Implementing Configuration Management and IaC

Andrew Mallett
LINUX AUTHOR AND TRAINER

@theurbanpenguin   www.theurbanpenguin.com

Objectives

Understanding Infrastructure as Code

- IaC
- File Formats
  - JSON
  - YAML
  - Ruby
- Utilities
  - Ansible, Salt, Terraform, Vagrant
- Provisioning
- CI/CD
  - Continuous Integration
  - Continuous Deployment

```
        mirror_mod.use_x = False
        mirror_mod.use_y = True
        mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
        mirror_mod.use_x = False
        mirror_mod.use_y = False
        mirror_mod.use_z = True

        #selection at the end -add back the deselected mirror modifier
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
        #mirror_ob.select = 0
```

Infrastructure as Code (IaC)

- The managing and provisioning of your infrastructure using code in place of a manual process
- Configuration files define the desired state of your systems acting as both documentation and provisioner

CI/CD

- CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous development and deployment

File Formats

Depending on your IaC program you will use different file formats for the desired state configuration

- YAML: Salt and Ansible
- JSON: Often used as backend data
- Ruby: Vagrant, Puppet, Chef

Tools

We will look in detail at some the IaC tools as we go through the course. These tools include

- Ansible
- Chef
- Vagrant
- Puppet
- Terraform

```
$ vim ~/.vimrc
set modeline
$ mkdir -p ~/apache/web && cd ~/apache
$ vagrant init --minimal ubuntu/focal64
$ sed -i '1i # vi: ft=ruby:ts=2:ai:sw=2' Vagrantfile
```

Vagrantfile

Vagrant, as we have seen, can be used to deploy systems. This provides the OS, using provisioners, we can further configure the system. The file format of a Vagrantfile is Ruby. Adding a modeline can help vi/vim understand the syntax and format

Demo

Working on the Host System:

- Enable vim modeline
- Create new Vagrantfile
- Add Ruby configuration

~/apache/Vagrantfile

```ruby
# vi: ft=ruby:ts=2:ai:sw=2

$script = <<-SCRIPT
echo "Provisioning OS"
sudo apt update
sudo apt install -y apache2
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.provision "shell", inline: $script
  config.vm.synced_folder "web/", "/var/www/html"
  config.vm.network "forwarded_port", guest: 80, host: 8080
end
```

```
$ vagrant plugin list
$ vagrant plugin install vagrant-vbguest
$ echo "Hello" > ~/apache/web/index.html
$ cd ~/apache
$ vagrant up
$ vagrant port
$ curl http://localhost:8080
```

Vagrant Plugins

Plugins add functionality, here we allow vagrant to install Virtual Box Guest Additions if needed in the virtual machine. We need this for synced folders to work properly

Demo

Improving the Vagrantfile

- Install web server
- Map directories
- Map ports
- Test web server deployment

```
YAML
--- # doc Header
file: my.file # simple key value pair
names: # List
     - fred
     - bill
package: # dictionary (list of key value pairs)
     name: tree
     state: installed
... # doc footer
```

YAML

False: Yelling At My Laptop

False: Yet Another Markup Language

True: YAML Ain't Markup Language
It is a data structure language like JSON

Demo

Writing YAML:

- Create YAML file
- Configure VIM
- YAML lint

Demo

Compare YAML and JSON:

- JavaScript Object Notation
- yaml-online-parser.appspot.com

Summary

Understanding IaC

- Infrastructure as Code
- CI/CD
- vagrant
- Ruby
- vim modeline
- ansible
- terraform
- desired state
- YAML
- JSON

Implementing Ansible for Configuration Management