WHY USE VAGRANT?

For years, I've been using a VM (virtual machine) based workflow. I develop on Windows 7 primarily, but I always deploy my apps to some sort of Linux stack. This leads to a disconnect, a potential area for problems. There are many differences between Windows and Linux, many of which affect PHP, Node.js, Ruby, etc, languages I work with. I didn't want to develop my apps on Windows, and then find out they don't work quite right on my testing and production environments.

To solve this problem, I run Linux in VMWare workstation. I used shared folders so I could edit all my code in Windows using my IDE, and access it all from Linux. I only ever used Linux as a server, running a mirror of my testing/production environments. I mapped it to host names like http://brandonwamboldt.dev/ by editing my hosts file. This worked great for me, but it's not perfect for a team. If everyone did this, or more commonly, some people did this, some people run WAMP, and some people are on Macs, you end up with a lot of different environments.

Recently I decided to try out Vagrant, and it solves all of these problems. It's website does a poor job of communicating why you'd use the software, so let me try and sum it up. You create a file that defines the OS, some port forwarding

information, and some provisioning information (Aka use shell scripts, Puppet manifests, or Chef recipes to provision the machine). Then you define your provisioning scripts using your method of choice to setup the environment however you want (aka install Apache, MySQL, PHP, run Composer to fetch dependencies, then import some dummy data into MySQL). You commit both the VM config (Your `Vagrantfile`) and your provisioning scripts to the same repo as your project. Then, whenever somebody clones your project, they just run `vagrant up` from the project directory to provision an identical VM to the one you used for development.

Yes, this means you have a different VM for every project, but it affords so many benefits that it's worth it. Also, Vagrant does way more than what I just said. You can run `vagrant suspend` and `vagrant resume` to suspend/resume your VM, `vagrant halt` to stop it, `vagrant destroy` to delete it and then `vagrant up` to create it all over again. Your VMs are completely disposable, and only take a single command to setup again.

You can still SSH into your VMs by typing `vagrant ssh`. Virtual environments have never been so easy!

Source: http://brandonwamboldt.ca/why-use-vagrant-1236/