BESPIN GLOBAL

Kubernetes 핵심운영

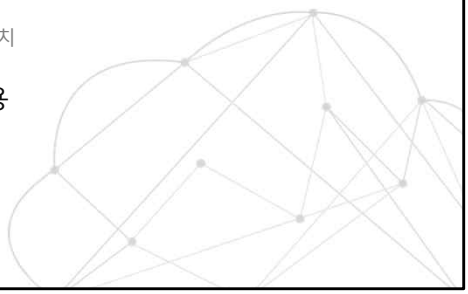
베스핀아카데미 이성미(seongmi.lee@bespinglobal.com)

001. 컨테이너 오케스트레이션
- 컨테이너 오케스트레이션
- Kubernetes 소개

002. Kubernetes Architecture
- Kubernetes Components
- Kubernetes 핵심 운영

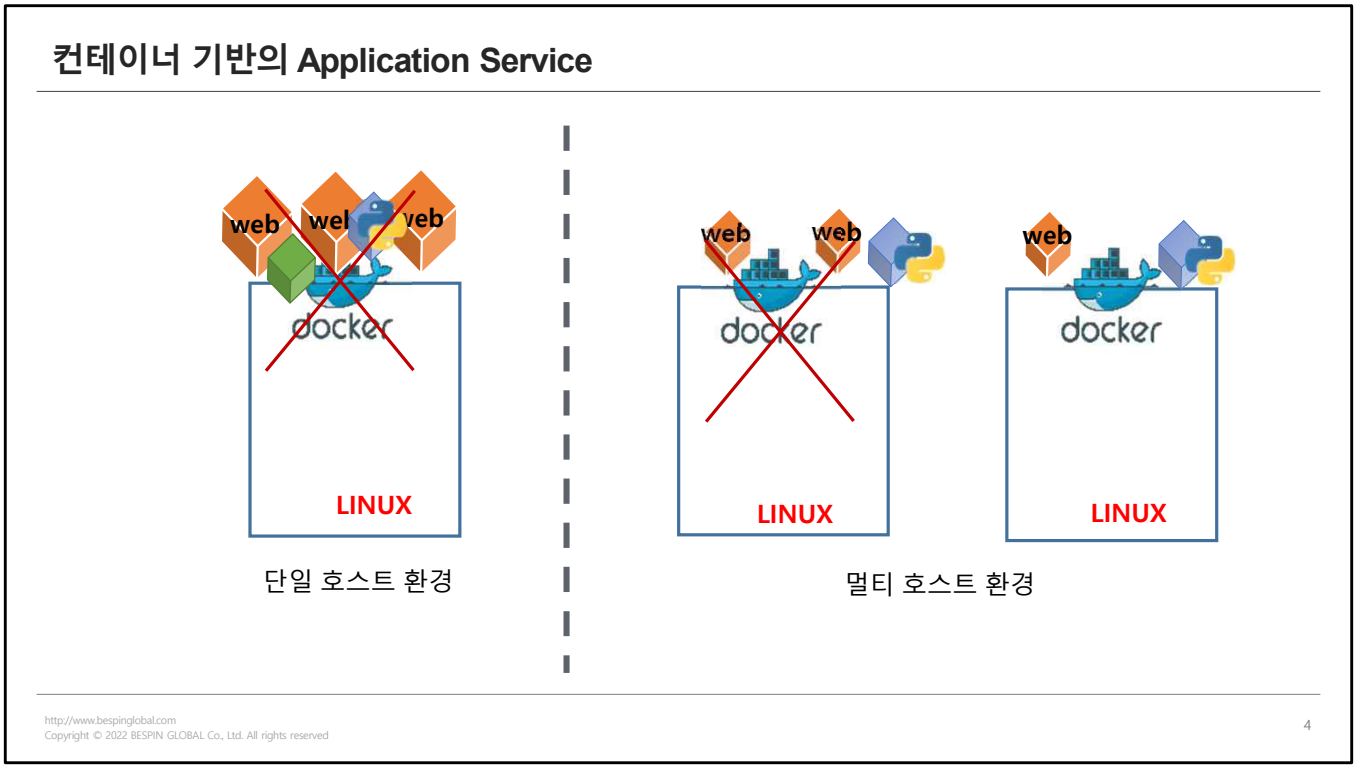
003. Kubernetes 설치
- Vm에 Kubernetes 설치
- 클라우드 인스턴스에 Kubernetes 설치

004. 설치 후 기본 명령어 사용
- kubectl 명령어 사용



[Kubernetes 핵심 운영]

컨테이너 오케스트레이션



멀티 호스트 환경과 클러스터링

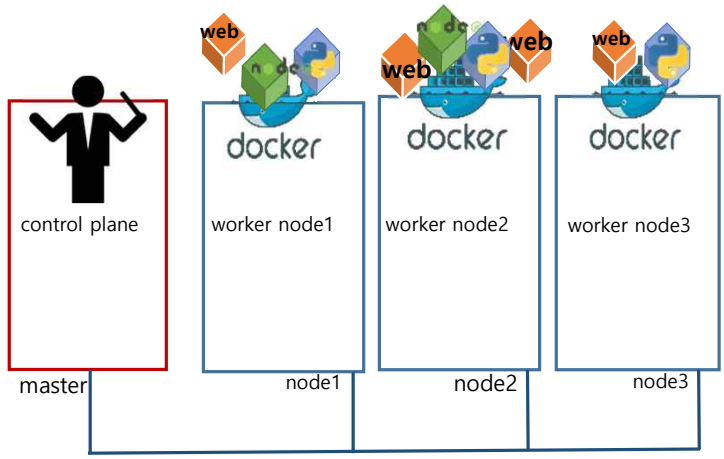
Docker에서는 단일 호스트가 아닌 멀티 호스트 환경에서 도커를 구동시켜 높은 가용성 및 확장성을 가진 애플리케이션을 구축할 수 있다.

가용성(Availability)
- 가용성이란 시스템이 연속적으로 가동할 수 있는 능력을 의미한다.
- 서버 에러나 하드웨어에 장애가 발생하여도 다른 정상 서버나 하드웨어로 전환되어 기존 처리가 이어질수 있기 때문에 높은 신뢰성을 제공할 수 있다.

확장성(Scalability)
- 높은 부하로 시스템이 다운되는 것을 피하기 위해 여러 컴퓨터를 '클러스터링'으로 구축·분산하여 신뢰성을 높일 수 있다.
- 클라우드 가상 서버는 오토 스케일 기능을 제공하는 경우도 있다.

클러스터링(Clustering)
- 클러스터링이란 여러 대의 서버와 하드웨어를 1대 처럼 묶어주는 기술이다.
- 이는 예기치 않은 상황에서 시스템 일부분에 장애가 발생하더라도 시스템 전체에 영향을 미치지 않도록 한다.
- 이러한 기능은 가용성과 확장성을 향상시킬수 있다.

컨테이너 오케스트레이션



컨테이너 오케스트레이션(Container Orchestration)

오케스트레이션(orchestration)은 관현악 연주를 위하여 작곡을 하는 기법을 뜻한다.
오케스트레이션은 1~3개의 피아노 악보가 될 수있는 낮은 악보를 가지고 오케스트라가 궁극적으로 연주 할 음악의 다른 부분을 연주하는 악기를 기록하는 과정이다.
오케스트라의 경우 현악기, 목관악기, 금관악기, 타악기, 피아노 및 하프 등이 있고, 작곡가의 지시와 악보를 통해 음악의 완성도가 달라진다.

컨테이너 오케스트레이션은 컨테이너의 배포, 관리, 확장, 네트워킹을 자동화한다. 수백 또는 수천 개의 컨테이너와 호스트를 배포하고 관리해야 하는 기업에서는 컨테이너 오케스트레이션을 활용 할 수 있다.

컨테이너 오케스트레이션은 컨테이너를 사용하는 어떤 환경에서든 사용할 수 있다. 또한 재설계할 필요 없이 각기 다른 환경 전반에 동일한 애플리케이션을 배포하는 데에도 도움이 된다. 컨테이너에 마이크로서비스를 구현하면 스토리지, 네트워킹, 보안과 같은 서비스를 간편하게 오케스트레이션할 수 있다.

컨테이너는 마이크로서비스 기반 애플리케이션에 이상적인 애플리케이션 배포 유닛 및 독립적인 실행 환경을 제공한다.
동일한 하드웨어의 마이크로서비스에서 애플리케이션의 여러 부분들을 독립적으로 실행시키고 개별 요소 및 라이프사이클을 더욱 효과적으로 제어할 수 있다.

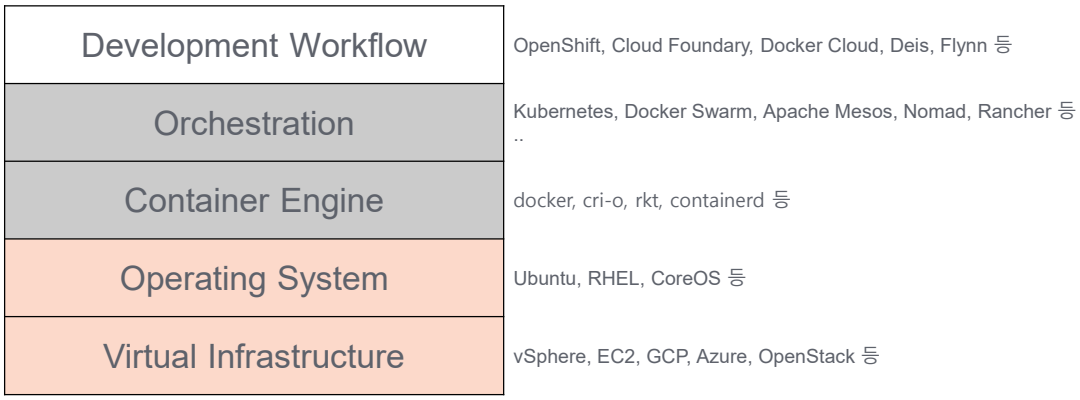
오케스트레이션을 통해 컨테이너 라이프사이클을 관리하면 CI/CD 워크플로우에 이를 통합하는 DevOps 팀을 지원할 수도 있다. 컨테이너화된 마이크로서비스는 애플리케이션 프로그래밍 인터페이스(API) 및 DevOps 팀과 함께 클라우드 네이티브 애플리케이션의 기반을 이룬다.

컨테이너 오케스트레이션을 사용해 다음과 같은 태스크를 자동화하고 관리할 수 있다.

- 프로비저닝 및 배포
- 설정 및 스케줄링
- 리소스 할당
- 컨테이너 가용성
- 인프라 전반의 워크로드 밸런싱을 기반으로 컨테이너 스케일링 또는 제거
- 로드 밸런싱 및 트래픽 라우팅
- 컨테이너 상태 모니터링
- 실행 될 컨테이너를 기반으로 애플리케이션 설정
- 컨테이너 간 상호 작용의 보안 유지

원본: https://www.redhat.com/ko/topics/containers/what-is-container-orchestration

컨테이너 계층구조



컨테이너 계층 구조

Development Workflow

- OpenShift : OpenShift는 기업에 Docker와 Kubernetes를 제공하는 컨테이너 애플리케이션 플랫폼이다.

Orchestration

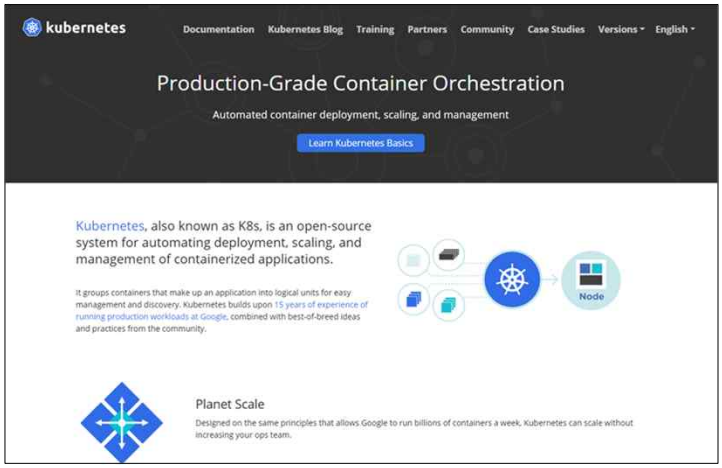
- Docker Swarm : Docker Swarm은 호스트머신, 클라우드, 가상 환경 등에 커맨드로 Docker 실행 환경을 작성 할 수 있는 툴로, docker 클러스터링을 지원한다. 여러 Docker 컨테이너를 클라이언트 1대에서 docker 커맨드로 한 번에 관리할 수 있다는 특징이 있다.
- Apache Mesos : Apache Software Foundation이 오픈 소스로 만든 리소스 관리 툴. 여러 머신을 하나의 클러스터로 관리하고 남은 머신을 탐색하여 리소스를 할당하는 방식으로 효율적으로 태스크를 할당할 수 있는 툴이다.

Container Engine

- Kubernetes는 컨터이너 런타임(Container Runtime)을 플러그처럼(Pluggable) 교체 가능한 방법을 제공하기 위해 CRI (컨테이너 런타임 인터페이스: Container Runtime interface) 라고 부르는 API 를 만들었다.
- Kubelet은 CRI 인터페이스를 사용하여 컨테이너를 조작할 수 있다. CRI 표준을 따르는 어떠한 컨테이너 런타임이라도 Kubernetes와 통합하여 사용할 수 있다.
- 이러한 CRI 표준의 등장과 함께 Kubernetes 에서 사용할 수 있는 런타임은 docker, cri-o, rkt, containerd 등이 있다.

Kubernetes 소개

- k8s = kubernetes
- Container Cluster Manager
- 컨테이너 오케스트레이터(실행 및 관리)
- 다양한 클라우드 및 베어메탈 환경지원
- Google Borg에서 시작되어 오픈소스화 됨
- 2015년 7월 CNCF에 가입
- go 언어로 작성



https://kubernetes.io/

쿠버네티스란?

Kubernetes는 google이 2014년 6월에 배포한 컨테이너 클러스터 매니저로 오픈소스 프로젝트이다. Google은 자신의 인프라 및 기술 이점을 커뮤니티와 공유하기 위한 노력의 일환으로 이 프로젝트를 배포했다.
Google은 인프라에 매주 20억 개의 컨테이너를 런칭하며 10년 넘게 컨테이너 기술을 사용해왔다. 원래 끊임없이 확장되는 DataCenter 영역 전체에 걸쳐 방대한 양의 워크로드를 스케줄링하기 위해 지금은 오메가라 부르는 과거의 보그(borg) 시스템을 만드는 중이었다. 그들은 수년 동안 많은 지식을 축적했고 다양한 곳에서 널리 사용할 수 있도록 기존의 데이터 센터 관리 도구를 다시 작성했다. 그 결과가 쿠버네티스 오픈소스 프로젝트인 것이다.
쿠버네티스는 2014년 첫 공개 이후 레드햇, VMWare, 캐노니컬을 비롯해 오픈소스 커뮤니티 전체의 기여로 급속한 발전을 이루었다. 쿠버네티스의 1.0 버전이 2015년 7월에 출시되었다.

쿠버네티스의 장점 및 주요 기능
사용자의 환경에서 쿠버네티스를 사용할 경우 얻을 수 있는 주요 이점은, 특히 클라우드를 위한 애플리케이션 개발을 최적화하는 중인 경우, 쿠버네티스를 통해 물리 또는 가상 머신의 클러스터에서 컨테이너를 예약하고 실행할 수 있는 플랫폼이 확보된다는 것이다. 더 넓게 보면, 프로덕션 환경에 컨테이너 기반 인프라를 완전히 구현해서 사용할 수 있다. 또한 쿠버네티스는 운영 작업 자동화와 관련이 있으므로 다른 애플리케이션 플랫폼 또는 관리 시스템에서 가능한 작업의 상당수를 컨테이너를 사용해 수행할 수 있다.

Public Cloud Provider 기반의 Kubernetes

- Amazon Elastic Container Service for Kubernetes, Amazon (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)

Public Cloud Provider 기반의 Kubernetes

Amazon EKS

- Amazon Elastic Kubernetes Service(Amazon EKS)는 AWS 클라우드 또는 온프레미스에서 Kubernetes 애플리케이션을 시작, 실행 및 조정할 수 있는 유연성을 제공한다. Amazon EKS는 고가용성의 안전한 클러스터를 제공하는 데 도움이 되며 패치 적용, 노드 프로비저닝 및 업데이트와 같은 주요 태스크를 자동화한다.
- https://aws.amazon.com/ko/eks/

Azure Kubernetes Service (AKS)

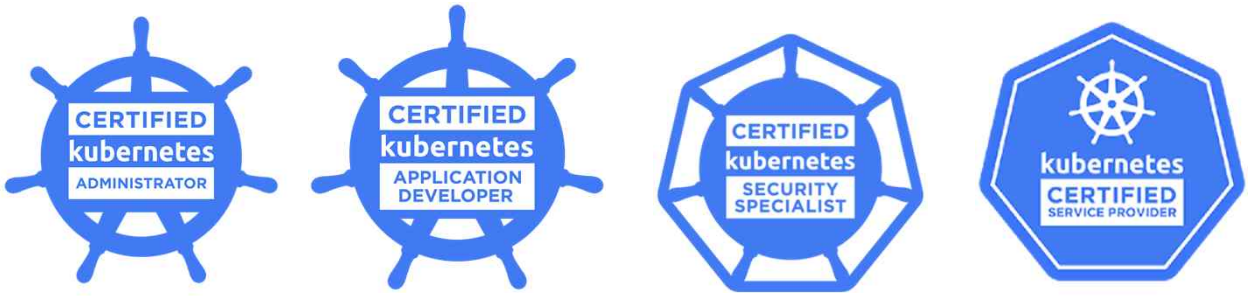
- AKS(Azure Kubernetes Service)는 서버리스 Kubernetes, 통합 CI/CD(지속적인 통합 및 지속적인 업데이트) 환경, 엔터프라이즈급 보안과 거버넌스를 제공한다.
- https://azure.microsoft.com/ko-kr/services/kubernetes-service/

Google Kubernetes Engine (GKE)

- 간단하게 Kubernetes를 자동으로 배포, 확장, 관리할 수 있다.
- K8s 엔지니어링에 가장 크게 기여한 기업에서 빌드한 플랫폼에서 Kubernetes 실행
- 클릭 한 번으로 작동하는 클러스터로 빠르게 시작하고 노드를 15,000개까지 확장
- 멀티 영역과 리전 클러스터를 비롯한 고가용성 제어 영역 활용
- 업계 최초의 4방향 자동 확장을 통해 운영 오버헤드 제거
- 컨테이너 이미지의 취약점 스캔, 데이터 암호화 등의 기본 보안
- https://cloud.google.com/kubernetes-engine

Kubernetes 자격증

- https://cncf.io/



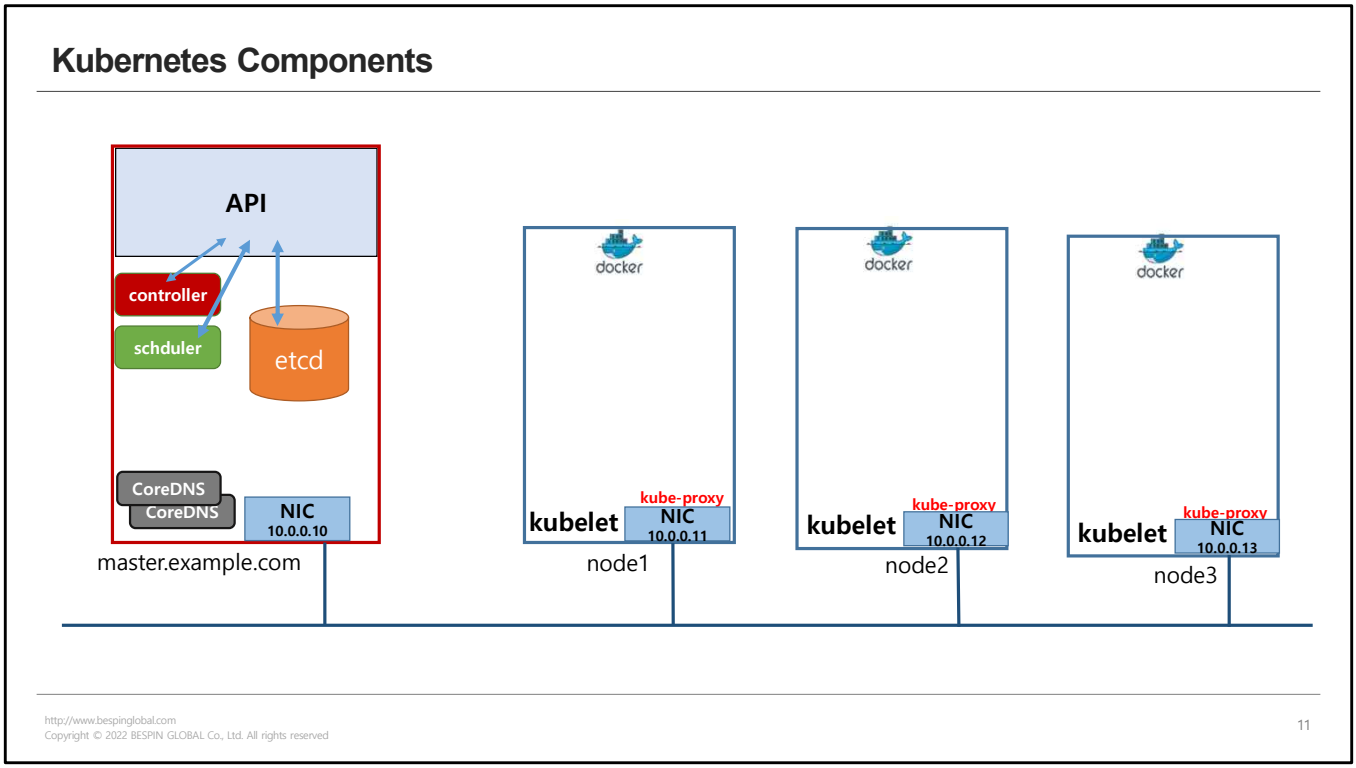
쿠버네티스 자격증 소개

- **CKA(Certified Kubernetes Administrator)**
  - Kubernetes 관리자의 책임을 수행할 수 있는 기술, 지식 및 역량을 갖추고 있음을 보증
  - KCSP가 되려면 최소 3명의 CKA를 고용
  - 응시자는 작업을 완료하는 데 2시간
  - 비용은 $375이며 1회 무료 재응시가 포함

- **CKAD(Certified Kubernetes Application Developer)**
  - kubernetes 애플리케이션 개발자의 책임을 수행할 수 있는 기술, 지식 및 역량을 갖추고 있음을 보증
  - 애플리케이션 리소스를 정의하고 핵심 프리미티브를 사용하여 Kubernetes에서 확장 가능한 애플리케이션 및 도구를 구축, 모니터링 및 문제를 해결
  - 응시자는 작업을 완료하는 데 2시간
  - 비용은 $375이며 1회 무료 재 응시가 포함

- **Certified Kubernetes Security Specialist (CKS)**
  - CKS가 빌드, 배포 및 런타임 중에 컨테이너 기반 애플리케이션 및 Kubernetes 플랫폼을 보호하기 위한 광범위한 모범 사례에 대한 기술, 지식 및 역량을 보유하고 있음을 보
  - CKS 응시자는 CKS에 응시 하기 전에 충분한 Kubernetes 전문 지식을 보유하고 있음을 입증하기 위해 현재 CKA(공인 Kubernetes 관리자) 인증을 보유해야 한다.
  - CKS는 구매할 수 있지만 CKA 인증을 획득할 때까지 예약되지 않습니다.

Kubernetes Architecture



Kubernetes Components

Control plane components
Control plane 컴포넌트는 일반적으로 하나의 노드에서 동작하지만, 고가용성 환경이나 대규모 클러스터에서는 여러 노드에 분산된 형태도 가능하다.

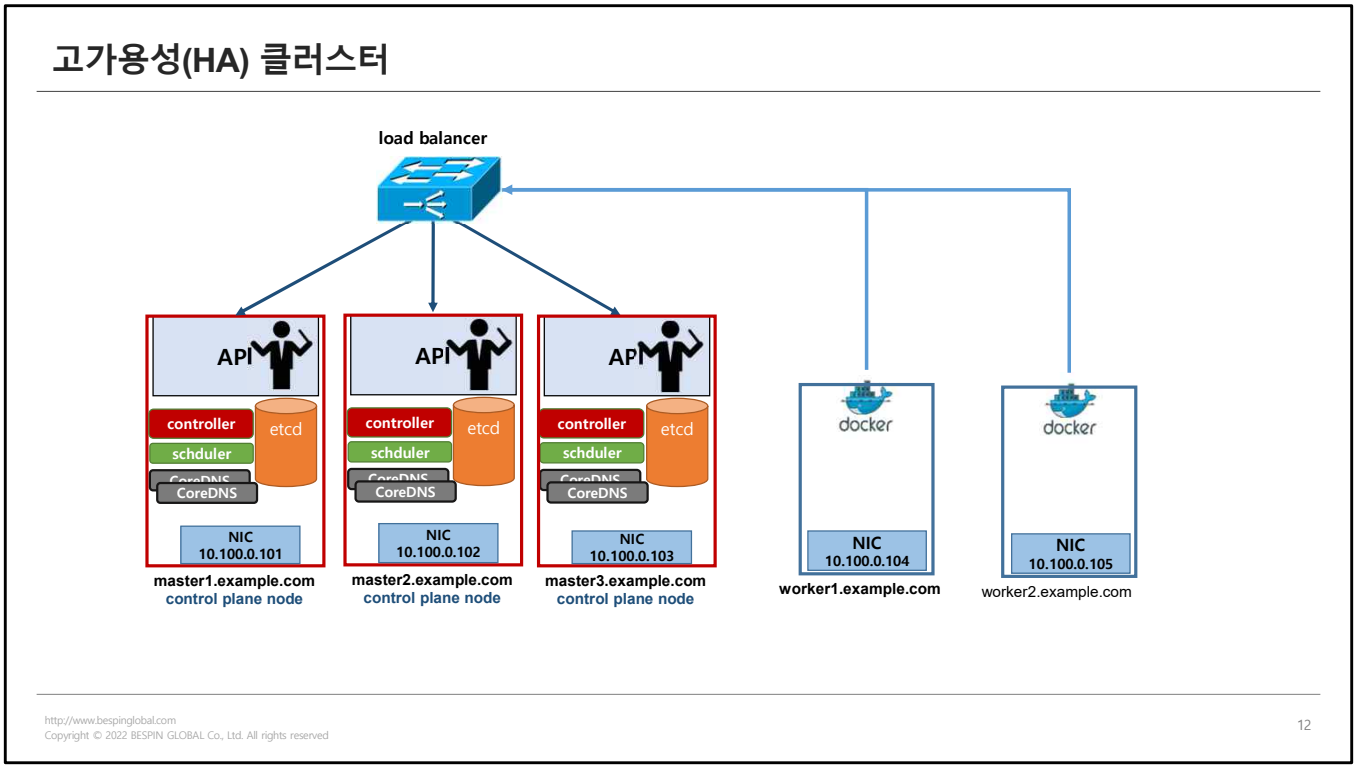
- **API Server**
  쿠버네티스 REST API를 공개한다. 모든 데이터를 etcd 클러스터에 저장하므로 쉽게 수평 확장이 가능하다. API 서버는 쿠버네티스 control-plane 을 구체화 한 것이다.
- **etcd**
  분산데이터 저장소. 쿠버네티스는 이것을 사용해 전체 클러스터의 상태를 저장한다. 규모가 작고 일시적인 클러스터의 경우에는 etcd 단일 인스턴스가 다른 마스터 컴포넌트와 함께 동일한 노드에서 동작할 수 있다. 하지만 대규모 클러스터의 경우 일반적으로 이중화와 고가용성을 위해 3개에서 많으면 5개 노드의 etcd 클러스터를 갖기도 한다.
- **controller manager**
  컨트롤러 매니저는 API를 사용해 클러스터의 상태를 감시하고, 클러스터를 원하는 상태로 조정한다. 컨트롤러 매니저에는 replication controller, pod controller, service controller, endpoint controller 등이 포함된다.
- **scheduler**
  Kube scheduler는 node에 pod를 스케줄링 하는 역할을 담당한다. 서비스의 리소스 요구사항, 서비스 요구사항, hardware와 software 정책 제약 사항 등 여러 가지 상호작용을 요인으로 고려해야 하므로 사실상 매우 복잡한 작업이다.

Add-On Component

- Dashboard
- CNI(Container Network Interface Plugin)
- Ingress
- coreDNS(Kubernetes DNS Server)
  쿠버네티스 1.3버전부터 core DNS 서비스를 표준클러스터에 포함하였고, 일반적인 Pod로 스케줄링 된다. headless 서비스를 제외한 모든 서비스는 DNS 이름을 가지며 Pod 역시 DNS 이름을 가진다. 이것은 자동검색(automatic discovery)에 매우 유용하다.

Worker Node components
클러스터의 Worker Node가 클러스터 Master components와 상호작용하고, 클러스터를 실행하고 업데이트하기 위한 워크로드를 받기 위해서는 몇 가지 관련 components가 필요하다.

- **kube-proxy**
  - 각 node에서 저수준의 네트워크 관리 업무를 수행한다.
  - 쿠버네티스 서비스를 지역적으로 반영하고 TCP와 UDP 포워딩을 수행하며 환경변수나 DNS를 통해 클러스터 IP를 찾는다.
- **kubelet**
  - Kubelet은 쿠버네티스를 대표하는 Node라고 할 수 있다.
  - Kubelet은 마스터 컴포넌트와 통신을 수행하며 실행 중인 포드를 관리하고 감독한다.
  - 구체적인 역할은 다음과 같다.
    - API 서버에서 Pod Secret 다운로드
    - 볼륨 마운트
    - 포드의 컨테이너 실행
    - Node와 각 Pod의 상태 보고
    - 실행중인 컨테이너의 활성 여부조사

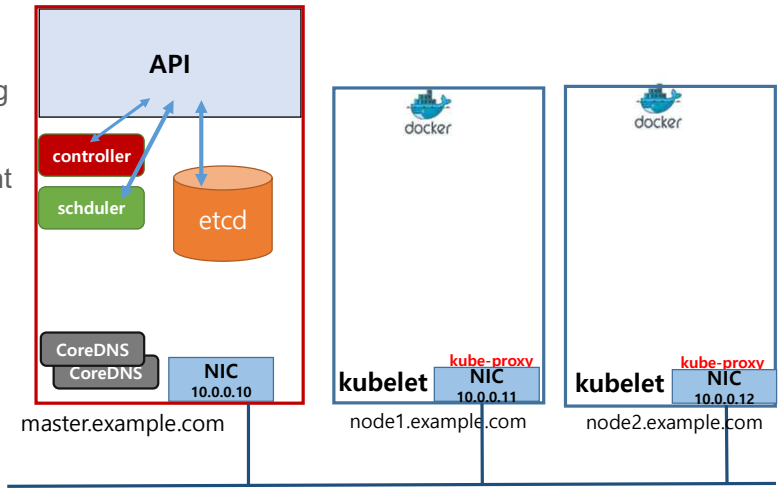


고가용성(HA : High Availability) 쿠버네티스 클러스터

쿠버네티스는 마스터 노드에 내부적으로 API-SERVER를 가지고 있어 마스터 노드를 통해 워커 노드로 통신이 전달된다. 대량에 통신량이 발생하게 되면 Master Node는 부하를 많이 받아 장애 발생 가능성이 커지게 되는데, Master Node를 HA 클러스터로 운영하게되면 통신량을 분산시킬 수 있고, 일부 Master Node에 장애가 발생되더라도 Worker Node의 운영환경에는 영향을 안 줄 수 있다.

Kubernetes 핵심기능

- Automatic binpacking
- Self-healing
- Horizontal scaling
- Service discovery and Load balancing
- Automatic rollouts and rollbacks
- Secret and configuration management
- Storage orchestration
- Batch execution

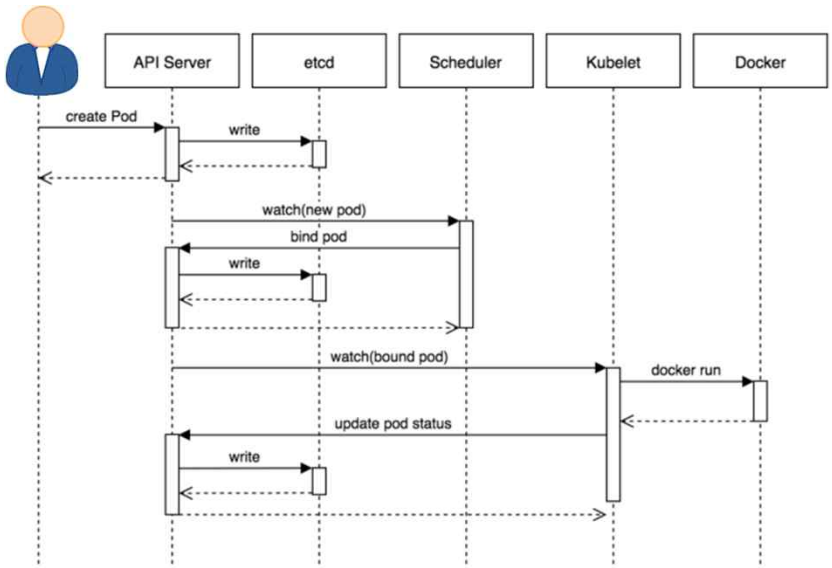


Kubernetes 핵심기능

다음은 쿠버네티스가 제공하는 몇 가지 기능이다.

- **Automatic binpacking**
  가용성에 대한 희생 없이, 리소스 사용과 제약 사항을 기준으로 자동으로 컨테이너를 스케줄
- **Self-healing**
  자동으로 문제가 발행한 노드의 컨테이너를 대체(룰/정책에 따른 헬스 체크)
- **Horizontal scaling**
  CPU와 메모리와 같은 리소스 사용에 따라 자동으로 애플리케이션을 확장경우에 따라서, 사용자 정의 측정값을 기준으로 한 동적인 확장 가능
- **Service discovery and Load balancing**
  container에 고유한 IP를 부여. 여러 개의 Container를 묶어 단일 service로 부여하는 경우 단일 name으로 접근하도록 로드 밸런싱을 제공
- **Automatic rollouts and rollbacks**
  다운타임 없이 애플리케이션의 새로운 버전 및 설정에 대한 롤아웃/롤백 가능
- **Secret and configuration management**
  애플리케이션의 secret과 configuration 정보를 이미지와 독립적으로 구분하여 별도의 이미지 재생성 없이 관리
- **Storage orchestration**
  소프트웨어 정의 저장장치(Software Defined Storage)를 기반으로 로컬, 외부 및 저장소 솔루션 등을 동일한 방법으로 컨테이너에 마운트 할 수 있음
- **Batch execution**
  CI 워크로드와 같은 Batch성 작업 지원crontab 형식으로 스케줄링도 가능

Kubernetes 시퀀스 다이어그램



Control Plane의 스퀀스 다이어그램

API 서비스를 기준으로 Kubernetes 요청이 처리되는 흐름을 위의 그림을 통해 확인해보자.

1. 스퀀스 다이어그램을 통해 API Server의 역할을 설명하시오.

2. 스퀀스 다이어그램을 통해 Kubelet과 Docker의 역할을 설명하시오.

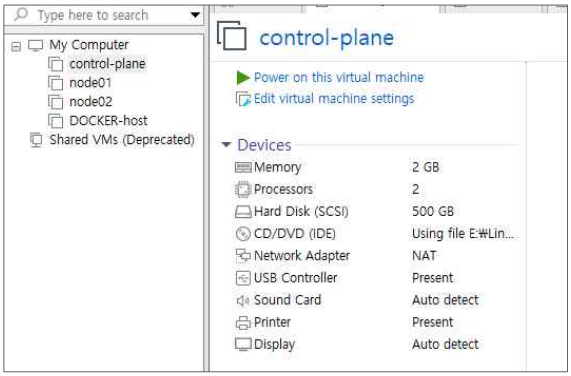
3. 스퀀스 다이어그램을 통해 etcd를 설명하시오.

4. 스퀀스 다이어그램을 통해 scheduler와 API Server의 관계를 설명하시오.

Kubernetes 설치

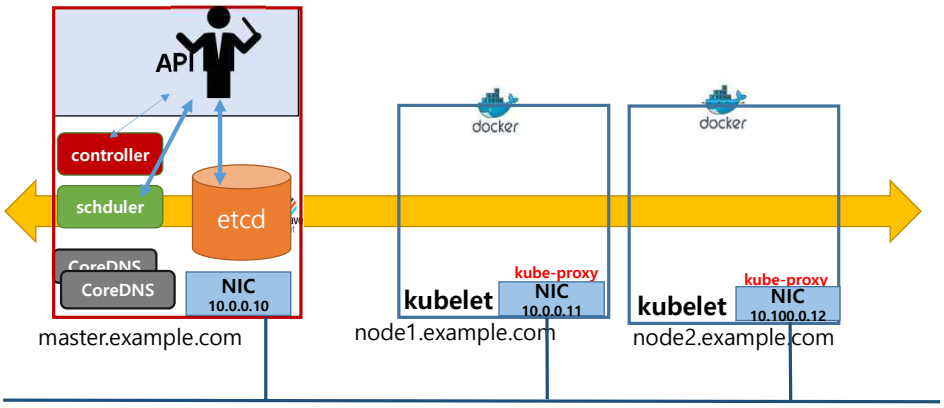
Kubernetes Install 방법

- Minikube (It is a single node kubernetes cluster)
- kops(Multi node kubernetes setup into AWS)
  - 쿠버네티스 클러스터의 자동 프로비저닝을 위한 CLI tool
  - Kubernetes 프로젝트의 일부로 AWS 전용도구에서 구글 클라우드 지원, other
- Kubespray
  - on-premises 베어 메탈 서버에 설치하는 중점을 둠
  - 고가용성과 다중 플랫폼을 지원
- Kubeadm (Multi Node Cluster in our on-premises)
- Puppet Kubernetes Module



kubernetes install with kubeadm

- kubeadm을 이용한 single control-plane cluster 구축



kubeadm을 이용해 single control-plane cluster 구축

1. Before you begin
A compatible Linux host. The Kubernetes project provides generic instructions for Linux distributions based on Debian and Red Hat, and those distributions without a package manager.

- 2 GB or more of RAM per machine (any less will leave little room for your apps).
- 2 CPUs or more.
- Full network connectivity between all machines in the cluster (public or private network is fine).
- Unique hostname, MAC address, and product_uuid for every node. See here for more details.
- Certain ports are open on your machines. See here for more details.
- Swap disabled. You MUST disable swap in order for the kubelet to work properly.

```
# Disable firewall
systemctl stop firewalld
systemctl disable firewalld

# Swap disabled. You MUST disable
swapoff -a && sed -i '/swap/s/^/#/' /etc/fstab
```

2. Letting iptables see bridged traffic

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

3. Installing runtime
파드가 노드에서 실행될 수 있도록 클러스터의 각 노드에 컨테이너 런타임을 설치해야 한다.
기본적으로, 쿠버네티스는 컨테이너 런타임 인터페이스(CRI)를 사용하여 사용자가 선택한 컨테이너 런타임과 인터페이스한다.
kubelet은 빌트인 dockershim CRI 구현을 통해 도커와 통합된다.

```
# Runtime       Path to Unix domain socket
# Docker        /var/run/dockershim.sock
# containerd    /run/containerd/containerd.sock
# CRI-O         /var/run/crio/crio.sock

#docker install link: https://docs.docker.com/engine/install/centos/
yum install -y yum-utils
yum-config-manager ₩
```

```
    --add-repo ₩
    https://download.docker.com/linux/centos/docker-ce.repo

yum install docker-ce docker-ce-cli containerd.io -y

systemctl enable --now docker
docker version


# 컨테이너의 cgroup 관리에 systemd를 사용하도록 Docker 데몬을 구성
mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

systemctl enable docker
systemctl daemon-reload
systemctl restart docker
```

4. Installing kubeadm, kubelet and kubectl

```
# Installing kubeadm, kubelet and kubectl
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-₩$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF

# Set SELinux in permissive mode (effectively disabling it)
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

sudo systemctl enable --now kubelet
```

5. Install a single control-plane Kubernetes cluster

```
# 어떤 CNI?
# Create a single-host Kubernetes cluster
# kubeadm init --pod-network-cidr=192.168.0.0/16
# Install Calico
# kubectl apply -f https://docs.projectcalico.org/manifests/tigera-operator.yaml

# initialize the control-plane
kubeadm init


# kubectl 명령을 쓸 수 있도록 허용
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config

## token 별도로 저장
cat > token.txt
kubeadm init 명령 시 출력된 토큰을 저장해서 이후에 worker node들이 join할 때 사용
```

Installing a Pod network add-on
#CNI - weave
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '₩n')"
kubectl get nodes

6. Worker Nodes Join
worker nodes
kubeadm join 10.100.0.104:6443 --token bxxxxxxxxxxxxxxxxxxx ₩
--discovery-token-ca-cert-hash sha256:5cc1xxxxxxxxxxxxxxxxxxx

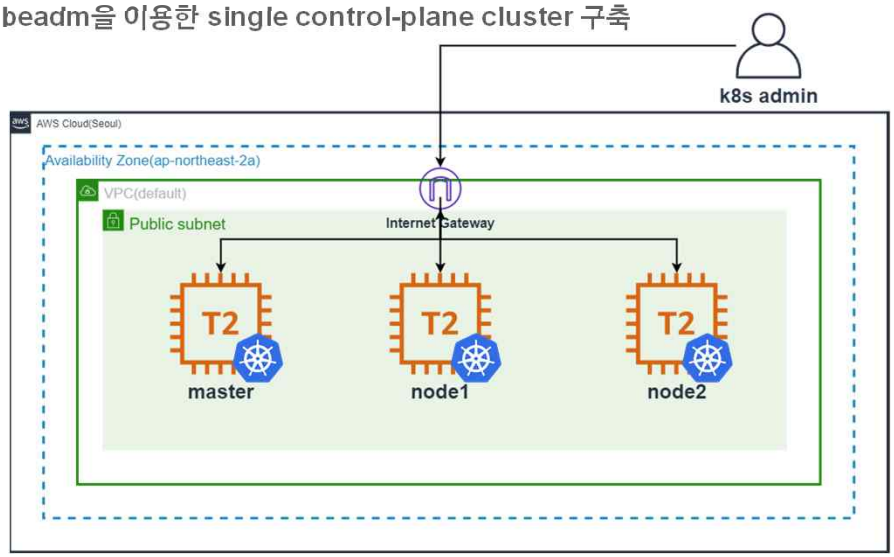
7. kubectl command 자동완성 기능 추가
yum install -y bash-completion
source /usr/share/bash-completion/bash_completion

source <(kubectl completion bash)
echo "source <(kubectl completion bash)" >> ~/.bashrc

8. 설치확인
kubectl get nodes
kubectl get nodes -o wide
kubectl describe node node1.example.com
kubectl get pod --all-namespaces

실습 | kubernetes install with kubeadm On Cloud

- **kubeadm을 이용한 single control-plane cluster 구축**



Security Group
- name: **sg_k8s_all**
- VPC: default VPC
- Inbound rules
  - type : All traffic
  - source : any(0.0.0.0/0)

클라우드 가상머신에 k8s 설치

1. 가상머신 생성
- 서울 리전
- AMI: Amazon Linux 2 AMI(HVM), SSD Volume Type
- Instance Type : **t2.mdium(cpu 2, memory 4GiB)**
- Numer of instances: **3**

- Network: Default VPC
- Subnet: **ap-northeast-2a**

- Tag : **Name=k8s**

- Storage : **20Gi SSD**

- Security Group:
  - name: **sg_k8s_all**
  - Description : Open full traffic for k8s service
  - [Add Rule]
    - Type : All traffic
    - Protocol: All
    - Port Range: 0-65535
    - Source : Custom 172.31.0.0/16(default VPC)
    - Description : Open All traffic for k8s
  - Login Key pair 선택 및 생성

2. Instances 이름 변경
Name: k8s-master
Name: k8s-node1
Name: k8s-node2

3. Elastic IP 설정 - k8s-master, k8s-node1, k8s-node2

4. Remote Login(ec2-user) 후 hostname 변경 : master, node2, node3
$ sudo -i

5. 표준 시간대 변경
https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/set-time.html
date
cat /etc/localtime
TZif2UTCTZif2
UTC0

```
rm /etc/localtime

ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime
date
```

* node1, node2도 동일하게 표준 시간대 변경

6. vi 에디터 기본 구성

```
cat > .vimrc
set paste
```

7. kubeadm 설치 및 k8s 구성

- docker 설치

```
# Letting iptables see bridged traffic
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system

# Runtime     Path to Unix domain socket
# Docker      /var/run/dockershim.sock
# containerd                    /run/containerd/containerd.sock
# CRI-O       /var/run/crio/crio.sock
yum install docker -y
systemctl enable --now docker
docker version


# 컨테이너의 cgroup 관리에 systemd를 사용하도록 Docker 데몬을 구성
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
systemctl enable --now docker && sudo systemctl status docker
docker version
```

- kubeadm 설치

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-₩$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF


yum install -y kubelet-1.22.4-0 kubeadm-1.22.4-0 kubectl-1.22.4-0 --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
```

- control-plane 구성 : kubeadm init

```
# initialize the control-plane
```

kubeadm init

- CNI - Weave 설치
  #CNI - weave
  kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '₩n')"
  kubectl get nodes

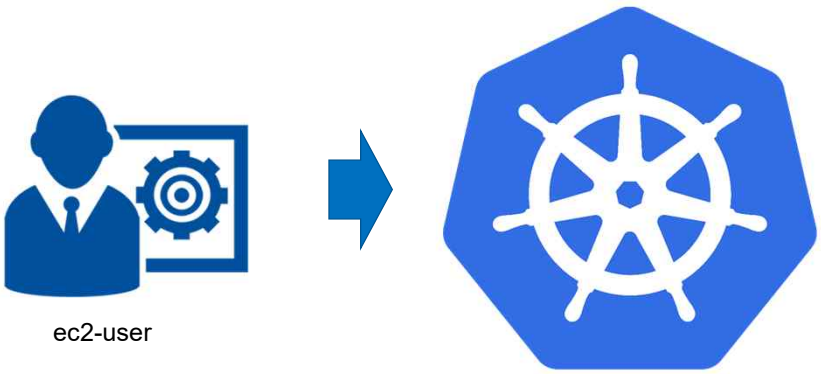
- Worker-Node Join
  kubeadm join 172.XX.X.XXX:6443 --token bxxxxxxxxxxxxxxxxxxx ₩
  --discovery-token-ca-cert-hash sha256:5cc1xxxxxxxxxxxxxxxxxxx

- kubectl command 자동완성 기능 추가
  source <(kubectl completion bash)
  echo "source <(kubectl completion bash)" >> ~/.bashrc

8. 설치확인
kubectl get nodes
kubectl get nodes -o wide

쿠버네티스 관리자 계정 추가

- 쿠버네티스 관리자
  - kubectl, kubeadm 명령을 실행할 수 있는 권한을 가진 사용자



쿠버네티스 관리자 계정 추가

쿠버네티스 관리 명령인 kubectl 을 사용할 수 있도록 권한 주기

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

실습 : **ec2-user** 계정 사용자가 **kubectl** 명령으로 쿠버네티스 관리하도록 설정

- ec2-user 계정 사용자에게 kubectl 명령을 이용해서 쿠버네티스 관리를 할 수 있도록 허용

```
mkdir ~admin/.kube
cp /etc/kubernetes/admin.conf ~admin/.kube/config
chown admin.admin ~admin/.kube/config
```

- 구성 확인. ec2-user 유저로 다음 명령 실행

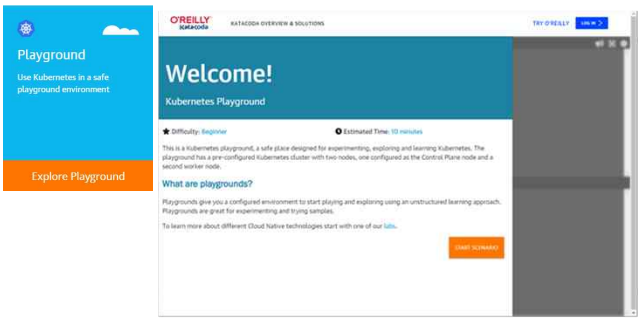
```
kubectl get nodes
```

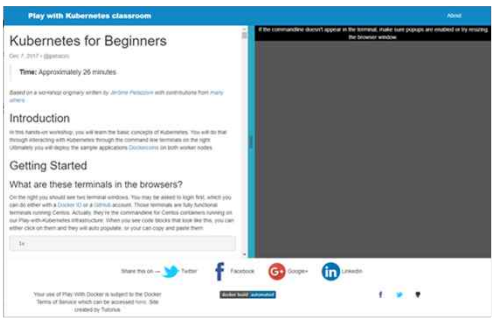
- kubectl command 자동완성 기능 추가

```
source <(kubectl completion bash)
echo "source <(kubectl completion bash)" >> ~/.bashrc
```

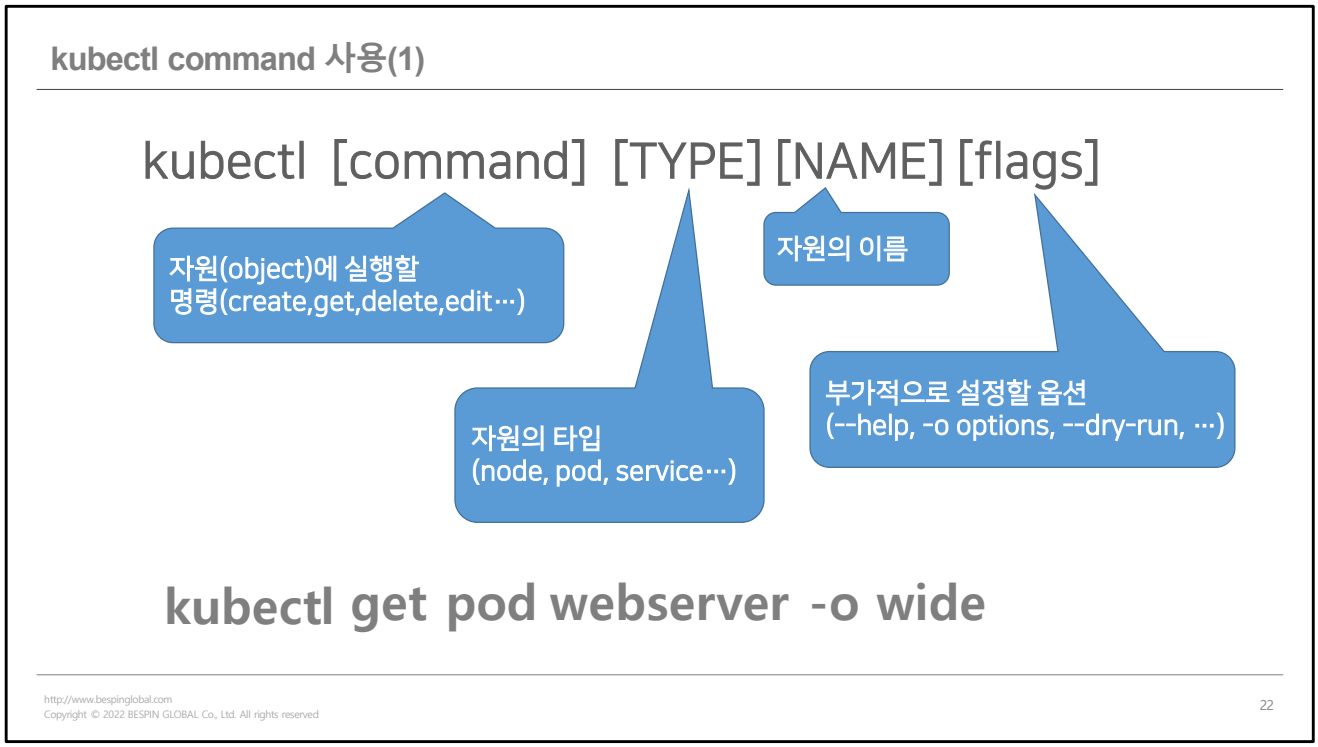
Kubernetes Community

- 누구나 참여하여 기여 가능
  - 공식 깃허브 : https://github.com/kubernetes
  - 공식 슬랙 채널: http://slack.k8s.io/
  - 페이스북 그룹: https://www.facebook.com/groups/k8skr
- 웹기반 실습환경
  - https://www.katacoda.com/courses/kubernetes
  - https://training.play-with-kubernetes.com/





설치 후 기본 명령어 사용

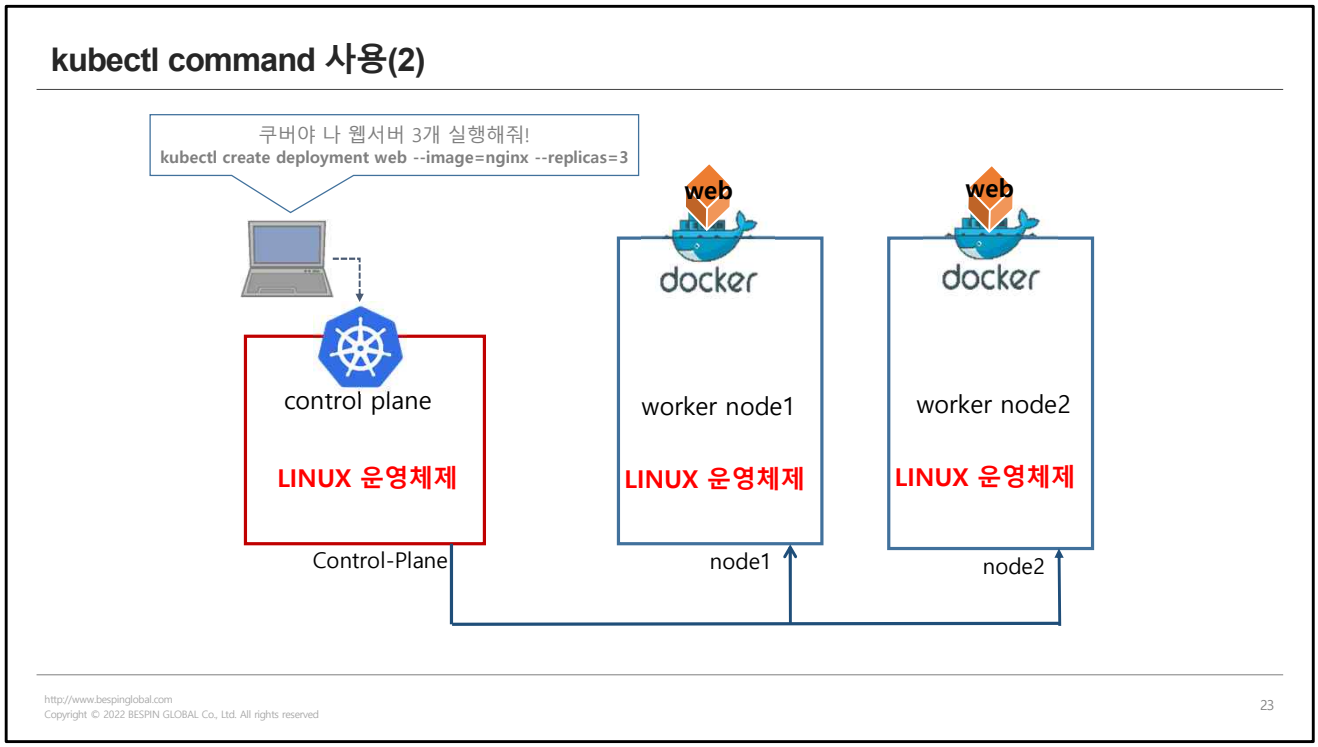


kubectl command 사용

$ kubectl --help
$ kubectl command --help

kubectl run <자원이름> <옵션>
$ kubectl create -f obj.yaml
$ kubectl apply -f obj.yaml

kubectl get <자원이름> <객체이름>
kubectl edit <자원이름> <객체이름>
kubectl describe <자원이름> <객체이름>
kubectl delete <자원이름> <객체이름



kubectl command 사용

실습: kubectl 명령을 통해 nginx : 1.14를 동작

1. kubectl run 명령을 이용해 웹 서버 컨테이너(nginx:1.15) 실행하기
2. deployment API를 이용해 nginx:1.14 컨테이너를 실행
3. pod yaml 파일 만들기
4. 단축 명령 사용확인
5. 실시간 pod 동작 상태 모니터링
6. kubectl 명령어 도움말 보기

심화 | kubectl 명령어 연습

kubectl 명령어 연습

1. kubectl 명령어 사용법 익히기
   - kubernetes master와 worker node의 정보를 확인 : kubectl cluster-info
   - 지원하는 API resource 정보보기 : kubectl api-resources
   - Pod의 API version과 단축키는 무엇인가?
   - statefulsets의 API version과 단축키는?

2. 다음 조건에 실행되어야 할 명령어는 무엇인가?
   - node1에 설치된 OS정보, memory, cpu등의 하드웨어 정보를 확인하시오.
   - webserver.yaml 파일의 replicas를 2로 변경한 후 실행하시오.
   - 현재 pod는 어느 노드에서 실행 중인가?
   - 현재 동작중인 pod의 정보를 자세히 확인하시오.

지금까지 배운 내용을 기준으로

- 컨테이너 오케스트레이션
  - 컨테이너와 컨테이너 오케스트레이션을 설명할수 있어야합니다.
  - 컨테이너 오케스트레이션이 필요한 이유를 설명할수 있어야합니다.

- 쿠버네티스 아키텍쳐
  - Master Node와 Worker Node를 설명할수 있어야합니다.
  - Master Node의 컴포넌트의 역할을 설명할수 있어야합니다.
  - 컨테이너 동작원리를 설명할수 있어야합니다.

- 쿠버네티스 설치
  - kubeadm으로 쿠버네티스 설치를 할수 있어야합니다.
  - kubernetes 관리자 계정을 생성할수 있어야합니다.

- 설치후 기본 명령어 사용
  - kubectl 명령의 사용법을 알아야합니다.

문제 : 쿠버네티스 핵심기능을 설명하시오.