AI Platform with Python

From ML Infrastructure to Large-Scale Inference Pipeline

Park JunSeong, Byun Kyuhyun

ML Infrastructure with Python

박준성
Park JunSeong

- ML Infrastructure Team
    - Software Engineer

01

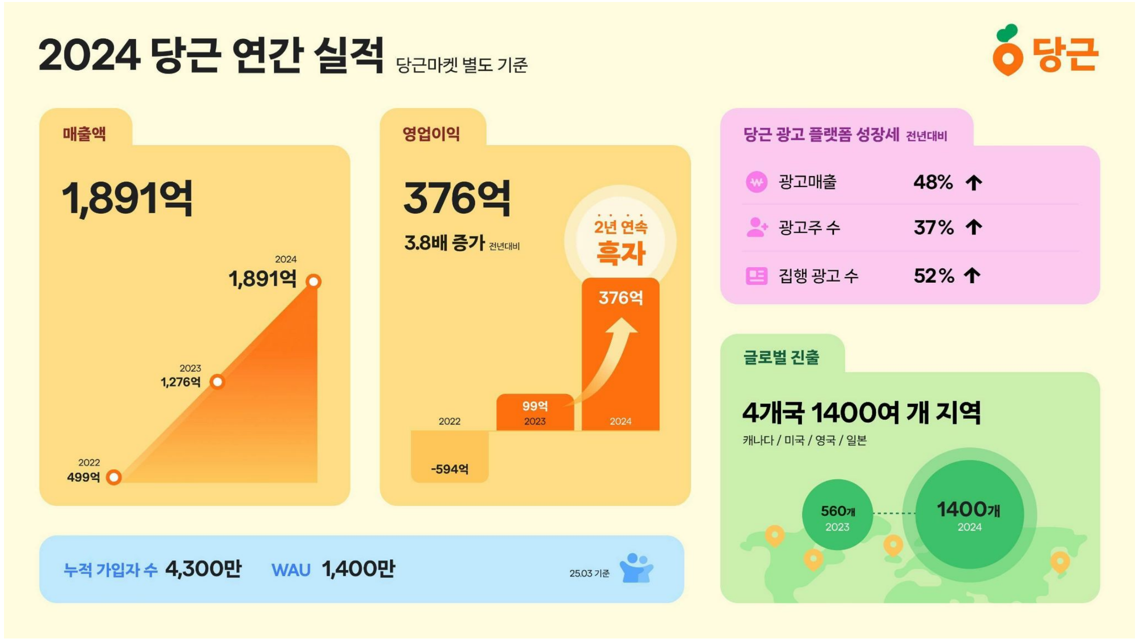
Service Growth with AI Models

Service Growth in 2024

with AI Models

3.8x YoY Operating Profit Growth

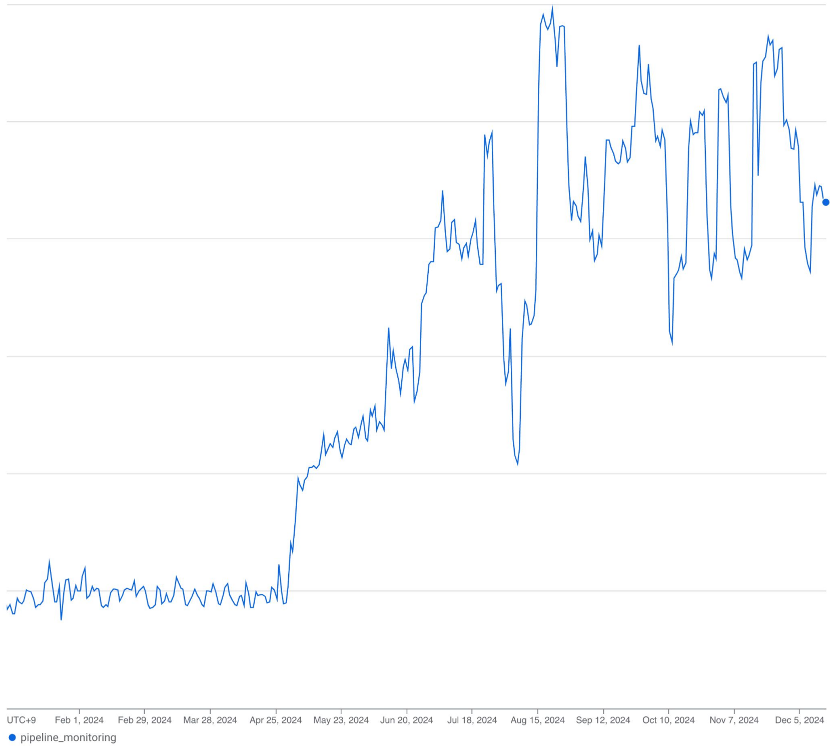
· Achieved 189.1 billion KRW in revenue
· Reached 43 million ARU, with 14 million WAU
· Advertising revenue up 48%
· Service available in Canada, the United States, the United Kingdom, and Japan



Source : https://about.daangn.com/company/pr/archive/당근-2024년-매출-1891억-원-영업이익-376억-원-기록/

Increasing Number of Training Pipelines

Thousands of daily pipeline increased by 3–4 times



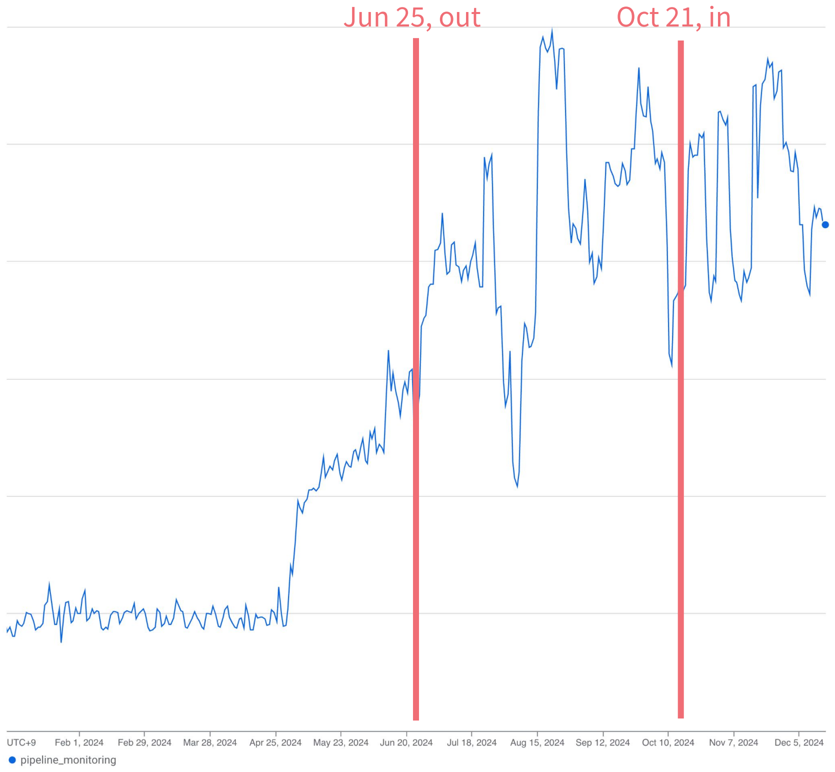
02

Operating ML Infrastructure

with Python

But… Limited Team Capacity

Growing fast with a limited team



But… Limited Team Capacity

Growing fast with a limited team

Like…
We're in the endgame now.

How to Manage Training Pipelines?

with Kubeflow + TFX

Kubeflow Pipelines

· ML Pipeline Orchestrator based on Argo Workflow
· Provides **extensibility and flexibility with Python**
· Run workflows through **reusable components**
· Supports TensorFlow, PyTorch, XGBoost, Etc

Kubeflow

Source : https://github.com/kubeflow/kubeflow

How to Manage Training Pipelines?

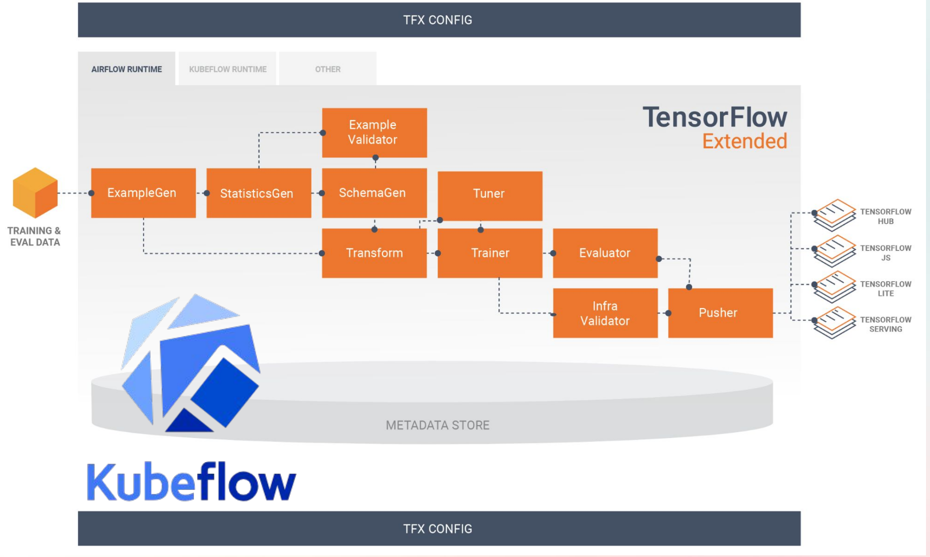
with Kubeflow + TFX

Kubeflow Pipelines

· ML Pipeline Orchestrator based on Argo Workflow
· Provides extensibility and flexibility with Python
· Run workflows through reusable components
· Supports **TensorFlow**, PyTorch, XGBoost, Etc

TFX(TensorFlow eXtended)

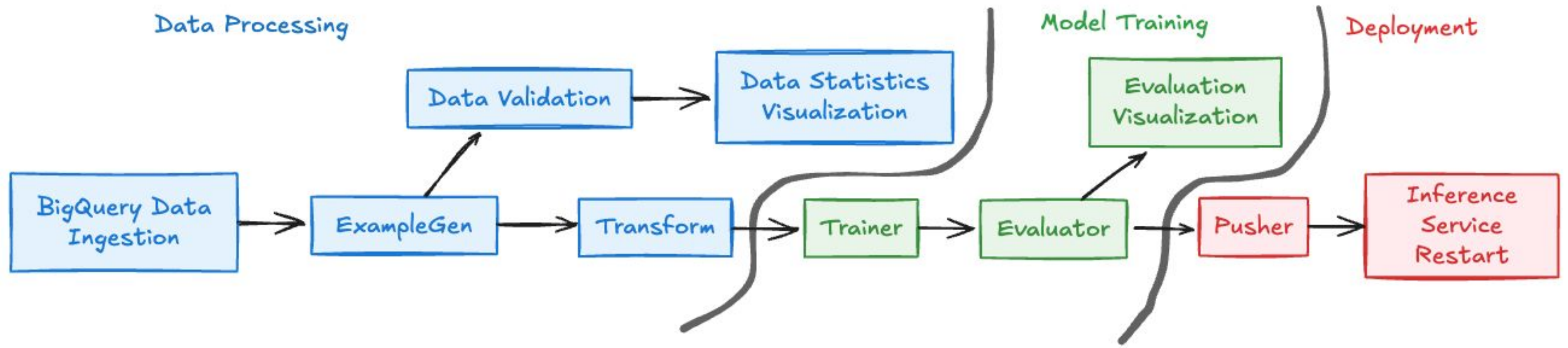
· End-to-end platform for ML pipelines
· Provides **a comprehensive set of components and libraries** to handle **various stages of an ML workflow**
· Supports **ML metadata for Kubeflow** and **Apache Beam** for distributed data processing and scalable workloads



Source : https://github.com/tensorflow/tfx

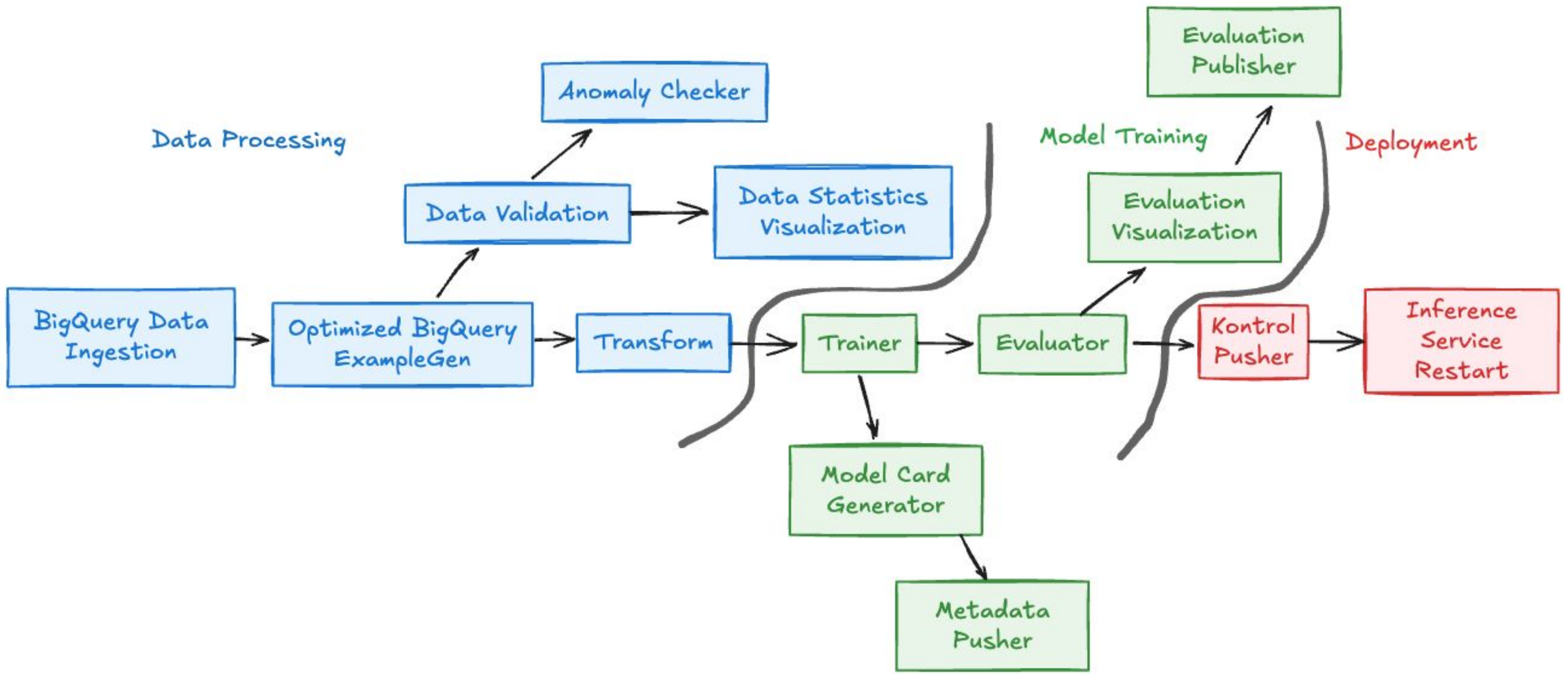
How to Manage Training Pipelines?

with Kubeflow + TFX



How to Manage Training Pipelines?

with Kubeflow + TFX

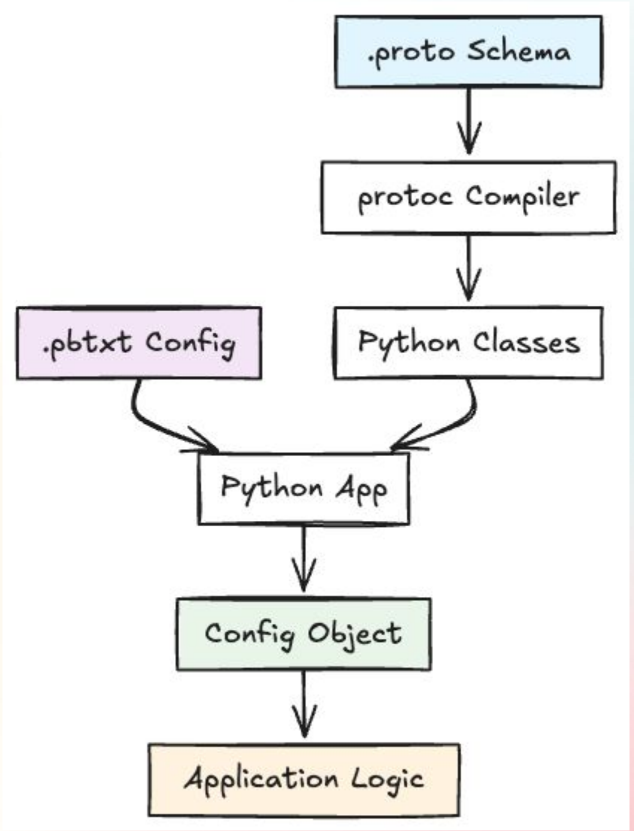


How to Manage Training Pipelines?

with Protobuf

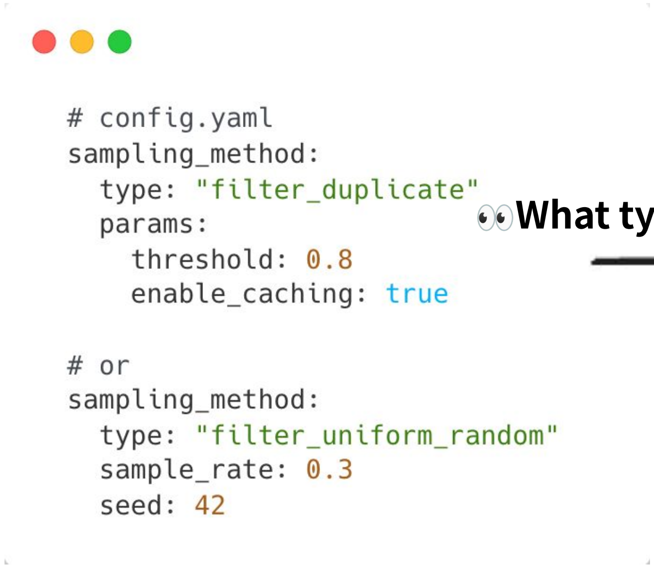
Configuration with Protobuf

- **Preventing type-related bugs and runtime errors** from **dynamic typing**
  - Protobuf based validation **eliminates additional checking code**
- Configuration **behavior understood** through Protobuf specs **without code analysis**
- **Backward-compatible field** addition/deletion on schema changes
- **Human readable** .pbtxt file



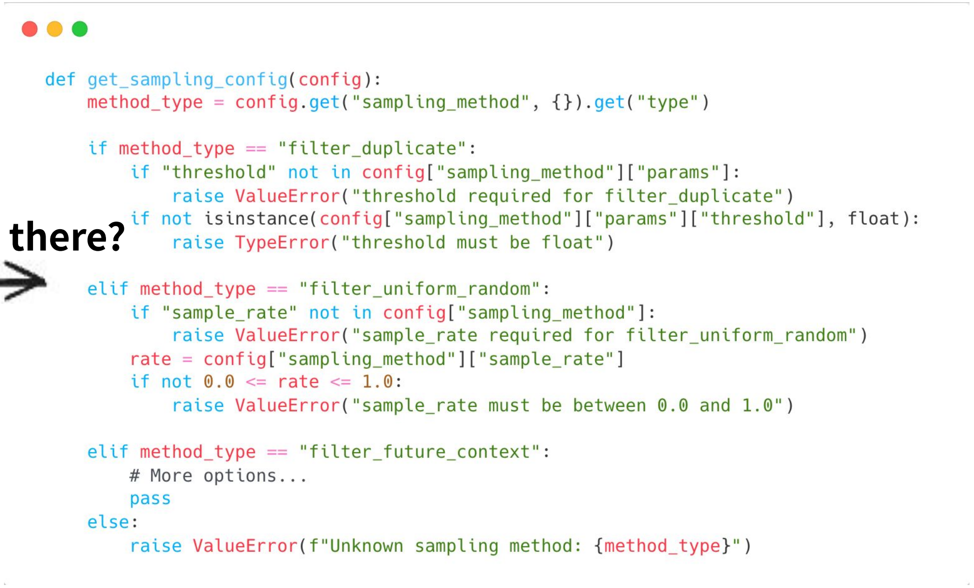
How to Manage Training Pipelines?

with Protobuf



```yaml
# config.yaml
sampling_method:
  type: "filter_duplicate"
  params:
    threshold: 0.8
    enable_caching: true

# or
sampling_method:
  type: "filter_uniform_random"
  sample_rate: 0.3
  seed: 42
```

👀What types are there?

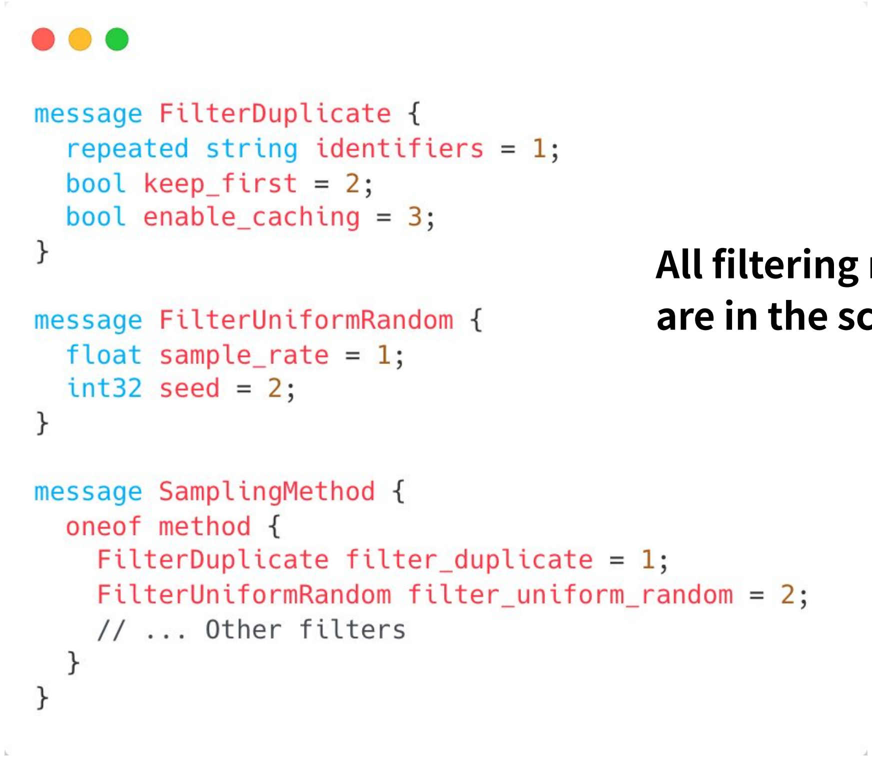


```python
def get_sampling_config(config):
    method_type = config.get("sampling_method", {}).get("type")

    if method_type == "filter_duplicate":
        if "threshold" not in config["sampling_method"]["params"]:
            raise ValueError("threshold required for filter_duplicate")
        if not isinstance(config["sampling_method"]["params"]["threshold"], float):
            raise TypeError("threshold must be float")

    elif method_type == "filter_uniform_random":
        if "sample_rate" not in config["sampling_method"]:
            raise ValueError("sample_rate required for filter_uniform_random")
        rate = config["sampling_method"]["sample_rate"]
        if not 0.0 <= rate <= 1.0:
            raise ValueError("sample_rate must be between 0.0 and 1.0")

    elif method_type == "filter_future_context":
        # More options...
        pass
    else:
        raise ValueError(f"Unknown sampling method: {method_type}")
```

Source : https://carbon.now.sh

How to Manage Training Pipelines?

with Protobuf



```
message FilterDuplicate {
  repeated string identifiers = 1;
  bool keep_first = 2;
  bool enable_caching = 3;
}

message FilterUniformRandom {
  float sample_rate = 1;
  int32 seed = 2;
}

message SamplingMethod {
  oneof method {
    FilterDuplicate filter_duplicate = 1;
    FilterUniformRandom filter_uniform_random = 2;
    // ... Other filters
  }
}
```

All filtering method types are in the schema!

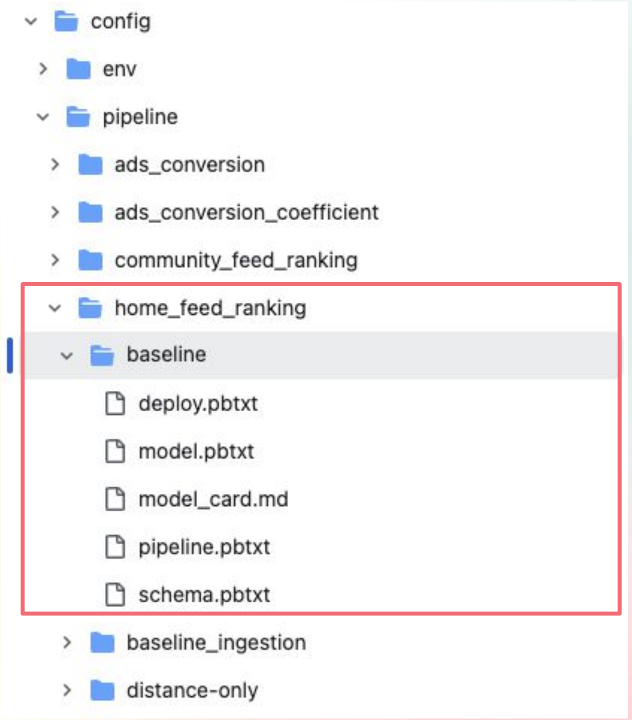
Source : https://carbon.now.sh

How to Manage Training Pipelines?

with Protobuf

Experiments with Protobuf

· **Reliable development** through Protobuf specifications reduces runtime errors
· **Single repository collaboration** enables reusability and knowledge sharing
· **Accelerated iteration cycles** enables faster experimentation and deployment
· **Reduces ML infrastructure operational burden** through standardized patterns



```
config
  env
  pipeline
    ads_conversion
    ads_conversion_coefficient
    community_feed_ranking
    home_feed_ranking
      baseline
        deploy.pbtxt
        model.pbtxt
        model_card.md
        pipeline.pbtxt
        schema.pbtxt
      baseline_ingestion
      distance-only
```

Source : Internal GitHub repository

How to Manage Training Pipelines?

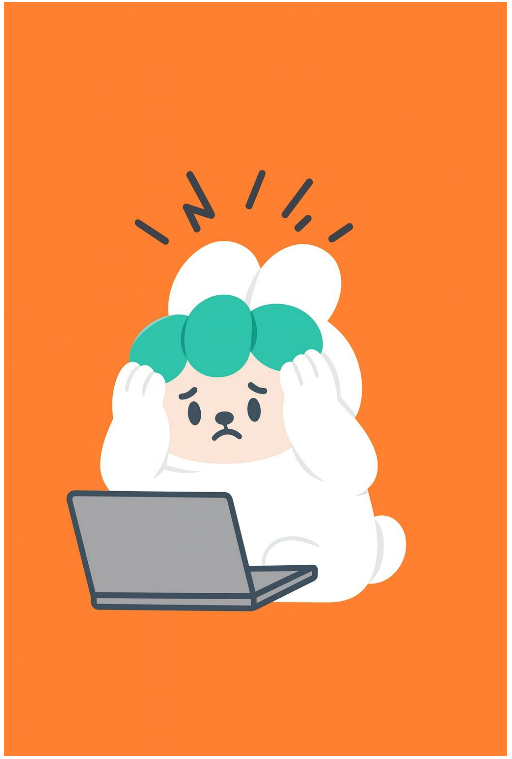
with GCP Vertex AI Pipelines

Monitoring?

Alert?

Oncall?

Network Failure?



Quota?

Autoscaler?

Kubeflow Problems?

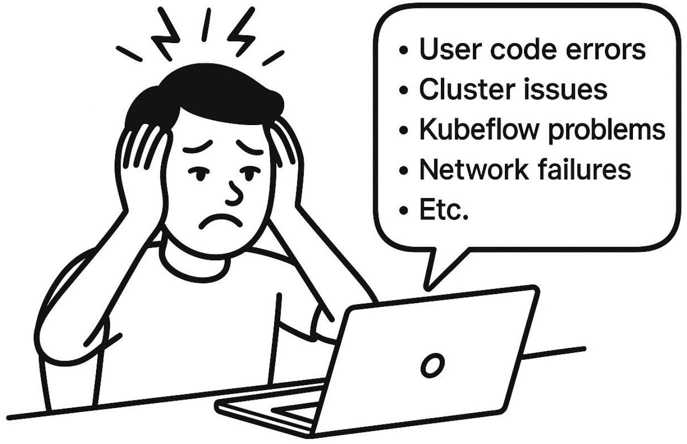
Kubernetes Problems?

How to Manage Training Pipelines?

with GCP Vertex AI Pipelines

Training pipeline Alert

· **Direct K8S/Kubeflow operations** require complex failure diagnosis
· **All failures trigger ML Infra team callouts**
· **Limited development time** for ML Infra improvements



ML Infrastructure Team Member

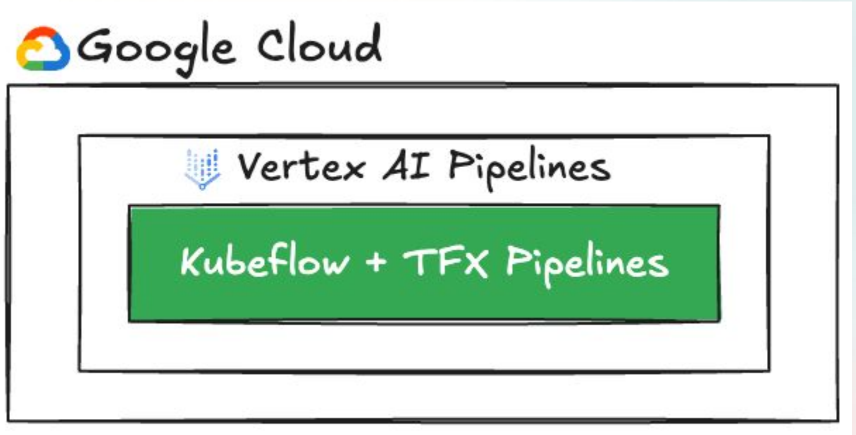
Source : ChatGPT

How to Manage Training Pipelines?

with GCP Vertex AI Pipelines

Training pipeline on Google Cloud Platform

- GCP Vertex AI Pipelines is **a serverless service** for ML Workflows
- Supports **Kubeflow Pipelines** and **TFX** framework
- **Reduced ML Infra team callouts** on pipeline failures
- **Eliminates operational burden**
  - No cluster management & upgrade
  - Auto-scaling within quotas
  - Easily differentiate between errors in user code and infrastructure



How to Manage Training Pipelines?

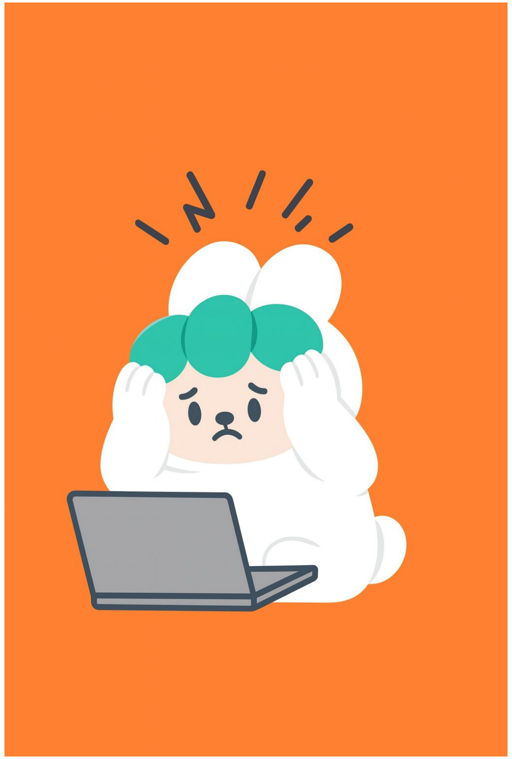
with GCP Vertex AI Pipelines

Monitoring?

Alert?

Oncall?

Network Failure?



Quota?

Autoscaler?

Kubeflow Problems?

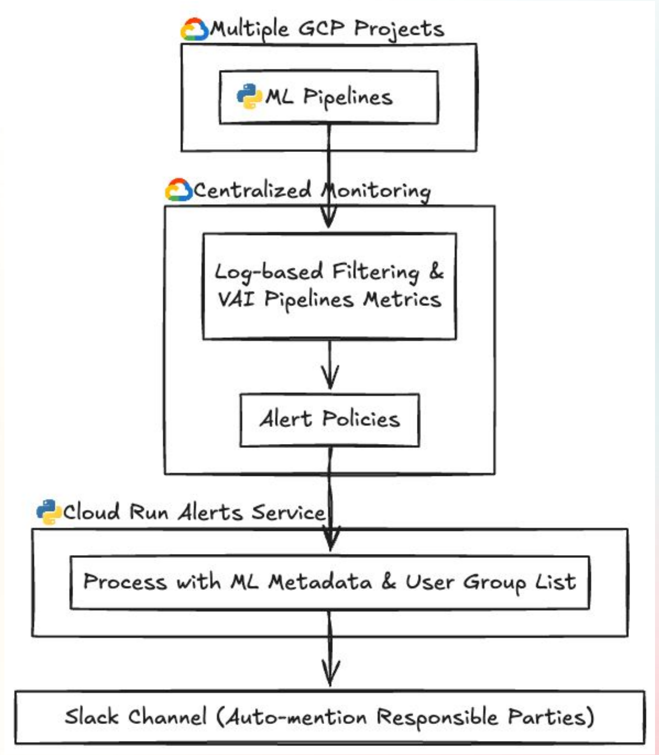
Kubernetes Problems?

How to Manage Training Pipelines?

with monitoring and alert

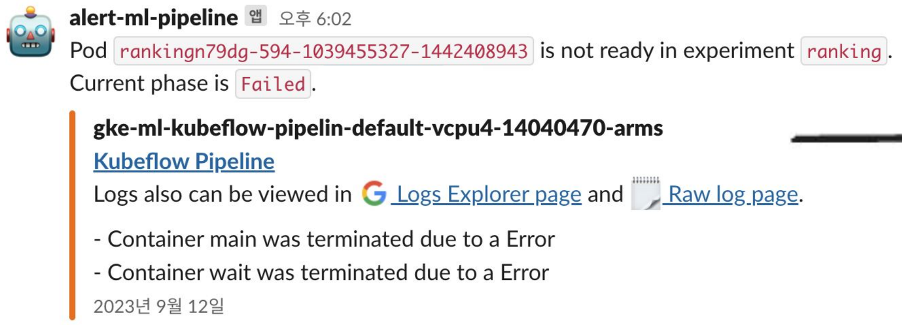
Python based monitoring

· **Collect ML pipeline logs from multiple GCP projects** via log-based filtering and Vertex AI Pipelines metrics
· **Create Alert Policies** from collected logs
· **Generate Slack alerts** via GCP Cloud Run **using Alert Policies and ML Metadata**
· **Auto-mention responsible parties and users** through SDK and user group lists


Multiple GCP Projects
ML Pipelines
Centralized Monitoring
Log-based Filtering & VAI Pipelines Metrics
Alert Policies
Cloud Run Alerts Service
Process with ML Metadata & User Group List
Slack Channel (Auto-mention Responsible Parties)

How to Manage Training Pipelines?

with monitoring and alert



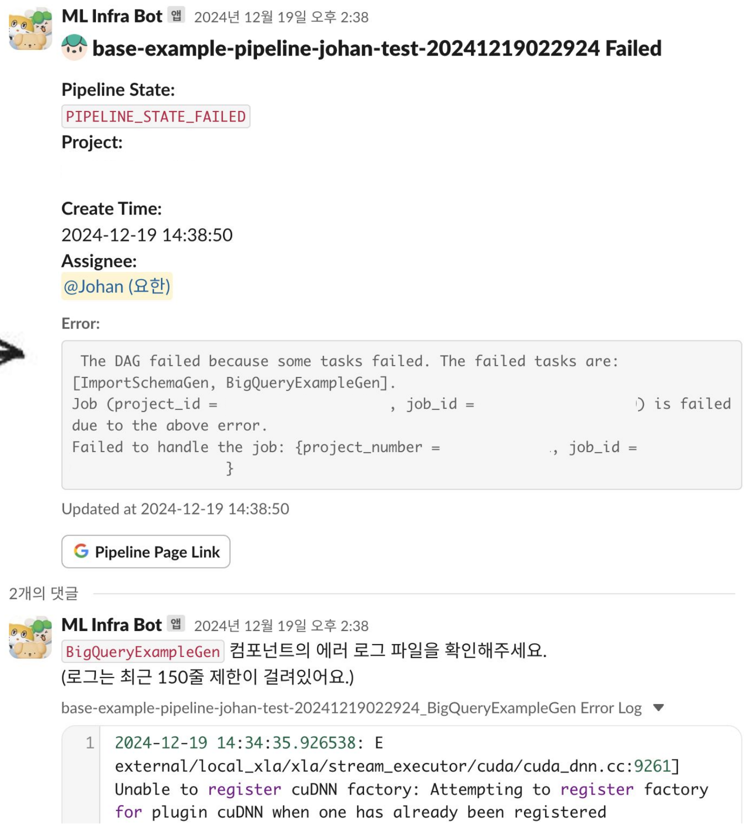
alert-ml-pipeline 앱 오후 6:02

Pod `rankingn79dg-594-1039455327-1442408943` is not ready in experiment `ranking`.
Current phase is `Failed`.

gke-ml-kubeflow-pipelin-default-vcpu4-14040470-arms
[Kubeflow Pipeline]()
Logs also can be viewed in [Logs Explorer page]() and [Raw log page]().

- Container main was terminated due to a Error
- Container wait was terminated due to a Error

2023년 9월 12일



ML Infra Bot 앱 2024년 12월 19일 오후 2:38

base-example-pipeline-johan-test-20241219022924 Failed

Pipeline State:
`PIPELINE_STATE_FAILED`
Project:

Create Time:
2024-12-19 14:38:50
Assignee:
@Johan (요한)

Error:

```
The DAG failed because some tasks failed. The failed tasks are:
[ImportSchemaGen, BigQueryExampleGen].
Job (project_id =                      , job_id =                    ) is failed
due to the above error.
Failed to handle the job: {project_number =              , job_id =
                  }
```

Updated at 2024-12-19 14:38:50

Pipeline Page Link

2개의 댓글

ML Infra Bot 앱 2024년 12월 19일 오후 2:38

`BigQueryExampleGen` 컴포넌트의 에러 로그 파일을 확인해주세요.
(로그는 최근 150줄 제한이 걸려있어요.)

base-example-pipeline-johan-test-20241219022924_BigQueryExampleGen Error Log

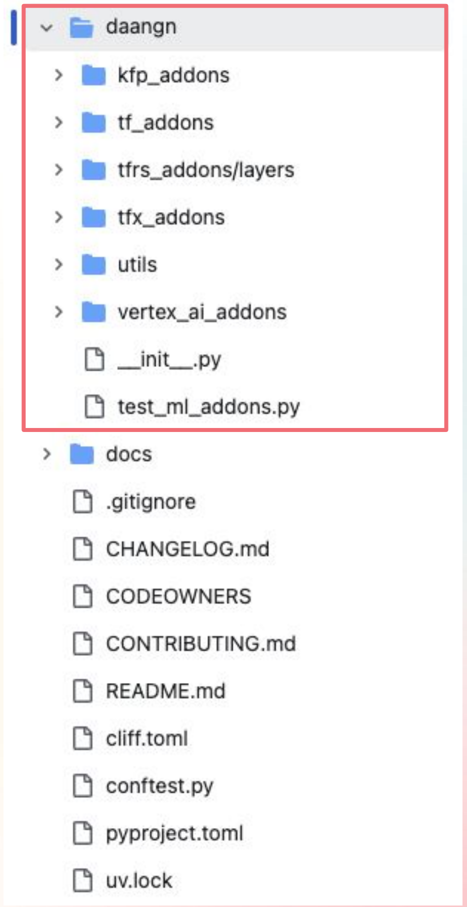
```
2024-12-19 14:34:35.926538: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261]
Unable to register cuDNN factory: Attempting to register factory
for plugin cuDNN when one has already been registered
```

How to Manage Training Pipelines?

with internal SDK

SDK for utility and reusability

· Frequent training pipeline elements as SDK with **cross-team contributions**
· **TFX custom components + additional ML pipeline utilities**
· **CalVer versioning (YYYY.MM.DD.timestamp)** with .dev suffix for development
· Modern Python packaging with **uv package manager and pyproject.toml**
· Python package **multi-cloud deployment** via GCP Artifact Registry and AWS Code Artifact
· **Operational configurations managed through Central Dogma**



```
daangn
  kfp_addons
  tf_addons
  tfrs_addons/layers
  tfx_addons
  utils
  vertex_ai_addons
  __init__.py
  test_ml_addons.py
docs
.gitignore
CHANGELOG.md
CODEOWNERS
CONTRIBUTING.md
README.md
cliff.toml
conftest.py
pyproject.toml
uv.lock
```

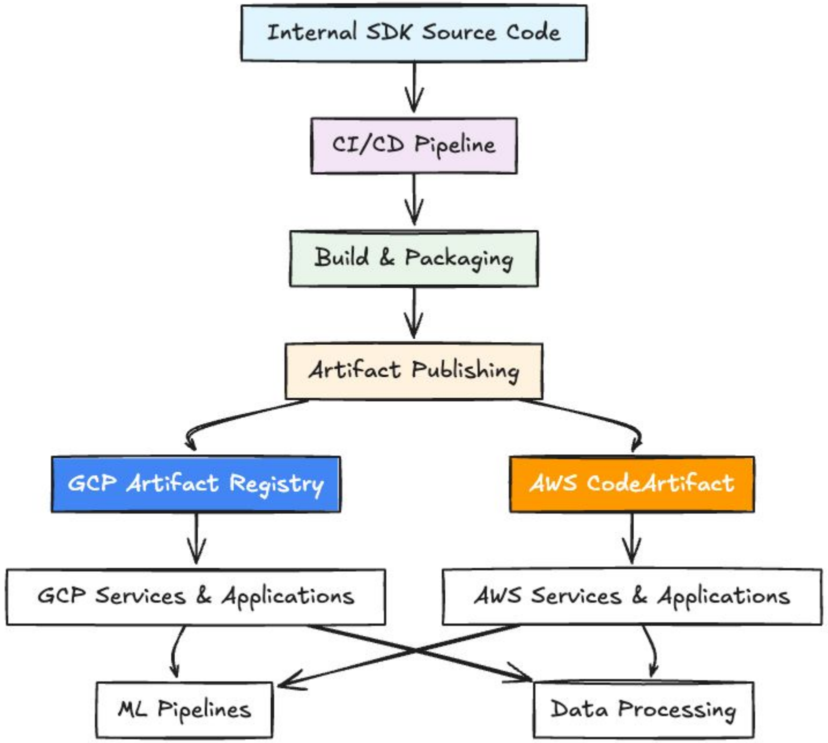
Source : Inhouse GitHub repository

How to Manage Training Pipelines?

with internal SDK

SDK for utility and reusability

· Frequent training pipeline elements as SDK with **cross-team contributions**
· **TFX custom components + additional ML pipeline utilities**
· **CalVer versioning (YYYY.MM.DD.timestamp)** with .dev suffix for development
· Modern Python packaging with **uv package manager and pyproject.toml**
· Python package **multi-cloud deployment** via GCP Artifact Registry and AWS Code Artifact
· **Operational configurations managed through Central Dogma**



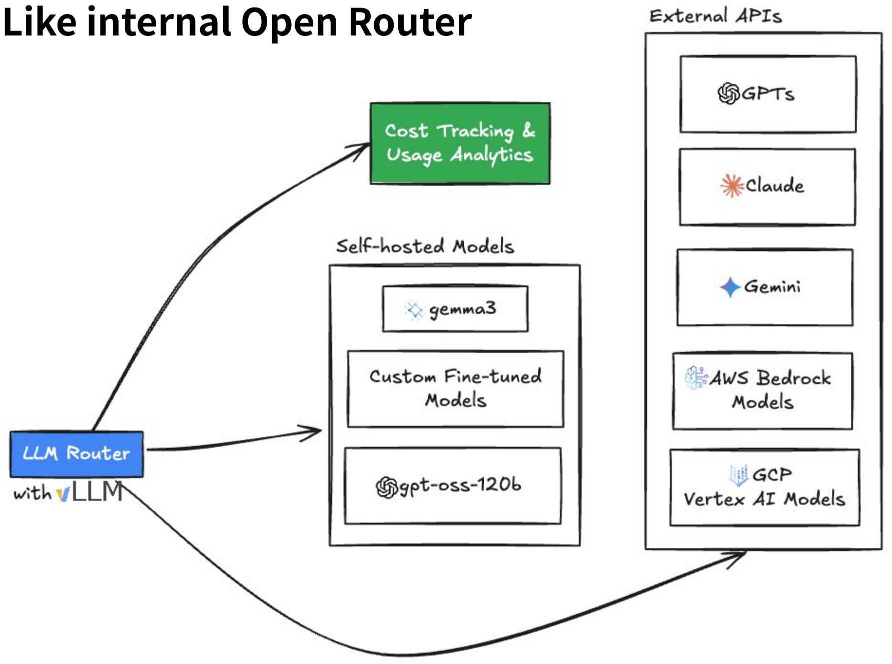
03

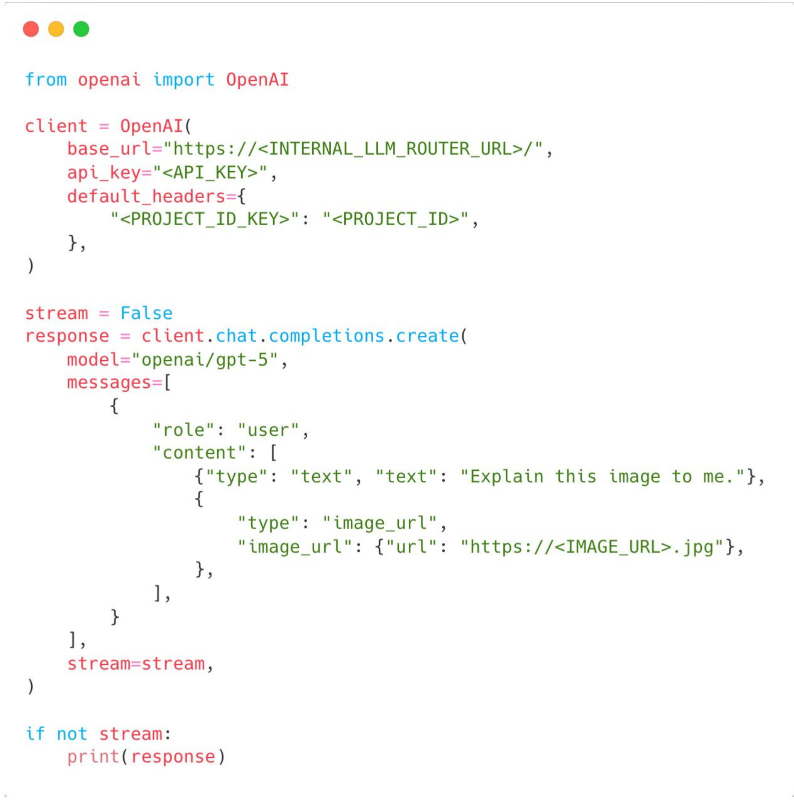
More Time, More Projects

LLM Router

For self-hosted models and external LLM API usage

Like internal Open Router



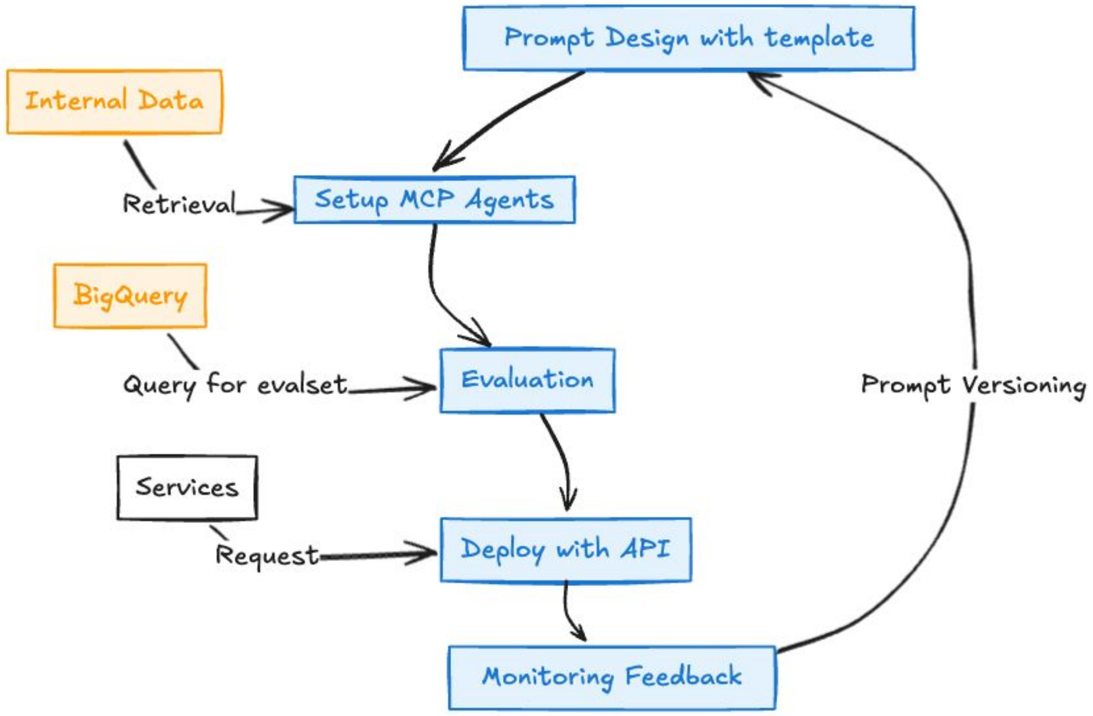


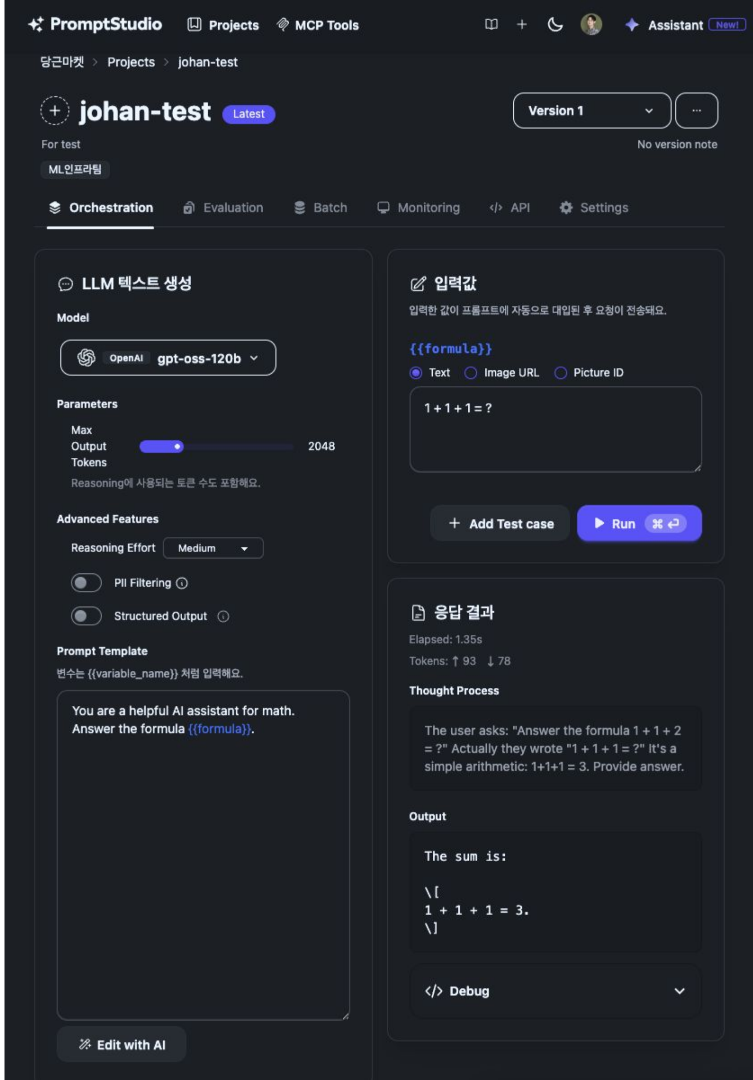
```python
from openai import OpenAI

client = OpenAI(
    base_url="https://<INTERNAL_LLM_ROUTER_URL>/",
    api_key="<API_KEY>",
    default_headers={
        "<PROJECT_ID_KEY>": "<PROJECT_ID>",
    },
)

stream = False
response = client.chat.completions.create(
    model="openai/gpt-5",
    messages=[
        {
            "role": "user",
            "content": [
                {"type": "text", "text": "Explain this image to me."},
                {
                    "type": "image_url",
                    "image_url": {"url": "https://<IMAGE_URL>.jpg"},
                },
            ],
        }
    ],
    stream=stream,
)

if not stream:
    print(response)
```

Prompt Management Platform

For service with LLM

Prompt Studio integrated with LLM Router





johan-test Latest

For test

ML인프라팀

Version 1

No version note

LLM 텍스트 생성

Model

OpenAI gpt-oss-120b

Parameters

Max Output Tokens 2048

Reasoning에 사용되는 토큰 수도 포함해요.

Advanced Features

Reasoning Effort Medium

PII Filtering

Structured Output

Prompt Template

변수는 {{variable_name}} 처럼 입력해요.

You are a helpful AI assistant for math.
Answer the formula {{formula}}.

Edit with AI

입력값

입력한 값이 프롬프트에 자동으로 대입된 후 요청이 전송돼요.

{{formula}}

[✓ Text] Image URL Picture ID

1 + 1 + 1 = ?

Add Test case

Run ⌘ ↩

응답 결과

Elapsed: 1.35s

Tokens: ↑ 93 ↓ 78

Thought Process

The user asks: "Answer the formula 1 + 1 + 2 = ?" Actually they wrote "1 + 1 + 1 = ?" It's a simple arithmetic: 1+1+1 = 3. Provide answer.

Output

```
The sum is:

\[
1 + 1 + 1 = 3.
\]
```

Debug

Custom Builds

For optimization and internal usage

TensorFlow IO

- S3 Native integration with aws-sdk-cpp

TensorFlow Serving

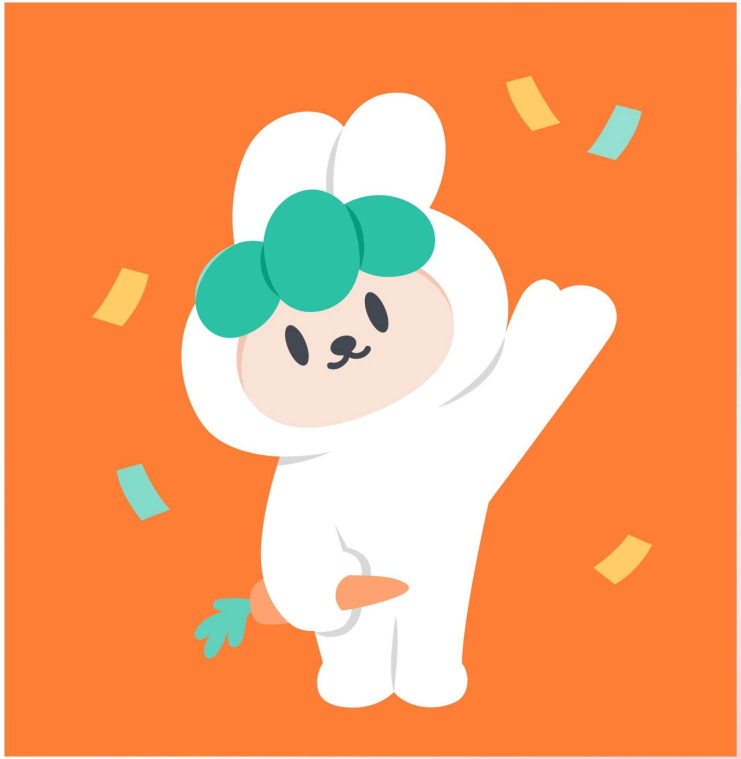
- Tensorflow Runtime support
- ARM architecture compatibility patches
- S3 file system integration

ScaNN

- gcc-10 compatibility patches
- Supports TensorFlow Serving 2.17 version

Optimized TFX Components

- optimized component for Apache Beam



Inference Pipeline with Apache Beam Python

변규현
Byun Kyuhyun

- ML Data Platform Team
    - Software Engineer
- AWS Serverless HERO

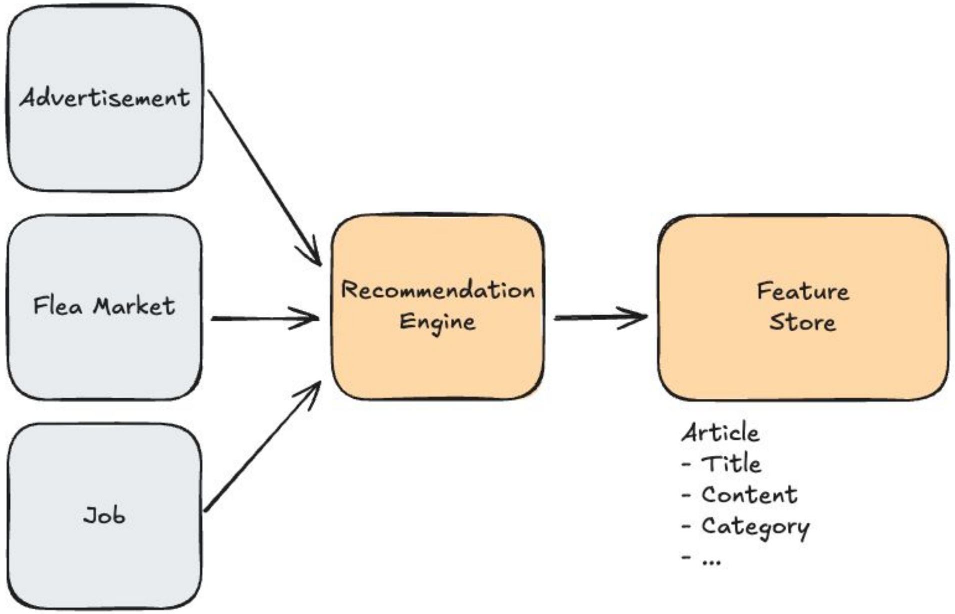
01

The Shift to Embedding-based Systems

Why We Need Embedding Data

From keyword search to semantic understanding

- Previously, recommendations were driven by a keyword-based approach



From Traditional Features to ANN-based Recommendations

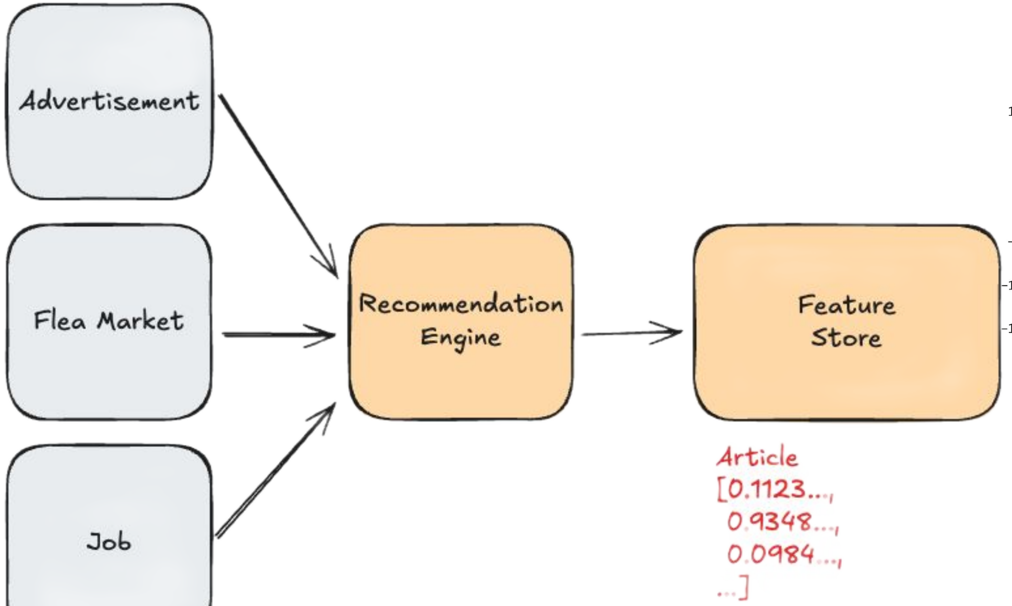
How vector representations transform the recommendation process

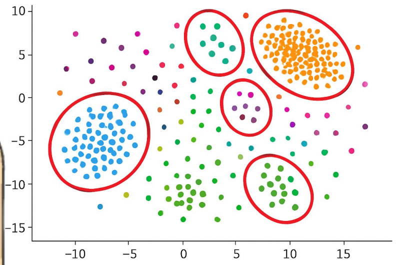
- Before
  - Recommendations relied on manually engineered features like category IDs, keyword tags, or numerical scores.
  - The matching process often used **rule-based filtering** or exact matching in structured fields.
  - **Similarity between items was limited** to **predefined attributes** (e.g., same category, matching title keyword).

Why We Need Embedding Data

From keyword search to semantic understanding

- With LLM: Generate embeddings for data representation





From Traditional Features to ANN-based Recommendations

How vector representations transform the recommendation process

- After (with Embeddings + ANN)
  - **Each item** is represented as **a dense vector embedding**, capturing **semantic meaning** from content, images, or user interactions.
  - Instead of exact keyword match, Approximate Nearest Neighbor (ANN) **search finds items closest in vector space**.
  - This enables recommendations based on **semantic similarity** (e.g., “visually similar”, “conceptually related”), **even if the metadata doesn’t match exactly**.

How Embeddings Change the Data Handling

A new way to represent and process information

- Data handling changes from keyword/text matching to vector-based matching
- **Embeddings provide flexibility in representation**
- Can be used across search, recommendation, and classification tasks

From Traditional Features to ANN-based Recommendations

Capabilities unlocked by embedding-based recommendations

- Enable these...
  - **Unlocks multi-modal recommendations** (text, image, audio, behavior data).
  - **Supports cold-start scenarios by leveraging embedding similarity** instead of relying solely on historical interactions.
  - More **flexible** and **scalable** than manual feature engineering.

02

Story of the Inference Pipeline

Product requirements

Key requirements for our inference pipeline

- Process **billions of records within hours**

Product requirements

Key requirements for our inference pipeline

- Process **billions of records within hours**
- **GPU-powered inference** with various embedding models

Product requirements

Key requirements for our inference pipeline

- Process **billions of records within hours**
- **GPU-powered inference** with various embedding models
- **Dynamic scaling** based on data volume

Product requirements

Key requirements for our inference pipeline

- Process **billions of records within hours**
- **GPU-powered inference** with various embedding models
- **Dynamic scaling** based on data volume
- Develop in **Python**
- **Minimal** infrastructure management

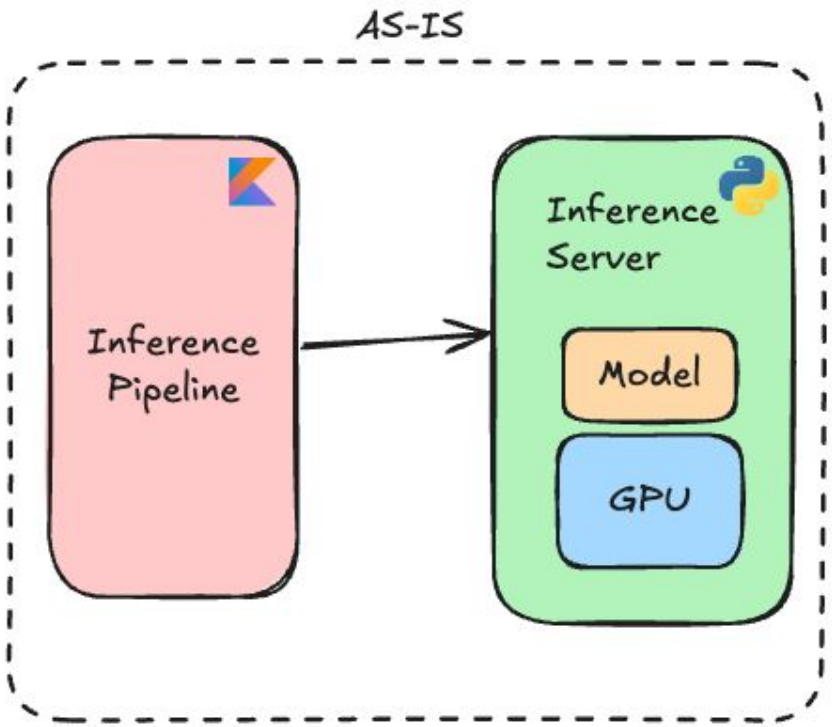
Product requirements

Key requirements for our inference pipeline

- Process **billions of records within hours**
- **GPU-powered inference** with various embedding models
- **Dynamic scaling** based on data volume
- Develop in **Python**
- **Minimal** infrastructure management
- Utilize BigQuery datasets and GCS images

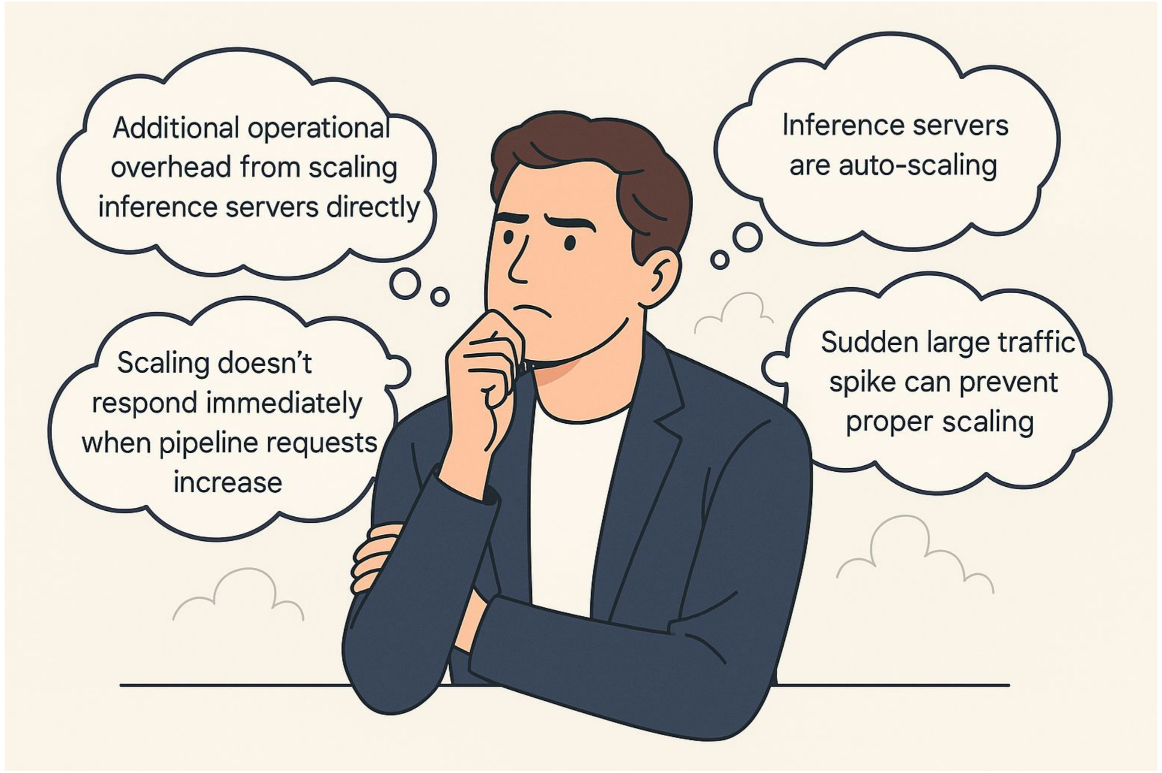
Product requirements

From Separate Inference Servers to Integrated Pipelines



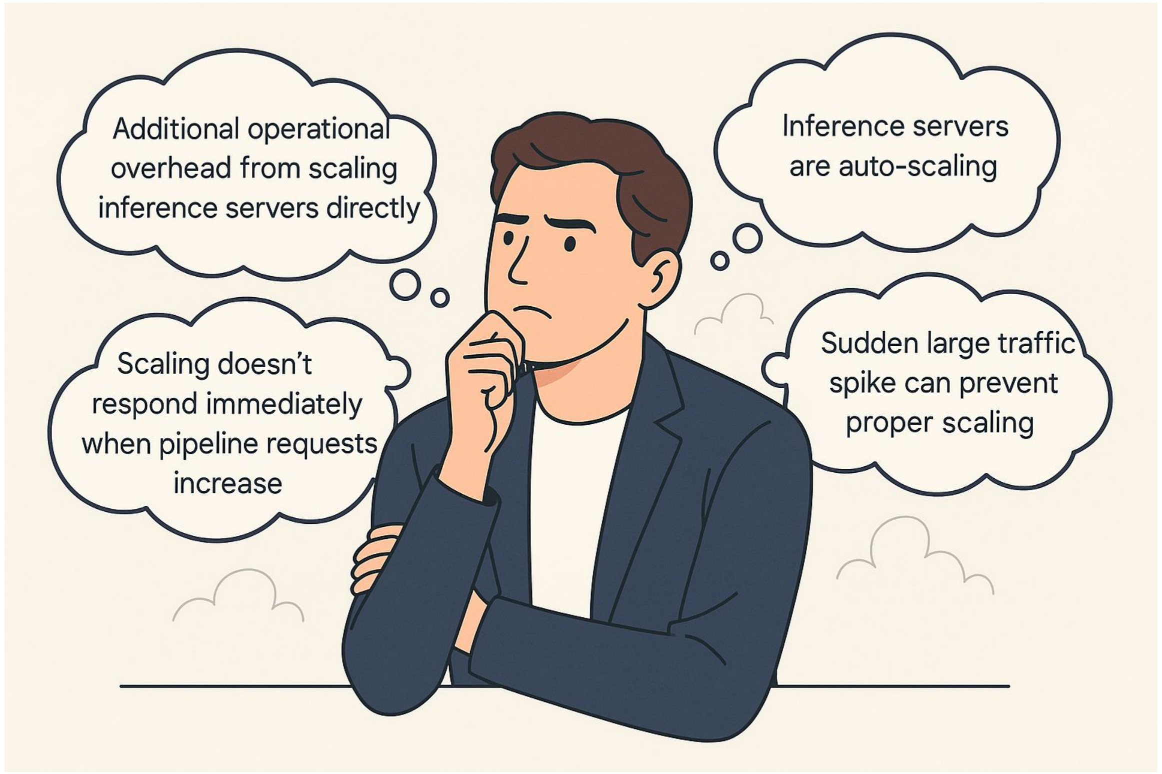
Product requirements

POV of an ML Engineer working on the old inference pipeline


Additional operational overhead from scaling inference servers directly
Inference servers are auto-scaling
Scaling doesn't respond immediately when pipeline requests increase
Sudden large traffic spike can prevent proper scaling

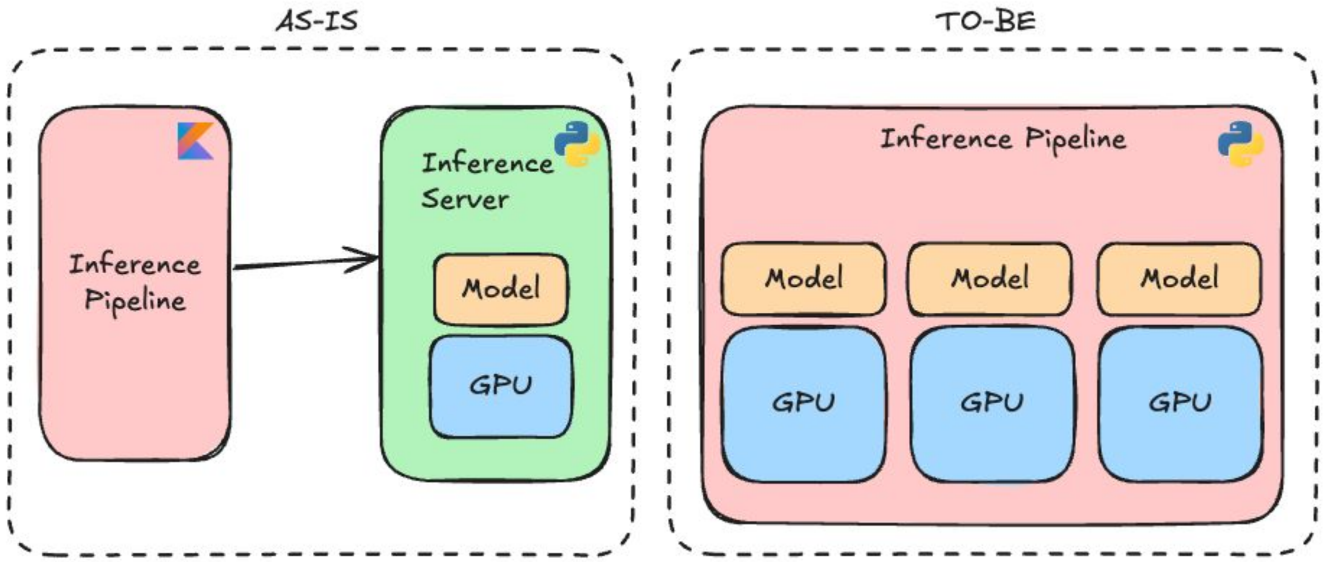
Product requirements

POV of an ML Engineer working on the old inference pipeline


Additional operational overhead from scaling inference servers directly
Inference servers are auto-scaling
Scaling doesn't respond immediately when pipeline requests increase
Sudden large traffic spike can prevent proper scaling

Product requirements

From Separate Inference Servers to Integrated Pipelines



Solution Candidates

Evaluation of Candidates

Criteria	Beam+Dataflow	Spark+DataProc	Flink
Large-scale batch support	**Fully auto**	Configure algorithm factors	Streaming Focus
GPU usage	**Custom container**	**Native GPU**	Limited GPU
Python Support	**Beam SDK**	Pyspark	Limited PyFlink
Infra managemet	**Serverless**	Cluster	Cluster + Complex Config
GCP integration	**Native BQ/GCS**	**SDK support**	Extra setup

Introduction to Apache Beam

Apache Beam: Write once, run anywhere

- **Unified programming model** for batch and streaming data processing
- Allows you to **write your pipeline once and run it on different runners** (e.g., Google Dataflow, Apache Spark, Flink)
- Supports multiple languages, including **Python**, Java, and Go
- Portable, scalable, and **integrates well with cloud services**

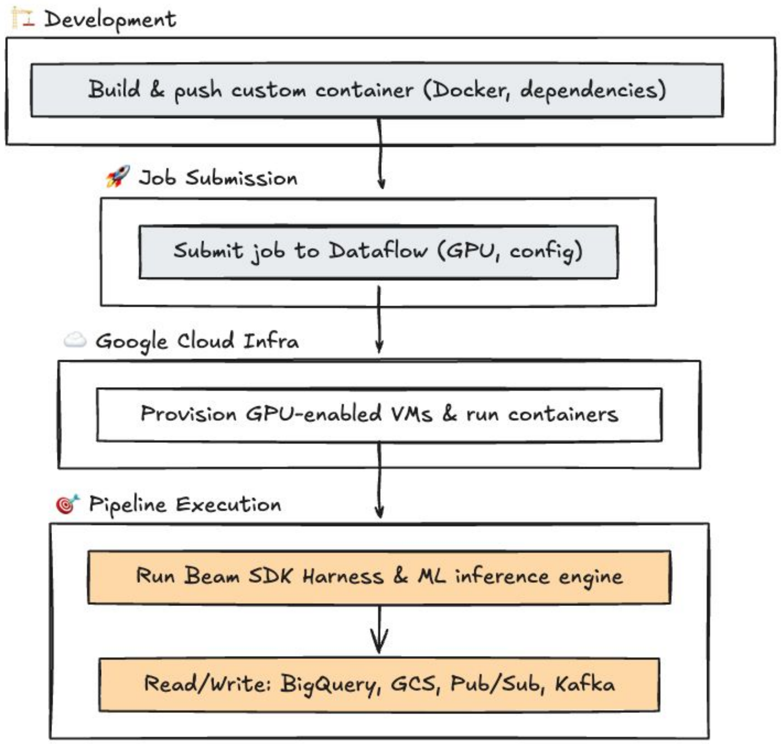
Introduction to Google Cloud Dataflow

Serverless data processing at scale

- **Fully managed**, **serverless** data processing service on **Google Cloud**
- Runs Apache Beam pipelines for both batch and streaming workloads
- **Automatically handles resource provisioning**, scaling, and optimization
- Integrates seamlessly with **BigQuery, Cloud Storage, Pub/Sub**, and more
- Supports multiple languages via Apache Beam SDK (**Python**, Java, Go)

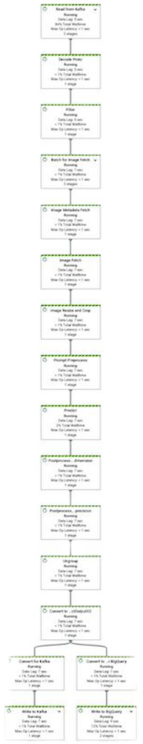
Pipeline Execution Flow of Dataflow

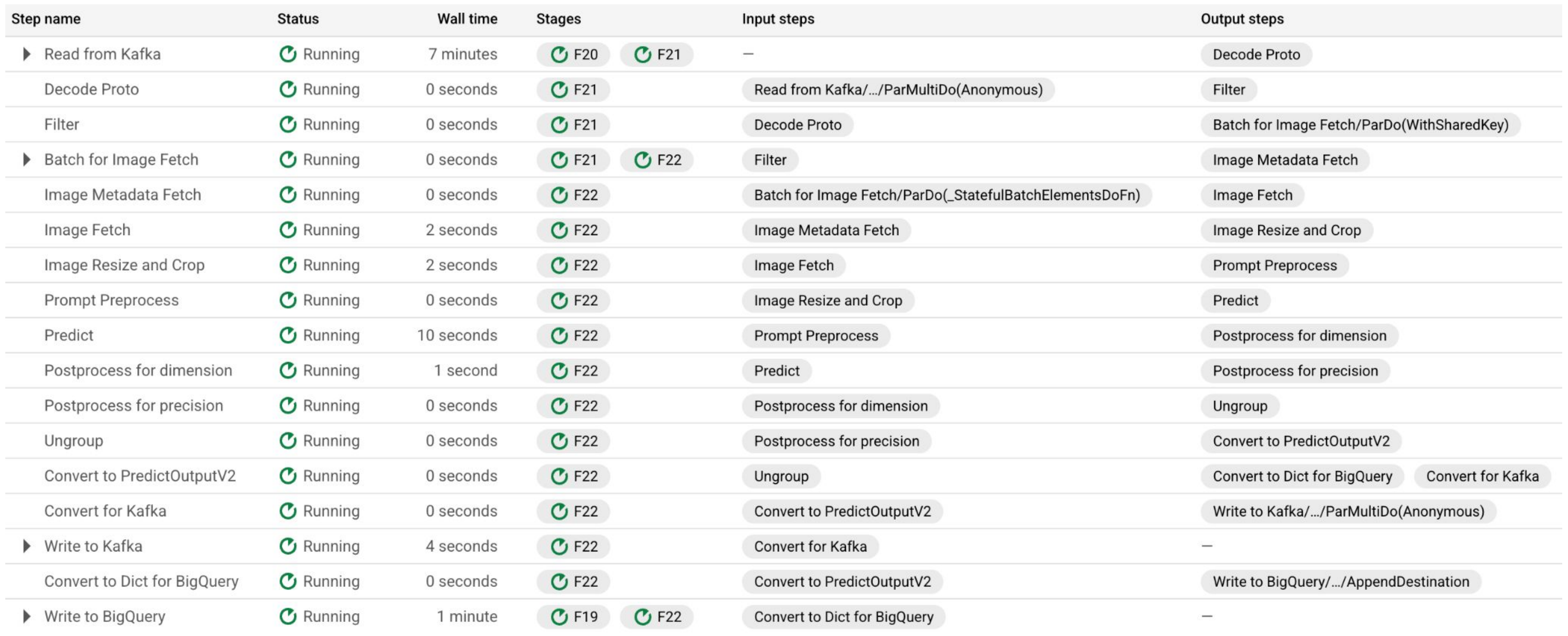
End-to-end execution path of an Apache Beam job on Google Cloud Dataflow


Development
Build & push custom container (Docker, dependencies)
Job Submission
Submit job to Dataflow (GPU, config)
Google Cloud Infra
Provision GPU-enabled VMs & run containers
Pipeline Execution
Run Beam SDK Harness & ML inference engine
Read/Write: BigQuery, GCS, Pub/Sub, Kafka

Pipeline Execution Flow of Dataflow

Example of production service





Step name	Status	Wall time	Stages	Input steps	Output steps
▸ Read from Kafka	Running	7 minutes	F20 F21	–	Decode Proto
Decode Proto	Running	0 seconds	F21	Read from Kafka/.../ParMultiDo(Anonymous)	Filter
Filter	Running	0 seconds	F21	Decode Proto	Batch for Image Fetch/ParDo(WithSharedKey)
▸ Batch for Image Fetch	Running	0 seconds	F21 F22	Filter	Image Metadata Fetch
Image Metadata Fetch	Running	0 seconds	F22	Batch for Image Fetch/ParDo(_StatefulBatchElementsDoFn)	Image Fetch
Image Fetch	Running	2 seconds	F22	Image Metadata Fetch	Image Resize and Crop
Image Resize and Crop	Running	2 seconds	F22	Image Fetch	Prompt Preprocess
Prompt Preprocess	Running	0 seconds	F22	Image Resize and Crop	Predict
Predict	Running	10 seconds	F22	Prompt Preprocess	Postprocess for dimension
Postprocess for dimension	Running	1 second	F22	Predict	Postprocess for precision
Postprocess for precision	Running	0 seconds	F22	Postprocess for dimension	Ungroup
Ungroup	Running	0 seconds	F22	Postprocess for precision	Convert to PredictOutputV2
Convert to PredictOutputV2	Running	0 seconds	F22	Ungroup	Convert to Dict for BigQuery Convert for Kafka
Convert for Kafka	Running	0 seconds	F22	Convert to PredictOutputV2	Write to Kafka/.../ParMultiDo(Anonymous)
▸ Write to Kafka	Running	4 seconds	F22	Convert for Kafka	–
Convert to Dict for BigQuery	Running	0 seconds	F22	Convert to PredictOutputV2	Write to BigQuery/.../AppendDestination
▸ Write to BigQuery	Running	1 minute	F19 F22	Convert to Dict for BigQuery	–

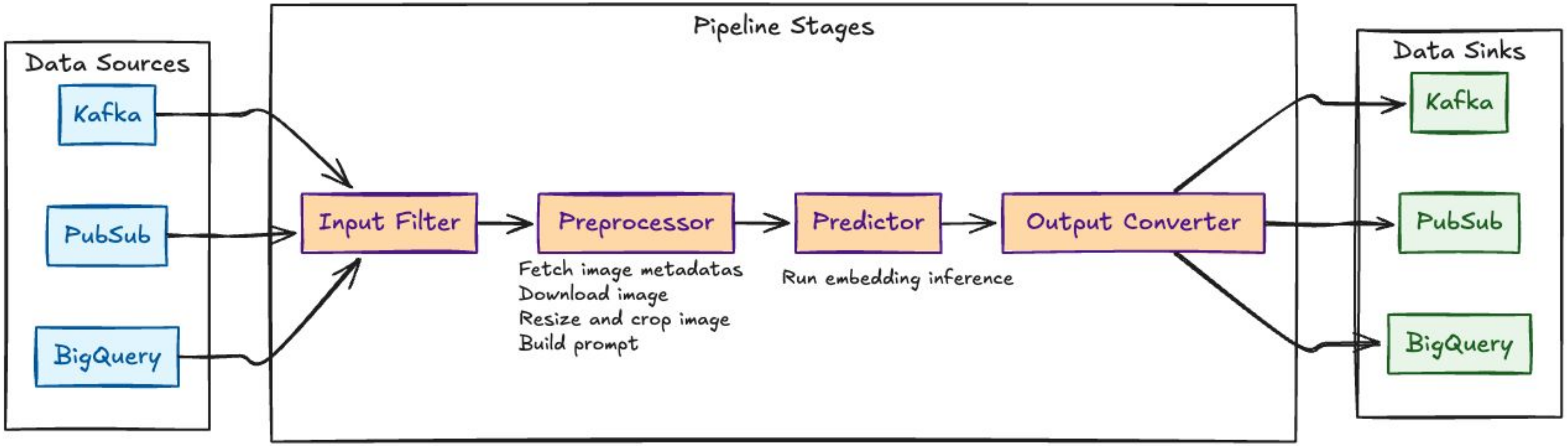
Pipeline Execution Flow of Dataflow

Example of the pipeline

```
input_collection = pipeline | "Read from Kafka" >> ReadFromKafka(
    topics=["my_topic"],
    consumer_config={... },
)
image_processed_collection = input_collection | "Image Process" >> ParDo(ImageProcessor(...))
prompt_processed_collection = image_processed_collection | "Prompt Process" >> ParDo(PromptProcessor())
predicted_collection = prompt_processed_collection | "Predict" >> ParDo(Predictor(...))
postprocessed_collection = predicted_collection | "Postprocess" >> ParDo(Postprocessor(...))
postprocessed_collection | "Converter 1" >> ParDo(...) | "Write to BigQuery" >> WriteToBigQuery(...)
postprocessed_collection | "Converter 2" >> ParDo(...) | "Write to Kafka" >> WriteToKafka(...)
```

Code Architecture

Contribution-friendly and easy-to-understand patterns



Code Architecture

Contribution-friendly and easy-to-understand patterns

```
├── client/          # 외부 서비스 클라이언트 (GCS, Redis, BigPicture)
├── inputfilter/     # 데이터 소스 필터링 및 검증
├── outputconverter/ # 예측 결과 형식 변환
├── pipelines/       # 실제 파이프라인 구현체
├── postprocessor/   # 출력 후 추가 처리 로직
├── predictor/       # ML 모델 예측 실행
├── preprocessor/    # 데이터 전처리 및 정제
├── record/          # 데이터 모델 정의
├── scheme/          # BigQuery 스키마 정의
├── script/          # 테스트용 스크립트 (미사용)
├── sink/            # 데이터 출력 대상
├── source/          # 데이터 입력 소스
└── util/            # 공통 유틸리티
```

03

Practical Performance Tips

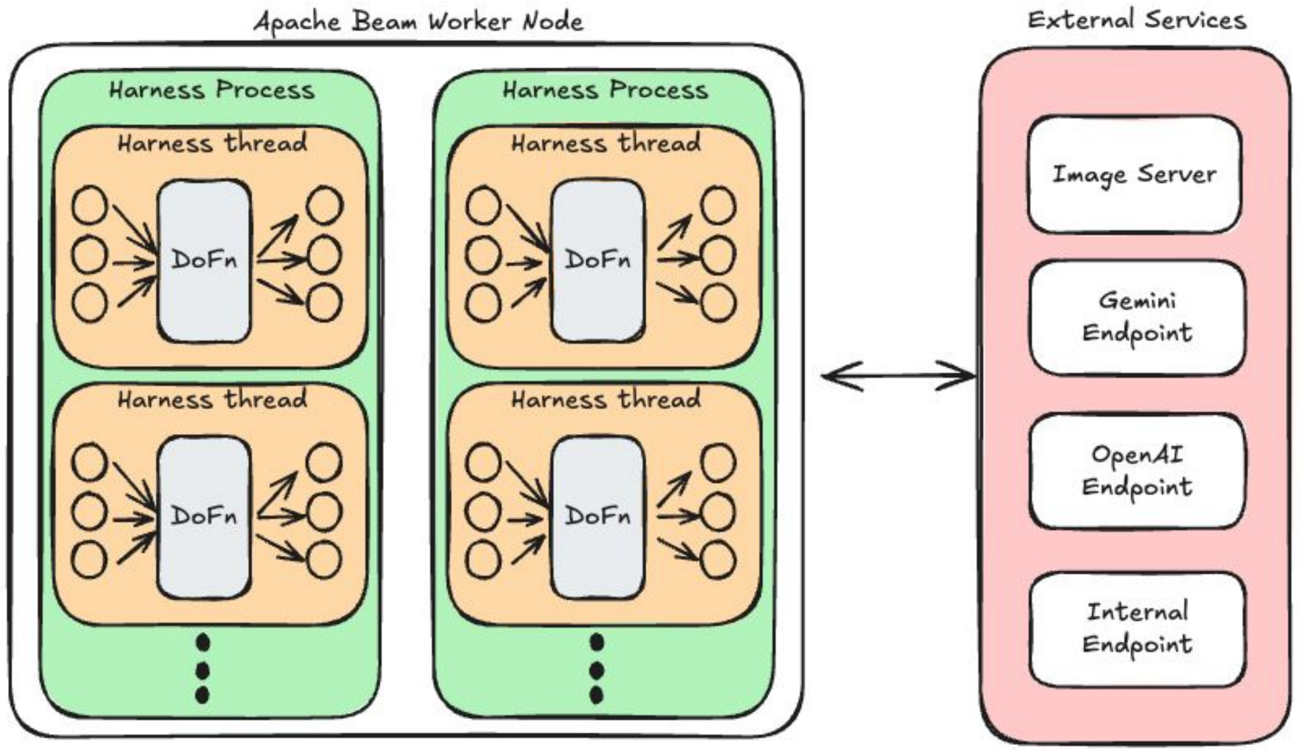
Diagnosing a Network-Bound Stage

Why low CPU doesn’t mean you can scale-out your way to speed

- Workers show low CPU usage, yet throughput remains flat
- Autoscaler sees backlog growth → adds more workers (“scale-out”)
- Result: More workers, small throughput, higher cost

Diagnosing a Network-Bound Stage

Why low CPU doesn’t mean you can scale-out your way to speed



Why It Happens

Root Causes

- **Per-element synchronous calls – Blocking HTTP calls stall threads**
- **Low concurrency within a worker – Limited SDK harness threads; blocking I/O kills parallelism**
- **Pipeline fusion & tiny bundles – Small bundles → low in-flight concurrency**
- External system quotas / single-connection limits – No pooling → QPS ceiling
- Retry/backoff stalls – Rate-limits + exponential backoff = long idle times
- Network plumbing constraints – Latency, port limits, DNS throttling, disabled keep-alive

Fixing a Network-Bound Stage in Beam/Dataflow

Practical changes that actually improve throughput

- **Make I/O concurrent & non-blocking**
  - Async client + connection pool + concurrency limits
  - Batch elements before API calls
- **Break fusion before I/O**
  - Use beam.Reshuffle() to get larger bundles into the I/O stage

Fixing a Network-Bound Stage in Beam/Dataflow

Practical changes that actually improve throughput

```
class AsyncHTTPDoFn(beam.DoFn):
  def setup(self):
    self.sem = asyncio.Semaphore(128)
    self.client = httpx.AsyncClient(http2=True)
  async def _call_one(self, item):
    async with self.sem:
      r = await self.client.post(URL, json=item)
      return r.json()
  async def _call_batch(self, batch):
    return await asyncio.gather(*(self._call_one(it) for it in batch))
  def process(self, batch):
    yield from asyncio.run(self._call_batch(batch))
```

```
input_collection
| beam.Reshuffle()
| beam.BatchElements(
        min_batch_size=32,
        max_batch_size=256
)
| beam.ParDo(AsyncHTTPDoFn())
```

Problem: GPU Memory Overload

How Beam worker processes & threads can exhaust GPU memory

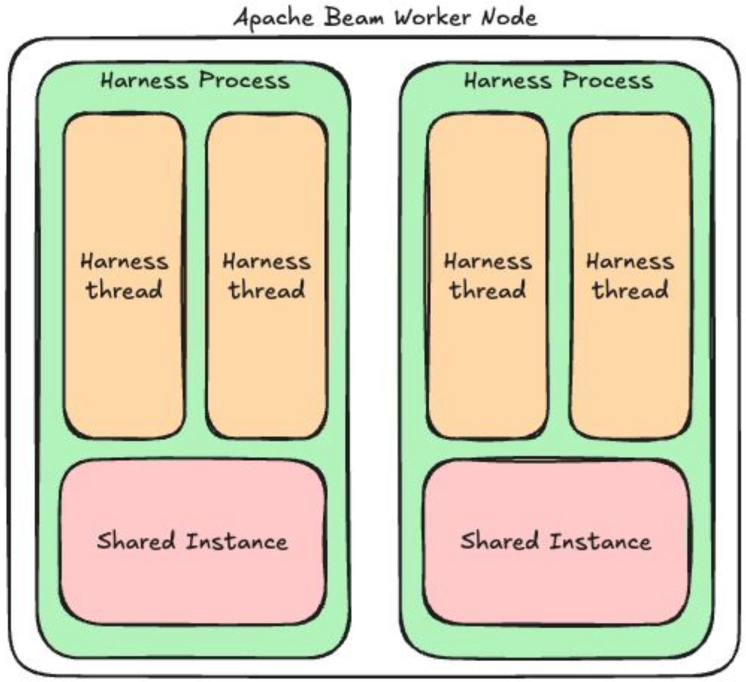
- Default behavior: **Beam spawns 1 process per CPU core**
- Each process dynamically creates worker threads
- Each **thread loads the model for its step**
- **GPU memory is limited** (~16 GB)
- Model load consumes at least **3GB per thread**

If every worker thread loads the model
→ **CUDA Out of Memory**

Introducing Shared in Apache Beam

Optimizing resource usage for model inference

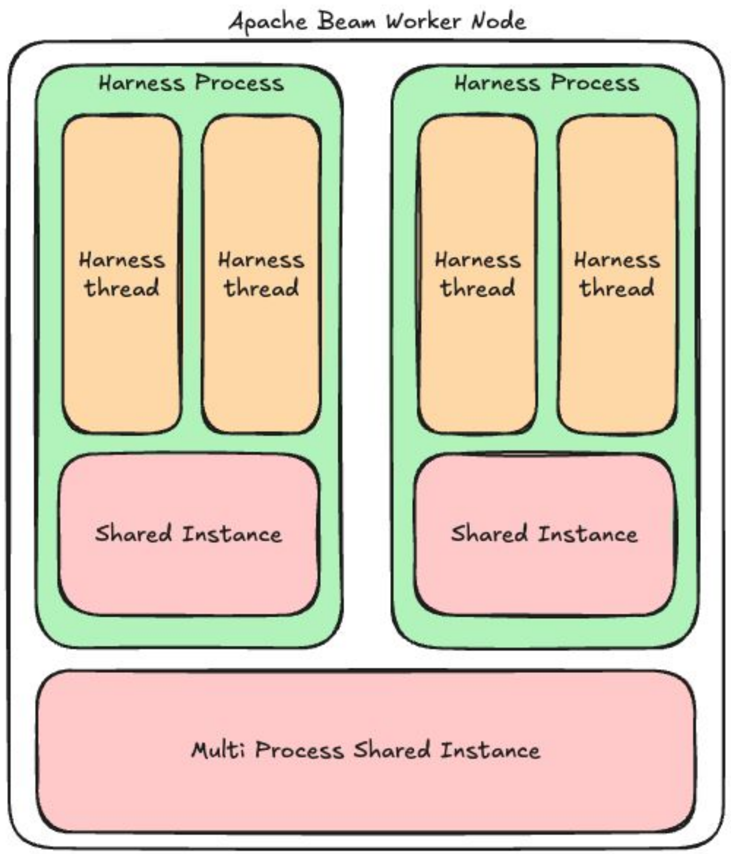
- Allows multiple threads within a single process to share an instance
- Reduces memory duplication for expensive objects in multi-threaded workers



Introducing MultiprocessShared

Optimizing resource usage for model inference

- Allows multiple processes on the same worker to share a single instance
- Greatly reduces memory footprint for large models
- Added in Beam Python 2.49.0



Problem: GPU Memory Overload

How Beam worker processes & threads can exhaust GPU memory

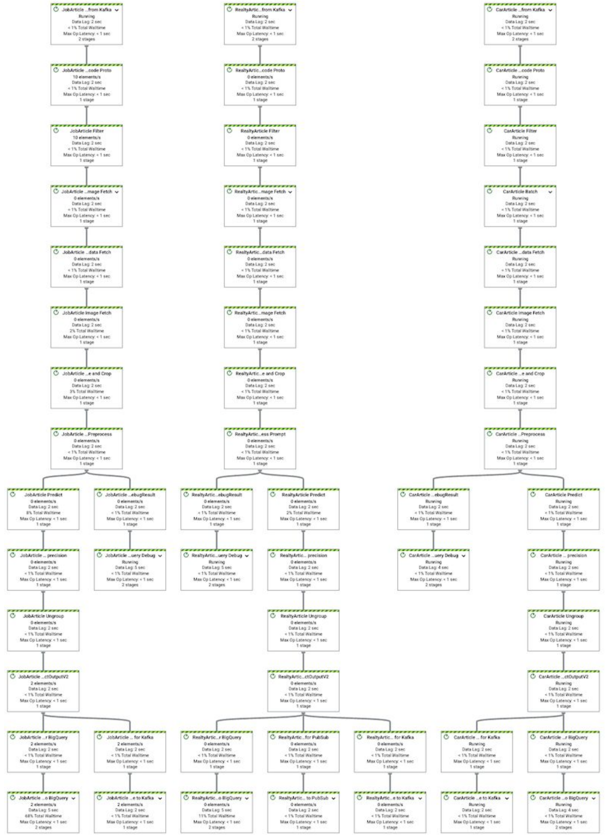
No more “CUDA out of memory”

Source : https://i.namu.wiki/i/45ad00iM-3ONvtUxfXmr5SF-RkJnlXjUsxg1fb2LSwnXxo3whiO1qAHovKfMqYWZxQBc-v9W5mq0WWfeMmwasQ.webp

Pipeline Consolidation for Cost Efficiency

Merging low-traffic pipelines to reduce baseline costs

- **Running all pipelines separately → High baseline cost**
- Identified low-traffic pipelines with underutilized resources
- **Consolidated these into shared pipelines**
- **Reduced idle resource** usage without impacting performance



What's Next?

Continuing the journey after this talk

- **Expand embedding-based pipelines to more products**
  - Deploy the current embedding-powered architecture beyond the initial use case, enabling search, recommendations, and personalization features across multiple services.
- **Improve customer experience with more models**
  - Integrate additional ML/LLM models to enhance relevance, accuracy, and responsiveness, focusing on multi-modal support (text, image, and video) for richer user interactions.

We are hiring🙌

당근

Thank you

박준성

Linkedin: linkedin.com/in/johan-park/
GitHub: github.com/Writtic

변규현

Linkedin: linkedin.com/in/novemberde/
Blog: novemberde.github.io