Composer Best Practices *2018*

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

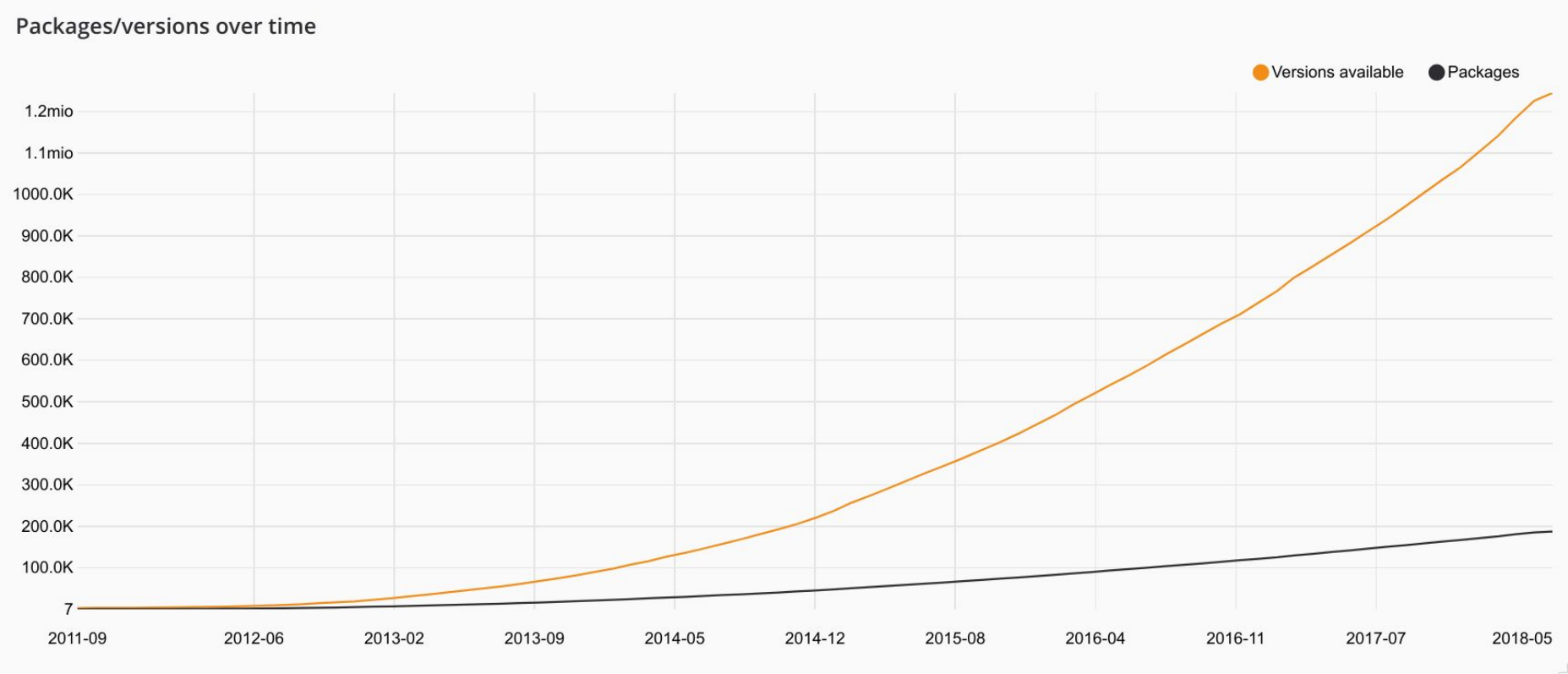
2018?

Delete your lock files

2018?

~~Delete your lock files~~

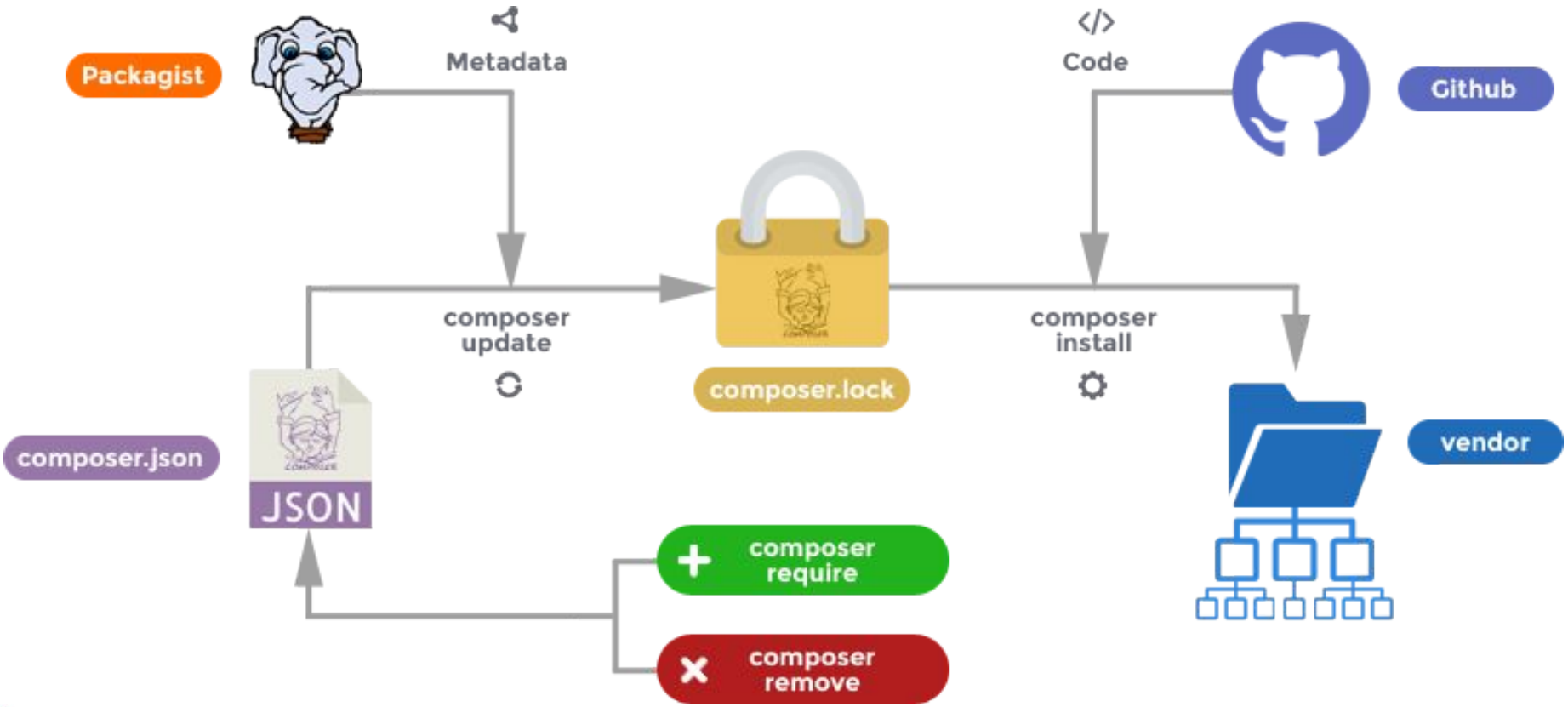
Composer Ecosystem Reality Update 2018



PRIVATE PACKAGIST

Nils Adermann
@naderman

Best Practices?



Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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-autoload --optimize --apcu`

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

Nils Adermann
@naderman

Autoloader Optimization

- *Use this one*
  `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!)

Nils Adermann
@naderman

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.

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

Nils Adermann
@naderman

New Features

`--with-all-dependencies`

Released in 1.6.0, Jan 2018

Nils Adermann
@naderman

Partial Updates

```
{   “name”: “zebra/zebra”,
    “require”: {
        “horse/horse”: “^1.0” }}

{   “name”: “giraffe/giraffe”,
    “require”: {
        “duck/duck”: “^1.0” }}
```


Nils Adermann
@naderman

Partial Updates

```
{   “name”: “horse/horse”,
    “require”: {
        “giraffe/giraffe”: “^1.0” }}

{   “name”: “duck/duck”,
    “require”: {}}
```

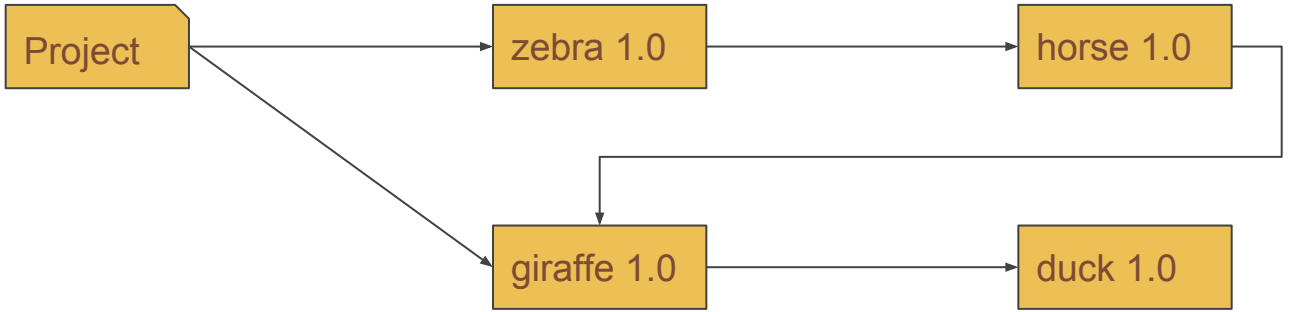

Nils Adermann
@naderman

Partial Updates

```
{
    “name”: “my-project”,
    “require”: {
        “zebra/zebra”: “^1.0”,
        “giraffe/giraffe”: “^1.0”
    }
}
```


Nils Adermann
@naderman

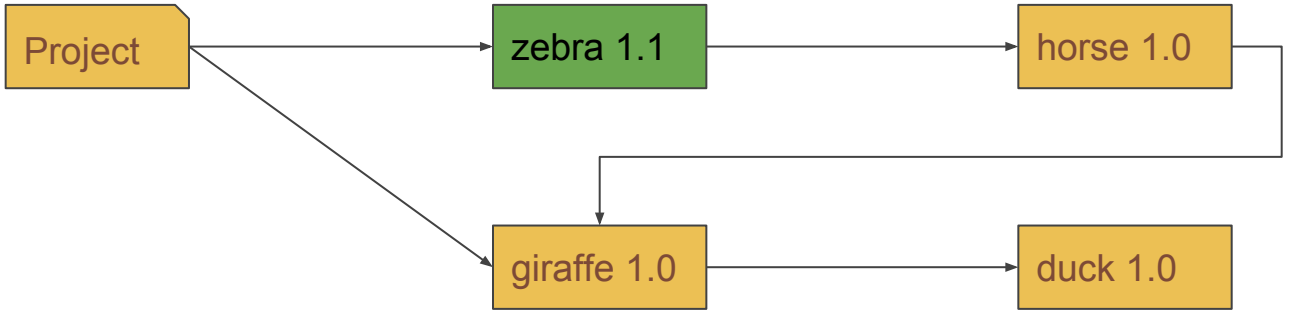
Partial Updates



Now each package releases 1.1

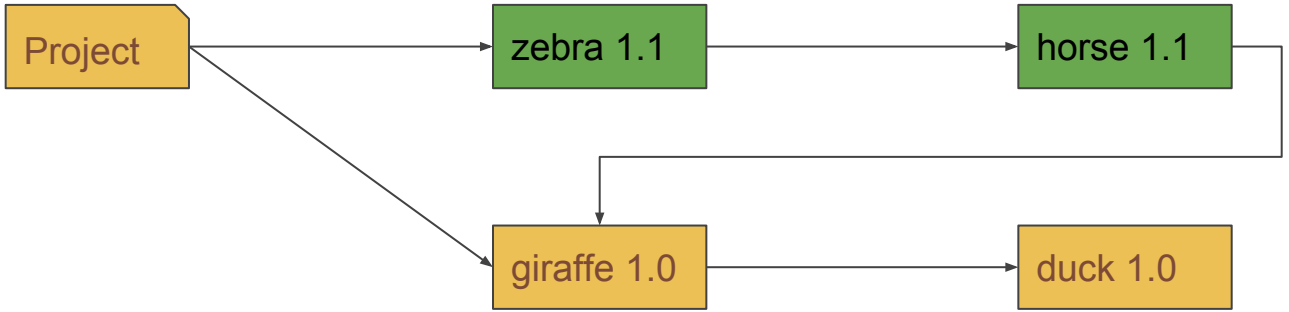
Nils Adermann
@naderman

Partial Updates




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

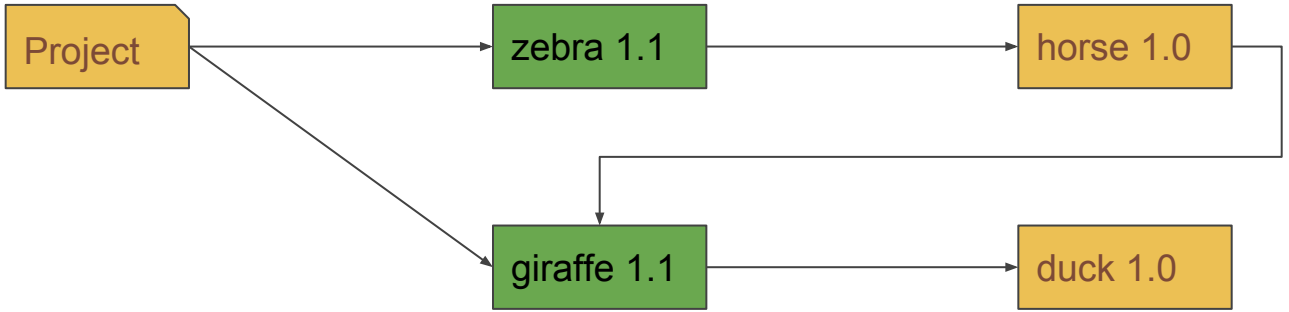
Partial Updates




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


Nils Adermann
@naderman

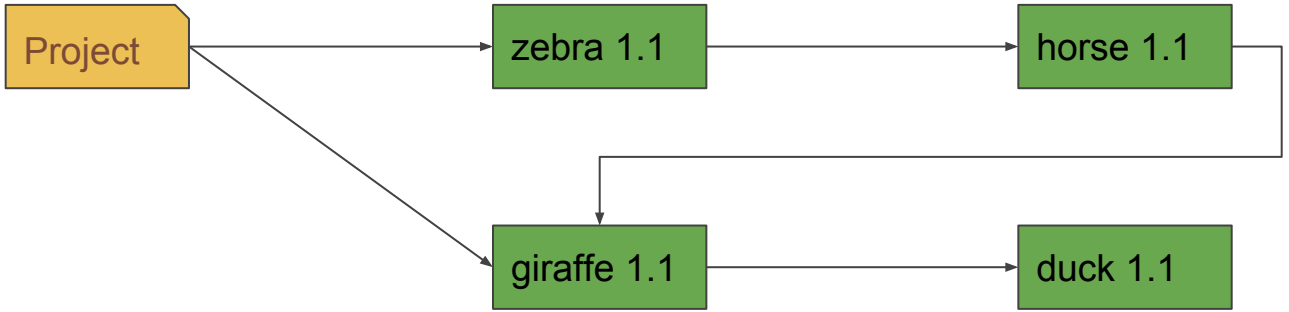
Partial Updates




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

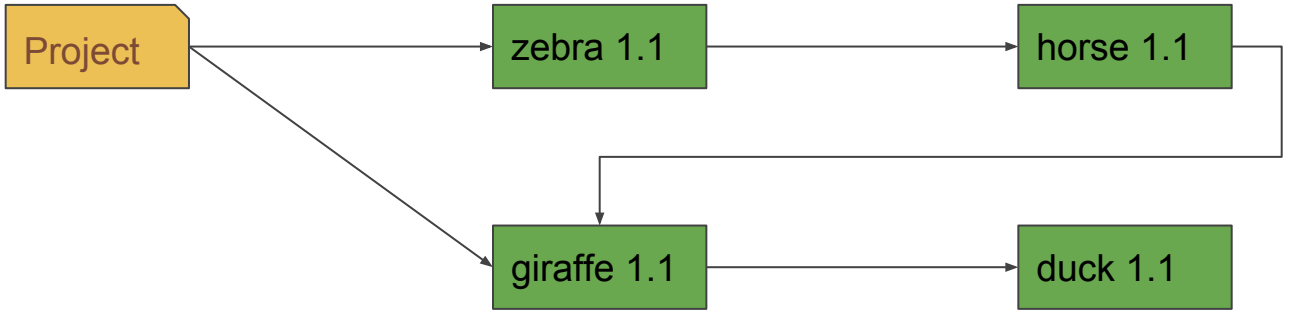

Nils Adermann
@naderman

Partial Updates




```
$ 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)
```

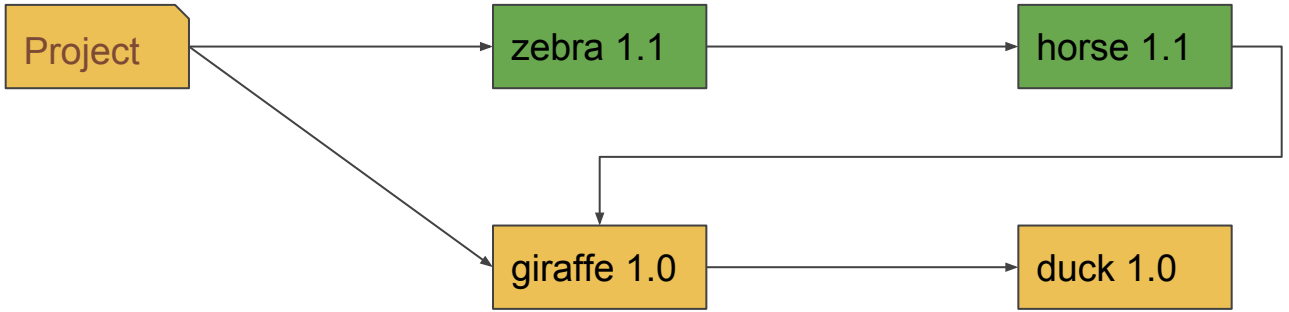
Partial Updates




```
$ 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)
```


Nils Adermann
@naderman

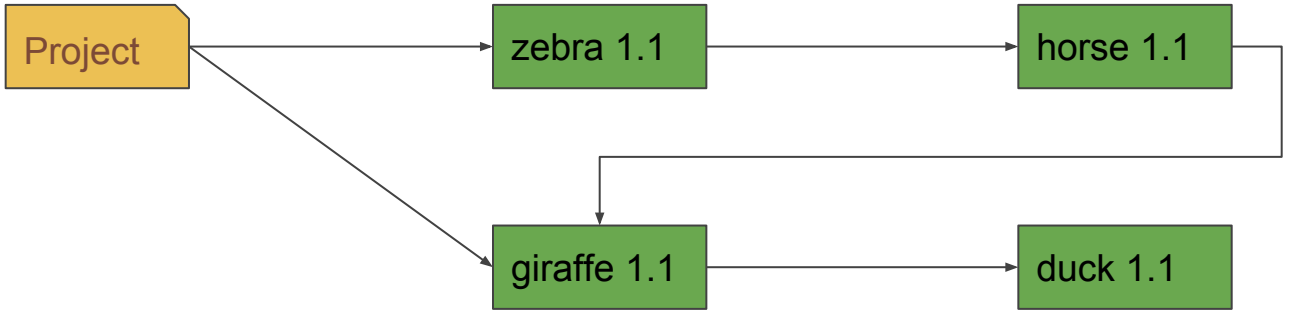
Partial Updates




```
$ composer update zebra/zebra --with-dependencies
    Updating horse/horse (1.0 -> 1.1)
    Updating zebra/zebra (1.0 -> 1.1)
```


Nils Adermann
@naderman

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)
```


Nils Adermann
@naderman

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

Nils Adermann
@naderman

Best Practice: Semantic Versioning

Promise of Compatibility

X.Y.Z

- Must be used consistently
  Dare to increment **X**!
- Only valuable if BC/Compatibility promise formalized
  - See http://symfony.com/doc/current/contributing/code/bc.html
  - Document in Changelog

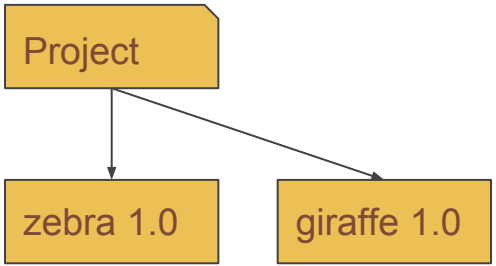
Nils Adermann
@naderman

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

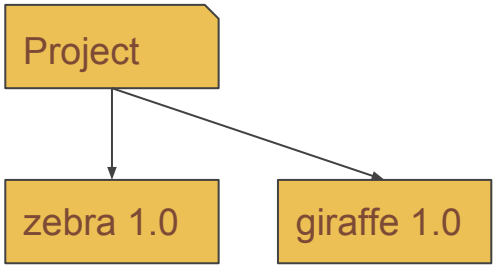
The Lock file will conflict

Day 0: “Initial Commit”



master

composer.lock
- zebra 1.0
- giraffe 1.0

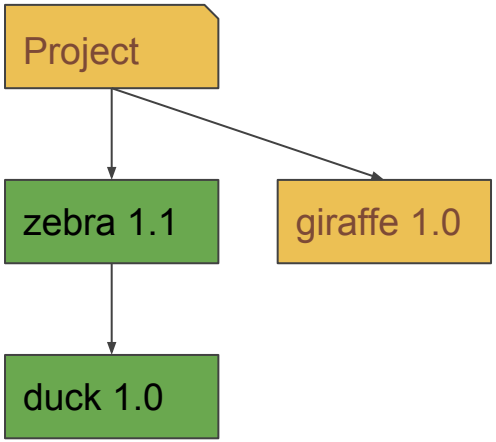


dna-upgrade

composer.lock
- zebra 1.0
- giraffe 1.0

Nils Adermann
@naderman

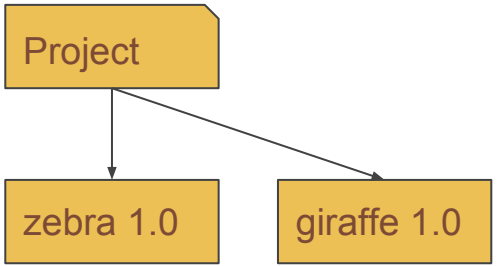
Week 2: Strange new zebras require duck



master

composer.lock

- zebra	1.1
- giraffe	1.0
- duck	1.0



dna-upgrade

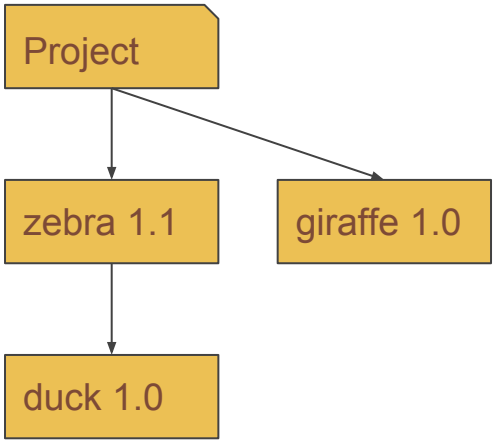
composer.lock

- zebra	1.0
- giraffe	1.0

Nils Adermann
@naderman


Week 3: Duck 2.0

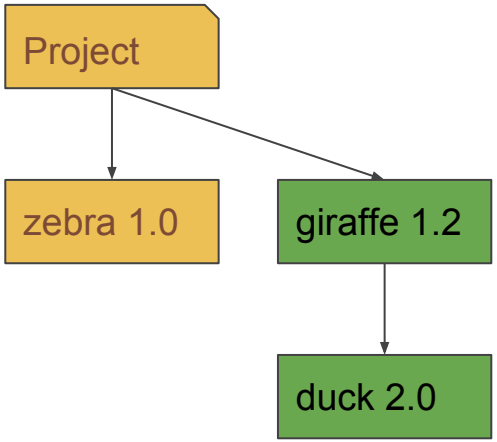
Week 4: Giraffe evolves to require duck 2.0



master

composer.lock

- zebra	1.1
- giraffe	1.0
- duck	1.0



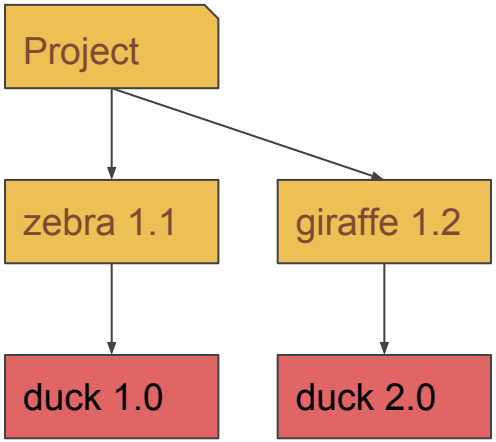
dna-upgrade

composer.lock

- zebra	1.0
- giraffe	1.2
- duck	2.0

Nils Adermann
@naderman

Text-based Merge

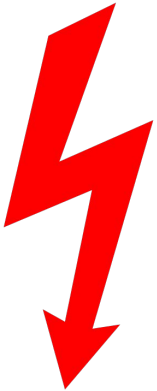


master

composer.lock

- zebra 1.1
- giraffe 1.2
- **duck 1.0**
- **duck 2.0**

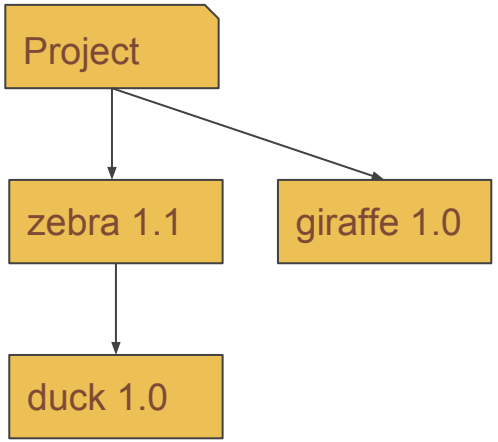
Merge results in invalid dependencies



Nils Adermann
@naderman

Reset composer.lock

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



dna-upgrade

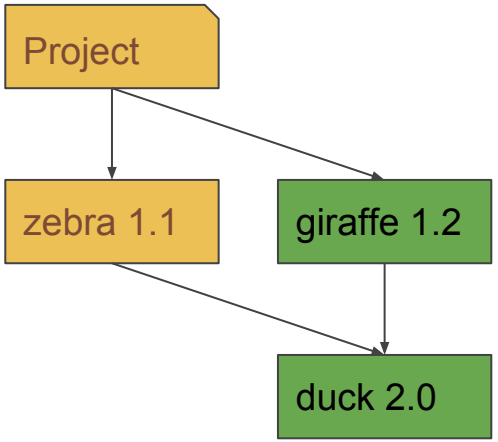
composer.lock

- zebra	1.1
- giraffe	1.0
- duck	1.0

Nils Adermann
@naderman

Apply the update again

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



master

composer.lock

- zebra	1.1
- giraffe	1.2
- duck	2.0

Nils Adermann
@naderman

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>`

Nils Adermann
@naderman

New Features

`--check-platform-reqs`

Released in 1.6.0, Jan 2018

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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-iconv           7.2.5    The iconv 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
```


Nils Adermann
@naderman

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.
```


Nils Adermann
@naderman

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

Nils Adermann
@naderman

Platform Requirements

```
{
    “require”: {
        “php”: “^7.1.1”
    }
}
```

```
$ php -v
PHP 5.6.10

$ composer update --ignore-platform-reqs
```

No idea if dependencies even work on PHP 7.1.1

Nils Adermann
@naderman

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
```

PRIVATE PACKAGIST

Nils Adermann
@naderman

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

Nils Adermann
@naderman

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

PRIVATE PACKAGIST

Nils Adermann
@naderman

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

Nils Adermann
@naderman

Thank you!

Questions / Feedback?

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