Vagrant Box User Guide

Eugenio Gianniti

October 9, 2016

1 Introduction

This guide gives a quick how-to for the installation and use of the *Vagrant box* provided for the Algorithms and Parallel Computing (APC) course. Since the installation is a lengthy procedure, read and follow this guide *before* any practical session. Please, make sure that you can follow closely the example provided in Section 4, thus verifying that the installation succeeded.

2 Installation

First of all, install Virtualbox[1] and Vagrant[2]. On a Windows host you may need to restart your computer afterwards.

As soon as the installation is over, *create a new directory* to store your working files. Open a Terminal (Mac OS X and GNU/Linux) or a command prompt (Windows), move to the newly created directory and type:

```
$ vagrant init eg123/apc
$ vagrant up
```

Listing 1: `vagrant init` example output.

```
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.
```

Listing 2: `vagrant up` example output.

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'eg123/apc'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'eg123/apc' is up to date...
==> default: Setting the name of the VM:
```

[1]https://www.virtualbox.org

[2]https://www.vagrantup.com

```
acp_default_1471946500170_58184
==> default: Clearing any previously set network interfaces ...
==> default: Preparing network interfaces based on configuration ...
default : Adapter 1: nat
==> default: Forwarding ports...
default : 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few
minutes...
default : SSH address: 127.0.0.1:2222
default : SSH username: vagrant
default : SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
default : /vagrant => /Users/eugenio/Desktop/acp
```

On Windows, `vagrant up` might fail to download the Vagrant box. If this is the case, install also Microsoft Visual C++ 2010 Redistributable x86[3].

Now you can gracefully shutdown your virtual machine (VM) with:

```
$ vagrant halt
```

Beware that the entire procedure might take up to an hour, depending on your connection bandwidth. You have to install your working environment *before* any practical sessions.

3 Usage

Every time you need your working environment, head to the directory created in Section 2 and type:

```
$ vagrant up
```

This time it will not download the box from the Internet, but only start the local copy that was prepared in the first run. As soon as the VM is ready, you can access it via:

```
$ vagrant ssh
```

Again, Windows might error out saying your system lacks `ssh`. Crazy as it might seem, the solution involving the least hassle is to install Git for Windows[4]. The installer will provide, among other stuff, Git Bash, from where you can access a fully functional `ssh` client.

After connecting to your Vagrant box, you will find yourself in a GNU/Linux VM ready for C++ development. If you are not familiar with UNIX shells,

[3]https://www.microsoft.com/en-us/download/details.aspx?id=26999
[4]https://git-for-windows.github.io

consider reading a tutorial[5] to learn some basic skills, such as moving around the file system, handling files, and working with the environment.

From version 0.0.2 on, the provided VM sports the same distribution and the same module system as the servers in the Department of Maths. Due to this reason, when you want to use the development toolchain, you should use the following commands:

```
$ /u/sw/bin/bash
$ module load gcc-glibc/5
```

These will start the shell associated with the modules, thus providing an insulated base system, and load the toolchain. In this case, it is the GNU Compiler Collection version 5 and the relative GNU C library.

Now, if you enter the commands:

```
$ cd /vagrant
$ ls
```

you will see the content of the working directory where you initialized Vagrant. Vagrant takes care of synchronizing this folder with `/vagrant` in the box, hence you can easily develop your source code with your favorite *text editor*[6], with all the ease and responsiveness of a graphical user interface (GUI) running on bare metal, switching back to the Vagrant box command line only to compile and run your programs.

When you are over with your work on the VM, you can shut it down via:

```
$ vagrant halt
```

In the unlucky event that your Vagrant box refused to work properly, you could completely erase it running:

```
$ vagrant destroy
```

The synchronized working directory will not be affected, but other than that you will obtain a clean VM, exactly identical to the first installation, at your next `vagrant up`.

4 Hello, World!

This section shows a complete example on how to build your first program, the evergreen *Hello, World!*, using your Vagrant box as development environment.

First of all, head to the Vagrant working directory, i.e., the folder where you set up your box via `vagrant init`, and save a file named `hello.cc` with the content shown in Listing 3. It is the source code for a basic program that will only print "Hello, World!" on the standard output. Still, it is a quick way to ensure that everything works as expected.

[5]For instance, `http://linuxcommand.org/index.php`

[6]Which is *not* a word processor like Microsoft Word or LibreOffice Write. If you have no favorites, either try out Atom (`https://atom.io`) or look up the list of text editors available in Wikipedia.

Listing 3: hello.cc

```
#include <iostream>

int main (void)
{
  std::cout << "Hello, World!" << std::endl;
  return 0;
}
```

As soon as the hello.cc file is ready, you can start up the VM and connect to it. In order to do so, fire up a terminal emulator, move to the working directory, and type in:

```
  $ vagrant up
  $ vagrant ssh
```

After `vagrant up`, Vagrant will output on screen some information about the box boot process, whilst after `vagrant ssh` you will see the following output:

```
Last login: Sun Oct  9 17:03:23 2016 from 10.0.2.2
[vagrant@localhost ~]$
```

This means that now you are no more interacting with the host operating system (OS) running on your computer, but with the guest OS that lives inside the Vagrant box. In particular, the guest OS is a GNU/Linux distribution, CentOS 7[7].

In order to access the C++ compiler, type:

```
  $ /u/sw/bin/bash
  $ module load gcc-glibc/5
```

Now, using the `cd` builtin you can change to the `/vagrant` directory, where the `ls` utility should list your Vagrant configuration, `Vagrantfile`, and the freshly baked `hello.cc` source file. This will amount to the following:

```
  $ cd /vagrant
  $ ls
Vagrantfile   hello.cc
```

The output from `ls` confirms that the content of your Vagrant working directory gets automatically synchronized between host and guest. The next step is compiling the source code: afterwards you might want to look up the working directory in your host OS GUI to check that the synchronization works in both directions and the executable appears outside the Vagrant box, too.

In order to compile, type in the following:

```
  $ g++ hello.cc -o hello
```

[7]https://www.centos.org

where `g++` is the C++ compiler, while the arguments instruct it to build the source code from `hello.cc` and store the obtained executable as `hello` in the current directory. If the compilation succeeds, you will not see any output. Notwithstanding, running `ls` again will reveal a new file in the `/vagrant` folder:

```
  $ ls
Vagrantfile   hello   hello.cc
```

In the end, we are left with the final check: the example program should run and print on screen the expected output.

```
  $ ./hello
Hello, World!
```

Congratulations! If you could follow up to this point, then you have a fully functional development environment ready for the APC course. On the other hand, if anything went awry give it another try from scratch: Vagrant makes it very fast and easy to just trash your broken VM and start afresh.