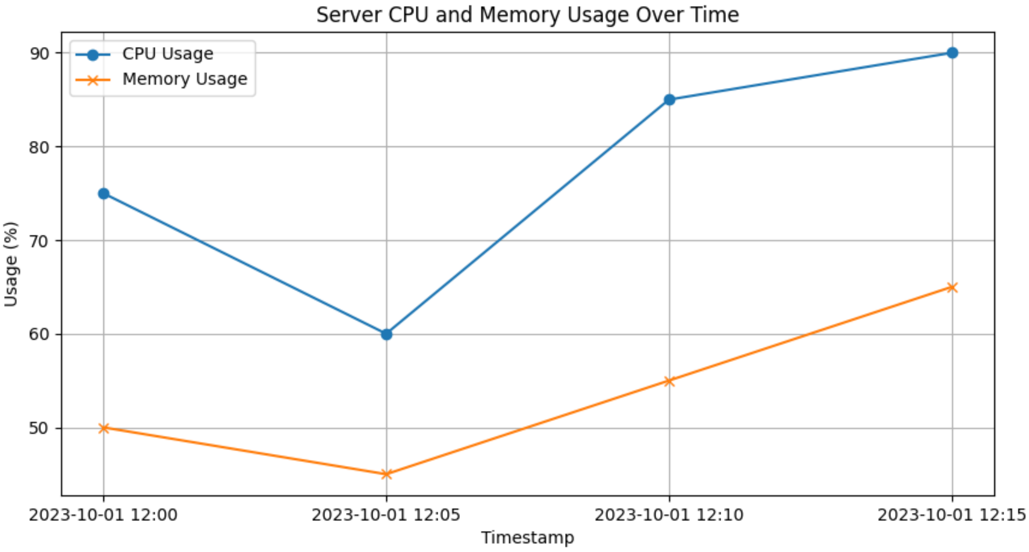
App.py

```
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 = {}
        self.db_status = {}
        self.alerts = []

    def update_server_status(self, server_id, metrics):
        cpu = metrics["cpu_usage"]
        if cpu > 80:
            status = "yellow"
        elif cpu > 90:
            status = "red"
        else:
            status = "green"
        self.server_status[server_id] = status
        print(f"Server {server_id} status: {status}")

    def update_db_status(self, db_id, metrics):
        latency = metrics["query_latency"]
        if latency > 60:
            status = "yellow"
        elif latency > 80:
            status = "red"
        else:
            status = "green"
        self.db_status[db_id] = status
        print(f"DB {db_id} status: {status}")

    def display_alerts(self):
        for alert in self.alerts:
            print(f"Alert: {alert['message']} (Severity: {alert['severity']})")

    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 = []  # Add this if needed

    def add_widget(self, widget):
        self.widgets.append(widget)  # Store widgets in a list
        print(f"Widget '{widget}' added to the dashboard.")

def main():
    visualization = "Sample Chart"
    dashboard = Dashboard(visualization)  # Ensure correct instantiation
    dashboard.add_widget("server_status")  # Call the method correctly


if __name__ == "__main__":
    main()



# --- 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 :

```
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
```


Server CPU and Memory Usage Over Time
CPU Usage
Memory Usage
Usage (%)
90
80
70
60
50
2023-10-01 12:00
2023-10-01 12:05
2023-10-01 12:10
2023-10-01 12:15
Timestamp

Sony.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)}]
```

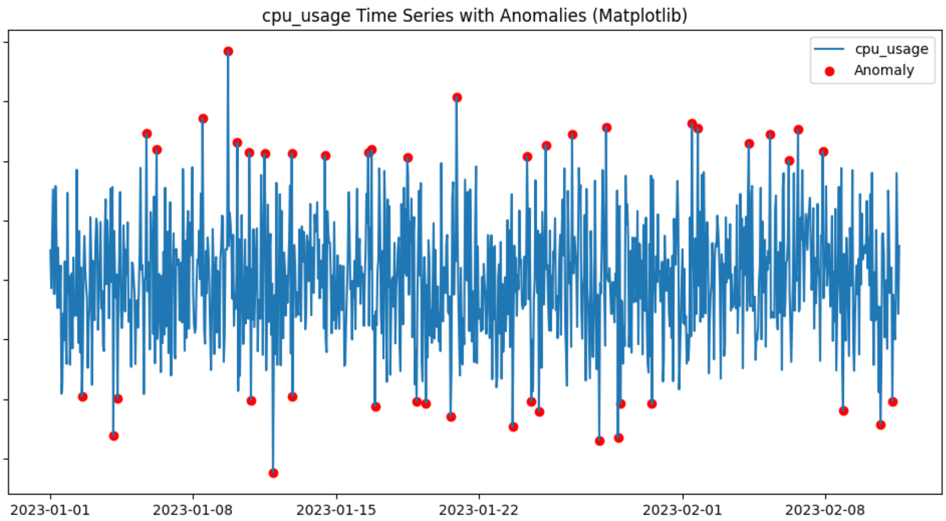
dv.py

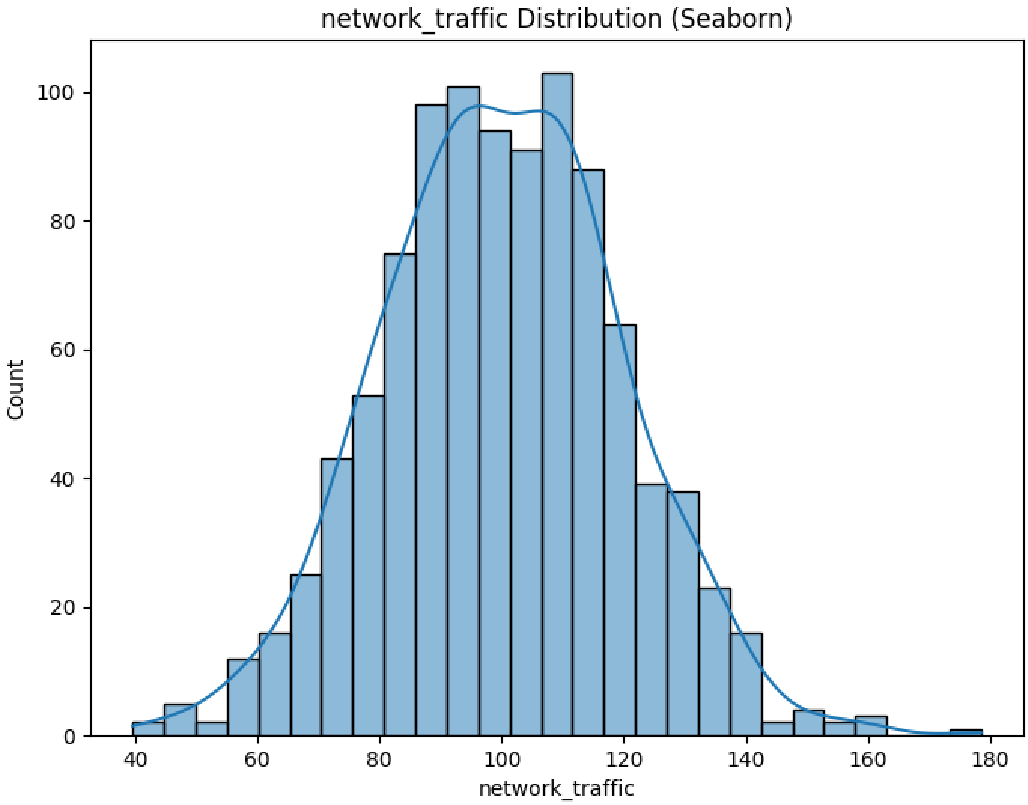
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

# 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)
```

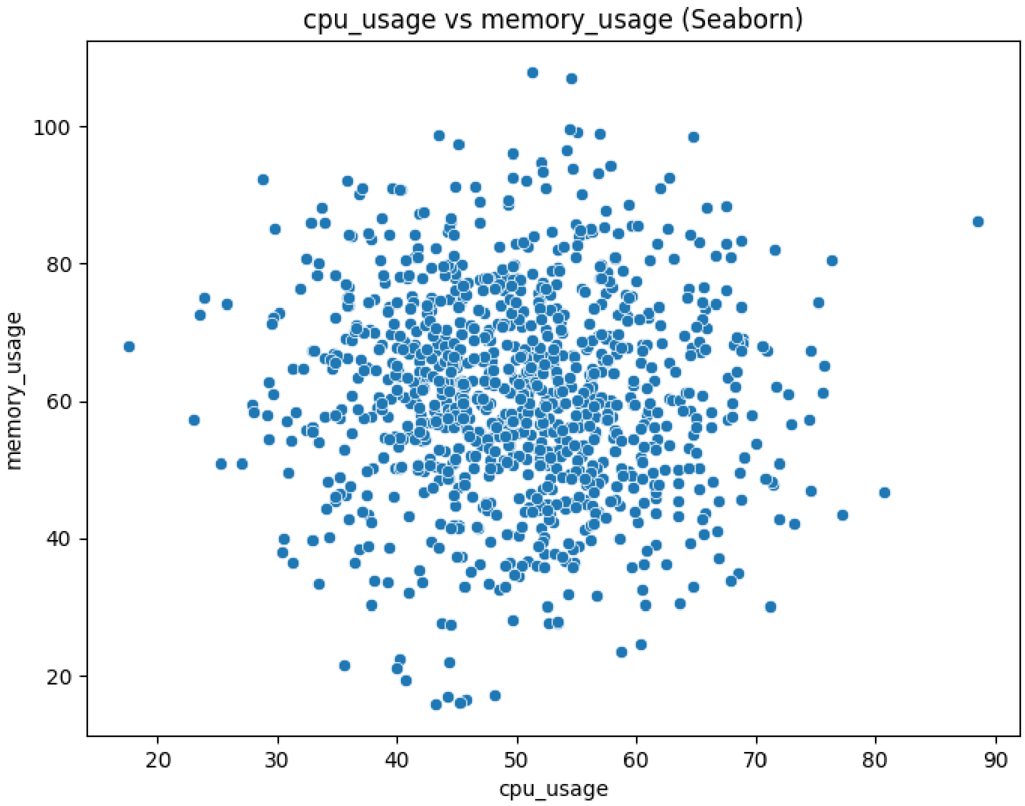
```
df['latency_anomaly'] = df['database_latency'].apply(lambda x: 1 if abs(x - df['database_latency'].mean()) > 2 * df['database_latency'].std() else 0)

# 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)
```

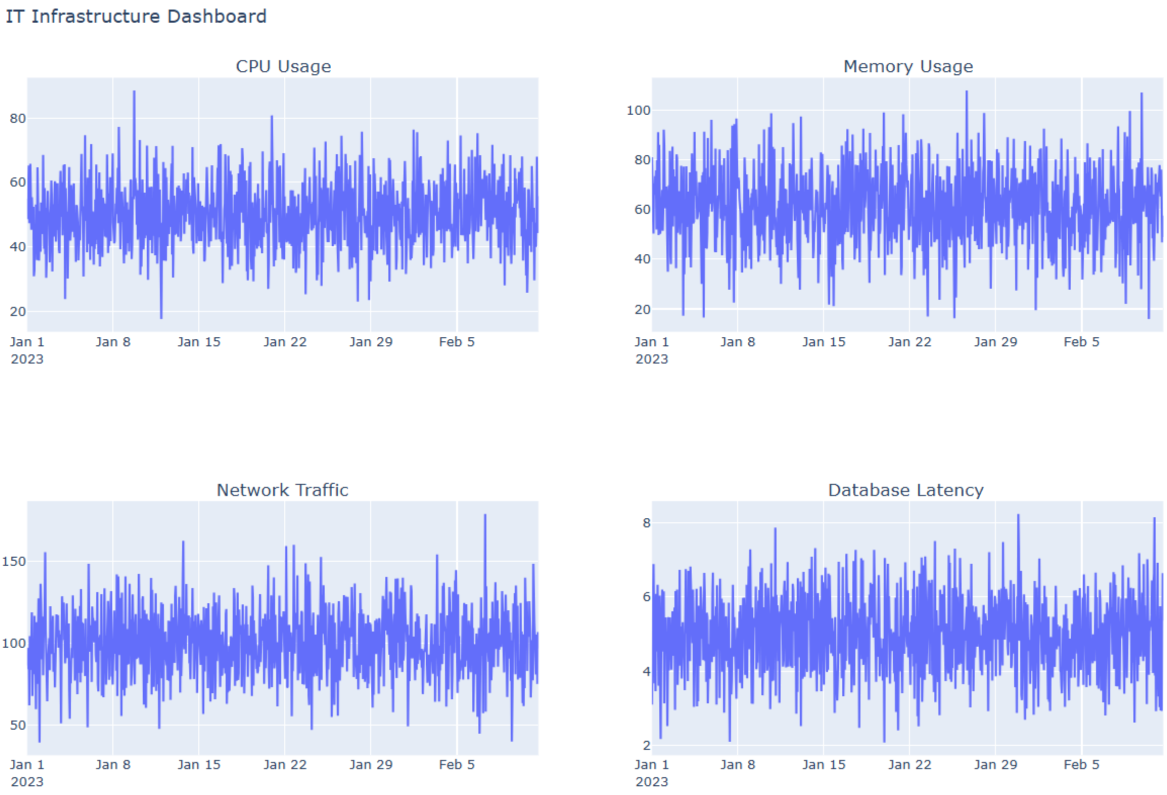
OUTPUT:

```
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')
```


cpu_usage Time Series with Anomalies (Matplotlib)
cpu_usage
Anomaly
90
80
70
60
50
40
30
20
2023-01-01
2023-01-08
2023-01-15
2023-01-22
2023-02-01
2023-02-08


network_traffic Distribution (Seaborn)
Count
100
80
60
40
20
0
40
60
80
100
120
140
160
180
network_traffic


cpu_usage vs memory_usage (Seaborn)
memory_usage
20
40
60
80
100
cpu_usage
20
30
40
50
60
70
80
90


IT Infrastructure Dashboard
CPU Usage
80
60
40
20
Memory Usage
100
80
60
40
20
Jan 1
2023
Jan 8
Jan 15
Jan 22
Jan 29
Feb 5
Network Traffic
150
100
50
Database Latency
8
6
4
2
Jan 1
2023
Jan 8
Jan 15
Jan 22
Jan 29
Feb 5

SS.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).
```