

Kubernetes 기반 MSA 인프라 구축 프로젝트

NetFort 예매 플랫폼

김현수 김민성 김혜수 신영민 신지혜 조준한

목차

- | | | | |
|----|------------|----|----------|
| 01 | 프로젝트 개요 | 05 | 배포 및 테스트 |
| 02 | 팀원 구성 및 일정 | 06 | 프로젝트 결과 |
| 03 | 시스템 구성 | 07 | 트러블슈팅 |
| 04 | 서비스 구성 | 08 | 향후 계획 |



01 프로젝트 개요

01 프로젝트 개요

프로젝트의 목적 및 목표

- 예매 시스템을 **마이크로서비스 아키텍처(MSA)**로 전환하고, Kubernetes 기반의 컨테이너 환경에 배포하여 각 서비스의 **독립성과 시스템 확장성**을 확보
- 각 **Worker Node**에 서비스들을 역할별로 **분산 배치**하고, 수동 배포 방식을 적용하여 **인프라의 안정적인 분산 구성**을 실현
- 운영 중 장애 발생 시 **신속한 대응력과 안정적인 운영 환경 확보**를 궁극적인 목표로 설정

대상 사용자 및 비즈니스 목적

본 프로젝트는 공연 및 전시 등 다양한 콘텐츠를 제공하는 플랫폼의 **시스템 안정성과 확장성**을 확보하기 위해 기획되었으며, 이를 통해 플랫폼을 **운영·개발하는 인력의 관리 효율성과 대응력을 향상**시키는 것이 주요 목표

특히, **다수 사용자 동시 접속 환경**에서 발생할 수 있는 **시스템 병목 및 단일 장애 지점** 문제를 해결할 수 있도록 **MSA 및 Kubernetes 기반 인프라**를 도입하는 것을 주요 추진 방향으로 설정

01 프로젝트 개요

핵심 기술 스택



인프라 계층	VMware, Ubuntu 24.04
--------	----------------------

오케스트레이션	Kubernetes
---------	------------

배포 도구	Helm
-------	------

이미지 관리	Docker + Nexus
--------	----------------

애플리케이션 계층	Python Flask 기반 MSA
-----------	---------------------

데이터 계층	MySQL, Redis
--------	--------------

네트워크/보안	Nginx + SSL + Cloudflare
---------	--------------------------

모니터링	Grafana + Prometheus
------	----------------------



02 팀원 구성 및 일정

02 팀원 구성 및 일정

김현수

PM

프로젝트 운영
Redis 서버 구성

김민성

Nexus 서비스 담당

이미지 저장소 구성

김혜수

MySQL 서비스 담당

DB 구성

신영민

Flask 웹 서버 담당

웹 서비스 구성

신지혜

모니터링 담당

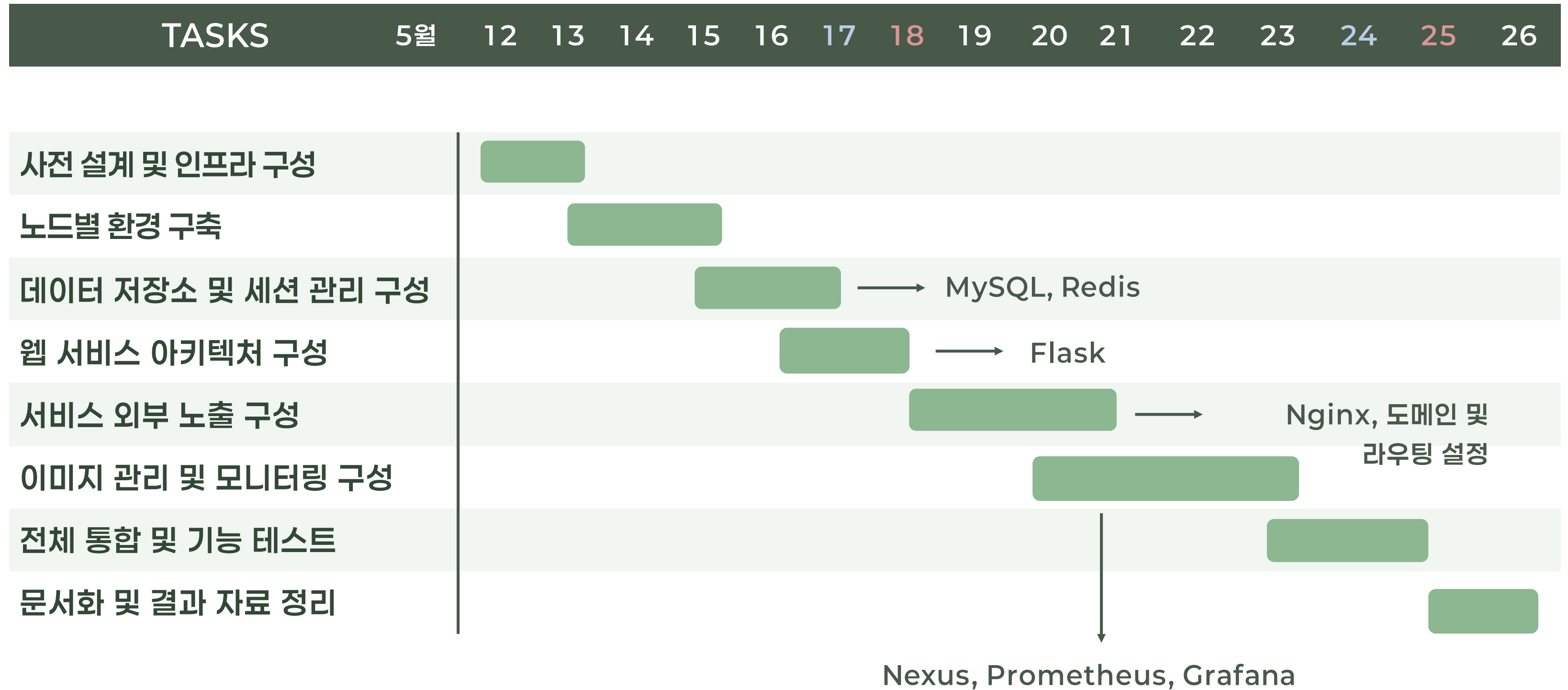
Grafana/
Prometheus 구성

조준한

프록시 및 라우팅 담당

Nginx (Reverse
Proxy) 및 도메인 구성

02 팀원 구성 및 일정





03 시스템 구성

03 시스템 구성

인프라 구성

하드웨어 및 환경 사양

- CPU: 4 Core
- RAM: 16 GB
- 스토리지: 100 GB SSD
- 네트워크: VMware Bridge
- 가상화 플랫폼: VMware Workstation
- 운영체제: Ubuntu Server 24.04.2 LTS
- 컨테이너 오케스트레이션: Kubernetes 1.29.15
- 네트워크 플러그인(CNI): Cilium 1.17.2
- 컨테이너 런타임: containerd 1.7.27
- 로드밸런서 구성: MetalLB 0.13.10

Kubernetes 노드 구성 및 역할

호스트명	IP 주소	역할	배포 서비스 및 비고
master	192.168.31.240	마스터 노드	컨트롤 플레인, etcd, Helm 관리
work1	192.168.31.241	워커 노드	MySQL 배포
work2	192.168.31.242	워커 노드	Nexus 배포
work3	192.168.31.243	워커 노드	Grafana & Prometheus 배포
work4	192.168.31.244	워커 노드	일반 워커
work5	192.168.31.245	워커 노드	일반 워커

03 시스템 구성



Kubernetes 클러스터 구성

1대의 마스터 노드와 5대의 워커 노드, 총 6대로 구성된
K8s 클러스터의 노드 및 네트워크 정상 상태를 확인

```
root@master:~# cilium status

  Cilium:      OK
  Operator:    OK
  Envoy DaemonSet: OK
  Hubble Relay: disabled
  ClusterMesh: disabled

DaemonSet      cilium          Desired: 6, Ready: 6/6, Available: 6/6
DaemonSet      cilium-envoy    Desired: 6, Ready: 6/6, Available: 6/6
Deployment      cilium-operator Desired: 1, Ready: 1/1, Available: 1/1
Containers:    cilium          Running: 6
                cilium-envoy    Running: 6
                cilium-operator  Running: 1
                clustermesh-apiserver
                hubble-relay

Cluster Pods:   22/22 managed by Cilium
Helm chart version: 1.17.2
Image versions  cilium          quay.io/cilium/cilium:v1.17.2@sha256:3c4c9932b5c8def5f41c18e410bcc84025fcd385b1: 6
                  cilium-envoy    quay.io/cilium/cilium-envoy:v1.31.5-1741765102-c44c93d316b846a211@sha256:377c78c13d2731f3720f931721ee309159e782d882251709cb0fac3b42c03f4
                  cilium-operator  quay.io/cilium/operator-generic:v1.17.2@sha256:3a8361e4c68d47d19c68a0d42f0b7b6e3f0523f249: 1
```

```
root@master:~/project/grafana# kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
master	Ready	control-plane	27d	v1.29.15	192.168.31.240	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27
work1	Ready	<none>	27d	v1.29.15	192.168.31.241	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27
work2	Ready	<none>	27d	v1.29.15	192.168.31.242	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27
work3	Ready	<none>	27d	v1.29.15	192.168.31.243	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27
work4	Ready	<none>	27d	v1.29.15	192.168.31.244	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27
work5	Ready	<none>	27d	v1.29.15	192.168.31.245	<none>	Ubuntu 24.04.2 LTS	6.8.0-60-generic	containerd://1.7.27



04 서비스 구성

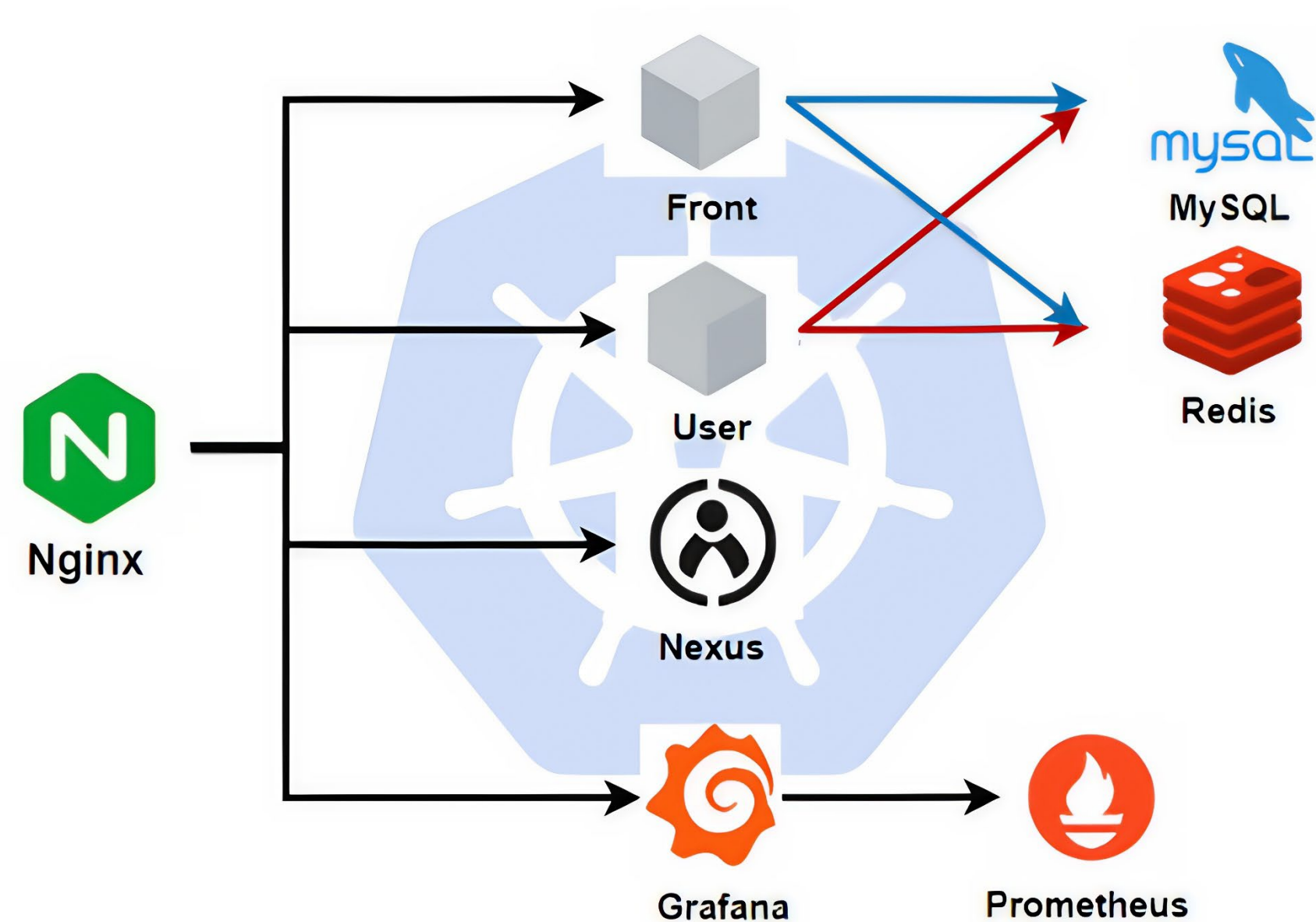
04 서비스 구성

서비스 요약

서비스 아키텍처 개요

본 프로젝트는 k8s 환경에서 Nginx를 진입점으로 하여
각 마이크로 서비스로 라우팅 되는 구조로 설계

- 외부 트래픽 관리와 보안을 위한 Nginx 기반 리버스 프록시 구성
- 독립적으로 운영되는 Frontend 및 User의 마이크로서비스 배포
- MySQL과 Redis를 활용한 데이터베이스 및 세션 저장소 구축
- Nexus, Grafana, Prometheus를 통한 이미지 저장 및 시스템 모니터링 환경 구축



04 서비스 구성

서비스 요약

서비스 아키텍처 요약

- **Nginx** : Nginx Reverse Proxy 및 외부 서비스 라우팅 구성
- **Front** : MySQL 과 연동하여 공연 정보 및 예매
- **User** : Redis 와 연결되어 인증 및 세션 처리
- **Nexus** : Docker 이미지 저장소로 활용
- **Grafana** : Prometheus 에서 수집한 모니터링 데이터를 시각화

각 서비스는 Kubernetes 클러스터 내에서 독립적으로 배포되어 관리되며, 내부 도메인 기반으로 통신

구조 요약

Name	Image	Service Type	Port	Endpoint
Nginx	nginx	Load Balancer	80, 443	Reverse Proxy
Front	nexus/front	ClusterIP	9000	ticket.netfort.kr
User	nexus/user	ClusterIP	9001	user.netfort.kr
MySQL	mysql	ClusterIP	3306	-
Redis	redis	ClusterIP	6379	-
Nexus	Sonatype/nexus:	ClusterIP	8081	nexus.netfort.kr

04 서비스 구성



Nginx 서비스 구성

리버스 프록시 역할을 수행하며, 외부 요청을 내부 Flask 서비스(Front, User 등)로 라우팅
LoadBalancer로 외부에 노출

- 다중 Pod로 구성된고가용성 환경
- 외부 접근을 위한 LoadBalancer 서비스
- 도메인 기반 리버스 프록시 구성
- Cloudflare는 해당 IP와 도메인을 연동하여 SSL 인증과 보안 기능 및 CDN을 제공

```
### ConfigMap
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
  namespace: msa-service
data:
  flask-front.conf: |
    server {
      listen 80;
      server_name ticket.netfort.kr;

      location / {
        proxy_pass http://flask-front.msa-
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      }
    }

  flask-user.conf: |
    server {
      listen 80;
      server_name user.netfort.kr;

      location / {
        proxy_pass http://flask-user.msa-
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      }
    }

  grafana.conf: |
    server {
      listen 80;
      server_name grafana.netfort.kr;
    }

```

```
### Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: msa-service
  labels:
    service: nginx
    project: netfort
spec:
  replicas: 5
  selector:
    matchLabels:
      service: nginx
      project: netfort

```

```
### Service
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: msa-service
spec:
  type: LoadBalancer
  selector:
    service: nginx
    project: netfort
  ports:
    - port: 80
      targetPort: 80

```

netfort.kr 또는 하위 도메인 중 하나에 새 SSL/TLS 인증서가 발급되었습니다

인증서 세부 정보

1. 로그 날짜: 2025-05-22 20:00:38 UTC
2. 발급자: CN=E6,O=Let's Encrypt,C=US
3. 유효 기간: 2025-05-22 19:01:35 UTC - 2025-08-20 19:01:34 UTC
4. DNS 이름: *.netfort.kr, netfort.kr

유형 ①	이름 ①	콘텐츠 ①	프록시 상태 ①
CNAME	grafana	netfort.kr	프록싱됨
CNAME	nexus	netfort.kr	프록싱됨
CNAME	ticket	netfort.kr	프록싱됨
CNAME	user	netfort.kr	프록싱됨

04 서비스 구성



Flask(웹 서비스) 서비스 구성

각 서비스는 Flask 기반의 독립 컨테이너로 구성되어 있으며, 프론트 유저 서비스가 분리되어 있어, 유지보수 및 확장에 유리

- Flask 기반의 경량 웹 서비스
- 기능 단위의 마이크로서비스 분리
- 서비스 간 API 통신 구조
- Docker 이미지는 Nexus 프라이빗 저장소에서 관리 및 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-front
  namespace: msa-service
  labels:
    service: front
    project: netfort
spec:
  replicas: 3
  selector:
    matchLabels:
      service: front
      project: netfort
  template:
    metadata:
      labels:
        service: front
        project: netfort
    spec:
      containers:
        - name: flask-front
          image: master:30500/project-hosted/flask-front:latest
          ports:
            - containerPort: 9000
          imagePullSecrets:
            - name: nexus-secret
---
apiVersion: v1
kind: Service
metadata:
  name: flask-front
  namespace: msa-service
spec:
  type: ClusterIP
  selector:
    service: front
    project: netfort
  ports:
    - protocol: TCP
      port: 9000
      targetPort: 9000

apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-user
  namespace: msa-service
  labels:
    service: user
    project: netfort
spec:
  replicas: 3
  selector:
    matchLabels:
      service: user
      project: netfort
  template:
    metadata:
      labels:
        service: user
        project: netfort
    spec:
      containers:
        - name: flask-user
          image: master:30500/project-hosted/flask-user:latest
          ports:
            - containerPort: 9001
          imagePullSecrets:
            - name: nexus-secret
---
apiVersion: v1
kind: Service
metadata:
  name: flask-user
  namespace: msa-service
spec:
  type: ClusterIP
  selector:
    service: user
    project: netfort
  ports:
    - protocol: TCP
      port: 9001
      targetPort: 9001
```


04 서비스 구성



Flask(웹 서비스) 서비스 구성

MSA 환경에서 msa-service 네임스페이스를
통해 Redis, MySQL, Flask 간 연결을 구성

로그인 및 회원가입은 MySQL을
통해 검증, 서비스간 세션은 Redis로 유지

다양한 사용자 흐름(공연 조회 → 예매 →
로그인/로그아웃 → 회원가입)을 지원

```
11 REDIS_HOST = os.environ.get('REDIS_HOST', 'redis.msa-se try:
12 REDIS_DB = int(os.environ.get('REDIS_DB', 0))
13 try:
14     REDIS_PORT = int(os.environ.get('REDIS_PORT', 6379))
15 except ValueError:
16     REDIS_PORT = 6379
17
18 FLASK_SECRET_KEY = os.environ.get("FLASK_SECRET_KEY", '
19 FLASK_PORT = int(os.environ.get("FLASK_PORT", 9000))
20
21 # 비밀키 설정
22 app.secret_key = FLASK_SECRET_KEY
23
24 # Redis 설정
25 app.config['SESSION_TYPE'] = 'redis'
26 app.config['SESSION_PERMANENT'] = False
27 app.config['SESSION_USE_SIGNER'] = True
28 app.config['SESSION_REDIS'] = redis.StrictRedis(
29     host=REDIS_HOST,
30     port=REDIS_PORT,
31     db=REDIS_DB
32 )
33
34 # 세션 설정
35 app.config['SESSION_COOKIE_DOMAIN'] = '.netfort.kr'
36 Session(app)
37
38 # 특정 공연 정보 가져오기
39 def get_show_data(show_id):
40     shows = load_shows_data()
41     for show in shows:
42         if show["id"] == int(show_id):
43             return show
44     return None # 해당 ID의 공연이 없는 경우
45
46 # MySQL 서버에 접속
47 conn = mysql.connector.connect(
48     host=db_host,
49     user=db_user,
50     password=db_password
51 )
52 cursor = conn.cursor()
53
54 # 데이터베이스가 없으면 생성
55 cursor.execute(f"CREATE DATABASE IF NOT EXISTS {db_name}")
56 print(f"데이터베이스 '{db_name}' 확인 또는 생성 완료")
57
58 # 해당 데이터베이스 사용
59 cursor.execute(f"USE {db_name}")
60
61 # users 테이블 생성
62 cursor.execute('''
63 CREATE TABLE IF NOT EXISTS users (
64     id INT AUTO_INCREMENT PRIMARY KEY,
65     username VARCHAR(255) NOT NULL UNIQUE,
66     password VARCHAR(255) NOT NULL
67 )
68 ''')
69 conn.commit()
70 print("users 테이블 생성 완료")
71
72 except mysql.connector.Error as err:
73     if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
74         print("사용자 이름 또는 비밀번호 오류")
75     elif err.errno == errorcode.ER_BAD_DB_ERROR:
76         print("데이터베이스가 존재하지 않음")
77     else:
78         print(err)
```

04 서비스 구성



MySQL(데이터베이스) 서비스 구성

웹 서비스용 MySQL 서버를 구축하고,
Headless 서비스를 통해 클러스터 내에서
DNS 기반으로 접근할 수 있도록 구성

- StatefulSet 기반의 안정적인 MySQL 인스턴스 배포
- Headless Service를 통한 고정 DNS 접근 지원
- hostPath 볼륨을 활용한 데이터 영속성 보장
- 노드 고정 배치(Affinity)로 특정 노드에 실행 제한

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
  namespace: msa-service
  labels:
    service: mysql
    project: netfort
spec:
  selector:
    matchLabels:
      service: mysql
      project: netfort
  serviceName: mysql
  replicas: 1
  template:
    metadata:
      labels:
        service: mysql
        project: netfort
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: kubernetes.io/hostname
                    operator: In
                    values:
                      - work1
      containers:
        - name: mysql
          image: mysql:latest
          ports:
            - containerPort: 3306
              name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: rootpass
          volumeMounts:
            - name: mysql-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-storage
          hostPath:
            path: /k8s-data/mysql
            type: Directory

---
apiVersion: v1
kind: Service
metadata:
  name: mysql
  namespace: msa-service
  labels:
    service: mysql
    project: netfort
spec:
  clusterIP: None
  selector:
    service: mysql
    project: netfort
  ports:
    - name: mysql
      port: 3306
      targetPort: 3306
      protocol: TCP
```

04 서비스 구성



Redis(세션 저장소) 서비스 구성

웹 서비스의 세션용 Redis 서버를 구축하고
세션 데이터를 단일 Redis 인스턴스에
저장하여, 데이터 일관성을 보장

- 단일 복제본(Replica)으로 간단한 Redis 배포
- 경량화된 캐시/데이터 저장소 운영에 적합한 구성
- 노드 상태 변화에 대응하는 내결함성 설정으로 안정적인 서비스 유지

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  namespace: msa-service
  labels:
    service: redis
    project: netfort
spec:
  replicas: 1
  selector:
    matchLabels:
      service: redis
      project: netfort
  template:
    metadata:
      labels:
        service: redis
        project: netfort
    spec:
      tolerations:
        - key: "node.kubernetes.io/not-ready"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 0
        - key: "node.kubernetes.io/unreachable"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 0
      containers:
        - name: redis
          image: redis:latest
          ports:
            - containerPort: 6379
              protocol: TCP
```

```
apiVersion: v1
kind: Service
metadata:
  name: redis
  namespace: msa-service
spec:
  type: ClusterIP
  selector:
    service: redis
    project: netfort
  ports:
    - port: 6379
      targetPort: 6379
      protocol: TCP
```

04 서비스 구성



Nexus(이미지 저장소) 서비스 구성

웹 서비스의 버전 관리와 접근 제어를 위해 Nexus 서버를 구축하고 다양한 웹 서비스 개발 아티팩트를 저장 및 관리하여 효율적으로 공유하고 재사용 할 수 있도록 구성

- Docker 이미지뿐만 아니라 다양한 패키지 포맷을 한 곳에서 통합 관리할 수 있어, 여러 개발 환경을 동시에 지원
- 자체 Nexus 저장소를 운영하여 민감한 소스 코드와 패키지 데이터를 외부 의존 없이 안전하게 관리

```
apiVersion: v1
kind: Service
metadata:
  name: nexus
  namespace: msa-service
  labels:
    service: nexus
    project: netfort
spec:
  type: NodePort
  selector:
    service: nexus
    project: netfort
  ports:
    - name: web
      port: 8081
      targetPort: 8081
      protocol: TCP
    - name: docker
      port: 5000
      targetPort: 5000
      nodePort: 30500

---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nexus
  namespace: msa-service
  labels:
    service: nexus
    project: netfort
spec:
  serviceName: nexus
  replicas: 1
  selector:
    matchLabels:
      service: nexus
```

```
template:
  metadata:
    labels:
      service: nexus
      project: netfort
  spec:
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
            - matchExpressions:
                - key: kubernetes.io/hostname
                  operator: In
                  values:
                    - work2
    containers:
      - name: nexus
        image: sonatype/nexus3:latest
        ports:
          - containerPort: 8081
            protocol: TCP
        volumeMounts:
          - name: nexus-volume
            mountPath: /nexus-data
    volumes:
      - name: nexus-volume
        hostPath:
          path: /k8s-data/nexus
          type: DirectoryOrCreate
```


04 서비스 구성



Grafana/Prometheus 서비스 구성

K8s 클러스터의 컨테이너와 시스템 성능
데이터를 실시간으로 수집하고 저장하며 대시보드를
구성하여 시각화 및 분석

- Prometheus를 통한 실시간 메트릭 수집 및 저장
- 내부 및 외부 네트워크 모두에서 안정적으로 접근
가능하도록 구성하여 관리 편의성 향상
- 데이터 영속성 확보를 위해 사전 생성된 PV와 PVC를
활용한 저장 구조 적용

kube-prometheus-stack

Installs core components of the [kube-prometheus stack](#), a collection of Kubernetes manifests combined with documentation and scripts to provide easy to operate end-to-end Kubernetes monitoring.

See the [kube-prometheus](#) readme for details about components, dashboards, and alerts.

Note: This chart was formerly named `prometheus-operator` chart, now renamed to more accurately reflect the project stack, within which Prometheus Operator is only one component. This chart does not notably exclude the Prometheus Adapter and Prometheus black-box exporter.

Prerequisites

- Kubernetes 1.19+
- Helm 3+

Get Helm Repository Info

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

See [helm repo](#) for command documentation.

Install Helm Chart

```
helm install [RELEASE_NAME] prometheus-community/kube-prometheus-stack
```

See [configuration](#) below.

See [helm install](#) for command documentation.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-grafana
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: grafana-storage
  local:
    path: /k8s-data/grafana
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - work3
  persistentVolumeReclaimPolicy: Retain
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-grafana
  namespace: prometheus-stack
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: grafana-storage
  volumeName: pv-grafana
```

<https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack>



05 배포 및 테스트

05 배포 및 테스트



Nginx / Web 연결 테스트

- 배포된 공연 예매 사이트를 Nginx 리버스 프록시를 통해 외부 접속 테스트 진행
- 접속 주소: `https://ticket.netfort.kr`
- 테스트 항목:
 - 메인 페이지 접속 (공연 목록 확인)
 - 공연 상세 정보 페이지 접속 (`/show/<공연ID>`)
 - Nginx 로그를 통해 요청 경로 및 정상 라우팅 확인

Nginx 접속 로그 확인

```
10.0.0.35 - - [18/Jun/2025:01:36:46 +0000] "GET /static/images/25003775-04.jpg HTTP/1.1" 200 2711412 "https://ticket.netfort.kr/show/2" AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36" "112.221.246.164, 172.71.215.173"
10.0.0.35 - - [18/Jun/2025:01:37:15 +0000] "POST /login HTTP/1.1" 302 203 "https://user.netfort.kr/login" "https://ticket.netfort.kr/show/2" AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36" "112.221.246.164, 172.71.215.173"
10.0.0.35 - - [18/Jun/2025:01:37:15 +0000] "GET /profile HTTP/1.1" 200 1555 "https://user.netfort.kr/login" "https://ticket.netfort.kr/show/2" AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36" "112.221.246.164, 172.71.215.173"
```

공연 예매 사이트 접속 성공

The screenshot shows two browser windows. The top window displays the NetFort Ticket homepage with a featured concert 'THE WORLD TOUR SEUL KINTEX HALL 10 SEP 13, 14 2025 NEW SHOW ADDED' by Tyler, The Creator. The bottom window shows the concert details page for 'The Magic, 조수미 & 위너스' (SUMI JO & WINNERS) on June 22, 2025, at Lotte Concert Hall. The details include the venue, performance time (120 minutes), audience rating (8+), and a price list. A calendar at the bottom shows the performance date highlighted.

장소	롯데콘서트홀
공연기간	2025.06.22
공연시간	120분(인터미션 15분 포함)
관람연령	8세 이상 관람가
가격	전체가격보기 ▲
R석	165,000원
S석	132,000원
A석	110,000원
B석	88,000원
시아방해R	115,500원
시아방해S	92,400원
시아방해A	77,000원
시아방해B	61,600원

05 배포 및 테스트



MySQL 연결 테스트

- 회원가입 및 로그인 기능 테스트를 통해 Web 서비스와 MySQL 간의 연결이 정상적으로 동작함을 확인
- 테스트 항목:
 - Frontend 서비스와 MySQL 간 연결 확인
 - DB 인증 및 접속 후 데이터베이스 확인
 - 회원가입 및 로그인 성공. DB에 사용자 정보 저장 확인

DB에 저장된 유저와 패스워드 확인

```
mysql> select * from users;
```

id	username	password
1	admin	\$2b\$12\$n5y0Y/0Ghf2RFm4yTkPPQuhr0Z0GkCuyFQpkoJj22lTJnfDz/AkA0
2	test	\$2b\$12\$6dvKdrdI2W0QTwfajAQLieQkhMLM.QvmuJrEzGyZrkyf82ALq1EQW
5	netfort	\$2b\$12\$ZrbhrH33K88Ut/E6mDQR0eny8Z1/2vsmeUfwgmLud5Zfgn0EcHvGK

공연 예매 사이트 회원가입 및 로그인 성공

NetFort Ticket

NetFort Ticket

LOGOUT

Sign Up

Username:
netfort

Password:
.....

Sign Up

Already have an account? Log in

netfort 님 반갑습니다.

05 배포 및 테스트



Redis 연결 테스트

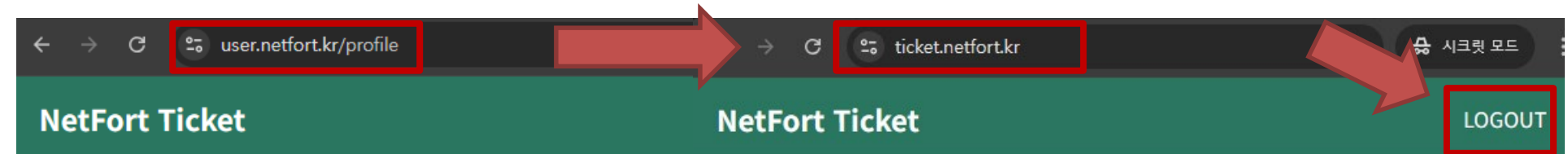
- 공연 예매 사이트에서 로그인 후 세션 유지 여부 확인
- 테스트 항목:
 - Frontend 서비스와 Redis 간 정상 연결 확인
 - 로그인 후 세션이 Redis에 정상 저장되는지 점검
 - 마이크로서비스 간 세션 정보가 정상적으로 공유되는지 확인

Redis에 저장된 세션 확인

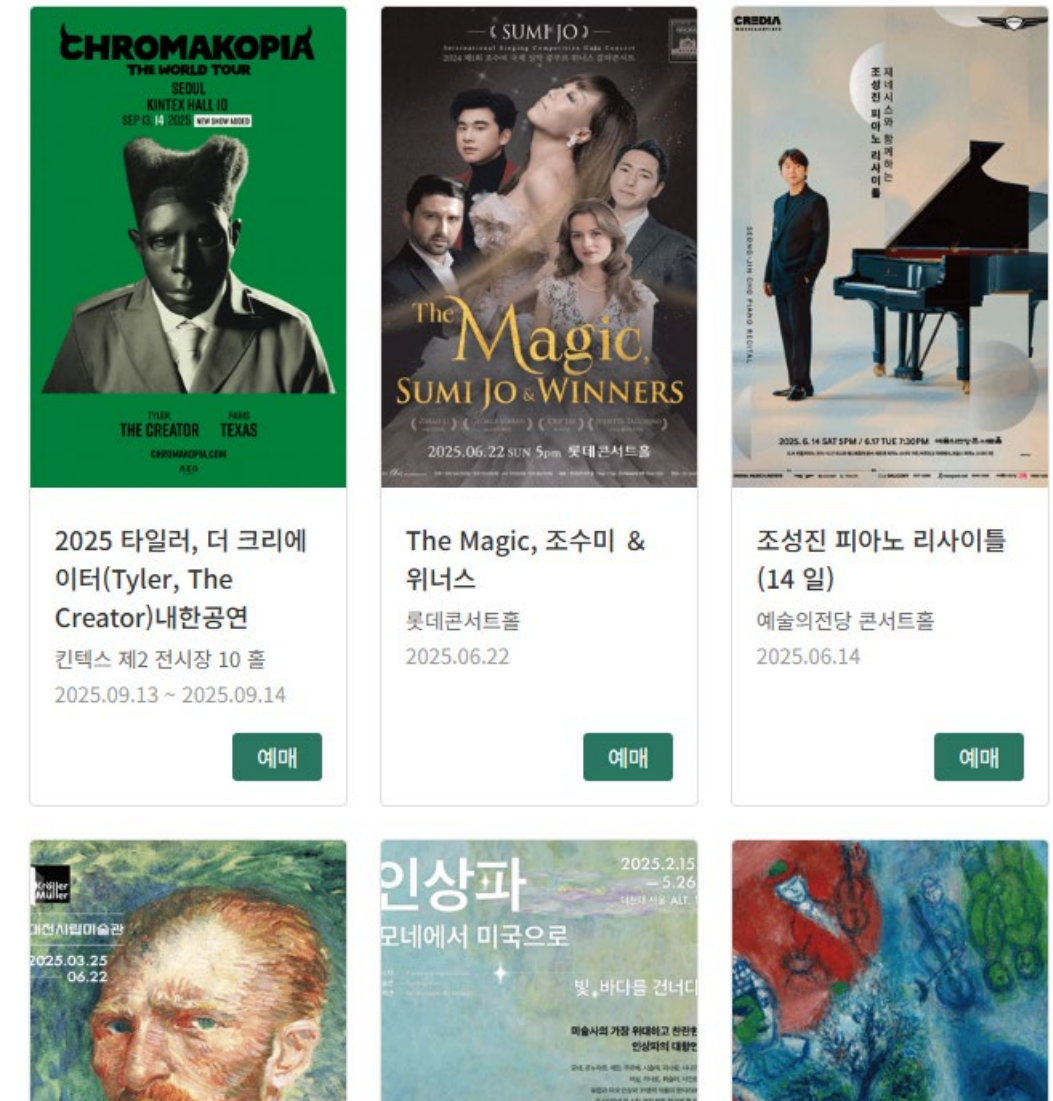
```
root@master:~/project# kubectl exec -it -n msa-service pod/redis-66b67c5954-w2wb6 -- redis-cli
127.0.0.1:6379> KEYS *
```

1) "session:1vp00_AUP5ed07WUUEqFG5fUzMhvHs1HR1kL_f50Xv0"
2) "session:ETPZYbgZqi5SY1nAiysv78xZR26T5CW7UopekYIOnCw"
127.0.0.1:6379>

User 로그인 상태가 Ticket 서비스에서도 유지되는지 검증



netfort 님 반갑습니다



05 배포 및 테스트

Nexus 연결 테스트

- Nexus와의 연결 상태 및 이미지 배포 가능 여부 확인
- 테스트 항목:
 - 로컬에서 Docker 이미지 빌드 후 Nexus로 Push
 - K8s 클러스터에서 Nexus로부터 이미지 Pull
 - 이미지 다운로드 및 Pod 정상 기동 여부 확인
 - Nexus 웹 UI에서 업로드된 이미지 확인

웹 서비스의 이미지를 Nexus 레지스트리 Push

```
root@master:~/project/flask/user# docker push 192.168.31.240:30500/project-hosted/flask-user:latest
The push refers to repository [192.168.31.240:30500/project-hosted/flask-user]
e79932574b70: Pushed
17f082b318b6: Pushed
9009893e54c5: Pushed
e6c5004ee77f: Pushed
997b8e79e84f: Pushed
3054512b6f71: Pushed
ae2d55769c5e: Pushed
e2ef8a51359d: Pushed
latest: digest: sha256:7c42200f85ad452f0a7a3ac3aa5c486a07cc165ab51e01b0b14cf02ad2e07ab5 size: 1007
root@master:~/project/flask/user# docker push 192.168.31.240:30500/project-hosted/flask-front:latest
The push refers to repository [192.168.31.240:30500/project-hosted/flask-front]
172e4d507810: Pushed
112636bbe444: Pushed
b843c96bd7ba: Pushed
```

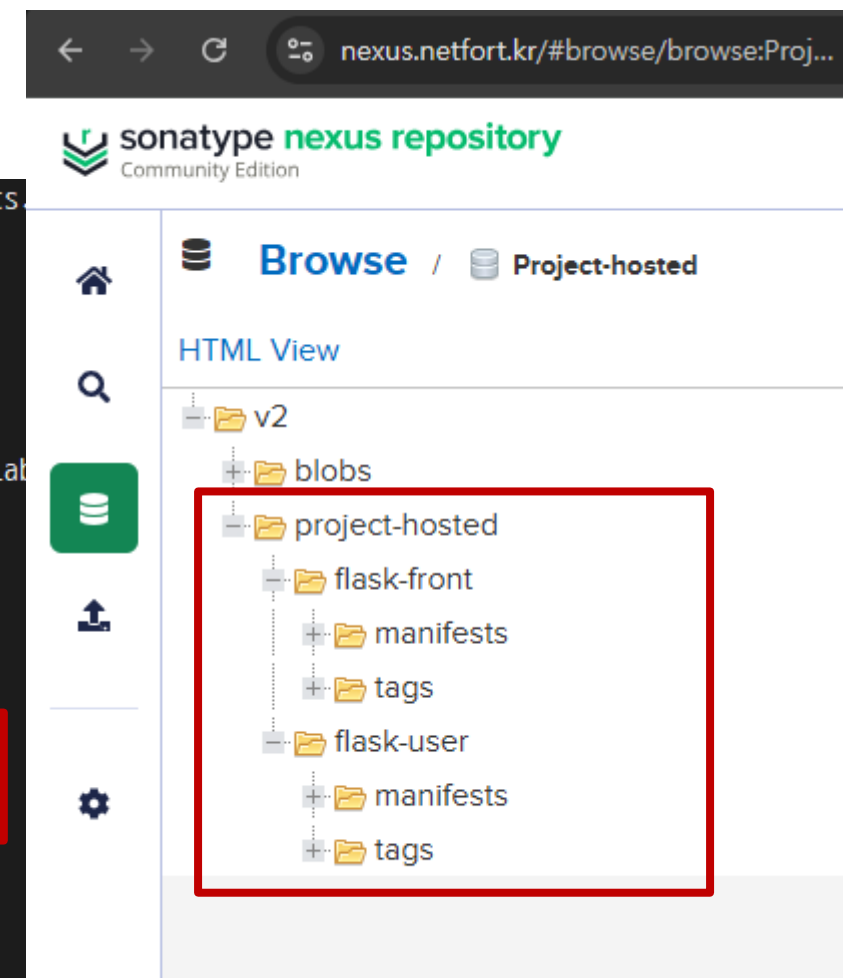
```
server = "http://master:30500"

[host."http://master:30500"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
```

containerd에서 Nexus 서버 설정

```
root@master:~/project# kubectl describe -n msa-service deployments
Name: flask-front
Namespace: msa-service
CreationTimestamp: Thu, 22 May 2025 16:26:47 +0000
Labels: project=netfort
        service=front
Annotations: deployment.kubernetes.io/revision: 1
Selector: project=netfort,service=front
Replicas: 3 desired | 3 updated | 3 total | 3 available
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels: project=netfort
         service=front
  Containers:
    flask-front:
      Image: master:30500/project-hosted/flask-front:latest
      Port: 9000/TCP
      Host Port: 0/TCP
      Environment: <none>
      Mounts: <none>
      Volumes: <none>
  Conditions:
```

K8s에서 Nexus 레지스트리 이미지 Pull 완료



업로드된 이미지 확인 완료

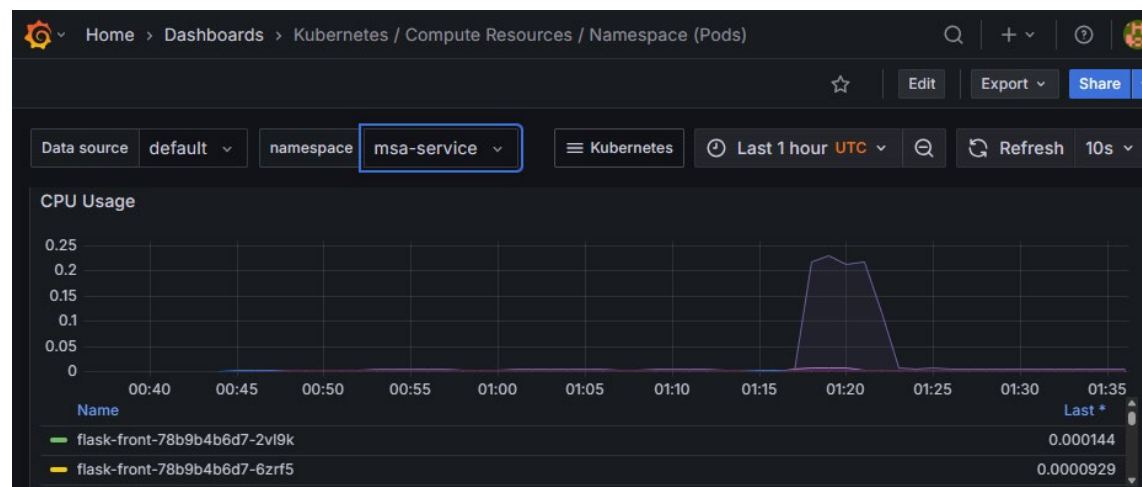
05 배포 및 테스트



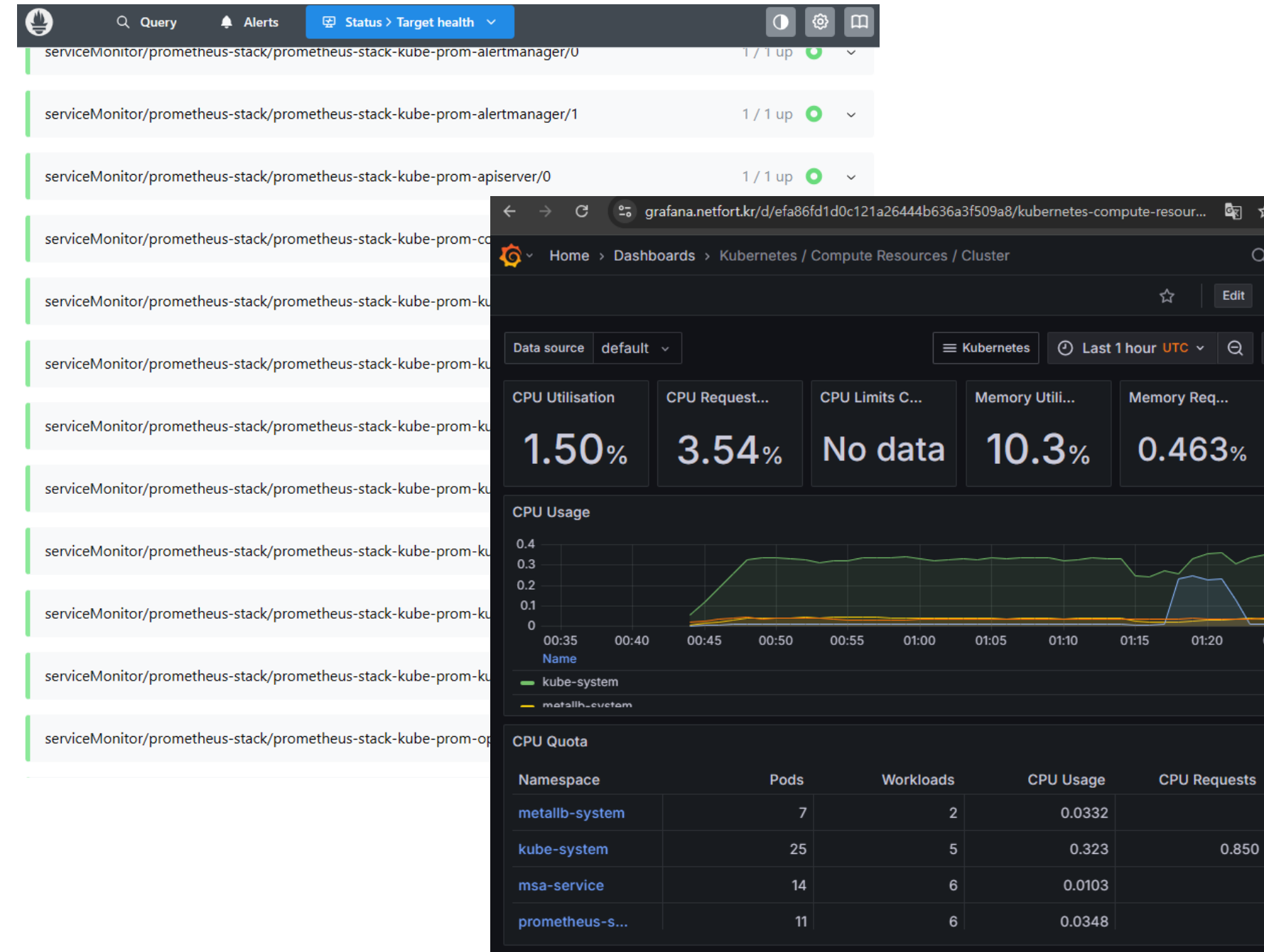
모니터링 테스트

- Grafana 및 Prometheus 기반 모니터링 기능 검증
- 테스트 항목:
 - Prometheus 웹 UI에서 주요 메트릭 수집 상태 확인
 - Grafana 대시보드에서 CPU, Memory, 네트워크 사용량 시각화 확인 및 네임스페이스별 리소스 상태 모니터링

네임스페이스별 리소스 상태 모니터링



Prometheus 메트릭 수집 상태 확인








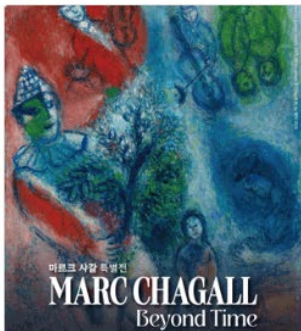


K8s 클러스터 리소스 확인



06 프로젝트 결과

06 프로젝트 결과

메인 페이지

NetFort Ticket		LOGIN	
 <p>2025 타이틀, 더 크리에이터(Tyler, The Creator)내한공연</p> <p>컨택스 제2 전시장 10 홀</p> <p>2025.09.13 ~ 2025.09.14</p> <p>예매</p>	 <p>The Magic, 조수미 & 위너스</p> <p>롯데콘서트홀</p> <p>2025.06.22</p> <p>예매</p>	 <p>조성진 피아노 리사이틀 (14 일)</p> <p>예술의전당 콘서트홀</p> <p>2025.06.14</p> <p>예매</p>	 <p>불멸의 화가 반 고흐 in 대전</p> <p>대전시립미술관</p> <p>2025.03.25 ~ 2025.06.22</p> <p>예매</p>
 <p>인상파, 모네에서 미국으로: 빛, 바다를 건너다</p> <p>더현대 서울 ALT.1</p> <p>2025.02.15 ~ 2025.05.26</p> <p>예매</p>	 <p>[1 차 얼리버드] 마르크 샤갈 특별전: BEYOND TIME</p> <p>예술의 전당 한가람미술관 1 층</p> <p>2025.05.23 ~ 2025.07.13</p> <p>예매</p>	 <p>[디뮤지엄] 취향가죽: Art in Life, Life in Art</p> <p>디뮤지엄</p> <p>2024.11.15 ~ 2025.05.18</p> <p>예매</p>	 <p>라우트 대한공연 - 대구</p> <p>엑스코 5 층 컨벤션홀</p> <p>2025.05.25</p> <p>예매</p>

공연 상세 페이지

NetFort Ticket

LOGOUT

홈 > 공연 예매 > 2025 타악

NetFort Ticket

LOGOUT

홈 > 공연 예매 > The Magic

The Magic

조성진 피아노 리사이틀 (14 일)

장소

예술의전당 콘서트홀

공연기간

2025.06.14

공연시간

100분(인터미션 포함)

관람연령

8세 이상 관람가

가격

전체가격보기 ▼

관람일

월

화

수

목

금

토

일

7	1	2	3	4	5	6
14	8	9	10	11	12	13
21	15	16	17	18	19	20
28	22	23	24	25	26	27
	29	30				

회차

관람일

월

화

수

목

금

토

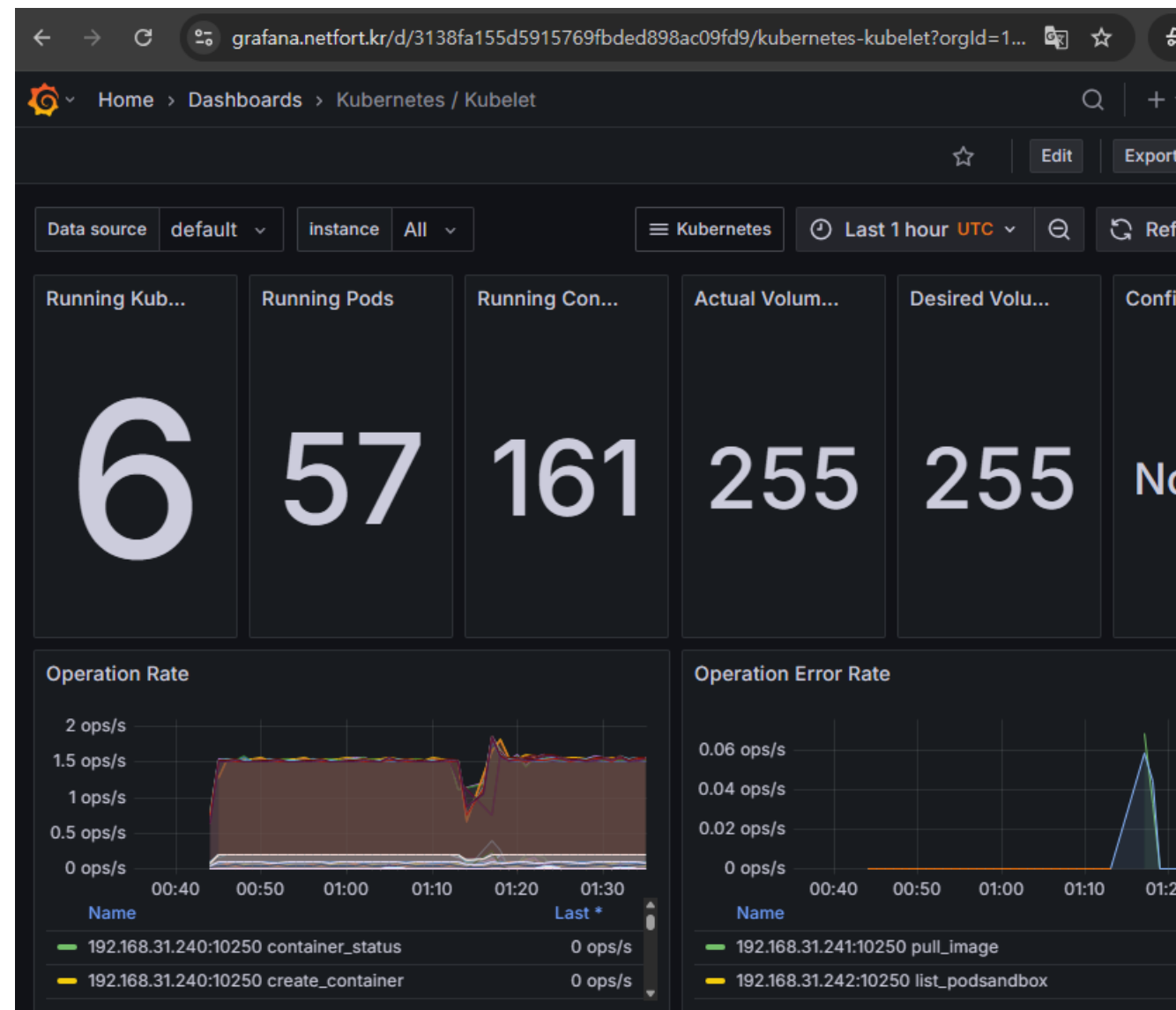
일

7	1	2	3	4	5	6
14	8	9	10	11	12	13
21	15	16	17	18	19	20
28	22	23	24	25	26	27
	29	30				

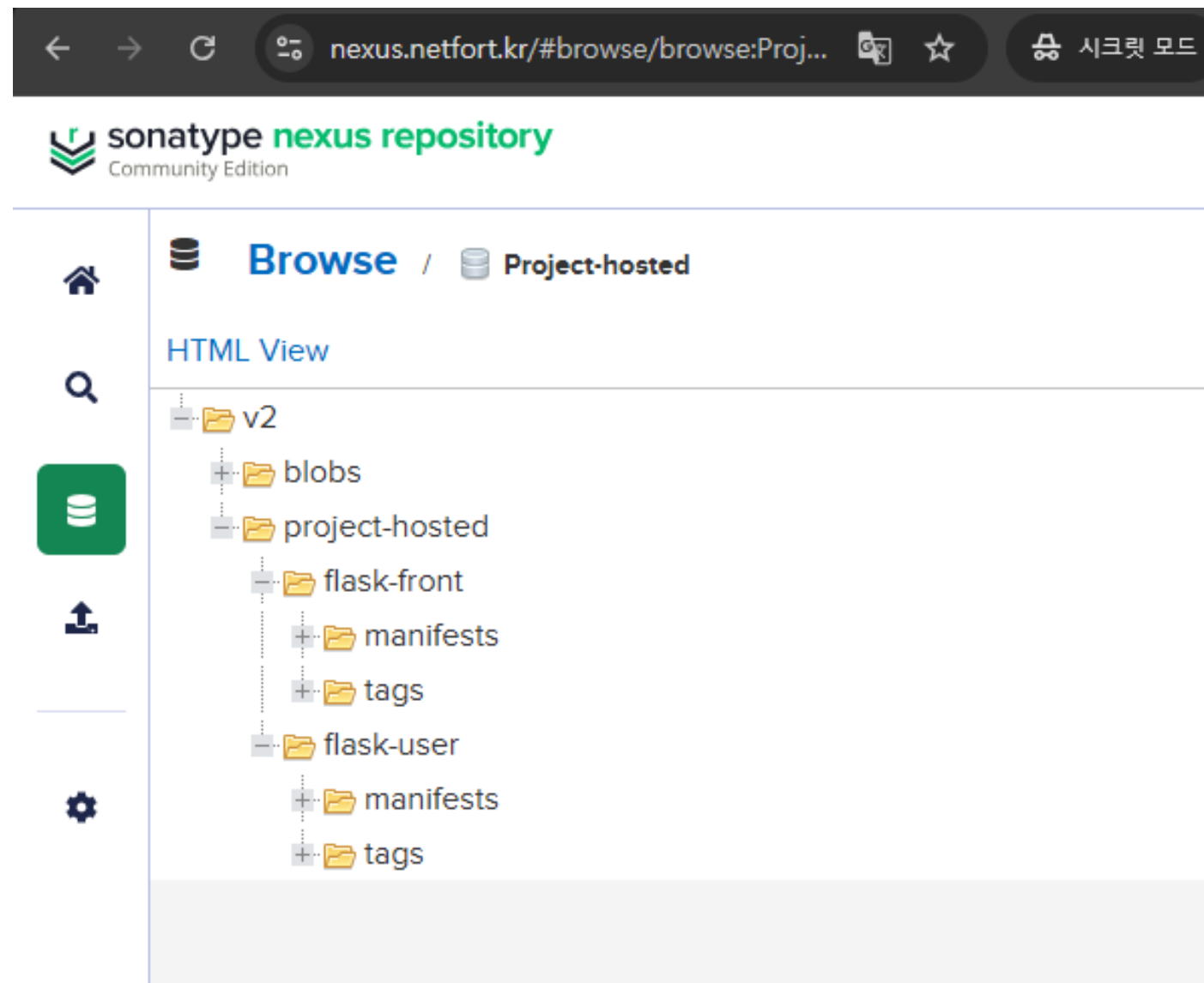
회차

06 프로젝트 결과

모니터링 페이지(Grafana)



Nexus 이미지 저장소



06 프로젝트 결과

초기 목표	구현 결과
MSA 기반의 안정적인 웹 서비스 구축	Front, User 등 핵심 기능 독립 마이크로서비스 배포 완료
모듈화된 마이크로서비스 설계 및 구현	각 도메인별 컨테이너 분리 및 내부 DNS 기반 서비스 통신 완성
Kubernetes 컨테이너 오케스트레이션 구축	Kubernetes 클러스터 운영, Cilium CNI 및 MetalLB 구성 완료
데이터 영속성 및 안정성 확보	MySQL 및 Grafana/Prometheus 데이터 영속성 확보 완료
효율적인 모니터링 체계 구축	Prometheus를 통한 메트릭 수집과 Grafana 대시보드 구축으로 실시간 시스템 상태 모니터링 및 시각화 완성
이미지 저장소(Nexus) 통한 아티팩트 관리 최적화	Nexus 프라이빗 레지스트리 운영, 버전 관리 및 롤백 체계 완성

06 프로젝트 결과

도입 예상 효과

시스템 안정성 향상

- 서비스 간 독립성 확보로 장애 격리 및 개별 유지보수 용이
- 컨테이너 기반 운영으로 환경 일관성 및 배포 신뢰성 증가
- 데이터 영속성 확보로 예기치 않은 종료나 장애 발생 시 복구 가능성 향상

운영 효율성 개선

- Kubernetes 기반 자원 관리로 서비스 확장 및 배포 자동화 기반 마련
- 서비스별 YAML 템플릿화로 반복 배포 시 구성 오류 최소화 및 유지보수 용이
- Nexus 이미지 저장소 도입으로 빌드/배포 흐름 간소화 및 롤백 가능성 확보

실시간 감시 및 문제 대응력 강화

- Prometheus + Grafana 구성으로 리소스 사용량 및 시스템 상태 실시간 모니터링 가능
- 장애 징후 조기 감지 및 대응 체계 구성 → 운영 리스크 최소화

DevOps 기반 개선 기반 마련

- Git 기반 소스 관리 + 컨테이너화 → CI/CD 자동화로 발전 가능한 구조 완성
- 향후 고가용성(HA), 오토스케일링 등 고도화 확장 용이



07 트러블슈팅

07 트러블슈팅



일부 Prometheus 메트릭 수집 실패 이슈 발생

serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-apiserver/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-coredns/0	2 / 2 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-controller-manager/0	1 / 6 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-etcd/0	0 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-proxy/0	0 / 6 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-scheduler/0	0 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kubelet/0	6 / 6 up	●	^

Endpoint	Labels	Last scrape	State
https://192.168.31.243:10250/metrics	endpoint="https-metrics" instance="192.168.31.243:10250"	4m 243ms ago 18ms	UP



해결 방안

kube-proxy

```
apiVersion: v1
data:
  config.conf: |-
    apiVersion: kubeproxy.config.k8s.io/v1alpha1
    bindAddress: 0.0.0.0
    bindAddressOnStart: false
    clientConnection:
      acceptContentTypes: ""
      burst: 0
```

kubectl edit -n kube-system cm kube-proxy

kube-etcd

```
--experimental-watch-progress-notify-interval=5s
--initial-advertise-peer-urls=https://192.168.31.240:2380
--initial-cluster=master=https://192.168.31.240:2380
--key-file=/etc/kubernetes/pki/etcd/server.key
--listen-client-urls=https://127.0.0.1:2379,https://192.168.31.240:2379
--listen-metrics-urls=http://127.0.0.1:2381,http://192.168.31.240:2381
--listen-peer-urls=https://192.168.31.240:2380
--name=master
--peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
--peer-client-cert-auth=true
```

/etc/kubernetes/manifests/etcd.yaml

kube-controller-manager kube-scheduler

```
--authentication-kubeconfig=/etc/kubernetes/
--authorization-kubeconfig=/etc/kubernetes/
# --bind-address=127.0.0.1
--bind-address=0.0.0.0
--kubeconfig=/etc/kubernetes/scheduler.conf
--leader-elect=true
image: registry.k8s.io/kube-scheduler:v1.29.0
imagePullPolicy: IfNotPresent
```

/etc/kubernetes/manifests/kube-controller-manager.yaml
/etc/kubernetes/manifests/kube-scheduler.yaml

Helm 차트 설정

```
kubeEtcd:
  service:
    enabled: true
    port: 2381
    targetPort: 2381
```

values.yaml

✓ bindAddress 0.0.0.0 및 서비스 포트를 추가

참고 링크

<https://pythaac.tistory.com/457>



07 트러블슈팅



K8s에서 Nexus 이미지 불러오기 실패 이슈 발생

```
root@master:~/project/nginx-yaml# crictl pull master:30500/project-hosted/flask-user:
E0619 09:01:25.889884 41794 remote_image.go:180] "PullImage from image service fail
r="rpc error: code = Unknown desc = failed to pull and unpack image \"master:30500/pr
hosted/flask-user:latest\": failed to resolve reference \"master:30500/project-hosted
-user:latest\": failed to do request: Head \"https://master:30500/v2/project-hosted/f
ser/manifests/latest\": http: server gave HTTP response to HTTPS client" image="maste
0/project-hosted/flask-user:latest"
FATA[0000] pulling image: failed to pull and unpack image "master:30500/project-hoste
k-user:latest": failed to resolve reference "master:30500/project-hosted/flask-user:l
: failed to do request: Head "https://master:30500/v2/project-hosted/flask-user/manif
atest": http: server gave HTTP response to HTTPS client
root@master:~/project/nginx-yaml#
```



해결 방안

containerd

```
160
161 [plugins "io.containerd.grpc.v1.cri" registry]
162   config_path = "/etc/containerd/certs.d"
163
/etc/containerd/config.toml
```

```
server = "http://master:30500"

[host."http://master:30500"]
  capabilities = ["pull", "resolve", "push"]
  skip_verify = true
```

/etc/containerd/certs.d/master:30500/hosts.toml

✓ containerd에 Nexus 서버 등록 및 설정

07 트러블슈팅

모든 Prometheus 메트릭 수집 성공

serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-alertmanager/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-alertmanager/1	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-apiserver/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-coredns/0	2 / 2 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-controller-manager/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-etcd/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-proxy/0	6 / 6 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kube-scheduler/0	1 / 1 up	●	▼
serviceMonitor/prometheus-stack/prometheus-stack-kube-prom-kubelet/0	6 / 6 up	●	▼

Nexus 이미지 가져오기 완료

Events:				
Type	Reason	Age	From	Message
Normal	Scheduled	21s	default-scheduler	Successfully assigned msa-service/flask-user-ff7c748dd-chic9 to work2
Normal	Pulling	20s	kubelet	Pulling image "master:30500/project-hosted/flask-user:latest"
Normal	Pulled	20s	kubelet	Successfully pulled image "master:30500/project-hosted/flask-user:latest" in 22ms (38ms including waiting)
Normal	Created	20s	kubelet	Created container: flask-user
Normal	Started	20s	kubelet	Started container flask-user



08 향후 계획

08 향후 계획

서비스 안정성과 확장성 강화

- 마스터 노드 및 워커 노드 수 확장에 대비한 클러스터 확장 계획 수립
- 리소스 사용량 기반 오토스케일링(HPA, VPA) 기능 도입 검토
- Pod 간 부하 분산을 위한 Ingress Controller 구성 고려

고가용성(HA) 구조 도입

- 현재 단일 복제본으로 구성된 MySQL에 대해 StatefulSet + Read Replica 구성 시나리오 검토
- Prometheus Alertmanager 도입을 통한 장애 알림 시스템 강화

데이터 및 스토리지 안정성 확보

- Ceph, Longhorn 등 분산 블록 스토리지 솔루션 도입을 통한 데이터 내결함성 확보
- 정기 백업 및 자동 복구 시스템 구성 (예: Velero, cronjob 스크립트 등)

클라우드 이전 및 서비스 확장 고려

- 온프레미스 기반 클러스터에서 AWS EKS, GCP GKE 등 클라우드 환경으로 이전 가능성 탐색
- 스토리지(PV/PVC) 및 로드밸런서 구성 변경에 따른 마이그레이션 전략 수립
- 모니터링 및 로깅 스택을 Cloud-native 방식으로 재정비 (예: CloudWatch, Stackdriver 등)

08 향후 계획

Ceph 도입 검토

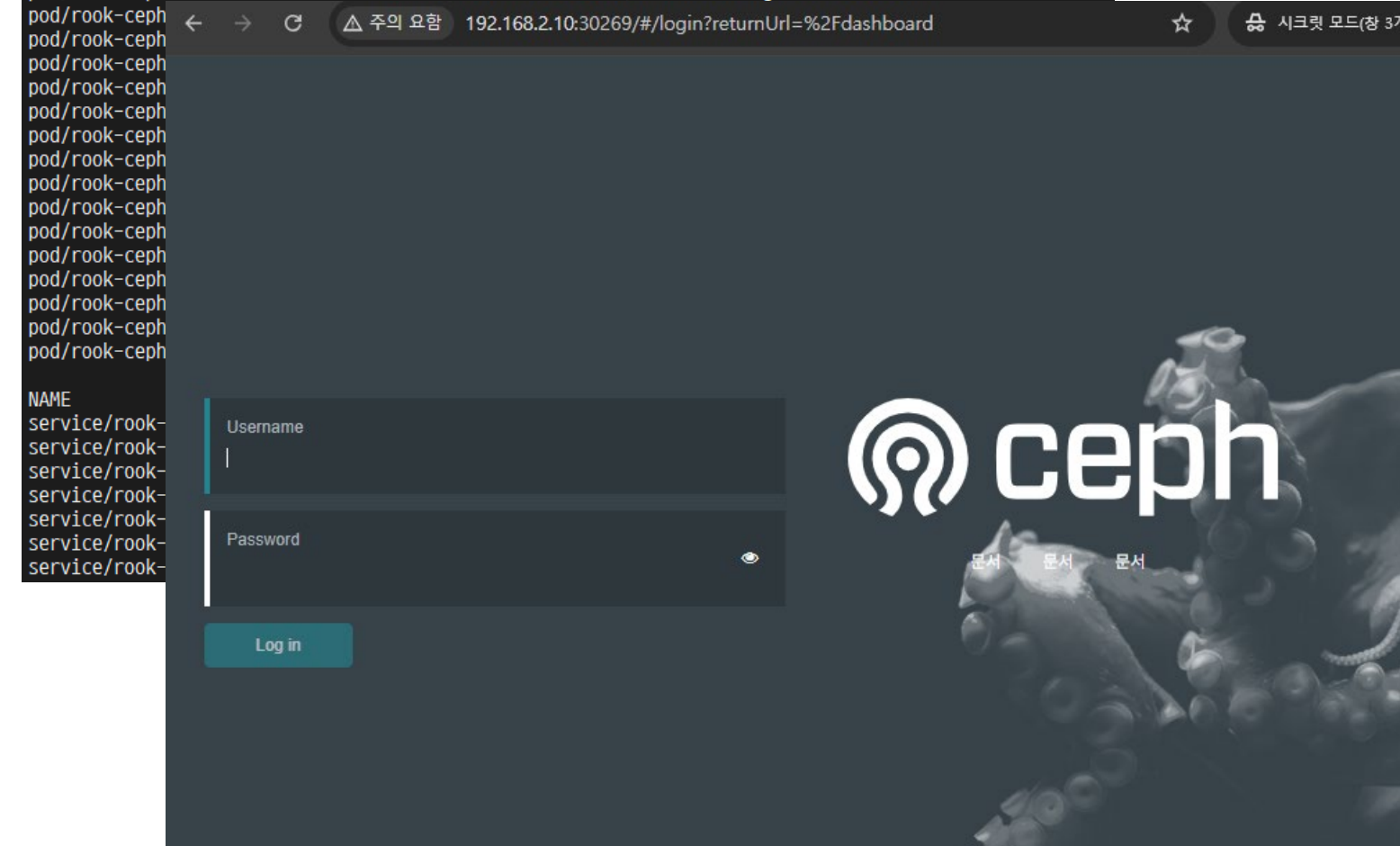
- 고가용성 및 데이터 내결함성 확보를 위해 Ceph 기반의 분산 스토리지 도입 예정
- 실제 6대 노드 기반 환경에서 Rook-Ceph 오퍼레이터를 활용한 클러스터 구성 시도 완료

현재 제외 사유

- Ceph는 안정적인 서비스 운영을 위해 다수의 스토리지 노드와 높은 네트워크 자원을 요구
 - 현재 6대 노드 환경에서 Ceph 운영을 위한 충분한 리소스 확보가 어려움
- ✓ 추후 노드 확장 및 인프라 고도화 시 Ceph 도입
재검토 예정

```
root@master:~# kubectl get all -n rook-ceph
```

NAME	READY	STATUS	RESTARTS	AGE
pod/csi-cephfsplugin-4wg85	3/3	Running	6 (18m ago)	43m
pod/csi-cephfsplugin-mdnmq	3/3	Running	6 (17m ago)	43m
pod/csi-cephfsplugin-provisioner-d8ccbc6bc-f62c8	6/6	Running	7 (21m ago)	43m
pod/csi-cephfsplugin-provisioner-d8ccbc6bc-v28qr	6/6	Running	7 (21m ago)	43m
pod/csi-cephfsplugin-sxdb7	3/3	Running	6 (18m ago)	43m
pod/csi-rbdplugin-provisioner-68b5f754c9-gs9jl	6/6	Running	0	3m54s
pod/csi-rbdplugin-provisioner-68b5f754c9-st8rn	6/6	Running	8 (21m ago)	43m
pod/csi-rbdplugin-qlvf8	3/3	Running	6 (17m ago)	43m
pod/csi-rbdplugin-sblrk	3/3	Running	6 (18m ago)	43m
pod/csi-rbdplugin-zr25w	3/3	Running	6 (18m ago)	43m
pod/rook-ceph-crashcollector-work1-d899c4577-t4vhg	1/1	Running	0	2m1s
pod/rook-ceph-crashcollector-work2-6c8fc79486-csvkv	1/1	Running	0	4m35s



감사합니다