Ditto Framework (E/19/166 - Jayathunga W.W.K.)

Eclipse Ditto is a powerful open-source framework designed to facilitate the creation, management, and interaction of **digital twins**. In the context of this air quality management system, Ditto plays a central role as the backbone of digital twin orchestration, enabling real-time monitoring, control, and integration of IoT devices such as air quality sensors and purifiers

Step 1: Setup Prerequisites

Install Required Tools:

- **Docker and Docker Compose**: For running Ditto and related services.
    - Download and install Docker from [Docker: Accelerated Container Application Development]
    - Verify installation:

```
D:\AAApera\Sem6\CO326\project\ditto>docker --version
Docker version 26.1.1, build 4cf5afa
```

    - Install Docker Compose (included in newer Docker versions)
    - Verify installation:

```
D:\AAApera\Sem6\CO326\project\ditto>docker-compose --version
Docker Compose version v2.23.3
```

- **Java Development Kit (JDK)**: Ensure Java 11 or higher is installed.

```
sudo apt update
sudo apt install openjdk-11-jdk
java -version
```

- Node.js
    - Install Node.js and npm

```
curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt install -y nodejs
node -v
npm -v
```

- Other Dependencies:
  - Install curl:

```
sudo apt install curl
```

Step 2: Configure Docker Compose for Ditto and Dependencies

Create a docker-compose.yml file
[e19-co326-air-quality-management-system/Code/Ditto/setup/docker-compose.yaml at main · cepdnaclk/e19-co326-air-quality-management-system]

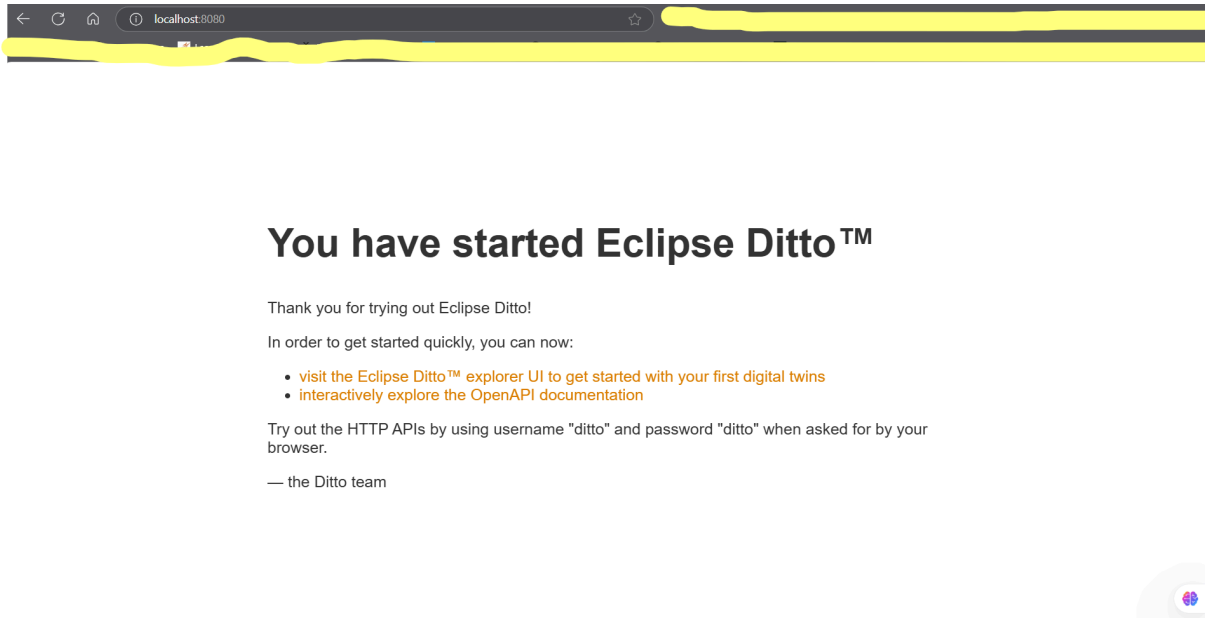
Start Docker Services:

```
PS C:\Users\Wishula> docker-compose up -d
```

Step 3: Configure Eclipse Ditto

Access Ditto API

Verify the Ditto Service is running by visiting:



Create a Twin:
[e19-co326-air-quality-management-system/Code/Ditto/air.py at main · cepdnaclk/e19-co326-air-quality-management-system]

Send the request using curl:

```
PS C:\Users\Wishula> curl -X PUT -H "Content-Type: application/json" -d @air
.json http://localhost:8080/api/2/things/namespace:air_quality_management
```

Step 4: Define Ditto Thing Types and Twins

[e19-co326-air-quality-management-system/Code/Ditto/setup/thing_definitions at main · cepdnaclk/e19-co326-air-quality-management-system]

Use the Ditto API to create the Things:

```
D:\AAApera\Sem6\CO326\project\ditto>curl -X PUT -H "Content-Type: applicatio
n/json" -d @air.json http://localhost:8080/api/2/things/namespace:air_qualit
y_management
```

```
PS C:\Users\Wishula> curl -X PUT -H "Content-Type: application/json" -d @thi
ng-definitions/air-quality-sensor.json http://localhost:8080/api/2/things/na
mespace:air-quality-sensor
```

```
PS C:\Users\Wishula> curl -X PUT -H "Content-Type: application/json" -d @thi
ng-definitions/purifier-device.json http://localhost:8080/api/2/things/names
pace:purifier-device
```

e19-co326-air-quality-management-system/Code/Ditto/setup/policies at main · cepdnaclk/e19-co326-air-quality-management-system

Use curl to create policies in Ditto

```
PS C:\Users\Wishula> curl -X PUT -H "Content-Type: application/json" -d @pol
icies/sensor-policy.json http://localhost:8080/api/2/policies/namespace:sens
or-policy
```

```
PS C:\Users\Wishula> curl -X PUT -H "Content-Type: application/json" -d @pol
icies/purifier-policy.json http://localhost:8080/api/2/policies/namespace:pu
rifier-policy
```

Step 5: Extend Ditto with Custom API

Create ditto-extended-api/ Service:
e19-co326-air-quality-management-system/Code/Ditto/extended_api at main · cepdnaclk/e19-co326-air-quality-management-system

- Use Node.js:

```
D:\AAApera\Sem6\CO326\project\ditto>npm init -y
```

```
D:\AAApera\Sem6\CO326\project\ditto>npm install express axios
```

- Create index.js:
  e19-co326-air-quality-management-system/Code/Ditto/extended_api/src/index.js at main · cepdnaclk/e19-co326-air-quality-management-system

- Run the service:

```
D:\AAApera\Sem6\CO326\project\ditto>node src/index.js
```

Step 6: Integrate Kafka & InfluxDB

Configure Kafka Topics:

Create a Kafka topic for sensor data:

```
PS C:\Users\Wishula> kafka-topics.sh --create --topic sensor-data --bootstra
p-server localhost:9092 --replication-factor 1 --partitions 1
```

Connect Ditto to Kafka:

Update Ditto's configuration to publish events to Kafka. Use its management API:

```
PS C:\Users\Wishula> curl -X POST -H "Content-Type: application/json" -d '{"
name": "kafka-connection", "type": "kafka", "uri": "tcp://localhost:9092", "
topics": ["sensor-data"]}' http://localhost:8080/connections
```

Step 7: Test the System

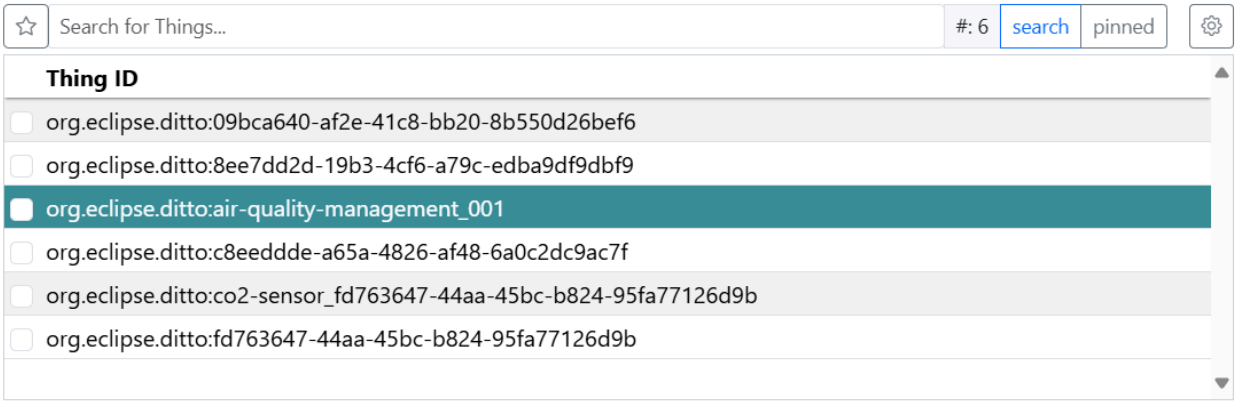
1. Start the API:

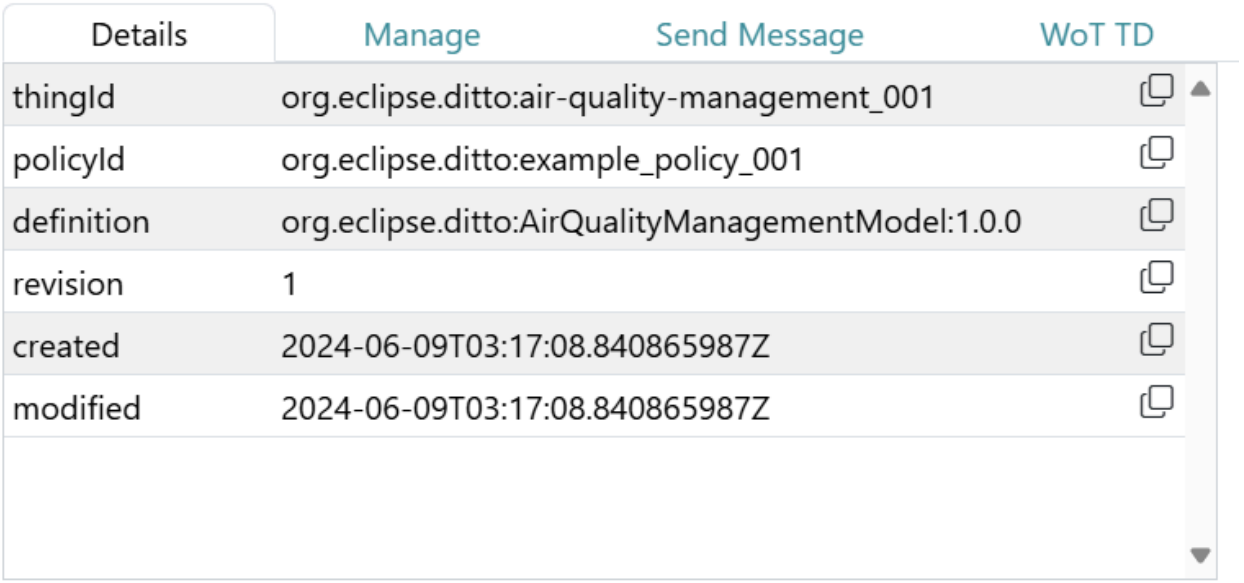
```
PS C:\Users\Wishula> node src/index.js
```

2. Create Things & Policies via API:
    - Use Postman or curl to send requests to:
        - POST /thing/:id for Things
        - POST /policy/:id for Policies

3. Verify in Eclipse Ditto:
    - Check `http://localhost:8080/api/2/things/namespace:air-quality-sensor` for the sensor Thing.
    - Check `http://localhost:8080/api/2/policies/namespace:sensor-policy` for the policy.

ScreenShots of the final Eclipse Ditto User Interface


Search for Things...
#: 6
search
pinned
Thing ID
org.eclipse.ditto:09bca640-af2e-41c8-bb20-8b550d26bef6
org.eclipse.ditto:8ee7dd2d-19b3-4cf6-a79c-edba9df9dbf9
org.eclipse.ditto:air-quality-management_001
org.eclipse.ditto:c8eeddde-a65a-4826-af48-6a0c2dc9ac7f
org.eclipse.ditto:co2-sensor_fd763647-44aa-45bc-b824-95fa77126d9b
org.eclipse.ditto:fd763647-44aa-45bc-b824-95fa77126d9b

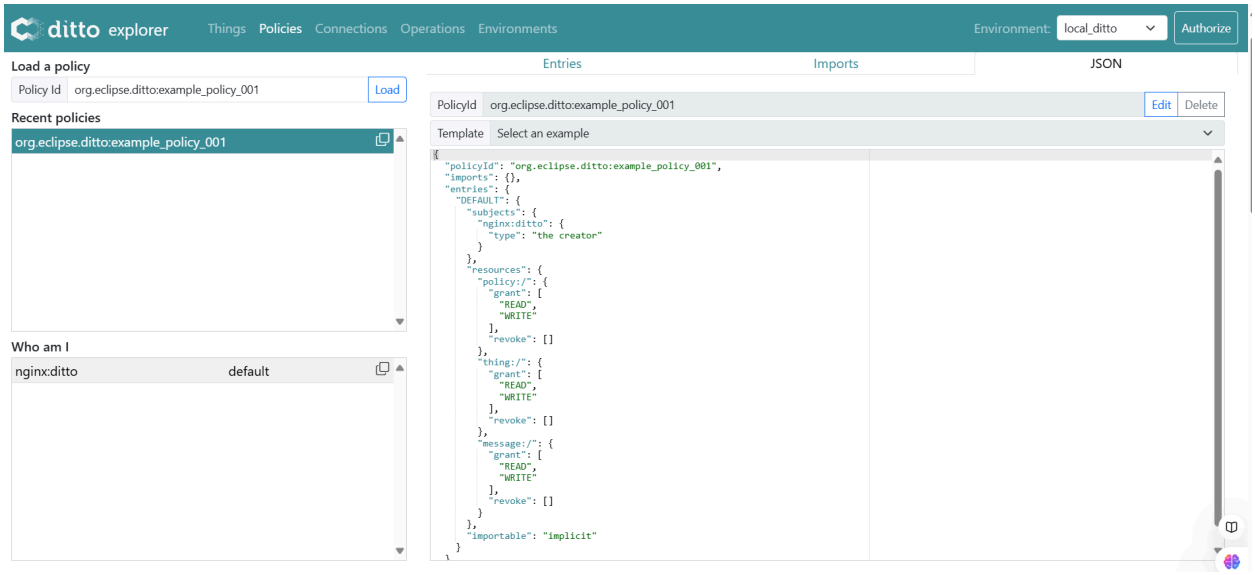

Details
Manage
Send Message
WoT TD
thingId
org.eclipse.ditto:air-quality-management_001
policyId
org.eclipse.ditto:example_policy_001
definition
org.eclipse.ditto:AirQualityManagementModel:1.0.0
revision
1
created
2024-06-09T03:17:08.840865987Z
modified
2024-06-09T03:17:08.840865987Z

Attributes 1

location	{"latitude":44.673856,"longitude":8.261719}

Features 10

- co2_sensor
- oxygen_sensor
- pm25_sensor
- pm10_sensor
- temperature_sensor
- humidity_sensor
- voc_sensor
- no2_sensor
- so2_sensor
- co_sensor



Experience with OpenTwin

Attempt on Windows 11

While exploring the capabilities of OpenTwin for managing digital twins in our system, I attempted to run the framework on a Windows 11 machine. Unfortunately, this attempt was not successful due to the inherent limitations of Windows as a non-UNIX-based system. OpenTwin relies on specific UNIX system calls and environment setups that are not natively supported by Windows.

Key Issues Encountered:

- **Dependency Compatibility**: Several scripts and dependencies within OpenTwin are optimized for UNIX-like environments.
- **Script Failures**: Core functionalities failed during execution, particularly those dependent on shell scripting.
- **Environment Mismatch**: Lack of certain UNIX tools and environment configurations that are critical to OpenTwin's operations.

References to Scripts

The scripts that encountered issues include:

- [e19-co326-air-quality-management-system/Code/Ditto/setup/myuser@WISHULAJAYATHUNGA mntdAAAperaSem6CO326projectsetup.txt at main · cepdnaclk/e19-co326-air-quality-management-system]

- e19-co326-air-quality-management-system/Code/Ditto/setup/myuser@WISHULAJAYATHUNGA mntdAAAperaSem6CO326projectsetup1.txt at main · cepdnaclk/e19-co326-air-quality-management-system
- e19-co326-air-quality-management-system/Code/Ditto/setup/myuser@WISHULAJAYATHUNGA mntdAAAperaSem6CO326projectsetup2.txt at main · cepdnaclk/e19-co326-air-quality-management-system

Workaround and Recommendations:

To successfully use OpenTwin, I recommend the following:

1. **Use a UNIX-based Operating System**: Preferably Ubuntu or any other Linux distribution.
2. **Ensure Proper Configuration**: Follow OpenTwin's official documentation for setting up dependencies and environments.