Composer
Best Practices
2018

Nils Adermann
@naderman
Private Packagist
https://packagist.com
2018?

Delete your lock files
Delete your lock files

2018?
Packages/versions over time

The last data point is for the current month and shows partial data.
Best Practices?
Deployment
Improving your deployment process

- Slow Deployment
  - You will not enjoy deploying
- Unreliable deployment
  - You will be scared to deploy
- You deploy infrequently
  - more work to debug older problems
  - no incentive to improve the process
- Vicious cycle
  - Reliability and speed are key to breaking it
Reduce dependence on external services

- **Build Process** (move more into this)
  - Install dependencies (Composer, npm, ...)
  - Generate assets (Javascript, CSS, generated PHP code, ...)
  - Create an artifact with everything in it

- **Deploy Process** (make this as small as possible)
  - Move the artifact to your production machine
    - sftp, rsync, apt-get install
  - Machine dependent configuration
  - Database modifications
  - Start using new version
Never Deploy without a Lock File

Do not run composer update during deployments
Reduce dependence on external services

- composer install loads packages from URLs in composer.lock
  - Packagist.org is metadata only
  - *Open-source dependencies could come from anywhere*

- **Solutions to unavailability**
  - Composer cache in ~/.composer/cache
    - Unreliable, not intended for this use
  - Fork every dependency
    - huge maintenance burden
  - Your own Composer repository mirroring all packages
    - e.g. Private Packagist
composer install performance

- Use `--prefer-dist` to avoid git clones
  - Will always download zip files if possible (default for stable versions)

- Store `~/.composer/cache` between builds
  - How depends on CI product/setup you use
Autoloader Optimization

- composer install --optimize-autoloader
  - composer dump-autoload --optimize
- composer install --optimize-autoloader --classmap-authoritative
  - composer dump-autoload --optimize --classmap-authoritative
- composer install --optimize-autoloader --apcu-autoloader
  - composer dump-autoloader --optimize --apcu

https://getcomposer.org/doc/articles/autoloader-optimization.md
Autoloader Optimization

- Use this one
  
  ```bash
  composer dump-autoload --optimize --classmap-authoritative
  ```

- Requires PHP7 to be optimal
  - opcache can keep static array definition in shared memory
  - no loading overhead on PHP request startup

- Will not search for classes not in lookup table
  - not useful for development
  - not useful for dynamically generated code (don’t do that!)
It’s 2018 - What’s new in Composer?

- Current version: 1.6.5 (released May 4, 2018)
  - 22 releases since January 2017

- Bugfixes & Performance Improvements
  - Over 900 issues closed since January 2017 (~250 open)
  - Over 300 pull requests closed since January 2017 (~25 open)
    - Not all bug reports / bugfixes, feature requests, support issues, etc.
It’s 2018 - What’s new in Composer?

- Interoperability
  - GitLab API v4
    - released in 1.5.0 in August 2017
  - Bitbucket API v2
    - released in v1.4.0 in March 2017
  - New Git versions
    - v1.4.3 in August 2017
- Upcoming: GitHub deprecated Services
  - GitHub App for packagist.org
It’s 2018 - What’s new in Composer?

- New features
  - usually very small things
  - often not useful for everyone

- Let’s look at a couple
New Features

SPDX 3.0
License Identifier Update

GPL2.0 => GPL2.0-only
GPL2.0+ => GPL2.0-or-later

Packagist now rejects updates with invalid license identifiers now

https://github.com/composer/spdx-licenses
New Features

--with-all-dependencies

Released in 1.6.0, Jan 2018
Partial Updates

```json
{
    "name": "zebra/zebra",
    "require": {
        "horse/horse": "^1.0"
    }
}

{
    "name": "giraffe/giraffe",
    "require": {
        "duck/duck": "^1.0"
    }
}
```
Partial Updates

```json
{
  "name": "horse/horse",
  "require": {
    "giraffe/giraffe": "^1.0"
  }
}

{
  "name": "duck/duck",
  "require": {}
}
```
Partial Updates

```json
{
  "name": "my-project",
  "require": {
    "zebra/zebra": "^1.0",
    "giraffe/giraffe": "^1.0"
  }
}
```
Partial Updates

Now each package releases 1.1
Partial Updates

$ composer update --dry-run zebra/zebra
Updating zebra/zebra (1.0 -> 1.1)
Partial Updates

$ composer update --dry-run zebra/zebra --with-dependencies
Updating horse/horse (1.0 -> 1.1)
Updating zebra/zebra (1.0 -> 1.1)
Partial Updates

$ composer update --dry-run zebra/zebra giraffe/giraffe
  Updating zebra/zebra (1.0 -> 1.1)
  Updating giraffe/giraffe (1.0 -> 1.1)
Partial Updates

Project

- zebra 1.1
- giraffe 1.1
- horse 1.1
- duck 1.1

$ composer update zebra/zebra giraffe/giraffe --with-dependencies

  Updating duck/duck (1.0 -> 1.1)
  Updating giraffe/giraffe (1.0 -> 1.1)
  Updating horse/horse (1.0 -> 1.1)
  Updating zebra/zebra (1.0 -> 1.1)
Partial Updates

$ composer update zebra/zebra --with-all-dependencies
  Updating duck/duck (1.0 -> 1.1)
  Updating giraffe/giraffe (1.0 -> 1.1)
  Updating horse/horse (1.0 -> 1.1)
  Updating zebra/zebra (1.0 -> 1.1)
Partial Updates

$ composer update zebra/zebra --with-dependencies
  Updating horse/horse (1.0 -> 1.1)
  Updating zebra/zebra (1.0 -> 1.1)
$ composer update zebra/zebra --with-all-dependencies
  Updating duck/duck (1.0 -> 1.1)
  Updating giraffe/giraffe (1.0 -> 1.1)
  Updating horse/horse (1.0 -> 1.1)
  Updating zebra/zebra (1.0 -> 1.1)
Best Practice: CI for Libraries

- Multiple runs
  - `composer install` from lock file
  - `composer update` for latest deps
  - `composer update --prefer-lowest --prefer-stable` for oldest (stable) deps

- Potentially multiple composer.json files with different platform configurations
  - `COMPOSER=composer-customer1.json php composer.phar update`
  - `COMPOSER=composer-customer1.json php composer.phar install`

- Don’t use this except for testing - you’ll ruin our wonderful world where every PHP library can be installed with a plain composer install
Best Practice: Semantic Versioning

Promise of Compatibility

\textbf{X.Y.Z}

- Must be used consistently
  - Dare to increment \textbf{X}!
- Only valuable if BC/Compatibility promise formalized
  - See \url{http://symfony.com/doc/current/contributing/code(bc).html}
  - Document in Changelog
Versions Constraints

- **Exact Match:**
  - 1.0.0
  - 1.2.3-beta2
  - dev-master

- **Wildcard Range:**
  - 1.0.*
  - 2.*

- **Hyphen Range:**
  - 1.0-2.0
  - 1.0.0 - 2.1.0
  - >=1.0.0 <2.1
  - >=1.0.0 <=2.1.0

- **(Unbounded Range:**
  - >= 1.0)
  - Bad!

- **Next Significant Release**
  - ~1.2
  - >=1.2.0 <2.0.0
  - >=1.2.3 <2.3.0

- **Caret/Semver Operator**
  - ^1.2
  - ^1.2.3
  - >=1.2.0 <2.0.0
  - >=1.2.3 <2.0.0
  - **Best Choice for Libraries**

Operatoren: " " AND, "||" OR
Stabilities

- **Order**
  dev -> alpha -> beta -> RC -> stable

- **Automatically from tags**
  
  - 1.2.3 -> stable
  - 1.3.0-beta3 -> beta

- **Automatically from branches**
  
<table>
<thead>
<tr>
<th>Branch</th>
<th>Version (Stability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.0.x-dev (dev)</td>
</tr>
<tr>
<td>master</td>
<td>dev-master (dev)</td>
</tr>
<tr>
<td>myfeature</td>
<td>dev-myfeature (dev)</td>
</tr>
</tbody>
</table>

- **Choosing**
  
  - "foo/bar": "1.3.*@beta"
  - "foo/bar": "2.0.x-dev"

  "minimum-stability": "alpha"
$ php composer.phar validate

./composer.json is valid for simple usage with composer but has strict errors that make it unable to be published as a package:
See https://getcomposer.org/doc/04-schema.md for details on the schema
name : The property name is required
description : The property description is required
require.composer/composer : unbound version constraints (dev-master) should be avoided

Common: Version entry in composer.json conflicts with tag

$ php composer.phar self-update

$ php composer.phar update -vvv
Resolution Conflicts: Overly Strict Requirements

// composer.json

"require": {
  "cool/alice": "~1.3",
  "lazy/bob": "~1.2"
}

// dependencies

"name": "cool/alice",
"require": {
  "monolog/monolog": "~1.6"
}

"name": "lazy/bob",
"require": {
  "monolog/monolog": "1.3.*"
Resolution Conflicts: Overly Strict Requirements

Your requirements could not be resolved to an installable set of packages.

Problem 1
- Installation request for lazy/bob ~1.2 -> satisfiable by lazy/bob[1.4.0].
- Installation request for cool/alice ~1.3 -> satisfiable by cool/alice[1.3.0].
- lazy/bob 1.4.0 requires monolog/monolog 1.3.* -> satisfiable by monolog/monolog[1.3.0, 1.3.1].
- cool/alice 1.3.0 requires monolog/monolog ~1.6 -> satisfiable by monolog/monolog[1.6.0, 1.7.0].
- Can only install one of: monolog/monolog[1.6.0, 1.3.0].
- Can only install one of: monolog/monolog[1.6.0, 1.3.1].
- Conclusion: don't install monolog/monolog 1.3.1
- Conclusion: don't install monolog/monolog 1.7.0
- Conclusion: don't install monolog/monolog 1.3.0
- Conclusion: don't install monolog/monolog 1.6.0
Resolution Conflicts: Overly Strict Requirements

// composer.json

"require": {
    "cool/alice": "~1.3",
    "lazy/bob": "~1.2"
}

// dependencies

"name": "cool/alice",
"require": {
    "monolog/monolog": "~1.6"
}

"name": "lazy/bob",
"require": {
    "monolog/monolog": "1.3.*"
Resolution Conflicts: Stabilities

// composer.json

"minimum-stability": "beta",
"require": {
    "monolog/monolog": "1.*",
    "symfony/symfony": "~2.4",
    "bad/package": "dev-master"
}

// dependencies

"name": "bad/package",
"require": {
    "monolog/monolog": "dev-master",
}
Resolution Conflicts: Stabilities

Your requirements could not be resolved to an installable set of packages.

Problem 1
- Installation request for bad/package dev-master -> satisfiable by bad/package[dev-master].
- bad/package dev-master requires monolog/monolog dev-master -> no matching package found.
Resolution Conflicts: Stabilities

// composer.json

"minimum-stability": "beta",
"require": {
    "monolog/monolog": "1.*",
    "symfony/symfony": "~2.4",
    "bad/package": "dev-master"
}

// dependencies

"name": "bad/package",
"require": {
    "monolog/monolog": "dev-master",
}
// composer.json

"minimum-stability": "beta",
"require": {
    "monolog/monolog": "1.*@dev",
    "symfony/symfony": "~2.4",
    "bad/package": "dev-master"
}

// dependencies

"name": "bad/package",
"require": {
    "monolog/monolog": "dev-master",
}
Resolution Conflicts: Stabilities

// monolog

"name": "monolog/monolog",
"extra": {
    "branch-alias": {
        "dev-master": "1.12.x-dev"
    }
}

- Installing monolog/monolog (dev-master 5ad421d)
  Cloning 5ad421d6a1d5d7066a45b617e5164d309c4e2852
// monolog

"name": "monolog/monolog",
"extra": {
    "branch-alias": {
        "dev-master": "2.0.x-dev"
    }
}

Resolution Conflicts: Stabilities
Resolution Conflicts: Stabilities

Your requirements could not be resolved to an installable set of packages.

Problem 1
- Installation request for monolog/monolog 1.*@dev -> satisfiable by monolog/monolog[1.12.0].
- Installation request for bad/package dev-master -> satisfiable by bad/package[dev-master].
- bad/package dev-master requires monolog/monolog dev-master -> satisfiable by monolog/monolog[dev-master].
- Can only install one of: monolog/monolog[1.12.0, dev-master].

We require “2.*@dev” instead
- Resolution works
- Project is probably broken:
  bad/package may not be compatible with 2.*
No error but unexpected result?

- composer why [--tree] foo/bar
  mydep/here 1.2.3 requires foo/bar (^1.0.3)

- composer why-not [--tree] foo/bar ^1.2
  foo/bar 1.2.3 requires php (>=7.1.0 but 5.6.3 is installed)
Application/Project Versioning

- There are no other packages depending on yours?
  - BC - for Composer consumption - doesn’t matter
- Options:
  - Don’t use versions at all, rely on your VCS
  - Increment a single integer
  - Use semver if you ship the application
The Lock file will conflict
Day 0: “Initial Commit”

```
Project
zebra 1.0
- zebra 1.0
- giraffe 1.0

dna-upgrade
- zebra 1.0
- giraffe 1.0
```
Week 2: Strange new zebras require duck

Project
- zebra 1.1
- giraffe 1.0
- duck 1.0

master
- composer.lock
  - zebra 1.1
  - giraffe 1.0
  - duck 1.0

dna-upgrade
- composer.lock
  - zebra 1.0
  - giraffe 1.0

Nils Adermann
@naderman
Week 3: Duck 2.0
Week 4: Giraffe evolves to require duck 2.0
Text-based Merge

Merge results in invalid dependencies

```
master
composer.lock
- zebra 1.1
- giraffe 1.2
- duck 1.0
- duck 2.0
```
Reset composer.lock

git checkout <refspec> -- composer.lock

Project
  ↓
zebra 1.1
  ↓
duck 1.0
  ↓
giraffe 1.0

DNA-upgrade

composer.lock
- zebra 1.1
- giraffe 1.0
- duck 1.0
Apply the update again

```
composer update giraffe --with-dependencies
```

```
- zebra 1.1
- giraffe 1.2
- duck 2.0
```
How to resolve lock merge conflicts?

- composer.lock cannot be merged without conflicts
  - contains hash over relevant composer.json values

- git checkout <refspec> -- composer.lock
  - git checkout master -- composer.lock

- Reapply changes
  - composer update <list of deps>
New Features

check-platform-reqs

Released in 1.6.0, Jan 2018
Platform Requirements

- Platform repository
  - implicitly defined additional package repository
  - contains packages for
    - PHP
    - extensions
    - system libraries (e.g. libxml)
  - packages cannot be updated/installed/removed
Platform Requirements

$ ./composer.phar show --platform

composer-plugin-api 1.1.0 The Composer Plugin API
ext-apcu 5.1.8 The apcu PHP extension
ext-ctype 7.2.5 The ctype PHP extension
ext-curl 7.2.5 The curl PHP extension
ext-date 7.2.5 The date PHP extension
ext-dom 20031129 The dom PHP extension
ext-fileinfo 1.0.5 The fileinfo PHP extension
ext-filter 7.2.5 The filter PHP extension
ext-ftp 7.2.5 The ftp PHP extension
ext-hash 1.0 The hash PHP extension
ext-icnv 7.2.5 The icnv PHP extension
ext-intl 1.1.0 The intl PHP extension
ext-json 1.6.0 The json PHP extension
ext-libxml 7.2.5 The libxml PHP extension
lib-curl 7.59.0 The curl PHP library
lib-ICU 58.2 The intl PHP library
lib-libxml 2.9.5 The libxml PHP library
lib-openssl 2.5.5 LibreSSL 2.5.5
lib-pcre 8.41 The pcre PHP library
php 7.2.5 The PHP interpreter
php-64bit 7.2.5 The PHP interpreter, 64bit
php-ipv6 7.2.5 The PHP interpreter, with IPv6 support
Platform Requirements

```
{
    "require": {
        "php": "^7.1.1"
    }
}
```

$ php -v
PHP 5.6.10

$ composer update

Your requirements could not be resolved to an installable set of packages.

Problem 1
- This package requires php ^7.1.1 but your PHP version (5.6.10) does not satisfy that requirement.
Platform Requirements

- What if you maintain multiple projects on your local system to be deployed to different platforms?
  - e.g. Server A running PHP 7.0, Server B running PHP 7.2
- What if you want to build Composer automation tools
  - Private Packagist at packagist.com runs on a single PHP version, managed projects have lots of different requirements
Platform Requirements

```
{
    "require": {
        "php": "^7.1.1"
    }
}
```

$ php -v
PHP 5.6.10

$ composer update --ignore-platform-reqs
Success

No idea if dependencies even work on PHP 7.1.1
Platform Requirements

```
"require": {
    "php": "^7.1.1",
    "ext-intl": "*"
}
"config": {
    "platform": {
        "php": "7.1.2",
        "ext-intl": "1.1.0"
    }
}
```

$ php -v
PHP 5.6.10

$ composer update
Success
Platform Requirements

- Watch out if you are using Plugins!
  - Composer plugins (Composer installers are plugins, too)
    - Packages with type “composer-plugin”
    - Will be installed before all other packages if dependencies allow it
    - Code will be executed in Composer process during update/install
  - Can be disabled with **--no-plugins**
  - no easy way to run them on prod later

- Watch out if you are using scripts
  - Use **--no-scripts**
  - Run them separately in production with `composer run-script <name>`
Platform Requirements

```
"require": {
    "php": "^7.1.1",
    "ext-intl": "*
}
"config": {
    "platform": {
        "php": "7.1.2",
        "ext-intl": "1.1.0"
    }
}
```

$ composer update

Success

- Create ZIP
- deploy to prod

**PHP Fatal Error**

Prod was actually still on PHP 5.6
Platform Requirements

```
"require": {
    "php": "^7.1.1",
    "ext-intl": "*
}
"config": {"platform":{
    "php": "7.1.2",
    "ext-intl": "1.1.0"
}}
```

- dev$ composer update
- Create ZIP
- upload to prod
- composer check-platform-reqs
  - no error? switch to new code
Summary

- composer show --platform
  `{"config":{"platform":{"php":"7.2.5"}}}`
- composer check-platform-reqs
  Watch out for plugins & scripts!
- composer install --prefer-dist
- Create a build artifact and do as little work in prod as possible
- composer dump-autoload --optimize
  --classmap-authoritative
- Update your license identifiers to SPDX 3.0
- SemVer: Don’t be afraid to increase the major version
- Library CI: composer update
  --prefer-lowest --prefer-stable
- composer update <package>
  --with-all-dependencies
- git checkout <branch> -- composer.lock
  && repeat composer update
Thank you!

Questions / Feedback?
https://joind.in/talk/fee50

https://packagist.com
10% off first 12 months with code phptek2018

E-Mail: n.adermann@packagist.com
Twitter: @naderman