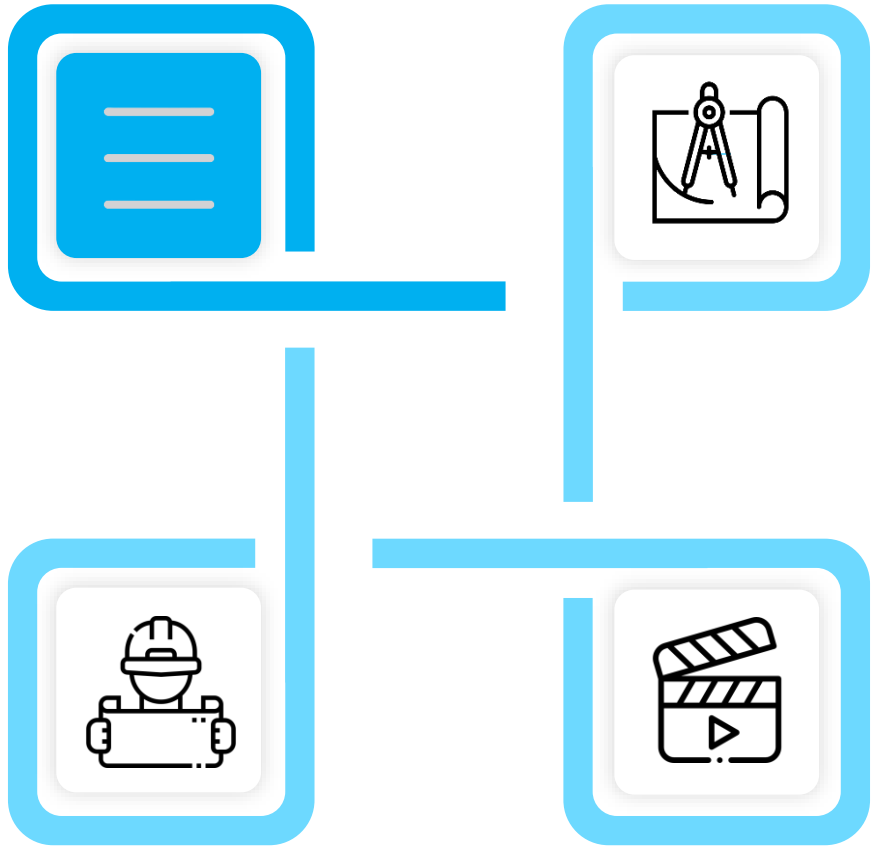
2조 전중석

Jenkins를 활용한 WAS CI/CD

목차



관심있던 기술들을 활용하여

AWS 상에서 **CI/CD**가 가능한 WAS 서버를 배포하여

백앤드 개발자가 **서버 설정을 변경하고 재배포** 하는 과정을 용이하게 함.

Kubernetes

- 컨테이너 오케스트레이션 시스템
- 파드 인프라 컨테이너

Ansible

- 플레이북 기본, 변수, 템플릿
- 플레이북 블록, 역할, 에러처리

AWS

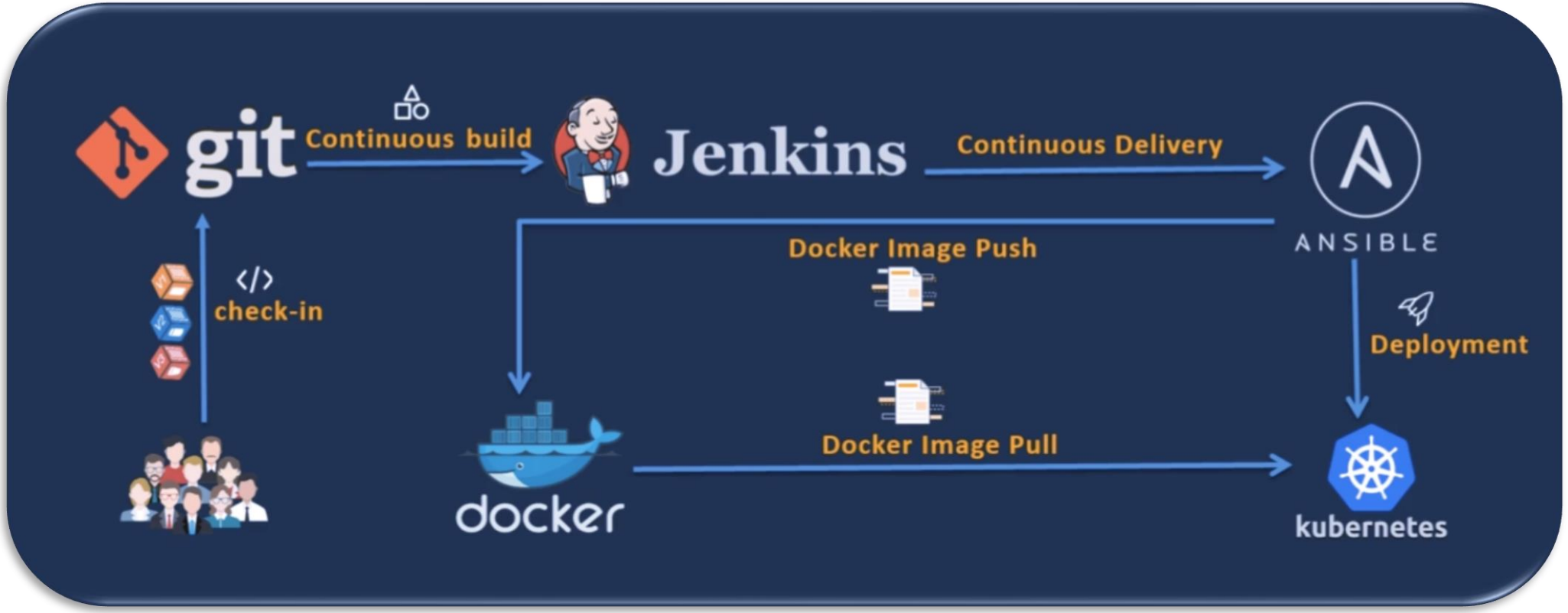
- IAM 정책
- 관리형 Amazon EKS
- S3

Docker

- 명령어를 이용한 Docker 이미지 생성
- 멀티스테이지 빌드를 사용한 애플리케이션 개발
- 데몬 실행, 환경 및 네트워크 설정

Vagrant (Test VM 설정)

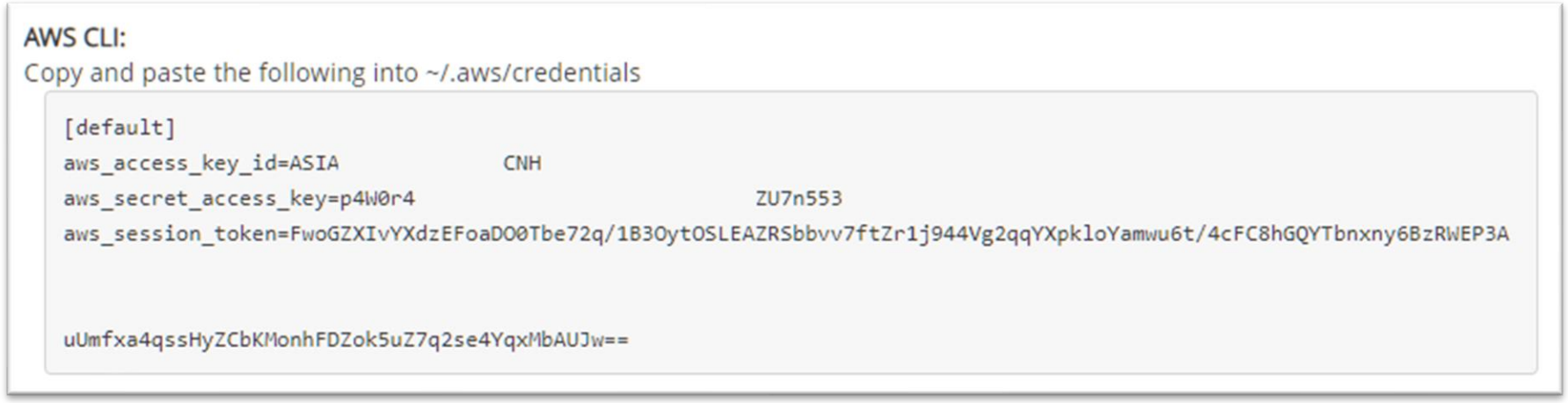
- CPU : 2개
- Memory : 2000MB
- Disk : 20GB
- OS : Ubuntu/focal



CI/CD란 각각의 개발자들의 개발환경을 사용자가 사용 가능한 서비스로 코드를 빌드하고,
테스트하고 배포하는 활동으로 모든 과정을 **지속 가능한 형태**로 **자동화**해서 개발자와 사용자 사이의 격차를 없애준다.

AWS CLI

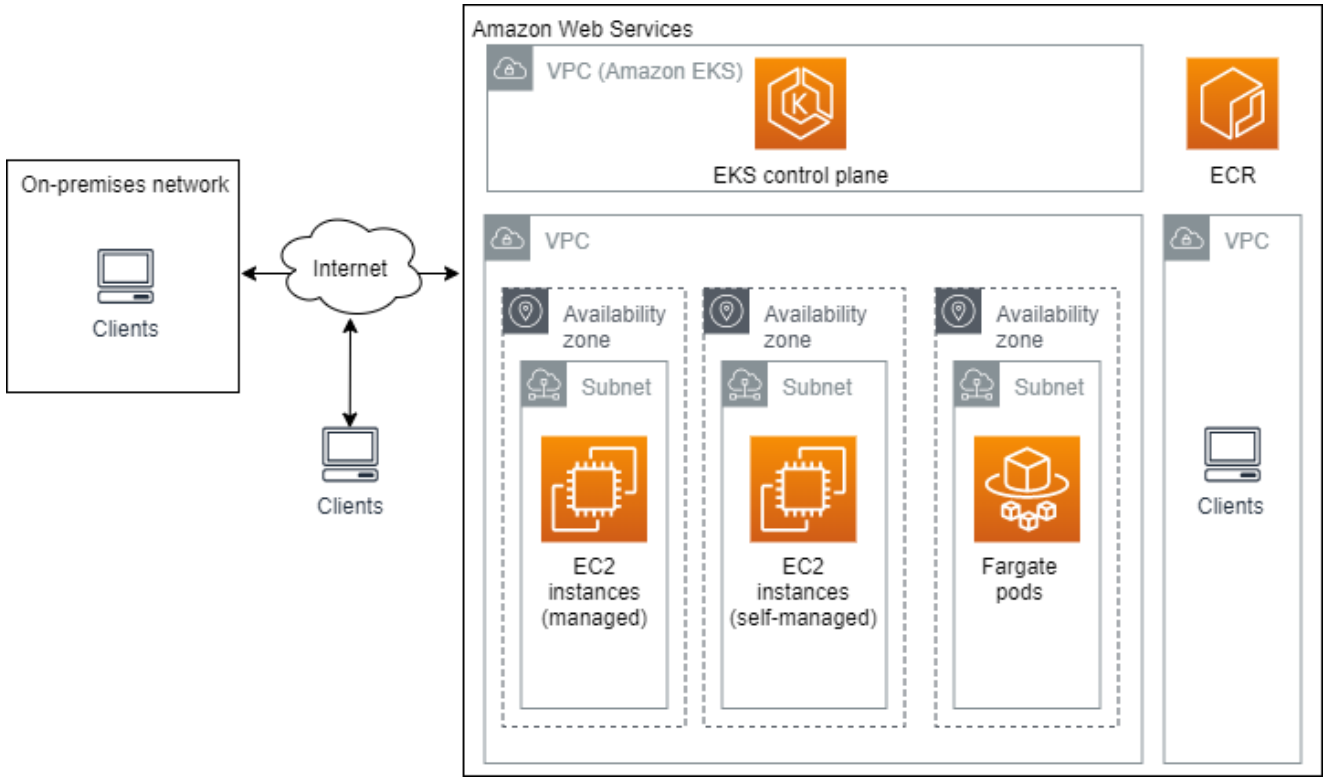
AWS 서비스를 관리하는 통합 도구



여러 AWS 서비스를 **CLI**에서 제어하고, **스크립트**를 통해 자동화

AWS EKS

Amazon Elastic Kubernetes Service는
클라우드 또는 온프레미스에서
Kubernetes 애플리케이션을 실행하고
크기를 조정하는 **관리형 컨테이너 서비스**



Terraform을 활용한 EKS 리소스 생성

Amazon EC2를 Worker 노드로 사용하는
Amazon **EKS 클러스터를 생성**하기 위한 YAML 파일

us-east-1 리전의 a, b, c 가용영역 사용
t3.medium을 사용하여 EC2 배포

```
aws_region      = "us-east-1"
azs             = ["us-east-1a", "us-east-1b", "us-east-1c"]
cidr       = "10.1.0.0/16"
enable_igw = true
enable_ngw = true
single_ngw = true
name            = "myeks"
tags = {
  env  = "dev"
  test = "tc1"
}
kubernetes_version  = "1.22"
enable_ssm          = true
managed_node_groups = [
  {
    name          = "Jenkins"
    min_size      = 1
    max_size      = 1
    desired_size  = 1
    instance_type = "t3.medium"
  },
    {
    name          = "realmytrip"
    min_size      = 1
    max_size      = 3
    desired_size  = 1
    instance_type = "t3.large"
  }
```

YAML 파일을 활용한 EKS 리소스 생성

출처 : AWS Documentation의 Amazon EKS

Amazon EC2를 Worker 노드로 사용하는
Amazon **EKS 클러스터를 생성**하기 위한 YAML 파일

us-east-1 리전의 a, b, c 가용영역 사용
t3.medium을 사용하여 EC2 배포

추가 설정

- NLB for LoadBalancer Service
- Metrics Server
- Cluster Autoscaler
- CloudWatch Container Insight
- Vertical Pod Autoscaler

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: myeks
  region: us-east-1
  version: "1.22"

# AZ
availabilityZones: ["us-east-1a", "us-east-1b",  "us-east-1c"]

# IAM OIDC & Service Account
iam:
  withOIDC: true
  serviceAccounts:
    - metadata:
        name: aws-load-balancer-controller
        namespace: kube-system
      wellKnownPolicies:
        awsLoadBalancerController: true
    - metadata:
        name: ebs-csi-controller-sa
        namespace: kube-system
      wellKnownPolicies:
        ebsCSIController: true
    - metadata:
        name: cluster-autoscaler
        namespace: kube-system
      wellKnownPolicies:
        autoScaler: true
...
```

Jenkins - PV, PVC 생성

내결함성 보장을 위해
Kubernetes 클러스터 상에서 Pod가 중지되더라도
Jenkins 서버의 운영과 관련된 데이터를 **EBS** 내 저장하여
Pod 재배포 시 **동일한 운영 환경** 확보

```yaml
# Persistent Volume Claim
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
...
```

```yaml
# Persistent Volume
apiVersion: v1
kind: PersistentVolume
metadata:
  name: jenkins-pv
spec:
  capacity:
    storage: 10G
  accessModes:
    - ReadWriteOnce
...
```

Jenkins - Deployment, NodePort

Deployment 배포를 통해
Pod와 ReplicaSet에 대한
선언적 업데이트를 제공

IAM 권한으로 인해
서비스로 NodePort 사용
+) EC2 포트 개방

```yaml
# Deployment Config
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins-deployment
spec:
...
    spec:
      containers:
        - name: jenkins
          image: jenkins/jenkins:lts
          resources:
            limits:
              memory: "1500Mi"
              cpu: "1500m"
            requests:
              memory: "1500Mi"
              cpu: "1500m"
          ports:
            - name: httpport
              containerPort: 8080
            - name: jnlpport
              containerPort: 50000
...
```

```yaml
# Service Config
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/path:   /
      prometheus.io/port:   '8080'
spec:
  selector:
    app: jenkins
  type: NodePort
  ports:
    - name: httpport
      port: 8080
      targetPort: 8080
      nodePort: 32000
    - name: jnlpport
      port: 50000
      targetPort: 50000
```

Docker & Ansible Playbook

Node를 위한 컨테이너로, node:14의 이미지를 사용
package.json 의존성 설치 후 실행될 수 있도록 구성

이미지 생성 후 **Docker Hub**에 이미지 Push

형상관리를 위해 Docker 이미지 뒤에 **태그**를
BUILD_NUMBER로 설정

```
FROM node:14

WORKDIR /

COPY package.json .
RUN npm install
COPY . .

EXPOSE 3000
CMD [ "npm", "start" ]
```

```
- hosts: k8s
  gather_facts: no

  tasks:
    - name: Login to Docker Hub
      docker_login:
        username: ddung1203
        password: "{{ lookup('env', 'PASS') }}"
        reauthorize: yes

    - name: Build and Push Image
      docker_image:
        build:
          path: realmytrip/
        name: ddung1203/realmytrip
        tag: "{{ lookup('env', 'BUILD_NUMBER') }}"
        push: yes
        source: build
```

Ansible Playbook

WAS 배포를 위한 Ansible Playbook

앞서 생성한 “realmytrip” 이미지를 사용하여

Deployment를 탬플릿 형태로 구성

- 리소스 용량 설정
- 환경변수 설정

```
- hosts: k8s
  gather_facts: no

  tasks:
    - name: Create Deployment
      k8s:
        state: present
        definition:
          apiVersion: apps/v1
          kind: Deployment
          metadata:
            name: realmytrip
            namespace: default
          spec:
            replicas: 1
            selector:
              matchLabels:
                app: realmytrip
            template:
              metadata:
                labels:
                  app: realmytrip
              spec:
                containers:
                  - name: realmytrip
                    image: "ddung1203/realmytrip:{{ lookup('env', 'BUILD_NUMBER') }}"
                    imagePullPolicy: Always
                    ports:
                      - containerPort: 3000
                    resources:
                      requests:
                        cpu: 500m
                        memory: 500M
                      limits:
                        cpu: 800m
                        memory: 800M
                    env:
                      - name: AWS_ACCESS_KEY_ID
                        value: "{{ lookup('env', 'AWS_ACCESS_KEY_ID') }}"
                      - name: AWS_SECRET_ACCESS_KEY
                        value: "{{ lookup('env', 'AWS_SECRET_ACCESS_KEY') }}"
                      - name: S3_BUCKET
                        value: "{{ lookup('env', 'S3_BUCKET') }}"
```

Ansible Playbook

WAS 배포를 위한 Ansible Playbook

“realmytrip” Deployment의 네트워크 설정을 위한 **Service**

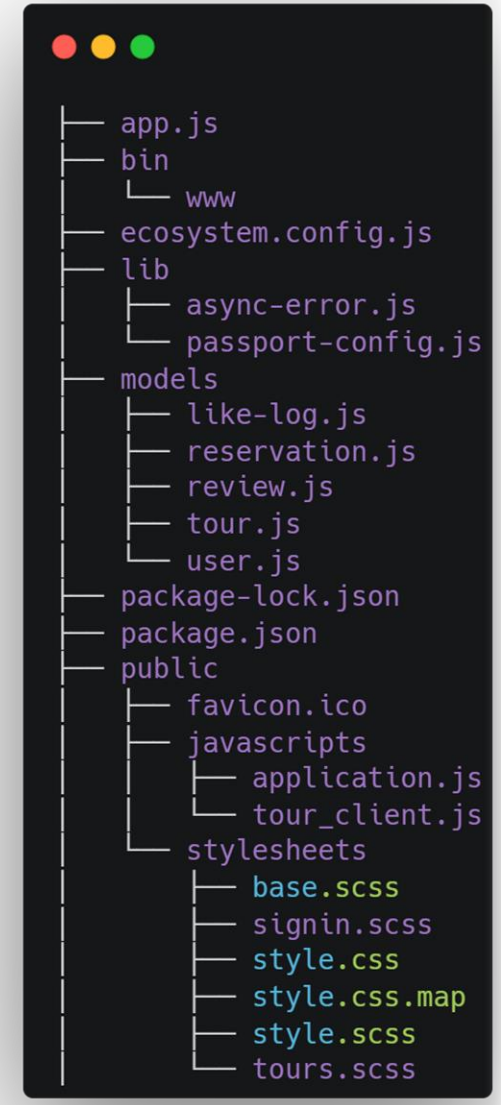
annotation을 통해 인터넷 통신이 가능한 로드밸런서 설정

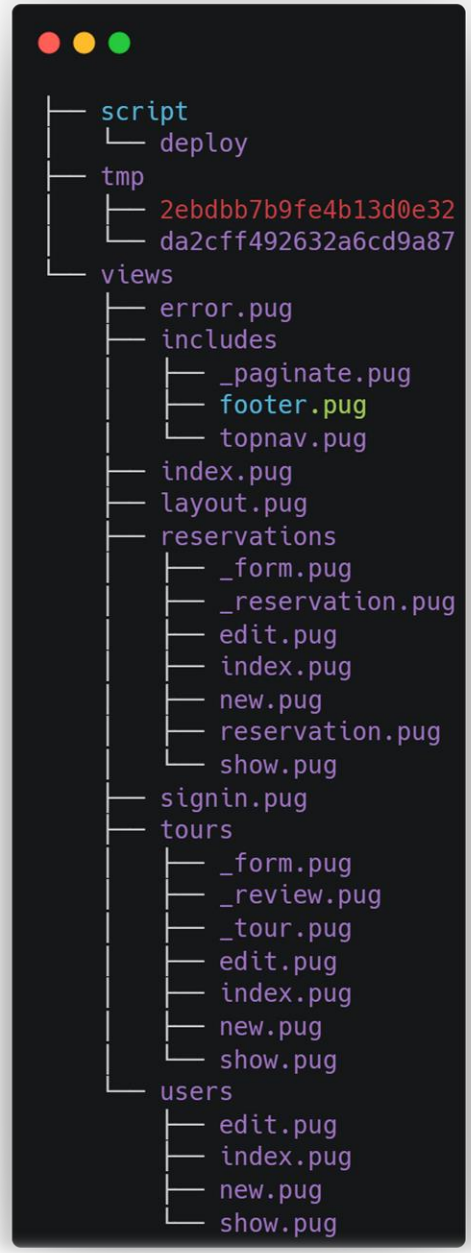
→ **Learner Lab**의 **IAM** 문제로 인해 **NodePort**로 설정

```yaml
- name: Create Service
  k8s:
    state: present
    definition:
      apiVersion: v1
      kind: Service
      metadata:
        name: realmytrip-np
        namespace: default
        annotations:
         service.beta.kubernetes.io/aws-load-balancer-type: "external"
         service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
         service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
      spec:
        type: NodePort
        selector:
          app: realmytrip
        ports:
          - port: 80
            targetPort: 3000
            nodePort: 32123
```

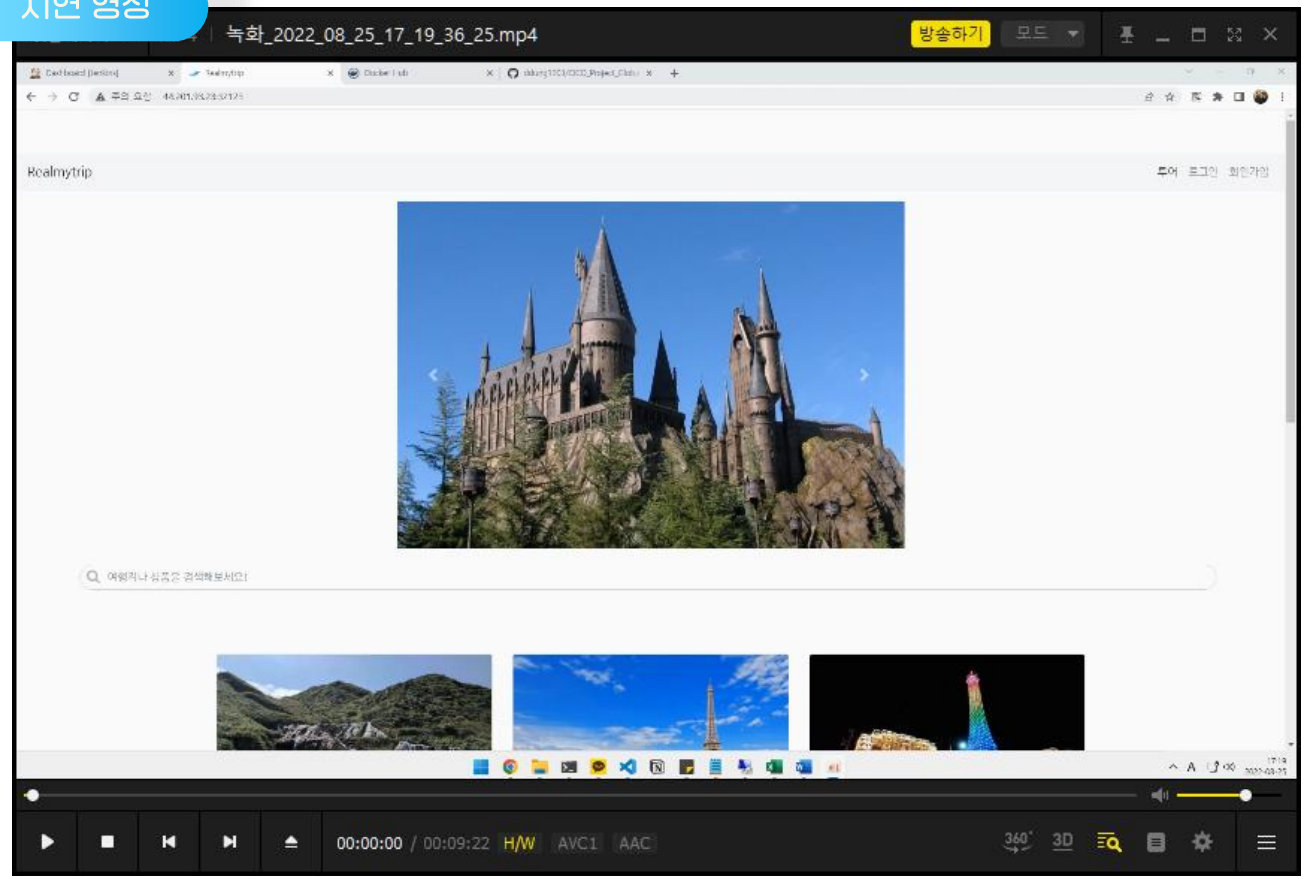
WAS

WAS의 tree 구조





시연 영상


녹화_2022_08_25_17_19_36_25.mp4
방송하기
모드
Realmytrip
00:00:00 / 00:09:22 H/W AVC1 AAC
360
3D

2조 전중석

감사합니다.