redhat.

Road to DevOps: From SysAdmin to Cloud

Automation 101

Alex Callejas
Technical Account Manager | GCS México
September, 2018

About me

Alex Callejas
Technical Account Manager @Red Hat

@dark_axl

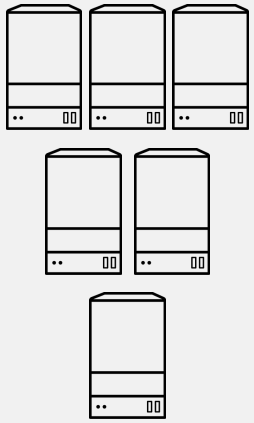
/rootzilopochtli

www.rootzilopochtli.com

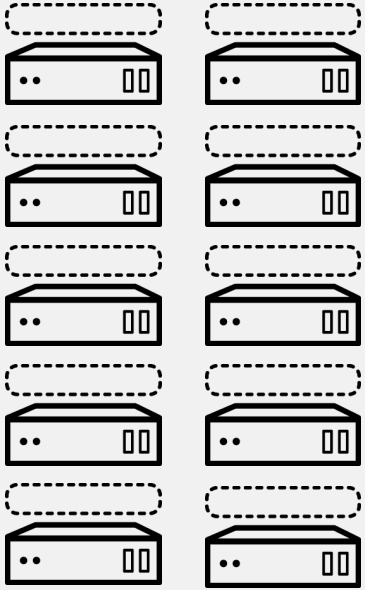
Geek by nature, Linux by choice, Fedora of course!

The time has changed

+15 years ago

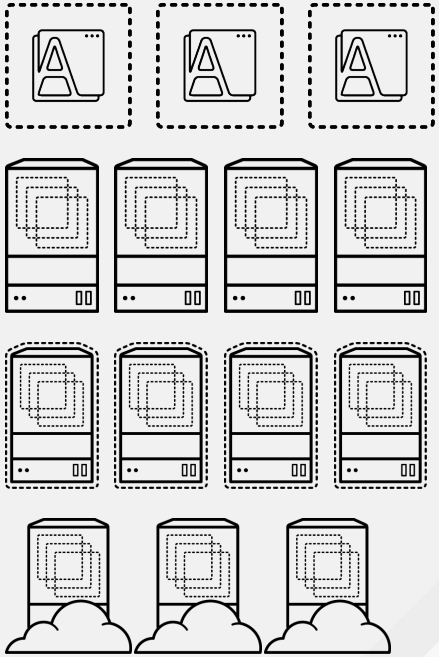


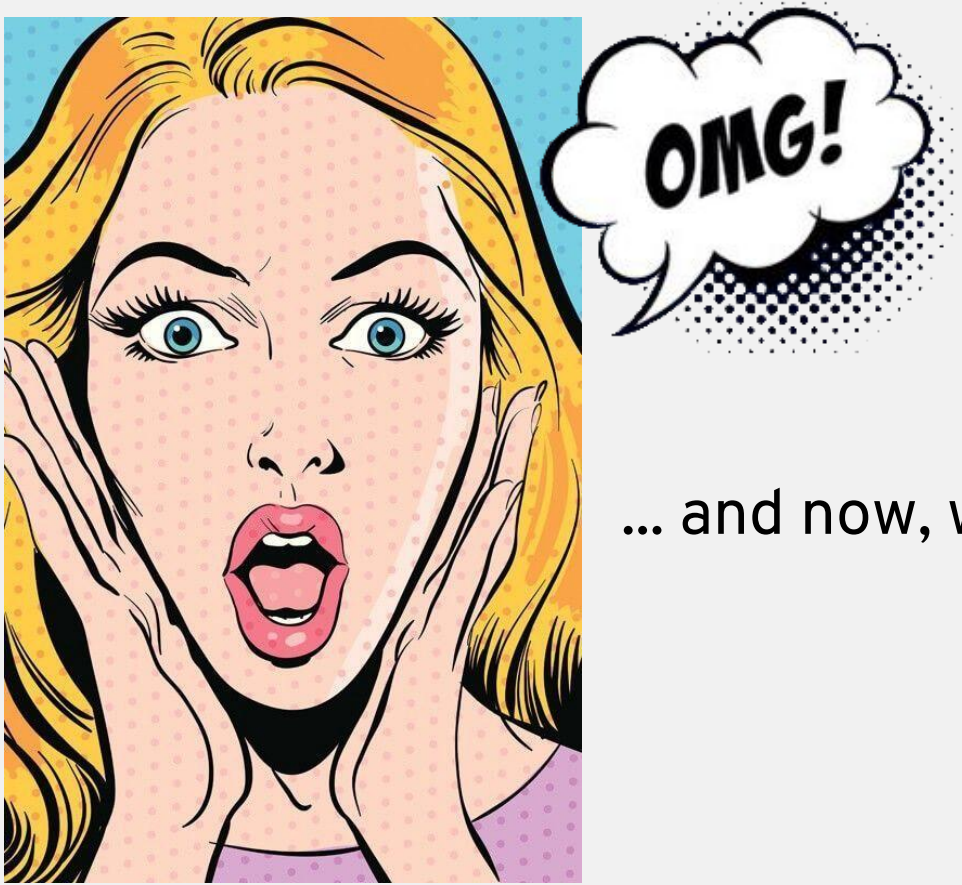
10 years ago



5 years ago

Today





... and now, we're going to the cloud

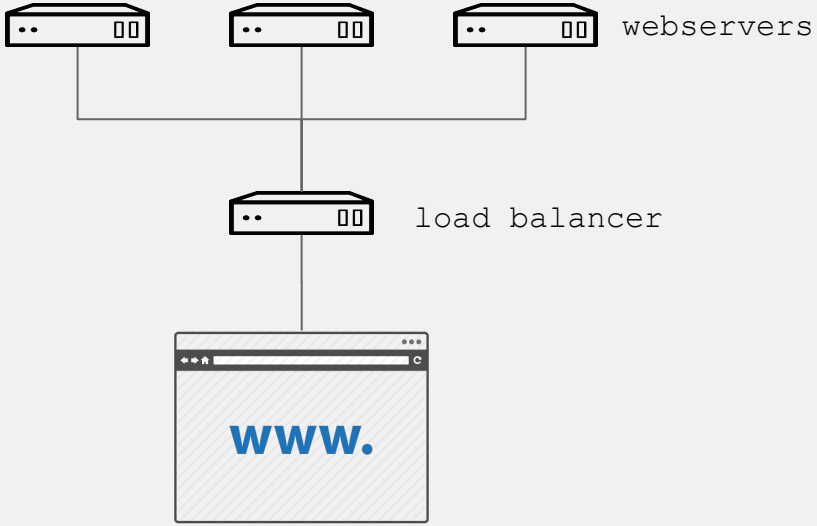
But don't worry, it's nothing that the practice can't solve

How to start?

simple

Simplicity is the ultimate sophistication.
Leonardo Da Vinci

A simple and basic example



Prerequisites

- Account on Google Cloud Platform
- Ansible installed on management node
  - Google Cloud Platform Guide

- We must also create a pair of RSA keys:

```
$ ssh-keygen -t rsa -b 4096 -f <rsa key file>
```

- If you use RHEL instances, you need subscribed with full support from Red Hat
  - Getting started with Red Hat Cloud Access

The role

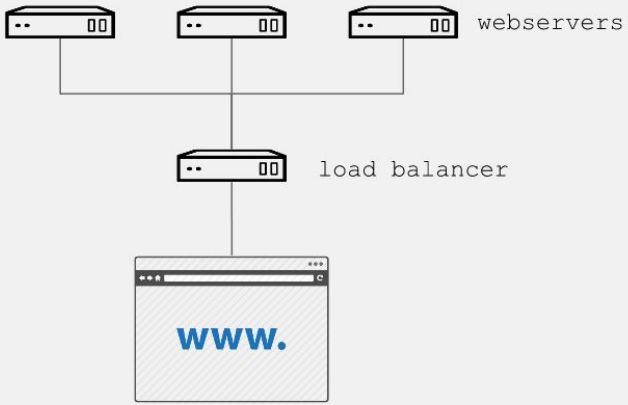
- Install and enable apache and firewalld
- Configure apache with a start page that shows the ip of each gce instance, for example:

```
$ cat apache_indexhtml.j2
<!-- {{ ansible_managed }} -->
<html>
<head><title>Apache is running!</title></head>
<body>
<h1>
Hello from {{ inventory_hostname }}
</h1>
</body>
</html>
$
```

- Open the http port (80)
- Restart apache and firewalld to confirm the configuration

The playbooks

1. The firewall rule to allow http traffic to our instances
2. Three instances based on Red Hat Enterprise Linux or CentOS 7, for the preparation of each instance, the aforementioned role will be used
3. The load balancer, indicating the name of our backend instances



The playbooks

```
$ cat gce-apache.yml
---
- name: Create gce webserver instances
  hosts: localhost
  connection: local
  gather_facts: True

  vars:
       service_account_email:   <Your gce service account email>
       credentials_file:   <Your json credentials  file>
       project_id: <Your project id>
       instance_names:   web1,web2,web3
       machine_type: n1-standard-1
       image: rhel-7 < centos-7 >

  tasks:
  - name: Create firewall rule to allow http traffic
       gce_net:
       name: default
       fwname: "my-http-fw-rule"
       allowed: tcp:80
       state: present
       src_range: "0.0.0.0/0"
       target_tags: "http-server"
       service_account_email: "{{ service_account_email  }}"
       credentials_file: "{{ credentials_file  }}"
       project_id: "{{ project_id }}"
```

The playbooks

```
- name: Create instances based on image {{ image }}
    gce:
    instance_names: "{{ instance_names }}"
    machine_type: "{{ machine_type }}"
    image: "{{ image }}"
    state: present
    preemptible: true
    tags: http-server
    service_account_email: "{{ service_account_email }}"
    credentials_file: "{{ credentials_file }}"
    project_id: "{{ project_id }}"
    metadata: '{"sshKeys":"<Y our  gce user: Your id_rsa_public  key>"}'
    register: gce

- name: Save hosts data within a group
    add_host:
    hostname: "{{ item.public_ip }}"
    groupname: gce_instances_temp
    with_items: "{{ gce.instance_data }}"
```

Note: *We must wait for the SSH port to be available, since if it is not listening, the playbook can send us an error and not execute the subsequent tasks and inject our previously created RSA public key to perform the post-creation tasks.*

The playbooks

```
    - name: Wait for ssh to come up
        wait_for: host={{ item.public_ip }} port=22 delay=10 timeout=60
        with_items: "{{ gce.instance_data }}"

    - name: Setting ip as instance fact
        set_fact: host={{ item.public_ip }}
        with_items: "{{ gce.instance_data }}"

    - name: Configure instance post-creation
        hosts: gce_instances_temp
        gather_facts: True
        remote_user: <Your gce user>
        become: yes
        become_method: sudo

        roles:
        - <path_to_role>/myapache
$
```

The playbooks

```
$ cat gce-lb.yml

---
- name: Playbook to create gce load balancing instance
  hosts: localhost
  connection: local
  gather_facts: True

  vars:
        service_account_email:  < Your gce service account email>
        credentials_file:  < Your json credentials  file>
        project_id: < Your project id>

 tasks:
        - name: Create gce load balancer
        gce_lb:
        name: lbserver
        state: present
        region: us-central1
        members: ['us-central1-a/web1','us-central1-a/web2','us-central1-a/web3']
        httphealthcheck_name: hc
        httphealthcheck_port: 80
        httphealthcheck_path: "/"
        service_account_email: "{{ service_account_email }}"
        credentials_file: "{{ credentials_file }}"
        project_id: "{{ project_id }}"
$
```

The playbooks

We use the following playbook to join both tasks and obtain the simple instances of GCE Red Hat Enterprise Linux / Apache with load balancing:

```
$ cat gce-lb-apache.yml

---
# Playbook to create simple instances of gce rhel/apache with load balancing
- import_playbook: gce-apache.yml
- import_playbook: gce-lb.yml
$
```

The playbooks

Run the playbook:

```
$ ansible-playbook gce-lb-apache.yml --key-file <Your_id_rsa_key>
```

We’re ready!!

Next steps

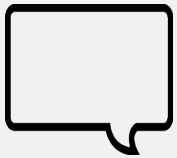
RED HAT®
DEVELOPER
PROGRAM

Join Us!

You only need a mail account: developers.redhat.com

Try it

Get the trial: cloud.google.com



Share

Share your doubts, scripts, playbooks, github, tricks, etc.

Source

- **Red Hat's TAM Blog**:
  - Creating a load-balanced web service on cloud with Ansible

Other Posts:

- OpenSource.com:
  - A sysadmin's guide to SELinux: 42 answers to the big questions
  - A sysadmin's handy cheat sheet for SELinux

redhat.

THANK YOU!

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos