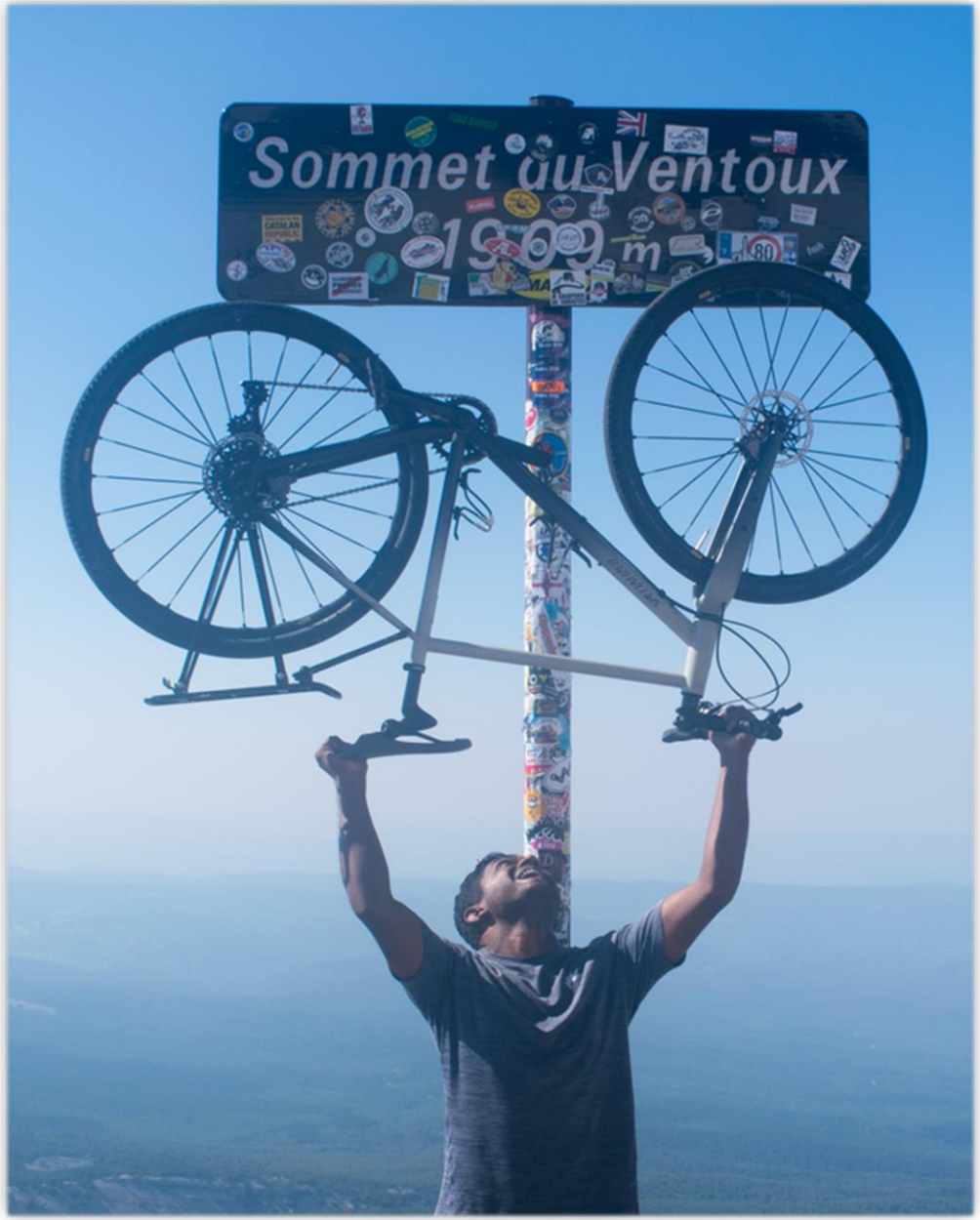
From Zero to

Tested
Automated
Documented

Open Source Packages

Ayesh Karunaratne | https://ayesh.me/talk/PHP-Open-Source

Hallo!



Ayesh Karunaratne

Security Researcher, Freelance Software Developer

Kandy, Sri Lanka - Everywhere

https://ayesh.me

Ayesh

@Ayeshlive

Ayesh

https://ayesh.me/talk/PHP-Open-Source

and you?

Pull Requests

Code Styling

GPL Licensing

Functional Tests

Code Review

composer

Issues

PHPDoc

Interfaces

Continuous Integration

git

Autoloading

Test Mockups

Test Mockups

Copyright

Unit Tests

KISS

Linting

Dev-Depenencies

Continuous Delivery

Semantic Versioning

IoC Containers

Test Coverage

CVE

Merge Branches

Exception Handling

Documentation

SOLID

Commit Signing

PHP Standard Recommendations

Backwards Compatibility

SVN

Dependencies

Release Notes

1. Why Open Source?
   Advantages to the community and ourselves, and how easy it is to get started.

2. Modern PHP Today
   Modern PHP practices, tooling, and interoperability.

3. Development Environment Setup
   Prepare the development environment with version controlling, composer, gpg, etc.

4. Our First Project
   Let's create a small package with Git and Composer, in small and easy steps.

5. Package Dependencies
   How to specify and use dependencies via Composer. Declare PHP environment requirements, conflicts, etc.

6. Testing
   Using PHPUnit, let's test our small PHP package. Discover code testability factors, assertions, etc.

7. Documentation
   We write the documentation for our PHP package. Use of PHPDoc, and a README file, code examples, etc

8. Automation
   Continuous Integration / Continuous Delivery for east and automated package testing and releasing.

9. Releases
   Releasing the initial and subsequent versions of the packages. Semantic Versioning.

10. Maintenance
    Issue queues, Pull Requests, and how to collaborate with other contributors.

11. Being Human
    How to be a friendly and cheerful contributor as well as a leader.

12. Community and Collaboration
    Collaborate on a big project, how and why the community is the best thing behind any Open Source Project.

Why Open Source?

Because

Open Source Is Not Intimidating

Open Source Is Not Intimidating

Most of us start from a simple contribution

We all make mistakes

We all learn new things in the process

Because

The Community is awesome!

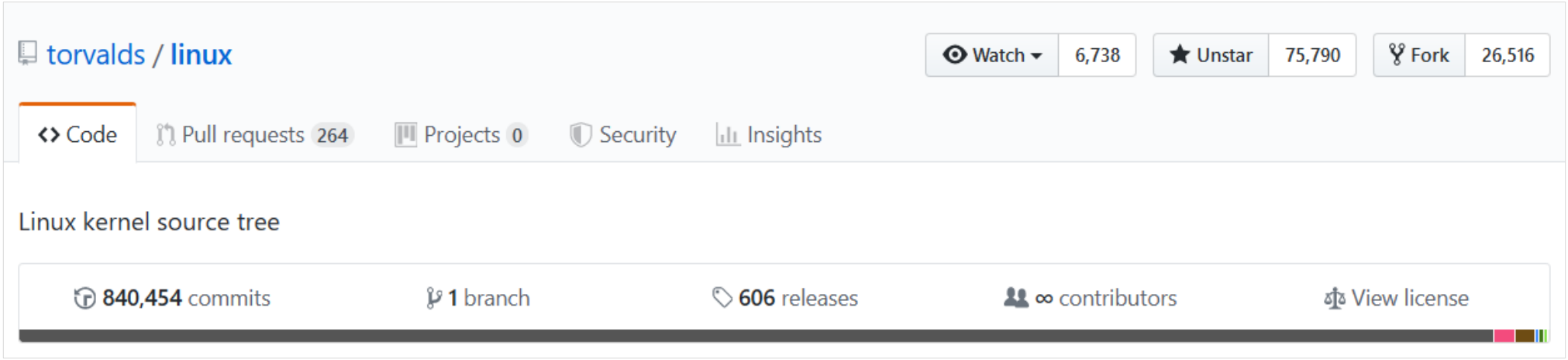

HALLE B


DrupalCon
DrupalCon

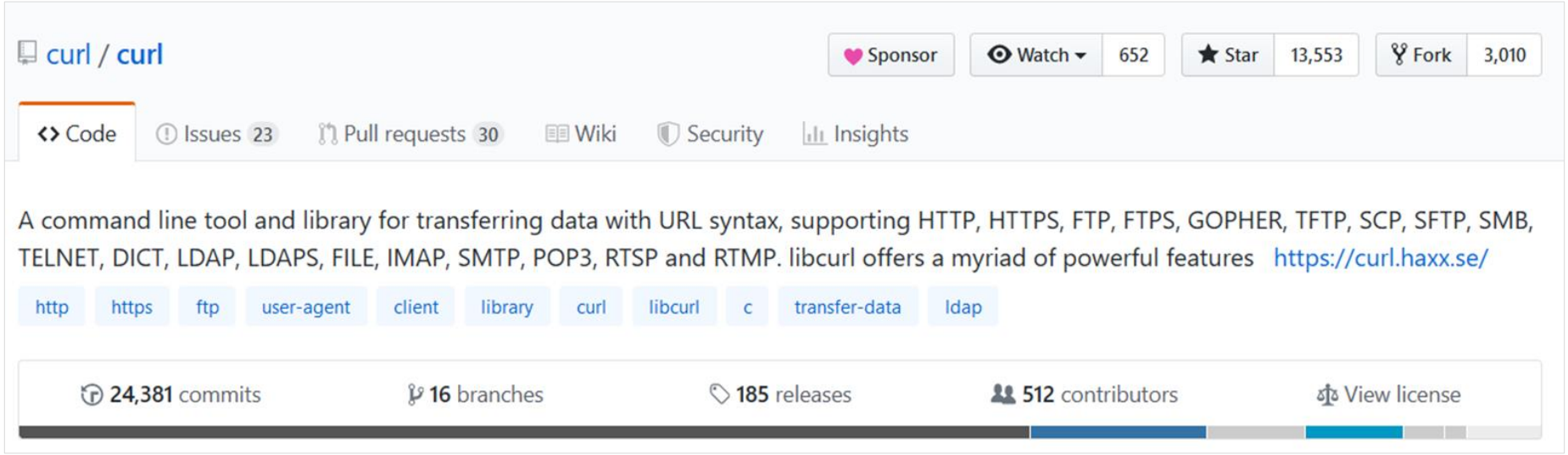
Because

We Should Give Back

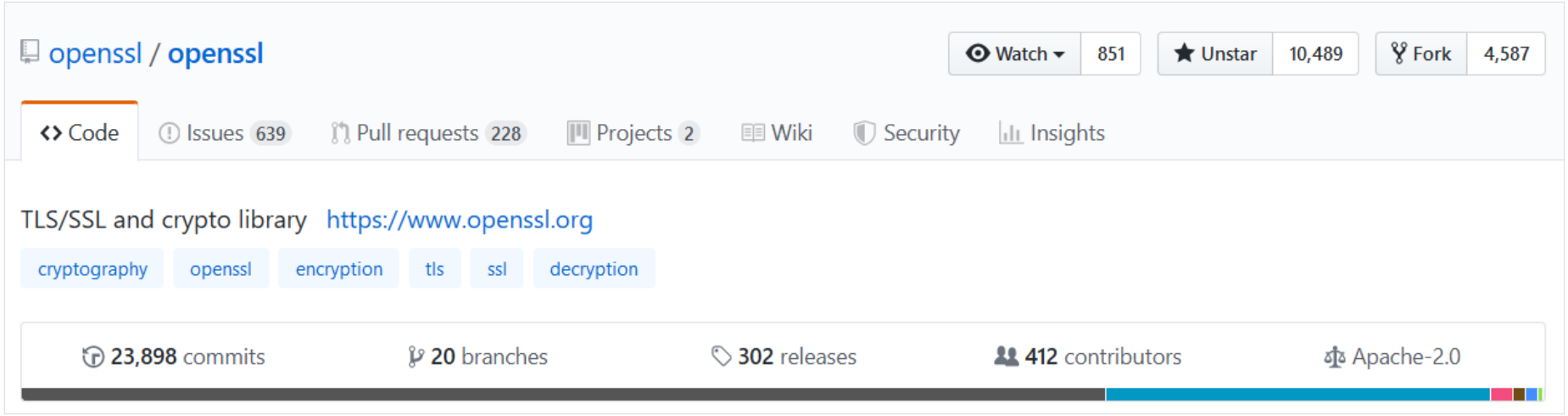
We Should Give Back



We Should Give Back



We Should Give Back



We Should Give Back

Because

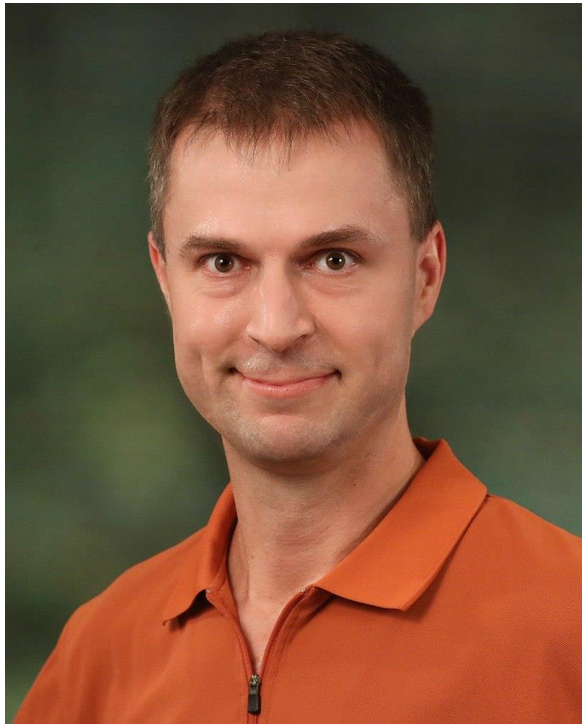
Opens Doors to New Opportunities

Opens Doors to New Opportunities

curl://

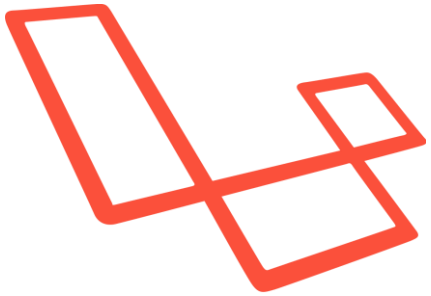
Daniel Stenberg

Opens Doors to New Opportunities



Polhems Prize

Opens Doors to New Opportunities



Because

Self-Improvement

Because

It’s Fun!

PHP Today

PHP Today

Use Object Oriented Programming

PHP Today

Enforce Strict Types

Strict Types

```
function add($num_1, $num_2) {
 return $num_1 + $num_2;
}
```

Strict Types

```php
function add(int $num_1, int $num_2): int {
 return $num_1 + $num_2;
}
```

PHP Today

Use Exceptions

Use Exceptions

```php
if (empty($user)) {
 return false;
}
```

```php
if (empty($user)) {
 throw new UserNotFoundException();
}
```

PHP Today

Autoloading

Autoloading

```php
include 'User.class.php';
include 'Database.class.php';
$user = new User();
```

```php
spl_autoload_register(function (){});
$user = new User();
```

PHP Today

Composer

Composer

PHP Dependency manager

Manages, downloads, and autoloads PHP packages

```
composer install phpunit/phpunit
composer install your/your-package
composer install your/other-package:1.2
composer remove  your/useless-package
```

PHP Today

Proudly Invented Elsewhere

PHP Today

Interoperable

Interoperable

```php
interface Cache {
 public function set(string $key, $value): void;
 public function get(string $key);
}

$cache->set('Foo', 'Bar');
$cache->get('Foo');
```

```php
class FileCache implements Cache {}

class RedisCache implements Cache {}

class NullCache implements Cache {}
```

Interoperable

PHP Standard Recommendations

Interoperable

PSR-0 Autoloading Standard

PSR-1 Basic Coding Standard

PSR-2 Coding Style Guide

PSR-3 Logger Interface

PSR-4 Autoloading Standard

PSR-6 Caching Interface

PSR-7 HTTP Message Interface

PSR-11 Container Interface

PSR-13 Hypermedia Links

PSR-14 Event Dispatcher

PSR-15 HTTP Handlers

PSR-16 Simple Cache

PSR-17 HTTP Factories

PSR-18 HTTP Client

Dev Environment Setup


php

https://windows.php.net/download/
https://laragon.org

https://brew.sh/

```
brew install php@7.3
```

```
sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt-get install php7.3 php7.3-cli …
```

Is It Working?


```
php -v  7.3
```

```
php -m    curl
          mbstring
          bcmath
          zip
```


COMPOSER

https://getcomposer.org/download/


git

https://git-scm.com/download/win

https://git-scm.com/download/mac

```
sudo apt install git
```

Is It Working?


```
git --version       2.11.0
```

Introduce Yourself


```
git config --global user.name 'Your Name Here'
git config --global user.email 'you@you.com'
```


PS

Our First Project

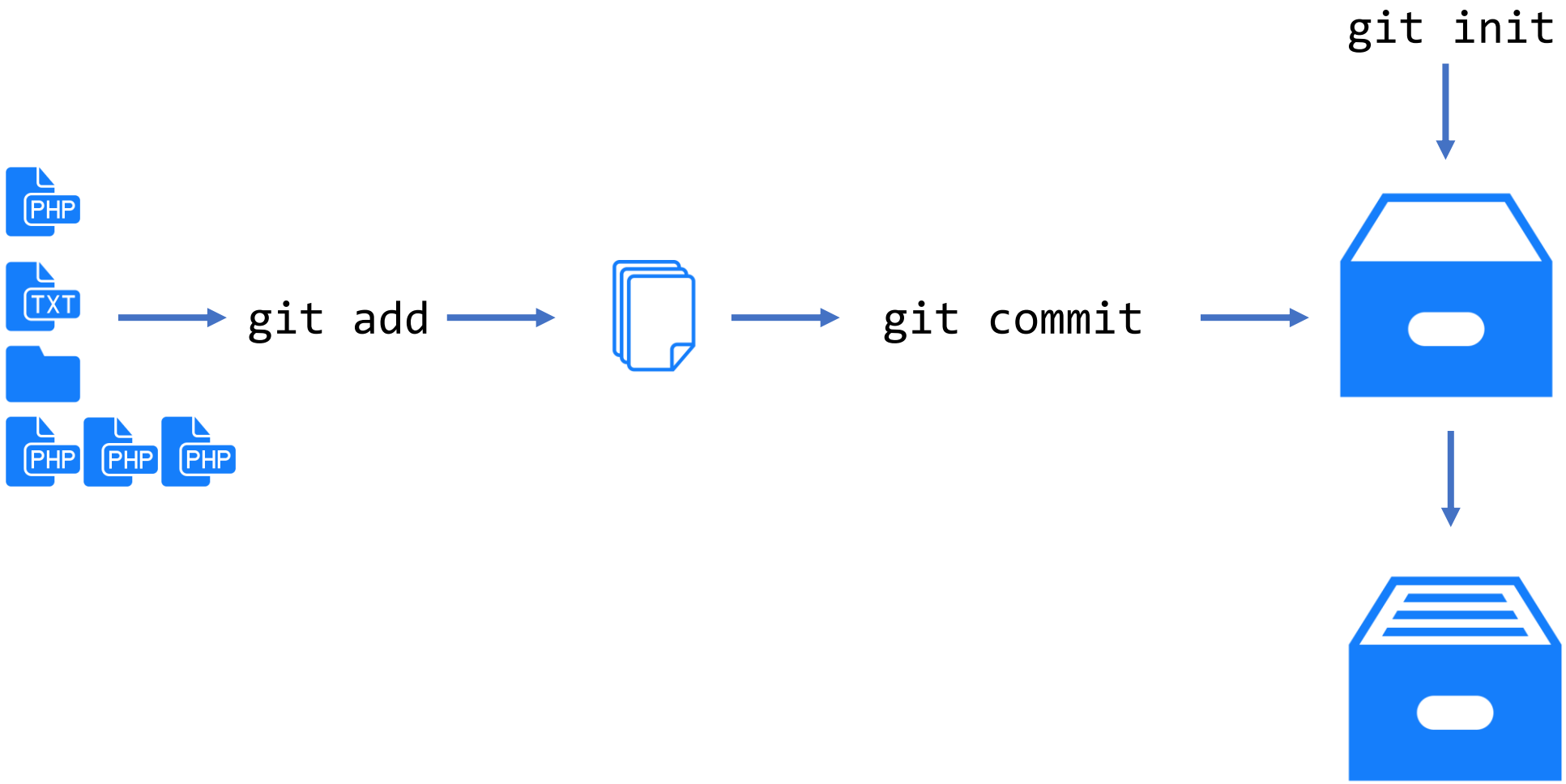

```
composer init
```

`composer init`

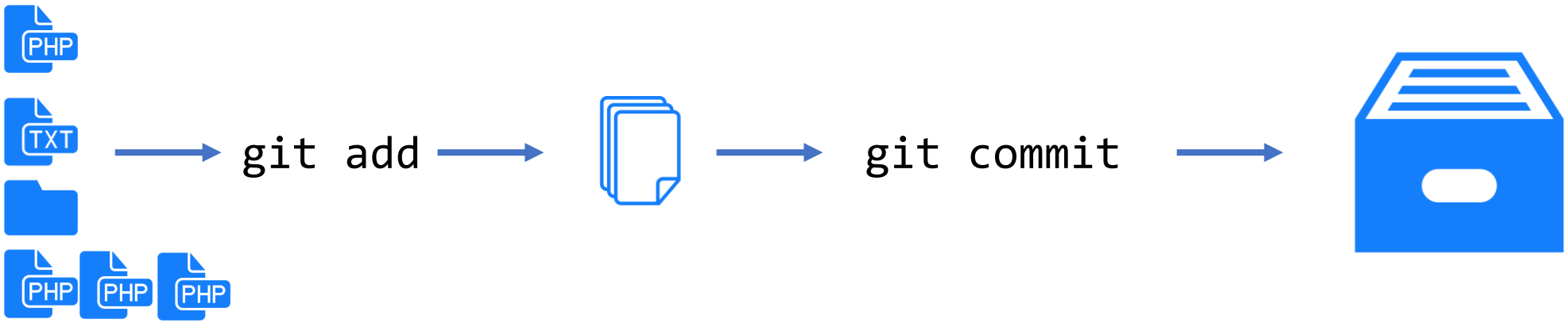
Package Name	yourname/package-name
Description	A Short introduction to your package
Author	Your name and email, preferably same as Git configuration
Minimum Stability	`stable` | `rc` | `beta` | `alpha` | `dev`
License	`proprietary` | `GPLv2` | `GPLv3` | `MIT`
Dependencies	Minimum PHP Version

Git Initialization

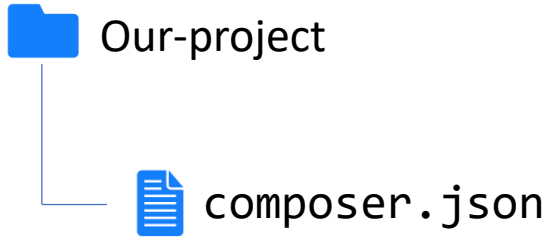
Git Initialization



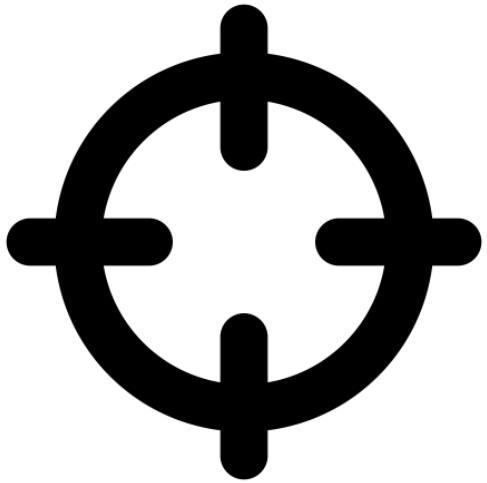
Git Commit Cycle



Commit Our Changes




```
git add -A  →  [ ]  →  git commit –m 'Initial Commit'
```



Write a simple project to greet a person by name

Write a simple project to greet a person by name

`getGreeting(name) -> “Hello name”`

greet.php

```php
function getGreeting($name) {
 return "Hello $name";
}
```

greet.php

```php
function getGreeting(string $name): string {
 return "Hello $name";
}
```

Scalar Typing

greet.php

```php
class Greet {
 static function getGreeting(string $name): string {
  return "Hello $name";
 }
}
```

Classes

greet.php

```php
namespace Ayesh\Greet;

class Greet {
 static function getGreeting(string $name): string {
  return "Hello $name";
 }
}
```

Name Spaces

Greet.php

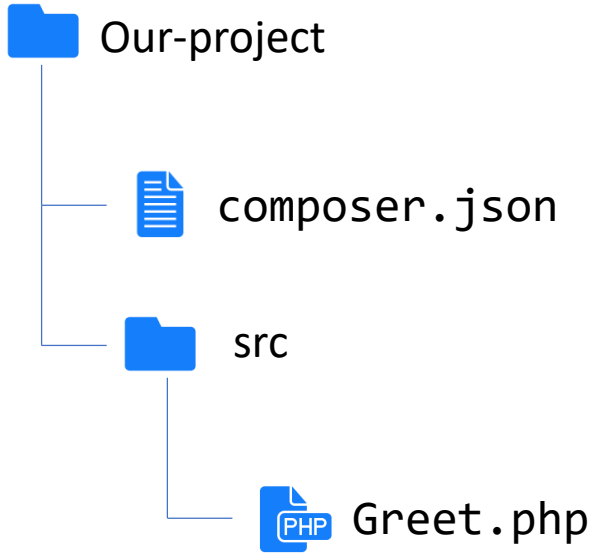
```php
namespace Ayesh\Greet;

class Greet {
 static function getGreeting(string $name): string {
  return "Hello $name";
 }
}
```

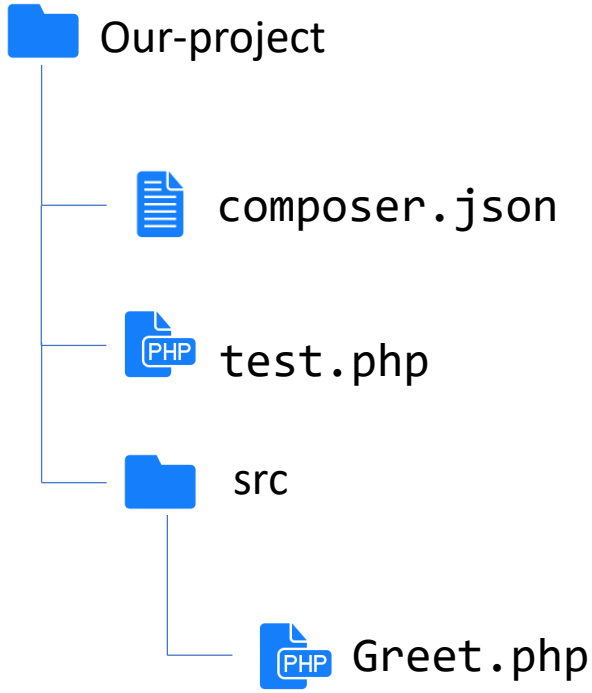
PSR-4 Class Names

src/Greet.php

```php
namespace Ayesh\Greet;

class Greet {
 static function getGreeting(string $name): string {
  return "Hello $name";
 }
}
```

Directory Structure


Our-project
composer.json
src
PHP
Greet.php


Our-project
composer.json
test.php
src
Greet.php

test.php

```php
require_once 'greet.php’;
echo getGreeting('Alexander');
```

test.php

```php
require_once 'greet.php';
echo Ayesh\Greet\Greet::getGreeting('Alexander');
```

test.php

```php
require_once 'greet.php';
echo Ayesh\Greet\Greet::getGreeting('Alexander');
```

Name Spaces

Autoloading Classes

Autoloading Classes

```php
spl_autoload_register('my_autoload_function');

function my_autoload_function(string $class) {
 require_once 'src/' . $class . '.php';
}
```

Autoloading with Composer

Autoloading with Composer

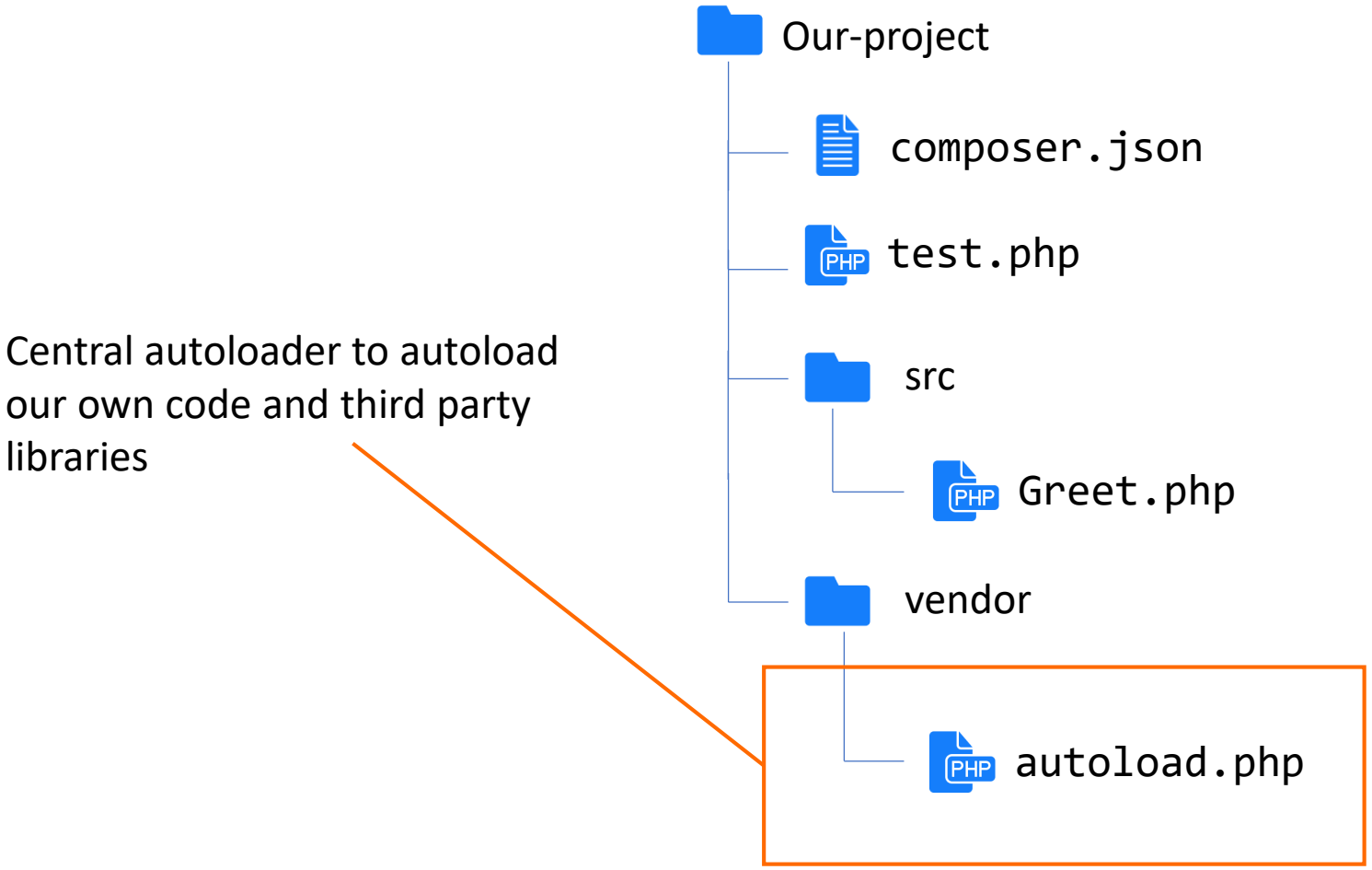
composer.json

```
"autoload": {
  "psr-4": {
    "Ayesh\\Greet\\": "src/"
  }
}
```

Autoloading with Composer


```
composer dump-autoload
```

Autoloading with Composer



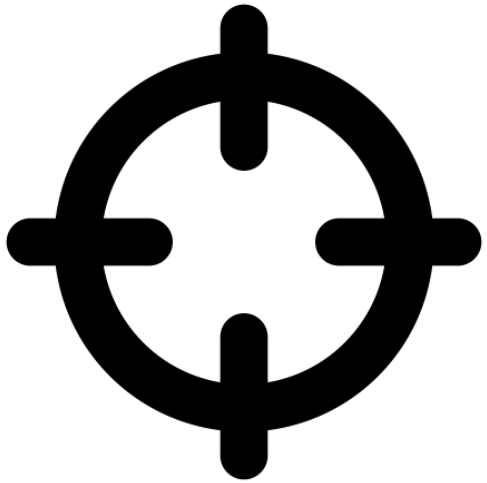
test.php

```php
require_once 'src/Greet.php';
echo Ayesh\Greet\Greet::getGreet('Alexander');
```

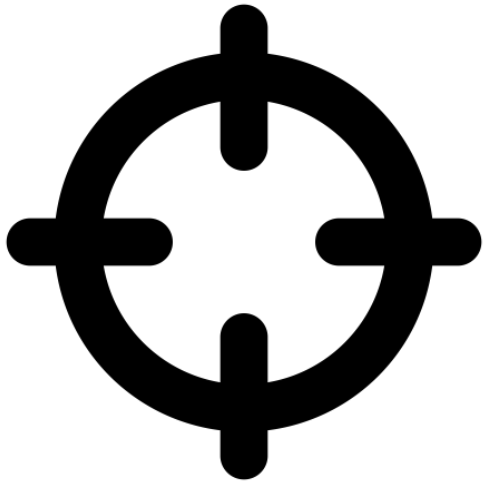
test.php

```php
require_once 'vendor/autoload.php';
echo Ayesh\Greet\Greet::greet('Alexander');
```

Composer Autoloader



Write a simple project to greet a person by name



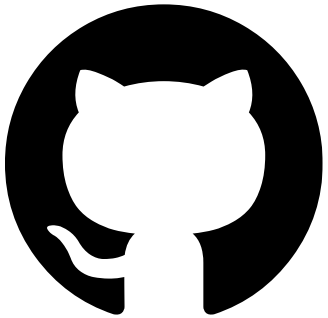
Write a simple project to greet a person by name and use different greetings by the time of the day.

Write a simple project to greet a person by name and use different greetings by the time of the day.

```
getGreeting(name) -> “Hello name”

getGreeting(name) -> 00:00-12:00 “Good Morning name”

                     12:00-03:30 “Good Afternoon name”

                     03:31-24:00 “Good Evening name”
```

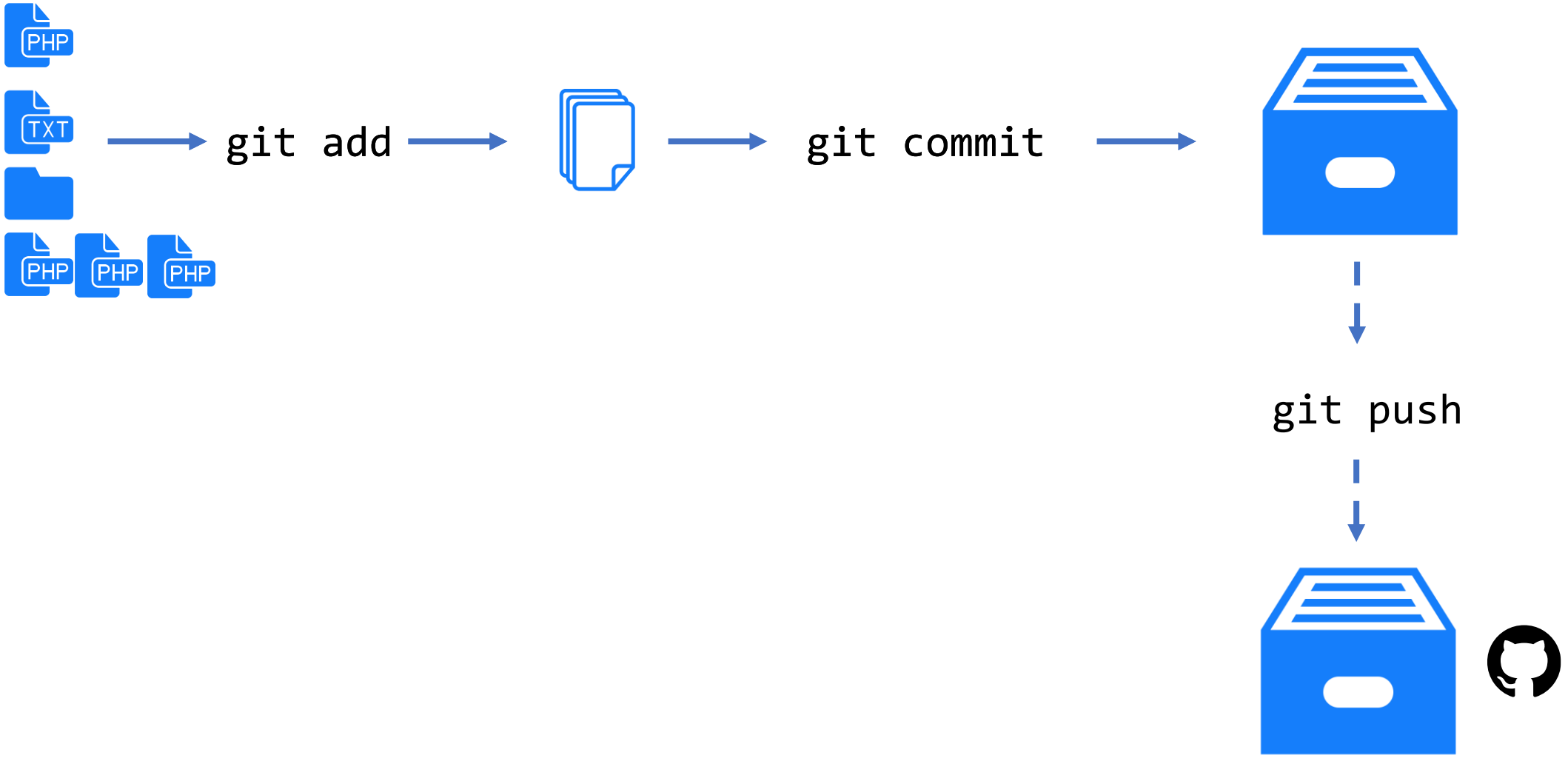
Src/Greet.php

```php
namespace Ayesh\Greet;

class Greet {
 static function getGreeting(string $name): string {
  return self::getGreetingText() . " $name";
 }

 private static function getGreetingText(): string {
  $time = date('G', time());
  if ($time < 12) {
   return 'Good Morning';
  }
  if ($time < 15) {
   return 'Good Afternoon';
  }
  return 'Good evening';
 }
}
```



Publishing Code on GitHub

Publishing Code on GitHub



Publishing Code on GitHub

1. Create Repository
2. Add Git remote URL to local repo
3. Push code

Dependencies

SemVer

Semantic Versioning

Semantic Versioning

1.6.8

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Semantic Versioning

1.6.8

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Semantic Versioning

~ 1.6

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Semantic Versioning

^ 1.6

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Semantic Versioning

^ 2

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Semantic Versioning

^ 1.6.9

1.5	1.5.0	1.5.1	1.5.2			
1.6	1.6.0	1.6.6	1.6.6	1.6.8	1.6.9	1.6.10
1.7	1.7.0	1.7.9	1.7.10	1.7.11		
2.0	2.0.1	2.0.9	2.0.10			
3.0	3.0.0	3.0.4	3.0.97			

Adding Dependencies with Composer

Adding Dependencies with Composer


```
composer require package-vendor/package-name
```

```
composer require package-vendor/package-name --dev
```

Dependencies necessary in development

Adding Dependencies with Composer


```
composer require phpunit/phpunit:^8.1.6
```

PHPUnit versions `>= 8.1.6` and `< 9.0`

Adding Dependencies with Composer

composer.json

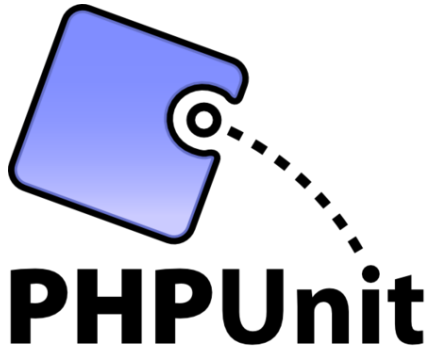
```
"require": {
  "php": "^7.1",
  "phpunit/phpunit": "^8.1.6",
  "ext-curl": "*"
}
```

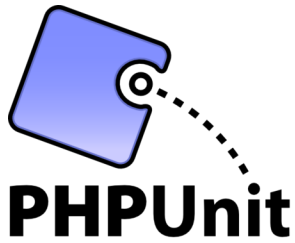
Adding Dependencies with Composer

composer.json

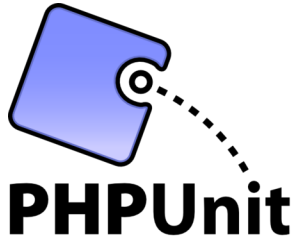
```
"require": {
  "php": "^7.1"
},
"require-dev": {
  "phpunit": "^8.1.6"
}
```

Software Testing


PHPUnit

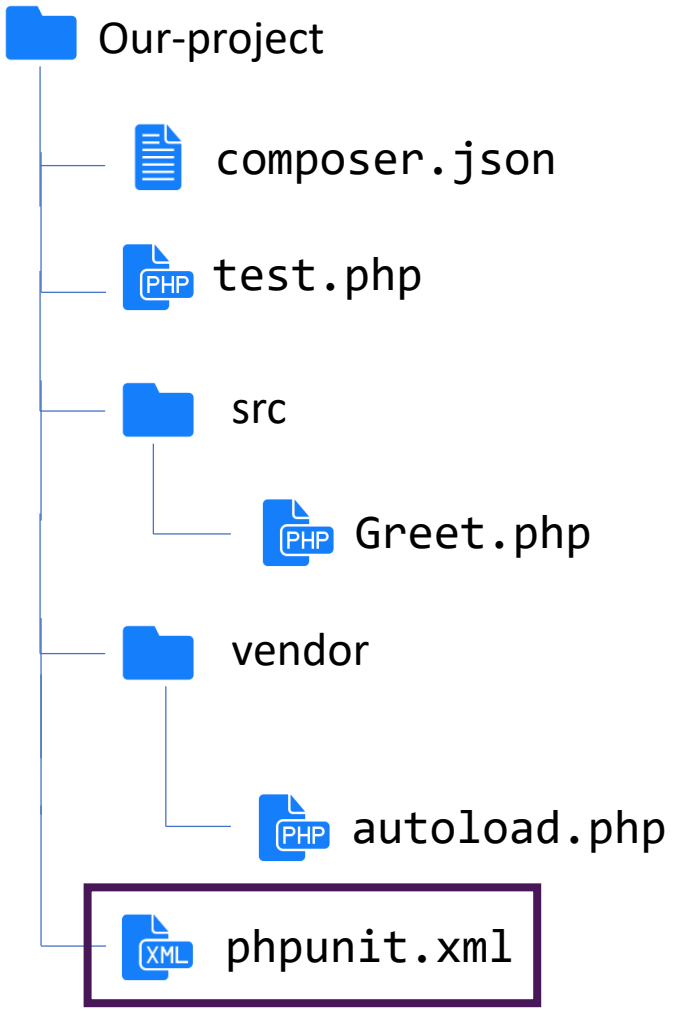



```
composer require phpunit/phpunit:^8.1.6 --dev
```

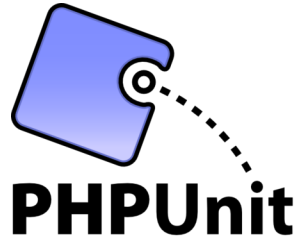



```
phpunit --generate-configuration
```

Bootstrap Script	vendor/autoload.php
Tests Directory	tests
Source Directory	src


Our-project
composer.json
test.php
src
Greet.php
vendor
autoload.php
phpunit.xml

PHPUnit



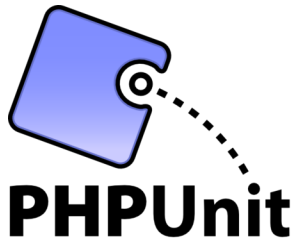
Tests/GreetTest.php

```php
use PHPUnit\Framework\TestCase;

class GreetTest extends TestCase {
    public function testTest() {
        $this->assertSame(2, 1 + 1);
    }
}
```

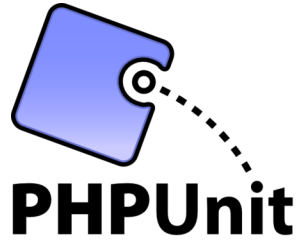
Extend PHPUnit Test

Indicates we write a test

The assertion



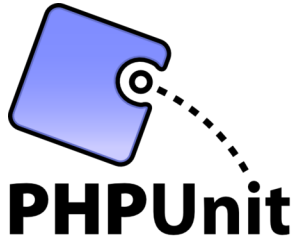

```
phpunit
```

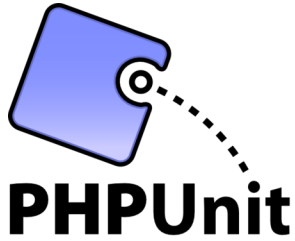


Tests/GreetTest.php

```
public function testGreeting() {
   $name = 'Ayesh';
   $return = Greet::getGreeting($name);
   $this->assertStringContainsString($name, $return);
}
```

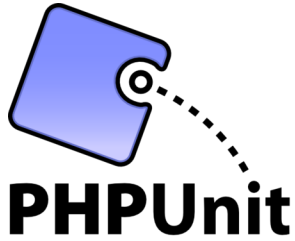
```php
public function testGreetingTextTime() {
  $name = 'John’;
  $text = $this->getExpectedGreetText();
  $return = Greet::getGreeting($name);
  $this->assertEquals("{$text} {$name}", $return);
}

private function getExpectedGreetText(): string {
  $hour = date('G', time());
  if ($hour < 12) {
    return 'Good Morning’;
  }
  if ($hour < 15) {
    return 'Good Afternoon’;
  }
  return 'Good evening';
}
```



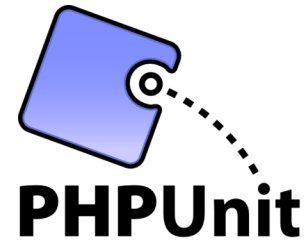


Mocking in Tests

Mocking in Tests



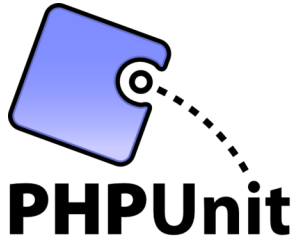

```
composer require symfony/phpunit-bridge --dev
```




```php
namespace Ayesh\Greet\Tests;

use Ayesh\Greet\Greet;
use PHPUnit\Framework\TestCase;

/**
 * @package Ayesh\PHP_Timer\Tests
 * @group time-sensitive
 */
class GreetTest extends TestCase {
  public function testGreeting() {
…
…
```

Give a namespace

Mark as “time-sensitive”
(Symfony-specific)




```php
public function testGreetingTextTime() {
  $name = 'John';
  $text = $this->getExpectedGreetText();
  $return = Greet::getGreeting($name);
  $this->assertEquals("{$text} {$name}", $return);

  sleep(43200); // 12 hours

  $text_2 = $this->getExpectedGreetText();
  $this->assertNotEquals($text_2, $text);
}
```

Doesn't affect the real time

Documentation

Meaningful
Package name

Explain what it does: Meaningful and short

Clear and on-point
Description

Explain what it does, how it does it, and the major requirements.

Helpful
Installation Instructions

- How to use a dependency manager
- Major versions and their differences
- Platform-specific instructions
- Major dependencies

Minimal and functional

Example snippets

- Comment on the examples
- Secure-by-default examples
- Simplest form of the examples

Complete and Up-to-date

Optional features and parameters

- Explain additional parameters
- Type of the parameter required, and their default values
- Exceptions thrown and when they are thrown

Welcome
Contributions

Welcome and explain

How to contribute

- The level of support you provide
- How to contact you for security issues
- Template suggestion for issues/tickets
- How to send patches/pull-requests, and requirements

Meaningful Commit Messages

Meaningful Commit Messages

- “Issue 115: Fix commit messages to be helpful”

- “Fix bug in commit messages to make them more helpful”

- “Fix commit message text

  We can leave additional commit message text by leaving an empty line between the first line and the longer commit message. You can leave as much as information you want, and the longer message will not be shown when a short list of commit messages are shown.”

Meaningful Commit Messages

- A short subject line and a body, separated by a new line
- Subject line length maximum around 50 characters
- Explain why and what, instead of how
- Capitalize first letter of the message
- Imperative mode verbs

PHP Doc and code comments

PHP Doc and code comments

```
/**
 * This is a doc block.
 */
```

PHP Doc and code comments

```php
/**
 * A summary informing the user what the associated element does.
 *
 * A *description*, that can span multiple lines, to go _in-depth_ into
 * the details of this element.
 *
 * @param string $myArgument With a *description* can be multi-line
 *
 * @return void
 */
function myFunction($myArgument)
{
}
```

PHP Doc and code comments

https://phpdoc2cheatsheet.joseruzafa.com/

Continuous Integration
Continuous Delivery

Continuous Integration

Merging all developers' working copies to a shared mainline several times a day.

Continuous Delivery

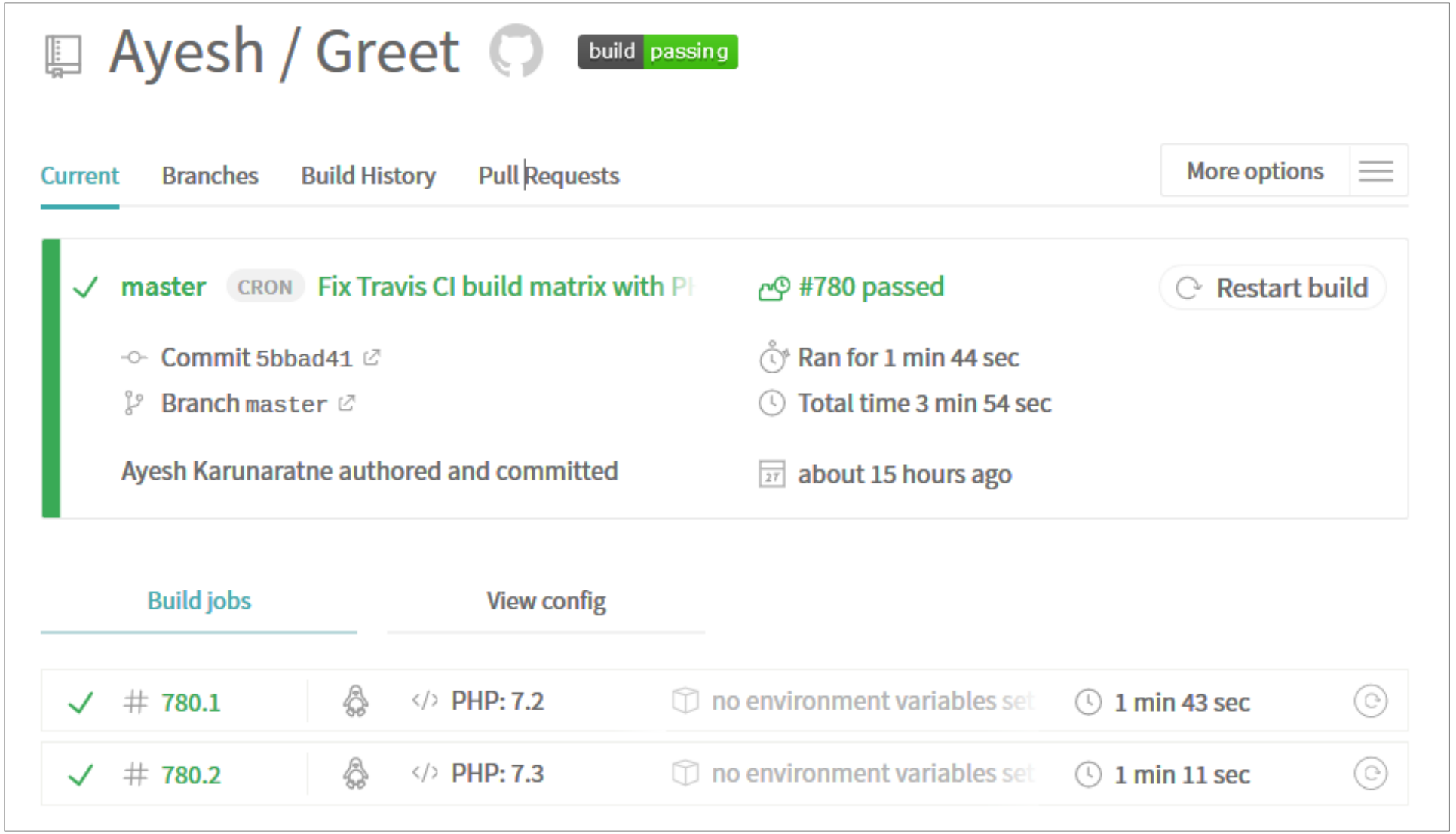
Produce software in short cycles, ensuring fast, frequent and reliable releases,
with frequent building and testing

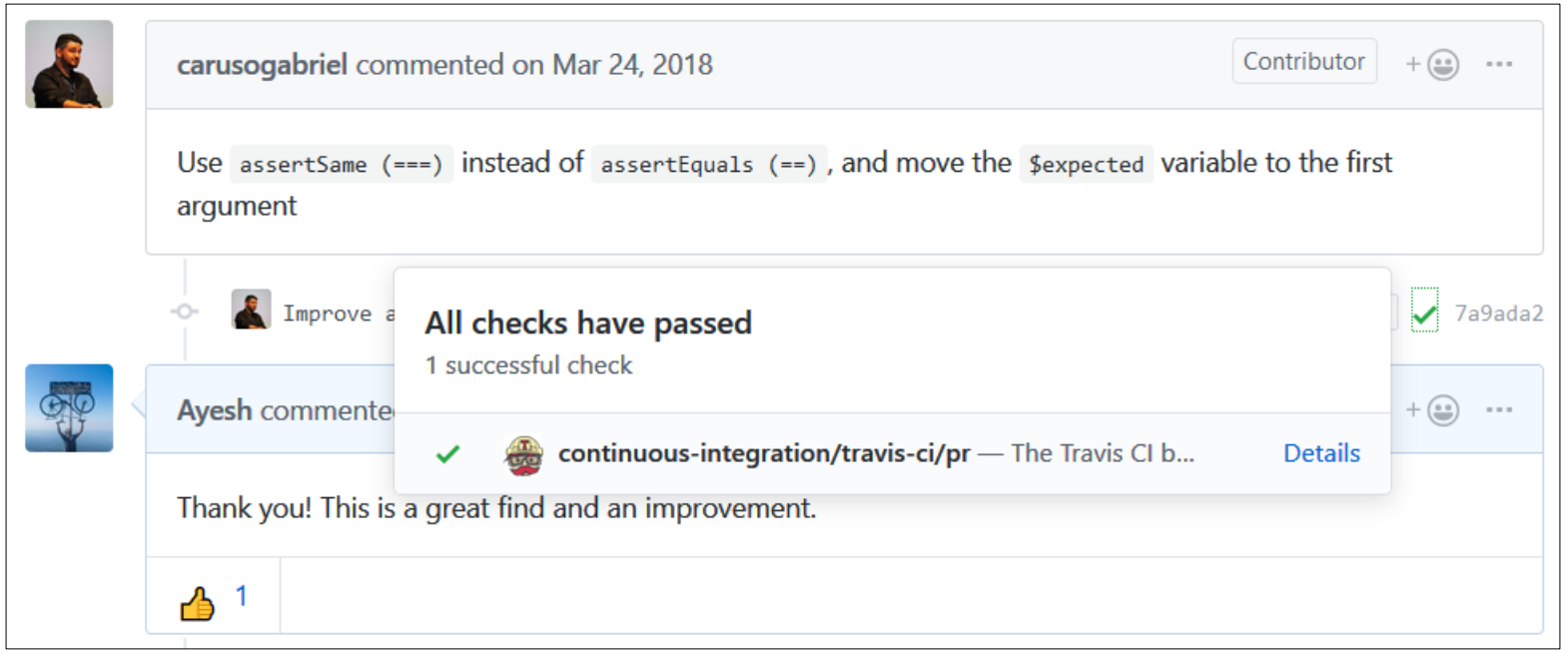

circleci

https://travis-ci.org/

`.travis.yml`

```yaml
language: php
matrix:
  include:
    - php: 7.2
    - php: 7.3
    - php: nightly
  allow_failures:
    - php: 7.4
    - php: nightly
before_script:
  - composer self-update
  - travis_retry composer install --no-interaction
script:
  - vendor/bin/phpunit
```


Ayesh / Greet
build passing
Current
Branches
Build History
Pull Requests
More options
master
CRON
Fix Travis CI build matrix with P
#780 passed
Restart build
Commit 5bbad41
Branch master
Ran for 1 min 44 sec
Total time 3 min 54 sec
Ayesh Karunaratne authored and committed
about 15 hours ago
Build jobs
View config
780.1
PHP: 7.2
no environment variables set
1 min 43 sec
780.2
PHP: 7.3
no environment variables set
1 min 11 sec


carusogabriel commented on Mar 24, 2018
Contributor
Use assertSame (===) instead of assertEquals (==), and move the $expected variable to the first argument
Improve a
7a9ada2
All checks have passed
1 successful check
Ayesh commente
continuous-integration/travis-ci/pr — The Travis CI b...
Details
Thank you! This is a great find and an improvement.
1

Releasing

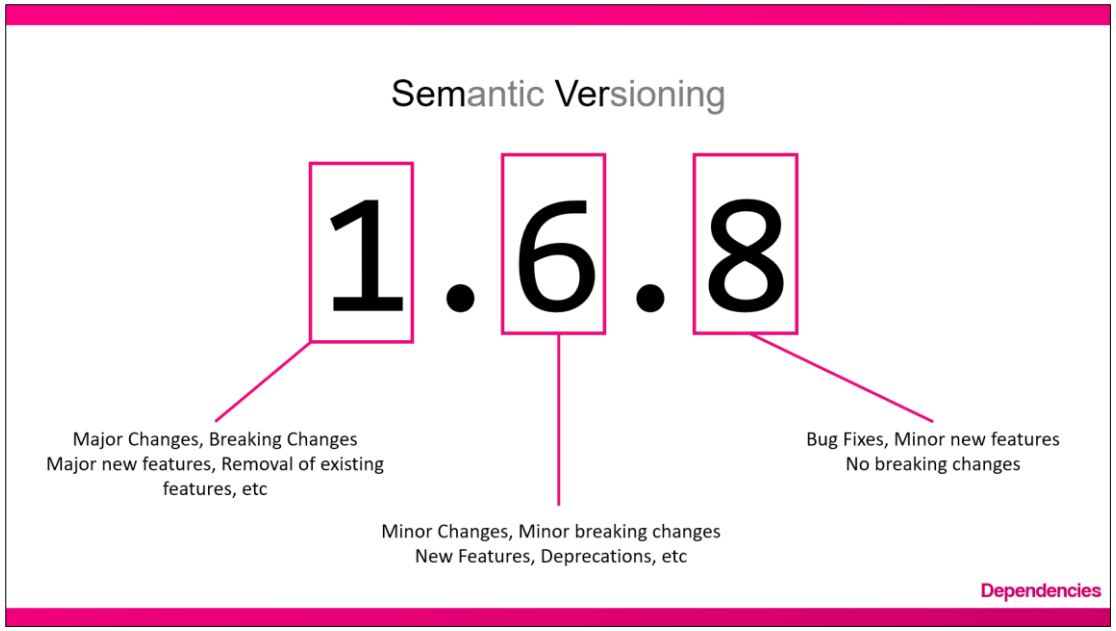
Git Tags

Git Tags

Create a Git tag for every release.

```
git tag v1.0.2
```

```
git push --tags
```

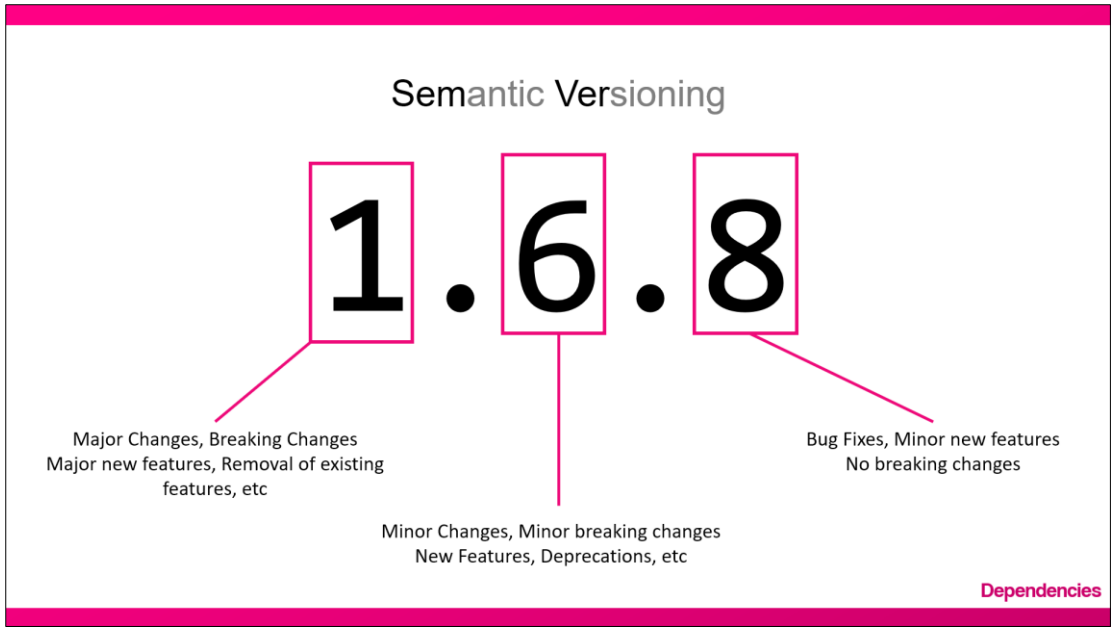


Git Branches

Create a Git branch for every major version

```
git checkout -b v2.x
```

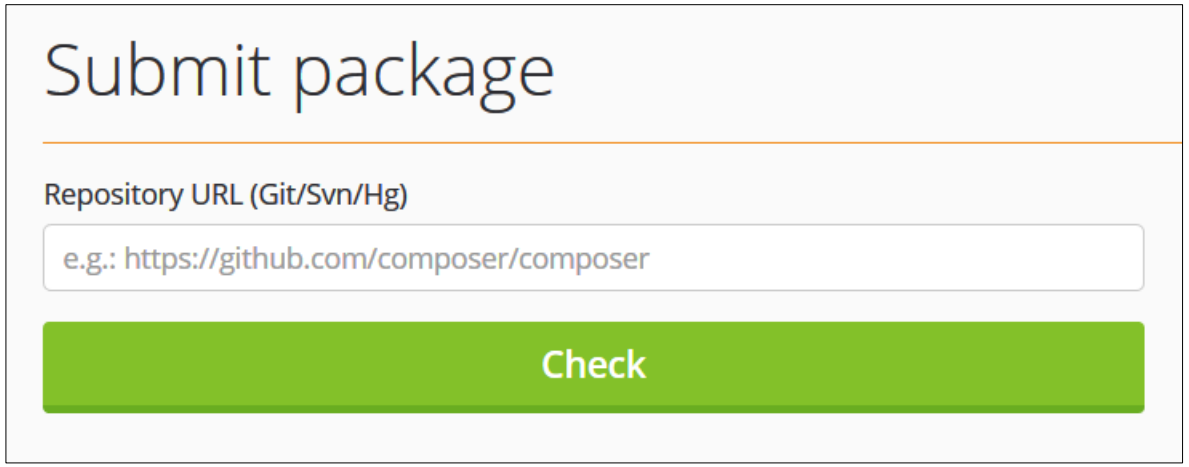
```
git push origin v2.x
```



Publishing on Packagist

Publishing on Packagist

https://packagist.org



Publishing on Packagist

ayesh/is_countable-polyfill

`composer require ayesh/is_countable-polyfill`

A trivial but working polyfill for PHP 7.3 is_countable function.

Supports PHP versions >= 5.3

Abandon Update Edit

Publishing on Packagist

Maintainers

Details

github.com/Ayesh/is_countable-polyfill
Source
Issues

Installs:	29 123
Dependents:	1
Suggesters:	0
Stars:	15
Watchers:	1
Forks:	1
Open Issues:	0

Publishing on Packagist


```
composer require your-name/package-name
```

Maintenance

Backwards-compatible

Bug fixes

Prompt and responsible

Security Releases

Friendly and Welcoming

User Contributions

Proper

Time Management

Being Human

Remember

Behind Every Contribution Is a Human

Appreciate
Contributors Time and Effort

Don’t Forget

We All Make Mistakes

A Key To Success

Strong and Wise Leadership

Important to Have

Rules and Procedures

Make Things Clear

Documentation and Communication

Community and Collaboration

Most Valuable Asset

Community

Real-Life

Meetups and Conferences

Remember
Goals and the Vision

Why Open Source?

PHP Today

Dev Environment Setup

Our First Project

Dependencies

Software Testing

Documentation

Continuous Integration
Continuous Delivery

Releasing

Maintenance

Being Human

Community and Collaboration

Open Source ♡

Questions?

No question is too small. No question is stupid.

Ruby Lounge (upstairs)

@Ayeshlive ayesh@ayesh.me

https://ayesh.me/talk/PHP-Open-Source


THANK YOU
dziękuję
Ďakujem
tak
arigatô
paldies
mahalo
kop khun
děkuji
diolch
dankie
köszönöm
감사합니다
хвала
shukran
tänan
a dank
gràcies
ngiyabonga
Баярлалаа
dhanyavād
Дякую
Благодарам
ευχαριστώ
спасибо
благодаря
tack
grazie
Ǹh’gōi
Dank u
gracias
merci
ස්තුතියි
mulţumesc
ačiū
תודה.
nandri
takk
danke
faleminderit
Xièxiè
choukrane
teşekkür ederim
obrigado
kiitos
Շնորհակալություն
terima kasih
hvala
grazzi

From Zero to Tested Automated Documented Open Source Packages

Ayesh Karunaratne | https://ayesh.me/talk/PHP-Open-Source