

# Magento Coding Standard

The **Magento Coding Standard** is a set of rules and guidelines for writing clean, consistent, and high-quality code in Magento. These standards are designed to ensure code readability, maintainability, and compatibility with the Magento ecosystem.

Magento leverages **PHP\_CodeSniffer (PHPCS)** to enforce coding standards. The Magento Coding Standard is provided as a custom set of rules for PHP\_CodeSniffer and helps developers detect and fix issues in their code.

## Key Features

1. **Consistency:** Ensures a uniform coding style across the codebase.
2. **Error Detection:** Identifies potential bugs or anti-patterns.
3. **Compliance:** Helps developers write code that adheres to Magento's best practices.

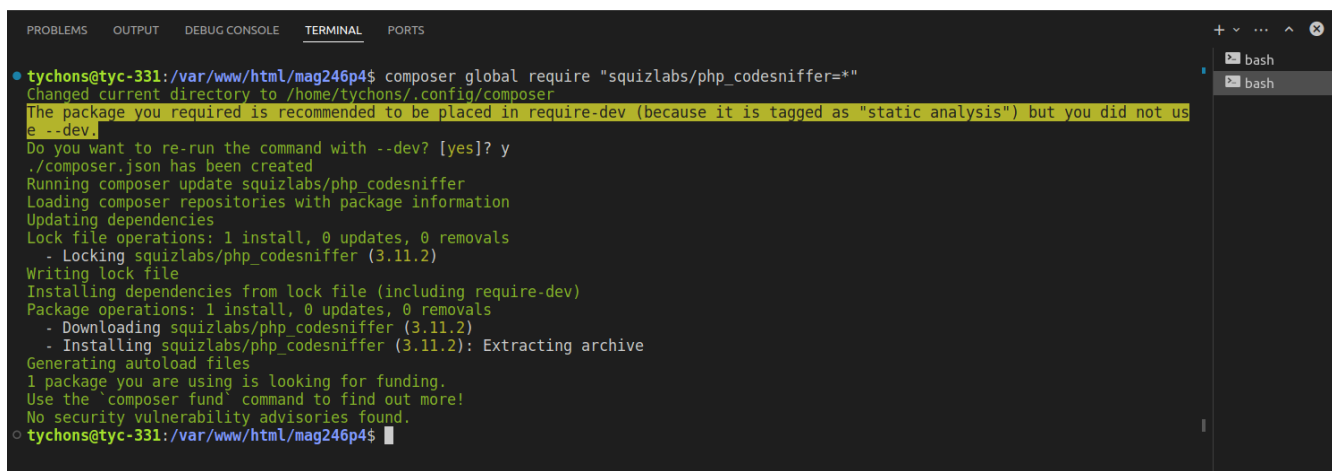
## How to Use Magento Coding Standard

### Step 1: Install PHP\_CodeSniffer

You need PHP\_CodeSniffer installed on your system. Use Composer to install it globally or locally:

Bash

**composer global require "squizlabs/php\_codesniffer=\*"**



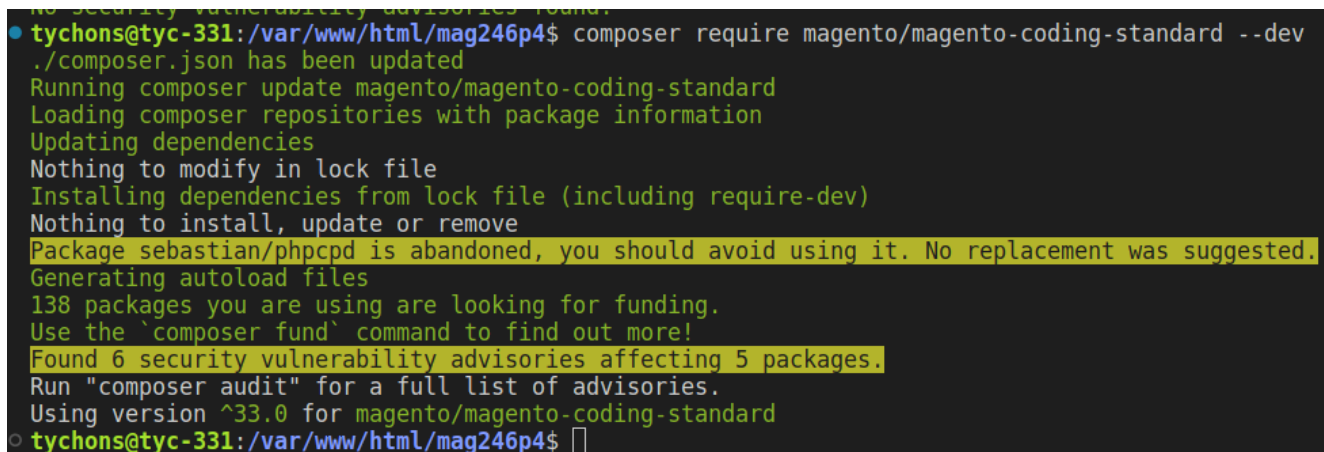
```
tychons@tyc-331:/var/www/html/mag246p4$ composer global require "squizlabs/php_codesniffer=*"
Changed current directory to /home/tychons/.config/composer
The package you required is recommended to be placed in require-dev (because it is tagged as "static analysis") but you did not use --dev.
Do you want to re-run the command with --dev? [yes]? y
./composer.json has been created
Running composer update squizlabs/php_codesniffer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking squizlabs/php_codesniffer (3.11.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading squizlabs/php_codesniffer (3.11.2)
- Installing squizlabs/php_codesniffer (3.11.2): Extracting archive
Generating autoload files
1 package you are using is looking for funding.
Use the 'composer fund' command to find out more!
No security vulnerability advisories found.
tychons@tyc-331:/var/www/html/mag246p4$
```

## Step 2: Install Magento Coding Standard

Magento provides a custom set of coding standards that you can include in your project. Use Composer to install the Magento Coding Standard package:

bash

**composer require magento/magento-coding-standard --dev**



```
no security vulnerability advisories found.
• tychons@tyc-331:/var/www/html/mag246p4$ composer require magento/magento-coding-standard --dev
./composer.json has been updated
Running composer update magento/magento-coding-standard
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Package sebastian/phpcpd is abandoned, you should avoid using it. No replacement was suggested.
Generating autoload files
138 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
Found 6 security vulnerability advisories affecting 5 packages.
Run "composer audit" for a full list of advisories.
Using version ^3.0 for magento/magento-coding-standard
• tychons@tyc-331:/var/www/html/mag246p4$
```

## Step 3: Configure PHP\_CodeSniffer

Specify Magento's coding standard in PHP\_CodeSniffer:

bash

**phpcs --config-set installed\_paths vendor/magento/magento-coding-standard**

This command sets up Magento's coding standard rules for PHP\_CodeSniffer.

## Step 4: Validate Code Against Magento Standards

Run the following command to analyze your code:

bash

**phpcs --standard=Magento2 /path/to/your/code**

- Replace **/path/to/your/code** with the directory or file you want to analyze.
- It will display warnings and errors that violate Magento's standards.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
tychons@tyc-331:/var/www/html/mag246p4$ vendor/bin/phpcs --standard=Magento2 app/code/Ragu

FILE: /var/www/html/mag246p4/app/code/Ragu/CustomOrderID/Ui/Component/Listing/Column/RandomOrderId.php
-----
FOUND 0 ERRORS AND 3 WARNINGS AFFECTING 3 LINES
-----
 11 | WARNING | Missing PHP DocBlock for class property.
 13 | WARNING | Comment block is missing
 24 | WARNING | Comment block is missing
-----

FILE: /var/www/html/mag246p4/app/code/Ragu/CustomOrderID/Model/ResourceModel/Order/Grid/Collection.php
-----
FOUND 0 ERRORS AND 6 WARNINGS AFFECTING 5 LINES
-----
 15 | WARNING | [ ] Missing PHP DocBlock for class property.
 17 | WARNING | [ ] Comment block is missing
 17 | WARNING | [ ] Possible useless method overriding detected
 25 | WARNING | [x] The closing parenthesis and the opening brace of a multi-line function declaration must be on the
    |         |     same line
 29 | WARNING | [ ] Comment block is missing
 35 | WARNING | [x] Expected 1 newline at end of file; 0 found
-----
PHPCBF CAN FIX THE 2 MARKED SNIFF VIOLATIONS AUTOMATICALLY
-----
```

### Step 5: Fix Issues (Optional)

You can use the phpcbf command to **automatically fix** some coding standard issues:

bash

phpcbf --standard=Magento2 /path/to/your/code