클라우드 환경에서 컨테이너 서비스 구현

어때요 갖고싶조

조원 구성

김선경

오정환

이현찬

이창한

허윤하

목 차

Project 계획

SUN	MON	TUE	WED	THU	FRI	SAT
						17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

주제 선정 및 역할 분담

Local 환경 구축 및 Image 생성

Cloud 환경

Kops 구축

Cloud 환경 Cluster 및 Node 생성

테스트

PPT 작성 및 발표 준비

발표

시나리오

(주)어때컴퍼니는 클라우드에서 웹 모바일 서버를 운영 중인데,

개발 및 배포 시간이 오래 걸리는 이슈가 있습니다.

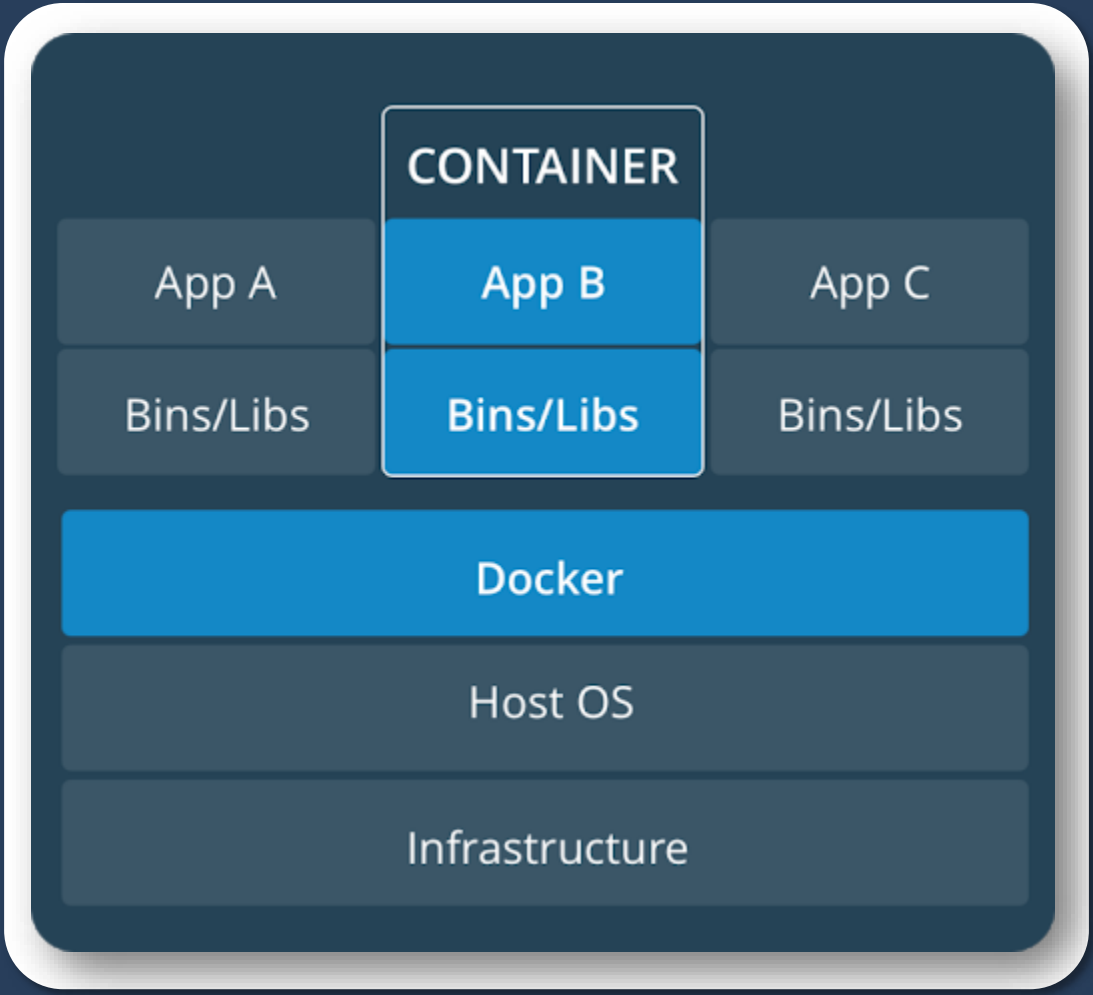
해당 업체에서 이를 단축시키는 방법으로 컨테이너 서비스를 활용해보고자 하는데,

이에 대한 개념과 어떻게 구축하여 사용하는지를 의뢰해왔습니다.

저희 '어때요 갖고싶조'는 이에 대해 말씀드리고자 합니다.

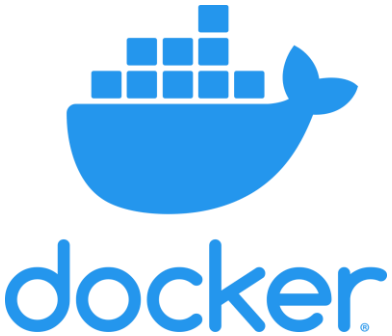
클라우드를 사용하는 이유

컨테이너 서비스



- 호스트 운영체제의 커널을 공유하면서 격리된 컴퓨팅 자원을 제공하는 가상화 기술
- 소프트웨어 서비스를 실행하는데 필요한 특정 버전의 프로그래밍 언어 런타임 및 라이브러리와 같은 종속 항목과 어플리케이션 코드와 함께 포함하는 경량 패키지

컨테이너 서비스



- Docker는 Go 언어로 작성되었으며, 리눅스 컨테이너에 여러 기능을 추가함으로써 어플리케이션을 컨테이너로 사용하기 쉽게 만들어진 오픈소스 프로젝트

- LXC(LinuX Container), 네임스페이스, 제어그룹, SE리눅스 등 20년 가까이 된 기술들을 잘 조합해 새로운 컨셉으로 등장

컨테이너 서비스 장점

쉽고 빠른 구축 및 실행

효율적인 하드웨어 자원 사용

공유 환경 제공 및 쉬운 배포

기업용으로 충분히 활용 가능

컨테이너 서비스 단점

개발 초기의 오버헤드

리눅스 친화적

다양한 하드웨어를 지원하는
호스트 운영체제 선택이 필수

보안에 대한 구멍이 쉽게
노출 가능

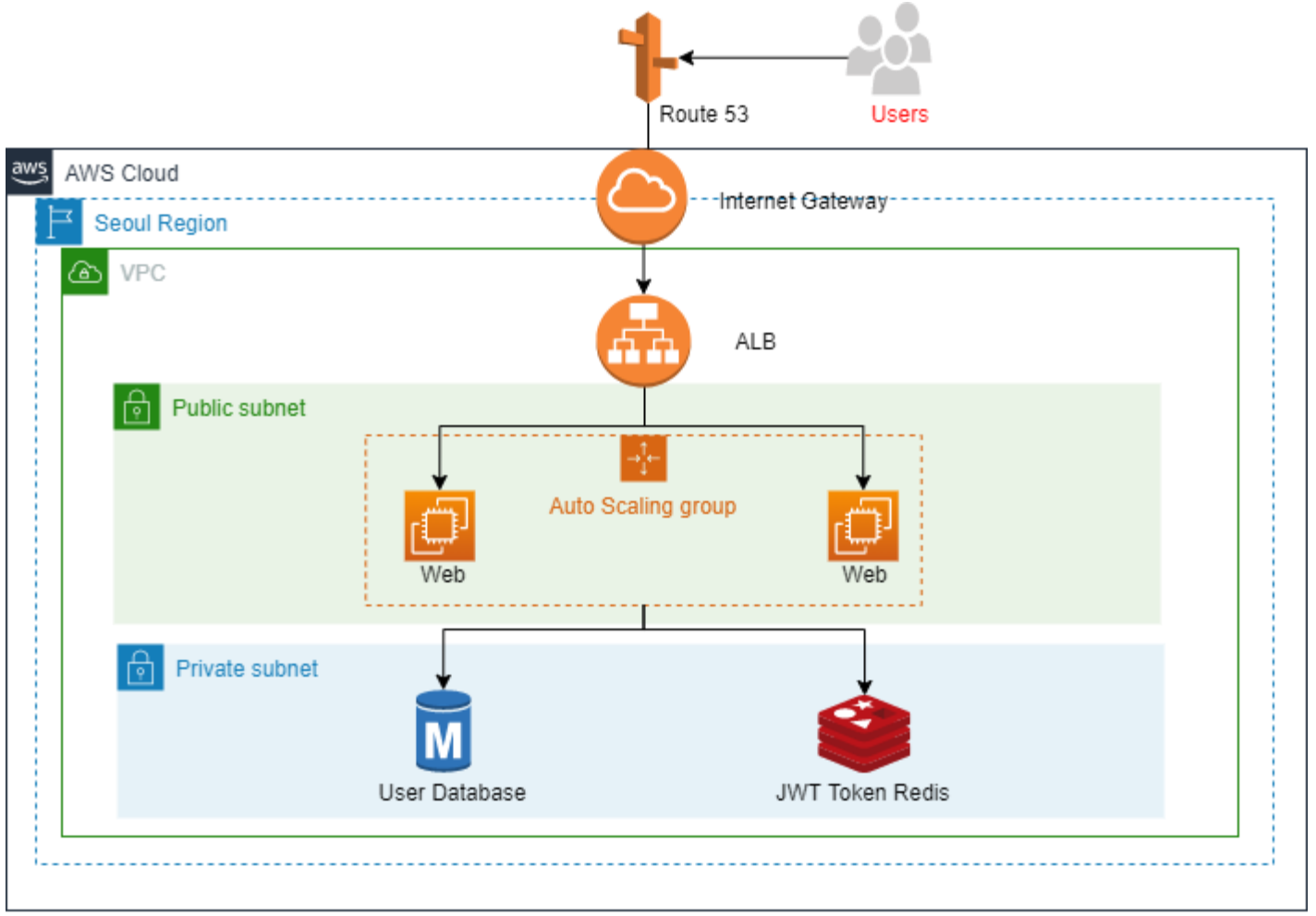
인력을 구하기 어려움

Container Orchestration

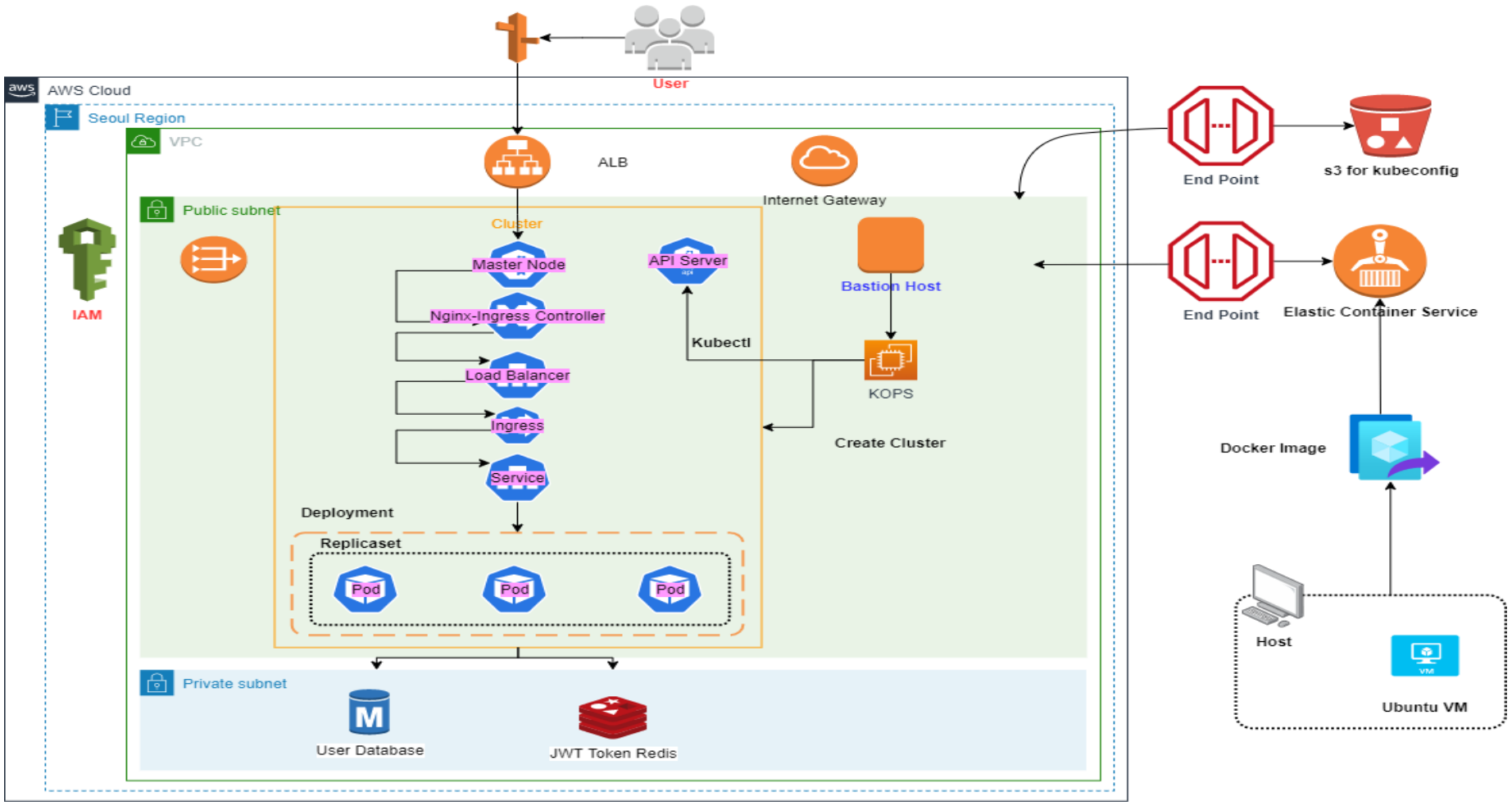
쿠버네티스

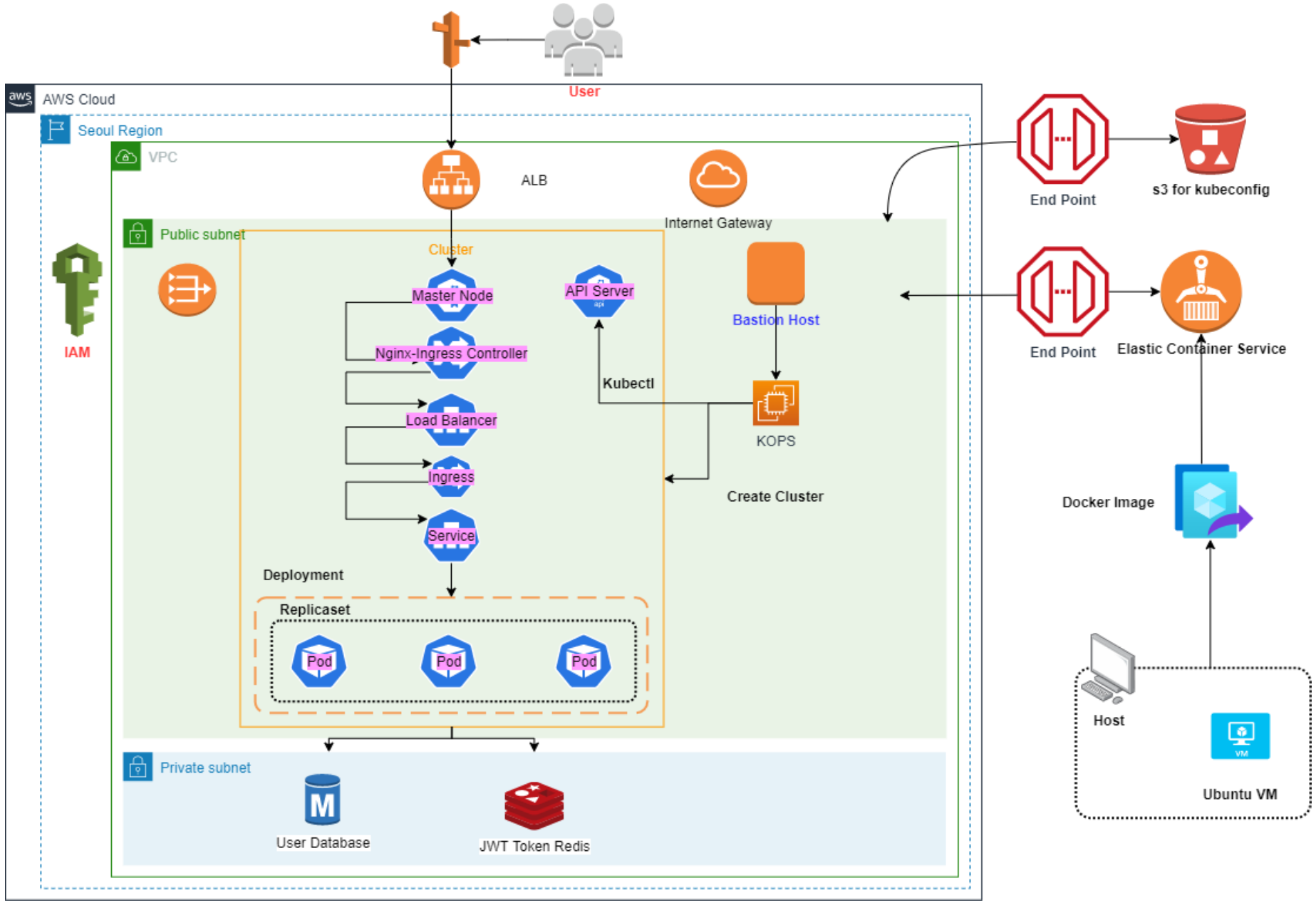
- 컨테이너를 쉽고 빠르게 배포/확장하고 관리를 자동화해주는 컨테이너 오케스트레이션 도구
- 구글에서 2014년 공개한 오픈소스로 많은 회사들이 실제 서비스 운영에 도입해 사용하고 있다.

이전 구성도


Route 53
Users
AWS Cloud
Internet Gateway
Seoul Region
VPC
ALB
Public subnet
Auto Scaling group
Web
Web
Private subnet
User Database
JWT Token Redis

컨테이너 도입 후 구성도


User
AWS Cloud
Seoul Region
VPC
ALB
Internet Gateway
Public subnet
Cluster
Master Node
API Server
Nginx-Ingress Controller
Bastion Host
IAM
Kubectl
Load Balancer
KOPS
Ingress
Create Cluster
Service
Deployment
Replicaset
Pod
Pod
Pod
Private subnet
User Database
JWT Token Redis
End Point
s3 for kubeconfig
End Point
Elastic Container Service
Docker Image
Host
Ubuntu VM


User
AWS Cloud
Seoul Region
VPC
ALB
Internet Gateway
Public subnet
Cluster
Master Node
API Server
Nginx-Ingress Controller
Load Balancer
Ingress
Service
Deployment
Replicaset
Pod
Pod
Pod
IAM
Bastion Host
Kubectl
KOPS
Create Cluster
Private subnet
User Database
JWT Token Redis
End Point
s3 for kubeconfig
End Point
Elastic Container Service
Docker Image
Host
Ubuntu VM

KOPS (Kubernetes Operations)

- 서버 인스턴스와 네트워크 리소스 등을 클라우드에서 자동으로 생성해 쿠버네티스를 설치할 수 있게 도와주는 툴
- 세부적인 리소스에 익숙하지 않더라도 서버 인프라를 프로비저닝 하여 쿠버네티스를 설치할 수 있음

구현 환경 구축하기

```go
package main

import (
    "net/http"
)

func main() {
    http.Handle("/", new(testHandler))

    http.ListenAndServe(":5000", nil)
}

type testHandler struct {
    http.Handler
}

func (h *testHandler) ServeHTTP(w http.ResponseWriter, req *http.Request) {
    str := "Your Request Path is " + req.URL.Path
    w.Write([]byte(str))
}
```

- 로컬

  OS : Ubuntu 20.04 LTS
- Tool

  Vscode

  Awscliv2 2.1.39

  Docker-CE 19.03.15

  Kubectl 1.18.0

  Kops 1.18.0
- Language : Go (1.16)

KOPS (Kubernetes Operations)

--zones : 가용 영역

-- state : cluster 정보를 저장할 S3 버킷

--master-zones : 마스터 노드의 가용영역

--topology : 클러스터를 public/private으로 오픈

--api-loadbalancer-type : 로드밸런서를 public/internal로 오픈

--vpc : 사용할 vpc

--image : 사용할 ami

--networking : overlay network driver를 선택 (calico, weave 등)

--ssh-public-key : ssh 연결에 사용할 키

```
kops create cluster \
  --zones=ap-northeast-2c \
  --state s3://whitewhale-k8s-bucket \
  --master-zones=ap-northeast-2c \
  --node-count=2 \
  --node-size=t3a.large \
  --node-volume-size=8 \
  --master-count=1 \
  --master-size=t3a.large \
  --master-volume-size=8 \
  --topology=public \
  --api-loadbalancer-type=public \
  --vpc=vpc-b5678dde \
  --network-cidr=172.31.0.0/16 \
  --image='ami-00f1068284b9eca92' \
  --networking=calico \
  --cloud-labels "Owner=whitewhale" \
  --ssh-public-key ./id_rsa.pub \
  $NAME
```

```
kops create cluster \
  --zones=ap-northeast-2c \
  --state s3://whitewhale-k8s-bucket \
  --master-zones=ap-northeast-2c \
  --node-count=2 \
  --node-size=t3a.large \
  --node-volume-size=8 \
  --master-count=1 \
  --master-size=t3a.large \
  --master-volume-size=8 \
  --topology=public \
  --api-loadbalancer-type=public \
  --vpc=vpc-b5678dde \
  --network-cidr=172.31.0.0/16 \
  --image='ami-00f1068284b9eca92' \
  --networking=calico \
  --cloud-labels "Owner=whitewhale" \
  --ssh-public-key ./id_rsa.pub \
  $NAME
```

KOPS (Kubernetes Operations)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-webapp-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-webapp
  template:
    metadata:
      name: urmywebapp
      labels:
        app: my-webapp
    spec:
      containers:
      - name: urmywebapp
        image: 088755231083.dkr.ecr.ap-northeast-2.amazonaws.com/webhandler:1.0
        ports:
        - containerPort: 5000
          name: flask-port
      imagePullSecrets:
      - name: regcred
```

deploymentwebapp.yml

Replicas : 사용할 포드 개수

Image : 이미지를 가져올 ECR 주소

containerPort : 컨테이너에서 구동되는 어플리케이션이 사용할 port

Name(port) : 해당 포트의 이름 지정

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-webapp-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-webapp
  template:
    metadata:
      name: urmywebapp
      labels:
        app: my-webapp
    spec:
      containers:
      - name: urmywebapp
        image: 088755231083.dkr.ecr.ap-northeast-2.amazonaws.com/webhandler:1.0
        ports:
        - containerPort: 5000
          name: flask-port
      imagePullSecrets:
      - name: regcred
```

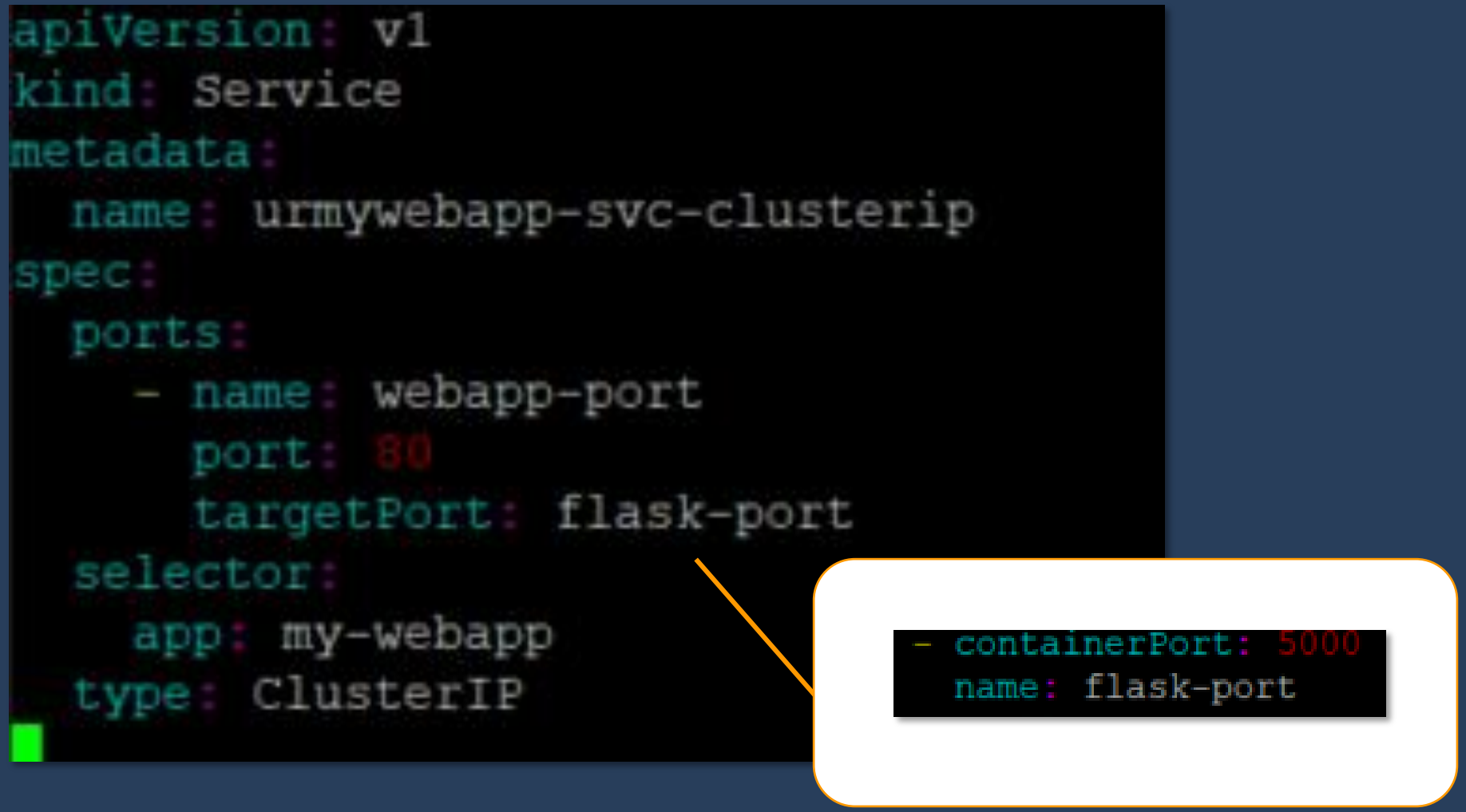
```go
func main() {
    http.Handle("/", new(testHandler))

    http.ListenAndServe(":5000", nil)
}
```

KOPS (Kubernetes Operations)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: urmywebapp-svc-clusterip
spec:
  ports:
    - name: webapp-port
      port: 80
      targetPort: flask-port
  selector:
    app: my-webapp
  type: ClusterIP
```

app-svc.yml

targetPort : Deployment 야믈 파일에서 설정한 port의 이름을 지정하여, Service에서 지정하는 port와 매핑


apiVersion: v1
kind: Service
metadata:
name: urmywebapp-svc-clusterip
spec:
ports:
- name: webapp-port
port: 80
targetPort: flask-port
selector:
app: my-webapp
type: ClusterIP
- containerPort: 5000
name: flask-port

KOPS (Kubernetes Operations)

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: ad554a0f2b99f40608b5f794331c5a5d-2fac0bf235fa8d3c.elb.ap-northeast-2.amazonaws.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
      - path: /register
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
      - path: /friendlist
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
```

host : nginx-ingress를 설치하면서 받아오는 External IP를 설정

Path : 외부로부터 들어오는 요청에 해당 경로가 있는 경우 설정된 서비스의 설정된 포트로 전달

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: ad554a0f2b99f40608b5f794331c5a5d-2fac0bf235fa8d3c.elb.ap-northeast-2.amazonaws.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
      - path: /register
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
      - path: /friendlist
        pathType: Prefix
        backend:
          serviceName: urmywebapp-svc-clusterip
          servicePort: 80
```

KOPS (Kubernetes Operations)

Nginx Ingress Controller

```
~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.46.0/deploy/static/provider/aws/deploy.yaml
```

```
ubuntu@ip-172-31-100-29:~/yaml$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   100.64.247.74   ad554a0f2b99f40608b5f794331c5a5d-2fac0bf235fa8d3c.elb.ap-northeast-2.amazonaws.com   80:32614/TCP,443:30810/TCP   21m
ingress-nginx-controller-admission   ClusterIP      100.66.34.87    <none>                                                                     443/TCP                      21m
```

KOPS (Kubernetes Operations)

```
ubuntu@ip-172-31-100-29:~/yaml$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP      EXTERNAL-IP
ingress-nginx-controller             LoadBalancer   100.64.247.74   ad554a0f2b99f40608b5f794331c5a5d-2fac0b
ingress-nginx-controller-admission   ClusterIP      100.66.34.87    <none>
```

KOPS (Kubernetes Operations)

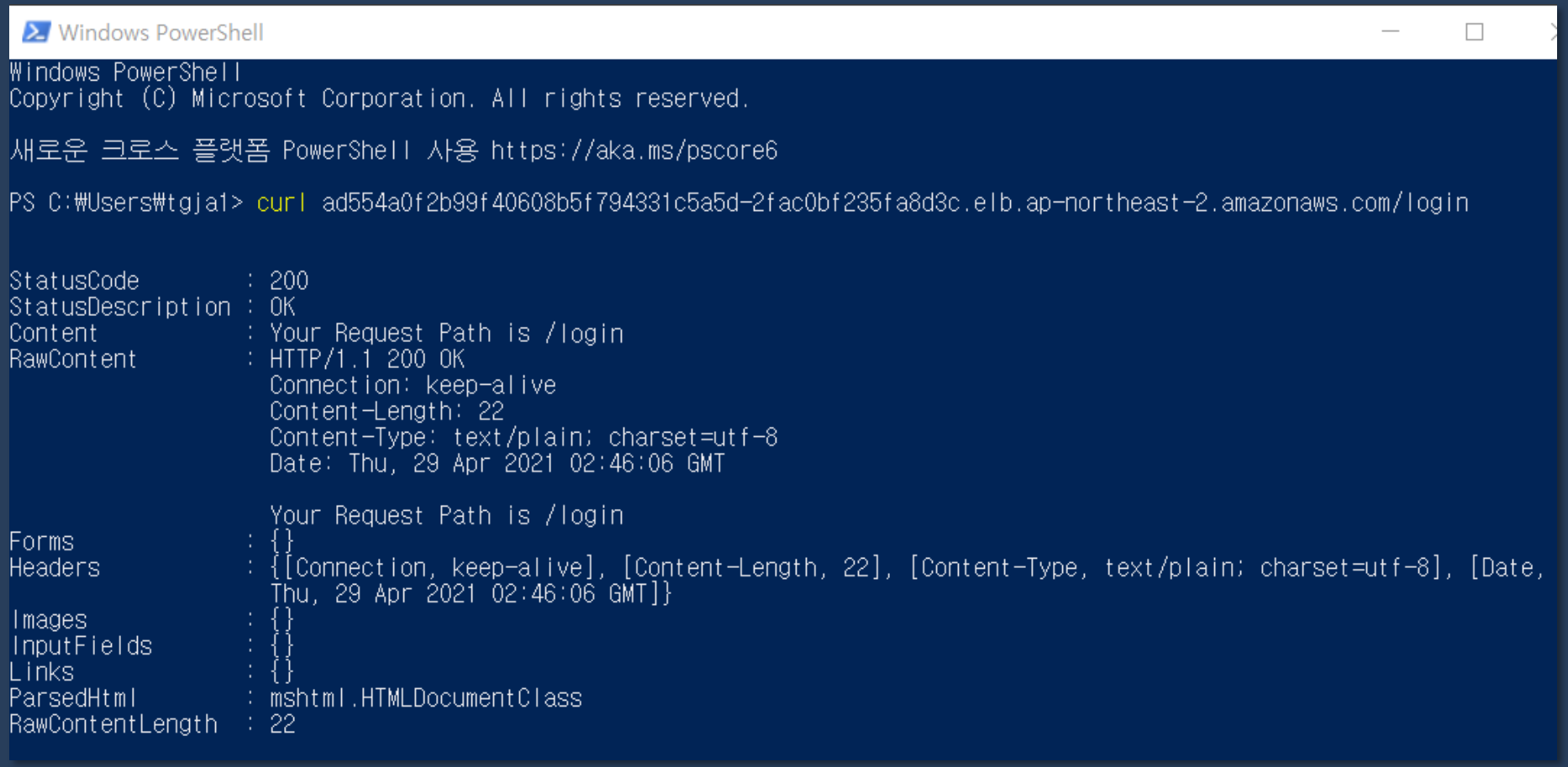
```
                                                                  PORT(S)                      AGE
f40608b5f794331c5a5d-2fac0bf235fa8d3c.elb.ap-northeast-2.amazonaws.com   80:32614/TCP,443:30810/TCP   21m
                                                                  443/TCP                      21m
```

KOPS (Kubernetes Operations)

```
ubuntu@ip-172-31-100-29:~/yaml$ kubectl get pods,services
NAME                                        READY   STATUS    RESTARTS   AGE
pod/my-webapp-deployment-5ff846787b-9rs7p   1/1     Running   0          40m
pod/my-webapp-deployment-5ff846787b-jfs49   1/1     Running   0          40m
pod/my-webapp-deployment-5ff846787b-n85pq   1/1     Running   0          40m

NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/kubernetes                  ClusterIP   100.64.0.1       <none>        443/TCP   53m
service/urmywebapp-svc-clusterip    ClusterIP   100.70.130.237   <none>        80/TCP    40m
```

```
ubuntu@ip-172-31-100-29:~/yaml$ kubectl get pods,deployment -n ingress-nginx
NAME                                            READY   STATUS      RESTARTS   AGE
pod/ingress-nginx-admission-create-r421d        0/1     Completed   0          21m
pod/ingress-nginx-admission-patch-vn46z         0/1     Completed   0          21m
pod/ingress-nginx-controller-6c94f69c74-ddrzj   1/1     Running     0          21m

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           21m
```

KOPS (Kubernetes Operations)



리뷰

- kops를 이용하여 기존에 만들어진 네트워크 환경에 적용시키는 부분 문제.
- bastion host를 통해 kops instance를 Private subnet 에 추가하여 진행했을 때 오류 발생
- Loadbalancer와 node들을 각각 다른 서브넷으로 적용시켰을 때 장애가 발생.
- EKS용 Fargate 구현 시 외부에서 curl 접근 시 접근이 안되는 문제 미해결.


Q & A

감사합니다.