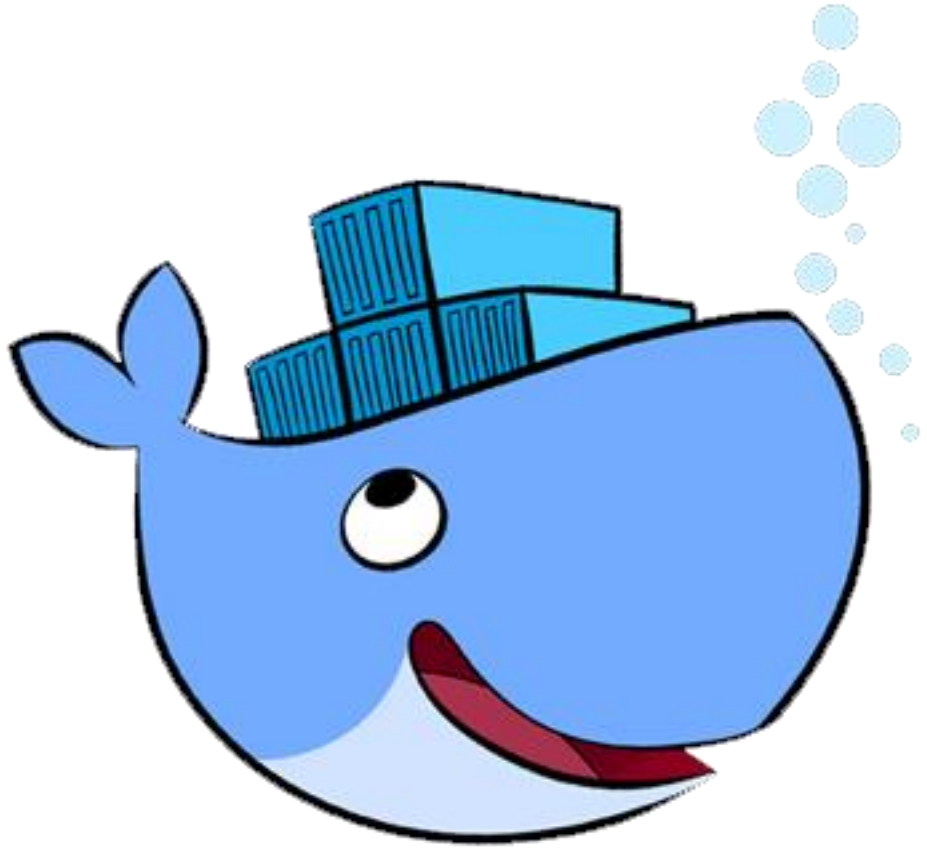
Docker? ARM! 되고 말고!

NHN PAYCO 서버기술개발1팀
유재은

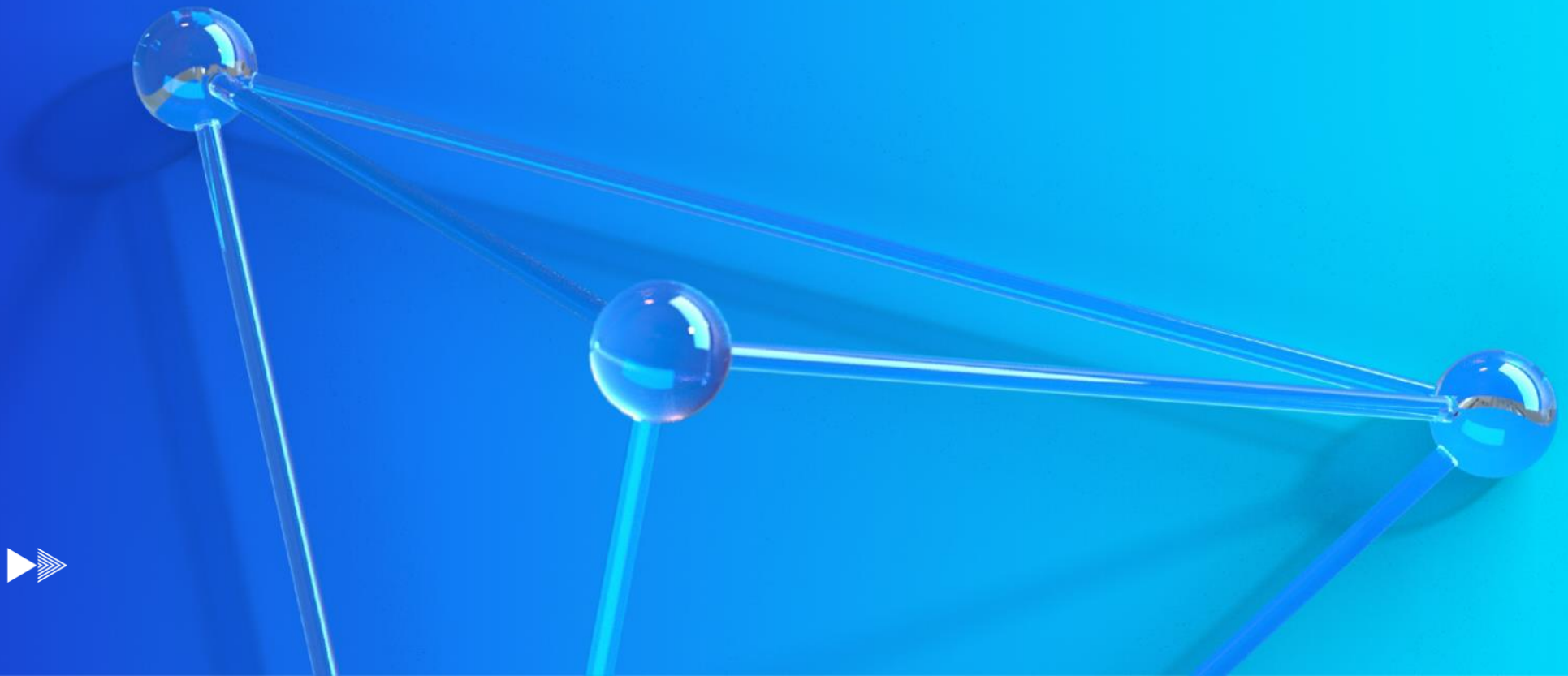
NHN FORWARD

다룰 내용(feat. 의식의 흐름)

1. 뭘 하려고 했나?
2. 이게 왜 안되지? 이게 왜 되지?
3. Multi-Arch 이미지를 빌드하는 방법들
4. BuildX를 이용하여 훨씬 더 쉽게!
5. 후기



뭘 하려고 했나?



NHN FORWARD

MSA로 넘어가는 아름다운 이야기

NHN FORWARD

Tomcat 깔고!

Nginx 깔고!

모니터링 툴 깔고!

JDK 깔고!

스크립트 깔고!

Apache 깔고!

뭐 또 깔고!

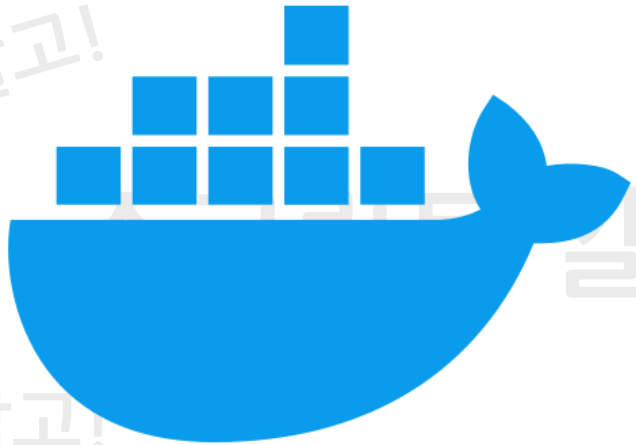
서버세팅 하고

ACL 등록하고

배포 시나리오 짜고

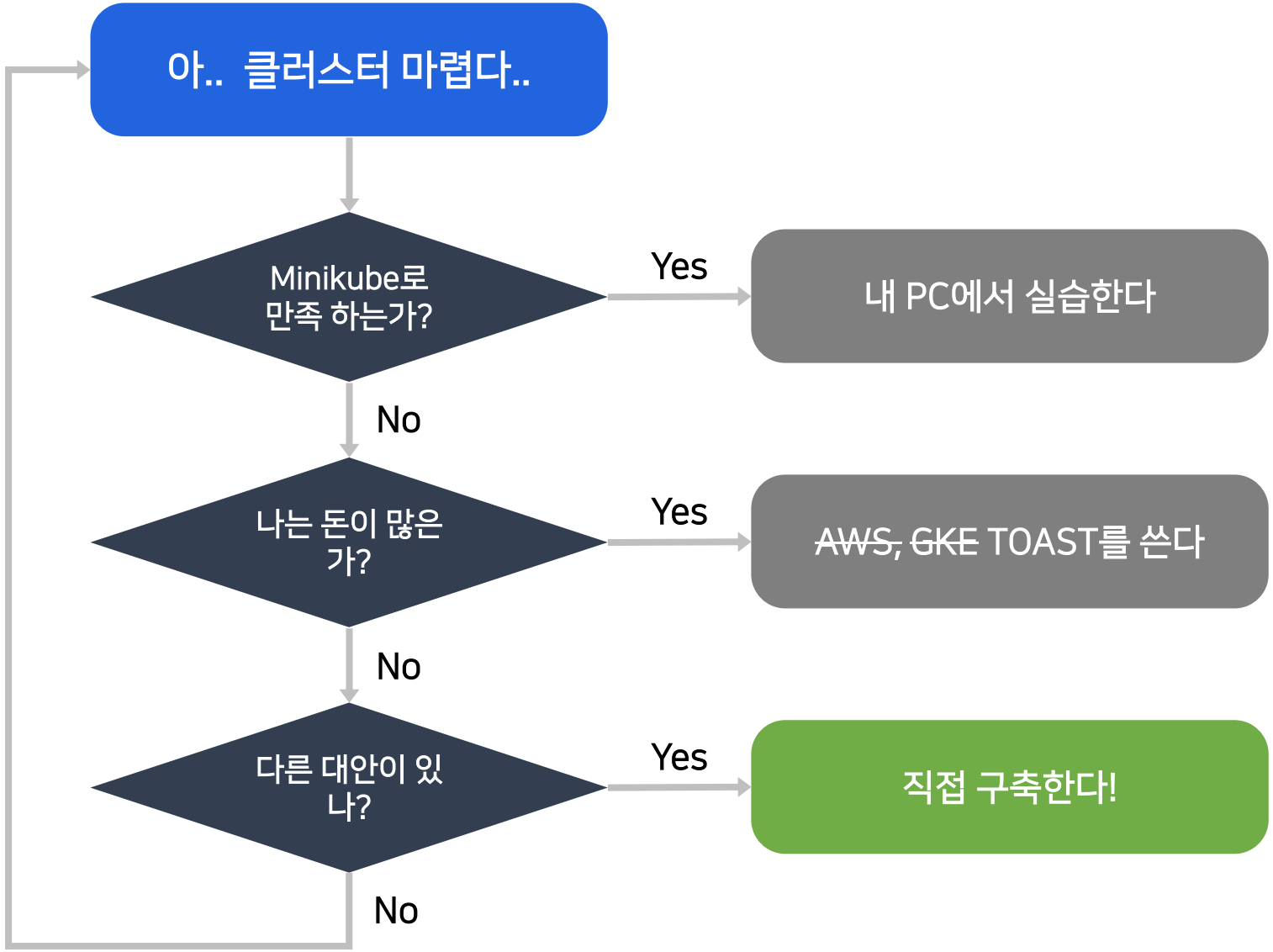
Kubernetes와 Docker로 편하게 해보자

NHN FORWARD



Tomcat 깔고!
Nginx 깔고!
모니터링 툴
JDK 깔고!
뭐 또 깔고!
서버세팅 하고
ACL 등록하고
배포 시나리오 짜고

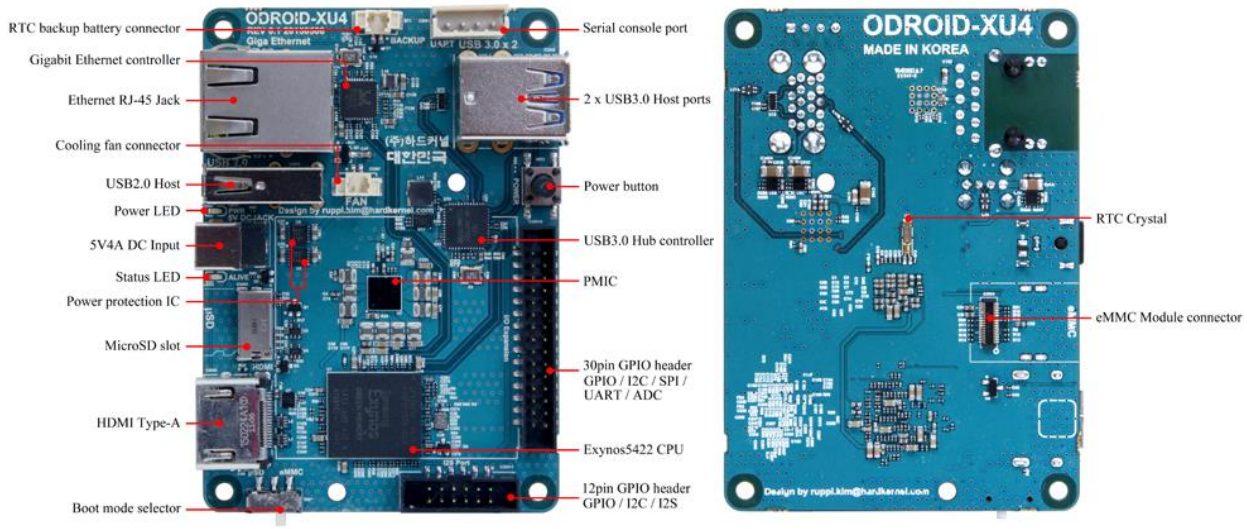
멀티 노드 클러스터는 사드세요..... 제발



준비물

SBC(Single Board Computer)

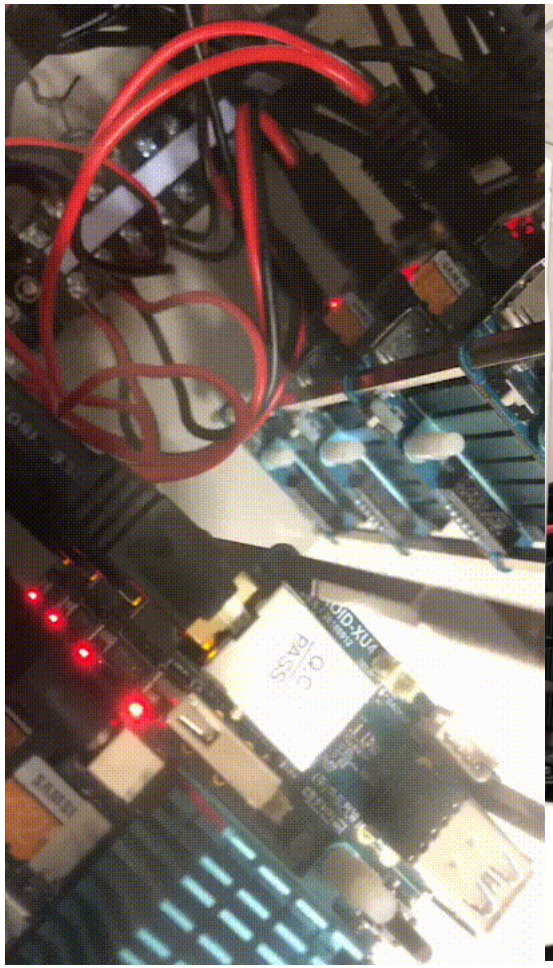
- 하나의 기판에 CPU, 메모리, I/O, OS까지 모두 갖춘 컴퓨터
- 대체로 Linux 기반 OS 사용
- 실제 서버 장비에 비해 저전력 저비용
- Odroid, RaspberryPi 등

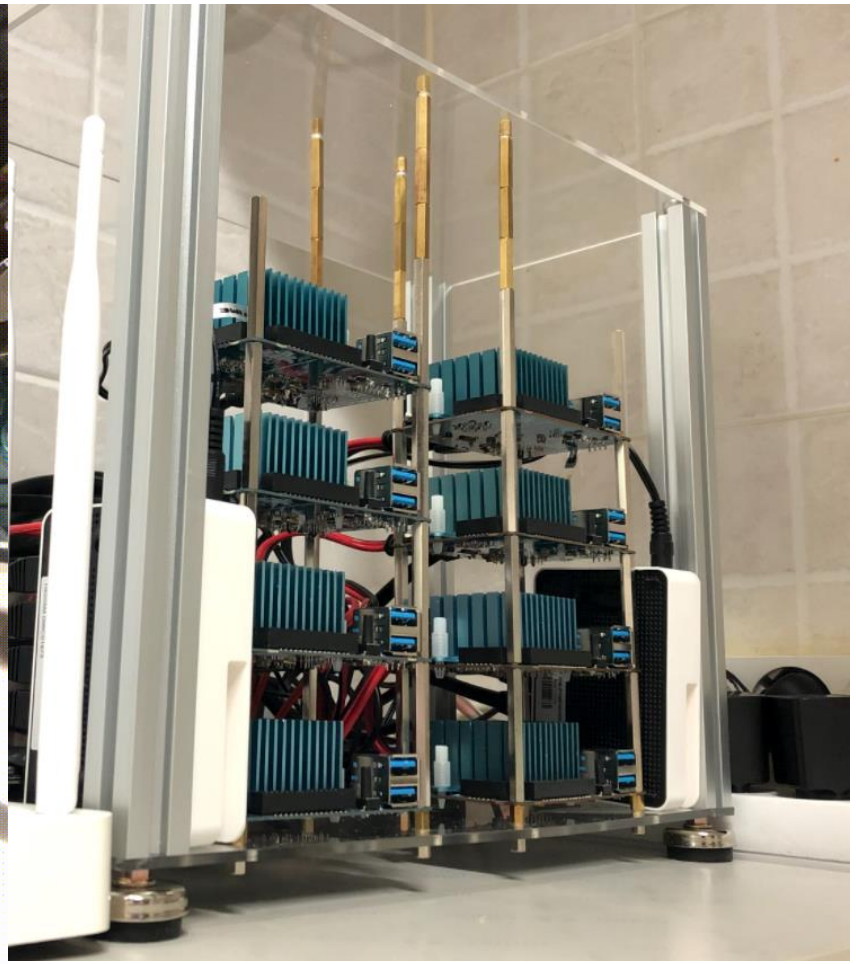


스카이넷 프로젝트

프로젝트 목표

- 급 만들고 싶은 게 생겼을 때!
  간단히 테스트 해보고 싶을 때 !
  과금 부담 없이!
  활용할 수 있는!
  Kubernetes 클러스터를 구축한다.





Docker 실행 테스트 @PC


```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
...
```

Docker 실행 테스트 @PC


```
$ docker run jaeeunyoo/sample-py-app
 * Serving Flask app "sample-webapp" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Docker 실행 테스트 @SBC


```
jaeeunyoo@skynet-201:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm32v7)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
...
```

Docker 실행 테스트 @SBC

NHN FORWARD

```
jaeeunyoo@skynet-201:~$ docker run jaeeunyoo/sample-py-app
standard_init_linux.go:211: exec user process caused "exec format error"
```

테스트 결과

PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
정상	정상	정상

hello-world 이미지

빌드 \ 실행	PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
PC(Mac OS X)	정상	정상	에러
PC(Windows)	정상	정상	에러
SBC(Ubuntu)	정상	정상	정상

sample-py-app 이미지

테스트 결과

PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
정상	정상	정상

hello-world 이미지

빌드 \ 실행	PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
PC(Mac OS X)	정상	정상	에러
PC(Windows)	정상	정상	에러
SBC(Ubuntu)	정상	정상	정상

sample-py-app 이미지

테스트 결과

PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
정상	정상	정상

hello-world 이미지

빌드 \ 실행	PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
PC(Mac OS X)	정상	정상	에러
PC(Windows)	정상	정상	에러
SBC(Ubuntu)	정상	정상	정상

sample-py-app 이미지

테스트 결과

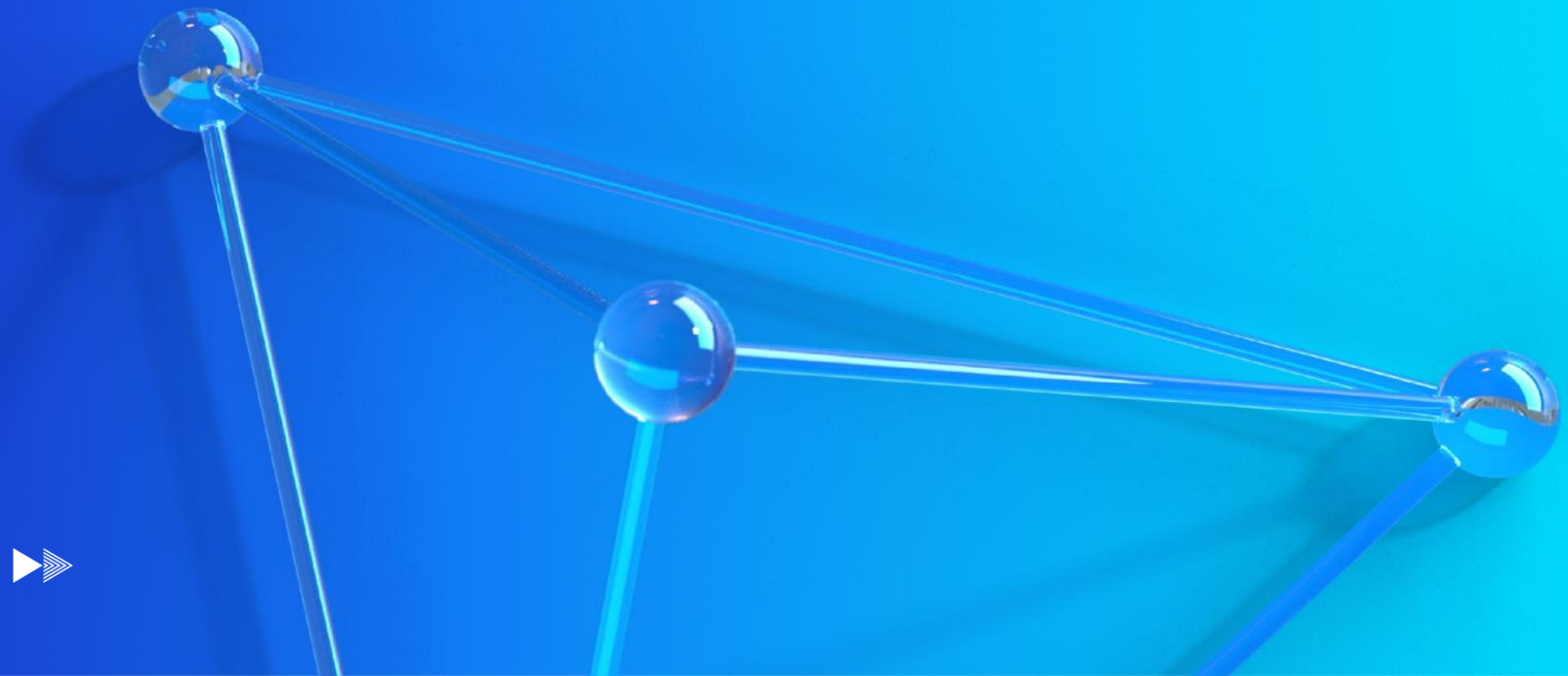
PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
정상	정상	정상

hello-world 이미지

빌드 \ 실행	PC(Mac OS X)	PC(Windows)	SBC(Ubuntu)
PC(Mac OS X)	정상	정상	에러
PC(Windows)	정상	정상	에러
SBC(Ubuntu)	정상	정상	정상

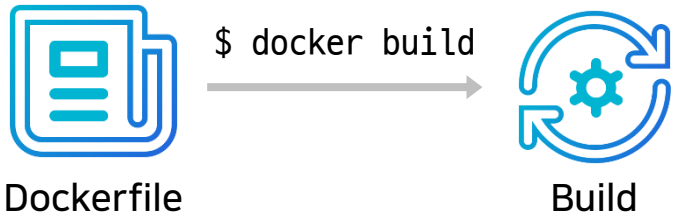
sample-py-app 이미지

이게 왜 안되지? 이게 왜 되지?

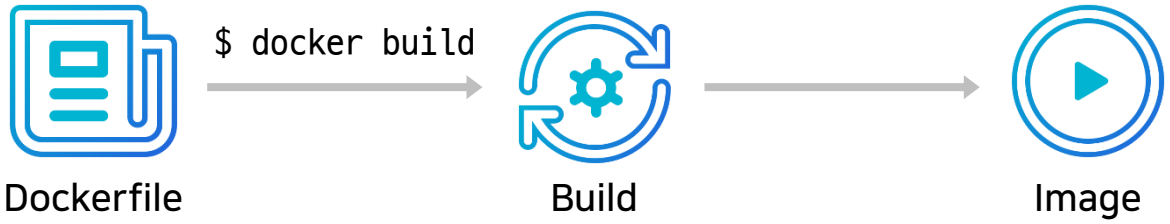


NHN FORWARD

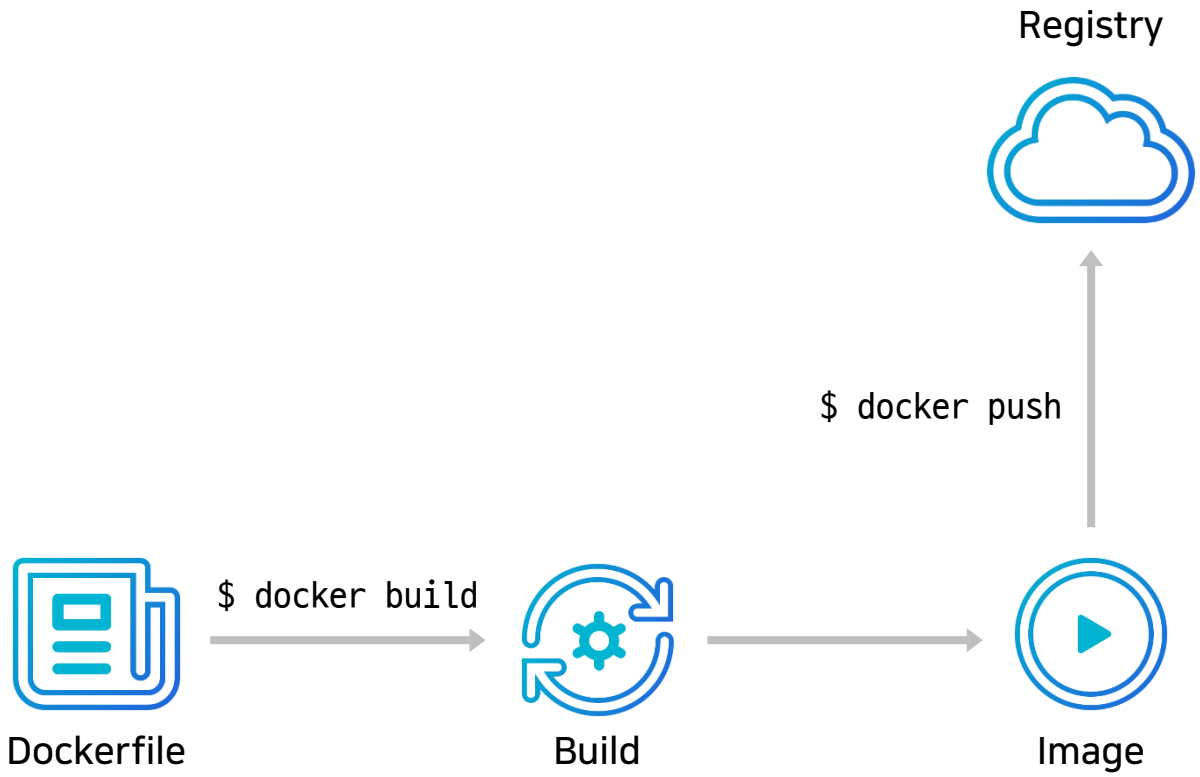
PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유



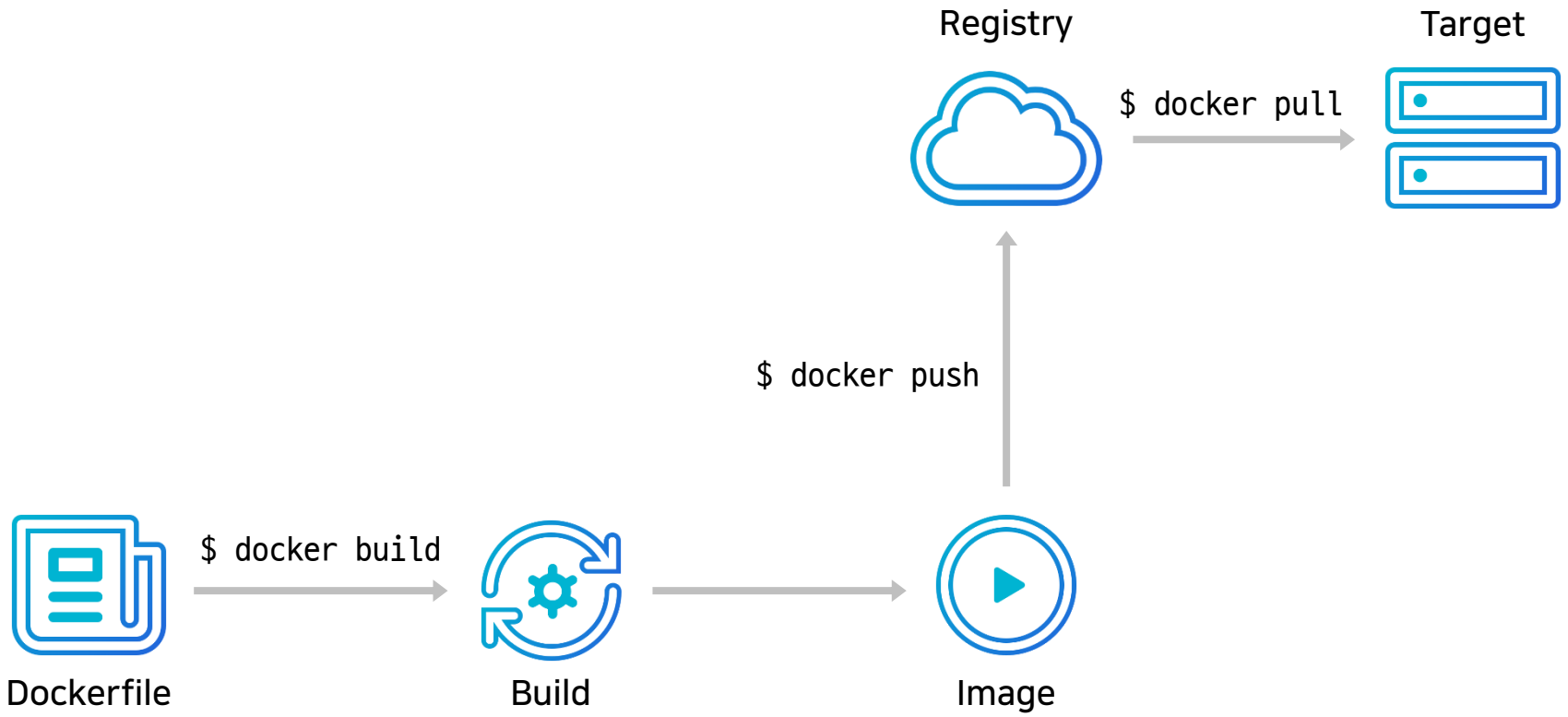
PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유



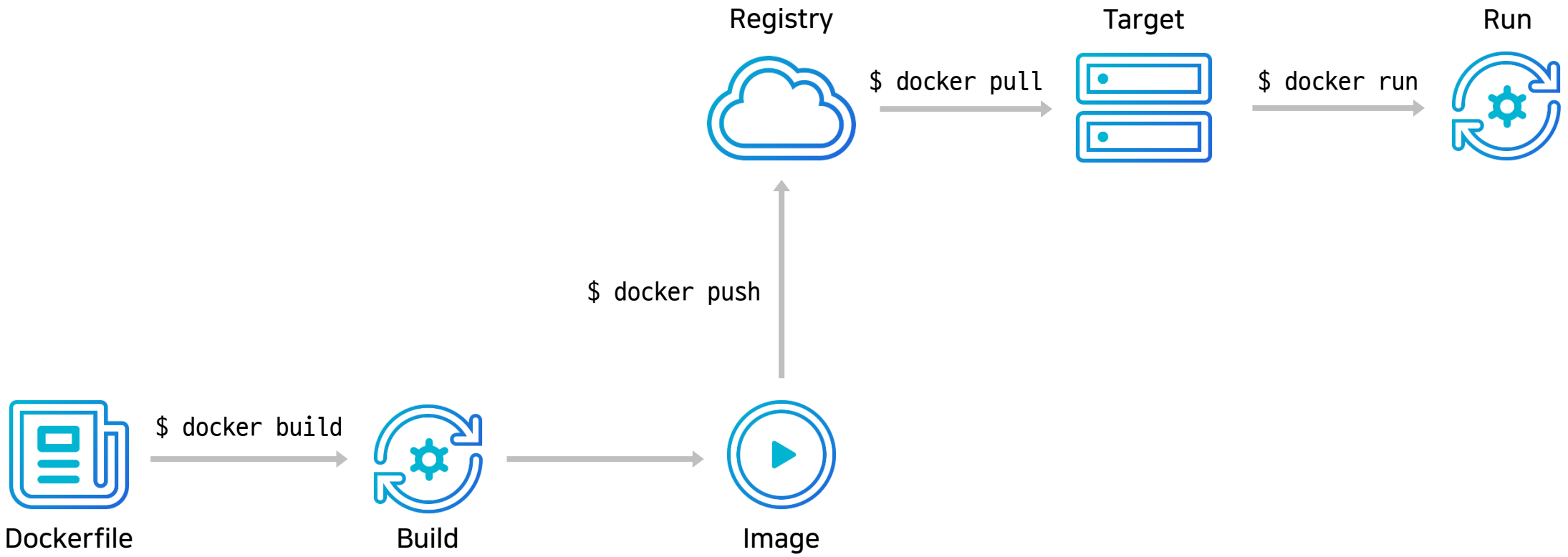
PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유



PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유

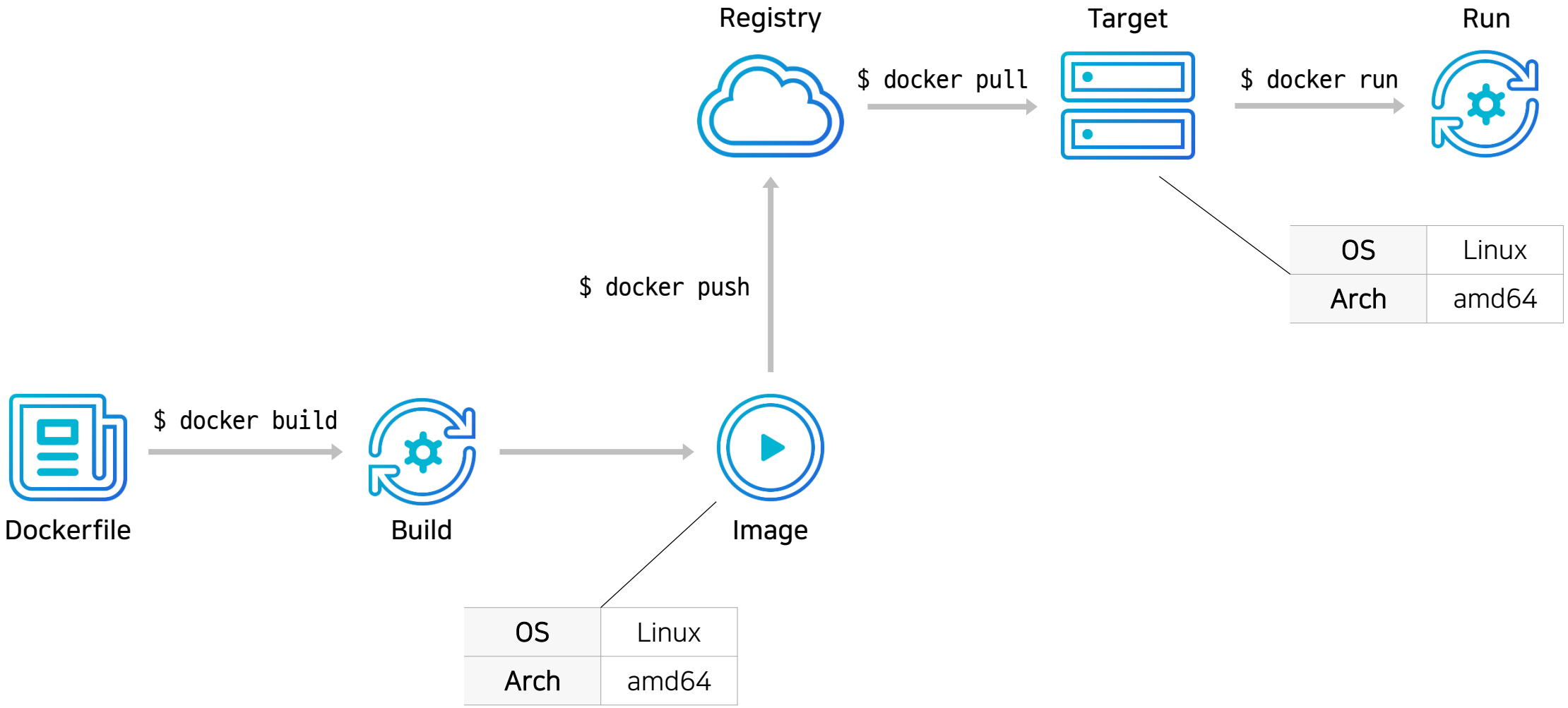


PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유



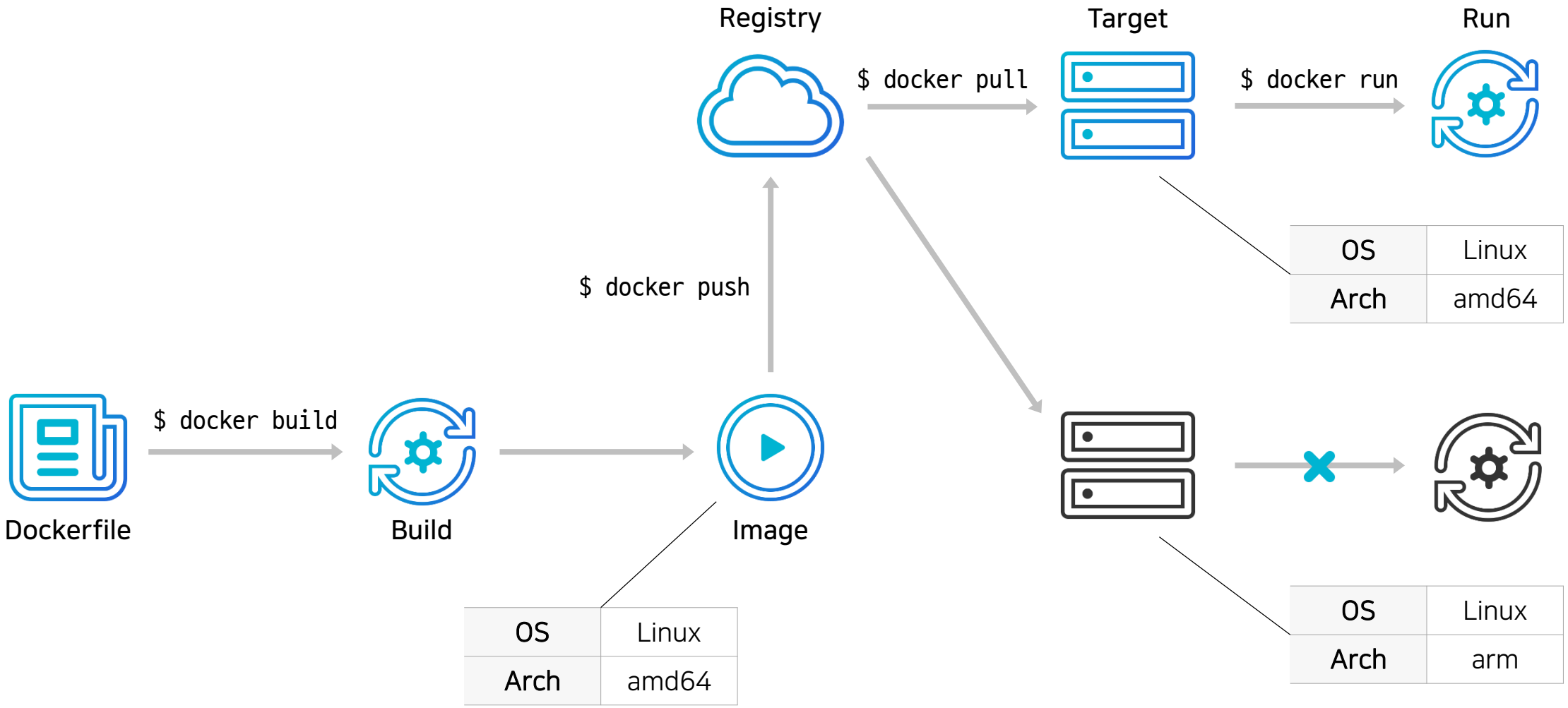
PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유

NHN FORWARD



PC에서 빌드한 이미지가 SBC에서 실행되지 않았던 이유

NHN FORWARD

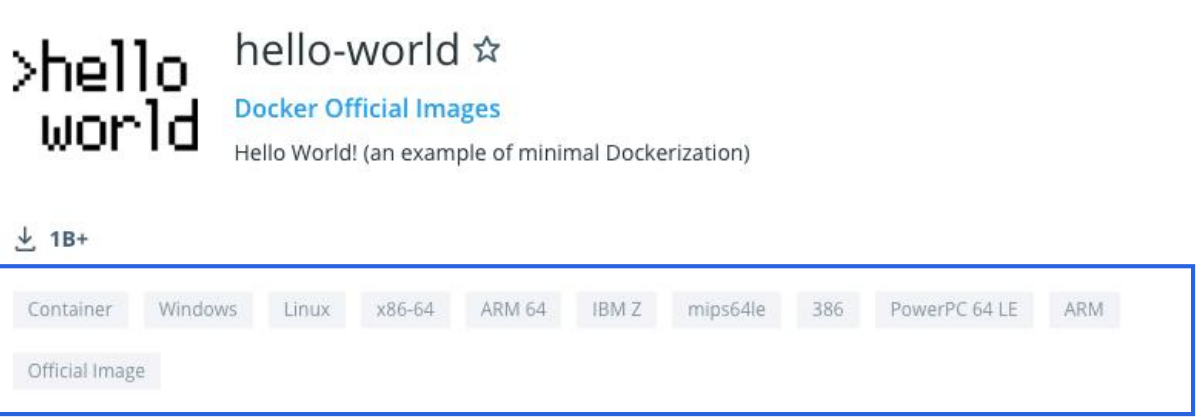


hello-world 이미지가 실행된 이유


```
$ docker run hello-world
...

 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
...

jaeeunyoo@skynet-201:~$ hello-world
...

 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm32v7)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
```

hello-world 이미지가 실행된 이유

NHN FORWARD



Description Reviews Tags

Filter Tags

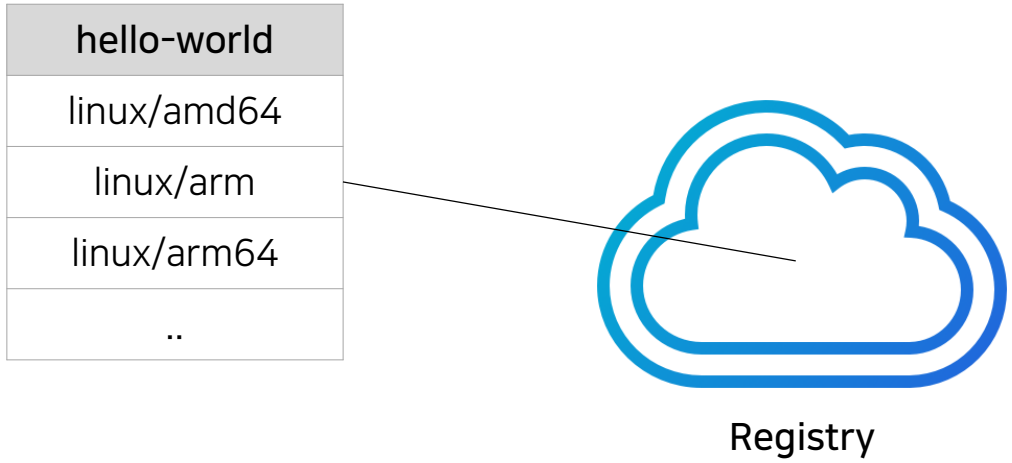
Sort by Latest

TAG
latest
Last pushed a month ago by doijanky

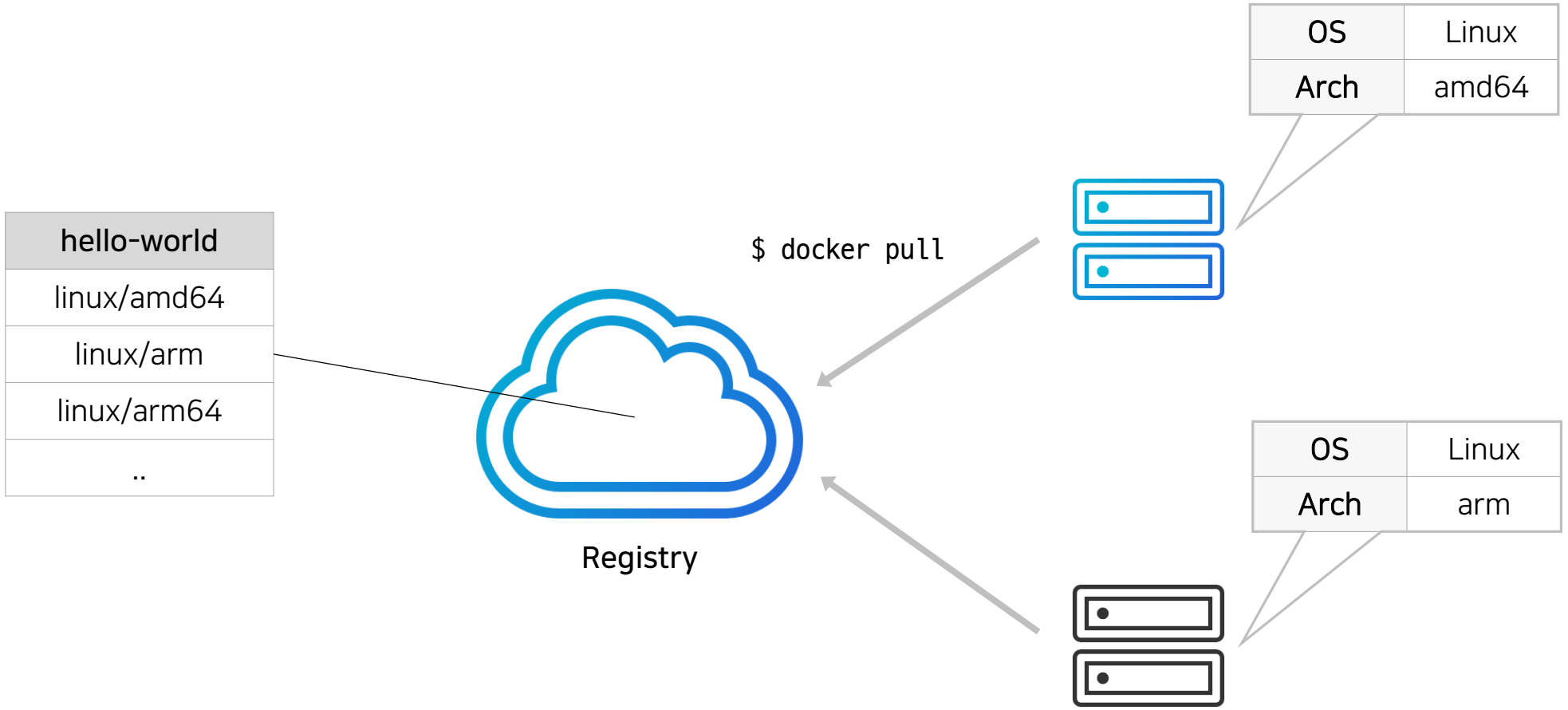
docker pull hello-world:latest

DIGEST	OS/ARCH	COMPRESSED SIZE
ebf526c198a1	linux/386	2.71 KB
73de1a34e3ed	windows/amd64	96.52 MB
90659bf80b44	linux/amd64	2.47 KB
e5785cb0c62c	linux/arm/v5	3.6 KB
50b8560ad574	linux/arm/v7	3.01 KB
963612c5503f	linux/arm64/v8	3.29 KB
88b2e00179bd	linux/mips64le	3.93 KB
bb7ab0fa94fd	linux/ppc64le	3.85 KB
e49abad529e5	linux/s390x	3.23 KB

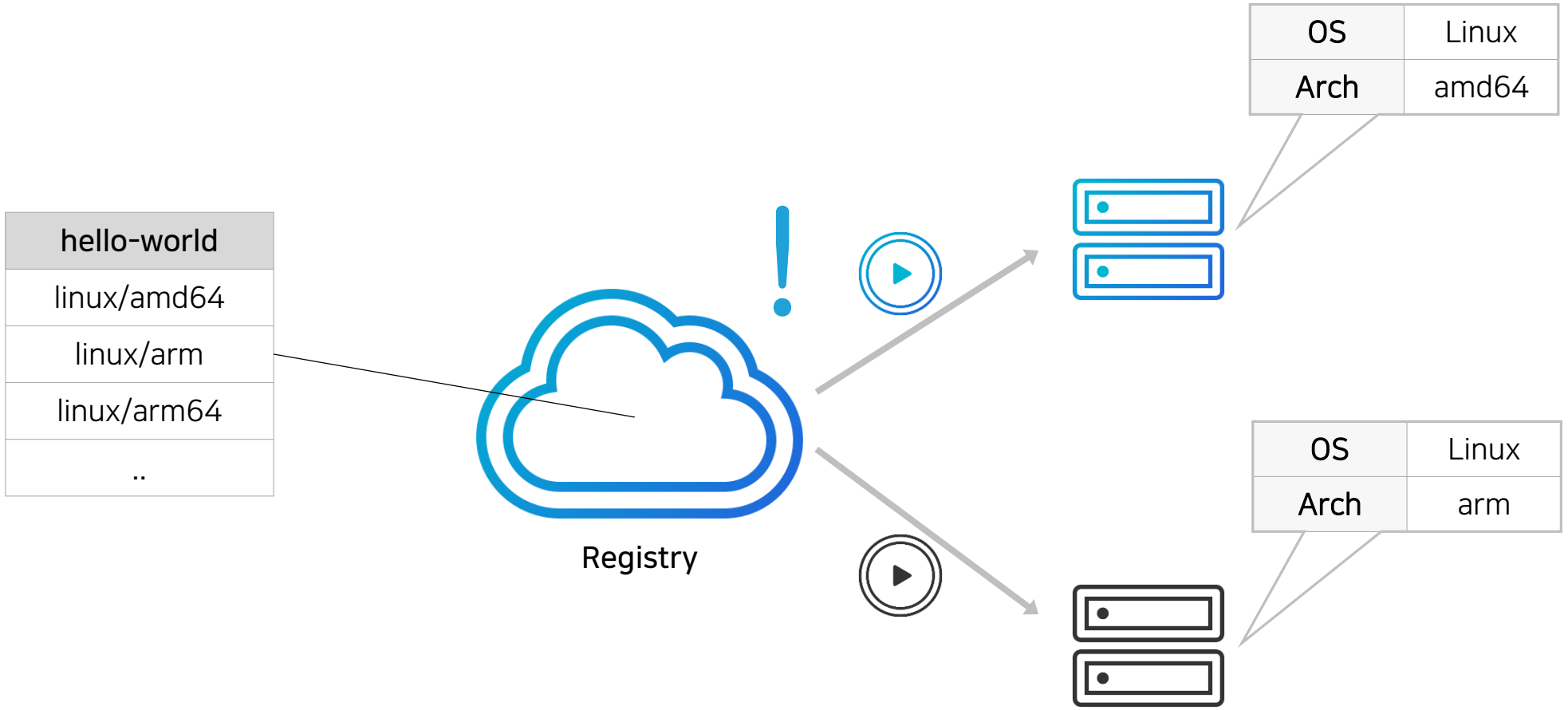
hello-world 이미지가 실행된 이유



hello-world 이미지가 실행된 이유



hello-world 이미지가 실행된 이유



ARM에서 빌드한 이미지가 PC에서 실행된 이유

NHN FORWARD

frontiersofme/from-arm ☆

By frontiersofme • Updated a day ago

Container

Manage Repository

Pulls 2

Overview Tags

Filter Tags

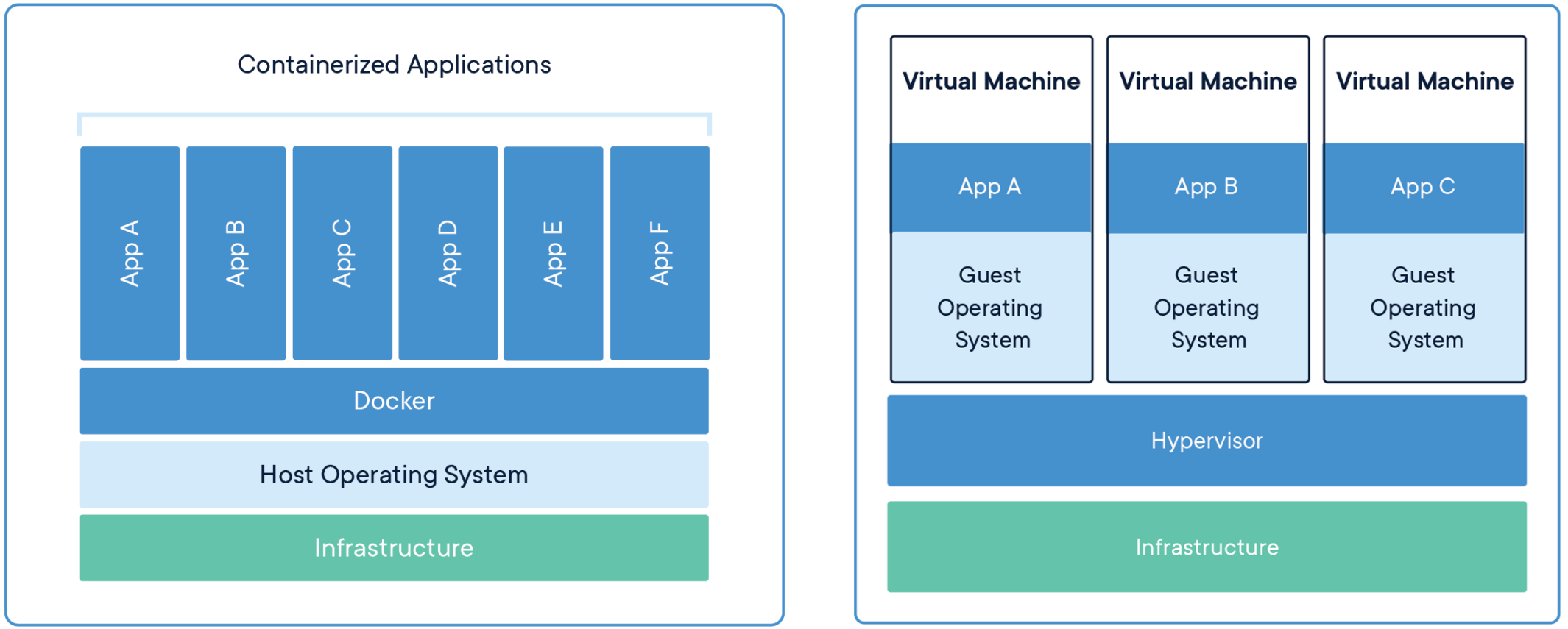
Sort by Latest

TAG
latest
Last pushed a day ago by frontiersofme

docker pull frontiersofme/from-arm:late

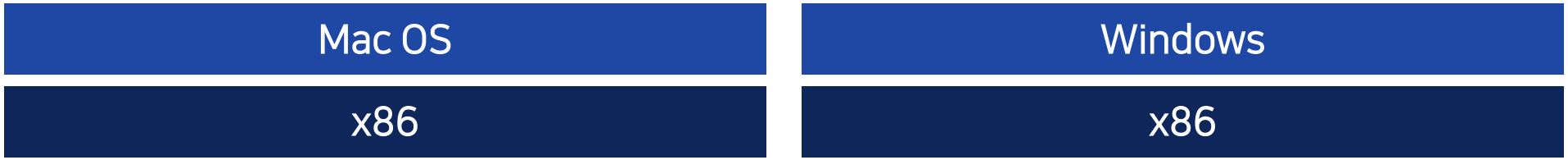
DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
f8729b5be8fe	linux/arm	a day ago	302.07 MB

ARM에서 빌드한 이미지가 PC에서 실행된 이유



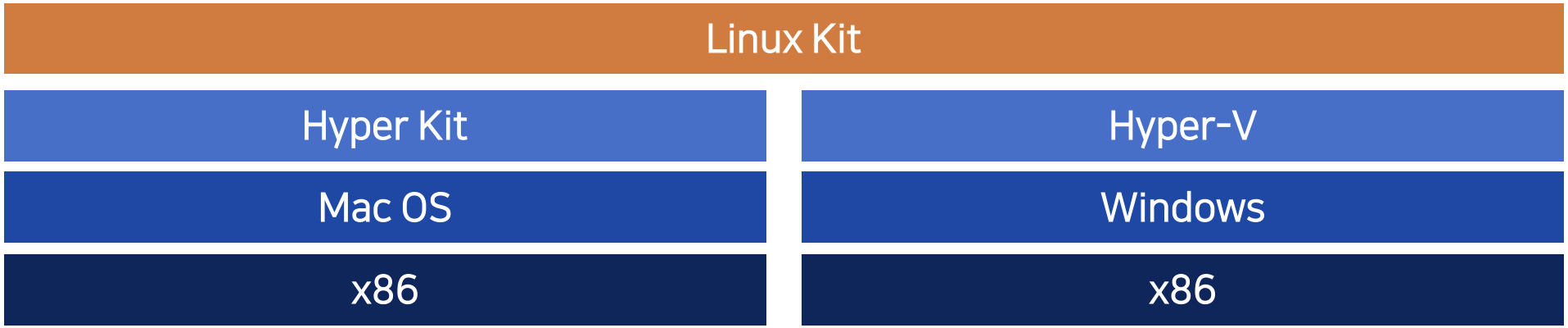
흔한 Docker와 VM 차이.jpg

ARM에서 빌드한 이미지가 PC에서 실행된 이유



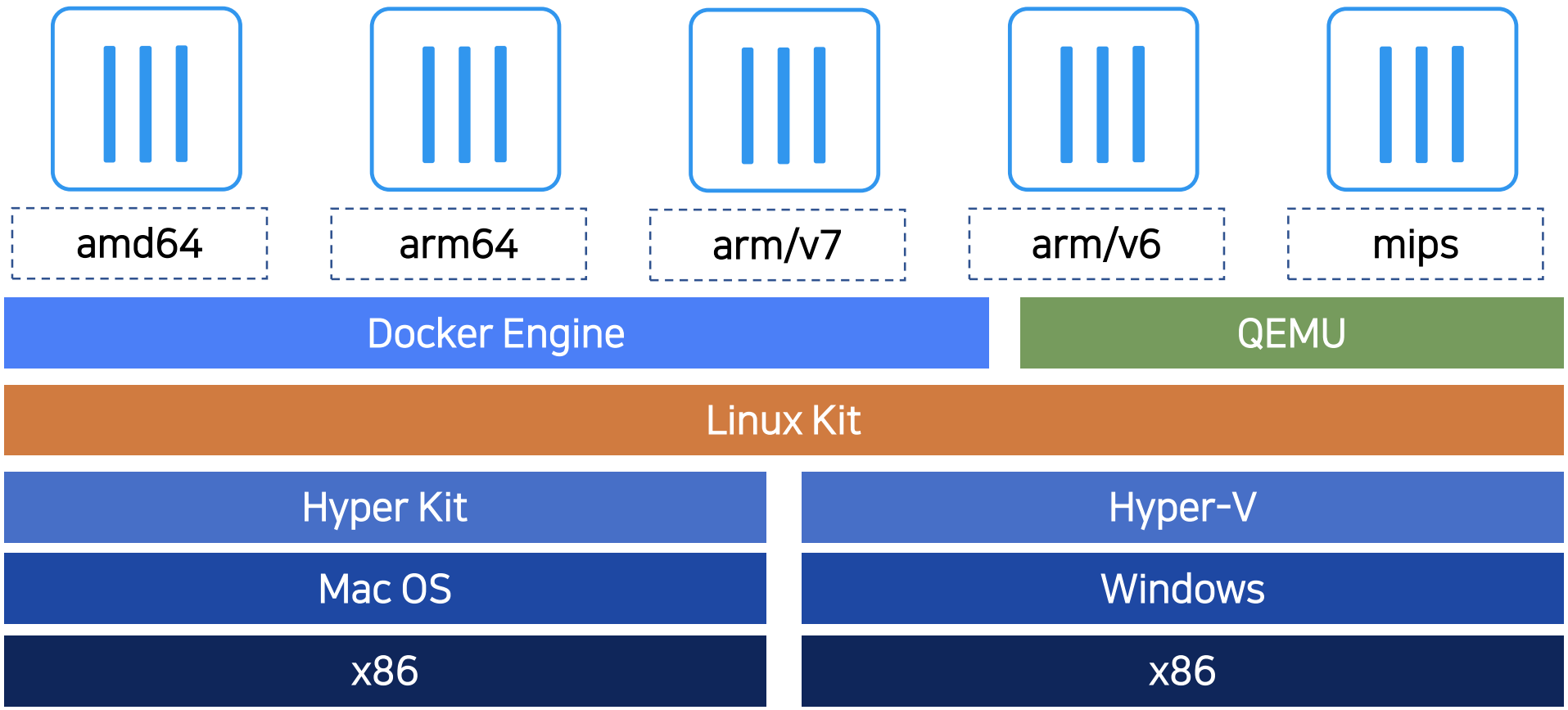
최신 Docker Desktop의 구조

ARM에서 빌드한 이미지가 PC에서 실행된 이유



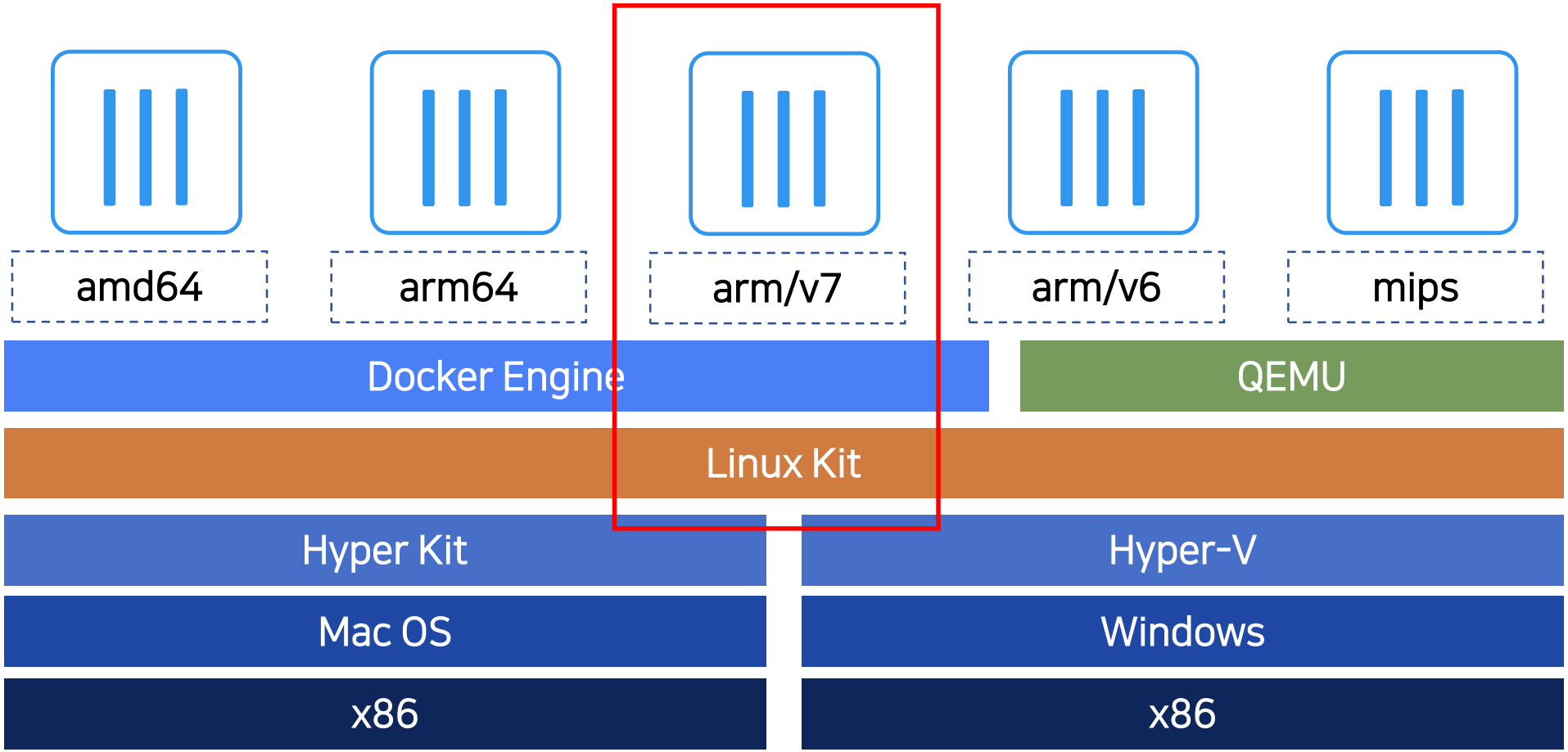
최신 Docker Desktop의 구조

ARM에서 빌드한 이미지가 PC에서 실행된 이유



최신 Docker Desktop의 구조

ARM에서 빌드한 이미지가 PC에서 실행된 이유



최신 Docker Desktop의 구조

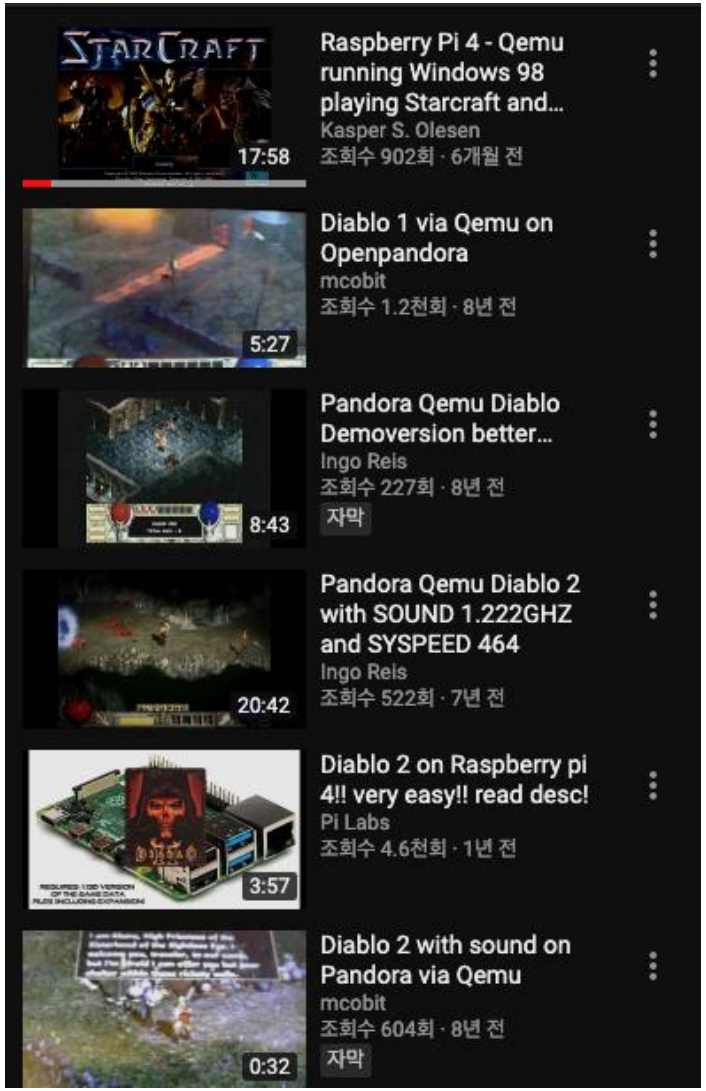
ARM에서 빌드한 이미지가 PC에서 실행된 이유

binfmt_misc(Binary Format Miscellaneous)

- 이기종 포맷의 바이너리가 실행될 때 적절한 인터프리터를 등록

QEMU

- 오픈 소스 에뮬레이터
- 전체 시스템 에뮬레이션 지원
- binfmt_misc를 이용한 사용자 공간 에뮬레이션 지원



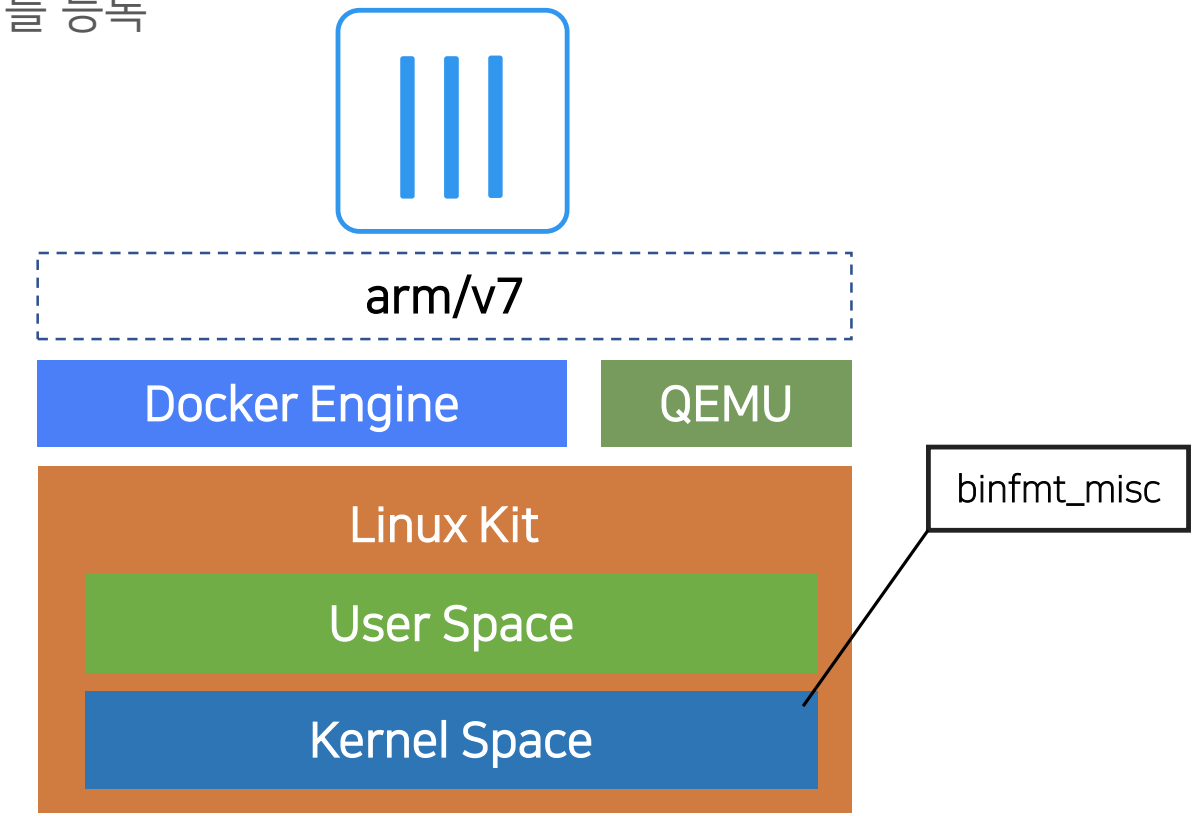
ARM에서 빌드한 이미지가 PC에서 실행된 이유

binfmt_misc(Binary Format Miscellaneous)

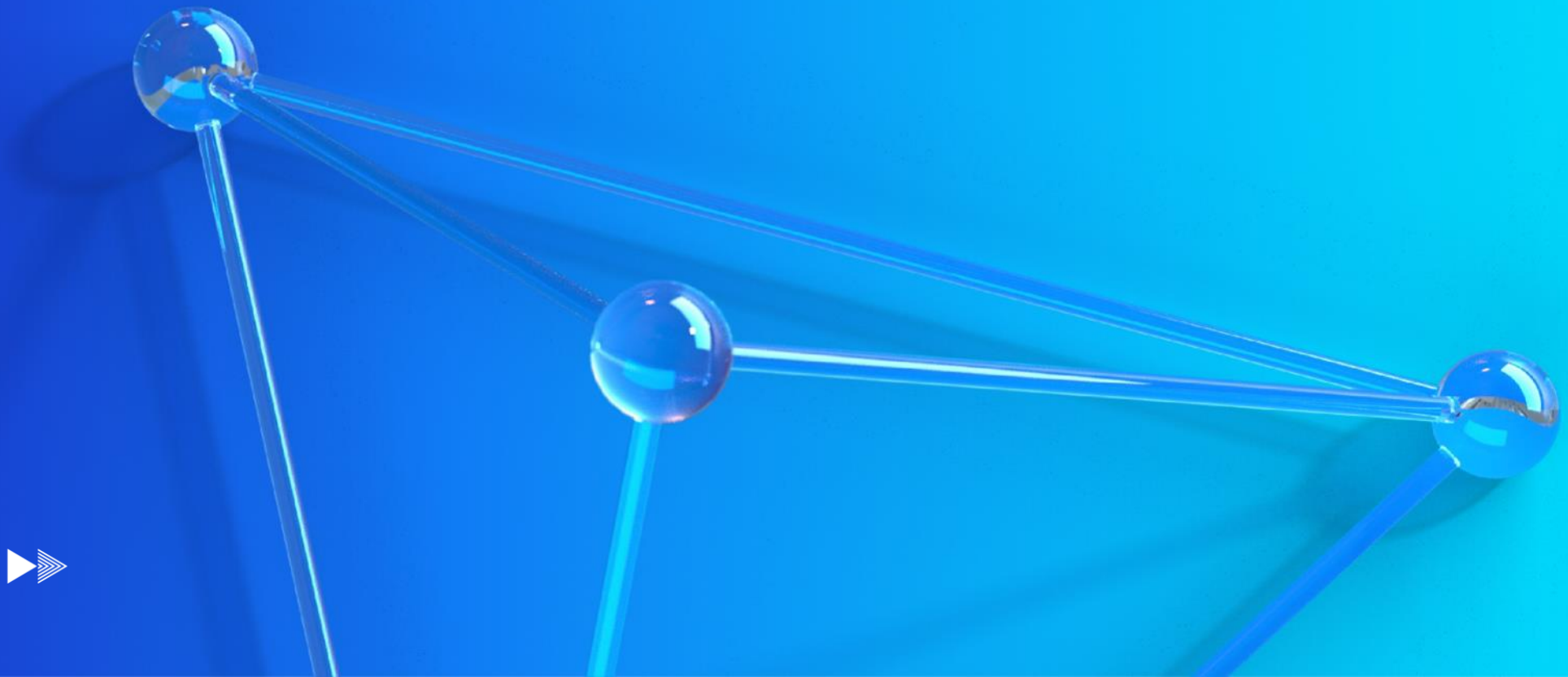
- 이기종 포맷의 바이너리가 실행될 때 적절한 인터프리터를 등록

QEMU

- 오픈 소스 에뮬레이터
- 전체 시스템 에뮬레이션 지원
- binfmt_misc를 이용한 사용자 공간 에뮬레이션 지원



Multi-Arch 이미지를 빌드하는 다양한 방법들

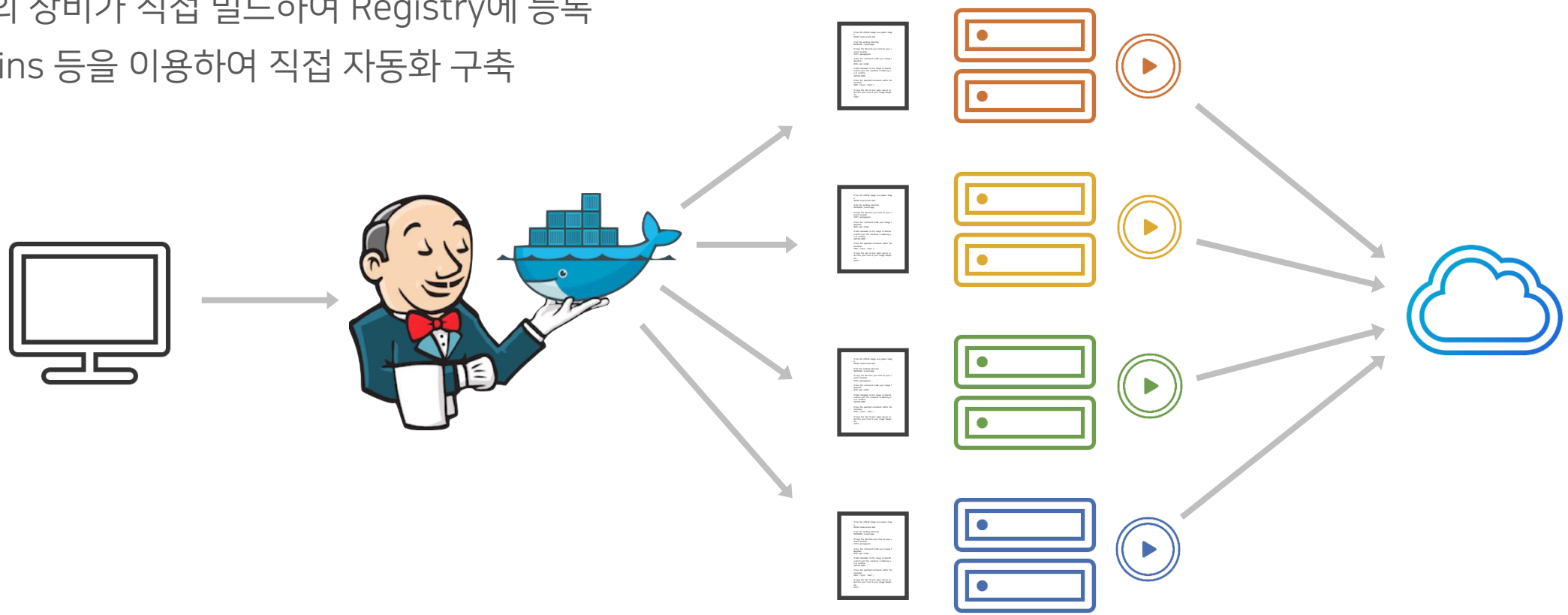


NHN FORWARD

Target에서 직접 빌드

Build Farm을 구축해서 해결

- 모든 종류의 장비가 필요
- 각각의 장비가 직접 빌드하여 Registry에 등록
- Jenkins 등을 이용하여 직접 자동화 구축



Cross Compiler를 이용한 빌드

hello-world 개발자가 이용한 방법

- 임베디드 S/W 분야에서는 여전히 사용하는 방법
- 개발자 PC에 모든 종류의 크로스 컴파일러 설치 필요
- Dockerfile 작성이 매우 복잡

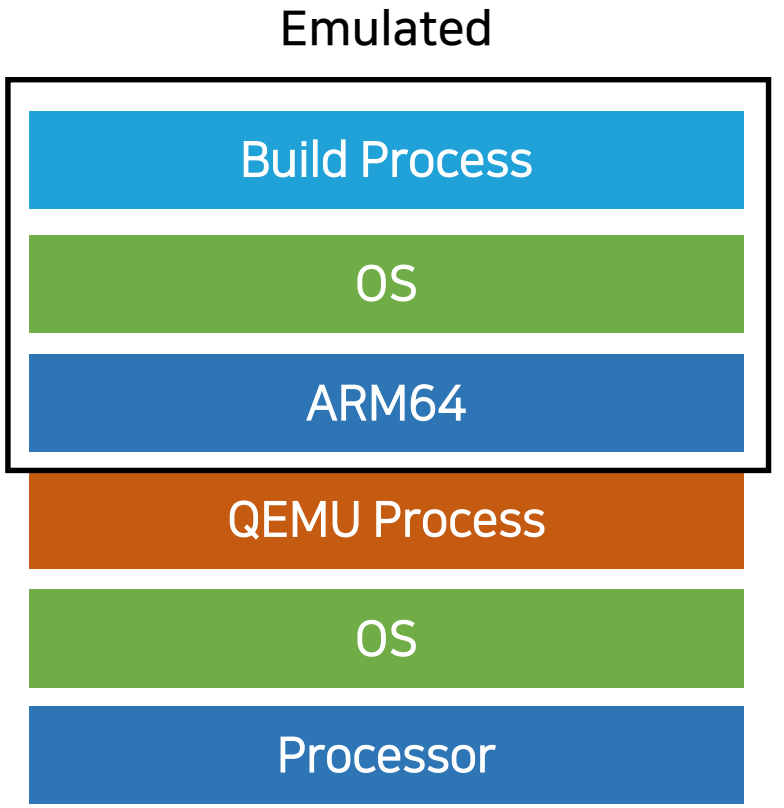
97 lines (84 sloc) | 2.26 KB

```
# explicitly use Debian for maximum cross-architecture compatibility
FROM debian:buster-slim

RUN set -eux; \
        apt-get update; \
        apt-get install -y --no-install-recommends \
                ca-certificates \
                gnupg dirmngr \
                wget \
                \
                gcc \
                libc6-dev \
                make \
                \
                libc6-dev-arm64-cross \
                libc6-dev-armel-cross \
                libc6-dev-armhf-cross \
                libc6-dev-i386-cross \
                libc6-dev-mips64el-cross \
                libc6-dev-ppc64el-cross \
                libc6-dev-s390x-cross \
                \
                gcc-aarch64-linux-gnu \
                gcc-arm-linux-gnueabi \
                gcc-arm-linux-gnueabihf \
                gcc-i686-linux-gnu \
                gcc-mips64el-linux-gnuabi64 \
                gcc-powerpc64le-linux-gnu \
                gcc-s390x-linux-gnu \
                \
                file \
        ; \
        rm -rf /var/lib/apt/lists/*
```

QEMU를 이용한 빌드

전체 시스템 에뮬레이션

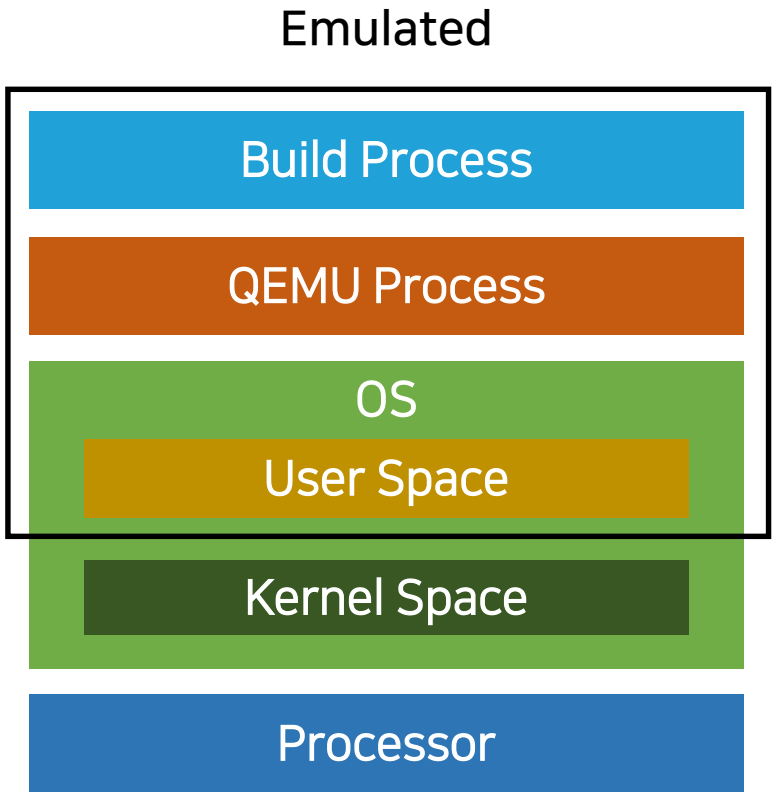
- Target의 Arch와 동일하게 에뮬레이션 한 뒤에 그 안에서 빌드 수행
- 실제 애플리케이션의 수행 범위와 상관없이 모든 시스템을 에뮬레이션
- 동시에 여러 이미지를 빌드하면 과도한 리소스 사용 우려



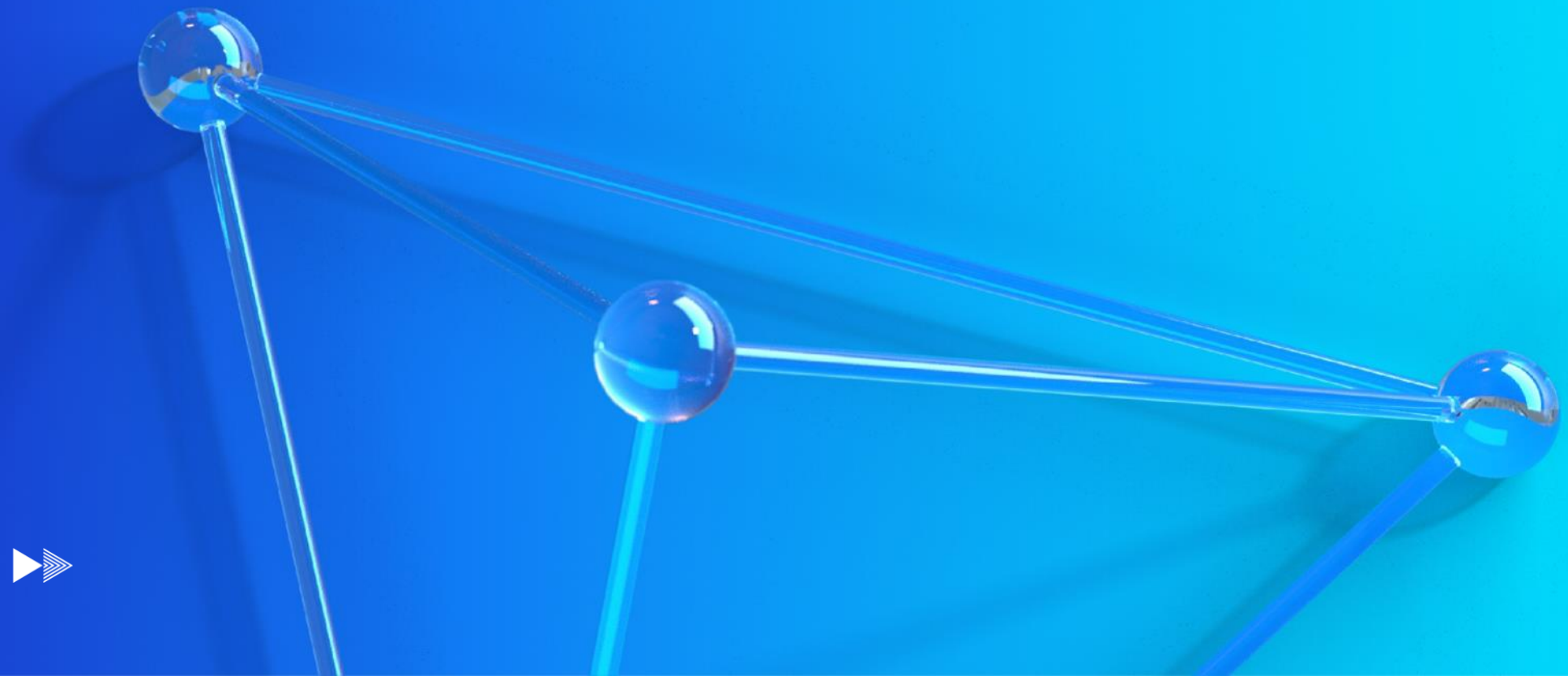
QEMU를 이용한 빌드

사용자 공간 에뮬레이션

- QEMU의 옵션으로 OS의 사용자 영역만 에뮬레이션하여 빌드
- 빌드 수행에 binfmt_misc가 필요
- 전체 시스템 에뮬레이션에 비해 빠르고 효율적
- Docker BuildX의 기반이 됨



BuildX를 이용하여 훨씬 더 쉽게!



NHN FORWARD

TL:DR

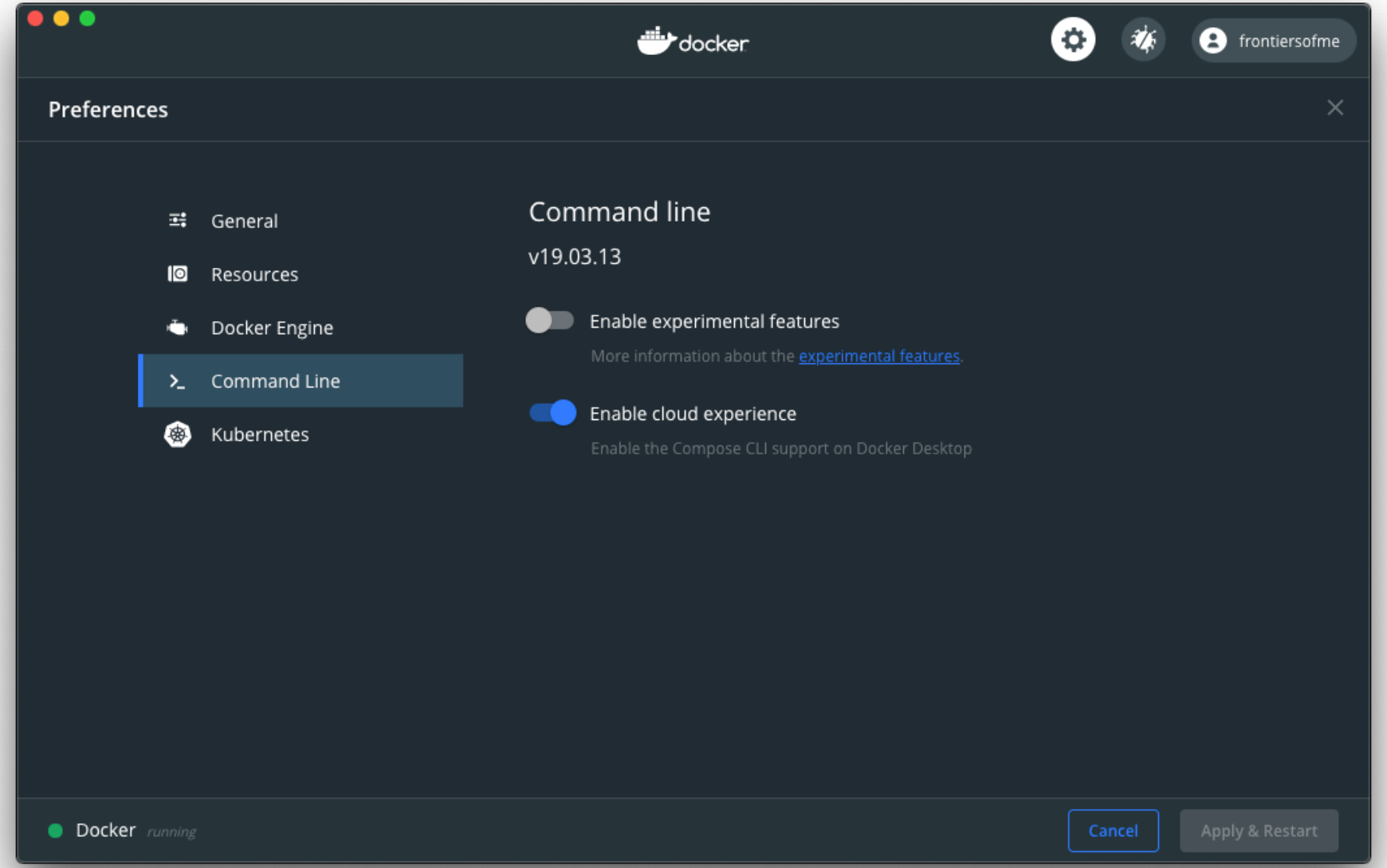
NHN FORWARD

BuildX가 뭔데?

주요 기능

- CLI 실험적 기능 중 하나
- 기존 빌드 명령어와 유사한 인터페이스
- 다중 빌더 인스턴스 제공
- 별도의 Dockerfile 작성이 필요 없음
- In-Container 드라이버 지원(Docker & Kubernetes)
- Compose 빌드 지원
- Moby BuildKit에서 제공되는 모든 기능 지원

CLI Experimental Features 활성화



Buildx Builder 생성


```
# Builder Instance 확인
$ docker buildx ls
NAME/NODE             DRIVER/ENDPOINT             STATUS PLATFORMS
default               docker
  default             default                     running linux/amd64, linux/arm64,
linux/riscv64, linux/ppc64le, linux/s390x, linux/386, linux/arm/v7, linux/arm/v6
```

Buildx Builder 생성


```
# Builder Instance 확인
$ docker buildx ls
NAME/NODE             DRIVER/ENDPOINT               STATUS PLATFORMS
default               docker
  default             default                       running linux/amd64, linux/arm64,
linux/riscv64, linux/ppc64le, linux/s390x, linux/386, linux/arm/v7, linux/arm/v6

# Builder Instance 생성 및 사용 설정
$ docker buildx create --name <builder> --driver docker-container --use
<builder>
```

생성된 Builder Instance 확인


```
# 생성된 Builder Instance 확인
$ docker buildx ls
NAME/NODE           DRIVER/ENDPOINT             STATUS  PLATFORMS
all-arch-builder *  docker-container
  all-arch-builder0 unix:///var/run/docker.sock running linux/amd64, linux/arm64,
linux/riscv64, linux/ppc64le, linux/s390x, linux/386, linux/arm/v7, linux/arm/v6
default             docker
  default           default                     running linux/amd64, linux/arm64,
linux/riscv64, linux/ppc64le, linux/s390x, linux/386, linux/arm/v7, linux/arm/v
```

Multi-Arch 이미지 빌드

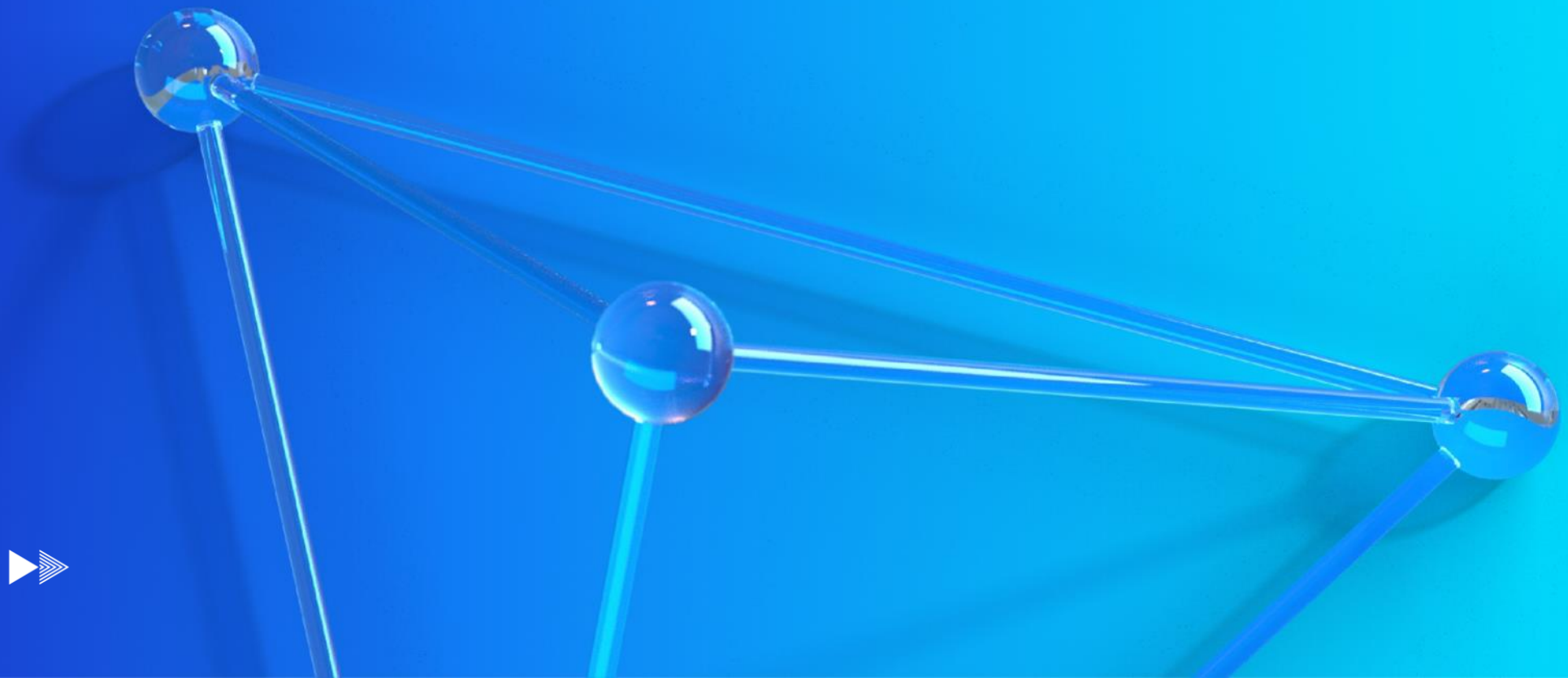

```
# platform 설정과 함께 이미지 build 및 완료 시 registry로 push
$ docker buildx build --platform <platform>,.. -t <repository>/<image> --push .

...

# 병렬로 빌드 진행
 => [linux/amd64 internal] load metadata for docker.io/library/python:latest    2.7s
 => [linux/386 internal] load metadata for docker.io/library/python:latest      2.8s
 => [linux/arm64 internal] load metadata for docker.io/library/python:latest    2.7s
 => [linux/arm/v7 internal] load metadata for docker.io/library/python:latest   2.9s

...
```

참 쉽죠?

BuildX의 제약 사항

- Docker Engine v.19.03 부터 내장됨
- 이전 버전에서는 플러그인 형태로 별도 설치 필요
- Linux에서 사용시 Kernel 버전 4.8 이상(binfmt_misc)
- Linux에서 사용시 QEMU를 비롯한 별도 의존성 설치 필요

후기

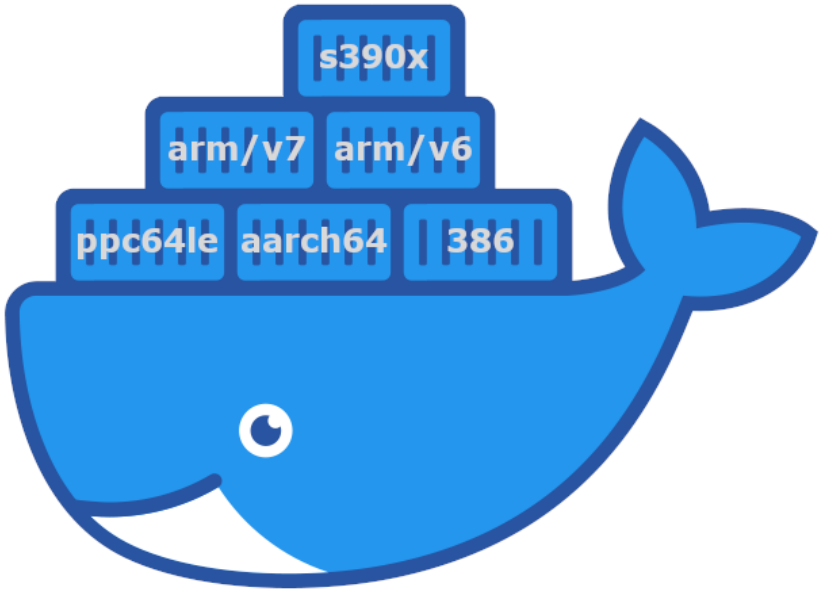


NHN FORWARD

후기

잘 몰랐던 Docker 이야기

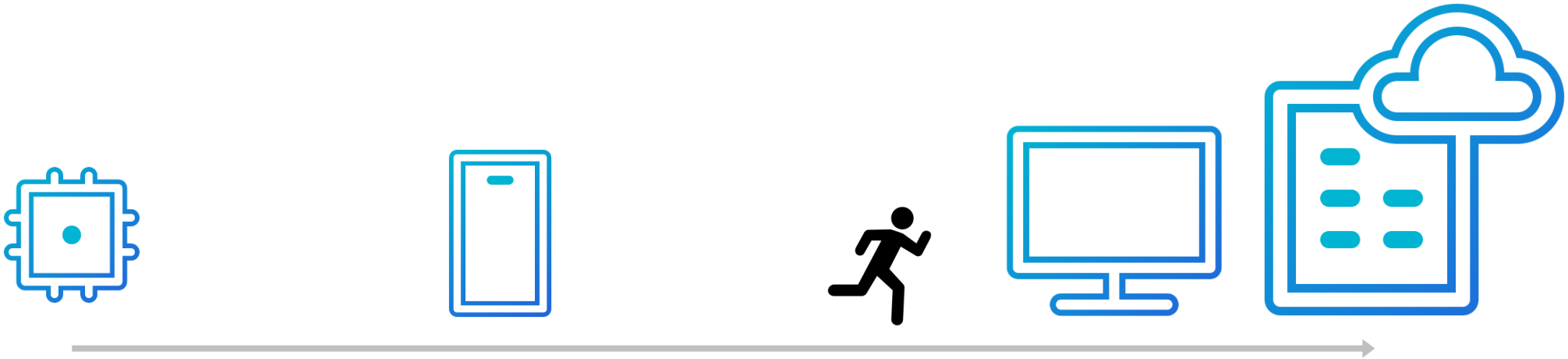
- Host OS가 Linux가 아니라면 가상 머신이 필요(~~컨테이너 기술은 가상 머신과 다르다~~)
- 기존 Build 명령어로 만든 이미지는 플랫폼 의존적
- 여러 플랫폼을 지원하기 위해 별도의 처리가 필요
- Docker Registry가 적합한 이미지를 찾아줌
- 굉장히 번거롭고 오래 걸리는 작업을 BuildX가 해결



그래도 의미는 있다

애플리케이션 & 서버 개발자에게도 점점 다가오는 ARM

- 더이상 남의 일이 아닐 수도 있다



참고 자료

- https://www.docker.com
- https://www.youtube.com/user/dockerrun
- https://youtu.be/ALn0hUxNszI

이미지 출처

- p02 : https://miro.medium.com/max/700/0*D9HRbEeH5vNKYN1p.jpg
- p05 : https://github.com/kubernetes/kubernetes/blob/master/logo/logo.svg
  p05 : https://www.iconfinder.com/icons/4373190/docker_logo_logos_icon
- p07 : https://upload.wikimedia.org/wikipedia/commons/4/4e/Odroid-XU4_Board_Layout.jpg
- p31 : https://www.docker.com/sites/default/files/d8/2018-11/docker-containerized-appliction-blue-border_2.png
- p31 : https://www.docker.com/sites/default/files/d8/2018-11/container-vm-whatcontainer_2.png
- p39 : https://miro.medium.com/max/700/0*WPUrgP0-oC4NeYwA.png
- p53 : https://nexus.eddiesinentropy.net/2020/01/12/Building-Multi-architecture-Docker-Images-With-Buildx/multi-architecture-docker.png

고맙습니다.

NHN FORWARD