쿠버네티스

엔지니어링 18기 김인섭

CONTENTS

01 쿠버네티스 소개

02 쿠버네티스 장점 및 기능

03 쿠버네티스 실습

- kubectl 명령어 실습
- pod 실행 및 webserver 접속
- replicationcontroller 실습

01

쿠버네티스 소개

쿠버네티스 소개

쿠버네티스 간단 소개

https://youtu.be/8ypDYpKKLxY?t=398

쿠버네티스 소개

kubernetes

컨테이너를 **쉽고 빠르게 배포/확장**하고 **관리를 자동화**해주는 오픈소스 플랫폼

쿠버네티스 소개

용어	뜻
컨테이너	앱이 구동되는 환경까지 감싸서 실행할 수 있도록 하는 격리 기술
컨테이너 런타임	컨테이너를 다루는 도구
도커	컨테이너를 다루는 도구 중 가장 유명한 것
쿠버네티스	컨테이너 런타임을 통해 컨테이너를 오케스트레이션 하는 도구
오케스트레이션	여러 서버에 걸친 컨테이너 및 사용하는 환경 설정을 관리하는 행위

컨테이너 : 우리가 구동하려는 애플리케이션을 실행할 수 있는 환경까지 감싸서, 어디서든 쉽게 실행할 수 있도록 해 주는 기술

컨테이너 런타임 : 컨테이너를 사용할 때 필요한 도구

쿠버네티스 : 컨테이너 런타임을 통해 컨테이너를 다루는 도구

쿠버네티스 소개

1주일에 20억개의 컨테이너를 생성하는 google이
컨테이너 배포 시스템으로 사용하던 **borg를 기반으로 만든 오픈소스**
v1.0 release (2015)

CLOUD NATIVE
COMPUTING FOUNDATION

01 쿠버네티스 소개

쿠버네티스 소개

운영에서 사용가능한 컨테이너 오케스트레이션

01 쿠버네티스 소개

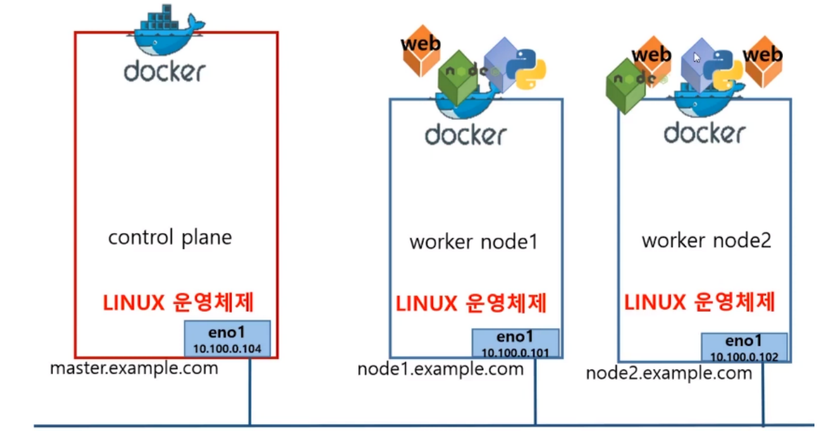
쿠버네티스 소개

컨테이너 오케스트레이션

Container Orchestration

복잡한 컨테이너 환경을 **효과적으로 관리**하기 위한 도구

컨테이너 오케스트레이션



01 쿠버네티스 소개

쿠버네티스 소개

행성 규모 확장성

Google이 일주일에 수십억 개의 컨테이너들을 운영하게 해준 원칙들에 따라 디자인되었기 때문에, 쿠버네티스는 운영팀의 규모를 늘리지 않고도 확장될 수 있습니다.

무한한 유연성

지역적인 테스트든지 글로벌 기업 운영이든지 상관없이, 쿠버네티스의 유연성은 사용자의 복잡한 니즈를 모두 수용하기 때문에 사용자의 애플리케이션들을 끊임없고 쉽게 전달할 수 있습니다.

K8s를 어디서나 실행

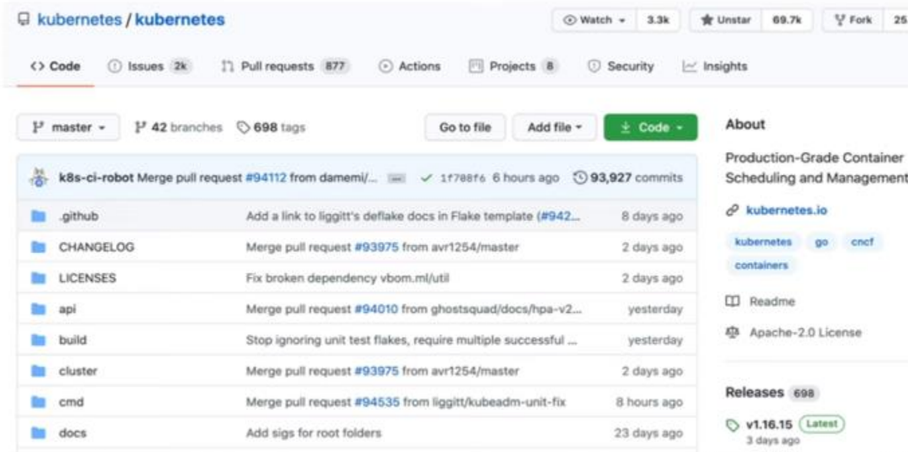
쿠버네티스는 오픈소스로서 온-프레미스, 하이브리드, 또는 퍼블릭 클라우드 인프라스트럭처를 활용하는 데 자유를 제공하며, 워크로드를 사용자에게 관건이 되는 곳으로 손쉽게 이동시켜 줄 수 있습니다.

쿠버네티스 장점

- 오픈소스
- 엄청난 인기
- 무한한 확장성
- 사실상의 표준 (de facto)

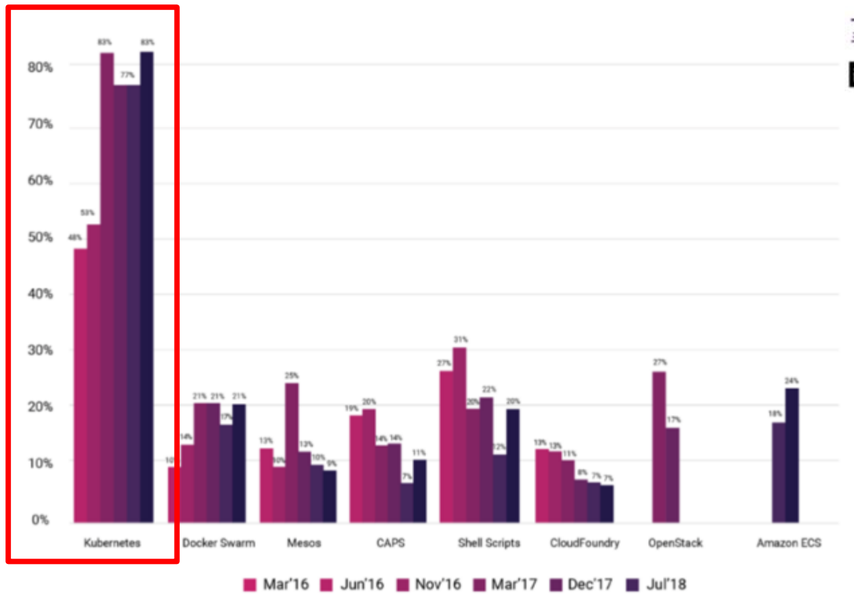
쿠버네티스 장점

오픈소스



쿠버네티스 장점

엄청난 인기



Use of CNCF Projects

쿠버네티스의 인기

if(kakao)dev2019

R1 • 15:00 - 15:45

카카오톡 적용 사례를 통해 살펴보는 카카오 클라우드의 Kubernetes as a Service

카카오 프라이빗 클라우드의 KaaS(Kubernetes as a Service) DKOSv3 를 소개 하고, 카카오톡에 적용된 쿠버네티스 사례와 서비스 운영에 대해 공유합니다.

- 서비스 중단 없는 쿠버네티스 기반 컨테이너 클라우드 운영 (dennis.hong)
* 카카오의 Kubernetes as a Service, DKOSv3 소개
* 사내 인증과 통합된 쿠버네티스 멀티 클러스터 CI/CD 서비스 제공
* 자동화된 쿠버네티스 클러스터의 라이프사이클 관리 기능을 통한 무중단 운영
* Custom Controller, Custom Resource Definition 을 통한 On-premise 환경에서의 쿠버네티스와 물리 인프라 자원 통합

- 서비스 중단 없는 쿠버네티스 기반 카카오톡 백엔드 배포 및 운영 (greg.47)
* 카카오톡 메시징 운영

#클라우드 #쿠버네티스 #k8s #컨테이너 #대규모 #카카오톡 #실서비스 #운영 #사례

쿠버네티스 장점

무한한 확장성

Kubeflow 머신러닝

TEKTON CI/CD

서비스메시

K n 서버리스

무한한 확장성

CIO FROM IDG

가상화 / 개발자 / 데이터센터 / 비즈니스|경제 / 신기술|미래 / 애플리케이션 / 오픈소스 / 운영체제 / 클라우드 2018.11.22

칼럼 | 쏘리! 리눅스, 이제 주인공은 '쿠버네티스'다

Matt Asay | InfoWorld

이제 운영체제는 더 이상 중요하지 않다. 이는 개발자나 클라우드에 있어 리눅스가 더 이상 중요하지 않다는 의미이다.

일어나지 않은 일이 그 증거다. 우분투를 운영하는 캐노니컬은 IBM으로부터 340억 달러의 인수 제안을 받지 않았다. 매각에 관심이 없다는 이야기가 있긴 했지만, 캐노니컬 설립자인 마크 셔틀워스라면 그러한 제안이 왔을 때 받아들였을 것이다.

인수 제안이 캐노니컬에게 가지 않았고, 당분간은 그럴 일이 없다. 이유는 IT산업이 운영체제 자체를 더 이상 가치 있게 생각하지 않기 때문이다. 아니, 차라리 IT산업이 가치 있게 생각하는 새로운 운영체계가 있기 때문이라는 말이 맞겠다. 이는 쿠버네티스라고 불린다.

02 쿠버네티스 장점

쿠버네티스 장점

사실상의 표준

데 팍토

법률 등의 명시적인 방법으로 인정되진 않았지만 다들 암묵적으로 공식처럼 여기는 것

데 팍토(라틴어: De facto)는 **사실상**의 의미로 쓰이는 표현으로, 법적으로 공인된 사항이 아니더라도 실제 존재하는 사례를 가리키는 말이다.)이라 부르기도 한다.

EKS

Amazon
Elastic Kubernetes Service

AKS

Azure
Kubernetes Service

GKE

Google
Kubernetes Engine

쿠버네티스 기본 기능

쿠버네티스는 컨테이너 오케스트레이션으로 기본 기능 충실하게 제공

. 상태관리

-> 상태를 선언하고 선언한 상태를 유지 / 노드가 죽거나 컨테이너 응답이 없을 경우 자동으로 복구

. 스케줄링

-> 클러스터의 여러 노드 중 조건에 맞는 노드를 찾아 컨테이너를 배치

. 클러스터

-> 가상 네트워크를 통해 하나의 서버에 있는 것처럼 통신

. 서비스 디스커버리

-> 서로 다른 서비스를 쉽게 찾고 통신할 수 있음

쿠버네티스 기본 기능

쿠버네티스는 컨테이너 오케스트레이션으로 기본 기능 충실하게 제공

. 리소스 모니터링

-> cAdvisor를 통한 리소스 모니터링

. 스케일링

-> 리소스에 따라 자동으로 서비스를 조정함

. RollOut / RollBack

-> 배포 / 롤백 및 버전 관리

쿠버네티스

컨테이너를 쉽고 빠르게 배포/확장하고
관리를 자동화해주는 오픈소스 플랫폼

이제는 '쿠버네티스'가 서버관리의 대세가 되어가고 있다.

도커가 '하나의 프로그램을 관리하는 방식'
쿠버네티스는 '여러 개의 프로그램을 관리하는 방식'

쿠버네티스는 컨테이너를 다루기 위해 도커 이외에도 다양한 컨테이너 런타임 소프트웨어 사용 가능!

03

실습

실습 환경 구성

실습 환경 구성

play with Kubernetes

Dcoker에서 제공

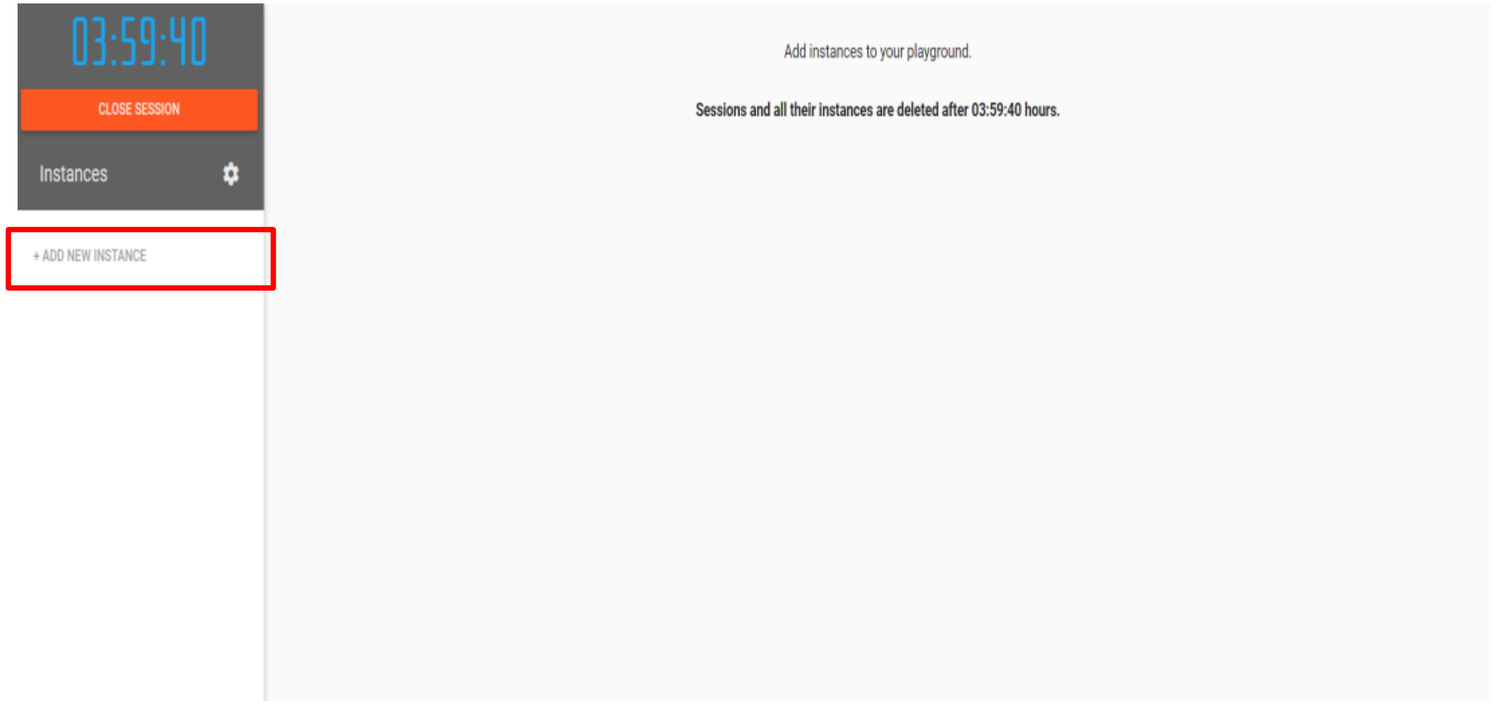
4시간 사용 가능

Master, worker node를 직접 구성한 후 사용 가능

https://labs.play-with-k8s.com

03 실습

실습 환경 구성


03:59:40
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
Add instances to your playground.
Sessions and all their instances are deleted after 03:59:40 hours.

03 실습

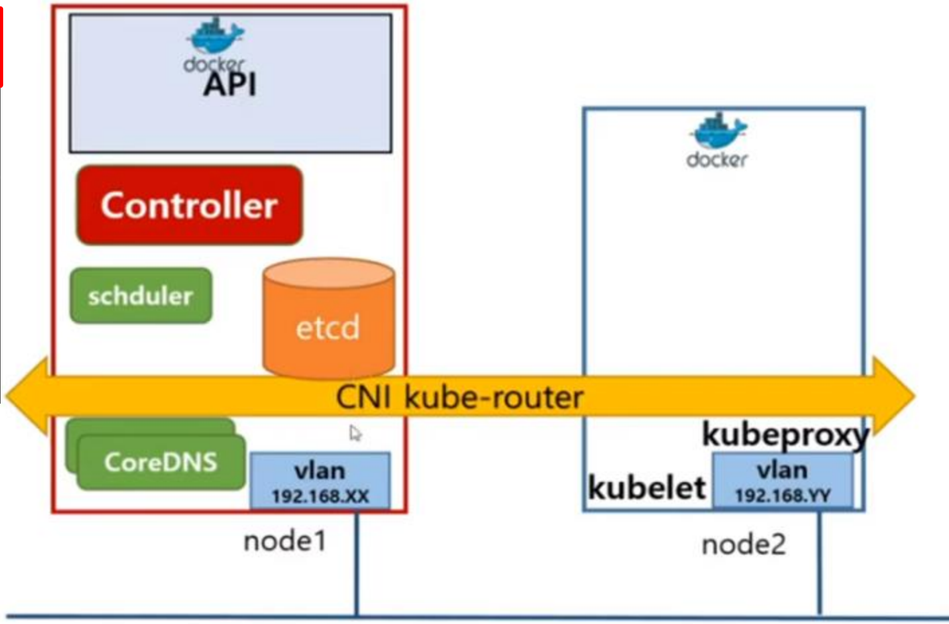
실습 환경 구성

```
                    WARNING!!!!

This is a sandbox environment. Using personal credentials
is HIGHLY! discouraged. Any consequences of doing so, are
completely the user's responsibilites.

You can bootstrap a cluster as follows:

1. Initializes cluster master node:

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

2. Initialize cluster networking:

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

3. (Optional) Create an nginx deployment:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

                    The PWK team.

[node1 ~]$
```

03 실습

실습 환경 구성

```
1. Initializes cluster master node:

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

2. Initialize cluster networking:

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

3. (Optional) Create an nginx deployment:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

                        The PWK team.


[node1 ~]$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
```



03 실습

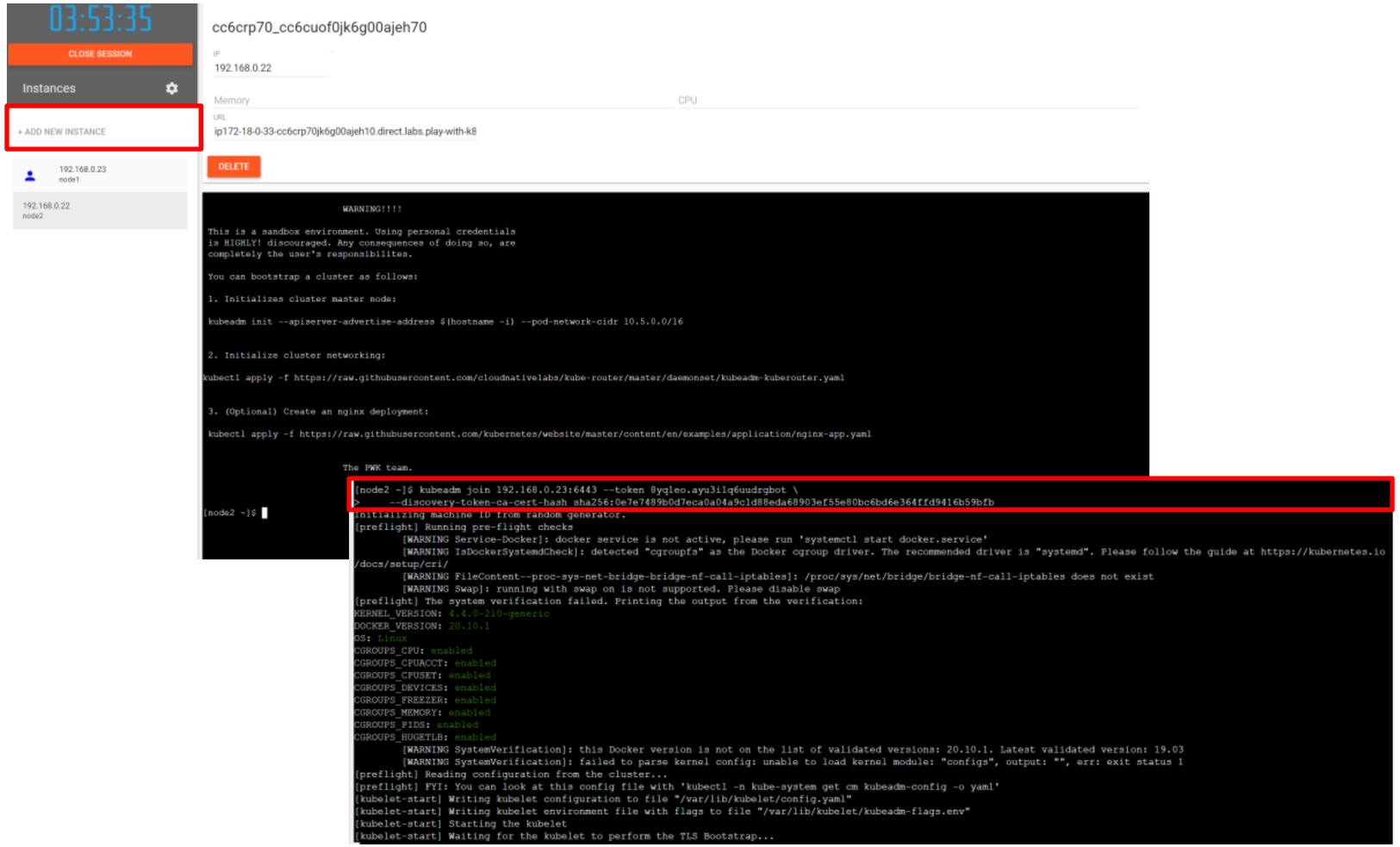
실습 환경 구성

```
 2. Initialize cluster networking:

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm
-kuberouter.yaml
```

```
Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.23:6443 --token 8yqleo.ayu3i1q6uudrgbot \
    --discovery-token-ca-cert-hash sha256:0e7e7489b0d7eca0a04a9c1d88eda68903ef55e80bc6bd6e364ffd9416b59
bfb
Waiting for api server to startup
Warning: resource daemonsets/kube-proxy is missing the kubectl.kubernetes.io/last-applied-configuration
 annotation which is required by kubectl apply. kubectl apply should only be used on resources created
declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be p
atched automatically.
daemonset.apps/kube-proxy configured
No resources found
[node1 ~]$ kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemon
set/kubeadm-kuberouter.yaml
configmap/kube-router-cfg created
daemonset.apps/kube-router created
serviceaccount/kube-router created
clusterrole.rbac.authorization.k8s.io/kube-router created
clusterrolebinding.rbac.authorization.k8s.io/kube-router created
```

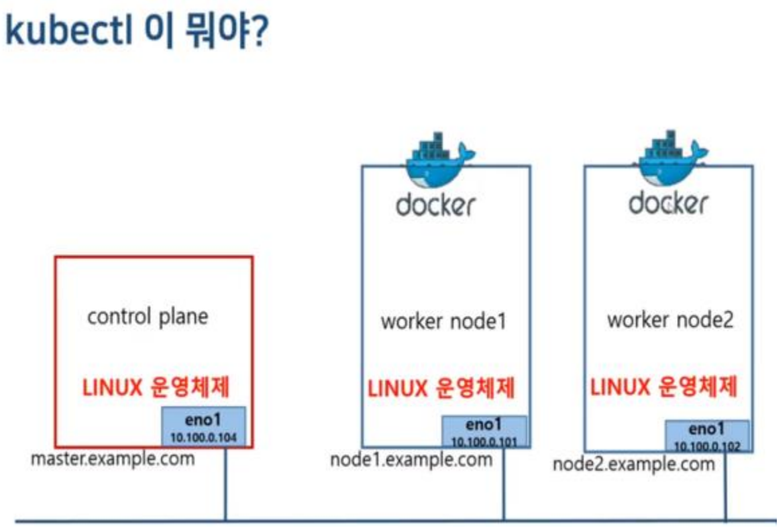
03 실습

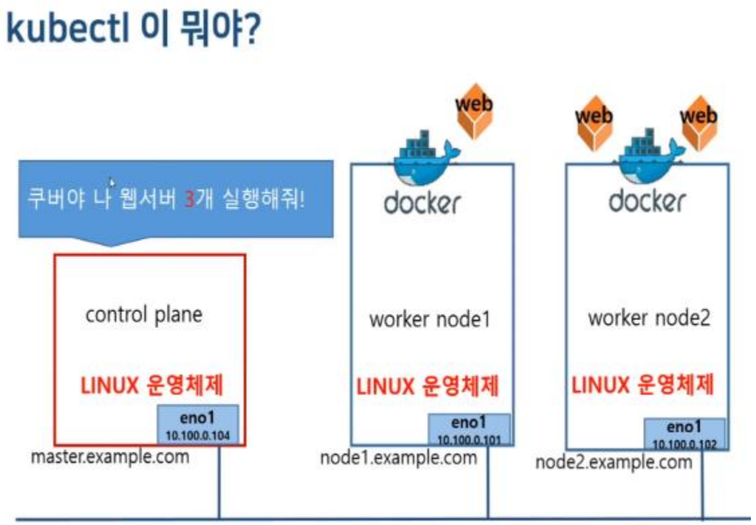
실습 환경 구성


03:53:35
CLOSE SESSION
Instances
+ ADD NEW INSTANCE
192.168.0.23
node1
192.168.0.22
node2
cc6crp70_cc6cuof0jk6g00ajeh70
IP
192.168.0.22
Memory
CPU
URL
ip172-18-0-33-cc6crp70jk6g00ajeh10.direct.labs.play-with-k8
DELETE
WARNING!!!!
This is a sandbox environment. Using personal credentials
is HIGHLY! discouraged. Any consequences of doing so, are
completely the user's responsibilites.
You can bootstrap a cluster as follows:
1. Initializes cluster master node:
kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
2. Initialize cluster networking:
kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
3. (Optional) Create an nginx deployment:
kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml
The PWK team.
[node2 ~]$
[node2 ~]$ kubeadm join 192.168.0.23:6443 --token 8yqleo.ayu3ilq6uudrgbot \
> --discovery-token-ca-cert-hash sha256:0e7e7489b0d7eca0a04a9c1d88eda68903ef55e80bc6bd6e364ffd9416b59bfb
Initializing machine ID from random generator.
[preflight] Running pre-flight checks
[WARNING Service-Docker]: docker service is not active, please run 'systemctl start docker.service'
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[WARNING Swap]: running with swap on is not supported. Please disable swap
[preflight] The system verification failed. Printing the output from the verification:
KERNEL_VERSION: 4.4.0-210-generic
DOCKER_VERSION: 20.10.1
OS: Linux
CGROUPS_CPU: enabled
CGROUPS_CPUACCT: enabled
CGROUPS_CPUSET: enabled
CGROUPS_DEVICES: enabled
CGROUPS_FREEZER: enabled
CGROUPS_MEMORY: enabled
CGROUPS_PIDS: enabled
CGROUPS_HUGETLB: enabled
[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.1. Latest validated version: 19.03
[WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

03 실습

kubectl 명령어 사용하기





03 실습

kubectl 명령어 사용하기

kubectl 명령어 기본구조

자원의 타입
(node,pod,service)

kubectl [command] [TYPE] [NAME] [flags]

자원에 실행할 명령
(Create,get,delete,eidt..)

03 실습

kubectl 명령어 사용하기

kbectl api-resources : 약어 정보 확인

kubectl --help : kubectl 뒤에 쓸 수 있는 명령어 확인

kubectl logs --help : logs 명령어의 도움을 받고 싶을 때

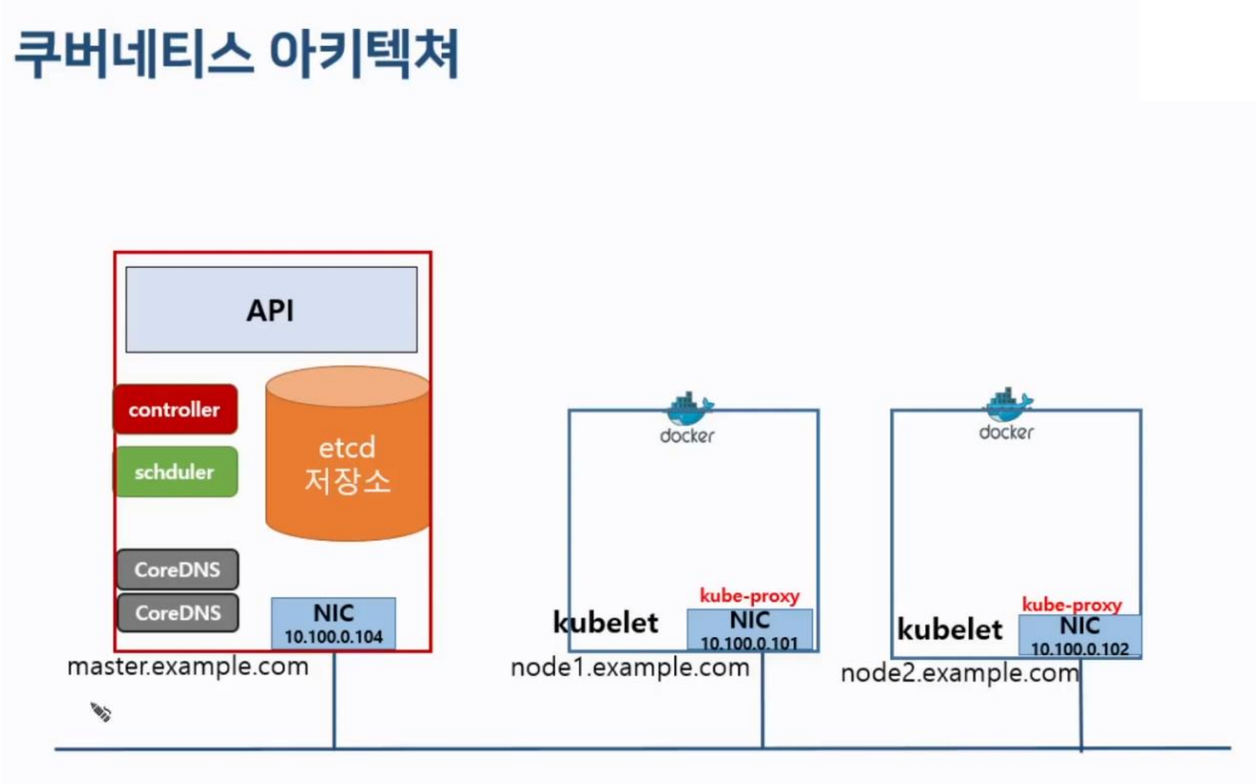
03 실습

kubectl 명령어 사용하기

Q. 작업 노드의 IP 주소 확인해보기

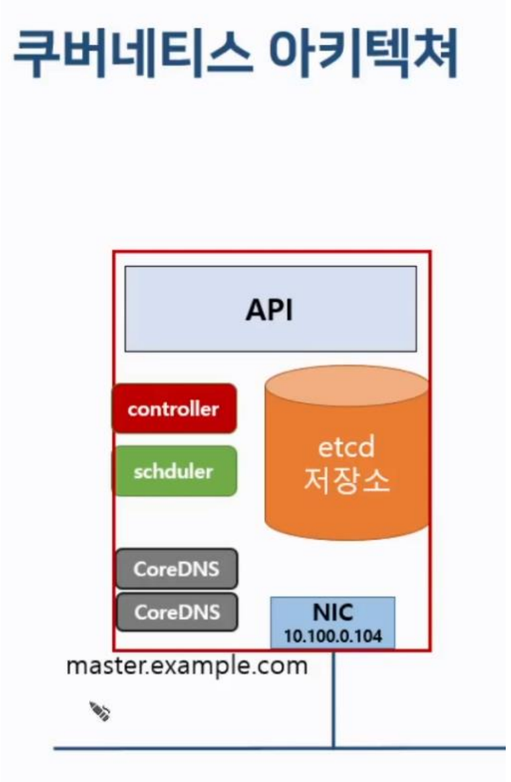
03 실습

쿠버네티스 아키텍쳐



03 실습

쿠버네티스 아키텍쳐



Master 상세 - API server

상태를 바꾸거나 조회

etcd와 유일하게 통신하는 모듈

REST API 형태로 제공

권한을 체크하여 적절한 권한이 없을 경우 요청을 차단

관리자 요청 뿐 아니라 다양한 내부 모듈과 통신

수평으로 확장되도록 디자인

Master 상세 - etcd

모든 상태와 데이터를 저장

분산 시스템으로 구성하여 안전성을 높임 (고가용성)

가볍고 빠르면서 정확하게 설계 (일관성)

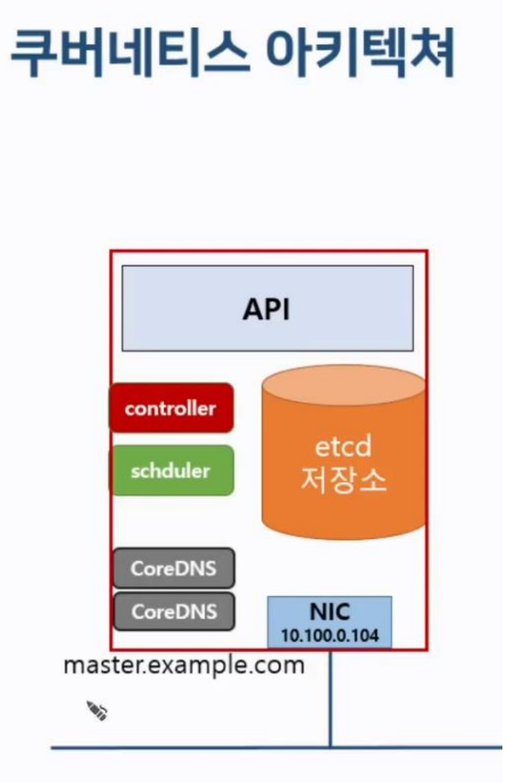
Key(directory)-Value 형태로 데이터 저장

TTL(time to live), watch같은 부가 기능 제공

백업은 필수!

03 실습

쿠버네티스 아키텍쳐



Master 상세 - Scheduler

새로 생성된 Pod을 감지하고 실행할 노드를 선택

노드의 현재 상태와 Pod의 요구사항을 체크

- 노드에 라벨을 부여
- ex) a-zone, b-zone 또는 gpu-enabled, ...

Master 상세 - Controller

논리적으로 다양한 컨트롤러가 존재

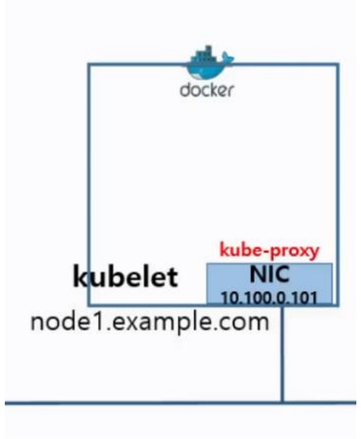
- 복제 컨트롤러
- 노드 컨트롤러
- 엔드포인트 컨트롤러...

끊임 없이 상태를 체크하고 원하는 상태를 유지

복잡성을 낮추기 위해 하나의 프로세스로 실행

03 실습

쿠버네티스 아키텍쳐



Node 상세 - kubelet

각 노드에서 실행

Pod을 실행/중지하고 상태를 체크

CRI (Container Runtime Interface)

- docker
- Containerd
- CRI-O
- ...

Node 상세 - proxy

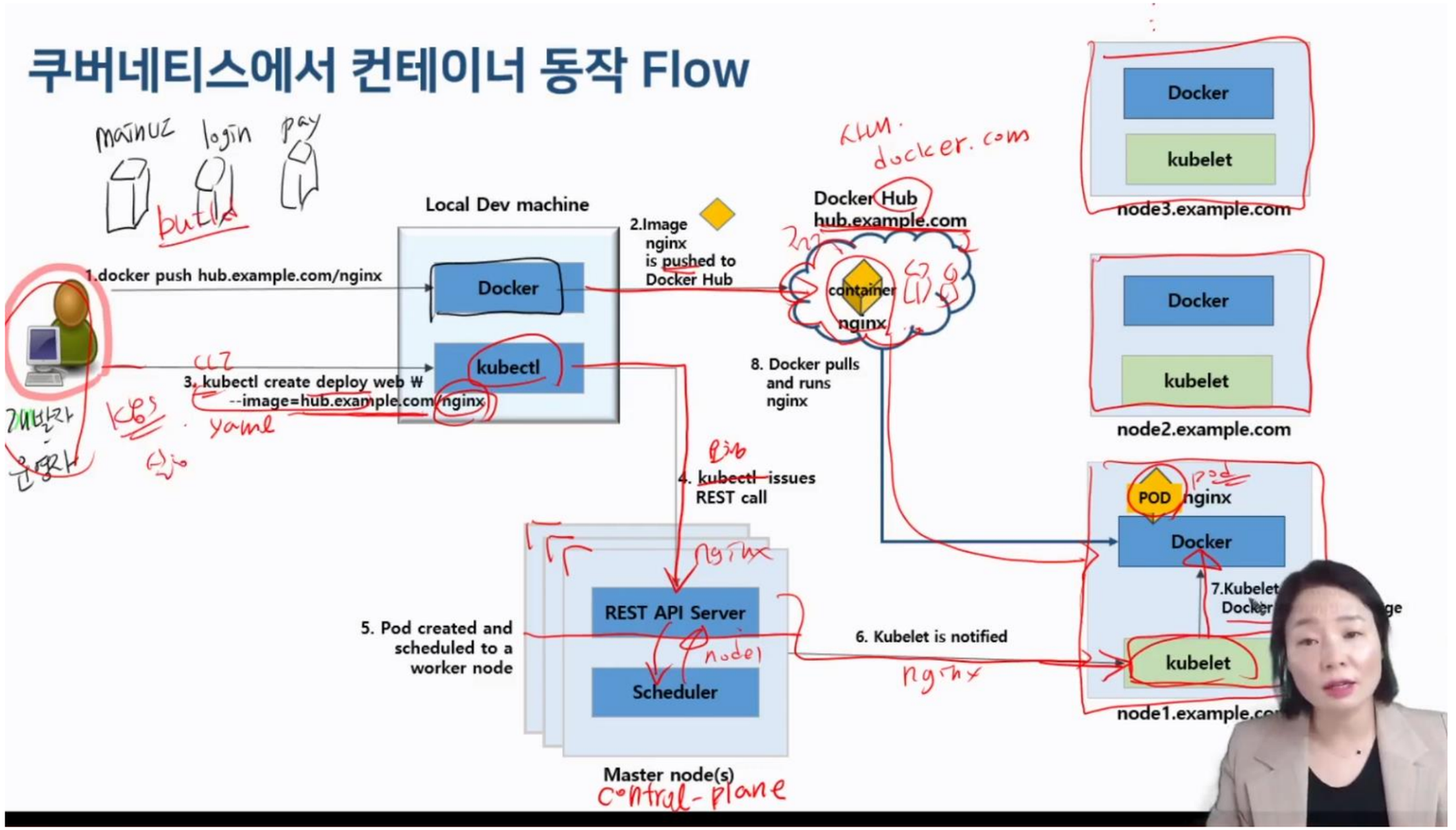
네트워크 프록시와 부하 분산 역할

성능상의 이유로 별도의 프록시 프로그램 대신

iptables 또는 IPVS를 사용 (설정만 관리)

03 실습

쿠버네티스 동작 Flow



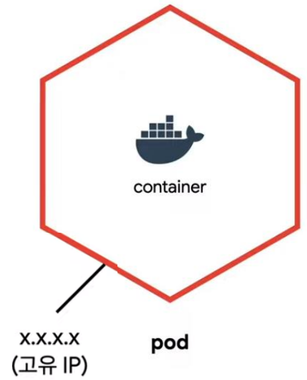
https://www.youtube.com/watch?v=Iue9TC13vPQ&list=PLApuRlvrZKohaBHvXAOhUD-RxD0uQ3z0c&index=7

03 실습

Pod실행

Pod

가장 작은 배포 단위

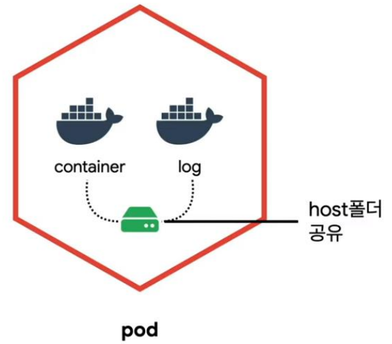


Pod

전체 클러스터에서 고유한 IP를 할당

Pod

여러개의 컨테이너가 하나의 Pod에 속할 수 있음

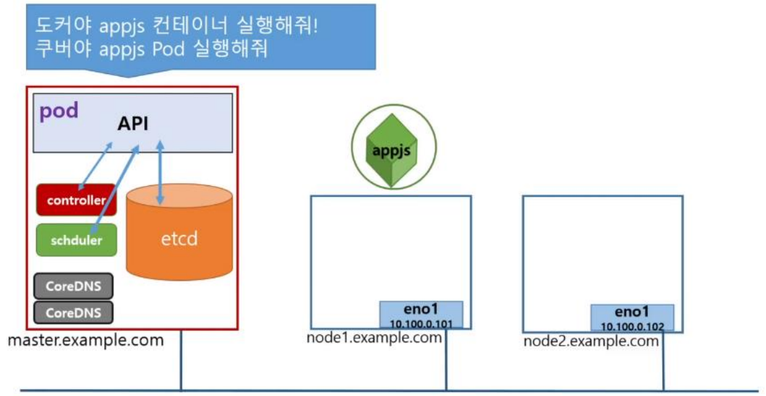


03 실습

Pod실행

Pod 란?

- 컨테이너를 표현하는 K8S API의 최소 단위



간단한 Pod 실행(웹서버 실행해보기)

```
[node1 ~]$ kubectl run webserver --image=nginx:1.14 --port 80
pod/webserver created
```

컨테이너 Pod 3개 실행해줘~

```
[node1 ~]$ kubectl create deployment mainui --image=httpd --replicas=3
deployment.apps/mainui created
```

03 실습

Webserver 이름 바꿔보기

컨테이너 안으로 들어가서 webserver 이름 바꿔보기!

```
[node1 ~]$ kubectl exec webserver -it -- /bin/bash
root@webserver:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

```
root@webserver:/usr/share/nginx/html# ls
50x.html  index.html
root@webserver:/usr/share/nginx/html# cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

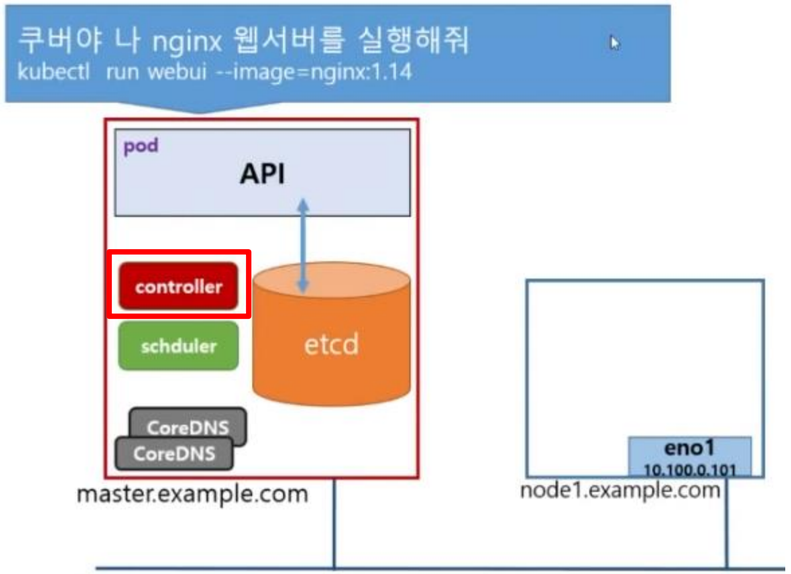
```
root@webserver:/usr/share/nginx/html# echo BOAZ ZZANG > index.html
root@webserver:/usr/share/nginx/html# ls
50x.html  index.html
root@webserver:/usr/share/nginx/html# cat index.html
BOAZ ZZANG
```

```
[node1 ~]$ curl 10.5.1.2
BOAZ ZZANG
```

03 실습

Controller 실습

- 컨테이너를 실행하는 K8S의 최소 API Pod



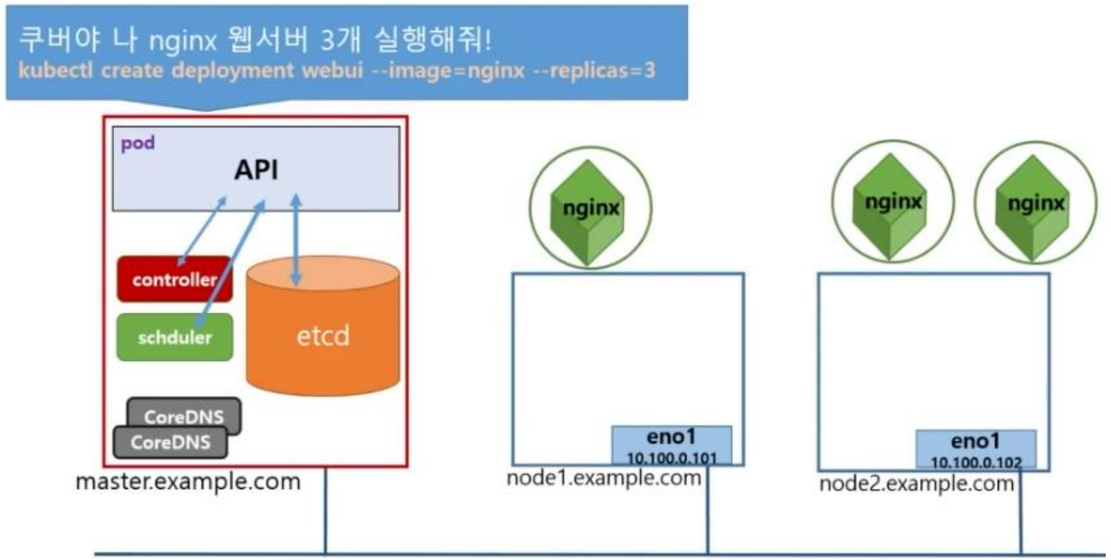
- ReplicationController
- ReplicaSet
- Deployment
- DaemonSet
- StatefulSet
- Job
- CronJob

03 실습

Controller 실습

Controller 란

- Pod의 개수를 보장



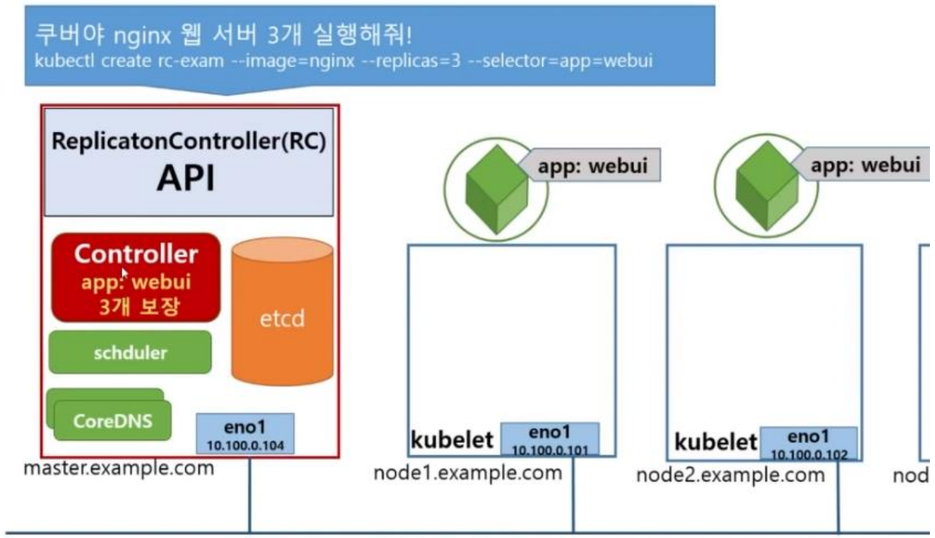
03 실습

Controller 실습

ReplicationController

- 요구하는 Pod의 개수를 보장하며 파드 집합의 실행을 항상 안정적으로 유지하는 것을 목표
  - 요구하는 Pod의 개수가 부족하면 template를 이용해 Pod를 추가
  - 요구하는 Pod 수 보다 많으면 최근에 생성된 Pod를 삭제

ReplicationController 동작원리




```
apiVersion: v1
kind: ReplicationController
metadata:
  name: <RC_이름>
spec:
  replicas: <배포갯수>
  selector:
    key: value
  template:
    <컨테이너 템플릿>
      ....
```

03 실습

Controller 실습

Yaml 파일 불러오기!

```
[node1 ~]$ cat > rc-nginx.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc-nginx
spec:
  replicas: 3
  selector:
    app: webui
    version: "1.14"
  template:
    metadata:
      name: nginx-pod
      labels:
        app: webui
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.14
```

```
[node1 ~]$ cat rc-nginx.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc-nginx
spec:
  replicas: 3
  selector:
    app: webui
    version: "1.14"
  template:
    metadata:
      name: nginx-pod
      labels:
        app: webui
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.14
```

03 실습

Controller 실습

결과 확인

```
[node1 ~]$ kubectl create -f rc-nginx.yaml
replicationcontroller/rc-nginx created
```

```
[node1 ~]$ kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
rc-nginx-pcq6l   1/1     Running   0          76s   10.5.1.3    node2   <none>           <none>
rc-nginx-twjkk   1/1     Running   0          76s   10.5.1.4    node2   <none>           <none>
rc-nginx-vhkr2   1/1     Running   0          76s   10.5.1.2    node2   <none>           <none>
```

```
[node1 ~]$ kubectl get replicationcontrollers
NAME       DESIRED   CURRENT   READY   AGE
rc-nginx   3         3         3       2m3s
```

03 실습

Controller 실습

결과 확인

```
[node1 ~]$ kubectl describe rc rc-nginx
Name:         rc-nginx
Namespace:    default
Selector:     app=webui
Labels:       app=webui
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=webui
  Containers:
   nginx-container:
    Image:        nginx:1.14
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                    Message
  ----    ------            ----  ----                    -------
  Normal  SuccessfulCreate  3m    replication-controller  Created pod: rc-nginx-vhkr2
  Normal  SuccessfulCreate  3m    replication-controller  Created pod: rc-nginx-twjkk
  Normal  SuccessfulCreate  3m    replication-controller  Created pod: rc-nginx-pcq6l
```

03 실습

Controller 실습

결과 확인

```
[node1 ~]$ kubectl scale rc rc-nginx --replicas=2
replicationcontroller/rc-nginx scaled
[node1 ~]$ kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
rc-nginx-v8vmf   1/1     Running   0          6s    10.5.1.5    node2   <none>           <none>
rc-nginx-vhkr2   1/1     Running   0          23m   10.5.1.2    node2   <none>           <none>
```

```
[node1 ~]$ kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
rc-nginx-7rrt4   1/1     Running   0          34s   10.5.1.4    node2   <none>           <none>
rc-nginx-b222q   1/1     Running   0          34s   10.5.1.3    node2   <none>           <none>
rc-nginx-gvbv4   1/1     Running   0          34s   10.5.1.2    node2   <none>           <none>
[node1 ~]$ kubectl delete pod rc-nginx-7rrt4
pod "rc-nginx-7rrt4" deleted
^C
[node1 ~]$ kubectl get pods -o wide
NAME             READY   STATUS        RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS G
ATES
rc-nginx-7rrt4   0/1     Terminating   0          67s   10.5.1.4    node2   <none>           <none>
rc-nginx-b222q   1/1     Running       0          67s   10.5.1.3    node2   <none>           <none>
rc-nginx-gvbv4   1/1     Running       0          67s   10.5.1.2    node2   <none>           <none>
rc-nginx-sxg7x   1/1     Running       0          16s   10.5.1.5    node2   <none>           <none>
```

03 실습

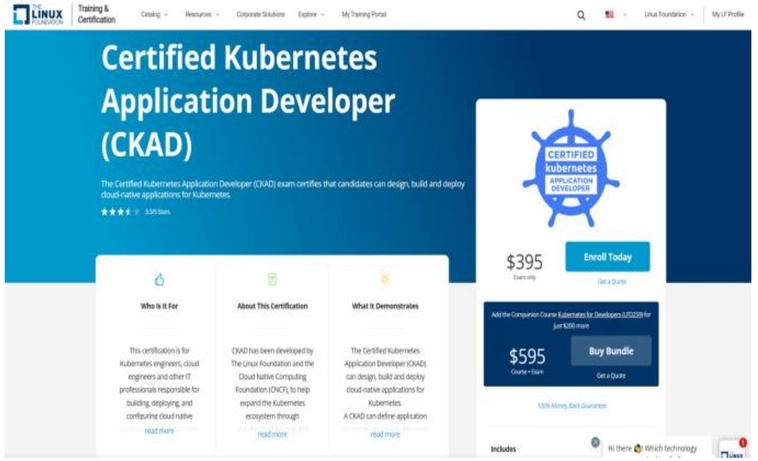
Controller 실습

QUESTION & ANSWER

1. 다음의 조건으로 ReplicationController를 사용하는 rc-lab.yaml 파일을 생성하고 동작시킵니다.
 - labels(name: apache, app:main, rel:stable)를 가지는 httpd:2.2 버전의 Pod를 2개 운영합니다.
  - rc name : rc-mainui
  - container : httpd:2.2
 - 현재 디렉토리에 rc-lab.yaml 파일이 생성되어야 하고, 애플리케이션 동작은 파일을 이용해 실행합니다.

04 정리

CKAD



Domains & Competencies Collapse All

Application Design and Build 20%

- Define, build and modify container images
- Understand Jobs and CronJobs
- Understand multi-container Pod design patterns (e.g. sidecar, init and others)
- Utilize persistent and ephemeral volumes

Application Deployment 20%

- Use Kubernetes primitives to implement common deployment strategies (e.g. blue/green or canary)
- Understand Deployments and how to perform rolling updates
- Use the Helm package manager to deploy existing packages

Application Observability and Maintenance 15%

- Understand API deprecations
- Implement probes and health checks
- Use provided tools to monitor Kubernetes applications
- Utilize container logs
- Debugging in Kubernetes

Application Environment, Configuration and Security 25%

- Discover and use resources that extend Kubernetes (CRD)
- Understand authentication, authorization and admission control
- Understanding and defining resource requirements, limits and quotas
- Understand ConfigMaps
- Create & consume Secrets
- Understand ServiceAccounts
- Understand SecurityContexts

감사합니다