Abhängigkeiten in PHP Projekten mit Composer verwalten

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

Wie installiert man eine PHP Anwendung?

- Bevor es los geht: Der alte Weg
  - README / PDF / Wiki
    - Betriebssystempakete installieren (z.B. mod_rewrite)
    - Webserver konfigurieren (Dateien kopieren, Einträge anlegen)
    - PHP Extensions installieren (z.B. imagick)
    - PHP Extension abschalten (z.B. mbstring)
    - PHP konfigurieren
  - Oder

    Hoffen dass Hoster die richtigen Dinge installiert / Hoster um Installation bitten

Nils Adermann
@naderman

Wie installiert man eine PHP Anwendung?

- Bevor es los geht: Der neue Weg
  - Configuration Management (puppet, chef, ansible, salt, …)
  - Programm stellt Systemzustand automatisch korrekt her

- Vorteile:
  - weniger manuelle Fehler
  - leicht wiederherstellbar
    - z.B. Server kaputt, weitere Server, lokale Staging Umgebung

Nils Adermann
@naderman

Wie installiert man eine PHP Anwendung?

- Der alte Weg
  - README / PDF / Wiki
  - Alle Bibliotheken/Abhängigkeiten im gleichen Repository

- Nachteile
  - Identifikation von Versionen durch manuelles (oft unvollständiges) dokumentieren
  - Möglicherweise manuell gepatchte Abhängigkeiten
  - Updates schwierig

Nils Adermann
@naderman

Wie installiert man eine PHP Anwendung?

PRIVATE PACKAGIST

Nils Adermann
@naderman

Abhängigkeiten mit Composer verwalten

- Ein Befehl zum Herstellen des Systemzustands

```
composer install
```

- Ein Befehl zum (lokalen) Aktualisieren der Abhängigkeiten

```
composer update
```


Nils Adermann
@naderman

Composer Installation

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') ===
'669656bab3166a7aff8a7506b8cb2d1c292f042046c5a994c43155c0be6190fa0355160742ab2e1c88d40d5
be660b410') { echo 'Installer verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Überprüft aktuelle Checksumme
Deshalb von https://getcomposer.org/download/ installieren

Abhängigkeiten definieren

composer.json:

- Im Projekt-Root

```
{
    "require": {
        "silex/silex": "~2.0",
        "symfony/finder": "3.1-dev",
        "twig/twig": "1.*",
        "predis/service-provider": "dev-master"
        "symfony/console": "^3.2.2"
    },
    "require-dev": {
        "mikey179/vfsStream": "*"
    }
}
```


Nils Adermann
@naderman

Alternative Repositories

composer.json:

```
"repositories": [
    {
        "type": "composer",
        "url": "https://repo.packagist.com/my-org/"
    },
    {
        "type": "vcs",
        "url": "git://example.org/MyRepo.git"
    },
    {
        "packagist.org": false
    }
]
```


Nils Adermann
@naderman

Repository Authentication

`$ composer config --global --auth http-basic.repo.packagist.com naderman 1234-token`

Writes ***`$COMPOSER_HOME/auth.json`***

- `C:\Users\<user>\AppData\Roaming\Composer`
- `/Users/<user>/.composer`
- `$XDG_CONFIG_HOME/composer - usually /home/<user>/.config/composer`
- `/home/<user>/.composer`

`$COMPOSER_HOME/config.json`

- `process-timeout`
- `cache-files-ttl`
- `github-protocols`

Abhängigkeiten installieren

```
$ php composer.phar install

Installing from lock file
  - Package twig/extensions (dev-master)
    Downloading
    Unpacking archive
    Cleaning up

[...]

  - Package twig/twig (1.8.0)
    Downloading
    Unpacking archive
    Cleaning up

  - Package symfony/symfony (dev-master)
    Downloading
    Unpacking archive
    Cleaning up

Generating autoload files
```

Nils Adermann
@naderman

Abhängigkeiten verwenden

```
composer.json
composer.lock
vendor/
    autoload.php
    composer/
    monolog/
        monolog/
    symfony/
        symfony/
        monolog-bundle/
    twig/
        twig/
        extensions/
    [...]
```

web/index.php

```
require __DIR__.'/../vendor/autoload.php'
```

Pakete definieren

```
{
    "name": "predis/predis",
    "type": "library",
    "description": "Flexible and feature-complete Redis client",
    "keywords": ["nosql", "redis", "predis"],
    "homepage": "http://github.com/nrk/predis",
    "license": "MIT",
    "authors": [
        {
            "name": "Daniele Alessandri",
            "email": "suppakilla@gmail.com",
            "homepage": "http://clorophilla.net"
        }
    ],
    "require": {
        "php": ">=5.3.9"
    },
    "autoload": {
        "psr-4": {"Predis\\": "src/"}
    }
}
```

Autoloading für foo/bar

- **PSR-0**
  `{“psr-0”: {“Foo\\”: “src”}}`
  `new Foo\Bar\Baz\Qux - vendor/foo/bar/src/Foo/Bar/Baz/Qux.php`

- **PSR-4**
  `{“psr-4”: {“Foo\\Bar\\Baz\\”: “src”}}`
  `new Foo\Bar\Baz\Qux - vendor/foo/bar/src/Qux.php`

- **Classmap**
  `{“classmap”: [“src/”, “library/legacy/”]}`
  `new Foo\Bar\Baz\Qux - vendor/foo/bar/src/Some/Weird/Name.php`

- **Files**
  `{“files”: [“src/some/functions.php”]}`

Nils Adermann
@naderman

Autoloader Optimierung

```
$ composer dump-autoload --optimize
```

Wandelt PSR-0/4 in Classmap um.

```
$ composer dump-autoload --optimize --apcu
```

Wandelt PSR-0/4 in Classmap um, speichert Classmap in APCu (PHP Extension)

```
$ composer dump-autoload --optimize --classmap-authoritative
```

Wandelt PSR-0/4 in Classmap um, verlässt sich auf opcache, neue Klassen werden nicht gefunden.

Nils Adermann
@naderman

Semantic Versioning

x.y.z

(BC-break).(new functionality).(bug fix)

http://semver.org/

http://symfony.com/doc/current/contributing/code/bc.html

Nils Adermann
@naderman

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	1.0.0 - 2.1.0 >=1.0.0 <=2.1.0	
- *(Unbounded Range: Problematisch!*	*>= 1.0)*		
- **Next Significant Release**	~1.2 >=1.2.0 <2.0.0	~1.2.3 >=1.2.3 <1.3.0	
- **Caret/Semver Operator**	^1.2 >=1.2.0 <2.0.0	^1.2.3 >=1.2.3 <2.0.0	**Beste Wahl für Bibliotheken**

Operatoren: “ “ AND, “||” OR

Nils Adermann
@naderman

Stabilitäten

- **Rangfolge** dev -> alpha -> beta -> RC -> stable
- **Erkennung über Tags**
  1.2.3 -> stable
  1.3.0-beta3 -> beta
- **Erkennung in Branches**
  Branch -> Version (Stabilität)
  2.0 -> 2.0.x-dev (dev)
  master -> dev-master (dev)
  myfeature -> dev-myfeature (dev)
- **Auswählen**

```
“foo/bar”: “1.3.*@beta”
“foo/bar”: “2.0.x-dev”

“minimum-stability”: “alpha”
```


Nils Adermann
@naderman

Die Lock Datei: **composer.lock**

- Inhalt
  - Alle Abhängigkeiten und deren Abhängigkeiten, etc. (transitive Abhängikeiten)
  - Exakte Version jedes Pakets
  - Download URLs (source, dist, mirror)

- Absicht
  - Reproduzierbarkeit im Team, für Benutzer und auf verschiedenen Systemen
  - Isolation von Bug Reports auf eigenen Code vs. Abhängigkeiten
  - Transparenz über den Update Prozess

Die Lock Datei: **composer.lock**

Die Lock Datei unbedingt mit einchecken und verteilen!

Ansonsten

- **composer install** führt ohne lock ein **composer update** aus
- Konflikte können während der Installation passieren
- Eventuell wird anderer Code installiert als geplant
- Man verliert die Kontrolle über die Abhängigkeiten

Die Lock Datei existiert um eingecheckt zu werden!

Nils Adermann
@naderman

Abhängigkeiten aktualisieren

- **`$ composer update`**
  - Alles auf einmal, keine Isolation
- **`$ composer update <package...>`**
  - Einzelne Pakete aktualisieren
- **`$ composer update --with-dependencies <package...>`**
  - Einzelne Pakete und ihre Abhängigkeiten
- **`$ composer update --dry-run [--with-dependencies <package...>]`**
  - Nichts ausführen, nur ausprobieren und anzeigen

Falls es einen Fehler gibt

```
$ 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

$ php composer.phar self-update

$ php composer.phar update -vvv
```


Nils Adermann
@naderman

Wie funktionieren partielle Updates?

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

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


Nils Adermann
@naderman

Wie funktionieren partielle Updates?

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

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

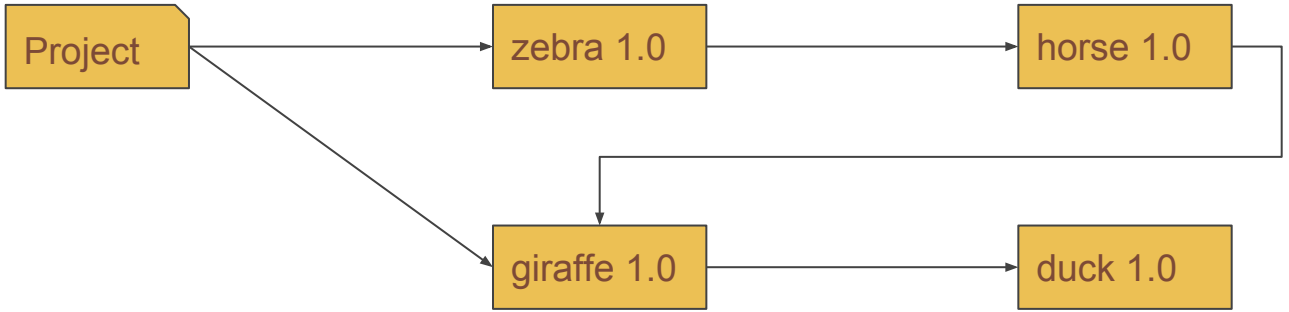

Nils Adermann
@naderman

Wie funktionieren partielle Updates?

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


Nils Adermann
@naderman

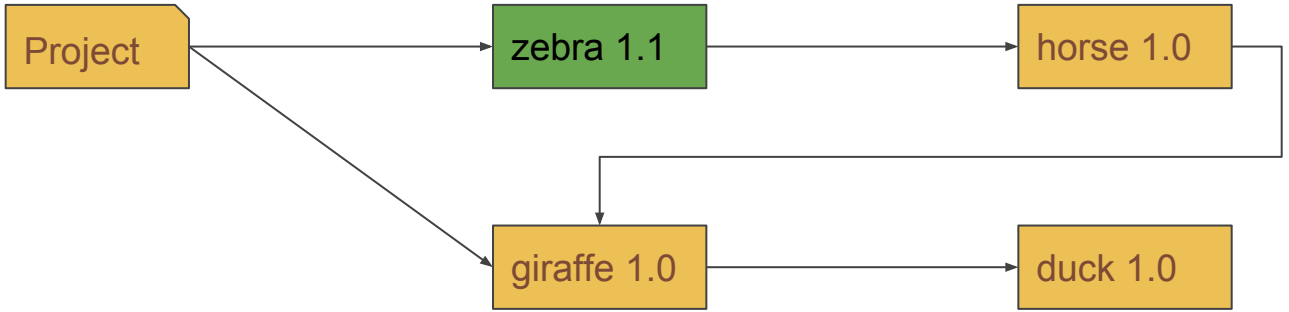
Wie funktionieren partielle Updates?



Now each package releases 1.1

Nils Adermann
@naderman

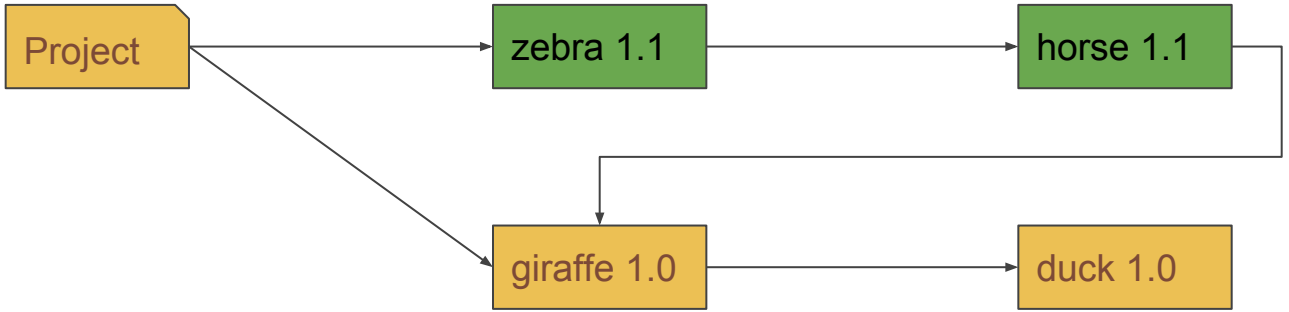
Wie funktionieren partielle Updates?




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


Nils Adermann
@naderman

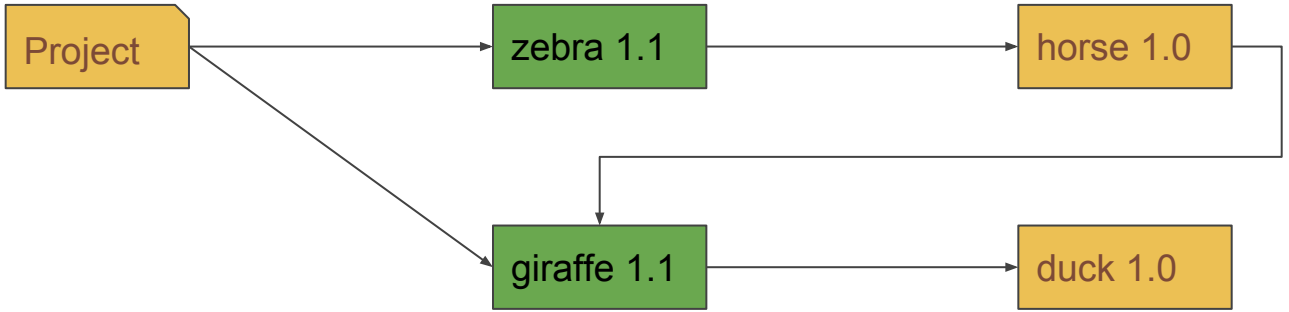
Wie funktionieren partielle Updates?




```
$ 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

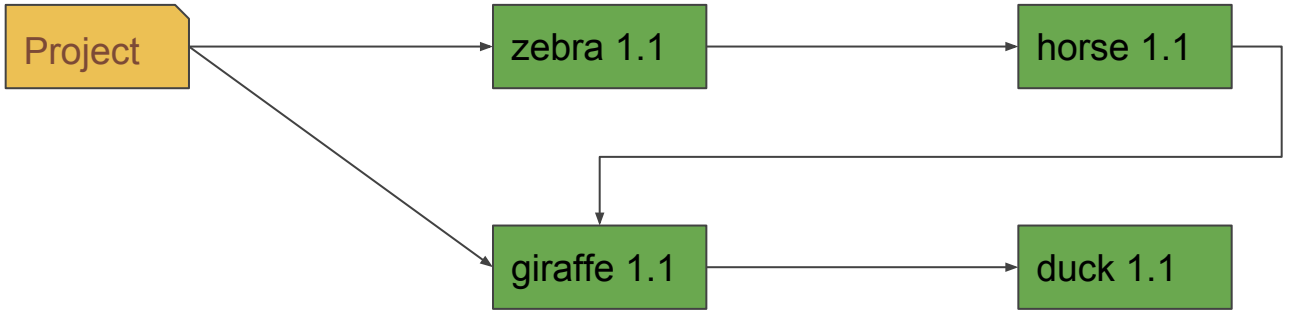
Wie funktionieren partielle Updates?




```
$ 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

Wie funktionieren partielle 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)
```

Forks / Aliase

```
{
    "repositories": [
        {
            "type": "vcs",
            "url": "https://github.com/naderman/symfony-console"
        }
    ],
    "require": {
        "symfony/symfony-console": "dev-my-patch as 3.3.1"
        "contao/core-bundle": "1.0.*"
    }
}
```


Nils Adermann
@naderman

Nützliche Befehle

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

- **`composer why-not foo/bar`**
  `foo/bar 1.2.3 requires php (>=7.1.0 but 5.6.3 is installed)`

- **`composer outdated`**
  Displays table of dependencies and available newer versions

- **`composer update --prefer-lowest, --prefer-stable`**
  Installiert ältest mögliche stabile Abhängigkeiten

Nils Adermann
@naderman

Vielen Dank!

Fragen / Feedback?

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