



GEdge Platform

GEdge(Griffin-Edge) Platform

- 초저지연 지능형 클라우드 엣지 SW 플랫폼 -

엣지 기반 AI/ML 서비스 흐름관리 및 실행환경 최적화 기술

2023.12.07

GS-AI 프레임워크 리더

이승(s.lee@softonnet.com)

“GEdge Platform”은 클라우드 중심의 엣지 컴퓨팅 플랫폼을 제공하기 위한
핵심 SW 기술 개발 커뮤니티 및 개발 결과물의 코드명입니다.

- New Leap Forward of

GEdge Platform Community 7th Conference (GEdge Platform v4.0 Release) -

Contents

- I** 연구 기술 개요
- II** 지능형 서비스 실행 런타임
- III** 엣지 기반 지능형 서비스 흐름 관리
- IV** 향후 연구 과제

GEdge 플랫폼 내 GS-Aiflow의 역할

초저지연 지능형 클라우드 엣지 플랫폼 (GEdge Platform)

클라우드 엣지 관리 플랫폼 (GM : GEdge Management Platform)

플랫폼 관리 도구 프레임워크 (GM-Tool)

Framework I/F

플랫폼 관리 기능 프레임워크 (GM-Center)

Platform I/F

지능형 서비스 운용 프레임워크 (GS-AI)

엣지 AI 서비스 환경
(GS-Aiflow)

엣지 협업 학습 환경
(GS-Optops)

서비스 협업 프레임워크 (GS-Link)

협업 게이트웨이
(GS-Linkgw)

협업 정책 생성
(GS-Linkhq)

Framework I/F

Framework I/F

초저지연 데이터 처리 프레임워크 (GS-Engine)

엣지 전용 스케줄러
(GS-Scheduler)

엣지 메시지 브로커
(GS-Broker)

클라우드 엣지 서비스 플랫폼 (GS : GEdge Service Platform)



연구 기술 개요



- 연구 기술

- 클라우드 엣지 기반 지능형 서비스의 운영 관리 기술
- 지능형 서비스 실행 런타임 및 관리 기술
- 클라우드 엣지 기반 지능형 서비스를 위한 경량화 기술

연구 주제

엣지 기반
AL/ML 서비스
흐름 관리 기술
및
실행환경 최적화 기술

연구 기술 상세

클라우드 엣지 기반 지능형 서비스의 운영 관리 기술

지능형 서비스 실행 런타임 및 관리 기술

클라우드 엣지 기반 지능형 서비스를 위한 경량화 기술

엣지 AI 서비스 환경

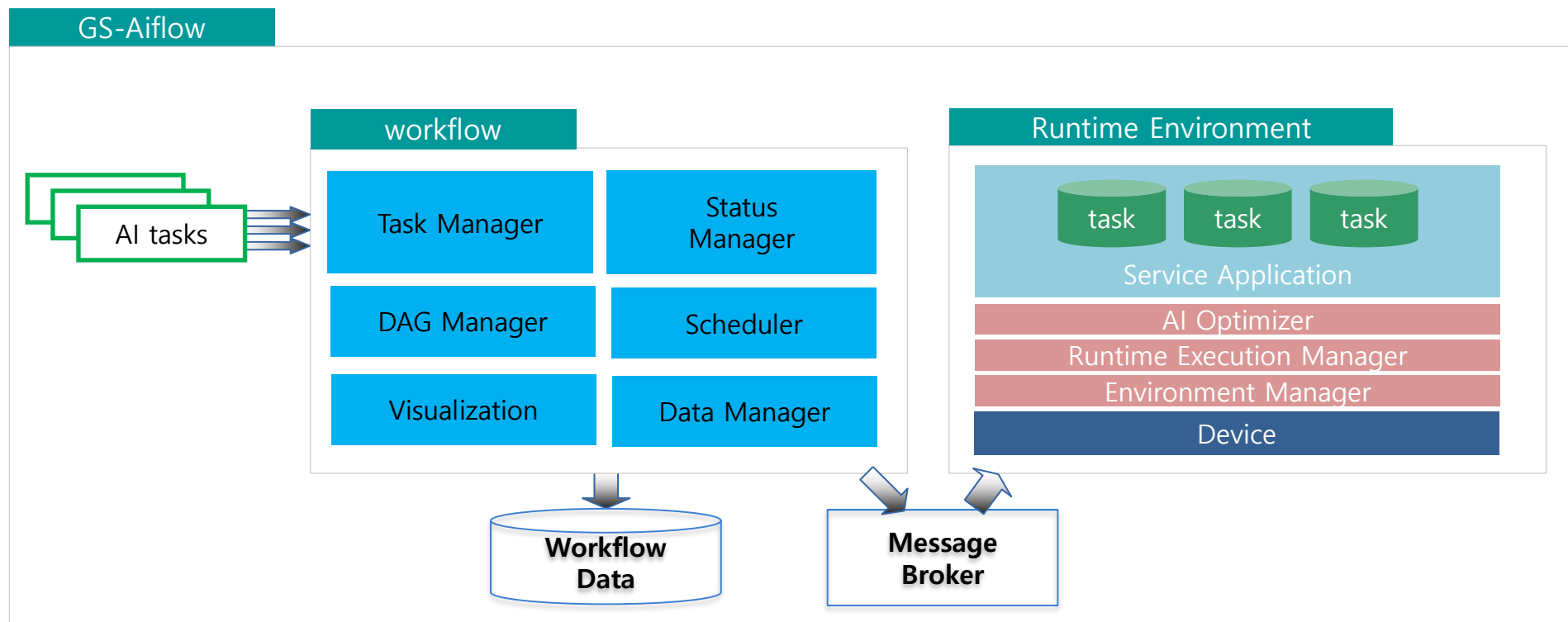
Aiflow

워크플로 관리 기술

실행 런타임 관리 기술

경량화/최적화 기술

- 클라우드 엣지 기반 지능형 서비스 워크플로 관리 프레임워크
 - 클라우드 엣지 기반 지능형 서비스 워크플로 관리 기술 연구 및 사용자 인터페이스 개발
 - 워크플로 관리 부분과 환경 관리 부분으로 분할 구성
 - 실행 런타임 관리 기술과 지능형 서비스를 위한 최적화/경량화 기술 반영을 위한 태스크 단위 관리



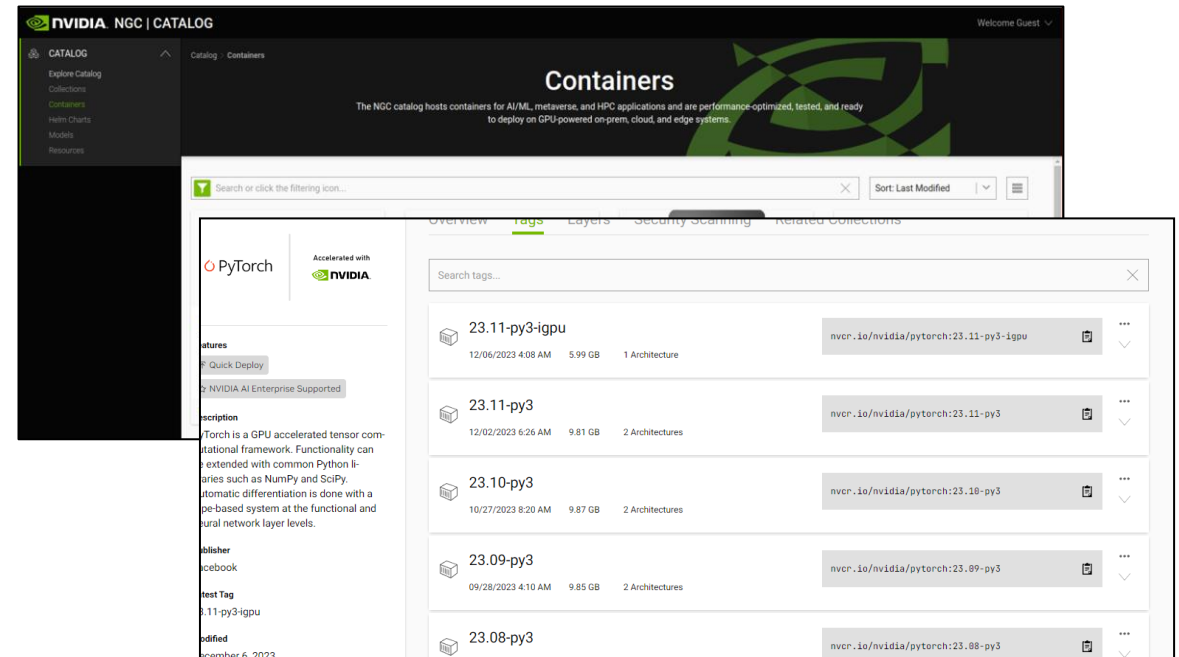


지능형 서비스 실행 런타임



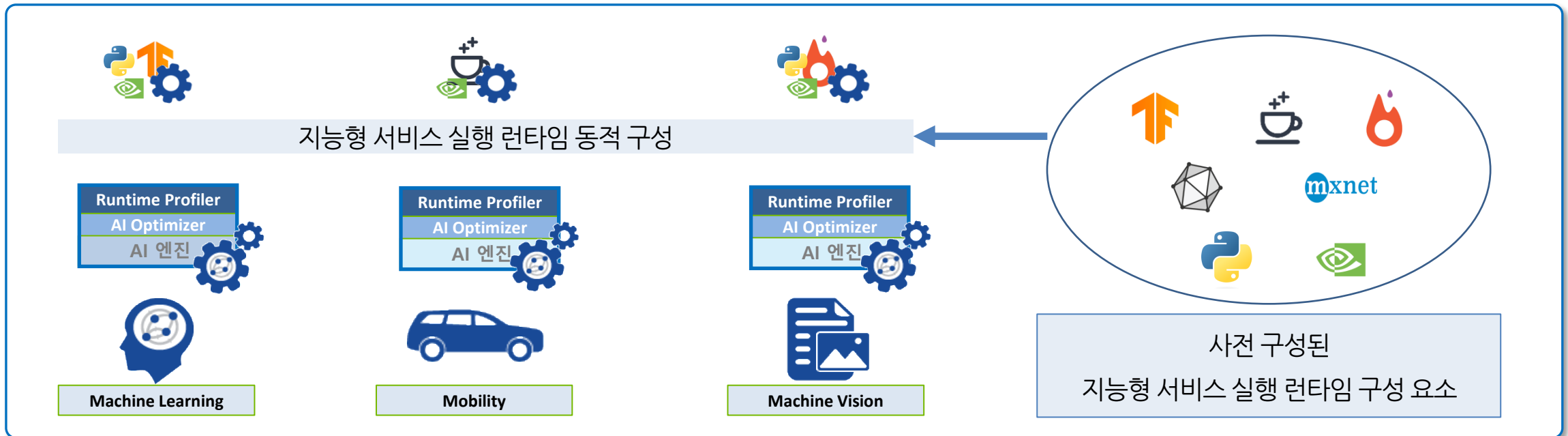
- **지능형 서비스 실행 런타임**

- 구성 요소 : AI 프레임워크, 디바이스 라이브러리 등 AI 서비스를 구현하거나 실행하는 데 필요한 환경
 - 예:) CUDA 11.1 + CUDNN 5.8 + Python 3.10 + PyTorch 2.1 + tensorflow 2.4
- 지능형 서비스 실행 런타임 (특정 조합) + Linux OS ≡ NGC 배포 컨테이너

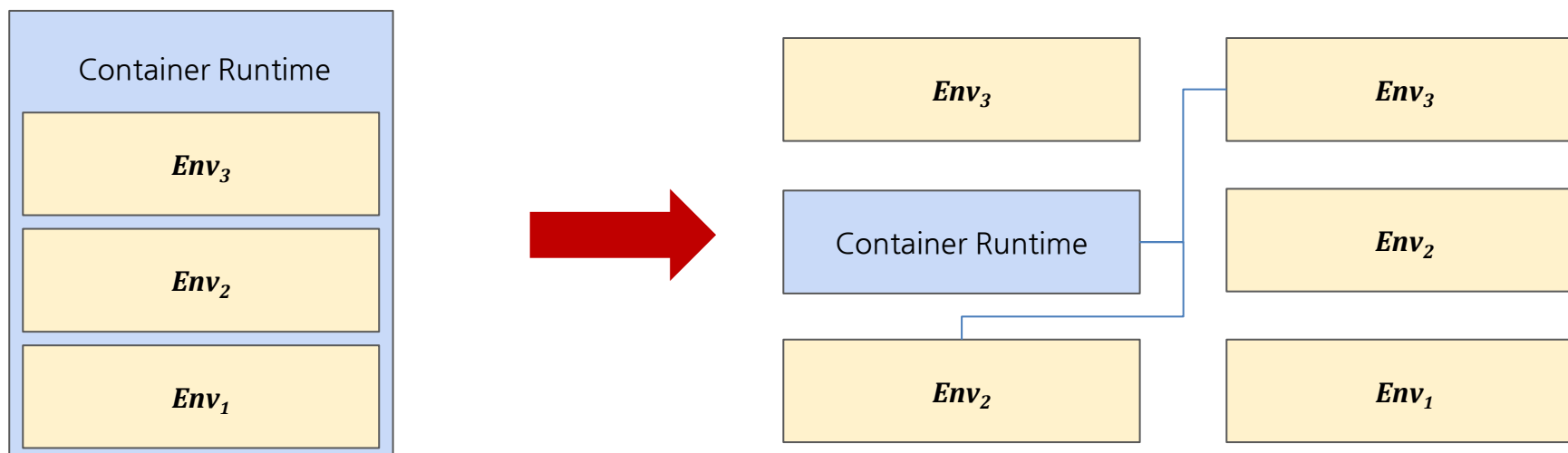


● 지능형 서비스 실행 런타임 및 관리 기술

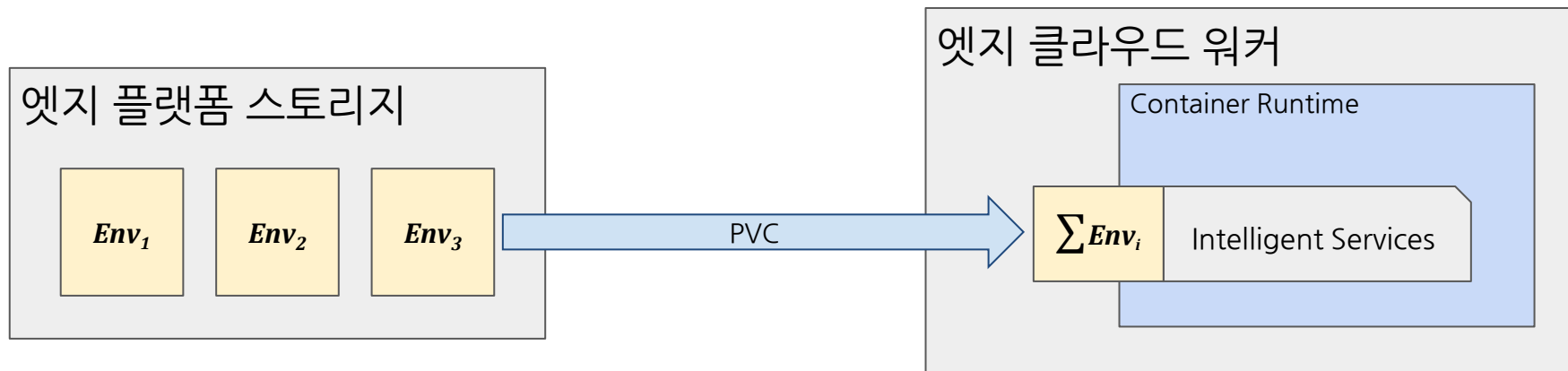
- 컨테이너의 이미지와 지능형 서비스 실행 런타임을 분리, 동적 연결
- 지능형 서비스 실행 런타임 구성 요소 사전 구성
- 사전 구성된 구성 요소의 동적 조합으로 서비스 실행 런타임 활성화
- 도메인, 모델에 특화된 AI 서비스의 컨테이너에 지능형 서비스 실행 런타임을 동적 적용



- 컨테이너 이미지 경량화
 - 필요한 AI 구성 요소는 컨테이너와 분리되어 존재
 - NGC의 컨테이너를 사용하거나 Layer로 구성하는 경우, 컨테이너 이미지 크기 증가 불가피
- 다양한 AI 라이브러리 / 프레임워크 조합
 - 동적으로 다양한 구성 요소 조합
 - 버전이 고정된 기존 방식에 비해 최소의 할당 리소스로 최적의 조합 선택 가능



- 런타임 재사용성 강화 : 컨테이너 이식/ 동작이 가능한 형태로
 - 컨테이너 의존성 제거, 재시작 등 관리 편의성 제고
 - 활용 가능한 오브젝트를 연구하여 영속성 볼륨(PV) - 영속성 볼륨 클레임(PVC)에 구성 요소를 저장하는 것으로 결정
- 다양한 조합, 최소의 리소스, 경량의 이미지
 - 필요한 구성 요소만 선택적으로 제공
 - 조합된 구성 요소를 사용할 수 있도록 시스템 라이브러리를 활용한 선택적 경로 설정



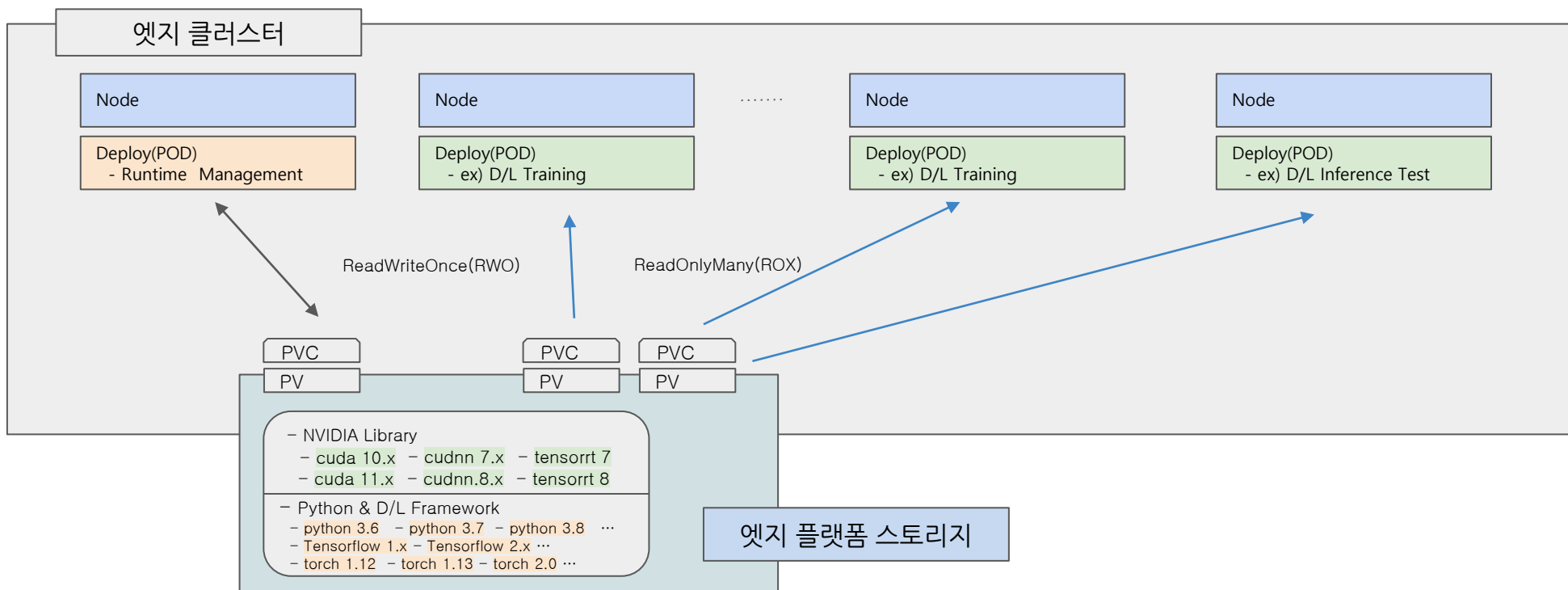
- 지능형 서비스 실행 런타임 구성 요소 수동 설치 데이터 준비
 - 이식성 강화, 활용도 제고를 위한 것
 - 설치 경로를 설정하여 활용할 수 있도록 OS Package 설치 방식 회피
- 지능형 서비스 실행 런타임 구성 요소 데이터 저장소 구성
 - 다양한 조합 지원을 위해 구성 요소 및 버전 별로 각각 PV 구성
- Linux 컨테이너에 구성 요소 설치
 - 별도의 라이브러리가 추가되지 않은 기본 Linux OS 이미지로 컨테이너 준비
 - 준비한 구성 요소 별 저장소를 PVC로 지정하여 구성
 - 설치 경로를 마운트된 PV를 활용하도록 지정하여 설치

```
command: ["/bin/bash", "-c"]
args: ["source /root/path.sh; env; tail -f /dev/null"]
env:
- name: NEW_PATH
  value: '/root/volume/miniconda3/envs/tf2_py38/bin:/root/volume/cuda/cuda-11.2/bin'
- name: LD_LIBRARY_PATH
  value: '/root/volume/cuda/lib64:/root/volume/cudnn/lib64'
```

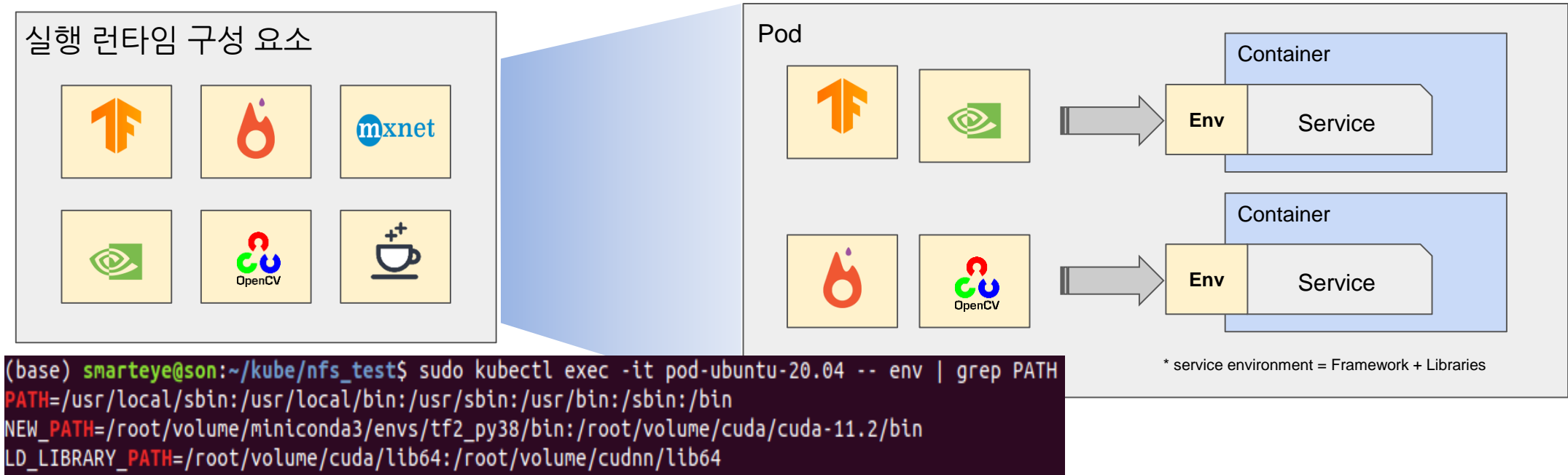
```
volumeMounts:
- mountPath: "/root/path.sh"
  name: fileconfig
  subPath: path.sh
- mountPath: "/root/volume/cuda"
  name: nfs-volume-total
  subPath: cuda/cuda-11.2
- mountPath: "/root/volume/miniconda3"
  name: nfs-volume-total
  subPath: miniconda3
- mountPath: "/root/volume/cudnn"
  name: nfs-volume-total
  subPath: cudnn/cudnn-v8.2.1-11.x-linux-x64
volumes:
- name: nfs-volume-total
  persistentVolumeClaim:
    claimName: nfs-pvc-total
```

컨테이너 구성 시점: 실행 런타임 이식

- 사전 구성된 실행 런타임 구성 요소의 PV를 조합하여 지능형 서비스 실행 런타임 구성 -> 버전 다양성 제공
 - 클러스터간 활용이 필요한 런타임 구성 요소는 NFS PV를 활용
 - 지역적 활용(클러스터 내부) 대상 런타임 구성 요소는 클러스터 내 Object Storage PV 활용



- 컨테이너 시작 시점: 실행 런타임 환경 적용
 - POD(컨테이너) 배포 시 전달하는 변수/파라미터 활용
 - 리눅스 시스템 라이브러리를 통한 환경 변수 적용



- NGC 사용 시나리오 대비 빠른 배포
 - 엣지 클러스터 환경에서 약 50배 차이 (6분 47초 vs 7초)
 - Ubuntu 20.04, Pytorch, CUDA, cuDNN, TensorRT 환경
- 컨테이너 이미지 경량화로 이미지 풀링 코스트 감소
 - 엣지 클러스터 환경에서 100배 이상 차이

NGC 컨테이너 사용 시

```
conditions:
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:11:48Z"
  status: "True"
  type: Initialized
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:18:35Z"
  status: "True"
  type: Ready
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:18:35Z"
  status: "True"
  type: ContainersReady
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:11:48Z"
  status: "True"
  type: PodScheduled
```

6분
47초

nvc.io/nvidia/pytorch 23.05-py3 22GB

지능형 서비스 실행 런타임 적용 시

```
conditions:
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:02:36Z"
  status: "True"
  type: Initialized
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:02:44Z"
  status: "True"
  type: Ready
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:02:44Z"
  status: "True"
  type: ContainersReady
- lastProbeTime: null
  lastTransitionTime: "2023-12-05T03:02:36Z"
  status: "True"
  type: PodScheduled
```

8초

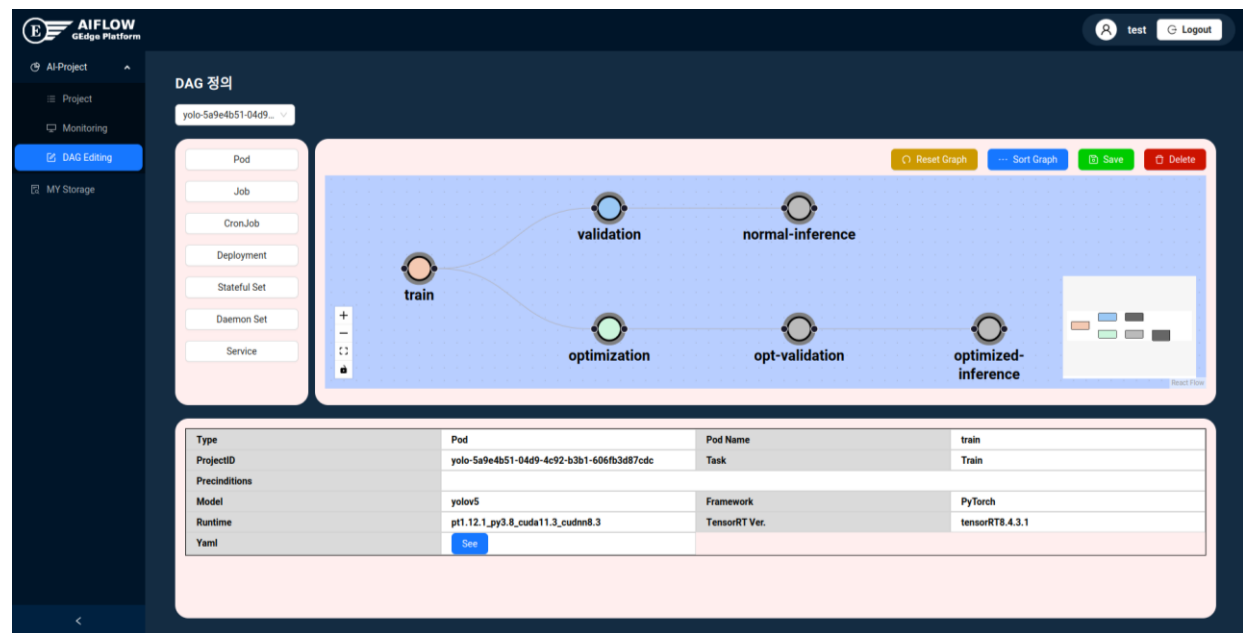
Ubuntu 22.04 77.8MB



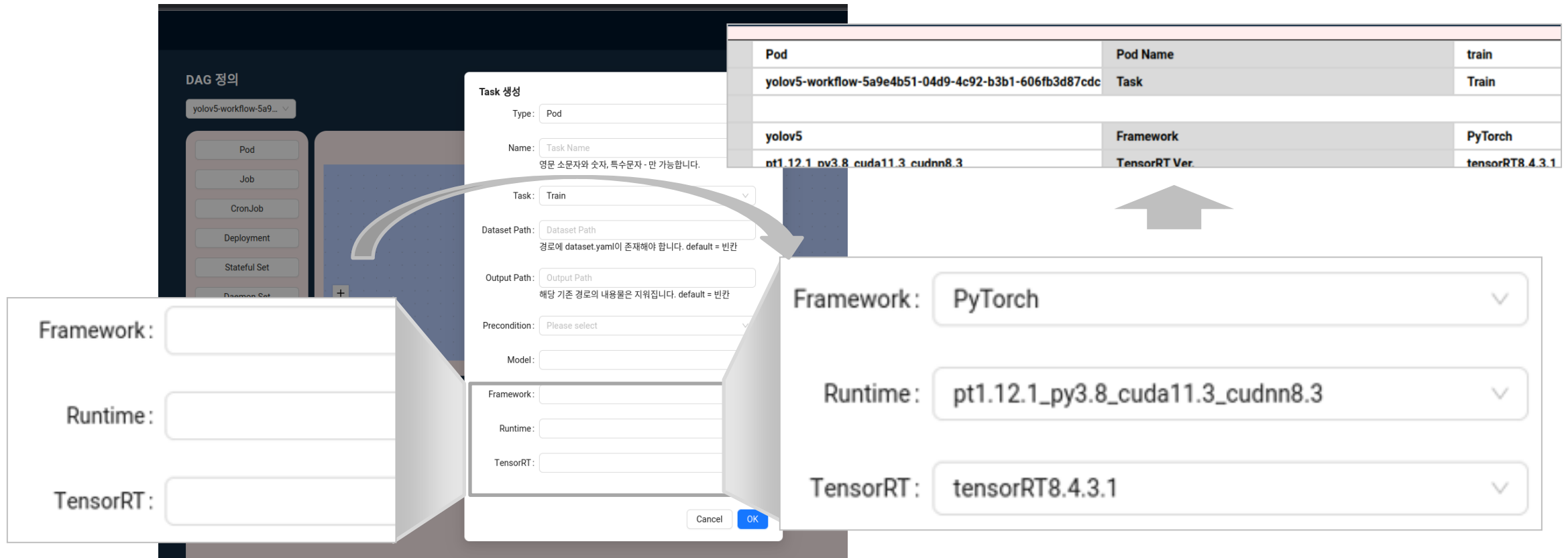
엣지 기반 지능형 서비스 흐름 관리



- 엣지 클러스터 내 AI/ML 서비스 생성 관리를 제공하는 프레임워크
 - 엣지 클러스터 환경에서 오브젝트 생성, 모니터링, 배포
 - 활용도 확대를 위한 엣지 플랫폼 외의 별도의 사용자 관리
- 태스크/ DAG 기반 관리
 - 학습, 검증, 모델 최적화, 최적화 검증, 추론 서비스 등 태스크 정의, DAG 기반 관리
 - 선행 태스크 상태와 실행 조건에 따른 흐름 관리



- 태스크 정의 시점에 지능형 서비스 실행 런타임 지정
- DAG 내 태스크 별, 실행 런타임 적용
 - 태스크의 특성에 따라 필요 런타임을 선택하여 적용



The image shows a screenshot of a DAG (Directed Acyclic Graph) definition interface. On the left, a sidebar lists various task types: Pod, Job, CronJob, Deployment, Stateful Set, and Daemon Set. The 'Pod' type is selected. The main area displays the 'Task 생성' (Task Creation) form for a 'Train' task. The form includes fields for Name, Task, Dataset Path, Output Path, Precondition, and Model. A 'Task 생성' dialog box is open, showing the configuration for the 'Train' task. The dialog box has three sections: Framework, Runtime, and TensorRT. The Framework is set to 'PyTorch', the Runtime is set to 'pt1.12.1_py3.8_cuda11.3_cudnn8.3', and the TensorRT is set to 'tensorRT8.4.3.1'. An arrow points from the 'Task 생성' dialog box to the 'Task 생성' form, indicating the application of the configuration.

Pod	Pod Name	Task
yolov5-workflow-5a9e4b51-04d9-4c92-b3b1-606fb3d87cdc	train	Train

Framework	TensorRT Ver.
PyTorch	tensorRT8.4.3.1

Task 생성

Type: Pod

Name: Task Name
영문 소문자와 숫자, 특수문자 - 만 가능합니다.

Task: Train

Dataset Path: Dataset Path
경로에 dataset.yaml이 존재해야 합니다. default = 빈칸

Output Path: Output Path
해당 기존 경로의 내용물은 지워집니다. default = 빈칸

Precondition: Please select

Model:

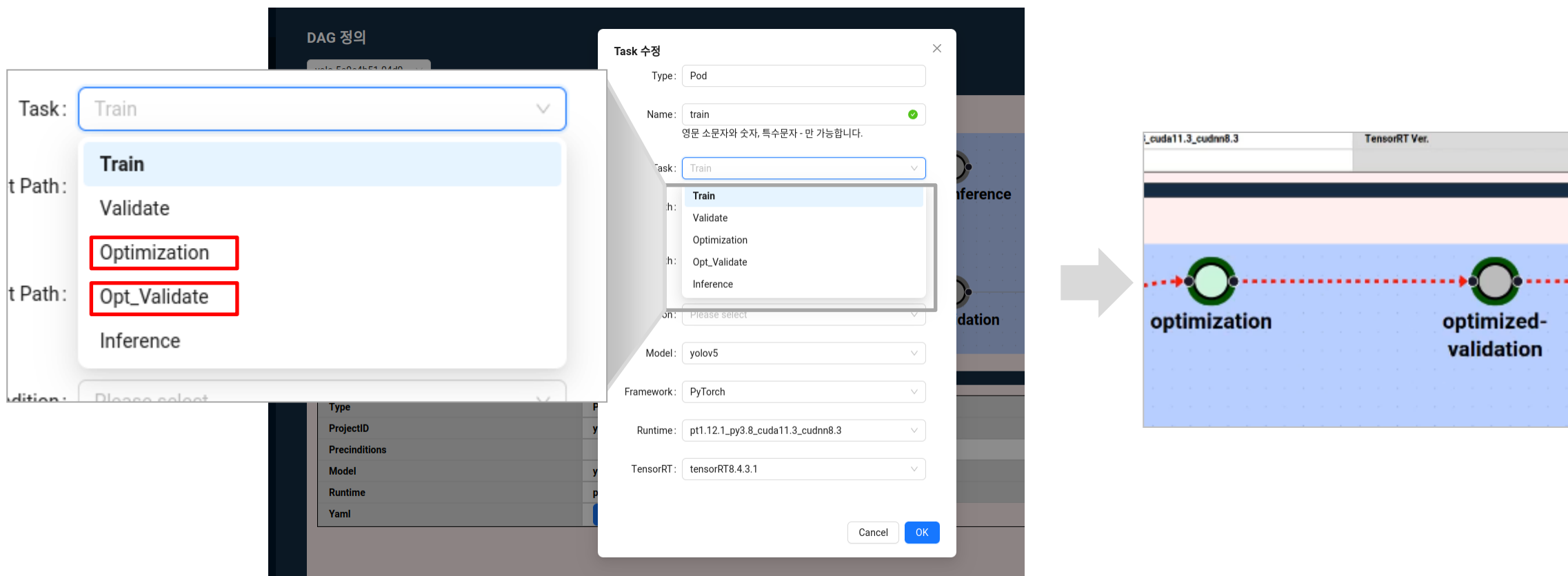
Framework: PyTorch

Runtime: pt1.12.1_py3.8_cuda11.3_cudnn8.3

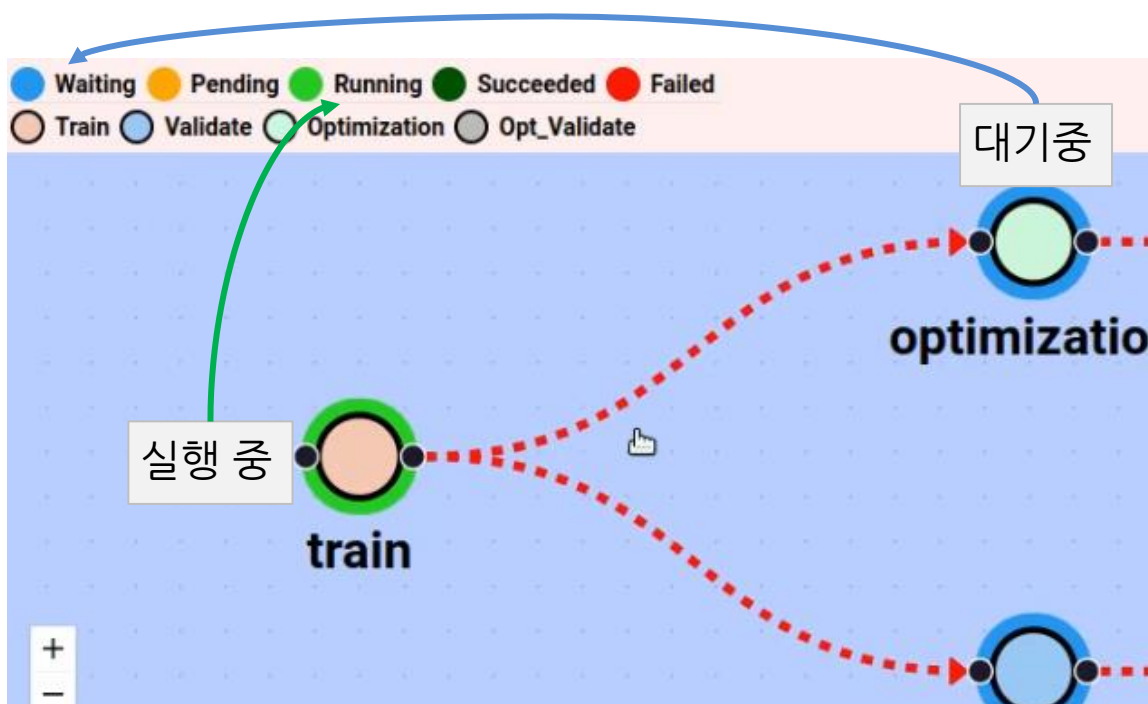
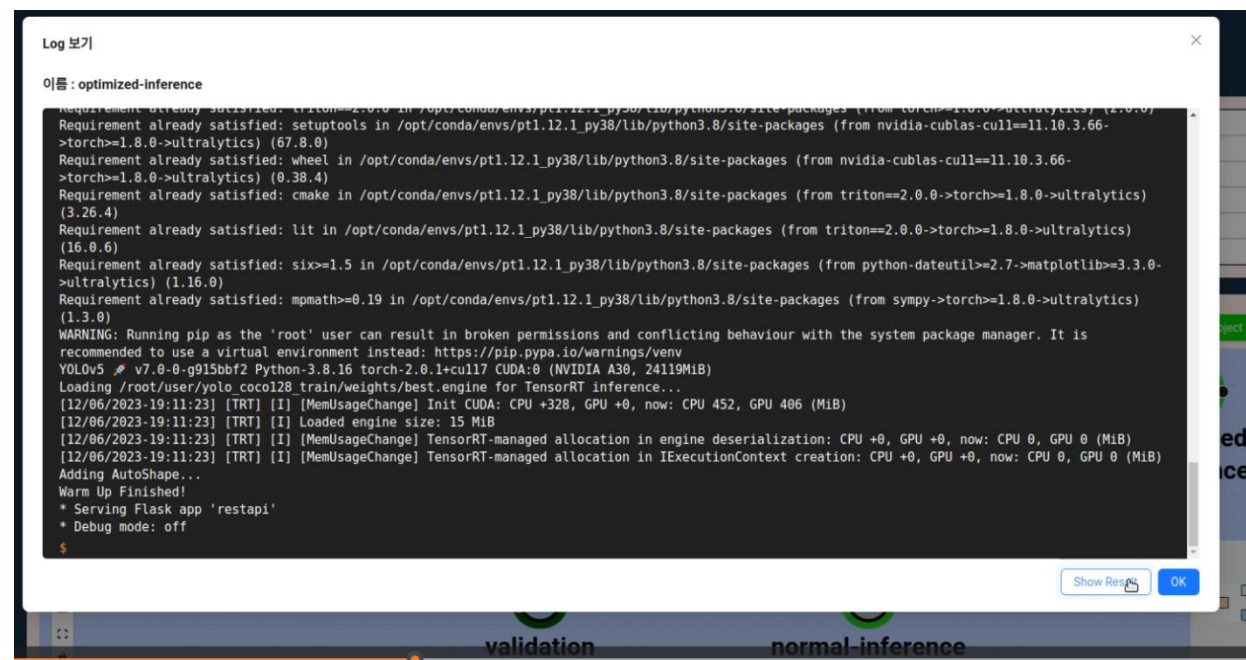
TensorRT: tensorRT8.4.3.1

Cancel OK

- 정의 가능한 태스크로 모델 최적화/경량화 태스크 구성
 - 학습으로 생성된 모델에 대해 최적화/경량화 프로세스 수행
- 경량화 검증 태스크 별도 구성



- DAG 기반의 흐름 관리
 - 실시간 모니터링 되는 선행 태스크의 상태와 태스크의 조건에 따라 인스턴스 시작
- 태스크 실행 환경 및 로그 조회
 - 엣지 플랫폼의 로그 조회 인터페이스로 조회

The image shows a log viewer interface with the title 'Log 보기' (View Log). The log content is as follows:

```
이름: optimized-inference
Requirement already satisfied: triton==2.0.0 in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from torch==1.8.0->ultralytics) (2.0.0)
Requirement already satisfied: setuptools in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from nvidia-cublas-cu11==11.10.3.66->torch==1.8.0->ultralytics) (67.8.0)
Requirement already satisfied: wheel in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from nvidia-cublas-cu11==11.10.3.66->torch==1.8.0->ultralytics) (0.38.4)
Requirement already satisfied: cmake in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from triton==2.0.0->torch==1.8.0->ultralytics) (3.26.4)
Requirement already satisfied: lit in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from triton==2.0.0->torch==1.8.0->ultralytics) (16.0.6)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultralytics) (1.16.0)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/envs/pt1.12.1_py38/lib/python3.8/site-packages (from sympy->torch==1.8.0->ultralytics) (1.3.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
YOLov5 v7.0-0-g915bbf2 Python-3.8.16 torch-2.0.1+cu117 CUDA:0 (NVIDIA A30, 24119MiB)
Loading /root/.user/yolo_coco128_train/weights/best.engine for TensorRT inference...
[12/06/2023-19:11:23] [TRT] [I] [MemUsageChange] Init CUDA: CPU +328, GPU +0, now: CPU 452, GPU 406 (MiB)
[12/06/2023-19:11:23] [TRT] [I] Loaded engine size: 15 MiB
[12/06/2023-19:11:23] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in engine deserialization: CPU +0, GPU +0, now: CPU 0, GPU 0 (MiB)
[12/06/2023-19:11:23] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecutionContext creation: CPU +0, GPU +0, now: CPU 0, GPU 0 (MiB)
Adding AutoShape...
Warm Up Finished!
* Serving Flask app 'restapi'
* Debug mode: off
$
```

At the bottom of the interface, there are tabs for 'validation' and 'normal-inference'.

IV

향후 연구 과제



- **메타데이터 : 런타임 실행 환경 - HW(Driver) - Model 간 상호 의존성, 호환성 데이터 구축**
 - 런타임 실행환경 내 Library - Framework 간 상호 의존성, 버전 호환성 명세화
 - HW - 런타임 실행 환경 간 의존성, 버전 호환성 명세화
 - 모델 - 런타임 실행 환경 간 의존성 버전 호환성 명세화
- **Model 파일/데이터와 메타데이터 통합관리**
 - Database, metadata file, file header 등 관리 방안 연구
- **객체 Driven 런타임 실행 환경 선정 정책 구성**
 - Model 이 선택 되어있을 때, 호환 가능한 런타임 실행 환경 선정
 - 런타임 실행 환경이 선정되었을 때, 호환 가능한 HW 선정
 - Model 을 활용한 지능형 서비스를 위해 할당할 수 있는 Node 선정
- **흐름 관리 및 운영 관리 기술로 구현**

- **MSA / Serverless 특화 서비스 관리**
 - 서비스 Load, Capacity, Pod Replica에 특화
 - 사용자 제공 정보 단순화, 사용 편의성 확보
 - 서비스 콘텐츠 업데이트 필요 시, 워크플로 방식 갱신
- **서비스 특성 반영 Load Factor 발굴**
 - GPU/CPU Load가 Constant 특성을 지니는 지능형 서비스를 위한 Load Factor 발굴, 정량화
 - RTT, Network Delay, Outlier 감지 정책
 - Ingress 등 로드밸런서 타입의 객체를 통한 로드팩터 감지
 - Custom Factor를 위한 API 개발
 - POD replica 수정을 위한 Threshold Factor 개발
- **엣지 플랫폼 기반 MSA Monitoring Provisioning**
 - 서비스 인스턴스의 플랫폼 내 클러스터간 이동 및 추가, 관리

감사합니다.

<http://gedge-platform.github.io>



GS-AI 프레임워크 리더
이승 (s.lee@softonnet.com)

Welcome to GEdge Platform

An Open Cloud Edge SW Platform to enable Intelligent Edge Service

GEdge Platform will lead Cloud-Edge Collaboration