Reference Architecture for OpenStack managed Virtualisation

**Table of Contents**

[1. Introduction 9](#_Toc80264182)

[1.1. Overview 9](#_Toc80264183)

[1.1.1. Vision 9](#_Toc80264184)

[1.2. Use Cases 9](#_Toc80264185)

[1.3. Anuket OpenStack Reference Release 10](#_Toc80264186)

[1.4. Principles 10](#_Toc80264187)

[1.4.1. OpenStack specific principles 10](#_Toc80264188)

[1.5. Document Organisation 11](#_Toc80264189)

[1.6. Terminology 11](#_Toc80264190)

[1.7. Abbreviations 16](#_Toc80264191)

[1.8. References 19](#_Toc80264192)

[1.9. Conventions 25](#_Toc80264193)

[2. Architecture Requirements 25](#_Toc80264194)

[2.1. Introduction 25](#_Toc80264195)

[2.2. Reference Model Requirements 25](#_Toc80264196)

[2.2.1. Cloud Infrastructure Software Profile Requirements for Compute (source RM 5.2 [1]) 26](#_Toc80264197)

[2.2.1.1. Cloud Infrastructure Software Profile Extensions Requirements for Compute 27](#_Toc80264198)

[2.2.2. Cloud Infrastructure Software Profile Requirements for Networking (source RM 5.2.3 [1]) 27](#_Toc80264199)

[2.2.2.1. Cloud Infrastructure Software Profile Extensions Requirements for Networking 28](#_Toc80264200)

[2.2.3. Cloud Infrastructure Software Profile Requirements for Storage (source RM 5.2 [1] 29](#_Toc80264201)

[2.2.3.1. Cloud Infrastructure Software Profile Extensions Requirements for Storage 30](#_Toc80264202)

[2.2.4. Cloud Infrastructure Hardware Profile Requirements (source RM 5.4 [1]) 30](#_Toc80264203)

[2.2.4.1. Cloud Infrastructure Hardware Profile-Extensions Requirements (source RM 5.4 [1]) 31](#_Toc80264204)

[2.2.5. Cloud Infrastructure Management Requirements (source RM 4.1.5 [1]) 32](#_Toc80264205)

[2.2.6. Cloud Infrastructure Security Requirements 32](#_Toc80264206)

[2.2.6.1. System Hardening (source RM 7.9.1 [1]) 32](#_Toc80264207)

[2.2.6.2. Platform and Access (source RM 7.9.2 [1]) 34](#_Toc80264208)

[2.2.6.3. Confidentiality and Integrity (source RM7.9.3 [1]) 36](#_Toc80264209)

[2.2.6.4. Workload Security (source RM7.9.4 [1] 37](#_Toc80264210)

[2.2.6.5. Image Security (source RM7.9.5 [1] 38](#_Toc80264211)

[2.2.6.6. Security LCM (source RM7.9.6 [1]) 39](#_Toc80264212)

[2.2.6.7. Monitoring and Security Audit (source RM7.9.7 [1] 40](#_Toc80264213)

[2.2.6.8. Open Source Software (source RM7.9.8 [1] 43](#_Toc80264214)

[2.2.6.9. IaaC security (source RM7.9.9 [1]) 43](#_Toc80264215)

[2.2.6.9.1. Secure Code Stage Requirements 43](#_Toc80264216)

[2.2.6.9.2. Continuous Build, Integration and Testing Stage Requirements 44](#_Toc80264217)

[2.2.6.9.3. Continuous Delivery and Deployment Stage Requirements 44](#_Toc80264218)

[2.2.6.9.4. Runtime Defence and Monitoring Requirements 45](#_Toc80264219)

[2.2.6.10. Compliance with Standards (source RM7.9.10 [1]) 45](#_Toc80264220)

[2.3. Architecture and OpenStack Requirements 46](#_Toc80264221)

[2.3.1. General Requirements 46](#_Toc80264222)

[2.3.2. Infrastructure Requirements 46](#_Toc80264223)

[2.3.3. VIM Requirements 48](#_Toc80264224)

[2.3.4. Interfaces & APIs Requirements 49](#_Toc80264225)

[2.3.5. Tenant Requirements 50](#_Toc80264226)

[2.3.6. Operations and LCM 50](#_Toc80264227)

[2.3.7. Assurance Requirements 51](#_Toc80264228)

[2.4. Architecture and OpenStack Recommendations 51](#_Toc80264229)

[2.4.1. General Recommendations 51](#_Toc80264230)

[2.4.2. Infrastructure Recommendations 52](#_Toc80264231)

[2.4.3. VIM Recommendations 54](#_Toc80264232)

[2.4.4. Interfaces and APIs Recommendations 54](#_Toc80264233)

[2.4.5. Tenant Recommendations 54](#_Toc80264234)

[2.4.6. Operations and LCM Recommendations 54](#_Toc80264235)

[2.4.7. Assurance Recommendations 55](#_Toc80264236)

[2.4.8. Security Recommendations 55](#_Toc80264237)

[2.4.8.1. System Hardening (source RM 7.9.1 [1]) 55](#_Toc80264238)

[2.4.8.2. Platform and Access (source RM 7.9.2 [1]) 56](#_Toc80264239)

[2.4.8.3. Confidentiality and Integrity (source RM7.9.3 [1]) 56](#_Toc80264240)

[2.4.8.4. Workload Security (source RM7.9.4 [1]) 56](#_Toc80264241)

[2.4.8.5. Image Security (source RM7.9.5 [1]) 57](#_Toc80264242)

[2.4.8.6. Security LCM (source RM7.9.6 [1]) 57](#_Toc80264243)

[2.4.8.7. Monitoring and Security Audit (source RM7.9.7 [1]) 57](#_Toc80264244)

[2.4.8.8. Open Source Software Security (source RM7.9.8 [1]) 57](#_Toc80264245)

[2.4.8.9. IaaC security (source RM7.9.9 [1]) 58](#_Toc80264246)

[2.4.8.9.1. Secure Design and Architecture Stage 58](#_Toc80264247)

[2.4.8.9.2. Secure Code Stage Requirements 58](#_Toc80264248)

[2.4.8.9.3. Continuous Build, Integration and Testing Stage Requirements 59](#_Toc80264249)

[2.4.8.9.4. Continuous Delivery and Deployment Stage Requirements 60](#_Toc80264250)

[2.4.8.9.5. Runtime Defence and Monitoring Requirements 60](#_Toc80264251)

[2.4.8.10. Compliance with Standards (source RM7.9.10 [1]) 61](#_Toc80264252)

[3. Cloud Infrastructure Architecture - OpenStack 62](#_Toc80264253)

[3.1. Introduction 62](#_Toc80264254)

[3.2. Consumable Infrastructure Resources and Services 64](#_Toc80264255)

[3.2.1. Multi-Tenancy (execution environment) 64](#_Toc80264256)

[3.2.2. Virtual Compute (vCPU and vRAM) 64](#_Toc80264257)

[3.2.3. Virtual Storage 64](#_Toc80264258)

[3.2.4. Virtual Networking Neutron standalone 65](#_Toc80264259)

[3.2.5. Virtual Networking – 3rd party SDN solution 65](#_Toc80264260)

[3.2.5.1. Tungsten Fabric (SDN Controller) 66](#_Toc80264261)

[3.2.6. Acceleration 67](#_Toc80264262)

[3.3. Virtualised Infrastructure Manager (VIM) 67](#_Toc80264263)

[3.3.1. VIM Core services 67](#_Toc80264264)

[3.3.1.1. OpenStack Services Topology 68](#_Toc80264265)

[3.3.1.2. Foundation Services 69](#_Toc80264266)

[3.3.1.3. Cloud Controller Services 70](#_Toc80264267)

[3.3.1.4. Cloud Workload Services 71](#_Toc80264268)

[3.3.2. Tenant Isolation 72](#_Toc80264269)

[3.3.3. Cloud partitioning: Host Aggregates, Availability Zones 72](#_Toc80264270)

[3.3.4. Flavor management 72](#_Toc80264271)

[3.4. Underlying Resources 73](#_Toc80264272)

[3.4.1. Virtualisation 73](#_Toc80264273)

[3.4.2. Physical Infrastructure 73](#_Toc80264274)

[3.4.2.1. Compute 74](#_Toc80264275)

[3.4.2.2. Network 74](#_Toc80264276)

[3.4.2.3. Storage 75](#_Toc80264277)

[3.5. Cloud Topology 75](#_Toc80264278)

[3.5.1. Topology Overview 76](#_Toc80264279)

[3.5.2. Topology Detail 77](#_Toc80264280)

[3.5.2.1. Topology 1 - Local Redundancy 77](#_Toc80264281)

[3.5.2.2. Topology 2 - Regional Redundancy 78](#_Toc80264282)

[3.5.2.3. Topology 3 - Global Redundancy 78](#_Toc80264283)

[4. Cloud Infrastructure + VIM Component Level Architecture 79](#_Toc80264284)

[4.1. Introduction. 79](#_Toc80264285)

[4.2. Underlying Resources 79](#_Toc80264286)

[4.2.1. Virtualisation 79](#_Toc80264287)

[4.2.2. Compute 80](#_Toc80264288)

[4.2.2.1. Cloud Deployment (Foundation/management) Node 80](#_Toc80264289)

[4.2.2.2. OpenStack Control Plane Servers (Control Nodes) 80](#_Toc80264290)

[4.2.2.3. Network nodes 80](#_Toc80264291)

[4.2.2.4. Storage nodes 81](#_Toc80264292)

[4.2.2.5. Compute Nodes 81](#_Toc80264293)

[4.2.2.6. Compute Resource Pooling Considerations 84](#_Toc80264294)

[4.2.2.7. Reservation of Compute Node Cores 84](#_Toc80264295)

[4.2.2.8. Pinned and Unpinned CPUs 85](#_Toc80264296)

[4.2.2.9. Compute node configurations for Profiles and OpenStack Flavors 85](#_Toc80264297)

[4.2.2.9.1. Cloud Infrastructure Hardware Profile 85](#_Toc80264298)

[4.2.2.9.2. CPU Allocation Ratio and CPU Pinning 86](#_Toc80264299)

[4.2.2.9.3. Server Configurations 86](#_Toc80264300)

[4.2.2.9.4. Leaf and Compute Ports for Server Flavors must align 86](#_Toc80264301)

[4.2.2.9.5. Example Host Configurations 87](#_Toc80264302)

[4.2.2.9.6. Using Hosts of a Host Profile type 91](#_Toc80264303)

[4.2.3. Network Fabric 92](#_Toc80264304)

[4.2.3.1. Physical Network Topology 92](#_Toc80264305)

[4.2.3.2. High Level Logical Network Layout 92](#_Toc80264306)

[4.2.3.3. Octavia v2 API conformant Load Balancing 94](#_Toc80264307)

[4.2.3.4. Neutron Extensions 94](#_Toc80264308)

[4.2.3.5. Network quality of service 95](#_Toc80264309)

[4.2.3.6. Integration Interfaces 95](#_Toc80264310)

[4.2.4. Storage Backend 95](#_Toc80264311)

[4.2.4.1. Ceph Storage Cluster 96](#_Toc80264312)

[4.3. Virtualised Infrastructure Manager (VIM) 98](#_Toc80264313)

[4.3.1. VIM Services 98](#_Toc80264314)

[4.3.1.1. Keystone 98](#_Toc80264315)

[4.3.1.2. Glance 98](#_Toc80264316)

[4.3.1.3. Cinder 98](#_Toc80264317)

[4.3.1.4. Swift 98](#_Toc80264318)

[4.3.1.5. Neutron 99](#_Toc80264319)

[4.3.1.5.1. Issues with the standard networking (centralized routing) approach 100](#_Toc80264320)

[4.3.1.5.2. Distributed Virtual Routing (DVR) 101](#_Toc80264321)

[4.3.1.5.3. Software Defined Networking (SDN) 101](#_Toc80264322)

[4.3.1.6. Nova 101](#_Toc80264323)

[4.3.1.7. Ironic 102](#_Toc80264324)

[4.3.1.8. Heat 102](#_Toc80264325)

[4.3.1.9. Horizon 102](#_Toc80264326)

[4.3.1.10. Placement 102](#_Toc80264327)

[4.3.1.11. Barbican 103](#_Toc80264328)

[4.3.2. Containerised OpenStack Services 103](#_Toc80264329)

[4.4. Consumable Infrastructure Resources and Services 104](#_Toc80264330)

[4.4.1. Support for Cloud Infrastructure Profiles and flavors 104](#_Toc80264331)

[4.4.2. Logical segregation and high availability 106](#_Toc80264332)

[4.4.3. Transaction Volume Considerations 107](#_Toc80264333)

[4.5. Cloud Topology and Control Plane Scenarios 107](#_Toc80264334)

[4.5.1. Edge Cloud Topology 109](#_Toc80264335)

[4.5.1.1. Edge Cloud Deployment 110](#_Toc80264336)

[5. Interfaces and APIs 110](#_Toc80264337)

[5.1. Introduction 110](#_Toc80264338)

[5.2. Core OpenStack Services APIs 111](#_Toc80264339)

[5.2.1. Keystone 111](#_Toc80264340)

[5.2.2. Glance 112](#_Toc80264341)

[5.2.3. Cinder 112](#_Toc80264342)

[5.2.4. Swift 112](#_Toc80264343)

[5.2.5. Neutron 113](#_Toc80264344)

[5.2.6. Nova 117](#_Toc80264345)

[5.2.7. Placement 118](#_Toc80264346)

[5.2.8. Heat 119](#_Toc80264347)

[5.3. Consolidated Set of APIs 119](#_Toc80264348)

[5.3.1. OpenStack Interfaces 119](#_Toc80264349)

[5.3.2. Kubernetes Interfaces 120](#_Toc80264350)

[5.3.3. KVM Interfaces 120](#_Toc80264351)

[5.3.3.1. Libvirt Interfaces 120](#_Toc80264352)

[5.3.4. Barbican 120](#_Toc80264353)

[6. Security 120](#_Toc80264354)

[6.1. Introduction 120](#_Toc80264355)

[6.2. Security Requirements 120](#_Toc80264356)

[6.3. Cloud Infrastructure and VIM Security 121](#_Toc80264357)

[6.3.1. System Hardening 121](#_Toc80264358)

[6.3.1.1. Server boot hardening 121](#_Toc80264359)

[6.3.1.2. System Access 121](#_Toc80264360)

[6.3.1.3. Password policy 121](#_Toc80264361)

[6.3.1.4. Function and Software 122](#_Toc80264362)

[6.3.1.5. Patches 122](#_Toc80264363)

[6.3.1.6. Network Protocols 123](#_Toc80264364)

[6.3.1.7. Anti-Virus and Firewall 123](#_Toc80264365)

[6.3.1.8. Vulnerability Detection and Prevention 123](#_Toc80264366)

[6.3.2. Platform Access 123](#_Toc80264367)

[6.3.2.1. Identity Security 123](#_Toc80264368)

[6.3.2.2. Authentication 123](#_Toc80264369)

[6.3.2.2.1. Keystone Tokens 124](#_Toc80264370)

[6.3.2.3. Authorization 124](#_Toc80264371)

[6.3.2.4. RBAC 124](#_Toc80264372)

[6.3.3. Confidentiality and Integrity 126](#_Toc80264373)

[6.3.3.1. Confidentiality and Integrity of communications (sec.ci.001) 126](#_Toc80264374)

[6.3.3.2. Integrity of OpenStack components configuration 127](#_Toc80264375)

[6.3.3.3. Confidentiality and Integrity of tenant data (sec.ci.001) 128](#_Toc80264376)

[6.3.4. Workload Security 128](#_Toc80264377)

[6.3.4.1. SR-IOV and DPDK Considerations 129](#_Toc80264378)

[6.3.5. Image Security 129](#_Toc80264379)

[6.3.6. Security LCM 130](#_Toc80264380)

[6.3.6.1. Provisioning/Deployment 130](#_Toc80264381)

[6.3.6.2. Configuration and management 130](#_Toc80264382)

[6.3.6.3. Platform backup 130](#_Toc80264383)

[6.3.6.4. Security upgrades 130](#_Toc80264384)

[6.3.7. Monitoring and Security Audit 131](#_Toc80264385)

[6.3.7.1. Creating Logs 131](#_Toc80264386)

[6.3.7.2. What to Log / What NOT to Log 131](#_Toc80264387)

[6.3.7.2.1. What to log 131](#_Toc80264388)

[6.3.7.2.2. What NOT to log 132](#_Toc80264389)

[6.3.7.3. Where to Log 132](#_Toc80264390)

[6.3.7.4. Required Fields 132](#_Toc80264391)

[6.3.7.5. Data Retention 132](#_Toc80264392)

[6.3.7.6. Security Logs Time Synchronisation 133](#_Toc80264393)

[7. Operations and Life Cycle Management 133](#_Toc80264394)

[7.1. Introduction 133](#_Toc80264395)

[7.1.1. Procedural versus Declarative code 134](#_Toc80264396)

[7.1.2. Mutable versus Immutable infrastructure 134](#_Toc80264397)

[7.2. Cloud Infrastructure and VIM configuration management 134](#_Toc80264398)

[7.2.1. Provisioning 134](#_Toc80264399)

[7.2.2. Configuration Management 135](#_Toc80264400)

[7.3. Cloud Infrastructure and VIM Maintenance 136](#_Toc80264401)

[7.4. Logging, Monitoring and Analytics 136](#_Toc80264402)

[7.4.1. Logging 136](#_Toc80264403)

[7.4.2. Monitoring 137](#_Toc80264404)

[7.4.3. Alerting 137](#_Toc80264405)

[7.4.4. Logging, Monitoring, and Analytics (LMA) Framework 138](#_Toc80264406)

[8. Gaps, Innovation, and Development 138](#_Toc80264407)

[8.1. Introduction 138](#_Toc80264408)

[8.2. The Gap 139](#_Toc80264409)

[8.2.1. Autoscaling 139](#_Toc80264410)

[8.3. OpenStack Release Gaps 139](#_Toc80264411)

# Introduction

## Overview

This Reference Architecture is focussed on OpenStack as the Virtualised Infrastructure Manager (VIM) chosen based on the criteria laid out in the Reference Model [1]. 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 the operators have existing staff with the right skill sets to support a Cloud Infrastructure (NFVI [3]) deployment into development, test, and production. 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.

## Vision

The OpenStack-based Anuket Reference Architecture will host NFV workloads, primarily VNFs, of interest to the Anuket community. The Reference Architecture document can be used by operators to deploy Anuket conformant infrastructure.

## Use Cases

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

* **Overlay networks**: The overlay functionality design includes OpenStack Networking in OpenvSwitch [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 optimize bandwidth utilization 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**: Network functions is a broad category but encompasses workloads that support the rest of a system’s network. These workloads tend to consist of large amounts of small packets that are very short lived, such as DNS queries or SNMP traps. These messages need to arrive quickly and do not deal with packet loss as there can be a very large volume of them. There are a few extra considerations to take into account for this type of workload and this can change a configuration all the way to 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 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 availability. This workload benefits from having services local to the consumers of the service. 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.

## Anuket OpenStack Reference Release

This Reference Architecture document conforms to theOpenStack Train [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 Train release. For ease, this Reference Architecture document version can be referred to as "RA-1 OSTK Train."

## Principles

OpenStack Reference Architecture must obey to the following set of principles:

* Requirements Principles [1]
* Architectural Principles [1]

### 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.

## Document Organisation

Section 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. Section 3 and 4 cover the Cloud Infrastructure resources and the core OpenStack services, while the APIs are covered in Section 5. Section 6 covers the implementation and enforcement of security capabilities and controls. Life Cycle Management of the Cloud Infrastructure and VIM are covered in Section 7 with stress on Logging, Monitoring and Analytics (LMA), configuration management and some other operational items, please note that Section 7 is not a replacement for the implementation, configuration and operational documentation that accompanies the different OpenStack distributions. Section 8 identifies certain Gaps that currently exist and plans on how to address them. For example, Service Function Chaining support needs to be addressed to realise the full potential and value of SDN and NFV.

## Terminology

General OpenStack terminology definitions can be found in [the](https://github.com/cntt-n/CNTT/blob/master/doc/common/glossary.md) Glossary [7] and specific terms relating to this reference architecture are listed below.

**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.

**Cloud Infrastructure:** A generic term covering **NFVI**, **IaaS** and **CaaS** capabilities - essentially the infrastructure on which a **Workload** can be executed.

*Note:* **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 Configuration:** a set of settings (Key:Value) that are applied/mapped to **Cloud Infrastructure** HW deployment.

**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 Configuration:** a set of settings (Key:Value) that are applied/mapped to **cloud infrastructure** SW deployment.

**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 is developed using Cloud Native Principles including immutable infrastructure, declarative APIs, and a “repeatable deployment process”.

*Note:* This definition is derived from the Cloud Native Thinking for Telecommunications Whitepaper [8], which also includes further detail and examples.

**Compute flavor:** defines the sizing of the virtualised resources (compute, memory, and storage) required to run a workload.

*Note:* used to define the configuration/capacity limit of a virtualised container.

**Compute Node:** An abstract definition of a server.

*Note:* 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.

*Note:* 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.

**Flavor Capability:** The capability of the Cloud Infrastructure Profile, such as CPU Pinning, NUMA or huge pages.

**Flavor Geometry:** Flavor sizing such as number of vCPUs, RAM, disk, etc.

**Fluentd [8]:** 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.

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

**Hugepages:** 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.

*Note:* Can be used to specify VM Instance or Container Instance.

**Kibana:** An open source data visualisation system.

**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.

*Note:* 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.

**Observability:** Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs.

**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 [10].

*Note:* 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).

**Resilience:** Resilience is the ability to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation.

**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.

**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.

**Software Defined Networking (SDN)**

**Tenant:** cloud service users sharing access to a set of physical and virtual resources, ITU [10].

*Note* Tenants represent an independently manageable logical pool of compute, storage and network resources abstracted from physical hardware.

**Tenant Instance:** refers to 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.

*Note* 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.

*Note:* 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 Flavor.

**Virtual Network Function (VNF):** a software implementation of a **Network Function**, capable of running on the **Cloud Infrastructure**.

**VNF**s are built from one or more VNF Components (**VNFC**) and, in most cases, the VNFC is hosted on a single VM or Container.

**Virtual resources:**

**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.

**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**. Most relevant workload categories in the context of the Cloud Infrastructure are:

**Data Plane Workloads:** that perform tasks related to packet handling of the end-to-end communication between applications. These tasks are expected to be very I/O and memory read/write operations intensive.

**Control Plane Workloads:** that perform tasks related to any other communication between NFs that is not directly related to the end-to-end data communication between applications. For example, this category includes session management, routing or authentication.

**Storage Workloads:** that perform tasks related to disk storage (either SSD or HDD or other). Examples range from non-intensive router logging to more intensive database read/write operations.

## Abbreviations

|  |  |
| --- | --- |
| **Abbreviation/Acronym** | **Definition** |
| API | Application Programming Interface |
| DNS | Domain Name System |
| DPDK | Data Plane Development Kit |
| ECMP | Equal Cost Multi-Path routing |
| ETSI | European Telecommunications Standards Institute |
| FPGA | Field Programmable Gate Array |
| GB/TB | GigaByte/TeraByte |
| GPU | Graphics Processing Unit |
| GRE | Generic Routing Encapsulation |
| 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 |
| IaaC (also IaC) | Infrastructure as a Code |
| IaaS | Infrastructure as a Service |
| ICMP | Internet Control Message Protocol |
| IO | Input/Output |
| IOPS | Input/Output per Second |
| 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 |
| OCI | Open Container Initiative |
| OS | Operating System |
| OSTK | OpenStack |
| 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 |
| 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 |
| 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 |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| ToR | Top of Rack |
| TPM | Trusted Platform Module |
| 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 |

## References

|  |  |  |
| --- | --- | --- |
| **Ref** | **Document Title** | **Source** |
|  | GSMA NG.126 | “Cloud Infrastructure Reference Model”. Available at [https://www.gsma.com/newsroom/wp-content/uploads//NG.126-v1.0-2.pdf](https://www.gsma.com/newsroom/wp-content/uploads/NG.126-v1.0-2.pdf). |
|  | OpenStack | “OpenStack”. OpenInfra Foundation. Available at <https://docs.openstack.org>. |
|  | ETSI GS NFV-INF 001 | “Network Functions Virtualisation (NFV); Infrastructure Overview”. Available at <https://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf>. |
|  | OpenStack Use Cases | “OpenStack Use Cases.” OpenInfra Foundation. Available at <https://docs.openstack.org/arch-design/use-cases.html>. |
|  | Open vSwitch | “Open vSwitch (OVS).” Available at https://www.openvswitch.org. |
|  | OpenStack Train | “OpenStack Train.” OpenInfra Foundation. Available at <https://docs.openstack.org/train/projects.html>. |
|  | OpenStack Glossary | “OpenStack Glossary”. Available at <https://docs.openstack.org/image-guide/common/glossary.html>. More pertinent glossary is: “Reference Model Glossary”. [1] Annex B. |
|  | Cloud Native Thinking for Telecommunications Whitepaper | Available at <https://github.com/cncf/telecom-user-group/blob/master/whitepaper/cloud_native_thinking_for_telecommunications.md#1.4> |
|  | Fluentd | Available at <https://www.fluentd.org/> |
|  | ITU-T Y.3500 | Available at <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.3500-201408-I!!PDF-E&type=items> |
|  | IETF RFC 2119 | “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997. Available at <http://www.ietf.org/rfc/rfc2119.txt>. |
|  | CIS Password Policy Guide | https://www.cisecurity.org/white-papers/cis-password-policy-guide/ |
|  | CIS Controls | “Controlled Access Based on the Need to Know.” Available at https://www.cisecurity.org/controls/controlled-access-based-on-the-need-to-know/. |
|  | CVE | “CVE (Common Vulnerabilities and Exposures).” Available at <https://cve.mitre.org/>. |
|  | OpenStack Cores Configuration | “Dedicating host cores to certain workloads (e.g., OpenStack services).” Available at <https://docs.openstack.org/nova/latest/configuration/config.html#compute.cpu_dedicated_set>. |
|  | OpenStack CPU topologies | “Configuring libvirt compute nodes for CPU pinning.” Available at <https://docs.openstack.org/nova/latest/admin/cpu-topologies.html>. |
|  | OpenStack Neutron Plugins | "OpenStack Neutron Plugins." Available at <https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers>. |
|  | OpenStack Resource Tags | "OpenStack Resource Tags." Available at <https://specs.openstack.org/openstack/api-wg/guidelines/tags.html>. |
|  | OpenStack stateful services | “Configuring the stateful services.” Available at <https://docs.openstack.org/ha-guide/control-plane-stateful.html>. |
|  | Senlin | “Senlin.” Available at <https://docs.openstack.org/senlin/train/>. |
|  | OpenStack Smart-NIC support | “OpenStack Future - Specs defined.” Available at <https://specs.openstack.org/openstack/neutron-specs/specs/stein/neutron-ovs-agent-support-baremetal-with-smart-nic.html>. |
|  | NIST SP 800-207 | “Zero Trust Architecture (ZTA)” Described in NIST SP 800-207 |
|  | SBOM | “Software Bill of Materials (SBOM).” Available at <https://www.ntia.gov/SBOM>. |
|  | CIS Controls | “Center for Internet Security CIS Controls.” Available at <https://www.cisecurity.org/>. |
|  | CSA | “CSA Security Guidance for Critical Areas of Focus in Cloud Computing (latest version).” Available at <https://cloudsecurityalliance.org/>. |
|  | OWASP Cheat Sheet Series | “OWASP Cheat Sheet Series (OCSS).” Available at <https://github.com/OWASP/CheatSheetSeries>. |
|  | OWASP Top Ten Security Risks | “OWASP Top Ten Security Risks.” Available at <https://owasp.org/www-project-top-ten/>. |
|  | OWASP Software Maturity Model | “OWASP Software Maturity Model (SAMM).” Available at <https://owaspsamm.org/blog/2019/12/20/version2-community-release/>. |
|  | OWASP Web Security Testing Guide | “OWASP Web Security Testing Guide.” Available at <https://github.com/OWASP/wstg/tree/master/document>. |
|  | ISO/IEC 27001:2013 | “Information security management systems – Requirements” ISO/IEC 27001:2013. Available at <https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en>. |
|  | ISO/IEC 27002:2013 | “Code of practice for information security controls” ISO/IEC 27002:2013 (or latest). Available at <https://www.iso.org/obp/ui/#iso:std:iso-iec:27002:ed-2:v1:en>. |
|  | ISO/IEC 27032:2012 | “Guidelines for Cybersecurity techniques” ISO/IEC 27032:2012 (or latest) Available at <https://www.iso.org/obp/ui/#iso:std:iso-iec:27032:ed-1:v1:en>. |
|  | OpenStack Storage Table | “OpenStack Storage Table”. Available at <https://docs.openstack.org/arch-design/design-storage/design-storage-concepts.html#table-openstack-storage>. |
|  | OpenStack compatible storage backend drivers | “OpenStack compatible storage backend drivers”. Available at <https://docs.openstack.org/cinder/latest/reference/support-matrix.html>. |
|  | Tungsten Fabric | “Tungsten Fabric”. Available at <https://tungsten.io/>. |
|  | OpenStack Nova feature support | “Feature Support Matrix.” Available at <https://docs.openstack.org/nova/latest/user/support-matrix.html>. |
|  | OpenStack Storage | “OpenStack Storage.” (3.4.2.3) Available at <https://docs.openstack.org/arch-design/design-storage.html>. |
|  | OpenStack KVM | “[OpenStack](https://docs.openstack.org/nova/train/admin/configuration/hypervisor-kvm.html) Configuration.” Available at <https://docs.openstack.org/nova/train/admin/configuration/hypervisor-kvm.html>. |
|  | OpenStack Hardening | "Hardening the virtualization layers." Available at https://docs.openstack.org/security-guide/compute/hardening-the-virtualization-layers.html. |
|  | OpenStack Flavors | https://docs.openstack.org/nova/latest/user/flavors.html |
|  | DPDK release notes | “DPDK release notes.” Available at <http://doc.dpdk.org/guides/rel_notes/>. |
|  | DPDK performance reports | “DPDK performance reports.” Available at http://core.dpdk.org/perf-reports/. |
|  | Octavia | “Octavia.” Available at <https://docs.openstack.org/octavia/latest/reference/introduction.html>. |
|  | FwaaS | “FwaaS (Firewall as a Service).” Available at https://docs.openstack.org/neutron/train/admin/fwaas.html. |
|  | LbaaS | “LbaaS (Load Balancer as a Service).” Available at https://governance.openstack.org/tc/reference/projects/octavia.html. |
|  | VPNaaS | “VPNaaS (VPN as a Service).” Available at https://opendev.org/openstack/neutron-vpnaas/. |
|  | Neutron plugins | “Neutron plugins.” Available at https://wiki.openstack.org/wiki/Neutron#Plugins. |
|  | Neutron plugin common methods | “Neutron plugin common methods.” Available at https://docs.openstack.org/neutron/train/contributor/internals/api\_extensions.html. |
|  | OpenStack Networking API extensions | “List Extensions API.” Available at https://docs.openstack.org/api-ref/network/v2/#list-extensions. |
|  | OpenStack networking API | “Extension details API.” Available at <https://docs.openstack.org/api-ref/network/v2/#show-extension-details>. |
|  | OpenStack ML2 | “OpenStack ML2 documentation.” Available at https://wiki.openstack.org/wiki/Neutron/ML2. |
|  | Cinder Support Matrix | “Cinder Support Matrix.” Available at <https://docs.openstack.org/cinder/latest/reference/support-matrix.html>. |
|  | Cinder Drivers | “Cinder Drivers.” Available at  [Available Drivers](https://docs.openstack.org/cinder/latest/drivers.html) <https://docs.openstack.org/cinder/latest/drivers.html>. |
|  | Cinder Configuration | “Cinder Configuration.” Available at https://docs.openstack.org/cinder/latest/configuration/index.html. |
|  | Cinder Administration | “Cinder Administration.” Available at https://docs.openstack.org/cinder/latest/admin/index.html. |
|  | Ceph | “Ceph.” Available at https://ceph.io/. |
|  | DVR | “Distributed Virtual Routing (DVR).” Available at <https://docs.openstack.org/liberty/networking-guide/scenario-dvr-ovs.html>. |
|  | DVR with VRRP | “DVR with VRRP.” Available at <https://docs.openstack.org/neutron/train/admin/config-dvr-ha-snat.html>. |
|  | Placement service | “Placement service.” Available at <https://docs.openstack.org/placement/train/index.html>, and https://docs.openstack.org/placement/latest/user/index.html. |
|  | Placement [Provider Trees](https://docs.openstack.org/placement/latest/user/provider-tree.html) | “Placement [Provider Trees](https://docs.openstack.org/placement/latest/user/provider-tree.html).” Available at <https://docs.openstack.org/placement/latest/user/provider-tree.html>. |
|  | Barbican | “Barbican.” Available at <https://docs.openstack.org/barbican/train/>. |
|  | Open Glossary of Edge Computing | “Open Glossary of Edge Computing.” Available at <https://github.com/State-of-the-Edge/glossary/blob/master/edge-glossary.md>. |
|  | Edge computing whitepaper | “Edge computing whitepaper.” Available at <https://www.openstack.org/use-cases/edge-computing/edge-computing-next-steps-in-architecture-design-and-testing/>. |
|  | OpenStack Reference Deployment Architecture | “OpenStack Reference Deployment Architecture.” Available at <https://fuel-ccp.readthedocs.io/en/latest/design/ref_arch_100_nodes.html#services-placement-summary>. |
|  | Airship | “Airship.” Available at https://docs.airshipit.org/. |
|  | Starling-X | “Starling-X.” Available at <https://www.starlingx.io/>. |
|  | Triple-O | “Triple-O.” Available at <https://wiki.openstack.org/wiki/TripleO>. |
|  | OpenStack Compute API Guide | https://docs.openstack.org/api-guide/compute/microversions.html |
|  | OpenStack API Reference | https://docs.openstack.org/api-ref/ |
|  | Identity API v3 extensions | <https://docs.openstack.org/api-ref/identity/v3-ext/> |
|  | Security compliance and PCI-DSS | <https://docs.openstack.org/keystone/train/admin/configuration.html#security-compliance-and-pci-dss> |
|  | Image Service Versions | <https://docs.openstack.org/api-ref/image/versions/index.html#version-history> |
|  | Cinder REST API Version | <https://docs.openstack.org/cinder/latest/contributor/api_microversion_history.html> |
|  | Swift Discoverability | <https://docs.openstack.org/swift/latest/api/discoverability.html> |
|  | Networking API v2 | <https://docs.openstack.org/api-ref/network/v2/> |
|  | Nova REST API Version | <https://docs.openstack.org/nova/latest/reference/api-microversion-history.html> |
|  | Placement REST API Version | <https://docs.openstack.org/placement/latest/placement-api-microversion-history.htHml> |
|  | Heat Orchestration Template version | <https://docs.openstack.org/heat/latest/template_guide/hot_spec.html> |
|  | Heat Orchestration Template specification | <https://docs.openstack.org/heat/latest/template_guide/hot_spec.html#rocky> |
|  | Kubernetes API | https://kubernetes.io/docs/concepts/overview/kubernetes-api/ |
|  | KVM API Documentation | https://www.kernel.org/doc/Documentation/virtual/kvm/api.txt |
|  | Reference Manual for libvirt | https://libvirt.org/html/index.html |
|  | Barbican | <https://docs.openstack.org/barbican/latest/api/> |
|  | OpenStack Security Guide | <https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html> |
|  | NIST Vulnerability Metrics | https://nvd.nist.gov/vuln-metrics/cvss |
|  | OpenStack Security Guide- Identity Service | https://docs.openstack.org/security-guide/identity.html |
|  | OpenStack Keystone- Default Roles | https://docs.openstack.org/keystone/latest/admin/service-api-protection.html |
|  | OpenStack Secure Communications | Introduction to TLS and SSL — Security Guide documentation (openstack.org),  https://docs.openstack.org/security-guide/secure-communication/introduction-to-ssl-and-tls.html |
|  | CIS-CAT | Center for Internet security- Configuration Assessment Tool. Available at https://www.cisecurity.org/cybersecurity-tools/cis-cat-pro/. |
|  | CIS Benchmarks | Center for Internet security Benchmarks. Available at <https://www.cisecurity.org/cis-benchmarks/>. |
|  | Glance image signing feature | https://docs.openstack.org/glance/pike/user/signature.html |
|  | SR-IOV Passthrough For Networking | https://wiki.openstack.org/wiki/SR-IOV-Passthrough-For-Networking |
|  | OpenStack Security for Instances | <https://docs.openstack.org/security-guide/instance-management/security-services-for-instances.html#trusted-images/> |
|  | OpenStack Virtual Machine Image Guide | https://docs.openstack.org/image-guide/ |
|  | OpenStack Operations Guide | https://docs.openstack.org/operations-guide/ops-user-facing-operations.html#adding-signed-images |
|  | ETSI GS NFV-SOL 004 V2.3.1 | “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification”. Available at https://www.etsi.org/deliver/etsi\_gs/NFV-SOL/001\_099/004/02.03.01\_60/gs\_nfv-sol004v020301p.pdf. |
|  | ETSI GS NFV-SEC 021 V2.6.1 | “Network Functions Virtualisation (NFV) Release 2; Security; VNF Package Security Specification”. Available at https://www.etsi.org/deliver/etsi\_gs/NFV-SEC/001\_099/021/02.06.01\_60/gs\_nfv-sec021v020601p.pdf. |
|  | TripleO Deployment Guide | https://docs.openstack.org/project-deploy-guide/tripleo-docs/latest/index.html |
|  | Reference Implementation | https://cntt.readthedocs.io/en/latest/ref\_impl/cntt-ri/ |
|  | Airship Treasuremap | <https://readthedocs.org/projects/airship-treasuremap/downloads/pdf/latest/> |
|  | OpenStack Autoscaling with Heat | https://docs.openstack.org/senlin/latest/scenarios/autoscaling\_heat.html |
|  | OpenStack Releases | https://releases.openstack.org/ |
|  | OSSN-0075 | wiki.openstack.org/wiki/OSSN/OSSN-0075 |
|  | OpenStack- Support Pre-Upgrade Checks | https://governance.openstack.org/tc/goals/selected/stein/upgrade-checkers.html |

## 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 RFC2119 [11].

# Architecture Requirements

## Introduction

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

## Reference Model Requirements

The tables below contain the requirements from the Reference Model 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 Storage Extensions (S extension)
* Those relating to Network Acceleration Extensions (A extension)
* Those relating to Cloud Infrastructure Management

Note: "(if offered)" used in the Reference Model has been replaced with "Optional" in the tables below so as to align with RFC2119 [8].

### Cloud Infrastructure Software Profile Requirements for Compute (sourceRM 5.2 [1])

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 VM by the Cloud Infrastructure | At least 16 | At least 16 |  |
| e.cap.002 | Max memory that can be assigned to a single VM by the Cloud Infrastructure | at least 32 GB | at least 32 GB |  |
| e.cap.003 | Max storage that can be assigned to a single VM by the Cloud Infrastructure | at least 320 GB | at least 320 GB |  |
| e.cap.004 | Max number of connection points that can be assigned to a single VM by the Cloud Infrastructure | 6 | 6 |  |
| e.cap.005 | Max storage that can be attached / mounted to VM by the Cloud Infrastructure | Up to 16TB1 | Up to 16TB1 |  |
| e.cap.006/ infra.com.cfg.003 | CPU pinning support | Not required | Must support |  |
| e.cap.007/ infra.com.cfg.002 | NUMA support | Not required | Must support |  |
| e.cap.018/ infra.com.cfg.005 | Simultaneous Multithreading (SMT) enabled | Not required | Must support |  |
| i.cap.018/ infra.com.cfg.004 | Huge Pages configured | Not required | Must support |  |

Table 1: Reference Model Requirements: Cloud Infrastructure Software Profile Capabilities

**1** Defined in the “.bronze” configuration inRM section 4.2.6 Storage Extensions [1]

#### Cloud Infrastructure Software Profile Extensions Requirements for Compute

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reference** | **Description** | **Profile Extensions** | **Profile Extra-Specs** | **Specification Reference** |
| e.cap.008/ infra.com.acc.cfg.001 | IPSec Acceleration using the virtio-ipsec interface | Compute Intensive GPU |  |  |
| e.cap.010/ infra.com.acc.cfg.002 | Transcoding Acceleration | Compute Intensive GPU | Video Transcoding |  |
| e.cap.011/ infra.com.acc.cfg.003 | Programmable Acceleration | Firmware-programmable adapter | Accelerator |  |
| e.cap.012 | Enhanced Cache Management: L=Lean; E=Equal; X=eXpanded | E | E |  |
| e.cap.014/ infra.com.acc.cfg.004 | Hardware coprocessor support (GPU/NPU) | Compute Intensive GPU |  |  |
| e.cap.016/ infra.com.acc.cfg.005 | FPGA/other Acceleration H/W | Firmware-programmable adapter |  |  |

Table 2: Cloud Infrastructure Software Profile Extensions Requirements for Compute

### Cloud Infrastructure Software Profile Requirements for Networking (sourceRM 5.2.3 [1])

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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |
| 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* |  |
| infra.net.cfg.003 | Network Address Translation | Must support | Must support |  |
| infra.net.cfg.004 | Security Groups | Must support | Must support |  |
| infra.net.cfg.005 | SFC support | Not required | Must support |  |
| infra.net.cfg.006 | Traffic patterns symmetry | Must support | Must support |  |

Table 3**:** Reference Model Requirements - Virtual Networking

The required number of connection points to a VM is described in e.cap.004 inTable 1. The table below specifies the required bandwidth of those connection points.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |
| n10, n20, n30, n40, n50, n60 | 10, 20, 30, 40, 50, 60 Gbps | Must support | Must support |  |
| n25, n50, n75, n100, n125, n150 | 25, 50, 75, 100, 125, 150 Gbps | Optional | Must support |  |
| n50, n100, n150, n200, n250, n300 | 50, 100, 150, 200, 250, 300 Gbps | Optional | Must support |  |
| n100, n200, n300, n400, n500, n600 | 100, 200, 300, 400, 500, 600 Gbps | Optional | Must support |  |

Table 4**:** Reference Model Requirements - Network Interface Specifications

#### 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 |  |
| e.cap.019/ infra.net.acc.cfg.001 | vSwitch optimisation (DPDK) | N | Y |  |
| e.cap.015/ infra.net.acc.cfg.002 | SmartNIC (for HW Offload) | N | Optional |  |
| e.cap.009/ infra.net.acc.cfg.003 | Crypto acceleration | N | Optional |  |
| infra.net.acc.cfg.004 | Crypto Acceleration Interface | N | Optional |  |

Table 5: Cloud Infrastructure Software Profile Extensions Requirements for Networking

### Cloud Infrastructure Software Profile Requirements for Storage (sourceRM 5.2 [1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reference** | **Description** | **Requirement for Basic Profile** | **Requirement for High Performance Profile** | **Specification Reference** |
| infra.stg.cfg.002 | Storage Block | Must support | Must support |  |
| infra.stg.cfg.003 | Storage with replication | Not required | Must support |  |
| infra.stg.cfg.004 | Storage with encryption | Must support | Must support |  |
| infra.stg.acc.cfg.001 | Storage IOPS oriented | Not required | Must support |  |
| infra.stg.acc.cfg.002 | Storage capacity oriented | Not required | Not required |  |

Table 6**:** Reference Model Requirements - Cloud Infrastructure Software Profile Requirements for Storage

#### 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 |  |  |
| infra.stg.acc.cfg.002 | Storage capacity oriented | High Capacity |  |  |

Table 7 Reference Model Requirements - Cloud Infrastructure Software Profile Extensions Requirements for Storage

### Cloud Infrastructure Hardware Profile Requirements (sourceRM 5.4 [1])

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |
| infra.hw.cpu.cfg.002 | Minimum number of Cores per CPU | 20 | 20 |  |
| infra.hw.cpu.cfg.003 | NUMA | Not required | Must support |  |
| infra.hw.cpu.cfg.004 | Simultaneous Multithreading/Symmetric Multiprocessing (SMT/SMP) | Must support | Must support |  |
| infra.hw.stg.hdd.cfg.001 | Local Storage HDD | *No requirement specified* | *No requirement specified* |  |
| infra.hw.stg.ssd.cfg.002 | Local Storage SSD | Should support | Should support |  |
| infra.hw.nic.cfg.001 | Total Number of NIC Ports available in the host | 4 | 4 |  |
| infra.hw.nic.cfg.002 | Port speed specified in Gbps (minimum values) | 10 | 25 |  |
| infra.hw.pci.cfg.001 | Number of PCIe slots available in the host | 8 | 8 |  |
| infra.hw.pci.cfg.002 | PCIe speed | Gen 3 | Gen 3 |  |
| infra.hw.pci.cfg.003 | PCIe Lanes | 8 | 8 |  |
| infra.hw.nac.cfg.003 | Compression | *No requirement specified* | *No requirement specified* |  |

Table 8: Reference Model Requirements - Cloud Infrastructure Hardware Profile Requirements

#### Cloud Infrastructure Hardware Profile-Extensions Requirements (sourceRM 5.4 [1])

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reference** | **Description** | **Requirement for Basic Profile** | **Requirement for High Performance Profile** | **Specification Reference** |
| e.cap.014/ infra.hw.cac.cfg.001 | GPU | N | Optional |  |
| e.cap.016/ infra.hw.cac.cfg.002 | FPGA/other Acceleration H/W | N | Optional |  |
| e.cap.009/ infra.hw.nac.cfg.001 | Crypto Acceleration | N | Optional |  |
| e.cap.015/ infra.hw.nac.cfg.002 | SmartNIC | N | Optional |  |
| infra.hw.nac.cfg.003 | Compression | Optional | Optional |  |
| e.cap.013/ infra.hw.nac.cfg.004 | SR-IOV over PCI-PT | N | Yes |  |

Table 9: Reference Model Requirements - Cloud Infrastructure Hardware Profile-Extensions Requirements

### Cloud Infrastructure Management Requirements (sourceRM 4.1.5 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference** | **Description** | **Requirement (common to all Profiles)** | **Specification Reference** |
| e.man.001 | Capability to allocate virtual compute resources to a workload | Must support |  |
| e.man.002 | Capability to allocate virtual storage resources to a workload | Must support |  |
| e.man.003 | Capability to allocate virtual networking resources to a workload | Must support |  |
| e.man.004 | Capability to isolate resources between tenants | Must support |  |
| e.man.005 | Capability to manage workload software images | Must support |  |
| e.man.006 | Capability to provide information related to allocated virtualised resources per tenant | Must support |  |
| e.man.007 | Capability to notify state changes of allocated resources | Must support |  |
| e.man.008 | Capability to collect and expose performance information on virtualised resources allocated | Must support |  |
| e.man.009 | Capability to collect and notify fault information on virtualised resources | Must support |  |

Table 10**:** Reference Model Requirements - Cloud Infrastructure Management Requirements

### Cloud Infrastructure Security Requirements

#### System Hardening (sourceRM 7.9.1 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.gen.001 | Hardening | The Platform **must** maintain the specified configuration. | Security LCM, [Cloud Infrastructure and VIM configuration management](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter07.md#72-cloud-infrastructure-and-vim-configuration-management) |
| sec.gen.002 | Hardening | All systems part of Cloud Infrastructure **must** support password hardening as defined inCIS Password Policy Guide [12]. | [Password policy](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6313-password-policy) |
| sec.gen.003 | Hardening | All servers part of Cloud Infrastructure **must** support a root of trust and secure boot. | [Server boot hardening](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6311-server-boot-hardening) |
| 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). | [Function and Software](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6314-function-and-software) |
| sec.gen.005 | Hardening | The Platform **must** support Operating System level access control. | [System Access](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6312-system-access) |
| sec.gen.006 | Hardening | The Platform **must** support Secure logging. Logging with root account must be prohibited when root privileges are not required. | [System Access](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6312-system-access) |
| sec.gen.007 | Hardening | All servers part of Cloud Infrastructure **must** be Time synchronized with authenticated Time service. | [Security Logs Time Synchronisation](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6376-security-logs-time-synchronisation) |
| sec.gen.008 | Hardening | All servers part of Cloud Infrastructure **must** be regularly updated to address security vulnerabilities. | [Patches](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6315-patches), [Security LCM](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#636-security-lcm) |
| sec.gen.009 | Hardening | The Platform **must** support Software integrity protection and verification. | [Integrity of OpenStack components configuration](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration), [Image Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#635-image-security) |
| 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 ([13]). | [Confidentiality and Integrity of tenant data (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6333-confidentiality-and-integrity-of-tenant-data-secmon012-and-secmon013) |
| sec.gen.012 | Hardening | The Operator **must** ensure that only authorized actors have physical access to the underlying infrastructure. | This requirement’s verification goes beyond Anuket testing scope |
| sec.gen.013 | Hardening | The Platform **must** ensure that only authorized actors have logical access to the underlying infrastructure. | [System Access](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6312-system-access) |
| 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. | [Security LCM](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#636-security-lcm) |

Table 11**:** Reference Model Requirements - System Hardening Requirements

#### Platform and Access (sourceRM 7.9.2 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.sys.001 | Access | The Platform **must** support authenticated and secure access to API, GUI and command line interfaces | [RBAC](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6324-rbac) |
| sec.sys.002 | Access | The Platform **must** support Traffic Filtering for workloads (for example, Fire Wall). | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |
| sec.sys.003 | Access | The Platform **must** support Secure and encrypted communications, and confidentiality and integrity of network traffic. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6331-confidentiality-and-integrity-of-communications) |
| sec.sys.004 | Access | The Cloud Infrastructure **must** support authentication, integrity and confidentiality on all network channels. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6331-confidentiality-and-integrity-of-communications) |
| sec.sys.005 | Access | The Cloud Infrastructure **must** segregate the underlay and overlay networks. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6331-confidentiality-and-integrity-of-communications) |
| sec.sys.006 | Access | The Cloud Infrastructure **must** be able to utilize the Cloud Infrastructure Manager identity lifecycle management capabilities. | [Identity Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6321-identity-security) |
| 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). | [RBAC](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6324-rbac) |
| 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.) | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |
| sec.sys.009 | Access | The Platform **must** support creation of Trust Relationships between trust domains. These may be uni-directional relationships where the trusting domain trusts another domain (the “trusted domain”) to authenticate users for them or to allow access to its resources from the trusted domain. In a bidirectional relationship, both domains are “trusting” and “trusted”. |  |
| 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. |  |
| 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). | [System Access](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6312-system-access) |
| 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) |  |
| sec.sys.013 | Access | The Platform **must** generate secrets dynamically as and when needed. |  |
| sec.sys.015 | Access | The Platform **must not** contain back door entries (unpublished access points, APIs, etc.). |  |
| 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 | [Security LCM](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#636-security-lcm) |
| 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. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6331-confidentiality-and-integrity-of-communications) |
| sec.sys.018 | Access | The Platform **must** provide the capability of allowing certificate renewal and revocation. |  |
| 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). |  |

Table 12**:** Reference Model Requirements - Platform and Access Requirements

#### Confidentiality and Integrity (sourceRM7.9.3 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.ci.001 | Confidentiality/Integrity | The Platform **must** support Confidentiality and Integrity of data at rest and in transit. | [Confidentiality and Integrity](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#633-confidentiality-and-integrity) |
| sec.ci.003 | Confidentiality/Integrity | The Platform **must** support Confidentiality and Integrity of data related metadata. |  |
| sec.ci.004 | Confidentiality | The Platform **must** support Confidentiality of processes and restrict information sharing with only the process owner (e.g., tenant). |  |
| 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). |  |
| sec.ci.006 | Confidentiality/Integrity | The Platform **must** support Confidentiality and Integrity of workload resource utilization (RAM, CPU, Storage, Network I/O, cache, hardware offload) and restrict information sharing with only the workload owner (e.g., tenant). |  |
| sec.ci.007 | Confidentiality/Integrity | The Platform **must not** allow Memory Inspection by any actor other than the authorized 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. |  |
| sec.ci.008 | Confidentiality | The Cloud Infrastructure **must** support tenant networks segregation. | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |

Table 13**:** Reference Model Requirements - Confidentiality and Integrity Requirements

#### Workload Security (sourceRM7.9.4 [1]

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.wl.001 | Workload | The Platform **must** support Workload placement policy. | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |
| 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). |  |
| sec.wl.003 | Workload | The Platform **must** support secure provisioning of Workloads. | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |
| sec.wl.004 | Workload | The Platform **must** support Location assertion (for mandated in-country or location requirements). | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |
| sec.wl.005 | Workload | The Platform **must** support the separation of production and non-production Workloads. | This requirement’s verification goes beyond Anuket testing scope |
| sec.wl.006 | Workload | The Platform **must** support the separation of Workloads based on their categorisation (for example, payment card information, healthcare, etc.) | [Workload Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#634-workload-security) |

Table 14**:** Reference Model Requirements - Workload Security Requirements

#### Image Security (sourceRM7.9.5 [1]

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.img.001 | Image | Images from untrusted sources **must not** be used. | [Image Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#635-image-security) |
| sec.img.002 | Image | Images **must** be scanned to be maintained free from known vulnerabilities. | [Image Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#635-image-security) |
| sec.img.003 | Image | Images **must not** be configured to run with privileges higher than the privileges of the actor authorized to run them. |  |
| sec.img.004 | Image | Images **must** only be accessible to authorized actors. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration) |
| sec.img.005 | Image | Image Registries **must** only be accessible to authorized actors. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration) |
| sec.img.006 | Image | Image Registries **must** only be accessible over networks that enforce authentication, integrity and confidentiality. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration) |
| sec.img.007 | Image | Image registries **must** be clear of vulnerable and out of date versions. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration), [Image Security](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#635-image-security) |

Table 15**:** Reference Model Requirements - Image Security Requirements

#### Security LCM (sourceRM7.9.6 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| 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. | [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| 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. | [Security LCM](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#636-security-lcm) |
| sec.lcm.003 | LCM | The Cloud Operator **must** implement and strictly follow change management processes for Cloud Infrastructure, Cloud Infrastructure Manager and other components of the cloud, and Platform change control on hardware. | [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| sec.lcm.005 | LCM | Platform **must** provide logs and these logs must be monitored for anomalous behaviour. | [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| sec.lcm.006 | LCM | The Platform **must** verify the integrity of all Resource management requests. | [Confidentiality and Integrity of tenant data (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6333-confidentiality-and-integrity-of-tenant-data-secmon012-and-secmon013) |
| sec.lcm.007 | LCM | The Platform **must** be able to update newly instantiated, suspended, hibernated, migrated and restarted images with current time information. |  |
| sec.lcm.008 | LCM | The Platform **must** be able to update newly instantiated, suspended, hibernated, migrated and restarted images with relevant DNS information. |  |
| 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. |  |
| 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). |  |
| 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. | [Patches](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6315-patches) |
| sec.lcm.012 | LCM | The Platform **must** log any access privilege escalation. | [What to Log / What NOT to Log](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6372-what-to-log--what-not-to-log) |

Table 16**:** Reference Model Requirements - Security LCM Requirements

#### Monitoring and Security Audit (sourceRM7.9.7 [1]

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.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| 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. | [Creating Logs](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6371-creating-logs), [Required Fields](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6374-required-fields) |
| sec.mon.002 | Monitoring | Security logs **must** be time synchronised. | [Security Logs Time Synchronisation](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6376-security-logs-time-synchronisation) |
| sec.mon.003 | Monitoring | The Platform **must** log all changes to time server source, time, date and time zones. | [Security Logs Time Synchronisation](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6376-security-logs-time-synchronisation) |
| sec.mon.004 | Audit | The Platform **must** secure and protect Audit logs (containing sensitive information) both in-transit and at rest. | [Security LCM](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#636-security-lcm) |
| 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 actions accordingly | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration), [What to Log / What NOT to Log](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6372-what-to-log--what-not-to-log) |
| sec.mon.006 | Monitoring/ Audit | The Platform **must** Monitor and Audit operations by authorized account access after login to detect malicious operational activity and take corrective actions. | [Integrity of OpenStack components configuration](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration), [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| sec.mon.007 | Monitoring/ Audit | The Platform **must** Monitor and Audit security parameter configurations for compliance with defined security policies. | [Integrity of OpenStack components configuration](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration) |
| sec.mon.008 | Monitoring/ Audit | The Platform **must** Monitor and Audit externally exposed interfaces for illegal access (attacks) and take corrective security hardening measures. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6331-confidentiality-and-integrity-of-communications) |
| 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. | [Confidentiality and Integrity of communications (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6332-integrity-of-openstack-components-configuration), [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| sec.mon.010 | Monitoring/ Audit | The Platform **must** Monitor and Audit running processes to detect unexpected or unauthorized processes and take corrective actions accordingly. | [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| 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. | [Creating Logs](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6371-creating-logs) |
| sec.mon.012 | Monitoring/ Audit | The Platform **must** Monitor and Audit Traffic patterns and volumes to prevent malware download attempts. | [Confidentiality and Integrity of tenant data (sec.ci.001)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6333-confidentiality-and-integrity-of-tenant-data-secmon012-and-secmon013) |
| 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). |  |
| 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. | [Monitoring and Security Audit](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#637-monitoring-and-security-audit) |
| sec.mon.017 | Audit | The Platform **must** audit systems for any missing security patches and take appropriate actions. | [Patches](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6315-patches) |
| sec.mon.018 | Monitoring | The Platform, starting from initialization, **must** collect and analyze logs to identify security events, and store these events in an external system. | [Where to Log](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6373-where-to-log) |
| sec.mon.019 | Monitoring | The Platform’s components **must not** include an authentication credential, e.g., password, in any logs, even if encrypted. | [What to Log / What NOT to Log](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6372-what-to-log--what-not-to-log) |
| sec.mon.020 | Monitoring/ Audit | The Platform’s logging system **must** support the storage of security audit logs for a configurable period of time. | [Data Retention](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6375-data-retention) |
| 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. | [Where to Log](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter06.md#6373-where-to-log) |

Table 17**:** Reference Model Requirements - Monitoring and Security Audit Requirements

#### Open Source Software (sourceRM7.9.8 [1]

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.oss.001 | Software | Open source code **must** be inspected by tools with various capabilities for static and dynamic code analysis. |  |
| 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 [14] |  |
| sec.oss.003 | Software | High severity rated vulnerabilities **must** be fixed. Refer to the CVSS (Common Vulnerability Scoring System) to know a vulnerability score. |  |
| sec.oss.004 | Software | A dedicated internal isolated repository separated from the production environment **must** be used to store vetted open source content. |  |

Table 18**:** Reference Model Requirements - Open Source Software Security Requirements

#### IaaC security (sourceRM7.9.9 [1])

##### Secure Code Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.code.001 | IaaC | SAST -Static Application Security Testing **must** be applied during Secure Coding stage triggered by Pull, Clone or Comment trigger. Security testing that analyses application source code for software vulnerabilities and gaps against best practices. Example: open source OWASP range of tools. |  |

Table 19**:** Reference Model Requirements - IaaC Security Requirements, Secure Code Stage

##### Continuous Build, Integration and Testing Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.bld.003 | IaaC | Container and 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 container registry may trigger a vulnerability scan before the image becomes available in the registry. |  |

Table 20**:** Reference Model Requirements - IaaC Security Requirements, Continuous Build, Integration and Testing Stage

##### Continuous Delivery and Deployment Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.del.001 | IaaC | 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. |  |
| sec.del.002 | IaaC | Code Signing **must** be applied during the Continuous Delivery and Deployment stage triggered by Publish to Artifact and Image Repository trigger. Code Signing provides authentication to assure that downloaded files are form the publisher named on the certificate. |  |
| sec.del.004 | IaaC | 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). |  |

Table 21**:** Reference Model Requirements - IaaC Security Requirements, Continuous Delivery and Deployment Stage

##### Runtime Defence and Monitoring Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| sec.run.001 | IaaC | 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. |  |

Table 22**:** Reference Model Requirements - IaaC Security Requirements, Runtime Defence and Monitoring Stage

#### Compliance with Standards (sourceRM7.9.10 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| 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); |  |

Table 23**:** Reference Model Requirements - Cloud Infrastructure Security Requirements

## Architecture and OpenStack Requirements

"Architecture" in this section refers to Cloud infrastructure (referred to as NFVI by ETSI) + VIM (as specified in Reference Model [1] section 3).

### General Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| gen.ost.01 | Open source | The Architecture **must** use OpenStack APIs. | [Consolidated Set of APIs](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#5.3) |
| gen.ost.02 | Open source | The Architecture **must** support dynamic request and configuration of virtual resources (compute, network, storage) through OpenStack APIs. | Consolidated Set of APIs |
| gen.rsl.01 | Resiliency | The Architecture **must** support resilient OpenStack components that are required for the continued availability of running workloads. |  |
| gen.avl.01 | Availability | The Architecture **must** provide High Availability for OpenStack components. | [Underlying Resources](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#42-underlying-resources) |

Table 24**:** General Requirements

### Infrastructure Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| inf.com.01 | Compute | The Architecture **must** provide compute resources for VM instances. | [Cloud Workload Services](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#3314-cloud-workload-services) |
| inf.com.04 | Compute | The Architecture **must** be able to support multiple CPU type options to support various infrastructure profiles (Basic and High Performance). | [Support for Cloud Infrastructure Profiles and flavors](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4.4.1) |
| inf.com.05 | Compute | The Architecture **must** support Hardware Platforms with NUMA capabilities. | [Support for Cloud Infrastructure Profiles and flavors](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4.4.1) |
| inf.com.06 | Compute | The Architecture **must** support CPU Pinning of the vCPUs of a VM instance. | [Support for Cloud Infrastructure Profiles and flavors](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4.4.1) |
| inf.com.07 | Compute | The Architecture **must** support different hardware configurations to support various infrastructure profiles (Basic and High Performance). | [Cloud partitioning: Host Aggregates, Availability Zones](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#333-host-aggregates-providing-resource-pooling) |
| inf.com.08 | Compute | The Architecture **must** support allocating a 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” [15]. Please see example,  “Configuring libvirt compute nodes for CPU pinning” [16]. |
| 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. | Achieved through configuring the "cpu\_dedicated\_set" and "cpu\_shared\_set" parameters in nova.conf correctly. |
| inf.stg.01 | Storage | The Architecture **must** provide remote (not directly attached to the host) Block storage for VM Instances. | [Storage](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#3423-storage) |
| inf.stg.02 | Storage | The Architecture **must** provide Object storage for VM Instances. Operators **may** choose not to implement Object Storage but must be cognizant of the risk of "Compliant VNFs" failing in their environment. | OpenStack Swift Service ([Swift](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4314-swift)) |
| inf.ntw.01 | Network | The Architecture **must** provide virtual network interfaces to VM instances. | [Neutron](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#525-neutron) |
| inf.ntw.02 | Network | The Architecture **must** include capabilities for integrating SDN controllers to support provisioning of network services, from the OpenStack Neutron service, such as networking of VTEPs to the Border Edge based VRFs. | [Virtual Networking – 3rd party SDN solution](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#325-virtual-networking--3rd-party-sdn-solution) |
| inf.ntw.03 | Network | The Architecture **must** support low latency and high throughput traffic needs. | [Network Fabric](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#423-network-fabric) |
| inf.ntw.05 | Network | The Architecture **must** allow for East/West tenant traffic within the cloud (via tunnelled encapsulation overlay such as VXLAN or Geneve). | [Network Fabric](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#423-network-fabric) |
| inf.ntw.07 | Network | The Architecture **must** support networkresiliency [Terminology]. | [Network](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#3422-network) |
| inf.ntw.10 | Network | The Cloud Infrastructure Network Fabric **must** be capable of enabling highly available (Five 9’s or better) Cloud Infrastructure. | [Network](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#3422-network) |
| inf.ntw.15 | Network | The Architecture **must** support multiple networking options for Cloud Infrastructure to support various infrastructure profiles (Basic and High Performance). | "Neutron ML2-plugin Integration" [Neutron Extensions] andOpenStack Neutron Plugins [17]. |
| inf.ntw.16 | Network | The Architecture **must** support dual stack IPv4 and IPv6 for tenant networks and workloads. |  |

Table 25**:** Infrastructure Requirements

### VIM Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| vim.01 | General | The Architecture **must** allow infrastructure resource sharing. | [Consumable Infrastructure Resources and Services](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#32-consumable-infrastructure-resources-and-services) |
| vim.03 | General | The Architecture **must** allow VIM to discover and manage Cloud Infrastructure resources. | [Placement](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#527-placement) |
| vim.05 | General | The Architecture **must** include image repository management. | [Glance](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4312-glance) |
| vim.07 | General | The Architecture **must** support multi-tenancy. | [Multi-Tenancy (execution environment)](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#321-multi-tenancy-execution-environment) |
| vim.08 | General | The Architecture **must** support resource tagging. | "OpenStack Resource Tags" [18] |

Table 26: VIM Requirements

### Interfaces & APIs Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| int.api.01 | API | The Architecture **must** provide APIs to access the authentication service and the associated mandatory features detailed in section 5. | [Keystone](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#521-keystone) |
| int.api.02 | API | The Architecture **must** provide APIs to access the image management service and the associated mandatory features detailed in section 5. | [Glance](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#522-glance) |
| int.api.03 | API | The Architecture **must** provide APIs to access the block storage management service and the associated mandatory features detailed in section 5. | [Cinder](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#523-cinder) |
| int.api | API | The Architecture **must** provide APIs to access the object storage management service and the associated mandatory features detailed in section 5. | [Swift](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#524-swift) |
| int.api.05 | API | The Architecture **must** provide APIs to access the network management service and the associated mandatory features detailed in section 5. | [Neutron](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#525-neutron) |
| int.api.06 | API | The Architecture **must** provide APIs to access the compute resources management service and the associated mandatory features detailed in section 5. | [Nova](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#526-nova) |
| int.api.07 | API | The Architecture **must** provide GUI access to tenant facing cloud platform core services except at Edge/Far Edge clouds. | [Horizon](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4319-horizon) |
| int.api.08 | API | The Architecture **must** provide APIs needed to discover and manage Cloud Infrastructure resources. | [Placement](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#527-placement) |
| int.api.09 | API | The Architecture **must** provide APIs to access the orchestration service. | [Heat](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#528-heat) |
| 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. | [Core OpenStack Services APIs](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter05.md#52-core-openstack-services-apis) |

Table 27**:** Interfaces and APIs Requirements

### Tenant Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| tnt.gen.01 | General | The Architecture **must** support self-service dashboard (GUI) and APIs for users to deploy, configure and manage their workloads. | [Horizon](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4319-horizon) and [Cloud Workload Services](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#3314-cloud-workload-services) |

Table 28**:** Tenant Requirements

### Operations and LCM

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| 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. |  |
| 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). |  |

Table 29**:** LCM Requirements

### Assurance Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Traceability** |
| asr.mon.01 | Integration | The Architecture **must** include integration with various infrastructure components to support collection of telemetry for assurance monitoring and network intelligence. |  |
| asr.mon.03 | Monitoring | The Architecture **must** allow for the collection and dissemination of performance and fault information. |  |
| 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. |  |

Table 30**:** Assurance Requirements

## Architecture and OpenStack Recommendations

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

### General Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 utilize a database. For latter see "Configuring the stateful services"[19**]** |
| 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](https://docs.openstack.org/senlin/train/) [20], cluster management service. |
| gen.rsl.02 | Resiliency | The Architecture **should** support resilient OpenStack service components that are not subject to gen.rsl.01. |  |

Table 31**:** General Recommendations

### Infrastructure Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 VM Instances. | [Virtual Storage](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#323-virtual-storage) |
| inf.ntw.04 | Network | The Architecture **should** support service function chaining. |  |
| inf.ntw.06 | Network | The Architecture **should** support Distributed Virtual Routing (DVR) to allow compute nodes to route traffic efficiently. |  |
| inf.ntw.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.ntw.09 | Network | The Cloud Infrastructure Network Fabric **should** embrace open-based standards and technologies. |  |
| inf.ntw.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.ntw.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). | [Acceleration](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter03.md#326-acceleration) |
| inf.acc.02 | Acceleration | The Architecture **should** support Cloud Infrastructure Acceleration (such as SmartNICs). | "OpenStack Future - Specs defined" [21] |
| inf.acc.03 | Acceleration | The Architecture **may** rely 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. | [Glance](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#4312-glance) |

Table 32**:** Infrastructure Recommendations

### VIM Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| vim.02 | General | The Architecture **should** support deployment of OpenStack components in containers. | [Containerised OpenStack Services](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/chapters/chapter04.md#432-containerised-openstack-services) |
| 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. |  |

Table 33**:** VIM Recommendations

### Interfaces and APIs Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| int.acc.01 | Acceleration | The Architecture **should** provide an open and standard acceleration interface to VNFs. |  |

Table 34**:** Interfaces and APIs Recommendations

### Tenant Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |

Table 35**:** Tenant Recommendations

### Operations and LCM Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 minimize 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. |  |

Table 36**:** LCM Recommendations

### Assurance Recommendations

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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. |  |

Table 37**:** Assurance Recommendations

### Security Recommendations

#### System Hardening (sourceRM 7.9.1 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 authorized 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. |  |

Table 38**:** System Hardening Recommendations

#### Platform and Access (sourceRM 7.9.2 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 [22] |

Table 39**:** Platform and Access Recommendations

#### Confidentiality and Integrity (sourceRM7.9.3 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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. |  |

Table 40**:** Confidentiality and Integrity Recommendations

#### Workload Security (sourceRM7.9.4 [1])

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

Table 41**:** Workload Security Recommendations

#### Image Security (sourceRM7.9.5 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |

Table 42**:** Image Security Recommendations

#### Security LCM (sourceRM7.9.6 [1])

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

Table 43**:** LCM Security Recommendations

#### Monitoring and Security Audit (sourceRM7.9.7 [1])

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.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 |  |

Table 44**:** Monitoring and Security Audit Recommendations

#### Open Source Software Security (sourceRM7.9.8 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| sec.oss.004 | 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 | “Software Bill of Materials (SBOM)” [23] |

Table 45**:** Open Source Software Security Recommendations

#### IaaC security (sourceRM7.9.9 [1])

##### Secure Design and Architecture Stage

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| sec.arch.001 | IaaC | 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 | IaaC | 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. |

Table 46**:** Reference Model Requirements - IaaC Security, Design and Architecture Stage

##### Secure Code Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| sec.code.002 | IaaC | 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 | IaaC | Source Code Review **should** be performed continuously during Secure Coding stage. | Typically done manually. |
| sec.code.004 | IaaC | 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 | IaaC | 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. |  |

Table 47**:** Reference Model Requirements - IaaC Security, Secure Code Stage

##### Continuous Build, Integration and Testing Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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 | DAST – 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. |

Table 48**:** Reference Model Requirements - IaaC Security, Continuous Build, Integration and Testing Stage

##### Continuous Delivery and Deployment Stage Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **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. |

Table 49**:** Reference Model Requirements - IaaC Security, Continuous Delivery and Deployment Stage

##### Runtime Defence and Monitoring Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| sec.run.002 | IaaC | 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 | IaaC | 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 | IaaC | Penetration Testing **should** be continuously applied during the Runtime Defence and Monitoring stage. | Typically done manually. |

Table 50**:** Reference Model Requirements - Iaac Security, Runtime Defence and Monitoring Stage

#### Compliance with Standards (sourceRM7.9.10 [1])

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref #** | **sub-category** | **Description** | **Notes** |
| sec.std.001 | Standards | The Cloud Operator **should** comply with Center for Internet Security CIS Controls [24] |  |
| 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)[25] |  |
| sec.std.003 | Standards | The Platform and Workloads **should** follow the guidance in the OWASP Cheat Sheet Series (OCSS) [26] |  |
| 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[27] |  |
| sec.std.005 | Standards | The Cloud Operator, Platform and Workloads **should** strive to improve their maturity on the OWASP Software Maturity Model (SAMM) [28] |  |
| sec.std.006 | Standards | The Cloud Operator, Platform and Workloads **should** utilize the OWASP Web Security Testing Guide [29] |  |
| sec.std.007 | Standards | The Cloud Operator, and Platform **should** satisfy the requirements for Information Management Systems specified in ISO/IEC 27001 [30]; ISO/IEC 27001 is the international Standard for best-practice information security management systems (ISMSs) |  |
| 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)[31] |  |
| sec.std.009 | Standards | The Cloud Operator, and Platform **should** implement the ISO/IEC 27032:2012 (or latest) Guidelines for Cybersecurity techniques [32]; 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 is the international Standard for ICT readiness for business continuity. |  |

Table 51**:** Security Recommendations

# Cloud Infrastructure Architecture - OpenStack

## Introduction

This Reference Architecture (RA-1) 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 section 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 separated into virtual resources via a hypervisor. The VIM is expected to be OpenStack in line with the OpenStack Foundation core release.

This section is organized as follows:

* Consumable Infrastructure Resources and Services: these are infrastructure services and resources being exposed northbound 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
  + Flavor\* 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
* 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" will be used in the OpenStack and this document context including when specifying configurations; the OpenStack term flavor includes the profile configuration information as "extra specs".

## Consumable Infrastructure Resources and Services

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 Cloud Infrastructure and with external networks

### Multi-Tenancy (execution environment)

The multi tenancy service will permit to host several VNF projects with the insurance to have isolated environment 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 security section 6.3.2.4).
* quota system to provide maximum resources that can be consumed.

This RA does not intend to restrict how workloads are distributed across tenants however where multiple related OpenStack clouds are deployed it is important that naming and quotas are kept consistent. section 4 provides a proposed naming convention for users and tenants (link-TBA).

### 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 flavor (resource sizing) needed to host VNF components. Profiles are defined in theReference Model section 2.5 [1].

### Virtual Storage

The three storage services offered by Cloud Infrastructure are:

* Persistent storage
* Ephemeral storage
* Image storage

Two types of persistent data storage are supported in OpenStack:

* Block storage
* Object storage

The OpenStack services, Cinder for block storage and Swift for Object Storage, are discussed below in Section 3.3 “Cloud Infrastructure Management Software (VIM)”.

Ephemeral data is typically stored on the compute host’s local disks, except in environments that support live instance migration between compute hosts. In the latter case, 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.

Images are stored using the OpenStack Glance service discussed below in Section 3.3 “Cloud Infrastructure Management Software (VIM)”.

TheOpenStack Storage Table [33] explains the differences between the storage types and typical use cases. TheOpenStack compatible storage backend drivers [34] table lists the capabilities that each of these drivers support.

### 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 utilize 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.

### 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 [35], 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 that today support the wide-area networks of the world’s major service providers but repurposed to work with virtualized workloads and cloud automation in data centers that can range from large scale enterprise data centers 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 theReference Model Section 3.5 [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.

### Acceleration

Acceleration deals with both hardware and software accelerations. Hardware acceleration is the use of specialized 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.

## Virtualised Infrastructure Manager (VIM)

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

### VIM Core services

OpenStack is a complex, multi-project framework, so we initially will 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 1; 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.

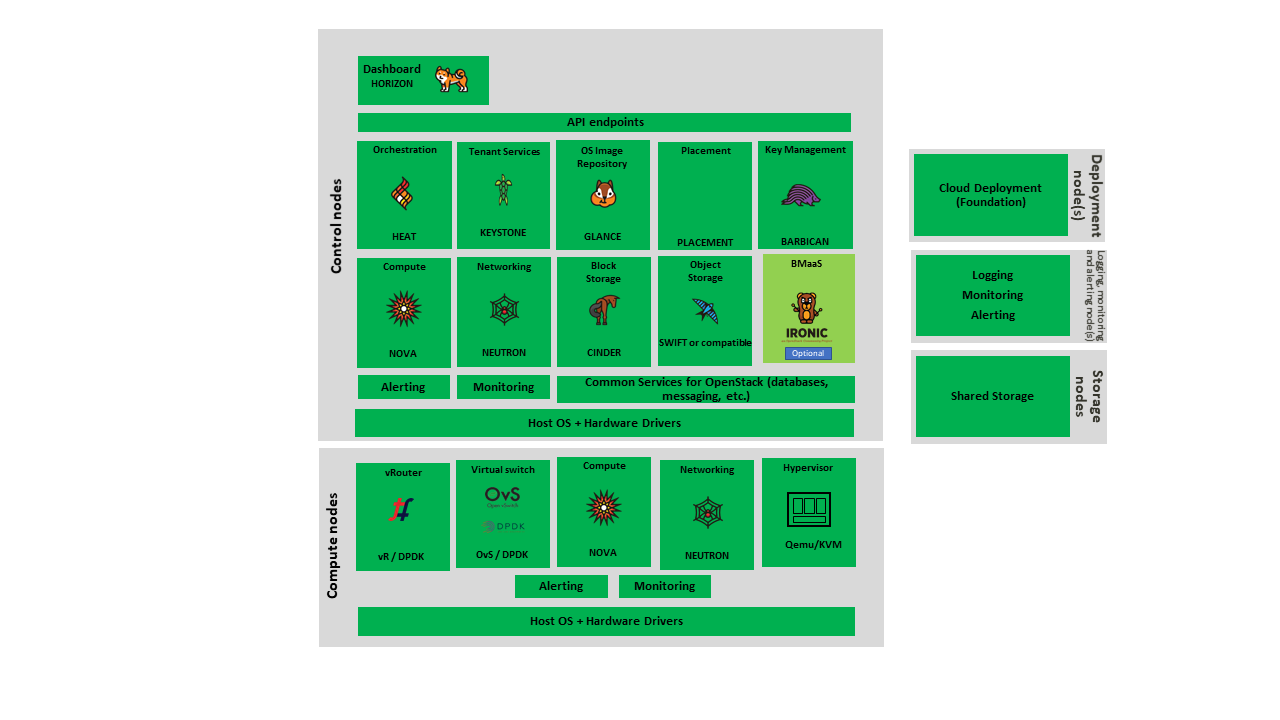


Figure 1**:** OpenStack Core Services

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.

#### 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. Figure 2 depicts the 4 types of nodes constitutive of the Infrastructure: control, compute, network and storage nodes.

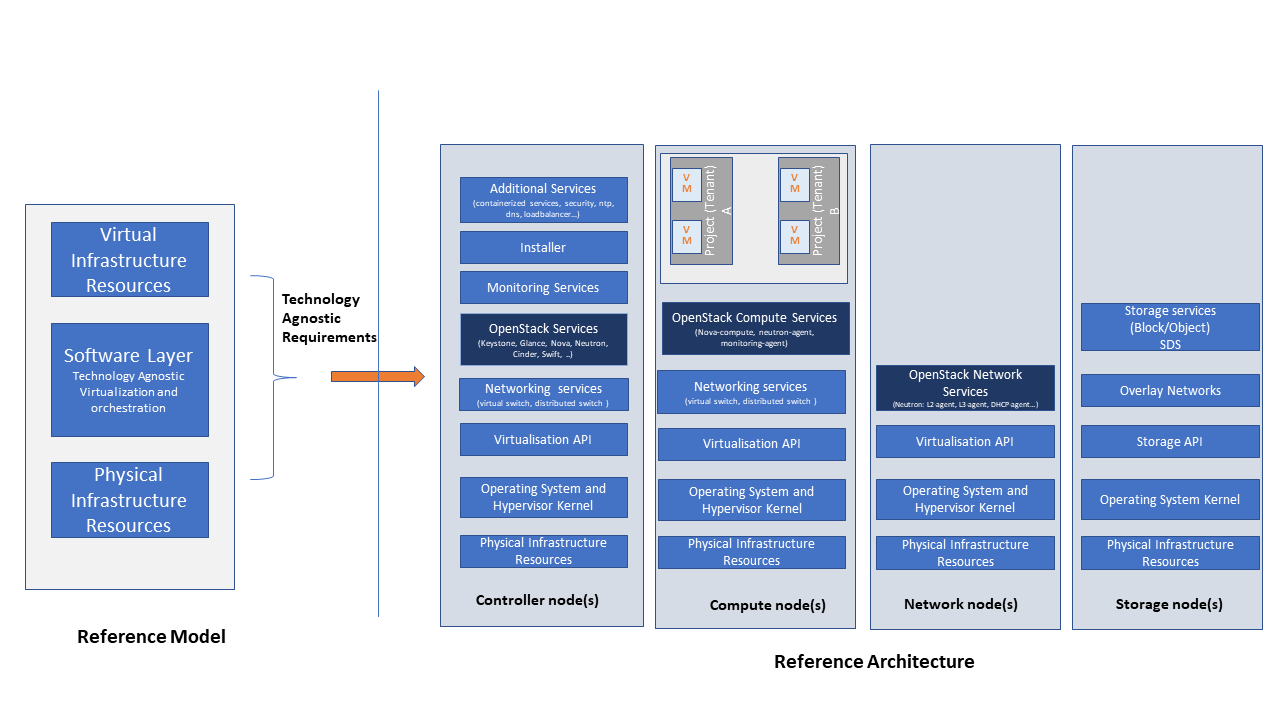


Figure 2**:** OpenStack Services Topology

Deployments can be structured using the distribution of services amongst the 4 node types as depicted in Figure 2, 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 also maps the OpenStack core services to the Reference Model (RM) Virtual Infrastructure ManagerReference Model section 3.2.2 Virtual Infrastructure Manager [1].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 utilizes 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 |  |

Table 52: Cloud Controller Services

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 containerized or VM hosted as long as they provide the high availability principles described above.

The APIs for these OpenStack services are listed insection “Interfaces and APIs”.

#### Cloud Workload Services

This section describes the core set of services and service components needed to run workloads including 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 instances)
  + Neutron Networking service: neutron-l2-agent (manage local Open vSwitch (OVS) configuration), VXLAN
  + Local Storage (Ephemeral, Root, etc.)
  + Attached Storage (using Local drivers)

### 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. However, as Anuket RA uses Keystone v3 in section 5.2.1, so it is recommended to use the term "project" when referring to OpenStack and usetenant (RM Section 3.2.1 [1]) when referring to multi-tenancy. According to [OpenStack glossary](https://docs.openstack.org/doc-contrib-guide/common/glossary.html) [7], 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.

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

### 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 thesoftware and hardware profile configurations (RM Section 5 [1]).

## 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.

### Virtualisation

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 HypervisorFeature Support Matrix [36]. 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 it can interoperate with other OpenStack components in this Reference Architecture using standard interfaces and APIs as specified in Section 5.

### 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
  + discussed insection 4.2.4
* Acceleration
  + SmartNIC
  + GPU
  + FPGA

#### Compute

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 inReference Model sections 5.3 and 5.4 [1].

#### Network

The recommended network architecture is spine and leaf topology.

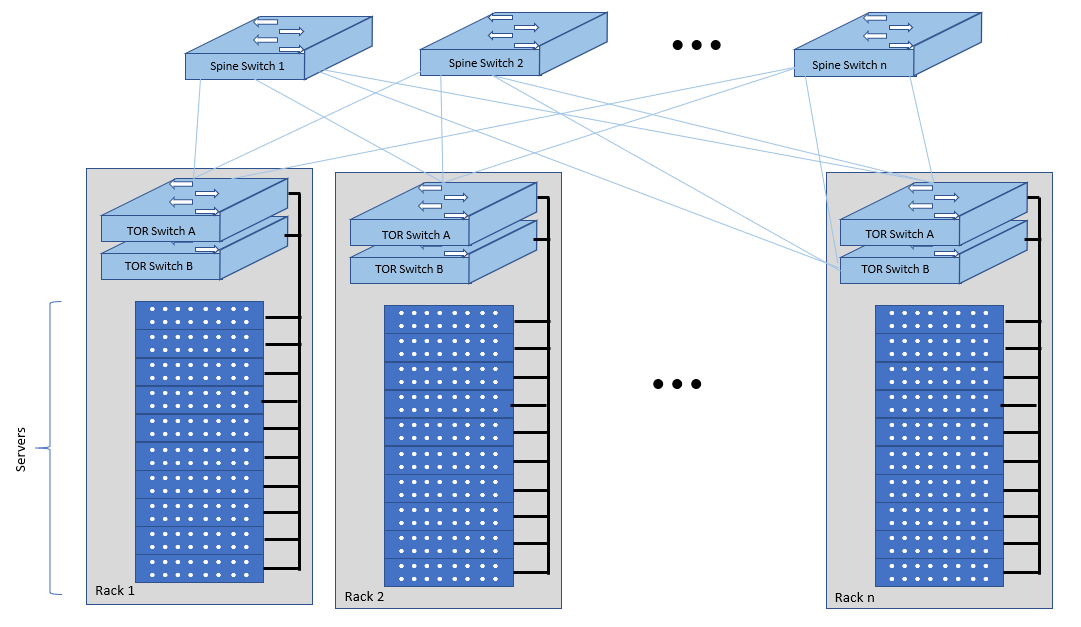


Figure 3**:** Network Fabric – Physical

Figure 3 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](https://docs.openstack.org/arch-design/design-storage.html) Storage [37] supports many different storage architectures and backends. 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.

## 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 endpoints) will be deployed and they all contain isolated resources that do not rely on each other, by design. The application layer must span such endpoints in order to provide the required service SLA. Irrespective of the nature of the deployment characteristics (e.g., number of racks, number of hosts, etc.), 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 center capable of hosting potentially thousands of servers and the networking to support them
* Intermediate data center (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.

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 high availability based on quorum (aligned with vendor implementation)
* compute nodes sized to handle the entire workload following local failure scenario

### Topology Overview

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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 upgrades |
| 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 upgrades |

Table 53: Topology overview

### Topology Detail

#### 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, 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 failures and 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 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 and so any fault is isolated to a single cloud and also allows maintenance to take place without service impact.

# Cloud Infrastructure + VIM Component Level Architecture

## Introduction.

Section 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 section delves deeper into the capabilities of these different resources and their needed configurations to create and operate an OpenStack-based IaaS cloud. This section 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 section also covers implementation support for theReference Model profiles and flavours (RM Section 2.4 [1]); the OpenStack flavor types capture both the sizing and the profile configuration (of the host).

## Underlying Resources

### Virtualisation

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

* Configuration:OpenStack [38] specifies the following KVM configuration 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 utilize compatible virtual machine images
  + Configure Compute backing storage
  + Specify the CPU Model for KVM guests (VMs)
  + KVM Performance Tweaks
* Hardening the virtualization layers [39]
  + OpenStack recommends minimizing the code base by removing unused components
  + sVirt (Secure Virtualisation) provides isolation between VM processes, devices, data files and system processes

### 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 insection 5.4 of the Reference Model [1]. Additionally, for OpenStack we need to set the following boot parameters:

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

Table 54: BIOS/boot parameters – Control node

* 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)).
  + TheServices Placement Summary table [40] specifies the number of instances that are required based upon the cloud size (number of nodes).

#### Network nodes

Network nodes are mainly used for L3 traffic management for overlay tenant network (see more detail in section [4.3.1.5 Neutron](#_4.3.1.5_Neutron))

* BIOS requirements

|  |  |
| --- | --- |
| **BIOS/boot Parameter** | **Value** |
| Boot disks | RAID 1 |

Table 55: Boot parameters – Network node

* 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 [4.3.1.5 Neutron](#_4.3.1.5_Neutron))

#### Storage nodes

* BIOS requirements

|  |  |
| --- | --- |
| **BIOS/boot Parameter** | **Value** |
| Boot disks | RAID 1 |

Table 56: Boot parameter – Storage node

* HW specifications
* How many nodes to meet SLA
* Sizing rules

#### 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 (RM 2.4 Profile Extensions [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 theReference Model section 5.4 [1]
* BIOS requirement
  + The general BIOS requirements are described in theReference Model section 5.4 [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. Table 57 lists a few simple examples of profile extensions and some of their capabilities.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Profile Extensions** | **Description** | **CPU Allocation Ratio** | **SMT** | **CPU Pinning** | **NUMA** | **Huge Pages** | **Data Traffic** |
| B1 | Basic Profile  No CPU over-subscription profile extension | 1:1 | Y | N | N | N | OVS-kernel |
| B4 | Basic Profile  4x CPU over-subscription profile extension | 4:1 | 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 |

Table 57: Profile Extensions and Capabilities

**BIOS Settings**

A number of capabilities need to be enabled in the BIOS (such as NUMA and SMT); the Reference Model [1] section 5.1 on “Cloud Infrastructure Software profile description” specifies where each of the capabilities is 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:

|  |  |  |
| --- | --- | --- |
| **BIOS/boot Parameter** | **Basic** | **High Performance** |
| Boot disks | RAID 1 | RAID 1 |

Table 58: Boot parameters – Compute node

* 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:

|  |  |
| --- | --- |
| **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” [41] and“DPDK performance reports” [42]. 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. |

Table 59: DPDK configuration

* Sizing rules

|  |  |
| --- | --- |
| 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 60: Sizing values

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **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) |

Table 61: Sizing rules

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 specific to the profile. If not properly sized or when demand changes 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 type. This is operationally efficient as any server can be utilized 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[R2.3.2 Infrastructure Requirement](#_2.3.2_Infrastructure_Requirements) req.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 use 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 Train 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 VM 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 utilize 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 Virtual Machines (VMs or vServers) 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 VMs of these flavor types can be hosted on similarly configured (host profile) compute hosts. Similarly, a VM 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 VM 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 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 the next section.

##### 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 VM. OpenStack and MANO manage SR-IOV configurations for Tenant VM 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 VM 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*

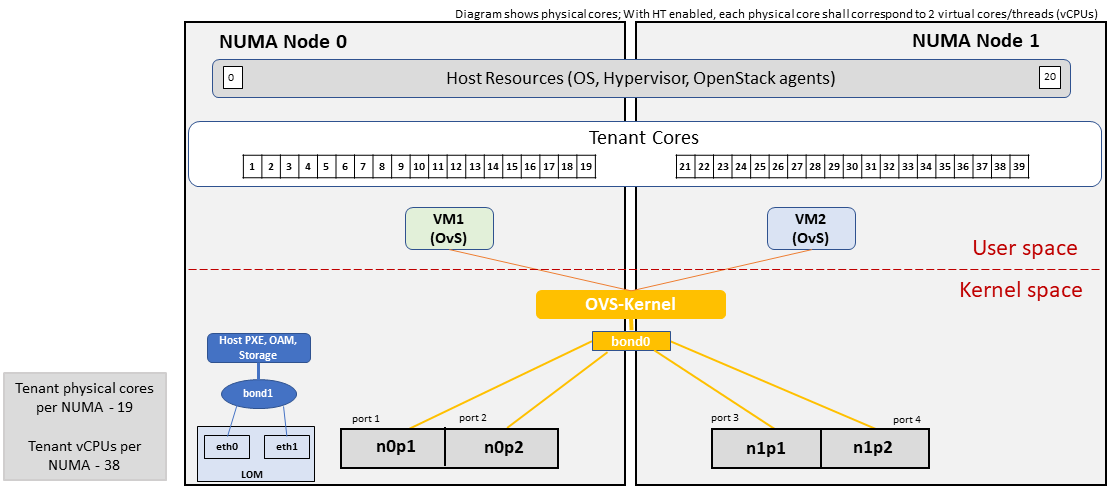


Figure 4: Basic Profile Host Configuration (example and simplified).

Let us refer to the data traffic networking configuration of Figure 4 to be part of the hp-B1-a and hp-B4-a host profiles and this requires the configurations as Table 62.

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

Table 62: Configuration of Basic Flavor Capabilities

Figure 5 shows the networking configuration where the storage and OAM share networking but are independent of the PXE network.

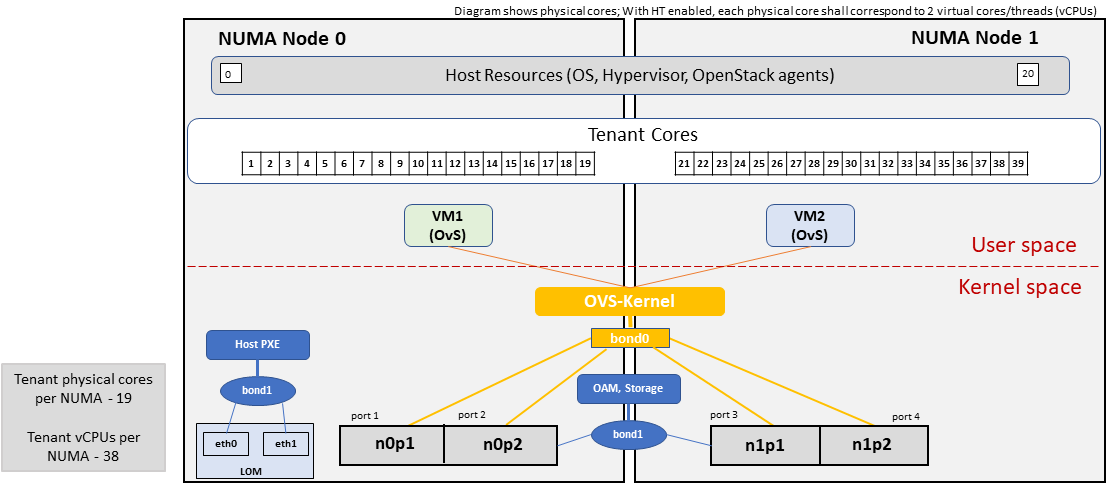


Figure 5: Basic Profile Host Configuration with shared Storage and OAM networking (example and simplified).

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 62.

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:

|  |  |  |
| --- | --- | --- |
| **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 |

Table 63: Tenant CPU isolation, Basic profile

*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 64 gives examples of three different host profiles; one each for HV, HD and HS Profile Extensions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |

Table 64: Configuration of High Performance Flavor Capabilities

*Host Networking configuration for HD Profile Extensions*

An example of the data traffic configuration for the HD (OVS-DPDK) Profile Extensions is shown in Figure 6.

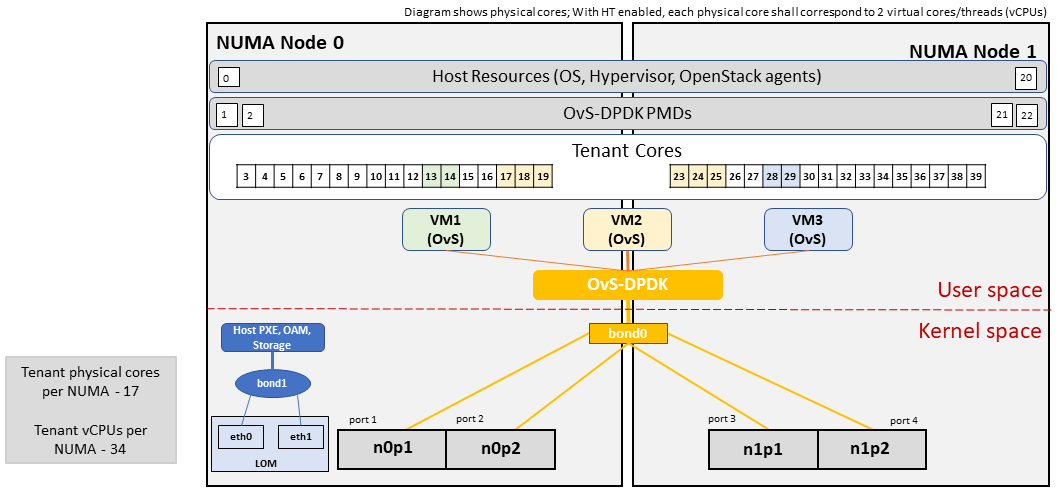


Figure 6: High Performance Profile Host Configuration with DPDK acceleration (example and simplified).

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

|  |  |  |
| --- | --- | --- |
| **Hardening the virtualization layers** | **Hardening the virtualization layers** | **Hardening the virtualization layers** |
| 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 |

Table 65: Tenant and DPDK CPU isolation, HD profile

*Host Networking configuration for HS Profile Extensions*

An example of the data traffic configuration for the HS (SR-IOV) Profile Extensions is shown in Figure 7.

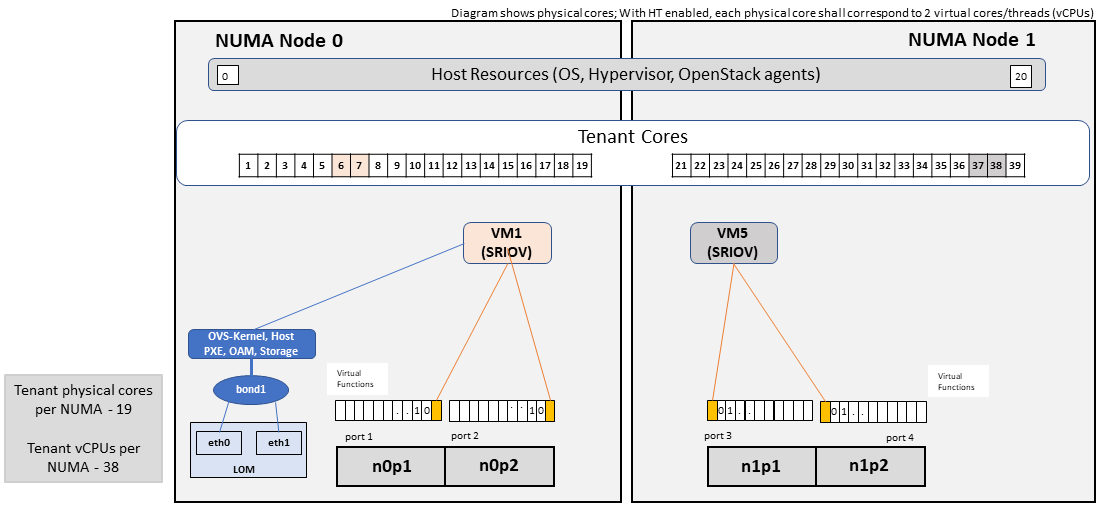


Figure 7: High Performance Profile Host Configuration with SR-IOV (example and simplified).

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

|  |  |  |
| --- | --- | --- |
| **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 |

Table 66: Tenant CPU isolation, HS profile

##### 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 VM instantiation of B1 flavor hosts from the ha-B1 host aggregate will be selected.

### 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

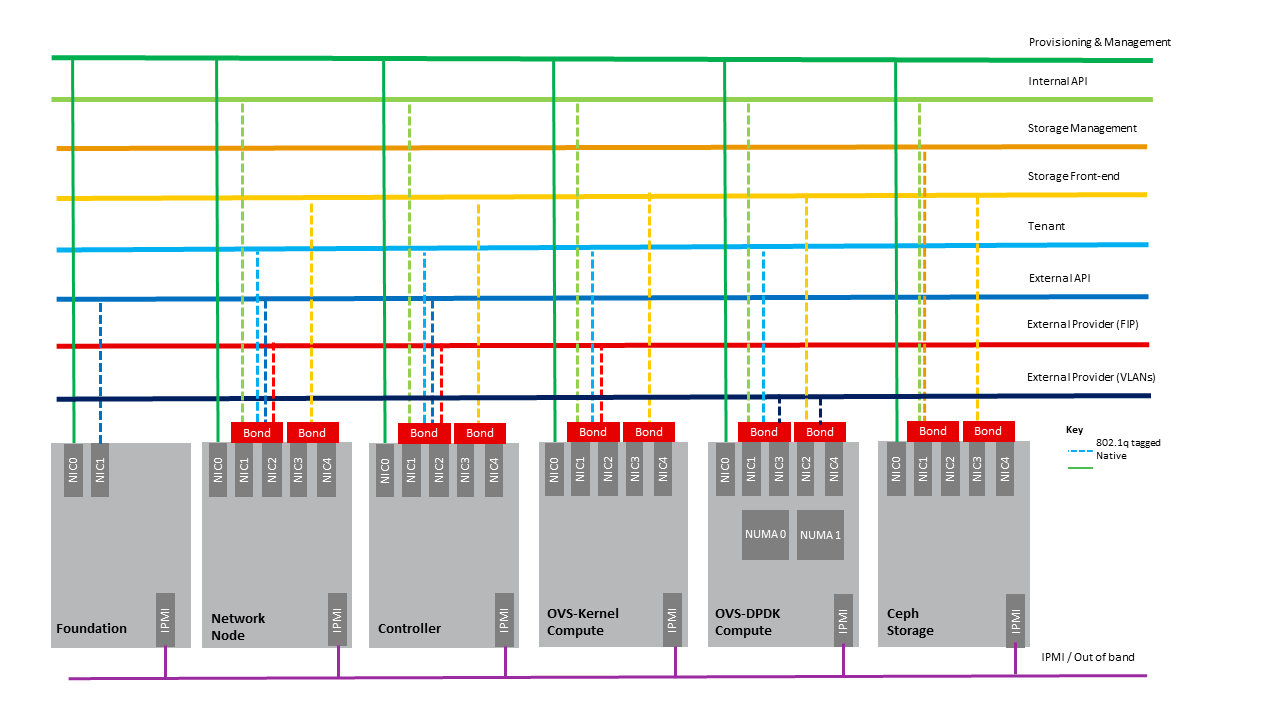


Figure 8: Indicative OpenStack Network Layout.

|  |  |  |
| --- | --- | --- |
| **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. | Security Domain: Management  Externally Routable: No  Connected to: All nodes |
| Internal API | Intra-OpenStack service API communications, messaging and database replication | 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 | Security Domain: Storage  Externally Routable: No  Connected to: All nodes except foundation |
| Storage Front-end | Block/Object storage access via cinder/swift | Security Domain: Storage  Externally Routable: No  Connected to: All nodes except foundation |
| Tenant | VXLAN / Geneve project overlay networks (OVS kernel mode) – i.e. RFC1918 re-usable private networks as controlled by cloud administrator | Security Domain: Underlay  Externally Routable: No  Connected to: controllers and computes |
| External API | Hosts the public OpenStack API endpoints including the dashboard (Horizon) | 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 private networks | 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 | 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 | Security Domain: Management  Externally routable: No  Connected to: IPMI port on all servers |

Table 67: Logical networks decription

A VNF application network topology is expressed in terms of VMs, vNIC interfaces with vNet access networks, and WAN Networks while the VNF Application VMs require multiple vNICs, VLANs, and host routes configured within the VM’s Kernel.

#### Octavia v2 API conformant Load Balancing

Load balancing is needed for automatic scaling, managing availability and changes.Octavia [43] 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 utilizes Amphorae Load Balancer. Amphorae consists of a fleet of 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 utilized 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 provides 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.

Anuket Reference Architecture support the ML2 plugin (see below) as well as the service plugins including forFwaaS (Firewall as a Service) [44],LbaaS (Load Balancer as a Service) [45], andVPNaaS (VPN as a Service) [46]. The OpenStack wiki provides a list ofNeutron plugins [47].

Every Neutron plugin needs to implement a minimum set of commonmethods (actions for Train release) [48]. 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.

[Section 5 "Interfaces and APIs"](#_5._Interfaces_and) of this Reference Architecture provides a list of“[Neutron Extensions](#_5.2.5._Neutron)”. The current available extensions can be obtained using“List Extensions API” [49] and details about an extension using“Show extension details API” [50].

**Neutron ML2 integration** The OpenStack Modular Layer 2 (ML2) plugin simplifies adding networking technologies by utilizing 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. TheOpenStack ML2 documentation [51] lists example mechanism drivers.

#### Network quality of service

With support of VNF workloads, the resources 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 VMs, 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 VM and send the response to the VM with the Ipv4 and/or Ipv6 addresses and the lease time. Please see OpenStack provider Network.
* DNS
* LDAP
* IPAM

### Storage Backend

Storage systems are available from multiple vendors and can also utilize 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 OpenStackCinder documentation [52] 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 help 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 list the supported methods for each of the vendor drivers. The OpenStack CinderAvailable Drivers [53] documentation provides a list of all OpenStack compatible drivers and their configuration options.

TheCinder Configuration [54] document provides information on how to configure cinder including Anuket required capabilities for volume encryption, Policy configuration, quotas, etc. TheCinder Administration [55] document provides information on the capabilities required by Anuket including managing volumes, snapshots, multi-storage backends, migrate volumes, etc.

Ceph [56] 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 handle 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.

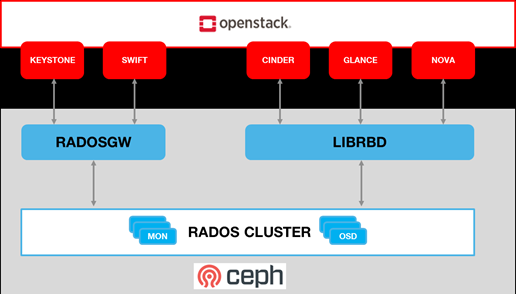


Figure 9: Ceph Storage System.

**BIOS Requirement for Ceph servers**

|  |  |
| --- | --- |
| **BIOS/boot Parameter** | **Value** |
| Boot disks | RAID 1 |

Table 68: Boot parameter – Ceph server

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

## Virtualised Infrastructure Manager (VIM)

This section covers:

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

### VIM Services

A high level overview of the core OpenStack Services was provided insection 3. In this section we describe the core and other needed services in more detail.

#### Keystone

Keystone 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 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 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 is the object storage management service, 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 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. Table 69 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 utilized for networking.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 VMs on tenant networks and supports services such as Firewall-as-a-service (FwaaS) [41] and Load Balancer-as-a-service (LbaaS) [42] | 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 [5] or Linux Bridge) configuration and local networking configuration for VMs hosted on that node. | Required | Every Compute Node |

Table 69: Neutron Services Placement

##### Issues with the standard networking (centralized routing) approach

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

Two VMs on different compute nodes and using different project networks (a.k.a. tenant networks) where both of the project networks are connected by a project router. For communication between the two VMs (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)” [57] 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 [58] 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 (3.2.5. Virtual Networking – 3rd party SDN solution). With SDN incorporated in OpenStack, changes to the network are triggered by workloads (and users), translated into Neutron APIs and then handled through neutron plugins by the corresponding SDN agents.

#### Nova

Nova is the compute management service, depends on all above components and is deployed after. 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 is the bare metal provisioning service. Ironic depends on all above components and is deployed after. 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. As Ironic is currently not invoked directly (only invoked through other services such as Nova) hence its APIs will not be specified.

#### Heat

Heat is the orchestration service using template 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 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 OpenStackPlacement service [59] 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 utilized by nova-compute and nova-scheduler. Other OpenStack services such as Neutron or Cyborg can also utilize placement and do so by creatingProvider Trees [60]. The following data objects are utilized in theplacement service [59]:

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 specifies 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 [61] 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.”.

### Containerised OpenStack Services

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

In section 3, [Figure 2](https://github.com/cntt-n/CNTT/blob/master/doc/ref_arch/openstack/figures/RA1-Ch03-OpenStack-Services-Topology.png) shows a high level Virtualised OpenStack services topology. The containerized OpenStack services topology version is shown in Figure 10.

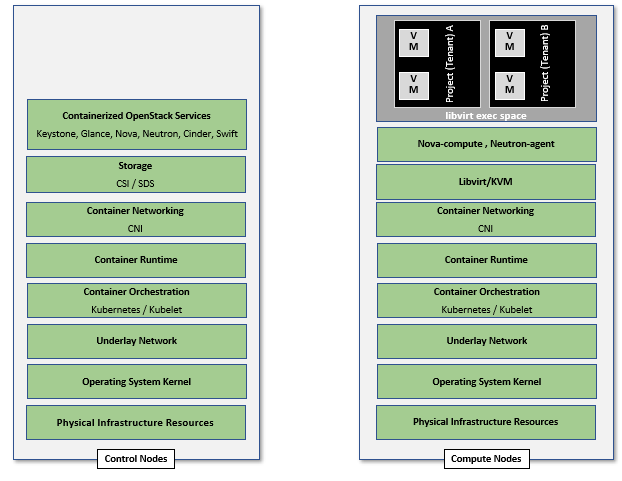


Figure 10: Containerised OpenStack Services Topology.

## Consumable Infrastructure Resources and Services

### Support for Cloud Infrastructure Profiles and flavors

Reference Model [1] sections 4 and 5 provide information about the Cloud Infrastructure Profiles and their size information. OpenStack flavors with their set of properties describe the VM capabilities and size required to determine the compute host which will run this VM. 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.

|  |  |  |  |
| --- | --- | --- | --- |
| **Flavor Capabilities** | **Reference**  **RM section 4 and 5** | **Basic** | **High Performance** |
| CPU allocation ratio (custom extra\_specs) | infra.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 | infra.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 | infra.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 | infra.com.cfg.004 |  | --property hw:mem\_page\_size=<small |large | size> |
| SMT | infra.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:STORAGE\_DISK\_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\_inbound\_average=3125000  and  vif\_outbound\_average=3125000  Note: 25 Gbps = 3125000 kilobytes per second |

Table 70: Flavors properties per profile

* For example as defined inReference Model Profile Extensions (RM [1] section 2.4.2).

In addition, 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[40].

### 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 VM 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 VM recovery.
* Physical host reboot does not trigger the automatic start of VM

**Limitations and constraints**

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

### 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.

## Cloud Topology and Control Plane Scenarios

Typically, Clouds have been implemented in large (central) data centers with 100’ to tens of thousands of servers. Telco Operators have also been creating intermediate data centers in central office locations, colocation centers, and now edge centers 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). Section 3 of this document, discusses“[Cloud Topology](#_3.5._Cloud_Topology)” and lists 3 types of data centers: Large, Intermediate and Edge.

For ease of convenience, unless specifically required, in this section we will use Central Cloud Center, Edge Cloud Center and Intermediate Cloud Center as representative terms for cloud services hosted at centralised large data centers, Telco edge locations and for locations with capacity somewhere in between the large data centers and edge locations, respectively. The mapping of various terms, including the Reference Model [1] terminology specified in Section 8.3.5 andthe “Open Glossary of Edge Computing” [62] is as follows:

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

In the Intermediate and Edge cloud centers, 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 centers. InReference Model [1], section 8.3 “Telco Edge Cloud” specifies the physical and environmental characteristics, infrastructure capabilities and deployment scenarios of different locations.

Section3.3.1.1 “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 subsections of this Section 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 center resource capacity and capabilities determine the deployment of OpenStack services in the different node types.

The Central Cloud Centers are organized around a Centralised Control Plane. With the introduction of Intermediate and Edge Cloud Centers, 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 Center, Local Control Planes hosting a full-set or subset of the controller services are also deployed on the Intermediate and Edge Cloud Centers. Table 71 presents examples of such deployment choices.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **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 DCs | \*\* in other DC | \* in Large DC | \* in Large DC | \*\* in another DC | \*\* in another DC | \*\* in another 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) |  |

Table 71**:** Distribution of OpenStack services on different nodes depending upon Control Plane Scenario

### Edge Cloud Topology

The Reference Model [1] section 8.3 “Telco Edge Cloud”, presents the deployment environment characteristics, infrastructure characteristics and new values for the Infrastructure Profiles at the Edge.

TheEdge computing whitepaper [63] includes information such as the services that run on various nodes. The information from the whitepaper coupled with that from theOpenStack Reference Deployment Architecture 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 Center 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.

The Reference Model [1]section 8.3.4 “Telco Edge Cloud: Platform Services Deployment” 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.

The Reference Model [1]section 8.3.3 “Telco Edge Cloud Infrastructure Profiles” 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.

#### Edge Cloud Deployment

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

* [Airship](https://docs.airshipit.org/) [65]: declaratively configure, deploy and maintain an integrated virtualization and containerization platform

* [Starling-X](https://www.starlingx.io/) [66]: cloud infrastructure software stack for the edge

* [Triple-O](https://wiki.openstack.org/wiki/TripleO) [67]: for installing, upgrading and operating OpenStack clouds

The Reference Implementation (RI-1) is responsible to choose the tools for the implementation and shall specify implementation and usage details of the chosen tools.

# Interfaces and APIs

## Introduction

This section presents a consolidated set of OpenStack Service APIs corresponding to the ETSI NFV Nf-Vi, Vi-Vnfm and Or-Vi interfaces. The OpenStack Train 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** can be considered as Anuket RA Conformant.

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

OpenStack is a multi-project framework composed of services evolving independently. 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 to each OpenStack service. In addition to major API versions, some OpenStack services (Nova, Glance, Keystone, Cinder...) support microversions. The microversions allow to introduce new features over time. In this section, the **major version** and **microversion** are specified per service. The mentioned microversion is the minimal microversion that supports the features requested for Anuket. For the purpose of conformance tests, this section also identifies the set of the features, offered by a service, that are mandatory for Anuket compliant implementation.

## 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 Train 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 theCompute Service [68], “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.”

### Keystone

|  |  |  |
| --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal API Microversion** |
| Identity: Keystone | v3 | 3.13 |

Table 72: Keystone API version and microversion

|  |  |
| --- | --- |
| **Keystone Features** | **Mandatory** |
| application\_credentials | X |
| external\_idp |  |
| federation |  |
| oauth1 |  |
| project\_tags | X |
| security\_compliance | X |
| trust | X |

Table 73: Keystone features

Identity API v3: [69]

Identity API v3 extensions:[70]

Security compliance and PCI-DSS:[71]

### Glance

|  |  |  |
| --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal API Microversion** |
| Image: Glance | v2 | 2.9 |

Table 74: Glance API version and microversion

Image Service Versions: [72]

### Cinder

|  |  |  |
| --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal API Microversion** |
| Block Storage: Cinder | v3 | 3.59 |

Table 75: Cinder API version and microversion

|  |  |
| --- | --- |
| **Cinder Features** | **Mandatory** |
| backup | X |
| clone | X |
| consistency\_group |  |
| extend\_attached\_volume |  |
| manage\_snapshot | X |
| manage\_volume | X |
| multi\_backend |  |
| snapshot | X |
| volume\_revert | X |

Table 76: Cinder features

Block Storage API: [69]

REST API Version History: [73]

### Swift

|  |  |
| --- | --- |
| **OpenStack Service** | **API Version** |
| Object Storage: Swift | v1 |

Table 77: Swift API version

|  |  |
| --- | --- |
| **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 |

Table 78: Swift features

Object Storage API: [69]

Discoverability: [74]

### Neutron

|  |  |
| --- | --- |
| **OpenStack Service** | **API Version** |
| Networking: Neutron | v2.0 |

Table 79: Neutron API version

|  |  |
| --- | --- |
| **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 |  |
| 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-security-groups |  |
| 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 |  |
| trunk | X |
| trunk-details | X |
| uplink-status-propagation |  |

Table 80: Neutron extensions

|  |  |
| --- | --- |
| **Neutron Type Drivers** | **Mandatory** |
| geneve |  |
| gre |  |
| vlan | X |
| vxlan |  |

Table 81: Neutron type drivers

Networking Service APIs: [69]

The exhaustive list of extensions is available at[75]

### Nova

|  |  |  |  |
| --- | --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal API Microversion** | |
| Compute: Nova | v2.1 | 2.79 |

Table 82: Nova API version and microversion

|  |  |
| --- | --- |
| **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 |
| snapshot | X |
| spice\_console |  |
| suspend | X |
| swap\_volume |  |
| vnc\_console |  |
| volume\_multiattach |  |

Table 83: Nova features

Compute API: [69]

REST API Version History: [76]

### Placement

|  |  |  |  |
| --- | --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal API Microversion** | |
| Placement | v1 | 1.36 |

Table 84: Placement API version and microversion

Placement API: [69]

REST API Version History: [77]

### Heat

|  |  |  |
| --- | --- | --- |
| **OpenStack Service** | **API Version** | **Maximal Template Version** |
| Orchestration: Heat | v1 | 2018-08-31 |

Table 85: Heat API version and microversion

Orchestration Service API:[69]

Template version history: [78]

Heat Orchestration Template (HOT) specification:[79]

## Consolidated Set of APIs

### OpenStack Interfaces

This section illustrates some of the Interfaces provided by OpenStack; the exhaustive list of APIs is available at[69].

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 programming language specific cloud libraries.

**APIs**

|  |  |  |  |
| --- | --- | --- | --- |
| **OpenStack Service** | **Link for API list** | **API Version** | **Maximal API Microversion** |
| Identity: Keystone | <https://docs.openstack.org/api-ref/identity/v3/> | v3 | 3.13 |
| Compute: Nova | <https://docs.openstack.org/api-ref/compute/> | v2.1 | 2.79 |
| Networking: Neutron | <https://docs.openstack.org/api-ref/network/v2/> | v2.0 |  |
| Image: Glance | <https://docs.openstack.org/api-ref/image/v2/> | v2 | 2.9 |
| Block Storage: Cinder | <https://docs.openstack.org/api-ref/block-storage/v3/> | v3 | 3.59 |
| Object Storage: Swift | <https://docs.openstack.org/api-ref/object-store/> | v1 |  |
| Placement | <https://docs.openstack.org/api-ref/placement/> | v1 | 1.36 |
| Orchestration: Heat | <https://docs.openstack.org/api-ref/orchestration/v1/> | v1 |  |

Table 86: OpenStack APIs versions and microversions

### Kubernetes Interfaces

The Kubernetes APIs are available at[80].

### KVM Interfaces

The KVM APIs are documented in Section 4 of the document [81].

#### Libvirt Interfaces

The Libvirt APIs are documented in[82].

### Barbican

|  |  |
| --- | --- |
| **OpenStack Service** | **API Version** |
| Key Manager: Barbican | v1 |

Table 87: Barbican API version

Barbican API Documentation: [83]

# Security

## Introduction

This guide is intended to provide basic security requirements to architects who are implementing Cloud Infrastructure usingOpenStack 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.

## Security Requirements

[Section 2](#_Reference_Model_Requirements) gathers all requirements and recommendations regarding security topics developed in this section.

## Cloud Infrastructure and VIM Security

In the “Security boundaries and threats” section of the OpenStack security guide [84], 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.

### 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 thePrinciple 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 theCIS Password Policy guide [12] 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 organization, etc.

#### Function and Software

Infrastructure must be implemented to perform the minimal function that’s practically needed to support 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 system 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):

|  |  |
| --- | --- |
| **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 |

Table 88: Timescale remediation

**See**Common Vulnerability Scoring System [14] (and NIST Vulnerability Metrics [82]).

#### 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 refresh daily
* 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

### Platform Access

#### Identity Security

TheOpenStack Identity service (Keystone) [86] provides identity, token, catalog, and policy services for use specifically by services in the OpenStack family. Identity service is organized 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 defense 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 tothe “Authentication Methods” section in [86].

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 authorization 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.

#### Authorization

Authorization serves as the next level of defense. 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 action 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 theOpenStack “Policies” section in [86].

#### RBAC

In order to properly manage user access to OpenStack services, service providers must utilize the Role Based Access Control (RBAC) system (sec.sys.001, sec.sys.007). Based on the OpenStack Identify 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 threedefault roles [87]: 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

### Confidentiality and Integrity

Confidentiality implies that data and resources must be protected against unauthorized introspection/exfiltration. Integrity implies that the data must be protected from unauthorized 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 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 theOpenStack security guide [88] 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 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 minimize 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 (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 tool such as CIS-CAT (Center for Internet security- Configuration Assessment Tool [89]) to check the compliance of systems configuration against respective CIS benchmarks [90].

It is strongly recommended to protect Linux repositories and Docker registries against the corruption of their data, 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 Glance image signing feature [91] 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 key manager service of OpenStack.

### 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, but also AZs if engineered to do so. 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 can take a long time to identify faults and correct them. Thus, 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 section including secure boot.

#### SR-IOV and DPDK Considerations

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 [92].

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

### 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 theOpenStack Security Guide [93]. The OpenStack Security Guide includes reference to the “OpenStack Virtual Machine Image Guide” [94] 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 [95]. 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 GS NFV-SOL004 [96].

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 GS NFV-SEC021 [97].

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

### Security LCM

Cloud Infrastructure LCM encompasses provisioning, deployment, configuration and management (resources scaling, services upgrades…) as described in [Section 7](#_7._Operations_and). 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 7.2.1 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…). 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 distant storage. The communication channel used for log collection must be protected in integrity and confidentiality and logs protected against unauthorized 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.

### Monitoring and Security Audit

This 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 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 (ex. Nova in OpenStack) that form the Cloud Infrastructure (sec.mon.001).
* All security logging mechanisms must be active from system initialization (sec.mon.018):
  + These mechanisms include any automatic routines necessary to maintain the activity records and cleanup 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 unauthorized 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 (ex. Password) (sec.mon.019);
* Keystone Token;
* Proprietary or Sensitive Personal Information.

#### Where to Log

* The logs must be store 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).

# Operations and Life Cycle Management

## Introduction

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 section 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 section.

### 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.

### 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.

## Cloud Infrastructure and VIM configuration management

Section 9 “Configuration and Lifecycle Management” of the Reference Model [1] defines the functions of Configuration and Life Cycle Management (LCM). To operate and manage a scalable cloud, that minimizes 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.

### 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.

**Configuration and subsequent software installation** is then handed over to a configuration management tool or life cycle manager.

OpenStack TripleO documentation [98], and similar documentation from OpenStack vendors, delves into great detail on the provisioning of servers (bare metal), deploying and configuring OpenStack services.

Section 6 of the Reference Implementation [99] a set of Installer requirements are specified with a couple of Installers (such as Airship and Triple-O) are described in section 8.5 of the Reference Implementation [99]. It should be noted that the installers chosen in order to automate deployment depend on the cloud provider.

Systems such as Airship [65] are not only provisioning tools but also a configuration management system. For example,Airship [100] specifies how to provision and deploy the IaaS, and on how to update configuration including OpenStack services.

For Airship, section 8.5.1.1 of the Reference Implementation [99] specifies the required descriptor files and section 8.5.1.2 of the Reference Implementation [99] describes the steps to provision the OpenStack based IaaS.

### 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 are 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.

## 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 need 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.

## Logging, Monitoring and Analytics

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

### 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 Section 4 “Infrastructure Capabilities, Metrics and Catalogue” of 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 prioritization, 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

### Monitoring

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

Monitoring systems fulfil 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.

### 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.

### Logging, Monitoring, and Analytics (LMA) Framework

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

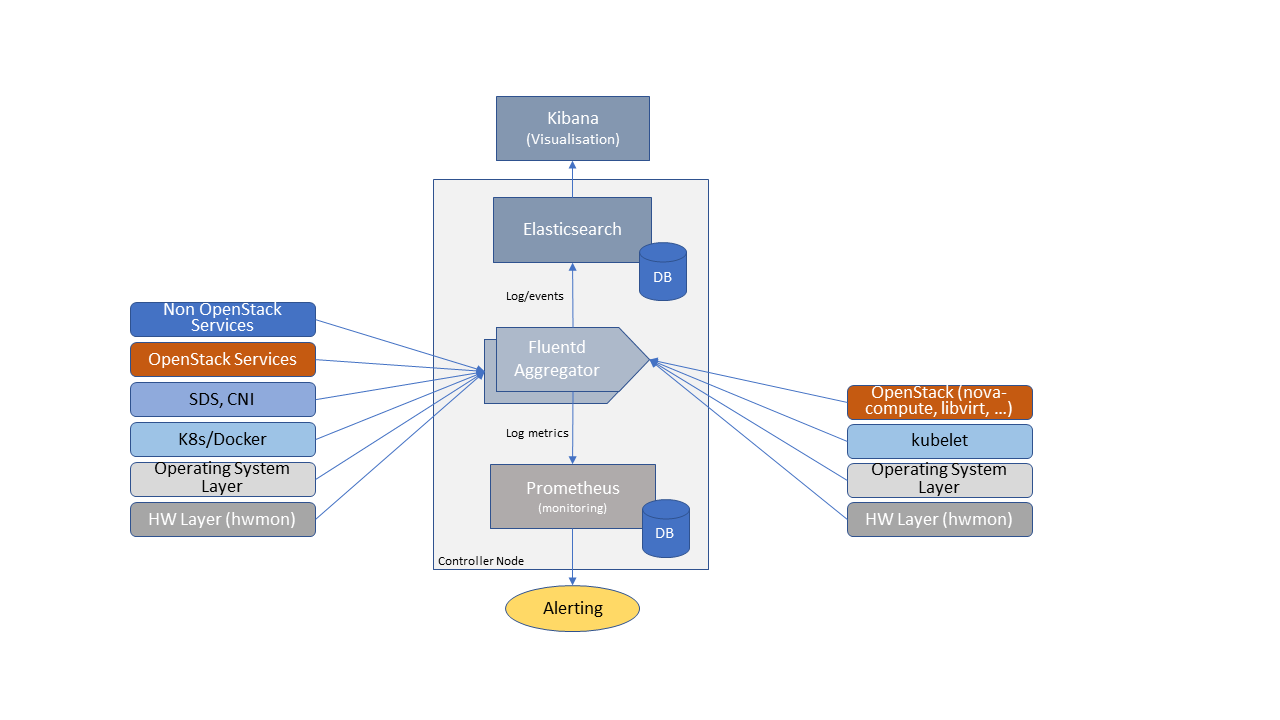


Figure 11: Monitoring and Logging Framework

The monitoring and logging framework (Figure 11) leverages Prometheus as the monitoring engine and Fluentd for logging. In addition, the framework uses Elasticsearch to store and organize 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.

# Gaps, Innovation, and Development

## Introduction

The purpose of this section 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

## The Gap

### Autoscaling

With regards to resource autoscaling (req.gen.scl.01) 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 utilization, 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 [101]. Please note that the OpenStack Senlin service is still under development with major architectural changes made in the OpenStack Ussuri release. It might be possible for the next version of this RA to recommend Senlin for auto-scaling.

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

## OpenStack Release Gaps

Section contains the APIs versions and key differences between the chosen Anuket baseline version (Train), the current version for RI (Ocata) and the potential future version for RI (Stein). The table below gives only an overview of the differences. For detailed changes, please check theOpenStack Releases [102].

|  |  |  |  |
| --- | --- | --- | --- |
| **Service Name** | **OpenStack (Ocata) (RI version)** | **OpenStack (Train) (Anuket baseline)** | **OpenStack (Stein) (potential future RI version)** |
| Keystone | 3.8 | 3.13  - Support for delegating fine-grained privileges.  - Supports the admin, member, and reader default roles across system-scope, domain-scope, and project-scope.  - Different role API uses new default policies that make it more accessible to end users and administrators in a secure way. | 3.12  - Support for project tags, application credential, domain level resource limits, JSON Web Tokens.  - Introduced system scoped roles  - Introduced new role ‘reader’ along with ‘member’ and ‘admin’ |
| Glance | 2.5 | 2.9  - Support for compressed container formats.  - Block Storage service always creates a new secret in Barbican when it uploads a volume as an image. | 2.7  - Version v1 is removed  - Support for hidden images, interoperable image import using image data (glance-direct) or image URL(web-download),  - Fixed OpenStack Security Note OSSN-0075 [103]  - Multi backend support to configure multiple stores |
| Cinder | 3.27 | 3.59 | 3.59  - Support for multi attach and deferred deletion for RBD driver  - Support for image signature verification when creating volume from image |
| Nova | 2.42 | 2.79  - Support for servers with a NUMA topology, pinned CPUs and/or huge pages, and SR-IOV ports attached when using the libvirt compute driver.  - Support for hardware-based encryption of guest memory to protect users against attackers or rogue administrators snooping on their workloads. | 2.72  - Support for vGPUs  - Support for volume type in server create API  - Support to create servers with ports that have QoS minimum bandwidth rule  - Security enhancements when using Glance signed images |
| Swift | 1.0 | 1.0 | 1.0 |
| Neutron | 2.0 | 2.0  - Support for Smart NIC in ML2/OVS mechanism driver to bind the Neutron port for the baremetal host with Smart NIC.  - Introduced support for a notifier that sends notifications on relevant resource events/changes to the Openstack Baremetal service (ironic). | 2.0 |
| Orchestration | 1.0 | 1.0 | 1.0 |
| Placement |  | 1.36 |  |

Table 89:Main differences between OpenStack releases

Additionally, Stein release also provide an upgrade check before actually upgrading any of the services. See more details on upgrade-check [104].