
Anuket Reference Architecture for OpenStack based cloud infrastructure (RA1)

Anuket Project of Linux Foundation Networking

Sep 12, 2022

Contents

1	Introduction	4
1.1	Overview	4
1.1.1	Vision	4
1.2	Use Cases	4
1.3	OpenStack Reference Release	5
1.4	Principles	5
1.4.1	Architectural principles	5
1.4.2	OpenStack specific principles	5
1.5	Document Organisation	6
1.6	Terminology	6
1.7	Abbreviations	9
1.8	Conventions	11
2	Architecture Requirements	11
2.1	Reference Model Requirements	11
2.1.1	Cloud Infrastructure Software Profile Requirements for Compute	12
2.1.2	Cloud Infrastructure Software Profile Extensions Requirements for Compute	13
2.1.3	Cloud Infrastructure Software Profile Requirements for Networking	13
2.1.4	Cloud Infrastructure Software Profile Extensions Requirements for Networking	14
2.1.5	Cloud Infrastructure Software Profile Requirements for Storage	15
2.1.6	Cloud Infrastructure Software Profile Extensions Requirements for Storage	15
2.1.7	Cloud Infrastructure Hardware Profile Requirements	16
2.1.8	Cloud Infrastructure Management Requirements	17
2.1.9	Cloud Infrastructure Security Requirements	18
2.2	Architecture and OpenStack Requirements	28
2.2.1	General Requirements	28
2.2.2	Infrastructure Requirements	28
2.2.3	VIM Requirements	30
2.2.4	Interfaces & APIs Requirements	31
2.2.5	Tenant Requirements	31
2.2.6	Operations and LCM	32
2.2.7	Assurance Requirements	32
2.2.8	Architecture and OpenStack Recommendations	32
2.2.9	General Recommendations	33

2.2.10	Infrastructure Recommendations	33
2.2.11	VIM Recommendations	34
2.2.12	Interfaces and APIs Recommendations	35
2.2.13	Tenant Recommendations	35
2.2.14	Operations and LCM Recommendations	35
2.2.15	Assurance Recommendations	36
2.2.16	Security Recommendations	36
3	Cloud Infrastructure Architecture - OpenStack	41
3.1	Resources and Services exposed to VNFs	42
3.1.1	Multi-Tenancy (execution environment)	42
3.1.2	Virtual Compute (vCPU and vRAM)	42
3.1.3	Virtual Storage	43
3.1.4	Virtual Networking Neutron standalone	43
3.1.5	Virtual Networking - 3rd party SDN solution	44
3.1.6	Acceleration	45
3.2	Virtualised Infrastructure Manager (VIM)	45
3.2.1	VIM Core services	45
3.2.2	Tenant Isolation	49
3.2.3	Cloud partitioning: Host Aggregates, Availability Zones	49
3.2.4	Flavor management	50
3.3	Underlying Resources	50
3.3.1	Virtualisation and hypervisors	50
3.3.2	Physical Infrastructure	50
3.4	Cloud Topology	52
4	Cloud Infrastructure & VIM Component Level Architecture	54
4.1	Underlying Resources Configuration and Dimensioning	55
4.1.1	Virtualisation layer	55
4.1.2	Compute	55
4.1.3	Network Fabric	64
4.1.4	Storage Backend	68
4.2	VIM OpenStack Services	70
4.2.1	VIM Services	70
4.2.2	Containerised OpenStack Services	75
4.3	Consumable Infrastructure Resources and Services	77
4.3.1	Support for Cloud Infrastructure Profiles and flavors	77
4.3.2	Logical segregation and high availability	79
4.3.3	Transaction Volume Considerations	79
4.4	Cloud Topology and Control Plane Scenarios	79
4.4.1	Edge Cloud Topology	82
5	Interfaces and APIs	83
5.1	Core OpenStack Services APIs	83
5.1.1	Keystone API	83
5.1.2	Glance API	84
5.1.3	Cinder API	84
5.1.4	Swift API	85
5.1.5	Neutron API	85
5.1.6	Nova API	87
5.1.7	Placement API	88
5.1.8	Heat API	89
5.2	Consolidated Set of APIs	89
5.2.1	OpenStack Interfaces	89

5.2.2	Kubernetes Interfaces	89
5.2.3	KVM Interfaces	89
5.2.4	Barbican API	90
6	Security	90
6.1	Security Requirements	90
6.2	Cloud Infrastructure and VIM Security	90
6.2.1	System Hardening	90
6.2.2	Platform Access	93
6.2.3	Confidentiality and Integrity	95
6.2.4	Workload Security	97
6.2.5	SR-IOV and DPDK Considerations	97
6.2.6	Image Security	97
6.2.7	Security LCM	98
6.2.8	Monitoring and Security Audit	98
7	Operations and Life Cycle Management	101
7.1	Procedural versus Declarative code	101
7.2	Mutable versus Immutable infrastructure	101
7.3	Cloud Infrastructure provisioning and configuration management	101
7.3.1	Underlying resources provisioning	102
7.3.2	VIM deployment	102
7.3.3	Configuration Management	103
7.4	Cloud Infrastructure and VIM Maintenance	103
7.5	Logging, Monitoring and Analytics	103
7.5.1	Logging	104
7.5.2	Monitoring	104
7.5.3	Alerting	104
7.5.4	Logging, Monitoring, and Analytics (LMA) Framework	105
8	Conformance	105
8.1	Requirements and Testing Principles	106
8.2	Test Case Integration and Tooling	106
8.2.1	Anuket Toolchains	107
8.2.2	Test Case Integration	107
8.2.3	Testing Cookbooks	107
8.3	Conformance Test Suite	108
8.3.1	FuncTest in a nutshell	108
8.3.2	Test Case traceability	110
8.4	Test Cases Traceability to Requirements	126
8.4.1	RM/RA-1 Requirements	126
8.4.2	TC Mapping to Requirements	126
8.5	OpenStack Testing Cookbook	127
8.5.1	OpenStack API testing configuration	127
8.5.2	Run OpenStack based cloud infrastructure Conformance	127
9	Gaps, Innovation, and Development	128
9.1	The Gap	128
9.1.1	Autoscaling	128
	References	128

1 Introduction

1.1 Overview

This Reference Architecture is focussed on OpenStack as the Virtualised Infrastructure Manager (VIM) chosen based on the criteria laid out in the Cloud Infrastructure Reference Model [1] (referred to as “Reference Model” or “RM” in the document). OpenStack [2] has the advantage of being a mature and widely accepted open-source technology; a strong ecosystem of vendors that support it, the OpenInfra Foundation for managing the community, and, most importantly, it is widely deployed by the global operator community for both internal infrastructure and external facing products and services. This means that resources with the right skill sets to support a Cloud Infrastructure (or Network Function Virtualisation Infrastructure, NFVI [3]) are available. Another reason to choose OpenStack is that it has a large active community of vendors and operators, which means that any code or component changes needed to support the Common Telco Cloud Infrastructure requirements can be managed through the existing project communities’ processes to add and validate the required features through well-established mechanisms.

1.1.1 Vision

This Reference Architecture specifies OpenStack based Cloud Infrastructure for hosting NFV workloads, primarily VNFs (Virtual Network Functions). The Reference Architecture document can be used by operators to deploy Anuket conformant infrastructure; hereafter, “conformant” denotes that the resource can satisfy tests conducted to verify conformance with this reference architecture.

1.2 Use Cases

Several NFV use cases are documented in OpenStack. For more examples and details refer to the OpenStack Use cases [4].

Examples include:

- **Overlay networks:** The overlay functionality design includes OpenStack Networking in Open vSwitch [5] GRE tunnel mode. In this case, the layer-3 external routers pair with VRRP, and switches pair with an implementation of MLAG to ensure that you do not lose connectivity with the upstream routing infrastructure.
- **Performance tuning:** Network level tuning for this workload is minimal. Quality of Service (QoS) applies to these workloads for a middle ground Class Selector depending on existing policies. It is higher than a best effort queue, but lower than an Expedited Forwarding or Assured Forwarding queue. Since this type of application generates larger packets with longer-lived connections, you can optimise bandwidth utilisation for long duration TCP. Normal bandwidth planning applies here with regards to benchmarking a session’s usage multiplied by the expected number of concurrent sessions with overhead.
- **Network functions:** are software components that support the exchange of information (data, voice, multi-media) over a system’s network. Some of these workloads tend to consist of a large number of small-sized packets that are short lived, such as DNS queries or SNMP traps. These messages need to arrive quickly and, thus, do not handle packet loss. Network function workloads have requirements that may affect configurations including at the hypervisor level. For an application that generates 10 TCP sessions per user with an average bandwidth of 512 kilobytes per second per flow and expected user count of ten thousand (10,000) concurrent users, the expected bandwidth plan is approximately 4.88 gigabits per second. The supporting network for this type of configuration needs to have a low latency and evenly distributed load across the topology. These types of workload benefit from having services local to the consumers of the service. Thus, use a multi-site approach, as well as, deploying many copies of the application to handle load as close as possible to consumers. Since these applications function independently, they do not warrant running overlays to interconnect tenant networks. Overlays also have the drawback of performing poorly with rapid flow setup and may incur too much overhead with large quantities of small packets and therefore we do not recommend them. QoS is desirable for some

workloads to ensure delivery. DNS has a major impact on the load times of other services and needs to be reliable and provide rapid responses. Configure rules in upstream devices to apply a higher-Class Selector to DNS to ensure faster delivery or a better spot in queuing algorithms.

1.3 OpenStack Reference Release

This Reference Architecture document conforms to the OpenStack Wallaby [6] release. While many features and capabilities are conformant with many OpenStack releases, this document will refer to features, capabilities and APIs that are part of the OpenStack Wallaby release. For ease, this Reference Architecture document version can be referred to as “RA-1 OSTK Wallaby.”

1.4 Principles

1.4.1 Architectural principles

This Reference Architecture for OpenStack based Cloud Infrastructure must obey the following set of architectural principles:

1. **Open-source preference:** for building Cloud Infrastructure solutions, components and tools, using open-source technology.
2. **Open APIs:** to enable interoperability, component substitution, and minimise integration efforts.
3. **Separation of concerns:** to promote lifecycle independence of different architectural layers and modules (e.g., disaggregation of software from hardware).
4. **Automated lifecycle management:** to minimise the end-to-end lifecycle costs, maintenance downtime (target zero downtime), and errors resulting from manual processes.
5. **Automated scalability:** of workloads to minimise costs and operational impacts.
6. **Automated closed loop assurance:** for fault resolution, simplification, and cost reduction of cloud operations.
7. **Cloud nativeness:** to optimise the utilisation of resources and enable operational efficiencies.
8. **Security compliance:** to ensure the architecture follows the industry best security practices and is at all levels compliant to relevant security regulations.
9. **Resilience and Availability:** to withstand Single Point of Failure.

1.4.2 OpenStack specific principles

OpenStack considers the following Four Opens essential for success:

- Open Source
- Open Design
- Open Development
- Open Community

This OpenStack Reference Architecture is organised around the three major Cloud Infrastructure resource types as core services of compute, storage and networking, and a set of shared services of identity management, image management, graphical user interface, orchestration engine, etc.

1.5 Document Organisation

Chapter 2 defines the Reference Architecture requirements and, when appropriate, provides references to where these requirements are addressed in this document. The intent of this document is to address all of the mandatory (“must”) requirements and the most useful of the other optional (“should”) requirements. Chapter 3 and 4 cover the Cloud Infrastructure resources and the core OpenStack services, while the APIs are covered in Chapter 5. Chapter 6 covers the implementation and enforcement of security capabilities and controls. Life Cycle Management of the Cloud Infrastructure and VIM are covered in Chapter 7 with stress on Logging, Monitoring and Analytics (LMA), configuration management and some other operational items. Please note that Chapter 7 is not a replacement for the implementation, configuration and operational documentation that accompanies the different OpenStack distributions. Chapter 8 addresses the conformance. It provides an automated validation mechanism to test the conformance of a deployed cloud infrastructure to this reference architecture. Finally, Chapter 9 identifies certain Gaps that currently exist and plans on how to address them (for example, resources autoscaling).

1.6 Terminology

Abstraction: Process of removing concrete, fine-grained or lower-level details or attributes or common properties in the study of systems to focus attention on topics of greater importance or general concepts. It can be the result of decoupling.

Anuket: A LFN open-source project developing open reference infrastructure models, architectures, tools, and programs.

Cloud Infrastructure: A generic term covering **NFVI**, **IaaS** and **CaaS** capabilities - essentially the infrastructure on which a **Workload** can be executed. **NFVI**, **IaaS** and **CaaS** layers can be built on top of each other. In case of CaaS some cloud infrastructure features (e.g.: HW management or multitenancy) are implemented by using an underlying **IaaS** layer.

Cloud Infrastructure Hardware Profile: defines the behaviour, capabilities, configuration, and metrics provided by a cloud infrastructure hardware layer resources available for the workloads.

Host Profile: is another term for a Cloud Infrastructure Hardware Profile.

Cloud Infrastructure Profile: The combination of the Cloud Infrastructure Software Profile and the Cloud Infrastructure Hardware Profile that defines the capabilities and configuration of the Cloud Infrastructure resources available for the workloads.

Cloud Infrastructure Software Profile: defines the behaviour, capabilities and metrics provided by a Cloud Infrastructure Software Layer on resources available for the workloads.

Cloud Native Network Function (CNF): A cloud native network function (CNF) is a cloud native application that implements network functionality. A CNF consists of one or more microservices. All layers of a CNF are developed using Cloud Native Principles including immutable infrastructure, declarative APIs, and a “repeatable deployment process”. This definition is derived from the Cloud Native Thinking for Telecommunications Whitepaper, which also includes further detail and examples.

Compute Node: An abstract definition of a server. A compute node can refer to a set of hardware and software that support the VMs or Containers running on it.

Container: A lightweight and portable executable image that contains software and all of its dependencies. OCI defines **Container** as “An environment for executing processes with configurable isolation and resource limitations. For example, namespaces, resource limits, and mounts are all part of the container environment.” A **Container** provides operating-system-level virtualisation by abstracting the “user space”. One big difference between **Containers** and **VMs** is that unlike VMs, where each **VM** is self-contained with all the operating systems components are within the **VM** package, containers “share” the host system’s kernel with other containers.

Container Image: Stored instance of a container that holds a set of software needed to run an application.

Core (physical): An independent computer processing unit that can independently execute CPU instructions and is integrated with other cores on a multiprocessor (chip, integrated circuit die). Please note that the multiprocessor chip is also referred to as a CPU that is placed in a socket of a computer motherboard.

CPU Type: A classification of CPUs by features needed for the execution of computer programs; for example, instruction sets, cache size, number of cores.

Decoupling, Loose Coupling: Loosely coupled system is one in which each of its components has, or makes use of, little or no knowledge of the implementation details of other separate components. Loose coupling is the opposite of tight coupling

Encapsulation: Restricting of direct access to some of an object's components.

External Network: External networks provide network connectivity for a cloud infrastructure tenant to resources outside of the tenant space.

Fluentd: An open-source data collector for unified logging layer, which allows data collection and consumption for better use and understanding of data. **Fluentd** is a CNCF graduated project.

Functest: An open-source project part of Anuket LFN project. It addresses functional testing with a collection of state-of-the-art virtual infrastructure test suites, including automatic VNF testing.

Hardware resources: Compute/Storage/Network hardware resources on which the cloud infrastructure platform software, virtual machines and containers run on.

Huge pages: Physical memory is partitioned and accessed using the basic page unit (in Linux default size of 4 KB). Hugepages, typically 2 MB and 1GB size, allows large amounts of memory to be utilised with reduced overhead. In an NFV environment, huge pages are critical to support large memory pool allocation for data packet buffers. This results in fewer Translation Lookaside Buffers (TLB) lookups, which reduces the virtual to physical pages' address translations. Without huge pages enabled high TLB miss rates would occur thereby degrading performance.

Hypervisor: a software that abstracts and isolates workloads with their own operating systems from the underlying physical resources. Also known as a virtual machine monitor (VMM).

Instance: is a virtual compute resource, in a known state such as running or suspended, that can be used like a physical server. It can be used to specify VM Instance or Container Instance.

Kibana: An open-source data visualisation system.

Kubernetes: An open-source system for automating deployment, scaling, and management of containerised applications.

Monitoring (Capability): Monitoring capabilities are used for the passive observation of workload-specific traffic traversing the Cloud Infrastructure. Note, as with all capabilities, Monitoring may be unavailable or intentionally disabled for security reasons in a given cloud infrastructure instance.

Multi-tenancy: feature where physical, virtual or service resources are allocated in such a way that multiple tenants and their computations and data are isolated from and inaccessible by each other.

Network Function (NF): functional block or application that has well-defined external interfaces and well-defined functional behaviour. Within **NFV**, a **Network Function** is implemented in a form of **Virtualised NF** (VNF) or a **Cloud Native NF** (CNF).

NFV Orchestrator (NFVO): Manages the VNF lifecycle and **Cloud Infrastructure** resources (supported by the **VIM**) to ensure an optimised allocation of the necessary resources and connectivity.

Network Function Virtualisation (NFV): The concept of separating network functions from the hardware they run on by using a virtual hardware abstraction layer.

Network Function Virtualisation Infrastructure (NFVI): The totality of all hardware and software components used to build the environment in which a set of virtual applications (VAs) are deployed; also referred to as cloud infrastructure. The NFVI can span across many locations, e.g., places where data centres or edge nodes are operated. The network providing connectivity between these locations is regarded to be part of the cloud infrastructure. **NFVI**

and VNF are the top-level conceptual entities in the scope of Network Function Virtualisation. All other components are sub-entities of these two main entities.

Network Service (NS): composition of **Network Function(s)** and/or **Network Service(s)**, defined by its functional and behavioural specification, including the service lifecycle.

Open Network Automation Platform (ONAP): A LFN project developing a comprehensive platform for orchestration, management, and automation of network and edge computing services for network operators, cloud providers, and enterprises.

ONAP OpenLab: ONAP community lab.

Open Platform for NFV (OPNFV): A collaborative project under the Linux Foundation. OPNFV is now part of the LFN Anuket project. It aims to implement, test, and deploy tools for conformance and performance of NFV infrastructure.

OPNFV Verification Program (OVP): An open-source, community-led compliance and verification program aiming to demonstrate the readiness and availability of commercial NFV products and services using OPNFV and ONAP components.

Platform: A cloud capabilities type in which the cloud service user can deploy, manage and run customer-created or customer-acquired applications using one or more programming languages and one or more execution environments supported by the cloud service provider. Adapted from ITU-T Y.3500. This includes the physical infrastructure, Operating Systems, virtualisation/containerisation software and other orchestration, security, monitoring/logging and life-cycle management software.

Prometheus: An open-source monitoring and alerting system.

Quota: An imposed upper limit on specific types of resources, usually used to prevent excessive resource consumption by a given consumer (tenant, VM, container).

Resource pool: A logical grouping of cloud infrastructure hardware and software resources. A resource pool can be based on a certain resource type (for example, compute, storage and network) or a combination of resource types. A **Cloud Infrastructure** resource can be part of none, one or more resource pools.

Simultaneous Multithreading (SMT): Simultaneous multithreading (SMT) is a technique for improving the overall efficiency of superscalar CPUs with hardware multithreading. SMT permits multiple independent threads of execution on a single core to better utilise the resources provided by modern processor architectures.

Shaker: A distributed data-plane testing tool built for OpenStack.

Software Defined Storage (SDS): An architecture which consists of the storage software that is independent from the underlying storage hardware. The storage access software provides data request interfaces (APIs) and the SDS controller software provides storage access services and networking.

Tenant: cloud service users sharing access to a set of physical and virtual resources, ITU-T Y.3500. Tenants represent an independently manageable logical pool of compute, storage and network resources abstracted from physical hardware.

Tenant Instance: refers to an Instance owned by or dedicated for use by a single **Tenant**.

Tenant (Internal) Networks: Virtual networks that are internal to **Tenant Instances**.

User: Natural person, or entity acting on their behalf, associated with a cloud service customer that uses cloud services. Examples of such entities include devices and applications.

Virtual CPU (vCPU): Represents a portion of the host's computing resources allocated to a virtualised resource, for example, to a virtual machine or a container. One or more vCPUs can be assigned to a virtualised resource.

Virtualised Infrastructure Manager (VIM): Responsible for controlling and managing the Network Function Virtualisation Infrastructure (NFVI) compute, storage and network resources.

Virtual Machine (VM): virtualised computation environment that behaves like a physical computer/server. A **VM** consists of all of the components (processor (CPU), memory, storage, interfaces/ports, etc.) of a physical computer/server. It is created using sizing information or Compute Flavour.

Virtualised Network Function (VNF): A software implementation of a Network Function, capable of running on the Cloud Infrastructure. **VNFs** are built from one or more VNF Components (VNFC) and, in most cases, the VNFC is hosted on a single VM or Container.

Virtual Compute resource (a.k.a. virtualisation container): partition of a compute node that provides an isolated virtualised computation environment.

Virtual Storage resource: virtualised non-volatile storage allocated to a virtualised computation environment hosting a VNFC.

Virtual Networking resource: routes information among the network interfaces of a virtual compute resource and physical network interfaces, providing the necessary connectivity.

VMTP: A data path performance measurement tool built specifically for OpenStack clouds.

Workload: an application (for example **VNF**, or **CNF**) that performs certain task(s) for the users. In the Cloud Infrastructure, these applications run on top of compute resources such as **VMs** or **Containers**.

1.7 Abbreviations

Abbreviation/Acronym	Definition
API	Application Programming Interface
BGP VPN	Border gateway Protocol Virtual Private network
CI/CD	Continuous Integration/Continuous Deployment
CNTT	Cloud Infrastructure Task Force
CPU	Central Processing Unit
DNS	Domain Name System
DPDK	Data Plane Development Kit
DHCP	Dynamic Host Configuration Protocol
ECMP	Equal Cost Multi-Path routing
ETSI	European Telecommunications Standards Institute
FPGA	Field Programmable Gate Array
MB/GB/TB	MegaByte/GigaByte/TeraByte
GPU	Graphics Processing Unit
GRE	Generic Routing Encapsulation
GSM	Global System for Mobile Communications (originally Groupe Spécial Mobile)
GSMA	GSM Association
GSLB	Global Service Load Balancer
GUI	Graphical User Interface
HA	High Availability
HDD	Hard Disk Drive
HTTP	HyperText Transfer Protocol
HW	Hardware
IaaS (also IaC)	Infrastructure as a Code
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IMS	IP Multimedia Sub System
IO	Input/Output
IOPS	Input/Output per Second

continues on next page

Table 1.1 – continued from previous page

Abbreviation/Acronym	Definition
IPMI	Intelligent Platform Management Interface
KVM	Kernel-based Virtual Machine
LCM	LifeCycle Management
LDAP	Lightweight Directory Access Protocol
LFN	Linux Foundation Networking
LMA	Logging, Monitoring and Analytics
LVM	Logical Volume Management
MANO	Management ANd Orchestration
MLAG	Multi-chassis Link Aggregation Group
NAT	Network Address Translation
NFS	Network File System
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NIC	Network Interface Card
NPU	Numeric Processing Unit
NTP	Network Time Protocol
NUMA	Non-Uniform Memory Access
OAI	Open Air Interface
OS	Operating System
OSTK	OpenStack
OPNFV	Open Platform for NFV
OVS	Open vSwitch
OWASP	Open Web Application Security Project
PCIe	Peripheral Component Interconnect Express
PCI-PT	PCIe PassThrough
PXE	Preboot Execution Environment
QoS	Quality of Service
RA	Reference Architecture
RA-1	Reference Architecture 1 (i.e., Reference Architecture for OpenStack-based Cloud Infrastructure)
RBAC	Role-based Access Control
RBD	RADOS Block Device
REST	Representational state transfer
RI	Reference Implementation
RM	Reference Model
SAST	Static Application Security Testing
SDN	Software Defined Networking
SFC	Service Function Chaining
SG	Security Group
SLA	Service Level Agreement
SMP	Symmetric MultiProcessing
SMT	Simultaneous MultiThreading
SNAT	Source Network Address Translation
SNMP	Simple Network Management Protocol
SR-IOV	Single Root Input Output Virtualisation
SSD	Solid State Drive
SSL	Secure Sockets Layer
SUT	System Under Test
TCP	Transmission Control Protocol

continues on next page

Table 1.1 – continued from previous page

Abbreviation/Acronym	Definition
TLS	Transport Layer Security
ToR	Top of Rack
TPM	Trusted Platform Module
UDP	User Data Protocol
VIM	Virtualised Infrastructure Manager
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Function
VRRP	Virtual Router Redundancy Protocol
VTEP	VXLAN Tunnel End Point
VXLAN	Virtual Extensible LAN
WAN	Wide Area Network
ZTA	Zero Trust Architecture

1.8 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 as described in RFC 2119 [7].

2 Architecture Requirements

This chapter includes both “Requirements” that must be satisfied in an RA-1 conformant implementation and “Recommendations” that are optional for implementation.

2.1 Reference Model Requirements

The tables below contain the requirements from the Reference Model [1] to cover the Basic and High-Performance profiles.

To ensure alignment with the infrastructure profile catalogue, the following requirements are referenced through:

- Those relating to Cloud Infrastructure Software Profiles
- Those relating to Cloud Infrastructure Hardware Profiles
- Those relating to Cloud Infrastructure Management
- Those relating to Cloud Infrastructure Security

2.1.1 Cloud Infrastructure Software Profile Requirements for Compute

Table 2.1: Reference Model Requirements: Cloud Infrastructure Software Profile Capabilities

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
e.cap.001	Max number of vCPU that can be assigned to a single instance by the Cloud Infrastructure	At least 16	At least 16	<i>Compute Nodes</i>
e.cap.002	Max memory that can be assigned to a single instance by the Cloud Infrastructure	at least 32 GB	at least 32 GB	<i>Virtual Storage</i>
e.cap.003	Max storage that can be assigned to a single instance by the Cloud Infrastructure	at least 320 GB	at least 320 GB	<i>Virtual Storage and Storage Backend</i>
e.cap.004	Max number of connection points that can be assigned to a single instance by the Cloud Infrastructure	6	6	Not Detailed
e.cap.005	Max storage that can be attached / mounted to an instance by the Cloud Infrastructure	Up to 16TB [*]	Up to 16TB [*]	<i>Storage Backend</i>
e.cap.006 / infra.com.cfg.003	CPU pinning support	Not required	Must support	<i>Consumable Infrastructure Resources and Services</i>
e.cap.007 / infra.com.cfg.002	NUMA support	Not required	Must support	<i>Consumable Infrastructure Resources and Services</i>
e.cap.018 / infra.com.cfg.005	Simultaneous Multi-threading (SMT) enabled	Must	Optional support	<i>Consumable Infrastructure Resources and Services</i>
i.cap.018 / infra.com.cfg.004	Huge pages configured	Not required	Must support	<i>Consumable Infrastructure Resources and Services</i>

[*] Defined in the .bronze configuration in “Storage extensions” in [1].

2.1.2 Cloud Infrastructure Software Profile Extensions Requirements for Compute

Table 2.2: Cloud Infrastructure Software Profile Extensions Requirements for Compute

Reference	Description	Profile Ex-tensions	Profile Extra-Specs	Specification Reference
e.cap.008 / infra.com.acc.cfg.001	IPSec Acceleration using the virtio-ipsec interface	Compute Intensive GPU		<i>Acceleration</i>
e.cap.010 / infra.com.acc.cfg.002	Transcoding Acceleration	Compute Intensive GPU	Video Transcoding	<i>Acceleration</i>
e.cap.011 / infra.com.acc.cfg.003	Programmable Acceleration	Firmware-programmable adapter	Accelerator	<i>Acceleration</i>
e.cap.012	Enhanced Cache Management: L=Lean; E=Equal; X=eXpanded	E	E	Not detailed
e.cap.014 / infra.com.acc.cfg.004	Hardware coprocessor support (GPU/NPU)	Compute Intensive GPU		<i>Acceleration</i>
e.cap.016 / infra.com.acc.cfg.005	FPGA/other Acceleration H/W	Firmware-programmable adapter		<i>Acceleration</i>

2.1.3 Cloud Infrastructure Software Profile Requirements for Networking

The features and configuration requirements related to virtual networking for the two (2) types of Cloud Infrastructure Profiles are specified below followed by networking bandwidth requirements.

Table 2.3: Reference Model Requirements - Virtual Networking

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
infra.net.cfg.001	IO virtualisation using virtio1.1	Must support	Must support	<i>Virtualisation layer</i>
infra.net.cfg.002	The overlay network encapsulation protocol needs to enable ECMP in the underlay to take advantage of the scale-out features of the network fabric	Must support VXLAN, MPLSoUDP, GENEVE, other	No requirement specified	<i>Network Fabric</i>
infra.net.cfg.003	Network Address Translation	Must support	Must support	<i>Network Fabric</i>
infra.net.cfg.004	Security Groups	Must support	Must support	<i>Workload Security</i>
infra.net.cfg.005	SFC support	Not required	Must support	<i>Virtual Networking - 3rd party SDN solution</i>
infra.net.cfg.006	Traffic patterns symmetry	Must support	Must support	Not detailed

The required number of connection points to an instance is described in e.cap.004 above. The table below specifies the required bandwidth of those connection points.

Table 2.4: Reference Model Requirements - Network Interface Specifications

Reference	Description	Requirement for Basic Profile	Requirement for High Performance Profile	Specification Reference
n1, n2, n3, n4, n5, n6	1, 2, 3, 4, 5, 6 Gbps	Must support	Must support	Not detailed
n10, n20, n30, n40, n50, n60	10, 20, 30, 40, 50, 60 Gbps	Must support	Must support	Not detailed
n25, n50, n75, n100, n125, n150	25, 50, 75, 100, 125, 150 Gbps	Optional	Must support	Not detailed
n50, n100, n150, n200, n250, n300	50, 100, 150, 200, 250, 300 Gbps	Optional	Must support	Not detailed
n100, n200, n300, n400, n500, n600	100, 200, 300, 400, 500, 600 Gbps	Optional	Must support	Not detailed

2.1.4 Cloud Infrastructure Software Profile Extensions Requirements for Networking

Table 2.5: Cloud Infrastructure Software Profile Extensions Requirements for Networking

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
e.cap.013 / infra.hw.nac.cfg.004	SR-IOV over PCI-PT	N	Y	<i>Compute Nodes</i>
e.cap.019 / infra.net.acc.cfg.001	vSwitch optimisation (DPDK)	N	Y	<i>Compute Nodes and Network quality of service</i>
e.cap.015 / infra.net.acc.cfg.002	SmartNIC (for HW Offload)	N	Optional	<i>Acceleration</i>
e.cap.009 / infra.net.acc.cfg.003	Crypto acceleration	N	Optional	Not detailed
infra.net.acc.cfg.004	Crypto Acceleration Interface	N	Optional	Not detailed

2.1.5 Cloud Infrastructure Software Profile Requirements for Storage

Table 2.6: Reference Model Requirements - Cloud Infrastructure Software Profile Requirements for Storage

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
infra.stg.cfg.002	Storage Block	Must support	Must support	<i>Storage and Cinder</i>
infra.stg.cfg.003	Storage with replication	Not required	Must support	<i>Storage and Transaction Volume Considerations</i>
infra.stg.cfg.004	Storage with encryption	Must support	Must support	<i>Storage</i>
infra.stg.acc.cfg.001	Storage IOPS oriented	Not required	Must support	<i>Storage</i>
infra.stg.acc.cfg.002	Storage capacity oriented	Not required	Not required	<i>Storage</i>

2.1.6 Cloud Infrastructure Software Profile Extensions Requirements for Storage

Table 2.7: Reference Model Requirements - Cloud Infrastructure Software Profile Extensions Requirements for Storage

Reference	Description	Profile Extensions	Profile Extra-Specs	Specification Reference
infra.stg.acc.cfg.001	Storage IOPS oriented	Storage Intensive High-performance storage		Not detailed
infra.stg.acc.cfg.002	Storage capacity oriented	High Capacity		Not detailed

2.1.7 Cloud Infrastructure Hardware Profile Requirements

Table 2.8: Reference Model Requirements - Cloud Infrastructure Hardware Profile Requirements

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
infra.hw.001	CPU Architecture (Values such as x64, ARM, etc.)			
infra.hw.cpu.cfg.001	Minimum number of CPU (Sockets)	2	2	<i>Compute</i>
infra.hw.cpu.cfg.002	Minimum number of Cores per CPU	20	20	<i>Compute</i>
infra.hw.cpu.cfg.003	NUMA	Not required	Must support	<i>Compute</i>
infra.hw.cpu.cfg.004	Simultaneous Multi-threading/Symmetric Multiprocessing (SMT/SMP)	Must support	Optional	<i>Compute</i>
infra.hw.stg.hdd.cfg.001	Local Storage HDD	No requirement specified	No requirement specified	<i>Consumable Infrastructure Resources and Services</i>
infra.hw.stg.ssd.cfg.002	Local Storage SSD	Should support	Should support	<i>Consumable Infrastructure Resources and Services</i>
infra.hw.nic.cfg.001	Total Number of NIC Ports available in the host	4	4	<i>Compute</i>
infra.hw.nic.cfg.002	Port speed specified in Gbps (minimum values)	10	25	<i>Consumable Infrastructure Resources and Services</i>
infra.hw.pci.cfg.001	Number of PCIe slots available in the host	8	8	Not detailed
infra.hw.pci.cfg.002	PCIe speed	Gen 3	Gen 3	Not detailed
infra.hw.pci.cfg.003	PCIe Lanes	8	8	Not detailed
infra.hw.nac.cfg.003	Compression	No requirement specified	No requirement specified	Not detailed

Cloud Infrastructure Hardware Profile-Extensions Requirements

Table 2.9: Reference Model Requirements - Cloud Infrastructure Hardware Profile Extensions Requirements

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
e.cap.014 / infra.hw.cac.cfg.001	GPU	N	Optional	<i>Acceleration</i>
e.cap.016 / infra.hw.cac.cfg.002	FPGA/other Acceleration H/W	N	Optional	<i>Acceleration</i>
e.cap.009 / infra.hw.nac.cfg.001	Crypto Acceleration	N	Optional	<i>Acceleration</i>
e.cap.015 / infra.hw.nac.cfg.002	SmartNIC	N	Optional	<i>Acceleration</i>
infra.hw.nac.cfg.003	Compression	Optional	Optional	<i>Acceleration</i>
e.cap.013 / infra.hw.nac.cfg.004	SR-IOV over PCI-PT	N	Yes	<i>Compute node configurations for Profiles and OpenStack Flavors</i>

2.1.8 Cloud Infrastructure Management Requirements

Table 2.10: Reference Model Requirements - Cloud Infrastructure Management Requirements

Reference	Description	Requirement (common to all Profiles)	Specification Reference
e.man.001	Capability to allocate virtual compute resources to a workload	Must support	<i>Resources and Services exposed to VNFs</i>
e.man.002	Capability to allocate virtual storage resources to a workload	Must support	<i>Resources and Services exposed to VNFs</i>
e.man.003	Capability to allocate virtual networking resources to a workload	Must support	<i>Resources and Services exposed to VNFs</i>
e.man.004	Capability to isolate resources between tenants	Must support	<i>Tenant Isolation</i>
e.man.005	Capability to manage workload software images	Must support	<i>Glance</i>
e.man.006	Capability to provide information related to allocated virtualised resources per tenant	Must support	<i>Logging, Monitoring and Analytics</i>
e.man.007	Capability to notify state changes of allocated resources	Must support	<i>Logging, Monitoring and Analytics</i>
e.man.008	Capability to collect and expose performance information on virtualised resources allocated	Must support	<i>Logging, Monitoring and Analytics</i>
e.man.009	Capability to collect and notify fault information on virtualised resources	Must support	<i>Logging, Monitoring and Analytics</i>

2.1.9 Cloud Infrastructure Security Requirements

System Hardening Requirements

Table 2.11: Reference Model Requirements - System Hardening Requirements

Reference	sub-category	Description	Specification Reference
sec.gen.001	Hardening	The Platform must maintain the specified configuration	<i>Security LCM and Cloud Infrastructure provisioning and configuration management</i>
sec.gen.002	Hardening	All systems part of Cloud Infrastructure must support hardening as defined in CIS Password Policy Guide [8]	<i>Password policy</i>
sec.gen.003	Hardening	All servers part of Cloud Infrastructure must support a root of trust and secure boot	<i>Server boot hardening</i>
sec.gen.004	Hardening	The Operating Systems of all the servers part of Cloud Infrastructure must be hardened by removing or disabling unnecessary services, applications and network protocols, configuring operating system user authentication, configuring resource controls, installing and configuring additional security controls where needed, and testing the security of the Operating System (NIST SP 800-123)	<i>Function and Software</i>
sec.gen.005	Hardening	The Platform must support Operating System level access control	<i>System Access</i>
sec.gen.006	Hardening	The Platform must support Secure logging. Logging with root account must be prohibited when root privileges are not required	<i>System Access</i>
sec.gen.007	Hardening	All servers part of Cloud Infrastructure must be Time synchronised with authenticated Time service	<i>Security Logs Time Synchronisation</i>
sec.gen.008	Hardening	All servers part of Cloud Infrastructure must be regularly updated to address security vulnerabilities	<i>Security LCM</i>
sec.gen.009	Hardening	The Platform must support software integrity protection and verification	<i>Integrity of Open-Stack components configuration</i>
sec.gen.010	Hardening	The Cloud Infrastructure must support encrypted storage, for example, block, object and file storage, with access to encryption keys restricted based on a need to know (Controlled Access Based on the Need to Know [9])	<i>Confidentiality and Integrity</i>
sec.gen.012	Hardening	The Operator must ensure that only authorised actors have physical access to the underlying infrastructure	This requirement's verification must be part of the organisation's security process

continues on next page

Table 2.11 – continued from previous page

Reference	sub-category	Description	Specification Reference
sec.gen.013	Hardening	The Platform must ensure that only authorised actors have logical access to the underlying infrastructure	<i>System Access</i>
sec.gen.015	Hardening	Any change to the Platform must be logged as a security event, and the logged event must include the identity of the entity making the change, the change, the date and the time of the change	<i>Security LCM</i>

Platform and Access Requirements

Table 2.12: Reference Model Requirements - Platform and Access Requirements

Reference	sub-category	Description	Specification Reference
sec.sys.001	Access	The Platform must support authenticated and secure access to API, GUI and command line interfaces	<i>RBAC</i>
sec.sys.002	Access	The Platform must support Traffic Filtering for workloads (for example, Firewall)	<i>Workload Security</i>
sec.sys.003	Access	The Platform must support Secure and encrypted communications, and confidentiality and integrity of network	<i>Confidentiality and Integrity</i>
sec.sys.004	Access	The Cloud Infrastructure must support authentication, integrity and confidentiality on all network channels	<i>Confidentiality and Integrity</i>
sec.sys.005	Access	The Cloud Infrastructure must segregate the underlay and overlay networks	<i>Confidentiality and Integrity</i>
sec.sys.006	Access	The Cloud Infrastructure must be able to utilise the Cloud Infrastructure Manager identity lifecycle management capabilities	<i>Identity Security</i>
sec.sys.007	Access	The Platform must implement controls enforcing separation of duties and privileges, least privilege use and least common mechanism (Role-Based Access Control)	<i>RBAC</i>
sec.sys.008	Access	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	<i>Workload Security</i>
sec.sys.009	Access	The Platform must support creation of Trust Relationships between trust domains. These maybe 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 domain are “trusting” and “trusted”	<i>Logical segregation and high availability</i>

continues on next page

Table 2.12 – continued from previous page

Reference	sub-category	Description	Specification Reference
sec.sys.010	Access	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	<i>Logical segregation and high availability</i>
sec.sys.011	Access	The Platform must not reuse the same authentication credentials (e.g., key pairs) on different Platform components (e.g., different hosts, or different services)	<i>System Access</i>
sec.sys.012	Access	The Platform must protect all secrets by using strong encryption techniques and storing the protected secrets externally from the component (e.g., in OpenStack Barbican)	<i>Barbican</i>
sec.sys.013	Access	The Platform must generate secrets dynamically as and when needed	<i>Barbican</i>
sec.sys.015	Access	The Platform must not contain back door entries (unpublished access points, APIs, etc.)	Not detailed
sec.sys.016	Access	Login access to the Platform's components must be through encrypted protocols such as SSH v2 or TLS v1.2 or higher. Note: Hardened jump servers isolated from external networks are recommended	<i>Security LCM</i>
sec.sys.017	Access	The Platform must provide the capability of using digital certificates that comply with X.509 standards issued by a trusted Certification Authority	<i>Confidentiality and Integrity</i>
sec.sys.018	Access	The Platform must provide the capability of allowing certificate renewal and revocation	<i>Confidentiality and Integrity</i>
sec.sys.019	Access	The Platform must provide the capability of testing the validity of a digital certificate (CA signature, validity period, non revocation identity)	<i>Confidentiality and Integrity</i>

Confidentiality and Integrity Requirements

Table 2.13: Reference Model Requirements - Confidentiality and Integrity Requirements

Reference	sub-category	Description	Specification Reference
sec.ci.001	Confidentiality / Integrity	The Platform must support Confidentiality and Integrity of data at rest and in transit	<i>Confidentiality and Integrity</i>
sec.ci.003	Confidentiality / Integrity	The Platform must support Confidentiality and Integrity of data related metadata	<i>Confidentiality and Integrity</i>
sec.ci.004	Confidentiality	The Platform must support Confidentiality of processes and restrict information sharing with only the process owner (e.g., tenant)	<i>Confidentiality and Integrity</i>
sec.ci.005	Confidentiality / Integrity	The Platform must support Confidentiality and Integrity of process- related metadata and restrict information sharing with only the process owner (e.g., tenant)	<i>Confidentiality and Integrity</i>
sec.ci.006	Confidentiality / Integrity	The Platform must support Confidentiality and Integrity of workload resource utilisation (RAM, CPU, Storage, Network I/O, cache, hardware offload) and restrict information sharing with only the workload owner (e.g., tenant)	<i>Platform Access</i>
sec.ci.007	Confidentiality / Integrity	The Platform must not allow Memory Inspection by any actor other than the authorised actors for the Entity to which Memory is assigned (e.g., tenants owning the workload), for Lawful Inspection, and for secure monitoring services. Administrative access must be managed using Platform Identity Lifecycle Management	<i>Platform Access</i>
sec.ci.008	Confidentiality	The Cloud Infrastructure must support tenant networks segregation	<i>Workload Security</i>

Workload Security Requirements

Table 2.14: Reference Model Requirements - Workload Security Requirements

Reference	sub-category	Description	Specification Reference
sec.wl.001	Workload	The Platform must support Workload placement policy	<i>Workload Security</i>
sec.wl.002	Workload	The Cloud Infrastructure must provide methods to ensure the platform's trust status and integrity (e.g., remote attestation, Trusted Platform Module)	<i>Cloud Infrastructure and VIM Security</i>
sec.wl.003	Workload	The Platform must support secure provisioning of Workloads	<i>Workload Security</i>
sec.wl.004	Workload	The Platform must support Location assertion (for mandated in- country or location requirements)	<i>Workload Security</i>
sec.wl.005	Workload	The Platform must support the separation of production and non- production Workloads	<i>Workload Security</i>
sec.wl.006	Workload	The Platform must support the separation of Workloads based on their categorisation (for example, payment card information, healthcare, etc.)	<i>Workload Security</i>
sec.wl.007	Workload	The Operator must implement processes and tools to verify NF authenticity and integrity	<i>Image Security</i>

Image Security Requirements

Table 2.15: Reference Model Requirements - Image Security Requirements

Reference	sub-category	Description	Specification Reference
sec.img.001	Image	Images from untrusted sources must not be used	<i>Image Security</i>
sec.img.002	Image	Images must be scanned to be maintained free from known vulnerabilities	<i>Image Security</i>
sec.img.003	Image	Images must not be configured to run with privileges higher than the privileges of the actor authorised to run them	<i>Image Security</i>
sec.img.004	Image	Images must only be accessible to authorised actors	<i>Integrity of Open-Stack components configuration</i>
sec.img.005	Image	Image Registries must only be accessible to authorised actors	<i>Integrity of Open-Stack components configuration</i>
sec.img.006	Image	Image Registries must only be accessible over networks that enforce authentication, integrity and confidentiality	<i>Integrity of Open-Stack components configuration</i>
sec.img.007	Image	Image registries must be clear of vulnerable and out of date versions	<i>Image Security</i>
sec.img.008	Image	Images must not include any secrets. Secrets include passwords, cloud provider credentials, SSH keys, TLS certificate keys, etc.	<i>Image Security</i>

Security LCM Requirements

Table 2.16: Reference Model Requirements - Security LCM Requirements

Reference	sub-category	Description	Specification Reference
sec.lcm.001	LCM	The Platform must support Secure Provisioning, Availability, and Deprovisioning (Secure Clean-Up) of workload resources where Secure Clean-Up includes tear-down, defense against virus or other attacks	<i>Monitoring and Security Audit</i>
sec.lcm.002	LCM	The Cloud Operator must use management protocols limiting security risk such as SNMPv3, SSH v2, ICMP, NTP, syslog and TLS v1.2 or higher	<i>Security LCM</i>
sec.lcm.003	LCM	The Cloud Operator must implement and strictly follow change management processes for Cloud Infrastructure, Infrastructure Manager and other-components of the cloud, and Platform change control on hardware	<i>Monitoring and Security Audit</i>
sec.lcm.005	LCM	Platform must provide logs and these logs must be monitored for anomalous behaviour	<i>Monitoring and Security Audit</i>
sec.lcm.006	LCM	The Platform must verify the integrity of all Resource management requests	<i>Confidentiality and Integrity of tenant data (sec.ci.001)</i>
sec.lcm.007	LCM	The Platform must be able to update newly instantiated, suspended, hibernated, migrated and restarted images with current time information	Not detailed
sec.lcm.008	LCM	The Platform must be able to update newly instantiated, suspended, hibernated, migrated and restarted images with relevant DNS information	Not detailed
sec.lcm.009	LCM	The Platform must be able to update the tag of newly instantiated, suspended, hibernated, migrated and restarted images with relevant geolocation (geographical) information	Not detailed
sec.lcm.010	LCM	The Platform must log all changes to geolocation along with the mechanisms and sources of location information (i.e. GPS, IP block, and timing)	Not detailed
sec.lcm.011	LCM	The Platform must implement Security life cycle management processes including the proactive update and patching of all deployed Cloud Infrastructure software	<i>Patches</i>
sec.lcm.012	LCM	The Platform must log any access privilege escalation	<i>What to Log / What NOT to Log</i>

Monitoring and Security Audit Requirements

The Platform is assumed to provide configurable alerting and notification capability and the operator is assumed to have automated systems, policies and procedures 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.

Table 2.17: Reference Model Requirements - Monitoring and Security Audit Requirements

Reference	sub-category	Description	Specification Reference
sec.mon.001	Monitoring / Audit	Platform must provide logs and these logs must be regularly monitored for events of interest. The logs must contain the following fields: event type, date/time, protocol, service or program used for access, success/failure, login ID or process ID, IP address and ports (source and destination) involved	<i>Required Fields</i>
sec.mon.002	Monitoring	Security logs must be time synchronised	<i>Security Logs Time Synchronisation</i>
sec.mon.003	Monitoring	The Platform must log all changes to time server source, time, date and time zones	<i>Security Logs Time Synchronisation</i>
sec.mon.004	Audit	The Platform must secure and protect Audit logs (containing sensitive information) both in-transit and at rest	<i>Security LCM</i>
sec.mon.005	Monitoring / Audit	The Platform must Monitor and Audit various behaviours of connection and login attempts to detect access attacks and potential access attempts and take corrective accordingly actions	<i>What to Log / What NOT to Log</i>
sec.mon.006	Monitoring / Audit	The Platform must Monitor and Audit operations by authorised account access after login to detect malicious operational activity and take corrective actions	<i>Monitoring and Security Audit</i>
sec.mon.007	Monitoring / Audit	The Platform must Monitor and Audit security parameter configurations for compliance with defined security policies	<i>Integrity of Open-Stack components configuration</i>
sec.mon.008	Monitoring / Audit	The Platform must Monitor and Audit externally exposed interfaces for illegal access (attacks) and take corrective security hardening measures	<i>Confidentiality and Integrity of communications (sec.ci.001)</i>
sec.mon.009	Monitoring / Audit	The Platform must Monitor and Audit service for various attacks (malformed messages, signalling flooding and replaying, etc.) and take corrective actions accordingly	<i>Monitoring and Security Audit</i>
sec.mon.010	Monitoring / Audit	The Platform must Monitor and Audit running processes to detect unexpected or unauthorised processes and take corrective actions accordingly	<i>Monitoring and Security Audit</i>
sec.mon.011	Monitoring / Audit	The Platform must Monitor and Audit logs from infrastructure elements and workloads to detected anomalies in the system components and take corrective actions accordingly	<i>Creating Logs</i>

continues on next page

Table 2.17 – continued from previous page

Reference	sub-category	Description	Specification Reference
sec.mon.012	Monitoring / Audit	The Platform must Monitor and Audit Traffic patterns and volumes to prevent malware download attempts	<i>Confidentiality and Integrity</i>
sec.mon.013	Monitoring	The monitoring system must not affect the security (integrity and confidentiality) of the infrastructure, workloads, or the user data (through back door entries)	Not detailed
sec.mon.015	Monitoring	The Platform must ensure that the Monitoring systems are never starved of resources and must activate alarms when resource utilisation exceeds a configurable threshold	<i>Monitoring and Security Audit</i>
sec.mon.017	Audit	The Platform must audit systems for any missing security patches and take appropriate actions	<i>Patches</i>
sec.mon.018	Monitoring	The Platform, starting from initialisation, must collect and analyse logs to identify security events, and store these events in an external system	<i>Where to Log</i>
sec.mon.019	Monitoring	The Platform's components must not include an authentication credential, e.g., password, in any logs, even if encrypted	<i>What to Log / What NOT to Log</i>
sec.mon.020	Monitoring / Audit	The Platform's logging system must support the storage of security audit logs for a configurable period of time	<i>Data Retention</i>
sec.mon.021	Monitoring	The Platform must store security events locally if the external logging system is unavailable and shall periodically attempt to send these to the external logging system until successful	<i>Where to Log</i>

Open-Source Software Security Requirements

Table 2.18: Reference Model Requirements - Open-Source Software Security Requirements

Reference	sub-category	Description	Specification Reference
sec.oss.001	Software	Open-source code must be inspected by tools with various capabilities for static and dynamic code analysis	<i>Image Security</i>
sec.oss.002	Software	The CVE (Common Vulnerabilities and Exposures) must be used to identify vulnerabilities and their severity rating for open-source code part of Cloud Infrastructure and workloads software	<i>Patches</i>
sec.oss.003	Software	Critical and high severity rated vulnerabilities must be fixed in a timely manner. Refer to the CVSS (Common Vulnerability Scoring System) to know a vulnerability score and its associated rate (low, medium, high, or critical)	<i>Patches</i>
sec.oss.004	Software	A dedicated internal isolated repository separated from the production environment must be used to store vetted open-source content	<i>Workload Security</i>

IaaS security Requirements

Secure Code Stage Requirements

Table 2.19: Reference Model Requirements: IaaS Security Requirements, Secure Code Stage

Reference	sub-category	Description	Specification Reference
sec.code.001	IaaS	SAST -Static Application Security Testing must be applied during Secure Coding stage triggered by Pull, Clone or Commit trigger. Security testing that analyses application source code for software vulnerabilities and gaps against bestpractices. Example: open source OWASP range of tools	<i>Workload Security</i>

Continuous Build, Integration and Testing Stage Requirements

Table 2.20: Reference Model Requirements - IaaS Security Requirements, Continuous Build, Integration and Testing Stage

Reference	sub-category	Description	Specification Reference
sec.bld.003	IaaS	Image Scan must be applied during the Continuous Build, Integration and Testing stage triggered by Package trigger, example: A push of a container image to a containerregistry may trigger a vulnerability scan before the image becomes available in the registry	<i>Image Security</i>

Continuous Delivery and Deployment Stage Requirements

Table 2.21: Reference Model Requirements - IaaS Security Requirements, Continuous Delivery and Deployment Stage

Reference	sub-category	Description	Specification Reference
sec.del.001	IaaS	Image Scan must be applied during the Continuous Delivery and Deployment stage triggered by Publish to Artifact and Image Repository trigger. Example: GitLab uses the open source Clair engine for container image scanning	<i>Image Security</i>
sec.del.002	IaaS	Code Signing must be applied during the Continuous Delivery and Deployment stage and Image Repository trigger. Code Signing provides authentication to assure that downloaded files are from the publisher named on the certificate	<i>Image Security</i>
sec.del.004	IaaS	Component Vulnerability Scan must be applied during the Continuous Delivery and Deployment stage triggered by Instantiate Infrastructure trigger. The vulnerability scanning system is deployed on the cloud platform to detect security vulnerabilities of specified components through scanning and to provide timely security protection. Example: OWASP Zed Attack Proxy (ZAP)	<i>Image Security</i>

Runtime Defence and Monitoring Requirements

Table 2.22: Reference Model Requirements - IaaS Security Requirements, Runtime Defence and Monitoring Stage

Reference	sub-category	Description	Specification Reference
sec.run.001	IaaS	Component Vulnerability Monitoring must be continuously applied during the Runtime Defence and monitoring stage. Security technology that monitors components like virtual servers and assesses data, applications, and infrastructure for security risks	Not detailed

Compliance with Standards Requirements

Table 2.23: Reference Model Requirements: Compliance with Standards

Reference	sub-category	Description	Specification Reference
sec.std.012	Standards	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	Not detailed

2.2 Architecture and OpenStack Requirements

“Architecture” in this chapter refers to Cloud Infrastructure (referred to as NFVI by ETSI) and VIM, as specified in Reference Model Chapter 3.

2.2.1 General Requirements

Table 2.24: General Requirements

Reference	sub-category	Description	Specification Reference
gen.ost.01	Open source	The Architecture must use OpenStack APIs	<i>Consolidated Set of APIs</i>
gen.ost.02	Open source	The Architecture must support dynamic request and configuration of virtual resources (compute, network, storage) through OpenStack APIs	<i>Consolidated Set of APIs</i>
gen.rsl.01	Resiliency	The Architecture must support resilient OpenStack components that are required for the continued availability of running workloads	<i>Containerised OpenStack Services</i>
gen.avl.01	Availability	The Architecture must provide High Availability for OpenStack components	<i>Underlying Resources Configuration and Dimensioning</i>

2.2.2 Infrastructure Requirements

Table 2.25: Infrastructure Requirements

Reference	sub-category	Description	Specification Reference
inf.com.01	Compute	The Architecture must provide compute resources for instances	<i>Cloud Workload Services</i>
inf.com.04	Compute	The Architecture must be able to support multiple CPU type options to support various infrastructure profiles (Basic and High Performance)	<i>Support for Cloud Infrastructure Profiles and flavors</i>
inf.com.05	Compute	The Architecture must support Hardware Platforms with NUMA capabilities	<i>Support for Cloud Infrastructure Profiles and flavors</i>
inf.com.06	Compute	The Architecture must support CPU Pinning of the vCPUs of an instance	<i>Support for Cloud Infrastructure Profiles and flavors</i>
inf.com.07	Compute	The Architecture must support different hardware configurations to support various infrastructure profiles (Basic and High Performance)	<i>Cloud partitioning: Host Aggregates, Availability Zones</i>

continues on next page

Table 2.25 – continued from previous page

Reference	sub-category	Description	Specification Reference
inf.com.08	Compute	The Architecture must support allocating certain number of host cores for all non-tenant workloads such as for OpenStack services. SMT threads can be allocated to individual OpenStack services or their components. Dedicating host cores to certain workloads (e.g., OpenStack services) [10]. Please see example, Configuring libvirt compute nodes for CPU pinning [11]	<i>Cloud partitioning: Host Aggregates, Availability Zones</i>
inf.com.09	Compute	The Architecture must ensure that the host cores assigned to non-tenant and tenant workloads are SMT aware: that is, a host core and its associated SMT threads are either all assigned to non-tenant workloads or all assigned to tenant workloads	<i>Pinned and Un-pinned CPUs</i>
inf.stg.01	Storage	The Architecture must provide remote (not directly attached to the host) Block storage for Instances	<i>Storage</i>
inf.stg.02	Storage	The Architecture must provide Object storage for Instances. Operators may choose not to implement Object Storage but must be cognizant of the the risk of “Compliant VNFs” failing in their environment	<i>Swift</i>
inf.nw.01	Network	The Architecture must provide virtual network interfaces to instances	<i>Neutron API</i>
inf.nw.02	Network	The Architecture must include capabilities for integrating SDN controllers to support provisioning of network services, from the SDN OpenStack Neutron service, such as networking of VTEPs to the Border Edge based VRFs	<i>Virtual Networking - 3rd party SDN solution</i>
inf.nw.03	Network	The Architecture must support low latency and high throughput traffic needs	<i>Network Fabric</i>
inf.nw.05	Network	The Architecture must allow for East/West tenant traffic within the cloud (via tunnelled encapsulation overlay such as VXLAN or Geneve)	<i>Network Fabric</i>
inf.nw.07	Network	The Architecture must support network resiliency	<i>Network</i>
inf.nw.10	Network	The Cloud Infrastructure Network Fabric must be capable of enabling highly available (Five 9’s or better) Cloud Infrastructure	<i>Network</i>
inf.nw.15	Network	The Architecture must support multiple networking options for Cloud Infrastructure to support various infrastructure profiles (Basic and High Performance)	<i>Neutron Extensions and OpenStack Neutron Plugins [12]</i>
inf.nw.16	Network	The Architecture must support dual stack IPv4 and IPv6 for tenant networks and workloads	Not detailed

2.2.3 VIM Requirements

Table 2.26: VIM Requirements

Reference	sub-category	Description	Specification Reference
vim.01	General	The Architecture must allow infrastructure resource sharing	<i>Resources and Services exposed to VNFs</i>
vim.03	General	The Architecture must allow VIM to discover and manage Cloud Infrastructure resources	<i>Placement API</i>
vim.05	General	The Architecture must include image repository management	<i>Glance API</i>
vim.07	General	The Architecture must support multi-tenancy	<i>Multi-Tenancy (execution environment)</i>
vim.08	General	The Architecture must support resource tagging	OpenStack Resource Tags [13]

2.2.4 Interfaces & APIs Requirements

Table 2.27: Interfaces and APIs Requirements

Reference	sub-category	Description	Specification Reference
int.api.01	API	The Architecture must provide APIs to access the authentication service and the associated mandatory features detailed in chapter 5	<i>Keystone API</i>
int.api.02	API	The Architecture must provide APIs to access the image management service and the associated mandatory features detailed in chapter 5	<i>Glance API</i>
int.api.03	API	The Architecture must provide APIs to access the block storage management service and the associated mandatory features detailed in chapter 5	<i>Cinder API</i>
int.api.04	API	The Architecture must provide APIs to access the object storage management service and the associated mandatory features detailed in chapter 5	<i>Swift API</i>
int.api.05	API	The Architecture must provide APIs to access the network management service and the associated mandatory features detailed in chapter 5	<i>Neutron API</i>
int.api.06	API	The Architecture must provide APIs to access the compute resources management service and the associated mandatory features detailed in chapter 5	<i>Nova API</i>
int.api.07	API	The Architecture must provide GUI access to tenant facing cloud platform core services except at Edge/Far Edge clouds	<i>Horizon</i>
int.api.08	API	The Architecture must provide APIs needed to discover and manage Cloud Infrastructure resources	<i>Placement API</i>
int.api.09	API	The Architecture must provide APIs to access the orchestration service	<i>Heat API</i>
int.api.10	API	The Architecture must expose the latest version and microversion of the APIs for the given Anuket OpenStack release for each of the OpenStack core services	<i>Core OpenStack Services APIs</i>

2.2.5 Tenant Requirements

Table 2.28: Tenant Requirements

Reference	sub-category	Description	Specification Reference
tnt.gen.01	General	The Architecture must support self-service dashboard (GUI) and APIs for users to deploy, configure and manage their workloads	<i>Horizon and Cloud Workload Services</i>

2.2.6 Operations and LCM

Table 2.29: LCM Requirements

Reference	sub-category	Description	Specification Reference
lcm.gen.01	General	The Architecture must support zero downtime of running workloads when the number of compute hosts and/or the storage capacity is being expanded or unused capacity is being removed	Not detailed
lcm.adp.02	Automated deployment	The Architecture must support upgrades of software, provided by the cloud provider, so that the running workloads are not impacted (viz., hitless upgrades). Please note that this means that the existing data plane services should not fail (go down)	<i>Containerised OpenStack Services</i>

2.2.7 Assurance Requirements

Table 2.30: Assurance Requirements

Reference	sub-category	Description	Specification Reference
asr.mon.01	Integration	The Architecture must include integration with various infrastructure components to support collection of telemetry for assurance monitoring and network intelligence	<i>Logging, Monitoring and Analytics</i>
asr.mon.03	Monitoring	The Architecture must allow for the collection and dissemination of performance and fault information	<i>Logging, Monitoring and Analytics</i>
asr.mon.04	Network	The Cloud Infrastructure Network Fabric and Network Operating System must provide network operational visibility through alarming and streaming telemetry services for operational management, engineering planning, troubleshooting, and network performance optimisation	<i>Logging, Monitoring and Analytics</i>

2.2.8 Architecture and OpenStack Recommendations

The requirements listed in this section are optional, and are not required in order to be deemed a conformant implementation.

2.2.9 General Recommendations

Table 2.31: General Recommendations

Reference	sub-category	Description	Notes
gen.cnt.01	Cloud nativeness	The Architecture should consist of stateless service components. However, where state is required it must be kept external to the component	OpenStack consists of both stateless and stateful services where the stateful services utilise a database. For latter see Configuring the stateful services [14]
gen.cnt.02	Cloud nativeness	The Architecture should consist of service components implemented as microservices that are individually dynamically scalable	
gen.scl.01	Scalability	The Architecture should support policy driven auto-scaling.	This requirement is currently not addressed but will likely be supported through Senlin [15], cluste management service
gen.rsl.02	Resiliency	The Architecture should support resilient Open-Stack service components that are not subject to gen.rsl.01	

2.2.10 Infrastructure Recommendations

Table 2.32: Infrastructure Recommendations

Reference	sub-category	Description	Notes
inf.com.02	Compute	The Architecture should include industry standard hardware management systems at both HW device level (embedded) and HW platform level (external to device)	
inf.com.03	Compute	The Architecture should support Symmetric Multiprocessing with shared memory access as well as Simultaneous Multithreading	
inf.stg.08	Storage	The Architecture should allow use of externally provided large archival storage for its Backup / Restore / Archival needs	
inf.stg.09	Storage	The Architecture should make available all non-host OS / Hypervisor / Host systems storage as network-based Block, File or Object Storage for tenant/management consumption	
inf.stg.10	Storage	The Architecture should provide local Block storage for Instances	<i>Virtual Storage</i>
inf.nw.04	Network	The Architecture should support service function chaining	

continues on next page

Table 2.32 – continued from previous page

Reference	sub-category	Description	Notes
inf.nw.06	Network	The Architecture should support Distributed Virtual Routing (DVR) to allow compute nodes to route traffic efficiently	
inf.nw.08	Network	The Cloud Infrastructure Network Fabric should embrace the concepts of open networking and disaggregation using commodity networking hardware and disaggregated Network Operating Systems	
inf.nw.09	Network	The Cloud Infrastructure Network Fabric should embrace open-based standards and technologies	
inf.nw.11	Network	The Cloud Infrastructure Network Fabric should be architected to provide a standardised, scalable, and repeatable deployment model across all applicable Cloud Infrastructure sites	
inf.nw.17	Network	The Architecture should use dual stack IPv4 and IPv6 for Cloud Infrastructure internal networks	
inf.acc.01	Acceleration	The Architecture should support Application Specific Acceleration (exposed to VNFs)	<i>Acceleration</i>
inf.acc.02	Acceleration	The Architecture should support Cloud Infrastructure Acceleration (such as SmartNICs)	OpenStack Future - Specs defined [16]
inf.acc.03	Acceleration	The Architecture may rely on on SR-IOV PCI-Pass through to provide acceleration to VNFs	
inf.img.01	Image	The Architecture should make the immutable images available via location independent means	<i>Glance</i>

2.2.11 VIM Recommendations

Table 2.33: VIM Recommendations

Reference	sub-category	Description	Notes
vim.02	General	The Architecture should support deployment of OpenStack components in containers	<i>Containerised OpenStack Services</i>
vim.04	General	The Architecture should support Enhanced Platform Awareness (EPA) only for discovery of infrastructure resource capabilities	
vim.06	General	The Architecture should allow orchestration solutions to be integrated with VIM	
vim.09	General	The Architecture should support horizontal scaling of OpenStack core services	

2.2.12 Interfaces and APIs Recommendations

Table 2.34: Interfaces and APIs Recommendations

Reference	sub-category	Description	Notes
int.acc.01	Acceleration	The Architecture should provide an open and standard acceleration interface to VNFs	
int.acc.02	Acceleration	The Architecture should not rely on SR-IOV PCI-Pass through for acceleration interface exposed to VNFs	duplicate of inf.acc.03 under “Infrastructure Recommendation”

2.2.13 Tenant Recommendations

This section is left blank for future use.

Table 2.35: Tenant Recommendations

Reference	sub-category	Description	Notes

2.2.14 Operations and LCM Recommendations

Table 2.36: LCM Recommendations

Reference	sub-category	Description	Notes
lcm.adp.01	Automated deployment	The Architecture should allow for cookie cutter automated deployment, configuration, provisioning and management of multiple Cloud Infrastructure sites	
lcm.adp.03	Automated deployment	The Architecture should support hitless upgrade of all software provided by the cloud provider that are not covered by lcm.adp.02. Whenever hitless upgrades are not feasible, attempt should be made to minimise the duration and nature of impact	
lcm.adp.04	Automated deployment	The Architecture should support declarative specifications of hardware and software assets for automated deployment, configuration, maintenance and management	
lcm.adp.05	Automated deployment	The Architecture should support automated process for Deployment and life-cycle management of VIM Instances	
lcm.cid.02	CI/CD	The Architecture should support integrating with CI/CD Toolchain for Cloud Infrastructure and VIM components Automation	

2.2.15 Assurance Recommendations

Table 2.37: Assurance Recommendations

Reference	sub-category	Description	Notes
asr.mon.02	Monitoring	The Architecture should support Network Intelligence capabilities that allow richer diagnostic capabilities which take as input broader set of data across the network and from VNF workloads	

2.2.16 Security Recommendations

System Hardening Recommendations

Table 2.38: System Hardening Recommendations

Reference	sub-category	Description	Notes
sec.gen.011	Hardening	The Cloud Infrastructure should support Read and Write only storage partitions (write only permission to one or more authorised actors)	
sec.gen.014	Hardening	All servers part of Cloud Infrastructure should support measured boot and an attestation server that monitors the measurements of the servers	

Platform and Access Recommendations

Table 2.39: Platform and Access Recommendations

Reference	sub-category	Description	Notes
sec.sys.014	Access	The Platform should use Linux Security Modules such as SELinux to control access to resources	
sec.sys.020	Access	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

Confidentiality and Integrity Recommendations

Table 2.40: Confidentiality and Integrity Recommendations

Reference	sub-category	Description	Notes
sec.ci.002	Confidentiality / Integrity	The Platform should support self-encrypting storage devices	
sec.ci.009	Confidentiality / Integrity	For sensitive data encryption, the key management service should leverage a Hardware Security Module to manage and protect cryptographic keys	

Workload Security Recommendations

Table 2.41: Workload Security Recommendations

Reference	sub-category	Description	Notes
sec.wl.007	Workload	The Operator should implement processes and tools to verify VNF authenticity and integrity	

Image Security Recommendations

Table 2.42: Image Security Recommendations

Reference	sub-category	Description	Notes
sec.img.009	Image	CIS Hardened Images should be used whenever possible	
sec.img.010	Image	Minimalist base images should be used whenever possible	

Security LCM Recommendations

Table 2.43: LCM Security Recommendations

Reference	sub-category	Description	Notes
sec.lcm.004	LCM	The Cloud Operator should support automated templated approved changes; Templated approved changes for automation where available	

Monitoring and Security Audit Recommendations

The Platform is assumed to provide configurable alerting and notification capability and the operator is assumed to have automated systems, policies and procedures 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.

Table 2.44: Monitoring and Security Audit Recommendations

Reference	sub-category	Description	Notes
sec.mon.014	Monitoring	The Monitoring systems should not impact IaaS, PaaS, and SaaS SLAs including availability SLAs	
sec.mon.016	Monitoring	The Platform Monitoring components should follow security best practices for auditing, including secure logging and tracing	

Open-Source Software Security Recommendations

Table 2.45: Open-Source Software Security Recommendations

Reference	sub-category	Description	Notes
sec.oss.005	Software	A Software Bill of Materials (SBOM) should be provided or build, and maintained to identify the software components and their origins. Inventory of software components	NTIA SBOM [17]

IaaS security Recommendations

Secure Design and Architecture Stage

Table 2.46: Reference Model Requirements: IaaS Security, Design and Architecture Stage

Reference	sub-category	Description	Notes
sec.arch.001	IaaS	Threat Modelling methodologies and tools should be used during the Secure Design and Architecture stage triggered by Software Feature Design trigger. Methodology to identify and understand threats impacting a resource or set of resources	It may be done manually or using tools like open source OWASP Threat Dragon
sec.arch.002	IaaS	Security Control Baseline Assessment should be performed during the Secure Design and Architecture stage triggered by Software Feature Design trigger	Typically done manually by internal or independent assessors

Secure Code Stage Recommendations

Table 2.47: Reference Model Requirements: IaaS Security, Secure Code Stage

Reference	sub-category	Description	Notes
sec.code.002	IaaS	SCA - Software Composition Analysis should be applied during Secure Coding stage triggered by Pull, Clone or Comment trigger. Security testing that analyses application source code or compiled code for software components with known vulnerabilities	Example: open source OWASP range of tools
sec.code.003	IaaS	Source Code Review should be performed continuously during Secure Coding stage.	Typically done manually.
sec.code.004	IaaS	Integrated SAST via IDE Plugins should be used during Secure Coding stage triggered by Developer Code trigger. On the local machine: through the IDE or integrated test suites; triggered on completion of coding by developer	
sec.code.005	IaaS	SAST of Source Code Repo should be performed during Secure Coding stage triggered by Developer Code trigger. Continuous delivery pre - deployment: scanning prior to deployment	

Continuous Build, Integration and Testing Stage Recommendations

Table 2.48: Reference Model Requirements: IaaC Security, Continuous Build, Integration and Testing Stage

Reference	sub-category	Description	Notes
sec.bld.001	IaaC	SAST -Static Application Security Testing should be applied during the Continuous Build, Integration and Testing stage triggered by Build and Integrate trigger	Example: open source OWASP range of tools.
sec.bld.002	IaaC	SCA - Software Composition Analysis should be applied during the Continuous Build, Integration and Testing stage triggered by Build and Integrate trigger	Example: open source OWASP range of tools
sec.bld.004	IaaC	SDAST - Dynamic Application Security Testing should be applied during the Continuous Build, Integration and Testing stage triggered by Stage & Test trigger. Security testing that analyses a running application by exercising application functionality and detecting vulnerabilities based on application behaviour and response	Example: OWASP ZAP
sec.bld.005	IaaC	Fuzzing should be applied during the Continuous Build, Integration and testing stage triggered by Stage & Test trigger. Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program	Example: GitLab Open Sources Protocol Fuzzer Community Edition
sec.bld.006	IaaC	IAST - Interactive Application Security Testing should be applied during the Continuous Build, Integration and Testing stage triggered by Stage & Test trigger. Software component deployed with an application that assesses application behaviour and detects presence of vulnerabilities on an application being exercised in realistic testing scenarios	Example: Contrast Community Edition

Continuous Delivery and Deployment Stage Recommendations

Table 2.49: Reference Model Requirements: IaaC Security, Continuous Delivery and Deployment Stage

Reference	sub-category	Description	Notes
sec.del.003	IaaC	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

Runtime Defence and Monitoring Recommendations

Table 2.50: Reference Model Requirements: Iaac Security, Runtime Defence and Monitoring Stage

Reference	sub-category	Description	Notes
sec.run.002	IaaS	RASP - Runtime Application Self-Protection 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	
sec.run.003	IaaS	Application testing and Fuzzing should be continuously applied during the Runtime Defence and Monitoring stage. Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program	Example: GitLab Open Sources Protocol Fuzzer Community Edition
sec.run.004	IaaS	Penetration Testing should be continuously applied during the Runtime Defence and Monitoring stage	Typically done manually

Compliance with Standards Recommendations

Table 2.51: Compliance with Security Recommendations

Reference	sub-category	Description	Notes
sec.std.001	Standards	The Cloud Operator should comply with Center for Internet Security CIS Controls [18]	
sec.std.002	Standards	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)- CSA, Cloud Security Alliance [19]	
sec.std.003	Standards	The Platform and Workloads should follow the guidance in the OWASP Cheat Sheet Series (OCSS) [20] - OWASP, Open Web Application Security Project [21]	
sec.std.004	Standards	The Cloud Operator, Platform and Workloads should ensure that their code is not vulnerable to the OWASP Top Ten Security Risks [22]	
sec.std.005	Standards	The Cloud Operator, Platform and Workloads should strive to improve their maturity on the OWASP Software Maturity Model (SAMM) [23]	
sec.std.006	Standards	The Cloud Operator, Platform and Workloads should utilise the OWASP Web Security Testing Guide [24]	
sec.std.007	Standards	The Cloud Operator, and Platform should satisfy the requirements for Information Management Systems specified in ISO/IEC 27001 [25]; ISO/IEC 27001 is the international Standard for best-practice information security management systems (ISMSs)	

continues on next page

Table 2.51 – continued from previous page

Reference	sub-category	Description	Notes
sec.std.008	Standards	The Cloud Operator, and Platform should implement the Code of practice for Security Controls specified ISO/IEC 27002:2013 (or latest) [26]	
sec.std.009	Standards	The Cloud Operator, and Platform should implement the ISO/IEC 27032:2012 (or latest) Guidelines for Cybersecurity techniques [27]; ISO/IEC 27032 is the international Standard focusing explicitly on cybersecurity	
sec.std.010	Standards	The Cloud Operator should conform to the ISO/IEC 27035 standard for incidence management; ISO/IEC 27035 is the international Standard for incident management	
sec.std.011	Standards	The Cloud Operator should conform to the ISO/IEC 27031 standard for business continuity; ISO/IEC 27031 - ISO/IEC 27031 is the international Standard for ICT readiness for business continuity	

3 Cloud Infrastructure Architecture - OpenStack

This Reference Architecture aims to provide an OpenStack distribution agnostic reference architecture that includes the Network Function Virtualisation Infrastructure (NFVI) and Virtual Infrastructure Manager (VIM). The different OpenStack distributions, without the not up-streamed vendor specific enhancements, are assumed to be Anuket conformant. This Reference Architecture allows operators to provide a common OpenStack-based architecture for any Anuket compliant VNF to be deployed and operated as expected. The purpose of this chapter is to outline all the components required to provide the Cloud Infrastructure (NFVI and the VIM) in a consistent and reliable way.

OpenStack [2] is already very well documented and, hence, this document will describe the specific OpenStack services and features, Cloud Infrastructure features and how we expect them to be implemented.

This reference architecture provides optionality in terms of pluggable components such as SDN, hardware acceleration and support tools.

The Cloud Infrastructure layer includes the physical infrastructure which is then offered as virtual resources via a hypervisor. The VIM is the OpenStack Wallaby release.

This chapter is organised as follows:

- Consumable Infrastructure Resources and Services: these are infrastructure services and resources being exposed northbound for consumption
 - Multi-tenancy with quotas
 - * Virtual compute: vCPU / vRAM
 - * Virtual storage: Ephemeral, Persistent and Image
 - * Virtual networking - neutron standalone: network plugin, virtual switch, accelerator features
 - * Virtual networking - 3rd party SDN solution
 - * Additional network services: Firewall, DC Gateway
- Cloud Infrastructure Management Software (VIM): is how we manage the Consumable Infrastructure Resources and Services

- VIM Core services (keystone, cinder, nova, neutron, etc.)
 - * Tenant Separation
 - * Host aggregates providing resource pooling
 - * Flavor¹ management
- Underlying Resources: are what provides the resources that allow the Consumable Infrastructure Resources and Services to be created and managed by the Cloud Infrastructure Management Software (VIM).
 - Virtualisation
 - Physical infrastructure
 - * Compute
 - * Network: Spine/Leaf; East/West and North/South traffic
 - * Storage

3.1 Resources and Services exposed to VNFs

This section will describe the different services that are exposed for the VNF consumption within the execution zone:

- Tenants: to provide isolated environments
- Virtual Compute: to provide computing resources
- Virtual Storage: to provide storage capacity and performance
- Virtual networking: to provide connectivity within the Cloud Infrastructure and with external networks

3.1.1 Multi-Tenancy (execution environment)

The multi tenancy service permits hosting of several VNF projects with the assurance of isolated environments for each project. Tenants or confusingly “Projects” in OpenStack are isolated environments that enable workloads to be logically separated from each other with:

- differentiated set of associated users
- role-based access of two levels - admin or member (see *RBAC*).
- quota system to provide maximum resources that can be consumed.

This RA does not intend to restrict how workloads are distributed across tenants.

3.1.2 Virtual Compute (vCPU and vRAM)

The virtual compute resources (vCPU and vRAM) used by the VNFs behave like their physical counterparts. A physical core is an actual processor and can support multiple vCPUs through Simultaneous Multithreading (SMT) and CPU overbooking. With no overbooking and SMT of 2 (2 threads per core), each core can support 2 vCPUs. With the same SMT of 2 and overbooking factor of 4, each core can support 8 vCPUs. The performance of a vCPU can be affected by various configurations such as CPU pinning, NUMA alignment, and SMT.

The configuration of the virtual resources will depend on the software and hardware profiles and the flavour (resource sizing) needed to host VNF components. Profiles are defined in “Profiles, Profile Extensions & Flavours” in [1].

¹ Please note “flavours” is used in the Reference Model and shall continue to be used in the context of specifying the geometry of the virtual resources. The term “flavor” is used in this document in the OpenStack context including when specifying configurations; the OpenStack term flavor includes the profile configuration information as “extra specs”.

3.1.3 Virtual Storage

In the Reference Model [1], the “Storage for tenant consumption” section details consumption models for tenants: Platform native, object storage, shared file storage and archival. The choice of a solution will depend on the storage use case needs.

The two storage services offered by Cloud Infrastructure are:

- Persistent storage
- Ephemeral storage

The OpenStack services, Cinder for block storage and Swift for Object Storage, are discussed below in Section “Cloud Infrastructure Management Software (VIM)”.

Ephemeral data is typically stored on the compute host’s local disks, in the form of a file system as part of the provisioning. This storage is volatile, it is deleted when instances are stopped. In environments that support live instance migration between compute hosts, the ephemeral data would need to be stored in a storage system shared between the compute hosts such as on persistent block or object storage.

Three types of persistent data storage are supported in OpenStack:

- Block storage
- Object storage
- Shared file systems storage

The OpenStack Storage Table [28] explains the differences between the storage types and typical use cases.

Block storage is dedicated to persistent data. Data is stored in the form of volumes. Block storage is managed by OpenStack Cinder service and storage Backends. OpenStack compatible storage backend drivers table [29] lists the storage backends compatible with Cinder and their capabilities.

The Object storage is a persistent data storage, not attached to an instance. Data is accessed via API. Object storage is managed by OpenStack Swift.

Images are persistent data, stored using the OpenStack Glance service.

Cinder, Swift, and Glance services are discussed in the section *VIM OpenStack Services*.

3.1.4 Virtual Networking Neutron standalone

Neutron is an OpenStack project that provides “network connectivity as a service” between interface devices (e.g., vNICs) managed by other OpenStack services (e.g., Nova). Neutron allows users to create networks, subnets, ports, routers, etc. Neutron also facilitates traffic isolation between different subnets - within as well as across project(s) by using different type drivers/mechanism drivers that use VLANs, VxLANs, GRE (Generic Routing Encapsulation) tunnels, etc. For Neutron API consumer, this is abstracted and provided by Neutron. Multiple network segments are supported by Neutron via ML2 plugins to simultaneously utilise variety of layer 2 networking technologies like VLAN, VxLAN, GRE, etc. Neutron also allows to create routers to connect layer 2 networks via “neutron-l3-agent”. In addition, floating IP support is also provided that allows a project VM to be accessed using a public IP.

3.1.5 Virtual Networking - 3rd party SDN solution

SDN (Software Defined Networking) controllers separate control and data (user) plane functions where the control plane programmatically configures and controls all network data path elements via open APIs. Open Networking Forum (ONF) defines SDN as “Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today’s applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.”

The key messages of the SDN definition are:

- Decoupling of control and forwarding functions into control plane and data plane
- Networking capabilities that can be instantiated, deployed, configured, and managed like software. Network control is programmable and supports dynamic, manageable, and adaptable networking.
- Support for both overlay and underlay networking

OpenStack Neutron supports open APIs and a pluggable backend where different plugins can be incorporated in the neutron-server.

Plugins for various SDN controllers include either the standard ML-2 plugin or specific monolithic plugins. Neutron supports both core plugins that deal with L2 connectivity and IP address management, and service plugins that support services such as L3 routing, Load Balancers, Firewalls, etc.

Below we will explore an example of an SDN controller from LFN projects, that can be integrated with a Neutron plugin, to help overcome a number of shortcomings of the vanilla Neutron and provide many needed features that can be consumed by VNF/CNF.

Tungsten Fabric (SDN Controller)

Tungsten Fabric [30], an open source SDN in Linux Foundation Networking, offers neutron networking through ML2 based plugin, additionally it supports advanced networking features beyond basic neutron networking via monolithic plugin. It also supports the same advanced networking features via CNI plugin in Kubernetes. Hence, it works as a multi-stack SDN to support VMs, containers, and baremetal workloads. It provides separation of control plane functions and data plane functions with its two components:

- Tungsten Fabric Controller- a set of software services that maintains a model of networks and network policies, typically running on several servers for high availability
- Tungsten Fabric vRouter- installed in each host that runs workloads (virtual machines or containers), the vRouter performs packet forwarding and enforces network and security policies

It is based on proven, standards-based networking technologies but repurposed to work with virtualised workloads and cloud automation in data centres that can range from large scale enterprise data centres to much smaller telco DC (aka POPs). It provides many enhanced features over the native networking implementations of orchestrators, including:

- Highly scalable, multi-tenant networking
- Multi-tenant IP address management
- DHCP, ARP proxies to avoid flooding into networks
- Efficient edge replication for broadcast and multicast traffic
- Local, per-tenant DNS resolution
- Distributed firewall with access control lists
- Application-based security policies
- Distributed load balancing across hosts

- Network address translation (1:1 floating IPs and distributed SNAT)
- Service chaining with virtual network functions
- Dual stack IPv4 and IPv6
- BGP peering with gateway routers
- BGP as a Service (BGPaaS) for distribution of routes between privately managed customer networks and service provider networks

Based on the network layering concepts introduced in the “Network” section in [1], the Tungsten Fabric Controller performs functions of both the SDN underlay (SDNu) and overlay (SDNo) controllers.

The SDN controller exposes a NB API that can be consumed by ETSI MANO for VNF/CNF onboarding, network service onboarding and dynamic service function chaining.

3.1.6 Acceleration

Acceleration deals with both hardware and software accelerations. Hardware acceleration is the use of specialised hardware to perform some function faster than is possible by executing the same function on a general-purpose CPU or on a traditional networking (or other I/O) device (e.g., NIC, switch, storage controller, etc.). The hardware accelerator covers the options for ASICs, SmartNIC, FPGAs, GPU, etc. to offload the main CPU, and to accelerate workload performance. Cloud Infrastructure should manage the accelerators by plugins and provide the acceleration capabilities to VNFs.

With the acceleration abstraction layer defined, hardware accelerators as well as software accelerators can be abstracted as a set of acceleration functions (or acceleration capabilities) which exposes a common API to either the VNF or the host.

3.2 Virtualised Infrastructure Manager (VIM)

The Cloud Infrastructure Management Software (VIM) provides the services for the management of Consumable Resources/Services.

3.2.1 VIM Core services

OpenStack is a complex, multi-project framework, and so we will initially focus on the core services required to provide Infrastructure-as-a-Service (IaaS) as this is generally all that is required for Cloud Infrastructure/VIM use cases. Other components are optional and provide functionality above and beyond Cloud Infrastructure/VIM requirements.

The architecture consists of the core services shown in the figure below; ironic is an optional OpenStack service needed only for bare-metal containers. The rest of this document will address the specific Anuket conformant implementation requirements and recommendations for the core services.

We will refer to the functions above as falling into the following categories to avoid any confusion with other terminology that may be used:

- Foundation node
- Control nodes
- Compute nodes
- Other supporting service nodes, e.g. network, shared storage, logging, monitoring and alerting.

Each deployment of OpenStack should be a unique cloud with its own API endpoint. Sharing underlying cloud resources across OpenStack clouds is not recommended.



Figure 3.1: OpenStack Core Services

OpenStack Services Topology

OpenStack software services are distributed over 2 planes:

- Control Plane that hosts all Control and Management services
- Data Plane (a.k.a. User plane) that provides physical and virtual resources (compute, storage and networking) for the actual virtual workloads to run.

The architecture based on OpenStack technology relies on different types of nodes associated with specific roles:

- Controller node types with control and management services, which include VIM functionalities
- Compute node types running workloads
- Network node types offering L3 connectivity
- Storage node types offering external attached storage (block, object, flat files)

The data plane consists of the compute nodes. It is typical to consider the other node types to be part of the control plane. The following figure depicts the 4 types of nodes constitutive of the Infrastructure: control, compute, network and storage nodes.

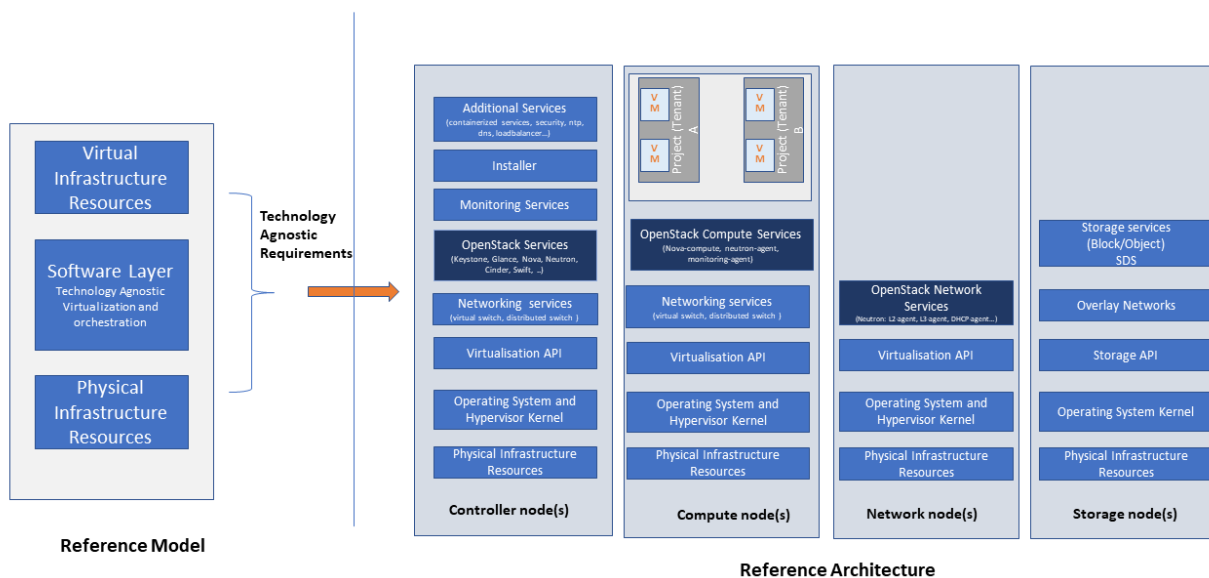


Figure 3.2: OpenStack Services Topology

Deployments can be structured using the distribution of services amongst the 4 node types as depicted in the figure above, but depending on workloads requirements, OpenStack services can also be hosted on the same nodes. For instance, services related to Controller, network and storage roles can be hosted on controller nodes.

Foundation Services

To build and lifecycle manage an OpenStack cloud, it is typically necessary to deploy a server or virtual machine as a deployment node or foundation node.

This function must be able to manage the bare-metal provisioning of the hardware resources but since this does not affect cloud execution it can be detached from the OpenStack cloud and an operator can select their own tooling as they wish. Functional requirements of this node include:

- Build the cloud (control, compute, storage, network hardware resources)
- Patch management / upgrades / change management
- Grow / Shrink resources

Cloud Controller Services

The following OpenStack components are deployed on the Infrastructure. Some of them will be only deployed on control hosts and some of them will be deployed within both control and compute hosts. The table below also maps the OpenStack core services to the Virtual Infrastructure Manager in the Reference Model (RM) [1].

Table 3.1: OpenStack components deployment

RM Management Software	Service	Description	Required / Optional	Deployed on Controller Nodes	Deployed on Compute Nodes
Identity Management (Additional Management Functions) + Catalogue	Keystone	the authentication service	Required	X	
Storage Resources Manager	Glance	the image management service	Required	X	
Storage Resources Manager	Cinder	the block storage management service	Required	X	
Storage Resources Manager	Swift	the Object storage management service	Required	X	
Network Resources Manager	Neutron	the network management service	Required	X	X
Compute Resources Inventory	Placement	resource provider inventory service	Required	X	
Compute Resources Manager + Scheduler	Nova	the compute resources management service	Required	X	X
Compute Resources Manager	Ironic	the Bare Metal Provisioning service	Optional	X	X
(Tool that utilises APIs)	Heat	the orchestration service	Required	X	
UI	Horizon	the WEB UI service	Required	X	
Key Manager	Barbican	the secret data management service	Optional	X	
Acceleration Resources Manager	Cyborg	the acceleration resources and their life cycle management	Optional	X	X

All components must be deployed within a high available architecture that can withstand at least a single node failure and respects the anti-affinity rules for the location of the services (i.e. instances of a same service must run on different nodes).

The services can be containerised or VM hosted as long as they provide the high availability principles described above. The APIs for these OpenStack services are listed in *Interfaces and APIs*.

Cloud Workload Services

This section describes the core set of services and service components needed to run workloads; instances (such as VMs), their networks and storage are referred to as the “Compute Node Services” (a.k.a. user or data plane services). Contrast this with the Controller nodes which host OpenStack services used for cloud administration and management. The Compute Node Services include virtualisation, hypervisor instance creation/deletion, networking and storage services; some of these activities include RabbitMQ queues in the control plane including the scheduling, networking and cinder volume creation/attachment.

- Compute, Storage, Network services:
 - Nova Compute service: nova-compute (creating/deleting servers (a.k.a. instances))
 - Neutron Networking service: neutron-l2-agent (manage local Open vSwitch (OVS) configuration), VXLAN
 - Local Storage (Ephemeral, Root, etc.)
 - Attached Storage (using Local drivers)

3.2.2 Tenant Isolation

In Keystone v1 and v2 (both deprecated), the term “tenant” was used in OpenStack. With Keystone v3, the term “project” got adopted and both the terms became interchangeable. According to OpenStack glossary [31], Projects represent the base unit of resources (compute, storage and network) in OpenStack, in that all assigned resources in OpenStack are owned by a specific project. OpenStack offers multi-tenancy by means of resource (compute, network and storage) separation via projects. OpenStack offers ways to share virtual resources between projects while maintaining logical separation. As an example, traffic separation is provided by creating different VLAN ids for neutron networks of different projects. As another example, if host separation is needed, nova scheduler offers AggregateMultiTenancyIsolation scheduler filter to separate projects in host aggregates. Thus, if a host in an aggregate is configured for a particular project, only the instances from that project are placed on the host. Overall, tenant isolation ensures that the resources of a project are not affected by resources of another project.

This document uses the term “project” when referring to OpenStack services and “tenant” (RM Section “Virtual resources”) to represent an independently manageable logical pool of resources.

3.2.3 Cloud partitioning: Host Aggregates, Availability Zones

Cloud administrators can partition the hosts within an OpenStack cloud using Host Aggregates and Availability Zones.

A Host Aggregate is a group of hosts (compute nodes) with specific characteristics and with the same specifications, software and/or hardware properties. Example would be a Host Aggregate created for specific hardware or performance characteristics. The administrator assigns key-value pairs to Host Aggregates, these are then used when scheduling VMs. A host can belong to multiple Host Aggregates. Host Aggregates are not explicitly exposed to tenants.

Availability Zones (AZs) rely on Host Aggregates and make the partitioning visible to tenants. They are defined by attaching specific metadata information to an aggregate, making the aggregate visible for tenants. Hosts can only be in a single Availability Zone. By default a host is part of a default Availability Zone, even if it doesn’t belong to an aggregate. Availability Zones can be used to provide resiliency and fault tolerance for workloads deployments, for example by means of physical hosting distribution of Compute Nodes in separate racks with separate power supply and eventually in different rooms. They permit rolling upgrades - an AZ at a time upgrade with enough time between AZ upgrades to allow recovery of tenant workloads on the upgraded AZ. AZs can also be used to segregate workloads.

An over use of Host Aggregates and Availability Zones can result in a granular partition of the cloud and, hence, operational complexities and inefficiencies.

3.2.4 Flavor management

In OpenStack a flavor defines the compute, memory, and storage capacity of nova instances. When instances are spawned, they are mapped to flavors which define the available hardware configuration for them. For simplicity, operators may create named flavors specifying both the sizing and the “Software and Hardware Profile Configurations” [1].

3.3 Underlying Resources

The number of Compute nodes (for workloads) determines the load on the controller nodes and networking traffic and, hence, the number of controller nodes needed in the OpenStack cloud; the number of controller nodes required is determined on the load placed on these controller nodes and the need for High Availability and quorum requires at least 3 instances of many of the services on these controller nodes.

3.3.1 Virtualisation and hypervisors

Virtualisation is a technology that enables a guest Operating System (OS) to be abstracted from the underlying hardware and software. This allows to run multiple Virtual Machines (VMs) on the same hardware. Each such VMs have their own OS and are isolated from each other i.e. application running on one VM does not have the access to resources of another VM. Such virtualisation is supported by various hypervisors available as open-source (KVM, Xen, etc.) as well as commercial (Hyper-V, Citrix XenServer, etc.). Selecting a hypervisor depends on the workload needs and the features provided by various hypervisors as illustrated in Hypervisor Feature Support Matrix [32]. OpenStack (Nova) allows the use of various hypervisors within a single installation by means of scheduler filters like ComputeFilter, ImagePropertiesFilter etc.

Virtualisation Services: The OpenStack nova-compute service supports multiple hypervisors natively or through libvirt. The preferred supported hypervisor in this Reference Architecture is KVM.

Note: Other hypervisors (such as ESXi) can also be supported as long as they can interoperate with other OpenStack components (e.g., those listed in this Reference Architecture) using standard interfaces and APIs as specified in Chapter 5.

3.3.2 Physical Infrastructure

The aim is to specify the requirements on deploying the VIM, from ground up (in a shipping container), and what resources are required of the DC (Data Centre).

- Servers
 - Compute
 - Storage
 - Control (min 3 for Core DC)
- Network considerations
 - Data centre gateway
 - Firewall (around the control plane, storage, etc.)
 - Data centre network fabric / Clos (spine/leaf) - Horizontal scale

- Storage networking, control plane and data plane
- Raw packet - tenant networking allowing “wild west” connection
- Storage
 - Storage technologies are multiple, they are extensively described in “Storage Implementation Stereotypes” [1]. Storage backends are discussed in *Storage Backend*.
- Acceleration
 - SmartNIC
 - GPU
 - FPGA

Physical nodes

Cloud Infrastructure physical Nodes

The physical resources required for the Cloud Infrastructure are mainly based on COTS x86 hardware for control and data plane nodes. HW profiles are defined in the chapters “Cloud Infrastructure Hardware Profile Description” and “Cloud Infrastructure Hardware Profiles Features and Requirements” in [1].

Network

The recommended network architecture is spine and leaf topology.



Figure 3.3: Network Fabric - Physical

The figure above shows a physical network layout where each physical server is dual homed to TOR (Leaf/Access) switches with redundant (2x) connections. The Leaf switches are dual homed with redundant connections to spines.

Storage

OpenStack supports many different storage architectures and backends [33]. The choice of a particular backend storage is driven by a number of factors including: scalability, resiliency, availability, data durability, capacity and performance.

Most cloud storage architectures incorporate a number of clustered storage nodes that provide high bandwidth access to physical storage backends connected by high speed networks. The architecture consists of multiple storage controller units, each a generic server (CPU, Cache, storage), managing a number of high-performance hard drives. The distributed block storage software creates an abstract single pool of storage by aggregating all of the controller units. Advanced and high-speed networking (data routing) and global load balancing techniques ensure high-performance, high availability storage system.

3.4 Cloud Topology

A telco cloud will typically be deployed in multiple locations (“sites”) of varying size and capabilities (HVAC, for example); or looking at this in the context of OpenStack, multiple clouds (i.e. OpenStack end-points) will be deployed that do not rely on each other, by design; each cloud consists of a set of resources isolated from resources of the other clouds. The application layer must span such end-points in order to provide the required service SLA. Irrespective of the nature of the deployment characteristics (e.g., number of racks, number of hosts), the intent of the architecture would be to allow VNFs to be deployed in these sites without major changes.

Some examples of such topologies include:

- Large data centre capable of hosting potentially thousands of servers and the networking to support them
- Intermediate data centre (such as a central office) capable of hosting up to a hundred servers
- Edge (not customer premise) capable of hosting ten to fifty servers

In order to provide the expected availability for any given service, a number of different OpenStack deployment topologies can be considered. This section explores the main options and highlights the characteristics of each. Ultimately the decision rests with the operator to achieve specific availability target taking into account use case, data centre capabilities, economics and risks.

Topology Overview

Availability of any single OpenStack cloud is dependent on a number of factors including:

- environmental - dual connected power and PDUs, redundant cooling, rack distribution, etc.
- resilient network fabric - ToR (leaf), spine, overlay networking, underlay networking, etc. It is assumed that all network components are designed to be fault tolerant and all OpenStack controllers, computes and storage are dual-homed to alternate leaf switches.
- controller nodes setup in-line with the vendor recommendation (e.g., min 3 physical nodes)
- network nodes (where applicable)
- backend storage nodes setup for highly availability based on quorum (aligned with vendor implementation)
- compute nodes sized to handle the entire workload following local failure scenario

Assumptions and conventions:

- Region is represented by a single OpenStack control plane.
- Resource Failure Domain is effectively the “blast radius” of any major infrastructure failure such as loss of PDU or network leafs.
- Control plane includes redundant network nodes where OVS-kernel is used.
- Controller nodes should be setup for high availability based on quorum (aligned with vendor implementation).

- Shared storage is optional, but it is important to ensure shared assets are distributed across serving clouds such as boot images. Storage needs, per deployment and use cases, can be found in “Storage Scenarios and Architecture Fit” [1].

Table 3.2: Cloud Topology: Redundancy Models

Topology Ref	Type	Control Planes	Shared Storage (optional)	Compute AZs	Achievable Service Availability %	Service Multi-region awareness	Notes
1	Local Redundancy - workload spread across servers	1	1	1	Variable	Not required	Suitable where only limited local application availability is required e.g. nova anti-affinity
2	Regional Redundancy - workload spread across AZs	1	≥ 2	≥ 2	$>99.n$	Not required	Suitable where local application HA is required. Control plane should be distributed across DC failure domains (assuming layer 2 connectivity) but may be unavailable during up grades
3	Global Redundancy - workload spread across multiple Regions	≥ 2	≥ 2	≥ 2	$>99.nn$	Required	Suitable where local and region application HA is required Control plane could be kept available in one site during up-grades

Topology Overview

Topology 1 - Local Redundancy

Under normal operation this deployment can handle a single failure of a controller node or storage node without any impact to the service. If a compute node fails the application layer (often the VNFM) would need to restart workloads on a spare compute node of similar capability i.e., cloud may need to be provided with $n+1$ capacity. In the case of an active/active application deployed to separate compute nodes (with hypervisor anti-affinity) there would be no service impact.

Important to consider:

- Where possible servers should be distributed and cabled to reduce the impact of any failure e.g., PDU (Power Distribution Unit), rack failure. Because each operator has individual site constraints this document will not propose a standard rack layout.
- During maintenance of the control plane, whilst the data (forwarding) plane remains unaffected, the control plane API may not be available and some applications may be relying on it during normal application operation for example for scaling. Additionally if the upgrade involves updating OpenStack services on the compute nodes care needs to be taken. OVS-kernel networking operations may also be impacted during this time.

- During maintenance of storage (e.g., ceph) there is an increased risk of a service-impacting failure, so it is generally recommended to deploy at least one more server than the minimum required for redundancy.

Topology 2 - Regional Redundancy

Under normal operation this topology can handle a single failure of a controller node but provides additional protection to the compute plane and storage. If the application is deployed across 2 or more AZs a major failure impacting the nodes in one AZ can be tolerated assuming the application deployment allows for this. There is a risk with split-brain so a means of deciding application quorum is recommended or by using a third AZ or arbitrator.

Important to consider:

- All those points listed for Topology 1 above.
- When using 3 controller nodes and distributing these physically across the same locations as the computes, if you lose the location with 2 controllers the OpenStack services would be impacted as quorum cannot be gained with a single controller node. It is also possible to use more than 3 controller nodes and co-locate one with each compute AZ allowing lower-risk maintenance but care must be taken to avoid split brain.
- The distributed network fabric must support L2 for the OpenStack control plane VIPs.

Topology 3 - Global Redundancy

Following the example set by public cloud providers who provide Regions and Availability Zones this is effectively a multi-region OpenStack. Assuming the application can make use of this model this provides the highest level of availability but would mean IP level failure controlled outside of OpenStack by global service load balancing (GSLB) i.e., DNS with minimum TTL configured, or client applications that are capable of failing over themselves. This has the added advantage that no resources are shared between different Regions so any fault is isolated to a single cloud and also allows maintenance to take place without service impact.

4 Cloud Infrastructure & VIM Component Level Architecture

Chapter 3 introduced the components of an OpenStack-based IaaS:

- Consumable Infrastructure Resources and Services
- Cloud Infrastructure Management Software (VIM: OpenStack) core services and architectural constructs needed to consume and manage the consumable resources
- Underlying physical compute, storage, and networking resources

This chapter delves deeper into the capabilities of these different resources and their needed configurations to create and operate an OpenStack-based IaaS cloud. This chapter specifies details on the structure of control and user planes, operating systems, hypervisors, and BIOS configurations, and architectural details of underlay and overlay networking, and storage, and the distribution of OpenStack service components among nodes. The chapter also covers implementation support for the “Profiles, Profile Extensions & Flavours” [1]; the OpenStack flavor types capture both the sizing and the profile configuration (of the host).

4.1 Underlying Resources Configuration and Dimensioning

4.1.1 Virtualisation layer

In OpenStack, KVM is configured as the default hypervisor for compute nodes.

- Configuration: OpenStack [34] specifies the steps/instructions to configure KVM:
 - Enable KVM based hardware virtualisation in BIOS. OpenStack provides instructions on how to enable hardware virtualisation for different hardware platforms (x86, Power)
 - * QEMU is similar to KVM in that both are libvirt controlled, have the same feature set and utilise compatible virtual machine images
 - Configure Compute backing storage
 - Specify the CPU Model for KVM guests (VMs)
 - KVM Performance Tweaks
- Hardening the virtualisation layers [35]
 - OpenStack recommends minimizing the code base by removing unused components
 - sVirt (Secure Virtualisation) provides isolation between VM processes, devices, data files and system processes

4.1.2 Compute

Cloud Deployment (Foundation/management) Node

Minimal configuration: 1 node

OpenStack Control Plane Servers (Control Nodes)

- BIOS Requirements

For OpenStack control nodes we use the BIOS parameters for the basic profile defined in “Cloud Infrastructure Hardware Profiles Features and Requirements” [1]. Additionally, for OpenStack we need to set the following boot parameters:

Table 4.1: Boot parameters

BIOS/boot Parameter	Value
Boot disks	RAID 1
CPU reservation for host (kernel)	1 core per NUMA
CPU allocation ratio	2:1

- How many nodes to meet SLA
 - Minimum 3 nodes for high availability
- HW specifications
 - Boot disks are dedicated with Flash technology disks
- Sizing rules
 - It is easy to horizontally scale the number of control nodes

- The number of control nodes is determined by a minimum number needed for high availability (viz., 3 nodes) and the extra nodes needed to handle the transaction volumes, in particular, for Messaging service (e.g., RabbitMQ) and Database (e.g., MySQL) to track state.
- The number of control nodes only needs to be increased in environments with a lot of changes, such as a testing lab, or a very large cloud footprint (rule of thumb: number of control nodes = 3 + quotient (number of compute nodes/1000)).
- The Services Placement Summary table [36] specifies the number of instances that are required based upon the cloud size (number of nodes).

Network nodes

Networks nodes are mainly used for L3 traffic management for overlay tenant network (see more detail in Neutron section).

- BIOS requirements

Table 4.2: BIOS requirements

BIOS/boot Parameter	Value
Boot disks	RAID 1

- How many nodes to meet SLA
 - Minimum 2 nodes for high availability using VRRP.
- HW specifications
 - 3 NICs card are needed if we want to isolate the different flows:
 - * 1 NIC for Tenant Network
 - * 1 NIC for External Network
 - * 1 NIC for Other Networks (PXE, Mngt ...)
- Sizing rules
 - Scale out of network node is not easy
 - DVR can be an option for large deployment (see more detail in section Neutron)

Storage nodes

- BIOS requirements

Table 4.3: BIOS requirements

BIOS/boot Parameter	Value
Boot disks	RAID 1

- HW specifications: please see “Storage” in [1]
- How many nodes to meet SLA: Active-Passive is the default and recently OpenStack started to support Active-Active
- Sizing rules: minimum 2 x 1 TB; recommended 2 x 10 TB

Compute Nodes

This section specifies the compute node configurations to support the Basic and High-Performance profiles; in OpenStack this would be accomplished by specifying the configurations when creating “flavors”. The cloud operator may choose to implement certain profile-extensions (Profile Extensions (Specialisations) [1]) as a set of standard configurations, of a given profile, capturing some of the variability through different values or extra specifications.

- The software and hardware configurations are as specified in the Cloud Infrastructure Hardware Profiles Features and Requirements in [1].
- BIOS requirement
 - The general BIOS requirements are described in the Cloud Infrastructure Hardware Profiles Features and Requirements [1].

Example Profiles and their Extensions

The Reference Model specifies the Basic (B) and High-Performance (H) profile types. The Reference Model also provides a choice of network acceleration capabilities utilising, for example, DPDK and SR-IOV technologies. The table below lists a few simple examples of profile extensions and some of their capabilities.

Table 4.4: Profile Extensions and Capabilities

Profile Extensions	Description	CPU Allocation Ratio	SMT	CPU Pinning	NUMA	Huge pages	Data Traffic
B1	Basic Profile NoCPU over-subscription profile extension	1:1	Y	N	N	N	OVS- kernel
HV	High Performance Profile	1:1	Y	Y	Y	Y	OVS- kernel
HD	High Performance Profile with DPDK profile extension	1:1	Y	Y	Y	Y	OVS-DPDK
HS	High Performance Profile with SR-IOV profile extension	1:1	Y	Y	Y	Y	SR-IOV

BIOS Settings

A number of capabilities need to be enabled in the BIOS (such as NUMA and SMT); the “Cloud Infrastructure Software Profile Description” section in the Reference Model specifies the capabilities required to be configured. Please note that capabilities may need to be configured in multiple systems. For OpenStack, we also need to set the following boot parameters:

Table 4.5: BIOS requirements

BIOS/boot Parameter	Basic	High Performance
Boot disks	RAID 1	RAID 1

- How many nodes to meet SLA
 - minimum: two nodes per profile
- HW specifications
 - Boot disks are dedicated with Flash technology disks
- In case of DPDK usage:

Table 4.6: DPDK usage

Layer	Description
Cloud infrastructure	Important is placement of NICs to get NUMA-balanced system (balancing the I/O, memory, and storage across both sockets), and configuration of NIC features. Server BIOS and Host OS kernel command line settings are described in DPDK release notes [37] and DPDK performance reports [38]. Disabling power settings (like Intel Turbo Boost Technology) brings stable performance results, although understanding if and when they benefit workloads and enabling them can achieve better performance results.
Workload	DPDK uses core affinity along with 1G or 2M huge pages, NUMA settings (to avoid crossing interconnect between CPUs), and DPDK Poll Mode Drivers (PMD, on reserved cores) to get the best performance. DPDK versions xx.11 are Long-Term Support maintained stable release with back-ported bug fixes for a two-year period.

- Sizing rules

Table 4.7: Mnemonic

Description	Mnemonic
Number of CPU sockets	s
Number of cores	c
SMT	t
RAM	rt
Storage	d
Overcommit	o
Average vCPU per instance	v
Average RAM per instance	ri

Table 4.8: Sizing rules

		Basic	High-Performance
# of VMs per node (vCPU)	$(s*c*t*o)/v$	$4*(s*c*t)/v$	$(s*c*t)/v$
# of VMs per node (RAM)	rt/ri	rt/ri	rt/ri
Max # of VMs per node		$\min(4*(s*c*t)/v, rt/ri)$	$\min((s*c*t)/v, rt/ri)$

Caveats:

- These are theoretical limits
- Affinity and anti-affinity rules, among other factors, affect the sizing

Compute Resource Pooling Considerations

- Multiple pools of hardware resources where each resource pool caters for workloads of a specific profile (for example, High-Performance) leads to inefficient use of the hardware as the server resources are configured specifically for a profile. If not properly sized or when demand changes, this can lead to oversupply/starvation scenarios; reconfiguration may not be possible because of the underlying hardware or inability to vacate servers for reconfiguration to support another profile type.
- Single pool of hardware resources including for controllers have the same CPU configuration. This is operationally efficient as any server can be utilised to support any profile or controller. The single pool is valuable with unpredictable workloads or when the demand of certain profiles is insufficient to justify individual hardware selection.

Reservation of Compute Node Cores

The *Infrastructure Requirements* `inf.com.08` requires the allocation of “certain number of host cores/threads to non-tenant workloads such as for OpenStack services.” A number (“n”) of random cores can be reserved for host services (including OpenStack services) by specifying the following in `nova.conf`:

```
reserved_host_cpus = n
```

where n is any positive integer.

If we wish to dedicate specific cores for host processing we need to consider two different usage scenarios:

1. Require dedicated cores for Guest resources
2. No dedicated cores are required for Guest resources

Scenario #1, results in compute nodes that host both pinned and unpinned workloads. In the OpenStack Wallaby release, scenario #1 is not supported; it may also be something that operators may not allow. Scenario #2 is supported through the specification of the `cpu_shared_set` configuration. The cores and their sibling threads dedicated to the host services are those that do not exist in the `cpu_shared_set` configuration.

Let us consider a compute host with 20 cores with SMT enabled (let us disregard NUMA) and the following parameters specified. The physical cores are numbered ‘0’ to ‘19’ while the sibling threads are numbered ‘20’ to ‘39’ where the vCPUs numbered ‘0’ and ‘20’, ‘1’ and ‘21’, etc. are siblings:

```
cpu_shared_set = 1-7,9-19,21-27,29-39 (can also be specified as cpu_shared_set = 1-19,&8,21-39,&28)
```

This implies that the two physical cores ‘0’ and ‘8’ and their sibling threads ‘20’ and ‘28’ are dedicated to the host services, and 19 cores and their sibling threads are available for Guest instances and can be over allocated as per the specified `cpu_allocation_ratio` in `nova.conf`.

Pinned and Unpinned CPUs

When a server (viz., an instance) is created the vCPUs are, by default, not assigned to a particular host CPU. Certain workloads require real-time or near real-time behavior viz., uninterrupted access to their cores. For such workloads, CPU pinning allows us to bind an instance’s vCPUs to particular host cores or SMT threads. To configure a flavor to use pinned vCPUs, we use a dedicated CPU policy.

```
openstack flavor set .xlarge -property hw:cpu_policy=dedicated
```

While an instance with pinned CPUs cannot use CPUs of another pinned instance, this does not apply to unpinned instances; an unpinned instance can utilise the pinned CPUs of another instance. To prevent unpinned instances from disrupting pinned instances, the hosts with CPU pinning enabled are pooled in their own host aggregate and hosts with CPU pinning disabled are pooled in another non-overlapping host aggregate.

Compute node configurations for Profiles and OpenStack Flavors

This section specifies the compute node configurations to support profiles and flavors.

Cloud Infrastructure Hardware Profile

The Cloud Infrastructure Hardware (or simply “host”) profile and configuration parameters are utilised in the reference architecture to define different hardware profiles; these are used to configure the BIOS settings on a physical server and configure utility software (such as Operating System and Hypervisor).

An OpenStack flavor defines the characteristics (“capabilities”) of a server (viz., VMs or instances) that will be deployed on hosts assigned a host-profile. A many to many relationship exists between flavors and host profiles. Multiple flavors can be defined with overlapping capability specifications with only slight variations that servers of these flavor types can be hosted on similar configured (host profile) compute hosts. Similarly, a server can be specified with a flavor that allows it to be hosted on, say, a host configured as per the Basic profile or a host configured as per the High-Performance profile. Please note that workloads that specify a server flavor so as to be hosted on a host configured as per the High-Performance profile, may not be able to run (adequately with expected performance) on a host configured as per the Basic profile.

A given host can only be assigned a single host profile; a host profile can be assigned to multiple hosts. Host profiles are immutable and hence when a configuration needs to be changed, a new host profile is created.

CPU Allocation Ratio and CPU Pinning

A given host (compute node) can only support a single CPU Allocation Ratio. Thus, to support the B1 and B4 Basic profile extensions (Section *Compute Nodes*) with CPU Allocation Ratios of 1.0 and 4.0 we will need to create 2 different host profiles and separate host aggregates for each of the host profiles. The CPU Allocation Ratio is set in the hypervisor on the host.

When the CPU Allocation Ratio exceeds 1.0 then CPU Pinning also needs to be disabled.

Server Configurations

The different networking choices - OVS-Kernel, OVS-DPDK, SR-IOV - result in different NIC port, LAG (Link Aggregation Group), and other configurations. Some of these are shown diagrammatically in section *Compute Nodes*.

Leaf and Compute Ports for Server Flavors must align

Compute hosts have varying numbers of Ports/Bonds/LAGs/Trunks/VLANs connected with Leaf ports. Each Leaf port (in A/B pair) must be configured to align with the interfaces required for the compute flavor.

Physical Connections/Cables are generally the same within a zone, regardless of these specific L2/L3/SR-IOV configurations for the compute.

Compute Bond Port: TOR port maps VLANs directly with IRBs on the TOR pair for tunnel packets and Control Plane Control and Storage packets. These packets are then routed on the underlay network GRT.

Server Flavors: B1, B4, HV, HD

Compute SR-IOV Port: TOR port maps VLANs with bridge domains that extend to IRBs, using VXLAN VNI. The TOR port associates each packet’s outer VLAN tag with a bridge domain to support VNF interface adjacencies over the local EVPN/MAC bridge domain. This model also applies to direct physical connections with transport elements.

Server Flavors: HS

Notes on SR-IOV

SR-IOV, at the compute server, routes Guest traffic directly with a partitioned NIC card, bypassing the hypervisor and vSwitch software, which provides higher bps/pps throughput for the Guest server. OpenStack and MANO manage SR-IOV configurations for Tenant server interfaces.

- Server, Linux, and NIC card hardware standards include SR-IOV and VF requirements
- High Performance profile for SR-IOV (hs series) with specific NIC/Leaf port configurations
- OpenStack supports SR-IOV provisioning
- Implement Security Policy, Tap/Mirror, QoS, etc. functions in the NIC, Leaf, and other places

Because SR-IOV involves Guest VLANs between the compute server and the ToR/Leafs, Guest automation and server placement necessarily involves the Leaf switches (e.g., access VLAN outer tag mapping with VXLAN EVPN).

- Local VXLAN tunneling over IP-switched fabric implemented between VTEPs on Leaf switches
- Leaf configuration controlled by SDN-Fabric/Global Controller
- Underlay uses VXLAN-enabled switches for EVPN support

SR-IOV-based networking for Tenant Use Cases is required where vSwitch-based networking throughput is inadequate.

Example Host Configurations

Host configurations for B1, B4 Profile Extensions

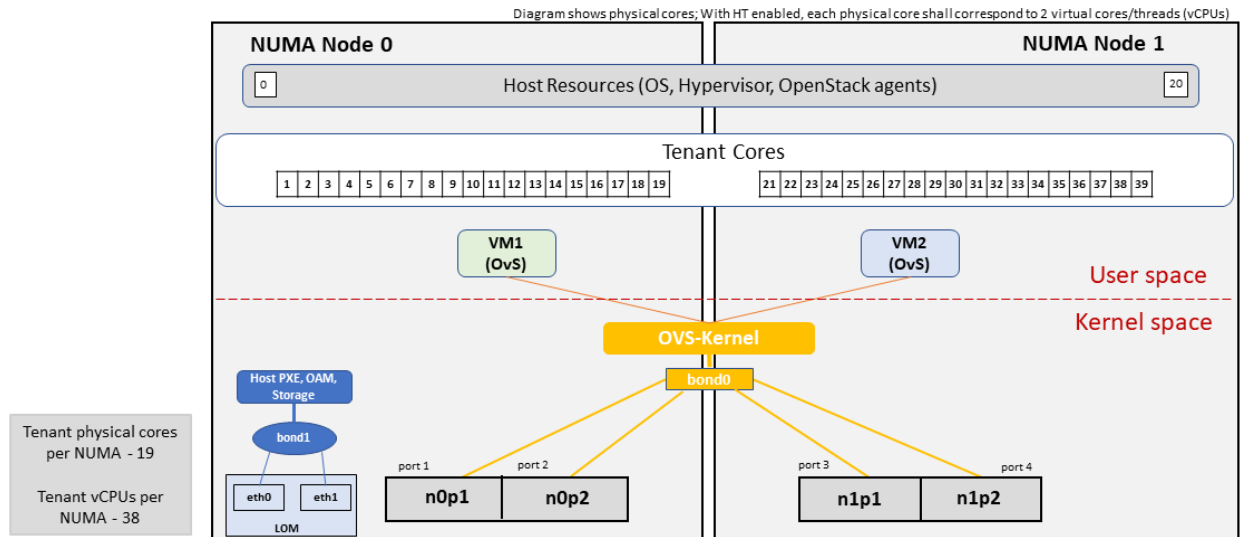


Figure 4.1: Basic Profile Host Configuration (example and simplified)

Let us refer to the data traffic networking configuration depicted in the figure above to be part of the hp-B1-a and hp-B4-a host profiles and this requires the configurations as Table *Configuration of Basic Flavor Capabilities*.

Table 4.9: Configuration of Basic Flavor Capabilities

Capability	Configured in	Host profile: hp-B1-a	Host profile: hp-B4-a
CPU Allocation Ratio	Hypervisor	1:1	4:1
CPU Pinning	BIOS	Enable	Disable
SMT	BIOS	Enable	Enable
NUMA	BIOS	Disable	Disable
Huge pages	BIOS	No	No
Profile Extensions		B1	B4

The figure below shows the networking configuration where the storage and OAM share networking but are independent of the PXE network.

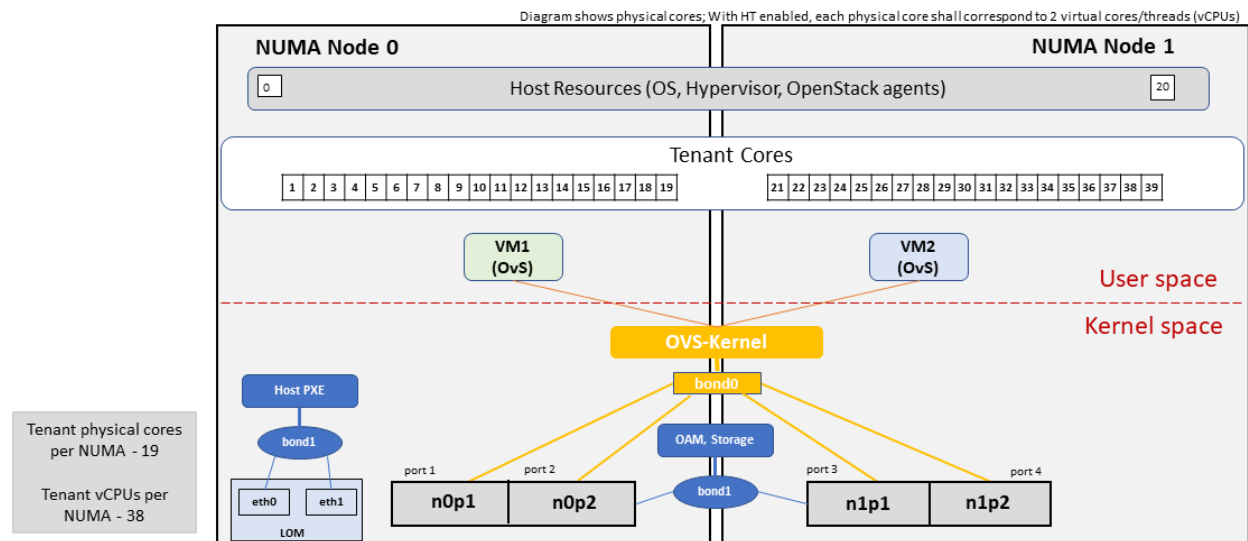


Figure 4.2: Basic Profile Host Configuration with shared Storage and OAM networking (example and simplified)

Let us refer to the above networking set up to be part of the hp-B1-b and hp-B4-b host profiles but the basic configurations as specified in Table *Configuration of Basic Flavor Capabilities*.

In our example, the Profile Extensions B1 and B4, are each mapped to two different host profiles hp-B1-a and hp-B1-b, and hp-B4-a and hp-B4-b respectively. Different network configurations, reservation of CPU cores, Lag values, etc. result in different host profiles.

To ensure Tenant CPU isolation from the host services (Operating System (OS), hypervisor and OpenStack agents), the following needs to be configured:

Table 4.10: GRUB Configuration of Basic Profile with shared Storage

GRUB Bootloader Parameter	Description	Values
isolcpus (Applicable only on Compute Servers)	A set of cores isolated from the host processes. Contains vCPUs reserved for Tenants and DPDK	isolcpus=1-19, 21-39, 41-59, 61-79

Host configuration for HV Profile Extensions

The above examples of host networking configurations for the B1 and B4 Profile Extensions are also suitable for the HV Profile Extensions; however, the hypervisor and BIOS settings will be different (see table below) and hence there will be a need for different host profiles. Table *Configuration of High Performance Flavor Capabilities* gives examples of three different host profiles; one each for HV, HD and HS Profile Extensions.

Table 4.11: Configuration of High Performance Flavor Capabilities

Capability	Configured in	Host profile: hp-hv-a	Host profile: hp-hd-a	Host profile: hp-hs-a
Profile Extensions		HV	HD	HS
CPU Allocation Ratio	Hypervisor	1:1	1:1	1:1
NUMA	BIOS, Operating System, Hypervisor and OpenStack Nova Scheduler	Enable	Enable	Enable
CPU Pinning (requires NUMA)	OpenStack Nova Scheduler	Enable	Enable	Enable
SMT	BIOS	Enable	Enable	Enable
Huge pages	BIOS	Yes	Yes	Yes

Host Networking configuration for HD Profile Extensions

An example of the data traffic configuration for the HD (OVS-DPDK) Profile Extensions is shown in the figure below.

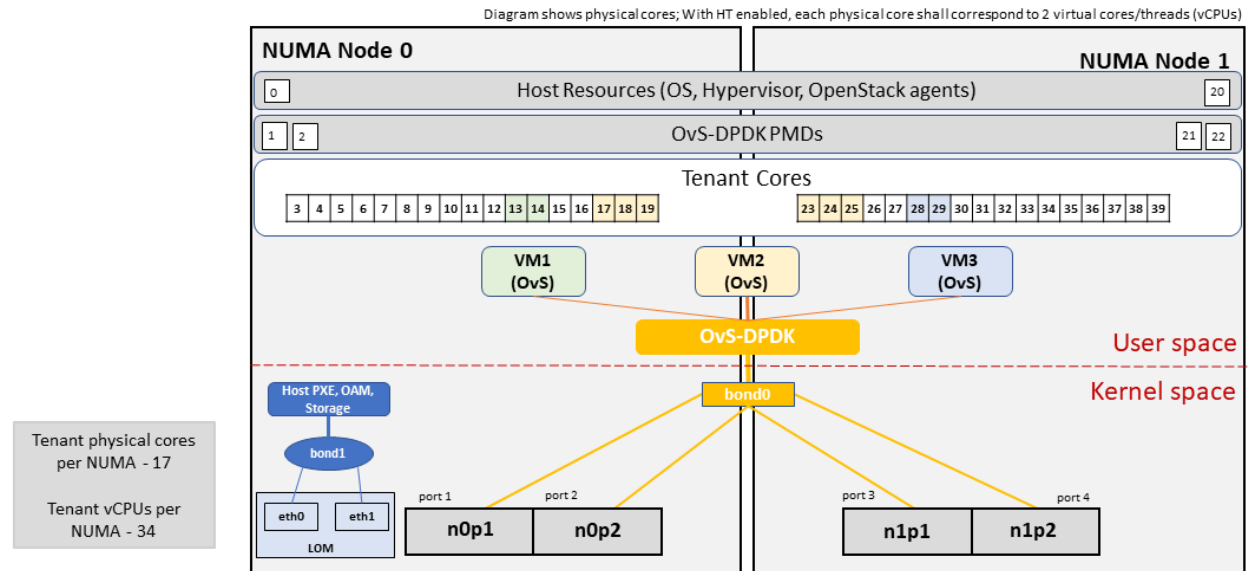


Figure 4.3: High Performance Profile Host Configuration with DPDK acceleration (example and simplified)

To ensure Tenant and DPDK CPU isolation from the host services (Operating System (OS), hypervisor and OpenStack agents), the following needs to be configured:

Table 4.12: GRUB Configuration of High Performance Flavor with DPDK

GRUB Bootloader Parameter	Description	Values
isolcpus (Applicable only on Compute Servers)	A set of cores isolated from the host processes. Contains vCPUs reserved for Tenants and DPDK	isolcpus=3-19, 23-39, 43-59, 63-79

Host Networking configuration for HS Profile Extensions

An example of the data traffic configuration for the HS (SR-IOV) Profile Extensions is shown in the figure below.



Figure 4.4: High Performance Profile Host Configuration with SR-IOV (example and simplified)

To ensure Tenant CPU isolation from the host services (Operating System (OS), hypervisor and OpenStack agents), the following needs to be configured:

Table 4.13: GRUB Configuration of High Performance Flavor with SR-IOV

GRUB Bootloader Parameter	Description	Values
isolcpus (Applicable only on Compute Servers)	A set of cores isolated from the host processes. Contains vCPUs reserved for Tenants	isolcpus=1-19, 21-39, 41-59, 61-79

Using Hosts of a Host Profile type

As we have seen Profile Extensions are supported by configuring hosts in accordance with the Profile Extensions specifications. For example, an instance of flavor type B1 can be hosted on a compute node that is configured as an hp-B1-a or hp-B1-b host profile. All compute nodes configured with hp-B1-a or hp-B1-b host profile are made part of a host aggregate, say, ha-B1 and, thus, during server instantiation of B1 flavor hosts from the ha-B1 host aggregate will be selected.

4.1.3 Network Fabric

Networking Fabric consists of:

- Physical switches, routers...
- Switch OS
- Minimum number of switches
- Dimensioning for East/West and North/South
- Spine / Leaf topology - east - west
- Global Network parameters

- OpenStack control plane VLAN / VXLAN layout
- Provider VLANs

Physical Network Topology

High Level Logical Network Layout



Figure 4.5: Indicative OpenStack Network Layout

Table 4.14: OpenStack Network Characteristics

Network	Description	Characteristics
Provisioning & Management	Initial OS bootstrapping of the servers via PXE, deployment of software and thereafter for access from within the control plane	<ul style="list-style-type: none"> • Security Domain: Management • Externally Routable: No • Connected to: All nodes
Internal API	Intra-OpenStack service API communications, messaging, and database replication	<ul style="list-style-type: none"> • Security Domain: Management • Externally Routable: No • Connected to: All nodes except foundation
Storage Management	Backend connectivity between storage nodes for heartbeats, data object replication and synchronisation	<ul style="list-style-type: none"> • Security Domain: Storage • Externally Routable: No • Connected to: All nodes except foundation
Storage Front-end	Block/Object storage access via cinder/swift	<ul style="list-style-type: none"> • Security Domain: Storage • Externally Routable: No • Connected to: All nodes except foundation
Tenant	VXLAN / Geneve project overlay networks (OVS kernel mode) - i.e., RFC1918 [39] re-usable private networks as controlled by cloud administrator	<ul style="list-style-type: none"> • Security Domain: Underlay • Externally Routable: No • Connected to: controllers and computes
External API	Hosts the public OpenStack API endpoints including the dashboard (Horizon)	<ul style="list-style-type: none"> • Security Domain: Public • Externally routable: Yes • Connected to: controllers
External Provider (FIP)	Network with a pool of externally routable IP addresses used by neutron routers to NAT to/from the tenant RFC1918 [39] private networks	<ul style="list-style-type: none"> • Security Domain: Data Centre • Externally routable: Yes • Connected to: controllers, OVS computes
External Provider (VLAN)	External Data Centre L2 networks (VLANs) that are directly accessible to the project Note: External IP address management is required	<ul style="list-style-type: none"> • Security Domain: Data Centre • Externally routable: Yes • Connected to: OVS DPDK computes
IPMI / Out of Band	The remote “lights-out” management port of the servers e.g., iLO, IDRAC / IPMI / Redfish	<ul style="list-style-type: none"> • Security Domain: Management • Externally routable: No • Connected to: IPMI port on all servers

A VNF application network topology is expressed in terms of servers, vNIC interfaces with vNet access networks,

and WAN Networks while the VNF Application Servers require multiple vNICs, VLANs, and host routes configured within the server's Kernel.

Octavia v2 API conformant Load Balancing

Load balancing is needed for automatic scaling, managing availability and changes. Octavia [40] is an open-source load balancer for OpenStack, based on HAProxy, and replaces the deprecated (as of OpenStack Queens release) Neutron LBaaS. The Octavia v2 API is a superset of the deprecated Neutron LBaaS v2 API and has a similar CLI for seamless transition.

As a default Octavia utilises Amphorae Load Balancer. Amphorae consists of a fleet of servers (VMs, containers or bare metal servers) and delivers horizontal scaling by managing and spinning these resources on demand. The reference implementation of the Amphorae image is an Ubuntu virtual machine running HAProxy.

Octavia depends upon a number of OpenStack services including Nova for spinning up compute resources on demand and their life cycle management; Neutron for connectivity between the compute resources, project environment and external networks; Keystone for authentication; and Glance for storing of the compute resource images.

Octavia supports provider drivers which allows third-party load balancing drivers (such as F5, AVI, etc.) to be utilised instead of the default Amphorae load balancer. When creating a third-party load balancer, the **provider** attribute is used to specify the backend to be used to create the load balancer. The **list providers** lists all enabled provider drivers. Instead of using the provider parameter, an alternate is to specify the `flavor_id` in the create call where provider-specific Octavia flavors have been created.

Neutron Extensions

OpenStack Neutron is an extensible framework that allows incorporation through plugins and API Extensions. API Extensions provide a method for introducing new functionality and vendor specific capabilities. Neutron plugins support new or vendor-specific functionality. Extensions also allow specifying new resources or extensions to existing resources and the actions on these resources. Plugins implement these resources and actions.

This Reference Architecture supports the ML2 plugin (see below) as well as the service plugins including for LBaaS (Load Balancer as a Service) [41], and VPNaaS (VPN as a Service) [42]. The OpenStack wiki provides a list of Neutron plugins [43].

Every Neutron plugin needs to implement a minimum set of common methods (actions for Wallaby release) [44]. Resources can inherit Standard Attributes and thereby have the extensions for these standard attributes automatically incorporated. Additions to resources, such as additional attributes, must be accompanied by an extension.

Interfaces and APIs of this Reference Architecture provides a list of *Neutron Extensions*. The current available extensions can be obtained using the List Extensions API [45] and details about an extension using the Show extension details API [46].

Neutron ML2 integration The OpenStack Modular Layer 2 (ML2) plugin simplifies adding networking technologies by utilising drivers that implement these network types and methods for accessing them. Each network type is managed by an ML2 type driver and the mechanism driver exposes interfaces to support the actions that can be performed on the network type resources. The OpenStack ML2 documentation [47] lists example mechanism drivers.

Network quality of service

For VNF workloads, the resource bottlenecks are not only the CPU and the memory but also the I/O bandwidth and the forwarding capacity of virtual and non-virtual switches and routers within the infrastructure. Several techniques (all complementary) can be used to improve QoS and try to avoid any issue due to a network bottleneck (mentioned per order of importance):

- Nodes interfaces segmentation: Have separated NIC ports for Storage and Tenant networks. Actually, the storage traffic is bursty, and especially in case of service restoration after some failure or new service implementation, upgrades, etc. Control and management networks should rely on a separate interface from the interface used to handle tenant networks.
- Capacity planning: FW, physical links, switches, routers, NIC interfaces and DCGW dimensioning (+ load monitoring: each link within a LAG or a bond shouldn't be loaded over 50% of its maximum capacity to guaranty service continuity in case of individual failure).
- Hardware choice: e.g., ToR/fabric switches, DCGW and NIC cards should have appropriate buffering and queuing capacity.
- High Performance compute node tuning (including OVS-DPDK).

Integration Interfaces

- DHCP:

When the Neutron-DHCP agent is hosted in controller nodes, then for the servers, on a Tenant network, that need to acquire an IPv4 and/or IPv6 address, the VLAN for the Tenant must be extended to the control plane servers so that the Neutron agent can receive the DHCP requests from the server and send the response to the server with the IPv4 and/or IPv6 addresses and the lease time. Please see OpenStack provider Network.

- DNS
- LDAP
- IPAM

4.1.4 Storage Backend

Storage systems are available from multiple vendors and can also utilise commodity hardware from any number of open-source based storage packages (such as LVM, Ceph, NFS, etc.). The proprietary and open-source storage systems are supported in Cinder through specific plugin drivers. The OpenStack Cinder documentation [48] specifies the minimum functionality that all storage drivers must support. The functions include:

- Volume: create, delete, attach, detach, extend, clone (volume from volume), migrate
- Snapshot: create, delete and create volume from snapshot
- Image: create from volume

The document also includes a matrix for a number of proprietary drivers and some of the optional functions that these drivers support. This matrix is a handy tool to select storage backends that have the optional storage functions needed by the cloud operator. The cloud workload storage requirements helps determine the backends that should be deployed by the cloud operator. The common storage backend attachment methods include iSCSI, NFS, local disk, etc. and the matrix lists the supported methods for each of the vendor drivers. The OpenStack Cinder Available Drivers [49] documentation provides a list of all OpenStack compatible drivers and their configuration options.

The Cinder Configuration [50] document provides information on how to configure Cinder including Anuket required capabilities for volume encryption, Policy configuration, quotas, etc. The Cinder Administration [51] document provides information on the capabilities required by Anuket including managing volumes, snapshots, multi-storage backends, migrate volumes, etc.

Ceph [52] is the default Anuket Reference Architecture storage backend and is discussed below.

Ceph Storage Cluster

The Ceph storage cluster is deployed on bare metal hardware. The minimal configuration is a cluster of three bare metal servers to ensure High availability. The Ceph Storage cluster consists of the following components:

- CEPH-MON (Ceph Monitor)
- OSD (object storage daemon)
- RadosGW (Rados Gateway)
- Journal
- Manager

Ceph monitors maintain a master copy of the maps of the cluster state required by Ceph daemons to coordinate with each other. Ceph OSD handles the data storage (read/write data on the physical disks), data replication, recovery, rebalancing, and provides some monitoring information to Ceph Monitors. The RadosGW provides Object Storage RESTful gateway with a Swift-compatible API for Object Storage.



Figure 4.6: Ceph Storage System

BIOS Requirement for Ceph servers

Table 4.15: BIOS Requirement for Ceph servers

BIOS/boot Parameter	Value
Boot disks	RAID 1

How many nodes to meet SLA :

- minimum: three bare metal servers where Monitors are collocated with OSD. Note: at least 3 Monitors and 3 OSDs are required for High Availability.

HW specifications :

- Boot disks are dedicated with Flash technology disks
- For an IOPS oriented cluster (Flash technology), the journal can be hosted on OSD disks
- For a capacity-oriented cluster (HDD), the journal must be hosted on dedicated Flash technology disks

Sizing rules :

- Minimum of 6 disks per server
- Replication factor : 3
- 1 Core-GHz per OSD
- 16GB RAM baseline + 2-3 GB per OSD

4.2 VIM OpenStack Services

This section covers:

- Detailed breakdown of OpenStack core services
- Specific build-time parameters

4.2.1 VIM Services

A high-level overview of the core OpenStack Services was provided in *Virtualised Infrastructure Manager (VIM)*. In this section we describe the core and other needed services in more detail.

Keystone

Keystone [53] is the authentication service, the foundation of identity management in OpenStack. Keystone needs to be the first deployed service. Keystone has services running on the control nodes and no services running on the compute nodes:

- Keystone admin API
- Keystone public API - in Keystone V3 this is the same as the admin API

Glance

Glance [54] is the image management service. Glance has only a dependency on the Keystone service therefore it is the second one deployed. Glance has services running on the control nodes and no services running on the compute nodes:

- Glance API
- Glance Registry

The Glance backends include Swift, Ceph RBD, and NFS.

Cinder

Cinder [55] is the block device management service, depends on Keystone and possibly Glance to be able to create volumes from images. Cinder has services running on the control nodes and no services running on the compute nodes:
- Cinder API - Cinder Scheduler - Cinder Volume - the Cinder volume process needs to talk to its backends

The Cinder backends include SAN/NAS storage, iSCSI drives, Ceph RBD, and NFS.

Swift

Swift [56] is the object storage management service, Swift depends on Keystone and possibly Glance to be able to create volumes from images. Swift has services running on the control nodes and the compute nodes:

- Proxy Services
- Object Services
- Container Services
- Account Services

The Swift backends include iSCSI drives, Ceph RBD, and NFS.

Neutron

Neutron [57] is the networking service, depends on Keystone and has services running on the control nodes and the compute nodes. Depending upon the workloads to be hosted by the Infrastructure, and the expected load on the controller node, some of the Neutron services can run on separate network node(s). Factors affecting controller node load include number of compute nodes and the number of API calls being served for the various OpenStack services (nova, neutron, cinder, glance etc.). To reduce controller node load, network nodes are widely added to manage L3 traffic for overlay tenant networks and interconnection with external networks. The Table below lists the networking service components and their placement. Please note that while network nodes are listed in the table below, network nodes only deal with tenant networks and not provider networks. Also, network nodes are not required when SDN is utilised for networking.

Table 4.16: Neutron Services Placement

Networking Service component	Description	Required or Optional Service	Placement
neutron server (neutron-server and neutron-*.plugin)	Manages user requests and exposes the Neutron APIs	Required	Controller node
DHCP agent (neutron-dhcp-agent)	Provides DHCP services to tenant networks and is responsible for maintaining DHCP configuration. For High availability, multiple DHCP agents can be assigned.	Optional depending upon plug-in	Network node (Controller node if no network node present)
L3 agent (neutron-l3-agent)	Provides L3/NAT forwarding for external network access of servers on tenant networks and supports services such as Firewall-as-a-service (FWaaS) and Load Balancer-as-a-service (LBaaS)	Optional depending upon plug-in	Network node (Controller node if no network node present) NB in DVR based OpenStack Networking, also in all Compute nodes.
neutron metadata agent (neutron-metadata-agent)	The metadata service provides a way for instances to retrieve instance-specific data. The networking service, neutron, is responsible for intercepting these requests and adding HTTP headers which uniquely identify the source of the request before forwarding it to the metadata API server. These functions are performed by the neutron metadata agent.	Optional	Network node (Controller node if no network node present)
neutron plugin agent (neutron-*.agent)	Runs on each compute node to control and manage the local virtual network driver (such as the Open vSwitch or Linux Bridge) configuration and local networking configuration for servers hosted on that node.	Required	Every Compute Node

Issues with the standard networking (centralised routing) approach

The network node performs both routing and NAT functions and represents both a scaling bottleneck and a single point of failure.

Consider two servers on different compute nodes and using different project networks (a.k.a. tenant networks) where the both of the project networks are connected by a project router. For communication between the two servers (instances with a fixed or floating IP address), the network node routes East-West network traffic among project networks using the same project router. Even though the instances are connected by a router, all routed traffic must flow through the network node, and this becomes a bottleneck for the whole network.

While the separation of the routing function from the controller node to the network node provides a degree of scaling it is not a truly scalable solution. We can either add additional cores/compute-power or network node to the network node cluster, but, eventually, it runs out of processing power especially with high throughput requirement. Therefore, for scaled deployments, there are multiple options including use of Dynamic Virtual Routing (DVR) and Software Defined Networking (SDN).

Distributed Virtual Routing (DVR)

With DVR, each compute node also hosts the L3-agent (providing the distributed router capability) and this then allows direct instance to instance (East-West) communications.

The OpenStack “High Availability Using Distributed Virtual Routing (DVR) [58]” provides an in-depth view into how DVR works and the traffic flow between the various nodes and interfaces for three different use cases. Please note that DVR was introduced in the OpenStack Juno release and, thus, its detailed analysis in the Liberty release documentation is not out of character for OpenStack documentation.

DVR addresses both scalability and high availability for some L3 functions but is not fully fault tolerant. For example, North/South SNAT traffic is vulnerable to single node (network node) failures. DVR with VRRP [59] addresses this vulnerability.

Software Defined Networking (SDN)

For the most reliable solution that addresses all the above issues and Telco workload requirements requires SDN to offload Neutron calls.

SDN provides a truly scalable and preferred solution to support dynamic, very large-scale, high-density, telco cloud environments. OpenStack Neutron, with its plugin architecture, provides the ability to integrate SDN controllers (*Virtual Networking - 3rd party SDN solution*). With SDN incorporated in OpenStack, changes to the network is triggered by workloads (and users), translated into Neutron APIs and then handled through neutron plugins by the corresponding SDN agents.

Nova

Nova [60] is the compute management service, depends on all above components and is deployed after their deployment. Nova has services running on the control nodes and the compute nodes:

- nova-metadata-api
- nova-compute api
- nova-consoleauth
- nova-scheduler
- nova-conductor
- nova-novncproxy
- nova-compute-agent which runs on Compute node

Please note that the Placement-API must have been installed and configured prior to nova compute starts.

Ironic

Ironic [61] is the bare metal provisioning service. Ironic depends on all above components and is deployed after them. Ironic has services running on the control nodes and the compute nodes:

- Ironic API
- ironic-conductor which executes operation on bare metal nodes

Note: This is an optional service. The Ironic APIs [62] are still under development.

Heat

Heat [63] is the orchestration service using templates to provision cloud resources, Heat integrates with all OpenStack services. Heat has services running on the control nodes and no services running on the compute nodes:

- heat-api
- heat-cfn-api
- heat-engine

Horizon

Horizon [64] is the Web User Interface to all OpenStack services. Horizon has services running on the control nodes and no services running on the compute nodes.

Placement

The OpenStack Placement service [65] enables tracking (or accounting) and scheduling of resources. It provides a RESTful API and a data model for the managing of resource provider inventories and usage for different classes of resources. In addition to standard resource classes, such as vCPU, MEMORY_MB and DISK_GB, the Placement service supports custom resource classes (prefixed with “CUSTOM_”) provided by some external resource pools such as a shared storage pool provided by, say, Ceph. The placement service is primarily utilised by nova-compute and nova-scheduler. Other OpenStack services such as Neutron or Cyborg can also utilise placement and do so by creating Provider Trees [66]. The following data objects are utilised in the placement service [67]:

- Resource Providers provide consumable inventory of one or more classes of resources (CPU, memory or disk). A resource provider can be a compute host, for example.
- Resource Classes specify the type of resources (vCPU, MEMORY_MB and DISK_GB or CUSTOM_*)
- Inventory: Each resource provider maintains the total and reserved quantity of one or more classes of resources. For example, RP_1 has available inventory of 16 vCPU, 16384 MEMORY_MB and 1024 DISK_GB.
- Traits are qualitative characteristics of the resources from a resource provider. For example, the trait for RPA_1 “is_SSD” to indicate that the DISK_GB provided by RP_1 are solid state drives.
- Allocations represent resources that have been assigned/used by some consumer of that resource.
- Allocation candidates is the collection of resource providers that can satisfy an allocation request.

The Placement API is stateless and, thus, resiliency, availability and scaling, it is possible to deploy as many servers as needed. On start, the nova-compute service will attempt to make a connection to the Placement API and keep attempting to connect to the Placement API, logging and warning periodically until successful. Thus, the Placement API must be installed and enabled prior to Nova compute.

Placement has services running on the control node: - nova-placement-api

Barbican

Barbican [68] is the OpenStack Key Manager service. It is an optional service hosted on controller nodes. It provides secure storage, provisioning, and management of secrets as passwords, encryption keys and X.509 Certificates. Barbican API is used to centrally manage secrets used by OpenStack services, e.g., symmetric encryption keys used for Block storage encryption or Object Storage encryption or asymmetric keys and certificates used for Glance image signing and verification.

Barbican usage provides a means to fulfill security requirements such as sec.sys.012 “The Platform **must** protect all secrets by using strong encryption techniques and storing the protected secrets externally from the component” and sec.ci.001 “The Platform **must** support Confidentiality and Integrity of data at rest and in transit.”.

Cyborg

Cyborg [69] is the OpenStack project for the general purpose management framework for accelerators (including GPUs, FPGAs, ASIC-based devices, etc.), and their lifecycle management.

Cyborg will support only a subset of the Nova operations [70]; the set of Nova operations supported in Cyborg depends upon the merge of a set of Nova patches in Cyborg. In Wallaby, not all the required Nova patches have been merged. The list of Cyborg operations with Nova dependencies supported in Wallaby are listed in [71]; the Nova operations supported in Cyborg at any given time are also available in [72].

Cyborg supports:

- Acceleration Resource Discovery
- Accelerator Life Cycle Management

Accelerators can be of type:

- Software: dpdk/spdk, pmem, ...
- Hardware (device types): FPGA, GPU, ARM SoC, NVMe SSD, CCIX based Caches, ...

The Cyborg architecture [73] consists of the cyborg-api, cyborg-conductor, cyborg-db, cyborg-agent, and generic device type drivers. cyborg-api, cyborg-conductor and cyborg-db are hosted on control nodes. cyborg-agent, which runs on compute nodes, interacts with generic device type drivers on those nodes. These generic device type drivers are an abstraction of the vendor specific drivers; there is a generic device type driver for each device type (see above for list of some of the device types). The current list of the supported vendor drivers is listed under “Driver Support [72]”.

4.2.2 Containerised OpenStack Services

Containers are lightweight compared to Virtual Machines and leads to efficient resource utilisation. Kubernetes auto manages scaling, recovery from failures, etc. Thus, it is recommended that the OpenStack services be containerised for resiliency and resource efficiency.

The Chapter 3 shows a high level Virtualised OpenStack services topology. The containerised OpenStack services topology version is shown in the figure below.



Figure 4.7: Containerised OpenStack Services Topology

4.3 Consumable Infrastructure Resources and Services

4.3.1 Support for Cloud Infrastructure Profiles and flavors

Chapters 4 and 5 in [1] provide information about the Cloud Infrastructure Profiles and their size information. OpenStack flavors with their set of properties describe the server capabilities and size required to determine the compute host which will run this server. The set of properties must match compute profiles available in the infrastructure. To implement these profiles and sizes, it is required to set up the flavors as specified in the tables below.

Table 4.17: Neutron Services Placement

Flavor Capabilities	Reference RM Chapter 4 and 5	Basic	High-Performance
CPU allocation ratio (custom extra_specs)	in-fra.com.cfg.001	In flavor create or flavor set -property cpu_allocation_ratio=4.0	In flavor create or flavor set -property cpu_allocation_ratio=1.0
NUMA Awareness	in-fra.com.cfg.002		In flavor create or flavor set specify -property hw:numa_nodes=<integer range of 0 to #numa_nodes - 1>. To restrict an instance's vCPUs to a single host NUMA node, specify: -property hw:numa_nodes=1. Some compute intensive* workloads with highly sensitive memory latency or bandwidth requirements, the instance may benefit from spreading across multiple NUMA nodes: -property hw:numa_nodes=2
CPU Pinning	in-fra.com.cfg.003	In flavor create or flavor set specify -property hw: cpu_policy=shared (default)	In flavor create or flavor set specify -property hw:cpu_policy=dedicated and -property hw:cpu_thread_policy=<prefer, require, isolate>. Use "isolate" thread policy for very high compute intensive workloads that require that each vCPU be placed on a different physical core
Huge pages	in-fra.com.cfg.004		-property hw:mem_page_size=<small large size>
SMT	in-fra.com.cfg.005		In flavor create or flavor set specify -property hw:cpu_threads=<integer#threads (usually 1 or 2)>
OVS-DPDK	infra.net.acc.cfg.001		ml2.conf.ini configured to support [OVS] datapath_type=netdev Note: huge pages should be configured to large
Local Storage SSD	infra.hw.stg.ssd.cfg.002	trait:STORAGEDISK_SSD=required	trait:STORAGE_DISK_SSD=required
Port speed	infra.hw.nic.cfg.002	-property quota vif_inbound_average=1310720 and vif_outbound_average=1310720. Note:10 Gbps = 1250000 kilobytes per second	-property quota vif_inboundaverage=3125000 and vif_outbound_average=3125000 Note: 25 Gbps = 3125000 kilobytes per second

- To configure profile-extensions, for example, the "Storage Intensive High Performance" profile, as defined in Profile Extensions (Specialisations) [1], in addition to the above, need to configure the storage IOPS: the following two parameters need to be specified in the flavor create: -property quota:disk_write_iops_sec=<IOPS#> and -property quota:disk_read_iops_sec=<IOPS#>.

The flavor create command and the mandatory and optional configuration parameters is documented in OpenStack Flavors [74].

4.3.2 Logical segregation and high availability

To ensure logical segregation and high availability, the architecture will rely on the following principles:

- Availability zone: provide resiliency and fault tolerance for VNF deployments, by means of physical hosting distribution of compute nodes in separate racks with separate power supply, in the same or different DC room
- Affinity-groups: allow tenants to make sure that VNFC instances are on the same compute node or are on different compute nodes.

Note: The Cloud Infrastructure doesn't provide any resiliency mechanisms at the service level. Any server restart shall be triggered by the VNF Manager instead of OpenStack:

- It doesn't implement Instance High Availability which could allow OpenStack Platform to automatically re-spawn instances on a different compute node when their host compute node breaks.
- Physical host reboot does not trigger automatic server recovery.
- Physical host reboot does not trigger the automatic start of a server.

Limitations and constraints

- NUMA Overhead: isolated core will be used for overhead tasks from the hypervisor.

4.3.3 Transaction Volume Considerations

Storage transaction volumes impose a requirement on North-South network traffic in and out of the storage backend. Data availability requires that the data be replicated on multiple storage nodes and each new write imposes East-West network traffic requirements.

4.4 Cloud Topology and Control Plane Scenarios

Typically, Clouds have been implemented in large (central) data centres with hundreds to tens of thousands of servers. Telco Operators have also been creating intermediate data centres in central office locations, colocation centres, and now edge centres at the physical edge of their networks because of the demand for low latency and high throughput for 5G, IoT and connected devices (including autonomous driverless vehicles and connected vehicles). *Cloud Topology* discusses and lists 3 types of data centres: Large, Intermediate and Edge.

For ease of convenience, unless specifically required, in this section we will use Central Cloud Centre, Edge Cloud Centre and Intermediate Cloud Centre as representative terms for cloud services hosted at centralised large data centres, Telco edge locations and for locations with capacity somewhere in between the large data centres and edge locations, respectively. The mapping of various terms, including the Reference Model terminology specified in the chapter "Comparison of Deployment Topologies and Edge Terms" and Open Glossary of Edge Computing [75], is as follows:

- Central Cloud Centre: Large Centralised Data Centre, Regional Data Centre
- Intermediate Cloud Centre: Metro Data Centre, Regional Edge, Aggregation Edge
- Edge Cloud Centre: Edge, Mini-/Micro-Edge, Micro Modular Data Centre, Service Provider Edge, Access Edge, Aggregation Edge

In the Intermediate and Edge cloud centres, there may be limitations on the resource capacity, as in the number of servers, and the capacity of these servers in terms of # of cores, RAM, etc. restricting the set of services that can be deployed and, thus, creating a dependency between other data centres. "Telco Edge Cloud" chapter in [1] specifies the physical and environmental characteristics, infrastructure capabilities and deployment scenarios of different locations.

OpenStack Services Topology of this document, specifies the differences between the Control Plane and Data Plane, and specifies which of the control nodes, compute nodes, storage nodes (optional) and network nodes (optional) are components of these planes. The previous sections of this Chapter 4 include a description of the OpenStack services

and their deployment in control nodes, compute nodes, and optionally storage nodes and network nodes (rarely). The Control Plane deployment scenarios determine the distribution of OpenStack and other needed services among the different node types. This section considers the Centralised Control Plane (CCP) and Distributed Control Plane (DCP) scenarios. The choice of control plane and the cloud centre resource capacity and capabilities determine the deployment of OpenStack services in the different node types.

The Central Cloud Centres are organised around a Centralised Control Plane. With the introduction of Intermediate and Edge Cloud Centres, the Distributed Control Plane deployment becomes a possibility. A number of independent control planes (sometimes referred to as Local Control Planes (LCP)) exist in the Distributed Control Plane scenario, compared with a single control plane in the Centralised Control Plane scenario. Thus, in addition to the control plane and controller services deployed at the Central Cloud Centre, Local Control Planes hosting a full-set or subset of the controller services are also deployed on the Intermediate and Edge Cloud Centres. The following table presents examples of such deployment choices.

Table 4.18: Distribution of OpenStack services on different nodes depending upon Control Plane Scenario

Control Plane	Deployed in	Orchestration	Identity Management	Image Management	Compute	Network Management	Storage Management
CCP	Centralised DC - control nodes	heat-api, heat-engine, nova-placement-api	Identity Provider (IdP), Keystone API	Glance API, Glance Registry	nova-compute api, nova-scheduler, nova-conductor	neutron-server, neutron-dhcp-agent, neutron-L2-agent, neutron-L3-agent (optional), neutron-metadata-agent	Cinder API, Cinder Scheduler, Cinder Volume
DCP: combination of services depending upon Center size	Any DC - Control nodes Option 1	heat-api, heat-engine, nova-placement-api	Identity Provider (IdP), Keystone API	Glance API, Glance Registry	nova-compute api, nova-scheduler, nova-conductor	neutron-server, neutron-dhcp-agent, neutron-L2-agent, neutron-L3-agent (optional), neutron-metadata-agent	Cinder API, Cinder Scheduler, Cinder Volume
	Any DC - Control nodes Option 2: split services between two or more DCs	in one of the DC	in the Large DC	in the Large DC	in one of the DC	in one of ther DC	in one of the DC
CCP or DCP	Compute nodes				nova-compute-agent	neutron-L2-agent, neutron-L3-agent (optional)	
CCP	Compute nodes	nova-placement-api			nova-compute-agent, nova-conductor	neutron-server, neutron-dhcp-agent, neutron-L2-agent, neutron-L3-agent (optional)	

4.4.1 Edge Cloud Topology

The Reference Model “Telco Edge Cloud” chapter [1] presents the deployment environment characteristics, infrastructure characteristics and new values for the Infrastructure Profiles at the Edge.

The Edge computing whitepaper [76] includes information such as the services that run on various nodes. The information from the whitepaper coupled with that from the OpenStack Reference Architecture [77] for 100, 300 and 500 nodes will help in deciding which OpenStack and other services (such as database, messaging) run on which nodes in what Cloud Centre and the number of copies that should be deployed. These references also present the pros and cons of DCP and CCP and designs to address some of the challenges of each of the models.

“Telco Edge Cloud: Platform Services Deployment” [1] lists the Platform Services that may be placed in the different node types (control, compute and storage). Depending upon the capacity and resources available only the compute nodes may exist at the Edge thereby impacting operations.

“Telco Edge Cloud: Infrastructure Profiles” [1] lists a number of Infrastructure Profile characteristics and the changes that may need to be made for certain Edge clouds depending upon their resource capabilities. It should be noted that none of these changes affect the definition of OpenStack flavors.

The previous section listed the OpenStack services deployed on the controller nodes depending upon the control plane distribution. As specified earlier in this chapter, at least 3 controller nodes should be deployed for HA. Compute nodes may also exist at the sites where controller nodes are deployed.

Control plane services are not hosted at edge sites. Each edge site can be treated as its own OpenStack AZ. The compute nodes, will host *nova-compute*, a component of the the Compute Service (Nova), and *neutron-L2-agent*, a component of the Network Service (Neutron).

The Edge sites may or may not contain local storage. If the edge sites contain storage, then the Block Storage service (Cinder) is usually deployed to run in an active/active mode with the centrally deployed Block Storage service. Instance images are downloaded and stored locally; they can be downloaded even prior to use.

If the edge site doesn’t contain storage then the images would need to be cached from the central site. Two options exist:

- The instance images would be downloaded and cached in the Nova cache on first use; they will then be available for subsequent use.
- Pre-caching the instance images for low time-to-boot latency. This has been supported in Nova since the OpenStack Ussuri release.

Image caching and considerations for its use are discussed in the OpenStack document Image Caching [78].

Edge Cloud Deployment Tools

Deployment at the Edge requires support for large scale deployment. A number of open-source tools are available for the purpose including:

- Airship [79]: declaratively configure, deploy and maintain an integrated virtualisation and containerisation platform
- Starling-X [80]: cloud infrastructure software stack for the edge
- Triple-O [81]: for installing, upgrading and operating OpenStack clouds

These installers are described in more details in

Operations and Life Cycle Management.

5 Interfaces and APIs

This chapter presents a consolidated set of OpenStack Service APIs corresponding to the ETSI NFV Nf-Vi, Vi-Vnfm and Or-Vi interfaces. The OpenStack Wallaby version is used as the baseline for these APIs and CLIs in this Reference Architecture (RA-1) version. Any Cloud Infrastructure + VIM reference implementations that **get certified by RC** (Reference Conformance) can be considered as Anuket RA Conformant.

The Chapter presents the APIs for the core OpenStack services defined in Chapter 3 and a consolidated view of these and other APIs that are of interest.

OpenStack is a multi-project framework composed of independently evolving services. It is not enough to rely only on the OpenStack release to characterise the capabilities supported by these services. Regarding OpenStack services APIs, an “API version” is associated with each OpenStack service. In addition to major API versions, some OpenStack services (Nova, Glance, Keystone, Cinder...) support microversions. The microversions allow new features to be introduced over time. In this chapter, the **major version** and **microversion** are specified per service. The specified microversion is the minimal microversion that supports the features requested for this RA. For the purpose of conformance tests, this chapter also identifies the set of features, offered by a service, that are mandatory for Anuket compliant implementation.

5.1 Core OpenStack Services APIs

Please note that OpenStack provides a maximum microversion to be used with an OpenStack release. In the following sections the “Maximal API Version” refers to this maximum microversion specified for the OpenStack Wallaby release. Please note that in Reference Conformance (RC-1) testing, the System Under Test (SUT) can utilise newer microversions because of the OpenStack microversion policies. As per multiple OpenStack services documentation, for example the Compute Service [82], “A cloud that is upgraded to support newer microversions will still support all older microversions to maintain the backward compatibility for those users who depend on older microversions.”

5.1.1 Keystone API

Table 5.1: Keystone

OpenStack Service	API Version	Maximal API Microversion
Identity: Keystone	v3	3.14

Table 5.2: Keystone Features

Keystone Features	Mandatory
access_rules	
application_credentials	X
external_idp	
federation	
oauth1	
project_tags	X
security_compliance	X
trust	X

Identity API v3: [83]

Identity API v3 extensions: [84]

Security compliance and PCI-DSS: [85]

5.1.2 Glance API

Table 5.3: Glance

OpenStack Service	API Version	Maximal API Microversion
Image: Glance	v2	2.9

Table 5.4: Glance Features

Glance Features	Mandatory
import_image	
os_glance_reserved	
web-download import	

Image Service API: [86]

Image Service Versions: [87]

5.1.3 Cinder API

Table 5.5: Cinder

OpenStack Service	API Version	Maximal API Microversion
Block Storage: Cinder	v3	3.64

Table 5.6: Cinder Features

Cinder Features	Mandatory
backup	X
clone	X
consistency_group	
extend_attached_volume	
extend_attached_encrypted_volume	
manage_snapshot	X
manage_volume	X
multi_backend	
snapshot	X
volume_revert	X

Block Storage API: [88]

Cinder REST API Version History: [89]

5.1.4 Swift API

Table 5.7: Swift

OpenStack Service	API Version
Object Storage: Swift	v1

Table 5.8: Swift Features

Swift Features	Mandatory
account_quotas	X
bulk_delete	X
bulk_upload	X
container_quotas	X
container_sync	
crossdomain	X
discoverability	X
form_post	X
ratelimit	X
s3api	
slo	X
staticweb	X
symlink	X
temp_url	X
tempauth	X
versioned_writes	X

Object Storage API: [90]

Discoverability: [91]

5.1.5 Neutron API

Table 5.9: Neutron

OpenStack Service	API Version
Networking: Neutron	v2.0

Table 5.10: Neutron Extensions

Neutron Extensions	Mandatory
address-scope	X
agent	X
allowed-address-pairs	X
auto-allocated-topology	X
availability_zone	X
availability_zone_filter	X
binding	X
binding-extended	X
default-subnetpools	X
dhcp_agent_scheduler	

continues on next page

Table 5.10 – continued from previous page

Neutron Extensions	Mandatory
dns-domain-ports	
dns-integration	
dvr	
empty-string-filtering	X
ext-gw-mode	X
external-net	X
extra_dhcp_opt	X
extraroute	X
extraroute-atomic	
flavors	X
filter-validation	
fip-port-details	
floating-ip-port-forwarding	
floatingip-pools	
ip-substring-filtering	X
l3_agent_scheduler	
l3-flavors	
l3-ha	
logging	
metering	
multi-provider	X
net-mtu	X
net-mtu-writable	X
network_availability_zone	X
network-ip-availability	X
network-segment-range	
pagination	X
port-mac-address-regenerate	
port-resource-request	
port-security	X
port-security-groups-filtering	X
project-id	X
provider	X
rbac-policies	X
router	X
router_availability_zone	X
qos	X
qos-bw-limit-direction	X
qos-bw-minimum-ingress	X
qos-default	X
qos-fip	X
qos-gateway-ip	X
qos-rule-type-details	X
qos-rules-alias	X
quotas	X
quota_details	X
revision-if-match	X
rbac-address-scope	
rbac-security-groups	

continues on next page

Table 5.10 – continued from previous page

Neutron Extensions	Mandatory
rbac-subnetpool	
router-interface-fip	
security-group	X
service-type	X
sorting	X
standard-attr-description	X
standard-attr-revisions	X
standard-attr-tag	X
standard-attr-timestamp	X
subnet_allocation	X
subnet-service-types	X
subnetpool-prefix-ops	
tag-ext	
tag-ports-during-bulk-creation	
trunk	X
trunk-details	X
uplink-status-propagation	

Table 5.11: Neutron Type Drivers

Neutron Type Drivers	Mandatory
geneve	
gre	
vlan	X
vxlan	

Networking Service APIs: [92]

The exhaustive list of extensions is available at [93]

5.1.6 Nova API

Table 5.12: Nova

OpenStack Service	API Version	Maximal API Microversion
Compute: Nova	v2.1	2.88

Table 5.13: Nova Features

Nova Features	Mandatory
attach_encrypted_volume	
cert	
change_password	
cold_migration	X
console_output	X
disk_config	X
instance_password	X
interface_attach	X
live_migration	X
metadata_service	X
pause	X
personality	
rdp_console	
rescue	X
resize	X
serial_console	
shelve	X
shelve_migrate	
snapshot	X
stable_rescue	
spice_console	
suspend	X
swap_volume	
vnc_console	
volume_multiattach	
xenapi_apis	

Compute API: [94]

Compute REST API Version History: [95]

5.1.7 Placement API

Table 5.14: Placement

OpenStack Service	API Version	Maximal API Microversion
Placement	v1	1.36

Placement API: [96]

Placement REST API Version History: [97]

5.1.8 Heat API

Table 5.15: Heat

OpenStack Service	API Version	Maximal Template Version
Orchestration: Heat	v1	2021-04-16

Orchestration Service API: [98]

Template version history: [99]

Heat Orchestration Template (HOT) specification: [100]

5.2 Consolidated Set of APIs

5.2.1 OpenStack Interfaces

This section illustrates some of the Interfaces provided by OpenStack; the exhaustive list of APIs is available at [101].

OpenStack REST APIs are simple to interact with using either of two options. Clients can either call the APIs directly using the HTTP or REST library, or they can use one of the many cloud specific programming language libraries.

APIs

Table 5.16: APIs

OpenStack Service	Link for API list	API Version	Maximal API Microversion
Identity: Keystone	[83]	v3	3.14
Compute: Nova	[94]	v2.1	2.88
Networking: Neutron	[93]	v2.0	
Image: Glance	[86]	v2	2.9
Block Storage: Cinder	[88]	v3	3.64
Placement	[96]	v1	1.36
Orchestration: Heat	[98]	v1	2021-04-06 (template)

5.2.2 Kubernetes Interfaces

The Kubernetes APIs are available at [102].

5.2.3 KVM Interfaces

The KVM APIs are documented in Section 4 of the document [103].

Libvirt Interfaces

The Libvirt APIs are documented in [104].

5.2.4 Barbican API

Table 5.17: Barbican

OpenStack Service	API Version
Key Manager: Barbican	v1

Barbican API: [105]

6 Security

This guide is intended to provide basic security requirements to architects who are implementing Cloud Infrastructure using OpenStack [2] technology. This is a minimal set of high-level general security practices, not intended to cover all implementation scenarios. Please ensure to also reference your enterprise security and compliance requirements in addition to this guide.

6.1 Security Requirements

Cloud Infrastructure Security Requirements and *Security Recommendations* gather all requirements and recommendations regarding security topics developed in this chapter.

6.2 Cloud Infrastructure and VIM Security

In the “Security boundaries and threats” section [106] of the OpenStack security guide [107], there is extensive description on security domains, threat classifications, and attack vectors. The following only touches on some of the topics and at a high level.

6.2.1 System Hardening

All infrastructure components should undergo system hardening, establish processes to govern the hardening, and documents to cover at a minimal for the following areas.

Server boot hardening

Server boot process must be trusted. For this purpose, the integrity and authenticity of all BIOS firmware components must be verified at boot. Per sec.gen.003 requirement, Secure Boot based on UEFI must be used. By verifying the signatures of all BIOS components, Secure Boot will ensure that servers start with the firmware expected and without malware insertion into the system. Secure Boot checks the digital signatures locally. To implement a chain of trust, Secure Boot must be complemented by the use of a hardware based Root of Trust provided by a TPM (Trusted Platform Module).

System Access

Access to all the platform's components must be restricted (sec.gen.013) applying the following rules:

- Remove, or at a minimal, disable all unnecessary user accounts
- Change all default user accounts where technically feasible
- Change all default credentials
- Prohibit logging with root account when root privileges are not required (sec.gen.006)
- Restrict access according to only those protocols/service/address adhering to the Principle of Least Privilege
- The same authentication credentials must not be reused on different components (sec.sys.011)
- Restrict access to Operating System (sec.gen.005)

Password policy

For all infrastructure components, passwords must be hardened, and a strict password policy must be applied (sec.gen.002).

Passwords must be strengthened:

- All vendors default passwords must be changed
- Passwords must contain at least 8 characters as a minimal value, 14 characters length passwords are recommended
- Passwords must contain at least one upper case letter, one lower case letter and one non-alphabetic character
- For administration privileges accounts, passwords must contain at least one upper case letter, one lower case letter, one numeral and one special (non-alphanumeric) character

For passwords updates, the identity of users must be verified before permitting a password change.

Passwords must be encrypted at rest and in-transit. Password files must be stored separately from application system data.

Password's composition, complexity and policy should follow the recommendations consolidated within the CIS Password Policy guide [8] such as:

- Check the password for known bad passwords (repetitive or sequential characters, dictionary words, context-specific words, previously used passwords, etc.)
- Limit number of failed login attempts
- Implement Multi-factor Authentication
- Periodic (for example, Yearly, Quarterly, etc.) password change or on key events such as indication of compromise, change of user roles, a defined period of inactivity, when a user leaves the organisation, etc.

Function and Software

Infrastructure must be implemented to perform the minimal functions needed to operate the Cloud Infrastructure.

Regarding software (sec.gen.004):

- Install only software which is required to support the functions
- Remove any unnecessary software or packages
- Where software cannot be removed, disable all services to it

Patches

All deployed Cloud Infrastructure software must be audited and must be implemented to allow installation of the latest patches to address security vulnerabilities in the following timescale from discovery (sec.gen.008, sec.lcm.011):

Table 6.1: Time to Remediate

Severity	Time to Remediate
Zero-Day	Immediately or as soon as practically possible
Critical	30 days
High	60 days
Medium	90 days
Low	180 days

See Common Vulnerability Scoring System [108] and NIST Vulnerability Metrics [109].

Network Protocols

- Only allow protocols that are required by the system functions (sec.sys.002)
- Tighten all required TCP/IP (Transmission Control Protocol/Internet Protocol) services

Anti-Virus and Firewall

- Install and run your Enterprise approved anti-virus software/ intrusion protection/ malware/ spyware endpoint security software with up-to-date profiles; minimal daily refresh
- Install and run firewall software where applicable

Vulnerability Detection and Prevention

- Implement DoS (Denial of Service) protection where applicable
- Ensure logging and alerting is actively running
- Run host-based scanning and fix all findings per vulnerability severity
- Run network-based scanning and fix all findings per vulnerability severity

6.2.2 Platform Access

Identity Security

The OpenStack Identity service (Keystone) [110] provides identity, token, catalog, and policy services for use specifically by services in the OpenStack family. Identity service is organised as a group of internal services exposed on one or many endpoints. Many of these services are used in a combined fashion by the front end (sec.sys.006).

OpenStack Keystone can work with an Identity service that your enterprise may already have, such as LDAP with Active Directory. In those cases, the recommendation is to integrate Keystone with the cloud provider's Identity Services.

Authentication

Authentication is the first line of defence for any real-world implementation of OpenStack. At its core, authentication is the process of confirming the user logging in is who they claim to be. OpenStack Keystone supports multiple methods of authentication, such as username/password, LDAP, and others. For more details, please refer to OpenStack Authentication Methods [111].

Limiting the number of repeated failed login attempts (configurable) reduces the risk of unauthorised access via password guessing (Bruce force attack) - sec.mon.006. The restriction on the number of consecutive failed login attempts ("lockout_failure_attempts") and any actions post such access attempts (such as locking the account where the "lockout_duration" is left unspecified) should abide by the operator's policies. For example, an operator may restrict the number of consecutive failed login attempts to 3 ("lockout_failure_attempts = 3") and lock the account preventing any further access and where the account is unlocked by getting necessary approvals.

Keystone Tokens

Once a user is authenticated, a token is generated for authorisation and access to an OpenStack environment and resources. By default, the token is set to expire in one hour. This setting can be changed based on the business and operational needs, but it's highly recommended to set the expiration to the shortest possible value without dramatically impacting your operations.

Special Note on Logging Tokens: since the token would allow access to the OpenStack services, it *MUST* be masked before outputting to any logs.

Authorisation

Authorisation serves as the next level of defence. At its core, it checks if the authenticated users have the permission to execute an action. Most Identity Services support the notion of groups and roles. A user belongs to groups and each group has a list of roles that permits certain actions on certain resources. OpenStack services reference the roles of the user attempting to access the service. OpenStack policy enforcer middleware takes into consideration the policy rules associated with each resource and the user's group/roles and association to determine if access will be permitted for the requested resource. For more details on policies, please refer to the OpenStack Policies [112].

RBAC

In order to properly manage user access to OpenStack services, service providers must utilise the Role Based Access Control (RBAC) system (sec.sys.001, sec.sys.007). Based on the OpenStack Identity Service (Keystone v3) Group and Domain component, the RBAC system implements a set of access roles that accommodate most use cases. Operations staff can create users and assign them to roles using standard OpenStack commands for users, groups, and roles.

Keystone provides three default roles [113] admin, member, and reader. As of Train release, Keystone applies the following personas consistently across its API.

- The reader role provides read-only access to resources within the system, a domain, or a project.
- The member role is the same as reader in Keystone, but allows to introduce granularity between admin and reader to other OpenStack services.
- The admin role is reserved for the most privileged operations within a given scope for managing resources.

For specific use-case, policies can be overridden, and new roles can be created for each OpenStack service by editing the policy.json file.

Rules

The following rules govern create, read, update, and delete (CRUD) level access.

- *member* can create, read, update, and delete the resources defined at the tenant level.
- *support_member* can create and read the resources defined at the tenant level.
- *viewer* can read the resources defined at the tenant level.
- *admin* can create, read, update, and delete all resources.

Recommended Default Roles to Start

site_admin (HIGHLY RESTRICTED)

- *Site Level Super Admin* - usually assign to Operation Staffs who already have root level access to hosts
- Permission to create/read/update/delete all tenants and resources at the site, including creating snapshot and upload public images
- Limited ability to create/read/update/delete tenant projects

site_admin_support

- *Site Level Admin* - usually assign to Operation Staffs who need to manage resource except delete
- Permission to create/read/update all tenants and resources at the site
- Cannot create snapshots

site_admin_viewer

- *Site Level Admin Read Only* - usually assign to groups who need to view all resources, such as Capacity Planners
- Permission to read all tenants and resources at the site
- Cannot create/update/delete

site_image_manager

- Site wide admin level privileges to Glance API (via CLI)

- Restricted to Image team

tenant_member

- *Tenant Level Admin* - typically assign to majority of tenant users to manage their resources
- Permission to create/read/update/delete to all resources at the tenant project level
- Cannot upload image or create snapshot
- Cannot touch any other tenant except the one the role is located

tenant_snapshot_member

- *Tenant Level Admin with Snapshot* - typically assign to tenant users who need to create snapshot via special request to Operations Staff
- Permission is same as tenant_member except the user can also create snapshots

tenant_support_member

- *Tenant Level Support* - typically assign to tenant users who need to create resource in the project space
- Permission to create/read all resources at the tenant project level
- Cannot update/delete or create snapshots

tenant_viewer

- *Tenant Level Read Only* - typically assign to tenant users who need to read all resources in the project space
- Permission to read all resources at the tenant level
- Cannot create/update/delete

6.2.3 Confidentiality and Integrity

Confidentiality implies that data and resources must be protected against unauthorised introspection/exfiltration. Integrity implies that the data must be protected from unauthorised modifications or deletions.

Regarding confidentiality and integrity in Cloud Infrastructure, 2 main concerns are raised:

- confidentiality and integrity of the Cloud Infrastructure components (networks, hypervisor, OpenStack services)
- confidentiality and integrity of the tenant's data

The Cloud Infrastructure must also provide the mechanism to identify corrupted data.

Confidentiality and Integrity of communications (sec.ci.001)

It is essential to secure the infrastructure from external attacks. To counter this threat, API endpoints exposed to external networks must be protected by either a rate-limiting proxy or web application firewall (WAF), and must be placed behind a reverse HTTPS proxy (sec.mon.008). Attacks can also be generated by corrupted internal components, and for this reason, it is security best practice to ensure integrity and confidentiality of all network communications (internal and external) by using Transport Layer Security (TLS) protocol (sec.sys.003, sec.sys.004). When using TLS, according to the OpenStack security guide [114] recommendation, the minimum version to be used is TLS 1.2.

3 categories of traffic will be protected using TLS:

- traffic from and to external domains
- communications between OpenStack components (OpenStack services, Bus message, Data Base)
- management traffic

Certificates used for TLS encryption must be compliant with X.509 standards and be signed by a trusted authority (sec.sys.017). To issue certificates for internal OpenStack users or services, the cloud provider can use a Public Key Infrastructure (PKI) with its own internal Certification Authority (CA), certificate policies, and management.

Integrity of OpenStack components configuration

The cloud deployment components/tools store all the information required to install the infrastructure including sensitive information such as credentials. It is recommended to turn off deployment components after deployment to minimise the attack surface area, limit the risk of compromise, and to deploy and provision the infrastructure through a dedicated network (VLAN).

Configuration files contain sensitive information. These files must be protected from malicious or accidental modifications or deletions by configuring strict access permissions for such files. All access, failed attempts to change and all changes (pre-change, post-change and by who) must be securely logged, and all failed access and failed changes must be alerted on (sec.mon.005).

The Cloud Infrastructure must provide the mechanisms to identify corrupted data (sec.gen.009):

- the integrity of configuration files and binaries must be checked by using cryptographic hash
- it is recommended to run scripts (such as checksec.sh) to verify the properties of the QEMU/KVM
- it is recommended to use tools such as CIS-CAT (Center for Internet security- Configuration Assessment Tool [115]) to check the compliance of systems configuration against respective CIS benchmarks [116].

It is strongly recommend to protect all repositories, such as Linux repositories and Docker registries, against the corruption of their data and unauthorised access, by adopting protection measures such as hosting a local repository/registry with restricted and controlled access, and using TLS (sec.img.004, sec.img.005, sec.img.006). This repository/registry must contain only signed images or packages.

Confidentiality and Integrity of tenant data (sec.ci.001)

Tenant data are forwarded unencrypted over the network. Since the VNF is responsible for its security, it is up to the VMs to establish secure data plane, e.g., using IPsec over its tenant network.

A Cloud actor must not be able to retrieve secrets used by VNF managers. All communications between the VNFM or orchestrator, and the infrastructure must be protected in integrity and confidentiality (e.g., by using TLS) and controlled via appropriate IP filtering rules (sec.lcm.006).

The Cloud Infrastructure must onboard only trusted and verified VM images, implying that VNF vendors provide signed images (sec.img.001); images from non-trusted sources may contain security breaches or unsolicited malicious code (spoofing, information disclosure). It is recommended to scan all VM images with a vulnerability scanner(sec.img.002). The scan is mandatory for images from unknown or untrusted sources.

To mitigate tampering attacks, it is recommended to use the Glance image signing feature [117] to validate an image when uploading. In this case, Barbican service must be installed.

In order to protect data, VNFs must encrypt the volumes they use. In this case, the encryption key must not be stored on the infrastructure. When a key management service is provided by the infrastructure, OpenStack can encrypt data on behalf of tenants (sec.gen.010). It is recommended to rely on Barbican, as the key manager service of OpenStack.

6.2.4 Workload Security

OpenStack segregates its infrastructure (sec.ci.008) (for example, hosts) by Regions, Host Aggregates and Availability Zones (AZ). Workloads can also be segregated by server groups (affinity and non-affinity groups) (sec.sys.008). These options support the workloads placement requirement (sec.wl.001, sec.wl.004).

Separation of non-production and production workloads, or by workload category (for example, payment card information, healthcare, etc.) requires separation through server groups (for example, Regions, AZs), but also requires network and storage segregation as in Regions. Thus, the separation of these workloads is handled through placement of workloads in separate AZs and/or Regions (sec.wl.005 and sec.wl.006).

Regions also support the sec.wl.004 requirement for separation by Location (for example, country).

Operational security is handled through a combination of mechanisms including the above and security groups (sec.sys.002). Security groups limit the types of traffic that have access to instances. One or more security groups can be automatically assigned to an instance at launch. The rules associated with a security group control the incoming traffic. Any incoming traffic not matched by a rule is denied access. The security group rules govern access through the setting of different parameters: traffic source, protocols and destination port on a VM. Errors in provisioning/managing OpenStack Security Groups can lead to non-functioning applications, and it can take a long time to identify faults and correct them. Thus, the use of tools for auto provisioning and continued inspection of security groups and network policies is required.

Given the rate of change in the workload development and deployment, and the cloud environment itself, sec.wl.003 requires that the workloads must be assessed during the CI/CD process as the images are created and then whenever they are deployed. In addition, the infrastructure must be configured for security as discussed elsewhere in this chapter including secure boot.

6.2.5 SR-IOV and DPDK Considerations

The SR-IOV agent only works with NoopFirewallDriver when Security Groups are enabled, but can still use other firewall_driver for other Agents by updating their conf with the requested firewall driver. Please see SR-IOV Passthrough for Networking [118].

Operators typically do not implement Security Groups when using SR-IOV or DPDK networking technologies.

6.2.6 Image Security

Images from untrusted sources must not be used (sec.img.001). Valuable guidance on trusted image creation process and image signature verification is provided in the “Trusted Images” section of the OpenStack Security Guide [119]. The OpenStack Security Guide includes reference to the “*OpenStack Virtual Machine Image Guide* :cite:p: `openstackim`” that describes how to obtain, create, and modify OpenStack compatible virtual machine images.

Images to be ingested, including signed images from trusted sources, need to be verified prior to ingestion into the Image Service (Glance) (sec.gen.009). The operator will need toolsets for scanning images, including for virus and malware detection (sec.img.002). Adding Signed Images to the Image Service (Glance) is specified in OpenStack Operations Guide [120]. Image signing and verification protects image integrity and authenticity by enabling deployers to sign images and save the signatures and public key certificates as image properties. The creation of signature per individual artifact in the VNF package is required by ETSI NFV SOL004 [121].

The chain of trust requires that all images are verified again in the Compute service (Nova) prior to use. Integrity verification at the time of instantiation is required by ETSI NFV SEC021 [122].

Images must be also updated to benefit from the latest security patches (sec.gen.008, sec.img.007).

6.2.7 Security LCM

Cloud Infrastructure LCM encompasses provisioning, deployment, configuration and management (resources scaling, services upgrades, etc.) as described in *Operations and Life Cycle Management*. These operations must be securely performed in order to keep the infrastructure safe and operational (sec.lcm.003).

Provisioning/Deployment

Regarding the provisioning of servers, switches, routers and networking, tools must be used to automate the provisioning eliminating human error. For Infrastructure hardware resources, a set of recommendations is detailed in *Underlying resources provisioning* to automate and secure their provisioning (sec.lcm.001).

For OpenStack services and software components, deployment tools or components must be used to automate the deployment and avoid errors. The deployment tool is a sensitive component storing critical information (deployment scripts, credentials, etc.). The following rules must be applied:

- The boot of the server or the VM hosting the deployment tool must be protected
- Integrity of the deployment images must be checked, before starting deployment
- Deployment must be done through dedicated network (e.g. VLAN)
- When the deployment is finished, the deployment tool must be turned-off, if the tool is only dedicated to deployment. Otherwise, any access to the deployment tool must be restricted.
- Strict access permissions must be set on OpenStack configuration files.

Configuration and management

Configuration operations must be tracked (sec.gen.015, sec.mon.006, sec.mon.007). Events such as system access attempts, actions with high privileges, modification of configuration, must be logged and exported on the fly to a non-local storage. The communication channel used for log collection must be protected for integrity and confidentiality, and the logs protected against unauthorised modification (sec.mon.004).

Per sec.sys.0016 and sec.lcm.002 requirements, management protocols limiting security risks must be used such as SNMPv3, SSH v2, ICMP, NTP, syslog and TLS. How to secure logging is described in the following section.

Platform backup

The storage for backup must be independent of storage offered to tenants.

Security upgrades

To defend against virus or other attacks, security patches must be installed for firmware, OS, Hypervisor and OpenStack services according to their criticality.

6.2.8 Monitoring and Security Audit

The intent of this section is to provide a key baseline and minimum requirements to implement logging that can meet the basic monitoring and security auditing needs. This should provide sufficient preliminary guidance, but is not intended to provide a comprehensive solution. Regular review of security logs that record user access, as well as session (sec.mon.010) and network activity (sec.mon.012), is critical in preventing and detecting intrusions that could disrupt business operations. This monitoring process also allows administrators to retrace an intruder's activity and may help correct any damage caused by the intrusion (sec.mon.011).

The logs have to be continuously monitored and analysed with alerts created for anomalies (sec.lcm.005). The resources for logging, monitoring and alerting also need to be logged and monitored, and corrective actions taken so that they are never short of the needed resources (sec.mon.015).

Creating Logs

- All resources to which access is controlled, including but not limited to applications and operating systems, must have the capability of generating security audit logs (sec.mon.001).
- Logs must be generated for all components (e.g., Nova in OpenStack) that form the Cloud Infrastructure (sec.mon.001).
- All security logging mechanisms must be active from system initialisation (sec.mon.018):
 - These mechanisms include any automatic routines necessary to maintain the activity records and clean-up programs to ensure the integrity of the security audit/logging systems.
- Logs must be time synchronised (sec.mon.002).

What to Log / What NOT to Log

What to log

Where technically feasible the following system events must be recorded (sec.mon.005):

- Successful and unsuccessful login attempts including:
 - Command line authentication (i.e., when initially getting token from keystone)
 - Horizon authentication
 - SSH authentication and sudo on the computes, controllers, network and storage nodes
- Logoffs
- Successful and unsuccessful changes to a privilege level (sec.lcm.012)
- Successful and unsuccessful configuration changes
- Successful and unsuccessful security policy changes
- Starting and stopping of security logging
- Creating, removing, or changing the inherent privilege level of users (sec.lcm.012)
- Connections to a network listener of the resource
- Starting and stopping of processes including attempts to start unauthorised processes
- All command line activity performed by the following innate OS programs known to otherwise leave no evidence upon command completion including PowerShell on Windows systems (e.g., Servers, Desktops, and Laptops)
- Where technically feasible, any other security events should be recorded

What NOT to log

Security audit logs must NOT contain:

- Authentication credentials, even if encrypted (e.g., password) (sec.mon.019);
- Keystone Token;
- Proprietary or Sensitive Personal Information.

Where to Log

- The logs must be stored in an external system (sec.mon.018), in a manner where the event can be linked to the resource on which it occurred.
- Where technically feasible, events must be recorded on the device (e.g. VM, physical node, etc.) where the event occurs, if the external logging system is not available (sec.mon.021).
- Security audit logs must be protected in transit and at rest (sec.mon.004).

Required Fields

The security audit log must contain at minimum the following fields (sec.mon.001) where applicable and technically feasible:

- Event type
- Date/time
- Protocol
- Service or program used for access
- Success/failure
- Login ID — Where the Login ID is defined on the system/application/authentication server; otherwise, the field should contain 'unknown', in order to protect authentication credentials accidentally entered at the Login ID prompt from appearing in the security audit log.
- Source and destination IP Addresses and ports

Data Retention

- Log files must be retained for 180 days, or the relevant regulator mandate, or your customer mandate, whichever is higher (sec.mon.020).
- Implementation and monitoring: after 180 days or your mandated retention period, security audit logs must be destroyed.

Security Logs Time Synchronisation

The host and various system clocks must be synchronised with an authenticated time service/NTP server (sec.gen.007).

In any time synchronisation, we need to specify the synchronisation interval and the tolerance where the latter specifies the permissible difference the local time can be out of synchronisation. Whenever the time synchronisation forces the local time to change or the use of another NTP server, the change details must be logged including time server source, time, date and time zones (sec.mon.003).

7 Operations and Life Cycle Management

To create an Infrastructure as a Service (IaaS) cloud requires the provisioning and deployment of the underlying infrastructure (compute, networking and storage) and deployment, configuration and management of the necessary software on the infrastructure; in the process of deploying the software, configuration of the infrastructure may also need to be performed.

Instead of deploying the infrastructure components and services manually, the current best practice is to write *code* (Infrastructure as Code, IaC) to define, provision, deploy, configure and manage the IaaS cloud infrastructure and services. IaC tools allows the entire provisioning, configuration and management processes to be automated. The desired state of the infrastructure and services is represented in a set of human readable, machine executable, and version-controlled files. With version control, it is easy to roll back to an older version and have access to the history of all committed changes.

The provisioning of the infrastructure is typically performed by provisioning tools while the deployment of the software and the configuration of the software, and where needed the infrastructure, falls in the domain of configuration management tools. A single tool may support both provisioning and configuration management.

Operators may choose certain paradigms with respect to how they provision and configure their IaaS cloud. These paradigms will drive the selection of the provisioning and configuration tools. In this chapter we will discuss the capabilities of provisioning and configuration management systems; some open-source tools may be mentioned but their capabilities are beyond the scope of this chapter.

7.1 Procedural versus Declarative code

The procedural style IaC tools require code that specifies how to achieve the desired state. Whilst the declarative style IaC tools require code that specifies the desired state (what not how). The major difference between the two styles emerges when changes to the desired state are required. In the procedural style, the change is coded in terms of the difference between the desired and current states while in the declarative style the new desired state is specified. In the procedural style since the state difference has to be coded, a new code file has to be created for each change; in the declarative style the existing code file is updated with the new state information. In the declarative style knowledge of the current state is not required. In the procedural style, knowledge of the current state has to be manually figured by tracing the created code files and the order in which they were applied.

7.2 Mutable versus Immutable infrastructure

In the mutable infrastructure paradigm, software updates are made in place. Over time this can lead to configuration drift where each server becomes slightly different from all other servers. In the immutable infrastructure paradigm, new servers are deployed with the new software version and then the old servers are undeployed.

7.3 Cloud Infrastructure provisioning and configuration management

In the Reference Model [1], the “Configuration and Lifecycle Management” chapter defines the functions of Configuration and Life Cycle Management (LCM). To operate and manage a scalable cloud, that minimises operational costs, requires tools that incorporates systems for automated provisioning and deployment, and managing configurations that ensures the correctness and integrity of the deployed and configured systems.

7.3.1 Underlying resources provisioning

This section deals with automated provisioning of the Cloud Infrastructure; for example, provisioning the servers, switches, routers, networking (e.g., subnets, routing tables, load balancers, etc.), databases and all required operating systems (Servers, switches, etc.).

The following are the minimum tasks that need to be performed by automation:

- **Pre-boot configuration** such as BIOS/RAID/IPMI settings: Hardware manufacturers typically have their proprietary interface for these tasks but standards such as Redfish are being increasingly utilised. Consider using tooling to ensure consistency across all infrastructure components.
- **Bootloader installation** of base Network Operating System (NOS) on networking equipment or the Operating System (OS) should be performed using PXE; again consider tooling to ensure consistency across all infrastructure components.

To ensure operational efficiency and save cost and time, the lifecycle management for physical and virtual servers must be automated using tools which will handle the repetitive tasks like provisioning, configuration, and monitoring. Foreman [123] is commonly used to automate the provisioning and management of bare metal infrastructure. Foreman is an open-source project, base of several commercial products. Foreman provides the full management of PXE configuration and the installation for many Operating Systems (CentOS, Fedora, Ubuntu, Debian, Red Hat Enterprise Linux, OpenSUSE, etc.). Foreman service can be installed by Ansible playbooks [124]. Ansible playbooks are basic tools for the automation of the infrastructure virtualisation layer deployments.

7.3.2 VIM deployment

When the underlying resources are installed and configured, the VIM software is deployed. An automated deployment is highly recommended for the same reasons of efficiency. Open-source installers are available to perform the deployments of the OpenStack services. A subset of these tools is described below.

- OpenStack TripleO [81], “OpenStack on OpenStack”

TripleO is an official OpenStack project which allows to deploy and manage a production cloud onto bare metal hardware using a subset of existing OpenStack components. The first step of deployment is the creation of an “undercloud” or deployment cloud. The undercloud contains the necessary OpenStack components to deploy and manage an “overcloud”, representing the deployed cloud. The architecture document [125] describes the solution. Nova and Ironic are used in the undercloud to manage the servers in bare metal environment. TripleO leverages on Heat templates.

- Airship v2 [79]

Airship is supported by the OpenStack Foundation. It is a collection of interoperable open-source components allowing to declaratively automate cloud provisioning. The configurations are defined by YAML documents. All services are running on containers. Airship v2 is aligned with maturing CNCF projects such as Kubernetes, Kubectl, Kubeadmin, Argo, Cluster API, Kustomize, and Metal3. Airship v2.1, released in November 2021, leverages on Kubernetes 1.21. It includes cloud provisioning at edge and for 3rd party cloud. The use of the OpenStack-Helm project allows the deployment of OpenStack on top of Kubernetes. Airship is not only a provisioning tool, but also a configuration management system.

- StarlingX [80]

StarlingX is dedicated to cloud infrastructure deployment at the edge, taking into account the specific edge use cases requirements for low latency and precision clock synchronisation. It aims to install a containerised version of OpenStack services, leveraging on Kubernetes, Docker registry, Airship Armada, and Helm.

OpenStack-Helm is used as a starting point. OpenStack is installed and managed as an Armada application. Armada Applications are a set of one or more interdependent Application Helm charts. In the case of StarlingX, there is generally a Helm chart for every OpenStack service.

7.3.3 Configuration Management

The configuration management system ensures the correctness and integrity of the deployed and configured systems. The tools provide the assurance that the expected software is running with the expected configurations on correctly configured nodes that continue to be configured correctly.

Configuration Management is composed of the following activities:

- **Desired (Target) State:** a version of the software and hardware and their configurations. Depending upon the configuration management system these configurations are specified in cookbooks, playbooks, manifests, etc. The configuration specifications in these artefacts is used to configure the different types of nodes, BIOS, operating systems, hypervisor and OpenStack services (through settings within their config files such as nova.conf, etc.).
- **Current State:** the current configuration of software and hardware as provided by monitoring systems
- **State variance mitigation:** The CM system, on discovering a variance between the desired and current states, acts to drive the state to the desired state. Each CM system accomplishes the task in different ways.

7.4 Cloud Infrastructure and VIM Maintenance

Cloud Infrastructure and VIM Maintenance activities can be classified as

1. Deployment of additional infrastructure components (or removal of infrastructure components)
2. Cloud Infrastructure Configuration changes
3. VIM Configuration changes
4. Version changes (upgrade) of Cloud Infrastructure software (for example, Host Operating System, Hypervisor, etc.)
5. Version changes of VIM Software (or component services)

Deployment (or removal) of infrastructure components

In declarative tools, the code with the specified desired state (for example, number of compute servers) is modified to the new desired state. The IaC tool then ensures that the desired state is achieved. In procedural tools, the step-by-step code to deploy (remove) infrastructure components needs to be specified. Existing code can be cloned, and appropriate changes made to get to the desired state.

Configuration and Version Changes

Configuration and Version Changes are made in a similar fashion to the “Deployment of infrastructure components” except that the IaC tools used maybe different.

7.5 Logging, Monitoring and Analytics

- Logging
- Monitoring
- Alerting
- Logging, Monitoring, and Analytics (LMA) Framework

7.5.1 Logging

A log, in the context of computing, is the automatically produced and time-stamped documentation of events relevant to a particular system. All software, including operating systems, middleware and applications produce log files. Enterprises and vendors may have custom monitoring and logging solutions. The intent of the logging and monitoring is to capture events and data of interest to the Cloud Infrastructure and workloads so that appropriate actions can be taken. For example,

- Operating systems and web servers maintain an access log of all access requests, session details and file access.
- Databases maintain a transaction log of all transaction executed including an added, changed and deleted data.
- Audit logs record chronological documentation of any activities that could have affected a particular operation or event. Data typically includes resources accessed, destination and source addresses, and a timestamp and login information for the person who accessed the resources.

Some of the data is to support the metrics collection specified in the Reference Model [1].

Logs have multiple operational uses including for:

1. Regulatory Compliance and Security Information and Event Management (SIEM) featuring the automated gathering, analysis and correlation of log data across all systems and devices across an operator to provide real-time analysis, event prioritisation, reporting, notification and alerting.
2. Monitoring across systems in real-time to detect particular log events, patterns, anomalies or inactivity to gauge system and application health
3. Identify system and application performance and configuration issues
4. Root cause analysis for system and application failures and errors
5. Ensuring that operational objectives and SLAs are met

7.5.2 Monitoring

Monitoring is the process of collecting, aggregating, and analysing values that improve awareness of the components' characteristics and behavior. The data from various parts of the environment are collected into a monitoring system that is responsible for storage, aggregation, visualisation, and initiating automated responses when the values meet specific threshold.

Monitoring systems fulfill many related functions. Their first responsibility is to accept and store incoming and historical data. While values representing the current point in time are useful, it is almost always more helpful to view those numbers in relation to past values to provide context around changes and trends.

7.5.3 Alerting

Alerting is the responsive component of a monitoring system that performs actions based on changes in metric values. Alert definitions are composed of two components: a metrics-based condition or threshold, and an action to perform when the values fall outside of the acceptable conditions.

While monitoring systems are incredibly useful for active interpretation and investigation, one of the primary benefits of a complete monitoring system is letting administrators disengage from the system. Alerts allow the specification of situations that make sense to actively manage, while relying on the passive monitoring of the software to watch for changing conditions.

7.5.4 Logging, Monitoring, and Analytics (LMA) Framework

In this section, a possible framework utilising Prometheus, Fluentd, Elasticsearch and Kibana is given as an example only.



Figure 7.1: Monitoring and Logging Framework

The monitoring and logging framework (figure above) leverages Prometheus as the monitoring engine and Fluentd for logging. In addition, the framework uses Elasticsearch to store and organise logs for easy access. Prometheus agents pull information from individual components on every host. Fluentd, an Open Source data collector, unifies data collection and consumption for better use and understanding of data. Fluentd captures the access, application and system logs.

8 Conformance

The objective of this chapter is to provide an automated mechanism to validate OpenStack based cloud infrastructure against the standard set of requirements defined in *Architecture Requirements*. Through this validation mechanism, a provider of cloud infrastructure will be able to test their cloud infrastructure's conformance to this reference architecture. This will ease the integration of network functions into operator environments that host compatible cloud infrastructures, thereby reducing cost, complexity, and time of integration.

The overall workstream requires the close coordination of the following:

- **Requirements** - The agreed upon capabilities and conditions that a compliant cloud infrastructure must provide or satisfy.
- **Tests** - The verification mechanism that determines that a given cloud infrastructure complies with one or more requirements.
- **Conformance Specifications** - The definition of the requirements, tests, and circumstances (test case integration, etc.) that must be met to be deemed conformant.

8.1 Requirements and Testing Principles

If there is no clear traceability and strong links between **Requirements**, **Tests**, and **Conformance Specifications**, then it becomes difficult to determine if a cloud infrastructure is compliant. With this in mind, below are the set of recommended principles for each of the three components to follow. Adherence to these principles will provide the following:

- Enable clear progress tracking and linkage between independent projects (i.e. know what has and has not been covered, and track changes over time)
- Help users better understand if they meet requirements
- Provide a stable set of point-in-time requirements and tests to achieve conformance
- Reduce ambiguity in testing, requirements, and conformance

Testing Principles:

- There must be traceability between test cases and requirement being validated
- Failures should provide additional content to inform the user where or how the requirement was violated (e.g. which file or resource violated the requirement). Put another way, do not require the user to read the test to understand what went wrong
- Testing tools should support selection of tests based on category or profile.
- Tests must be available to run locally by both VNF and cloud infrastructure providers
- Testing tools must produce machine-readable result formats that can be used as input into any badging program

Conformance Specifications:

- Conformance specifications must refer to or define the *versioned* requirements that must be satisfied
- Conformance specifications must refer to the *versioned* test implementations that must be used to validate the requirements
- Conformance specifications must define the expected preconditions and environment requirements for any test tooling
- Conformance specifications must define which tests must be executed in the given testing tools to achieve conformance
- The conformance specifications must provide the mapping between tests and requirements to demonstrate traceability and coverage.

8.2 Test Case Integration and Tooling

The OpenStack based cloud infrastructure suite must utilise the Anuket test case integration toolchain to deliver overall integration, the same end user actions, and a unique test result format (e.g. Anuket test result database) needed by the end users and any test case result verification program (e.g. OVP [126]).

8.2.1 Anuket Toolchains

Anuket, previously named OPNFV, has built a complete CI/CD toolchain for continuously deploying and testing cloud infrastructure.

As for all installer projects, Jenkins [127] triggers scenario deployments, runs the Anuket gating test cases and then publishes all test results in the centralised test database [128] and all artifacts (reports, logs, etc.) to an S3 compatible storage service [129].

The verification, validation, and conformance processes leverage existing Anuket testing knowledge (projects) and experience (history) by utilising the toolchain design already in-place. The conformance toolchain only requires for the local deployment of the components instead of leveraging the common Anuket centralised services. However, the interfaces remain unchanged for leveraging test jobs, the common test case execution, the test result database and the S3 protocol to publish the artifacts. It's worth mentioning that dumping all results and logs required for conformance is already in place in Functest daily jobs (see functest-wallaby-zip [130]).

It should be noted that Xtesting CI [131] supports both centralised and distributed deployment models as described before. It has deployed the full toolchain in one small virtual machine to verify ONAP Openlab via Functest.

8.2.2 Test Case Integration

To reach all goals in terms of verification, validation, compliance, and conformance, all test cases must be delivered as Docker containers [132] to simplify the CI toolchain setup including:

- the common test case execution
- the unified way to manage all the interactions with the CI/CD components and with third-parties (e.g. dump all test case logs and results for conformance)

For their part, the Docker containers simply enforce that the test cases are delivered with all runtime dependencies. This prevents lots of manual operations when configuring the servers running the test cases and prevents conflicts between the test cases due to any dependencies.

It's worth mentioning that all the conformance test cases already leverage Xtesting [133] which is a simple framework to assemble sparse test cases and to accelerate the adoption of CI/CD best practices. By managing all the interactions with the CI/CD components (test scheduler, test results database, artifact repository), it allows the developer to work only on the test suites without diving into CI/CD integration. Even more, it brings the capability to run heterogeneous test cases in the same CI toolchains thanks to a few, quickly achievable constraints [134].

The Docker containers proposed by the test projects must also embed the Xtesting Python package [135] and the related test case execution description files [136] as required by Xtesting.

8.2.3 Testing Cookbooks

Xtesting CI [131] leverages the common test case execution proposed by Xtesting. Thanks to a simple test case list, this tool deploys plug-and-play CI/CD toolchains in a few commands [137]. In addition, it supports multiple components such as Jenkins and Gitlab CI (test schedulers) and multiple deployment models [138] such as all-in-one or centralised services.

Xtesting [133] and Xtesting CI [131] combined meet the requirements about verification, validation, compliance, and conformance:

- smoothly assemble multiple heterogeneous test cases
- generate the Jenkins jobs in Anuket Releng [139] to verify conformance with Anuket specifications
- deploy local CI/CD toolchains everywhere to check conformance with Anuket specifications
- dump all test case results and logs [140] for third-party conformance review

Here are a couple of publicly available playbooks :

- Xtesting samples [141]
- OpenStack verification [142]
- Anuket RC1 [143]
- Kubernetes verification [144] (it can be used prior to the deployment of OpenStack services when OpenStack services are containerised)

8.3 Conformance Test Suite

8.3.1 Functest in a nutshell

Functest [145] was initially created to verify OPNFV Installers and Scenarios and then to publish fair, trustable, and public results regarding the status of the different open-source technologies, especially for Neutron backends (e.g., Neutron agents, OpenDaylight, OVN, etc.). It has been continuously updated to offer the best testing coverage for any kind of OpenStack and Kubernetes deployments including production environments. It also ensures that the platforms meet Network Functions Virtualisation requirements by running and testing VNFs amongst all tests available.

Functest is driven by a true verification of the platform under test as opposed to the interoperability programs such as RefStack [146] or OPNFV Verification Program [126] which select a small subset of Functional tests passing in many different open-source software combinations:

- tests are skipped if an optional support is missing (e.g. Barbican [68] or networking features such as BGPVPN interconnection [147] or Service Function Chaining [148])
- tests are parameterised (e.g. shared vs non-shared live migration)
- blacklist mechanisms are available if needed

It should be noted that the RefStack lists [146] are included as they are in Functest in the next 3 dedicated testcases:

- refstack_compute (OpenStack Powered Compute)
- refstack_object (OpenStack Powered Object Storage)
- refstack_platform (OpenStack Powered Platform)

Then Functest conforms with the upstream rules (versions, code quality, etc.) and especially their gates [149] (a.k.a. the automatic verification prior to any code review) to preserve the quality between code and deployment. In that case, Functest can be considered as a smooth and lightweight integration of tests developed upstream (and the Functest team directly contributes in these projects: Rally [150], Tempest [151], etc.). It is worth mentioning that, as opposed to the OpenStack Gates leveraging on DevStack [152], it can check the same already deployed SUT over and over even from a Raspberry PI [153]. Here the testcases can be executed in parallel vs the same deployment instead of being executed vs different pools of virtual machines.

Here are the functional tests (>2000) running in OpenStack gates integrated in Functest Smoke (see Functest daily jobs [154] for more details):

Table 8.1: Functional tests

Testcases	Gates
tempest_neutron	Neutron
tempest_cinder	Cinder
tempest_keystone	Keystone
rally_sanity	General
refstack_defcore	General
tempest_full	General
tempest_slow	General
tempest_scenario	General
patrole	Patrole
tempest_barbican	Barbican
networking-bgpvpn	Networking BGP VPN
networking-sfc	Networking SFC

To complete functional testing, Functest also integrates a few performance tools [155] (2-3 hours) as proposed by OpenStack:

Table 8.2: Performance tools

Testcases	Benchmarking
rally_full	Control Plane (API) testing
rally_jobs	Control Plane (API) testing
vmtp	Data Plane testing
shaker	Data Plane testing

And VNFs automatically deployed and tested :

Table 8.3: VNFs

Testcases	Benchmarking
cloudify	Cloudify deployment
cloudify_ims	Clearwater IMS deployed via Cloudify
heat_ims	Clearwater IMS deployed via Heat
vynos_vrouter	VyOS deployed via Cloudify
juju_epc	OAI deployed via Juju

Functest should be considered as a whole as it meets multiple objectives about the reference implementation:

- verify all APIs (services, advances, features, etc.) exposed by the reference implementation
- compare the reference implementation and local deployments from a functional standpoint and from OpenStack control plane and dataplane capabilities

Additional links:

- Homepage [145]
- Run Alpine Functest containers (Wallaby) [156]
- Deploy your own Functest CI/CD toolchains [137]
- Functest gates [157]

8.3.2 Test Case traceability

Interfaces & APIs

The OpenStack Gates [149] verify all changes proposed mostly by running thousands of Tempest tests completed by Rally scenarios in a few cases. Skipping tests is allowed in all OpenStack Gates and only failures rate the review -1 because of the multiple capabilities and backends selected in the different Gate jobs. The classical Functest containers [156] conform to this model which also fits the heterogeneous user deployments.

From an OpenStack based cloud infrastructure conformance state point, the capabilities are well described in *Interfaces and APIs* which allows tuning the test configurations and the test lists to avoid skipping any test. It results that all tests covering optional capabilities and all upstream skipped tests due to known bugs are not executed. All remaining tests must be executed and must pass successfully.

New Functest containers [158] have been proposed for Anuket Compliance which simply override the default test configurations and the default test lists. Any optional capability or services (e.g. Barbican) can be still verified by the classical Functest containers.

The next subsections detail the Tempest tests which must not be executed from a compliance state point. The remaining tests have to pass successfully. They cover all together the API testing requirements as asked by *Interfaces and APIs*

The following software versions are considered here to verify OpenStack Wallaby:

Table 8.4: Software versions

Software	Version
Functest	wallaby
Cinder Tempest plugin	1.4.0
Keystone Tempest plugin	0.7.0
Heat Tempest plugin	1.2.0
Neutron Tempest plugin	1.4.0
Rally OpenStack	2.2.1.dev11
Tempest	27.0.0

Identity - Keystone API testing

Keystone API is covered in the OpenStack Gates via Tempest [151] and keystone-tempest-plugin [159] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.5: Keystone API testing

Test rejection regular expressions	Reasons
.*api.identity.v3.test_oauth1_tokens	oauth1
.*scenario.test_federated_authentication	federation
.*identity.admin.v2	API v2
.*identity.v2	API v2
.*identity.v3.test_access_rules	access_rules
.*identity.v3.test_application_credentials.\nApplicationCredentialsV3Test.\ntest_create_application_credential_access_rules	access_rules

Keystone API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.keystone
- KeystoneBasic.add_and_remove_user_role
- KeystoneBasic.create_add_and_list_user_roles
- KeystoneBasic.create_and_list_tenants
- KeystoneBasic.create_and_delete_role
- KeystoneBasic.create_and_delete_service
- KeystoneBasic.get_entities
- KeystoneBasic.create_update_and_delete_tenant
- KeystoneBasic.create_user
- KeystoneBasic.create_tenant
- KeystoneBasic.create_and_list_users
- KeystoneBasic.create_tenant_with_users

Image - Glance API testing

Glance API is covered in the OpenStack Gates via [151] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.6: Glance API testing

Test rejection regular expressions	Reasons
.*image.v1	API v1
.*image.v2.admin.test_images.ImportCopyImagesTest	import_image
.*image.v2.test_images_negative.ImagesNegativeTest.\test_create_image_reserved_property	os_glance_reserved
.*image.v2.test_images_negative.ImagesNegativeTest.\test_update_image_reserved_property	os_glance_reserved
.*image.v2.test_images_negative.ImportImagesNegativeTest.\test_image_web_download_import_with_bad_url	web-downloadimport
.*image.v2.test_images.ImportImagesTest	import_image
.*image.v2.test_images.MultiStoresImportImages	import_image

Glance API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.validate_glance
- GlanceImages.create_and_delete_image
- GlanceImages.create_and_list_image
- GlanceImages.list_images
- GlanceImages.create_image_and_boot_instances

Block Storage - Cinder API testing

Cinder API is covered in the OpenStack Gates via Tempest [151] and cinder-tempest-plugin [161] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.7: Cinder API testing

Test rejection regular expressions	Reasons
.*test_incremental_backup	Functest review 68881 [162]
.*test_consistencygroups	consistency_group
.*test_backup_crossproject_admin_negative	Functest review 71011 [163]
.*test_backup_crossproject_user_negative	Functest review 71011 [163]
.*test_volume_encrypted.TestEncryptedCinderVolumes	attach_encrypted_volume
.*test_encrypted_volumes_extend	extend_attached_encrypted_volume
.*test_group_snapshots.GroupSnapshotsV319Test.\ntest_reset_group_snapshot_status	OpenStack bug 1770179 [164]
.*test_multi_backend	multi-backend
.*test_volume_retype.VolumeRetypeWithMigrationTest	multi-backend
.*test_volume_delete_cascade.VolumesDeleteCascade.\ntest_volume_from_snapshot_cascade_delete	OpenStack bug 1677525 [165]
.*test_volumes_backup.VolumesBackupsTest.\ntest_volume_backup_create_get_detailed_list_restore_delete	ceph
.*test_volumes_extend.VolumesExtendAttachedTest.\ntest_extend_attached_volume	extend_attached_volume
.*tempest.scenario.test_volume_migrate_attached	multi-backend

Cinder API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.validate_cinder
- CinderVolumes.create_and_delete_snapshot
- CinderVolumes.create_and_delete_volume
- CinderVolumes.create_and_extend_volume
- CinderVolumes.create_from_volume_and_delete_volume
- CinderQos.create_and_list_qos
- CinderQos.create_and_set_qos
- CinderVolumeTypes.create_and_list_volume_types
- CinderVolumeTypes.create_volume_type_and_encryption_type
- Quotas.cinder_update_and_delete
- Quotas.cinder_update

Object Storage - Swift API testing

Swift API is covered in the OpenStack Gates via Tempest [151] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.8: Swift API testing

Test rejection regular expressions	Reasons
.*test_container_sync.ContainerSyncTest.\ntest_container_synchronization	OpenStack bug 1317133 [166]
.*test_container_sync_middleware.ContainerSyncMiddlewareTest.\ntest_container_synchronization	container_sync
.*test_object_services.ObjectTest.\ntest_create_object_with_transfer_encoding	OpenStack bug 1905432 [167]

Swift API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- SwiftObjects.create_container_and_object_then_list_objects
- SwiftObjects.list_objects_in_containers
- SwiftObjects.create_container_and_object_then_download_object
- SwiftObjects.create_container_and_object_then_delete_all
- SwiftObjects.list_and_download_objects_in_containers

Networking - Neutron API testing

Neutron API is covered in the OpenStack Gates via Tempest [151] and neutron-tempest-plugin [168] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.9: Neutron API testing

Test rejection regular expressions	Reasons
.*admin.test_agent_availability_zone	DHCP agent and L3 agent
.*admin.test_dhcp_agent_scheduler	dhcp_agent_scheduler
.*admin.test_l3_agent_scheduler	l3_agent_scheduler
.*admin.test_logging	logging
.*admin.test_logging_negative	logging
.*admin.test_network_segment_range	network-segment-range
.*admin.test_ports.PortTestCasesAdmin.\ntest_regenerate_mac_address	port-mac-address-regenerate
.*admin.test_ports.PortTestCasesResourceRequest	port-resource-request
.*admin.test_routers_dvr	dvr
.*admin.test_routers_flavors	l3-flavors
.*admin.test_routers_ha	l3-ha
.*test_floating_ips.FloatingIPPoolTestJSON	floatingip-pools
.*test_floating_ips.FloatingIPTestJSON.\ntest_create_update_floatingip_port_details	fip-port-details
.*test_metering_extensions	metering

continues on next page

Table 8.9 – continued from previous page

Test rejection regular expressions	Reasons
.*test_metering_negative	metering
.*test_networks.NetworksSearchCriteriaTest.\ntest_list_validation_filters	filter-validation
.*test_networks.NetworksTestAdmin.\ntest_create_tenant_network_vxlan	vxlan
.*test_networks.NetworksTestJSON.\ntest_create_update_network_dns_domain	dns-integration
.*test_port_forwardings	floating-ip-port-forwarding
.*test_port_forwarding_negative	floating-ip-port-forwarding
.*test_ports.PortsTaggingOnCreation	tag-ports-during-bulk-creation
.*test_ports.PortsTestJSON.\ntest_create_port_with_propagate_uplink_status	uplink-status-propagation
.*test_ports.PortsTestJSON.\ntest_create_port_without_propagate_uplink_status	uplink-status-propagation
.*test_ports.PortsTestJSON.\ntest_create_update_port_with_dns_domain	dns-domain-ports
.*test_ports.PortsTestJSON.\ntest_create_update_port_with_dns_name	dns-integration
.*test_ports.PortsTestJSON.\ntest_create_update_port_with_no_dns_name	dns-integration
.*test_revisions.TestRevisions.\ntest_update_dns_domain_bumps_revision	dns-integration
.*test_revisions.TestRevisions.\ntest_update_router_extra_attributes_\nbumps_revision	l3-ha
.*test_router_interface_fip	router-interface-fip
.*test_routers.DvrRoutersTest	dvr
.*test_routers.HaRoutersTest	l3-ha
.*test_routers.RoutersIPv6Test.\ntest_extra_routes_atomic	extraroute-atomic
.*test_routers.RoutersTest.\ntest_extra_routes_atomic	extraroute-atomic
.*test_routers_negative.DvrRoutersNegativeTest	dvr
.*test_routers_negative.\nDvrRoutersNegativeTestExtended	dvr
.*test_routers_negative.HaRoutersNegativeTest	l3-ha
.*test_security_groups.RbacSharedSecurityGroupTest	rbac-security-groups
.*test_subnetpool_prefix_ops	subnetpool-prefix-ops
.*test_subnetpools.RbacSubnetPoolTest	rbac-subnetpool
.*test_subnetpools_negative.SubnetPoolsNegativeTestJSON.\ntest_tenant_create_subnetpool_associate_shared_address_scope	rbac-subnetpool
.*test_subnetpools.SubnetPoolsSearchCriteriaTest.\ntest_list_validation_filters	filter-validation
.*test_subnets.SubnetsSearchCriteriaTest.\ntest_list_validation_filters	filter-validation
.*test_timestamp.TestTimeStamp.\ntest_segment_with_timestamp	standard-attr-segment
.*test_trunk.TrunkTestInheritJSONBase.\ntest_add_subport	OpenStack bug 1863707 [169]
.*test_trunk.TrunkTestMtusJSON	vxlan

continues on next page

Table 8.9 – continued from previous page

Test rejection regular expressions	Reasons
.*test_trunk_negative.TrunkTestJSON.\test_create_subport_invalid_inherit_network_\segmentation_type	vxlan
.*test_trunk_negative.TrunkTestMtusJSON	vxlan
.*test_qos.QosMinimumBandwidthRuleTestJSON	Functest review 69105 [170]
.*network.test_tags	tag-ext
.*test_routers.RoutersIpV6Test.\test_create_router_set_gateway_with_fixed_ip	OpenStack bug 1863707 [171]
.*test_routers.RoutersTest.\test_create_router_set_gateway_with_fixed_ip	OpenStack bug 1863707 [171]
.*test_network_basic_ops.\TestNetworkBasicOps.test_router_rescheduling	l3_agent_scheduler
.*test_network_advanced_server_ops.\TestNetworkAdvancedServerOps.\test_server_connectivity_cold_migration_revert	OpenStack bug 1836595 [172]

Neutron API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.validate_neutron
- NeutronNetworks.create_and_delete_networks
- NeutronNetworks.create_and_delete_ports
- NeutronNetworks.create_and_delete_routers
- NeutronNetworks.create_and_delete_subnets
- NeutronNetworks.create_and_list_networks
- NeutronNetworks.create_and_list_ports
- NeutronNetworks.create_and_list_routers
- NeutronNetworks.create_and_list_subnets
- NeutronSecurityGroup.create_and_delete_security_groups
- NeutronSecurityGroup.create_and_delete_security_group_rule
- NeutronNetworks.set_and_clear_router_gateway
- Quotas.neutron_update

Compute - Nova API testing

Nova API is covered in the OpenStack Gates via Tempest [151] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.10: Nova API testing

Test rejection regular expressions	Reasons
.*admin.test_agents	xenapi_apis
.*test_fixed_ips	neutron

continues on next page

Table 8.10 – continued from previous page

Test rejection regular expressions	Reasons
.*test_fixed_ips_negative	neutron
.*test_auto_allocate_network	shared networks
.*test_flavors_microversions.FlavorsV255TestJSON	max_microversion: 2.53
.*test_flavors_microversions.FlavorsV261TestJSON	max_microversion: 2.53
.*test_floating_ips_bulk	nova-network
.*test_live_migration.\nLiveAutoBlockMigrationV225Test.test_iscsi_volume	block live migration
.*test_live_migration.\nLiveAutoBlockMigrationV225Test.\ntest_live_block_migration	block live migration
.*test_live_migration.\nLiveAutoBlockMigrationV225Test.\ntest_live_block_migration_paused	block live migration
.*test_live_migration.\nLiveAutoBlockMigrationV225Test.\ntest_volume_backed_live_migration	volume-backed live migration
.*test_live_migration.LiveMigrationTest.\ntest_iscsi_volume	block live migration
.*test_live_migration.LiveMigrationTest.\ntest_live_block_migration	block live migration
.*test_live_migration.LiveMigrationTest.\ntest_live_block_migration_paused	block live migration
.*test_live_migration.LiveMigrationTest.\ntest_volume_backed_live_migration	volume-backed live migration
.*test_live_migration.\nLiveMigrationRemoteConsolesV26Test	serial_console
.*test_quotas.QuotasAdminTestV257	max_microversion: 2.53
.*test_servers.ServersAdminTestJSON.\ntest_reset_network_inject_network_info	xenapi_apis
.*certificates.test_certificates	cert
.*test_quotas_negative.\nQuotasSecurityGroupAdminNegativeTest	OpenStack bug 1186354 [173]
.*test_novnc	vnc_console
.*test_server_personality	personality
.*test_servers.ServerShowV263Test.\ntest_show_update_rebuild_list_server	certified_image_ref
.*test_servers_microversions.ServerShowV254Test	max_microversion: 2.53
.*test_servers_microversions.ServerShowV257Test	max_microversion: 2.53
.*test_servers_negative.ServersNegativeTestJSON.\ntest_personality_file_contents_not_encoded	personality
.*test_server_actions.ServerActionsTestJSON.\ntest_change_server_password	change_password
.*test_server_actions.ServerActionsTestJSON.\ntest_get_vnc_console	vnc_console
.*test_server_actions.ServerActionsTestJSON.\ntest_reboot_server_soft	OpenStack bug 1014647 [174]
.*test_server_rescue.\nServerBootFromVolumeStableRescueTest	stable_rescue
.*test_server_rescue.ServerStableDeviceRescueTest	stable_rescue
.*test_security_group_default_rules	OpenStack bug 1311500 [175]

continues on next page

Table 8.10 – continued from previous page

Test rejection regular expressions	Reasons
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_security_group_create_with_duplicate_name	neutron
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_security_group_create_with_\ninvalid_group_description	OpenStack bug 1161411 [176]
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_security_group_create_with_invalid_group_name	OpenStack bug 1161411 [176]
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_update_security_group_with_invalid_sg_description	neutron
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_update_security_group_with_invalid_sg_description	neutron
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_update_security_group_with_invalid_sg_id	neutron
.*test_security_groups_negative.\nSecurityGroupsNegativeTestJSON.\ntest_update_security_group_with_invalid_sg_name	neutron
.*test_server_metadata.ServerMetadataTestJSON	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_delete_metadata_non_existent_server	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_metadata_items_limit	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_set_metadata_invalid_key	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_set_metadata_non_existent_server	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_set_server_metadata_blank_key	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_set_server_metadata_missing_metadata	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_update_metadata_non_existent_server	xenapi_apis
.*test_server_metadata_negative.\nServerMetadataNegativeTestJSON.\ntest_update_metadata_with_blank_key	xenapi_apis

continues on next page

Table 8.10 – continued from previous page

Test rejection regular expressions	Reasons
.*test_list_server_filters.\nListServerFiltersTestJSON.\ntest_list_servers_filtered_by_ip_regex	OpenStack bug 1540645 [177]
.*servers.test_virtual_interfaces	nova-network
.*compute.test_virtual_interfaces_negative	nova-network
.*compute.test_networks	nova-network
.*test_attach_volume.AttachVolumeMultiAttach	volume_multiattach
.*test_volume_boot_pattern.\nTestVolumeBootPattern.\ntest_boot_server_from_encrypted_volume_luks	attach_encrypted_volume
.*test_volume_swap	swap_volume
.*test_encrypted_cinder_volumes	attach_encrypted_volume
.*test_minbw_allocation_placement	microversion
.*test_volumes_negative.\nUpdateMultiattachVolumeNegativeTest.\ntest_multiattach_rw_volume_update_failure	volume_multiattach
.*test_shelve_instance.TestShelveInstance.\ntest_cold_migrate_unshelved_instance	shelve_migrate

Nova API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.validate_nova
- NovaServers.boot_and_live_migrate_server
- NovaServers.boot_server_attach_created_volume_and_live_migrate
- NovaServers.boot_server_from_volume_and_live_migrate
- NovaKeypair.boot_and_delete_server_with_keypair
- NovaServers.boot_server_from_volume_and_delete
- NovaServers.pause_and_unpause_server
- NovaServers.boot_and_migrate_server
- NovaServers.boot_server_and_list_interfaces
- NovaServers.boot_server_associate_and_dissociate_floating_ip
- NovaServerGroups.create_and_delete_server_group
- Quotas.nova_update

Orchestration - Heat API testing

Heat API is covered in the OpenStack Gates via heat-tempest-plugin [178] as integrated in Functest Smoke CNTT [160].

According to *Interfaces and APIs* the following test names must not be executed:

Table 8.11: Heat API testing

Test rejection regular expressions	Reasons
.*functional.test_lbaasv2	lbaasv2
.*functional.test_encryption_vol_type	OpenStack story [179]
.*RemoteStackTest.\ntest_stack_create_with_cloud_credential	Functest review 69926 [180]
.*scenario.test_aodh_alarm	aodh
.*tests.scenario.test_autoscaling_lb	lbaas
.*scenario.test_autoscaling_lbv2	lbaasv2
.*scenario.test_server_software_config	Functest review 69926 [180]
.*test_volumes.\nVolumeBackupRestoreIntegrationTest	Functest review 69926 [181]
.*scenario.test_octavia_lbaas	octavia
.*scenario.test_server_cfn_init	Functest review 70004 [182]

Heat API is also covered by Rally [150].

Here are the mainline tasks integrated in Functest Smoke CNTT [160]:

- Authenticate.validate_heat
- HeatStacks.create_update_delete_stack
- HeatStacks.create_check_delete_stack
- HeatStacks.create_suspend_resume_delete_stack
- HeatStacks.list_stacks_and_resources

Dashboard

Horizon is covered in the OpenStack Gates via tempest-horizon [183] as integrated in Functest Healthcheck [184].

OpenStack API benchmarking

Rally [150] is tool and framework that allows to perform OpenStack API benchmarking.

Here are the Rally-based test cases proposed by Functest Benchmarking CNTT [185]:

- rally_full [186]: Functest scenarios iterating 10 times the mainline Rally scenarios
- rally_jobs [187]: Neutron scenarios executed in the OpenStack gates

The default SLA proposed in Functest Benchmarking CNTT [185] is a maximum failure rate of 0%.

Identity - Keystone API benchmarking

Functest rally_full_cntt [186]:

Table 8.12: Keystone API benchmarking

Scenarios	Iterations
Authenticate.keystone	10
KeystoneBasic.add_and_remove_user_role	10
KeystoneBasic.create_add_and_list_user_roles	10
KeystoneBasic.create_and_list_tenants	10
KeystoneBasic.create_and_delete_role	10
KeystoneBasic.create_and_delete_service	10
KeystoneBasic.get_entities	10
KeystoneBasic.create_update_and_delete_tenant	10
KeystoneBasic.create_user	10
KeystoneBasic.create_tenant	10
KeystoneBasic.create_and_list_users	10
KeystoneBasic.create_tenant_with_users	10

Image - Glance API benchmarking

Functest rally_full_cntt [186]:

Table 8.13: Glance API benchmarking

Scenarios	Iterations
Authenticate.validate_glance	10
GlanceImages.create_and_delete_image	10
GlanceImages.create_and_list_image	10
GlanceImages.list_images	10
GlanceImages.create_image_and_boot_instances	10
GlanceImages.create_and_deactivate_image	10
GlanceImages.create_and_download_image	10
GlanceImages.create_and_get_image	10
GlanceImages.create_and_update_image	10

Block Storage - Cinder API benchmarking

Functest rally_full_cntt [186]:

Table 8.14: Cinder API benchmarking

Scenarios	Iterations
Authenticate.validate_glance	10
CinderVolumes.create_and_attach_volume	10
CinderVolumes.create_and_list_snapshots	10
CinderVolumes.create_and_list_volume	10
CinderVolumes.create_and_upload_volume_to_image	10
CinderVolumes.create_nested_snapshots_and_attach_volume	10
CinderVolumes.create_snapshot_and_attach_volume	10
CinderVolumes.create_volume	10
CinderVolumes.list_volumes	10
CinderVolumes.create_and_delete_snapshot	10
CinderVolumes.create_and_delete_volume	10
CinderVolumes.create_and_extend_volume	10
CinderVolumes.create_from_volume_and_delete_volume	10
CinderQos.create_and_get_qos	10
CinderQos.create_and_list_qos	10
CinderQos.create_and_set_qos	10
CinderVolumeTypes.create_and_get_volume_type	10
CinderVolumeTypes.create_and_list_volume_types	10
CinderVolumeTypes.create_and_update_volume_type	10
CinderVolumeTypes.create_volume_type_and_encryption_type	10
CinderVolumeTypes.create_volume_type_add_and_list_type_access	10
Quotas.cinder_update_and_delete	10
Quotas.cinder_update	10

Object Storage - Swift API benchmarking

Functest rally_full_cntt [186]:

Table 8.15: Swift API benchmarking

Scenarios	Iterations
SwiftObjects.create_container_and_object_then_list_objects	10
SwiftObjects.list_objects_in_containers	10
SwiftObjects.create_container_and_object_then_download_object	10
SwiftObjects.create_container_and_object_then_delete_all	10
SwiftObjects.list_and_download_objects_in_containers	10

Networking - Neutron API benchmarking

Functest rally_full_cntt [186]:

Table 8.16: Neutron API benchmarking

Scenarios	Iterations
Authenticate.validate_neutron	10
NeutronNetworks.create_and_update_networks	10
NeutronNetworks.create_and_update_ports	10
NeutronNetworks.create_and_update_routers	10
NeutronNetworks.create_and_update_subnets	10
NeutronNetworks.create_and_delete_networks	10
NeutronNetworks.create_and_delete_ports	10
NeutronNetworks.create_and_delete_routers	10
NeutronNetworks.create_and_delete_subnets	10
NeutronNetworks.create_and_list_networks	10
NeutronNetworks.create_and_list_ports	10
NeutronNetworks.create_and_list_routers	10
NeutronNetworks.create_and_list_subnets	10
NeutronSecurityGroup.create_and_delete_security_groups	10
NeutronSecurityGroup.create_and_delete_security_group_rule	10
NeutronSecurityGroup.create_and_list_security_group_rules	10
NeutronSecurityGroup.create_and_show_security_group	10
NeutronNetworks.set_and_clear_router_gateway	10
NeutronNetworks.create_and_show_ports	10
NeutronNetworks.create_and_show_routers	10
NeutronNetworks.create_and_show_subnets	10
Quotas.neutron_update	10

Functest rally_jobs_cntt [187]:

Table 8.17: Neutron API benchmarking

Scenarios	Iterations
NeutronNetworks.create_and_delete_networks	40
NeutronNetworks.create_and_delete_ports	40
NeutronNetworks.create_and_delete_routers	40
NeutronNetworks.create_and_delete_subnets	40
NeutronNetworks.create_and_list_networks	100
NeutronNetworks.create_and_list_ports	8
NeutronNetworks.create_and_list_routers	40
NeutronNetworks.create_and_list_subnets	40
NeutronNetworks.create_and_update_networks	40
NeutronNetworks.create_and_update_ports	40
NeutronNetworks.create_and_update_routers	40
NeutronNetworks.create_and_update_subnets	100
NeutronTrunks.create_and_list_trunks	4
Quotas.neutron_update	40

Compute - Nova API benchmarking

Functest rally_full_cntt [186]:

Table 8.18: Nova API benchmarking

Scenarios	Iterations
Authenticate.validate_nova	10
NovaKeypair.create_and_delete_keypair	10
NovaKeypair.create_and_list_keypairs	10
NovaServers.boot_and_bounce_server	10
NovaServers.boot_and_delete_server	10
NovaServers.boot_and_list_server	10
NovaServers.boot_and_rebuild_server	10
NovaServers.snapshot_server	10
NovaServers.boot_server_from_volume	10
NovaServers.boot_server	10
NovaServers.list_servers	10
NovaServers.resize_server	10
NovaServers.boot_and_live_migrate_server	10
NovaServers.boot_server_attach_created_volume_and_live_migrate	10
NovaServers.boot_server_from_volume_and_live_migrate	10
NovaKeypair.boot_and_delete_server_with_keypair	10
NovaServers.boot_server_from_volume_and_delete	10
NovaServers.pause_and_unpause_server	10
NovaServers.boot_and_migrate_server	10
NovaServers.boot_server_and_list_interfaces	10
NovaServers.boot_and_get_console_url	10
NovaServers.boot_server_and_attach_interface	10
NovaServers.boot_server_attach_volume_and_list_attachments	10
NovaServers.boot_server_associate_and_dissociate_floating_ip	10
NovaServers.boot_and_associate_floating_ip	10
NovaServerGroups.create_and_delete_server_group	10
NovaServerGroups.create_and_get_server_group	10
NovaServerGroups.create_and_list_server_groups	10
Quotas.nova_update	10

Orchestration - Heat API benchmarking

Functest rally_full_cntt [186]:

Table 8.19: Heat API benchmarking

Scenarios	Iterations
Authenticate.validate_heat	10
HeatStacks.create_and_delete_stack	10
HeatStacks.create_and_list_stack	10
HeatStacks.create_update_delete_stack	10
HeatStacks.create_check_delete_stack	10
HeatStacks.create_suspend_resume_delete_stack	10
HeatStacks.list_stacks_and_resources	10

Dataplane benchmarking

Functest Benchmarking CNTT [185] offers two benchmarking dataplane test cases leveraging on:

- VMTP [188]
- Shaker [189]

VMTP [188] is a small python application that will automatically perform ping connectivity, round trip time measurement (latency) and TCP/UDP throughput measurement on any OpenStack deployment.

[189] wraps around popular system network testing tools like iperf, iperf3 and netperf (with help of flent). [189] is able to deploy OpenStack instances and networks in different topologies. [189] scenario specifies the deployment and list of tests to execute.

The SLA is the default SLA proposed in Functest Benchmarking CNTT [185].

VMTP

Here are the scenarios [190] executed by Functest [191]: - VM to VM same network fixed IP (intra-node) - VM to VM different network fixed IP (intra-node) - VM to VM different network floating IP (intra-node) - VM to VM same network fixed IP (inter-node) - VM to VM different network fixed IP (inter-node) - VM to VM different network floating IP (inter-node)

Here are all results per scenario:

Table 8.20: All results per scenario

protocol	pkt_size	results
ICMP	64	rtt_avg_ms
ICMP	64	rtt_max_ms
ICMP	64	rtt_min_ms
ICMP	64	rtt_stddev
ICMP	391	rtt_avg_ms
ICMP	391	rtt_max_ms
ICMP	391	rtt_min_ms
ICMP	391	rtt_stddev
ICMP	1500	rtt_avg_ms
ICMP	1500	rtt_max_ms
ICMP	1500	rtt_min_ms
ICMP	1500	rtt_stddev
UDP	128	loss_rate
UDP	128	throughput_kbps
UDP	1024	loss_rate
UDP	1024	throughput_kbps
UDP	8192	loss_rate
UDP	8192	throughput_kbps
TCP	65536	rtt_ms
TCP	65536	throughput_kbps

Shaker

Here are the scenarios [192] executed by Shaker:

- OpenStack L2
- OpenStack L3 East-West
- OpenStack L3 North-South
- OpenStack L3 North-South Performance

Here are all samples:

Table 8.21: All samples

test	samples
Bi-directional	ping_icmp (ms)
Bi-directional	tcp_download (Mbits/s)
Bi-directional	tcp_upload (Mbits/s)
Download	ping_icmp (ms)
Download	tcp_download (Mbits/s)
Upload	ping_icmp (ms)
Upload	tcp_upload (Mbits/s)
Ping	ping_icmp (ms)
Ping	ping_udp (ms)
TCP	bandwidth (bit/s)
TCP	retransmits
UDP	packets (pps)

Open-source VNF onboarding and testing

Running open-source VNFs is a key technical solution to ensure that the platforms meet Network Functions Virtualisation requirements. Functest VNF [193] offers 5 test cases which automatically onboard and test the following 3 open-source VNFs:

- Clearwater IMS [194]
- VyOS vRouter [195]
- OpenAirInterface vEPC [196]

Here are the full list of orchestrators used for all these deployments:

- Cloudify [197]
- Heat [63]
- Juju [198]

The VNF are covered by upstream tests when possible (see clearwater-live-test [199]) and by Functest VNF tests in the other cases.

8.4 Test Cases Traceability to Requirements

8.4.1 RM/RA-1 Requirements

The following test cases must pass as they are for OpenStack based cloud infrastructure Conformance:

Table 8.22: OpenStack based cloud infrastructure Conformance

container	test case	criteria
opnfv/functest-healthcheck:wallaby	tempest_horizon	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_neutron_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_cinder_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_keystone_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	rally_sanity_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_full_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_scenario_cntt	PASS
opnfv/functest-smoke-cntt:wallaby	tempest_slow_cntt	PASS
opnfv/functest-benchmarking-cntt:wallaby	rally_full_cntt	PASS
opnfv/functest-benchmarking-cntt:wallaby	rally_jobs_cntt	PASS
opnfv/functest-benchmarking-cntt:wallaby	vmtp	PASS
opnfv/functest-benchmarking-cntt:wallaby	shaker	PASS
opnfv/functest-vnf:wallaby	cloudify	PASS
opnfv/functest-vnf:wallaby	cloudify_ims	PASS
opnfv/functest-vnf:wallaby	heat_ims	PASS
opnfv/functest-vnf:wallaby	vyos_vrouter	PASS
opnfv/functest-vnf:wallaby	juju_epc	PASS

8.4.2 TC Mapping to Requirements

test case	requirements
tempest_horizon	Horizon testing
tempest_neutron_cntt	Neutron API testing
tempest_cinder_cntt	Cinder API testing
tempest_keystone_cntt	Keystone API testing
rally_sanity_cntt	Keystone, Glance, Cinder, Swift, Neutron, Nova and Heat API testing
tempest_full_cntt	Keystone, Glance, Cinder, Swift, Neutron and Nova API testing
tempest_scenario_cntt	Keystone, Glance, Cinder, Swift, Neutron and Nova API testing
tempest_slow_cntt	Keystone, Glance, Cinder, Swift, Neutron and Nova API testing
rally_full_cntt	Keystone, Glance, Cinder, Swift, Neutron, Nova and Heat API benchmarking
rally_jobs_cntt	Neutron API benchmarking
vmtp	Dataplane benchmarking
shaker	Dataplane benchmarking
cloudify	open-source VNF onboarding and testing
cloudify_ims	open-source VNF onboarding and testing
heat_ims	open-source VNF onboarding and testing
vyos_vrouter	open-source VNF onboarding and testing
juju_epc	open-source VNF onboarding and testing

8.5 OpenStack Testing Cookbook

Please note the next two points depending on the GNU/Linux distributions and the network settings:

- SELinux: you may have to add `--system-site-packages` when creating the virtualenv (“Aborting, target uses selinux but python bindings (libselinux-python) aren’t installed!”)
- Proxy: you may set your proxy in env for Ansible and in systemd for Docker [200]

To deploy your own CI toolchain running OpenStack based cloud infrastructure Conformance:

```
virtualenv functest --system-site-packages
. functest/bin/activate
pip install ansible
ansible-galaxy install collivier.xtesting
ansible-galaxy collection install ansible.posix community.general community.grafana
↪kubernetes.core community.docker community.postgresql
git clone https://gerrit.opnfv.org/gerrit/functest functest-src
(cd functest-src && git checkout -b stable/wallaby origin/stable/wallaby)
ansible-playbook functest-src/ansible/site.cntt.yml
```

8.5.1 OpenStack API testing configuration

Here is the default Functest tree as proposed in Functest Wallaby [156]:

- `/home/opnfv/functest/openstack.creds`
- `/home/opnfv/functest/images`

Download the images and fill `/home/opnfv/functest/openstack.creds` as proposed in Functest Wallaby [156].

You may have to modify a few Functest env vars according to the SUT (see env in Functest Wallaby [156]). Be free to modify `functest-src/ansible/host_vars/127.0.0.1` at your convenience and then to reconfigure the toolchain:

```
ansible-playbook functest-src/ansible/site.cntt.yml
```

8.5.2 Run OpenStack based cloud infrastructure Conformance

Open <http://127.0.0.1:8080/job/functest-wallaby-daily/> in a web browser, login as admin/admin and click on “Build with Parameters” (keep the default build_tag value).

If the System under test (SUT) is compliant, a link to the full archive containing all test results and artifacts will be printed in functest-wallaby-zip’s console. Be free to download it and then to send it to any reviewer committee.

To clean your working directory:

```
deactivate
rm -rf functest-src functest
```

9 Gaps, Innovation, and Development

The purpose of this chapter is to identify the gaps between what is required for automated deployment of VNFs on Cloud Infrastructure frameworks and the framework offered by OpenStack. Once gaps are identified, the next step will be to propose a plan to address these gaps. The most obvious way to address the gaps will be to propose a set of APIs in the upstream OpenStack community

9.1 The Gap

9.1.1 Autoscaling

With regards to resource autoscaling (gen.scl.01 *General Recommendations*) it is recommended that the NFVO/VNFM manages the policy and triggers a scale-up or scale-down action based on application telemetry, event, AI, or ML etc. While the use of telemetry and alarming system can trigger a scaling operation based on resource utilisation, without application context this may not provide the granularity or reaction time required by the application. It is therefore suggested that an OpenStack scaling operation is called using an appropriate autoscaling web-hook by the NFVO/VNFM.

For more information on auto-scaling with Heat please see the OpenStack document “Autoscaling with heat [201]”. Please note that the OpenStack Senlin service is still under development with major architectural changes made in the OpenStack Ussuri release.

Please note: physical compute node autoscaling is out of scope.

References

- [1] Cloud infrastructure reference model. GSMA PRD NG.126 v3.0, 2022.
- [2] Openstack documentation. URL: <https://docs.openstack.org/>.
- [3] Network functions virtualisation (nfv); infrastructure overview. ETSI GS NFV-INF 001 V1.1.1, January 2015. URL: https://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf.
- [4] Openstack use cases. URL: <https://docs.openstack.org/arch-design/use-cases.html>.
- [5] Open vswitch. URL: <https://www.openvswitch.org/>.
- [6] OpenStack Community. Openstack wallaby projects. URL: <https://docs.openstack.org/wallaby/projects.html>.
- [7] Scott O. Bradner. Key words for use in rfcs to indicate requirement levels. RFC 2119, March 1997. URL: <https://www.rfc-editor.org/info/rfc2119>, doi:10.17487/RFC2119.
- [8] H. Philip White. Center for internet security - password policy guide. 2020. URL: <https://www.cisecurity.org/insights/white-papers/cis-password-policy-guide>.
- [9] Center for internet security - controls v7.1. URL: <https://www.cisecurity.org/controls/cis-controls-list>.
- [10] Openstack - cpu dedicated set. URL: https://docs.openstack.org/nova/latest/configuration/config.html#compute.cpu_dedicated_set.
- [11] Openstack - cpu topologies. URL: <https://docs.openstack.org/nova/latest/admin/cpu-topologies.html>.
- [12] Openstack - neutron plugins and drivers. URL: https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers.
- [13] Openstack - tags. URL: <https://specs.openstack.org/openstack/api-wg/guidelines/tags.html>.

- [14] Openstack - configuring the stateful services. URL: <https://docs.openstack.org/ha-guide/control-plane-stateful.html>.
- [15] Openstack - senlin documentation. URL: <https://docs.openstack.org/senlin/wallaby/>.
- [16] Openstack - neutron ovs agent support for baremetal with smart nic. URL: <https://specs.openstack.org/openstack/neutron-specs/specs/stein/neutron-ovs-agent-support-baremetal-with-smart-nic.html>.
- [17] National telecommunications and information administration - software bill of materials. URL: <https://www.ntia.gov/SBOM>.
- [18] Center for internet security. URL: <https://www.cisecurity.org/>.
- [19] Cloud security alliance. URL: <https://cloudsecurityalliance.org/>.
- [20] Open web application security project - cheat sheet series. URL: <https://github.com/OWASP/CheatSheetSeries>.
- [21] Open web application security project. URL: <https://www.owasp.org>.
- [22] Open web application security project - top ten security risks. URL: <https://owasp.org/www-project-top-ten/>.
- [23] Open web application security project - software maturity model (samm). URL: <https://owaspsamm.org/blog/2019/12/20/version2-community-release/>.
- [24] Open web application security project - web security testing guide. URL: <https://github.com/OWASP/wstg/tree/master/document>.
- [25] Iso (international organization for standardization) and iec (international electrotechnical commission) iso/iec 27001:2013. 2013. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en>.
- [26] Iso (international organization for standardization) and iec (international electrotechnical commission) iso/iec 27002:2013. 2013. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27002:ed-2:v1:en>.
- [27] Iso (international organization for standardization) and iec (international electrotechnical commission) iso/iec 7032:2012. 2012. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27032:ed-1:v1:en>.
- [28] Openstack storage. URL: <https://docs.openstack.org/arch-design/design-storage/design-storage-concepts.html#table-openstack-storage>.
- [29] Openstack cinder driver support matrix. URL: <https://docs.openstack.org/cinder/latest/reference/support-matrix.html>.
- [30] Tungsten fabric - multicloud multistack sdn. URL: <https://tungsten.io>.
- [31] Openstack glossary. URL: <https://docs.openstack.org/doc-contrib-guide/common/glossary.html>.
- [32] Openstack feature support matrix. URL: <https://docs.openstack.org/nova/latest/user/support-matrix.html>.
- [33] Openstack storage architecture design. URL: <https://docs.openstack.org/arch-design/design-storage.html>.
- [34] Openstack nova: kvm. URL: <https://docs.openstack.org/nova/wallaby/admin/configuration/hypervisor-kvm.html>.
- [35] Openstack - hardening the virtualization layers. URL: <https://docs.openstack.org/security-guide/compute/hardening-the-virtualization-layers.html>.
- [36] Openstack reference architecture for 100, 300 and 500 nodes. URL: https://fuel-ccp.readthedocs.io/en/latest/design/ref_arch_100_nodes.html.
- [37] (dpdk) release notes. URL: http://doc.dpdk.org/guides/rel_notes.
- [38] (dpdk) performance reports. URL: <http://core.dpdk.org/perf-reports/>.
- [39] Robert Moskowitz, Daniel Karrenberg, Yakov Rekhter, Eliot Lear, and Geert Jan de Groot. Address allocation for private internets. RFC 1918, February 1996. URL: <https://www.rfc-editor.org/info/rfc1918>, doi:10.17487/RFC1918.

- [40] Openstack - introducing octavia. URL: <https://docs.openstack.org/octavia/latest/reference/introduction.html>.
- [41] Openstack octavia (load-balancer service). URL: <https://governance.openstack.org/tc/reference/projects/octavia.html>.
- [42] Openstack/neutron-vpnaas. URL: <https://opendev.org/openstack/neutron-vpnaas>.
- [43] Openstack neutron: plugins. URL: <https://wiki.openstack.org/wiki/Neutron#Plugins>.
- [44] Openstack neutron: api extensions. URL: https://docs.openstack.org/neutron/latest/contributor/internals/api_extensions.html.
- [45] Openstack networking api v2.0: list extensions. URL: <https://docs.openstack.org/api-ref/network/v2/#list-extensions>.
- [46] Openstack networking api v2.0: show extension details. URL: <https://docs.openstack.org/api-ref/network/v2/#show-extension-details>.
- [47] Openstack neutron/ml2. URL: <https://wiki.openstack.org/wiki/Neutron/ML2>.
- [48] Openstack cinder driver support matrix. URL: <https://docs.openstack.org/cinder/latest/reference/support-matrix.html>.
- [49] Openstack (cinder) available drivers. URL: <https://docs.openstack.org/cinder/latest/drivers.html>.
- [50] Openstack cinder service configuration. URL: <https://docs.openstack.org/cinder/latest/configuration/index.html>.
- [51] Openstack cinder administration. URL: <https://docs.openstack.org/cinder/latest/admin/index.html>.
- [52] Ceph - the future of storage. URL: <https://ceph.io/en>.
- [53] Keystone, the openstack identity service. URL: <https://docs.openstack.org/keystone/wallaby/>.
- [54] Openstack - welcome to glance's documentation! URL: <https://docs.openstack.org/glance/wallaby/>.
- [55] Openstack block storage (cinder) documentation. URL: <https://docs.openstack.org/cinder/wallaby/>.
- [56] Openstack - welcome to swift's documentation! URL: <https://docs.openstack.org/swift/wallaby/>.
- [57] Openstack - welcome to neutron's documentation! URL: <https://docs.openstack.org/neutron/wallaby/>.
- [58] Openstack - scenario: high availability using distributed virtual routing (dvr). URL: <https://docs.openstack.org/liberty/networking-guide/scenario-dvr-ovs.html>.
- [59] Openstack neutron: distributed virtual routing with vrrp. URL: <https://docs.openstack.org/neutron/wallaby/admin/config-dvr-ha-snat.html>.
- [60] Openstack compute (nova). URL: <https://docs.openstack.org/nova/wallaby/>.
- [61] Openstack - welcome to ironic's documentation! URL: <https://docs.openstack.org/ironic/wallaby/>.
- [62] Openstack ironic api reference: bare metal api. URL: <https://docs.openstack.org/api-ref/baremetal/>.
- [63] Openstack - welcome to the heat documentation! URL: <https://docs.openstack.org/heat/wallaby/>.
- [64] Horizon: the openstack dashboard project. URL: <https://docs.openstack.org/horizon/wallaby/>.
- [65] Openstack - placement. URL: <https://docs.openstack.org/placement/wallaby/index.html>.
- [66] Openstack - placement: modeling with provider trees. URL: <https://docs.openstack.org/placement/latest/user/provider-tree.html>.
- [67] Openstack - placement usage. URL: <https://docs.openstack.org/placement/latest/user/index.html>.
- [68] Openstack key manager (barbican). URL: <https://docs.openstack.org/barbican/wallaby/>.
- [69] Openstack accelerator (cyborg). URL: <https://docs.openstack.org/cyborg/wallaby/>.

- [70] Openstack compute api guide 2.1.0: server concepts. URL: https://docs.openstack.org/api-guide/compute/server_concepts.html.
- [71] Openstack cyborg support matrix (wallaby). URL: <https://docs.openstack.org/cyborg/wallaby/reference/support-matrix.html>.
- [72] Openstack cyborg support matrix. URL: <https://docs.openstack.org/cyborg/latest/reference/support-matrix.html>.
- [73] Openstack cyborg architecture. URL: <https://docs.openstack.org/cyborg/latest/user/architecture.html>.
- [74] Openstack nova: flavors. URL: <https://docs.openstack.org/nova/latest/user/flavors.html>.
- [75] Open glossary of edge computing. URL: <https://github.com/State-of-the-Edge/glossary/blob/master/edge-glossary.md>.
- [76] Openstack - edge computing: next steps in architecture, design and testing. URL: <https://www.openstack.org/use-cases/edge-computing/edge-computing-next-steps-in-architecture-design-and-testing>.
- [77] Openstack reference architecture for 100, 300 and 500 nodes: services placement summary. URL: https://fuel-ccp.readthedocs.io/en/latest/design/ref_arch_100_nodes.html#services-placement-summary.
- [78] Openstack nova: image pre-caching. URL: <https://docs.openstack.org/nova/latest/admin/image-caching.html#image-pre-caching>.
- [79] Airship v2. URL: <https://www.airshipit.org/>.
- [80] Starlingx - deploy your edge cloud now. URL: <https://www.starlingx.io/>.
- [81] Openstack tripleo. URL: <http://opendev.org/openstack/tripleo-common>.
- [82] Openstack compute microversions. URL: <https://docs.openstack.org/api-guide/compute/microversions.html>.
- [83] Identity api v3. URL: <https://docs.openstack.org/api-ref/identity/v3/index.html>.
- [84] Identity api v3 extensions. URL: <https://docs.openstack.org/api-ref/identity/v3-ext/>.
- [85] Security compliance and pci-dss. URL: <https://docs.openstack.org/keystone/wallaby/admin/configuration.html#security-compliance-and-pci-dss>.
- [86] Image service api. URL: <https://docs.openstack.org/api-ref/image/v2/>.
- [87] Image service versions. URL: <https://docs.openstack.org/api-ref/image/versions/index.html#version-history>.
- [88] Block storage api. URL: <https://docs.openstack.org/api-ref/block-storage/>.
- [89] Cinder rest api version history. URL: https://docs.openstack.org/cinder/latest/contributor/api_microversion_history.html.
- [90] Object storage api. URL: <https://docs.openstack.org/api-ref/object-store/index.html>.
- [91] Discoverability. URL: <https://docs.openstack.org/swift/latest/api/discoverability.html>.
- [92] Networking service apis. URL: <https://docs.openstack.org/api-ref/network/>.
- [93] Networking api v2.0. URL: <https://docs.openstack.org/api-ref/network/v2/>.
- [94] Compute api. URL: <https://docs.openstack.org/api-ref/compute/>.
- [95] Compute rest api version history. URL: <https://docs.openstack.org/nova/latest/reference/api-microversion-history.html>.
- [96] Placement api. URL: <https://docs.openstack.org/api-ref/placement/>.
- [97] Placement rest api version history. URL: <https://docs.openstack.org/placement/latest/placement-api-microversion-history.html>.
- [98] Orchestration service api. URL: <https://docs.openstack.org/api-ref/orchestration/>.

- [99] Template version history. URL: https://docs.openstack.org/heat/latest/template_guide/hot_spec.html.
- [100] Heat orchestration template (hot) specification. URL: https://docs.openstack.org/heat/latest/template_guide/hot_spec.html#rocky.
- [101] Openstack apis. URL: <https://docs.openstack.org/api-ref/>.
- [102] Kubernetes apis. URL: <https://kubernetes.io/docs/concepts/overview/kubernetes-api/>.
- [103] Kvm apis. URL: <https://www.kernel.org/doc/Documentation/virtual/kvm/api.txt>.
- [104] Libvirt apis. URL: <https://libvirt.org/html/index.html>.
- [105] Barbican api. URL: <https://docs.openstack.org/barbican/latest/api/>.
- [106] Openstack - security boundaries and threats. URL: <https://docs.openstack.org/security-guide/introduction/security-boundaries-and-threats.html>.
- [107] Openstack security guide. URL: <https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html>.
- [108] Mitre - common vulnerabilities and exposures. URL: <https://cve.mitre.org/>.
- [109] National institute of standards and technology vulnerabilities metrics. URL: <https://nvd.nist.gov/vuln-metrics/cvss>.
- [110] Openstack security guide - identity. URL: <https://docs.openstack.org/security-guide/identity.html>.
- [111] Openstack security guide - authentication methods. URL: <https://docs.openstack.org/security-guide/identity/authentication-methods.html>.
- [112] Openstack security guide - policies. URL: <https://docs.openstack.org/security-guide/identity/policies.html#policy-section>.
- [113] Openstack keystone default roles. URL: <https://docs.openstack.org/keystone/latest/admin/service-api-protection.html>.
- [114] Openstack - introduction to tls and ssl. URL: <https://docs.openstack.org/security-guide/secure-communication/introduction-to-ssl-and-tls.html>.
- [115] Center for internet security cis-cat pro. URL: <https://www.cisecurity.org/cybersecurity-tools/cis-cat-pro/>.
- [116] Center for internet security benchmarks. URL: <https://www.cisecurity.org/cis-benchmarks/>.
- [117] Openstack image signature verification. URL: <https://docs.openstack.org/glance/wallaby/user/signature.html>.
- [118] Openstack - sr-iov passthrough for networking. URL: <https://wiki.openstack.org/wiki/SR-IOV-Passthrough-For-Networking>.
- [119] Openstack trusted images. URL: <https://docs.openstack.org/security-guide/instance-management/security-services-for-instances.html#trusted-images>.
- [120] Adding signed images. URL: <https://docs.openstack.org/operations-guide/ops-user-facing-operations.html#adding-signed-images>.
- [121] Network functions virtualisation (nfv) release 4; protocols and data models; vnf package and pnfd archive specification. ETSI GS NFV-SOL 004 V4.3.1, July 2022. URL: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/04.03.01_60/gs_NFV-SOL004v040301p.pdf.
- [122] Network functions virtualisation (nfv) release 2; security; vnf package security specification. ETSI GS NFV-SEC 021 V2.6.1, June 2019. URL: https://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/021/02.06.01_60/gs_nfv-sec021v020601p.pdf.
- [123] Foreman. URL: <https://www.theforeman.org/>.
- [124] Ansible documentation. URL: <https://docs.ansible.com/>.

- [125] Openstack tripleo architecture. URL: <https://docs.openstack.org/tripleo-docs/latest/install/introduction/architecture.html#project-architecture>.
- [126] Ovp. URL: <https://www.opnfv.org/verification>.
- [127] Jenkins. URL: <https://build.opnfv.org/>.
- [128] Test dabase. URL: https://docs.opnfv.org/en/stable-hunter/_images/OPNFV_testing_working_group.png.
- [129] S3 compatible storage service. URL: <http://artifacts.opnfv.org/>.
- [130] Functest-wallaby-zip. URL: <https://build.opnfv.org/ci/job/functest-wallaby-zip/4/console>.
- [131] Xtesting ci. URL: <https://galaxy.ansible.com/collivier/xtesting>.
- [132] Docker. URL: <https://www.docker.com/>.
- [133] Xtesting. URL: <https://xtesting.readthedocs.io/en/latest/>.
- [134] Opnfv fraser. URL: <https://www.sdxcentral.com/articles/news/opnfvs-6th-release-brings-testing-capabilities-that-orange-is-already-2018/05/>.
- [135] Xtesting python package. URL: <https://pypi.org/project/xtesting/>.
- [136] Test case execution description. URL: <https://git.opnfv.org/functest-xtesting/tree/docker/core/testcases.yaml>.
- [137] Ci/cd toolchains in a few commands. URL: <https://github.com/collivier/ansible-role-xtesting#readme>.
- [138] Ci/cd deployment models. URL: <https://lists.opnfv.org/g/opnfv-tsc/message/5702>.
- [139] Anuket releng. URL: <https://git.opnfv.org/releng/tree/jjb/functest>.
- [140] Test case result dump. URL: <http://artifacts.opnfv.org/functest/9ID39XK47PMZ.zip>.
- [141] Xtesting samples. URL: <https://git.opnfv.org/functest-xtesting/plain/ansible/site.yml?h=stable/wallaby>.
- [142] Openstack verification. URL: <https://git.opnfv.org/functest/plain/ansible/site.yml?h=stable/wallaby>.
- [143] Anuket rc1. URL: <https://git.opnfv.org/functest/plain/ansible/site.cntt.yml?h=stable/wallaby>.
- [144] Kubernetes verification. URL: <https://git.opnfv.org/functest-kubernetes/plain/ansible/site.yml?h=stable/v1.22>.
- [145] Functest. URL: <https://functest.readthedocs.io/en/stable-wallaby/>.
- [146] Refstack. URL: <https://refstack.openstack.org/>.
- [147] Networking bgpvpn. URL: <https://docs.openstack.org/networking-bgpvpn/latest/>.
- [148] Networking sfc. URL: <https://docs.openstack.org/networking-sfc/latest/>.
- [149] Devstack gates. URL: <https://docs.openstack.org/developer/system-config/latest/devstack-gate.html>.
- [150] Rally. URL: <https://github.com/openstack/rally-openstack>.
- [151] Temptest. URL: <https://github.com/openstack/tempest>.
- [152] Temptest. URL: <https://docs.openstack.org/devstack/latest/>.
- [153] Raspberry pi. URL: <https://www.raspberrypi.org/>.
- [154] Functest daily jobs. URL: <https://build.opnfv.org/ci/view/functest/job/functest-wallaby-daily/17/>.
- [155] Openstack performance tools. URL: <https://docs.openstack.org/developer/performance-docs/methodologies/tools.html>.
- [156] Run alpine functest containers (wallaby). URL: <https://wiki.anuket.io/display/HOME/Functest+Wallaby>.
- [157] Functest gates. URL: <https://build.opnfv.org/ci/view/functest>.
- [158] New functest cntt containers. URL: <https://lists.opnfv.org/g/opnfv-tsc/message/5717>.

- [159] Keystone-tempest-plugin. URL: <https://opendev.org/openstack/keystone-tempest-plugin>.
- [160] Functest smoke cntt. URL: <https://git.opnfv.org/functest/tree/docker/smoke-cntt/testcases.yaml?h=stable%2Fwallaby>.
- [161] Cinder-tempest-plugin. URL: <https://opendev.org/openstack/cinder-tempest-plugin>.
- [162] Functest review 68881. URL: <https://gerrit.opnfv.org/gerrit/68881>.
- [163] Functest review 71011. URL: <https://gerrit.opnfv.org/gerrit/71011>.
- [164] Openstack bug 1770179. URL: <https://launchpad.net/bugs/1770179>.
- [165] Openstack bug 1677525. URL: <https://launchpad.net/bugs/1677525>.
- [166] Openstack bug 1317133. URL: <https://launchpad.net/bugs/1317133>.
- [167] Openstack bug 1905432. URL: <https://launchpad.net/bugs/1905432>.
- [168] Neutron-tempest-plugin. URL: <https://opendev.org/openstack/neutron-tempest-plugin>.
- [169] Openstack bug 1863707. URL: <https://launchpad.net/bugs/1863707>.
- [170] Functest review 69105. URL: <https://gerrit.opnfv.org/gerrit/69105>.
- [171] Openstack bug 1676207. URL: <https://launchpad.net/bugs/1676207>.
- [172] Openstack bug 1836595. URL: <https://launchpad.net/bugs/1836595>.
- [173] Openstack bug 1186354. URL: <https://launchpad.net/bugs/1186354>.
- [174] Openstack bug 1014647. URL: <https://launchpad.net/bugs/1014647>.
- [175] Openstack bug 1311500. URL: <https://launchpad.net/bugs/1311500>.
- [176] Openstack bug 1161411. URL: <https://launchpad.net/bugs/1161411>.
- [177] Openstack bug 1540645. URL: <https://launchpad.net/bugs/1540645>.
- [178] Heat-tempest-plugin. URL: <https://opendev.org/openstack/heat-tempest-plugin>.
- [179] Openstack story 2007804. URL: <https://storyboard.openstack.org/#!/story/2007804>.
- [180] Functest review 69926. URL: <https://gerrit.opnfv.org/gerrit/69926>.
- [181] Functest review 69931. URL: <https://gerrit.opnfv.org/gerrit/69931>.
- [182] Functest review 70004. URL: <https://gerrit.opnfv.org/gerrit/70004>.
- [183] Tempest-horizon. URL: <https://github.com/openstack/tempest-horizon>.
- [184] Functest healthcheck. URL: <https://git.opnfv.org/functest/tree/docker/healthcheck/testcases.yaml?h=stable%2Fwallaby>.
- [185] Functest benchmarking cntt. URL: <https://git.opnfv.org/functest/tree/docker/benchmarking-cntt/testcases.yaml?h=stable%2Fwallaby>.
- [186] Rally_full_cntt. URL: http://artifacts.opnfv.org/functest/KDBNITEN317M/functest-opnfv-functest-benchmarking-cntt-wallaby-rally_full_cntt-run-5/rally_full_cntt/rally_full_cntt.html.
- [187] Rally_jobs_cntt. URL: http://artifacts.opnfv.org/functest/KDBNITEN317M/functest-opnfv-functest-benchmarking-cntt-wallaby-rally_jobs_cntt-run-5/rally_jobs_cntt/rally_jobs_cntt.html.
- [188] Vmtp. URL: <http://vmtp.readthedocs.io/en/latest>.
- [189] Shaker. URL: <https://pyshaker.readthedocs.io/en/latest/>.

- [190] Vmtp scenarios. URL: <http://artifacts.opnfv.org/funcstest/KDBNITEN317M/funcstest-opnfv-funcstest-benchmarking-wallaby-vmtp-run-8/vmtp/vmtp.json>.
- [191] Functest vmtp. URL: <http://artifacts.opnfv.org/funcstest/KDBNITEN317M/funcstest-opnfv-funcstest-benchmarking-wallaby-vmtp-run-8/vmtp/vmtp.html>.
- [192] Shaker scenarios. URL: <http://artifacts.opnfv.org/funcstest/KDBNITEN317M/funcstest-opnfv-funcstest-benchmarking-wallaby-shaker-run-8/shaker/report.json>.
- [193] Functest vnf. URL: <https://git.opnfv.org/funcstest/tree/docker/vnf/testcases.yaml?h=stable%2Fwallaby>.
- [194] Clearwater ims. URL: <https://clearwater.readthedocs.io/en/stable/>.
- [195] Vyos vrouter. URL: <https://www.vyos.io/>.
- [196] Openairinterface vepc. URL: <https://www.openairinterface.org/>.
- [197] Cloudify. URL: <https://cloudify.co>.
- [198] Juju. URL: <https://jaas.ai/>.
- [199] Clearwater-live-test. URL: <https://github.com/Metaswitch/clearwater-live-test>.
- [200] Docker http/https proxy. URL: <https://docs.docker.com/config/daemon/systemd/#httphttps-proxy>.
- [201] Openstack autoscaling with heat. URL: https://docs.openstack.org/senlin/latest/scenarios/autoscaling_heat.html.