

NG.126 Cloud Infrastructure Reference Model
Version 5.0
10 June 2025

Security Classification: Non-Confidential

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

Copyright Notice

Copyright © 2025 GSM Association

Disclaimer

The GSMA makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Compliance Notice

The information contain herein is in full compliance with the GSMA Antitrust Compliance Policy.

This Permanent Reference Document is classified by GSMA as an Industry Specification, as such it has been developed and is maintained by GSMA in accordance with the provisions set out GSMA AA.35 - Procedures for Industry Specifications.

•

Table of Contents

1 Introduction Error! Bookmark not defined.
1.1 Overview Error! Bookmark not defined.
1.2 Scope **Error! Bookmark not defined.**
1.3 Principles **Error! Bookmark not defined.**
1.4 Definitions/terminology/abbreviations **Error! Bookmark not defined.**
1.5 Conventions **Error! Bookmark not defined.**
1.6 References **Error! Bookmark not defined.**
1.7 Bibliography **Error! Bookmark not defined.**
2 Workload Requirements and Analysis Error! Bookmark not defined.
2.1 Workloads collateral **Error! Bookmark not defined.**
2.2 Use cases **Error! Bookmark not defined.**
2.2.1 Use case no. 1: Edge CDN with eMBB core network slicing **Error! Bookmark not defined.**
2.2.2 Use case no. 2: Edge private 5G with core network slicing **Error! Bookmark not defined.**
2.2.3 Use case no. 3: Edge automotive (V2X) with uRLLC core network slicing **Error! Bookmark not defined.**
2.2.4 Use case no. 4: Edge vRAN deployments **Error! Bookmark not defined.**
2.2.5 Use case no. 5: Telepresence experience **Error! Bookmark not defined.**
2.2.6 Use case no. 6: Digital twins for manufacturing **Error! Bookmark not defined.**
2.3 Analysis **Error! Bookmark not defined.**
2.4 Profiles, profile extensions, and flavours **Error! Bookmark not defined.**
2.4.1 Profiles (top-level partitions) **Error! Bookmark not defined.**
2.4.2 Profile extensions (specialisations) **Error! Bookmark not defined.**
3 Modelling Error! Bookmark not defined.
3.1 Model **Error! Bookmark not defined.**
3.2 Virtual Infrastructure Layer **Error! Bookmark not defined.**
3.2.1 Virtual Resources **Error! Bookmark not defined.**
3.2.2 Virtual Infrastructure Manager **Error! Bookmark not defined.**
3.3 Hardware Infrastructure Layer **Error! Bookmark not defined.**
3.3.1 Hardware Infrastructure Resources **Error! Bookmark not defined.**
3.3.2 Hardware Infrastructure Manager **Error! Bookmark not defined.**
3.4 The Redfish® Standard based Hardware Infrastructure Manager **Error! Bookmark not defined.**
3.5 How Redfish fits into the ETSI NFV Networking Reference Model **Error! Bookmark not defined.**
3.6 Network **Error! Bookmark not defined.**
3.6.1 Network Principles **Error! Bookmark not defined.**
3.6.2 Network Layering and Concepts **Error! Bookmark not defined.**
3.6.3 Networking Reference Model **Error! Bookmark not defined.**
3.6.4 Deployment Examples Based on the Networking Reference Model **Error! Bookmark not defined.**
3.6.5 Service Function Chaining **Error! Bookmark not defined.**

3.6.6 Time Sensitive Networking **Error! Bookmark not defined.**
3.6.7 Load Balancer **Error! Bookmark not defined.**
3.6.8 Kubernetes Networking Semantics **Error! Bookmark not defined.**
3.7 Storage **Error! Bookmark not defined.**
3.7.1 Introduction to Storage **Error! Bookmark not defined.**
3.7.2 Storage Implementation Stereotypes **Error! Bookmark not defined.**
3.7.3 Storage for Tenant Consumption **Error! Bookmark not defined.**
3.7.4 Storage Scenarios and Architecture Fit **Error! Bookmark not defined.**
3.8 Sample reference model realization **Error! Bookmark not defined.**
3.9 Hardware Acceleration Abstraction **Error! Bookmark not defined.**
3.9.1 Types of Accelerators **Error! Bookmark not defined.**
3.9.2 Infrastructure and Application Level Acceleration **Error! Bookmark not defined.**
3.9.3 Example of O-RAN Acceleration Abstraction Layer Interface **Error! Bookmark not defined.**
3.9.4 Workload Placement **Error! Bookmark not defined.**
3.9.5 CPU Instructions **Error! Bookmark not defined.**
3.9.6 Fixed Function Accelerators **Error! Bookmark not defined.**
3.9.7 Firmware-programmable Adapters **Error! Bookmark not defined.**
3.9.8 SmartNICs **Error! Bookmark not defined.**
3.9.9 Smart Switches **Error! Bookmark not defined.**
3.9.10 Decoupling Applications from Infrastructure and Platform with Hardware Acceleration **Error! Bookmark not defined.**
3.10 Address Family For XDP (AF_XDP) **Error! Bookmark not defined.**
3.11 Energy efficiency **Error! Bookmark not defined.**
4 Infrastructure Capabilities, Measurements and Catalogue Error! Bookmark not defined.
4.1 Capabilities and performance measurements **Error! Bookmark not defined.**
4.1.1 Exposed versus internal **Error! Bookmark not defined.**
4.1.2 Exposed infrastructure capabilities **Error! Bookmark not defined.**
4.1.3 Internal infrastructure capabilities **Error! Bookmark not defined.**
4.1.4 Cloud Infrastructure Management Capabilities **Error! Bookmark not defined.**
4.1.5 Cloud Infrastructure Management Performance Measurements **Error! Bookmark not defined.**
4.1.6 Acceleration/offload API requirements **Error! Bookmark not defined.**
4.2 Profiles and workload flavours **Error! Bookmark not defined.**
4.2.1 Profiles **Error! Bookmark not defined.**
4.2.2 Profiles specifications and capability mapping **Error! Bookmark not defined.**
4.2.3 Profile extensions **Error! Bookmark not defined.**
4.2.4 Workload flavours and specifications of other capabilities **Error! Bookmark not defined.**
4.2.5 Virtual network interface specifications **Error! Bookmark not defined.**
4.2.6 Storage extensions **Error! Bookmark not defined.**
5 Feature Set and Requirements from Infrastructure Error! Bookmark not defined.
5.1 Cloud Infrastructure software profile description **Error! Bookmark not defined.**
5.1.1 Virtual compute profiles **Error! Bookmark not defined.**

5.1.2 Virtual storage profiles **Error! Bookmark not defined.**
5.1.3 Virtual networking profiles **Error! Bookmark not defined.**
5.1.4 Security details **Error! Bookmark not defined.**
5.1.5 Platform services **Error! Bookmark not defined.**
5.2 Features and requirements of the Cloud Infrastructure software profiles **Error! Bookmark not defined.**
5.2.1 Virtual compute features and configuration **Error! Bookmark not defined.**
5.2.2 Virtual storage features and configuration **Error! Bookmark not defined.**
5.2.3 Virtual networking features and configuration **Error! Bookmark not defined.**
5.3 Cloud Infrastructure hardware profile description **Error! Bookmark not defined.**
5.4 Features and requirements of the Cloud Infrastructure hardware profiles **Error! Bookmark not defined.**
5.4.1 Compute resources **Error! Bookmark not defined.**
5.4.2 Storage configurations **Error! Bookmark not defined.**
5.4.3 Network resources **Error! Bookmark not defined.**
6 External Interfaces Error! Bookmark not defined.
6.1 Introduction to external interfaces **Error! Bookmark not defined.**
6.2 Cloud Infrastructure APIs **Error! Bookmark not defined.**
6.2.1 Tenant-level APIs **Error! Bookmark not defined.**
6.2.2 Hardware acceleration interfaces **Error! Bookmark not defined.**
6.3 Intra-Cloud Infrastructure interfaces **Error! Bookmark not defined.**
6.3.1 Hypervisor hardware interface **Error! Bookmark not defined.**
6.4 Enabler services interfaces **Error! Bookmark not defined.**
7 Security Error! Bookmark not defined.
7.1 Introduction to security **Error! Bookmark not defined.**
7.2 Potential attack vectors **Error! Bookmark not defined.**
7.3 Security scope **Error! Bookmark not defined.**
7.3.1 In-scope and out-of-scope definition **Error! Bookmark not defined.**
7.3.2 High-level security requirements **Error! Bookmark not defined.**
7.3.3 Common security standards **Error! Bookmark not defined.**
7.4 Cloud infrastructure security **Error! Bookmark not defined.**
7.4.1 General platform security **Error! Bookmark not defined.**
7.4.2 Platform backend access security **Error! Bookmark not defined.**
7.4.3 Platform frontend access security **Error! Bookmark not defined.**
7.4.4 Infrastructure as a Code security **Error! Bookmark not defined.**
7.4.5 Security of production and non-production environments **Error! Bookmark not defined.**
7.5 Workload security and vendor responsibility **Error! Bookmark not defined.**
7.5.1 Software hardening **Error! Bookmark not defined.**
7.5.2 Port protection **Error! Bookmark not defined.**
7.5.3 Software code quality and security **Error! Bookmark not defined.**
7.5.4 Alerting and monitoring **Error! Bookmark not defined.**
7.5.5 Logging **Error! Bookmark not defined.**
7.6 Workload security: cloud infrastructure operator responsibility **Error! Bookmark not defined.**

7.6.1 Remote attestation/OpenCIT **Error! Bookmark not defined.**
7.6.2 Workload image **Error! Bookmark not defined.**
7.6.3 Networking security zoning **Error! Bookmark not defined.**
7.6.4 Volume encryption **Error! Bookmark not defined.**
7.6.5 Root of trust for measurements **Error! Bookmark not defined.**
7.6.6 Zero Trust Architecture **Error! Bookmark not defined.**
7.7 Software supply chain security **Error! Bookmark not defined.**
7.7.1 Software security **Error! Bookmark not defined.**
7.7.2 Open-source software security **Error! Bookmark not defined.**
7.7.3 Software Bill of Materials **Error! Bookmark not defined.**
7.7.4 Vulnerability identification **Error! Bookmark not defined.**
7.8 Testing and certification **Error! Bookmark not defined.**
7.8.1 Testing demarcation points **Error! Bookmark not defined.**
7.8.2 Certification requirements **Error! Bookmark not defined.**
7.9 Cloud infrastructure regulatory compliance **Error! Bookmark not defined.**
7.9.1 United States of America (U.S.A.) **Error! Bookmark not defined.**
7.9.2 European Union (EU) **Error! Bookmark not defined.**
7.9.3 United Kingdom (UK) **Error! Bookmark not defined.**
7.9.4 Australia **Error! Bookmark not defined.**
7.10 Consolidated security requirements **Error! Bookmark not defined.**
7.10.1 System hardening **Error! Bookmark not defined.**
7.10.2 Platform and access **Error! Bookmark not defined.**
7.10.3 Confidentiality and integrity **Error! Bookmark not defined.**
7.10.4 Workload security **Error! Bookmark not defined.**
7.10.5 Image security **Error! Bookmark not defined.**
7.10.6 Security LCM **Error! Bookmark not defined.**
7.10.7 Monitoring and security audit **Error! Bookmark not defined.**
7.10.8 Open-source software **Error! Bookmark not defined.**
7.10.9 IaaC - Secure design and architecture stage requirements **Error! Bookmark not defined.**
7.10.10 IaaC - Secure code stage requirements **Error! Bookmark not defined.**
7.10.11 IaaC - Continuous build, integration, and testing stage requirements **Error! Bookmark not defined.**
7.10.12 IaaC - Continuous delivery and deployment stage requirements **Error! Bookmark not defined.**
7.10.13 IaaC - Runtime defence and monitoring requirements **Error! Bookmark not defined.**
7.10.14 Compliance with standards **Error! Bookmark not defined.**
7.11 Additional security references **Error! Bookmark not defined.**
8 Hybrid Multicloud: Data Centre to Edge Error! Bookmark not defined.
8.1 Introduction to Hybrid Multicloud **Error! Bookmark not defined.**
8.2 Hybrid multicloud architecture **Error! Bookmark not defined.**
8.3 Characteristics of a federated cloud **Error! Bookmark not defined.**
8.4 Telco cloud **Error! Bookmark not defined.**
8.4.1 Telco operator platform conceptual architecture **Error! Bookmark not defined.**

8.5 Multicloud interactions model **Error! Bookmark not defined.**
8.5.1 Stereotypical scenarios **Error! Bookmark not defined.**
8.5.2 Requirements, reference architecture and industry standards intersect **Error! Bookmark not defined.**
8.5.3 Hybrid, Edge, and Multicloud unified management Platform **Error! Bookmark not defined.**
8.5.4 Aspects of multicloud security **Error! Bookmark not defined.**
8.6 Telco Edge Cloud (TEC) **Error! Bookmark not defined.**
8.6.1 Telco Edge Cloud: deployment environment characteristics **Error! Bookmark not defined.**
8.6.2 Telco Edge Cloud: infrastructure characteristics **Error! Bookmark not defined.**
8.6.3 Telco Edge Cloud: infrastructure profiles **Error! Bookmark not defined.**
8.6.4 Telco Edge Cloud: platform services deployment **Error! Bookmark not defined.**
8.6.5 Comparison of deployment topologies and edge terms **Error! Bookmark not defined.**
8.6.6 O-RAN alignment and interaction **Error! Bookmark not defined.**
9 Infrastructure Operations and Lifecycle Management Error! Bookmark not defined.
9.1 Introduction to Infrastructure Operations and Lifecycle Management **Error! Bookmark not defined.**
9.2 Infrastructure lifecycle model **Error! Bookmark not defined.**
9.3 Configuration and lifecycle management **Error! Bookmark not defined.**
9.4 Capacity management **Error! Bookmark not defined.**
9.5 Assurance **Error! Bookmark not defined.**
9.5.1 Telemetry and observability **Error! Bookmark not defined.**
9.6 Automation **Error! Bookmark not defined.**
9.6.1 Infrastructure lifecycle management automation **Error! Bookmark not defined.**
9.6.2 Software onboarding automation and CI/CD requirements **Error! Bookmark not defined.**
9.6.3 Tenant creation automation **Error! Bookmark not defined.**
10 Challenges and Gaps Error! Bookmark not defined.
10.1 Introduction to Challenges and Gaps **Error! Bookmark not defined.**
10.2 Challenges **Error! Bookmark not defined.**
10.3 Gaps **Error! Bookmark not defined.**
10.3.1 Discovery **Error! Bookmark not defined.**
10.3.2 Supporting the load balancing of VNF/CNFs **Error! Bookmark not defined.**
10.3.3 Closed-loop automation **Error! Bookmark not defined.**
10.3.4 Hybrid multicloud: APIs **Error! Bookmark not defined.**
10.3.5 AF_XDP **Error! Bookmark not defined.**
11 Appendix A: Guidelines For Application Vendors Error! Bookmark not defined.
11.1 Goals **Error! Bookmark not defined.**
11.2 Introduction and terminology **Error! Bookmark not defined.**
11.3 VNF design guidelines **Error! Bookmark not defined.**
11.4 Miscellaneous **Error! Bookmark not defined.**
11.4.1 VNF ntwork monitoring capabilities: use case **Error! Bookmark not defined.**

12 Document Management **216**
A.1 Document History 216
A.2 Other Information 216

1 Introduction

1.1 Overview

The Reference Model (RM) specifies a virtualisation technology-agnostic (VM-based and container-based) cloud infrastructure abstraction. It acts as a catalogue of the exposed infrastructure capabilities, resources, and interfaces required by the workloads. This document has been developed by the Linux Foundation Networking project cntt:common/chapter00.

Problem statement: Based on community consultations, including Telco operators, technology suppliers, and software developers, it is understood that there are significant technical, operational, and business challenges to the development and deployment of Virtual Network Functions (VNF) and Cloud-Native Network Functions (CNF), due to the lack of a common cloud infrastructure platform. These include, but are not limited to, the following:

1. Higher development costs, due to the need to develop virtualised/containerised network applications on multiple custom platforms for each operator.
2. Increased complexities, due to the need to maintain multiple versions of applications to support each custom environment.
3. Lack of testing and validation commonalities, leading to inefficiencies and increased time to market. While the operators will still perform internal testing, the application developers utilising an industry standard verification program on a common cloud infrastructure would lead to efficiencies and faster time to market.
4. Slower adoption of cloud-native applications and architectures. A common Telco cloud may provide an easier path to methodologies that will drive faster cloud-native development.
5. Increased operational overheads, due to the need for operators to integrate diverse and sometimes conflicting cloud platform requirements.

One of the main challenges holding back the more rapid and widespread adoption of virtualised/containerised network applications is when, while building or designing their virtualised services, specific infrastructure assumptions and requirements are implied, often with custom design parameters. This forces the operators to build complex integrations of various vendor-/function-specific silos which are incompatible with each other and might possibly have differing and conflicting operating models. This also makes the onboarding and conformance processes of the VNFs/CNFs (coming from different vendors) hard to automate and standardise. There is a clear need for a common model across the industry to facilitate more rapid adoption.

This document starts from the abstract and, as it progresses, becomes more detailed. It follows the traditional design process where you start from the core principles, progress to abstract concepts and models, and then finish with operational considerations, such as security and lifecycle management.

1. **Chapter 01 - Introduction**: This provides an overall scope of the Reference Model document, including the goals and objectives of the project.

1. **Audience**: This chapter is aimed at a general technical audience with an interest in this topic.

1. **Chapter 02 - Workload Requirements and Analysis**: This chapter covers the high-level requirements and core principles needed to understand how the model was developed. It addresses the thinking behind the decisions that were made.

2. **Audience**: This chapter is aimed at architects and others with an interest in how the decisions were made.

2. **Chapter 03 - Modelling**: This chapter covers the high-level cloud infrastructure model itself.

3. **Audience**: This chapter is aimed at architects and others who want to gain a quick high-level understanding of the model.

3. **Chapter 04 - Infrastructure Capabilities, Metrics, and Catalogue**: This chapter provides details about the capabilities needed to support the various types of workloads and how the capabilities are applied to the model. Details regarding T-shirt sizes and other considerations are found in this section.

4. **Audience**: This chapter is aimed at architects, developers, and others who need to deploy infrastructure or develop applications.

4. **Chapter 05 - Feature set and Requirements from Infrastructure**: This chapter goes into more detail on what needs to be part of the cloud infrastructure. It describes the software and hardware capabilities, and the configurations recommended for the different types of cloud infrastructure profiles.

5. **Audience**: This chapter is aimed at architects, developers, and others who need to deploy infrastructure or develop applications.

5. **Chapter 06 - External Interfaces**: This chapter covers APIs and other interfaces needed to communicate with the workloads and other external components.

6. **Audience**: This chapter is aimed at architects, developers, and others who need to develop APIs or applications that use the APIs.

6. **Chapter 07 - Security**: This chapter identifies the security requirements that need to be taken into consideration when designing and implementing a cloud infrastructure environment. It does not cover details related to company-specific requirements to meet regulatory requirements.

7. **Audience**: This chapter is aimed at security professional, architects, developers, and others who need to understand the role of security in the cloud infrastructure environment.

7. **Chapter 08 - Hybrid Multicloud: Data Centre to Edge**: A generic Telco cloud is a hybrid multicloud, or a federated cloud, that is deployed in large data centres, central offices or colocation facilities, and the edge. This chapter discusses the characteristics of Telco edge and hybrid multicloud.

8. **Audience**: This chapter is aimed at a general technical audience with an interest in this topic.

8. **Chapter 09 - Lifecycle Management**: This chapter focuses on the operational aspects of the cloud infrastructure. Discussions include deployment considerations, ongoing management, upgrades, and other lifecycle concerns and requirements. It does not cover details related to company-specific operational requirements, nor does it go into how the cloud infrastructure interfaces with the existing BSS/OSS systems.

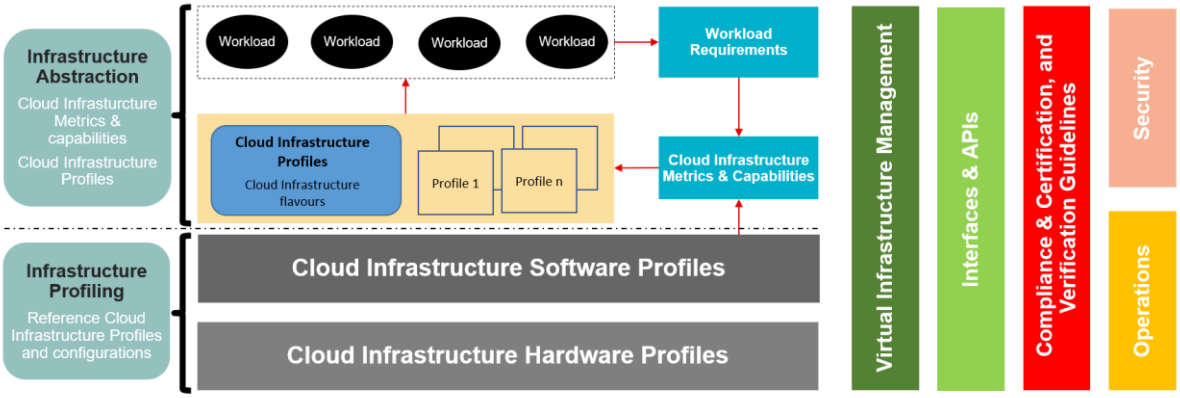
9. **Audience**: This chapter is aimed at lifecycle managers, operational support teams, and others who need to support the infrastructure or the applications.

9. **Chapter 10 - Challenges and Gaps**: This chapter discusses the opportunities for future developments, as the technology changes over time.

10. **Audience**: This chapter is aimed at a general technical audience with an interest in this topic.

1.2 Scope

This **Reference Model** document focuses on identifying the abstractions and associated concepts that are needed to represent the cloud infrastructure. See scope-of-reference-model for the scope of the Reference Model.



Scope of the Reference Model

This document specifies the following:

10. **Cloud infrastructure abstraction**: In context with how it interacts with the other components required to build a complete cloud system that supports workloads deployed in virtual machines (VMs) or containers. Network function workloads that are deployed on virtual machines and containers are referred to as virtual network functions (VNFs) and containerised network functions (CNFs), respectively.

Note: CNFs are now more commonly referred to as cloud-native network functions.

1. **Cloud infrastructure capabilities and metrics**: This is a set of cloud infrastructure capabilities and metrics required to perform Telco scale network functions, and satisfy their performance criteria.
2. **Infrastructure profiles catalogue**: This is a catalogue of standard infrastructure software and hardware configurations, referred to as profiles. These profiles abstract the infrastructure for the workloads. Only a few profiles, with well-defined characteristics, can meet the operational and performance requirements of all the workloads.

11. **Cloud infrastructure software and hardware profiles**:
    1. **Cloud infrastructure software profiles**: These software profiles are components of the corresponding infrastructure profiles within the infrastructure profiles catalogue. They specify the host infrastructure software configurations.
    2. **Cloud infrastructure hardware profiles**: These hardware profiles are components of the corresponding infrastructure profiles within the infrastructure profiles catalogue. They specify the host infrastructure hardware configurations.
12. **Conformance and verification**:

1. **Conformance programs**: These define the requirements for verification and validation programs for the cloud infrastructure and workloads.
2. **Test framework**: This document provides the input for the test suites to allow the conformance of the cloud infrastructure and the workloads.

1.3 Principles

The Reference Model specifications conform to the overall principles defined in common/chapter00:anuket general principles.

1.4 Definitions/terminology/abbreviations

To help guide the reader, the Anuket Glossary [48] provides an introduction to the main terms used within this document and throughout the project as a whole. These definitions are, with a few exceptions, based on the ETSI GR NFV 003 [7] definitions. In a few cases, they have been modified to avoid deployment technology dependencies, only when it is necessary to avoid confusion.

For a full list of abbreviations used in this document, see the Anuket Abbreviations [49].

1.5 Definition of Acronyms and Terms

1.6

1.7 Acronyms

Term	Description
3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorisation, and Accounting
AAL	Acceleration Abstraction Layer
AAP	Anuket Assured Program
AES	Advanced Encryption Standard
AES-NI	AES New Instructions
AF_XDP	Address Family For XDP
AI	Artificial Intelligence
AICPA	American Institute of Certified Public Accountants
AMF	Access and Mobility management Function
API	Application Programming Interface
ARM	Advanced RISC Machines
AS	Application Server
ASIC	Application-Specific Integrated Circuit
B2B	Business to Business
B2C	Business to Consumer
BIOS	Basic Input Output System
BLOB	Binary Large Object
BMC	Baseband Management Controller

Term	Description
BNG	Broadband Network Gateway
BOOTP	Bootstrap Protocol
BSS	Business Support Systems
CaaS	Container as a Service
CAPEX	Capital Expenditure
CDN	Content Distribution (or Delivery) Network
CI/CD	Continuous Integration / Continuous Deployment
CIFS	Common Internet File System
CIM	Cloud Infrastructure Management
CIS	Center for Internet Security
CIT	Cloud Integrity Tool
CLI	Command Line Interface
CNCF	Cloud Native Computing Foundation
CNF	Cloud Native Network Function
CNI	Container Network Interface
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurements
CSA	Cloud Security Alliance
CSCF	Call Session Control Function
CSP	Cloud Service Provider
CU	Centralised Unit (O-RAN context)
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DBaaS	Data Base as a Service
DC	Data Center
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DNS	Domain Name System
DPDK	Data Plane Development Kit
DPU	Data Processing Unit
DRAM	Dynamic Random Access Memory
DRTM	Dynamic Root of Trust for Measurements
DU	Distributed Unit (O-RAN context)
E2E	End to End
eBPF	Extended Berkley Packet Filter
eMBB	Enhanced Mobile BroadBand
EPA	Enhanced Platform Awareness

Term	Description
ESXi	(VMware) ESX Integrated
eTOM	Enhanced Telecom Operations Map
ETSI	European Telecommunications Standards Institute
EUAG	(Linux Foundation Networking) End User Advisory Group
EUD	End User Device
EULA	End-User License Agreement
EVPN	Ethernet Virtual Private Network
FAT	File Allocation Table
FCAPS	fault, configuration, accounting, performance, security
FC-AL	Fibre Channel Arbitrated Loop
FCIP	Fibre Channel over IP
FPGA	Field Programmable Gate Array
FTTx	Fiber to the x
GB	Giga Byte
GDPR	General Data Protection Regulation
Gi or GiB	Gibibyte (1024^3 bytes)
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRE	Generic Routing Encapsulation
GSM	Global System for Mobile Communications, previously Groupe Speciale Mobile Association
GSMA	GSM Association
GUI	Graphical User Interface
HA	High Availability
HBA	Host Bus Adapter
HCP	Hyperscaler Cloud Providers
HDD	Hard Disk Drive
HEM-clouds	Hybrid, Edge, and Multi-clouds
HEMP	Hybrid, Edge, and Multi-Cloud unified management Platform
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
IaaC	Infrastructure as a Code
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code (or "as a")
ICMP	Internet Control Message Protocol
ID	Identifier

Term	Description
IMS	IP Multimedia Subsystem
IO	Input/Output
IOMMU	Input/Output Memory Management Unit
IOPS	Input/Output per Second
IoT	Internet of Things
IP	Internet Protocol
IPMI	Intelligent Platform Management Interface
IPSec	Internet Protocol Security
iSCSI	Internet Small Computer Systems Interface
IT	Information Technology
ITIL	IT Infrastructure Library
JSON	JavaScript Object Notation
K8s	Kubernetes
KPI	Key Performance Indicator
KVM	Keyboard, Video and Mouse
LAN	Local Area Network
LB	Load Balancer
LBaaS	Load Balancer as a Service
LCM	LifeCycle Management
LDAP	Lightweight Directory Access Protocol
LF	Linux Foundation
LMS	Log Management Service
LTD	Less Trusted Domain
MANO	Management and Orchestration
Mi or MiB	Mebibyte (1024^2 bytes)
ML	Machine Learning
MME	Mobility Management Entity
MPLS	Multi-Protocol Label Switching
MTD	More Trusted Domain
MVNO	Mobile Virtual Network Operator
NAS	Network Attached Storage
NAT	Network Address Translation
NBI	North Bound Interface
NF	Network Function
NFS	Network File System
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	Network Function Virtualisation Orchestrator

Term	Description
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NMS	Network Management System
NPL	Network Programming Language
NPN	Non-Public Network
NPU	Numeric Processing Unit
NR	New Radio (5G context)
NTIA	National Telecommunications and Information Administration
NTP	Network Time Protocol
NUMA	Non-Uniform Memory Access
NVMe	Non-Volatile Memory Express
OAM	Operations, Administration and Maintenance
OCI	Open Container Initiative
OFCS	Offline Charging System
ONAP	Open Network Automation Platform
OOB	Out of Band
OPEX	Operational Expenditure
OPNFV	Open Platform for NFV
O-RAN	Open Radio Access Network
OS	Operating System
OSS	Operational Support Systems
OSSA	OpenStack Security Advisories
OVP	OPNFV Verified Program
OWASP	Open Web Application Security Project
PaaS	Platform as a Service
PCIe	Peripheral Component Interconnect Express
PCI-PT	PCIe PassThrough
PCR	Platform Configuration Register
PF	Physical Function
PLMN	Public Land Mobile Network
PM	Performance Measurements
POD	Point of Delivery
PRD	Permanent Reference Document
PTP	Precision Time Protocol
PXE	Pre-boot Execution Environment
QCOW	QEMU copy-on-write
QEMU	Quick EMUlator
QoS	Quality of Service

Term	Description
R/W	Read/Write
RA	Reference Architecture
RADOS	Reliable Autonomic Distributed Object Store
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAN	Radio Access Network
RAW	Raw disk format
RBAC	Role-bases Access Control
RC	Reference Conformance
RFC	Request for Comments
RI	Reference Implementation
RM	Reference Model
RTM	Root of Trust for Measurements
RTT	Round Trip Time
RU	Radio Unit (O-RAN context)
S3	(Amazon) Simple Storage Service
SA	Service Assurance
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial AT Attachment
SBA	Service Based Architecture
SBC	Session Border Controller
SBI	South Bound Interface
SBOM	Software Bill of Materials
SCAP	Security Content Automation Protocol
SDN	Software-Defined Networking
SDNC	SDN Controller
SDNo	SDN Overlay
SDNu	SDN Underlay
SDO	Standard Development Organisation
SDS	Software-Defined Storage
SD-WAN	Software Defined Wide Area Network
Sec-GW	Security GateWay
SF	Service Function
SFC	Service Function Chaining
SFF	Service Function Forwarder
SFP	Service Function Path
SIP	Session Initiation Protocol

Term	Description
SLA	Service Level Agreement
SMF	Session Management Function
SMT	Simultaneous multithreading
SNMP	Simple Network Management Protocol
SOC	System and Organization Controls
SONiC	Software for Open Networking in the Cloud
SR-IOV	Single Root Input Output Virtualisation
SRTM	Static Root of Trust for Measurements
SSD	Solid State Drive
SSDF	Secure Software Development Framework
SSH	Secure SHell protocol
SUT	System Under Test
SW	Software
TCDI	Trusted Cross-Domain Interface
TCP	Transmission Control Protocol
TEC	Telco Edge Cloud
TIP	Telecom Infra Project
TLB	Translation Lookaside Buffers
TLS	Transport Layer Security
TOR	Top of Rack
TOSCA	Topology and Orchestration Specification for Cloud Applications
TPM	Trusted Platform Module
UDR	Unified Data Repository
UEFI	Unified Extensible Firmware Interface
UI	User Interface
UPF	User Plane Function
uRLLC	Ultra-Reliable Low-Latency Communications
V2I	Vehicle to Infrastructure
V2N	Vehicle to Network
V2P	Vehicle to Pedestrian
V2V	Vehicle to Vehicle
V2X	Vehicle-to-everything
VA	Virtual Application
VAS	Value Added Service
vCPU	Virtual CPU
VF	Virtual Function
VI	Vendor Implementation
VIM	Virtualised Infrastructure Manager

Term	Description
VLAN	Virtual LAN
VM	Virtual Machine
VMDK	VMware Virtual Machine Disk File
vNAS	virtual Network Attached Storage
VNF	Virtualised Network Function
VNFC	Virtualised Network Function Component
VNFM	Virtualisedl Network Function Manager
VNI	VXLAN Network Identifier
vNIC	Virtual Network Interface Card
VPN	Virtual Private Network
vRAN	Virtualised Radio Access Network
VTEP	Virtual Termination End Point
VxLAN	Virtual Extensible LAN
vXYZ	virtual XYZ, e.g., as in vNIC
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
XDP	eXpress Data Path
XML	eXtensible Markup Language
ZAP	Zed Attack Proxy
ZTA	Zero Trust Architecture

The key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional", in this document, are to be interpreted according to the descriptions in RFC2119 [1].

1.8 References

Ref	Doc Number	Title
[1]	RFC 2119	{Key words for use in RFCs to Indicate Requirement Levels}
[2]	RFC 5905	{Network Time Protocol Version 4: Protocol and Algorithms Specification}
[3]	RFC 5906	{Network Time Protocol Version 4: Autokey Specification}
[4]	RFC 5907	{Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)}
[5]	RFC 5908	{Network Time Protocol (NTP) Server Option for DHCPv6}
[6]	RFC 2544	{Benchmarking Methodology for Network Interconnect Devices}
[7]	ETSI GR NFV 003 V1.5.1	{Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV}

[8]	ETSI GS NFV 002 V1.2.1	{Network Functions Virtualisation (NFV); Architectural Framework}
[9]	ETSI GR NFV-IFA 029 v3.3.1	{Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS"}
[10]	ETSI GS NFV-TST 008 v3.2.1	{Testing; NFVI Compute and Network Metrics Specification}
[11]	ETSI GS NFV-IFA 027 v2.4.1	{Management and Orchestration; Performance Measurements Specification}
[12]	ETSI GS NFV-IFA 002 v2.1.1	{Acceleration Technologies; VNF Interfaces Specification}
[13]	ETSI NFV-IFA 019 V3.1.1	{Network Functions Virtualisation (NFV); Acceleration Technologies; Acceleration Resource Management Interface Specification}
[14]	ETSI GS NFV-INF 004 V1.1.1	{Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain}
[15]	ETSI GS NFV-IFA 005 V3.1.1	{Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification}
[18]	ORAN-WG4.IOT.0-v01.00	{Front haul Interoperability Test Specification (IOT)}
[19]	ETSI GS NFV-TST 009 V3.1.1	{Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI}
[20]	ETSI GR NFV IFA-012	{Management and Orchestration; Report on Os-Ma-Nfvo reference point - application and service management use cases and recommendations}
[21]	ETSI GS NFV-SEC 001 V1.1.1	{Network Functions Virtualisation (NFV); NFV Security; Problem Statement}
[22]	ETSI GS NFV-SEC 003 V1.1.1	{NFV Security; Security and Trust Guidance}
[23]	ETSI GS NFV-SEC 014 V3.1.1	{NFV Security; Security Specification for MANO Components and Reference points}
[24]	ETSI GS NFV-SEC 013 V2.6.1	{Security; VNF Package Security Specification}
[25]	GSMA FS.31 version 2.0	{Baseline Security Controls}
[28]	ISO/IEC 27000:2018	{Information technology — Security techniques — Information security management systems — Overview and vocabulary}
[30]	NIST SP 800-207	{Zero Trust Architecture (ZTA)}
[32]	IEEE 1588-2019	{Precision Clock Synchronization Protocol for Networked Measurement and Control Systems}
[33]	ITU-T G.8262	{Timing characteristics of a synchronous equipment slave clock}

[34]	ITU-T G.8275.2	{Precision time protocol telecom profile for time/phase synchronization with partial timing support from the network}
[35]	GSMA OPG.02	{Operator Platform Telco Edge Requirements}
[36]	O-RAN.WG6.AAL-GAnP-v01.00	{O-RAN Acceleration Abstraction Layer General Aspects and Principles}
[37]	GSMA FS.40-v02.00	{5G Security Guide}
[38]	ETSI TS 103 457	{CYBER; Trusted Cross-Domain Interface: Interface to offload sensitive functions to a trusted domain}
[40]	ITU-T L.1330	{Energy efficiency measurement and metrics for telecommunication networks}
[41]	ETSI EN 303 471	{Energy Efficiency measurement methodology and KPI/metrics for NFV}
[42]	ETSI ES 203 539	{Measurement method for energy efficiency of Network Functions Virtualisation (NFV) in laboratory environment}
[43]	ITU-T L.1361	{Measurement method for energy efficiency of network functions virtualization}
[44]	GSMA FS.16 v2.2	{Network Equipment Security Assurance Scheme – Development and Lifecycle Security Requirements, version 2.2, available at

Table 1References

1.9 Bibliography

Ref	Document Title	Source
[1]	{DMTF RedFish Specification}	https://www.dmtf.org/sites/default/files/standards/documents/DSP0268_2022.2.pdf
[2]	{NGMN Overview on 5GRAN Functional Decomposition}	https://www.ngmn.org/wp-content/uploads/Publications/2018/180226_NGMN_RANFSX_D1_V20_Final.pdf
[3]	{Open Networking & the Security of Open Source Software Deployment}	https://www.gsma.com/futurenetworks/resources/open-networking-the-security-of-open-source-software-deployment/
[4]	{The Six Pillars of	https://safecode.org/resource-secure-development-practices/the-six-pillars-of-devsecops-automation/

	DevSecOps: Automation}	
[5]	{Information Security Management through Reflexive Security}	https://cloudsecurityalliance.org/artifacts/information-security-management-through-reflexive-security/
[6]	{Edge Computing: Next Steps in Architecture, Design and Testing}	https://www.openstack.org/use-cases/edge-computing/edge-computing-next-steps-in-architecture-design-and-testing/
[7]	{WG6: Cloudification and Orchestration Workgroup specfications}	https://www.o-ran.org
[8]	{Open RAN Technical Priority - Focus on Energy Efficiency}	https://www.o-ran.org/ecosystem-resources
[9]	{QuEST Forum - NFV Workload Efficiency Whitepaper}	https://tl9000.org/resources/documents/NFV%20Workload%20Efficiency%20Whitepaper.pdf
[10]	{Next G Alliance Green G: The path towards sustainable 6G}	https://nextgalliance.org/white_papers/green-g-the-path-towards-sustainable-6g/
[11]	{Glossary of the Anuket Specifications}	https://cntt.readthedocs.io/en/latest/common/glossary.html
[12]	{Abbreviations of the Anuket Specifications}	https://cntt.readthedocs.io/en/latest/common/abbreviations.html
[13]	{ONAP VNF or PNF Requirements Documentation}	https://docs.onap.org/projects/onap-vnfrqts-requirements/en/istanbul/

[14]	{Open Web Application Security Project}	https://owasp.org/
[15]	{Open Web Application Security Project}	https://owasp.org/
[16]	{OWASP Cheat Sheet Series}	https://github.com/OWASP/CheatSheetSeries
[17]	{Information technology — Redfish scalable platforms management API specification}	https://www.iso.org/standard/83853.html
[18]	{Redfish Data Model Specification}	https://www.dmtf.org/sites/default/files/standards/documents/DSP0268_2022.2.pdf
[19]	{Redfish Specification}	https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.8.0.pdf
[20]	{Redfish-Interop-Validator}	https://github.com/DMTF/Redfish-Interop-Validator
[21]	{Redfish Data Model Specification}	https://www.dmtf.org/sites/default/files/standards/documents/DSP0268_2023.2.pdf
[22]	{Redfish Composition Whitepaper}	https://www.dmtf.org/sites/default/files/standards/documents/DSP0268_2023.2.pdf
[23]	{Redfish Telemetry Whitepaper}	https://www.dmtf.org/sites/default/files/standards/documents/DSP2051_1.0.0.pdf
241]	{Precision Time Protocol version 2}	https://standards.ieee.org/ieee/1588/6825/
[25]	{SyncE}	https://www.itu.int/rec/T-REC-G.8262
[26]	{Container Network Interface}	https://github.com/containernetworking/cni
[27]	{Multus}	https://github.com/k8snetworkplumbingwg/multus-cni
[28]	{Network Plumbing	https://github.com/k8snetworkplumbingwg/community

	Working Group}	
[29]	{Kubernetes Network Custom Resource Definition De-facto Standard}	https://docs.google.com/document/d/1Ny03h6IDVy_e_vmElOqR7UdTPAG_RNydhVE1Kx54kFQ/edit
[30]	{P4 Open Source Programming Language}	https://p4.org/
[31]	{OpenConfig}	https://openconfig.net/
[32]	{SONiC}	https://github.com/sonic-net/SONiC
[33]	{NPL}	https://nplang.org/
[34]	{Stratum}	https://opennetworking.org/stratum/
[35]	{Address Family For XDP}	https://www.kernel.org/doc/html/latest/networking/af_xdp.html
[36]	{eXpress Data Path (XDP)}	https://prototype-kernel.readthedocs.io/en/latest/networking/XDP/
[37]	{Extended Berkley Packet Filter}	https://ebpf.io/
[38]	{libbpf}	https://www.kernel.org/doc/html/latest/bpf/libbpf/index.html
[39]	{AF_XDP Poll Mode Driver}	https://doc.dpdk.org/guides/nics/af_xdp.html
[40]	{AF_XDP Device Driver}	https://s3-docs.fd.io/vpp/23.02/developer/devicedrivers/af_xdp.html
[41]	{MITRE ATT&CK® Framework}	https://www.mitre.org/sites/default/files/2021-11/getting-started-with-attack-october-2019.pdf
[42]	{Center for Internet Security}	https://www.cisecurity.org/
[43]	{Cloud Security Alliance}	https://cloudsecurityalliance.org/
[44]	{Open Web Application Security Project}	https://owasp.org/
[45]	{Guide to General	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf

	Server Security}	
[46]	{Building Secure Microservices-based Applications Using Service-Mesh Architecture}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204A.pdf
[47]	{Attribute-based Access Control for Microservices-based Applications Using a Service Mesh}	https://csrc.nist.gov/pubs/sp/800/204/b/final
[48]	{Zero Trust Architecture}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf
[49]	{Secure Software Development Framework (SSDF)}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf
[50]	{FedRAMP Certification}	https://www.fedramp.gov/
[51]	{ETSI Cyber Security Technical Committee (TC CYBER)}	https://www.etsi.org/committee/cyber
[52]	{ETSI Industry Specification Group Network Functions Virtualisation (ISG NFV)}	https://www.etsi.org/technologies/nfv
[53]	{The International Organization for Standardization (ISO) and the International Electrotechnical	https://www.iso.org/home.html

	Commission (IEC)}	
[54]	{GSMA}	https://www.gsma.com/
[55]	{Common Vulnerabilities and Exposures (CVE) databases}	https://cve.mitre.org/
[56]	{MITRE}	https://www.mitre.org/
[57]	{Common Vulnerability Scoring System (CVSS)}	https://www.first.org/cvss/
[58]	{Trusted Boot}	https://sourceforge.net/projects/tboot/
[59]	{Shim}	https://github.com/rhboot/shim
[60]	{Integrity Measurement Architecture (IMA) Wiki}	https://sourceforge.net/p/linux-ima/wiki/Home
[61]	{Open CIT 3.2 Product Guide}	https://github.com/opencit/opencit/wiki/Open-CIT-3.2-Product-Guide
[62]	{Building Secure Microservices-based Applications Using Service-Mesh Architecture}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204A.pdf
[163]	{Attribute-based Access Control for Microservices-based Applications Using a Service Mesh}	https://csrc.nist.gov/pubs/sp/800/204/b/final
[64]	{Aqua Security's Argon Experts Find Software Supply Chain Attacks more than tripled in 2021}	https://www.aquasec.com/news/aqua-securitys-argon-experts-find-software-supply-chain-attacks-more-than-tripled-in-2021/

[65]	{Kaseya ransomware attack sets off race to hack service providers - researchers}	https://www.reuters.com/technology/kaseya-ransomware-attack-sets-off-race-hack-service-providers-researchers-2021-08-03/
[66]	{Codecov hackers breached hundreds of restricted customer sites - sources}	https://www.reuters.com/technology/codecov-hackers-breached-hundreds-restricted-customer-sites-sources-2021-04-19/
[67]	{SolarWinds supply chain attack explained: Need-to-know info}	https://www.techtarget.com/searchsecurity/news/252494495/Defending-against-SolarWinds-attacks-What-can-be-done
[68]	{Apache Log4j Vulnerability Guidance}	https://www.cisa.gov/news-events/news/apache-log4j-vulnerability-guidance
[69]	{Managing Security Risks Inherent in the Use of Third-party Components}	https://safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf
[70]	{Twelve-Factor App}	https://12factor.net/
[71]	{Secure Coding Practices - Quick Reference Guide}	https://owasp.org/
[72]	{Tactical Threat Modelling}	https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf
[73]	{Open Networking & the Security of Open Source Software Deployment - Future Networks}	https://www.gsma.com/futurenetworks/resources/open-networking-the-security-of-open-source-software-deployment/
[74]	{US NTIA}	https://ntia.gov/page/software-bill-materials

[75]	{The Minimum Elements for a Software Bill of Materials (SBOM)}	https://ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf
[76]	{NTIA}	https://ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf
[78]	{SPDX}	https://spdx.dev/
[79]	{SPDX specification 2.2}	https://spdx.dev/use/specifications/
[80]	{CycloneDX}	https://cyclonedx.org/
[81]	{SWID Tags}	https://nvd.nist.gov/products/swid
[82]	{Technical Guide to Information Security Testing and Assessment}	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf
[83]	{American Institute of Certified Public Accountants (AICPA)}	https://us.aicpa.org/content/aicpa/interestareas/frc/assuranceadvisoryservices/sorhome.html
[84]	{Federal Communications Commission}	https://www.fcc.gov/
[85]	{National Institute of Standards and Technology}	https://www.nist.gov/cyberframework/
[86]	{The European Electronic Communications Code}	https://digital-strategy.ec.europa.eu/en/policies/eu-electronic-communications-code
[87]	{The European Union Agency for Cybersecurity}	https://www.enisa.europa.eu/
[88]	{European Cybersecurity Certification Scheme}	https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme/
[89]	{The General Data	https://gdpr-info.eu/

	Protection Regulation}	
[90]	{The Office of Communications}	https://www.ofcom.org.uk/
[91]	{The Security of Networks and Information Systems}	https://www.gov.uk/government/collections/nis-directive-and-nis-regulations-2018/
[92]	{The UK's National Cyber Security Centre}	https://www.ncsc.gov.uk/
[93]	{Data Protection Act 2018}	https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted/
[94]	{Australian Competition & Consumer Commission (ACCC)}	https://www.accc.gov.au/by-industry/telecommunications-and-internet
[95]	{Information Security Manual (ISM)}	https://www.cyber.gov.au/resources-business-and-government/essential-cyber-security/ism
[96]	{Privacy Act 1988}	https://www.oaic.gov.au/privacy/privacy-legislation/the-privacy-act
[97]	{NIST SP 800-53 Security and Privacy Controls for Federal Information Systems and Organizations}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf
[98]	{NIST SP 800-53A Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective	https://serdp-estcp.mil/

	Assessment Plans}	
[99]	{NIST SP 800-63B Digital Identity Guidelines}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf
[100]	{NIST SP 800-115 Technical Guide to Information Security Testing and Assessment}	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf
[101]	{NIST SP 800-125 Guide to Security for Full Virtualization Technologies}	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-125.pdf
[102]	{NIST SP 800-125a Security Recommendations for Server-based Hypervisor Platforms}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-125Ar1.pdf
[103]	{NIST SP 800-125b Secure Virtual Network Configuration for Virtual Machine (VM) Protection}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-125B.pdf
[104]	{NIST SP 800-137 Information Security Continuous Monitoring for Federal Information Systems and Organizations}	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-137.pdf
[105]	{NIST SP 800-145 The NIST Definition of Cloud Computing}	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[105]	{NIST SP 800-190 Application Container Security Guide}	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf
[106]	{CYBER; Trusted Cross-Domain Interface: Interface to offload sensitive functions to a trusted domain}	https://www.etsi.org/deliver/etsi_ts/103400_103499/103457/01.01.01_60/ts_103457v010101p.pdf
[107]	{Jenkins}	https://www.jenkins.io/
[106]	{Gitlab CI/CD}	https://about.gitlab.com/
[107]	{XtestingCI}	https://galaxy.ansible.com/collivier/xtesting
[108]	{CIS Password Policy Guide}	https://www.cisecurity.org/insights/white-papers/cis-password-policy-guide
[109]	{The 18 CIS Critical Security Controls}	https://www.cisecurity.org/controls/cis-controls-list
[110]	{CVE® Program Mission}	https://cve.mitre.org
[111]	{Common Vulnerability Scoring System SIG}	https://www.first.org/cvss/
[112]	{Center for Internet Security}	https://www.cisecurity.org/
[113]	{Open Web Application Security Project}	https://owasp.org/
[114]	{OWASP Top Ten Security Risks}	https://owasp.org/www-project-top-ten/
[115]	{OWASP Software	https://owaspsamm.org/blog/2019/12/20/version2-community-release/

	Maturity Model (SAMM)}	
[116]	{OWASP Web Security Testing Guide}	https://github.com/OWASP/wstg/tree/master/document
[117]	{Information Management Systems specified in ISO/IEC 27001}	https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en
[118]	{Security Controls specified ISO/IEC 27002:2013}	https://www.iso.org/obp/ui/#iso:std:iso-iec:27002:ed-2:v1:en
[119]	{Guidelines for Cybersecurity techniques}	https://www.iso.org/obp/ui/#iso:std:iso-iec:27032:ed-1:v1:en
[120]	{Tekton project}	https://tekton.dev/
[121]	{CNCF Harbor}	https://goharbor.io/

Table 2 Bibliography

1.10 Conventions

The key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional", in this document, are to be interpreted according to the descriptions in RFC2119 [1].

2 Workload Requirements and Analysis

The cloud infrastructure is the totality of all the hardware and software components which build up the environment in which the VNFs and CNFs (workloads) are deployed, managed, and executed. It is, therefore, inevitable that different workloads will require different capabilities and have different expectations from the cloud infrastructure.

Among the main targets of the Reference Model is to define an agnostic cloud infrastructure, to remove any dependencies between the workloads and the deployed cloud infrastructure, and to offer infrastructure resources to the workloads in an abstracted way, with defined capabilities and metrics.

This means that operators will be able to host their Telco workloads (VNFs/CNFs) with different traffic types and behaviours, and will be able to buy from any vendor on a unified consistent cloud infrastructure.

Additionally, a well-defined cloud infrastructure is needed for other types of workloads such as IT, machine learning, and artificial intelligence.

This chapter analyses various Telco workloads and their requirements, and recommends certain cloud infrastructure parameters necessary to specify the desired performance expected by these workloads.

2.1 Workloads collateral

There are different ways in which workloads can be classified, for example:

13. **By function type:**
1. Data plane (also known as user plane, media plane, and forwarding plane)
2. Control plane (also known as signalling plane)
3. Management plane

14. **By service offered:**
1. mobile broadband service
2. fixed broadband service
3. voice service
4. value-added services

15. **By technology:** 2G, 3G, 4G, 5G, IMS, FTTx, Wi-Fi, and so on.

The following list of network functions most likely to be virtualised, covering almost 95% of the Telco workloads, is organised by network segment and function type.

16. **Radio Access Network (RAN)**
1. Data plane
    1. BaseBand Unit (BBU)
    2. Centralised Unit (CU)
    3. Distributed Unit (DU)

17. **2G/3G/4G mobile core network**
    1. Control plane
        1. Mobility Management Entity (MME)
        2. Authentication, Authorisation, and Accounting (3GPP AAA)
        3. Policy and Charging Rules Function (PCRF)
        4. Online Charging system (OCS)
        5. Offline Charging System (OFCS)
        6. Home Subscriber Server (HSS)
        7. Diameter Routing Agent (DRA)
        8. Home Location Register (HLR)
        9. Serving GateWay Control plane (SGW-C)
        10. Packet data network GateWay Control plane (PGW-C)
    2. Data plane
        1. Serving GateWay (SGW)
        2. Serving GateWay User plane (SGW-U)
        3. Packet data network GateWay (PGW)
        4. Packet data network GateWay User plane (PGW-U)
        5. Evolved Packet Data GateWay (ePDG)

6. Mobile Switching Center (MSC)
7. Serving GPRS Support Node (SGSN)
8. Gateway GPRS Support Node (GGSN)
9. Short Message Service Center (SMSC)

18. **5G core network**

5G core nodes are virtualizable by design and are a strong candidate for onboarding into the Telco cloud as cloud-native applications.

1. Data plane
    1. User Plane Function (UPF)
2. Control plane
    1. Access and Mobility management Function (AMF)
    2. Session Management Function (SMF)
    3. Policy Control Function (PCF)
    4. Authentication Server Function (AUSF)
    5. Network Slice Selection Function (NSSF)
    6. Unified Data Management (UDM)
    7. Unified Data Repository (UDR)
    8. Network Repository Function (NRF)
    9. Network Exposure Function (NEF)
    10. Charging Function part of the Converged Charging System (CHF)

19. **IP Multimedia Subsystem (IMS)**

1. Data plane
    1. Media GateWay (MGW)
    2. Session Border Controller (SBC)
    3. Media Resource Function (MRF)
2. Control plane
    1. Call Session Control Function (CSCF)
    2. Mobile Telephony Application Server (MTAS)
    3. Border Gateway Control Function (BGCF)
    4. Media Gateway Control Function (MGCF)

20. **Fixed network**

1. Data plane
    1. MultiService Access Node (MSAN)
    2. Optical Line Termination (OLT)
    3. WLAN Controller (WLC)
    4. Broadband Network Gateway (BNG)
    5. Broadband Remote Access Server (BRAS)
    6. Residential GateWay (RGW)
    7. Customer Premises Equipment (CPE)
2. Control plane
    1. Authentication, Authorisation, and Accounting (AAA)

21. **Other network functions**

1. Data plane
    1. Label Switching Router (LSR)
    2. Deep Packet Inspection (DPI)
    3. Carrier-Grade Network Address Translation (CG-NAT)
    4. Application Delivery Controller (ADC)
    5. Firewall (FW)
    6. Security GateWay (Sec-GW)
    7. Content Delivery Network (CDN)

2. Control plane
    1. Route Reflector (RR)
    2. Domain Name System (DNS)

3. Management plane
    1. Network Management System (NMS)

2.2 Use cases

The intent of this section is to describe some important use cases that are pertinent to this Reference Model. We will start with some typical Edge-related use cases. The list of use cases will be extended in future releases.

Telco Edge is commonly coupled with 5G use cases, seen as one of the ingredients of the Ultra-Reliable Low-latency Communication (URLLC) and Enhanced Mobile Broadband (eMBB) network slicing. The requirements for user plane local breakout/termination are common. They stipulate that value-added services (VASs) and any Gi-LAN applications are locally hosted at the Edge. The Telco Edge is a perfect fit for centralized vRAN deployments and vDU/vCU hosting that satisfy the latency requirements.

It is expected that with the technology evolution (for example, 6G), the use cases will be more demanding. For example, to achieve either less than 1 ms. latency or an ultrafast data rate, it will be required to evolve the architecture. These use cases, once available, can be used for life saving decisions, such as for remote automation in environments not supporting life (for example, in deep space communication), to ensure that the car autonomous driving can be done in real time, and even for holographic communications. Such use cases can be seen as the evolution of 5G use cases, where such requirements cannot be met due to technology constraints.

2.2.1 Use case no. 1: Edge CDN with eMBB core network slicing

2.2.1.1 Business objectives

Monetizing 5G by provisioning eMBB network slices with a distributed content delivery network (CDN) as a service that enables ultra-HD (UHD) streaming, video optimization, caching for large files, and other capabilities that can either be bundled by the network slice offering or implicitly enabled by the operator.

2.2.1.2 Targeted segments

1. B2C: targeting high-tier packages and bundles
2. content owners: potential revenue sharing model
3. mobile virtual network operators (MVNOs): wholesale

4. stadiums and venues

2.2.1.3 Architecture

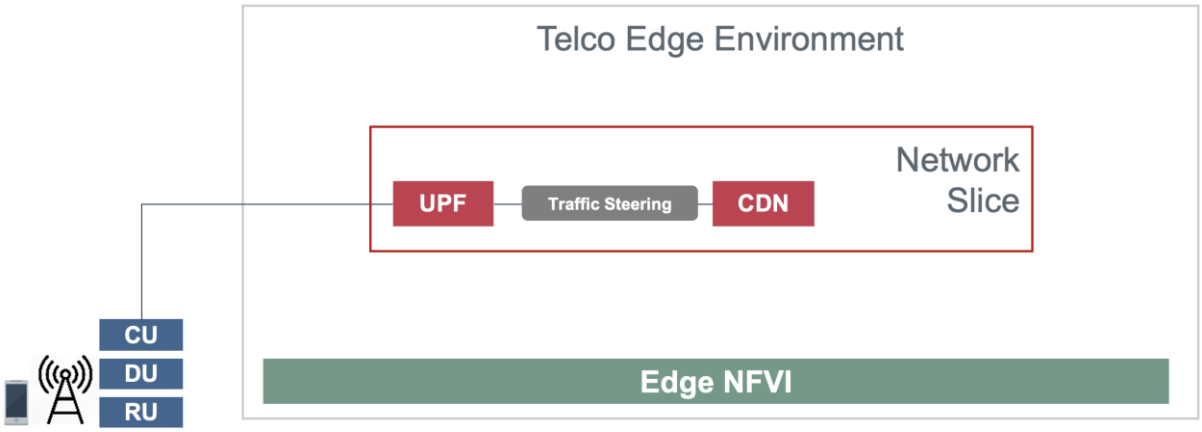


Figure 1Edge CDN with eMBB core network slicing

2.2.2 Use case no. 2: Edge private 5G with core network slicing

2.2.2.1 Business objectives

Private 5G is considered to be one of the most eagerly awaited business use cases in the coming years. It will enable mobile operators to provide a standalone private mobile network to enterprises that may include all the ingredients of the PLMN, such as radio, core, infrastructure, and services covering business requirements in terms of security, performance, reliability, and availability.

2.2.2.2 Targeted segments

1. governmental sectors and public safety (mission-critical applications)
2. factories and the industry sector
3. enterprises with business-critical applications
4. enterprises with strict security requirements, with respect to the reachability of assets
5. enterprises with strict KPI requirements that mandate the on-premise deployment

2.2.2.3 Architecture

1. There are multiple flavours for private 5G deployments or for the non-public network (NPN), as defined by 3GPP.
2. This use case addresses the technical realization of the NPN as a network slice of a PLMN, according to Annex D – 3GPP TS 23.501 R16. It does not cover the other scenarios of deployment.
3. Thise use case assumes a network slice that is constructed from a single UPF deployed on customer premises, while sharing the 5G control plane (AMF, SMF, and other CP network functions) with the PLMN.
4. This use case does not cover the requirements of the private application servers (ASs), as they may vary with each customer setup.
5. Hosting the CU/DU on-customer infrastructure depends on the enterprise offering by the mobile operator and the selected private 5G setup.

6. The Edge cloud infrastructure can be governed by the client or handled by the service provider (mobile operator) as part of managed-services model.

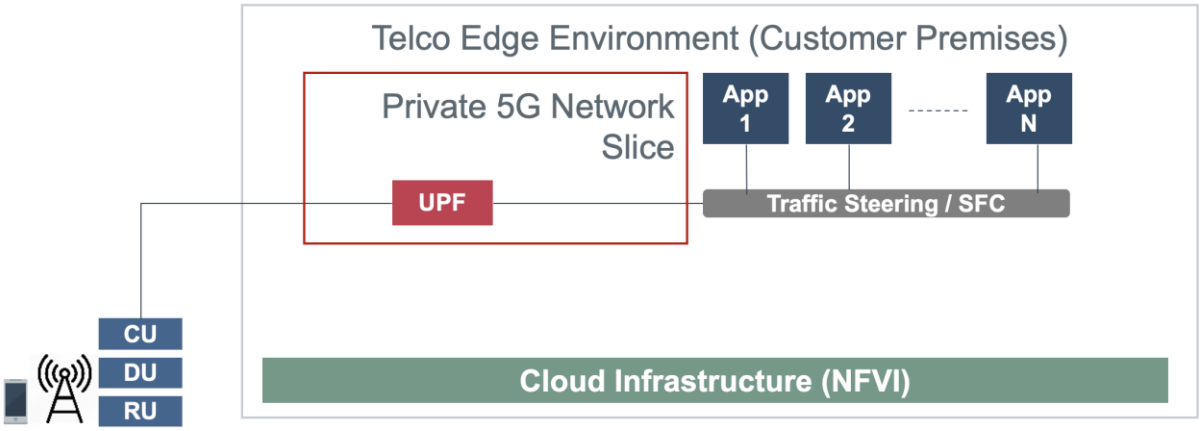


Figure 2Edge private 5G with core network slicing.

2.2.3 Use case no. 3: Edge automotive (V2X) with uRLLC core network slicing

2.2.3.1 Business objectives

The vehicle-to-everything (V2X) set of use cases provides a 5G monetization framework for mobile operators developing 5G URLLC business use cases targeting the automotive industry, smart city regulators, and public safety.

2.2.3.2 Targeted segments

1. the automotive industry
2. governmental departments (smart cities, transport, police, emergency services, and so on)
3. private residences (compounds, hotels, and resorts)
4. enterprise and industrial campuses

2.2.3.3 Architecture

4. 5G NR-V2X is a work item in 3GPP Release 16 that has not been completed at the time of writing this document.

1. Cellular V2X (C-V2X) has two modes of communication:

1. Direct mode, commonly described by Sidelink (SL), by 3GPP: this includes the V2V, V2I, and V2P using a direct interface (PC5) operating in ITS and intelligent transport bands (for example, 5.9 GHZ).
2. Network mode (UL/DL): this covers the V2N while operating in the common telecom-licensed spectrum. This use case capitalizes on this mode.
3. The potential use cases that may consume services from the Edge are the network model (V2N) and potentially the V2I (according to how the infrastructure will be mapped to an Edge level).

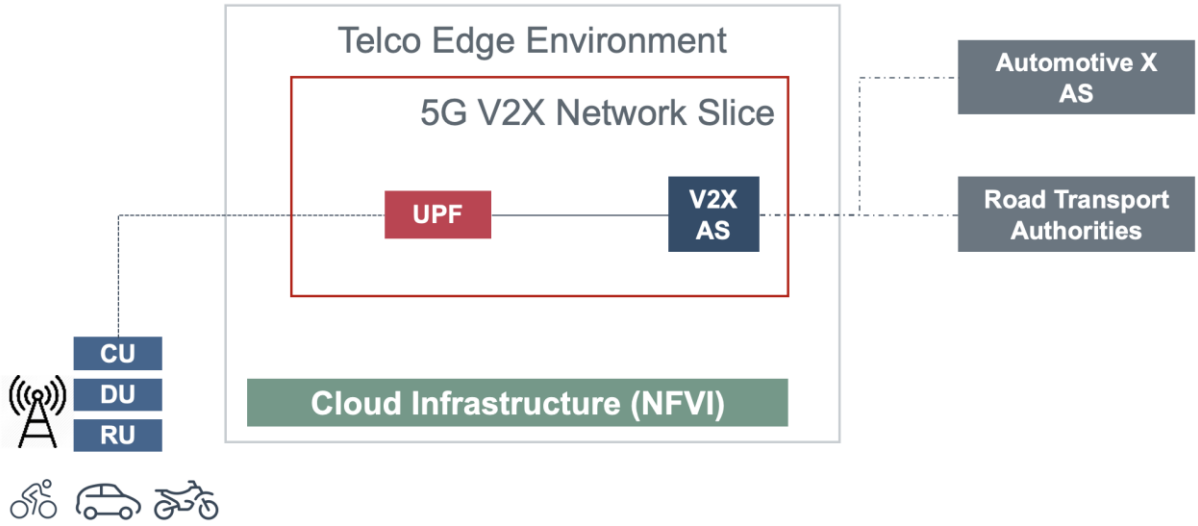


Figure 3Edge automotive (V2X) with uRLLC core network slicing

2.2.4 Use case no. 4: Edge vRAN deployments

2.2.4.1 Business Objectives

vRAN is one of the trending RAN deployment technologies that fits all radio access technologies. vRAN helps to provide coverage for rural and uncovered areas with a compelling CAPEX reduction, compared to traditional and legacy RAN deployments. This coverage can be extended to all area types with 5G greenfield deployments as a typical example.

2.2.4.2 Targeted segments

1. Private 5G customers: vRAN can be part of the non-public network (NPN).
2. B2B customers and MVNOs: vRAN can be part of E2E network slicing.
3. B2C: the mobile consumers segment.

2.2.5 Architecture

1. There are multiple deployment models for the centralized unit (CU) and the distributed unit (DU). This use case covers the placement case of having the DU and the CU co-located and deployed on the Telco Edge. For details, see the NGMN Overview on 5GRAN Functional Decomposition ver. 1.0 [17].
2. This use case covers the 5G vRAN deployment. However, this can be extended to cover 4G vRAN, as well.
3. Following Split Option 7.2, the average market latency for RU-DU (fronthaul) is 100 microseconds to 200 microseconds, while the latency for DU-CU (midhaul) is tens of milliseconds. For details, see ORAN-WG4.IOT.0-v01.00 [18].

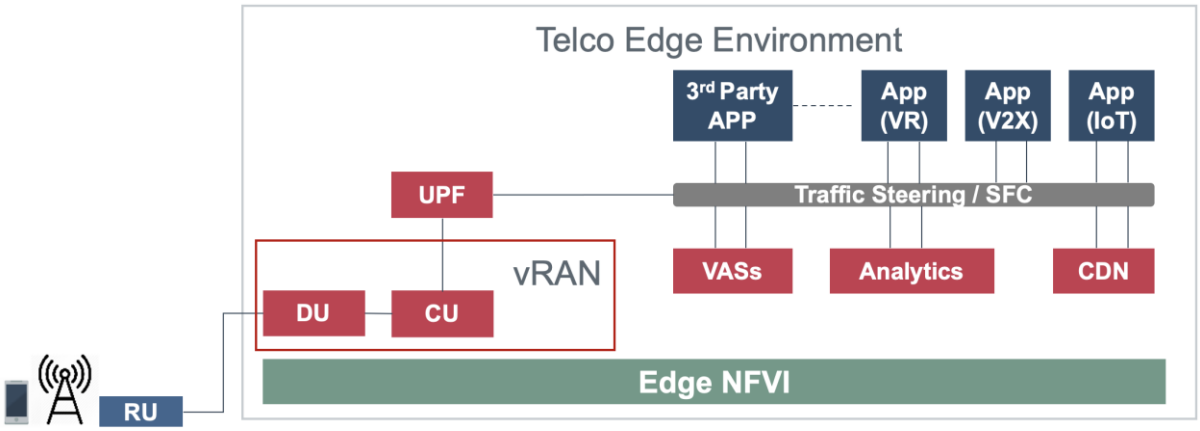


Figure 4Edge vRAN deployments

2.2.6 Use case no. 5: Telepresence experience

2.2.6.1 Business objectives

This service allows communication with one or more persons and creates the impression of being present in the same location without being physically in a virtual environment. This service makes use of eMBB and URLLC network slices, and a distributed deployment which would offload processing.

2.2.6.2 Targeted segments

1. B2B customers and MVNOs
2. B2C: the mobile consumers segment
3. enterprises which use of communication platforms

2.2.6.3 Architecture

1. The architecture takes the form of distributed deployment models across the ecosystem. It should be possible to deploy the workloads at the extreme edge, which would allow real-time processing for video, and would offload processing for network load prediction. This would in turn support the quality of experience that is defined for such a use case.
2. This use case covers the placement at the management plane and the control plane (for example, the Core and the Edge domain).
3. There are high-level requirements for such a use case, such as a latency of 1ms, and an available bandwidth of 8 Gbps.

2.2.7 Use case no. 6: Digital twins for manufacturing

2.2.7.1 Business Objectives

The business objectives are to provide the ability to design and create a product/service as a digital twin which can be tested before moving into the production environment. Therefore, only once acceptance is achieved in the digital world does the service become available. This leads to an extreme reduction of the total cost of ownership (TCO), and minimizes the risks that are commonly associated with the design and testing of a service for the industrial environment.

2.2.7.2 Targeted segments

1. private networks
2. enterprise
3. factories (making extensive use of automation)

2.2.7.3 Architecture

1. This use case demands low latency (less than 1 ms.) and high reliability.
2. The trustworthiness of the feature needs to be guaranteed. This is usually associated with performance, security, and resource efficiency/cost, and, subsequently, productivity.
3. This use case can process massive volumes of data.

2.3 Analysis

Studying the various requirements of the workloads helps us to understand what expectations they will have from the underlying cloud infrastructure. Some of the requirement types on which various workloads may have different expectation levels are set out below:

2.3.1 Computing

1. speed (for example, the CPU clock and the number of physical cores)
2. predictability (for example, CPU and RAM sharing level; real-time pre-emptive scheduling and/or settings for BIOS, kernel and OS services)
3. specific processing (for example, cryptography and transcoding)

2.3.2 Networking

1. throughput (that is, bit rate or packet rate, or both)
2. latency and time-sensitive networking (TSN)
3. platform synchronization (such as GNSS, PTP, and SyncE)
4. the number of connection points or interfaces (that is, vNICs and VLANs)
5. specific traffic control (for example, firewalling, NAT, and cyphering)
6. specific external network connectivity (for example, MPLS and VXLAN)

2.3.3 Storage

1. IOPS (that is, input/output rate or byte rate, or both)
2. volume
3. ephemeral or persistent
4. specific features (for example, object storage and local storage)

In trying to sort workloads into different categories based on the observed requirements, we have identified two different profiles, detailed below. These profiles are mainly driven by the expected performance levels.

2.3.4 Profile One

5. Workload types:

1. control plane functions without specific needs, and management plane functions
2. *examples: OFCS, AAA, and NMS*
3. Requirements:
4. There are no specific requirements.

2.3.5 Profile Two

Workload types

1. data plane functions (that is, functions with specific networking and computing needs)
2. *examples: BNG, S/PGW, UPF, Sec-GW, DPI, CDN, SBC, MME, AMF, IMS-CSCF, and UDR*

Requirements:

1. predictable computing
2. high network throughput
3. low network latency

2.4 Profiles, profile extensions, and flavours

Profiles are used to tag infrastructure, such as hypervisor hosts or Kubernetes worker nodes, and associate it with a set of capabilities that are exploitable by the workloads.

There are two profile *layers*:

- Top-level **profiles**: The top-level profiles represent macro-characteristics that partition the infrastructure into separate pools. This means that an infrastructure object can belong to one profile only, and workloads can only be created using a single profile. Workloads requesting a given profile **must** be instantiated on the infrastructure of that same profile.
- Profile extensions: For a given profile, **profile extensions** represent small variations of the profile, such as infrastructure sizing differences (for example, memory size), that do not require the partitioning of the infrastructure into separate pools, but that have specifications with a finer granularity of the profile. Profile extensions can be *optionally* requested by workloads that want a more granular control over the infrastructure on which they run, that is, an infrastructure resource can have **more than one profile extension label** attached to it, and workloads can request resources to be instantiated on infrastructure with a certain profile extension. Workloads requesting a given profile extension **must** be instantiated on infrastructure with the same profile extension. The operator may instantiate workloads on infrastructure tagged with more profile extensions than requested, as long as the minimum requirements are satisfied.

The workloads specify infrastructure capability requirements as workload metadata, indicating what kind of infrastructure they must run on to achieve functionality or the intended level of performance, or both. The workloads request resources, specifying the profiles and profile extensions, and a set of sizing metadata that may be expressed as flavours that are required for the workload to run as intended. A resource request by a workload can be met by any infrastructure node that has the same or a more specialised profile, and the necessary capacity to support the requested flavour or resource size.

Profiles, profile extensions, and flavours are considered in greater detail in Profile extensions.

2.4.1 Profiles (top-level partitions)

Based on the analysis in Profiles, profile extensions, and flavours, the following cloud infrastructure profiles are as follows (see also Infrastructure profiles based on the categorisation of the VNFs):

Basic: this is for workloads that can tolerate resource over-subscription and variable latency.

High-performance: this is for workloads that require predictable computing performance, high network throughput, and low network latency.

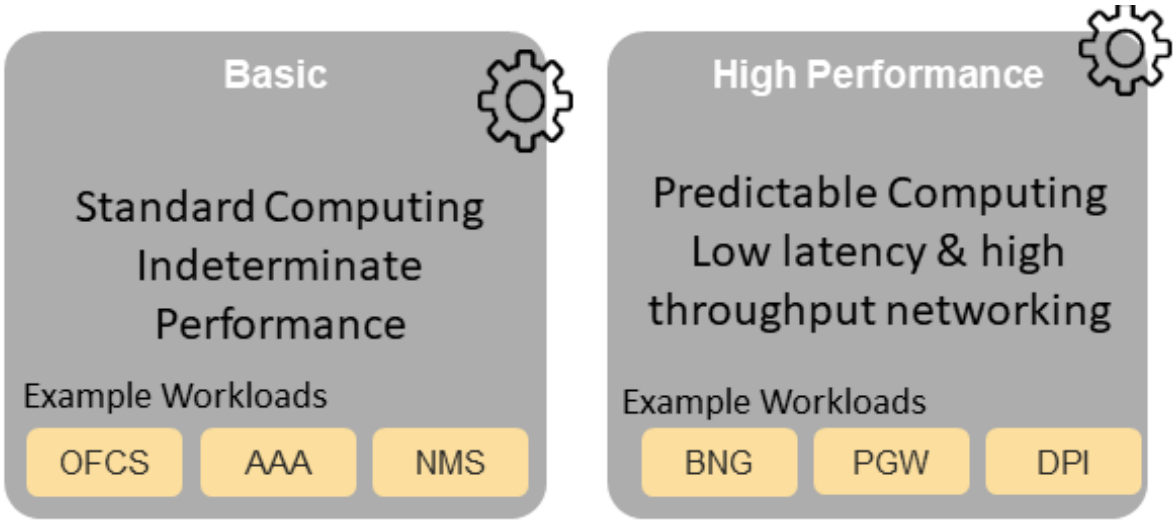


Figure 5Infrastructure profiles based on the categorization of the VNFs

In Infrastructure Capabilities, Measurements and Catalogue these **Basic (B)** and **High-performance (H)** infrastructure profiles are defined in greater detail for use by the workloads.

Profiles partition the infrastructure: an infrastructure object (host/node) **must** have only one profile associated to it.

2.4.2 Profile extensions (specialisations)

Profile extensions are intended to be used as labels for infrastructure. They identify the nodes that implement special capabilities that go beyond the profile baseline. Certain profile extensions may only be relevant for some profiles. The **profile extensions** are detailed in the following table.

Table 3

Profile extension name	Mnemonic	Applicable to the basic profile	Applicable to the high-performance profile	Description	Notes
Compute-intensive high-performance CPU	compute-high-perf-cpu	❌	✅	Nodes that have predictable computing performance and	May use vanilla VIM/K8S

				higher clock speeds.	scheduling instead.
Storage-intensive high-performance storage	storage-high-perf	❌	✅	Nodes that have low storage latency or high storage IOPS, or both.	
Compute-intensive high memory	compute-high-memory	❌	✅	Nodes that have high amounts of RAM.	May use vanilla VIM/K8S scheduling instead.
Compute-intensive GPU	compute-gpu	❌	✅	For compute-intensive workloads that require GPU compute resources on the node.	May use Node Feature Discovery.
Network-intensive high-speed network (25G)	high-speed-network	❌	✅	Denotes the presence of network links (to the DC network) with a speed of 25 Gbps or greater on the node.	
Network-intensive very-high-speed network (100G)	very-high-speed-network	❌	✅	Denotes the presence of network links (to the DC network) with a speed of 100 Gbps or greater on the node.	
Low latency Edge sites	low-latency-edge	✅	✅	Labels a host/node as located in an Edge site, for workloads requiring low latency (specify value), to final users or geographical distribution.	
Exceptionally low latency Edge sites	very-low-latency-edge	✅	✅	Labels a host/node as located in an Edge site, for workloads	

				requiring low latency (specify value), to final users or geographical distribution.	
Ultra-low latency Edge sites	ultra-low-latency-edge	✅	✅	Labels a host/node as located in an Edge site, for workloads requiring low latency (specify value), to final users or geographical distribution.	
Real-time and time-sensitive networking - RAN cell sites	rt-tsn	❌	✅	Labels a host/node configured for Real-Time predictability and Time Sensitive Networking	For example, nodes to run vDU.
Fixed-function accelerator	compute-ffa	❌	✅	Labels a host/node that includes a consumable fixed-function accelerator (non-programmable, such as a Crypto- or vRAN-specific adapter).	
Firmware-programmable adapter	compute-fpga	❌	✅	Labels a host/node that includes a consumable firmware-programmable adapter (programmable, such as a network/storage FPGA with a programmable part of the firmware image).	
SmartNIC enabled	network-smartnic	❌	✅	Labels a host/node that includes a programmable	

				accelerator for vSwitch/vRouter, network function, and/or hardware infrastructure.
SmartSwitch-enabled	network-smartswitch	✗	✓	Labels a host/node that is connected to a programmable switch fabric or a TOR switch.

Table 2-1: Profile extensions

*__Note:__ This is an initial set of proposed profiles and profile extensions. It is expected that more profiles or profile extensions, or both, will be added as more requirements are gathered, and as technology evolves and matures.

3 Modelling

It is necessary to clearly define the infrastructure resources and their capabilities that a shared cloud infrastructure (network function virtualisation infrastructure, NFVI) provides for hosting workloads including virtual network functions (VNFs) and/or cloud-native network functions (CNFs). A common understanding of which resources and their corresponding capabilities a cloud infrastructure provides or shall provide will help improve workload onboarding efficiency and avoid issues that could negatively impact the time and the cost of onboarding and maintaining target workloads and solutions on top of a virtualised infrastructure.

The abstraction model presented in this Reference Model (RM) specifies a common set of virtual infrastructure resources that a cloud infrastructure will need to provide to be able to host most of the typical VNF/CNF telco workloads. The intention of this Reference Model is to follow the following principles:

Scope: the model should describe the most relevant virtualised infrastructure resources (incl. acceleration technologies) a cloud infrastructure needs to host Telco workloads

Separation of Concern: the model should support a clear distinction between the responsibilities related to maintaining the network function virtualisation infrastructure and the responsibilities related to managing the various VNF workloads

Simplicity: the number of different types of resources (including their attributes and relationships amongst one another) should be kept to a minimum to reduce the configuration spectrum which needs to be considered

Declarative: the model should allow for the description of the intended state and configuration of the cloud infrastructure resources for automated life cycle management

Explicit: the model needs to be rich enough to cover the instantiation and the on-going operation of the cloud infrastructure

Lifecycle: the model must distinguish between resources which have independent lifecycles but should group together those resources which share a common lifecycle

Aligned: the model should clearly highlight the dependencies between its components to allow for a well-defined and simplified synchronisation of independent automation tasks.

To summarise: the abstraction model presented in this document will build upon existing modelling concepts and simplify and streamline them to the needs of telco operators who intend to distinguish between infrastructure related and workload related responsibilities.

3.1 Model

The abstraction model for the cloud infrastructure is divided into two logical layers: the virtual infrastructure layer and the hardware infrastructure layer, with the intention that only the virtual infrastructure layer will be directly exposed to workloads (VNFs/CNFs):

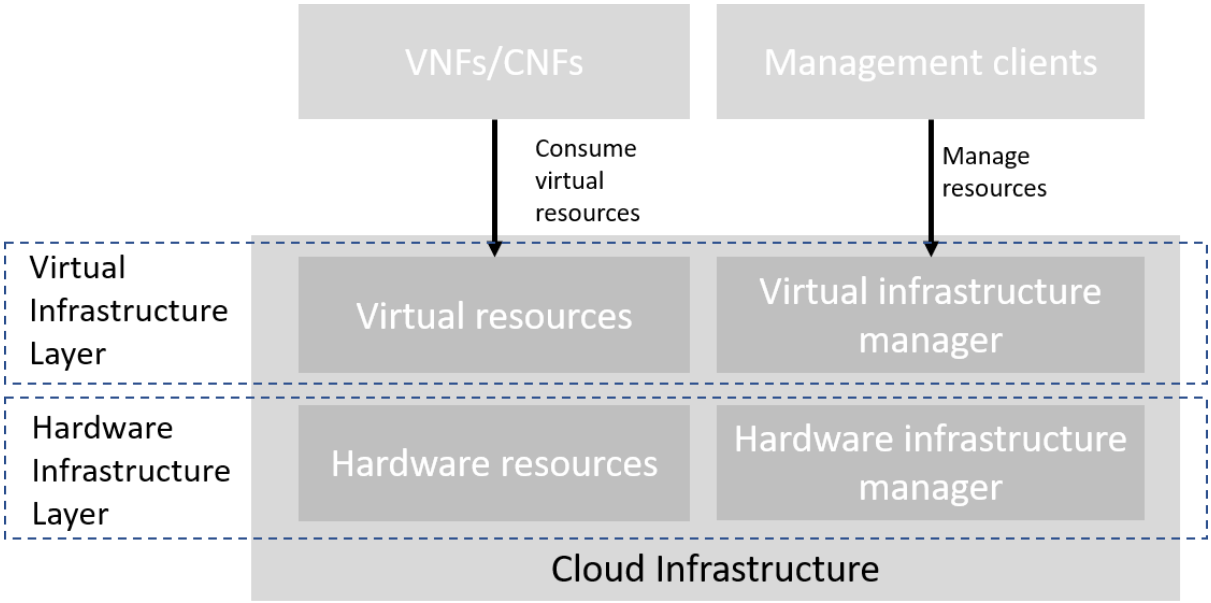


Figure 6Cloud Infrastructure Model Overview

The functionalities of each layer are as follows:

Virtual Infrastructure Layer

Virtual infrastructure resources: These are all the infrastructure resources (compute, storage, and networks) which the cloud infrastructure provides to the VNF/CNF and other workloads. These virtual resources can be managed by the tenants and tenant workloads directly or indirectly via an application programming interface (API).

Virtual infrastructure manager: This consists of the software components that manage the virtual resources and make those management capabilities accessible via one or more APIs. The responsibilities of this functionality include the management of logical constructs such as tenants, tenant workloads, resource catalogues, identities, access controls, security policies, etc.

Hardware Infrastructure Layer

Hardware infrastructure manager: This is a logical block of functionality responsible for the management of the abstracted hardware resources (compute, network, and storage) and as such it is shielded from the direct involvement with server host software.

Hardware resources: These consist of physical hardware components such as servers, (including random access memory, local storage, network ports, and hardware acceleration devices), storage devices, network devices, and the basic input output system (BIOS).

Workload Layer

Workloads (VNFs/CNFs): These consist of workloads such as virtualized and/or containerized network functions that run within a virtual machine (VM) or as a set of containers.

3.2 Virtual Infrastructure Layer

3.2.1 Virtual Resources

The virtual infrastructure resources provided by the Cloud Infrastructure can be grouped into four categories as shown in the diagram below:

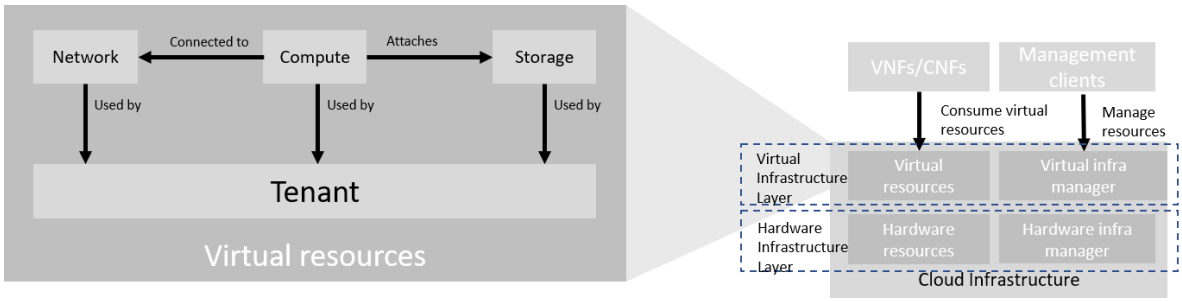


Figure 7Virtual Infrastructure Resources provide virtual computing, storage and networks in a tenant context

Tenants: represent an isolated and independently manageable elastic pool of compute, storage, and network resources

Compute resources: represent virtualised computes for workloads and other systems as necessary

Storage resources: represent virtualised resources for persisting data

Network resources: represent virtual resources providing layer 2 and layer 3 connectivity

The virtualised infrastructure resources related to these categories are listed below.

3.2.1.1 Tenant

A cloud infrastructure needs to be capable of supporting multiple tenants and must isolate sets of infrastructure resources dedicated to specific workloads (VNF/CNF) from one another. Tenants represent an independently manageable logical pool of compute, storage and network resources abstracted from physical hardware.

Example*: a tenant within an OpenStack environment or a Kubernetes cluster.*

Attribute	Description
name	name of the logical resource pool
type	type of tenant (e.g. OpenStack tenant, Kubernetes cluster, …)
vcpus	max. number of virtual CPUs
ram	max. size of random access memory in GB
disk	max. size of ephemeral disk in GB
networks	description of external networks required for inter-domain connectivity
metadata	key/value pairs for selection of the appropriate physical context (e.g. location, availability zone, …)

Table 3-1: Attributes of a tenant

3.2.1.2 Virtual Compute

A virtual machine or a container/pod capable of hosting the application components of workloads (VNFs/CNFs) of the tenant. A virtual compute therefore requires a tenant context and, since it will need to communicate with other communication partners, it is assumed that the networks have been provisioned in advance.

Example*: a virtual compute descriptor as defined in TOSCA Simple Profile for NFV.*

Attribute	Description
name	name of the virtual host
vcpus	number of virtual CPUs
ram	size of random access memory in GB
disk	size of root disc in GB
nics	sorted list of network interfaces connecting the host to the virtual networks
acceleration	key/value pairs for selection of the appropriate acceleration technology
metadata	key/value pairs for selection of the appropriate redundancy domain

Table 3-2: Attributes of compute resources

3.2.1.3 Virtual Storage

A virtual machine and container can consume storage through several means. These include storage that is:

22. managed via the hypervisor and container runtime (Hypervisor Attached for virtual machine and Container Persistent for containers) and is connected via cloud infrastructure underlay network and
23. Shared File Storage and the Object storage which is connected via the tenant / user overlay network. The details of the tenant storage consumption model are covered in section Storage for Tenant Consumption.

In managing the provision of virtual storage the tenant should be able to request alternate performance levels, capacity, and behaviours. The set of selectable attributes includes:

24. Storage class: Block, File, Object.

25. Retention Policy - persistent (storage volume / data) is persistent across stop/start of workload; ephemeral storage -there is no data retention across stop/start events for the workload.
26. Underlying physical device type (HDD, SSD, etc.).
27. Performance characteristic - defined as: Latency, IOPS (Input/Output Operations per second), and throughput.
28. Enhanced features - set of selectable features such as auto-replicate, encryption, snapshot support.

Note that approximate numeric ranges for the qualitative values used above are given in the Storage extensions section.

Storage resources have the following attributes, with metric definitions that support verification through passive measurements (telemetry) where appropriate:

Attribute	Description
name	name of storage resources
data retention policy	persistent or ephemeral
performance	Read and Write Latency, The average amount of time to perform a R/W operation, in milliseconds
	Read and Write IOPS, The average rate of performing R/W in IO operations per second
	Read and Write Throughput, The average rate of performing R/W operations in Bytes per second
enhanced features	replication, encryption
type	block, object or file
size	size in GB, telemetry includes the amount of free, used, and reserved disk space, in bytes

Table 3-3: Attributes of storage resources

3.2.1.4 Virtual Network

This topic is covered in Network section.

3.2.1.5 Availability Zone

An availability zone is a logical pool of physical resources (e.g. compute, block storage, and network). These logical pools segment the physical resources of a cloud based on factors chosen by the cloud operator. The cloud operator may create availability zones based on location (rack, datacentre), or indirect failure domain dependencies like power sources. Workloads can leverage availability zones to use multiple locations or avoid sharing failure domains for a workload, and thus increase the workloads' fault-tolerance.

As a logical group with operator-specified criteria, the only mandatory attribute for an Availability Zone is the name.

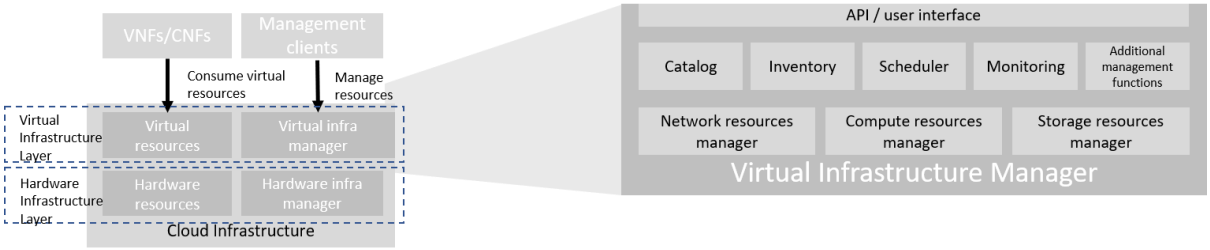
Attribute	Description
name	name of the availability zone

Table 3-4: Attributes of availability zones

3.2.2 Virtual Infrastructure Manager

The virtual infrastructure manager allows:

29. setup, manage and delete tenants,
30. setup, manage and delete user- and service-accounts,
31. manage access privileges and
32. provision, manage, monitor, and delete virtual resources.



Virtual Infrastructure Manager

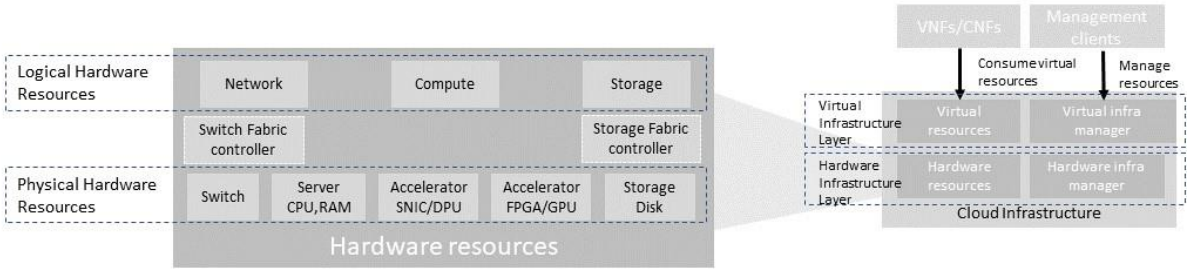
The virtual infrastructure manager needs to support the following functional aspects:

33. **API/UI**: an application programming interface / user interface providing access to the virtual resource management function
34. **Catalogue**: manages the collection of available templates for virtual resource the cloud infrastructure can provide
35. **Inventory**: manages the information related to virtual resources of a cloud infrastructure
36. **Scheduler**: receives requests via API/UI, provisions and manages virtual resources by coordinating the activities of the compute-, storage- and network resources managers
37. **Monitoring**: monitors and collects information on all events and the current state of all virtual resources
38. **Additional Management Functions**: include identity management, access management, policy management (e.g. to enforce security policies), etc.
39. **Compute Resources Manager**: provides a mechanism to provision virtual resources with the help of hardware compute resources
40. **Storage Resources Manager**: provides a mechanism to provision virtual resources with the help of hardware storage resources
41. **Network Resources Manager**: provides a mechanism to provision virtual resources with the help of hardware network resources

3.3 Hardware Infrastructure Layer

3.3.1 Hardware Infrastructure Resources

Compute, Storage and Network resources serve as the foundation of the cloud infrastructure. They are exposed to and used by a set of networked Host Operating Systems in a cluster that normally manages the Virtual Infrastructure Layer offering Virtual Machines or Containers where the application workloads (VNFs/CNFs) runs.



Cloud Infrastructure Hardware Resources

In managed Hardware Infrastructure systems, these consumable Compute, Storage and Network resources can be provisioned through operator commands or through software APIs. There is a need to distinguish between these consumable resources, which are treated as leased resources, from the actual physical hardware resources that are installed in the data centre. For this purpose, the hardware resource layer is conceptually split into a Logical Resource Layer that surfaces the consumable resources to the software layer above, and the Physical Resource Layer that is operated and managed by the Cloud Infrastructure Providers Operations team from the Hardware Infrastructure Management functions perspective.

Some installations might use a cluster of managed switches or storage components controlled by a Switch Fabric controller and/or a Storage Fabric controller acting as an appliance system. These systems should be federated with the HW Infrastructure Management system over some API to facilitate exchange of configuration intent, status and telemetry information allowing the Hardware Infrastructure Management and Management stack to automate Cloud Infrastructure operations. These appliance systems normally also have their own Equipment Management APIs and procedures for the hardware installation and maintenance staff.

An example could be a Cloud Infrastructure stack federated with a commercial Switch Fabric where the Cloud Infrastructure shall be able to "send" networking configuration intent to the Switch Fabric and the Switch Fabric shall be able to "send" (see note below) status and telemetry information to the Cloud Infrastructure e.g. Port/Link Status and packet counters of many sorts. This allows Hardware Infrastructure Management and Cloud Infrastructure management stack to have network automation that includes the switches that are controlled by the federated Switch Fabric. This would be a rather normal case for Operators that have a separate Networking Department that owns and runs the Switch Fabric separately from the Data Centre.

NOTE: The word "send" is a very lose definition of getting a message across to the other side, and could be implemented in many different ways.

3.3.1.1 Hardware Acceleration Resources

For a given software network function and software infrastructure, Hardware Acceleration resources can be used to achieve requirements or improve cost/performance. Following table gives reasons and examples for using Hardware Acceleration.

Reason for using Hardware Acceleration	Example	Comment
Achieve technical requirements	Strict latency or timing accuracy	Must be done by optimizing compute node; cannot be solved by adding more compute nodes
Achieve technical requirements	Fit within power or space envelope	Done by optimizing cluster of compute nodes

Improve cost/performance	Better cost and less power/cooling by improving performance per node	Used when functionality can be achieved through usage of accelerator or by adding more compute nodes

Table 3-5: Reasons and examples for using Hardware Acceleration

Hardware Accelerators can be used to offload software execution for purpose of accelerating tasks to achieve faster performance, or offloading the tasks to another execution entity to get more predictable execution times, efficient handling of the tasks or separation of authority regarding who can control the tasks execution.

More details about Hardware Acceleration are in hardware acceleration abstraction.

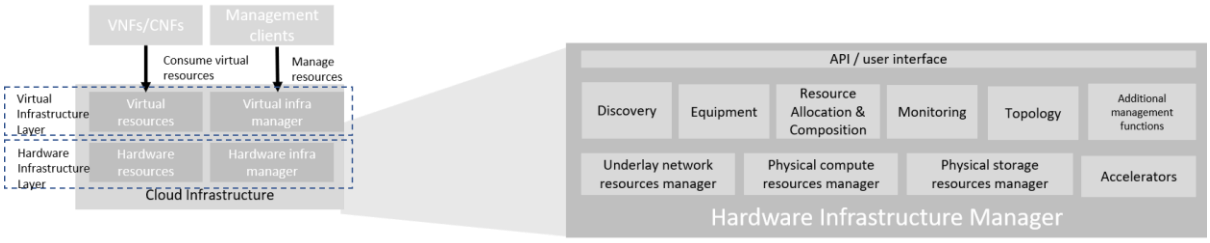
3.3.2 Hardware Infrastructure Manager

The Hardware Infrastructure Manager shall at least support equipment management for all managed physical hardware resources of the Cloud Infrastructure.

In most deployments the Hardware Infrastructure Manager should also be the HW Infrastructure Layer provisioning manager of the Compute, Storage and Network resources that can be used by the Virtualization Infrastructure Layer instances. It shall provide an API enabling vital resource recovery and control functions of the provisioned functions e.g. Reset and Power control of the Computes.

For deployments with more than one Virtualization Infrastructure Layer instance that will be using a common pool of hardware resources there is a need for a HW Infrastructure Layer provisioning manager of the Compute, Storage and Network resources to handle the resource assignment and arbitration.

The resource allocation could be a simple book-keeping of which Virtualization Infrastructure Layer instance that have been allocated a physical hardware resource or a more advanced resource Composition function that assemble the consumed Compute, Storage and Network resources on demand from the pools of physical hardware resources.



Hardware Infrastructure Manager

The hardware infrastructure manager allows to:

42. provision, manage, monitor, and delete hardware resources
43. manage physical hardware resource discovery, monitoring and topology
44. manage hardware infrastructure telemetry and log collection services

The hardware infrastructure manager needs to support the following functional aspects:

45. **API/UI**: an application programming interface / user interface providing access to the hardware resource management functions
46. **Discovery**: discover physical hardware resources and collect relevant information about them
47. **Topology**: discover and monitor physical interconnection (e.g. cables) in between the physical hardware resources

48. **Equipment**: manages the physical hardware resources in terms of configuration, firmware status, health/fault status and autonomous environmental control functions such as fan and power conversion regulations
49. **Resource Allocation and Composition** creates, modifies, and deletes logical Compute, Network and Storage Resources through Composition of allocated physical hardware resources
50. **Underlay Network Resources Manager**: provides a mechanism to provision hardware resources and provide separation in between multiple Virtualization Infrastructure instances for the use of the underlay network (e.g. switch fabric, switches, SmartNICs)
51. **Monitoring**: monitors and collects information on events, current state and telemetry data of physical hardware resources, autonomous equipment control functions as well as Switch and Storage Fabric systems
52. **Additional Management Functions**: include software and configuration life cycle management, identity management, access management, policy management (e.g. to enforce security policies), etc.

3.4 The Redfish® Standard based Hardware Infrastructure Manager

This section proposes a Redfish based hardware infrastructure manager for the Anuket project. Open Distributed Infrastructure Management (ODIM) is an open source software platform that delivers means of distributed physical infrastructure lifecycle management based on the industry standard DMTF Redfish API and Model specification.

Chapter 9 (Configuration and lifecycle management) of this reference model specifies Redfish standard for managing Infrastructure Hardware. The GSMA's "Cloud Infrastructure Reference Model (NG.126)" also specifies Redfish as the standard interface that should be made available by the infrastructure and Cloud Infrastructure Management components, in the "Infrastructure Hardware" layer.

Redfish is an internationally recognized standard ISO/IEC 30115:2018 [54]. The Redfish interface specifies a HTTP RESTful interface that a client can use to manage conformant platforms. The Redfish standard consists of a Redfish interface specification Redfish Specification [56] and model specification Redfish Data Model Specification [55]. The interface specification defines the RESTful behaviour of the resources. The data model specification defines the structure of the HTTP resources. The model is expressed as schema using OpenAPI and json-schema formats. The schema allows implementing redfish clients using the OpenAPI and json-schema toolchains.

There are several opensource implementations of Redfish Clients and Redfish Services. A refish resource aggregator could implement a scalable infrastructure manager which aggregates and exposes a standards-based Redfish interface and services northbound to an Orchestrator.

DMTF specifies the Redfish interface and model, seeking to expand the manageability domains. The prescription of what subset of the Redfish model needs to be implemented for a specific manageability domain is left to other standards bodies. Redfish had defined a JSON syntax for the prescription call a Redfish Profile and implemented an application which reads the Redfish Profile and tests for conformance of an implementation Interop Validator [57]. The Redfish Profile mechanism is being utilized by the Open Compute Project and Open Process Automation Forum.

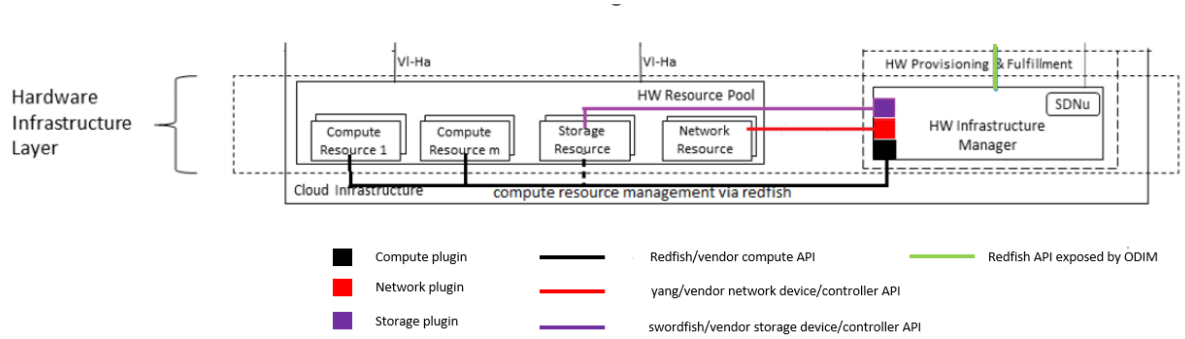
Redfish fulfils the following requirements stated in the above HW Infrastructure Manager section:

53. **API/UI**: Redfish exactly specifies a programming interface. The HTTP interface is accessible remotely via IP and locally via the Ethernet loopback mechanism.

54. **Discovery**: The Redfish schema provides a physical hardware resource model including relevant information
55. **Topology**: The Redfish schema contains a cable model
56. **Equipment**: The Redfish schema contains a physical model and a functional model. The physical model expresses the chassis and containers and interconnects between them and environmental control functions. The functional model expresses the logical aspects and includes configuration and firmware status. Both the physical and functional model have their own health/fault status information
57. **Resource Allocation and Composition**: The Redfish schema has a composition model through which a client can compose a logical resource by allocating physical resources. Redfish Composition Whitepaper [59].
58. **Underlay Network Resources Manager**: The Redfish schema has models for fabrics, switches, and SmartNICs.
59. **Monitoring**: The Redfish schema contains event model for the client to receive hardware events and telemetry model for collecting information across the entire model (physical and functional) Redfish Telemetry Whitepaper [60]
60. **Additional Management Functions**: The Redfish schema has models for access management and identity management Redfish Data Model Specification [58].

3.5 How Redfish fits into the ETSI NFV Networking Reference Model

As shown in the figure below a redfish resource aggregator can play the role of Hardware Infrastructure Manager in the ETSI NFV Networking Reference Model. Whereas this resource manager would expose a Redfish interface to the northbound, the infrastructure pieces can themselves be managed using plugins.



The plugins may manage compute, storage, and network devices from multiple vendors.

3.6 Network

Networking, alongside Compute and Storage, is an integral part of the Cloud Infrastructure (Network Function Virtualisation Infrastructure). The general function of networking in this context is to provide the connectivity between various virtual and physical resources required for the delivery of a network service. Such connectivity may manifest itself as a virtualised network between VMs and/or containers (e.g. overlay networks managed by SDN controllers, and/or programmable network fabrics) or as an integration into the infrastructure hardware level for offloading some of the network service functionality.

Normalization of the integration reference points between different layers of the Cloud Infrastructure architecture is one of the main concerns. In the networking context the primary focus is directed on the packet flow and control flow interfaces between the virtual resources (referred to as Software (SW) Virtualisation Layer) and physical resources (referred to as Hardware (HW) Infrastructure Layer), as well as on related integration into the various MANO

reference points (hardware/network infrastructure management, orchestration). The identification of these two different layers (SW Virtualisation Layer and HW Infrastructure Layer) remains in alignment with the separation of resources into virtual and physical resources, generally used in this document, see e.g., Cloud Infrastructure Model Overview. The importance of understanding the separation of concerns between SW Virtualisation Layer and HW Infrastructure Layer is important because without it, the cardinality of having multiple CaaS and IaaS instances executing on their own private virtual resources from the single shared HW Infrastructure Layer cannot be expressed into separate administrative domains.

3.6.1 Network Principles

Principles that should be followed during the development and definition of the networking scope for the Reference Model, Reference Architectures, Reference Implementations and Reference Conformance test suites:

61. Abstraction: A standardized network abstraction layer between the Virtualisation Layers and the Network Physical Resources Layer that hides (or abstracts) the details of the Network Physical resources from the Virtualisation Layers.
62. Agnosticism: Define Network Fabric concepts and models that can carry any type of traffic in terms of:
    1. Control, User and Management traffic types
    2. Acceleration technologies that can support multiple types of infrastructure deployments and network function workloads
63. Automation: Enable end-to-end automation, from Physical Fabric installation and provisioning to automation of workloads (VNF/CNF) onboarding.
64. Openness: All networking is based on open source or standardized APIs (North Bound Interfaces (NBI) and South Bound Interfaces (SBI)) and should enable integration of open source networking components such as SDN controllers.
65. Programmability: Network model enables a programmable forwarding plane controlled from a separately deployed control plane.
66. Scalability: Network model enables scalability to handle all traffic traverse North-South and East-West enabling small up to large deployments in a non-blocking manner.
67. Workload agnostic: Network model can provide connectivity to any type of workloads, including VNF, CNF and BareMetal workloads.
68. Carrier Grade: Network model is capable of supporting deployments of the carrier grade workloads.
69. Future proof: Network model is extendible to support known and emerging technology trends including SmartNICs, FPGAs and Programmable Switches, integrated for multi-clouds, and Edge related technologies.

3.6.2 Network Layering and Concepts

The Cloud Infrastructure Networking Reference Model is an essential foundation that governs all Reference Architectures and Cloud Infrastructure implementations to enable multiple cloud infrastructure virtualisation technology choices and their evolution. These include:

70. Single Infrastructure as a Service (IaaS) based virtualisation instances with Virtual Machines (VM)
71. Multi IaaS based virtualisation instances
72. Cloud Native Container as a Service (CaaS) based virtualisation instances, and

73. Hybrid multi IaaS and CaaS based virtualisation instances

To retain the cloud paradigms of automation, scalability and usage of shared hardware resources when introducing CaaS instances it is necessary to enable an ability to co-deploy multiple simultaneous IaaS and CaaS instances on a shared pool of hardware resources.

Compute and Storage resources are rarely shared in between IaaS or CaaS instances, but the underpinning networking, most commonly implemented with Ethernet and IP, must be shared and managed as a shared pool of underlay network resources to enable the pooled usage of Compute and Storage from a managed shared pool.

Throughout this chapter and its figures several references to ETSI NFV are made and they explicitly are made towards the ETSI NFV models in the Architectural Framework:

74. ETSI GS NFV 002 V1.2.1 citeetsigsnfv002

75. ETSI GR NFV-IFA 029 V3.3.1 [9]

Cloud and Telco networking are layered, and it is especially important to keep the dependencies between the layers low to enable security, separation and portability in between multiple implementations and generations.

Before we start developing a deep model we need to agree on some foundational concepts and layering that allow decoupling of implementations in between the layers. We will emphasize four concepts in this section:

76. Underlay and Overlay Networking concepts

77. Hardware and Virtual Infrastructure Layer concepts

78. Software Defined Underlay and Overlay Networking concepts

79. Programmable Networking Fabric concept

3.6.2.1 Underlay and Overlay Networking Concepts

The ETSI Network Functions Virtualisation Architectural Framework (as referred above) describes how a Virtual Infrastructure Layer instance abstracts the hardware resources and separates Virtualisation Tenants (Workload) from each other. It does also specifically state that the control and implementation of the hardware layer is out of scope for that specification.

When having multiple Virtual Infrastructure Layer instances on a shared hardware infrastructure, the networking can be layered in an Underlay and an Overlay Network layer. The purpose with this layering is to ensure separation of the Virtualisation Tenants (Workload) Overlay Networks from each other, whilst allowing the traffic to flow on the shared Underlay Network in between all Ethernet connected hardware (HW) devices.

The Overlay Networking separation is often done through encapsulation of Tenants traffic using overlay protocols e.g. through VxLAN or EVPN on the Underlay Networks e.g. based on L2 (VLAN) or L3 (IP) networks.

The Overlay Network for each Cloud Infrastructure deployment must support a basic primary Tenant Network between the Instances within each Tenant. Due to the nature of Telecom applications handling of Networks and their related Network Functions they often need access to external non-translated traffic flows and have multiple separated or secondary traffic channels with abilities for different traffic treatments.

In some instances, the Virtualisation Tenants can bypass the Overlay Networking encapsulation to achieve better performance or network visibility/control. A common method to bypass the Overlay Networking encapsulation normally done by the Virtualisation Layer, is the VNF/CNF usage of SR-IOV that effectively take over the Physical and Virtual Functions of the NIC directly into the VNF/CNF Tenant. In these cases, the Underlay Networking must

handle the separation e.g. through a Virtual Termination End Point (VTEP) that encapsulate the Overlay Network traffic.

Note: Bypassing the Overlay Networking layer is a violation of the basic decoupling principles, but is in some cases unavoidable with existing technologies and available standards. Until suitable technologies and standards are developed, a set of agreed exemptions has been agreed that forces the Underlay Networking to handle the bypassed Overlay Networking separation.

VTEP could be manually provisioned in the Underlay Networking or be automated and controlled through a Software Defined Networking controller interfaces into the underlying networking in the HW Infrastructure Layer.

3.6.2.2 Hardware and Virtual Infrastructure Layer Concepts

The Cloud Infrastructure (based on ETSI NFV Infrastructure with hardware extensions) can be composed of two distinct layers, here referred to as HW Infrastructure Layer and Virtual Infrastructure Layer. When there are multiple separated simultaneously deployed Virtual Infrastructure domains, the architecture and deployed implementations must enable each of them to be in individual non-dependent administrative domains. The HW Infrastructure must then also be enabled to be a fully separated administrative domain from all the Virtualisation domains.

For Cloud Infrastructure implementations of multiple well separated simultaneous Virtual Infrastructure Layer instances on a shared HW Infrastructure there must be a separation of the hardware resources i.e. servers, storage and the Underlay Networking resources that interconnect the hardware resources e.g. through a switching fabric.

To allow multiple separated simultaneous Virtual Infrastructure Layer instances onto a shared switching fabric there is a need to split up the Underlay Networking resources into non overlapping addressing domains on suitable protocols e.g. VxLAN with their VNI Ranges. This separation must be done through an administrative domain that could not be compromised by any of the individual Virtualisation Infrastructure Layer domains either by malicious or unintentional Underlay Network mapping or configuration.

These concepts are very similar to how the Hyperscaler Cloud Providers (HCP) offer Virtual Private Clouds for users of Bare Metal deployment on the HCP shared pool of servers, storage, and networking resources.

The separation of Hardware and Virtual Infrastructure Layers administrative domains makes it important that the Reference Architectures do not include direct management or dependencies of the pooled physical hardware resources in the HW Infrastructure Layer e.g. servers, switches, and underlay networks from within the Virtual Infrastructure Layer. All automated interaction from the Virtual Infrastructure Layer implementations towards the HW Infrastructure with its shared networking resources in the HW Infrastructure Layer must go through a common abstracted Reference Model interface.

3.6.2.3 Software Defined Underlay and Overlay Networking Concepts

A major point with a Cloud Infrastructures is to automate as much as possible. An important tool for Networking automation is Software Defined Networking (SDN) that comes in many different shapes and can act on multiple layers of the networking. In this section we will deal with the internal networking of a datacentre and not how datacentres interconnect with each other or get access to the world outside of a datacentre.

When there are multiple simultaneously deployed instances of the Virtual Infrastructure Layers on the same HW Infrastructure, there is a need to ensure Underlay networking separation in the HW Infrastructure Layer. This separation can be done manually through provisioning of a statically configured separation of the Underlay Networking in the HW Infrastructure Layer. A

better and more agile usage of the HW Infrastructure is to offer each instance of the Virtual Infrastructure Layer a unique instance of a SDN interface into the shared HW Infrastructure. Since these SDN instances only deal with a well separated portion (or slice) of the Underlay Networking we call this interface SDN-Underlay (SDNu).

The HW Infrastructure Layer is responsible for keeping the different Virtual Infrastructure Layer instances separated in the Underlay Networking. This can be done through manual provisioning methods or be automated through a HW Infrastructure Layer orchestration interface. The separation responsibility is also valid between all instances of the SDNu interface since each Virtual Infrastructure Layer instance shall not know about, be disturbed by or have any capability to reach the other Virtual Infrastructure instances.

An SDN-Overlay control interface (here denoted SDNo) is responsible for managing the Virtual Infrastructure Layer virtual switching and/or routing as well as its encapsulation and its mapping onto the Underlay Networks.

In cases where the VNF/CNF bypasses the Virtual Infrastructure Layer virtual switching and its encapsulation, as described above, the HW Infrastructure Layer must perform the encapsulation and mapping onto the Underlay Networking to ensure the Underlay Networking separation. This should be a prioritized capability in the SDNu control interface since Anuket currently allow exemptions for bypassing the virtual switching (e.g. through SR-IOV).

SDNo controllers can request Underlay Networking encapsulation and mapping to be done by signalling to an SDNu controller. There are however today no standardized way for this signalling and because of that there is a missing reference point and API description in this architecture.

Multiple instances of Container as a Service (CaaS) Virtual Infrastructure Layers running on an Infrastructure as a Service (IaaS) Virtual Infrastructure Layer could make use of the IaaS layer to handle the required Underlay Networking separation. In these cases, the IaaS Virtualisation Infrastructure Manager (VIM) could include an SDNu control interface enabling automation.

Note: The Reference Model describes a logical separation of SDNu and SDNo interfaces to clarify the separation of administrative domains where applicable. In real deployment cases an Operator can select to deploy a single SDN controller instance that implements all needed administrative domain separations or have separate SDN controllers for each administrative domain. A common deployment scenario today is to use a single SDN controller handling both Underlay and Overlay Networking which works well in the implementations where there is only one administrative domain that owns both the HW Infrastructure and the single Virtual Infrastructure instance. However a shared Underlay Network that shall ensure separation must be under the control of the shared HW Infrastructure Layer. One consequence of this is that the Reference Architectures must not model collapsed SDNo and SDNu controllers since each SDNo must stay unaware of other deployed implementations in the Virtual Infrastructure Layer running on the same HW Infrastructure.

3.6.2.4 Programmable Networking Fabric Concept

The concept of a Programmable Networking Fabric pertains to the ability to have an effective forwarding pipeline (a.k.a. forwarding plane) that can be programmed and/or configured without any risk of disruption to the shared Underlay Networking that is involved with the reprogramming for the specific efficiency increase.

The forwarding plane is distributed by nature and must be possible to implement both in switch elements and on SmartNICs (managed outside the reach of host software), that both can be managed from a logically centralised control plane, residing in the HW Infrastructure Layer.

The logically centralised control plane is the foundation for the authoritative separation between different Virtualisation instances or Bare Metal Network Function applications that are regarded as untrusted both from the shared layers and each other.

Although the control plane is logically centralized, scaling and control latency concerns must allow the actual implementation of the control plane to be distributed when required.

All VNF, CNF and Virtualisation instance acceleration as well as all specific support functionality that is programmable in the forwarding plane must be confined to the well separated sections or stages of any shared Underlay Networking. A practical example could be a Virtualisation instance or VNF/CNF that controls a NIC/SmartNIC where the Underlay Networking (Switch Fabric) ensures the separation in the same way as it is done for SR-IOV cases today.

The nature of a shared Underlay Network that shall ensure separation and be robust is that all code in the forwarding plane and in the control plane must be under the scrutiny and life cycle management of the HW Infrastructure Layer.

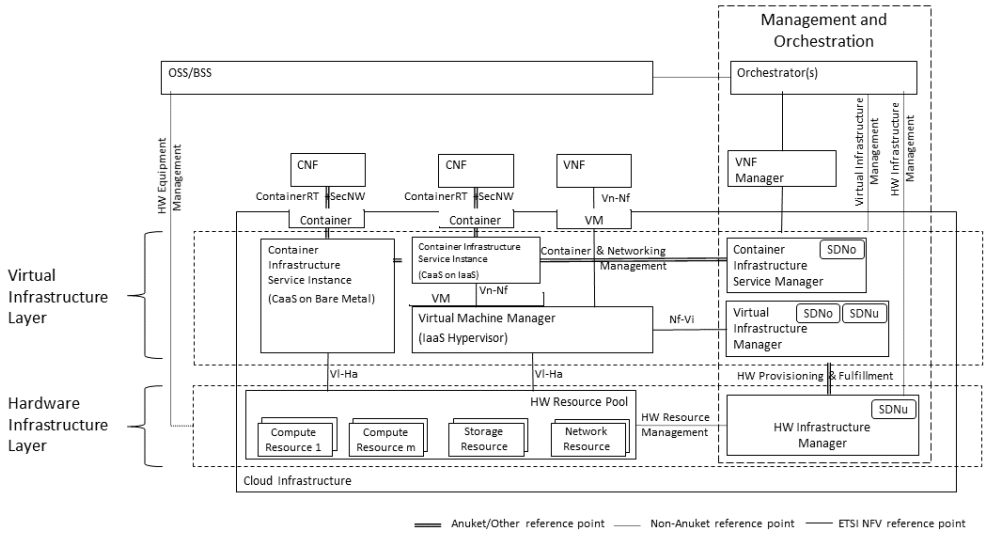
This also implies that programmable forwarding functions in a Programmable Networking Fabric are shared resources and by that will have to get standardised interfaces over time to be useful for multiple VNF/CNF and multi-vendor architectures such as ETSI NFV. Example of such future extensions of shared functionality implemented by a Programmable Networking Fabric could be L3 as a Service, Firewall as a Service and Load Balancing as a Service.

Note: Appliance-like applications that fully own its infrastructure layers (share nothing) could manage and utilize a Programmable Networking Fabric in many ways, but that is not a Cloud Infrastructure implementation and falls outside the use cases for these specifications.

3.6.3 Networking Reference Model

The Cloud Infrastructure Networking Reference Model depicted in Networking Reference Model based on the ETSI NFV is based on the ETSI NFV model enhanced with Container Virtualisation support and a strict separation of the HW Infrastructure and Virtualization Infrastructure Layers in NFVI. It includes all above concepts and enables multiple well separated simultaneous Virtualisation instances and domains allowing a mix of IaaS, CaaS on IaaS and CaaS on Bare Metal on top of a shared HW Infrastructure.

It is up to any deployment of the Cloud Infrastructure to decide what Networking related objects to use, but all Reference Architectures have to be able to map into this model.



Networking Reference Model based on the ETSI NFV

3.6.4 Deployment Examples Based on the Networking Reference Model

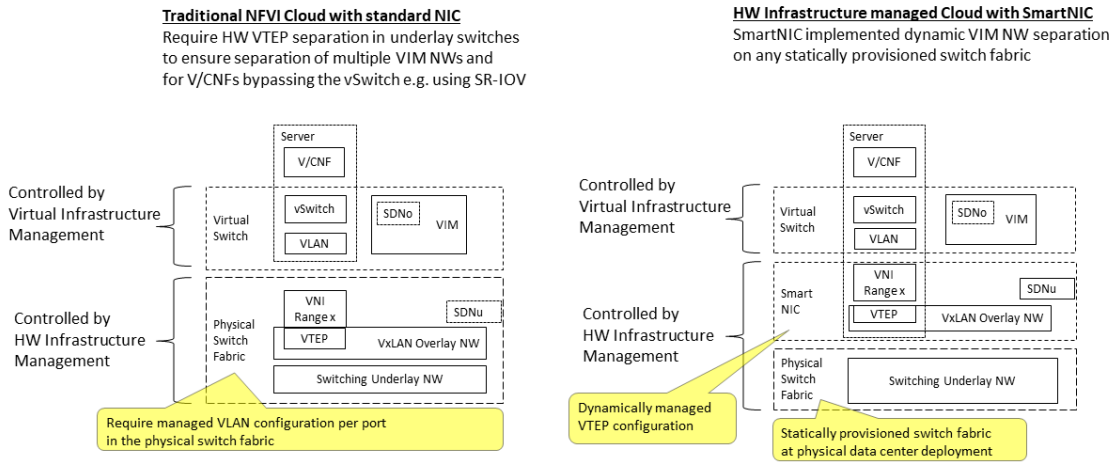
3.6.4.1 Switch Fabric and SmartNIC Examples For Underlay Networking Separation

The Hardware Infrastructure Layer can implement the Underlay Networking separation in any type of packet handling component. This may be deployed in many different ways depending on target use case requirements, workload characteristics and available platforms. Two of the most common ways are: (1) within the physical Switch Fabric and (2) in a SmartNIC connected to the Server CPU being controlled over a management channel that is not reachable from the Server CPU and its host software. In either way the Underlay Networking separation is controlled by the HW Infrastructure Manager.

In both cases the Underlay Networking can be externally controlled over the SDNu interface that must be instantiated with appropriate Underlay Networking separation for each of the Virtualization administrative domains.

Note: The use of SmartNIC in this section is only pertaining to Underlay Networking separation of Virtual instances in separate Overlay domains in much the same way as AWS do with their Nitro SmartNIC. This is the important consideration for the Reference Model that enables multiple implementation instances from one or several Reference Architectures to be used on a shared Underlay Network. The use of SmartNIC components from any specific Virtual instance e.g. for internal virtual switching control and acceleration must be regulated by each Reference Architecture without interfering with the authoritative Underlay separation laid out in the Reference Model.

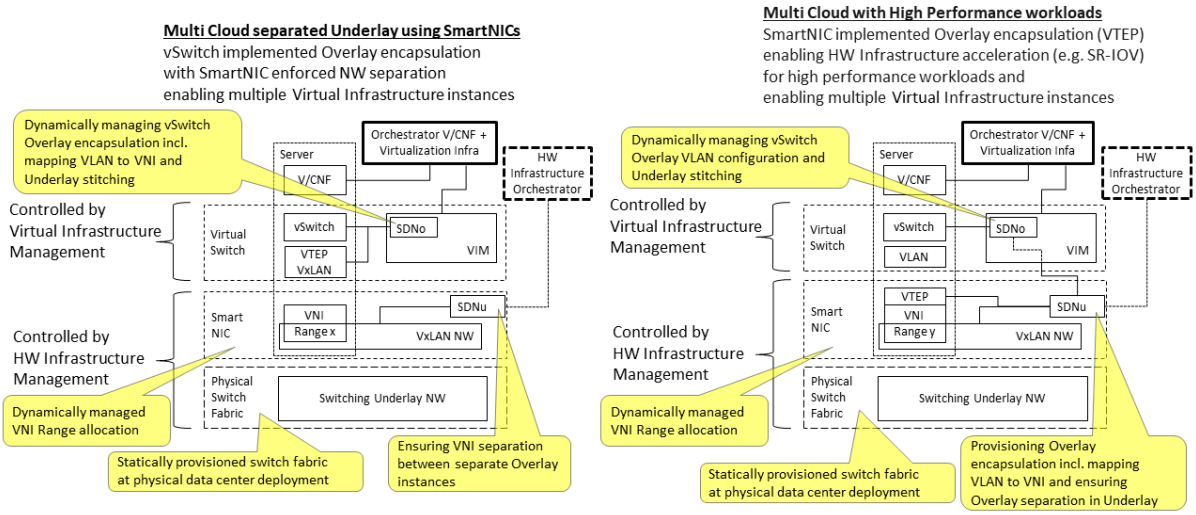
Two exemplifications of different common HW realisations of Underlay Network separation in the HW Infrastructure Layer can be seen in Underlay Networking separation examples below.



Underlay Networking separation examples

3.6.4.2 SDN Overlay and SDN Underlay layering and relationship example

Two use case examples with both SDNo and SDNu control functions depicting a software based virtual switch instance in the Virtual Infrastructure Layer and another high performance oriented Virtual Infrastructure instance (e.g. enabling SR-IOV) are described in SDN Controller relationship examples (below). The examples are showing how the encapsulation and mapping could be done in the virtual switch or in a SmartNIC on top of a statically provisioned underlay switching fabric, but another example could also have been depicted with the SDNu controlling the underlay switching fabric without usage of SmartNICs.

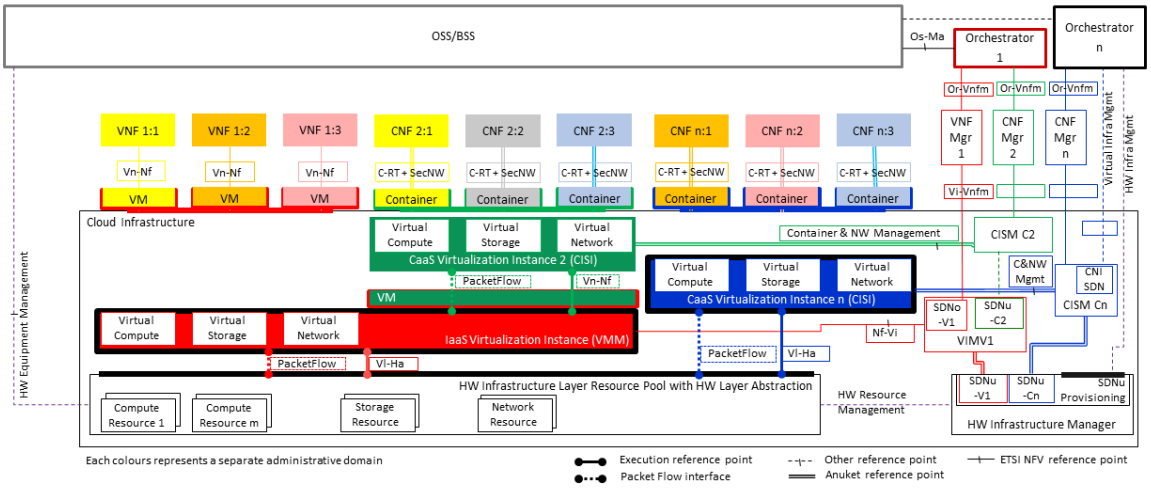


SDN Controller relationship examples

3.6.4.3 Example of IaaS and CaaS Virtualization Infrastructure Instances on a Shared HW Infrastructure With SDN

A Networking Reference Model deployment example is depicted in Networking Reference Model deployment example (below) to demonstrate the mapping to ETSI NFV reference points with additions of packet flows through the infrastructure layers and some other needed reference points. The example illustrates individual responsibilities of a complex organization with multiple separated administrative domains represented with separate colours.

The example is or will be a common scenario for operators that modernise their network functions during a rather long period of migration from VNFs to Cloud Native CNFs. Today the network functions are predominantly VNFs on IaaS environments and the operators are gradually moving a selection of these into CNFs on CaaS that either sit on top of the existing IaaS or directly on Bare Metal. It is expected that there will be multiple CaaS instances in most networks, since it is not foreseen any generic standard of a CaaS implementation that will be capable to support all types of CNFs from any vendor. It is also expected that many CNFs will have dependencies to a particular CaaS version or instances which then will prohibit a separation of Life Cycle Management in between individual CNFs and CaaS instances.



Networking Reference Model deployment example

3.6.5 Service Function Chaining

Over the past few years there has been a significant move towards decomposing network functions into smaller sub-functions that can be independently scaled and potentially reused across multiple network functions. A service chain allows composition of network functions by passing selected packets through multiple smaller services.

To support this capability in a sustainable manner, there is a need to have the capability to model service chains as a high level abstraction. This is essential to ensure that the underlying connection setup, and (re-)direction of traffic flows can be performed in an automated manner. At a very high level a service chain can be considered a directed acyclic graph with the composing network functions being the vertices. Building on top of this, a service chain can be modelled by defining two parameters:

80. An acyclic graph defining the service functions that need to be traversed for the service chain. This allows for multiple paths for a packet to traverse the service chain.

81. A set of packet/flow classifiers that determine what packets will enter and exit a given service chain

These capabilities need to be provided for both virtualised and containerised (cloud-native) network functions as there will be a need to support both for the foreseeable future. Since virtualised network functions have existed for a while there is existing, albeit partial, support for service chaining in virtualised environments in orchestration platforms like OpenStack. Container orchestration platforms such as Kubernetes do not support service chaining and may require development of new primitives to support advanced networking functions.

It is expected that reference architectures will provide a service chain workflow manager that would accept the service function acyclic graph and be able to identify/create the necessary service functions and the networking between them to instantiate such a chain.

There is also a need to provide specialised tools to aid troubleshooting of individual services and the communication between them in order to investigate issues in the performance of composed network functions. Minimally, there is a need to provide packet level and byte level counters and statistics as the packets pass through the service chain in order to ascertain any issues with forwarding and performance. Additionally, there is a need for mechanisms to trace the paths of selected subsets of traffic as they flow through the service chain.

3.6.5.1 Service Function Chaining Model Introduction

Service Function Chaining (SFC) can be visualized as a layered structure where the Service Function plane (SFC data plane, consists of service function forwarder, classifier, service function, service function proxy) resides over a Service Function overlay network. SFC utilizes a service-specific overlay that creates the service topology. The service overlay provides service function connectivity built "on top" of the existing network topology. It leverages various overlay network technologies (e.g., Virtual eXtensible Local Area Network (VXLAN)) for interconnecting SFC data-plane elements and allows establishing Service Function Paths (SFPs).

In a typical overlay network, packets are routed based on networking principles and use a suitable path for the packet to be routed from a source to its destination.

However, in a service-specific overlay network, packets are routed based on policies. This requires specific support at network level such as at CNI in CNF environment to provide such specific routing mechanism.

3.6.5.2 SFC Architecture

The SFC Architecture is composed of functional management, control, and data components as categorised in the Table 3-6 below.

The table below highlights areas under which common SFC functional components can be categorized.

Table 4SFC Architecture Components

Components	Example	Responsibilities
Management	SFC orchestrator	High Level of orchestrator Orchestrate the SFC based on SFC Models/Policies with help of control components.
Management	SFC OAM Components	Responsible for SFC OAM functions
Management	VNF MANO	NFVO, VNFM, and VIM Responsible for SFC Data components lifecycle

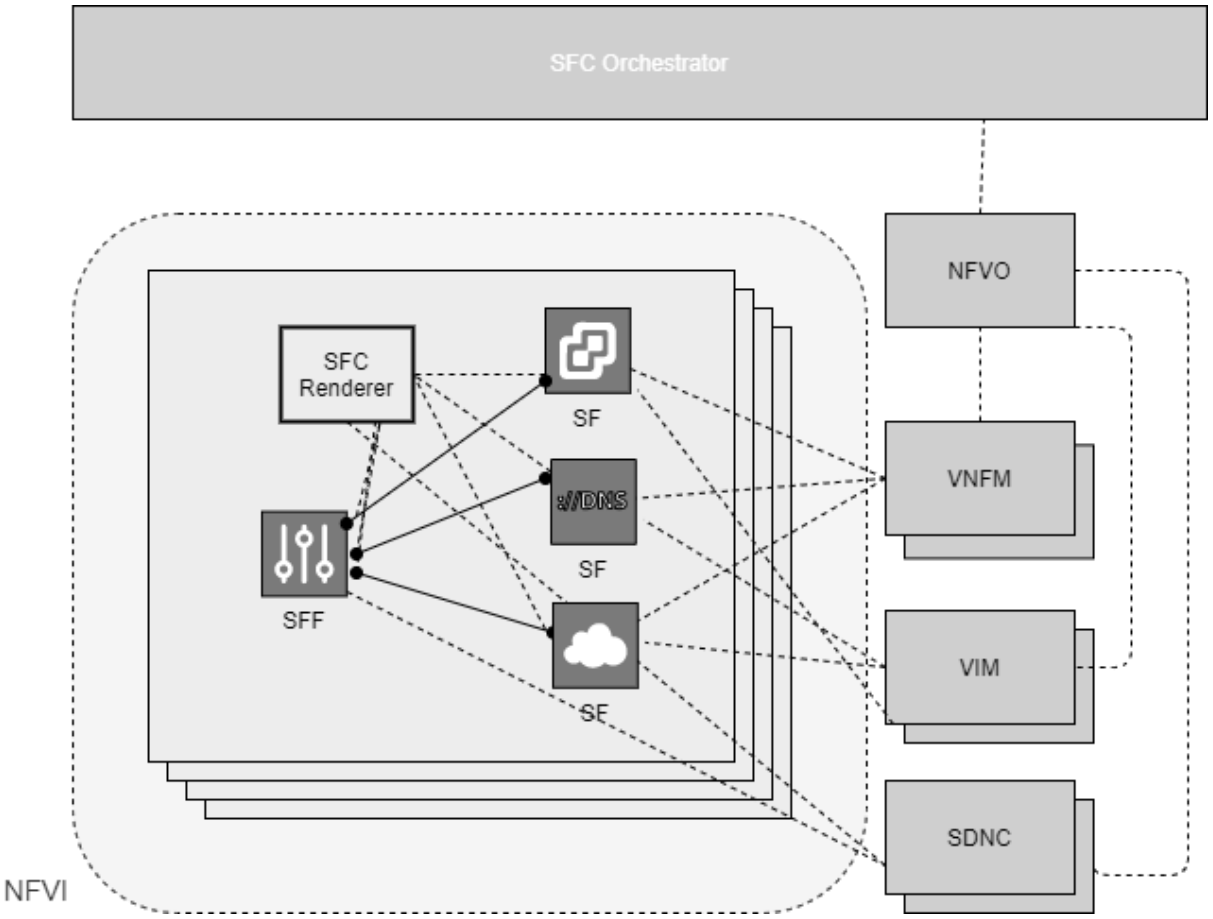
Management	CNF MANO	CNF DevOps Components Responsible for SFC data components lifecycle
Control	SFC SDN Controller	SDNC responsible to create the service specific overlay network. Deploy different techniques to stitch the wiring but provide the same functionality, for example l2xconn, SRv6, Segment routing etc.
Control	SFC Renderer	Creates and wires ports/interfaces for SF data path
Data	Core ComponentsSF, SFF, SF Proxy	Responsible for steering the traffic for intended service functionalities based on Policies

Table 3-6: SFC Architecture Components

Note: These are logical components and listed for their functionalities only.

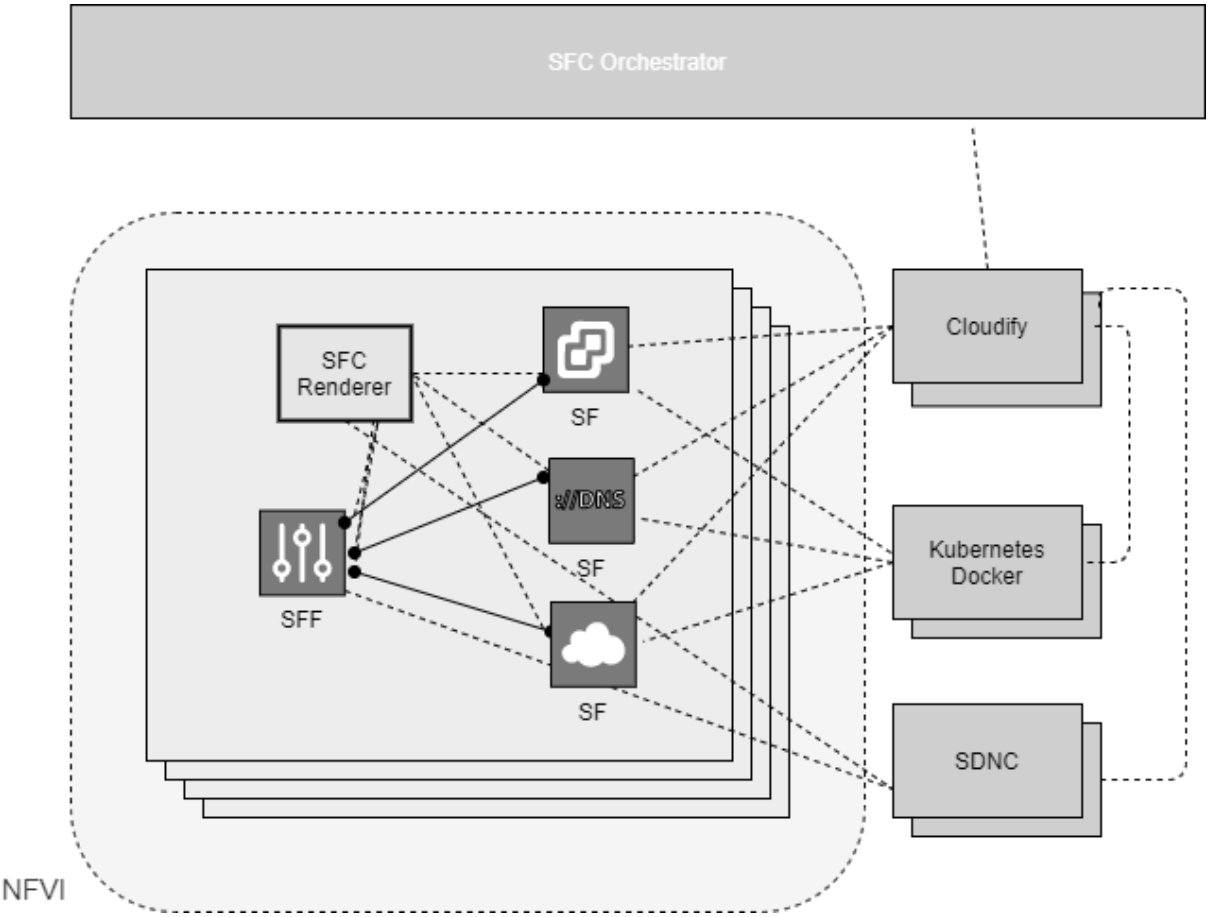
The SFC Architecture components can be viewed as:-

SFC Architecture for VNF based SFs shows a simple architecture of an SFC with multiple VNFs, as SF data plane components, along with SFC management and NFV MANO components.



SFC Architecture for VNF based SFs

SFC Architecture for CNF based SFs shows a simple architecture of an SFC with multiple CNFs, as SF data plane components, along with SFC management and CNF MANO components.



SFC Architecture for CNF based SFs

The SFC management components together with the control components are responsible for rendering SFC requests to Service Function paths. For this they convert requisite SFC policies into network topology dependent paths and forwarding steering policies. Relevant SFC data components - classifiers, service function forwarders - are responsible for managing the steering policies.

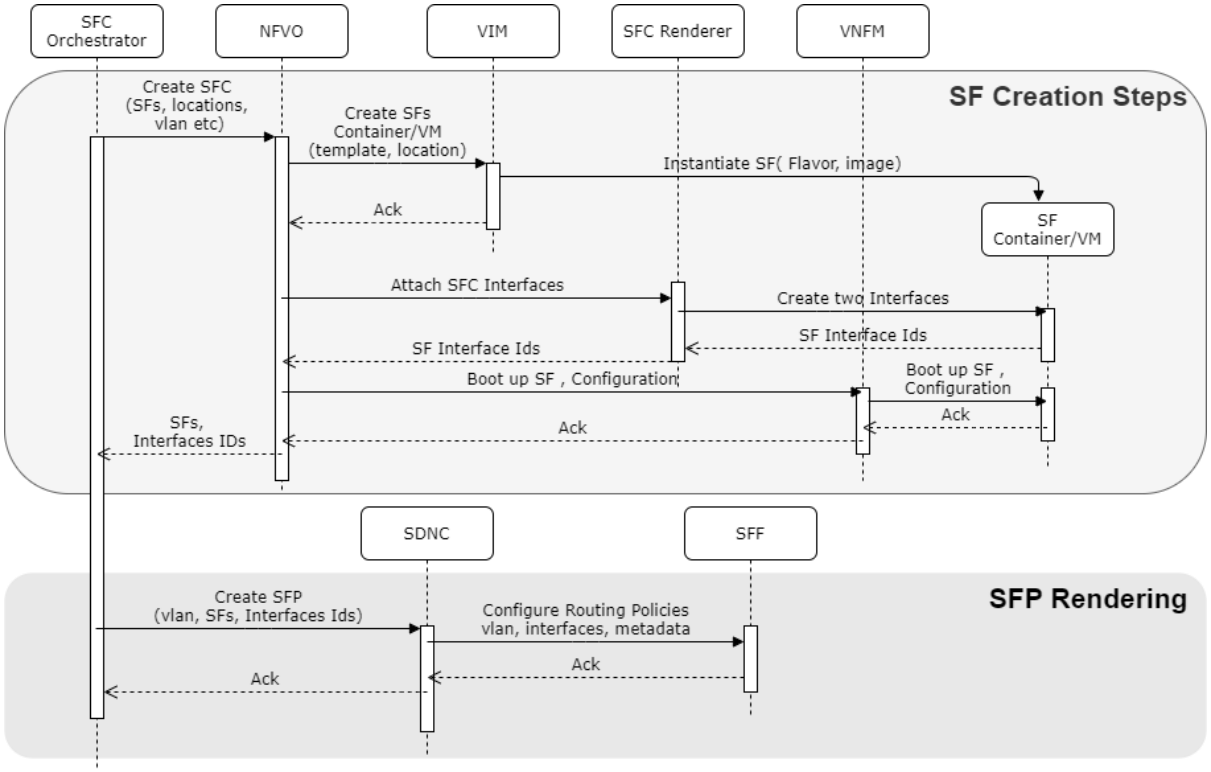
3.6.5.3 Information Flows in Service Function Chaining

3.6.5.3.1 Creation of Service Function Chain

The creation of the SFC might include design/preparation phase as:

82. The service functions that are included in the SFC.
83. The routing order in the service function, if the SFC is composed of more than one service function.

Creation of Service Function Chain shows SFC creation call flow, separated logically in two steps.



Creation of Service Function Chain

- Creation of service functions of SFC.

84. The flow of steps to enable the SFC creation can be as follows:
    a. SFC orchestrator creates the SFs with help of VNF MANO or CNF MANO.
    b. SFC Renderer attaches the SFC aware interfaces at SFs to enable Service plane
    c. NFVO boots up the relevant SF configurations at SF.

    Note: These steps are optional, if SFC orchestrator discovers that SFs are already created and existing.

2. Creation of Service Function Path (SFP) using the created SFs and associated interfaces.

85. A Service Function Path consists of:
    1. A set of ports(in VNF environment) or interfaces (in CNF environment) , that define the sequence of service functions
    2. A set of flow classifiers that specify the classified traffic flows entering the chain.

86. This step creates a new chain policy with chain rules. Chain rules can include the identifier of a traffic flow, service characteristics, the SFC identifier and related information to route the packets along the chain. Service characteristics can be application layer matching information (e.g., URL). Traffic flow identifier can be kind of traffic (e.g., Video, TCP, HTTP) flow need to be serviced. It can be specific Subscriber to apply service (e.g., parental control). The SFC identifier to steer the matched traffic along the SFP with SFC encapsulation.
    a. SFC orchestrator creates SFP with help of SDNC.
    b. SDNC pushes the SFC traffic steering policies to SFF(s).
    c. SFC classifier Policy provided for SFP to SFC classifier by SFC Controller. **Note:** not shown in call flow.

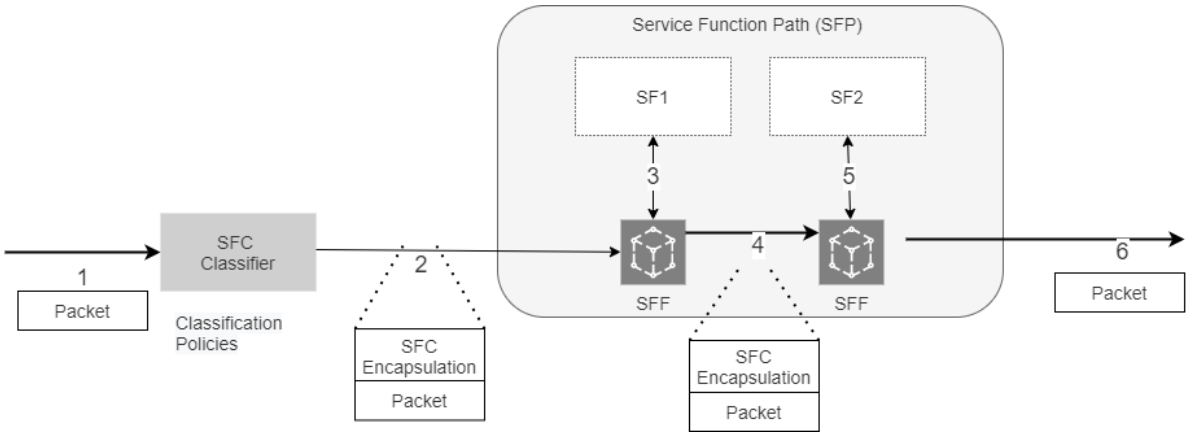
3.6.5.3.2 Updating Service Function Chain

SFP or SFC can be updated for various reasons and some of them are:

87. SFC controller monitors the SFP status and alerts SFC controller in case of not meeting SLA or some anomaly.
88. SFC design changes to update SF order, inclusion/removal of SFs
89. SFC Policy Rules changes

3.6.5.3.3 Data Steering in Service Function Chain

DataSteering shows traffic steering along SFP.



Data steering in Service Function Chain

90. SFC classifier detects the traffic flow based on classification policies. For example, to enable SGi-Lan feature as SFC, 5G User plane function (UPF) acts as SFC classifier. UPF receives the classification policies from 5G Policy control function (PCF) as traffic steering policies.
91. SFC classifier applies the SFC encapsulation (e.g., SCH, NSH) and routes traffic towards SFF, acts as entry point to SFP. The SFC Encapsulation provides, at a minimum, SFP identification, and is used by the SFC-aware functions, such as the SFF and SFC-aware SFs.
92. SFF based on SFC encapsulation routes the traffic to SF for service functionalities.
93. SF updates the SFC encapsulation based on its policies for further services.
94. At end of SFP, SFC encapsulation is removed and packet is routed out of SFP.

3.6.6 Time Sensitive Networking

Many network functions have time sensitivity for processing and require high precision synchronized clock for the Cloud Infrastructure. Subset of these workloads, like RAN, in addition require support for Synchronous Ethernet as well.

Reason for using Synchronous Precision Clock	Example	Comment
Achieve technical requirements	Strict latency or timing accuracy	Must be done for precise low latency communication between data source and receiver

Achieve technical requirements	Separation of processing pipeline	Ability to separate RAN into RU, DU, CU on different or stretch clusters

Table 3-7: Reasons and examples for Precise Clock and Synchronization

Precise Synchronization require specialized card that can be on server or network device motherboard or be part of NIC or both.

OpenStack and Kubernetes clusters use Network Time Protocol (NTP) (Protocol and Algorithms Specification [2], Autokey Specification [3], Managed Objects [4], Server Option for DHCPv6 [5]) as the default time synchronization for the cluster. That level of synchronization is not sufficient for some network functions. Just like real-time operating systems instead of base OS, so is precision timing for clock synchronization. Precision Time Protocol version 2 PTP [61] __ [32] is commonly used for Time-Sensitive Networking. This allow synchronization in microsecond range rather than millisecond range that NTP provides.

Some Network functions, like vDU, of vRAN, also require SyncE [62] [33]. Control, User and Synchronization (CUS) Plane specification defines different topology options that provides Lower Layer Split Control plane 1-4 (LLS-C1 - LLS-C4) with different synchronization requirements (ITU-T G.8275.2 [34]).

SyncE was standardized by the ITU-T, in cooperation with IEEE, as three recommendations:

95. ITU-T Rec. G.8261 that defines aspects about the architecture and the wander performance of SyncE networks
96. ITU-T Rec. G.8262 that specifies Synchronous Ethernet clocks for SyncE
97. ITU-T Rec. G.8264 that describes the specification of Ethernet Synchronization Messaging Channel (ESMC) SyncE architecture minimally requires replacement of the internal clock of the Ethernet card by a phase locked loop in order to feed the Ethernet PHY.

3.6.7 Load Balancer

Load Balancing is the process of distributing a set of tasks over a set of resources (computing units), with the goal of improving overall system's scalability and availability. Load balancing can optimize the response time and avoid unevenly overloading some compute nodes while other compute nodes are left idle.

Two main approaches to Load Balancing are: static algorithms which do not consider the state of different machines, and dynamic algorithms which are usually more general and more efficient but require exchanges of information between the different compute units, with possible trade off of integration complexity and reduced efficiency.

Load Balancers can be categorized or configured in following ways:

98. based on hardware devices or written in software,
99. operate on network Layer 4 or 7,
100. balance incoming traffic/requests using target selection like static: random, round robin, based on performance of target, based on client IP address, based on URL path or hash; and dynamic: least connections, least response time.

To increase redundancy, multiple replicas of load balancer are placed into a load balancing cluster.

3.6.8 Kubernetes Networking Semantics

The support for traditional network orchestration is non-existent in Kubernetes as it is foremost a Platform as a Service (PaaS) environment and not an Infrastructure as a Service (Iaas) component. There is no network orchestration API, like Neutron in OpenStack, and there is no way to create L2 networks, instantiate network services such as L3aaS and LBaaS and then connect them all together as can be done using Neutron.

Kubernetes networking can be divided into two parts, built in network functionality available through the pod's mandatory primary interface and network functionality available through the pod's optional secondary interfaces.

3.6.8.1 Built in Kubernetes Network Functionality

Kubernetes currently only allows for one network, the *cluster* network, and one network attachment for each pod. All pods and containers have an *eth0* interface, this interface is created by Kubernetes at pod creation and attached to the cluster network. All communication to and from the pod is done through this interface. Allowing only for one interface in a pod removes the need for traditional networking tools such as *VRFs* and additional routes and routing tables inside the pod network namespace.

3.6.8.2 Multiple Networks and Advanced Configurations

Currently Kubernetes does not in itself support multi networks, pod multi network attachments or network orchestration. This is supported by using a Container Network Interface [63] multiplexer such as Multus [64]. The Network Plumbing Working Group [65] has produced the Kubernetes Network Custom Resource Definition De-facto Standard [66]. This document describes how secondary networks can be defined and attached to pods.

3.7 Storage

3.7.1 Introduction to Storage

The general function of storage subsystem is to provide the persistent data store required for the delivery of a network service. In the context of Cloud Infrastructure the storage sub-system needs to accommodate needs of: the tenanted applications and the platform management. Each of:

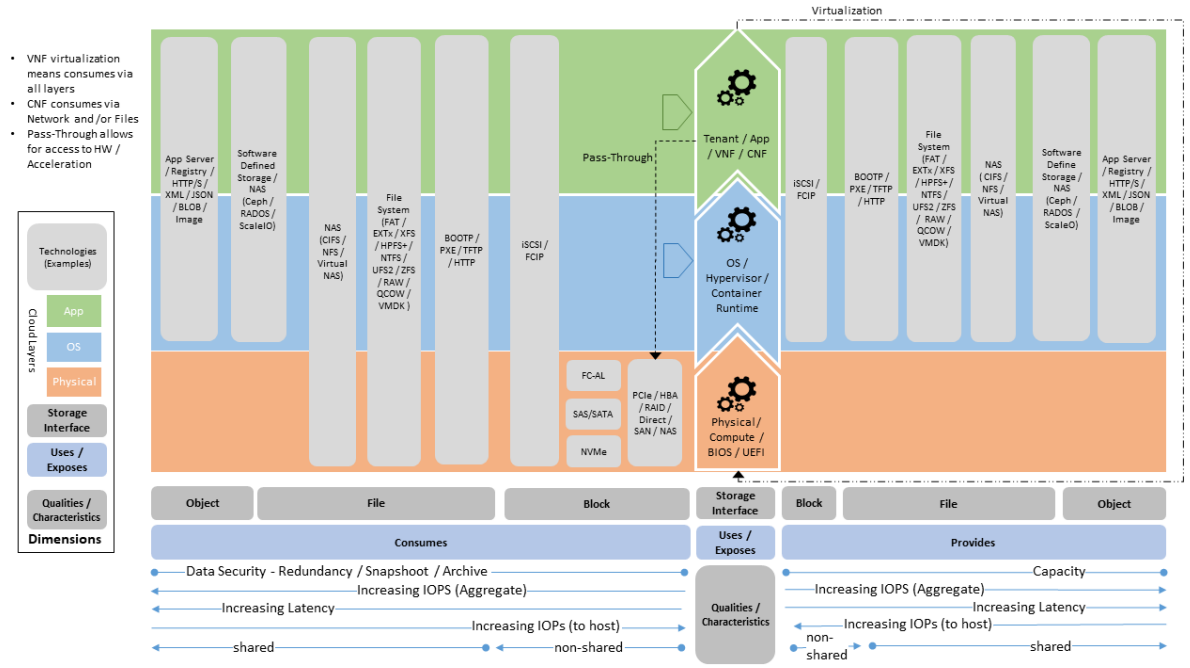
101. underlying compute host boot and virtual machine hosting,
102. control plane configuration and management plane storage for fault and performance management and automation, capacity management and reporting and
103. tenant application and VNF storage needs

have common and specific needs for storage in terms of performance, capacity and consumption models.

The combination of common but diverse needs in conjunction with the differences in the hosting environments (from large data-centres to small edge deployments) has resulted in the proliferation of storage technologies and their deployment architectures. To address this the "Reference Model" outlines a "General Cloud Storage Model" (see General Cloud Storage Model - "General Cloud Storage Model"). The model will outline the different types of storage technologies and how they can be used to meet the need for:

104. providing storage via dedicated storage systems,
105. multi-tenant cloud storage,
106. Control and Management Plane storage needs,

across both large data-centres and small edge deployments; the model can then be used for implementing Reference Architectures.



General Cloud Storage Model

Storage is multi-faceted and so can be classified based on its: cost, performance (IOPS, throughput, latency), capacity and consumption model (platform native, network shared, object or archival) and the underlying implementation model (in chassis, software defined, appliance). The objective of the model and set of stereotypes and perspectives is to provide guidance to architects and implementers in establishing storage solutions for Cloud Infrastructure.

The following principles apply to Storage scope for the Reference Model, Reference Architectures, Reference Implementations and Reference Conformance test suites:

107. Abstraction: A standardized storage abstraction layer between the Virtualisation Layers and the Storage Physical Resources Layer that hides (or abstracts) the details of the Storage Physical resources from the Virtualisation Layers.
108. Agnosticism: Define Storage subsystem concepts and models that can provide various storage types and performance requirements (more in section Virtual Storage).
109. Automation: Enable end-to-end automation, from Physical Storage installation and provisioning to automation of workloads (VNF/CNF) onboarding.
110. Openness: All storage is based on open source or standardized APIs (North Bound Interfaces (NBI) and South Bound Interfaces (SBI)) and should enable integration of storage components such as Software Defined Storage controllers.
111. Scalability: Storage model enables scalability to enable small up to large deployments.
112. Workload agnostic: Storage model can provide storage functionality to any type of workloads, including tenant VNF, CNF and Infrastructure Management whether this is via BareMetal or Virtualised Deployments.
113. Operationally Amenable: The storage must be amenable to consistent set of operational processes for: Non-Disruptive Capacity Expansion and Contraction, Backup/Restoration and Archive and Performance Management. Where applicable (examples are

Backup/Restoration/Archive) these processes should also be able to be provided to tenants for their own delegated management.

114. Security Policy Amenable: The storage sub-systems must be amenable to policy based security controls covering areas such as: Encryption for Data at Rest / In Flight, Delegated Tenant Security Policy Management, Platform Management Security Policy Override, Secure Erase on Device Removal and others

115. Future proof: Storage model is extendible to support known and emerging technology trends covering spectrum of memory-storage technologies including Software Defined Storage with mix of SATA- and NVMe-based SSDs, DRAM and Persistent Memory, integrated for multi-clouds, and Edge related technologies.

The above principles should be understood as storage specific specialisations of the common/chapter00:anuket general principles.

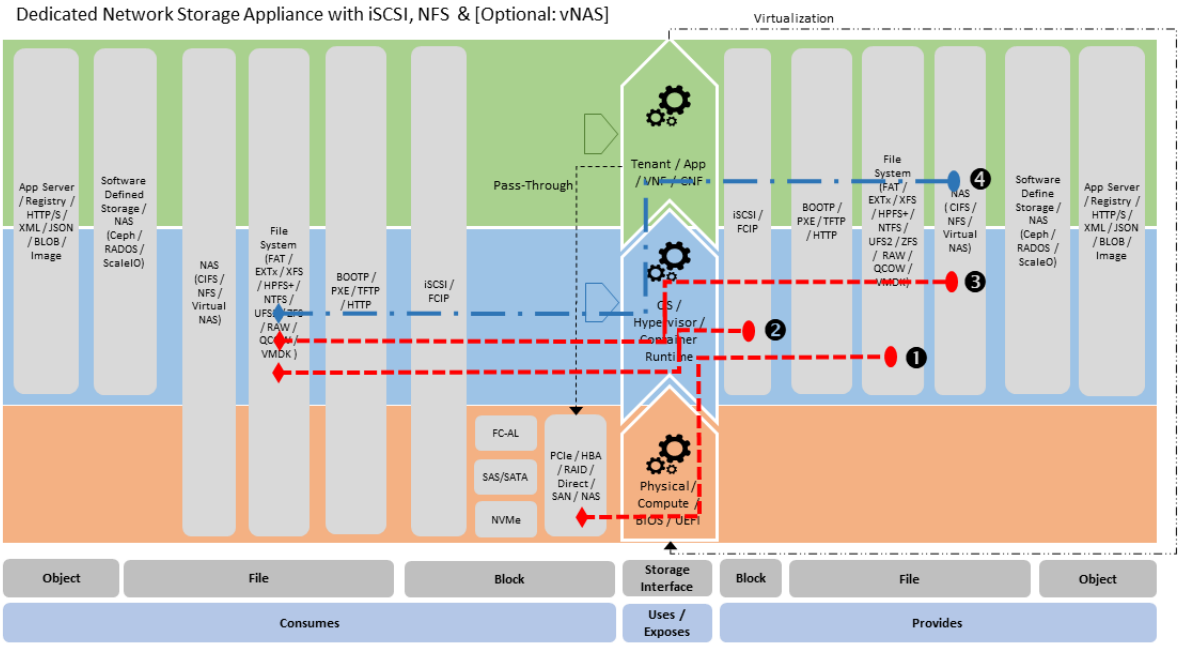
3.7.2 Storage Implementation Stereotypes

The following set of storage implementations outline some of the most prevalent stereotypical storage implementations.

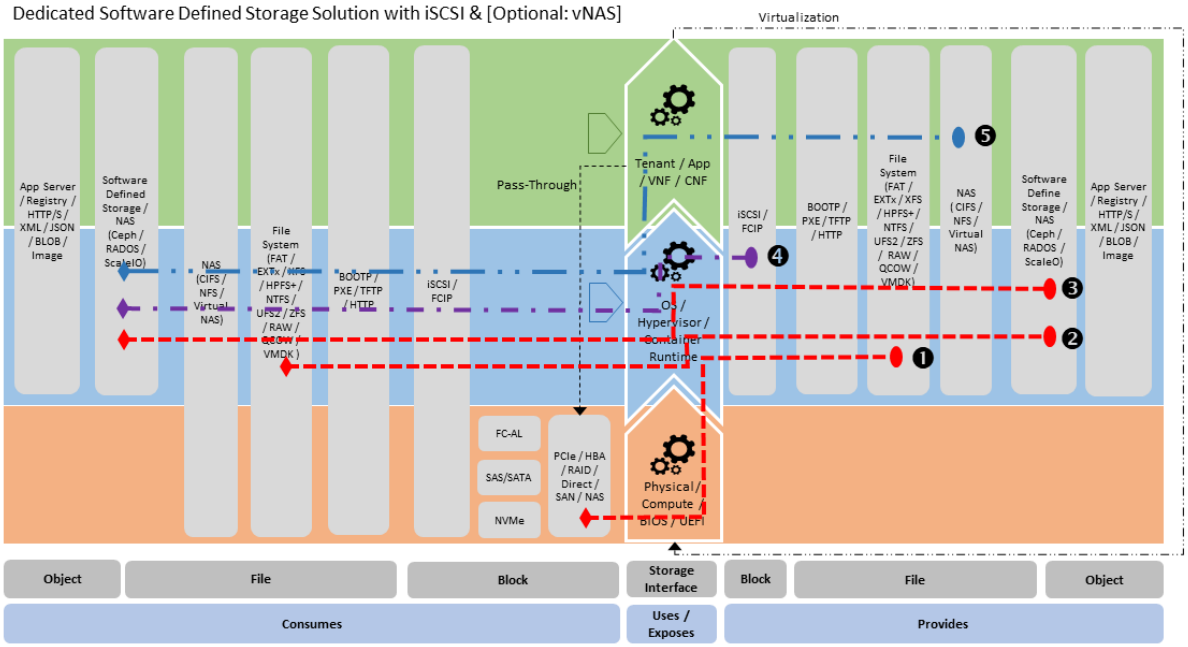
The first of these are for Data Centre Storage cases, with stereotypes of:

116. Dedicated storage appliance (Storage Appliance Stereotype) - that provide network based storage via iSCSI (2), NFS/CIFS (3) with potentially virtual NAS (vNAS) (4) capability. Having virtual network software (4) allows the establishment of storage tenancies, where storage tenancy have their own virtual storage services which are exposed on their own network,

117. Software defined storage (Software Defined Storage Stereotype) - which is able to provide similar capabilities as the dedicated storage appliance (see (3),(4) & (5) in diagram). In this case this is provided as a software solution on top of a hyper-converged infrastructure.



Storage Appliance Stereotype

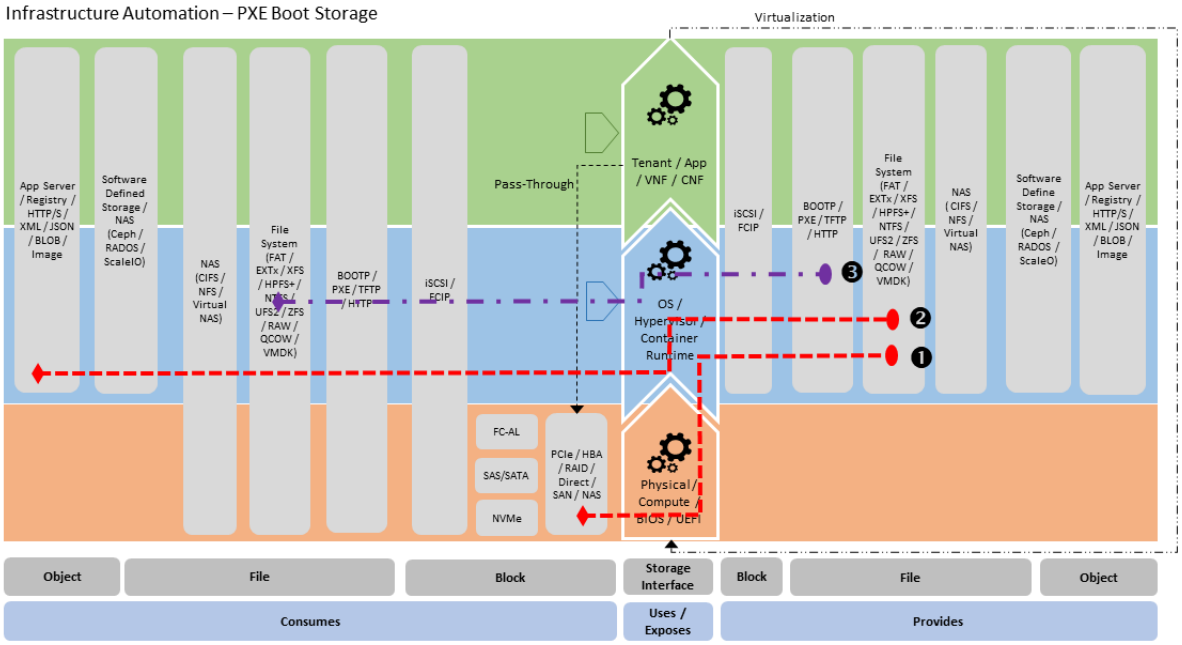


Software Defined Storage Stereotype

Both of these stereotypes can be used to support very broad storage needs from: machine boot (via iSCSI), providing storage to the Cloud Platform Control and Management Planes, Platform Native (viz., Hypervisor Attached and Container Persistence storage, as defined in section Storage for Tenant Consumption") and Application/VNF/CNF managed network storage. To provide this requires connectivity within the Cloud Infrastructure Underlay and Tenant Overlay networks.

Successful management of Cloud Infrastructure requires high levels of automation, including the ability to stand up rapidly new storage and hosting infrastructure. This Cloud Infrastructure boot-strapping process is managed through Infrastructure Automation tooling. A typical part of the boot-strap process is to use PXE (Pre-boot Execution Environment) boot to manage the deployment of initial images to physical hosts and a similar approach is used for "Bare Metal-as-a-Service" provisioning. The storage stereotype that covers this use case is:

118. Infrastructure Automation (Infrastructure Automation - PXE Boot Server Stereotype) - where PXE Boot Server provides a cache of boot images that are stored in local storage (2) which are then conditionally served up as PXE boot images (3). The PXE boot server can run within bootstrap management hosting in data-centre or within the routing / switch layer for an edge deployment case aimed to minimise physical footprint. The Infrastructure Automation PXE server is aware of the provisioning status of the physical infrastructure and will serve specific images or even not respond to PXE boot requests for hosts which have already been provisioned and are considered "in service".



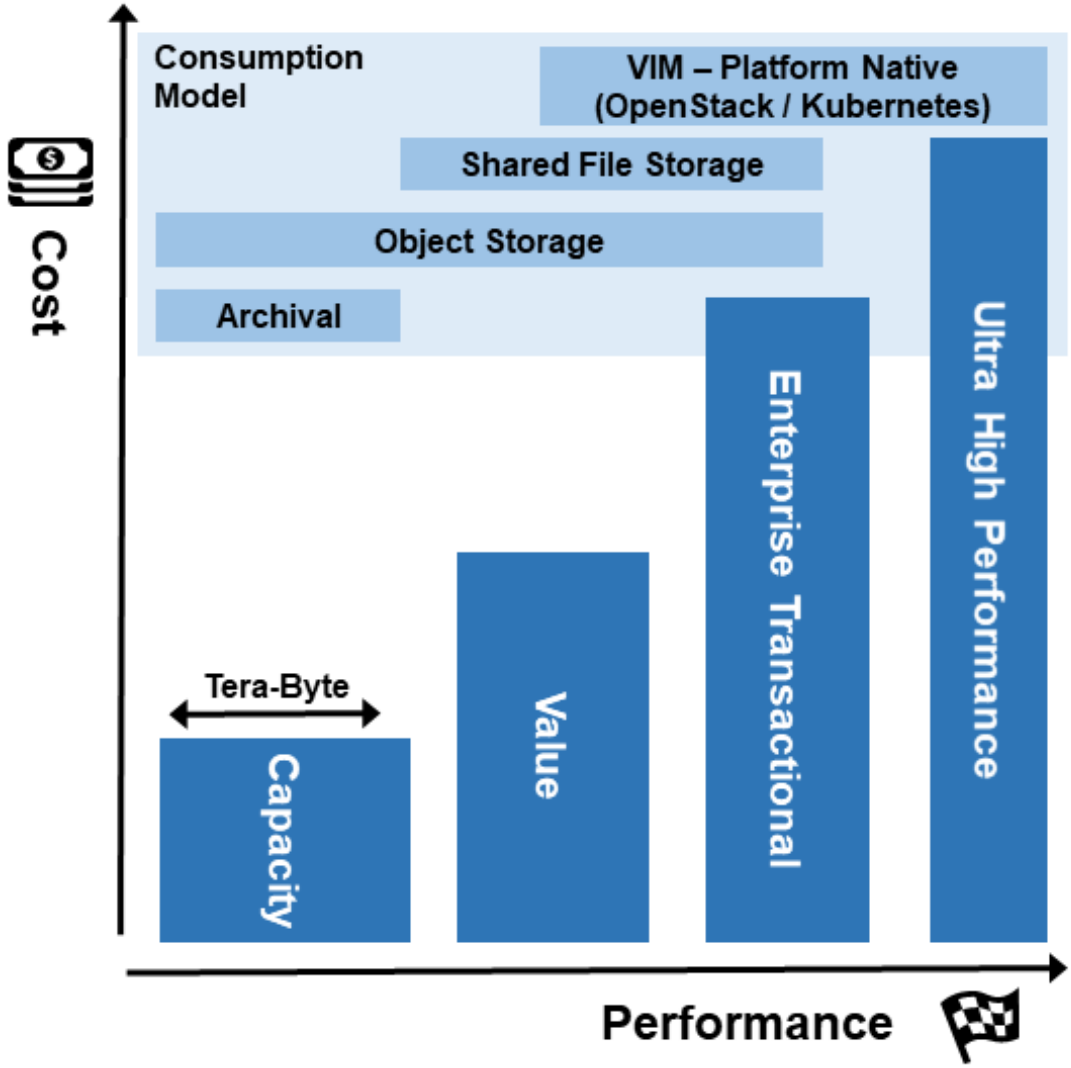
Infrastructure Automation - PXE Boot Server Stereotype

To provide PXE boot service to the underlying resource hosts, the PXE server must be connected to the same network as the NIC that is configured for PXE boot. The "Infrastructure Automation - PXE Server" stereotype is also applicable to booting tenant Virtual Machines. In this case, the PXE server is on the same network as one of the machines vNICs. For tenant use this is provided as part of tenant consumable boot infrastructure services.

For each of the defined stereotypes, the storage service uses physical Block storage for boot (Physical Layer - Block Consumption -> OS File Systems Exposure (1) on stereotype diagrams). This is the primary use case for use of in chassis physical storage, which is not being used for consumption and exposure as network-based storage. In general, it is desirable to use network based storage solution for provision of Cloud Infrastructure storage. The "Infrastructure Automation - PXE Server" is an exception to the preferential use of network-based storage, and as it is managing the bootstrap process, it cannot be dependent on a separate storage system for maintaining its image cache.

3.7.3 Storage for Tenant Consumption

Storage is made available for tenant consumption through a number of models. A simplified view of this is provided in the following illustrative model.



Storage Model - Cost vs Performance with Consumption Model Overlay

Where:

119. (Comparative) Cost - is monetary value / unit of end user storage capacity

120. Performance - is defined by IOPS / Latency / Throughput as typically each of these increases with successive generations of storage

121. Capacity - consumption needs are represented by width of the: Ultra High Performance, Enterprise Transactional, Value and Capacity storage options.

122. Storage Types - is how the storage is accessed and used, where:

    1. Platform Native - is managed by the hypervisor / platform (examples are a virtual disk volume from which a VNF boots and can write back to, the storage interface that is exposed by the container runtime), this storage is typically not shared across running VNF / CNF instances;
    2. Shared File Storage - is storage that is accessed through a file systems interface (examples are network based storage such as CIFS or NFS) where the storage volumes can be accessed and shared by multiple VNF / CNF instances;
    3. Object Storage - is storage that is accessed via API interfaces (the most common example being HTTP restful services API), which support get/put of structured objects; and

4. Archival - is storage that is targeted for provision of long term storage for purpose of disaster recovery, meeting legal requirements or other historical recording where the storage mechanism may go through multiple stages before landing at rest.

The storage model provides a relatively simple way for the storage consumer to specify / select their storage needs. This is shown in the following table which highlights key attributes and features of the storage classes and "epic use cases" for common usage patterns.

Storage Type	Consumption Model	Performance & Capacity	Cost	Infrastructure Strategy	Use Case
Platform Native	Managed by the VIM / Hypervisor and attached as part of VNF/CNF start up via VNF Descriptor, Volumes shareability across VNF/CNF instances is determined by platform and storage capabilities	Ultra-High Performance & Very High Performance, Capacity: 10GB - 5TB, "Tier 1"	High to Very High	Always part of VIM deployment, Storage is directly next to vCPU, Can support highest performance use cases, Always available to support VNF/CNF boot/startup	Boot/Start VNF/CNF, Live Migrate Workload within and across VIMs
Shared File Storage	Access via Network File System, Concurrent consumption across multiple VNF/CNFs, Sharing can be constrained to tenancy, cross tenancy and externally accessible	Enterprise Transactional Performance (real time transaction processing), Capacity: 5GB - 100TB, Selectable "Tier 1" to "Tier 3"	High -Mid	Leverage existing capabilities, Only build if needed (this is not needed by many data plane VNF/CNFs), If needed for Edge deployment then aim to unify with "Platform Native" deployment	VNF/CNF's able to share the same file content
Object Storage	Consumed via HTTP/S restful services,	Highly distributable and scalable Provided by serving application which manages storage needs, Location Independent	High to Mid	Primarily tenant application responsibility	Cloud Native Geo-Distributed VNF/CNFs

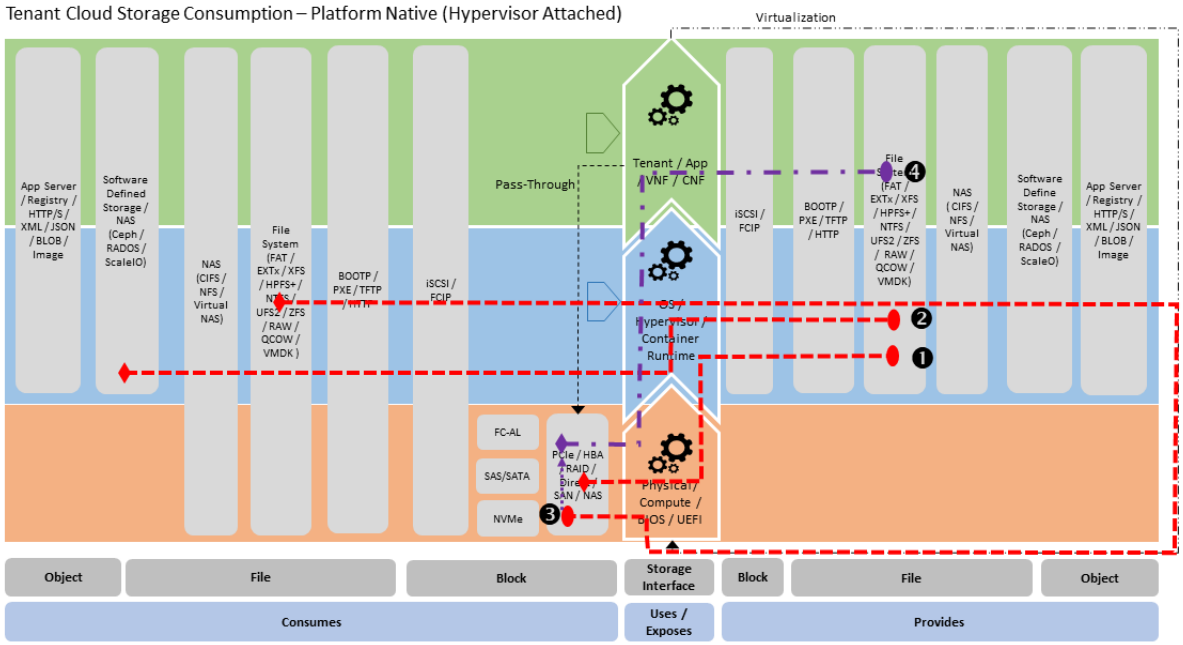
Capacity	Typically accessed as per "Shared Storage" but will likely have additional storage stages, Not suitable for real time processing	Very low transactional performance, Need throughput to accommodate large data flow, "Tier 3"	Low	Use cheapest storage available that meets capacity & security needs	Archival storage for tenant/platform backup/restore, DR

Table 3-8: Tenant Storage Types

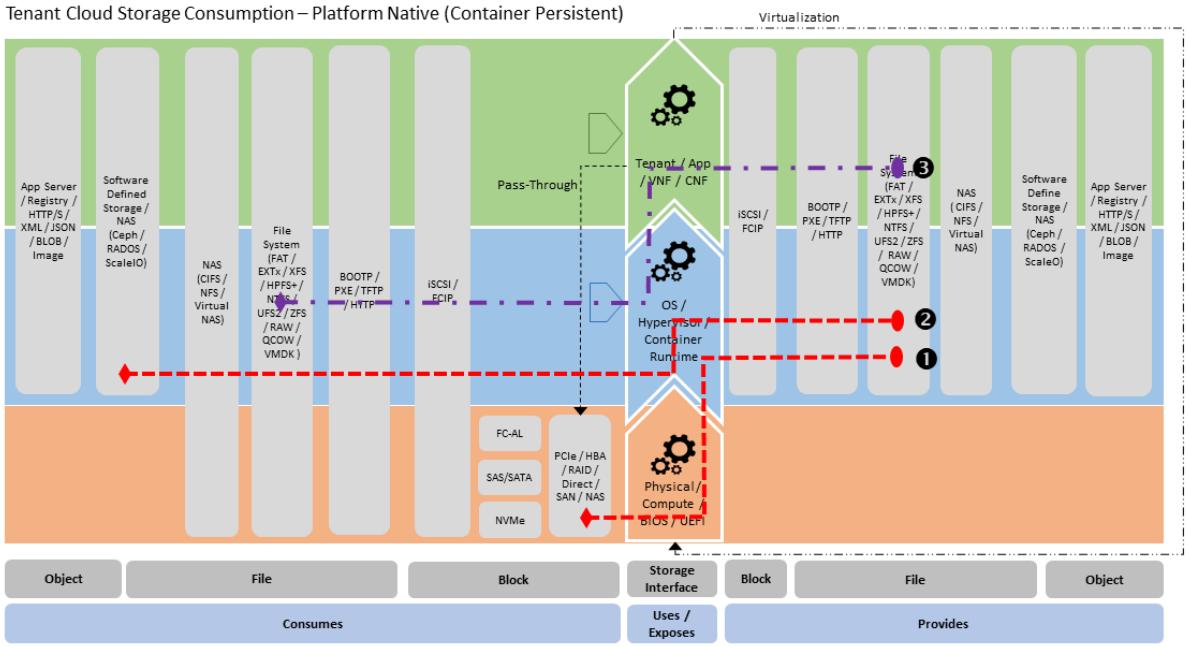
In section "3.6.2 Storage Implementation Stereotypes" the General Cloud Storage Model is used to illustrate the provision of storage. The model can also be used to illustrate the consumption of storage for use by Tenants (see below for "Platform Native" stereotypes):

123. Platform Native - Hypervisor Attached Consumption Stereotype (Platform Native - Hypervisor Attached Consumption Stereotype) - where hypervisor consumes Software Defined Storage via Network (RA-1 - Cinder backend (2)) and the Block Image is attached to Virtual Machine (RAW or QCOW file within File System), which is used for boot and exposure to virtual machine OS as Block Storage (3). The virtual machine OS in turn consumes this for use by Tenant Application via File System,

124. Platform Native - Container Persistent Consumption Stereotype (Platform Native - Container Persistent Consumption Stereotype) - is simpler case with Container Runtime consuming Software Defined Storage (via Reliable Autonomic Distributed Object Store (RADOS) backend (2)) and exposes this to Container as a file system mount (3).



Platform Native - Hypervisor Attached Consumption Stereotype



Platform Native - Container Persistent Consumption Stereotype

Note that a stereotype for Network File Storage consumption is not illustrated as this is simply managed by the Tenant Application by doing a file systems mount.

In cloud infrastructure, the storage types may manifest in various ways with substantive variations in the architecture models being used. Examples of this are provided in section Storage Implementation Stereotypes, with stereotypes for "Dedicated Storage Appliance" and "Software Defined Storage". In the consumption case, again there is use of in-chassis storage to support hypervisor and container host OS/Runtime boot, not for Tenant / User Plane storage consumption.

3.7.4 Storage Scenarios and Architecture Fit

The storage model and stereotypical usage scenarios illustrate the key storage uses cases and their applicability to support storage needs from across a range of cloud deployments. This set of storage uses cases is summarised in the following tables, including how the stereotypes can support the Anuket Reference Architectures, followed by the key areas for consideration in such a deployment scenario. The structure of the table is:

125. Use Case - what is the target storage use case being covered (large data-centre, small data-centre, standalone cloud, edge etc.)
126. Stereotype - which of defined stereotypes is used
127. Infra / Ctrl / Mgt - is the storage stereotype able to support the:
    1. Infrastructure - for host computer boot (from either local host storage or PXE),
    2. Control Plane - for cloud infrastructure control (such as OpenStack (RA1) or Kubernetes (RA2) control functions) and
    3. Management Plane Needs - for Infrastructure Automation, Tenant VNF/CNF Orchestration and cloud infrastructure monitoring and assurance
128. Tenant / User - is the storage stereotype able to support Tenant / User Plane needs including: Platform Native, Shared File Storage & Object Storage (as per section Storage for Tenant Consumption)

Where:

129. "Y" - Yes and almost always provided
130. "O" - Optional and readily accommodated
131. "N" - No, not available
132. "NA" - Not Applicable for this Use Case / Stereotype

Columns relevant for Infra / Ctrl / Mgmt are:

133. Boot
134. Ctrl
135. Mgt

Columns relevant for Tenant / User are:

136. Platform Native:
    1. Hypervisor Attached
    2. Container Persistent
137. Shared File:
    1. Within
    2. Cross
    3. Ext
    4. vNAS
138. Object

Table 5Storage Use Cases and Stereotypes

Use Case	Stereotype	Boot	Ctrl	Mgt	Hypervisor Attached	Container Persistent	Within	Cross	Ext	vNAS	Object
Data-centre Storage	Dedicated Network Storage Appliance	Y	Y	Y	Y	Y	O	O	O	O	O
Data-centre Storage	Dedicated Software Defined Storage	O	O	O	Y	Y	O	O	O	O	O
Data-centre Storage	Traditional SAN	Y	Y	Y	N	N	N	N	N	N	N
Satellite data-centre Storage	Small Software Defined Storage	O	O	O	Y	Y	O	O	O	O	O

Small data-centre Storage	Converged Software Defined Storage	O	O	O	Y	Y	O	O	O	O	O	O			
Edge Cloud	Edge Cloud for VNF/CNF Storage	NA	O	NA	Y	Y	O	O	O	O	O				
Edge Cloud	Edge Cloud for Apps Storage	NA	O	NA	Y	Y	O	O	O	O	Y				
Edge Cloud	Edge Cloud for Content Mgt Storage	NA	O	NA	Y	Y	O	O	O	O	Y				
Split Control/User Plane Edge Cloud	Split Edge Ctrl Plane Storage	NA	N	NA	Y	Y	O	O	O	O	O				
Split Control/User Plane Edge Cloud	Split Edge User Plane Storage	NA	N	NA	N	N	N	N	N	N	N				

Table 3-9: Storage Use Cases and Stereotypes

The storage sub-system is a foundational part of any Cloud Infrastructure, as such it is important to identify the storage needs, based on target tenant use cases, at inception. This will allow the right set of considerations to be addressed for the deployment. A set of typical considerations is provided:

139. for various use cases to meet functional and performance needs and
140. to avoid the need for significant rework of the storage solution and the likely ripple through impact on the broader Cloud Infrastructure.

The considerations will help to guide the build and deployment of the Storage solution for the various Use Cases and Stereotypes outlined in the Table 3-10.

Table 6Storage Considerations

Use Case	Description
Data-centre Storage	Provide a highly reliable and scalable storage capability that has flexibility to meet diverse needs
Meets Needs of	141. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 142. Cloud Infrastructure Management Plane (Cloud Infrastructure fault and performance management and platform automation) 143. Cloud Infrastructure Tenant / User Plane
General Considerations: What are the general considerations, irrespective of the deployment stereotype/technology used in the storage sub-system?	• Can storage support Virtual Machine (RA-1) and Container (RA-2) Hosting cases from single instance? Noting that if you wish to have single storage instance providing storage across multiple clusters and/or availability zones within the same data-centre then this needs to be factored into the underlay network design.

	• Can the storage system support Live Migration/Multi-Attach within and across Availability Zones (applicable to Virtual Machine hosting (RA-1)) and how does the Cloud Infrastructure solution support migration of Virtual Machines between availability zones in general? • Can the storage system support the full range of Shared File Storage use cases: including the ability to control how network exposed Share File Storage is visible: Within Tenancy, Across Tenancy (noting that a Tenancy can operate across availability zones) and Externally? • Can the storage system support alternate performance tiers to allow tenant selection of best Cost/Performance option? For very high performance storage provision, meeting throughput and IOP needs can be achieved by using very high IOP flash storage, higher bandwidth networking, performance optimised replication design and storage pool host distribution, while achieving very low latency targets require careful planning of underlay storage VLAN/switch networking.
Specific Considerations: In selecting a particular stereotype/technology this can bring with it considerations that are specific to this choice	Dedicated Software Defined Storage: • Need to establish the physical disk data layout / encoding scheme choice, options could be: replication / mirroring of data across multiple storage hosts or CRC-based redundancy management encoding (such as "erasure encoding"). This typically has performance/cost implications as replication has a lower performance impact, but consumes larger number of physical disks. If using replication then increasing the number of replicas provide greater data loss prevention, but consumes more disk system backend network bandwidth, with bandwidth need proportional to number of replicas. • In general with Software Defined Storage solution it is not to use hardware RAID controllers, as this impacts the scope of recovery on failure as the failed device replacement can only be managed within the RAID volume that disk is part of. With Software Defined Storage failure recovering can be managed within the host that the disk failed in, but also across physical storage hosts. 3. Can storage be consumed optimally irrespective of whether this is at Control, Management or Tenant / User Plane? Example is iSCSI/NFS, which while

	available and providing a common technical capability, does not provide best achievable performance. Best performance is achieved using provided OS layer driver that matches the particular software defined storage implementation (example is using RADOS driver in Ceph case vs. Ceph ability to expose iSCSI). Dedicated Network Storage Appliance: 1. Macro choice is made based on vendor / model selection and configuration choices available Traditional SAN: • This is generally made available via Fiber Channel Arbitrated Loop (FC-AL)/SCSI connectivity and hence has a need for very specific connectivity. To provide the features required for Cloud Infrastructure (Shared File Storage, Object Storage and Multi-tenancy support), a SAN storage systems needs to be augmented with other gateway/s to provide an IP Network consumable capability. This is often seen with current deployments where NFS/CIFS (NAS) Gateway is connected by FC-AL (for storage back-end) and IP Network for Cloud Infrastructure consumption (front-end). This model helps to extent use of SAN storage investment. NOTE: This applies to SANs which use SAS/SATA physical disk devices, as direct connect FC-AL disk devices are no longer manufactured.
Satellite Data-centre Storage	Satellite data-centre is a smaller regional deployment which has connectivity to and uses resources available from the main Data-centre, so only provides support for subset of needs
Meets Needs of	144. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 145. Cloud Infrastructure Tenant/User Plane
General Considerations: What are the general considerations, irrespective of the deployment stereotype/technology used in storage sub-system?	• Is there a need to support multiple clusters/availability zones at the same site? If so then use "Data-Centre Storage" use case, otherwise, consider how to put Virtual Machine & Container Hosting control plane and Storage control plane on the same set of hosts to reduce footprint. • Can Shared File Storage establishment be avoided by using capabilities provided by large Data-Centre Storage? • Can very large capacity storage needs be moved to larger Data-Centre Storage capabilities?

Specific Considerations: In selecting a particular stereotype/technology this can bring with it considerations that are specific to this choice	Small Software Defined Storage: • Leverage same technology as "Dedicated Software Defined Storage" scenarios, but avoid/limit Infrastructure boot and Management plane support and Network Storage support • Avoid having dedicated storage instance per cluster/availability zone • Resilience through rapid rebuild (N + 1 failure scenario)
Small Data-centre Storage	Small data-centre storage deployment is used in cases where software-defined storage and virtual machine / container hosting are running on a converged infrastructure footprint with the aim of reducing the overall size of the platform. This solution behaves as a standalone Infrastructure Cloud platform.
Meets Needs of	146. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 147. Cloud Infrastructure Management Plane (Cloud Infrastructure fault and performance management and platform automation) 148. Cloud Infrastructure Tenant / User Plane
General Considerations: What are the general considerations, irrespective of the deployment stereotype/technology used in the storage sub-system?	• Is there needed to support multiple clusters / availability zones at same site? See guidance for "Satellite Data-centre Storage" use case(1). • Is Shared File Storage required? Check sharing scope carefully as fully virtualised NFs solution adds complexity and increases resources needs. • Is there need for large local capacity? With large capacity flash (15-30 TB/device), the solution can hold significant storage capacity, but need to consider carefully data loss prevention need and impact on rebuilt/recovery times
Specific Considerations: In selecting a particular stereotype/technology this can bring with it considerations that are specific to this choice	Converged Software Defined Storage: • Leverage same technology as "Dedicated Software-Defined Storage" scenarios, but on converged infrastructure. To meet capacity needs provision three hosts for storage and the rest for virtual infrastructure and storage control and management and tenant workload hosting. • If the solution needs to host two clusters/availability zones then have sharable storage instances. • Resilience through rapid rebuild (N + 0 or N + 1)

Edge Cloud for App Storage	Support the deployment of Applications at the edge, which tend to have greater storage needs than a network VNF/CNF
Meets Needs of	149. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 150. Cloud Infrastructure Tenant / User Plane - very limited configuration storage
Edge Cloud for VNF/CNF Storage	Support the deployment of VNF / CNF at the edge.
Meets Needs of	151. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 152. Cloud Infrastructure Tenant / User Plane - limited configuration storage
Edge Cloud for Content Storage	Support the deployment of deployment of media content cache at the edge. This is a very common Content Distribution Network (CDN) use case
Meets Needs of	153. Cloud Infrastructure Control Plane (tenant Virtual Machine and Container life-cycle management and control) 154. Cloud Infrastructure Tenant / User Plane - Media Content Storage
General Considerations: What are the general considerations, irrespective of the deployment stereotype/technology used in the storage sub-system?	• Consuming and exposing Object storage through Tenant application • Use Embedded Shared File Storage for Control and Tenant Storage Needs
Specific Considerations: In selecting a particular stereotype/technology this can bring with it considerations that are specific to this choice	Embedded Shared File Storage: • What is the best way to achieve some level of data resilience, while minimising required infrastructure? (i.e. do not have luxury of having host (VMs) dedicated to supporting storage control and storage data needs)

Table 3-10: Storage Considerations

The General Storage Model illustrates that at the bottom of any storage solution there is always the physical storage layer and a storage operating system of some sort. In a Cloud Infrastructure environment what is generally consumed is some form of network storage which can be provided by the:

155. Infrastructure platform underlay network for Control Plan and Platform Native - Hypervisor Attached and Container Runtime Managed

156. Tenant / User overlay network for Shared File Storage and Object Storage

In general for the provision of storage as shared resource it is not desirable to use "in chassis storage" for anything other than in the storage devices for platform hypervisor/OS boot or for the hosts providing the storage sub-systems deployment itself. This is due to difficulty in resulting operational management (see principles section Introduction to Storage - "Operationally Amenable" above).

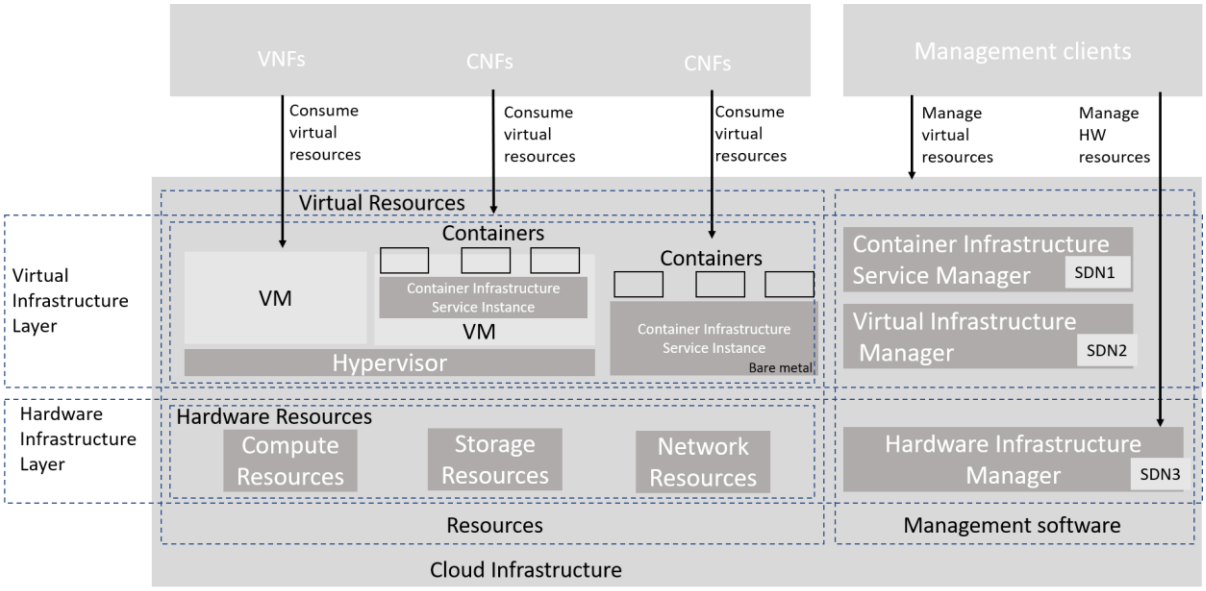
For cloud-based storage, "Ephemeral" storage (hypervisor attached or container images which are disposed when VNF/CNF is stopped) is often distinguished from other persistent storage, however this is a behaviour variation that is managed via the VNF descriptor rather than a specific Storage Type.

Storage also follows the alignment of separated virtual and physical resources of Virtual Infrastructure Layer and HW Infrastructure Layer. Reasons for such alignment are described more in Section Network.

While there are new storage technologies being made available and a trend towards the use of flash for all physical storage needs, for the near future, the core storage architecture for Cloud Infrastructure is likely to remain consistent with the network-based consumption model, as described through the stereotypes.

3.8 Sample reference model realization

The following diagram presents an example of the realization of the reference model, where a virtual infrastructure layer contains three coexisting but different types of implementation: a typical IaaS using VMs and a hypervisor for virtualisation, a CaaS on VM/hypervisor, and a CaaS on bare metal. This diagram is presented for illustration purposes only and it does not preclude validity of many other different combinations of implementation types. Note that the model enables several potentially different controllers orchestrating different type of resources (virtual and/or hardware). Management clients can manage virtual resources via Virtual Infrastructure Manager (Container Infrastructure Service Manager for CaaS, or Virtual Infrastructure Manager for IaaS), or alternatively hardware infrastructure resources via hardware infrastructure manager. The latter situation may occur for instance when an orchestrator (an example of a management client) is involved in provisioning the physical network resources with the assistance of the controllers. Also, this realization example would enable implementation of a programmable fabric.



Reference model realization example

The terms Container Infrastructure Service Instance and Container Infrastructure Service Manager should be understood as defined in ETSI GR NFV-IFA 029 V3.3.1 [9]. More detailed deployment examples can be found in network of this Reference Model chapter.

3.9 Hardware Acceleration Abstraction

The purpose of a Hardware Accelerator is either to Accelerate the execution of an application or to Offload functions from the generic CPU to make the application and/or Cloud Infrastructure more efficient from one or more aspects.

Hardware Accelerators are often used in Telco Clouds for many reasons. Some applications require an Hardware Accelerator to perform tasks that a generic CPU cannot perform fast enough, with enough timing accuracy, or handle the traffic that must be kept in a single context. Other applications could be satisfied with a generic CPU performance in some deployment cases, whilst being inefficient in other situations. The Cloud Infrastructure might also benefit from specialised accelerated HW devices to perform its tasks with less power, space, or cost than a generic CPU.

The Accelerators are specialized resources and generally not expected to exist in large quantities, which makes it important that these limited HW Accelerators are carefully assigned to where they can be best used most of the time. In general, this requires that there be software-based alternative functions that can be used for the occasions when HW Accelerators cannot be assigned to accelerate or offload applications or Cloud Infrastructure tasks.

It is preferred that the accelerated or offloaded functions have abstracted interfaces since that would hide the different implementations from a functional point of view and make orchestrator choices simpler and more transparent to deploy. It will also allow support for multiple different HW Accelerators, and reducing the operator's integration and test efforts of the accelerators and their applications and/or Cloud Infrastructure.

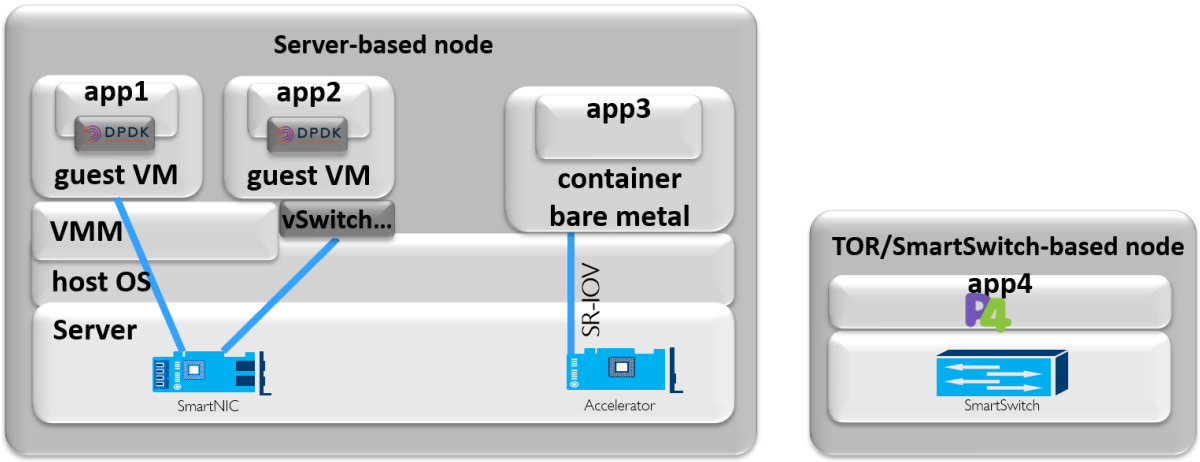
3.9.1 Types of Accelerators

Accelerator technologies can be categorized depending on where they are realized in the hardware product and how they get activated, life cycle managed and supported in running infrastructure.

Acceleration technology/hardware	Example implementation	Activation/LCM/support	Usage by application tenant
CPU instructions	Within CPU cores	None for hardware	Application to load software library that recognizes and uses CPU instructions
Fixed function accelerator	Crypto, vRAN-specific adapter	Rare updates	Application to load software library/driver that recognizes and uses the accelerator
Firmware-programmable adapter	Network/storage adapter with programmable part of firmware image	Rare updates	Application normally not modified or aware
SmartNIC	Programmable accelerator for vSwitch/vRouter, NF and/or	Programmable by Infrastructure operator(s) and/or application tenant(s)	3 types/operational modes: 1. Non-programmable normally with

	Hardware Infrastructure		unaware applications; 2. Once programmable to activate; 3 Reprogrammable
SmartSwitch-based	Programmable Switch Fabric or TOR switch	Programmable by Infrastructure operator(s) and/or application tenant(s)	3 operational modes: 1. Non-programmable normally with unaware applications; 2. Once programmable to activate; 3. Reprogrammable

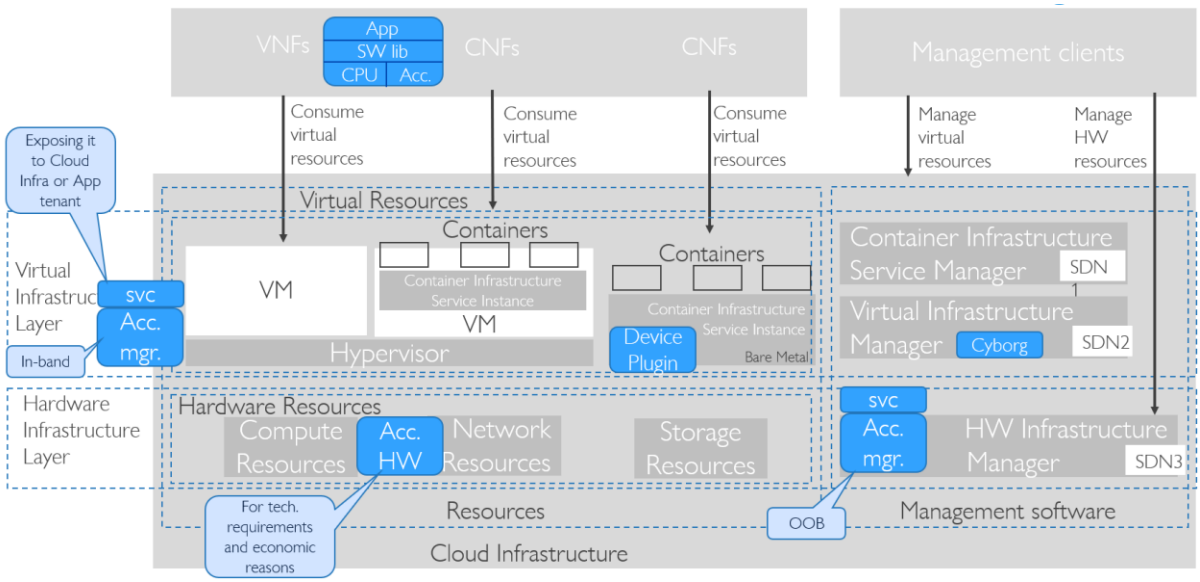
Table 3-11: Hardware acceleration categories, implementation, activation/LCM/support and usage



Examples of server- and SmartSwitch-based nodes (for illustration only)

3.9.2 Infrastructure and Application Level Acceleration

Hardware Acceleration in RM Realization Diagram gives examples for the Hardware Accelerators shown in Reference model realization example (the Sample reference model realization diagram).



Hardware Acceleration in RM Realization Diagram

Hardware Accelerators are part of the Hardware Infrastructure Layer. Those that need to be activated/programmed will expose management interfaces and have Accelerator Management software managing them in-band (from host OS) or out of band (OOB, over some network to the adapter without going through host OS). For more flexibility in management, such Accelerator Management can be carried over appropriate service with authentication mechanism before being exposed to Cloud Infrastructure operator and/or Application tenant.

Application uses software library supporting hardware acceleration and running on generic CPU instructions. Mapping workload to acceleration hardware is done with Cyborg in OpenStack or Device Plugin framework in Kubernetes. Hardware accelerator supports both in-band and/or out of band management, with service exposing it to Cloud Infrastructure operator or Application tenant roles.

Hardware Accelerators can be used as:

157. Virtualization Infrastructure layer acceleration: Example can be vSwitch, which can be leveraged agnostically by VNFs if standard host interfaces (like VirtIO) are used.

158. Application layer acceleration: Example of software library/framework (like DPDK) in VM providing Application level acceleration with (where available) hardware-abstracted APIs to access platform Hardware Acceleration and providing software equivalent libraries when hardware assist not available.

159. Hardware Infrastructure layer offload: Example can be an OOB managed underlay network separation providing network separation secured from host OS reach on any provisioned transport switch infrastructure.

Two levels of consumption are for underlay separation or overlay acceleration. Underlay Separation ensures that multiple different Virtualization Infrastructure instances are kept in separate underlay network access domains. Overlay Acceleration offloads Virtualization Infrastructure instance vSwitch/vRouter or virtual termination endpoints (for applications that bypass the Virtual Infrastructure Layer).

Preferably, Application or Infrastructure acceleration can take benefit from underlying hardware acceleration and still be decoupled from it by using open multi-vendor API for Hardware Acceleration devices like for example:

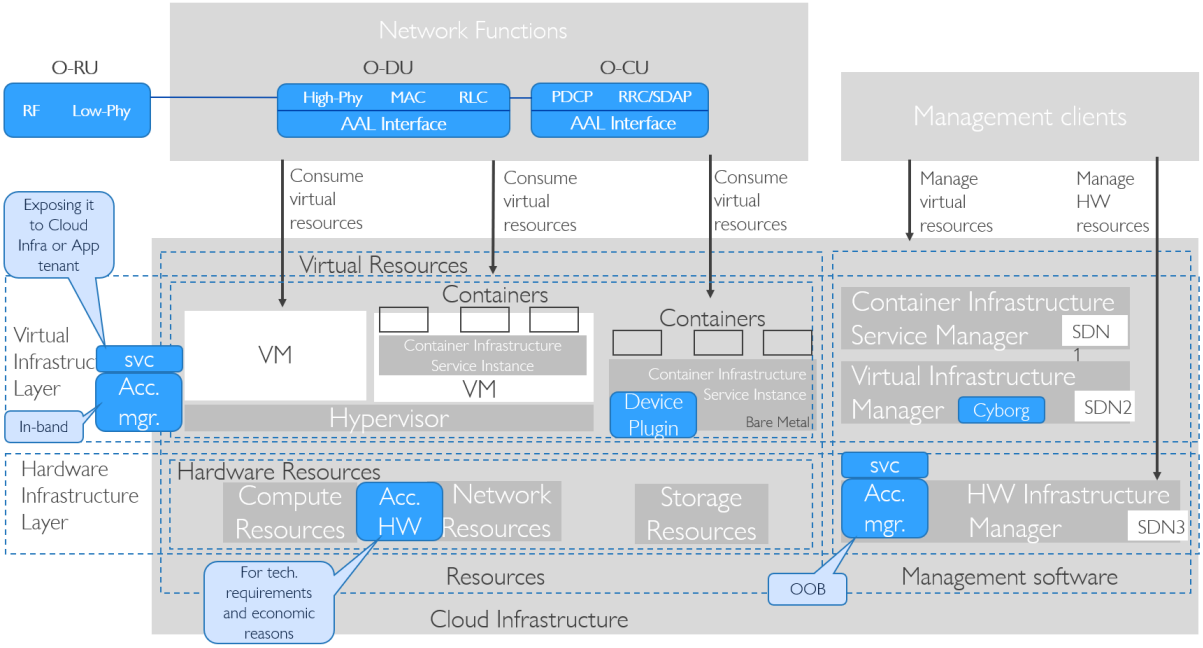
160. For Linux IO virtualization: VirtIO

161. For Network Functions using DPDK libraries: Crypto Device, EthDev, Event Device and Base Band Device

162. For O-RAN Network functions: O-RAN Acceleration Abstraction Layer Interface.

3.9.3 Example of O-RAN Acceleration Abstraction Layer Interface

O-RAN Alliance's Cloudification and Orchestration Workgroup (WG6) defines the Acceleration Abstraction Layer (AAL), an application-level interface, as the recommended way of decoupling software vendors' network functions from the different hardware accelerator implementations.



AAL Interface in RM Realization Diagram

The document "O-RAN Acceleration Abstraction Layer General Aspects and Principles 1.0" [36], and the latest O-RAN WG6 Cloudification and Orchestration Workgroup specifications [39]:

163. Describe the functions conveyed over the AAL interface, including configuration and management functions.

164. Identify the requirements as well as general procedures and operations.

165. Introduce the initial set of the O-DU/O-CU AAL profiles.

3.9.4 Workload Placement

Workload placement can be done by a combination of filters/selectors to find appropriate compute resources, subsystems to manage assignment of scheduled workloads to Hardware Accelerator, and intelligence in the workload to detect the presence of Hardware Accelerators.

For initial limited cloud deployments of network functions on private clouds it is possible to have a workload placement orchestrator that handles optimizations of selected virtualisation clusters and available hardware resources. This will however soon become too complex with the increasing number of acceleration devices, hardware composability and hybrid multi-cloud deployments.

Growing lists of individual optimizations including hardware acceleration during scheduling makes it more complex to map workloads to lists of individual optimizations, so such optimizations get grouped together into higher level categories. An example is having category

for real-time and data plane-optimized category instead of specifying individual optimizations required to reach it.

With further growth in size of clusters and the variety of hardware acceleration, in a hybrid or multi-cloud deployment, it will be necessary to enable separate optimization levels for the workload placement and each Cloud Infrastructure provider. The workload placement orchestrator will operate on one or several Cloud Infrastructures resources to satisfy the workloads according to Service Level Agreements (SLA) that do not specify all implementation and resource details. Each Cloud Infrastructure provider will make internal Infrastructure optimisations towards their own internal optimisation targets whilst fulfilling the SLAs.

3.9.5 CPU Instructions

The CPU architecture often includes instructions and execution blocks for most common compute-heavy algorithms like block cypher (example AES-NI), Random Number Generator or vector instructions. These functions are normally consumed in infrastructure software or applications by using enabled software libraries that run faster when custom CPU instructions for the execution of such functions are available in hardware and slower when these specific instructions are not available in hardware as only the general CPU instructions are used. Custom CPU instructions do not need to be activated or life-cycle-managed. When scheduling workloads, compute nodes with such custom CPU instructions can be found by applications or an orchestrator using OpenStack Nova filters or Kubernetes Node Feature Discovery labels, or directly from the Hardware Management layer.

3.9.6 Fixed Function Accelerators

Fixed function accelerators can come as adapters with in-line (typically PCIe adapter with Ethernet ports or storage drives) or look-aside (typically PCIe adapters without any external ports) functionality, additional chip on motherboard, included into server chipsets or packaged/embedded into main CPU. They can accelerate cryptographic functions, highly parallelized or other specific algorithms. Initial activation and rare life cycle management events (like updating firmware image) can typically be done from the Host OS (e.g. the OS driver or a Library), the Hardware Infrastructure Manager (from a library) or the NF (mostly through a library).

Beyond finding such compute nodes during scheduling workloads, those workloads also need to be mapped to the accelerator, both of which in Kubernetes can be done with Device Plugin framework. Once mapped to the application, the application can use enabled software libraries and/or device drivers that will use hardware acceleration. If hardware acceleration is used to improve cost/performance, then application can also run on generic compute node without hardware accelerator when application will use the same software library to run on generic CPU instructions.

3.9.7 Firmware-programmable Adapters

Firmware-programmable network adapters with programmable pipeline are types of network adapters where usual Ethernet controller functionality (accelerates common network overlays, checksums or protocol termination) can be extended with partially programmable modules so that additional protocols can be recognized, parsed and put into specific queues, which helps increase performance and reduce load on main CPU.

Firmware-programmable storage adapters can offload some of the storage functionality and include storage drive emulation to enable partial drive assignments up to the accessing host OS. These adapters can over time include more supported storage offload functions or support more drive emulation functions.

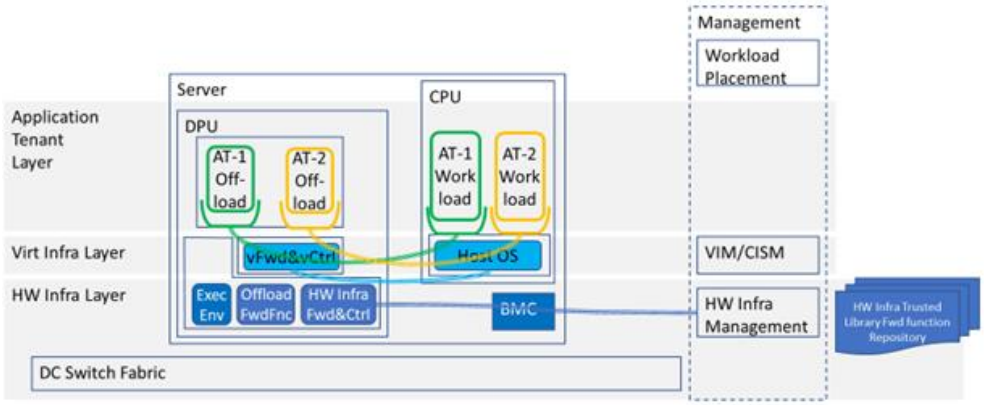
Before being used, such adapters have to be activated by loading programmable module that typically accelerates the Virtualization Infrastructure, so it is not often reprogrammed. Doing

this in multivendor environments can lead to complexities because the adapter hardware is typically specified, installed and supported by server vendor while the programmable image on the adapter is managed by SDN, Storage Controller or Software Infrastructure vendor.

3.9.8 SmartNICs

Programmable SmartNIC accelerators can come as programmable in-line adapters (typically PCIe adapter with Ethernet ports), or network connected pooled accelerators like farms of GPU or FPGA where the normal CPU PCIe connection is extended with an Ethernet hop.

There are two main types of Smart NICs that can accelerate network functions in-line between CPU and Ethernet ports of servers. The simpler types have a configurable or programmable packet pipeline that can implement offload for the infrastructure virtual switching or part of an application functions data plane. The more advanced type, often called Data Processing Unit (DPU), have a programmable pipeline and some strong CPU cores that simultaneously can implement underlay networking separation and trusted forwarding functions, infrastructure virtual switching data and control plane as well as part of an application functions control plane.



Example SmartNIC Deployment Model That Accelerates Two Workloads and Has OOB Management

3.9.8.1 Simple SmartNIC

The preferred usage of a simple SmartNIC is for the Virtualization Infrastructure usage that typically implements the data (forwarding) plane of the virtual switch or router. These deployments can offer a standardized higher-level abstract interface towards the application tenants such as VirtIO that supports good portability and is by that the preferred usage method.

Simple SmartNICs direct usage by the application tenant (VNF or CNF), where it acts as a dedicated accelerator appliance, require the application tenant to manage loading and the function that is loaded in the SmartNIC as well as any interface to the offloaded network functions. Such deployment is similar to the NIC PCI Pass-Through in that it bypasses the Virtualization Infrastructure layer's virtual switching, which require all network encapsulation, mapping and separation to be done by the underlay network, often by manual provisioning and therefore is not a preferred usage method.

3.9.8.2 DPU

The DPU can accelerate software infrastructure functions (vSwitch/vRouter) from the main CPU and simultaneously offer networking services e.g. load balancers, firewalls and application tenant offload functions. Through Out of band management it can also ensure underlay separation and map a selected part of the underlay network to the specific Virtualization Infrastructure instance that the server it is mounted on requires allowing them to be used on any statically provisioned underlay network.

The forwarding path (data plane) needs to be installed and controlled by the Hardware Infrastructure Manager through an isolated Out of band management channel into the DPU control and operating system completely out of reach for the main CPU Host SW. All content in the forwarding path must come from Hardware Infrastructure operator trusted code since any fault or malicious content can seriously disturb the whole network for all connected devices.

The trusted forwarding functions must be handled through a Hardware Infrastructure Management repository and have APIs for their respective control functions. These APIs must have an ability to handle some version differences since the forwarding and control planes life cycle management will not be atomic. The offload functions that should be offered as services must have published and preferably standardized open APIs, but the application specific forwarding functions do not have to be open APIs since they will only communicate with the application tenant provided control functions. P4 [67] and OpenConfig [68] are examples of suitable languages and models, with different levels of flexibility, usable for these forwarding and control functions.

The separated management channel could either come in through the BMC, a direct management port on the DPU or through a management VPN on the switch ports. This enable the Hardware Infrastructure Management to automate its networking through the DPU without any need to dynamically manage the switch fabric, thereby enabling a free choice of switch fabric vendor. These deployments allow the switch fabric to be statically provisioned by the operators networking operation unit, as it is often required.

The DPU can offload control and data plane of the virtual switching to the DPU as well as trusted hardware offload for virtualized Packet Core and Radio data plane networking and transport related functionality in a power efficient way. It can also offload relevant application tenant control functions if the DPU offers an Execution Environment for VMs or containers and there is space and performance headroom. In such cases the DPU must also setup a communication channel into respective application tenant environment.

3.9.9 Smart Switches

Smart Switches can be broadly categorized into Configurable Switches and Programmable Switches.

Configurable Smart Switches run generic “smart” configurable network operating system offering full range of network functionality and are flexible enough to support most network solutions. The most common such network operating system is Linux-based SONiC [69] allowing hardware and software disaggregation by running on switches from multiple switch vendors with different types of vendor fixed-function ASICs. Still, SONiC today cannot implement new type of data plane functionality or patch/modify/correct an ASIC, which is the type of support offered by programmable smart switches.

Programmable Smart Switches make it possible to quickly support new or correct/modify existing protocols and network functions, allow end customers to implement network functions, and to only implement and load functionality that is needed. Such switches contain one or more programmable switch ASICs of the same or different types. The two most used programming languages are P4 [67] and NPL [70], and both can be used with vendor-specific toolchains to program their switch ASICs and/or FPGAs. Open Networking Foundation Stratum [71] is an example of network operating system that offers generic life cycle management control services for the P4 components and a management API. The control API for the individual network functions are not part of the Stratum APIs.

Based on Smart Switches, products exist for fully integrated edge and fabric solutions from vendors like Arista, Cisco or Kaloom.

3.9.10 Decoupling Applications from Infrastructure and Platform with Hardware Acceleration

Decoupling common/glossary: cloud platform abstraction related terminology: applications from hardware accelerator is normally accomplished using drivers that, if available, are preferred with standardised interfaces across vendors and their products, or if not available then through drivers specific to the vendor hardware device. Decoupling infrastructure software from hardware accelerators is also preferred using standard interfaces. If those are not available for target hardware accelerator, coupling one or limited number of software infrastructures is less of an issue compared to coupling multiple applications.

Taking advantage of RM and RA environments with common capabilities, applications can be developed and deployed more rapidly, providing more service agility and easier operations. The extent to which this can be achieved will depend on levels of decoupling between application and infrastructure or platform underneath the application:

3.9.10.1 Infrastructure:

166. Application functionality or application control requires infrastructure components beyond RM profiles or
167. infrastructure configuration changes beyond APIs specified by RA. Generally, such an application is tightly coupled with the infrastructure which results in an Appliance deployment model (see common/glossary: cloud platform abstraction related terminology:).
168. Application control using APIs specified by RA finds nodes (already configured in support of the profiles) with
169. the required infrastructure component(s), and in that node using APIs specified by RA configures infrastructure components that make application work. Example is an application that to achieve latency requirements needs certain hardware acceleration available in RM profile and is exposed through APIs specified by RA.
170. Application control using APIs specified by RA finds nodes (already configured in support of the profiles) with
171. optional infrastructure component(s), and in these nodes using APIs specified by RA configures infrastructure component(s) that make application work better (like more performant) than without that infrastructure component. Example is an application that would have better cost/performance with certain acceleration adapter but can also work without it.
172. Application control using APIs specified by RA finds general profile nodes without any specific infrastructure
173. components.

3.9.10.2 Platform Services:

174. Application functionality or application control can work only with its own components instead of using defined
175. Platform Services. Example is an application that brings its own Load Balancer.
176. With custom integration effort, application can be made to use defined Platform Services. Example is application
177. that with custom integration effort can use defined Load Balancer which can be accelerated with hardware acceleration in way that is fully decoupled from application (i.e. application does not have awareness of Load Balancer being hardware-accelerated).

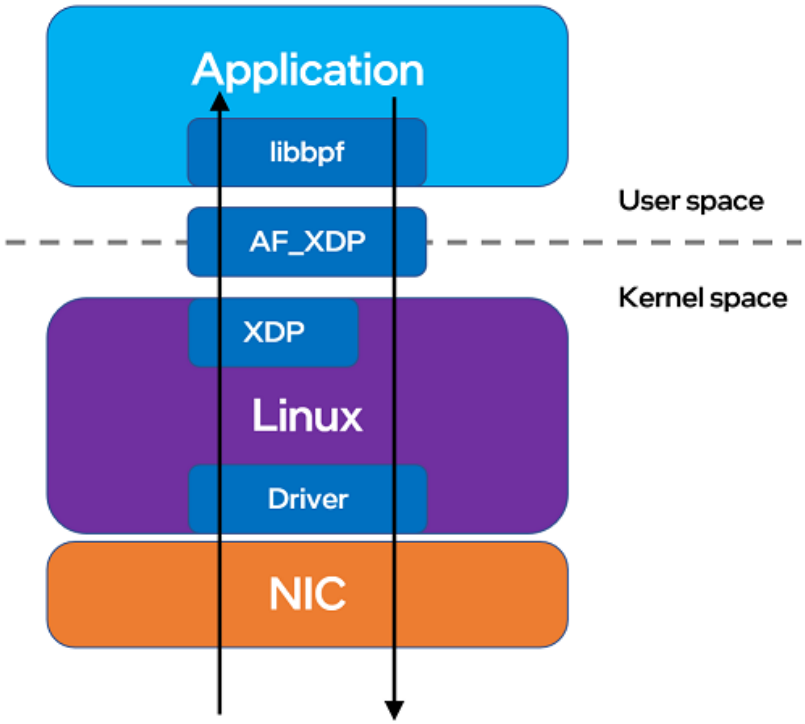
178. Application is designed and can be configured for running with defined Platform Services. Example is application

179. that can be configured to use defined Load Balancer which can be accelerated with hardware acceleration.

3.10 Address Family For XDP (AF_XDP)

Address Family For XDP (AF_XDP) [72] is optimized for high performance packet processing and introduced in Linux kernel v4.18. This new socket type leverages the eXpress Data Path (XDP) [73] in-kernel fast-path to transfer traffic frames from the NIC driver directly to user space without the need for full network stack. XDP is an Extended Berkley Packet Filter [74] (eBPF) software program.

By using the XDP_REDIRECT action from that XDP program, ingress frames can be redirected to other XDP-enabled network devices. The fastest working mode of operation is Zero-Copy mode in enabled XDP drivers.



AF_XDP Architecture

Linux-native applications can open an AF_XDP socket to receive raw packets directly from the NIC, by using libbpf [75] library functions to register a packet buffer area where packets will be located, and to create and bind the socket to a networking interface. DPDK-based applications can use AF_XDP Poll Mode Driver [76]. VPP-based applications can use AF_XDP Device Driver [77].

In virtualized environments AF_XDP could be used as interface between guest Kernel and user space application, but still need SR-IOV or virtio to get traffic to the VM.

3.11 Energy efficiency

Energy efficiency should be an overall requirement for the cloud infrastructure itself, the workloads hosted by this infrastructure, and the interface layer between them.

For telecommunication networks, energy efficiency is defined by ITU-T L.1330 [40] as "the relation between the useful output and energy consumption", the useful output being a metric

which represents the capacity provided by the service whose energy efficiency is assessed. As an example, the useful output of a traffic forwarding function can be the data volume forwarded (e.g., measured in Byte) and the assessment of its energy efficiency is then based on the ratio between this volume and the energy consumed for processing it (e.g., measured in Watt.hour) : Energy Efficiency (B/Wh) = Traffic Volume / Consumed Energy.

As elaborated in the Next Generation Alliance's whitepaper NGA Green G [47], with the global migration from 4G to 5G, one can observe the rise in data centre power consumption with the parallel reduction in the energy consumption of core network elements. This observation emphasizes the importance of the energy efficiency on the infrastructure and workload levels, and on the interface layer between them.

Examples of opportunities for the energy demand and cost reductions for the telecommunications operators are classified below.

180. Optimisation based on workload load demand, enhanced by AI-based smart monitoring

1. Smart sleep and shutdowns of elements of infrastructure
2. Adaptive power consumption
3. Time of usage optimisation
4. Multiple input and output muting
5. Cross base-station optimisation

181. Technology advancements based optimisation

1. 2G/3G legacy shutdowns
2. 3G/4G/5G optimisation
3. Cooling optimisation
4. Improved insulation

182. Energy source

1. Sustainable energy generation
2. Procurement of green energy
3. Fuel usage optimisation

In this Reference Model, the focus is on the first group, which is related to the optimisation on the workload to infrastructure interface level.

The method for assessing energy efficiency depends on the service targeted and the objectives. For NFV, ETSI proposes a method for production environment in ETSI EN 303 471 [41] and another one for laboratory one in ETSI ES 203 539 [42] (which is a common work with ITU-T which published as ITU-T L.1361 [43]).

Whatever the method and the service, it requires the cloud infrastructure to provide some **energy consumption metrics** for different parts of the infrastructure hardware (server, CPU etc.) as included in Internal Performance Measurement Capabilities. These metrics can be an amount of consumed energy (measured in Joule or Watt.hour) or a real-time power utilisation (measured in Watt or Joule/second) as proposed by DMTF Redfish DSP0268 2022.2 [16] which specifies metrics EnergykWh and PowerWatts for this purpose.

Some relevant information regarding NFV energy efficiency can also be found in Open RAN Technical Priority - Focus on Energy Efficiency (March 2022) [44] and QuEST Forum - NFV Workload Efficiency Whitepaper (October 2016) [45].

4 Infrastructure Capabilities, Measurements and Catalogue

4.1 Capabilities and performance measurements

This section describes the capabilities provided by the Cloud Infrastructure, and the performance measurements (PMs) generated by the Cloud Infrastructure (that is, without the use of external instrumentation).

The capability and PM identifiers conform to the following schema:

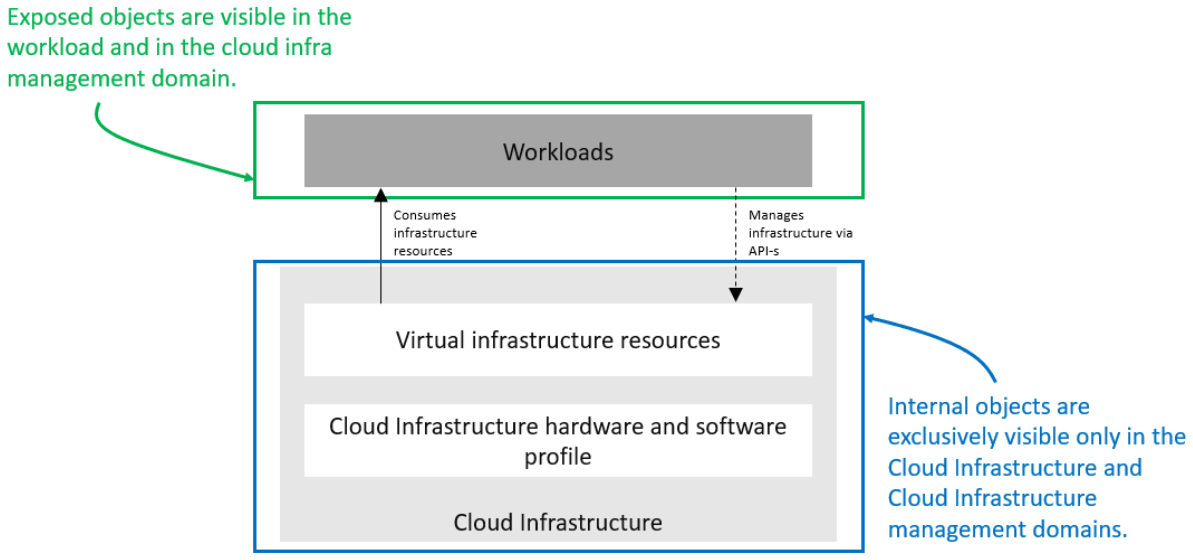
a.b.c (E.g., "e.pm.001")
a = Scope <(e)xternal | (i)nternal | (t)hird_party_instrumentation>
b = Type <(cap) capability | (man) management | (pm) performance | (man-pm)>
c = Serial Number

4.1.1 Exposed versus internal

The following definitions specify the context of the Cloud Infrastructure resources, capabilities and performance measurements (PMs).

Exposed: This refers to any object (for example, resource discovery/configuration/consumption, platform telemetry, interface, and so on) that exists in, or pertains to, the domain of the Cloud Infrastructure and is made visible, or exposed to, a workload. When an object is exposed to a given workload, the scope of visibility within a given workload is at the discretion of the workload's designer. From an infrastructure perspective, the infra-resident object is simply being exposed to one or more virtual environments, or workloads. It is the responsibility of the kernel, or the supervisor or executive within the resource instance (VM or container) to control how, when, and where the object is further exposed within the resource instance, with regard to permissions, security, and so on. An object is, by definition, visible within its domain of origin.

Internal: This is the opposite of exposed. Objects are exclusively available for use within the Cloud Infrastructure.



Exposed versus internal scope

As illustrated in the figure above, objects designated as internal are only visible within the area inside the blue oval (the Cloud Infrastructure), and only when the entity accessing the object has the appropriate permissions. Objects designated as exposed are potentially visible from both the area within the green oval (the Workloads), as well as from within the Cloud Infrastructure, again provided the entity accessing the object has the appropriate permissions.

Note

The figure above indicates the areas from where the objects are *visible*. It is not intended to indicate where the objects are *instantiated*. For example, the virtual resources are instantiated within the Cloud Infrastructure (the blue area), but are exposed, and are therefore *visible* to the workloads within the green area.

4.1.2 Exposed infrastructure capabilities

This section describes a set of exposed Cloud Infrastructure capabilities and performance measurements. These capabilities and PMs are well-known to workloads, as they provide the capabilities on which the workloads rely.

Note

The Cloud Infrastructure capabilities and measurements are expected to expand over time, as more capabilities are added, and the technology improves and matures.

4.1.2.1 Exposed resource capabilities

Table 4-1 below shows resource capabilities of the Cloud Infrastructure available to the workloads.

Ref	Cloud Infrastructure capability	Unit	Definition/Notes
e.cap.001	# vCPU	number	The maximum number of vCPUs that can be assigned to a single VM or Pod (1).
e.cap.002	RAM Size	MB	The maximum memory, in MB, that can be assigned to a single VM or Pod by the Cloud Infrastructure (2).
e.cap.003	Total per-instance (ephemeral) storage	GB	The maximum storage, in GB, that can be assigned to a single VM or Pod by the Cloud Infrastructure.
e.cap.004	# Connection points	number	The maximum number of connection points that can be assigned to a single VM or Pod by the Cloud Infrastructure.
e.cap.005	Total external (persistent) storage	GB	The maximum storage, in GB, that can be attached or mounted to a VM or Pod by the Cloud Infrastructure.

Table 4-1: Exposed resource capabilities of the Cloud Infrastructure

- In a Kubernetes-based environment, this means the CPU limit of a pod.
- In a Kubernetes-based environment, this means the memory limit of a pod.

4.1.2.2 Exposed performance optimisation capabilities

Table 4-2 lists the performance optimisation capabilities exposed to the workloads by the Cloud Infrastructure.

Ref	Cloud Infrastructure capability	Unit	Definition/Notes
e.cap.006	CPU pinning	Yes/No	Indicates if the Cloud Infrastructure supports CPU pinning.

e.cap.007	NUMA alignment	Yes/No	Indicates if the Cloud Infrastructure supports NUMA alignment.
e.cap.008	IPSec Acceleration	Yes/No	IPSec acceleration.
e.cap.009	Crypto Acceleration	Yes/No	Crypto acceleration.
e.cap.010	Transcoding Acceleration	Yes/No	Transcoding acceleration.
e.cap.011	Programmable Acceleration	Yes/No	Programmable acceleration.
e.cap.012	Enhanced Cache Management	Yes/No	If supported, L=Lean; E=Equal; X=eXpanded. L and X cache policies require CPU pinning to be active.
e.cap.013	SR-IOV over PCI-PT	Yes/No	Traditional SR-IOV. These capabilities generally require hardware-dependent drivers to be injected into the workloads.
e.cap.014	GPU/NPU	Yes/No	Hardware coprocessor. These capabilities generally require hardware-dependent drivers to be injected into the workloads.
e.cap.015	SmartNIC	Yes/No	Network acceleration.
e.cap.016	FPGA/other Acceleration HW	Yes/No	These capabilities generally require hardware-dependent drivers to be injected into the workloads.
e.cap.023	Huge pages	Yes/No	Indicates if the Cloud Infrastructure supports huge pages.
e.cap.024	CPU allocation ratio	Yes/No	N:1: Denotes the number of virtual cores per physical core. It is also known as the CPU overbooking ratio.
e.cap.025	AF_XDP	Yes/No	Indicates whether or not the Cloud Infrastructure supports AF_XDP.
e.cap.026	Real-Time	Yes/No	Indicates whether the Cloud Infrastructure supports real-time versions and/or configuration for BIOS, kernel and OS services.
e.cap.027	Time Sensitive Networking	Yes/No	Indicates whether the Cloud Infrastructure node supports timing accuracy with PTP and synchronization with SyncE.

Table 4-2: Exposed performance optimisation capabilities of the Cloud Infrastructure

Enhanced Cache Management is a compute performance enhancer that applies a cache management policy to the socket hosting a given virtual compute instance, provided the associated physical CPU microarchitecture supports it. A cache management policy can be used to specify the static allocation of cache resources to cores within a socket. The "Equal" policy distributes the available cache resources equally across all of the physical cores in the socket. The "eXpanded" policy provides additional resources to the core pinned to a workload that has the "X" attribute applied. The "Lean" attribute can be applied to workloads which do not realise significant benefit from a marginal cache size increase and are therefore willing to relinquish unneeded resources.

In addition to static allocation, an advanced Reference Architecture implementation can implement dynamic cache management control policies, operating with tight (~ms) or standard (tens of seconds) control loop response times, thereby achieving higher overall performance for the socket.

4.1.2.3 Exposed monitoring capabilities

Monitoring capabilities are used for the passive observation of workload-specific traffic traversing the Cloud Infrastructure. As with all capabilities, monitoring may be unavailable or intentionally disabled for security reasons in a given Cloud Infrastructure deployment. If this functionality is enabled, it must be subject to strict security policies. For further details, see the Reference Model Security chapter.

Table 4-3 shows the possible monitoring capabilities available from the Cloud Infrastructure for the workloads.

Ref	Cloud Infrastructure capability	Unit	Definition/notes
e.cap.017	Monitoring of L2-7 data	Yes/No	Ability to monitor L2-L7 data from the workload.

Table 4-3: Exposed monitoring capabilities of the Cloud Infrastructure

Table 4-4: Place holder

4.1.3 Internal infrastructure capabilities

This section covers a list of implicit Cloud Infrastructure capabilities and measurements. These capabilities and metrics are hidden from the workloads (that is, workloads may not know about them). However, they will impact the overall performance and capabilities of a given Cloud Infrastructure solution.

Note

The implicit Cloud Infrastructure capabilities and metrics are expected to evolve with time, as more capabilities are added as technology improves and matures.

4.1.3.1 Internal resource capabilities

Table 4-5 shows the resource capabilities of the Cloud Infrastructure. These include the capabilities offered to the workloads and resources consumed internally by the Cloud Infrastructure.

Ref	Cloud Infrastructure capability	Unit	Definition/notes
i.cap.014	CPU cores consumed by the Cloud Infrastructure overhead on a worker (compute) node	%	The ratio, expressed as a percentage, of cores consumed by the Cloud Infrastructure components (including the host OS) in a compute node to the total number of cores available.
i.cap.015	Memory consumed by the Cloud Infrastructure overhead on a worker (compute) node	%	The ratio, expressed as a percentage, of memory consumed by the Cloud Infrastructure components (including host OS) in a worker (compute) node to the total available memory.

Table 4-5: Internal resource capabilities of the Cloud Infrastructure

4.1.3.2 Internal SLA capabilities

Table 4-6 below shows the Service-Level Agreement (SLA) capabilities of the Cloud Infrastructure. These include Cloud Infrastructure capabilities required by workloads as well as required internal to Cloud Infrastructure. Application of these capabilities to a given workload is determined by its Cloud Infrastructure Profile.

Ref	Cloud Infrastructure Capability	Unit	Definition/Notes
i.cap.017	Connection point QoS	Yes/No	QoS enablement of the connection point (vNIC or interface)

Table 4-6: Internal SLA capabilities to Cloud Infrastructure

4.1.3.3 Internal Performance Measurement Capabilities

Table 4-8 shows possible performance measurement capabilities for the Cloud Infrastructure. The availability of these capabilities will be determined by the Cloud Infrastructure Profile used by the workloads. These measurements or events should be collected and monitored by monitoring tools.

Table 7Internal Measurement Capabilities of Cloud Infrastructure

Ref	Cloud Infrastructure Capability	Unit	Definition/Notes
i.pm.001	Host CPU usage	nanoseconds	Per Compute node. It maps to ETSI GS NFV-TST 008 V3.5.1 [10] processor usage metric (Cloud Infrastructure internal).
i.pm.002	Virtual compute resource (vCPU) usage	nanoseconds	Per VM or Pod. It maps to ETSI GS NFV-IFA 027 v2.4.1 [11] Mean vCPU usage and Peak vCPU usage (Cloud Infrastructure external).
i.pm.003	Host CPU utilisation	%	Per Compute node. It maps to ETSI GS NFV-TST 008 V3.5.1 [10] processor usage metric (Cloud Infrastructure internal).
i.pm.004	Virtual compute resource (vCPU) utilisation	%	Per VM or Pod. It maps to ETSI GS NFV-IFA 027 v2.4.1 [11] Mean vCPU usage and Peak vCPU usage (Cloud Infrastructure external).
i.pm.005	Network metric, Packet count	Number of packets	Number of successfully transmitted or received packets per physical or virtual interface, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.006	Network metric, Octet count	8-bit bytes	Number of 8-bit bytes that constitute successfully transmitted or received packets per physical or virtual interface, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].

i.pm.007	Network metric, Dropped Packet count	Number of packets	Number of discarded packets per physical or virtual interface, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.008	Network metric, Errored Packet count	Number of packets	Number of erroneous packets per physical or virtual interface, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.009	Memory buffered	KiB	Amount of temporary storage for raw disk blocks, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.010	Memory cached	KiB	Amount of RAM used as cache memory, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.011	Memory free	KiB	Amount of RAM unused, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.012	Memory slab	KiB	Amount of memory used as a data structure cache by the kernel, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.013	Memory total	KiB	Amount of usable RAM, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.014	Storage free space	Bytes	Amount of unused storage for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.015	Storage used space	Bytes	Amount of storage used for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.016	Storage reserved space	Bytes	Amount of storage reserved for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.017	Storage Read latency	Milliseconds	Average amount of time to perform a Read operation for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.018	Storage Read IOPS	Operations per second	Average rate of performing Read operations for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.019	Storage Read Throughput	Bytes per second	Average rate of performing Read operations for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1.
i.pm.020	Storage Write latency	Milliseconds	Average amount of time to perform a Write operation for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1.
i.pm.021	Storage Write IOPS	Operations per second	Average rate of performing Write operations for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].

i.pm.022	Storage Write Throughput	Bytes per second	Average rate of performing Write operations for a given storage system, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.023	Host power utilization	Watt (Joule/s)	Real-time electrical power used by a node (1).
i.pm.024	Host energy consumption	Watt.hour (Joule)	Electrical energy consumption of a node since the related counter was last reset (2).
i.pm.025	CPU power utilization	Watt (Joule/s)	Real-time electrical power used by the processor(s) of a node (1).
i.pm.026	CPU energy consumption	Watt.hour (Joule)	Electrical energy consumption of the processor(s) of a node since the related counter was last reset (2).
i.pm.027	PCIe device power utilization	Watt (Joule/s)	Real-time electrical power used by a specific PCI device of a node (1).
i.pm.028	PCIe device energy consumption	Watt.hour (Joule)	Electrical energy consumption of a specific PCI device of a node since the related counter was last reset (2).
i.pm.029	RAM power utilization	Watt (Joule/s)	Real-time electrical power used by the memory of a node (1).
i.pm.030	RAM energy consumption	Watt.hour (Joule)	Electrical energy consumption of the memory of a node since the related counter was last reset (2).
i.pm.031	Disk power utilization	Watt (Joule/s)	Real-time electrical power used by a specific storage device of a node (1).
i.pm.032	Disk energy consumption	Watt.hour (Joule)	Electrical energy consumption of a specific storage device of a node since the related counter was last reset (2).
i.pm.033	Hugepages pool total	Integer	The number of Hugepages currently configured in the pool, which is the total of pages available, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.034	Hugepages used	Integer	The number of used pages in the Hugepage Pool, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].
i.pm.035	Hugepages free	Integer	The number of free pages in the Hugepage Pool, as defined in ETSI GS NFV-TST 008 V3.5.1 [10].

Table 4-8: Internal Measurement Capabilities of Cloud Infrastructure

(1) for example, relying on PowerWatts metrics as defined by DMTF Redfish specification DSP0268 2022.2 [11], provided by a sensor metering "the arithmetic mean of product terms of instantaneous voltage and current values measured over integer number of line cycles for a circuit, in watt units"

(2) for example, relying on EnergykWh metrics as defined by DMTF Redfish specification DSP0268 2022.2 [11], provided by a sensor metering "the energy, integral of real power over time" reflecting "the power consumption since the sensor metrics were last reset"

4.1.4 Cloud Infrastructure Management Capabilities

The Cloud Infrastructure Manager (CIM) is responsible for controlling and managing the Cloud Infrastructure compute, storage, and network resources. Resources are dynamically allocated based on workload requirements. This section covers the list of capabilities offered by the CIM to workloads or service orchestrator.

Table 4-9 shows capabilities related to resources allocation.

Ref	Cloud Infrastructure Capability	Unit	Definition/Notes
e.man.001	Virtual Compute allocation	Yes/No	Capability to allocate virtual compute resources to a workload
e.man.002	Virtual Storage allocation	Yes/No	Capability to allocate virtual storage resources to a workload
e.man.003	Virtual Networking resources allocation	Yes/No	Capability to allocate virtual networking resources to a workload
e.man.004	Multi-tenant isolation	Yes/No	Capability to isolate resources between tenants
e.man.005	Images management	Yes/No	Capability to manage workload software images
e.man.010	Compute Availability Zones	list of strings	The names of each Compute Availability Zone that was defined to separate failure domains
e.man.011	Storage Availability Zones	list of strings	The names of each Storage Availability Zone that was defined to separate failure domains

Table 4-9: Cloud Infrastructure Management Resource Allocation Capabilités

4.1.5 Cloud Infrastructure Management Performance Measurements

Table 4-10 shows performance measurement capabilities.

Ref	Cloud Infrastructure Capability	Unit	Definition/Notes
e.man.006	Virtual resources inventory per tenant	Yes/No	Capability to provide information related to allocated virtualised resources per tenant
e.man.007	Resources Monitoring	Yes/No	Capability to notify state changes of allocated resources
e.man.008	Virtual resources Performance	Yes/No	Capability to collect and expose performance information on virtualised resources allocated
e.man.009	Virtual resources Fault information	Yes/No	Capability to collect and notify fault information on virtualised resources

Table 4-10: Cloud Infrastructure Management Performance Measurement Capabilities

4.1.5.1 Resources Management Measurements

Table 4-11 shows resource management measurements of CIM as aligned with ETSI GR NFV IFA-012 [20]. The intention of this table is to provide a list of measurements to be used

in the Reference Architecture specifications, where the values allowed for these measurements in the context of a particular Reference Architecture will be defined.

Ref	Cloud Infrastructure Management Measurement	Unit	Definition/Notes
e.man-pm.001	Time to create Virtual Compute resources (VM/container) for a given workload Time to delete Virtual Compute resources (VM/container) of a given workload Time to start Virtual Compute resources (VM/container) of a given workload Time to stop Virtual Compute resources (VM/container) of a given workload Time to pause Virtual Compute resources (VM/container) of a given workload Time to create internal virtual network Time to delete internal virtual network Time to update internal virtual network Time to create external virtual network Time to delete external virtual network Time to update external virtual network Time to create external storage ready for use by workload	Max ms	
e.man-pm.002		Max ms	
e.man-pm.003		Max ms	
e.man-pm.004		Max ms	
e.man-pm.005		Max ms	
e.man-pm.006		Max ms	
e.man-pm.007		Max ms	
e.man-pm.008		Max ms	
e.man-pm.009		Max ms	
e.man-pm.010		Max ms	
e.man-pm.011		Max ms	
e.man-pm.012		Max ms	

Table 4-11: Cloud Infrastructure Resource Management Measurements

4.1.6 Acceleration/offload API requirements

Hardware accelerators and offload functions with abstracted interfaces are preferred and can functionally be interchanged. However, their characteristics might vary. It is also likely that the CNFs/VNFs and the Cloud Infrastructure have certification requirements for the implementations. A software implementation of these functions is also often required to have the same abstracted interfaces for the deployment situations when there are no more hardware accelerator or offload resources available.

For accelerators and offload functions with externally exposed differences in their capabilities or management functionality, these differences must be clear through the management API, either explicitly for the differing functions or implicitly through the use of unique APIs.

Regardless of the exposed or internal capabilities and characteristics, the operators generally require a choice of implementations for accelerators and offload function realisation, and, therefore, the need for ease of portability between implementations and vendors.

The following table of requirements is derived from the needs of VNF/CNF applications, Cloud Infrastructure, and Telco Operators to have multiple realisations of hardware acceleration and offload functions that can also be implemented through the software, when no special hardware is available. These requirements should be adopted in the Reference Architectures to ensure that the different implementations on the market are as aligned as possible in their interfaces, and that the hardware acceleration and offload functions enjoy an efficient ecosystem of accelerators that compete on their technical merits, and not through obscure or proprietary interfaces.

Table 4-12 shows the acceleration/offload API capabilities.

Ref	Acceleration/offload API capability	Unit	Definition/notes
e.api.001	VNF/CNF usage of accelerator standard i/f	Yes/No	The VNF/CNF uses abstracted standardised interfaces to the acceleration/offload functions. This enables use of hardware and software implementations of the accelerated/offloaded functions from multiple vendors in the Cloud Infrastructure.
e.api.002	Virtualisation infrastructure SW usage of accelerator standard i/f	Yes/No	The virtualisation infrastructure software uses abstracted standardised interfaces to the hardware acceleration/offload function. This enables multiple hardware and software implementations in the hardware infrastructure layer of the accelerated functions from multiple vendors.
e.api.003	Accelerators offering standard i/f to the hardware infra layer	Yes/No	The acceleration/offload functions offer abstracted standardised interfaces for the virtualisation infrastructure and the VNF/CNF applications.
e.api.004	Accelerators offering virtualised functions	Yes/No	Acceleration/Offload functions for VNFs/CNFs are virtualised to allow multiple VNFs/CNFs to use the same Acceleration/Offload instance.
e.api.005	VNF/CNF accelerator management functions access rights	Yes/No	The VNF/CNF management functions are able to request acceleration/offload invocation without requiring elevated access rights.
e.api.006	Accelerators offering standard i/f to VNF/CNF management	Yes/No	VNF/CNF management functions are able to request acceleration/offload invocation through abstracted standardised management interfaces.
e.api.007	VNFs/CNFs and virtualisation infrastructure accelerator portability	Yes/No	The VNFs/CNFs and virtualisation infrastructure software is designed to manage multiple types of accelerator or offload function realisations, even when their differences are exposed to the infrastructure or the applications layers.
e.api.008	VNFs/CNFs and virtualisation infrastructure accelerator flexibility	Yes/No	The VNFs/CNFs and virtualisation infrastructure software is able to use any assigned instance and type of accelerator or offload function for which they are certified.

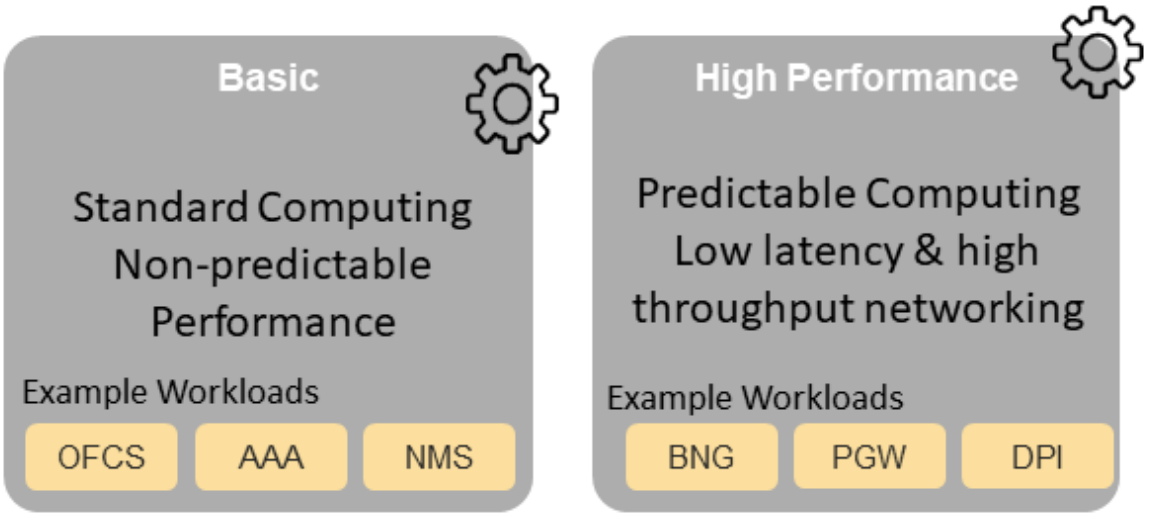
Table 4-12: Acceleration/offload API capabilities

4.2 Profiles and workload flavours

Section 4.1 Capabilities and performance measurements enumerates the different capabilities exposed by the infrastructure resources. Not every workload is sensitive to all the listed

capabilities of the cloud infrastructure. In section 2, the analysis of the use cases led to the definition of two Profiles (top-level partitions) and the need for specialisation through Profile extensions (specialisations). Profiles and Profile Extensions are used to configure the cloud infrastructure nodes. They are also used by workloads to specify the infrastructure capabilities on which they need to run. Workloads specify the flavours and additional capabilities information.

In this section we will specify the capabilities and features associated with each of the defined profiles and extensions. Each profile (for example, Cloud infrastructure Profiles) and each extension associated with that profile, specifies a predefined standard set of infrastructure capabilities that workload vendors can use to build their workloads for deployment on conformant cloud infrastructure. A workload can use several profiles and associated extensions to build its overall functionality, as discussed below.



Cloud infrastructure profiles

The two profiles-profile-extensions--flavours are as follows:

Basic (B): for Workloads that can tolerate resource over-subscription and variable latency. High Performance (H): for Workloads that require predictable computing performance, high n etwork throughput and low network latency.

The availability of these two profiles facilitates and accelerates the workload deployment. The intent of the above profiles is to match the Cloud Infrastructure to the most common needs of the workloads, and to allow for a more comprehensive configuration using profile extensions when needed. These profiles are offered with extensions, that specify capability deviations and allow for the specification of more capabilities. The Cloud Infrastructure will have nodes configured as with options, such as virtual interface options, storage extensions, and acceleration extensions.

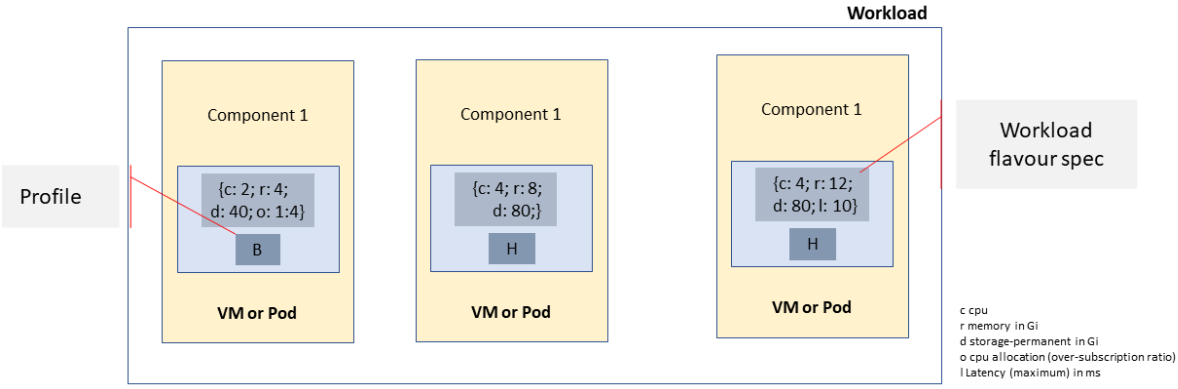
The justification for defining these two profiles, and a set of extensible profile extensions, is provided in the section profiles-profile-extensions--flavours. It includes the following:

183. Workloads can be deployed by requesting compute hosts configured according to a specific profile (basic or high performance).
184. Profile extensions allow a more granular compute host configuration for the workload (such as GPU, high-speed network, and Edge deployment).
185. Cloud infrastructure "scattering" is minimised.

186. Workload development and testing optimisation by using predefined and commonly supported (Telco operators) profiles and extensions.

187. Better usage of cloud objects (memory, processor, network, and storage).

Workload flavours specify the resource sizing information including network and storage (size, throughput, and IOPS). Workloads built against Cloud Infrastructure Profiles and Workload Flavours shows three resources (VM or Pod) on nodes configured in accordance with the specified profile ('B' and 'H'), and the resource sizes.



Workloads built against Cloud Infrastructure profiles and workload flavours

A node configuration can be specified using the following syntax:

<profile name>[.<profile_extension>][.<extra profile specs>]

In this syntax, the specifications enclosed within the square brackets ([and]) are optional. The 'extra profile specs' are needed to capture the special node configurations not accounted for by the profile and profile extensions.

Example node configurations can be as follows: B, B.low-latency, H, and H.very-high-speed-network.very-low-latency-edge.

A workload needs to specify the configuration and capabilities of the infrastructure that it can run on, the size of the resources it needs, and additional information (extra-specs), such as whether or not the workload can share core siblings (SMT thread), whether or not it has affinity (that is, it needs to be placed on the same infrastructure node) with other workloads, and so on. The capabilities required by the workload can therefore be specified using the following syntax:

<profile name>[.<profile_extension>][.<extra profile specs>].<workload flavour specs>[.<extra-specs>]

In this syntax, the <workload flavour specs> are specified as defined in section 4.2.4.3 Workload Flavours and Other Capabilities Specifications Format <#workload-flavours-and-other-capabilities-specifications-format>__ below.

4.2.1 Profiles

4.2.1.1 Basic profile

Hardware resources are configured in accordance with the Basic profile (B) in such a way that they are only suited for workloads that tolerate variable performance. This includes latency and resource oversubscription. Only Simultaneous Multi-Threading (SMT) if available is configured on nodes that support the Basic profile. With no NUMA alignment, the executing processes of the vCPUs may not be on the same NUMA node as the memory used by these processes. When the vCPU and memory are on different NUMA nodes, memory accesses are not local to the vCPU node and therefore add latency to memory accesses. The Basic

profile supports oversubscription (using the CPU Allocation Ratio) which is specified as part of the sizing information in the workload profiles.

4.2.1.2 High-performance profile

The high-performance profile (H) is intended to be used for workloads that require predictable performance, high network throughput requirements, and/or low network latency. To satisfy predictable performance needs, NUMA alignment, CPU pinning, and huge pages are enabled. For obvious reasons, the high-performance profile does not support oversubscription.

4.2.2 Profiles specifications and capability mapping

Ref	Capability	Basic	High performance	Notes
e.cap.006	CPU pinning	No	Yes	Exposed performance capabilities according to Table 4-2.
e.cap.007	NUMA alignment	No	Yes	
e.cap.013	SR-IOV over PCI-PT	No	Yes	
e.cap.018	Simultaneous Multithreading (SMT)	Yes if SMT is supported	Optional	
e.cap.019	vSwitch optimisation (DPDK)	No	Yes	DPDK does not have to be used if another network acceleration method is being used.
e.cap.020	CPU Architecture	<value>	<value>	Values, such as x64, ARM, and so on.
e.cap.021	Host operating system (OS)	<value>	<value>	Values, such as a specific Linux version, Windows version, and so on.
e.cap.022	Virtualisation infrastructure Layer1	<value>	<value>	Values, such as KVM, Hyper-V, Kubernetes, and so on, when relevant, depending on technology.
e.cap.023	Huge page support according to Table 4-2.	No	Yes	Internal performance capabilities, according to Table 4-2.
e.cap.025	AF_XDP	No	Optional	These capabilities require workload support for the AF_XDP socket type.
i.cap.019	CPU clock speed	<value>	<value>	This capability specifies the Cloud Infrastructure CPU clock speed, in GHz.
i.cap.020	Storage encryption	Yes	Yes	This capability specifies whether or not the Cloud Infrastructure supports storage encryption.

Table 4-13: Profiles specifications and capability mapping

4.2.3 Profile extensions

Profile extensions represent small deviations from, or further qualification of, the profiles that do not require the partitioning of the infrastructure into separate pools, but which have specifications with a finer granularity of the profile. Profile Extensions provide workloads with a more granular control over what kind of infrastructure they can run on.

Table 8Profile extensions

Profile Extension Name	Mnemonic	Applicable to the basic profile	Applicable to the high performance profile	Description	Notes
Compute-intensive high-performance CPU	compute-high-perf-cpu	✗	✓	Nodes that have predictable computing performance and higher clock speeds.	May use vanilla VIM/K8S scheduling instead.
Storage-intensive high-performance storage	storage-high-perf	✗	✓	Nodes that have low storage latency and/or high-storage IOPS.	
Compute-intensive high memory	compute-high-memory	✗	✓	Nodes that have high amounts of RAM.	May use vanilla VIM/K8S scheduling instead.
Compute-intensive GPU	compute-gpu	✗	✓	For compute-intensive workloads that require GPU compute resources on the node.	May use node feature discovery.
Network-intensive	high-speed-network	✗	✓	Nodes configured to support SR-IOV.	
Network-intensive high-speed network (25G)	high-speed-network	✗	✓	Denotes the presence of network links (to the DC network) of speeds of 25 Gbps or more on the node.	

Network-intensive high-speed speed network (100G)	very-high-speed-network	✗	✓	Denotes the presence of network links (to the DC network) of speeds of 100 Gbps or more on the node.
Low latency - Edge sites	low-latency-edge	✓	✓	Labels a host or node as located in an Edge site, for workloads requiring low latency (specify value) to final users or geographical distribution.
Very low latency - Edge sites	very-low-latency-edge	✓	✓	Labels a host or node as located in an Edge site, for workloads requiring low latency (specify value) to final users or geographical distribution.
Ultra-low low latency - Edge sites	ultra-low-latency-edge	✓	✓	Labels a host or node as located in an Edge site, for workloads requiring low latency (specify value) to final users or geographical distribution.
Fixed-function accelerator	compute-ffa	✗	✓	Labels a host or node that includes a consumable fixed-function accelerator (non-programmable, for example, Crypto, vRAN-specific adapter).

Firmware-programmable adapter	compute-firmware programmable	✗	✓	Labels a host or node that includes a consumable firmware-programmable adapter (for example, Network/storage adapter).
SmartNIC-enabled	network-smartnic	✗	✓	Labels a host or node that includes a programmable accelerator.
SmartSwitch-enabled	network-smartswitch	✗	✓	Labels a host or node that is connected to a programmable switch fabric or a TOR switch.

Table 4-14: Profile extensions

4.2.4 Workload flavours and specifications of other capabilities

The workload requests a set of resource capabilities, including its components, that it needs to run successfully. The GSMA document OPG.02 "Operator Platform: Requirements and Architecture" [35] defines "Resource Flavour" as this set of capabilities. A Resource Flavour specifies the resource profile, any profile extensions, and the size of the resources needed (workload flavour), as well as extra specifications for workload placement, as defined in Section 4.2 Profiles and Workload Flavours above.

This section provides details of the capabilities that need to be provided in a resource request. The profiles, the profile specifications, and the profile extensions specify the infrastructure (hardware and software) configuration. In a resource request, they need to be augmented with workload-specific capabilities and configurations, including the sizing of requested resource, extra specifications including those related to the placement of the workload section 4.2.4.2, virtual network section 4.2.5, and storage extensions section 4.2.6.

4.2.4.1 Geometry of workload flavours (Sizing)

Workload flavours (sometimes also referred to as “compute flavours”) are sizing specifications beyond the capabilities specified by the node profiles. Workload flavours represent the compute, memory, storage, and network resource sizing templates used in requesting resources on a host that is conformant with the profiles and profile extensions. The workload flavour specifies the compute, memory, and storage characteristics of the requested resource (VM, container). Workload Flavours can also specify different storage resources, such as ephemeral storage, swap disk, network speed, and storage IOPs.

Workload flavour sizing consists of the following:

Element	Mnemonic	Description
cpu	c	Number of virtual compute resources (vCPUs).
memory	r	Virtual resource instance memory, in megabytes.

storage - ephemeral	e	Specifies the size of an ephemeral/local data disk that exists only for the life of the instance. The default value is 0. The ephemeral disk may be partitioned into boot (base image) and swap space disks.
storage - persistent	d	Specifies the disk size of persistent storage.

Table 4-15: Workload flavour geometry specification.

The flavours' syntax consists of <element, value> pairs, separated by a colon (":"), for example: {cpu: 4; memory: 8 Gi; storage-permanent: 80 Gi}.

4.2.4.2 Specifications of the extra capabilities of the workloads

Besides the sizing information, a workload may need to specify additional capabilities. These include capabilities for workload placement, such as latency, and workload affinity and non-affinity. They also include capabilities such as workload placement on multiple NUMA nodes. The extra specifications include the Virtual Network Interface Specifications and Storage Extensions.

Attribute	Description
CPU allocation ratio	This attribute specifies the maximum CPU allocation (a.k.a. oversubscription) ratio supported by a workload.
Compute-intensive	This attribute is for demanding workloads with stringent memory access requirements, where the single NUMA bandwidth may be a limitation. The compute-intensive workload profile is used to enable the workload to be spread across all NUMA nodes.
Latency	This attribute specifies the latency requirements used for locating workloads.
Affinity	This attribute specifies the workloads that should be hosted on the same computer node.
Non-affinity	This attribute specifies workloads that should not be hosted on the same computer node.
Dedicated cores	This attribute specifies whether or not the workload can share sibling threads with other workloads. The default is No such that it allows different workloads on different threads.
Network interface option	See Section 4.2.5.
Storage extension	See Section 4.2.6.

4.2.4.3 Format of the workload flavours and other capability specifications

The following table shows a complete list of the specifications that need to be specified by the workloads.

Table 9Specifications of resource flavours

Profile extension name	Mnemonic	Applicable to the basic profile	Applicable to the high performance profile	Description	Notes
CPU	c	✅	✅	The number of virtual compute resources (vCPUs).	Required
memory	r	✅	✅	The virtual resource instance memory, in megabytes.	Required
storage - ephemeral	e	✅	✅	This profile extension specifies the size of an ephemeral/local data disk that exists only for the life of the instance. The default value is 0. The ephemeral disk may be partitioned into boot (base image) and swap space disks.	Optional
storage - persistent	d	✅	✅	This profile extension specifies the disk size of persistent storage.	Required
storage - root disk	b	✅	✅	This profile extension specifies the size of the root disk.	Optional
CPU Allocation Ratio	o	✅	❌	This profile extension specifies the CPU allocation (or oversubscription) ratio. It can only be specified for the basic profile. For workloads that utilise nodes configured according to the high-performance profile, the CPU allocation ratio is 1:1.	Required for Basic profile
Compute-intensive	ci	✅	❌	This profile applies to demanding workloads with stringent memory access requirements, where the single NUMA bandwidth maybe a bandwidth. The compute-intensive workload profile is used to enable the workload to be spread across all NUMA nodes.	Optional

Latency	l	✓	✓	This profile specifies the latency requirements used for locating workloads.	Optional
Affinity	af	✓	✓	This profile specifies the workloads that should be hosted on the same computer node.	Optional
Non-affinity	naf	✓	✓	This profile specifies the workloads that should not be hosted on the same computer node.	Optional
Dedicate cores	dc	✓	✗	This profile specifies whether or not the workload can share sibling threads with other workloads. The default value is No, thereby allowing different workloads on different threads.	Optional
Network interface option	dc	✓	✓	See below.	Optional
Storage extension	s	✓	✓	See below.	Optional
Profile name	pn	✓	✓	This profile specifies profile "B" or "H".	Required
Profile extension	pe	✗	✓	This profile specifies the profile extensions.	Optional
Profile extra specs	pes	✗	✓	This profile specifies the special node configurations not accounted for by the profile and the profile extensions.	Optional

Table 4-16: Specifications of resource flavours (complete list of workload capabilities)

4.2.5 Virtual network interface specifications

The virtual network interface specifications extend a flavour customisation with network interfaces with an associated bandwidth. They are identified by the literal “n”, followed by the interface bandwidth, in Gbps. Multiple network interfaces can be specified by repeating the “n” option.

Virtual interfaces may be of an access type and therefore untagged, or of a trunk type, with one or more 802.1Q tagged logical interfaces. Tagged interfaces are encapsulated by the overlay, so that tenant isolation (that is, security) is maintained, irrespective of the tag values applied by the workload.

Note: The number of virtual network interfaces, or vNICs, associated with a virtual compute instance, is directly

related to the number of vNIC extensions declared for the environment. The vNIC extension is not part of the base flavour.

<network interface bandwidth option> :: <"n"><number (bandwidth in Gbps)>

Virtual Network Interface Option	Interface Bandwidth
n1, n2, n3, n4, n5, n6	1, 2, 3, 4, 5, 6 Gbps
n10, n20, n30, n40, n50, n60	10, 20, 30, 40, 50, 60 Gbps
n25, n50, n75, n100, n125, n150	25, 50, 75, 100, 125, 150 Gbps
n50, n100, n150, n200, n250, n300	50, 100, 150, 200, 250, 300 Gbps
n100, n200, n300, n400, n500, n600	100, 200, 300, 400, 500, 600 Gbps

Table 4-17: Virtual network interface specification examples

4.2.6 Storage extensions

Persistent storage is associated with workloads via storage extensions. The storage qualities specified by the Storage Extension pertain to the "Platform Native - Hypervisor Attached" and "Platform Native - Container Persistent" storage types, as defined in "3.6.3 Storage for Tenant Consumption". The size of an extension can be specified explicitly in increments of 100 GB (Table 4-18), ranging from a minimum of 100 GB to a maximum of 16 TB. Extensions are configured with the required performance category, in accordance with Table 4-18. Multiple persistent storage extensions can be attached to virtual compute instances.

Note: This specification uses GB and GiB to refer to a Gibibyte (2^30 bytes), except where otherwise stated.

.conf	Read IO/s	Write IO/s	Read throughput (MB/s)	Write throughput (MB/s)	Max. ext. size
.bronze	Up to 3K	Up to 1.5K	Up to 180	Up to 120	16TB
.silver	Up to 60K	Up to 30K	Up to 1200	Up to 400	1TB
.gold	Up to 680K	Up to 360K	Up to 2650	Up to 1400	1TB

Table 4-18: Storage Extensions

Note: Performance is based on a block size of 256 KB or larger.

5 Feature Set and Requirements from Infrastructure

A profile profiles-profile-extensions--flavours specifies the configuration of a Cloud Infrastructure node (host or server). Profile extensions (specialisations) may specify additional configurations. Workloads use profiles to describe the configuration of nodes on which they can be hosted to execute on. Workload flavours provide a mechanism to specify the VM or Pod sizing information to host the workload. Depending on the requirements of the workloads, a VM or a Pod is deployed according to the specified flavour information on a node configured according to the specified profile. Not only do the nodes (the hardware) have to be configured but some of the capabilities also need to be configured in the software layers (such as Operating System and Virtualisation Software). Therefore, a profile can be defined in terms of

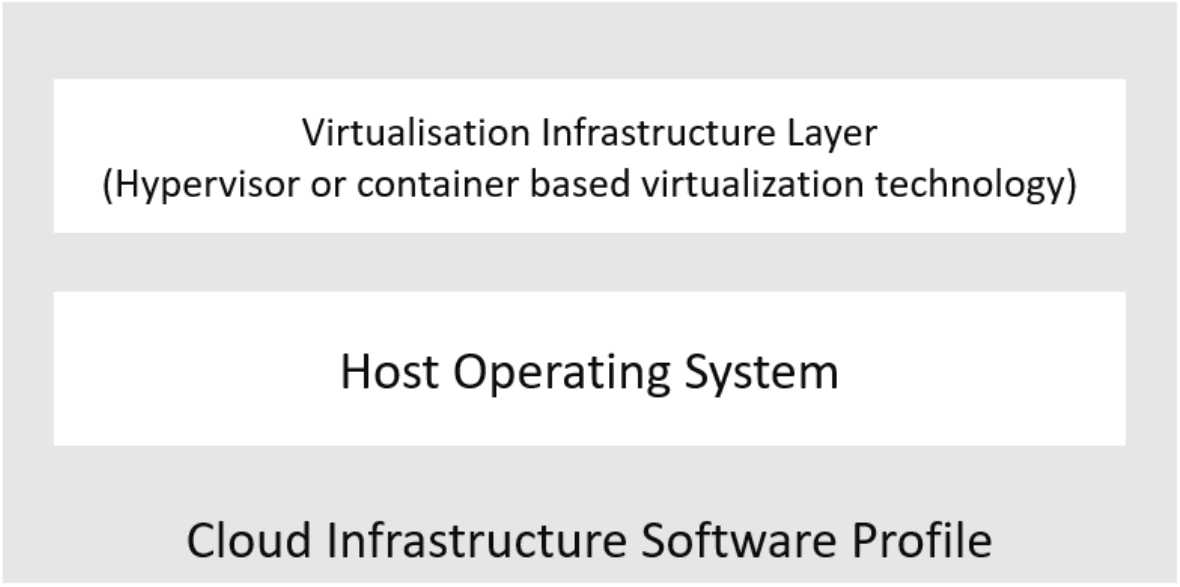
the configuration required in the software layers, the Cloud Infrastructure software profile, and the hardware, the Cloud Infrastructure Hardware Profile.

5.1 Cloud Infrastructure software profile description

The Cloud Infrastructure software layer is composed of the following two layers:

188. The virtualisation infrastructure layer. This layer is based on hypervisor virtualisation technology or container-based virtualisation technology. Container virtualisation can be nested in hypervisor-based virtualisation.

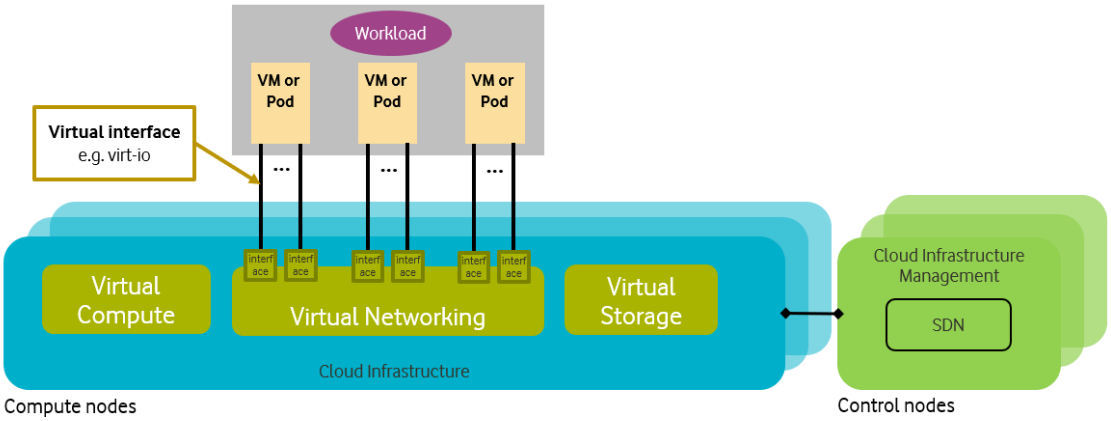
189. The host operating system (OS) layer.



Cloud Infrastructure software layers

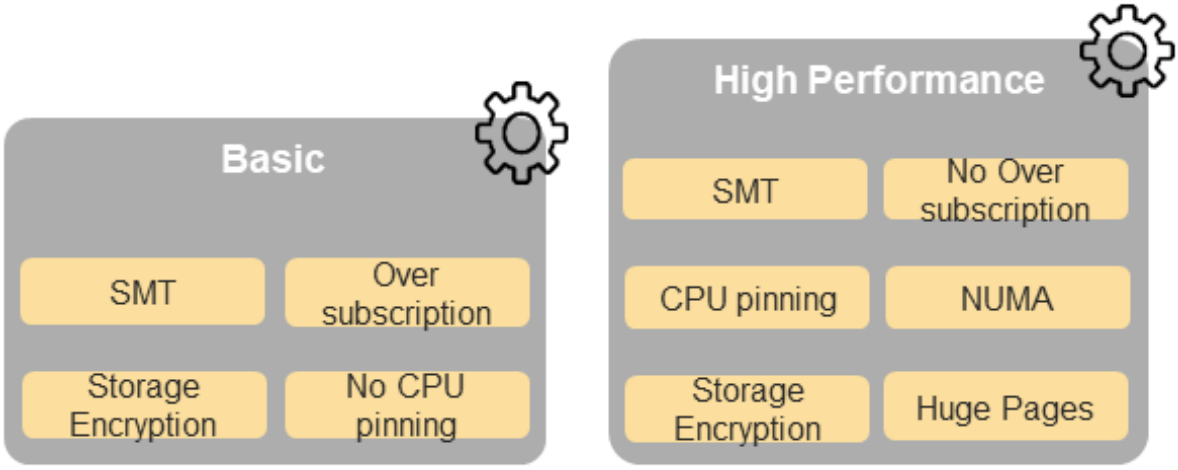
Ref	Cloud Infrastructure software	Type	Definition/notes	Capabilities reference (1)
infra.sw.001	Host operating system	<value>	Values, such as Ubuntu20.04, Windows 10 Release #, and so on.	e.cap.021
infra.sw.002	Virtualisation infrastructure layer	<value>	Values, such as KVM, Hyper-V, Kubernetes, and so on.	e.cap.022

For a host (a compute node or a physical server), the virtualisation layer is an abstraction layer between the hardware components (compute, storage, and network resources) and the virtual resources allocated to a VM or a Pod. Cloud Infrastructure Virtual resources represents the virtual resources (virtual compute, virtual network, and virtual storage) allocated to a VM or a Pod and managed by the Cloud Infrastructure manager.



Cloud Infrastructure virtual resources

A Cloud Infrastructure software profile is a set of features, capabilities, and metrics offered by a Cloud Infrastructure software layer and configured in the software layers (the Operating System (OS) and the virtualisation software (such as the hypervisor)). Cloud Infrastructure Software Profiles depicts a high-level view of the basic and high-performance Cloud Infrastructure profiles.



Cloud Infrastructure software profiles

The following sections detail the Cloud Infrastructure software profile capabilities according to the type of virtual resource.

5.1.1 Virtual compute profiles

Table 5-1 and **Table 5-2** depict the features related to virtual compute.

Reference	Feature	Type	Description	Capabilities reference
infra.com.cfg.001	CPU allocation ratio	<value>	The number of virtual cores per physical core.	i.cap.016
infra.com.cfg.002	NUMA alignment	Yes/No	Support of NUMA at the host OS and the virtualisation	e.cap.007

			layers, in addition to the hardware.	
infra.com.cfg.003	CPU pinning	Yes/No	This feature binds a vCPU to a physical core or an SMT thread. It is configured in the OS and virtualisation layers.	e.cap.006
infra.com.cfg.004	Huge pages	Yes/No	The ability to manage huge pages of memory. It is configured in the OS and virtualisation layers.	i.cap.018
infra.com.cfg.005	Simultaneous multithreading (SMT)	Yes/No/Optional	SMT allows multiple execution threads to be executed on a single physical CPU core. It is configured in the OS, in addition to the hardware.	e.cap.018

Table 5-1: Virtual compute features.

Reference	Feature	Type	Description	Capabilities reference
infra.com.acc.cfg .001	IPSec acceleration	Yes/No/Optional	IPSec acceleration	e.cap.008
infra.com.acc.cfg .002	Transcoding acceleration	Yes/No/Optional	Transcoding acceleration	e.cap.010
infra.com.acc.cfg .003	Programmable acceleration	Yes/No/Optional	Programmable acceleration	e.cap.011
infra.com.acc.cfg .004	GPU	Yes/No/Optional	Hardware coprocessor	e.cap.014
infra.com.acc.cfg .005	FPGA/other acceleration H/W	Yes/No/Optional	This is non-specific hardware. These capabilities generally require hardware-dependent drivers to be injected into the workloads.	e.cap.016

Table 5-2: Virtual compute acceleration features.

5.1.2 Virtual storage profiles

Table 5-3 and **Table 5-4** depict the features related to virtual storage.

Reference	Feature	Type	Description
infra.stg.cfg.001 infra.stg.cfg.002 infra.stg.cfg.003 infra.stg.cfg.004	Catalogue storage types Storage block Storage with replication Storage with encryption	Yes/No Yes/No Yes/No Yes/No	The support of storage types described in the catalogue

Table 5-3: Virtual storage features.

Reference	Feature	Type	Description
infra.stg.acc.cfg.001 infra.stg.acc.cfg.002	Storage IOPS oriented Storage capacity oriented	Yes/No Yes/No	

Table 5-4: Virtual storage acceleration features.

5.1.3 Virtual networking profiles

Table 5-5 and **Table 5-6** depict the features related to virtual networking.

Reference	Feature	Type	Description	Capabilities Reference
infra.net.cfg.001	Connection point interface IO virtualisation	IO virtualisation	For example, virtio1.1.	
infra.net.cfg.002	Overlay protocol	Protocols	The overlay network encapsulation protocol needs to enable ECMP in the underlay to take advantage of the scale-out features of the network fabric.	
infra.net.cfg.003	NAT	Yes/No	Support of network address translation.	
infra.net.cfg.004	Security groups	Yes/No	The set of rules managing incoming and outgoing network traffic.	
infra.net.cfg.005	Service function chaining	Yes/No	Support of service function chaining (SFC).	
infra.net.cfg.006	Traffic patterns symmetry	Yes/No	Traffic patterns should be optimal, in terms of packet flow. North-south traffic must not be concentrated in specific elements in the architecture, making those critical choke-points, unless strictly necessary (for example,	

			when NAT 1:many is required).	

Table 5-5: Virtual networking features.

Reference	Feature	Type	Description	Capabilities reference
infra.net.acc.cfg.001	vSwitch optimisation	Yes/No and SW optimisation	For example, DPDK.	e.cap.019
infra.net.acc.cfg.002 infra.net.acc.cfg.003 infra.net.acc.cfg.004	SmartNIC (for HW offload) Crypto acceleration Crypto acceleration interface	Yes/No Yes/No Yes/No	HW offload	e.cap.015 e.cap.009

Table 5-6: Virtual networking acceleration features.

5.1.4 Security details

For details, see section 7 Security.

5.1.5 Platform services

This section details the services that may be made available to the workloads by the Cloud Infrastructure.

Reference	Feature	Type	Description
infra.svc.stg.001	Object storage	Yes/No	Object storage service (for example, S3-compatible).

Table 5-7: Cloud Infrastructure platform services.

Platform service category	Platform service examples
Data stores/databases	Ceph, etcd, MongoDB, Redis
Streaming and messaging	Apache Kafka, Rabbit MQ
Load balancer and service proxy	Envoy, Istio, NGINX
Service mesh	Envoy, Istio
Security and compliance	Calico, cert-manager
Monitoring	Prometheus, Grafana (for visualisation), Kiali (for service mesh)
Logging	Fluentd, ElasticSearch (Elastic.io, Open Distro), ELK Stack (Elasticsearch, Logstash, and Kibana)
Application definition and image build	Helm
CI/CD	Argo, GitLab, Jenkins
Ingress/egress controllers	Envoy, Istio, NGINX
Network related services	CoreDNS, Istio

Coordination and service discovery	CoreDNS, etcd, Zookeeper
Automation and configuration	Ansible
Key management	Vault
Tracing	Jaeger

Table 5-7a: Service examples.

5.1.5.1 Platform services - load balancer requirements

The table below specifies a set of requirements for the load balancer platform service.

Reference	Requirement	Notes
pas.lb.001	The load balancer must support workload resource scaling.	
pas.lb.002	The load balancer must support resource resiliency.	
pas.lb.003	The load balancer must support scaling and resiliency in the local environment.	Local environment: within a subnet, tenant network, Availability Zone of a cloud, ...
pas.lb.004	The load balancer must support OSI Level 3/4 load balancing.	Specifically, OSI Level 3 load balancing decisions on the source and destination IP addresses, and OSI Level 4 TCP port numbers.
pas.lb.005	The load balancer must, at a minimum, support round-robin load balancing.	
pas.lb.006	The load balancer must create event logs with the appropriate severity levels (catastrophic, critical, and so on).	
pas.lb.007	The load balancer must support monitoring of endpoints.	
pas.lb.008	The load balancer must support Direct Server Return (DSR).	Other modes can also be supported, but DSR should always be supported.
pas.lb.009	The load balancer must support stateful TCP load balancing.	
pas.lb.010	The load balancer must support UDP load-balancing.	
pas.lb.011	The load balancer must support load balancing and the correct handling of fragmented packets.	
pas.lb.012	The load balancer may support stateful SCTP load balancing.	
pas.lb.013	The load balancer may support stateful M-TCP load balancing.	

pas.lb.014	The load balancer may support Level 7 load balancing.	Application characteristics-based OSI Level 7 should support HTTP and HTTPS.
pas.lb.015	The L7 load balancer may support HTTP2.	
pas.lb.016	The L7 load balancer may support HTTP3.	
pas.lb.017	The L7 load balancer may support QUIC.	

Table 5-7b: Platform services - load balancer requirements.

5.1.5.2 Platform services - log management service (LMS)

The table below specifies a set of requirements for the log management service (LMS).

Reference	Requirement	Notes
pas.lms.001	The LMS must support log management from multiple distributed sources.	
pas.lms.002	The LMS must manage log rotation at configurable periods.	
pas.lms.003	The LMS must manage log rotation at configurable log file status (%full).	
pas.lms.004	The LMS must manage archival and retention of logs for configurable periods by different log types.	
pas.lms.005	The LMS must ensure log file integrity (no changes, particularly changes that may affect the completeness, consistency, and accuracy, including event times, of the log file content).	This is covered by req.sec.mon.005: "The Prod-Platform and Non Prod-Platform must secure and protect all logs (containing sensitive information) both in-transit and at rest."
pas.lms.006	The LMS must monitor log rotation and log archival processes.	
pas.lms.007	The LMS must monitor the logging status of all the log sources.	
pas.lms.008	The LMS must ensure that the clock of each logging host is synchronized to a common time source.	
pas.lms.009	The LMS must support the reconfiguring of logging as needed, based on policy changes, technology changes, and other factors.	
pas.lms.010	The LMS must support the documenting and reporting of anomalies in log settings, configurations, and processes.	

pas.lms.011	The LMS must support the correlating of entries from multiple logs that relate to the same event.
pas.lms.012	The LMS must support the correlating of multiple log entries from a single source or multiple sources, based on logged values (for example, event types, timestamps, and IP addresses).
pas.lms.013	The LMS should support rule-based correlation.

Table 5-7c: Platform services - log management service (LMS) requirements

5.1.5.3 Platform services - monitoring service requirements

The table below specifies a set of requirements for the monitoring service (aka monitoring system).

Reference	Requirement	Notes
pas.mon.001	The monitoring service must be able to collect data generated by or collected from any resource (physical and virtual infrastructure, application, network, and so on).	Ability to monitor applications, services, operating systems, network protocols, system metrics, and infrastructure components.
pas.mon.002	The monitoring service must be able to aggregate the collected data.	
pas.mon.003	The monitoring service must be able to correlate data from different systems.	
pas.mon.004	The monitoring service must be able to perform active or passive monitoring, or both.	
pas.mon.005	The monitoring service must support the configuration of thresholds, outside which the resource cannot function normally, for alert generation.	
pas.mon.006	The monitoring service must support the configuration of alert notification media (such as email, SMS, phone, and so on).	
pas.mon.007	The monitoring service must support configurable realerting after a configurable period, if the metric remains outside the threshold.	
pas.mon.008	The monitoring service must support configurable alert escalations.	
pas.mon.009	The monitoring service must support alert acknowledgments by disabling the future alerting of the same resource/reason.	
pas.mon.010	The monitoring service must support the selective enabling and disabling of alerts	

	by resource, category of resources, and lengths of time.	
pas.mon.011	The monitoring service must publish its APIs for programmatic invocation of all monitoring service functions.	
pas.mon.012	The monitoring service must itself be monitored through a logging service.	
pas.mon.013	The monitoring service should be implemented for high availability to ensure the non-stop monitoring of critical infrastructure components.	
pas.mon.014	The monitoring service should run separately from production services.	
pas.mon.015	A failure of the monitoring of the system should not cause a failure in the monitoring service.	
pas.mon.016	An inoperative monitoring service should not generate alerts about the monitored system.	
pas.mon.017	The monitoring service should provide a consolidated view of the entire monitored infrastructure.	View: dashboard or report

Table 5-7d: Platform services - monitoring service requirements

5.2 Features and requirements of the Cloud Infrastructure software profiles

This section details the Cloud Infrastructure software profiles and the associated configurations for the two types of Cloud Infrastructure profiles: Basic and High Performance.

5.2.1 Virtual compute features and configuration

Table 5-8 depicts the features and configurations related to virtual compute for the two Cloud Infrastructure profiles.

Reference	Feature	Type	Basic	High performance
infra.com.cfg.001	CPU allocation ratio	<value>	N:1	1:1
infra.com.cfg.002	NUMA alignment	Yes/No	N	Y
infra.com.cfg.003	CPU pinning	Yes/No	N	Y
infra.com.cfg.004	Huge pages	Yes/No	N	Y
infra.com.cfg.005	Simultaneous multithreading (SMT)	Yes/No/Optional	Y if SMT is supported	Optional

Table 5-8: Virtual compute features and configuration for the two types of Cloud Infrastructure profiles

Table 5-9 lists the features related to compute acceleration for the high-performance profile. The table also lists the applicable Profile extensions and extra specifications that may need to be specified.

Reference	Feature	Profile-Extensions	Profile extra specs
infra.com.acc.cfg.001	IPSec acceleration	Compute-intensive GPU	
infra.com.acc.cfg.002	Transcoding acceleration	Compute-intensive GPU	Video transcoding
infra.com.acc.cfg.003 infra.com.acc.cfg.004 infra.com.acc.cfg.005	Programmable acceleration GPU FPGA/other acceleration H/W	Firmware-programmable adapter Compute-intensive GPU Firmware-programmable adapter	accelerator

Table 5-9: Virtual compute acceleration features

5.2.2 Virtual storage features and configuration

Table 5-10 and **Table 5-11** depict the features and configurations related to virtual storage for the two Cloud Infrastructure profiles.

Reference	Feature	Type	Basic	High performance
infra.stg.cfg.001	Catalogue storage types	Yes/No	Y	Y
infra.stg.cfg.002	Storage block	Yes/No	Y	Y
infra.stg.cfg.003	Storage with replication	Yes/No	N	Y
infra.stg.cfg.004	Storage with encryption	Yes/No	Y	Y

Table 5-10: Virtual storage features and configuration for the two profiles

Table 5-11 depicts the features related to virtual storage acceleration.

Reference	Feature	Type	Basic	High performance
infra.stg.acc.cfg.001	Storage IOPS oriented	Yes/No	N	Y
infra.stg.acc.cfg.002	Storage capacity oriented	Yes/No	N	N

Table 5-11: Virtual storage acceleration features.

5.2.3 Virtual networking features and configuration

Table 5-12 and **Table 5-13** depict the features and configurations related to virtual networking for the two types of Cloud Infrastructure profiles.

Reference	Feature	Type	Basic	High performance
infra.net.cfg.001	Connection point interface	IO virtualisation	virtio1.1	virtio1.1*
infra.net.cfg.002	Overlay protocol	Protocols	VXLAN, MPLSoUDP, GENEVE, other	VXLAN, MPLSoUDP, GENEVE, other
infra.net.cfg.003	NAT	Yes/No	Y	Y
infra.net.cfg.004	Security group	Yes/No	Y	Y
infra.net.cfg.005	Service function chaining	Yes/No	N	Y

infra.net.cfg.006	Traffic patterns symmetry	Yes/No	Y	Y

Table 5-12: Virtual networking features and configuration for the two types of SW profiles

Note: * might have other interfaces (such as SR-IOV VFs to be directly passed to a VM or a Pod) or NIC-specific drivers on guest machines transiently allowed until mature enough solutions are available with a similar efficiency level (for example, regarding CPU and energy consumption).

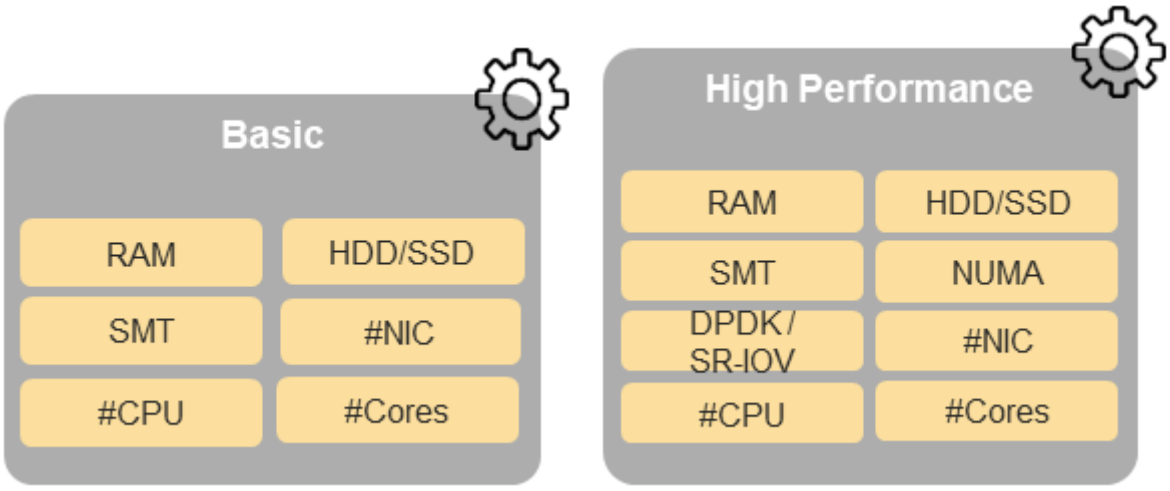
Reference	Feature	Type	Basic	High performance
infra.net.acc.cfg.001	vSwitch optimisation (DPDK)	Yes/No and SW optimisation	N	Y
infra.net.acc.cfg.002	SmartNIC (for HW Offload)	Yes/No/Optional	N	Optional
infra.net.acc.cfg.003	Crypto acceleration	Yes/No/Optional	N	Optional
infra.net.acc.cfg.004	Crypto acceleration interface	Yes/No/Optional	N	Optional

Table 5-13: Virtual networking acceleration features

5.3 Cloud Infrastructure hardware profile description

The support of various workload types, each with different, and sometimes conflicting, compute, storage, and network characteristics, including accelerations and optimizations, drives the need to aggregate these characteristics as a hardware (host) profile and capabilities. A host profile is a “personality” assigned to a compute host (also known as physical server, compute host, host, node, or pServer). The host profiles and related capabilities consist of the intrinsic compute host capabilities, such as the number of CPU sockets, the number of cores per CPU, RAM, local disks and their capacity, and so on, as well as capabilities enabled in the hardware/BIOS, specialized hardware, such as accelerators, the underlay networking, and storage.

This chapter defines a simplified host, profile, and related capabilities model associated with each of the different Cloud Infrastructure hardware profile and related capabilities. The two profiles-profile-extensions--flavours (also known as host profiles, node profiles, and hardware profiles), and some of their associated capabilities, are shown in Cloud Infrastructure Hardware Profiles and host-associated capabilities.



Cloud Infrastructure hardware profiles and host-associated capabilities

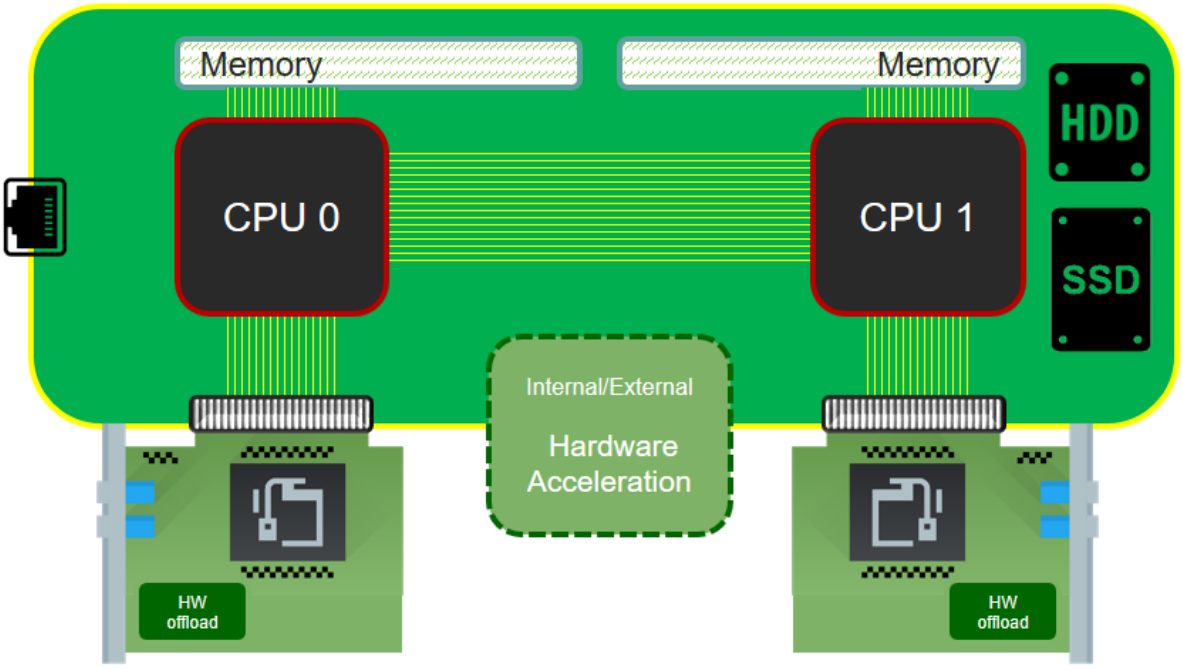
The profiles can be considered to be the set of Enhanced Performance Awareness (EPA)-related configurations on Cloud Infrastructure resources.

Note: In this chapter we will not list all of the EPA-related configuration parameters.

A given host can only be assigned a single host profile. A host profile can be assigned to multiple hosts. In addition to the host profile, Profiles and workload flavours, and additional capability specifications for the configuration of the host can be specified. Different cloud service providers (CSPs) may use different naming standards for their host profiles. For the profiles to be configured, the architecture of the underlying resource needs to be known.

Ref	Cloud Infrastructure resource	Type	Definition/Notes	Capabilities reference
infra.hw.001	CPU architecture	<value>	Values such as x64, ARM, and so on.	e.cap.020

The host profile properties are specified in the following subsections. The following diagram (Generic model of a compute host for use in Host Profile configurations) represents a high-level abstraction of a physical server (host).



Generic model of a compute host for use in host profile configurations

5.4 Features and requirements of the Cloud Infrastructure hardware profiles

The configurations specified in this section are used to specify the hardware profile configurations for each of the Cloud Infrastructure hardware profiles depicted in **Figure 5-4**.

5.4.1 Compute resources

Reference	Feature	Description	Basic	High performance
infra.hw.cpu.cfg.001	Minimum number of CPU sockets	This feature specifies the minimum number of populated CPU sockets within each host (*).	2	2
infra.hw.cpu.cfg.002	Minimum number of cores per CPU	This feature specifies the number of cores needed per CPU (*).	20	20
infra.hw.cpu.cfg.003	NUMA alignment	NUMA alignment is enabled and BIOS is configured to enable NUMA.	N	Y
infra.hw.cpu.cfg.004	Simultaneous Multithreading (SMT)	SMT is enabled, allowing each core to work multiple streams of data simultaneously.	Y if SMT is supported	Y if SMT is supported

Table 5-14: Minimum sizing and capability configurations for general purpose servers

5.4.1.1 Compute acceleration hardware specifications

Reference	Feature	Description	Basic	High performance	Capabilities reference
infra.hw.cac.cfg.001	GPU	GPU	N	Optional	e.cap.014
infra.hw.cac.cfg.002	FPGA/other acceleration HW	HW accelerators	N	Optional	e.cap.016

Table 5-15: Compute acceleration configuration specifications

5.4.2 Storage configurations

Reference	Feature	Description	Basic	High performance
infra.hw.stg.hdd.cfg.001*	Local storage HDD	Hard disk drive		

infra.hw.stg.ssd.cfg.002*	Local storage SSD	Solid state drive	Recommended	Recommended

Table 5-16: Storage configuration specification

Note: *These are specified local storage configurations including # and capacity of storage drives.

5.4.3 Network resources

5.4.3.1 NIC configurations

Reference	Feature	Description	Basic	High performance
infra.hw.nic.cfg.001	NIC ports	Total number of NIC ports available in the host.	4	4
infra.hw.nic.cfg.002	Port speed	Port speed, in Gbps (minimum values).	10	25

Table 5-17: Minimum NIC configuration specification

5.4.3.2 PCIe configurations

Reference	Feature	Description	Basic	High performance
infra.hw.pci.cfg.001 infra.hw.pci.cfg.002 infra.hw.pci.cfg.003	PCIe slots PCIe speed PCIe lanes	Number of PCIe slots available in the host.	8 Gen 3 8	8 Gen 3 8

Table 5-18: PCIe configuration specifications

5.4.3.3 Network acceleration configurations

Reference	Feature	Description	Basic	High performance	Capabilities Reference
infra.hw.nac.cfg.001	Crypto acceleration	IPSec, Crypto.	N	Optional	e.cap.009
infra.hw.nac.cfg.002	SmartNIC	Offloads network functionality.	N	Optional	e.cap.015
infra.hw.nac.cfg.003	Compression		Optional	Optional	
infra.hw.nac.cfg.004	SR-IOV over PCI-PT	SR-IOV	N	Optional	e.cap.013
infra.hw.nac.cfg.005	Time Sensitive Networking	Timing accuracy with PTP Hardware Clock and synchronization with SyncE.	N	Optional	e.cap.027

Table 5-19: Network acceleration configuration specification

6 External Interfaces

6.1 Introduction to external interfaces

In the earlier sections in this document, the various resources and capabilities of the Cloud Infrastructure were catalogued and the workloads profiled with respect to those capabilities. The intent behind this section and an “API Layer” is to similarly provide a single place to catalogue, and thereby codify, a common set of open APIs to access (that is, request, consume, control, and so on) the aforementioned resources, be they directly exposed to the workloads, or purely internal to the Cloud Infrastructure.

This section and document also aim to ensure that the APIs adopted for the Cloud Infrastructure implementations are open and not proprietary, in support of compatibility, component substitution, and the ability to realize the maximum value from existing and future test heads and harnesses.

This section aims to catalogue the APIs. It does not aim to reprint the APIs, as this would make the maintenance of this section impractical and its length disproportionate within the Reference Model document. Instead, the APIs selected for the Cloud Infrastructure implementations and specified in this section will be incorporated by reference and URLs for the latest authoritative versions of the APIs, provided in the References section of this document.

Although this document does not attempt to reprint the APIs themselves, where appropriate and where the mapping of the resources and capabilities within the Cloud Infrastructure to objects in the APIs would be otherwise ambiguous, this section will provide explicit identification and mapping.

In addition to the raw or base-level Cloud Infrastructure functionality to API and object mapping, this section aims to specify an explicit, normalized set of APIs and mappings to control the logical interconnections and relationships between these objects, notably, but not limited to, support of Service Function Chaining (SFC) and other networking and network management functionalities.

This section specifies the abstract interfaces (API, CLI, and so on) supported by the Cloud Infrastructure Reference Model. The purpose of this section is to define and catalogue a common set of open (not proprietary) APIs, of the following types:

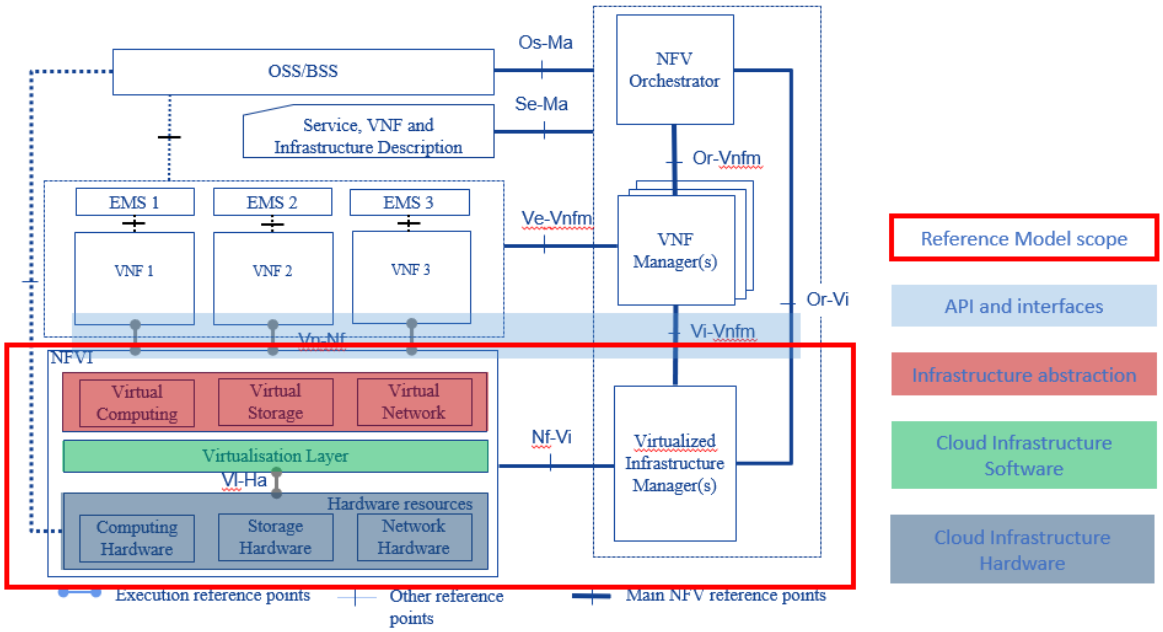
190. Cloud Infrastructure APIs: These APIs are provided to the workloads (that is, exposed), by the infrastructure, in order for workloads to access (that is, request, consume, control, and so on) Cloud Infrastructure resources.

191. Intra-Cloud Infrastructure APIs: These APIs are provided and consumed directly by the infrastructure. These APIs are purely internal to the Cloud Infrastructure and are not exposed to the workloads.

192. Enabler Services APIs: These APIs are provided by non-Cloud Infrastructure services and provide capabilities that are required for a majority of workloads, such as DHCP, DNS, NTP, DBaaS, and so on.

6.2 Cloud Infrastructure APIs

The Cloud Infrastructure APIs consist of sets of APIs that are externally and internally visible. The externally visible APIs are made available for the orchestration and management of the execution environments that host the workloads, while the internally visible APIs support actions on the hypervisor and the physical resources. The ETSI NFV Reference MANO Architecture (ETSI NFV architecture mapping) shows a number of Interface points where specific APIs, or sets of APIs, are supported. For the scope of the reference model, the relevant interface points are shown in **Table 6-1**.



ETSI NFV architecture mapping

Interface point	Cloud Infrastructure exposure	Interface between	Description
Vi-Ha	Internal NFVI	Software layer and hardware resources	1. Discover/collect the resources and their configuration information. 2. Create an execution environment (for example, VM) for the workloads (VNF).
Vn-Nf	External	NFVI and VM (VNF)	Here, VNF represents the execution environment. The interface is used to specify the interactions between the VNF and the abstract NFVI accelerators. The interfaces can be used to discover, configure, and manage these accelerators, and for the VNF to register/ deregister for receiving accelerator events and data.
NF-Vi	External	NFVI and VIM	1. Discover/collect physical/virtual resources and their configuration information. 2. Manage (create, resize, suspend, unsuspend, reboot, and so on) physical/virtualised resources. 3. Configuration changes of physical/virtual resources. 4. Physical/Virtual resource configuration.
Or-Vi	External	VNF orchestrator and VIM	See below
Vi-Vnfm	External	VNF manager and VIM	See below

Table 6-1: NFVI and VIM interfaces with other system components in the ETSI NFV architecture

The Or-Vi and Vi-Vnfm both specify interfaces provided by the VIM and are therefore related. The Or-Vi reference point is used for exchanges between the NFV orchestrator and the VIM, and supports the interfaces listed below. Virtualised resources refer to virtualised compute, storage, and network resources:

193. Software image management

194. Virtualised resources information management

195. Virtualised resources capacity management (only VNF orchestrator and VIM (Or-Vi))

196. Virtualised resources management

197. Virtualised resources change management

198. Virtualised resources reservation management

199. Virtualised resources quota management

200. Virtualised resources performance management

201. Virtualised resources fault management

202. Policy management

203. Network forwarding path (NFP) management (only VNF orchestrator and VIM (Or-Vi))

6.2.1 Tenant-level APIs

In the abstraction model of the Cloud Infrastructure (see **Chapter 3**), a conceptual model of a tenant represents the slice of a cloud zone dedicated to a workload. This slice, the tenant, is composed of virtual resources being utilized by workloads within that tenant. The tenant has an assigned quota of virtual resources. A set of users can perform operations according to their assigned roles, and the tenant exists within a cloud zone. The APIs specify the allowed operations on the tenant, including its component virtual resources. The different APIs can only be executed by users with the appropriate roles. For example, a tenant may only be allowed to be created and deleted by the cloud zone administrators, while the virtual compute resources could be allowed to be created and deleted by the tenant administrators.

The creation of a workload in a tenant also requires APIs for the management (creation, deletion, and operation) of the tenant, software flavours (see Chapter 5), operating system and workload images (“Images”), identity and authorization (“Identity”), virtual resources, security, and the workload application (“stack”).

A virtual compute resource is created according to the flavour template, specifying the compute, memory, and local storage capacity. It is launched using an image with access and security credentials. Once launched, it is referred to as a virtual compute instance or simply “Instance”. Instances can be launched by specifying the compute, memory, and local storage capacity parameters, instead of an existing flavour. Reference to flavours covers the situation where the capacity parameters are specified. IP addresses and storage volumes can be attached to a running Instance.

Resource	Create	List	Attach	Detach	Delete	Notes
Flavour	x	x			x	
Image	x	x			x	Created and deleted by the appropriate administrators.
Key pairs	x	x			x	
Privileges						Created and managed by the Cloud Service Provider (CSP) administrators.

Role	x	x			x	Created and deleted by authorized administrators where roles are assigned privileges and mapped to the users in scope.
Security groups	x	x			x	Created and deleted only by the VDC administrators.
Stack	x	x			x	Created and deleted by VDC users with the appropriate role.
Virtual storage	x	x	x	x	x	Created and deleted by VDC users with the appropriate role.
User	x	x		x	x	Created and deleted only by the VDC administrators.
Tenant	x	x		x	x	Created and deleted only by the Cloud Zone administrators.
Virtual compute	x	x		x	x	Created and deleted by VDC users with the appropriate role. Additional operations include suspend and unsuspend.
Virtual network	x	x	x	x	x	Created and deleted by VDC users with the appropriate role.

Table 6-2: API types for a minimal set of resources

Table 6-2 specifies a minimal set of operations for a minimal set of resources that are needed to orchestrate the workloads. The APIs for the listed operations are specified in the Reference Architectures. Each listed operation can have a number of associated APIs with a different set of parameters. For example, create a virtual resource using an image or a device.

6.2.2 Hardware acceleration interfaces

Acceleration Interface Specifications ETSI GS NFV-IFA 002 [12] defines a technology- and implementation-independent virtual accelerator, and the accelerator interface requirements and specifications that would allow a workload to leverage a virtual accelerator. The virtual accelerator is modelled on extensible para-virtualised devices (EDP). ETSI GS NFV-IFA 002 [12] specifies the architectural model in Chapter 4 and the abstract interfaces for management, configuration, monitoring, and data exchange in Chapter 7.

ETSI NFV-IFA 019 3.1.1 [13] has defined a set of technology-independent interfaces for acceleration resource lifecycle management. These operations allow allocation, release, and querying of acceleration resources, get and reset statistics, subscribe/unsubscribe (terminate) to fault notifications, notify (only used by NFVI), and get alarm information.

These acceleration interfaces are summarized here in Table 6.3 for your convenience only.

Table 10Hardware acceleration interfaces in the ETSI NFV architecture

Request	Response	From, To	Type	Parameter	Description
InitAccRequest	InitAccResponse	VNF → NFVI	Input	accFilter	The accelerator subsyste

					ms to initialize and retrieve their capabilities.
RegisterForAccEventRequest	RegisterForAccEventResponse	VNF → NFVI	Input	accEvent	The event in which the VNF is interested.
AccEventNotificationRequest	AccEventNotificationResponse	NFVI → VNF	Input	vnfEventHandlerId	The handler used by the VNF registering for this event.
DeRegisterForAccEventRequest	DeRegisterForAccEventResponse	VNF → NFVI	Input	accEvent	The event from which the VNF is deregistering.
ReleaseAccRequest	ReleaseAccResponse	VNF → NFVI			
ModifyAccConfigurationRequest	ModifyAccConfigurationResponse	VNF → NFVI	Input	accConfigurationData	The configuration data for the accelerator.
GetAccConfigsRequest	GetAccConfigsResponse	VNF →	Input	accFilter	The filter for the subsys

		NFVI			tems from which the configuration data is requested.
ResetAccConfigsRequest	ResetAccConfigsResponse	VNF → NFVI	Input	accFilter	The filter for the subsystems for which the configuration is to be reset.
AccDataRequest	AccDataResponse	VNF → NFVI	Input	accData	The data (metadata) sent to the accelerator.
AccSendDataRequest	AccSendDataResponse	VNF → NFVI	Input	accData	The data (metadata) sent to the accelerator.
AccReceiveDataRequest	AccReceiveDataResponse	VNF → NFVI	Input	maxNumberOfDataItems	The maximum number of data items

					to be received.
RegisterForAccDataAvailableEventRequest	RegisterForAccDataAvailableEventResponse	VNF → NFVI	Input	regHandlerId	Registration identifier.
AccDataAvailableEventNotificationRequest	AccDataAvailableEventNotificationResponse	NFVI → VNF	Input	regHandlerId	Reference used by the VNF when registering for the event.
DeRegisterForAccDataAvailableEventRequest	DeRegisterForAccDataAvailableEventResponse	VNF → NFVI	Input	accChannel	Channel related to the event.
AllocateAccResourceRequest	AllocateAccResourceResponse	VIM → NFVI	Input	attachTargetInfo	The resource to which the accelerator is to be attached (for example, VM).
ReleaseAccResourceRequest	ReleaseAccResourceResponse	VIM → NFVI	Input	accResourceId	ID of the resource to be released.
QueryAccResourceRequest	QueryAccResourceResponse	VIM → NFVI	Input	hostId	ID of the specified host.

GetAccStatisticsRequest	GetAccStatisticsResponse	VIM → NFVI	Input	accFilter	Accelerator subsystems from which data is requested.
ResetAccStatisticsRequest	ResetAccStatisticsResponse	VIM → NFVI	Input	accFilter	Accelerator subsystems for which the data is to be reset.
SubscribeRequest	SubscribeResponse	VIM → NFVI	Input	hostId	ID of the specified host.
UnsubscribeRequest	UnsubscribeResponse	VIM → NFVI	Input	hostId	ID of the specified host.
Notify		NFVI → VIM			NFVI notifies an alarm to VIM.
GetAlarmInfoRequest	GetAlarmInfoResponse	VIM → NFVI	Input	hostId	ID of the specified host.
AccResourcesDiscoveryRequest	AccResourcesDiscoveryResponse	VIM → NFVI	Input	hostId	ID of the specified host.

OnloadAccImageRequest	OnloadAccImageResponse	VIM → NFVI	Input	accResourceId	Identifier of the chosen accelerator in the NFVI.

Table 6-3: Hardware acceleration interfaces in the ETSI NFV architecture

6.3 Intra-Cloud Infrastructure interfaces

6.3.1 Hypervisor hardware interface

Table 6-1 lists a number of NFVI and VIM interfaces, including the internal VI-Ha interface. The VI-Ha interface allows the hypervisor to control the physical infrastructure. The hypervisor acts under VIM control. The VIM issues all requests and responses using the NF-VI interface. Requests and responses include commands, configuration requests, policies, updates, alerts, and response to infrastructure results. The hypervisor also provides information about the health of the physical infrastructure resources to the VM. All these activities, on behalf of the VIM, are performed by the hypervisor using the VI-Ha interface. While no abstract APIs have yet been defined for this internal VI-Ha interface, ETSI GS NFV-INF 004 [14] defines a set of requirements and details of the information that is required by the VIM from the physical infrastructure resources. Hypervisors utilize various programs to get this data, including BIOS, IPMI, PCI, I/O Adapters/Drivers, and so on.

6.4 Enabler services interfaces

In order to function properly, an operational cloud needs a set of standard services. These services comprise NTP for time synchronization, DHCP for IP address allocation, DNS for obtaining IP addresses for domain names, and LBaaS (version 2) to distribute incoming requests amongst a pool of designated resources.

7 Security

7.1 Introduction to security

Security vulnerabilities and attack vectors are everywhere. The telecom industry and its cloud infrastructures are especially vulnerable to potential attacks due to the ubiquitous nature of the infrastructures and services, combined with the vital role telecommunications play in the modern world. The attack vectors are many and varied, ranging from the potential for exposure of sensitive data, both personal and corporate, to weaponized disruption to the global telecommunications networks. These threats can take the form of physical attacks on the locations where the infrastructure hardware is housed, to network attacks, such as denial of service and targeted corruption of the network service applications themselves. Whatever the source, a Cloud Infrastructure needs to be able to withstand attacks in whatever form they take.

This chapter examines multiple aspects of security, as it relates to cloud infrastructure and security aspects for workloads. After discussing security attack vectors, this chapter delves into the security requirements. Regarding security requirements and best practices,

specifications and documents are published by standards organisations. A selection of standards of interest for cloud infrastructure security is listed in a dedicated section. This chapter culminates with a consolidated set of requirements recommendations. It is suggested that operators carefully evaluate the recommendations for possible implementation.

7.2 Potential attack vectors

Previously, attacks designed to place and migrate workloads outside the legal boundaries were not possible using the traditional infrastructure, due to the closed nature of these systems. However, using cloud infrastructure, violation of regulatory policies and laws becomes possible by actors diverting or moving an application from an authenticated and legal location to a potentially illegal location. The consequences of violating regulatory policies may take the form of a complete banning of service and/or an exaction of a financial penalty by a governmental agency or through SLA enforcement. Such vectors of attack may be the original intention of the attacker in an effort to harm the service provider. One possible attack scenario is when an attacker exploits the insecure NF API to dump personal data records from the database, in an attempt to violate user privacy. Cloud infrastructure operators should ensure that the APIs are secure and accessible over a secure network (transport layer security or TLS), under strict set of security best practices and RBAC policies to limit exposure of this vulnerability.

Typical cloud-associated attacker tactics have been identified in the widely accepted MITRE ATT&CK® Framework [78]. This framework provides a systematic approach to capture adversarial tactics targeting cloud environments. Examples of such adversarial tactics are listed in the following table.

Attacker tactics	Examples
Initial access	Compromising user administration accounts that are not protected by multifactor authentication.
Evasion	Modifying cloud compute instances in the production environment by modifying virtual instances for the staging of attacks.
Discovery	Discovering which cloud services are operating and then disabling them at a later stage.
Data exfiltration	Moving data from the compromised tenant’s production databases to the hacker’s cloud service account, or transferring data out of the communication service provider (CSP) to the attacker's private network.
Service impact	Creating denial-of-service availability issues by modifying the Web Application Firewall (WAF) rules and compromising APIs and web-based GUIs.

Table 7-1: Examples of cloud attacker tactics

7.3 Security scope

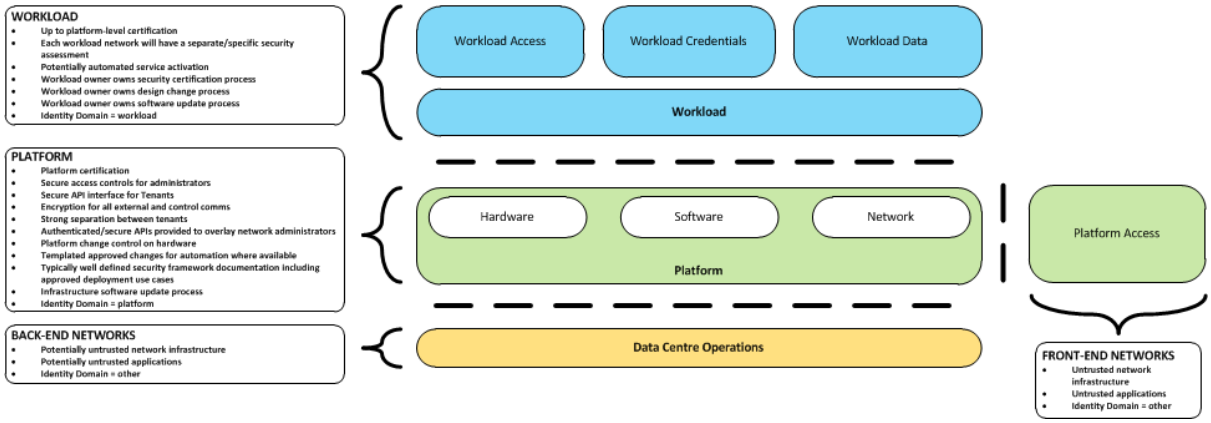
7.3.1 In-scope and out-of-scope definition

The scope of the requirements of the security controls maps to the scope of the Reference Model architecture.

Cloud infrastructure requirements must cover the virtual infrastructure layer and the hardware infrastructure layer, including virtual resources, hardware resources, the virtual infrastructure manager, and the hardware infrastructure manager, as described in Chapter 3.

7.3.2 High-level security requirements

The following diagram shows the different security domains that impact the Reference Model:



Reference Model Security Domains

..note:

"Platform" refers to the cloud infrastructure, with all its hardware and software components.

7.3.2.1 Platform security requirements

At a high level, the following areas/requirements cover platform security for a particular deployment:

204. Platform certification.
205. Secure access controls for administrators.
206. A secure API interface for tenants.
207. Encryption for all external and control communications.
208. Strong separation between tenants, ensuring network, data, memory, and runtime process (CPU running core) isolation between tenants.
209. Authenticated/secure APIs provided to overlay network administrators.
210. Platform change control on hardware.
211. Templated approved changes for automation where available.
212. Typically well-defined security framework documentation, including approved deployment use cases.
213. Infrastructure software update processes.

7.3.2.2 Workload security requirements

At a high level, the following areas/requirements cover workload security for a particular deployment:

214. Certification up to platform level.
215. Requirement of each workload network to undertake it own security self-assessment and accreditation, and not inherit a security accreditation from the platform.
216. Potentially automated service activation.
217. The workload owner owns the workload security certification process.
218. The workload owner owns the workload design change process.

219. The workload owner owns the workload software update process.

7.3.3 Common security standards

The Cloud Infrastructure Reference Model and the supporting architectures are not only required to optimally support networking functions. They must also be designed, from inception, with common security principles and standards. These best practices must be applied at all layers of the infrastructure stack and across all points of interconnection (internal or with outside networks), APIs, and contact points with the NFV network functions overlaying or interacting with that infrastructure.

A good place to start to understand the security requirements is to use the following widely accepted definitions and core principles developed by the Open Web Application Security Project (OWASP):

220. Confidentiality: only allow access to data which the user is permitted to view.

221. Integrity: ensure the data has not been tampered with or altered by unauthorised users.

222. Availability: ensure the systems and data are available to authorised users when they need it.

These three principles are complemented for cloud infrastructure security by authenticity. Authenticity refers to the ability to confirm that the users are valid and have the correct rights to access the systems or data.

Standards organisations with recommendations and best practices, and certifications that need to be taken into consideration, include the following examples. This is not an exhaustive list, it contains only some of the more important standards in current use.

223. Center for Internet Security - [79]

224. Cloud Security Alliance - [80]

225. Open Web Application Security Project - [81]

226. The National Institute of Standards and Technology (NIST), with the following special publications:

    1. NIST SP 800-123 Guide to General Server Security [82]
    2. NIST SP 800-204A Building Secure Microservices-based Applications Using Service-Mesh Architecture [83]
    3. NIST SP 800-204B Attribute-based Access Control for Microservices-based Applications Using a Service Mesh [84]
    4. NIST SP 800-207 Zero Trust Architecture [85]
    5. NIST SP 800-218 Secure Software Development Framework (SSDF) [86]

227. FedRAMP Certification [87]

228. ETSI Cyber Security Technical Committee (TC CYBER) - [88]

229. ETSI Industry Specification Group Network Functions Virtualisation (ISG NFV) [89] and its Security Working Group NFV-SEC

230. The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) - [90]. The following ISO standards are of particular interest for NFVI:

    1. ISO/IEC 27002:2013 and ISO/IEC 27001: these are the international standards for best-practice information security management systems (ISMSs).

2. ISO/IEC 27032: this is the international standard focusing explicitly on cybersecurity.
3. ISO/IEC 27035: this is the international standard for incident management.
4. ISO/IEC 27031: this is the international standard for ICT readiness for business continuity.

In the mobile network field, the GSM Association (GSMA [91]) and its Fraud and Security working group of experts have developed a set of documents specifying how to secure the global mobile ecosystem.

231. The document “Baseline Security controls”, FS.31 v2.0 [25], published in February 2020, is a practical guide intended for operators and stakeholders to check mobile network’s internal security. It lists a set of security controls from business controls (including security roles, organizational policies, business continuity management, and so on) to technological controls (for user equipment, networks, operations, and so on), covering all areas of mobile network, including Cloud Infrastructure. A checklist of questions allows the operator to improve the security of a deployed network.

232. The document "Network Equipment Security Assurance Scheme – Development and Lifecycle Security Requirements" [46], is part of a set of documents that aim to build a security assurance scheme for network equipment. Focusing on critical controls, it defines a set of requirements to be met by the vendors' development and product lifecycle processes.

The GSMA security activities are currently focused on 5G services and the new challenges posed by the virtualisation of network functions and by open-source software. The following two documents are in the scope of cloud infrastructure security:

233. The white paper “Open Networking & the Security of Open Source Software deployment” [26], deals with open-source software security. It highlights the importance of layered security defences and lists recommendations and security concepts that are able to secure deployments.

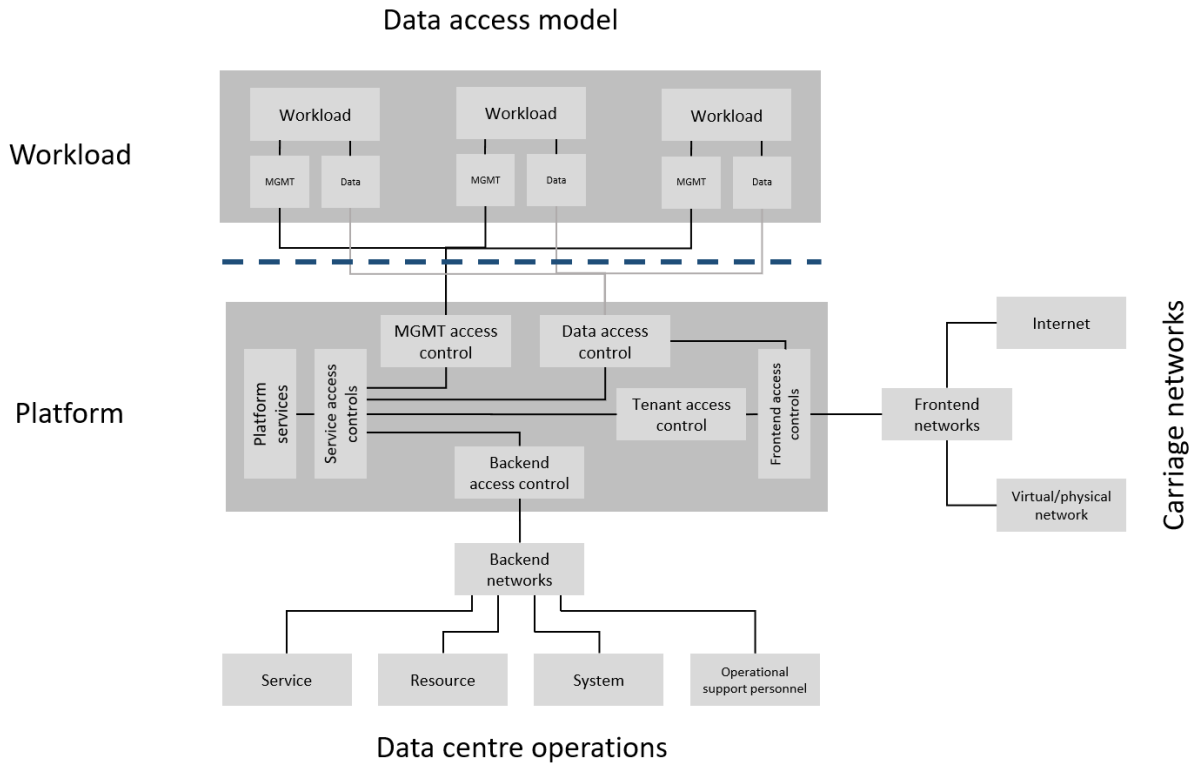
234. The “5G Security Guide” [37] (non-binding Permanent Reference Document), covers 5G security in a holistic way, from user equipment to networks. This document describes the new security features in 5G. It includes a dedicated section on the impact of the cloud on 5G security, with recommendations on virtualisation, cloud-native applications, and containerisation security.

7.4 Cloud infrastructure security

7.4.1 General platform security

The security certification of the platform typically needs to be the same as, or higher than, the workload requirements.

The platform supports the workload, and in effect controls access to the workload to and from the external endpoints, such as carriage networks used by workloads, by data centre operations staff supporting the workload, or by tenants accessing workloads. From an access security perspective, the following diagram shows where the different access controls operate within the platform to provide access controls throughout the platform.



Reference Model access controls

7.4.1.1 High-level functions of access controls

235. **MGMT ACCESS CONTROLS** - These provide platform access to workloads for service management. Typically, all management and control-plane traffic is encrypted.
236. **DATA ACCESS CONTROLS** - These control the east-west traffic between workloads and control of north-south traffic between the network function (NF) and other platform services, such as front-end carriage networks and platform services. Inherently strong separation between tenants is mandatory.
237. **SERVICES ACCESS CONTROLS** - These protect the platform services from any platform access.
238. **BACK-END ACCESS CONTROLS** - These provide data centre operations access to the platform, and subsequently, the workloads. Typically, stronger authent,ication, such as Two-Factor Authentication (2FA), is required, as well as the use of technologies, such as Role-Based Access Control (RBAC) and encryption. Application Programming Interface (API) gateways may be required for automated/script-driven processes.
239. **FRONT-END ACCESS CONTROLS** - These protect the platform from malicious carriage network access and provide connectivity for specific workloads to specific carriage networks. Carriage networks are those networks that are provided as public networks and are operated by carriers, in this case with interfaces that are usually subnetworks or virtual networks.
240. **TENANT ACCESS CONTROLS** - These provide appropriate tenant access controls to specific platform services and tenant workloads. These controls include Role-Based Access Control (RBAC), authentication controls as appropriate for the access arrangement, and Application Programming Interface (API) gateways for automated/script-driven processes.

7.4.1.2 General security requirements for the cloud infrastructure

System hardening

241. Adhering to the principle of least privilege, there is no login to the root on any platform systems when root privileges are not required.

..note:

Platform systems are those systems that are associated with the platform. They include systems that directly or indirectly affect the viability of the platform.

242. Ensure that all the platform's components, including hypervisors, VMs, and so on, are kept up to date with the latest patch.

243. To tightly control access to resources and protect them from malicious access and introspection, Linux Security Modules, such as SELinux, should be used to enforce access rules.

Vulnerability management

244. Security defects must be reported.

245. The Cloud Infrastructure components must be continuously analysed from deployment to runtime. The cloud infrastructure must offer tools to check the code libraries and all other code against the Common Vulnerabilities and Exposures (CVE) databases [92] to identify the presence of any known vulnerabilities. The CVE is a list of publicly disclosed vulnerabilities and exposures that is maintained by MITRE [93]. Each vulnerability is characterised by an identifier, a description, a date, and comments.

246. When a vulnerability is discovered on a component (from operating systems to virtualisation layer components), the remediation action depends on its severity. The Common Vulnerability Scoring System (CVSS) [94] allows you to calculate a vulnerability score. It is an open framework widely used in vulnerability management tools. The CVSS is owned and managed by the Forum of Incident Response and Security Teams (FIRST). The CVSS consists of three metric groups: base, temporal, and environmental. The base metrics produce a score ranging from 0 to 10. This score can then be refined using temporal and environmental metrics. The numerical score can be translated into a qualitative representation of the severity: low, medium, high, or critical. The severity score (or the associated qualitative representation) allows organisations to prioritise the remediation activities, with high scores mandating a fast response time. The vulnerable components must then be patched or replaced, or their access must be restricted.

247. Security patches must be obtained from an authorised source, to ensure their integrity. Patches must be tested and validated in a preproduction environment before being deployed into production.

Platform access

248. Restrict traffic only to traffic that is necessary, and deny all other traffic, including traffic to and from the backend.

249. Provide protections between the internet and the workloads, including web and volumetrics attack preventions.

250. All host-to-host communications within the cloud provider network are to be cryptographically protected in transit.

251. Use cryptographically protected protocols for administrative access to the platform.

252. Data centre operations staff and systems must use management protocols that limit security risk, such as SNMPv3, SSH v2, ICMP, NTP, syslog, and TLS v1.2 or higher.

253. Processes for managing platform access control filters must be documented, followed, and monitored.

254. Role-Based Access Control (RBAC) must apply to all access to platform systems.

255. All access to APIs, including backend APIs, must use the TLS protocol.

Security hardware assist for data in use

256. Server hardware architectures offer various technologies to assist with protecting data in use. The following table categorizes such technologies.

HW technology	Which security threat it mitigates	Where it must be enabled	How to operationally activate	How to assure benign workloads are run
Memory encryption on the level of the whole physical server.	Protects data going between the CPU and the memory DIMMs.	Server HW and BIOS.	Configure BIOS. On virtualised software infrastructure label nodes to influence scheduling.	Performed by application scheduling using node labels.
Memory encryption on the level of the VMs.	HW-protected data between the VMs.	Server HW, BIOS, and hypervisor and guests (paravirtualised generally only in the guest BIOS). Attestation if assurance of workload is required.	Configure BIOS and hypervisor. On virtualised software infrastructure label nodes to influence scheduling.	Remote attestation of the freshly spun up VM, to provide measurements of the VM and of the platform patch level.
Secure enclaves within application.	HW-protected specific application code and data in memory, from processes running at higher privilege levels, such as OS or hypervisor.	Server HW and BIOS, hypervisor if used, device plugin if Kubernetes is used, and in application. Attestation if assurance of workload required.	Configure the BIOS and the hypervisor, if used. If Kubernetes is used, then the pod descriptor requests for such resources.	Remote attestation of freshly spun up enclave, to provide measurements of the enclave and of the platform patch level.

257. Using computing accelerators, such as FPGA or GPU, that are connected via an I/O link, such as PCI Express, breaks the confidentiality property, unless HW-assisted encryption of the I/O transfers, runtime encryption of the accelerated workload, and attestation of the accelerated workload can be guaranteed.

Workload security

258. Restrict traffic to and from the workload only to traffic that is necessary, and deny all other traffic.

259. Support zoning within a tenant workload using application-level filtering.

260. Do not expose tenant-internal IP address details to another tenant.

261. All production workloads must be separated from all non-production workloads, including separation between non-hosted non-production external networks.

Confidentiality and integrity

262. All data persisted to primary, replica, or backup storage is to be encrypted.

Monitoring and security audit

263. All platform security logs are to be time-synchronised.

264. Logs are to be regularly scanned for events of interest.

265. The cloud services must be regularly tested for vulnerability and penetration.

Platform provisioning and LCM

266. A platform change management process that is documented, properly communicated to staff and tenants, and rigorously followed.

267. A process to check change management adherence that is implemented and rigorously followed.

268. An approved system or process for last-resort access must exist for the platform.

269. Where there are multiple hosting facilities used in the provisioning of a service, network communications between the facilities for the purpose of backup, management, and workload communications are cryptographically protected in transit between data centre facilities.

270. Continuous cloud security compliance is mandatory.

271. An incident response plan must exist for the platform.

7.4.2 Platform backend access security

272. Validate and verify the integrity of the resources management requests coming from a higher orchestration layer to the cloud infrastructure manager.

7.4.3 Platform frontend access security

273. Frontend network security at the application level is the responsibility of the workload. However, the platform must ensure the isolation and integrity of tenant connectivity to frontend networks.

274. The frontend network may provide Distributed Denial Of Service (DDoS) support.

7.4.4 Infrastructure as a Code security

Infrastructure as a Code (IaaC), alternatively known as Infrastructure as Code (IaC), refers to the software used for the declarative management of the cloud infrastructure resources. To dynamically address user requirements, release features incrementally, and deliver more speedily, DevSecOps teams use best practices, including continuous integration and delivery, and integrate information security controls and scanning tools into these processes, with with the aim of providing timely and meaningful feedback. This includes identifying vulnerabilities and security policy violations. With these automated security testing and analysis capabilities, it is of critical value in detecting vulnerabilities early and maintaining a consistent security policy.

Because of the high complexity of modern Telco cloud infrastructures, even minor IaaC code changes could disproportionately and sometimes disastrously affect downstream security and privacy. Therefore, the integration of security testing into the IaaC software development pipeline requires security activities to be automated using security tools, and integrated with the native DevOps and DevSecOps tools and procedures.

The DevSecOps Automation best practices advocate, among other things, the implementation of a framework for security automation and programmatic execution, as well as the monitoring of security controls to identify, protect, detect, respond, and recover from cyber threats. The framework used for IaaC security is based on the joint publication of the Cloud Security Alliance (CSA) and SAFECode, "The Six Pillars of DevSecOps: Automation (2020)" [27]. This document uses the base definitions and constructs from ISO 27000 [28], and the CSA's Information Security Management through Reflexive Security [29].

The framework identifies the following five distinct stages:

- Secure design and architecture.
- Secure coding (developer IDE and code repository).
- Continuous build, integration, and testing.
- Continuous delivery and deployment.
- Continuous monitoring and runtime defence.

Triggers and checkpoints define transitions within stages. When designing DevSecOps security processes, it should be borne in mind that when a trigger condition is met, one or more security activities is activated. The outcomes of these security activities need to determine whether or not the requirements of the process checkpoint are satisfied. If the outcome of the security activities meets the requirements, the next set of security activities is performed as the process transition to the next checkpoint, or, alternatively, to the next stage, if the checkpoint is the last one in the current stage. If, however, the outcome of the security activities does not meet the requirements, then the process should not be allowed to advance to the next checkpoint. In Consolidated security requirements, the IaaC security activities are presented as security requirements mapped to particular stages and trigger points.

7.4.5 Security of production and non-production environments

Telecommunications operators often focus their security efforts on the production environments actively used by their customers or their employees, or both. This is critical because a breach of these systems can seriously damage the company and its customers. In addition, production systems often contain the most valuable data, making them attractive targets for intruders. However, an insecure non-production (development and testing) environment can also create real problems because they may leave a company open to corporate espionage, sabotage by competitors, and theft of sensitive data.

Security is about mitigating risk. If operators do not have the same level of security in their non-production environments compared to production, then an additional level of risk may be introduced. This is especially true if such non-production environments accept outside connections (for example, for suppliers or partners, which is quite normal in complex Telco ecosystems). There is a need to monitor the security of these non-production environments. The gold standard, therefore, is to implement the same security policies in production and non-production infrastructure. This would reduce risk and typically simplify operations by using the same control tools and processes. However, for practical reasons, some of the security monitoring rules may differ. As an example, if a company maintains a separate, isolated environment for infrastructure software development experimentation, the configuration monitoring rules may be relaxed, in comparison with the production environment, where such experimentation is not allowed. Therefore, in this document, when dealing with such dilemmas, the focus has been placed on those non-production security requirements that must be on the same level as in the production environment (typically of **must** type), leaving relaxed requirements (typically of **should** or **may**) in cases where there is no such necessity.

In the context of contemporary telecommunications technology, the cloud infrastructure typically is considered Infrastructure as a Code (IaaC). This fact implies that many aspects of code-related security automatically apply to IaaC. Security aspects of IaaC in the Telco context

are discussed in "Infrastructure as a Code security", which introduces the relevant framework for security automation and programmatic execution, as well as monitoring of security controls. Organisations need to identify which of the stages, or activities within these stages, should be performed within the non-production versus production environments. This mapping will then dictate which security activities defined for particular stages and triggers (such as vulnerability tests, patch testing, and penetration tests) are mandatory, and which can be regarded as discretionary.

7.5 Workload security and vendor responsibility

7.5.1 Software hardening

275. There must be no hard-coded credentials or clear text passwords in the code and images. The software must support configurable or industry-standard password complexity rules.
276. The software should be independent of the infrastructure platform (no OS point release dependencies to patch).
277. The software must be code-signed. All individual subcomponents are assessed and verified for End-user License Agreement (EULA) violations.
278. The software should have a process for discovery, classification, communication, and timely resolution of security vulnerabilities (that is, bug bounty, penetration testing/scan findings, and so on).
279. The software should support recognised encryption standards. Encryption should be decoupled from the software.
280. The software should have support for configurable banners to display authorised use criteria/policies.

7.5.2 Port protection

281. Unused software and unused network ports should be disabled by default.

7.5.3 Software code quality and security

282. Vendors should use industry-recognized software testing suites.
    1. Static and dynamic scanning.
    2. Automated static code review with remediation of medium/high/critical security issues. The tool used for static code analysis, and analysis of the code being released, must be shared.
    3. Dynamic security tests with remediation of medium/high/critical security issues. The tool used for dynamic security analysis of code being released must be shared.
    4. Penetration tests (pen tests) with remediation of medium/high/critical security issues.
    5. The methodology for ensuring that security is included in the Agile/DevOps delivery lifecycle for ongoing feature enhancement/maintenance.

7.5.4 Alerting and monitoring

283. Security event logging: all security events must be logged, including informational events.
284. Privilege escalation must be detected.

7.5.5 Logging

285. Logging output should support customizable log retention and log rotation.

7.6 Workload security: cloud infrastructure operator responsibility

The operator's responsibility is not only to make sure that security is included in all the vendor-supplied infrastructure and NFV components, but also to maintain the security functions from an operational and management perspective. This includes, but is not limited to, securing the following elements:

286. Maintaining standard security operational management methods and processes.

287. Monitoring and reporting functions.

288. Processes to address regulatory compliance failures.

289. Support for appropriate incident response and reporting.

290. Methods to support appropriate remote attestation certification of the validity of the security components, architectures, and methodologies used. This can include the automated TLS certificate lifecycle management for workloads:

    1. Accept signing requests for certificates,
    2. Generate and manage private keys and Certificate Signing Requests (CSRs),
    3. Manage renewal before expiry.

7.6.1 Remote attestation/OpenCIT

Cloud infrastructure operators must ensure that remote attestation methods are used to remotely verify the trust status of a given cloud infrastructure platform. The concept is based on boot integrity measurements leveraging the Trusted Platform Module (TPM) built into the underlying hardware. Remote attestation can be provided as a service. It may be used by the platform owner or by a customer to verify that the platform has booted in a trusted manner. Practical implementations of the remote attestation service include the Open Cloud Integrity Tool (OpenCIT). OpenCIT provides 'Trust' visibility of the cloud infrastructure. It enables compliance in cloud datacentres by establishing the root of trust, and builds a chain of trust across hardware, operating system, hypervisor, virtual machines, and containers. It includes asset tagging for location and boundary control. The platform trust and asset tag attestation information is used by orchestrators and/or policy compliance management to ensure workloads are launched on trusted and location-or boundary-compliant platforms. They provide the necessary visibility and auditability of infrastructure in both public and private cloud environments.

7.6.2 Workload image

Only workload images from trusted sources may be used. Secrets must be stored outside the images.

It is easy to tamper with workload images. It requires only a few seconds to insert some malware into a workload image file while it is being uploaded to an image database or transferred from an image database to a compute node. To guard against this danger, workload images must be cryptographically signed and verified during launch time. This can be achieved by setting up a signing authority and modifying the hypervisor configuration to verify an image's signature before it is launched.

To implement image security, the workload operator must test the image and the supplementary components, to verify that everything conforms to the security policies and

best practices. The use of image scanners, such as OpenSCAP or Trivy, to determine security vulnerabilities is recommended.

CIS hardened images should be used whenever possible. CIS provides, for example, virtual machine hardened images based on CIS benchmarks for various operating systems. Another best practice is to use minimalist base images whenever possible.

Images are stored in registries. The images registry must contain vetted images only. The registry must remain a source of trust for images over time. Images must therefore be continuously scanned to identify vulnerabilities and out-of-date versions, as described previously. Access to the registry is an important security risk. It must be granted by a dedicated authorisation process and through secure networks that enforce authentication, integrity, and confidentiality.

7.6.3 Networking security zoning

Network segmentation is important for ensuring that applications can only communicate with those applications with which they are supposed to communicate. To prevent a workload from impacting other workloads or hosts, it is good practice to separate workload traffic and management traffic. This prevents attacks by VMs or containers breaking into the management infrastructure. It is also best to separate the VLAN traffic into appropriate groups and disable all other VLANs that are not in use. Likewise, workloads of similar functionalities can be grouped into specific zones and their traffic isolated. Each zone can be protected using access control policies and a dedicated firewall based on the required security level.

It is recommended to set network security policies following the principle of least privileged, only allowing approved protocol flows. For example, set 'default deny' for inbound flows and add the approved policies required for the functioning of the application running on the NFV infrastructure.

7.6.4 Volume encryption

Virtual volume disks associated with workloads may contain sensitive data. Therefore, they need to be protected. It is best practice is to secure the workload volumes by encrypting them and storing the cryptographic keys in safe locations. Encryption functions rely on a cloud infrastructure internal key management service. Be aware that the decision to encrypt the volumes might cause reduced performance. Therefore, the decision to encrypt needs to be dependent on the requirements of the given infrastructure. The Trusted Platform Module (TPM) module can also be used to store these keys securely. The hypervisor should also be configured to securely erase the virtual volume disks, in the event of application crashes, or in case it is intentionally destroyed to protect it from unauthorized access.

For sensitive data encryption, when data sovereignty is required, an external Hardware Security Module (HSM) should be integrated, in order to protect the cryptographic keys. A HSM is a physical device which manages and stores secrets. Usage of an HSM strengthens the security of the secrets. For 5G services, GSMA FASG recommends the implementation of an HSM to secure the storage of the Universal Integrated Circuit Card (UICC) credentials.

7.6.5 Root of trust for measurements

The following sections define the mechanisms to ensure the integrity of the infrastructure pre-boot and post-boot (running). The following definitions detail a set of terms used in those sections.

291. The hardware root of trust helps with the pre-boot and post-boot security issues.

292. The Unified Extensible Firmware Interface (UEFI) adheres to standards defined by an industry consortium. Vendors (hardware and software) and solution providers collaborate to define common interfaces, protocols, and structures for computing platforms.

293. The Platform Configuration Register (PCR) is a memory location in the TPM that is used to store TPM Measurements (hash values generated by the SHA-1 standard hashing algorithm). The PCRs are cleared only on TPM reset. UEFI defines 24 PCRs, of which the first 16, PCR 0 - PCR 15, are used to store measures created during the UEFI boot process.
294. The Root of Trust for Measurement (RTM) is a computing engine that is capable of making integrity measurements.
295. The Core Root of Trust for Measurements (CRTM) is a set of instructions executed when performing an RTM.
296. Platform Attestation provides proof of validity of the platform's integrity measurements. For details, see "Remote attestation/OpenCIT".

Values stored in a PCR cannot be reset (or forged), as they can only be extended. Whenever a measurement is sent to a TPM, the hash of the concatenation of the current value of the PCR and the new measurement is stored in the PCR. The PCR values are used to encrypt data. If the proper environment is not loaded, which will result in different PCR values, the TPM will be unable to decrypt the data.

7.6.5.1 Static Root of Trust for Measurement

The Static Root of Trust for Measurement (SRTM) begins with measuring and verifying the integrity of the BIOS firmware. It then measures additional firmware modules, verifies their integrity, and adds each component's measure to an SRTM value. The final value represents the expected state of the boot path loads. The SRTM stores the results as one or more values stored in PCR storage. In the SRTM, the CRTM resets PCRs 0 to 15 only at boot.

Using a Trusted Platform Module (TPM) as a hardware root of trust, measurements of platform components, such as firmware, bootloader, and OS kernel, can be securely stored and verified. Cloud infrastructure operators should ensure that the TPM support is enabled in the platform firmware, so that the platform measurements are correctly recorded during boot time.

A simple process works in the following way:

- The BIOS CRTM (Bios Boot Block) is executed by the CPU and is used to measure the BIOS firmware.
- The SHA1 hash of the result of the measurement is sent to the TPM.
- The TPM stores this new result hash by extending the currently stored value.
- The hash comparisons can validate settings, as well as the integrity of the modules.

Cloud infrastructure operators should ensure that OS kernel measurements can be recorded by using a TPM-aware bootloader (for example, tboot, (see [95]) or shim, (see [96])), which can extend the root of trust up to the kernel level.

The validation of the platform measurements can be performed by the TPM's launch control policy (LCP) or through the remote attestation server.

7.6.5.2 Dynamic Root of Trust for Measurement

In the Dynamic Root of Trust for Measurement (DRTM), the RTM for the running environment is stored in the PCRs starting with PCR 17.

If a remote attestation server is used to monitor platform integrity, the operators should ensure that the attestation is performed periodically or in a timely manner. Additionally, platform monitoring can be extended to monitor the integrity of the static file system at run-time by using a TPM-aware kernel module, such as Linux Integrity Measurement Architecture (IMA).

For details, see [97]. Alternatively, you can use the trust policies functionality of OpenCIT. See [98] for details.

The static file system includes a set of important files and folders which do not change between reboots during the platform's lifecycle. This allows the attestation server to detect any tampering with the static file system during the platform's runtime.

7.6.6 Zero Trust Architecture

The sections "Remote attestation/OpenCIT" and "Root of trust for measurements" provide methods to ensure the integrity of the infrastructure. The Zero Trust concept goes a step further, by enabling the operator to build secure by design cloud infrastructure, from hardware to applications. The adoption of Zero Trust principles mitigates the threats and attacks within an enterprise, a network, or an infrastructure, thereby ensuring a fine grained segmentation between each component of the system.

Zero Trust Architecture (ZTA), as described in NIST SP 800-207 publication [30], assumes there is no implicit trust granted to assets or user accounts, whatever their location or ownership. The Zero Trust approach focuses on protecting all types of resources: data, services, devices, infrastructure components, and virtual and cloud components. Trust is never granted implicitly, and must be evaluated continuously.

The ZTA principles applied to the cloud infrastructure components are as follows:

297. The adoption of least privilege configurations.

298. The requirement of authentication and authorization for each entity, service, or session.

299. Fine-grained segmentation.

300. Separation of the control plane and the data plane.

301. Secure internal and external communications.

302. The continuous monitoring, testing, and analysing of security.

Zero Trust principles should also be applied to cloud-native applications. With the increasing use of these applications, which are designed with microservices and deployed using containers as packaging and Kubernetes as an orchestrator, the security of east-west communications between components must be carefully addressed. The use of secured communication protocols brings a first level of security. However, considering each component as non-trustworthy will minimize the risk for applications to be compromised. A good practice is to implement the proxy-based service mesh. This will provide a framework to build a secured environment for microservices-based applications, offering services such as service discovery, authentication and authorisation policies enforcement, network resilience, and security monitoring capabilities. The two documents, NIST SP 800-204A [99] (Building Secure Microservices-based Applications Using Service-Mesh Architecture) and NIST SP 800-204B [100] (Attribute-based Access Control for Microservices-based Applications Using a Service Mesh), describe service mesh, and provide guidance for the deployment of service mesh components.

7.7 Software supply chain security

Software supply chain attacks are increasing worldwide and can cause serious damage. Many enterprises and organisations are experiencing these threats. Aqua Security's experts estimated that software supply chain attacks [101]. Reuters reported in August 2021 that the ransomware affecting Kaseya Virtual System Administration product [102] caused downtime for over 1500 companies. In the case of the backdoor inserted in codecov software [103], hundreds of customers were affected. The SolarWinds attack detailed in Defending against SolarWinds attacks [104] is another example of how software suppliers are targeted and, by rebound, their customers affected. Open-source code weaknesses can also be exploited by

attackers. The Log4J [105] vulnerability, impacting many applications, is a recent example in this field. When addressing cyber security, the vulnerabilities of the software supply chain are often not taken into account. Some governments are already alerting and requesting actions to face these risks. The British government is hardening the law and standards of cyber security for the supply chain. The US government requested actions to enhance software supply chain security. The security of the software supply chain is a challenge also pointed out by the European Network and Information Security Agency, ENISA, in its report NFV Security in 5G - Challenges and Best Practices [105].

7.7.1 Software security

Software supply chain security is crucial and is made complex by the greater attack surface provided by the many different supply chains in virtualised, containerised, and edge environments. All software components must be trusted, from commercial software and open-source code to proprietary software, as well as the integration of these components. The SAFECode white paper "Managing Security Risks Inherent in the Use of Third-party Components" [106] provides a detailed risk management approach.

To secure the software code, the following methods must be applied:

303. Use best practices coding, such as design pattern, recommended in the Twelve-Factor App [107] or OWASP [108].
304. Perform threat modelling, as described in the "Tactical Threat Modeling" [109] document, published by SAFECode.
305. Use trusted, authenticated, and identified software images that are provided by authenticated software distribution portals.
306. Require suppliers to provide a Software Bill of Materials to identify all the components parts of their product's software releases with their dependencies, and eventually identify the open-source modules.
307. Test the software in a pre-production environment to validate integration.
308. Detect vulnerabilities using security tools scanning and Common Vulnerabilities and Exposures (CVE), and apply remediation actions according to their severity ratings.
309. Report and remove vulnerabilities by upgrading components using authenticated software update distribution portals.
310. Actively monitor the open-source software repositories to determine if new versions have been released that address identified vulnerabilities discovered in the community.
311. Secure the integration process by securing the software production pipeline.
312. Adopt a DevSecOps approach and rely on testing automation throughout the software build, integration, delivery, deployment, and runtime operation to perform automatic security checks, as described in ”Infrastructure as a Code Security”.

7.7.2 Open-source software security

Open-source code is present in cloud infrastructure software from BIOS and host operating systems to virtualisation layer components. The most obvious is represented by Linux, KVM, QEMU, OpenStack, and Kubernetes. Workloads components can also be composed of open-source code. The proportion of open-source code to an application source code can vary. It can be partial or total, visible or invisible. Open-source code can be upstream code coming directly from open-source public repositories, or code within a commercial application or network function.

The strength of open-source code is the availability of code source developed by a community which maintains and improves it. Integration of open-source code with application source code

helps to develop and produce applications faster. This could, however, introduce security risks if a risk management DevSecOps approach is not implemented. The GSMA white paper "Open Networking & the Security of Open Source Software Deployment - Future Networks" [110] highlights these risks and addresses the challenges coming with open-source code usage. Among the security risks are poor code quality containing security flaws, an obsolete code with known vulnerabilities, and the lack of knowledge of the branch activity of the open-source communities. An active branch comes with bugs fixes. This is not the case with an inactive branch. The GSMA white paper develops means to mitigate these security issues.

Poor code quality is a risk factor. One advantage of open-source code is its transparency. Code can be inspected using tools with various capabilities, such as open-source software discovery, and static and dynamic code analysis.

Each actor in the whole chain of software production must use a dedicated internal isolated repository separated from the production environment to store vetted open-source content. This content can include images, as well as the installer and utilities. These software packages must be signed and the signature must be verified prior to the installation of the packages or images. Access to the repository must be granted by a dedicated authorization. The code must be inspected and vulnerabilities identified as described previously. After the software has been validated, it can be moved to the appropriate production repository.

7.7.3 Software Bill of Materials

To ensure software security, it is crucial to identify the software components and their origins. The Software Bill of Materials (SBOM), described by the US National Telecommunications and Information Administration US NTIA [111], is a recommended tool for identifying software components. The SBOM is an inventory of software components and the relationships between them. The NTIA describes how to establish an SBOM and provides SBOM standard data formats. In the event of vulnerability being detected in a component, the SBOM inventory is an effective means of identifying the impacted component and of enabling remediation.

A transparent software supply chain offers benefits for the remediation of vulnerabilities. It also offers benefits for licensing management and provides assurance of the source and integrity of the components. To achieve and benefit from this transparency, a shared model must be supported by industry. This is the goal of the work performed by the US Department of Commerce and the National Telecommunications and Information Administration (NTIA) and published, in the report "The Minimum Elements for a Software Bill of Materials (SBOM)" [112], in July 2021. This document gives guidance and specifies the minimum elements for the SBOM as a starting point.

A piece of software can be modelled as a hierarchical tree with components and subcomponents. Each component should have its SBOM, including, as a baseline, the information detailed in the following table.

Data field	Description
Supplier name	The name of an entity that creates, defines, and identifies the components.
Component name	The designation assigned to a unit of software defined by the original supplier.
Component version	The identifier used by the supplier to specify a change in the software from a previously identified version.
Other unique identifiers	Other identifiers that are used to identify a component, or to serve as a look-up key for relevant databases.
Dependency relationship	Characterization of the relationship that an upstream component X is included in software Y.

Author of SBOM data	The name of the entity that creates the SBOM data for this component.
Timestamp	The record of the date and time of the SBOM data assembly.

Table 7-2: SBOM components of the data fields. (Source: NTIA [113])

For more details about each data field, see the NTIA SBOM document. Examples of commonly used identifiers are provided.

To use SBOMs efficiently and encourage their widespread adoption, information must be generated and shared in a standard format. This format must be machine-readable, to allow automation. Proprietary formats should not be used. Multiple data formats exist covering baseline SBOM information. The three key formats, Software Package Data eXchange (SPDX), CycloneDX, and Software Identification Tags (SWID tags), are interoperable for the core data fields and use common data syntax representations.

313. SPDX [114] is an open-source machine-readable format developed under the umbrella of the Linux Foundation. SPDX specification 2.2 [115] has been published as the standard ISO/IEC 5962:2021. It provides a language for communicating the data, licenses, copyrights, and security information associated with software components. With SPDX specification 2.2, multiple file formats are available: YAML, JSON, RDF/XML, tag: value flat text, and xls spreadsheets.

314. CycloneDX [116] was designed in 2017 for use with the Open Web Application Security Project (OWASP) Dependency-Track tool, an open-source component analysis platform that identifies risk in the software supply chain. CycloneDX supports a wide range of software components, including applications, containers, libraries, files, firmware, frameworks, and operating systems. The CycloneDX project provides standards in XML, JSON, and Protocol Buffers, as well as a large collection of official and community supported tools that create or interoperate with the standard.

315. SWID Tags [117] is an international XML-based standard used by commercial software publishers. It has been published as the standard ISO/IEC 19770-2. The specification defines four types of SWID tags: primary, patch, corpus, and supplemental, to describe a software component.

The SBOM should be integrated into the operations of the secure development lifecycle, especially for vulnerabilities management. It should also evolve in time. When a software component is updated, a new SBOM must be created. The elements described in this section are part of an ongoing effort. Improvements, such as SBOM integrity and authenticity, will be added in the future.

7.7.4 Vulnerability identification

Vulnerability management must be continuous: from development to runtime, not only on the development process, but during the entire life of the application, workload, or service. When a public vulnerability on a component is released, an update of the component must be triggered. When an SBOM recording the code composition is provided, the affected components are easier to identify. It is essential to remediate the affected components as soon as possible, because the vulnerability can be exploited by attackers who can exploit code weaknesses.

The CVE and the CVSS must be used to identify vulnerabilities and their severity rating. The CVE identifies, defines, and catalogues publicly disclosed cybersecurity vulnerabilities. The CVSS is an open framework to calculate the severity rating of the vulnerabilities.

Image scanning tools, including open-source tools such as Clair or Trivy, are useful for auditing images for security vulnerabilities. The results of a vulnerabilities scan audit must be

analysed carefully when it is applied to a vendor offering packaged solutions. As patches are not detected by scanning tools, some components can be detected as obsolete.

7.8 Testing and certification

7.8.1 Testing demarcation points

It is not enough simply to secure all the potential points of entry and hope for the best. Any cloud infrastructure architecture must be able, as much as possible, to be tested and confirmed to be protected from attack. The ability to continuously test the infrastructure for vulnerabilities is critical for maintaining the highest possible level of security. Testing needs to be done both from the inside and the outside of the systems and networks. Below is a sample of some of the available testing methodologies and frameworks.

316. OWASP testing guide
317. Penetration Testing Execution Standard, PTES
318. Technical Guide to Information Security Testing and Assessment, NIST 800-115 [118]
319. Vulnerability Assessment Framework for Cloud Computing (VULCAN), IEEE 2013
320. Penetration Testing Framework, VulnerabilityAssessment.co.uk
321. Information Systems Security Assessment Framework (ISSAF)
322. Open Source Security Testing Methodology Manual (OSSTMM)
323. FedRAMP Penetration Test Guidance (US Only)
324. CREST Penetration Testing Guide

Ensuring that the security standards and best practices are incorporated into the cloud infrastructure and architectures must be a shared responsibility. The telecommunications operators interested in building and maintaining the infrastructures in support of their services, the application vendors developing the network services to be used by the operators, and the cloud infrastructure vendors creating the infrastructures for their telecommunications customers must all be responsible for performing this task. Each party needs to incorporate security and testing components, and maintain operational processes and procedures to address any security threats or incidents in an appropriate manner. Each of the stakeholders needs to contribute to the creation of effective security for the cloud infrastructure.

7.8.2 Certification requirements

Security certification should encompass the following elements:

325. Security test cases executed and test case results.
326. Industry standard compliance achieved (NIST, ISO, PCI, FedRAMP Moderate, and so on).
327. Output and analysis from automated static code reviews, dynamic tests, and penetration tests with remediation of medium, high, and critical security issues. Tools used for the security testing of software that is being released must be shared.
328. Details of unremedied low-severity security issues must be shared.
329. Threat models performed during the design phase. This includes remediation summaries to mitigate threats identified.
330. Details of unremedied low-severity security issues.
331. Any additional security and privacy requirements implemented in the software deliverable beyond the default rules used by security analysis tools.

332. Execution of resiliency tests, such as hardware failures or power failure tests.

7.9 Cloud infrastructure regulatory compliance

Evolving cloud adoption in the telecom industry, now encroaching on its inner sanctum of network services, brings many benefits for the network operators and their partners, and ultimately to the consumers of the telecommunication services. However, it also brings major compliance challenges. The telecommunication industry players can mitigate these challenges by arming themselves with information about which laws they need to comply with, why they need to comply with them, and how they can do this.

The costs of non-compliance can be heavy. Organisations may not only have to contend with hefty fines and possible lawsuits, but they may also end up damaging their reputation and, consequently, losing customers, which would, in turn, adversely affect revenues and profitability.

Compliance means that an operator's systems, processes, and workflows align with the requirements mandated by the regulatory regimes imposed by the relevant governmental and industry regulatory bodies. The need for compliance extends to the cloud. Therefore, operators must ensure that any data that is stored in their cloud infrastructure, as well as data that is transferred to and from it, complies with all the relevant data protection laws, including data residency and privacy laws.

To comply with the laws that apply to an operator's business, the correct security controls need to be applied. The applicable laws have specific rules and constraints about how companies can collect, store, and process data in the cloud. To satisfy these constraints and ensure compliance, the telecom operators should work with their cloud providers and other partners to implement tight controls. To speed up this process, the operators may start from augmenting their existing cybersecurity and information security frameworks to guide their security programs to implement controls to secure their cloud infrastructure and to achieve regulatory compliance. This process can also be assisted by support from the cloud providers and from third parties, who can offer their well-proven compliance offerings, resources, audit reports, dashboards, and certain security controls as a service.

After implementing these controls, companies need to train their employees and partners to use the controls properly, to protect data and maintain the required compliance posture. This is a critical requirement to maintain compliance by enforcing relevant security guidelines in all aspects of everyday operations, as well as for ensuring a process of regular assessment of the compliance posture.

Due to the localised nature of the regulatory regimes, this document may not provide any specific compliance requirements. However, some examples provided below may help in an operator' compliance considerations.

Commonly used (in many jurisdictions) compliance audit reports are based on the SOC 2 report from the System and Organization Controls (SOC) suite of services, standardised by the American Institute of Certified Public Accountants (AICPA) and meant for service organizations, such as cloud providers. See AICPA SOC[119] for details. A SOC 2 report shows whether the cloud provider has implemented the security controls required to comply with the AICPA's five "trust services criteria": security, availability, confidentiality, processing integrity, and privacy. Operators should request a SOC 2 report from their cloud providers (public or internal to their organisations). SOC 2 comprises two types: type 1 and type 2. A type 1 report shows the status and suitability of the provider's controls at a particular moment. A type 2 report shows the operational effectiveness of these controls over a certain period. In cases where a cloud provider is not willing to share an SOC 2 report because it may contain sensitive information, operators can ask for an SOC 3 report. This is intended as a general-use report, but can still help to assess the provider's compliance posture.

Some cloud providers also provide attestations (or, in the case of the private cloud, telecoms should seek such attestations) to show which of their cloud services have achieved compliance with different frameworks, such as SOC, but also commonly used frameworks, such as OWASP, ISAE, NIST, ETSI, and the ISO 27000 series, as well as more geographically localised standard frameworks such as NIST (as used in the U.S.A.), ENISA, GDPR, and ISM.

The use of the ISO 2700s, OWASP, ISAE, NIST, and ETSI security frameworks for the cloud infrastructure is referenced in the Common Security Standards and Compliance with Standards sections.

Examples of regulatory frameworks are presented below. It is intended to expand this list of examples in future releases to cover more jurisdictions and to accommodate changes in the rapidly evolving security and regulatory landscape.

7.9.1 United States of America (U.S.A.)

In the United States, the Federal Communications Commission (FCC) [120] regulates interstate and international communications by radio, television, wire, satellite, and cable in all 50 states, the District of Columbia, and all U.S. overseas territories. The FCC is an independent U.S. government agency overseen by Congress. The Commission is the federal agency responsible for implementing and enforcing America's communications laws and regulations.

National Institute of Standards and Technology (NIST) Cybersecurity Framework [121], compliance is mandatory for the supply chain for all U.S. federal government agencies. Because this framework references globally accepted standards, guidelines, and practice, telecom organisations in the U.S.A. and worldwide can use it to efficiently operate in a global environment and manage new and evolving cybersecurity risks in the cloud adoption area.

7.9.2 European Union (EU)

The overall telecommunications regulatory framework in the European Union (EU) is provided in The European Electronic Communications Code [122].

The European Union Agency for Cybersecurity (ENISA) [123] contributes to EU cyber policy, enhances the trustworthiness of Information and Communications Technology (ICT) products, services, and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. In particular, ENISA performs risk assessments of cloud computing and works on the European Cybersecurity Certification Scheme (EUCS) [124] for Cloud Services, which looks into the certification of the cybersecurity of cloud services.

The General Data Protection Regulation (GDPR) [125] is a set of EU regulations that govern how data should be protected for EU citizens. It affects organisations that have EU-based customers, even if these organisations are not themselves based in the EU.

7.9.3 United Kingdom (UK)

The Office of Communications (Ofcom) [126] is the regulator and competition authority for the UK communications industries. It regulates the television and radio sectors, fixed-line telecoms, mobiles, postal services, and the airwaves over which wireless devices operate.

The Security of Networks and Information Systems NIS Regulations in UK[127], provides legal measures to boost the level of security (both cyber and physical resilience) of network and information systems for the provision of essential services and digital services.

The UK's National Cyber Security Centre (NCSC) [128] acts as a bridge between industry and government. It provides a unified source of advice, guidance, and support on cyber security, including the management of cyber security incidents. From this perspective, it is critical for

cloud-related security in the UK telecommunications industry. The NCSC is not a regulator. Within the general UK cyber security regulatory environment, including both NIS and GDPR, the NCSC's aim is to operate as a trusted, expert, and impartial advisor to all interested parties. The NCSC supports Security of Networks & Information Systems (NIS) Regulations.

Data protection in UK is controlled by Data Protection Act 2018 [129], which is UK's implementation of the EU's General Data Protection Regulation (GDPR).

7.9.4 Australia

In Australia, the telecommunication sector is regulated by the Australian Competition & Consumer Commission (ACCC) [130]. The ACCC is responsible for the economic regulation of the communications sector. This includes telecommunications, the National Broadband Network (NBN), and the broadcasting and content sectors.

From the point of view of cloud services security, the Information Security Manual (ISM) [131], produced by the Australian Cyber Security Centre (ACSC), is of particular importance. The purpose of the ISM is to outline a cyber security framework that organisations can apply, using their risk management framework, to protect their information and systems from cyber threats. The ISM is intended for Chief Information Security Officers, Chief Information Officers, cyber security professionals, and information technology managers. While in general the ISM provides guidelines rather than mandates, several security controls are, by law, mandatory for cloud-based services used by the Australian telecommunications operators, in situations involving strategically important data and/or services.

Australia regulates data privacy and protection through a mix of federal, state, and territory laws. The federal Privacy Act 1988 [132] (currently under review by the Australian Government) and the Australian Privacy Principles (APPs), contained in the Privacy Act, regulate the handling of personal information by relevant entities and under the Privacy Act. The Privacy Commissioner has the authority to conduct investigations, including its own motion investigations, to enforce the Privacy Act and to seek civil penalties for serious and egregious breaches, or for repeated breaches of the APPs where an entity has failed to implement remedial efforts.

7.10 Consolidated security requirements

7.10.1 System hardening

Table 11Profile extensions

Ref	Requirement	Definition/Note
req.sec.gen.001	The platform **must** maintain the specified configuration.	
req.sec.gen.002	The systems part of the cloud infrastructure **must** support password hardening, as defined in CIS Password Policy Guide [147].	Hardening: CIS Password Policy Guide
req.sec.gen.003	The servers part of the cloud infrastructure **must** support a root of trust and secure boot.	
req.sec.gen.004	The operating systems of all the parts of the cloud infrastructure dealing with servers **must** be hardened by removing or disabling unnecessary services, applications, and network protocols, configuring operating system user authentication, configuring resource controls, installing and	NIST SP 800-123

	configuring additional security controls where needed, and testing the security of the operating system.
req.sec.gen.005	The platform **must** support operating system-level access control.
req.sec.gen.006	The platform **must** support secure logging. Logging with the root account must be prohibited when the root privileges are not required.
req.sec.gen.007	All parts of the cloud infrastructure dealing with servers **must** be time-synchronized with the authenticated time service.
req.sec.gen.008	All parts of the cloud infrastructure dealing with servers **must** be regularly updated to address security vulnerabilities.
req.sec.gen.009	The platform **must** support software integrity protection and verification, and **must** scan the source code and manifests.
req.sec.gen.010	The cloud infrastructure **must** support encrypted storage, for example, block, object, and file storage, with access to encryption keys restricted on a need-to-know basis. Controlled access based on a need to know [148].
req.sec.gen.011	The cloud infrastructure **should** support read- and write-only storage partitions (write-only permission to one or more authorized actors).
req.sec.gen.012	The operator **must** ensure that only authorized actors have physical access to the underlying infrastructure.
req.sec.gen.013	The platform **must** ensure that only authorized actors have logical access to the underlying infrastructure.
req.sec.gen.014	All parts of the cloud infrastructure dealing with servers **should** support measured boot and an attestation server that monitors the measurements of the servers.
req.sec.gen.015	Any change to the platform **must** be logged as a security event. The logged event must include the identity of the entity making the change, the change itself, and the date and time of the change.

Table 7-3: System hardening requirements

7.10.2 Platform and access

Ref	Requirement	Definition/Note
req.sec.sys.001	The platform **must** support authenticated and secure access to APIs, GUIs, and command line interfaces (CLIs).	

req.sec.sys.002	The platform **must** support traffic filtering for workloads (for example, firewalls).	
req.sec.sys.003	The platform **must** support secure and encrypted communications, and the confidentiality and integrity of network traffic.	
req.sec.sys.004	The cloud infrastructure **must** support authentication, integrity, and confidentiality on all network channels.	A secure channel enables the transferring of data that is resistant to overhearing and tampering.
req.sec.sys.005	The cloud infrastructure **must** segregate the underlay and overlay networks.	
req.sec.sys.006	The cloud infrastructure must be able to utilize the cloud infrastructure manager identity lifecycle management capabilities.	
req.sec.sys.007	The platform **must** implement controls enforcing the separation of duties and privileges, least privilege use and least common mechanism (role-based access control).	
req.sec.sys.008	The platform **must** be able to assign the entities that comprise the tenant networks to different trust domains.	Communication between different trust domains is not allowed, by default.
req.sec.sys.009	The platform **must** support the creation of trust relationships between trust domains.	These may be uni-directional relationships where the trusting domain trusts another domain (the “trusted domain”) to authenticate users for them, or to allow access to its resources from the trusted domain. In a bidirectional relationship, both domains are “trusting” and “trusted”.
req.sec.sys.010	For two or more domains without existing trust relationships, the platform **must not** allow the effect of an attack on one domain to impact the other domains, either directly or indirectly.	
req.sec.sys.011	The platform **must not** reuse the same authentication credentials (for example, a key-pair) on different platform components (for example, on	

	different hosts, or different services).	
req.sec.sys.012	The platform **must** protect all secrets by using strong encryption techniques and storing the protected secrets externally from the component.	For example, in OpenStack Barbican.
req.sec.sys.013	The platform **must** provide secrets dynamically as and when needed.	
req.sec.sys.014	The platform **should** use Linux Security Modules, such as SELinux, to control access to resources.	
req.sec.sys.015	The platform **must not** contain back-door entries (such as unpublished access points, APIs, and so on).	
req.sec.sys.016	Login access to the platform's components **must** be through encrypted protocols, such as SSH v2 or TLS v1.2, or higher.	Hardened jump servers isolated from external networks are recommended.
req.sec.sys.017	The platform **must** provide the capability of using digital certificates that comply with X.509 standards issued by a trusted certification authority.	
req.sec.sys.018	The platform **must** provide the ability to allow certificate renewal and revocation.	
req.sec.sys.019	The platform **must** provide the ability to test the validity of a digital certificate (CA signature, validity period, non-revocation, identity).	
req.sec.sys.020	The cloud infrastructure architecture **should** rely on Zero Trust principles to build a secure by design environment.	Zero Trust Architecture (ZTA) described in NIST SP 800-207

Table 7-4: Platform and access requirements

7.10.3 Confidentiality and integrity

Ref	Requirement	Definition/Note
req.sec.ci.001	The platform **must** support the confidentiality and integrity of data at rest and in transit.	
req.sec.ci.002	The platform **should** support self-encrypting storage devices.	

req.sec.ci.003	The platform **must** support the confidentiality and integrity of data-related metadata.	
req.sec.ci.004	The platform **must** support the confidentiality of processes and restrict information sharing to the process owner only (for example, the tenant).	
req.sec.ci.005	The platform **must** support the confidentiality and integrity of process-related metadata and restrict information sharing to the process owner only (for example, the tenant).	
req.sec.ci.006	The platform **must** support the confidentiality and integrity of workload resource utilization (RAM, CPU, storage, network I/O, cache, hardware offload), and restrict information sharing to the workload owner only (for example, the, tenant).	
req.sec.ci.007	The platform **must not** allow memory inspection by any other actor than the authorized actors for the entity to which memory is assigned (such as the tenants owning the workload), for lawful inspection, and by secure monitoring services.	Admin access must be carefully regulated.
req.sec.ci.008	The cloud infrastructure **must** support the segregation of the tenant networks.	
req.sec.ci.009	For sensitive data encryption, the key management service **should** leverage a hardware security module to manage and protect cryptographic keys.	

Table 7-5: Confidentiality and integrity requirements

7.10.4 Workload security

Ref	Requirement	Definition/Note
req.sec.wl.001	The platform **must** support a workload placement policy.	
req.sec.wl.002	The cloud infrastructure **must** provide methods to ensure the platform's trust status and integrity (for example, remote attestation, trusted platform module).	
req.sec.wl.003	The platform **must** support the secure provisioning of workloads.	
req.sec.wl.004	The platform **must** support location assertion (for mandated in-country or location requirements).	
req.sec.wl.005	The platform **must** support the separation of production and non-production workloads.	
req.sec.wl.006	The platform **must** support the separation of workloads based on their categorisation (for example, payment card information, healthcare, and so on).	
req.sec.wl.007	The operator **should** implement processes and tools to verify NF authenticity and integrity.	

Table 7-6: Workload security requirements

7.10.5 Image security

Ref	Requirement	Definition/Note
req.sec.img.001	Images must be scanned, in order to be kept free from known vulnerabilities.	
req.sec.img.002	Images must be scanned, in order to be kept free from known vulnerabilities.	
req.sec.img.003	Images must not be configured to run with privileges higher than those of the actor who is authorized to run them.	
req.sec.img.004	Images **must** only be accessible to authorized actors.	
req.sec.img.005	Image registries **must** only be accessible to authorized actors.	
req.sec.img.006	Image registries **must** only be accessible over secure networks that enforce authentication, integrity, and confidentiality.	
req.sec.img.007	Image registries **must** be clear of vulnerable and out of date versions.	
req.sec.img.008	Images **must not** include any secrets. Secrets include passwords, cloud provider credentials, SSH keys, TLS certificate keys, and so on.	
req.sec.img.009	CIS hardened images **should** be used whenever possible.	
req.sec.img.010	Minimalist base images **should** be used whenever possible.	

Table 7-7: Image security requirements

7.10.6 Security LCM

Ref	Requirement	Definition/Note
req.sec.lcm.001	The platform **must** support secure provisioning, availability, and deprovisioning (secure clean-up) of workload resources where secure clean-up includes tear-down, and defence against virus attacks or other attacks.	Secure clean-up: tear-down, defending against virus attacks or other attacks, or observing of cryptographic or user service data.
req.sec.lcm.002	Cloud operations staff and systems **must** use management protocols limiting security risk, such as SNMPv3, SSH v2, ICMP, NTP, syslog, and TLS v1.2 or higher.	
req.sec.lcm.003	The cloud operator **must** implement and strictly follow change management processes for the cloud infrastructure, cloud infrastructure manager, and other components of the cloud, and platform change control on hardware.	

req.sec.lcm.004	The cloud operator **should** support automated templated approved changes.	Templated approved changes for automation where available.
req.sec.lcm.005	The platform **must** provide logs. These logs must be regularly monitored for anomalous behaviour.	
req.sec.lcm.006	The platform **must** verify the integrity of all resource management requests.	
req.sec.lcm.007	The platform **must** be able to update newly instantiated, suspended, hibernated, migrated, and restarted images with the current time information.	
req.sec.lcm.008	The platform **must** be able to update newly instantiated, suspended, hibernated, migrated, and restarted images with the relevant DNS information.	
req.sec.lcm.009	The platform **must** be able to update the tags of newly instantiated, suspended, hibernated, migrated, and restarted images with the relevant geolocation (geographical) information.	
req.sec.lcm.010	The platform **must** log all changes to geolocation along with the mechanisms and sources of the location information (that is, GPS, IP block, and timing).	
req.sec.lcm.011	The platform **must** implement security lifecycle management processes, including the proactive update and patching of all deployed cloud infrastructure software.	
req.sec.lcm.012	The platform **must** log any access privilege escalation.	

Table 7-8: Security LCM requirements

7.10.7 Monitoring and security audit

The platform is assumed to provide configurable alerting and notification capabilities. The operator is assumed to have systems, policies, and procedures in place to act on alerts and notifications in a timely fashion. In the following, the monitoring and logging capabilities can trigger alerts and notifications for appropriate action. In general, it is recommended to have the same security monitoring and auditing capabilities in both production and non-production environments. However, we distinguish between the requirements for the production platform (Prod-Platform) and the non-production platform (NonProd-Platform), as some of the requirements may, in practice, need to differ. See Security of production and non-production environments for a general discussion of this topic. In the table below, when a requirement mentions only Prod-Platform, it is assumed that this requirement is optional for the NonProd-Platform. If a requirement does not mention either platform, it is assumed that it is valid for both the Prod-Platform and the NonProd-Platform.

Ref	Requirement	Definition/Note

req.sec.mon.001	The Prod-Platform and NonProd-Platform **must** provide logs. The logs **must** contain the following fields: event type, date/time, protocol, the service or program used for access, success/failure, the login ID or process ID, the IP address, and the ports (source and destination) involved.
req.sec.mon.002	The logs **must** be regularly monitored for events of interest.
req.sec.mon.003	The logs **must** be time-synchronised for the Prod-Platform, as well as for the NonProd-Platform.
req.sec.mon.004	The Prod-Platform and the NonProd-Platform **must** log all changes to the time server source, time, and date and time zones.
req.sec.mon.005	The Prod-Platform and the NonProd-Platform **must** secure and protect all the logs containing sensitive information, both in transit and at rest.
req.sec.mon.006	The Prod-Platform and the NonProd-Platform **must** monitor and audit the various behaviours of connection and login attempts to detect access attacks and potential access attempts, and take corrective action accordingly.
req.sec.mon.007	The Prod-Platform and the NonProd-Platform **must** Monitor and Audit operations by authorized account access after login, to detect malicious operational activity and take corrective action.
req.sec.mon.008	The Prod-Platform **must** monitor and audit security parameter configurations for compliance with defined security policies.
req.sec.mon.009	The Prod-Platform and the NonProd-Platform **must** monitor and audit externally exposed interfaces for illegal access (attacks) and take corrective security hardening measures.
req.sec.mon.010	The Prod-Platform **must** monitor and audit the service for various attacks (malformed messages, signalling flooding and replaying, and so on), and take corrective action.
req.sec.mon.011	The Prod-Platform **must** monitor and audit running processes to detect unexpected or unauthorized processes, and take corrective action.
req.sec.mon.012	The Prod-Platform and the NonProd-Platform **must** monitor and audit logs from the infrastructure elements and workloads to detect anomalies in the system components and take corrective action.
req.sec.mon.013	The Prod-Platform and the NonProd-Platform **must** monitor and audit traffic patterns and volumes to prevent malware download attempts.
req.sec.mon.014	The monitoring system **must not** affect the security (integrity and confidentiality) of the infrastructure,

	workloads, or the user data (through back-door entries).
req.sec.mon.015	The monitoring systems **should not** impact IaaS, PaaS, and SaaS SLAs, including availability SLAs.
req.sec.mon.016	The Prod-Platform and the NonProd-Platform **must** ensure that the monitoring systems are never starved of resources and **must** activate alarms when resource utilisation exceeds a configurable threshold.
req.sec.mon.017	The Prod-Platform and the NonProd-Platform monitoring components **should** follow security best practices for auditing, including secure logging and tracing.
req.sec.mon.018	The Prod-Platform and the NonProd-Platform **must** audit systems for any missing security patches and take appropriate action.
req.sec.mon.019	The Prod-Platform, starting from initialization, **must** collect and analyse logs to identify security events and store these events in an external system.
req.sec.mon.020	The components of the Prod-Platform and the NonProd-Platform **must not** include any authentication credentials, such as passwords, in any logs, even if they are encrypted.
req.sec.mon.021	The logging system of the Prod-Platform and the NonProd-Platform **must** support the storage of the security audit logs for a configurable period.
req.sec.mon.022	The Prod-Platform **must** store the security events locally, if the external logging system is unavailable. It must periodically attempt to send them to the external logging system until it is successful.

Table 7-9: Monitoring and security audit requirements

7.10.8 Open-source software

Table 12Profile extensions

Ref	Requirement	Definition/Note
req.sec.oss.001	Open-source code **must** be inspected by tools with various capabilities for static and dynamic code analysis.	
req.sec.oss.002	The Common Vulnerabilities and Exposures (CVE) **must** be used to identify vulnerabilities and assess their severity rating for the open-source code part of the cloud infrastructure and workloads software.	https://cve.mitre.org/
req.sec.oss.003	Critical- and high-severity-rated vulnerabilities **must** be fixed in a timely manner. See the Common Vulnerability Scoring System (CVSS) to find out a	https://www.first.org/cvss/

	vulnerability score and its associated rate (low, medium, high, or critical).	
req.sec.oss.004	A dedicated internal isolated repository separated from the production environment **must** be used to store vetted open-source content.	
req.sec.oss.005	A Software Bill of Materials (SBOM) **should** be provided or built, and maintained to identify the software components and their origins.	Inventory of software components, https://ntia.gov/SBOM

Table 7-10: Open-source software requirements

7.10.9 IaaC - Secure design and architecture stage requirements

Ref	Requirement	Definition/Note
req.sec.arch.001	Threat modelling methodologies and tools **should** be used during the Secure Design and Architecture stage, triggered by the Software Feature Design trigger.	These are methodologies for identifying and understanding threats impacting a resource or a set of resources. It may be done manually or by using tools such as the open-source OWASP Threat Dragon.
req.sec.arch.002	A Security Control Baseline Assessment **should** be performed during the Secure Design and Architecture stage, triggered by the Software Feature Design trigger.	Security Control Baseline Assessments are typically done manually by internal or independent assessors.

Table 7-11: IaaC - Secure design and architecture stage requirements

7.10.10 IaaC - Secure code stage requirements

Ref	Requirement	Definition/Note
req.sec.code.001	Static Application Security Testing (SAST) **must** be applied during the secure coding stage, triggered by the Pull, Clone or Comment trigger.	SAST is a type of security testing that analyses application source code for software vulnerabilities and gaps against best practices. An example of this is the open-source OWASP range of tools.
req.sec.code.002	Software Composition Analysis (SCA) **should** be applied during the secure coding stage, triggered by the Pull, Clone, or Comment trigger.	SCA is a type of security testing that analyses application source code or compiled code for software components with known vulnerabilities. An example of this is the open-source OWASP range of tools.
req.sec.code.003	A source code review **should** be performed continuously during Secure Coding stage.	This is typically done manually.

req.sec.code.004	Integrated SAST via IDE plugins **should** be used during the secure coding stage, triggered by the Developer Code trigger.	On the local machine: through the IDE or integrated test suites and triggered on completion of the coding development.
req.sec.code.005	Static Application Security Testing of the source code repo **should** be performed during the secure coding stage, triggered by the Developer Code trigger.	Continuous delivery pre-deployment: scanning prior to deployment.

Table 7-12: IaaC - Secure code stage requirements

7.10.11 IaaC - Continuous build, integration, and testing stage requirements

Ref	Requirement	Definition/Note
req.sec.bld.001	Static Application Security Testing (SAST) **should** be applied during the continuous build, integration, and testing stage, triggered by the Build and Integrate trigger.	Example: the open-source OWASP range of tools.
req.sec.bld.002	Software Composition Analysis (SCA) **should** be applied during the continuous build, integration, and testing stage, triggered by the Build and Integrate trigger.	Example: the open-source OWASP range of tools.
req.sec.bld.003	An image scan **must** be applied during the continuous build, integration, and testing stage, triggered by the Package trigger.	Example: pushing a container image to a container registry may trigger a vulnerability scan before the image becomes available in the registry.
req.sec.bld.004	Dynamic Application Security Testing (DAST) **should** be applied during the continuous build, integration, and testing stage, triggered by the Stage and Test trigger.	DAST is a type of security testing that analyses a running application by exercising the application functionality and detecting vulnerabilities based on the behaviour and response of the application. Example: OWASP Zed Attack Proxy (ZAP).
req.sec.bld.005	Fuzzing **should** be applied during the continuous build, integration, and testing stage, triggered by the Stage and Test trigger.	Fuzzing, or fuzz testing, is an automated software testing technique that involves providing invalid, unexpected, or random data as input to a computer program. Example: GitLab Open Sources Protocol Fuzzer Community Edition.
req.sec.bld.006	Interactive Application Security Testing (IAST) **should** be applied during the continuous build, integration,	A software component deployed with an application that assesses the behaviour of the application and detects the presence of vulnerabilities on

	and testing stage, triggered by the Stage and Test trigger.	applications that are being exercised in realistic testing scenarios. Example: Contrast Community Edition.

Table 7-13: IaaC - Continuous build, integration, and testing stage requirements

7.10.12 IaaC - Continuous delivery and deployment stage requirements

Ref	Requirement	Definition/Note
req.sec.del.001	An image scan **must** be applied during the continuous delivery and deployment stage, triggered by the Publish to Artifact and Image Repository trigger.	Example: GitLab uses the open-source Clair engine for container image scanning.
req.sec.del.002	Code signing **must** be applied during the continuous delivery and deployment stage, triggered by the Publish to Artifact and Image Repository trigger.	Code signing provides authentication to assure that downloaded files are from the publisher named on the certificate.
req.sec.del.003	An artifact and image repository scan **should** be continuously applied during the continuous delivery and deployment stage.	Example: GitLab uses the open-source Clair engine for container scanning.
req.sec.del.004	A component vulnerability scan **must** be applied during the continuous delivery and deployment stage, triggered by the Instantiate Infrastructure trigger.	The vulnerability scanning system is deployed on the cloud platform to detect the security vulnerabilities of specified components through scanning and to provide timely security protection. Example: OWASP Zed Attack Proxy (ZAP).

Table 7-14: IaaC - Continuous delivery and deployment stage requirements

7.10.13 IaaC - Runtime defence and monitoring requirements

Ref	Requirement	Definition/Note
req.sec.run.001	Component vulnerability monitoring **must** be continuously applied during the runtime defence and monitoring stage, and remediation actions **must** be applied for high-severity-rated vulnerabilities.	Security technology that monitors components such as virtual servers and assesses data, applications, and infrastructure for security risks.
req.sec.run.002	Runtime Application Self-Protection (RASP) **should** be continuously applied during the runtime defence and monitoring stage.	Security technology deployed within the target application in production for detecting, alerting, and blocking attacks.
req.sec.run.003	Application testing and fuzzing **should** be continuously applied	Fuzzing, or fuzz, testing is an automated software testing

	during the runtime defence and monitoring stage.	technique that involves providing invalid, unexpected, or random data as input to a computer program. Example: GitLab Open Sources Protocol Fuzzer Community Edition.
req.sec.run.004	Penetration testing **should** be continuously applied during the runtime defence and monitoring stage.	This is typically done manually.

Table 7-15: IaaC - Runtime defence and monitoring requirements

7.10.14 Compliance with standards

Table 13Profile extensions

Ref	Requirement	Definition/Note
req.sec.std.001	The cloud operator **should** comply with the Center for Internet Security CIS Controls.	Center for Internet Security - https://www.cisecurity.org/
req.sec.std.002	The cloud operator, platform, and workloads **should** follow the guidance in the CSA Security Guidance for Critical Areas of Focus in Cloud Computing (latest version).	Cloud Security Alliance - https://cloudsecurityalliance.org/
req.sec.std.003	The platform and workloads **should** follow the guidance in the OWASP Cheat Sheet Series (OCSS).	Open Web Application Security Project - https://owasp.org
req.sec.std.004	The cloud operator, platform, and workloads **should** ensure that their code	https://owasp.org/www-project-top-ten/

	is not vulnerable to the OWASP Top Ten Security Risks.	
req.sec.std.005	The cloud operator, platform, and workloads **should** strive to improve their maturity on the OWASP Software Maturity Model (SAMM).	https://owaspsamm.org/blog/2019/12/20/version2-community-release/
req.sec.std.006	The cloud operator, platform, and workloads **should** utilize the OWASP Web Security Testing Guide.	https://github.com/OWASP/wstg/tree/master/document
req.sec.std.007	The cloud operator and platform **should** satisfy the requirements for Information Management Systems, specified in ISO/IEC 27001.	ISO/IEC 27002:2013 - ISO/IEC 27001 is the international standard for best-practice information security management systems (ISMSs).
req.sec.std.008	The cloud operator and platform **should** implement the code of practice for Security Controls specified in ISO/IEC 27002:2013 (or latest).	https://www.iso.org/obp/ui/#iso:std:iso-iec:27002:ed-2:v1:en
req.sec.std.009	The cloud operator and	https://www.iso.org/obp/ui/#iso:std:iso-iec:27032:ed-1:v1:en

	platform **should** implement the ISO/IEC 27032:2012 (or latest) Guidelines for Cybersecurity techniques.	
req.sec.std.010	The cloud operator **should** conform to the ISO/IEC 27035 standard for incidence management.	ISO/IEC 27035 - ISO/IEC 27035 is the international standard for incident management.
req.sec.std.011	The cloud operator **should** conform to the ISO/IEC 27031 standard for business continuity.	ISO/IEC 27031 is the international standard for ICT readiness for business continuity.
req.sec.std.012	The public cloud operator **must**, and the private cloud operator **may**, be certified to be compliant with the International Standard on Awareness Engagements (ISAE) 3402 (in the US: SSAE 16).	International Standard on Awareness Engagements (ISAE) 3402. US Equivalent: SSAE16.

Table 7-16: Compliance with standards requirements

7.11 Additional security references

In addition to the security standards used throughout this specification, the following lists gather together additional standards of interest for cloud infrastructure security.

ETSI Documents

333. Network Functions Virtualisation (NFV); NFV Security; Problem Statement, ETSI GS NFV-SEC 001 V1.1.1 (2014-10)

334. Network Functions Virtualisation (NFV); NFV Security; Security and Trust Guidance, ETSI GS NFV-SEC 003 V1.1.1 (2014-12)

335. Network Functions Virtualisation (NFV) Release 3; Security; Security Management and Monitoring specification, ETSI GS NFV-SEC 013 V3.1.1 (2017-02)

336. Network Functions Virtualisation (NFV) Release 3; NFV Security; Security Specification for MANO Components and Reference points, ETSI GS NFV-SEC 014 V3.1.1 (2018-04)

337. Network Functions Virtualisation (NFV) Release 2; Security; VNF Package Security Specification, ETSI GS NFV-SEC 021 V2.6.1 (2019-06)

NIST Documents

338. NIST SP 800-53 Security and Privacy Controls for Federal Information Systems and Organizations [133]

339. NIST SP 800-53A Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans [134]

340. NIST SP 800-63B Digital Identity Guidelines [135]

341. NIST SP 800-115 Technical Guide to Information Security Testing and Assessment [136]

342. NIST SP 800-125 Guide to Security for Full Virtualization Technologies [137]

343. NIST SP 800-125a Security Recommendations for Server-based Hypervisor Platforms [138]

344. NIST SP 800-125b Secure Virtual Network Configuration for Virtual Machine (VM) Protection [139]

345. NIST SP 800-137 Information Security Continuous Monitoring for Federal Information Systems and Organizations [140]

346. NIST SP 800-145 The NIST Definition of Cloud Computing [141]

347. NIST SP 800-190 Application Container Security Guide [142]

8 Hybrid Multicloud: Data Centre to Edge

8.1 Introduction to Hybrid Multicloud

The Modelling section focuses on cloud infrastructure abstractions. While these are generic abstractions, they and the associated capabilities of the cloud infrastructure are specified for data centres, central offices, and colocation centres. The environmental conditions, facility and other constraints, and the variability of deployments on the edge are significantly different and therefore require separate consideration.

It is unrealistic to expect a private cloud to meet the needs of all the workloads in a cost-effective way when the private cloud must also meet the needs for peak loads and disaster recovery. For this reason alone, enterprises will need to implement a hybrid cloud. In a hybrid cloud deployment, two or more distinct cloud infrastructures are interconnected. In a multicloud, the distinct cloud infrastructures of the hybrid cloud may be implemented using one or more technologies. The hybrid multicloud infrastructure has differences requiring different abstractions. These hybrid multiclouds can be considered to be federated.

In the Modelling section, the cloud infrastructure is defined. The tenants are required to provide certain necessary services, such as load balancer (LB) and messaging. Therefore, the VNF/CNFs incorporate different versions of the same services with the resultant issues related to an explosion of services, together with their integration and management complexities. To mitigate these issues, the Reference Model must specify the common services that every Telco cloud must support, and thereby require workload developers to utilise these prespecified services.

A generic Telco cloud is a hybrid multicloud, or a federated cloud, that has deployments in large data centres, central offices or colocation facilities, and the edge sites. This chapter discusses the characteristics of Telco edge and hybrid multicloud.

8.2 Hybrid multicloud architecture

The GSMA white paper "Operator Platform Concept Phase 1: Edge Cloud Computing" (January 2020) states: "Given the wide diversity of use cases that the operators will tasked to address, from healthcare to industrial IoT, it seems logical for operators to create a generic platform that can package the existing assets and capabilities (e.g., voice messaging, IP data services, billing, security, identity management, etc. ...) as well as the new ones that 5G makes available (e.g., Edge cloud, network slicing, etc.) in such a way as to create the necessary flexibility required by this new breed of enterprise customers."

Cloud computing has evolved and matured since 2010 when NIST [141] published its definition of cloud computing, with its 5 essential characteristics, 3 service models and 4 deployment models.

The generic model for an enterprise cloud has to be "hybrid", with the special cases of purely private or public clouds as subsets of the generic hybrid cloud deployment model. In a hybrid cloud deployment, two or more distinct cloud infrastructures are interconnected.

Cloud deployments can be created using a variety of technologies, such as OpenStack and Kubernetes, and commercial technologies, such as VMware, AWS, Azure, and so on. A multicloud deployment can use more than one technology at a time.

A generic Telco cloud is a hybrid multicloud. However, a better way to describe it would be a federation of clouds, or a federated cloud. The principal characteristics of a federated cloud are as follows:

348. A federated cloud is a collection of cooperating, interoperable autonomous component clouds.
349. The component clouds perform their local operations (internal requests), while at the same time participating in the federation and responding to other component clouds (external requests).
    1. the component clouds are autonomous in terms of, for example, execution autonomy; please note that in a centralised control plane scenario (please see the section "Centralised Control Plane" in the "Edge Computing: Next Steps in Architecture, Design and Testing" whitepaper [31]) the edge clouds do not have total autonomy and are subject to constraints (e.g., workload LCM)
    2. execution autonomy is the ability of a component cloud to decide the order in which internal and external requests are performed
    3. also, a federation controller does not impose changes to the component cloud except for running some central component(s) of the federated system (for example, a broker agent – executes as a workload)
350. The component clouds are likely to differ in, for example, infrastructure resources and their cloud platform software.
351. Workloads may be distributed on single or multiple clouds, where the clouds may be co-located or geographically distributed.
352. The component clouds only expose their northbound APIs (NBIs). Note that VMware deployed in a private and a public cloud can be treated as a single cloud instance.

8.3 Characteristics of a federated cloud

In this section, we will further explore the characteristics of the federated cloud architecture and the architecture building blocks that constitute the federated cloud. For example, Example hybrid multicloud component cloud shows a Telco cloud that consists of four sub-clouds:

353. Private on premise

354. Cloud vendor provided on premise

355. Private outsourced (commercial cloud provider, such as a Hyperscaler Cloud Provider (HCP)

356. Public outsourced (see the diagram below)

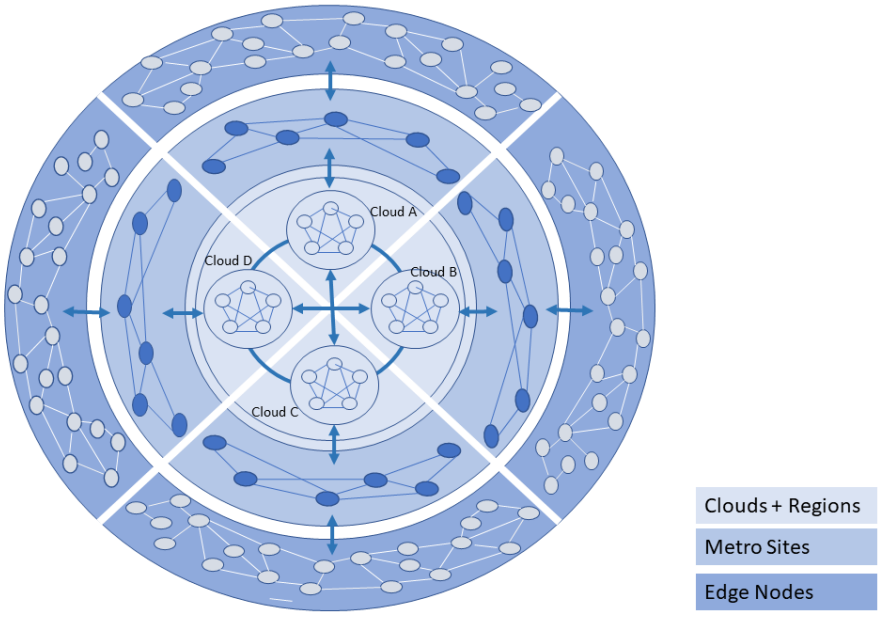
Such an implementation of a Telco cloud allows for a mix and match of price points, flexibility in market positioning and time to market, capacity with the objective of attaining near unlimited capacity, scaling within a sub-cloud or through bursting across sub-clouds, access to local capacity near the user base, and access to specialised services.



Example hybrid multicloud component cloud

8.4 Telco cloud

The Telco Cloud: Data Centre to Edge presents a visualisation of a Telco operator cloud, or Telco cloud, with clouds and cloud components distributed across regional data centres, metro locations, such as a central office or a co-location site, and at the Edge, that are interconnected using a partial mesh network. Note that at the regional centre level the interconnections are likely to form a denser mesh, while at the edges they are likely to form a sparser mesh.



Telco cloud: data centre to edge

The Telco operator may own and/or have partnerships and network connections to utilize multiple clouds for network services, IT workloads, and external subscribers. The types of the component clouds include the following:

357. On-Premise Private

    1. On-Premise Private is open-source, deployed and managed by the operator or the vendor, and based on OpenStack or Kubernetes.
    2. On-Premise Private is developed by the vendor and is deployed and managed by the operator or the vendor. Examples: Azure on Prem, VMware, Packet, Nokia, Ericsson, and so on.

358. On-Premise Public: This commercial cloud service is hosted at the operator location, but is intended for operator and public use. Example: AWS Wavelength.

359. Outsourced Private: With this component cloud, hosting is outsourced. Hosting can be at a commercial cloud service. Examples: Equinix, AWS, and so on.

360. (Outsourced) Public: This is a commercial cloud service. Examples: AWS, Azure, VMware, and so on.

361. Multiple different clouds can be co-located in the same physical location and may share some of the physical infrastructure (for example, racks).

362. Outsourced Private: hosting outsourced. Hosting can be at a commercial cloud service. Examples: Equinix, AWS, and so on.

363. (Outsourced) Public: This is a commercial cloud service. Examples: AWS, Azure, VMware, and so on.

364. Multiple different clouds can be co-located in the same physical location and may share some of the physical infrastructure (for example, racks).

In general, a Telco cloud consists of multiple interconnected large data centres that serve transcontinental areas or regions. A Telco cloud region may connect to multiple regions of another Telco cloud via large-capacity networks. A Telco cloud also consists of interconnected local/metro sites (multiple possible scenarios). A local site cloud may connect to multiple regions within that Telco cloud or in another Telco cloud. A Telco cloud also consists of a large number of interconnected edge nodes. These edge nodes may be impermanent. A Telco cloud's edge node may connect to multiple local sites within that Telco cloud or in another Telco cloud. An edge node may rarely connect to a Telco cloud region.

Table 8-1 provides the essential information about the types of deployments, and responsible parties for cloud artefacts.

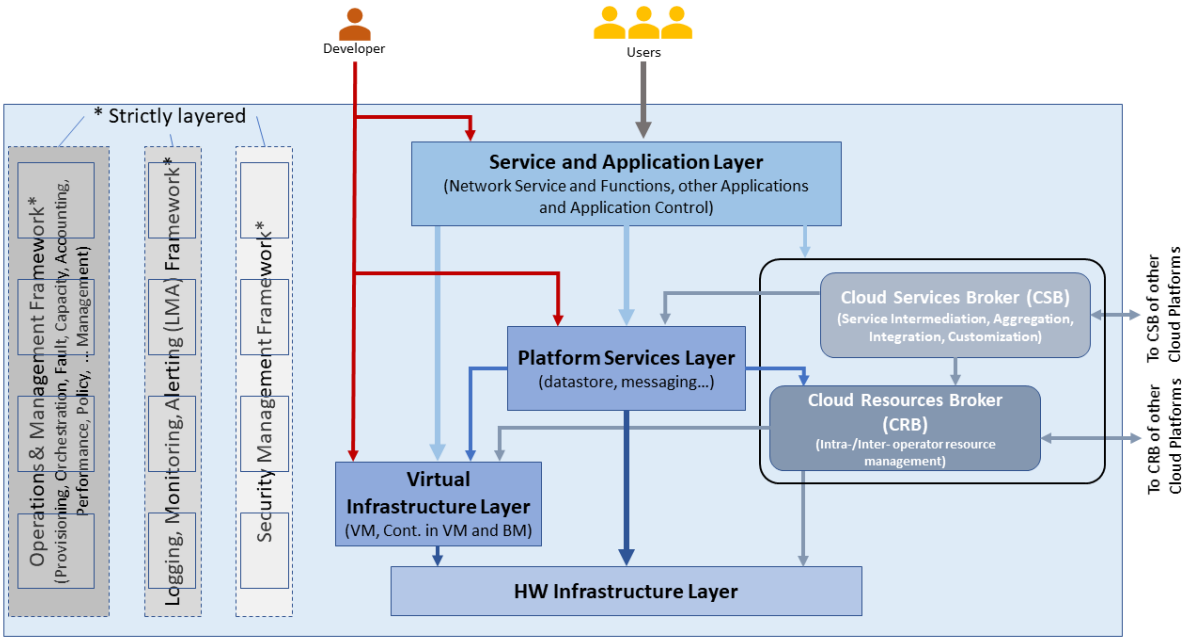
Type	System developer	System maintenance	System operator and manager	Location where deployed	Primary resource consumption models
Private (internal users)	Open-source	Self/vendor	Self/vendor	On-premise	Reserved, dedicated
Private	Vendor, HCP	Self/vendor	Self/vendor	On-premise	Reserved, dedicated
Public	Vendor, HCP	Self/vendor	Self/vendor	On-premise	Reserved, on-demand

Private	HCP	Vendor	Vendor	Vendor locations	Reserved, dedicated
Public (all users)	HCP	Vendor	Vendor	Vendor locations	On-demand, reserved

Table 8-1: Cloud types and the parties responsible for the artefacts

8.4.1 Telco operator platform conceptual architecture

Conceptual Architecture of a Telco Operator Platform shows a conceptual Telco operator platform architecture. The cloud infrastructure resources layer exposes virtualised (including containerised) resources on the physical infrastructure resources and also consists of various types of virtualisation and management software (see details later in this chapter). The cloud platform components layer makes available both elementary and composite objects for use by application and service developers, and for use by services during runtime. The cloud services layer exposes the services and applications that are available to the users. Some of the services and applications may be sourced from or executed on other cloud platforms. Note that while the architecture is shown as a set of layers, this is not an isolation mechanism. Therefore, users may, for example, access the cloud infrastructure resources directly without interacting with a broker.



Conceptual architecture of a Telco operator platform

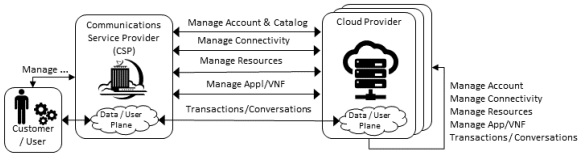
The cloud services and the cloud resources brokers provide value-added services in addition to the fundamental capabilities such as service and resource discovery. These brokers are critical for a multicloud environment to function and utilise cloud-specific plugins to perform the necessary activities. These brokers can, for example, provision and manage environments with resources and services for machine learning (ML) services, augmented/virtual reality, or specific industries.

8.5 Multicloud interactions model

To realise a federated cloud requires the definition of, and agreement on, a set of APIs. These APIs should allow each of the parties to interact cooperatively. They need to cover the

management layer: business management and service operations interactions, as well as the data plane, customer and user, transactions, and conversational interfaces.

As outlined in Conceptual Architecture of a Telco Operator Platform above, the exposure point for the management interactions is the Cloud Service Broker and the Cloud Resource Broker. The set of interactions that these interface points need to provide are defined by the Multicloud interactions model below. Multicloud interactions model provides a taxonomy for the interactions between the communications service provider and the cloud providers.



Multicloud interactions model

The multicloud interactions model defines the following core roles:

365. Communications service provider (CSP): this is the party responsible for providing an end-user service to the customer.

366. Customer/user: these are the parties that use the service (user), and establish the business agreement for the service provision (customer). For retail services, the customer and the user are the same party, while for enterprise services, the enterprise is the customer (responsible for the business agreement), and its representatives are the users.

367. Cloud providers: these are the parties that provide the cloud services. These services could be any XaaS service. A CSP may have an agreement with a SaaS cloud, which in turn uses an IaaS cloud provider to deliver their service.

The set of high-level interactions covers the following:

368. Manage account and catalog: this covers the account, users, subscriptions, billing, and the catalogue of the available services, where the service provider (not necessarily CSP-only) is responsible for the creation and publication of the catalogue contents.

369. Manage connectivity: this covers the public or private network, the VPN configuration, the CSP edge/cloud Connection configuration, and the Connection Security Profile.

370. Manage resource: this covers resource pool management, VM/VNF management (CPU, memory, storage, and network), image repository management, storage management, VNF/CNF LCM, and monitoring of resources.

371. Manage app/VNF: this covers image/container/registry management, deploy/configure/scale/start/stop app/VNF, and monitoring of app/VNFs.

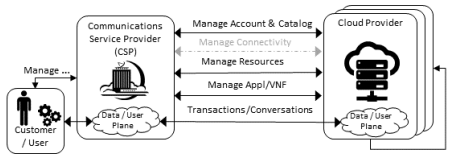
372. Transactions/conversations: this convers the Use Communications Services, Use Edge Applications Services, and Use Cloud Services.

This model, its actors (roles), and the interactions discussed below, are focused on the provision and consumption of the cloud services in different stereotypical deployment scenarios: IaaS, SaaS, CaaS, and Edge. The model presented in Chapter 9 deals with the cloud build and maintenance processes in different scenarios. It also defines the boundaries of the automation domains. These two views complement each other.

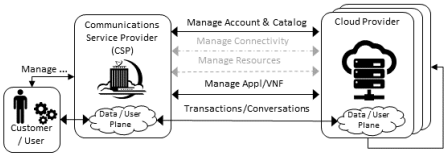
8.5.1 Stereotypical scenarios

A set of stereotypical interactions cases are illustrated for simple Infrastructure-as-a-Service (IaaS) and Software-as-a-Service (SaaS) cases, where deployment is on a cloud provider's centralised sites and/or edge sites. The scenarios help to highlight the needs for the cloud

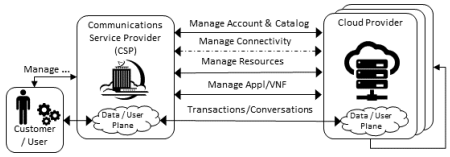
service broker and the cloud resources broker (in accordance with Conceptual Architecture of a Telco Operator Platform), and therefore the extent of the orchestration required to manage the interactions.



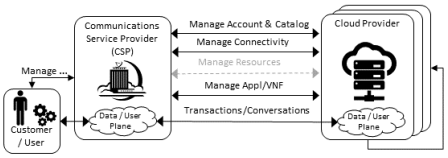
Simple IaaS
Direct Resource Management (CPU, Storage, Network)
Centrally Deployed
Manage Repositories
[Optional: Private Connectivity]



Simple SaaS
Indirect Resource Management
Centrally Deployed
[Optional: Manage Registry]
[Optional: Private Connectivity]



Simple IaaS (@ Edge)
Direct Resource Management (CPU, Storage, Network)
Edge Deployed
Manage Repositories
Wireless Access Edge / Cloud Connectivity Configuration



Simple SaaS (@ Edge)
Indirect Resource Management
Edge Deployed
[Optional: Manage Registry]
Wireless Access Edge / Cloud Connectivity Configuration

Simple stereotypical interactions

The following patterns are visible:

373. For IaaS cloud integration:

    1. The cloud behaves like a set of virtual servers. Therefore, it requires virtual server lifecycle management and orchestration.
    2. Depending on whether the cloud is accessed via the public internet or a private connection changes the extent of the connectivity management.

374. For SaaS cloud integration:

    1. The cloud behaves like a running application/service. It requires subscription management. Complex orchestration of the app/service and underlying resources is managed by the SaaS provider. The user is relieved of having to provide direct control of the resources.

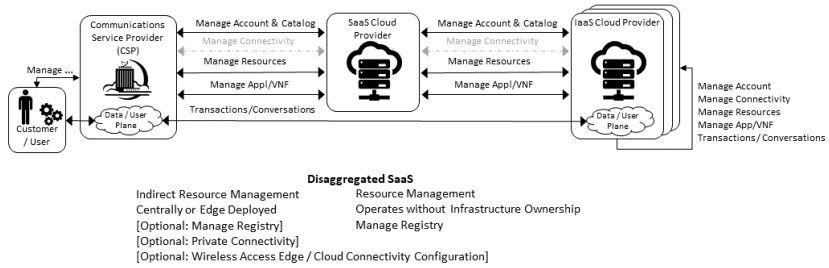
375. For CaaS cloud integration:

    1. The registry for pulling the containers could be from either of the following:
        1. The cloud. In this case, the consumption model is closer to SaaS.
        2. A private or public registry. In this case, the integration model requires specific registry management elements.

376. For edge cloud integration:

    1. This scenario adds the requirement for the communications service provider and the cloud provider to provide physical, network underlay and overlay connectivity management.

A disaggregated scenario for a CSP using SaaS that uses IaaS is illustrated in the following diagram:



Disaggregated SaaS stereotypical interaction

In the disaggregated SaaS scenario, the application provider can operate as an "infra-structureless" organisation. This may be achieved through SaaS organisation using public IaaS Cloud Providers, which could include the CSP itself. A key consideration for CSP in both cloud provision and consumption in multicloud scenarios is how to manage the integration across the cloud providers.

To make this manageable and to avoid integration complexity, there are a number of models. They are as follows:

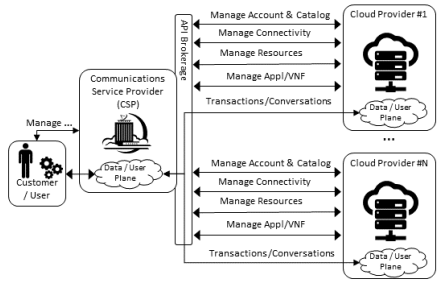
377. Industry-standard APIs that allow consistent consumption across cloud providers.

378. API brokerage, which provides a consistent set of consumer-facing APIs that manage adaption to proprietary APIs.

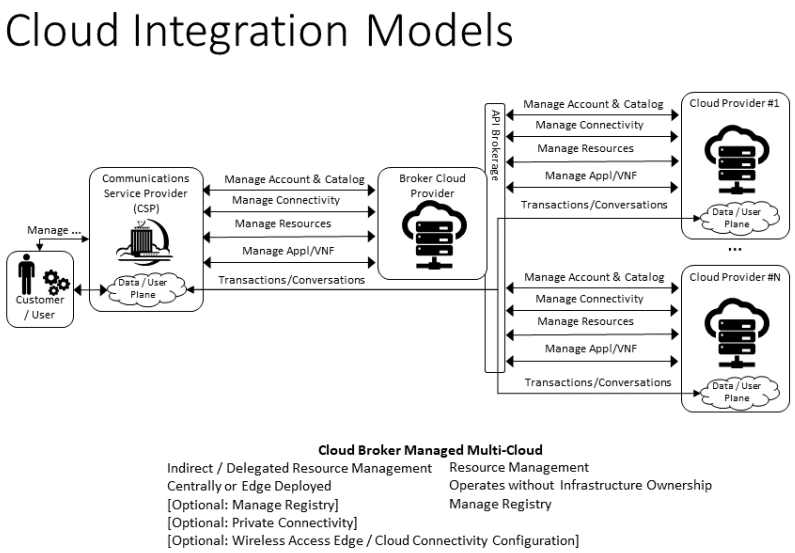
379. Cloud brokerage, where the brokerage function is provided as a service and allows a "single pane of glass" to be presented for the management of the multicloud environment.

The different means of integrating with and managing cloud providers is broadly covered under the umbrella topic of cloud management platforms. A survey of applicable standards for achieving this is provided in section 8.5.2. Requirements, reference architecture and industry standards intersect.

The API and cloud brokerage models are illustrated in the following diagrams:



API brokerage multicloud stereotypical interaction



Cloud brokerage multicloud stereotypical interaction

8.5.2 Requirements, reference architecture and industry standards intersect

The communications service provider (CSP) is both a provider and a consumer of cloud-based services. When the CSP is acting as a consumer, then the typical consideration is the total cost of ownership, as the consumption is usually to support internal business operations and BSS/OSS systems. When the CSP is acting as a provider of cloud services, through the operation of their own cloud or the reselling of the cloud services, then the typical consideration is marginal (cost to offer services versus income received).

These two stances will drive differing approaches to how a CSP would look, in order to manage the way it interacts within a multicloud environment.

As a consumer of cloud services to support internal business operations and BSS/OSS, the focus is on meeting the needs of the organisation's applications . Historically, this came with the need to operate and support the organisation's infrastructure needs. This resulted in the splitting of the CIO organisation into Delivery and Operations groups. At the same time that the CIO application workloads are moving to SaaS and other cloud providers, the CTO network systems are migrating from running on custom-dedicated infrastructure to run on virtualised COTS infrastructure. Examples of this include IMS and 3GPP (4G and 5G) functions. IP routers and firewalls are being provided as VNFs and CNFs. These network workloads are now also being deployed in private CSP clouds, as well as in public clouds.

As outlined in section "8.4 Telco Cloud", the result is that the CSP network is now an interconnected set of distributed cloud infrastructures supported by different cloud providers, including the CSP. Therefore, the term Hybrid Multicloud, and the need for the CSP to be able to support and use this interconnected cloud, are both inevitable and essential.

As a consumer and provider of cloud services, the necessity for the CSP to build and manage its own cloud infrastructure will continue. The CSP will also have to provide the following:

380. Cloud orchestration solutions, to orchestrate the use of cloud services and capabilities from its own and other cloud providers.

381. Network orchestration solutions, to manage the interconnectivity across its own and other cloud provider networks.

The interactions for this are outlined in the Multicloud Interactions Model. However, to realise this, the CSP will need to adopt and sponsor a set of standards that are necessary to support these interactions. The identification of existing applicable standards and gaps across the interactions needs to be completed. As a first step, the following criteria for the inclusion of a standard/technology is defined. The following must be true of these standards and technologies:

382. They must provide the capabilities that are necessary to achieve a hybrid multicloud vision and multicloud interactions.
383. They must be already mature Open Standards that have either been adopted or nurtured by recognised bodies with the telecommunications industry (for example, ITU, ETSI, TMForum, GSMA, 3GPP, ISO, and national standards organisations, such as ANSI, NIST, and so on).
384. They must have reference implementations, or active open source projects or consortia providing implementations (for example, the Cloud Native Computing Foundation (CNCF) and the Open Infrastructure Foundation).
385. They must allow the CSP to source delivery and support services based on these from multiple vendors.
386. They must allow the CSP to actively contribute to and request the capabilities and coverage of the standard or technology.
387. They must not be the sole property of a vendor or company.
388. They must not be focused on transactions or conversations, or user or data plane standards (typically IETF, IEEE, MEF/Carrier Ethernet, and so on).

8.5.3 Hybrid, Edge, and Multicloud unified management Platform

As organisations spread their resources across on-premises, multiple clouds, and the edge, there is a clear need for a single set of tools and processes to manage and operate across these hybrid, edge, and multiclouds (HEM clouds), as can be seen from the following simplified scenarios.

Scenario: An operator has private clouds that they use for their workloads. Over time, the operator evolves the environment of these private clouds:

389. A: The operator has decided to use one or more public clouds for some of their workloads.
390. B: The operator has decided to use an edge cloud for some of their clients.
391. C: The operator has decided to create edge clouds for some of their clients.

Scenario B can be treated as the same as Scenario A. Scenario C is akin to the private cloud, except for the location and control over the facilities at that location. For its workloads, the operator will have to use the target clouds tools or APIs to create the necessary accounts, billing arrangements, quotas, and so on. The operator then creates the necessary resources, such as VMs or Kubernetes clusters, and so on. The operator follows up with creating needed storage, networking, and so on, before onboarding the workload and operating it. This is a complex task, even when the operator is dealing with only one other cloud, in addition to operating their own cloud. The operator is faced with a number of challenges, including acquiring a new set of skills, knowledge of APIs, tools, and the complexity of managing different policies, updates, and so on. This becomes impossible to manage when incorporating more than one other cloud. Hence the need for a single pane of glass.

This Hybrid, Edge, and Multicloud unified management Platform (HEMP), also known as single pane of glass, provides the capabilities to consistently perform the following set of tasks through a common set of governance and operational practices:

392. Manage accounts, credentials, resources, and services across facilities (regions, data centres, and edge locations).

393. Interoperate the different clouds.

394. Implement common policies and governance standards.

395. Manage a common security posture.

396. Provide an integrated visualisation into the infrastructure and workloads.

GSMA's Operator Platform Group (OPG) specifies a federated model and the requirements for the edge platforms (Operator Platform Telco Edge Requirements v2.0 [35]) While the document is for edge, most of the requirements can be applied to other cloud deployments. Anuket RM is implementation agnostic, that is, whether the implementation uses agents, federations, or other mechanisms.

The following tables list some of the requirements for the Hybrid, Edge, and Multicloud operator Platform (HEMP). These requirements are in addition to the requirements in other chapters of this RM.

HEMP general requirements

Table 14General requirements of the Hybrid, Edge, and Multicloud operator Platform (HEMP)

Ref	Requirement	Definition/Note
hem.gen.001	The HEMP should only use published APIs in managing component clouds.	For example, to accomplish the example in hem.gen.003, it uses the published APIs of the target cloud.
hem.gen.002	The HEMP should publish all of the APIs used by any of its components.	For example, the available GUI portal only uses APIs that have been published by the HEMP.
hem.gen.003	The HEMP should provide common terms for interaction with its constituent clouds.	For example, “create Account” should be used across the different clouds.
hem.gen.004	The HEMP should generalise and define a common set of resources available to be managed in the constituent clouds.	Example resources include hosts (including BareMetal), virtual machines (VMs), vCPU, memory, storage, network, Kubernetes clusters, kubernetes nodes, images (OS and others), and credentials. For the private cloud, additional example resources include racks, ToR/CE switches, and platform images.
hem.gen.005	The HEMP should provide a common interface for managing component clouds.	
hem.gen.006	The HEMP should expose resources from all the cloud operators and locations (regions, sites, and so on).	See the example resources in hem.gen.004

hem.gen.007	The HEMP should allow reservation of resources, if the component cloud operator allows it.	
hem.gen.008	The HEMP should support multitenancy.	

Table 8-2: General requirements of the Hybrid, Edge, and Multicloud operator Platform (HEMP)

Requirements **of HEMP operations**

Ref	Requirement	Definition/Note
hem.ops.001	The HEMP should generalise and define a common set of management operations available in the constituent clouds. Required operations include: create, deploy, configure, start, suspend, stop, resume, reboot, delete, scale, and list. Some operations may only be available for a subset of resources.	
hem.ops.002	The HEMP should centrally manage all resources (across all constituent clouds).	
hem.ops.003	The HEMP should centrally operate all constituent clouds.	
hem.ops.004	The HEMP should provide mechanisms to integrate new clouds.	This may require pre-development of necessary capabilities for the support of HEMP abstractions, and implementation of connectivity with the new cloud
hem.ops.005	The HEMP should provide mechanisms to drop a constituent cloud.	For example, the provided GUI portal shall only use HEMP published APIs
hem.ops.006	The HEMP should provide mechanisms and processes for onboarding existing assets (such as resources, connectivity, and so on).	
hem.ops.007	The HEMP should provide mechanisms and processes for the automated configuration management of all environments and resources.	

Table 8-3: Operability requirements of the Hybrid, Edge, and Multicloud operator Platform (HEMP)

HEMP LCM requirements

Ref	Requirement	Definition/Note
hem.lcm.001	The HEMP should monitor all environments and assets.	

hem.lcm.002	The HEMP should provide visibility into the health of all assets.	
hem.lcm.003	The HEMP should provide capabilities for centralised visibility and management of all alerts.	
hem.lcm.004	The HEMP should provide capabilities for the centralised analysis of all logs.	This does not preclude local log analytics.

Table 8-4: Lifecycle Management (LCM) requirements of the Hybrid, Edge, and Multicloud operator Platform (HEMP)

HEMP security requirements

Table 15HEMP security requirements

hem.sec.001	Requirement: The HEMP should provide capabilities for the centralised management of all security policies.	Definition/Note: (empty)
hem.sec.002	Requirement: The HEMP should provide capabilities for the centralised tracking of compliance of all security requirements (Consolidated security requirements).	Definition/Note: (empty)
hem.sec.003	Requirement: The HEMP should provide capabilities for insights into the changes that resulted from resource non-compliance.	Definition/Note: (empty)

Table 8-5: Hybrid, Edge, and Multicloud operator Platform (HEMP) security requirements

8.5.4 Aspects of multicloud security

Cloud infrastructures, emerging as a key element in the Telco operator ecosystem, are part of the attack surface landscape. This is particularly worrying with the 5G rollout becoming a critical business necessity. It is important to be vigilant regarding the cloud-focused threats and associated adversarial behaviours, methods, tools, and strategies that cyber threat actors use. In the multicloud ecosystem, composed of different security postures and policies, network domains, products, and business partnerships, the responsibility for managing the different cloud environments necessary to support 5G use cases falls to different enterprises, creating new levels of complexities and a new range of security risks.

For services deployed on hybrid multicloud environments, the security responsibility can be delegated to cloud service providers. However, the Telco operator is always accountable for their customers' data protection (at rest, in transit, and in use) and for the security posture of the deployments. It implies that a consistent security posture is ensured across multiple cloud service providers. The white paper "Evolving 5G security for the cloud", 5G Americas, September 2022, addresses this issue. A Mobile Network Operator (MNO) deploying 5G networks in a hybrid multicloud environment is a cloud consumer and is accountable for the security of all the layers of the cloud stack. The white paper details the cloud shared security model in the three cloud service models: IaaS, PaaS, and SaaS. The MNO must ensure the cloud service agreement articulation of the security responsibilities. The white paper also highlights the importance of applying a zero-trust mindset for cloud-based deployment, for RAN and core functions to secure the networks.

In a hybrid multicloud environment, there are additional security principles that need to be considered. These principles, set out in the table below, are drawn from the collaboration with the GSMA Fraud and Security Group (FASG) and the "5G security Guide", FS.40 v2.0 document [37].

Multicloud security principle	Description
Policy synchronization	Consistency in applying the right security policies across environments, services, interfaces, and configured resources.
Visibility	A common data model approach to share events and behaviours across all the key compute, storage, network, and applications resources; environments, virtualised platforms, containers, and interfaces.
Monitoring	Centralisation, correlation, and visualisation of security information across the different cloud environments, to provide an end-to-end view and enable timely response to attacks.
Automation	Automation of critical activities, including cloud security posture management, continuous security assessments, compliance monitoring, detection of misconfigurations, and identification and remediation of risks.
Access management	A wide range of users, including administrators, testers, DevOps, and developers and customers, should be organised into security groups with privileges appropriate to the different resources and environments.
Security operations model	Augmentation of security services provided by cloud service providers, with the vetted third-party and/or open-source tools and services, all incorporated into the established overall security operations model.

Table 8-6: Multicloud security principles

For Telco operators to run their network functions in a multicloud environment, specifically, in public clouds, the industry will need a set of new standards and new security tools to manage and regulate the interactions between the parties participating in the multicloud. For an example of a step in this direction, see ETSI specification TS 103 457 [143] “Interface to offload sensitive functions to a trusted domain”. This document provides extra security requirements for public clouds, to allow Telco operators the option of running network functions in public clouds.

There is another security aspect to consider, which is related to the autonomous nature of the participants in the multicloud. We can prescribe certain things and if not satisfied treat that party as "untrusted". This problem has been addressed to some extent in TS 103 457. This standard introduces the idea of a Less Trusted Domain (LTD) and a More Trusted Domain (MTD), and specifies the Trusted Cross-Domain Interface (TCDI) to standardise secure interactions between them. The standard defines the following elementary functions of the TCDI:

397. Connection and session management

398. Data and value management

399. Transferring cryptography functionality. This comprises the following:

400. Entropy request

401. Encryption keys request

402. Trusted timestamping

403. Secure archive

404. Secure storage

405. Search capabilities

As described in section 1 Scope of the TS 103 457 document [38], it specifies "a high-level service-oriented interface, as an application layer with a set of mandatory functions, to access secured services provided by, and executed in a More Trusted Domain. The transport layer is out of scope and left to the architecture implementation". The standard provides extra security features for sensitive functions, down to individual virtual machines or containers. As such, it is recommended that the relevant components of the reference models, reference architecture, reference implementations, and reference compliance take note of this standard and ensure their compatibility, wherever possible.

8.6 Telco Edge Cloud (TEC)

This section presents the characteristics and capabilities of different edge cloud deployment locations, infrastructure, footprint, and so on. Note that many terms are used in this section. For this reason, this section includes a table that tries to map these different terms.

8.6.1 Telco Edge Cloud: deployment environment characteristics

Telco Edge Cloud (TEC) deployment locations can be in any of the following environments:

406. Friendly environments, such as offices, apartments, or other similar indoor locations.
407. Challenging environments, such as busy streets, near to network radio transmitters, or other noisy outdoor locations.
408. Harsh environments: places where there is a likelihood of chemical, heat, or electromagnetic exposure, such as factories, power stations, processing plants, and so on.

Some of the more salient characteristics can be seen in Table 8-7.

Table 16TEC deployment location characteristics and capabilities

Environmental type	Facility type	Environmental characteristics	Capabilities	Physical security	Implications	Deployment locations
Environmentally friendly	Indoors: typically commercial or residential buildings.	Protected, and therefore safe for common infrastructure.	409. Easy access to a continuous electricity supply. 410. High/medium bandwidth. 411. Fixed and/or wireless network	Controlled access	Commoditised infrastructure with minimal need or no need for hardening or ruggedisation. Operational benefits for installation and maintenance.	Indoor venues: homes, shops, offices, stationary and secure cabinets, data centres, central offices, colocation facilities, vendor premises, custom

			access.			er premises.
Environmentally challenging	Outdoors and/or exposed to environmentally harsh conditions.	412. Lack of protection. 413. Exposure to abnormally high levels of noise, vibration, heat, chemical, and electromagnetic pollution.	414. Possibility of devices having to rely on battery power only. 415. Low/medium bandwidth. 416. Fixed and/or mobile network access.	Little or no access control.	417. Ruggedisation is likely to be expensive. 418. The system is likely to be complex to operate.	Example locations: curb side, near cellular radios.

Table 8-7: TEC deployment location characteristics and capabilities

8.6.2 Telco Edge Cloud: infrastructure characteristics

Commodity hardware is only suited for environmentally friendly environments. Commodity hardware has standardised designs and form factors. Cloud deployments in data centres typically use such commodity hardware with standardised configurations. This results in operational benefits for procurement, installation, and ongoing operations.

In addition to the type of infrastructure hosted in the data centre clouds, facilities with smaller infrastructure deployments, such as central offices or colocation facilities, may also host non-standard hardware designs, including specialised components. The introduction of specialised hardware and custom configurations increases the cloud operations and management complexity.

At the edge, the infrastructure may further include ruggedised hardware for harsh environments and hardware with different form factors. With the evolution of the Internet of Things (IoT) and ubiquitous connectivity (including personal devices) to consider extreme-edge devices as part of the ecosystem, this will require the infrastructure to integrate with and offer programmability and processing capabilities for these devices.

The end-to-end orchestration will need to support the extreme edge use cases.

8.6.3 Telco Edge Cloud: infrastructure profiles

The Profiles and workload flavours section specifies the following two infrastructure profiles:

419. The **Basic** cloud infrastructure profile. This is intended for use by both IT and Network Function workloads that have low to medium network throughput requirements.
420. The **High Performance** cloud infrastructure profile. This is intended for use by applications that have high network throughput requirements (up to 50 Gbps).

The High-Performance profile can specify extensions for hardware offloading. For details, see Hardware Acceleration Abstraction. The Reference Model High-Performance profile includes an initial set of Profile extensions.

Based on the infrastructure deployed at the edge, Table 8-8 specifies the Feature Set and Requirements from Infrastructure that would need to be relaxed.

Table 17TEC exceptions to infrastructure profile features and requirements

Reference	Feature	Description	As specified in RM Chapter 05 - Basic type	As specified in RM Chapter 05 - High performance	Exception for edge - Basic type	Exception for edge - High performance
infra.stg.cfg.003	Storage with replication		N	Y	N	Optional
infra.stg.cfg.004	Storage with encryption		Y	Y	N	Optional
infra.hw.cpu.cfg.001	Minimum number of CPU sockets	This determines the minimum number of CPU sockets within each host.	2	2	1	1
infra.hw.cpu.cfg.002	Minimum Number of cores per CPU	This determines the minimum number of cores needed per CPU.	20	20	1	1
infra.hw.cpu.cfg.003	NUMA alignment	NUMA alignment support and BIOS configured to enable NUMA.	N	Y	N	Y (*)

Table 8-8: TEC exceptions to infrastructure profile features and requirements

421. This is immaterial if the number of CPU sockets (infra.hw.cpu.cfg.001) is 1.

Note

None of the listed parameters forms part of a typical OpenStack flavour, except that the vCPU and memory requirements of a flavour cannot exceed the available hardware capacity.

8.6.4 Telco Edge Cloud: platform services deployment

This section characterises the hardware capabilities for different edge deployments and the platform services that run on the infrastructure.

Note

The platform services are containerised to save resources, and benefit from intrinsic availability and autoscaling capabilities.

Platform services are:

422. Identity
423. Image
424. Placement
425. Compute
426. Networking
427. Message Queue
428. DB Server

Storage services are:

429. Ephemeral
430. Persistent Block
431. Persistent Object

Network services are:

432. Management
433. Underlay (Provider)
434. Overlay

Table 18Characteristics of infrastructure nodes

Node type	Identity	Image	Placement	Compute	Networking	Message Queue	DB Server	Ephemeral	Persistent Block	Persistent Object	Management	Underlay (Provider)	Overlay
Control nodes	✅	✅	✅	✅	✅	✅	✅		✅		✅	✅	✅
Workload nodes				✅	✅			✅	✅	✅	✅	✅	✅

(Compute)													
Storage nodes									✅	✅	✅	✅	✅

Table 8-9: Characteristics of infrastructure nodes

Depending on the facility capabilities, deployments at the edge may be similar to one of the following:

435. Small footprint edge device.

436. Single server: deploying multiple (one or more) workloads.

437. Single server: single controller and multiple (one or more) workloads.

438. HA at the edge (at least two edge servers): multiple controllers and multiple workloads.

8.6.5 Comparison of deployment topologies and edge terms

Table 19Profile extensions

This Specification	Compute	Storage	Networking	RTT	Security	Scalability	Elasticity	Resiliency	Preferred Workload Architecture	Upgrades	OpenStack	OPNFV Edge	Edge Glossary
Regional Data Centre (DC), Fixed	1000's, Standardised, >1 CPU >20 cores / CPU	10's EB, Standardised, HDD and NVMe, Permanence	>100 Gbps, Standardised	~100 ms	Highly Secure	Horizontal and unlimited scaling	Rapid spin up and down	Infrastructure architected for resiliency, Redundancy for FT and HA	Microservices based, Stateless, Hosted on Containers	Firmware: When required, Platform SW: CD	Central Data Center		
Metro Data	10's to 100's, Stand	100's PB, Stan	> 100 Gbps,	~10 ms	Highly Se	Horizontal but	Rapid spin up	Infrastructure	Microservices	Firmware:	Edge Site	Large	Aggregation

Centres, Fixed	ardised, >1 CPU >20 cores / CPU	dardised, NVMe on PCIe, Permanence	Standardised		cure	limited scaling	and down	architected for some level of resiliency, Redundancy for limited FT and HA	based, Stateless, Hosted on Containers	When required, Platform SW: CD		Edge	Edge
Edge, Fixed / Mobile	10's, Some Variability, >=1 CPU, >10 cores / CPU	100 TB, Standardised, NVMe on PCIe, Permanence / Ephemeral	50 Gbps, Standardised	~5 ms	Low Level of Trust	Horizontal but highly constrained scaling, if any	Rapid spin up (when possible) and down	Applications designed for resiliency against infra failures No or highly limited redundancy	Microservices based, Stateless, Hosted on Containers	Firmware: When required, Platform SW: CD	Far Edge Site	Medium Edge	Access Edge / Aggregation Edge
Mini / Micro Edge, Mobil / Fixed	1's, High Variability, Harsh Environments, 1 CPU >2 cores / CPU	10's GB, NVMe, Ephemeral, Caching	10 Gbps, Connectivity not Guaranteed	<2 ms Located in network proximit	Untrusted	Limited Vertical Scaling (resizing)	Constrained	Applications designed for resiliency against infra failu	Microservices based or monolithic, Stateless or	Platform	Fog Computing (Mostly deprecated terminology)	Small Edge	Access Edge

		y of EUD / IoT		res No or highly limited redundancy	Stateful, Hosted on Containers or VMs, Subject to QoS, adaptive to resource availability, viz. reduce resource consumption as they saturate	, Extreme Edge, Far Edge

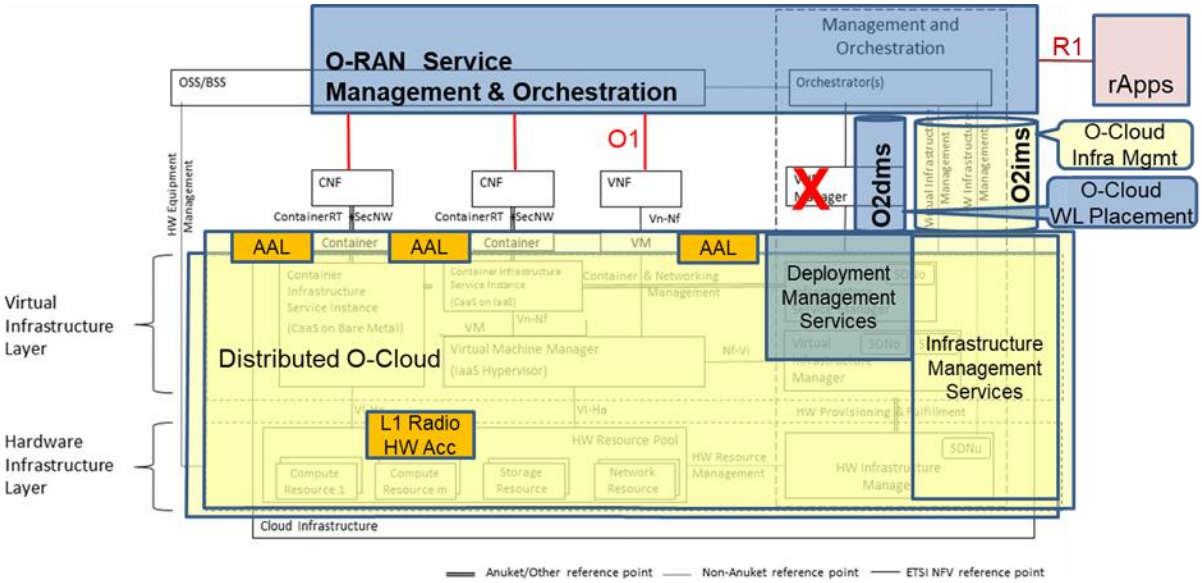
Table 8-10: Comparison of Deployment Topologies

8.6.6 O-RAN alignment and interaction

O-RAN is an operator led alliance group with members from the major Telco Operators, Vendors, and other interested ecosystem participants around the specification of Open Radio Access Networks. Its task is to cloudify the 3GPP specified RAN Network Functionalities with multi-vendor open interfaces in between the Network Functions, the Cloud Infrastructure, and the management. The Service Management and Orchestration (SMO) of multiple O-Clouds is also specified including a framework for 3rd party applications (rApps). There are also a few other open interfaces that are aimed to be specified e.g., for Radio Layer 1 HW Accelerators and some low-level Radio functions.

The O-RAN architecture is built up by a set of individual O-Clouds that provide the execution platforms for the Cloudified RAN Network Functions in a similar way as the Anuket NFVI infrastructure, although with O-RAN specified management interfaces. Each O-Cloud can be distributed on a set of Cloud Sites where they can provision VM and Container Node Clusters. The provisioning of O-Clouds and their resources are managed and orchestrated from a centralized RAN Service Management and Orchestration framework (SMO) over the O-RAN specified O2 interface like any other Telco Operations Support Systems (OSS).

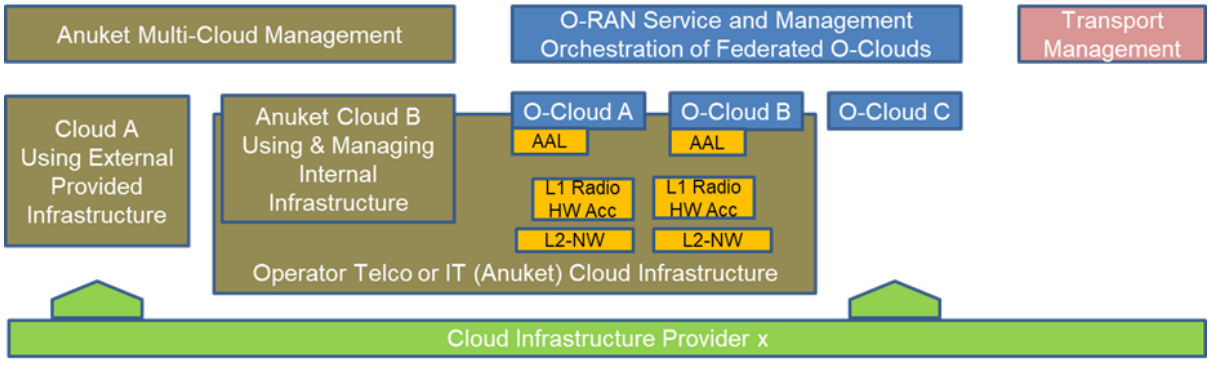
On a high-level O-RAN covers similar specification grounds as what Anuket do, but there are some noteworthy differences both on specification level and on the aim for how O-Clouds are realized. O-RAN specifies how management and orchestration of the Network Functions and Cloud Infrastructure shall be done with a set of internal Services that also have a set of interface specifications for how the rApps could enhance the management functionality. O-RAN have also articulated that O-Clouds can be distributed over multiple Cloud Sites that are stitched over an externally specified WAN interconnect transport that is not part of the O-RAN.



O-RAN architecture mapped onto Anuket Reference Model

O-Clouds are in some ways similar to the Anuket Cloud Infrastructure with the notable differences that they have an O-RAN specified interface of how the O-Cloud infrastructure is managed (O2ims) and how workloads (e.g., whole or parts of Network Functions) are deployed on the O-Cloud clusters (O2dms). On a more detailed level the O-Clouds are internally very Layer2 (Ethernet) centric, today with strict requirements of determinism and low latency for Cloud Site internal connectivity in between the Network Functions. The O-Clouds also have the set of O-RAN specified HW Accelerators and an Acceleration Adaptation Layer (AAL) of how they are used from the Network Functions for their Radio-near functions.

A potential alignment between Anuket and O-RAN's O-Cloud specifications can be investigated. This would require an analysis how an Anuket Reference Architecture based on open-source technology can support the O-RAN HW Accelerators (as stated in RM Ch3 section Example of O-RAN AAL Interface Example of O-RAN Acceleration Abstraction Layer Interface) and a Layer2-centric networking infrastructure. It would enable the operators to have an internal Telco Cloud that supports both Core and RAN Network Functions, and in the extension possibly also other workloads in a shared Cloud that supports required Telco features and characteristics.



Anuket as potential under-cloud to O-Clouds in O-RAN

9 Infrastructure Operations and Lifecycle Management

9.1 Introduction to Infrastructure Operations and Lifecycle Management

The purpose of this section is to define the capabilities required of the infrastructure to ensure that it is effectively supported, maintained, and otherwise lifecycle-managed by the operations teams. This includes requirements relating to the need to be able to maintain infrastructure services "in-service" without impacting the applications and workloads, while minimising human labour. It must also capture any exceptions and related assumptions.

There are three main business operating frameworks that are commonly known and used across the telecommunications industry and that are related to the topics in this section:

439. FCAPS: the ISO model for network management.

440. The TM Forum Business Process Framework (eTOM).

441. ITIL: ITIL 4.0 attempts to adapt IT service management practices to the needs of the cloud environment.

The sections below roughly map to these frameworks as follows:

Chapter title	FCAPS	eTOM	ITIL
Configuration and lifecycle management	Configuration	Fulfilment	Configuration, release, change
Assurance	Performance, fault	Assurance	Event, incident
Capacity management	Configuration	Fulfilment	capacity management

Table 9-1: Operating Frameworks

Note: The above mapping is provided for general orientation purposes only. Detailed mapping of the required Cloud Infrastructure Lifecycle Management capabilities to any of these frameworks is beyond the scope of this document.

9.2 Infrastructure lifecycle model

The model discussed in this chapter is focused on the cloud infrastructure lifecycle. It provides a complementary view to the cloud consumption model presented in Chapter 8.

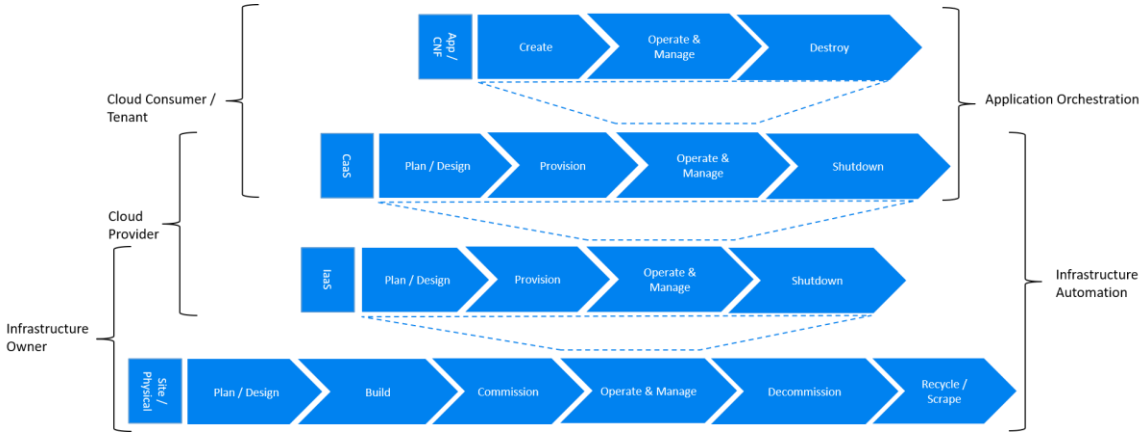
The following diagrams provide mapping between the different stages of the infrastructure lifecycle across all the layers of the stack, to the owners of the infrastructure and the cloud, and the tenant as the consumer of the cloud services, in three different scenarios:

442. Applications running as containers within virtual machines (the CaaS on IaaS scenario).

443. Applications running as containers on bare metal (the CaaS on BM scenario).

444. Applications running as VNFs within virtual machines (the IaaS scenario).

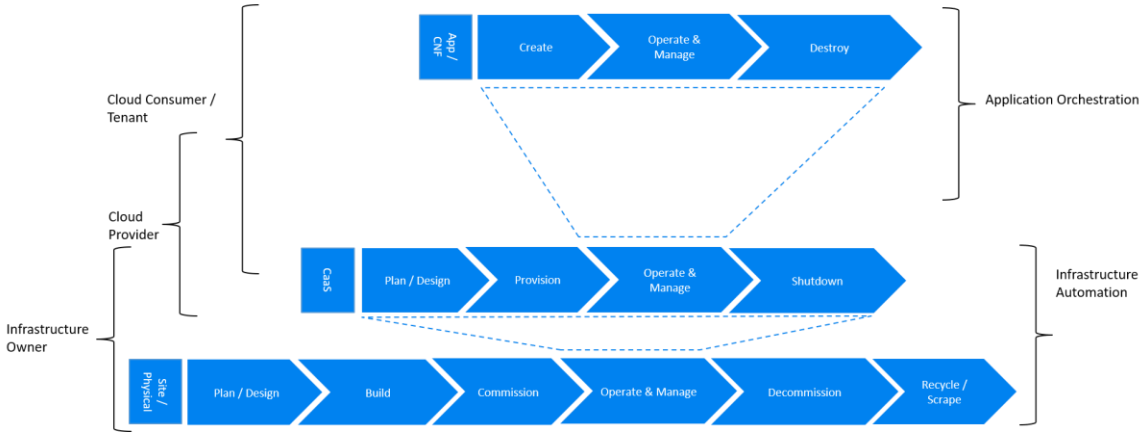
The diagrams also define also the scope of the Infrastructure LCM Automation for each of these scenarios. The dotted lines symbolise the interactions between the layers of each of the models.



Infrastructure LCM in CaaS on IaaS scenario

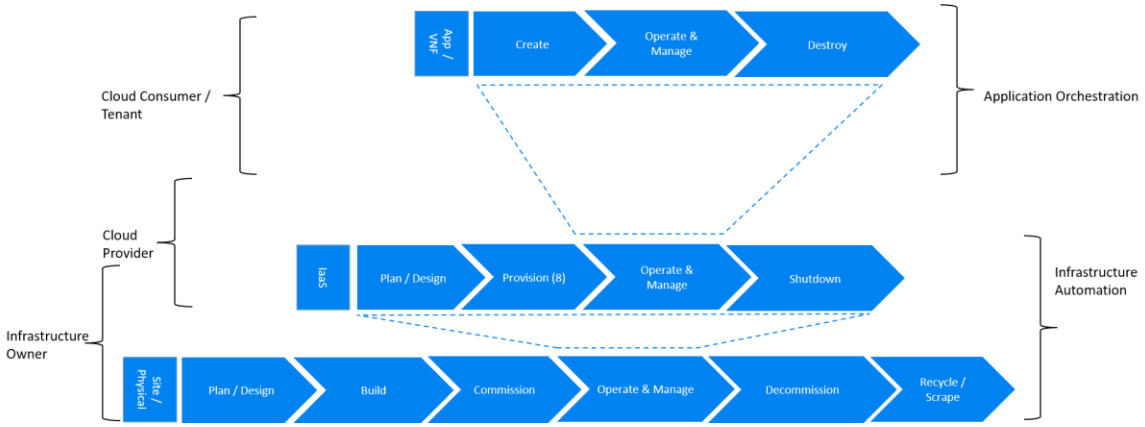
In the CaaS on IaaS scenario, the Infrastructure LCM scope covers the site/physical layer, the IaaS layer, and the CaaS layer. From the lifecycle perspective (on the left-hand side of the diagram), the site/physical layer is entirely owned by the infrastructure owner. The virtualised infrastructure layer (IaaS) is shared between the infrastructure owner and the cloud provider. Similarly, the container orchestration layer (CaaS) is shared between the cloud provider and the cloud consumer/tenant. These relationships can be illustrated by a scenario in which a telecom operator owns the physical infrastructure on which an external cloud provider runs the virtualisation software (hypervisor).

Sharing the CaaS layer between the cloud provider and the cloud consumer reflects the fact that the container management/orchestration software, like Kubernetes, is life cycled by the cloud provider (for example, when scaling out containers), but also by the cloud consumer, due to the close lifecycle relationship between an application and a container in this model. For example, destroying an application means also destroying related containers. Therefore, CaaS can be also considered as a part of the Application Orchestration layer.



Infrastructure LCM in the CaaS on BM scenario

The main difference in the CaaS on BM scenario is the absence of the IaaS layer. Therefore, the scope of the Infrastructure LCM and its automation is limited to only two layers: the site/physical layer and the CaaS layer. From the lifecycle ownership perspective, the CaaS layer is now shared not only between the cloud provider and the cloud consumer (for the same reasons as in the CaaS on IaaS scenario), but also with the infrastructure owner. The latter observation is related to the fact that in the bare-metal deployments lacking the hypervisor separation, the CaaS layer is much more dependent on the underlying physical infrastructure.

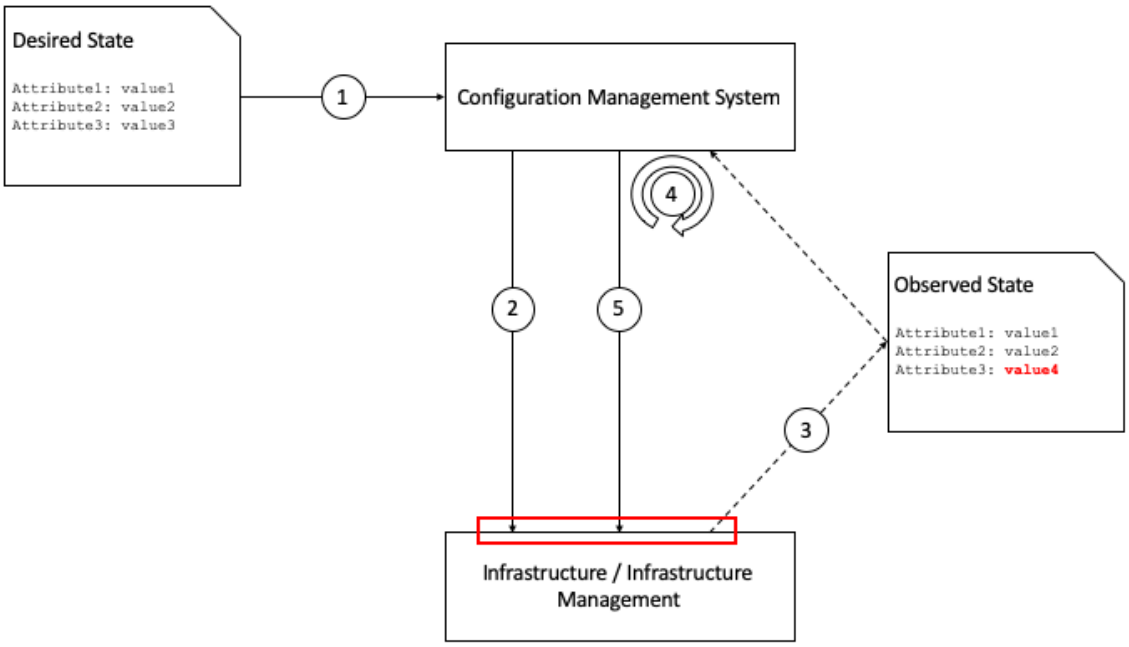


Infrastructure LCM in IaaS scenario

In this "classical" scenario, the scope of the Infrastructure LCM and its automation is defined by the site/physical layer and the IaaS layer. From the lifecycle perspective, the ownership of IaaS is shared between the infrastructure owner and the cloud provider. This scenario is characterised by a clear separation between the lifecycle (and hence its automation) of the infrastructure, and the application lifecycle owned by the cloud consumer/tenant in the role of the application owner.

9.3 Configuration and lifecycle management

Configuration management is concerned with defining the configuration of infrastructure and its components, and tracking or observing the running configuration of that infrastructure and any changes that take place. Modern configuration management practices, such as desired state configuration management, also mean that any changes from the desired state that are observed (also known as the delta) are rectified by an orchestration/fulfilment component of the configuration management system. This closed loop mitigates the configuration drift in the infrastructure and its components. Our recommendation is to keep these closed loops as small as possible to reduce complexity and the risk of error. Configuration and Lifecycle Management shows the configuration management loop and how it relates to lifecycle management.



Configuration and lifecycle management

The initial desired state might be for 10 hosts with a particular set of configuration attributes, including the version of the hypervisor and any management agents. The configuration management system takes that as input (1) and configures the infrastructure as required (2). It then observes the current state periodically (3) and, in case of a difference between the desired state and the observed state, it calculates the delta (4) and reconfigures the infrastructure (5). This loop takes place for each lifecycle stage (create, update, and delete). For example, if an update to the hypervisor version is defined in the desired state, the configuration management system calculates the delta (for example, v1 --> v2) and reconfigures the infrastructure as required.

However, the key requirements for the infrastructure and for infrastructure management are those interfaces and reference points in the red box, where the configuration is **set**, and where it is **observed**. Table 9-2 lists the main components and capabilities required to manage the configuration and lifecycle of those components.

Table 20Profile extensions

Component	Set/Observe	Capability	Example
Cloud infrastructure management software	Set	Target software/firmware version	Software: v1.2.1
	Observe	Observed software/firmware version	Software: v1.2.1
Cloud infrastructure software	Set	Target software version	Hypervisor software: v3.4.1
	Observe	Observed software/firmware version	Hypervisor software: v3.4.1
Infrastructure hardware	Set	Target software/firmware version	Storage controller firmware: v10.3.4
	Observe	Observed software/firmware version	Storage controller firmware: v10.3.4

Table 9-2: Configuration and lifecycle management capabilities

Table 9-3 below defines the standard interfaces that should be made available by the infrastructure and cloud infrastructure management components to allow for successful Configuration Management.

Component	Interface standard	Link
Infrastructure management	Defined in the RA specifications	RA-1, RA-2
Infrastructure software	Defined in the RA specifications	RA-1, RA-2
Infrastructure hardware	Redfish API	DMTF RedFish specification [16]

Table 9-3: Interface standards for configuration management

9.4 Capacity management

Capacity management is a potentially wide-ranging process that includes taking demands across lines of business, analysing data about the infrastructure that is running, and calculating when additional infrastructure might be required, or when infrastructure might need to be decommissioned.

As such, the requirements for capacity management on the infrastructure are covered by the Assurance, and the Configuration and Lifecycle Management sections below. The Assurance section deals with the collection of data. There is no reason to consider that this would be done by a different mechanism for capacity management, as it is for assurance. The Configuration and Lifecycle Management section deals with the changes being made to the infrastructure hardware, software, and management components (for example, changing of the number of hypervisor hosts from 10 to 12).

9.5 Assurance

Assurance is concerned with the following:

445. The proactive and reactive maintenance activities that are required to ensure that the infrastructure services are available, in accordance with the defined performance and availability levels.

446. The continuous monitoring of the status and performance of the individual components, and of the service as a whole.

447. The collection and analysis of performance data, which is used to identify potential issues, including the ability to resolve the issue with no customer impact.

The requirement types are as follows:

- Data collection from all the components, such as:
    1. The ability to collect data relating to events (transactions, security events, physical interface up/down events, warning events, error events, and so on).
    2. The ability to collect data relating to component status (up/down, physical temperature, disk speed, and so on).
    3. The ability to collect data relating to component performance (used CPU resources, storage throughput, network bandwidth in/out, API transactions, transaction response times, and so on).
- Capabilities of the infrastructure management software to allow for the in-service maintenance of the infrastructure software and hardware under its management, for example:

4. The ability to mark a physical compute node as being in a kind of "maintenance mode" and for the infrastructure management software to ensure that all the running workloads are moved off or rescheduled on to other available nodes (after checking that there is sufficient capacity), before marking the node as being ready for whatever maintenance activity needs to be performed.
5. The ability to coordinate, automate, and allow the declarative input of in-service software component upgrades, such as internal orchestration and scheduler components in the infrastructure management software.

Note

The above only refers to the components. It is expected that any service-level assurance does not add any further requirements to the infrastructure, but rather takes the extracted data and builds service models based on the knowledge it has of the services being offered.

9.5.1 Telemetry and observability

Operating complex distributed systems, such as a Telco network, is a task that is becoming increasingly challenging as the network complexity and the production excellence requirements grow. There are multiple reasons for this, although they originate in the nature of the system concept. To be able to provide Telco services, a complex system is decomposed into multiple different functional blocks called network functions. Internal communication between the diverse network functions of a distributed system is based on message exchange. To formalize this communication, clearly defined interfaces are introduced, and protocols designed. Even though the architecture of a Telco network is systematically formalized on the worldwide level, heterogeneity of services, functions, interfaces, and protocols cannot be avoided. By adding the multivendor approach in the implementation of Telco networks, the outcome is a system with a remarkably high level of complexity that requires significant effort for managing and operating it.

A large and complex ecosystem of end-user services requires a formalized approach for achieving high reliability and scalability. The discipline which applies the well-known practices of software engineering to operations is known as Site Reliability Engineering. It was conceived at Google as a means of overcoming the limitations of the common DevOps approach.

The requirements of the common support system (comprising the Operation Support System (OSS) and the Business Support System (BSS)) are redefined. They are driven by the introduction of new technologies in computing infrastructure and modern data centres, with the abstraction of resources, known as virtualization and cloud computing. This brings many advantages, such as easy scaling, error recovery, reaching a high level of operational autonomy, and so on. It also brings many new challenges in the Telecom network management space. These challenges are mostly directed towards the dynamic nature of the system, the orientation towards microservices instead of a silo approach, and huge amounts of data which have to be processed, in order to understand the internal status of the system. Therefore, the need arises for improved ways to monitor the systems: observability.

9.5.1.1 Why observability?

Knowing the status of all the services and functions at all levels in a cloud-based service offering is essential, in order to act fast and proactively before the users notice any issues and, most importantly, before they call the help desk.

A common approach to understanding the aforementioned Telco network status in conventional non-cloud environments is referred to as monitoring. Usually it would include metric information related to resources, such as CPUs, memory, HDD, Network I/O, as well as business-related technical key performance indicators (KPIs), such as the number of active users, the number of registrations, and so on. This monitoring data is represented as a time

series, retrieved at regular intervals, usually with granulation of five to 30 minutes. In addition, asynchronous messages, such as alarms and notifications, are exposed by the monitored systems, in order to provide information about foreseen situations. It is worth noting that metric data provides an approximation of the health of the system, while the alarms and notifications supply more information about the problem. In general, they provide information about known unknowns: anticipated situations occurring at random time. However, such information is rarely sufficient for understanding the problem (root cause analysis (RCA)). Therefore, it is necessary to retrieve more data related to the problem, such as logs and network signalization. Logs are an invaluable means with which to obtain more granular information about the code execution. Network packet captures and traces are useful, since telecommunication networks are distributed systems where components communicate using various protocols, and the communication can be examined, in order to obtain details of the problem.

As the transition towards cloud environments takes place simultaneously with the introduction of the DevOps mindset, the conventional monitoring approach becomes suboptimal. Cloud environments allow greater flexibility as the microservice architecture is embraced to bring improvements in operability. Therefore, the automation can be used to a greater extent than ever before. Automation in telecom networks usually supposes actions based on decisions derived from the system output data (system observation). To derive useful decisions, data with a rich context is necessary. The conventional monitoring approach has to be improved, in order to retrieve sufficient data, not only from the wider context, but also without delays, as soon as the data is produced or available. The new, enhanced approach was introduced as a concept of observability, borrowed from the control theory which states that it is possible to draw conclusions about a system's internal state based on external outputs.

This requires the collection of alarms and telemetry data from the physical layer (wires), the cloud infrastructure up to the network, and applications and services (virtualized network functions (VNFs)) running on top of the cloud infrastructure, typically isolated by tenants.

Long-term trending data is essential for capacity planning purposes. It is typically collected, aggregated, and retained over the full lifespan. To keep the amount of collected data manageable, automatic data reduction algorithms are typically used, for example, by merging the data points from the smallest intervals to more granular intervals.

The Telco cloud infrastructure typically consists of one or more regional data centres, central offices, and edge sites. These are managed from the redundant central management sites, each hosted in their own data centres.

The network services and applications deployed on a Telco cloud, and the Telco cloud infrastructure, are usually managed by separate teams. Therefore, the monitoring solution must be able to keep the access to the monitoring data isolated between the tenants and the cloud infrastructure operations. Some monitoring data from the cloud infrastructure layer must be selectively available to the tenant monitoring applications, in order to correlate, for example, the Network Functions/Services data with the underlying cloud infrastructure data.

9.5.1.2 What to observe

Typically, when it comes to data collection, the following three questions arise:

- What kind of data should be collected?
- Where should the data be sent?
- Which protocol, interface, and format should be used?

9.5.1.3 What kind of data to collect

An assessment of the kind of data that needs to be collected should begin by iterating over the following physical and virtual infrastructure components:

448. Network Services across sites and tenants.
449. Virtualized functions per site and tenant.
450. Individual virtual machines and containers.
451. Virtualization infrastructure components.
452. Physical servers (compute) and network elements.
453. Tool servers with their applications (DNS, identity management, Zero-Touch Provisioning, and so on).
454. Cabling.

9.5.1.4 Data categories

There are four main observability categories. They are as follows:

- **Metrics**: Metrics, or telemetry report counters and gauge levels, can either be pulled periodically, for example, via SNMP or REST, or pushed as streams, using gRPC or NETCONF. Receivers of the published data can register for certain sensors, or for a certain publisher to a message broker. These messages must be structured, in order to be parsed successfully.
- **Events**: Events indicate state variance beyond a specified threshold. They are categorized by severity, often with a description of what happened. The most common transport protocol is SNMP with its trap-and-inform messages. These messages are generated by physical and logical network elements. The messages can also be generated by monitoring applications with statically configured thresholds, or dynamically by Machine Learning

  (ML) algorithms. Generally, they are describing anomalies.
- **Logs**: Logs are a record of the messages generated by the software for most devices (compute and network) and virtual applications. Logs are transported over SYSLOG and tend to come in high volumes.
- **Traces**: Traces are end-to-end signalling messages (events) created to fulfil the execution of requests on the distributed system services. In other words, traces are action points executed to provide responses to the requests set to the distributed system service. Even the call can be thought of as a request which begins with a SIP INVITE message.

9.5.1.5 Where to send the data

If the observability data has to be sent from its sources, or producers, to specific destinations, or consumers, then this creates a high degree of dependency between the producers and the consumers. It is prone to errors, especially in the case of configuration changes. Ideally, the data producers must not be impacted by any change in the data consumers and vice versa. This is achieved by decoupling the data producers from the data consumers through the use of brokers. The producers always send their data to the same endpoint: the broker. The consumers register with the broker for data that is of interest to them and always receive their data from the broker.

9.5.1.6 Which protocol, interface, and format to use

While the protocols and interfaces are dictated by the selection of the message broker (common data bus) system, the data format is usually customizable according to the users' needs. The concept of the Schema Registry mechanism, well known in the world of big data, is helpful here to make sure that the message structures and formats are consistently used.

9.5.1.7 The architecture

In geographically dispersed large cloud deployments, a given Telco cloud may have several cloud infrastructure components, as well as a large set of virtualized workloads (VNFs/CNFs). It is important to monitor all of these workloads and infrastructure components. It is even more important to be able to distinguish between the metrics provided by these entities to determine the performance and/or issues in such deployments.

The cloud deployment tends to shrink and expand according to customer demand. Therefore, an architecture is required that can scale on demand and does not force a strong tie between various entities. This means that the workloads and cloud infrastructure components that provide telemetry and performance metrics must not be burdened to discover each other. The capacity (for example, speed and storage) of one component must not force overrun or underrun situations that could cause critical data to be lost or delayed to a point where they become useless.

Operators in charge of the cloud infrastructure (physical infra plus virtualization platform) require detailed alarms and metrics to run their platform efficiently. While they need indicators as to how well or poorly individual virtual machines and containers are running, they do not need a view inside these workloads. In fact, information on what the workloads are doing and how they are performing should not be accessible to the NFVI operators. The architecture must allow for different consumers to grant or deny access to available resources.

Multiple workloads or network services can be deployed on one or more sites. These workloads require logical separation, for security and privacy reasons, or to avoid an accidental mixing up of their respective metrics. This can be achieved by deploying these workloads within their own tenant spaces. All virtualization platforms offer such isolation down to virtual networks per tenant.

9.5.1.8 Push versus pull

Push and pull are two widely deployed models for providing telemetry data.

9.5.1.8.1 Pull model

The following characteristics are typical of the pull model:

455. The consumers are required to identify the producers of the data.
456. Once the producers have been identified, there should be a tight relationship (synchronization) between the producer and the consumer. This makes the systems very complex in terms of configuration and management. For example, if a producer moves to a different location, or restarts, then the consumer must rediscover the producer and bind their relationship again.
457. Data is pulled explicitly by the consumer. The consumer must have the appropriate bandwidth, compute power, and storage to deal with this data, for example, SNMP pull or walk.
458. One problem with the pull model is that both consumers and producers have to have the means for load/ performance regulation in cases where the set of consumers overload the pull request serving capabilities of the producer.

9.5.1.8.2 Push model

The following characteristics are typical of the push model:

459. A declarative definition of destination: the data producers know explicitly where to stream or push their data.

460. A well-known data broker is used: all the consumers and producers know about it through declarative definition. The data broker can be a bus, such as RabitMQ, Apache Kafka, or Apache Pulsar.

461. There are no restrictions on the bandwidth or data storage for the producers or consumers. The producers produce the data, and stream or push it to the broker. The consumers pull the data from the broker. No explicit synchronisation is required between the producers and the consumers.

462. Lifecycle Management (LCM) events, such as moves and reboots or restarts, of the consumers or producers have no impact on others.

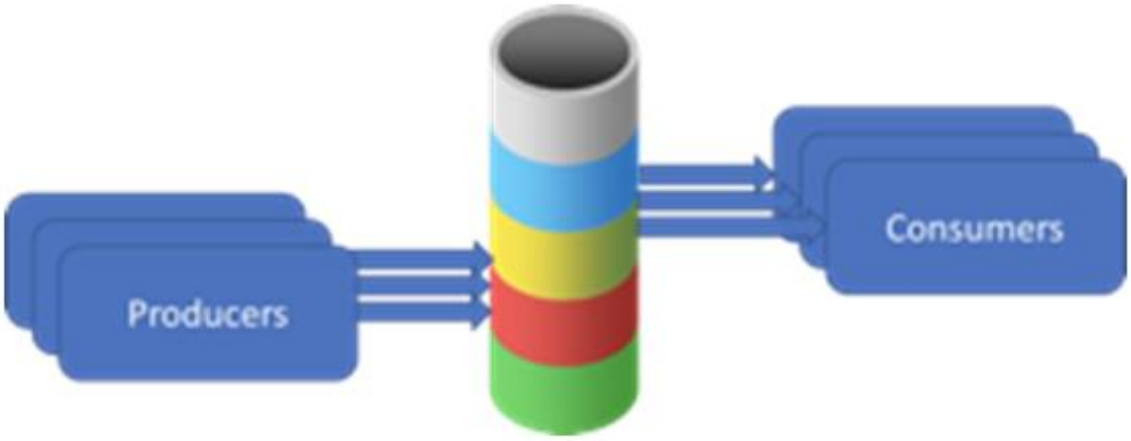
463. Producers and consumers can be added or removed at will. There is no impact on the system. This makes this model flexible and scalable, and better suited for large or small geographically dispersed Telco clouds.

464. Examples of push model include gRPC, SNMP traps, and syslogs.

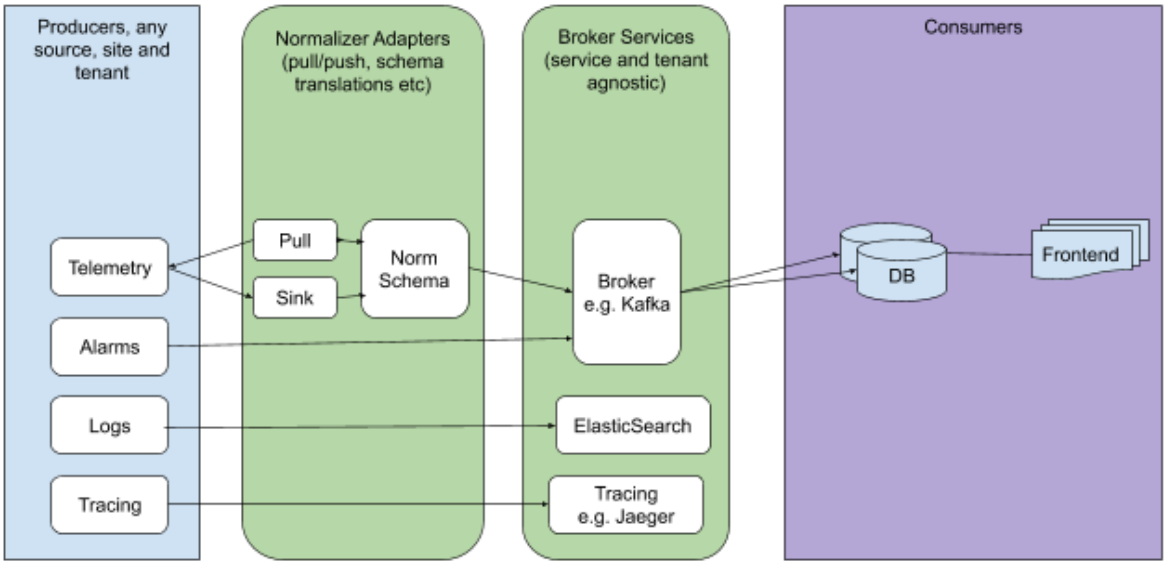
9.5.1.9 Producers, consumers, and message brokers

In an ideal case, the observability data is sent directly to the message broker in an agreed format, so that consumers can then take the data and understand it without additional logic. The message brokers do not impose limits on the data types.

Enforcing the correct message structures (carrying the data) is performed using Schema Registry concepts. Although it is not necessary to use a Schema Registry, it is recommended.



Producers and consumers



Broker services

9.6 Automation

9.6.1 Infrastructure lifecycle management automation

In a typical telecom operator environment, infrastructure lifecycle management (LCM) is complex and error-prone. The environment, with its multiple vendors and products, is expensive to maintain, in terms of time and costs, owing to the need for complex planning, testing, and the out-of-business-hours execution required to perform disruptive maintenance, such as upgrades, and to mitigate outages to mission-critical applications. Processes and tooling for infrastructure management across hybrid environments create additional complexity due to the different levels of access to infrastructure: hands-on access to the on-premise infrastructure, but only restricted access to consumable services offered by public clouds.

Lifecycle operations, such as software or hardware upgrades (including complex and risky firmware updates), typically involve time-consuming manual research and substantive testing to ensure that an upgrade is available or required, and does not conflict with the current versions of other components. In a complex and at-scale hybrid multicloud environment, consisting of multiple on-premise and public clouds, such a manual process is ineffective and, in many cases, impossible to execute in a controlled manner. Hence the need for automation.

The goals of LCM are to provide a reliable administration of a system from its provisioning, through its operational stage, to its final retirement.

The key functions of infrastructure LCM are as follows:

465. Hybrid multicloud support. This means that LCM works across physical, virtual, and cloud environments, supporting on-premise, cloud, and distributed environments.

466. Complete system lifecycle control (plan/design, build, provision, operate/manage, retire, and recycle/scrap).

467. Enablement for the automation of most system maintenance tasks.

The key benefits of infrastructure LCM automation are as follows:

468. Agility: standardisation of the LCM process by writing and running IaaC allows you to develop, stage, and produce environments quickly and easily.

469. Operational consistency: automation of the lifecycle results in consistently maintaining the desired state. It reduces the possibility of errors and decreases the chances of incompatibility issues within the infrastructure.

470. Mitigation of human-related risks: automation reduces risks related to human error and rogue activities, and safeguards against leakage of information about the company, in the event of an employee leaving the organization.

471. Higher efficiency: this is achieved by minimizing human inaccuracies and eliminating the lack of knowledge about the infrastructure installed base and its configuration, using the CI/CD techniques adapted to infrastructure.

472. Cost/time saving: engineers save on time and cost. This can be wisely invested in performing higher-value jobs. Additional cost savings can be made in the cloud with better use of cloud resources using LCM automation.

9.6.1.1 Infrastructure LCM Automation Framework

The essential building blocks for infrastructure LCM automation are as follows:

473. representation model

474. repository functions

475. available software versions and dependencies

476. orchestration engine

Automated LCM uses the representation model to do the following:

477. Abstract various automation technologies.

478. Promote evolution from automation understood as automation of human tasks, to autonomous systems using intent-based, declarative automation, supported by evolving AI/ML technologies.

Automated LCM uses the repository functions to do the following:

479. Store and manage configuration data.

480. Store and manage metrics-related data, such as event data, alert data, and performance data.

481. Maintain currency of data through the use of the discovery of current versions of software modules.

482. Track and account for all systems, assets, and subscriptions (monitoring).

483. Provide an inventory of all virtual and physical assets.

484. Provide a topological view of interconnected resources.

485. Support network design functions.

Automated LCM uses the available IAC software versions and dependencies component to do the following:

486. Store information about the available software versions, software patches, and dependency expectations.

487. Determine the recommended version of a software item, such as firmware, as well as dependencies on other items in the node, to ensure compliance, and to maintain system integrity.

488. Determine the recommended versions of the foundation software running on the cluster.

Automated LCM uses the orchestration engine to do the following:

489. Dynamically remediate dependencies during the change process, to optimise the outcome.

490. Ensure that the system is consistent across its lifecycle by maintaining it in accordance with the intent templates.

9.6.1.2 LCM automation principles and best practices

Infrastructure LCM automation best practices should be guided by the following principles:

491. Everything Codified: use explicit coding to configure files not only for initial provisioning but also as a single source of truth for the whole infrastructure lifecycle, to ensure consistency with the intended configuration templates, and to eliminate configuration drift.

492. Version Controlled: use stringent version control for the infrastructure code to allow proper lifecycle automation.

493. Self-Documentation: the code itself represents the updated documentation of the infrastructure, to minimise the documentation maintenance burden and to ensure the currency of the documentation.

494. Code Modularisation: this applies to the IaaC principles of the microservices architecture where the modular units of the code can be independently deployed and life cycled in an automated fashion.

495. Immutability: IT infrastructure components are required to be replaced for each deployment during the system lifecycle, to be consistent with immutable infrastructure, to avoid configuration drift, and to restrict the impact of undocumented changes in the stack.

496. Automated Testing: this is the key to error-free post-deployment lifecycle processes and to the elimination of lengthy manual testing processes.

497. Unified Automation: this uses the same Infrastructure LCM Automation templates, toolsets, and procedures across different environments such as Dev, Test, QA, and Prod, to ensure the consistency of the lifecycle results and to reduce operational costs.

498. Security Automation: the security of the infrastructure is critical to the overall security, dictating the use of consistent automated security procedures for threat detection, investigation, and remediation through all infrastructure lifecycle stages and all environments.

9.6.2 Software onboarding automation and CI/CD requirements

9.6.2.1 Software onboarding automation

For software deployment, as far as cloud infrastructure services and workloads are concerned, automation is the core of the DevOps concept. Automation allows the elimination of manual processes, thereby reducing human errors and speeding up software deployments. The prerequisite is to install the CI/CD tools chain to do the following:

499. Build, package, and test the application/software.

500. Store environment's parameters and configurations.

501. Automate the delivery and deployment.

The CI/CD pipeline is used to deploy, test, and update the cloud infrastructure services, as well as to onboard workloads hosted on the infrastructure. Typically, this business process consists of the following key phases:

- Tenant engagement and software evaluation:

1. In this phase, the request from the tenant to host a workload on the cloud infrastructure platform is assessed and a decision is made on whether or not to proceed with the hosting request.
2. If the cloud infrastructure software needs to be updated or installed, an evaluation is made of the impact (including to tenants), and whether or not it is all right to proceed.
3. This phase may also involve the tenant accessing a prestaging environment to perform their own evaluation or prestaging activities, or both, in preparation for later onboarding phases.

- Software packaging:
  4. The main outcome of this phase is to produce the software-deployable image and the deployment manifests, such as TOSCA blueprints, HEAT templates, or Helm charts, that will define the service attributes of the cloud infrastructure.
  5. The software packaging can be automated or performed by designated personnel, through self-service capabilities (for tenants), or by the cloud infrastructure operations team.
- Software validation and certification:
  6. In this phase, the software is deployed and tested, to validate it against the service design and other operator-specific acceptance criteria, as required.
  7. Software validation and certification should be automated using CI/CD toolsets/pipelines, and Test-as-a-Service (TaaS) capabilities.
- Publish software:
  8. Tenant workloads: after the software is certified, the final onboarding process phase is for it to be published in the cloud infrastructure production catalogue, from where it can be instantiated on the cloud infrastructure platform by the tenant.
  9. Cloud infrastructure software: after the software is certified, it is scheduled for deployment in concurrence with the user community.

All the phases described above can be automated using technology-specific toolsets and procedures. Therefore, details of such automation are left for the technology-specific Reference Architecture and Reference Implementation specifications.

9.6.2.2 Software CI/CD requirements

The requirements, including for CI/CD for ensuring software security scans, image integrity checks, OS version checks, and so on, prior to deployment, are listed in Table 9-4 (below).

..Note:: The tenant processes for application LCM, such as updates, are out of scope. For the purpose of these requirements, CI includes Continuous Delivery, and CD refers to Continuous Deployment.

Table 21Automation CI/CD

Ref #	Description	Comments/Notes
auto.cicd.001	The CI/CD pipeline must support deployment on any cloud and cloud infrastructures, including different hardware accelerators.	CI/CD pipelines automate CI/CD best practices into repeatable workflows for integrating code and configurations into builds, testing builds including validation against design and operator-specific criteria, and delivery of the product onto a runtime

		environment. Example of an open-source cloud native CI/CD framework is the Tekton project ([159])
auto.cicd.002	The CI/CD pipelines must use event-driven task automation	
auto.cicd.003	The CI/CD pipelines should avoid scheduling tasks	
auto.cicd.004	The CI/CD pipeline is triggered by a new or updated software release being loaded into a repository	The software release can be source code files, configuration files, images, manifests. Operators may support a single or multiple repositories and may specify which repository is to be used for these releases. An example of an open source repository is the CNCF Harbor ([160])
auto.cicd.005	The CI pipeline must scan source code and manifests to validate compliance with design and coding best practices.	
auto.cicd.006	The CI pipeline must support the build and packaging of images and deployment manifests from source code and configuration files.	
auto.cicd.007	The CI pipeline must scan images and manifests to validate for compliance with security requirements.	See section 7.10 (Consolidated security requirements). Examples of such security requirements include only ingesting images, source code, configuration files, etc., only from trusted sources.
auto.cicd.008	The CI pipeline must validate images and manifests	Example: different tests
auto.cicd.009	The CI pipeline must validate with all hardware offload permutations and without hardware offload	
auto.cicd.010	The CI pipeline must promote validated images and manifests to be deployable.	Example: promote from a development repository to a production repository
auto.cicd.011	The CD pipeline must verify and validate the tenant request	Example: RBAC, request is within quota limits, affinity/anti-affinity, etc.
auto.cicd.012	The CD pipeline after all validations must turn over	

	control to orchestration of the software
auto.cicd.013	The CD pipeline must be able to deploy into Development, Test, and Production environments
auto.cicd.014	The CD pipeline must be able to automatically promote software from Development to Test and Production environments
auto.cicd.015	The CI pipeline must run all relevant Reference Conformance test suites

Table 9-4: Automation CI/CD

9.6.2.3 CI/CD design requirements

A number of CI/CD pipeline properties and rules must be agreed between the different actors to allow the smooth deployment and testing of the cloud infrastructures and the hosted network functions, irrespective of whether the jobs operate open-source or proprietary software. These properties and rules prevent a specific deployment or testing operation from forcing a particular CI/CD design or, worse, from requesting the deployment of a fully dedicated CI/CD toolchain for a particular network service.

At first glance, the deployment and test job must not request specific CI/CD tools, such as [144] or [145]. However, there are many other ways where deployment and test jobs can constrain the end users, from the build servers to artefact management. Any manual operation is discouraged, regardless of whether it is about the deployment or the test resources.

The following requirements also aim to deploy all CI/CD toolchains smoothly and easily via simple playbooks, as targeted by the Reference Conformance suites currently leveraging [146].

Ref #	Description	Comments/Notes
design.cicd.001	The pipeline must allow the chaining of independent CI/CD jobs.	For example, all deployment and test operations, from bare metal to Kubernetes and OpenStack, and to network services.
design.cicd.002	The pipeline jobs should be modular.	This allows the execution of jobs independently of others, for example, starting with an existing OpenStack deployment.
design.cicd.003	The pipeline must decouple the deployment and the test steps.	
design.cicd.004	The pipeline should leverage the job artefacts specified by the CI/CD tools, provided by the operator.	
design.cicd.005	The pipeline must execute all the relevant Reference Conformance suites without modification.	

design.cicd.006	The software vendors/providers must use CI/CD tools provided by the operator.	
design.cicd.007	All jobs must be packaged as containers.	
design.cicd.008	All jobs must leverage a common execution to allow the templating of all the deployment and test steps.	
design.cicd.009	The deployment jobs must publish all outputs as artefacts in a specified format.	For example, OpenStack RC, kubeconfig, yaml, and so on. Anuket specifies the formats in the RC.
design.cicd.010	The test jobs must pull all the inputs as artefacts in a specified format.	For example, OpenStack RC, kubeconfig, yaml, and so on. Anuket specifies the formats in the RC.
design.cicd.011	The test jobs must conform to the Reference Conformance test case integration requirements.	

Table 9-5: CI/CD design

9.6.3 Tenant creation automation

9.6.3.1 Pretenant creation requirements

Topics related to pretenant creation requirements include the following:

- Tenant Approval: use, capacity, data centres, and so on.
    10. Validate that the Tenant's planned use meets the Operators Cloud Use policies (see common/glossary: operational and administrative terminology).
    11. Validate that the capacity available within the requested cloud sites can satisfy the tenant requested quota for the vCPU, RAM, disk, and network bandwidth.
    12. Validate that the cloud infrastructure can meet the tenant’s performance requirements (for example, I/O, latency, jitter, and so on).
    13. Validate that the cloud infrastructure can meet tenant’s resilience requirements.
- For environments that support Profiles and workload flavours:
    14. Verify that any requested private flavours have been created.
    15. Verify that the metadata for these private flavours has been created.
    16. Verify that the tenant has the necessary permissions to use the requested private flavours.
    17. Validate that host aggregates are available for the specified flavours (public and private).
    18. Verify that the metadata matches the requested new flavours and host aggregates.
- Tenant networks
    19. Verify that the networks requested by the tenant exist.

20. Verify that the security policies are correctly configured to approved ingress and egress only.

- Tenant admin, tenant member, and other tenant role approvals for user by role

21. Add all the tenant members and configure their assigned roles in the Enterprise Identity and Access management system (for example, LDAP).
22. Verify that these roles have been created for the tenant.

- Approvals of the tenants' images and manifests

23. Verify and validate the tenants' images and manifests using virus scans, verifying the correct OS version and patch, and so on.

..Note:: Tenants may also add new images or replace existing images after their environments are created and will be subject to image security measures.

- Create, verify, and validate the tenant

24. Create the tenant.
25. Use a proto- or tenant-provided HEAT template or Helm chart for an NF, and perform a sanity test (for example, by using scripts to test the creation of the VM or container, ping tests, and so on).

10 Challenges and Gaps

10.1 Introduction to Challenges and Gaps

This section is dedicated to identifying the challenges and gaps found in the course of the development of the Reference Model to ensure that it continues to be of strategic and tactical value for the long term. Should a challenge or gap not be identified that has not already been addressed in the model itself, the community may assume that it will remain an unknown. Therefore, the community is encouraged to engage with and raise an issue with the appropriate working groups to close the gap. In this manner, the Reference Model can continuously improve.

10.2 Challenges

The continuous challenge is to finalize a stable version from which all the stakeholders in the application value chain can derive the intended value of a common cloud infrastructure. This maturity level is reached when the released Reference Model version is adopted by the stakeholders into their application development and deployment cycles.

10.3 Gaps

This section addresses the major open issues identified in the development of the Reference Model, the Reference Architecture, and the Reference Implementation of the Common Cloud Infrastructure Lifecycle Framework.

10.3.1 Discovery

The workloads (VNFs and CNFs) and the cloud infrastructure should be able to discover each other and exchange their capabilities as required or offered. One of the key pain points for most of the operators is the VNF/CNF onboarding, in terms of time and complexity. It could take weeks or months to onboard a VNF or a CNF. A lot of static and laborious checks have to be performed to ensure the compatibility of the workloads with the corresponding cloud infrastructure. The onboarding of the workloads (network functions) should be automated as

much as possible. The workloads and cloud infrastructure should be able to discover and negotiate their capabilities. The following needs to be supported:

502. Discovery and advertising of capabilities:

1. The cloud infrastructure should be able to publish the capabilities it offers to the workloads or network functions.
2. The workloads should be able to query the cloud infrastructure for specific capabilities, such as the number of cores and performance parameters.

503. Negotiation of capabilities/handshake API:

1. The workloads and cloud infrastructure should be able to negotiate on certain capabilities. For example, a workload desires hardware acceleration for high throughput, but should be able to fall back to high throughput offered by the cloud infrastructure via a DPDK offering, and vice versa.

10.3.2 Supporting the load balancing of VNF/CNFs

The ability to dynamically scale a network function by load balancing across multiple instances or replicas of the same VNF or CNF is essential. New architectures and application patterns, such as microservices, make this even more crucial. It must not only be possible to load balance and scale each service layer independently, it is also essential to provide support for chaining the different layers together through Service Function Chaining.

The load balancing and scaling needed for typical enterprise applications is well supported in OpenStack by the Octavia v2 API. The Octavia v2 API is a backwards-compatible superset of the Neutron LBaaS v2 API that it is replacing.

The built-in mechanism in Kubernetes for scaling enterprise-type services and PODs is also sufficient for applications that only use one interface.

What is not supported either in OpenStack or Kubernetes is the scaling and load balancing of a typical VNF or CNF. There is no support in OpenStack to scale stateful L3 applications such as SCTP, QUIC, mTCP, and gRPC. In Kubernetes it is even worse. The built-in Kubernetes network support is tied to the first POD/container interface. Support for the secondary interfaces is managed through the Container Network Interface (CNI) and by CNI plugins, such as Multus, that support the “Kubernetes Network Customs Resource Definition”, specified by the Kubernetes Network Plumbing Group. This specification supports the attachment of network endpoints to PODs, IP address management, and the ability to define interface-specific static routes. There is no support for network orchestration and functions, such as load balancing, routing, ACL, and firewalls.

10.3.3 Closed-loop automation

The state of a system is defined by a set of variables that fully describe the system and determine the response of the system to any given set of inputs. A closed-loop automation system automatically maintains the specified desired state of the controlled system.

Closed-loop automation is evolving as a major advancement in telecommunication network automation. In the context of telecommunications systems, closed-loop automation is a system that, in a continuous loop, programmatically validates the state of the cloud infrastructure against the declared desired state. In the case of deviation from the desires state, it automatically takes remediation actions necessary for bringing the actual state to the desired state. The Reference Model specification will address this important area in the next releases.

10.3.4 Hybrid multicloud: APIs

Section "8.5 Multi-Cloud Interactions Model" defines several core roles within the multicloud model and discusses stereotypical interactions between them. However, the model realises that a federated cloud requires the definition of, and agreement on, a set of APIs. The current fragmentation in the industry is caused by the following factors:

504. Proprietary APIs, some of which have been adopted as default industry standards.
505. A number of open-source community projects aiming to provide abstract interfaces to wrap proprietary APIs.
506. Vendors offering to act as brokers.
507. Standards and industry APIs to address specific subsets of interactions.

10.3.5 AF_XDP

Linux-native AF_XDP promises high enough packet processing performance and simplification, compared to what SR-IOV and DPDK require for initial installation and later lifecycle management. Nevertheless, it will take time until AF_XDP-based solutions are financially invested and have matured enough in both virtualization infrastructure and network functions.

11 Appendix A: Guidelines For Application Vendors

11.1 Goals

This Appendix has the following two goals:

- To provide guidance to the VNF, or more generally to the application vendors, on how to consume the CNTT Reference Model and Architectures.
- To provide usable definitions of maturity levels for the VNF software architecture between physical-to-virtual migration and cloud native.

The goal is not to be prescriptive on how to design new applications or redesign existing applications, but rather to stay within the scope of focusing on the interface and interaction between the applications and the platform.

11.2 Introduction and terminology

Taking advantage of the RM and RA environments with common capabilities, applications can be developed and deployed more rapidly, providing more service agility and easier operations. The extent to which this can be achieved depends on the levels of decoupling between the applications and the infrastructure, or the platform underneath the applications:

1. Infrastructure:

a. Application functionality or application control requires infrastructure components beyond RM profiles or infrastructure configuration changes beyond RA-exposed APIs. Generally, such an application is tightly coupled with the infrastructure. This results in an appliance deployment model.

b. Application control using RA APIs finds nodes (already configured in support of the profiles) with the required infrastructure components. In these nodes, using RA APIs configures the infrastructure components that make the application work. An example of this is an application that needs certain acceleration adapters available in the RM profile to achieve the latency requirements and is exposed through RA APIs.

c. Application control using RA APIs finds nodes (already configured in support of the profiles) with optional infrastructure components. In these nodes, using RA APIs configures the infrastructure components in a way that improves the application's performance. An example of this is an application that has a better TCO with certain acceleration adapters, but can also work without them.

d. Application control using RA APIs finds general profile node without any specific infrastructure components.

2. Platform services

a. Application functionality or application control can work only with its own components, instead of using RA-defined platform services.

b. With a custom integration effort, an application can be made to use RA-defined platform services.

c. An application is designed and can be configured for running with RA-defined platform services.

3. Application resilience

a. The application is designed and tested to run only on a carrier-grade platform with predictable infrastructure availability and performance.

b. The application is designed and tested for full failures of infrastructure hardware and software components. It is not designed and tested for infrastructure impairment, as the application still needs predictable infrastructure performance (such as CPU cycles and network latencies).

c. The application is designed to run on shared cloud platforms and is tested for resilience to infrastructure impairments. This is relevant for sizing infrastructure and application operations (which are often other telecom operators' organizational units or external third parties).

4. Other application functionalities (decomposition and manageability for scaling, availability, and upgrades):

a. The application consists of monolithic components, including algorithms that have different scaling (for example, depending on the type of traffic) or availability requirements, or both.

b. The application consists of smaller, tightly coupled components.

c. The application is decomposed, with loosely coupled or decoupled components.

d. Availability, such as N+K or 1+1, is defined during the application design and is not configurable at the time of deployment.

e. Mutable or immutable instances of application components.

11.3 VNF design guidelines

A number of software design guidelines, or industry best practices, have been developed over the years. These include microservices, cohesion, and coupling. In addition to the industry best practices, additional guidelines and requirements are specified by ONAP in the VNF or PNF Requirements Documentation. [50]. This section does not supplant these guidelines and practices. The content here only draws attention to some other design considerations that VNF developers need to incorporate into their practices.

Note: Some of these guidelines may be incorporated by operators into their contracts with the VNF vendors.

These guidelines are written in an informal style. Any resemblance to requirements is incidental. The VNF developer **should** ensure that their software and the resultant VNF images conform to the following rules:

- They do not contain malicious code, such as malware, logic bombs, and so on.
- They do not contain code, such as daemons, that exposes them to risk.
- They do not contain clear-text secrets.
- They are only created with content and files from trusted sources.
- They are only packaged with files that have been found to be free of malware and vulnerabilities.

Additionally, in the design and implementation of their software, the VNF developer **should** follow the guidance in the following literature:

- CSA Security Guidance for Critical Areas of Focus in Cloud Computing (latest version) [53].
- OWASP Cheat Sheet Series (OCSS) [53] from the Open Web Application Security Project [52].
- Workload security and vendor responsibility section of the Reference Model.

The VNF Developer **should** ensure that their code is not vulnerable to the OWASP Top Ten Security Risks [153], created by the Open Web Application Security Project [51].

11.4 Miscellaneous

11.4.1 VNF network monitoring capabilities: use case

Network monitoring capabilities exposed by the NFVI platform are used for the passive observation of VNF-specific traffic traversing the NFVI in the following cases:

508. Performance issues or packet drops, or both, are reported in the VNF.

509. Determining performance bottlenecks at VNF level.

510. Performing anomaly detection and network forensics.

Note: It is the responsibility of the NFVI platform to provide the capability to create a virtual interface that has mirrored traffic from the monitored VNF. This port can be attached to the monitoring VNF, so that all traffic from the monitored VNF is available for troubleshooting/debugging purposes.

Annex A Document Management

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	11 Nov 2020	Initial version		Walter Kozlowski / Telstra
2.0	26 Oct 2021	Updated in alignment with LFN Anuket Kali release		Walter Kozlowski / Telstra
3.0	09 Sep 2022	Updated in alignment with LFN Anuket Moselle release		Walter Kozlowski / Telstra
4.0	November 2023	CR1004 Align with Anuket Orinoco Release		Walter Kozlowski / Telstra
5.0	June 2025	CR1005 Align with Anuket Quinnipiac Release	OIG	Petar Torre / Intel Tomas Fredberg / Ericsson Joao A Rodrigues / Nokia Bharath Kumar / HPE

Other Information

Type	Description
Document Owner	Networks Group
Editor / Company	Petar Torre / Intel Tomas Fredberg / Ericsson Joao A Rodrigues / Nokia Bharath Kumar / HPE

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.