AIRISS API 배포 완전 가이드

📋 사전 준비사항 체크리스트

✅ 필수 요구사항

- [ ] Python 3.9 이상
- [ ] OpenAI API 키 (유료 계정 권장)
- [ ] 서버 또는 클라우드 인스턴스
- [ ] 도메인 (선택사항, 운영환경에서 권장)
- [ ] SSL 인증서 (선택사항, HTTPS 사용시)

🔧 시스템 요구사항

- **CPU**: 최소 2코어 (권장 4코어)
- **RAM**: 최소 4GB (권장 8GB)
- **디스크**: 최소 20GB (결과 파일 저장용)
- **네트워크**: 안정적인 인터넷 연결

📦 1단계: 개발 환경 설정

1.1 Python 가상환경 생성

bash

```
# Windows
python -m venv airiss_env
airiss_env\Scripts\activate

# macOS/Linux
python3 -m venv airiss_env
source airiss_env/bin/activate
```

1.2 필수 패키지 설치

bash

```
# requirements.txt 파일 생성
cat > requirements.txt << EOF
fastapi==0.104.1
uvicorn[standard]==0.24.0
pandas==2.1.3
openpyxl==3.1.2
openai==1.3.5
aiofiles==23.2.1
python-multipart==0.0.6
pydantic==2.5.0
python-jose[cryptography]==3.3.0
passlib[bcrypt]==1.7.4
structlog==23.2.0
prometheus-client==0.19.0
psutil==5.9.6
EOF

# 패키지 설치
pip install -r requirements.txt
```

1.3 프로젝트 구조 생성

bash

```
mkdir airiss_api
cd airiss_api

# 디렉토리 구조 생성
mkdir -p {app,config,logs,results,tests}

# 파일 구조
touch app/__init__.py
touch app/main.py
touch config/settings.py
touch config/__init__.py
touch .env
touch .env.example
touch Dockerfile
touch docker-compose.yml
touch README.md
```

🔑 2단계: 환경변수 설정

2.1 OpenAI API 키 발급

1. **OpenAI 계정 생성**
   - https://platform.openai.com 접속
   - 계정 생성 및 로그인
2. **API 키 생성**

```bash
# OpenAI 대시보드 → API Keys → Create new secret key
# 예시: sk-proj-abc123def456...
```

3. **크레딧 충전**

```bash
# Billing → Add payment method
# 최소 $10 충전 권장 (1000건 분석 기준)
```

2.2 환경변수 파일 생성

```bash
# .env 파일 생성
cat > .env << 'EOF'
# OpenAI 설정
OPENAI_API_KEY=sk-proj-your-actual-api-key-here

# 보안 설정
API_SECRET_TOKEN=airiss-super-secret-token-2025

# 서버 설정
HOST=0.0.0.0
PORT=8000
WORKERS=1

# 파일 처리 설정
MAX_FILE_SIZE=10  # MB 단위
BATCH_SIZE=3
API_TIMEOUT=30

# 로그 설정
LOG_LEVEL=INFO
LOG_FILE=logs/airiss.log

# 운영 환경 설정
ENVIRONMENT=development
DEBUG=true
EOF
```

2.3 예시 환경변수 파일 생성

bash

```
# .env.example 파일 (깃허브 공유용)
cat > .env.example << 'EOF'
# OpenAI 설정
OPENAI_API_KEY=sk-proj-your-openai-api-key

# 보안 설정
API_SECRET_TOKEN=your-secret-token

# 서버 설정
HOST=0.0.0.0
PORT=8000
WORKERS=1

# 파일 처리 설정
MAX_FILE_SIZE=10
BATCH_SIZE=3
API_TIMEOUT=30

# 로그 설정
LOG_LEVEL=INFO
LOG_FILE=logs/airiss.log

# 운영 환경 설정
ENVIRONMENT=development
DEBUG=true
EOF
```

⚙️ 3단계: 설정 파일 생성

3.1 config/settings.py

python

```python
# config/settings.py
import os
from pathlib import Path
from typing import Optional

class Settings:
    # 기본 설정
    PROJECT_NAME: str = "AIRISS API"
    VERSION: str = "1.0.0"
    DESCRIPTION: str = "AI 기반 직원 성과/역량 스코어링 시스템"

    # 서버 설정
    HOST: str = os.getenv("HOST", "0.0.0.0")
    PORT: int = int(os.getenv("PORT", "8000"))
    WORKERS: int = int(os.getenv("WORKERS", "1"))

    # OpenAI 설정
    OPENAI_API_KEY: str = os.getenv("OPENAI_API_KEY", "")
    OPENAI_MODEL: str = os.getenv("OPENAI_MODEL", "gpt-3.5-turbo")

    # 보안 설정
    API_SECRET_TOKEN: str = os.getenv("API_SECRET_TOKEN", "")

    # 파일 처리 설정
    MAX_FILE_SIZE: int = int(os.getenv("MAX_FILE_SIZE", "10")) * 1024 * 1024
    BATCH_SIZE: int = int(os.getenv("BATCH_SIZE", "3"))
    API_TIMEOUT: int = int(os.getenv("API_TIMEOUT", "30"))

    # 디렉토리 설정
    BASE_DIR: Path = Path(__file__).parent.parent
    RESULTS_DIR: Path = BASE_DIR / "results"
    LOGS_DIR: Path = BASE_DIR / "logs"

    # 로그 설정
    LOG_LEVEL: str = os.getenv("LOG_LEVEL", "INFO")
    LOG_FILE: str = os.getenv("LOG_FILE", "logs/airiss.log")

    # 운영 환경 설정
    ENVIRONMENT: str = os.getenv("ENVIRONMENT", "development")
    DEBUG: bool = os.getenv("DEBUG", "false").lower() == "true"

    def __init__(self):
        # 필수 디렉토리 생성
        self.RESULTS_DIR.mkdir(exist_ok=True)
        self.LOGS_DIR.mkdir(exist_ok=True)
```

```
        # 필수 환경변수 검증
        if not self.OPENAI_API_KEY:
            raise ValueError("OPENAI_API_KEY 환경변수가 필요합니다")

        if not self.API_SECRET_TOKEN:
            raise ValueError("API_SECRET_TOKEN 환경변수가 필요합니다")

# 전역 설정 인스턴스
settings = Settings()
```

3.2 로깅 설정 파일

python

```python
# config/logging_config.py
import logging
import logging.handlers
from pathlib import Path
from config.settings import settings

def setup_logging():
    """로깅 설정"""

    # 로그 디렉토리 생성
    settings.LOGS_DIR.mkdir(exist_ok=True)

    # 로거 생성
    logger = logging.getLogger("airiss")
    logger.setLevel(getattr(logging, settings.LOG_LEVEL))

    # 포매터 생성
    formatter = logging.Formatter(
        '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
    )

    # 파일 핸들러 (회전 로그)
    file_handler = logging.handlers.RotatingFileHandler(
        settings.LOG_FILE,
        maxBytes=10*1024*1024,  # 10MB
        backupCount=5
    )
    file_handler.setFormatter(formatter)
    logger.addHandler(file_handler)

    # 콘솔 핸들러
    console_handler = logging.StreamHandler()
    console_handler.setFormatter(formatter)
    logger.addHandler(console_handler)

    return logger
```

🐳 4단계: Docker 설정

4.1 Dockerfile 생성

dockerfile

```
# Dockerfile
FROM python:3.11-slim

# 작업 디렉토리 설정
WORKDIR /app

# 시스템 패키지 업데이트 및 필수 패키지 설치
RUN apt-get update && apt-get install -y \
    gcc \
    g++ \
    curl \
    && rm -rf /var/lib/apt/lists/*

# Python 의존성 파일 복사 및 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사
COPY . .

# 디렉토리 생성
RUN mkdir -p logs results

# 포트 노출
EXPOSE 8000

# 헬스체크 추가
HEALTHCHECK --interval=30s --timeout=30s --start-period=5s --retries=3 \
    CMD curl -f http://localhost:8000/health || exit 1

# 애플리케이션 실행
CMD ["python", "app/main.py"]
```

4.2 docker-compose.yml 생성

yaml

```yaml
# docker-compose.yml
version: '3.8'

services:
  airiss-api:
    build: .
    container_name: airiss-api
    ports:
      - "8000:8000"
    volumes:
      - ./results:/app/results
      - ./logs:/app/logs
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
      - API_SECRET_TOKEN=${API_SECRET_TOKEN}
      - HOST=0.0.0.0
      - PORT=8000
      - WORKERS=1
      - MAX_FILE_SIZE=10
      - BATCH_SIZE=3
      - API_TIMEOUT=30
      - LOG_LEVEL=INFO
      - ENVIRONMENT=production
      - DEBUG=false
    restart: unless-stopped
    networks:
      - airiss-network

  # Redis (선택사항 - 캐싱용)
  redis:
    image: redis:7-alpine
    container_name: airiss-redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    restart: unless-stopped
    networks:
      - airiss-network

  # Nginx (선택사항 - 리버스 프록시)
  nginx:
    image: nginx:alpine
    container_name: airiss-nginx
    ports:
      - "80:80"
```

```
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/ssl:/etc/nginx/ssl
    depends_on:
      - airiss-api
    restart: unless-stopped
    networks:
      - airiss-network

volumes:
  redis_data:

networks:
  airiss-network:
    driver: bridge
```

4.3 Nginx 설정 (선택사항)

bash

```
# nginx/nginx.conf 생성
mkdir -p nginx
cat > nginx/nginx.conf << 'EOF'
events {
    worker_connections 1024;
}

http {
    upstream airiss_api {
        server airiss-api:8000;
    }

    # HTTP → HTTPS 리다이렉트
    server {
        listen 80;
        server_name your-domain.com;
        return 301 https://$server_name$request_uri;
    }

    # HTTPS 서버
    server {
        listen 443 ssl http2;
        server_name your-domain.com;

        # SSL 설정
        ssl_certificate /etc/nginx/ssl/cert.pem;
        ssl_certificate_key /etc/nginx/ssl/key.pem;

        # 보안 헤더
        add_header X-Frame-Options DENY;
        add_header X-Content-Type-Options nosniff;
        add_header X-XSS-Protection "1; mode=block";

        # 파일 업로드 크기 제한
        client_max_body_size 50M;

        location / {
            proxy_pass http://airiss_api;
            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;

            # 타임아웃 설정
            proxy_connect_timeout 60s;
            proxy_send_timeout 60s;
```

```
            proxy_read_timeout 60s;
        }

        # API 문서 접근 제한 (운영환경)
        location /docs {
            allow 192.168.1.0/24;  # 사내 IP만 허용
            deny all;
            proxy_pass http://airiss_api;
        }
    }
}
EOF
```

🚀 5단계: 로컬 테스트

5.1 기본 테스트

bash

```
# 1. 환경변수 로드
source .env  # Linux/macOS
# 또는 .env 파일의 내용을 수동으로 설정

# 2. 애플리케이션 실행
python app/main.py

# 3. 다른 터미널에서 테스트
curl http://localhost:8000/health
```

5.2 API 테스트

bash

```bash
# 1. 헬스체크
curl -X GET "http://localhost:8000/health"

# 2. API 문서 접근
open http://localhost:8000/docs

# 3. 테스트 파일 업로드 (샘플 CSV 생성)
cat > test_data.csv << 'EOF'
이름,부서,의견
김철수,개발팀,성실하고 책임감이 강함. 기술적 역량이 뛰어나지만 커뮤니케이션 개선 필요
이영희,마케팅팀,창의적이고 적극적임. 팀워크가 좋고 고객 응대 능력이 우수함
박민수,인사팀,꼼꼼하고 신중함. 업무 처리 속도를 높일 필요가 있음
EOF

# 4. 파일 업로드 테스트
curl -X POST "http://localhost:8000/upload" \
  -H "Authorization: Bearer airiss-super-secret-token-2025" \
  -F "file=@test_data.csv"
```

5.3 Docker 테스트

bash

```bash
# 1. Docker 이미지 빌드
docker build -t airiss-api .

# 2. 컨테이너 실행
docker run -d \
  --name airiss-api-test \
  -p 8000:8000 \
  -e OPENAI_API_KEY="your-api-key" \
  -e API_SECRET_TOKEN="your-secret-token" \
  -v $(pwd)/results:/app/results \
  -v $(pwd)/logs:/app/logs \
  airiss-api

# 3. 로그 확인
docker logs airiss-api-test

# 4. 컨테이너 정지/삭제
docker stop airiss-api-test
docker rm airiss-api-test
```

☁️ 6단계: 클라우드 배포

6.1 AWS EC2 배포

bash

```bash
# 1. EC2 인스턴스 생성 (Ubuntu 22.04 LTS 권장)
# 인스턴스 타입: t3.medium 이상
# 보안 그룹: 22(SSH), 80(HTTP), 443(HTTPS), 8000(API) 포트 열기

# 2. 서버 접속
ssh -i your-key.pem ubuntu@your-ec2-ip

# 3. 기본 패키지 설치
sudo apt update && sudo apt upgrade -y
sudo apt install -y docker.io docker-compose git nginx certbot python3-certbot-nginx

# 4. Docker 권한 설정
sudo usermod -aG docker ubuntu
newgrp docker

# 5. 프로젝트 클론
git clone https://github.com/your-repo/airiss-api.git
cd airiss-api

# 6. 환경변수 설정
cp .env.example .env
nano .env  # 실제 값으로 수정

# 7. 배포
docker-compose up -d

# 8. SSL 인증서 설정 (도메인이 있는 경우)
sudo certbot --nginx -d your-domain.com
```

6.2 Google Cloud Platform 배포

bash

```bash
# 1. GCP 프로젝트 생성 및 gcloud CLI 설치

# 2. Compute Engine 인스턴스 생성
gcloud compute instances create airiss-api-vm \
  --zone=asia-northeast3-a \
  --machine-type=e2-medium \
  --boot-disk-size=30GB \
  --boot-disk-type=pd-standard \
  --image-family=ubuntu-2204-lts \
  --image-project=ubuntu-os-cloud \
  --tags=http-server,https-server

# 3. 방화벽 규칙 생성
gcloud compute firewall-rules create allow-api-port \
  --allow tcp:8000 \
  --source-ranges 0.0.0.0/0 \
  --description "Allow API port 8000"

# 4. 인스턴스 접속 및 배포
gcloud compute ssh airiss-api-vm --zone=asia-northeast3-a
# 이후 과정은 EC2와 동일
```

6.3 Heroku 배포

bash

```bash
# 1. Heroku CLI 설치 및 로그인
heroku login

# 2. Heroku 앱 생성
heroku create airiss-api-prod

# 3. 환경변수 설정
heroku config:set OPENAI_API_KEY="your-api-key"
heroku config:set API_SECRET_TOKEN="your-secret-token"
heroku config:set ENVIRONMENT="production"

# 4. Procfile 생성
echo "web: python app/main.py" > Procfile

# 5. 배포
git add .
git commit -m "Deploy to Heroku"
git push heroku main

# 6. 앱 열기
heroku open
```

🔒 7단계: 보안 강화

7.1 API 키 보안

bash

```bash
# 1. AWS Secrets Manager 사용 (AWS 배포시)
aws secretsmanager create-secret \
  --name "airiss/openai-api-key" \
  --secret-string "your-openai-api-key"

# 2. 환경변수 암호화
# HashiCorp Vault 또는 Azure Key Vault 사용 권장

# 3. API 키 로테이션 설정
# 정기적으로 API 키 변경하는 스크립트 작성
```

7.2 네트워크 보안

bash

```bash
# 1. 방화벽 설정 (UFW 예시)
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 8000  # 개발환경에서만

# 2. fail2ban 설치 (무차별 대입 공격 방지)
sudo apt install fail2ban
sudo systemctl enable fail2ban

# 3. VPN 설정 (사내망에서만 접근)
# WireGuard 또는 OpenVPN 사용 권장
```

📊 8단계: 모니터링 설정

8.1 기본 모니터링

bash

```bash
# 1. Prometheus + Grafana 설치
cat > monitoring/docker-compose.yml << 'EOF'
version: '3.8'
services:
  prometheus:
    image: prom/prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin123
EOF

# 2. Uptime 모니터링
# UptimeRobot 또는 Pingdom 사용 권장
```

8.2 로그 모니터링

bash

```bash
# 1. ELK Stack 설정 (선택사항)
# Elasticsearch + Logstash + Kibana

# 2. 간단한 로그 모니터링
# 로그 파일 크기 체크 스크립트
cat > scripts/log_monitor.sh << 'EOF'
#!/bin/bash
LOG_FILE="/app/logs/airiss.log"
MAX_SIZE=100  # MB

if [ -f "$LOG_FILE" ]; then
    size=$(du -m "$LOG_FILE" | cut -f1)
    if [ $size -gt $MAX_SIZE ]; then
        echo "$(date): 로그 파일이 ${size}MB입니다. 확인이 필요합니다." >> /app/logs/monitor.log
    fi
fi
EOF

chmod +x scripts/log_monitor.sh

# 3. 크론탭 등록
echo "0 */6 * * * /app/scripts/log_monitor.sh" | crontab -
```

🚀 9단계: 성능 최적화

9.1 서버 최적화

bash

```bash
# 1. 시스템 리소스 최적화
# /etc/sysctl.conf 설정
echo "vm.swappiness=10" >> /etc/sysctl.conf
echo "net.core.rmem_max=134217728" >> /etc/sysctl.conf

# 2. 파일 디스크립터 제한 증가
echo "* soft nofile 65536" >> /etc/security/limits.conf
echo "* hard nofile 65536" >> /etc/security/limits.conf

# 3. Nginx 튜닝 (프록시 사용시)
cat > nginx/nginx_optimized.conf << 'EOF'
worker_processes auto;
worker_connections 2048;

http {
    # 성능 최적화
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;

    # Gzip 압축
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css application/json application/javascript;

    # 캐싱 설정
    location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
EOF
```

9.2 애플리케이션 최적화

python

```python
# config/optimization.py
import asyncio
from functools import lru_cache

class OptimizationConfig:
    # 연결 풀 설정
    HTTP_POOL_SIZE = 100
    HTTP_POOL_MAXSIZE = 100

    # 캐시 설정
    CACHE_TTL = 3600  # 1시간
    MAX_CACHE_SIZE = 1000

    # 비동기 처리 설정
    MAX_CONCURRENT_TASKS = 10
    BATCH_TIMEOUT = 300  # 5분

# 캐시 데코레이터 사용 예시
@lru_cache(maxsize=1000)
def get_cached_analysis(text_hash: str):
    """분석 결과 캐싱"""
    pass
```

✅ 10단계: 배포 체크리스트

배포 전 최종 확인사항

bash

```bash
# 1. 환경변수 확인
echo "✅ OpenAI API 키: $(echo $OPENAI_API_KEY | cut -c1-10)..."
echo "✅ API 토큰: $(echo $API_SECRET_TOKEN | cut -c1-10)..."

# 2. 의존성 확인
pip freeze > deployed_requirements.txt
echo "✅ Python 패키지 설치 완료"

# 3. 포트 확인
netstat -tulpn | grep :8000
echo "✅ 포트 8000 사용 가능"

# 4. 디렉토리 권한 확인
ls -la logs/ results/
echo "✅ 디렉토리 권한 설정 완료"

# 5. API 테스트
curl -f http://localhost:8000/health || echo "❌ 헬스체크 실패"
echo "✅ API 헬스체크 통과"

# 6. 로그 파일 확인
tail -n 10 logs/airiss.log
echo "✅ 로그 정상 작동"

# 7. 보안 점검
echo "✅ API 키 환경변수 설정"
echo "✅ 인증 토큰 설정"
echo "✅ 방화벽 설정"

echo "🎉 배포 준비 완료!"
```

배포 후 모니터링

bash

```bash
# 1. 서비스 상태 모니터링 스크립트
cat > scripts/health_monitor.sh << 'EOF'
#!/bin/bash
API_URL="http://localhost:8000"
SLACK_WEBHOOK="your-slack-webhook-url"

# 헬스체크
response=$(curl -s -o /dev/null -w "%{http_code}" $API_URL/health)

if [ $response -eq 200 ]; then
    echo "$(date): API 정상 작동"
else
    echo "$(date): API 오류 - HTTP $response"
    # Slack 알림 (선택사항)
    curl -X POST -H 'Content-type: application/json' \
        --data '{"text":"AIRISS API 오류 발생: HTTP '$response'"}' \
        $SLACK_WEBHOOK
fi
EOF

# 2. 크론탭으로 5분마다 체크
echo "*/5 * * * * /app/scripts/health_monitor.sh" | crontab -

# 3. 리소스 모니터링
cat > scripts/resource_monitor.sh << 'EOF'
#!/bin/bash
# CPU, 메모리, 디스크 사용량 체크
cpu=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | awk -F% '{print $1}')
memory=$(free | grep Mem | awk '{printf("%.1f", $3/$2 * 100.0)}')
disk=$(df -h / | awk 'NR==2{printf "%s", $5}')

echo "$(date): CPU: ${cpu}%, Memory: ${memory}%, Disk: ${disk}"

# 임계값 체크 (CPU 80%, Memory 85%, Disk 90%)
if (( $(echo "$cpu > 80" | bc -l) )); then
    echo "⚠️  CPU 사용률 높음: ${cpu}%"
fi
EOF

echo "✅ 배포 완료 및 모니터링 설정 완료!"
```

이제 **AIRISS API가 완전히 배포 준비 완료**되었습니다! 🎉

각 단계를 차근차근 따라하시면 **프로덕션 환경에서 안정적으로 운영**할 수 있는 엔터프라이즈급 API 서비스를 구축할 수 있습니다.