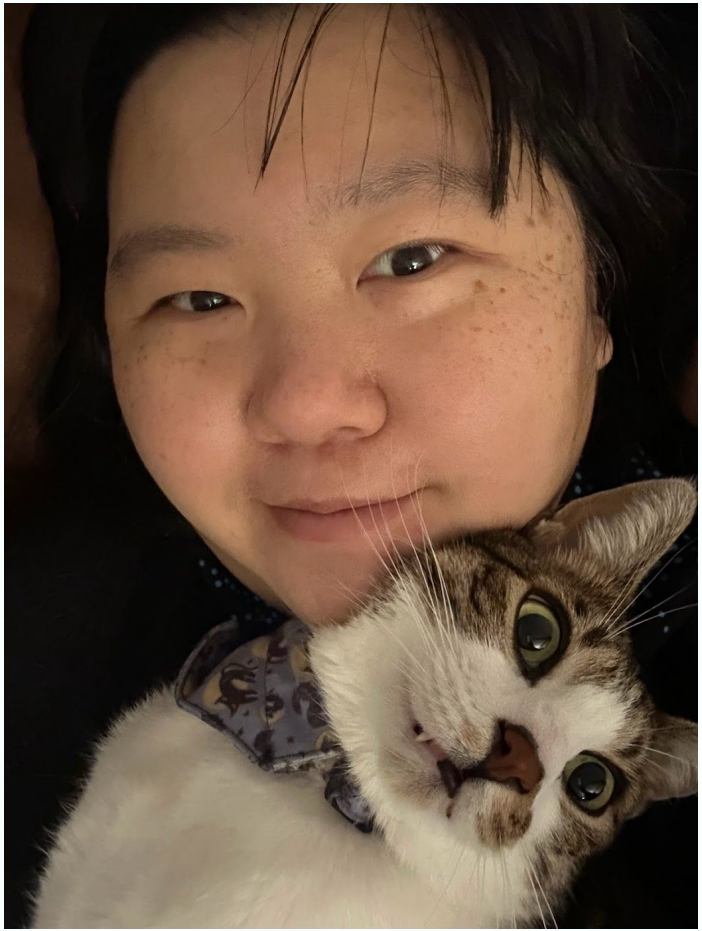

BEAM
SUMMIT

Running Beam Multi Language Pipeline on Flink Cluster on K8s

Lydian Lee
Affirm

Lydian Lee

- Worked in Affirm
- Streaming Infra Team
- Data, MLOps, ML
- Cat lover



affirm by the numbers

As of FQ3'2023

Active Consumers

26% Growth

88% of transactions from repeat users

FQ3'23 Transactions

36% Growth

34% Increased transactions per active consumer

Merchants

19% Growth

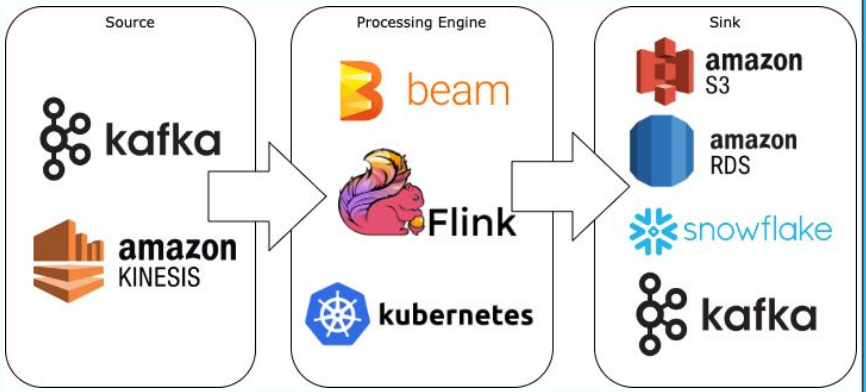
63% higher cart sizes for merchants using Affirm compared to other payment methods in CY 2022

Agenda

- Affirm Tech Stack Overview
- Beam: The Right Choice
- Challenges and Solutions
- Final Architecture

Affirm Tech Stack Overview

- Python and Kotlin are our primary languages.
  - Python is widely used throughout the company.
  - Kotlin engineering expertise is growing steadily.
- We employ a Lambda architecture.
  - Batch processing: Spark.
  - Streaming processing: Flink/Beam.
  - Our ML teams are exploring the Kappa architecture using Apache Beam.

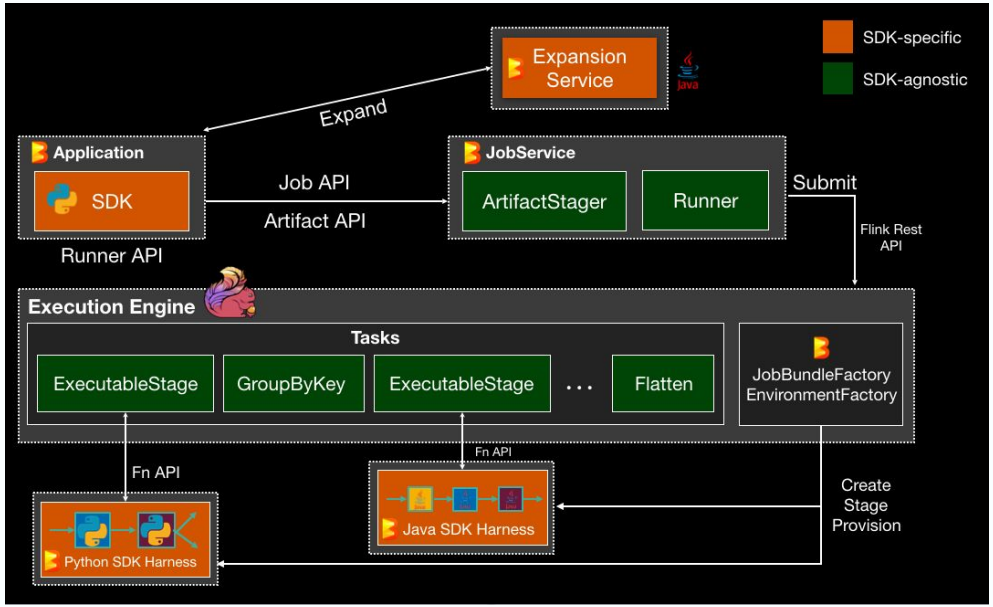


Beam: The Right Choice

- Apache Beam offers a user-friendly interface that our Python engineers find simple and intuitive.
- It seamlessly integrates with the Flink operator, enabling easy deployment on our Kubernetes cluster.
- The unified interface for Spark and Flink fulfills our requirements for real-time feature generation through stream processing, as well as batch processing for backfilling features from S3.

Beam Multi Language Support

- Our event processing function is written in Python, but KafkaIO is written in Java.
- Fortunately, Beam multi-language support bridges the gap between Java and Python, enabling seamless integration.



Challenges Encountered

1. Expansion service runs in Docker.
2. Flink task manager fails to locate artifacts.
3. Docker image size is too large.

Challenge 1: Expansion Service runs in Docker

- The Expansion Service runs as Docker by default
  - It is incompatible with Kubernetes
  - Workaround: DooD / DinD
    - security concerns

- The final solution involves passing the environment when launching the expansion service
  - This is not documented anywhere

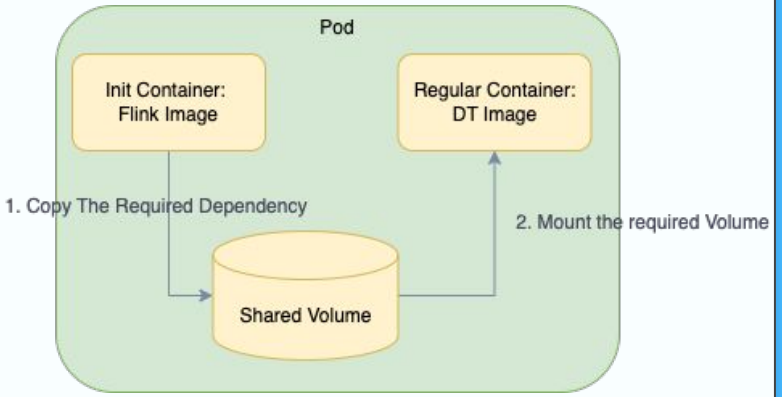
```
from apache_beam.io.kafka import default_io_expansion_service
from apache_beam.io.kafka import ReadFromKafka


ReadFromKafka(
      consumer_config,
      topics,
      with_metadata=False,
      expansion_service=default_io_expansion_service(
        append_args=[
            '--defaultEnvironmentType=PROCESS',

'--defaultEnvironmentConfig={"command":"/opt/apache/beam_java/boot"}',
        ]
  )
)
```

Challenge 2: Flink task manager fails to locate artifacts

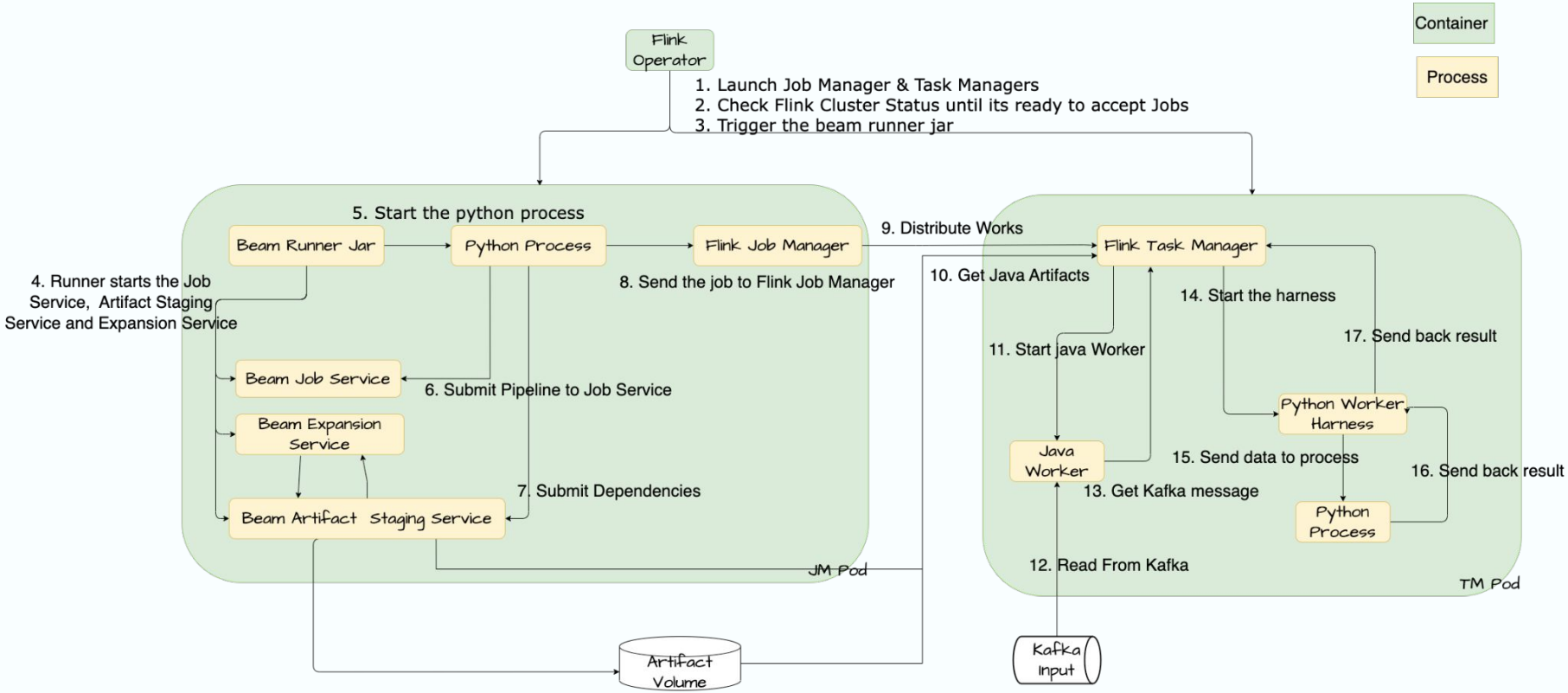
- Default artifacts storage is located in the Job Manager's local directory
- Expansion service does not respect arguments to store artifacts in S3
- Final Solution: Mount a shared volume on both the Job Manager and Task Manager for consistent artifact storage.

Challenge 3: Docker image size is too large

- Our company's tech dependencies contribute to a large Docker image size.
- Flink + Beam dependencies added approximately 500MB to the image layer, but they are only necessary for stream processing.
- Final Solution: Utilize a K8s webhook to dynamically add the required dependencies, reducing the Docker image size.



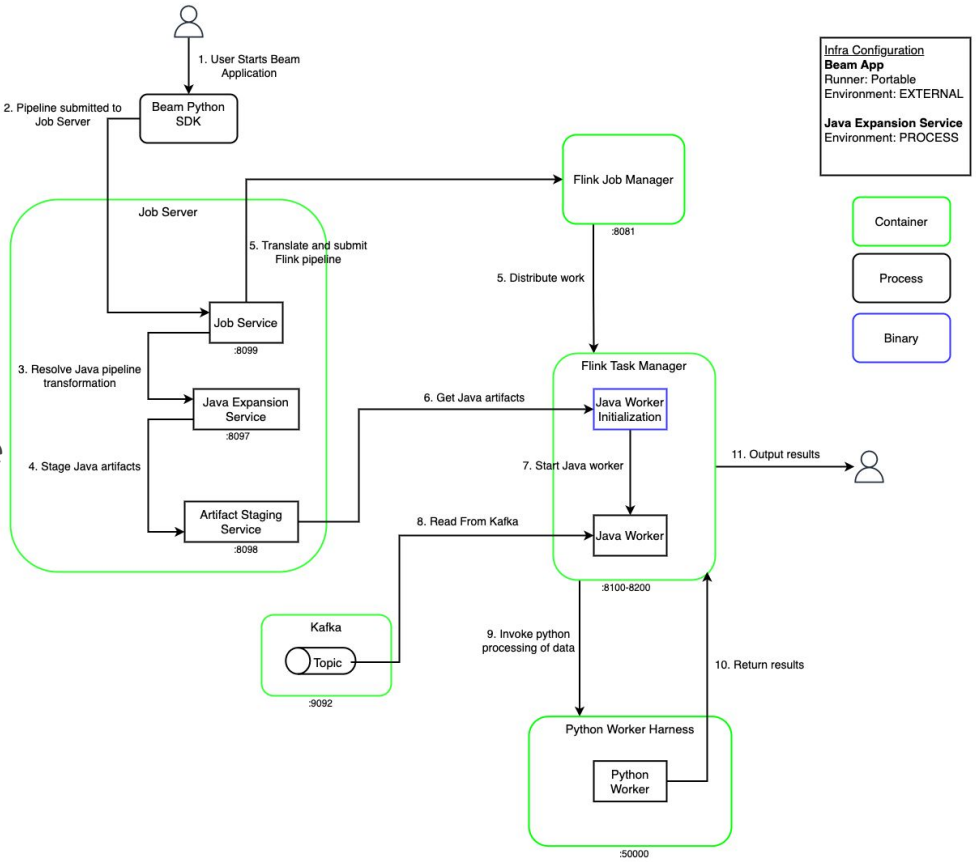
Final Architecture



Issue 4: Building a development Infra

- Direct Runner doesn't support for the unbounded data, therefore we cannot use it for local development and integration tests

- For rapid development, we utilize the docker-compose to create the Beam infra locally



Conclusion

- Configuring the Kubernetes framework for Beam infrastructure presents challenges due to the lack of documentation and missing functions.

- We aim to share our experience to help bridge the gap and simplify the setup process for Beam Infra on K8s.

We will be posting these challenges and solutions on Affirm Tech Blog soon!

QUESTIONS?

Affirm Tech Blog
Medium: @lydianlee
Github: lydian