Repost from freshconsulting.com:

Have you ever managed a remote team and needed to get developers or designers set up so they could start contributing to a project? How many hours (or days!) do you wish you could get back?

Earlier this year Hashicorp released a product that changed the face of team-based development. Vagrant was originally a simple product and brainchild of Mitchell Hashimoto but has rapidly grown into a full-fledged enterprise-supported platform for application development.

Since then Vagrant has aimed to do one thing very well: *"Create and configure lightweight, reproducible, and portable development environments."* You might be wondering why this is important, or how additional steps and yet more apps and "things" to install can be anything but a pain for a development team.

I have been a part of or managed teams of developers for the last twelve years, and I've seen the benefits of Vagrant Up increase the efficiency and flexibility of development projects, even for a one-person team.

What Is Vagrant?

Vagrant is a tool for building and managing development environments with a heavy focus on configuration, and most importantly on automation. Leveraging other tools for core components like virtualization and configuration, Vagrant combines the vital parts of your application into a simple command that will get anyone on your team, on any workstation up and *working* in moments.

Why Should I Use Vagrant Up?

Have you ever had to change workstations and needed to work on your project? Remember how long it took to get to a point where you could actually get to work?

Have you ever been a manager and heard a team member say, “I don’t know why, it worked locally!”

Do you travel and wish you could work on the road away from your primary workstation without sacrificing components of your development environment?

Do you need to give a demo and can’t access your staging/demo server for whatever reason? Imagine being able to spin up a demo server on any workstation in seconds.

What About Production Environments?

The idea behind Vagrant is that the same scripts and/or tools used to provision and “launch” development environments are used to launch your production environments, keeping the two as close to identical as possible.

Chef, Puppet, Salt, Ansible, and other tools are used alongside Vagrant to make this utopian scenario come to life, and speaking from experience I can assure skeptical system administrators that it is possible!

Where Do I Start?

A computer with Linux, OSX, or Windows installed is all you need to implement Vagrant and get your development environment up and running in seconds. Let’s first break down the primary components and explain what we’re doing.

Here’s how to build a development environment on UBUNTU 14.04 with a traditional LEMP stack.

First, install all the necessary software:

- Vagrant
- VirtualBox
- Ansible on Windows:

- Ansible on Linux/MAC:

COPY

```
$ sudo easy_install pip
$ sudo pip install ansible
$ vagrant plugin install ansible
```

At this point you have all the tools necessary to get started!

Take this Vagrantfile for example, and run with it.

COPY

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
ip_address = "192.168.0.1"
hostname = "myapp.dev"

Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "private_network", ip: ip_address
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "provisioning/vagrant.yml"
    ansible.extra_vars = {
      hostname: hostname
    }
  end
end
```

Now the configuration magic happens in your Ansible playbooks, which you'll be creating in

your provisioning directory.

This example calls provisioning/vagrant.yml, which does a very basic installation of composer and some common software packages, and finally enables php-mcrypt:

COPY

```
# vim:ft=ansible:
#
---
- name: install common software
  apt:
    name={{ item }}
    state=present
    with_items:
      - apache2
      - git
      - imagemagick
      - make
      - mcrypt
      - mysql-server
      - npm
      - php5
      - php5-curl
      - php5-dev
      - php5-mcrypt
      - php5-mysql
      - pkg-config

  - name: install composer
  shell: curl -sS https://getcomposer.org/installer | php && mv
composer.phar /usr/local/bin/composer
```

```
args:
  creates=/usr/local/bin/composer
- name: enable php-mcrypt
command: php5enmod mcrypt
args:
  creates: /etc/php5/cli/conf.d/20-mcrypt.ini
notify: restart apache
```

Now is when the real fun starts: Ansible playbooks and deep dive configuration are at your fingertips.

Bringing It All Together

Once this initial setup is complete, you'll be able to hand off your environment to any developer/designer, and with one simple command they can be up and running with an exact replica of your environment, dramatically reducing onboarding and setup time for growing and evolving teams.

Share this:

- LinkedIn
- Twitter
- Email
- Print