Continuous testing of oneM2M IoT products with Docker and Jenkins
Presented by Bogdan Stanca-Kaposta (Spirent)
Dale Seed, Bob Flynn (InterDigital)
What is oneM2M?
oneM2M Partnership Project

Over 200 member organizations in oneM2M

www.oneM2M.org

All document are publicly available
oneM2M Service Layer

- A software “framework” that sits between IoT apps and underlying network protocol and communication stack
- Supports a common set of horizontal services that IoT devices and apps across different industry verticals commonly require
- Can be deployed on devices, gateways and servers, highly distributed and scalable
oneM2M Breaks Down the Silos

Smart Emergency Services
Smart Transportation
Smart Infrastructure

Horizontal Information Flow

Smart Emergency Services
Smart Transportation
Smart Infrastructure

Service Layer

Vertical Information Flow

User Conference on
Advanced Automated Testing
oneM2M is Distributive and Scalable
The Problem

• Assure the quality of the development process of oneM2M components and their tests

• All components were under development
  • TTCN-3 Test Suite
  • Test Adaptation
  • System Under Test (SUT)

• Multiple configurations possible
oneM2M

- Provides interoperability for Machine-to-Machine and IoT technologies
- TTCN-3 Test Cases under development
  - 700+ Test Cases
  - 4 Bindings (HTTP, MQTT, CoAP, WebSockets)
  - 3 Encodings (JSON, XML, CBOR)
  - 3 Standard Releases (4\textsuperscript{th} release is being currently developed)
  - 7 Profiles
SUT issues

• SUT still under development
  • Software SUT regularly updated

• How to make sure that
  • The developers have all the same SUT configuration
  • All machines run the same OS version and libraries
  • The build servers can handle multiple SUTs
Docker

- Containers are portable
- Uses 50% less resources comparing to VMs
- Ideal for
  - Micro services
  - Continuous integration and continuous delivery
Example Dockerfile

- Simple configuration
- Reusable

```dockerfile
# Use official node as base image.
FROM node:carbon

# Install the latest app
WORKDIR /root/app
COPY app/* ./

RUN npm install

# Expose the ports used by application
# 3000 HTTP
EXPOSE 3000

ENTRYPOINT [ "npm", "start" ]
CMD [ "127.0.0.1:4141" ]
```
Docker features match our needs

- Portable
- Hide the configuration from users
- Fast container start/restart
- Consistent SUT configuration, host OS and libraries
- Multiple instances possible
Manual Test Execution
Jenkins

- Automation server
  - Used for Continuous Integration and Continuous Delivery
  - Distributed
  - Extensible
  - Huge community
Automated Test Execution using Jenkins

- Execute regression tests
- Start multiple configurations in parallel
- Analyze the test results
Automation is great but how to analyze this?

700 tests *

12 configurations *

3 Releases

<table>
<thead>
<tr>
<th>TC</th>
<th>HTTP JSON</th>
<th>CoAP JSON</th>
<th>MQTT JSON</th>
<th>WS JSON</th>
<th>HTTP XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC_1</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
<td>✔</td>
</tr>
<tr>
<td>TC_3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_4</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_5</td>
<td>✔</td>
<td>×</td>
<td>✔</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>TC_6</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_7</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
</tr>
</tbody>
</table>
Interpreting the results

<table>
<thead>
<tr>
<th>TC</th>
<th>HTTP</th>
<th>CoAP</th>
<th>MQTT</th>
<th>WS</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC_1</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
<td>✔</td>
</tr>
<tr>
<td>TC_3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_4</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_5</td>
<td>✔</td>
<td>×</td>
<td>✔</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>TC_6</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TC_7</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>×</td>
</tr>
</tbody>
</table>

What do we identify here?

Special problem spots or even single failing tests are identified
Interpreting the results

- Where is the issue?
  - Configuration
  - Adaptation
  - DUT

<table>
<thead>
<tr>
<th>TC</th>
<th>HTTP JSON</th>
<th>CoAP JSON</th>
<th>MQTT JSON</th>
<th>WS JSON</th>
<th>HTTP XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC_1</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_2</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_3</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_4</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_5</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_6</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_7</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Interpreting the results

- Where is the issue?
- Test case

<table>
<thead>
<tr>
<th>TC</th>
<th>HTTP JSON</th>
<th>CoAP JSON</th>
<th>MQTT JSON</th>
<th>WS JSON</th>
<th>HTTP XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC_1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TC_2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TC_3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TC_4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TC_5</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>TC_6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TC_7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Problems solved?

- Reproduceable setup on all machines
- Faster execution due automated parallel execution
- Visual analysis of the results highlight hotspots
- Faster feedback to the development teams
Future work

• Stress tests
• Testing the oneM2M application in the cloud
• Complex scenarios
Thank you!

Questions?