Provisioning Virtual Networks using Junos, Vagrant and Ansible

Alexandros Kosiaris <akosiaris@gmail.com>

GRNOG - 3rd Technical Meeting

10 Jun 2016

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more
  - Download OS/Install OS/Reboot/Setup OS/Reboot/Reboot/Reboot

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more
  - Download OS/Install OS/Reboot/Setup OS/Reboot/Reboot/Reboot
  - Fix Errors, Rinse, Repeat

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more
  - Download OS/Install OS/Reboot/Setup OS/Reboot/Reboot/Reboot
  - Fix Errors, Rinse, Repeat
- Causes divergence in infrastructure

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more
  - Download OS/Install OS/Reboot/Setup OS/Reboot/Reboot/Reboot
  - Fix Errors, Rinse, Repeat
- Causes divergence in infrastructure
  - People tend to do things in different ways

Problem statement

Provisioning a new server/switch/router/VM has traditionally been a hard process

- Time/Resource consuming
  - At least one person allocated to it, if not more
  - Download OS/Install OS/Reboot/Setup OS/Reboot/Reboot/Reboot
  - Fix Errors, Rinse, Repeat
- Causes divergence in infrastructure
  - People tend to do things in different ways
  - Up to X * Y * Z possible configurations (X=# people, Y=# number of settings, Z=# ways of doing something)

Solution?

!!!Automation!!!

Solutions

Various approaches of automation at multiple levels:

Solutions

Various approaches of automation at multiple levels:

- At the image Level (Norton Ghost, Live Image, VMWare, Virtualbox, EC2, etc) - Cattle vs Pets!

Solutions

Various approaches of automation at multiple levels:

- At the image Level (Norton Ghost, Live Image, VMWare, Virtualbox, EC2, etc) - Cattle vs Pets!
- At the installation level (Unattended Installation, Debian preseed, FAI, Kickstart, custom)

Solutions

Various approaches of automation at multiple levels:

- At the image Level (Norton Ghost, Live Image, VMWare, Virtualbox, EC2, etc) - Cattle vs Pets!
- At the installation level (Unattended Installation, Debian preseed, FAI, Kickstart, custom)
- At the configuration level (CFEngine, Puppet, Chef, Ansible, custom) - Configuration as code!

Solutions

Various approaches of automation at multiple levels:

- At the image Level (Norton Ghost, Live Image, VMWare, Virtualbox, EC2, etc) - Cattle vs Pets!
- At the installation level (Unattended Installation, Debian preseed, FAI, Kickstart, custom)
- At the configuration level (CFEngine, Puppet, Chef, Ansible, custom) - Configuration as code!

And their combinations whenever applicable!!!

An informal survey

4 different classes (as devised by the presenter)

- No automation at all
- Image/Installation level automation
- Configuration Level automation
- Combination of 2 and 3

What is Vagrant?

Wikipedia says:

Vagrant is computer software that creates and configures virtual development environments. It can be seen as a higher-level wrapper around virtualization software such as VirtualBox, VMware, KVM and Linux Containers (LXC), and around configuration management software such as Ansible, Chef, Salt, and Puppet.

What is Vagrant?

Wikipedia says:

Vagrant is computer software that creates and configures virtual development environments. It can be seen as a higher-level wrapper around virtualization software such as VirtualBox, VMware, KVM and Linux Containers (LXC), and around configuration management software such as Ansible, Chef, Salt, and Puppet.

Sounds great! Now, WHAT ACTUALLY IS Vagrant?

WHAT IS VAGRANT?

In the presenter's own words ?

WHAT IS VAGRANT?

In the presenter's own words ?

- A framework AND a tool

WHAT IS VAGRANT?

In the presenter's own words ?

- A framework AND a tool
- A really easy way to spin up a fleet of VMs (and/or containers)

WHAT IS VAGRANT?

In the presenter's own words ?

- A framework AND a tool
- A really easy way to spin up a fleet of VMs (and/or containers)
- A way to manage the lifecycle of VMs

WHAT IS VAGRANT?

In the presenter's own words ?

- A framework AND a tool
- A really easy way to spin up a fleet of VMs (and/or containers)
- A way to manage the lifecycle of VMs
- A way to provision, ensure and manage the configuration of VMs

WHAT IS VAGRANT?

In the presenter's own words ?

- A framework AND a tool
- A really easy way to spin up a fleet of VMs (and/or containers)
- A way to manage the lifecycle of VMs
- A way to provision, ensure and manage the configuration of VMs
- A tool to make working with systems reproducible, less buggy and faster

A tool to just make my (our?) lives easier

Vagrant as a Framework

It's a framework. It's scaffolding and plugins, plugins, plugins

Vagrant as a Framework

It's a framework. It's scaffolding and plugins, plugins, plugins

- Plugins to manage VMs, called Providers (Virtualbox, VMware, EC2, Ganeti, Azure, Rackspace, Docker, you name it)

Vagrant as a Framework

It's a framework. It's scaffolding and plugins, plugins, plugins

- Plugins to manage VMs, called Providers (Virtualbox, VMware, EC2, Ganeti, Azure, Rackspace, Docker, you name it)
- Plugins to provision VMs, called Provisioners (Puppet, Chef, Ansible, fabric, docker compose, you write it)

Vagrant as a Framework

It's a framework. It's scaffolding and plugins, plugins, plugins

- Plugins to manage VMs, called Providers (Virtualbox, VMware, EC2, Ganeti, Azure, Rackspace, Docker, you name it)
- Plugins to provision VMs, called Provisioners (Puppet, Chef, Ansible, fabric, docker compose, you write it)
- Plugins to "know" VMs (Linux, Windows, BSDs, CoreOS, and others)

Vagrant as a Framework

It's a framework. It's scaffolding and plugins, plugins, plugins

- Plugins to manage VMs, called Providers (Virtualbox, VMware, EC2, Ganeti, Azure, Rackspace, Docker, you name it)
- Plugins to provision VMs, called Provisioners (Puppet, Chef, Ansible, fabric, docker compose, you write it)
- Plugins to "know" VMs (Linux, Windows, BSDs, CoreOS, and others)
- Plugins to communicate with VMs (SSH, WinRM, Shared folders, etc)

And a growing set of things like DNS management, Host communication, snapshots and others

Vagrant as a Tool

It's a framework. But it does ship with multiple plugins by default

Vagrant as a Tool

It's a framework. But it does ship with multiple plugins by default

- Providers: Docker, HyperV, Virtualbox. Virtualbox is the default

Vagrant as a Tool

It's a framework. But it does ship with multiple plugins by default

- Providers: Docker, HyperV, Virtualbox. Virtualbox is the default
- Provisioners: Docker, Puppet (server and solo), Chef (server and solo), Ansible, CFEngine, shell.

Vagrant as a Tool

It's a framework. But it does ship with multiple plugins by default

- Providers: Docker, HyperV, Virtualbox. Virtualbox is the default
- Provisioners: Docker, Puppet (server and solo), Chef (server and solo), Ansible, CFEngine, shell.

And allows you to download VM images off the internet

Vagrant as a Tool

It's a framework. But it does ship with multiple plugins by default

- Providers: Docker, HyperV, Virtualbox. Virtualbox is the default
- Provisioners: Docker, Puppet (server and solo), Chef (server and solo), Ansible, CFEngine, shell.

And allows you to download VM images off the internet
And YES, Juniper does provide some to the world!

Vagrant => Reproducible Environments

Key Takeaway ?

Vagrant => Reproducible Environments

Key Takeaway ?
REPRODUCIBLE ENVIRONMENTS!

Vagrant for the first time

Run **vagrant init** in an empty folder. You get the following Vagrantfile (minus the comments)

```
1 Vagrant.configure(2) do |config|
2   config.vm.box = "base"
3 end
```

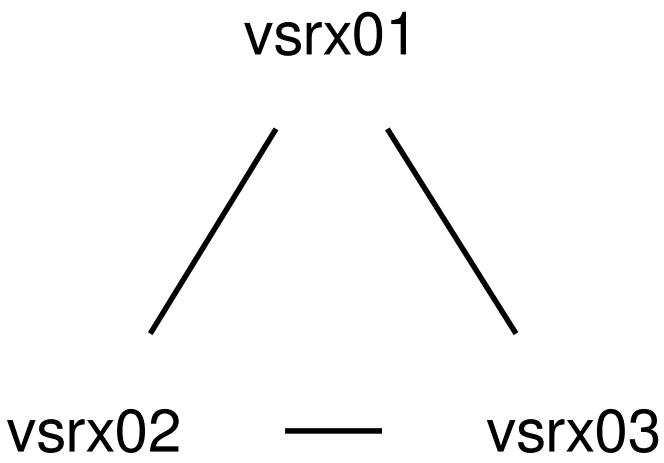
Now run **vagrant up** and you have your one base VM. Run **vagrant destroy** and kill it

Let's spice it up a bit

vagrant init juniper/ffp-12.1X47-D15.4-packetmode. Edit. It's Ruby

```
1  Vagrant.configure(2) do |config|
2    config.vm.box = "juniper/ffp-12.1X47-D15.4-packetmode"
3    config.vm.define "vsrx01" do |vsrx01|
4      vsrx01.vm.host_name = "vsrx01"
5      vsrx01.vm.network "private_network", ip: "192.168.12.11", virtualbox__intnet: "01-to-02"
6      vsrx01.vm.network "private_network", ip: "192.168.31.11", virtualbox__intnet: "03-to-01"
7    end
8    config.vm.define "vsrx02" do |vsrx02|
9      vsrx02.vm.host_name = "vsrx02"
10     vsrx02.vm.network "private_network", ip: "192.168.23.12", virtualbox__intnet: "02-to-03"
11     vsrx02.vm.network "private_network", ip: "192.168.12.12", virtualbox__intnet: "01-to-02"
12   end
13   config.vm.define "vsrx03" do |vsrx03|
14     vsrx03.vm.host_name = "vsrx03"
15     vsrx03.vm.network "private_network", ip: "192.168.31.13", virtualbox__intnet: "03-to-01"
16     vsrx03.vm.network "private_network", ip: "192.168.23.13", virtualbox__intnet: "02-to-03"
17   end
18 end
```

Let's spice it up a bit (2)



What is Ansible?

Wikipedia says:

Ansible, a free-software platform for configuring and managing computers, combines multi-node software deployment, ad hoc task execution, and configuration management. It manages nodes (which must have Python 2.4 or later installed on them) over SSH or over PowerShell. Modules work over JSON and standard output and can be written in any programming language. The system uses YAML to express reusable descriptions of systems.

What is Ansible?

Wikipedia says:

Ansible, a free-software platform for configuring and managing computers, combines multi-node software deployment, ad hoc task execution, and configuration management. It manages nodes (which must have Python 2.4 or later installed on them) over SSH or over PowerShell. Modules work over JSON and standard output and can be written in any programming language. The system uses YAML to express reusable descriptions of systems.

That sounds... unhelpful! WHAT ACTUALLY IS Vagrant?

WHAT IS ANSIBLE?

In the presenter's own words ?

WHAT IS ANSIBLE?

In the presenter's own words ?

- A way to write configuration

WHAT IS ANSIBLE?

In the presenter's own words ?

- A way to write configuration
- A way to push above said configuration to nodes

WHAT IS ANSIBLE?

In the presenter's own words ?

- A way to write configuration
- A way to push above said configuration to nodes
- A way to ensure configuration of nodes

WHAT IS ANSIBLE?

In the presenter's own words ?

- A way to write configuration
- A way to push above said configuration to nodes
- A way to ensure configuration of nodes

A Configuration Management System!

How does it work?

So how does ansible work? The basics:

How does it work?

So how does ansible work? The basics:

- Create an inventory file

How does it work?

So how does ansible work? The basics:

- Create an inventory file
- Define some tasks you want

How does it work?

So how does ansible work? The basics:

- Create an inventory file
- Define some tasks you want
- Group tasks and hosts into a playbook

How does it work?

So how does ansible work? The basics:

- Create an inventory file
- Define some tasks you want
- Group tasks and hosts into a playbook
- Apply the playbook => $ ansible-playbook -i hosts apache2.yaml

How does it work?

So how does ansible work? The basics:

- Create an inventory file
- Define some tasks you want
- Group tasks and hosts into a playbook
- Apply the playbook => $ ansible-playbook -i hosts apache2.yaml

Let's see them quickly one by one

Ansible inventory file

Here's a very very simple ansible inventory.

```
1 [webservers]
2 www1.example.com
3 www2.example.com
4
5 [dbservers]
6 db0.example.com
7 db1.example.com
```

Ansible sample tasks

Just install apache2 as a task

```
1 - name: Install apache2 package
2   sudo: true
3   apt: name=apache2 state=installed
```

Ansible playbooks

Group them together in a playbook

```
1 - hosts: webservers
2   tasks:
3   - name: Install apache2 package
4     sudo: true
5     apt: name=apache2 state=installed
```

How all this ties together?

How all this ties together?

- Vagrant provides an inventory file for us and calls ansible-playbook

How all this ties together?

- Vagrant provides an inventory file for us and calls ansible-playbook
- Junos supports NETCONF

How all this ties together?

- Vagrant provides an inventory file for us and calls ansible-playbook
- Junos supports NETCONF
- Juniper provides an ansible library with various tasks for junos

How all this ties together?

- Vagrant provides an inventory file for us and calls ansible-playbook
- Junos supports NETCONF
- Juniper provides an ansible library with various tasks for junos
- We get to write the playbooks

How all this ties together?

- Vagrant provides an inventory file for us and calls ansible-playbook
- Junos supports NETCONF
- Juniper provides an ansible library with various tasks for junos
- We get to write the playbooks

https://github.com/Juniper/ansible-junos-stdlib

Enable RIP 1/3 - Vagrantfile

```
1  Vagrant.configure(2) do |config|
2    config.vm.box = "juniper/ffp-12.1X47-D15.4-packetmode"
3    config.vm.define "vsrx01" do |vsrx01|
4      vsrx01.vm.host_name = "vsrx01"
5      vsrx01.vm.network "private_network", ip: "192.168.12.11", virtualbox__intnet: "01-to-02"
6      vsrx01.vm.network "private_network", ip: "192.168.31.11", virtualbox__intnet: "03-to-01"
7    end
8    config.vm.define "vsrx02" do |vsrx02|
9      vsrx02.vm.host_name = "vsrx02"
10     vsrx02.vm.network "private_network", ip: "192.168.23.12", virtualbox__intnet: "02-to-03"
11     vsrx02.vm.network "private_network", ip: "192.168.12.12", virtualbox__intnet: "01-to-02"
12   end
13   config.vm.define "vsrx03" do |vsrx03|
14     vsrx03.vm.host_name = "vsrx03"
15     vsrx03.vm.network "private_network", ip: "192.168.31.13", virtualbox__intnet: "03-to-01"
16     vsrx03.vm.network "private_network", ip: "192.168.23.13", virtualbox__intnet: "02-to-03"
17   end
18
19   config.vm.provision "ansible" do |ansible|
20       ansible.playbook = "rip.yaml"
21   end
22 end
```

Enable RIP 2/3 - rip.yaml - The playbook

```
1  ---
2  - hosts: all
3    gather_facts: no
4    connection: local
5    tasks:
6    - name: Checking NETCONF connectivity
7      wait_for: host={{ ansible_ssh_host }} port={{ ansible_ssh_port }} timeout=5
8    - name: enable RIP
9      junos_install_config:
10       host: "{{ ansible_ssh_host }}"
11       port: "{{ ansible_ssh_port }}"
12       user: root
13       ssh_private_key_file: "{{ ansible_ssh_private_key_file }}"
14       file: rip.conf
```

Enable RIP 3/3 - rip.conf - The configuration

```
protocols {
    rip {
        group rip-group {
            export advertise-routes-through-rip;
            neighbor ge-0/0/1;
            neighbor ge-0/0/2;
        }
    }
}
policy-options {
    policy-statement advertise-routes-through-rip {
        term 1 {
            from protocol [ direct rip ];
            then accept;
        }
    }
}
```

Demo

DEMO time!!

Use cases

Where to use that thing?

Use cases

Where to use that thing?

- Learning

Use cases

Where to use that thing?

- Learning
- Keeping firewall rules consistent across HA setups

Use cases

Where to use that thing?

- Learning
- Keeping firewall rules consistent across HA setups
- NFV

Use cases

Where to use that thing?

- Learning
- Keeping firewall rules consistent across HA setups
- NFV
- You name it

Questions

Questions ?