Extracted from:

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

Anthony Burns
and Tom Copeland

Edited by Brian P. Hogan

2.3 Running Multiple VMs

Vagrant enables us to run multiple guest VMs on a single host. This is handy for configuring database replication between servers, building software firewall rules, tweaking monitoring checks, or almost anything for which it's helpful to have more than one host for proper testing. Let's give it a try now; first we need a new directory to hold our new VM.

```
$ mkdir ~/deployingrails/multiple_vms
$ cd ~/deployingrails/multiple_vms
```

Now we'll need a Vagrantfile. The syntax to define two separate configurations within our main configuration is to call the define method for each with a different name.

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
  end
  config.vm.define :db do |db_config|
  end
end
```

We'll want to set vm.name and a memory size for each VM.

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
    app_config.vm.customize ["modifyvm", :id, "--name", "app", "--memory", "512"]
  end
  config.vm.define :db do |db_config|
    db_config.vm.customize ["modifyvm", :id, "--name", "db", "--memory", "512"]
  end
end
```

We'll use the same box name for each VM, but we don't need to forward port 80 to the db VM, and we need to assign each VM a separate IP address. Let's add these settings to complete our Vagrantfile.

vagrant/multiple_vms/Vagrantfile

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
    app_config.vm.customize ["modifyvm", :id, "--name", "app", "--memory", "512"]
    app_config.vm.box = "lucid64_with_ruby193"
    app_config.vm.host_name = "app"
    app_config.vm.forward_port 22, 2222, :auto => true
    app_config.vm.forward_port 80, 4567
    app_config.vm.network :hostonly, "33.33.13.37"
  end
  config.vm.define :db do |db_config|
    db_config.vm.customize ["modifyvm", :id, "--name", "db", "--memory", "512"]
    db_config.vm.box = "lucid64_with_ruby193"
```

Instances Need Unique Names

If you attempt to start more than one VM with the same name (in other words, the same modifyvm --name value), you'll get an error along the lines of "VBoxManage: error: Could not rename the directory." The fix is to either choose a different instance name for the new VM or, if the old VM is a leftover from previous efforts, just destroy the old VM. Note that halting the old VM is not sufficient; it needs to be destroyed.

```
    db_config.vm.host_name = "db"
    db_config.vm.forward_port 22, 2222, :auto => true
    db_config.vm.network :hostonly, "33.33.13.38"
  end
end
```

Our VMs are defined, so we can start them both with vagrant up. The output for each VM is prefixed with the particular VM's name.

```
$ vagrant up
[app] Importing base box 'lucid64_with_ruby193'...
[app] Matching MAC address for NAT networking...
[app] Clearing any previously set forwarded ports...
[app] Forwarding ports...
[app] -- 22 => 2222 (adapter 1)
[app] -- 80 => 4567 (adapter 1)
[app] Creating shared folders metadata...
[app] Clearing any previously set network interfaces...
[app] Preparing network interfaces based on configuration...
[app] Running any VM customizations...
[app] Booting VM...
[app] Waiting for VM to boot. This can take a few minutes.
[app] VM booted and ready for use!
[app] Configuring and enabling network interfaces...
[app] Setting host name...
[app] Mounting shared folders...
[app] -- v-root: /vagrant
[db] Importing base box 'lucid64_with_ruby193'...
[db] Matching MAC address for NAT networking...
[db] Clearing any previously set forwarded ports...
[db] Fixed port collision for 22 => 2222. Now on port 2200.
[db] Fixed port collision for 22 => 2222. Now on port 2201.
[db] Forwarding ports...
[db] -- 22 => 2201 (adapter 1)
[db] Creating shared folders metadata...
[db] Clearing any previously set network interfaces...
[db] Preparing network interfaces based on configuration...
[db] Running any VM customizations...
[db] Booting VM...
[db] Waiting for VM to boot. This can take a few minutes.
[db] VM booted and ready for use!
```

```
[db] Configuring and enabling network interfaces...
[db] Setting host name...
[db] Mounting shared folders...
[db] -- v-root: /vagrant
```

We can connect into each VM using our usual vagrant ssh, but this time we'll also need to specify the VM name.

```
$ vagrant ssh app
Last login: Wed Dec 21 19:47:36 2011 from 10.0.2.2
app $ hostname
app
app $ exit
$ vagrant ssh db
Last login: Thu Dec 22 21:19:54 2011 from 10.0.2.2
db $ hostname
db
```

Generally, when we apply any Vagrant command to one VM in a multiple-VM cluster, we need to specify the VM name. For a few commands (for example, vagrant halt), this is optional, and we can act on all VMs by not specifying a host name.

To verify inter-VM communications, let's connect from db to app via ssh using the vagrant account and a password of vagrant.

```
db $ ssh 33.33.13.37
The authenticity of host '33.33.13.37 (33.33.13.37)' can't be established.
RSA key fingerprint is ed:d8:51:8c:ed:37:b3:37:2a:0f:28:1f:2f:1a:52:8a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '33.33.13.37' (RSA) to the list of known hosts.
vagrant@33.33.13.37's password:
Last login: Thu Dec 22 21:19:41 2011 from 10.0.2.2
app $
```

Finally, we can shut down and destroy both VMs with vagrant destroy.

```
$ vagrant destroy --force
[db] Forcing shutdown of VM...
[db] Destroying VM and associated drives...
[app] Forcing shutdown of VM...
[app] Destroying VM and associated drives...
```

We used two VMs in this example, but Vagrant can handle as many VMs as you need up to the resource limits of your computer. So, that ten-node Hadoop cluster on your laptop is finally a reality.