Configuring PhpStorm to work with a VM

Sometimes we want to work on a project on one platform, but deploy and run it on another. That's where PhpStorm helps a lot with its extensive support for remote interpreters and remote environments. In this tutorial, we'll see how we can configure PhpStorm to work on a virtual machine using Vagrant.

Prerequisites

We'll need a virtual machine on which we will deploy and run our code. We'll use Oracle's VirtualBox and Vagrant for this, both of which have to be installed on our system.

1. Download and install Vagrant

Download and install Vagrant from their official website.

Verify that *`vagrant.bat`* / *`vagrant.sh`* are added to the system path. This should be done automatically by the installer.

2. Download and install Oracle VirtualBox

From the VirtualBox website, download and install the latest version of VirtualBox.

Make sure that *`VBoxManage.exe`* from Oracle's VirtualBox installation is added to the system path.

3. Make sure the Vagrant plugin is installed and enabled in PhpStorm

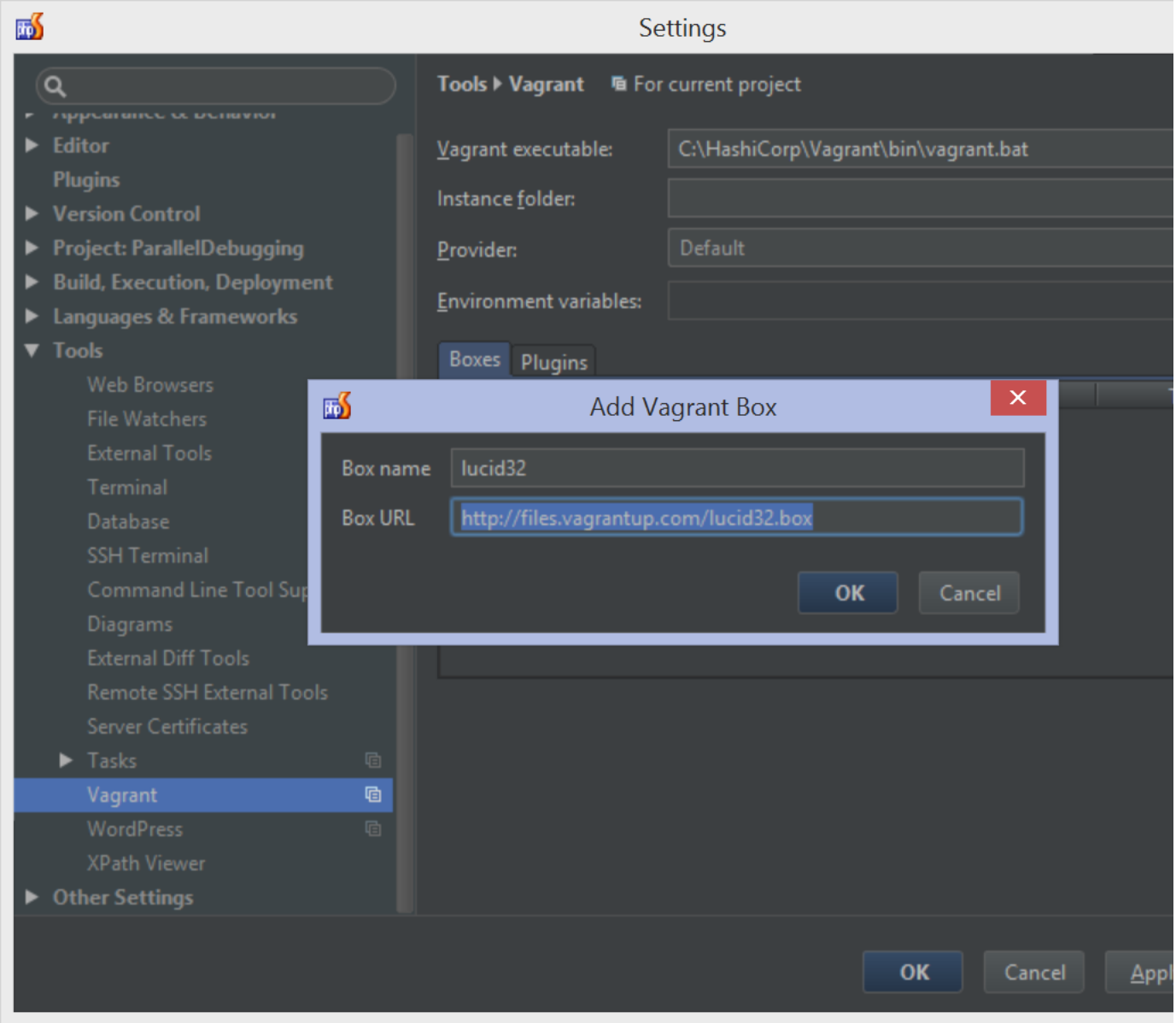
From the settings (Ctrl+Alt+S / CMD-,), click the Plugins node and make sure the Vagrant plugin is installed and enabled. PhpStorm bundles this plugin and enables it by default.

1. Creating a virtual box

1.1. Downloading a virtual box

From the settings (Ctrl+Alt+S / CMD-,), expand the Tools | Vagrant node and enter the Vagrant executable and Vagrant instance folder.

If boxes are already defined, they will appear in the list and we can select one. If there is no suitable virtual box, click the green + button to add a new one. PhpStorm will provide the name and URL to the lucid32 box by default; other boxes can be specified as well.



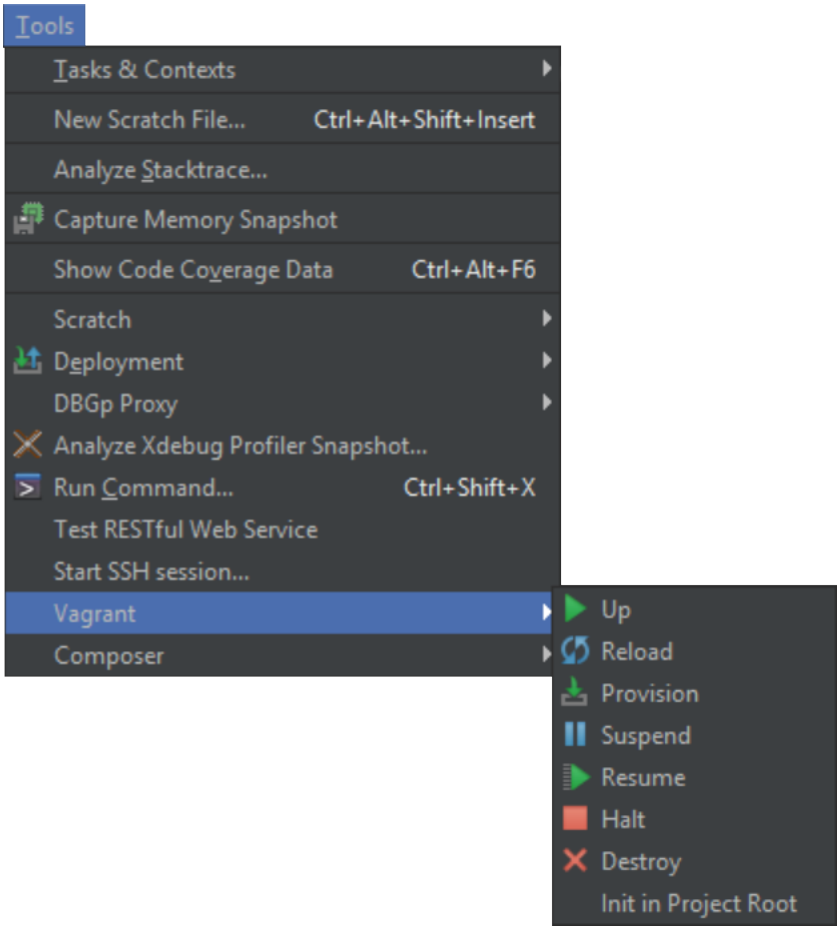
Clicking OK downloads the virtual box and makes it available for use with Vagrant.

1.2. Initialize VagrantFile

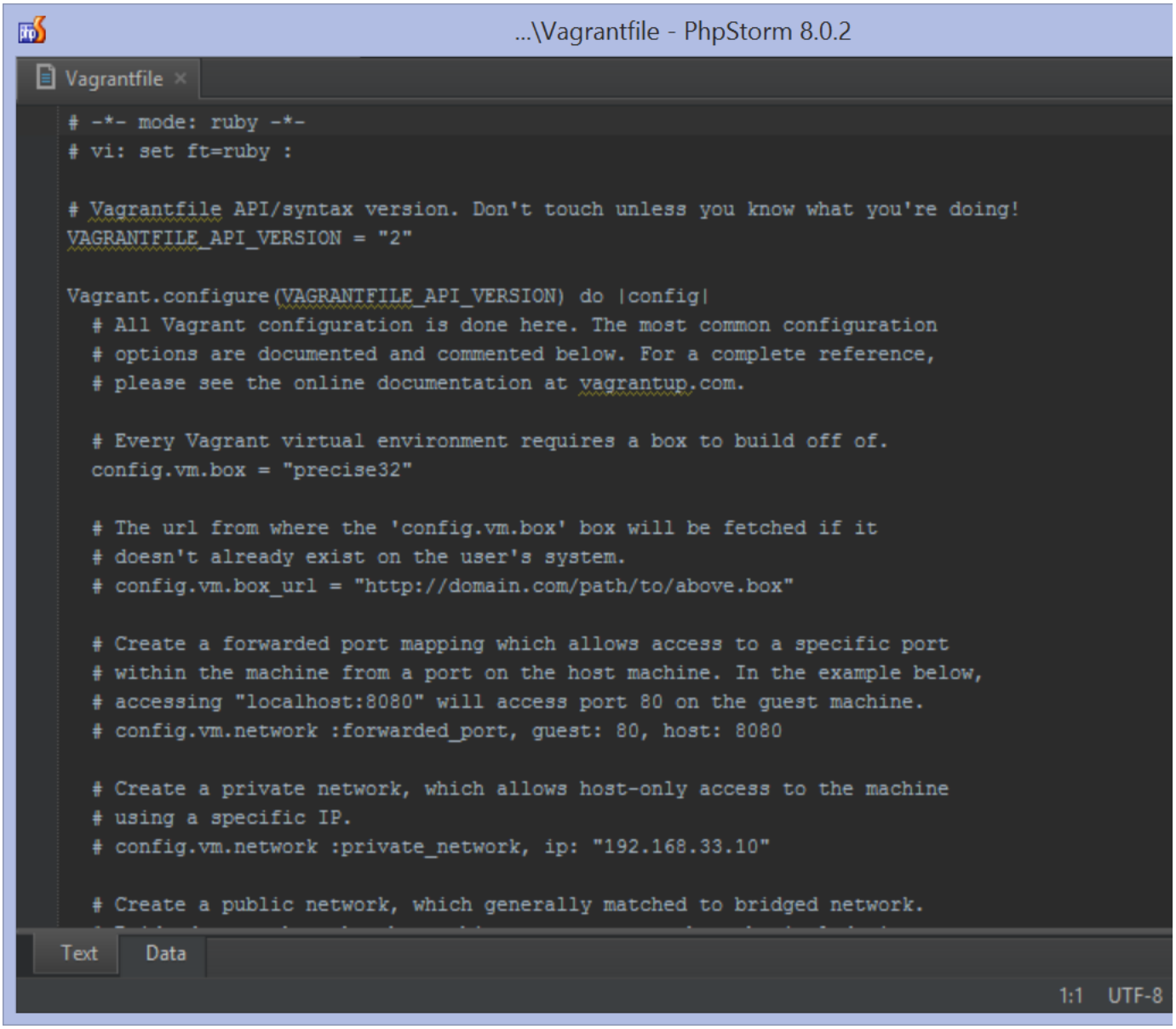
Vagrant uses a file named *`VagrantFile`*, containing all information about the virtual machine we will work with. It contains details such as the virtual IP address, port mappings and amount of memory to assign. Next to that, it can specify which folders are shared and which third-party software should be installed on the machine.

> The PuPHPet website has a great tool for configuring a Vagrant machine and generating a VagrantFile that contains popular software and tools to get started with.

The VagrantFile can be created using the Tools | Vagrant | Init in Project Root menu.

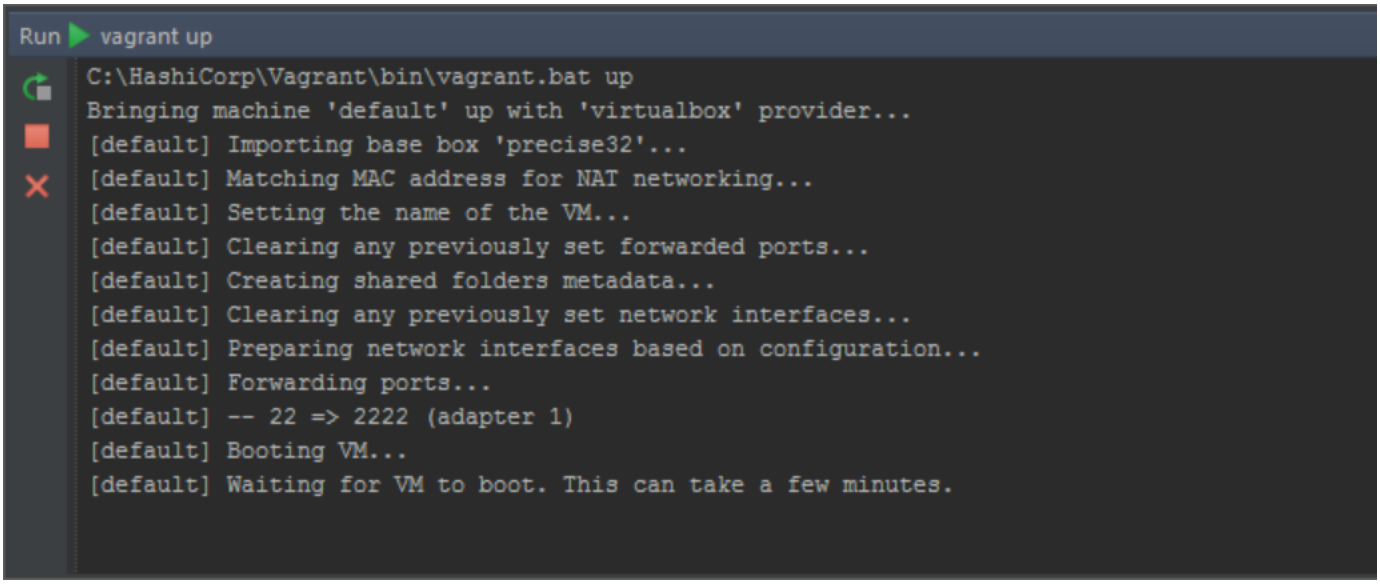


This will trigger creation of a default VagrantFile in the root of our project.



1.3. Run the virtual box

The Vagrant machine can be started from the Tools | Vagrant | Up menu. This will configure the virtual machine in VirtualBox and boot it. The Run tool window shows us the progress of starting the machine.



Once completed, the Vagrant machine will be ready for use. We can now continue Working with Advanced Vagrant features in PhpStorm.

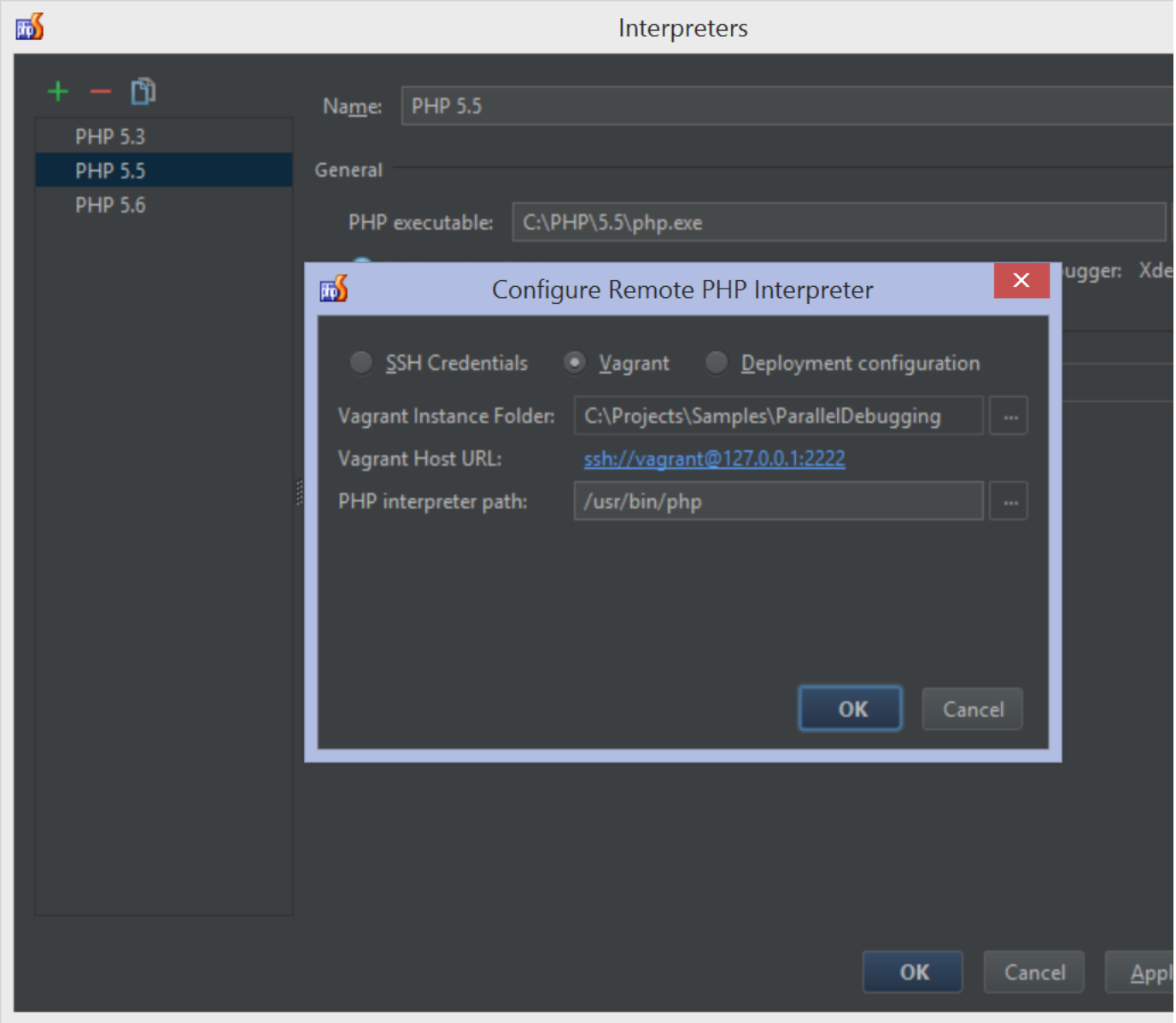
When the VagrantFile is modified, use the Tools | Vagrant | Reload menu to make sure the virtual machine is updated with the latest changes.

2. Configuring the remote PHP interpreter in the virtual box

2.1. Registering a new PHP interpreter

We can specify the PHP interpreter we want to use from the settings, under Languages & Frameworks | PHP. Clicking ... next to the selected interpreter opens a list of registered PHP interpreters and lets us add the remote PHP interpreter from our virtual machine.

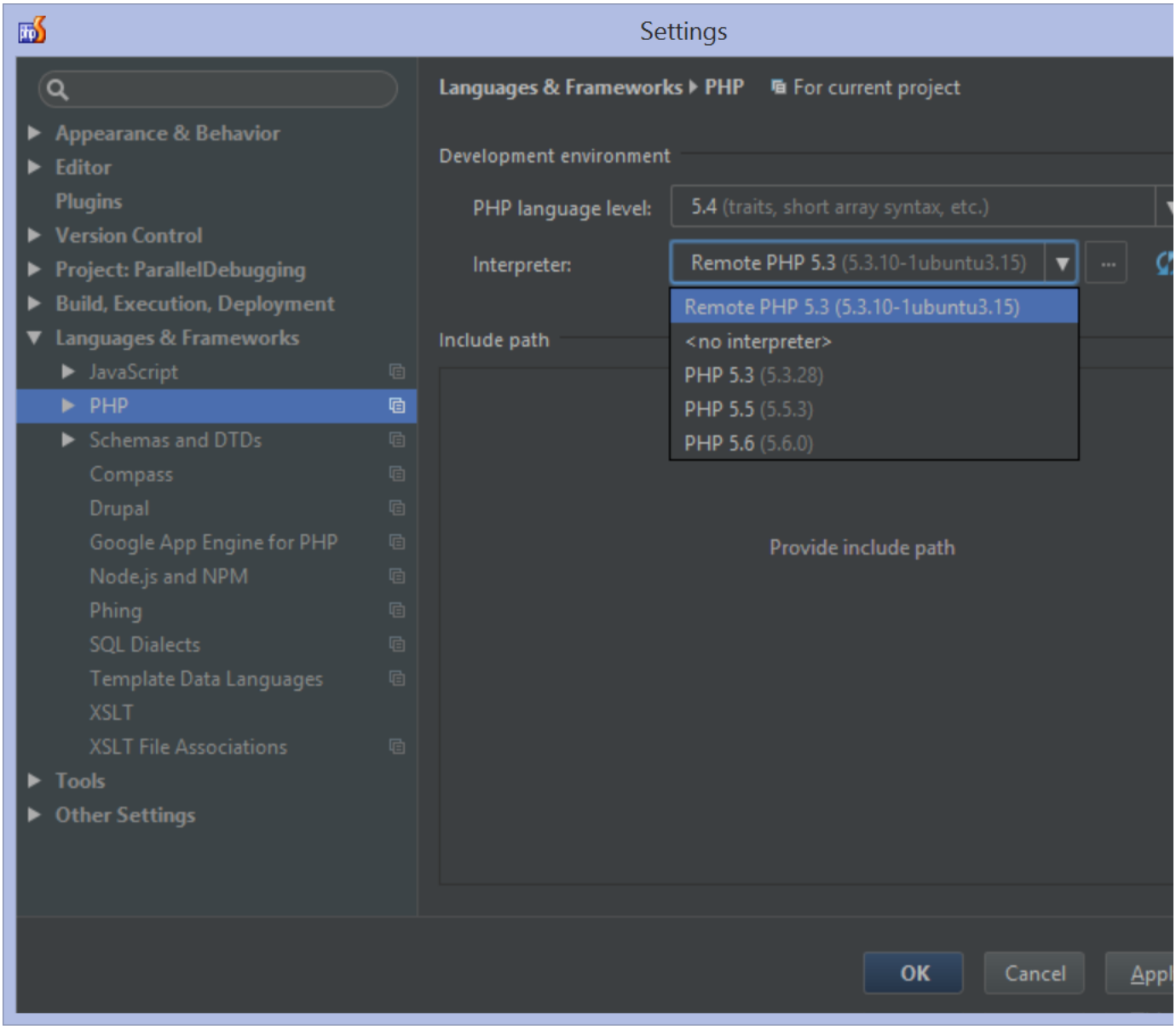
Click the green + and add a new Remote.... We can select the Vagrant option to automatically populate connection details. In most cases, PhpStorm can also figure out the path to the PHP interpreter on the Vagrant machine. If not, we can adjust it manually.



The lucid32 box does not come with PHP installed by default. To install it, connect to the SSH terminal and run *sudo apt-get install php5-cli --fix-missing*.

2.2. Select the remote PHP interpreter for the current project

Once we register the remote PHP interpreter, we can set up our project to use it during development. Simply select it in the dropdown list in Languages & Frameworks | PHP.

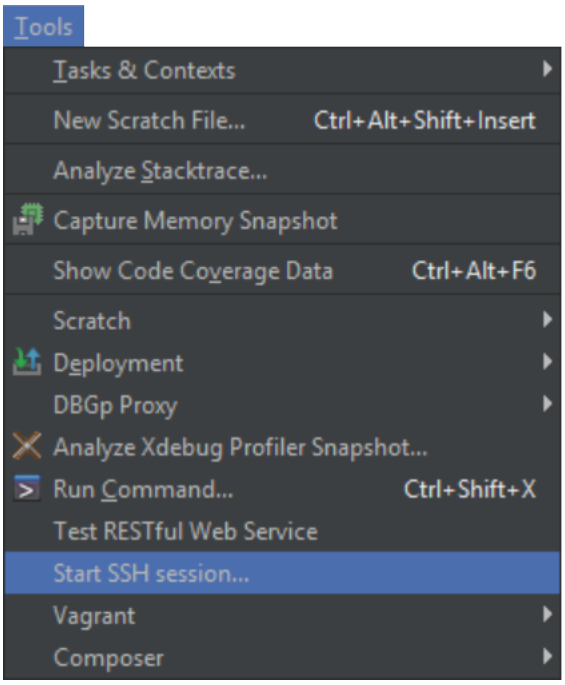


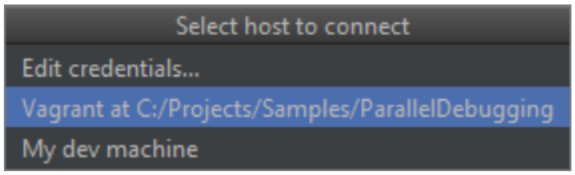
From now on, all our Run/Debug configurations will make use of the configured PHP interpreter.

Learn more about remote interpreters in our Working with Remote PHP Interpreters in PhpStorm tutorial. It contains some tips and tricks on how to run scripts, unit tests and Composer.

3. Connecting to the SSH terminal

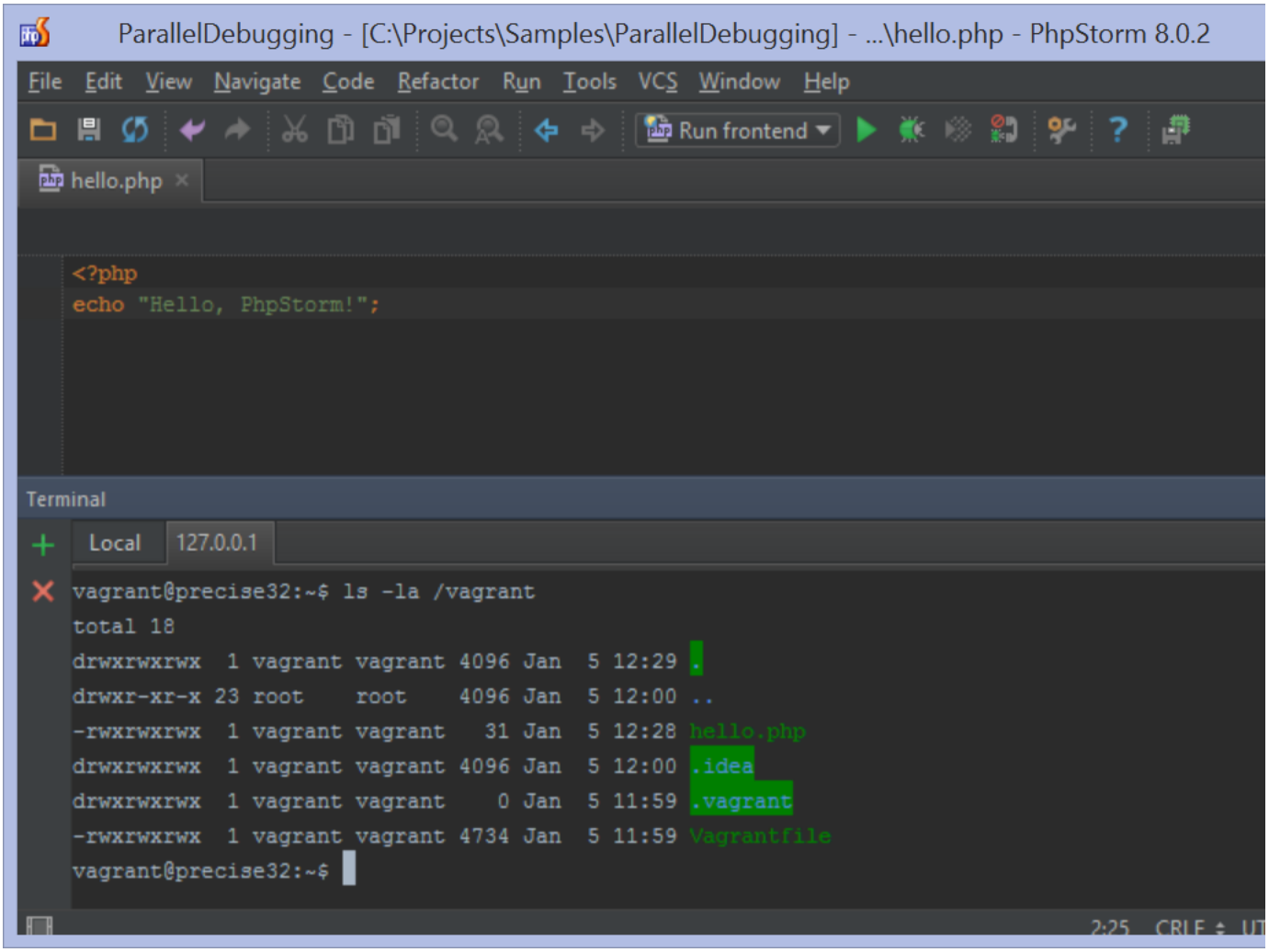
PhpStorm lets us log in to our virtual box via SSH and work in its console, without leaving the IDE. Using the Tools | Start SSH session... menu, we can select our Vagrant machine from the window that opens. If more than one host is already defined, we can select the one we want to connect to (in our case, this is our remote interpreter):





To always connect to the current Vagrant machine, open the Tools | SSH Terminal node from the settings and select C urrent Vagrant as the default.

Once connected, we can work with the SSH terminal on the remote machine. The `/vagrant` path in the virtual machine is mapped to our project directory. For example if we do *`ls -la /vagrant`*, we will see the contents of our PhpStorm project folder.



The Using the PhpStorm built-in SSH terminal and remote SSH external tools tutorial explains additional tips and tricks for working with the built-in SSH terminal.

For help on installing and configuring Xdebug on your newly running Vagrant VM, see Configuring a Vagrant VM for Debugging.

For additional help on VM-based development in PhpStorm, check out the tutorials Vagrant Support in PhpStorm and Working with Remote PHP Interpreters in PhpStorm.

Tweet