Computing Infrastructures Provisioning

Linux Virtual Machines with Vagrant

Lab setup with

Provisioning Linux Virtual Machines with Vagrant

Versão 0.2.0

Initial draft:

- Luís Osório

Contributions:

- Luís Osório
- …

Table of Contents

1 Introduction

The execution environment for an Informatics System (ISystem) abstraction considers that the Service elements might be running directly on a bare-metal computer system, on virtual machine operating systems, e.g., Linux, macOS, Windows, on Linux Containers, or else on a Kubernetes cluster. A service element can even run on an operating system of a cyberphysical system while logically and computationally being part of an Informatics System.

This document outlines key issues to help manage such a distributed execution environment that we formally refer to as distributed systems infrastructures. The reason is that such a network of computers or virtual execution environments (machine or container) connect through a TCP/IP physical wired or wireless network. We consider such networked computational artifacts as part of Informatics Systems for simplicity. If organized under an Informatics System context, some of such computational artifacts may play some role in infrastructural computing. We are concerned with computing artifacts that directly or indirectly contribute to reliable computer and communication-related technology landscapes. Therefore, we structure such networked computing artifacts as service elements, making them part or grouped into CES elements and then part of an informatics system (ISystem). The ISystem model concept also applies to software tools (design, development, and configuration management) when considering that its technology artifacts are managed under an integrated strategy. It is paramount for the security and reliability of informatics systems to manage all the artifacts contributing to critical technology landscapes' development and operations (DevOps). As discussed and included in the document, a change to some infrastructure aspects of any technology artifact might use an open standard domain-specific language to maintain the necessary meta-data. The Vagrant/Ruby script language and the associated metadata for managing virtual machines we discuss in the following chapter exemplify such a domain-specific language.

The remaining document follows the structure:

- Chapter 2, Managing Virtual Machines with Vagrant, introduces a valuable virtual environment provisioning and configuration tool dealing with installations of scale;
- Chapter 3 presents conclusions and outlines open research challenges.

2 Managing Virtual Machines with Vagrant

The Vagrant tool [1] is a command-line wrapper and declarative language interpreter to help manage virtual machines' generation (or life cycle management). Hasicoprp developed Vagrant for infrastructure as a code as a domain-specific language (DSL) under the open-source model (link). They aimed to answer the need to automate increasing hard-to-configure operating-system-level execution environments. By execution environment, we mean, among other resources, the classic operating system abstraction, which has recently gotten consensus around the Linux kernel software system level.

Creating and configuring a new virtual installation with one of the Linux distros (name attributed to the number of Linux distributions) is a complex and error-prone job. After creating a new virtual machine from an ISO image or a Vagrant box, it follows the execution of time-consuming and error-prone shell script commands to install and configure the required resources. To simplify the job, HashiCorp develops Vagrant as a provisioning and configuration software tool to streamline the creation of execution environments.

The Ansible language and Automation Platform (link, link), proposed and maintained by RedHat, is frequently used to configure Vagrant-generated artifacts (virtual machines). The Ansible language is a domain-specific language, according to (link) *"… A Simple Model-Driven Configuration*

Management And Command Execution Framework", meaning it intersects Vagrant with a complementary scope.

Hashicorp further maintains the Terraform initiative as a more ambitious suit of tools addressing both on-premises/cloud and public/cloud environments [ref], defined as "*Infrastructure automation to provision and manage resources in any cloud or data center*." [ref].

The chapter structure is as follows: section 2.1 sketches how to install and summarize basic Vagrant commands and configuration issues and presents simple examples to validate in the laboratory. Section 2.2 outlines and explains the scripts for installing the Apache Zookeeper.

2.1 Vagrant Installation

Vagrant installs from a simple executable (link) that interprets and executes a building workflow expressed by a vagrant file whose default name is ***Vagrantfile***. A simple use case is the provisioning of virtual machines on a virtualization runtime, e.g., the Virtualbox from Oracle, by running a Vagrantfile script.

Obtain the installation executable of Vagrant for your laptop from (link), of this writing, version 2.4.1. After executing the install file vagrant_2.2.19_x86_64.msi for your OS, you can verify if the installation worked properly by running the vagrant command line in a Windows terminal:

```
> vagrant --version
Vagrant 2.4.1
```

A straightforward utilization is to call Vagrant with the init command and a reference for a base image as an argument to announce the image to configure the Vagrantfile used to create a virtual machine on the Virtualbox (Usage: Vagrant [options] <command> [<args>]). If you do that in the HelloVagrant directory, the respective Vagrantfile is generated (iesd2324sv git repository, Kubernetes folder).

```
> vagrant init ubuntu/focal64
```

Followed by:

```
> vagrant up
```

The HelloVagrant_default_1649032757288_29466 virtual machine is created in the VirtualBox you can access through a Secure Shell Protocol (SSH) connection:

```
> vagrant ssh
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-172-generic x86_64)
…
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
vagrant@ubuntu-focal:~$
```

Or you can try to provision a RedHat Enterprise Linux version 9 virtual machine using the Vagrant Box [link] to look for the generic/rhel9 at [link].

```
>vagrant init generic/rhel9
>vagrant up
```

You can login with SSH.

```
> vagrant ssh
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-
dashboard
Last login: Sat Feb 24 11:12:30 2024 from 10.0.2.2
```

```
[vagrant@rhel9 ~]$
```

By default, the hostname is the name of the base box, in this first example, ***ubuntu-focal***, and in the second, ***rhel9***. In both examples, the hostname and default vagrant user name can change, as explained in the following examples.

While the target virtualization platform is Virtualbox by default, others exist, e.g., Vmware, libvirt, Hyperv, openstack, xenserver, etc. A Box is a Vagrant concept representing a reusable working image with the Linux operating system, making it possible to replicate (automate) its creation.

The simplest example to generate a minimal Vagrantfile in a directory named HelloVagrant:

```
> Vagrant init --minimal
A `Vagrantfile` has been placed in this directory. You are now ready to
`vagrant up` your first virtual environment! Please read the comments in the
Vagrantfile as well as documentation on `vagrantup.com` for more information
on using Vagrant.
```

The resulting Vagrantfile includes, beyond comment lines starting with the '#' character, the following Ruby script lines:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  config.vm.box = "base"
end
```

The machine or machines provisioned using a Vagrantfile can update through the reload vagrant command. The Vagrant tool uses the concept of Provisioner [link] to realize configurations that are usually necessary to tailor the virtual machine for a specific purpose. While an SSH connection also makes it possible to realize configurations, they are automated using the most basic provider, the Vagrant Shell provisioner. The Shell provider allows the upload and execution of a script within the guest machine.

```
vagrant reload --provision
```

An exciting feature is the application of similar configurations to a set of machines see (link) as reproduced here:

```
(1..3).each do |i|
  config.vm.define "node-#{i}" do |node|
    node.vm.provision "shell",
      inline: "echo hello from node #{i}"
  end
end
```

The example of the vagrant project to create a Zookeeper ensemble, zQuorumVagrant, applies the constructor to replicate the same configuration for five virtual machines.

Networking and name server resolution (DNS)

Networking is a critical aspect, namely related to version changes. We adopt static IP addresses to simplify the approach to the examples. Further information on how Vagrant configures VirtualBox network issues can be accessed here (link).

Summary of More Used Vagrant Commands

A virtual machine created and provisioned with a Vagrant command should always be managed with Vagrant. If managed directly, there is the risk of the .vagrant metadata becoming inconsistent.

```
> vagrant [options] <command> [<args>])
```

The command ***init*** creates a vagrant deployment from a Box at [link]. The result is a Vagrantfile and the directory .vagrant with metadata required by the Vagrant tools.

```
> vagrant init <a box>
```

The command up executes the Vagrant file and creates the virtual machine or machines following the Vagrantfile instructions.

```
> vagrant up
```

The command ***halt*** to gracefully stop a virtual machine. The machine is started again with the up command.

```
> vagrant halt
```

The command ***destroy*** with --force argument eliminates the virtual machines without questioning.

```
> Vagrant destroy --force
```

The reload command updates the virtual machine or machines, and the optional argument --provision forces the execution of provisioning providers, e.g., via Shell scripts. One example is the Shell provisioner demonstrated by the HelloVagrant-Webapp example.

```
> Vagrant reload  --provision
```

The following configuration lines demonstrate the adoption of Shell provisioning in the HelloVagrant-Webapp example in the next section.

```
…
# Install and start WebApp
    webapp.vm.provision "shell", path: "installresources.sh"
…
```

The halt command stops the virtual machines.

```
> vagrant halt -force
```

In the HelloVagrant-Webapp example, a Shell script installs a deno web server runtime [link]. Opening an SSH session to the created machine makes it possible to inspect the provisioning results.

```
vagrant ssh
```

Like other frameworks or tools, Vagrant also offers the possibility to install additional packages (modules), known as plugins or Vagrant plugins, to add new functionalities. For example, the Vagrant Share plugin enables the installation and publishing of services by invoking the following Vagrant command:

```
> vagrant plugin install vagrant-share
```

The Vagrant Share relies on Ngrok services [link]. Ngrok provides mechanisms to make intra-NAT services accessible using a secure channel from the Internet. The service is free for a single Ngrok process, four tunnels, and forty connections per minute (as of February 2024).

2.2 A Simple guest machine publishing a web application on the host example

The HelloVagrant-webapp Vagrant project in iesd2324/ISosIFR considers a Ubuntu guest installation using Deno [link] web backend runtime to execute a simple web application on port 8000 accessible on the host machine at port 8080 (http://localhost:8080/). The Vagrantfile includes a forwarding statement to configure the virtual machine through the VirtualBox programmatic interface to forward the web service from the iserver-webapp:8000 to the host:8080.

```
# This Vagrant script creates a Web App virtual machine on Virtualbox.
# Vagrantfile API/syntax version
VAGRANTFILE_API_VERSION = "2"
WEB_APP_SERVER_IP = "192.168.59.100"


Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  # Web Application server, nodewa
  config.vm.define 'hello-webapp' do |webapp|
    webapp.vm.box = 'ubuntu/focal64'
    webapp.vm.hostname = 'iserverwa'
    webapp.vm.network "private_network", ip: WEB_APP_SERVER_IP, hostname: true
    webapp.vm.network "forwarded_port", guest: 8000, host: 8080
    # Configure VM
    webapp.vm.provider 'virtualbox' do |vb, override|
      vb.name = "iserver-webapp"
      vb.memory = '1024'
      vb.cpus = 1
    end
    # Install and start WebApp
    webapp.vm.provision "shell", path: "installresources.sh"
  end
end
```

The installresources.sh shell script installs the Deno runtime and starts the web application, showing a browser's "IESD 2324 - Hello, World!" message.

2.3 Provisioning of virtual machines for Apache Zookeeper

The Apache Zookeeper is a reliable tree-like storage where z-nodes associate a name to a data block. As we will study, the Apache Zookeeper has two installation modes. The simplest one is a standalone or unreliable installation since it considers instantiation on a single host. The quorum mode is based on a cluster configuration known as an Ensemble, where a majority of nodes are sufficient to maintain the services offered.

The first example, iesd2324/ISosIFR/ZkUnreliableVagrant project, creates a single virtual machine on VirtualBox based on the Linux operating system Ubuntu 22.04, box 'ubuntu/focal64' and installs the last version Apache Zookeeper.

With the VirtualBox installed, open a command line console on ZkUnreliableVagrant and execute the '*vagrant up*' command to create the newly configured virtual machine.

```
> vagrant up
```

To test the new installation, install the Apache Zookeeper in your host machine and open the Zookeeper command line interface.

```
C:\Java\apache-zookeeper-3.9.2-bin\bin\zkCli -server 192.168.59.120
```

2.4 Provisioning of a virtual machine for Apache Karaf

To facilitate the instantiation of Apache Karaf, the iesd2324/ISosIFR/KarafHelloVagrant project makes it possible to create a virtual machine on VirtualBox based on an Ubuntu box.

The associated '*readme. adoc'* file gives additional cues to install the Karaf web console, a step that is left for students for automation. Apache Karaf needs to be configured to install predefined features, such as the http and webconsole features; see Karaf documentation [ref].

Be aware of a change of the Karaf host since there is a need to remove the previous host from the .ssh known hosts. When the virtual machine is rebuilt, if you try to execute the SSH access.

```
ssh -p 8101 karaf@192.168.59.130
```

The following message is returned.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:of9v1iUt3FQ3Q5cpwUkpV71ndPt32540kNdCfuW+NiM.
Please contact your system administrator.
```

To solve the problem, remove the respective (host with IP 192.168.59.130) entry from the %userprofile%/.ssh/known_hosts file.

```
[192.168.59.130]:8101 ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCiR4DlEmnR8gC8J+qhfsLS3Uqc0/0D0+YePeUH3lvl7kcKjk
LoJdZg9H1CUoltn2Qo1UNCuWwPDSIWCbfrQvlCHsQw0brtzHYtKxbJdbcFy9NYaZkUyIVYPA22FR5E
Y3ICCilurcRL0dHq/VXuu2ibw/higyFs/9SgO5xSQcTg5qgyXRQBda2pc6rv1mUBAYiCj/NJCg94FO
xKjz28tb10rr3heaiTjR4dgRIqN6esZ4ODcFDnK8sHJhO/taBrHg/yrLAIJqbz/tuP2RLD9msmXzql
ODKHfPeuKY47UyJIIPHYIsb7BqwDjn5YvTt5NYuzz7lt9/rtTDuDCR/bKTpO1
```

Changes on Virtualbox virtual machines created with Vagrant shall be managed from the project directory to maintain the metadata's consistency of '*.vagrant*' directory.

3 Conclusions

The Vagrant tool is discussed as a provisioning tool for creating and configuring virtual machines. Compared to traditional methods to create and manually configure machine by machine, Vagrant offers a mechanism based on a Vagrantfile Ruby script to create virtual machines using a declarative approach. The trend is generally known by infrastructure as a code (IaaC) to provisioning and configuration management.

References

[1] Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, and Dan Mackin. *UNIX and Linux System Administration Handbook (5th edition)*. Addison-Wesley Professional, 5th edition, 2018.