Cloud Computing Applications and Services
Warm-up

2023

Vagrant

The goal of this guide is to explain 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 (version 7.0) - `https://www.virtualbox.org`
- Vagrant (version 2.3) - `https://www.vagrantup.com`

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`.

Tasks

Vagrantfile. Vagrant uses a configuration file to specify the provisioning of VMs. Inspect the Vagrantfile example, provided with this guide, in order to understand the following aspects.

1. The example uses variables for keeping the number of VMs to be provisioned, the IP range for their private network, the local path for user's SSH public key and the content of the public key.

   ```
   Number_VMs = 2
   IP_RANGE= "192.168.56"
   PUBLIC_KEY_PATH = "~/.ssh/id_rsa.pub"
   READ_PUBLIC_KEY = File.read(File.expand_path(PUBLIC_KEY_PATH)).strip
   ```

   Note 1: All these variables need to be adapted for your deployment.
   Note 2: Further information regarding the networking and SSH topics can be found in the slides that accompany this guide.
   Note 3: Vagrant allows calling ruby code. The `READ_PUBLIC_KEY` variable stores the content of the public key found at `PUBLIC_KEY_PATH`.

2. The Linux distribution (*i.e.*, VM image), the virtualization provider (*i.e.*, VirtualBox) and the general VM configurations (*i.e.*, RAM, number of CPUs) are defined at:

```
config.vm.box = "bento/ubuntu-20.04"
config.vm.provider "virtualbox" do |vb|
  vb.memory = "1024"
  vb.cpus = 2
end
```

3. Configurations can also be specified independently for each VM. The example below uses a loop to change the hostname and IP address in a per VM basis.

```
(1..Number_VMs).each do |i|
    config.vm.define "server#{i}" do |node|
      node.vm.hostname = "VM#{i}"
      node.vm.network :private_network, ip: "#{IP_RANGE}.#{100+i}"
    end
  end
```

Note 1: A private network is only accessible for the host and for other VMs at the same network. A NAT public network is used for communication between VMs and the internet. Check the slides for further information.

4. The *config.vm.provision* parameter specifies provision tasks that should execute after starting the desired VMs. For instance, one can use the *shell* environment to execute shell scripts at the VM.

```
config.vm.provision "shell", inline: <<-SHELL
  echo "#{READ_PUBLIC_KEY}" >> /home/vagrant/.ssh/authorized_keys
  (...)
  apt update
  apt-get install -y vim
SHELL
```

Note 1: The first command copies the content of the public SSH key (stored at the `READ_PUBLIC_KEY` variable) to the `authorized_keys` file at the VM's file system.
Note 2: All the previous commands execute inside the VM!

Testing

1. Deploy two VMs by running (at the same folder where the Vagrantfile is stored):

```
vagrant up
```

2. Check if the VMs were created within the VirtualBox console.
3. Connect to each VM by using SSH. Check (with the `ip add` command) that the VMs have a public (NAT network created automatically by Vagrant) and private network interface configured.
4. Check if the VMs can communicate through the private network (*e.g.*, `ping` command).
5. Re-run the provision step, without shutting down the VMs, with:

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

6. Power off the VMs and restart only VM1:

```
vagrant halt
vagrant up server1
```

7. Stop and clean the VMs deployment with:

```
vagrant destroy
```

Explore Explore the following commands and check their usage and/or output:

- `man` - manual pages for commands, library functions, system calls, etc.
- `vim / nano` - text editors;
- `cat / more / less` - file content visualization;
- `mkdir / rmdir` - directory manipulation;
- `ls / pwd / cd` - file system navigation;
- `mv / cp / rm / scp` - file manipulation;

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