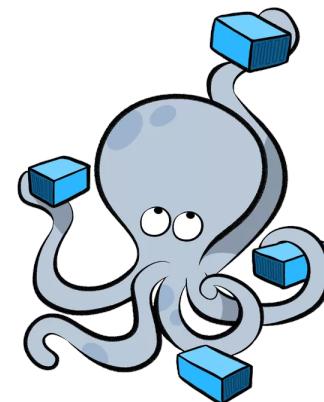


#### *4. Kubernetes*

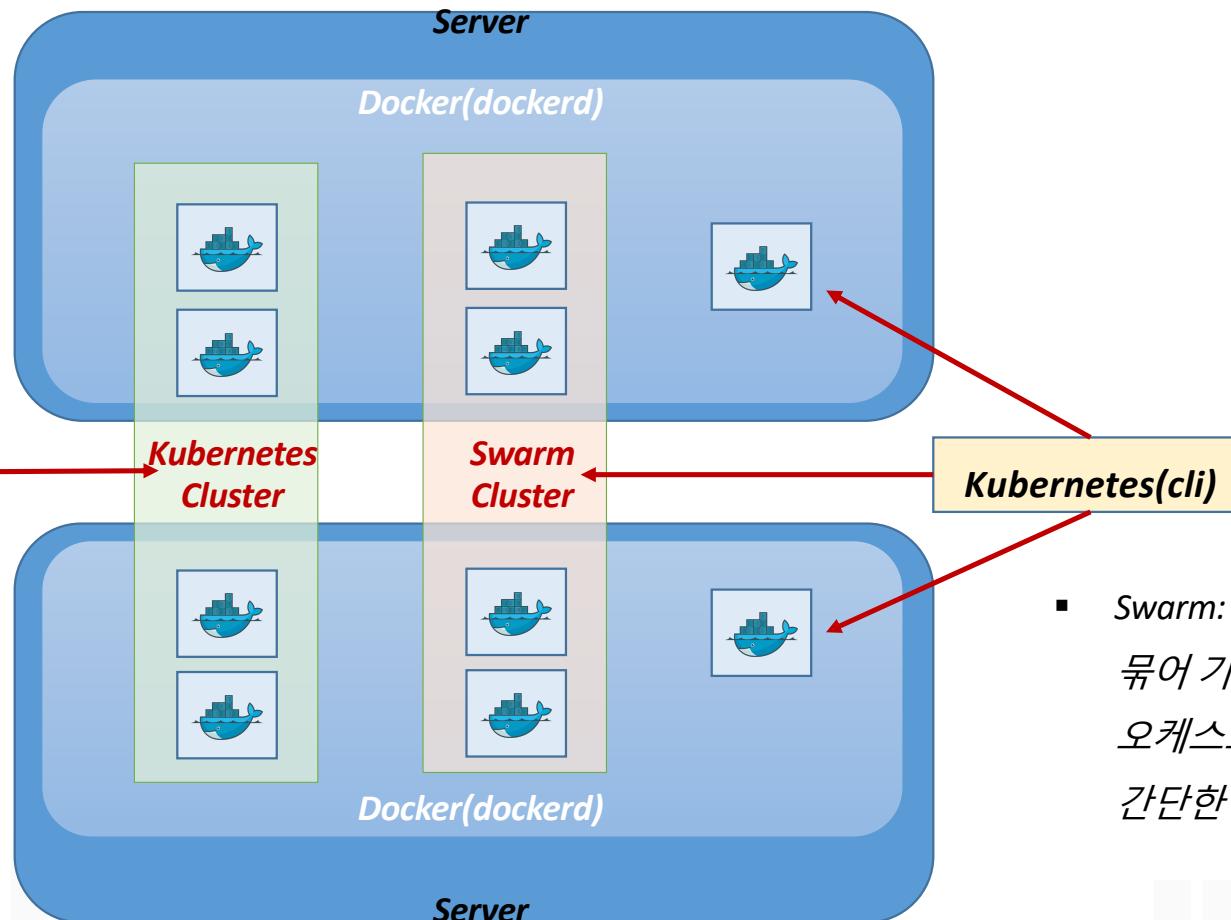
## ■ *Kubernetes*

- **Docker Container 운영을 자동화하기 위한 컨테이너 오케스트레이션 툴**
  - 컨테이너 배포 및 배치 전략
  - Scale in/Scale out
  - Service discovery
  - 기타 운영
- 구글의 *Borg* 프로젝트에서 시작
- 2017년 Docker에 정식으로 통합된 사실상 표준
  - DockerCon EU 2017



**kubernetes**

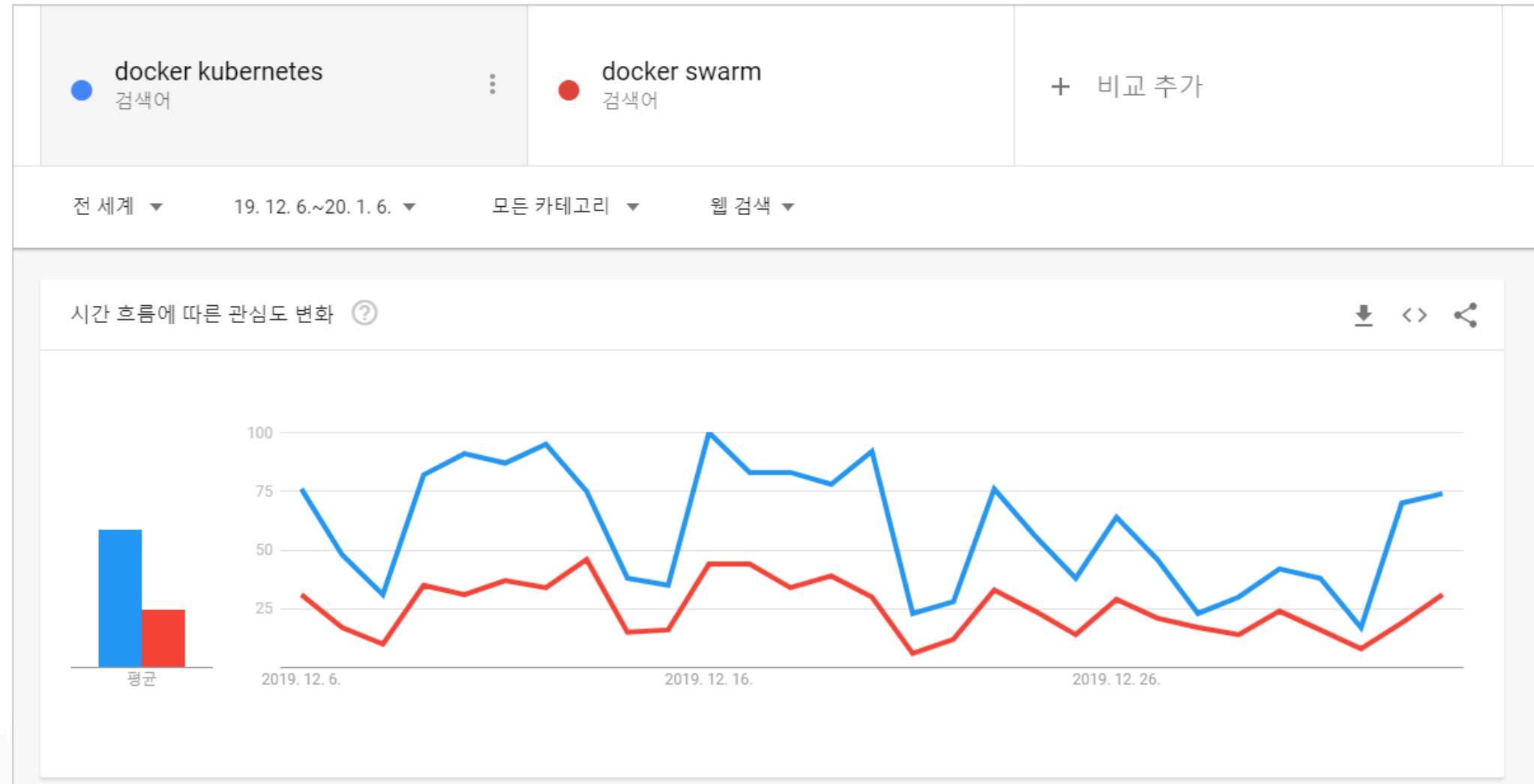
## ■ Kubernetes



- Kubernetes: Swarm 보다 충실한 기능을 갖춘 컨테이너 오케스트레이션 시스템

- Swarm: 여러 대의 호스트를 묶어 기초적인 컨테이너 오케스트레이션 기능 제공, 간단한 멀티 컨테이너 구축

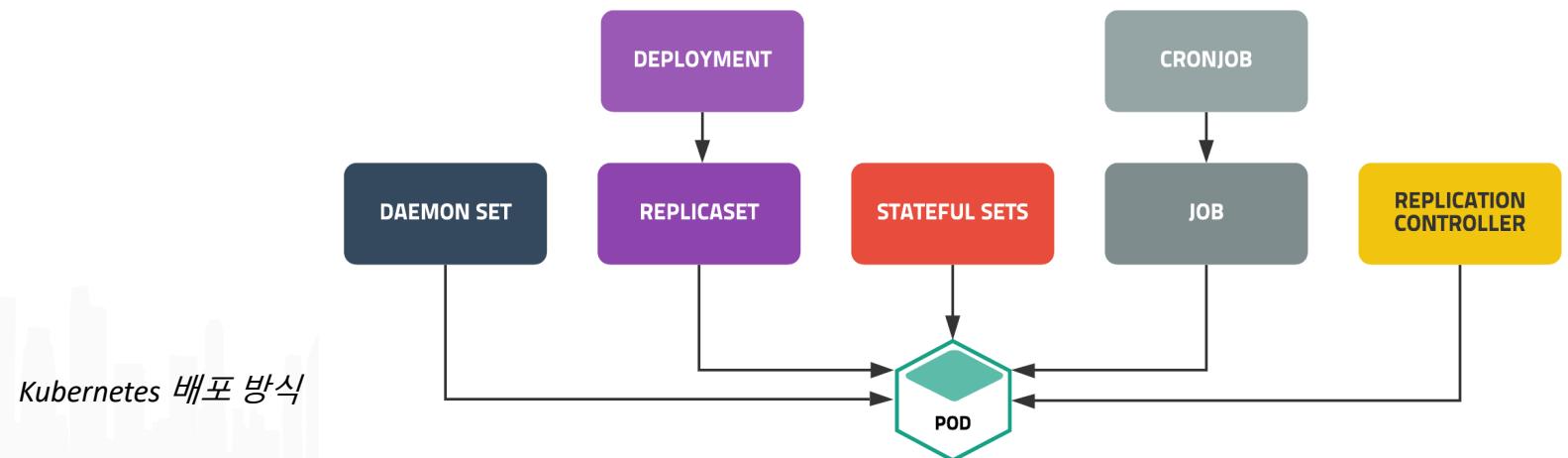
## ■ Kubernetes



## ■ Kubernetes

### ■ Kubernetes 설치

- Minikube
  - 이전에는 로컬 환경에서 Kubernetes를 구축하기 위해 사용
  - 2018년부터 안정버전에서도 설치 가능
  - 윈도우 설치)
    - 바탕화면 트레이 > 도커 아이콘 (오른쪽 클릭) > Setting 메뉴 > Kebernetes 탭
    - Enable Kubernetes 선택 → Install



## ■ *Kubernetes*

### ▪ *kubectl 설치*

- Kubernetes를 다루기 위한 Command Line Interface
- 원도우 설치)
  - <https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/windows/amd64/kubectl.exe>
  - 다운로드, PATH 추가
- \$ kubectl version

```
▶ kubectl version
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.0", GitCommit:"70132b0f130acc0bed193d9ba59dd186f0e634cf",
GitTreeState:"clean", BuildDate:"2019-12-07T21:20:10Z", GoVersion:"go1.13.4", Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.6", GitCommit:"96fac5cd13a5dc064f7d9f4f23030a6aeface6cc",
GitTreeState:"clean", BuildDate:"2019-08-19T11:05:16Z", GoVersion:"go1.12.9", Compiler:"gc", Platform:"linux/amd64"}
```

## ■ *Kubernetes*

### ■ *Dashboard 설치*

- Kubernetes에 배포된 컨테이너 등에 대한 정보를 보여주는 관리 도구
- \$ kubectl apply -f

<https://raw.githubusercontent.com/kubernetes/dashboard/v1.8.3/src/deploy/recommended/kubernetes-dashboard.yaml>

- \$ kubectl get pod --namespace=kube-system -l k8s-app=kubernetes-dashboard

NAME	READY	STATUS	RESTARTS	AGE
kubernetes-dashboard-6fd7f9c494-2bkjp	1/1	Running	0	30h

- \$ kubectl proxy (웹 브라우저로 대시보드를 사용할 수 있도록 프록시 서버 설정)

<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

# Kubernetes

## Dashboard 설치

The screenshot shows the Kubernetes Dashboard interface. On the left, there's a sidebar with navigation links like Overview, Workloads, Discovery and Load Balancing, and Config and Storage. The main area has two sections: 'Services' and 'Secrets'. In the 'Services' section, there's one entry for 'kubernetes' with a Cluster IP of 10.96.0.1. It has internal endpoints at kubernetes:443 TCP and external endpoints at kubernetes:0 TCP. The 'Secrets' section lists a single secret named 'default-token-lrjh' of type kubernetes.io/service-account-token, which is a day old.

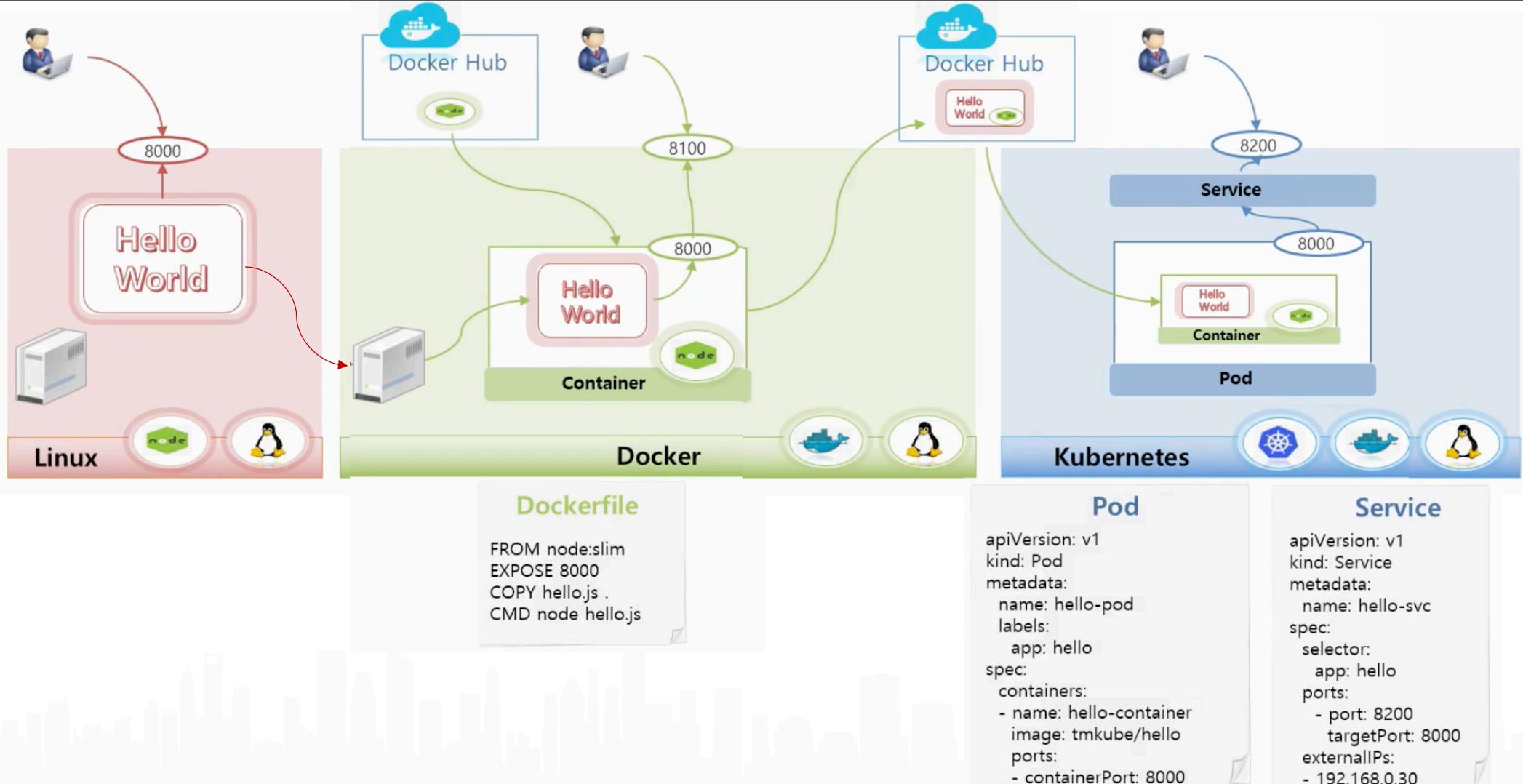
Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
kubernetes	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	52 seconds

Name	Type	Age
default-token-lrjh	kubernetes.io/service-account-token	a day

## ■ Kubernetes 주요 개념

Resource or Object	용도
Node	컨테이너가 배치되는 서버
Namespace	쿠버네티스 클러스터 안의 가상 클러스터
Pod	컨테이너의 집합 중 가장 작은 단위, 컨테이너의 실행 방법 정의
Replica Set	같은 스펙을 갖는 파드를 여러 개 생성하고 관리하는 역할
Deployment	레플리카 세트의 리비전을 관리
Service	파드의 집합에 접근하기 위한 경로를 정의
Ingress	서비스를 쿠버네티스 클러스터 외부로 노출
ConfigMap	설정 정보를 정의하고 파드에 전달
Persistent Volume	파드가 사용할 스토리지의 크기 및 종류를 정의
Persistent Volume Claim	퍼시스턴트 볼륨을 동적으로 확보

## Kubernetes Demo



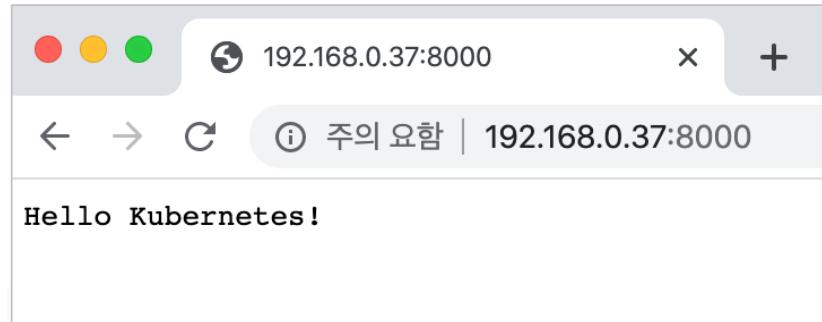
## ■ Kubernetes Demo

### ■ Linux version

```
$ vi hello.js
```

```
1 var http = require('http');
2 var content = function(req, resp) {
3   resp.end("Hello Kubernetes!" + "\n");
4   resp.writeHead(200);
5 }
6 var w = http.createServer(content);
7 w.listen(8000);
```

```
$ node hello.js
```

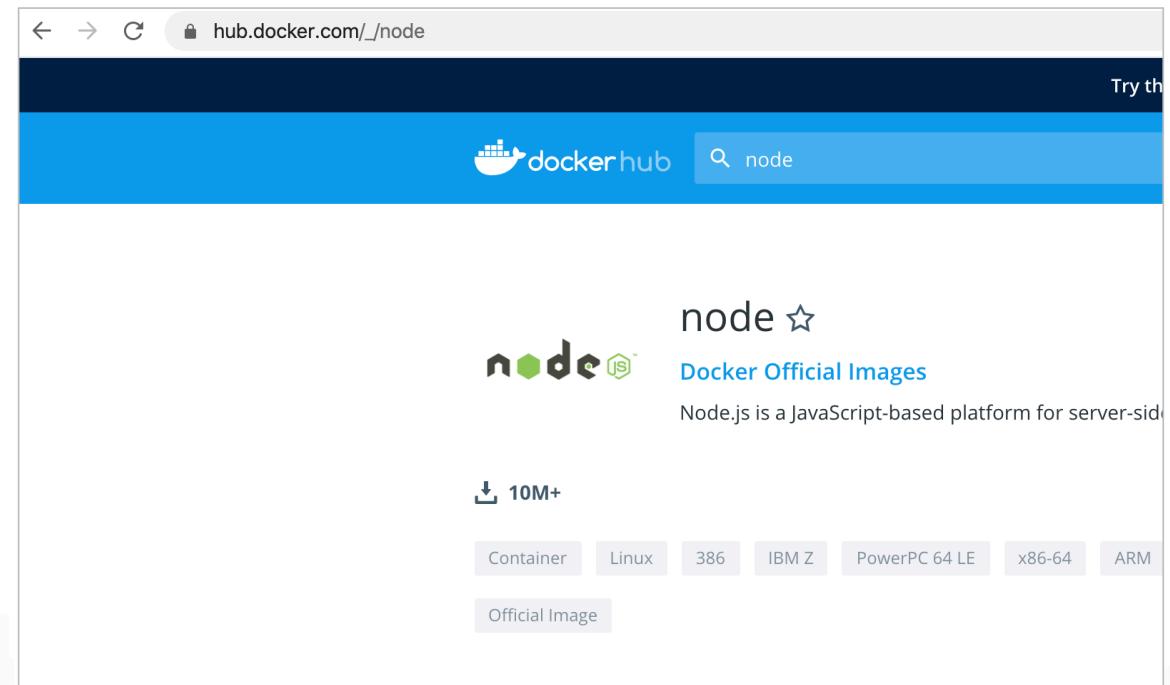


## ■ Kubernetes Demo

### ■ Docker version

\$ vi Dockerfile.js

```
1 FROM node:slim
2
3 EXPOSE 8000
4
5 COPY hello.js .
6
7 CMD node hello.js
```



## ■ Kubernetes Demo

### ■ Docker version

```
$ docker build -t edowon0623/hello .
```

```
$ docker images
```

```
▶ docker build -t edowon0623/hello .
Sending build context to Docker daemon 3.072kB
Step 1/4 : FROM node:slim
slim: Pulling from library/node
804555ee0376: Already exists
2706bdf80250: Pull complete
3a1861ab5a61: Pull complete
2a2939395b29: Pull complete
738ba111f3fd: Pull complete
Digest: sha256:65600ac92bd94a647ddfebea5ef141b44ef02ffed9e431389000b245c3b84330
Status: Downloaded newer image for node:slim
    --> b515d98fd8cd
Step 2/4 : EXPOSE 8000
    --> Running in 696e95575d7f
Removing intermediate container 696e95575d7f
    --> c19d3c1cc743
Step 3/4 : COPY hello.js .
    --> 786edbe52048
Step 4/4 : CMD node hello.js
    --> Running in efd1148f0070
Removing intermediate container efd1148f0070
    --> 2eb447ee279a
Successfully built 2eb447ee279a
Successfully tagged edowon0623/hello:latest
```

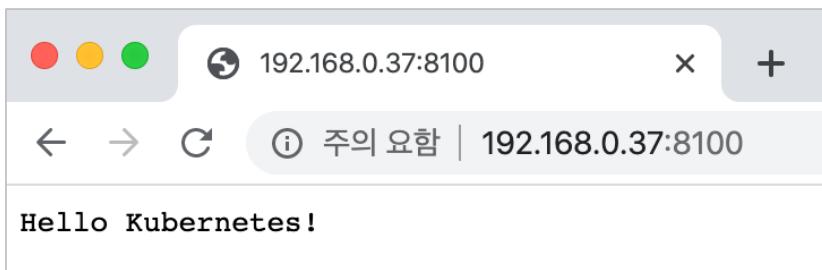
## ■ Kubernetes Demo

### ■ Docker version

```
$ docker run -d -p 8100:8000 edowon0623/hello
```

```
▶ docker run -d -p 8100:8000 edowon0623/hello:latest
46cf06dc11297c35eae06c88202e8255b7ba93798c7e53be00594fcff20e344c
dowon@DOWON-MacBook ▶ ~/Desktop/Work/docker/kubernetes/getstarted▶
```

```
$ docker exec -it edowon0623/hello bash
```



```
▶ docker exec -it 46cf06dc1129 bash
root@46cf06dc1129:/# ls -al
total 76
drwxr-xr-x  1 root root 4096 Jan  5 23:04 .
drwxr-xr-x  1 root root 4096 Jan  5 23:04 ..
-rw-r--r--  1 root root   0 Jan  5 23:04 .dockerenv
drwxr-xr-x  2 root root 4096 Dec 24 00:00 bin
drwxr-xr-x  2 root root 4096 Sep  8 10:51 boot
drwxr-xr-x  5 root root 340 Jan  5 23:04 dev
drwxr-xr-x  1 root root 4096 Jan  5 23:04 etc
-rw-r--r--  1 root root 179 Jan  5 22:54 hello.js
drwxr-xr-x  1 root root 4096 Dec 28 23:33 home
drwxr-xr-x  1 root root 4096 Dec 24 00:00 lib
drwxr-xr-x  2 root root 4096 Dec 24 00:00 lib64
drwxr-xr-x  2 root root 4096 Dec 24 00:00 media
drwxr-xr-x  2 root root 4096 Dec 24 00:00 mnt
drwxr-xr-x  1 root root 4096 Dec 28 23:35 opt
dr-xr-xr-x 253 root root   0 Jan  5 23:04 proc
drwx-----  1 root root 4096 Dec 28 23:34 root
drwxr-xr-x  3 root root 4096 Dec 24 00:00 run
drwxr-xr-x  2 root root 4096 Dec 24 00:00 sbin
drwxr-xr-x  2 root root 4096 Dec 24 00:00 srv
dr-xr-xr-x 13 root root   0 Dec 30 16:41 sys
drwxrwxrwt  1 root root 4096 Dec 28 23:34 tmp
drwxr-xr-x  1 root root 4096 Dec 24 00:00 usr
drwxr-xr-x  1 root root 4096 Dec 24 00:00 var
```

## ■ *Kubernetes Demo*

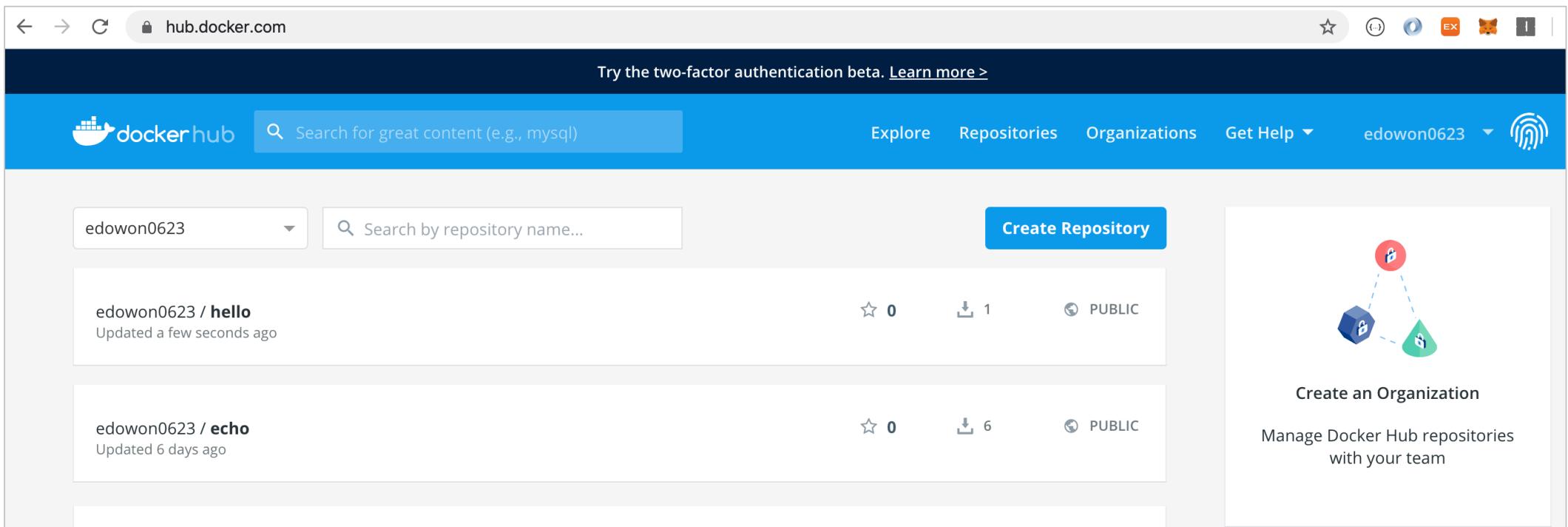
### ■ *Kubernetes version*

```
▶ docker login
Authenticating with existing credentials...
Login Succeeded
dowon@DOWON-MacBook ~ ~/Desktop/Work/docker/kubernetes/getstarted▶
```

```
▶ docker push edowon0623/hello:latest
The push refers to repository [docker.io/edowon0623/hello]
ce63c2b86199: Pushed
0bd549a35bb4: Mounted from library/node
a954d4bc6a8d: Mounted from library/node
a2357d53df3b: Mounted from library/node
62dac45972d5: Mounted from library/node
814c70fdae62: Mounted from library/node
latest: digest: sha256:c46636d9b118c67c2248b59bb6df93f16d12004be86f41db44cdbca3bc521396 size: 1574
```

## ■ Kubernetes Demo

### ■ Kubernetes version



The screenshot shows the Docker Hub homepage. At the top, there's a navigation bar with links for Explore, Repositories, Organizations, Get Help, and a user dropdown for edowon0623. A search bar at the top right allows searching for great content like mysql. Below the navigation, a user dropdown shows edowon0623 and a search bar for repository names. Two repositories are listed: edowon0623/hello (updated a few seconds ago) and edowon0623/echo (updated 6 days ago). Both repositories have 0 stars, 1 download, and are public. To the right, there's a promotional section for creating an organization, featuring icons for locks and keys, and text encouraging team management.

Try the two-factor authentication beta. [Learn more >](#)

dockerhub

Search for great content (e.g., mysql)

Explore    Repositories    Organizations    Get Help

edowon0623

edowon0623 / hello

Updated a few seconds ago

0    1    PUBLIC

edowon0623 / echo

Updated 6 days ago

0    6    PUBLIC

Create Repository

Create an Organization

Manage Docker Hub repositories with your team

## Kubernetes Demo

### Kubernetes version

The screenshot shows the Kubernetes Dashboard running at `localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/overview?namespace=default`. The interface has a blue header with the title "kubernetes". On the left, there's a sidebar with "Overview" selected, and other tabs like "Namespaces", "Nodes", "Persistent Volumes", "Roles", and "Storage Classes". Below the sidebar, there's a "Namespace" dropdown set to "default". The main content area has two sections: "Services" and "Secrets".

**Services**

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
kubernetes	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	7 hours

**Secrets**

Name	Type	Age
default-token-lrjh	kubernetes.io/service-account-token	a day

## ■ Kubernetes Demo

### ■ Kubernetes version

The screenshot shows the Kubernetes UI for creating resources. The top navigation bar includes a logo, the word "kubernetes", a search bar, and a "+ CREATE" button. A red box highlights the "+ CREATE" button. Below the header is a blue bar labeled "Resource creation". On the left, a sidebar lists "Cluster" and "Namespaces" under "Cluster", and "Nodes", "Persistent Volumes", "Roles", and "Storage Classes" under "Namespaces". Under "Namespaces", "default" is selected. The main area has tabs for "CREATE FROM TEXT INPUT" (which is active), "CREATE FROM FILE", and "CREATE AN APP". A red arrow points from the "CREATE FROM TEXT INPUT" tab to the text input area. Another red box highlights the text input area, which contains the following YAML code:

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: hello-pod
5   labels:
6     app: hello
7 spec:
8   containers:
9     - name: hello-container
10    image: kubetm/hello
11    ports:
12      - containerPort: 8000
13 |
```

## ■ Kubernetes Demo

### ■ Kubernetes version

The screenshot shows the Kubernetes Dashboard interface. At the top left is the Kubernetes logo. To its right is a search bar with a magnifying glass icon and the word "Search". On the far right is a blue button with a plus sign and the text "CREATE". Below the header is a blue navigation bar with the text "Overview".

The main area has two columns. The left column, under the heading "Cluster", contains links for Namespaces, Nodes, Persistent Volumes, Roles, and Storage Classes. Under "Namespace", there is a dropdown menu set to "default". The right column, under the heading "Workloads", contains a link for "Workloads Statuses".

The "Workloads Statuses" section features a large orange pie chart with a single slice labeled "100.00%" and the word "Pods" below it.

The "Pods" section has a table with the following columns: Name (sorted by name), Node, Status (sorted by status), Restarts, and Age. One pod is listed: "hello-pod" is pending, with 0 restarts and 0 seconds age.

At the bottom right of the dashboard, the text "19/106" is visible.

## Kubernetes Demo

### Kubernetes version

The screenshot shows the Kubernetes UI interface. At the top, there's a navigation bar with a search bar and two buttons: 'EXEC' (highlighted with a red box) and 'LOGS'. Below the navigation, the path 'Workloads > Pods > hello-pod' is shown. On the left, a sidebar lists various cluster components: Cluster (Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes), Namespace (default selected), Overview, Workloads (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods selected), Replica Sets, and Replication Controllers. The main content area has three sections: 'Details' (Name: hello-pod, Namespace: default, Labels: app: hello, Network: Node: docker-desktop, IP: 10.1.0.41, Creation Time: 2020-01-05T23:12 UTC, Status: Running, QoS Class: BestEffort); 'Containers' (hello-container, Image: kuberm/hello, Environment variables: -, Commands: -, Args: -); and 'Conditions' (a table with columns: Type, Status, Last heartbeat time, Last transition time, Reason, Message). The 'Containers' section is currently active.

## Kubernetes Demo

### Kubernetes version

The screenshot shows a Kubernetes web interface with a sidebar on the left and a main content area on the right.

**Left Sidebar:**

- Cluster
- Namespaces
  - default
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

**Right Content Area:**

Shell in hello-container ▾ in hello-pod

```
root@hello-pod:/# ls -al
total 76
drwxr-xr-x  1 root root 4096 Jan  5 23:12 .
drwxr-xr-x  1 root root 4096 Jan  5 23:12 ..
-rw-r--r--  1 root root 179 Nov 11 19:47 hello.js
drwxr-xr-x  1 root root 4096 Oct 17 03:09 home
drwxr-xr-x  1 root root 4096 Oct 14 00:00 lib
drwxr-xr-x  2 root root 4096 Oct 14 00:00 lib64
drwxr-xr-x  2 root root 4096 Oct 14 00:00 media
drwxr-xr-x  2 root root 4096 Oct 14 00:00 mnt
drwxr-xr-x  1 root root 4096 Nov  8 02:27 opt
dr-xr-xr-x 260 root root   0 Jan  5 23:12 proc
drwx----- 1 root root 4096 Nov  8 02:27 root
drwxr-xr-x  1 root root 4096 Jan  5 23:12 run
drwxr-xr-x  1 root root 4096 Jan  5 23:20 sbin
drwxr-xr-x  2 root root 4096 Oct 14 00:00 srv
dr-xr-xr-x 13 root root   0 Dec 30 16:41 sys
drwxrwxrwt  1 root root 4096 Jan  5 23:20 tmp
drwxr-xr-x  1 root root 4096 Oct 14 00:00 usr
drwxr-xr-x  1 root root 4096 Oct 14 00:00 var
root@hello-pod:/# curl http://localhost:8000
Hello Kubernetes!
root@hello-pod:/#
```

## Kubernetes Demo

### Kubernetes version

#### - Service 생성

The screenshot shows the Kubernetes UI for creating a new resource. The top navigation bar has a 'kubernetes' logo and a search bar. On the right, there is a '+ CREATE' button with a red box around it. The main area is titled 'Resource creation' and has a sidebar with categories like Workloads, Config and Storage, and Settings. The 'Workloads' section is expanded, showing options like Cron Jobs, Daemon Sets, Deployments, etc. In the center, there are three creation methods: 'CREATE FROM TEXT INPUT' (selected), 'CREATE FROM FILE', and 'CREATE AN APP'. Below these is a text area with placeholder text: 'Enter YAML or JSON content specifying the resources to deploy to the currently selected namespace.' A red arrow points from the 'CREATE FROM TEXT INPUT' label to this text area. Another red box surrounds the text area where the YAML code is pasted. The YAML code is as follows:

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: hello-svc
5 spec:
6   selector:
7     app: hello
8   ports:
9     - port: 8200
10    targetPort: 8000
11    type: LoadBalancer
12    externalIPs:
13      - 172.30.56.67
```

At the bottom of the central area are 'UPLOAD' and 'CANCEL' buttons.

## ■ Kubernetes Demo

### ■ Kubernetes version

- Service 생성

kubernetes

Discovery and load balancing > Services

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

- Ingresses
- Services**

Config and Storage

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
hello-svc	-	10.97.208.182	hello-svc:8200 TCP hello-svc:31670 TCP	localhost:8200 172.30.56.67:8200	-
kubernetes	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	8 days

localhost:8200

Hello Kubernetes!