Docker로 보는 서버 운영의 미래

The Future of System Engineering

2014 Deview

2014. 9. 29.
2015. 8. 29. update

Daekwon Kim

SMARTSTUDY Software & System Engineer

@nacyo_t

가 각 간 갓

나 낙 난 낫

다 닥 단 닷

ㄱ ㄴ ㅅ

(ㄱ,ㄴ,ㅅ)x(가,나,다)

낚읏

Deployment 그리고 Docker

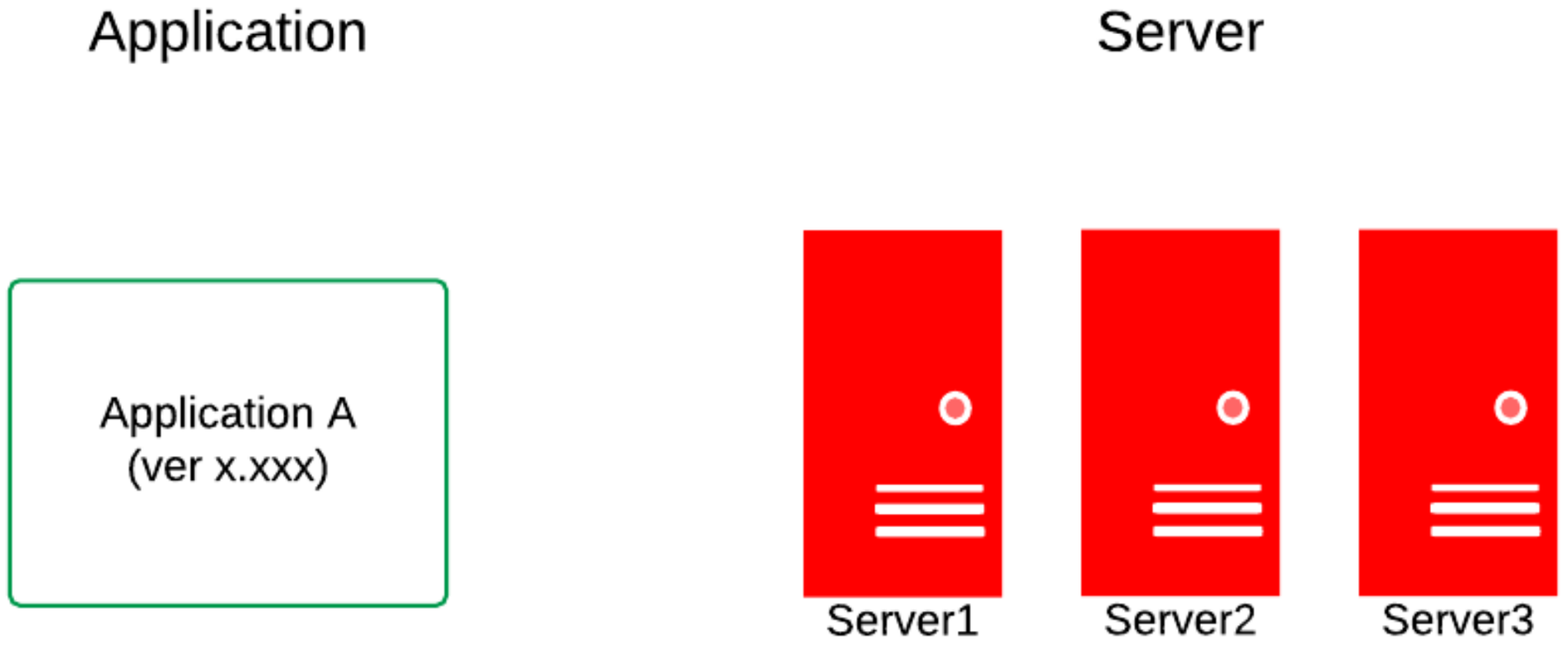
TOC

State (상태)

사물 · 현상이 놓여 있는 모양이나 형편

1. 배포와 서버운영

배포의 요소


Application
Application A
(ver x.xxx)
Server
Server1
Server2
Server3

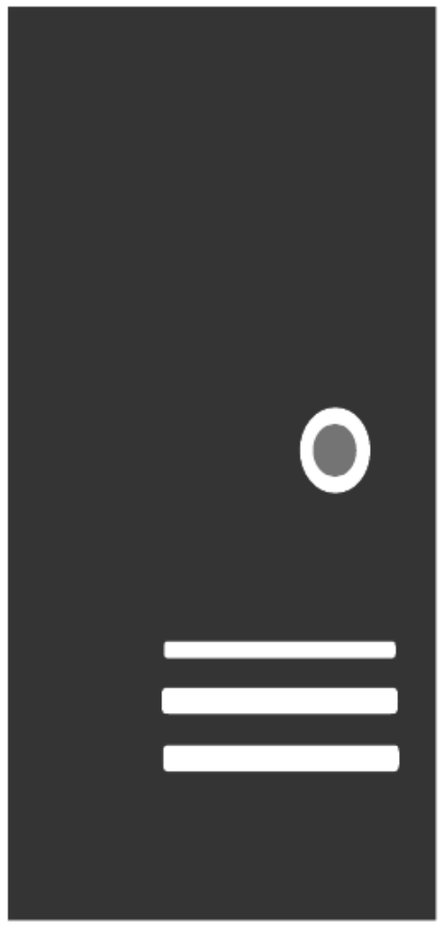
배포란?

대상 서버를 어플리케이션이 실행 가능한 상태로 만드는 일

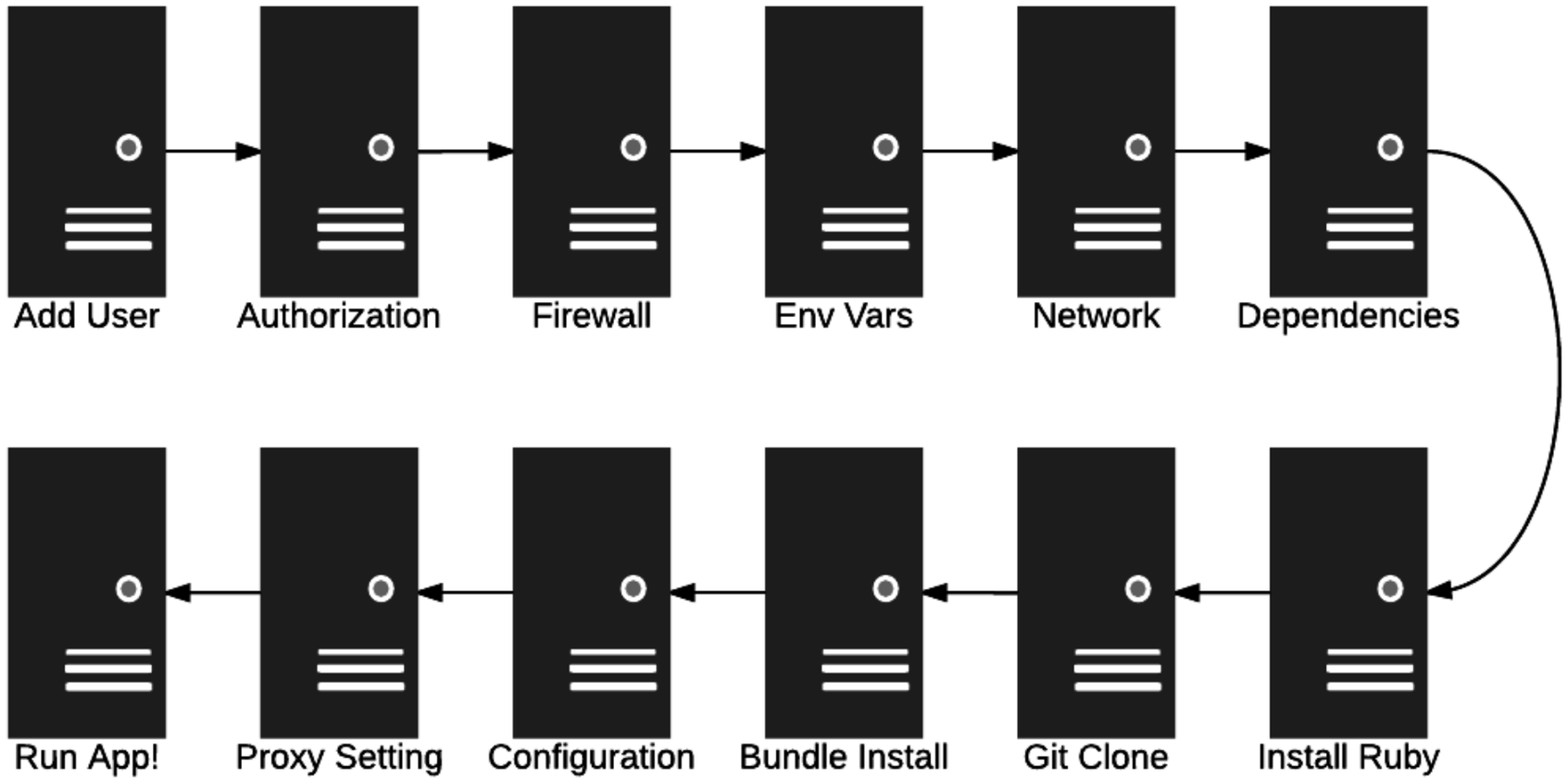
예제

Ruby on Rails 어플리케이션

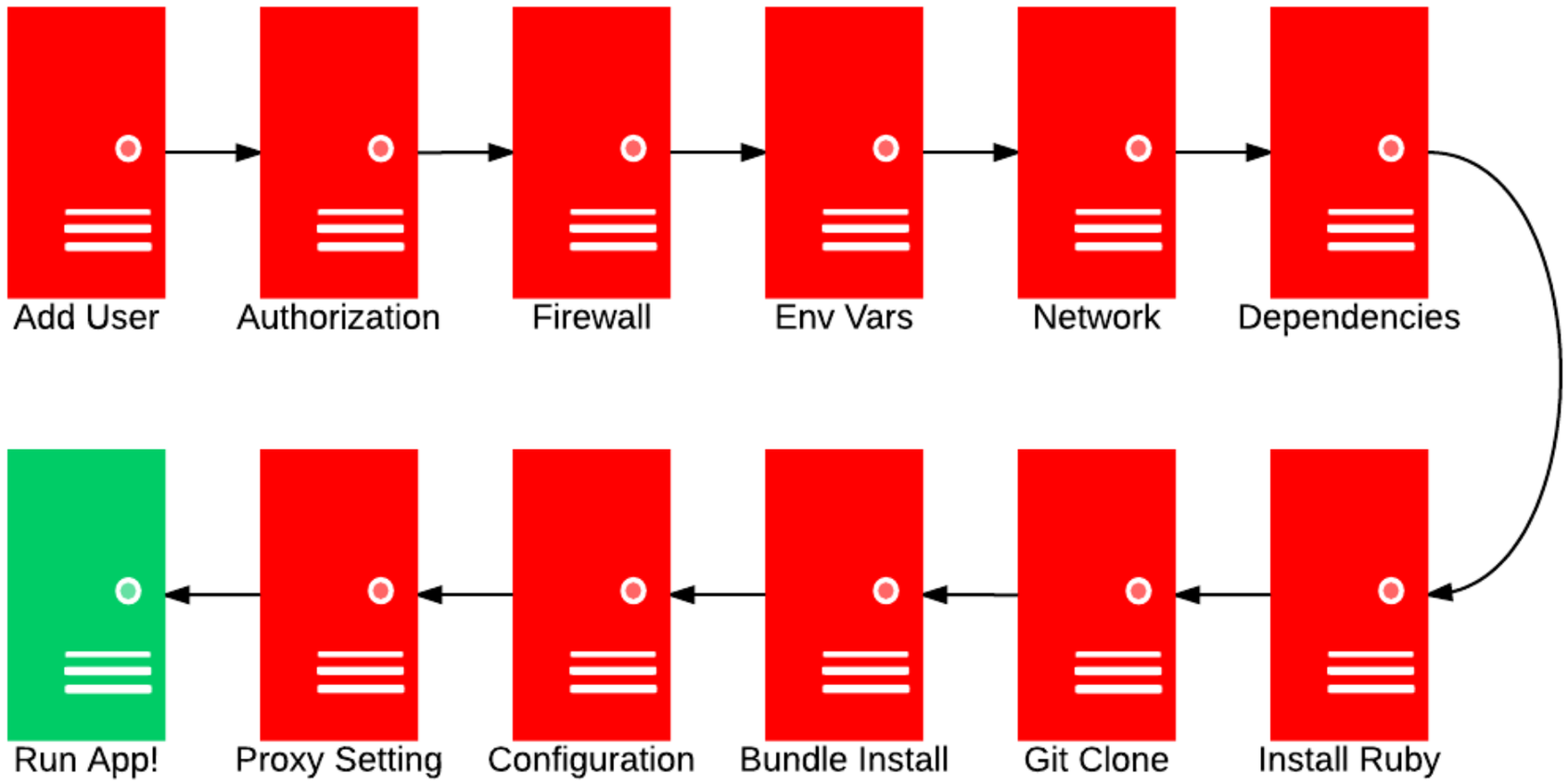
하나의 서버



배포 과정

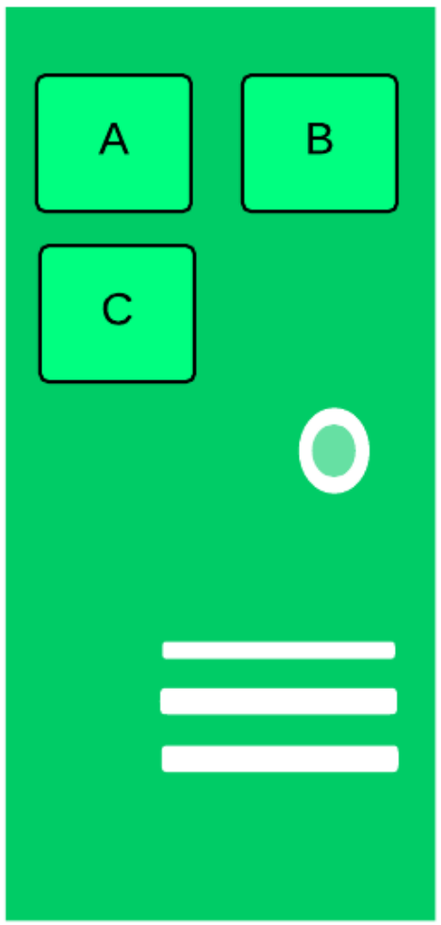

Add User
Authorization
Firewall
Env Vars
Network
Dependencies
Install Ruby
Git Clone
Bundle Install
Configuration
Proxy Setting
Run App!

배포 과정에 따른 상태 변화

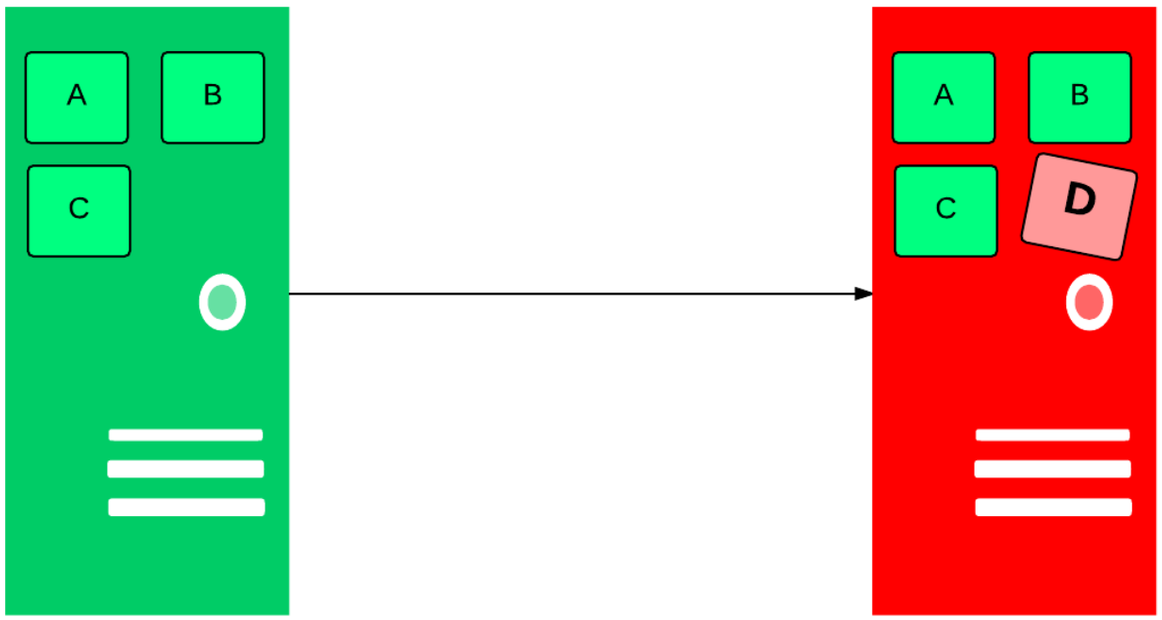


더 복잡한 상황

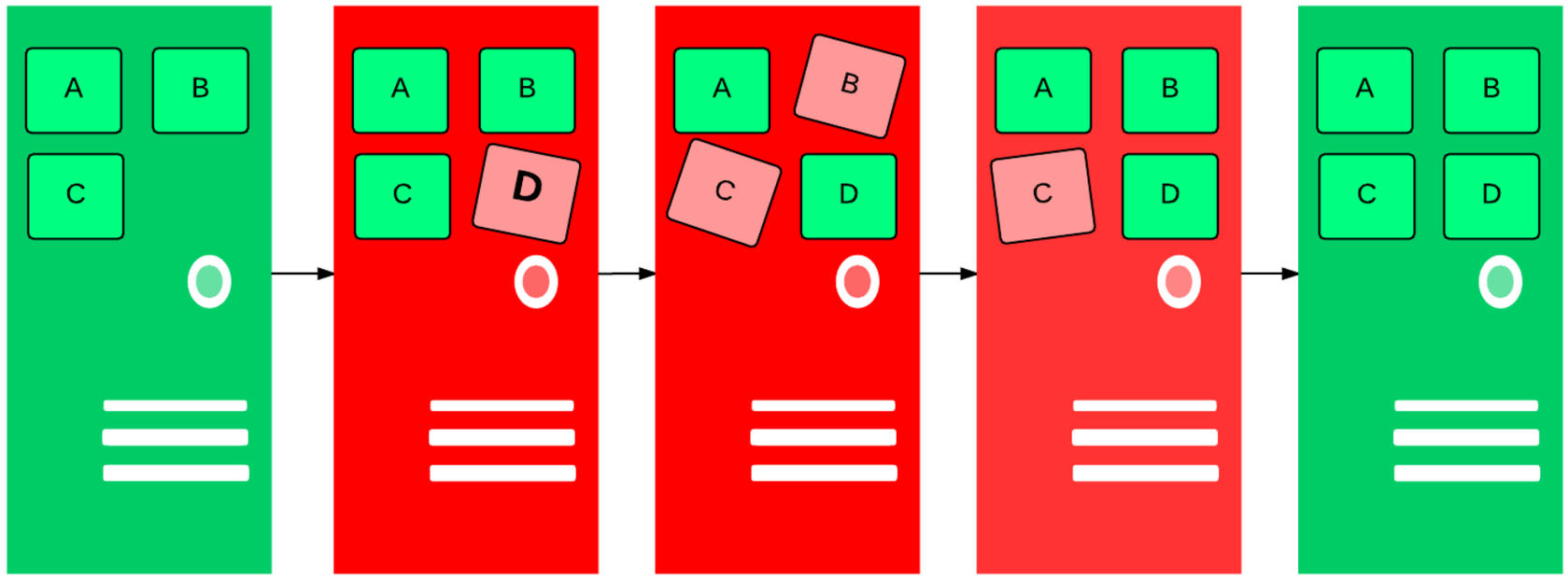
다수의 어플리케이션



새로운 어플리케이션

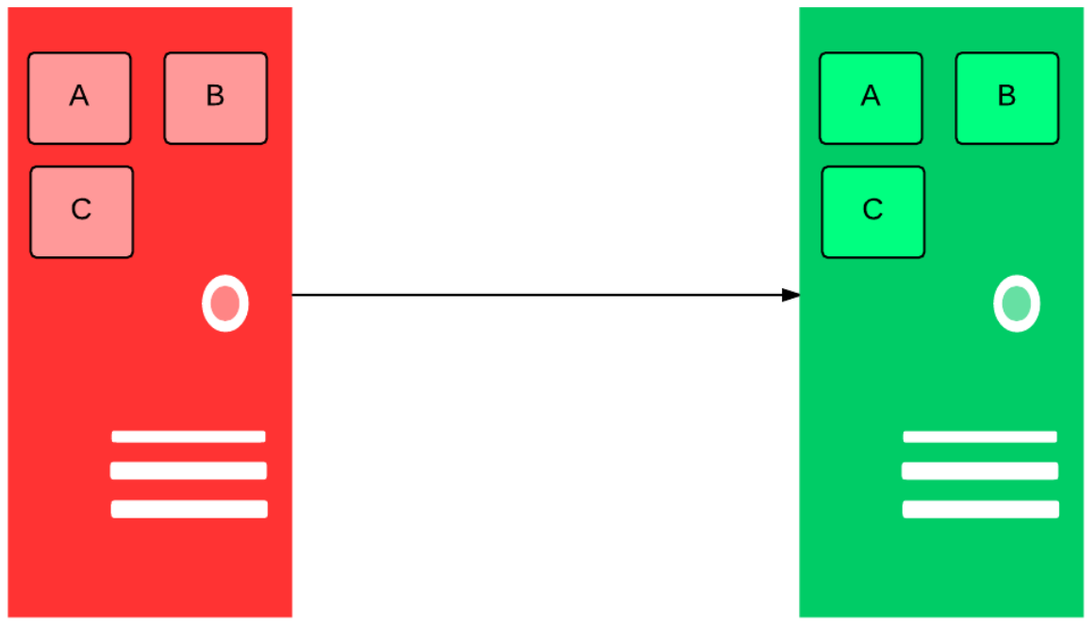


배포 과정 (정상화)



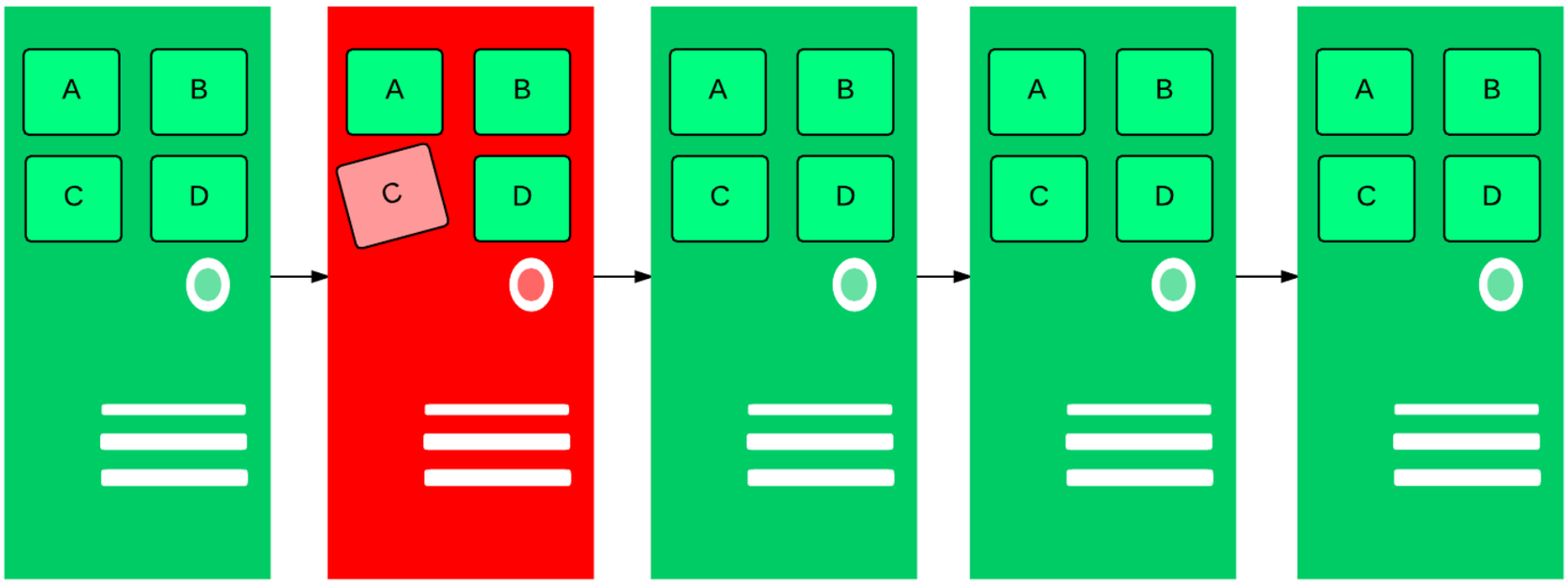
배포

어플리케이션들이 모두 실행 가능한 상태로 만드는 일



서버 운영

모든 어플리케이션이 동작하는 상태를 유지하는 일



서버 운영이 어려운 이유

서버 운영 = 전역적 상태 관리

카드로 만든 집

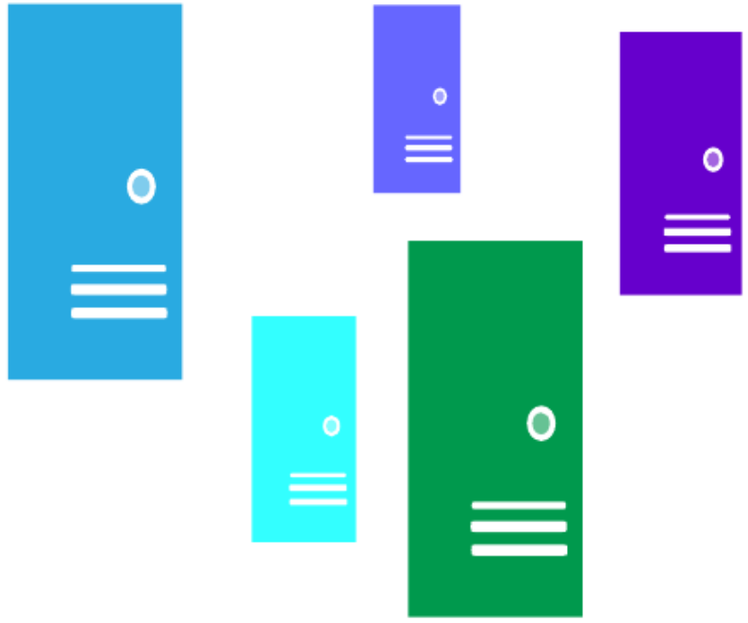
The system becomes a house of cards. You fear any change and you fear replacing it since you don't know everything about how it works.

– Chad Fowler

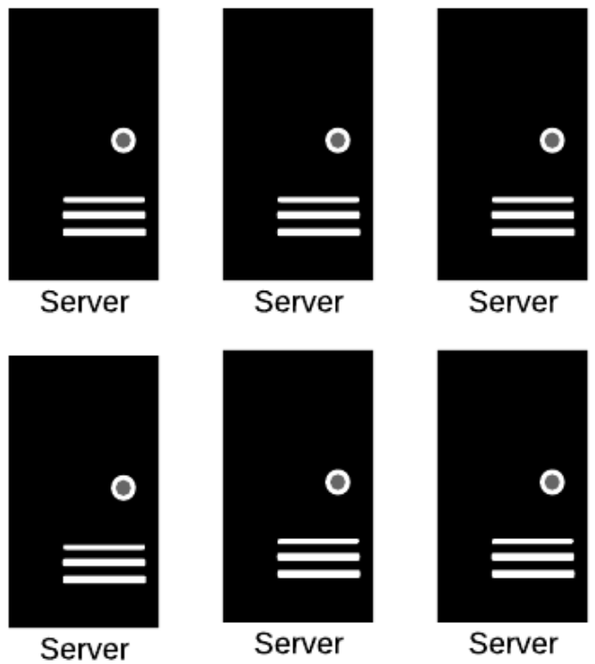
클라우드

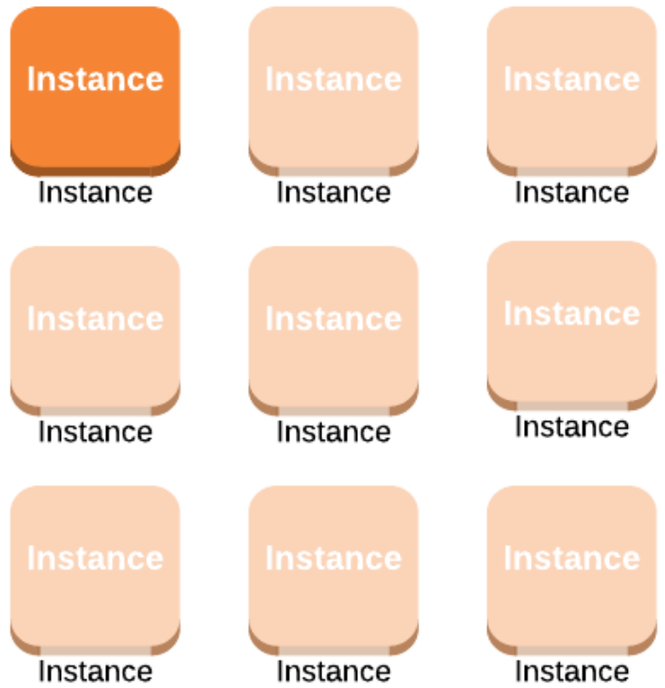
2. Infrastructure as a Service

같은 환경을 가지는 가상화된 환경



원하는 만큼 사용 가능한 컴퓨팅 자원





Disposable Component

- 서버는 물리적으로 고정된 자원
- 인스턴스는 쓰고 버리는 자원

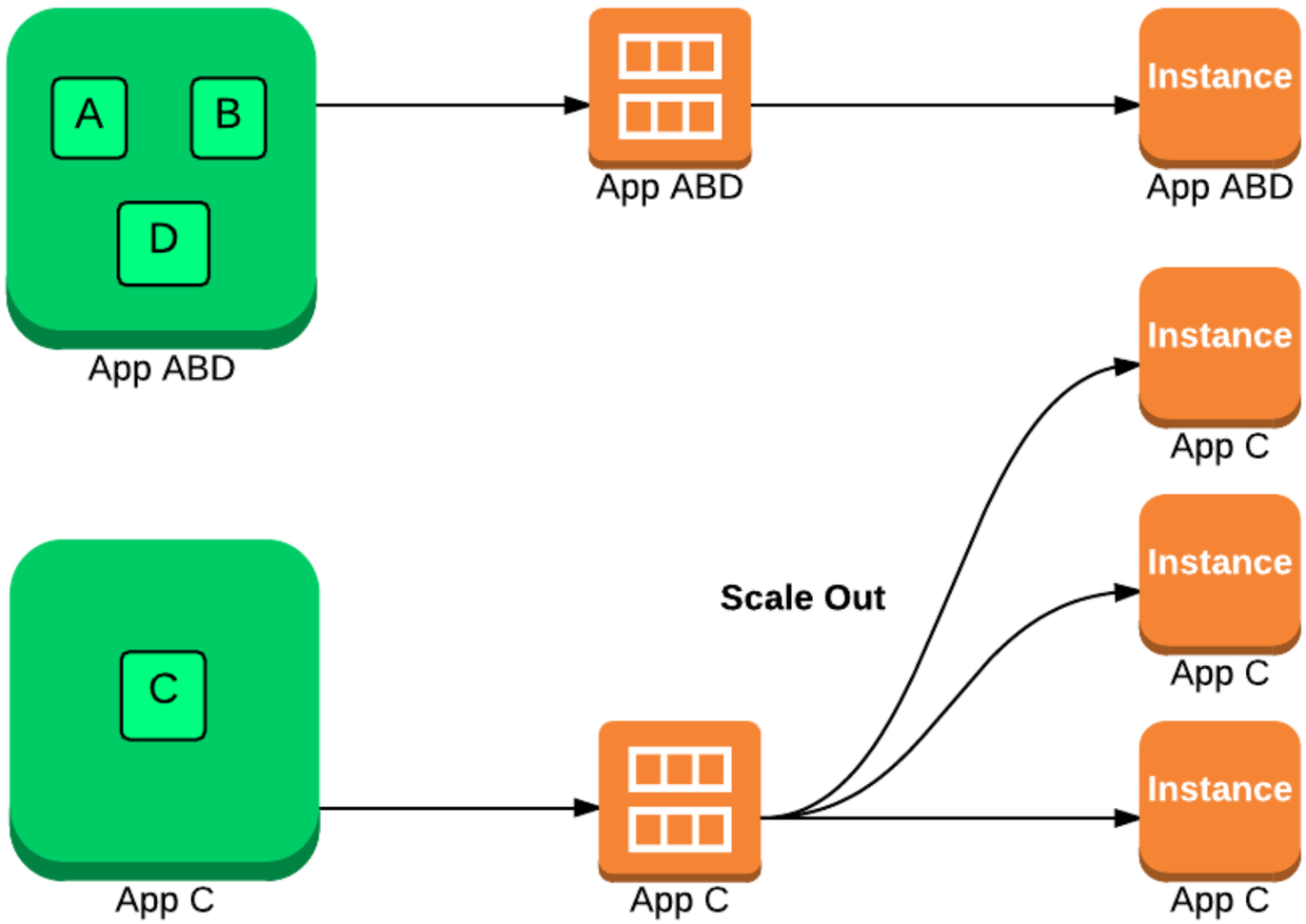
인터넷의 발전소

*"Amazon Web Service*는 인터넷의 발전소다

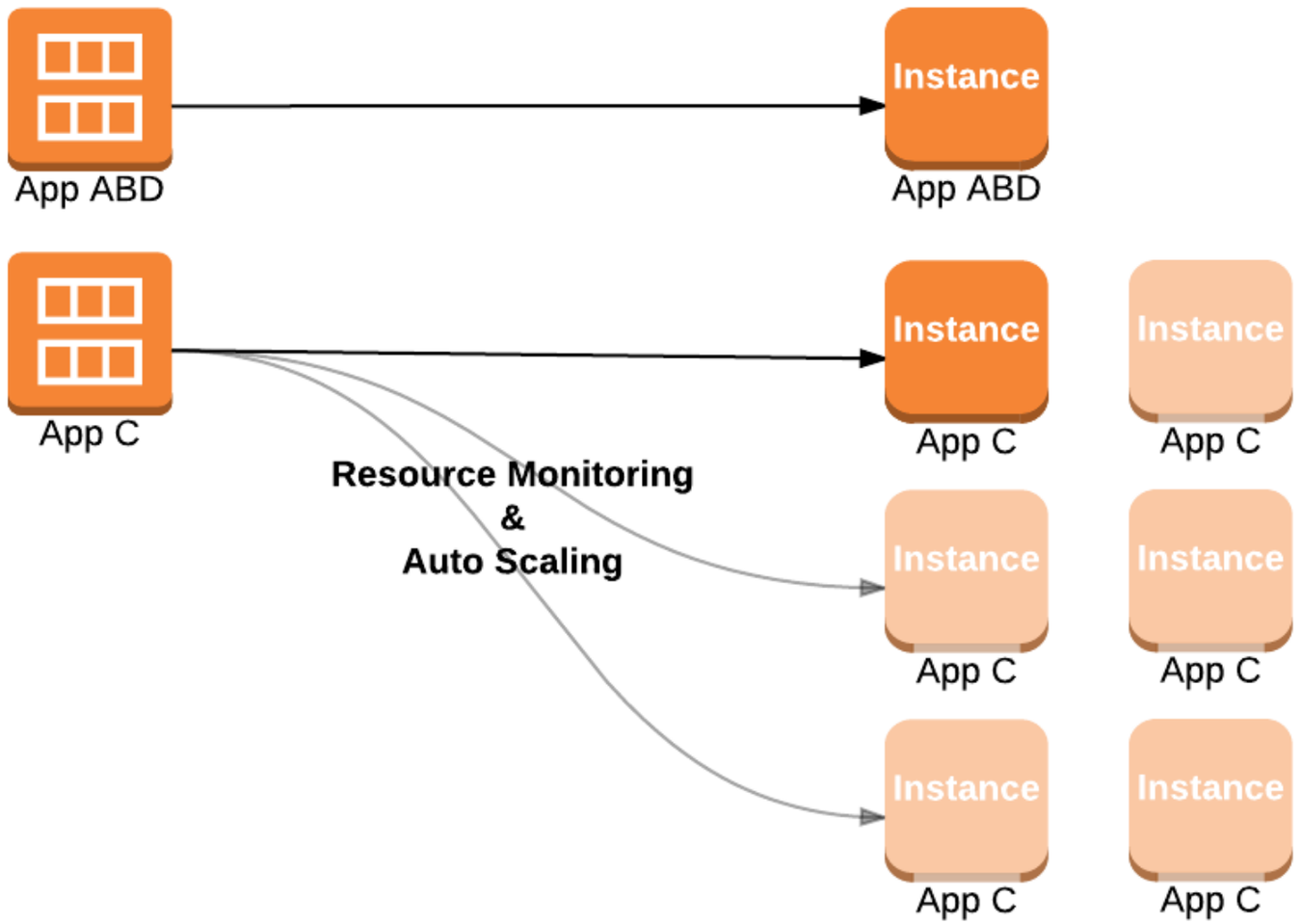
– 타마카와 켄

이미지

이미지화 & 스케일 인/아웃


A
B
D
App ABD
App ABD
Instance
App ABD
Instance
App C
Instance
App C
Scale Out
C
App C
App C
Instance
App C

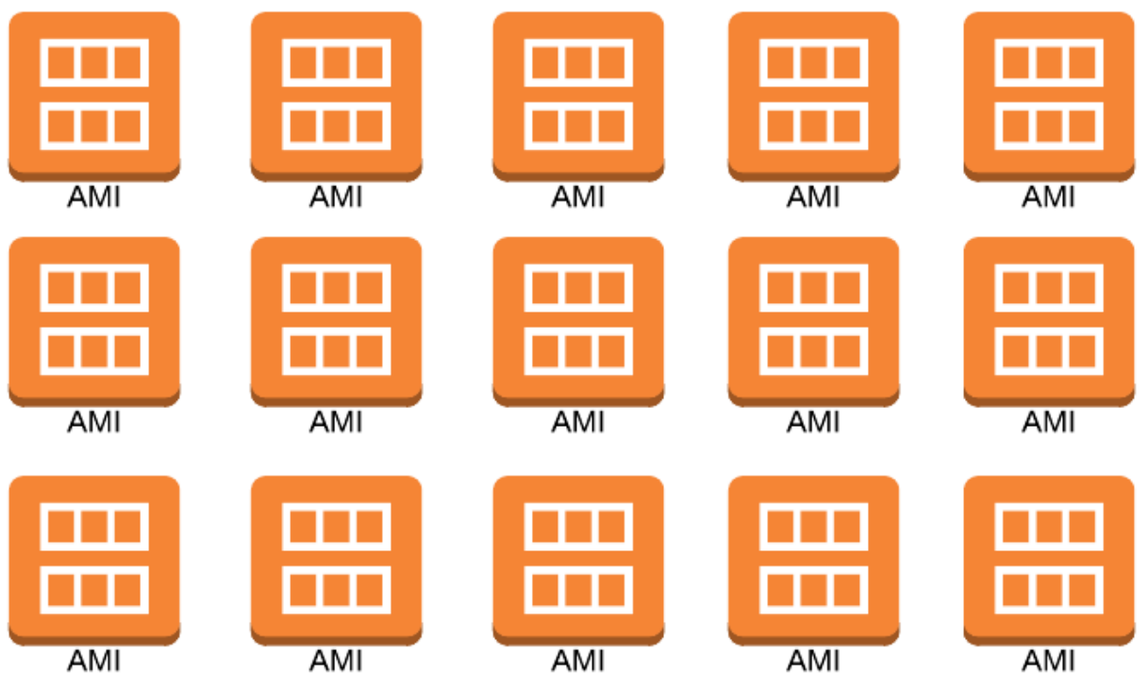
오토 스케일링


App ABD
Instance
App ABD
App C
Instance
App C
Instance
App C
Resource Monitoring
&
Auto Scaling
Instance
App C
Instance
App C
Instance
App C
Instance
App C

이미지 관리

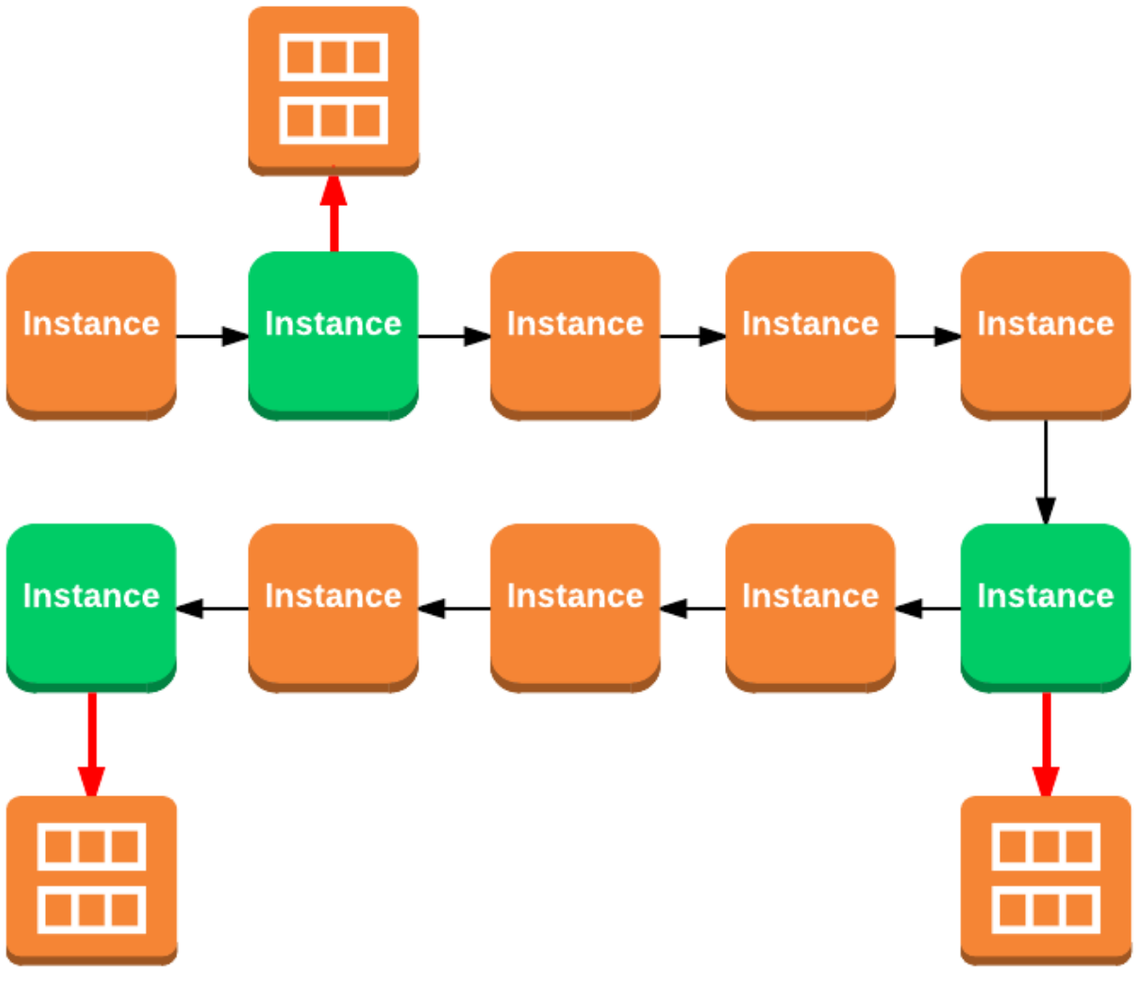
이미지 수의 증가

이미지 수에 비례해서 관리가 어려워짐
(어플리케이션 수 * 시간)



상태에 기반한 이미지

이미지는 어떤 특정 시점의 상태를 저장한 것 뿐



최종적으로 남은 이미지

어떻게 만들었는지 아무도 모름 (복원 불가능)

이미지 관리

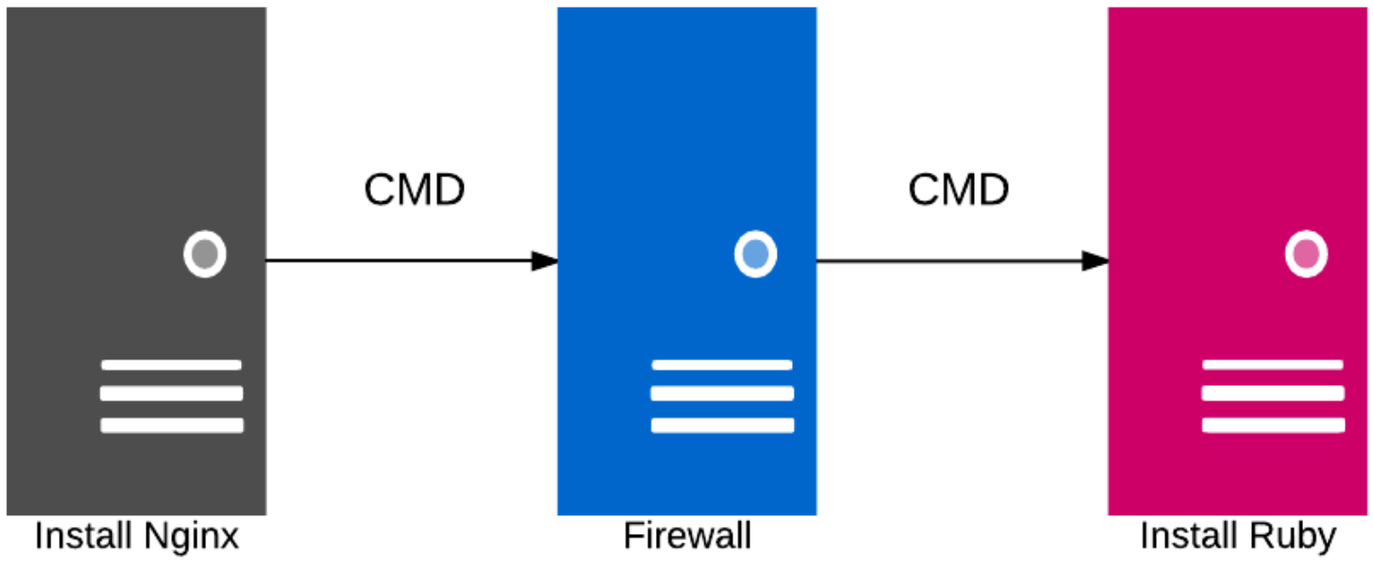
- 다양한 시점의 상태 관리
- 어떻게 하면 상태를 더 잘 관리할 수 있을까?
- 이미지 / 인스턴스 활용의 극대화

Configuration Management

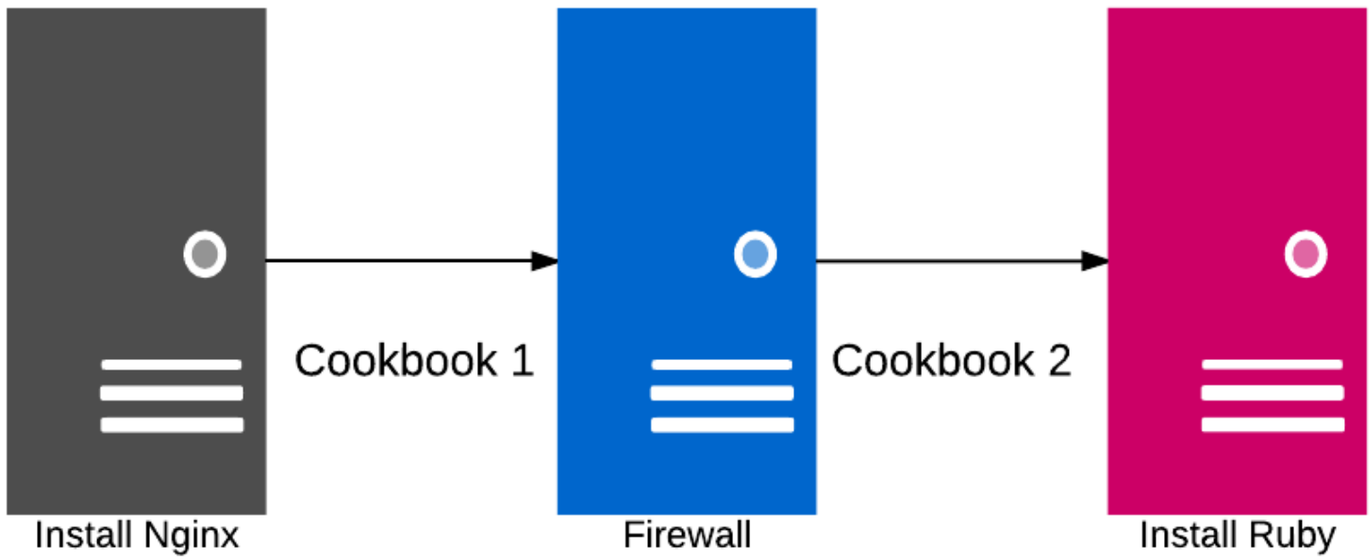
CHEF™

ANSIBLE

명령어를 통한 상태 변화(과거)

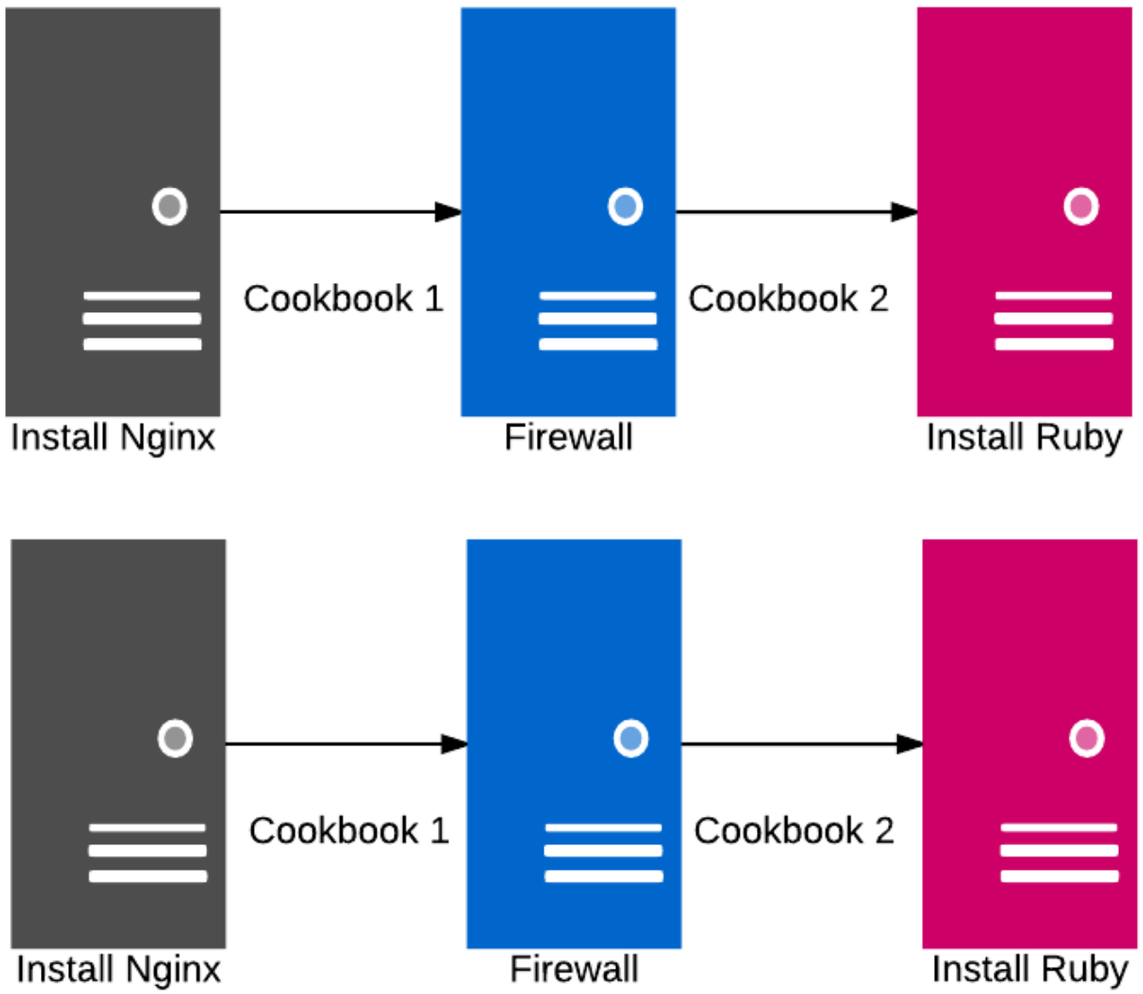


Cookbook을 통한 상태 변화(Chef)

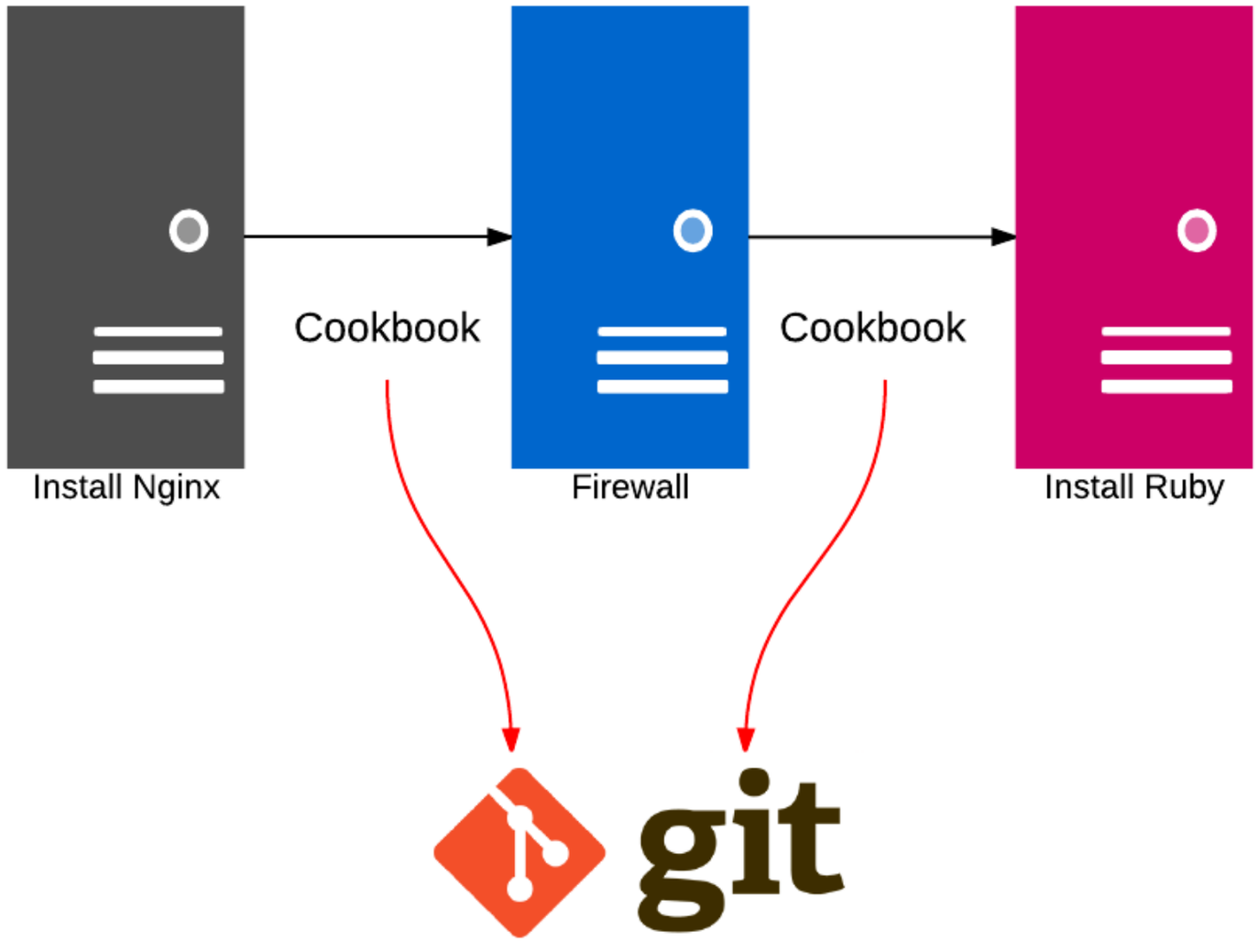


멱등

같은 과정을 거치면 서버는 똑같은 상태가 된다



Infrastructure의 코드화



Apache Cookbook 예제

CMD

```
$ apt-get install apache2
```

Chef Cookbook

```
package 'httpd'

service 'httpd' do
  action [:start, :enable]
end
```

Configuration Management 효과 (1)

- 초기 적용이 어려움
  - 학습 비용
  - 많은 시행착오가 필요
  - 작업 속도가 느림

Configuration Management 효과 (2)

- 서버 운영를 단순화
  - 서버의 상태를 코드로 관리
  - 어떤 상태를 재현 가능하게 만들어줌
  - 다수의 서버를 운용중에 특정 상태로 변화시킬 수 있음

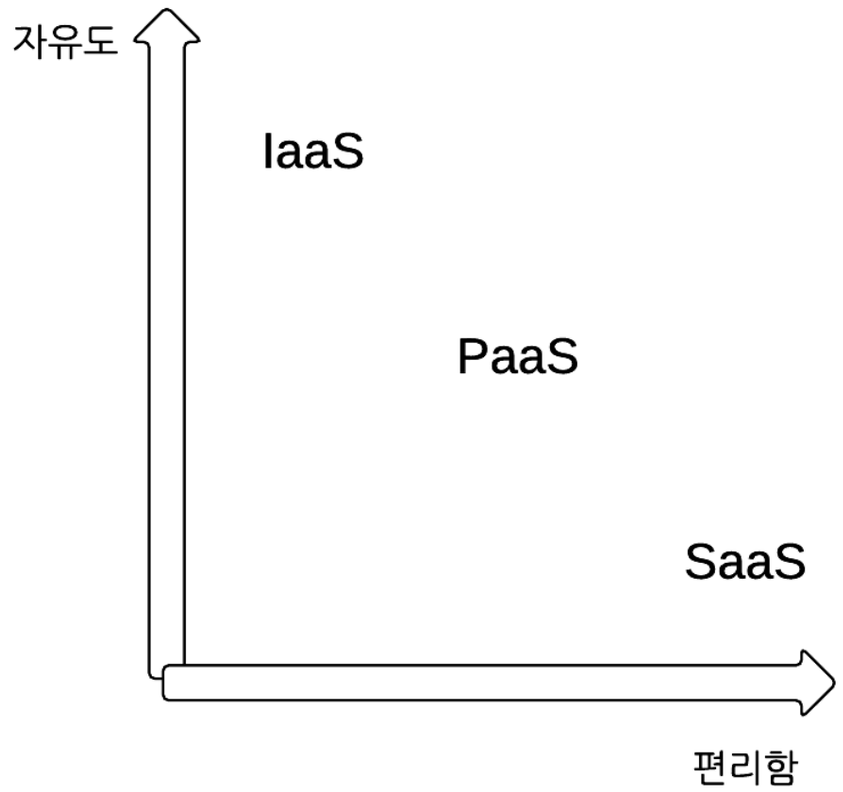
Configuration Management

상태를 관리하는 도구

3. Platform as a Service

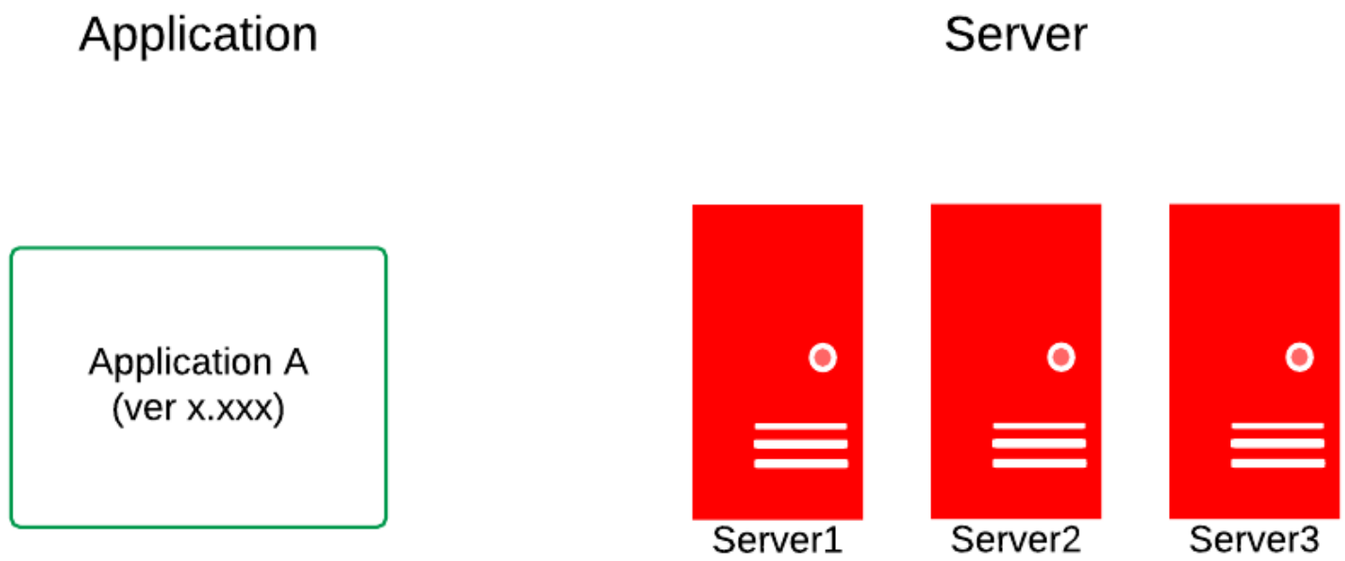
PaaS

자유도는 낮지만, 좀 더 편리한?

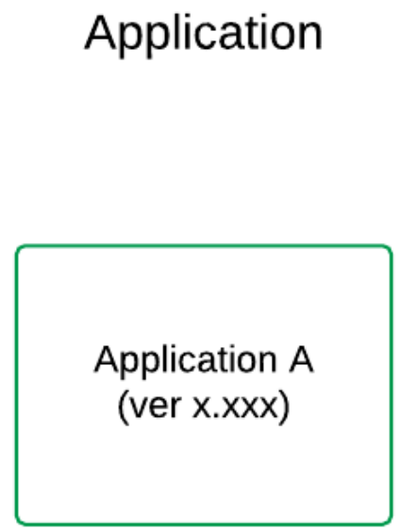



heroku

배포의 요소(과거)

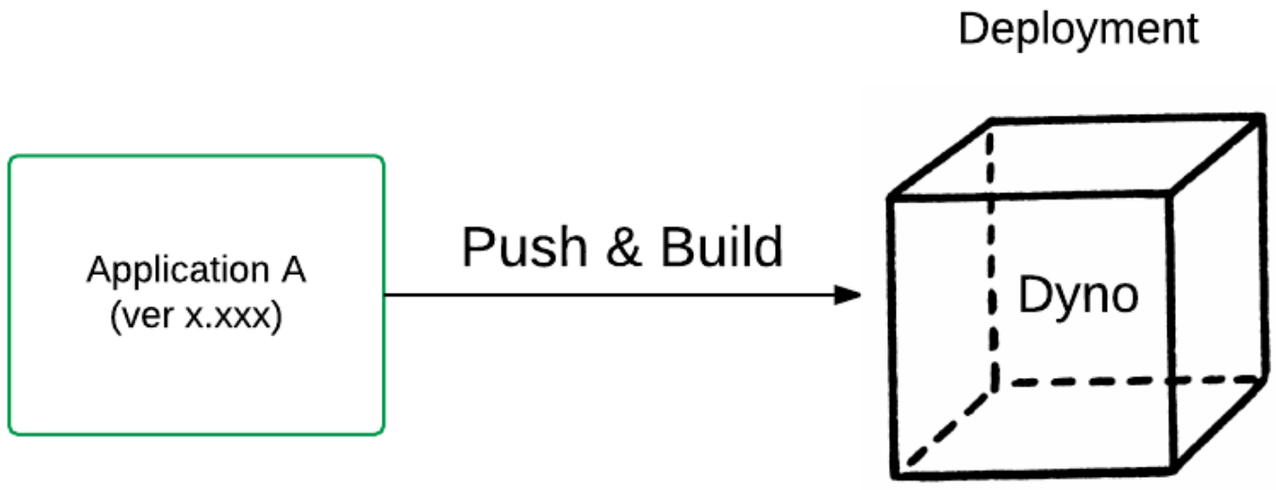

Application
Server
Application A
(ver x.xxx)
Server1
Server2
Server3

배포의 요소(PaaS)



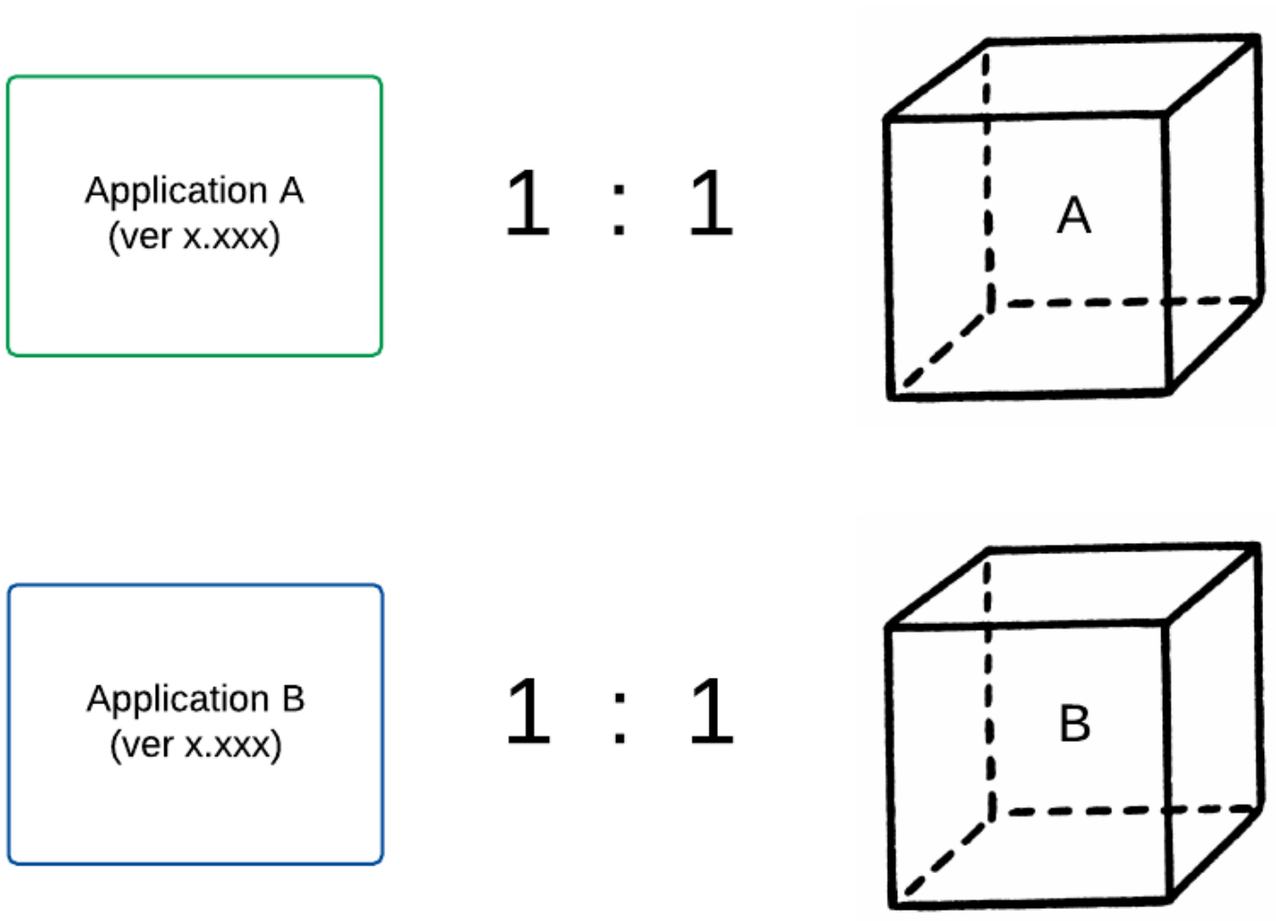
어플리케이션 배포 단위

배포 환경은 Heroku가 제공

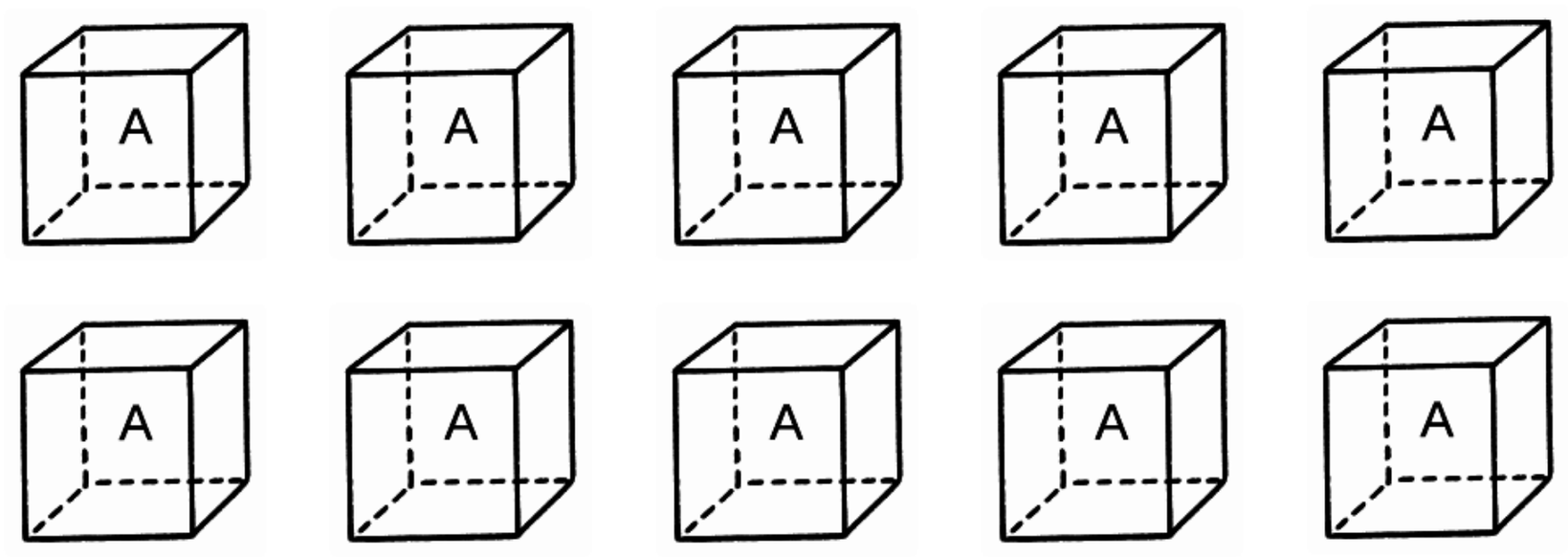


어플리케이션과 배포 단위의 1:1 매치

어플리케이션 별 독립된 배포 환경 제공

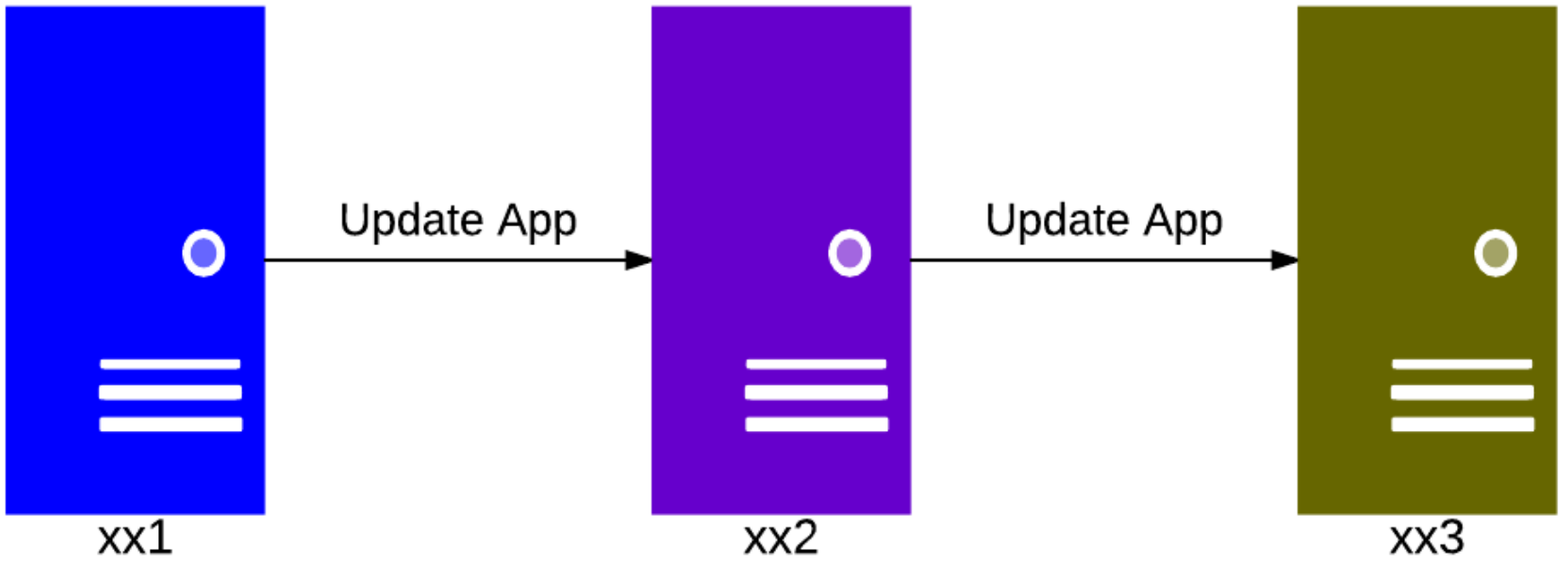


Scale In / Out 기본 지원

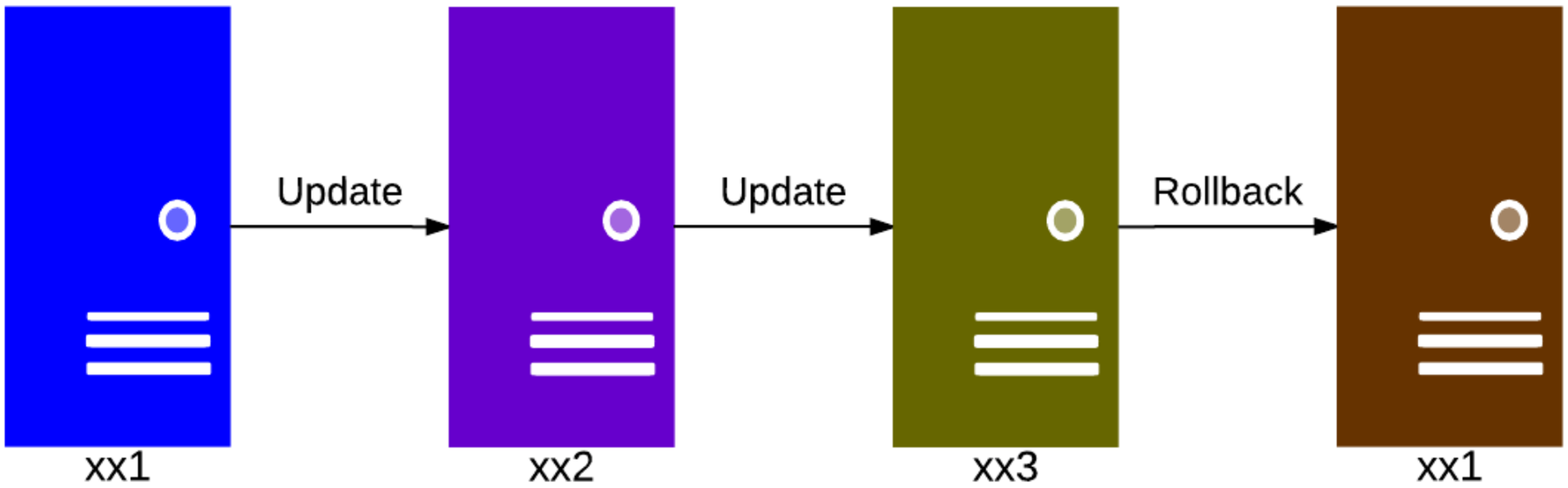


어플리케이션 업데이트

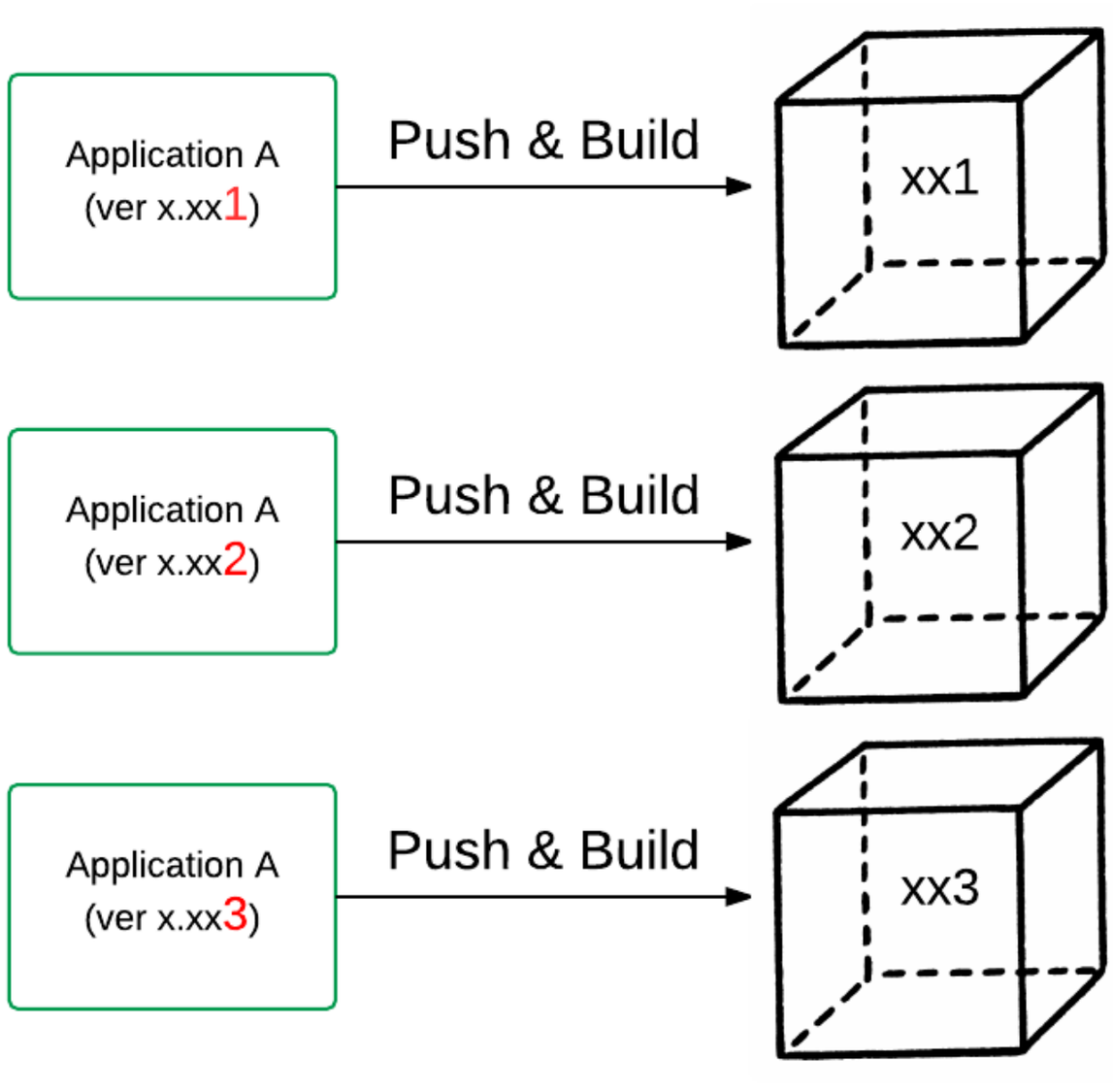
기존의 어플리케이션 업데이트



기존의 어플리케이션 롤백



Heroku에서의 어플리케이션 업데이트

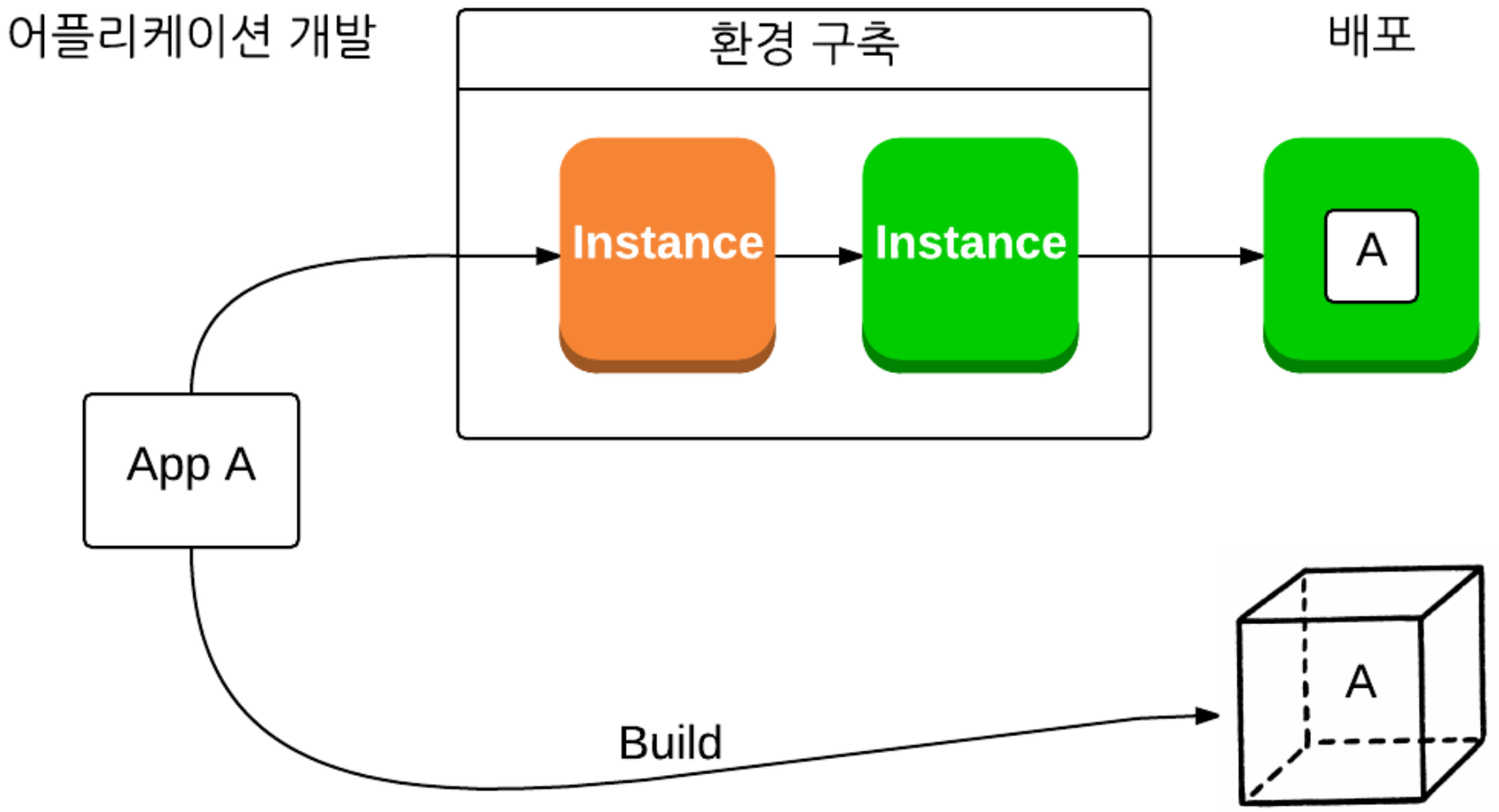


Heroku에서의 롤백

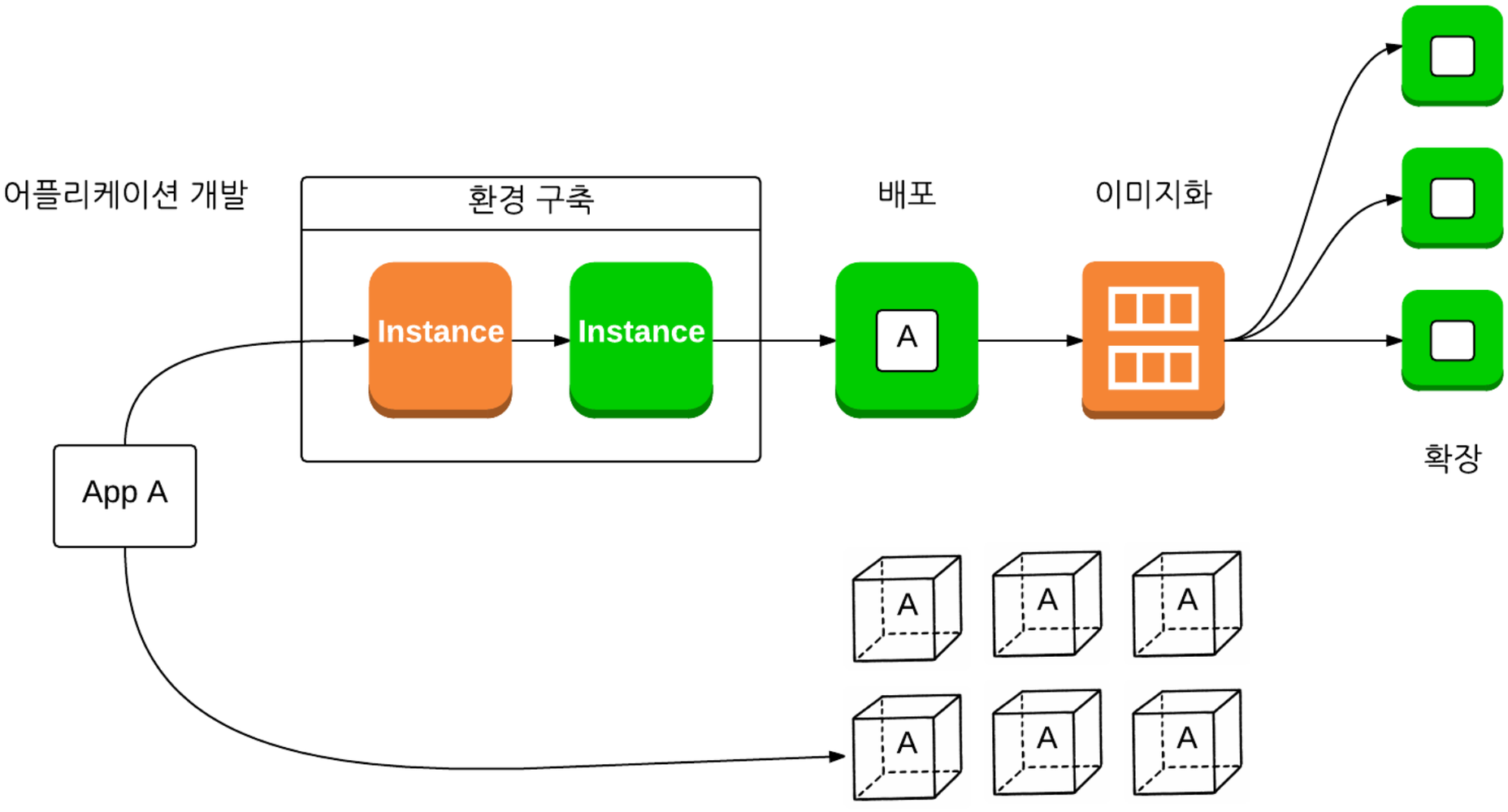
```
$ heroku releases
Rel   Change                 By                  When
----  ---------------------  ----------          ----------
v52   Config add AWS_S3_KEY  jim@example.com      5 minutes ago
v51   Deploy de63889         stephan@example.com  7 minutes ago
v50   Deploy 7c35f77         stephan@example.com  3 hours ago
v49   Rollback to v46        joe@example.com      2010-09-12 15:32:17 -0700
...
```

```
$ heroku rollback v40
```

배포의 단순화



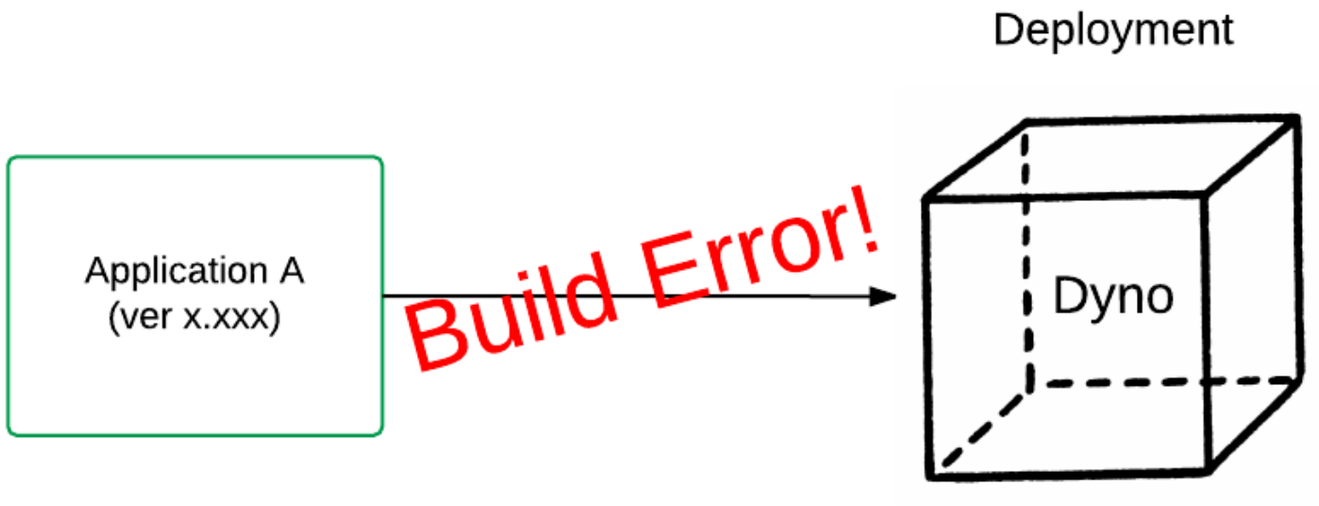
확장의 단순화



PaaS를 사용하지 않는 이유

비용

PaaS스럽지 않은 어플리케이션



PaaS != IaaS

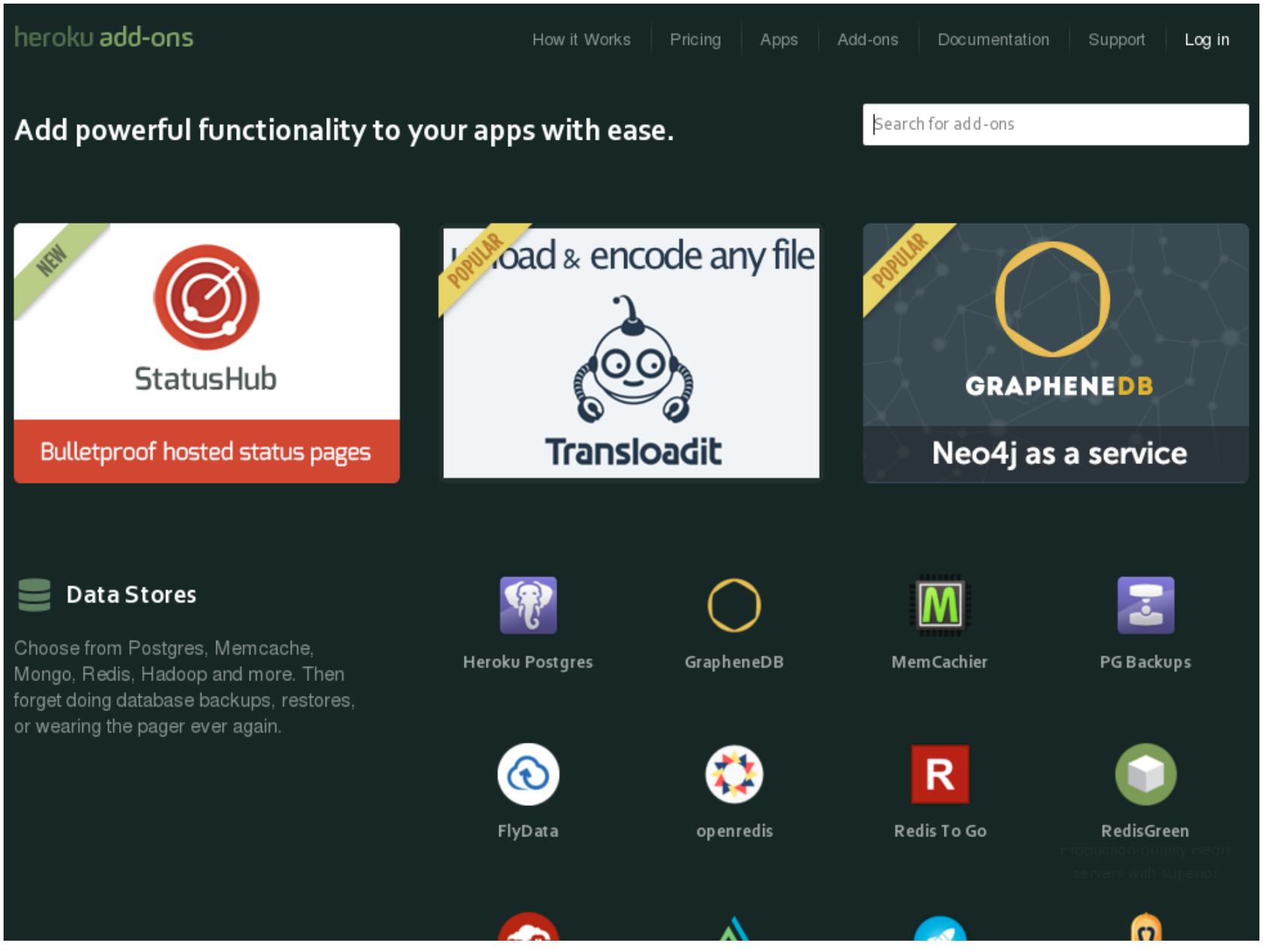
- IaaS에 비해서 큰 학습 비용
- 원격 접속 시스템이 없거나 제한
- 파일 시스템 이용에 제한
- Site 패키지 설치 제한
- 로그 수집 제한적 허용 (STDOUT)

문제를 해결하는 법이 다름

파일 시스템

- 외부 Storage 서비스를 사용

애드온 (외부 서비스)

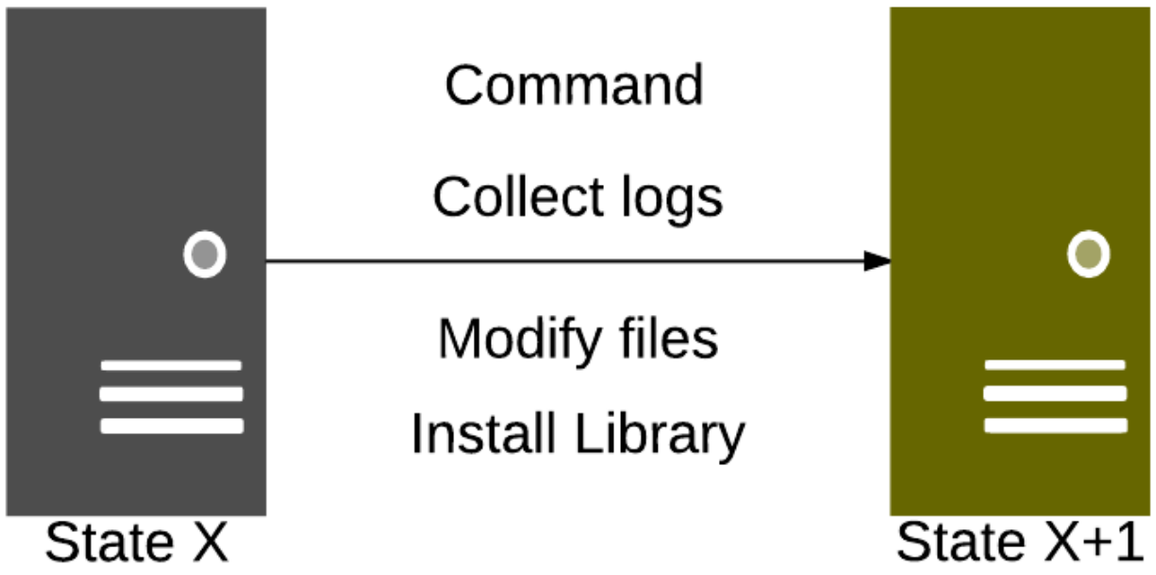

heroku add-ons
How it Works
Pricing
Apps
Add-ons
Documentation
Support
Log in
Add powerful functionality to your apps with ease.
Search for add-ons
NEW
StatusHub
Bulletproof hosted status pages
POPULAR
Transloadit
POPULAR
GRAPHENEDB
Neo4j as a service
Data Stores
Choose from Postgres, Memcache, Mongo, Redis, Hadoop and more. Then forget doing database backups, restores, or wearing the pager ever again.
Heroku Postgres
GrapheneDB
MemCachier
PG Backups
FlyData
openredis
Redis To Go
RedisGreen

4. 컨테이너형 어플리케이션 가상화

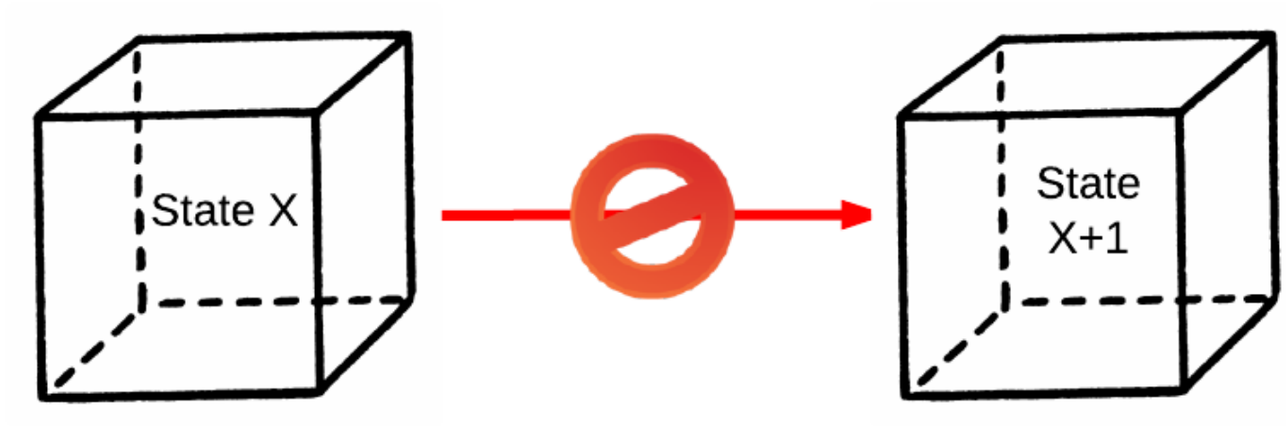
PaaS의 단점?

- 원격 접속 시스템이 없거나 제한
- 파일 시스템 이용에 제한
- Site 패키지 설치 제한
- 로그 수집 제한적 허용 (STDOUT)

상태 변화에 의존한 작업


Command
Collect logs
Modify files
Install Library
State X
State X+1

Immutable(Stateless)



뒤집어보면 컨테이너형 가상화의 특징

Heroku Dyno

LinuX Container (LXC)
Chroot on steroid

Chroot (Change Root)

- pivot root 기능
- 파일, 라이브러리는 직접 준비
- 사용이 까다로움
- 프로세스 격리의 초기 버전?

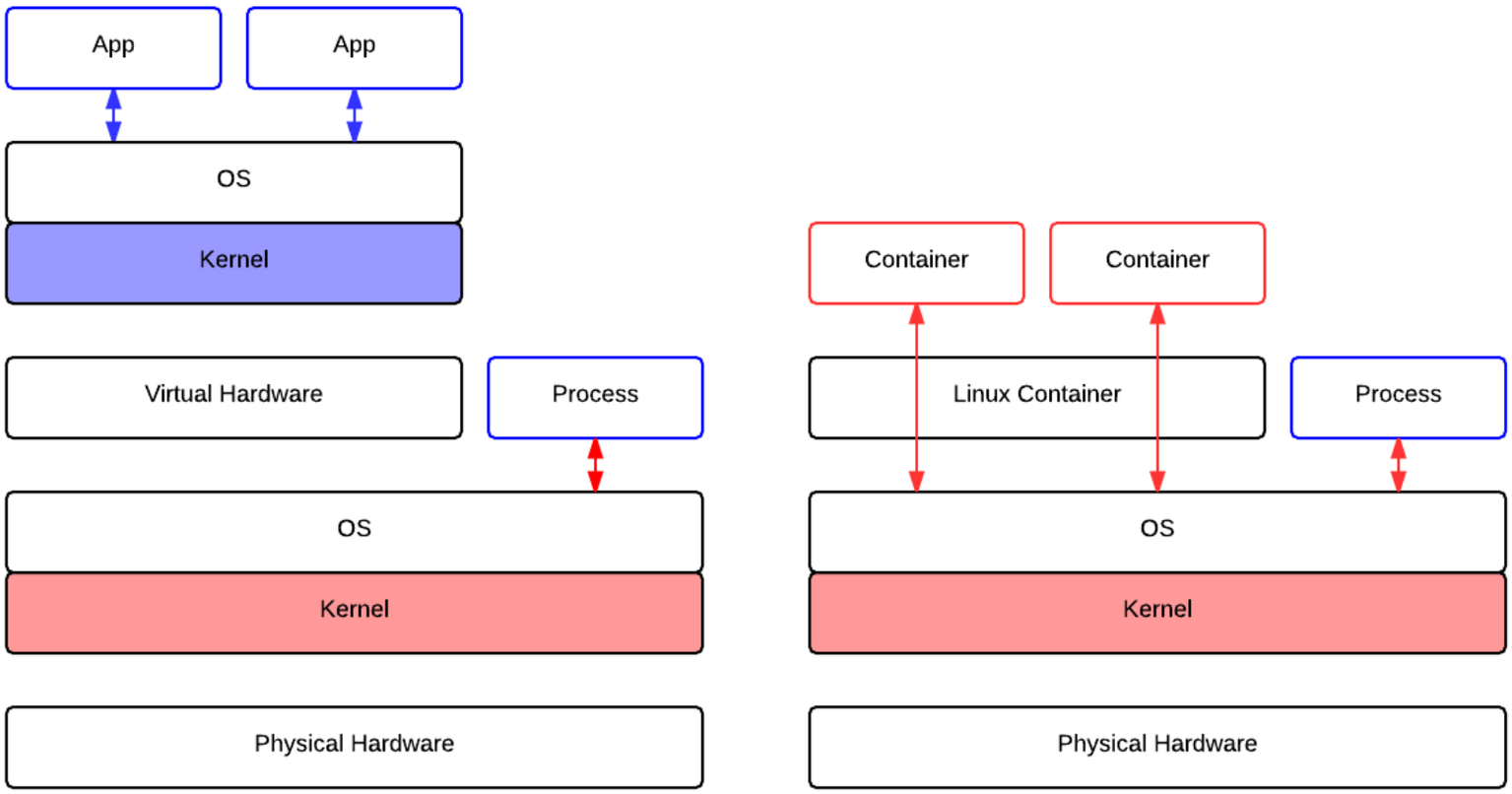
LinuX Container

- Kernel Namespaces
- Apparmor and SELinux profiles
- Seccomp policies
- *Chroots (using* `pivot_root`*)*
- Kernel capabilities
- Control groups (cgroups)

예제 1) 가상 머신?

```
nacyot $ sudo lxc-start -n centos

CentOS release 6.5 (Final)
Kernel 3.11.0-18-generic on an x86_64

centos login: init: rcS main process (7) killed by TERM signal
Entering non-interactive startup
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:
Determining IP information for eth0... done.
                                                           [  OK  ]
Starting system logger:                                    [  OK  ]
awk: cmd. line:1: fatal: cannot open file `/etc/mtab' for reading (No
such file or directory)
Mounting filesystems:                                      [  OK  ]
Generating SSH1 RSA host key:                              [  OK  ]
Generating SSH2 RSA host key:                              [  OK  ]
Generating SSH2 DSA host key:                              [  OK  ]
Starting sshd:                                             [  OK  ]
```

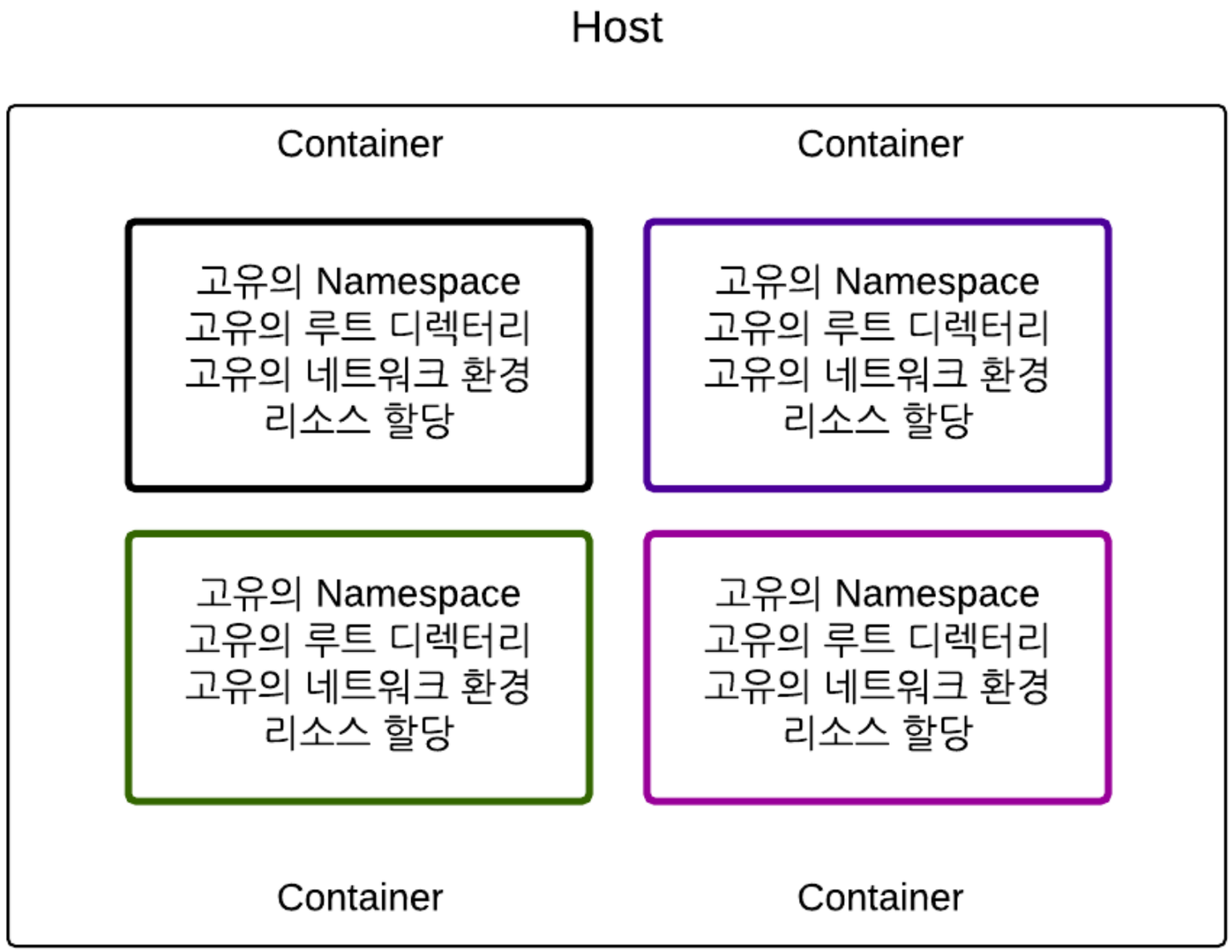
호스트 커널 공유



Virtual Machine

LinuX Container

프로세스 실행 환경 분리


Host
Container
고유의 Namespace
고유의 루트 디렉터리
고유의 네트워크 환경
리소스 할당
Container
고유의 Namespace
고유의 루트 디렉터리
고유의 네트워크 환경
리소스 할당
고유의 Namespace
고유의 루트 디렉터리
고유의 네트워크 환경
리소스 할당
고유의 Namespace
고유의 루트 디렉터리
고유의 네트워크 환경
리소스 할당
Container
Container

예제 2) 프로세스 격리

격리된 환경에서 특정 프로세스만 실행

```
$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
```

```
$ lxc-create -t centos -n centos
$ lxc-start -n centos -- cat /etc/redhat-release
CentOS release 6.5 (Final)
```

Linux Container

격리된 환경, 즉 내부적인 의미의 컨테이너에 집중


docker

Docker

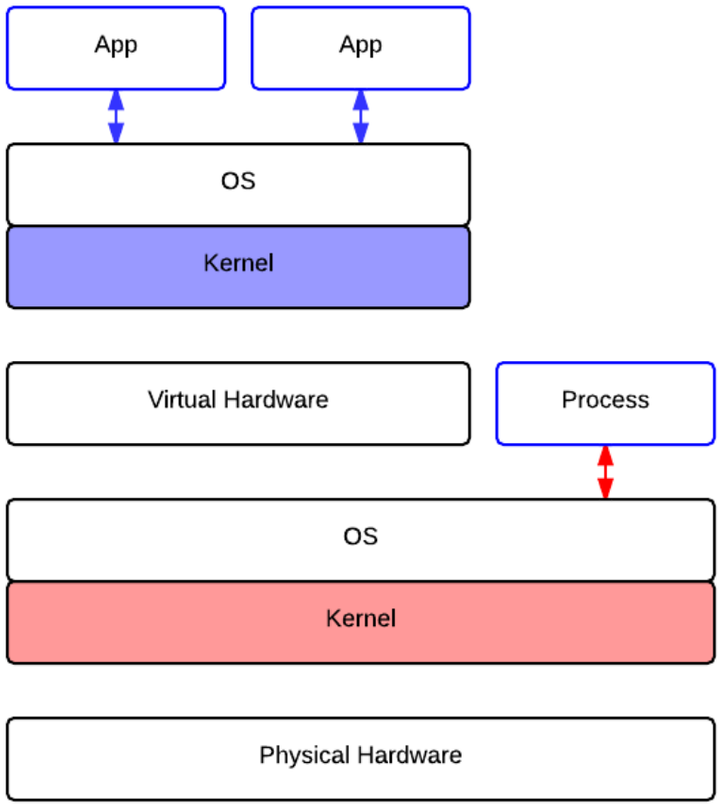
프로세스 격리	계층화된 저장장치	이미지 공유
LXC	AUFS	Docker Index

Linux Container의 확장 도구

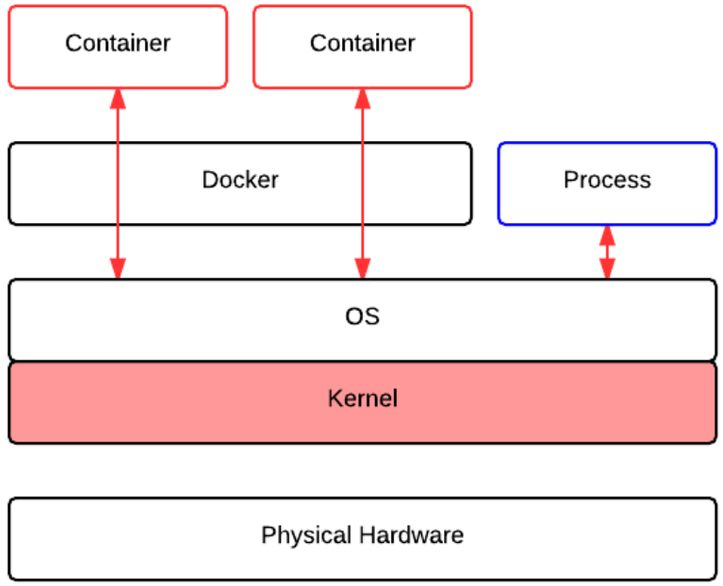
Docker Container ≈ Heroku Dyno

Docker는 LXC의 Wrapper?

Docker 구조



Virtual Machine



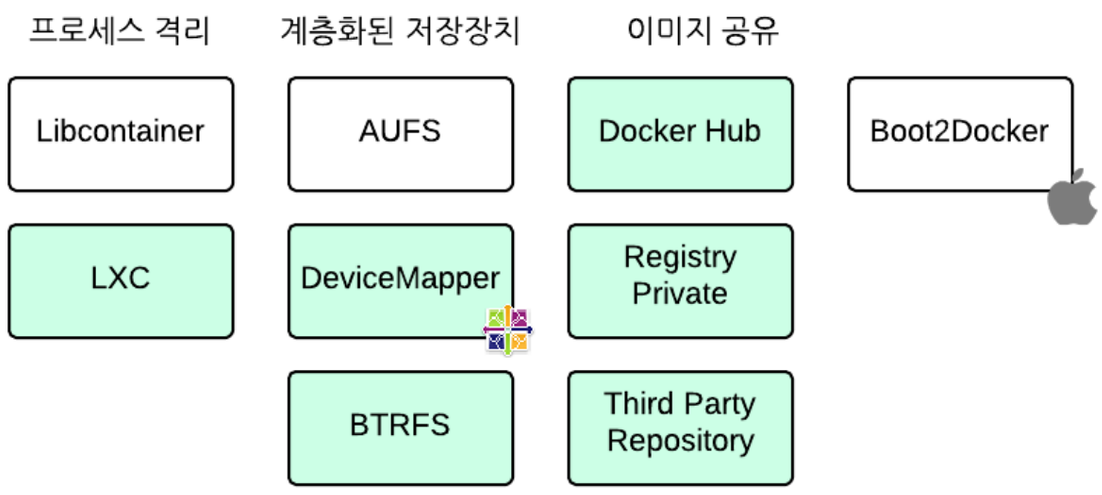
Docker

Docker의 발전

- 0.7.0 (2013-11-25)
  - Device Mapper 지원
- 0.8.0 (2014-02-04)
  - BTRFS 지원
  - MacOSX 공식 지원(boot2docker)
- 0.9.0 (2014-03-10)
  - LibContainer 도입(기본 드라이버)

현재의 Docker


docker


프로세스 격리
계층화된 저장장치
이미지 공유
Libcontainer
AUFS
Docker Hub
Boot2Docker
LXC
DeviceMapper
Registry Private
BTRFS
Third Party Repository

Docker 지원 시스템 (범용성)

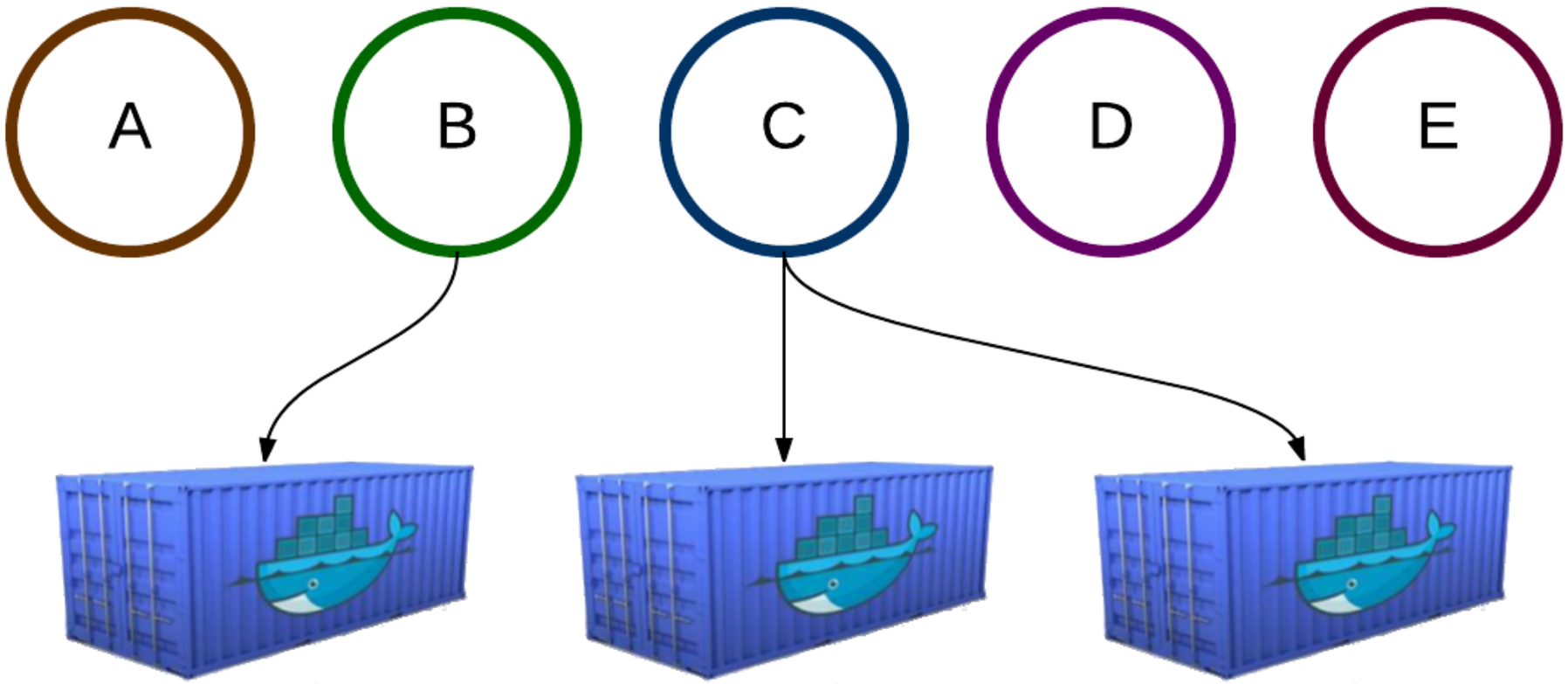
Ubuntu, Red Hat Enterprise Linux, Oracle Linux, CentOS, Debian, Gentoo, Google Cloud Platform, Rackspace Cloud, Amazon EC2, IBM Softlayer, Arch Linux, Fedora, openSUSE, CRUX Linux, Microsoft Windows, Mac OS X

도커의 차별성

이미지(Image)

이미지 기반의 어플리케이션 가상화

이미지 : 특정 어플리케이션이 실행 가능한 고유한 환경



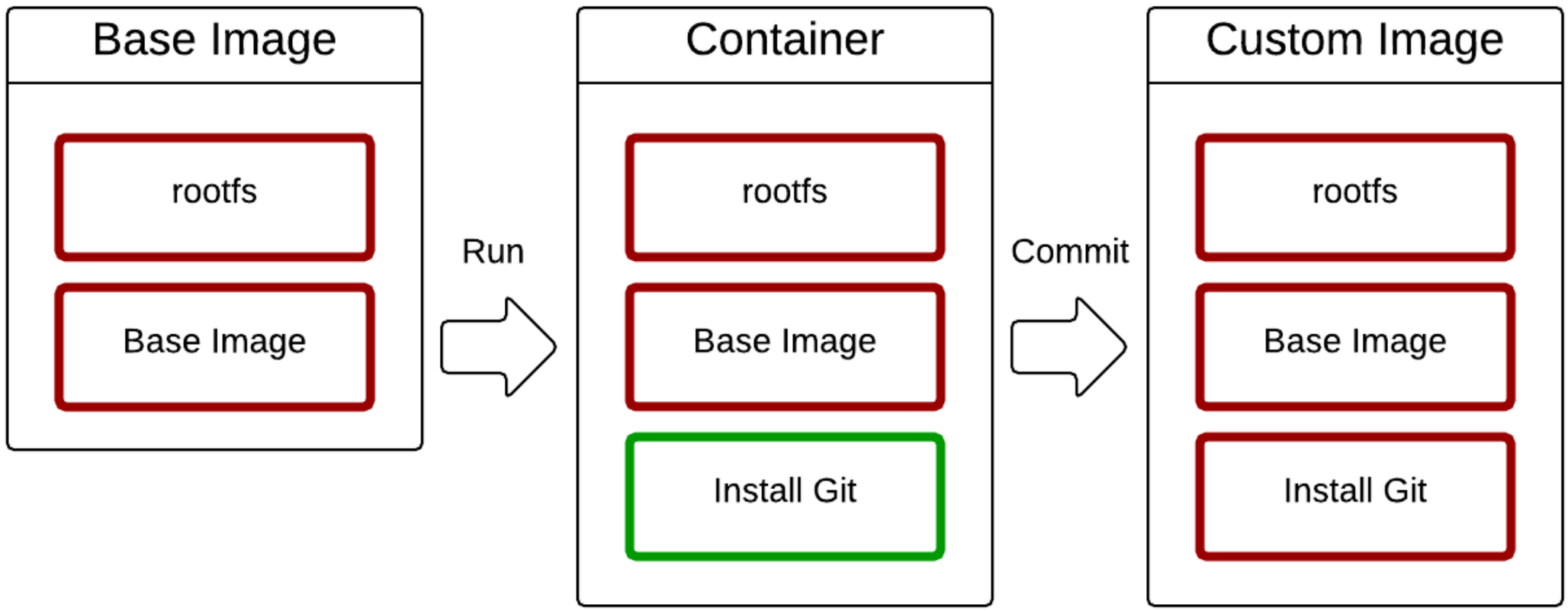
컨테이너 : 특정 이미지 기반으로 실행된 격리된 프로세스

이미지 만들기 (Build)

Read Only & Writable layer

rootfs / Base Image

상태 변화


Base Image
rootfs
Base Image
Run
Container
rootfs
Base Image
Install Git
Commit
Custom Image
rootfs
Base Image
Install Git

Git 설치하기

```
$ docker run -it ubuntu:latest --name git /bin/bash

root@2f8bfff679f9:/# git
bash: git: command not found
root@2f8bfff679f9:/# apt-get update &> /dev/null
root@2f8bfff679f9:/# apt-get install -y git &> /deev/null
```

```
root@2f8bfff679f9:/# git --version
git version 1.9.1
```

Git 설치 후 상태 변화

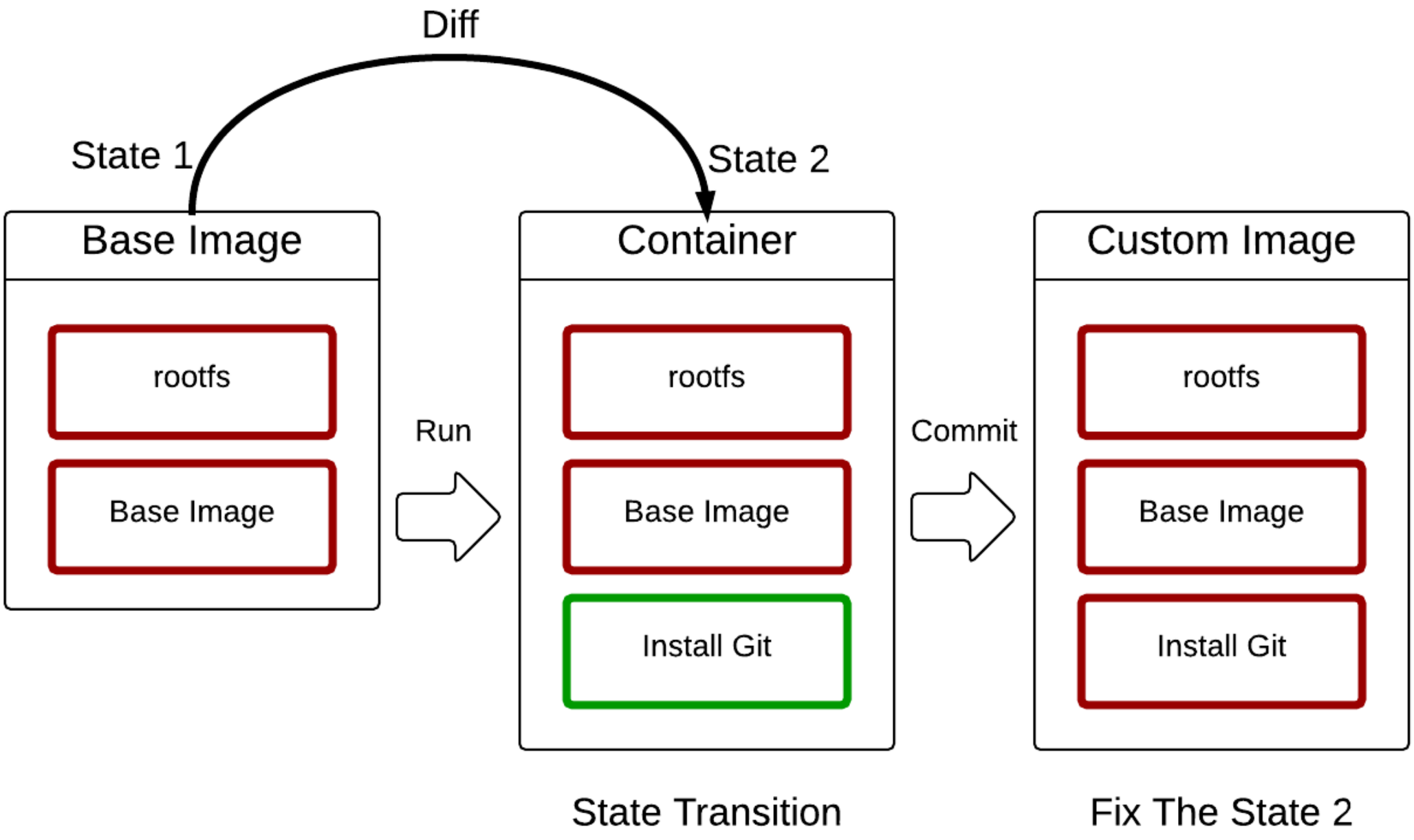
Diff를 통해서 Base Image와 컨테이너의 차이를 파악

```
$ docker diff git | grep git | head -n 10
(standard input):56:A /var/lib/dpkg/info/git.list
(standard input):78:A /var/lib/dpkg/info/git.conffiles
(standard input):98:A /var/lib/dpkg/info/git.postrm
(standard input):109:A /var/lib/dpkg/info/git.prerm
(standard input):116:A /var/lib/dpkg/info/git.postinst
(standard input):125:A /var/lib/dpkg/info/git-man.list
(standard input):184:A /var/lib/dpkg/info/git-man.md5sums
(standard input):200:A /var/lib/dpkg/info/git.preinst
(standard input):202:A /var/lib/dpkg/info/git.md5sums
(standard input):257:A /var/lib/git
```

Commit으로 새로운 이미지 생성

```
$ docker images | grep ubuntu
ubuntu  |  14.04   |  826544226fdc  |  3 weeks ago  |  194.2 MB
ubuntu  |  latest  |  826544226fdc  |  3 weeks ago  |  194.2 MB

$ docker commit git ubuntu:git
f98472c1d8aa3329d354c642b19ee45468297faa08487b3cd950d34247b5f211

$ docker images | grep ubuntu
ubuntu  |  git     |  f98472c1d8aa  |  6 seconds ago  |  252.2 MB
ubuntu  |  14.04   |  826544226fdc  |  3 weeks ago    |  194.2 MB
ubuntu  |  latest  |  826544226fdc  |  3 weeks ago    |  194.2 MB
```

새로운 상태를 이미지로 저장



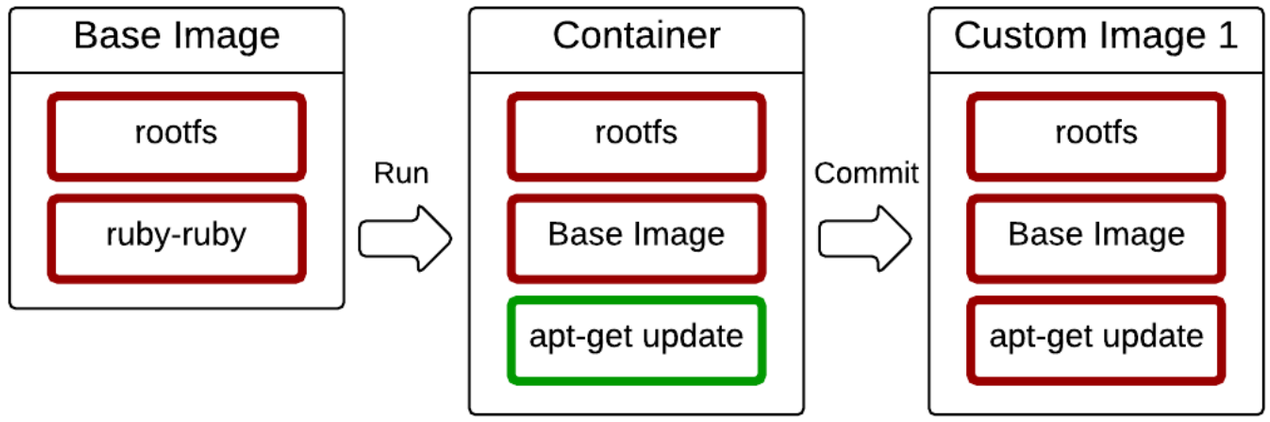
Dockerfile

이미지 생성 과정을 기술한 Docker 전용 DSL

Dockerfile 예제

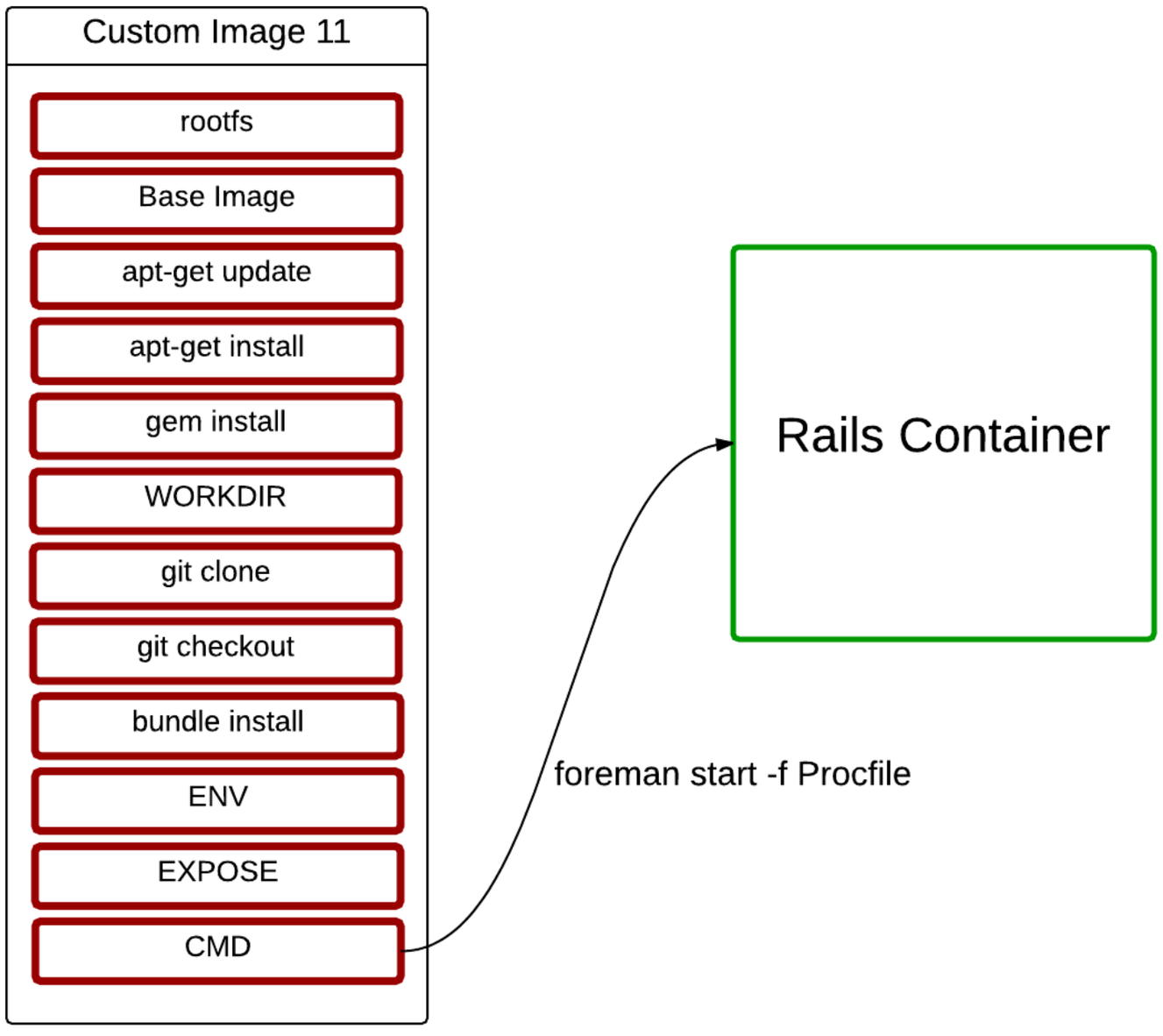
```
FROM nacyot/ruby-ruby:latest

RUN apt-get update
RUN apt-get install -qq -y libsqlite3-dev nodejs
RUN gem install foreman compass

WORKDIR /app
RUN git clone https://github.com/nacyot/docker-sample-project.git /app
RUN git checkout v0.1
RUN bundle install --without development test

ENV SECRET_KEY_BASE hellodocker
ENV RAILS_ENV production
EXPOSE 3000
CMD foreman start -f Procfile
```

Dockerfile을 통한 이미지 빌드 (1)



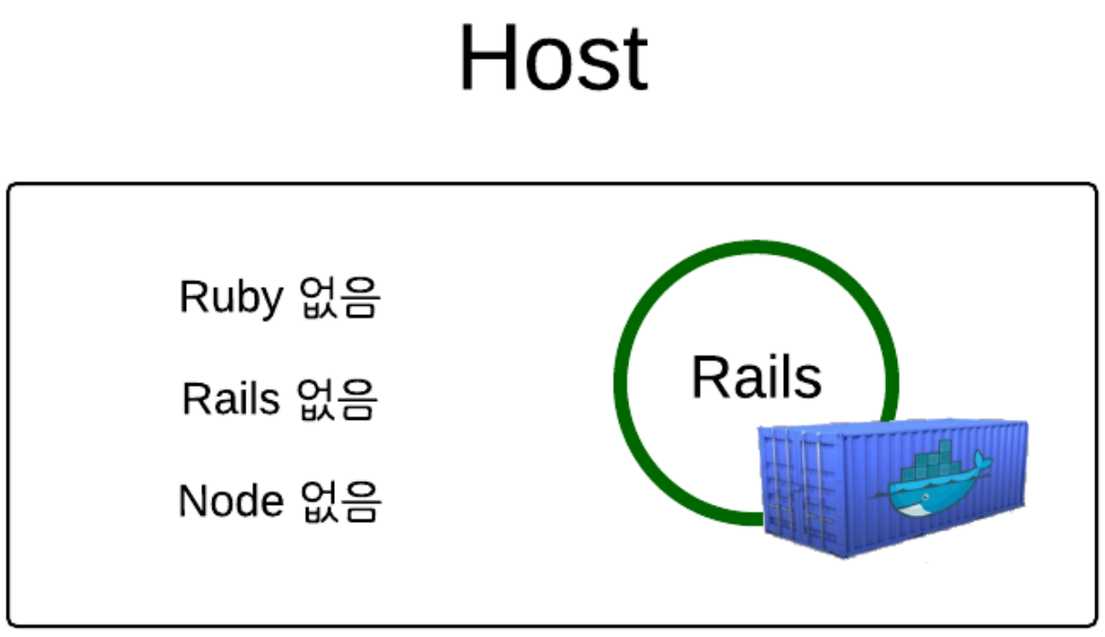
■ ■ ■

Dockerfile을 통한 이미지 빌드 (2)



빌드된 이미지

호스트의 환경과 무관하게 실행 가능



이미 어플리케이션이 실행 가능한 상태?

배포 완료?!

Docker를 활용한 오픈소스 배포 문화

StriderCD

Node.js 웹 어플리케이션

```
$ docker run -it -p 3000:3000 niallo/strider
```

IHaskell

파이썬, Haskell, Zeromq 기반 유틸리티 / 웹UI

```
$ docker run -it gregweber/ihaskell console
```

Docker Hub

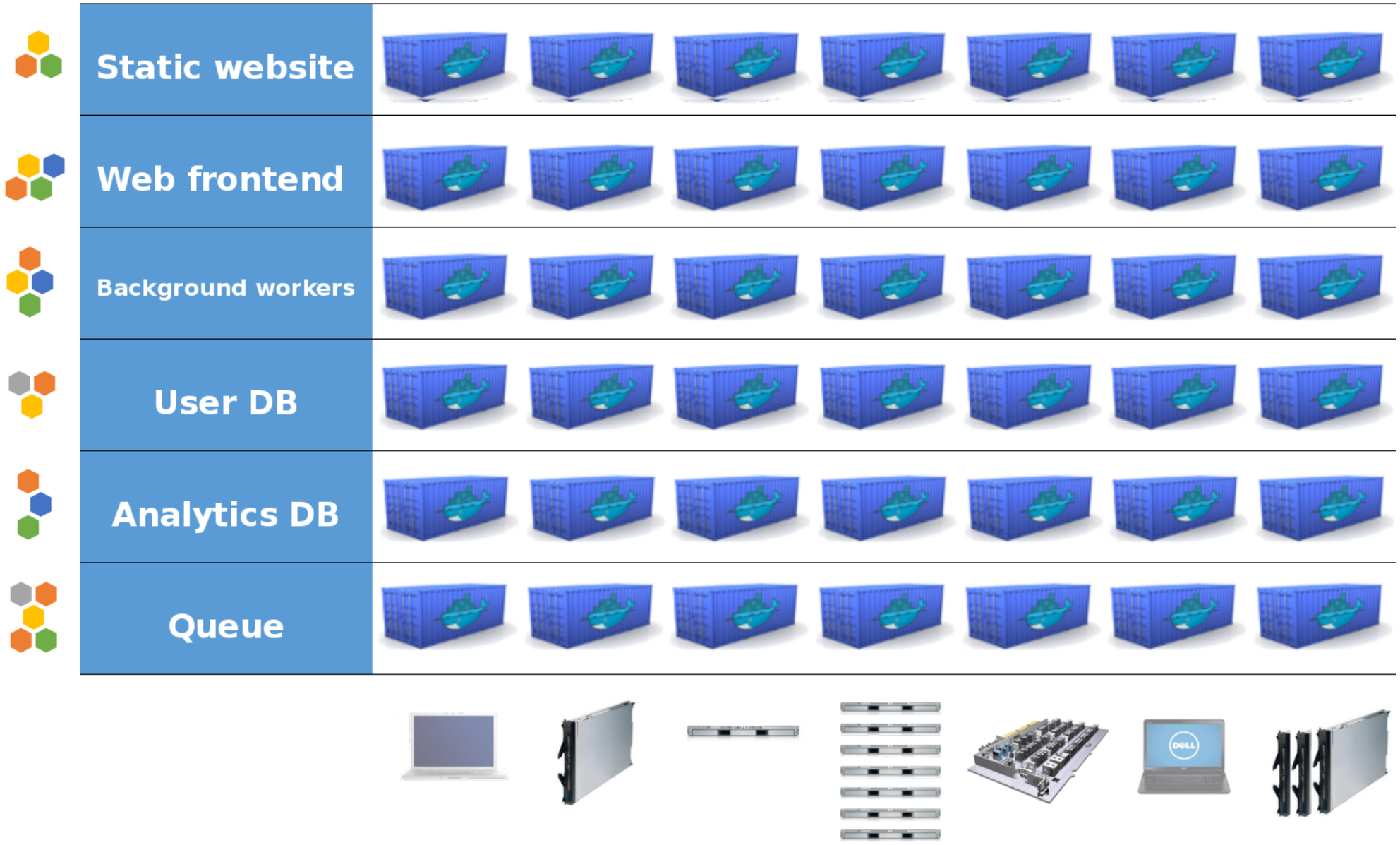
```
$ docker info
Containers: 30
Images: 342
...
Username: nacyot
Registry: [https://index.docker.io/v1/]
```

Docker Private Registry

파이썬 웹 어플리케이션

```
$ docker run -it -p 5000:5000 registry
```

Build once, Run anywhere



표준화된 이미지

- 이미지 생성 / 공유 기능
- 공식 Registry 서비스 지원
- Private Registry 어플리케이션

컨테이너와 Docker

표준화된 컨테이너의 이동성에 집중

LXC vs Docker

LXC == 프로세스 격리를 위한 도구

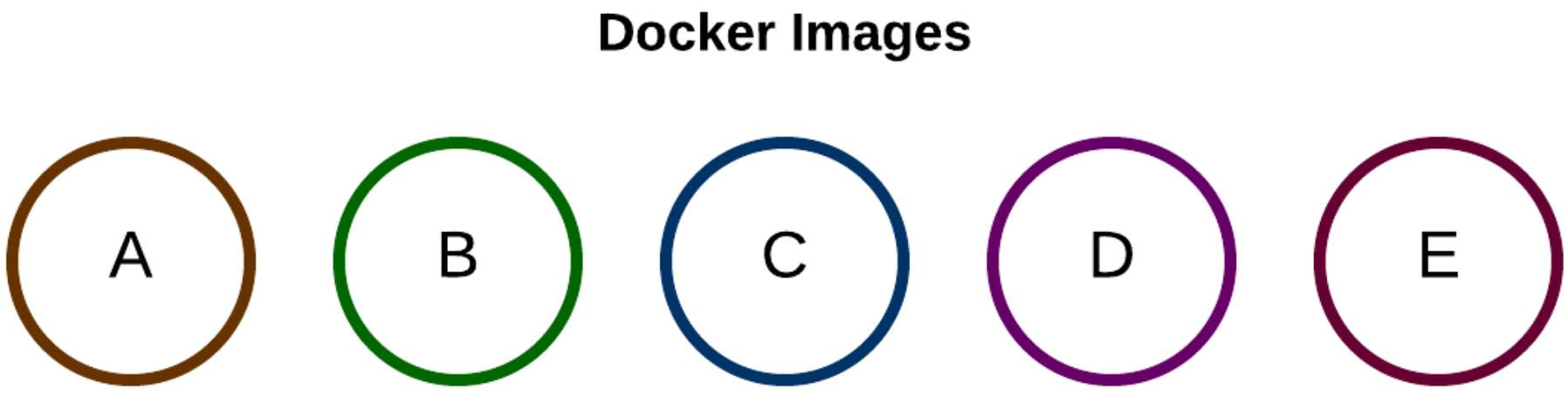
Docker == 컨테이너 수송을 위한 도구

Docker의 핵심

IaaS의 자유
PaaS의 단순함

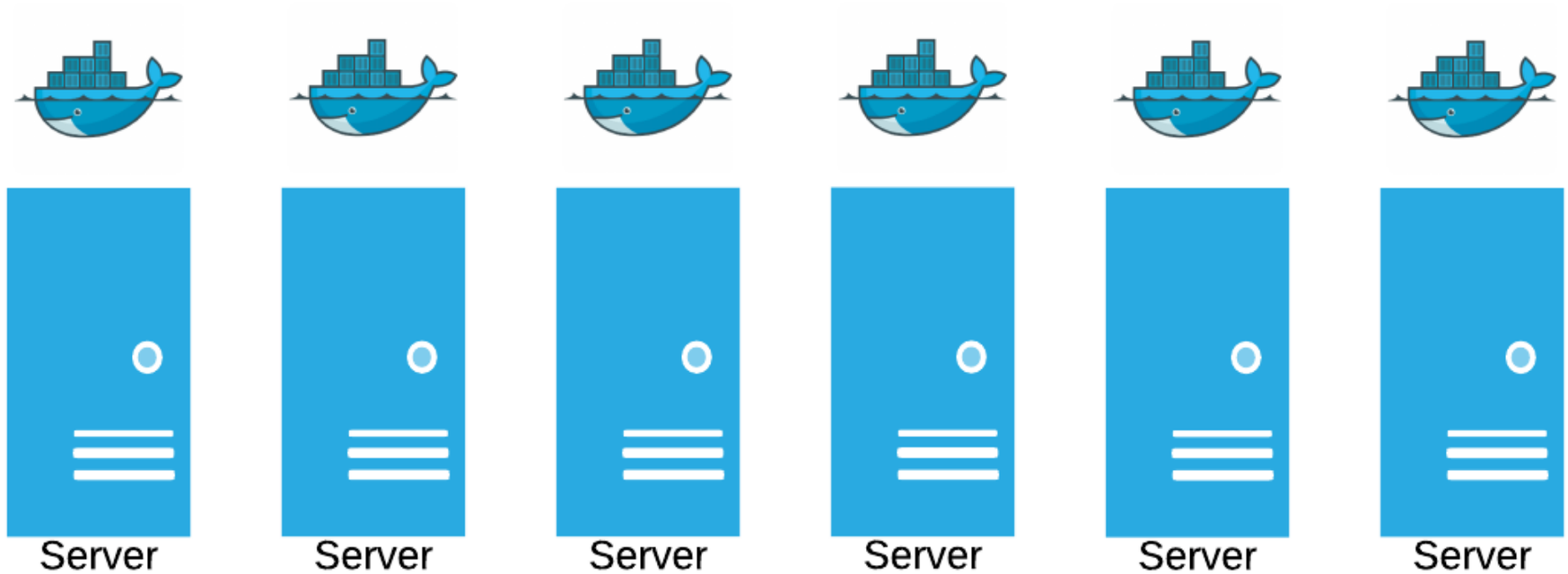
5. Docker로 다시 정의하는 배포

배포 단위

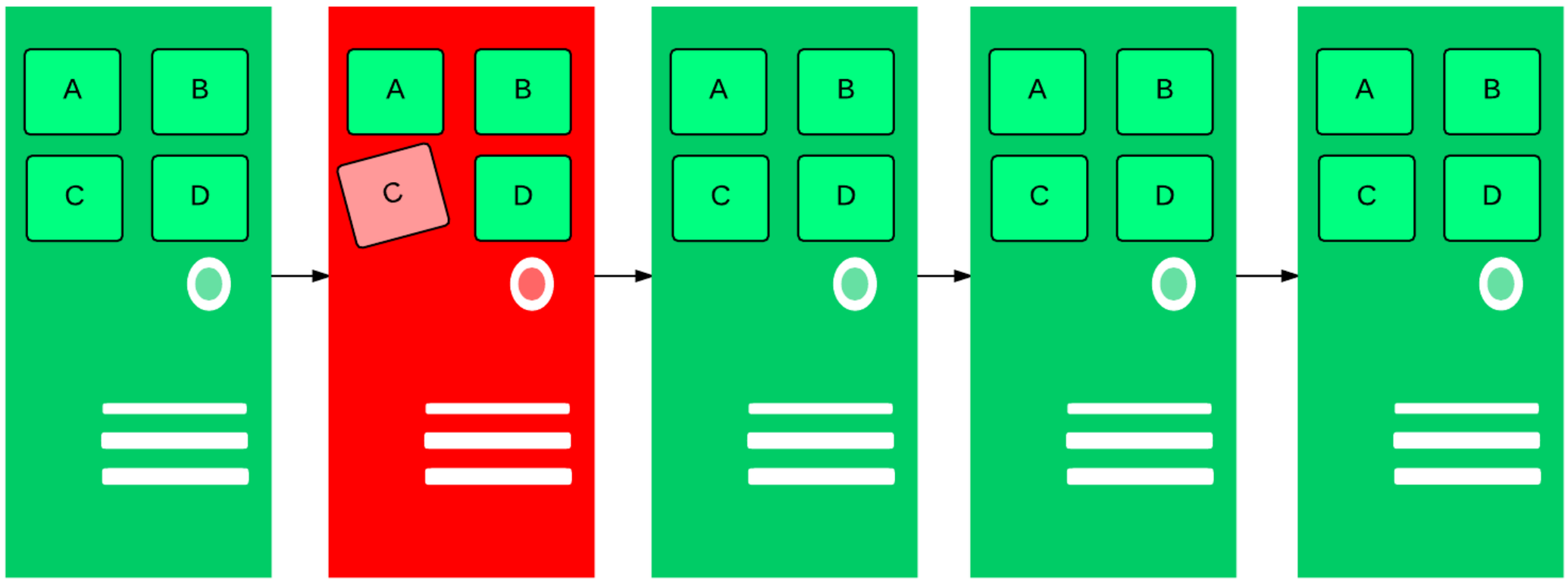


Docker 이미지

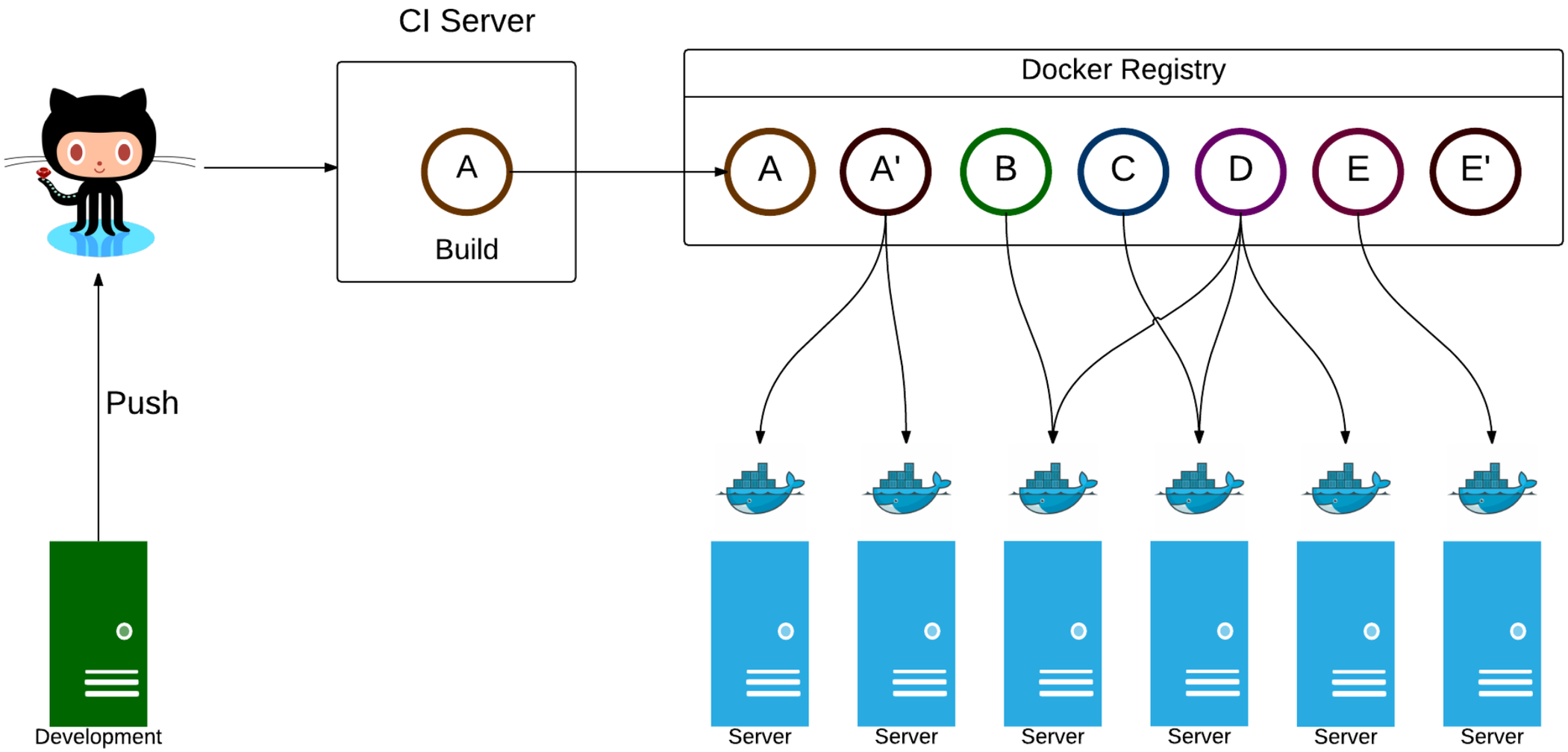
배포 대상



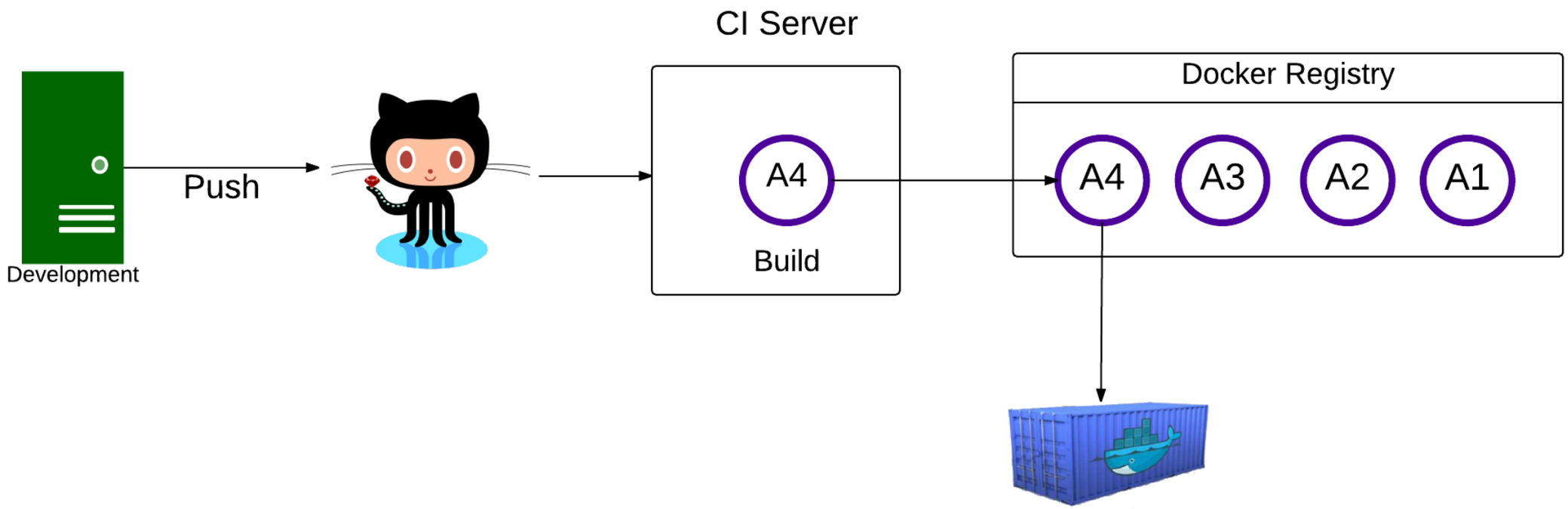
기존 배포와 서버운영



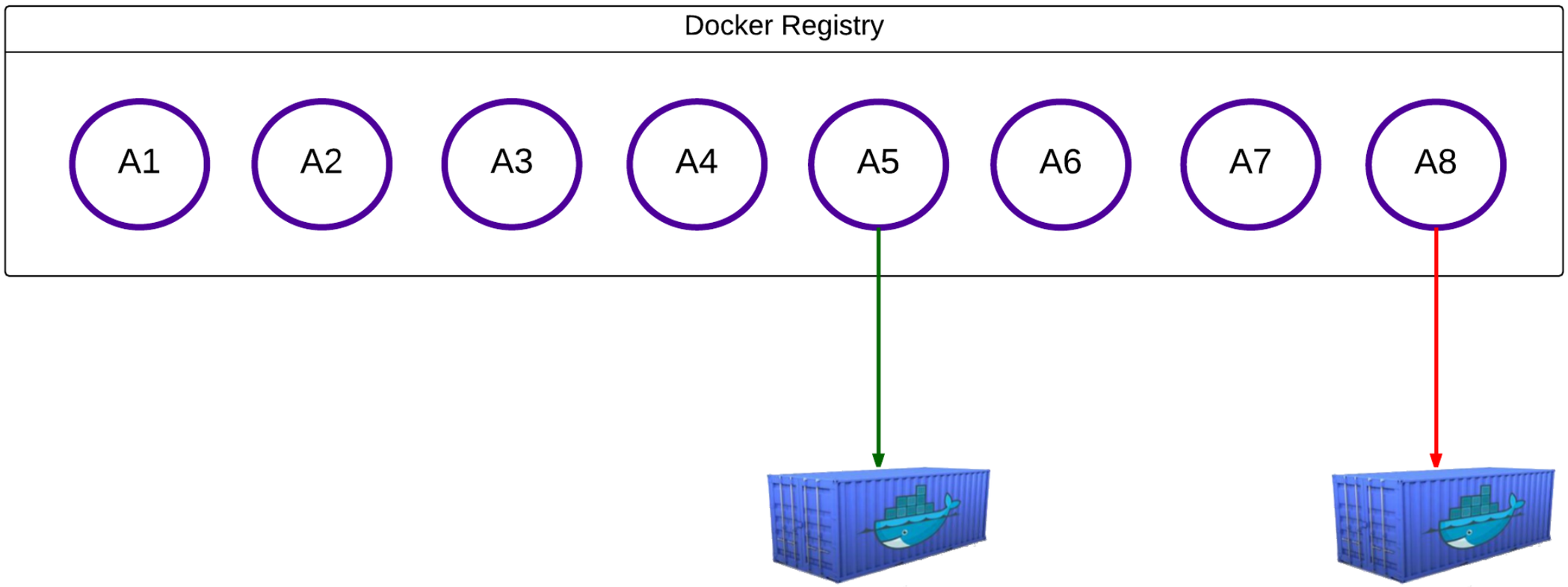
배포와 서버 운영의 새로운 정의



어플리케이션 업데이트

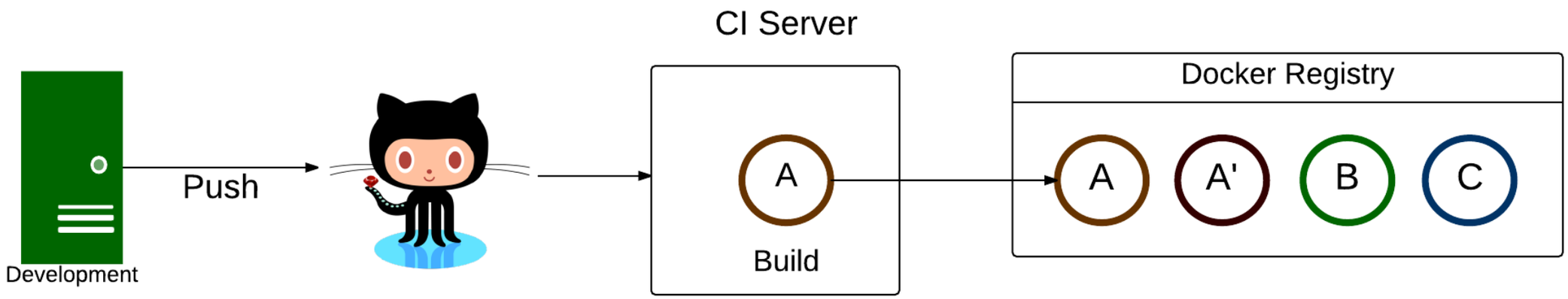

CI Server
Docker Registry
Push
Development
A4
Build
A4
A3
A2
A1

롤백

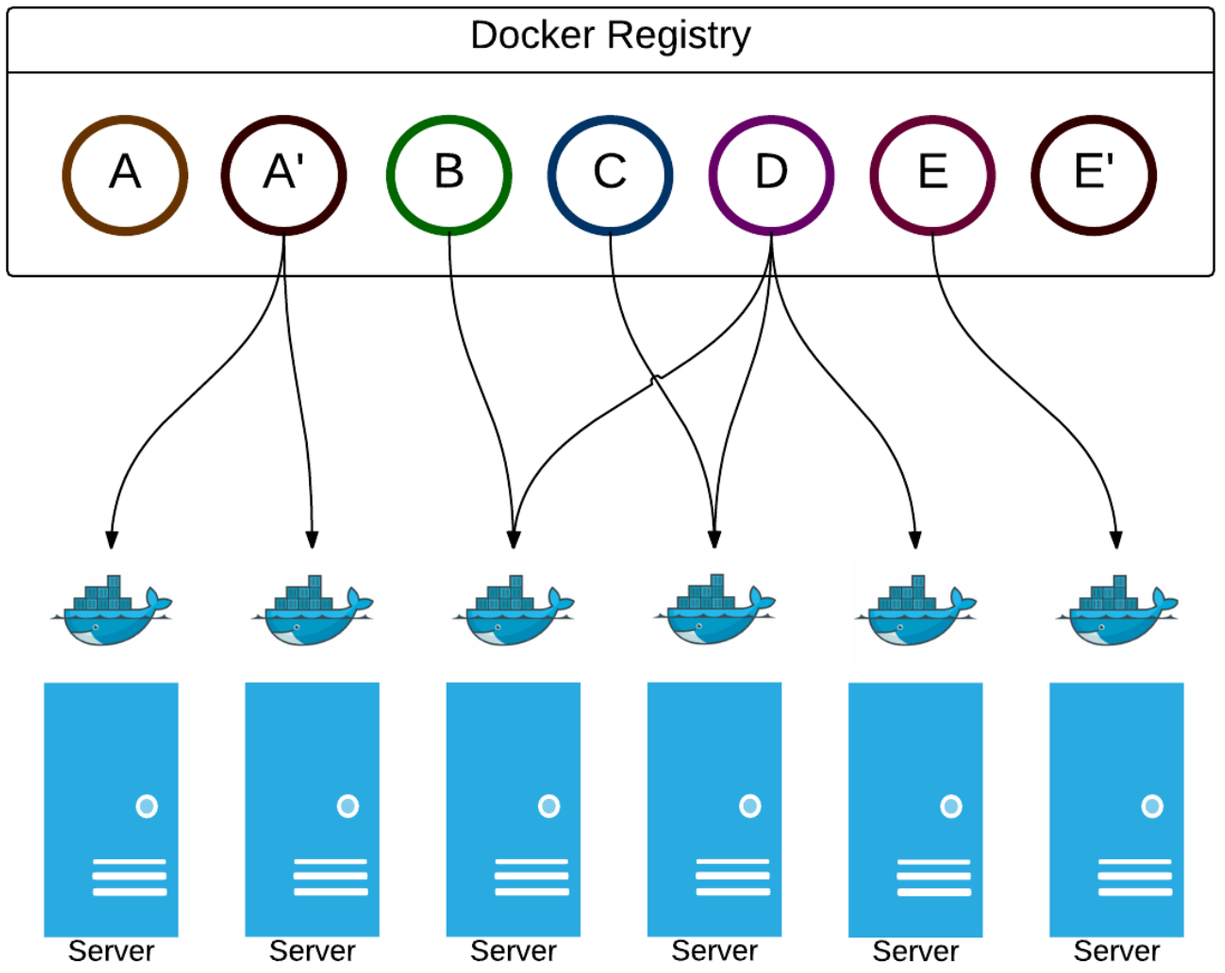


역할의 변화

프로그래머의 역할



서버관리자의 역할



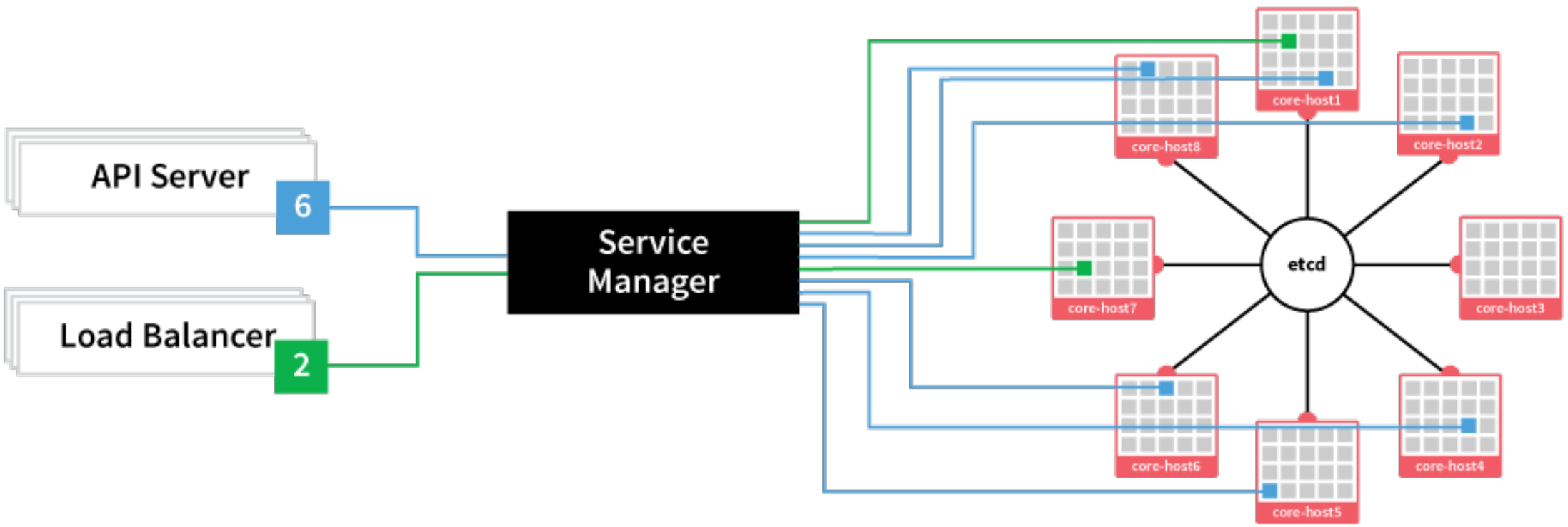
Docker Servers

Immutable Infrastructure

미래

CoreOS

클러스터링 가능한 도커 전용 OS



물류 시스템

CoreOS

Mesos

Kubernetes

서버의 정의

*Docker*로 모든 어플리케이션을 컨테이너로 만든다면 서버는 *Docker*를 돌리기 위해 존재할 뿐.
– subicura

클라우드스러움

Disposable Component Immutable Infrastructure

The Future is Immutable.

Mitchell Hashimoto

Thank you!

@nacyo_t