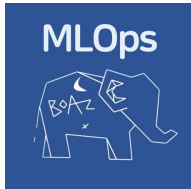
Model Serving

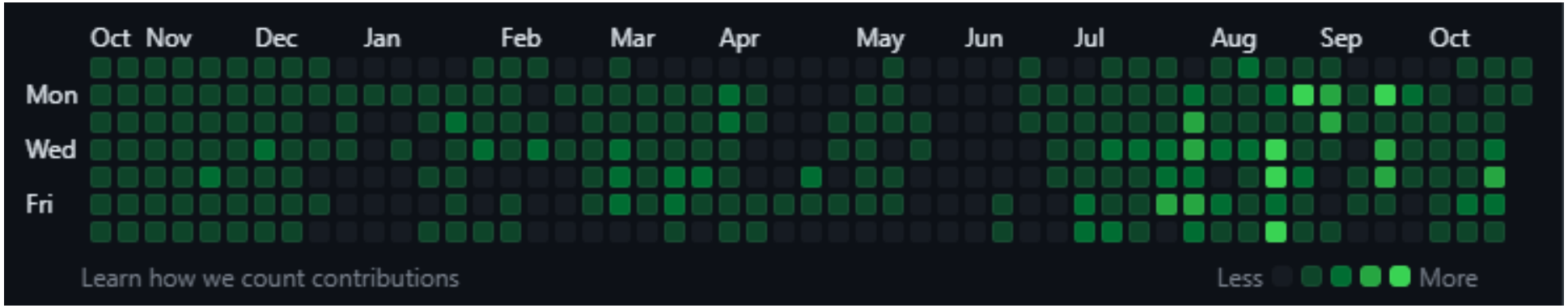
BOAZ MLOps 소모임

19기 분석 오효근



발표자 소개

- 오효근
  - BOAZ 19기 분석 부문 수료
  - AgileSoDA Machine Vision MLE 재직 중
  - Algorithm, MLOps, 자동화에 관심이 많습니다!



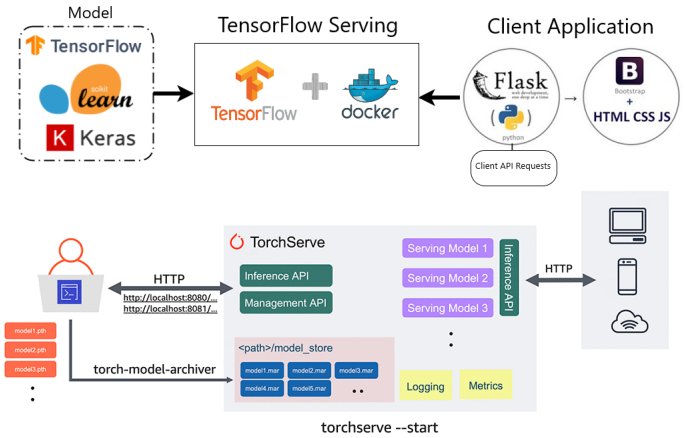
GitHub

Blog

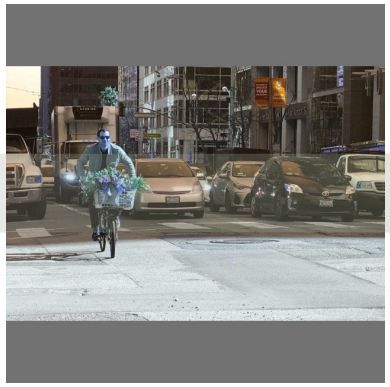
Contents

Introduction

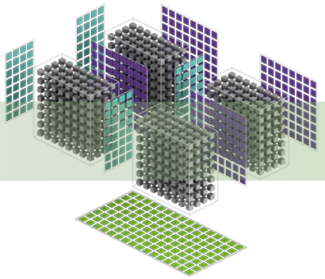
- Model Serving
  - Definition: 학습된 machine learning model을 실제 환경에서 사용할 수 있도록 하는 과정
  - Tools / Platforms
    - TensorFlow Serving
    - TorchServe
    - Triton Inference Server
    - MLflow



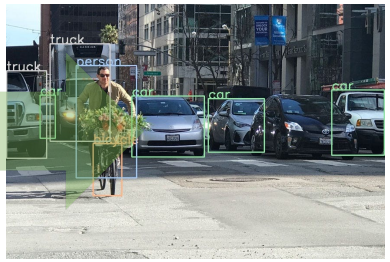
Pre-processing



Inferencing



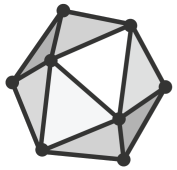
Post-processing



MLOps

2.1. Export

- ONNX (Open Neural Network Exchange)
  - 다양한 플랫폼에서 실행 가능
  - Deep learning에 대한 표준화 제공



ONNX

```
git clone https://github.com/ultralytics/yolov5
cd yolov5
conda create -n yolov5 python=3.8 -y
conda activate yolov5
pip install -r requirements.txt
python export.py --weights yolov5s.pt --include onnx
```

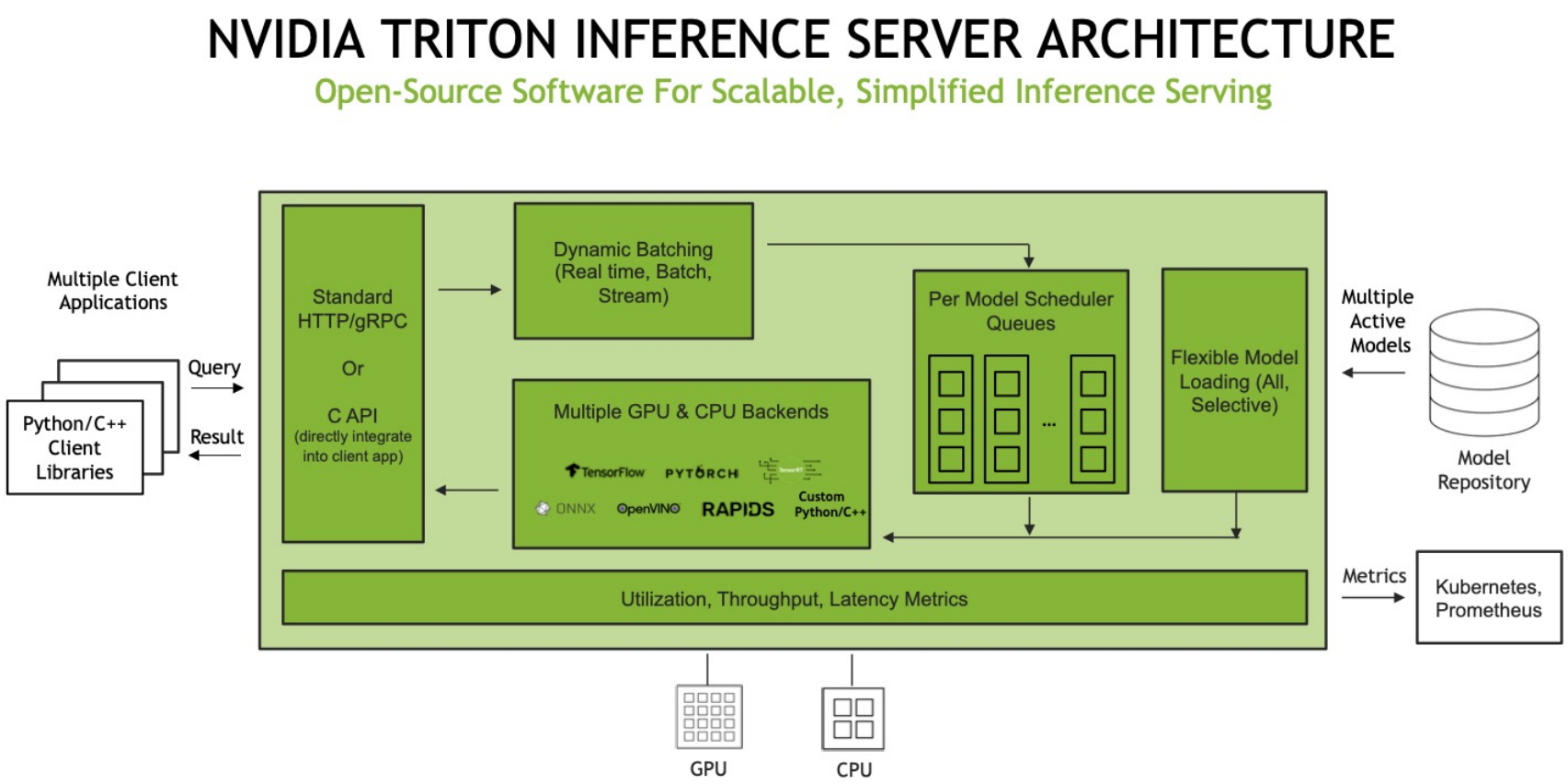
```
Export complete (4.7s)
Results saved to /home/zerohertz/Zerohertz/yolov5
Detect:          python detect.py --weights yolov5s.onnx
Validate:        python val.py --weights yolov5s.onnx
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'yolov5s.onnx')
Visualize:       https://netron.app

~/Zerohertz/yolov5 on master                    took 7s yolov5 with zerohertz@0hz at 15:36:45
❯ ls
benchmarks.py  classify        data      export.py   LICENSE  README.md       requirements.txt  setup.cfg  tutorial.ipynb  val.py       yolov5s.pt
CITATION.cff   CONTRIBUTING.md  detect.py hubconf.py  models   README.zh-CN.md segment           train.py   utils           yolov5s.onnx
```

2.2. Triton Inference Server

- Multiple Client Applications: 다양한 클라이언트 애플리케이션들이 Triton Inference Server에 쿼리를 보낼 수 있습니다.
- Python/C++ Client Libraries: Triton은 Python과 C++을 위한 클라이언트 라이브러리를 제공하여 사용자들이 서버에 쿼리를 쉽게 보낼 수 있게 합니다.
- Standard HTTP/gRPC: 클라이언트 애플리케이션은 HTTP 또는 gRPC를 통해 서버에 접속할 수 있습니다.
- C API: 클라이언트 애플리케이션은 C API를 사용하여 직접 Triton과 통합될 수 있습니다.
- Dynamic Batching: Triton은 실시간, 배치, 스트림과 같은 다양한 요청 유형을 동적으로 처리할 수 있는 능력을 가지고 있습니다.
- Multiple GPU & CPU Backends: 다양한 백엔드 프레임워크를 지원하여 GPU와 CPU에서 동작할 수 있습니다. TensorFlow, PyTorch, ONNX, OpenVINO, RAPIDS, Custom Python/C++ 백엔드 등을 지원합니다.
- Per Model Scheduler Queues: 각 모델마다 스케줄러 큐가 있어 추론 요청을 관리합니다.
- Flexible Model Loading: 모든 모델을 로드하거나 선택적으로 모델을 로드할 수 있는 유연성을 제공합니다.
- Model Repository: 모든 활성화된 모델들은 모델 저장소에서 관리됩니다.
- Metrics: 서버의 성능과 관련된 다양한 메트릭들을 제공합니다. (예: 활용도, 처리량, 지연 시간)
- Kubernetes, Prometheus: Triton은 Kubernetes와 Prometheus와 같은 플랫폼 및 도구와 통합하여 스케일링과 모니터링을 지원합니다.

2.2. Triton Inference Server

- NVIDIA에서 제공하는 여러 AI 모델들을 동시에 서빙 하도록 설계된 open source 추론 서버



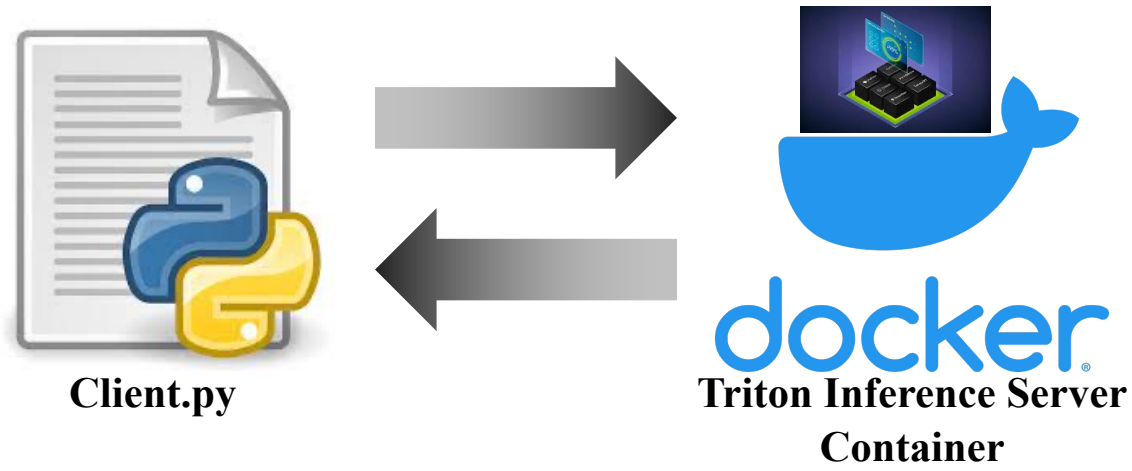
MLOps

2.2. Triton Inference Server

- **Service Port**
  - 8000: RestAPI 를 위한 HTTPService 포트
  - 8001: gRPC 를 위한 GRPCInferenceService 포트
  - 8002: Metric 수집을 위한 포트

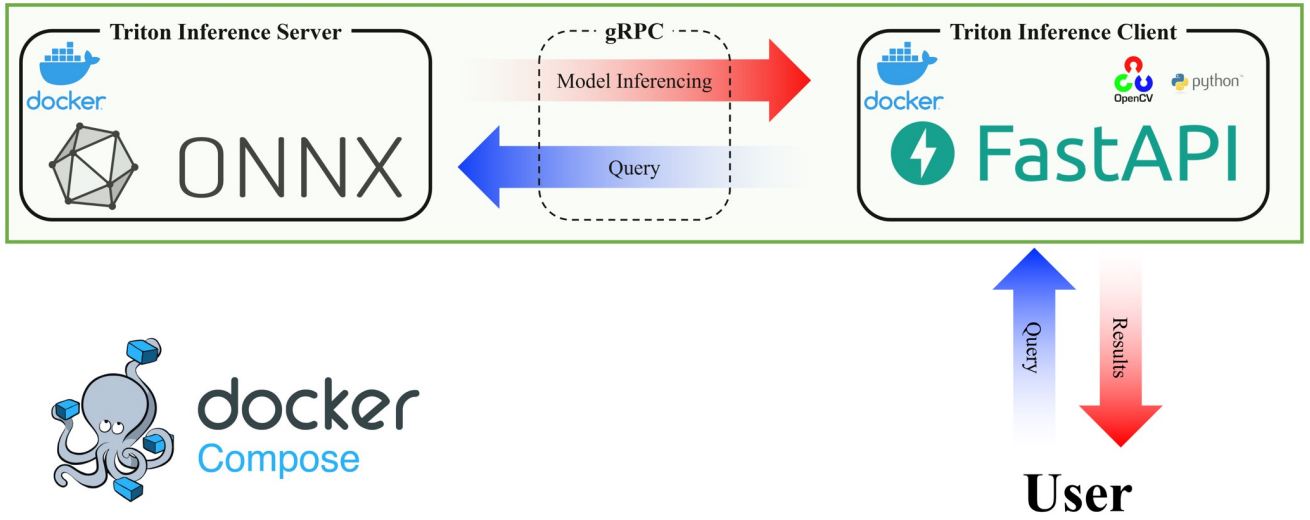
```
# I1023 1 grpc_server.cc:2445] Started GRPCInferenceService at 0.0.0.0:8001
# I1023 1 http_server.cc:3555] Started HTTPService at 0.0.0.0:8000
# I1023 1 http_server.cc:185] Started Metrics Service at 0.0.0.0:8002
```

```python
def inference(input_image: np.ndarray):
    SERVER_URL = "0.0.0.0:8001"
    MODEL_NAME = "YOLO"
    cv2.imwrite("input_image.jpg", input_image)
    input_image = input_image.astype("float32")
    input_image = input_image.transpose((2, 0, 1))[np.newaxis, :] / 255.0
    input_image = np.ascontiguousarray(input_image)
    print("input_image.shape:", input_image.shape)
    with grpcclient.InferenceServerClient(SERVER_URL) as triton_client:
        inputs = [
            grpcclient.InferInput(
                "images", input_image.shape,
np_to_triton)dtype(np.float32)
        ]
        inputs[0].set_data_from_numpy(input_image)
        outputs = [grpcclient.InferRequestedOutput("output0")]
        response = triton_client.infer(
            model_name=MODEL_NAME, inputs=inputs, outputs=outputs
        )
        response.get_response()
        output0 = response.as_numpy("output0")
    print("output0.shape:", output0.shape)
    return output0
```



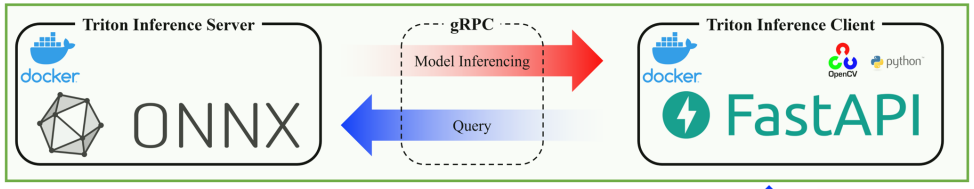
MLOps

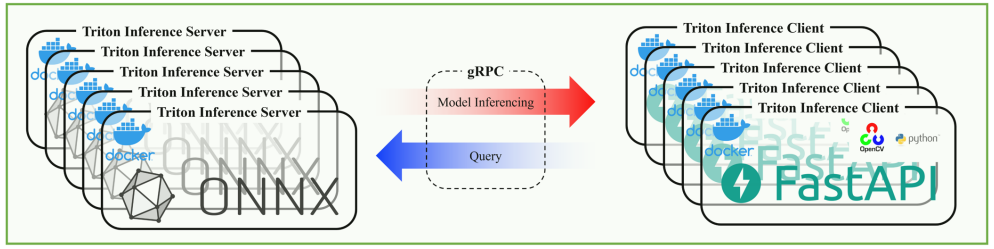
3.1. Docker Compose




```yaml
version: "3.5"

services:
    tritoninferenceserver:
        container_name: server
        image: nvcr.io/nvidia/tritonserver:23.06-py3
        environment:
            - NVIDIA_VISIBLE_DEVICES=0
        volumes:
            - ${PWD}/src/server:/models
        command: tritonserver --model-
repository=/models
    fastapi:
        container_name: client
        build:
            context: .
            dockerfile: ./src/client/Dockerfile
        ports:
            - "80:80"

networks:
    default:
        name: mlops-network
```

3.2. Kubernetes





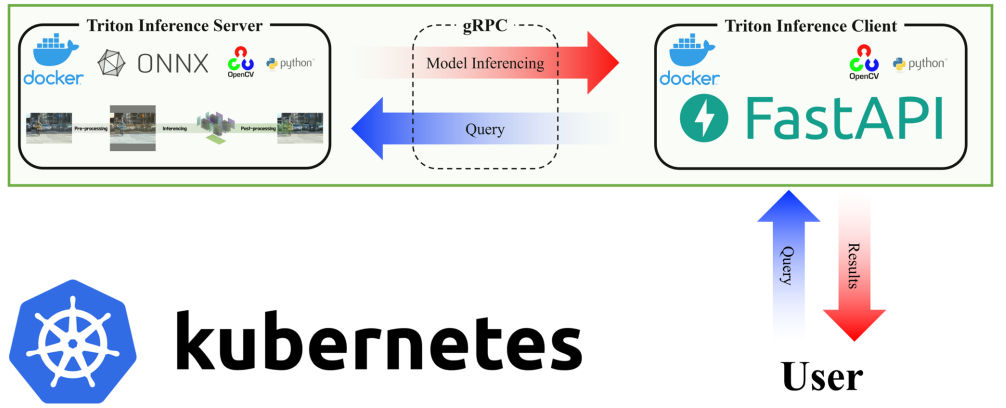

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: triton-inference-server
spec:
  replicas: 5
  selector:
    matchLabels:
      app: triton-inference-server
  template:
    metadata:
      labels:
        app: triton-inference-server
    spec:
      containers:
        - name: server
          image: yolo-server:latest
          imagePullPolicy: IfNotPresent
          env:
            - name: NVIDIA_VISIBLE_DEVICES
              value: "0"
          command:
            ["tritonserver", "--model-repository=/models", "--log-
verbose=2"]
          resources:
            requests:
              memory: "1024Mi"
              cpu: "1000m"
            limits:
              memory: "2048Mi"
              cpu: "2000m"
---
apiVersion: v1
kind: Service
metadata:
  name: triton-inference-server-svc
spec:
  selector:
    app: triton-inference-server
  ports:
    - protocol: TCP
      port: 8001
      targetPort: 8001
```


MLOps

3.3. Ensemble

- 여러 모델들을 연결하여 하나의 단일 모델처럼 작동하도록 설정

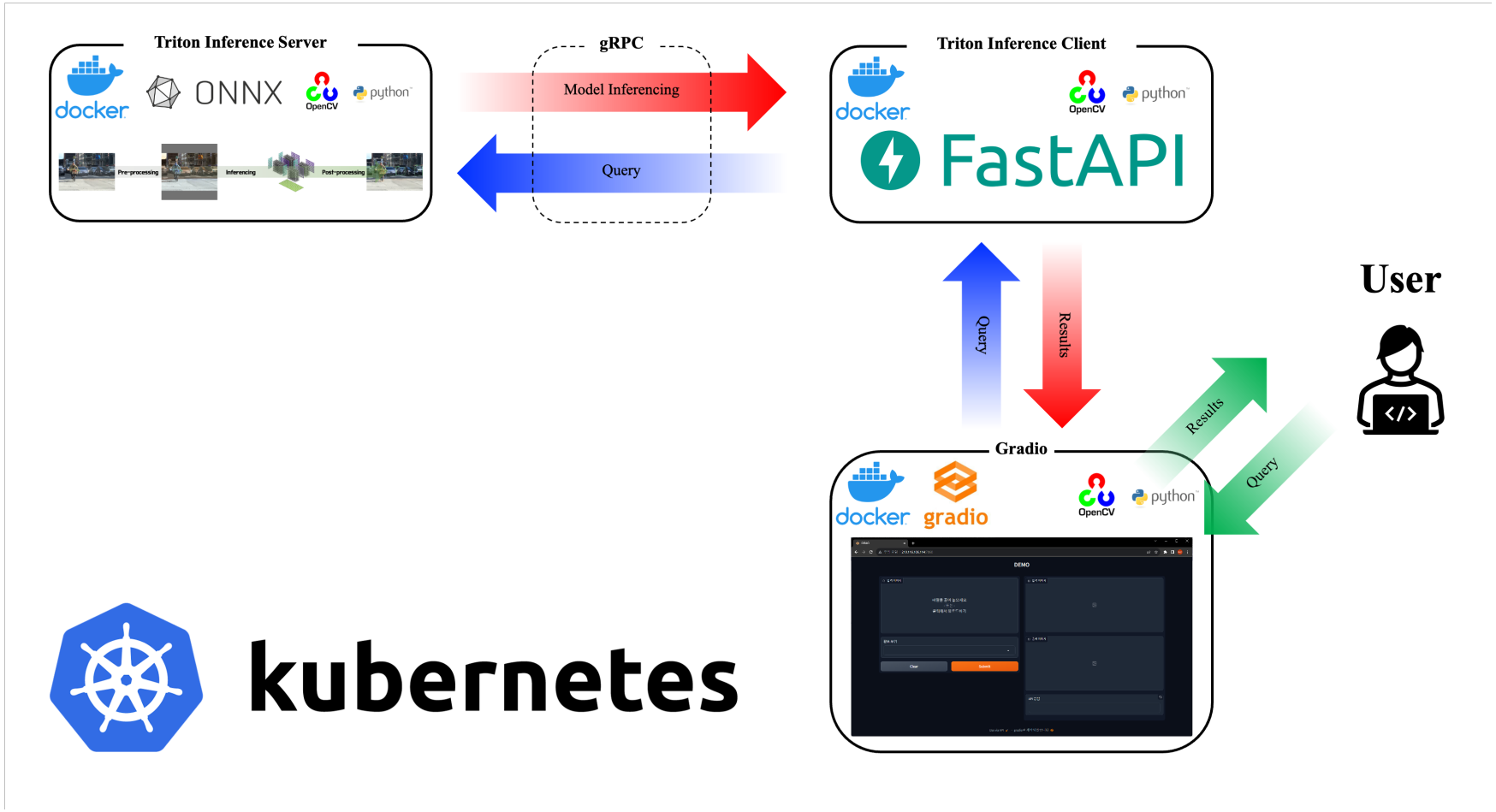
```
name: "YOLO"
platform: "ensemble"

input [
    {
        name: "INPUT_IMAGE"
        data_type: TYPE_UINT8
        dims: [-1,-1,-1]
    }
]

output [
    {
        name: "RESULTS"
        data_type: TYPE_FP32
        dims: [-1,-1]
    },
    {
        name: "VISUALIZE"
        data_type: TYPE_UINT8
        dims: [-1,-1,-1]
    }
]

ensemble_scheduling {
    step [
        {
        model_name: "preprocess"
        model_version: -1
        input_map {
        ...
        }
        output_map {
        ...
        }
        },
        {
        model_name: "model"
        model_version: -1
        input_map {
        ...
        }
        output_map {
        ...
        }
        },
        {
        model_name: "postprocess"
        model_version: -1
        input_map {
        ...
        }
        output_map {
        ...
        }
        }
    ]
}
```



MLOps

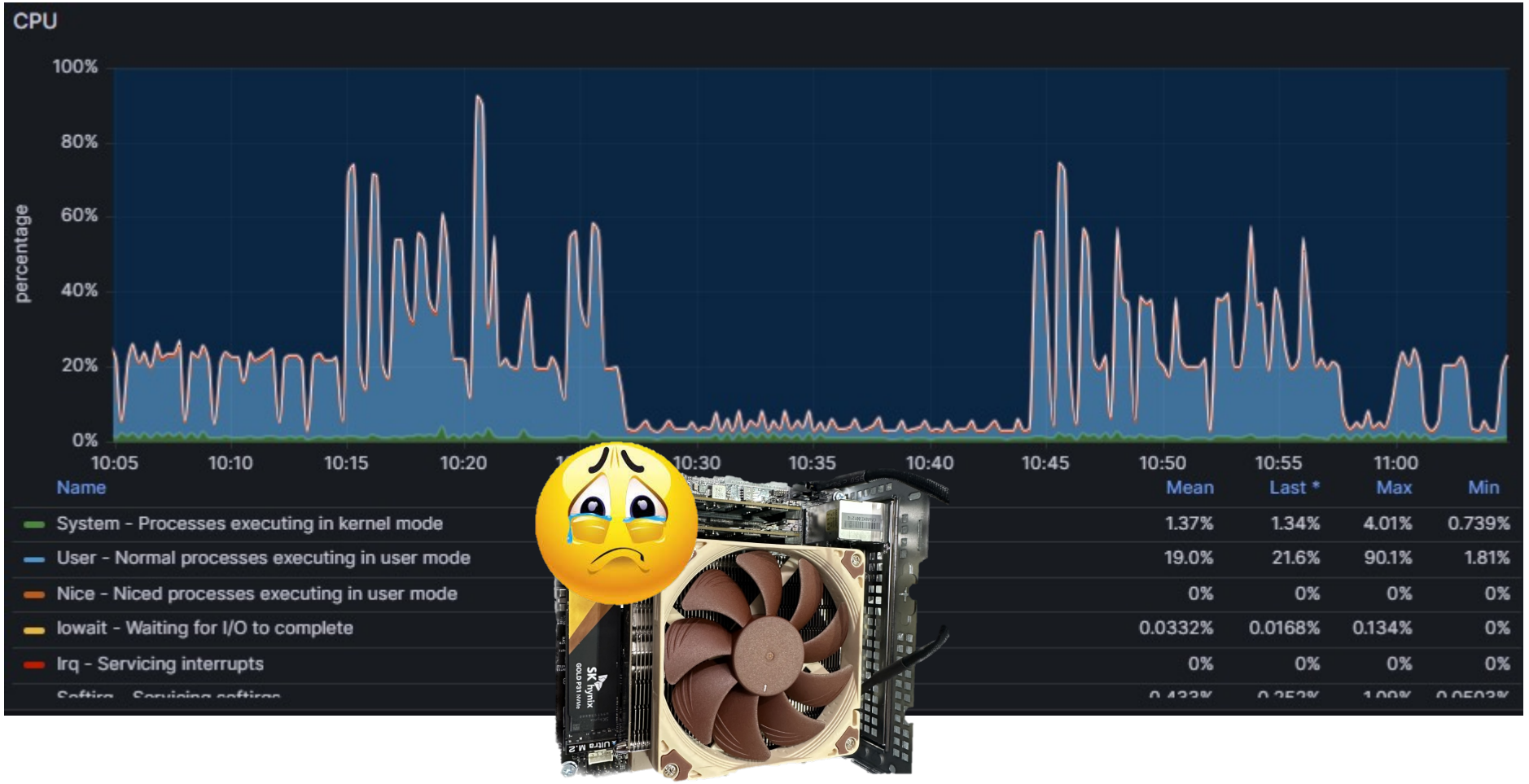
3.4. Gradio

- Machine learning model을 위한 user interface 생성 라이브러리



MLOps

Special Thanks



MLOps

Q & A