Overview

This document describes the Proof of Concept (PoC) to build for the Neutron Group Policy Abstractions. For reference, that document can be found at this link.

The plan is to try and implement this as a PoC in the Icehouse timeframe to vet out the thinking, and work to get this upstream in the Juno timeframe.

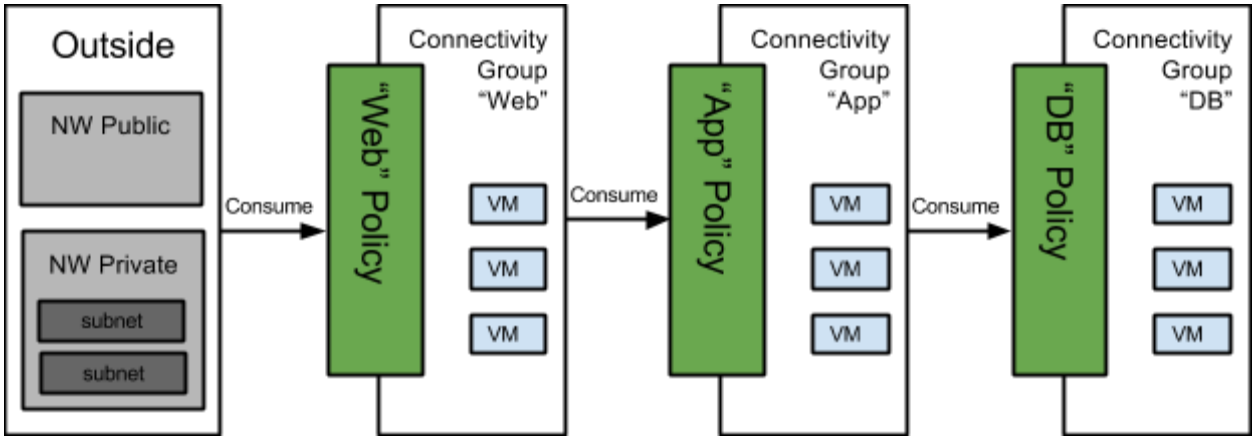
PoC Use Case

Goals

- Show clear separation of concerns between application developer and infrastructure manager. That is:
    - The application developer will deal with a higher level abstraction that does not concern itself with networking specific issues like networks/routers/etc.
    - The infrastructure manager will deal with infrastructure specific policy abstractions and not have to understand application specific concerns like specific ports that have been opened or which of them expect to be limited to secure or insecure traffic.
    - Allow for independent provider/consumer model with late binding and n-to-m relationships.
- Show automatic orchestration that can respond to changes in policy or infrastructure without requiring human interaction to translate intent to specific actions

Use Case

- Provision three tier web app as described in group based policy blueprint (we will start with 1 layer - the web layer, and add 2nd and 3rd layer as time permits).



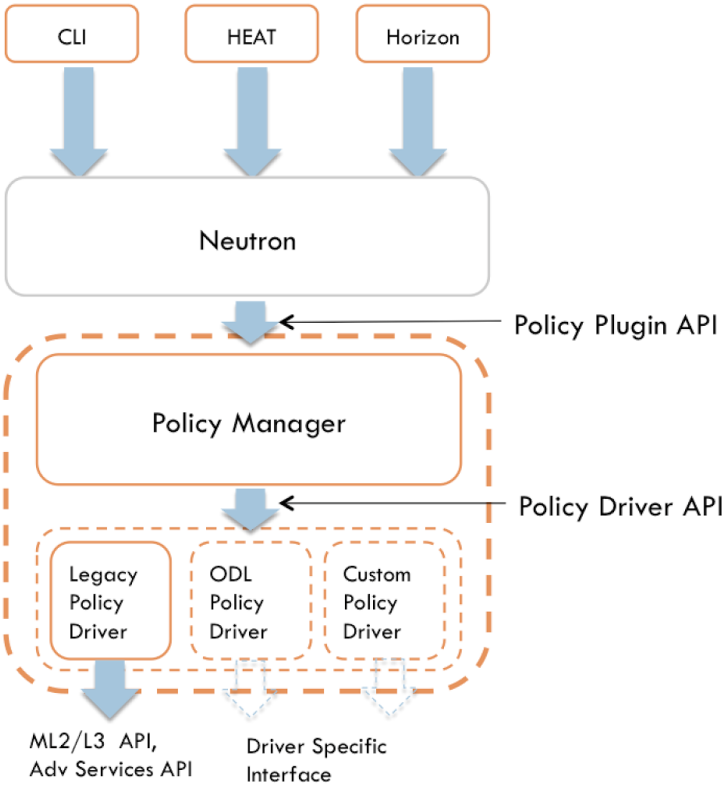
a. The app developer creates groups and contracts specifying the network service (using labels in contracts for public, private and mgmt policies). The specific steps required should be
    i. Create the required network contracts
        1. Assume that the labels public, private and mgmt already exist in the database
        2. Assume that the labels to identify connectivity types, security, QoS, etc also already exist in the database
        3. Identify specific policy rules as applicable to specific usage using required labels
    ii. Create the required end point groups
    iii. Associate “provides” relationship with corresponding EPGs as in the diagram above
    iv. Associate “consumes” relationships with corresponding EPGs as in the diagram above
    v. As a side effect of the steps above, the appropriate orchestration is done
b. The app developer creates VM instances and puts them in the appropriate end point group
c. The infra manager identifies public/private/mgmt end points and consumes the Web Policy contract by applying the labels to the “consumes” relationship.
d. As a side-effect of steps (b & c), the “orchestration update” happens in the background as required
e. The infra admin creates a policy to redirect all public traffic via a firewall. As a side effect (and without any app developer visible impact), that change is “orchestrated”.
f. The app developer adds a new port for mgmt traffic (say port 7000 for updating cassandra), and as a side effect that is “orchestrated” without any visible impact on infra manager’s abstractions.

NOTE: We will try to achieve the same use case using:

1. CLI, with updates for the group policy API
2. HEAT, with updates for the new group policy API
3. Horizon (if we can get the appropriate UI updates in time for PoC)

High Level Design Decisions

1. CLI — Add GP API
2. HEAT — Add GP API
3. Horizon — Add GP API
4. Devstack — Add GP support
5. Neutron — Add Intercept API
6. Policy Manager
   1. Define Policy API
   2. Define Driver API
   3. Implement PM
7. Policy Driver
   1. Driver loading like ML2
   2. Driver base class
   3. Legacy policy driver



The following are the high level design decisions for the PoC:

- In base neutron, we will create an “interceptor API” (which can be used for multiple uses like debugging, tracing, decoration, etc).
- We will use this interceptor API to implement the Group Policy API as equivalent to core_plugin + policy_abstractions.
- The plugin will be composed of two parts, the high level policy manager (that also acts as the root of the policy configuration - that is, the record of intent of the user) and the policy enforcement (that implements policy - and for PoC this will be done using existing south bound interfaces to ML2/L3/advanced_services). In future, this API will be exposed and networking technology specific plugins can be created (like ML2 mechanism drivers).
- We will also create appropriate HEAT extensions to make the new API consumable at higher layers.
- We will also build the devstack updates and Horizon updates required for the PoC.