포팅메뉴얼
(무근본배틀)

작성자
K12B206T
상승규

목 차

1. 개발 환경

사용 툴	버전 및 사양
UBUNTU	22.04.5
AWS LIGHTSAIL	Vcpu 2, RAM 8GB / Vcpu 2, RAM 4GB
AWS EC2	Vcpu 4, RAM 16GB
Kubernetes	1.29.15
Kustomize	5.0.4
Docker	28.1.1
Containerd	1.7.27
Python	3.10.12
Ansible-core	2.14.18
Java	17
Spring boot	3.4.4
Spring Data Redis	
Spring Data Mongo	
Redis	
Mongo	

2. 환경 변수

- Backend

MONGO_AUTH=admin
MONGO_CONNECT_NAME=S12P31B206
MONGO_DATABASE=S12P31B206
MONGO_HOST=ssafy.ngivl.mongodb.net
MONGO_PASSWORD=Xcj3WvZssD
MONGO_PORT=27017
MONGO_USERNAME=S12P31B206
OPENAI_KEY=sk-proj-
FiT_IE4s9Ck2rv6DGteoFaOXLeW9DTRyr54nJX6qwaloWtCNwod9Poe0IG7f6wqi2OeLvirgErT3BlbkFJHL
zsVv5lKQPlUil5G18pztu6zAjROB1R2ErpqoL6XZZeWTeNzBtD5K0YITUqN7gvWBw_Zit0YA
REDIS_HOST=localhost
REDIS_PORT=6379
S3_ACCESSKEY=AKIA4IM3HBAPUTNAV6AX
S3_BUCKETNAME=nobasebattle-s3
S3_SECRETKEY=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
JWT_SECRET_KEY=asdwdasdawdasdasdasdqwdasdgretqwfsdfgsdfsefgwegweg
JWT_ACCESS_EXP=1111111
JWT_HEADER=Authorization
JWT_PREFIX=Bearer
LOGGING_DIR=/log

- Frontend

NEXT_PUBLIC_BASE_URL = 'http://3.36.130.152'
NEXT_PUBLIC_GOOGLE_ANALYTICS_ID = 'G-FZ3Y42M3BN'

- Nginx

```
server {
        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        ssl_certificate /etc/letsencrypt/live/nobasebattle.com/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/nobasebattle.com/privkey.pem; # managed by
Certbot
        add_header Strict-Transport-Security "max-age=31536000";
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nobasebattle.com www.nobasebattle.com;
```

```
        access_log /var/log/nginx/access.log main;
        location ^~ /api/next {
                proxy_pass http://localhost:3000;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }
        location ^~ /api {
                proxy_pass http://localhost:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }
        location / {
           proxy_pass http://localhost:3000;
           proxy_set_header Host $host;
           proxy_set_header X-Real-IP $remote_addr;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
           proxy_set_header Host $http_host;
        }
}
server {
    listen 80;
    listen [::]:80;
    server_name nobasebattle.com;
    if ($host = nobasebattle.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
}
```

- Jenkins (front)

```
pipeline {
    agent any

    environment {
        KUBECONFIG = credentials('kubeconfig')
        IMAGE_TAG = "fdev-${env.BUILD_NUMBER}"
        IMAGE_URL = "docker.io/unrequiredone/nobasebattle:${IMAGE_TAG}"
```

```
        NEXT_PUBLIC_BASE_URL = 'http://3.36.130.152'
        NEXT_PUBLIC_GOOGLE_ANALYTICS_ID = 'G-FZ3Y42M3BN'
    }

    stages {
        stage('Clone') {
            steps {
                git url: 'https://lab.ssafy.com/s12-final/S12P31B206.git',
                    branch: 'front-develop',
                    credentialsId: 'labssafy'
            }
        }

        stage('Build & Push Image') {
            steps {
                dir('frontend/nobasebattle') {
                    withCredentials([usernamePassword(credentialsId:                    'dockerhub',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
                        catchError(buildResult: 'FAILURE', stageResult: 'FAILURE') {
                            sh '''
                            docker login -u $DOCKER_USER -p $DOCKER_PASS

                            docker build -t $IMAGE_URL ₩
                              --build-arg NEXT_PUBLIC_BASE_URL=$NEXT_PUBLIC_BASE_URL ₩
                              --build-arg
NEXT_PUBLIC_GOOGLE_ANALYTICS_ID=$NEXT_PUBLIC_GOOGLE_ANALYTICS_ID .

                            docker push $IMAGE_URL
                            '''
                        }
                    }
                }
            }
        }

        stage('Deploy to K8s Dev') {
            steps {
                dir('infra-deploy/apps/nextjs/overlays/dev') {
```

```
                    sh '''
                    kustomize edit set image nextjs=$IMAGE_URL
                    kubectl apply -k .
                    '''
                }
            }
        }
    }

    post {
        success {
            script {
                def authorId = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def authorEmail = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'good',
                    message: """✅ 프론트 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER}
- 브랜치 : front-develop
- 작성자 : ${authorId} (${authorEmail})
- 이미지 : ${IMAGE_URL}
- [빌드 상세보기](${env.BUILD_URL})"""
                )
            }
        }

        failure {
            script {
                def authorId = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def authorEmail = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend(
                    color: 'danger',
                    message: """❌ 프론트 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER}
- 브랜치 : front-develop
- 작성자 : ${authorId} (${authorEmail})
- [빌드 상세보기](${env.BUILD_URL})"""
                )
            }
        }
    }
```

}

- Jenkins (backend)

```
pipeline {
    agent any

    environment {
        KUBECONFIG = credentials('kubeconfig')
        IMAGE_TAG = "dev-${env.BUILD_NUMBER}"
        IMAGE_URL = "docker.io/unrequiredone/nobasebattle:${IMAGE_TAG}"
    }

    stages {
        stage('Clone') {
            steps {
                git url: 'https://lab.ssafy.com/s12-final/S12P31B206.git',
                    branch: 'backend-develop',
                    credentialsId: 'labssafy'
            }
        }

        stage('Inject Env ConfigMap') {
            steps {
                withCredentials([file(credentialsId: 'env', variable: 'ENV_FILE')]) {
                    sh '''
                    mkdir -p infra-deploy/apps/springboot/overlays/dev
                    kubectl create configmap spring-env ₩
                      --from-env-file=$ENV_FILE ₩
                      -n k8m ₩
                      --dry-run=client -o yaml > infra-deploy/apps/springboot/overlays/dev/spring-env.yaml
                    '''
                }
            }
        }

        stage('Build & Push Image') {
            steps {
                dir('backend/nobasebattle') {
                    withCredentials([usernamePassword(credentialsId: 'dockerhub',
```

```
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
                                catchError(buildResult: 'FAILURE', stageResult: 'FAILURE') {
                                    sh '''
                                    docker login -u $DOCKER_USER -p $DOCKER_PASS

                                    chmod +x gradlew
                                    ./gradlew clean build -x test

                                    docker build -t $IMAGE_URL .
                                    docker push $IMAGE_URL
                                    '''
                                }
                            }
                        }
                    }

                    stage('Deploy to K8s Dev') {
                        steps {
                            dir('infra-deploy/apps/springboot/overlays/dev') {
                                sh '''
                                kustomize edit set image springboot=$IMAGE_URL
                                kubectl apply -k .
                                '''
                            }
                        }
                    }
                }

                post {
                    success {
                        script {
                            def authorId = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                            def authorEmail = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                            mattermostSend(
                                color: 'good',
                                message: """✅ 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER}
- 브랜치 : backend-develop
- 작성자 : ${authorId} (${authorEmail})
```

```
- 이미지 : ${IMAGE_URL}
- [빌드 상세보기](${env.BUILD_URL})"""
                    )
                }
            }

            failure {
                script {
                    def authorId = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                    def authorEmail = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                    mattermostSend(
                        color: 'danger',
                        message: """❌ 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER}
- 브랜치 : backend-develop
- 작성자 : ${authorId} (${authorEmail})
- [빌드 상세보기](${env.BUILD_URL})"""
                    )
                }
            }
        }
}
```

3. 필수 소프트웨어

Kubernetes

Kustomize

Docker

Nginx

4. 배포 및 설치

1. 사전 준비 사항

Ubuntu 22.04 이상 설치된 서버

다음 포트 오픈 필요: 80, 443, 22, 3000, 8080, 5000, 9000, 6443 등

Docker / Kubernetes 환경 구성 완료 (kubeadm init 및 kubeadm join 완료)

Jenkins 설치 및 docker 권한 부여 (sudo usermod -aG docker $USER / newgrp docker)

GitLab 또는 코드 저장소 준비

2. 시작

kubectl apply -f infra/jenkins/

nginx 설정하기

jenkins에 pipeline 연결

각 script 실행하여 키기