



OpenStack Neutron

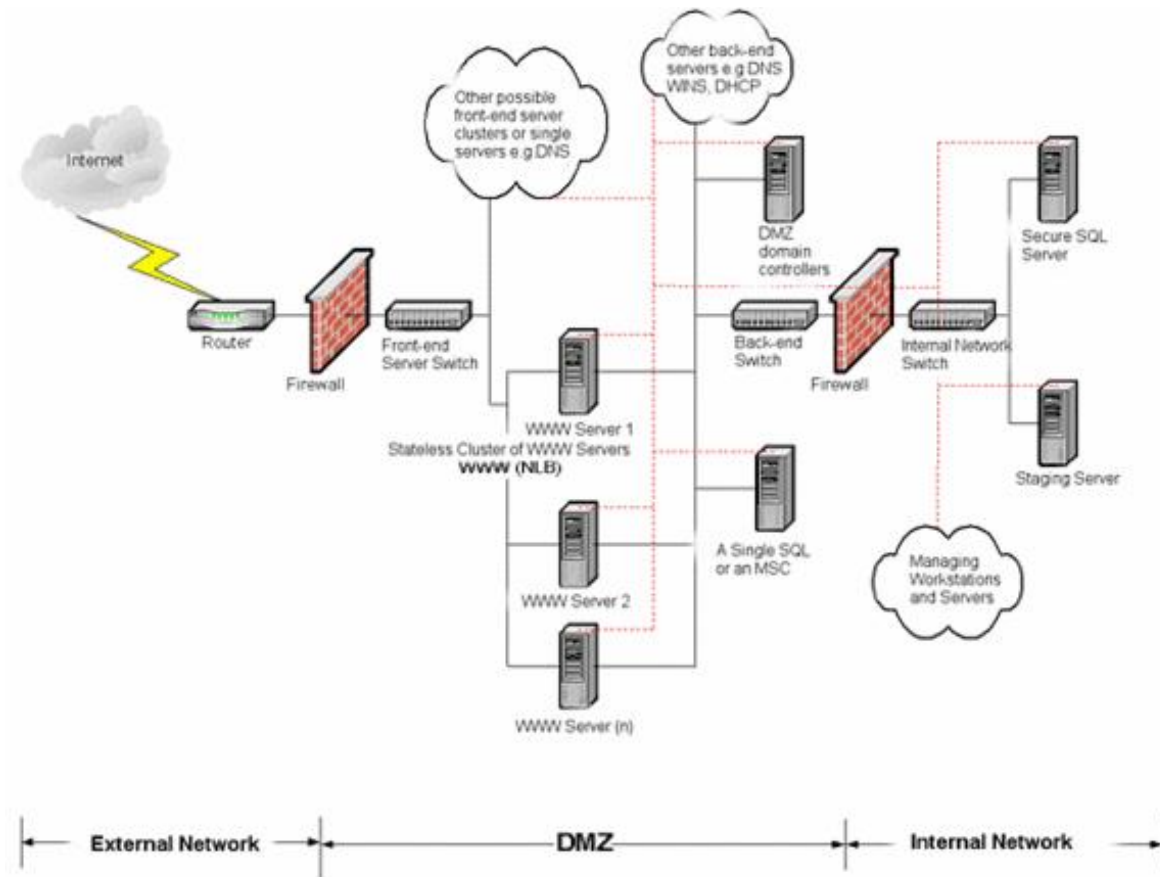
Outline

- Why Neutron?
- What is Neutron?
 - API Abstractions
 - Plugin Architecture



Why Neutron?

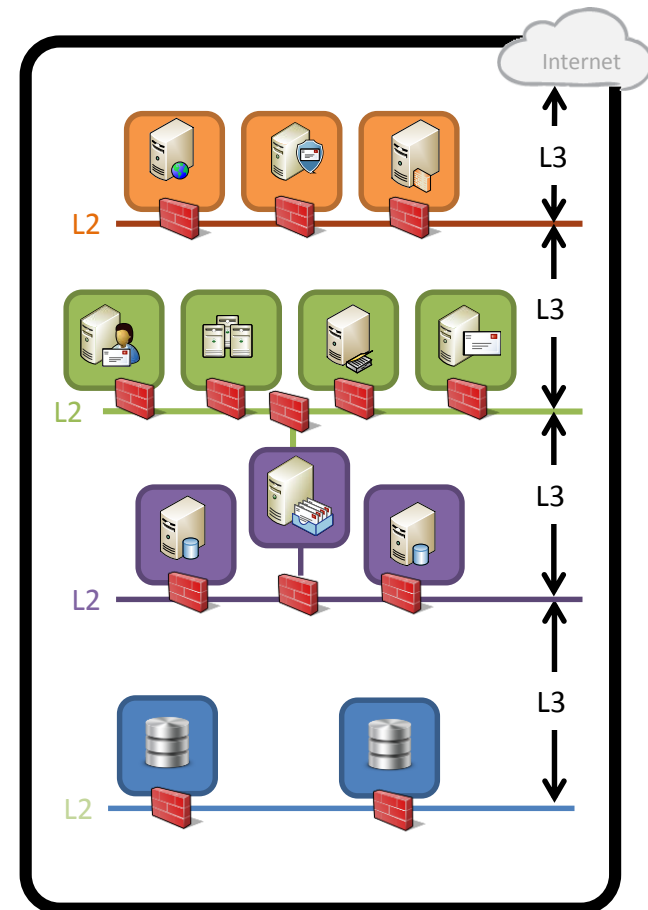
Networks for Enterprise Applications are Complex....



Why Neutron? Reason #1

On-demand Enterprise-Class Networking

- Neutron has Tenants API to:
 - create multiple private L2 networks
 - control IP addressing (can use same IP space as existing datacenter deployment)
 - Connect to an upstream router for external access.
 - Insert advanced network services: routers, firewalls, VPN, IDS, etc.
 - Monitor network status



Why Neutron?

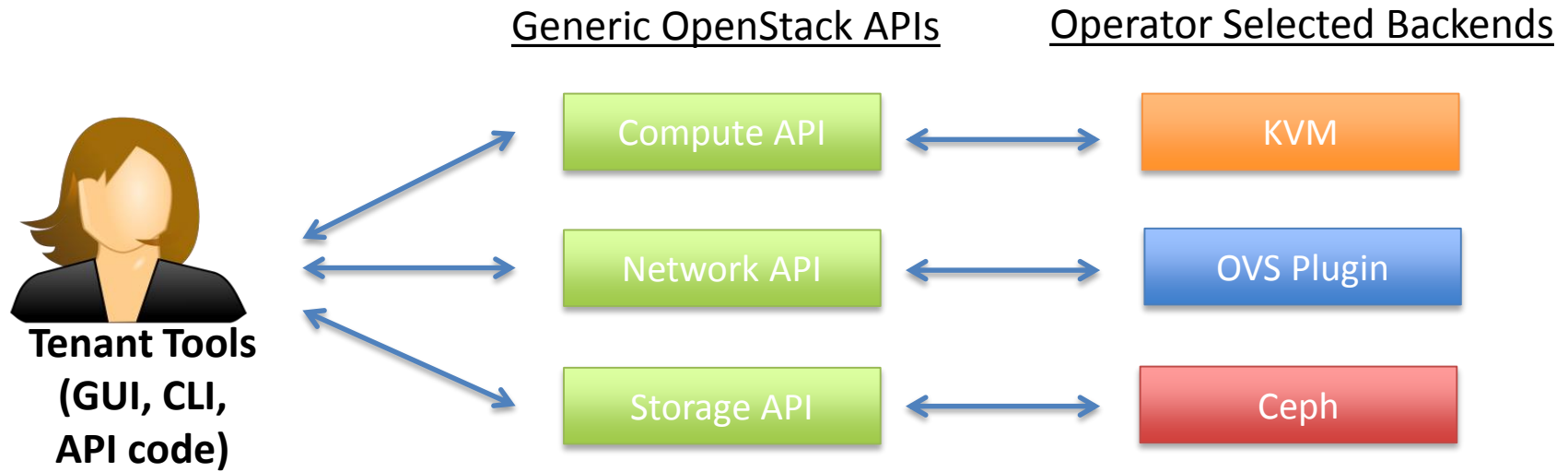
#2: Leveraging Advanced Technologies

- New networking technologies are emerging to try and tackle these challenges.
 - Network virtualization
 - Overlay tunneling: VXLAN, NVGRE, STT
 - Software-defined Networking (SDN) / OpenFlow
 - L2 Fabric solutions: FabricPath, Qfabric, etc.
 - [insert other solution here]
- Neutron provides a “plugin” mechanism to enable different technologies (more later).



What is Neutron?

Neutron Architecture

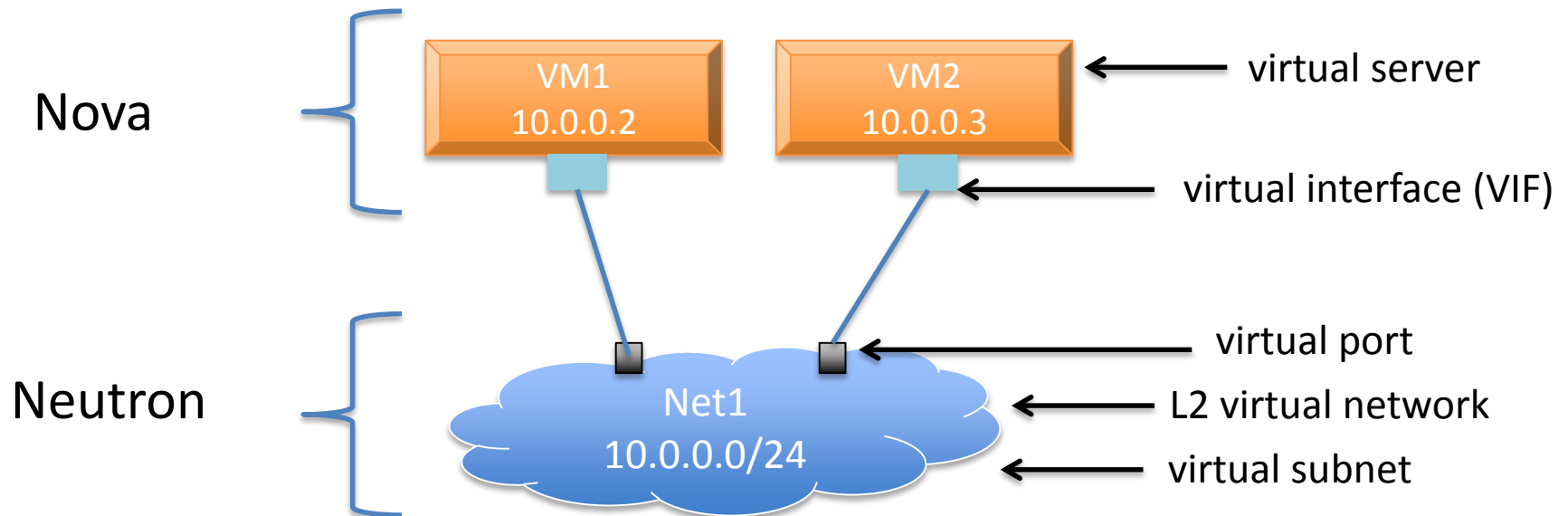


An eco-system of tools that leverage the Neutron API.

A generic tenant API to create and configure “virtual networks”

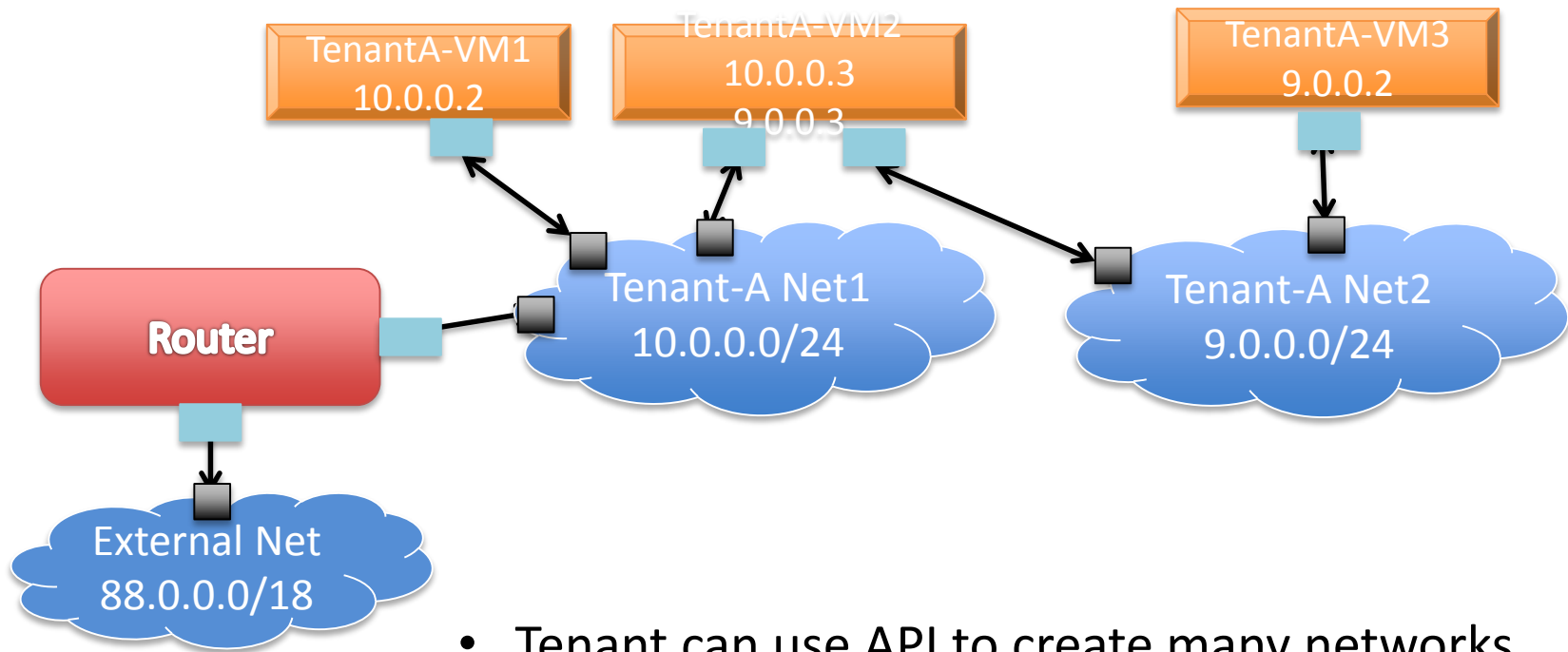
A “plugin” architecture with different back-end “engines”

Basic API Abstractions






“virtual networks” and “virtual subnets” are fundamentally multi-tenant, just like virtual servers (e.g., overlapping IPs can be used on different networks).

Neutron Model: Dynamic Network Creation + Association

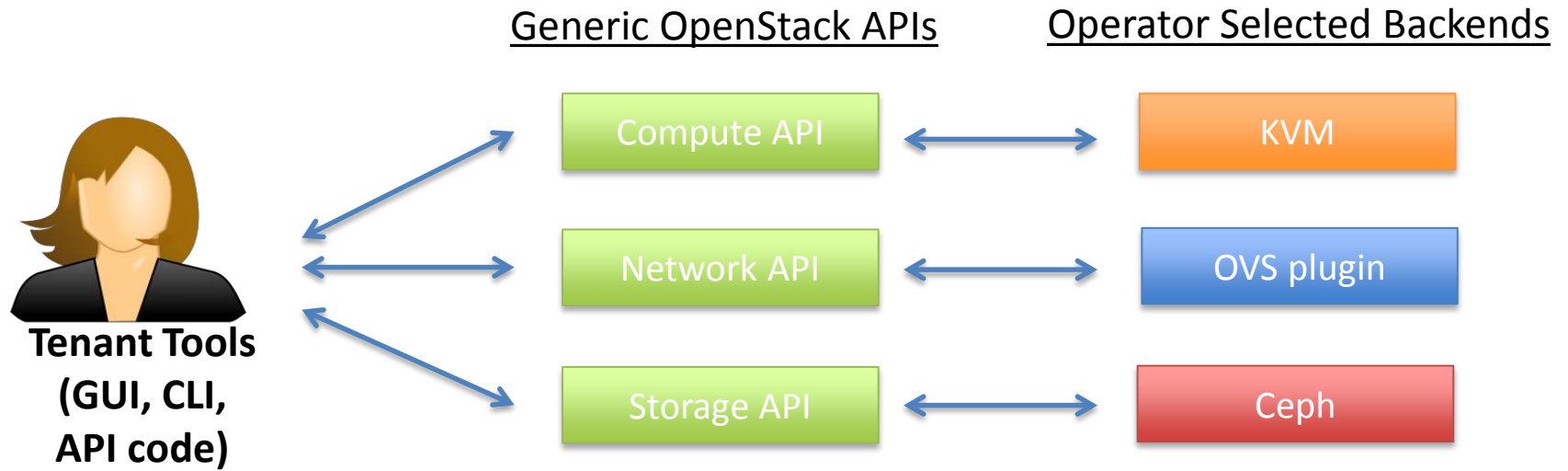


- Tenant can use API to create many networks.
- When booting a VM, define which network(s) it should connect to.
- Can even plug-in “instances” that provide more advanced network functionality (e.g., routing + NAT).

Neutron API Extensions

- Enables innovation in virtual networking.
 - Tenants can query API to programmatically discover supported extensions.
 - Overtime, extensions implemented by many plugins can become “core”.
- Add properties on top of existing network/port abstractions:
 - QoS/SLA guarantees / limits 
 - Security Filter Policies 
 - port statistics / netflow 
- New Services
 - L3 forwarding, ACLs + NAT (“elastic” or “floating” IPs)
 - VPN connectivity between cloud and customer site, or another cloud datacenter.

Neutron Architecture



An eco-system of tools that leverage the Neutron API.

A generic tenant API to create and configure “virtual networks”

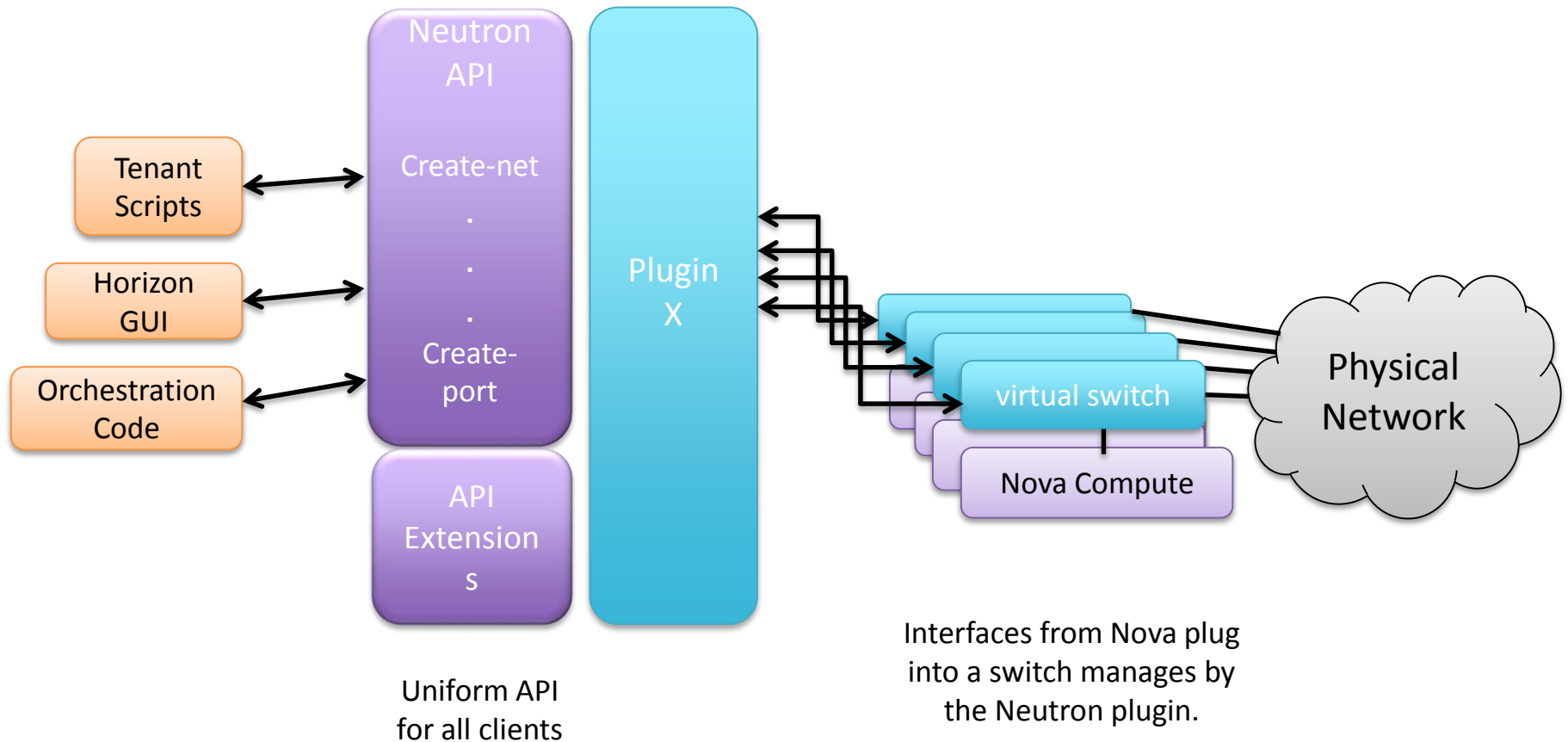
A “plugin” architecture with different back-end “engines”

Neutron Architecture (generic)

API Clients

Neutron Service

Backend X



Neutron Plugins Trade-offs

- Different back-end “engines” present different trade-offs:
 - Scalability
 - Forwarding performance
 - Hypervisor Compatibility
 - Network HW Compat (vendor specific? Allow L3 scale-out?)
 - Manageability / troubleshooting
 - Advanced Features (exposed as API extensions)
 - Production testing
 - High Availability (control & data plane)
 - Open source vs. Free vs. Paid
- Cloud Operators weigh trade-offs, choose a plugin.
- Note: Back-end technology hidden behind logical core API
 - Example: VLANs vs. tunneling

Two API Deployment Models

- Cloud Operator creates networks for tenants
 - Neutron API is admin only, tenants do not use it.
 - Similar to nova-network model, but with flexibility around network topology, IP addressing, etc.
- Expose API to tenants directly
 - True “self-service networking”.
 - Tenants use scripts, CLI, or web GUI to manage networks & subnets.
- Can also mix-and-match strategies
 - Provider creates default network connectivity, tenants can choose to extend.