# MetaMedical

## 하이브리드 기반 보건 행정 서비스 클라우드 구축

**메타5형제** | 조은새 박상원 김지우 이은지 육지현

클라우드 인프라 엔지니어 양성과정

KOSA Korea Software Industry Association

Metanet

MetaMedical

# 프로젝트 목차

# 팀원 소개

## 조은새

Public Cloud

Public Cloud Architecture 구성
Terraform Infra 유지보수
App 개발 및 테스트

## 박상원

Private Cloud

Private Cloud Architecture 구성
OpenVPN Site to Site 연결
DB 이중화

## 김지우

CI Pipeline

CI Pipeline 구축
Jenkins 환경 구축
이미지 취약점 분석

## 이은지

CD Pipeline

CD Pipeline 구축
ArgoCD 환경 구축
코드 취약점 분석

## 육지현

Monitoring

ArgoCD, Web 서버,
Node, MySQL 모니터링
장애,오류 인지와 예방
Slack 알람 모니터링

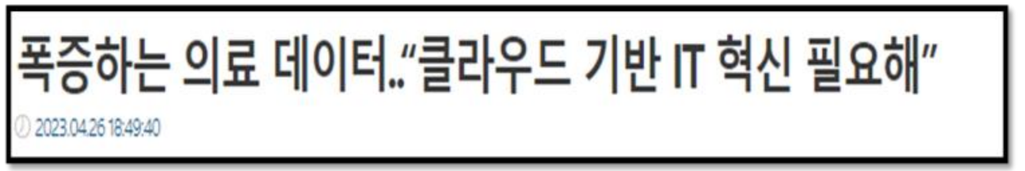
KOSA 한국소프트웨어산업협회
Metanet

# 개요

백신 예약 이어 또 '먹통'...쿠브앱 전산장애, 질병청 "송구"

홍완기 기자 wangi0602@doctorsnews.co.kr | 승인 2021.12.14 10:15 | 댓글 1

"시행 첫날 방역패스 적용 안해...13일 위반, 과태료 부과 안 한다"
서버 증설 했지만 실시간 대량 인증처리 장애 등 과부하 대응 '미흡'



폭증하는 의료 데이터.."클라우드 기반 IT 혁신 필요해"

2023.04.26 18:49:40

55~59세 접종예약 재개…시스템 불안 또 '접속대기'

송고시간 | 2021-07-14 21:09

신선미 기자
기자 페이지

(서울=연합뉴스) 신선미 기자 = '예상 대기시간 111시간 23분 52초'.

만 55~59세 대상 신종 코로나바이러스 감염증(코로나19) 백신 접종 사전예약이 이틀 만인 14일 오후 8시 재개된 가운데 초반부터 접속이 원활하지 않아 이용자들이 불편을 겪고 있다.

예약 신청자가 한꺼번에 몰리면서 코로나19 예방접종 사전예약 시스템에 또다시 접속 장애 현상이 발생한 것이다.

# 시나리오

## Meta NEWS


현재 대한민국은 펜데믹 상황으로 **일정기간 동안 백신접종 예약**을 신청 받는다.

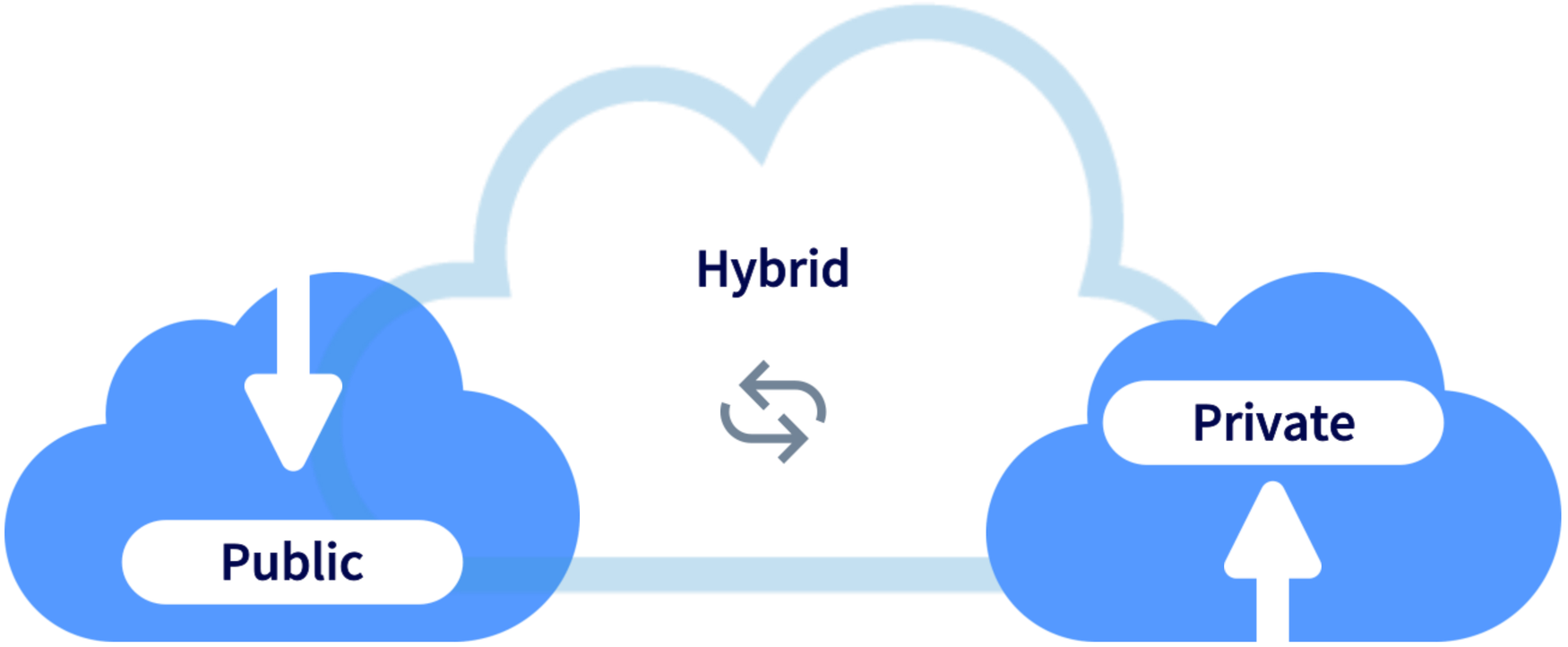
기존 온프레미스로 운영중인 중앙방역 대책 본부는 **일정기간동안 만 40세에서 59세 대상**으로 백신 접종 예약을 받는다.

**민감정보를 보호**하면서 **고가용성**을 갖춘 백신접종 예약서비스를 제공할 수 있는 인프라가 필요한 상황.

## Check List

- ☑ 민감정보 보호
- ☑ 고가용성
- ☑ 안정적인 서비스
- ➔ 하이브리드 클라우드 구축

# 프로젝트 목표




대량의 워크로드 안정적이고 탄력적으로 처리

중요 데이터 분리 및 접근 제어
로 데이터 보안 향상

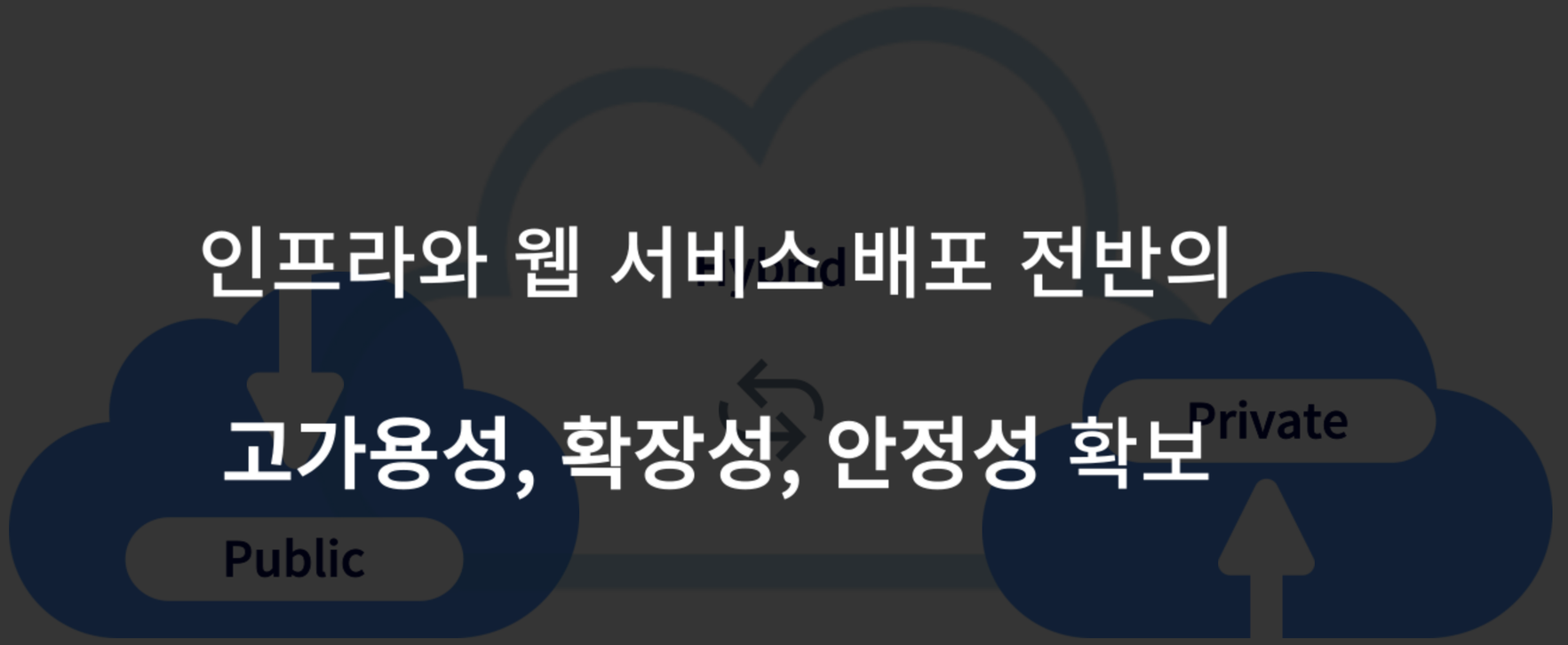
**3-Tier 구성의 웹 어플리케이션 구축**
**신속하고 유연한 대응**

# 프로젝트 목표

MetaMedical
메타5형제

인프라와 웹 서비스배포 전반의

고가용성, 확장성, 안정성 확보




대량의 워크로드 안정적이고 탄력적으로 처리

중요 데이터 분리 및 접근 제어
로 데이터 보안 향상

3-Tier 구성의 웹 어플리케이션 구축
신속하고 유연한 대응

KOSA 한국소프트웨어산업협회
Metanet

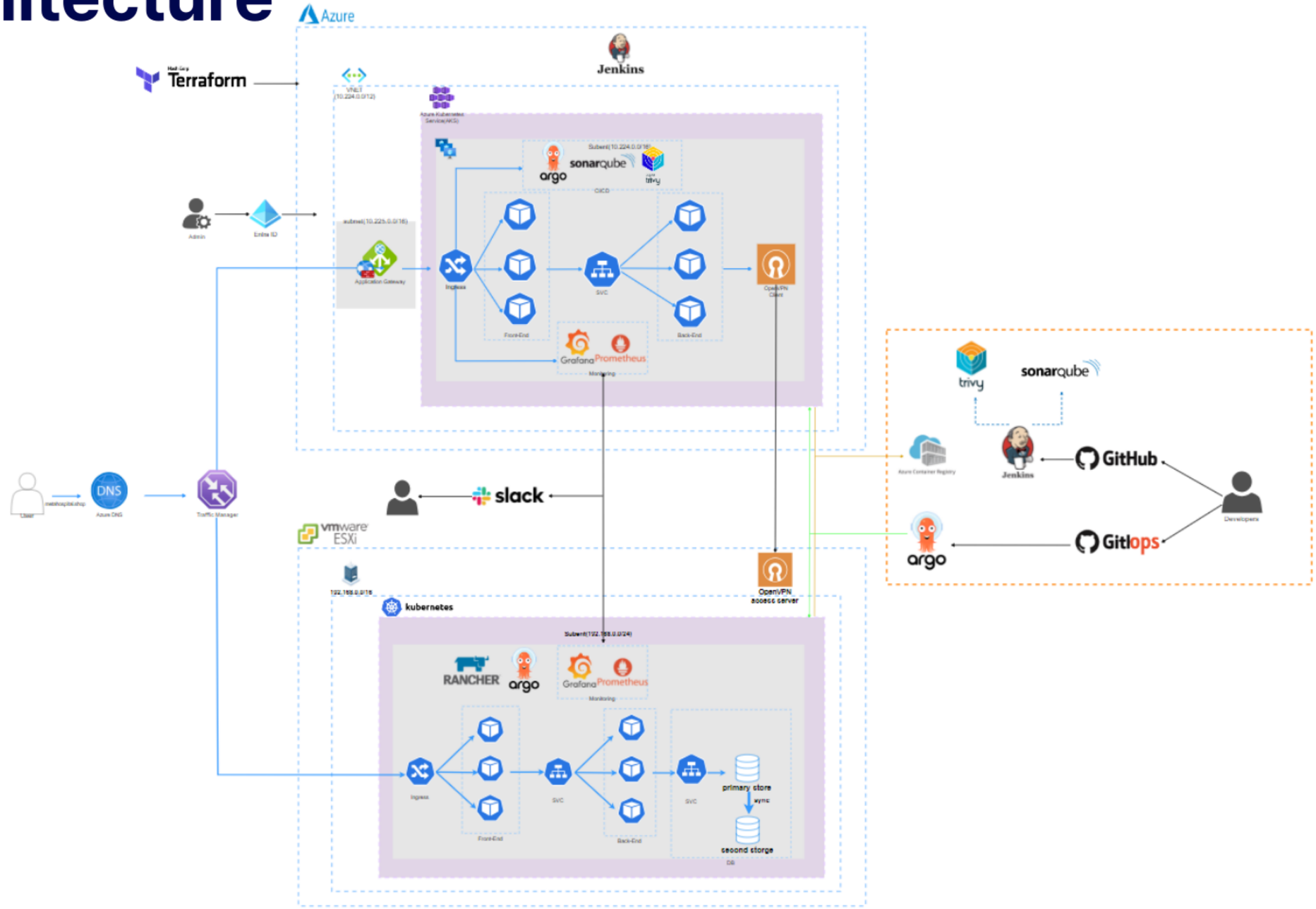
# 기술 스택

MetaMedical
메타5형제

| 퍼블릭 클라우드 | 쿠버네티스 운영/관리 | | | 모니터링 | | |
|---|---|---|---|---|---|---|
| Microsoft Azure | kubernetes | HELM | Azure Kubernetes Service(AKS) | Prometheus | AlertManager | Grafana |
| **프로비저닝 도구** | **DB** | | **VCS** | | **CI/CD** | |
| HashiCorp Terraform | MySQL | | GitHub | Gitops | argo | Jenkins |
| **협업 도구** | | **프레임워크** | | | **컨테이너 관리** | |
| slack | GitHub | spring | | | Azure Container Registry | docker |
| **하이퍼바이저** | | **VPN** | | **기타** | | |
| vmware ESXi | | OPENVPN | | trivy | RANCHER BY SUSE | sonarqube |

KOSA 한국소프트웨어산업협회
Metanet

# Architecture




MetaMedical
메타5형제

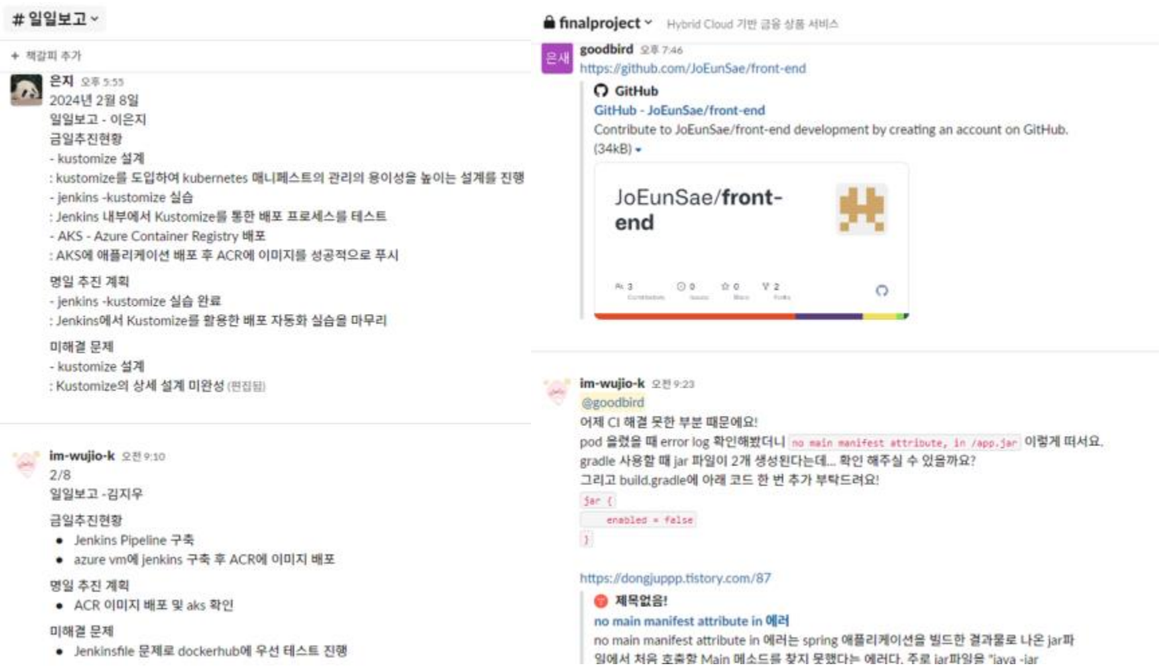
KOSA 한국소프트웨어산업협회
Metanet

# 프로젝트 일정


Task
Week 1
Week 2
Public Cloud 설계/구축
Private Cloud 설계/구축
VPN S2S 연결
CI 설계/구축
CD 설계/구축
Kustomize 구현
APP 개발
모니터링
보안점 추가

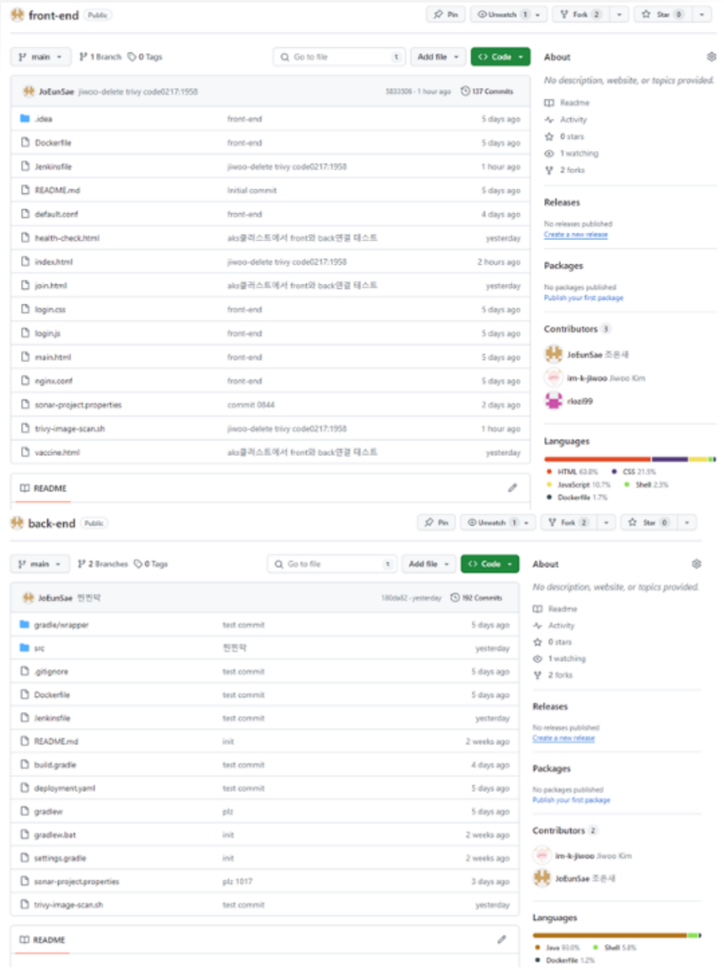
# 협업

slack & GitHub










MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

MetaMedical
메타5형제

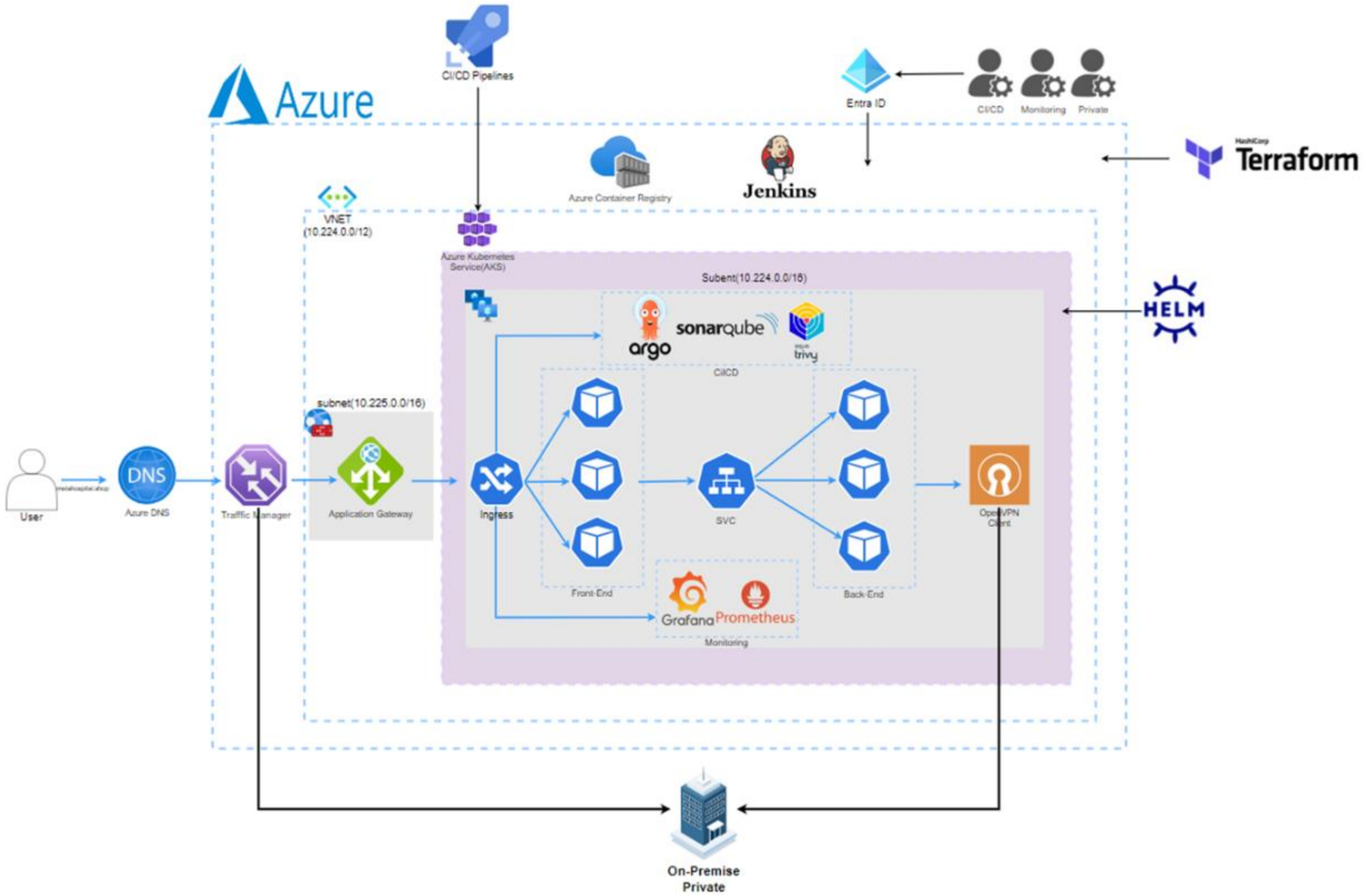
# Public Cloud

## Azure AKS Cluster 기반 탄력적인 인프라 구축

**발표자: 조은새**

KOSA 한국소프트웨어산업협회
Metanet

# Public Architecture

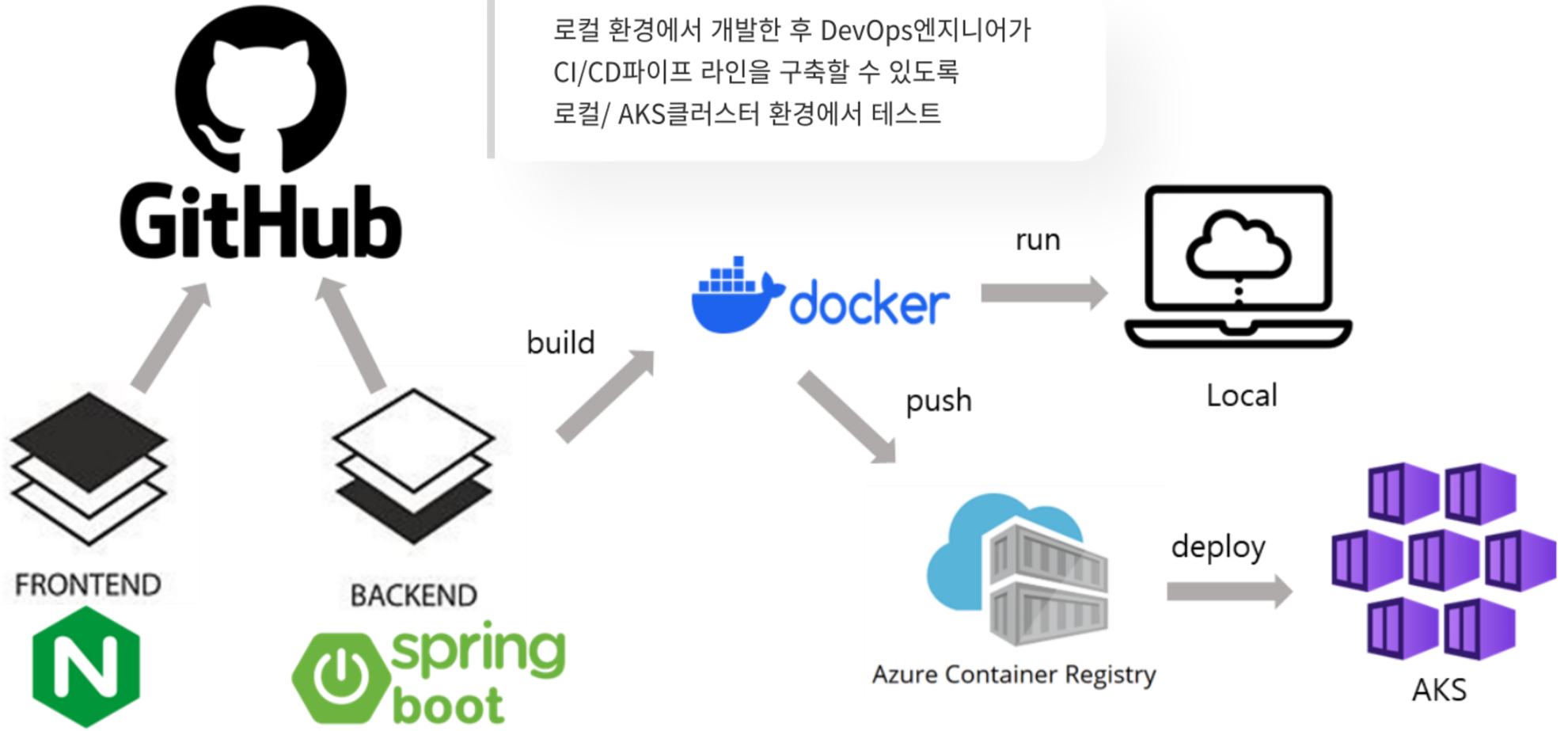



MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

# 앱 개발 및 테스트

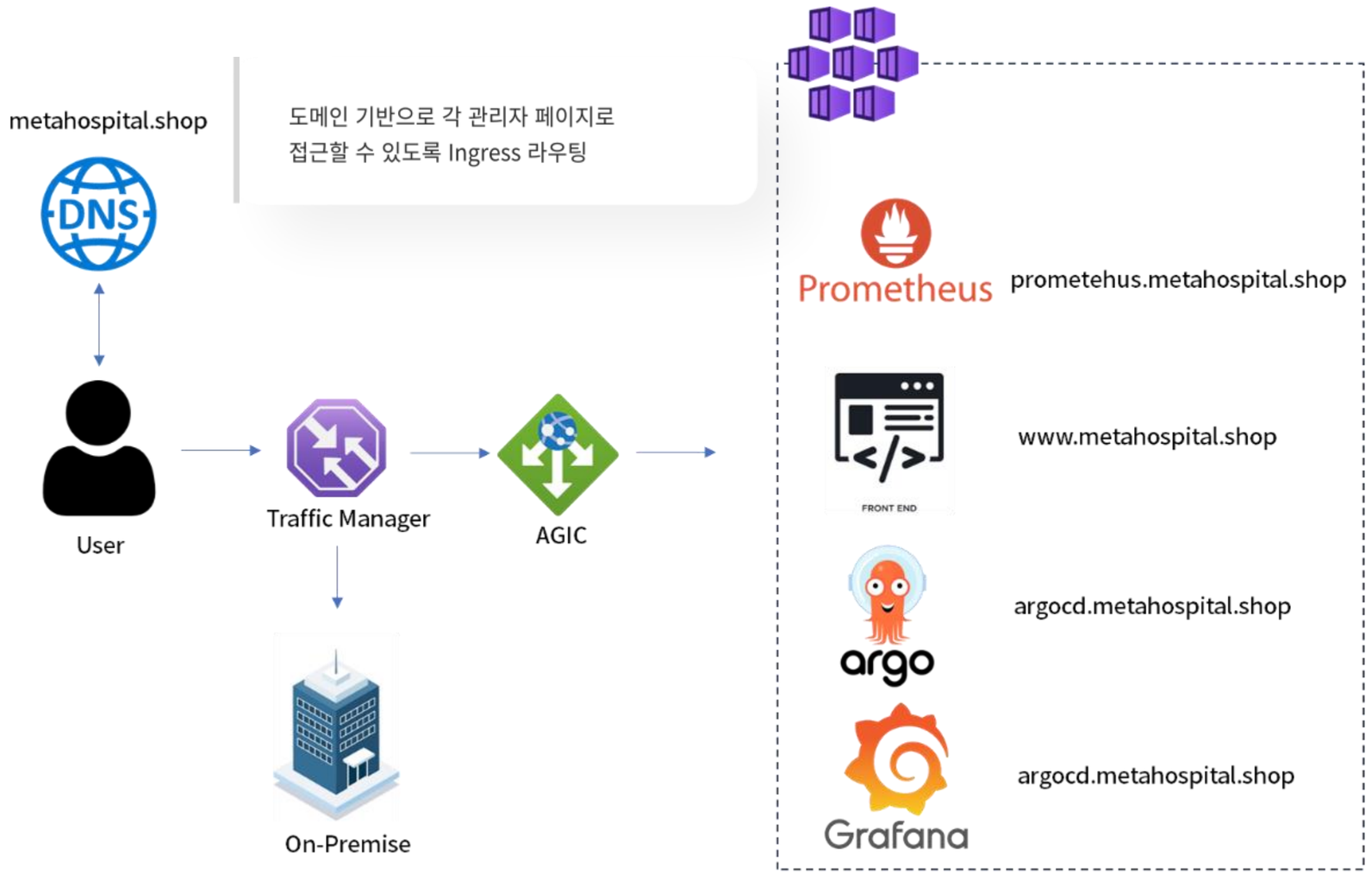
로컬 환경에서 개발한 후 DevOps엔지니어가
CI/CD파이프 라인을 구축할 수 있도록
로컬/ AKS클러스터 환경에서 테스트



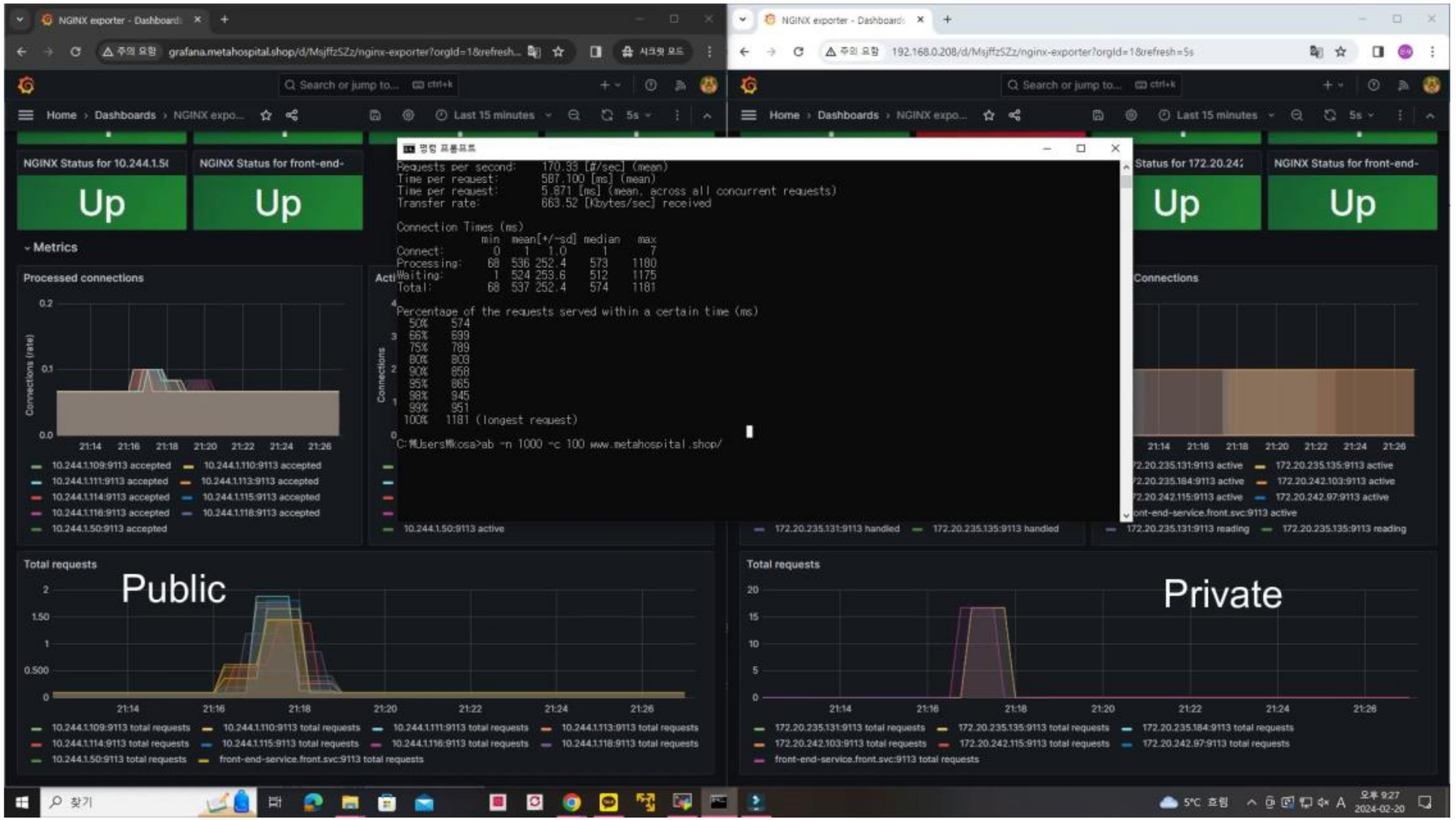
# 앱 개발 및 테스트


회원가입!
ID
user
Password
•••
이름
조은새
나이
20
주소
서울
핸드폰 번호
010-1111-2222
Register
login
user
•••
Remember
회원이 아니신가요? 회원가입
메타 의료 시스템
백신 신청
백신을 신청하세요!
신청하기
건강검진 신청
건강검진 서비스를 신청하세요!
신청하기
백신 신청
이름
조은새
나이
20
주소
서울
신청하기
건강검진 신청
이름
나이
주소
신청하기
DataBase

# TrafficManager를 통한 트래픽분산

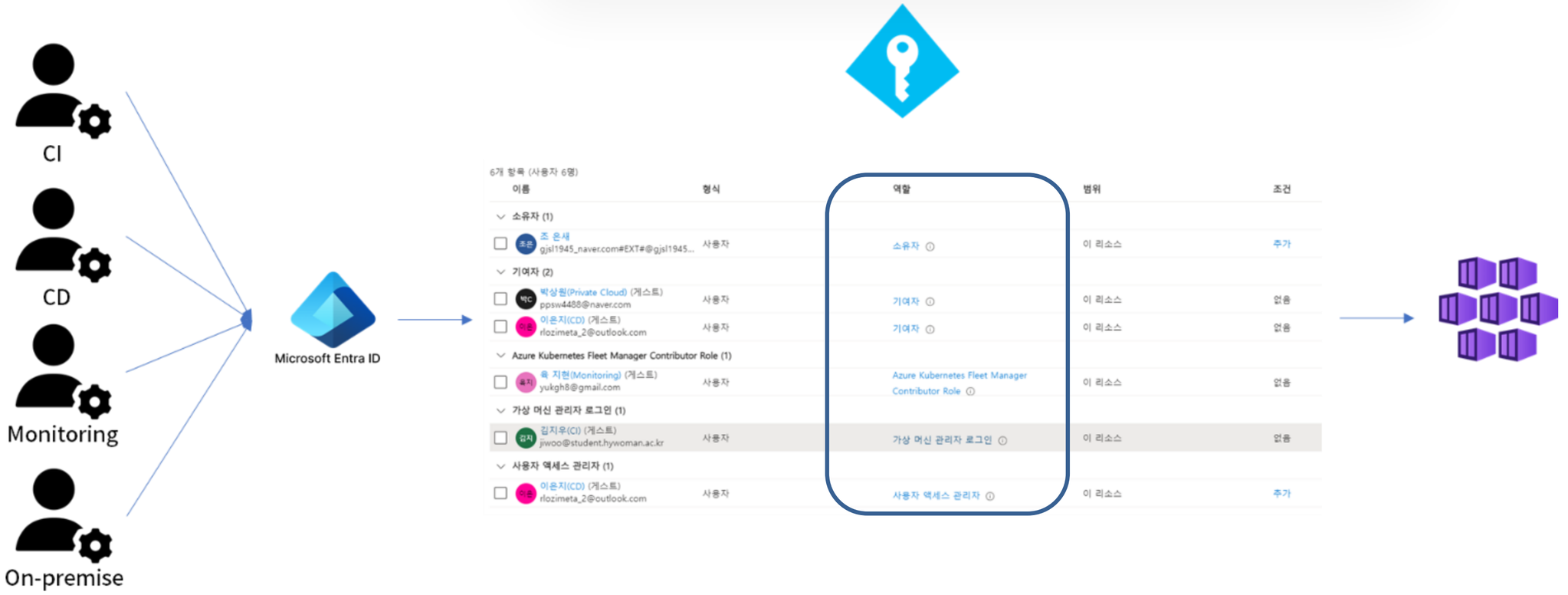


# TrafficManager를 통한 트래픽분산 시연 영상

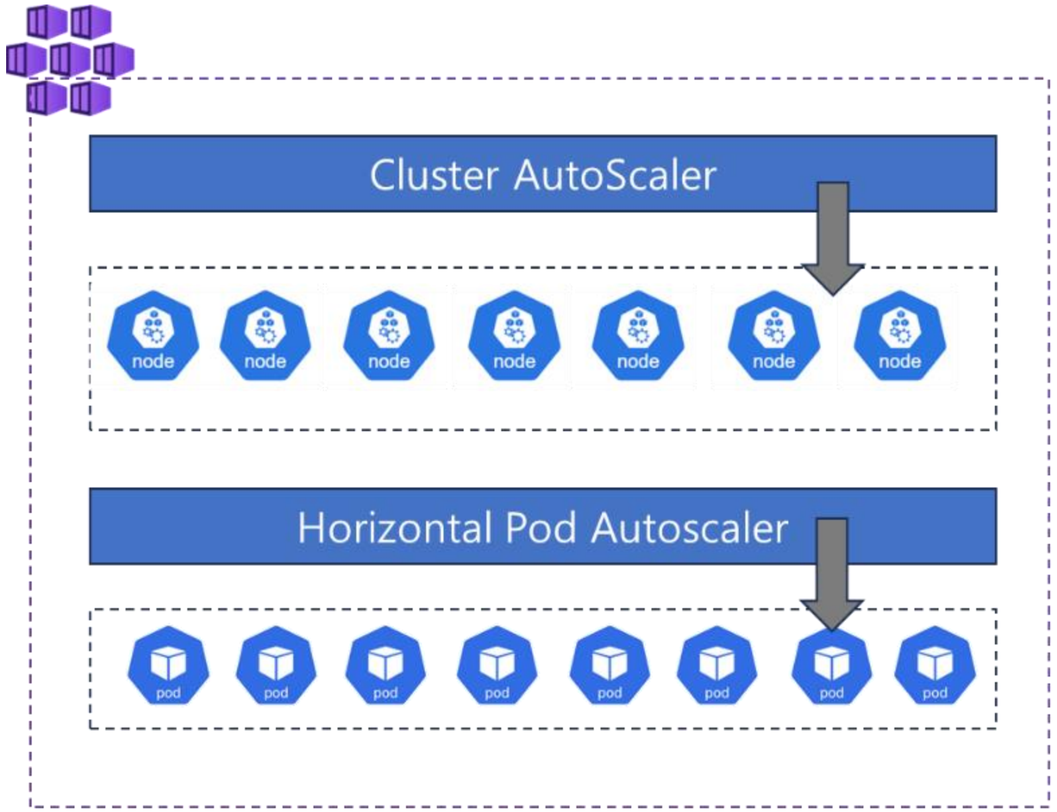


# Role Based Access Control

여러 관리자들을 사용자로 추가하여 각자의 역할에 맞게
액세스를 제어하도록 설정



# Azure AKS Horizontal Pod Autoscaler




HPA.yaml

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: frontend-hpa
  namespace: front
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: front-end-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 10
```

CPU 사용률이 10%가 넘으면 초대 10개까지 pod 확장

# Auto scaling


메모리 사용량이 100%가 넘어
Pod Scaling 실패 이슈 발생


노드 풀 Scale-Up 후 노드 풀 Scaling 구성

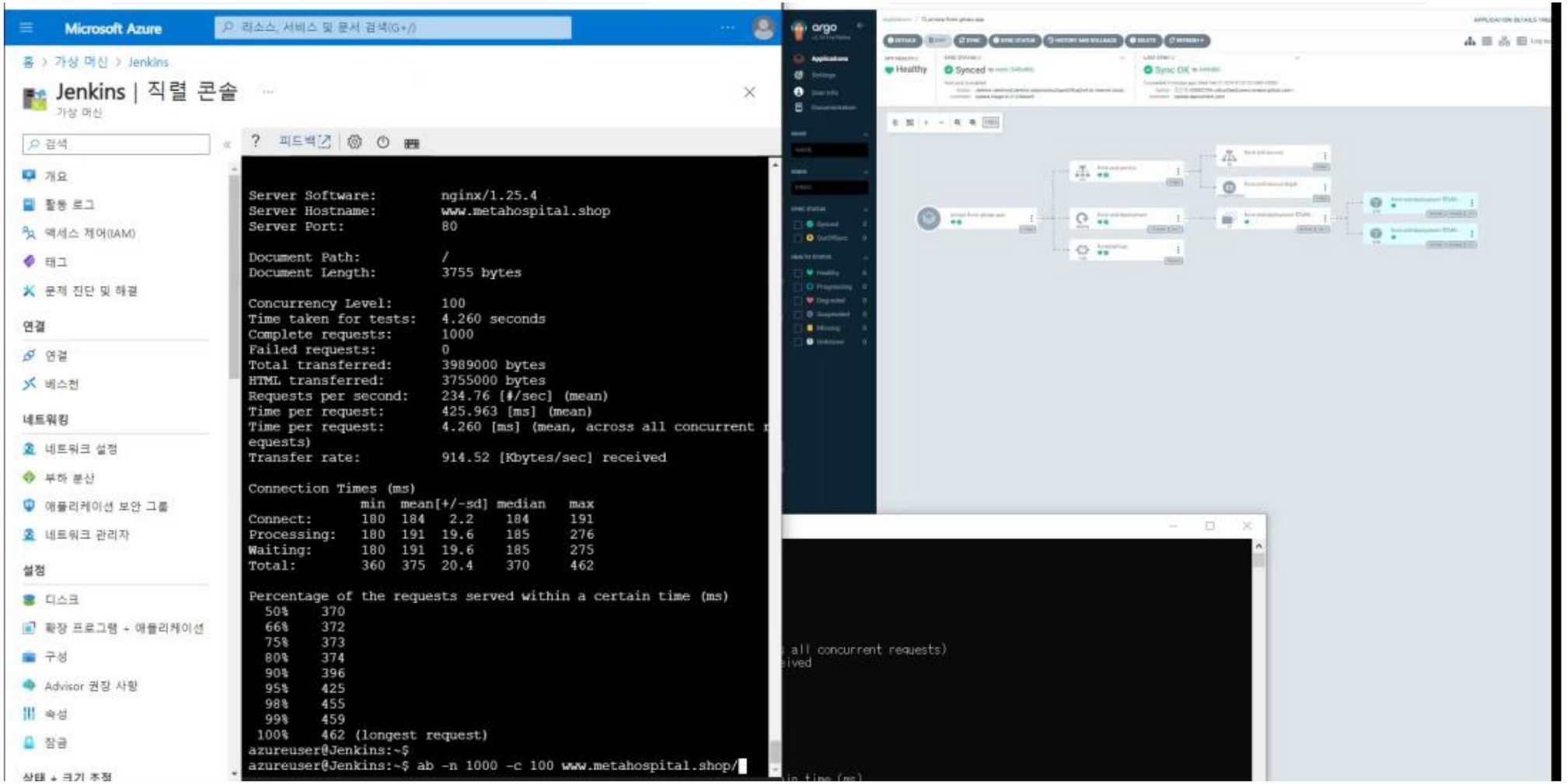
```
D:₩ingress>kubectl top nodes
NAME                                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
aks-agentpool-36424117-vmss000001   314m         8%     5592Mi          48%
```

메모리 공간 확보 후 Pod 확장 확인

MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

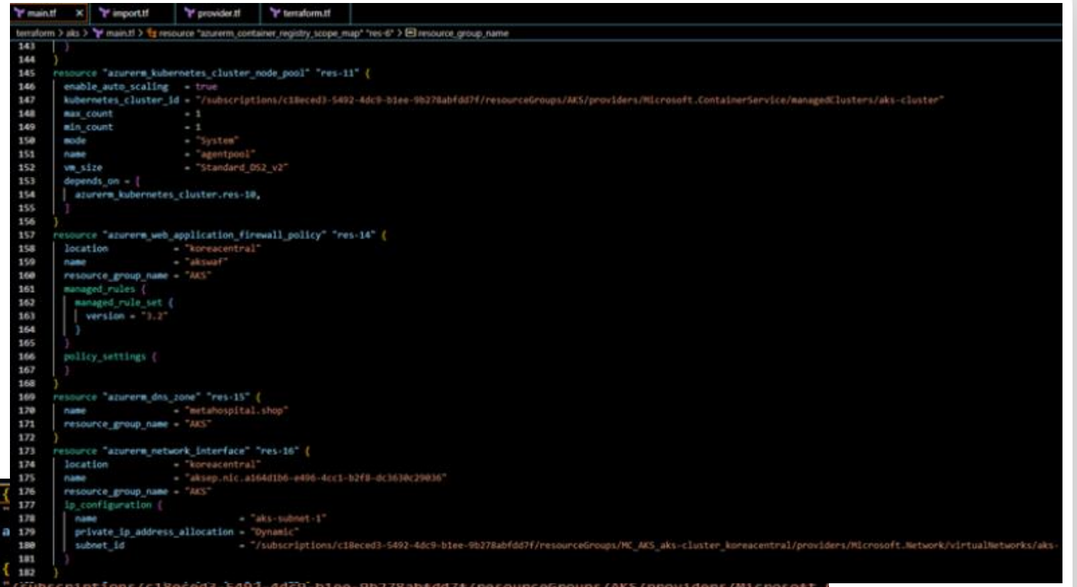
# Auto scaling 시연 영상




MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

# Terraform을 이용한 Azure Infra 관리

```
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_ssh_public_key.res-1
}
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_linux_virtual_machine.res-2
}
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_virtual_machine_extension.res-3
}
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_container_registry.res-4
}
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_container_registry_scope_map.res-5
}
import {
  id = "/subscriptions/c18eced3-5492-4dc9-b1ee-9b278abfdd7f/resourceGroups/AKS/providers/Microsoft.
  to = azurerm_container_registry_scope_map.res-6
}
```

MetaMedical
메타5형제

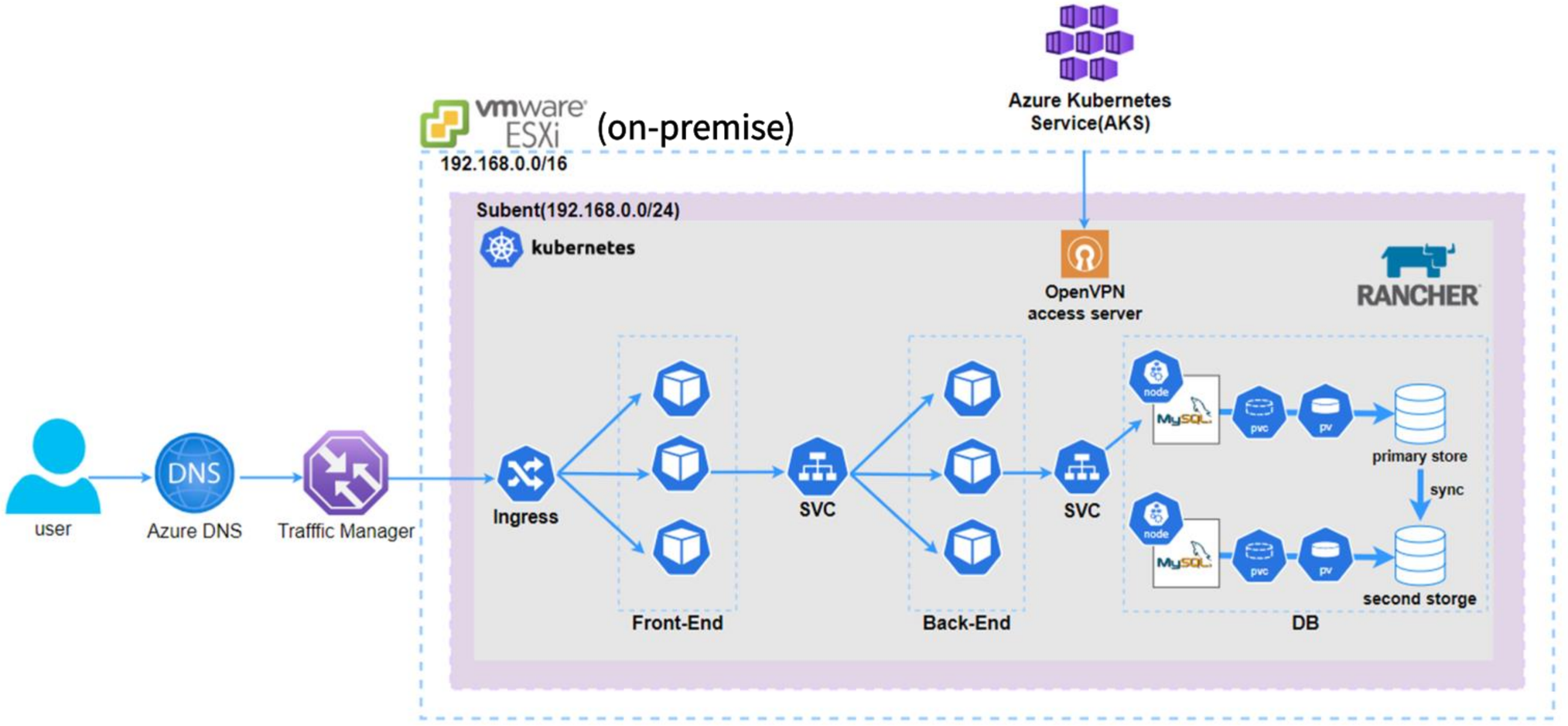
# Private Cloud

## VMware ESXi 기반 K8S 서비스 인프라 환경 구축

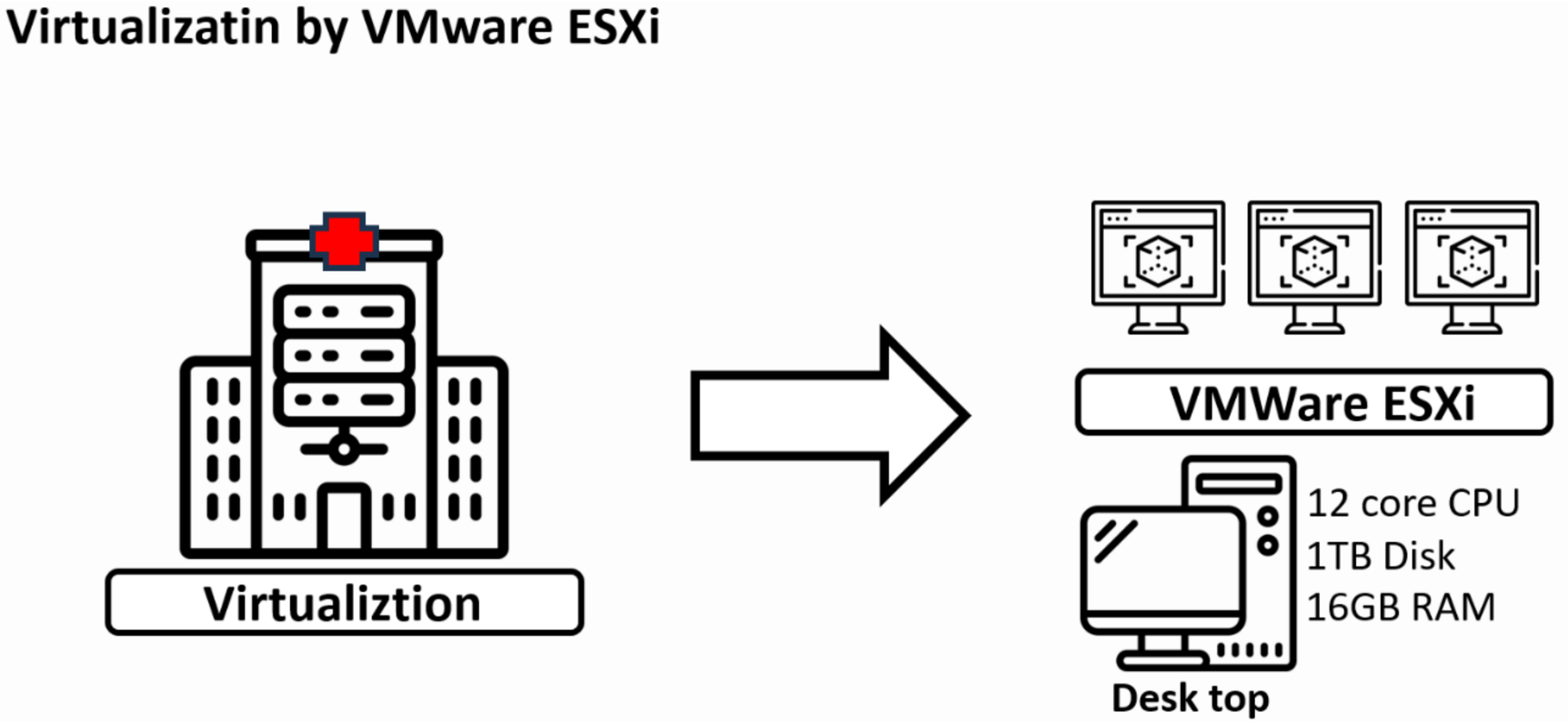
**발표자: 박상원**

KOSA 한국소프트웨어산업협회
Metanet

# Private Cloud Architecture



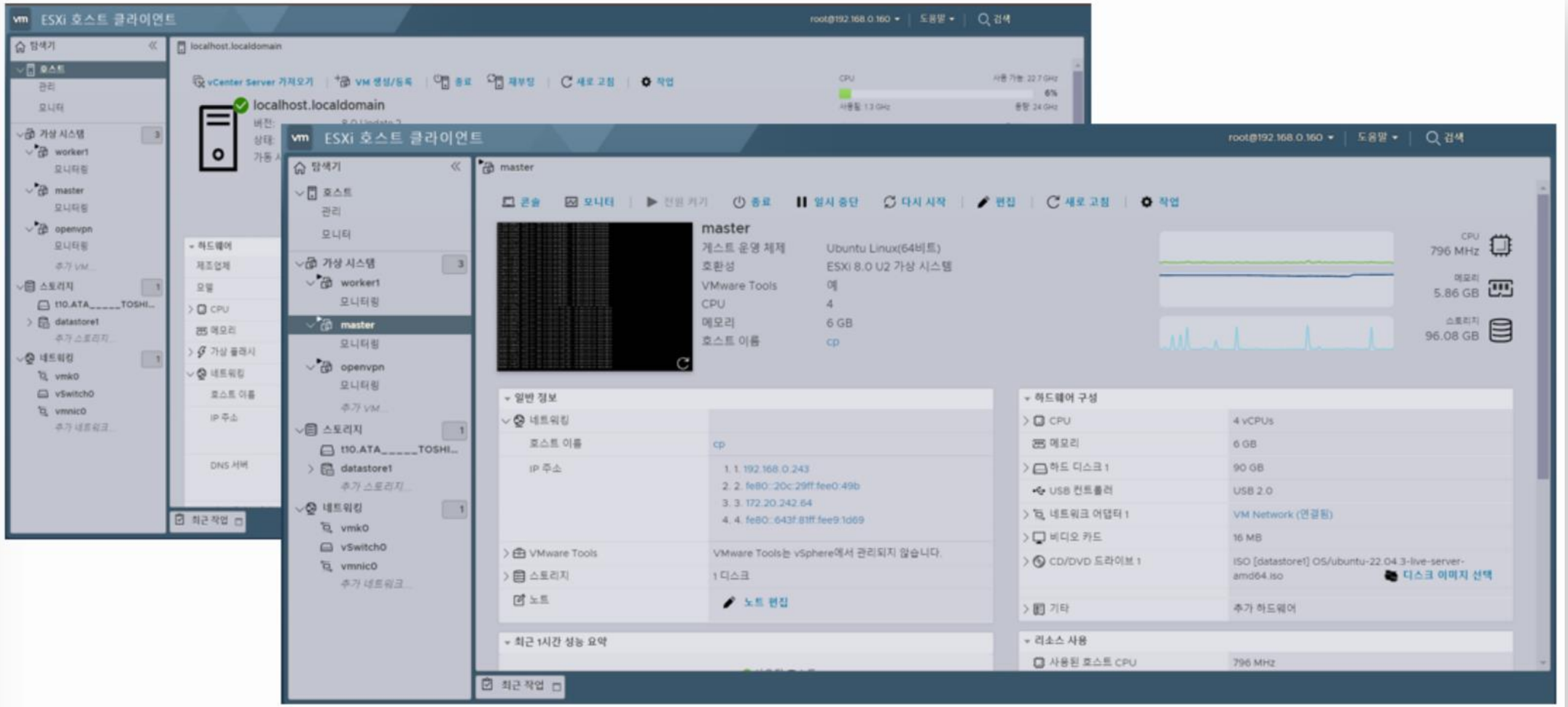
# Virtualization by VMware ESXi




병원의 데이터 센터의 가상화 환경을 가정하기위해 Desktop을 Vmware ESXi 8 을 통해 가상화

MetaMedical
메타5형제

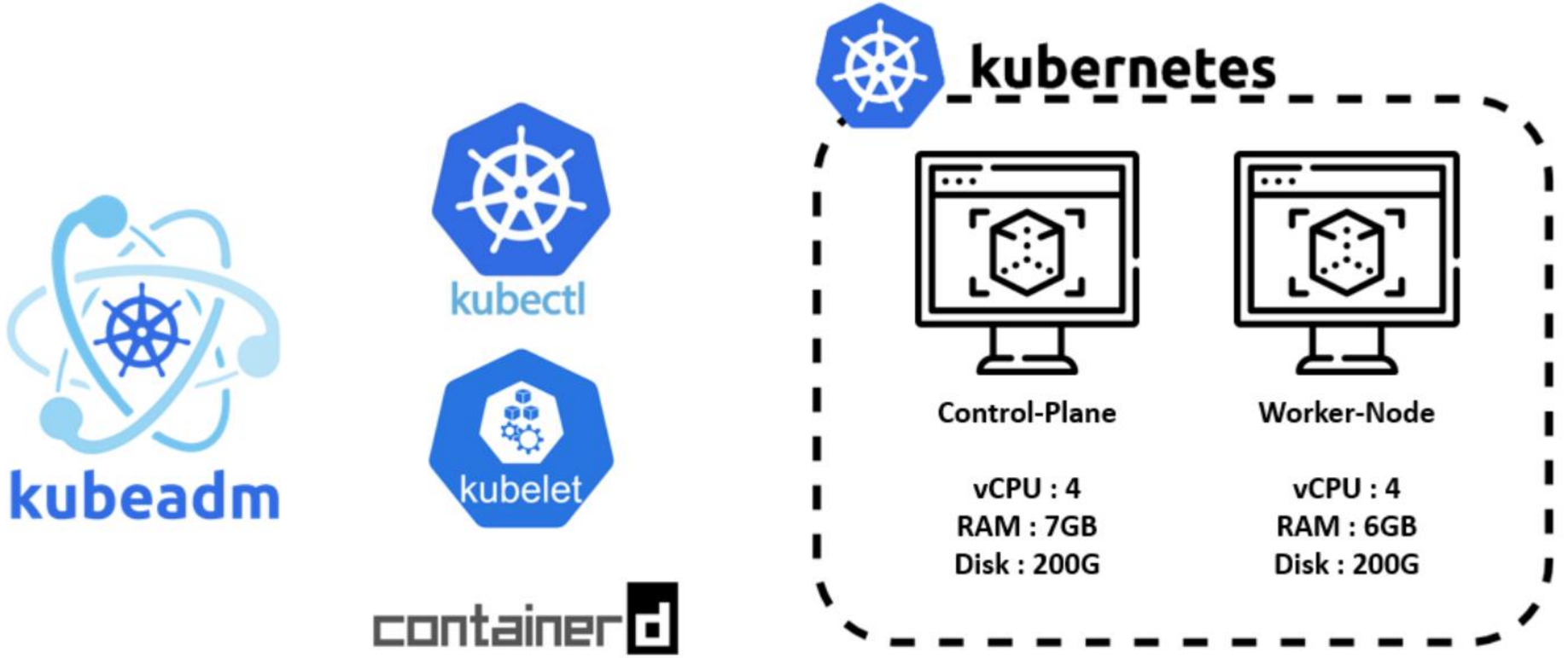
# VMware ESXi를 통한 가상화




ESXi 호스트 클라이언트를 통해 가상머신 생성

KOSA 한국소프트웨어산업협회
Metanet

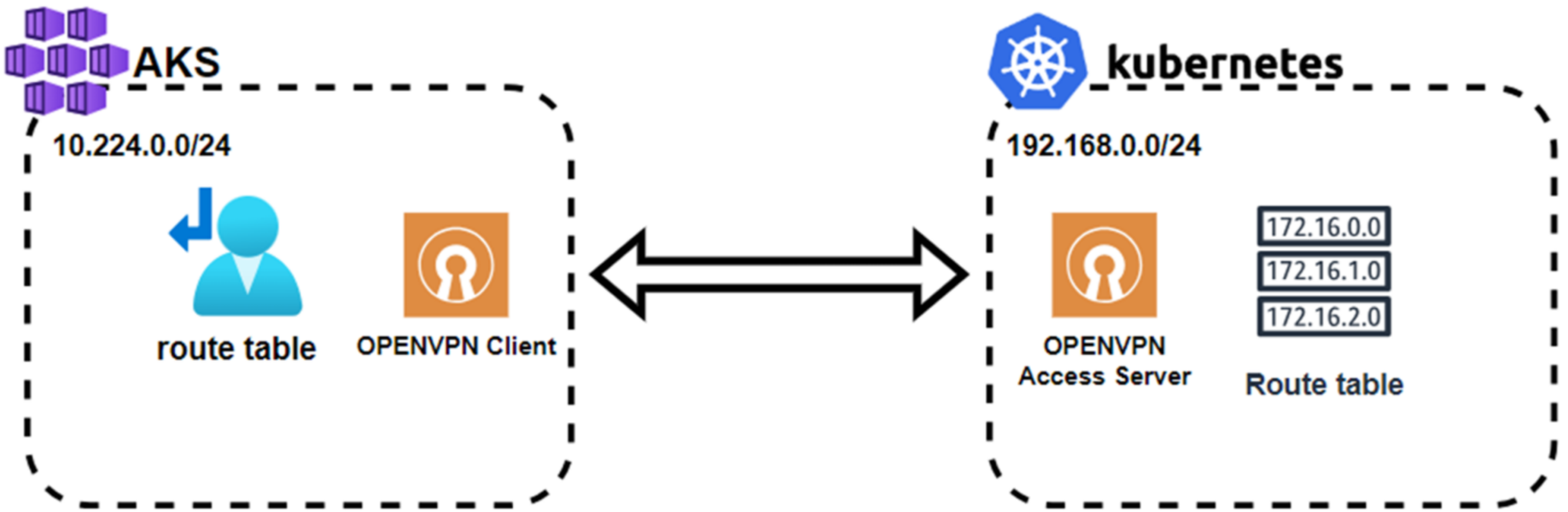
# Kubernetes 클러스터 생성




```
root@cp:~# kubectl get node -o wide
NAME      STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
cp        Ready    control-plane   47h   v1.27.1   192.168.0.243   <none>        Ubuntu 22.04.4 LTS   5.15.0-94-generic   containerd://1.6.28
worker1   Ready    worker          47h   v1.27.1   192.168.0.245   <none>        Ubuntu 22.04.4 LTS   5.15.0-94-generic   containerd://1.6.28
root@cp:~#
```

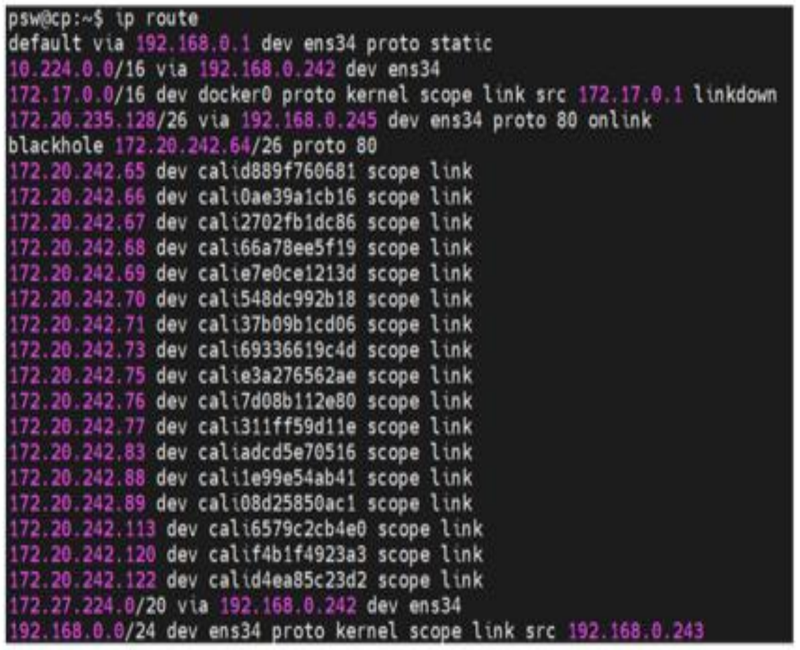
2개의 VM으로 K8S 클러스터 구축

# OpenVPN 을 통한 Site to Site 연결

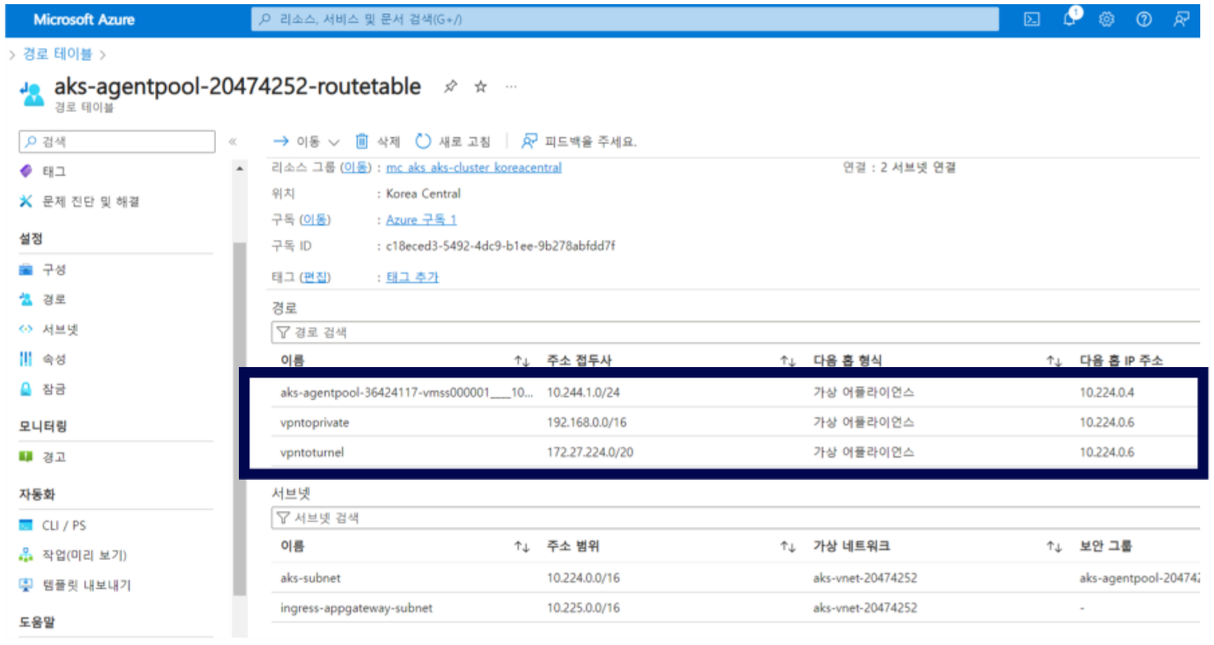



- [x] OpenVPN Access Server Esxi VM 생성
- [x] OpenVPN Client Azure VM 생성
- [x] Route Table 설정을 통한 게이트 웨이 설정
- [x] 공유기 포트포워딩
- [x] PING 테스트를 통한 연결 확인

# OpenVPN 을 통한 Site to Site 연결

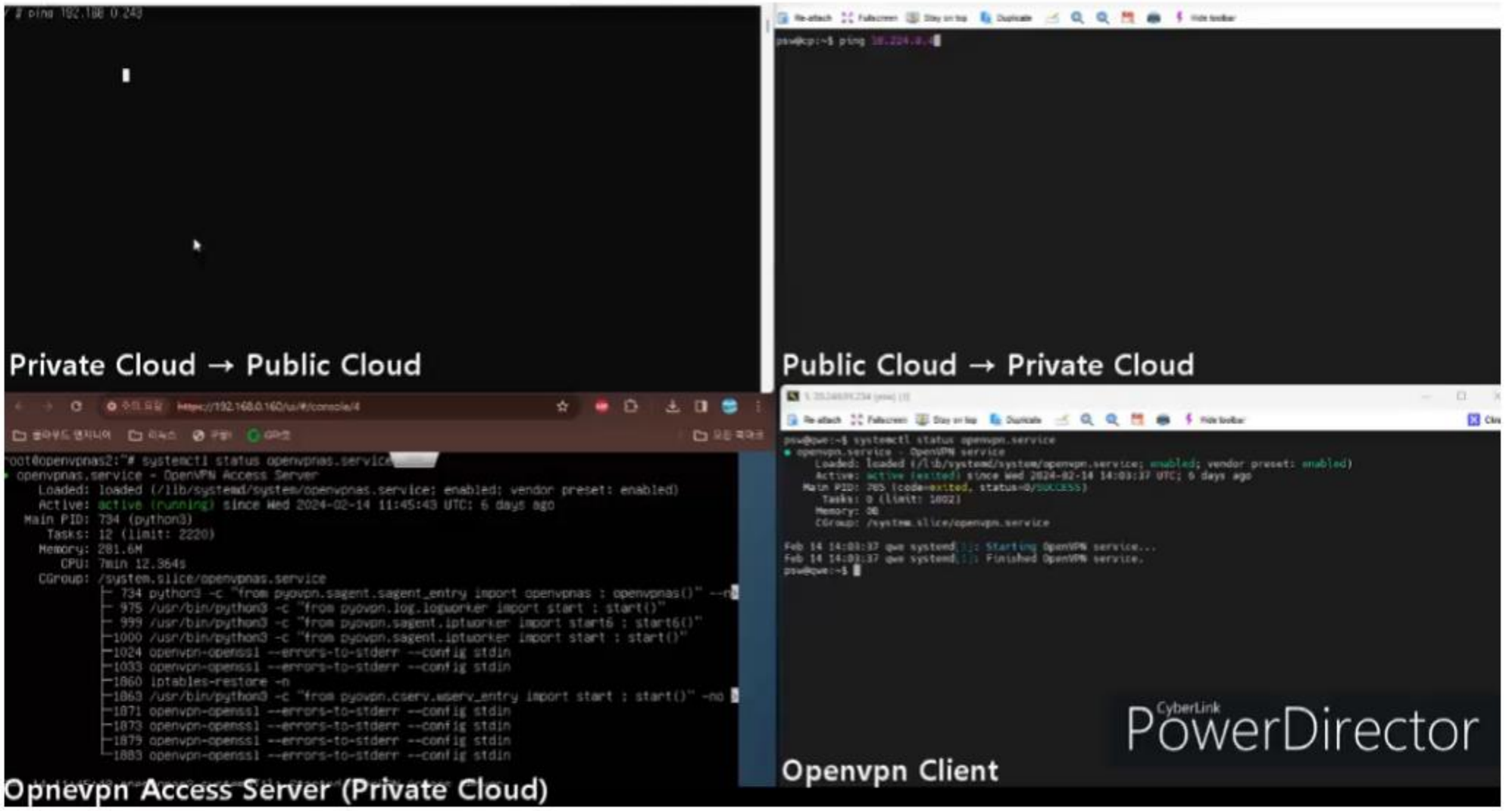



10.244.0.0/16 이 목적지인 트래픽의 라우팅 설정

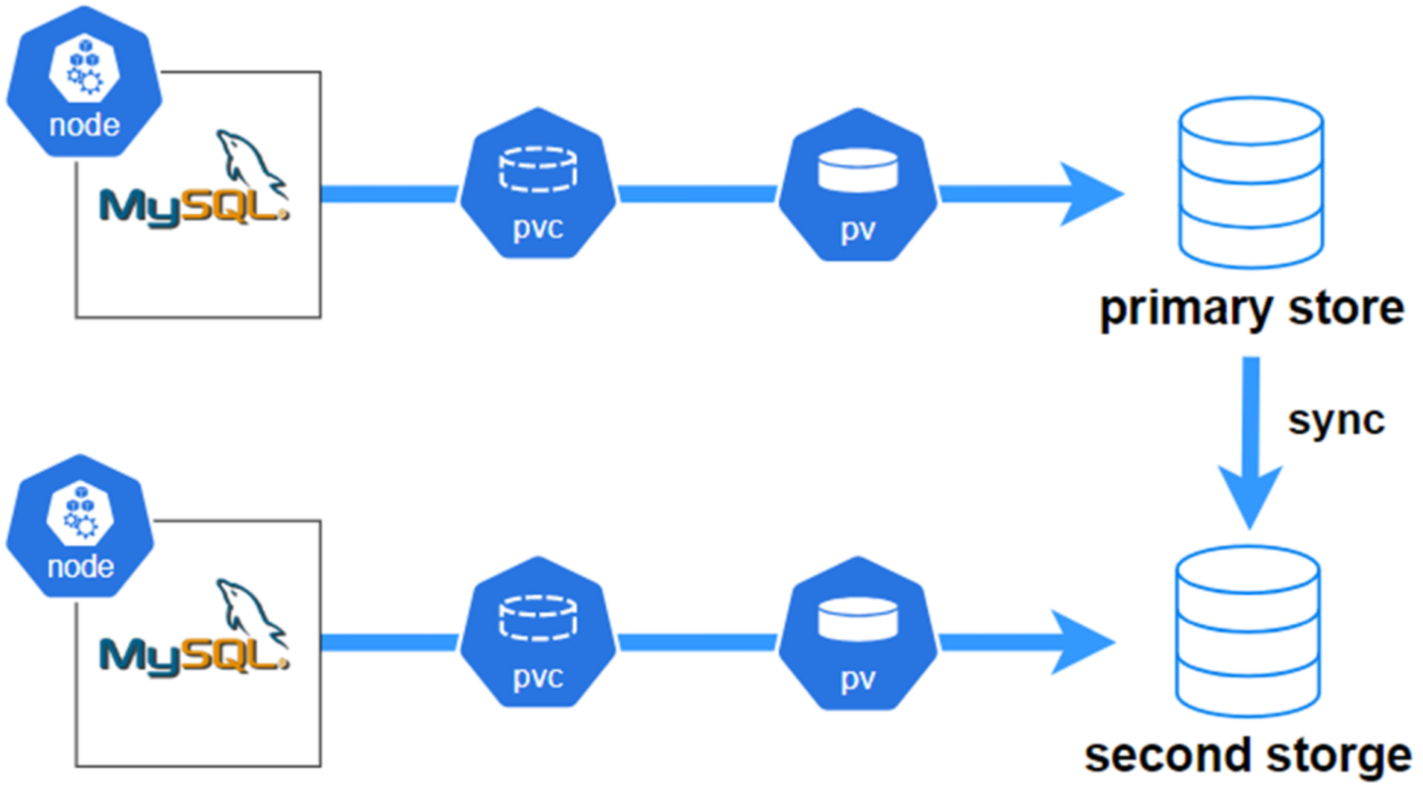



192.168.0.0/16 이 목적지인 트래픽의 라우팅 설정

# OpenVPN 을 통한 Site to Site 연결 시연 영상



# K8S MySQL DB 이중화



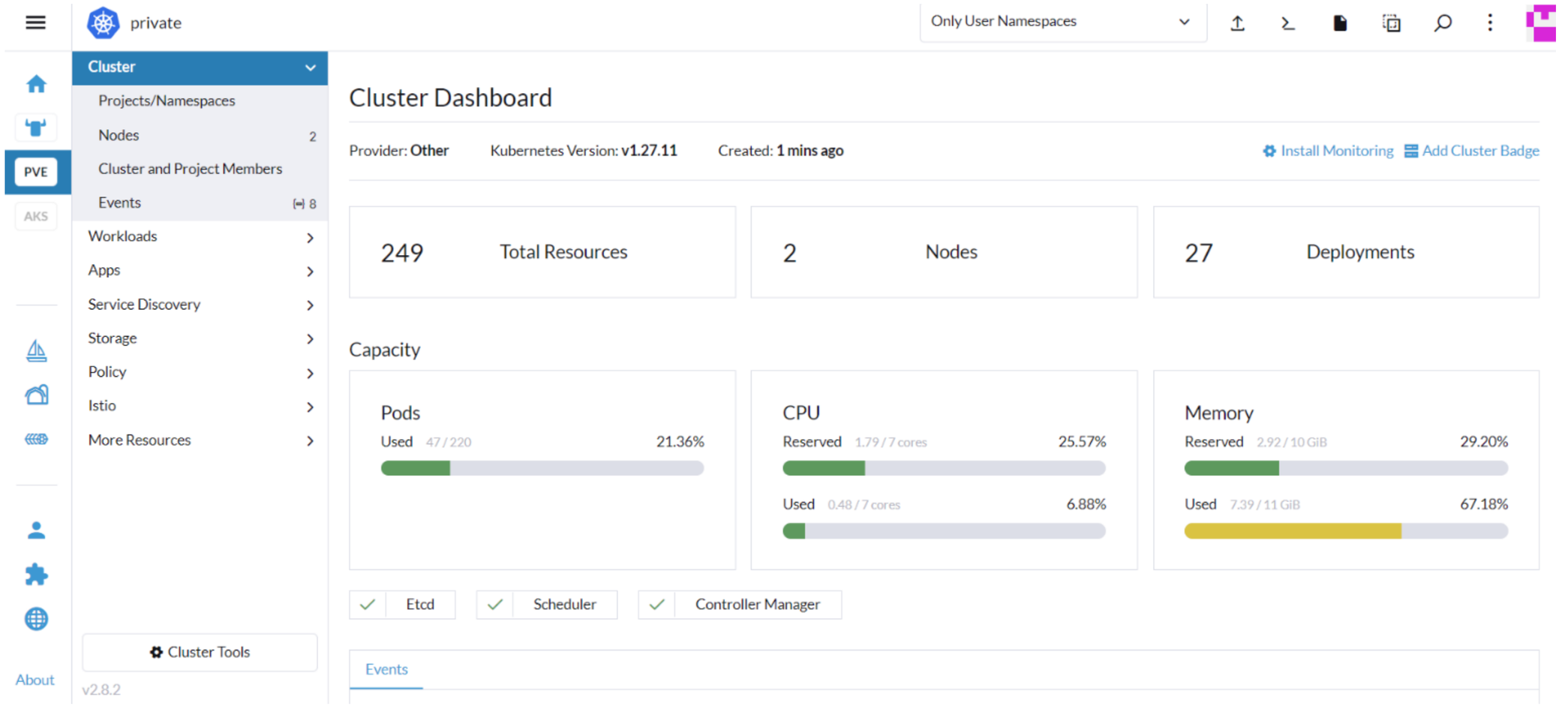

Persistent Volumes 바운드를 통한 데이터 저장

Master/Slave 이중화 구성을 통한 단일실패 지점 극복

# DB 이중화 시연 영상

```
root@mysql-0:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1926
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| healthcheck    |
| users          |
| vaccine        |
+----------------+
3 rows in set (0.01 sec)

mysql>
```

# Rancher를 사용한 멀티클러스터 관리



MetaMedical
메타5형제

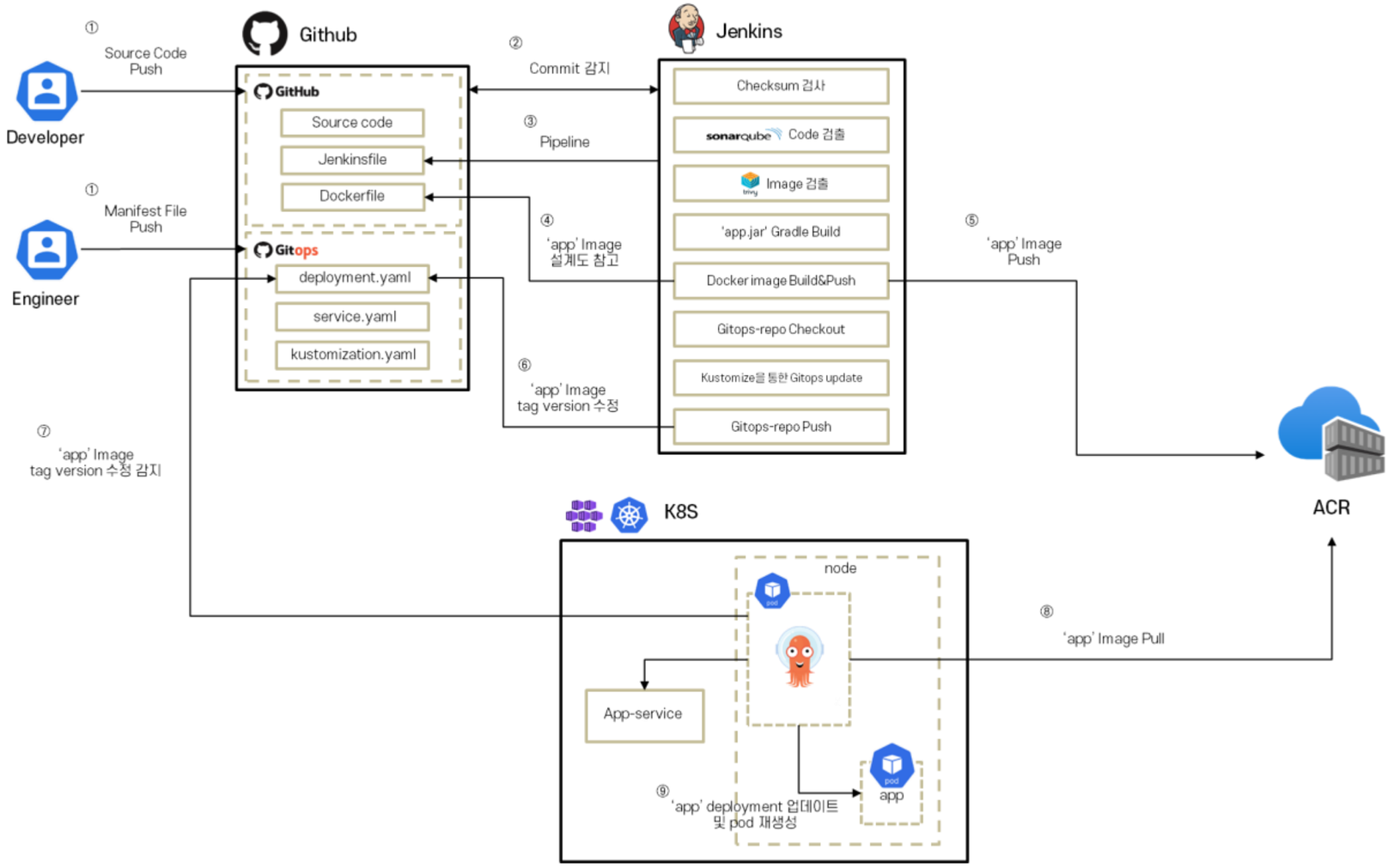
# CICD

## Jenkins 기반 CI 자동화 및 이미지 점검

**발표자: 김지우**

KOSA 한국소프트웨어산업협회
Metanet

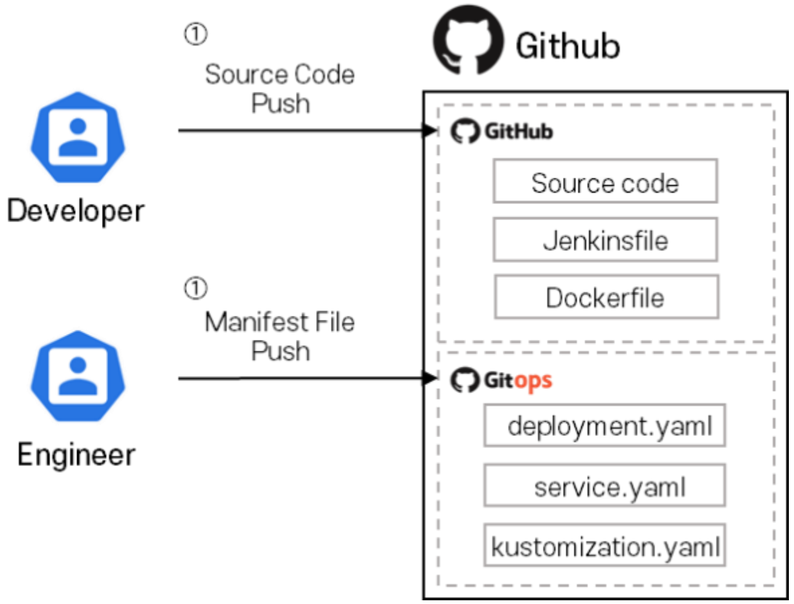
# CICD Architecture



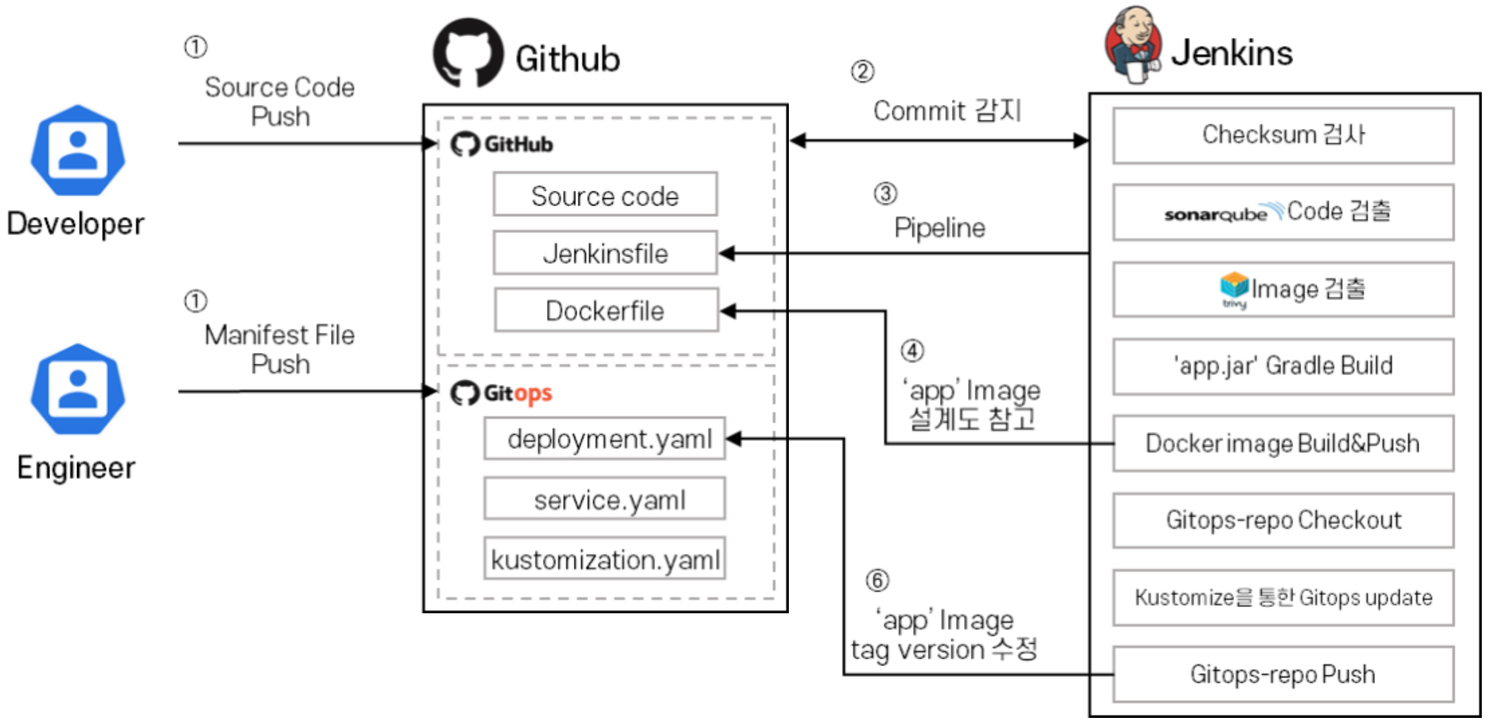

MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

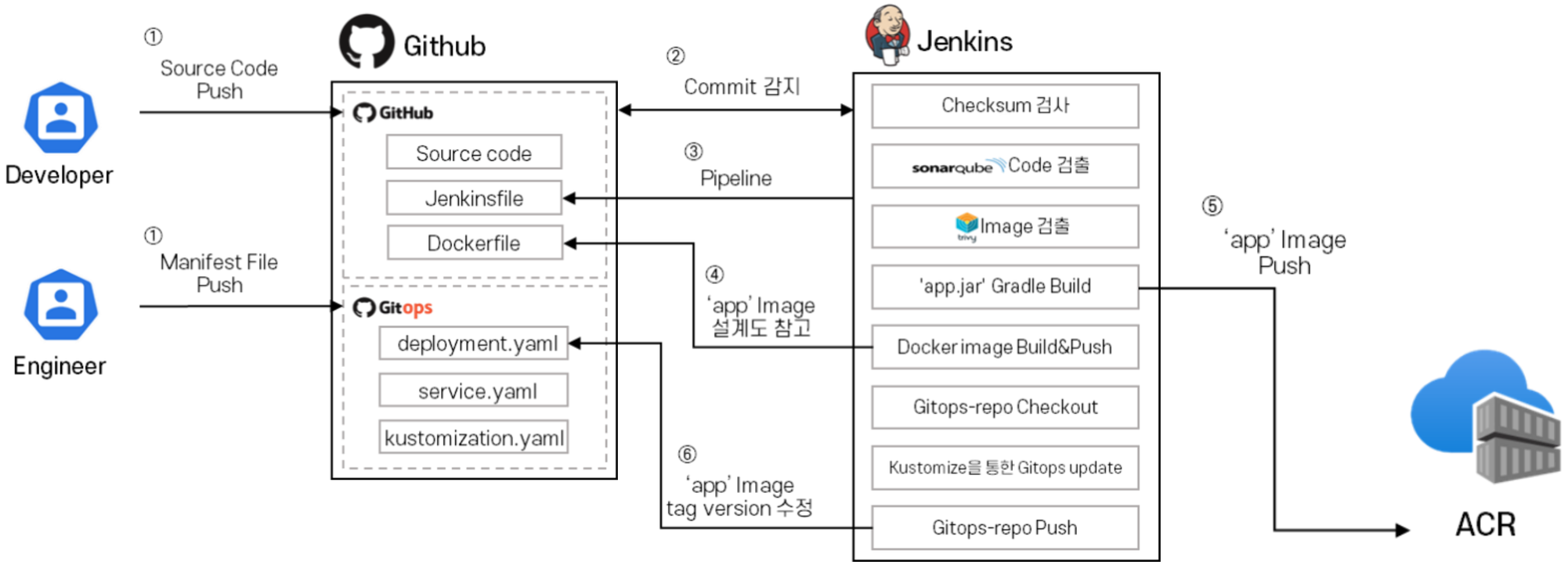
# CI Architecture



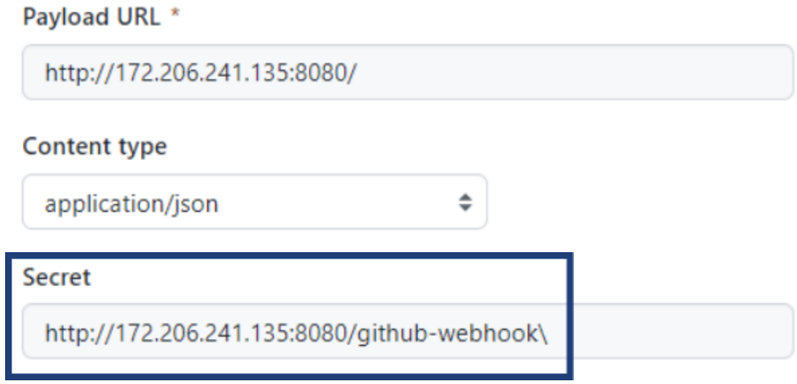
# CI Architecture



# CI Architecture

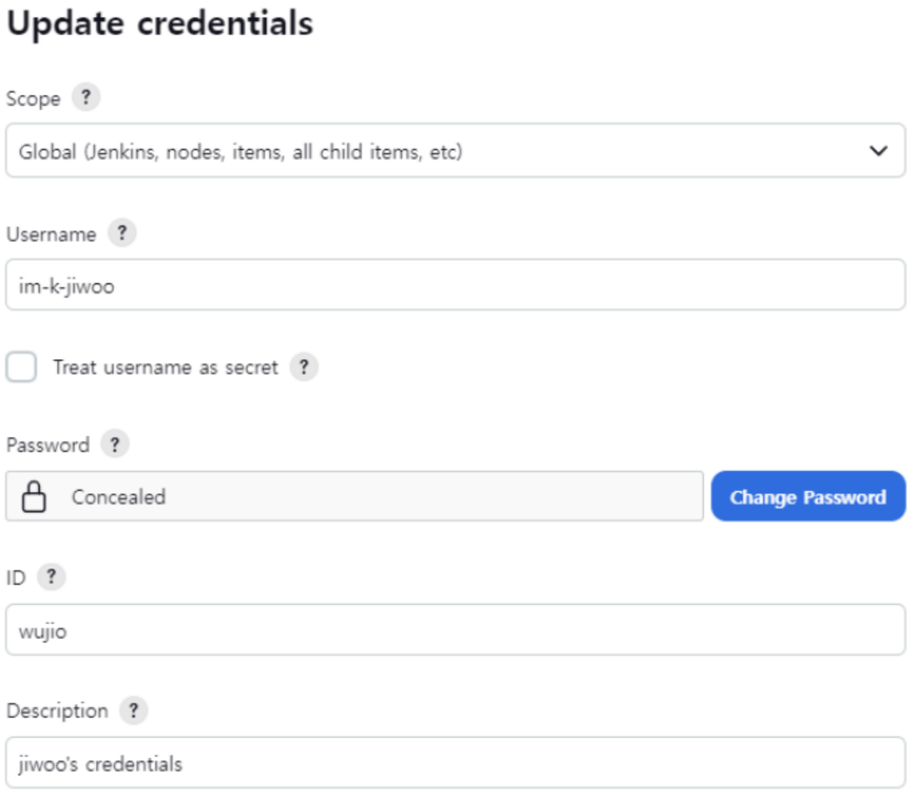


# Github Webhook 설정




▲ Github Webhook 설정

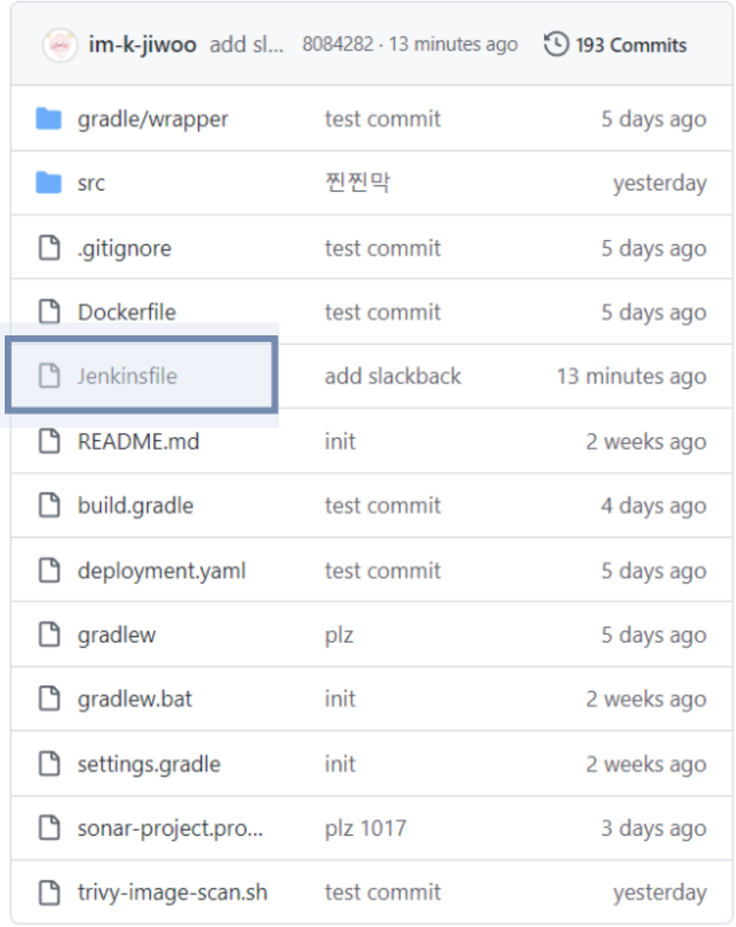

▲ Build Trigger 설정



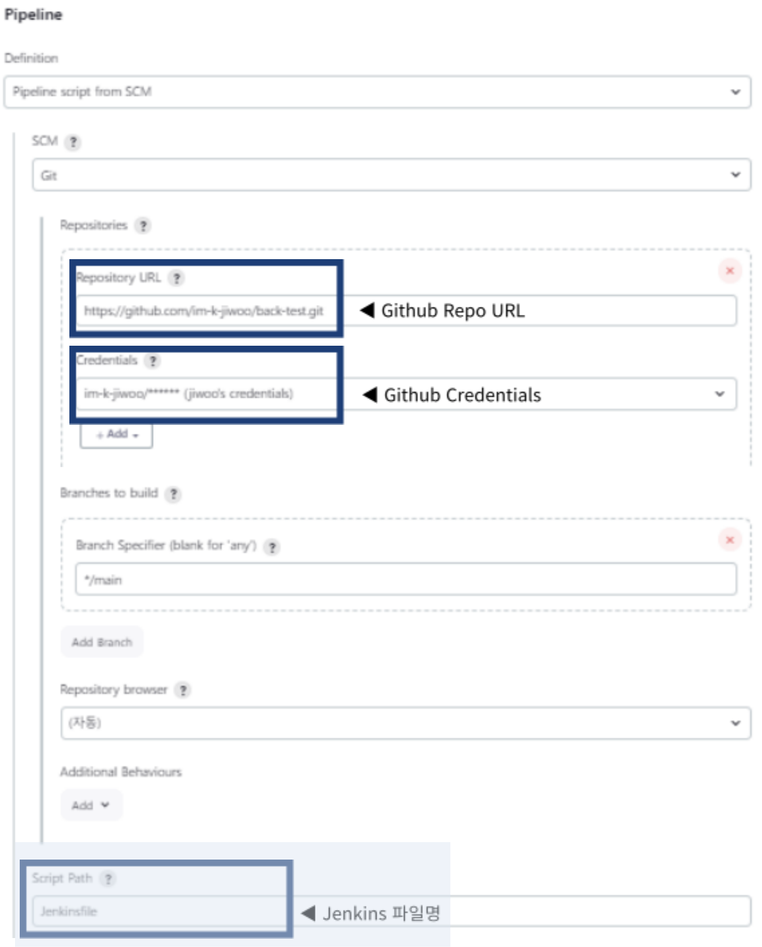

▲ Github Credential 설정

# Jenkins Pipeline 설정

MetaMedical
메타5형제




▲ Github - Jenkinsfile




▲ Jenkins Pipeline 설정

KOSA 한국소프트웨어산업협회
Metanet

# Jenkins Pipeline

MetaMedical
메타5형제

1

```
environment {
    //Azure 계정 정보 설정
    AZURE_SUBSCRIPTION_ID = 'c8ce3edc-0522-48a3-b7e4-afe8e3d731d9'
    AZURE_TENANT_ID = '4ccd6048-181f-43a0-ba5a-7f48e8a4fa35'
    CONTAINER_REGISTRY = 'goodbirdacr.azurecr.io'
    RESOURCE_GROUP = 'AKS'


    REPO = 'medicine/back'
    IMAGE_NAME = 'medicine/back:latest'
    //TAG = 'latest'
    TAG_VERSION = "v1.0.Beta"
    TAG = "${TAG_VERSION}${env.BUILD_ID}"
    NAMESPACE = 'back'
```

Azure 및 Docker 이미지 정보 설정

2

```
// 체크섭 검사
stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

// gradle 실행 권한 부여
stage('Grant Execute Permission to Gradle Wrapper') {
            steps {
                sh 'chmod +x ./gradlew'
            }
        }
```

소스 코드 체크아웃

KOSA 한국소프트웨어산업협회
Metanet

# Jenkins Pipeline

MetaMedical
메타5형제

3

```
// JAR 파일 빌드 단계 추가
stage('Build JAR') {
    steps {
        script {
            withEnv(['JAVA_HOME=/usr/lib/jvm/jdk-21.0.2']) {
                // Gradle를 사용하여 JAR 파일 빌드
                sh './gradlew --version'
                sh './gradlew clean build --warning-mode=none
                        -x test --info'
            }
        }
    }
}
```

Gradle을 통한 JAR 파일 빌드

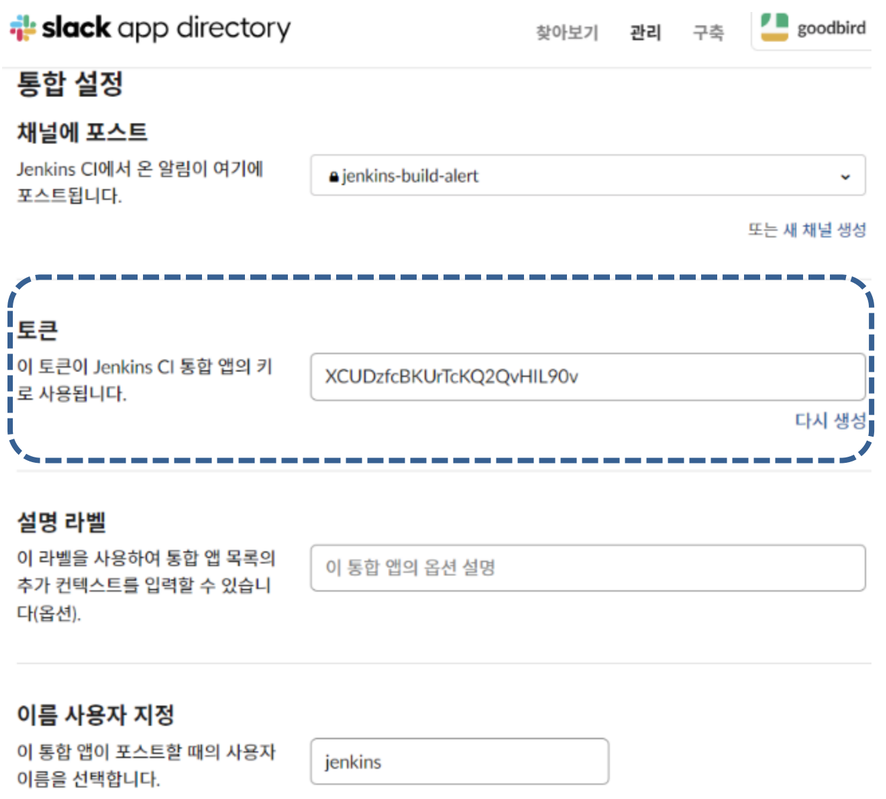
4

```
stage('Build and Push Docker Image to ACR') {
    steps {
        script {
            withCredentials([usernamePassword(
                    credentialsId: 'acr-credential-id',
                    passwordVariable: 'ACR_PASSWORD',
                    usernameVariable: 'ACR_USERNAME')]) {
                // Log in to ACR
                sh "az acr login --name $CONTAINER_REGISTRY
                    --username $ACR_USERNAME --password $ACR_PASSWORD"
                // Dockerfile에 있는 JAR 파일을 사용하여 Docker 이미지 빌드
                sh "docker build -t $REPO:$TAG ."
                // 이미지 태그 지정 및 ACR로 푸시
                 sh "docker tag $REPO:$TAG $CONTAINER_REGISTRY/$REPO:$TAG"
                sh "docker push $CONTAINER_REGISTRY/$REPO:$TAG"
            }
        }
    }
```

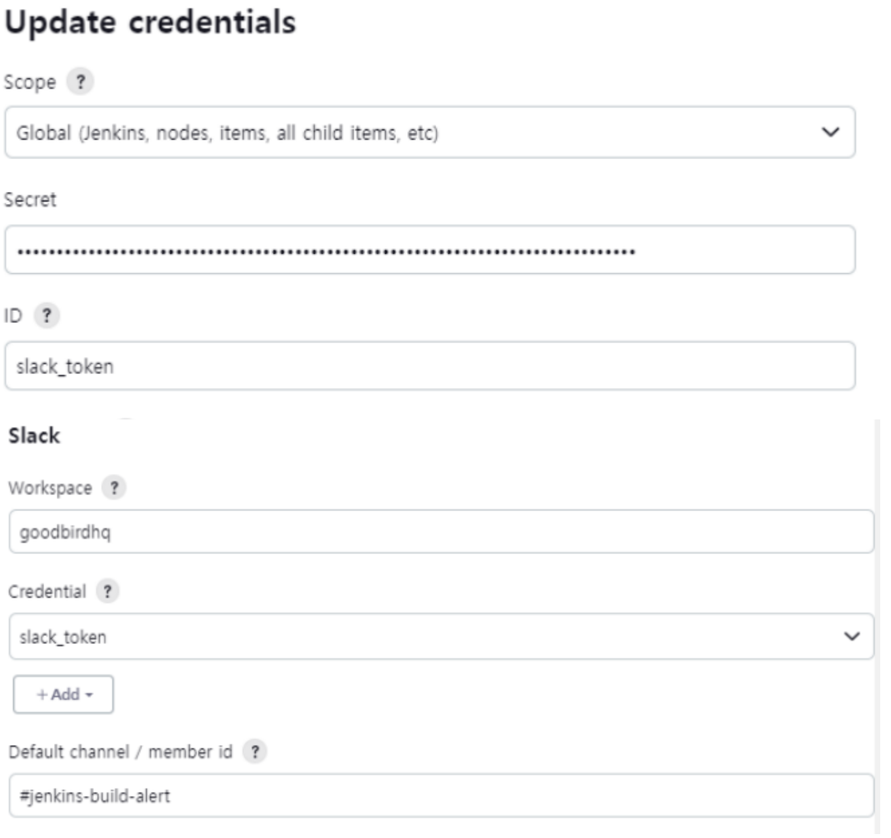
Docker 이미지 빌드 및 푸시

KOSA 한국소프트웨어산업협회
Metanet

# Jenkins+Slack 알림




▲ Slack Token 생성

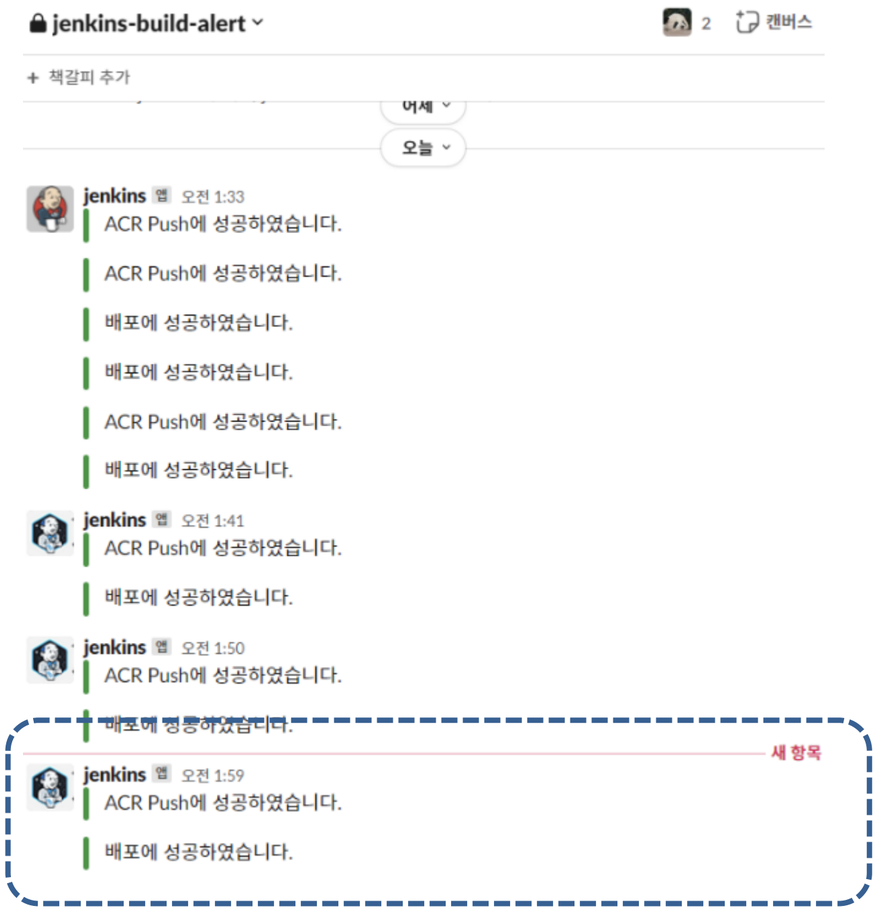



▲ Jenkins Slack Token 설정

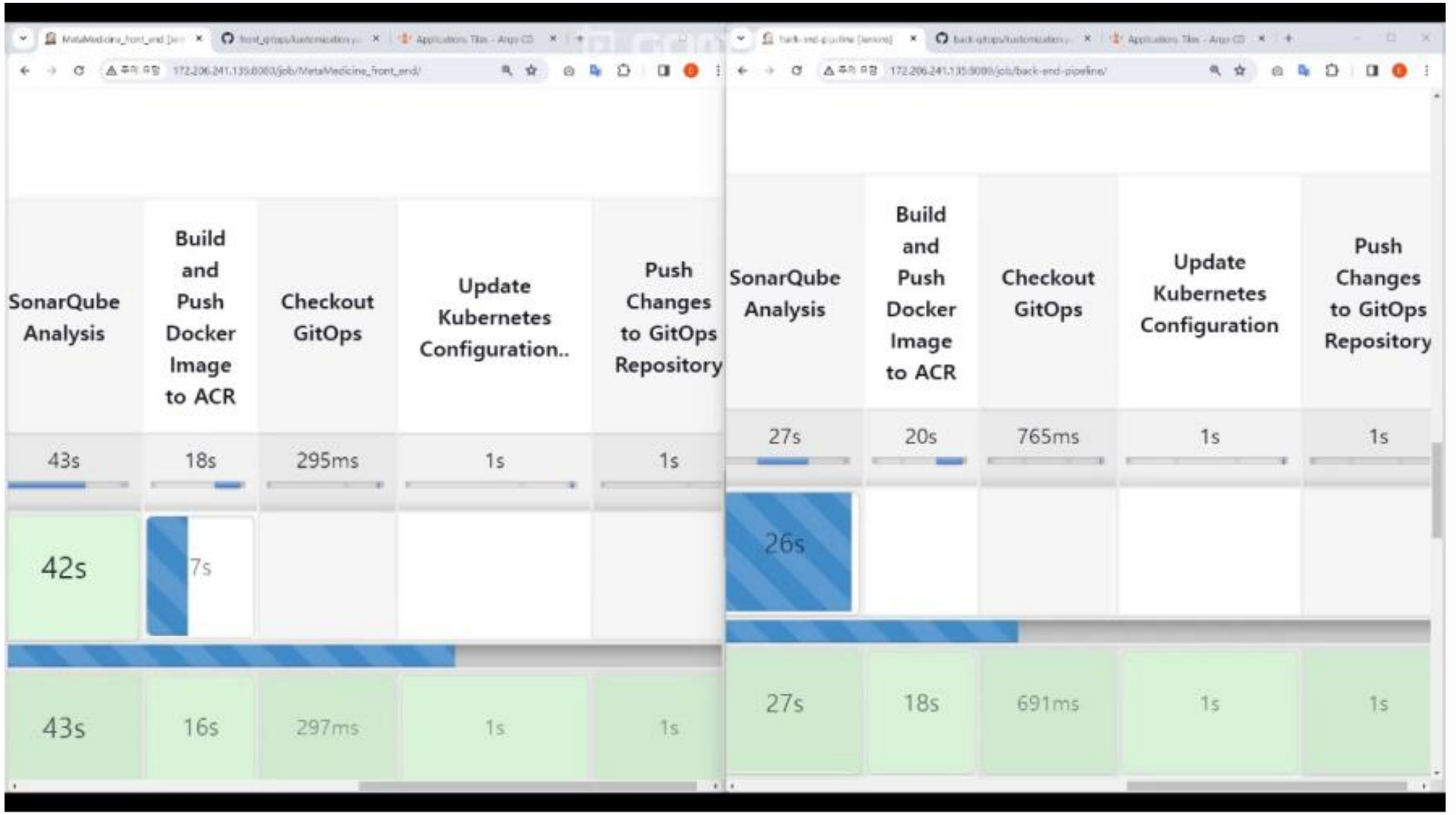
# Jenkins+Slack 알림

```
post {
    success {
        slackSend (
            channel: SLACK_CHANNEL,
            color: SLACK_SUCCESS_COLOR,
            message: "ACR Push에 성공하였습니다."
        )
    }
    failure {
        slackSend (
            channel: SLACK_CHANNEL,
            color: SLACK_FAIL_COLOR,
            message: "ACR Push에 실패하였습니다."
        )
    }
}
```

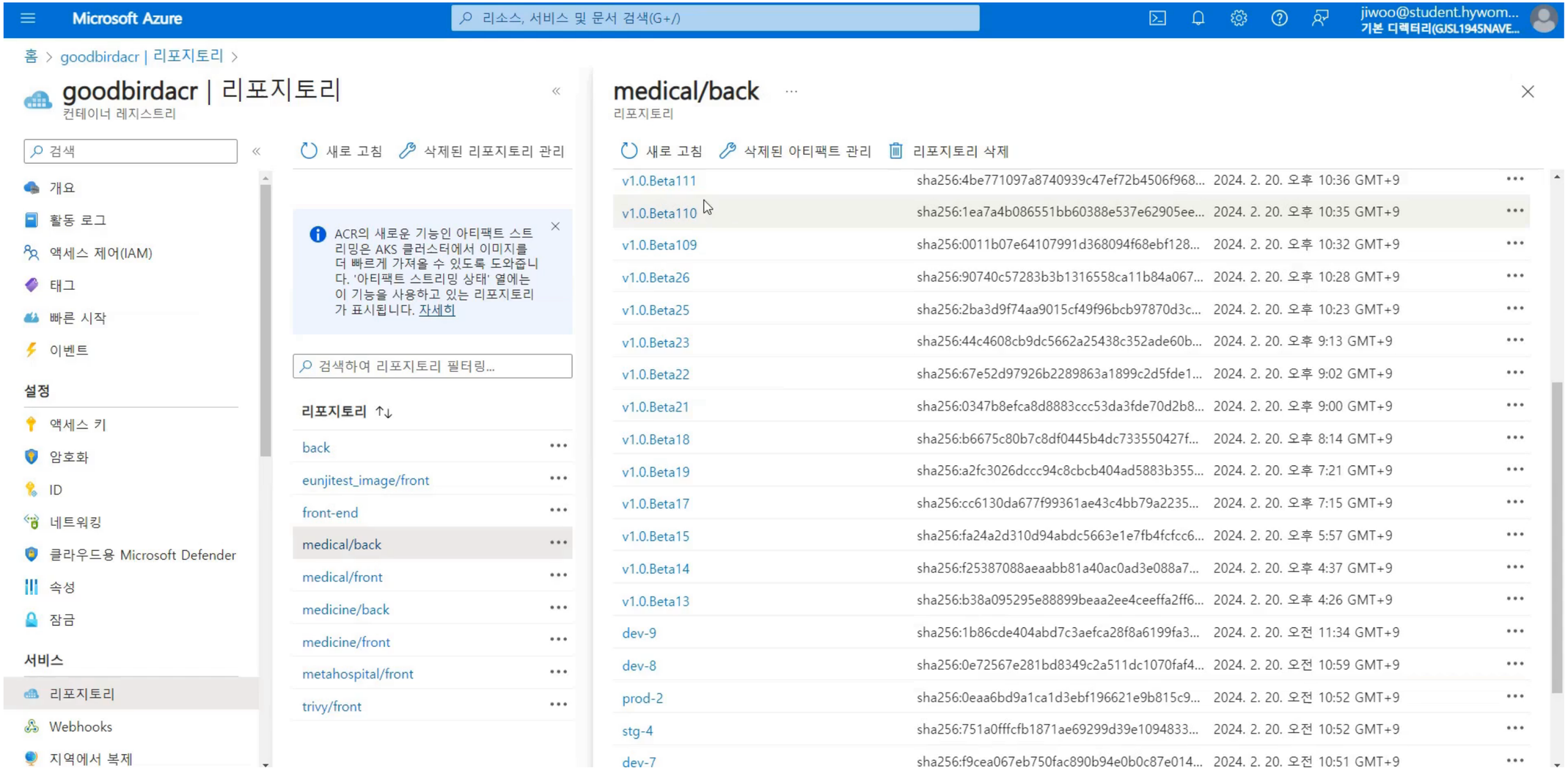
```
post {
    success {
        slackSend (
            channel: SLACK_CHANNEL,
            color: SLACK_SUCCESS_COLOR,
            message: "배포에 성공하였습니다."
        )
    }
    failure {
        slackSend (
            channel: SLACK_CHANNEL,
            color: SLACK_FAIL_COLOR,
            message: "배포에 실패하였습니다."
        )
    }
}
```



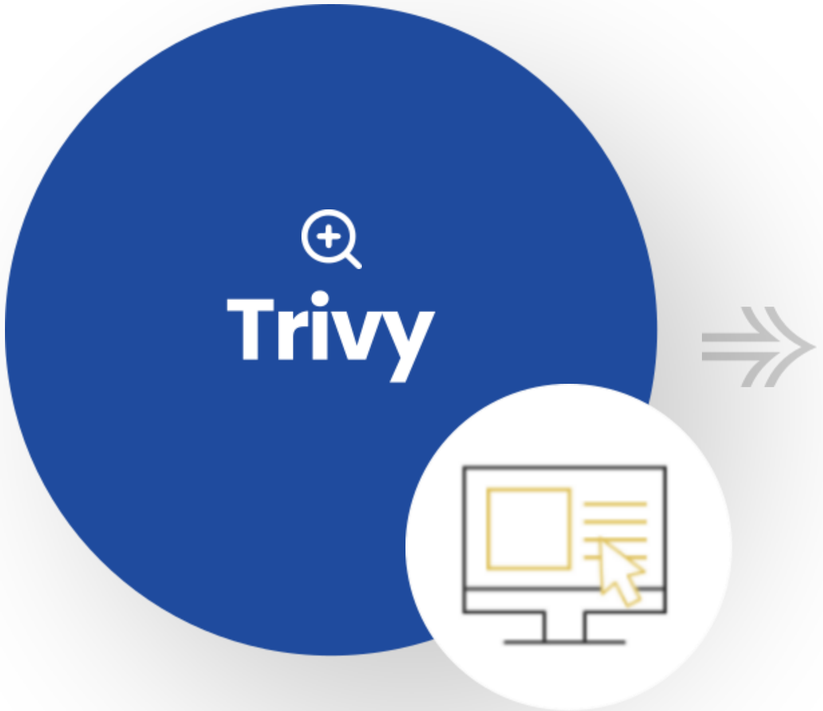
# CI 시연 영상(Build+Slack)



# CI 시연 영상(Build+Slack)



# Trivy




**이미지 빌드 시 스캔 및 스캔 결과에 따른 빌드 Fail/Success**

| Scanner | OS Packages | Application Dependencies | Easy to use | Accuracy | Suitable for CI |
|---|---|---|---|---|---|
| Trivy | ✓ | ✓ (8 languages) | ★★★ | ★★★ | ★★★ |
| Clair | ✓ | × | ★ | ★★ | ★★ |
| Anchore Engine | ✓ | ✓ (4 languages) | ★★ | ★★ | ★★★ |
| Quay | ✓ | × | ★★★ | ★★ | × |
| Docker Hub | ✓ | × | ★★★ | ★ | × |
| GCR | ✓ | × | ★★★ | ★★ | × |

▲ https://aquasecurity.github.io/trivy/v0.17.2/comparison/

# Trivy

```
#!/bin/bash
## Docker 이미지 이름
dockerImageName=$(awk 'NR==1 {print $2}' Dockerfile)
echo $dockerImageName

## Trivy 이미지 스캔 실행
# -v $WORKSPACE:/root/.cache/: Trivy 캐시를 Jenkins 워크스페이스로 캐시 공유
# -q: Trivy의 출력 간소화
# --exit-code 1: 취약점이 발견되면 'exit_code' 1 반환
# --severity CRITICAL: 심각도가 CRITICAL인 취약점만 보고
# --light: 가볍게 스캔, 패키지 버전만 확인
docker run --rm -v $WORKSPACE:/root/.cache/ -e
TRIVY_GITHUB_TOKEN="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"
aquasec/trivy:0.17.2 -q image --exit-code 1
--severity CRITICAL --light $dockerImageName

## Trivy 스캔 결과 처리
exit_code=$?
echo "종료 코드: $exit_code"
#
## 스캔 결과 확인
if [[ "$exit_code" == 1 ]]; then
    echo "이미지 스캔에 실패했습니다. CRITICAL 취약점이 발견되었습니다."
    exit 1
else
    echo "이미지 검사가 통과되었습니다. CRITICAL 취약점이 발견되지 않았습니다."
fi
```

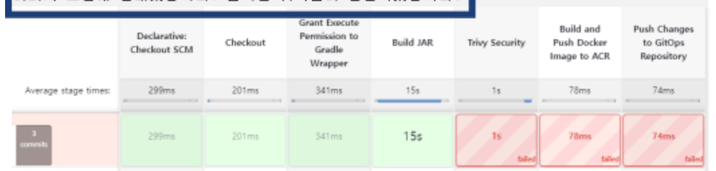
**O**

```
종료 코드: 1
./trivy-image-scan.sh: 12: [[: not found
이미지 검사가 통과되었습니다. 심각한 취약점이 발견되지 않았습니다.
```

**X**

```
Total: 5 (CRITICAL: 5)

+-------------------------+------------------+----------+-------------------+----------
-----------------+
|        LIBRARY          | VULNERABILITY ID | SEVERITY | INSTALLED VERSION |
FIXED VERSION        |
+-------------------------+------------------+----------+-------------------+----------
-----------------+
| glibc                   | CVE-2019-9169    | CRITICAL | 2.28-225.0.3.el8  | 2:2.28-
151.0.1.ksplice2.el8 |
+-------------------------+                  +          +                   +
+
| glibc-common            |                  |          |                   |
|
+-------------------------+                  +          +                   +
+
| glibc-minimal-langpack  |                  |          |                   |
|
+-------------------------+------------------+          +-------------------+----------
-----------------+
| gnutls                  | CVE-2021-20231   |          | 3.6.16-6.el8_7    | 10:3.6.16-
4.0.1.el8_fips    |
+                         +------------------+          +                   +
+
|                         | CVE-2021-20232   |          |                   |
|
+-------------------------+------------------+----------+-------------------+----------
-----------------+
종료 코드: 1
```

이미지 스캔에 실패했습니다. 심각한 취약점이 발견되었습니다.



MetaMedical
메타5형제

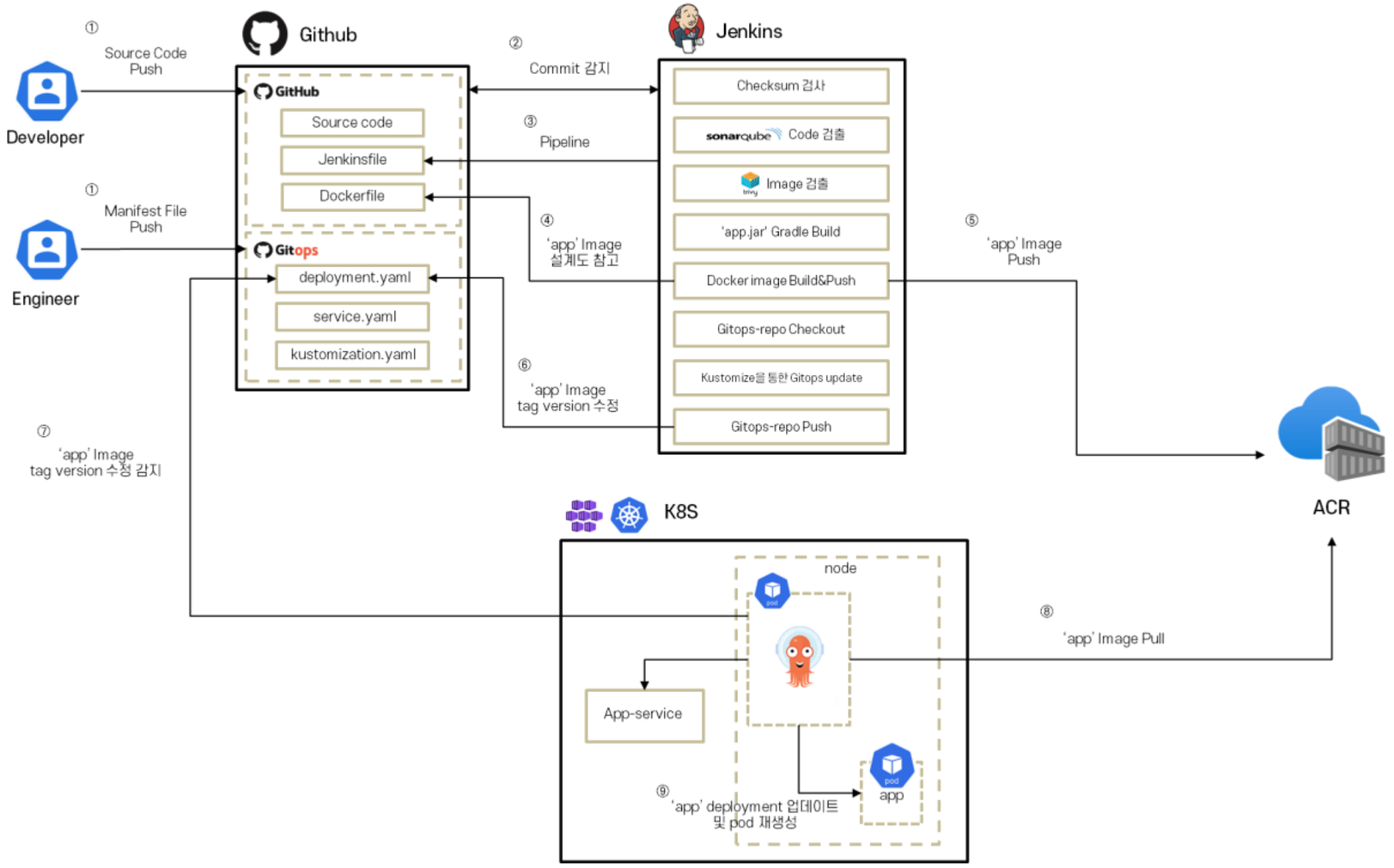
# CICD

## ArgoCD 기반 CD 자동화 및 코드 점검

**발표자: 이은지**

KOSA 한국소프트웨어산업협회
Metanet

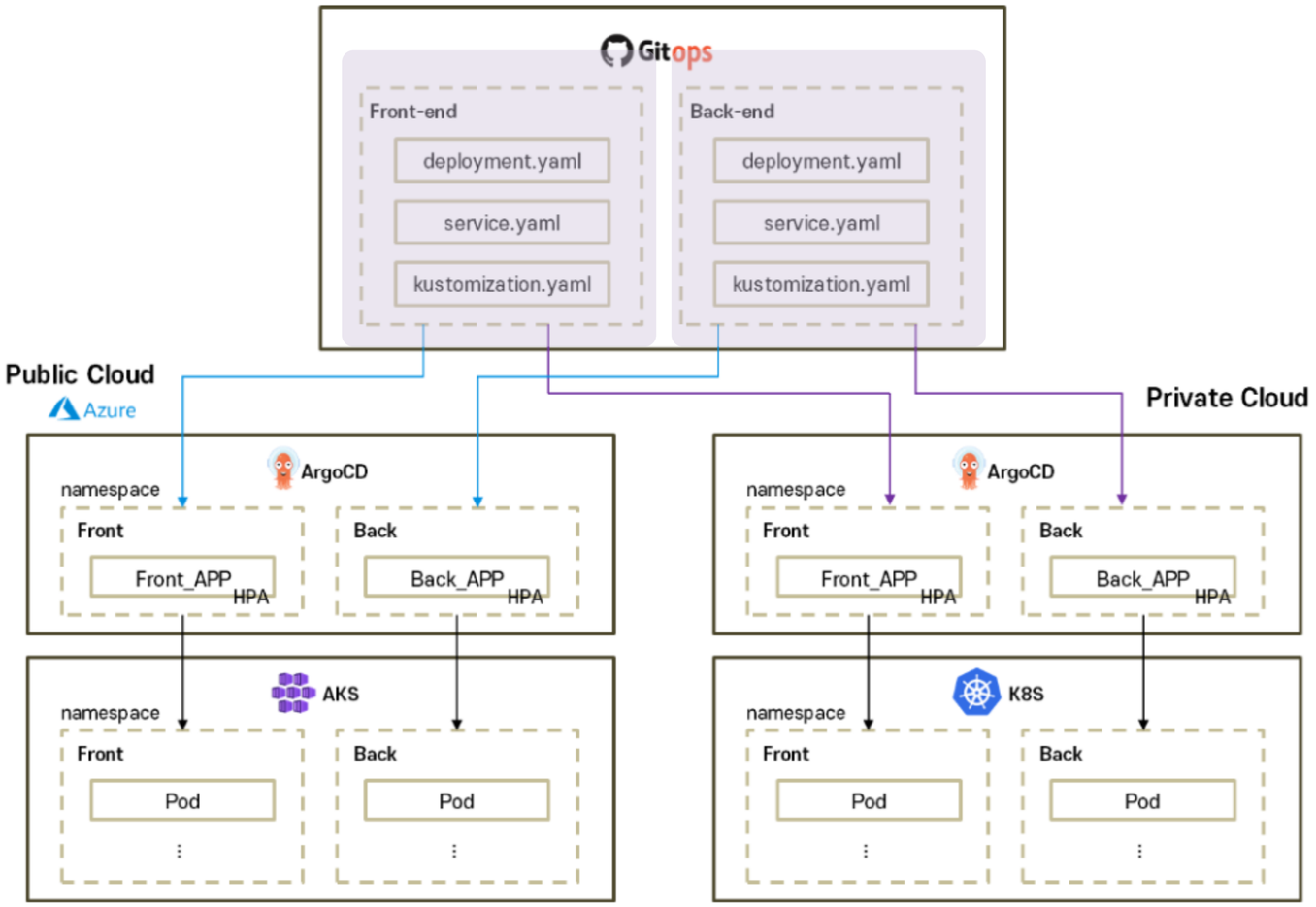
# CICD Architecture




MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet

# Front – Back 배포 과정




1. Front repo – Back repo 로 분리

2. Code repo – K8S Cluster repo 로 분리

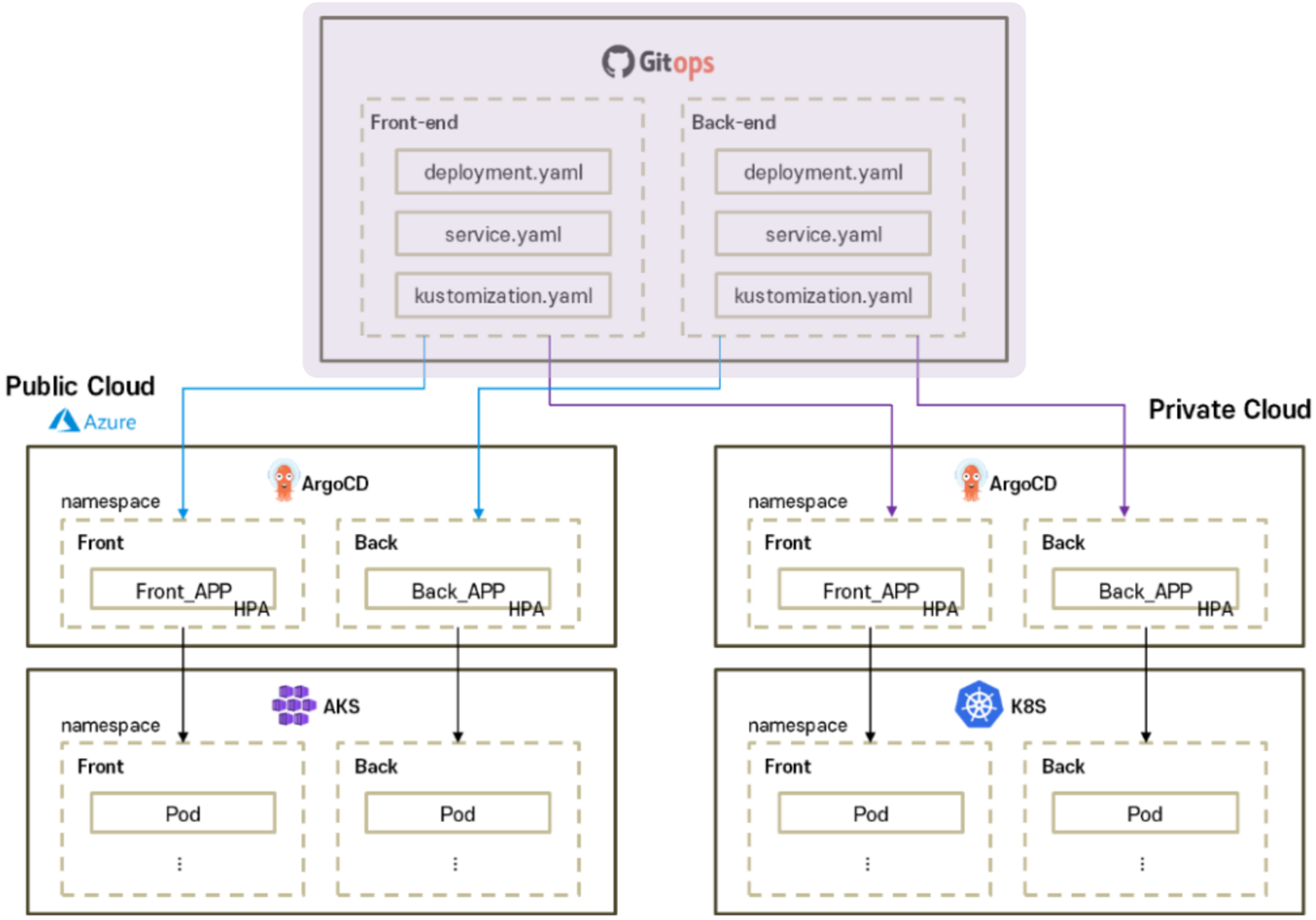
➡ 유지보수와 확장성을 향상

Namespace로 argocd application 구분

➡ AKS에 Namespace로 구분하여 배포

★ Hybrid Cloud를 트래픽에 따라
탄력적으로 확장하기 위한 인프라 구축

# Front – Back 배포 과정




1. Front repo – Back repo 로 분리

2. Code repo – K8S Cluster repo 로 분리

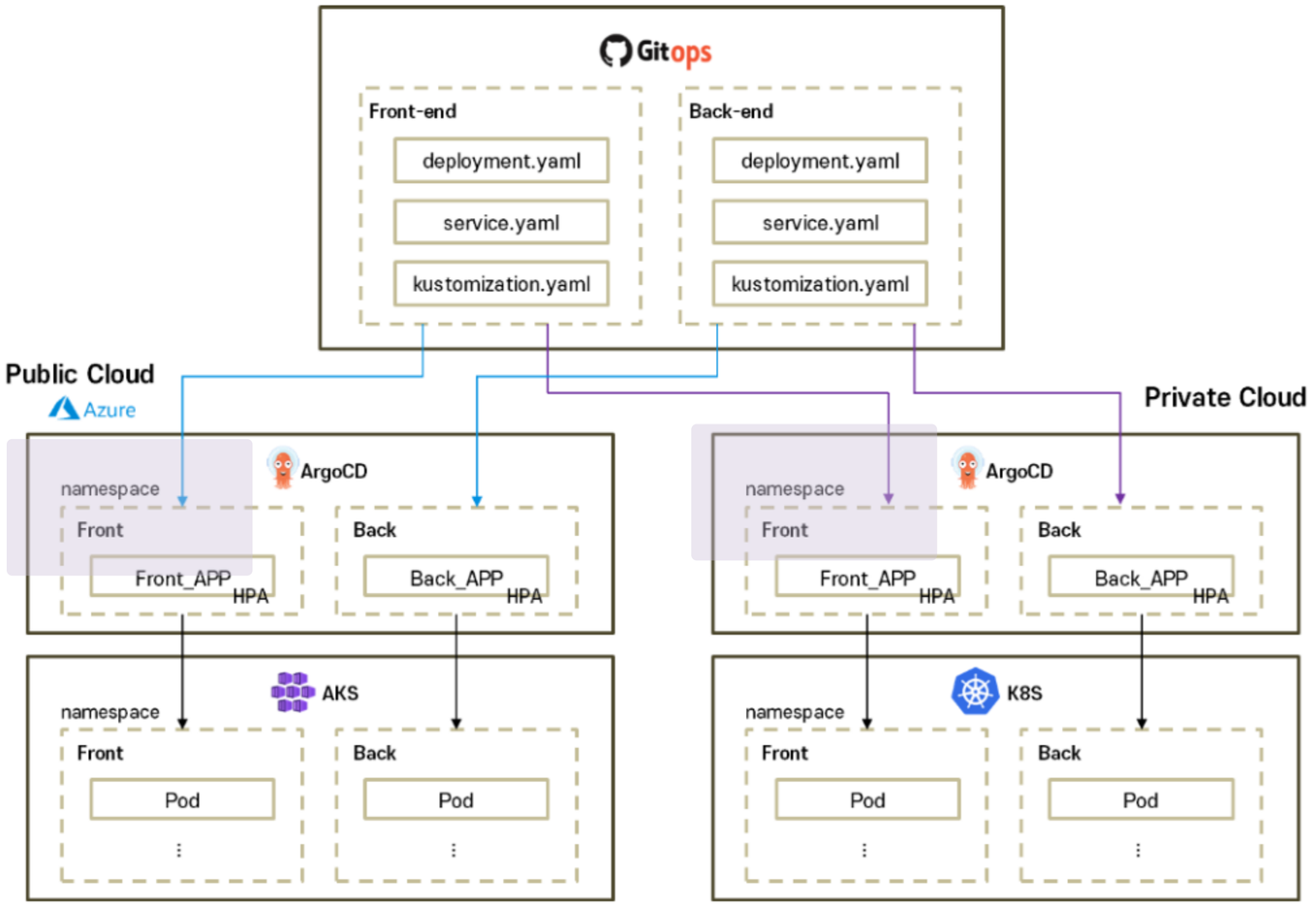
➡ 유지보수와 확장성을 향상

Namespace로 argocd application 구분

➡ AKS에 Namespace로 구분하여 배포

★ Hybrid Cloud를 트래픽에 따라
탄력적으로 확장하기 위한 인프라 구축

# Front – Back 배포 과정




1. Front repo – Back repo 로 분리

2. Code repo – K8S Cluster repo 로 분리

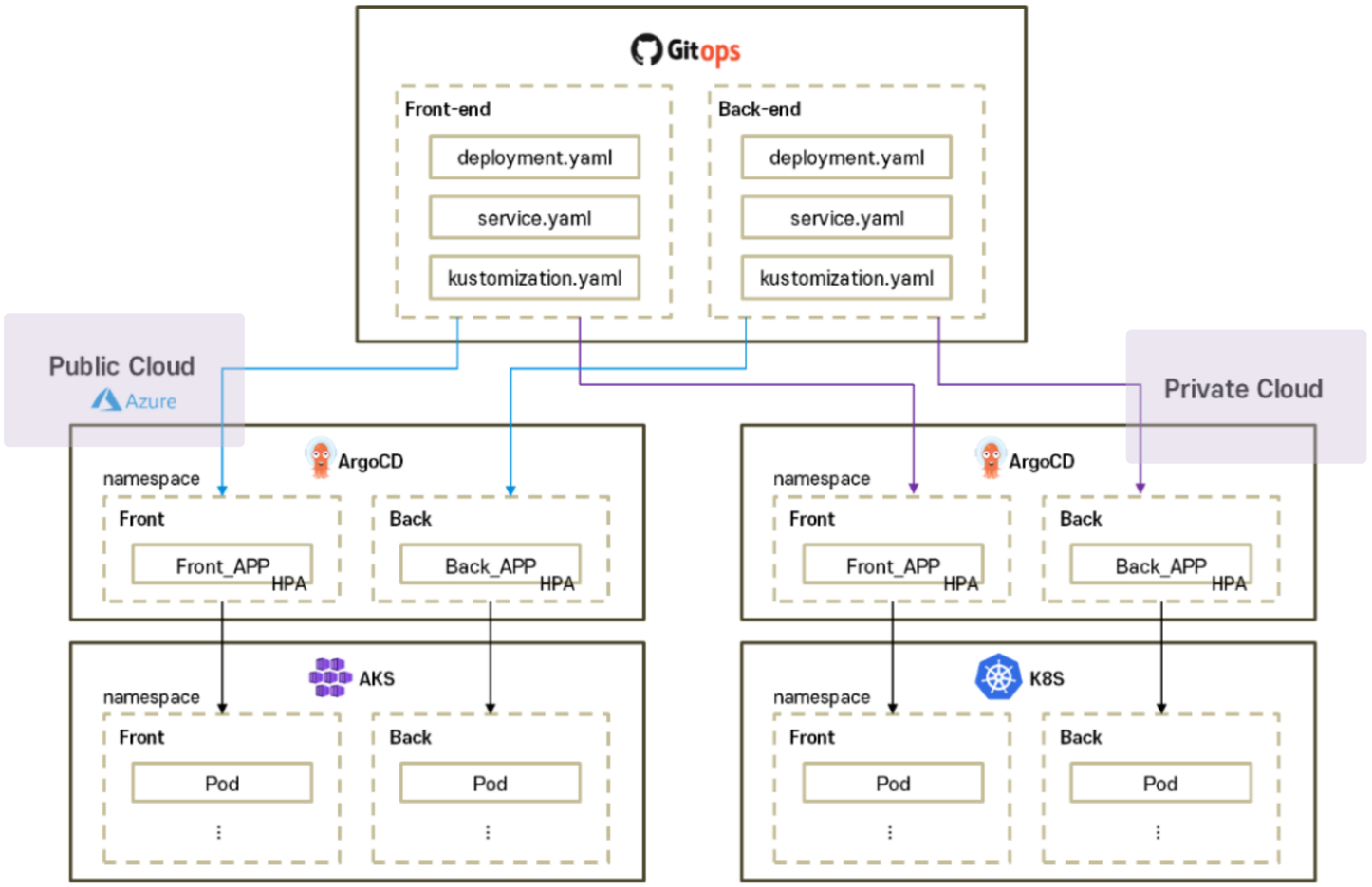
➡ 유지보수와 확장성을 향상

Namespace로 argocd application 구분

➡ AKS에 Namespace로 구분하여 배포

★ Hybrid Cloud를 트래픽에 따라
탄력적으로 확장하기 위한 인프라 구축

# Front – Back 배포 과정




1. Front repo – Back repo 로 분리

2. Code repo – K8S Cluster repo 로 분리

➡ 유지보수와 확장성을 향상

Namespace로 argocd application 구분

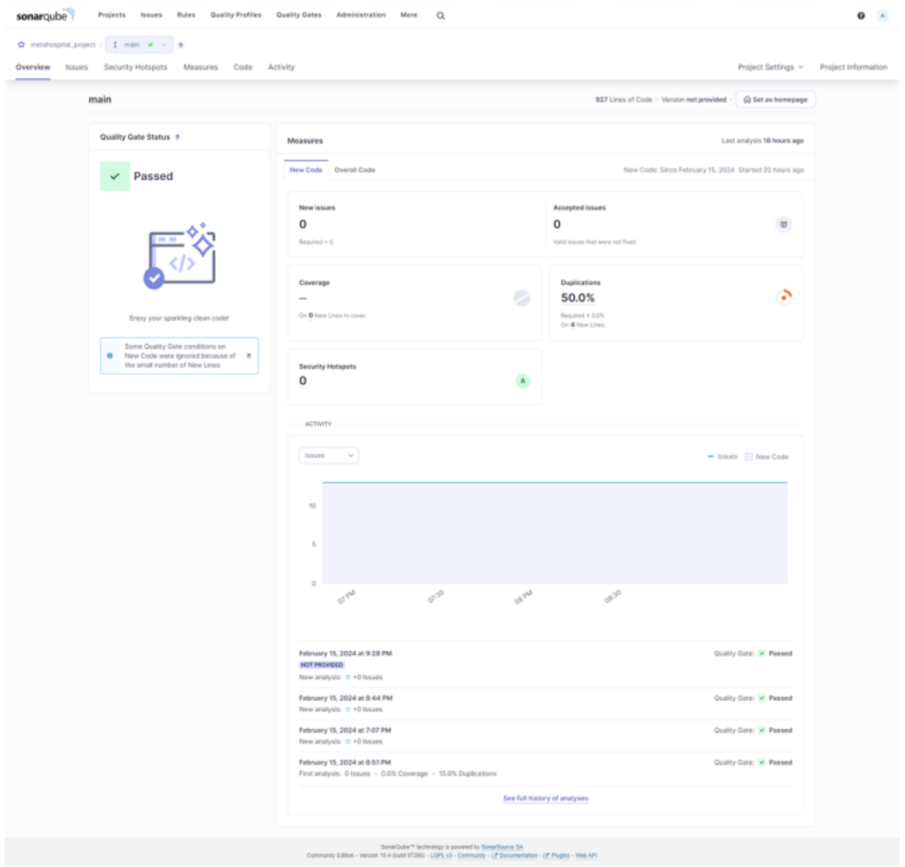
➡ AKS에 Namespace로 구분하여 배포

★ Hybrid Cloud를 트래픽에 따라
탄력적으로 확장하기 위한 인프라 구축

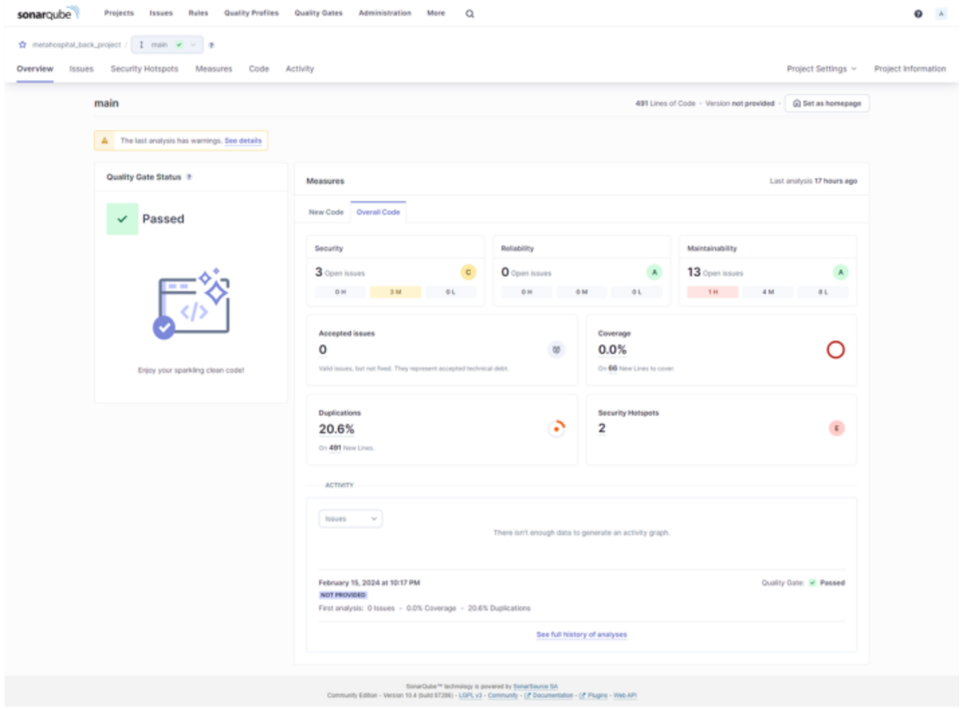
# SonarQube – 코드 검증

MetaMedical
메타5형제

## Front




## Back




KOSA 한국소프트웨어산업협회
Metanet

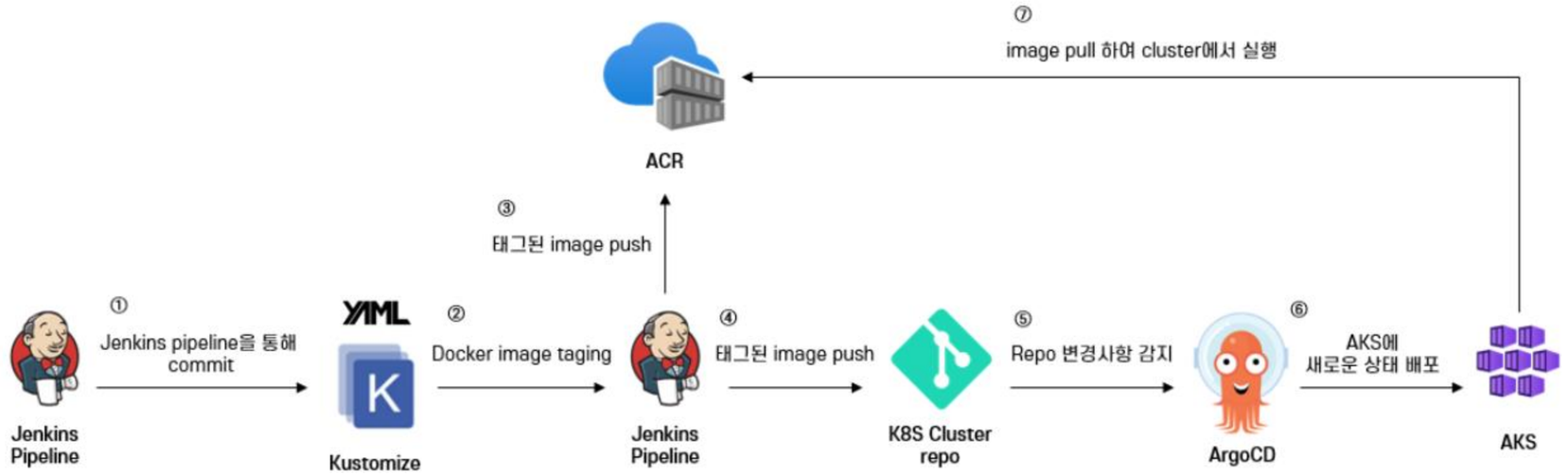
# Kustomize

Github
K8S Cluster repo

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- deployment.yaml
- service.yaml

images:
- name: goodbirdacr.azurecr.io/medicine/front
  newName: goodbirdacr.azurecr.io/medicine/front
  newTag: v1.0.Beta4
```

- Kustomization.yaml 파일
- Docker image의 tag 변경

→ ArgoCD가 변경사항 감지



# Kustomize

Jenkins
Pipeline

## Jenkinsfile 환경설정

```
environment {
    AZURE_SUBSCRIPTION_ID = 'c8ce3edc-0522-48a3-b7e4-afe8e3d731d9'
    AZURE_TENANT_ID = '4ccd6048-181f-43a0-ba5a-7f48e8a4fa35'
    CONTAINER_REGISTRY = 'goodbirdacr.azurecr.io'
    RESOURCE_GROUP = 'AKS'
    REPO = 'medicine/front'
    IMAGE_NAME = 'medicine/front:latest'

    TAG_VERSION = "v1.0.Beta"
    TAG = "${TAG_VERSION}${env.BUILD_ID}"
    NAMESPACE = 'front'
    GIT_CREDENTIALS_ID = 'jenkins-git-access'
}
```

Github
K8S Cluster repo

## Kustomization.yaml 파일

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- deployment.yaml
- service.yaml

images:
- name: goodbirdacr.azurecr.io/medicine/front
  newName: goodbirdacr.azurecr.io/medicine/front
  newTag: latest
```

## Jenkinsfile Kustomize 실행

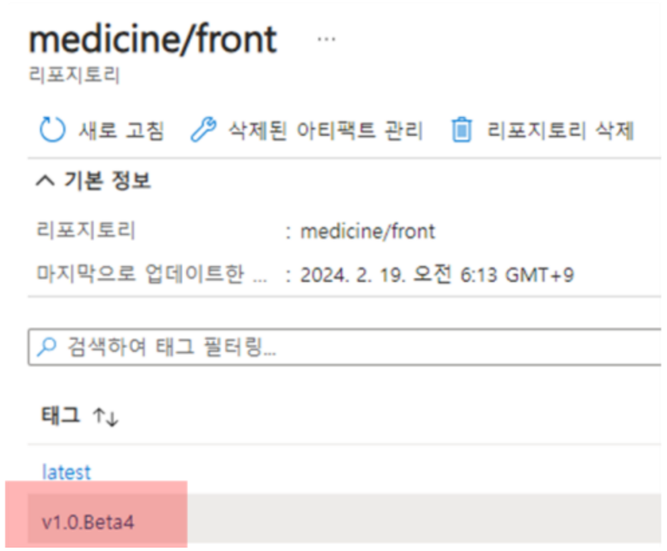
```
stage('Update Kubernetes Configuration..') {
            steps {
                script {
                    sh "kustomize edit set image ${CONTAINER_REGISTRY}/${REPO}=${CONTAINER_REGISTRY}/${REPO}:${TAG}"
                    sh "git add ."
                    sh "git commit -m 'Update image to ${TAG}'"
                }
            }
        }
```

# Kustomize

Jenkins Pipeline

```
stage('Build and Push Docker Image to ACR') {
    steps {
        script {
            withCredentials([usernamePassword(credentialsId: 'acr-credential-id', passwordVariable: 'ACR_PASSWORD', usernameVariable: 'ACR_USERNAME')]) {
                sh "az acr login --name $CONTAINER_REGISTRY --username $ACR_USERNAME --password $ACR_PASSWORD"

                sh "docker build -t $CONTAINER_REGISTRY/$REPO:$TAG ."
                sh "docker push $CONTAINER_REGISTRY/$REPO:$TAG"
            }
        }
    }
}
```

ACR




- Jenkinsfile 에서 ACR에 push

→ IMAGE 태그 변경

# Kustomize

Jenkins
Pipeline

```
stage('Push Changes to GitOps Repository') {
    steps {
        script {
            withCredentials([usernamePassword(credentialsId: 'jenkins-git-access', passwordVariable: 'GIT_PASSWORD', usernameVariable: 'GIT_USERNAME')]) {
                // 현재 브랜치 확인 및 main으로 체크아웃
                def currentBranch = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim()
                if (currentBranch != "main") {
                    sh "git checkout main"
                }
                // 원격 저장소에서 최신 변경사항 가져오기
                sh "git pull --rebase origin main"
                def remote = "https://${GIT_USERNAME}:${GIT_PASSWORD}@github.com/JoEunSae/front-end.git"
                // 원격 저장소에 푸시
                sh "git push https://${GIT_USERNAME}:${GIT_PASSWORD}@github.com/rlozi99/front_gitops.git main"
            }
        }
    }
```

Github
K8S Cluster repo

front_gitops / kustomization.yaml

Jenkins and Jenkins Update image to v1.0.Beta4

Code | Blame | 11 lines (9 loc) · 234 Bytes | Code 55% faster with GitHub

```
1   apiVersion: kustomize.config.k8s.io/v1beta1
2   kind: Kustomization
3
4   resources:
5   - deployment.yaml
6   - service.yaml
7
8   images:
9   - name: goodbirdacr.azurecr.io/medicine/front
10    newName: goodbirdacr.azurecr.io/medicine/front
11    newTag: v1.0.Beta4
```

- Jenkinsfile 에서 git에 push

→ Kustomization.yaml 파일 태그 변경

# ArgoCD Slack 알림

**ArgoCD** 앱 오후 11:28

✅ Application back-gitops-app has been successfully synced at 2024-02-18T14:28:25Z.
Sync operation details are available at: https://localhost:4000/applications/back-gitops-app?operation=true .

back-gitops-app

| Sync Status | Repository |
| --- | --- |
| Synced | https://github.com/rlozi99/back-gitops.git |

✅ Application back-gitops-app is now running new version of deployments manifests.

back-gitops-app

| Sync Status | Repository |
| --- | --- |
| Synced | https://github.com/rlozi99/back-gitops.git |

**Revision**
8803ff2aa5812f7fc7c3331368d3a69c2a8c31a8

**ArgoCD** 앱 오전 5:15

❗ The sync operation of application front-gitops-app has failed at 2024-02-17T20:15:20Z with the following error: one or more synchronization tasks are not valid (retried 5 times).
Sync operation details are available at: https://localhost:4000/applications/front-gitops-app?operation=true .

front-gitops-app

| Sync Status | Repository |
| --- | --- |
| OutOfSync | https://github.com/rlozi99/front_gitops.git |

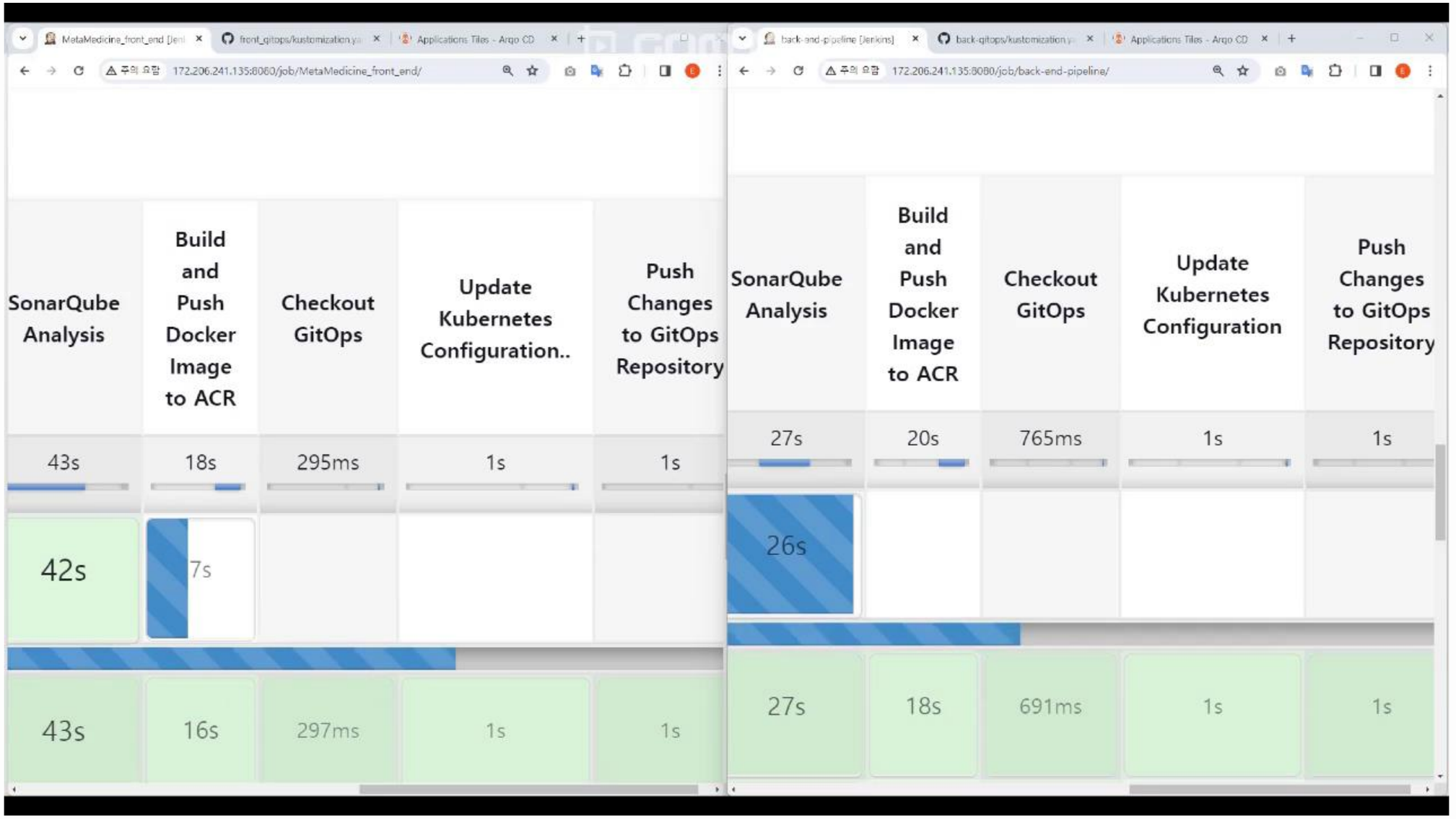
**ArgoCD** 앱 오전 5:20

❗ The sync operation of application front-gitops-app has failed at 2024-02-17T20:20:31Z with the following error: one or more synchronization tasks are not valid (retried 5 times).
Sync operation details are available at: https://localhost:4000/applications/front-gitops-app?operation=true .

front-gitops-app

| Sync Status | Repository |
| --- | --- |
| OutOfSync | https://github.com/rlozi99/front_gitops.git |

# 시연 영상


MetaMedicine_front_end [Jen
front_gitops/kustomization.ya
Applications Tiles - Argo CD
주의 요함 172.206.241.135:8060/job/MetaMedicine_front_end/
SonarQube Analysis
Build and Push Docker Image to ACR
Checkout GitOps
Update Kubernetes Configuration..
Push Changes to GitOps Repository
43s
18s
295ms
1s
1s
42s
7s
43s
16s
297ms
1s
1s
back-end-pipeline [Jenkins]
back-gitops/kustomization.y
Applications Tiles - Argo CD
주의 요함 172.206.241.135:8080/job/back-end-pipeline/
SonarQube Analysis
Build and Push Docker Image to ACR
Checkout GitOps
Update Kubernetes Configuration
Push Changes to GitOps Repository
27s
20s
765ms
1s
1s
26s
27s
18s
691ms
1s
1s

MetaMedical
메타5형제

# Monitoring

## 서비스 성능 및 장애 관리

**발표자: 육지현**

KOSA 한국소프트웨어산업협회
Metanet

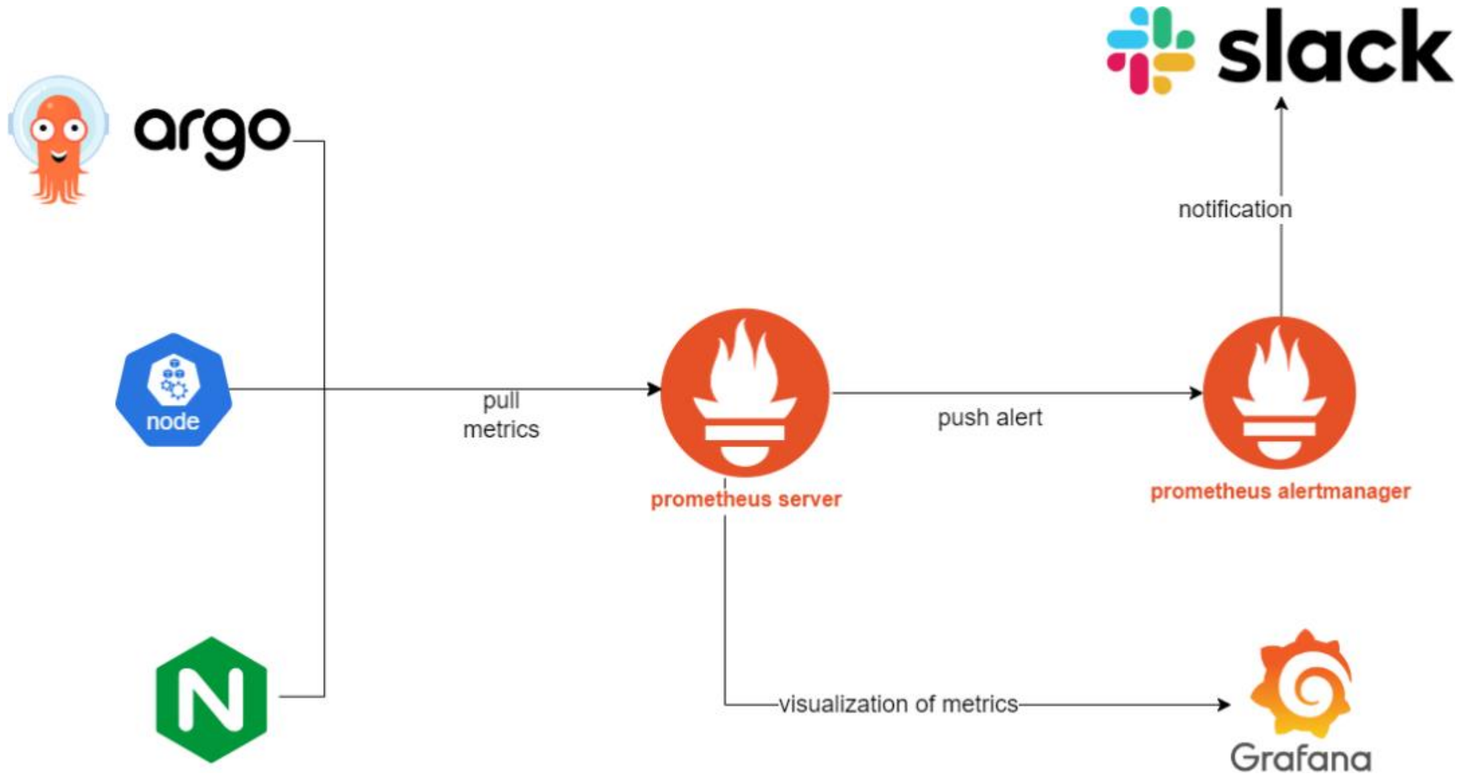
# Monitoring Architecture

**Prometheus**

**Exporter**

**Grafana**

**Prometheus Alertmanager (Slack)**



# Exporter 설치

MetaMedical
메타5형제

Nginx exporter

```
- name: metrics
  image: nginx/nginx-prometheus-exporter
  resources:
    requests:
      cpu: 10m
      memory: 50Mi
    limits:
      cpu: 100m
      memory: 50Mi
  ports:
    - containerPort: 9113
```

```
- name: metrics
  protocol: TCP
  port: 9113
  targetPort: metrics
```

Nginx

```
location /metrics {
    stub_status on;
    access_log off;
    allow all;
}
```

```
- name: metrics
  args:
  - --nginx.scrape-uri
  - http://127.0.0.1:80/metrics
  image: nginx/nginx-prometheus-exporter
  resources:
    requests:
      cpu: 10m
      memory: 100Mi
    limits:
      cpu: 30m
      memory: 500Mi
  ports:
  - containerPort: 9113
imagePullSecrets:
- name: goodbirdacr
```

# Exporter 설치

MetaMedical
메타5형제

Mysql

```
    spec:
      containers:
      - name: mysqld-exporter
        image: prom/mysqld-exporter:v0.12.1
        args:
        - --collect.info_schema.tables
        - --collect.info_schema.innodb_tablespaces
        - --collect.info_schema.innodb_metrics
        - --collect.global_status
        - --collect.global_variables
        - --collect.slave_status
        - --collect.info_schema.processlist
        - --collect.perf_schema.tablelocks
        - --collect.perf_schema.eventsstatements
        - --collect.perf_schema.eventsstatementssum
        - --collect.perf_schema.eventswaits
        - --collect.auto_increment.columns
        - --collect.binlog_size
        - --collect.perf_schema.tableiowaits
        - --collect.perf_schema.indexiowaits
        - --collect.info_schema.userstats
        - --collect.info_schema.clientstats
        - --collect.info_schema.tablestats
        - --collect.info_schema.schemastats
        - --collect.perf_schema.file_events
        - --collect.perf_schema.file_instances
        - --collect.perf_schema.replication_group_member_stats
        - --collect.perf_schema.replication_applier_status_by_worker
        - --collect.slave_hosts
        - --collect.info_schema.innodb_cmp
        - --collect.info_schema.innodb_cmpmem
        - --collect.info_schema.query_response_time
        - --collect.engine_tokudb_status
        - --collect.engine_innodb_status
        ports:
        - containerPort: 9104
          protocol: TCP
        env:
        - name: DATA_SOURCE_NAME
          value: "mysqld-exporter:123456@(mysql-svc:3306)/"
```

Node exporter

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: node-exporter
  namespace: monitoring
  labels:
    k8s-app: node-exporter
spec:
  selector:
    matchLabels:
      k8s-app: node-exporter
  template:
    metadata:
      labels:
        k8s-app: node-exporter
    spec:
      containers:
      - name: node-exporter
        image: quay.io/prometheus/node-exporter:v1.2.2
        ports:
        - name: http
          containerPort: 910
```

KOSA 한국소프트웨어산업협회
Metanet

# Prometheus

- 메트릭 수집 대상 지정

```
- job_name: 'nginx'
  static_configs:
    - targets: ['front-end-service.front.svc:9113']

- job_name: 'node-exporter'
  static_configs:
    - targets: ['node-exporter.exporter.svc:9100']

- job_name: 'argocd'
  static_configs:
    - targets: ['argocd-server-metrics.argocd.svc:8083', 'argocd-repo-server.argocd.svc:8084']
```

- alertmanager 설정

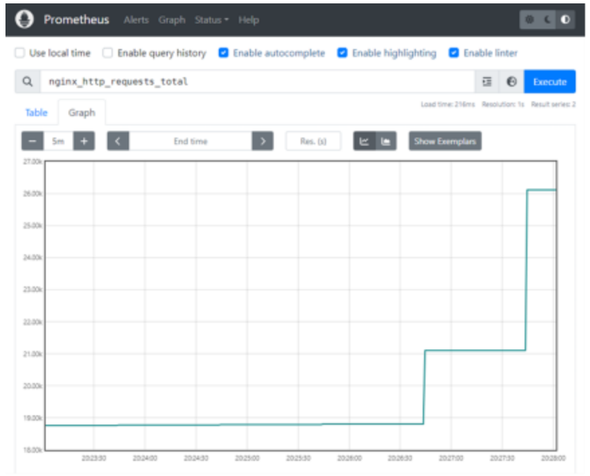
```
config:
  global:
    resolve_timeout: 5m
    slack_api_url: 'https://hooks.slack.com/services/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'
  receivers:
  - name: 'slack-notifications'
    slack_configs:
    - channel: '# project-alert'
      send_resolved: true
```

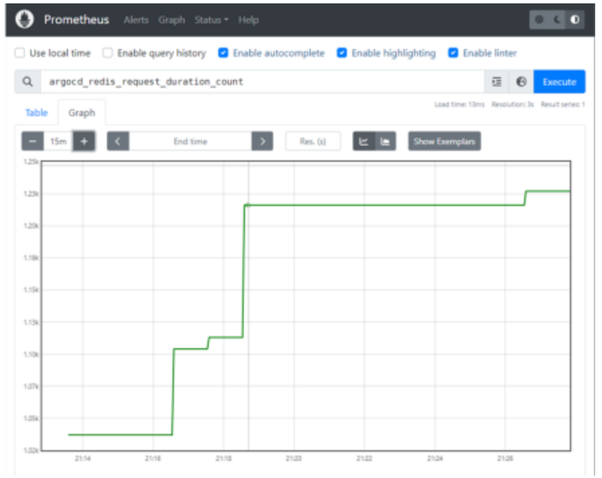
```
- name: node-exporter
  rules:
    - alert: HighCPUusage
      expr: sum without(mode) (avg without (cpu) (rate(node_cpu_seconds_total{job="prometheus", mode!="idle"}[2m]))) * 100 > 80
      for: 5m
      labels:
        severity: warning
      annotations:
        description: 'The Nginx server is experiencing a high connection rate.'
        summary: 'High Nginx Site Access'
```

MetaMedical
메타5형제

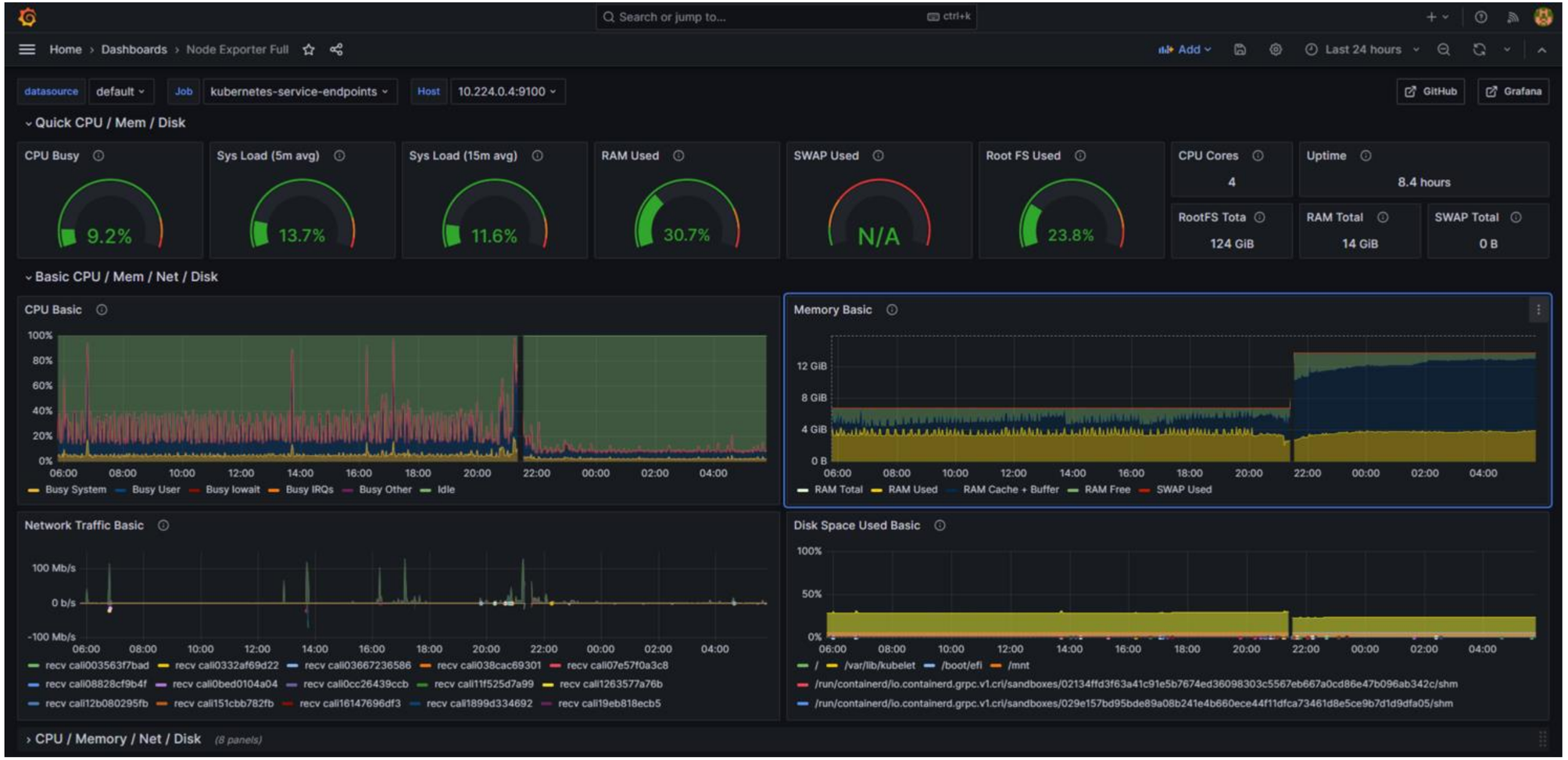
KOSA 한국소프트웨어산업협회
Metanet

# Prometheus Metrics&Alert status



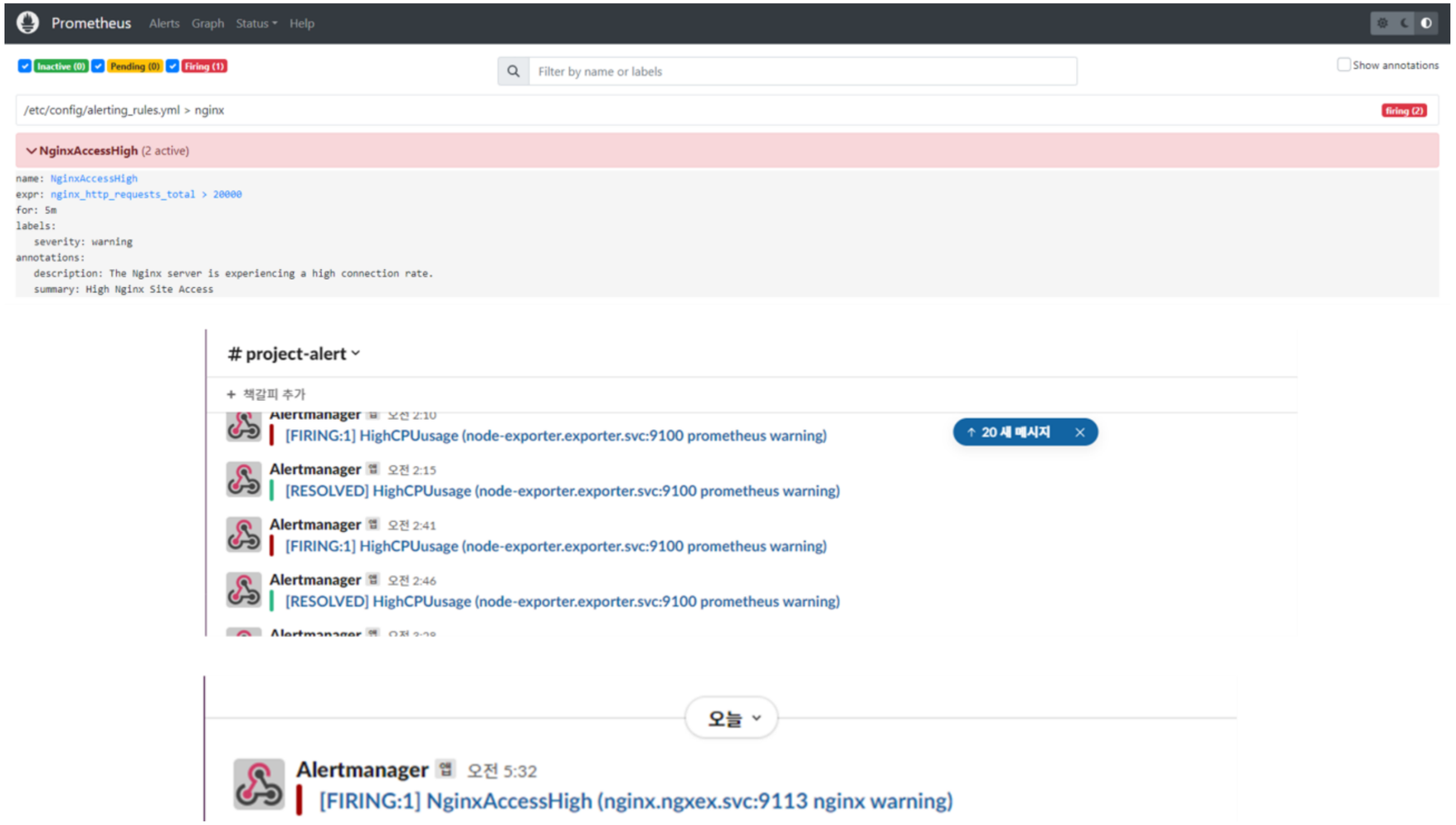




# Grafana

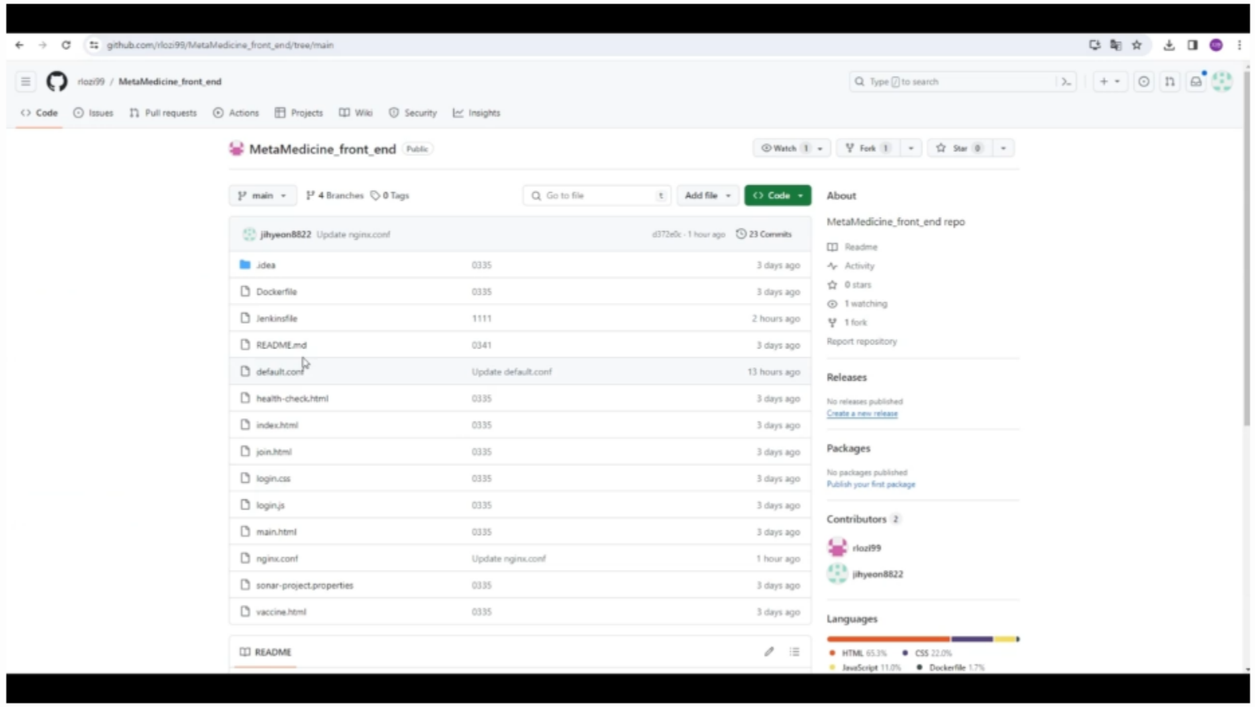



▲ metrics 시각화

# Alert manager – slack



# 모니터링 시연 영상(Nginx+node+Argo+DB)




MetaMedical
메타5형제

KOSA 한국소프트웨어산업협회
Metanet