Let's Build a Composer Package

Ben Ramsey

24 Oct 2019 • php[world]

Hi. I'm Ben.

- Organizer at Nashville PHP
- Open source advocate
- ramsey/uuid library
- Craft beer nerd
- @ramsey on Twitter
- @ramsey@phpc.social on Mastodon

You will:

- Learn the properties of a “good” package
- Create a package
- Push your package to a repository
- Connect your repo to Travis CI & Coveralls
- Publish your package on Packagist

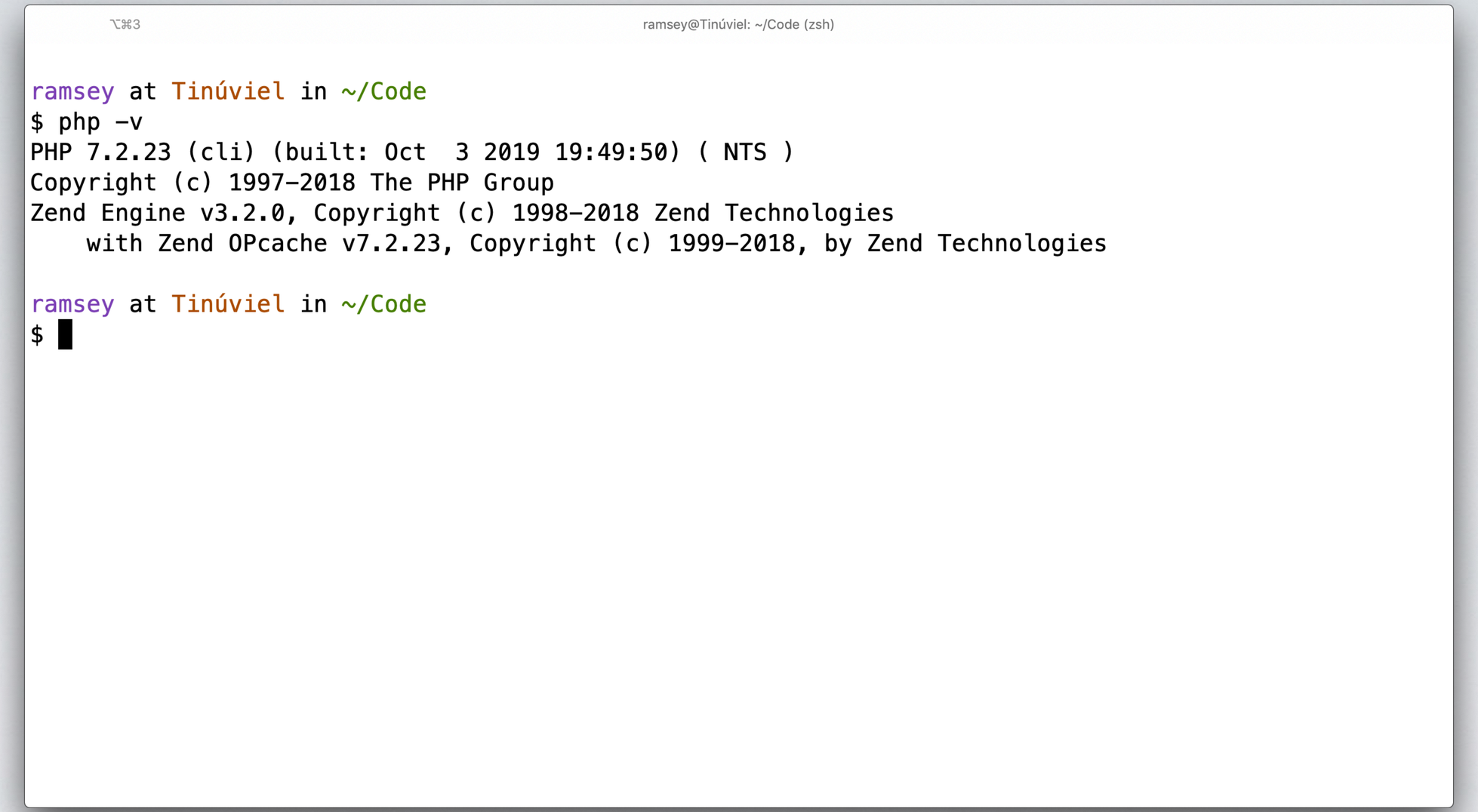
Requirements.

- PHP 7.2+
- Composer

macOS

Homebrew: brew.sh

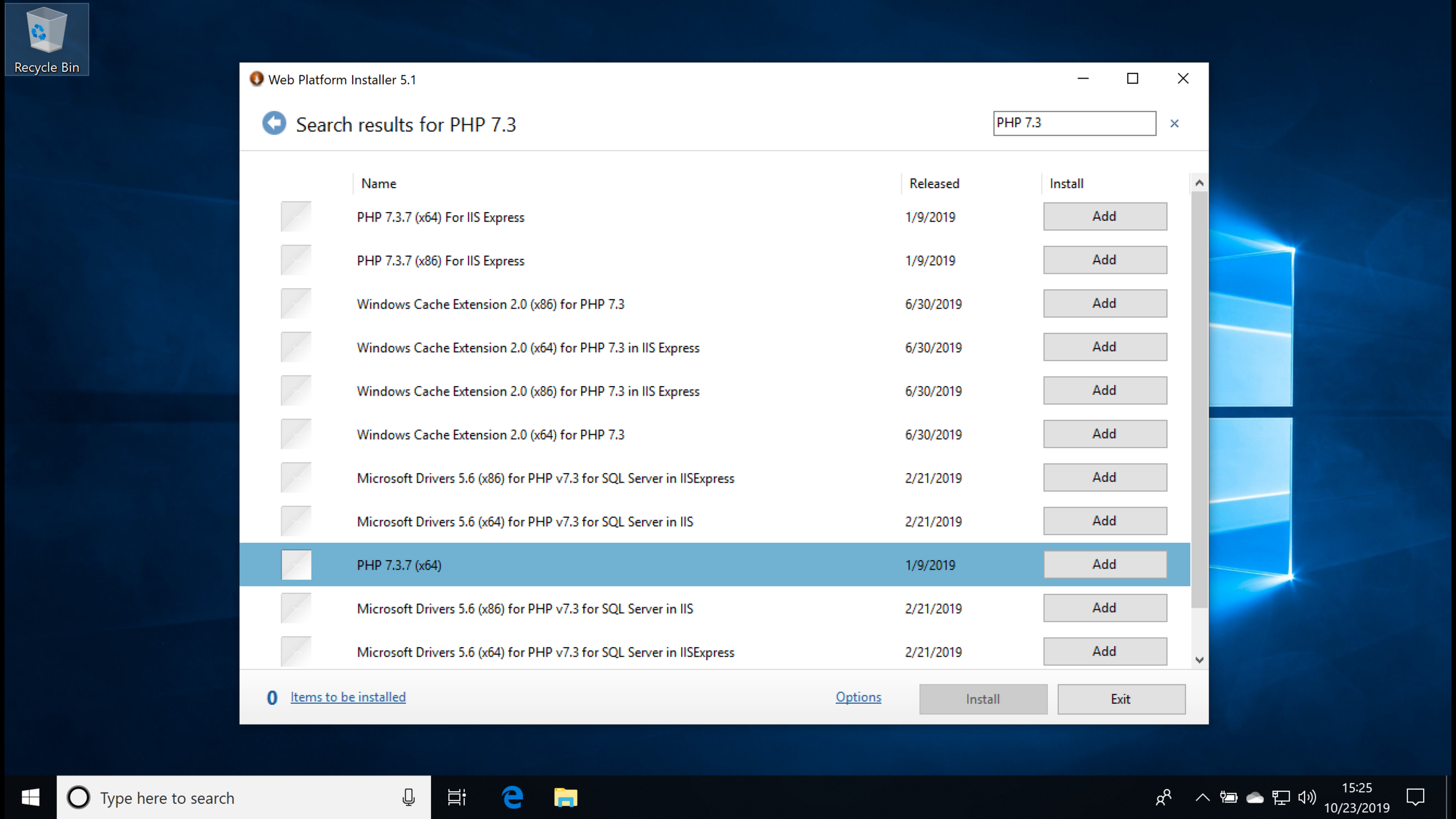
```
brew install php@7.2
export PATH="/usr/local/opt/php@7.2/bin:$PATH"

php -v
```

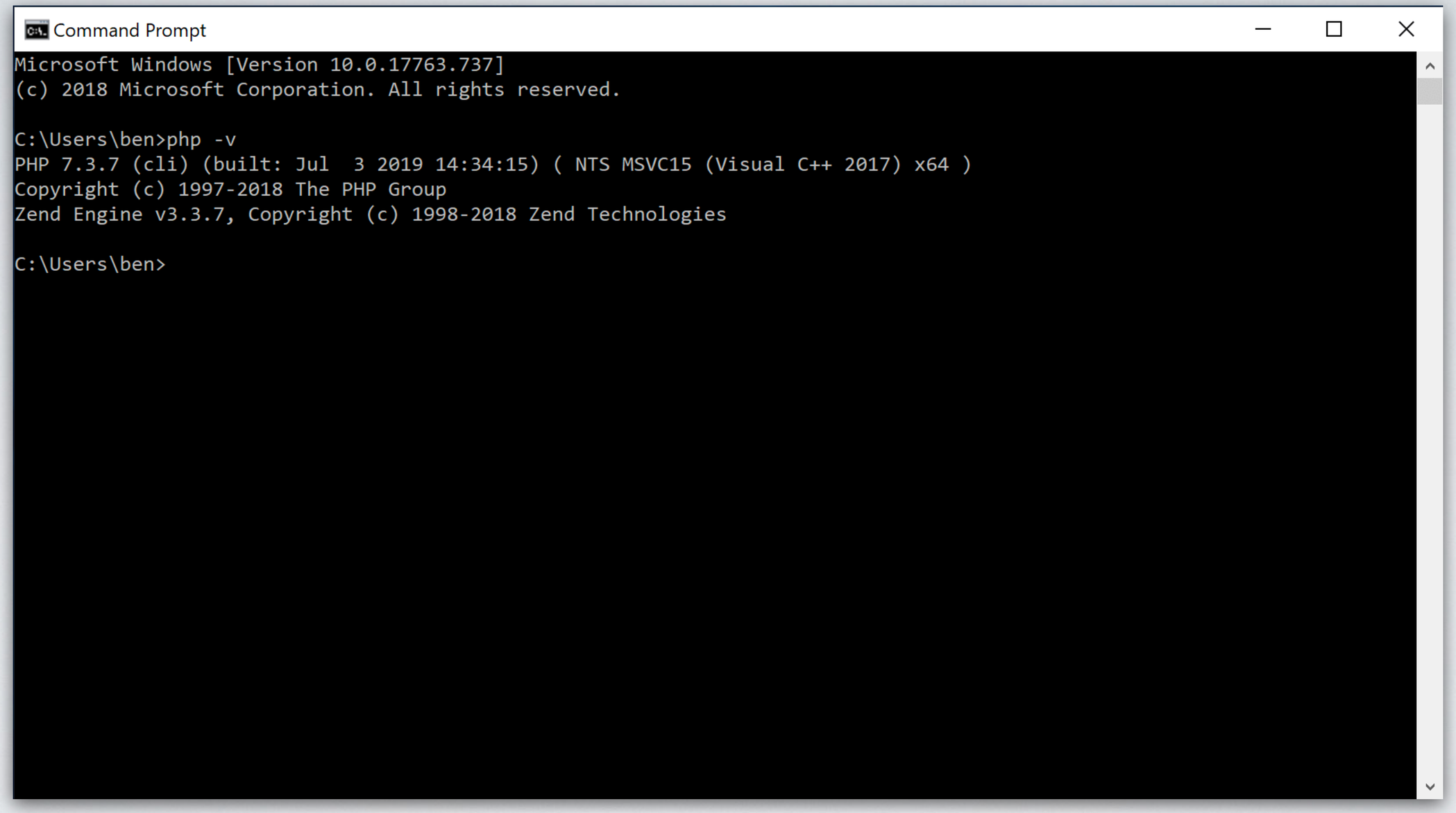

⌥⌘3
ramsey@Tinúviel: ~/Code (zsh)
ramsey at Tinúviel in ~/Code
$ php -v
PHP 7.2.23 (cli) (built: Oct 3 2019 19:49:50) (NTS)
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
with Zend OPcache v7.2.23, Copyright (c) 1999-2018, by Zend Technologies
ramsey at Tinúviel in ~/Code
$

Windows

Web Platform Installer:
microsoft.com/web/downloads/platform.aspx

```
php -v
```


Recycle Bin
Web Platform Installer 5.1
Search results for PHP 7.3
PHP 7.3
Name
Released
Install
PHP 7.3.7 (x64) For IIS Express 1/9/2019 Add
PHP 7.3.7 (x86) For IIS Express 1/9/2019 Add
Windows Cache Extension 2.0 (x86) for PHP 7.3 6/30/2019 Add
Windows Cache Extension 2.0 (x64) for PHP 7.3 in IIS Express 6/30/2019 Add
Windows Cache Extension 2.0 (x86) for PHP 7.3 in IIS Express 6/30/2019 Add
Windows Cache Extension 2.0 (x64) for PHP 7.3 6/30/2019 Add
Microsoft Drivers 5.6 (x86) for PHP v7.3 for SQL Server in IISExpress 2/21/2019 Add
Microsoft Drivers 5.6 (x64) for PHP v7.3 for SQL Server in IIS 2/21/2019 Add
PHP 7.3.7 (x64) 1/9/2019 Add
Microsoft Drivers 5.6 (x86) for PHP v7.3 for SQL Server in IIS 2/21/2019 Add
Microsoft Drivers 5.6 (x64) for PHP v7.3 for SQL Server in IISExpress 2/21/2019 Add
0 Items to be installed
Options
Install
Exit
Type here to search
15:25
10/23/2019


Command Prompt
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\ben>php -v
PHP 7.3.7 (cli) (built: Jul 3 2019 14:34:15) (NTS MSVC15 (Visual C++ 2017) x64)
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.7, Copyright (c) 1998-2018 Zend Technologies
C:\Users\ben>

*nix

Use your favorite package manager.

Composer

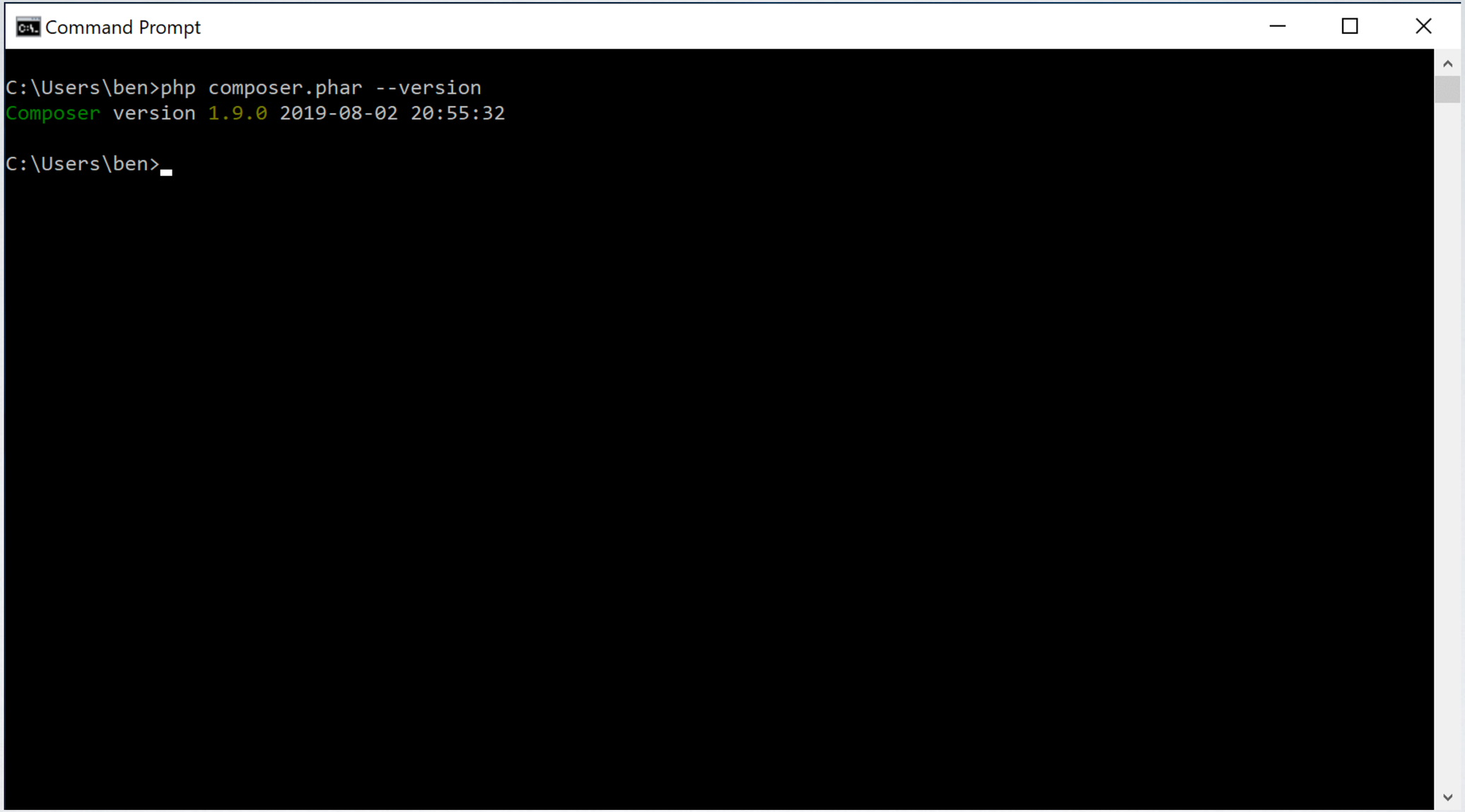
Install Composer: getcomposer.org/download

```
php composer.phar --version
```

Or

```
composer --version
```

```
ramsey at Tinúviel in ~/Code
$ php composer.phar --version
Composer version 1.9.0 2019-08-02 20:55:32

ramsey at Tinúviel in ~/Code
$
```


Command Prompt
C:\Users\ben>php composer.phar --version
Composer version 1.9.0 2019-08-02 20:55:32
C:\Users\ben>


Getting started.

What problem do you want to solve?

You own the code.

How do you want
others to use it?

Licenses & Standards | Open S

https://opensource.org/licenses

Home From The Board Contact Donate Login Search this website..

Open Source Initiative

Guaranteeing the 'our' in source...

ABOUT LICENSES MEMBERSHIP COMMUNITY RESOURCES NEWS & EVENTS

Licenses & Standards

About Open Source Licenses

Open source licenses are licenses that comply with the Open Source Definition — in brief, they allow software to be freely used, modified, and shared. To be approved by the Open Source Initiative (also known as the OSI), a license must go through the Open Source Initiative's license review process.

open source initiative
Approved License®

Popular Licenses

The following OSI-approved licenses are popular, widely used, or have strong communities:

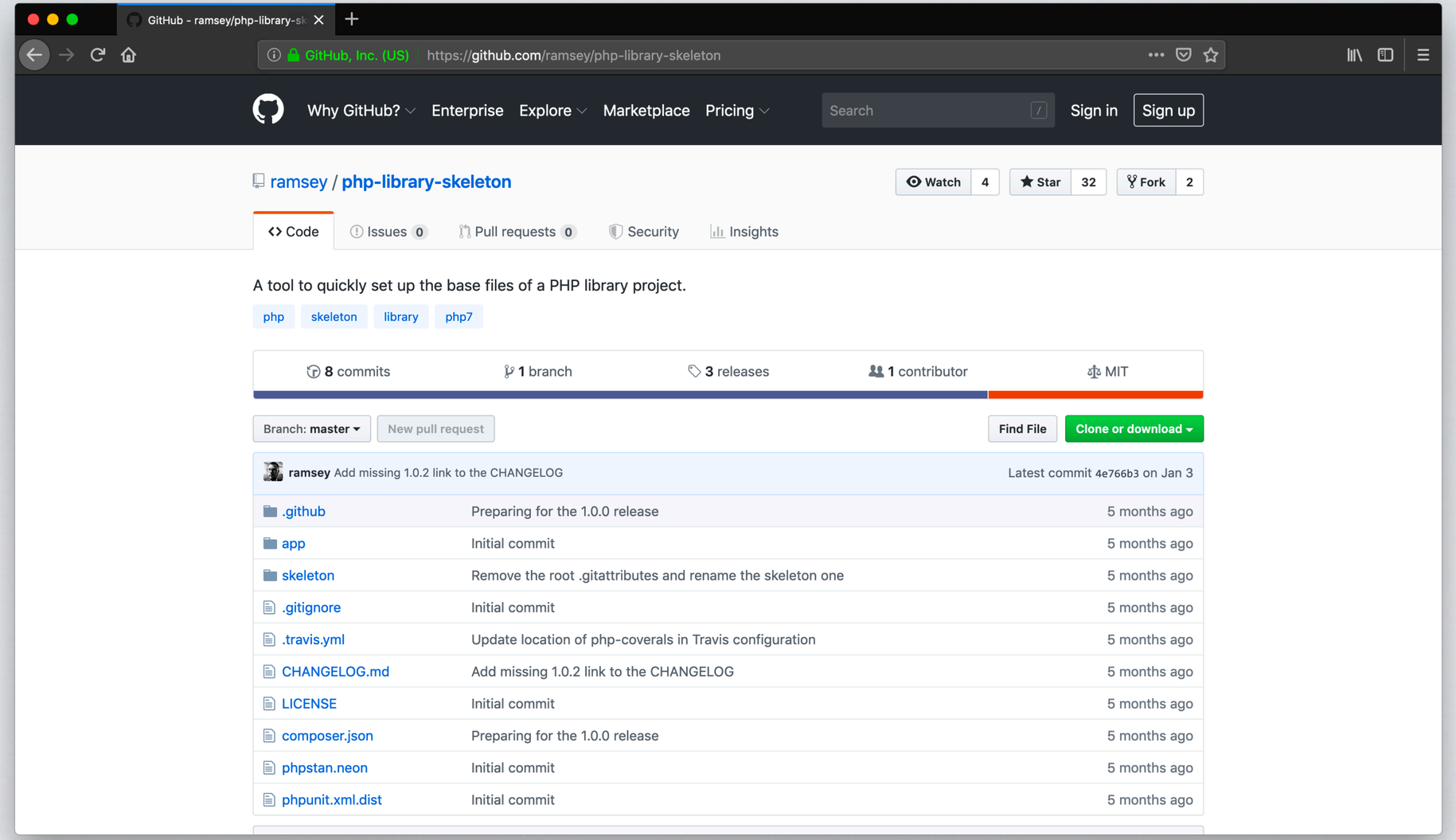
- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0

- MIT license
- BSD 2-Clause “Simplified”
- Apache License 2.0
- GNU Library or “Lesser” General Public License (LGPL)

Now we can begin.

The project.

github.com/ramsey/php-library-skeleton



```
ramsey at Tinúviel in ~/Code
$ composer create-project --remove-vcs ramsey/php-library-skeleton phpworld-package
```

Let’s Build!

Maintaining your package.

.gitattributes

- Add your `tests/` directory
- Add your `docs/` directory
- Add your `phpunit.xml.dist` file

Add anything that shouldn’t be bundled into the distribution when someone requires your package with Composer.

```
.gitattributes      export-ignore
.github/            export-ignore
.gitignore          export-ignore
.travis.yml         export-ignore
docs/               export-ignore
phpstan.neon        export-ignore
phpunit.xml.dist    export-ignore
tests/              export-ignore
```

Testing.

- PHPUnit, Atoum, Kahlan, etc.
- Aim for *sufficient* code coverage
- Make it easy to run your tests:
  `composer test`

Static analysis.

- PHPStan, Exakat, Phan, Psalm, PHP Inspections (JetBrains), etc.
- Finds errors in your code without running it
- Especially helpful when multiple people are working on the same project

Coding standards.

- PSR-12, Symfony, CakePHP, WordPress, etc.
- Use PHP_CodeSniffer to ensure your code follows your chosen standards
- Catch standards issues before code review

Scripts.

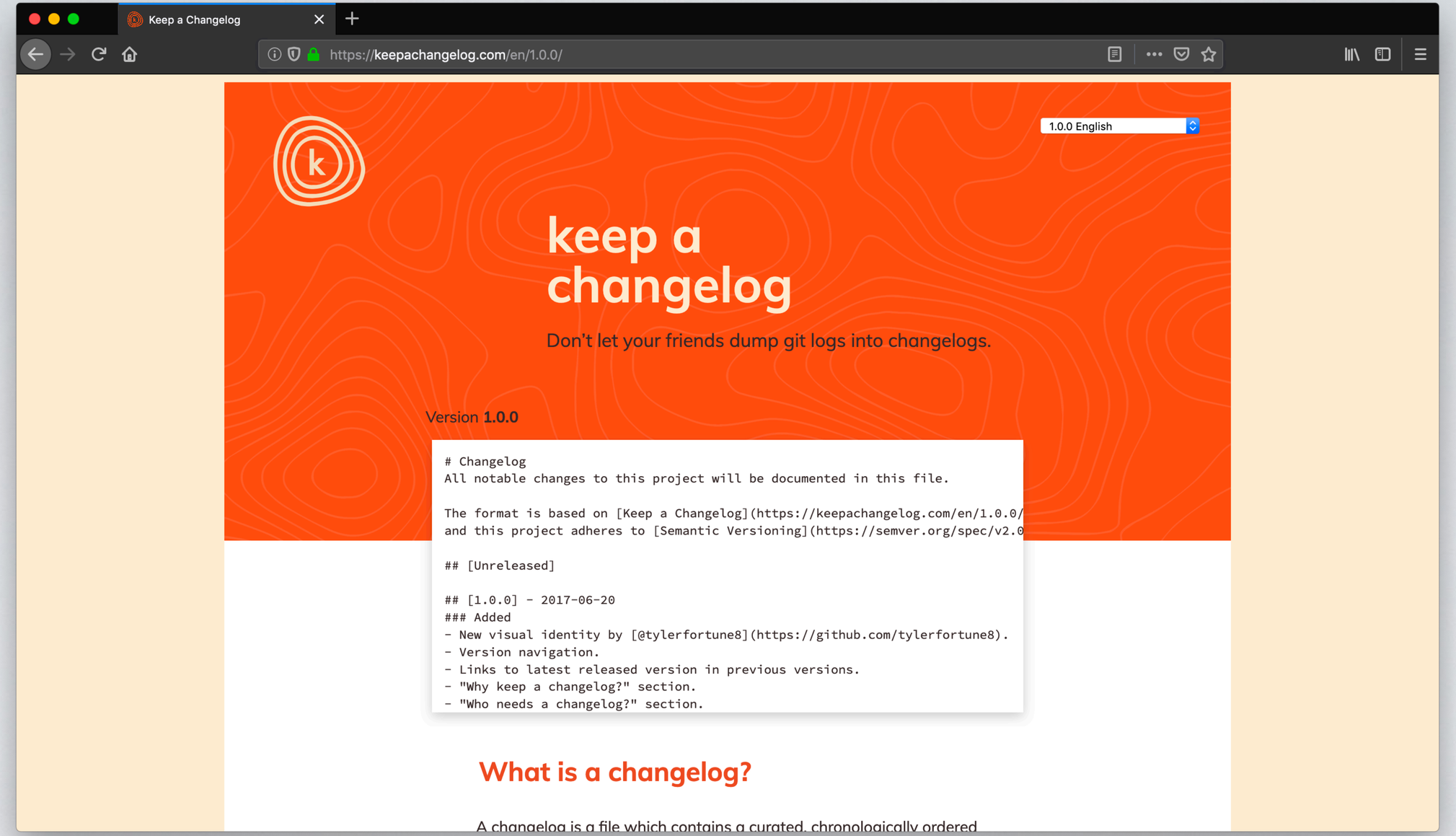
- Just like with npm and package.json, Composer supports custom scripts
- Custom scripts can help your contributors
- Use them to consolidate tools into simple, easy-to-remember commands

```
"scripts": {
  "lint": "parallel-lint src tests",
  "phpcs": "phpcs src tests --standard=psr12 -sp --colors",
  "phpstan": [
    "phpstan analyse src -c phpstan.neon --level max --no-progress",
    "phpstan analyse tests -c phpstan.neon --level 4 --no-progress"
  ],
  "phpunit": "phpunit --verbose --colors=always",
  "phpunit-ci": "phpunit --coverage-clover build/logs/clover.xml",
  "phpunit-coverage": "phpunit --coverage-html build/coverage",
  "test": ["@lint", "@phpcs", "@phpstan", "@phpunit"],
  "test-ci": ["@lint", "@phpcs", "@phpstan", "@phpunit-ci"]
}
```

```
ramsey at Tinúviel in ~/Code/phpworld-package
$ composer test
```

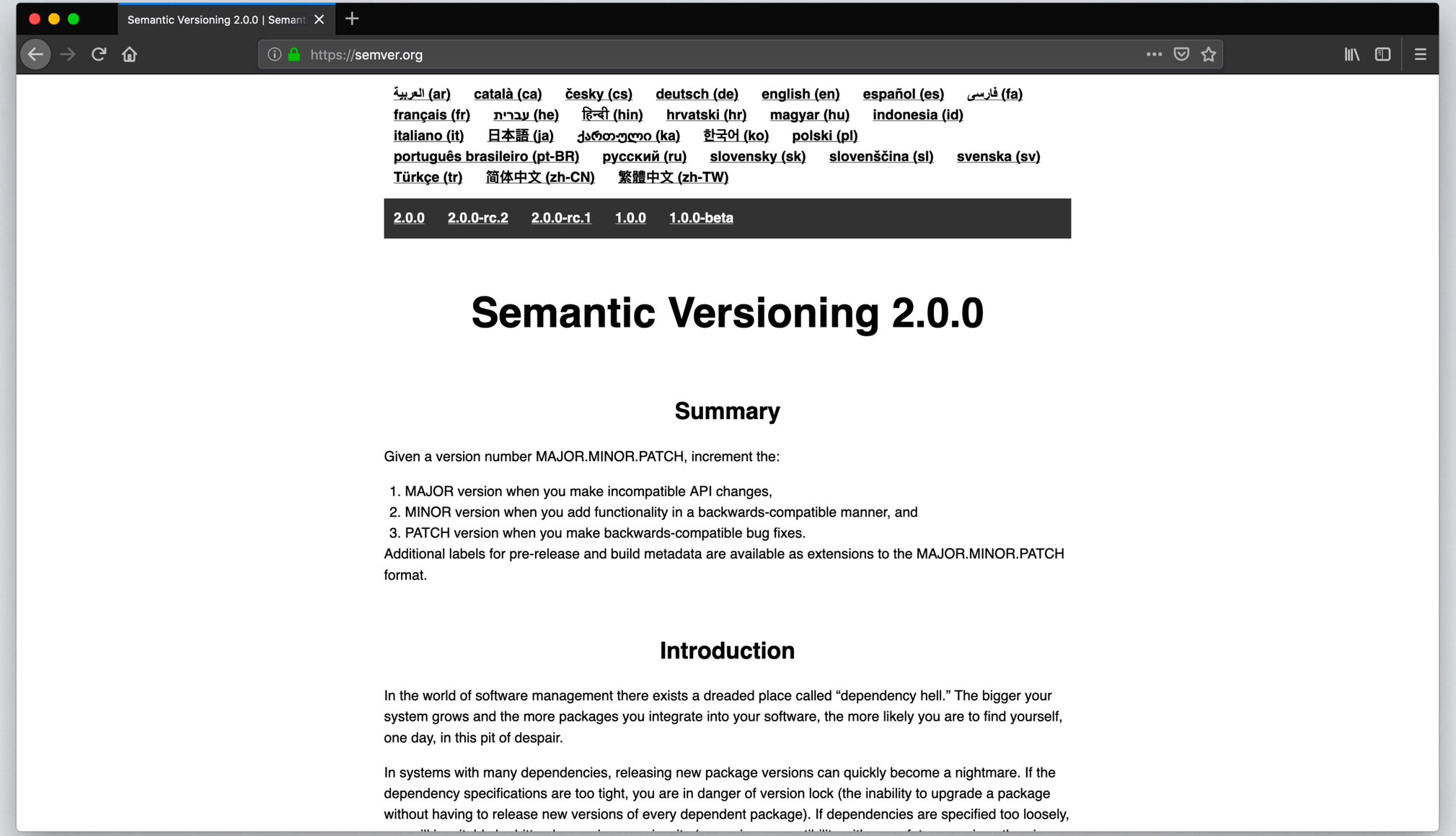
Changelog.

- Tells your users what has changed between versions
- Log changes, no matter how big or small
- Log what you've added, changed, deprecated, removed, fixed, & secured


Keep a Changelog
https://keepachangelog.com/en/1.0.0/
1.0.0 English
keep a changelog
Don't let your friends dump git logs into changelogs.
Version 1.0.0
Changelog
All notable changes to this project will be documented in this file.
The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.0.0/
and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0
[Unreleased]
[1.0.0] - 2017-06-20
Added
- New visual identity by [@tylerfortune8](https://github.com/tylerfortune8).
- Version navigation.
- Links to latest released version in previous versions.
- "Why keep a changelog?" section.
- "Who needs a changelog?" section.
What is a changelog?
A changelog is a file which contains a curated, chronologically ordered

Versioning.

- Follow semantic versioning
- Backwards-compatibility breaks?
  - ***Major release***
- New features that don’t break BC?
  - ***Minor release***
- Bug fixes that don’t break BC?
  - ***Patch release***



Semantic Versioning 2.0.0

Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards-compatible manner, and
3. PATCH version when you make backwards-compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

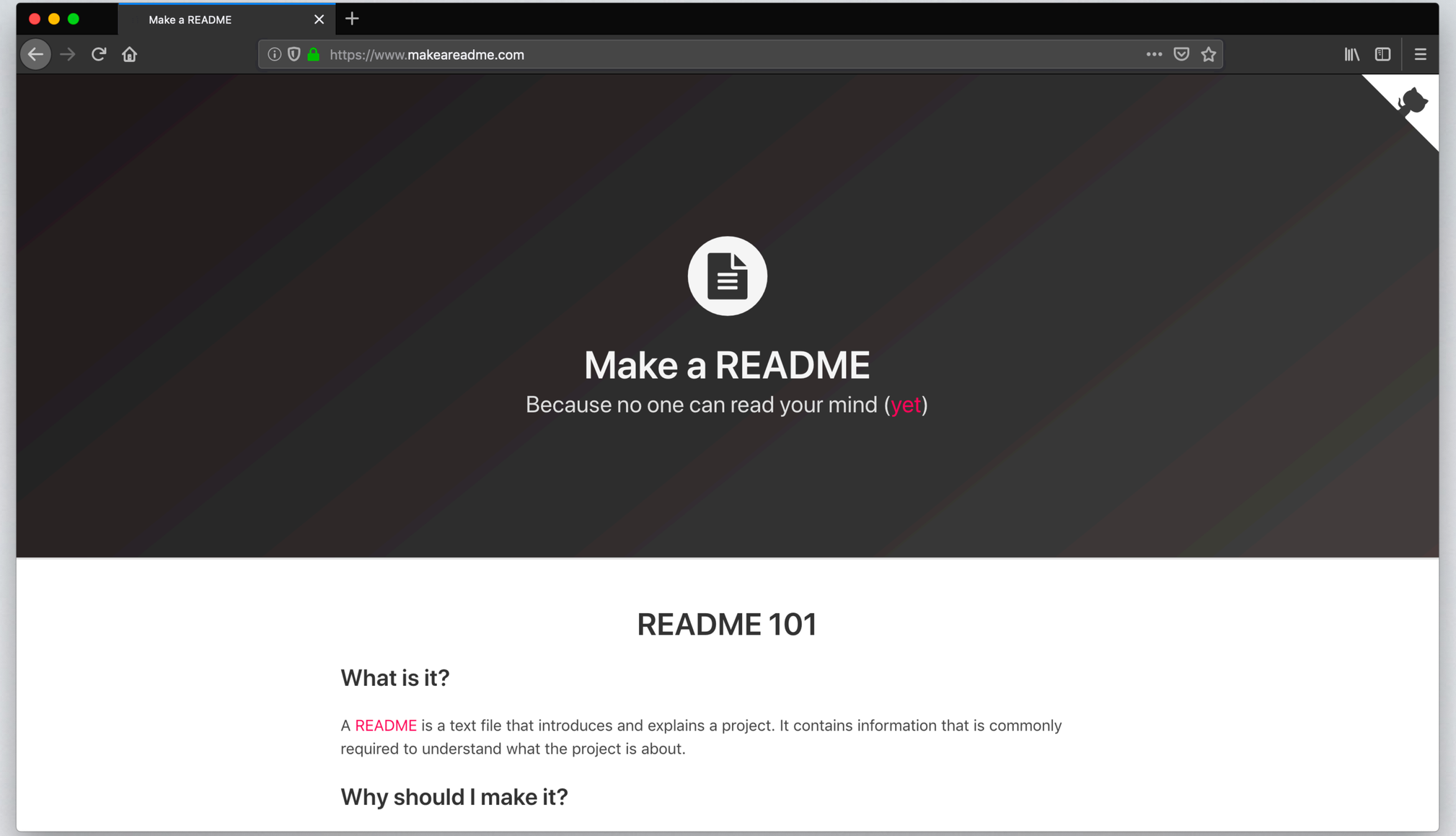
Introduction

In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

In systems with many dependencies, releasing new package versions can quickly become a nightmare. If the dependency specifications are too tight, you are in danger of version lock (the inability to upgrade a package without having to release new versions of every dependent package). If dependencies are specified too loosely,

Building your community.

Readme.


Make a README
https://www.makeareadme.com
Make a README
Because no one can read your mind (yet)
README 101
What is it?
A README is a text file that introduces and explains a project. It contains information that is commonly required to understand what the project is about.
Why should I make it?

Contributing.

Wrangling Web Contributions: How

https://mozillascience.github.io/working-open-workshop/contributing/

mozilla Science Lab

Wrangling Web Contributions: How to Build a CONTRIBUTING.md

> *In this activity you will learn how to create and maintain a CONTRIBUTING.md for an open source project.*

30 minutes
For beginners

Format

This exercise works well as an in-person workshop or an online exercise. It can be done individually or in a group.

Target Audience

Open science project and community leads and Mozilla Study Group leads seeking to attract and grow communities of contributors around their projects

Materials

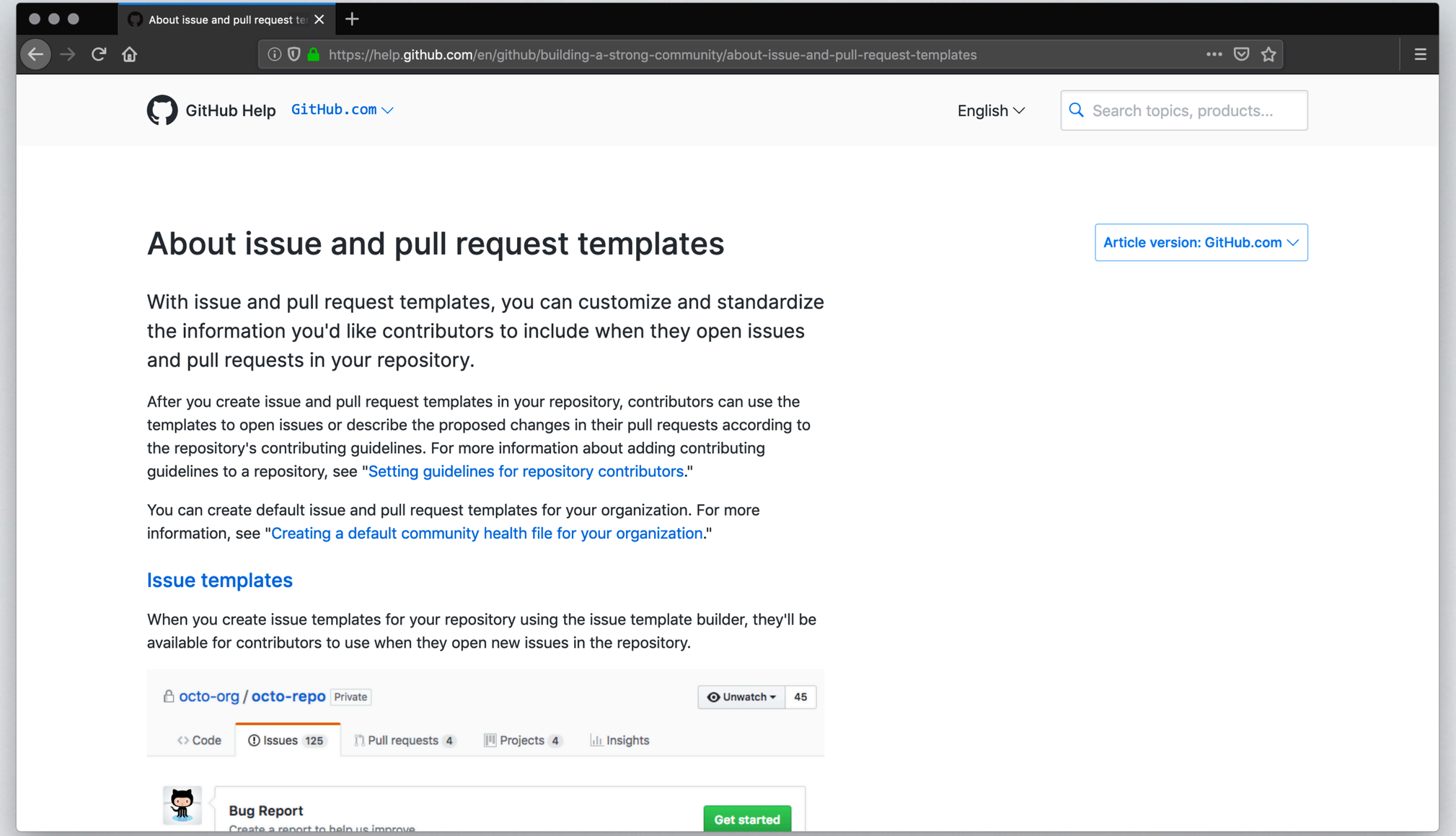
- Pen/pencil & paper
- Collaborative document editor like Etherpad or Google Docs
- A markdown viewer (such as Mou) to test your file before posting online

Introduction

A CONTRIBUTING.md file, in your open source repository or site, provides potential project contributors with a short guide to how they can help with your project or study group. It is convention to capitalize the word "contributing" as the file title, and to save it as a resource in markdown (hence the extension .md).

This file is for:

Templates.


About issue and pull request te
https://help.github.com/en/github/building-a-strong-community/about-issue-and-pull-request-templates
GitHub Help
GitHub.com
English
Search topics, products...
About issue and pull request templates
Article version: GitHub.com
With issue and pull request templates, you can customize and standardize the information you'd like contributors to include when they open issues and pull requests in your repository.
After you create issue and pull request templates in your repository, contributors can use the templates to open issues or describe the proposed changes in their pull requests according to the repository's contributing guidelines. For more information about adding contributing guidelines to a repository, see "Setting guidelines for repository contributors."
You can create default issue and pull request templates for your organization. For more information, see "Creating a default community health file for your organization."
Issue templates
When you create issue templates for your repository using the issue template builder, they'll be available for contributors to use when they open new issues in the repository.
octo-org / octo-repo
Private
Unwatch
45
Code
Issues 125
Pull requests 4
Projects 4
Insights
Bug Report
Get started

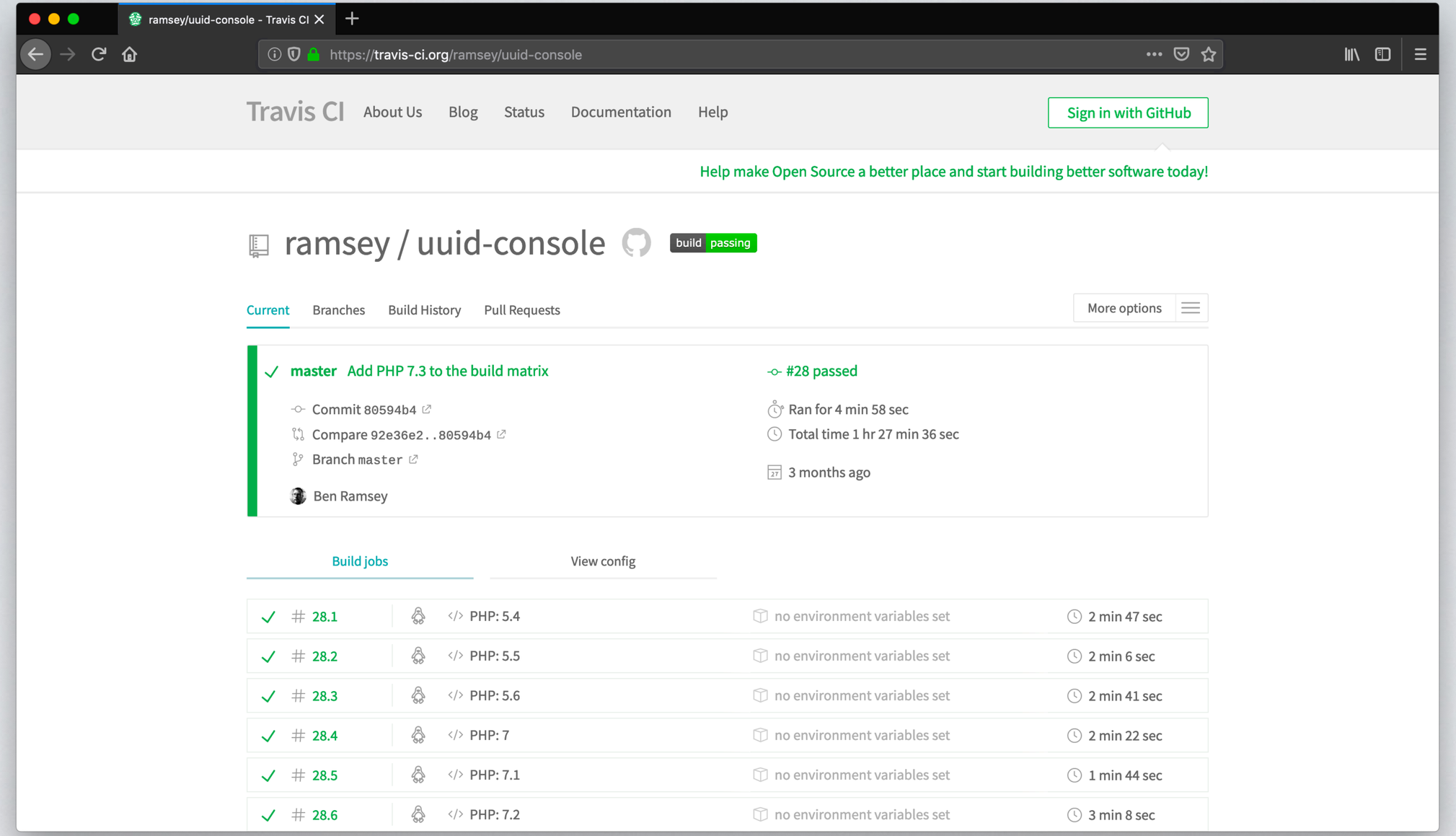
Code of conduct.

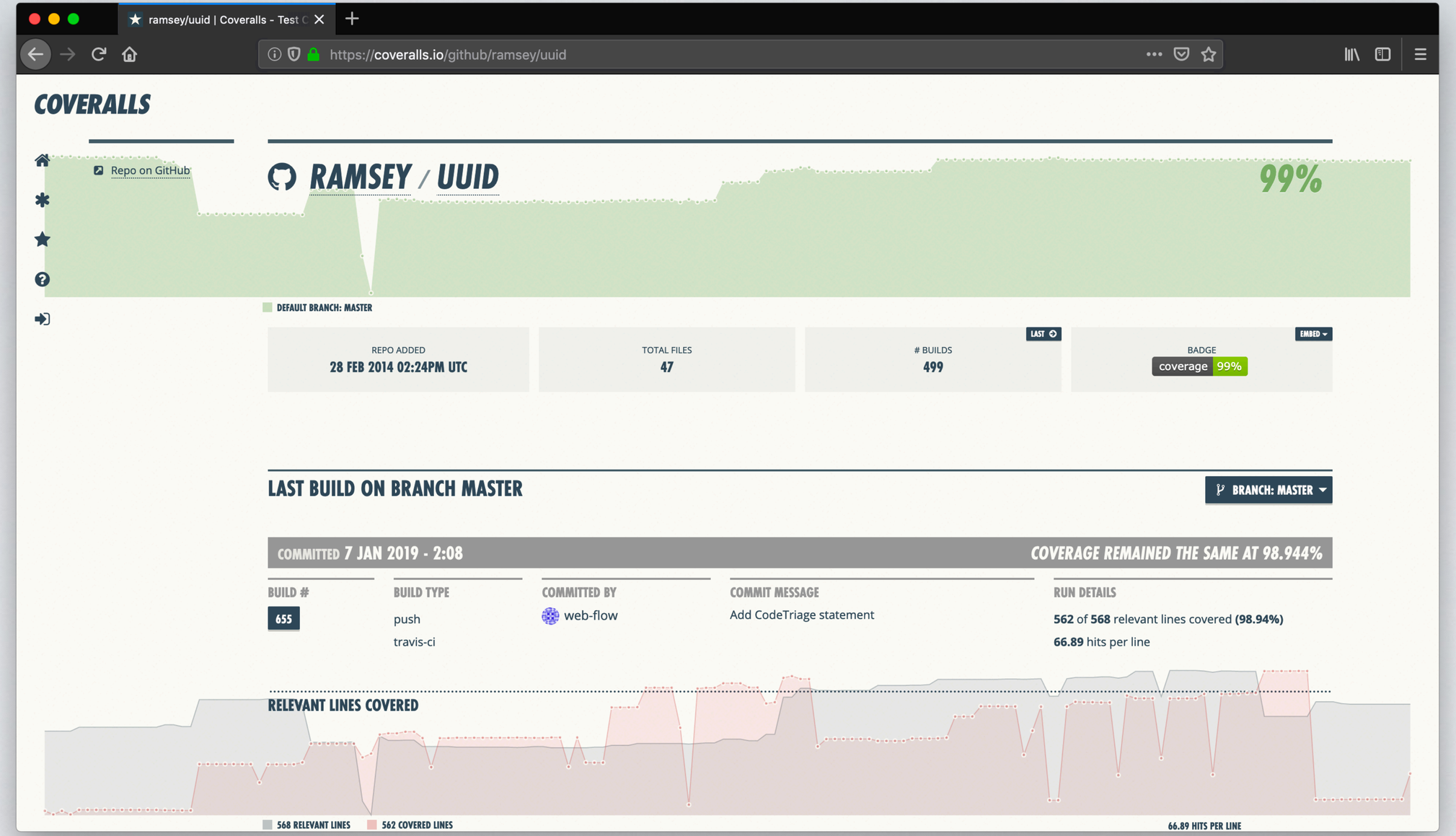
Your Code of Conduct

Facilitate healthy and constructive community behavior by adopting and enforcing a code of conduct.

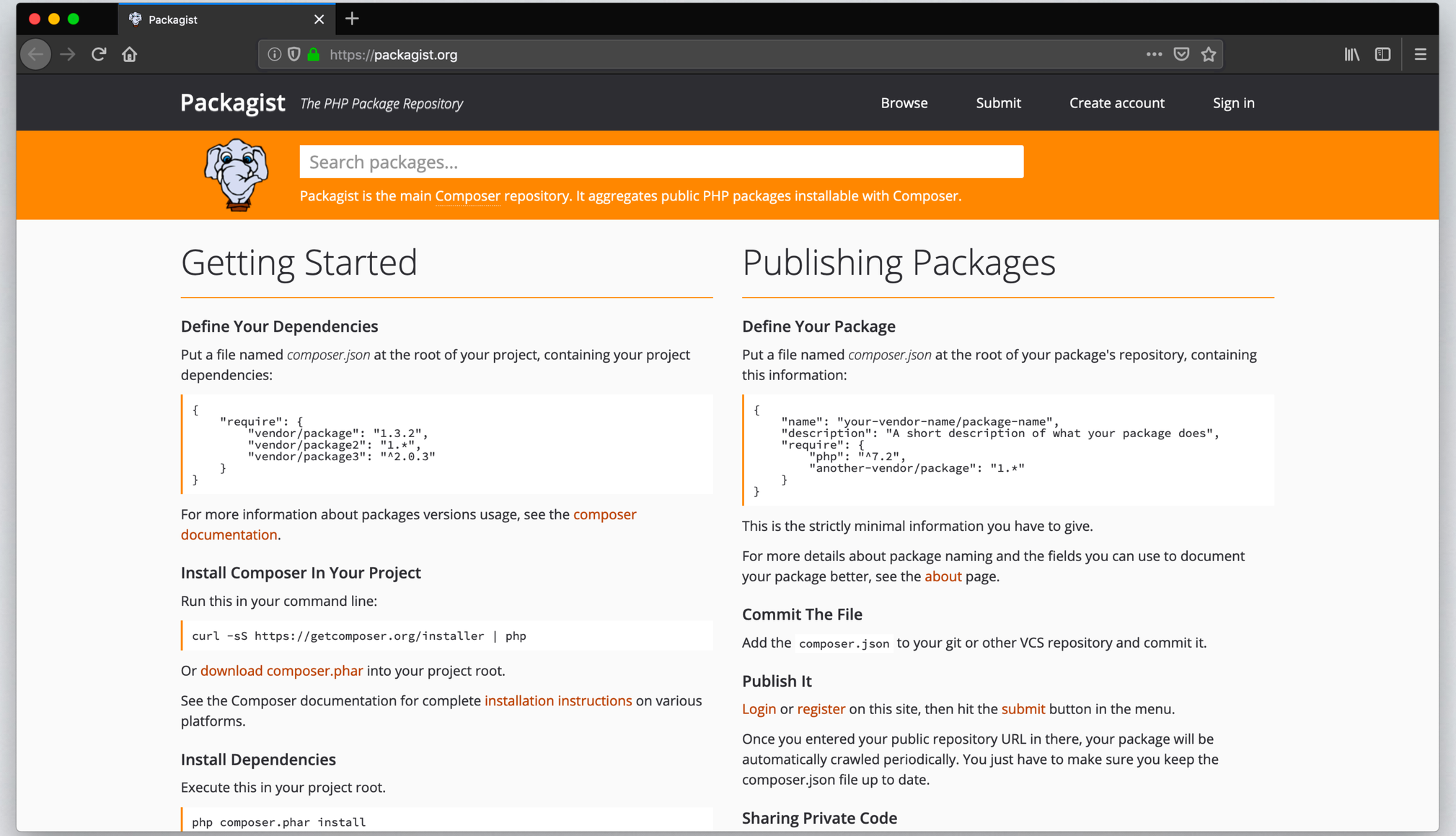
Table of Contents

Automation.


ramsey/uuid-console - Travis CI
https://travis-ci.org/ramsey/uuid-console
Travis CI
About Us
Blog
Status
Documentation
Help
Sign in with GitHub
Help make Open Source a better place and start building better software today!
ramsey / uuid-console
build passing
Current
Branches
Build History
Pull Requests
More options
master Add PHP 7.3 to the build matrix
#28 passed
Commit 80594b4
Compare 92e36e2..80594b4
Branch master
Ran for 4 min 58 sec
Total time 1 hr 27 min 36 sec
3 months ago
Ben Ramsey
Build jobs
View config
28.1 PHP: 5.4 no environment variables set 2 min 47 sec
28.2 PHP: 5.5 no environment variables set 2 min 6 sec
28.3 PHP: 5.6 no environment variables set 2 min 41 sec
28.4 PHP: 7 no environment variables set 2 min 22 sec
28.5 PHP: 7.1 no environment variables set 1 min 44 sec
28.6 PHP: 7.2 no environment variables set 3 min 8 sec


ramsey/uuid | Coveralls - Test C
https://coveralls.io/github/ramsey/uuid
COVERALLS
Repo on GitHub
RAMSEY / UUID
99%
DEFAULT BRANCH: MASTER
REPO ADDED
28 FEB 2014 02:24PM UTC
TOTAL FILES
47
BUILDS
499
LAST
EMBED
BADGE
coverage 99%
LAST BUILD ON BRANCH MASTER
BRANCH: MASTER
COMMITTED 7 JAN 2019 - 2:08
COVERAGE REMAINED THE SAME AT 98.944%
BUILD #
655
BUILD TYPE
push
travis-ci
COMMITTED BY
web-flow
COMMIT MESSAGE
Add CodeTriage statement
RUN DETAILS
562 of 568 relevant lines covered (98.94%)
66.89 hits per line
RELEVANT LINES COVERED
568 RELEVANT LINES
562 COVERED LINES
66.89 HITS PER LINE

Publishing.


Packagist
https://packagist.org
Packagist The PHP Package Repository
Browse
Submit
Create account
Sign in
Search packages...
Packagist is the main Composer repository. It aggregates public PHP packages installable with Composer.
Getting Started
Define Your Dependencies
Put a file named composer.json at the root of your project, containing your project dependencies:
{
"require": {
"vendor/package": "1.3.2",
"vendor/package2": "1.*",
"vendor/package3": "^2.0.3"
}
}
For more information about packages versions usage, see the composer documentation.
Install Composer In Your Project
Run this in your command line:
curl -sS https://getcomposer.org/installer | php
Or download composer.phar into your project root.
See the Composer documentation for complete installation instructions on various platforms.
Install Dependencies
Execute this in your project root.
php composer.phar install
Publishing Packages
Define Your Package
Put a file named composer.json at the root of your package's repository, containing this information:
{
"name": "your-vendor-name/package-name",
"description": "A short description of what your package does",
"require": {
"php": "^7.2",
"another-vendor/package": "1.*"
}
}
This is the strictly minimal information you have to give.
For more details about package naming and the fields you can use to document your package better, see the about page.
Commit The File
Add the composer.json to your git or other VCS repository and commit it.
Publish It
Login or register on this site, then hit the submit button in the menu.
Once you entered your public repository URL in there, your package will be automatically crawled periodically. You just have to make sure you keep the composer.json file up to date.
Sharing Private Code


Enjoy the FOSS community.

Use a welcoming voice.

Build up.

Don’t tear down.

Open Source Guides | Learn ho

https://opensource.guide

About Contribute English

Open Source Guides

Open source software is made by people just like you. Learn how to launch and grow your project.

How to Contribute to Open Source

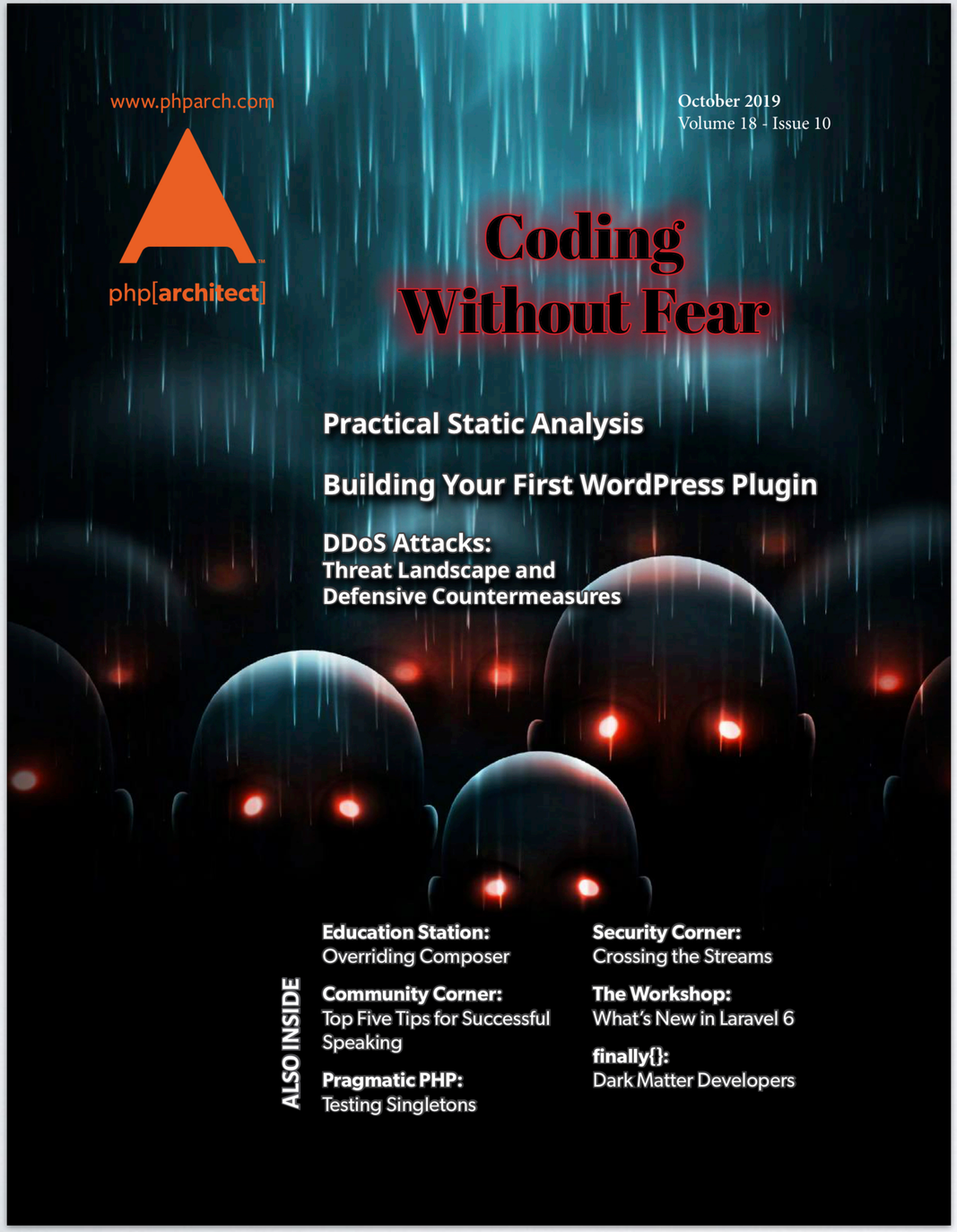
Want to contribute to open source? A guide to making open source contributions, for first-timers and for veterans.

Starting an Open Source Project

Learn more about the world of open source and get ready to launch your own project.

LINKS TO RESOURCES

1. Composer: https://getcomposer.org/
2. Packagist: https://packagist.org/
3. Open Source Licenses: https://opensource.org/licenses
4. ramsey/php-library-skeleton: https://github.com/ramsey/php-library-skeleton
5. PHP Framework Interoperability Group: http://www.php-fig.org/
6. PHP Package Development Standards: http://php-pds.com/
7. Semantic Versioning: http://semver.org/
8. PHP CodeSniffer: https://github.com/squizlabs/PHP_CodeSniffer
9. PHPUnit: https://phpunit.de/
10. phpstan: https://github.com/phpstan/phpstan
11. Travis CI: https://travis-ci.org/
12. CHANGELOG.md: http://keepachangelog.com
13. Code of Conduct
    - Contributor Covenant: http://contributor-covenant.org/
    - Citizen Code of Conduct: http://citizencodeofconduct.org/
14. CONTRIBUTING.md: http://mozillascience.github.io/working-open-workshop/contributing/
15. Open Source Guides: https://opensource.guide/
16. A Beginner's Guide to Creating a Readme: https://changelog.com/posts/a-beginners-guide-to-creating-a-readme
17. Use .gitattributes: https://blog.madewithlove.be/post/gitattributes/



Overriding Composer

Chris Tankersley

Composer[1] is one of the most influential tools to have come out of the PHP ecosystem. Not only did it help revolutionize the PHP package ecosystem by making it easy to autoload code, find packages, and keep track of dependencies, it also stands as one of the best package managers from any language, hands-down. Python's pip, NuGet for .NET, Rust's cargo, and Go's dep all pale in comparison to the stability and simple usage of Composer.

Modern PHP development requires developers to understand package and dependency management. Unless you are writing code in a vacuum and are not allowed to use third party packages, Composer is the first tool you reach for in a project. It helps download that shiny framework you want to use or grabs the handful of starter packages you need.

I am going to assume that you, the reader, have a basic understanding of how to run the Composer binary, and what `require` versus `install` does. I want to highlight some different sections of the `composer.json` file you can use to streamline your development, especially if you are releasing code to the public.

The Before Times

Before Composer, there were basically two ways to pull in code from other people: using PEAR or manually.

The "official" way was to find a package published on PEAR[2]. It was a command-line tool that would (usually) download packages to a shared directory, and you could include that directory to pull in packages.

PEAR was reasonably simple to use. `pear install foo` downloaded the `foo` package, much like `composer require foo/bar` downloads a package. PEAR also handled dependencies as packages and could declare other PEAR packages they required. It even supported pulling from different servers, called Channels, so you could run your PEAR cache or even a custom server for your project.

The only real legacy most developers see from this is PSR-0, which took it's naming convention from how PEAR separated directories and code. Under the PEAR naming convention, you convert underscores to directory separators, and the class lived in a file named after the class. For example, you'd find the class named `Foo_Bar` in `foo/bar.php`.

Modern namespacing and PSR-0 and PSR-4 autoloaders have supplanted this to a point, so we no longer have to do `new Foo_Bar()` anymore. PEAR made it easier to find packages, namespacing made it easier to name and organize code, and Composer was able to bundle autoloading in by default to make including it easier.

The other way to include code was to copy the code, either by downloading an archive or copying it into a new file and figuring out the best way to manually include it. We do not want to talk about the number of problems that caused. At least with PEAR, we could easily get dependency resolution, and it was somewhat curated.

Anatomy of a Basic Composer.json

Composer requires a `composer.json` file to run. This file contains all the information it can use to figure out what dependencies your application needs. If you do not intend for anyone to use your code as a dependency (e.g., if you are building a full application) you may only have a `require` section, which lists all the dependencies—including PHP versions and extensions—for your application. Listing 1 is an example of `composer.json`.

A close companion to the `require` section is the `require-dev` section. Any

Listing 1

```
{
    "require": {
        "php": "^7.2",
        "dragonmantank/cron-expression": "~2.1",
        "nexmo/client": "^2.0",
        "slim/slim": "3.12.2"
    },
    "require-dev": {
        "phpunit/phpunit": "^8"
    }
}
```

1 *Composer:* https://getcomposer.org

2 *PEAR:* https://pear.php.net

THANK YOU.
ANY QUESTIONS?

If you want to talk more, feel free to contact me.

benramsey.com

@ramsey

@ramsey@phpc.social

github.com/ramsey

ben@benramsey.com

This presentation was created using Keynote. The text is set in Chunk Five and Helvetica Neue. The source code is set in Menlo. The iconography is provided by Font Awesome.

Unless otherwise noted, all photographs are from Unsplash or Pixabay and are used with permission.

Ramsey, Ben. “Let’s Build a Composer Package” php[world]. Sheraton Tysons Hotel, Tysons, VA. 24 Oct. 2019. Conference presentation.