

2025 DevSecOps 환경 구축 가이드



Korea IT DevSecOps

2025. 02. 20

서론

2025 DevSecOps 환경 구축 가이드는 DevSecOps 방식을 통해 개발, 운영, 보안을 효과적으로 통합하여 CI/CD 프로세스를 최적화하고 시스템의 안전성을 강화하는 방법을 제시합니다. 특히, Jenkins, SonarQube, Docker, Kubernetes와 Wazuh을 활용하여 소스 코드 품질을 개선하고 보안 자동화를 구현하는 구체적인 접근법을 제공합니다.

이 문서는 다음을 목표로 합니다:

1. 개발 산성 증대

CI/CD 파이프라인 자동화를 통해 효율적인 빌드 및 배포 프로세스 구현.

2. 보안 내재화

코드 분석 및 취약점 점검 도구를 활용하여 개발 초기 단계에서부터 보안을 강화.

3. 안정적인 환경 구축

Kubernetes 기반 확장 가능하고 안정적인 애플리케이션 실행 환경 설계.

4. 실시간 모니터링 및 분석

Wazuh를 사용하여 시스템 로그, 보안 이벤트를 통합 관리하고, 빠른 문제 분석과 대응지원.

5. 협업화 촉진

개발, 운영, 보안 팀 간의 긴밀한 협력을 통해 조직 전반의 보안 인식 제고.

본 가이드는 실무적인 설정 및 적용 사례를 중심으로 DevSecOps 환경을 단계적으로 구축하는 방법을 설명하며, 이를 통해 안전하고 효율적인 소프트웨어 개발, 운영, 그리고 보안을 제공합니다.

팀 소개



Team DevSecOps

기획

정재호 장진영 박진우 안정훈 지윤정

개발

배준성 윤광혁 최민재 장태경

운영

김범준 김지홍 노윤서 손혜영 이주원

Lab Git Repository

<https://github.com/jinyeong001/DevSecOps.Full-Project.git>

https://github.com/GH6679/web_wargamer.git

목차

2025 DEVSECOPS 환경 구축 가이드.....	1
서론	1
팀 소개	2
LAB GIT REPOSITORY.....	2
1. DEVSECOPS.....	5
1.1. DevSecOps 란?.....	5
1.2. DevSecOps의 이점.....	6
2. 사전 준비 사항	7
2.1 사전 요구 사항 (필요 설치 파일).....	7
2.2 사전 요구 사항 설치.....	7
3. YAML 작성법	10
3.1 yaml 이란?.....	10
3.2 주요 필드.....	10
3.3 워커 노드 추가.....	11
3.4 기타 필드.....	12
4. KUBERNETES	15
4.1 Kubernetes 란?.....	15
4.2 DevSecOps에서 Kubernetes의 역할.....	16
4.3 kind.....	16
4.4 오류 발생 또는 충돌 시.....	19
5. JENKINS	20
5.1 Jenkins 란?.....	20
5.2 DevSecOps에서의 Jenkins의 역할.....	20
5.3 Jenkins worker node 생성.....	21
5.4 Jenkins worker node 실행.....	24
6. WAZUH	30
6.1 Wazuh 란?.....	30
6.2 Wazuh의 역할.....	30
6.3 Wazuh 사용계획 및 기대되는 이점.....	30
7. WEB 구축	31

7.1. WebPage 소스코드	31
7.2. Web worker node 생성	32
7.3 Web worker node 실행	36



1. DevSecOps

1.1. DevSecOps 란?

DevSecOps는 개발(Development), 보안(Security), 운영(Operations)을 통합하여 소프트웨어 개발의 전 과정에서 보안을 자동화하고 내재화하는 방법론입니다. 기존 DevOps 모델은 개발과 운영 간 협업에 초점을 맞췄으나, 보안은 종종 후순위로 다뤄졌습니다. 이에 반해 DevSecOps는 초기 설계 단계부터 배포, 운영에 이르기까지 보안을 모든 과정에 포함시키며, 이를 통해 **보안은 모두의 책임**이라는 문화를 강조합니다.

DevSecOps의 주요 원칙은 다음과 같습니다:

- **자동화된 보안 통합:** CI/CD 파이프라인에 보안 도구를 통합하여 코드 분석, 취약점 점검, 종속성 관리를 자동화합니다.
- **보안 중심의 협업:** 개발, 운영, 보안 팀이 협력하여 보안 문제를 조기에 발견하고 해결합니다.
- **지속적인 개선:** 위협 모델링과 실시간 모니터링으로 보안을 지속적으로 강화합니다.

우리는 DevSecOps 구현을 위해 Jenkins와 같은 CI/CD 도구를 기반으로 SonarQube로 코드 품질 및 보안 점검을 수행하며, Wazuh를 활용해 시스템 로그를 분석하고 이상 행동을 탐지합니다. 또한, Docker와 Kubernetes로 컨테이너 보안을 강화하고, Falco 등의 런타임 보안 도구로 운영 환경의 보안성을 유지합니다.

이러한 접근법은 보안을 단순한 방어 전략이 아닌 개발과 운영의 핵심 요소로 자리 잡게 합니다. 이를 통해 조직 내 보안 문화가 정착되고, 빠르고 안전한 소프트웨어 배포가 가능해집니다.

1.2. DevSecOps의 이점

DevSecOps를 채택하면 다양한 이점을 누릴 수 있습니다. 먼저, CI/CD 파이프라인에서 자동화된 보안 테스트와 취약점 분석 도구를 활용함으로써 개발 초기부터 보안을 강화할 수 있습니다. 보안이 개발 초기 단계에 통합되면, 후반 단계에서 보안 취약점을 수정하는 데 드는 비용을 크게 절감할 수 있습니다. 이는 보안 문제를 사전에 식별하고 해결하는 데 중요한 역할을 하며, 전반적인 개발 비용 절감으로 이어집니다.

또한, DevOps의 자동화와 속도를 유지하면서도 보안 검사를 주기적으로 수행할 수 있습니다. 이를 통해 안전한 코드 배포가 보장되며, 지속적인 통합 및 배포(CI/CD)를 지원하는 보안 관행이 자연스럽게 형성됩니다. 주기적인 보안 점검을 통해 개발과 배포 과정에서 발생할 수 있는 보안 리스크를 사전에 최소화할 수 있습니다.

DevSecOps에서는 보안이 단지 보안 팀의 책임만이 아닙니다. 개발자와 운영 팀도 보안에 대한 책임을 공유하고 협업하는 문화가 필요합니다. 이를 통해 조직 내 보안 인식 수준이 높아지고, 더 빠르고 효율적인 보안 대응이 가능합니다. 또한, 위협 모델링과 모니터링을 통해 실시간으로 보안을 관리하고, 발생할 수 있는 보안 문제를 예방할 수 있습니다. 이러한 방식은 다양한 산업에서 요구하는 보안 표준과 규제를 준수하는 데도 유효합니다.

이와 같은 이유로 우리는 DevSecOps를 채택하여 개발, 보안, 운영이 유기적으로 협력하고 효율적으로 관리될 수 있도록 하는 통합 프로젝트를 추진하고 있습니다. DevSecOps 환경을 통해 보안을 강화하면서도 개발과 운영의 효율성을 유지할 수 있는 시스템을 구축할 수 있습니다.

2. 사전 준비 사항

2.1 사전 요구 사항 (필요 설치 파일)

- WSL(WSL2)
- Docker Desktop(Kind)
- Git
- Chocolatey
- Kuberctl
- 구축파일(.yaml)
- localtunnel

본 프로젝트의 작업은 바탕화면에서 진행됩니다.

Docker Desktop 설치 후 설정에서 Kubernetes 해제 필요 (기본 설정)

Kubernetes

v1.30.5



Enable Kubernetes

Start a Kubernetes single-node cluster when starting Docker Desktop.



Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

2.2 사전 요구 사항 설치

Window PowerShell 관리자 권한으로 실행

1) WSL(Windows Subsystem for Linux) 설치

Windows에서 Linux 환경을 구축할 수 있게 해주는 핵심 기능을 한다.

```
# WSL 설치
1. dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
2. dism.exe /online /enable-feature /featurename:VirtualMachinePlatform
```

```
/all /norestart

# WSL2 기본 버전으로 설정
3. wsl -set-default-version 2
```

2) Chocolatey 설치

Chocolatey는 Windows용 패키지 관리자 도구이다.

```
# Chocolatey 패키지 관리자 설치
1. Set-ExecutionPolicy Bypass -Scope Process -Force
2. [System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072
3. iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.o
rg/install.ps1'))
```

3) Git 설치

Github 및 Jenkins를 통한 자동화 핵심 기능을 한다.

```
# Git 기본 패키지 설치
1. choco install git -y
```

4) Docker Desktop, kubectl 및 kind 설치

Kind(Kubernetes IN Docker)는 Docker 컨테이너를 사용하여 Kubernetes 클러스터를 쉽게 배포하고 관리할 수 있게 해주는 도구이다..

```
# Docker Desktop 설치
1. choco install docker-desktop -y

# kubectl과 kind 설치
2. choco install kubernetes-cli -y
3. choco install kind -y
```

5) Localtunnel 설치

localtunnel 설명 추가

```
# node.js 다운로드
1. https://nodejs.org/ko/
```

Window cmd 관리자 권한 실행

```
# localtunnel 다운로드
2. npm install -g localtunnel
```

DevSecOps

Localtunnel 사용 방법

```
# Window cmd 관리자 권한 실행  
1. lt -subdomain <도메인 이름> --port <사용될 포트>  
# 터널 비밀번호  
2. https://loca.lt/mytunnelpassword
```



3. yaml 작성법

3.1 yaml이란?

YAML은 "YAML Ain't Markup Language"의 약자로, 데이터 직렬화에 사용되는 가볍고 사람이 읽기 쉬운 형식입니다. JSON이나 XML과 비교해 간결하고 이해하기 쉬운 구문을 제공하며, 설정 파일이나 데이터 저장에 자주 활용됩니다. YAML은 들여쓰기를 통해 데이터 구조를 정의하며, 주로 다음과 같은 특징이 있습니다.

- **간결성:** 태그나 괄호 없이 간단한 텍스트 기반 형식.
- **유연성:** 다양한 데이터 타입과 구조를 지원.
- **사람이 읽기 쉬움:** 들여쓰기로 계층 구조를 표현.

3.2 주요 필드

이 작성법은 DevSecOps 환경에서 사용되는 주요 필드(apiVersion, kind, metadata, spec)와 Kubernetes 워커 노드를 추가하는 방법을 중심으로 작성되었습니다.

1) apiVersion

- 뜻: API 버전을 지정합니다. Kubernetes 리소스 종류마다 지원하는 API 버전이 다를 수 있습니다.
- 사용 목적: Kubernetes가 리소스를 처리할 때 올바른 API를 호출하도록 지시합니다.
- 예시: 여기서 apps/v1은 Deployment와 같은 애플리케이션 관련 리소스를 정의할 때 사용합니다.

```
apiVersion: apps/v1
```

2) kind

- 뜻: YAML 파일이 정의하는 리소스의 유형을 지정합니다.
- 사용 목적: Kubernetes에서 정의된 리소스 종류(Deployment, Pod 등)를 나타냅니다.
- 예시: 여기서 Deployment는 애플리케이션 배포를 정의함을 의미합니다.

```
kind: Deployment
```

3) metadata

- 뜻: 리소스에 대한 메타데이터를 정의합니다. 리소스 이름, 라벨, 주석 등을 포함합니다.
- 사용 목적: 리소스를 식별하거나 관리할 때 사용되는 정보를 제공합니다.
- 주요 필드:
 - name: 리소스의 이름

DevSecOps

- labels: 리소스에 태그를 추가하여 그룹화하거나 필터링에 사용
- 예시

```
metadata:
  name: example-deployment
  labels: app: example
```

4) spec

- 뜻: 리소스의 동작 방식과 구성 세부 사항을 정의합니다.
- 사용 목적: Pod 템플릿, 컨테이너 정의, 서비스 포트 등을 지정합니다.
- 주요 필드:
 - replicas: Pod 복제본 수
 - containers: 실행할 컨테이너 이미지와 설정
- 예시

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: nginx
          image: nginx:1.21.6
          ports:
            - containerPort: 80
```

3.3 워커 노드 추가

1) kind-config.yaml 파일

워커 노드 설정은 nodes 필드에 추가합니다.

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
  - role: worker
  labels:
    node-type: webserver
```

```
  purpose: apache-php
- role: worker
  labels:
    node-type: database
  purpose: mysql
```

- role: worker: 워커 노드로 역할을 지정
- labels: 각 워커 노드에 태그를 추가하여 특정 워크노드에 연결

2) Deployment에서 워커 노드 참조

특정 노드에서 실행되도록 nodeSelector를 사용합니다.

```
spec:
  template:
    spec:
      nodeSelector:
        node-type: webserver
```

3) 노드 상태 확인

Windows PowerShell 실행 후, 명령어를 통해 클러스터 상태 확인.

```
1. kubectl get nodes --show-labels
```

3.4 기타 필드

1) status

- 뜻: 리소스의 현재 상태를 나타냅니다.
- 사용 목적: 리소스가 배포된 후 Kubernetes가 관리하는 동적인 상태 정보를 기록합니다. 사용자가 수동으로 설정할 필요는 없습니다.
- 예시: 현재 3개의 Pod 중 2개가 정상적으로 실행되고 있음을 나타냅니다.

```
status:
  replicas: 3
  availableReplicas: 2
```

2) selector

- 뜻: 리소스를 특정 레이블을 기준으로 연결합니다.
- 사용 목적: Deployment와 같은 리소스가 Pod를 선택하거나 Service가 대상 Pod를 식별할 때 사용됩니다.
- 예시: app: example 레이블이 포함된 리소스를 선택합니다.

```
selector:
  matchLabels:
```

DevSecOps

```
app: example
```

3) template

- 뜻: 리소스가 생성할 객체(Pod 등)의 템플릿을 정의합니다.
- 사용 목적: Deployment, DaemonSet 등의 리소스에서 사용되며, 생성할 Pod의 세부 설정을 포함합니다.
- 예시: app: example 레이블을 가진 Pod를 생성하며, nginx 컨테이너를 실행합니다.

```
template:
  metadata:
    labels:
      app: example
  spec:
    containers:
      - name: nginx
        image: nginx:1.21.6
```

4) volumes

- 뜻: 컨테이너에서 사용할 볼륨을 정의합니다.
- 사용 목적: 데이터의 영속성을 보장하거나 컨테이너 간 데이터를 공유할 때 사용됩니다.
- 예시: Persistent Volume Claim(PVC)을 참조하여 데이터를 영구 저장합니다.

```
volumes:
- name: example-volume
  persistentVolumeClaim:
    claimName: example-pvc
```

5) resources

- 뜻: 컨테이너에서 사용할 CPU와 메모리 리소스를 제한하거나 요청합니다.
- 사용 목적: 클러스터의 리소스를 효율적으로 관리합니다.
- 예시:
 - limits: 컨테이너가 사용할 수 있는 최대 리소스 설정
 - requests: 컨테이너가 요청하는 최소 리소스 설정

```
resources:
  limits:
    memory: "512Mi"
    cpu: "500m"
  requests:
    memory: "256Mi"
    cpu: "250m"
```

6) env

- 뜻: 컨테이너 내에서 사용할 환경 변수를 정의합니다.
- **사용 목적:** 동적인 환경 설정을 제공하거나 비밀 값을 주입합니다.
- **예시:** 데이터베이스 호스트와 포트를 환경 변수로 정의합니다.

```
env:  
- name: DB_HOST  
  value: "example-database"  
- name: DB_PORT  
  value: "3306"
```

7) ports

- 뜻: 컨테이너가 노출하는 네트워크 포트를 정의합니다.
- **사용 목적:** 외부 네트워크와의 통신 경로를 설정합니다.
- **예시:** 컨테이너의 80번 포트를 외부로 노출합니다.

```
ports:  
- containerPort: 80
```

8) nodeSelector

- 뜻: 특정 노드에서 리소스를 실행하도록 지정합니다.
- **사용 목적:** 지정된 레이블을 기준으로 Pod가 실행될 노드를 선택합니다.
- **예시:** node-type=database 레이블이 설정된 노드에서 Pod가 실행되도록 지정합니다.

```
nodeSelector:  
  node-type: database
```

9) affinity

- 뜻: Pod 배치 전략을 정의합니다.
- **사용 목적:** 특정 노드에 Pod를 배치하거나, 특정 노드에서 실행되지 않도록 제약 조건을 설정합니다.
- **예시:** node-type=webserver 노드에만 Pod가 배치되도록 지정합니다.

```
affinity:  
  nodeAffinity:  
    requiredDuringSchedulingIgnoredDuringExecution:  
      nodeSelectorTerms:  
        - matchExpressions:  
          - key: node-type  
            operator: In  
            values:  
              - webserver
```


4. Kubernetes

4.1 Kubernetes 란?

Kubernetes(쿠버네티스)는 **컨테이너화된 애플리케이션을 쉽고 빠르게 배포 및 확장하며 관리**를 자동화해주는 오픈소스 오케스트레이션 플랫폼입니다. 애플리케이션의 안정적이고 효율적인 운영을 지원하며, 클라우드와 온프레미스 환경 모두에서 사용할 수 있습니다.

Kubernetes는 다양한 구성 요소를 통해 컨테이너화된 애플리케이션을 관리합니다. 주요 개념은 다음과 같습니다:

1. 컨테이너(Container)

- Docker와 같은 컨테이너 기술을 해 애플리케이션을 격리된 환경에서 실행합니다.
- Kubernetes는 이러한 컨테이너의 배포와 관리를 담당합니다.

2. 노드(Node)

- Kubernetes 클러스터를 구성하는 서버 단위입니다.
- **마스터 노드**는 클러스터를 관리하고, **워커 노드**는 컨테이너를 실행합니다.

3. 클러스터(Cluster)

- Kubernetes가 관리하는 시스템 단위로 여러 노드로 구성됩니다.
- 작업을 노드 간에 분배하여 효율적인 리소스 활용을 지원합니다.

4. 파드(Pod)

- Kubernetes에서 가장 작은 배포 단위로, 하나 이상의 컨테이너가 포함됩니다.
- 동일한 네트워크 및 스토리지를 공유하며 함께 작동합니다.

5. 디플로이먼트(Deployment)

- 애플리케이션 배포 및 관리를 담당하며, 원하는 상태(예: 컨테이너 수)를 정의합니다.
- Kubernetes는 이 상태를 유지하도록 자동으로 조정합니다.

6. 서비스(Service)

- 파드 간의 통신을 가능하게 하며, 외부 네트워크와의 연결을 제공합니다.

7. 이벤트 기반 관리

- 애플리케이션 상태를 지속적으로 모니터링하며, 필요 시 재배포, 재시작, 확장을 수행합니다.

따라서 우리는 Kubernetes의 주요개념을 기반으로 DevSecOps 환경을 효율적으로 관리할 계획입니다. 컨테이너를 통해 애플리케이션을 격리된 환경에서 실행하고, 노드와 클러스터를 활용해 리소스를 최적화하며 안정적인 시스템 운영을 지원합니다.

파드와 디플로이먼트를 이용해 애플리케이션을 유연하게 배포하고, 서비스를 통해 내부 및 외부 네트워크 통신을 원활히 처리합니다. 또한, Kubernetes의 이벤트 기반 관리 기능을 활용해 애플리케이션 상태를 지속적으로 모니터링하고, 필요 시 자동으로 확장하거나 복구하는 방안을 적용할 것입니다.

4.2 DevSecOps에서 Kubernetes의 역할

우리가 Kubernetes를 사용하는 이유는 DevSecOps 환경에서 애플리케이션을 효율적이고 안정적으로 관리하기 위함입니다. Kubernetes는 리소스를 최적화하여 비용을 절감하고, 빠른 배포와 자동 복구를 통해 운영 안정성을 제공합니다.

또한, 클라우드와 온프레미스 환경을 모두 지원하며, 애플리케이션 확장과 멀티 클라우드 통합이 가능해 유연성과 확장성이 뛰어납니다. CI/CD 도구와의 손쉬운 통합으로 보안을 강화할 수 있어 우리의 프로젝트에 필수적인 플랫폼입니다.

Kubernetes를 사용함으로써 DevSecOps 환경에서 생산성과 안정성을 동시에 향상시킬 수 있습니다. 자동화된 배포와 리소스 최적화를 통해 개발 및 운영 시간을 단축하고 비용을 절감할 수 있습니다. 장애 발생 시 자동 복구와 중단 없는 업데이트를 제공하여 시스템 가용성과 사용자 경험을 크게 개선합니다.

또한, 멀티 클라우드 및 하이브리드 환경을 지원해 유연한 인프라 구성이 가능하며, 애플리케이션의 확장성을 확보할 수 있습니다. Jenkins, SonarQube, Wazuh 등 다양한 DevSecOps 도구와의 통합으로 파이프라인 전반의 보안과 효율성을 강화할 수 있어, 전체적인 운영 관리가 간소화되고 보안 위험이 최소화됩니다. 이를 통해 안정적이고 신뢰할 수 있는 애플리케이션 서비스를 제공할 수 있습니다.

4.3 kind

우리는 kind(Kubernetes IN Docker)를 사용하여 클러스터를 구성하고 노드를 설치 및 관리합니다. kind는 Docker 컨테이너 내에서 Kubernetes 클러스터를 실행할 수 있어, 간편하고 빠르게 개발 및 테스트 환경을 구축할 수 있다는 장점이 있습니다.

1) Kind 설치 및 환경설정

Window PowerShell 관리자 권한 실행

```
# KIND 다운로드
```

```
1. Invoke-WebRequest -Uri "https://kind.sigs.k8s.io/dl/v0.20.0/kind-windows-amd64" -OutFile "$env:USERPROFILE\kind.exe"
```

```
# bin 디렉토리 생성 및 이동
2. New-Item -Path "$env:USERPROFILE\bin" -ItemType Directory -Force
3. Move-Item -Path "$env:USERPROFILE\kind.exe" -Destination
   "$env:USERPROFILE\bin"

# 환경변수 설정
4. $env:Path += ";$env:USERPROFILE\bin"
5. [Environment]::SetEnvironmentVariable('Path', $env:Path, 'User')

# 설치 확인
6. kind --version
```

2) kind-config.yaml 생성

kind-config.yaml 파일은 쿠버네티스에서 어떠한 클러스터를 생성할지 설정하는 파일이다. DevSecOps폴더에 k8s-yaml폴더 생성 및 kind-config.yaml 파일 생성 후, 아래 코드 작성

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
# 컨트롤 플레인
- role: control-plane
  extraPortMappings:
  - containerPort: 30080 #DevSecOps Web
    hostPort: 30080
    listenAddress: "0.0.0.0"
    protocol: TCP

# 워커 노드 1: 테스트 웹서버
- role: worker
  labels:
    node-type: webserver
    purpose: apache-php
```

3) 클러스터 생성 및 확인

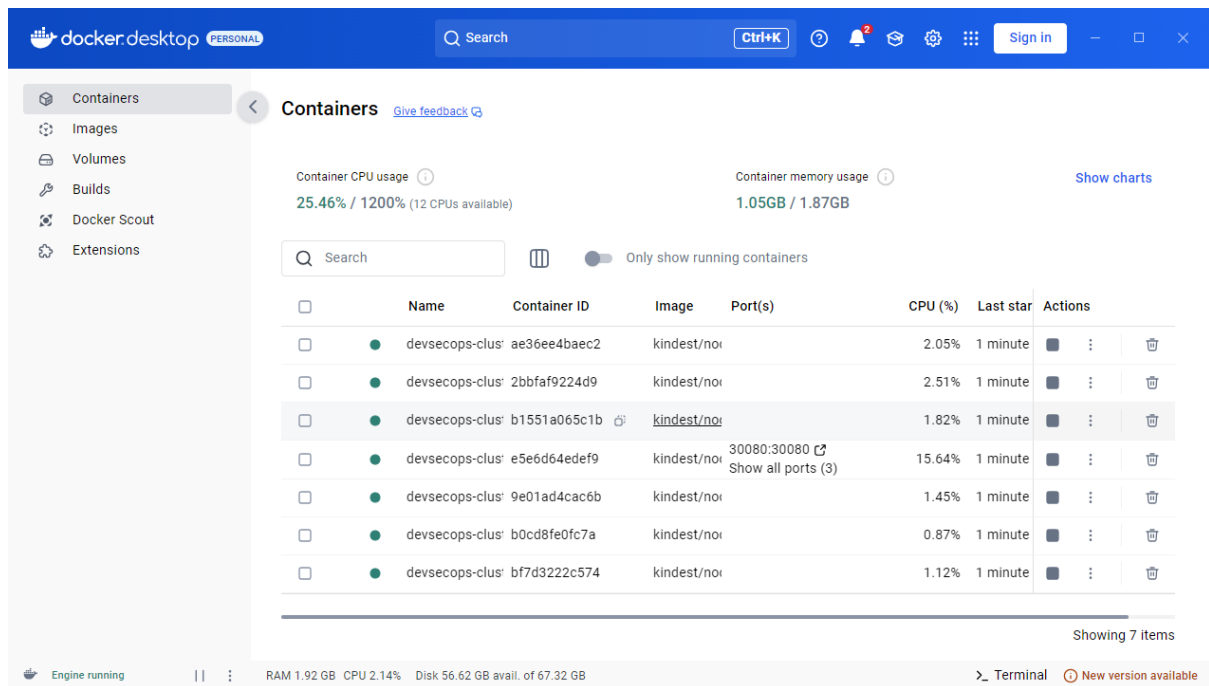
Window PowerShell 관리자 권한 실행

※ Docker Desktop 실행 필요

```
# DevSecOps\k8s폴더로 이동
#(컴퓨터 설정에 따라 Users폴더 안 사용자 폴더이름 변경 필요)
1. cd C:\Users\Administrator\Desktop\DevSecOps\k8s\
```

```
# kind 명령어로 새 클러스터 생성
2. kind create cluster --config kind-config.yaml --name devsecops-cluster

# 클러스터 생성 확인
3. Kind get clusters
```



Docker Desktop 화면

kind-config.yaml 파일에서 설정한 각 노드들이 컨테이너로 실행되고 있는 것을 볼 수 있다.

4.4 오류 발생 또는 충돌 시

1) 클러스터 삭제 및 Docker Desktop 재실행

Window PowerShell 관리자 권한 실행

```
# 모든 클러스터 삭제  
1. kind delete clusters --all
```

2) Docker Desktop 재실행 후, kindest/node 이미지 삭제

1. Docker Desktop 해당 이미지의 Delete 버튼(휴지통) 클릭

<input type="checkbox"/>	<input type="radio"/>	kindest/node	<none>	3966ac761ae0	2 years ago	1.37 GB	▶	:	🗑
--------------------------	-----------------------	--------------	--------	--------------	-------------	---------	---	---	---

2. Docker Desktop 터미널에서 명령어 입력

```
# Docker Image 삭제  
1. docker rmi <Image ID>
```



5. Jenkins

5.1 Jenkins 란?

Jenkins는 CI/CD를 위한 오픈소스로 도구로, 다양한 플러그인을 지원하여 코드 변경 사항이 발생할 때마다 **자동으로 빌드, 테스트, 배포 과정을 수행할 수 있는 도구**입니다.

이를 통해 개발팀은 소프트웨어 개발 및 배포 과정을 효율적으로 관리할 수 있으며, 반복 작업을 자동화하여 개발 생산성을 크게 향상시킬 수 있습니다.

CI/CD란 무엇인가?

- **CI (지속적 통합, Continuous Integration)**: 개발자들이 각자 작업한 코드 변경 사항을 중앙 저장소에 정기적으로 통합하고, 이를 자동화된 빌드와 테스트를 통해 검증하는 프로세스입니다. 이를 통해 코드 품질을 유지하고, 통합 시 발생할 수 있는 문제를 조기에 발견할 수 있습니다.
- **CD (지속적 배포, Continuous Deployment)**: CI 과정에서 통합된 코드가 자동으로 프로덕션 환경에 배포되기까지의 전 과정을 자동화하는 것을 의미합니다. 이로 인해 새로운 기능과 수정 사항이 사용자에게 빠르고 안정적으로 전달될 수 있습니다.

Jenkins는 이러한 CI/CD 프로세스를 손쉽게 구축하고 관리할 수 있는 도구로, 소프트웨어 개발과 배포의 효율성을 크게 향상시킵니다.

5.2 DevSecOps에서의 Jenkins의 역할

Jenkins는 DevSecOps 환경에서 **자동화의 핵심 도구**로 활용됩니다.

- 코드 변경 시 **자동으로 빌드하고 테스트하여 오류를 빠르게 발견**할 수 있습니다.
- 수천 개의 플러그인을 통해 다양한 기능을 추가하여 복잡한 요구 사항을 충족시킬 수 있습니다.
- **배포 파이프라인을 코드로 정의하고 관리**할 수 있어, 재사용성과 유지보수성을 극대화합니다.
- **보안 검증, 취약점 스캔**과 같은 DevSecOps 요구사항도 Jenkins 파이프라인에 통합하여 자동화할 수 있습니다.

이러한 기능을 통해 Jenkins는 DevSecOps의 핵심 원칙인 **자동화, 통합, 보안**을 구현하는 데 중요한 역할을 수행합니다.

5.3 Jenkins worker node 생성

1) *jenkins-deployment.yaml* 생성

jenkins-deployment.yaml 파일은 jenkins worke node 설정 파일이다.

k8s-yaml 폴더에 jenkins-deployment.yaml 파일 생성 후, 아래 코드 작성

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      nodeSelector:
        purpose: jenkins
      securityContext:
        runAsUser: 0
        fsGroup: 0
      initContainers:
        - name: init-jenkins-home
          image: busybox
          command: ['sh', '-c', 'mkdir -p /var/jenkins_home && chown -R 1000:1000 /var/jenkins_home']
          volumeMounts:
            - name: jenkins-home
              mountPath: /var/jenkins_home
      containers:
        - name: jenkins
          image: jenkins/jenkins:lts
          ports:
            - containerPort: 8080
              name: http
            - containerPort: 50000
              name: jnlp
          volumeMounts:
            - name: jenkins-home
```

```

    mountPath: /var/jenkins_home
  env:
  - name: JAVA_OPTS
    value: "-Xmx2048m"
  resources:
    limits:
      memory: "2Gi"
      cpu: "1000m"
    requests:
      memory: "1Gi"
      cpu: "500m"
  startupProbe:
    httpGet:
      path: /login
      port: 8080
    failureThreshold: 30
    periodSeconds: 10
  readinessProbe:
    httpGet:
      path: /login
      port: 8080
    initialDelaySeconds: 60
    timeoutSeconds: 5
    periodSeconds: 10
  livenessProbe:
    httpGet:
      path: /login
      port: 8080
    initialDelaySeconds: 60
    timeoutSeconds: 5
    periodSeconds: 10
  volumes:
  - name: jenkins-home
    persistentVolumeClaim:
      claimName: jenkins-pvc

```

2) *kind-config.yaml* 수정

Jenkins 클러스터를 생성하기 위해 kind-config.yaml 파일 수정, 아래 코드 추가

```

kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  # 컨트롤 플레인
  # Jenkins 웹 인터페이스 포트

```

DevSecOps


```

# - CI/CD 파이프라인 관리 웹 UI
# - 빌드 및 배포 모니터링
# - 호스트 접근: http://localhost:8080
- containerPort: 30800
  hostPort: 8080
  listenAddress: "0.0.0.0"
  protocol: TCP

# Jenkins JNLP 에이전트 포트
# - Jenkins 워커 노드 연결
# - 분산 빌드 에이전트 통신
# - 호스트 접근: localhost:50000
- containerPort: 30850
  hostPort: 50000
  listenAddress: "0.0.0.0"
  protocol: TCP

# 워커 노드 2: Jenkins
- role: worker
  labels:
    node-type: auto
    purpose: jenkins

```

3) Jenkins-pv.yaml 생성

PV(Persistent Volume)와 PVC(Claim)은 쿠버네티스에서 영구 스토리지를 관리하기 위한 핵심 리소스이다.

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: jenkins-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/jenkins-volume"

```

```

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi

```

4) Jenkins-service.yaml 생성

jenkins worker node의 연동을 위한 파일

```

apiVersion: v1
kind: Service
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30800
      name: http
    - port: 50000
      targetPort: 50000
      nodePort: 30850
      name: jnlp
  selector:
    app: Jenkins

```

5.4 Jenkins worker node 실행

1) kind 클러스터 재생성

Window PowerShell 관리자 권한 실행

```

# DevSecOps폴더로 이동
1. cd C:\Users\Administrator\Desktop\DevSecOps

```

DevSecOps

```
# kind 클러스터 생성
2. kind create cluster --name devsecops-cluster --config k8s-yaml/kind-config.yaml

# 노드 상태 확인
3. kubectl get nodes --show-labels
```

디플로이먼트와 서비스 배포

```
# deployment, service yaml 파일 배포
1. kubectl apply -f k8s-yaml/jenkins-deployment.yaml
2. kubectl apply -f k8s-yaml/jenkins-services.yaml

# 배포 상태 확인
3. kubectl get pods -o wide
4. kubectl get services
```

2) Jenkins 관리자 암호

Jenkins worker node로부터 초기 관리자 암호 추출

```
# Jenkins Administrator Password에 필요한 패스워드값 출력
1. kubectl exec -it $(kubectl get pods -l app=jenkins -o jsonpath='{.items[0].metadata.name}') -- cat /var/jenkins_home/secrets/initialAdminPassword
```

3) 웹 페이지 접속

browser 주소창: localhost:8080 or http://Windows Host IP:8080

4) localtunnel을 이용한 도메인 연결

Window cmd 관리자 권한 실행

```
1. lt --subdomain jenkins1234 --port 8080
```

6. SonarQube

6.1 SonarQube yaml 파일 생성

DevSecOps 폴더에 SonarQube 폴더 생성 후 yaml 설정 파일 생성

Window PowerShell 관리자 권한 실행

1) postgres-deployment.yaml 생성

2) sonarqube-deployment.yaml 생성

3) services.yaml 생성

6.2 SonarQube node 실행

1) 디플로이먼트와 서비스 배포

Window PowerShell 관리자 권한 실행

```
# SonarQube 폴더로 이동
1. cd C:\Users\Administrator\Desktop\DevSecOps\SonarQube

# deployment, service yaml 파일 배포
2. kubectl apply -f k8s\postgres-deployment.yaml
3. kubectl apply -f k8s\sonarqube-deployment.yaml
4. kubectl apply -f k8s\services.yaml

# 배포 상태 확인
5. kubectl get pods -o wide
6. kubectl get services
```

2) 웹 페이지 접속

```
browser 주소창: localhost:30900
```

3) localtunnel을 이용한 도메인 연결

Window cmd 관리자 권한 실행

```
1. lt --subdomain sonarqube1234 --port 30900
```

DevSecOps

6.3 Github Web hook 설정

Github Repository -> Settings -> Code and automation -> Webhooks -> Add webhook 클릭

Payload URL: <https://jenkins1234.loca.lt/github-webhook/> 입력

Current type: application/json 설정

Webhooks / Add webhook


We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type *

Secret

SSL verification

 By default, we verify SSL certificates when delivering payloads.

☒ **Enable SSL verification** ☐ **Disable (not recommended)**

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

6.4 Github Access Token 생성

Github Profile -> Settings -> Developer settings -> Personal access tokens -> Tokens(classic)
-> Generate new token (classic) 클릭

Note: jenkins_token 입력

Select scopes: repo, workflow, write:packges, admin:repo_hook 체크

Select scopes
Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users



8. Wazuh

8.1 Wazuh 란?

Wazuh는 오픈소스 기반의 보안 플랫폼으로, **호스트 기반 침입 탐지 시스템(HIDS)** 역할을 수행하며, **위협 탐지, 규정 준수, 보안 모니터링**을 제공합니다. Wazuh는 에이전트와 서버, 대시보드로 구성되어 있으며, 클라우드 및 온프레미스 환경에서 모두 효율적으로 작동합니다.

8.2 Wazuh의 역할

DevSecOps 환경에서 Wazuh는 다음과 같은 주요 역할을 수행합니다:

- **보안 이벤트 모니터링:** 호스트와 컨테이너에서 발생하는 이벤트를 실시간으로 수집하고 분석하며, 로그, 시스템 호출, 네트워크 트래픽을 기반으로 위협을 탐지합니다.
- **취약점 관리:** 에이전트가 설치된 서버와 애플리케이션의 취약점을 스캔하며, 알려진 CVE(Common Vulnerabilities and Exposures)를 감지하고 경고를 제공합니다.
- **규정 준수 보고:** ISO 27001, GDPR, HIPAA 등 다양한 규정을 준수하는지 확인합니다. 감사(Audit) 데이터를 수집하고 상세 보고서를 생성합니다.
- **파일 무결성 모니터링(FIM):** 중요한 파일 및 디렉토리의 변경 사항을 추적합니다. 예상치 못한 변경 감지 시 경고 알림을 보냅니다.
- **컨테이너 및 클라우드 보안:** Docker 및 Kubernetes와의 통합을 통해 컨테이너의 보안 상태를 점검합니다. 클라우드 환경(AWS, GCP, Azure)의 보안 설정을 검사합니다.
- **중앙 집중식 관리:** 모든 에이전트를 중앙에서 관리하며, 대시보드를 통해 전체 시스템 상태를 모니터링합니다.

8.3 Wazuh 사용계획 및 기대되는 이점

DevSecOps 환경에서 Wazuh 활용 방식은 다음과 같습니다.

1. 설치 및 배포

- Kubernetes 클러스터 내에 Wazuh 서버 및 에이전트를 설치합니다.
- Docker를 활용하여 Wazuh 컨테이너를 배포하고 구성합니다.

2. 통합 및 모니터링

- Jenkins, SonarQube, 애플리케이션 서버와 통합합니다.
- 시스템 및 애플리케이션 로그를 Wazuh로 전송하여 분석합니다.

3. 경고 및 보고

- 보안 위협 감지 시 경고를 설정합니다.
- DevSecOps 파이프라인 단계별 보안 상태 리포트를 생성합니다.

4. 실시간 대시보드 활용

- Kibana를 통해 실시간 보안 데이터 시각화합니다.
- 취약점 및 규정 준수 상태 확인합니다.

Wazuh는 다양한 플랫폼을 지원하며, Linux, Windows, macOS, Docker와 같은 환경에서 효과적으로 동작합니다. 수천 개의 노드에서도 원활히 작동할 수 있는 확장성을 제공하며, Slack, 이메일, 웹훅 등을 통한 알림 시스템을 갖추고 있습니다. 또한, ELK(Elasticsearch, Logstash, Kibana) 스택과의 통합을 통해 실시간 보안 데이터 시각화를 지원하며, 오픈소스 기반으로 무료로 사용이 가능합니다.

Wazuh를 사용함으로써 DevSecOps 환경에서 보안을 크게 강화할 수 있습니다. CI/CD 파이프라인 전체의 보안 상태를 실시간으로 모니터링하고, 취약점과 위협을 사전에 탐지 및 제거하여 보안 위험을 최소화합니다. 또한, ISO 27001, GDPR, HIPAA 등 다양한 규정 준수를 자동화된 방식으로 지원함으로써 감사 및 규정 준수 보고가 간소화됩니다. 중앙 집중식 관리 대시보드를 통해 모든 에이전트를 효율적으로 관리할 수 있어 운영 효율성도 향상됩니다.

Wazuh는 이러한 기능을 통해 DevSecOps 파이프라인 전반에서 신뢰성과 보안성을 확보하는 강력한 도구로 자리 잡습니다.

9. Web 구축

9.1. WebPage 소스코드

방법 1) 직접 생성

1. DevSecOps폴더에 Web폴더 생성 후, 웹 소스코드 복사

Practice Web Source code: [DevSecOps Team Web Source code](#)

2. Github의 특수문자를 허용하기 위한 설정

Window PowerShell 관리자 권한 실행

```
# DevSecOps\Web폴더로 이동
1. cd C:\Users\Administrator\Desktop\DevSecOps\Web
```

```
# Github 특수문자 허용
2. git config core.protectNTFS false
```

3. DevSecOps폴더에 k8s-yaml 폴더 생성

방법 2) Window PowerShell 사용

Window PowerShell 관리자 권한 실행

```
# DevSecOps폴더 이동
1. cd C:\Users\Administrator\Desktop\DevSecOps

# 소스코드 클론
2. git clone https://github.com/GH6679/web_wargamer.git
3. cd web_wargamer

# 깃허브 안에 있는 특수문자들 허용으로 설정
4. git config core.protectNTFS false
5. cd ..

# k8s-yaml 디렉토리 생성
6. mkdir k8s-yaml
```

9.2. Web worker node 생성

1) web-deployment.yaml 생성

web-deployment.yaml 파일은 웹 worker node 설정 파일이다.

k8s-yaml 폴더에 web-deployment.yaml 파일 생성 후, 아래 코드 작성

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wargame-web
spec:
  selector:
    matchLabels:
      app: wargame-web
  template:
    metadata:
      labels:
        app: wargame-web
```

DevSecOps

```

    node-type: webserver
spec:
  nodeSelector:
    node-type: webserver
  containers:
  - name: wargame-web
    image: wargame-web:latest
    imagePullPolicy: Never
    ports:
    - containerPort: 80
    env:
    - name: DB_HOST
      value: "wargame-db-service"
    - name: DB_PORT
      value: "3307"
    - name: DB_DATABASE
      value: "LED_WG"
    - name: DB_USERNAME
      value: "root"
    - name: DB_PASSWORD
      value: "1234"
    volumeMounts:
    - name: web-storage
      mountPath: /var/www/html
  resources:
    limits:
      memory: "512Mi"
      cpu: "500m"
    requests:
      memory: "256Mi"
      cpu: "250m"
  volumes:
  - name: web-storage
    persistentVolumeClaim:
      claimName: web-pvc

```

2) db-deployment.yaml 생성

db-deployment.yaml 파일은 웹 데이터베이스 worker node 설정 파일이다.

k8s-yaml 폴더에 db-deployment.yaml 파일 생성 후, 아래 코드 작성

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: wargame-db

```

```

spec:
  selector:
    matchLabels:
      app: wargame-db
  template:
    metadata:
      labels:
        app: wargame-db
        node-type: web-db
    spec:
      nodeSelector:
        node-type: web-db
      containers:
        - name: mariadb
          image: mariadb:latest
          resources:
            limits:
              memory: "512Mi"
              cpu: "500m"
          ports:
            - containerPort: 3306
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "1234"
            - name: MYSQL_DATABASE
              value: "LED_WG"
            - name: MYSQL_ALLOW_EMPTY_PASSWORD
              value: "no"
            - name: MYSQL_ROOT_HOST
              value: "%"
            - name: MARIADB_MYSQL_LOCALHOST_USER
              value: "1"
            - name: MARIADB_MYSQL_LOCALHOST_GRANTS
              value: "ALL"
          volumeMounts:
            - name: mariadb-storage
              mountPath: /var/lib/mysql
            - name: mariadb-backup
              mountPath: /backup
      volumes:
        - name: mariadb-storage
          persistentVolumeClaim:
            claimName: mariadb-pvc
        - name: mariadb-backup
          persistentVolumeClaim:

```

DevSecOps

```
claimName: mariadb-backup-pvc
```

3) db-init-configmap.yaml 생성

db-init-configmap.yaml 파일은 웹 데이터베이스 worker node의 초기화 설정 파일이다.

k8s-yaml 폴더에 db-init-configmap.yaml 파일 생성 후, 아래 코드 작성

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mariadb-init
data:
  init.sql: |
    USE LED_WG;
    CREATE TABLE IF NOT EXISTS users (
      u_id INT NOT NULL AUTO_INCREMENT,
      nickname VARCHAR(50) NOT NULL,
      username VARCHAR(50) NOT NULL,
      password VARCHAR(50) NOT NULL,
      email VARCHAR(50) NOT NULL,
      user_role VARCHAR(50) NOT NULL,
      PRIMARY KEY (u_id)
    );
    CREATE TABLE IF NOT EXISTS challenges_data (
      c_id INT NOT NULL AUTO_INCREMENT,
      c_title VARCHAR(50) NOT NULL,
      c_ssh VARCHAR(50),
      c_web TINYINT(1) NOT NULL,
      c_link VARCHAR(50),
      c_hint TEXT,
      c_point int,
      c_difficulty VARCHAR(50),
      c_key VARCHAR(50),
      c_text TEXT,
      c_solves int,
      PRIMARY KEY (c_id)
    );
    CREATE TABLE IF NOT EXISTS user_data (
      d_id INT NOT NULL AUTO_INCREMENT,
      d_uid INT,
      d_cid INT,
      d_time datetime,
      PRIMARY KEY (d_id),
      FOREIGN KEY (d_uid) REFERENCES users(u_id) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
FOREIGN KEY (d_cid) REFERENCES challenges_data(c_id) ON DELETE CASCADE
ON UPDATE CASCADE);
```

4) service.yaml 생성

service.yaml 파일은 웹과 데이터베이스의 연동을 위한 설정 파일이다.

k8s-yaml 폴더에 service.yaml 파일 생성 후, 아래 코드 작성

```
apiVersion: v1
kind: Service
metadata:
  name: wargame-web-service
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
  selector:
    app: wargame-web
---
apiVersion: v1
kind: Service
metadata:
  name: wargame-db-service
spec:
  type: ClusterIP
  ports:
    - port: 3306
      targetPort: 3306
      protocol: TCP
  selector:
    app: wargame-db
```

9.3 Web worker node 실행

1) 클러스터 삭제 및 kind 클러스터 생성

Window PowerShell 관리자 권한 실행

```
# DevSecOps폴더로 이동
1. cd C:\Users\Administrator\Desktop\DevSecOps

# 클러스터 삭제 명령어
2. kind delete cluster --name devsecops-cluster
```

DevSecOps

```
3. kind delete clusters --all

# kind 클러스터 생성
4. kind create cluster --name devsecops-cluster --config k8s-yaml/kind-config.yaml

# 노드 상태 확인
5. kubectl get nodes --show-labels
```

Web 이미지 빌드 및 kind 클러스터로 로드

```
# 웹 애플리케이션 이미지 빌드
1. docker build -t wargame-web:latest .

# 빌드된 이미지를 kind 클러스터로 로드
2. kind load docker-image wargame-web:latest --name devsecops-cluster
```

디플로이먼트와 서비스 배포

```
# deployment, service yaml 파일 배포
1. kubectl apply -f k8s-yaml/db-deployment.yaml
2. kubectl apply -f k8s-yaml/web-deployment.yaml
3. kubectl apply -f k8s-yaml/services.yaml

# 배포 상태 확인
4. kubectl get pods -o wide
5. kubectl get services
```

2) 웹 페이지 접속

```
browser 주소창: localhost:30080 or http://Windows Host IP:30080
```

