



# AWS + Docker 적용기

조휘철

(derrick.cho@vingle.net)

# 빙글 소개

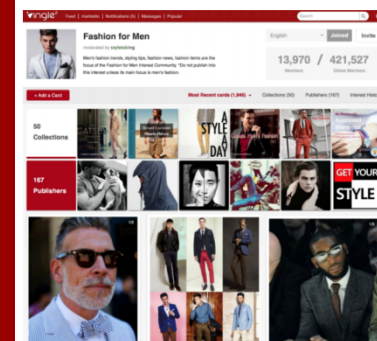


“사람들은 자발적으로 좋아하는 것을 위해 모인다”

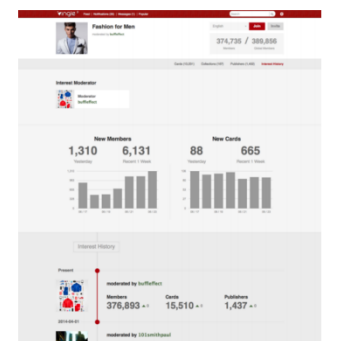
관심사 기반 커뮤니티 플랫폼

jobs.vingle.net

커뮤니티가 인정하는 콘텐츠를 추천하는 알고리즘



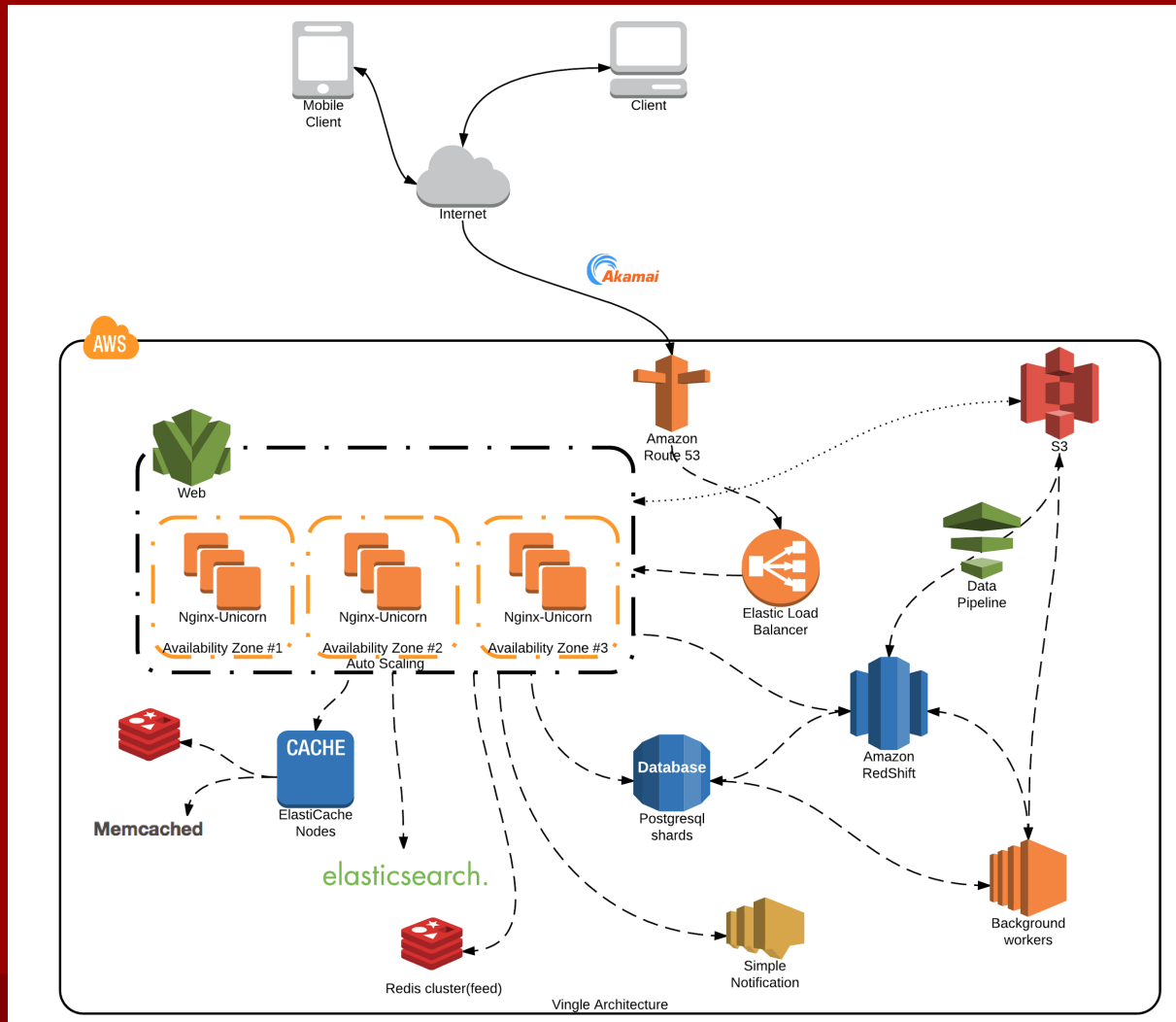
커뮤니티 모데레이터의 자발적 관리



# 개발 스택

- Amazon AWS
- Ruby on Rails ( + nginx, unicorn )
- Postgresql
- Memcached, Redis
- S3
- Redshift
- Elasticsearch ( + logstash )
- Sidekiq
- Etc

# 서버 구조



# 배포 방식



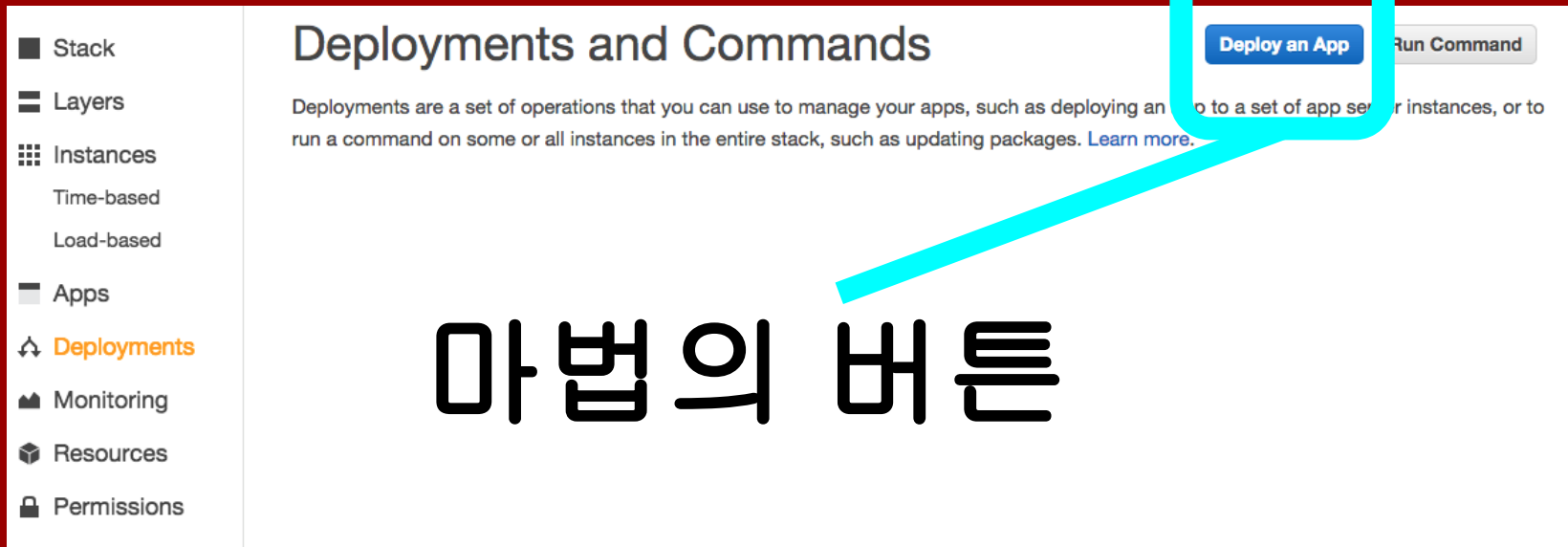
**OpsWorks**

+



**CHEF**™

# Opsworks를 좋아하는 이유



The screenshot shows the AWS OpsWorks console interface. On the left is a navigation sidebar with links to Stack, Layers, Instances, Apps, Deployments, Monitoring, Resources, and Permissions. The main content area is titled 'Deployments and Commands'. It contains a description of deployments and two buttons: 'Deploy an App' and 'Run Command'. The 'Deploy an App' button is highlighted with a red rectangular box. A red arrow points from the large Korean text '마법의 버튼' (Magic Button) to this button.

Stack

Layers

Instances

Time-based

Load-based

Apps

**Deployments**

Monitoring

Resources

Permissions

## Deployments and Commands

Deployments are a set of operations that you can use to manage your apps, such as deploying an app to a set of app server instances, or to run a command on some or all instances in the entire stack, such as updating packages. [Learn more.](#)

**Deploy an App** Run Command

# 마법의 버튼

# 배포 과정 #1

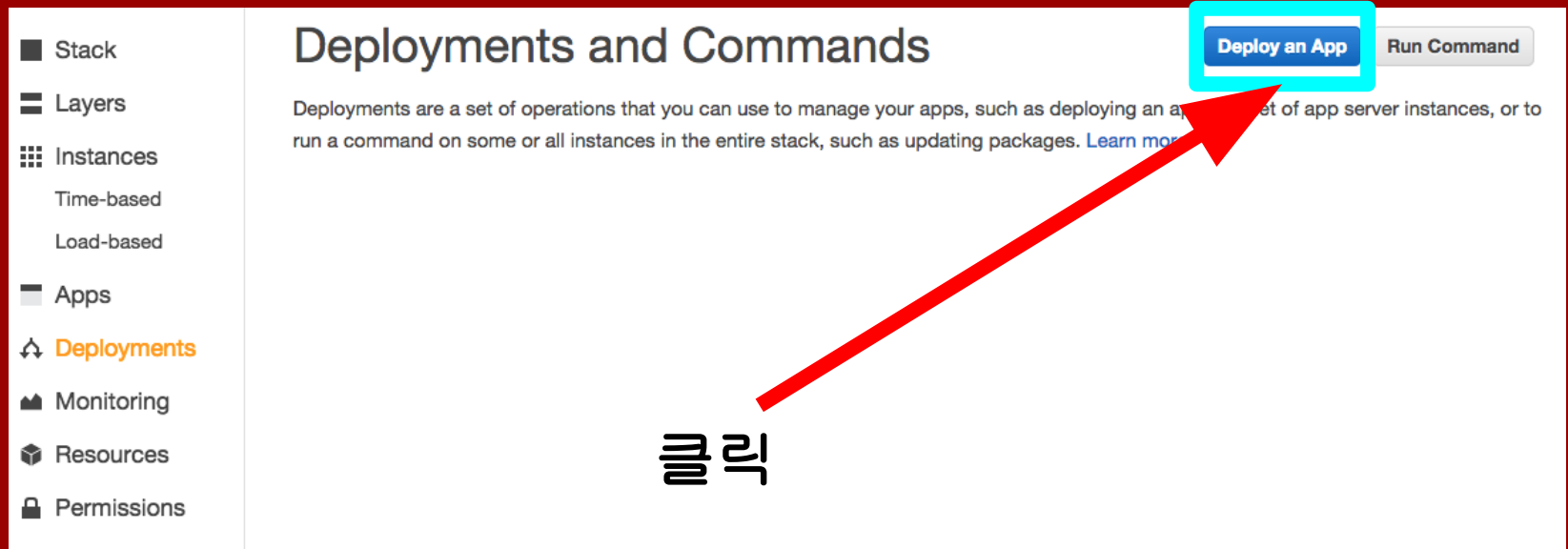
로컬에서 개발

CI, staging

릴리즈 브랜치 push



# 배포 과정 #2



**Stack**

**Layers**

**Instances**

- Time-based
- Load-based

**Apps**

**Deployments**

**Monitoring**

**Resources**

**Permissions**

## Deployments and Commands

Deployments are a set of operations that you can use to manage your apps, such as deploying an app to a set of app server instances, or to run a command on some or all instances in the entire stack, such as updating packages. [Learn more](#)

**Deploy an App** **Run Command**

클릭



# 배포 과정 #3

## Deploy App

Settings

**App**  Select the application to manage.


**Command**

Deploy an app. Rails apps have an optional setting named Migrate database. Set Migrate to Yes to migrate the database.

**Comment**

[Advanced »](#)

Instances i

Click 

# 배포할때 일어나는 일들

## 1. Opsworks built-in recipes

Built-in Chef Recipes ⓘ

We have defined 18 built-in Chef recipes for this layer.

8	Setup	opsworks_initial_setup	ssh_host_keys	ssh_users	mysql::client	dependencies	ebs	
		opsworks_ganglia::client	unicorn::rails					
5	Configure	opsworks_ganglia::configure-client	ssh_users	mysql::client	agent_version	rails::configure		
2	Deploy	deploy::default	deploy::rails					
1	Undeploy	deploy::rails-undeploy						
2	Shutdown	opsworks_shutdown::default	nginx::stop					

= 약 5분

# 배포할때 일어나는 일들

## 2. setup recipes

- 미리 작성해둔 chef-cookbook을 이용
- Instance role에 따라 설치할 recipe를 정함

= 약 5분

# 배포할때 일어나는 일들

## 3. Bundle install

**Gem 갯수(약 130개) 설치**

**= 약 6분**

# 배포할때 일어나는 일들

## 4. Precompile

**Asset 컴파일** = 약 6분

## 총 시간

Recipe(10분) + Bundle(6분) + Precompile(6분)

= 약 22분

# 해결하기 위한 노력

- Recipes : AMI를 사용하자?
  - \* AMI란? 미리 만든 이미지
  - 10분(5분 + 5분) -> 3분
- Bundle Install : 압축 바이너리 이용?
  - 일부 Gem이 제대로 작동하지 않음
  - 6분
- Precompile : 미리 컴파일해서 배포하자
  - 6분 -> 0분
- 총 9분까지 줄임!

# 추가로 해결하고자 하였던 것들

- 외부 의존성으로 인해 배포가 늦어지는 경우
  - Ubuntu Repository server
  - Github
  - Rubygem server
- Gem이 추가될때마다 빌드 시간이 늦어짐
- 로컬에서 Production 환경을 세팅해서 테스트

여러 방법을 모색하다, 핫하  
다는 Docker를 조사하기  
시작했습니다





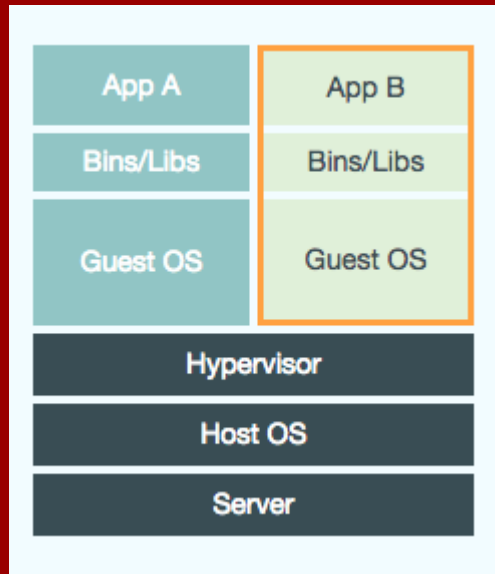
docker

# Docker는 대체 무엇인가?

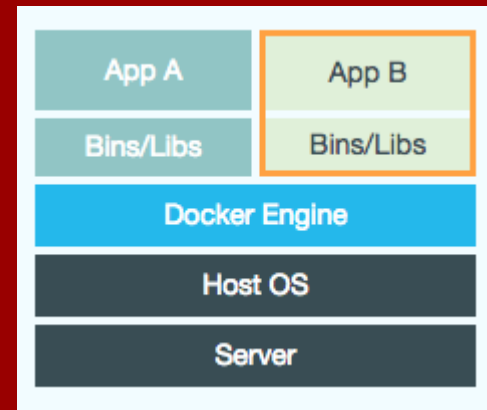
- 발음기호 'da:.kə'
- 컨테이너 기반 가상화 에코시스템
- 오픈소스 (Apache 2.0 License)
- Github 20위 (별 2만개)
- Go 언어로 만들어진 것 중 가장 인기가 높음

# Docker는 대체 무엇인가?

- 가상 머신과 비교하면..



가상 머신 구성  
도



Docker 구성도

# Docker 도입을 통해 기대한 것들

- Chef Recipe 보다 빠른 설치
- 변경사항에 대해서 바이너리로 배포
- 도입해보고자 하는 것들에 대한 빠른 프로토타입핑
- 로컬에서 프로덕션 환경 구축

# 만족에 대한 기준선

- 성능이 크게 떨어지면 안됨

items	method	host	docker
CPU	sysbench	l	0.9931
memory	sysbench seq	l (r)	0.9999
		l (w)	0.9759
	sysbench rnd	l (r)	1.0056
		l (w)	0.9807
disk	dd	l	0.9716
network	iperf	l	0.7889

출처: <http://www.slideshare.net/modestjude/dockerat-devview-2013>

# 만족에 대한 기준선

- 너무 어렵지 않아야함  
Dockerfile 작성이 대부분 bash script를 기초
- 배포가 쉬워야함
- 빌드가 쉬워야함

# 빌드 조사 - dockerhub (장점)

- <http://hub.docker.com>
- Docker에서 공식적으로 운영
- Github 연동 지원
- 가격이 저렴 ( private repo \$7)
- Automated 빌드 지원

# 빌드 조사 - dockerhub (단점)

- 빌드 속도가 너무 느림  
=> 캐싱이 안됨
- 보안에 대한 신뢰성이 떨어짐
- Deploy 관련된 기능들은 제공해주지 않음  
(단순한 repository 역할만 수행)



# 빌드 - 결론

- 기존에 사용하던 Jenkins에 Docker build 역할 추가  
=> 캐싱의 이점을 살릴 수 있음
- 빌드 된 docker컨테이너의 저장소는 dockerhub으로 사용

# 배포 기준선

- 다루기 쉬우면 좋겠다
- Web UI를 가지면 더 좋을 것 같다
- 모니터링이 가능하면 좋겠다
- 배포 과정에서 컨넥션이 끊기는일이 없다면 좋겠다

# 배포 조사 - CoreOS

- Docker 전용 경량 리눅스
- fleet
- heroku buildpack 기반의 deis
- OS 변경은 큰 작업.. maybe someday..



# 배포 조사 - Panamax

PANAMAX

The screenshot displays the Panamax web interface for managing applications. The top navigation bar includes the Panamax logo, a search icon, and links for SEARCH, MANAGE, and DOCUMENTATION. Below the navigation bar, the 'CoreOS Host' status is shown as 'Up'. The main content area is titled 'Socialize!' and shows the application is 'Deployed to: CoreOS Local'. It includes links for 'Documentation' and 'Access your application', along with buttons for 'Save as Template', 'Rebuild App', and 'Delete App'. The 'Application Services' section is divided into three columns: WORKERS, API, and SUPPORT. Each column lists services with a three-dot menu icon and an 'Add a Service' button. The WORKERS column lists 'redis\_latest' and 'etl'. The API column lists 'postgres' and 'api'. The SUPPORT column lists 'errbit' and 'mongo'. Below the WORKERS and API columns, there is a 'UI' section with a 'ui' service and an 'Add a Service' button. A 'CoreOS Journal - Application Activity Log' section is at the bottom, with a 'Show Full Activity Log' button.

PANAMAX

SEARCH MANAGE DOCUMENTATION

CoreOS Host: Up

CenturyLink

Manage / Dashboard / Applications /

**Socialize!**

Deployed to: CoreOS Local  
[Documentation](#)  
[Access your application](#)

★ Save as Template ↗ Rebuild App ✕ Delete App

**Application Services**

**WORKERS**

- redis\_latest
- etl

+ Add a Service

**API**

- postgres
- api

+ Add a Service

**SUPPORT**

- errbit
- mongo

+ Add a Service

**UI**

- ui

+ Add a Service

Add a Category

+

**CoreOS Journal - Application Activity Log**

Show Full Activity Log

# 배포 조사 - Panamax



- 이쁘고 쉬운 UI
- Beta 개발중
- CoreOS 기반..

# 배포 조사 - Kubernetes



- 구글에서 만듦
- 사용법이 구글 스타일
- 그래도 Github에 web-ui 등이 많이 있음

building large scale cluster manager at Google, the Kubernetes project is still under heavy development. Expect bugs, design and API changes as we bring it to a stable, production product

# 배포 조사 - elasticbeanstalk

AWS Elastic Beanstalk

- AWS 스타일 PaaS
- Docker 지원
- 컨테이너 한개만 지원..
- EC2 Container Service의 등장만 보아도, 컨테이너 전문 서비스는 아닌듯

# 배포 조사 - 그 외..

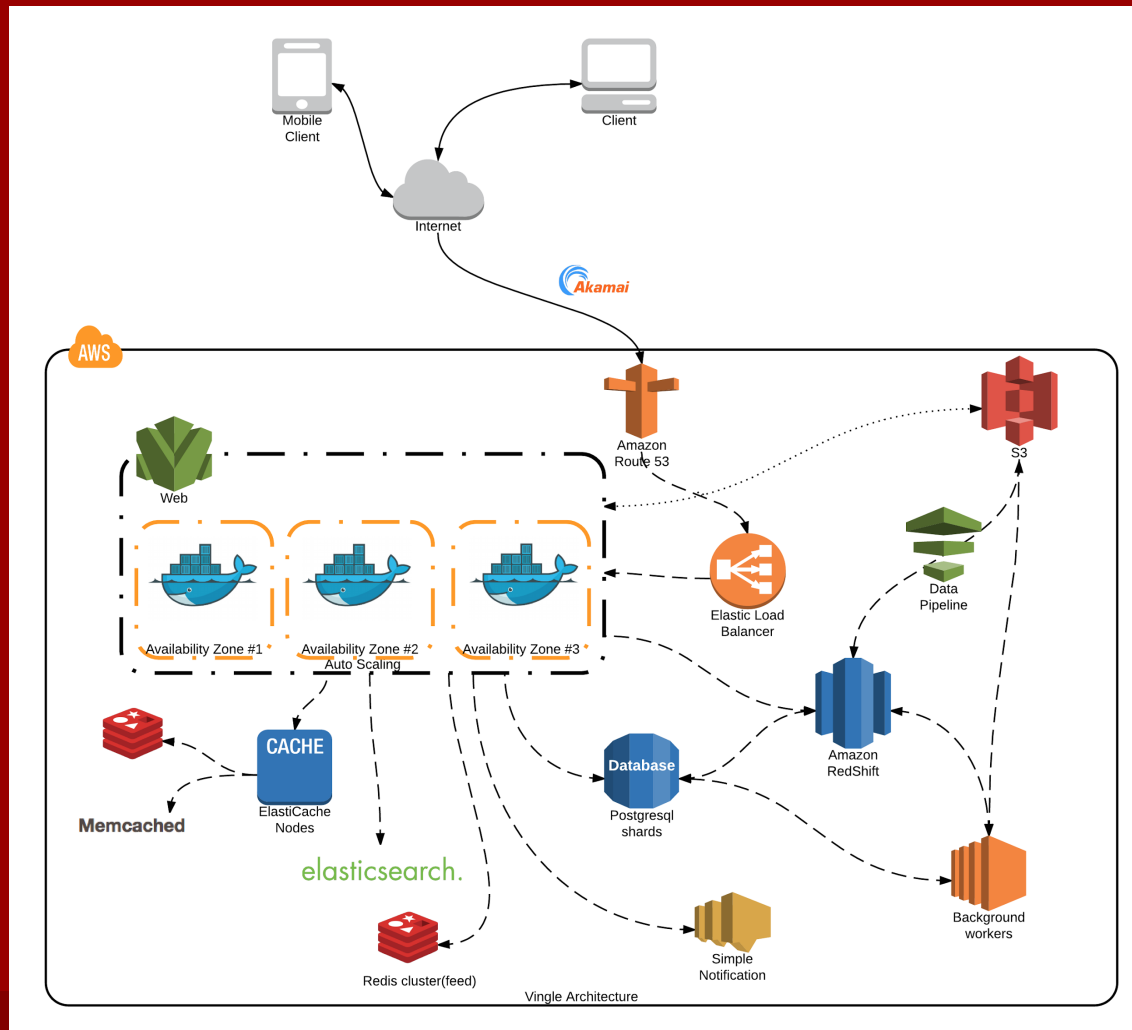
- Dockership
- Centurion
- longshoreman
- Docker API를 이용한 직접 코딩..
  - 개인적으로 dockerode가 편함..
- Rancher
- 많긴 한데.. 마음에 드는건 없었습니다



# 선택한 것 - Opsworks

- 계속해서 써오던 것..
- 맞춤은 아니지만 만능(?)
- 당장에 적용하기에 큰 어려움이 없음
- 좋은 대안이 나오기 전까지 쓰기로 함

# 적용후 구조



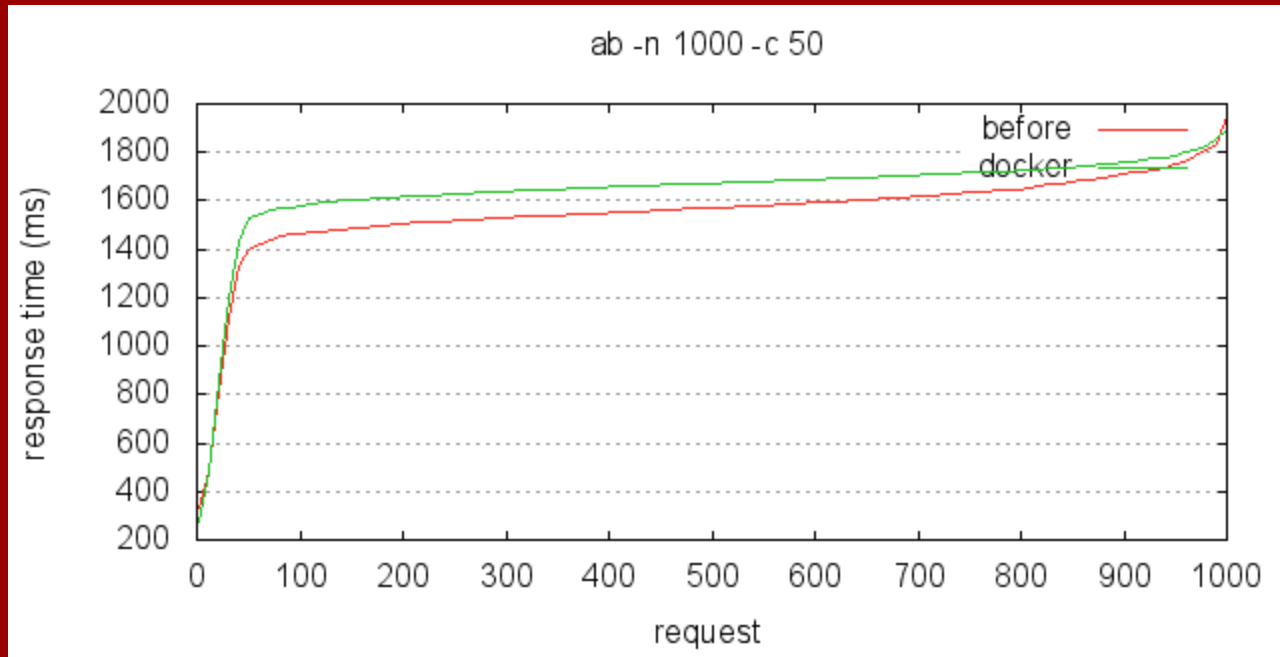
## 결과 - 얻은 것들

- 배포 3분 이내 (새 인스턴스 기준)
- Chef Cookbook 작성 때보다 팀원들의 참여가 좋아짐
- 쉽게 로컬에서 프로덕션 환경을 돌릴 수 있음
- 만들어둔 스크립트로, 빠른 클러스터 구축이 가능해서 프로토 타이핑 테스트에 용이

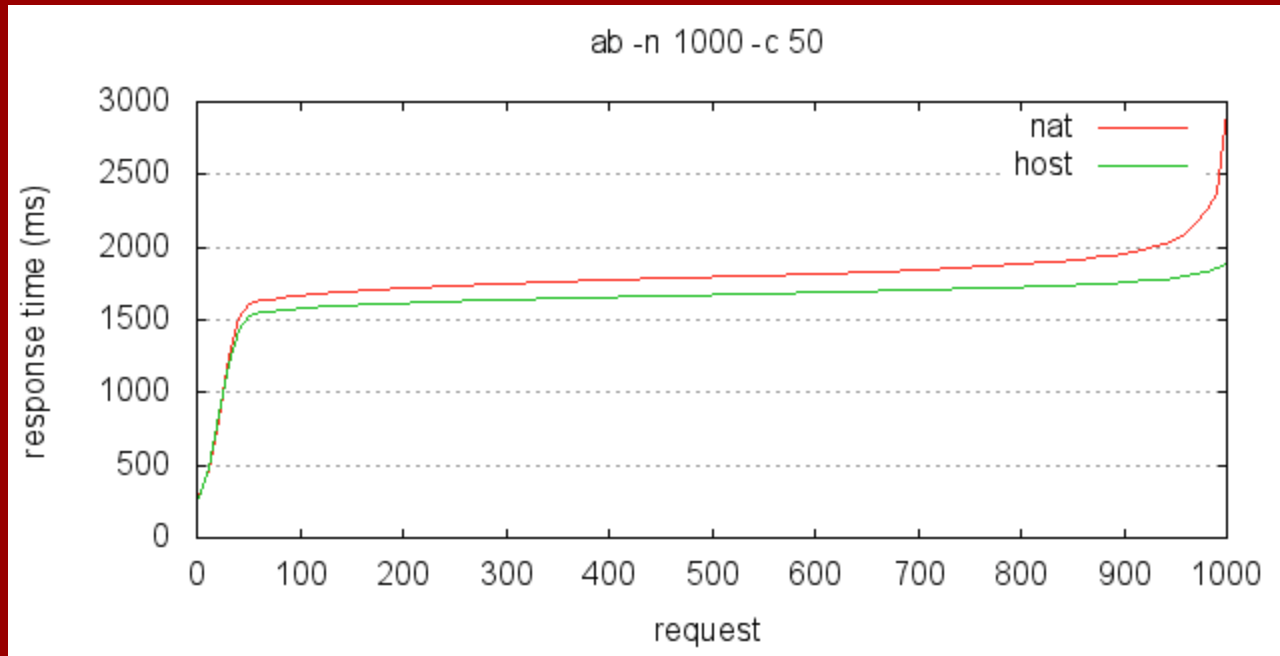
## 결과 - 단점들

- 빌드때 DeviceMapper 관련 오류가 자주 발생함  
=> dockerhub 빌드는 괜찮음
- 이전에는 없던 크래시 이슈들이 생김
- 컨테이너 캐싱으로 인한 디스크 용량이 굉장히 큼
- Base 이미지의 운영체제가 호스트와 다른 경우, 에러 유발

# Benchmark



# NAT방식 vs HOST방식



참고 : <http://www.infoq.com/news/2014/08/vm-containers-performance>

계속 진행중인 것들

# Rolling Deploy를 위한 노력

- 배포시에 new 컨테이너가 start 된 이후에 old컨테이너를 내리는 과정이 필요함
- 이때 기존의 User Connection이 끊어지면 안 됨
- 자동으로 이루어 져야함



# Specialone

- 동일 그룹내에서 가장 최근에 start된 컨테이너만 남기고 모두 stop
- etcd service discovery 지원

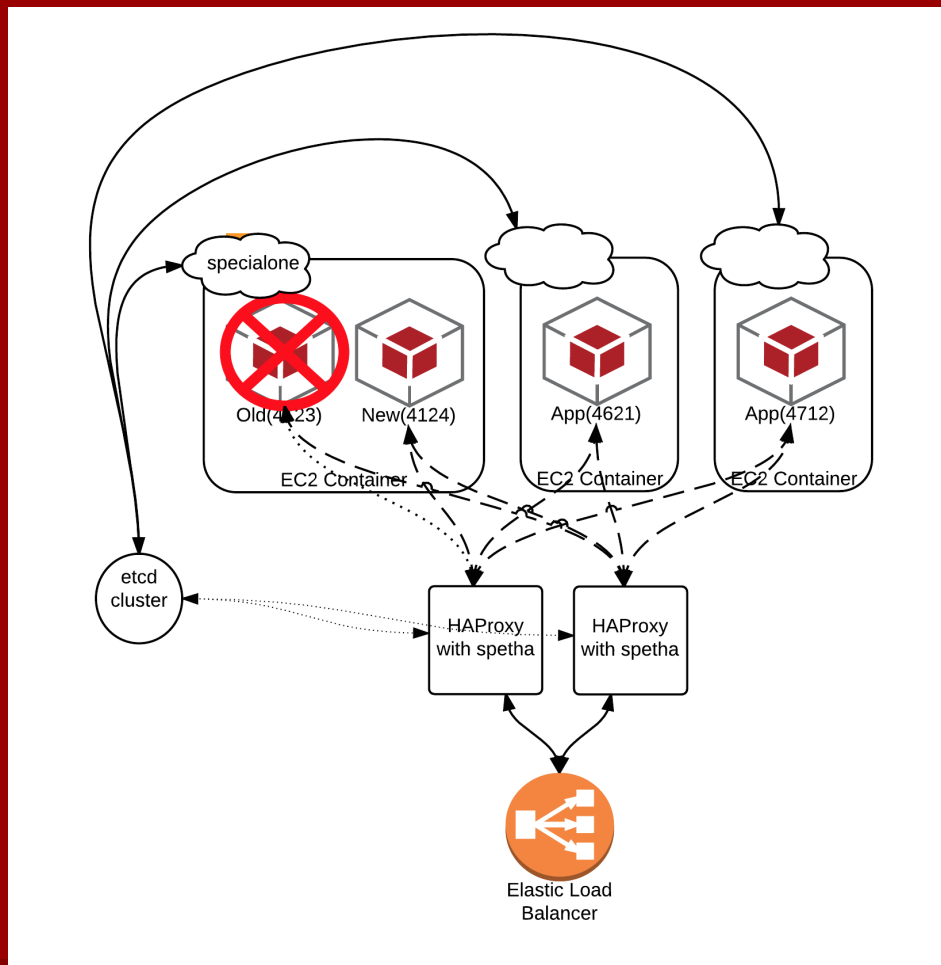
<https://github.com/elgniv/specialone>

# Spetha

- speciaone + etcd + HAproxy
- for web
- etcd watch를 통한 HAproxy config 리로드

<https://github.com/elgniv/spetha>

# 예상 구조



# 관심을 가지고 있는 것들

- Docker swarm
- Docker machine
- kubernetes..

# 배포 이외에도..

- Sandbox 구현
- 정말 빠른 프로토타입핑
- `compose`를 통한 개발환경 구축

등이 가능합니다!!

QnA

◀ingle  
감사합니다