Learning Kubernetes

1. 도커 기본 다지기

1.1 도커 설치하기

1.1.1 Ubuntu 설치

```
sudo apt-get update
sudo apt-get install docker.io
sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
```

> 우분투 패키지 메니저는 apt-get , apt-cache, apt 가 있습니다. 모두 동일한 명령어라고 보면 되지만, apt 를 쓸경우 일단 글자수가 적고, 출력 Output 에 색상이 추가되어 좀더 예쁘게 보입니다.

1.1.2 CentOS 6

```
sudo yum install http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-
8.noarch.rpm
sudo yum install docker-io
```

1.1.3 CentOS 7

```
sudo yum install docker
sudo service docker start
sudo chkconfig docker on # 부팅시에 자동 스타트업
```

1.2 도커 기본 명령어

1.2.1 도커 hub 사용을 위한 계정생성

```
sudo docker login
```

> **docker hub** (hub.docker.dom) 에 가입 후 명령어를 실행 해야 로그인이 가능 합니다.

1.2.2 도커 이미지 검색

search 명령으로 nginx 를 검색해 봅니다.

```
sudo docker search nginx
```

```
dangtong@dangtongui-MacBook-Pro ~
dangtong@dangtongui-MacBook-Pro ~ docker search nginx
NAME                              DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
nginx                             Official build of Nginx.                        12428               [OK]
jwilder/nginx-proxy               Automated Nginx reverse proxy for docker con…   1712                                    [OK]
richarvey/nginx-php-fpm           Container running Nginx + PHP-FPM capable of…   749                                     [OK]
linuxserver/nginx                 An Nginx container, brought to you by LinuxS…   84
bitnami/nginx                     Bitnami nginx Docker Image                      75                                      [OK]
tiangolo/nginx-rtmp               Docker image with Nginx using the nginx-rtmp…   60                                      [OK]
nginxdemos/hello                  NGINX webserver that serves a simple page co…   35                                      [OK]
jc21/nginx-proxy-manager          Docker container for managing Nginx proxy ho…   34
nginx/unit                        NGINX Unit is a dynamic web and application …   33
jlesage/nginx-proxy-manager       Docker container for Nginx Proxy Manager        31                                      [OK]
nginx/nginx-ingress               NGINX Ingress Controller for Kubernetes         22
privatebin/nginx-fpm-alpine       PrivateBin running on an Nginx, php-fpm & Al…   19                                      [OK]
schmunk42/nginx-redirect          A very simple container to redirect HTTP tra…   17                                      [OK]
nginxinc/nginx-unprivileged       Unprivileged NGINX Dockerfiles                  12
centos/nginx-18-centos7           Platform for running nginx 1.8 or building n…   12
blacklabelops/nginx               Dockerized Nginx Reverse Proxy Server.          12                                      [OK]
centos/nginx-112-centos7          Platform for running nginx 1.12 or building …   11
nginx/nginx-prometheus-exporter   NGINX Prometheus Exporter                       9
sophos/nginx-vts-exporter         Simple server that scrapes Nginx vts stats a…   6                                       [OK]
1science/nginx                    Nginx Docker images that include Consul Temp…   5                                       [OK]
mailu/nginx                       Mailu nginx frontend                            5                                       [OK]
pebbletech/nginx-proxy            nginx-proxy sets up a container running ngin…   2                                       [OK]
ansibleplaybookbundle/nginx-apb   An APB to deploy NGINX                          1                                       [OK]
centos/nginx-110-centos7          Platform for running nginx 1.10 or building …   0
wodby/nginx                       Generic nginx                                   0                                       [OK]
```

1.2.3 도커 이미지 다운로드

nodejs 이미지를 설치해봅니다. 최신 안정버전인 12.14 버전을 설치 합니다.

```
sudo docker pull nodejs:latest    # 최신버전 다운로드
sudo docker pull nodejs:12.14.0   # 특정버전 다운로드
sudo docker pull -a nodejs        # 모든버전 다운로드
```

```
 ✘ dangtong@dangtongui-MacBook-Pro ~ docker pull node:12.14
12.14: Pulling from library/node
146bd6a88618: Pull complete
9935d0c62ace: Pull complete
db0efb86e806: Pull complete
e705a4c4fd31: Pull complete
c877b722db6f: Pull complete
645c20ec8214: Pull complete
291b293f2ff3: Pull complete
f748cfa05737: Pull complete
44a597fb116e: Pull complete
Digest: sha256:f490ebb9c7d5dcf1a8a1e4d3b3a65e133be44d26abb66815ca1612ef69410c51
Status: Downloaded newer image for node:12.14
docker.io/library/node:12.14
 dangtong@dangtongui-MacBook-Pro ~
```

1.2.4 도커 이미지 목록 보기

```
docker image list
docker image ls
docker images
```

```
 dangtong@dangtongui-MacBook-Pro ~ docker image list
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
node                    12.14               6b5991bf650f        6 days ago          913MB
dangtong/kubia          latest              7464540a4dc7        13 days ago         660MB
kubia                   latest              7464540a4dc7        13 days ago         660MB
dangtong76/echo_test    latest              490fec1ca30e        3 weeks ago         917MB
python                  3.7                 fbf9f709ca9f        6 weeks ago         917MB
node                    7                   d9aed20b68a4        2 years ago         660MB
 dangtong@dangtongui-MacBook-Pro ~ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
node                    12.14               6b5991bf650f        6 days ago          913MB
dangtong/kubia          latest              7464540a4dc7        13 days ago         660MB
kubia                   latest              7464540a4dc7        13 days ago         660MB
dangtong76/echo_test    latest              490fec1ca30e        3 weeks ago         917MB
python                  3.7                 fbf9f709ca9f        6 weeks ago         917MB
node                    7                   d9aed20b68a4        2 years ago         660MB
 dangtong@dangtongui-MacBook-Pro ~ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
node                    12.14               6b5991bf650f        6 days ago          913MB
dangtong/kubia          latest              7464540a4dc7        13 days ago         660MB
kubia                   latest              7464540a4dc7        13 days ago         660MB
dangtong76/echo_test    latest              490fec1ca30e        3 weeks ago         917MB
python                  3.7                 fbf9f709ca9f        6 weeks ago         917MB
node                    7                   d9aed20b68a4        2 years ago         660MB
 dangtong@dangtongui-MacBook-Pro ~
```

1.3 도커 이미지 생성 하기

1.3.1 서비스를 위한 Application 코드 작성

hostname_finder 라는 폴더를 만들고 그 아래 app.js 및 Dockerfiles 2개 파일을 작성 합니다.

먼저 vi 또는 gedit 를 실행해서 아래 파일을 app.js 라는 이름 으로 작성 합니다.

```
package main

import (
  "fmt"
  "os"
  "log"
  "net/http"
)
func handler(w http.ResponseWriter, r *http.Request){
  name, err := os.Hostname()
  if err != nil {
    panic(err)
  }

  fmt.Fprintln(w,"hostname:", name)
}
func main() {
  fmt.Fprintln(os.Stdout,"Starting GoApp Server......")
  http.HandleFunc("/",handler)
  log.Fatal(http.ListenAndServe(":8080",nil))
}
```

1.3.2 이미지 생성을 위한 Dockerfile 작성

```
FROM golang:1.11-alpine AS build

WORKDIR /src/
COPY main.go go.* /src/
RUN CGO_ENABLED=0 go build -o /bin/demo

FROM scratch
COPY --from=build /bin/demo /bin/demo
ENTRYPOINT ["/bin/demo"]
```

1.3.3 컨테이너 이미지 생성

hostname_finder 라는 폴더 안에서 아래 명령어를 실행 합니다.

```
docker build -t goapp .
```

> . 은 현재 디렉토리서 Dockerfile 참조해서 first-container 라는 이미지를 생성 합니다.

[출력]

```
Sending build context to Docker daemon  3.072kB
Step 1/7 : FROM golang:1.11-alpine AS build
 ---> e116d2efa2ab
Step 2/7 : WORKDIR /src/
 ---> Using cache
 ---> c3210d8eb11f
Step 3/7 : COPY main.go go.* /src/
 ---> ef55118ea78c
Step 4/7 : RUN CGO_ENABLED=0 go build -o /bin/demo
 ---> Running in e557730bf11c
Removing intermediate container e557730bf11c
 ---> d55bd9bd3f81
Step 5/7 : FROM scratch
 --->
Step 6/7 : COPY --from=build /bin/demo /bin/demo
 ---> bb4b1250a05e
Step 7/7 : ENTRYPOINT ["/bin/demo"]
 ---> Running in 4419d56988aa
Removing intermediate container 4419d56988aa
 ---> 36f5c919e3b8
Successfully built 36f5c919e3b8
Successfully tagged goapp:latest
```

이미지가 생성 되었는지 명령어를 통해 확인 합니다.

```
docker images
```

[출력]

```
REPOSITORY                          TAG                 IMAGE ID            CREATED
         SIZE
<none>                              <none>              d55bd9bd3f81        About a
minute ago   325MB
goapp                               latest              36f5c919e3b8        About a minute
ago   6.51MB
<none>                              <none>              1c688e9c7e3c        3 days ago
        325MB
<none>                              <none>              9b60c66a5b82        3 days ago
        6.51MB
<none>                              <none>              7fc44021a96f        3 days ago
        325MB
<none>                              <none>              2caa0c2ac791        3 days ago
        325MB
<none>                              <none>              c46d81105b65        3 days ago
        6.51MB
<none>                              <none>              2d78705fb4ae        3 days ago
        312MB
```

1.4 도커 컨테이너 시작 및 서비스 확인

1.4.1 도커 컨테이너 시작

```
docker run --name goapp-project -p 8080:8080 -d goapp
docker run -it --name goapp-project -p 8080:8080 -d goapp /bin/bash
```

> --name : 실행한 도커 컨테이너의 이름 지정
>
> -p : 포트 맵핑 정보 localhost 와 컨테이너 포트를 맵핑 합니다.
>
> -d : Docker 컨테이너를 백그라운드로 수행하고 컨테이너 ID를 출력 합니다.
>
> -i : STDIN 계속 interactive 모드로 유지

1.4.2 도커 컨테이스 서비스 확인

curl 명령어를 통해 정상적인 서비스 수행 여부를 확인 합니다.

```
curl localhost:8080
```

[출력]

```
hostname: 96fc3a5eb914
```

1.4.3 도커 프로세서 확인

```
docker ps
```

[출력]

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED
STATUS              PORTS                   NAMES
96fc3a5eb914        goapp                  "/bin/demo"              43 seconds ago
Up 42 seconds       0.0.0.0:8080->8080/tcp   goapp-project
```

1.4.4 프로세서 상세 정보 출력

```
docker inspect goapp-project
```

[출력]

```
[
    {
        "Id": "96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd276c9",
        "Created": "2020-01-10T04:32:12.269012485Z",
        "Path": "/bin/demo",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 31285,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2020-01-10T04:32:12.902395362Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image":
"sha256:36f5c919e3b88f1c991eda67d96e62fe02b763182e44f19fb78b2fc055165f3d",
        "ResolvConfPath":
"/var/lib/docker/containers/96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd27
6c9/resolv.conf",
```

```
        "HostnamePath":
"/var/lib/docker/containers/96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd27
6c9/hostname",
        "HostsPath":
"/var/lib/docker/containers/96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd27
6c9/hosts",
        "LogPath":
"/var/lib/docker/containers/96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd27
6c9/96fc3a5eb914c58ed83e088681d53a46188edeaab061ff2de0b9852e9dd276c9-json.log",
```

1.4.5 Docker 컨테이너 접속

```
# docker 컨테이너 접속
docker exec -it goapp-project bash

# docker 외부에서 컨테이너에 명령수행 (ls 명령 수행)
docker exec goapp-project ls
```

> -i 또는 --interactive : STDIN을 오픈한 상태로 인터랙티브 모드상태
>
> -t 또는 --tty : terminal 모드
>
> container 이름이 보이지 않고 ID 만 기본적으로 보입니다. 컨테이너 이름을 출력하려면 아래 명령어를 수행 하면 됩니다.
>
> docker ps --format "{{.Names}}"

1.4.6 도커 인스턴스 중단 및 삭제

```
docker stop goapp-project
docker rm goapp-project
```

1.5. 도커 이미지를 도커 허브에 업로드

1.5.1 도커 허브양식에 맞게 tag 수정하기

```
docker tag goapp  dangtong/goapp
```

[출력]

```
REPOSITORY                          TAG                 IMAGE ID            CREATED
        SIZE
dangtong/goapp                      latest              12e9a84d9e23        3 days ago
       6.51MB
goapp                               latest              12e9a84d9e23        3 days ago
       6.51MB
```

> dangtong/firstapp 과 first-container 의 image ID 가 같은 것을 확인 할 수 있습니다.
>
> 사실 하나의 이미지를 서로 다른 TAGID 로 공유 하는 것입니다. 디스크 공간이 늘어나지 않습니다.

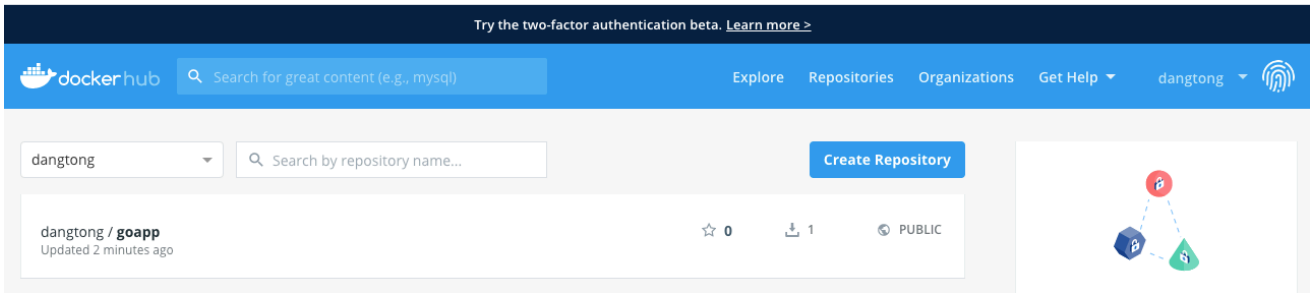
1.5.2 도커 허브에 이미지 업로드 하기

```
docker login --username dangtong
docker push dangtong/goapp
```

[출력]

```
The push refers to repository [docker.io/dangtong/goapp]
cc282a374c26: Pushed
latest: digest: sha256:b18b5ff03599893a7361feda054ebe26de61a71f019dc8725bb33d87f2115968
size: 528
```

도커 허브에 로그인 하게 되면 아래와 같이 goapp 이미지가 업로드 된것을 확인 할 수 있습니다. 이제 인터넷만 연결 되면 어디서는 자신이 만든 이미지로 컨테이너를 실행 할 수 있습니다.



1.5.3 도커 허브의 이미지로 컨테이너 실행

docker hub 에 있는 이미지를 로딩하여 컨테이너 생성

```
docker run --name goapp-project -p 8080:8080 -d dangtong/goapp
```

```
docker ps
```

[출력]

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED
STATUS              PORTS                   NAMES
0938068f8709        dangtong/goapp         "/bin/demo"              5 seconds ago
Up 4 seconds        0.0.0.0:8080->8080/tcp  goapp-project
```

현재 서비스에 접속하여 확인

```
curl http://localhost:8080
```

```
hostname: 0938068f8709
```

[연습문제 #1]

1. 서버 호스트명을 출력하는 node.js 프로그램을 만드세요
2. 해당 소스로 node 버전 7 기반으로 서비스 하는 Docker file을 만들고 이미지를 build 하세요
3. tag 명령을 이용해서 docker hub 에 올릴수 있도록 이름을 바꾸세요
4. docker login 후에 docker hub 에 업로드하여 실제 업로드가 되었는지 확인하세요

- node.js 프로그램

```
const http = require('http');
const os = require('os');

console.log("Kubia server starting...");

var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

- Dcokerfile

```
# FROM 으로 BASE 이미지 로드
FROM node:7

# ADD 명령어로 이미지에 app.js 파일 추가
ADD app.js /app.js

# ENTRYPOINT 명령어로 node 를 실행하고 매개변수로 app.js 를 전달
ENTRYPOINT ["node", "app.js"]
```

2. 쿠버네티스 간단하게 맛보기

2.1 도커 허브 이미지로 컨테이너 생성 및 확인

- 컨테이너 생성 : run/v1 으로 수행 합니다.

```
# POD 및 Replication Controller 생성 (향후 버전에서 deprecated 될 예정)
$ kubectl run goapp-project --image=dangtong/goapp --port=8080 --generator=run/v1
# POD 만 생성
$ kubectl run goapp-project --image=dangtong/goapp --port=8080 --generator=run-pod/v1
```

> generator 를 run/v1 으로 수행 할 경우 내부적으로 goapp-project-{random-String} 이라는 컨테이너를 만들면서 goapp-project- 이름의 replication controller 도 생기게 됩니다.

  - Generator 의 종류 : **run-pod/v1** 외에 모두 deprecated 될 예정

Resource	API group	kubectl command
Pod	v1	`kubectl run --generator=run-pod/v1`
ReplicationController *(deprecated)*	v1	`kubectl run --generator=run/v1`
Deployment *(deprecated)*	extensions/v1beta1	`kubectl run --generator=deployment/v1beta1`
Deployment *(deprecated)*	apps/v1beta1	`kubectl run --generator=deployment/apps.v1beta1`
Job *(deprecated)*	batch/v1	`kubectl run --generator=job/v1`
CronJob *(deprecated)*	batch/v2alpha1	`kubectl run --generator=cronjob/v2alpha1`
CronJob *(deprecated)*	batch/v1beta1	`kubectl run --generator=cronjob/v1beta1`

- 컨테이너 확인

```
$ kubectl get pods
$ kubectl get rc
```

```
$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
goapp-project-bcv5q   1/1     Running   0          2m26s

$ kubectl get rc
NAME            DESIRED   CURRENT   READY   AGE
goapp-project   1         1         1       8m58s
```

> 아래 명령어를 추가적으로 수행해 보세요
>
> ```
> kubectl get pods -o wide
> kubectl describe pod goapp-project-bcv5q
> ```

- k8s 서비스 생성

```
$ kubectl expose rc goapp-project --type=LoadBalancer --name goapp-http
```

```
service/firstapp-http exposed
```

- 생성한 서비스 조회

```
$ kubectl get services
```

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
goapp-http   LoadBalancer   10.96.225.172   <pending>     8080:31585/TCP   104s
kubernetes   ClusterIP      10.96.0.1       <none>        443/TCP          14d
```

- 서비스 테스트 (여러번 수행)

```
curl http://10.96.225.172:8080
```

[출력]

```
hostname: goapp-project-bcv5q
hostname: goapp-project-bcv5q
hostname: goapp-project-bcv5q
hostname: goapp-project-bcv5q
```

- replication controller 를 통한 scale-out 수행

```
kubectl scale rc goapp-project --replicas=3
```

[출력]

```
replicationcontroller/goapp-project scaled
```

- Scale-Out 결과 확인

```
kubectl get pod
```

[출력]

```
NAME                 READY   STATUS    RESTARTS   AGE
goapp-project-bcv5q   1/1     Running   0          16m
goapp-project-c8hml   1/1     Running   0          26s
goapp-project-r7kx5   1/1     Running   0          26s
```

- 서비스 테스트 (여러번 수행)

```
curl http://10.96.225.172:8080
```

[출력]

```
NAME                  READY   STATUS    RESTARTS   AGE
goapp-project-bcv5q   1/1     Running   0          16m
goapp-project-c8hml   1/1     Running   0          26s
goapp-project-r7kx5   1/1     Running   0          26s
```

- POD 삭제

Replication Controller 를 통해 생성된 POD 는 개별 POD 가 삭제 되지 않습니다. Replication Controller 자체를 삭제 해야 합니다.

```
kubectl delete rc goapp-project
```

[연습문제]

3. PODS

3.1 POD 기본

3.1.1 POD 설정을 yaml 파일로 가져오기

```
kubectl get pod goapp-project-bcv5q -o yaml
kubectl get po goapp-project-bcv5q -o json
```

크게 **metadata, spec, status** 항목으로 나누어 집니다.

[출력 - yaml]

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2020-01-10T07:37:49Z"
  generateName: goapp-project-
  labels:
    run: goapp-project
  name: goapp-project-bcv5q
  namespace: default
  ownerReferences:
```

```
  - apiVersion: v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicationController
    name: goapp-project
    uid: e223237f-17e2-44f4-aaee-07e8ff4592b2
  resourceVersion: "3082141"
  selfLink: /api/v1/namespaces/default/pods/goapp-project-bcv5q
  uid: 72bebcbd-8e6e-4906-9f66-1c3821fa49ec
spec:
  containers:
  - image: dangtong/goapp
    imagePullPolicy: Always
    name: goapp-project
    ports:
    - containerPort: 8080
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-qz4fh
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: worker01.sas.com
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: default-token-qz4fh
    secret:
      defaultMode: 420
      secretName: default-token-qz4fh
status:
```

```
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T07:37:49Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T07:37:55Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T07:37:55Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T07:37:49Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID:
docker://de3418e51fea7f7a19e7b1d6ff5a4a6f386f337578838e6f1ec0589275ca6f48
    image: dangtong/goapp:latest
    imageID: docker-
pullable://dangtong/goapp@sha256:e5872256539152aecd2a8fb1f079e132a6a8f247c7a2295f0946ce20
05e36d05
    lastState: {}
    name: goapp-project
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2020-01-10T07:37:55Z"
  hostIP: 192.168.56.103
  phase: Running
  podIP: 10.40.0.2
  podIPs:
  - ip: 10.40.0.2
  qosClass: BestEffort
  startTime: "2020-01-10T07:37:49Z"
```

3.1.2 POD 생성을 위한 YAML 파일 만들기

아래와 같이 goapp.yaml 파일을 만듭니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: goapp-pod
spec:
  containers:
  - image: dangtong/goapp
    name: goapp-container
    ports:
    - containerPort: 8080
      protocol: TCP
```

> ports 정보를 yaml 파일에 기록 하지 않으면 아래 명령어로 향후에 포트를 할당해도 됩니다.
>
> ```
> kubectl port-forward goapp-pod 8080:8080
> ```

3.1.3 YAML 파일을 이용한 POD 생성 및 확인

```
$ kubectl create -f goapp.yaml
```

[output]

```
pod/goapp-pod created
```

```
$ kubectl get pod
```

[output]

```
NAME                  READY   STATUS    RESTARTS   AGE
goapp-pod             1/1     Running   0          12m
goapp-project-bcv5q   1/1     Running   0          41m
goapp-project-c8hml   1/1     Running   0          25m
goapp-project-r7kx5   1/1     Running   0          25m
```

3.1.4 POD 및 Container 로그 확인

- POD 로그 확인

```
kubectl logs goapp-pod
```

[output]

```
Starting GoApp Server......
```

- Container 로그 확인

```
kubectl logs goapp-pod -c goapp-container
```

[output]

```
Starting GoApp Server......
```

> 현재 1개 POD 내에 Container 가 1이기 때문에 출력 결과는 동일 합니다. POD 내의 Container 가 여러개 일 경우 모든 컨테이너의 표준 출력이 화면에 출력됩니다.

3.2 Lable

3.2.1. Lable 정보를 추가해서 POD 생성하기

- goapp-with-lable.yaml 이라 파일에 아래 내용을 추가 하여 작성 합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: goapp-pod2
  labels:
    env: prod
spec:
  containers:
  - image: dangtong/goapp
    name: goapp-container
    ports:
    - containerPort: 8080
      protocol: TCP
```

- yaml 파일을 이용해 pod 를 생성 합니다.

```
$ kubectl create -f ./goapp-with-lable.yaml
```

[output]

```
pod/goapp-pod2 created
```

- 생성된 POD를 조회 합니다.

```
kubectl get po --show-labels
```

[output]

```
NAME                  READY   STATUS    RESTARTS   AGE     LABELS
goapp-pod             1/1     Running   0          160m    <none>
goapp-pod2            1/1     Running   0          3m53s   env=prod
goapp-project-bcv5q   1/1     Running   0          9h      run=goapp-project
goapp-project-c8hml   1/1     Running   0          9h      run=goapp-project
goapp-project-r7kx5   1/1     Running   0          9h      run=goapp-project
```

- Lable 태그를 출력 화면에 컬럼을 분리해서 출력

```
kubectl get pod -l env
```

[output]

```
NAME                  READY   STATUS    RESTARTS   AGE     ENV
goapp-pod             1/1     Running   0          161m
goapp-pod2            1/1     Running   0          5m19s   prod
goapp-project-bcv5q   1/1     Running   0          9h
goapp-project-c8hml   1/1     Running   0          9h
goapp-project-r7kx5   1/1     Running   0          9h
```

- Lable을 이용한 필터링 조회

```
kubectl get pod -l env=prod
```

[output]

```
NAME         READY   STATUS    RESTARTS   AGE
goapp-pod2   1/1     Running   0          39h
```

3.2.2 생성된 POD 로 부터 yaml 파일 얻기

```
kubectl get pod goapp-pod -o yaml
```

[output]

```
apiVersion: v1
kind: Pod
metadata:
```

```
  creationTimestamp: "2020-01-10T08:07:04Z"
  name: goapp-pod
  namespace: default
  resourceVersion: "3086366"
  selfLink: /api/v1/namespaces/default/pods/goapp-pod
  uid: 18cf0ed0-be56-4b54-869c-4473117800b1
spec:
  containers:
  - image: dangtong/goapp
    imagePullPolicy: Always
    name: goapp-container
    ports:
    - containerPort: 8080
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-qz4fh
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: worker02.sas.com
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: default-token-qz4fh
    secret:
      defaultMode: 420
      secretName: default-token-qz4fh
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T08:07:04Z"
```

```
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T08:07:09Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T08:07:09Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2020-01-10T08:07:04Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID:
docker://d76af359c556c60d3ac1957d7498513f42ace14998c763456190274a3e4a1d5e
    image: dangtong/goapp:latest
    imageID: docker-
pullable://dangtong/goapp@sha256:e5872256539152aecd2a8fb1f079e132a6a8f247c7a2295f0946ce20
05e36d05
    lastState: {}
    name: goapp-container
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2020-01-10T08:07:08Z"
  hostIP: 10.0.2.5
  phase: Running
  podIP: 10.32.0.4
  podIPs:
  - ip: 10.32.0.4
  qosClass: BestEffort
  startTime: "2020-01-10T08:07:04Z"
```

3.2.3 Lable을 이용한 POD 스케줄링

- 노드목록 조회

```
kubectl get nodes
```

[output]

```
NAME               STATUS   ROLES    AGE   VERSION
master.sas.com     Ready    master   16d   v1.17.0
worker01.sas.com   Ready    <none>   16d   v1.17.0
worker02.sas.com   Ready    <none>   16d   v1.17.0
```

- 특정 노드에 레이블 부여

```
kubectl label node worker02.sas.com memsize=high
```

- 레이블 조회 필터 사용하여 조회

```
kubectl get nodes -l memsize=high
```

[output]

```
NAME               STATUS   ROLES    AGE   VERSION
worker02.sas.com   Ready    <none>   17d   v1.17.0
```

- 특정 노드에 신규 POD 스케줄링

  아래 내용과 같이 goapp-label-node.yaml 파을을 작성 합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: goapp-pod-memhigh
spec:
  nodeSelector:
    memsize: "high"
  containers:
  - image: dangtong/goapp
    name: goapp-container-memhigh
```

- YAML 파일을 이용한 POD 스케줄링

```
kubectl create -f ./goapp-lable-node.yaml
```

[output]

```
pod/goapp-pod-memhigh created
```

- 생성된 노드 조회

```
kubectl get pod -o wide
```

[output]

```
NAME                  READY   STATUS    RESTARTS   AGE    IP          NODE
NOMINATED NODE   READINESS GATES
goapp-pod-memhigh     1/1     Running   0          17s    10.32.0.5   worker02.sas.com
<none>           <none>
```

3.3 Annotation

3.3.1 POD 에 Annotation 추가하기

```
kubectl annotate pod goapp-pod-memhigh maker="dangtong" team="k8s-team"
```

[output]

```
pod/goapp-pod-memhigh annotated
```

3.3.2 Annotation 확인하기

- YAML 파일을 통해 확인하기

```
kubectl get po goapp-pod-memhigh -o yaml
```

[output]

```
kind: Pod
metadata:
  annotations:
    maker: dangtong
    team: k8s-team
  creationTimestamp: "2020-01-12T15:25:05Z"
  name: goapp-pod-memhigh
  namespace: default
  resourceVersion: "3562877"
  selfLink: /api/v1/namespaces/default/pods/goapp-pod-memhigh
  uid: a12c35d7-d0e6-4c01-b607-cccd267e39ec
spec:
  containers:
```

- DESCRIBE 를 통해 확인하기

```
kubectl describe pod goapp-pod-memhigh
```

[output]

```
Name:         goapp-pod-memhigh
Namespace:    default
Priority:     0
Node:         worker02.sas.com/10.0.2.5
Start Time:   Mon, 13 Jan 2020 00:25:05 +0900
Labels:       <none>
Annotations:  maker: dangtong
              team: k8s-team
Status:       Running
IP:           10.32.0.5
```

3.3.3 Annotation 삭제

```
kubectl annotate pod  goapp-pod-memhigh maker=make- team-
```

3.4 Namespace

3.4.1 네이스페이스 조회

```
kubectl get namespace
```

> kubectl get ns 와 동일함

[output]

```
NAME              STATUS   AGE
default           Active   17d
kube-node-lease   Active   17d
kube-public       Active   17d
kube-system       Active   17d
```

3.4.2 특정 네임스페이스의 POD 조회

```
kubectl get pod --namespace kube-system
```

> kubectl get pod -n kube-system 과 동일함

[output]

```
coredns-6955765f44-glcdc                 1/1     Running   0          17d
coredns-6955765f44-h7fbb                 1/1     Running   0          17d
etcd-master.sas.com                      1/1     Running   1          17d
kube-apiserver-master.sas.com            1/1     Running   1          17d
kube-controller-manager-master.sas.com   1/1     Running   1          17d
kube-proxy-gm44f                         1/1     Running   1          17d
kube-proxy-ngqr6                         1/1     Running   0          17d
kube-proxy-wmq7d                         1/1     Running   0          17d
kube-scheduler-master.sas.com            1/1     Running   1          17d
weave-net-2pm2x                          2/2     Running   0          17d
weave-net-4wksv                          2/2     Running   0          17d
weave-net-7j7mn                          2/2     Running   0          17d
```

3.4.3 YAML 파일을 이용한 네임스페이스 생성

- YAML 파일 작성 : first-namespace.yaml 이름으로 파일 작성

```
apiVersion: v1
kind: Namespace
metadata:
  name: first-namespace
```

- YAML 파일을 이용한 네이스페이스 생성

```
kubectl create -f first-namespace.yaml
```

[output]

```
namespace/first-namespace created
```

- 생성된 네임스페이스 확인

```
kubectl get namespace
kubectl get ns
```

[output]

```
NAME              STATUS   AGE
default           Active   17d
first-namespace   Active   5s
kube-node-lease   Active   17d
kube-public       Active   17d
kube-system       Active   17d
```

> kubectl create namespace first-namespace 와 동일 합니다.

3.4.4 특정 네임스페이스에 POD 생성

- first-namespace 에 goapp 생성

```
kubectl create -f goapp.yaml -n first-namespace
```

[output]

```
pod/goapp-pod created
```

- 생성된 POD 확인하기

```
kubectl get pod -n first-namespace
```

[output]

```
NAME        READY   STATUS    RESTARTS   AGE
goapp-pod   1/1     Running   0          12h
```

3.4.5 POD 삭제

```
kubectl delete pod goapp-pod-memhigh
```

```
kubectl delete pod goapp-pod
```

```
kubectl delete pod goapp-pod -n first-namespace
```

> 현재 네임스페이스 에서 존재 하는 모든 리소스를 삭제하는 명령은 아래와 같습니다.

> kubectl delete all --all
>
> 현재 네임스페이스를 설정하고 조회 하는 명령은 아래와 같습니다.

```
# 네임스페이스 설정
kubectl config set-context --current --namespace=<insert-namespace-name-here>
# 확인
kubectl config view --minify | grep namespace:
```

4. kubectl 기본 사용법

4.1 단축형 키워드 사용하기

```
kubectl get po      # PODs
kubectl get svc     # Service
kubectl get rc      # Replication Controller
kubectl get deploy  # Deployment
kubectl get ns      # Namespace
kubectl get no      # Node
kubectl get cm      # Configmap
kubectl get pv      # PersistentVolumns
```

4.2 도움말 보기

```
kubectl -h
```

```
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service, deployment or pod and expose it
as a new Kubernetes Service
  run            Run a particular image on the cluster
  set            Set specific features on objects
```

```
Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
  edit           Edit a resource on the server
  delete         Delete resources by filenames, stdin, resources and names, or by
resources and label selector

Deploy Commands:
```

```
kubectl get -h
```

```
Display one or many resources

 Prints a table of the most important information about the specified resources. You can
filter the list using a label
selector and the --selector flag. If the desired resource type is namespaced you will
only see results in your current
namespace unless you pass --all-namespaces.

 Uninitialized objects are not shown unless --include-uninitialized is passed.

 By specifying the output as 'template' and providing a Go template as the value of the -
-template flag, you can filter
the attributes of the fetched resources.

Use "kubectl api-resources" for a complete list of supported resources.

Examples:
  # List all pods in ps output format.
  kubectl get pods

  # List all pods in ps output format with more information (such as node name).
  kubectl get pods -o wide
```

4.3 리소스 정의에 대한 도움말

```
kubectl explain pods
```

```
KIND:     Pod
VERSION:  v1

DESCRIPTION:
```

```
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources

   kind <string>
     Kind is a string value representing the REST resource this object
     represents. Servers may infer this from the endpoint the client submits
     requests to. Cannot be updated. In CamelCase. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#types-kinds

   metadata <Object>
     Standard object's metadata. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#metadata

   spec <Object>
     Specification of the desired behavior of the pod. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#spec-and-status

   status <Object>
     Most recently observed status of the pod. This data may not be up to date.
     Populated by the system. Read-only. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#spec-and-status
```

4.4 리소스 감시하기

- Kube-system 네임스페이스에 있는 모든 pod에 대해 모니터링 합니다.

```
kubectl get pods --watch -n kube-system
```

```
root@master:~# k get pods --watch -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
coredns-6955765f44-glcdc              1/1     Running   0          19d
coredns-6955765f44-h7fbb              1/1     Running   0          19d
etcd-master.sas.com                   1/1     Running   1          19d
kube-apiserver-master.sas.com         1/1     Running   1          19d
```

```
kube-controller-manager-master.sas.com   1/1     Running   1          19d
kube-proxy-gm44f                         1/1     Running   1          19d
kube-proxy-ngqr6                         1/1     Running   0          19d
kube-proxy-wmq7d                         1/1     Running   0          19d
kube-scheduler-master.sas.com            1/1     Running   1          19d
weave-net-2pm2x                          2/2     Running   0          19d
weave-net-4wksv                          2/2     Running   0          19d
weave-net-7j7mn                          2/2     Running   0          19d
...
```

4.5 리소스 비교하기

```
kubectl diff -f goapp.yaml
```

4.6 kubectx 및 kubens 사용하기

현재 컨텍스트 및 네임스페이스를 확인하고 전환 할때 손쉽게 사용 할수 있는 도구

4.6.1 kubectx 및 kubens 설치

```
git clone https://github.com/ahmetb/kubectx.git ~/.kubectx
COMPDIR=$(pkg-config --variable=completionsdir bash-completion)
ln -sf ~/.kubectx/completion/kubens.bash $COMPDIR/kubens
ln -sf ~/.kubectx/completion/kubectx.bash $COMPDIR/kubectx
cat << FOE >> ~/.bashrc


#kubectx and kubens
export PATH=~/.kubectx:\$PATH
FOE
```

4.6.2 kubectx 및 kubens 사용

- kubectx 사용

4.6 로그 확인

```
kubectl logs
```

4. 콘트롤러(Controller)

4.1 liveness probes

4.2 Replication Controller

5. 서비스 (Service)

6. 볼륨

Appendix

Appendix 1. 쿠버네티스 설치

1.설치 사전 작업

1.1 Nftable disable

```
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
sudo update-alternatives --set arptables /usr/sbin/arptables-legacy
sudo update-alternatives --set ebtables /usr/sbin/ebtables-legacy
```

1.2 호스트명 변경

```
# hostnamectl set-hostname [FQDN-HOST-NAME] --static
hostnamectl set-hostname master
hostnamectl set-hostname master.sas.com --static

hostnamectl set-hostname worker1
hostnamectl set-hostname worker1.sas.com --static

hostnamectl set-hostname worker2
hostnamectl set-hostname worker2.sas.com --static
```

1.3 각노드에 호스트 파일 등록

```
172.31.43.23 master.sas.com master
172.31.42.202 worker1.sas.com worker1
172.31.32.175 worker2.sas.com worker2
```

1.4 방화벽 점검

1.4.1 Control Plane 노드

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	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

1.4.2 worker 노드

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

2. Runtime 설치

2.1 도커 설치 (all nodes)

```
sudo apt update
sudo apt install docker.io
```

2.2 kubernetes 설치 (all nodes)

설치 스크립트 클릭

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

3. Post Installation

3.1 SWAP 기능 해제(all nodes)

```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

3.2 kubeadm을 사용하여 마스터 설정 (master node)

```
sudo kubeadm init
```

4. 워커 노드 등록

4.1 워커 노드 등록 (worker nodes)

> 반듯이 kubeadm init 을 통해 나온 로그에 표시된 ip 와 token 으로 worker 노드에서만 수행

```
kubeadm join 10.0.2.15:6443 --token ia7jd7.ma3f71fnvkzlyye0 \
    --discovery-token-ca-cert-hash
sha256:e011a1f415cf272a11190971fe697b0a415a578fb71482ba7e40d4b146cc4540
```

4.2 마스터 노드 kube client 설정

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

4.3 클러스터 상태 확인

결과가 모두 not ready 로 나옴

```
kubectl get nodes
```

4.4 네트워크 어플리케이션 설치

네트워크 툴 설치 후에는 모든 노드가 ready 로 나와야 함

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version |
base64 | tr -d '\n')"

kubectl get nodes
```

4.5 kubectl 명령어 자동완성 패키지 설치

```
source <(kubectl completion bash) # bash-completion 패키지를 먼저 설치한 후, bash의 자동 완성을 현재 셸에 설정한다
echo "source <(kubectl completion bash)" >> ~/.bashrc # 자동 완성을 bash 셸에 영구적으로 추가한다
alias k=kubectl
complete -F __start_kubectl k
```

5. etcd 설치

기본적으로 내장 etcd 가 있지만, etcd-io 의 릴리즈로 설치 합니다.

다운로드 url : 클릭

```
wget https://github.com/etcd-io/etcd/releases/download/v3.3.18/etcd-v3.3.18-linux-amd64.tar.gz
tar -xf etcd-v3.3.18-linux-amd64.tar.gz
cd etcd-v3.3.18-linux-amd64

sudo ETCDCTL_API=3 ./etcdctl --endpoints 127.0.0.1:2379 \
--cacert /etc/kubernetes/pki/etcd/ca.crt \
--cert /etc/kubernetes/pki/etcd/server.crt \
--key /etc/kubernetes/pki/etcd/server.key \
get / --prefix --keys-only
```

Appendix 2. 구글 GKE 사용하기

Appendix 3.