Curation Image Status and GPU PC Monitoring Dashboard Installation Guidelines

v1.0, Nov. 2024

Seoul National University Bundang Hospital and Promedius

□ 목적 및 범위 ····· 2

□ 필요한 프로그램 ····· 2

□ 사전에 준비해야 할 사항들 ····· 2

□ 설치 과정 ····· 2

□ 실행 ····· 5

□ 설정 완료 및 Dashboard 사용 ····· 7

<그림 차례>

그림 1. compose.yml 파일 수정 ····· 3

그림 2 prometheus.yml 파일 생성 ····· 4

그림 3 grafana 접속 ····· 5

그림 4 dashboard 환경 구축 결과 ····· 6

그림 5 prometheus 환경 구축 결과 ····· 7

□ 목적 및 범위

○ 의료 영상 데이터를 효율적으로 관리하고 분석하기 위해, Imaging_CDM 테이블과 CDM의 person 테이블 등을 활용한 대시보드를 설치하여 큐레이션을 위한 의료 영상 현황을 실시간으로 모니터링함. 이를 통해 연구자들은 다양한 의료 영상 데이터의 처리 및 분석을 용이하게 할 수 있으며, 자동 레이블링 도구를 사용하는 GPU PC의 상태를 실시간으로 확인하고 관리할 수 있음. GPU 자원 사용 현황을 파악하여 성능 최적화 및 장애 예방에 기여하고, 의료 영상 분석의 정확도를 높이는 데 목적을 둠

□ 필요한 프로그램

○ Grafana : 전달받은 데이터를 시각화하는 오픈소스 프로그램이며, 대시보드 구성

○ Prometheus : 데이터를 정기적으로 수집하여 저장하며, 데이터가 전달되면 다시 수집을 반복하며 보통 Grafana와 함께 사용

○ Node Exporter : Prometheus는 등록된 Exporter들을 확인하여 데이터를 수집하는 프로그램으로 Linux 서버에서는 Node Exporter를 사용하며, 필요에 따라 다양한 Exporter를 설치하여 사용

○ PostgreSQL : Grafana는 Prometheus 외에도 SQL DB를 데이터 소스로 연결할 있어, 본 프로젝트에서는 오픈소스 PostgreSQL을 통해 DB 구축. (본 프로젝트에서는 이미 구축된 PostgreSQL에 연결하여 사용하므로 직접적인 설치는 없음)

□ 사전에 준비해야 할 사항들

○ Docker 설치 및 학습을 위한 환경 구성

○ 각 프로그램의 Docker 이미지 다운로드

- 인터넷이 가능한 PC에서“dockerpull‘프로그램명:버전’”명령어를 사용하여 각 프로그램의 이미지를 다운 받음
- ‘dockerpullgrafana/grafana’ (latest 생략 시 최신 버전이 다운로드 됨)
- ‘dockerpullprom/prometheus’
- ‘dockerpullprom/node-exporter’
- 필요한 이미지가 있다면 ,DockerHub에서 검색후 다운로드 받음. 별도의 Exporter나 PostgreSQL이미지도 다운로드가 가능함
- 이미지다운로드가 완료되면, 각 이미지를 추출하여 tar파일로 생성함
- “dockersave-o‘파일명’‘프로그램명:버전’”명령어로tar파일을 생성함
- 생성한 tar파일을 필요한PC로 이동시켜 이미지 등록과정을 진행함
- “dockerload-i‘tar파일명’”명령어를 사용하여 tar 파일을 로드함

□ 설치 과정

○ 작업공간 생성

- Docker컨테이너는 작업종료시 작업내역이 저장되지 않으므로, 작업공간을 미리생성함
- 적절한 폴더명을 지정하여 작업공간을 생성함
  · 예시:~/workspace/data-curation-dashboard

○ 환경 구축 시작

- Docker컨테이너는 각프로그램이 독립적인 환경에서 실행될수 있도록 설계되어 있으며,이를 적절히 조립하여 프로젝트 환경에 맞게 구성함
- DockerCompose명령어를 사용하여 여러컨테이너를 조합하고 환경을 구성함
- 작업공간에'compose.yml'파일을 생성하여, 이를 바탕으로 Docker환경을 구축함

○ compose.yml 파일 수정

- compose.yml파일에 다음과 같은 내용을 입력함

```
services:
  grafana:
    image: grafana/grafana
    volumes:
      - ./grafana-data:/var/lib/grafana
    ports:
      - "127.0.0.1:3000:3000"

  prometheus:
    image: prom/prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
      - ./prometheus-data:/prometheus
    ports:
      - "127.0.0.1:9090:9090"

  node-exporter:
    image: prom/node-exporter
    ports:
      - "9100:9100"
    command:
      - --web.listen-address=:9100
```

그림 1. compose.yml 파일 수정

- icompose.yml 파일은 전체를 읽은 다음 한 번에 실행되며, 순서는 상관없음
- 각 옵션은 다음과 같은 역할을 수행함
  - · image: docker image의 이름
  - · volumes: 앞서 언급했던 작업의 저장을 위해 사용자의 저장공간과 docker의 저장공간을 연결하여 파일을 남기게 하고, 상호작용을 가능하게 하는 옵션임. '실제 경로:docker 경로'로 작동함. 따라서 위의 이미지의 경로는 예제로 자기가 원하는대로 실제 경로를 설정해도 상관없음
  - · ports: 컨테이너와의 통신을 위해 포트를 열기 위한 옵션으로, docker compose의 경우 각 프로그램의 이름으로 통신이 가능하며 일단 여기서는 '사용자의 포트:docker의 포트' 정도만 이해하면 됨.

○ 작업 폴더 생성 및 권한 수정

- volumes에서 설정한 경로를 실제로 생성하며 여기서는 '/grafana-data' '/prometheus-data' 폴더를 생성함
- docker의 경우 별도의 작업 환경을 사용하기 때문에 uid를 별도로 가지게 되어 권한 문제가 발생함. 이를 해소하기 위해 앞서 생성한 경로의 권한 문제를 해결해야하며. 해결 방법 두 가지
  - · 해당 폴더에 쓰기 권한등 모든 권한을 부여하여 문제 해결 (chmod 777)
  - · 해당 폴더의 소유자를 docker의 uid로 변경하여 (chown) 문제 해결. 이 방법의 경우 docker image를 bash 모드로 실행하여 uid를 확인한 다음 해당 uid 값을 입력함. (docker exec -it '컨테이너id' bash' -> id 입력 -> uid 확인 후 변경) : 이 방법이 좀 더 안전한 방법에 해당함

○ prometheus.yml 파일 생성

- prometheus의 설정을 위한 파일에 해당하며, 이를 생성함
- 인터넷에 검색하면 템플릿이 존재하는데, 다른 것은 다 기본값을 사용하되 다음 부분은 반드시 설정을 해 주어야 함

```
scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node-exporter"
    static_configs:
      - targets: ["node-exporter:9100"]
```

그림 2. prometheus.yml 파일 생성

- 해당 부분은 prometheus의 포트 설정및 각각의 exporter의 설정 등록 부분으로, 이 부분이 없으면 prometheus및 exporter는 정상 작동할 수 없음. (통신이 불가능하므로 사용이 불가능함)

○ 세부 설정

- 두 개의 yml 파일을 통해 컨테이너를 통제하는데, 필요에 따라 옵션을 추가하거나 수정할 수 있음. 예를 들어 예시에서는 모든 것이 한 개의 컴퓨터에서 기능하기 때문에 localhost를 사용했지만, 별도의 pc에 exporter가 설치되어 있는등 통신이 필요할 경우 ports 설정등을 적당히 수정해야 함

□ 실행

○ 이 모든 것을 완료했다면. 작업공간에서 ‘docker compose up’ 을 입력하여 환경 실행

- docker가 3개의 컨테이너를 적당히 구성하여 실행 가능한 상태로 만듦

○ 성공적으로 docker가 실행되었다면, docker ps를 입력해서 컨테이너가 잘 작동하는지 확인함

○ grafana 설정

- localhost:3000을 입력해서 grafana에 접속
- 초기 grafana 로그인 암호는 admin/admin 이며, 접속 후 수정을 하도록 설정됨(본 프로젝트에서는 pc의 암호와 동일하게 설정)
- grafana에 접속했다면 왼쪽 탭에서 connections – data sources로 이동하여 연결을 설정함

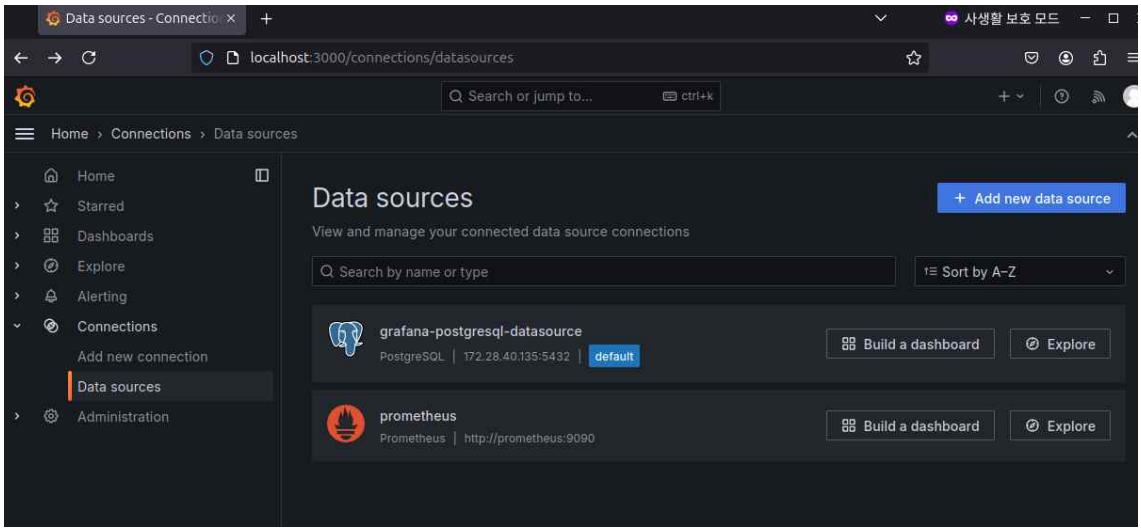

그림 3. grafana 접속

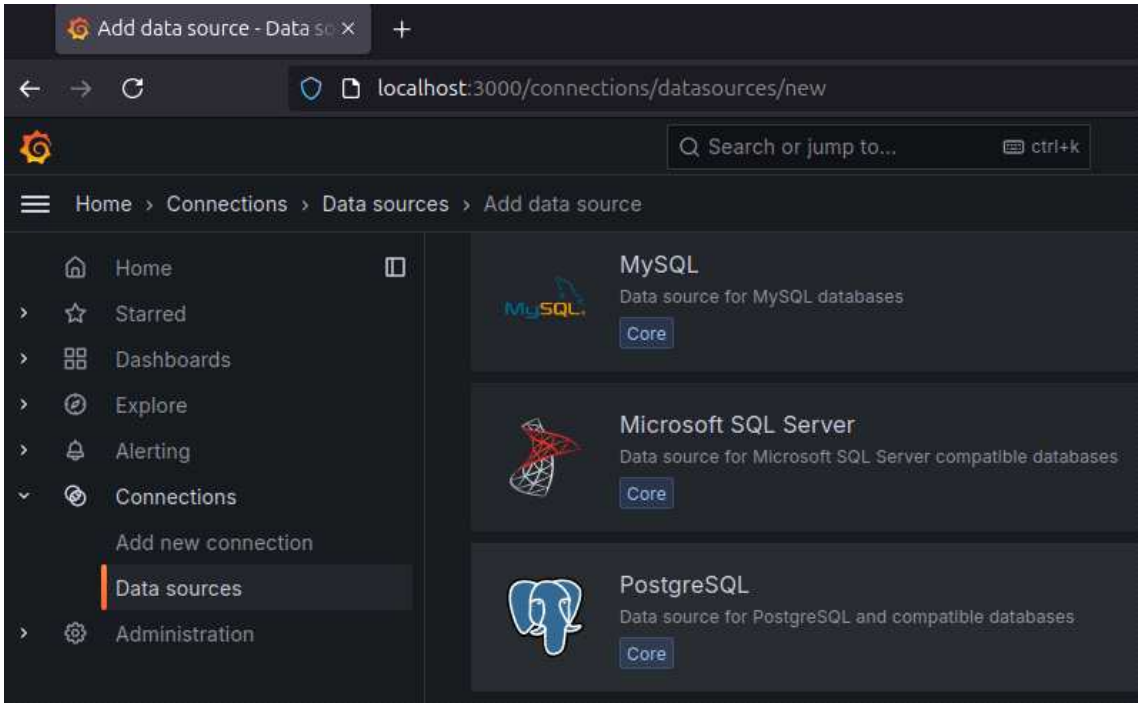

그림 4. dashboard 환경 구축 결과

- PostgreSQL을 사용할 것이므로 postgresql을 설정함. ip와 포트를 입력하고, db 이름 및 계정 정보, 프로그램 정보등을 입력하여 쿼리를 보낼 수 있도록 설정한후 save & test를 클릭하여 성공이 뜨면 연결이 된 것임

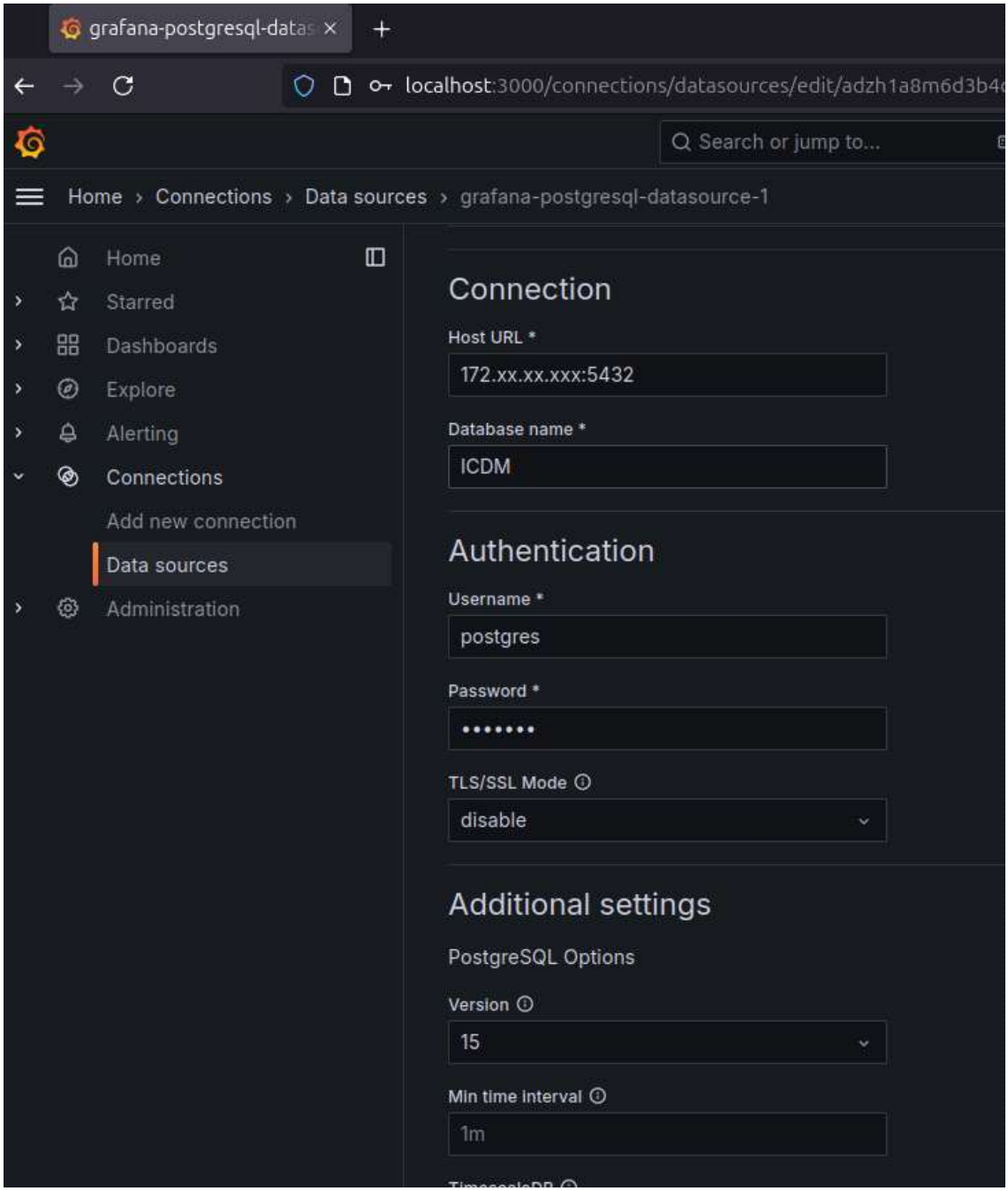

그림 5. prometheus 환경 구축 결과

- 동일하게 prometheus도 설정. 'http://prometheus:9090'을 입력후, Auth 설정은 끔. (No Authentication), TLS 설정도 다 체크 해제한 후 save & test를 클릭하여 성공하면 prometheus도 설정 완료됨

□ 설정 완료 및 Dashboard 사용

○ 여기까지 정상적으로 완료가 되었다면 dashboard 환경을 구축한 것으로, 실제로 작업하여 저장한 dashboard를 import 하여 사용하거나, 새로운 dashboard를 작업하여 사용