**Vagrant for VM Automation**

**Ganesh Palnitkar**

Get you copy of Vagrant and VirtualBox

If you don’t already have Vagrant and VirtualBox, grab those.

[Download Vagrant](http://www.vagrantup.com/downloads.html) installer for your operating system.

[Download VirtualBox](https://www.virtualbox.org/wiki/Downloads) installer for your operating system.

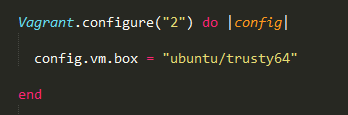
Building a Vagrant Box from Start to Finish

Vagrant makes it so simple to create a VM or entire environment that one would never think of using any other tool for automating the server / desktop virtualization.

Using few commands one can quickly build and run a VMs. Below are the very first commands that we use to get started.

$ vagrant init ubuntu/trusty64 … this command will create a new Vagrantfile and place the name ‘ubuntu/trusty64’ into the box statement.

The file would look like as below.



$ vagrant up … this will read thru the Vagrantfile and create a VM using the ‘ubuntu/trusty64’ box. Vagrant will look for the box availability in the folder where from we running the command. If the box file is not available, vagrant will download it from the Hashicorp’s Atlas repository.

Once the VM is up and running, we can login to the VM using the command,

$ vagrant ssh.

This will drop you into the VM.

We can also download the box and use it to create the VM. This can be done by first downloading the box file from any third party repository like, <http://vagrantbox.es> and then create vagrant box using the command,

$ vagrant box add <boxname> [file:///c:\<boxfilename.box](file:///c:\%3cboxfilename.box)>

This command will add a vagrant box to the vagrant database of available boxes. We can then run the command,

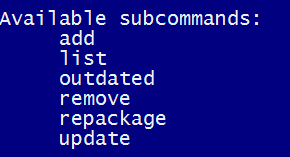
Depending on the Host operating system on which you will be running Vagrant and VirtualBox, some of the services running on the guest OS (VM) and configurations might vary. For example, if the guest Linux OS id Ubuntu, some of the services and configuration would work and in case of RedHat those would not start by default.

On the host machine from where you are running the vagrant,

You can run below commands to get information about running VMs, modify configuration for existing boxes and more.

$ vagrant box list … this will list all the configured VMs on the host machine.

There are few other sub commands for Vagrant Box command, such as.,



**Let’s look at the Vagrantfile**…

Vagrantfile is the main configuration file for automating VM creation. Vagrant uses pure Ruby syntax. A Vagrantfile can be written to create multiple VMs as well. Here is the sample code, for creating multiple VMs. Anserver, web and db are the variables that have been initialized and used to define VM variables and attributes, like hostname, network, box file name, and memory definition.

Every method started with the ‘do’ statement must end with the ‘end’ statement.



VM declaration

VM declaration

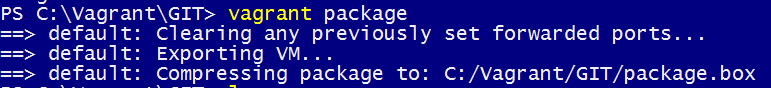
**Why Build A Box?**

There are many preconfigured / ready-to-use boxes available on sites like [vagrentbox.es](http://www.vagrantbox.es/) and [vagrantcloud.com](https://vagrantcloud.com/) so why would you want to build your own box?

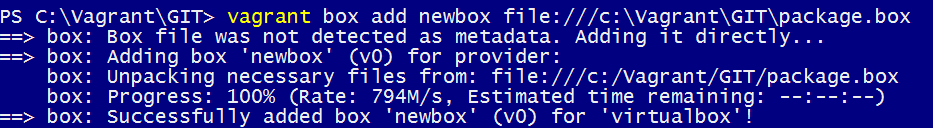
As a member of an operations team if I received a request for provisioning development environment, have to make sure all servers have uniform configuration, I would use a vagrant box and install required applications, services, framework and so on. IF this would be an ongoing request, instead of running the configuration exercise every time, I would prefer to create a box from the existing configured vagrant VM and use the box file for any such VM provisioning request.

Maybe you want your box to have more ram or you need your boxes to more closely mirror production and you are building a ram enriched, multiple server cluster with multiple provisioners.

This can be done using the command,



Now the package.box file can be used to add a vagrantbox to the vagrant database of available boxes.



The ‘newbox’ then can be used in the Vagrantfile to spin up a new VM.

What is a package.box file? When using the VirtualBox provider, it’s a tarred, gzip file containing the following:

Vagrantfile

box-disk.vmdk

box.ovf

metadata.json

The Vagrantfile has some information that will be merged into your Vagrantfile that is created when you run vagrant init boxname in a folder.

The box-disk.vmdk is the virtual hard disk drive.

The box.ovf defines the virtual hardware for the box.

The metadata.json tells vagrant what provider the box works with.

NOTE: These contents would be different for the VMWare provider, etc.

**Some of the frequently used Vagrant commands:**

$ vagrant –version …. Provides version information.

Vagrant 1.8.5

$ vagrant destroy … a VM can be deleted , including the files for the VM.

$ vagrant halt …. Stops a running VM.

$ vagrant reload --- restarts the running VM. Read and apply the Vagrantfile again. This way we can make changes to the Vagrantfile and reapply the config to the existing VM.

$ vagrant status … provides the status of the running VMs

Below is a listing of all available Vagrant commands and a brief description of what they do.

|  |  |
| --- | --- |
| Box | manages boxes: installation, removal, etc. |
| Cap | checks and executes capability |
| Connect | connect to a remotely shared Vagrant environment |
| Destroy | stops and deletes all traces of the vagrant machine |
| docker-exec | attach to an already-running docker container |
| docker-logs | outputs the logs from the Docker container |
| docker-run | run a one-off command in the context of a container |
| global-status | outputs status Vagrant environments for this user |
| halt | stops the vagrant machine |
| help | shows the help for a subcommand |
| init | initializes a new Vagrant environment by creating a Vagrantfile |
| list-commands | outputs all available Vagrant subcommands, even non-primary ones |
| login | log in to HashiCorp's Atlas |
| package | packages a running vagrant environment into a box |
| plugin | manages plugins: install, uninstall, update, etc. |
| port | displays information about guest port mappings |
| powershell | connects to machine via powershell remoting |
| provider | show provider for this environment |
| provision | provisions the vagrant machine |
| push | deploys code in this environment to a configured destination |
| rdp | connects to machine via RDP |
| reload | restarts vagrant machine, loads new Vagrantfile configuration |
| resume | resume a suspended vagrant machine |
| rsync | syncs rsync synced folders to remote machine |
| rsync-auto | syncs rsync synced folders automatically when files change |
| scp | copies data into a box via SCP |
| share | share your Vagrant environment with anyone in the world |
| snapshot | manages snapshots: saving, restoring, etc. |
| ssh | connects to machine via SSH |
| ssh-config | outputs OpenSSH valid configuration to connect to the machine |
| status | outputs status of the vagrant machine |
| suspend | suspends the machine |
| up | starts and provisions the vagrant environment |
| version | prints current and latest Vagrant version |