Truly understainding container

이어형

Me: 어형부형

KT: openstack 기반 public storage 서비스 개발,운영

Kakao: openstack 기반 private cloud 서비스 개발,운영

현재 Line: kubernetes 기반 private cloud native 서비스 개발,운영

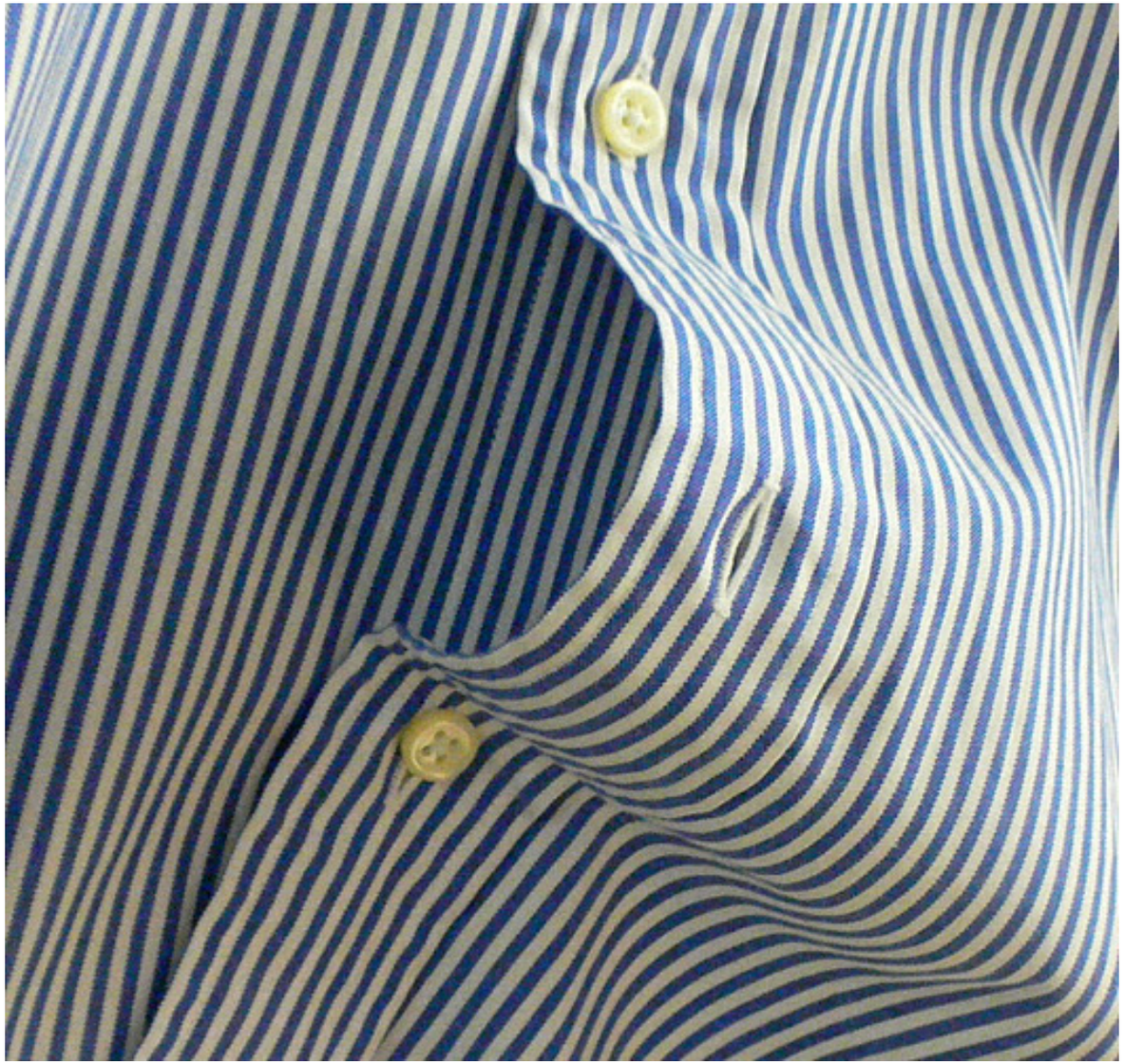
주제

컨테이너란 무엇인가 다시 생각해보고

컨테이너의 한계라고 여겨지는 것들에 대해

해결할 수 있는 방법들을 살펴보며

이로 향후 컨테이너의 나아갈 방향을 생각해봅니다.



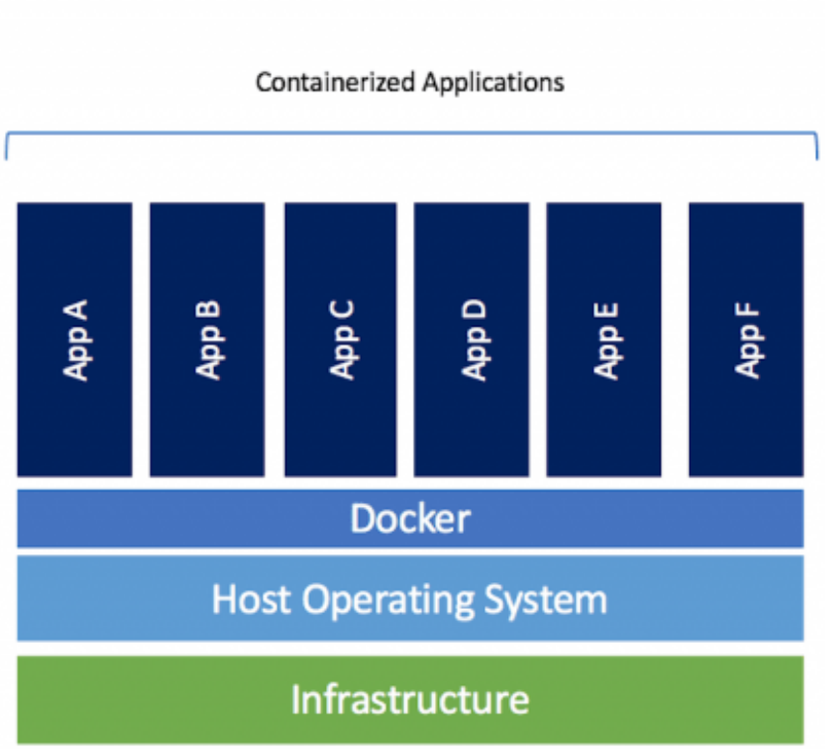
Container와 Virtual machine은 비교대상인가?

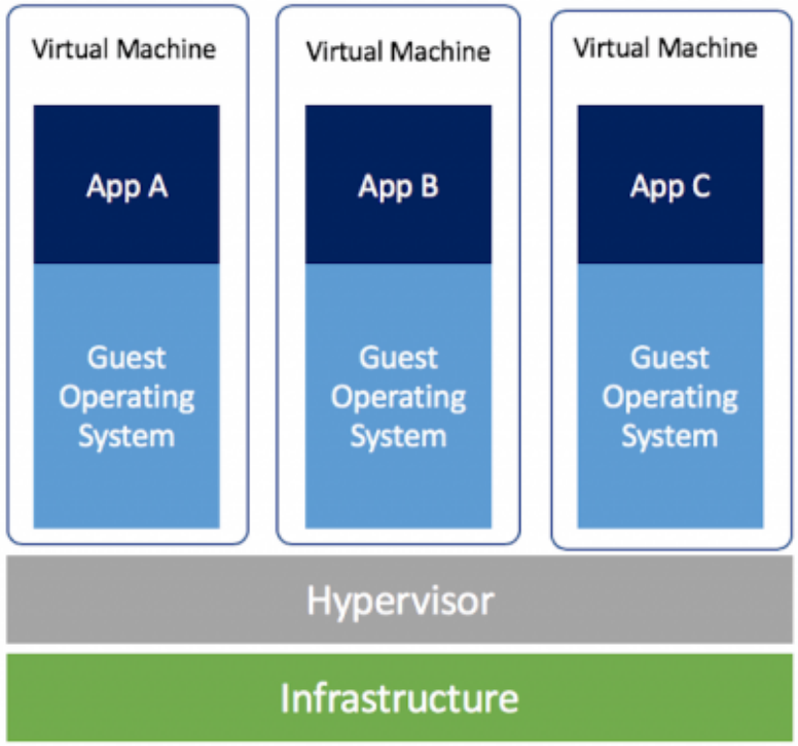
Container and Virtual Machine의 같은점

사용자의 **workload**를 처리

그렇다면 다른 점은?

구조?





Container vs Virtual Machine

image from docker blog

성능?

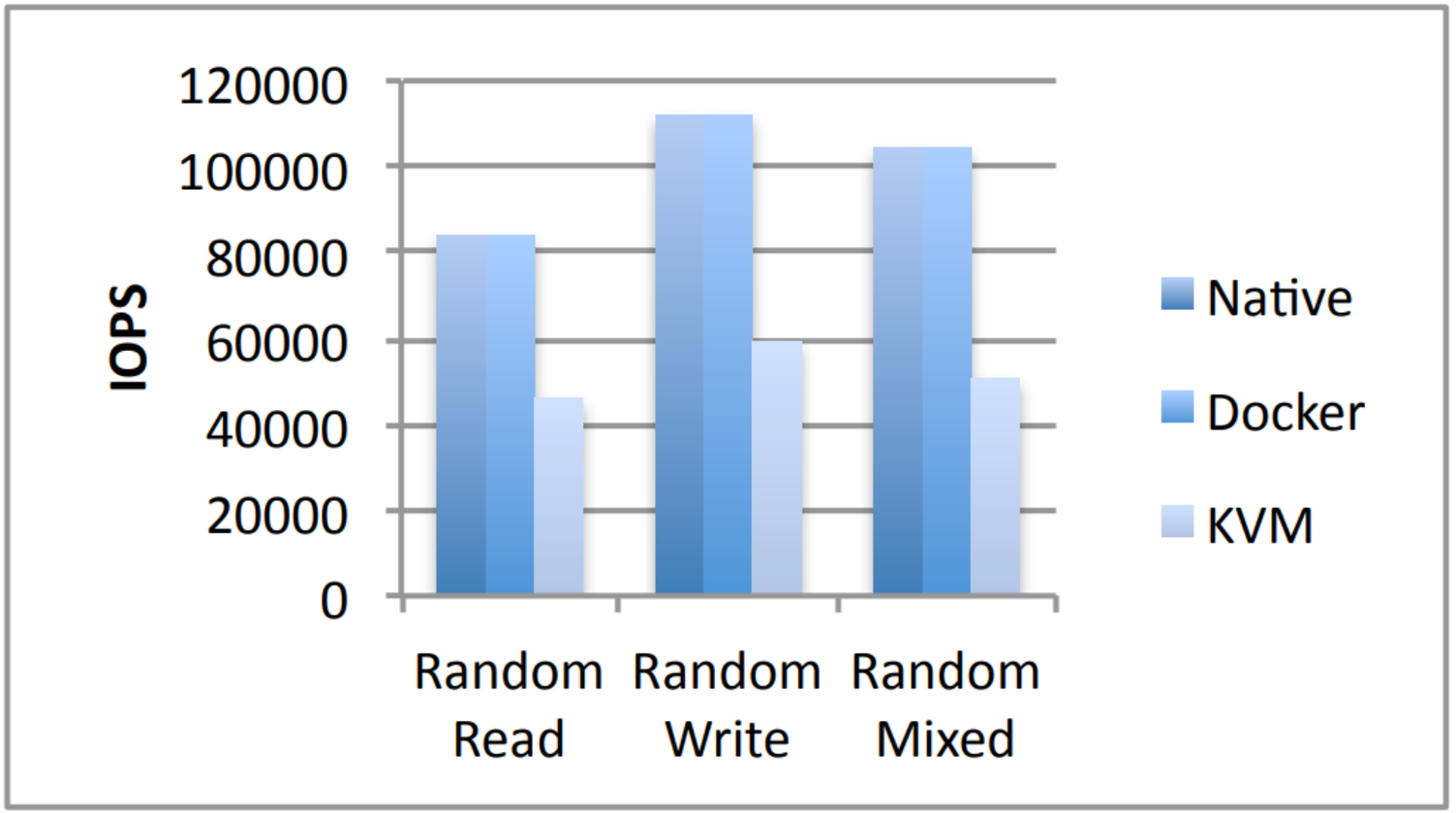


Fig. 6. Random I/O throughput (IOPS).

Container vs Virtual Machine - IOPS

image from An Updated Performance Comparison of Virtual Machines and Linux Containers

성능이 떨어지는 이유는 구조 때문인가?

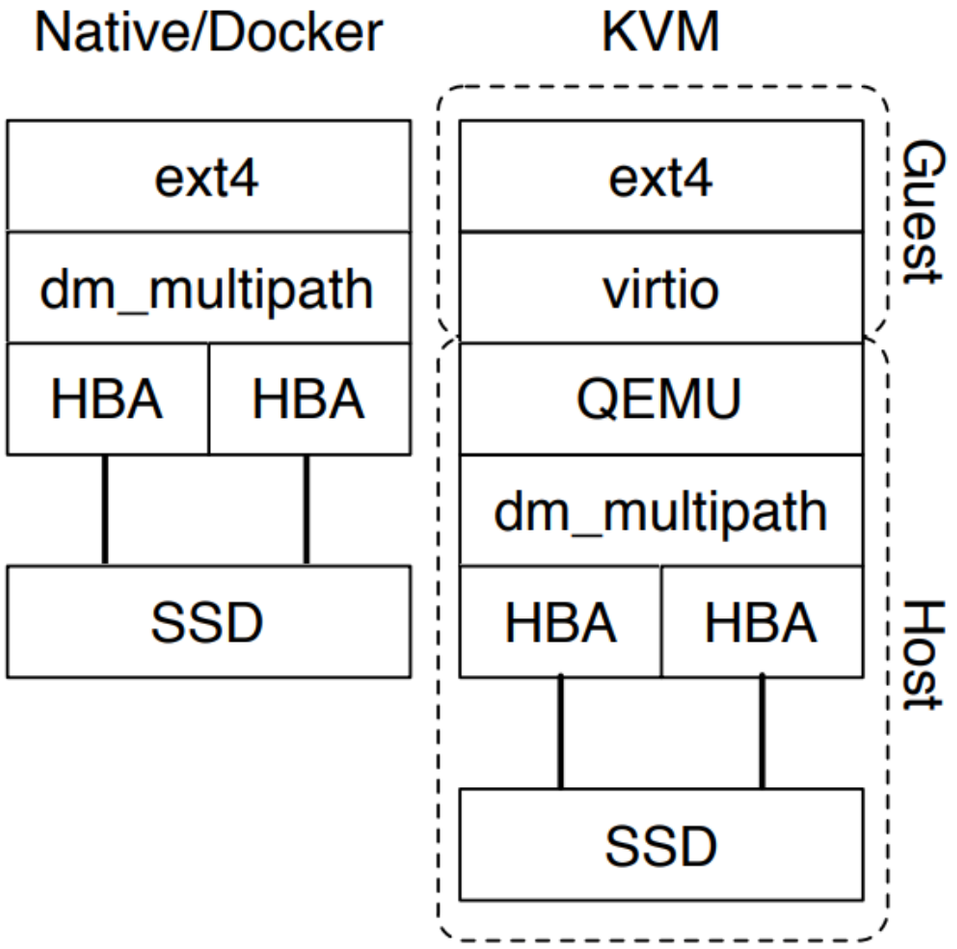
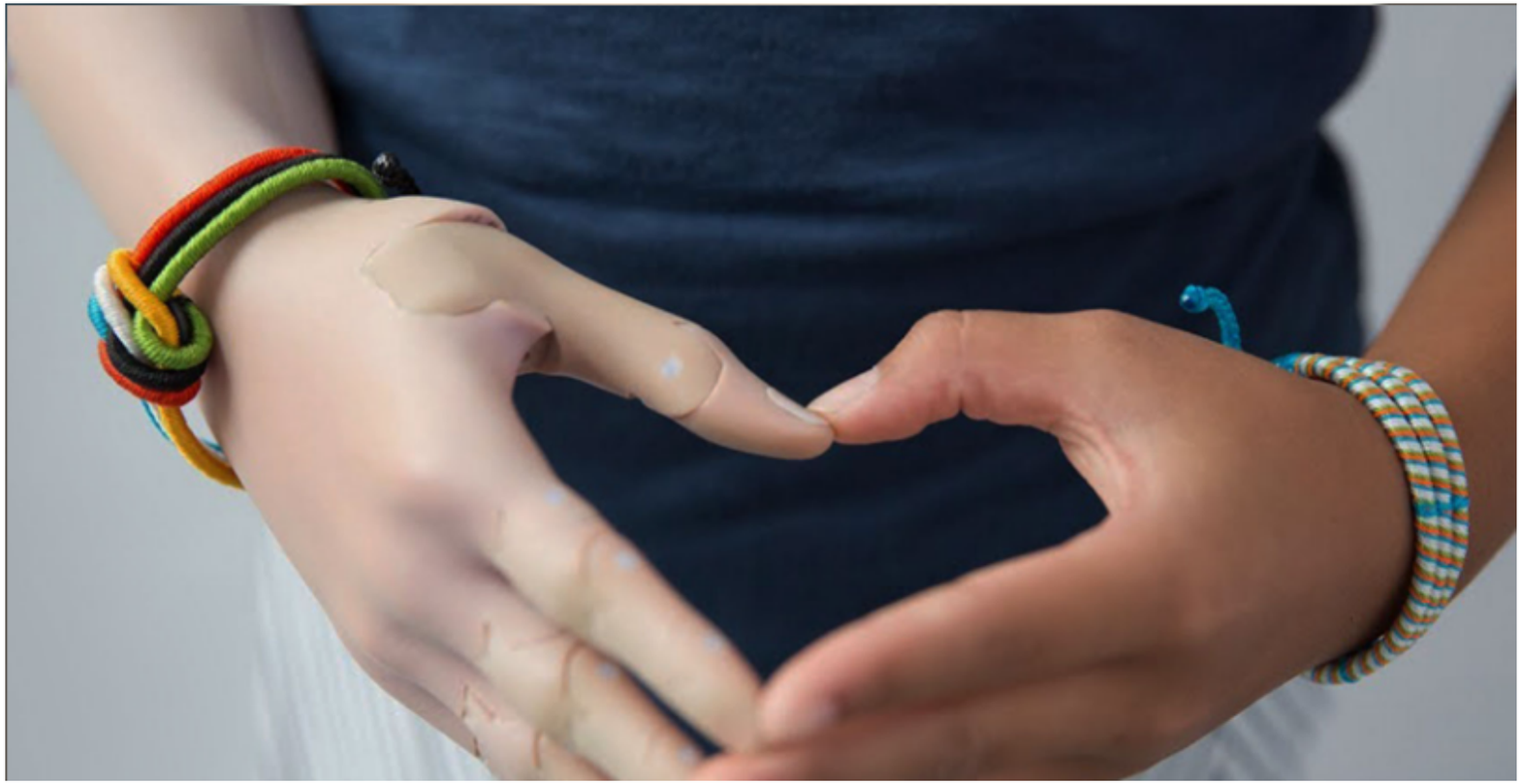


Fig. 4. Storage configurations used for `fio`

Container vs Virtual Machine - virtio

virtio, QEMU의 존재 이유?



emulation - 본뜸

image from unlimitedtomorrow

emulation

장점: 호환성을 제공하여 종래의 것을
그대로 치환(or 이전)가능함

단점: 모방하기 위한 비용(성능 등)이 큼

그러면 container는?

최초에는 isolation만이 고려

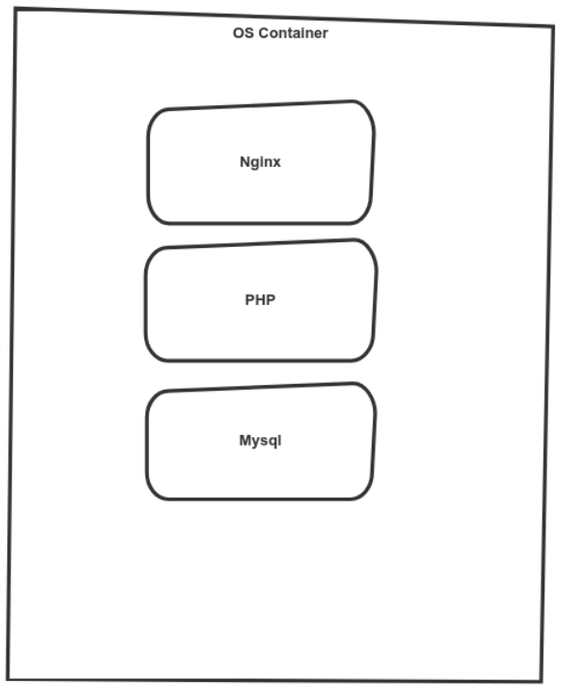
emulation의 오버헤드 없이 사용하고자 했음

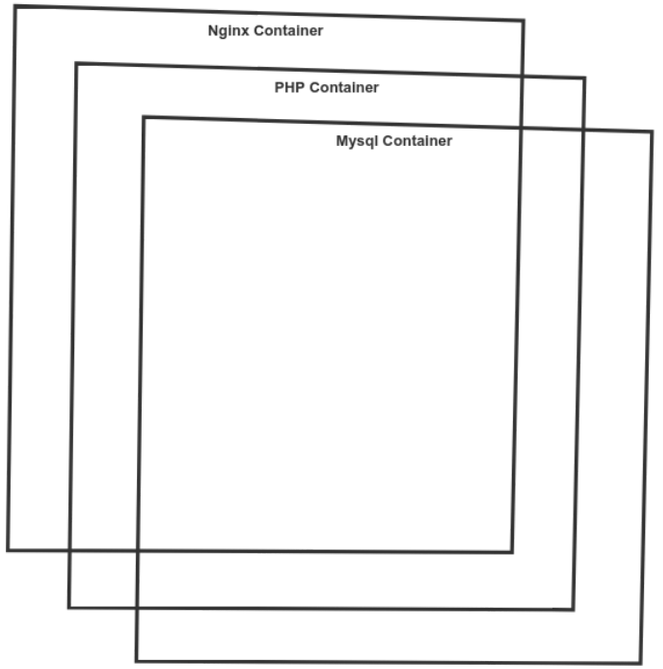
os container로 시작

VM이 hardware virtualization이라면

container는 os virtualization 라고 했던 시절이..

os container vs app container





OS Containers

- Run as an OS
- Run multiple services in the same container
- Use native resource isolation (Linux facilities)

Examples:

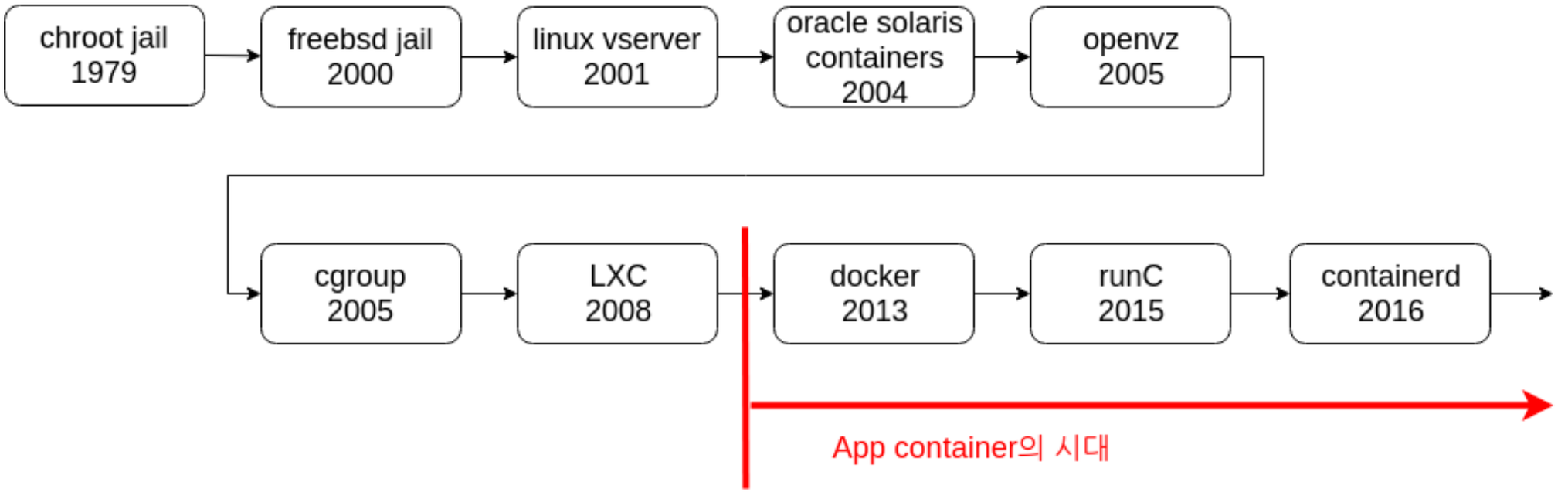
- LXC
- OpenVZ
- Linux-VServer
- Solaris Containers

App Containers

- Run as an islaoted application
- Run a single process/services per container
- Was built on top of OS Containers

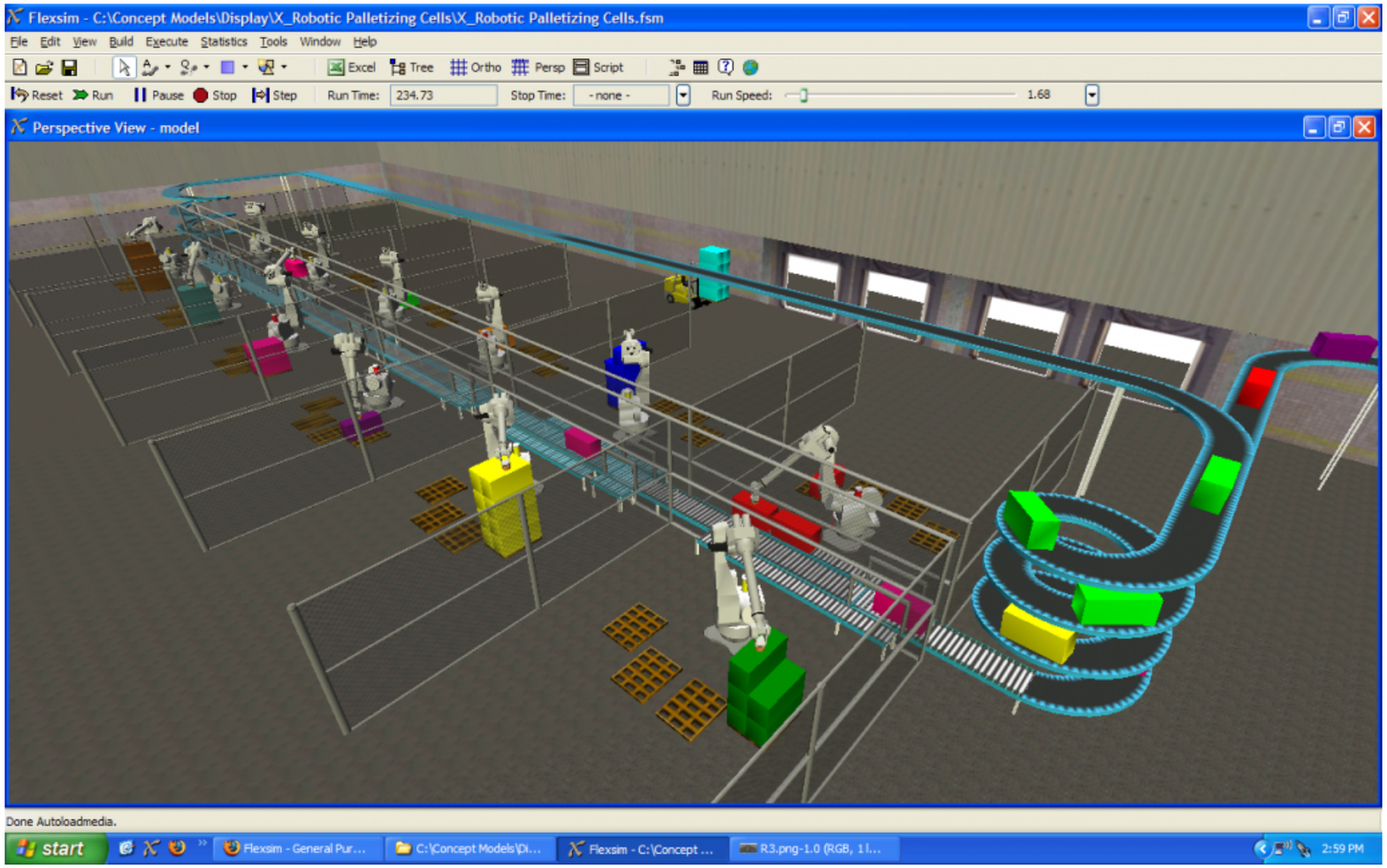
Examples

- Docker
- RKT



History of Container

app container를 한마디로 한다면?



simulation - 모의실험

from talumis

그러면 왜 simulation 이라 하는가?

결과를 예측가능하게 만들기 위해서
재사용성을 높인다.

이 과정에서 최적화가 가능

== 컨테이너 특성과 유사

그러면 무엇에 대한 simulation인가?

physical machine이 아닌 **process**

그러면 무엇으로 simlation을 제어하는가?

cgroup과 **namespace**

cgroup과 namespace

cgroup: resource(memory, cpu, ...)을 관리하기 위한 기술

namespace: 논리적인 독립공간을 제공하기 위한 기술

namespace 종류

mnt : mount

uts : hostname

ipc : SystemV IPC

pid : process

net : network

uid : user ID

time , syslog namespace not exist

namespace의 의미

어디에서나 해당 namespace의 내용이

유지되거나 조작가능하여 재사용성을 높임

어떻게 namespace를 생성하는가?

unshare - disassociate parts of the process execution context

```
$ # 이미지 추출
$ docker export blissful_goldstine -o dockercontainer.tar
$ mkdir rootfs
$ tar xf dockercontainer.tar --ignore-command-error -C rootfs/
$ # container 생성
$ unshare --mount --uts --ipc --net --pid --user \
   --fork --map-root-user chroot $PWD/rootfs ash
root:# mount -t proc none /proc
root:# mount -t sysfs none /sys
root:# mount -t tmpfs none /tmp
```

이를 이용해서 컨테이너 생성 가능

image from how-to-run-docker-containers-using-common-linux-tools-without-docker

다만 아직 남은게 있음

resource는 분할 했지만 아직 사용량제어를 하지 않음

그래서 여기에 cgroup을 적용해야 함

그리고 마지막으로

network을 외부 연결해줘야 함

현재의 network는 갇힌 외부로 연결이 불가능한 network namespace 상태

그래서 namespace를 가로지르는 통신방법이 필요

veth: virtual ethernet

pair로 존재하며 다른 namespace에 동일하게 존재하여

한 pair로 들어온 트래픽은 다른 pair의 veth로 복사됨

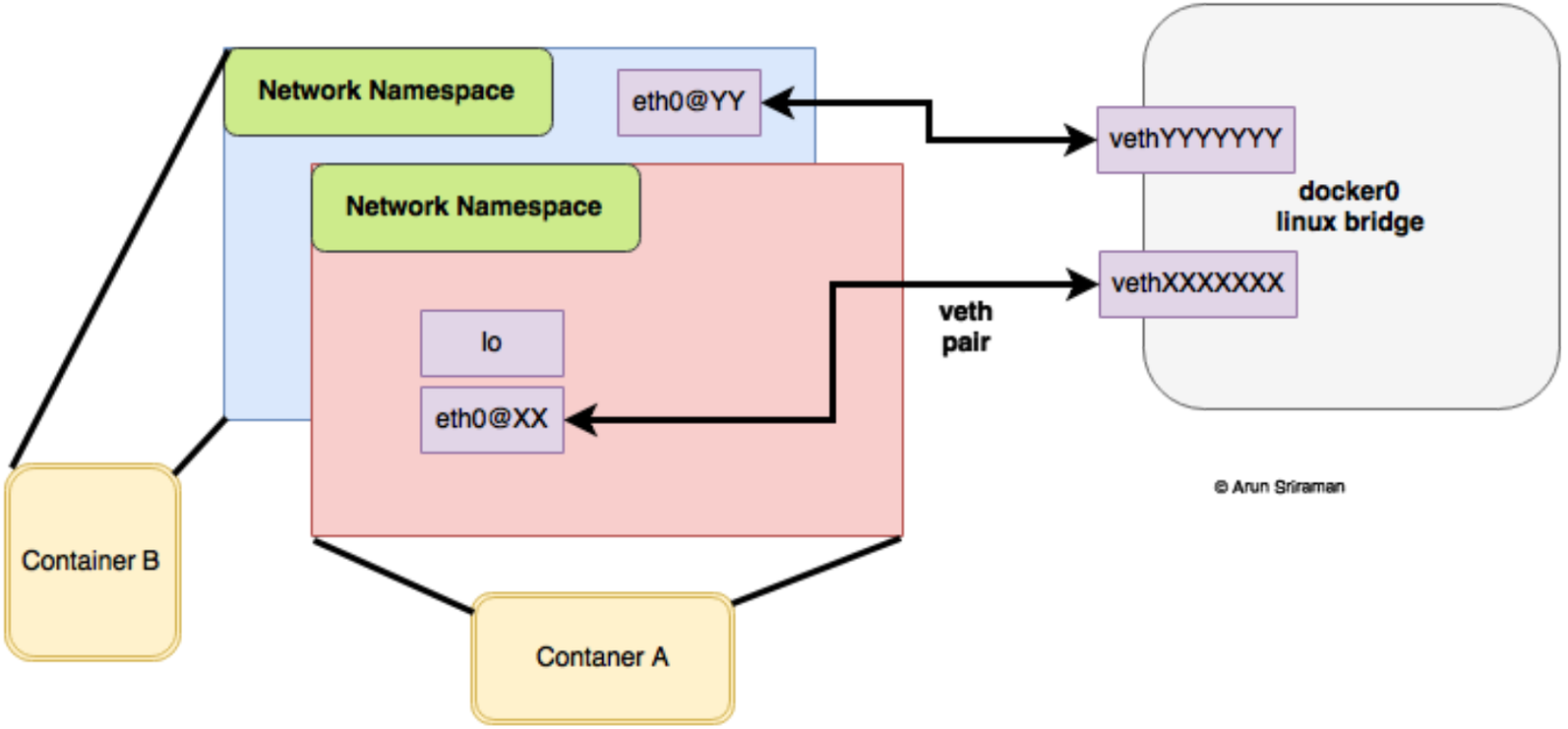


image from Container Namespaces – Deep Dive into Container Networking

최적화된 방법이란?

예 - IP 할당

	virtaul machine	container
예(IP할당)	실제 dhcp를 L2 layer로 broadcasting 한 후에 결과로 IP를 받음	dhcp를 사용하지 않고 container의 가상의 interface에 IP를 부여
의미	dhcp를 그대로 사용가능하여 physical machine등도 같이 사용가능	IP를 dhcp가 아닌 다른 방법으로 관리가 필요해짐

예 - container에게 restart가 필요한가?

kubernetes엔 container의 restart가 없음

왜 없을까?

kubectl 참고

Virtual machine vs container 목표

- VM: **physical machine**의 기능을 최대한 닮게 소프트웨어로 구현
- app container: **process**가 해야하는 목표를 재현율을 높이며 최적화된 방법을 소프트웨어로 구현

결국 physical machine 과 process의 차이 관점으로 봐야함

결국

이전보다 더 좁은 scope로
보다 목표 중심적으로 사고를 할 수 있게 됨

container를 사용하면서 기존의 방식을
재정의하여 최적화 한 방법들이 나타남

이는 **cloud native**로 연결됨

container의 한계

docker와 container와의 관계

old: docker **is** container

new: docker **is** container `platform`

docker의 문제

unregister netdevice error

아직까지 해결되지 않은
커널 패닉을 유발하는 문제가 존재

from kernel-crash-after-unregister_netdevice-waiting-for-lo-to-become-free-usage-count-3

docker daemon에 종속성이 있음

docker restart시 컨테이너에 영향

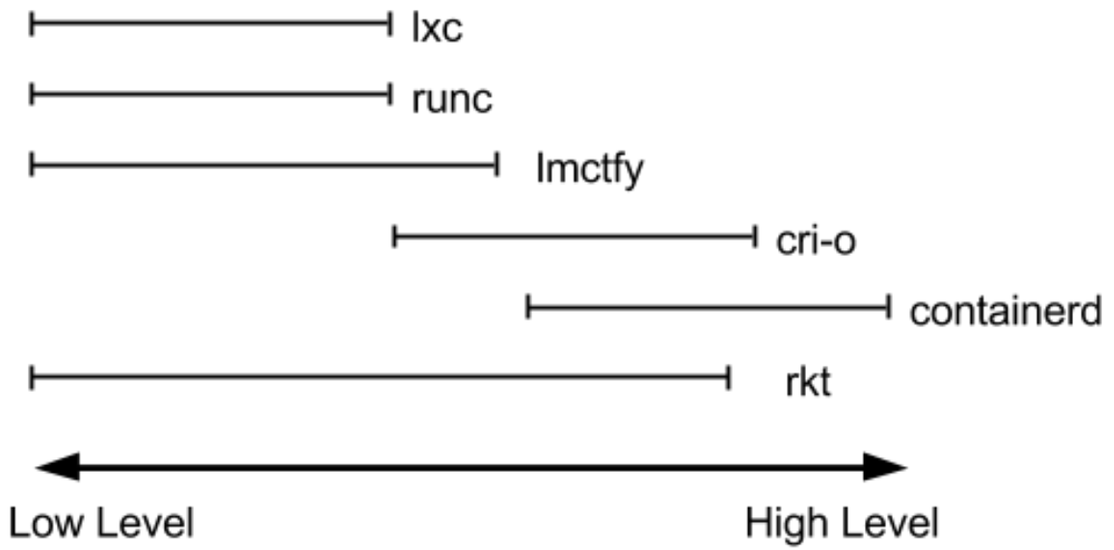
docker socket이 죽을 시 컨테이너에 영향

한계 #1

너무 많은 기능을 가지고 있는
docker daemon의 종속성으로
한번에 노드가 망가지기가 쉽다.

우선 그러면 docker는 대체 가능 한가?

runtimes



from container-runtimes-part-1-introduction-container-r

kubernetes container runtime interface

docker 없이 container를 사용하기 위한 interface들이 제공됨

containerd

단순성, 견고성, 휴대성을 중시한
산업 표준 container runtime

containerd performance

다만 새로운 runtime들을 사용시

대부분 이름 그대로 runtime만 가지고 있음

build는 별도

그렇다면 build는?

buildkit

동시성, 캐시 효율성 이 있으며
Dockerfile에 구속받지않는 빌더 툴킷

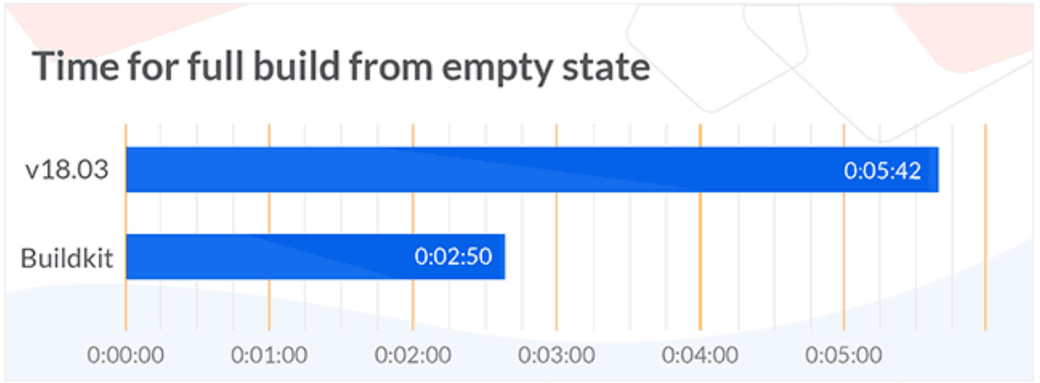


Fig 4a – Based on the docker build from scratch, the results are 2.5x faster build of the sample project.

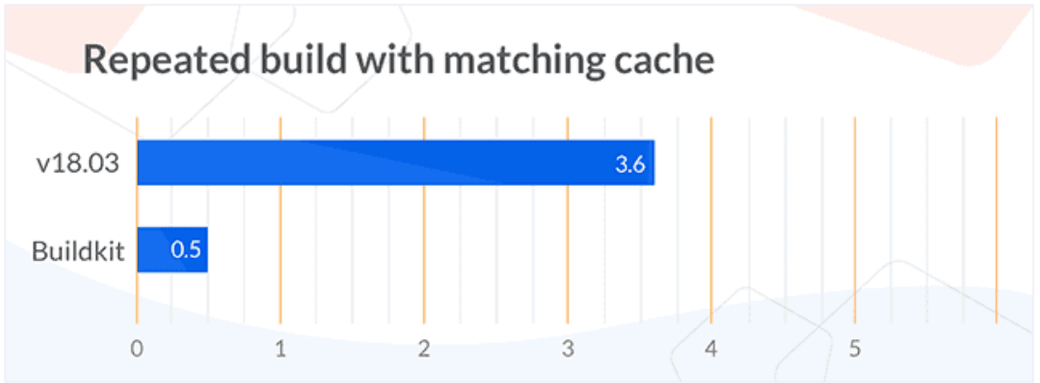


Fig 4b – Rerunning the same build with local cache the speed is 7x faster.

결론

새로운 runtime을 사용하면
buildtime, runtime을 구분 함

이를 통해 container service에서는
오직 runtime만 사용하여 보안성 및 성능 향상

한계 #2

docker는 root만이 해당 OS에 설치가능

취약점 발생시 OS레벨
문제 일으킬 가능성들이 존재함

root로 사용하는 컨테이너들

USER syntax등을 사용하지 않으면 기본적으로 root 이기에

일반적으로 root user를 사용해서 docker를 이용

```
$ docker run -ti busybox whoami
root
```

외부 device 혹은 directory mount시 일반 user사용이 힘든

권한 문제발생

```
$ cat Dockerfile
...

# Add chrome user
RUN groupmod -g 995 audio \
    && groupmod -g 986 video \
    && groupadd -g 1000 -r chrome \
    && useradd -u 1000 -r -g chrome -G audio,video chrome \
    && mkdir -p /home/chrome/Downloads \
    && chown -R chrome:chrome /home/chrome

...

# Run Chrome as non privileged user
USER chrome
```

root로 사용하는 컨테이너들

일반적으로 root를 사용해서 docker를 이용

```
vscode(){
    docker run -d \
        -v /etc/localtime:/etc/localtime:ro \
        -v /tmp/.X11-unix:/tmp/.X11-unix  \
        -v /usr/share/fonts:/usr/share/fonts \
        -v /usr/lib/locale/locale-archive:/usr/lib/locale/locale-archive \
        -v "${HOME}/.vscode/extension:/home/user/.vscode" \
        -v "${HOME}/.vscode/config:/home/user/.config" \
        -e "GTK_IM_MODULE=uim" \
        -e "XMODIFIERS=@im=uim" \
        -e "QT_IM_MODULE=uim" \
        -e "LC_ALL=ko_KR.UTF-8" \
        -v "$(pwd):/project" \
        -e "DISPLAY=unix${DISPLAY}" \
        --device /dev/dri \
        --name visualstudio \
```

#TODO 일반유저를 줘서 사용시 uid, gid 치환문제가 발생

rootless container

root 권한이없는 사용자가
컨테이너를 작성, 실행 및 관리 할 수있음

from rootlesscontaine.rs

이런건 rootless container가 아님

- `docker run --user foo`
- Dockerfile 안에 `USER`
- `usermod -aG docker foo`
- `sudo docker` or `chmod +s dockerd`
- `dockerd --userns-remap`

from rootlesscontaine.rs

podman

Podman linux 시스템에서 OCI 컨테이너를
개발, 관리 및 실행하기 위한 데몬이 없는 컨테이너 엔진

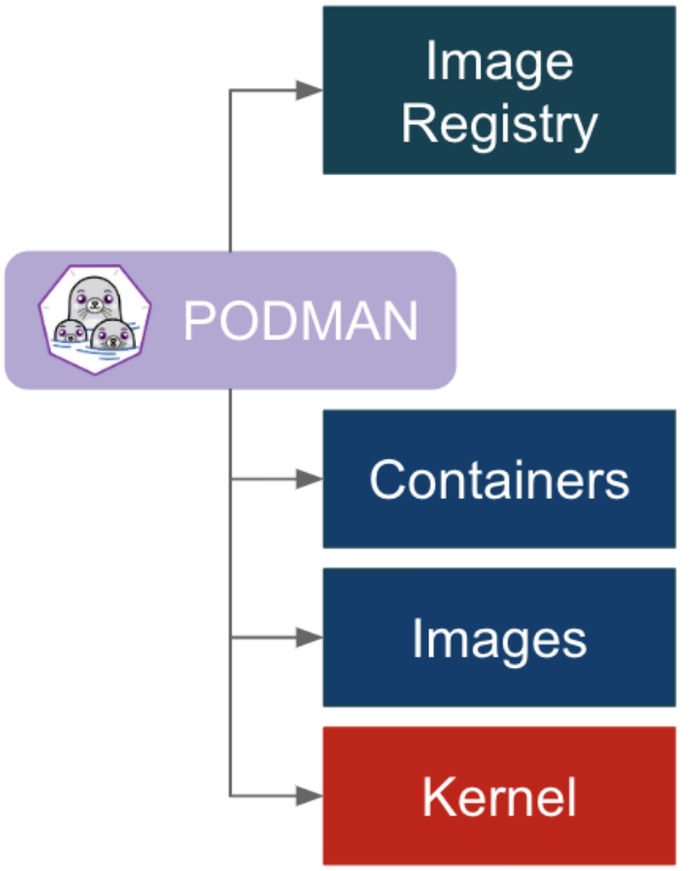
```
alias docker=podman.
```

podman

from podman

데몬 없는 podman

podman 커맨드가 직접 linux에 접근해서 모든것을 실행

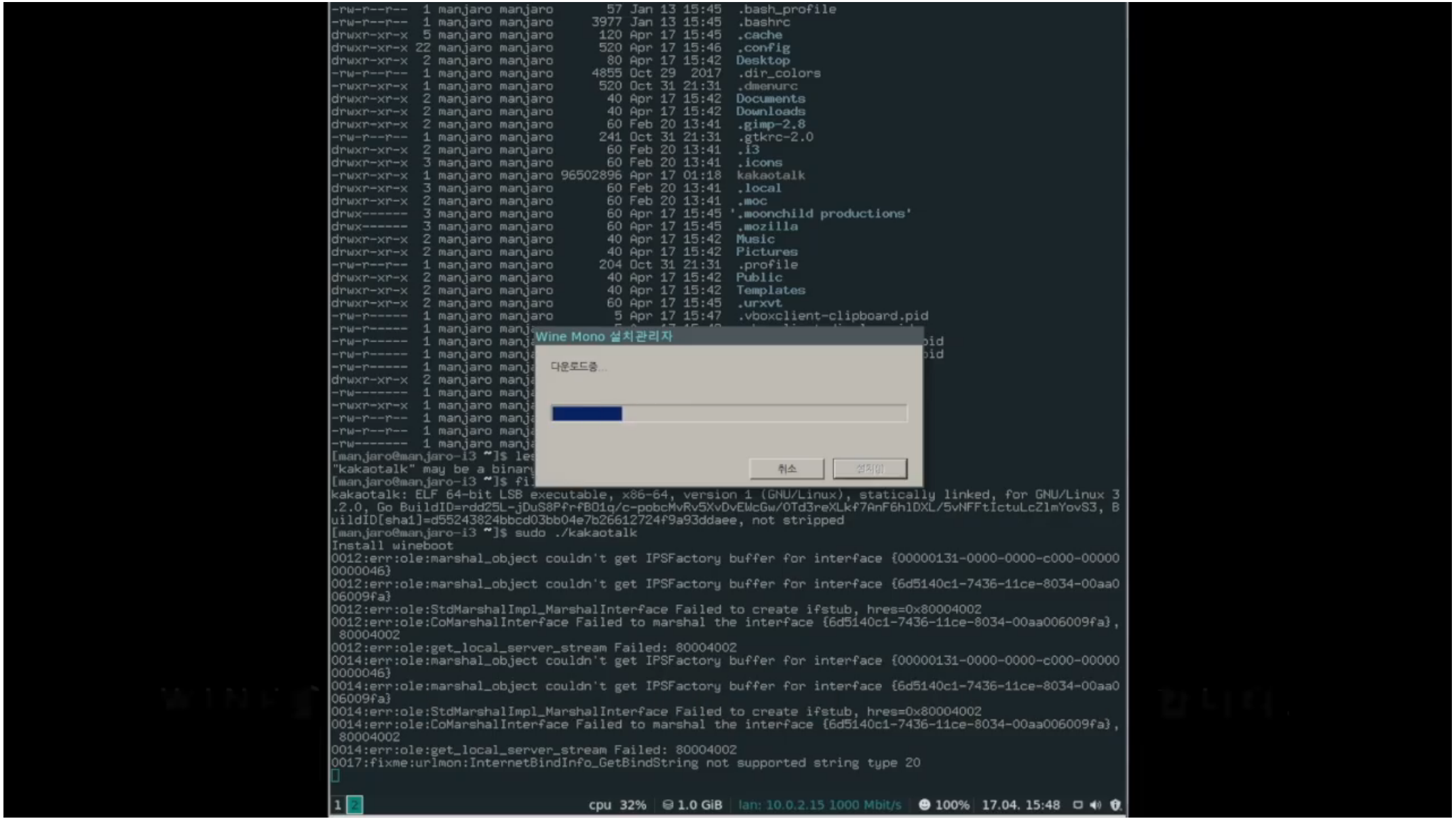


binctr

실행 가능한 바이너리로서 완전 정적이고 권한없이도
실행가능한 독립적 인 컨테이너.

from binctr

binctr 예


-rw-r--r-- 1 manjaro manjaro   57 Jan 13 15:45 .bash_profile
-rw-r--r-- 1 manjaro manjaro   3977 Jan 13 15:45 .bashrc
drwxr-xr-x 5 manjaro manjaro   120 Apr 17 15:45 .cache
drwxr-xr-x 22 manjaro manjaro   520 Apr 17 15:46 .config
drwxr-xr-x 2 manjaro manjaro   80 Apr 17 15:42 Desktop
-rw-r--r-- 1 manjaro manjaro   4855 Oct 29 2017 .dir_colors
-rwxr-xr-x 1 manjaro manjaro   520 Oct 31 21:31 .dmenurc
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Documents
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Downloads
drwxr-xr-x 2 manjaro manjaro   60 Feb 20 13:41 .gimp-2.8
-rw-r--r-- 1 manjaro manjaro   241 Oct 31 21:31 .gtkrc-2.0
drwxr-xr-x 2 manjaro manjaro   60 Feb 20 13:41 .i3
drwxr-xr-x 3 manjaro manjaro   60 Feb 20 13:41 .icons
-rwxr-xr-x 1 manjaro manjaro 96502896 Apr 17 01:18 kakaotalk
drwxr-xr-x 3 manjaro manjaro   60 Feb 20 13:41 .local
drwxr-xr-x 2 manjaro manjaro   60 Feb 20 13:41 .moc
drwx------ 3 manjaro manjaro   60 Apr 17 15:45 '.moonchild productions'
drwx------ 3 manjaro manjaro   60 Apr 17 15:45 .mozilla
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Music
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Pictures
-rw-r--r-- 1 manjaro manjaro   204 Oct 31 21:31 .profile
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Public
drwxr-xr-x 2 manjaro manjaro   40 Apr 17 15:42 Templates
drwxr-xr-x 2 manjaro manjaro   60 Apr 17 15:45 .urxvt
-rw-r----- 1 manjaro manjaro   5 Apr 17 15:47 .vboxclient-clipboard.pid
Wine Mono 설치관리자
다운로드중...
취소
설치(I)
[manjaro@manjaro-i3 ~]$
"kakaotalk" may be a binary
[manjaro@manjaro-i3 ~]$ fi
kakaotalk: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3
.2.0, Go BuildID=rdd25L-jDuS8PfrfBO1q/c-pobcMvRv5XvDvEWcGw/OTd3reXLkf7AnF6h1DXL/5vNFFtIctuLcZ1mYovS3, B
uildID[sha1]=d55243824bbcd03bb04e7b26612724f9a93ddaee, not stripped
[manjaro@manjaro-i3 ~]$ sudo ./kakaotalk
Install wineboot
0012:err:ole:marshal_object couldn't get IPSFactory buffer for interface {00000131-0000-0000-c000-00000
0000046}
0012:err:ole:marshal_object couldn't get IPSFactory buffer for interface {6d5140c1-7436-11ce-8034-00aa0
06009fa}
0012:err:ole:StdMarshalImpl_MarshalInterface Failed to create ifstub, hres=0x80004002
0012:err:ole:CoMarshalInterface Failed to marshal the interface {6d5140c1-7436-11ce-8034-00aa006009fa},
 80004002
0012:err:ole:get_local_server_stream Failed: 80004002
0014:err:ole:marshal_object couldn't get IPSFactory buffer for interface {00000131-0000-0000-c000-00000
0000046}
0014:err:ole:marshal_object couldn't get IPSFactory buffer for interface {6d5140c1-7436-11ce-8034-00aa0
06009fa}
0014:err:ole:StdMarshalImpl_MarshalInterface Failed to create ifstub, hres=0x80004002
0014:err:ole:CoMarshalInterface Failed to marshal the interface {6d5140c1-7436-11ce-8034-00aa006009fa},
 80004002
0014:err:ole:get_local_server_stream Failed: 80004002
0017:fixme:urlmon:InternetBindInfo_GetBindString not supported string type 20
1 2
cpu 32%
1.0 GiB
lan: 10.0.2.15 1000 Mbit/s
100%
17.04. 15:48

과정

#TODO 이쁘게 수정

바이너리 실행 ->

컨테이너 스펙 생성 ->

바이너리 안에 미리 넣은 이미지를 압축 풀어서 rootfs를 만듬 ->

컨테이너 실행 ->

컨테이너 안에서 이 바이너리를 실행 이때 뒤에 인자 init을 붙여서 분기 ->

컨테이너 안에서 다시 이 바이너리가 실행되면서
위의 rootfs로 pivot하여 위의 컨테이너 스펙으로 실행

이를 구현한 주요 코드

```
package main

import (
    "flag"
    "os"
    "runtime"

    "github.com/leoh0/binctr/container"
    "github.com/opencontainers/runc/libcontainer"
    _ "github.com/opencontainers/runc/libcontainer/nsenter"
    specs "github.com/opencontainers/runtime-spec/specs-go"
    "github.com/sirupsen/logrus"
)

const (
    defaultRoot      = "/tmp/kakaotalk-binctr"
```

이런 방법외에도

coreOS

linuxkit

등과 같은 공격받을 범위를 줄이는
적은 공격 범위를 갖는 전문 컨테이너 호스트를 사용

결론

가능한 적은 root 권한을 이용하도록 함

가능한 적은 권한을 갖는 OS 기반에서 사용하도록 함

한계 #3

container는 VM과 달리 multi tenant를
위한 isolation이 적합하지 못하다.

multi tenant

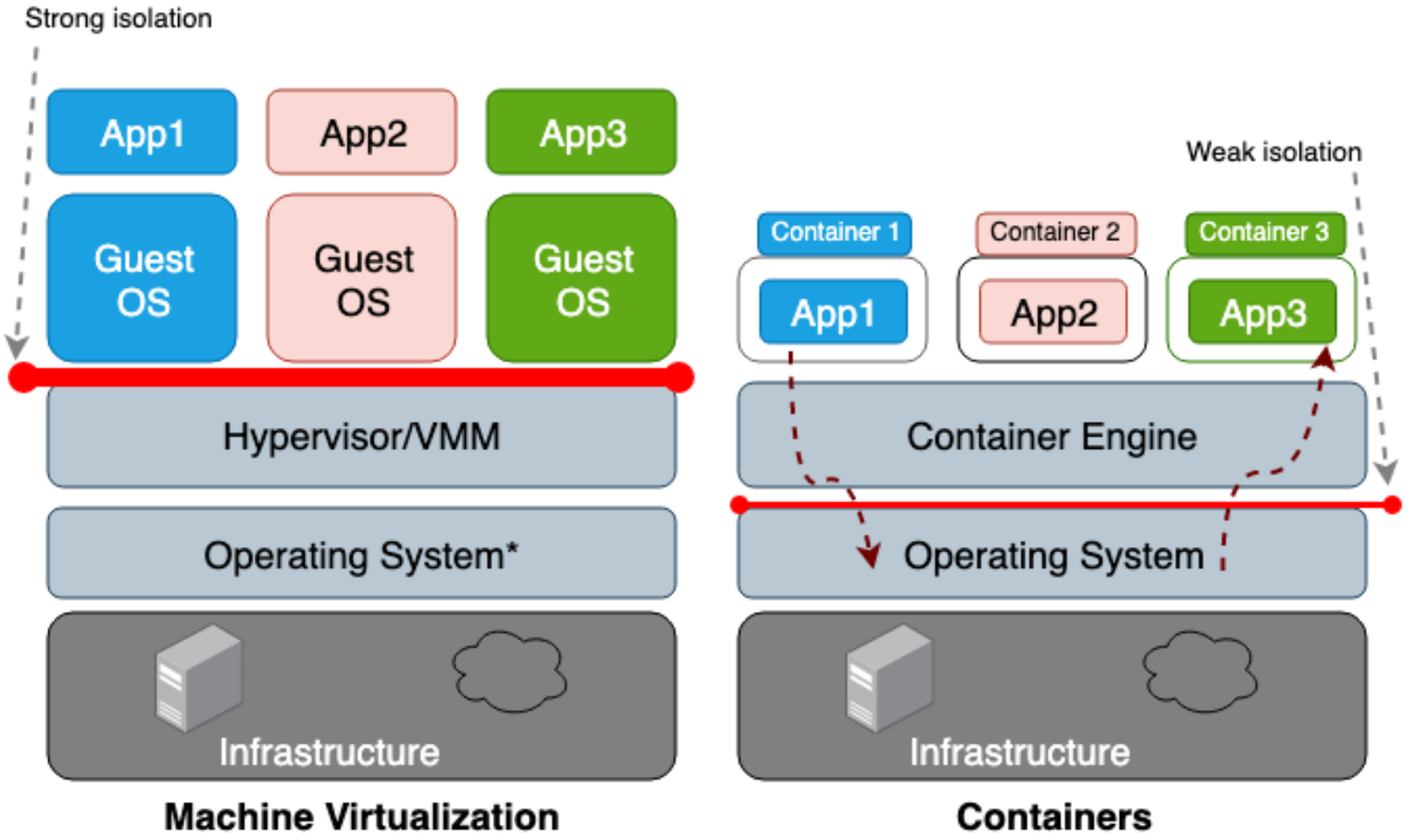
#TODO multi tenant가 왜 필요 한지?

kubesprawl

#TODO kubesprawl 에 대한 설명

https://tech.paulcz.net/blog/future-of-kubernetes-is-virtual-machines/

isolation 정도

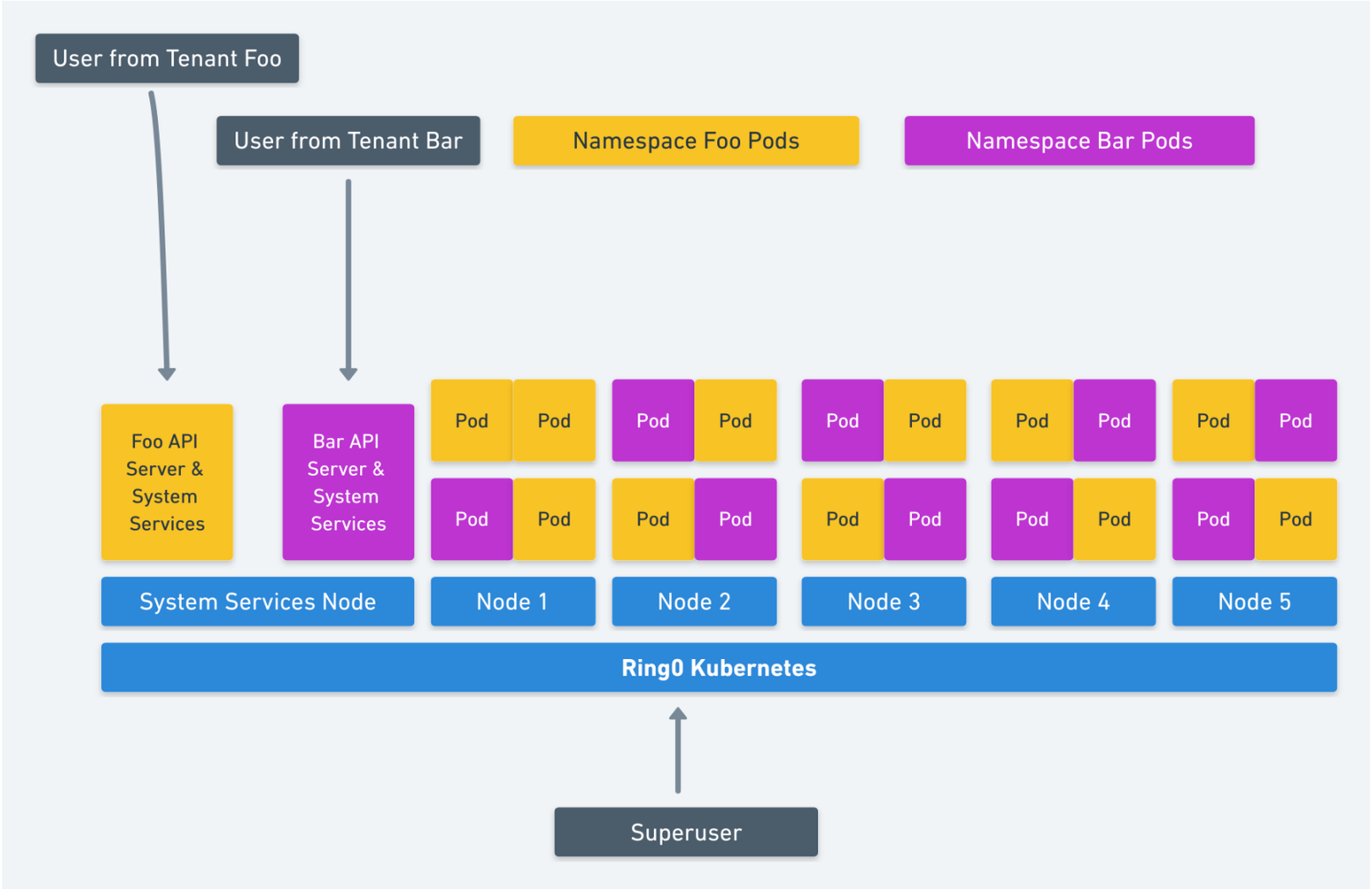


from making-containers-more-isolated-an-overview-of-sandboxed-container-technologies

kubernetes soft multi tenancy

#TODO namespace 소개 namespace

kubernetes hard multi tenancy



hard-multi-tenancy-in-kubernetes

sandboxed containers

a.k.a micro vm

	Supported container platforms	Dedicated guest kernel	Support different guest kernels	Open source	Hot-plug	Direct access to HW	Required hypervisors	Backed by
Nabla	Docker, K8s	Yes	Yes	Yes	No	No	None	IBM
gVisor	Docker, K8s	Yes	No	Yes	No	No	None	Google
Firecracker	Not yet	Yes	Yes	Yes	No	No	KVM	Amazon
Kata	Docker, K8s	Yes	Yes	Yes	Yes	Yes	KVM or Xen	OpenStack

from making-containers-more-isolated-an-overview-of-sandboxed-container-technologies

결론

기존의 VM의 강점을 가저가면서도
container의 최적화된 장점들을 가져갈 수 있는 방법들이
여러가지로 경쟁적으로 나오고 있는 상태

마지막으로


James Watters
@wattersjames
The hot new container technology is VMs
197 3:48 PM - Nov 27, 2018
37 people are talking about this

감사합니다.