ETSI GR MEC 027 V2.1.1 (2019-11)

Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies

Reference

DGR/MEC-0027ContainerStudy

Keywords

container, MEC

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

Contents

Intellectual Property Rights

Essential patents

Trademarks

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Multi-access Edge Computing (MEC).

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.

1 Scope

The present document focuses on identifying the additional support that needs to be provided by MEC when MEC applications run on alternative virtualization technologies, such as containers. The present document collects and analyses the use cases relating to the deployment of such alternative virtualization technologies, evaluates the gaps from the currently defined MEC functionalities, and identifies new recommendations. As ETSI NFV is also working on alternative virtualization technologies, the MEC work should be aligned with NFV where applicable. The present document also recommends the necessary normative work to close any identified gaps.

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 GS MEC 001: "Multi-access Edge Computing (MEC) Terminology".

[i.2] ETSI GS MEC 003: "Multi-access Edge Computing (MEC) Framework and reference architecture".

[i.3] ETSI GS MEC 010-2: "Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management".

[i.4] ETSI GS MEC 011: "Multi-access Edge Computing (MEC); Edge Platform Application Enablement".

[i.5] ETSI GR NFV-IFA 029: "Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS".

[i.6] ETSI GS NFV-EVE 004: "Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework".

[i.7] ETSI GS NFV 003 (V1.4.1): "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.8] ETSI GS NFV-INF 007: " Network Functions Virtualisation (NFV); Infrastructure; Methodology to describe Interfaces and Abstractions".

[i.9] ETSI GS MEC 012: "Multi-access Edge Computing (MEC); Radio Network Information API".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS MEC 001 [i.1] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS MEC 001 [i.1] and the following apply:

AVT	Alternative Virtualization Technology

4 Overview

The present document identifies the MEC features in order to enable the necessary support when MEC applications utilize alternative virtualization technologies, such as containers.

Clause 5 provides an overview of alternative virtualization technologies (AVTs), in a way that is heavily based on and aligned with NFV analysis in ETSI GS NFV-EVE 004 [i.6].

Clause 6 provides an analysis of impact of AVTs on MEC framework and reference architecture and provides a gap analysis against MEC framework and reference architecture specification (ETSI GS MEC 003 [i.2]). Recommendations for further work are provided.

Clause 7 provides an analysis of impact of AVTs on MEC management APIs and provides a gap analysis against relevant MEC specifications. Recommendations for further work are provided.

Clause 8 provides an analysis of impact of AVTs on MEC service exposure APIs and provides a gap analysis against relevant MEC specifications. Recommendations for further work are provided.

Annex A provides MEC deployment considerations based on container technology, and provides some guidance for application design when deployed on container.

5 Virtualization Technologies

5.1 Introduction

The ETSI MEC architectural framework as described in ETSI GS MEC 003 [i.2] introduces the virtualisation infrastructure of MEC host either as a generic or as a NFV Infrastructure (NFVI). Neither the generic virtualization infrastructure nor the NFVI restricts itself to using any specific virtualisation technology. Several virtualisation technologies are described in ETSI GS NFV-EVE 004 [i.6], including hypervisor-based solutions, OS containers, higher-level containers, nesting of virtualization technologies, mixing of virtualization technologies and mixing and nesting of virtualization technologies. This clause first introduces these several virtualisation technologies briefly and then analyse their impact on ETSI MEC architecture implementation.

5.2 Hypervisor-based solutions

5.2.1 Overview

The following text is based on clause 4.2.1 of ETSI GS NFV-EVE 004 [i.6] with minor changes.

A hypervisor is a software program that partitions the resources of a single hardware host and creates Virtual Machines (VM) isolated from each other. Each virtual machine appears to have the host's processor, memory and other resources, all to itself.

Each VM is assigned a virtualised CPU (vCPU), a virtualised NIC (vNIC) and a virtualised storage device (vStorage) created by the hypervisor. In practice, a vCPU may be a time sharing of a real CPU and/or in the case of multi-core CPUs, it may be an allocation of one or more cores to a VM. It is also possible that the hypervisor emulates a CPU instruction set that is different from the native CPU instruction set. However, emulation will significantly impact performance.

The hypervisor software runs either directly on top of the hardware (bare metal hypervisor, also known as Type I hypervisor) or on top of a hosting operating system (hosted hypervisor, also known as Type II hypervisor).

5.2.2 Application to MEC

Existing ETSI MEC specifications assume hypervisor based virtualization to be the default virtualization approach.

5.3 OS containers

5.3.1 Overview

The following text is based on clause 4.3.1 of ETSI GS NFV-EVE 004 [i.6] with no changes.

Container-based virtualisation, also called Operating System (OS)-level virtualisation, is an approach to virtualisation which allows multiple isolated user space instances on top of a kernel space within the OS. The isolated guests are called containers.

Figure 1 provides a high-level comparison of the software architectures for hypervisor solutions where the VNFC software image loaded in the virtualisation container includes both a guest OS kernel and the actual application, and OS container solutions where the VNFC software image loaded in the virtualisation container only includes the actual network application.

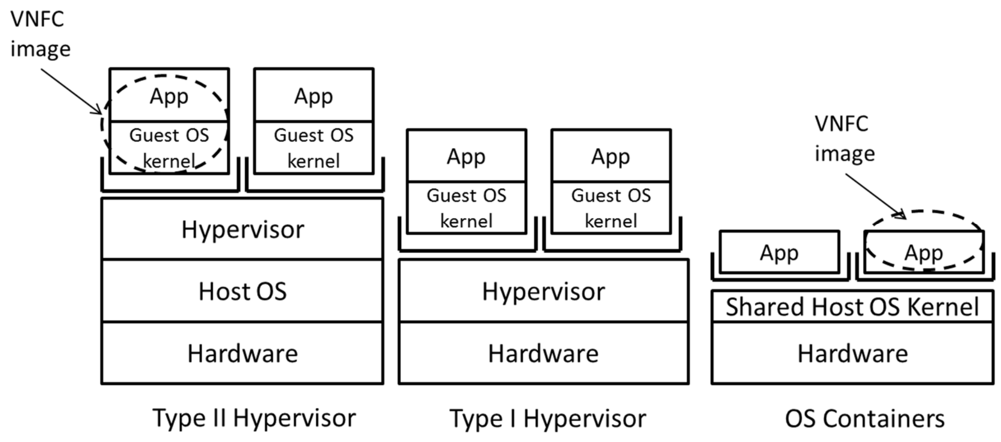

Figure 1: Hypervisor vs. OS Container solutions

The OS virtualisation technology allows partially shared execution context for different containers. Such a shared execution context is frequently referred to as a container pod. A pod might include shared file systems, shared network interfaces and other shared OS resources that are accessible from every container within that pod.

In addition to hypervisor-based execution environments that offer hardware abstraction and thread emulation services, the OS container execution environment provides kernel services as well. Kernel services include:

- Process control.

EXAMPLE 1: OS process creation; scheduling; wait and signal events; termination.

- Memory management.

EXAMPLE 2: Allocation and release of regular and large pages; handling memory-mapped objects and shared memory objects.

- File system management.
- File management.

EXAMPLE 3: Creation, removal, open, close, read and write file objects.

- Device management.

EXAMPLE 4: Request, release, configuration and access.

- Communication services.

EXAMPLE 5: Protocol stack services, channel establishment and release, PDU transmission and reception.

- System information maintenance.

EXAMPLE 6: Time and date, system and OS resource data, performance and fault indicators.

The OS container-to-VNFC logical interface is typically realized via:

- kernel system calls;
- signals to container processes;
- virtual file system mapped logical objects; and
- direct procedure calls into the container context.

OS virtualisation provides storage abstraction on file system level rather than on block device level. Each container has its separate file system view, where the guest file system is typically separated from the host file system. Containers within the same pod might share file systems where modifications made in one container are visible in the others.

Container file systems are realized either with standalone or with layered file systems. Standalone file systems are mapped into real file systems where all modifications made by the guest are stored in the backing real file system. Layered file systems take one or more base layers, and a writable overlay. A single layer is formed either from a real file system or from another layered file system structure. Layers are transparently overlaid and exposed as a single coherent file system. Typically, the lowermost layer contains an OS distribution with packages, libraries and run-times, while the overlay contains instance-specific customizations and modifications made by the container. While base layers are semi-permanently stored in image repositories, an overlay is disposable and its life time is coupled with the container life time.

5.3.2 Application to MEC

OS container solutions are more lightweight than hypervisor based VMs and enable faster application instantiation. As such, they may be an attractive option to a hypervisor based approach in edge-network situations that MEC is focused on.

5.4 Higher-level containers

5.4.1 Overview

The following text is based on clause 4.4.1 of ETSI GS NFV-EVE 004 [i.6] with no changes.

Higher level containers are a level of virtualisation technologies more dealing with software code and its development, deployment, and runtime environment. So the level of abstraction is on the runtime environment, where source code written in a certain programming or scripting language is deployed onto the NFVI. A few characteristics of such systems are that:

- source code is held and versioned in a code repository;
- source code dependencies are explicitly defined and packaged into the deployed software;
- code can be deployed into development, staging, or production environments without change;
- configuration of the software is stored in the environment, typically through environment variables;
- backing services such as data stores, message queues, and memory caches are accessed through a network and no distinction is made between local or third party services; and
- processes are stateless and therefore enable easy scale-out.

Typically, those containers are used in continuous deployment models enabling fast DevOps models for telecommunication services.

5.4.2 Application to MEC

Higher-level containers are often considered an alternative solution to OS containers and are particularly attractive when there is a need for deploying an application in form of source code, for example in DevOps environments. Like OS containers, the lightweight nature and ability to rapidly instantiate such applications may make them particularly attractive and important to MEC.

5.5 Nesting of virtualization technologies

5.5.1 Overview

The following text is based on clause 4.5.1 of ETSI GS NFV-EVE 004 [i.6] with minimal changes made to the original text and additional text added.