Group-Based Policy

This chapter contains the following sections:

- About Group-Based Policy, on page 1
- External Connectivity, on page 5
- Network Address Translation Pools and Floating IP Addresses, on page 6
- Example of Creating a Multi-Tier Application Using the Group-Based Policy CLI, on page 6
- Example of Creating a Multi-Tier Application Using the Group-Based Policy GUI, on page 8
- Group-Based Policy Deployment, on page 11

About Group-Based Policy

Group-Based Policy (GBP) is an API framework for OpenStack that offers an intent-driven model intended to describe application requirements in a way that is independent of the underlying infrastructure. Rather than offering network-centric constructs, such as Layer 2 domains, GBP introduces a generic "Group" primitive along with a policy model to describe connectivity, security, and network services between groups. While GBP has focused on the networking domain, it can be a generic framework that extends beyond networking.

GBP runs as a service plug-in within the Neutron process space.

Group-Based Policy Use Cases

Group-Based Policy (GBP) was designed to offer a powerful, but simple language for capturing the requirements of and deploying complex applications on OpenStack clouds. GBP addresses disconnect between application developers who understand application requirements and infrastructure teams who understand various infrastructure capabilities.

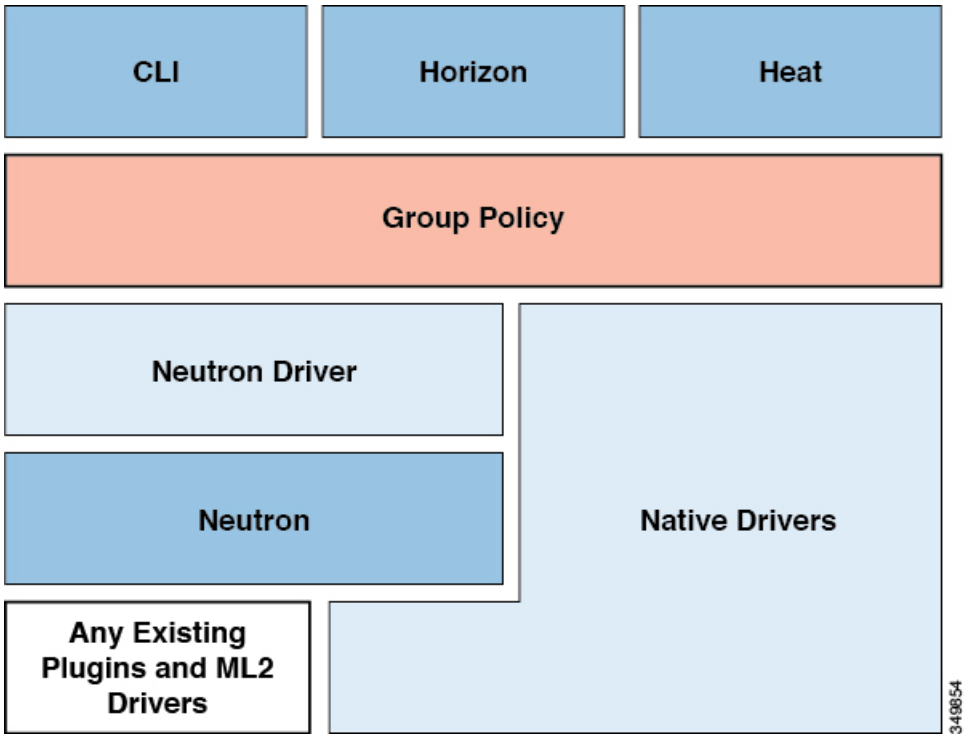
GBP offers the following capabilities beyond what is present in OpenStack:

Dependency mapping	GBP enables users to specify the relationships between different tiers of applications. This dependency map acts as documentation for the security requirements of the application and allows different tiers of the application to evolve separately. The dependency map also makes it extremely easy to scale and automate infrastructure.
Separation of concerns	GBP was designed to separate application security requirements, such as who can talk to whom, from network-specific requirements, such as what IP address ranges to use, where to draw network boundaries, and how to assign virtual IP addresses. This allows application, security, and operation teams to operate independently, but cooperatively.

How Group-Based Policy works

Group-Based Policy (GBP) offers a policy API through multiple OpenStack interfaces, including Horizon extensions (openstack-dashboard-gbp), Heat (openstack-heat-gbp), and cli (openstack-neutron-gbp). GBP was designed to act as a layer on top of Neutron.

GBP supports two forms of mapping to the underlying infrastructure:

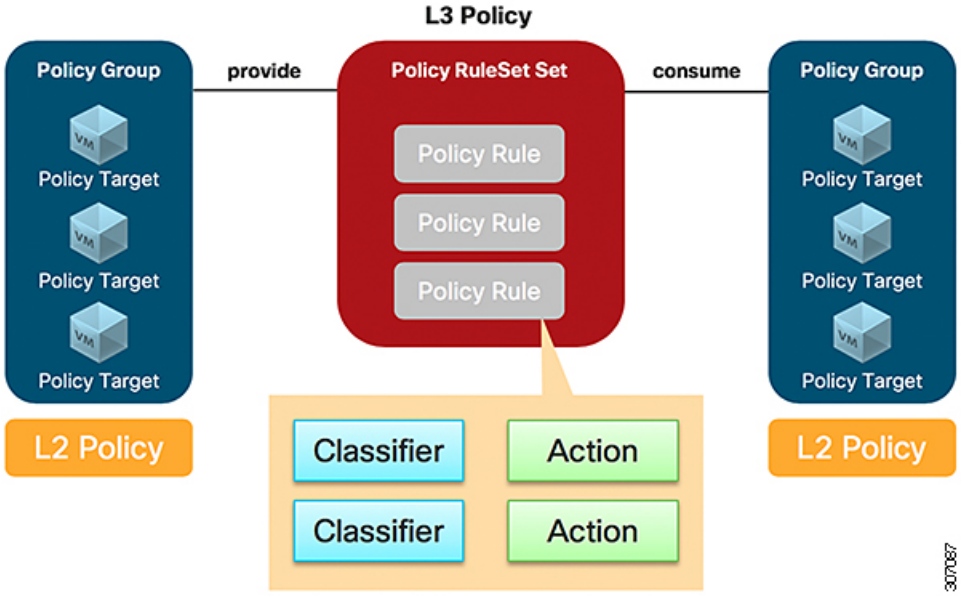


Neutron Mapping Driver	The Neutron mapping driver converts GBP resources into existing Neutron API calls. This allows Neutron to run any existing open source or vendor plugin, including ML2. It also allows GBP to be used in any OpenStack environment. You should define the OpenStack projects either with Neutron API or GBP API. Different projects of the same OpenStack installation could potentially use different APIs. For example, Neutron or GBP.
Native Drivers	You can create drivers that directly render policy constructs through a separate SDN controller or external entity without first converting them to Neutron APIs. This is valuable as it gives the controller additional flexibility on how to interpret and enforce policy without being tied to L2/L3 behaviors. There are currently four native drivers, including Cisco APIC, VMware NSX policy, and One Convergence NSP.

About the Group-Based Policy Model

Group-Based Policy (GBP) introduces a policy model to describe the relationships between different logical groups or tiers of an application. The primitives have been chosen in a way that separates their semantics from the underlying infrastructure capabilities. Resources can be public or local to a specific tenant.

The key primitives are:



Resource	Description
Policy Target	An individual network endpoint (generally a NIC). A policy target is a basic addressable unit in the architecture.
Policy Group	Policy targets with the same properties are organized into policy groups, which is the fundamental primitive of GBP. Policy groups offer an infrastructure agnostic grouping construct without specifying any network semantics in the same way as a broadcast. Each group models its dependencies by declaring the rule sets that it provides to groups as well as rule sets that it will consume.
Policy Classifier	A means of filtering network traffic, including protocol, port range, and direction (in, out, or bidirectional).
Policy Action	An action to take when a particular rule is applied. The supported type include "allow".
Policy Rules	Classifiers paired with actions.
Policy Rule Sets	Policy rule sets contain a number of policy rules. Rule sets can be nested through parent-child relationships.

About Network Policies

Group-Based Policy (GBP) aims to centralize the description of network policies and keep them separate from application-level policies, such as groups and rule sets. This allows a separation of concerns between application owners and cloud/infrastructure administrators.

Resource	Description
Layer 2 Policy	Specifies set of groups within the same switching domain. Layer 2 policies must reference a particular Layer 3 policy.

Resource	Description
Layer 3 Policy	Specifies potentially overlapping IP address space that contains any number of Layer 2 policies.
Network Service Policy	Specifies network-specific parameters that are required for network service chaining, such as VIP allocation.

About Shared Policies

Resources in the Group-Based Policy (GBP) model support a `shared` attribute that can be set to make a resource visible across tenants. The sharing of policy resources promotes the separation of concerns by allowing the relevant team to create a policy and for other users to consume the policy. A shared resource has global visibility, meaning that all tenants are able to see a shared resource. The `shared` attribute can be set or updated only by a user in the administrator role.

The following example creates a shared Layer 3 policy using the CLI:

```
# gbp l3policy-create --shared True my-shared-l3p
```

Similarly, the GUI has a checkbox that enables the sharing or unsharing a resource.

There are certain constraints on when and how a resource is shared:

- A shared resource can only be associated with other shared resources. For example, a shared Layer 2 policy can only exist on a shared Layer 3 policy.
- A shared resource cannot be unshared if it is currently being used.

Note A policy target resource does not support the `shared` attribute.

About the Neutron Mapping Driver

One of the most useful aspects of the Group-Based Policy (GBP) model and its implementation is the ability to map a policy directly into the Neutron API and thus be able to use the existing Neutron plugins as is. The default mapping is as follows:

GBP Resource	Neutron
Policy Target	Port
Policy Target Group	Subnet
Layer 2 Policy	Network
Layer 3 Policy	Router

Note You can design a custom mapping and implement it in a "resource mapping" policy driver.

External Connectivity

The Group-Based Policy (GBP) model supports an API that captures the user's intent to allow the policy targets to communicate with the external world. The following primitives are used to achieve this:

Resource	Description
External Segment	Models an external network that is defined by a Classless Inter-Domain Routing (CIDR) and any other networks that are reachable through this segment using a specified next-hop.
External Policy	Models a policy target group on a external segment. Think of this as an external policy target group that can provide and consume the policy rule set, but the difference is that no policy targets can be created on the external policy.
Neutron Mapping External Segment	The Neutron subnet that is created on a Neutron external network.

This example represents a configuration for an external network with the name "Datacenter-Out". By creating an external segment with the same name, you can automatically configure it with the relevant CIDR and other attributes.

The following command creates an external segment with the name "Datacenter-Out":

```
# export EXT_SEG_ID=$(gbp external-segment-create Datacenter-Out --shared
  True --external-route destination=0.0.0.0/0,nexthop= | awk "/ id / {print \$4}")
```

The following command creates an external policy that models a group that represents the external world:

```
# gbp external-policy-create my-external-policy --external-segments $EXT_SEG_ID
```

The following commands create a policy rule set to allow TCP access:

```
# gbp policy-classifier-create all-tcp-traffic --protocol tcp --direction in
# gbp policy-rule-create tcp-policy-rule --classifier all-tcp-traffic --actions allow
# gbp policy-rule-set-create   tcp-ruleset --policy-rules tcp-policy-rule
```

The following commands set up provide/consume relationships to allow all TCP traffic for the web policy target group to access the external world:

```
# gbp external-policy-update my-external-policy
  --provided-policy-rule-sets "tcp-ruleset=true"
```

This example assumes that the Layer 3 policy is called "default".

The following commands link the implicitly created Layer 3 policy for the web policy target group to the external segment that you created.

```
# gbp l3policy-update default -external-segment Datacenter-Out
```

Network Address Translation Pools and Floating IP Addresses

Each external segment can be configured with zero or more pools of IP addresses that can be used for assigning floating IP addresses. Floating IP addresses in this context is specific to the Neutron floating IP address resource.

Resource	Description
NAT Pool	A pool of IP addresses that is used to assign floating IP addresses for virtual machines.

In this example the **1.105.2.128/25** range can be anything as long it is routable. Also it does not have to correlate to a CIDR in a configuration.

The following command creates a NAT pool from the subnet that is used for the external subnet:

```
# gbp nat-pool-create --ip-version 4 --ip-pool 1.105.2.128/25
--external-segment Datacenter-Out my-nat-pool
```

The following command creates a Network Service Policy (NSP) that uses the NAT pool that you created by the previous command:

```
# gbp network-service-policy-create --network-service-params
type=ip_pool,name=nat_fip,value=nat_pool my-nat-pool-nsp
```

The following command associates the NSP that you created by the previous command with the web policy target group:

```
# gbp group-update web --network-service-policy my-nat-pool-nsp
```

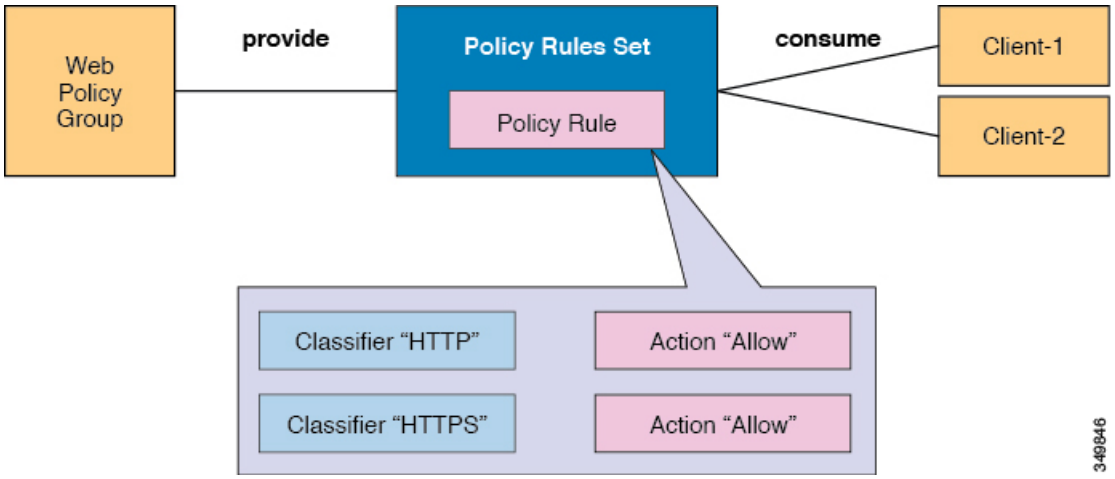
At this point, every new policy target that gets created in the web policy target group automatically gets a floating IP address assignment from the NAT pool.

The NAT pool can be any address range as long as it does not overlap with the Application Policy Infrastructure Controller (APIC) external network configuration as specified in the configuration file by the `cidr_exposed` and `host_pool_cidr` options.

The above workflow should not be used if the user wants to assign floating IP addresses explicitly only to specific members of a policy target group. In such cases, do not create an NSP, but instead follow the explicit workflow for associating a floating IP address to a Neutron port. Choose the floating IP address from the subnet that corresponds to the NAT pool that you created.

Example of Creating a Multi-Tier Application Using the Group-Based Policy CLI

The following example creates a simple policy using Group-Based Policy (GBP). This policy creates two groups and a policy rule set between them.



Procedure

Step 1 Set the rules and rule set, which describes a policy for a set of Web servers.

The rules contain classifiers designed to match a portion of the traffic and actions for dealing with that traffic. Common actions include actions to allow or redirect traffic to a network service.

a) Create the `allow` action:

```
# gbp policy-action-create allow --action-type allow
```

b) Create the 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
```

c) Create the 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
```

d) Create the Web rule set:

```
# gbp policy-rule-set-create web-ruleset --policy-rules "web-policy-rule
secure-web-policy-rule"
```

Step 2 Create the groups and associate the rule sets.

Rule sets describe a bidirectional set of rules. However, the API is designed to allow a group to "provide" a rule set that describes 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.

a) Create the groups:

```
# gbp group-create web
# gbp group-create client-1
# gbp group-create client-2
```

b) Associate the rule sets:

```
# gbp group-update client-1 --consumed-policy-rule-sets "web-ruleset=scope"
# gbp group-update client-2 --consumed-policy-rule-sets "web-ruleset=scope"
# gbp group-update web --provided-policy-rule-sets "web-ruleset=scope"
```

Step 3 Create a member within each of the previously-created groups.

Each member inherits all of the properties of the group to specify its connectivity and security requirements.

a) Create a policy target in the Web group, and extract the Neutron port that is associated with that policy target:

```
# export WEB_PORT=$(gbp policy-target-create web-pt-1 --policy-target-group web \
| awk "/port_id/ {print \$4}")
```

b) Create a Web group member (a virtual machine instance) using the Neutron port that you extracted earlier:

```
# nova boot --flavor m1.tiny --image image_name --nic port-id=$WEB_PORT web-vm-1
```

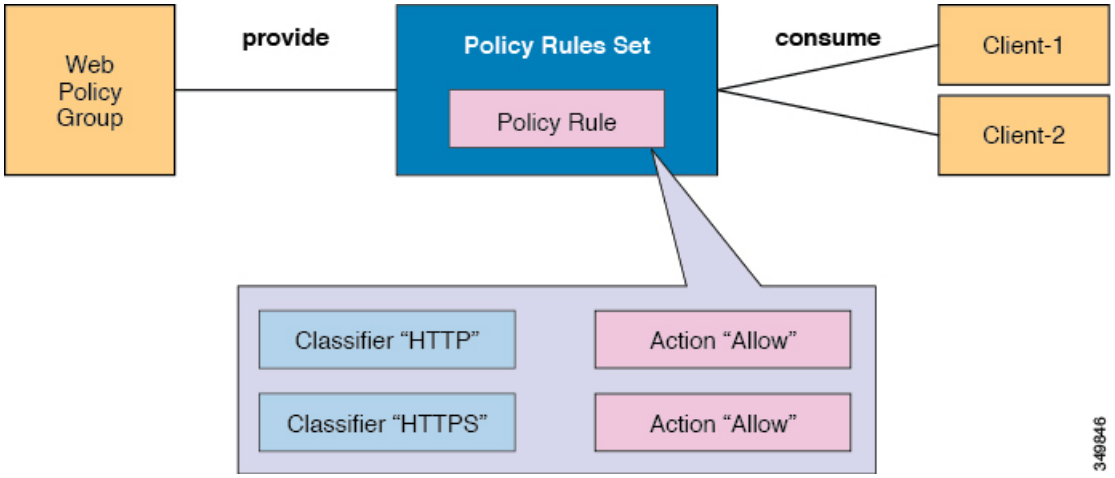
c) Create members in the `client-1` and `client-2` groups:

```
# export CLIENT1_PORT=$(gbp policy-target-create client-pt-1 \
--policy-target-groupclient-1 | awk "/port_id/ {print \$4}") nova boot \
--flavor m1.tiny --image image_name --nic port-id=$CLIENT1_PORT \
client-vm-1

# export CLIENT2_PORT=$(gbp policy-target-create client-pt-2 \
--policy-target-group client-2 | awk "/port_id/ {print \$4}") nova boot \
--flavor m1.tiny --image image_name --nic port-id=$CLIENT2_PORT \
client-vm-2
```

Example of Creating a Multi-Tier Application Using the Group-Based Policy GUI

The following example creates a simple policy for a multi-tier application using Group-Based Policy (GBP). This policy creates two groups and a policy rule set between them.



Procedure

Step 1 Log into the OpenStack platform GUI as an administrator.

Step 2 On the menu bar, choose **Project** > **Policy** > **Application Policy** > **Policy Rule Set**.

Step 3 In the **Policy Rule Set** pane, click **Create Policy Rule Set**.

Step 4 In the **Create Policy Rule Set** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy rule set (web-ruleset).
b) In the **Description** field, enter the description of the new policy rule set (web-ruleset).
c) Click **Next**.
d) In the **Policy Rules** field, click the **+** icon.

Step 5 In the **Create Policy-Rule** dialog box, perform the following actions to create a rule for HTTP:

a) In the **Name** field, enter the name of the new policy-rule (http-rule).
b) In the **Description** field, enter the description of the new policy-rule (http-rule).
c) Click **Next**.
d) In the **Policy Classifier** field, click the **+** icon.

Step 6 In the **Create Policy Classifier** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy classifier (http-classifier).
b) In the **Port/Range(min:max)** field, enter the port range (80).
c) In the **Direction** drop-down list, choose the direction (BI).
d) Click **Create Policy Classifier**.
e) Click **Next**.
f) In the **Policy Action** field, choose the action for your policy-rule (allow).
g) Click **Create**.

Step 7 In the **Create Policy Rule Set** dialog box, perform the following actions for HTTPS:

a) In the **Policy Rules** field, click the **+** icon.
b) In the **Name** field, enter the name of the new policy-rule (https-rule).
c) In the **Description** field, enter the description of the new policy-rule (https-rule).
d) Click **Next**.
e) In the **Policy Classifier** field, click the **+** icon.

Step 8 In the **Create Policy Classifier** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy classifier (https-classifier).
b) In the **Port/Range(min:max)** field, enter the port range (443).
c) In the **Direction** drop-down list, choose the direction (BI).
d) Click **Create Policy Classifier**.

Step 9 In the **Create Policy-Rule** dialog box, perform the following actions:

a) Click **Next**.
b) In the **Policy Action** field, choose the action for your policy-rule (allow).
c) Click **Create**.
d) In the **Policy Rules** field, choose the policy rules for your policy rule set (http-rule and https-rule).
e) Click **Create**.

Step 10 On the menu bar, choose **Project** > **Policy** > **Application Policy** > **Policy Rule Set**.

Step 11 In the **Policy Rule Set** pane, you can verify your rule set exists including the policy rules.

Step 12 Click **Policy Rules**.

Step 13 In the **Policy Rules** pane, you can verify your policy rules exists.

Step 14 Click **Policy Classifier**.

Step 15 In the **Policy Classifier** pane, you can verify your policy classifier exists.

Step 16 Click **Policy Actions**.

Step 17 In the **Policy Actions** pane, you can verify the previous created policy action exists.

Step 18 Create the policy target group. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 19 Click **Create Group**.

Step 20 In the **Create Group** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new group (web-group).
b) In the **Description** field, enter the description of the new group (web-group).
c) Click **Next**.
d) In the **Provided Policy Rule Set** field, choose the policy rule set for the group (web-ruleset).
e) Click **Next**.
f) In the **Network Policy** drop-down list, choose the network policy for the group (Default). Choosing **Default** creates a Layer 2 policy automatically that has the same name as the group.
g) Click **Create**.

Step 21 Create a second group for clients. Click **Create Group**.

Step 22 In the **Create Group** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new group for clients (client-group).
b) In the **Description** field, enter the description of the new group for clients (client-group).
c) Click **Next**.
d) In the **Consumed Policy Rule Set** field, choose the policy rule set to be consumed for the group (web-ruleset).
e) Click **Next**.
f) In the **Network Policy** field, choose the network policy for the group (Default). Choosing **Default** puts the group in the default Layer 2 policy that was implicitly created in Step 20, on page 10.
g) Click **Create**.

Step 23 On the menu bar, choose **Project** > **Policy** > **Network and Services' Policy**.

Step 24 In the **L3 Policy** pane, you can verify that the default Layer 3 policy was created.

Step 25 Click the network policy **default** to verify the Layer 2 policies were created.

Step 26 Create virtual machines. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 27 In the **Groups** pane, click on a group (web-group).

Step 28 In the **Members** pane, click **Create Member**.

Step 29 In the **Create Member** dialog box, perform the following actions:

a) In the **Instance Name** field, enter the instance name (web1).
b) In the **Instance Boot Source** drop-down list, choose the instance boot source (Boot from image).
c) In the **Image Name** drop-down list, choose the image (cirros-web).
d) Click **Launch**.
e) In the **Members** pane, you can verify the newly created group (web-group).

Step 30 Create the client virtual machine. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 31 In the **Groups** pane, click on a group (client-group).

Step 32 In the **Members** pane, click **Create Member**.

Step 33 In the **Create Member** dialog box, choose **Details**, and perform the following actions:

a) In the **Instance Name** field, enter the instance name (client1).
b) In the **Instance Boot Source** drop-down list, choose the instance boot source (Boot from image).
c) In the **Image Name** drop-down list, choose the image (cirros-web).
d) Click **Launch**.
e) In the **Members** pane, you can verify the newly created group (client-group).

Step 34 To view all instances, on the menu bar, choose **Admin** > **System** > **Instances**.

Group-Based Policy Deployment

For information on deploying Group-Based Policy (GBP), see: https://wiki.openstack.org/wiki/GroupBasedPolicy#Try_Group-based_Policy