Anuket OpenStack based Reference Architecture

Anuket

Jun 09, 2022

Contents

1 Introduction

1.1 Overview

This Reference Architecture is focussed on OpenStack as the Virtualised Infrastructure Manager (VIM) chosen based on the criteria laid out in the Introduction. OpenStack 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) 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; hereafter, "conformant" denotes that the resource can satisfy tests conducted to verify conformance with this reference architecture.

1.2 Use Cases

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

- **Overlay networks**: The overlay functionality design includes OpenStack Networking in Open vSwitch 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 exchange of information (data, voice, multi-media) over a system's network. Some of these workloads tend to consist of a large number of small-sized packets that are short lived, such as DNS queries or SNMP traps. These messages need to arrive quickly and, thus, do not handle packet loss. Network function workloads have requirements that may affect configurations including at the hypervisor level. For an application that generates 10 TCP sessions per user with an average bandwidth of 512 kilobytes per second per flow and expected user count of ten thousand (10,000) concurrent users, the expected bandwidth plan is approximately 4.88 gigabits per second. The supporting network for this type of configuration needs to have a low latency and evenly distributed load across the topology. These types of workload benefit from having services local to the consumers of the service. Thus, use a multi-site approach, as well as, deploying many copies of the application to handle load as close as possible to consumers. Since these applications function independently, they do not warrant running overlays to interconnect tenant networks. Overlays also have the drawback of performing poorly with rapid flow setup and may incur too much overhead with large quantities of small packets and therefore we do not recommend them. QoS is desirable for some workloads to ensure delivery. DNS has a major impact on the load times of other services and needs to be reliable and provide rapid responses. Configure rules in upstream devices to apply a higher-Class Selector to DNS to ensure faster delivery or a better spot in queuing algorithms.

1.3 Anuket OpenStack Reference Release

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

1.4 Principles

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

- Anuket General Principles
- Architectural Principles

OpenStack specific principles

OpenStack considers the following Four Opens essential for success:

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

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

1.5 Document Organisation

Chapter 2 defines the Reference Architecture requirements and, when appropriate, provides references to where these requirements are addressed in this document. The intent of this document is to address all of the mandatory ("must") requirements and the most useful of the other optional ("should") requirements. Chapter 3 and 4 cover the Cloud Infrastructure resources and the core OpenStack services, while the APIs are covered in Chapter 5. Chapter 6 covers the implementation and enforcement of security capabilities and controls. Life Cycle Management of the Cloud Infrastructure and VIM are covered in Chapter 7 with stress on Logging, Monitoring and Analytics (LMA), configuration management and some other operational items. Please note that Chapter 7 is not a replacement for the implementation, configuration and operational documentation that accompanies the different OpenStack distributions. Chapter 8 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.

1.6 Terminology

General terminology definitions can be found in Glossary and specific terms relating to this reference architecture are to be found in *OpenStack Related Terminology :ref:`cntt:common/glossary:openstack related terminology*.

2 Architecture Requirements

2.1 Introduction

must: Requirements that are marked as *must* are considered mandatory and must exist in the reference architecture and reflected in any implementation targeting this reference architecture. The same applies to *must not*.

should: Requirements that are marked as *should* are expected to be fulfilled by the reference architecture but it is up to each service provider to accept an implementation targeting this reference architecture that is not reflecting on any of those requirements. The same applies to *should not*. > RFC2119

may: Requirements that are marked as *may* are considered optional. The same applies to *may not*.

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

2.2 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 Cloud Infrastructure Management
- Those relating to Cloud Infrastructure Security

Cloud Infrastructure Software Profile Requirements for Compute

(source Cloud Infrastructure Software Profiles features and requirements)

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

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
e.cap.001	Max number of vCPU that can be assigned to a single instance by the Cloud Infrastructure	At least 16	At least 16	*Compute Nodes*
e.cap.002	Max memory that can be assigned to a single instance by the Cloud Infrastructure	at least 32 GB	at least 32 GB	*Virtual Storage*
e.cap.003	Max storage that can be assigned to a single instance by the Cloud Infrastructure	at least 320 GB	at least 320 GB	*Virtual Storage* and *Storage Backend*
e.cap.004	Max number of connection points that can be assigned to a single instance by the Cloud Infrastructure	6	6	Not Detailed
e.cap.005	Max storage that can be attached / mounted to an instance by the Cloud Infrastructure	Up to 16TB:sup:*1*	Up to 16TB:sup:*1*	*Storage Backend*
e.cap.006/infra.com.cfg.003	CPU pinning support	Not required	Must support	*Consumable Infrastructure Resources and Services*
e.cap.007/infra.com.cfg.002	NUMA support	Not required	Must support	*Consumable Infrastructure Resources and Services*
e.cap.018/infra.com.cfg.005	Simultaneous Multithreading (SMT) enabled	Must	Optional support	*Consumable Infrastructure Resources and Services*
i.cap.018/infra.com.cfg.004	Huge pages configured	Not required	Must support	*Consumable Infrastructure Resources and Services*

:sup:`1` Defined in the `.bronze` configuration in Storage Extensions

Cloud Infrastructure Software Profile Extensions Requirements for Compute

Table 2.2: 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		*Acceleration*
e.cap.010/infra.com.acc.cfg.002	Transcoding Acceleration	Compute Intensive GPU	Video Transcoding	*Acceleration*
e.cap.011/infra.com.acc.cfg.003	Programmable Acceleration	Firmware-programmable adapter	Accelerator	*Acceleration*
e.cap.012	Enhanced Cache Management: L=Lean; E=Equal; X=eXpanded	E	E	Not detailed
e.cap.014/infra.com.acc.cfg.004	Hardware coprocessor support (GPU/NPU)	Compute Intensive GPU		*Acceleration*
e.cap.016/infra.com.acc.cfg.005	FPGA/other Acceleration H/W	Firmware-programmable adapter		*Acceleration*

Cloud Infrastructure Software Profile Requirements for Networking

(source Virtual Networking)

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

Table 2.3: Reference Model Requirements - Virtual Networking

Reference	Description	Requirement for Basic Profile	Requirement for High-Performance Profile	Specification Reference
infra.net.cfg.001	IO virtualisation using virtio1.1	Must support	Must support	*Virtualisation*
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	*Network Fabric*
infra.net.cfg.003	Network Address Translation	Must support	Must support	*Network Fabric*
infra.net.cfg.004	Security Groups	Must support	Must support	*Workload Security*
infra.net.cfg.005	SFC support	Not required	Must support	*Virtual Networking – 3rd party SDN solution*
infra.net.cfg.006	Traffic patterns symmetry	Must support	Must support	Not detailed

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

Table 2.4: Reference Model Requirements - Network Interface Specifications

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

Cloud Infrastructure Software Profile Extensions Requirements for Networking

Table 2.5: Cloud Infrastructure Software Profile Extensions Requirements for Networking

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

Cloud Infrastructure Software Profile Requirements for Storage

(source Cloud Infrastructure Software Profiles features and requirements)

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

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

Cloud Infrastructure Software Profile Extensions Requirements for Storage

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

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

Cloud Infrastructure Hardware Profile Requirements

(source Cloud Infrastructure Hardware Profiles features and requirements.)

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

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

Cloud Infrastructure Hardware Profile-Extensions Requirements

(source Cloud Infrastructure Hardware Profiles features and requirements.)

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

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

Cloud Infrastructure Management Requirements

(source Cloud Infrastructure Management Capabilities)

Table 2.10: Reference Model Requirements - Cloud Infrastructure Management Requirements

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

Cloud Infrastructure Security Requirements

System Hardening Requirements

(source System Hardening)

Table 2.11: Reference Model Requirements - System Hardening Requirements

Reference	sub-category	Description	Specification Reference
sec.gen.001	Hardening	The Platform **must** maintain the specified configuration	*Security LCM* and *Cloud Infrastructure provisioning and configuration management*
sec.gen.002	Hardening	All systems part of Cloud Infrastructure **must** support hardening as defined in CIS Password Policy Guide	*Password policy*
sec.gen.003	Hardening	All servers part of Cloud Infrastructure **must** support a root of trust and secure boot	*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*
sec.gen.005	Hardening	The Platform **must** support Operating System level access control	*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*
sec.gen.007	Hardening	All servers part of Cloud Infrastructure **must** be Time synchronised with authenticated Time service	*Security Logs Time Synchronisation*
sec.gen.008	Hardening	All servers part of Cloud Infrastructure **must** be regularly updated to address security vulnerabilities	*Security LCM*
sec.gen.009	Hardening	The Platform **must** support software integrity protection and verification	*Integrity of OpenStack components configuration*
sec.gen.010	Hardening	The Cloud Infrastructure **must** support encrypted storage, for example, block, object and file storage, with access to encryption keys restricted based on a need to know (Controlled Access Based on the Need to Know)	*Confidentiality and Integrity*
sec.gen.012	Hardening	The Operator **must** ensure that only authorised actors have physical access to the underlying infrastructure	This requirement's verification must be part of the organisation security process
sec.gen.013	Hardening	The Platform **must** ensure that only authorised actors have logical access to the underlying infrastructure	*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*

Platform and Access Requirements

(source Platform and Access)

Table 2.12: Reference Model Requirements - Platform and Access Requirements

Reference	sub-category	Description	Specification Reference
sec.sys.001	Access	The Platform **must** support authenticated and secure access to API, GUI and command line interfaces	*RBAC*
sec.sys.002	Access	The Platform **must** support Traffic Filtering for workloads (for example, Firewall)	*Workload Security*
sec.sys.003	Access	The Platform **must** support Secure and encrypted communications, and confidentiality and integrity of network	*Confidentiality and Integrity*
sec.sys.004	Access	The Cloud Infrastructure **must** support authentication, integrity and confidentiality on all network channels	*Confidentiality and Integrity*
sec.sys.005	Access	The Cloud Infrastructure **must** segregate the underlay and overlay networks	*Confidentiality and Integrity*
sec.sys.006	Access	The Cloud Infrastructure **must** be able to utilise the Cloud Infrastructure Manager identity lifecycle management capabilities	*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*
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*
sec.sys.009	Access	The Platform **must** support creation of Trust Relationships between trust domains. These maybe uni-directional relationships where the trusting domain trusts another domain (the "trusted domain") to authenticate users for them them or to allow access to its resources from the trusted domain. In a bidirectional relationship both domain are "trusting" and "trusted"	*Logical segregation and high availability*
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	*Logical segregation and high availability*
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*
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)	*Barbican*
sec.sys.013	Access	The Platform **must** generate secrets dynamically as and when needed	*Barbican*
sec.sys.015	Access	The Platform **must not** contain back door	Not detailed

Confidentiality and Integrity Requirements

(source Confidentiality and Integrity)

Table 2.13: Reference Model Requirements - Confidentiality and Integrity Requirements

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

Workload Security Requirements

(source Workload security requirements)

Table 2.14: Reference Model Requirements - Workload Security Requirements

Reference	sub-category	Description	Specification Reference
sec.wl.001	Workload	The Platform **must** support Workload placement policy	*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)	*Cloud Infrastructure and VIM Security*
sec.wl.003	Workload	The Platform **must** support secure provisioning of Workloads	*Workload Security*
sec.wl.004	Workload	The Platform **must** support Location assertion (for mandated in- country or location requirements)	*Workload Security*
sec.wl.005	Workload	The Platform **must** support the separation of production and non- production Workloads	*Workload Security*
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*
sec.wl.007	Workload	The Operator **must** implement processes and tools to verify NF authenticity and integrity	*Image Security*

Image Security Requirements

(source Image Security)

Table 2.15: Reference Model Requirements - Image Security Requirements

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

Security LCM Requirements

(source Security LCM)

Table 2.16: Reference Model Requirements - Security LCM Requirements

Reference	sub-category	Description	Specification Reference
sec.lcm.001	LCM	The Platform **must** support Secure Provisioning, Availability, and Deprovisioning (Secure Clean-Up) of workload resources where Secure Clean-Up includes tear-down, defense against virus or other attacks	*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*
sec.lcm.003	LCM	The Cloud Operator **must** implement and strictly follow change management processes for Cloud Infrastructure, Infrastructure Manager and othercomponents of the cloud, and Platform change control on hardware	*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*
sec.lcm.006	LCM	The Platform **must** verify the integrity of all Resource management requests	*Confidentiality and Integrity of tenant data (sec.ci.001)*
sec.lcm.007	LCM	The Platform **must** be able to update newly instantiated, suspended, hibernated, migrated and restarted images with current time information	Not detailed
sec.lcm.008	LCM	The Platform **must** be able to update newly instantiated, suspended, hibernated, migrated and restarted images with relevant DNS information	Not detailed
sec.lcm.009	LCM	The Platform **must** be able to update the tag of newly instantiated, suspended, hibernated, migrated and restarted images with relevant geolocation (geographical) information	Not detailed
sec.lcm.010	LCM	The Platform **must** log all changes to geolocation along with the mechanisms and sources of location information (i.e. GPS, IP block, and timing)	Not detailed
sec.lcm.011	LCM	The Platform **must** implement Security life cycle management processes including the proactive update and patching of all deployed Cloud Infrastructure software	*Patches*
sec.lcm.012	LCM	The Platform **must** log any access privilege escalation	*What to Log / What NOT to Log*

Monitoring and Security Audit Requirements

(source Monitoring and Security Audit)

The Platform is assumed to provide configurable alerting and notification capability and the operator is assumed to have automated systems, policies and procedures to act on alerts and notifications in a timely fashion. In the following the monitoring and logging capabilities can trigger alerts and notifications for appropriate action.

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

Reference	sub-category	Description	Specification Reference
sec.mon.001	Monitoring/Audit	Platform **must** provide logs and these logs must be regularly monitored for events of interest. The logs must contain the following fields: event type, date/time, protocol, service or program used for access, success/failure, login ID or process ID, IP address and ports (source and destination) involved	*Required Fields*
sec.mon.002	Monitoring	Security logs **must** be time synchronised	*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*
sec.mon.004	Audit	The Platform **must** secure and protect Audit logs (containing sensitive information) both in-transit and at rest	*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 accordingly actions	*What to Log / What NOT to Log*
sec.mon.006	Monitoring/Audit	The Platform **must** Monitor and Audit operations by authorised account access after login to detect malicious operational activity and take corrective actions	*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*
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)*
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	*Monitoring and Security Audit*
sec.mon.010	Monitoring/Audit	The Platform **must** Monitor and Audit running processes to detect unexpected or unauthorised processes and take corrective actions accordingly	*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*
sec.mon.012	Monitoring/Audit	The Platform **must** Monitor and Audit Traffic patterns and volumes to prevent malware download attempts	*Confidentiality and Integrity*
sec.mon.013	Monitoring	The monitoring system **must not** affect the security (integrity and confidentiality) of the infrastructure, workloads, or the user data (through back door entries)	Not detailed
sec.mon.015	Monitoring	The Platform **must** ensure that the Monitoring systems are never starved of resources and must activate alarms when	*Monitoring and Security Audit*

Open-Source Software Security Requirements

(source Open-Source Software Security)

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

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

IaaC security Requirements

(source IaaC - Secure Design and Architecture Stage Requirements)

Secure Code Stage Requirements

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

Reference	sub-category	Description	Specification Reference
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 bestpractices. Example: open source OWASP range of tools	*Workload Security*

Continuous Build, Integration and Testing Stage Requirements

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

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

Continuous Delivery and Deployment Stage Requirements

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

Reference	sub-category	Description	Specification Reference
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	*Image Security*
sec.del.002	IaaC	Code Signing **must** be applied during the Continuous Deliveryand Deployment stage and Image Repository trigger. Code Signing provides authentication to assure that downloaded files are form the publisher named on the certificate	*Image Security*
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)	*Image Security*

Runtime Defence and Monitoring Requirements

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

Reference	sub-category	Description	Specification Reference
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 forsecurity risks	Not detailed

Compliance with Standards Requirements

(source Compliance with Standards)

Table 2.23: Reference Model Requirements: Compliance with Standards

Reference	sub-category	Description	Specification Reference
sec.std.012	Standards	The Public Cloud Operator **must**, and the Private Cloud Operator **may** be certified to be compliant with the International Standard on Awareness Engagements (ISAE) 3402 (in the US:SSAE 16); International Standard on Awareness Engagements (ISAE) 3402. US Equivalent: SSAE16	Not detailed

2.3 Architecture and OpenStack Requirements

"Architecture" in this chapter refers to Cloud Infrastructure (referred to as NFVI by ETSI) + VIM (as specified in Reference Model Chapter 3).

General Requirements

Table 2.24: General Requirements

Reference	sub-category	Description	Specification Reference
gen.ost.01	Open source	The Architecture **must** use OpenStack APIs	*Consolidated Set of APIs*
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	*Containerised OpenStack Services*
gen.avl.01	Availability	The Architecture **must** provide High Availability for OpenStack components	*Underlying Resources*

Infrastructure Requirements

Table 2.25: Infrastructure Requirements

Reference	sub-category	Description	Specification Reference
inf.com.01	Compute	The Architecture **must** provide compute resources for instances	*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*
inf.com.05	Compute	The Architecture **must** support Hardware Platforms with NUMA capabilities	*Support for Cloud Infrastructure Profiles and flavors*
inf.com.06	Compute	The Architecture **must** support CPU Pinning of the vCPUs of an instance	*Support for Cloud Infrastructure Profiles and flavors*
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*
inf.com.08	Compute	The Architecture **must** support allocating certain number of host cores for all non-tenant workloads such as for OpenStack services. SMT threads can be allocated to individual OpenStack services or their components. Dedicating host cores to certain workloads (e.g., OpenStack services). Please see example, Configuring libvirt compute nodes for CPU pinning	*Cloud partitioning: Host Aggregates, Availability Zones*
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	*Pinned and Unpinned CPUs*
inf.stg.01	Storage	The Architecture **must** provide remote (not directly attached to the host) Block storage for Instances	*Storage*
inf.stg.02	Storage	The Architecture **must** provide Object storage for Instances. Operators **may** choose not to implement Object Storage but must be cognizant of the the risk of “Compliant VNFs” failing in their environment	*Swift*
inf.nw.01	Network	The Architecture **must** provide virtual network interfaces to instances	*Neutron*
inf.nw.02	Network	The Architecture **must** include capabilities for integrating SDN controllers to support provisioning of network services, from the SDN OpenStack Neutron service, such as networking of VTEPs to the Border Edge based VRFs	*Virtual Networking – 3rd party SDN solution*
inf.nw.03	Network	The Architecture **must** support low latency and high throughput traffic needs	*Network Fabric*
inf.nw.05	Network	The Architecture **must** allow for East/West tenant traffic within the cloud (via tunnelled encapsulation overlay such as VXLAN or Geneve)	*Network Fabric*
inf.nw.07	Network	The Architecture must support network resiliency	*Network*
inf.nw.10	Network	The Cloud Infrastructure Network Fabric **must** be capable of enabling highly available	*Network*

VIM Requirements

Table 2.26: VIM Requirements

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

Interfaces & APIs Requirements

Table 2.27: Interfaces and APIs Requirements

Reference	sub-category	Description	Specification Reference
int.api.01	API	The Architecture **must** provide APIs to access the authentication service and the associated mandatory features detailed in chapter 5	*Keystone*
int.api.02	API	The Architecture **must** provide APIs to access the image management service and the associated mandatory features detailed in chapter 5	*Glance*
int.api.03	API	The Architecture **must** provide APIs to access the block storage management service and the associated mandatory features detailed in chapter 5	*Cinder*
int.api.04	API	The Architecture **must** provide APIs to access the object storage management service and the associated mandatory features detailed in chapter 5	*Swift*
int.api.05	API	The Architecture **must** provide APIs to access the network management service and the associated mandatory features detailed in chapter 5	*Neutron*
int.api.06	API	The Architecture **must** provide APIs to access the compute resources management service and the associated mandatory features detailed in chapter 5	*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*
int.api.08	API	The Architecture **must** provide APIs needed to discover and manage Cloud Infrastructure resources	*Placement*
int.api.09	API	The Architecture **must** provide APIs to access the orchestration service	*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*

Tenant Requirements

Table 2.28: Tenant Requirements

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

Operations and LCM

Table 2.29: LCM Requirements

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

Assurance Requirements

Table 2.30: Assurance Requirements

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

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

Table 2.31: General Recommendations

Reference	sub-category	Description	Notes
gen.cnt.01	Cloud nativeness	The Architecture **should** consist of stateless service components. However, where state is required it must be kept external to the component	OpenStack consists of both stateless and stateful services where the stateful services utilise a database. For latter see Configuring the stateful services
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, cluster management service
gen.rsl.02	Resiliency	The Architecture **should** support resilient OpenStack service components that are not subject to gen.rsl.01	

Infrastructure Recommendations

Table 2.32: Infrastructure Recommendations

Reference	sub-category	Description	Notes
inf.com.02	Compute	The Architecture **should** include industry standard hardware management systems at both HW device level (embedded) and HW platform level (external to device)	
inf.com.03	Compute	The Architecture **should** support Symmetric Multiprocessing with shared memory access as well as Simultaneous Multithreading	
inf.stg.08	Storage	The Architecture **should** allow use of externally provided large archival storage for its Backup / Restore / Archival needs	
inf.stg.09	Storage	The Architecture **should** make available all non-host OS / Hypervisor / Host systems storage as network-based Block, File or Object Storage for tenant/management consumption	
inf.stg.10	Storage	The Architecture **should** provide local Block storage for Instances	*Virtual Storage*
inf.nw.04	Network	The Architecture **should** support service function chaining	
inf.nw.06	Network	The Architecture **should** support Distributed Virtual Routing (DVR) to allow compute nodes to route traffic efficiently	
inf.nw.08	Network	The Cloud Infrastructure Network Fabric **should** embrace the concepts of open networking and disaggregation using commodity networking hardware and disaggregated Network Operating Systems	
inf.nw.09	Network	The Cloud Infrastructure Network Fabric **should** embrace open-based standards and technologies	
inf.nw.11	Network	The Cloud Infrastructure Network Fabric **should** be architected to provide a standardised, scalable, and repeatable deployment model across all applicable Cloud Infrastructure sites	
inf.nw.17	Network	The Architecture **should** use dual stack IPv4 and IPv6 for Cloud Infrastructure internal networks	
inf.acc.01	Acceleration	The Architecture **should** support Application Specific Acceleration (exposed to VNFs)	*Acceleration*
inf.acc.02	Acceleration	The Architecture **should** support Cloud Infrastructure Acceleration (such as SmartNICs)	OpenStack Future - Specs defined
inf.acc.03	Acceleration	The Architecture **may** rely on on SR-IOV PCI-Pass through to provide acceleration to VNFs	
inf.img.01	Image	The Architecture **should** make the immutable images available via location independent means	*Glance*

VIM Recommendations

Table 2.33: VIM Recommendations

Reference	sub-category	Description	Notes
vim.02	General	The Architecture **should** support deployment of OpenStack components in containers	*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	

Interfaces and APIs Recommendations

Table 2.34: Interfaces and APIs Recommendations

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

Tenant Recommendations

This section is left blank for future use.

Table 2.35: Tenant Recommendations

Reference	sub-category	Description	Notes

Operations and LCM Recommendations

Table 2.36: LCM Recommendations

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

Assurance Recommendations

Table 2.37: Assurance Recommendations

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

Security Recommendations

System Hardening Recommendations

(source System Hardening)

Table 2.38: System Hardening Recommendations

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

Platform and Access Recommendations

(source Platform and Access)

Table 2.39: Platform and Access Recommendations

Reference	sub-category	Description	Notes
sec.sys.014	Access	The Platform **should** use Linux Security Modules such as SELinux to control access to resources	
sec.sys.020	Access	The Cloud Infrastructure architecture **should** rely on Zero Trust principles to build a secure by design environment	Zero Trust Architecture (ZTA) described in NIST SP 800-207

Confidentiality and Integrity Recommendations

(source Confidentiality and Integrity)

Table 2.40: Confidentiality and Integrity Recommendations

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

Workload Security Recommendations

(source Workload Security)

Table 2.41: Workload Security Recommendations

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

Image Security Recommendations

(source Image Security)

This section is left blank for future use.

Table 2.42: Image Security Recommendations

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

Security LCM Recommendations

(source Security LCM)

Table 2.43: LCM Security Recommendations

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

Monitoring and Security Audit Recommendations

(source Monitoring and Security Audit)

The Platform is assumed to provide configurable alerting and notification capability and the operator is assumed to have automated systems, policies and procedures to act on alerts and notifications in a timely fashion. In the following the monitoring and logging capabilities can trigger alerts and notifications for appropriate action.

Table 2.44: Monitoring and Security Audit Recommendations

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

Open-Source Software Security Recommendations

(source Open Source Software)

Table 2.45: Open-Source Software Security Recommendations

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

IaaC security Recommendations

(source IaaC - Secure Design and Architecture Stage Requirements)

Secure Design and Architecture Stage

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

Reference	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

Secure Code Stage Recommendations

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

Reference	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	

Continuous Build, Integration and Testing Stage Recommendations

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

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

Continuous Delivery and Deployment Stage Recommendations

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

Reference	sub-category	Description	Notes
sec.del.003	IaaC	Artifact and Image Repository Scan **should** be continuously applied during the Continuous Delivery and Deployment stage	Example: GitLab uses the open source Clair engine for container scanning

Runtime Defence and Monitoring Recommendations

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

Reference	sub-category	Description	Notes
sec.run.002	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

Compliance with Standards Recommendations

(source Compliance with Standards)

Table 2.51: Compliance with Security Recommendations

Reference	sub-category	Description	Notes
sec.std.001	Standards	The Cloud Operator **should** comply with Center for Internet Security CIS Controls	
sec.std.002	Standards	The Cloud Operator, Platform and Workloads **should** follow the guidance in the CSA Security Guidance for Critical Areas of Focus in Cloud Computing (latest version)- CSA, Cloud Security Alliance	
sec.std.003	Standards	The Platform and Workloads **should** follow the guidance in the OWASP Cheat Sheet Series (OCSS) - OWASP, Open Web Application Security Project	
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	
sec.std.005	Standards	The Cloud Operator, Platform and Workloads **should** strive to improve their maturity on the OWASP Software Maturity Model (SAMM)	
sec.std.006	Standards	The Cloud Operator, Platform and Workloads should utilise the OWASP Web Security Testing Guide	
sec.std.007	Standards	The Cloud Operator, and Platform **should** satisfy the requirements for Information Management Systems specified in ISO/IEC 27001; 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)	
sec.std.009	Standards	The Cloud Operator, and Platform **should** implement the ISO/IEC 27032:2012 (or latest) Guidelines for Cybersecurity techniques; ISO/IEC 27032 is the international Standard focusing explicitly on cybersecurity	
sec.std.010	Standards	The Cloud Operator **should** conform to the ISO/IEC 27035 standard for incidence management; ISO/IEC 27035 is the international Standard for incident management	
sec.std.011	Standards	The Cloud Operator **should** conform to the ISO/IEC 27031 standard for business continuity; ISO/IEC 27031 - ISO/IEC 27031 is the international Standard for ICT readiness for business continuity	

3 Cloud Infrastructure Architecture - OpenStack

3.1 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 chapter is to outline all the components required to provide the Cloud Infrastructure (NFVI and the VIM) in a consistent and reliable way.

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

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

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

This chapter is organised as follows:

- Consumable Infrastructure Resources and Services: these are infrastructure services and resources being exposed northbound for consumption
  - Multi-tenancy with quotas
    - Virtual compute: vCPU / vRAM
    - Virtual storage: Ephemeral, Persistent and Image
    - Virtual networking – neutron standalone: network plugin, virtual switch, accelerator features
    - Virtual networking – 3rd party SDN solution
    - Additional network services: Firewall, DC Gateway
- Cloud Infrastructure Management Software (VIM): is how we manage the Consumable Infrastructure Resources and Services
  - VIM Core services (keystone, cinder, nova, neutron etc.)
    - Tenant Separation
    - Host aggregates providing resource pooling
    - 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” is used in this document in the OpenStack context including

when specifying configurations; the OpenStack term flavor includes the profile configuration information as "extra specs".

3.2 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 the Cloud Infrastructure and with external networks

Multi-Tenancy (execution environment)

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

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

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

Virtual Compute (vCPU and vRAM)

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

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

Virtual Storage

The Reference Model Storage for Tenant Consumption details consumption models for tenants: Platform native, object storage, shared file storage and archival. The choice of a solution will depend on the storage use case needs.

The two storage services offered by Cloud Infrastructure are:

- Persistent storage
- Ephemeral storage

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

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

Three types of persistent data storage are supported in OpenStack:

- Block storage
- Object storage
- Shared file systems storage

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

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

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

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

Cinder, Swift, and Glance services are discussed in the section *Virtualised Infrastructure Manager (VIM)*.

Virtual Networking Neutron standalone

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

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

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

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

- Highly scalable, multi-tenant networking
- Multi-tenant IP address management
- DHCP, ARP proxies to avoid flooding into networks
- Efficient edge replication for broadcast and multicast traffic
- Local, per-tenant DNS resolution
- Distributed firewall with access control lists
- Application-based security policies
- Distributed load balancing across hosts
- Network address translation (1:1 floating IPs and distributed SNAT)
- Service chaining with virtual network functions
- Dual stack IPv4 and IPv6
- BGP peering with gateway routers
- BGP as a Service (BGPaaS) for distribution of routes between privately managed customer networks and service provider networks

Based on the network layering concepts introduced in the Reference Model Section Network, 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 specialised hardware to perform some function faster than is possible by executing the same function on a general-purpose CPU or on a traditional networking (or other I/O) device (e.g., NIC, switch, storage controller, etc.). The hardware accelerator covers the options for ASICs, SmartNIC, FPGAs, GPU etc. to offload the main CPU, and to accelerate workload performance. Cloud Infrastructure should manage the accelerators by plugins and provide the acceleration capabilities to VNFs.

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

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

The architecture consists of the core services shown in the Fig. 3.1; 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.

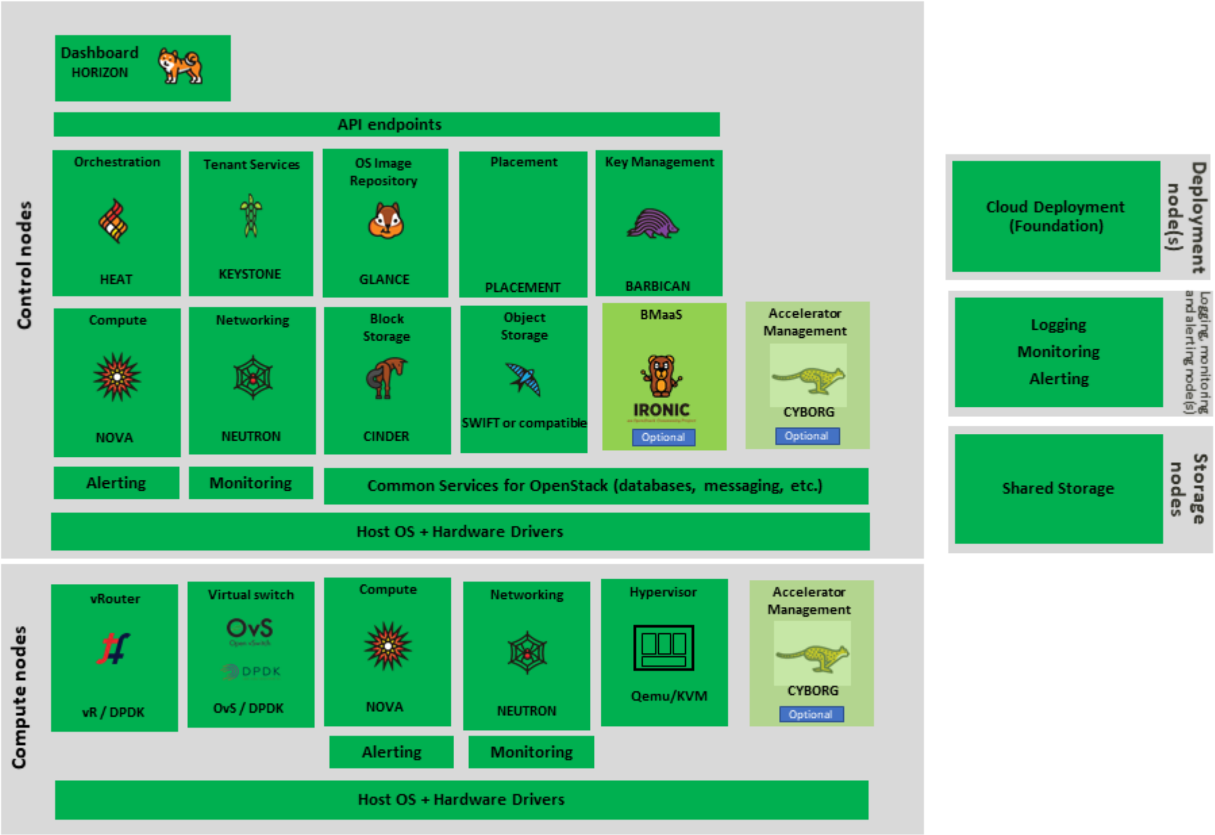

Fig. 3.1: OpenStack Core Services

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. Fig. 3.2 depicts the 4 types of nodes constitutive of the Infrastructure: control, compute, network and storage nodes.

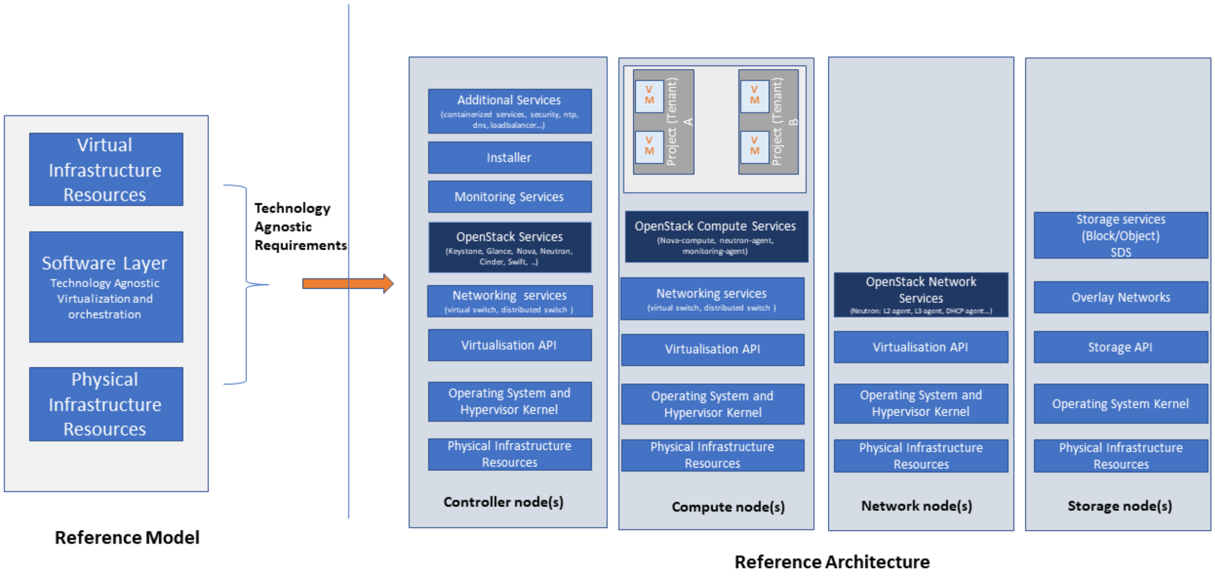
Virtual Infrastructure Resources
Software Layer
Technology Agnostic Virtualization and orchestration
Physical Infrastructure Resources
Reference Model
Technology Agnostic Requirements
Additional Services
(containerized services, security, ntp, dns, loadbalancer...)
Installer
Monitoring Services
OpenStack Services
(Keystone, Glance, Nova, Neutron, Cinder, Swift, ...)
Networking services
(virtual switch, distributed switch)
Virtualisation API
Operating System and Hypervisor Kernel
Physical Infrastructure Resources
Controller node(s)
VM
VM
Project (Tenant) A
VM
VM
Project (Tenant) B
OpenStack Compute Services
(Nova-compute, neutron-agent, monitoring-agent)
Networking services
(virtual switch, distributed switch)
Virtualisation API
Operating System and Hypervisor Kernel
Physical Infrastructure Resources
Compute node(s)
OpenStack Network Services
(Neutron: L2 agent, L3 agent, DHCP agent...)
Virtualisation API
Operating System and Hypervisor Kernel
Physical Infrastructure Resources
Network node(s)
Storage services (Block/Object) SDS
Overlay Networks
Storage API
Operating System Kernel
Physical Infrastructure Resources
Storage node(s)
Reference Architecture

Fig. 3.2: OpenStack Services Topology

Deployments can be structured using the distribution of services amongst the 4 node types as depicted in Fig. 3.2, 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 below also maps the OpenStack core services to the Reference Model (RM) Virtual Infrastructure Manager.

Table 3.1: OpenStack components deployment

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

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

The services can be containerised or VM hosted as long as they provide the high availability principles described above.

The APIs for these OpenStack services are listed in *Interfaces and APIs*.

Cloud Workload Services

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

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

- Local Storage (Ephemeral, Root, etc.)
- Attached Storage (using Local drivers)

Tenant Isolation

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

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

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 seggregate workloads.

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

Flavor management

In OpenStack a flavor defines the compute, memory, and storage capacity of nova instances. When instances are spawned, they are mapped to flavors which define the available hardware configuration for them. For simplicity, operators may create named flavors specifying both the sizing and the software and hardware profile configurations.

3.4 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 Hypervisor Feature Support Matrix. OpenStack (Nova) allows the use of various hypervisors within a single installation by means of scheduler filters like ComputeFilter, ImagePropertiesFilter etc.

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

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

Physical Infrastructure

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

- Servers
  - Compute
  - Storage
  - Control (min 3 for Core DC)
- Network considerations
  - Data centre gateway
  - Firewall (around the control plane, storage, etc.)
  - Data centre network fabric / Clos (spine/leaf) – Horizontal scale
  - Storage networking, control plane and data plane
  - Raw packet – tenant networking allowing “wild west” connection
- Storage
  - Storage technologies are multiple, they are extensively described in Storage Implementation Stereotypes. Storage backends are discussed in *Storage Backend*.
- 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 in Reference Model chapters Cloud Infrastructure Hardware Profile description and Cloud Infrastructure Hardware Profiles features and requirements..

Network

The recommended network architecture is spine and leaf topology.

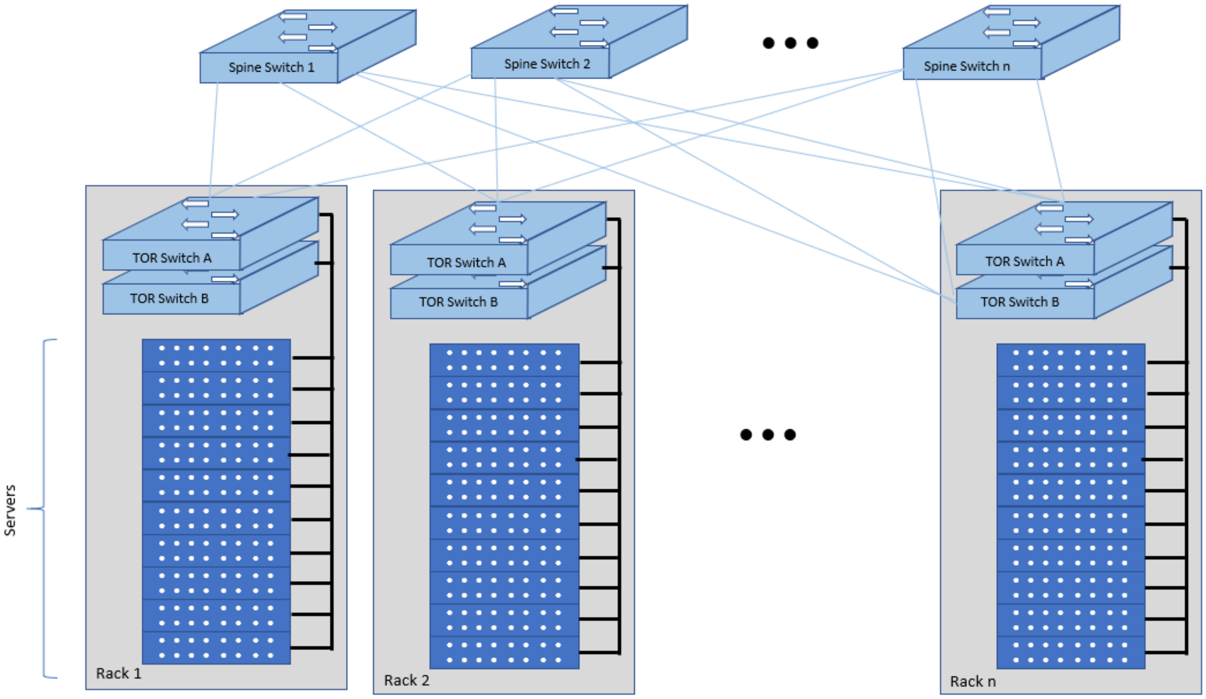

Fig. 3.3: Network Fabric – Physical

Fig. 3.3 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 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.

3.5 Cloud Topology

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

Some examples of such topologies include:

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

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

Topology Overview

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

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

Assumptions and conventions:

- Region is represented by a single OpenStack control plane.
- Resource Failure Domain is effectively the "blast radius" of any major infrastructure failure such as loss of PDU or network leafs.
- Control plane includes redundant network nodes where OVS-kernel is used.
- Controller nodes should be setup for high availability based on quorum (aligned with vendor implementation).
- Shared storage is optional, but it is important to ensure shared assets are distributed across serving clouds such as boot images. Storage needs, per deployment and use cases, can be found in Storage Scenarios and Architecture Fit.

Table 3.2: Cloud Topology: Redundancy Models

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

Topology Overview

Topology 1 - Local Redundancy

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

Important to consider:

- Where possible servers should be distributed and cabled to reduce the impact of any failure e.g., PDU, rack failure. Because each operator has individual site constraints this document will not propose a standard rack layout.
- During maintenance of the control plane, whilst the data (forwarding) plane remains unaffected, the control plane API may not be available and some applications may be relying on it during normal application operation for example for scaling. Additionally if the upgrade involves updating OpenStack services on the compute nodes care needs to be taken. OVS-kernel networking operations may also be impacted during this time.
- During maintenance of storage (e.g., ceph) there is an increased risk of a service-impacting failure, so it is generally recommended to deploy at least one more server than the minimum required for redundancy.

Topology 2 - Regional Redundancy

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

Important to consider:

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

Topology 3 - Global Redundancy

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

4 Cloud Infrastructure & VIM Component Level Architecture

4.1 Introduction

Chapter 3 introduced the components of an OpenStack-based IaaS

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

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

4.2 Underlying Resources

Virtualisation

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

- Configuration: OpenStack specifies the steps/instructions to configure KVM:
  - Enable KVM based hardware virtualisation in BIOS. OpenStack provides instructions on how to enable hardware virtualisation for different hardware platforms (x86, Power)

        - ∗ QEMU is similar to KVM in that both are libvirt controlled, have the same feature set and utilise compatible virtual machine images
    - Configure Compute backing storage
    - Specify the CPU Model for KVM guests (VMs)
    - KVM Performance Tweaks
- Hardening the virtualisation layers
    - 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 in Cloud Infrastructure Hardware Profiles features and requirements.. Additionally, for OpenStack we need to set the following boot parameters:

Table 4.1: Boot parameters

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

- How many nodes to meet SLA
    - Minimum 3 nodes for high availability
- HW specifications
    - Boot disks are dedicated with Flash technology disks
- Sizing rules
    - It is easy to horizontally scale the number of control nodes
    - The number of control nodes is determined by a minimum number needed for high availability (viz., 3 nodes) and the extra nodes needed to handle the transaction volumes, in particular, for Messaging service (e.g., RabbitMQ) and Database (e.g., MySQL) to track state.
    - The number of control nodes only needs to be increased in environments with a lot of changes, such as a testing lab, or a very large cloud footprint (rule of thumb: number of control nodes = 3 + quotient(number of compute nodes/1000)).
    - The Services Placement Summary table specifies the number of instances that are required based upon the cloud size (number of nodes).

Network nodes

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

- BIOS requirements

Table 4.2: BIOS requirements

BIOS/boot Parameter	Value
Boot disks	RAID 1

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

Storage nodes

- BIOS requirements

Table 4.3: BIOS requirements

BIOS/boot Parameter	Value
Boot disks	RAID 1

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

Compute Nodes

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

- The software and hardware configurations are as specified in the Cloud Infrastructure Hardware Profiles features and requirements.
- BIOS requirement

- The general BIOS requirements are described in the Cloud Infrastructure Hardware Profiles features and requirements.

Example Profiles and their Extensions

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

Table 4.4: Profile Extensions and Capabilities

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

BIOS Settings

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

Table 4.5: BIOS requirements

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

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

Table 4.6: DPDK usage

Layer	Description
Cloud infrastructure	Important is placement of NICs to get NUMA-balanced system (balancing the I/O, memory, and storage across both sockets), and configuration of NIC features. Server BIOS and Host OS kernel command line settings are described in DPDK release notes and DPDK performance reports. 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 inteconnect between CPUs), and DPDK Poll Mode Drivers (PMD, on reserved cores) to get the best performance. DPDK versions xx.11 are Long-Term Support maintained stable release with back-ported bug fixes for a two-year period.

- Sizing rules

Table 4.7: Mnemonic

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

Table 4.8: Sizing rules

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

Caveats:

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

Compute Resource Pooling Considerations

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

Reservation of Compute Node Cores

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

reserved_host_cpus = n

where n is any positive integer.

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

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

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

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

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

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

Pinned and Unpinned CPUs

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

openstack flavor set .xlarge –property hw:cpu_policy=dedicated

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

Compute node configurations for Profiles and OpenStack Flavors

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

Cloud Infrastructure Hardware Profile

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

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

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

CPU Allocation Ratio and CPU Pinning

A given host (compute node) can only support a single CPU Allocation Ratio. Thus, to support the B1 and B4 Basic profile extensions (Section 4.2.2.5) with CPU Allocation Ratios of 1.0 and 4.0 we will need to create 2 different host profiles and separate host aggregates for each of the host profiles. The CPU Allocation Ratio is set in the hypervisor on the host.

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

Server Configurations

The different networking choices – OVS-Kernel, OVS-DPDK, SR-IOV – result in different NIC port, LAG (Link Aggregation Group), and other configurations. Some of these are shown diagrammatically in section 4.2.9.5.

Leaf and Compute Ports for Server Flavors must align

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

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

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

Server Flavors: B1, B4, HV, HD

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

Server Flavors: HS

Notes on SR-IOV

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

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

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

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

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

Example Host Configurations

Host configurations for B1, B4 Profile Extensions

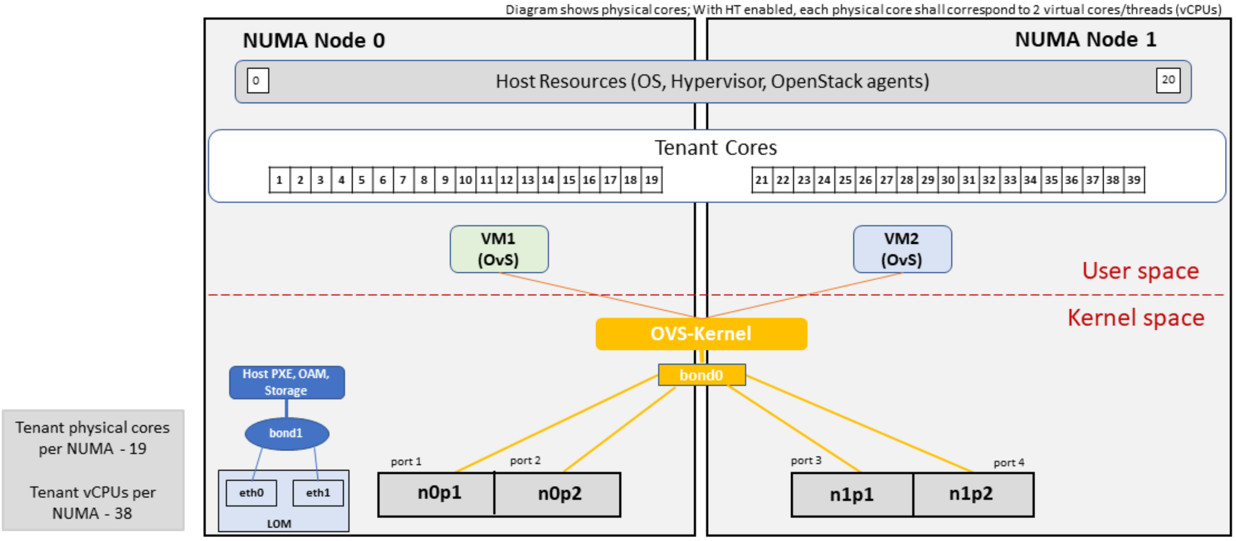

Fig. 4.1: Basic Profile Host Configuration (example and simplified)

Let us refer to the data traffic networking configuration of Fig. 4.1 to be part of the hp-B1-a and hp-B4-a host profiles and this requires the configurations as Table *Configuration of Basic Flavor Capabilities*.

Table 4.9: Configuration of Basic Flavor Capabilities

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

Fig. 4.2 shows the networking configuration where the storage and OAM share networking but are independent of the PXE network.

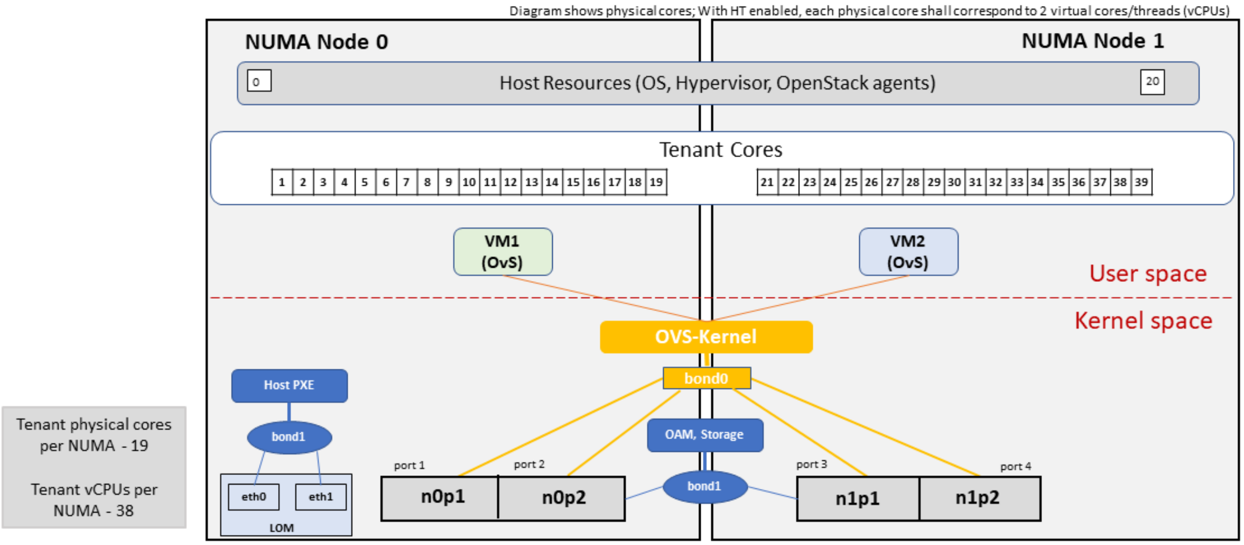

Fig. 4.2: Basic Profile Host Configuration with shared Storage and OAM networking (example and simplified)

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

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

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

Table 4.10: GRUB Configuration of Basic Profile with shared Storage

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

Host configuration for HV Profile Extensions

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

Table 4.11: Configuration of High Performance Flavor Capabilities

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

Host Networking configuration for HD Profile Extensions

An example of the data traffic configuration for the HD (OVS-DPDK) Profile Extensions is shown in Fig. 4.3.

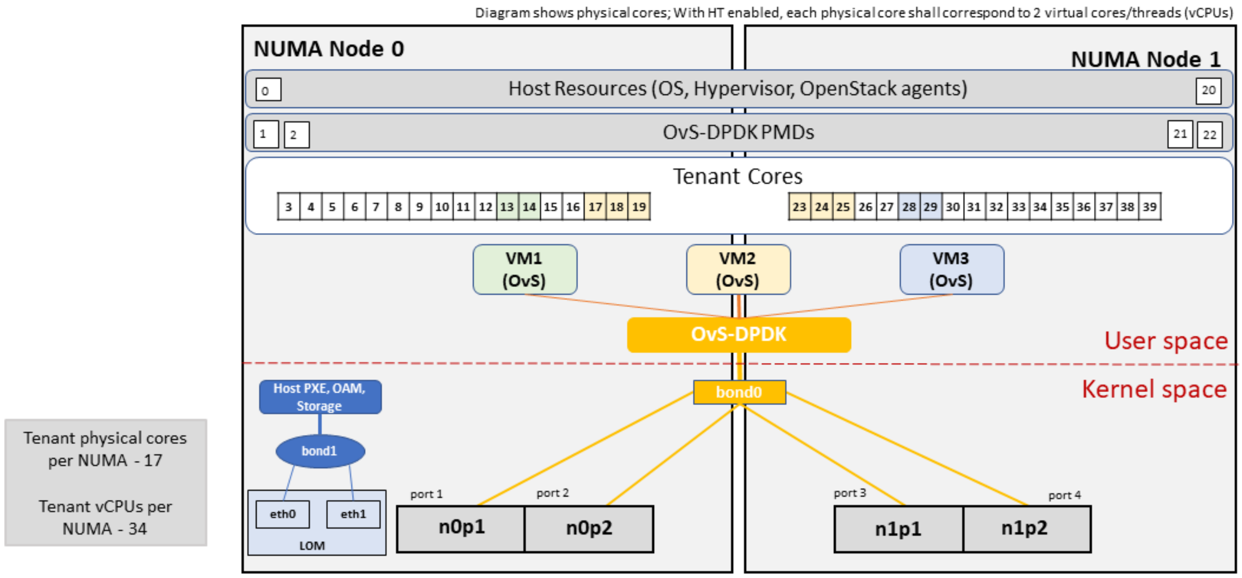

Fig. 4.3: High Performance Profile Host Configuration with DPDK acceleration (example and simplified)

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

Table 4.12: GRUB Configuration of High Performance Flavor with DPDK

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

Host Networking configuration for HS Profile Extensions

An example of the data traffic configuration for the HS (SR-IOV) Profile Extensions is shown in Fig. 4.4.

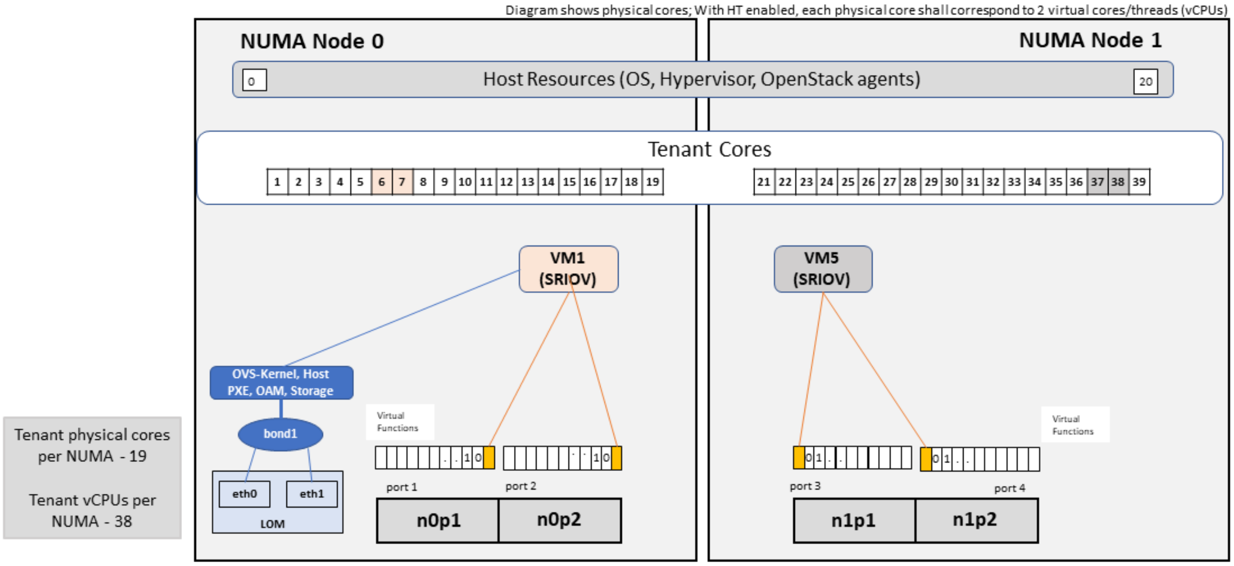
Diagram shows physical cores; With HT enabled, each physical core shall correspond to 2 virtual cores/threads (vCPUs)
NUMA Node 0
NUMA Node 1
0
Host Resources (OS, Hypervisor, OpenStack agents)
20
Tenant Cores
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
VM1 (SRIOV)
VM5 (SRIOV)
OVS-Kernel, Host PXE, OAM, Storage
bond1
eth0
eth1
LOM
Virtual Functions
port 1
port 2
port 3
port 4
n0p1
n0p2
n1p1
n1p2
Tenant physical cores per NUMA - 19
Tenant vCPUs per NUMA - 38

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

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

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

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

Using Hosts of a Host Profile type

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

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

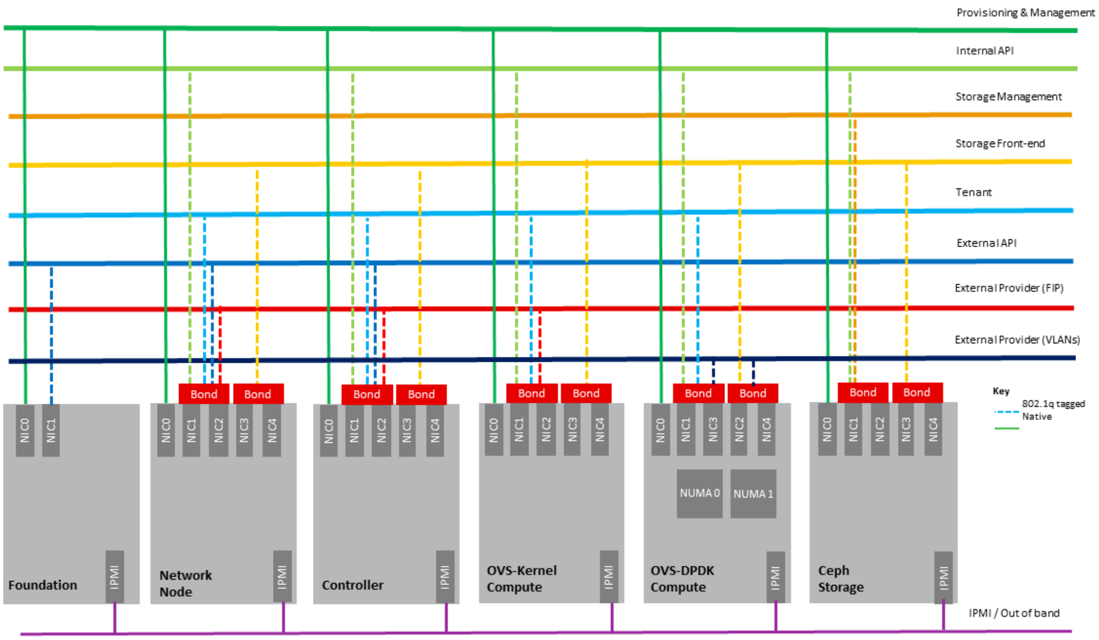

Fig. 4.5: Indicative OpenStack Network Layout

Table 4.14: OpenStack Network Characteristics

Network	Description	Characteristics
Provisioning & Management	Initial OS bootstrapping of the servers via PXE, deployment of software and thereafter for access from within the control plane.	• 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

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

Octavia v2 API conformant Load Balancing

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

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

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

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

Neutron Extensions

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

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

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

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

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

Network quality of service

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

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

Integration Interfaces

- DHCP:

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

- DNS
- LDAP
- IPAM

Storage Backend

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

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

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

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

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

Ceph Storage Cluster

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

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

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

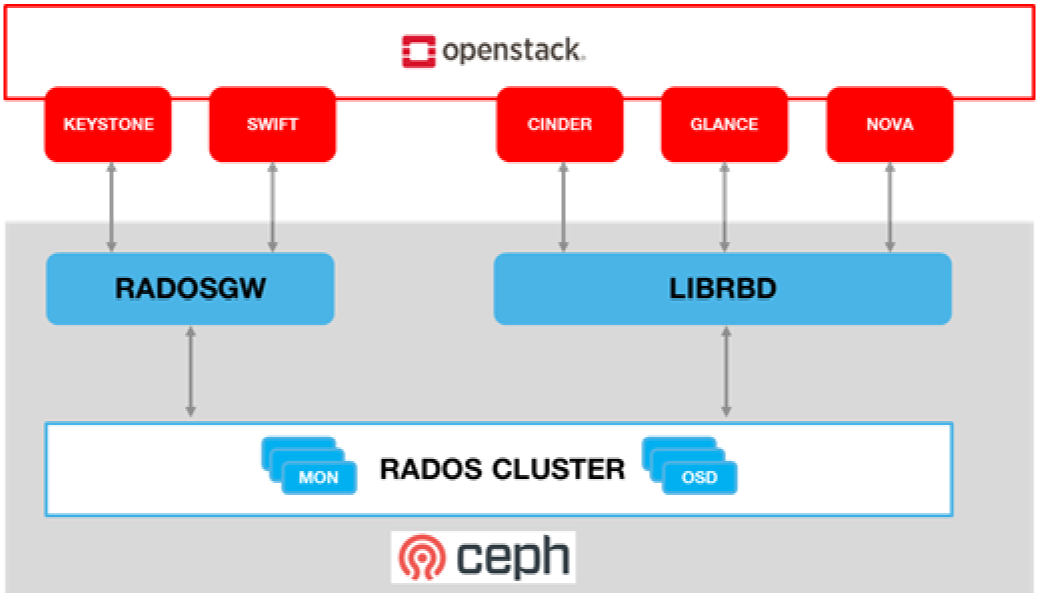

Fig. 4.6: Ceph Storage System

BIOS Requirement for Ceph servers

Table 4.15: BIOS Requirement for Ceph servers

BIOS/boot Parameter	Value
Boot disks	RAID 1

How many nodes to meet SLA :

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

HW specifications :

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

Sizing rules :

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

4.3 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 in Modelling. 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, Swift depends on Keystone and possibly Glance to be able to create volumes from images. Swift has services running on the control nodes and the compute nodes:

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

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

Neutron

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

Table 4.16: Neutron Services Placement

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

Issues with the standard networking (centralised routing) approach

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

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

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

Distributed Virtual Routing (DVR)

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

The OpenStack "High Availability Using Distributed Virtual Routing (DVR)" 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 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 suport dynamic, very large-scale, high-density, telco cloud environments. OpenStack Neutron, with its plugin architecture, provides the ability to integrate SDN controllers (*Virtual Networking – 3rd party SDN solution*). With SDN incorporated in OpenStack, changes to the network is triggered by workloads (and users), translated into Neutron APIs and then handled through neutron plugins by the corresponding SDN agents.

Nova

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

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

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

Ironic

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

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

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

Heat

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

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

Horizon

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

Placement

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

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

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

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

Barbican

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

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

Cyborg

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

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

Cyborg supports:

- Acceleration Resource Discovery
- Accelerator Life Cycle Management

Accelerators can be of type:

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

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

Containerised OpenStack Services

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

In Chapter 3, Fig. 3.2 shows a high level Virtualised OpenStack services topology. The containerised OpenStack services topology version is shown in Fig. 4.7.

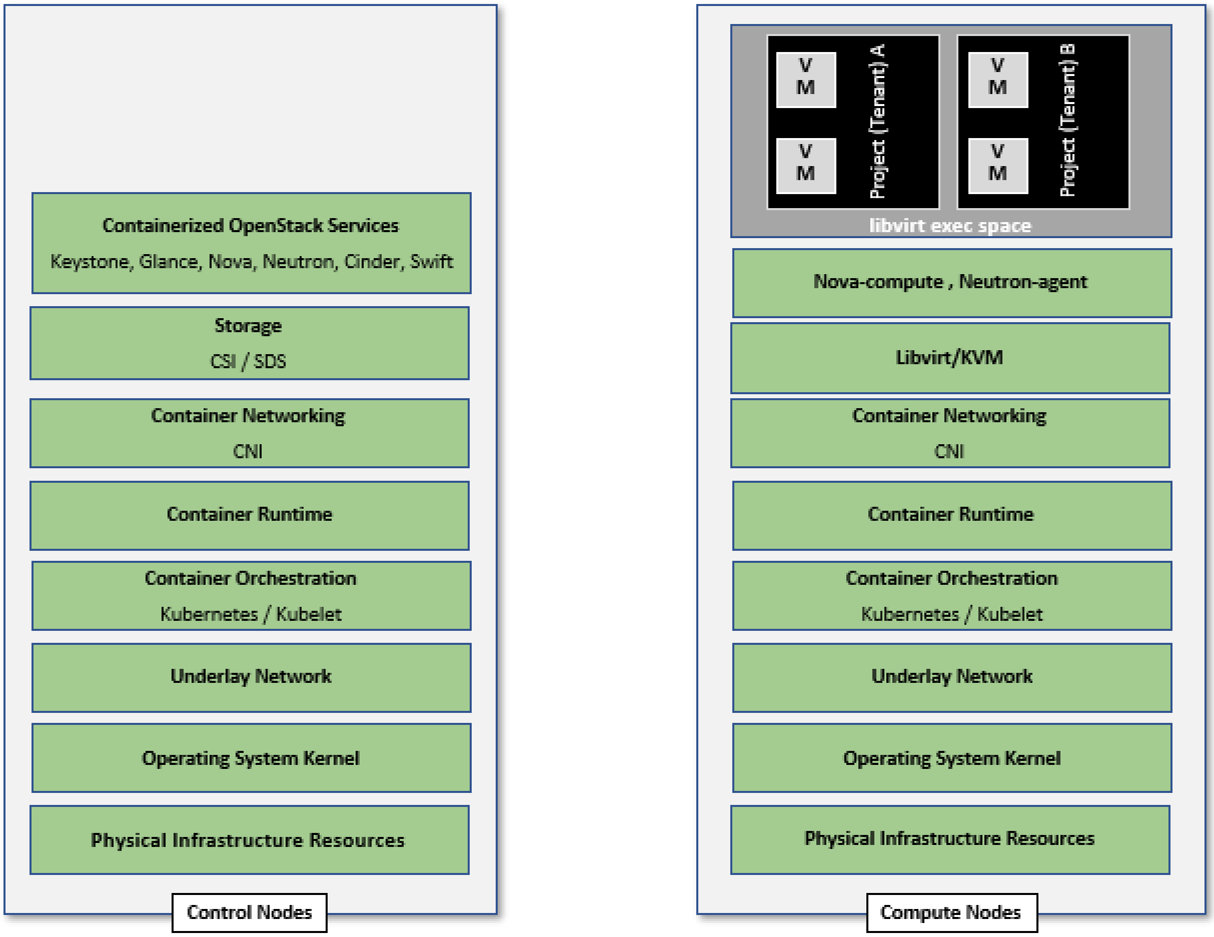

Fig. 4.7: Containerised OpenStack Services Topology

4.4 Consumable Infrastructure Resources and Services

Support for Cloud Infrastructure Profiles and flavors

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

Table 4.17: Neutron Services Placement

Flavor Capabilities	Reference RM Chapter 4 and 5	Basic	High-Performance
CPU allocation ratio (custom extra_specs)	infra.com.cfg.001	In flavor create or flavor set –property cpu_all ocation_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:STORAGEDISK_SSD=required	trait:STORAGE_DISK_SSD=required
Port speed	infra.hw.nic.cfg.002	–property quota vif_inbound_average=1310720 and vif_outbound_average=1310720. Note:10 Gbps = 1250000 kilobytes per second	–property quota vif_inboundaverage=3125000 and vif_outbound_average=3125000 Note: 25 Gbps = 3125000 kilobytes per second

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

The flavor create command and the mandatory and optional configuration parameters is documented in https://docs.

openstack.org/nova/latest/user/flavors.html.

Logical segregation and high availability

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

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

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

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

Limitations and constraints

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

Transaction Volume Considerations

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

4.5 Cloud Topology and Control Plane Scenarios

Typically, Clouds have been implemented in large (central) data centres with hundreds to tens of thousands of servers. Telco Operators have also been creating intermediate data centres in central office locations, colocation centres, and now edge centres at the physical edge of their networks because of the demand for low latency and high throughput for 5G, IoT and connected devices (including autonomous driverless vehicles and connected vehicles). Chapter 3.5 of this document, discusses *Cloud Topology* and lists 3 types of data centres: Large, Intermediate and Edge.

For ease of convenience, unless specifically required, in this section we will use Central Cloud Centre, Edge Cloud Centre and Intermediate Cloud Centre as representative terms for cloud services hosted at centralised large data centres, Telco edge locations and for locations with capacity somewhere in between the large data centres and edge locations, respectively. The mapping of various terms, including the Reference Model terminology specified in Table *8-5 :ref:`ref_model:chapters/chapter08:comparison of deployment topologies and edge terms* and Open Glossary of Edge Computing is as follows:

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

In the Intermediate and Edge cloud centres, there may be limitations on the resource capacity, as in the number of servers, and the capacity of these servers in terms of # of cores, RAM, etc. restricting the set of services that can be deployed and, thus, creating a dependency between other data centres. In Telco Edge Cloud, Table 8-5 specifies the physical and environmental characteristics, infrastructure capabilities and deployment scenarios of different locations.

OpenStack Services Topology of this document, specifies the differences between the Control Plane and Data Plane, and specifies which of the control nodes, compute nodes, storage nodes (optional) and network nodes (optional) are components of these planes. The previous sections of this Chapter 4 include a description of the OpenStack services and their deployment in control nodes, compute nodes, and optionally storage nodes and network nodes (rarely). The Control Plane deployment scenarios determine the distribution of OpenStack and other needed services among the different node types. This section considers the Centralised Control Plane (CCP) and Distributed Control Plane (DCP) scenarios. The choice of control plane and the cloud centre resource capacity and capabilities determine the deployment of OpenStack services in the different node types.

The Central Cloud Centres are organised around a Centralised Control Plane. With the introduction of Intermediate and Edge Cloud Centres, the Distributed Control Plane deployment becomes a possibility. A number of independent control planes (sometimes referred to as Local Control Planes (LCP)) exist in the Distributed Control Plane scenario, compared with a single control plane in the Centralised Control Plane scenario. Thus, in addition to the control plane and controller services deployed at the Central Cloud Centre, Local Control Planes hosting a full-set or subset of the controller services are also deployed on the Intermediate and Edge Cloud Centres. Table 4-5 presents examples of such deployment choices.

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

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

Edge Cloud Topology

The Reference Model Chapter Telco Edge Cloud, presents the deployment environment characteristics, infrastructure characteristics and new values for the Infrastructure Profiles at the Edge.

The Edge computing whitepaper includes information such as the services that run on various nodes. The information from the whitepaper coupled with that from the OpenStack Reference 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 Centre and the number of copies that should be deployed. These references also present the pros and cons of DCP and CCP and designs to address some of the challenges of each of the models.

Table 8-4 in the Reference Model 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.

Table 8-3 in the Reference Model Chapter 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.

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

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

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

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

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

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

Edge Cloud Deployment Tools

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

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

These installers are described in more details in *Operations and Life Cycle Management*.

5 Interfaces and APIs

5.1 Introduction

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

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

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

5.2 Core OpenStack Services APIs

Please note that OpenStack provides a maximum microversion to be used with an OpenStack release. In the following sections the “Maximal API Version” refers to this maximum microversion specified for the OpenStack Wallaby release. Please note that in Reference Conformance (RC-1) testing, the System Under Test (SUT) can utilise newer microversions because of the OpenStack microversion policies. As per multiple OpenStack services documentation, for example the Compute Service, “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

Table 5.1: Keystone

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

Table 5.2: Keystone Features

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

Identity API v3: https://docs.openstack.org/api-ref/identity/v3/index.html

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

Glance

Table 5.3: Glance

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

Table 5.4: Glance Features

Glance Features	Mandatory
import_image	
os_glance_reserved	
web-download import	

Image Service Versions: https://docs.openstack.org/api-ref/image/versions/index.html#version-history

Cinder

Table 5.5: Cinder

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

Table 5.6: Cinder Features

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

Block Storage API: https://docs.openstack.org/api-ref/block-storage/

REST API Version History: https://docs.openstack.org/cinder/latest/contributor/api_microversion_history.html

Swift

Table 5.7: Swift

OpenStack Service	API Version
Object Storage: Swift	v1

Table 5.8: Swift Features

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

Object Storage API: https://docs.openstack.org/api-ref/object-store/index.html

Discoverability: https://docs.openstack.org/swift/latest/api/discoverability.html

Neutron

Table 5.9: Neutron

OpenStack Service	API Version
Networking: Neutron	v2.0

Table 5.10: Neutron Extensions

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

continues on next page

Table 5.10 – continued from previous page

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

continues on next page

Table 5.10 – continued from previous page

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

Table 5.11: Neutron Type Drivers

Neutron Type Drivers	Mandatory
geneve	
gre	
vlan	X
vxlan	

Networking Service APIs: https://docs.openstack.org/api-ref/network/

The exhaustive list of extensions is available at https://docs.openstack.org/api-ref/network/v2/

Nova

Table 5.12: Nova

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

Table 5.13: Nova Features

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

Compute API: https://docs.openstack.org/api-ref/compute/

REST API Version History: https://docs.openstack.org/nova/latest/reference/api-microversion-history.html

Placement

Table 5.14: Placement

OpenStack Service	API Version	Maximal API Microversion
Placement	v1	1.36

Placement API: https://docs.openstack.org/api-ref/placement/

REST API Version History: https://docs.openstack.org/placement/latest/placement-api-microversion-history.html

Heat

Table 5.15: Heat

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

Orchestration Service API: https://docs.openstack.org/api-ref/orchestration/

Template version history: https://docs.openstack.org/heat/latest/template_guide/hot_spec.html

Heat Orchestration Template (HOT) specification: https://docs.openstack.org/heat/latest/template_guide/hot_spec.html#rocky

5.3 Consolidated Set of APIs

OpenStack Interfaces

This section illustrates some of the Interfaces provided by OpenStack; the exhaustive list of APIs is available at https://docs.openstack.org/api-ref/.

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

APIs

Table 5.16: APIs

OpenStack Service	Link for API list	API Version	Maximal API Microversion
Identity: Keystone	https://docs.openstack.org/api-ref/identity/v3/	v3	3.14
Compute: Nova	https://docs.openstack.org/api-ref/compute/	v2.1	2.88
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	3.64
Placement	https://docs.openstack.org/api-ref/placement/	v1	1.36
Orchestration: Heat	https://docs.openstack.org/api-ref/orchestration/v1/	v1	2021-04-06 (template)

Kubernetes Interfaces

The Kubernetes APIs are available at https://kubernetes.io/docs/concepts/overview/kubernetes-api/.

KVM Interfaces

The KVM APIs are documented in Section 4 of the document https://www.kernel.org/doc/Documentation/virtual/kvm/api.txt.

Libvirt Interfaces

The Libvirt APIs are documented in https://libvirt.org/html/index.html.

Barbican

Table 5.17: Barbican

OpenStack Service	API Version
Key Manager: Barbican	v1

Barbican API Documentation: https://docs.openstack.org/barbican/latest/api/

6 Security

6.1 Introduction

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

6.2 Security Requirements

Chapter 2 (*Cloud Infrastructure Security Requirements* and *Security Recommendations*) gathers all requirements and recommendations regarding security topics developed in this chapter.

6.3 Cloud Infrastructure and VIM Security

In the "Security boundaries and threats" section of the OpenStack security guide, 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 the Principle of Least Privilege
- The same authentication credentials must not be reused on different components (sec.sys.011)
- Restrict access to Operating System (sec.gen.005)

Password policy

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

Passwords must be strengthened:

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

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

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

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

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

Function and Software

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

Regarding software (sec.gen.004):

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

Patches

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

Table 6.1: Time to Remediate

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

See Common Vulnerability Scoring System and NIST Vulnerability Metrics.

Network Protocols

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

Anti-Virus and Firewall

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

Vulnerability Detection and Prevention

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

Platform Access

Identity Security

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

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

Authentication

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

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

Keystone Tokens

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

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

Authorisation

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

RBAC

In order to properly manage user access to OpenStack services, service providers must utilise the Role Based Access Control (RBAC) system (sec.sys.001, sec.sys.007). Based on the OpenStack 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 three default roles: 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 unauthorised introspection/exfiltration. Integrity implies that the data must be protected from unauthorised modifications or deletions.

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

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

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

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

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

3 categories of traffic will be protected using TLS:

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

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

Integrity of OpenStack components configuration

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

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

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

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

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

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

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

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

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

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

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

Workload Security

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

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

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

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

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

SR-IOV and DPDK Considerations

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

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 the OpenStack Security Guide. The OpenStack Security Guide includes reference to the "OpenStack Virtual Machine Image Guide" 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. Image signing and verification protects image integrity and authenticity by enabling deployers to sign images and save the signatures and public key certificates as image properties. The creation of signature per individual artifact in the VNF package is required by ETSI NFV SOL004.

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

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, etc.) as described in *Operations and Life Cycle Management*. These operations must be securely performed in order to keep the infrastructure safe and operational (sec.lcm.003).

Provisioning/Deployment

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

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

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

Configuration and management

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

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

Platform backup

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

Security upgrades

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

Monitoring and Security Audit

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

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

Creating Logs

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

What to Log / What NOT to Log

What to log

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

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

What NOT to log

Security audit logs must NOT contain:

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

Where to Log

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

Required Fields

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

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

Data Retention

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

Security Logs Time Synchronisation

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

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

7 Operations and Life Cycle Management

7.1 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 chapter we will discuss the capabilities of provisioning and configuration management systems; some Open Source tools may be mentioned but their capabilities are beyond the scope of this chapter.

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.

7.2 Cloud Infrastructure provisioning and configuration management

In the Reference Model, Configuration and Lifecycle Management 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.

Underlying resources provisioning

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

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

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

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

VIM deployment

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

- OpenStack TripleO, “OpenStack on OpenStack”

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

- Airship v2

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

- StarlingX

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

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

Configuration Management

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

Configuration Management is composed of the following activities:

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

7.3 Cloud Infrastructure and VIM Maintenance

Cloud Infrastructure and VIM Maintenance activities can be classified as

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

Deployment (or removal) of infrastructure components

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

Configuration and Version Changes

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

7.4 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 the Infrastructure Capabilities, Measurements and Catalogue.

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 behavior. The data from various parts of the environment are collected into a monitoring system that is responsible for storage, aggregation, visualisation, and initiating automated responses when the values meet specific threshold.

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

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 utilising Prometheus, Fluentd, Elasticsearch and Kibana is given as an example only.

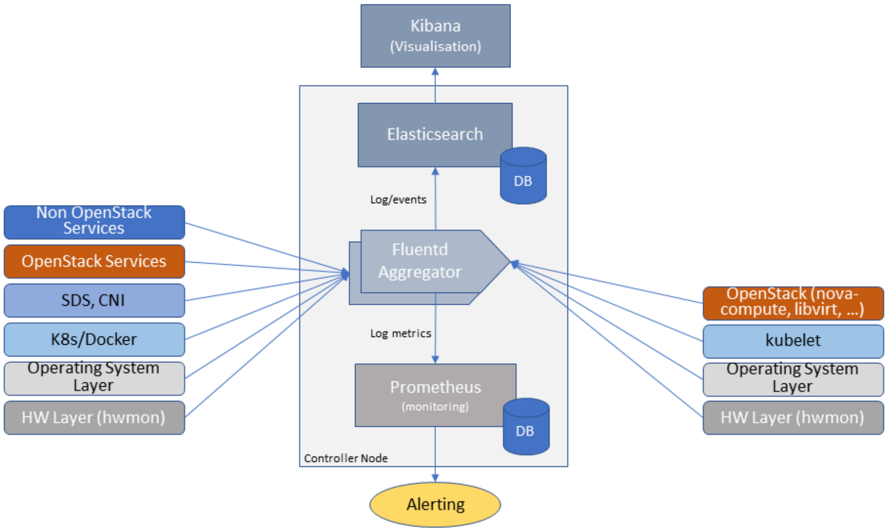

Fig. 7.1: Monitoring and Logging Framework

The monitoring and logging framework (Fig. 7.1) leverages Prometheus as the monitoring engine and Fluentd for logging. In addition, the framework uses Elasticsearch to store and organise logs for easy access. Prometheus agents pull information from individual components on every host. Fluentd, an Open Source data collector, unifies data collection and consumption for better use and understanding of data. Fluentd captures the access, application and system logs.

8 Gaps, Innovation, and Development

8.1 Introduction

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

8.2 The Gap

Autoscaling

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

For more information on auto-scaling with Heat please see: https://docs.openstack.org/senlin/latest/scenarios/autoscaling_heat.html. Please note that the OpenStack Senlin service is still under development with major architectural changes made in the OpenStack Ussuri release.

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