ETSI TR 103 529 V1.1.1 (2018-08)

SmartM2M;
IoT over Cloud back-ends:
A Proof of Concept

Reference

DTR/SmartM2M-103529

Keywords

cloud, IoT, open source, proof of concept, virtualisation

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

List of figures

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

In addition to interoperability and security that are two recognized key enablers to the development of large IoT systems, a new one is emerging as another key condition of success: virtualization. The deployment of IoT systems will occur not just within closed and secure administrative domains but also over architectures that support the dynamic usage of resources that are provided by virtualization techniques over cloud back-ends.

This new challenge for IoT requires that the elements of an IoT system can work in a fully interoperable, secure and dynamically configurable manner with other elements (devices, gateways, storage, etc.) that are deployed in different operational and contractual conditions. To this extent, the current architectures of IoT will have to be aligned with those that support the deployment of cloud-based systems (private, public, etc.).

Moreover, these architectures will have to support very diverse and often stringent non-functional requirements such as scalability, reliability, fault tolerance, massive data, security. This will require very flexible architectures for the elements (e.g. the application servers) that will support the virtualized IoT services, as well as very efficient and highly modular implementations that will make a massive usage of Open Source components.

These architectures and these implementations form a new approach to IoT systems and the solutions that this STF will investigate will also have to be validated: to this extent, a Proof-of-Concept implementation involving a massive number of virtualized elements will be made.

The present document is one of three Technical Reports addressing this issue:

- ETSI TR 103 527 [i.1]: "SmartM2M; Virtualized IoT Architectures with Cloud Back-ends".
- ETSI TR 103 528 [i.2]: "Landscape for open source and standards for cloud native software for a Virtualized IoT service layer".
- ETSI TR 103 529 (the present document): "IoT over Cloud back-ends: A Proof of Concept".

1 Scope

The present document:

- Recalls the main elements of the Proof-of-Concept (PoC) in support of IoT Virtualization: use case description, high-level architecture of the application developed, main technical choices.
- Presents the main implementation choices.
- Outlines the lessons learned and the possible impact of future IoT Virtualization implementations.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 527: "SmartM2M; Virtualized IoT Architectures with Cloud Back-ends".

[i.2] ETSI TR 103 528: "SmartM2M; Landscape for open source and standards for cloud native software applicable for a Virtualized IoT service layer".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Open Source Software (OSS): computer software that is available in source code form

NOTE: The source code and certain other rights normally reserved for copyright holders are provided under an open-source license that permits users to study, change, improve and at times also to distribute the software.

source code: any collection of computer instructions written using some human-readable computer language, usually as text

standard: output from an SSO

Standards Setting Organization (SSO): any entity whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting or otherwise maintaining standards that address the interests of a wide base of users outside the standards development organization

NOTE: In the present document, SSO is used equally for both Standards Setting Organization or Standards Developing Organizations (SDO).

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CPU	Central Processing Unit
CSE	Common Service Entity (in oneM2M)
CSP	Cloud Service Provider
EOF	End Of File
HLA	High-Level Architecture
HPA	Horizontal Pod Autoscaler
IaaS	Infrastructure as a Service
NFS	Network File System
OSS	Open Source Software
PaaS	Platform as a Service
PoC	Proof of Concept
RAM	Random Access Memory
RC	Replication Controller
REST	REpresentational State Transfer
SaaS	Software as a Service
SDO	Standards Development Organization
SSH	Secure SHell
SSO	Standards Setting Organization
STF	Specialist Task Force
VM	Virtual Machine
YAML	YAML Ain't Markup Language

4 Virtualization of IoT: A Proof-of-Concept

4.1 Virtualized IoT and Cloud-Native Computing

The IoT industry has started to understand and evaluate the potential benefits of Cloud-Native Computing for the fast, effective and future-safe development of IoT systems combining the strengths of both IoT and Cloud industries in a new value proposition. The expectation of Cloud-Native applications is to benefit from offerings from Cloud Service Providers (CSP) that may cover all or part of the layers of Virtualized application, via Infrastructure as a Service (IaaS), Platform as a Service (PaaS) or Software as a Service (SaaS).

In the case of IoT applications, the trade-off between what is delegated to the Cloud Service Provider and what is kept in the hands of the application developers may vary depending a large number of potential factors and will finally materialize into different architecture, design and implementation choices.

The approach of Cloud-Native Computing is now widely supported by a large set of technologies embedded in Cloud-Native Infrastructures in support of Cloud-Native Applications. These technologies are now very diverse, technology-ready (as shown in the landscape of Open Source components described in ETSI TR 103 528 [i.2]) and support all the layers of a Micro-Service Architecture (such as the one described in ETSI TR 103 527 [i.1]).

4.2 The rationale for a Proof-of-Concept

As pointed out in ETSI TR 103 527 [i.1], "there is probably a large number of [IoT] Use Cases for which a "traditional" (i.e. non-virtualized) approach can and will apply. However, the introduction of IoT Virtualization is expected to make some Use Cases more effective: it would generally improve the efficiency of their implementation or support interoperability at a more fine-grained level (or both)".

The Cloud-Native Computing technologies have been made much easier to apprehend, to master and to package into more and more complex systems. However, the effective usage of the vast catalogue of components potentially applicable for IoT Virtualization is still under evaluation in the IoT community.

The Proof-of-concept (PoC) of IoT Virtualization exposed in the present report is an implementation of the "Horizontal Up and Down Auto-Scaling" Use Case which has been selected in ETSI TR 103 527 [i.1] as a well-adapted example for the following reasons:

- It demonstrates the feasibility of IoT Virtualization on a "real-life" Use Case applicable to a large number of sectors (aka "verticals").
- It addresses a feature (auto-scaling) that is being deemed as very critical in virtualized IoT and for which an implementation via the use of "off-the-shelf" Open Source Software components requires some validation.
- It makes use of a great number of the Open Source Software components described in ETSI TR 103 528 [i.2].

4.3 Content of the report

Clause 5 outlines the main elements for the definition, design and implementation of the selected Use Case, mostly the High-Level Architecture, the main Open Source components selected and the message flow.

Clause 6 describes the initial deployment architecture and explains the steps to be followed for the set-up of the deployment infrastructure.

Clause 7 outlines the main lessons learned from the implementation and provides a few basic recommendations.

5 Main elements of the Proof-of-Concept

5.1 The Use Case

The amount and type of data transmitted by IoT devices may vary drastically in time depending on some events that can be internal or external to the virtualized IoT system (e.g. road traffic increase during holiday departure). A cloud-native IoT platform will be able to continuously monitor its resources, scale-up its capabilities when needed, then scale-down to an optimized state to avoid wasting resources. This capability is referred to as "Auto-Scaling.

The main objective of Auto Scaling is to ensure that the number of Virtual Machine (VM) instances available for and used by the virtualized application are optimal at a given time. Practically, a minimum number of VM instances is defined (lower threshold for the auto-scaling down) as well as a maximum number (upper threshold for the auto-scaling up). When needed, additional VMs are added (with an increment that can be predefined), used as long as needed and released when the usage is no longer needed.

5.2 High-Level Architecture

The main goal of this PoC is to experiment resource monitoring and autoscaling up and down in terms of communications, processing, storage and data analytics. A typical IoT architecture is considered composed of IoT devices, Message Brokers, Data Processors, Databases and Search Engines, and Dashboards. The monitoring and scaling service monitors resources consumed by each micro service and scale up and down the instances accordingly to optimize system performance. The PoC high level architecture is described in Figure 1.

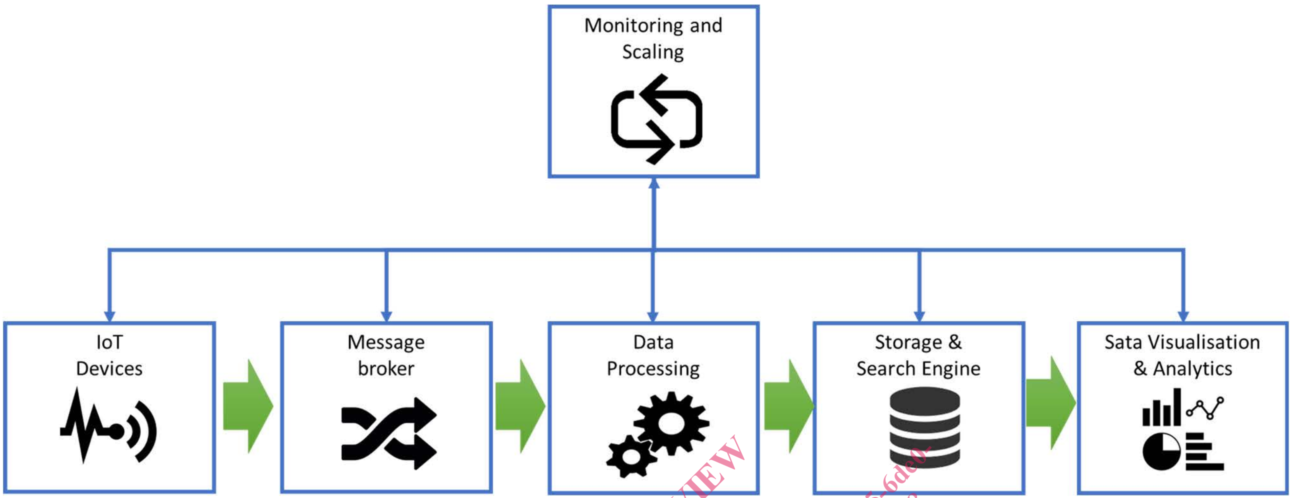

Figure 1: PoC High level architecture

5.3 Technical choices

The following open source components are considered to build the IoT system architecture as describe in Figure 2:

- Docker as Container solution: Docker is an open-source solution that automates the deployment of applications inside software containers. Docker implements a high-level API to provide lightweight containers that run processes in isolation. Building on top of facilities provided by the Linux kernel, a Docker container, unlike a virtual machine, does not require or include a separate operating system. Instead, it relies on the kernel's functionality and uses resource isolation.

- Kafka as Message Broker: Kafka is an open source a distributed streaming platform that follows a publish/subscribe architecture to manage data streams. It is used for two broad classes of application:

  1) Building real-time streaming data pipelines that reliably get data between systems or applications; and

  2) Building real-time streaming applications that transform or react to the streams of data. Kafka runs as a cluster on one or more servers. The Kafka cluster stores streams of records in categories called "topics".

- Logstash as data processor: Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to elastic search. It supports a variety of inputs that pull in events from a multitude of common sources, all at the same time. Easily ingest from logs, metrics, web applications, data stores, all in continuous, streaming fashion.

- Elasticsearch as database and search engine: Elasticsearch is an open source storage and real-time search and analytics engine capable of searching and analysing big volumes of data in near real time. It is open-source and is built to be used by distributed systems.

- Kibana as Visualization interface: Kibana is an open source web-based graphical interface for searching, analysing, and visualizing data stored in Elasticsearch. It uses the REST interface of Elasticsearch to retrieve the data and enables users to create customized dashboard views of their data and allows them to query and filter data. It allows to visualize data in a variety of charts, tables, lines, circles and maps.