

# 프로젝트 최종 보고서

개발자 특화 AI 취업 도우미 서비스

과 목 : 클라우드시스템 (001)

제출일 : 2025. 12. 13.

팀 명 : 11조

팀 원 : 김정은(2111844)  
정민정(2115813)

# 홍서현(2211015)

## 목차

|   |           |
|---|-----------|
| <b>1. 프로젝트 개요.....</b>                    | <b>3</b>  |
| 1.1. 프로젝트 선정 배경.....                      | 3         |
| 1.2. 서비스 주요 기능 요약.....                    | 3         |
| 1.3. 기대 효과.....                           | 7         |
| <b>2. 팀원 간 역할 분담.....</b>                 | <b>8</b>  |
| <b>3. 동작 시나리오.....</b>                    | <b>8</b>  |
| 3.1. 무중단 서비스 배포(Rolling Update).....      | 8         |
| 3.2. CronJob 기반의 자동화된 뉴스 데이터 수집.....      | 11        |
| 3.3. 트래픽 급증에 따른 HPA 오토스케일링 및 부하 분산.....   | 12        |
| 3.4. PVC 기반의 데이터 영속성(Persistence) 보장..... | 14        |
| <b>4. 시스템 설계.....</b>                     | <b>18</b> |
| 4.1. Docker.....                          | 18        |
| 4.2. Kubernetes.....                      | 21        |
| <b>5. 프로젝트 활용 방안.....</b>                 | <b>30</b> |
| 5.1. 개발자 취업 과정에서의 활용 방안.....              | 30        |
| 5.2. 향후 기능 확장성.....                       | 32        |
| 5.3. 다양한 사용자군에 대한 활용 가능성.....             | 32        |
| <b>6. 결론 및 향후 계획.....</b>                 | <b>34</b> |
| 6.1. 프로젝트 전체 요약.....                      | 34        |

|                         |    |
|-------------------------|----|
| 6.2. 향후 시스템 확장 가능성..... | 34 |
| 6.3. 한계점 및 개선 방향.....   | 34 |
| 7. 참고자료.....            | 35 |

# 1. 프로젝트 개요

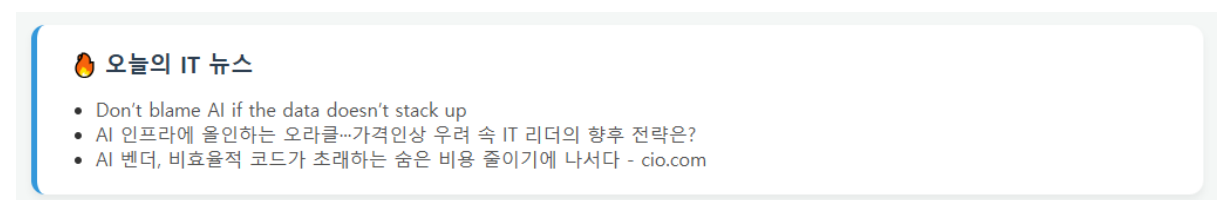
## 1.1. 프로젝트 선정 배경

최근 개발자 취업 및 이직 시장에서는 단순한 코딩 능력을 넘어, 이력서·포트폴리오·GitHub README 등 문서를 통해 자신의 기술 역량과 경험을 효과적으로 표현하는 능력이 중요한 평가 요소로 자리 잡고 있다. 그러나 많은 학생 및 현직 개발자들은 문서 작성 과정에서 표현력 부족, 구조화에 대한 막막함, 영문 작성에 대한 언어 장벽 등으로 어려움을 겪으며 이러한 부분에 많은 시간을 소비하고 있다.

이에 본 프로젝트는 ChatGPT API를 활용하여 개발자에게 특화된 문서 작성 지원 서비스를 구축하고자 하였으며, 이를 통해 이력서 번역, README 초안 작성 등 반복적인 문서 작업을 자동화함으로써 사용자가 조금 더 핵심적인 기술 개발과 면접 준비에 집중할 수 있도록 도움을 준다. 또한 AI 기반 지원을 통해 언어 및 문서 작성 역량 부분에서의 격차를 완화하고, 누구나 일정 수준 이상의 취업 관련 문서를 생성할 수 있도록 함으로써 개발자 취업 시장에 대한 기회 접근성을 높이는 것을 목표로 한다.

## 1.2. 서비스 주요 기능 요약

### 1.2.1. 최신 IT 뉴스 제공



크롤링을 통해 수집된 최신 IT 기사 정보를 제공하여 사용자가 IT 관련 기술 트렌드를 지속적으로 파악할 수 있도록 한다.



오늘의 IT 뉴스 탭에서 기사 제목을 클릭하면 기사 원문 링크로 이동하여 본문을 확인할 수 있다.

## 1.2.2. 개발자 특화 AI 챗봇



기본 챗봇은 개발 관련 질문, 커리어 고민, 기술 스택 선택 등에 대해 개발자 역할에 최적화된 프롬프트를 기반으로 맞춤형 답변을 제공한다.

### **1.2.3. README 생성**

localhost:30081

Dev-Helper

서비스 소개 개발자 커뮤니티 로그인

내 보관함

새로운 채팅

11월 20일 면접 연습

11월 15일 이력서 작성

프로젝트 아이디어 메모

김정은 님  
프론트엔드 개발자

오늘의 IT 뉴스

- 심평원, AI-빅데이터 시대 어떻게 본격 대응하나? - 메디칼업저버
- "소름"...맥도날드 크리스마스 'AI 광고' 삭제, 도대체 어땠길래 - 한겨레
- [초점] '국산 MSP '1조 달러' 미국 클라우드 시장 공략 본격화 - 아이티데일리

README 봇: 프로젝트 개요를 입력해주세요. README 작성을 도와드립니다.

나: 프로젝트 이름: News Scraper 설명: 한국과 해외 뉴스 사이트에서 주요 뉴스 기사를 크롤링하고, Redis에 저장하여 API로 제공하는 서비스 기능: 1. 주요 뉴스 사이트에서 일일 Top3 뉴스 크롤링 2. 뉴스 데이터를 Redis에 캐싱 3. REST API로 뉴스 데이터 조회 가능 사용 기술: - Python 3.11 - FastAPI - Redis - BeautifulSoup - Requests 설치 방법: 1. git clone https://github.com/example/news-scraper.git 2. pip install -r requirements.txt 3. uvicorn mainapp --reload 사용 예시: GET /news/top3 Response: [ {"title": "기사 제목 1", "link": "https://..."}, {"title": "기사 제목 2", "link": "https://..."}, {"title": "기사 제목 3", "link": "https://..."} ] 라이선스: MIT

... AI 멘토가 응답을 준비 중입니다.

내용을 입력하세요...

전송 중...

기본 챗봇 README 생성 이력서 생성 이력서 번역

localhost:30081

Dev-Helper

서비스 소개 개발자 커뮤니티 로그인

내 보관함

새로운 채팅

11월 20일 면접 연습

11월 15일 이력서 작성

프로젝트 아이디어 메모

김정은 님  
프론트엔드 개발자

오늘의 IT 뉴스

- 심평원, AI-빅데이터 시대 어떻게 본격 대응하나? - 메디칼업저버
- "소름"...맥도날드 크리스마스 'AI 광고' 삭제, 도대체 어땠길래 - 한겨레
- [초점] '국산 MSP '1조 달러' 미국 클라우드 시장 공략 본격화 - 아이티데일리

README 봇:

```

'''markdown
# News Scraper

## 개요
News Scraper는 한국과 해외 뉴스 사이트에서 주요 뉴스 기사를 크롤링하고, Redis에 저장하여 API로 제공하는 서비스입니다.

## 기능
1. 주요 뉴스 사이트에서 일일 Top3 뉴스 크롤링
2. 뉴스 데이터를 Redis에 캐싱
3. REST API로 뉴스 데이터 조회 가능

## 기술 스택
- Python 3.11
- FastAPI
- Redis
- BeautifulSoup

```

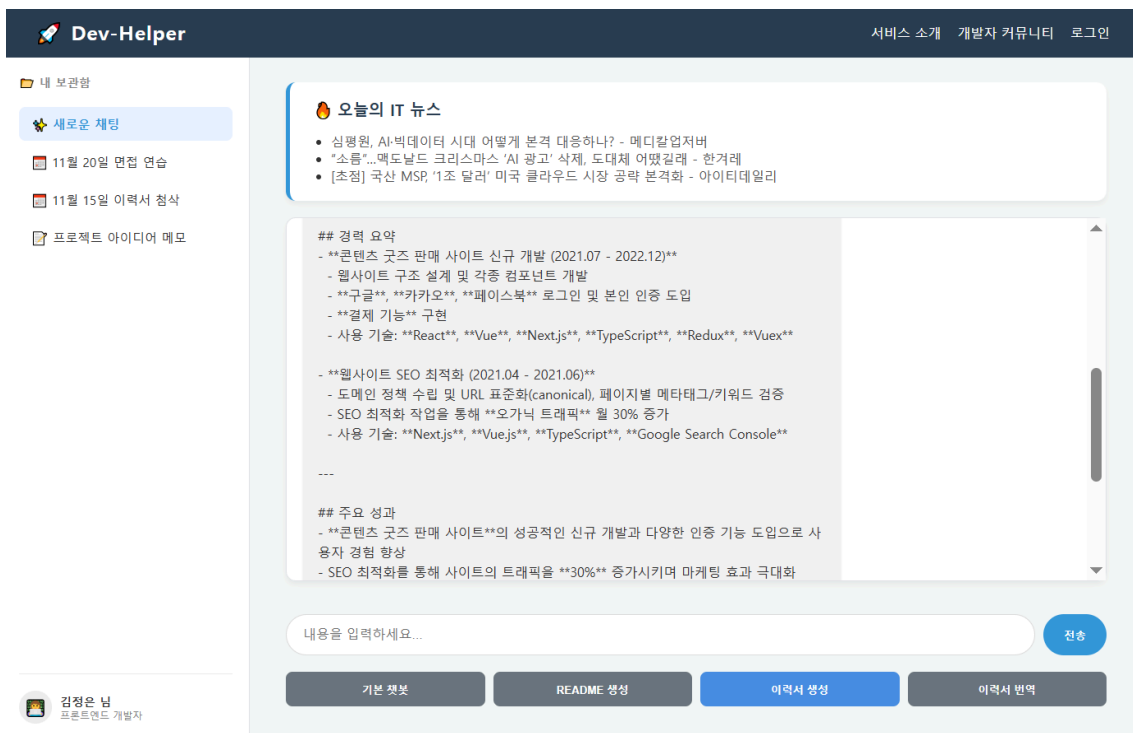
내용을 입력하세요...

전송

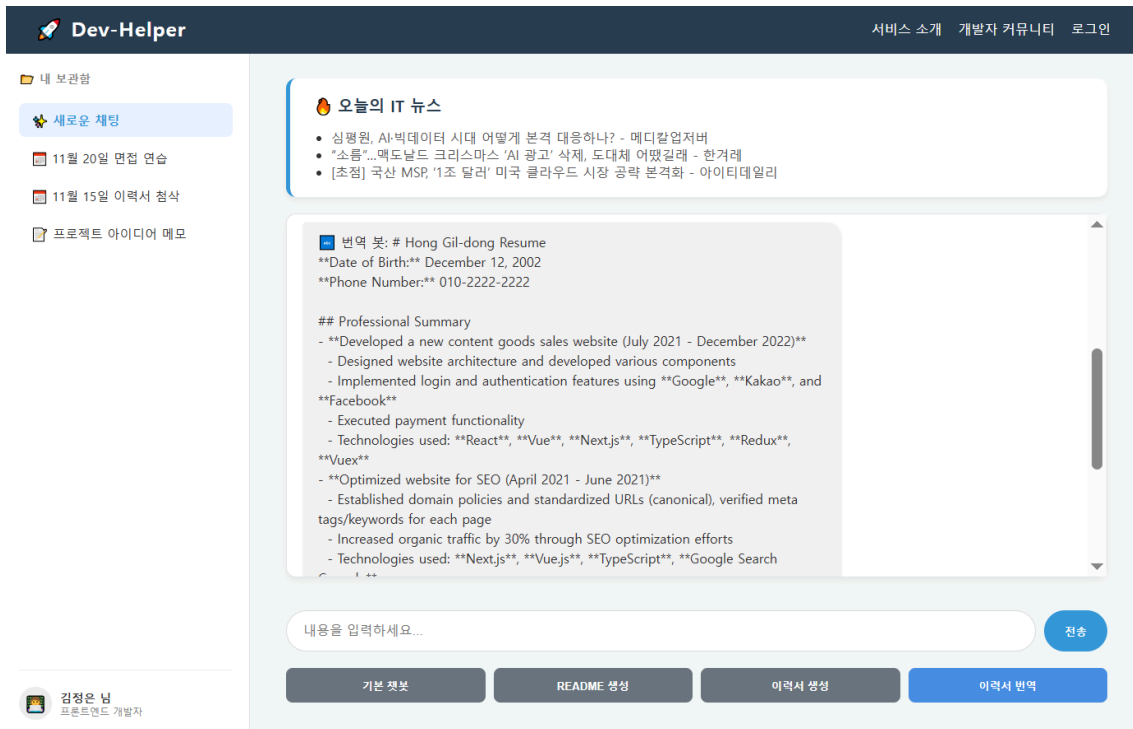
기본 챗봇 README 생성 이력서 생성 이력서 번역

프로젝트 개요, 사용 기술, 기여 내용 등을 입력하면 표준화된 README를 자동 생성하여 GitHub 포트폴리오로 활용할 수 있도록 지원한다.

#### 1.2.4. 이력서 생성



### 1.2.5. 이력서 영문 번역



한글 이력서를 IT 업계에서 자연스럽게 활용 가능한 영문 이력서로 변환한다. Resume 작성을 도와 글로벌 취업 준비를 지원한다.

## 1.3. 기대 효과

### 1.3.1. 취업 준비 시간 절감

취업 과정에서 반복적으로 발생하는 이력서, README, 직무 분석 리포트 작성 등의 비효율적 작업 시간을 단축시킨다. 이를 통해 사용자는 더 이상 문장 표현에 시간을 쏟지 않고, 본질적인 역량 개발과 면접 준비 등에 집중할 수 있다.

### 1.3.2. 문서 품질 및 경쟁력 향상

AI가 제공하는 정돈되고 전문적인 내용을 바탕으로, 개발자는 자신의 기술력과 경험을 보다 명확하고 체계적으로 표현하며 이를 통해 개인의 경쟁력 강화 및 포트폴리오 완성도 향상을 기대할 수 있다.

### 1.3.3. 지속적인 기술 트렌드 파악



IT 뉴스 기사 제공 기능을 통해, 사용자는 별도의 검색 없이도 최신 개발 트렌드·채용 키워드·기술 스택 흐름을 빠르게 파악할 수 있다. 이는 변화가 빠른 IT 시장 속에서 개발자가 지속적으로 트렌드를 따라가고, 그에 맞춰 성장할 수 있는 기반을 제공한다.

## 2. 팀원 간 역할 분담

| 팀원 이름 | 역할   |
|-------|--|
| 김정은   | Frontend(UI) 구현, Kubernetes 서비스 배포                   |
| 정민정   | OpenAI API 연동 및 프롬프트 설계, Kubernetes 서비스 배포           |
| 홍서현   | Backend API 개발, IT 뉴스 Crawling 구현, Kubernetes 서비스 배포 |

## 3. 동작 시나리오

### 3.1. 무중단 서비스 배포(Rolling Update)

#### 3.1.1. 개요

서비스 운영 중 기능 개선이 필요할 때 **Rolling Update**를 통해 사용자의 서비스 이용을 방해하지 않고(**Zero Downtime**) 새로운 도커 이미지 버전으로 교체한다.

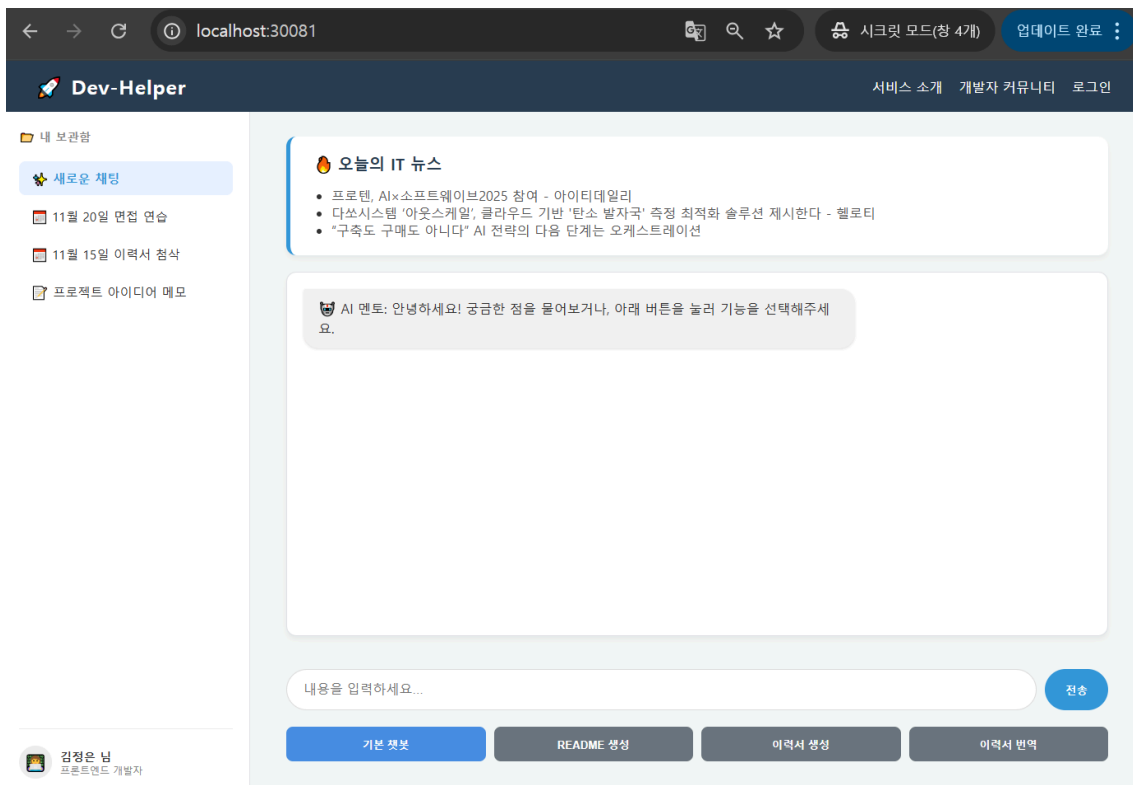
#### 3.1.2. 핵심 기술

Docker, Deployment(Rolling Update)

#### 3.1.3. 동작 흐름

1. 현재 v8 버전의 프론트엔드 Pod가 정상 실행 중이다.
2. 새로운 기능이 추가된 Docker 이미지를 빌드하고 새로운 태그를 붙인다.
3. 클러스터에 이미지 변경 명령을 내린다.
4. 기존 Pod가 하나씩 종료되고 새 Pod가 생성되는 과정을 실시간으로 확인한다.

### 3.1.4. 시연



현재 웹사이트 접속 시 정상적으로 페이지가 출력되는 것을 확인할 수 있다.

```
root@vm1:~/manifest# kubectl describe deployment front -n front-ns | grep Image
Image:          ghcr.io/25-2-cloudcomputingsystems-team11/front:v8
root@vm1:~/manifest# kubectl get pods -n front-ns
NAME                                READY   STATUS    RESTARTS   AGE
front-5f967dd64b-sntqm             1/1     Running   0           57s
front-5f967dd64b-zcgfn             1/1     Running   0           58s
root@vm1:~/manifest#
```

해당 도커 이미지의 버전은 v8이며, 현재 Pod의 상태를 확인했다.

```
! front-deployment.yaml X
manifest > ! front-deployment.yaml
20 spec:
21   imagePullSecrets:
22     - name: ghcr-secret
23   containers:
24     - name: front
25       image: ghcr.io/25-2-cloudcomputingsystems-team11/front:v10
```

새롭게 적용하고자 하는 이미지 버전으로 변경하여 front-deployment.yaml을 수정한다.  
v10으로 변경해주었다.

```
root@vm1:~/manifest# kubectl apply -f ./front-deployment.yaml
deployment.apps/front configured
root@vm1:~/manifest#
```

yaml 파일을 수정한 후 apply 해주었다.

```
root@vm1:~/manifest# kubectl get pods -n front-ns
```

| NAME                   | READY | STATUS            | RESTARTS | AGE |
|------------------------|-------|-------------------|----------|-----|
| front-5f967dd64b-sntqm | 1/1   | Terminating       | 0        | 92s |
| front-5f967dd64b-zcgfn | 1/1   | Running           | 0        | 93s |
| front-85b59f5f65-rdpzk | 1/1   | Running           | 0        | 2s  |
| front-85b59f5f65-wfsq9 | 0/1   | ContainerCreating | 0        | 1s  |

```
root@vm1:~/manifest# kubectl get pods -n front-ns
```

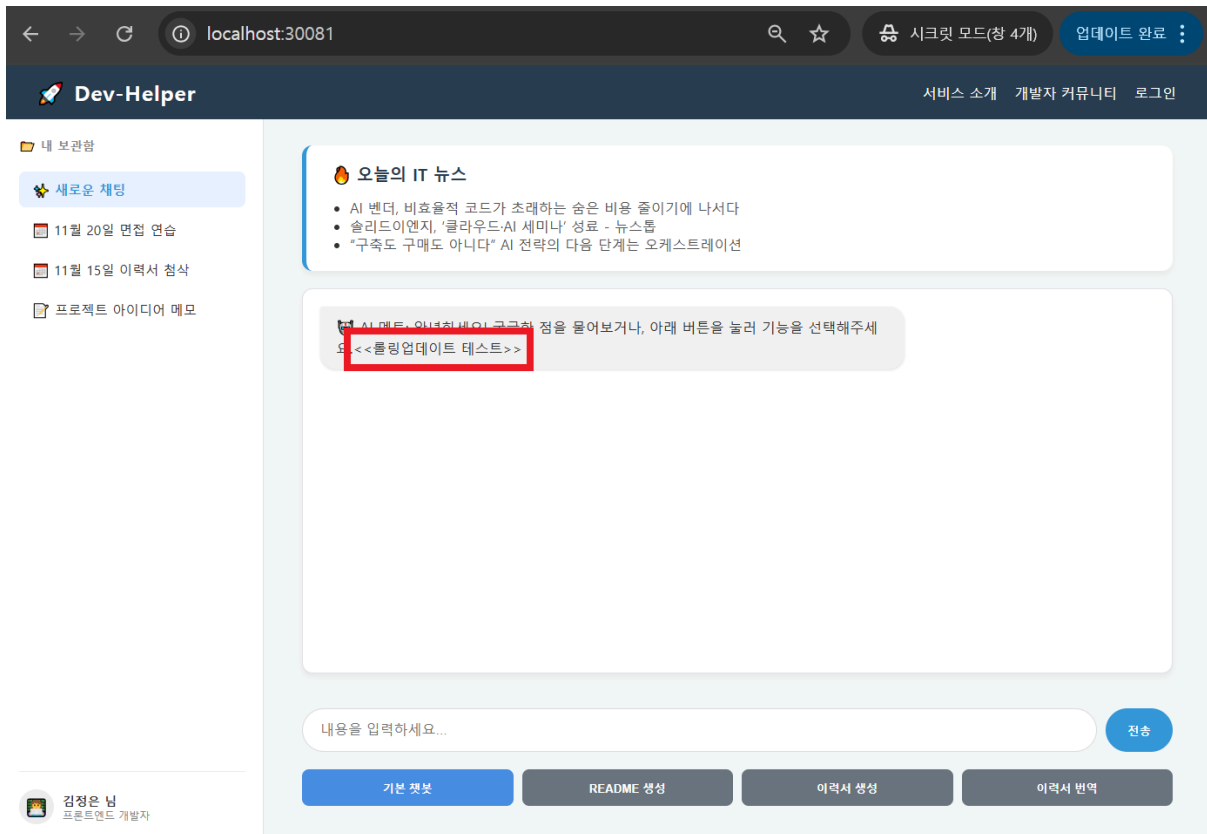
| NAME                   | READY | STATUS  | RESTARTS | AGE |
|------------------------|-------|---------|----------|-----|
| front-85b59f5f65-rdpzk | 1/1   | Running | 0        | 5s  |
| front-85b59f5f65-wfsq9 | 1/1   | Running | 0        | 4s  |

기존 Pod가 Terminating 되고, 새 Pod가 생성 및 Running되는 과정이다.

```
root@vm1:~/manifest# kubectl describe deployment front -n front-ns | grep Image
Image:          ghcr.io/25-2-cloudcomputingsystems-team11/front:v10
root@vm1:~/manifest#
```

```
root@vm1:~/manifest# kubectl rollout status deployment/front -n front-ns
deployment "front" successfully rolled out
root@vm1:~/manifest#
```

해당 도커 이미지의 버전이 v10으로 잘 적용됨을 확인했다.



Rolling Update 테스트를 위해 생성해둔 v10 이미지가 잘 적용되었다.

이처럼 해당 프로젝트에 Rolling Update를 적용하여 Deployment를 업데이트할 때 한 번에 전체를 바꾸지 않고, 점진적으로 교체하여 다운타임 없이 무중단으로 배포할 수 있도록 하였다.

## 3.2. CronJob 기반의 자동화된 뉴스 데이터 수집

### 3.2.1. 개요

최신 IT 트렌드 정보를 제공하기 위해 CronJob을 사용하여 5분마다 뉴스 크롤러를 자동으로 실행한다. 수집된 데이터는 PVC로 영구 저장된 MySQL 데이터베이스에 적재되며, 사용자는 항상 최신 뉴스를 볼 수 있다.

### 3.2.2. 핵심 기술

CronJob, Python Crawler, MySQL(PVC)

### 3.2.3. 동작 흐름

1. 설정된 스케줄(\*/\* \* \* \* \*)에 따라 크롤러 Pod가 자동으로 생성된다.
2. 크롤러는 외부 뉴스 사이트 데이터를 수집하여 MySQL에 저장(Insert)하고 종료(Completed)된다.
3. 로그를 통해 뉴스 수집 성공 메시지를 확인할 수 있다.

### 3.2.4. 시연

```
root@vm1:~/manifest# kubectl get cronjob -n news-app
NAME          SCHEDULE    TIMEZONE    SUSPEND    ACTIVE    LAST SCHEDULE    AGE
news-crawler  */5 * * * * <none>      False      0          109s     29h
root@vm1:~/manifest#
```

현재 등록된 CronJob을 확인했다.

```
root@vm1:~/manifest# kubectl get pod -n news-app -w
NAME          READY    STATUS    RESTARTS    AGE
news-api-798b576496-8plxh  1/1      Running   0           23m
news-api-798b576496-9nm85  1/1      Running   0           29h
news-api-798b576496-sdh94  1/1      Running   0           29h
news-crawler-29425535-wzllf 0/1      Completed 0           4m31s
news-crawler-29425540-nbvbs 0/1      Pending   0           0s
news-crawler-29425540-nbvbs 0/1      Pending   0           0s
news-crawler-29425540-nbvbs 0/1      ContainerCreating 0           0s
news-crawler-29425540-nbvbs 1/1      Running   0           14s
news-crawler-29425540-nbvbs 0/1      Completed 0           23s
news-crawler-29425540-nbvbs 0/1      Completed 0           24s
news-crawler-29425540-nbvbs 0/1      Completed 0           25s
news-crawler-29425535-wzllf 0/1      Completed 0           5m25s
news-crawler-29425535-wzllf 0/1      Completed 0           5m25s
```

news-crawler Pod가 5분마다 새롭게 생성되며, 크롤링 작업 후에 'Completed' 상태가 된다.

```
root@vm1:~# kubectl logs -n news-app -l app=crawler --tail=10
--- Start Job ---
[Crawler] RSS 조회: https://www.itworld.co.kr/rss/feed/index.php
[Crawler] RSS 조회: https://www.ciokorea.com/rss/feed/index.php
[Crawler] RSS 조회: https://news.google.com/rss/search?q=%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5+OR+%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4+OR+%ED%81%B4%EB%9D%BC%EC%9A%B0%EB%93%9C+when:1d&hl=ko&gl=KR&ceid=KR:ko
[Crawler] 총 15개 기사 확보 후 서버 완료.
[MySQL] 저장 완료: 3건 (중복 건너뛴)
[Redis] 캐시 업데이트 완료 (이번 턴에 선정된 뉴스: 3건).
--- Job Finished ---
root@vm1:~#
```

크롤러의 로그를 확인하여 크롤링 성공 여부를 검증했다.

### 3.3. 트래픽 급증에 따른 **HPA** 오토스케일링 및 부하 분산

#### 3.3.1. 개요

채용이 활발한 시즌 또는 마케팅 이벤트 등으로 인해 대량의 사용자 트래픽이 유입될 경우를 가정한다. CPU 사용량이 임계치(70%)를 초과하면 HPA가 이를 감지하여 Pod 개수를 자동으로 늘려(Scale-out) 서비스를 안정화한다.

#### 3.3.2. 핵심 기술

HPA(Horizontal Pod Autoscaler)

#### 3.3.3. 동작 흐름

1. 현재 프론트엔드 Pod는 최소 개수(MinReplicas: 2)로 유지 중이다.
2. HPA 정상 작동 테스트를 위해 Pod 내부에 들어가 강제로 CPU 연산을 시켜 부하를 발생시킨다.
3. HPA가 CPU 부하를 감지하고 REPLICAS 수치를 증가시킨다.
4. 부하가 중단되면 다시 Pod 개수가 서서히 감소(Scale-in)한다.

#### 3.3.4. 시연

```
root@vm1:~# kubectl get pod -n front-ns
NAME                                READY   STATUS    RESTARTS   AGE
front-85b59f5f65-rdpzk             1/1     Running   0           24m
front-85b59f5f65-wfsq9             1/1     Running   0           24m
root@vm1:~#
```

프론트엔드 Pod는 최소 개수인 2개로 유지 중이다.

```
root@vm1:~# kubectl exec -it front-85b59f5f65-rdpzk -n front-ns -- /bin/sh -c "yes > /dev/null"
^C
```

임의의 프론트엔드 Pod 내부에 들어가 CPU 연산을 강제로 시켜 부하를 주었다. yes 명령어는 무한히 “y”를 출력하는데, 이를 /dev/null로 보내 단시간에 CPU 사용량을 늘리는 방식을 사용했다.

```
root@vm1:~# kubectl get hpa -n front-ns -w
```

| NAME      | REFERENCE        | TARGETS       | MINPODS | MAXPODS | REPLICAS | AGE |
|-----------|------------------|---------------|---------|---------|----------|-----|
| front-hpa | Deployment/front | cpu: 0%/70%   | 2       | 10      | 2        | 29h |
| front-hpa | Deployment/front | cpu: 226%/70% | 2       | 10      | 2        | 29h |
| front-hpa | Deployment/front | cpu: 226%/70% | 2       | 10      | 4        | 29h |
| front-hpa | Deployment/front | cpu: 166%/70% | 2       | 10      | 7        | 29h |
| front-hpa | Deployment/front | cpu: 98%/70%  | 2       | 10      | 7        | 29h |
| front-hpa | Deployment/front | cpu: 50%/70%  | 2       | 10      | 7        | 29h |

TARGETS의 CPU 사용량이 설정해둔 70%를 넘어가니 REPLICAS가 2 -> 4 -> 7 과 같이 점진적으로 늘어나는 것을 확인할 수 있다.

```
root@vm1:~# kubectl get pod -n front-ns
```

| NAME                   | READY | STATUS  | RESTARTS | AGE |
|------------------------|-------|---------|----------|-----|
| front-85b59f5f65-8wt7r | 1/1   | Running | 0        | 43s |
| front-85b59f5f65-d6z64 | 1/1   | Running | 0        | 43s |
| front-85b59f5f65-g4k7q | 1/1   | Running | 0        | 58s |
| front-85b59f5f65-gwkvs | 1/1   | Running | 0        | 58s |
| front-85b59f5f65-rdpzk | 1/1   | Running | 0        | 33m |
| front-85b59f5f65-t72j9 | 1/1   | Running | 0        | 43s |
| front-85b59f5f65-wfsq9 | 1/1   | Running | 0        | 33m |

```
root@vm1:~#
```

프론트엔드 Pod가 7개로 확장되어 실행 중임을 확인했다.

## 3.4. PVC 기반의 데이터 영속성(Persistence) 보장

### 3.4.1. 개요

Kubernetes 환경에서 Pod는 언제든지 종료되거나 재생성될 수 있는 휘발성 리소스이다. 따라서 데이터베이스의 안정성을 위해 스토리지 영역을 Pod와 분리해야 한다. 본 시나리오는 MySQL Pod를 강제로 삭제하여 장애 상황을 연출한 뒤, 새로 생성된 Pod가 기존 PVC에 연결되어 데이터 유실 없이 서비스를 지속하는지 검증한다.

### 3.4.2. 핵심 기술

Kubernetes PVC(Persistent Volume Claim), MySQL, Deployment(Self-healing)

### 3.4.3. 동작 흐름

1. 현재 실행 중인 MySQL Pod에 접속하여 축적된 뉴스 데이터의 개수를 확인한다.
2. 실행 중인 MySQL Pod를 강제로 삭제(delete)하여 서버 다운 상황을 연출한다.
3. Kubernetes 컨트롤러가 즉시 새로운 Pod를 생성하여 기존에 사용하던 PVC(Volume)을 자동으로 다시 마운트한다.
4. 새로 생성된 Pod에 접속하여 데이터 개수를 다시 조회했을 때, 이전과 동일한 데이터가 그대로 보존되어 있음을 확인한다.

### 3.4.4. 시연

```
root@vm1:~# kubectl get pvc -n news-db
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS  AGE
mysql-pvc     Bound    pvc-53552fe7-a18f-4465-ad8d-b26ebb0e9068  1Gi        RWO             local-path
<unset>      3d20h
root@vm1:~#
```

mysql-pvc가 Bound 상태로, 1Gi 용량의 영구 스토리지가 MySQL Pod에 정상적으로 마운트되어 있다.

```
root@vm1:~# kubectl exec -it mysql-5b5c57b78c-tkgzx -n news-db -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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

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

mysql>
```

실제 크롤링된 데이터가 들어있는지 확인하기 위해 MySQL Pod에 직접 접속해 SQL 명령어를 실행할 것이다.



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

Database changed
mysql>
```

MySQL 접속 후 news\_db 데이터베이스를 선택해주었다.

```
mysql> SELECT COUNT(*) FROM it_news;
+-----+
| COUNT(*) |
+-----+
|      97 |
+-----+
1 row in set (0.00 sec)

mysql>
```

크롤링한 뉴스 데이터가 97개 쌓여 있음을 확인했다.

```
mysql> SHOW COLUMNS FROM it_news;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| title | varchar(255) | YES | UNI | NULL    |
| link  | varchar(768) | YES |     | NULL    |
| crawled_at | datetime | YES |     | NULL    |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

해당 데이터베이스에는 id, title, link, crawled\_at 항목이 저장된다.

```
mysql> SELECT id, title FROM it_news ORDER BY id LIMIT 7;
+-----+-----+
| id | title |
+-----+-----+
| 1 | 한국-Arm, 반도체·AI 인재 1,400명 양성 MOU 체결 |
| 2 | 과기정통부·KISA, '2025 소프트웨어 공급망 보안 모델 구축지원 성과공유회' 개최 - 데일리시큐 |
| 3 | Get data, and the data culture, ready for AI |
| 4 | 安全なLLMエージェントを作るためのリスクとガバナンス—幻のセキュリティ的責任 |
| 5 | 분산형 AI GPU 클라우드 플랫폼 '에이아이브', 국제 정보보호 표준 'ISO 27001' 인증 획득 - 인공지능신문 |
| 6 | SAS, 2026년 AI 산업을 이룰 8가지 전망 공개...책임성·ROI 중요성 커져 - cio.com |
| 7 | [미디어] [후기] "2026년 1월 인공지능법 시행, 어떤 일이 벌어지나" 기자 설명회 - 한국여성민우회 |
+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

DB 내부를 조회해보면, 크롤러가 5분마다 수집한 축적된 뉴스 데이터를 확인할 수 있다. Redis Pod를 통해 홈페이지 메인 화면에 띄워주는 상단의 3개 뉴스가 캐싱되어 있는 것을 확인해보겠다.

```
root@vm1:~# kubectl exec -it redis-5f86f8f9c7-qvp56 -n news-db -- redis-cli
127.0.0.1:6379>
```

임의의 Redis Pod 내부로 들어가 Redis CLI에 접속했다.

```
127.0.0.1:6379> KEYS *
1) "daily_top3"
127.0.0.1:6379>
```

Redis DB의 Key 목록에 daily\_top3가 있음을 확인했다.

```
127.0.0.1:6379> GET daily_top3
"[{"title": "\uAI \u\uba4\u\ub354, \u\ube44\u\ud6a8\u\uc728\u\uc801 \u\uf54\u\ub4dc\u\ud000 \u\ud008\u\ub798\u\ud558\u\ub294 \u\uc228\u\uc740 \u\ub
e44\u\uc6a9 \u\uc904\u\uc774\u\uae30\u\uc5d0 \u\ub098\u\uc11c\u\ub2e4", "link": "\uhttps://www.cio.com/article/4105436/ai-%eb%b2%a4%eb%8d%94
-%eb%b9%84%ed%9a%a8%ec%9c%a8%ec%a0%81-%ec%bd%94%eb%93%ea%b0%80-%ec%b4%88%eb%9e%98%ed%95%98%eb%8a%94-%ec%88%a8%ec%9d%80-%eb%b9%84%ec
%9a%a9-%ec%a4%84%ec%9d%b4%ea%b8%b0%ec%97%90.html"}, {"title": "\u\ud6c\u\ud000\u\uf54\u\ub9ac\u\uc544, \u\uc2e0\u\uc784 \u\uc0ac\u\uc7a5\u
\uc5d0 \u\uc724\u\ud6c \u\uc804 \u\uc560\u\ud50c\u\uf54\u\ub9ac\u\uc544 \u\ub300\u\ud45c \u\uc120\u\uc784", "link": "\uhttps://www.cio.com/art
icle/4105388/%ea%b5%ac%ea%b8%80%ec%bd%94%eb%a6%ec%95%84-%ec%8b%a0%ec%9e%84-%ec%82%ac%ec%9e%a5%ec%97%90-%ec%9c%a4%ea%b5%ac-%ec%a0%84
-%ec%95%a0%ed%94%8c%ec%bd%94%eb%a6%ac%ec%95%84-%eb%8c%80%ed%91%9c-%ec%84%a0.html"}, {"title": "\u\ud01c\u\uc18c\u\ub984\u\ud01d\u\ud026
\u\ub9e5\u\ub3c4\u\ub0a0\u\ub4dc \u\ud06c\u\ub9ac\u\uc2a4\u\ub9c8\u\uc2a4 \u\ud018AI \u\ud011\u\ud000\u\ud019 \u\uc0ad\u\uc81c, \u\ub3c4\u\ub300\u\uccb4
\u\uc5b4\u\ub560\u\uae38\u\ub798 - \u\ud55c\u\ud000\u\ub808", "link": "\uhttps://news.google.com/rss/articles/CBMibkFVX3lxTFBab1ducExDaWFW
M0V0dGVxOUZxbERRR1I5UnU3cUhzX3lBw1lNmGZaMVpTVmIzeGx0SzVJTW0wdXlkUnV2VmklUTBCWldadDZ1dXZudzVqNXAxWko4Z01jSXpnYWhKMm1tdDU1VmNR?oc=5"}]"
127.0.0.1:6379>
```

daily\_top3 키에는 최신 IT 기사 3개의 title, link 데이터가 저장되어 있다. 기사 정보는 JSON 문자열 형태로 저장되어 있고, 해당 데이터가 DB를 거치지 않고 바로 메모리에서 서빙되어 속도가 빠르다.

```
root@vm1:~# kubectl get pods -n news-db
NAME                                READY    STATUS    RESTARTS    AGE
mysql-5b5c57b78c-tkgzx             1/1     Running   0            30h
redis-5f86f8f9c7-qvp56             1/1     Running   0            30h
root@vm1:~#
```

```
root@vm1:~# kubectl delete pod mysql-5b5c57b78c-tkgzx -n news-db
pod "mysql-5b5c57b78c-tkgzx" deleted
root@vm1:~#
root@vm1:~# kubectl get pods -n news-db
NAME                                READY    STATUS    RESTARTS    AGE
mysql-5b5c57b78c-dp2ks             1/1     Running   0            22s
redis-5f86f8f9c7-qvp56             1/1     Running   0            30h
root@vm1:~#
```

PVC 테스트를 위해 MySQL Pod를 삭제하여 장애 발생을 연출할 것이다. Deployment 설정으로 인해 Pod가 자동으로 생성된다.

```
root@vm1:~# kubectl exec -it mysql-5b5c57b78c-dp2ks -n news-db -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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

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

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

Database changed
mysql> 
```

새롭게 생성된 MySQL Pod 내부에 접속해보겠다.

```
mysql> SELECT COUNT(*) FROM it_news;
+-----+
| COUNT(*) |
+-----+
|      99 |
+-----+
1 row in set (0.00 sec)

mysql> 
```

PVC 덕분에 Pod가 삭제된 후 새롭게 생성되는 상황에서도 데이터가 안전하게 보존되는 것을 확인했다.

## 4. 시스템 설계

### 4.1. Docker

#### a. 마이크로서비스 아키텍처(MSA) 도입 및 컨테이너 전략

시스템의 확장성과 유지보수 효율성을 극대화하기 위해 마이크로서비스 아키텍처(Microservices Architecture, MSA)를 채택하여 시스템을 설계하였다. 프론트엔드(Web)와 백엔드(API)를 비즈니스 기능 단위의 독립적인 서비스로 분리하고, 각각을 **Docker** 컨테이너로 캡슐화하여 서비스 간의 결합도를 획기적으로 낮추었다. 이를 통해 각 마이크로서비스는 서로 다른 기술 스택과 라이브러리 환경을 제약 없이 사용할 수 있으며 상호 간섭 없는 독자적인 배포 생명주기와 유연한 스케일링 환경을 확보하였다.

#### b. 컴포넌트별 상세 이미지 구현

##### 1) 프론트엔드

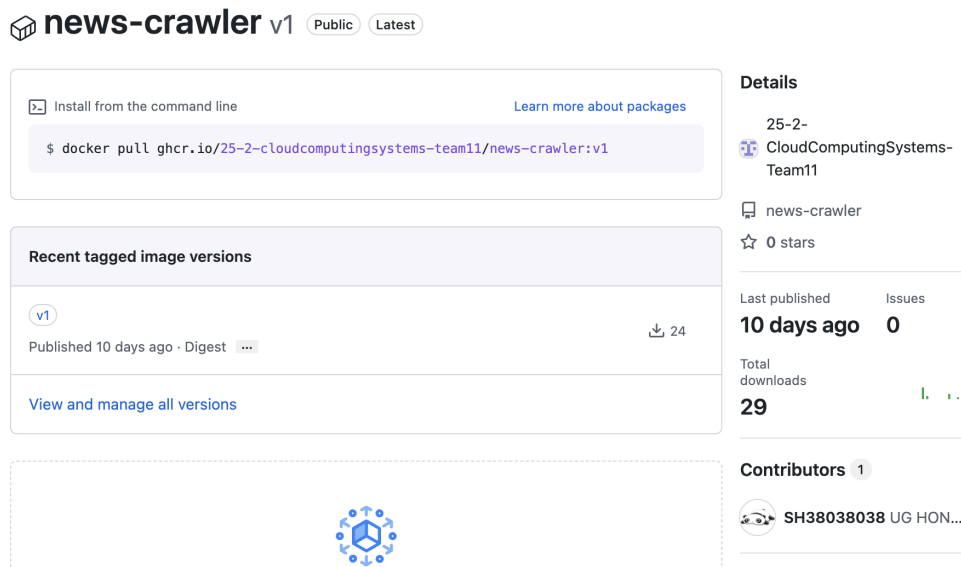
The screenshot displays the Docker Hub interface for the 'front' image. The main section shows the 'front v11' image, which is public and the latest version. The installation instructions are provided in a code block: `$ docker pull ghcr.io/25-2-cloudcomputingsystems-team11/front:v11`. Below this, a table lists recent tagged image versions: v11, v10, v9, v8, and v7. The details section on the right shows that the image was published 8 days ago, has 0 issues, and 72 total downloads. It also lists contributors: hiiminjeong and JEc0de-2002.

| Version | Published   | Digest | Downloads |
|---------|-------------|--------|-----------|
| v11     | 8 days ago  | ...    | 9         |
| v10     | 8 days ago  | ...    | 9         |
| v9      | 8 days ago  | ...    | 9         |
| v8      | 10 days ago | ...    | 23        |
| v7      | 10 days ago | ...    | 2         |

프론트엔드 이미지는 소스 코드를 빌드하는 환경과 실제 서비스를 제공하는 환경을 분리하는 'Multi-stage Build' 전략을 적용하여 최적화하였다.

- **Build Stage (Node.js 18):** 애플리케이션 빌드 단계에서는 **node:18** 이미지를 사용하여 소스 코드 컴파일 및 패키징을 수행한다. 이 단계에서는 개발 도구와 원본 소스 코드가 존재한다.
- **Serve Stage (Nginx 1.25):** 최종 배포되는 이미지 단계에서는 빌드 결과물(Static Files)만을 **nginx:1.25** 이미지로 복사(COPY)하여 구동한다.
- 구현 효과: 최종 이미지에는 **Node.js** 런타임이나 소스 코드가 포함되지 않으므로 이미지 크기를 획기적으로 경량화하였으며, 불필요한 파일 노출을 차단하여 보안성을 강화하였다.

## 2) 백엔드



**news-crawler v1** Public Latest

Install from the command line [Learn more about packages](#)

```
$ docker pull ghcr.io/25-2-cloudcomputingsystems-team11/news-crawler:v1
```

**Recent tagged image versions**

**v1** Published 10 days ago · Digest ... [24](#)

[View and manage all versions](#)

**Details**

25-2-CloudComputingSystems-Team11

news-crawler

0 stars

Last published 10 days ago Issues 0

Total downloads 29

**Contributors 1**

SH38038038 UG HON...

백엔드 이미지는 **Python** 기반의 **AI** 모델 연산과 **API** 처리를 전담하도록 설계되었으며, 불필요한 시스템 리소스 낭비를 막기 위해 **Base Image** 최적화에 집중하였다.

- **Base Image** 선정: 무거운 OS 패키지가 제외된 **python:3.9-slim**을 사용하여 컨테이너의 기본 용량을 최소화하였다.
- 의존성 격리: **FastAPI**, **Uvicorn**, **PyMySQL**, **OpenAI** 등 프로젝트 구동에 필수적인 라이브러리만을 **requirements.txt**를 통해 명시적으로 설치하였다. 이는 다른 서비스(프론트엔드 등)의 라이브러리 충돌 가능성을 원천적으로 차단한다.

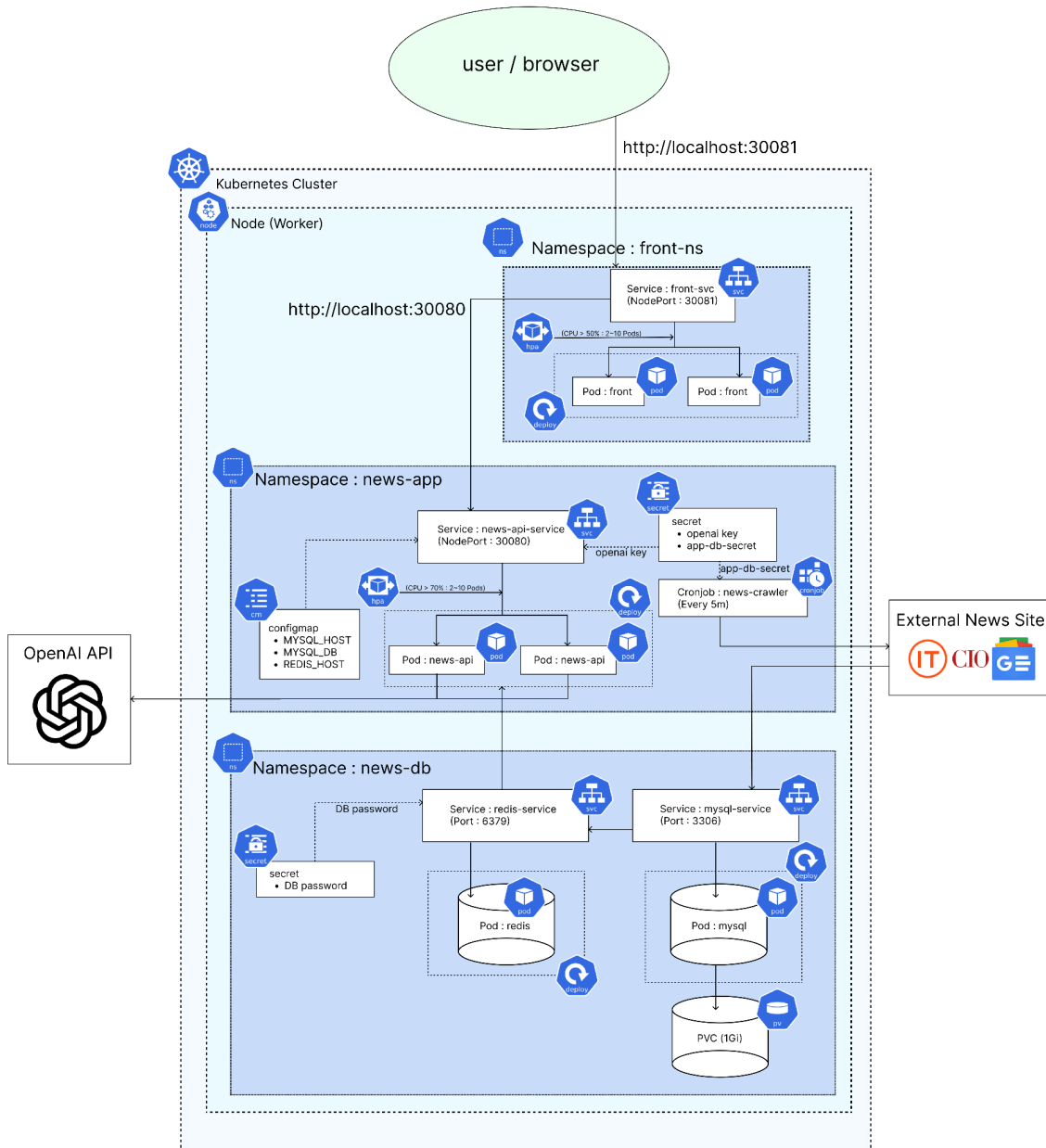
## c. 이미지 분리를 통한 시스템 고도화

이와 같은 철저한 이미지 분리 전략을 통해 다음과 같은 기술적 이점을 확보하였다.

- 1) 독립적 스케일링: 트래픽이 집중되는 백엔드 컨테이너만 쿠버네티스 상에서 개별적으로 복제(**Scale-out**)할 수 있는 구조를 확립하였다.
- 2) 장애 격리: 특정 컨테이너(예: 백엔드 로직 오류)에 문제가 발생하더라도, 분리된 프론트엔드(**Nginx**) 컨테이너는 정상적으로 정적 페이지를 서빙할 수 있어 전면적인 서비스 중단을 방지한다.
- 3) 배포 효율성: 각 이미지가 기능 단위로 경량화되어 있어, 수정 사항 배포 시 전체 시스템이 아닌 변경된 이미지만 빠르게 빌드 및 배포 가능하다.

## 4.2. Kubernetes

**Kubernetes** 클러스터 아키텍처 및 오케스트레이션



#### 4.2.1. HPA (Horizontal Pod Autoscaling) 기반 오토스케일링

```

root@vm1:~# kubectl get hpa -A

```

| NAMESPACE | NAME         | REFERENCE           | TARGETS      | MINPODS | MAXPODS | REPLICAS | AGE |
|-----------|--------------|---------------------|--------------|---------|---------|----------|-----|
| front-ns  | front-hpa    | Deployment/front    | cpu: 0%/70%  | 2       | 10      | 2        | 31h |
| news-app  | news-api-hpa | Deployment/news-api | cpu: 60%/70% | 2       | 10      | 4        | 31h |

```

root@vm1:~#

```

#### a. 사용 이유

시스템의 안정성을 확보하고 리소스 효율성을 극대화하기 위해 **CPU** 사용률을 기준으로 **Pod**의 개수를 자동으로 조절하는 **HPA**를 **front-ns**와 **news-app** 네임스페이스에 각각 적용하였다.

#### b. 상세 운영 메커니즘

- 1) 동적 스케일링: 특정 기능에 트래픽이 집중되어 부하가 급증할 경우(**CPU** 사용량 **70%**) **HPA**가 이를 감지하여 즉시 **Pod** 개수를 설정된 최대치(**Max Replicas: 10**)까지 자동으로 확장(**Scale-Out**)함으로써 응답 지연 및 서비스 중단을 방지하였다.
- 2) 자원 회수 및 최적화 (**Scale-In**): 트래픽이 감소하여 **Pod**들의 평균 **CPU** 사용률이 설정된 목표치 이하로 떨어지면, **HPA**가 이를 감지하여 불필요해진 리소스로 판단하고 **Pod** 개수를 자동으로 최소 개수(**Min Replicas: 2**)까지 점진적으로 감소시킨다. 이를 통해 유휴 자원 낭비를 막고 클러스터 리소스를 효율적으로 유지한다.

### 4.2.2. ReplicaSet을 이용한 Self-Healing (자동 복구)

#### a. 사용 이유

서버나 애플리케이션에 예기치 못한 오류가 발생하여 프로세스가 종료되더라도, 시스템이 사람의 개입 없이 스스로 복구할 수 있는 **Self-Healing** 체계를 갖추기 위함이다.

#### b. 상세 메커니즘

- 상태 유지: 쿠버네티스 컨트롤러는 현재 실행 중인 **Pod**의 개수가 선언된 개수와 일치하는지 지속적으로 감시한다.
- 자동 재시작: 만약 특정 노드의 장애나 애플리케이션 버그로 인해 **Pod**가 비정상 종료되거나 삭제될 경우, 즉시 새로운 **Pod**를 생성하고 스케줄링하여 서비스 가용성을 **24시간** 유지한다.



#### 4.2.3. CronJob을 활용한 데이터 파이프라인 자동화

```
root@vm1:~/manifest# kubectl get cronjob -n news-app
NAME          SCHEDULE    TIMEZONE    SUSPEND    ACTIVE    LAST SCHEDULE    AGE
news-crawler  */5 * * * * <none>      False      0          109s     29h
root@vm1:~/manifest#
```

##### a. 사용 이유

외부 뉴스 사이트로부터 최신 IT 트렌드 정보를 수집하기 위해 Kubernetes CronJob 리소스를 활용하여 데이터 수집 파이프라인을 자동화하였다.

##### b. 상세 기능 구현

- 1) 정기 크롤링 수행: **news-app** 네임스페이스에 배치된 **news-crawler** 크롤러가 5분 주기로 실행되도록 스케줄링하였다. 이를 통해 최신 뉴스를 지속적으로 수집하여 DB에 적재함으로써 사용자는 별도의 검색 과정 없이 실시간 트렌드를 제공할 수 있다.
- 2) 운영 안정성 확보: 수동 실행으로 인한 누락이나 실수를 배제하고, 시스템이 스스로 최신 상태를 유지하는 자율 운영 환경을 구축하였다.

#### 4.2.4. NodePort와 ClusterIP를 활용한 네트워크 구성

```
root@vm1:~# kubectl get svc -n front-ns
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
front-svc NodePort     10.43.72.194  <none>         80:30081/TCP     31h
root@vm1:~# kubectl get svc -n news-app
NAME              TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
news-api-service NodePort     10.43.158.59  <none>         80:30080/TCP     31h
root@vm1:~# kubectl get svc -n news-db
NAME              TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
mysql-service     ClusterIP   10.43.230.224 <none>         3306/TCP         3d22h
redis-service     ClusterIP   10.43.228.189 <none>         6379/TCP         3d22h
root@vm1:~#
```

##### a. 사용 이유

외부 접근이 필요한 서비스와 내부 보안이 필요한 서비스를 구분하기 위해 Service 타입을 NodePort와 ClusterIP로 이원화하여 네트워크 정책을 수립하였다.

## b. 상세 네트워크 흐름

- 1) 외부 접근 (NodePort): 사용자 접근이 필수적인 프론트엔드와 API 서버는 NodePort 타입으로 설정하여 외부 트래픽을 허용하였다.
  - Frontend: 30081 포트를 통해 사용자의 브라우저 요청을 front-svc로 연결한다.
  - Backend: 30080 포트를 통해 API 요청을 news-api-service로 라우팅한다.
- 2) 내부 통신 (ClusterIP): 보안이 중요한 데이터베이스(MySQL)와 캐시(Redis)는 news-db 네임스페이스 내에서 ClusterIP로 구성하여, 클러스터 외부에서의 직접적인 접근을 원천 차단하고 내부 서비스 간의 통신만 허용하여 데이터 보안을 강화하였다.

## 4.2.5. Rolling Update 전략을 통한 무중단 배포

```
root@vm1:~# kubectl describe deployment front -n front-ns | grep Strategy
StrategyType:          RollingUpdate
RollingUpdateStrategy: 25% max unavailable, 25% max surge
root@vm1:~# kubectl describe deployment news-api -n news-app | grep Strategy
StrategyType:          RollingUpdate
RollingUpdateStrategy: 0 max unavailable, 1 max surge
root@vm1:~#
```

## a. 사용 이유

서비스 업데이트 시 발생할 수 있는 다운타임을 제거하기 위해 쿠버네티스 Deployment의 기본 전략인 Rolling Update를 적용하여고가용성 환경을 구축하였다.

## b. 적용 효과

- 1) Zero Downtime: 기능 개선이나 프롬프트 수정 사항 배포 시, 기존 Pod를 유지한 상태에서 새로운 버전의 Pod를 순차적으로 생성 및 교체함으로써 사용자는 서비스 끊김 없이 최신 기능을 이용할 수 있다.
- 2) 리스크 관리: 배포 직후 치명적인 오류가 발견될 경우, 즉시 이전 리버전으로 롤백(Rollback)할 수 있는 체계를 갖추어 운영 리스크를 최소화하였다.

#### 4.2.6. Namespace 기반의 리소스 격리 및 멀티 테넌시

```
root@vm1:~# kubectl get namespace | grep -E "front-ns|news-app|news-db"
front-ns      Active   31h
news-app      Active   31h
news-db       Active   3d22h
root@vm1:~#
```

##### a. 사용 이유

단일 클러스터 내에서 서비스 간의 간섭을 방지하고 관리 효율을 높이기 위해 용도에 따라 Namespace를 front-ns, news-app, news-db 3가지로 논리적 분리하였다.

##### b. 구성 상세 및 효과

###### 1) 계층별 격리:

- front-ns: 사용자 인터페이스 담당 (UI)

```
root@vm1:~# kubectl get all -n front-ns
NAME                                READY   STATUS    RESTARTS   AGE
pod/front-5f967dd64b-7d4v2         1/1     Running   0           149m
pod/front-5f967dd64b-pt2km         1/1     Running   0           148m

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
service/front-svc                  NodePort      10.43.72.194  <none>       80:30081/TCP     33h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/front              2/2     2             2           33h

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/front-5f967dd64b    2         2         2       33h

NAME                                REFERENCE          TARGETS          MINPODS   MAXPODS
REPLICAS    AGE
horizontalpodautoscaler.autoscaling/front-hpa    Deployment/front   cpu: 0%/70%     2         10
2           33h
root@vm1:~#
```

- news-app: 비즈니스 로직 및 외부 API 연동 담당 (Application)

```

root@vm1:~# kubectl get all -n news-app
NAME                                READY    STATUS    RESTARTS   AGE
pod/news-api-798b576496-7mflp      1/1     Running   0           146m
pod/news-api-798b576496-9nm85      1/1     Running   0           32h
pod/news-api-798b576496-j42fh      1/1     Running   0           145m
pod/news-api-798b576496-sdh94      1/1     Running   0           32h
pod/news-crawler-29425675-9hdls     0/1     Completed 0           70m

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
service/news-api-service            NodePort      10.43.158.59  <none>       80:30080/TCP 32h

NAME                                READY    UP-TO-DATE    AVAILABLE   AGE
deployment.apps/news-api            4/4      4              4           32h

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/news-api-798b576496 4          4          4        32h

NAME                                REFERENCE    TARGETS    MINPODS    MAXPODS
REPLICAS    AGE
horizontalpodautoscaler.autoscaling/news-api-hpa  Deployment/news-api  cpu: 50%/70%  2          10
4          32h

NAME                                SCHEDULE    TIMEZONE    SUSPEND    ACTIVE    LAST    SCHEDULE    AGE
cronjob.batch/news-crawler          */5 * * * * <none>     False      0          70m      32h

NAME                                STATUS    COMPLETIONS    DURATION    AGE
job.batch/news-crawler-29425675     Complete  1/1            15s         70m
root@vm1:~#

```

- news-db: 데이터 영속성 담당 (Data Layer)

```

root@vm1:~# kubectl get all -n news-db
NAME                                READY    STATUS    RESTARTS   AGE
pod/mysql-5b5c57b78c-dp2ks         1/1     Running   0           151m
pod/redis-5f86f8f9c7-qvp56         1/1     Running   0           33h

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
service/mysql-service              ClusterIP      10.43.230.224  <none>       3306/TCP   3d23h
service/redis-service              ClusterIP      10.43.228.189  <none>       6379/TCP   3d23h

NAME                                READY    UP-TO-DATE    AVAILABLE   AGE
deployment.apps/mysql              1/1      1              1           3d23h
deployment.apps/redis              1/1      1              1           3d23h

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/mysql-5b5c57b78c   1          1          1        3d23h
replicaset.apps/redis-5f86f8f9c7   1          1          1        3d23h
root@vm1:~#

```

- 2) 안정성 보장: 애플리케이션 계층(news-app)에서 과도한 부하가 발생하거나 배포 실수가 있더라도, 데이터 계층(news-db)은 독립된 네임스페이스에서 보호되므로 시스템 전체의 붕괴를 막고 데이터 무결성을 유지할 수 있다.

### 4.3. 데이터베이스 및 캐시 아키텍처

시스템의 데이터 무결성을 보장하고 읽기 성능을 최적화하기 위해 관계형 데이터베이스(MySQL)와 인메모리 키-값 저장소(Redis)를 혼합한 Polyglot Persistence 전략을 news-db 네임스페이스 내에 구축하였다.

#### 4.3.1. RDBMS: MySQL 기반의 데이터 영속성 확보

##### a. 사용 이유

뉴스 데이터의 구조화된 저장과 안정적인 트랜잭션 처리를 위해 MySQL을 메인 데이터베이스로 채택하였으며, Kubernetes 환경의 특성인 'Pod의 휘발성'을 극복하기 위한 영속성 전략을 중점적으로 구현하였다.

##### b. 상세 기술 구현

###### 1) PVC(Persistent Volume Claim) 기반 스토리지 바인딩

```
root@vm1:~# kubectl get pvc -n news-db
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS  AGE
mysql-pvc     Bound     pvc-53552fe7-a18f-4465-ad8d-b26ebb0e9068  1Gi        RWO            local-path
<unset>      3d20h
```

- Pod가 재시작되거나 재생성되더라도 데이터가 유실되지 않도록 PVC(Persistent Volume Claim)을 1Gi 용량으로 설정하여 MySQL Pod에 마운트하였다. 이를 통해 데이터베이스 파일이 노드의 로컬 디스크가 아닌 별도의 영구 스토리지(PV)에 저장되도록 하여 데이터의 생명주기를 Pod와 분리하고 영구 보존성을 확보하였다.

###### 2) Character Set 최적화:

- 한글 뉴스 데이터의 깨짐 방지와 이모지(Emoji) 등의 특수 문자 저장을 완벽하게 지원하기 위해 데이터베이스의 Charset을 utf8mb4, Collation을 utf8mb4\_unicode\_ci로 강제 설정하여 데이터 정합성을 유지하였다.

#### 4.3.2. NoSQL: Redis를 활용한 고성능 캐싱 레이어

##### a. 사용 이유

반복적인 DB 조회로 인한 I/O 병목을 해소하고 응답 속도를 획기적으로 개선하기 위해 Redis(Remote Dictionary Server)를 도입하여 캐시 계층을 구성하였다.

##### b. 상세 기술 구현

###### 1) In-Memory Caching 전략:

- 서비스 접속 시 가장 먼저 노출되는 메인 화면에서 뉴스 목록을 필수적으로 호출해야 하는 구조적 특성을 고려하여 최적화를 진행하였다. 매번 디스크 기반의 **MySQL**을 조회하는 비효율을 제거하기 위해 메모리 기반의 **Redis**를 활용한 **Look-aside** 패턴을 적용하였다. 이를 통해 메인 화면 로딩 시 발생하는 **DB** 부하를 최소화하고 **API** 응답 속도를 밀리세컨드(**ms**) 단위로 단축시켰다.

## 2) 데이터 만료(TTL) 관리:

- 실시간성이 중요한 뉴스 데이터의 특성을 고려하여 캐시 데이터에 **TTL(Time-To-Live)**을 설정함으로써, 오래된 데이터가 자동으로 소멸되고 최신 데이터가 갱신되도록 하여 메모리 효율성을 최적화하였다.

### 4.3.3. Kubernetes Secrets를 이용한 보안 관리

#### a. 사용 이유

데이터베이스 접속 정보, **OpenAI API key** 값과 같은 민감 데이터를 소스 코드나 **ConfigMap**에 평문으로 노출하지 않고 **Kubernetes**의 암호화된 저장소 객체인 **Secret**을 통해 관리하였다.

#### b. 상세 기술 구현

##### 1) 환경 변수 주입

```

root@vm1:~# kubectl get secret -n news-db
NAME          TYPE      DATA   AGE
db-secret     Opaque    1       3d23h
root@vm1:~# kubectl describe secret db-secret -n news-db
Name:         db-secret
Namespace:    news-db
Labels:       <none>
Annotations:  <none>

Type: Opaque

Data
====
MYSQL_PASSWORD:  9 bytes
root@vm1:~#

```

```

root@vm1:~# kubectl get secret -n news-app
NAME                TYPE      DATA   AGE
app-db-secret       Opaque    1       33h
chatgpt-api-secret  Opaque    1       33h
root@vm1:~# kubectl describe secret chatgpt-api-secret -n news-app
Name:         chatgpt-api-secret
Namespace:    news-app
Labels:       <none>
Annotations:  <none>

Type: Opaque

Data
====
OPENAI_API_KEY:  164 bytes
root@vm1:~# kubectl describe secret app-db-secret -n news-app
Name:         app-db-secret
Namespace:    news-app
Labels:       <none>
Annotations:  <none>

Type: Opaque

Data
====
MYSQL_PASSWORD:  9 bytes
root@vm1:~#

```

- MYSQL\_PASSWORD 등의 내부 시스템 비밀번호와 함께 외부 API 호출에 필요한 OPENAI\_API\_KEY를 Kubernetes Secret 리소스로 생성하였다.
- 이후 백엔드 Pod가 생성될 때 이 Secret 값을 참조하여 환경 변수 형태로 안전하게 주입되도록 구성하였다. 이를 통해 인가되지 않은 사용자가 컨테이너 내부나 소스 코드를 조회하더라도 실제 인증키 값을 탈취할 수 없도록 보안 레벨을 강화하였으며, API 키 변경 시에도 소스 코드 수정 없이 Secret 값만 갱신하여 운영 유연성을 확보하였다.

#### 4.3.4. ConfigMap을 활용한 환경 설정 분리

##### a. 사용 이유

애플리케이션 소스 코드 수정 없이 배포 환경(개발, 운영)에 따라 변경되는 설정값(DB 주소, 스키마 명 등)을 유연하게 관리하기 위해 **ConfigMap**을 사용하였다.

#### b. 상세 기술 구현

```
root@vm1:~# kubectl get cm -n news-app
NAME          DATA   AGE
app-config    3       33h
kube-root-ca.crt 1       33h
```

```
root@vm1:~# kubectl describe cm app-config -n news-app
Name:         app-config
Namespace:    news-app
Labels:       <none>
Annotations:  <none>

Data
====
MYSQL_DB:
----
news_db

MYSQL_HOST:
----
mysql-service.news-db

REDIS_HOST:
----
redis-service.news-db

BinaryData
====

Events:  <none>
root@vm1:~#
```

- 민감하지 않은 데이터 관리: 보안이 필요한 비밀번호는 **Secret**으로 관리하는 반면, **MYSQL\_HOST**, **REDIS\_HOST**와 같은 일반 설정값은 **ConfigMap** 객체에 정의하였다.
- 환경 변수 매핑: **Pod**가 생성될 때 **ConfigMap**의 데이터를 환경 변수로 주입하도록 설정하였다. 로컬 테스트 환경과 쿠버네티스 배포 환경의 코드를 다르게 작성할 필요 없이 **ConfigMap** 값만 교체하여 환경을 전환할 수 있다.
- 

#### 4.3.5. 내부 네트워크 격리 및 서비스 디스커버리



#### a. 사용 이유

데이터 계층(news-db)은 외부 인터넷망과 완전히 격리되어야 하므로, 오직 클러스터 내부 통신만 허용하는 네트워크 정책을 적용하였다.

#### b. 상세 기술 구현

##### 1) ClusterIP 모드 적용

- MySQL(3306)과 Redis(6379) 서비스 모두 ClusterIP 타입으로 설정하여 외부 IP(External IP)를 할당하지 않았다. 이는 외부 해킹 시도로부터 원천적으로 차단된 Private Network 환경을 제공한다.

##### 2) FQDN 기반 통신

- 백엔드 API 서버(news-app Namespace)가 DB에 접근할 때는 IP 주소가 아닌 Kubernetes DNS가 제공하는 FQDN(Fully Qualified Domain Name)(예: mysql-service.news-db.svc.cluster.local)을 사용하여 통신하도록 구현하였다. 이를 통해 Pod의 IP가 동적으로 변경되더라도 서비스 연결이 끊기지 않는 안정적인 연결성을 보장하였다.

## 5. 프로젝트 활용 방안

### 5.1. 개발자 취업 과정에서의 활용 방안

#### 5.1.1. 취업 준비의 효율성 극대화 및 시간 절감

본 서비스는 취업 준비 과정에서 발생하는 반복적이고 시간 할애가 필요한 문서 작성 시간을 획기적으로 단축시키는 데 중점을 둔다.

##### a. README 및 이력서 문서 자동 생성

OpenAI API 기반의 챗봇 기능을 활용해 사용자가 입력한 핵심 정보(프로젝트 내용, 경력 등)를 마크다운 형식의 이력서 및 README.md로 사용할 수 있도록 즉시 변환한다.

##### b. 효율적인 시간 활용

사용자들은 더 이상 문장 표현이나 서식 정리와 같은 부가적인 작업에 시간을 낭비하지 않고 본질적인 개발 역량 강화, 코딩 테스트 준비, 기술 면접 대비 등에 집중할 수 있게 되어 취업 준비의 전반적인 효율성을 극대화할 수 있다.

### 5.1.2. 제출 문서의 품질 향상 및 경쟁력 강화

AI를 활용하여 생성되는 문서는 전문적이고 표준화된 어투를 가져 사용자의 경쟁력을 한층 높여준다.

#### a. 전문적인 표현 사용

AI가 제공하는 정돈되고 논리적인 문서 구조와 표현을 통해 개발자들은 자신의 기술 스택과 경험을 보다 명확하고 설득력 있게 전달할 수 있다. 이는 인사 담당자 및 면접관에게 긍정적인 인상을 주는 데 기여할 수 있다.

#### b. 글로벌 취업 기회 확대

‘이력서 번역’ 기능을 통해 기존에 작성된 국문 이력서를 고품질의 영문 이력서로 즉시 전환함으로써 해외 취업 또는 외국계 기업 지원에 필요한 장벽을 낮추고 사용자의 취업 선택지를 글로벌 시장으로 확장하는 기반을 제공한다.

#### c. 높은 포트폴리오 완성도

프로젝트의 핵심 내용을 잘 요약하고 강조하는 README.md 자동 생성 기능은 포트폴리오의 완성도를 향상시켜 사용자의 기술적 성과를 효과적으로 보여줄 수 있다.

### 5.1.3. IT 기술 트렌드 및 시장 동향 파악

급변하는 IT 시장에서 개발자가 지속적으로 성장하고 경쟁력을 유지할 수 있도록 최신 정보를 제공한다.

#### a. 최신 정보 접근성 향상

매일 크롤링하여 선별된 주요 IT 뉴스 기사 3건을 서비스 상단에 배치함으로써 사용자는 별도의 검색 과정 없이도 가장 최신의 기술 트렌드, 산업 동향, 채용 시장의 키워드를 편리하게 파악할 수 있다.

#### b. 맞춤형 성장 지원

이러한 트렌드 파악은 개발자가 자신의 학습 방향과 기술 스택을 시장 요구에 맞춰 능동적으로 조정하고, 면접 시 기술 동향에 대한 이해도를 보여줄 수 있는 기반 지식을 제공한다.

## 5.2. 향후 기능 확장성

### 5.2.1. 로그인 기능

사용자별 문서 생성 이력과 설정을 저장하여 개인화된 서비스 제공이 가능하도록 한다.

### 5.2.2. 사용자 특화 프롬프트 제공

직무 및 경력 수준에 대한 정보를 미리 습득한 뒤 그에 맞춰 맞춤형 프롬프트를 적용한 답변을 제공하여 개인화된 AI 서비스로 확장할 수 있다.

### 5.2.3. 채팅 보관 기능

사용자가 원하는 과거 대화 내용 및 생성 결과를 문서함에 폴더별로 저장하여 추후에도 쉽게 찾아보고 활용이 가능하도록 한다.

### 5.2.4. 답변 만족도 조사 기능

챗봇 답변에 대한 사용자 피드백을 사용자가 서비스를 이용하는 동안 수집하여 프롬프트 엔지니어링 개선에 활용할 수 있다.

## 5.3. 다양한 사용자군에 대한 활용 가능성

### 5.3.1 학생 개발자

- a. 사용자 상황: 대학에서 컴퓨터 공학적 이론(CS) 지식은 습득했으나 현업에서 실제로 사용되는 최신 기술 트렌드나 아키텍처에 대한 이해가 부족하여 진로에 대한 막연함을 느끼는 단계다.
- b. 활용 방안
  - 실무 중심의 최신 트렌드 학습: 학부 커리큘럼에서 접하기 힘든 최신 프레임워크나 업계 표준 기술(Docker, K8s, MSA 등)에 관한 뉴스를 매일 큐레이션 받아봄으로써, 이론과 실무 사이의 간극을 줄이고 감각을 배양한다.
  - 개인화된 전공 학습 보조: 과제나 프로젝트 수행 중 마주하는 낯선 개념에 대해 범용 LLM(ChatGPT, Gemini)을 직접 사용하는 것보다 최적화된 '전용 프롬프트'를 통해 학습한다. 이를 통해 맥락에 맞는 정확한 개념 설명과 예제 코드를 즉각적으로 제공받아 학습 효율을 극대화한다.
- c. 기대 효과: 학업과 병행 가능한 효율적인 정보 습득 루틴을 형성하여 졸업 시점에 현업이 요구하는 기술 스택을 겸비한 '준비된 인재'로 성장할 수 있다.

## 2) 전직·이직 준비생 및 해외 취업 희망자

- a. 사용자 상황: 비개발 직군에서 개발직으로 전직을 시도하거나 국내를 넘어 해외 빅테크 기업 취업을 목표로 하는 상황이다. 글로벌 기술 동향 파악과 영문 이력서 준비에 대한 부담을 가지고 있다.
- b. 활용 방안
  - 이력서 생성 및 IT 전문 번역: 일반 번역기를 통한 기계적인 번역이 아닌, IT 업계에서 통용되는 전문 용어와 뉘앙스를 반영한 이력서 번역 기능을 제공한다. 이를 통해 국문/영문 이력서 간의 자연스러운 변환을 지원하고 글로벌 표준에 맞는 포맷을 갖추도록 돕는다.
- c. 기대 효과: 해외 기업 지원 시 필수적인 기술 용어의 정확한 사용을 보장하여 서류 경쟁력을 확보할 수 있으며, 최신 기술 트렌드가 반영된 포트폴리오 구성을 통해 면접관에게 깊이 있는 인상을 남길 수 있다.

### 3) 부트캠프 참가자

- a. 사용자 상황: 단기간에 방대한 양의 기술을 소화해야 하며, 매일 쏟아지는 과제와 팀 프로젝트로 인해 절대적인 시간이 부족하다. 특히 멘토나 강사가 없는 심야 시간대에 발생하는 에러 해결에 어려움을 겪는다.
- b. 활용 방안
  - 24시간 에러 해결 멘토링: 프로젝트 진행 중 발생하는 버그나 에러 로그를 챗봇에 입력하여 즉각적인 원인 분석과 해결책을 제시받는다. 이를 통해 멘토의 부재 시에도 중단 없는 개발을 이어간다.
  - 프로젝트 문서화(README) 자동화: 부트캠프 기간 동안 수행한 수많은 프로젝트를 일일이 정리하기 힘든 상황에서, 'README.md 자동 생성 기능'을 활용한다. 소스 코드를 기반으로 프로젝트 개요, 설치 방법, 기술 스택 등을 일관성 있게 정리해주어 즉시 포트폴리오로 활용할 수 있도록 돕는다.
- c. 기대 효과: 단순 정보 검색이나 문서 작성에 소요되는 비생산적인 시간을 획기적으로 단축하여 코딩과 핵심 로직 구현에 집중할 수 있다. 결과적으로 가파른 러닝 커브를 효과적으로 극복하고 양질의 프로젝트 결과물을 산출한다.

## 6. 결론 및 향후 계획

### 6.1. 프로젝트 전체 요약

본 프로젝트는 개발자 취업 준비 과정에서 반복적이고 비효율적인 문서 작성 문제를 해결하기 위해 ChatGPT API와 Kubernetes 기반 클라우드 아키텍처를 활용한 개발자 특화 AI 취업 도우미 서비스를 구현하였다.

IT 주요 뉴스 제공, 개발자 맞춤 챗봇, README 생성, 이력서 생성 및 번역 기능을 통해 사용자가 문서 작업 부담을 줄이고 핵심 역량 개발에 집중할 수 있는 환경을 제공하였다. 또한 Docker와 Kubernetes를 활용하여 확장성과 운영 효율성을 고려한 시스템을 설계하고 구현했다.

## 6.2. 향후 시스템 확장 가능성

본 시스템은 컨테이너 환경 기반으로 설계되어 클라우드 서비스 플랫폼 환경에 유연하게 배포할 수 있으며, CI/CD 자동화를 통해 코드 변경 시 자동 빌드 및 배포가 가능하도록 확장할 수 있다. 또한 현재 사용중인 port-forwarding 방식 대신 Ingress를 활용함으로써 외부 요청을 보다 안정적으로 처리할 수 있으며, 내부 서비스는 분리하여 보안성과 운영 효율성을 함께 고려한 확장 가능한 구조로 발전할 수 있다.

## 6.3. 한계점 및 개선 방향

본 프로젝트는 제한된 개발 기간과 학습 목적에 맞추어 핵심 기능 구현과 클라우드 기반 아키텍처 설계에 중점을 두었다. 이에 사용자 인증을 통한 개인화 서비스나 데이터 기반 추천 기능과 같은 확장된 기능까지 구현을 하지 못한 한계가 있다.

향후에는 로그인 기능 및 사용자 친화적인 기능들을 추가로 도입할 예정이다. 또한 UI 개선과 함께 코드 리뷰, 채용 공고 분석 등 고도화된 취업 지원 기능을 추가적으로 고려해볼 수 있다. 이러한 모든 기능들에 대한 사용자 피드백 및 챗봇 답변 만족도 데이터를 기반으로 프롬프트 또한 사용자에게 적합하도록 고도화함으로써 보다 개인화되고 신뢰도 높은 서비스로 개선할 수 있을 것이다.

## 7. 참고자료

| 파일명             | 링크  |
|-----------------|---|
| FE Docker image | <a href="https://github.com/25-2-CloudComputingSystems-Team11/front/pkgs/container/front">https://github.com/25-2-CloudComputingSystems-Team11/front/pkgs/container/front</a> |
| FE Code         | <a href="https://github.com/25-2-CloudComputingSystems-Team11/front">https://github.com/25-2-CloudComputingSystems-Team11/front</a>   |

|                 |   |
|-----------------|---|
| BE Docker image | <a href="https://github.com/25-2-CloudComputingSystems-Team11/news-crawler/pkgs/container/news-crawler">https://github.com/25-2-CloudComputingSystems-Team11/news-crawler/pkgs/container/news-crawler</a> |
| BE Code         | <a href="https://github.com/25-2-CloudComputingSystems-Team11/news-crawler">https://github.com/25-2-CloudComputingSystems-Team11/news-crawler</a>   |
| manifest        | <a href="https://github.com/25-2-CloudComputingSystems-Team11/manifest">https://github.com/25-2-CloudComputingSystems-Team11/manifest</a>   |