Virtual Machine Deployment and Configuration

System Benchmarking and Deployment

2018/2019

The main goal of this guide is to understand how to deploy and configure virtual machines (VMs) in an automatic and repeatable fashion.

For the exercises described next, the following tools must be installed,

- VirtualBox - `https://www.virtualbox.org`
- Vagrant - `https://www.vagrantup.com`

while useful Vagrant documentation is available at:

- Command line utilities - `https://www.vagrantup.com/docs/cli/`
- Configuration file - `https://www.vagrantup.com/docs/vagrantfile/`
- Multiple VMs setup - `https://www.vagrantup.com/docs/multi-machine/`
- Network configuration - `https://www.vagrantup.com/docs/networking/`
- Shell Provisioner - `https://www.vagrantup.com/docs/provisioning/shell.html`.

Steps

Deploying a single VM automatically

1. Use the Vagrant box command utility for setting up a **ubuntu/xenial64** box (create a local Ubuntu VM image usable by Vagrant). Check that it was created successfully:

   ```
   vagrant box add ubuntu/xenial64
   vagrant box list
   ```

2. Setup a Vagrant configuration file that will deploy one VM with the following configuration.

   1 Virtual CPU, 512 MB of RAM and **ubuntu/xenial64** image (box).

   Note: use the command *vagrant init ubuntu/xenial64* to create an initial configuration file.

3. Deploy the VM with the command:

```
vagrant up
```

4. Check if the VM was created within the VirtualBox application and connect to the VM (*vagrant ssh*).

5. Stop and clean the VM deployment with the command:

```
vagrant destroy
```

6. During the provision phase, the VM should automatically run the following commands:

```
apt-get -y update
apt-get -y upgrade
apt-get -y autoremove
apt-get install -y vim
```

7. Halt the VM and restart. Check differences from using destroy and halt instructions.

```
vagrant halt
vagrant up
```

shut down

update das configurações

8. In terms of Network interfaces the VM must be configured with (remember to halt the VM before changing this configuration):

   a private network with a fixed ip (**10.0.0.101**).

   a public network for direct communication between the VM, the host machine and the internet (created by default)

9. Start the VM, use SSH to connect to the VM, check the network interfaces (*ifconfig* command).

10. Allow SSH access to the VM with a specific public key at the host machine.

    The following Ruby line can be used to read the content from a public SSH key at the host (replace the key path with the correct one for your host):

```
PUBLIC_KEY = File.read(File.expand_path('~/.ssh/id_rsa.pub')).strip
```

    The following command will copy the host's public SSH key content to the authorised keys at the VM:

```
echo "#{PUBLIC_KEY}" >> /home/ubuntu/.ssh/authorized_keys
```

    Re-run the provision step without shutting down the VM and test SSH (via the **10.0.0.101** ip address):

```
vagrant provision --provision-with shell
```

11. Use the same vagrant file to create an additional VM with a similar configuration and a novel private IP address (**10.0.0.102**).

12. Deploy the VMs, use SSH to connect, ensure that both VMs can communicate via the 10.0.0.X network (*e.g.*, with the *ping* command).

Learning outcomes Experiment linux virtual machines automatic deployment with VirtualBox and Vagrant. Assess how Vagrant helps simplifying the configuration and deployment of virtual machines. Revise Vagrant configuration parameters, scopes, and deployment/management commands.