

Build an ML Pipeline (Part 1) — Getting Started with Kubeflow V2 Pipelines

Discovering Kubeflow: Launching into ML Pipelines



Vinayak Shanawad · Following

Published in Analytics Vidhya · 6 min read · Dec 4, 2023



23



4



Goal

To build an end-to-end machine learning workflow, we will harness the

Medium

Search

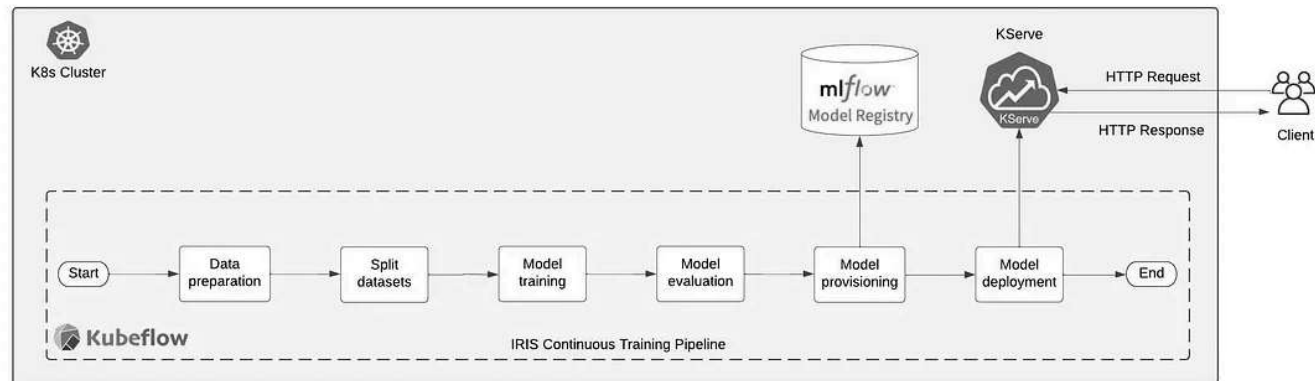
Write



source technologies — Kubeflow Pipelines, MLflow, and KServe.

By bringing together these components on minikube, we can go from data to deployment with an integrated pipeline — training effective models with Kubeflow, registering them with MLflow for versioning and lineage, and then serving predictions robustly with KServe. This unified environment

gives us an extensible platform to develop, deploy, and monitor our models with the tools purpose-built for machine learning and Kubernetes.



End-to-end ML Pipeline using Kubeflow, MLflow, and KServe (Image by Author)

Let's focus on setting up the minikube cluster, installing Kubeflow pipelines, and creating the Kubeflow pipeline using the popular data science IRIS dataset in this post.

We will look at the MLflow setup, register the model into the MLflow model registry, and serve the model using KServe in my [next post](#).

🤔 What is Minikube?

Minikube is a tool that makes it easy to run Kubernetes locally. Here are some key things to know about minikube:

- It is free, open source, and available for Linux, macOS and Windows.
- Minikube runs a single-node Kubernetes cluster (Master and Node processes run on ONE machine) inside a VM on your local machine.
- It will have docker pre-installed and set up kubectl to communicate with it.
- This provides a simple way to test and develop against a real Kubernetes cluster without needing the full production infrastructure.
- Once developed, the same apps can be deployed into cloud Kubernetes clusters.

Minikube Setup

We can reference the [minikube documentation](#) to install the minikube cluster on our Windows machine.

Well, I have the Chocolatey package manager installed if not please install the Chocolatey and use the following command to install the minikube cluster.

```
choco install minikube
```

Let's start our minikube cluster, we notice from logs that by default it uses the docker as a driver to start the cluster means that minikube will configure the Kubernetes cluster to use docker as its container runtime.

```
minikube start
```

A screenshot of a terminal window showing the output of the 'minikube start' command. The logs are displayed on a dark background with light-colored text. The output shows the minikube version (v1.31.1) and the operating system (Microsoft Windows 11). It details the process of selecting the Docker driver, pulling the base image, creating a Docker container, and preparing the Kubernetes cluster (v1.27.3) on Docker (24.0.4). The logs also mention generating certificates, booting the control plane, and configuring the bridge CNI. A warning is shown about the kubectrl.exe version (1.25.9) being incompatible with Kubernetes 1.27.3, and a suggestion to use 'minikube kubectrl -- get pods -A' is provided. The final status is 'Done! kubectrl is now configured to use "minikube" cluster and "default" namespace by default'.

Minikube start logs (Image by Author)

How to manage minikube cluster?

Check the minikube cluster status — `minikube status`

```
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Minikube cluster status (Image by Author)

Stop the cluster — `minikube stop`

Delete all of the minikube clusters — `minikube delete --all`

List all the pods within the minikube cluster — `minikube get pod -A`

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-5d78c9869d-8s5h2	1/1	Running	0	28m
kube-system	etcd-minikube	1/1	Running	0	28m
kube-system	kube-apiserver-minikube	1/1	Running	0	28m
kube-system	kube-controller-manager-minikube	1/1	Running	0	28m
kube-system	kube-proxy-b6mkc	1/1	Running	0	28m
kube-system	kube-scheduler-minikube	1/1	Running	0	28m
kube-system	storage-provisioner	1/1	Running	1 (27m ago)	28m

All pods within the Minikube cluster (Image by Author)

Awesome, we have a minikube cluster setup ready in our local Windows machine. Let's setup a Kubeflow pipelines on minikube cluster.

 **What is Kubeflow Pipeline?**

Kubeflow Pipelines provides a platform for building and deploying portable and scalable machine-learning workflows on Kubernetes. Some key capabilities:

- An SDK for declaring pipelines as code — the workflows are defined as Python functions decorated with Kubeflow’s domain-specific language (DSL).
- Run compute-intensive jobs as Kubernetes pods and jobs, leveraging Kubernetes for distribution and scaling.
- End-to-end ML workflows — steps for data ingestion, feature engineering, model training, evaluation, etc can be orchestrated.
- Notebooks to interactively develop and test pipelines.
- CLI and UI to monitor pipeline runs on Kubernetes.
- Portable — pipelines authored on a laptop can run on multi-node Kubernetes clusters.

Kubeflow Setup

Now that we have the prerequisite K8s cluster installed, it’s time for the main show.

The easiest way to install Kubeflow locally is to use the manifest files from the Git repo.

```
kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/cluster-scop  
kubectl wait --for condition=established --timeout=60s crd/applications.app.k8s.  
kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/env/dev?ref=
```

The Kubeflow Pipelines deployment may take several minutes to complete.

Let's check if kubeflow installed on minikube cluster.

```
kubectl get pods -A
```

This will list all pods across all namespaces. As long as the status is Running for all of them, you're good to go. If any of them are not, sometimes you just need to wait a little longer. Your list should look something like this:

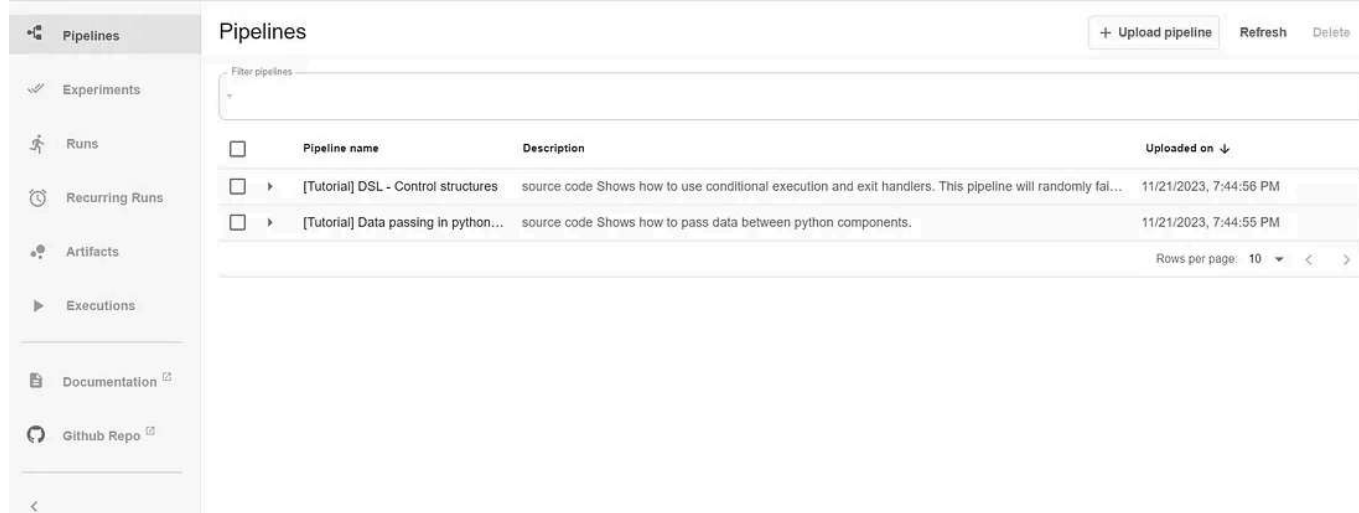
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-5d78c9869d-wswxv	1/1	Running	2 (2m56s ago)	21d
kube-system	etcd-minikube	1/1	Running	2 (2m56s ago)	21d
kube-system	kube-apiserver-minikube	1/1	Running	2 (2m56s ago)	21d
kube-system	kube-controller-manager-minikube	1/1	Running	2 (2m56s ago)	21d
kube-system	kube-proxy-648w8	1/1	Running	2 (2m56s ago)	21d
kube-system	kube-scheduler-minikube	1/1	Running	2 (2m56s ago)	21d
kube-system	storage-provisioner	1/1	Running	7 (2m3s ago)	21d
kubeflow	cache-deployer-deployment-64dc947fc7-97gfx	1/1	Running	1 (2m56s ago)	9d
kubeflow	cache-server-7f7d6bfb55-m9cxj	1/1	Running	1 (2m56s ago)	9d
kubeflow	controller-manager-dfbd6b98-xd7kk	1/1	Running	2 (2m ago)	9d
kubeflow	metadata-envoy-deployment-6dcd4ddcb8-zbxgt	1/1	Running	1 (2m56s ago)	9d
kubeflow	metadata-grpc-deployment-5644fb9768-7t7nr	1/1	Running	9 (97s ago)	9d
kubeflow	metadata-writer-9c4488669-drcw7	1/1	Running	6 (2m56s ago)	9d
kubeflow	minio-55464b6ddb-9ksjn	1/1	Running	1 (2m56s ago)	9d
kubeflow	ml-pipeline-65cc95bd6b-6lrcz	1/1	Running	2 (95s ago)	9d
kubeflow	ml-pipeline-persistenceagent-545d5c6786-mxf18	1/1	Running	4 (2m56s ago)	9d
kubeflow	ml-pipeline-scheduledworkflow-8f9b7654d-jhw8l	1/1	Running	1 (2m56s ago)	9d
kubeflow	ml-pipeline-ui-7c4cf85598-gnmf9	1/1	Running	1 (2m56s ago)	9d
kubeflow	ml-pipeline-viewer-crd-589c6c6569-4s7wt	1/1	Running	1 (2m56s ago)	9d
kubeflow	ml-pipeline-visualizationserver-9fcfbd447-f69qx	1/1	Running	1 (2m56s ago)	9d
kubeflow	mysql-7d8b8ff4f4-xkjdk	1/1	Running	1 (2m56s ago)	9d
kubeflow	proxy-agent-8dc6b5fd8-k2sp8	1/1	Running	42 (55s ago)	9d
kubeflow	workflow-controller-589ff7c479-xrrhn	1/1	Running	1 (2m56s ago)	9d

Kubeflow installation status (Image by Author)

Let's verify that the Kubeflow Pipelines UI or dashboard is accessible by port-forwarding:

```
kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
```

Then, open the Kubeflow Pipelines UI at <http://localhost:8080/>



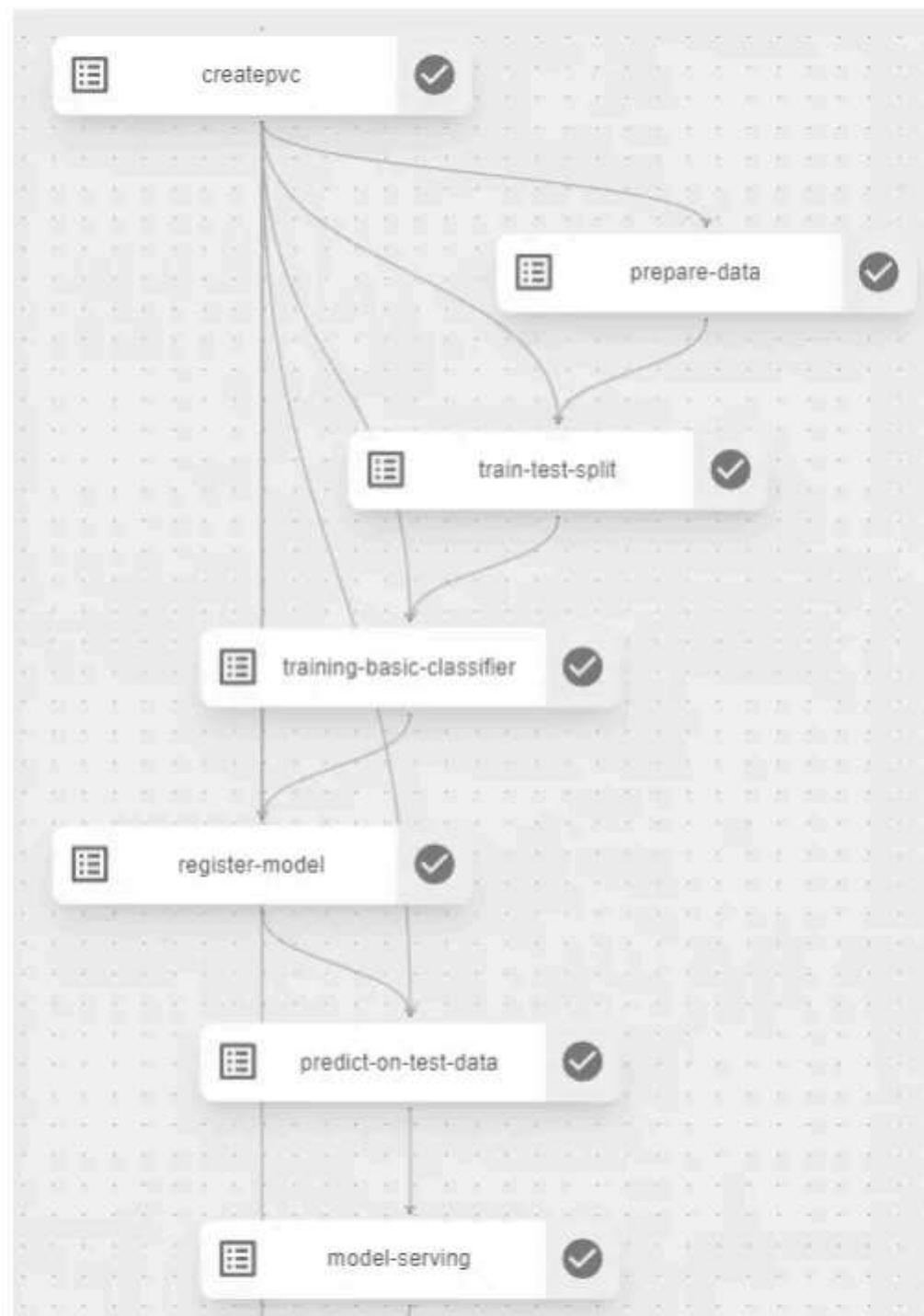
Kubeflow Pipeline UI (Image by Author)

Kubeflow Pipelines

Kubeflow pipelines consist of components, each representing a step in the pipeline. Each component runs in its Docker container, allowing each step to have its set of dependencies without affecting other components.

For every component we build, we'll either create a new Docker image or use an existing one. These images take inputs, execute specific operations, and produce outputs. Additionally, we'll maintain a distinct Python script, `pipeline.py`, which converts each Docker image into a pipeline component and assembles the components into a complete pipeline.

We will create data-preparation, train-test-split, model training, model evaluation, model provisioning/register model, and model serving steps using Kubeflow.



Let's go through each step in detail. Please note that we are going to build the Kubeflow pipeline using Kubeflow SDK v2.

1. Import the required libraries

```
from kfp.dsl import component, pipeline
import kfp
from kfp import kubernetes
```

We will use kfp.dsl module contains domain-specific language objects used to compose pipelines.

Component: Decorator for Python-function-based components.

Pipeline: Decorator used to construct a pipeline.

2. Data Preparation

We are using Python 3.9 image that has a Python interpreter that is compatible with KFP and specifies required libraries in the

`packages_to_install` parameter for data preparation.

Kubeflow will pull the Python 3.9 image from the docker hub if it is not already present locally, install the specified libraries, and drop NA values from the IRIS dataset.

```
@component(
    packages_to_install=["pandas", "numpy", "scikit-learn"],
    base_image="python:3.9",
)
def prepare_data(data_path: str):
    import pandas as pd
    import os
    from sklearn import datasets

    # Load dataset
    iris = datasets.load_iris()
    df = pd.DataFrame(iris.data, columns=iris.feature_names)
    df['species'] = iris.target

    df = df.dropna()
    df.to_csv(f'{data_path}/final_df.csv', index=False)
```

3. Split data into Train and Test set

We use a similar component as data preparation in this step.

```
@component(
    packages_to_install=["pandas", "numpy", "scikit-learn"],
    base_image="python:3.9",
)
def train_test_split(data_path: str):
    import pandas as pd
    import numpy as np
    import os
    from sklearn.model_selection import train_test_split

    final_data = pd.read_csv(f'{data_path}/final_df.csv')

    target_column = 'species'
    X = final_data.loc[:, final_data.columns != target_column]
    y = final_data.loc[:, final_data.columns == target_column]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stra

    np.save(f'{data_path}/X_train.npy', X_train)
    np.save(f'{data_path}/X_test.npy', X_test)
    np.save(f'{data_path}/y_train.npy', y_train)
    np.save(f'{data_path}/y_test.npy', y_test)
```

4. Model training

We use a similar component as data preparation in this step and train a logistic regression model.

```

@Component(
    packages_to_install=["pandas", "numpy", "scikit-learn"],
    base_image="python:3.9",
)
def training_basic_classifier(data_path: str):
    import pandas as pd
    import numpy as np
    import os
    from sklearn.linear_model import LogisticRegression

    X_train = np.load(f'{data_path}/X_train.npy', allow_pickle=True)
    y_train = np.load(f'{data_path}/y_train.npy', allow_pickle=True)

    classifier = LogisticRegression(max_iter=500)
    classifier.fit(X_train, y_train)
    import pickle
    with open(f'{data_path}/model.pkl', 'wb') as f:
        pickle.dump(classifier, f)

```

Summary

1. The article focuses on building an end-to-end machine learning workflow using Minikube, Kubeflow Pipelines, MLflow, and KServe.
2. Minikube was introduced as a tool to run Kubernetes locally, providing a simple and controlled environment for testing and development.
3. Kubeflow Pipelines, a platform for scalable ML workflows on Kubernetes, is deployed on Minikube, offering an SDK for declaring pipelines as code.

4. The article highlights key Minikube commands for managing the cluster and checking its status.
5. Kubeflow Pipelines UI is accessed through port-forwarding, providing a user-friendly interface for monitoring pipeline runs.
6. The upcoming posts in the series promise insights into MLflow setup, model registration in the MLflow model registry, and model serving using KServe.

I hope you enjoyed this blog post, if you have any questions, feel free to contact me on LinkedIn and share your experience in the comment section.

Kubernetes

Kubeflow

MLflow

Mlops



Written by Vinayak Shanawad

Following




156 Followers · Writer for Analytics Vidhya

Machine Learning Engineer | 3x Kaggle Expert | MLOps | LLMOps | Learning, improving and evolving.

More from Vinayak Shanawad and Analytics Vidhya



 Vinayak Shanawad

Serving Hugging Face Transformers: Optimizing Custo...

Transforming Deployment: Seldon Core Meets Hugging Face

Sep 14, 2023

 20

 1



 Kia Eisinga in Analytics Vidhya

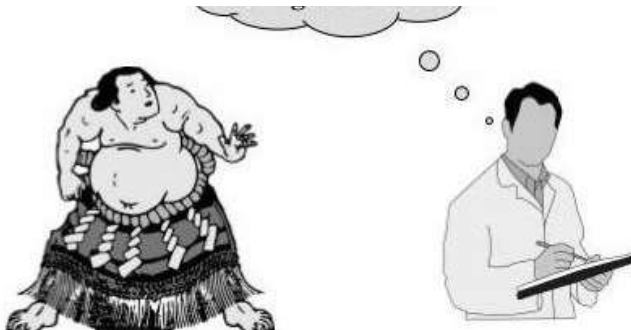
How to create a Python library

Ever wanted to create a Python library, albeit for your team at work or for some open...

Jan 27, 2020

 2.7K

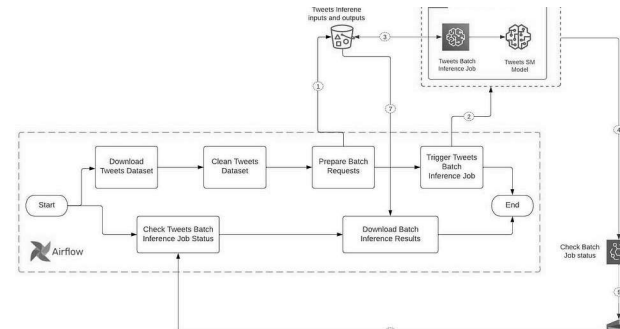
 30




 Hari Krishnan N B in Analytics Vidhya

Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric



 Vinayak Shanawad

Build an ML Inference Data Pipeline using SageMaker and...

Automate and streamline our ML inference pipeline with SageMaker and Airflow

Dec 10, 2019  1.1K  6



Mar 27, 2023

 68

 2



See all from Vinayak Shanawad

See all from Analytics Vidhya

Recommended from Medium



Peeush Agarwal

End-to-End MLOps Pipeline using MLFlow (Part-2)

Data Preprocessing, Model Training, and Hyperparameter Tuning

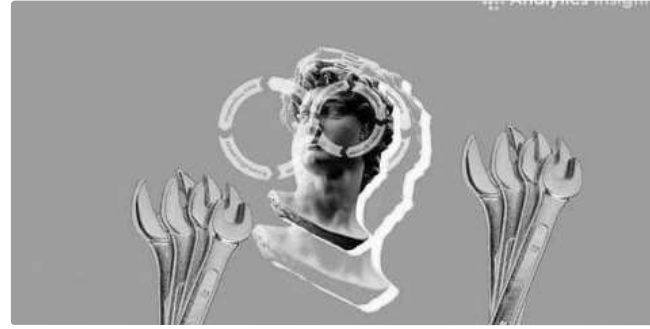
Oct 10



27



2



Analytics Insight

Top End-to-End Open-Source MLOps Tools for 2024

Top Open-Source MLOps Tools for Efficient Machine Learning Operations in 2024

Sep 6



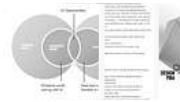
9



1



Lists



Business

41 stories · 154 saves



Natural Language Processing

1798 stories · 1408 saves



Raju Dawadi

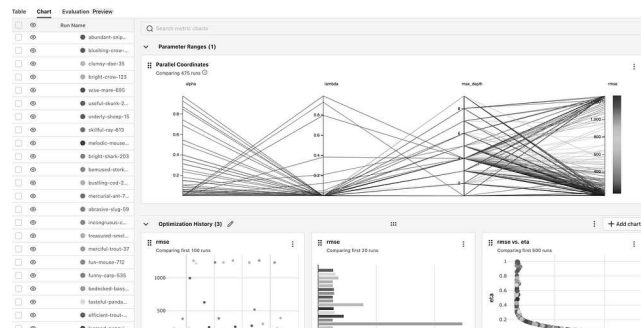
MLOps 101 with Kubeflow and Vertex AI

If you've ever tried taking an ML model from your local environment to production, you...

Oct 31



6



Anna

A Comprehensive Guide to MLflow: What It Is, Its Pros and Cons, and...

Machine learning (ML) development can be complex, involving many steps from data...

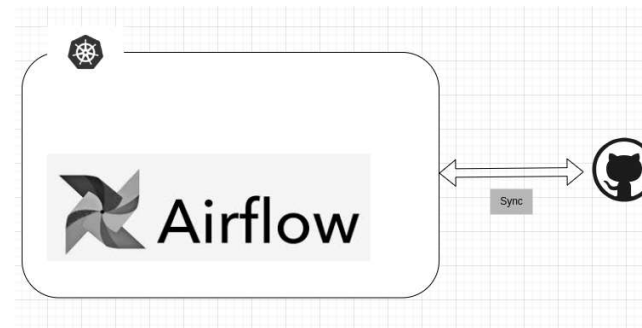


Ajith Kumar V

Setting Up a Workflow with DVC, Google Cloud Storage(GCS) buck...

Using Service Account Method with DVC in GitHub CI/CD Pipeline

Oct 21



Koushik Dutta in Dev Genius

Airflow with Git-Sync in Kubernetes

Install Airflow with Git-Sync in Kubernetes Cluster, so that once DAG is Pushed to Git...

May 30  62



Aug 8  4



See more recommendations

[Help](#) [Status](#) [About](#) [Careers](#) [Press](#) [Blog](#) [Privacy](#) [Terms](#) [Text to speech](#) [Teams](#)