컨테이너화 및 배포 가이드

0. 산출물

1. 컨테이너 구성

Frontend : React + Nginx

Backend : Spring Boot

AI : Python, Gunicorn

Database : mariadb, redis

File Server : Nginx

NFS Server : Ubuntu

2. 도커 설정

2.1 AI Dockerfile

```
FROM python:3.12-slim-bookworm


WORKDIR /app

# 의존성 패키지 설치.
RUN apt-get update && apt-get install -y --no-install-recommends \
    default-libmysqlclient-dev \
    build-essential \
    && rm -rf /var/lib/apt/lists/*

# Flask 의존성 설치
```

```
COPY requirements.txt .

RUN pip install --no-cache-dir -U -r requirements.txt


# 애플리케이션 코드 복사
COPY . .

# 디렉토리 미리 생성
RUN mkdir -p report/valid report/invalid report/attack


EXPOSE 5000

# 서버 실행
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "main:app"]
```

2.2 Backend Dockerfile

```
# 빌드 환경
FROM openjdk:17-alpine AS builder

RUN apk update && \
    apk add findutils && \
    rm -rf /var/cache/apk/*

WORKDIR /app

COPY .mvn .mvn/
COPY mvnw .
COPY mvnw.cmd .
COPY ./.mvn/wrapper/maven-wrapper.properties
./.mvn/wrapper/maven-wrapper.properties
RUN chmod +x mvnw


# 의존성 다운로드
COPY pom.xml ./
RUN ./mvnw dependency:go-offline -B
```

```
# 소스 코드 복사
COPY src ./src

# 빌드
RUN ./mvnw clean package -DskipTests


# 런타임 환경
FROM openjdk:17-alpine

RUN addgroup -S -g 1000 spring && adduser -S -u 1000 -G spring spring
USER spring

WORKDIR /app

COPY --from=builder /app/target/*.jar app.jar


EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]
```

2.3 Frontend Dockerfile

```
# 빌드
FROM node:22 AS builder

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

# 배포
FROM nginx:latest
```

```
COPY nginx.conf /etc/nginx/nginx.conf


RUN mkdir -p /app/log/nginx/client && chown -R nginx:nginx /app/log &&
chmod -R 755 /app/log

RUN rm -f /etc/nginx/conf.d/default.conf

COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

2.4 File Server Dockerfile

```
# Dockerfile (nginx/Dockerfile)
FROM nginx:latest

# 기본 Nginx 설정 파일을 제거하고 사용자 정의 설정 파일로 교체
# /etc/nginx/nginx.conf 는 Nginx의 메인 설정 파일입니다.
COPY nginx.conf /etc/nginx/nginx.conf
```

3. Nginx 설정

3.1 client nginx 설정

```
# nginx.conf

# 사용자 및 작업자 프로세스 설정
user  nginx;
worker_processes  auto;

error_log  /app/log/nginx/client/error.log warn;
pid        /var/run/nginx.pid;
```

```
events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local]
"$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /app/log/nginx/client/access.log  main;

    sendfile        on;

    keepalive_timeout  65;

    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        server_name  localhost;

        root   /usr/share/nginx/html;
        index  index.html index.htm;

        location / {
            try_files $uri $uri/ /index.html;
        }

    }
}
```

3.2 file server nginx 설정

```
# nginx/nginx.conf 파일 수정
user  nginx;
worker_processes  auto;

```

```
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local]
"$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;

    keepalive_timeout  65;


    server {
        listen       80;
        server_name  localhost;

        # health check
        location / {
            return 200 'OK';
            add_header Content-Type text/plain;
        }

        location /document/ {
            alias /app/document/;
            autoindex on;
            # CORS 설정
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, 
OPTIONS';
            add_header 'Access-Control-Allow-Headers' 
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-T
ype,Range';
            add_header 'Access-Control-Expose-Headers' 
'Content-Length,Content-Range';

            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, 
OPTIONS';
                add_header 'Access-Control-Allow-Headers' 
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-T
ype,Range,Authorization';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain; charset=utf-8';
                add_header 'Content-Length' 0;
                return 204;
            }
        }

        location /report/ {
            alias /app/report/;
            autoindex on;
            # CORS 설정
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, 
OPTIONS';
            add_header 'Access-Control-Allow-Headers' 
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-T
ype,Range';
            add_header 'Access-Control-Expose-Headers' 
'Content-Length,Content-Range';

            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, 
OPTIONS';
```

```
                add_header 'Access-Control-Allow-Headers'
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-T
ype,Range,Authorization';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain; charset=utf-8';
                add_header 'Content-Length' 0;
                return 204;
            }
        }

        error_page  404 /404.html;
        location = /404.html {
            root   /usr/share/nginx/html;
        }

        location /error/ {
            default_type "text/html";
            return 400 "<h1>잘못된 접근입니다.</h1><p>요청하신 URL을 찾을 수
없거나 접근 권한이 없습니다.</p>";
        }
    }
}
```

4. 산출물. Kubernetes 배포 설정

4.1 AI 배포

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: flask-config
  namespace: rookies-app
data:
  FLASK_ENV: "production"
  FLASK_DEBUG: "False"
  FLASK_APP: "main.py"
  PORT: "5000"
  LOG_LEVEL: "INFO"
  LOG_FILE: "security_analysis.log"
```

```yaml
  ENABLE_METRICS: "True"
  METRICS_PORT: "9090"
  CACHE_TYPE: "simple"
  CACHE_DEFAULT_TIMEOUT: "300"

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rookies-ai-server-deployment
  namespace: rookies-app
  labels:
    app: rookies-ai-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rookies-ai-server
  template:
    metadata:
      labels:
        app: rookies-ai-server
    spec:
      containers:
      - name: ai-server
        image: [image path]
        imagePullPolicy: Always
        ports:
        - containerPort: 5000
          protocol: TCP
        env:
          - name: DB_HOST
            value: "mariadb-server-service"
          - name: DB_PORT
            value: "3306"
          - name: DB_USER
            valueFrom:
              secretKeyRef:
                name: mariadb-secret
                key: user-name
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mariadb-secret
                key: user-password
          - name: DB_NAME
            valueFrom:
              secretKeyRef:
                name: mariadb-secret
                key: database-name


          - name: OPENAI_API_KEY
            valueFrom:
              secretKeyRef:
                name: ai-secret
                key: openai-api-key


          - name: FLASK_ENV
            valueFrom:
              configMapKeyRef:
                name: flask-config
                key: FLASK_ENV
          - name: FLASK_DEBUG
            valueFrom:
              configMapKeyRef:
                name: flask-config
                key: FLASK_DEBUG
          - name: FLASK_APP
            valueFrom:
              configMapKeyRef:
                name: flask-config
                key: FLASK_APP
          - name: PORT
            valueFrom:
              configMapKeyRef:
                name: flask-config
                key: PORT
          - name: LOG_LEVEL
```

```
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: LOG_LEVEL
        - name: LOG_FILE
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: LOG_FILE
        - name: ENABLE_METRICS
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: ENABLE_METRICS
        - name: METRICS_PORT
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: METRICS_PORT
        - name: CACHE_TYPE
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: CACHE_TYPE
        - name: CACHE_DEFAULT_TIMEOUT
          valueFrom:
            configMapKeyRef:
              name: flask-config
              key: CACHE_DEFAULT_TIMEOUT

      readinessProbe:
        httpGet:
          path: /health
          port: 5000
        initialDelaySeconds: 15
        periodSeconds: 10
        timeoutSeconds: 5
        failureThreshold: 3
      livenessProbe:
        httpGet:
          path: /health
          port: 5000
        initialDelaySeconds: 30
        periodSeconds: 10
        timeoutSeconds: 5
        failureThreshold: 3
      resources:
        requests:
          memory: "128Mi"
          cpu: "100m"
        limits:
          memory: "256Mi"
          cpu: "150m"
      volumeMounts:
      - name: document-storage
        mountPath: /app/document
      - name: report-storage
        mountPath: /app/report
      - name: log-storage
        mountPath: /app/logs
    volumes:
    - name: document-storage
      persistentVolumeClaim:
        claimName: document-pvc
    - name: report-storage
      persistentVolumeClaim:
        claimName: report-pvc
    - name: log-storage
      persistentVolumeClaim:
        claimName: log-pvc
```

4.2 API 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rookies-api-server-deployment
  namespace: rookies-app
  labels:
    app: rookies-api-server
```

```yaml
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rookies-api-server
  template:
    metadata:
      labels:
        app: rookies-api-server
    spec:
      securityContext:
        runAsUser: 1000
        runAsGroup: 1000
        fsGroup: 1000
      containers:
      - name: api-server
        image: [image path]
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
        readinessProbe:
          httpGet:
            path: /actuator/health
            port: 8080
          initialDelaySeconds: 90
          periodSeconds: 10
          timeoutSeconds: 5
          failureThreshold: 3
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: 8080
          initialDelaySeconds: 60
          periodSeconds: 10
          timeoutSeconds: 5
          failureThreshold: 3
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1024Mi"
            cpu: "750m"
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: prod
        - name: MARIADB_USERNAME
          valueFrom:
            secretKeyRef:
              name: mariadb-secret
              key: user-name
        - name: MARIADB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mariadb-secret
              key: user-password
        - name: REDIS_PASSWORD
          valueFrom:
            secretKeyRef:
              name: redis-password
              key: password
        - name: JWT_SECRET
          valueFrom:
            secretKeyRef:
              name: api-secret
              key: jwt-secret
        - name: SPRING_SECURITY_USER_NAME
          valueFrom:
            secretKeyRef:
              name: api-secret
              key: spring-security-username
        - name: SPRING_SECURITY_USER_PASSWORD
          valueFrom:
            secretKeyRef:
              name: api-secret
              key: spring-security-password
        volumeMounts:
        - name: document-storage
          mountPath: /app/document
        - name: log-storage
```

```
          mountPath: /app/logs
      volumes:
      - name: document-storage
        persistentVolumeClaim:
          claimName: document-pvc
      - name: log-storage
        persistentVolumeClaim:
          claimName: log-pvc
```

4.3 Client 배포

```
# client-server-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rookies-client-server-deployment
  namespace: rookies-app
  labels:
    app: rookies-client-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rookies-client-server
  template:
    metadata:
      labels:
        app: rookies-client-server
    spec:
      containers:
      - name: nginx
        image: [image path]
        command: ["/bin/bash", "-c"]
        args:
          - mkdir -p /app/log/nginx/client && nginx -g 'daemon off;'
        imagePullPolicy: Always
        ports:
          - containerPort: 80
        resources:
          requests:
            memory: "256Mi"
            cpu: "100m"
          limits:
            memory: "512Mi"
            cpu: "150m"
        volumeMounts:
          - name: log-storage
            mountPath: /app/log

      volumes:
        - name: log-storage
        persistentVolumeClaim:
          claimName: log-pvc
```

4.4 File Server 배포

```
# client-server-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rookies-client-server-deployment
  namespace: rookies-app
  labels:
    app: rookies-client-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rookies-client-server
  template:
    metadata:
      labels:
        app: rookies-client-server
    spec:
      containers:
        - name: nginx
          image: [image path]
          command: ["/bin/bash", "-c"]
          args:
            - mkdir -p /app/log/nginx/client && nginx -g 'daemon off;'
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          resources:
            requests:
              memory: "256Mi"
              cpu: "100m"
            limits:
              memory: "512Mi"
              cpu: "150m"
          volumeMounts:
            - name: log-storage
              mountPath: /app/log

      volumes:
        - name: log-storage
          persistentVolumeClaim:
            claimName: log-pvc
```

4.5 MariaDB 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb-deployment
  namespace: rookies-app
  labels:
    app: mariadb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mariadb
  template:
    metadata:
      labels:
        app: mariadb
```

```yaml
    spec:
      securityContext:
        fsGroup: 999
      volumes:
        - name: mariadb-storage
          persistentVolumeClaim:
            claimName: mariadb-pvc
      containers:
        - name: mariadb
          image: mariadb:latest
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mariadb-secret
                  key: root-password
            - name: MYSQL_DATABASE
              valueFrom:
                secretKeyRef:
                  name: mariadb-secret
                  key: database-name
            - name: MYSQL_USER
              valueFrom:
                secretKeyRef:
                  name: mariadb-secret
                  key: user-name
            - name: MYSQL_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mariadb-secret
                  key: user-password
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mariadb-storage
              mountPath: /var/lib/mysql
          resources:
            requests:
              memory: "256Mi"
              cpu: "200m"
            limits:
              memory: "512Mi"
              cpu: "500m"
```

4.6 Redis 배포

```
# redis-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rookies-redis-deployment
  namespace: rookies-app
  labels:
    app: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - name: redis
          image: redis:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 6379
          volumeMounts:
            - name: redis-data
              mountPath: /data
          command: ["redis-server"]
          args: ["--appendonly", "yes", "--requirepass",
"$(REDIS_PASSWORD)"]
          env:
            - name: REDIS_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: redis-password
                  key: password
          resources:
            requests:
              memory: "128Mi"
              cpu: "100m"
            limits:
              memory: "512Mi"
              cpu: "200m"
      volumes:
        - name: redis-data
          persistentVolumeClaim:
            claimName: redis-pvc
```

4.7 ingress 설정

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: multi-app-ingress
  namespace: rookies-app
  labels:
    app.kubernetes.io/name: multi-app-ingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: "rookies-app-static-ip"
    networking.gke.io/managed-certificates: "rookies-managed-cert"
    kubernetes.io/ingress.class: "gce"
    # ingress.kubernetes.io/force-ssl-redirect: "true"
spec:
  rules:
    - host: files.rookies-app.com
      http:
        paths:
          - path: /
```

```yaml
          pathType: Prefix
          backend:
            service:
              name: file-server-service
              port:
                number: 80
  - host: client.rookies-app.com
    http:
      paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: client-server-service
              port:
                number: 80
  # - host: ai.rookies-app.com
  #   http:
  #     paths:
  #       - path: /
  #         pathType: Prefix
  #         backend:
  #           service:
  #             name: ai-server-service
  #             port:
  #               number: 80

  - host: api.rookies-app.com
    http:
      paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: api-server-service
              port:
                number: 80
```

환경 설정 파일

```yaml
## Database Secret ##
apiVersion: v1
kind: Secret
metadata:
  name: mariadb-secret
  namespace: rookies-app
type: Opaque
stringData:
  root-password: "rookies"
  user-name:   "rookies"
  user-password: "rookies"
  database-name: "log2doc"
## Database Secret End ##\


---

## Redis Secret End ##
apiVersion: v1
kind: Secret
metadata:
  name: redis-password
  namespace: rookies-app
type: Opaque
stringData:
  password: "1234"
## Redis Secret End ##


---

## API Secret ##
apiVersion: v1
kind: Secret
metadata:
  name: api-secret
  namespace: rookies-app
type: Opaque
stringData:
  jwt-secret: "mySecretKey1234567890123456789012345678901234567890"
  spring-security-username: "admin"
  spring-security-password: "admin123"
## API Secret End ##

---


## AI Secret ##
apiVersion: v1
kind: Secret
metadata:
  name: ai-secret
  namespace: rookies-app
type: Opaque
data:
  openai-api-key:
## AI Secret End ##
```

5. 배포 가이드

~~~
## 배포 순서
### 0. secret 설정


### 1. namespace 제작
```bash
kubectl apply -f gke-namespace-settings.yaml
```
### 2. 저장소 pv, pvc 제작
```bash
kubectl apply -f gke-pv-pvc-settings.yaml
```
~~~

~~~
### 3. 네트워크 service 제작
```bash
kubectl apply -f gke-service-settings.yaml
```
### 4. 배포 server 제작
```bash
kubectl apply -f ./mariadb_server/mariadb-deployment.yaml
kubectl apply -f ./file_server/file-server-deployment.yaml
kubectl apply -f ./client_server/client-server-deployment.yaml
kubectl apply -f ./api_server/api-server-deployment.yaml
kubectl apply -f ./ai_server/ai-server-deployment.yaml
```
### 5 healthcheck 제작
```bash
kubectl apply -f gke-backend-config.yaml
```

### 6. HTTP Route 를 위한 ingress 설정
```bash
kubectl apply -f gke-ingress-settings.yaml
```

### 7. SSL 를 위한 managedCertificate 설정
```bash
kubectl apply -f gke-managed-certificate.yaml
```
~~~