Working with Advanced Vagrant features in PhpStorm

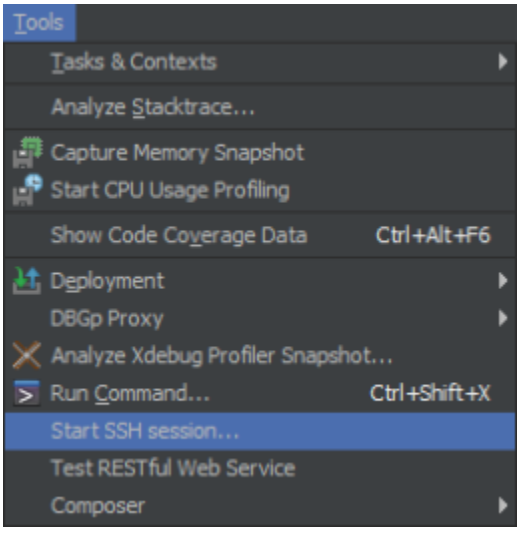
This tutorial describes how to work with more advanced Vagrant features in PhpStorm.

1. Using the built-in SSH terminal to connect to the Vagrant machine

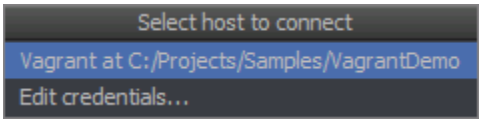
PhpStorm features a built-in SSH terminal which can be used to connect to a remote machine.

1.1. Start a connection

From the Tools | Start SSH session... menu, we can connect to the Vagrant machine.



This will open a list of hosts we can connect to.



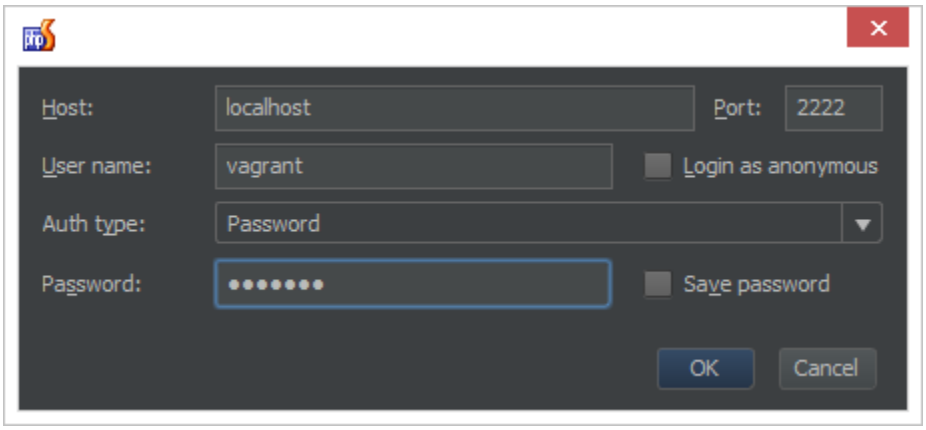
Our Vagrant machine should automatically be added to this list. Clicking it will open a connection to the SSH endpoint exposed by our Vagrant machine.

The Edit credentials... item allows us to provide connection information manually.

1.2. Provide connection information

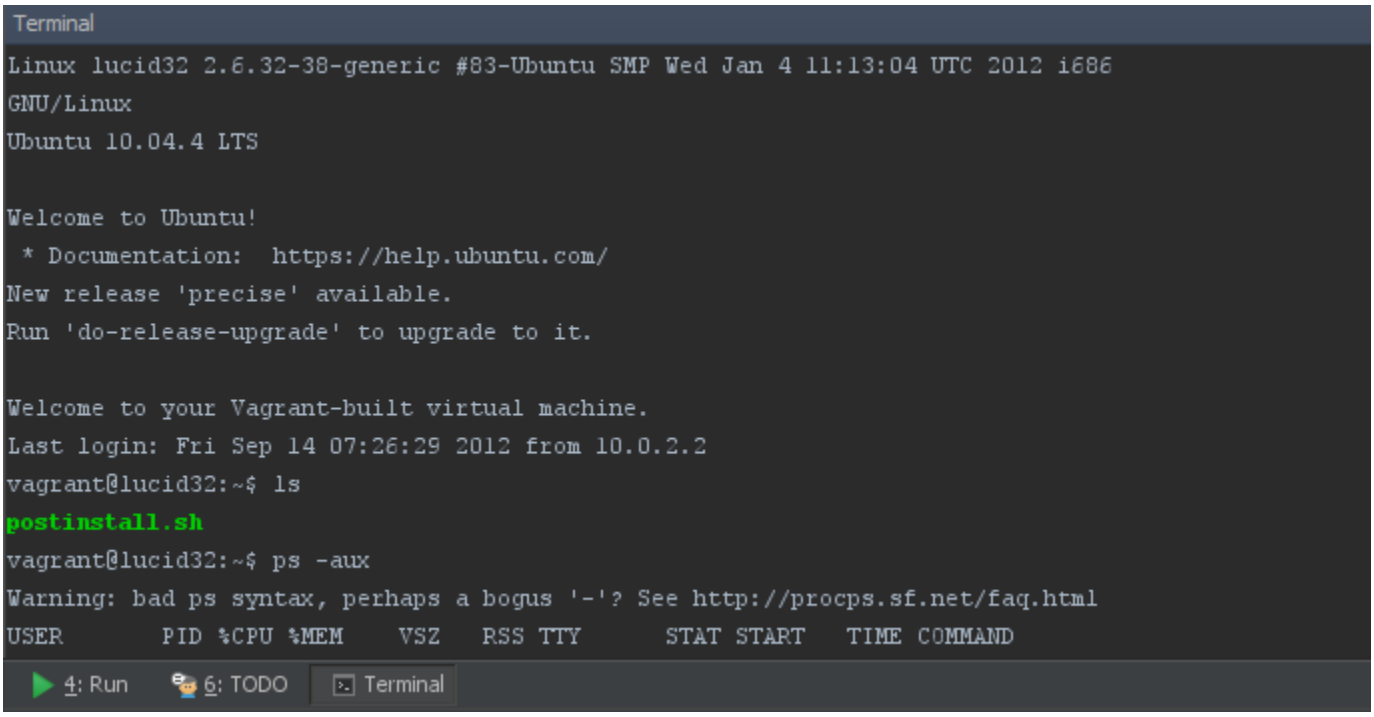
Next, we have to provide connection information. For the default lucid32 box, we can use the following values. Note that for other Vargrant Boxes this information may be different.

- Host: localhost
- Port: 2222 (which is forwarded to the Vagrant machine)
- User name: vagrant
- Password: vagrant



1.3. Working with SSH

After we click OK, PhpStorm will connect to the Vagrant machine using SSH server and show us a terminal to work with.



In the SSH terminal, we can run commands remotely as well as copy/paste data back and forth.

2. Working with shared folders

Vagrant allows sharing folders between the host machine and the Vagrant machine. They can be used, for example, to automatically provide web root contents from the current PhpStorm project to the Apache virtual host directory on the Vagrant machine.

2.1. Add a path mapping

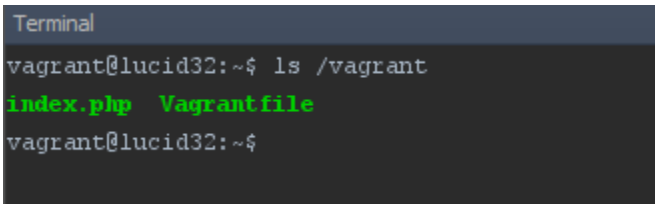
From the VagrantFile, we can add path mappings by adding a configuration entry for it:

```
Vagrant.configure("2") do |config|
  config.vm.synced_folder "src/", "/srv/website"
end
```

2.2. Vagrant reload

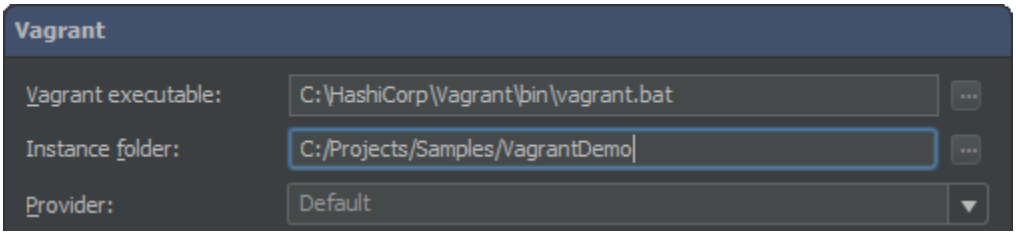
Reloading the VagrantFile can be done using the Tools | Vagrant | Reload menu. Once the Vagrant machine has been reloaded, a new path mapping will be available.

For example, when connecting to the Vagrant machine using the built-in SSH terminal, we can see the contents of the /vagrant folder which map to the PhpStorm local project folder. Be careful: deleting files from this folder will delete files on both ends!



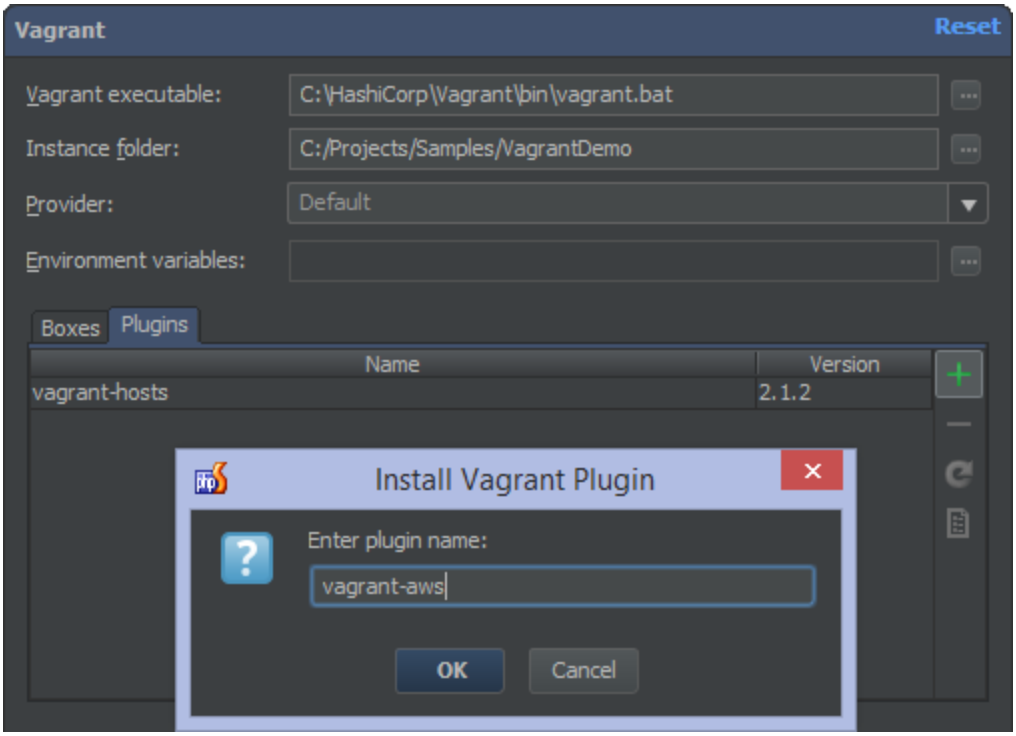
3. Specifying Vagrant instance folder

By default, the Vagrantfile and all other Vagrant specifics (like Puppet manifests) are placed in the root of a PhpStorm project. Since this is not always desired, the instance folder where the IDE should look for Vagrantfile can be configured through Project Settings | Vagrant.



4. Manage Vagrant plugins through settings

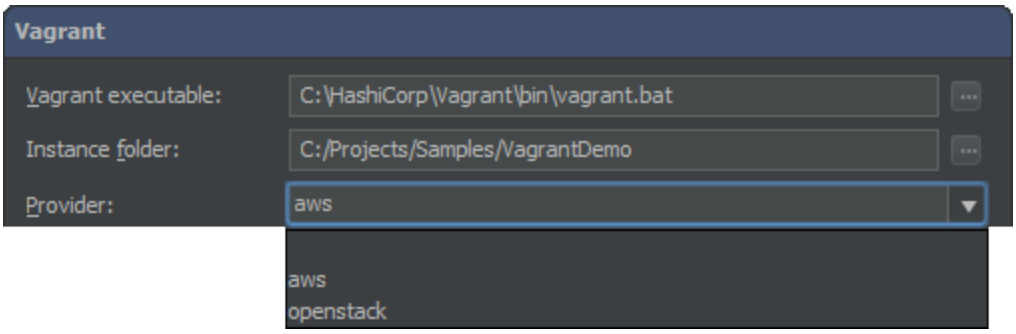
Through the Project Settings | Vagrant, Vagrant plugins can be managed. Use the toolbar buttons to install, uninstall and update plugins. Licenses can also be installed, for example for the VMWare Fusion Provider which allows running Vagrant machines on VMWare.



5. Providers support

Vagrant works with Oracle VirtualBox as the virtualization platform by default. Using providers, the virtualization platform can be changed and so virtual machines can be run by a system other than VirtualBox, such as VMWare or Amazon EC2. A list of available providers can be found on the Vagrant plugins list.

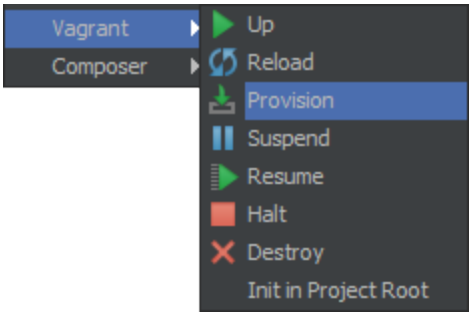
The provider to be used has to be passed to Vagrant for every command. To make this easier and have PhpStorm automatically add the provider name to every Vagrant command, we can specify the provider through Project Settings | Vagrant. All providers installed on our machine will be available from the settings. Once selected, PhpStorm will execute all Vagrant commands using the provider configured.



6. (Re-)provisioning a Vagrant machine

A Vagrantfile (the Vagrant configuration file) can contain a series of provisioners that can launch installation and configuration routines once a virtual machine is running. The Provision command invokes the configured provisioners on an already running Vagrant machine, without having to first destroy the virtual machine.

Using the Tools | Vagrant menu, we can run provisioning on a running environment.

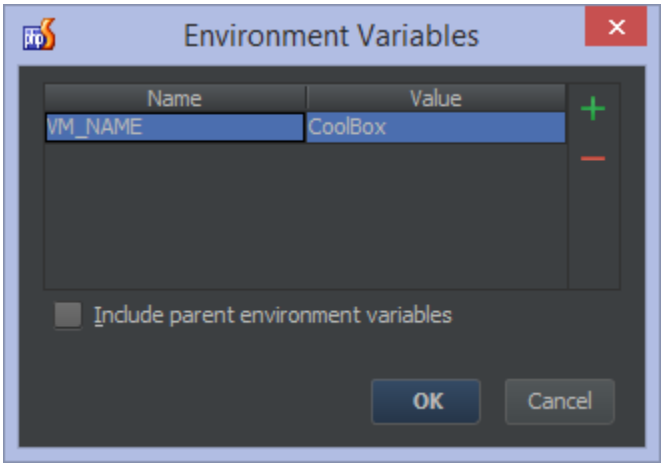


7. Working with Environment variables

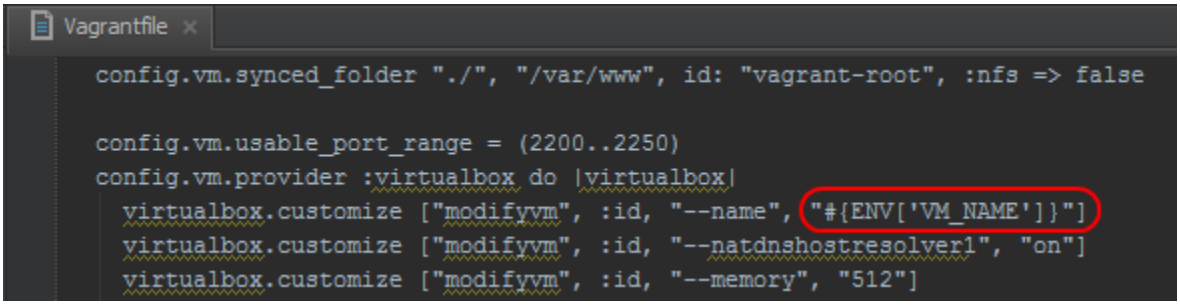
Working with environment variables in Vagrantfile is very useful for doing several things, like:

- setting the Puppet node
- setting the Puppet environment
- setting custom facts
- setting AWS keys
- ...

From the Project Settings | Vagrant we can specify project-specific environment variables that will be passed into the Vagrantfile.



Once set, we can make use of them in our Vagrantfile using the #{ENV['name_of_variable']} syntax:



8. Remote PHP Interpreters

The Vagrant virtual machine's PHP interpreter can be used as a remote PHP interpreter in PhpStorm. This lets us run our application and PHP-based tools on a production-like environment (our Vagrant machine), for example for running PHPUnit tests on the Vagrant machine. We can install only PhpStorm on our development machine, and run, debug and test our application on the Vagrant machine.