**To TEST Load balancer**

global

log /dev/log syslog notice

maxconn 24000

#chroot /usr/share/haproxy

user haproxy

group haproxy

daemon

stats socket /var/run/haproxy.sock mode 600 level admin

#debug

#quiet

defaults

log global

mode http

option httplog

option dontlognull

retries 3

option redispatch

option httpclose

option abortonclose

maxconn 2000

timeout connect 7s

timeout queue 30s

timeout client 300s

timeout server 300s

option httpchk

listen admin

bind \*:{{ loadbalancer\_monitor\_port }}

stats enable

stats uri /admin

# We're only connecting via ssh

# stats auth haproxy:{{ admin\_secret|default('notsosecret') }}

{% for front\_end in playbook\_plones %}

listen {{ front\_end.plone\_instance\_name|default(plone\_instance\_name) }}

bind \*:{{ front\_end.loadbalancer\_port|default(loadbalancer\_port) }}

default-server {{ front\_end.loadbalancer\_options|default(loadbalancer\_options) }}

{% for client in range(0, front\_end.plone\_client\_count|default(plone\_client\_count)|int) %}

server client{{ client + 1 }} 127.0.0.1:{{ "%s" % (front\_end.plone\_client\_base\_port|default(plone\_client\_base\_port)|int + client) }} {% if front\_end.get('loadbalancer\_healthcheck', True) %}check{% endif %}

{% endfor %}

{% endfor %}

**HA PROXY**

Before we start I just want to point that the example\_haproxy.(ini|yml|Vagrantfile) set was created as a draft to this example, so it will not be approached here. However the following Vagrantfile snippet, which is part of it, is still worth mentioning since it automates the whole provisioning process (ie, no need to run ansible-playbook after vagrant up) while also showing the use of advance settings for Ansible in Vagrant.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # provisioning using Ansible  config.vm.provision "ansible" do |ansible|  ansible.playbook = "example\_haproxy.yml"  ansible.inventory\_path = "example\_haproxy.ini"  # when using an inventory file, the path to the private key must also be specified  # either as an argument or in the inventory file itself (which it is)  #ansible.raw\_arguments = [  # "--private-key=./.vagrant/machines/default/virtualbox/private\_key"  #]  end |

Now for the load balancer example. In this example we deploy 2 nginx webservers and 1 HAProxy reverse proxy for load balancing.

Let us start by running it first:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | # Pick the Vagrantfile for this example  ln -f -s example\_load\_balanced\_website.Vagrantfile Vagrantfile  # Start the VM instances for the webservers and load balancer  vagrant up  # Refresh SSH fingerprints for the 192.168.22.5x range on the host, otherwise  # Ansible would fail during provisioning with the message:  # "SSH encountered an unknown error during the connection. ..."  ansible-playbook -i ../refresh\_ssh\_public\_keys/localhost.ini ../refresh\_ssh\_public\_keys/main.yml  # Perform the provisioning  ansible-playbook -i example\_load\_balanced\_website.ini example\_load\_balanced\_website.yml  # Check HAProxy stats  http://localhost:8080/haproxy?stats  # Access the website:  http://localhost:8080/ |

And if you refresh the page you’ll see that it gets served by a different webserver.

This Vagrantfile is responsible for providing the 3 machines:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | # -\*- mode: ruby -\*-  # vi: set ft=ruby :  Vagrant.configure(2) do |config|  # Choose a box with VBox guest tools already installed  #config.vm.box = "debian/jessie64"  config.vm.box = "ubuntu/wily64"  # Set up hostname  config.vm.hostname = "ansible-nginx"  # Message shown on vagrant up  config.vm.post\_up\_message = "After provisioning check the website at http://localhost:8080/"  # Share an additional folder with the guest VM.  host\_folder = ENV['HOME'] + "/home/downloads/share\_vagrant"  guest\_folder = "/shared/"  config.vm.synced\_folder host\_folder, guest\_folder  # Fine tune the virtualbox VM  config.vm.provider "virtualbox" do |vb|  vb.customize [  "modifyvm", :id,  "--cpus", "2",  "--cpuexecutioncap", "50",  "--memory", "512",  ]  end  # fix annoyance, http://foo-o-rama.com/vagrant--stdin-is-not-a-tty--fix.html  config.vm.provision "fix-no-tty", type: "shell" do |s|  s.privileged = false  s.inline = "sudo sed -i '/tty/!s/mesg n/tty -s \\&\\& mesg n/' /root/.profile"  end  # fix annoyance, http://serverfault.com/questions/500764/dpkg-reconfigure-unable-to-re-open-stdin-no-file-or-directory  config.vm.provision "shell", inline: "echo 'export DEBIAN\_FRONTEND=noninteractive' >> /root/.profile"  config.vm.provision "shell", inline: "for user in /home/\*; do echo 'export DEBIAN\_FRONTEND=noninteractive' >> $user/.profile; done"  #####################################  # multi-machine environment specific  #####################################  # web servers  (1..2).each do |i|  config.vm.define "web#{i}" do |web|  web.vm.hostname = "web#{i}"  # Assign a static IP to the guest  web.vm.network :private\_network, ip: "192.168.22.5#{i}"  # Create a forwarded port mapping  web.vm.network "forwarded\_port", guest: 80, host: "808#{i}"  # web server specific provisioning  web.vm.provision :shell, inline: "echo 'Web Server #{web.vm.hostname} reporting for duty.'"  end  end  # lb server  config.vm.define "lb" do |lb|  lb.vm.hostname = "lb"  # Assign a static IP to the guest  lb.vm.network :private\_network, ip: "192.168.22.50"  # Create a forwarded port mapping  lb.vm.network "forwarded\_port", guest: 80, host: "8080"  # override default settings  lb.vm.provider "virtualbox" do |vb|  vb.memory = "256"  end  # lb server specific provisioning  lb.vm.provision :shell, inline: "echo 'Load Balancer #{lb.vm.hostname} ready to distribute workload.'"  end  end |

After the provision finishes it’s then possible to ssh to the different machines using vagrant ssh $machine\_name, as well as running the playbook. The main playbook example\_load\_balanced\_website.yml has two plays. The first deploys the webservers and the second the load balancer.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ---  - name: deploy webservers  hosts: webservers  become: yes  become\_method: sudo  roles:  - common  - nginx  - name: deploy loadbalancer  hosts: lbservers  become: yes  become\_method: sudo  roles:  - common  - haproxy |

The inventory file example\_load\_balanced\_website.ini defines the lbservers and webservers host groups:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [lbservers]  lb ansible\_ssh\_host=192.168.22.50 ansible\_ssh\_port=22 ansible\_ssh\_user='vagrant' ansible\_ssh\_private\_key\_file='./.vagrant/machines/lb/virtualbox/private\_key'  [webservers]  web1 ansible\_ssh\_host=192.168.22.51 ansible\_ssh\_port=22 ansible\_ssh\_user='vagrant' ansible\_ssh\_private\_key\_file='./.vagrant/machines/web1/virtualbox/private\_key'  web2 ansible\_ssh\_host=192.168.22.52 ansible\_ssh\_port=22 ansible\_ssh\_user='vagrant' ansible\_ssh\_private\_key\_file='./.vagrant/machines/web2/virtualbox/private\_key' |

And the config file ansible.cfg enables persistent caching using redis so that we can use the hostvarsmagic variable when configuring load balancing:

|  |  |
| --- | --- |
| 1  2  3 | gathering = smart  fact\_caching = redis  fact\_caching\_timeout = 86400 |

The other variables used are defined in group\_vars/all, group\_vars/webservers, and group\_vars/lbservers:

|  |  |
| --- | --- |
| 1  2 | ---  website\_port: 80 |
| 1  2  3 | ---  website\_root: /var/www/mysite  # website\_port is declared in the all group\_vars since it's also used by the LB |
| 1  2  3  4  5 | ---  backend\_name: backend\_lbservers  daemon\_name: proxy\_daemon  balance: roundrobin  lb\_listen\_port: 80 |

The haproxy main playbook installs the package and notifies the handlers to restart the needed services:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | ---  # Install latest version of haproxy package.  # Cache is not updated here since that is done in the common role.  - name: install latest haproxy  apt: name=haproxy state=latest update\_cache=no  notify: restart haproxy  # Enable haproxy to start at boot.  - name: enable haproxy  service: name=haproxy enabled=yes  # Configure haproxy settings.  - name: configure haproxy settings  template: src=haproxy.cfg.j2 dest=/etc/haproxy/haproxy.cfg  # we need to restart rsyslog to enable haproxy logging to /var/log/haproxy.log  # https://serverfault.com/questions/645924/haproxy-logging-to-syslog/751631#751631  notify:  - restart rsyslog  - restart haproxy |