
dordoka/tomcat      Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base...  53                  [OK]
bitnami/tomcat      Bitnami Tomcat Docker Image                      31                  [OK]
kubeguide/tomcat-app  Tomcat image for Chapter 1                     28
consol/tomcat-7.0   Tomcat 7.0.57, 8080, "admin/admin"               17                  [OK]
.....
```

※ 공식이미지를 포함해 여러 public 이미지들이 Stars 순서대로 나열된다.
star는 docker hub를 사용하는 누군가가 해당 레파지토리를 즐겨찾기 한 것이다.
즉, 인기순으로 정렬된다.

1.3.4 Docker 이미지 다운로드

➢ 이미지 다운로드

- 명령어 : **docker pull [옵션] <이미지명>:[태그명]**

  ※ 옵션

  -a : 모든 태크의 이미지를 받기

  태크명 생략 : 기본적으로 가장 최신 버전(latest)를 다운로드 함

- 예제 : **docker pull tomcat**
- Output :

```
Using default tag: latest
latest: Pulling from library/tomcat
dc65f448a2e2: Pull complete
346ffb2b67d7: Pull complete
dea4ecac934f: Pull complete
8ac92ddf84b3: Pull complete
d8ef64070a18: Pull complete
6577248b0d6e: Pull complete
576c0a3a6af9: Pull complete
6e0159bd18db: Pull complete
944191e51caa: Pull complete
9ee6a5ca751e: Pull complete
Digest: sha256:d53c2079ea67db92f6d7c39e9450f641610336016fdddef5392c5afd41518e5e
Status: Downloaded newer image for tomcat:latest
docker.io/library/tomcat:latest
```

docker images

```
REPOSITORY                                TAG                 IMAGE ID       CREATED         SIZE
tomcat                                    latest              b56d8850aed5   2 days ago      529MB
sptek-shopping                            v01                 867a401712ef   6 days ago      520MB
sptek-member                              v3                  df557bbe96b3   12 days ago     520MB
mymbrcontainreg.azurecr.io/sptek-member   v3                  df557bbe96b3   12 days ago     520MB
tomcat                                    8.5.46-jdk8-openjdk 8973f493aa0a   4 months ago    508MB
```

1.3.5 Docker 이미지 삭제

➢ 이미지 삭제

- 명령어 : **docker rmi [저장소] <이미지명, ID>:[태그명]**

  ※ 옵션

  -f, --force=false : 이미지 강제 삭제

  --no-prune=false : 태크가 없는 부모 이미지를 삭제하지 않습니다.

- 예제 : **docker rmi b56d8850aed5**
- Output :

```
Untagged: tomcat:latest
Untagged: tomcat@sha256:d53c2079ea67db92f6d7c39e9450f641610336016fdddef5392c5afd41518e5e
Deleted: sha256:b56d8850aed5468715564198d27b2d05752eb8f63485ade8601651d57bae6067
Deleted: sha256:94e73673f905d62af6ac7a200c04f70c0b57d61eff9116cea18c664cca91045b
Deleted: sha256:5209932c823b5cdd2c17646264a887f87495a3136849cdc7db661c8a66956d54
Deleted: sha256:b5c6976d6644c3b38413f73ca6ccd72bb697b1916e5131bf533bfbb7ec35761f
Deleted: sha256:ac0fb599926db89be8716869d82aac3c6e62ea348773a69ad27c4224e8b86035
Deleted: sha256:b024e7dc232e8df80216d932be7a0e70861c98f5a18d28171b5d305b2e15fb72
Deleted: sha256:a426f9ec9cea304af7e78a6da88aeaecb8df1035bbc519018169da71a58a76a8
Deleted: sha256:79b74215e63df1e2e1b336d179f4cb31221aaf13eb7574bb31bc57265a407da1
Deleted: sha256:9af5ac7f79553c214feae23db6b73b26b676b4f5c49f6d9e5a3993eac01c2395
Deleted: sha256:9fe7a9e4f54169fec192350f6873eac00ec472d6341a2ea918bf5f877ed6941a
Deleted: sha256:ce8168f123378f7e04b085c9672717013d1d28b2aa726361bb132c1c64fe76ac
```

docker images

```
REPOSITORY                                TAG                 IMAGE ID       CREATED         SIZE
sptek-shopping                            v01                 867a401712ef   6 days ago      520MB
sptek-member                              v3                  df557bbe96b3   12 days ago     520MB
mymbrcontainreg.azurecr.io/sptek-member   v3                  df557bbe96b3   12 days ago     520MB
tomcat                                    8.5.46-jdk8-openjdk 8973f493aa0a   4 months ago    508MB
```


SPTek

1.3.6 Docker 컨테이너 실행하기

➢ 컨테이너 실행

- 명령어 : **docker run <옵션> <이미지명, ID>:[태그명] <명령> <매개변수]**

※ 옵션

-d, --detach=false : Deteched모드, 데몬모드이며 백그라운드로 실행

-I, --interactive=false : 표준입력(stdin)을 활성화, Bash 명령어를 입력.

-t, --tty : tty 모드를 사용, Bash 명령어를 입력

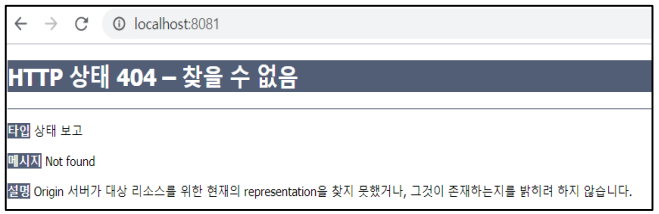
--name : 컨테이너의 이름을 설정

-p, --publish=[] 특정 포트의 연결

- <호스트 포트>:<컨테이너 포트> 예) -p 80:80
- <IP 주소>:<호스트 포트>:<컨테이너 포트> 호스트에 네트워크 인터페이스가 여러 개이거나 IP 주소가 여러 개 일 때 사용.
  예) -p 192.168.0.10:80:80
- <IP 주소>::<컨테이너 포트> 호스트 포트를 설정하지 않으면 호스트의 포트 번호가 무작위로 설정됩니다.
  예) -p 192.168.0.10::80
- <컨테이너 포트> 컨테이너 포트만 설정하면 호스트의 포트 번호가 무작위로 설정됩니다. 예) -p 8

- 예제 : **docker run -d -i -t --name="tomcat-test" -p 8081:8080 tomcat:8**
- Output :

```
C:₩Users₩temp>docker run -d -i -t --name="tomcat-test" -p 8081:8080 tomcat:8
fc665fc94d688279d89e8235131f0ec6d51306cb19e07c46f083445259924d0b
```



1.3.6 Docker 컨테이너 로그보기

➢ 컨테이너 로그

- 명령어 : **docker logs [container name]**
- 예제 : **docker logs tomcat-test**
- Output :

```
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/local/openjdk-8
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
10-Feb-2020 04:03:06.914 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version name:   Apache Tomcat/8.5.50
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:          Dec 7 2019 19:19:46 UTC
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version number: 8.5.50.0
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:               Linux
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:            4.19.76-linuxkit
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:          amd64
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:             /usr/local/openjdk-8/jre
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:           1.8.0_242-b08
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:            Oracle Corporation
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:         /usr/local/tomcat
10-Feb-2020 04:03:06.916 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:         /usr/local/tomcat
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=org.apache.catalina.webresources
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dorg.apache.catalina.security.SecurityListener.UMASK=0027
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.base=/usr/local/tomcat
10-Feb-2020 04:03:06.917 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.home=/usr/local/tomcat

.....
```


SPTek

1.3.6 Docker 컨테이너 명령어 실행

➢ 컨테이너 명령어 실행

- 명령어 : **docker exec <옵션> <이미지명, ID>:[태그명] <명령> <매개변수]**

  ※ 옵션

  -d, --detach=false : Deteched모드, 데몬모드이며 백그라운드로 실행

  -I, --interactive=false : 표준입력(stdin)을 활성화, Bash 명령어를 입력.

  -t, --tty : tty 모드를 사용, Bash 명령어를 입력

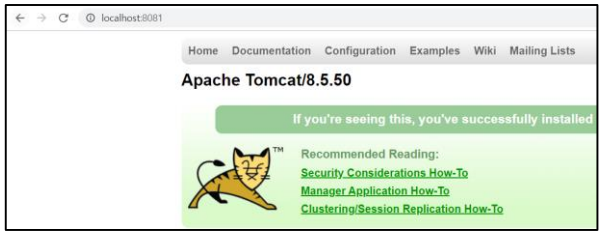
- 예제 : **docker exec -i -t tomcat-test /bin/bash**

- Output :

```
C:₩Users₩temp>docker exec -i -t tomcat-test /bin/bash
root@fc665fc94d68:/usr/local/tomcat# pwd
/usr/local/tomcat/webapps
root@fc665fc94d68:/usr/local/tomcat/webapps#
```

```
docker cp -a ROOT tomcat-test:/usr/local/tomcat/webapps
```

```
root@fc665fc94d68:/usr/local/tomcat# ls
BUILDING.txt     LICENSE  README.md      RUNNING.txt  conf     lib  native-jni-lib  webapps      work
CONTRIBUTING.md  NOTICE   RELEASE-NOTES  bin          include  logs  temp            webapps.dist
root@fc665fc94d68:/usr/local/tomcat# cd webapps
root@fc665fc94d68:/usr/local/tomcat/webapps# ls
ROOT
root@fc665fc94d68:/usr/local/tomcat/webapps#
```



1.3.6 Docker 컨테이너 목록확인

➢ 컨테이너 목록 확인

- 명령어 : **docker ps <옵션>**

※ 옵션

-a, --all=false: 모든 컨테이너를 출력합니다. docker ps 명령은 기본적으로 시작된 컨테이너만 출력합니다.

--before="": 특정 컨테이너가 생성되기 전에 생성된 컨테이너를 출력합니다. 정지된 컨테이너도 포함됩니다.

-f, --filter=[]: 출력 필터를 설정합니다. 예) "exited=0"

-l, --latest=false: 가장 마지막에 생성된 컨테이너를 출력합니다. 정지된 컨테이너도 포함됩니다.

-n=-1: 숫자를 지정하여 최근에 생성된 컨테이너를 일정 개수만 출력합니다. 정지된 컨테이너도 포함됩니다.

--no-trunc=false: 내용이 길어서 생략된 부분을 모두 출력합니다.

-q, --quiet=false: 컨테이너 ID만 출력합니다.

-s, --size=false: 컨테이너에서 변경된 데이터의 크기를 출력합니다.

--since="": 특정 컨테이너가 생성된 후에 생성된 컨테이너를 출력합니다. 정지된 컨테이너도 포함됩니다.

- 예제 : **docker ps -a**
- Output :

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fc665fc94d68	tomcat:8	"catalina.sh run"	21 minutes ago	Up 21 minutes	0.0.0.0:8081->8080/tcp	tomcat-test
9d4716385c9a	sptek-shopping:v01	"catalina.sh run"	6 days ago	Exited (255) 6 days ago	0.0.0.0:8081->8080/tcp	sptek-shopping

1.3.6 Docker 컨테이너 중지하기

➢ 컨테이너 중지

- 명령어 : **docker stop <옵션> container**

  ※ 옵션

  -t, --time: 해당 시간 동안 기다린 후 중지

- 예제 : **docker stop 9d4716385c9a**

- Output :

```
docker stop 9d4716385c9a
9d4716385c9a

docker stop fc665fc94d68
Fc665fc94d68

docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

1.3.6 Docker 컨테이너 삭제하기

➢ 컨테이너 삭제

- 명령어 : **docker rm <옵션> container**

※ 옵션

-f, --force: 강제삭제

-l, --link : 링크 파일 삭제

-v, --volumns : 컨테이너와 관련 있는 volumn 삭제

- 예제 : **docker rm**
- Output :

```
CONTAINER ID   IMAGE              COMMAND            CREATED          STATUS                      PORTS                    NAMES
fc665fc94d68   tomcat:8           "catalina.sh run"  29 minutes ago   Exited (143) 4 minutes ago                           tomcat-test
9d4716385c9a   sptek-shopping:v01 "catalina.sh run"  6 days ago       Exited (255) 6 days ago     0.0.0.0:8081->8080/tcp   sptek-shopping

docker rm fc665fc94d68
Fc665fc94d68

docker container ls -al
CONTAINER ID   IMAGE              COMMAND            CREATED          STATUS                      PORTS                    NAMES
9d4716385c9a   sptek-shopping:v01 "catalina.sh run"  6 days ago       Exited (255) 6 days ago     0.0.0.0:8081->8080/tcp   sptek-shopping
```

1.4.1 Docker 이미지 만들기

➢ Dockerfile 생성 키워드

- FROM
  - 가장 기본적인 커맨드이다. 어떤 이미지를 기반으로 새로운 이미지를 생성할 것인지를 나타낸다.
- RUN
  - RUN 커맨드는 정말 간단하게 생각해서, bash 쉘에서 입력하는것과 동일하다고 생각하면 된다
- ADD
  - ADD 명령어는 build 명령 중간에 호스트의 파일 시스템으로부터 파일을 가져오는 것이다
- CMD / ENTRYPOINT
  - 컨테이너 시작 시, 실행될 명령어를 정하는 커맨드이기에 build로 이미지가 만들어지고, 그 이미지로 컨테이너를 run할 때 효력을 갖는다

```
FROM tomcat:8.5.46-jdk8-openjdk

RUN rm -Rf /usr/local/tomcat/webapps/sptekshopping ## tomcat root 경로 삭제
RUN rm -Rf /usr/local/tomcat/webapps/sptekshopping.war ## tomcat root 경로 삭제
COPY ./sptekmember.war /usr/local/tomcat/webapps/sptekmember.war
```

1.4.2 Docker build 명령어

➢ Build 명령어

- 명령어 : **docker build <옵션> <alias> <dockerfile 경로>**

※ 옵션

-force-rm=false: 이미지 생성에 실패했을 때도 임시 컨테이너를 삭제합니다.

--no-cache=false: 이전 빌드에서 생성된 캐시를 사용하지 않습니다. Docker는 이미지 생성 시간을 줄이기 위해서 Dockerfile의 각 과정을 캐시하는데, 이 캐시를 사용하지 않고 처음부터 다시 이미지를 생성합니다.

-q, --quiet=false: Dockerfile의 RUN이 실행한 출력 결과를 표시하지 않습니다.

--rm=true: 이미지 생성에 성공했을 때 임시 컨테이너를 삭제합니다.

-t, --tag="": 저장소 이름, 이미지 이름, 태그를 설정합니다. <저장소 이름>/<이미지 이름>:<태그> 형식입니다.

hello

hello:0.1

exampleuser/hello

exampleuser/hello:0.1

- 예제 : **docker build -t sptek-member:v3 .**
- Output :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tomcat	8	b56d8850aed5	3 days ago	529MB
sptek-shopping	v01	867a401712ef	6 days ago	520MB
sptek-member	v3	df557bbe96b3	12 days ago	520MB
mymbrcontainreg.azurecr.io/sptek-member	v3	df557bbe96b3	12 days ago	520MB
tomcat	8.5.46-jdk8-openjdk	8973f493aa0a	4 months ago	508MB

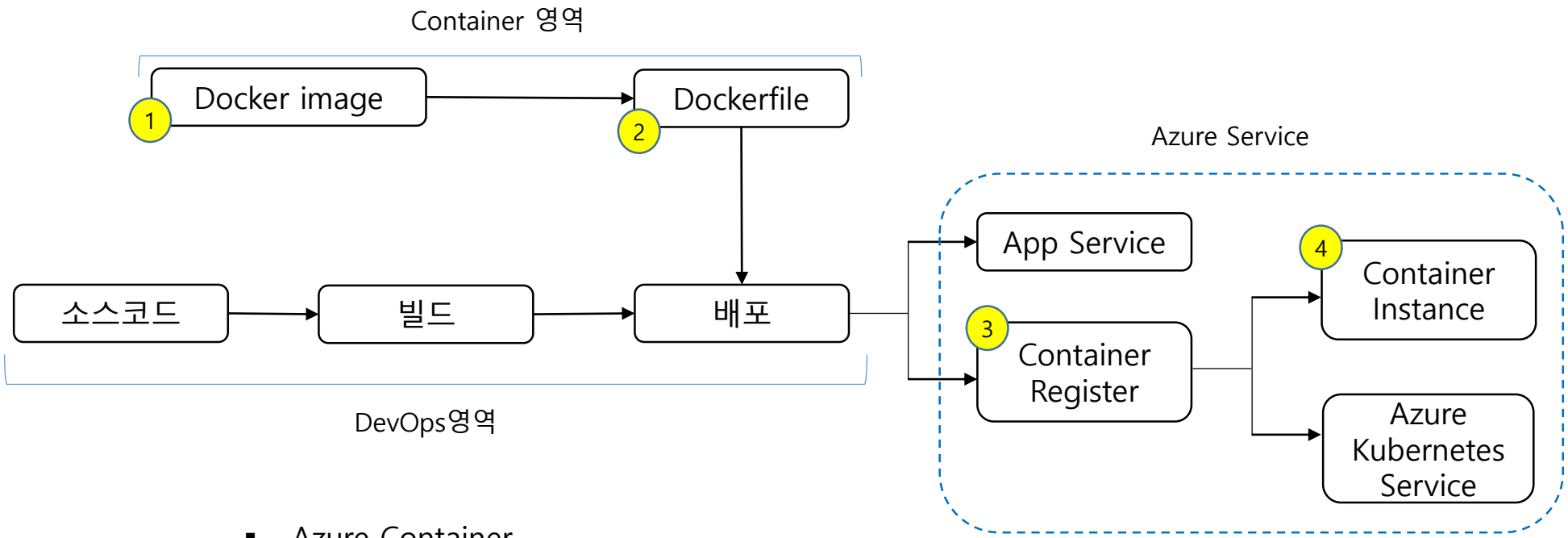
- 예제 : **docker run -d -i -t --name="sptekmember-test" -p 8081:8080 sptek-member:v3**

SPTek

2. Azure Container Register

2.1.1 Azure Container 활용

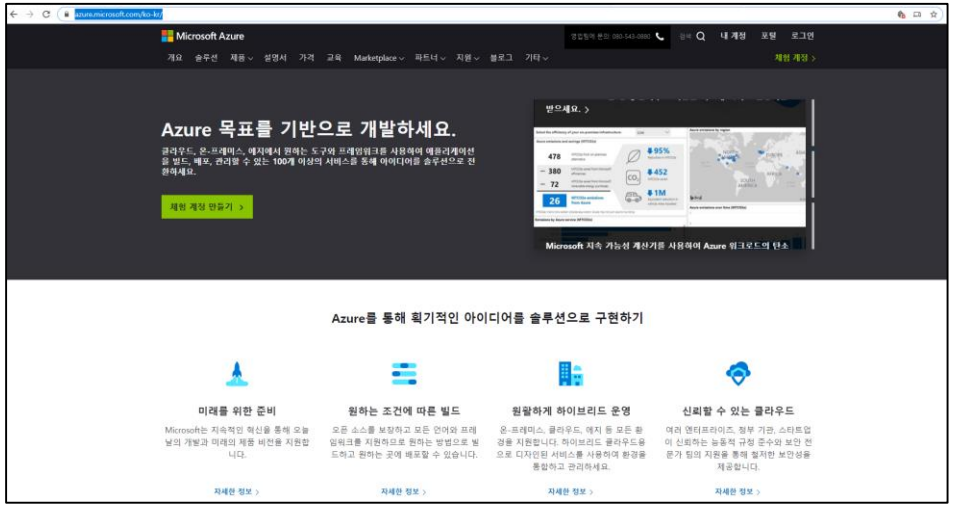
➢ Container 활용



- Azure Container
  - Docker의 Container를 ACR에 등록하고 ACI를 기반으로 서비스
- Azure DevOps & Azure App Service
  - DevOps의 build-pipeline과 release-pipeline를 통해 app service 기반으로 서비스
- Azure DevOps & Azure ACI
  - DevOps의 build-pipeline과 release-pipeline를 통해 ACI를 기반으로 서비스
- Azure DevOps & Azure Kubernetes Service
  - DevOps의 build-pipeline과 release-pipeline를 통해 Azure Kubernetes Service를 기반으로 서비스

2.1.2 Azure 체험계정

➢ 체험계정
https://azure.microsoft.com/ko-kr/



➢ 참고 URL

https://mixedcode.com/Article/Index?aidx=7730

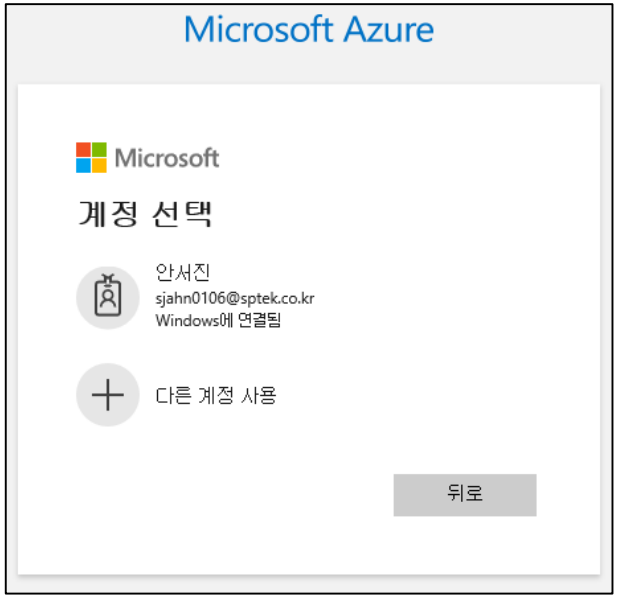
2.1.2 Azure CLI 설치

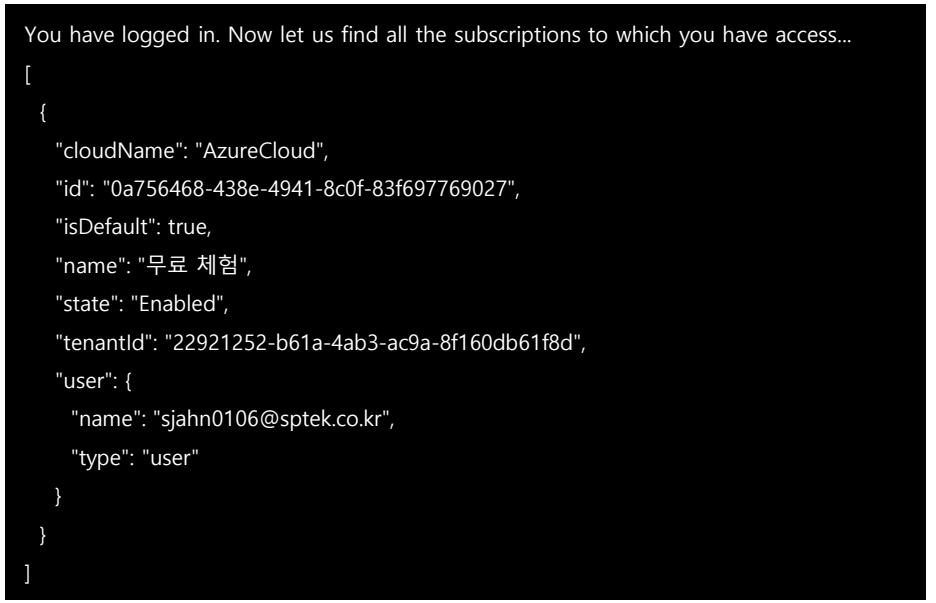
➢ Azure CLI 설치

- Azure CLI의 현재 버전은 __2.1.0__입니다
- MSI 배포 파일은 Windows에서 Azure CLI를 설치하거나 업데이트하는 데 사용됩니다.
- MSI 설치 프로그램을 사용하기 전에 현재 버전을 제거 할 필요는 없습니다.
- 설치 관리자가 컴퓨터를 변경해도 되는지 물어보면 "예" 상자를 클릭합니다.

  https://docs.microsoft.com/ko-kr/cli/azure/install-azure-cli-windows?view=azure-cli-latest
- Login 명령어 실행

  C:₩temp₩az login
- 자격증명





2.2.1 Azure Resource Group

➢ Resource Group

- Resource group is a container that holds related resources for an Azure solution.
- The resource group can include all the resources for the solution, or only those resources that you want to manage as a group.
- Azure Service를 사용하기 위해서는 반드시 Resource Group을 정의 해야 함

➢ Resource Group CLI 명령어

- List resource groups : **az group list**


```
[
  {
    "id": "/subscriptions/0a756468-438e-4941-8c0f-83f697769027/resourceGroups/myResourceGroup",
    "location": "eastus",
    "managedBy": null,
    "name": "myResourceGroup",
    "properties": {
      "provisioningState": "Succeeded"
    },
    "tags": null,
    "type": "Microsoft.Resources/resourceGroups"
  },
  {
    "id": "/subscriptions/0a756468-438e-4941-8c0f-83f697769027/resourceGroups/myResourceGrp",
    "location": "eastus",
    "managedBy": null,
    "name": "myResourceGrp",
    "properties": {
      "provisioningState": "Succeeded"
    },
    "tags": null,
    "type": "Microsoft.Resources/resourceGroups"
  }

,.....................
```

2.2.1 Azure Resource Group

➢ Resource Group CLI 명령어

- create resource groups : **az group create**

  ex) **az group create --name myResourceGrp --location eastus**

  --name : resource group 명

  -- location : region 정보

- delete resource groups : **az group delete**

  ex) **az group delete --name $resourceGroupName**

- delete resource groups : **az group show**

  az group show --name $resourceGroupName

2.2.2 ACR의 정의 및 생성

➢ Azure Container Register 정의

- 모든 OCI 아티팩트에 대한 지원이 제공되는 Docker 및 OCI(Open Container Initiative) 이미지 레지스트리
- Docker Registry 2.0 오픈 소스에 기반한 관리형의 프라이빗 Docker 레지스트리 서비스
- Azure 컨테이너 레지스트리를 만들고 유지 관리하여 프라이빗 Docker 컨테이너 이미지 및 관련 아티팩트를 저장하고 관리합니다.

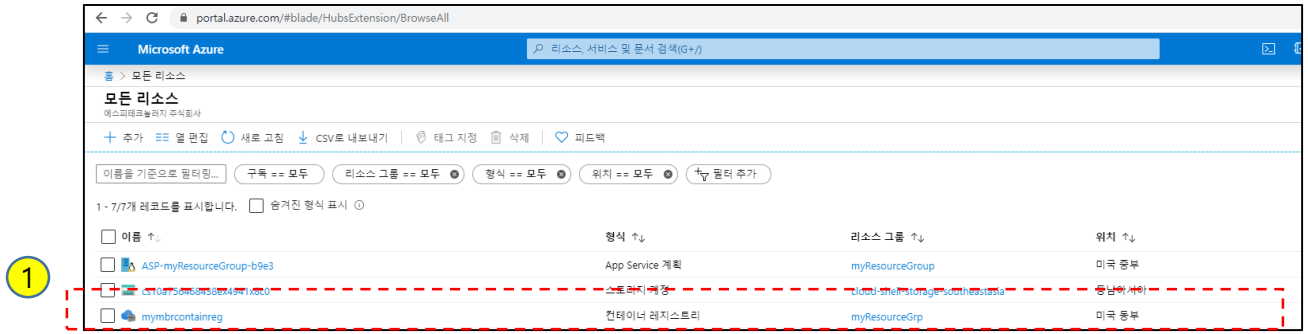
➢ ACR 생성

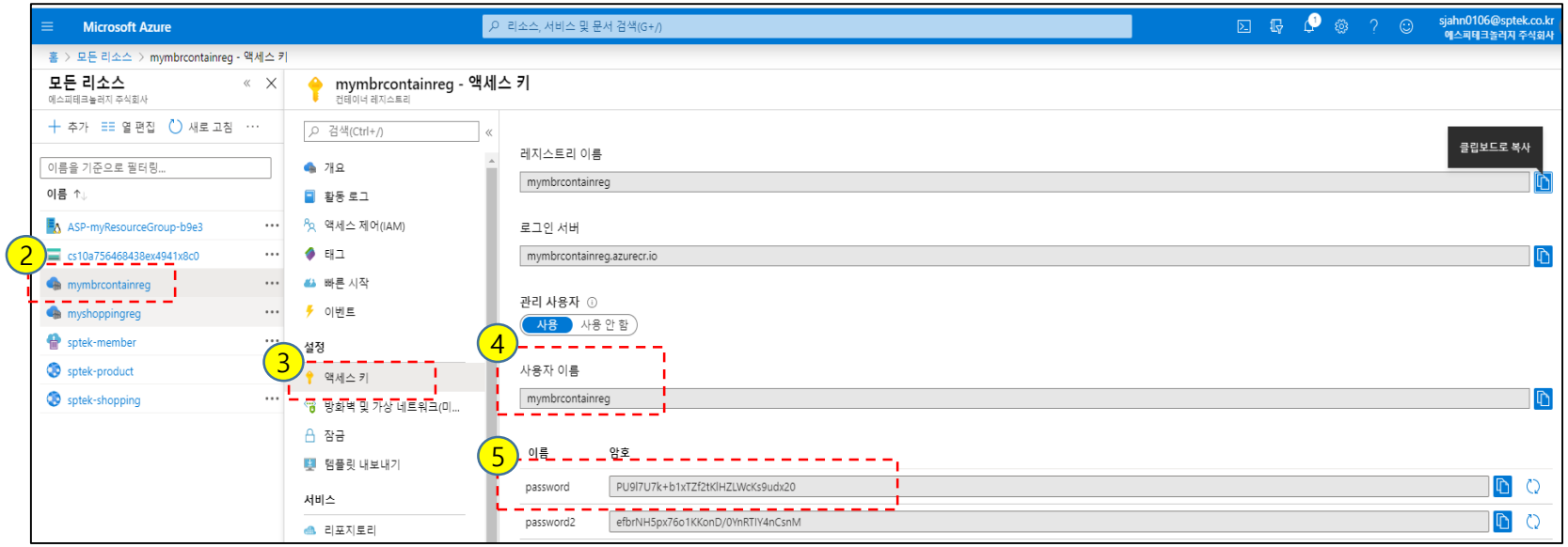
- -- name은 반드시 소문자로 작성
- 명령어

az acr create --resource-group myResourceGrp --name mymbrcontainreg --sku Basic

2.2.2 ACR 엑세스 키 확인

➢ Azure Portal 확인





2.3.1 로그인 서버 정보 확인

➢ ACR 선행 조건

- Azure 구독 내에서 컨테이너 레지스트리를 만듭니다.
- Docker CLI - 또한 Docker가 로컬에 설치되어 있어야 합니다

➢ ACR의 로그인

- 명령어 : az acr login --name [alias]

  ex) az acr login --name mymbrcontainreg

➢ ACR의 로그인 서버 정보

- 명령어 : az acr show --name <acrName> --query loginServer --output table

ex) az acr show --name mymbrcontainreg --query loginServer --output table

```
Result
------------------------
mymbrcontainreg.azurecr.io
```

2.3.1 이미지 tag

➢ Docker 이미지의 tag정보 설정

- tag 정보에 로그인 서버 정보를 설정
- 반드시 acr의 로그인 정보 확인이 필요
- 명령어 : docker tag aci-tutorial-app <acrLoginServer>/aci-tutorial-app:v1

p.36 azure-portal 로그인서버

ex) docker tag sptek-member:v3 mymbrcontainreg.azurecr.io/sptek-member:v3

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tomcat	8	b56d8850aed5	3 days ago	529MB
sptek-shopping	v01	867a401712ef	7 days ago	520MB
mymbrcontainreg.azurecr.io/sptek-member v3	df557bbe96b3	12 days ago	520MB	
sptek-member	v3	df557bbe96b3	12 days ago	520MB
tomcat 8.5.46-jdk8-openjdk	8973f493aa0a	4 months ago	508MB	

2.3.1 이미지 push

➢ Docker 이미지의 push

- 명령어 : docker push mymbrcontainreg.azurecr.io/sptek-member:v3
- push 지연 시간이 있음

➢ ACR push 확인

- 명령어 : az acr repository list --name mymbrcontainreg --output table

```
Result
------------
sptek-member
```

- 명령어 :

az acr repository show-tags --name mymbrcontainreg --repository sptek-member --output table

```
Result
--------
v3
```

3. Azure Container Instance

3.1.1 Azure Container Instance 정의

➢ ACI의 정의

- 어떠한 가상 머신도 관리하지 않고 또 더 높은 수준의 서비스를 채택하지 않고도 Azure에서 컨테이너를 실행하는 가장 빠르고 간단한 방법
- 간단한 애플리케이션, 작업 자동화 및 빌드 작업 등 격리된 컨테이너에서 작동할 수 있는 모든 시나리오에 적합한 솔루션

3.1.1 Azure Container Instance 생성

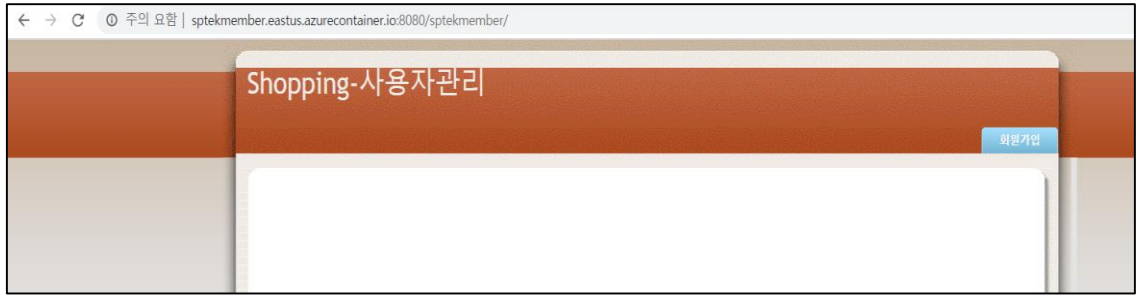
➢ ACI 생성

- --name : The container name must contain no more than 63 characters and must match the regex '[a-z0-9]([-a-z0-9]*[a-z0-9])?' (e.g. 'my-name').
- --dns-name-label : 접속 url prefix 명
- 명령어 : **az container create --resource-group myResourceGroup --name aci-tutorial-app --image <acrLoginServer>/aci-tutorial-app:v1 --cpu 1 --memory 1 --registry-login-server <acrLoginServer> --registry-username <ACR 액세스 키 이름> --registry-password <ACR 액세스 키 암호> --dns-name-label <aciDnsLabel> --ports 80**

ex) az container create --resource-group myResourceGrp --name sptek-member --image mymbrcontainreg.azurecr.io/sptek-member:v3 --cpu 1 --memory 1 --registry-login-server mymbrcontainreg.azurecr.io --registry-username mymbrcontainreg --registry-password PU9l7U7k+b1xTZf2tKlHZLWcKs9udx20 --dns-name-label sptekmember --ports 8080

- 서비스 실행 확인

http://sptekmember.eastus.azurecontainer.io:8080/sptekmember/



❖ 참고문헌 및 자료

- 정보문화사
  https://m.post.naver.com/viewer/postView.nhn?volumeNo=16678855&memberNo=15488377&vType=VERTICAL
- Subicura
  https://subicura.com/2017/01/19/docker-guide-for-beginners-1.html#%EB%8F%84%EC%BB%A4%EB%9E%80
  https://subicura.com/2017/01/19/docker-guide-for-beginners-2.html
  https://subicura.com/2017/02/10/docker-guide-for-beginners-create-image-and-deploy.html
- steemit
  https://steemit.com/kr/@mystarlight/docker
- Docker docs
  https://docs.docker.com/docker-for-windows/docker-toolbox/

Thank you

Mobile & Convergence Leading Company SPTek