F5 Neutron LBaaSv2 L7 Content Switching

Introduction

The OpenStack community introduced initial support for L7 content switching for Neutron LBaaS version 2 in the Mitaka release cycle. At present the F5 LBaaSv2 service provider driver for Mitaka does not implement the neutron-lbaas API to manage the L7 policies and rules that are the basis for this feature. This document describes the differences between the community L7 API and the BIG-IP Local Traffic Manager implementation of layer 7 traffic steering and how we can extend the F5 neutron-lbaas API to create, update and delete L7 policies and rules that would allow a user to manage virtual server traffic using the BIG-IP Local Traffic Policies.

Background

L7 content switching takes its name from the OSI Model, Layer 7, or the application layer. As its name implies switching decisions are based on the application data, or content, of request traffic as it passes through the virtual server. The API allows the operator to specify rules and policies that define the conditions to match in the content, and the actions to execute when a match is found. For more information about LBaaSv2 L7 content switching as proposed by the community see:

- https://wiki.openstack.org/wiki/Neutron/LBaaS/l7
- http://specs.openstack.org/openstack/neutron-specs/specs/mitaka/lbaas-l7-rules.html

Neutron LBaaSv2 API L7 Policies and Rules

In Neutron an L7 Policy is a collection of L7 rules associated with a Listener; it may also have an association to a back-end pool. Policies describe actions that should be taken by the load balancing software if all of the rules in the policy return true / match.

Policy Actions:

Valid L7 policy actions are one of:

- REJECT – The request is blocked and an appropriate response code is sent. The HTTP request is not forwarded to a backend pool.
- REDIRECT_TO_POOL – The request is forwarded to a member in the redirect pool.
- REDIRECT_TO_URL – The request is forward to the URL specified in the redirect URL.

Policy Rules:

An L7 Rule is a single, simple logical test that returns either true or false. An L7 rule has a type, a comparison type, a value, and an optional key that is used for certain rule types. Valid rule types are defined by the following operands:

- HOST_NAME – The rule does a comparison to the hostname in the HTTP ‘Host’ header with the specified *value* parameter.

- PATH – The rule compares the HTTP URL to specified *value* parameter
- FILE_TYPE – The rule compares the file extension of the HTTP URI with the specified rule *value* (e.g. 'pdf', 'jpg', etc.)
- HEADER -- The rule compares a HTTP header in the request, as specified by the *key* parameter, with the *value* specified in the rule.
- COOKIE – The rule searches for the cookie, as specified by the *key* parameter, and compares it to the rule's *value*.

In addition to a rule type there are five comparison types that are applied to the rule's operand and compared with the rule *value* to determine if a match exists.

- STARTS_WITH – operand starts with string
- ENDS_WITH – operand ends with string
- EQUAL_TO – operand matches the string
- CONTAINS – operand contains a substring value
- REGEX – operand matches the provided regular expression.

Policy Logic

All the rules in an L7 policy must match before the associated action is executed. So if a policy has a set of rules: R_1, R_2, … R_n, and an action, A, then the following logic holds:

If (R_1 *and* R_2 *and* … R_n) *then A*

Policy rules can also be negated by using the *–invert* parameter when specifying the rules. For example, the comparison type, EQUAL_TO can be transformed to NOT_EQUAL_TO, by specifying *–invert.*

L7 policies are ranked by a position value and are evaluated according to their rank. The first policy that evaluates to true is executed and all subsequent policies are skipped. Given a set of n policies, where policy P_n has a rank *n* and an action A_n, the following logic holds:

```
If ($P_1$) then $A_1$
Else if ($P_2$) then $A_2$
…
Else if ($P_n$) then $A_n$
Else:
   Send request to default pool
```

F5 LBaaSv2 API and LTM Policies and Rules

The Neutron L7 terminology does not directly align with the common vocabulary of BIG-IP Local Traffic Manager. In the BIG-IP LTM, policies also have a set of rules, but it is the rules that specify actions and not the policy. Also, policies attached to a virtual server on the BIG-IP are all evaluated regardless of the truth of the associated rules. In addition to this difference the BIG-

IP policies have no ordinal, it is the BIG-IP rules that have this attribute. Because of these confusing differences it is useful to attempt to define the terms as they apply to each domain.

Table 1 Neutron L7 to BIG-IP LTM

Neutron LBaaS L7	BIG-IP Local Traffic Manager
Policy	Policy Rules
Policy Action	Rule Action
Policy Position	Rule Ordinal
Rule	Rule Conditions

The BIG-IP LTM policy has a name, description, a set of rules, and a strategy on how those rule should be evaluated. In fact, L7 policies in OpenStack are more akin to a collection BIG-IP LTM policy rules that are evaluated with the 'First match' strategy.

The BIG-IP LTM rules have conditions, actions, and an ordinal and would need to be created based on the L7 policy and rule attributes.

Neutron LBaaSv2 API L7 Rules Implementation

A combination of L7Policy and L7Rule elements will be mapped to TMOS traffic policies and in the case of specific L7Rule compare_types, iRules.

The major reasons to implement LBaaS L7 Rules in TMOS traffic policies, instead of a pure iRule implementation, are:

1) Performance, all L7 Rule types map directly to TMOS traffic policy match conditions:

L7Rule type	TMOS traffic policy match condition
Hostname	HTTP Host
Path	HTTP URI + path
FileType	HTTP URI + extension
Header	HTTP Header
Cookie	HTTP Cookie

2) The LBaaS L7 Rules requirement that 'the first L7Policy that returns a match will be executed' directly maps to TMOS traffic policy execution strategy 'first'.

3) Four of the five L7Rule compare_type values directly map to TMOS traffic policy rule conditions:

L7Rule compare_type	L7 –invert specified	TMOS traffic policy rule match condition
STARTS_WITH	No	begins with
STARTS_WITH	Yes	does not begin with
ENDS_WITH	No	ends with
ENDS_WITH	Yes	does not end with
EQUAL_TO	No	is
EQUAL_TO	Yes	is not
CONTAINS	No	contains
CONTAINS	Yes	does not contain
REGEX	X	No Direct Mapping

The REGEX comparison type is the only one that doesn't map directly into the LTM rule comparison types. In order to implement it we would have to push iRules to the BIG-IP. It is expected that this work would be the last feature added to the L7 solution.

4) All L7Policy actions map directly to TMOS traffic policy rule actions

L7Policy action	TMOS traffic policy rule action
Reject	Reset traffic
RedirectToURL	Redirect
RedirectToPool	Forward traffic to pool

Since the L7 Rules are expected to match on each HTTP request, if a listener has an L7Policy attached, the presence of a OneConnect profile on the corresponding TMOS virtual server must be assured.

Initially, the support for Regex **will not** be available. Since there is a direct mapping of the other compare types in Neutron to BIG-IP LTM, and regular expressions can in some cases be expressed as the logical 'OR' of multiple string compares, we propose an initial implementation of L7 Policies and Rules without regular expressions with the understanding that it will be available as a feature in a future release.

Neutron LBaaSv2 API Regex Implementation

The question comes in, what to do with the regex L7 Rule compare_type. The recommendation is that the regex L7Policy actions map to an iRule that always gets applied if an L7 policy is enabled on a Listener.

The iRule should have a conditional check for the presence of a well-known variable for the corresponding L7 Rule compare_type. The well-known variable should be set using the TMOS traffic policy set variable action for the request if a non-mapped compare_type is present. If the variable is not set, no iRule logic should be implemented. Additionally the TMOS traffic policy should set variables indicating the match type, the requested value, the desired action, and optionally the desired action target to be used in the executed iRule logic.

The Regex iRule:

For the RegEx iRule, if the well know variable is present such that logic is executed, the presence of the match type variable should checked and the correct iRule command be issued to obtain the desired attribute to match and target action.

For the RegEx iRule it needs to be documented that the expression matching will be BRE matching conforming to the iRule string ***matches_regex*** operator only. No pre-validation of the regex expression requested will be done. The expression will be implemented directly in the iRule. This can lead to iRule execution errors and TCP connection resets. Such errors and TCP resets are a valid consequence to the tenant issuing invalid regex expressions.

Assumptions

- BIG-IP regex for policy are BRE based
- We will use the iRule tlc `matches_regex` operator for the regex rule
- BIG-IP supports all of the policy/rule directive
- Python regex library is adequate for our conversions and validations