A Low Code Structured Approach to Deploying Apache Beam ML Workloads on Kubernetes using Beamstack

Charles Adetiloye & Nate Salawe

BEAM SUMMIT

September 4-5, 2024

Sunnyvale, CA. USA

About the presenters

Charles Adetiloye is a Cofounder and Lead Machine Learning Platforms Engineer at MavenCode. He has well over 16 years of experience building large-scale distributed applications. He has extensive experience working and consulting with several companies implementing production grade ML platforms.

Nate is a Software Developer and Machine Learning Engineer at MavenCode. With a strong background in API development, Machine learning and AI, he specializes in implementing MLOps pipelines and managing model development and deployment. Nate holds a Bachelor's degree in Mathematics and has a keen interest in generative AI and cloud-based LLM solutions.

BEAM SUMMIT

About Mavencode

MavenCode is an Artificial Intelligence Solutions Company with HQ in Dallas, Texas and a remote delivery workforce across multiple time zones. We do training, product development and consulting services with specializations in:

- Provisioning Scalable AI and ML Infrastructure - OnPrem and In the Cloud
- Development & Production Operationalization of ML platforms - OnPrem and In the Cloud
- Streaming Data Analytics and Edge IoT Model Deployment for Federated Learning
- Building out Data lake, Feature Store, and ML Model Management platform

twitter.com/mavencode

Agenda

- Introduction to Beamstack
- Architectural Overview
- Key Features of Beamstack
- Beamstack Use Cases / Demos
- Future Roadmap

Introduction to Beamstack

What is Beamstack

- Beamstack is an open-source framework currently under development, aimed at facilitating the deployment of Machine Learning and GenAI workflow pipelines with Apache Beam on Kubernetes.

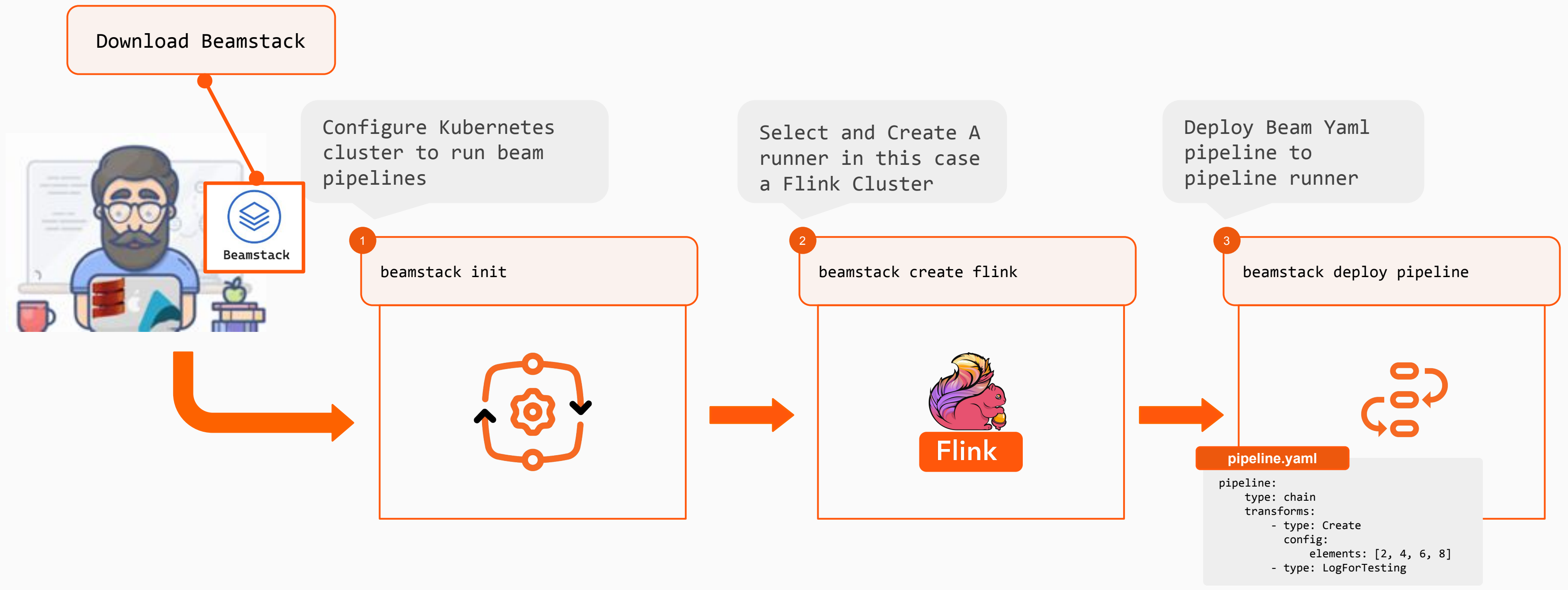
- Beamstack provides a robust Command Line Interface (CLI) that can potentially reduce pipeline deployment complexity and timelines drastically. It also possesses great monitoring and visualization features.

Getting Started with Apache Beam … Could be a lot

- What Runner should I use?
- What SDK should I use?
- Should I be running locally, on kubernetes or on GCP with dataflow?
- Is my code going to be "portable" if I switch runners?
- How do I optimize my code to run efficiently?

What if we could have a packaged tool with everything you need to get started???

Beamstack makes Beam Pipeline Job deployment as simple as …



Download Beamstack

Configure Kubernetes cluster to run beam pipelines

1 `beamstack init`

Select and Create A runner in this case a Flink Cluster

2 `beamstack create flink`

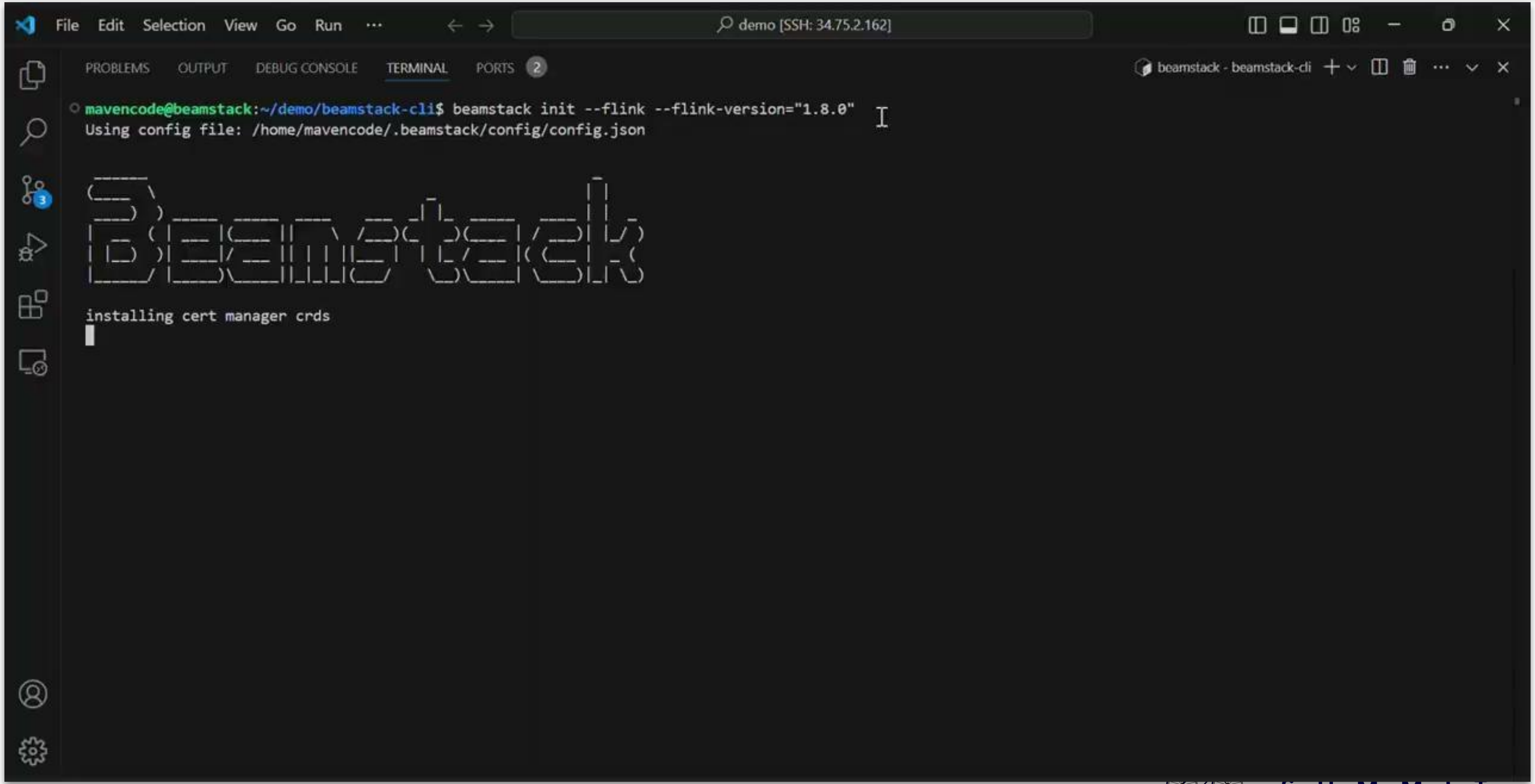
Deploy Beam Yaml pipeline to pipeline runner

3 `beamstack deploy pipeline`

pipeline.yaml

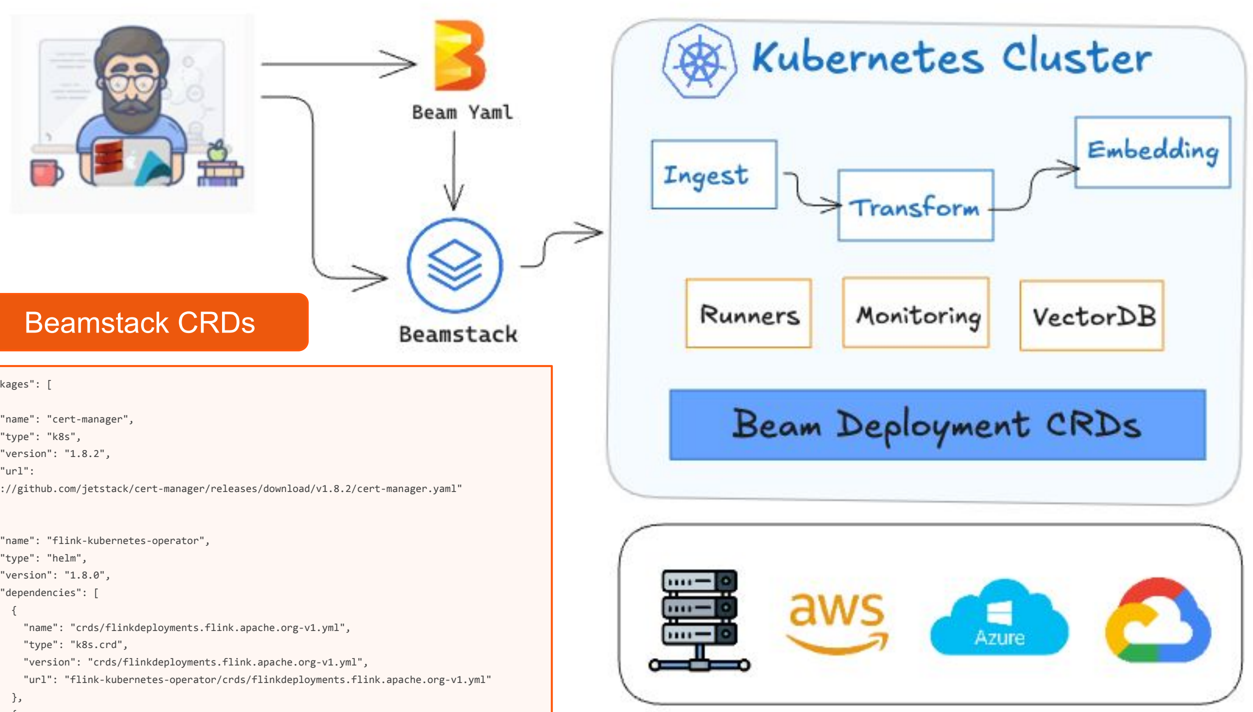
```yaml
pipeline:
    type: chain
    transforms:
        - type: Create
          config:
              elements: [2, 4, 6, 8]
        - type: LogForTesting
```

Beamstack makes Pipeline Job deployment as simple as …



```
mavencode@beamstack:~/demo/beamstack-cli$ beamstack init --flink --flink-version="1.8.0"
Using config file: /home/mavencode/.beamstack/config/config.json

 ______                            _
(____  \                      _   | |
 ____)  )_____ _____ ____   ___| |_ _____  ____| |  _
|  __  (| ___ (____ |    \ /___)  _|____ |/ ___) |_/ )
| |__)  ) ____/ ___ | | | |___ | |_/ ___ ( (___|  _ (
|______/|_____)_____|_|_|_(___/ \__)_____|\____)_| \_)

installing cert manager crds
```

Beamstack Initialization on the Kubernetes Cluster



Beamstack CRDs

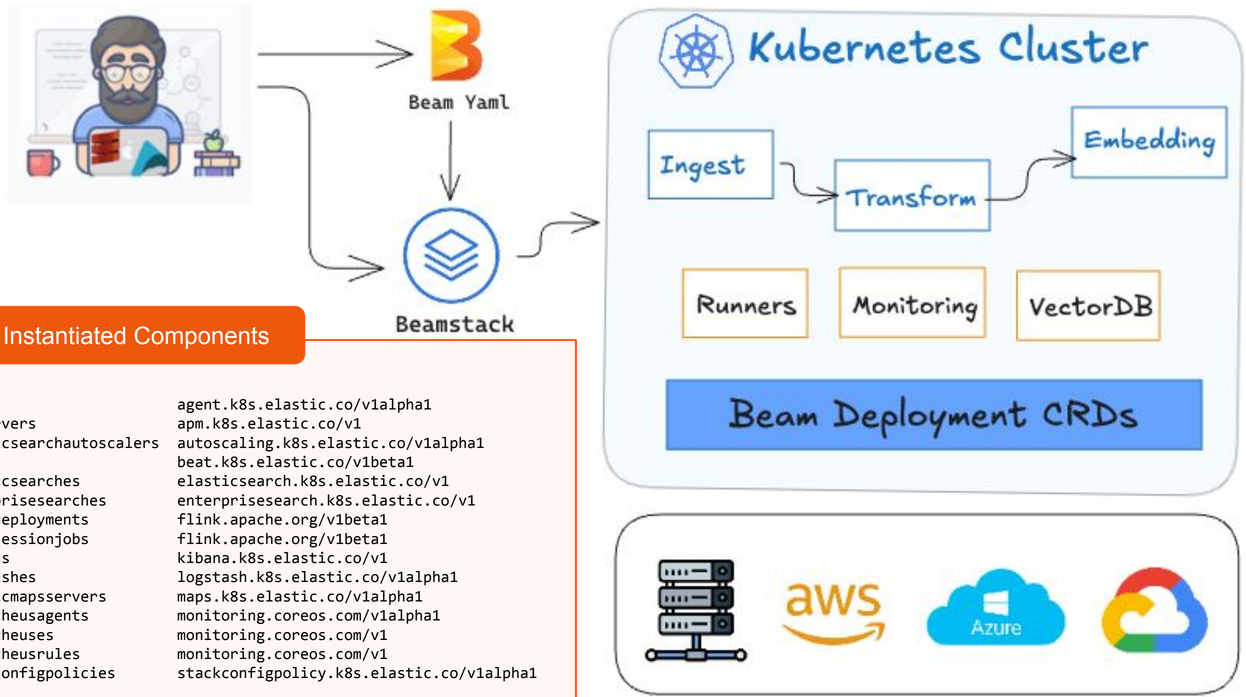
```
  "packages": [
    {
      "name": "cert-manager",
      "type": "k8s",
      "version": "1.8.2",
      "url":
"https://github.com/jetstack/cert-manager/releases/download/v1.8.2/cert-manager.yaml"
    },
    {
      "name": "flink-kubernetes-operator",
      "type": "helm",
      "version": "1.8.0",
      "dependencies": [
        {
          "name": "crds/flinkdeployments.flink.apache.org-v1.yml",
          "type": "k8s.crd",
          "version": "crds/flinkdeployments.flink.apache.org-v1.yml",
          "url": "flink-kubernetes-operator/crds/flinkdeployments.flink.apache.org-v1.yml"
        },
        {
          "name": "crds/flinksessionjobs.flink.apache.org-v1.yml",
          "type": "k8s.crd",
          "version": "crds/flinksessionjobs.flink.apache.org-v1.yml",
          "url": "flink-kubernetes-operator/crds/flinksessionjobs.flink.apache.org-v1.yml"
        }
      ],
      "url": "https://downloads.apache.org/flink/flink-kubernetes-operator-1.8.0/"
    },
```

3 Deploy the Beam YAML pipeline on the cluster

2 Instantiate the required core components typically the Runners, Monitoring Stack etc

1 Initialize Deployment CRDs for all the custom resources needed to run on Kubernetes

Beamstack will install the components you need …



Instantiated Components

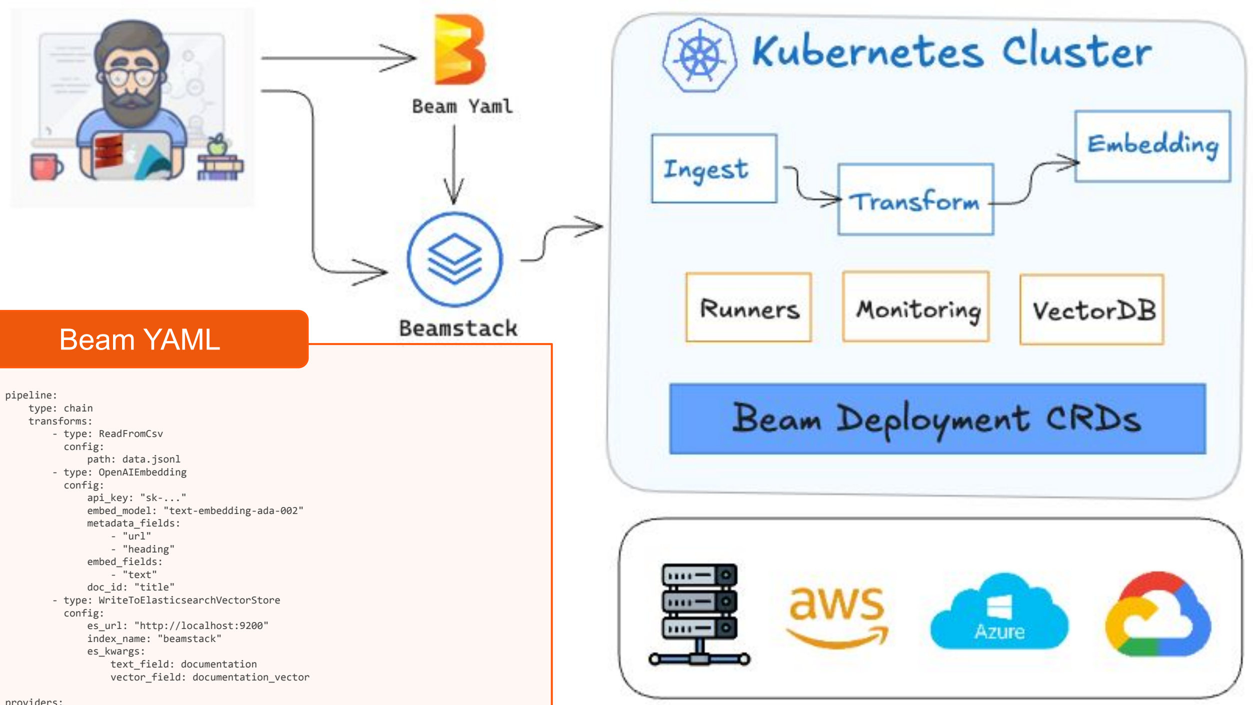
```
agents                      agent.k8s.elastic.co/v1alpha1
apmservers                  apm.k8s.elastic.co/v1
elasticsearchautoscalers    autoscaling.k8s.elastic.co/v1alpha1
beats                       beat.k8s.elastic.co/v1beta1
elasticsearches             elasticsearch.k8s.elastic.co/v1
enterprisesearches          enterprisesearch.k8s.elastic.co/v1
flinkdeployments            flink.apache.org/v1beta1
flinksessionjobs            flink.apache.org/v1beta1
kibanas                     kibana.k8s.elastic.co/v1
logstashes                  logstash.k8s.elastic.co/v1alpha1
elasticmapsservers          maps.k8s.elastic.co/v1alpha1
prometheusagents            monitoring.coreos.com/v1alpha1
prometheuses                monitoring.coreos.com/v1
prometheusrules             monitoring.coreos.com/v1
stackconfigpolicies         stackconfigpolicy.k8s.elastic.co/v1alpha1
```

3 Deploy the Beam YAML pipeline on the cluster

2 Instantiate the required core components typically the Runners, Monitoring Stack etc

1 Initialize Deployment CRDs for all the custom resources needed to run on Kubernetes

… And then you can deploy your Beam YAML jobs



Beam YAML

```yaml
pipeline:
    type: chain
    transforms:
        - type: ReadFromCsv
          config:
              path: data.jsonl
        - type: OpenAIEmbedding
          config:
              api_key: "sk-..."
              embed_model: "text-embedding-ada-002"
              metadata_fields:
                  - "url"
                  - "heading"
              embed_fields:
                  - "text"
              doc_id: "title"
        - type: WriteToElasticsearchVectorStore
          config:
              es_url: "http://localhost:9200"
              index_name: "beamstack"
              es_kwargs:
                  text_field: documentation
                  vector_field: documentation_vector

providers:
    - type: pythonPackage
      config:
          packages:
              -
https://raw.githubusercontent.com/BeamStackProj/transforms/main/package/beamstack_transfor
ms-0.1.0.tar.gz
      transforms:
          OpenAIEmbedding: "beamstack_transforms.embeddings.openai.CreateEmbeddings"
          WriteToElasticsearchVectorStore:
"beamstack_transforms.vectorstore.elasticsearch.WriteToElasticsearchVectorStore"
          ScrapeWebPages: "beamstack_transforms.io.webscrapper.ScrapeWebPages"
```

3 Deploy the Beam YAML pipeline on the cluster

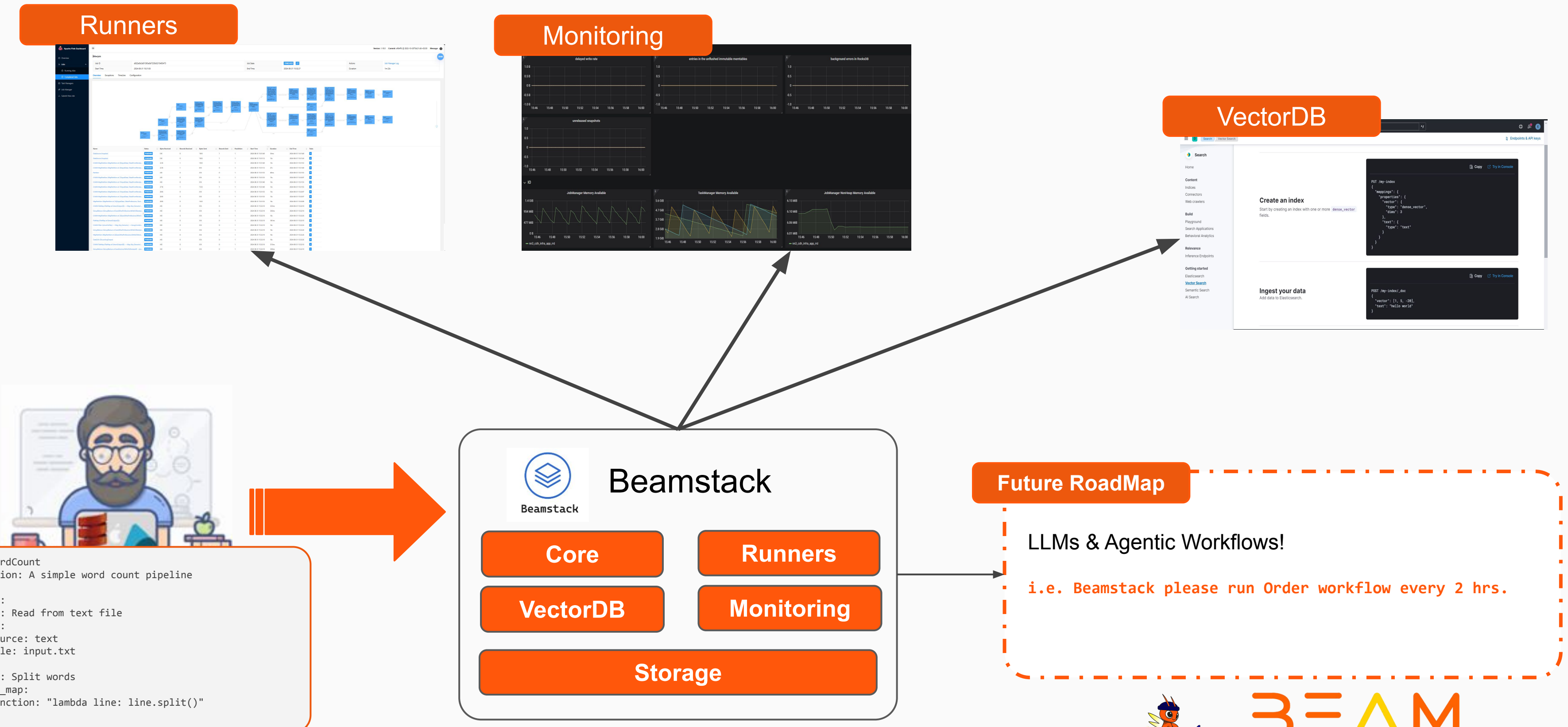
2 Instantiate the required core components typically the Runners, Monitoring Stack etc

1 Initialize Deployment CRDs for all the custom resources needed to run on Kubernetes

BEAM SUMMIT

Architectural Overview

Beamstack High Level Architecture



Create an index

Start by creating an index with one or more `dense_vector` fields.

```
PUT /my-index
{
  "mappings": {
    "properties": {
      "vector": {
        "type": "dense_vector",
        "dims": 3
      },
      "text": {
        "type": "text"
      }
    }
  }
}
```

Ingest your data

Add data to Elasticsearch.

```
POST /my-index/_doc
{
  "vector": [1, 5, -20],
  "text": "hello world"
}
```

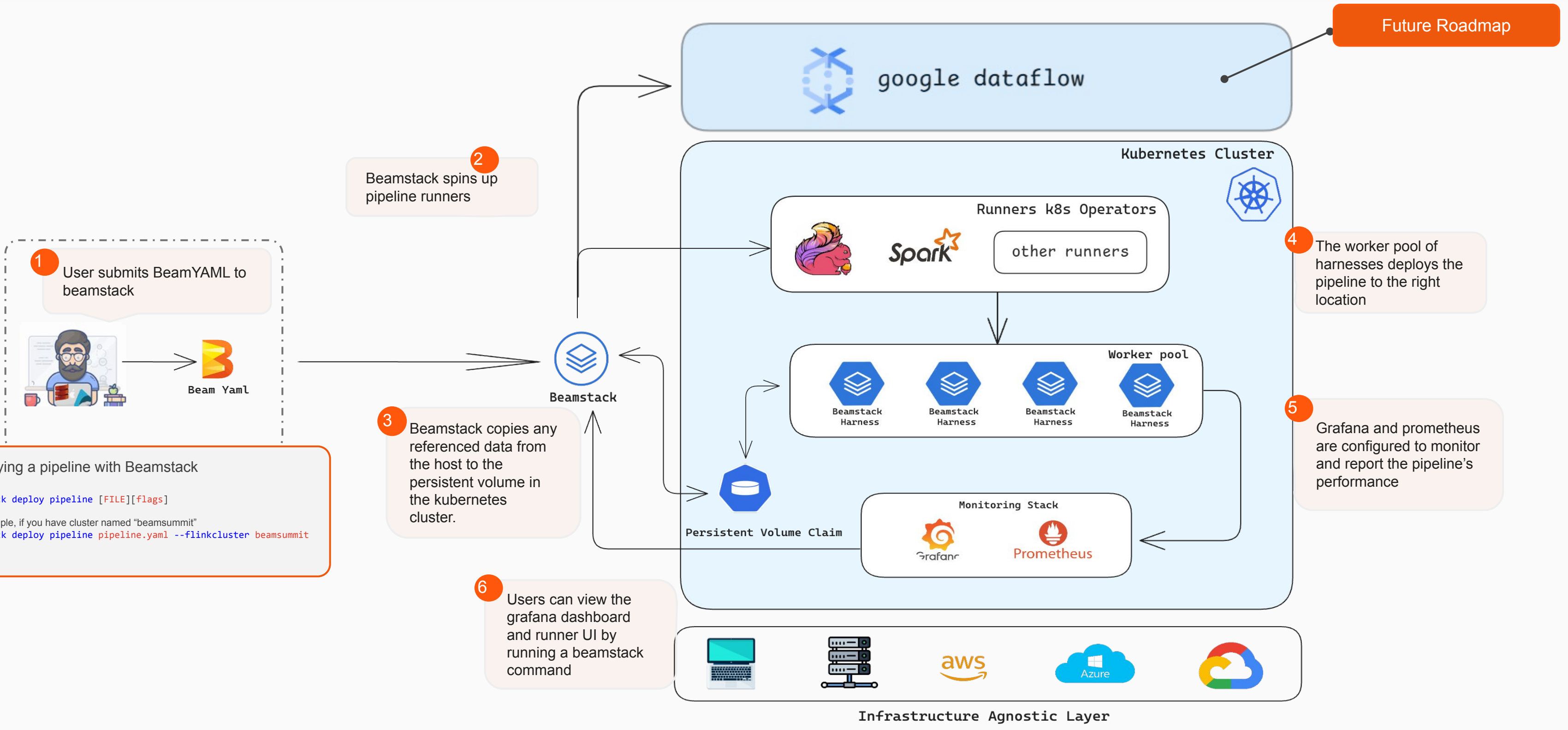
```
name: WordCount
description: A simple word count pipeline

pipeline:
  - name: Read from text file
    read:
      source: text
      file: input.txt

  - name: Split words
    flat_map:
      function: "lambda line: line.split()"
```

LLMs & Agentic Workflows!

`i.e. Beamstack please run Order workflow every 2 hrs.`

BEAM SUMMIT

How Developers Interact with Beamstack



1 User submits BeamYAML to beamstack

2 Beamstack spins up pipeline runners

3 Beamstack copies any referenced data from the host to the persistent volume in the kubernetes cluster.

4 The worker pool of harnesses deploys the pipeline to the right location

5 Grafana and prometheus are configured to monitor and report the pipeline's performance

6 Users can view the grafana dashboard and runner UI by running a beamstack command

Deploying a pipeline with Beamstack

```
beamstack deploy pipeline [FILE][flags]
```

For Example, if you have cluster named "beamsummit"

```
beamstack deploy pipeline pipeline.yaml --flinkcluster beamsummit
```

BEAM SUMMIT

Key Features of Beamstack

Key Features of Beamstack

Quick Cluster Configuration and Runner Setup

- Kubernetes Cluster Initialization
- Runner Installation and Configuration
- Resource allocation and preparation of the cluster for efficient utilization
- Deployment of additional resources i.e Grafana, Prometheus, ElasticSearch

Pipeline Runner Orchestration

- Runners lifecycle management
- Pipeline Artifacts migration
- Configure monitoring of runner metrics and logs
- Runner resource management

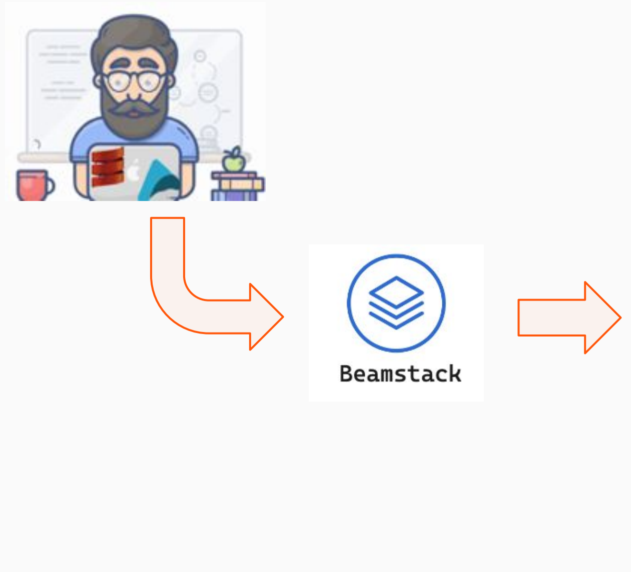
Custom Beam Transforms for AI workload

- Collection of PTranforms for AI
- integrates popular ai frameworks like openai and huggingface
- Easily extendable transforms for beam yaml pipelines

Monitoring and Observability

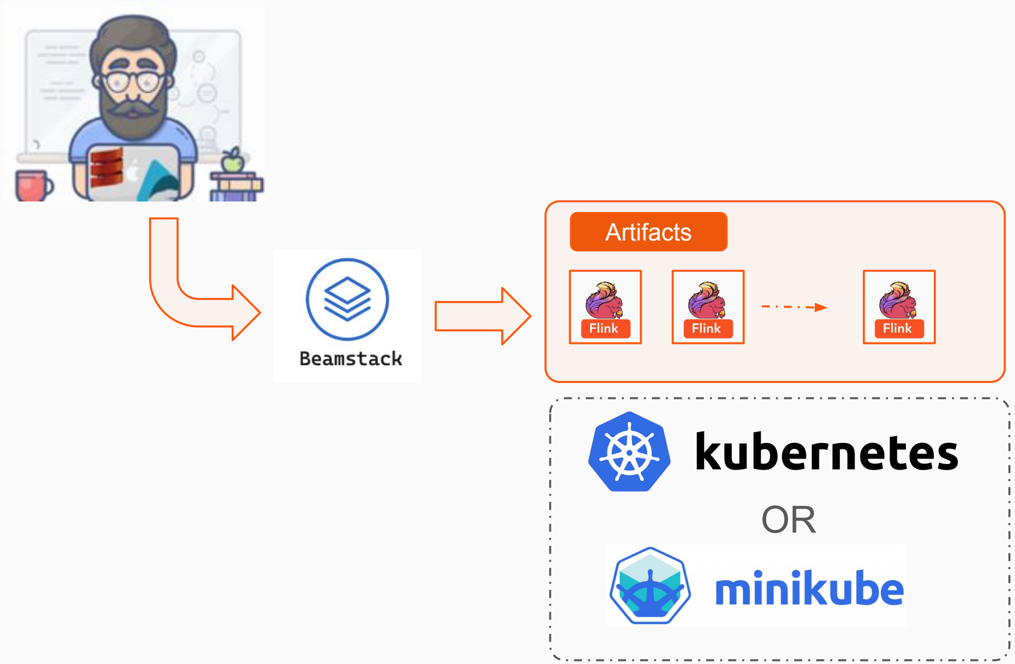
- Incorporates popular monitoring tools like prometheus and grafana
- Real time metrics collection from pipeline runners

Quick and Easy Cluster and Runner Setup


Beamstack
Beam Pipeline Deployment
Initialization of the Cluster Session
Other Core Components - Monitoring, VectorDB etc
Apache Beam Runner Configuration
Kubernetes CRD Config

- Configures kubernetes cluster optimized for ML workflows in less than 60 seconds.
- Automatically installs necessary workload components.
- Consistent and reproducible environment for deploying ML workloads.
- Seamless integration of pipeline components.

Runner Orchestration and Management

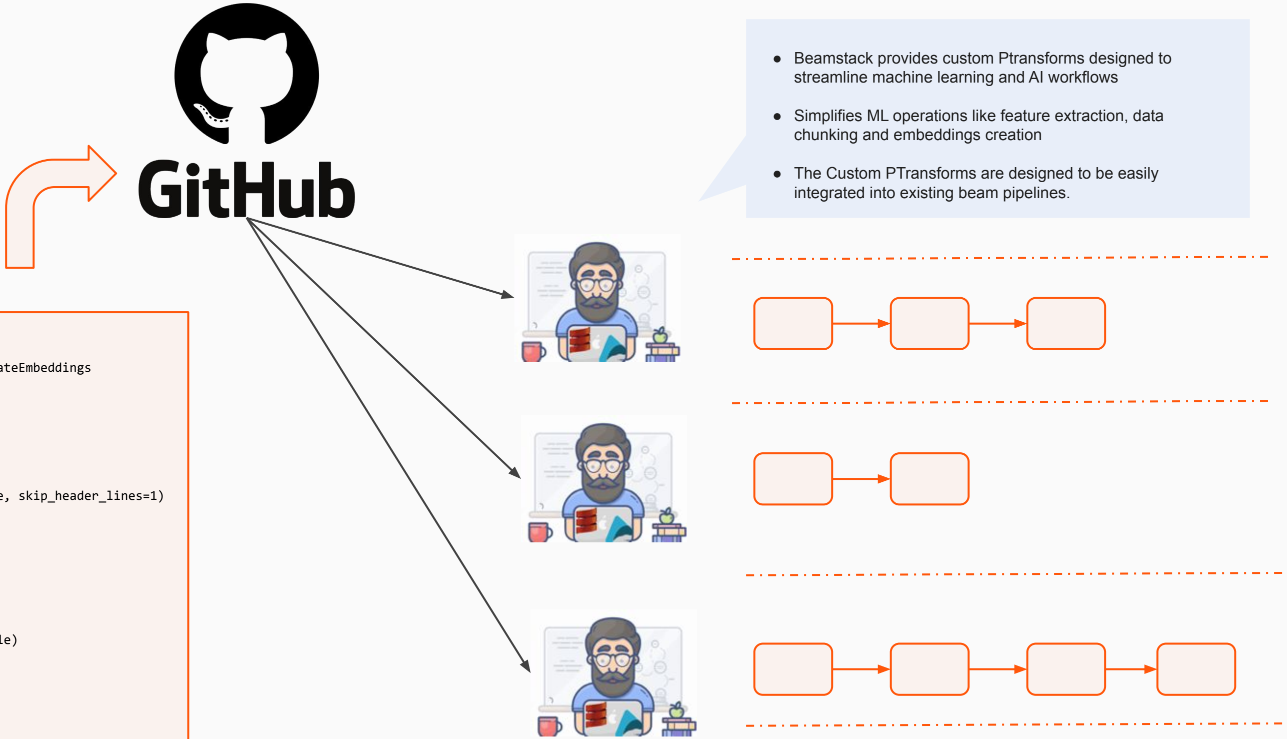


- Manages the creation, scaling, and termination of pipeline runners like Flink and Spark
- Seamless transfer of necessary data and artifacts to and from pipeline runners
- Configures monitoring of runner performance metrics and logs for tracking and diagnostics
- Optimizes resource allocation for pipeline runners

BEAM SUMMIT

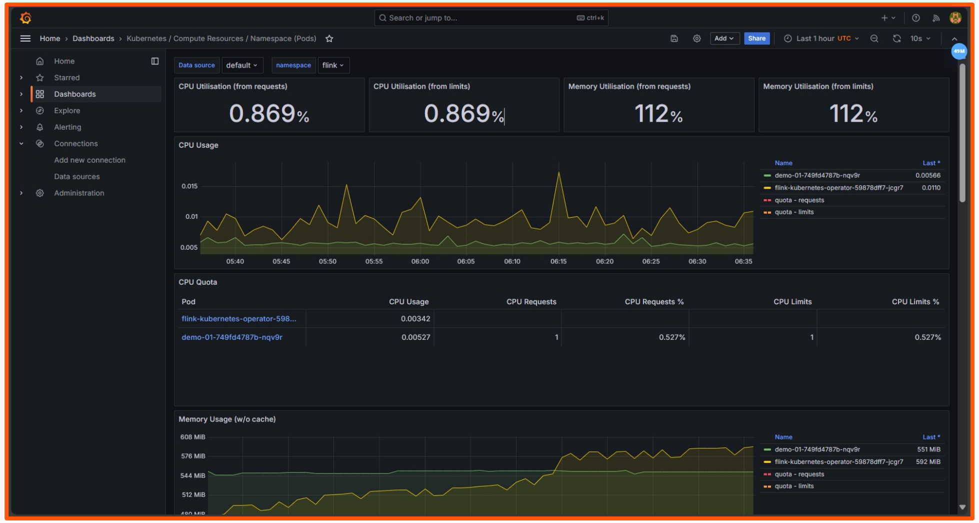
Custom Beam Transforms for AI and ML workloads

```
import apache_beam as beam

from beamstack_transforms.embeddings.huggingface import CreateEmbeddings



def run_pipeline(input_file: str, output_file: str):
    with beam.Pipeline() as p:
        text = (
            p
            | 'Read Text' >> beam.io.ReadFromText(input_file, skip_header_lines=1)
        )
        embeddings = (
            text
            | 'Convert to Embeddings' >>
CreateEmbeddings(model="thenlper/gte-large")
        )
        (
            embeddings
            | 'Write Text' >> beam.io.WriteToText(output_file)
        )


run_pipeline(“data.txt”, “output.txt”)
```



- Beamstack provides custom Ptransforms designed to streamline machine learning and AI workflows
- Simplifies ML operations like feature extraction, data chunking and embeddings creation
- The Custom PTransforms are designed to be easily integrated into existing beam pipelines.

Monitoring and Observability of Key Metrics



CPU Utilisation (from requests)	CPU Utilisation (from limits)	Memory Utilisation (from requests)	Memory Utilisation (from limits)
0.869%	0.869%	112%	112%

CPU Usage

Name	Last *
demo-01-749fd4787b-nqv9r	0.00566
flink-kubernetes-operator-59878dff7-jcgr7	0.0110
quota - requests	
quota - limits	

CPU Quota

Pod	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
flink-kubernetes-operator-598...	0.00342				
demo-01-749fd4787b-nqv9r	0.00527	1	0.527%	1	0.527%

Memory Usage (w/o cache)

Name	Last *
demo-01-749fd4787b-nqv9r	551 MiB
flink-kubernetes-operator-59878dff7-jcgr7	592 MiB
quota - requests	
quota - limits	

- Integrates with Grafana and Prometheus to capture key cluster metrics
- Collect Real-Time metrics from pipeline Runners
- Provides custom dashboards for supported runners.

*

Beamstack Use Cases

How we are currently using Beamstack

Data Preparation

Data ingestion and cleaning

- Data ingestion transforms
- Data cleaning transforms

Data Transformation

- Data transformation pipeline templates
- Feature engineering workflows

Data Vectorization

Text Embeddings and Custom Vectorization

- Text embedding pipelines
- Transforms for custom vectorization

Multi-artifact Embedding and Processing

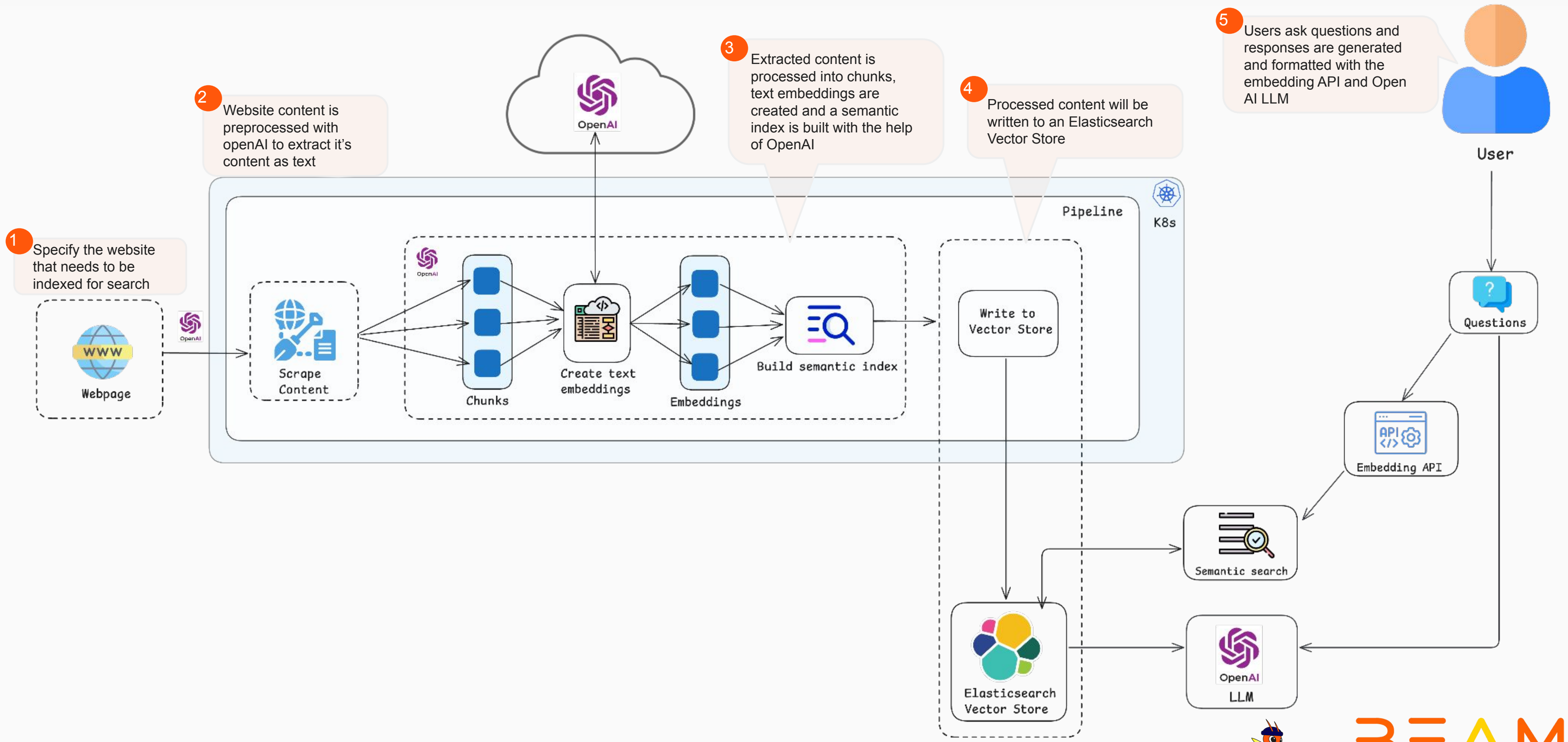
- Adaptive media processing transforms
- Enables cross-media embedding pipelines

Model Serving / Inferencing

Scalable Model Deployment and Real-time Inferencing

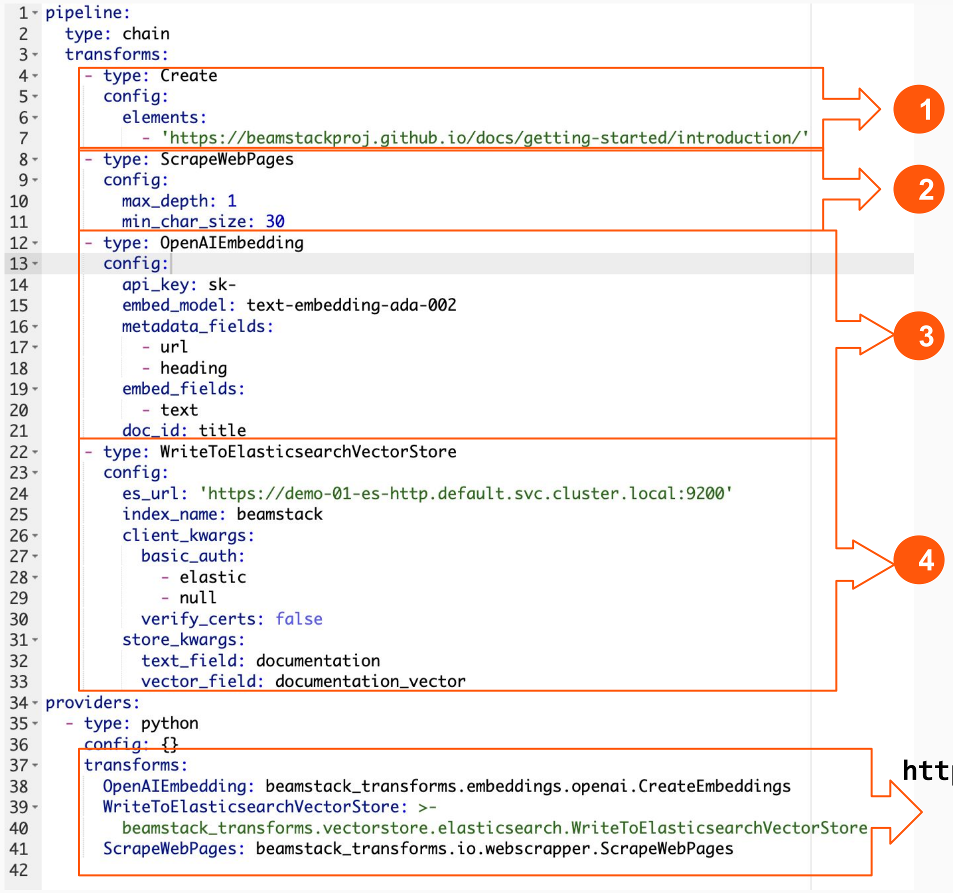
- Model deployment pipelines
- real-time/batch inference transforms

Example Use Case: Creating Text Embedding + Saving it to Vector Database

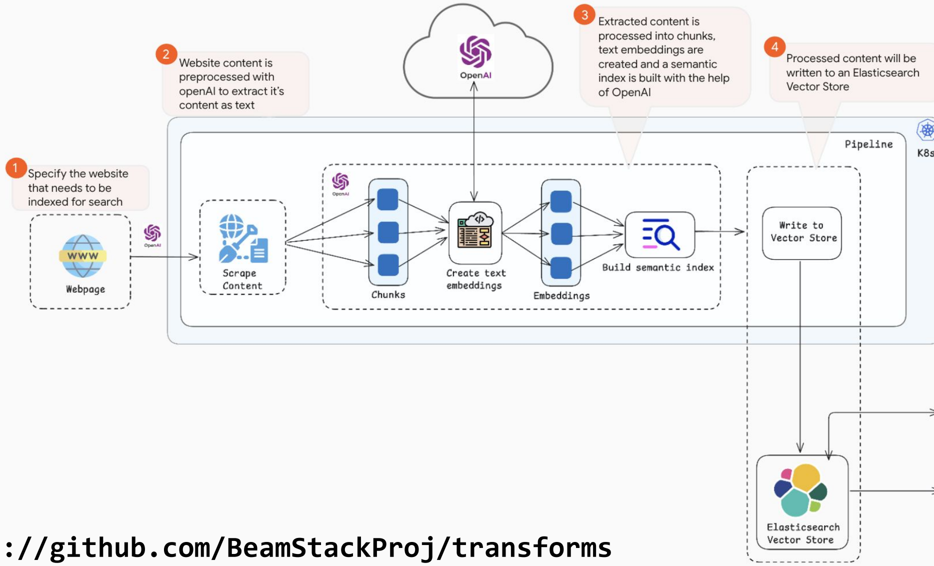

1 Specify the website that needs to be indexed for search
2 Website content is preprocessed with openAI to extract it's content as text
3 Extracted content is processed into chunks, text embeddings are created and a semantic index is built with the help of OpenAI
4 Processed content will be written to an Elasticsearch Vector Store
5 Users ask questions and responses are generated and formatted with the embedding API and Open AI LLM
OpenAI
Webpage
Scrape Content
Chunks
Create text embeddings
Embeddings
Build semantic index
Pipeline
K8s
Write to Vector Store
Elasticsearch Vector Store
User
Questions
Embedding API
Semantic search
LLM

BEAM SUMMIT

Example Use Case: Creating Text Embedding + Saving it to Vector Database



```yaml
pipeline:
  type: chain
  transforms:
    - type: Create
      config:
        elements:
          - 'https://beamstackproj.github.io/docs/getting-started/introduction/'
    - type: ScrapeWebPages
      config:
        max_depth: 1
        min_char_size: 30
    - type: OpenAIEmbedding
      config:
        api_key: sk-
        embed_model: text-embedding-ada-002
        metadata_fields:
          - url
          - heading
        embed_fields:
          - text
        doc_id: title
    - type: WriteToElasticsearchVectorStore
      config:
        es_url: 'https://demo-01-es-http.default.svc.cluster.local:9200'
        index_name: beamstack
        client_kwargs:
          basic_auth:
            - elastic
            - null
          verify_certs: false
        store_kwargs:
          text_field: documentation
          vector_field: documentation_vector
providers:
  - type: python
    config: {}
    transforms:
      OpenAIEmbedding: beamstack_transforms.embeddings.openai.CreateEmbeddings
      WriteToElasticsearchVectorStore: >-
        beamstack_transforms.vectorstore.elasticsearch.WriteToElasticsearchVectorStore
      ScrapeWebPages: beamstack_transforms.io.webscrapper.ScrapeWebPages
```



1 Specify the website that needs to be indexed for search

2 Website content is preprocessed with openAI to extract it's content as text

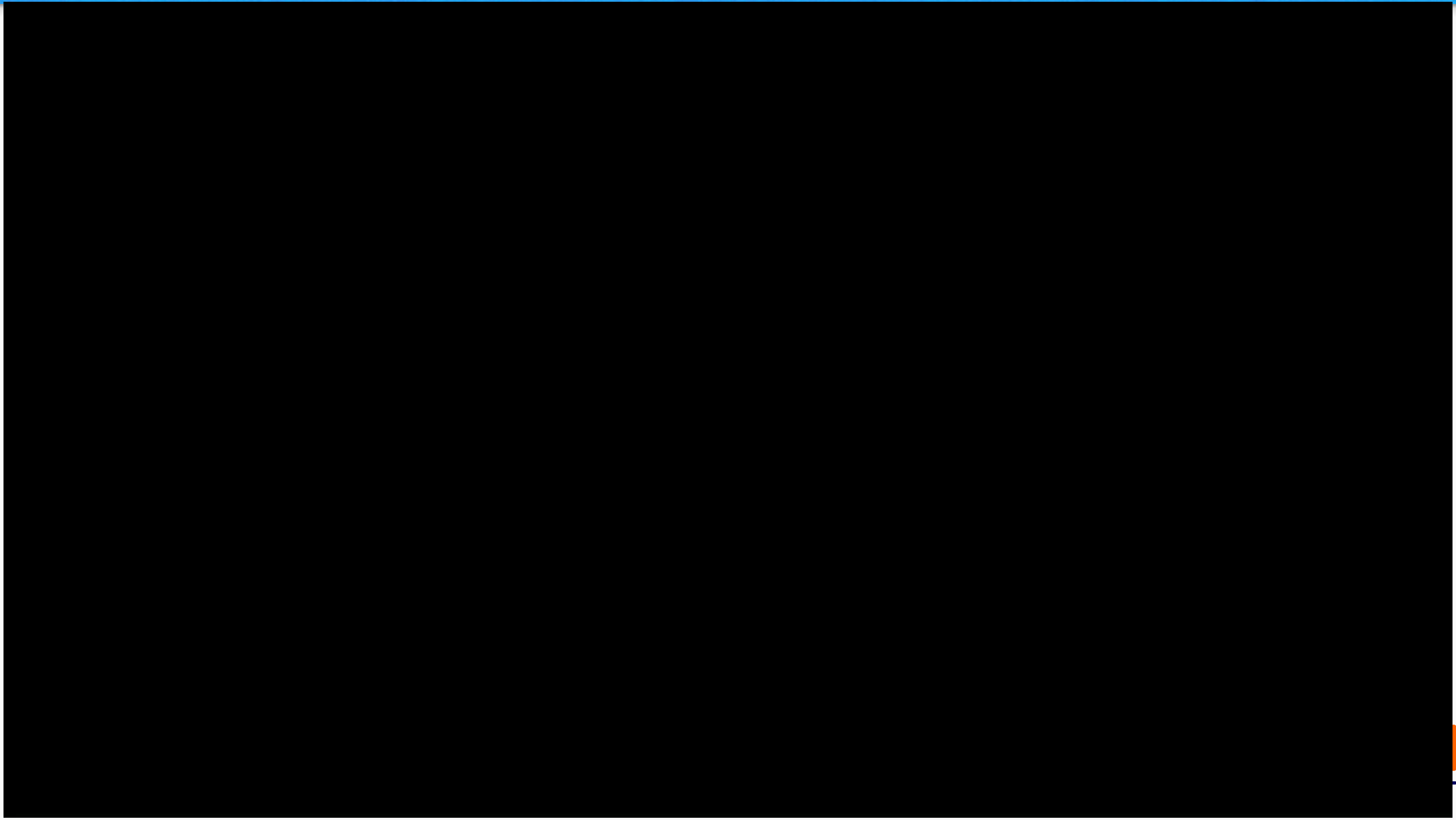
3 Extracted content is processed into chunks, text embeddings are created and a semantic index is built with the help of OpenAI

4 Processed content will be written to an Elasticsearch Vector Store

https://github.com/BeamStackProj/transforms

BEAM SUMMIT

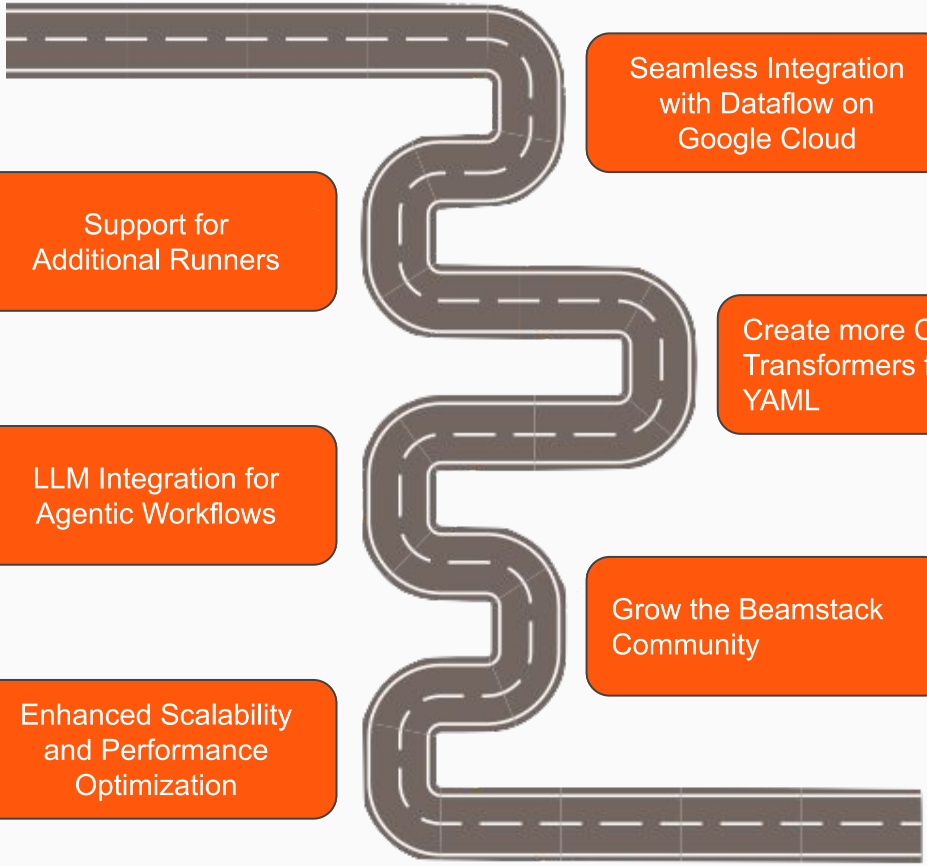
Nate’s Demo 🙂



Future Road Maps

Future Roadmap for Beamstack

Beamstack



Seamless Integration with Dataflow on Google Cloud

Support for Additional Runners

Create more Custom Transformers for Beam YAML

LLM Integration for Agentic Workflows

Grow the Beamstack Community

Enhanced Scalability and Performance Optimization

Thank you and please connect with us

https://github.com/beamstackproj

https://bit.ly/beamstack

https://beamstackproj.github.io/