

Document Title	MailHog, Xdebug, PHPCS, PHPMD Integration
Client (Merchant)	Kurt Geiger
SOW ID	
Document Version	1.0.0
Author	Arif Ahmad
Reviewer(s)	Deepak Moktan

# Contents

---

Overview:.....	2
Use Cases:.....	2
Mailhog: .....	2
Xdebug: .....	2
PHPCS: .....	2
PHPMD: .....	3
Mailhog:.....	3
Download Mailhog: .....	3
Configure MailHog .....	4
Initialization.....	5
Xdebug.....	7
Install Xdebug.....	7
Installing PHP Interpreter, PHPCS, PHPMD .....	9

# Overview:

---

This document is created as guideline to local development setup within Kurt Geiger namespace within the scope of GRA (Global Reference Architecture).

This document outlines the steps necessary to integrate MailHog, Xdebug, PHPCS, PHPMD and also Magento coding standard so that we can do development more easily and perform different tasks including but not limited to the following use cases.

# Use Cases:

---

## Mailhog:

Used for the emails testing including but not limited to the following:

- New Customer registration email
- Forgot password email
- Account activation email
- Forgot password email for admin
- Newsletter (un)subscription email
- Order related emails
- Shipment related emails
- Invoice related emails
- Credit memo related emails
- RMA related emails

## Xdebug:

For debugging of the issues and traceback.

## PHPCS:

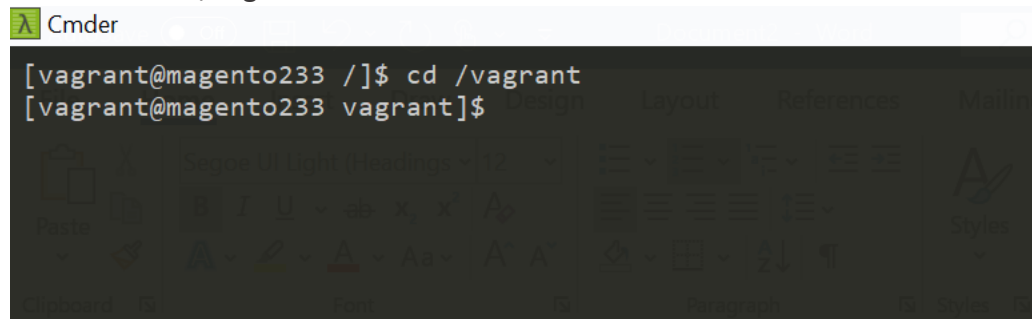
Code sniffing tool for Magento/PHP best practices and see the issues while doing development.

Issues related to coding standard and detecting issues like code redundancy.

## Mailhog:

### Download Mailhog:

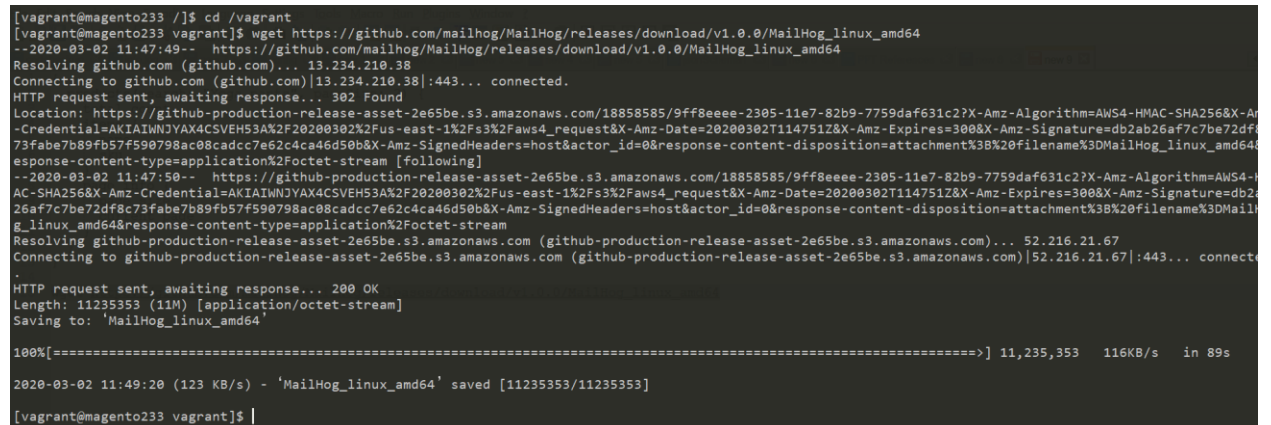
1. Change the current folder to your project directory. For example, if your working project directory is /vagrant,
  - a. **Command:** cd /vagrant



```
[vagrant@magento233 /]$ cd /vagrant
[vagrant@magento233 vagrant]$
```

2. Install Mailhog using wget command

[https://github.com/mailhog/MailHog/releases/download/v1.0.0/MailHog\\_linux\\_amd64](https://github.com/mailhog/MailHog/releases/download/v1.0.0/MailHog_linux_amd64)



```
[vagrant@magento233 /]$ cd /vagrant
[vagrant@magento233 vagrant]$ wget https://github.com/mailhog/MailHog/releases/download/v1.0.0/MailHog_linux_amd64
--2020-03-02 11:47:49-- https://github.com/mailhog/MailHog/releases/download/v1.0.0/MailHog_linux_amd64
Resolving github.com (github.com)... 13.234.210.38
Connecting to github.com (github.com)|13.234.210.38|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/18858585/9ff8eeee-2305-11e7-82b9-7759daf631c2?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200302%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200302T114751Z&X-Amz-Expires=300&X-Amz-Signature=db2ab26af7c7be72df73fab7b89fb57f590798ac08cadcc7e62c4ca46d50b8X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3DMailHog_linux_amd64
response-content-type=application%2Foctet-stream [following]
--2020-03-02 11:47:50-- https://github-production-release-asset-2e65be.s3.amazonaws.com/18858585/9ff8eeee-2305-11e7-82b9-7759daf631c2?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200302%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200302T114751Z&X-Amz-Expires=300&X-Amz-Signature=db2ab26af7c7be72df73fab7b89fb57f590798ac08cadcc7e62c4ca46d50b8X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3DMailHog_linux_amd64&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.21.67
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.21.67|:443... connect
HTTP request sent, awaiting response... 200 OK
Length: 11235353 (11M) [application/octet-stream]
Saving to: 'MailHog_linux_amd64'

100%[=====] 11,235,353 116KB/s in 89s

2020-03-02 11:49:20 (123 KB/s) - 'MailHog_linux_amd64' saved [11235353/11235353]

[vagrant@magento233 vagrant]$
```

## Configure MailHog

1. Make the changes in php configuration file.

```
[vagrant@magento233 vagrant]$ sudo vi /etc/php.ini
```

2. php.ini can be located at a different place as well. So, you can get it like this.

```
$ php -i | grep php.ini
Configuration File (php.ini) Path => /etc/php/7.2/cli
Loaded Configuration File => /etc/php/7.2/cli/php.ini
```

3. So, there are 2 files you should modify one is `/etc/php/7.2/cli/php.ini` and another one is `/etc/php/7.2/fpm/php.ini`
4. Comment the "sendmail" path

```
;phar.cache_list =

[mail function]
; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
sendmail_path = /usr/sbin/sendmail -t -i

; Force the addition of the specified parameters to be passed as extra parameters
; to the sendmail binary. These parameters will always replace the value of
; the 5th parameter to mail().
;mail.force_extra_parameters =
```

```
;phar.cache_list =

[mail function]
; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
; sendmail_path = /usr/sbin/sendmail -t -i
```

5. Add new mail path

```
;phar.cache_list =

[mail function]
; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
; sendmail_path = /usr/sbin/sendmail -t -i
sendmail_path = "/vagrant/MailHog_linux_amd64 sendmail test@example.com"
```

6. Restart the php-fpm service

```
[vagrant@magento233 vagrant]$ sudo service php-fpm restart
```

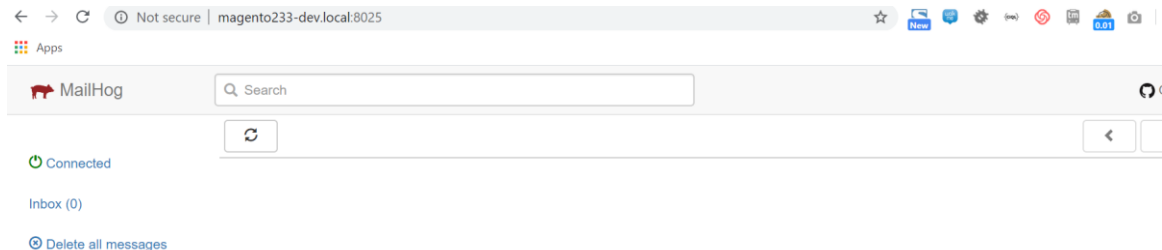
7. The command to restart php-fpm may vary depending on the installation you have. For example if you have more than one php version installed then you might have to run something like this - `sudo service php7.2-fpm restart`

## Initialization

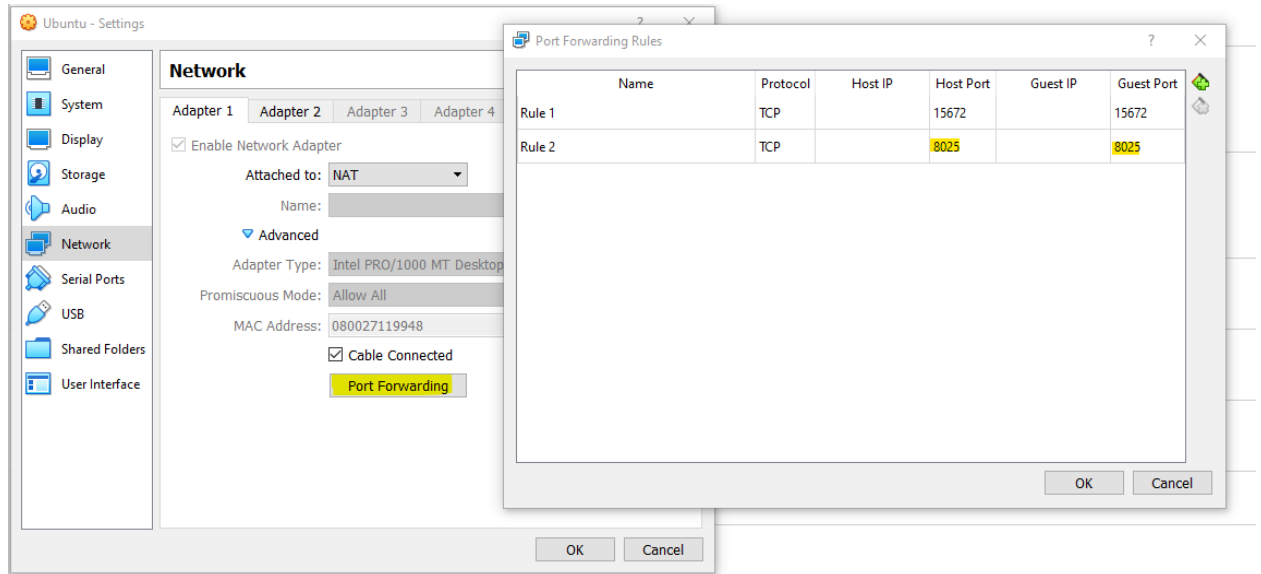
1. Create the mail directory
  - a. Command: `mkdir maildir`
2. Set the permission
  - a. Command: `chmod -R 777 MailHog_linux_amd64`
  - b. Make it executable: `chmod +x MailHog_linux_amd64`
3. Start the MailHog
  - a. Command: `./MailHog_linux_amd64 -storage "maildir" -maildir-path "/vagrant/maildir/"`

```
[vagrant@magento233 vagrant]$ sudo service php-fpm restart
Redirecting to /bin/systemctl restart php-fpm.service
[vagrant@magento233 vagrant]$ whereis sendmail
sendmail: /usr/sbin/sendmail /usr/sbin/sendmail.sendmail /usr/sbin/sendmail.postfix /usr/lib/sendmail /usr/share/man/man8/sendmail.8.gz
[vagrant@magento233 vagrant]$ sudo mkdir maildir
[vagrant@magento233 vagrant]$ sudo chmod -R 777 MailHog_linux_amd64
[vagrant@magento233 vagrant]$ ./MailHog_linux_amd64 -storage "maildir" -maildir-path "/vagrant/maildir/"
2020/03/03 06:20:47 Using maildir message storage
2020/03/03 06:20:47 Maildir path is /vagrant/maildir/
2020/03/03 06:20:47 [SMTP] Binding to address: 0.0.0.0:1025
[HTTP] Binding to address: 0.0.0.0:8025
2020/03/03 06:20:47 Serving under http://0.0.0.0:8025/
Creating API v1 with WebPath:
Creating API v2 with WebPath:
```

- b. You can choose the path as per the setup you have. It need not necessarily be in `/vagrant/maildir/` folder. You can also put in `/opt/mailhog/`
  - c. If you want to keep running mailhog in background then add “&” at the end of command in point “a”.
4. Access mails : <http://magento233-dev.local:8025/>



5. For remotely accessing from VM you can add a port forwarding like below:



And then can access from link - <http://localhost:8025/>



Adobe

# Xdebug

---

## Install Xdebug

1. Install xdebug using the below command:

- a. `sudo dnf install php-xdebug`

```
[vagrant@magento233 /]$ sudo dnf install php-xdebug
```

2. Check whether xdebug has been installed or not using the below command:  
`php -r "echo (extension_loaded('xdebug') ? 'xdebug up and running' : 'xdebug is not loaded');"`

```
[vagrant@magento233 /]$ php -r "echo (extension_loaded('xdebug') ? 'xdebug up and running' : 'xdebug is not loaded');"
xdebug up and running[vagrant@magento233 /]$
```

3. Check the correct paths using the below command:  
`repoquery --list php73-pec1-xdebug-2.7.2-2.el7.ius.x86_64`

```
[vagrant@magento233 php.d]$ sudo repoquery --list php73-pec1-xdebug-2.7.2-2.el7.ius.x86_64
/etc/php.d/15-xdebug.ini
/usr/bin/debugclient
/usr/lib64/php/modules/xdebug.so
/usr/share/doc/pec1/xdebug
/usr/share/doc/pec1/xdebug/CREDITS
/usr/share/doc/pec1/xdebug/README.rst
/usr/share/doc/pec1/xdebug/contrib
/usr/share/doc/pec1/xdebug/contrib/tracefile-analyser.php
/usr/share/doc/pec1/xdebug/contrib/xt.vim
/usr/share/doc/pec1/xdebug/xdebug.ini
/usr/share/licenses/php73-pec1-xdebug-2.7.2
/usr/share/licenses/php73-pec1-xdebug-2.7.2/LICENSE
/var/lib/php/peclxml/xdebug.xml
[vagrant@magento233 php.d]$
```

4. Edit the xdebug.ini to enable the xdebug and add the below lines:

`xdebug.remote_log="/tmp/xdebug.log"`

`xdebug.profiler_enable = 1`

`xdebug.remote_enable=on`

`xdebug.remote_port=9000`

`xdebug.remote_autostart=0`

`xdebug.remote_connect_back=on`

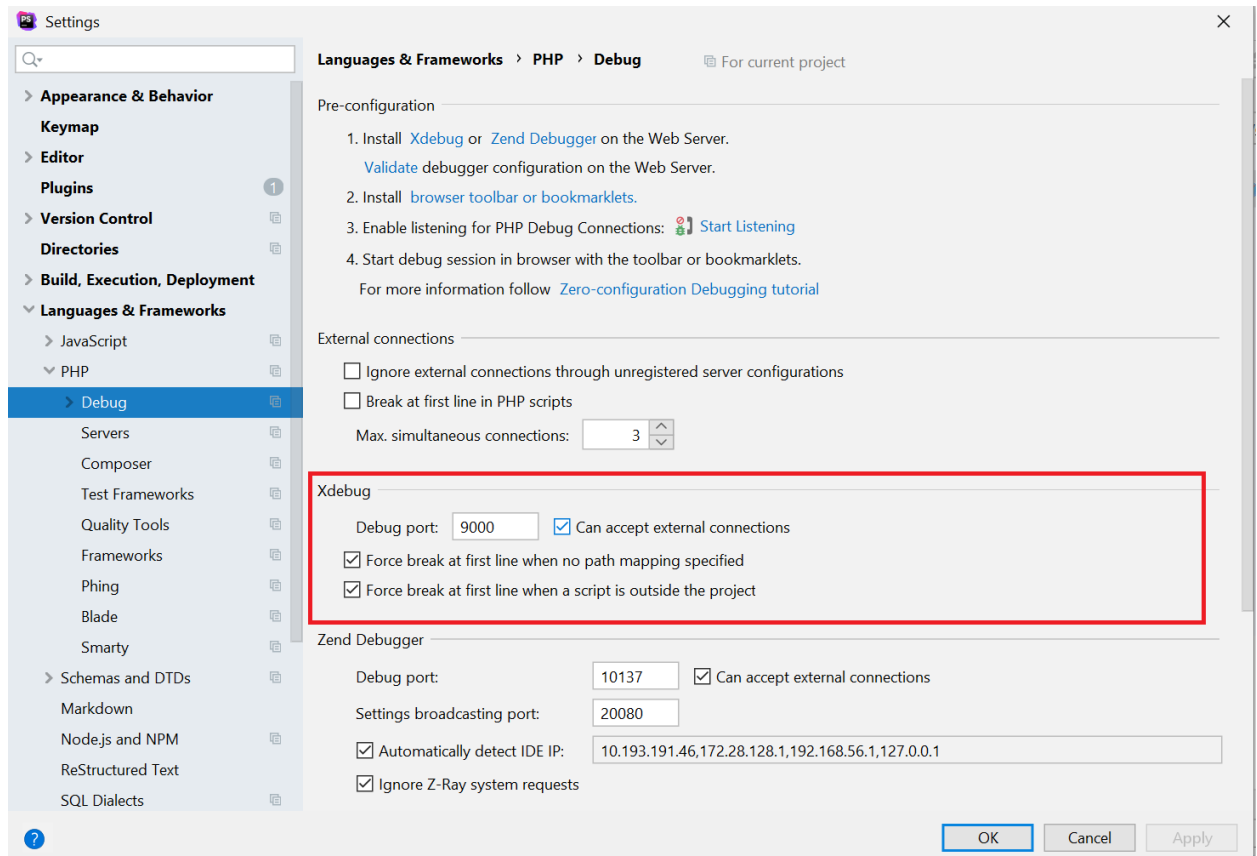
`xdebug.idekey=PHPSTORM`

5. Restart Nginx and php-fpm and check xdebug has been enabled or not using phpinfo()
  - a. Commands:
    - i. `sudo service nginx restart`
    - ii. `sudo service php-fpm restart`

#### xdebug

xdebug support	enabled
Version	2.7.2
IDE Key	PHPSTORM

6. Go to PhpStorm -> File -> Settings -> Languages & Framework -> PHP -> Debug -> Xdebug. Make sure the fields have these values.
  - a. Debug Port: 9000
  - b. Force break at the first line when no path mapping is specified is **checked**.
  - c. Force break at the first line when the script is outside the project is **checked**.

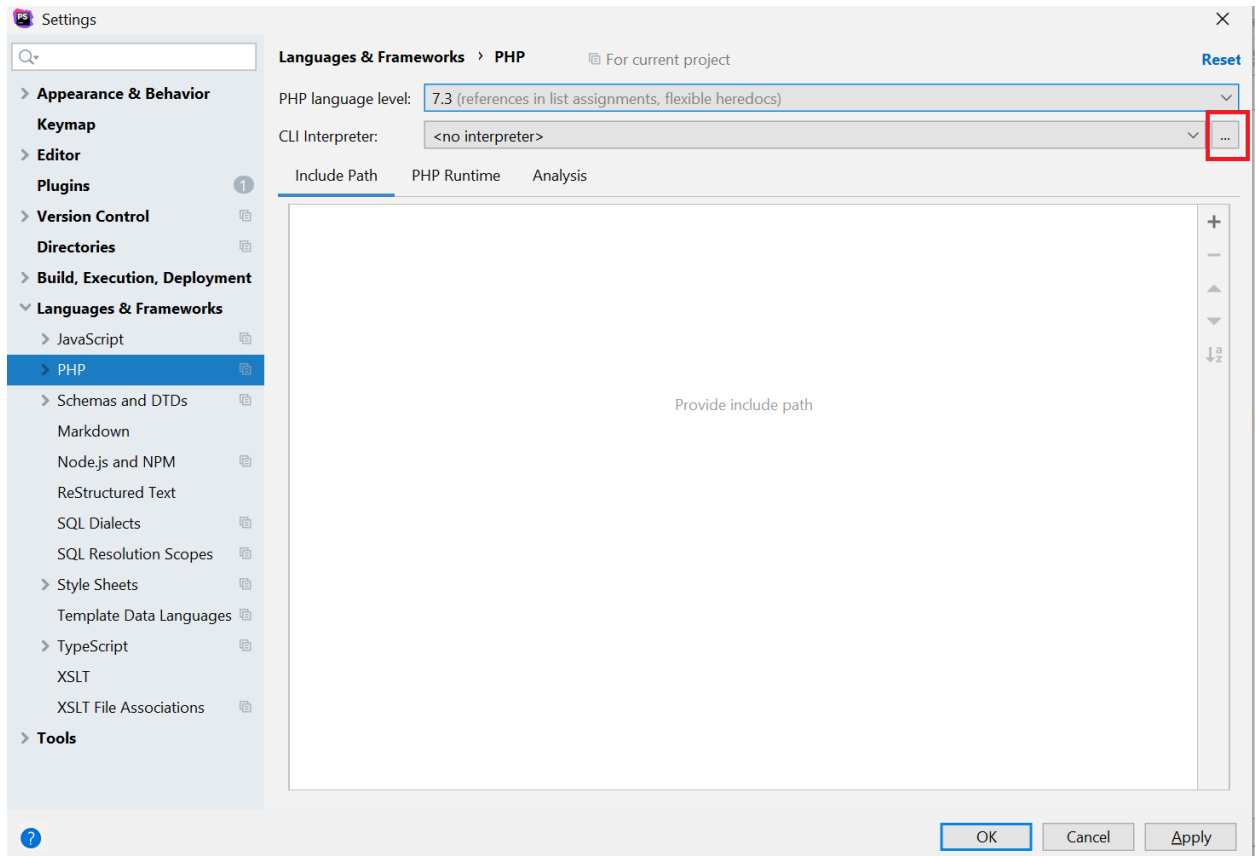


7. Click on "Apply" to apply and save the changes.

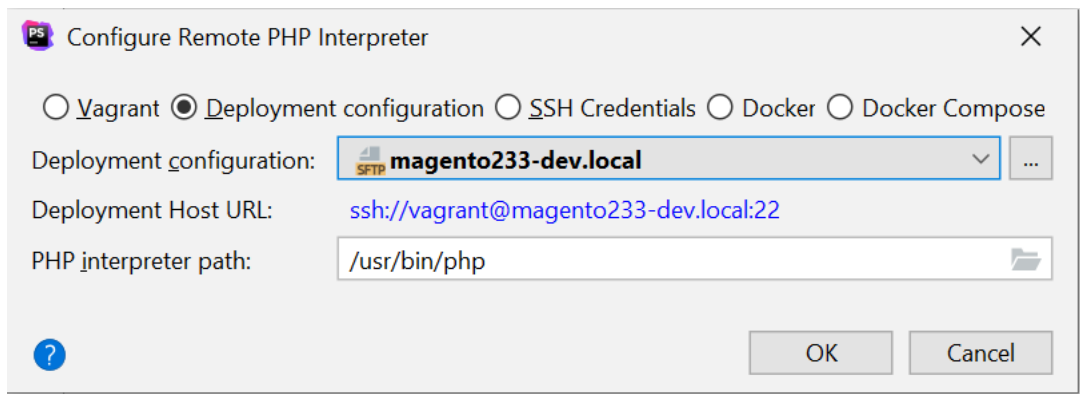
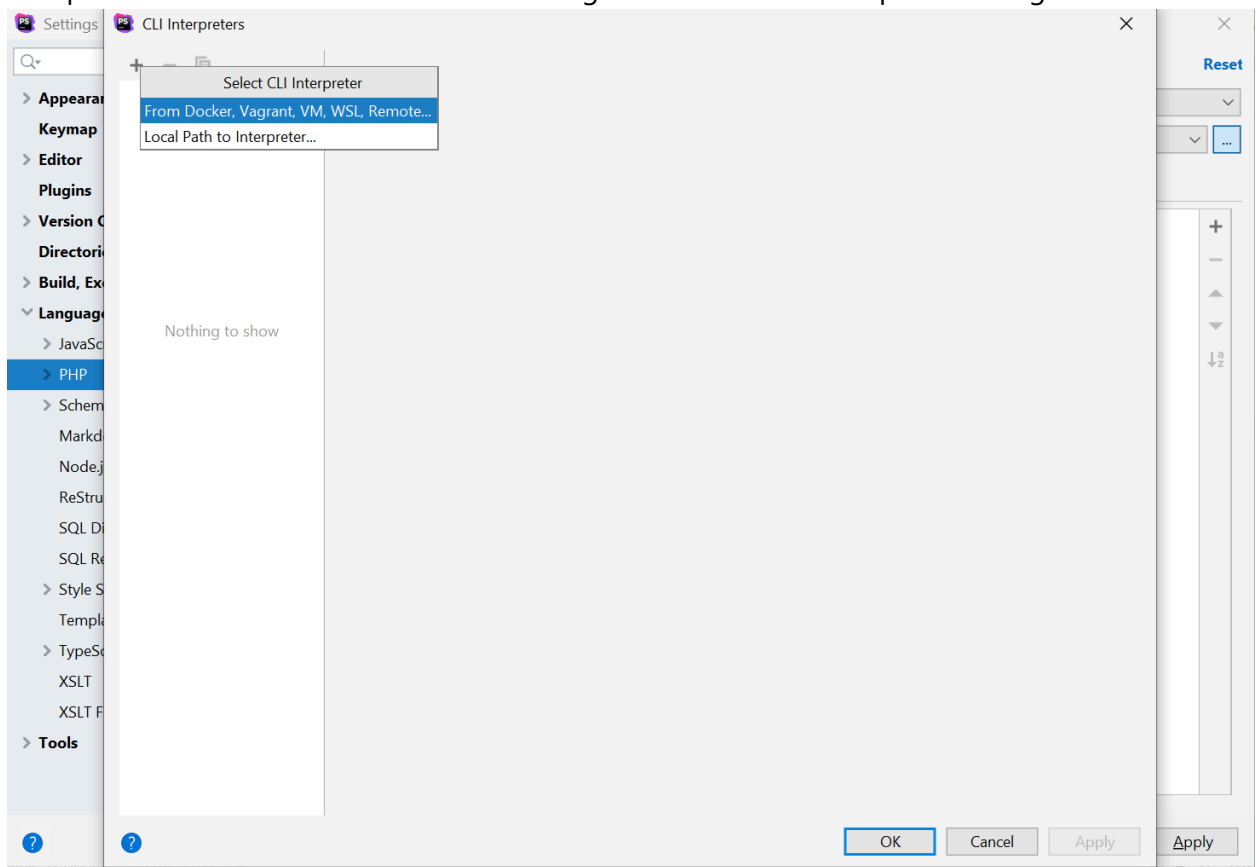


## Installing PHP Interpreter, PHPCS, PHPMD

1. Go to PHP Storm -> File -> Settings -> Languages & Framework -> PHP. Click on the ellipses in the right-hand corner in front of the CLI Interpreter field.



2. Clicking on the ellipses, will open a new window, click on the “plus” sign to add a remote interpreter. Click on the “Ok” button in Configure Remote PHP Interpreter dialog box.

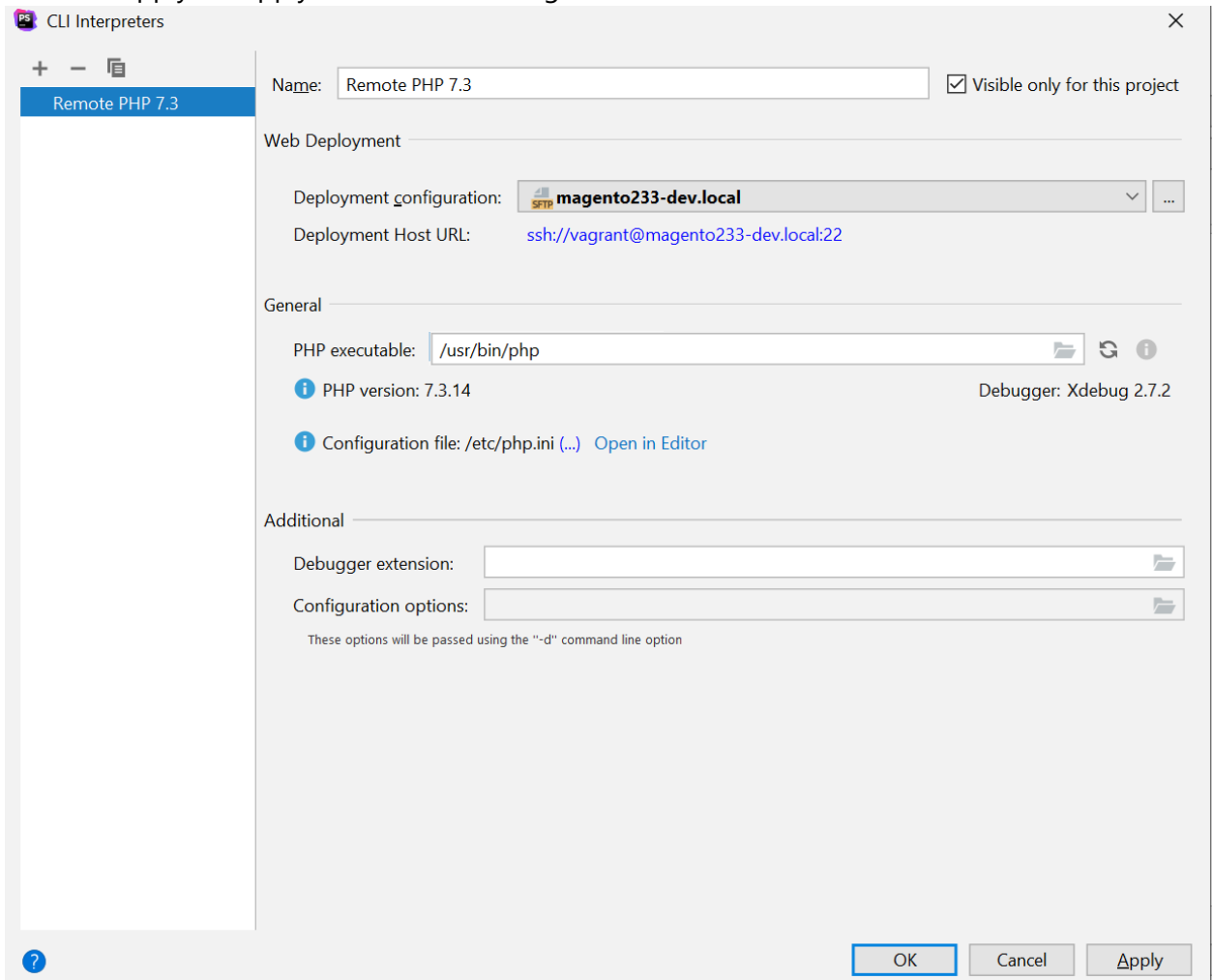


If you have a different PHP version then please type in the same. You can get details of what all php versions are installed in your system by the following approach.

- a. Run which php to know the location of the PHP executable.
- b. Type in the path. In our case /usr/bin/php and press tab 2 times to get hint. It will look something like the screenshot below.

```
> which php
/usr/bin/php
> /usr/bin/php
php      php5.6      php7.1      php7.2      phpcbf      php-config      php-config7.2  phpcs
> /usr/bin/php
```


3. After clicking "OK", another dialog box appears which shows the current configuration, click on "Apply" to apply and save the changes.



CLI Interpreters

Name: Remote PHP 7.3 ☒ Visible only for this project

Web Deployment

Deployment configuration:  magento233-dev.local

Deployment Host URL: <ssh://vagrant@magento233-dev.local:22>

General

PHP executable:

PHP version: 7.3.14 Debugger: Xdebug 2.7.2

Configuration file: </etc/php.ini> [Open in Editor](#)

Additional

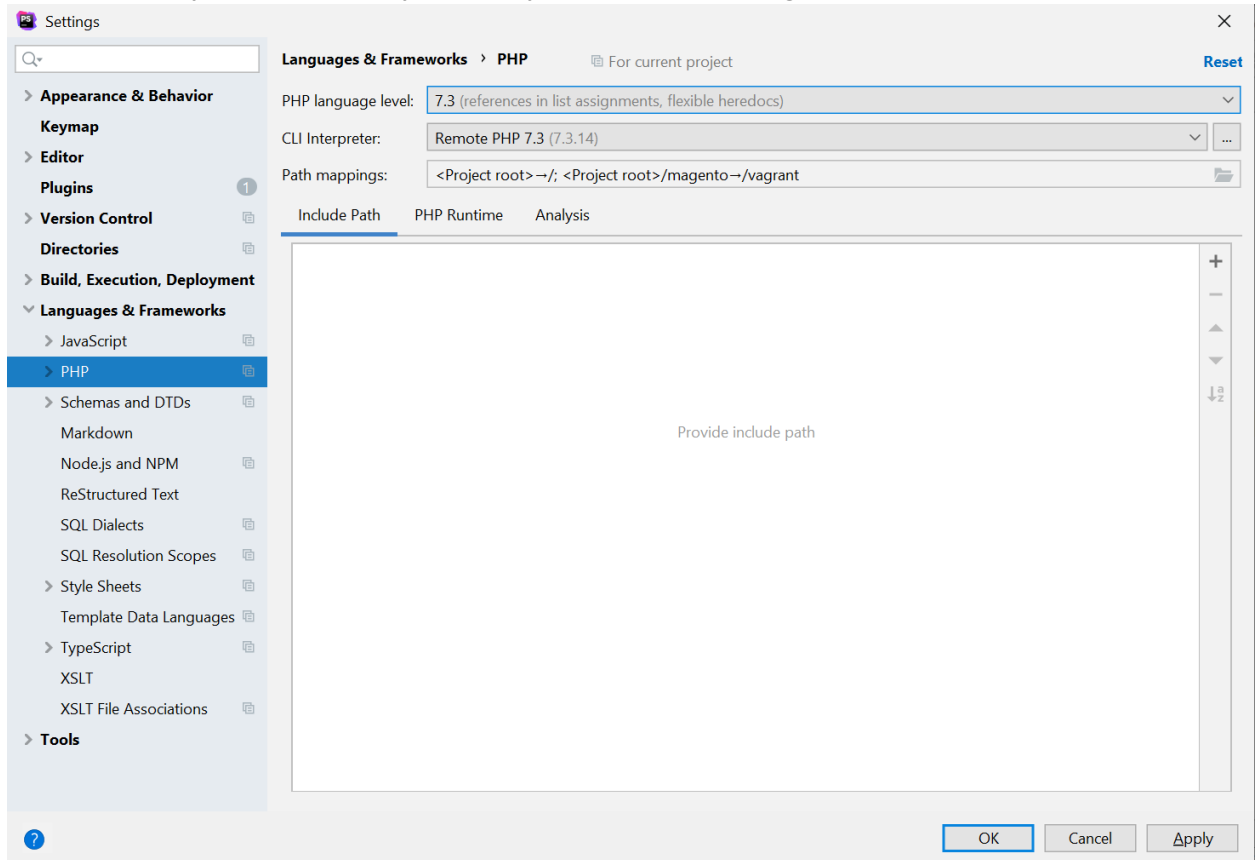
Debugger extension:

Configuration options:

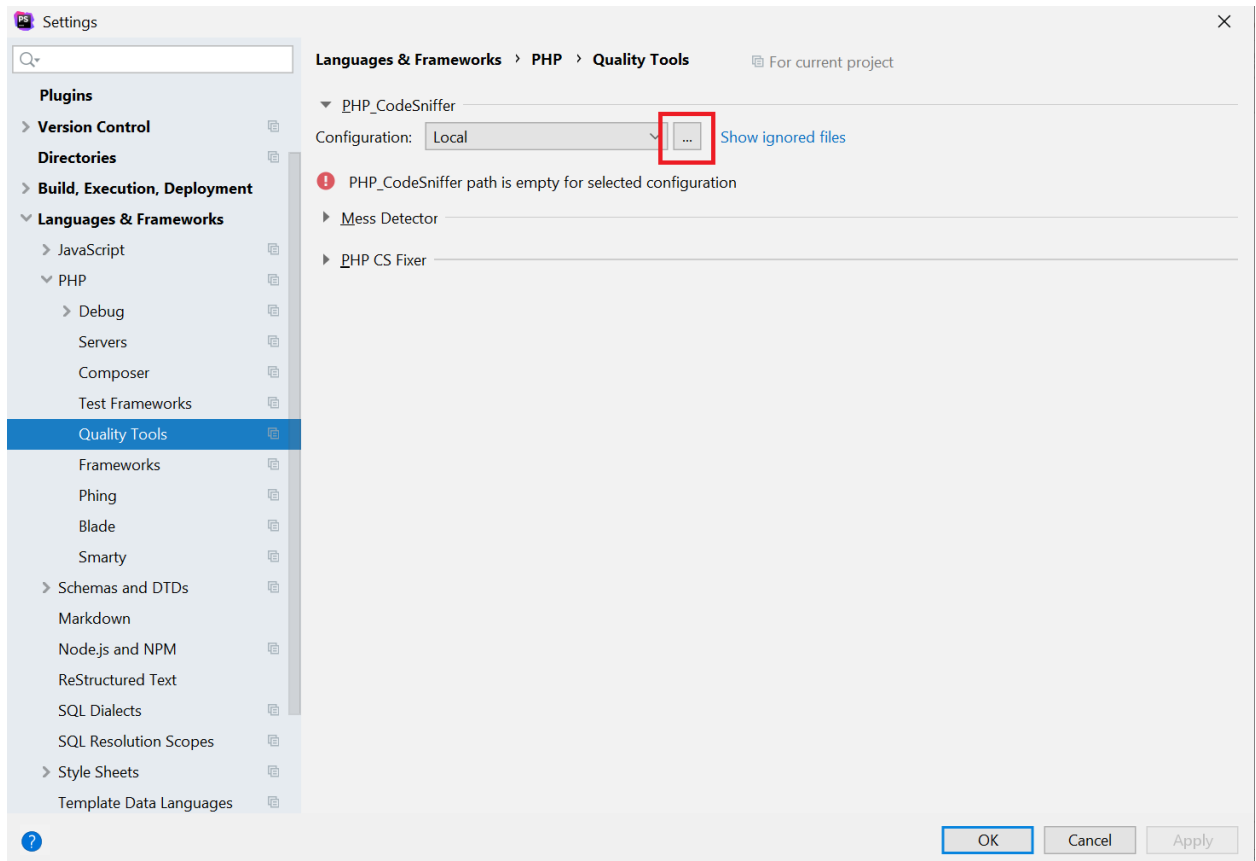
These options will be passed using the "-d" command line option

OK Cancel Apply

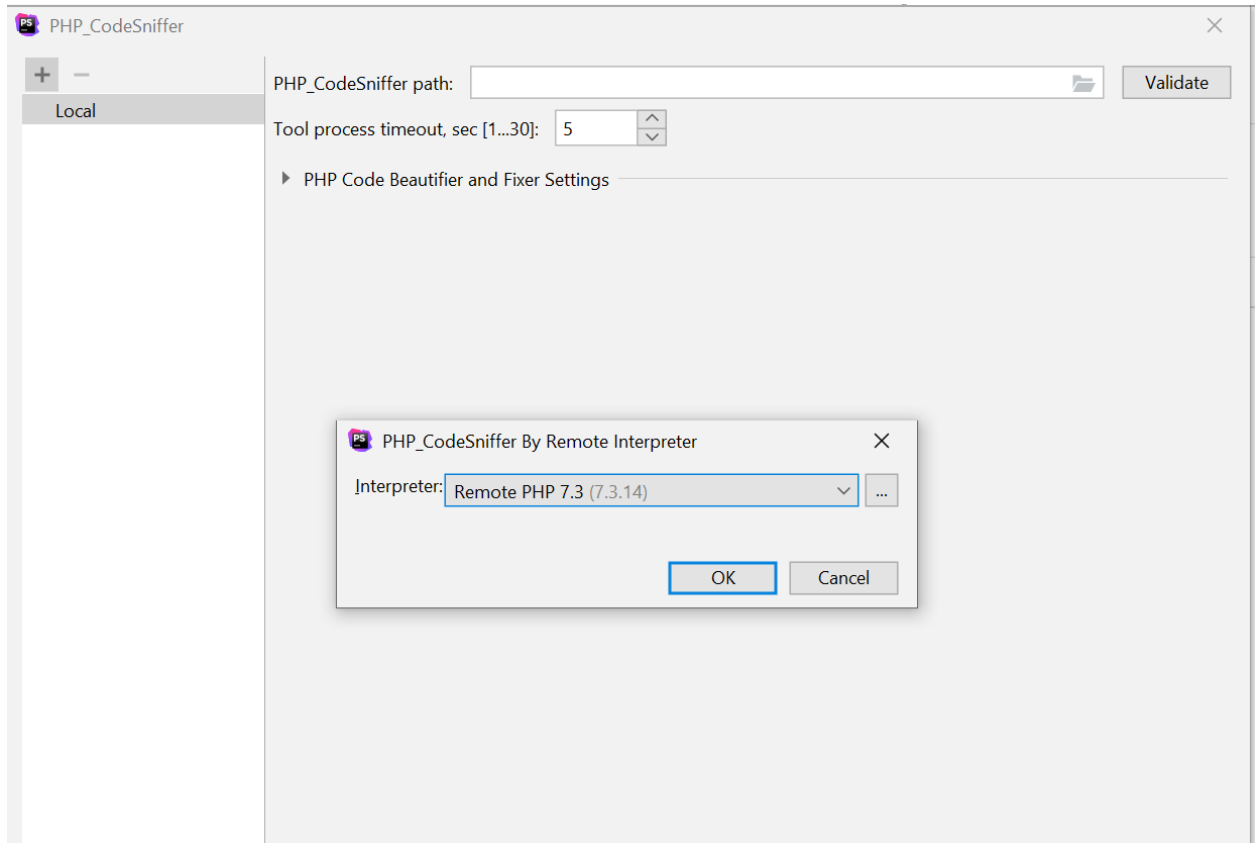
4. Once the changes are saved, the path mappings would be visible, if you have already configured the path mappings in the deployment configuration, else you will have to add it manually. Click on "Apply" to apply and save the changes.



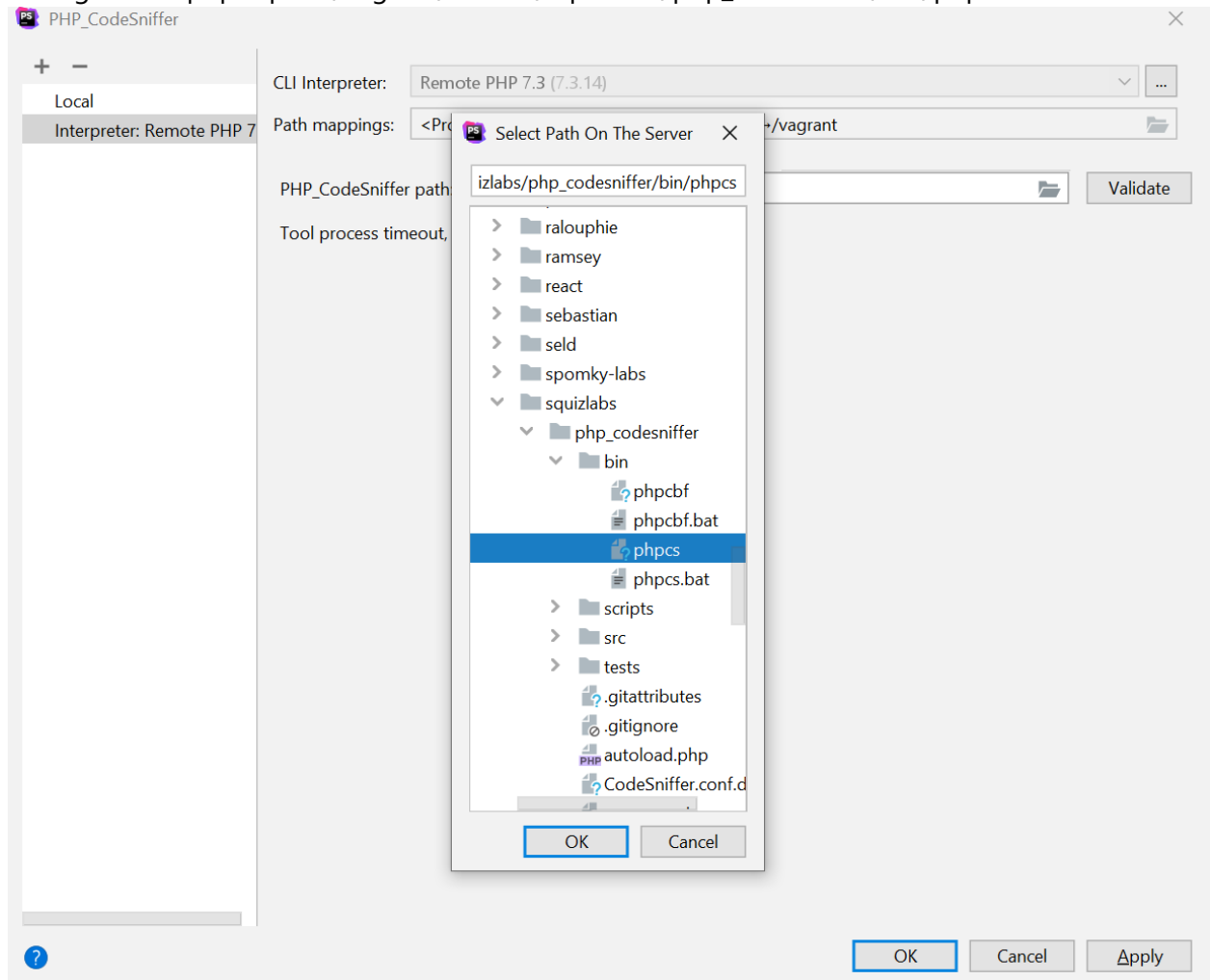
5. Now, we need to configure phpcs for the IDE. Go to PhpStorm -> File -> Settings -> Languages & Frameworks -> PHP -> Quality Tools -> PHP\_CodeSniffer. Click on the ellipses.



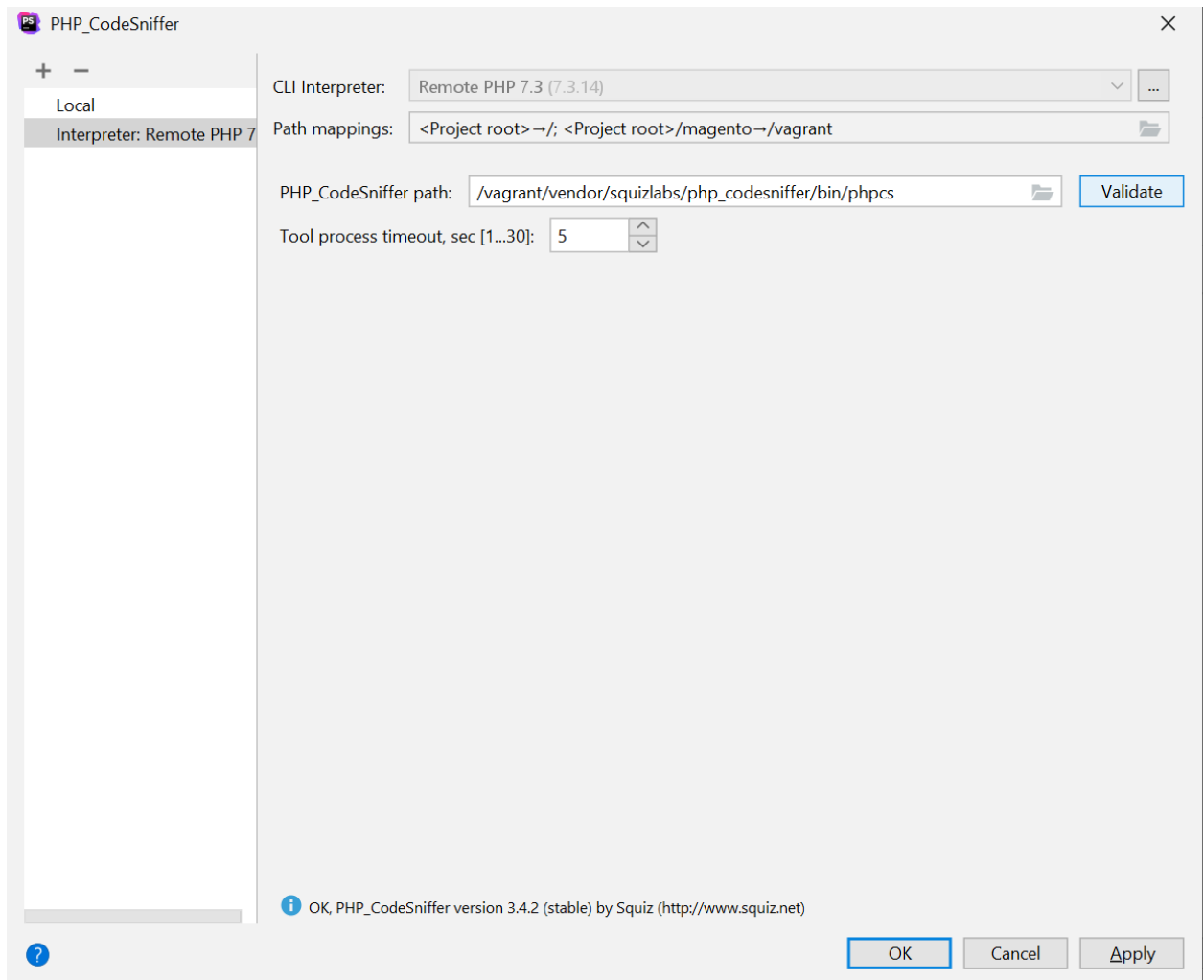
- This will open a dialog box, you need to add the remote phpcs details. Click on the “plus” sign in the left-hand top corner. From the dialog box, choose the remote interpreter that we just configured in the previous step. Click on “Ok”.



7. Configure the phpcs path `/vagrant/vendor/squizlabs/php_codesniffer/bin/phpcs`

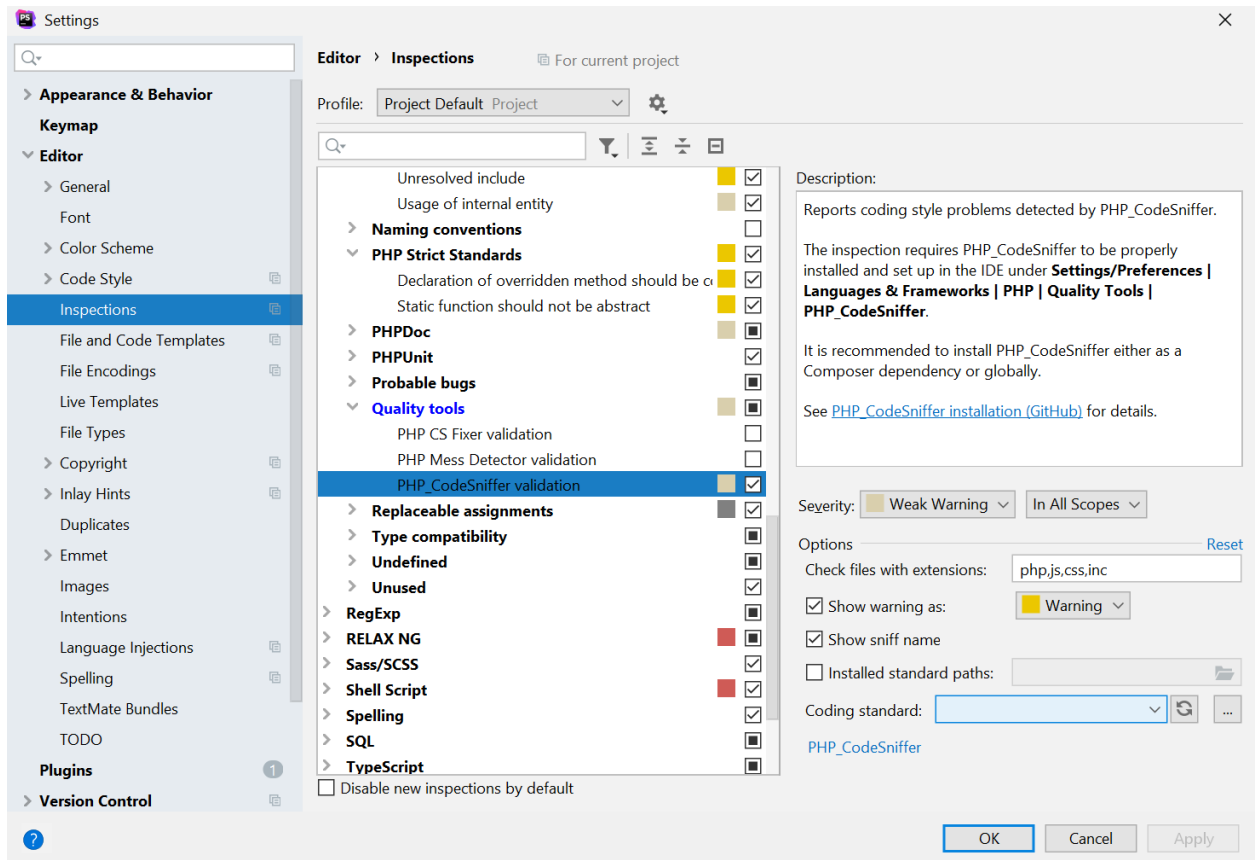


8. Click on "Validate" button to validate, if it is configured correctly. Click on "Apply" button to apply and save the changes.

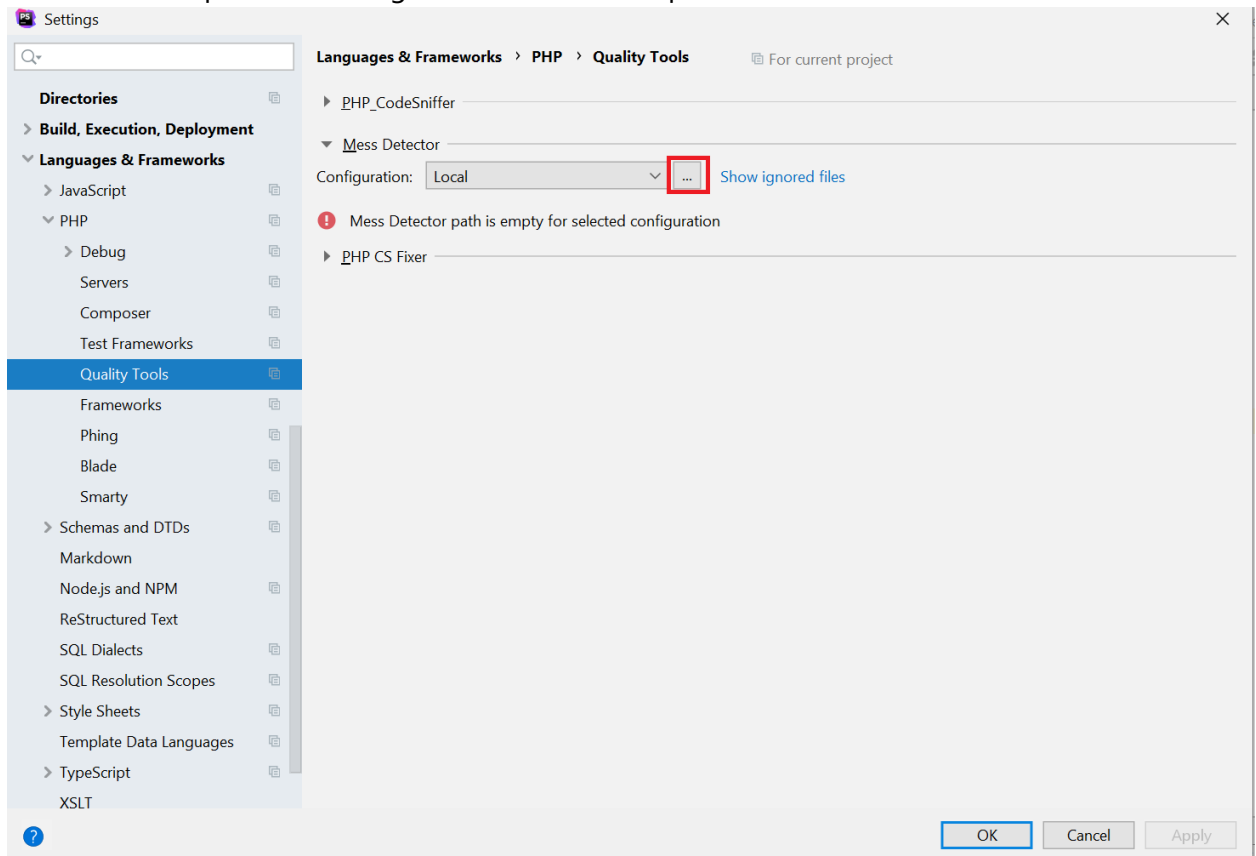




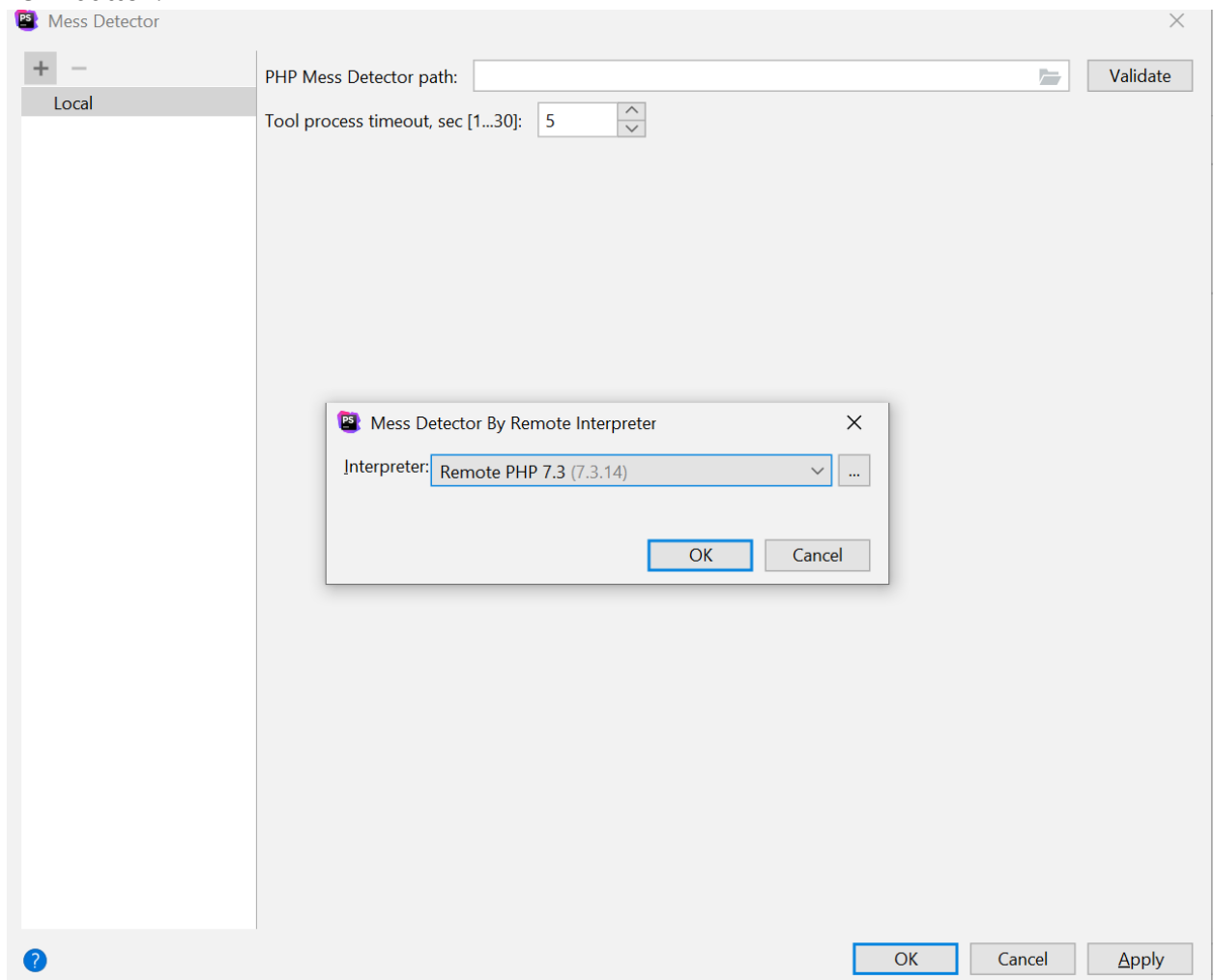
9. Enable the code sniffer validations, go to PhpStorm -> File -> Settings -> Editor -> Inspections -> PHP -> Quality Tools -> PHP\_CodeSniffer validation. Check the checkbox.



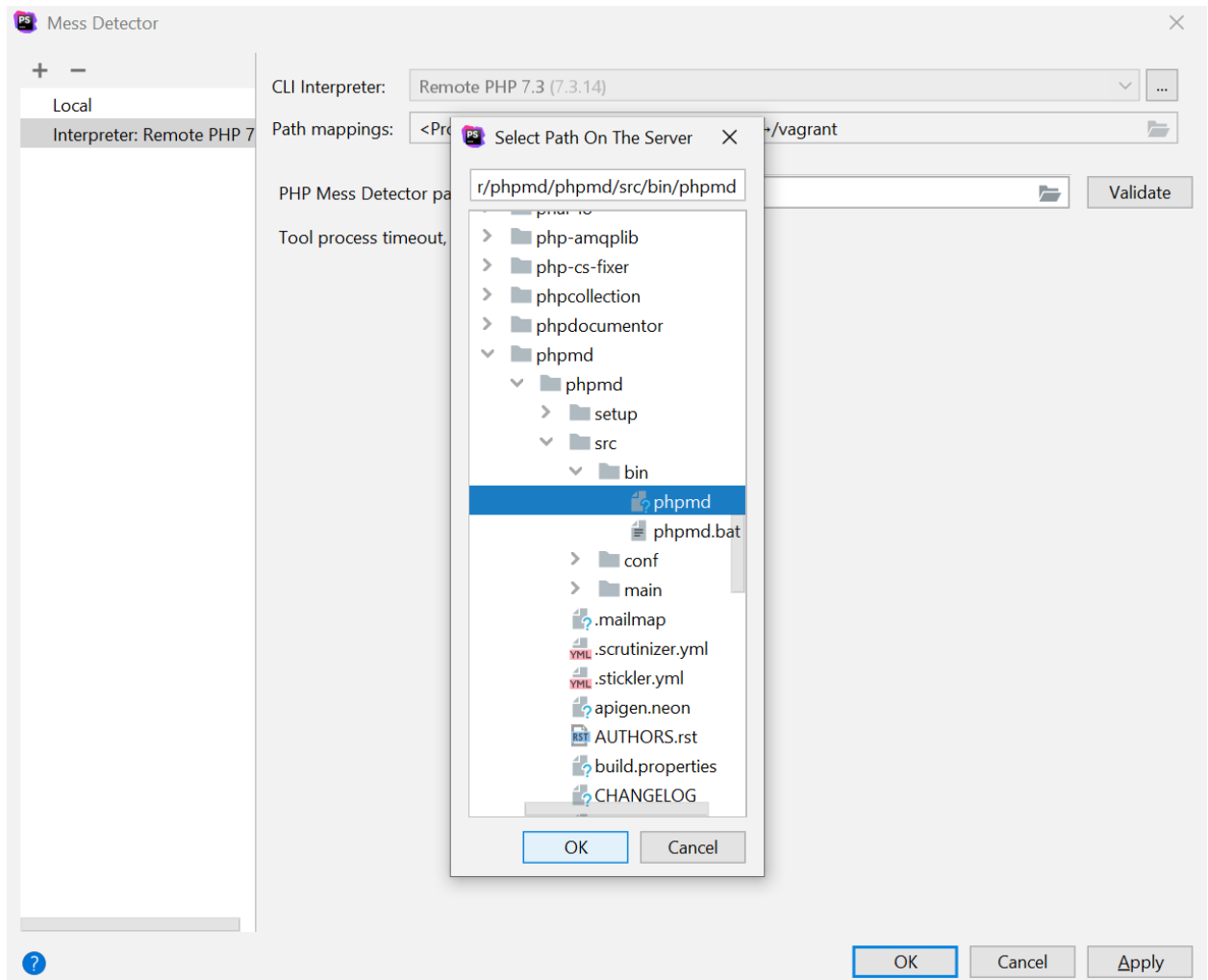
10. To enable the Mess Detector, we need to configure it first. To do so, go to PHPStorm -> File -> Settings -> Languages & Framework -> PHP -> Quality Tools -> Mess Detector. Click on the ellipses and configure the remote interpreter.



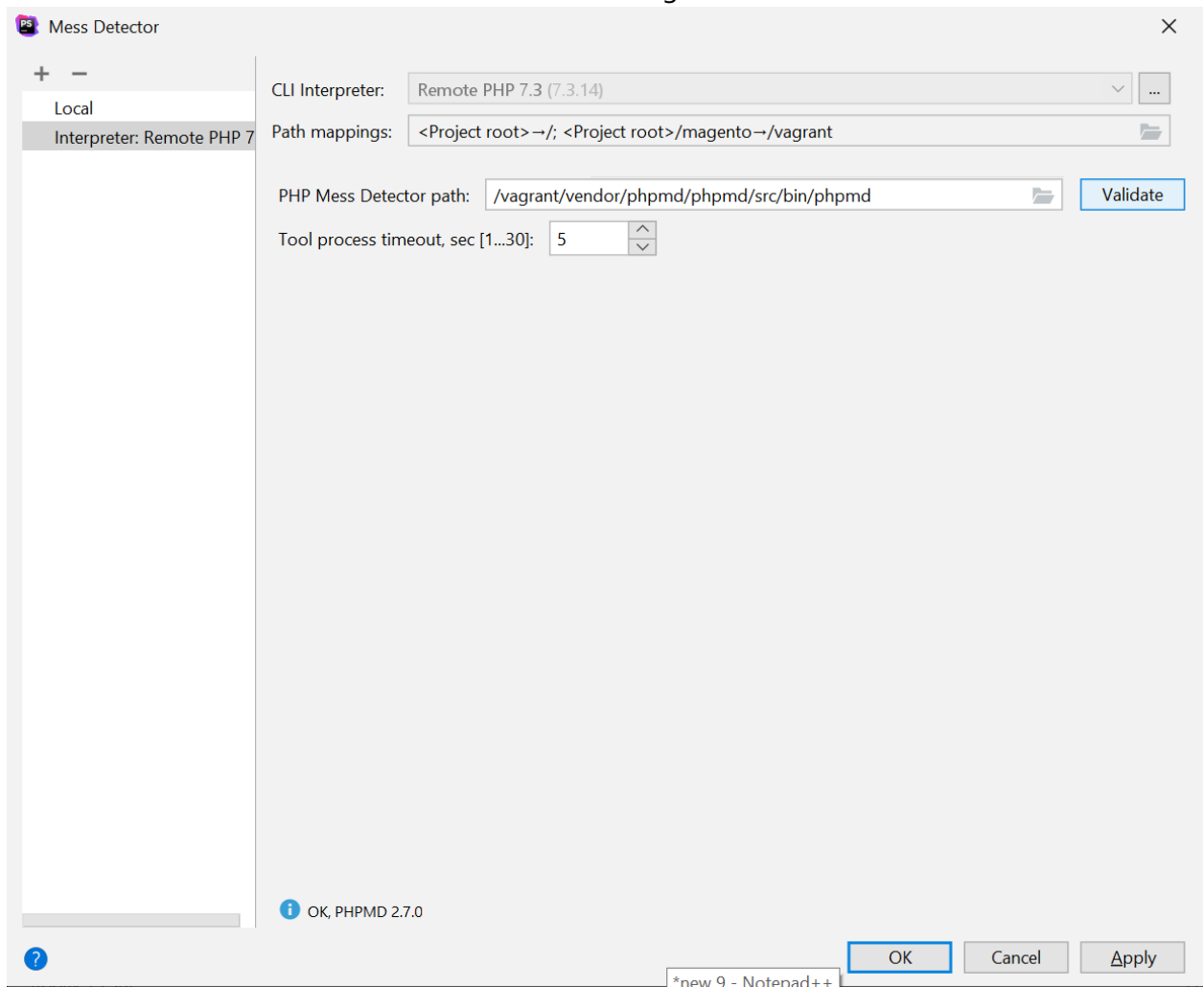
11. Click on the “plus” sign in the top left-hand corner, this will open the dialog box, Mass Detector by Remote Interpreter, select “Remote PHP 7.3” from the dropdown, click on “Ok” button.



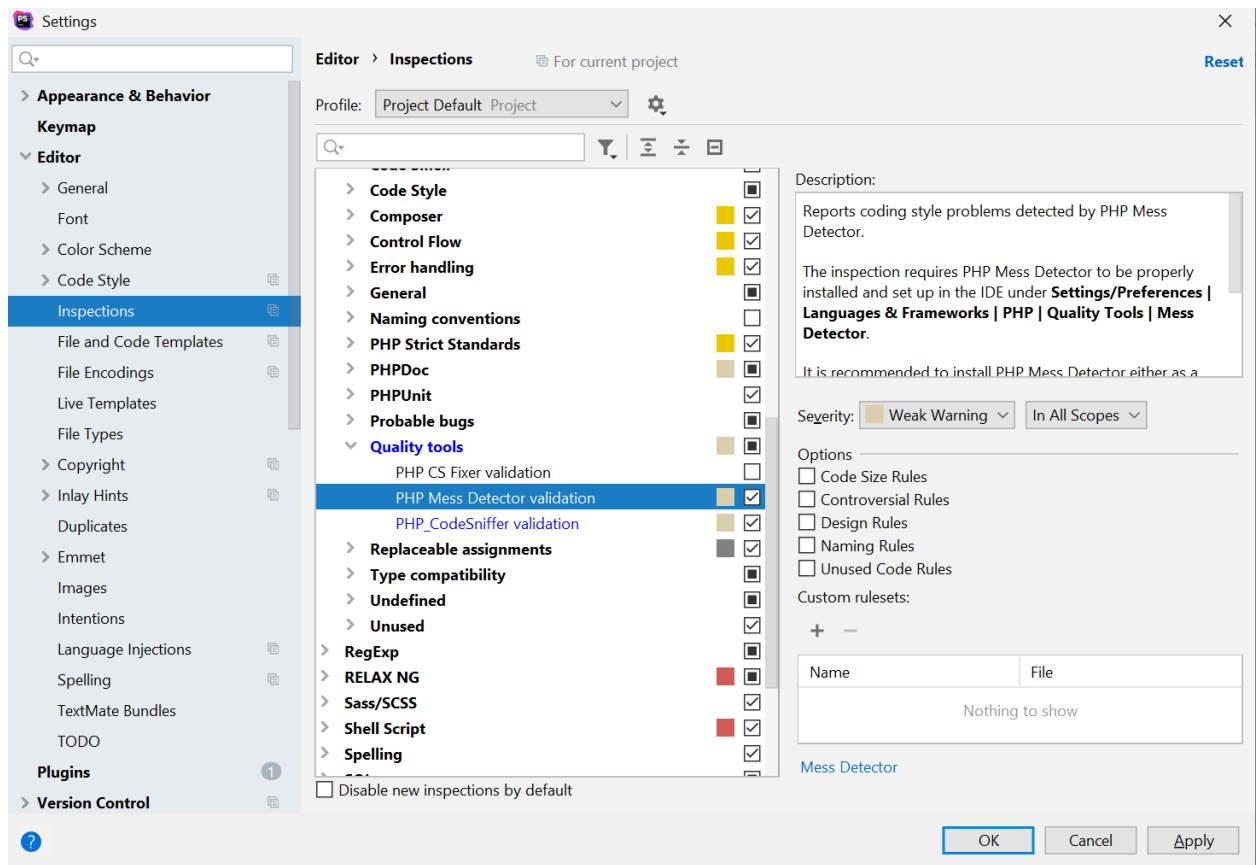
12. Enter the mess detector path: /vagrant/vendor/phpmd/phpmd/src/bin/phpmd



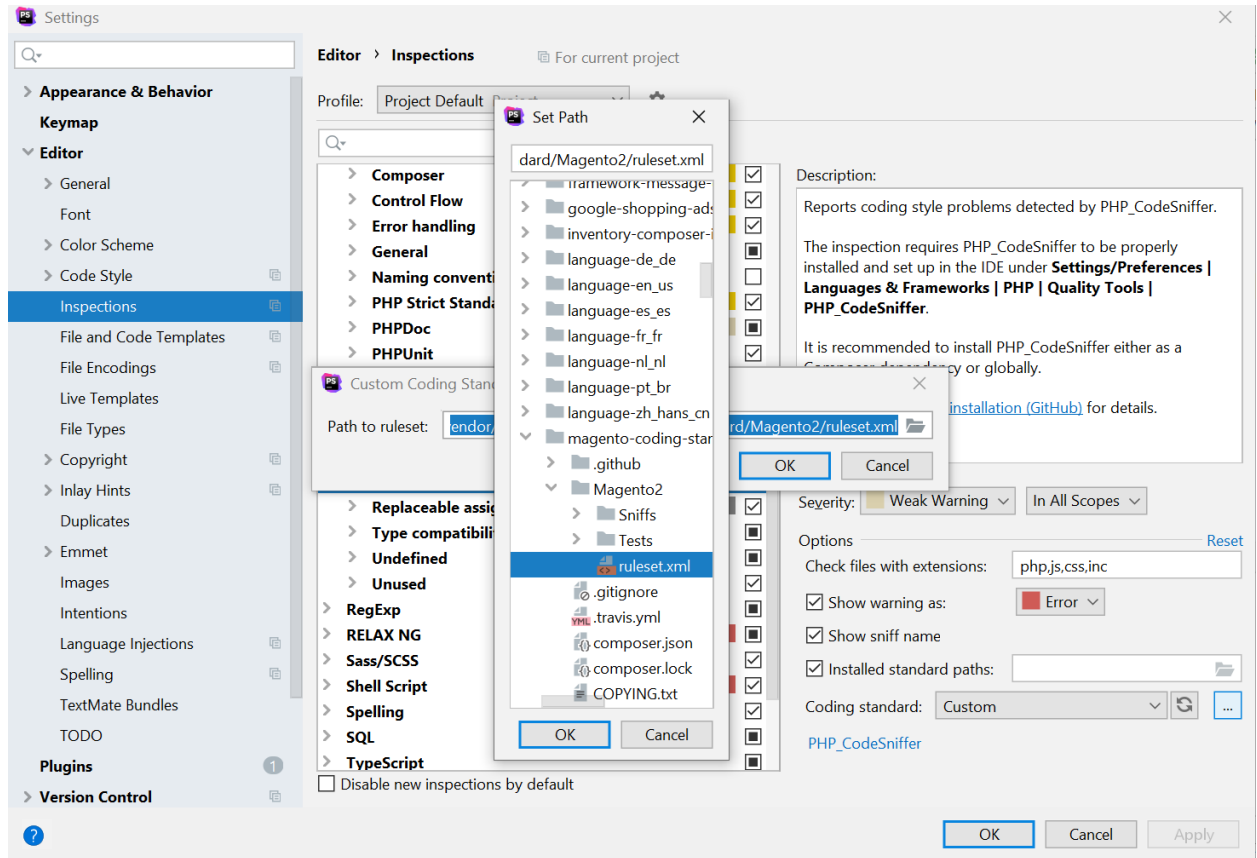
13. Click on "OK". Click on "validate" to validate the configuration.



14. Click on "Apply" to save and apply the changes. Enable the Mess detection by going to PHPStorm -> File -> Settings -> Editor -> Inspections -> PHP -> Quality Tools -> PHP Mess Detector Validation.



15. Click on "Apply" to apply and save the changes.
16. To add the Magento Standards for code sniffer validation, go to PHPStorm -> File -> Settings -> Editor -> Inspections -> PHP -> Quality Tools -> PHP\_CodeSniffer Validation.
17. In the right-hand side dropdown, select "Custom" in the Coding Standard dropdown and click on the ellipses and enter the path as follows: /vagrant/vendor/magento/magento-coding-standard/Magento2/ruleset.xml



18. Click on OK. Click on "Apply" to apply and save the changes.