ORACLE

Deploying IoT Solutions with ThingsBoard, Kafka, and K3s Kubernetes on Oracle Roving Edge

Version 1.0
Copyright © 2025, Oracle and/or its affiliates
Public

Purpose statement

The purpose of this white paper is to present a scalable and resilient IoT solution designed to address the challenges of real-time data processing and visualization in edge environments. By leveraging K3s for lightweight container orchestration, Kafka for efficient data streaming, and ThingsBoard for comprehensive monitoring and visualization, this solution provides a robust framework for IoT applications. Deployed on Oracle Linux 8.10 within an Oracle Roving Edge device, this architecture ensures low-latency data handling and high availability, making it suitable for remote or resource-constrained scenarios.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.
This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.
This document may include some forward-looking content for illustrative purposes only. Some products and features discussed are indicative of the products and features of a prospective future launch in the United States only or elsewhere. Not all products and features discussed are currently offered for sale in the United States or elsewhere. Products and features of the actual offering may differ from those discussed in this document and may vary from country to country. Any timelines contained in this document are indicative only. Timelines and product features may depend on regulatory approvals or certification for individual products or features in the applicable country or region.

ORACLE

Table of contents

Introduction

The rapid growth of IoT applications, especially in environments requiring real-time data processing and visualization, has increased the demand for scalable and robust edge computing solutions. In scenarios where IoT devices generate large amounts of data, efficient data collection, processing, and visualization are crucial.

This white paper presents a scalable IoT solution leveraging K3s for lightweight container orchestration, Kafka for data streaming, and ThingsBoard for monitoring and visualization, all deployed on Oracle Linux 8.10 within an Oracle Roving Edge device.

By combining K3s, Kafka, and ThingsBoard, this solution provides a resilient, scalable architecture for temperature monitoring, which can be easily extended to other use cases such as humidity, pressure, or industrial sensor data.

The Oracle Roving Edge device serves as the compact, edge-based deployment platform, delivering low-latency processing and reliable data handling in remote or constrained environments.

Note: This content is provided for informational purposes and self-supported guidance only. Consultancy or other assistance related to the content is not covered under the Oracle Support contract or associated service requests. If you have questions or additional needs, then please reach out to your Oracle Sales contact directly.

Architecture Overview

The proposed IoT solution architecture leverages a combination of lightweight container orchestration, data streaming, and visualization components, specifically designed for deployment on an Oracle Roving Edge device running Oracle Linux 8.10. The architecture is structured to efficiently collect, process, and visualize real-time data from IoT sensors, with temperature and humidity monitoring as a primary use case.

Key Components:

- **IoT Devices (Sensors):** The edge environment is equipped with IoT sensors (e.g., temperature and humidity sensors) that continuously generate data. These sensors transmit data using lightweight protocols like MQTT or HTTP for efficient ingestion.
- **Data Ingestion (MQTT/HTTP):** The sensor data is ingested through MQTT or HTTP protocols, which are well-suited for low-latency, real-time communication. This ingestion layer acts as a bridge between the IoT devices and the data pipeline, ensuring reliable data flow.
- **Kafka Broker (Data Streaming & Pipeline):** Kafka acts as the core data streaming and processing engine. It receives data from the ingestion layer and stores it in a scalable, fault-tolerant log. Kafka ensures data continuity and consistency while managing high throughput for real-time analytics.
- **Kafka Connector/Producer:** Kafka Connector a custom Kafka producer is configured to push the ingested data to the processing cluster. It serves as a dedicated pipeline handler that efficiently transfers streaming data to the K3s cluster.
- **K3s Cluster (Container Platform):** The K3s cluster hosts both Kafka and ThingsBoard in separate pods, orchestrating the deployment efficiently. Kafka pods handle streaming data aggregation and message queuing. ThingsBoard pods process incoming data and store it in a database, preparing it for visualization, and K3s provides a lightweight, robust, and scalable container platform suitable for edge deployments.
- **ThingsBoard UI (Visualization & Monitoring):** ThingsBoard serves as the front-end interface for data visualization and monitoring. It displays real-time sensor data through dashboards and widgets, allowing users to monitor temperature and humidity trends. Users can customize dashboards, set alerts, and analyze data directly through the ThingsBoard interface.

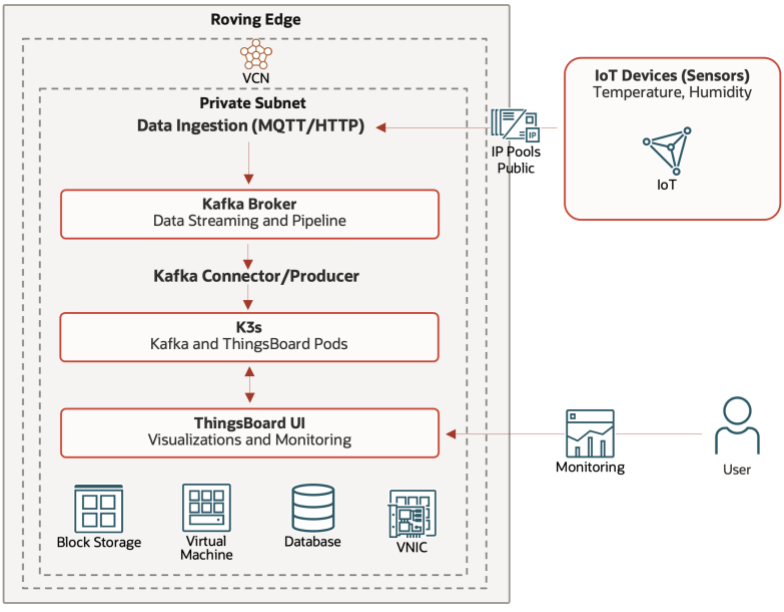

Figure 1. Architecture diagram of IoT Solution at the Edge with K3s, Kafka, and ThingsBoard on Oracle Roving Edge

Solution Deployment

This section outlines the detailed process to deploy the IoT monitoring solution on an Oracle Roving Edge device running Oracle Linux 8.10. The deployment leverages K3s for lightweight container orchestration, Kafka for real-time data streaming, and ThingsBoard for data visualization. The process starts by preparing the Oracle Roving Edge device, installing the necessary software components, and configuring the K3s environment. Next, Kafka is deployed within the cluster to handle data ingestion from IoT sensors, followed by integrating ThingsBoard for monitoring and dashboard creation. Finally, the solution is tested by simulating temperature and humidity data and visualizing it through the ThingsBoard UI. This step-by-step approach ensures a streamlined setup while maintaining scalability and reliability for real-time edge-based data monitoring.

Prerequisites

Before you begin, ensure you have the following:

- **Oracle Roving Edge Device with Oracle Linux 8.10:**
  - The device should be powered on and accessible (e.g., via SSH).
  - Ensure you have root or sudo privileges.
- **Network Connectivity:**
  - The device needs internet access for downloading packages and container images.
  - Ensure necessary ports are open (e.g., 22 for SSH, 6443 for K3s API, 80/443 for ThingsBoard UI).
- **System Resources:**
  - ThingsBoard can be resource intensive. Ensure your Roving Edge Device has sufficient CPU, RAM, and storage. A minimum of 4GB RAM and 2 CPU cores is recommended for a basic ThingsBoard setup, with more if you plan to handle significant data.
- **Basic Linux Administration Knowledge:**
  - Familiarity with the Linux command line, systemctl, firewalld, and vim/nano is assumed.

Prepare the Oracle Linux 8.10 System

First, update your system and disable firewalld and SELinux for K3s simplicity (for production, consider configuring them properly).

- **Update System Packages:**

```
sudo dnf update -y
sudo dnf upgrade -y
sudo dnf install -y curl vi git # Install useful tools
```

- **Disable firewalld (Recommended for K3s simplicity on edge devices):**

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```

Note: For a production environment, you might want to configure firewalld to allow specific K3s and ThingsBoard ports instead of disabling it entirely.

- **Disable SELinux (Recommended for K3s simplicity):** Edit the SELinux configuration file:

sudo vi /etc/selinux/config

```
Change SELINUX=enforcing to SELINUX=disabled.
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Process to be protected are selected for.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted

Reboot the system for SELinux changes to take effect:
sudo reboot
```

After rebooting, log back in.

Install K3s (Lightweight Kubernetes)

K3s is an excellent choice for edge devices due to its small footprint.

- **Install K3s:** The official K3s installation script is the easiest way.

  curl -sfL https://get.k3s.io | sh -

  This command will:

  - Install K3s as a systemd service.
  - Set up kubectl configuration in /etc/rancher/k3s/k3s.yaml.
  - Start the K3s server.

- **Verify K3s Installation:** Check the K3s service status:
- **sudo systemctl status k3s**

  You should see active (running).

  Verify kubectl access:

  sudo kubectl get nodes
  sudo kubectl get pods --all-namespaces

  You should see your node listed and various K3s system pods running.

- **Configure kubectl for your user (Optional but recommended):** To avoid using sudo with kubectl every time:

  mkdir -p ~/.kube
  sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
  sudo chown $(id -u):$(id -g) ~/.kube/config
  chmod 600 ~/.kube/config # Set appropriate permissions

Now you can run kubectl get nodes without sudo.

Prepare for ThingsBoard Deployment

You'll need Helm, the Kubernetes package manager, to deploy ThingsBoard.

- **Install Helm:**

  curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
  chmod 700 get_helm.sh
  ./get_helm.sh

- **Verify Helm installation:**

  helm version

- **Add ThingsBoard Helm Repository:**

  helm repo add thingsboard https://thingsboard.github.io/thingsboard-helm-charts
  helm repo update

Deploy ThingsBoard using Helm

ThingsBoard offers various deployment options. For a simple setup on an edge device, we'll use a basic configuration. ThingsBoard requires a database; we'll use PostgreSQL, which can be deployed as part of the Helm chart.

- **Create a Namespace for ThingsBoard:**

kubectl create namespace thingsboard

- **Configure ThingsBoard Values:** It's good practice to create a values.yaml file to customize your ThingsBoard deployment. This allows you to persist data, configure resources, and more.

  Create a file named thingsboard-values.yaml (e.g., using vim thingsboard-values.yaml):

```
# thingsboard-values.yaml
node:
kind: StatefulSet
  replicaCount: 1
  service:
    type: NodePort
    port: 8080
    nodePort: 30080 # NodePort for the main ThingsBoard node service
  resources:
```

```
    requests:
      cpu: 500m
      memory: 2Gi
    limits:
      cpu: 1000m
      memory: 3Gi
  persistence:
    enabled: true
    type: hostPath
    hostPath: /mnt/data/thingsboard-data
    size: 10Gi

webui:
  kind: Deployment
  replicaCount: 1
  service:
    type: NodePort
    port: 8084
    nodePort: 30084 # NodePort for the ThingsBoard Web UI
  env:
    - name: DISABLE_OAUTH2_UI
      value: "true" # Added to disable OAuth2 UI calls that cause 404

database:
  type: postgres
  host: thingsboard-postgresql
  port: 5432
  name: thingsboard
  username: postgres
  password: setplease # IMPORTANT: Change this to a strong, unique password!

postgresql:
  enabled: true
  auth:
    username: postgres
    password: setplease # IMPORTANT: Change this to a strong, unique password!
    database: thingsboard

  primary:
    # Use extraConfig to inject custom PostgreSQL settings
    extraConfig: |
      max_connections = 500
      shared_buffers = 512MB # Recommended: Adjust based on memory limits (e.g., 25% of memory)
    resources:
      requests:
        cpu: 1000m
        memory: 2Gi
      limits:
        cpu: 2000m
        memory: 4Gi

  persistence:
    enabled: true
    type: hostPath
    hostPath: /mnt/data/thingsboard-postgres-data
    size: 10Gi
```

Important:

- **password: setplease**: **CRITICAL!** Change setplease to a strong, unique password for both database.password and postgresql.auth.password.
- **nodePort values**: Ensure 30080 and 30084 (or whatever you choose) are not already in use on your Oracle Roving Edge Device.
- **hostPath**: Ensure these directories (/mnt/data/thingsboard-data and /mnt/data/thingsboard-postgres-data) exist on your Oracle Linux 8.10 system and have sufficient permissions and free space. Create them if they don't exist:

  ```
  sudo mkdir -p /mnt/data/thingsboard-data
  sudo mkdir -p /mnt/data/thingsboard-postgres-data
  sudo chmod -R 777 /mnt/data # Or more restrictive permissions if you understand them
  ```

- **Deploy ThingsBoard:** Now, deploy ThingsBoard using Helm, referencing your thingsboard-values.yaml file.

  ```
  helm upgrade thingsboard thingsboard/thingsboard \
    --namespace thingsboard \
    -f thingsboard-values.yaml
  ```

  This command will deploy ThingsBoard and its dependencies (like PostgreSQL) into the thingsboard namespace.

- **Monitor Deployment Progress:** You can watch the pods come up:

  ```
  kubectl get pods -n thingsboard -w
  ```

Wait until all pods (thingsboard, postgresql) are in the Running state. This might take several minutes depending on your device's performance and internet speed.

Access ThingsBoard UI via Public IP

Now that we've explicitly configured the webui service as NodePort, you should be able to access it using your Oracle Roving Edge Device's public IP address.

- **Get the NodePort for thingsboard-webui:**
- kubectl get svc -n thingsboard thingsboard-webui
    - Look for the PORT(S) column. It should now show 8084:30084/TCP (or whatever nodePort you chose for webui). The second number (30084 in this example) is the NodePort.
- **Find your Oracle Roving Edge Device's Public IP:** You'll need the public IP address of your Oracle Roving Edge Device. You can usually find this through the Oracle Cloud Infrastructure (OCI) console where your Roving Edge Device is managed, or by running a command like ip or hostname -I on the device itself (look for an IP that's reachable from outside your local network).
- **Login to ThingsBoard:** Open your web browser on your local machine or any team member's machine and navigate to: http://<YOUR_ROVING_EDGE_DEVICE_PUBLIC_IP>:<NODE_PORT>
- Replace <YOUR_ROVING_EDGE_DEVICE_PUBLIC_IP> with the actual public IP address of your Oracle Roving Edge Device, and <NODE_PORT> with the NodePort obtained from the kubectl get svc command (e.g., 30084).

The default ThingsBoard credentials are:

System Administrator: sysadmin@thingsboard.org / sysadmin
Tenant Administrator: tenant@thingsboard.org / tenant
Customer User: customer@thingsboard.org / customer

It is highly recommended to change these default passwords immediately after your first login.

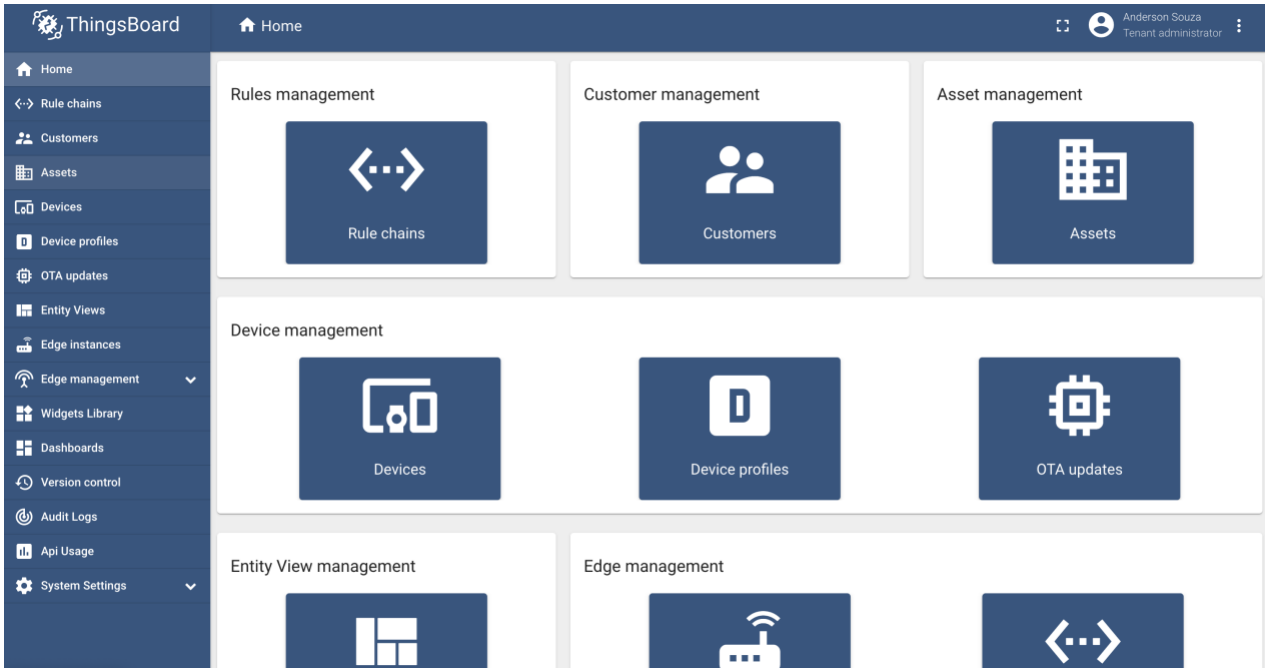

Figure 2. ThingsBoard Home Screen Overview.

Now that ThingsBoard and K3s are fully operational, we will deploy and configure Kafka (Broker), set up Kafka Producer and Consumer services, and create a ThingsBoard Dashboard to monitor simulated temperature and humidity metrics. In addition, we'll simulate telemetry data being sent directly to ThingsBoard via HTTP to represent external IoT devices.

Install and Run Kafka (Broker) via Helm

- **Add Bitnami Helm repo if not already added**

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
```

- **Create Kafka namespace**
  ```
  kubectl create namespace kafka
  ```

- **Install Kafka with Zookeeper (single-node)**
  ```
  helm install kafka bitnami/kafka \
    --namespace kafka \
    --set replicaCount=1 \
    --set zookeeper.enabled=true \
    --set listeners.client.protocol=PLAINTEXT \
    --set allowPlaintextListener=true \
    --set auth.enabled=false
  ```

- **Check if Kafka is up and running**

  ```
  kubectl get pods -n kafka

  NAME                 READY   STATUS    RESTARTS   AGE
  kafka-controller-0   1/1     Running   0          5m43s
  kafka-controller-1   1/1     Running   0          5m43s
  kafka-controller-2   1/1     Running   0          5m43s

  kubectl get svc -n kafka
  ```

Deploy Kafka Producer

The steps below show how to deploy Kafka Producer inside a K3s cluster to be utilized to send simulated temperature/humidity data to the existing Kafka Broker already configured in the system:

- Create a ConfigMap with the Python Kafka Producer script

```
cat <<EOF | kubectl apply -n thingsboard -f -
> apiVersion: v1
> kind: ConfigMap
> metadata:
>   name: kafka-producer-script
> data:
>   kafka_producer.py: |
>     from kafka import KafkaProducer
>     import json
>     import time
>     import random
>
>     producer = KafkaProducer(
>         bootstrap_servers='thingsboard-kafka:9092',
>         value_serializer=lambda v: json.dumps(v).encode('utf-8')
>     )
>
>     while True:
>         payload = {
>             "temperature": round(random.uniform(20, 30), 2),
>             "humidity": round(random.uniform(40, 60), 2)
>         }
>         producer.send('iot-data', payload)
>         print("Sent:", payload)
>         time.sleep(5)
> EOF
configmap/kafka-producer-script created
```

Create a Pod that runs the Kafka Producer

```
cat <<EOF | kubectl apply -n thingsboard -f -
> apiVersion: v1
> kind: Pod
> metadata:
>   name: kafka-producer
> spec:
>   restartPolicy: Never
>   containers:
>     - name: kafka-producer
>       image: python:3.9
>       command: ["python"]
>       args: ["/app/kafka_producer.py"]
>       volumeMounts:
>         - name: script
>           mountPath: /app
>   volumes:
>     - name: script
>       configMap:
>         name: kafka-producer-script
> EOF
pod/kafka-producer created
```

- Confirm that Kafka Producer pod is running:

kubectl get pods -n thingsboard

```
NAME                                     READY   STATUS    RESTARTS      AGE
kafka-producer                           1/1     Running   0             5m51s
thingsboard-coap-transport-0             1/1     Running   1 (51m ago)   121m
thingsboard-http-transport-0             1/1     Running   1 (51m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-4lz88  1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-8lwbm  1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-g2ppp  1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-jvrnq  1/1     Running   2 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-vzrqx  1/1     Running   3 (50m ago)   121m
thingsboard-kafka-0                      1/1     Running   3 (50m ago)   121m
thingsboard-mqtt-transport-0             1/1     Running   1 (51m ago)   121m
thingsboard-node-0                       1/1     Running   1 (51m ago)   121m
thingsboard-pg-pgpool-6cd4f945c6-wz8ld   1/1     Running   3 (51m ago)   121m
thingsboard-pg-postgresql-0              1/1     Running   1 (51m ago)   101m
thingsboard-redis-master-7c95c9fb56-fqn5m  1/1   Running   1 (51m ago)   121m
thingsboard-web-ui-5b5db6757b-ft785      1/1     Running   1 (51m ago)   121m
thingsboard-zookeeper-0                  1/1     Running   1 (51m ago)   121m
```

- Confirm that Kafka producer is running and sending the humidity and temperature metrics.

```
kubectl logs -f kafka-producer -n thingsboard
```

```
Collecting kafka-python
  Downloading kafka_python-2.2.10-py2.py3-none-any.whl (309 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 309.3/309.3 kB 3.2
MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.2.10
Sent: {'temperature': 27.43, 'humidity': 53.5}
Sent: {'temperature': 20.21, 'humidity': 45.8}
Sent: {'temperature': 24.91, 'humidity': 59.53}
Sent: {'temperature': 20.15, 'humidity': 58.18}
Sent: {'temperature': 26.3, 'humidity': 42.23}
Sent: {'temperature': 27.36, 'humidity': 43.61}
Sent: {'temperature': 22.76, 'humidity': 52.97}
Sent: {'temperature': 24.13, 'humidity': 42.68}
Sent: {'temperature': 20.51, 'humidity': 52.06}
Sent: {'temperature': 28.89, 'humidity': 41.37}
Sent: {'temperature': 21.13, 'humidity': 41.02}
```

Deploy Kafka Consumer

The steps below show how to deploy Kafka Consumer inside a K3s cluster to be utilized to receiver simulated temperature/humidity data to the existing Kafka Broker already configured in the system:

- Create a kafka_consumer.py script and make it executable.
- vi kafka_consumer.py (add the code listed below)
- chmod 755 kafka_consumer.py

```
from kafka import KafkaConsumer
import json
import requests

# Kafka settings
TOPIC_NAME = 'iot-data'
BOOTSTRAP_SERVERS = ['thingsboard-kafka:9092']

# ThingsBoard settings
THINGSBOARD_HOST = 'http://thingsboard-web-ui:8080'
DEVICE_TOKEN = 'YOUR_DEVICE_ACCESS_TOKEN'  # <-- Replace this with your actual token

# Start consumer
consumer = KafkaConsumer(
    TOPIC_NAME,
    bootstrap_servers=BOOTSTRAP_SERVERS,
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    group_id='iot-data-group',
    value_deserializer=lambda x: json.loads(x.decode('utf-8'))
)

print(f"[✓] Listening to Kafka topic '{TOPIC_NAME}'...")

# Stream and forward data to ThingsBoard
for message in consumer:
    data = message.value
    print(f"[✓] Received from Kafka: {data}")

    try:
        response = requests.post(
            f"{THINGSBOARD_HOST}/api/v1/{DEVICE_TOKEN}/telemetry",
            headers={'Content-Type': 'application/json'},
            data=json.dumps(data)
        )
        if response.status_code == 200:
            print("[✓] Successfully forwarded to ThingsBoard.")
        else:
            print(f"[!] ThingsBoard error: {response.status_code}, {response.text}")
    except Exception as e:
        print(f"[!] Exception while sending to ThingsBoard: {e}")
```

- Create the ConfigMap for Kubernetes to run the consumer inside of the K3s cluster, create the ConfigMap that holds the script with the code listed below.

NOTE: Replace YOUR_DEVICE_ACCESS_TOKEN above with your actual token (you can get this from the ThingsBoard UI under Devices → Your Device → Copy Token)

```
cat <<EOF | kubectl apply -n thingsboard -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: kafka-consumer-script
data:
  kafka_consumer.py: |
$(sed 's/^/    /' <<EOPYTHON
from kafka import KafkaConsumer
import json
import requests

TOPIC_NAME = 'iot-data'
BOOTSTRAP_SERVERS = ['thingsboard-kafka:9092']
THINGSBOARD_HOST = 'http://thingsboard-web-ui:8080'
DEVICE_TOKEN = 'FwQQXfKOS0anfilGZXXe'

consumer = KafkaConsumer(
    TOPIC_NAME,
    bootstrap_servers=BOOTSTRAP_SERVERS,
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    group_id='iot-data-group',
    value_deserializer=lambda x: json.loads(x.decode('utf-8'))
)

print(f"[✓] Listening to Kafka topic '{TOPIC_NAME}'...")

for message in consumer:
    data = message.value
    print(f"[✓] Received from Kafka: {data}")
    try:
        response = requests.post(
            f"{THINGSBOARD_HOST}/api/v1/{DEVICE_TOKEN}/telemetry",
            headers={'Content-Type': 'application/json'},
            data=json.dumps(data)
        )
        if response.status_code == 200:
            print("[✓] Successfully forwarded to ThingsBoard.")
        else:
            print(f"[!] ThingsBoard error: {response.status_code}, {response.text}")
    except Exception as e:
        print(f"[!] Exception while sending to ThingsBoard: {e}")
EOPYTHON
)
EOF
```

- Deploy the Kafka Consumer Pod

```
cat <<EOF | kubectl apply -n thingsboard -f -
apiVersion: v1
kind: Pod
metadata:
  name: kafka-consumer
spec:
  restartPolicy: Never
  containers:
    - name: kafka-consumer
      image: python:3.9
      command: ["/bin/sh"]
      args:
        - -c
        - |
          pip install kafka-python requests && \
          python /app/kafka_consumer.py
      volumeMounts:
        - name: script
          mountPath: /app
  volumes:
    - name: script
      configMap:
        name: kafka-consumer-script
EOF
```

- Confirm that Kafka Consumer pod is running:

  kubectl get pods -n thingsboard

```
NAME                                  READY   STATUS    RESTARTS      AGE
kafka-consumer                        1/1     Running   0             4m20s
kafka-producer                        1/1     Running   0             5m51s
thingsboard-coap-transport-0          1/1     Running   1 (51m ago)   121m
thingsboard-http-transport-0          1/1     Running   1 (51m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-4lz88   1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-8lwbm   1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-g2ppp   1/1     Running   3 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-jvrnq   1/1     Running   2 (50m ago)   121m
thingsboard-jsexecutor-786f8d9f4d-vzrqx   1/1     Running   3 (50m ago)   121m
thingsboard-kafka-0                   1/1     Running   3 (50m ago)   121m
thingsboard-mqtt-transport-0          1/1     Running   1 (51m ago)   121m
thingsboard-node-0                    1/1     Running   1 (51m ago)   121m
thingsboard-pg-pgpool-6cd4f945c6-wz8ld    1/1     Running   3 (51m ago)   121m
thingsboard-pg-postgresql-0           1/1     Running   1 (51m ago)   101m
thingsboard-redis-master-7c95c9fb56-fqn5m 1/1     Running   1 (51m ago)   121m
thingsboard-web-ui-5b5db6757b-ft785   1/1     Running   1 (51m ago)   121m
thingsboard-zookeeper-0               1/1     Running   1 (51m ago)   121m
```

- Confirm that Kafka consumer is running and receiving the humidity and temperature metrics.

kubectl logs -f kafka-consumer -n thingsboard

```
Collecting kafka-python
  Downloading kafka_python-2.2.10-py2.py3-none-any.whl (309 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 309.3/309.3 kB 3.3
MB/s eta 0:00:00
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl (64 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.9/64.9 kB 8.0
MB/s eta 0:00:00
Collecting urllib3<3,>=1.21.1
  Downloading urllib3-2.4.0-py3-none-any.whl (128 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 128.7/128.7 kB 12.4
MB/s eta 0:00:00
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.4.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(149 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.5/149.5 kB 15.2
MB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.10-py3-none-any.whl (70 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.4/70.4 kB 6.3
MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2025.4.26-py3-none-any.whl (159 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 159.6/159.6 kB 17.2
MB/s eta 0:00:00
Installing collected packages: kafka-python, urllib3, idna, charset-normalizer, certifi,
requests
Successfully installed certifi-2025.4.26 charset-normalizer-3.4.2 idna-3.10 kafka-python-2.2.10
requests-2.32.3 urllib3-2.4.0
[✓] Listening to Kafka topic 'iot-data'...
[✓] Received from Kafka: {'temperature': 23.23, 'humidity': 55.7}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 27.06, 'humidity': 43.67}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 24.44, 'humidity': 44.36}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 29.07, 'humidity': 59.95}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 24.16, 'humidity': 41.71}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 22.37, 'humidity': 44.08}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 24.6, 'humidity': 41.47}
[✓] Successfully forwarded to ThingsBoard.
[✓] Received from Kafka: {'temperature': 25.09, 'humidity': 43.98}
```

Publish Simulated Temperature/Humidity via HTTP

Create a new python script (**http_publisher.py**) with the code listed below, make the script executable, and replace <DEVICE_ACCESS_TOKEN> with your actual token from ThingsBoard.

```
import requests
import json
import time
import random

url = "http://YOUR_THINGSBOARD_IP:8080/api/v1/<DEVICE_ACCESS_TOKEN>/telemetry"

while True:
    payload = {
        "temperature": round(random.uniform(20, 30), 2),
        "humidity": round(random.uniform(40, 70), 2)
    }
    headers = {'Content-Type': 'application/json'}
    response = requests.post(url, headers=headers, data=json.dumps(payload))
    print(f"HTTP {response.status_code} -> {payload}")
    time.sleep(5)
```

Follow the steps below to access the ThingsBoard token:

- Login to your ThingsBoard instance as **System Admin** or **Tenant Admin**.
- On the left menu, go to **☰ Devices.**
- Select the device you're using or create a new one:
- Click "+" (plus) > **Add new device**.
- Enter a name like **Temperature/Humidity**, leave device type default, click Add.
- After the device is created:
- Click the device name to open it.
- Go to the "**Details**" tab.
- Scroll down or look for Access Token (on the right-side panel or near device credentials).
- Copy the **Access Token** — this is your DEVICE_ACCESS_TOKEN.

Create ThingsBoard Dashboard

- Navigate to: Devices → Your Device
- In the left panel, go to "Devices".
- Click on your device (the one associated with the access token used by your Kafka Consumer).
- Go to the "Latest Telemetry" tab, you should see temperature and humidity data arriving live.

Option 1. Add a Dashboard from Scratch

- Left Menu → Dashboards → click "+" (Add new dashboard)
- Name it: e.g., IoT Monitoring
- Open the dashboard → click "Edit" (pencil) → click "+" to add a widget

Option 2. Add Widget from Device

- While inside the device details page, click "Charts" → "+".
- Select widget type (e.g., “Timeseries Chart”)
- Select telemetry keys: temperature, humidity
- Add to existing dashboard or create new one.

Verify

Once added, the dashboard should begin plotting your incoming sensor values in real-time

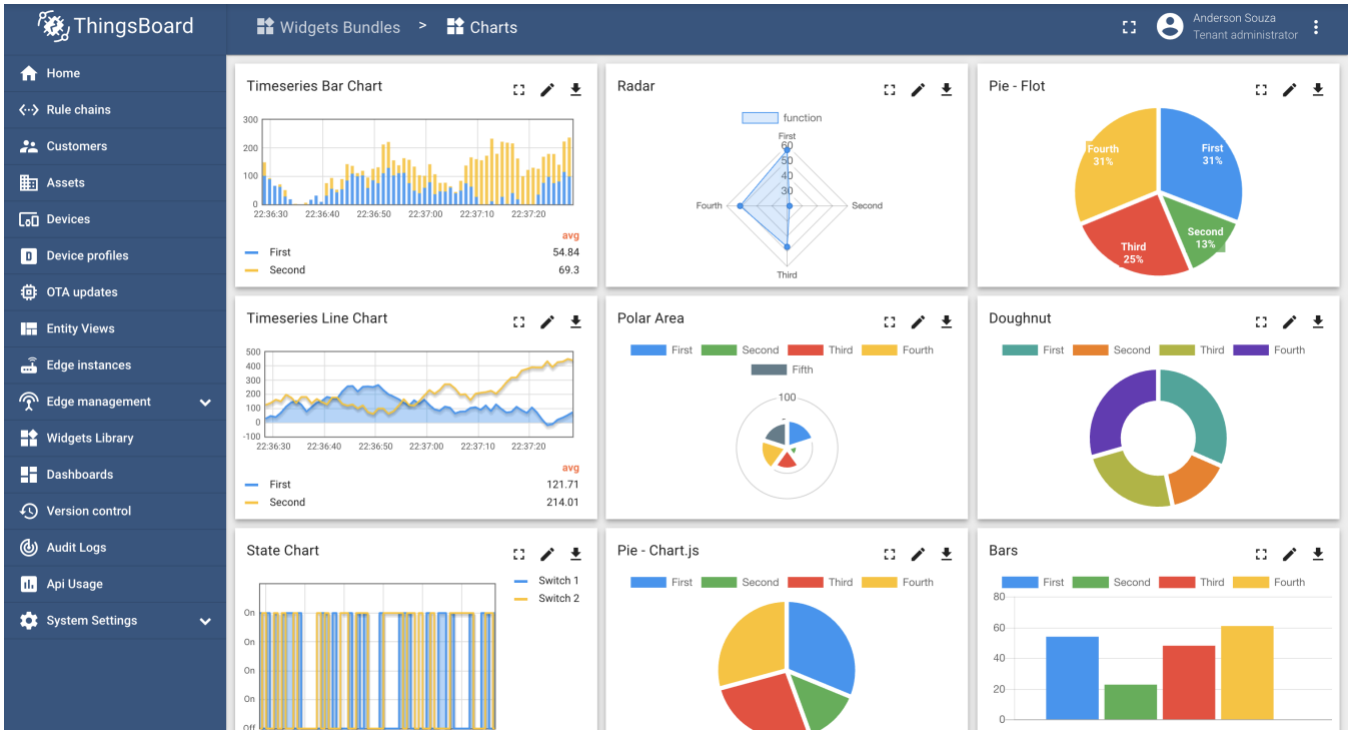

Figure 2. ThingsBoard dashboards for IoT solutions on Roving Edge.

ORACLE

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Copyright © 2025, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Some regulatory certifications or registrations to products or services referenced on this website are held by Cerner Corporation. Cerner Corporation is a wholly-owned subsidiary of Oracle. Cerner Corporation is an ONC-certified health IT developer and a registered medical device manufacturer in the United States and other jurisdictions worldwide.

This document may include some forward-looking content for illustrative purposes only. Some products and features discussed are indicative of the products and features of a prospective future launch in the United States only or elsewhere. Not all products and features discussed are currently offered for sale in the United States or elsewhere. Products and features of the actual offering may differ from those discussed in this document and may vary from country to country. Any timelines contained in this document are indicative only. Timelines and product features may depend on regulatory approvals or certification for individual products or features in the applicable country or region.

Author: Anderson Souza