1장

Kubernetes 소개 및 설치

전체 내용

Kubernetes 소개

Kubernetes
설치 및 구성

쉬어가는 코너

Kubernetes 소개

- Container 강점 및 약점
- Container 약점을 보완한 K8S
- 서비스 관리를 개별적에서 집단적으로
- 개발 배경
- kubernetes.io 둘러보기
- Kubernetes 특징
- Container Orchestration이란?

Kubernetes 소개

- **Container 강점**-Docker 관점
  - Application을 운영할 때 VM 보다 가볍다
  - Application 배포를 민첩하게 할 수 있다
  - 지속적인 개발과 통합 및 배포
  - 개발과 테스트, 운영 전반에 걸친 일관성 있는 환경 제공
  - Application 중심으로 관리
  - Resource를 충분하게 활용할 수 있다
  - Image를 Docker host에서도 가능하고, Cloud에서도 가능한 이식성을 제공한다
    - Azure, AWS, GCE와 같은 주요 Cloud Vendor에서 Container Hosting Service를 제공한다

Kubernetes 소개

- **Container 약점**-Docker 관점
  - 개발자 입장에서
    - Application Code 작성 ➔ **Build ➔ Ship ➔ Run**
    - **Container가 많으면** docker 명령어로 일일이 관리하기가 어렵다
  - 운영자 입장에서
    - Container가 갑자기 죽으면 어떻게 살리지?
    - Container의 업그레이드 및 Rollback은 어떻게하지?
    - 서비스의 이상 및 부하의 모니터링을 어떻게 할까?

Kubernetes 소개

- **Container 약점을 보완한 K8S-Container Orchestration**
  - 복잡한 Container 환경을 효과적으로 관리하기 위한 도구이다
    - 즉, Container 관리를 나 대신 K8S 네가 해줘!
  - **Clustering**
    - 여러 node의 중앙 관리
  - **상태(State) 관리**
    - 어떠한 환경에서도 실행되어야 하는 Pod 개수 보장
    - 각종 Controller의 replicas 설정 적용
    - Pod의 livenessProbe
  - **부하 조정(Scheduling)으로 배포 관리**
    - Pod의 배포시 부하 정도가 가장 적절한 Node를 선정하여 자동 배포
  - **배포 버전 관리**
    - Deployment로 새로운 버전의 App을 이전 버전을 유지한 채 Rolling Update 및 문제 상황 발생시 이전 버전으로 Roll back을 쉽게 구현

Kubernetes 소개

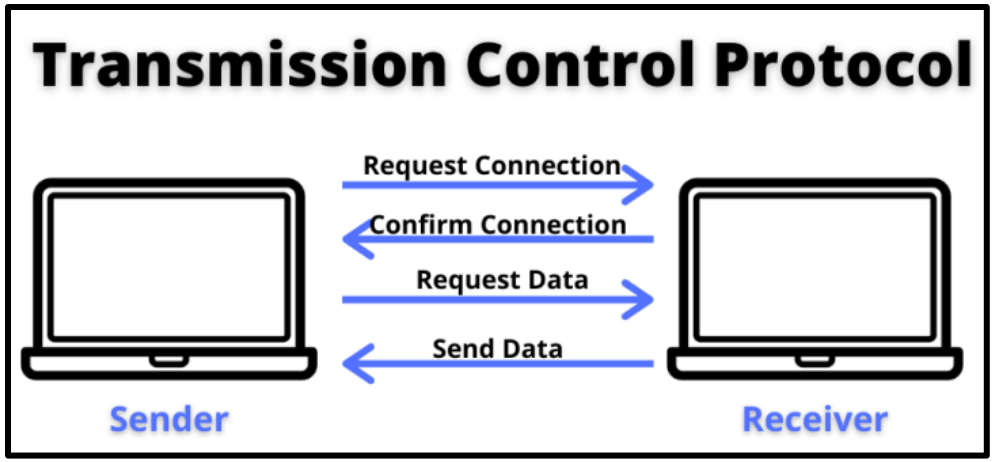
- **Container 약점을 보완한 K8S-Container Orchestration**
  - **Service Discovery**
    - Pod 집합에서 실행중인 애플리케이션을 **네트워크 서비스로 노출하여** Cluster 외부에 있는 User가 Application을 이용하도록 네트워크 경로 제공
    - K8S는 Pod에게 고유한 IP 주소와 Pod 집합에 대한 단일 DNS 명을 부여하고, 그것들 간에 로드-밸런스를 수행
    - 동일한 포트(80 포트 등)로 서비스하는 여러 개의 Container 운영 가능
  - **Volume Storage**
    - 각 Node별로 다양한 Storage 제공
    - node1은 로컬 디스크, node2는 NFS, node3는 AWS EBS 및 GCE PD를 이용
    - Volume이란 Pod에 종속되는 디스크**(Container 단위가 아니고 Pod 단위)**로서, 해당 Pod에 속해 있는 여러 개의 Container가 공유해서 사용

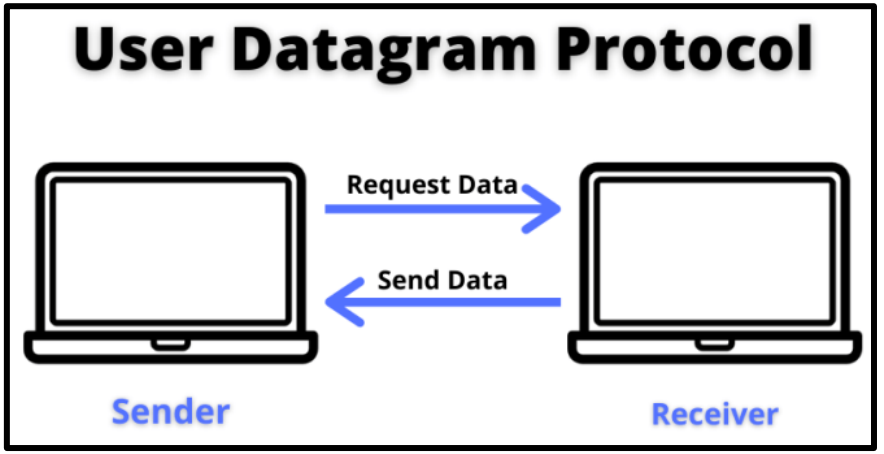
Kubernetes 소개

- 서비스 관리를 **개별적 → 집단적**
  - 서버를 관리할 때 각 서버 이름을 애완견 이름으로 짓는 습관이 있었다
  - 특정한 서버가 고장 나는 경우에는 복구하는데 어려움이 있었다
  - 몇 대의 서버가 아닌 수 십, 수백대의 서버를 관리하려면 애완견(Build Machine)이 아닌 소 떼(Service Machine)처럼 집단적으로 관리해야 한다
  - k8s는 소 떼처럼 집단 접근 방식을 취하고, Container를 특정 시스템에 할당하는 모든 부분을 담당한다

Kubernetes 소개

- 서비스 관리를 **개별적 → 집단적**
  - 대부분의 상황에서 **개별 시스템(node)과 상호작용할 필요는 없다**
  - Kubernetes는 비교적 Stateless(상태 비저장) 업무에 적합하다
    - 컴퓨터학에서는 **관계의 상태를 유지하는 것을 Stateful**이라고 하고, **관계의 상태를 유지하지 않는 것을 Stateless**라고 한다
    - Computer Network Communication의 예
      - 4계층 프로토콜: **TCP(stateful), UDP(stateless)**
      - 7계층 프로토콜: **HTTP(stateless)** → HTTP도 4계층에서는 TCP를 사용하지만 7계층에서는 작업 후에 **연결을 끊기 때문에** 다시 빠르게 연결하기 위해 보조적으로 Data를 Cache하여 처리하기도 한다



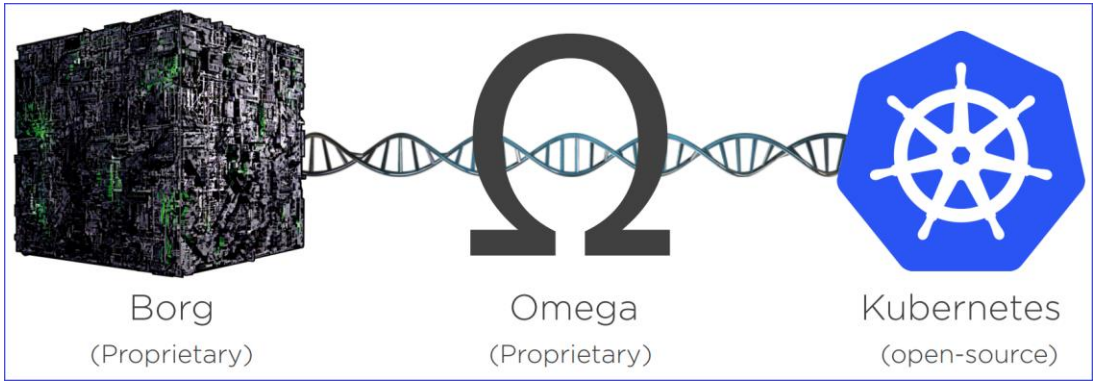


Kubernetes 소개

- **개발 배경**
  - Container는 stateless, immutable, mortal (상태를 가지고 있지 않고, 변화하지 않으며, 언제든 죽을 수 있는)개념을 기반으로 아키텍처를 구성하다 보면 운영에 앞서 반드시 필요한 것이 Container Orchestration(=Kubernetes)이다
  - Container Orchestration은 **다수의 Container를 다수의 node(Cluster)에 적절하게 분산 실행하고, 원하는 상태(Desired State)로 실행상태를 유지해 주고, 다운타임 없이 유동적으로 스케일을 확장/축소(Scale)할 수 있게 도와준다**
  - 사용자가 Container에 대한 동작과 다른 Container와의 관계를 정의하면 **배포/운영/스케일링**에 문제가 없도록 자동으로 관리되는 운영 시스템이라고 할 수 있겠다.

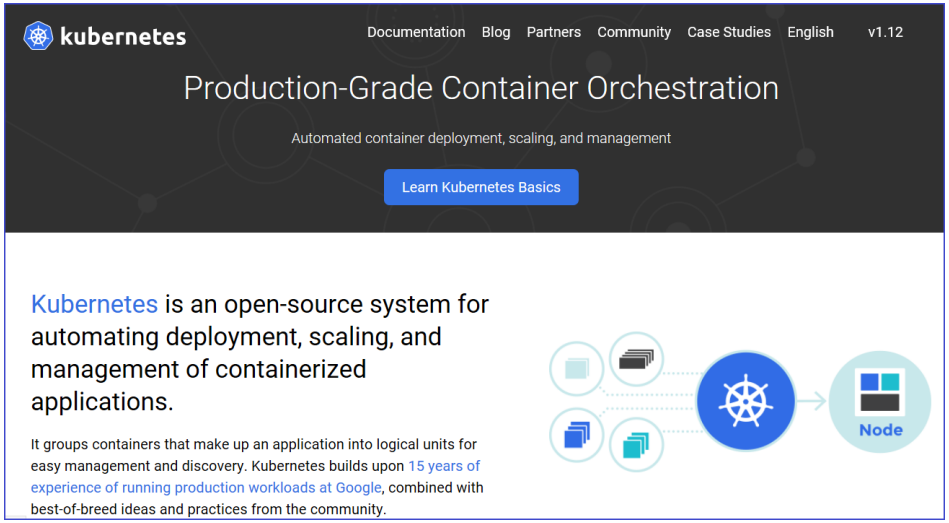
Kubernetes 소개

- **개발 배경**
  - Kubernetes 또는 k8s 로 약칭함
  - docker는 container를 **규모에 맞게 늘려가도록 배치하는 기능이 부족하다**
  - Kubernetes는 Containerized application를 관리하고 자동 배포, 확장, 관리하는 open-source system이다
  - 많은 수의 container를 배포하는 Orchestrator를 Google이 개발하였다
  - 하지만 CNCF(Cloud Native Computing Foundation)에 가입하면서 Container 기반의 Application 분야의 확실한 Leader가 되었다
  - k8s는 Google이 2014년 6월에 Open Source Project로 발표



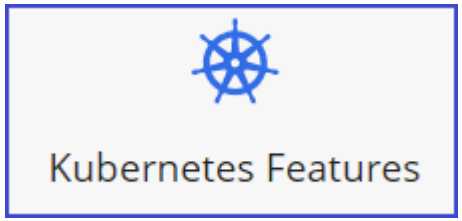
Kubernetes 소개

- **개발 배경**
  - 2014년 공개 후 RedHat, VMware, Canonical 등 오픈소스 커뮤니티 전체 참여로 급속하게 발전
  - 2015년 7월에 v1.0 출시
  - Go Language로 개발
  - 참고 자료
    - https://github.com/kubernetes/kubernetes
    - @kubernetesio
    - slack.k8s.io
    - kubernetes.io

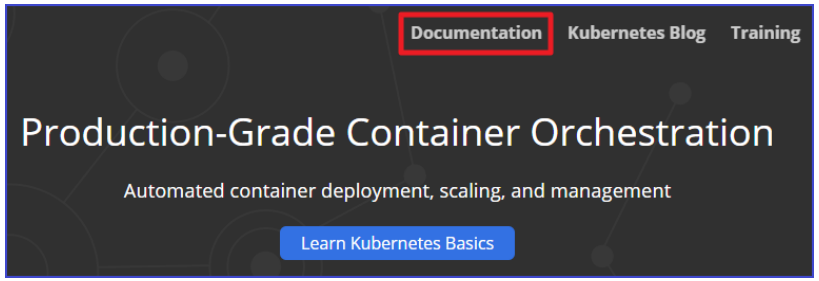


Kubernetes 소개

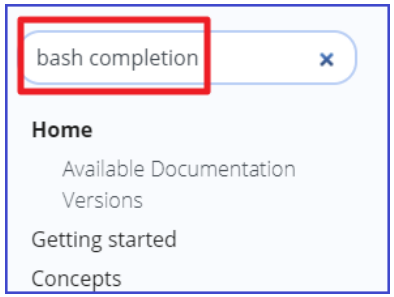
- **kubernetes.io 둘러보기**
  - Kubernetes 특징

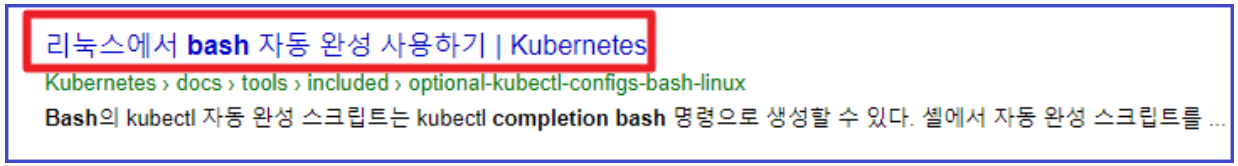


  - Documentation



    - 자료 검색





Kubernetes 소개

- **Kubernetes란?**
  - 계속 증가하고 있는 엄청난 수의 서비스와 기능이 포함된 플랫폼이다
  - Kubernetes의 핵심 기능은 인프라 전체에 걸쳐 Container에서 작업부하를 조정(Schedule)할 수 있는 능력이다

- **Kubernetes 특징**
  - Automatic bin packing
  - Self-healing
  - Horizontal scaling
  - Service discovery and load balancing
  - Automated rollouts and rollbacks
  - Secret and configuration management
  - Storage orchestration

Kubernetes 소개

- **Kubernetes 특징-세부 설명**
  - Automatic binpacking
    - 가용성을 희생하지 않는 범위안에서 물리적 리소스를 충분히 활용해서 Pod을 (자동) 배치한다
    - binpacking이란 컨테이너가 호스트의 물리적 리소스를 충분히 활용할 수 있도록 효율적으로 고르게 배치해서 불필요한 물리적 리소스의 낭비를 줄이는 것을 의미한다.
      - 캠핑갈 때 비좁은 트렁크에 캠핑용품을 이러저리 잘 정리해서 빈공간 없이 테트리스해서 쌓아 넣는 모습을 상상(Pack in a Camping Bin)
  - Self-healing
    - Container의 실행 실패, Node(K8s에서 Pod을 실행하는 Slave Host)가 죽거나 반응이 없는 경우, health check에 실패한 경우에 해당 Container를 자동으로 복구한다
    - Kubernetes가 관리하는 가장 작은 배포단위인 Pod은 언젠가는 반드시 죽는(mortal) 것으로 정의되어 있으며 Container, Node의 장애 상황발생 시에도 운영자가 기대하는 상태로 서비스가 무사히 실행될 수 있도록 Self-healing을 지원한다.

Kubernetes 소개

- **Kubernetes 특징-세부 설명**
  - Horizontal scaling
    - Pod의 CPU 사용량 혹은 app이 제공하는 metric을 기반으로 ReplicaSet (동일한 Pod을 몇 개 띄울지 정의) 을 scaling할 수 있다.
  - Service discovery and load balancing
    - 익숙하지 않는 Service Discovery 매커니즘을 위해 Application을 수정할 필요가 없다
    - K8s는 내부적으로 Pod에 고유 IP, 단일 DNS를 제공하고 이를 이용해 load balancing 한다
  - Automated rollouts and rollbacks
    - application, configuration의 변경이 있을 경우 전체 인스턴스의 중단이 아닌 점진적으로 Container에 적용(rolling update) 가능하다
    - Release revision이 관리되고 새로운 버전의 배포시점에 문제가 발생할 경우 자동으로 이전의 상태로 Rollback을 진행할 수 있다

Kubernetes 소개

- **Kubernetes 특징-세부 설명**
  - Secret and configuration management
    - Secret key, Configuration을 관리하기 위해 Secret, ConfigMaps Object가 제공되기 때문에 이미지의 변경없이 키/설정을 업데이트할 수 있고, 이를 외부에 노출(expose)하지 않고 관리/사용할 수 있다
    - Secret은 데이터를 binary 형태로, ConfigMap은 text 형태로 저장한다.
  - Storage orchestration
    - local storage를 비롯해서 Public Cloud(GCP, AWS), Network storage등을 구미에 맞게 자동 mount할 수 있다

Kubernetes 소개

- **Container Orchestration이란?**
  - k8s의 주요 임무는 Container Orchestration이다
    - 다양한 작업을 실행하는 모든 Container가 물리적인 컴퓨터나 가상 컴퓨터에서 실행되도록 예약되어 있는지 확인하는 것이 Container Orchestration이다
    - 배포 환경과 클러스터 설정의 제약 사항에 따라 Container가 효율적으로 Packaging되어야 한다
    - 실행중인 모든 Container를 모니터링하여 **중지되거나 응답이 없거나 상태가 불량한 container를 교체한다**
  - **Container Orchestration을 "Container 배포 관리"하고도 한다**
    - 여러 Container를 배포하고 관리하는 프로세스를 최적화하는데 그 목적을 두고 있다
    - 대표적인 Container Orchestration 도구에는 Kubernetes, Docker Swarm, Apache Mesos 등이 있다

Kubernetes 소개

- **Container Orchestration이란?**
  - Provisioning
    - Container를 자동으로 배치하고 복제하며, 예약하고 시작할 수 있다
  - 구성 Script
    - 특정한 Application의 구성정보를 Container에 로드할 수 있다
  - 모니터링
    - Container의 상태를 감시하고, Load balancing을 수행할 수 있다
    - Container의 장애를 탐지하고, 새로운 Instance를 기동한다
    - 필요한 경우 Container를 추가하거나 삭제할 수 있다(확장과 축소)
  - 서비스 탐색
    - 사용자가 일일이 서비스를 지정하지 않고, Container가 적합한 자원을 찾기 위해 서비스 탐색을 사용하고, 외부로 노출시킨다

Kubernetes 설치 및 구성

- 설치 방식
- Cluster 방식의 Kubernetes 구성 방법
- Kubernetes 구성
- Kubernetes 구조
- Kubernetes Playground 소개
- Play with Kubernetes 소개
- 설치 구성도

Kubernetes 설치 및 구성

- 설치 개요
- Add-on 요소
- 설치 사전 준비
- kubernetes 설치 및 시작하기
- K8S Clustering 구성하기
- Cluster 구성정보(ConfigMap) 확인하기
- Clustering 상태 및 Pod 상태 확인하기

Kubernetes 설치 및 구성

- Pod Network 구성하기
- node를 cluster에 추가하기
- Nginx Pod을 K8S cluster에 배포하기
- Lab에 사용되는 yaml 파일 다운로드하기
- K8S 구성 내용 확인하기

Kubernetes 설치

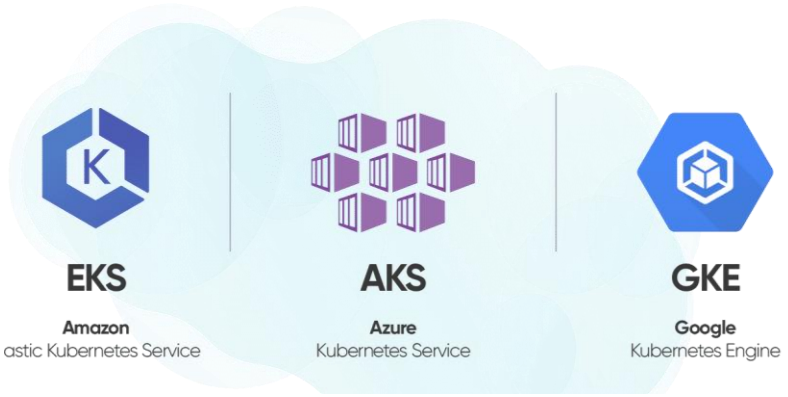
- 설치 방식
  - All-in-One 방식
    - Master와 Node가 동일한 Host에서 실행
    - 기본적인 작동 방식을 이해하는데 도움
    - 실제 환경에서는 사용하지 않음
  - Cluster 방식
    - 1개의 Master와 2개 이상의 Node로 구성
    - 필요할 때마다 Node를 추가해서 용량을 늘려 나감
    - 실제 환경에서 주로 사용

Kubernetes 설치

- **Cluster 방식의 Kubernetes 구성 방법**
  - **On-Premises (Bare metal)**
    - Kubernetes 설치를 위한 VM 및 Physical Machine 필요
  - **Public Cloud**
    - Google, Azure, AWS가 제공하는 Kubernetes 사용
  - **Free Cloud**
    - Kubernetes Playground (http://katacoda.com)
    - Play with Kubernetes (http://labs.play-with-k8s.com)

Kubernetes 설치

- **3가지 Kubernetes 배포 방법**
  - **관리형 Kubernetes**
    - Amazon **EKS,** Azure **AKS,** Google **GKE**

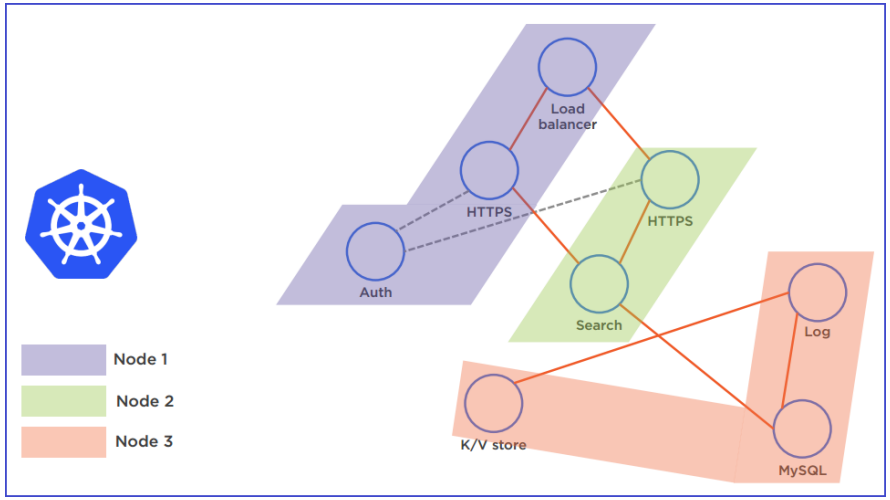


  - **설치형 Kubernetes**
    - **RANCHER**, RedHat **OPENSHIFT**

  - **직접 구성형 Kubernetes**
    - **Kubeadm**, Kubespray, Kops, Krib

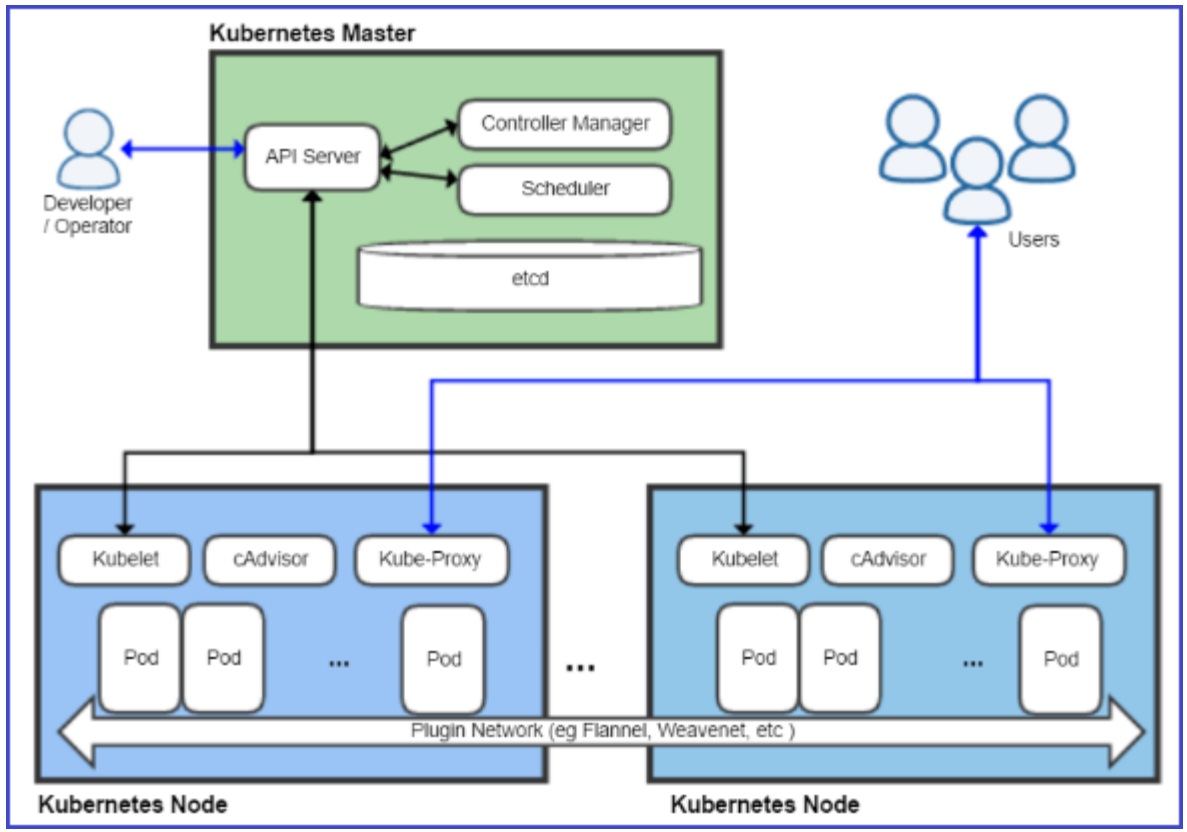
Kubernetes 설치

- **Kubernetes 구성**
  - 축구 경기를 하는 것은 선수들이지만 경기에 대한 작전을 세우고, 선수를 적절하게 배치하고, 승리할 수 있도록 코칭하는 것이 감독이다
  - Container가 실행되는 곳은 node이지만, node에 container를 배포, 확장, 관리하는 것은 Master이다
  - Containerized Application에는 web service, database, auth, log 등등이 있다



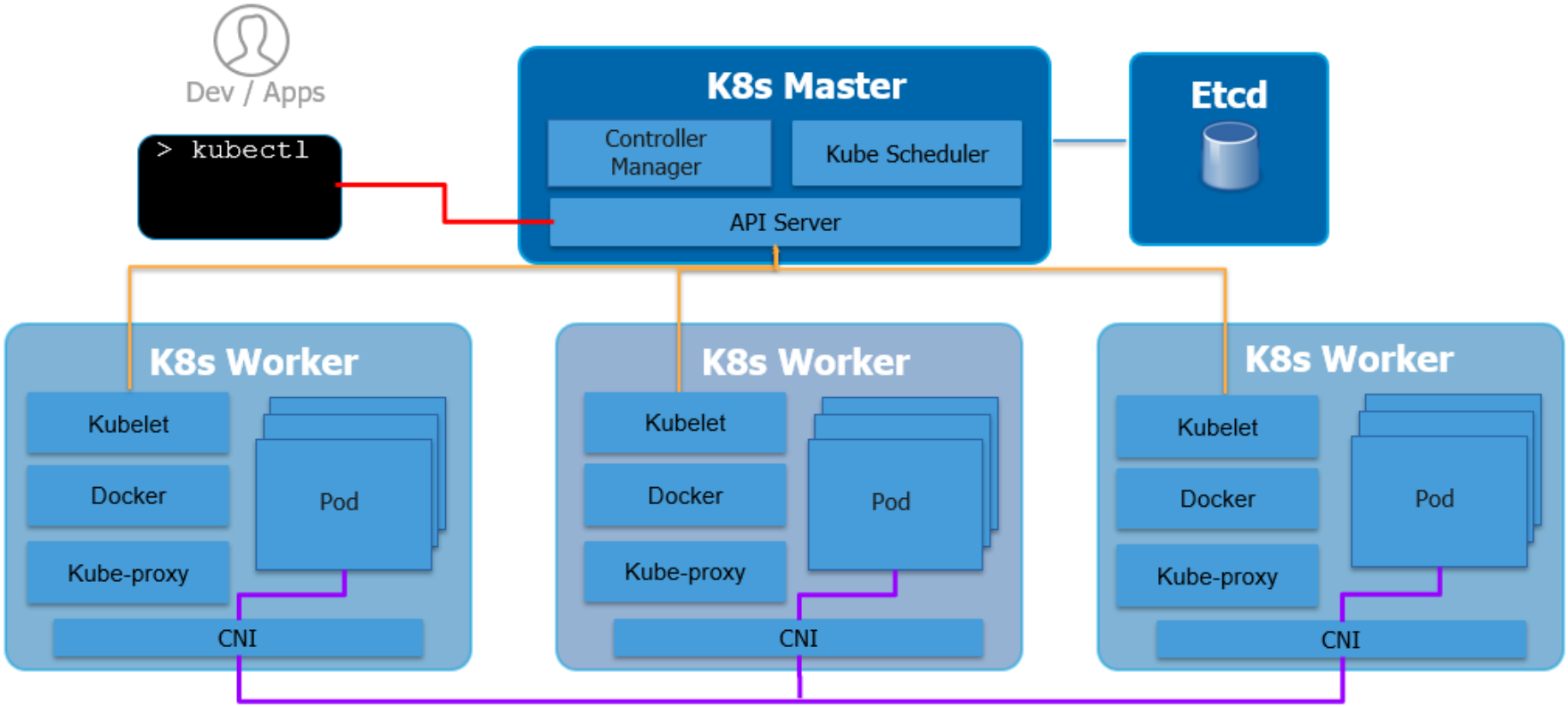
Kubernetes 설치

- **Kubernetes 구조**
  - k8s cluster = master + node
    - master: k8s cluster의 모든 상태를 저장 및 관리한다
    - node(=minion): Container가 들어 있는 Pod이 실제로 실행되는 곳



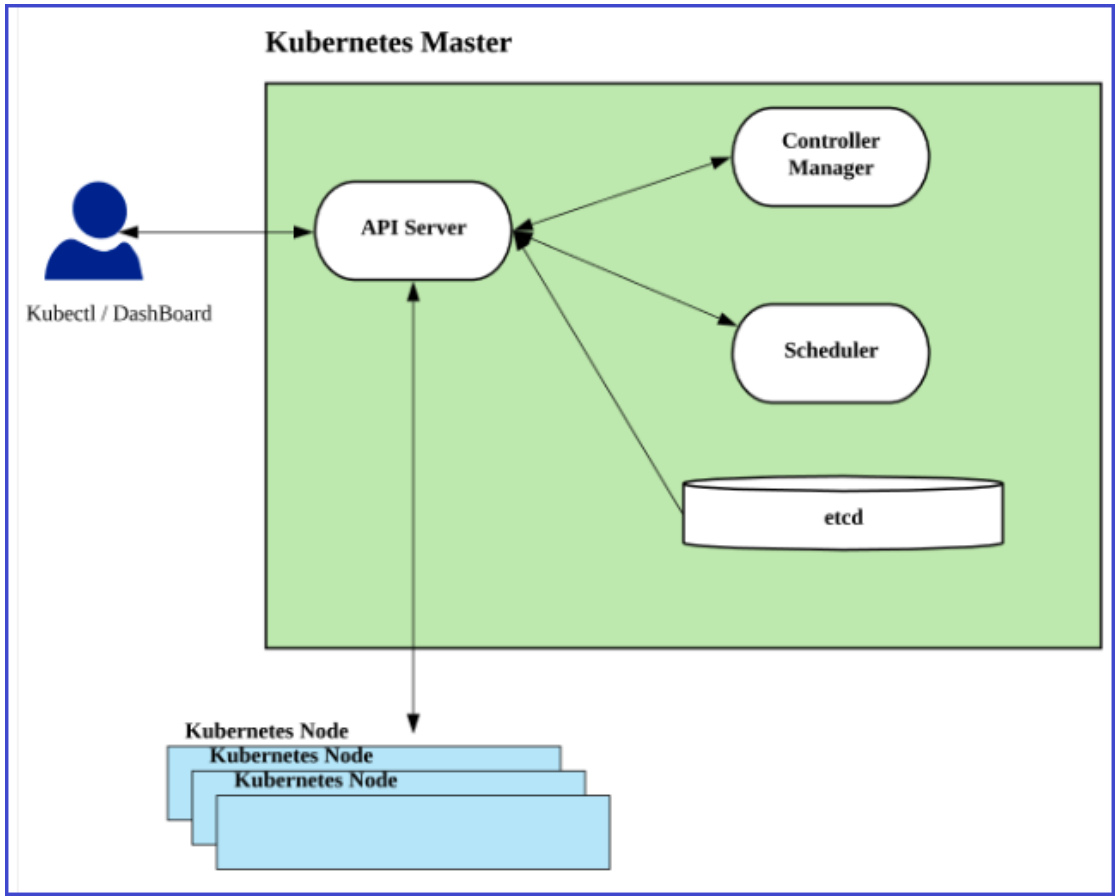
Kubernetes 설치

- **Kubernetes 구조**
  - k8s cluster = master + node
    - **kubectl** ➔ Master(**API server**) ➔ Worker(**kubelet**) ➔ Worker(**docker**)가 Container 생성



Kubernetes 설치

- **Kubernetes 구조**
  - Master(=Master node)
    - Master는 다음과 같은 K8s component로 구성된다
      - API Server, Controller Manager, Scheduler, etcd



Kubernetes 설치

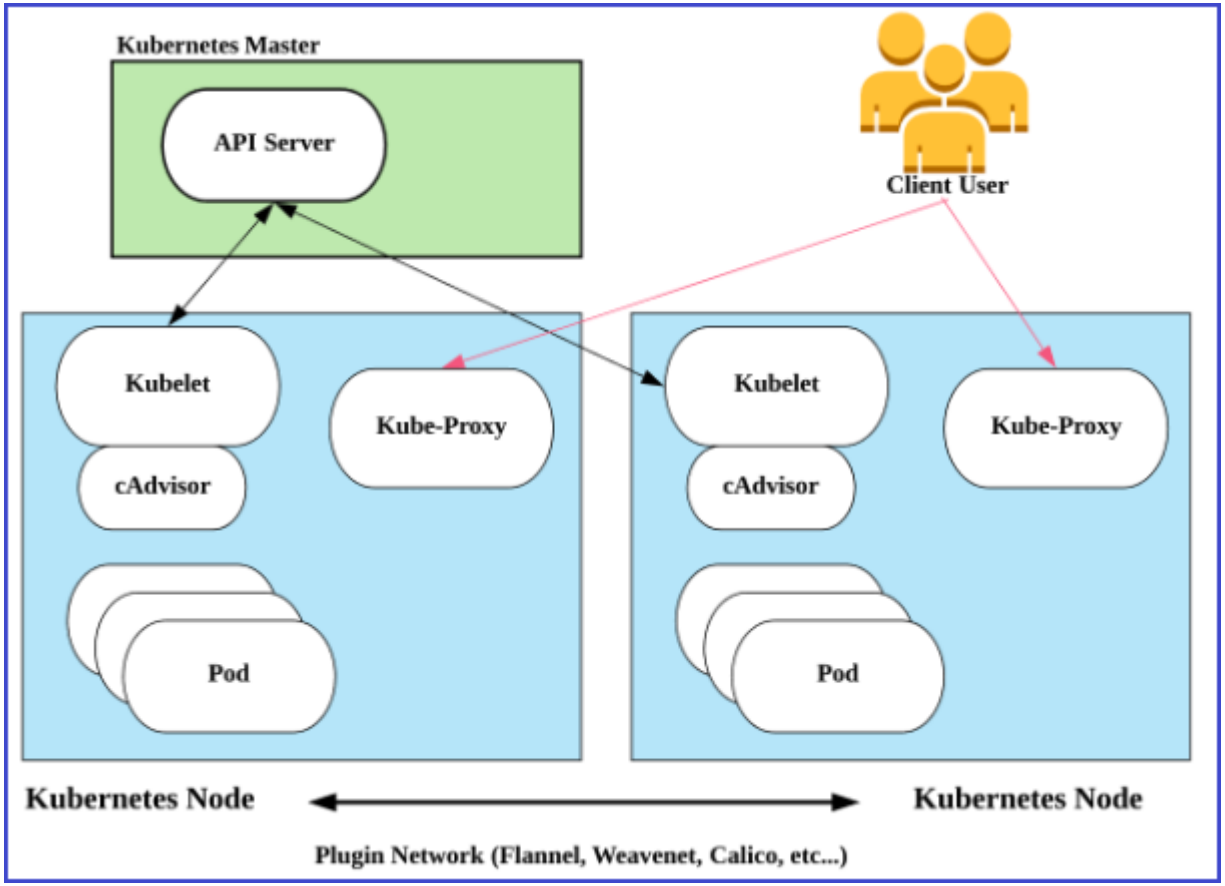
- **Kubernetes 구조**
  - Master(=Master node)
    - API server
      - **K8s에서 가장 중요한 component 이다**
      - K8s 내부의 모든 동작은 HTTP/HTTPS REST API를 통해서 호출된다
      - 각각의 K8s component 가 서로 직접 연결된 형태가 아니고 모든 component 는 API Server를 경유해서 통신한다
      - component 들은 API Server를 watch하고 있다가 자신과 관련된 부분의 변경사항이 있으면 변경사항을 업데이트하는 형태로 구현되어 있다
      - 앞 단에 Pluggable하게 인증시스템을 붙일 수 있다
      - kubectl (Kubernetes command-line tool)이나 GUI Dashboard에서 명령을 실행할 때도 REST API를 통해 Master의 API Server에 API를 호출하는 형태로 수행된다
      - **Cluster 정보를 저장하는 etcd**에도 각 component 가 직접 접속하지 않고, API Server가 etcd의 watch와 유사한 기능을 구현하고 component 들은 API Server를 watching하게 된다

Kubernetes 설치

- **Kubernetes 구조**
  - Master(=Master node)
    - Controller Manager
      - K8s에서 ReplicaSet, Deployment등의 Controller는 Pod의 복제/배포를 변경하는 명령을 수행하고 Pod의 상태를 관리한다
      - Controller Manager는 이 Controller들을 데몬 형태로 포함하고 있다
      - Controller Manager는 Pod의 상태를 항상 기대하는 상태로 유지한다
    - Scheduler
      - 어떤 Pod을 어떤 Node에서 실행할지 결정한다.
      - 각 Node의 자원 사용 상황, 노드당 클러스터 분배 비율을 기반으로 Container를 배치한다.
      - Node에 배치된 Pod은 각 Node의 Kubelet Component에 의해 Container로 생성된다.
    - etcd
      - 분산 Key/Value 저장소
      - Cluster 설정 및 현재 상태(클러스터의 각 노드, 노드에서 동작중인 Pod 등의 모든 상태)를 저장한다.
      - Service Discovery에서 사용하는 SkyDNS의 데이터를 저장한다.

Kubernetes 설치

- **Kubernetes 구조**
  - Node (=minion)
    - Node는 다음과 같은 K8s component로 구성된다
      - Kubelet, cAdvisor, Kube-Proxy, Pod



Kubernetes 설치

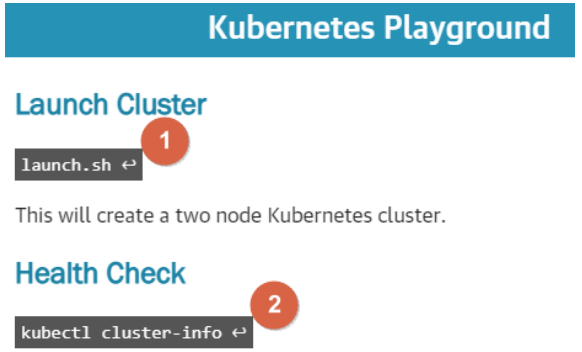
- **Kubernetes 구조**
  - Node(=Minion, Worker node, Slave node)
    - Kubelet
      - Node마다 한 개씩 떠있는 **Node(worker) Agent**라고 할 수 있다.
      - **Service, Pod을 실행/중지하고 desired 상태를 유지시킨다**.
      - 주기적으로 **API Server에 Node의 상태를 Check & Report access** 한다
    - cAdvisor
      - 동작중인 **Container의 리소스 사용량과 성능을 모니터링** 한다
      - 모니터링한 결과는 Kubelet에 의해 Master의 API Server로 전달된다
    - Kube-Proxy
      - **"네트워크 프록시 + 로드밸런서"**의 역할을 한다
      - **외부의 요청을** 분산 되어있는 **Pod으로 전달한다**
      - 컨테이너간의 통신을 위해서 iptables rules를 변경한다
    - Pod
      - K8s의 가장 작은 배포 단위는 Pod이다(Container가 아니다)

Kubernetes 설치-Free Cloud

- **Kubernetes Playground 소개**
  - http://katacoda.com
  - Online에서 k8s 연습할 수 있는 곳
  - OReilly에서 운영하며 로그인하지 않고도 사용 가능
  - **미리 k8s가 설치 및 구성되어 있으며**, 1대의 master와 node만 사용
  - 실습하기
    - https://www.katacoda.com/courses/kubernetes/playground

START SCENARIO

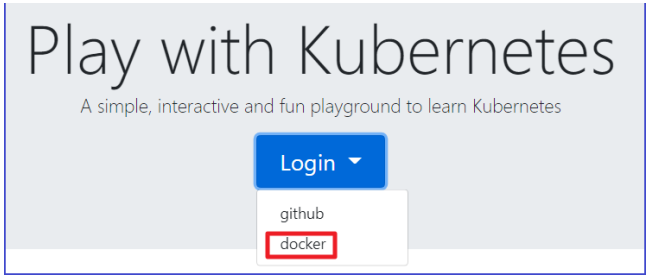
    - 사전 점검하기



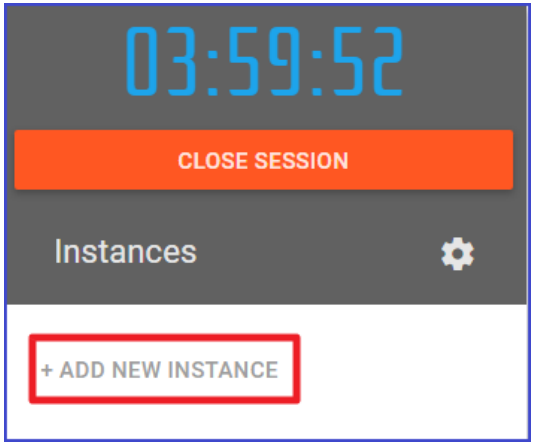
    - docker version
    - kubectl version -o yaml
    - kubectl get node -o wide
    - kubectl apply -f http://down.cloudshell.kr/k8s/lab/nginx-app.yaml
    - kubectl get pod -o wide

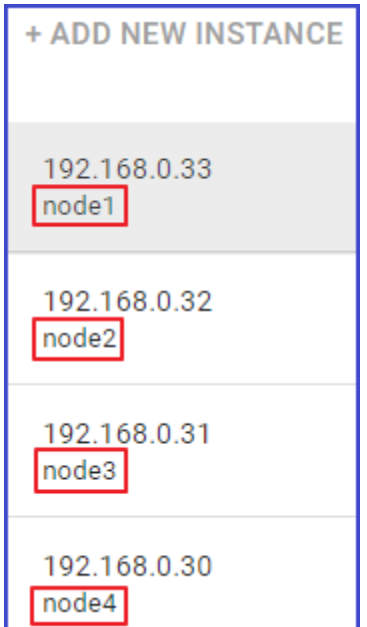
Kubernetes 설치-Free Cloud

- **Play with Kubernetes 소개**
  - http://labs.play-with-k8s.com
  - Docker Inc. 가 제공하는 무료 싸이트
    - Play with Kubernetes is a labs site **provided by Docker** and created by Tutorius.
  - docker나 github 계정으로 로그인한다
  - 한 번 접속했을 때 사용 시간은 4시간



  - node를 4개 추가한다
    - master: node1
    - worker node: node2, node3, node4



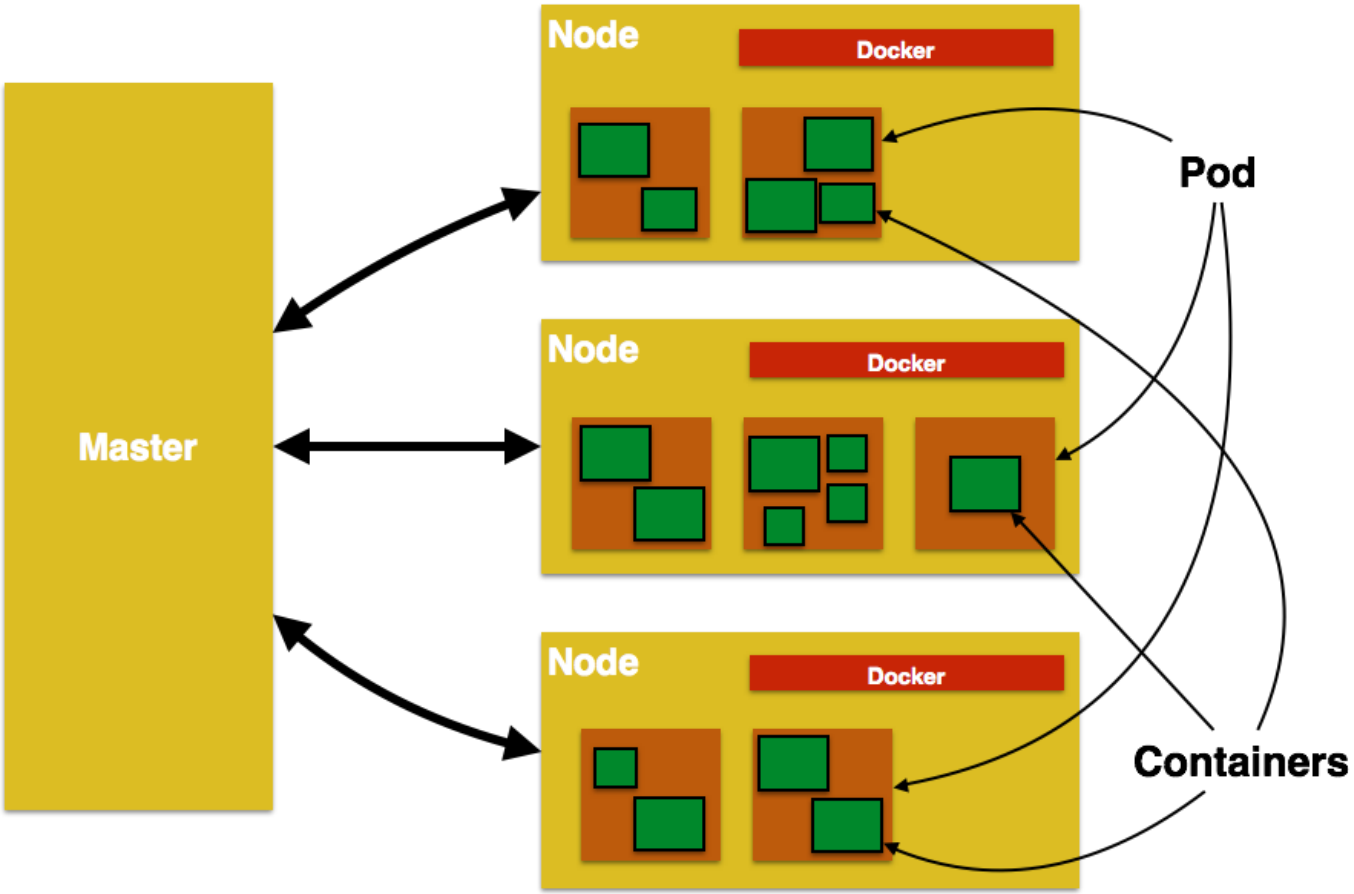


Kubernetes 설치-Free Cloud

- **Play with Kubernetes 소개**
  - 수동으로 k8s를 설치 및 구성한다
  - node1을 클릭하여 화면에 나오는 내용을 **"ctrl + insert"**로 복사하여 **"ctrl + v"**로 붙여넣기 한다
    - Initializes cluster master node
    - Initialize cluster networking
    - Create an nginx deployment
  - 1번의 Cluster를 구성한 후에 나타나는 Cluster에 가입하는 문구를 복사하여 자신의 컴퓨터의 메모자 프로그램에 붙여 넣는다
    - kubeadm join 192.168.0.8:6443 --token ~~~~~
    - 이것을 이용하여 node2, node3, node4가 Cluster에 참가하게 된다
  - 설치와 구성이 제대로 되었는지 확인하기
    - kubectl get node -o wide
    - kubectl get pod --all-namespaces
    - kubectl get pod -n default -o wide

Kubernetes 설치

- 설치 구성도
  - **Master: 1대**
  - **Node: 3대**

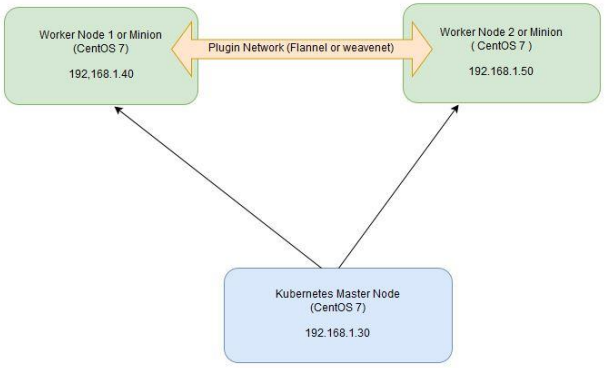


Kubernetes 설치

- **설치 개요**
  - Master에 설치될 component
    - **API Server**
      - It provides kubernetes API using Jason / Yaml over http, states of API objects are stored in etcd
    - **Scheduler**
      - It is a program on master node which performs the scheduling tasks like launching containers in worker nodes based on resource availability
    - **Controller Manager**
      - Main Job of Controller manager is to monitor replication controllers and create pods to maintain desired state.
    - **etcd**
      - It is a Key value pair data base. It stores configuration data of cluster and cluster state.
    - **Kubectl utility**
      - It is a command line utility which connects to API Server on port 6443.
      - It is used by administrators to create pods, services etc.

Kubernetes 설치

- **설치 개요**
  - Node에 설치될 component
    - **Kubelet**
      - It is an agent which runs on every worker node, it connects to docker and takes care of creating, starting, deleting containers.
    - **Kube-Proxy**
      - It routes the traffic to appropriate containers based on ip address and port number of the incoming request.
      - In other words we can say it is used for port translation.
    - **Pod**
      - Pod can be defined as a multi-tier or group of containers that are deployed on a single worker node or docker host.



Kubernetes 설치

- **Add-on 요소**
  - Pod Network (=CNI)
    - weave-net, kube-router, calico, flannel
  - DNS
    - coreDNS (** 내장됨)
  - Dashboard
  - Container Resource Monitoring
    - cAdvisor
  - Cluster Logging
    - DataDog
    - ELK(ElasticSearch, Logstash, Kibana)
    - EFK(ElasticSearch, Fludentd, Kibana)

Kubernetes 설치

- **설치 사전 준비**
  - **/etc/hosts 파일 수정**
    - **vi /etc/hosts**
      - 10.0.2.4 master
      - 10.0.2.5 node1
      - 10.0.2.6 node2
      - 10.0.2.7 node3
    - **scp** /etc/hosts node1:/etc/hosts
    - **scp** /etc/hosts node2:/etc/hosts
    - **scp** /etc/hosts node3:/etc/hosts

Kubernetes 설치

- **설치 사전 준비**
  - **SWAP 기능 비활성화하기**
    - Pod를 할당하고 제어하는 kubelet은 swap 상황을 처리하도록 설계되지 않았다
    - 이유는 kubernetes에서 가장 기본이 되는 **Pod의 컨셉 자체가 필요한 Resource 만큼만 호스트 자원에서 할당 받아 사용한다는 구조**이기 때문이다.
    - 따라서 kubernetes 개발팀은 Memory Swap을 고려하지 않고 설계했기 때문에 클러스터 노드로 사용할 서버 머신들은 모두 Swap을 비활성화 해줘야 한다
    - **swapoff -a**
    - **vi /etc/fstab**
      - 여기서 #을 입력하여 swap 부분을 주석처리 한다

Kubernetes 설치

- **설치 사전 준비**
  - **br_netfilter Kernel Module 기능 켜기**
    - Enable this kernel module so that the packets traversing the bridge are processed by iptables for filtering and for port forwarding, and the kubernetes pods across the cluster can communicate with each other.
    - **modprobe br_netfilter**
    - **echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables**
    - **docker info**
      - 여기서 WARNING 메시지를 없애기 위해 다음과 같이 한다
      - WARNING 메시지가 있으면 K8S에 가입을 할 수가 없다
    - **vi /etc/sysctl.conf**
      - net.bridge.bridge-nf-call-iptables = 1
      - net.bridge.bridge-nf-call-ip6tables = 1
  - **reboot**

- **방화벽(firewalld)과 Selinux는 이미 실행되지 못하도록 설정함**

Kubernetes 설치

- **설치 사전 준비**
  - **Kubernetes Repository 구성하기**
    - Kubernetes packages are not available in the default CentOS 7 & RHEL 7 repositories
    - vi /etc/yum.repos.d/kubernetes.repo

```
[root@centos1 yum.repos.d]# cat kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
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
```

    - 또는 repository 파일 다운로드하기
      - **cd /etc/yum.repos.d (##중요)**
      - **wget http://down.cloudshell.kr/k8s/kubernetes.repo**
      - **yum repolist -y**

```
repo id                          repo name
base/7/x86_64                    CentOS-7 - Base
docker-ce-stable/x86_64          Docker CE Stable - x86_64
epel/x86_64                      Extra Packages for Enterprise Linux 7 - x86_64
extras/7/x86_64                  CentOS-7 - Extras
kubernetes                       Kubernetes
```

Kubernetes 설치

- **설치 사전 준비**
  - **docker 설치 및 시작하기**
    - **curl -sSL http://get.docker.com | bash**
    - **systemctl start docker && systemctl enable docker**
    - **docker version**
    - **docker info | grep -i cgroup**

```
Cgroup Driver: cgroupfs
```

    - **Cgroup Driver와 관련된 사항**
      - Linux에는 namespace와 cgroup이란 것이 있으며, 이 덕분에 docker와 kubernetes가 가능하다
      - namespace로 시스템에 독립 뷰 제공이 가능하다
      - Cgroup으로 프로세스의 가용 리소스(cpu, 메모리,네트워크 등)를 제한할 수 있다
      - 최근에 Docker와 kubernetes가 동일한 cgroup driver을 가져야 하는 것으로 변경되었다
      - **Kubernetes**는 cgroup을 **systemd**를 사용하고 있는데, docker가 cgroupfs를 사용하고 있기 때문에, 미리 **docker를 systemd로 변경해주어야 한다**

Kubernetes 설치

- **설치 사전 준비**
  - **docker의 cgroup driver를 cgroupfs에서 systemd로 변경하기**
    - **kubernetes를 운영하기 위해서는 docker의 cgroup을 systemd로 변경해야 한다**
    - vi **/usr/lib/systemd/system/docker.service**
      - **ExecStart=/usr/bin/dockerd** 바로 뒤에 --exec-opt native.cgroupdriver=systemd 를 **삽입**한다
    - Docker 재시작하기
      - **systemctl daemon-reload**
        (##모든 unit file을 reload하기. systemd 구성파일을 reload하는 것은 아님)
      - **systemctl restart docker**
    - 변경된 cgroup 확인하기
      - **docker info | grep -i cgroup** (##systemd로 변경됨)

```
Cgroup Driver: systemd
```

Kubernetes 설치

- **kubernetes 설치 및 시작하기**

##특정한 이전 버전(1.21.1)의 K8S 설치하기
yum -v list kubelet --show-duplicates
yum install -y **kubelet-1.21.1-0 kubeadm-1.21.1-0 kubectl-1.21.1-0**

systemctl **daemon-reload**
systemctl **enable kubelet --now**

Kubernetes 설치

- **K8S Clustering 구성하기-Master에서만 작업**

##특정한 이전 버전(1.21.1)의 K8S Clustering 구성하기
kubeadm init --kubernetes-version=v1.21.1 --apiserver-advertise-address=**192.168.56.121**
--pod-network-cidr 10.244.0.0/16

Master의 IP 주소

  - ## 여기서 **kubeadm join** 구문을 notepad에 복사해 놓는다(매우 중요)
  - docker images ; docker ps
  - docker container inspect $(docker ps -q) | grep -i networkmode
- **kubernetes clustering를 사용할 수 있도록 구성하기**
  - k8s cluster에 접속하기 위해 KUBECONFIG 파일을 준비한다
  - **mkdir** -p $HOME/.kube
    - (##admin.conf 파일을 복사할 Directory 생성)
  - **sudo cp -i /etc/kubernetes/admin.conf** $HOME/.kube/config
    - (##admin.conf 파일을 config 파일 이름으로 복사)
  - **sudo chown $(id -u):$(id -g)** $HOME/.kube/config
    - (##config 파일에 엑세스하도록 permission 변경하기)

Kubernetes 설치

- **Kubernetes 버전 정보 확인하기**
  - **kubectl version --short**
  - **kubectl version -o yaml**
  - **kubeadm version -o yaml**
  - **kubelet --version**

- **Cluster 구성정보(ConfigMap) 확인하기-Master에서만 작업**
  - **kubectl get namespace** (##namespace 확인하기)
  - **kubectl get cm -n kube-system** (##kube-system의 ConfigMap들 확인)
  - **kubectl get cm -n kube-system kubeadm-config -o yaml (#최신방식)**

```
networking:
  dnsDomain: cluster.local
  podSubnet: 10.244.0.0/16
  serviceSubnet: 10.96.0.0/12
```

```
apiEndpoints:
  master:
    advertiseAddress: 192.168.56.121
    bindPort: 6443
```

master의 advertiseAddress가 정확한지 확인한다

  - **kubeadm config view (#이전 방식)**

Kubernetes 설치

- **Clustering 상태 및 Pod 상태 확인하기-Master에서만 작업**
  - **kubectl get nodes (##master가 NotReady 상태임)**
  - **kubectl get ns (##ns=namespace)**
  - **kubectl get pods --all-namespaces**
    - ##cluster 상태가 Not Ready이고 **coredns 상태가 Pending이다**. 이렇게 되면 각 Node들에서 실행중인 **Pod들간에 통신이 안된다**
    - **Pod Network를 생성하여 이 문제를 해결할 수 있다**
    - POD network는 node간에 통신을 하게 하는 overlay network이다
    - **--all-namespaces**와 같은 것: -A
  - **kubectl get pods -n kube-system**
    - kube-system에 pod들이 실행중이다
  - **kubectl get pods -n default**
    - 현재는 pod이 **실행되고 있지 않는다**

Kubernetes 설치

- **Pod Network 구성하기-Master에서만 작업**
  - **CNI**(Container Network Interface) **plugin**인 calico, flannel 등의 Pod Network를 구성할 수 있다
    - 참고: https://kubernetes.io/ko/docs/concepts/cluster-administration/addons/
    - Weave-Net은 2022년9월 30일에 서비스 종료함
  - Pod간의 통신을 위해 CNI plugin인 중에서 하나 만 설치한다
  - **kubectl get pods -A**
    - **Calico 설치**(Virtual box에서 **NAT와 Host-Only의 2개** 네트워크 사용시 추천)
      - **kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml**
      - 50개 노드 이하에서 사용
    - **flannel 설치** (Virtual box에서 **Bridged 네트워크** 사용시 추천/설정하는 시간이 빠름)
      - **kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml**
  - **kubectl get pods -A**
    - weave-net-xxxxx이라는 Pod이 master에 설치됨. 그리고 coredns는 드디어 running

Kubernetes 설치

- **Pod Network 구성하기-Master에서만 작업**
  - 30초 정도 기다린 후 다음 명령어를 실행한다
    - **kubectl get nodes** (##master가 Ready 상태가 됨)
    - **kubectl get pods -A**
    - **## 모두 정상으로 되었다. 이제서야 Kubernetes cluster의 master 초기화 및 구성이 완료되었다**

NAMESPACE	NAME	READY	STATUS
kube-flannel	kube-flannel-ds-xdb5s	1/1	Running
kube-system	coredns-558bd4d5db-9zlsq	1/1	Running
kube-system	coredns-558bd4d5db-ccltr	1/1	Running
kube-system	etcd-master	1/1	Running
kube-system	kube-apiserver-master	1/1	Running
kube-system	kube-controller-manager-master	1/1	Running
kube-system	kube-proxy-6mt4n	1/1	Running
kube-system	kube-scheduler-master	1/1	Running

    - **Calico가 정상 동작하려면 방화벽에 각각의 포트를 허용해주어야 한다**
      - firewall-cmd --permanent --add-port=179/tcp
      - firewall-cmd --permanent --add-port=4789/udp
      - firewall-cmd --permanent --add-port=5473/tcp
      - firewall-cmd --permanent --add-port=443/tcp
      - firewall-cmd --permanent --add-port=6443/tcp
      - firewall-cmd --permanent --add-port=2379/tcp
      - firewall-cmd --reload

Kubernetes 설치

- **node를 cluster에 추가하기- node에서만 작업하기**
  - 앞에서 notepad에 복사한 내용을 node1에서 붙여넣기 한다
    - **kubeadm join** 192.168.219.120:6443 --token v9bekv.xnunyj3yntpg0h73 --discovery-token-ca-cert-hash sha256:1dd1a00bd2dd4b970aafe5ce6ff93bcb3908e925ee507ddb9ef0e61801051062
    - 실패하면 **이 node에서 "kubeadm reset"**를 한 후 화면에 알려주는 파일을 지운 후에 다시 가입을 해 본다
  - **나머지 node에서도** kubeadm join으로 cluster에 가입한다
  - 30초 정도 기다린 후 **master에서** 다음 명령어를 실행한다
    - **kubectl get nodes**
    - **kubectl get pods -A**

```
[root@master ~]# kubectl get node
NAME     STATUS   ROLES                  AGE   VERSION
master   Ready    control-plane,master   85m   v1.21.1
node1    Ready    <none>                 68s   v1.21.1
node2    Ready    <none>                 52s   v1.21.1
```

  - 각 node에서 다음 명령어를 실행한다
    - **docker images**
    - **docker ps**

Kubernetes 설치

- **k8s 설치하기 - 요약 정리**
  - 설치 사전 준비
    - firewalld 및 selinux 제거
    - docker 설치 및 시작 -> docker의 cgroup을 systemd로 변경
    - swap 기능 제거 -> br_netfilter Kernel Module 기능 켜기
    - Kubernetes Repository 추가
  - 공통 설치(특정 버전)
    - yum install -y kubelet-1.21.1-0 kubeadm-1.21.1-0 kubectl-1.21.1-0
  - Master에만 설치
    - 특정 버전의 Cluster 생성: kubeadm init --kubernetes-version=v1.21.1 --apiserver-advertise-address=192.168.56.121 --pod-network-cidr 10.244.0.0/16
      - kubeadm join 구문을 메모장에 복사해 놓기
      - k8s cluster에 접속하기 위해 KUBECONFIG 파일 복사하기
      - Cluster 구성정보 확인: kubectl get cm -n kube-system kubeadm-config -o yaml
    - CNI 추가: flannel 설치
    - 설치 성공 여부 확인: kubectl get pods -A ; kubectl get nodes
  - Node에만 설치
    - Cluster에 가입하기(메모장에 복사한 것 활용): kubeadm join

Kubernetes 설치 후 작업

- **Tab키를 사용하여 명령어 자동 완성하기-Master에서 작업**
  - 아래 작업은 로그인하는 사용자마다 동일하게 작업해야 한다
    - yum install **bash-completion** -y
    - **source** /usr/share/bash-completion/bash_completion
    - **echo** 'source <(kubectl completion bash)' >> **~/.bashrc**
    - kubectl completion bash >/etc/bash_completion.d/kubectl
    - echo 'complete -F __start_kubectl k' >>~/.bashrc
- **kubectl의 Alias(k) 생성하기-Master에서 작업**
  - 아래 작업은 로그인하는 사용자마다 동일하게 작업해야 한다
    - **echo** 'alias k=kubectl' >> **~/.bashrc**
  - **Tab completion 및 Alias 설정시의 대안(##강추)**
    - **cd ; curl -sSL http://down.cloudshell.kr/down/k8slab.sh | bash**
    - **cd k8s ; sh ./kubelet_alias_bash_completion.sh**

- **k와 tab 완성 테스트하기**
  - **k get no탭**

Kubernetes 설치 후 작업

- **Nginx** Pod을 K8S cluster에 배포하기**-Master에서 작업**
  - K8S 환경에서 실행중인 Pod은 한 개 이상의 container로 구성되며, 그 Pod에 있는 docker container들은 storage와 Network을 공유한다
  - mynginx라는 deployment 생성하기
    - **kubectl create deployment** mynginx **--image=nginx**
    - **kubectl get all -o wide**
      - ##deployment를 생성했는데, Pod, ReplicaSet까지 동시에 생성되었음
  - 생성된 mynginx deployment의 세부 내용 확인하기
    - **kubectl describe deployment** mynginx
  - 생성된 pod의 세부 내용 확인하기
    - **kubectl describe pod** mynginx-5b686ccd46-bv7zh
  - 클러스터 내부 IP로 생성된 pod에 접속하기
    - **kubectl get pods -o wide** (##container가 실행중인 node 확인)
    - **curl** 10.44.0.1 (##pod ip)
    - ##내부에 있는 master나 각 node에서 이렇게 접속해 보기: 성공

Kubernetes 설치 후 작업

- **Nginx** Pod을 K8S cluster에 배포하기**-Master에서 작업**
  - 외부에서 접속하기 위해서는 nodePort라는 서비스가 필요하다. 외부에서 deployment에 접속하도록 **service 노출하기**
    - **kubectl create service nodeport** mynginx **--tcp=8080:80**
      - ##앞의 8080: **Cluster-IP**로 접속할 때 사용되는 포트번호
      - ##뒤의 80: Container의 Port 번호
  - 접속할 IP와 Port 번호 확인하기
    - **kubectl get pods -o wide** (##container가 실행중인 node 확인)
    - **kubectl get svc -o wide** (##node port 번호 확인)

```
[root@master ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP
mynginx      NodePort    10.106.133.5   <none>        8080:30752/TCP
```

  - maste에서 curl로 접속하기
    - **curl 10.106.133.5:8080**
  - container가 node1에서 실행중이면 Windows Desktop에서 Web Browser를 실행하여 다음과 같이 접속해 본다
    - **http://node1-ip:30752**

Kubernetes 설치 후 작업

- **Nginx** Pod을 K8S cluster에 배포하기**-Master에서 작업**
  - **kubectl create deployment myweb --image httpd**
  - **kubectl create svc nodeport myweb --tcp 8888:80**
  - **kubectl get pod -o wide**
  - **kubectl get svc -o wide**

```
[root@master ~]# kubectl get pod -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE     NOMI
mynginx-5b686ccd46-bv7zh   1/1     Running   0          9m31s   10.44.0.1    node1    <nor
myweb-6f795ffc84-z5s56     1/1     Running   0          2m9s    10.36.0.1    node2    <nor
[root@master ~]# kubectl get svc -o wide
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE   SELECTOR
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          10m   <none>
myweb        NodePort    10.107.217.22   <none>        8888:30387/TCP   29s   app=myweb
[root@master ~]# curl 10.36.0.1
<html><body><h1>It works!</h1></body></html>
[root@master ~]# curl 10.107.217.22:8888
<html><body><h1>It works!</h1></body></html>
[root@master ~]# curl node1:30387
<html><body><h1>It works!</h1></body></html>
[root@master ~]#
```

Kubernetes 설치 후 작업

- **Nginx** Pod을 K8S cluster에 배포하기**-Master에서 작업**
  - Node의 IP로 포트로 container에 접속하는 것을 **다시 정리하면…**
    - **kubectl create deployment** mygosmall **--image=jesuswithme/gosmall**
    - **kubectl scale deployment mygosmall --replicas 4**
    - **kubectl get all**
    - **kubectl create service nodeport** mygosmall --tcp=80:80
    - **kubectl get pods -o wide** (##container가 실행중인 node 확인)
    - **kubectl get services -o wide** (##node port 번호 확인)
    - curl http://node-ip:31270

Kubernetes 설치 후 작업

- **Lab에 사용되는 yaml 파일 다운로드하기**
  - **http://down.cloudshell.kr/k8s/lab/**에 모두 저장되어 있다
  - pod-sample.yaml을 다운로드하여 실행하기
    - wget **http://down.cloudshell.kr/k8s/lab/pod/**pod-sample.yaml
    - **kubectl apply -f** pod-sample.yaml
    - **kubectl get pod -o wide**
  - 삭제할 때는 다시 pod-sample.yaml을 사용한다
    - **kubectl delete -f** pod-sample.yaml
    - **kubectl get pod -o wide**

K8S 구성 내용 확인하기

- **Namespace 단위로 구성 정보 확인하기**
  - namespace 단위로 k8s cluster의 **구성 내용** 확인하기(##강추)
    - **kubectl get ns**
      - namespace 종류들 확인하기

```
root@master:~# kubectl get ns
NAME              STATUS   AGE
default           Active   26h
kube-node-lease   Active   26h
kube-public       Active   26h
kube-system       Active   26h
```

    - **kubectl -n kube-system get cm**
      - **kube-system**이라는 namespace의 **configmaps** 확인하기

```
[root@master pod]# kubectl -n kube-system get cm
NAME                                 DATA   AGE
coredns                              1      56m
extension-apiserver-authentication   6      56m
kube-proxy                           2      56m
kube-root-ca.crt                     1      56m
kubeadm-config                       1      56m
kubelet-config-1.22                  1      56m
weave-net                            0      52m
```

K8S 구성 내용 확인하기

- **Namespace 단위로 구성 정보 확인하기**
  - namespace 단위로 k8s cluster의 **구성 내용** 확인하기(##강추)
    - **kubeadm-config의 세부 내용을 확인**
      - **kubeadm-config에는 k8s의 Clustering 정보를 가지고 있다**
      - **kubectl -n kube-system get cm kubeadm-config -o yaml**
      - 또는 간단하게는 **kubeadm config view**

```
  clusterName: kubernetes
  controllerManager: {}
  dns:
    type: CoreDNS
  etcd:
    local:
      dataDir: /var/lib/etcd
  imageRepository: k8s.gcr.io
  kind: ClusterConfiguration
  kubernetesVersion: v1.17.3
  networking:
    dnsDomain: cluster.local
    serviceSubnet: 10.96.0.0/12
  scheduler: {}
ClusterStatus: |
  apiEndpoints:
    master:
      advertiseAddress: 10.0.0.4
```

    - **configmaps들의 각각의 상세한 내용을 꼭 확인해볼 것**
      - **kubectl get cm -n kube-system coredns -o yaml**
      - **kubectl get cm -n kube-system weave-net -o yaml**
      - **kubectl get cm -n kube-system kube-proxy -o yaml**

K8S 구성 내용 확인하기

- **Node에 다운로드된** docker image 확인하기
  - **docker images**

```
root@node2:~# docker images
REPOSITORY                  TAG
k8s.gcr.io/kube-proxy       v1.15.0
weaveworks/weave-kube       2.5.2
weaveworks/weave-npc        2.5.2
k8s.gcr.io/pause            3.1
```

  - ##node에선 4개의 image만 다운로드
  - ## master에서는 9개가 image가 다운로드(처음에는 7개 였고, Network Add-On을 설치한 후에 2개가 추가됨)

- Node에 실행중인 Container 확인하기
  - **docker info | more**
  - **docker ps**

Node 추가하기

- **Worker node 추가하기**
  - Kubernetes 운영 하던 중 **Node를 추가 해야 할 상황이 발생할 수 있다**
  - Worker Node나 Master를 추가하기 위해서는 kubeadm join 구문을 사용해야 한다
  - 그런데 kubeadm join을 하기 위해서는 token을 알아야 하는데, token 유효 기간이 하루여서 아예 새롭게 만들어서 사용하는 것이 낫다
  - 현재 token 현황 알아보기 (##현 Master node에서 작업한다)
    - **kubeadm token list**
  - 토큰을 만들 때 사용되는 인증서 키 확인하기
    - **kubeadm init phase upload-certs --upload-certs**
    - ##인증서 키를 복사해 둔다
  - 확인된 인증서 키를 사용하여 새로운 토큰을 만들면서 kubeadm join 생성하기
    - **kubeadm token create --certificate-key** 51ef3df6c5dffd6fcaaa40b164dd6cb67eab597f5d2c64580127cfbc34442cdc **--print-join-command**

Node 추가하기

- **Worker node 추가하기**
  - 여기서 나온 결과가 다음과 같다
    - kubeadm join 192.168.1.99:6443 --token 1gv1mz.b60yevjywap3bp1u --discovery-token-ca-cert-hash sha256:5a1e04a64ce47a39ed59ab92d1236c5b3ac3b4f834982545a75c585f3b87d96a --control-plane --certificate-key f990b7bf4b1bd139b5e6453166da0c6ce656d38d3c7a201d1825b68a13842050
    - 여기서 제일 뒷 부분인 --control-plane --certificate-key 이후를 모두 삭제하여 새로 추가할 worker node에 붙여 넣기하여 사용하면 된다
  - 위의 부분을 복사하여 **추가할 Worker Node에서** 붙여 넣는다
    - **kubeadm join 192.168.1.99:6443** --token 1gv1mz.b60yevjywap3bp1u --discovery-token-ca-cert-hash sha256:5a1e04a64ce47a39ed59ab92d1236c5b3ac3b4f834982545a75c585f3b87d96a
  - 현 Master node에서 추가된 것인지를 확인한다
    - **kubectl get nodes**
  - 혹시 Join에 실패하면 아래 명령어를 실행한 후 다시 가입한다
    - modprobe br_netfilter
    - echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
    - echo '1' > /proc/sys/net/ipv4/ip_forward

Node 추가하기

- **Master node 추가하기**
  - 새롭게 추가할 Master Node에서 위의 부분을 복사하여 붙여 넣는다
    - **kubeadm join 192.168.1.99:6443** --token 1gv1mz.b60yevjywap3bp1u --discovery-token-ca-cert-hash sha256:5a1e04a64ce47a39ed59ab92d1236c5b3ac3b4f834982545a75c585f3b87d96a **--apiserver-advertise-address 192.168.1.100**
    - worker node와 다른 것은 새로운 maste의 IP(192.168.1.100)를 추가하여 작업해야 한다
  - 새롭게 추가한 master(192.168.1.100)에서 다음을 작업한다
    - mkdir ~/.kube
    - **scp root@172.16.16.99:/etc/kubernetes/admin.conf ~/.kube/config**
  - 첫 번째 master에서 다음과 같이 하여 확인한다
    - **kubectl cluster-info**
    - **kubectl get nodes**
  - 새롭게 추가한 master(192.168.1.100)에서 다음을 작업한다
    - **kubectl get nodes**
    - **kubectl create deploy mynginx --image=nginx**

Lab 파일 다운로드하기

- **Lab 파일 다운로드하기**
  - master에서 다음과 같이 작업한다
    - **cd**
    - **curl -sSL http://down.cloudshell.kr/down/k8slab.sh | bash**
    - **ls -l**
    - **cd k8s**
    - **ls -l**

쉬어가는 코너

- worker node의 kubelet 대몬이 중지된 경우
- worker node의 docker 대몬이 중지된 경우
- master의 docker 대몬이 중지된 경우
- 특정한 버전의 Kubernetes 설치하기
- Cluster에 속하지 않는 컴퓨터에 kubectl을 설치하여 k8s 관리하기

쉬어가는 코너

- Pod Network 대역 지정하기
- 설치된 CNI 확인하기
- firewalld를 사용할 때 Inbound Port 허용하기
- Time Sync하기
- 시스템이 커널 메시지를 발생시키면서 정지될 때
- micro k8s 설치하기

worker node의 kubelet 대몬이 중지된 경우

- **worker node의 kubelet이 중지된 경우**
  - node2에서 kubelet 중지
    - **ssh node2**
    - **systemctl stop kubelet**
    - **exit**
  - master에서 3개의 Pod 생성하기
    - **kubectl create deployment mydeploy --image=jesuswithme/verify-node --replicas=3**
    - **kubectl get pod -o wide**

```
[root@master ~]# kubectl get pod -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE
GATES
mydeploy-7d94f7cfb8-4blkj   1/1     Running   0          10s   10.44.0.1    node1
mydeploy-7d94f7cfb8-jg2wb   1/1     Running   0          10s   10.44.0.4    node1
mydeploy-7d94f7cfb8-qhwpc   0/1     Pending   0          10s   <none>       node2
```

worker node의 kubelet 대몬이 중지된 경우

- **worker node의 kubelet이 중지된 경우**
  - node2에 kubelet이 중지되니 master의 scheduler가 pod을 할당하지 못하고 있다.
  - 즉, kubectl에서 3개 Pod 생성 명령 ➔ API server ➔ scheduler에서 각 노드의 부하를 보고 적당한 node에 Pod 생성 명령 ➔ API server ➔ node1의 kubelet에서는 docker에서 생성하지만 node2에서는 kubelet과 통신이 안되어 docker까지 가지도 못한채 Pending으로 남아 있음
  - 시간이 더 지나면 API Server가 node2의 kubelet과 통신이 안되는 것을 scheduler에게 알려주어 scheduler는 다시 API Server에게 다른 node(node1)에게 Pod을 할당하라고 명령하게 된다
  - 그 결과 생성되지 못한 Pod을 node1에 생성하게 된다
  - 다시 node2의 kubelet을 시작한다
    - **ssh node2**
    - **systemctl start kubectl**
    - **exit**

worker node의 docker 대몬이 중지된 경우

- **worker node의 docker가 중지된 경우**
  - node1에서 docker 중지
    - **ssh node1**
    - **systemctl stop docker.socket**
    - **systemctl stop docker.service**
    - **exit**
  - master에서 3개의 Pod 생성하기
    - **kubectl create deployment ex-deploy --image=jesuswithme/verify-node --replicas=3**
    - **kubectl get pod -o wide**

```
[root@master ~]# kubectl get pod -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP           NODE
SS GATES
ex-deploy-5ccb9dc7db-9dfh4    1/1     Running   0          13s   10.36.0.2    node2
ex-deploy-5ccb9dc7db-kdjgj    1/1     Running   0          13s   10.36.0.3    node2
ex-deploy-5ccb9dc7db-l8fwf    1/1     Running   0          13s   10.36.0.1    node2
```

    - node1에는 아예 Pod이 생성되지 않는다. 왜냐하면 node1의 kubelet이 docker가 중지되었다는 것을 API Server에 알려주었기 때문에 아예 node1에는 Pod의 생성을 배정하지 않는 것이다

master의 docker 대몬이 중지된 경우

- **worker node의 docker가 중지된 경우**
  - 먼저 master의 **kubelet 대몬** 중지하기
    - **systemctl stop kubelet**
    - **kubectl create deployment sample-deploy --image=jesuswithme/verify-node --replicas=3**
    - **kubectl get pod -o wide**

```
sample-deploy-d7d5957bf-6g8q2   1/1     Running   0          96s     10.36.0.14   node2
ne>
sample-deploy-d7d5957bf-g4l54   1/1     Running   0          96s     10.44.0.1    node1
ne>
sample-deploy-d7d5957bf-jprmg   1/1     Running   0          96s     10.36.0.13   node2
```

    - **아무 문제 없이** Pod이 잘 생성되어 운영된다
    - **systemctl start kubelet**

master의 docker 대몬이 중지된 경우

- **worker node의 docker가 중지된 경우**
  - master의 **docker 대몬** 중지하기
    - **systemctl stop docker.socket**
    - **systemctl stop docker.service**
    - **kubectl get pod -o wide**

```
[root@master ~]# kubectl get pod -o wide
The connection to the server 172.30.1.14:6443 was refused
```

    - 이렇게 master의 API server에 접속이 되지 않는다
    - 이것은 master의 docker가 kubernetes의 API Server를 컨트롤하기 때문이다
    - **kubectl create deployment yourdeploy --image=jesuswithme/verify-node --replicas=3**

```
[root@master ~]# kubectl create deployment yourdeploy --image=jesuswithme/verify-node --replicas=3
error: failed to create deployment: Post "https://172.30.1.14:6443/apis/apps/v1/namespaces/default/deployments
?fieldManager=kubectl-create": dial tcp 172.30.1.14:6443: connect: connection refused
```

    - 결론적으로 master의 docker가 죽으면 큰일이다. 그래서 master를 잘 보호해야 한다
    - **systemctl start docker.socket**
    - **systemctl start docker.service**

특정한 버전의 Kubernetes 설치하기

- **특정한 버전의 Kubernetes 설치하기**
  - 설치 가능한 Kubernetes 목록 확인하기
    - **yum list kubelet --showduplicates**
  - 특정한 버전(1.21.1) 설치하기
    - **yum install -y kubelet-1.21.1-0 kubeadm-1.21.1-0 kubectl-1.21.1-0**
  - 특정한 버전(1.21.1) Clustering 설치하기
    - **kubeadm config images pull**
      - 불안정한 네트워크 문제로 이미지 다운로드가 잘 되지 않을 때는 이 명령을 먼저 실행하여 7개 정도의 필요한 이미지를 먼저 다운로드를 한 후에 다음 명령어로 Clustering을 구성하면 된다
    - **kubeadm init --kubernetes-version=v1.21.1**

  - 설치 가능한 docker version 확인하기
    - **yum list docker-ce --showduplicates**
  - 특정 docker version 설치하기
    - **yum install docker-ce-19.03.0**

Cluster에 속하지 않는 컴퓨터에 kubectl을 설치하여 k8s 관리하기

- **Cluster에 속하지 않는 컴퓨터에 kubectl을 설치하여 k8s 관리하기**
  - master의 API-Server component에 kubectl 명령을 보내면 API-Server가 그 명령을 대신 처리하게 된다
  - 그러므로 굳이 Cluster의 master가 아니어도 **kubectl 명령어가 설치된 아무 컴퓨터에서도 k8s를 관리**할 수 있다는 것이다
  - docker가 설치되지 않아도 된다
  - 인증과 권한만 통과되면 아무 컴퓨터에서도 kubectl 명령어로 k8s를 관리할 수 있다
  - 컴퓨터를 하나 더 준비하여 k8s 클러스터를 관리해 본다. 설치할 내용은 다음과 같다
    - kubectl 패키지
    - 인증서 파일(master의 IP 주소와 인증서 포함) 복사하기

Cluster에 속하지 않는 컴퓨터에 kubectl을 설치하여 k8s 관리하기

- **k8s 클러스터를 관리할 컴퓨터 하나 더 만들기**
  - kubectl 설치하기
    - **cd /etc/yum.repos.d**
    - **wget http://down.cloudshell.kr/k8s/kubernetes.repo**
    - **yum install -y kubectl-1.21.1-0**
  - root 계정에 관한 인증서 복사해 오기
    - **mkdir -p $HOME/.kube**
    - **scp master:/etc/kubernetes/admin.conf $HOME/.kube/config**
    - **chown $(id -u):$(id -g) $HOME/.kube/config**
    - **kubectl get nodes**
  - kubectl 명령어 실행하기
    - **kubectl get nodes**

```
[root@outofcluster ~]# hostname
outofcluster
[root@outofcluster ~]# kubectl get nodes
NAME      STATUS     ROLES                  AGE     VERSION
master    Ready      control-plane,master   7d16h   v1.21.1
node1     Ready      <none>                 7d16h   v1.21.1
node2     NotReady   <none>                 7d16h   v1.21.1
node3     NotReady   <none>                 7d16h   v1.21.1
[root@outofcluster ~]# kubectl get pod
```

Pod Network 대역 지정하기

- **Pod Network 대역 지정하기**
  - 클러스터를 생성할 때 Pod Network 대역을 지정할 수 있다
    - **kubeadm init --pod-network-cidr=10.10.0.0/16**
  - **flannel CNI**를 사용할 때는 --pod-network-cidr 옵션을 **필수적으로** 사용해야 하며, **IP 대역도 고정**(10.244.0.0/16)되어 있다
    - **kubeadm init --pod-network-cidr=10.244.0.0/16**

설치된 CNI 확인하기

- **설치한 CNI 확인기**
  - **ls /etc/cni/net.d**

```
[root@master ~]# ls /etc/cni/net.d
10-flannel.conflist
```

    - 설치 된 CNI의 **설정 파일**이 존재한다
    - 이전에 적용되었던 CNI 파일도 그대로 남아있어서 현재 사용 중인 CNI를 정확하게 확인하기에 불충분하다
  - **ls -l /opt/cni/bin**
    - CNI의 **배포 패키지**를 확인할 수 있다
    - 이전에 적용되었던 CNI 파일도 그대로 남아있어서 현재 사용 중인 CNI를 정확하게 확인하기에 불충분하다

firewalld를 사용할 때 Inbound Port 허용하기

- **Firewalld를 사용할 때 Master와 각 Node에서 Port 열기**
  - **Master**

Protocol	Driection	Port Range	Purpose	Used By
TCP	Inbound	6443	Kubernetes API Server	All
TCP	Inbound	2378,2380	etcd Server Client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control Plane
TCP	Inbound	10251	Kube-scheduler	Self
TCP	Inbound	10252	Kube-controller-manager	Self

  - **각 Worker Node**

Protocol	Driection	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control Plane
TCP	Inbound	30000-32767	NodePort Service	All

    - **firewall-cmd --permanent --add-port=6443/tcp**
    - **firewall-cmd --permanent --add-port=30000-32767/tcp**
    - **firewall-cmd --list-all**
    - **firewall-cmd --reload**

Time Sync하기

- **kubeadm join시 master와 시간이 달라서 가입이 실패한 경우**
  - **master와 node간에 시간이 동기화되어야만 가입이 진행된다**
  - **master와 node에서 동일하게 다음과 같이 작업하여 시간을 동기화한다**
    - **보통 time-zone을 동일하게 하면 시간이 같아지는데, 그렇게 해도 안되면 아래와 같이 하면 해결된다**
    - **yum install chrony**
    - **timedatectl set-ntp yes**
    - **systemctl restart chronyd**
    - **timedatectl set-ntp true**
    - **systemctl status chronyd**

시스템이 커널 메시지를 발생시키면서 정지될 때

- **CentOS 7이 갑자기 멈춘 것 해결**
  - **갑자기 아래와 같은 메시지가 나오면서 System이 중지되는 현상이 발생한다**
    - Message from syslogd@master at Dec 8 23:11:00 ...
      kernel:NMI watchdog: BUG: soft lockup - CPU#1 stuck for 34s! [containerd-shim:3250]
  - 이럴 경우는 아래와 같이 해결한다
    - cat /proc/sys/kernel/watchdog_thresh (##10인지 확인)
    - echo "kernel.watchdog_thresh = 20" >> /etc/sysctl.conf
    - sysctl -p

micro k8s 설치하기

- **micro k8s란?**
  - Microk8s는 개발자, 클라우드, 클러스터, 워크스테이션, Edge 및 IoT를 위한 고가용성, 저 운영성, 생산 준비가 된 kubernetes이다
  - MicroK8은 Linux를 염두에 두고 만들어졌으며 VM이 필요하지 않다
  - 간단하게 snapd 패키지를 설치하여 snap 명령으로 설치할 수 있다
  - MicroK8s에는 VM이 필요하지 않기 때문에 제공된 시스템에서 애플리케이션을 실행하기 위해 더 많은 리소스를 사용할 수 있으므로 Edge 배포에 이상적이다
  - Raspberry Pi에서 Kubernetes를 설정하는 데 사용할 수도 있다

micro k8s 설치하기

- **Ubuntu에 micro k8s 설치하기(##권장)**
  - micro k8s 설치
    - sudo -i
    - snap install microk8s --classic --channel=1.25
    - usermod -aG microk8s $USER
    - chown -f -R $USER ~/.kube
    - su - $USER
  - micro k8s 상태 확인
    - microk8s status --wait-ready
    - microk8s enable dns ingress storage
    - microk8s.kubectl get ns

여기까지 진행하는 대신, 아래 명령어를 실행해도 된다
curl http://down.cloudshell.kr/k8s/install-microk8s.sh | bash

  - microk8s.kubectl의 Alias 만들기
    - echo 'source <(microk8s.kubectl completion bash)' >> ~/.bashrc
    - echo 'alias k=microk8s.kubectl' >> ~/.bashrc
    - echo 'complete -o default -F __start_kubectl k' >> ~/.bashrc
    - exec bash
  - micro k8s 관리하기
    - k get ns
    - k get node -o wide
    - k get pod -A

여기까지 진행하는 대신, 아래 명령어를 실행해도 된다
curl http://down.cloudshell.kr/k8s/alias-k-tab.sh | bash

micro k8s 설치하기

- **CentOS 7에 micro k8s 설치하기**
  - snapd 설치하기
    - sudo -i
    - yum install epel-release
    - yum install snapd -y
    - systemctl enable snapd.socket --now
    - ln -s /var/lib/snapd/snap /snap
  - micro k8s 설치
    - su - root (##중요)
    - snap install microk8s --classic --channel=1.25
    - usermod -aG microk8s $USER
    - chown -f -R $USER ~/.kube
    - su - $USER
  - micro k8s 상태 확인
    - microk8s status --wait-ready (##여기서 오류가 날 수 있다. 문제해결 못함)
    - microk8s enable dns ingress storage
    - microk8s.kubectl get ns
  - microk8s.kubectl의 Alias 만들기
    - echo 'source <(microk8s.kubectl completion bash)' >> ~/.bashrc
    - echo 'alias k=microk8s.kubectl' >> ~/.bashrc
    - echo 'complete -o default -F __start_kubectl k' >> ~/.bashrc
    - exec bash

micro k8s 설치하기

- **CentOS 7에 micro k8s 설치하기**
  - micro k8s 관리하기
    - k get ns
    - k get node -o wide
    - k get pod -A
- **micro k8s 클러스터에 가입하기**
  - 기존 master node(control plane node)에서 작업
    - **vi /etc/hosts**
      172.30.1.53 node6 (##필수 작업)
    - **microk8s add-node**
    - 여기에 나오는 것을 복사하여 가입하고자 하는 node에서 실행한다

```
root@node5:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 userver
172.30.1.53 node6
```

  - 가입할 worker node에서 작업
    - snap install microk8s --classic --channel=1.25
    - **microk8s join**
      172.30.1.16:25000/b4a497ca6f3ac3484237dd69636ee7bd/b609d88e56d3
  - 기존 master node에서 작업
    - **k get node -o wide**

```
root@node5:~# k get node -o wide
NAME    STATUS   ROLES    AGE     VERSION   INTERNAL-IP     CONTAINER-RUNTIME
node5   Ready    <none>   13h     v1.25.4   172.30.1.16     containerd://1.6.9
node6   Ready    <none>   3m20s   v1.25.4   172.30.1.253    containerd://1.6.9
```

micro k8s 설치하기

- **Deployment 배포 및 Pod 내용 수정하기**
  - deployment 생성하기
    - **k create deployment myweb --image nginx --replicas 5**
    - **k get pod -w** ##Pod이 생성되고 실행되는 상태 확인
    - **k get pod -o wide**

```
root@node5:~# k get pod -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP              NODE
myweb-d5b9458bc-sbpl9    1/1     Running   0          8s    10.1.33.161     node5
myweb-d5b9458bc-9sv8b    1/1     Running   0          8s    10.1.139.5      node6
myweb-d5b9458bc-ltbbx    1/1     Running   0          8s    10.1.33.162     node5
myweb-d5b9458bc-mb8f5    1/1     Running   0          8s    10.1.139.6      node6
myweb-d5b9458bc-rxjf8    1/1     Running   0          8s    10.1.139.7      node6
```

  - Pod에 접근하여 내용 확인 및 수정하기
    - **k exec** myweb-d5b9458bc-j8q8c **-- curl -s localhost | grep title**

```
root@node5:~# k exec myweb-d5b9458bc-j8q8c -- curl -s localhost | grep title
<title>Welcome to nginx!</title>
```

    - **k exec** myweb-d5b9458bc-j8q8c **-- sed -i 's/nginx/nginY :P/' /usr/share/nginx/html/index.html**
    - **k exec** myweb-*** **-- curl -s localhost | grep title**

```
root@node5:~# k exec myweb-d5b9458bc-j8q8c -- curl -s localhost | grep title
<title>Welcome to nginY :P!</title>
```

micro k8s 설치하기

- **micro k8s 제거하기**
  - Cluster에서 빠져 나오기
    - **microk8s leave**
    - **k get node -o wide**

```
root@node6:~# k get node
NAME     STATUS   ROLES
node5    Ready    <none>
node6    Ready    <none>
```

```
root@node6:~# microk8s leave
Generating new cluster certificates.
Waiting for node to start. .
root@node6:~# k get pod
No resources found in default namespace.
root@node6:~# k get node
NAME     STATUS   ROLES    AGE   VERSION
node6    Ready    <none>   14m   v1.25.4
root@node6:~# k get pod -A
NAMESPACE     NAME
kube-system   calico-node-ttm4v
kube-system   calico-kube-controllers-d8b9b6478-dntch
```

  - Cluster 삭제하기
    - **snap remove microk8s**

```
root@node6:~# snap remove microk8s
microk8s removed
root@node6:~# k get node
bash: /snap/bin/microk8s.kubectl: No such file or directory
```