Framework for Advanced Services in Virtual Machines

Document History

Version	Date	Authors	Changes
v0.1	9-16-2013	Greg Regnier	Initial Version
v0.2	9-27-2013	Greg Regnier	Added contributors; initial feedback incorporated
V0.3	10-15-2013	Greg Regnier	Added feedback from contributors and IRC discussion
V0.4	10-20-2013	Micheal Thompson, Greg Regnier, Geoff Arnold	Expansion of blueprint, added data model, updates of resources, limited use cases.

Collaborators	
Sumit Naiksatam, Kanzhe Jiang, Kuang-Ching Wang, Big Switch Networks	Gary Duan, Yi Sun , vArmour
Marc Benoit, Palo Alto Networks	Rudrajit Tapadar, David Chang, Sridar Kandaswami, Joseph Swaminathan, Bob Melander, Kyle Mestery, Cisco
Rajesh Mohan, Dell Software Group	Greg Regnier, Christian Maciocco, Uri Elzur, Greg Cummings, Isaku Yamahata, Intel
Geoff Arnold, Brocade	Balaji Patnala, Freescale Ravi Chunduru, Paypal
Micheal Thompson, Mani P A10 Networks	Hunter Thompson,AP Student, Virginia Beach Public Schools

Table of Contents

1. Introduction

Neutron now has multiple advanced services defined. These services are deployed and managed using the advanced service APIs and plugin structure of Neutron. Currently there is no framework within Neutron to manage and deploy services that reside in virtual machines (VMs). The goal of this specification is to provide a framework that allows advanced services that reside within VMs to be managed and deployed in a consistent manner within Neutron. The proposal is to extend the capabilities of the advanced services framework to include an internal interface that enables the deployment and management of VM-based services. The anticipated benefit of providing a framework is to lower the requirements for consumption of advanced services by coupling the compliance and security policy with the deployment pattern.

1. Terms

The following terms are used in this blueprint.

Neutron Logical Service Object (Servo) – Tenant visible service created via the Neutron API.

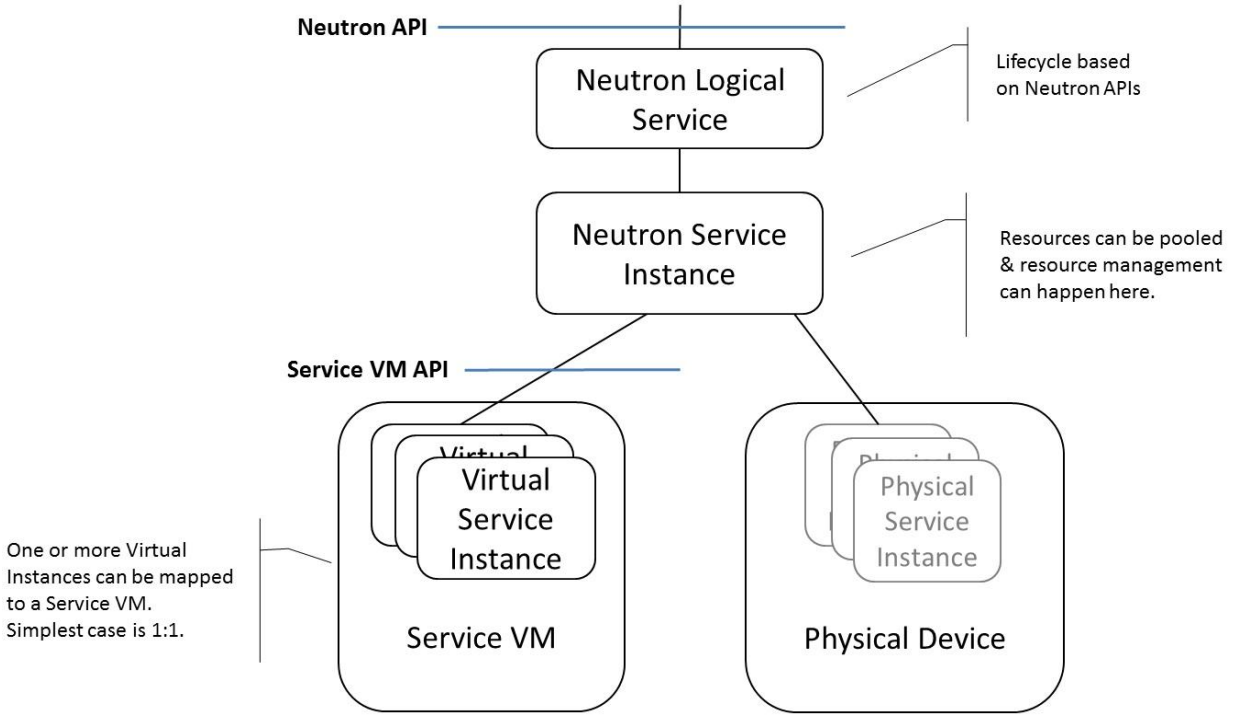
Service VM – a Virtual Machine that hosts specialized function(s) to be offered and consumed as services. A Service VM is a container that hosts one or more Logical Service Instances.

Logical Service Instance (LSI) – an instance of a neutron logical service hosted by a Service VM.

Svc-vm-mgmt-lib - module used to provide a common interface to advanced services to launch, deploy and manage the service VM.

1. Data Model

A Neutron Resource Instance is defined by the Neutron Advanced Services API. The terminology in this diagram *(contributed by Geoff Arnold*) needs to be normalized with agreed upon terminology

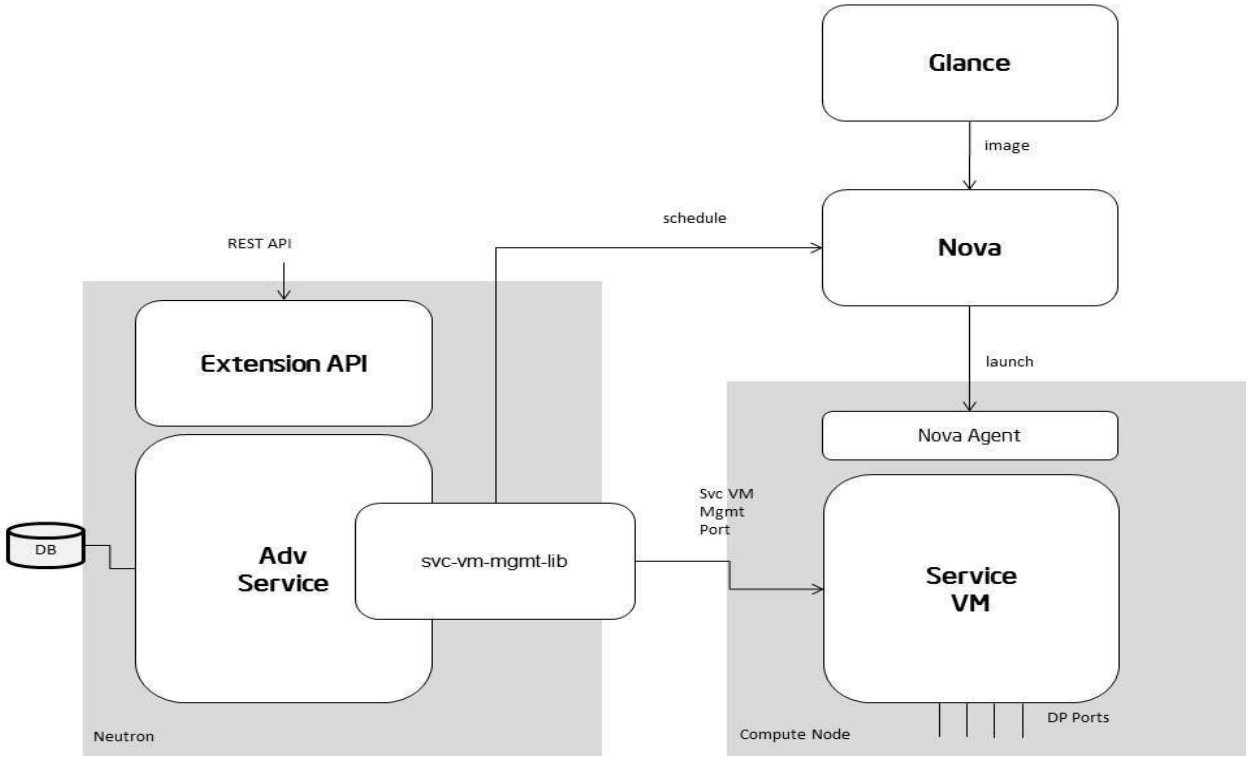


.

1. Theory of Operation

The advanced services framework allows a tenant to create one or more instances of service(s), which have networking dependencies (firewall, load balancer, ADC, WAF, or GSLB). The tenant is exposed to a logical model for a service, while there are multiple possible backend implementations. For example, a given service may be a process that resides in the host, it may be a physical device, or it may be contained within a VM.
A related blueprint (Neutron/ServiceTypeFramework) describes how an administrator can add services from multiple service providers to Neutron and how tenants can choose services based on ServiceType and Service Provider. Information for these instantiated services should be stored in the Neutron/DeviceInventory. In the case where a service VM can support virtual appliances, these appliances should be added as individual appliances until the deletion of the service VM.

In the case of a service that is implemented as a VM, the Neutron advanced services model abstracts the details of launching the service VM and deploying it into the tenant's network. The scope of this proposal is to provide common interfaces and mechanisms for instantiation of a Service VM into the network, with the ability to classify how a Service VM is consumed.



The baseline theory of operation for a VM has several deficiencies that need to be addressed with more advanced integration patterns in order for an advanced services API to be beneficial. There could be an assumption that the registered providers are equally capable of fulfilling the criteria. This could result in selection of a service, which can meet the baseline selection requirements provided during configuration, but does not meet service levels required for service usage. The “it’s a service; why do I care?” philosophy does not address considerations associated with regulation or compliance of services.

The figure above represents the high level elements to extend the existing advanced services framework to enable service VMs. The new element in the block diagram is the service VM management library (svc-vm-mgmt-lib). This module is used to provide a common interface to advanced services to launch, deploy and manage the service VM.

The deployment of a service VM requires the scheduling/instantiation of a virtual machine into a tenant’s network. In a Service VM is already instantiated that is capable of hosting multiple Logical Service Instances, the service may be scheduled onto an existing Service VM. If not, the library will use Nova to schedule and launch a service VM. The service VM is stored in Glance image store, and Nova launches the service VM onto a compute node.

Depending on the service that is configured, there could be a requirement to enable a “Service Proxy Agent”(SPA) as part of the svc-vm-mgmt-lib. This SPA can act as a proxy for device drivers to connect to and configure the desired service.

1. Entities

Service VM

A Service VM is a type of Virtual Machine that provides specialized functions to be offered and consumed as services. It is a container that hosts one or more Logical Service Instances (LSIs). A Service VM provides management connectivity for instantiation of services. The following attributes can be assigned to a Service VM.

Attribute	Type	Default Value	Required	Description
Id	uuid		Y	
name	varchar	None	Y	Descriptive name
Provider_name	varchar			
Service_Type	varchar			
tenant_id	uuid		Y	Administrator or tenant that owns this Service VM
Image	uuid	None	Y	The VM machine image
Flavor	uuid	None	Y	The VM flavor
vm_mgmt_if	TEXT	eth0	Y	Required
security_group	Object	Object		
multi-tenant	Boolean	False	N	Default to single tenant service VM
Max_LSI	INT	0	N	Number of Logical Service Instances supported
DP_IF_Types	Object	{“eth”:”if-	N	Data-plane

		id","VLAN":"vid", {other} }		interface types available; default is IP
classification	TEXT	NONE	N	Abstract metadata utilized for filtering services
availability_zone	uuid	NONE	N	NOVA availability zone
host_aggregate	uuid	NONE	N	NOVA host aggregate
allow_mgmt_access	Boolean	false	N	Should device be managed?
access_cred_uname	TEXT	None	N	Autogenerated if not supplied
access_cred_passwd	TEXT	None	N	Autogenerated if not supplied

Logical Service Instance

A Logical Service Instance (LSI) is the final expression of a neutron service object. Multiple LSI's may be contained within a Service VM.

Attribute	Type	Default Value	Description
id	uuid	None	ID of this LSI
name	varchar	None	Descriptive name
tenant_id	uuid	None	ID of the owner tenant
service_vm_id	uuid	None	ID of hosting service VM
multi-tenant	Boolean	false	Default is single ten
state	ENUM	DOWN	Operational State (states tbd)
mgmt-if	List	None	Management interface for

			this service instance
networks	List of Objects	None	List for network:subnet assignment
obj_store	Object		Service-specific attributes
cost_factor	?	?	CPU, Network, maybe this is policy…

A management interface is used to configure and update the policy and resources of the Logical Service Instance. This interface is service specific. An example use of the management interface is to add a rule to a firewall or VIP.
Management interface(s) must always be reachable. This means that even if the virtual networks are blocked, this interface must be reachable by plugin.

There are several methodologies that can be used to gain management access. In the case of an IP-only management interface, this can be achieved through a Service Proxy Agent, which once invoked can create a proxy connection to the management interface as needed. In order to achieve this additional security, related items would need to be considered, and an address reservation should occur.

Another approach is for the hypervisor to use a serial port with each service VM. This separates the management channel from the data network.

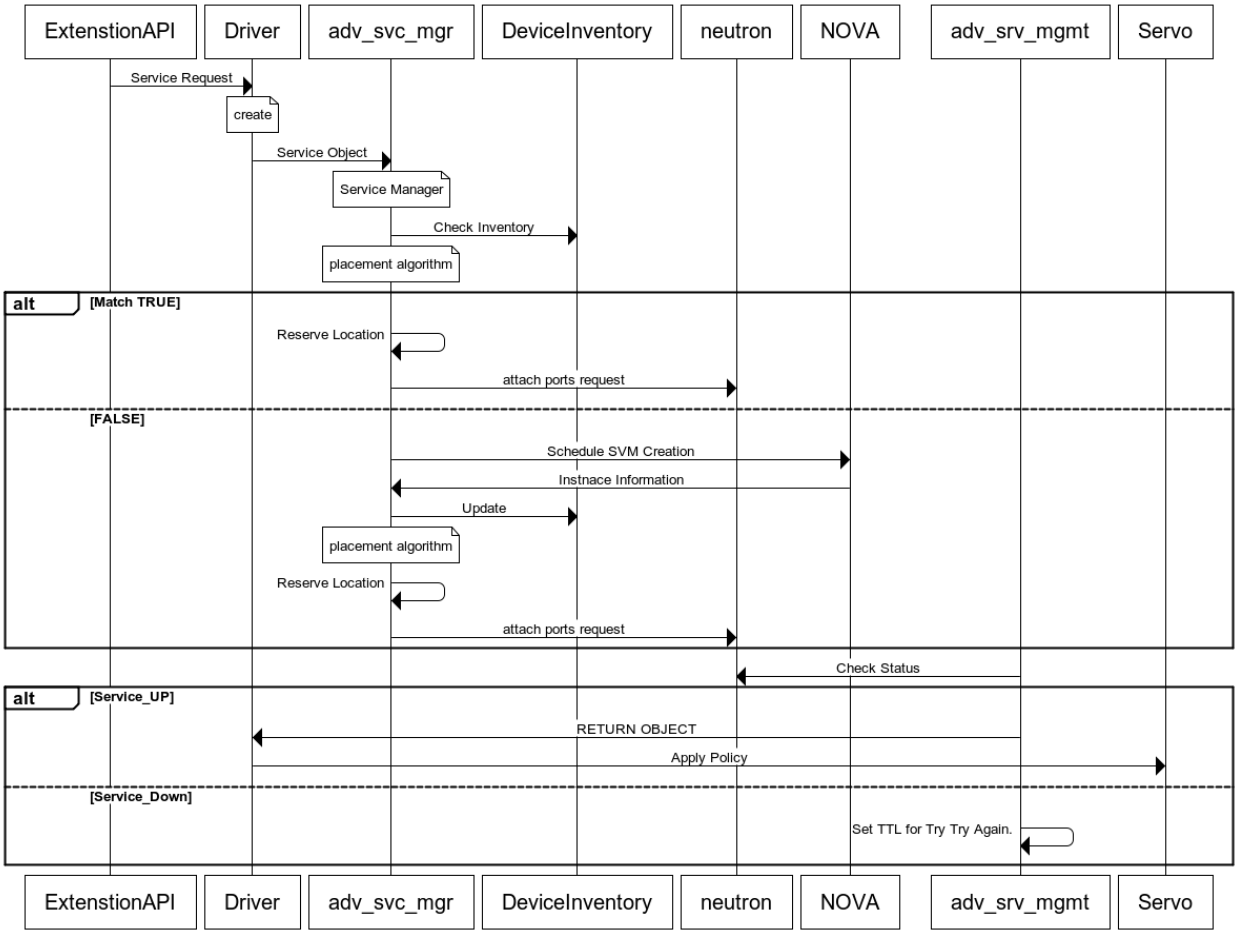
1. Scheduling and Resource Allocation

There are several strategies for scheduling and resource allocation, and five primary concerns associated with scheduling and resource allocation:

1. Provider
2. Security
3. High Availability
4. Capacity
5. Capability
6. Licensing

Related proposals for service scheduling and resource management strategies can be found at: https://blueprints.launchpad.net/neutron/+spec/dynamic-network-resource-mgmt and https://blueprints.launchpad.net/neutron/+spec/quantum-service-scheduler.

There are deficiencies in each blueprint, as neither covers all topics needed for the specific integration patterns associated with VM deployments. However, for the sake of this blueprint, the method of allocation is isolated to Service VMs instantiated and managed through the advanced services framework.

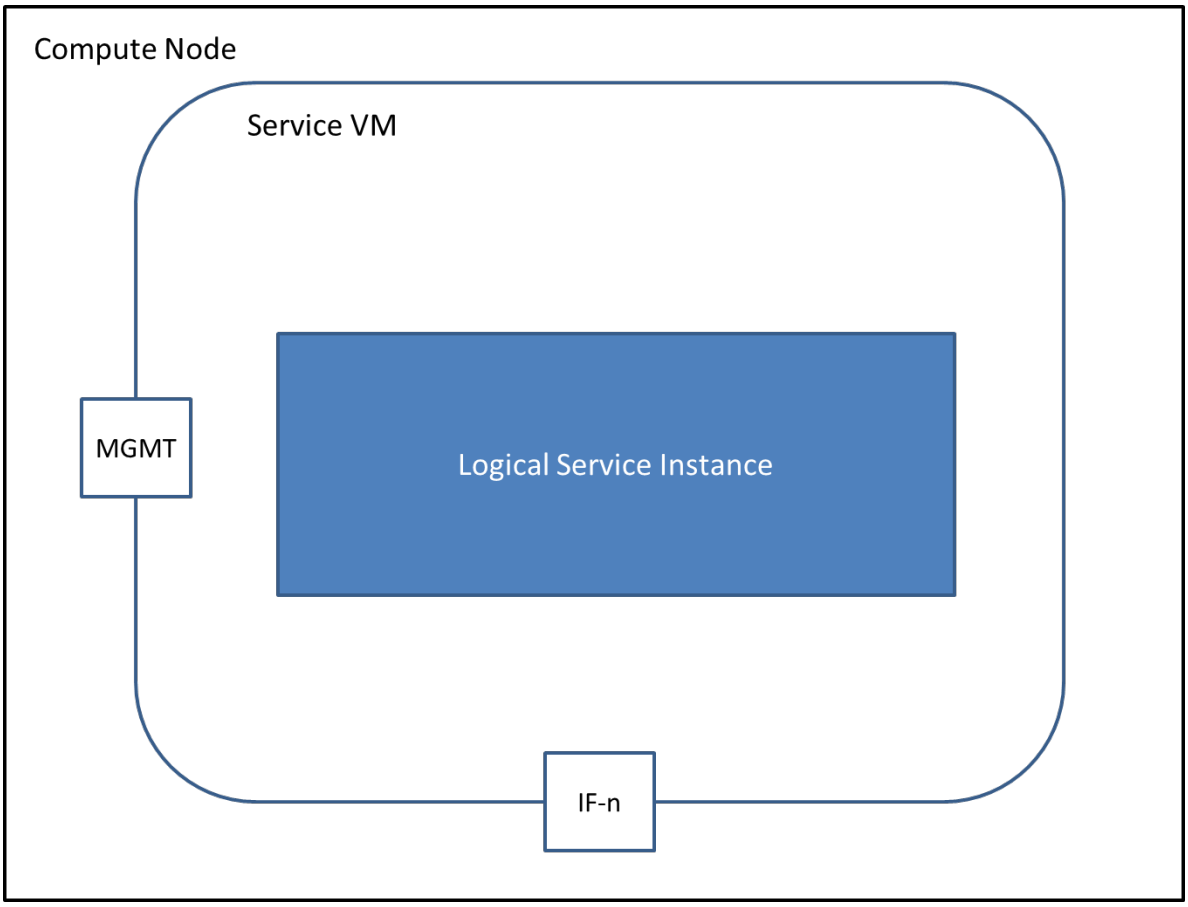


1. Use Cases

This section describes the most common use cases anticipated for Service VMs. *(Contributed by Mike Thompson). Additional use cases under discussion.*

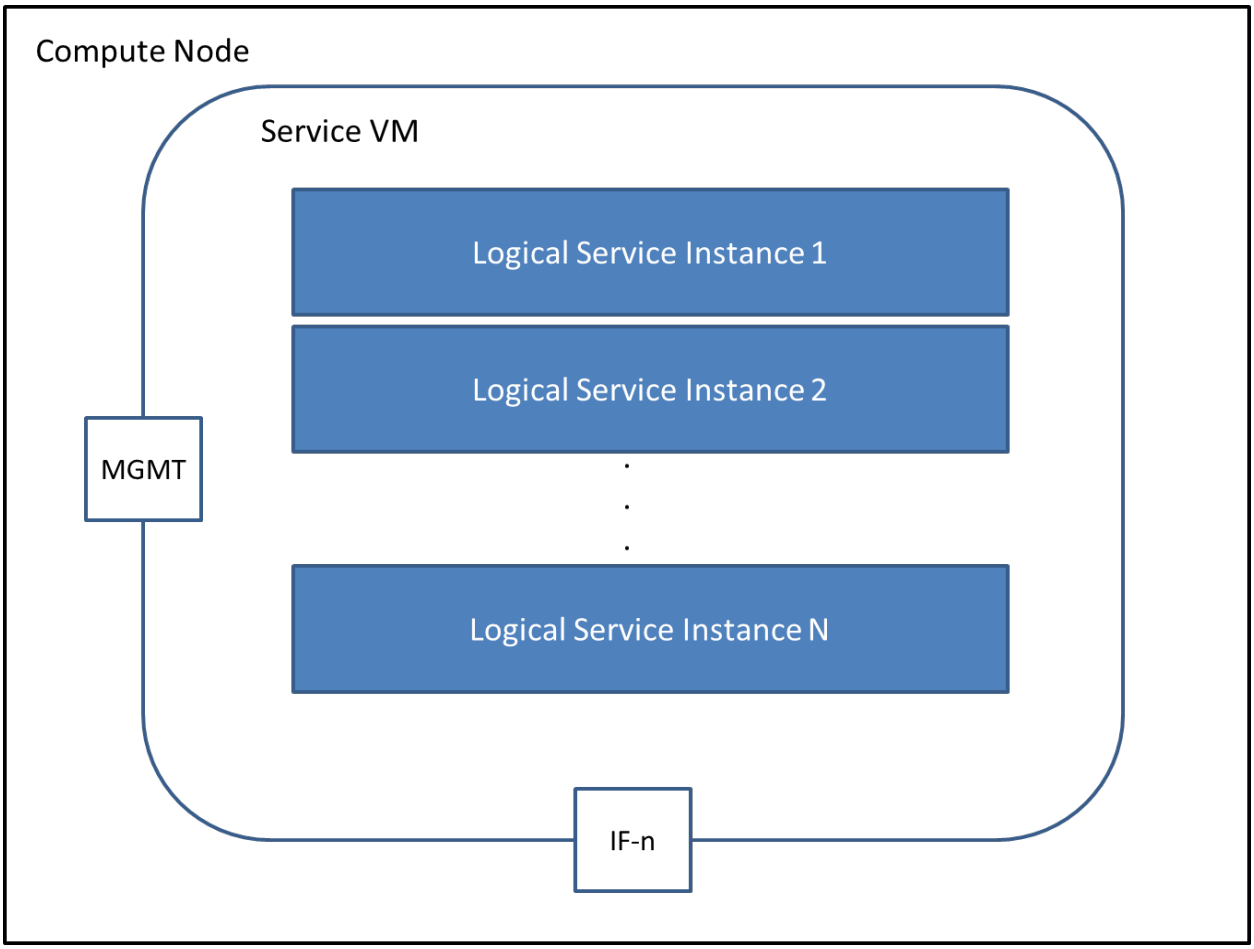
Use case 1: Single Tenant, Single Instance in a Service VM

- Default case.
- Service VM is created/scheduled, and owned by the tenant.
- Service VM hosts one Logical Service Instance.
- Data Ports can plug into only the network(s) owned by tenant.
- Tenant can choose to set credentials for direct access.



Use case 2: Single Tenant, Multiple Instances in a Service_VM

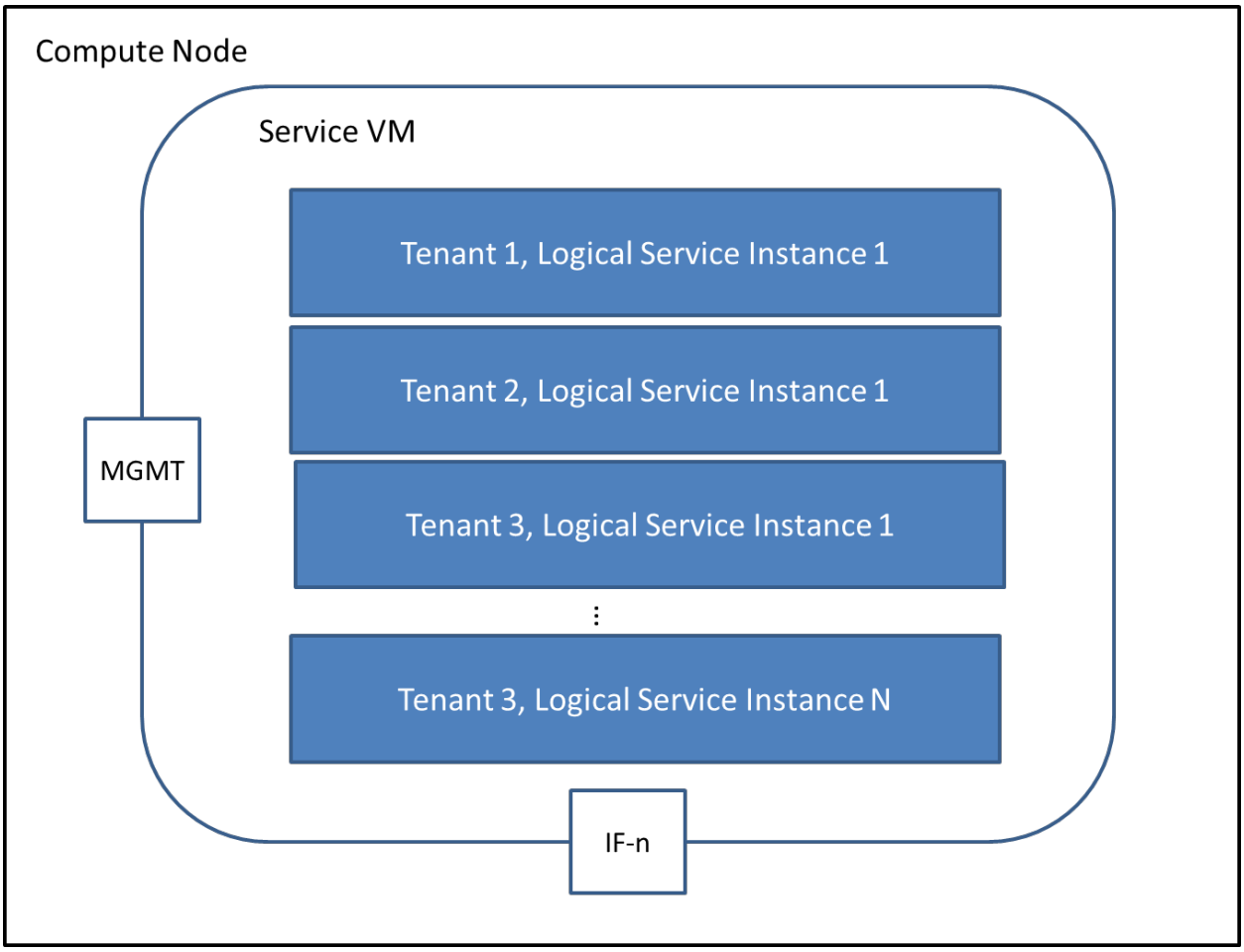
- Tenant has CRUD but cannot set administrative direct access.
- Service VM hosts multiple Logical Service Instances.
- Each Logical Service Instance is scheduled/pinned to a tenant.
- Data ports of each Logical Service Instance can be shared with other logical services placed on the same Service_VM, and can be plugged only into owner tenant's network(s).
- Source Network Address Translation may occur.



Use case 3: Multiple Tenant, with Logical Service Instances on a Service_VM

Note: Support for multi-tenant in the first rev of this blueprint is open for discussion – GR.

- Tenant has CRUD but cannot set administrative direct access.
- Each logical service instance is scheduled/pinned to a tenant but Service_VM is not owned by tenant.
- Data ports of each Logical Service Instance can be shared with other logical services placed on the same Service_VM, and can be plugged into only the owner of Service_VM tenants network(s). LSI also can have dedicated ports.
- Source Network Address Translation may occur.

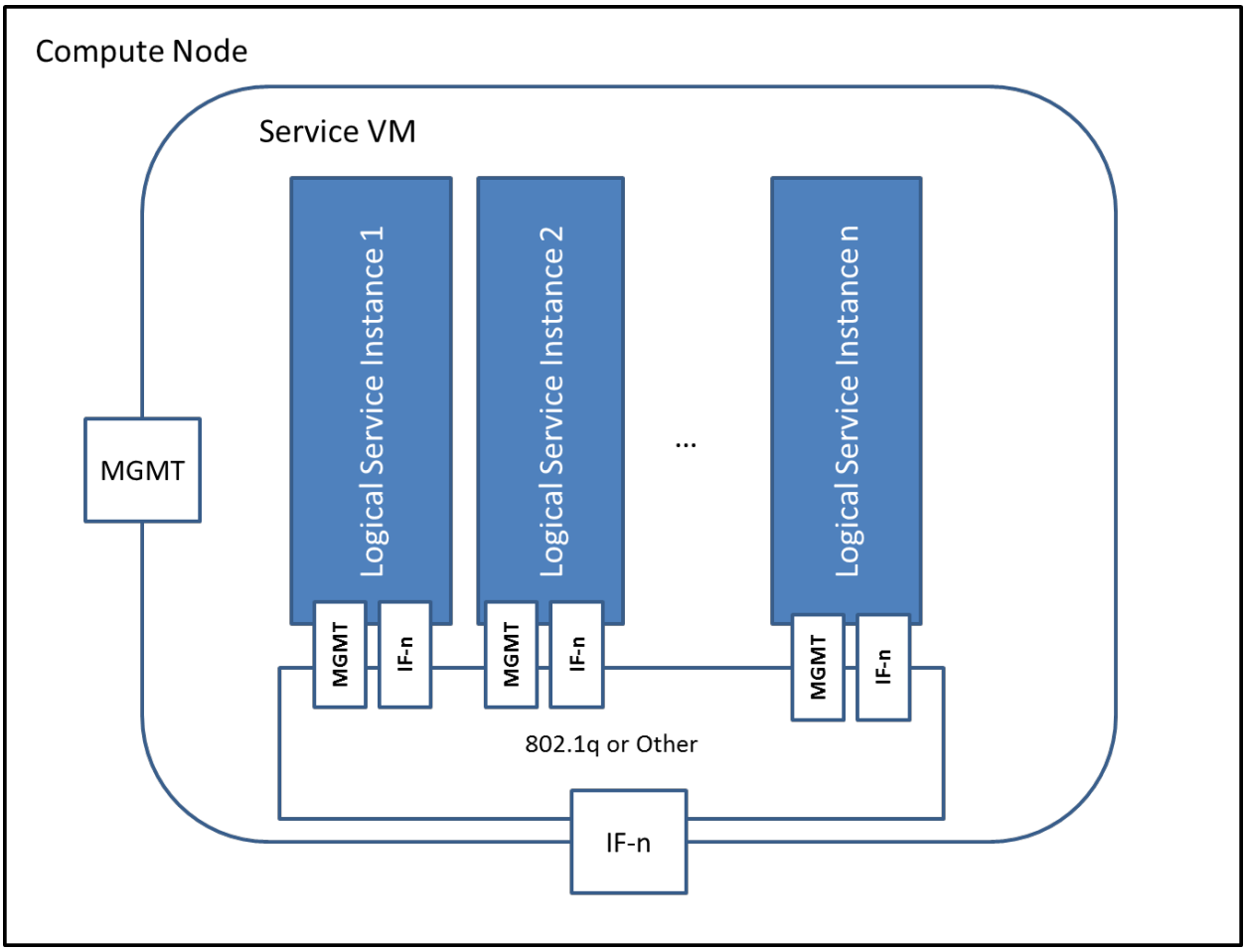


1. Service VM Ports and VIF attachment

A logical service instance has one or more data interfaces (ports). These ports are mapped to VIFs that are allocated in the VM. The VIFs can be allocated dynamically when the service VM supports hot-plugging. Alternately, the VIFs may be pre-provisioned within the VM and then mapped to logical service instances on demand.

To save on the number of required VIFs, a solution is VLAN trunking. The service VM has a single interface (or a small number of them) for data traffic. When attached to a particular tenant network, that network is trunked on the port used by the service VM. The benefits of this solution are that no VIF hot plugging is needed and fewer VIFs are needed The downside is that the plugin needs to support VIF trunking, which may or may not be the common case.

The figure below represents an example Service VM with multiple Logical Service Instances. A separate blueprint discusses the means to allow a Neutron port to connect to multiple networks: https://blueprints.launchpad.net/neutron/+spec/quantum-network-bundle-api.



The related proposal for service insertion and chaining is described in the blueprint: https://blueprints.launchpad.net/neutron/+spec/neutron-services-insertion-chaining-steering.