Dynatrace Full-Stack Monitoring Project – Step-by-Step Documentation

Monitoring a Node.js Application on WSL Ubuntu using Dynatrace OneAgent

1. Project Overview

This project demonstrates how to deploy and monitor a Node.js application using Dynatrace OneAgent on a local WSL Ubuntu 22.04 environment. The objective is to implement end-to-end observability on both the host and the application.

2. Technologies Used

• Dynatrace SaaS (OneAgent Full-Stack Monitoring, Dashboards, Davis AI anomaly detection)

• Node.js 20 and Express.js

• WSL Ubuntu 22.04 running on Windows 11

• Web browser and curl for generating HTTP traffic

3. High-Level Architecture

• Windows system runs WSL Ubuntu 22.04.

• Inside WSL, a Node.js/Express web application listens on port 3000.

• Dynatrace OneAgent is installed on the Ubuntu environment.

• OneAgent automatically detects the host, processes and Node.js service, and sends telemetry to the Dynatrace SaaS environment via HTTPS.

• Dashboards in Dynatrace visualize host metrics, service performance, and errors.

4. Step 1 – Prepare WSL Ubuntu

1) Open Windows Terminal or PowerShell.

2) Start Ubuntu:

```
wsl -d Ubuntu-22.04
```

3) Update packages:

```
sudo apt update && sudo apt upgrade -y
```

4) Install basic tools:

```
sudo apt install -y curl git
```

5. Step 2 – Install Node.js and Initialize the Project

1) Install Node.js (NodeSource repo):

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -

sudo apt install -y nodejs
```

2) Verify installation:

```
node -v

npm -v
```

3) Create the project folder:

```
mkdir dynatrace-node

cd dynatrace-node
```

4) Initialize the Node.js project:

```
npm init -y
```

6. Step 3 – Install Express and Create the Web Application

1) Install Express:

```
npm install express
```

2) Create the application file server.js:

```
nano server.js
```

3) Insert the following code into server.js:

```
const express = require("express");
const app = express();
```

```
  const port = 3000;

  app.get("/", (req, res) => {
    res.send("Hello from Node.js app monitored by Dynatrace on WSL!");
  });

  app.get("/slow", (req, res) => {
    setTimeout(() => {
      res.send("This is a slow endpoint to test Dynatrace response time.");
    }, 2000);
  });

  app.get("/error", (req, res) => {
    throw new Error("Simulated error for Dynatrace demo");
  });

  app.listen(port, () => {
    console.log(`App listening on http://localhost:${port}`);
  });
```

4) Save and exit the editor.

5) Run the application:

```
  node server.js
```

6) Test in your browser from Windows:

```
  http://localhost:3000

  http://localhost:3000/slow

  http://localhost:3000/error
```

7. Step 4 – Create and Configure the Dynatrace Environment

1) Sign up or log in to Dynatrace SaaS.

2) Create an environment if not already created.

3) From the Dynatrace menu, choose “Deploy Dynatrace” then select “Linux”.

4) Copy the OneAgent installation command generated by Dynatrace for Linux (64-bit).

8. Step 5 – Install Dynatrace OneAgent on WSL Ubuntu

1) In the Ubuntu terminal (inside WSL), run the command provided by Dynatrace, similar to:

```
  wget -O Dynatrace-OneAgent.sh "https://<your-env>.live.dynatrace.com/api/v1/deployment/installer/agent/unix/default/latest?arch=x86&…"
```

```
  sudo /bin/sh Dynatrace-OneAgent.sh
```

2) Wait for the installation to complete successfully.

3) Verify installation by checking OneAgent files:

```
  ls /opt/dynatrace/oneagent
```

4) Check that OneAgent processes are running:

```
  ps aux | grep -i oneagent
```

Note: On WSL, systemd may not fully manage services, so systemctl oneagent status might not be available. The presence of OneAgent processes and the host appearing in Dynatrace confirm successful installation.

9. Step 6 – Restart the Node.js Application and Generate Traffic

1) Restart the Node.js app so that it is monitored by OneAgent:

```
  cd ~/dynatrace-node

  node server.js
```

2) Generate traffic from your browser:

```
  http://localhost:3000

  http://localhost:3000/slow

  http://localhost:3000/error
```

3) Optionally, generate more load from the terminal:

```
  for i in {1..50}; do curl -s -o /dev/null http://localhost:3000/slow; done
```

10. Step 7 – Explore Services and Host in Dynatrace

1) In Dynatrace, open “Hosts” and locate your host (for example: Test.localdomain).

2) Check host metrics:

- CPU usage
- Memory usage
- Network traffic

3) Go to “Applications & Microservices” → “Services”.

4) Locate the Node.js service (for example: server.js or dynatrace-node).

5) Open the service and review:

- Requests per minute
- Response time
- Error rate
- Endpoints: /, /slow, /error
- Service flow and traces (PurePaths).

11. Step 8 – Create a Dynatrace Monitoring Dashboard

1) In Dynatrace, go to “Dashboards”.

2) Click “Create dashboard”.

3) Name it: WSL Node.js Dynatrace Monitoring – Faruk.

4) Add service tiles (Node.js service):

- Service response time.
- Requests per minute (RPS).
- Error rate (%).
- Top slowest requests.

5) Add host tiles (host Faruk.localdomain):

- CPU usage.
- Memory usage.
- Network traffic.

6) Arrange tiles in a clear 2-column layout, for example:

Left column – Service metrics (response time, RPS, error rate, slowest requests).

Right column – Host metrics (CPU, memory, network).

7) Save the dashboard. This dashboard can be used for screenshots in your portfolio or presentations.

12. Step 9 – Configure Anomaly Detection and Alerts (Optional)

1) Go to Settings → Anomaly detection → Services.

2) Select the Node.js service.

3) Enable custom thresholds, for example:

• Alert if error rate exceeds 5%.

• Alert if response time is above 2 seconds for a sustained period.

• Alert if the service becomes unavailable.

4) Generate errors to trigger alerts:

```
for i in {1..20}; do curl http://localhost:3000/error || true; done
```

5) Observe the generated "Problem" in Dynatrace:

• Impacted service.

• Root cause analysis.

• Timeline and evidence.

• Automatic Davis AI analysis.

13. Step 10 – Summary

This project shows a complete end-to-end observability chain for a Node.js application running on WSL:

• Preparation of a Linux environment (WSL Ubuntu).
• Deployment of a Node.js/Express web application with normal, slow and error endpoints.
• Installation and configuration of Dynatrace OneAgent.
• Automatic discovery of host, processes and services.
• Creation of a custom dashboard combining host and service metrics.

- Optional anomaly detection rules to simulate real production incidents.