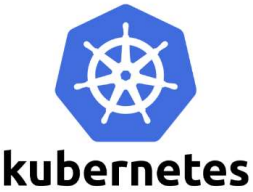
TABA_DB/SQL실습3

쿠버네티스란?

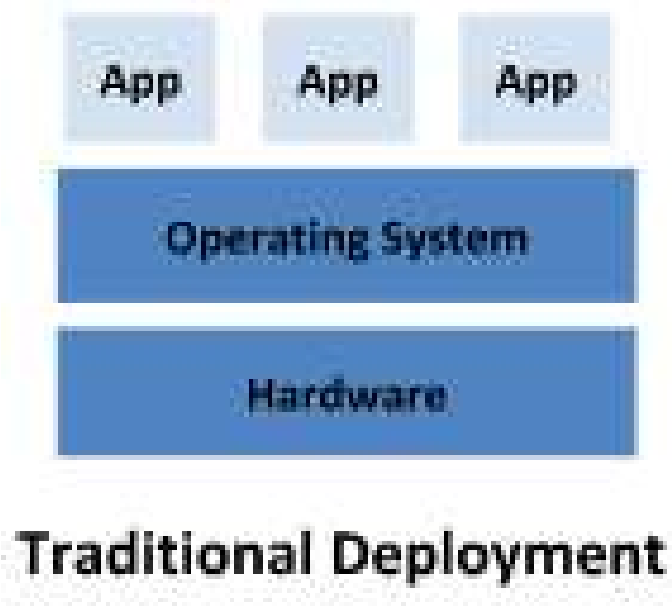


- 컨테이너를 쉽고 빠르게 배포, 확장 및 관리를 자동화해주는 오픈소스 플랫폼
- 명칭은 키잡이(helmsman)나 파일럿을 뜻하는 그리스어에서 유래
- K8s라는 표기는 "K"와 "s" 사이 8글자를 나타내는 약식 표기
- 구글이 2014년 쿠버네티스 프로젝트를 오픈소스화 함
- 관리자가 서버를 배포할 때 원하는 상태를 선언하는 방식 사용(Desired State)

쿠버네티스란?

kubernetes

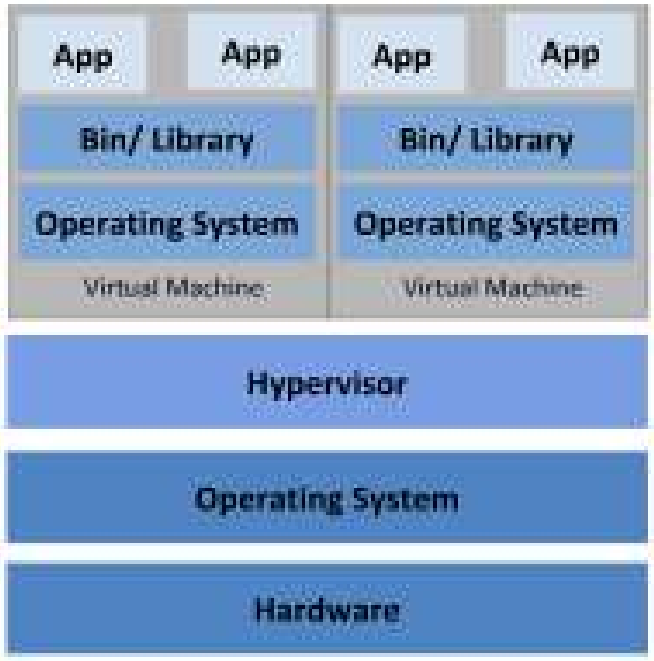
- 전통적인 배포 시대(Traditional Deployment)
  - 초기 조직은 애플리케이션을 물리 서버에서 실행
  - 리소스 할당 문제 발생
  - 여러 물리 서버에서 각 애플리케이션을 실행
  - 리소스가 충분히 활용되지 않음.
  - 물리 서버를 유지하는 데에 높은 비용



쿠버네티스란?

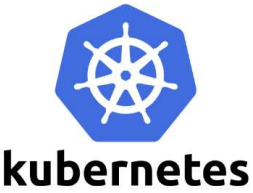
kubernetes

- 가상화된 배포 시대(Virtualized Deployment)
  - 전통적인 배포 시대의 해결책으로 가상화 도입
  - 단일 물리서버의 CPU에서 여러 가상 머신 실행
  - 가상화를 사용하면 VM간에 격리로 보안 제공
  - 물리서버에서 리소스를 효율적으로 사용

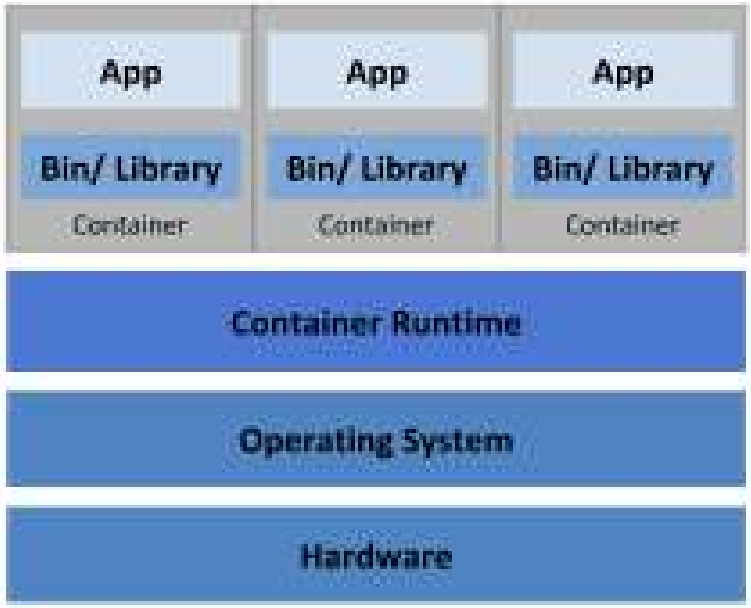


Virtualized Deployment

쿠버네티스란?



- 컨테이너 개발 시대(Container Deployment)
  - VM과 유사하지만 격리 속성 완화
  - 애플리케이션 간 OS 공유
  - VM이미지를 사용하는 것보다 컨테이너 이미지 생성이 쉽고 효율적
  - 클라우드 및 OS간 이식성(Ubuntu, RHEL, 퍼블릭 클라우드)



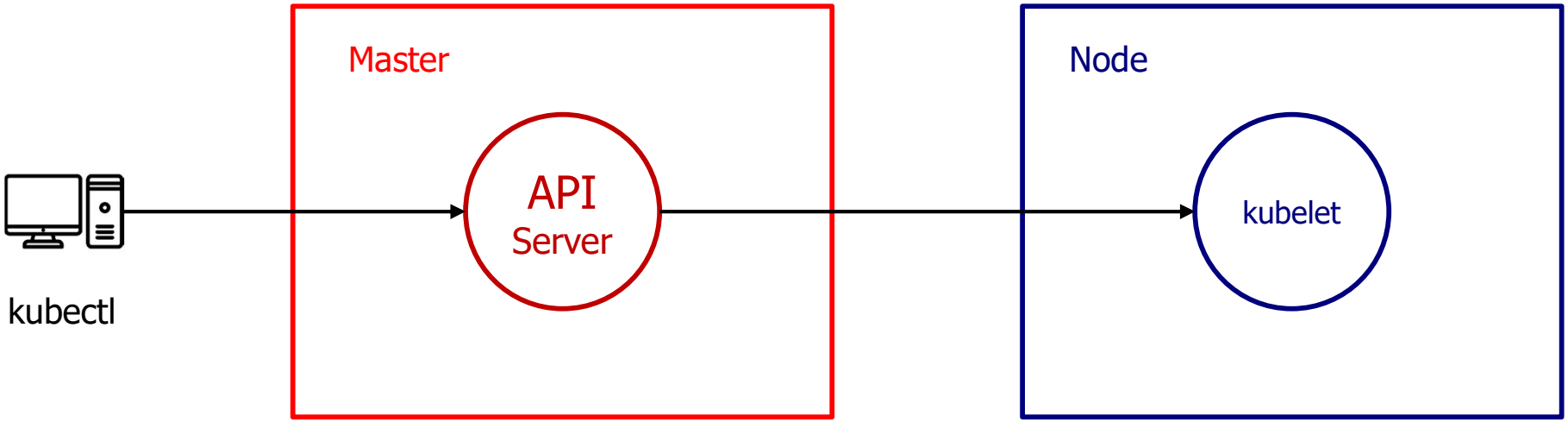
Container Deployment

쿠버네티스란?

- 컨테이너 오케스트레이션
  - 컨테이너 기반 애플리케이션 배포 관리, 제어 및 모니터링, 스케일링, 네트워킹 관리 도구
- 컨테이너 오케스트레이션 종류
  - 도커 스웜(Docker Swarm), 아파치 메소스(Apache Mesos), 노마드(Normad)
  - 쿠버네티스가 컨테이너 기반 인프라 시장에서 사실상 표준

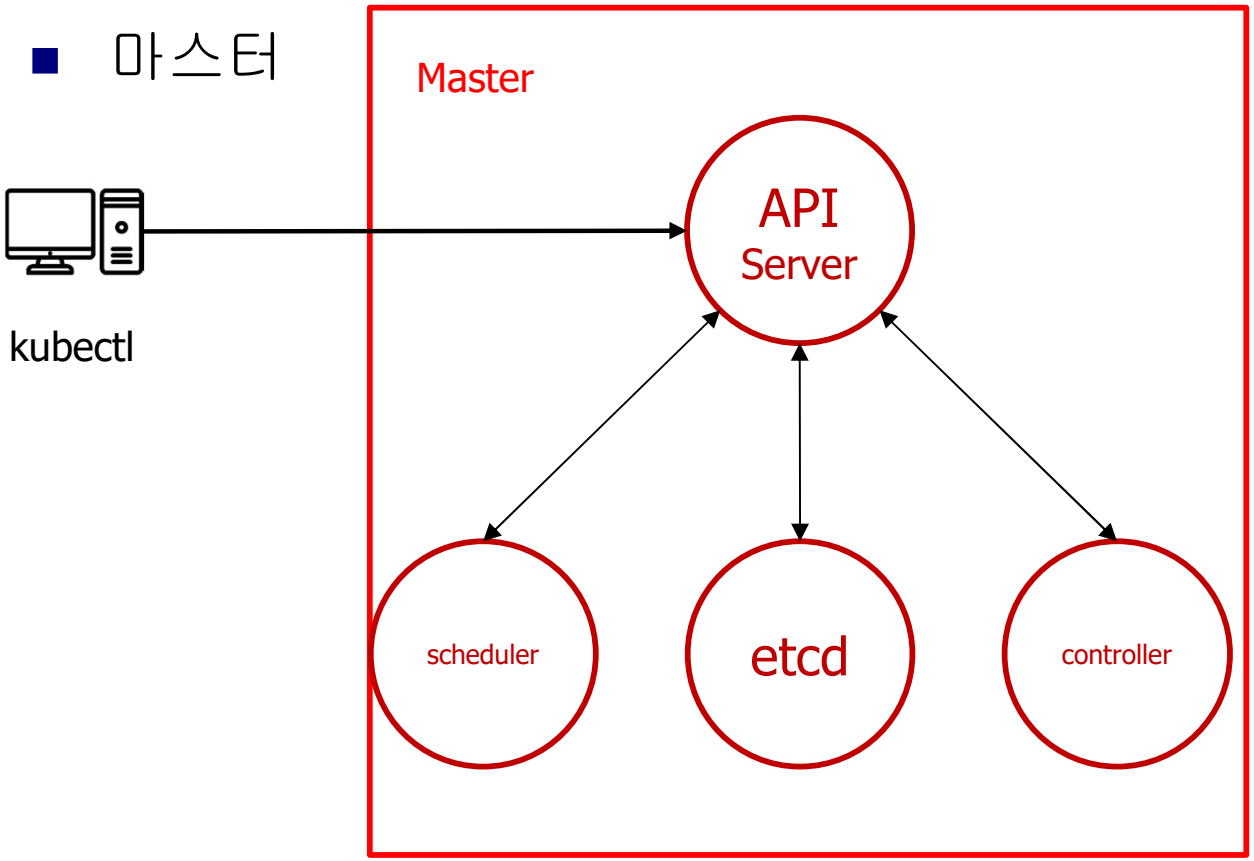
쿠버네티스 구조

- 마스터 - 노드 구조



쿠버네티스 구조

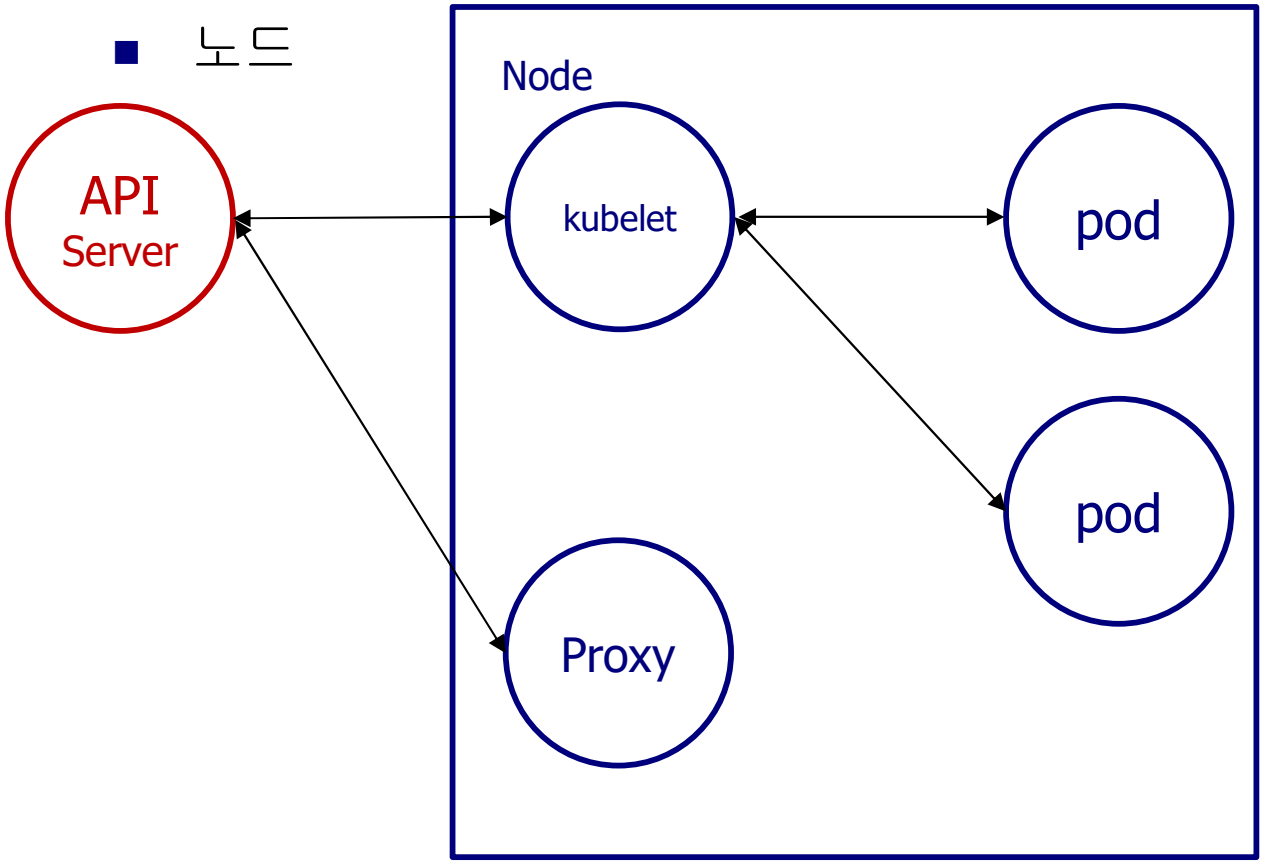
- 마스터



- 핵심 컴포넌트
- Kubectl
  - 마스터의 API Server 통신
- API Server
  - 관리자 요청 및 내부 모듈과 통신
  - etcd와 유일하게 통신
- etcd
  - 모든 상태와 데이터를 저장
  - Key-Value 형태
- Scheduler
  - 새로 생성된 pod을 감지
- controller
  - 다양한 컨트롤러 존재
  - 복제, 노드, 엔드포인트 등

쿠버네티스 구조

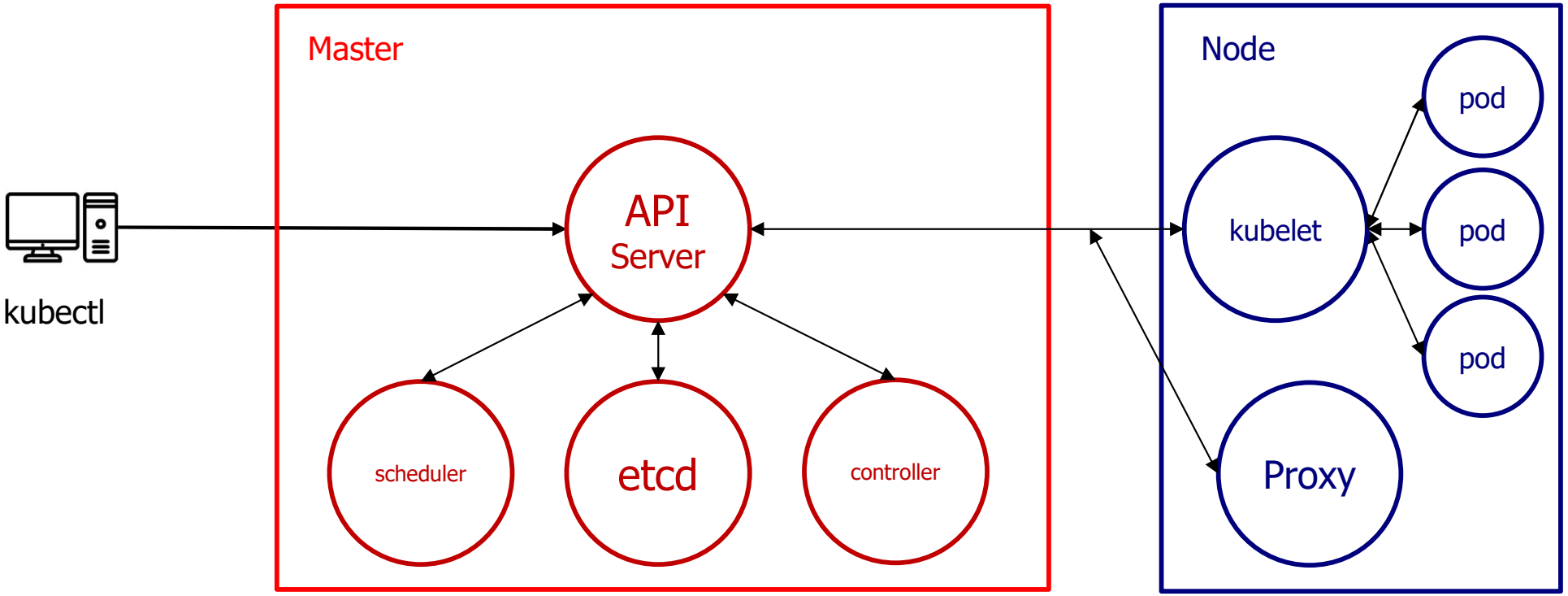
- 노드



- 핵심 컴포넌트
- kubelet
  - 각 노드에서 실행
  - pod을 실행, 중지 및 상태 체크
  - CRI(Container Runtime Interface)
  - - Docker, Containerd, CRI-O…
- proxy
  - 네트워크 프록시와 부하 분산 역할
  - 내/외부 통신 설정 등

쿠버네티스 구조

- 쿠버네티스 프로세스



minikube Install

- minikube
  - 쿠버네티스 클러스터를 설치하기 위해선 3대의 마스터 서버와 n개의 노드 서버가 필요.
  - mac, Linux, Windows에서 K8s 클러스터를 빠르게 설정해주는 도구
  - minikube를 이용하면 로컬에서 K8s 클러스터를 만들 수 있다.

minikube Install

- 업데이트 및 유틸리티 설치

sudo yum update -y && sudo yum install -y utils

- Docker-ce 레포 추가
  - sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

- Docker 설치

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

minikube Install

- Docker 실행 및 확인
- sudo systemctl start docker
- sudo systemctl enable docker
- sudo systemctl status docker

- 도커 그룹에 유저 추가

  sudo groupadd docker #(docker group 생성)

  sudo usermod -aG docker $USER #(docker group 해당 유저 추가)

  newgrp docker #(적용하기)

minikube Install

- minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
sudo rm minikube-linux-amd64
```

minikube Install

- minikube

  minikube start

```
spark@ubuntu-VirtualBox:~$ minikube start
* minikube v1.33.0 on Ubuntu 22.04 (vbox/amd64)
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.43 ...
* Downloading Kubernetes v1.30.0 preload ...
```

```
spark@ubuntu-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

minikube Install

- kubectl

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.i
nux/amd64/kubectl"

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    523      0 --:--:-- --:--:-- --:--:--   524
100 49.0M  100 49.0M    0     0  10.3M      0  0:00:04  0:00:04 --:--:-- 11.2M
```

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.i
nux/amd64/kubectl.sha256"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    560      0 --:--:-- --:--:-- --:--:--   560
100    64  100    64    0     0    202      0 --:--:-- --:--:-- --:--:--   202
spark@ubuntu-VirtualBox:~$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
```

minikube Install

- kubectl

  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

  chmod +x kubectl
  mkdir -p ~/.local/bin
  mv ./kubectl ~/.local/bin/kubectl

  kubectl version --client

```
spark@ubuntu-VirtualBox:~$ kubectl version --client
Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

minikube Install

- minikube 및 kubectl 명령어

minikube start : cluster 실행
minikube delete : cluster 삭제
minikube stop / pause : 정지 / 일시정지
minikube ip : 노드의 ip 확인
minikube ssh : node 접속

kubectl cluster-info : cluster 설정확인
kubectl delete pod pod명 --grace-period=0 --force :pod강제 삭제
kubectl get events : event 모니터링
kubectl describe pods/{pod명} or nodes/{node명} : 상세 보기

쿠버네티스 오브젝트

- 파드(Pod)
  - 쿠버네티스에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위
  - 하나 이상의 컨테이너 그룹
  - Pod는 스토리지 및 네트워크를 공유하고, 구동하는 방식에 대한 명세를 갖는다.
  - 컨테이너와 비슷한 개념이지만 완전히 같은 개념은 아님

쿠버네티스 오브젝트

- 파드 생성
- vim 편집기로 yaml작성

vi simple-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

kubectl run
- CLI로 pod를 생성

kubectl create
- yaml으로 pod 생성
- 동일한 pod가 있을 경우 에러 발생

kubectl apply
- yaml으로 pod 생성
- 동일한 pod가 없으면 새로운 pod 생성
- 동일한 pod가 있으면 기존 config와 비교해서 수정된 부분 업데이트

kubectl apply -f simple-pod.yaml #파드 생성
kubectl get po #생성된 파드 확인
kubectl delete -f simple-pod.yaml

```
spark@ubuntu-VirtualBox:~$ kubectl apply -f simple-pod.yaml
pod/nginx created
spark@ubuntu-VirtualBox:~$ kubectl get po nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          14s
spark@ubuntu-VirtualBox:~$ kubectl delete -f simple-pod.yaml
pod "nginx" deleted
```

DKU

DB Lab Dankook University 데이터베이스 연구실

쿠버네티스 오브젝트

- 네임스페이스(Namespace)
  - 쿠버네티스에서 사용되는 리소스들을 구분해 관리하는 그룹
  - Pod, Service 등은 네임스페이스별로 생성 및 관리 가능, 접근 권한도 다르게 설정 가능
  - 네임스페이스는 여러 개의 팀이나 프로젝트에 걸쳐 많은 사용자가 있는 환경에서 사용하도록 만들어짐
  - 쿠버네티스는 defalut, kube-node-lease, kube-public, kube-system이라는 네 개의 네임스페이스를 갖고 있다.

쿠버네티스 오브젝트

- 네임스페이스 조회
  kubectl get namespace or ns

```
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   15h
kube-node-lease   Active   15h
kube-public       Active   15h
kube-system       Active   15h
```

- 새 네임스페이스 생성1

vi my-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: newns1
```

- kubectl create -f my-namespace.yaml

- 아래 명령어로 네임스페이스 생성 가능
  kubectl create namespace | ns newns2

  kubectl get ns

```
spark@ubuntu-VirtualBox:~$ kubectl create -f my-namespace.yaml
namespace/newns1 created
spark@ubuntu-VirtualBox:~$ kubectl create namespace newns2
namespace/newns2 created
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   15h
kube-node-lease   Active   15h
kube-public       Active   15h
kube-system       Active   15h
newns1            Active   18s
newns2            Active   5s
```

kubectl delete ns newns2 #네임스페이스 삭제

쿠버네티스 오브젝트

- 서비스(Service)
  - 파드는 언제든 다른 노드로 옮겨지거나 삭제될 수 있음.
  - 생성될 때마다 새로운 IP를 받게 되는데, 내/외부 통신을 유지하기 어려움
  - 쿠버네티스에서 실행되고 있는 파드를 네트워크에 노출시키는 가상의 컴포넌트
  - 내/외부의 애플리케이션과 연결 혹은 사용자와 연결될 수 있도록 고정 IP를 갖는 서비스를 이용해 통신 가능

쿠버네티스 오브젝트

- 서비스 타입
  - Cluster IP
    - 가장 기본이 되는 Service 타입
    - 클러스터 내부 통신만 가능, 외부 트래픽 불가능
  - NodePort
    - 클러스터 내/외부 통신이 가능
    - 외부 트래픽을 전달받을 수 있음
    - 노드의 포트를 사용
  - LoadBlancer
    - 기본적으로 외부에 존재하며, 클라우드 프로바이더와 함께 사용되어 외부 트래픽을 받음
  - ExternalName
    - 위 3가지와 전혀 다른 서비스 타입
    - 외부로 나가는 트래픽을 변환하기 위한 용도
    - 도메인 이름을 변환하여 연결해주는 역할

쿠버네티스 오브젝트

- 볼륨(Volume)
  - 쿠버네티스에서 파드를 생성하면 디렉토리를 임시로 사용함
  - 컨테이너가 삭제되면 파일 손실되는 문제 발생
  - 파드가 사라지더라도 사용할 수 있는 디렉터리는 볼륨 오브젝트를 이용해 생성
  - Docker의 볼륨과 비슷한 개념
  - 쿠버네티스 버전에 따라 사용 가능한 볼륨이 있음
  - 볼륨 종류
    - emptyDir : 일시적 데이터 저장, 파드가 삭제되면 스토리지도 삭제
    - localPath: 노드의 파일시스템을 파드로 마운트
    - nfs
    - awsEBS, gcePersistentDisk, azureDisk : 클라우드 전용 스토리지
    - PV(Persistent Volumes) : 영구볼륨, 포드가 종료되어도 데이터는 삭제되지 않음.
    - PVC(Persistent Volumes Claim) : 생성된 PV를 사용

쿠버네티스 오브젝트

- PV yaml 작성

  demoPV.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: demo-pv
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/pv/log"
  persistentVolumeReclaimPolicy: Retain
```

- capacity : 볼륨크기
- accessModes
  - ReadWriteOnce : 하나의 노드에서 RW 가능
  - ReadOnlyMany : 여러노드에서 R 가능
  - ReadWriteMany : 여러 노드에서 RW가능
- persistentVolumeReclaimPolicy
  - PV 종료 시 볼륨에 저장된 데이터 삭제 옵션
  - Retain : PVC가 삭제되어도 PV의 데이터 보존
  - Delete : PVC가 삭제되면 데이터 및 PV도 삭제
  - Recycle : PVC가 삭제되면 데이터만 삭제

쿠버네티스 오브젝트

- PVC yaml 작성

demoPVC.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: demo-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

- capacity : 볼륨크기
- accessModes
  - 사용하는 PV와 동일한 옵션을 사용해야함
- requests
  - 사용을 원하는 스토리지 요구 명시
  - storage : 사용하고자 하는 최소한의 크기

쿠버네티스 오브젝트

- PV, PVC 생성 및 조회

  kubectl create -f demoPV.yaml

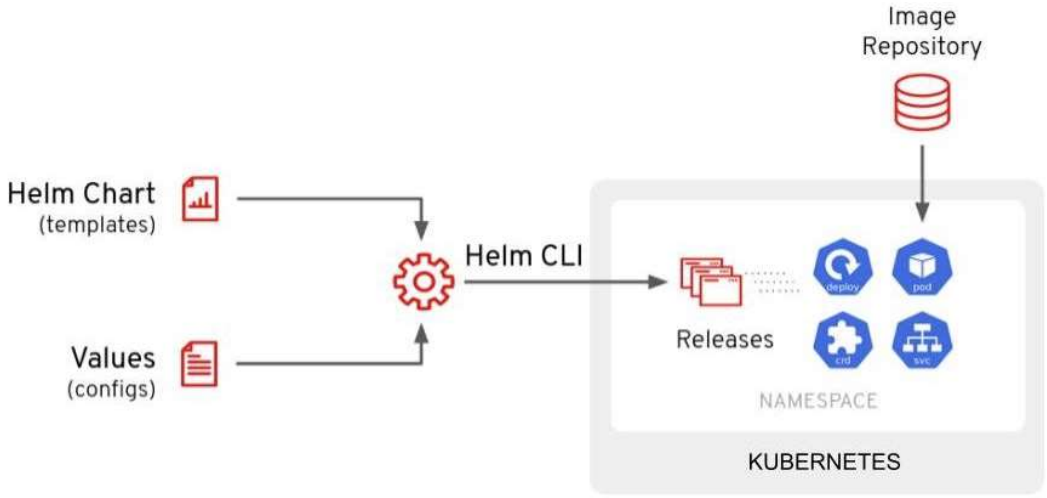
  kubectl create -f dempPVC.yaml

  kubectl get persistentvolume #pv

```
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPV.yaml
persistentvolume/demo-pv created
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPVC.yaml
persistentvolumeclaim/demo-pvc created
spark@ubuntu-VirtualBox:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
demo-pv                                    100Mi      RWX            Retain           Available
pvc-32e18ed2-a636-49e8-8a99-6c930e2a8257   2Gi        RWO            Delete           Bound       default/storage-prometheus-alertmanager-0
pvc-8cdb1b2e-13e0-4367-9f2d-03824f77c5eb   50Mi       RWX            Delete           Bound       default/demo-pvc
```

Helm

- Helm
  - K8s 애플리케이션을 위한 오픈소스 패키지 매니저
  - Helm 구성 요소
    - Chart : 애플리케이션을 배포하는데 사용되는 관련 Kubernetes YAML파일
    - Repository : 차트를 저장, 공유, 배포할 수 있는 곳
    - Release : Kubernetes 클러스터에 배포된 차트 특정 인스턴스



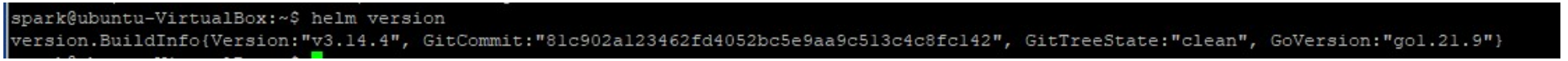
Helm

- Helm 설치

curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh

- 설치 확인

helm version



Helm

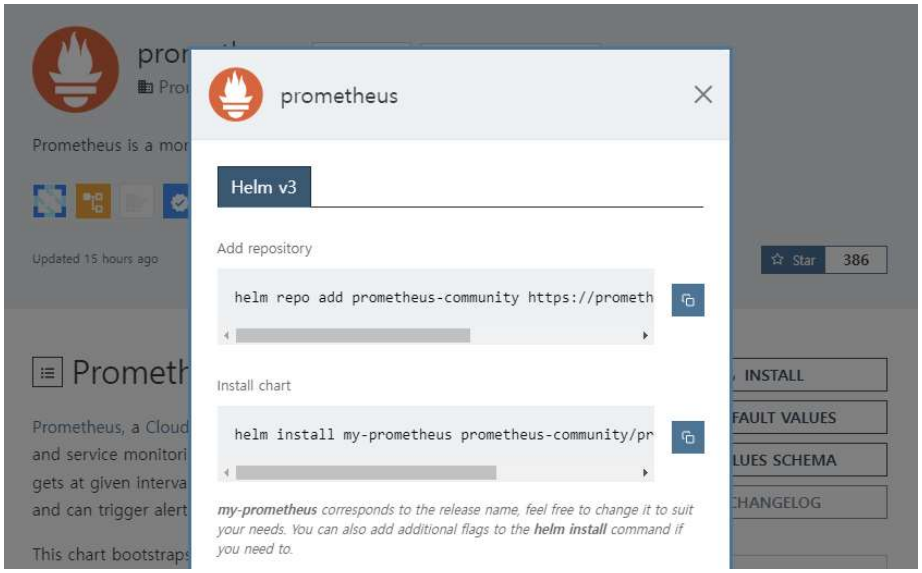
- Helm 명령어
  - helm search hub : 저장소에 있는 helm차트를 helm hub에서 검색
  - helm install chart명 : 특정 chart패키지 설치
  - helm show values chart명 : chart 옵션 설정가능한 values 조회
  - helm uninstall release명 : 릴리즈 삭제
  - helm repo list : 저장된 저장소 목록 조회
  - helm repo add {name} {url} : 저장소 저장
  - helm repo remove {name} : 저장소 삭제

Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
  - SoundCloud에서 만든 오픈소스 모니터링 툴
  - Kubernetes환경에서 모니터링하기 원하는 리소스로부터 metric을 수집하고 해당 metric을 이용해 모니터링
- 그라파나(Grafana)
  - 오픈소스 시각화 분석 도구
  - 여러 데이터 소스에 대한 대시보드 템플릿 제공
  - 프로메테우스도 UI를 제공하지만, 기능이 빈약해 그라파나와 연동 사용

Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- https://artifacthub.io/ 에서 Prometheus 검색 - Install - Add repo
  helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
  helm repo update



Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- helm install prometheus prometheus-community/prometheus
- kubectl get all

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                       1/1     Running   0          5m54s
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0          5m54s
pod/prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0          5m54s
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb   1/1     Running   0          5m54s
pod/prometheus-server-55768b86b9-5rjqj              2/2     Running   0          5m54s

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        83m
service/prometheus-alertmanager               ClusterIP   10.104.30.167    <none>        9093/TCP       5m54s
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       5m54s
service/prometheus-kube-state-metrics         ClusterIP   10.96.185.187    <none>        8080/TCP       5m54s
service/prometheus-prometheus-node-exporter   ClusterIP   10.106.132.146   <none>        9100/TCP       5m54s
service/prometheus-prometheus-pushgateway     ClusterIP   10.103.200.90    <none>        9091/TCP       5m54s
service/prometheus-server                     ClusterIP   10.109.183.118   <none>        80/TCP         5m54s
service/prometheus-server-ext                 NodePort    10.111.104.25    <none>        80:32753/TCP   101s

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1       1            1           kubernetes.io/os=linux   5m54s

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics  1/1     1            1           5m54s
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           5m54s
deployment.apps/prometheus-server              1/1     1            1           5m54s

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-76fc9c6f55  1         1         1       5m54s
replicaset.apps/prometheus-prometheus-pushgateway-7c758897fd   1         1         1       5m54s
replicaset.apps/prometheus-server-55768b86b9              1         1         1       5m54s
```

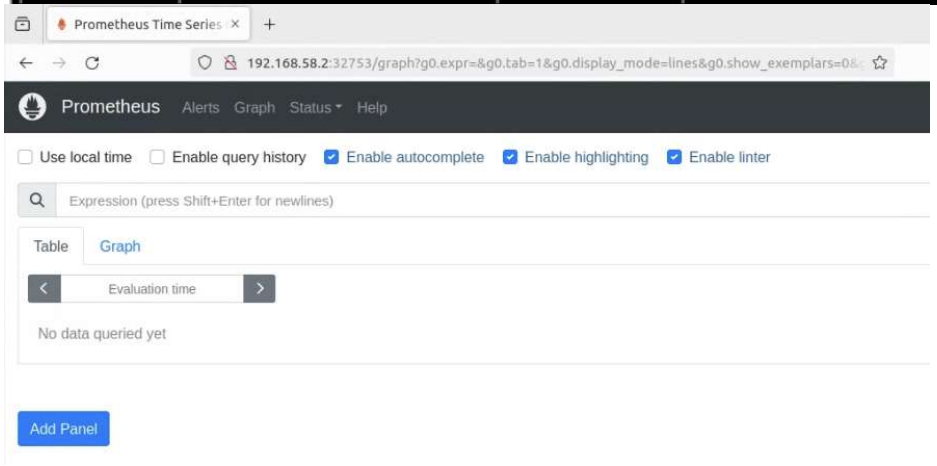
Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- 외부에서 접속(VM에서 접속)

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

minikube service prometheus-server-ext

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ minikube service prometheus-server-ext
|-----------|-----------------------|-------------|---------------------------|
| NAMESPACE |         NAME          | TARGET PORT |            URL            |
|-----------|-----------------------|-------------|---------------------------|
| default   | prometheus-server-ext |          80 | http://192.168.58.2:32753 |
|-----------|-----------------------|-------------|---------------------------|
```

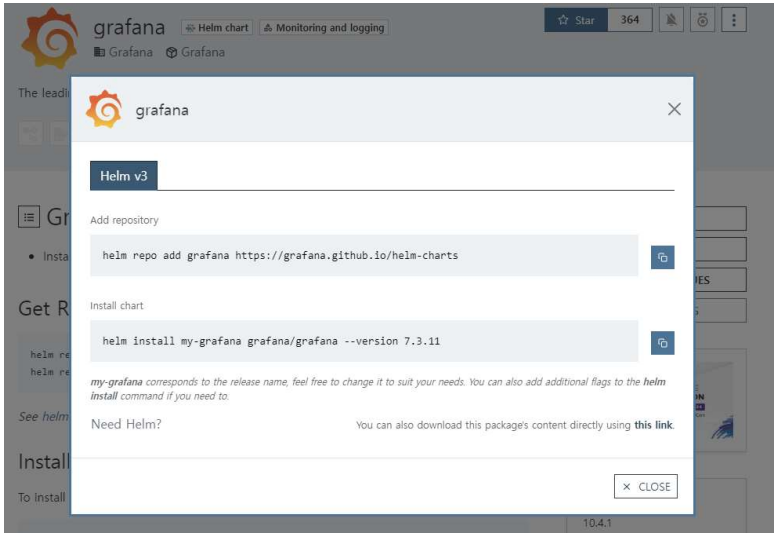


Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)
- https://artifacthub.io/ 에서 Grafana 검색 - Install - Add repo

  helm repo add grafana https://grafana.github.io/helm-charts

  helm repo update



Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

helm install grafana grafana/grafana

kubectl get all

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/grafana-5ccd97668d-cxl9x                        1/1     Running   0          3m14s
pod/prometheus-alertmanager-0                       1/1     Running   0          17m
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0          17m
pod/prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0          17m
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb  1/1     Running   0          17m
pod/prometheus-server-55768b86b9-5rjqj              2/2     Running   0          17m

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/grafana                               ClusterIP   10.106.237.143   <none>        80/TCP         3m14s
service/grafana-ext                           NodePort    10.98.8.196      <none>        80:31012/TCP   3m3s
service/kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        94m
service/prometheus-alertmanager               ClusterIP   10.104.30.167    <none>        9093/TCP       17m
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       17m
service/prometheus-kube-state-metrics         ClusterIP   10.96.185.187    <none>        8080/TCP       17m
service/prometheus-prometheus-node-exporter   ClusterIP   10.106.132.146   <none>        9100/TCP       17m
service/prometheus-prometheus-pushgateway     ClusterIP   10.103.200.90    <none>        9091/TCP       17m
service/prometheus-server                     ClusterIP   10.109.183.118   <none>        80/TCP         17m
service/prometheus-server-ext                 NodePort    10.111.104.25    <none>        80:32753/TCP   12m

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1       1            1           kubernetes.io/os=linux   17m

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana                        1/1     1            1           3m14s
deployment.apps/prometheus-kube-state-metrics  1/1     1            1           17m
deployment.apps/prometheus-prometheus-pushgateway  1/1     1            1           17m
deployment.apps/prometheus-server              1/1     1            1           17m
```

DKU

DB Lab Dankook University 데이터베이스 연구실

Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

kubectl get po

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get po
NAME                                            READY   STATUS    RESTARTS   AGE
grafana-5ccd97668d-cxl9x                        1/1     Running   0          23m
prometheus-alertmanager-0                       1/1     Running   0          37m
prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0          37m
prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0          37m
```

pod명 확인 후 패스워드 변경

kubectl exec --namespace default -it pod명 -- /bin/bash #pod 접속

grafana-cli admin reset-admin-password Tibero1! #Tibero1!는 패스워드

Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext

minikube service grafana-ext

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ minikube service grafana-ext
|-----------|-------------|-------------|---------------------------|
| NAMESPACE |    NAME     | TARGET PORT |            URL            |
|-----------|-------------|-------------|---------------------------|
| default   | grafana-ext |          80 | http://192.168.58.2:31012 |
|-----------|-------------|-------------|---------------------------|
```

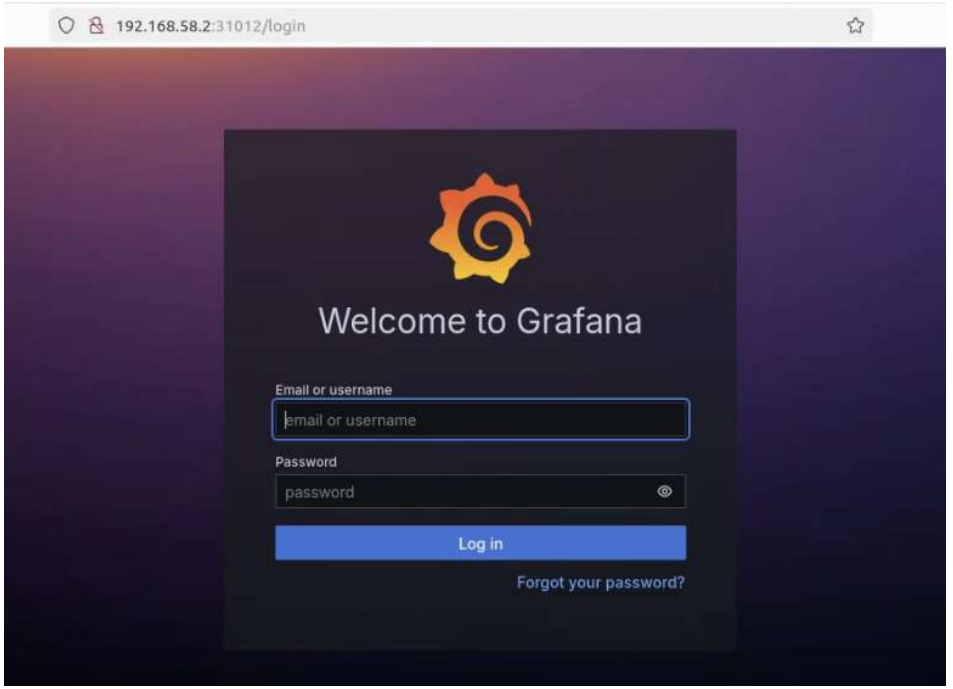
Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

192.168.58.2:31012

ID : admin

PW : 변경한 패스워드



Wordpress 만들기

- helm chart 구조
- <chart name> /

  Chart.yaml : 차트의 메타 데이터

  values.yaml : 패키지 사용자화

  templates/ : YAML 오브젝트 파일

- artifacthub.io 에서 wordpress 검색 후 repo 추가

  helm repo add bitnami https://charts.bitnami.com/bitnami

- 내부 저장소에서 검색

  helm search repo wordpress

```
ubuntu@ubuntu-VirtualBox:~$ helm search repo wordpress
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/wordpress       22.2.7          6.5.3           WordPress is the world's most popular blogging ...
bitnami/wordpress-intel 2.1.31          6.1.1           DEPRECATED WordPress for Intel is the most popu...
```

Wordpress 만들기

- wordpress chart 정보 확인
  - helm inspect values bitnami/wordpress

```
ubuntu@ubuntu-VirtualBox:~$ helm inspect values bitnami/wordpress
# Copyright Broadcom, Inc. All Rights Reserved.
# SPDX-License-Identifier: APACHE-2.0

## @section Global parameters
## Global Docker image parameters
## Please, note that this will override the image parameters, including dependencies, configured to use the global value
## Current available global Docker image parameters: imageRegistry, imagePullSecrets and storageClass
##

## @param global.imageRegistry Global Docker image registry
## @param global.imagePullSecrets Global Docker registry secret names as an array
## @param global.storageClass Global StorageClass for Persistent Volume(s)
##
global:
  imageRegistry: ""
  ## E.g.
  ## imagePullSecrets:
  ##   - myRegistryKeySecretName
  ##
  imagePullSecrets: []
  storageClass: ""
  ## Compatibility adaptations for Kubernetes platforms
  ##
  compatibility:
    ## Compatibility adaptations for Openshift
    ##
    openshift:
      ## @param global.compatibility.openshift.adaptSecurityContext Adapt the securityContext sections of the deployment to
ove runAsUser, runAsGroup and fsGroup and let the platform use their allowed default IDs. Possible values: auto (apply if th
he adaptation always), disabled (do not perform adaptation)
      ##
      adaptSecurityContext: auto
```

Wordpress 만들기

- wordpress 설치

  helm install my-wordpress bitnami/wordpress

  kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m44s
my-wordpress-mariadb-0          1/1     Running   0          3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m46s
my-wordpress-mariadb-0          1/1     Running   0          3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0          3m49s

NAME                          TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/kubernetes            ClusterIP      10.96.0.1        <none>        443/TCP
service/my-wordpress          LoadBalancer   10.106.216.205   <pending>     80:30589/TCP,443:3273
service/my-wordpress-mariadb  ClusterIP      10.96.207.31     <none>        3306/TCP

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress  1/1     1            1           3m49s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478   1         1         1       3m49s

NAME                                  READY   AGE
statefulset.apps/my-wordpress-mariadb 1/1     3m49s
```

Wordpress 만들기

- Pod 정보 확인
  - kubectl describe po my-wordpress-cf9758478-wtvqf #개인마다 Pod의 이름은 다름

```
Environment:
  BITNAMI_DEBUG:                  false
  ALLOW_EMPTY_PASSWORD:           yes
  WORDPRESS_SKIP_BOOTSTRAP:       no
  MARIADB_HOST:                   my-wordpress-mariadb
  MARIADB_PORT_NUMBER:            3306
  WORDPRESS_DATABASE_NAME:        bitnami_wordpress
  WORDPRESS_DATABASE_USER:        bn_wordpress
  WORDPRESS_DATABASE_PASSWORD:    <set to the key 'mariadb-password' in secret 'my-wordpres
  WORDPRESS_USERNAME:             user
  WORDPRESS_PASSWORD:             <set to the key 'wordpress-password' in secret 'my-wordpr
  WORDPRESS_EMAIL:                user@example.com
  WORDPRESS_FIRST_NAME:           FirstName
  WORDPRESS_LAST_NAME:            LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE:       no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE:  no
  WORDPRESS_BLOG_NAME:            User's Blog!
  WORDPRESS_TABLE_PREFIX:         wp_
  WORDPRESS_SCHEME:               http
  WORDPRESS_EXTRA_WP_CONFIG_CONTENT:
```

Wordpress 만들기

- wordpress 삭제

  helm uninstall my-wordpress bitnami/wordpress

  kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m44s
my-wordpress-mariadb-0          1/1     Running   0          3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m46s
my-wordpress-mariadb-0          1/1     Running   0          3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0          3m49s

NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP
service/my-wordpress            LoadBalancer   10.106.216.205   <pending>     80:30589/TCP,443:3273
service/my-wordpress-mariadb    ClusterIP      10.96.207.31     <none>        3306/TCP

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress    1/1     1            1           3m49s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478     1         1         1       3m49s

NAME                                    READY   AGE
statefulset.apps/my-wordpress-mariadb   1/1     3m49s
```

Wordpress 만들기

- wordpress 차트 커스터마이징

  helm install my-wordpress bitnami/wordpress --set mariadb.auth.username=hello_wordpress

  kubectl get po

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
custmz-wordpress-7c675b5474-kwz9r   1/1     Running   0          92s
custmz-wordpress-mariadb-0          1/1     Running   0          92s
```

  kubectl describe po custmz-wordpress-7c675b5474-kwz9r

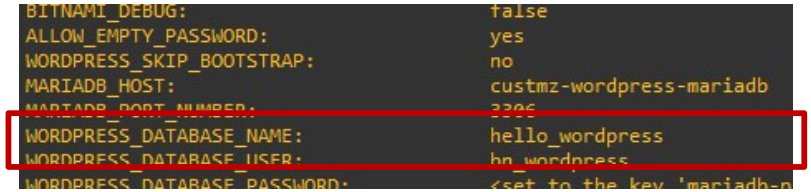
```
Environment:
  BITNAMI_DEBUG:                  false
  ALLOW_EMPTY_PASSWORD:           yes
  WORDPRESS_SKIP_BOOTSTRAP:       no
  MARIADB_HOST:                   custmz-wordpress-mariadb
  MARIADB_PORT_NUMBER:            3306
  WORDPRESS_DATABASE_NAME:        bitnami_wordpress
  WORDPRESS_DATABASE_USER:        hello_wordpress
  WORDPRESS_DATABASE_PASSWORD:    <set to the key 'mariadb-password' in secret 'custmz-wordpress-mariadb'>  Option
  WORDPRESS_USERNAME:             user
  WORDPRESS_PASSWORD:             <set to the key 'wordpress-password' in secret 'custmz-wordpress'>  Optional: fa
  WORDPRESS_EMAIL:                user@example.com
  WORDPRESS_FIRST_NAME:           FirstName
  WORDPRESS_LAST_NAME:            LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE:       no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE:  no
```

Wordpress 만들기

- wordpress 차트 커스터마이징

helm install custmz-wordpress bitnami/wordpress -f valuesF.yaml

kubectl describe po custmz-wordpress-6fc7c49494-bbv56



vi valuesF.yaml

```
mariadb:
  auth:
    database: hello_wordpress
```