Setting up *Devops from Scratch* Virtual Machine

Mark Bools

October 13, 2020

Last Modified: 2022-10-05

Abstract

A brief guide to setting up a virtual machine to follow along with the *Devops from Scratch* book.

We use a virtual machine (VM) in order to establish a consistent environment for all those following the material. This eliminates most of the issues with distance learning where students use their own machines. Using individual machines becomes complex because each user will have different hardware, different software, and different configurations all of which can 'cross-talk' producing different results when following along with course material. Using a VM isolates the learning environment from most of this variability.

As you will learn throughout the *Devops from Scratch* course, we can provide code to precisely setup an environment. Combining a VM with this code setup means we can provide consistent checkpoints, allowing students to skip material in the course or 'reset' their environments if they become lost in the material. This ability to always restore to a known state makes it much easier to keep all students 'on the same page', an especially important facility when dealing with complex material such as that covered in the *Devops from Scratch* course.

If you intend to follow any SaltyVagrant course then it is advisable to spend a few minutes familiarising yourself with the following basics.

Initial installs

You will need to install three tools to your computer:

1. `virtualbox`—to run virtual machines.
2. `vagrant`—to supervise the creation and setup of virtual machines.
3. `git`—for version control (and more immediately to get the files needed to setup the virtual machines).

The installation instructions for these tools tend to vary over time so it's best if you visit the websites, download the tool and follow the installation instructions relevant to your computer's operating system.

Resources

The *Devops from Scratch* material places various demands on your hardware. Earlier sections require fewer resources, some later parts require more resources but often we rely on cloud services, thereby relieving your local hardware. I generally recommend the following minimum specification (you can manage with less and more is always better):

- Memory: 8GB
- CPU: 4 cores
- Storage: 250GB

Creating a basic *Devops from Scratch* VM

Once you have the three tools installed you are ready to create your first *Devops from Scratch* VM.

Obtain the VM sources. These are typically highlighted as part of the course material. We will create a basic developer's VM which forms the base for many of the other VMs you will be using and illustrates well the process of managing a *Devops from Scratch* VM.

```bash
git clone https://gitlab.com/SaltyVagrant/developer_vm.git
```

Now create the VM, this uses Vagrant to both create the basic VM and run appropriate scripts to add tools and configuration required for the course.

```bash
cd developer_vm
vagrant up
```

The length of time required to create the VM will depend upon several factors.

- Your hardware—better hardware generally means shorter VM build times.
- Your Internet connection—when initially bringing up a new Vagrant VM a 'box' may be downloaded, these can be quite large so your Internet speed will be a factor. Once this download is done any additional VMs based on the same 'box' will not require a download.
- The base 'box'—Vagrant calls the predefined VMs form which your new VM is created a 'box'. The more this box resembles the final VM the less work the provisioning scripts need to do.

- The provisioning scripts—Once the basic VM is created Vagrant will likely run a series of scripts to add customised setup for the VM, these are called 'provisioning scripts' in `vagrant` terminology. Obviously the more work these scripts have to do the longer the initial setup will take.

After `vagrant up` has completed you will have a running VM suitably configured for the start of the *Devops from Scratch* course.

Accessing the new VM

Once your VM is running you can log into it using `ssh`. To make this process as simple as possible `vagrant` provides its own `ssh` command.

```bash
vagrant ssh
```

Issuing this `vagrant up` command from within the `developer\_vm` directory (or any of it's subdirectories) will start an `ssh` session with the project's VM.

```bash
Linux debian-10 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2 (2020-04-29) x86_64

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 13 09:18:06 2020 from 10.0.2.2
vagrant@debian-10:~$
```

To leave the VM you simply logout.

```bash
exit
```

Controlling the VM

When not using the VM you may want to shut it down to free resouces on your computer.

To temporarily stop the VM but preserve its state (that is, save any work you have done on that VM).

```bash
vagrant halt
```

To resume the VM.

```bash
vagrant up
```

Note that this will restart the VM without running any provisioning scripts.

To 'reboot' the VM.

```bash
vagrant reload
```

Sometimes it is useful to save the VM in its current state so that you can try out some changes to the VM and then restore the VM to the saved condition. The is especially useful when you are playing with ideas suggested in the course but want to be able to clear out any changes before continuing with the course iteself.

To save the current state of the VM we take a 'snapshot'.

```bash
vagrant snapshot save mysnapshot
```

Replace `mysnapshot` with any name you prefer.

To restore the VM to a save snapshot.

```bash
vagrant snapshot restore mysnapshot
```

Obviously (I hope), you use the name you used to save the snapshot rather than `mysnapshot`.

These snapshots can be taken or restored whether the VM is running or halted. If you restore a VM that is halted it will be started as part of the restore process.

If you forget the name you used to save the VM snapshot use the `list` command.

```bash
vagrant snapshot list
```

Once you are finished with a VM you can get rid of it completely.

```bash
vagrant destroy -f
```

The `-f` flag with 'force' the destruction without prompting. If you omit the `-f` option you will be prompted to confirm the distruction of the VM. (This may seem redundant when dealing with a single VM, but many parts of the *Devops from Scratch* course require multiple VMs and you may want to be selective about which your destroy—specifying the `-f` option will destroy all the VMs without prompting.)

Once you have destroyed the VM you can then cleanup the project directory.

```bash
cd ..
rm -rf developer_vm
```