Ruby meets iControl: Creating VIPs

Colin Walker, 2009-05-01

Whether you've listened to our Podcast talking with Jason Hoffman - Joyent CTO and long-time Ruby supporter - or not, you've very likely heard of Ruby and/or Ruby on Rails. Ruby is one of the more quickly growing languages out there on the net today. With a fervent fan-base, a strong support structure from said interested parties, and all the flexibility most people need, there are lots of reasons why people are, and have been, picking it up for their daily tasks for some time now.

According to ruby-lang.org Ruby is: "A dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write." If you want my personal opinion, I'd agree with that completely. Even though the syntax is a bit different for me and my Perl/PHP/TCL based brain to adjust to, it's definitely elegant and productive. You don't have to take my word for it though, just ask one of the huge number of Ruby converts that's helping to drive this language to great things.

It should come as no surprise to repeat readers that we at F5 like doing cool things with cool languages and cool products. As such it's only natural that Ruby should become one of several languages being used with iControl to perform awesome automation, management and administration functions on the BIG-IP. Just as Perl, Python and Powershell before it, Ruby is a powerful, although possibly less known language for building iControl scripts. It's time that "less known" part changed.

Lucky for us there are fantastic DevCentral users like F5's mkelly who contribute outstanding content such as the Ruby VIP Maker which is the first Ruby / iControl script out of several newly contributed examples that I hope to share to spread the word about the power of iControl via Ruby.

This provides an easily usable tool to create or delete any number of VIPs on your BIG-IP without the need for repeated commands or clicks in the GUI. This is something that's quite common when setting up environments for testing, or mirroring partial configs from one system to another. Having an easy to use tool that makes this simple and fast could save valuable time and effort. The instructions are simple - just provide a prefix, the number of VIPs to create/delete and some extra IP/Port info when creating - but there is a lot more going on under the covers that's worth looking into.

The entire solution consists of multiple files and far too many lines of code to post here, but here are a couple of glimpses at what's going on. In this first snippet you can see the beginning of the createVips section where the prefix, number of VIPs to be created, starting IP and port are passed in and preparations are made for passing the iControl calls to the BIG-IP.

```
def createVips(namePrefix, number, startIP, port)

  puts " "
  puts " "
  puts "create VIPs"

  #set up the variables
  protocol = CommonProtocolType.new("PROTOCOL_TCP")
  wildmask = "255.255.255.255"
  pool_names = Array.new
  vs_names = Array.new

  startIP =~ /\s*(\d+)\.(\d*)\.(\d*)\.(\d*)\s*/
  currIP = Array.new
  currIP[0] = $1.to_i
  currIP[1] = $2.to_i
  currIP[2] = $3.to_i
  currIP[3] = $4.to_i
```

In the next snippet below you can see the crunching done just prior to sending through the definitions and other needed info for the actual VIP creation. Note the math being done to sequentially work up from the starting IP address inside the do loop, as well as the way that these are effectively being queued up for batch processing inside the bigipDriver.create call. Pretty cool stuff.

```
    1.upto(number) do |x|

      ipString = "#{currIP[0]}.#{currIP[1]}.#{currIP[2]}.#{currIP[3]}"
      vs_name = "#{namePrefix}_vs_#{x}"
      pool_name = "#{namePrefix}_#{x}"

      definitions << CommonVirtualServerDefinition.new(vs_name, ipString, port, protocol)
      wildmasks << wildmask
      resources << LocalLBVirtualServerVirtualServerResource.new(rtype, pool_name)
      profiles << LocalLBVirtualServerVirtualServerProfileSequence.new

      vipIpport = CommonIPPortDefinition.new(ipString, port)
      monitor_member = LocalLBMonitorIPPort.new(LocalLBAddressType.new("ATYPE_EXPLICIT_ADDRESS_EXPLIC
      monitor = LocalLBPoolMemberMemberMonitorAssociation.new(monitor_member, monitor_rule )
      monitors = LocalLBPoolMemberMemberMonitorAssociationSequence.new << monitor
      monitor_associations << monitors
      pool_names << pool_name

      currIP[3] += 1
      if currIP[3] > 253
        currIP[3] = 0;
        currIP[2] += 1
      end
      if currIP[2] > 254
        currIP[2] = 0;
        currIP[1] += 1
      end
      if currIP[1] > 254
        currIP[1] = 0;
        currIP[0] += 1
      end
      if currIP[0] > 254
        puts "ERROR! Ran out of IPs after only #{namecount} IPs, bailing out."
        return
      end
    end

    # create the user on the bigIP
    begin

      result = @bigipDriver.create(definitions, wildmasks, resources, profiles)
      puts "Vips created."
...
```

So there's the first of several iControl / Ruby examples that have been contributed to the CodeShare that I'm hoping to highlight to show off a little more of what can be done with Ruby and iControl and how you can go about it. This is a great example of taking a commonly used task, such as creating multiple VIPs/pools, and automating it using Ruby as the means to get the job done. As you'll see in the zip attached in the CodeShare entry there is a lot more to it than just the above code, so be sure to go download it and check it out for yourself. Many, many thanks to mkelly for doing all the heavy lifting with these examples. You've gotta love the power of a community.

Get the Flash Player to see this player.
20090105-rubyvipmaker.mp3

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com