Data analysis.py

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.express as px

import boto3  # For AWS S3 integration

import psycopg2  # For PostgreSQL integration

from io import StringIO

# Sample server log data (replace with actual log data)

data = {

    "timestamp": ["2023-10-01 12:00", "2023-10-01 12:05", "2023-10-01 12:10", "2023-10-01 12:15"],

    "server\_id": ["srv1", "srv2", "srv1", "srv2"],

    "cpu\_usage": [75, 60, 85, 90],

    "memory\_usage": [50, 45, 55, 65],

    "status": ["ok", "ok", "high\_cpu", "high\_memory"]

}

# Convert to Pandas DataFrame

df = pd.DataFrame(data)

# Data Analysis with Pandas and NumPy

print("Server Logs Summary:")

print(df.describe())

# Calculate average CPU and memory usage

avg\_cpu = np.mean(df["cpu\_usage"])

avg\_memory = np.mean(df["memory\_usage"])

print(f"\nAverage CPU Usage: {avg\_cpu}%")

print(f"Average Memory Usage: {avg\_memory}%")

# Data Visualization with Matplotlib

plt.figure(figsize=(10, 5))

plt.plot(df["timestamp"], df["cpu\_usage"], label="CPU Usage", marker="o")

plt.plot(df["timestamp"], df["memory\_usage"], label="Memory Usage", marker="x")

plt.title("Server CPU and Memory Usage Over Time")

plt.xlabel("Timestamp")

plt.ylabel("Usage (%)")

plt.legend()

plt.grid()

plt.show()

# Data Visualization with Plotly (Interactive)

fig = px.line(df, x="timestamp", y=["cpu\_usage", "memory\_usage"], title="Server Usage Over Time")

fig.show()

# Cloud Integration: Upload logs to AWS S3

def upload\_to\_s3(dataframe, bucket\_name, file\_name):

    s3 = boto3.client("s3")

    csv\_buffer = StringIO()

    dataframe.to\_csv(csv\_buffer, index=False)

    s3.put\_object(Bucket=bucket\_name, Key=file\_name, Body=csv\_buffer.getvalue())

    print(f"Uploaded {file\_name} to S3 bucket {bucket\_name}")

# Replace with your S3 bucket name and file name

upload\_to\_s3(df, "your-s3-bucket-name", "server\_logs.csv")

# Database Management: Store logs in PostgreSQL

def insert\_logs\_to\_postgres(dataframe, table\_name):

    conn = psycopg2.connect(

        dbname="your\_db\_name",

        user="your\_db\_user",

        password="your\_db\_password",

        host="your\_db\_host",

        port="your\_db\_port"

    )

    cursor = conn.cursor()

    # Create table if it doesn't exist

    cursor.execute(f"""

        CREATE TABLE IF NOT EXISTS {table\_name} (

            timestamp TIMESTAMP,

            server\_id VARCHAR(50),

            cpu\_usage FLOAT,

            memory\_usage FLOAT,

            status VARCHAR(50)

    """)

    # Insert data into the table

    for \_, row in dataframe.iterrows():

        cursor.execute(f"""

            INSERT INTO {table\_name} (timestamp, server\_id, cpu\_usage, memory\_usage, status)

            VALUES (%s, %s, %s, %s, %s)

        """, (row["timestamp"], row["server\_id"], row["cpu\_usage"], row["memory\_usage"], row["status"]))

    conn.commit()

    cursor.close()

    conn.close()

    print(f"Inserted logs into PostgreSQL table {table\_name}")

# Replace with your table name

insert\_logs\_to\_postgres(df, "server\_logs")

Output :

Server Logs Summary:

cpu\_usage memory\_usage

count 4.000000 4.000000

mean 77.500000 53.750000

std 13.228757 8.539126

min 60.000000 45.000000

25% 71.250000 48.750000

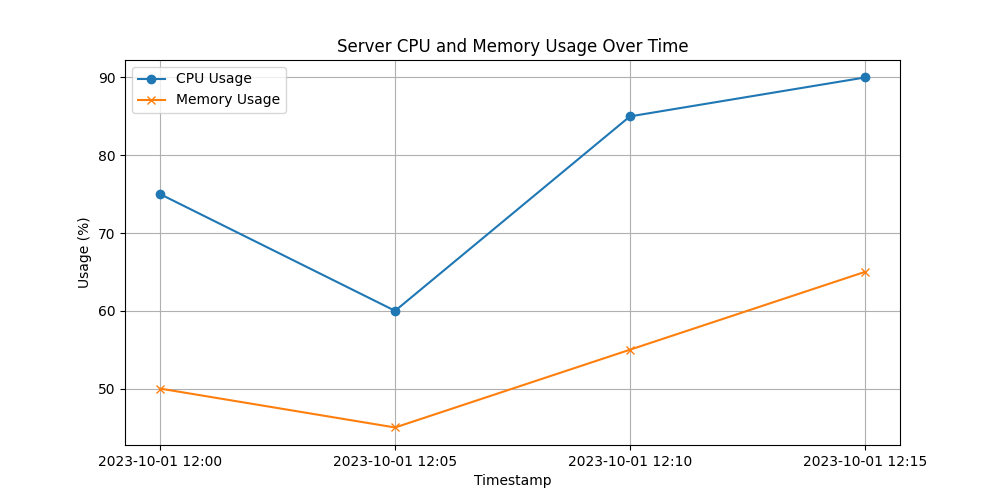
50% 80.000000 52.500000

75% 86.250000 57.500000

max 90.000000 65.000000

Average CPU Usage: 77.5%

Average Memory Usage: 53.75



Visualization.py

import time

import random

import threading

from datetime import datetime

# --- Hypothetical Data Sources and Libraries ---

class InfrastructureData:

    def get\_server\_metrics(self, server\_id):

        return {

            "cpu\_usage": random.uniform(10, 95),

            "memory\_usage": random.uniform(20, 90),

            "disk\_io": random.uniform(5, 80),

            "network\_traffic": random.uniform(100, 1000),

        }

    def get\_database\_metrics(self, db\_cluster\_id):

        return {

            "query\_latency": random.uniform(1, 100),

            "active\_connections": random.randint(50, 500),

        }

class AnomalyDetector:

    def detect\_cpu\_anomaly(self, cpu\_usage):

        if cpu\_usage > 90:

            return "High CPU Usage Alert", 95, "high"

        return None

    def detect\_latency\_anomaly(self, latency):

        if latency > 80:

            return "High Latency Alert", 90, "critical"

        return None

    def predict\_disk\_failure(self, disk\_io):

        if disk\_io > 70:

            if random.random() < 0.2:  # 20% chance of prediction.

                return "Predicted Disk Failure", 98, "critical"

        return None

# --- Visualization (Simplified) ---

class Visualization:

    def \_\_init\_\_(self):

        self.server\_status = {}  # Initialize as a dictionary

        self.db\_status = {}  # Initialize for database statuses

        self.alerts = []  # Initialize alerts as an empty list

    def update\_server\_status(self, server\_id, status):

        self.server\_status[server\_id] = status

    def update\_db\_status(self, db\_id, metrics):

        latency = metrics["query\_latency"]

        if latency > 80:

            status = "red"

        elif latency > 60:

            status = "yellow"

        else:

            status = "green"

        self.db\_status[db\_id] = status

        print(f"DB {db\_id} status: {status}")

    def add\_alert(self, message, severity, timestamp):

        self.alerts.append({"message": message, "severity": severity, "timestamp": timestamp})

# --- Customizable Dashboard (Simplified) ---

class Dashboard:

    def \_\_init\_\_(self, visualization):

        self.visualization = visualization

        self.widgets = []  # Store widgets in a list

    def add\_widget(self, widget):

        self.widgets.append(widget)

        print(f"Widget '{widget}' added to the dashboard.")

    def display(self):

        print("\n--- Dashboard View ---")

        print(f"Server Status: {self.visualization.server\_status}")

        print(f"DB Status: {self.visualization.db\_status}")

        print(f"Alerts: {self.visualization.alerts}\n")

# --- Main Program ---

def main():

    data\_source = InfrastructureData()

    anomaly\_detector = AnomalyDetector()

    visualization = Visualization()

    dashboard = Dashboard(visualization)

    dashboard.add\_widget("server\_status")

    dashboard.add\_widget("db\_status")

    dashboard.add\_widget("alerts")

    servers = ["SRV-001", "SRV-002", "SRV-003"]

    databases = ["DB-Cluster-01", "DB-Cluster-02"]

    def monitor():

        while True:

            for server in servers:

                metrics = data\_source.get\_server\_metrics(server)

                visualization.update\_server\_status(server, metrics)

                anomaly = anomaly\_detector.detect\_cpu\_anomaly(metrics["cpu\_usage"])

                prediction = anomaly\_detector.predict\_disk\_failure(metrics["disk\_io"])

                if anomaly:

                    visualization.add\_alert(anomaly[0], anomaly[2], datetime.now())

                if prediction:

                    visualization.add\_alert(prediction[0], prediction[2], datetime.now())

            for db in databases:

                metrics = data\_source.get\_database\_metrics(db)

                visualization.update\_db\_status(db, metrics)

                anomaly = anomaly\_detector.detect\_latency\_anomaly(metrics["query\_latency"])

                if anomaly:

                    visualization.add\_alert(anomaly[0], anomaly[2], datetime.now())

            dashboard.display()

            time.sleep(2)

    monitor\_thread = threading.Thread(target=monitor)

    monitor\_thread.daemon = True  # Allow program to exit even if thread is running

    monitor\_thread.start()

    try:

        while True:

            time.sleep(1)  # Keeps main thread alive.

    except KeyboardInterrupt:

        print("Monitoring stopped.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT :

tical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 9, 300754)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 11, 313325)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 13, 318261)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 17, 324721)}]

DB DB-Cluster-01 status: green

DB DB-Cluster-02 status: green

--- Dashboard View ---

Server Status: {'SRV-001': {'cpu\_usage': 65.58037417209809, 'memory\_usage': 54.33555323675775, 'disk\_io': 14.475976372236058, 'network\_traffic': 118.91986780193265}, 'SRV-002': {'cpu\_usage': 84.76552893080749, 'memory\_usage': 57.48045226871046, 'disk\_io': 52.95620438212628, 'network\_traffic': 296.85416394676406}, 'SRV-003': {'cpu\_usage': 51.29573449422377, 'memory\_usage': 43.10535166954207, 'disk\_io': 33.69952732475104, 'network\_traffic': 595.8864154646035}}

DB Status: {'DB-Cluster-01': 'green', 'DB-Cluster-02': 'green'}

Alerts: [{'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 48, 848487)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 54, 864408)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 0, 878425)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 4, 885228)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 8, 894738)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 16, 943647)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 31, 10549)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 41, 32762)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 43, 37473)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 45, 40445)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 49, 47389)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 51, 52687)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 53, 57808)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 57, 67362)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 59, 71369)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 19, 118230)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 23, 124671)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 45, 184947)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 47, 209514)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 55, 264264)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 57, 268541)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 59, 277844)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 5, 290932)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 9, 300754)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 11, 313325)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 13, 318261)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 17, 324721)}]

DB DB-Cluster-01 status: yellow

DB DB-Cluster-02 status: green

--- Dashboard View ---

Server Status: {'SRV-001': {'cpu\_usage': 86.88416539712946, 'memory\_usage': 52.10882264413986, 'disk\_io': 20.1873124679649, 'network\_traffic': 590.7451733116084}, 'SRV-002': {'cpu\_usage': 45.265745853531236, 'memory\_usage': 43.22626335575461, 'disk\_io': 45.36835668514759, 'network\_traffic': 165.43754062129557}, 'SRV-003': {'cpu\_usage': 34.66667459766009, 'memory\_usage': 48.595662036741246, 'disk\_io': 21.088530785731965, 'network\_traffic': 687.5629762381094}}

DB Status: {'DB-Cluster-01': 'yellow', 'DB-Cluster-02': 'green'}

Alerts: [{'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 48, 848487)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 54, 864408)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 0, 878425)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 4, 885228)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 8, 894738)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 16, 943647)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 31, 10549)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 41, 32762)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 43, 37473)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 45, 40445)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 49, 47389)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 51, 52687)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 53, 57808)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 57, 67362)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 59, 71369)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 19, 118230)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 23, 124671)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 45, 184947)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 47, 209514)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 55, 264264)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 57, 268541)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 59, 277844)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 5, 290932)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 9, 300754)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 11, 313325)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 13, 318261)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 17, 324721)}]

DB DB-Cluster-01 status: green

DB DB-Cluster-02 status: green

--- Dashboard View ---

Server Status: {'SRV-001': {'cpu\_usage': 34.422971085087596, 'memory\_usage': 32.519992981465734, 'disk\_io': 26.62510068032267, 'network\_traffic': 739.6291615858086}, 'SRV-002': {'cpu\_usage': 78.4228613505935, 'memory\_usage': 42.92781452742204, 'disk\_io': 33.77502565352653, 'network\_traffic': 767.3227495203773}, 'SRV-003': {'cpu\_usage': 53.17241611203543, 'memory\_usage': 39.124360438656296, 'disk\_io': 71.57397869526223, 'network\_traffic': 671.5937391613264}}

DB Status: {'DB-Cluster-01': 'green', 'DB-Cluster-02': 'green'}

Alerts: [{'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 48, 848487)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 54, 864408)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 37, 58, 874477)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 0, 878425)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 2, 881524)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 4, 885228)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 6, 889539)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 8, 894738)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 14, 937478)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 16, 943647)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 31, 10549)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 39, 29753)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 41, 32762)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 43, 37473)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 45, 40445)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 49, 47389)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 51, 52687)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 53, 57808)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 55, 63155)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 57, 67362)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 38, 59, 71369)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 19, 118230)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 21, 121106)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 23, 124671)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 33, 154594)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 41, 174158)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 45, 184947)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 47, 209514)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 55, 264264)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 57, 268541)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 39, 59, 277844)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 1, 282075)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 5, 290932)}, {'message': 'Predicted Disk Failure', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 9, 300754)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 11, 313325)}, {'message': 'High CPU Usage Alert', 'severity': 'high', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 13, 318261)}, {'message': 'High Latency Alert', 'severity': 'critical', 'timestamp': datetime.datetime(2025, 3, 6, 13, 40, 17, 324721)}]

Anomaly detection.py

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as go

# Sample Data Generation (Replace with your actual data source)

np.random.seed(42)

time\_index = pd.date\_range('2023-01-01', periods=1000, freq='H')

cpu\_usage = np.random.normal(50, 10, 1000)

memory\_usage = np.random.normal(60, 15, 1000)

network\_traffic = np.random.normal(100, 20, 1000)

database\_latency = np.random.normal(5, 1, 1000)

df = pd.DataFrame({

    'timestamp': time\_index,

    'cpu\_usage': cpu\_usage,

    'memory\_usage': memory\_usage,

    'network\_traffic': network\_traffic,

    'database\_latency': database\_latency

})

df.set\_index('timestamp', inplace=True)

# Add Anomaly Detection (using a simplified threshold example for visualization)

df['cpu\_anomaly'] = df['cpu\_usage'].apply(lambda x: 1 if abs(x - df['cpu\_usage'].mean()) > 2 \* df['cpu\_usage'].std() else 0)

df['latency\_anomaly'] = df['database\_latency'].apply(lambda x: 1 if abs(x - df['database\_latency'].mean()) > 2 \* df['database\_latency'].std() else 0)

# 1. Matplotlib: Time Series with Anomalies

def plot\_time\_series\_matplotlib(df, column, anomaly\_column):

    """Plots time series with anomalies using Matplotlib."""

    anomalies = df[df[anomaly\_column] == 1]

    plt.figure(figsize=(12, 6))

    plt.plot(df[column], label=column)

    plt.scatter(anomalies.index, anomalies[column], color='red', label='Anomaly')

    plt.title(f'{column} Time Series with Anomalies (Matplotlib)')

    plt.legend()

    plt.show()

plot\_time\_series\_matplotlib(df, 'cpu\_usage', 'cpu\_anomaly')

# 2. Seaborn: Distribution and Correlation

def plot\_distribution\_seaborn(df, column):

    """Plots distribution using Seaborn."""

    plt.figure(figsize=(8, 6))

    sns.histplot(df[column], kde=True)

    plt.title(f'{column} Distribution (Seaborn)')

    plt.show()

plot\_distribution\_seaborn(df, 'network\_traffic')

def plot\_correlation\_seaborn(df, col1, col2):

    """Plots correlation using Seaborn."""

    plt.figure(figsize=(8, 6))

    sns.scatterplot(x=col1, y=col2, data=df)

    plt.title(f'{col1} vs {col2} (Seaborn)')

    plt.show()

plot\_correlation\_seaborn(df, 'cpu\_usage', 'memory\_usage')

# 3. Plotly: Interactive Time Series and Scatter Plots

def plot\_time\_series\_plotly(df, column, anomaly\_column):

    """Plots interactive time series with anomalies using Plotly."""

    fig = px.line(df, x=df.index, y=column, title=f'{column} Time Series (Plotly)')

    anomalies = df[df[anomaly\_column] == 1]

    fig.add\_trace(go.Scatter(x=anomalies.index, y=anomalies[column], mode='markers', marker=dict(color='red'), name='Anomaly'))

    fig.show()

plot\_time\_series\_plotly(df, 'database\_latency', 'latency\_anomaly')

def plot\_scatter\_plotly(df, col1, col2):

    """Plots interactive scatter plot using Plotly."""

    fig = px.scatter(df, x=col1, y=col2, title=f'{col1} vs {col2} (Plotly)')

    fig.show()

plot\_scatter\_plotly(df, 'network\_traffic', 'cpu\_usage')

# 4. Plotly: Interactive Dashboard (Example)

def create\_dashboard\_plotly(df):

    """Creates a simple interactive dashboard using Plotly."""

    fig\_cpu = px.line(df, x=df.index, y='cpu\_usage', title='CPU Usage')

    fig\_memory = px.line(df, x=df.index, y='memory\_usage', title='Memory Usage')

    fig\_network = px.line(df, x=df.index, y='network\_traffic', title='Network Traffic')

    fig\_latency = px.line(df, x=df.index, y='database\_latency', title='Database Latency')

    from plotly.subplots import make\_subplots

    fig = make\_subplots(rows=2, cols=2, subplot\_titles=('CPU Usage', 'Memory Usage', 'Network Traffic', 'Database Latency'))

    fig.add\_trace(fig\_cpu.data[0], row=1, col=1)

    fig.add\_trace(fig\_memory.data[0], row=1, col=2)

    fig.add\_trace(fig\_network.data[0], row=2, col=1)

    fig.add\_trace(fig\_latency.data[0], row=2, col=2)

    fig.update\_layout(height=800, width=1200, title\_text="IT Infrastructure Dashboard")

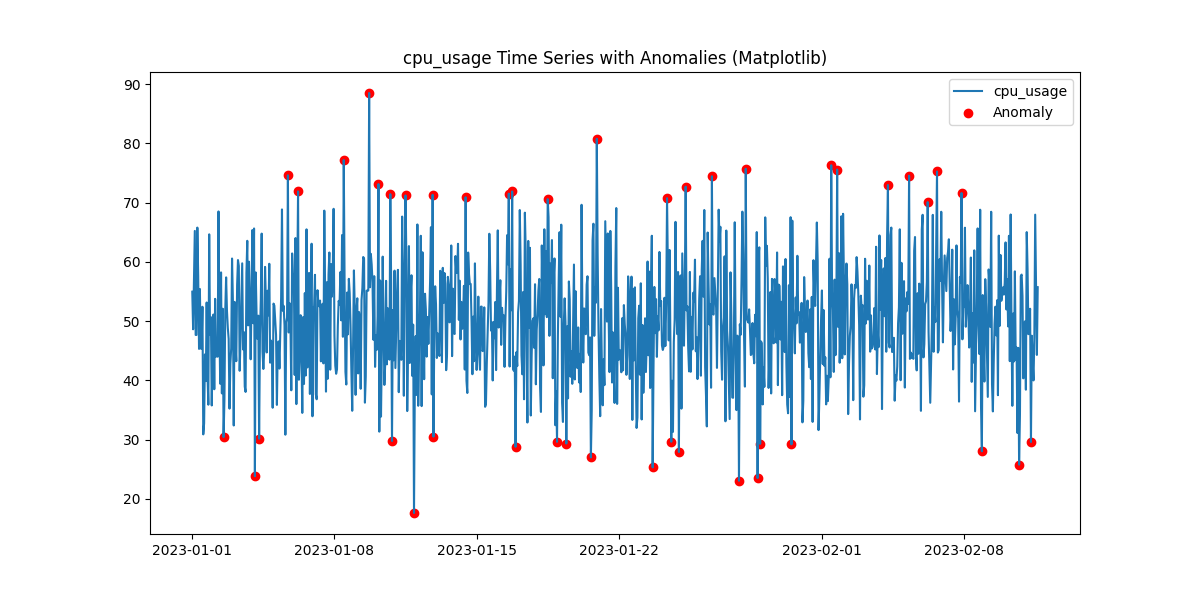
    fig.show()

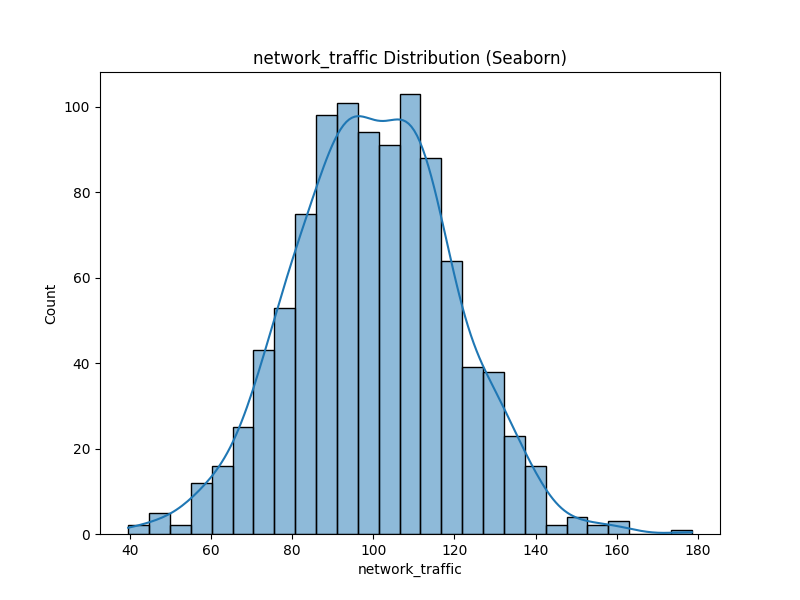
create\_dashboard\_plotly(df)

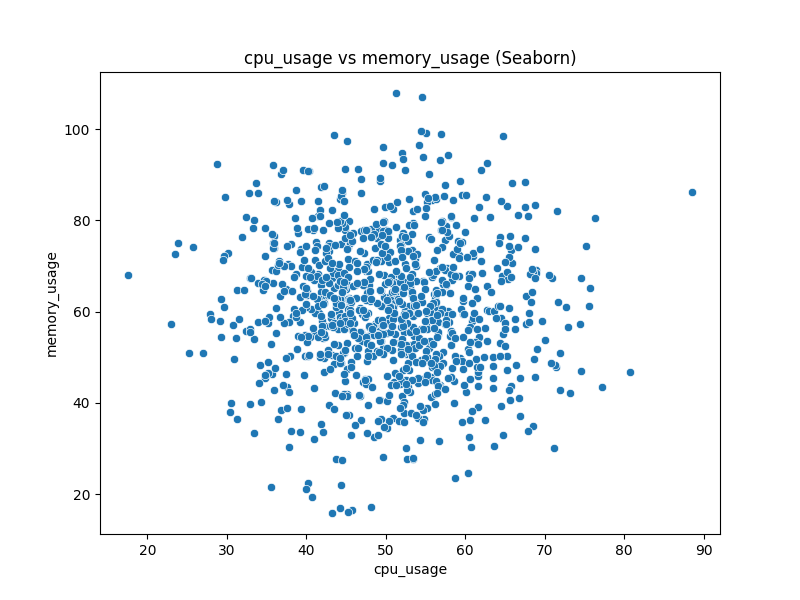
OUTPUT:

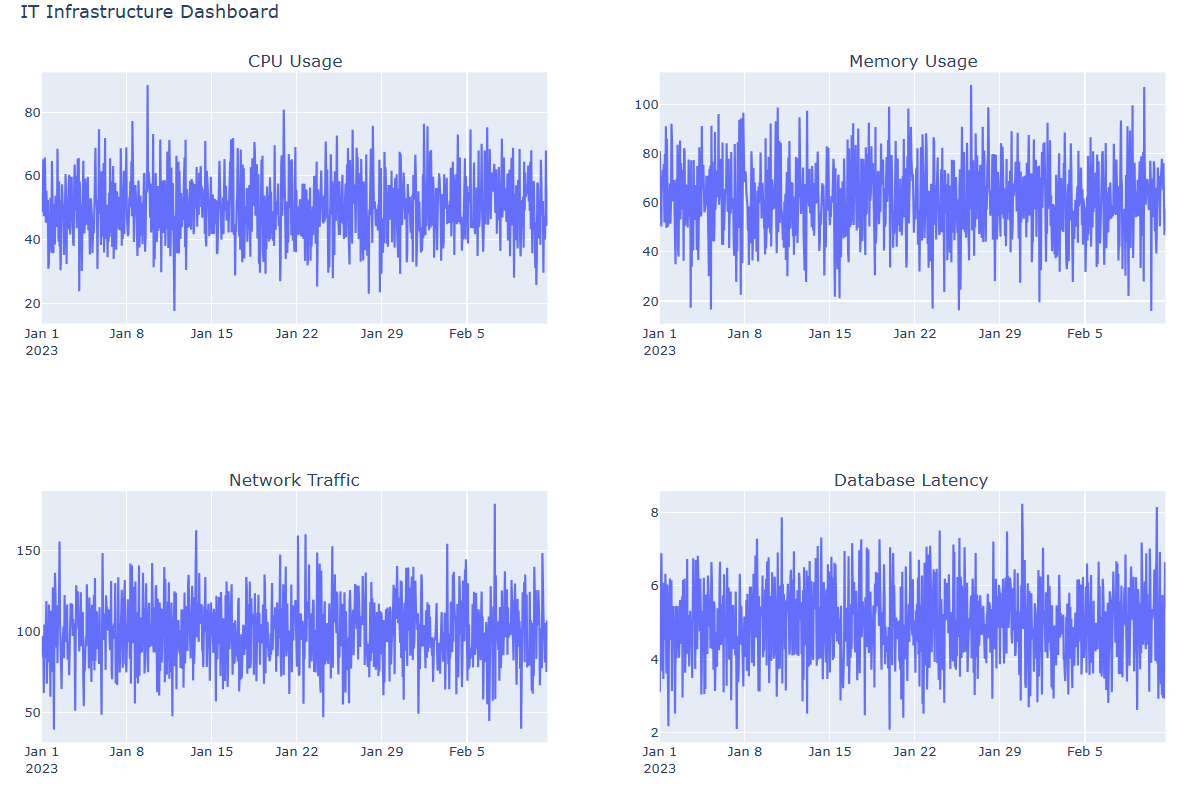
FutureWarning: 'H' is deprecated and will be removed in a future version, please use 'h' instead.

time\_index = pd.date\_range('2023-01-01', periods=1000, freq='H')









Cloud platform.py

import pandas as pd

import numpy as np

import datetime

# Sample Data Generation

np.random.seed(42)

time\_index = pd.date\_range('2023-01-01', periods=1000, freq='H')

cpu\_usage = np.random.normal(50, 10, 1000)

memory\_usage = np.random.normal(60, 15, 1000)

network\_traffic = np.random.normal(100, 20, 1000)

database\_latency = np.random.normal(5, 1, 1000)

df = pd.DataFrame({

    'timestamp': time\_index,

    'cpu\_usage': cpu\_usage,

    'memory\_usage': memory\_usage,

    'network\_traffic': network\_traffic,

    'database\_latency': database\_latency

})

# Preprocessing: Example (Add a day of week column)

df['day\_of\_week'] = df['timestamp'].dt.day\_name()

# Output

print("Sample Preprocessed Data:")

print(df.head())

# --- Cloud Platform Examples (Conceptual) ---

# AWS (Conceptual)

def aws\_data\_pipeline(df, bucket\_name, file\_key):

    """Conceptual AWS data pipeline (S3, Glue, Athena)."""

    # 1. Store data in S3

    # s3\_client.put\_object(Bucket=bucket\_name, Key=file\_key, Body=df.to\_csv(index=False))

    print(f"AWS: Data stored in S3 bucket '{bucket\_name}', key '{file\_key}'.")

    # 2. AWS Glue (ETL) - Example: Run a Glue job to transform data

    # glue\_client.start\_job\_run(JobName='my\_glue\_job')

    print("AWS: Glue job triggered (conceptual).")

    # 3. AWS Athena (Query) - Example: Query the data

    # athena\_client.start\_query\_execution(QueryString='SELECT \* FROM my\_table', ResultConfiguration={'OutputLocation': 's3://...'})

    print("AWS: Athena query executed (conceptual).")

# Azure (Conceptual)

def azure\_data\_pipeline(df, container\_name, file\_name):

    """Conceptual Azure data pipeline (Blob Storage, Data Factory, Synapse)."""

    # 1. Store data in Blob Storage

    # blob\_service\_client.get\_blob\_client(container=container\_name, blob=file\_name).upload\_blob(df.to\_csv(index=False))

    print(f"Azure: Data stored in Blob Storage container '{container\_name}', file '{file\_name}'.")

    # 2. Azure Data Factory (ETL) - Example: Run a Data Factory pipeline

    # data\_factory\_client.create\_pipeline\_run(resource\_group\_name='...', factory\_name='...', pipeline\_name='...')

    print("Azure: Data Factory pipeline triggered (conceptual).")

    # 3. Azure Synapse Analytics (Query) - Example: Query the data

    # synapse\_client.execute\_query(workspace\_name='...', sql\_query='SELECT \* FROM my\_table')

    print("Azure: Synapse query executed (conceptual).")

# GCP (Conceptual)

def gcp\_data\_pipeline(df, bucket\_name, blob\_name):

    """Conceptual GCP data pipeline (Cloud Storage, Dataflow, BigQuery)."""

    # 1. Store data in Cloud Storage

    # bucket = storage\_client.bucket(bucket\_name)

    # blob = bucket.blob(blob\_name)

    # blob.upload\_from\_string(df.to\_csv(index=False))

    print(f"GCP: Data stored in Cloud Storage bucket '{bucket\_name}', blob '{blob\_name}'.")

    # 2. Google Cloud Dataflow (ETL) - Example: Run a Dataflow job

    # dataflow\_client.projects().locations().templates().create(body={'gcsPath': 'gs://...'}, projectId='...', location='...')

    print("GCP: Dataflow job triggered (conceptual).")

    # 3. Google BigQuery (Query) - Example: Query the data

    # query\_job = bigquery\_client.query('SELECT \* FROM my\_dataset.my\_table')

    print("GCP: BigQuery query executed (conceptual).")

# Example Usage (Conceptual)

aws\_data\_pipeline(df, 'my-monitoring-bucket', 'monitoring\_data.csv')

azure\_data\_pipeline(df, 'monitoring-container', 'monitoring\_data.csv')

gcp\_data\_pipeline(df, 'my-monitoring-bucket', 'monitoring\_data.csv')

OUTPUT :

FutureWarning: 'H' is deprecated and will be removed in a future version, please use 'h' instead.

use 'h' instead.

time\_index = pd.date\_range('2023-01-01', periods=1000, freq='H')

Sample Preprocessed Data:

timestamp cpu\_usage memory\_usage network\_traffic database\_latency day\_of\_week

0 2023-01-01 00:00:00 54.967142 80.990332 86.496435 3.092192 Sunday

1 2023-01-01 01:00:00 48.617357 73.869505 97.109627 4.139615 Sunday

2 2023-01-01 02:00:00 56.476885 60.894456 84.151602 4.586394 Sunday

3 2023-01-01 03:00:00 65.230299 50.295948 93.840769 6.887688 Sunday

4 2023-01-01 04:00:00 47.658466 70.473350 62.127707 5.556553 Sunday

AWS: Data stored in S3 bucket 'my-monitoring-bucket', key 'monitoring\_data.csv'.

AWS: Glue job triggered (conceptual).

AWS: Athena query executed (conceptual).

Azure: Data stored in Blob Storage container 'monitoring-container', file 'monitoring\_data.csv'.

Azure: Data Factory pipeline triggered (conceptual).

Azure: Synapse query executed (conceptual).

GCP: Data stored in Cloud Storage bucket 'my-monitoring-bucket', blob 'monitoring\_data.csv'.

GCP: Dataflow job triggered (conceptual).

GCP: BigQuery query executed (conceptual).

Database1.py

from flask import Flask, request, jsonify, render\_template

from flask\_sqlalchemy import SQLAlchemy

import pickle

import numpy as np

# Initialize Flask app

app = Flask(\_\_name\_\_)

# Configure SQLite database

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///predictions.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

# Initialize the database

db = SQLAlchemy(app)

# Define the Prediction model

class Prediction(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    cpu\_usage = db.Column(db.Float, nullable=False)

    memory\_usage = db.Column(db.Float, nullable=False)

    disk\_usage = db.Column(db.Float, nullable=False)

    network\_usage = db.Column(db.Float, nullable=False)

    predicted\_capacity = db.Column(db.Float, nullable=False)

    def \_\_repr\_\_(self):

        return f"<Prediction {self.id}>"

# Create the database and tables

with app.app\_context():

    db.create\_all()

# Load the trained model

try:

    with open('model.pkl', 'rb') as f:

        model = pickle.load(f)

    print("Model loaded successfully!")

except Exception as e:

    print(f"Error loading model: {e}")

# Route to serve the frontend

@app.route('/')

def home():

    return render\_template('index.html')  # Serve the index.html template

# Route to handle prediction requests

@app.route('/predict', methods=['POST'])

def predict():

    try:

        # Get input data from the request

        data = request.get\_json()

        cpu\_usage = data['cpu\_usage']

        memory\_usage = data['memory\_usage']

        disk\_usage = data['disk\_usage']

        network\_usage = data['network\_usage']

        # Prepare input for the model

        input\_data = np.array([[cpu\_usage, memory\_usage, disk\_usage, network\_usage]])

        # Make prediction

        predicted\_capacity = model.predict(input\_data)[0]

        # Save the prediction to the database

        prediction = Prediction(

            cpu\_usage=cpu\_usage,

            memory\_usage=memory\_usage,

            disk\_usage=disk\_usage,

            network\_usage=network\_usage,

            predicted\_capacity=predicted\_capacity

        )

        db.session.add(prediction)

        db.session.commit()

        # Return the result

        return jsonify({

            'predicted\_capacity': predicted\_capacity

        })

    except Exception as e:

        return jsonify({

            'error': str(e)

        }), 500

# Route to view all predictions

@app.route('/predictions', methods=['GET'])

def get\_predictions():

    try:

        # Fetch all predictions from the database

        predictions = Prediction.query.all()

        predictions\_list = []

        for prediction in predictions:

            predictions\_list.append({

                'id': prediction.id,

                'cpu\_usage': prediction.cpu\_usage,

                'memory\_usage': prediction.memory\_usage,

                'disk\_usage': prediction.disk\_usage,

                'network\_usage': prediction.network\_usage,

                'predicted\_capacity': prediction.predicted\_capacity

            })

        return jsonify(predictions\_list)

    except Exception as e:

        return jsonify({

            'error': str(e)

        }), 500

# Run the Flask app

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Capacity Planning</title>

    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

    <div class="container">

        <h1>Capacity Planning for Predictive Maintenance</h1>

        <form id="capacityForm">

            <label for="cpuUsage">CPU Usage (%):</label>

            <input type="number" id="cpuUsage" name="cpuUsage" required>

            <label for="memoryUsage">Memory Usage (%):</label>

            <input type="number" id="memoryUsage" name="memoryUsage" required>

            <label for="diskUsage">Disk Usage (%):</label>

            <input type="number" id="diskUsage" name="diskUsage" required>

            <label for="networkUsage">Network Usage (%):</label>

            <input type="number" id="networkUsage" name="networkUsage" required>

            <button type="submit">Predict Capacity Requirement</button>

        </form>

        <h2>Predicted Capacity Requirement:</h2>

        <p id="predictionResult"></p>

        <h2>Predicted Capacity Requirement:</h2>

<p id="predictionResult"></p>

<!-- Add a canvas for the graph -->

<canvas id="capacityChart" width="400" height="200"></canvas>

        <button id="viewPredictions">View All Predictions</button>

        <div id="predictionsList"></div>

    </div>

    <script src="{{ url\_for('static', filename='script.js') }}"></script>

</body>

</html>

Styles.css

/\* General Styles \*/

body {

    font-family: 'Poppins', sans-serif;

    background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);

    margin: 0;

    padding: 0;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    color: #333;

    overflow: hidden;

}

/\* Container \*/

.container {

    background: rgba(255, 255, 255, 0.9);

    padding: 30px;

    border-radius: 20px;

    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

    backdrop-filter: blur(10px);

    border: 1px solid rgba(255, 255, 255, 0.2);

    width: 400px;

    text-align: center;

    animation: fadeIn 1s ease, slideIn 0.8s ease;

}

/\* Heading \*/

h1 {

    margin-bottom: 20px;

    font-size: 32px;

    font-weight: 600;

    color: #4a90e2;

    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1);

}

/\* Labels \*/

label {

    display: block;

    margin-bottom: 10px;

    font-weight: 500;

    color: #555;

}

/\* Input Fields \*/

input {

    width: 100%;

    padding: 12px;

    margin-bottom: 20px;

    border: 1px solid rgba(0, 0, 0, 0.1);

    border-radius: 10px;

    background: rgba(255, 255, 255, 0.9);

    color: #333;

    font-size: 16px;

    transition: all 0.3s ease;

}

input:focus {

    border-color: #4a90e2;

    outline: none;

    background: rgba(255, 255, 255, 1);

    box-shadow: 0 0 10px rgba(74, 144, 226, 0.3);

}

/\* Buttons \*/

button {

    width: 100%;

    padding: 12px;

    background: linear-gradient(135deg, #4a90e2 0%, #6a5acd 100%);

    color: #fff;

    border: none;

    border-radius: 10px;

    font-size: 16px;

    font-weight: 600;

    cursor: pointer;

    transition: all 0.3s ease;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

button:hover {

    background: linear-gradient(135deg, #6a5acd 0%, #4a90e2 100%);

    transform: translateY(-3px);

    box-shadow: 0 6px 8px rgba(0, 0, 0, 0.2);

}

button:active {

    transform: translateY(0);

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

/\* Prediction Result \*/

#predictionResult {

    font-size: 20px;

    font-weight: 600;

    margin-top: 20px;

    color: #4a90e2;

    text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);

    animation: fadeIn 0.5s ease;

}

/\* View Predictions Button \*/

#viewPredictions {

    margin-top: 20px;

    background: linear-gradient(135deg, #ff7e5f 0%, #feb47b 100%);

}

#viewPredictions:hover {

    background: linear-gradient(135deg, #feb47b 0%, #ff7e5f 100%);

}

/\* Predictions Table \*/

#predictionsList {

    margin-top: 20px;

    animation: fadeIn 0.8s ease;

}

table {

    width: 100%;

    border-collapse: collapse;

    margin-top: 20px;

    background: rgba(255, 255, 255, 0.9);

    border-radius: 10px;

    overflow: hidden;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

th, td {

    padding: 12px;

    text-align: center;

    border-bottom: 1px solid rgba(0, 0, 0, 0.1);

}

th {

    background: rgba(74, 144, 226, 0.1);

    font-weight: 600;

    color: #4a90e2;

}

tr:hover {

    background: rgba(74, 144, 226, 0.05);

    transform: scale(1.02);

    transition: all 0.3s ease;

}

/\* Animations \*/

@keyframes fadeIn {

    from {

        opacity: 0;

    }

    to {

        opacity: 1;

    }

}

@keyframes slideIn {

    from {

        transform: translateY(-20px);

    }

    to {

        transform: translateY(0);

    }

}

/\* Background Animation \*/

body::before {

    content: '';

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: linear-gradient(135deg, rgba(245, 247, 250, 0.8) 0%, rgba(195, 207, 226, 0.8) 100%);

    z-index: -1;

    animation: moveBackground 10s infinite alternate;

}

@keyframes moveBackground {

    0% {

        transform: translateY(0);

    }

    100% {

        transform: translateY(-20px);

    }

}

/\* Add a subtle pulse animation to the container \*/

@keyframes pulse {

    0% {

        transform: scale(1);

    }

    50% {

        transform: scale(1.02);

    }

    100% {

        transform: scale(1);

    }

}

.container:hover {

    animation: pulse 2s infinite;

}

Script.js

document.getElementById('capacityForm').addEventListener('submit', async function (event) {

    event.preventDefault();

    // Get input values

    const cpuUsage = parseFloat(document.getElementById('cpuUsage').value);

    const memoryUsage = parseFloat(document.getElementById('memoryUsage').value);

    const diskUsage = parseFloat(document.getElementById('diskUsage').value);

    const networkUsage = parseFloat(document.getElementById('networkUsage').value);

    // Prepare data for the API request

    const inputData = {

        cpu\_usage: cpuUsage,

        memory\_usage: memoryUsage,

        disk\_usage: diskUsage,

        network\_usage: networkUsage

    };

    try {

        // Send data to the backend API

        const response = await fetch('/predict', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json'

            },

            body: JSON.stringify(inputData)

        });

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        // Get the prediction result

        const result = await response.json();

        document.getElementById('predictionResult').textContent = `Predicted Capacity Requirement: ${result.predicted\_capacity.toFixed(2)}`;

    } catch (error) {

        console.error('Error:', error);

        document.getElementById('predictionResult').textContent = 'Error predicting capacity requirement.';

    }

});

// Fetch and display all predictions

document.getElementById('viewPredictions').addEventListener('click', async function () {

    try {

        const response = await fetch('/predictions');

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        const predictions = await response.json();

        const predictionsList = document.getElementById('predictionsList');

        predictionsList.innerHTML = '<h2>All Predictions</h2>';

        if (predictions.length === 0) {

            predictionsList.innerHTML += '<p>No predictions found.</p>';

        } else {

            const table = document.createElement('table');

            table.innerHTML = `

                <tr>

                    <th>ID</th>

                    <th>CPU Usage</th>

                    <th>Memory Usage</th>

                    <th>Disk Usage</th>

                    <th>Network Usage</th>

                    <th>Predicted Capacity</th>

                </tr>

            `;

            predictions.forEach(prediction => {

                table.innerHTML += `

                    <tr>

                        <td>${prediction.id}</td>

                        <td>${prediction.cpu\_usage}</td>

                        <td>${prediction.memory\_usage}</td>

                        <td>${prediction.disk\_usage}</td>

                        <td>${prediction.network\_usage}</td>

                        <td>${prediction.predicted\_capacity.toFixed(2)}</td>

                    </tr>

                `;

            });

            predictionsList.appendChild(table);

        }

    } catch (error) {

        console.error('Error:', error);

    }

});

Output :

