DGX Spark 완전 셋업 가이드북

First Boot ~ 개발 환경 구축 (초기화 ~ 서비스 기동)

목차

개요

DGX Spark는 NVIDIA Grace Blackwell 기반의 개인용 AI 슈퍼컴퓨터입니다. 부팅 직후 기본 운영 환경(Ubuntu 24.04 LTS ARM64)이 설정되어 있으며, 그 위에 개발/실험 환경을 순차적으로 구성합니다.

주요 특징:

- **기본 제공**: CUDA 12.x, Docker, Container Toolkit
- **기본 미제공**: Conda/Miniconda (따라서 Micromamba 사용)
- **원격 접속**: NVIDIA Sync를 통한 자동 SSH 터널·포트포워딩
- **CADD 워크플로우**: Vina, smina, GNINA 등 GPU 가속 지원

총 소요 시간: 약 30-45분 (네트워크 속도에 따라 가변)

0단계: 첫 부팅 및 접속

하드웨어 연결 및 전원 투입

연결 순서:

1. Ethernet 케이블 연결 (필수)
2. 전원 어댑터 연결
3. 자동 부팅 대기 (약 3-5분)

초기 설정 마법사 (First-Time Setup Wizard)

부팅 후 자동으로 실행되는 대화형 설정 프로세스:

설정 순서:

- 언어 및 시간대: Korean, Asia/Seoul
- 키보드 레이아웃: Korean
- 이용약관: I agree (필수)
- 사용자 계정: 사용자명 + 비밀번호 (8자 이상)
- 네트워크: Ethernet 또는 Wi-Fi 설정
- 시스템 업데이트: 진행 (재부팅 포함, 절대 전원 끄지 말 것)

완료 메시지:

```
✅ "Congratulations! Your DGX Spark is now ready to use."
```

1단계: 시스템 기본 점검

부팅 완료 후 터미널에서 기본 정보 확인:

```
# OS 정보
uname -a
lsb_release -a

# GPU 및 드라이버
nvidia-smi

# 디스크 여유 확인
df -h /
free -h

# 네트워크 연결 테스트
ping -c 1 8.8.8.8

# 호스트명 확인
hostname
```

2단계: Hostname 및 IP 주소 찾기

로컬에서 직접 확인 (모니터 연결 시)

```
hostname
# 출력 예: spark-a2d1

ip addr show | grep "inet " | grep -v 127.0.0.1
# 출력 예: inet 192.168.0.34/24 ...
```

다른 컴퓨터에서 탐색

Windows (PowerShell 관리자):

```
arp -a | findstr /i spark
ping spark.local
```

macOS/Linux:

```
avahi-browse -a | grep spark
ping spark.local
```

NVIDIA Sync 자동 발견 (가장 간편)

Sync 앱의 "Discover Devices on Network" 버튼 클릭 → 자동 리스트업

3단계: NVIDIA Sync 설치 및 연결

설치 방법

Windows/macOS: 공식 설치파일 실행

Linux (Ubuntu):

```
curl -fsSL https://workbench.download.nvidia.com/stable/linux/gpgkey | sudo tee -a /etc/a
echo "deb https://workbench.download.nvidia.com/stable/linux/debian default proprietary"
sudo apt update
sudo apt install nvidia-sync
```

기기 등록

NVIDIA Sync 앱 실행 후:

- "Add Device" 클릭
- **Name**: MyDGXSpark (자유)
- **Hostname or IP**: spark-a2d1.local 또는 192.168.0.34
- **Username**: 설정한 사용자명
- **Password**: 사용자 비밀번호
- "Connect" 클릭 → SSH 키 자동 생성·전송

4단계: 포트 충돌 방지 및 해결

기본 포트 맵

서비스	기본 포트	용도	충돌 가능성
SSH	22	원격 로그인	낮음
JupyterLab	8888	노트북 개발	중간
Docker Registry	5000	로컬 이미지	높음
MLflow	5000	실험 추적	**높음**
Prefect	4200	워크플로우 UI	낮음
MinIO API	9000	저장소	중간
MinIO Console	9001	관리 UI	낮음
Grafana	3000	모니터링	중간
Prometheus	9090	메트릭 수집	낮음

현재 열려 있는 포트 확인

```
sudo netstat -tulpn | grep LISTEN
sudo lsof -i :5000  # 특정 포트 확인
```

포트 충돌 해결

방법 1: Docker Compose에서 포트 변경 (권장)

```
mlflow:
  ports:
    - "5500:5000"  # 호스트 5500 → 컨테이너 5000
```

방법 2: 기존 서비스 포트 해제

```
sudo fuser -k 5000/tcp
sudo systemctl stop docker-registry
```

방법 3: NVIDIA Sync Custom Ports

- Sync 앱의 "Custom Ports"에서 로컬/원격 포트 매핑

5단계: OS 환경 업데이트

패키지 업데이트

```
sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
```

필수 개발 도구 설치

```
sudo apt install -y \
    build-essential cmake git curl wget \
    pkg-config libssl-dev libffi-dev \
    python3-pip python3-venv python3-dev \
    libhdf5-dev libopenblas-dev gfortran
```

6단계: Docker 및 GPU 런타임 설정

Docker 및 GPU 런타임 확인

```
docker --version
docker info | grep -i nvidia
cat /etc/docker/daemon.json
```

GPU 런타임 테스트

```
docker run --rm --gpus all nvcr.io/nvidia/cuda:12.6.2-base-ubuntu22.04 nvidia-smi
```

✅ GPU 메모리 표시 → 정상
❌ 표시 안 됨 → Docker 재시작

```
sudo systemctl restart docker
```

7단계: Micromamba 환경 구축

Micromamba 선택 이유

항목	Micromamba	Conda	시스템 Python
설치 크기	13MB	300MB+	기본 설치
속도	⚡ 빠름	느림	N/A
컨테이너 최적화	✅	⚠️	✅
ARM64 지원	✅	✅ 제한적	✅
CADD 도구	✅	✅	❌

설치

```
curl -Ls https://micro.mamba.pm/install.sh | bash
source ~/.bashrc
micromamba --version
```

기본 설정

```
export WS=/srv/dgx_ws
export MAMBA_ROOT_PREFIX=$WS/micromamba

echo "export WS=/srv/dgx_ws" &gt;&gt; ~/.bashrc
echo "export MAMBA_ROOT_PREFIX=$WS/micromamba" &gt;&gt; ~/.bashrc
source ~/.bashrc
```

8단계: Workspace 디렉토리 구성

디렉토리 생성

```
export WS=/srv/dgx_ws

sudo mkdir -p $WS/{compose,config,monitoring,scripts,storage,ssl,templates}
sudo chown -R $USER:$USER $WS

mkdir -p $WS/storage/{postgres,minio,prefect,portainer,grafana,prometheus,spark}
mkdir -p $WS/monitoring/{prometheus,grafana,alertmanager}
mkdir -p $WS/scripts/{deploy,setup,backup}
```

최종 레이아웃

```
/srv/dgx_ws/
├── compose/         # Docker Compose YAML
├── config/          # 환경 설정
├── monitoring/      # 모니터링 설정
├── scripts/         # 자동화 스크립트
├── storage/         # 데이터 볼륨
├── ssl/             # SSL 인증서
└── templates/       # 템플릿 파일
```

9단계: 외장 데이터 마운트 및 환경 복원

외장 디스크 마운트

```
# 1. 외장 디스크 확인
sudo lsblk

# 2. 마운트 포인트 생성
sudo mkdir -p /mnt/lhe_tmp_data

# 3. 마운트
sudo mount /dev/sda1 /mnt/lhe_tmp_data

# 4. 영구 마운트 설정
sudo bash -c 'echo "/dev/sda1 /mnt/lhe_tmp_data auto noatime 0 0" &gt;&gt; /etc/fstab'
```

데이터 링크 또는 복사

```
export WS=/srv/dgx_ws

# 심볼릭 링크 (권장)
ln -s /mnt/lhe_tmp_data/CADD_chemflow_g $WS/storage/CADD_chemflow_g

# 또는 완전 복사
rsync -avh --progress /mnt/lhe_tmp_data/CADD_chemflow_g/ $WS/storage/CADD_chemflow_g/
```

Micromamba 환경 복원

```
export WS=/srv/dgx_ws

micromamba create -n msmd -f $WS/storage/CADD_chemflow_g/env_store/envs/msmd.yaml -y
micromamba create -n dock -f $WS/storage/CADD_chemflow_g/env_store/envs/dock.yaml -y
micromamba create -n dais -f $WS/storage/CADD_chemflow_g/env_store/envs/dais.yaml -y

# 확인
micromamba env list
```

```
micromamba activate msmd
gnina --version
```

10단계: 핵심 서비스 구성

환경 변수 파일

```
export WS=/srv/dgx_ws

cat &gt; $WS/config/.env &lt;&lt;'EOF'
POSTGRES_USER=prefect
POSTGRES_PASSWORD=prefectpw
POSTGRES_DB=prefect

MINIO_ROOT_USER=admin
MINIO_ROOT_PASSWORD=minio123
MINIO_BUCKET=prefect-artifacts

MLFLOW_TRACKING_URI=http://mlflow:5000
PREFECT_API_URL=http://localhost:4200/api

AWS_ACCESS_KEY_ID=admin
AWS_SECRET_ACCESS_KEY=minio123
AWS_S3_BUCKET=prefect-artifacts
EOF
```

Docker Compose 파일 생성

```
export WS=/srv/dgx_ws

cat &gt; $WS/compose/stack.yml &lt;&lt;'YAML'
version: '3.8'

services:
  postgres:
    image: postgres:16
    environment:
      POSTGRES_USER: prefect
      POSTGRES_PASSWORD: prefectpw
      POSTGRES_DB: prefect
    volumes:
      - ${WS}/storage/postgres:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    restart: unless-stopped

  minio:
    image: quay.io/minio/minio:latest
    command: server /data --console-address ":9001"
    environment:
      MINIO_ROOT_USER: admin
```

```
      MINIO_ROOT_PASSWORD: minio123
    volumes:
      - ${WS}/storage/minio:/data
    ports:
      - "9000:9000"
      - "9001:9001"
    restart: unless-stopped

  prefect:
    image: prefecthq/prefect:3-latest
    command: prefect server start
    environment:
      PREFECT_API_DATABASE_CONNECTION_URL: postgresql+asyncpg://prefect:prefectpw@postgre
    ports:
      - "4200:4200"
    depends_on:
      - postgres
    restart: unless-stopped

  mlflow:
    image: ghcr.io/mlflow/mlflow:v2.16.0
    command: &gt;
      mlflow server
      --host 0.0.0.0 --port 5000
      --backend-store-uri postgresql://prefect:prefectpw@postgres:5432/prefect
      --default-artifact-root s3://prefect-artifacts
    environment:
      AWS_ACCESS_KEY_ID: admin
      AWS_SECRET_ACCESS_KEY: minio123
      MLFLOW_S3_ENDPOINT_URL: http://minio:9000
    ports:
      - "5500:5000"
    depends_on:
      - postgres
      - minio
    restart: unless-stopped

  prometheus:
    image: prom/prometheus:latest
    volumes:
      - ${WS}/monitoring/prometheus:/etc/prometheus
      - ${WS}/storage/prometheus:/prometheus
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
      - "--storage.tsdb.path=/prometheus"
    ports:
      - "9090:9090"
    restart: unless-stopped

  grafana:
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD: admin
    volumes:
```

```
      - ${WS}/storage/grafana:/var/lib/grafana
    depends_on:
      - prometheus
    restart: unless-stopped
YAML
```

서비스 기동

```
export WS=/srv/dgx_ws
cd $WS/compose

set -a
source $WS/config/.env
set +a

docker compose -f stack.yml up -d

# 상태 확인
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```

웹 UI 접속

- Prefect: http://localhost:4200
- MLflow: http://localhost:5500
- MinIO Console: http://localhost:9001
- Grafana: http://localhost:3000 (admin/admin)
- Prometheus: http://localhost:9090

11단계: 검증 및 다음 단계

검증 스크립트

```
cat &gt; $WS/scripts/setup/verify_setup.sh &lt;&lt;'BASH'
#!/bin/bash

export WS=/srv/dgx_ws
echo "🔍 DGX Spark Verification"
echo "========================"

# 1. OS
uname -a

# 2. GPU
nvidia-smi --query-gpu=name,driver_version --format=csv,noheader

# 3. Docker GPU
docker run --rm --gpus all nvidia/cuda:12.6.2-base-ubuntu22.04 nvidia-smi &gt; /tmp/docke
grep -q "NVIDIA" /tmp/docker_gpu.log &amp;&amp; echo "✅ Docker GPU OK" || echo "❌ Docker
```

```
# 4. Micromamba
micromamba env list | grep -E "msmd|dock|dais"

# 5. GNINA
micromamba run -n msmd gnina --version 2&gt;/dev/null &amp;&amp; echo "✅ GNINA OK" || ec

# 6. Services
docker ps --format "table {{.Names}}\t{{.Status}}"

echo "✅ Verification Complete"
BASH

chmod +x $WS/scripts/setup/verify_setup.sh
$WS/scripts/setup/verify_setup.sh
```

다음 단계

1. **CADD 파이프라인 테스트**

   ```
   micromamba activate msmd
   gnina -r receptor.pdbqt -l ligand.pdbqt --score_only --cnn_scoring rescore
   ```

2. **Prefect Flow 등록**

   ```
   prefect flow create-deployment ./cadd_pipeline.py:cadd_pipeline
   ```

3. **MLflow 실험 시작**

   ```
   export MLFLOW_TRACKING_URI=http://localhost:5000
   mlflow experiments create -n cadd_experiments
   ```

4. **GPU 모니터링**

   ```
   watch -n 1 nvidia-smi
   ```

최종 체크리스트

- ✅ First Boot 완료
- ✅ 시스템 기본 점검
- ✅ Hostname/IP 확인
- ✅ NVIDIA Sync 설치 및 연결
- ✅ 포트 충돌 해결
- ✅ OS 업데이트
- ✅ Docker GPU 런타임 확인
- ✅ Micromamba 설치
- ✅ Workspace 디렉토리 구성

- ✅ 외장 데이터 마운트
- ✅ Micromamba 환경 복원
- ✅ Docker Compose 실행
- ✅ 웹 UI 접속 확인
- ✅ 검증 완료

모든 항목이 완료되면 DGX Spark 개발 환경 셋업 완료!