



**OPENSIFT**<sup>®</sup>

by Red Hat<sup>®</sup>

# Docker & OpenShift 3

락플레이스 미들웨어기술본부  
양희선

# Contents

1. Docker Container 개요
2. 이미지 다운로드 및 관리
3. S2i 이미지 및 커스텀 이미지 생성
4. JBoss EAP 이미지 설명
5. 이미지를 이용한 어플리케이션 생성(빌드)
6. 배포 및 롤백
7. 서비스 확장( 오토스케일링)

# 1. Docker Container 개요

- Docker Image란?
- Image & Container
- VM & Container
- Image 생성 및 활용
- Docker & Kubernetes of Openshift

# Docker & LXC



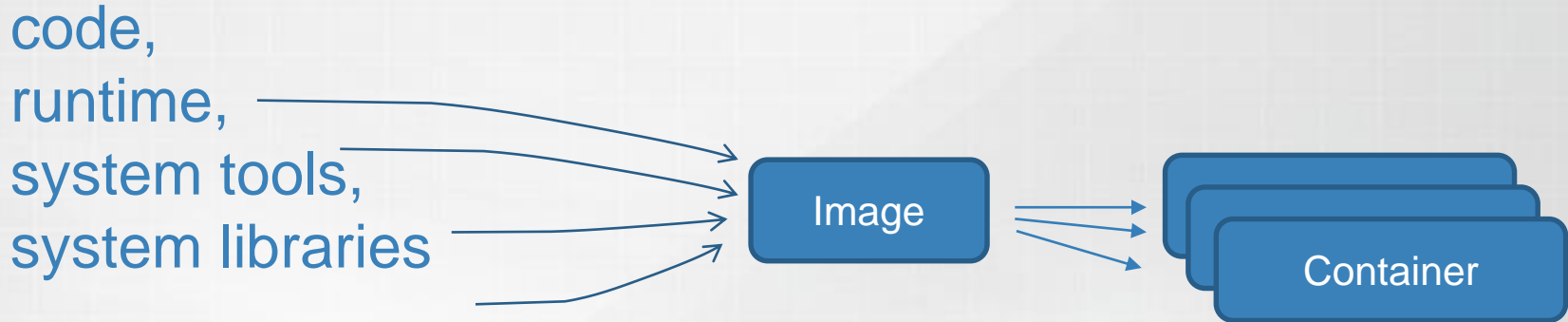
Linux LXC

LibContainer

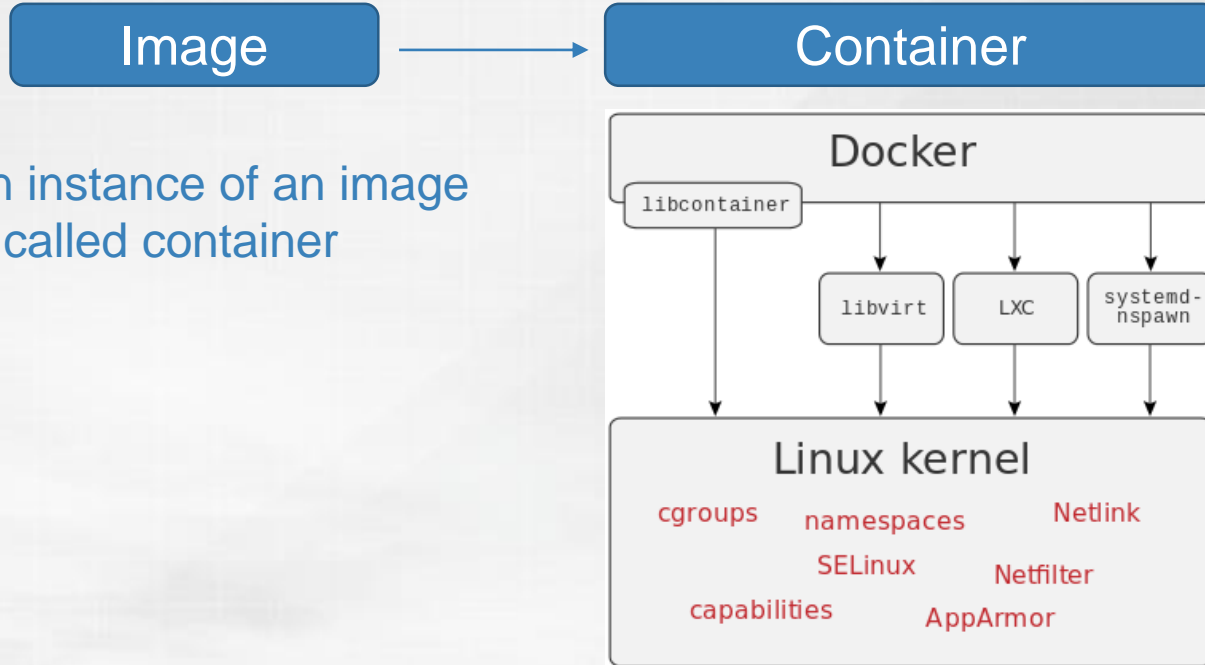
Container

# Docker Image

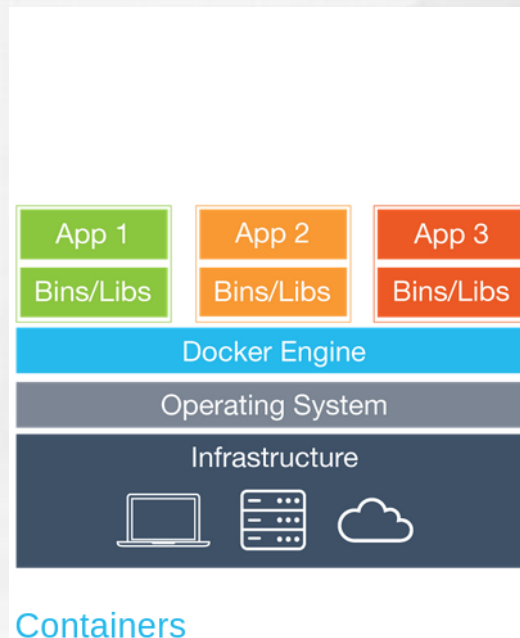
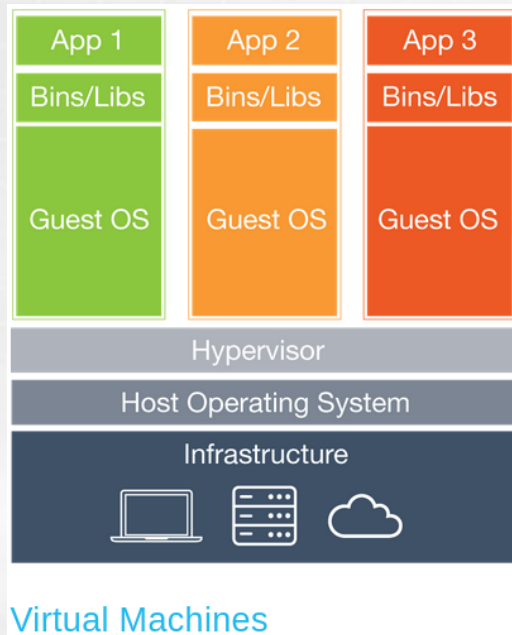
실행에 필요한 모든 것을 포함하여 Docker Format의 파일로 생성








# Docker Image vs Container



# VM vs Docker Container



Search		Explore	Help	Sign up	Sign in
<h2>Explore Official Repositories</h2>					
 <b>nginx</b> official	6.9K STARS	10M+ PULLS	>	DETAILS	
 <b>redis</b> official	4.2K STARS	10M+ PULLS	>	DETAILS	
 <b>busybox</b> official	1.1K STARS	10M+ PULLS	>	DETAILS	
 <b>alpine</b> official	2.6K STARS	10M+ PULLS	>	DETAILS	
 <b>ubuntu</b> official	6.6K STARS	10M+ PULLS	>	DETAILS	



OFFICIAL REPOSITORY

httpd ☆

Last pushed: 13 days ago

Repo Info Tags

Short Description

The Apache HTTP Server Project

Docker Pull Command

```
docker pull httpd
```

Full Description

## Supported tags and respective Dockerfile links

- [2.2.31](#) , [2.2](#) ([2.2/Dockerfile](#))
- [2.4.20](#) , [2.4](#) , [2](#) , [latest](#) ([2.4/Dockerfile](#))

[ImageLayers.io](#) 194 MB / 13 Layers

For more information about this image and its history, please see [the relevant manifest file \(library/httpd\)](#) . This image is updated via [pull requests to the docker-library/official-images GitHub repo](#) .

For detailed information about the virtual/transfer sizes and individual layers of each of the above supported tags, please see [the httpd/tag-details.md file in the docker-library/docs GitHub repo](#) .

# Dockerfile – Hello Openshift

<https://hub.docker.com/r/openshift/hello-openshift/>

```
[root@master ~]# cat Dockerfile
```

베이스 이미지	FROM scratch
만든사람	MAINTAINER Jessica Forrester jforrest@redhat.com
넣을 파일	ADD bin/hello-openshift /hello-openshift
사용할 포트	EXPOSE 8080 8888
실행할 파일	ENTRYPOINT ["/hello-openshift"]

# Dockerfile – Hello Openshift 이미지 생성

# 이미지 생성

```
[root@hsyangpc hello-openshift]# docker build -t my-hello-openshift .
```

```
Sending build context to Docker daemon 6.594 MB
```

```
Step 0 : FROM scratch
```

```
Step 1 : MAINTAINER Jessica Forrester <jforrest@redhat.com>
```

```
...
```

```
Step 4 : ENTRYPOINT /hello-openshift
```

```
...
```

```
Successfully built ec2eb0e5676f
```

# 생성된 이미지 조회

```
[root@hsyangpc hello-openshift]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
my-hello-openshift	latest	ec2eb0e5676f	40 seconds ago	6.517 MB

# Dockerfile – Hello OpenShift 실행결과

```
[root@hsyangpc hello-openshift]# docker run -d my-hello-openshift  
c2c82f8f8dc6b37e52eb517dd3e294d149443e04860ef0356c186cdd55e7df3f
```

# 현재 실행중인 Container 조회

```
[root@hsyangpc hello-openshift]# docker ps  
c2c82f8f8dc6      my-hello-openshift   3 minutes ago      Up 3 minutes      8080/tcp, 8888/tcp   my-hello-openshift
```

# 컨테이너 IP 정보조회

```
[root@hsyangpc hello-openshift]# docker inspect my-hello-openshift
```

```
...
```

```
"IPAddress": "172.17.0.4",...
```

```
"8080/tcp": null,
```

```
"8888/tcp": null
```

```
[root@hsyangpc hello-openshift]# curl -v 172.17.0.4:8080
```

```
Hello OpenShift!
```

# Hello-world

```
[root@hsyangpc hello-world]# docker run my-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.
```

```
.....
```

```
For more examples and ideas, visit:
```

```
https://docs.docker.com/engine/userguide/
```

```
[root@hsyangpc hello-world]#
```

# Dockerfile – Apache Webserver

```
[root@master ~]# cat Dockerfile
```

```
FROM docker.io/centos
```

```
USER root
```

```
RUN yum -y install tar unzip vi vim telnet
```

```
...
```

```
COPY files/jboss-ews-httpd-2.1.0.zip /tmp/
```

```
RUN cd /opt; unzip /tmp/jboss-ews-httpd-2.1.0.zip
```

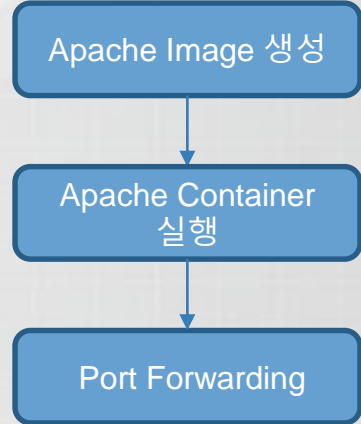
```
WORKDIR /opt/jboss-ews-2.1/httpd
```

```
RUN ./postinstall
```

```
...
```

```
EXPOSE 80
```

```
CMD ["/opt/jboss-ews-2.1/httpd/sbin/apachectl","-k","start","-D","FOREGROUND"]
```



# Docker Build

```
[root@master ~]# docker build -t ews21 .
```

```
...
```

```
[root@master ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
ews21	latest	47bd98336d1d	3 days ago	443.1 MB
docker.io/centos	6.6	12c9d795d85a	6 months ago	202.6 MB

# Container 실행 및 Port Forwarding

```
[root@master ~]# docker run -d --privileged -p 80:80 -h web1 ews21
```

# Apache Webserver 컨테이너 접속

# 실행중인 컨테이너 조회

```
[root@master ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d1e70a6efdac	ews21	"/opt/jboss-ews...!"	16 minutes ago	Up 16 minutes	80/tcp	tender_albattani

# 컨테이너 내부로 접속

```
[root@master ~]# docker exec -ti d1e70a6efdac /bin/bash
```

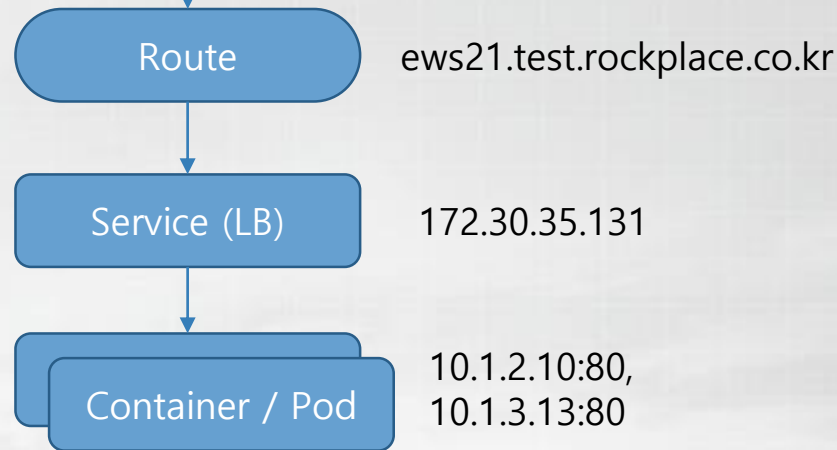
```
[root@tender_albattani~]#
```



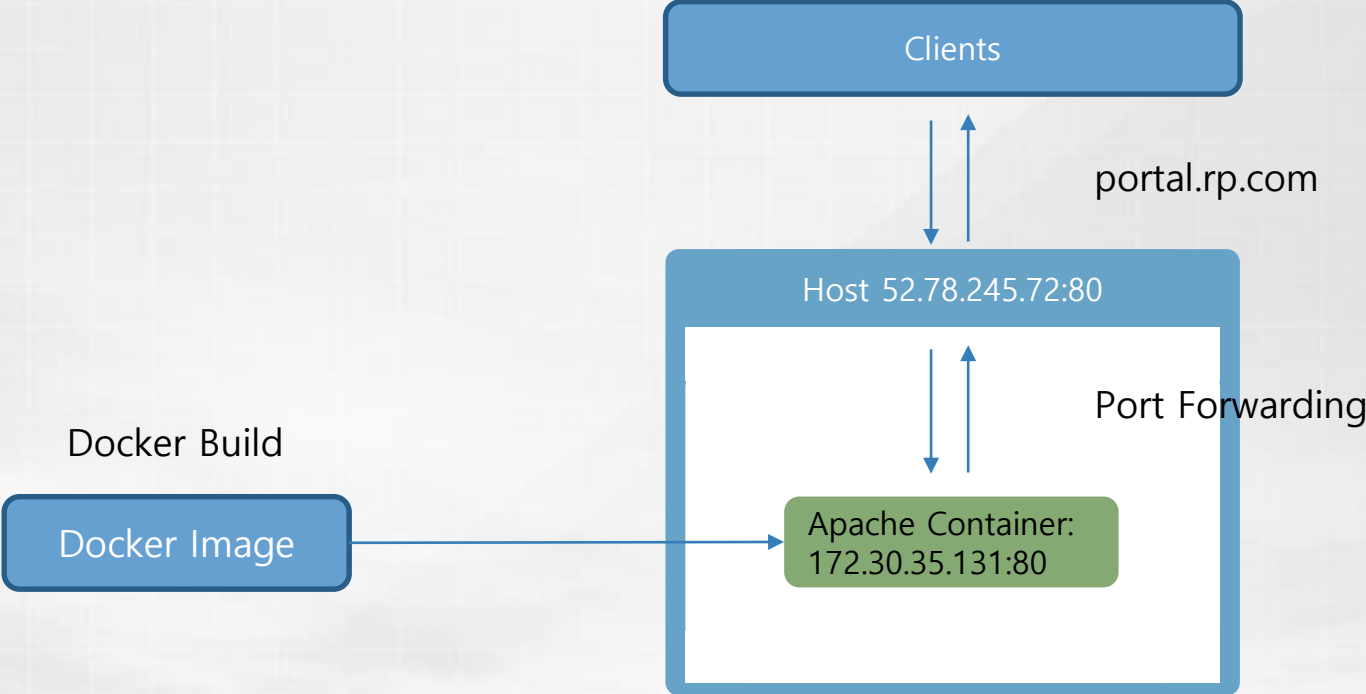
# Docker and Kubernetes of Openshift

Docker 만 사용하여  
컨테이너 서비스

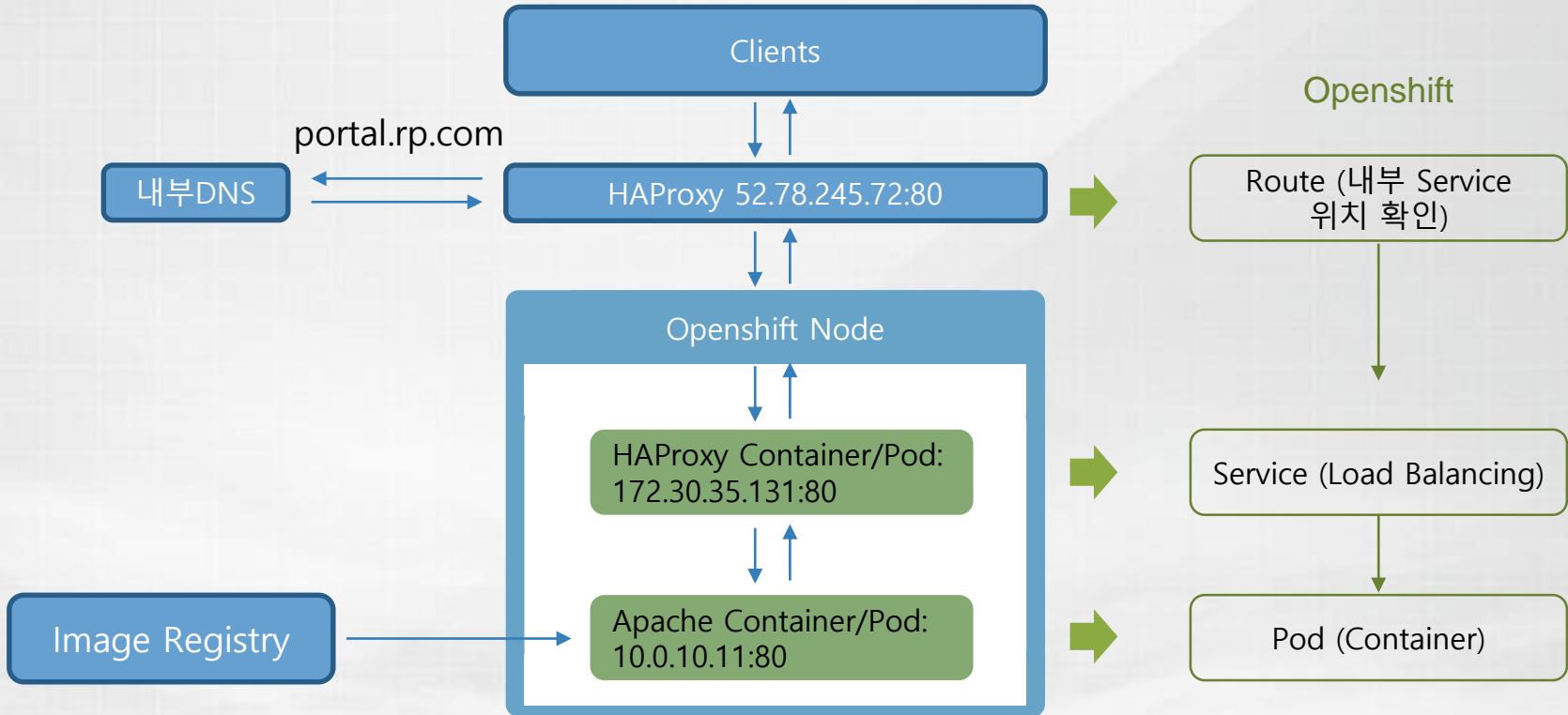
Kubernetes 를 사용하여  
컨테이너 서비스



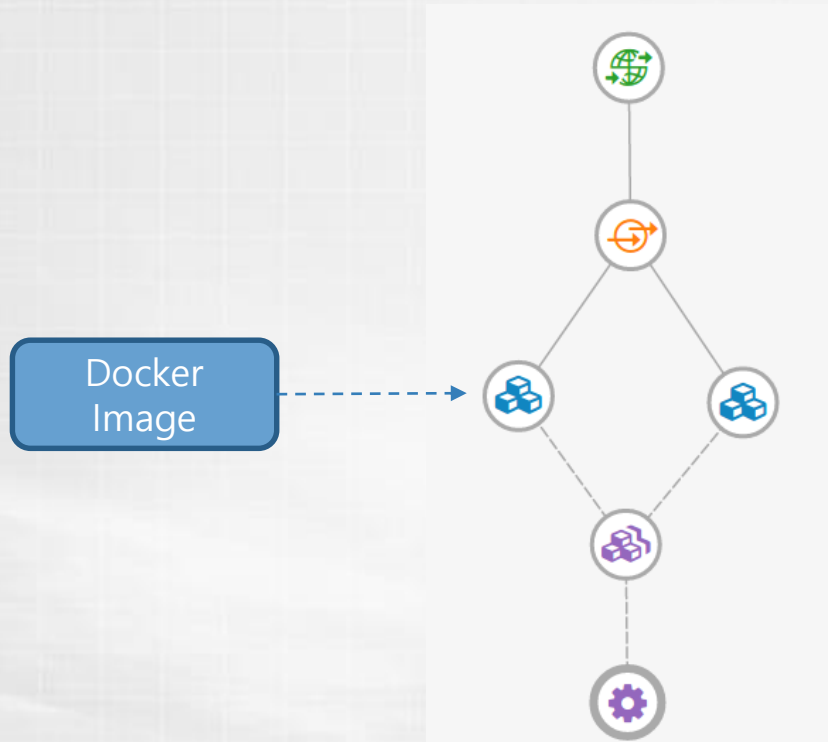
# Docker Working flow



# Docker and Kubernetes of Openshift



# Container Management by Openshift



Route: ews21

외부로부터의 진입점

ews21.test.rockplace.co.kr

service: ews21

Cluster IP (VIP)를 통한  
Loadbalancing

172.30.176.102

Pod

Kubernetes는  
Container를 Pod에 담아서 관리

ReplicationController

지정된 개수의 Pod가  
잘 돌고 있는지...

DeploymentConfig

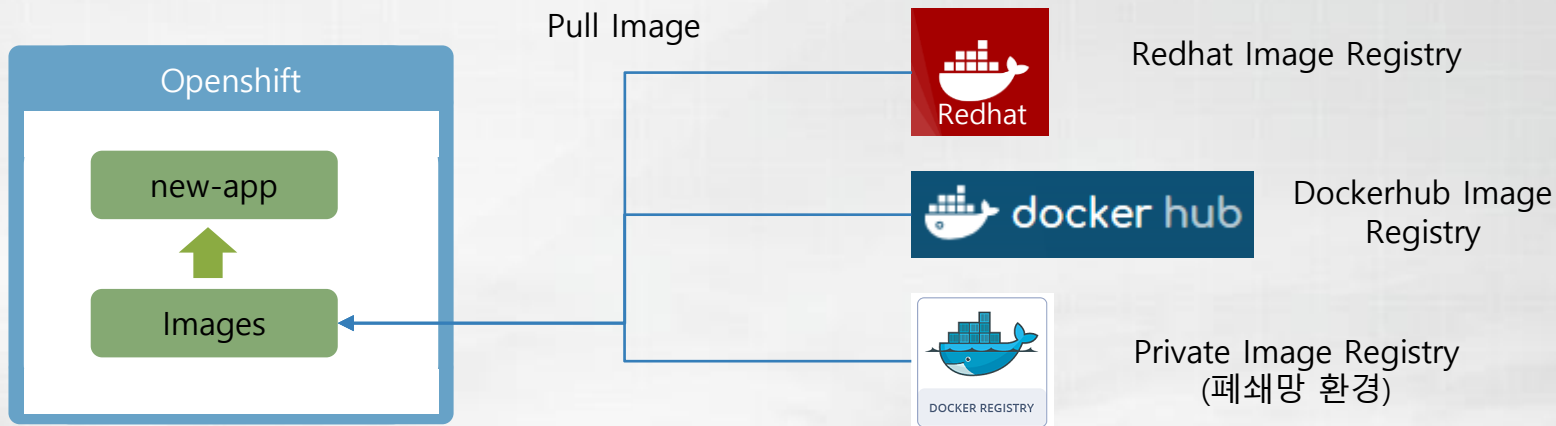
몇 개의 Pod를 만들지...  
어떻게 Pod를 배포 할지...

## 2. 이미지 다운로드 및 관리

- Image Registry
- Image Download & Push
- Openshift Image Stream
- Image Import

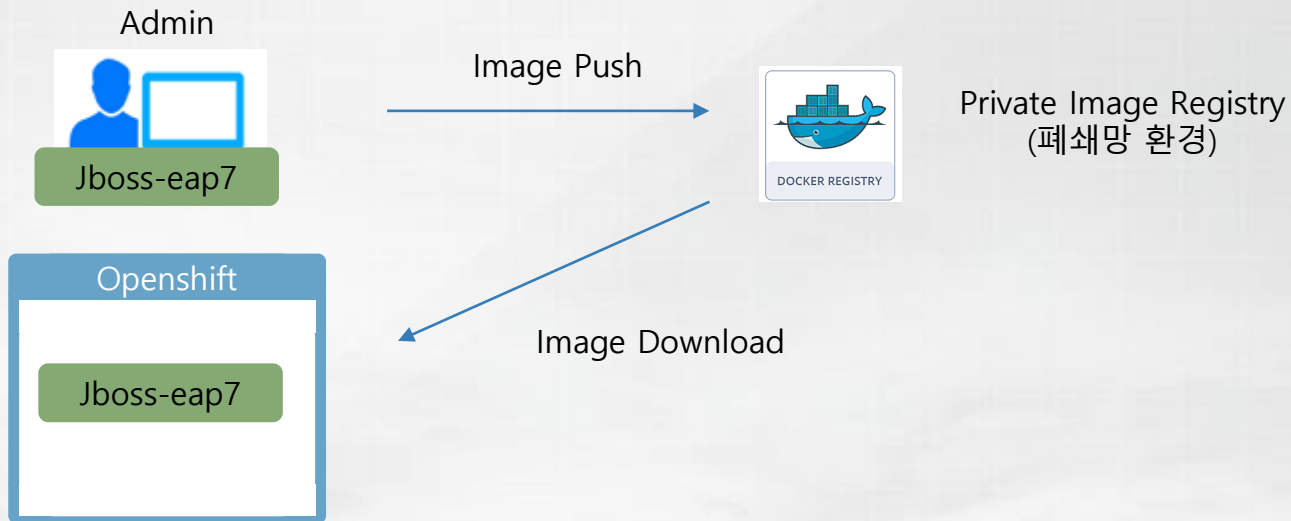
# Image Registry

- Docker Image는 Image Registry에 저장된다.
- Docker 명령을 이용하여 Image Registry로부터 이미지를 받을 수 있다.
- 폐쇄망에서는 내부에 Private Image Registry를 구성할 수 있다.



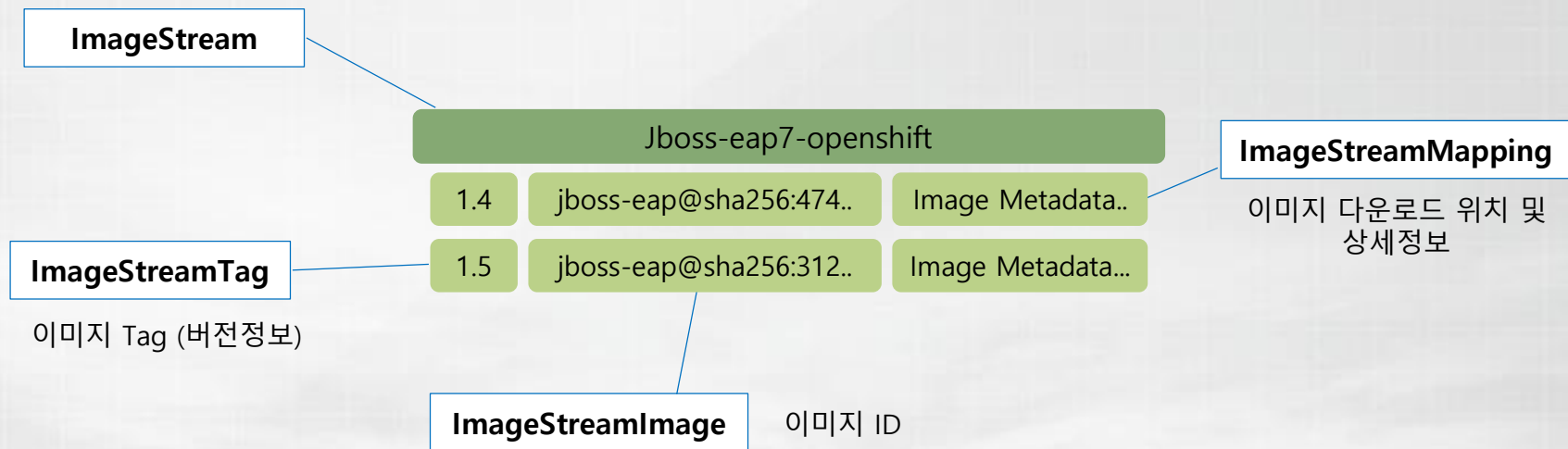
# Image Push & Download

- Admin이 외부 Registry 로부터 받거나 자체 제작한 이미지를 Image Registry 에 올린다.
- 내부의 Openshift에서 Image Registry로부터 이미지를 받아갈 수 있다.



# OpenShift Image Stream

- OpenShift에서는 ImageStream 형태로 이미지를 관리한다.
- 하나의 ImageStream 에는 Tag가 다른 여러 개의 이미지가 있고 각각 ImageStreamTag와 Image Metadata로 구별된다.





# Image Import

- ImageStream에 정의되어 있는 이미지를 다운로드 받아 Openshift 내부에 저장한다.
- ImageStream은 최초에 이미지가 들어 있지 않으므로 import-image 를 통해 이미지를 저장한다.
- 이미 이미지를 다운로드 받은 후에도 import-image를 통해 업데이트할 수 있다.

```
oc import-image jboss-eap70-openshift:1.4 -n openshift
```

Jboss-eap70-openshift

1.4

jboss-eap@sha256:474..

Image Metadata..

1.5

jboss-eap@sha256:312..

Image Metadata..

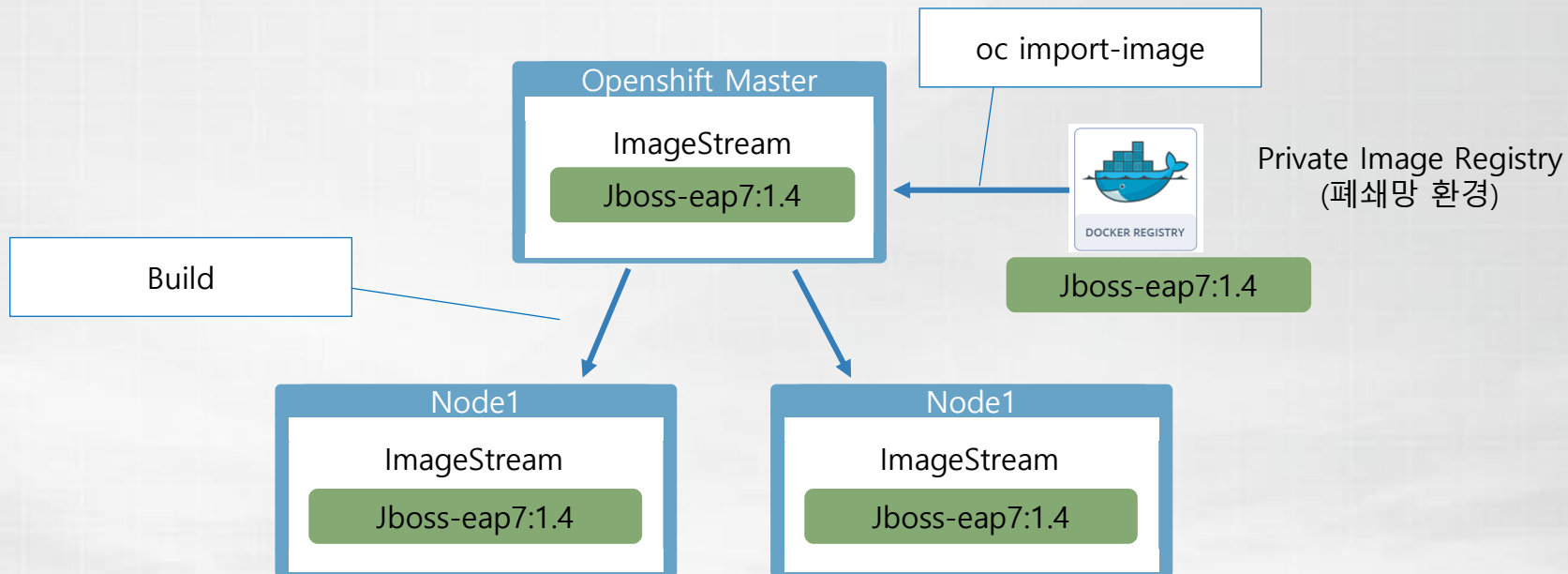


Private Image Registry  
(폐쇄망 환경)

```
"from": {  
  "kind": "DockerImage",  
  "name": "registry.access.redhat.com/jboss-eap-7/eap70-openshift:1.4"  
},
```

# Image Import

- import-image 를 수행한 이후에 Node에 수행중인 어플리케이션은 바로 변경되지 않는다.
- Build를 수행하면 새로운 이미지를 이용하여 어플리케이션을 생성한다.

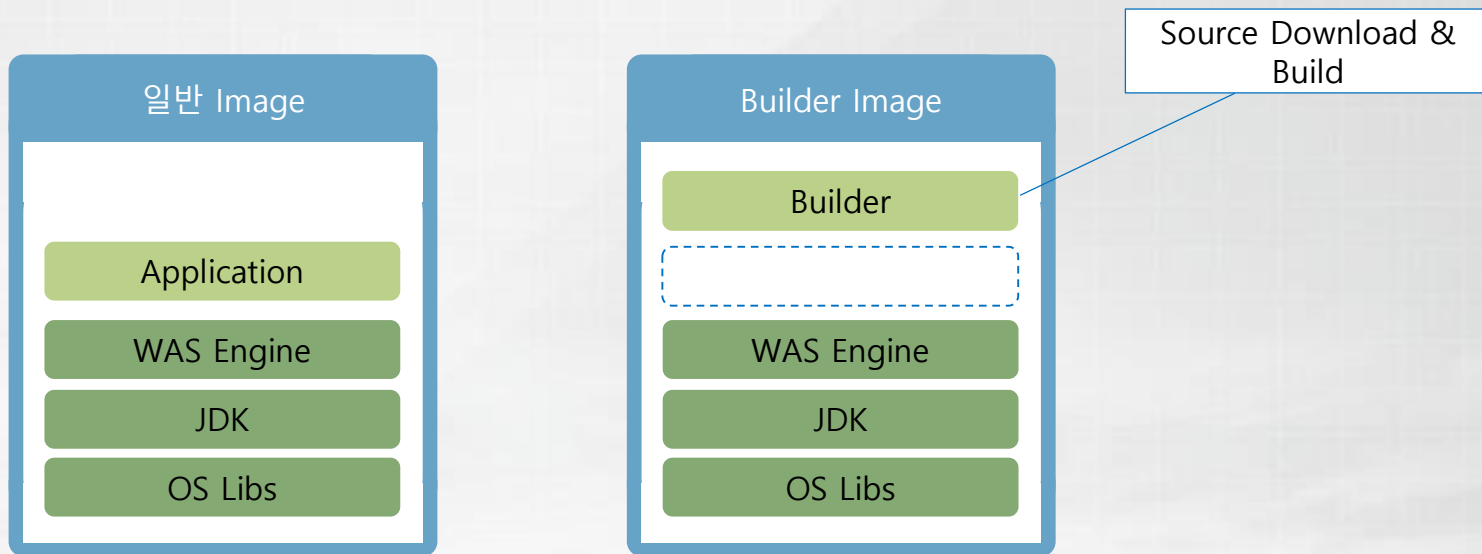


### 3. S2I Builder 이미지 및 커스텀 이미지 생성

- S2I Builder 이미지란?
- S2I Builder 이미지 작동방식
- 기존 이미지를 변경하여 커스텀 이미지 생성

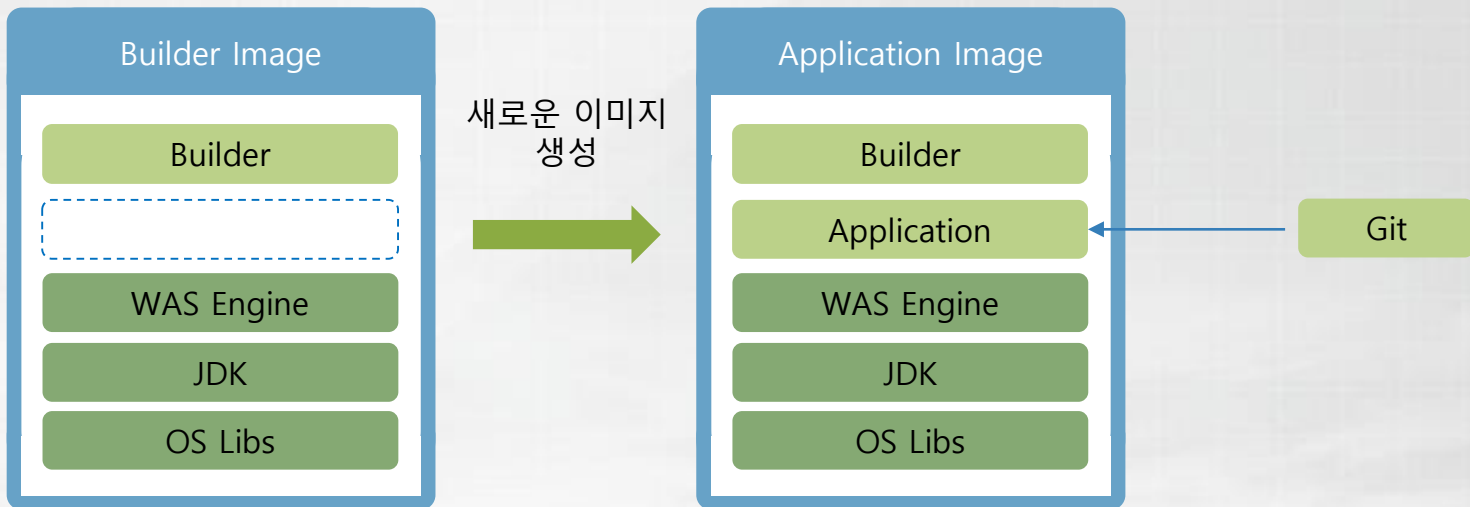
# S2I Builder

- 일반적으로 Java WAS 이미지는 OS Libs, JDK, WAS Engine, Application 으로 이루어진다.
- 이미지에서 변경이 지속적으로 일어나는 부분은 Application 이다.
- Builder Image에는 Application을 다운받아 Build 할 수 있는 Builder가 들어있다.



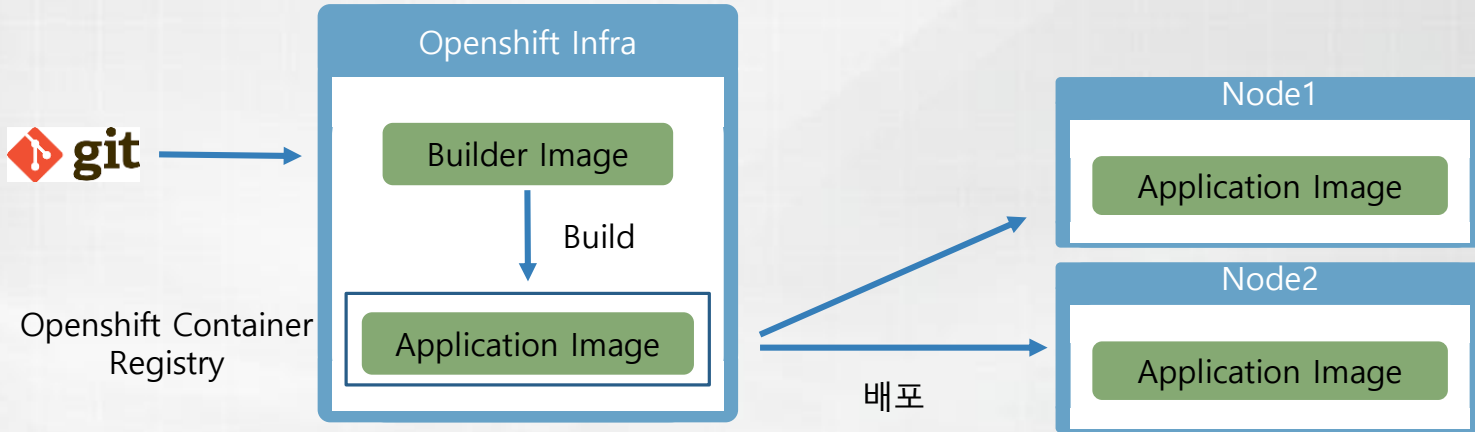
# S2I Builder

- 이미지를 컨테이너로 실행하면 Builder 가 작동하여 Application을 다운로드 받아 새로운 이미지를 만든다.



# S2I Builder

- Builder Image는 Git 서버로 부터 Application을 받아 새로운 Application Image를 생성한다.
- 새로운 Application Image는 Openshift 내부에 Openshift Container Registry에 저장된다.
- 각 Node로 Application Image를 배포한다.



# 기존 이미지 변경

- 기존 이미지를 변경하여 새로운 이미지를 만들 수 있다

Dockerfile

```
FROM jboss-eap-7/eap70-openshift

# Scripts for troubleshooting
COPY files/bin/launch/add_java_opts_append.sh /opt/eap/bin/launch/
...
# Install packages and change gclog name
USER 0

RUN sed -i "s/en_US/all/" /etc/yum.conf \
    && yum -y reinstall glibc-common \
    && yum clean all

# Maven Repo. Direcotry Permission
RUN rm -f /opteap/standalone/deployments/activemq-rar.rar \
    ...
    && chmod -R 775 /home/jboss/scripts

# Allow arbitrary
USER 185

ENV LANG=ko_KR.utf8 TZ=Asia/Seoul
```

Base Image

File 복사

root 유저로 변경

다국어 패키지 설치

불필요 파일 삭제

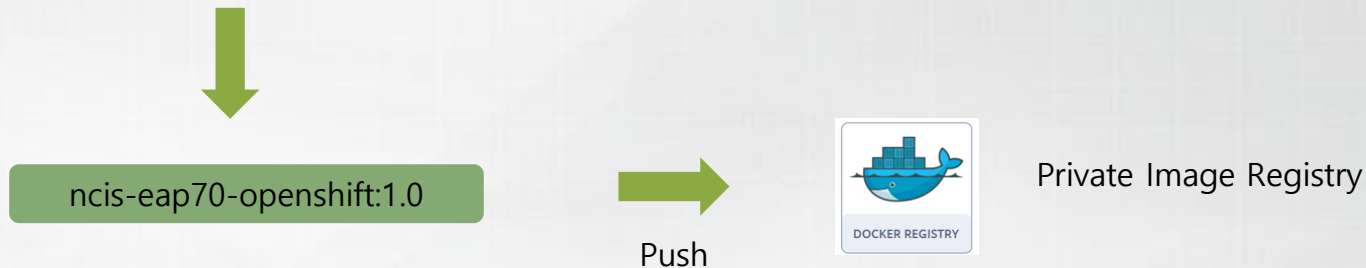
jboss 유저로 변경하여 실행

OS 문자셋 및 타임존설정

# 기존 이미지 변경

- Dockerfile을 이용하여 새로운 이미지를 생성한다.

```
# docker build -t ncis-eap70-openshift:1.0 .
```





# Guideline for Building Image

[https://docs.openshift.com/container-platform/3.3/creating\\_images/guidelines.html#openshift-container-platform-specific-guidelines](https://docs.openshift.com/container-platform/3.3/creating_images/guidelines.html#openshift-container-platform-specific-guidelines)

- Image 재사용
- Image에 Tag를 사용하여 유지관리
- Multiple Processes 회피
- Image Build 후 임시파일 제거
- Layer 최소화
- Support Arbitrary User IDs

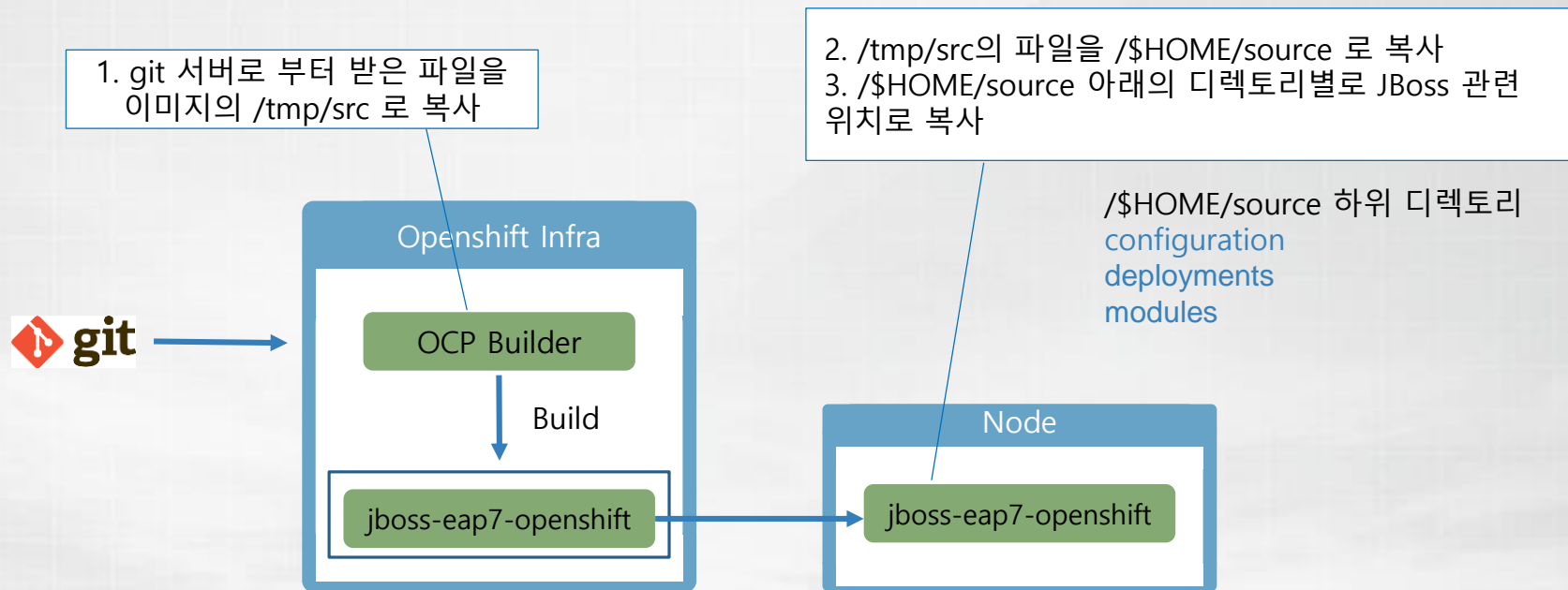
## 4. JBoss EAP 이미지

- Openshift용 JBoss EAP 이미지의 특징
- JBoss EAP 이미지 작동방식

# EAP7 for Openshift 이미지 특징

구분	내용
운영모드	Standalone
Admin Console	없음
JBOSS_HOME	/opt/eap
실행 스크립트	/opt/eap/bin/openshift-launch.sh
OS Version	Red Hat Enterprise Linux Server release 7.3 (Maipo)

# JBoss EAP7 이미지 작동방식



# 실행순서

순번	스크립트파일	용도
1	/usr/local/s2i/assemble	소스 및 설정파일을 JBoss 내부로 복사
2	/usr/local/s2i/run	openshift-launch.sh 실행
3	/opt/eap/bin/openshift-launch.sh	EAP 기동

# Git vs Container 파일위치

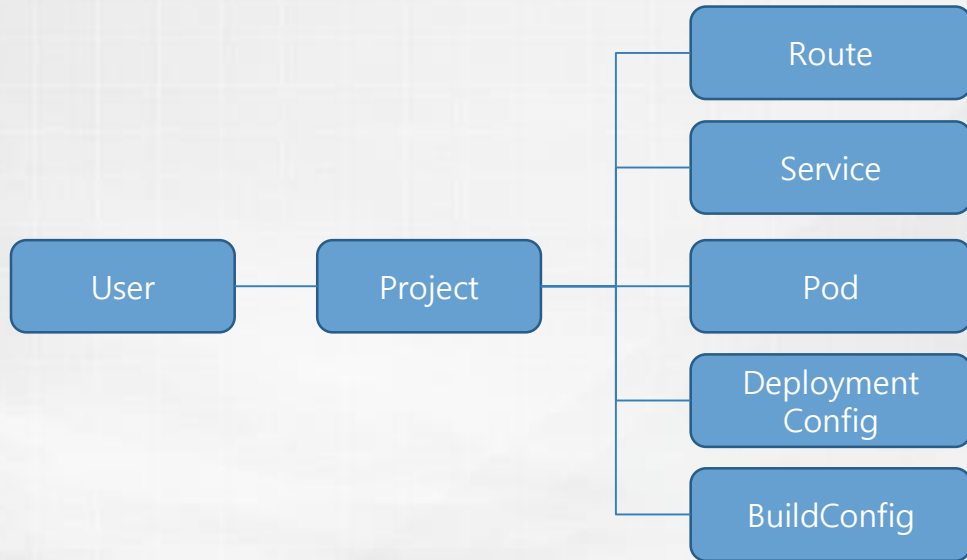
- Git 서버의 각 디렉토리 파일들은 아래와 같이 컨테이너 디렉토리로 복사된다.

구분	Git Directory	파일	Container Directory
소스	/	war ear rar jar	\$JBASS_HOME/deployments/
	/deployments/	war ear rar jar	\$JBASS_HOME/deployments/
설정파일	/configuration/	모든파일 (standalone-openshift.xml)	\$JBASS_HOME/configuration/
Module	/modules/	모든파일	\$JBASS_HOME/modules/

## 5. 이미지를 이용한 어플리케이션 생성

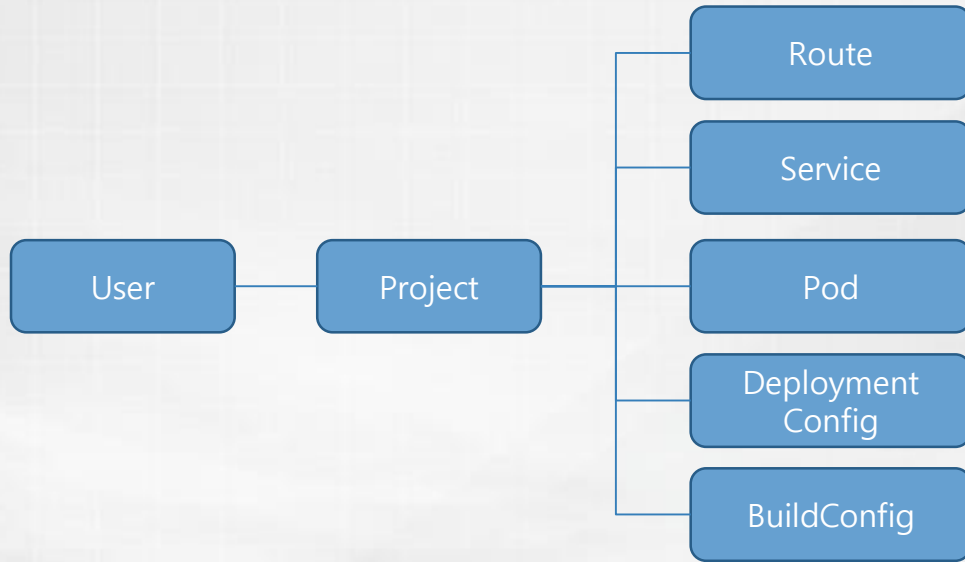
- **Openshift Resource**
- **Build**
- **Git**

# OpenShift User and Project Resources





# Build, Deployment, Pod



## Pod

Status	
<b>Status:</b>	🔄 Running
<b>Deployment:</b>	ncis, #14
<b>IP:</b>	172.40.4.61

## 배포 내역

Deployment	Status	Created
#14 (latest)	🔄 Active, 1 replica	8 days ago
#13	✔ Complete	8 days ago
#12	✔ Complete	15 days ago

## Build 내역

Build	Status	Duration
#14	✔ Complete	10 seconds
#13	✔ Complete	9 seconds
#12	✔ Complete	20 seconds

# BuildConfig

- BuildConfig는 설정된 내용을 기반으로 Build를 실행하는 역할을 한다.

## 가져올 Application 정보

Git Repository URL:

<http://gitlab.test.rp.co.kr/root/ncis-eap70.git>

Git Reference: `master`

Context Dir: `simple-app`

## 사용할 Image / 결과 Image

Build From :

`openshift/ncis-eap70-openshift:1.4`

Push To:

`myproject/ncis-eap70-openshift:latest`

# Git

- OCP Builder 는 원격지의 소스를 가져오기 위한 방법으로 Git 서버를 사용한다.
- Branch Reference는 master, Context는 /로 설정하면 git clone 시 Git 서버의 모든 히스토리까지 받게 되므로 소스의 용량이 커지고 오래 걸린다.
- 용량을 줄이려면 아래와 같이 Git Reference를 빈칸으로 남겨둬야 한다.

## 소스와 전체 히스토리 복사

Git Repository URL:

<http://gitlab.test.rp.co.kr/root/ncis-eap70.git>

Git Reference: master

Context Dir: /

## 소스와 최종 히스토리만 복사 (--depth=1)

Git Repository URL:

<http://gitlab.test.rp.co.kr/root/ncis-eap70.git>

Git Reference:

Context Dir: /

## 6. 배포 및 롤백

- Deployment
- Rollback

# Deployment

- Deployment는 Build에 의해 생성된 이미지를 Node로 배포하여 실행시키는 것을 말한다.

## 배포할 Image

Image Stream Tag: myproject/ncis:latest

Image가 변경되면 자동으로 배포: yes

설정이 바뀌면 자동으로 배포: yes

## 배포 방법

Strategy Type: Rolling

Max Num Unavailable Pods: 25% ( ¼ 씩 Rolling배포)

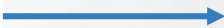
Replicas: 4 (동시에 유지할 Pod 개수)

# Rollback

- Rollback은 기존에 배포됐던 특정 시점으로 어플리케이션을 다시 배포하는 것을 말한다.
- Openshift는 기존 Deployment 이미지를 그대로 유지하고 있으므로 쉽게 Rollback이 가능하다.

## 배포 내역

Deployment	Status	Created
#14 (latest)	🔄 Active, 1 replica	8 days ago
#13	✔ Complete	8 days ago
#12	✔ Complete	15 days ago



Deployments » ncis » #12

ncis-12 created 15 days ago

app ncis openshift.io/deployment-config.name ncis

Details Environment Metrics Logs Events

**Status:** ✔ Complete

**Deployment Config:** ncis

**Status Reason:** image change

**Selectors:** deployment=ncis-12  
deploymentconfig=ncis

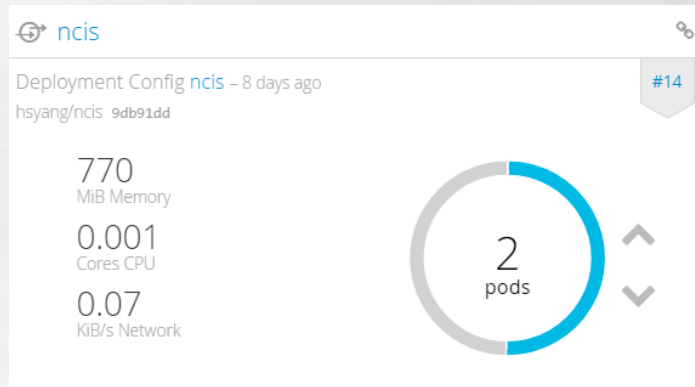
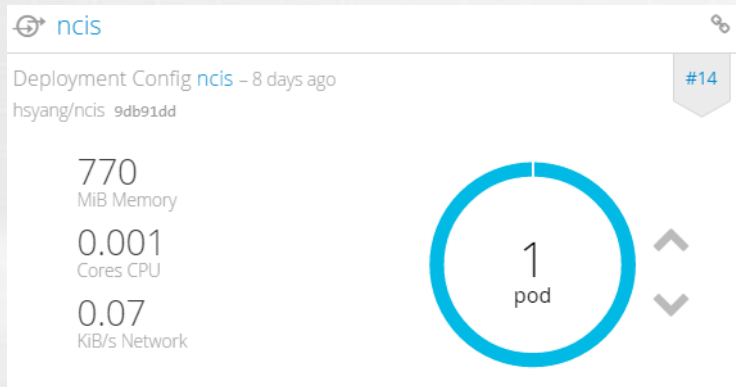
**Replicas:** 0 current / 0 desired

## 7. 서비스 확장

- **Manual Scale-Out**
- **Resource Limits**
- **Auto-Scaler**

# Manual Scale-out

- 실행될 Pod의 개수를 화살표로 조정할 수 있다.





# Resource Limit

- 컨테이너가 최대 사용할 수 있는 CPU와 Memory 를 지정할 수 있다.

## CPU

### Request

1000 millicores ▾

The minimum amount of CPU the container is guaranteed.

### Limit

2000 millicores ▾

The maximum amount of CPU the container is allowed to use when running.

[What are millicores?](#)

## Memory

### Request

1000 MiB ▾

The minimum amount of memory the container is guaranteed.

### Limit

2000 MiB ▾

The maximum amount of memory the container is allowed to use when running.

[What are MiB?](#)

# Auto-Scaler

- 지정된 CPU 사용율에 따라 Pod의 개수가 자동으로 증가 또는 감소하게 설정가능하다.

## \* Autoscaler Name

A unique name for the horizontal pod autoscaler within the project.

## Min Pods

The lower limit for the number of pods that can be set by the autoscaler. If not specified, defaults to 1.

## \* Max Pods

The upper limit for the number of pods that can be set by the autoscaler.

## CPU Request Target

The percentage of the CPU request that each pod should ideally be using. Pods will be added or removed periodically when CPU usage exceeds or drops below this target value. Defaults to 80%.

[Learn More](#) 

# EAP7 Application

NCIS <http://ncis-hsyang.ocp.seoul.test.rockplace.co.kr>

Build `ncis`, #14 ✔ Complete. 8 days ago [View Log](#) ✕

ⓘ `ncis` has containers without health checks, which ensure your application is running correctly. [Add Health Checks](#) ✕

[↻](#) `ncis` 🔗

Deployment Config `ncis` – 7 minutes ago #15  
hsyang/ncis 9db91dd

680 MiB Memory  
0.006 Cores CPU  
0.07 KiB/s Network

1 pod

Autoscaled: min: 1, max: 4

No grouped services.

No services are grouped with `ncis`. Add a service to group them together.

[Group Service](#)

**rock**PLACE

감사합니다

[middleware@rockplace.co.kr](mailto:middleware@rockplace.co.kr)