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

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

Disclaimer

The present document has been produced and approved by the Multi-access Edge Computing (MEC) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

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

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 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.

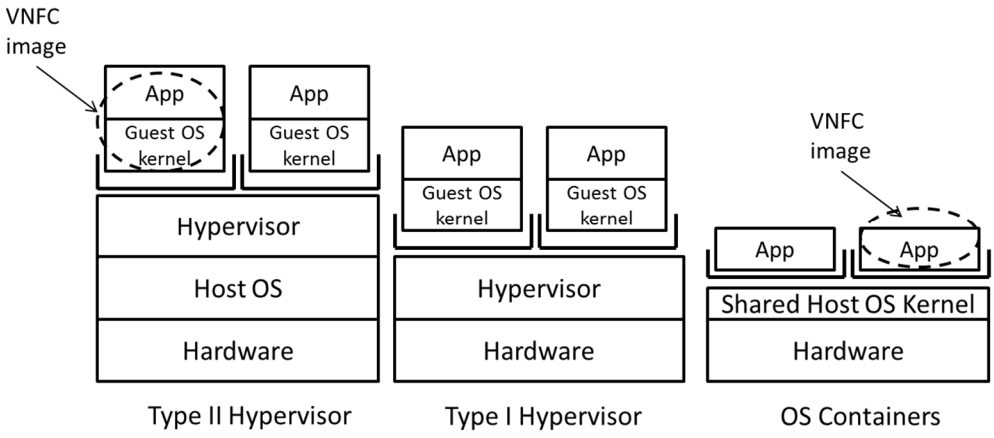

Figure 1: Hypervisor vs. OS Container solutions

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.

Within the NFVI, the virtualization layer may be composed of multiple nested sub-layers, each using a different virtualization technology. In this case, only the top sub-layer and its technology are visible to the Virtualized Infrastructure Manager (VIM); the top sub-layer creates partitions that provide the role of the virtualization container as defined in ETSI GS NFV 003 [i.7]. Resource partitioning in the other sub-layers is typically provisioned by means outside the scope of NFV Management and Orchestration functions (e.g. by a dedicated non-NFV infrastructure OSS). An example shown in Figure 2 is the case of a three-level virtualization layer, where the top level uses a higher layer virtualization technology, the layer below running OS container technology and the lowest layer uses the hypervisor technology. In this case, several higher-level containers, each hosting a VNFC instance, can run within each of the OS containers on one or several virtual machines created by the hypervisor.

Figure 2: Example sub-layering of nested virtualization technologies [i.7]

This is known as recursive virtualization in ETSI GS NFV-INF 007 [i.8], which highlights that an operating virtual functional block can itself be a host functional block.

One significant potential application, highlighted in Figure 2, is the use of non-NFV managed virtualization systems in the nested virtualization sub-layers in such a case. In the context of MEC, this is important as it creates opportunities to support multiple tenants under certain flexible infrastructure sharing scenarios as described in clause 5.5.2.

5.5.2 Application to MEC

A differentiating aspect of MEC, as compared to NFV, is the potential need to deploy applications belonging to multiple third parties. By extension, third-party application environments, e.g. OS-container based application environments or higher-layer container based environments, may be supported where application deployment inside each environment is managed by the environment operators, not the operator of the MEC host. Nesting of virtualization technologies allows MEC host operators to support multiple such environments by allocating a VNF (typically, but not always, consisting of a single VNFC) to each third-party application environment owner. Each third party application environment owner is then able to further allocate the resources assigned to its VNF to the multiple applications it runs, and to do so based on its own internal criteria. This situation is illustrated in Figure 3.

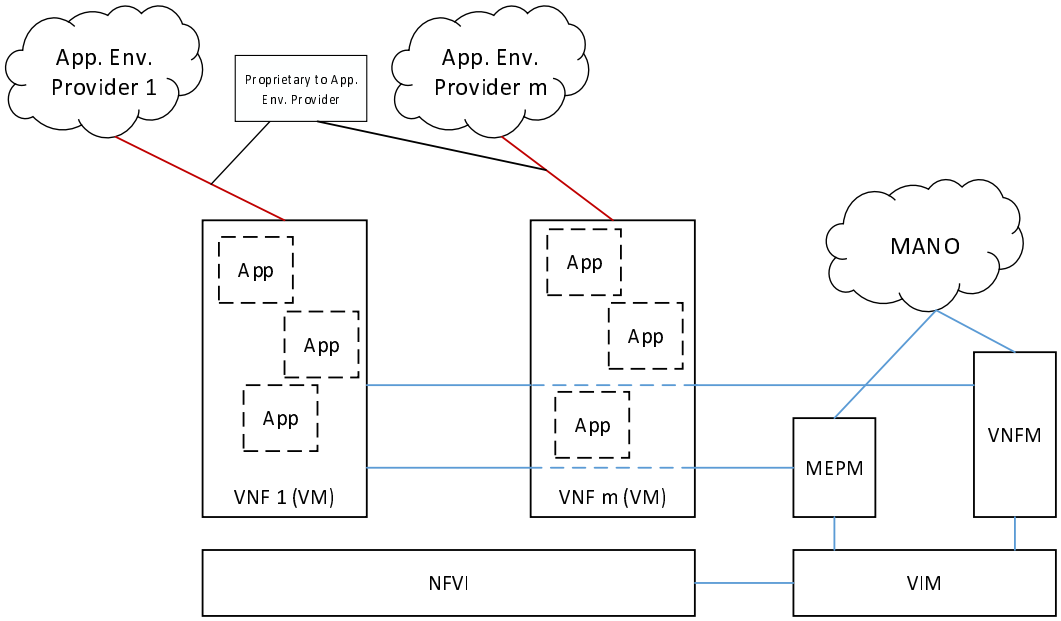

Figure 3: Illustrating a key use case for nested virtualization in MEC

As described above, Figure 3 illustrates VNFs 1 to m (each consisting of a single VM) instantiated on an NFVI managed by VIM. The management of VNFs VNF1 to VNFm is provided by the MANO components as defined by ETSI NFV and ETSI MEC (MEPM, VNFM, etc.). However, what happens inside each VNF is managed by the application environment providers own management system (usually elsewhere in the cloud). In particular, it is this management system that provides all application management within the VNF and such management does not need to be coordinated with the MEC/NFV MANO system.

Notably, in this case, the top-level MEC host operator does not perform any management functions for the applications in the sub-layer. Its management and service delivery interactions are strictly limited to the virtualization sub-layer 1, while the operators of the MEC application environments manage all interactions with their client applications. Moreover, at this time, this solution can be enabled only using traditional hypervisor-based virtualization and the top layer. As such, from a MEC operator point of view, this approach appears to be the same as a hypervisor-based solution - it is not aware of nesting.

5.6 Mixing of virtualization technologies

5.6.1 Overview

The following text is based on clause 4.6.1 of ETSI GS NFV-EVE 004 [i.6] with no changes.

In the case of mixing different virtualisation technologies, instances of the VNFCs of a certain VNF can run the different technologies, each benefiting of the particular characteristics of the virtualisation technology of choice. The different virtualisation technologies can be controlled by one or different VIMs. Figure 4 illustrates a deployment option where a single VIM controls three different virtualisation technologies.

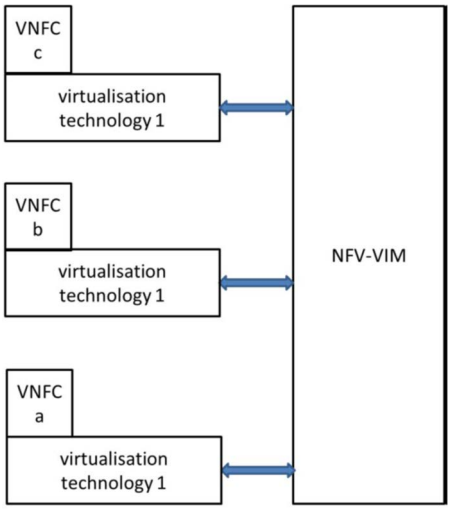

Figure 4: Example of Mixing of Virtualisation Technologies [i.7]

5.6.2 Application to MEC

Mixing of virtualization technologies is a flexible solution when deploying applications of a particular technology. There may be applications running on different virtualisation technologies on the MEC platform.

5.7 Mixing and nesting of virtualization technologies

The following text is based on clause 4.6.1 of ETSI GS NFV-EVE 004 [i.6] with no changes.

As with nesting the NFVI virtualisation layer can be composed of multiple nested sub-layers, each using a different virtualisation technology. In addition, as illustrated in Figure 5, the components of the VNF run on different nested layers, each benefiting of the particular characteristics of virtualisation technology of choice.

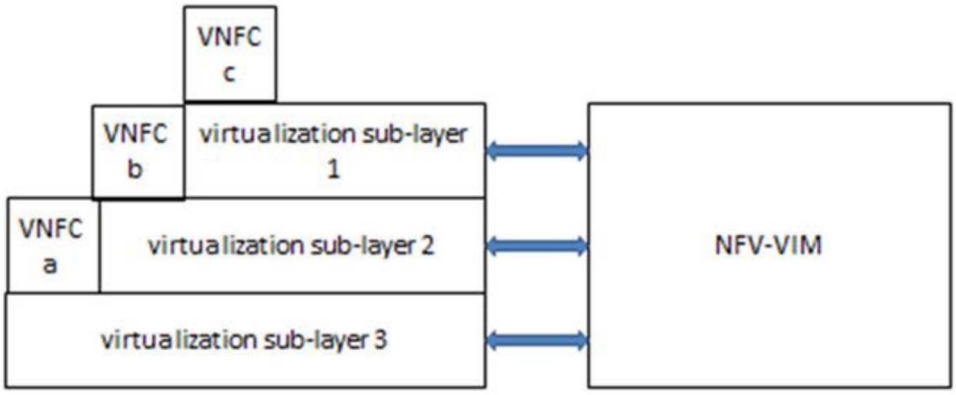

Figure 5: Example of Mixing and Nesting Virtualisation Technologies [i.7]

6 Impact of AVT on Framework and Reference Architecture

6.1 Overview

MEC framework and reference architecture is specified in ETSI GS MEC 003 [i.2]. The specification introduces the MEC reference architecture with the definitions of all relevant functional entities of MEC and the reference points between them. In MEC reference architecture the impact from AVTs is on those functional entities which are involved in application package handling, i.e. OSS, MEC orchestrator, NFVO and MEC platform manager, and on those which are managing or hosting the virtualised resources, i.e. VIM and the NFVI. Finally the definition of the MEC application itself is impacted by the virtualisation technology.

Due to the scope and depth of ETSI GS MEC 003 [i.2], all the above mentioned functional entities, except the MEC application, have been defined as agnostic to any specific virtualization technology. The same applies to the reference points in the reference architecture as defined in ETSI GS MEC 003 [i.2].

The definition of the MEC application in clause 7.1.3 of ETSI GS MEC 003 [i.2] defines the MEC application as a Virtual Machine only.

In the informative clause A.1 MEC host selection, the information considered by the MEC orchestrator when selecting a MEC host for a MEC application includes the required MEC features of the hosts, such as VM relocation capability. However, the wording implies that the VM relocation capability is only an example of the said MEC features.

6.2 Gap Analysis

The current definition of the MEC application does not accommodate any other virtualization technology than VM. Consequently the definition needs to be generalized to any virtualization technology.

6.3 Recommendations

It is recommended to change the MEC application definition as follows:

"*MEC application runs as a virtualized application, such as a virtual machine (VM) or a containerised application, on top of the virtualisation infrastructure provided by the MEC host, and can interact with the MEC platform to consume and provide MEC services (described in clause 8)*".

7 Impact of AVT on Management API Specifications

7.1 Overview

The rapid evolution and adaption of AVTs combined with the needs of real-world scenarios for edge computing services make it likely that multiple AVT solutions may need to be supported in the same MEC host. In order to do this, the following points need to be considered:

- Heterogeneous VIM(s): Support of a heterogeneous AVT environment may be achievable via either a VIM capable of supporting heterogeneous AVTs or by supporting the different virtualization technologies via different management functions, i.e. separate VIMs or entities interacting with each other to provide nested AVTs. A management framework should be able to support both. Notably, ETSI GR NFV-IFA 029 [i.5] is addressing these issues and has studied and addressed several potential approaches supporting OS-containers nested in VMs or deployed on bare-metal servers.

- Unified management view: the management framework should have a unified view of the different AVT infrastructure resources to ensure consistent management. Notably, ETSI GR NFV-IFA 029 [i.5] is addressing these issues and has studied and addressed several potential approaches.

- Lightweight infrastructure: The management infrastructure should be as lightweight as possible.
- On-demand usage: In many cases, MEC applications and services need infrastructure resources on demand and for very short period of time. The management infrastructure needs to be able to manage resources dynamically in response to application needs.

AVTs, by their very nature, require a different approach towards management of the underlying system resources than do traditional VMs. For example, OS containers are intended to be ephemeral and stateless; state (i.e. data that needs to live beyond the life of a container instance) is stored outside of a container. This means that management of resources for OS container-based applications needs to be very different than for VMs which are stateful, long-living and typically have all their resources packaged together. Another example is networking for OS containers, which shares share a common set of IP resources - quite different from VMs which are assigned their own virtual Network Interface Cards (NICs).

The impact of addressing these different requirements is spread between the key management entities. For examples, in the MEC-in-NFV Reference Architecture (ETSI GS MEC 003 [i.2]) these are the VIM, NFV Management Entities (VNFM and NFVO) and MEC Management Entities (MEPM and MEAO). Of these, the MEC Management entities are furthest removed from the details of virtualization and thus the impact on MEC Management is expected to be small, with most of the impact addressed by VIM and NFV management stack. Nonetheless, some impact on the MEC Application Lifecycle Management APIs is expected.

7.2 Gap Analysis

MEC reference architecture (ETSI GS MEC 003 [i.2]) should make clear the potential need to support heterogeneous AVTs via one or more VIMs.

ETSI GR NFV-IFA 029 [i.5] provides and in depth study of the architectural options to support OS containers nested on virtual machines and on bare metal. Assuming the recommendations of ETSI GR NFV-IFA 029 [i.5] are acted upon by ETSI NFV and the resulting features are available in standard NFV components, no further action from ETSI MEC should be needed in this regard.

ETSI GS MEC 010-2 [i.3] specifies a number of application lifecycle management operations as well as the application descriptor (AppD).

The information and data models provided in the application descriptor (AppD) may need to be adopted to the specifics of various AVTs.

The Application Lifecycle Management operations can be grouped as follows with the expected impact summarized below:

- **Create/Delete Application Instance ID:** no impact from AVTs is expected.
- **Instantiate/Terminate Application:** some potential impacts to the detailed API information elements and the Application Descriptor may be necessary to reflect AVT specific aspects.
- **Change Application Instance State:** some potential impacts to the detailed API information elements may be necessary to reflect AVT specific aspects.
- **Query Application Instance State:** the content of the query response may need to be adapted to reflect AVT specific information.
- **Query Application LCM Status:** the content of the query response may need to be adapted to reflect AVT specific information.

For the Application Lifecycle Change Notification interface, there is no impact expected on the Subscribe operation, with the impact on the Notify operation similar to that on the Query Application LCM Status operation above.

The Application Package Management operations are not expected to be impacted, however the Application Package itself may need to be adapted to the nature of AVTs (for example, OS Containers have separate state storage; high-level containers have code, not images included in the package).

7.3 Recommendations

Update ETSI GS MEC 010-2 [i.3] to expand definition of application lifecycle management operations to cover AVTs.

Update ETSI GS MEC 003 [i.2] to reflect options to support different AVTs.

8 Impact of AVT on Service Exposure API Specifications

8.1 Overview

ETSI MEC Service Exposure APIs, such as those specified in ETSI GS MEC 011 [i.4], ETSI GS MEC 012 [i.9], etc., are designed to provide a standardized set of APIs for exposure of services which are abstracted from the specifics of the implementation of service enablement. As such, a change in underlying virtualization technology is expected to have no impact on the API specifications.

8.2 Gap Analysis

No gaps are identified.

8.3 Recommendations

No changes to service exposure API specifications are recommended.

Annex A:
MEC deployment considerations based on container

A.1 Overview

Container technology is increasingly valued by MEC application developers. MEC will likely be introduced step by step into containerized MEC applications and container management platforms. Containers are used to package various applications and provide a unified development, testing, and production environment for upper-layer applications.

A.2 Application Deployment Suggestions

It is desirable that the image format of the application supports containing and describing applications using different AVTs.

History

Document history		
V2.1.1	November 2019	Publication