Group-Based Policy for OpenStack

Introduction

Over the past four years, OpenStack has grown from a simple open source project to a major community-based initiative including thousands of contributors in more than a hundred countries. It has played a critical role in bringing cloud computing across computing, storage, and networking resources to both public and private cloud environments. It has also built a growing a multivendor ecosystem, allowing users to bring together OpenStack software and hardware through a set of open APIs.

As OpenStack clouds begin to scale and the basic infrastructure-as-a-service (IaaS) use case evolves, develepors have an opportunity to look beyond the initial challenges set of setting up virtual machines, configuring block storage devices, and orchestrating network connectivity on demand. Future IaaS environments need to focus on deployment and delivery of applications and services with speed, agility, flexibility, security, and scale rather than just orchestrating infrastructure components. For such solutions, a declarative policy engine can be a critical component.

This document introduces the new Group-Based Policy (GBP), a new framework, designed to offer a new set of API extensions to manage OpenStack infrastructure through declarative policy abstractions. GBP is designed on the principle of capturing application requirements directly rather than converting the requirements into a specific set of infrastructure configurations. It introduces a new declarative API for automating OpenStack infrastructure. It is initially targeted at OpenStack networking resources, but the abstractions are general enough to apply to computing and storage resources as well.

Challenges in Neutron Networking

Neutron (originally Quantum) was created as a network abstraction layer for OpenStack. The model chosen created logical equivalents of the underlying networking constructs, primarily to enable tunnel-based overlays. This approach led the team to select a set of highly network-centric primitives for Neutron APIs: a logical Layer 2 “network” and logical Layer 3 “router.” Behind these logical constructs, Neutron supports a broad range of open source and vendor-specific plug-ins to work with different hardware and software devices.

Unfortunately, the current approach has some drawbacks. For instance the properties of a virtual machine are defined in a number of different places. Its security policies are captured in a security group construct, and the Neutron network is used to describe its connectivity requirements. A growing number of extensions separately describe Layer 3 requirements,

quality of service (QoS), and network services. Other non-networking properties of a virtual machine such as its placement affinity, are described through entirely separate constructs. The end result is a system that requires multiple updates to completely replicate a virtual machine. Even though these updates can be automated, the properties reside across the entire system, making automation more complex and potentially inconsistent.

The current APIs also assume deep knowledge of modern networking. Application developers, the real users of the cloud infrastructure, need to think about different layers of the OSI stack as they deploy their applications. So although the detailed network configuration is very natural for a networking team, in many cases it adds unnecessary complexity and confusion for application developers. This complexity only increases as new features such as network services are added to the mix. Application developers just want to describe the network and security requirements of their applications in the simplest terms possible.

Additionally, the current Neutron APIs were designed to fit a specific virtual networking model. As new vendors have developed software-defined networking (SDN) solutions, they have engaged a broad range of technologies and capabilities that in some cases deviate from traditional networking requirements. However, the current model based on Layer 2 and Layer 3 virtual networks offers relatively little flexibility to these solutions, tying them to a set of existing network behaviors.

Finally, although Neutron does offer a construct for provider networks, in which the network is fully managed by the operator, it does not have a clean mechanism whereby both the user and operator can specify co-existing requirements. This separation of concerns is a critical component in offering security and flexibility in cloud environments.

Introducing Group-Based Policy

Group-Based Policy aims to address these issues by offering a simple, abstract API designed to capture user intent. It is based on the following key concepts (Figure 1):

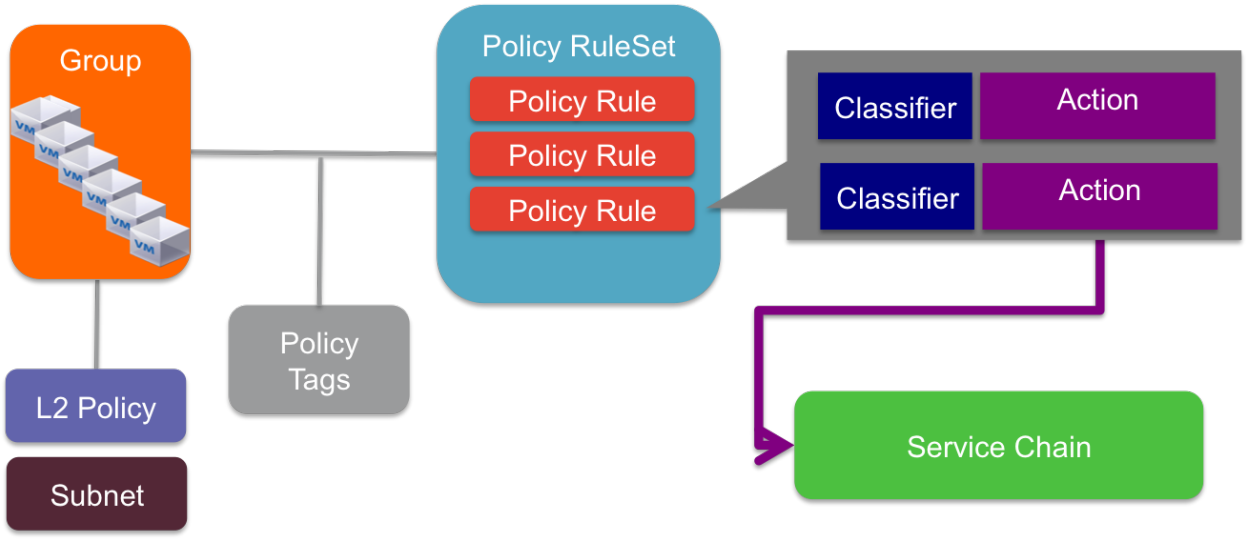
- **Groups**: GBP introduces a concept of a group that represents a collection of network endpoints and fully describes their properties. Everything in the same group must be treated the same way (that is it has the same policy). This approach is simple but powerful generalization of the constructs in OpenStack today and maps very well to scalable application tiers used by most developers.

- **Reusable policy rule sets**: GBP introduces rule sets to describe secure connectivity between Groups. Rule sets may imply switching or routing behaviors, but they offer a simple way to describe how sets of machines can communicate in non-networking terms. Critically, they are also **reusable**. The same rule set can be used for different combinations of Groups. This reusability reduces the number of places that must be updated as policies change, thus improving agility, security, and consistency.

- **Policy layering**: GBP is designed to allow policies to be layered based on different roles in an organization. For instance, layering allows application owners to specify the policy pertaining to an application, while infrastructure owners can prescribe security requirements such as redirection of traffic to a chain of firewall and intrusion-detection system (IDS) solutions before the traffic is sent to the application. Both policies can coexist and be described using nested primitives.

- **Network services:** The GBP model also supports a *redirect* operation that complex network service chains and graphs easy to abstract and consume. Network service chaining is a mechanism for connecting multiple Layer 4 through Layer 7 services such as load balancers and firewalls. GBP API thus allows application developers to specify these requirements as components between a combination of groups rather than through switching or routing configuration.

Figure 1: Group-Based Policy Model



The GBP model offers a number of very powerful advantages over the way the current Neutron API works today. Automation and security are much easier through GBP. By simply becoming a member of a group, a virtual machine inherits *all* of its policies, allowing developers to easily automate scaling up and down. This approach offers a naturally flexible and extensible framework for capturing the requirements of a virtual machine in a single location. It also makes consistency easier to achieve because only one step - becoming a member of the group – is required to inherit multiple policies.

The model is easy for application developers to use and offers them a simple way to describe application requirements. In fact, it was designed to make advanced capabilities such as service chaining extremely easy to use. Finally, the GBP model offers a means for allowing

operator and user requirements to coexist cleanly. After a user describes the requirements for an application, the infrastructure operators can impose additional policy constraints if desired.

GBP was designed by a community of developers including Big Switch, Cisco, Huawei, IBM, Midokura, Nuage Networks, and One Convergence.

Use Cases

Group-Based Policy was designed to make it easier and faster for tenants to describe their application requirements. However, the richness of its robust policy model can apply to broad range of use cases. Here are a few examples:

1. Self-service automation and application scaling

   GBP was designed to separate the complexity of the underlying infrastructure from the applications owners' intent. It makes it extremely easy to build reusable models of application requirements through its Group and policy rule set constructs. This approach can remove deployment steps in which user configurations are validated by various infrastructure teams because users rely only on abstract policies. Additionally, the grouping mechanism allows fast and easy scaling of an application tier as needed.

2. Metadata-driven Infrastructure

   GBP includes an extremely flexible primitive called a policy tag. Policy tags can be used to select membership in a Group or opt in or out of portions of a policy rule set. The end result could be a set of infrastructure policies entirely driven off these policy tags. A virtual machine is labeled with one or more policy tags to describe its properties, and the infrastructure automatically configures itself based on that information.

3. Separation of concerns: policies for operators and users

   GBP is designed to support separation between operator and user requirements. Although a user may be able to author a Group or policy rule set, these objects can also inherit behaviors from a set of operator-defined primitives. This approach allows an operator to impose constraints transparently on top of a user configuration without losing the original user intent for the application.

4. Enforcing security requirements

   GBP offers several different ways of tightly enforce a security and compliance policy, such as Payment Card Industry (PCI) requirements. GBP's service insertion capabilities can dictate firewall insertion between different groups to help ensure that proper security is enforced. Additionally, GBP requires a self-documenting policy configuration that

defines exactly what communication is allowed between parts of an application. This whitelist model can be used by external audit or compliance tools to document network behavior.

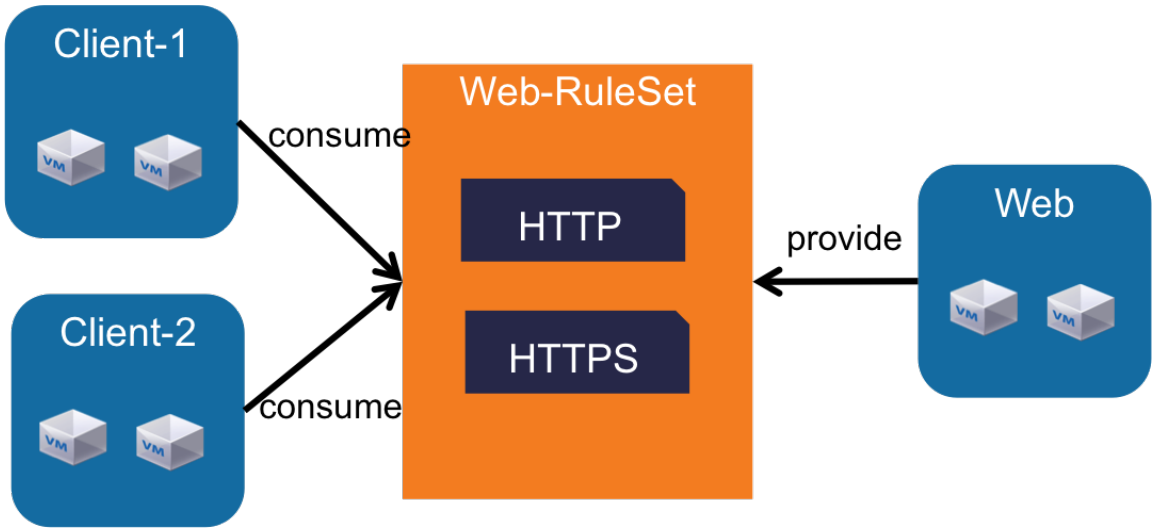
5. NFV / scale out services

   GBP's flexible approach to service chaining can be used to support next generation scale out services. Because the policy captures abstract services that must be inserted and chained, it can support back ends that are designed to be spread throughout the infrastructure. Scale-out services can be extremely useful in optimizing traffic flows and scaling application workloads.

Example: Using GBP Policy

This section presents an example of the use of GBP to create a simple set of rules between web servers and multiple clients. This example introduces three groups: web, which holds a scalable set of web servers, and client-1 and client-2, which are two sets of client machines with potentially separate policies (Figure 2).

Figure 2: Sample GBP Topology



Step 1: Setting up Rules and Rule Sets

```
# Create HTTP Rule

gbp policy-classifier-create web-traffic --protocol tcp --port-range 80 --direction in

gbp policy-rule-create web-policy-rule --classifier web-traffic --actions allow

# Create HTTPs Rule
```

```
gbp policy-classifier-create secure-web-traffic --protocol tcp --port-range 443 --direction in

gbp policy-rule-create secure-web-policy-rule --classifier secure-web-traffic --actions allow

# WEB RuleSet

gbp ruleset-create web-ruleset --policy-rules web-policy-rule
```

Step 1 creates a rule set describing a policy for a set of web servers. The rule set consists of a set of rules containing classifiers designed to match a portion of the traffic and actions for dealing with that traffic. Common actions include actions to allow or redirect to a network service.

Step 2: Creating Groups and Associating RuleSets

```
# Group creation

gbp group-create  web

gbp group-create  client-1

gbp group-create  client-2

# RuleSet Association

gbp group-update client-1 --consumed-rulesets "web-ruleset=scope"

gbp group-update client-2 --consumed-rulesets "web-ruleset=scope"

gbp group-update web --provided-rulesets "web-ruleset=scope"
```

Step 2 creates the groups and attaches the appropriate rule sets. Rule sets describe a bidirectional set of rules. However, the API is designed to allow a group to “provide” a rule set describing its behavior, and other groups to “consume” that rule set to connect to it. The model intends for groups to provide rule sets that describe their behavior, which other groups can then choose to access.

Step 3: Create Group Members

```
# Create members as needed

gbp member-create --group web web-1

gbp member-create --group client-1 client-1-1

gbp member-create --group client-2 client-2-1
```

Setp 3 creates a number of members within each group. Each member inherits all of the properties of the group to specify its connectivity and security requirements.

Group-Based Policy Architecture

Group-Based Policy was designed to live as a non-disruptive layer on top of existing OpenStack projects. In networking, GBP offers a number of potential backend solutions behind the GBP API.

- **Existing plug-ins**: Any existing Neutron plugin can be used through a mapping driver offered as part of GBP. This allows GBP to act as a frontend to existing Neutron constructs and existing plugins.

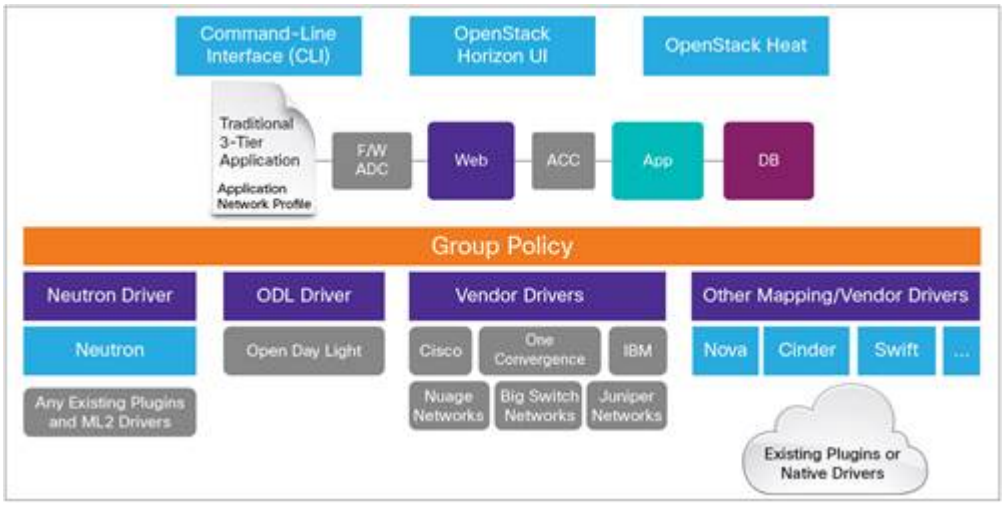
- **GBP in OpenDaylight (ODL):** As GBP has progressed in OpenStack, a corresponding project has been developed in the ODL community to build an open source network overlay solution using ODL and Open vSwitch (OVS). The GBP project can naturally support OpenDaylight in this configuration and allow it to act as a network controller through its existing southbound interfaces.

- **"Native" plug-ins**: A number of vendors including Big Switch, Cisco, IBM, Juniper, Midokura, Nuage Networks, and One Convergence have developed plans to build native drivers that directly absorb GBP APIs. This capability gives each back end flexibility to natively support policy constructs and offer advanced functionality such as service chaining.

Although the initial implementation of GBP covers networking and Neutron, the project is designed to offer an intent-driven API across all types of infrastructure, including compute and storage. For future OpenStack releases, the team plans to expand to cover those areas as well.

Group-based Policy is currently available on StackForge:
https://github.com/stackforge/group-based-policy
https://github.com/stackforge/group-based-policy-specs
https://github.com/stackforge/python-group-based-policy-client
https://github.com/stackforge/group-based-policy-ui
https://github.com/stackforge/group-based-policy-automation

Figure 3: OpenStack GBP Architecture



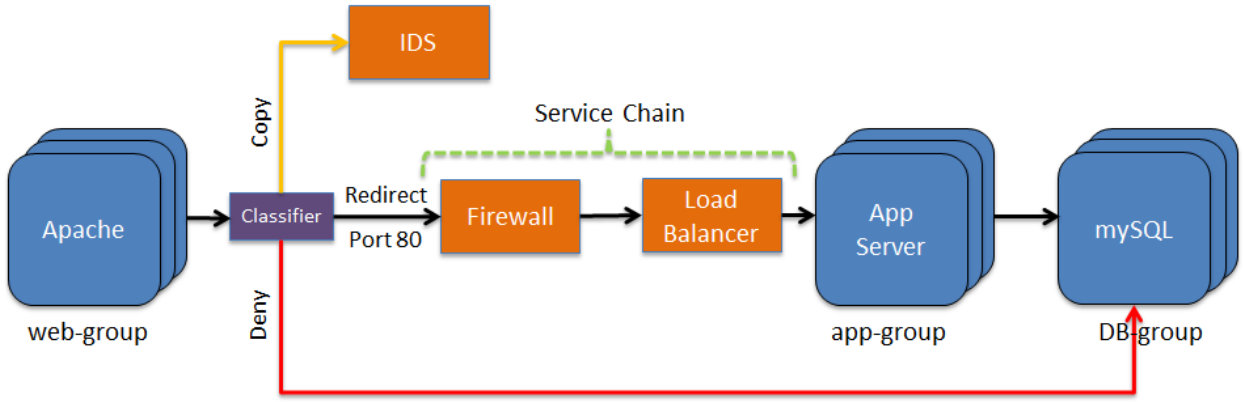
Service Insertion and Chaining Model

Network Services are an essential component of application deployment. The GBP model inherently provides constructs to specify network service requirements in an application policy. The GBP Services chaining model provides a simple API to define the topology of network services. The GBP services chain specification is independent of the network infrastructure model on which the services are implemented. The Services chaining API allows the insertion and chaining of services using primitives that are independent of services such as load balancers and firewalls. The model supports Layer 2, Layer 3, and TAP insertion modes to enable insertion of any network service.

The GBP API enables the application developer or the operator to specify the requirements of a chain of services between groups by using the service-chain as a redirect target in the policy definition. The versatility of the GBP API is demonstrated with the constructs that allow the operator to enforce availability and security policies by using additional service-chains to modulate the application developer's policy.

Figure 4 illustrates the use case for the Network Services chaining model in GBP. The section *Example: Using GBP policy* earlier in this document demonstrated the use of policy rule set to describe the connectivity requirements of an application such as the App Server. To deploy the App Server, the application administrator could specify a firewall service and a load balancer service to provide security and availability. The firewall and load balancer services are specified using a service chain resource. The application administrator creates a new policy rule set to redirect traffic to the service-chain. The original application policy rule set inherits the new service-chain policy rule set to specify the additional deployment requirements.

Figure 4: Network Services Chaining Model



Conclusion

OpenStack Group-based Policy was designed to offer a new set of abstractions to manage OpenStack infrastructure. The solution was designed to separate the intent of the application developer from the requirements of the infrastructure operators to offer a powerful, yet simple set of APIs. The solution runs on top of existing OpenStack services in a non-disruptive manner and has been developed by a community of engineers from Big Switch, Cisco, IBM, Juniper, Midokura, Nuage Networks, and One Convergence. GBP is available now on StackForge and is designed to work with the OpenStack Juno release.

For More Information:
https://wiki.openstack.org/wiki/GroupBasedPolicy