

Magia Composerera w Drupal 8

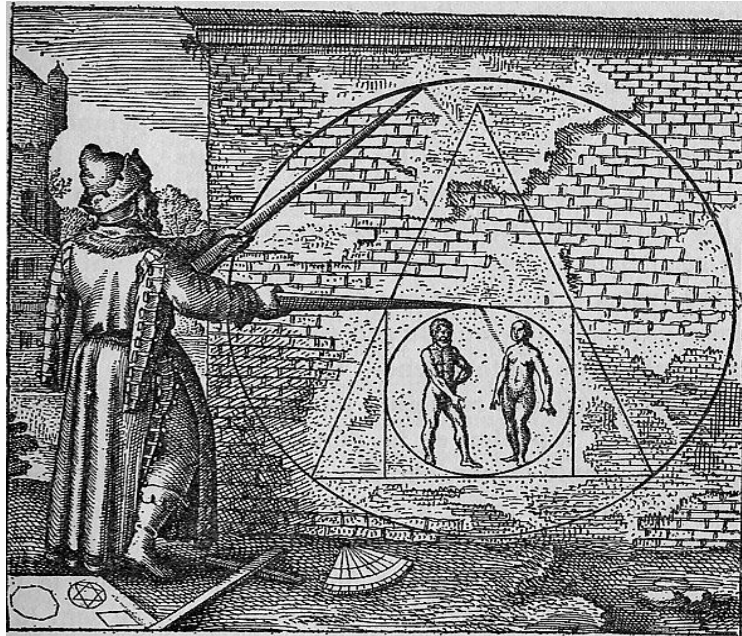
Prowadzący: Grzegorz Pietrzak



Magia?

Menadżer pakietów w PHP

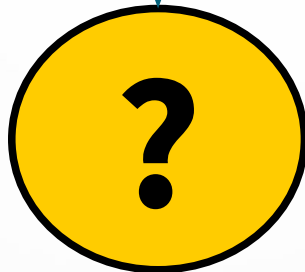
= kamień filozoficzny?



PHP/PEAR



ZEND, CakePHP (2005)
CodeIgniter (2006), Symfony 1.0 (2007)



Szybka aktualizacja?
Przejrzystość w VCS?
Automatyzacja?
Coś jak npm, bundler?

Ponowne użycie kodu?
Manager zależności?
Repozytorium pakietów?
Autoload?

“Developerzy PHP nienawidzą
pakietów.”

– **Phil Sturgeon, 2012**
“Packages: The Way Forward for PHP”

PEAR

COMPOSER

PYRUS

**DRUSH
MAKE**

**NPM,
BUNDLER,
...**

PEAR 2

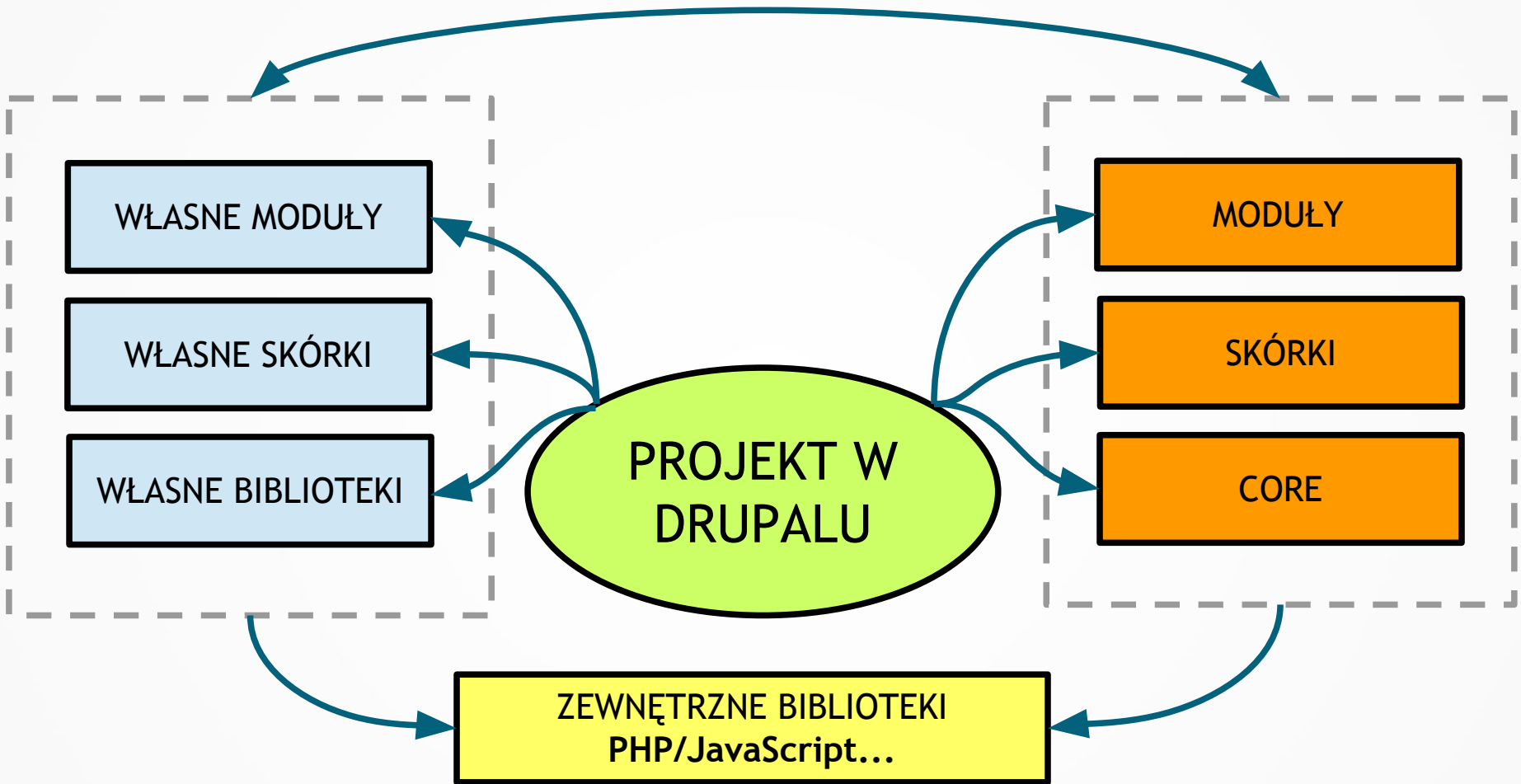


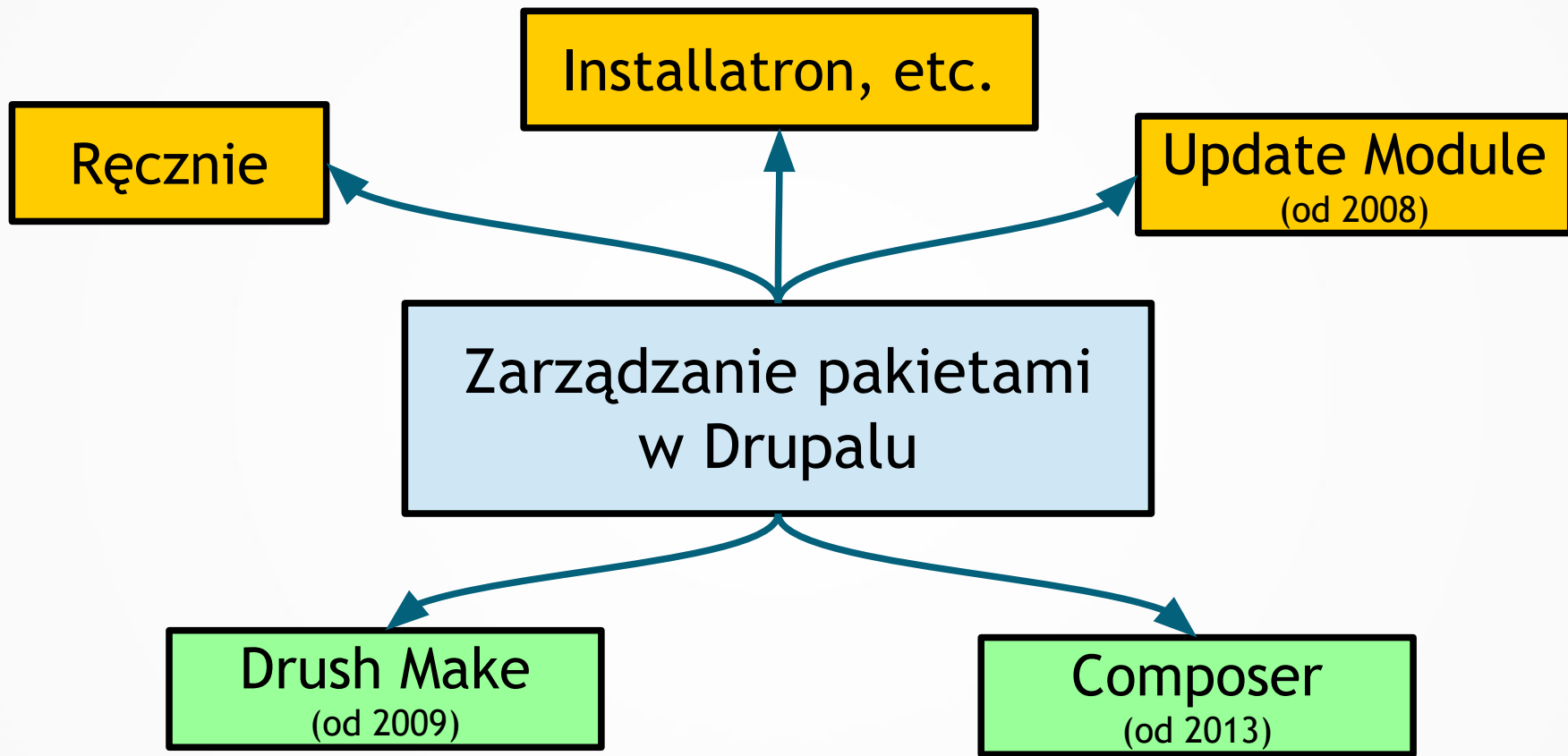
Nils Adermann
Jordi Boggiano
marzec 2012

COMPOSER



Pakiety w Drupalu?





Podstawy działania



Copyright © 2011 Nils Adermann, Jordi Boggiano

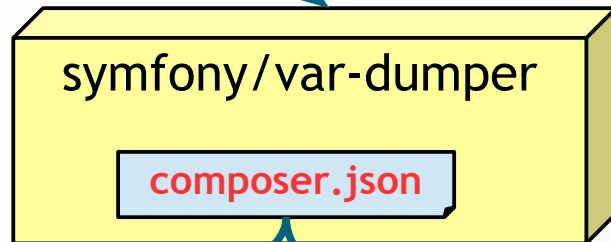
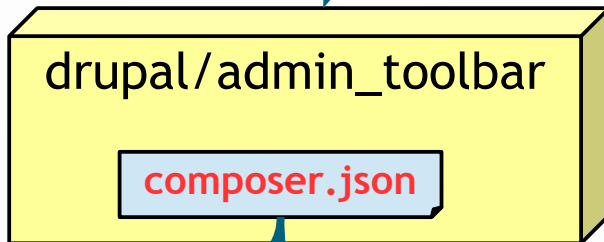
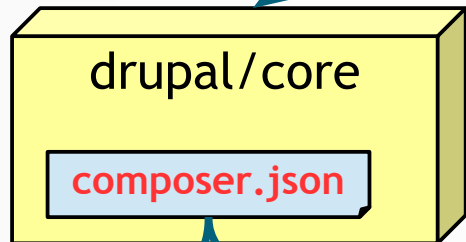
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The Software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the Software or the use or other dealings in the Software.

composer.json

```
"require": {  
  "drupal/core": "^8.3",  
  "drupal/admin_toolbar": "^1.19",  
  "symfony/var-dumper": "3.*"  
}
```



composer.phar

- Działa w CLI
- Napisany w PHP
- Wymaga PHP 5.3.2+
- Pojedynczy plik PHAR
- Dostępny instalator Windows
- Współpracuje z **packagist.org**

`https://getcomposer.org/download/`

Najprostszy workflow z Composerem

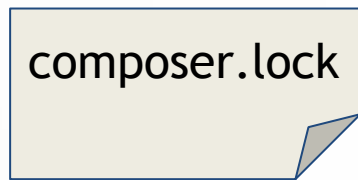
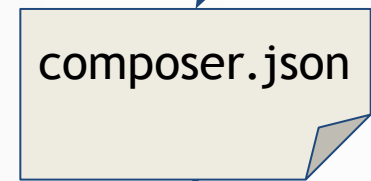
- 1) Tworzymy lub uaktualniamy **composer.json**
- 2) Wywołujemy polecenie `composer update` w środowisku DEV.
- 3) Wrzucamy pliki **composer.json** i **composer.lock** do VCS. Ignorujemy w VCS katalog z pakietami (`/vendor`).
- 4) Wywołujemy polecenie `composer install` w środowisku PROD.

DEV

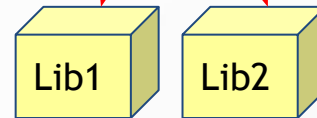
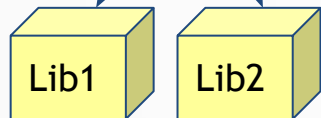
PROD

composer update

composer install



COMMIT

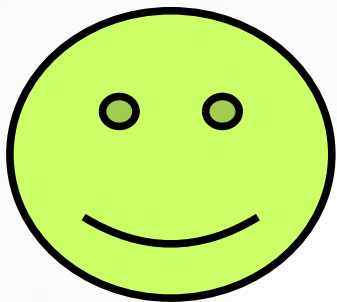


`composer self-update` - uaktualnia
Composera do najnowszej wersji.

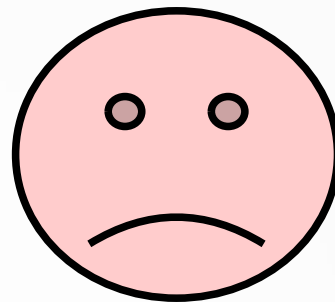
`composer outdated` - pokazuje listę pakietów
wymagających uaktualnienia

`composer require foo/bar` - dodaje pakiet
foo/bar do composer.json i wykonuje polecenie
update.

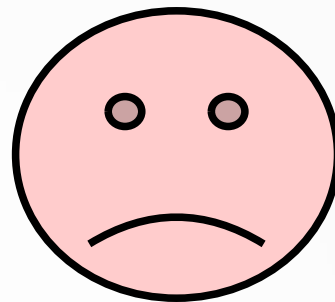
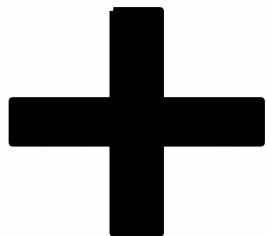
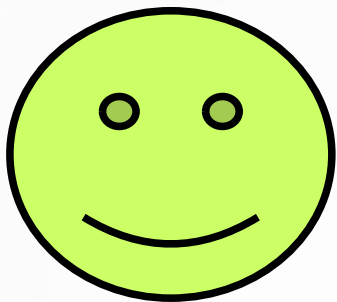
Composer + Drupal?



- Szybka aktualizacja
- Łatwa synchronizacja DEV i PROD
- Obsługa komponentów Symfony

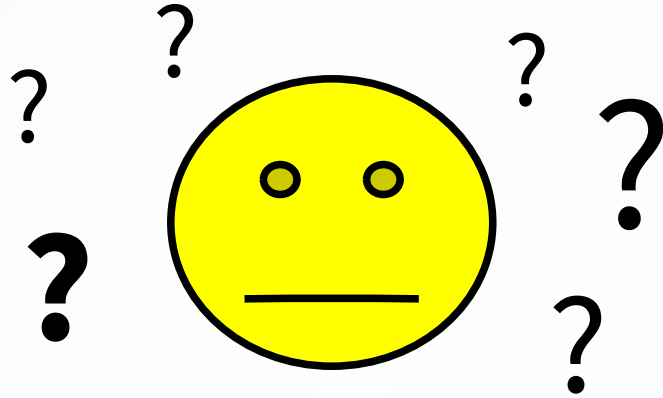


- Nie każdy ma dostęp do konsoli.
- Nie każdy umie obsługiwać CLI
- Jest już drush make



- Szybka aktualizacja
- Łatwa synchronizacja DEV i PROD
- Obsługa komponentów Symfony

- Nie każdy ma dostęp do konsoli.
- Nie każdy umie obsługiwać CLI
- Jest już drush make



“Composer and Drupal are still strange bedfellows”

– Jeff Geerling

Composer i Drupal - oficjalna dystrybucja

`drupal/drupal`

composer update

```
graph TD; A[composer update] --> B[Nie uaktualnia Drupal Core]; A --> C[Wymusza bałagan w VCS]; A --> D[Nie aktualizuje scaffold files]; A --> E[Umieszcza biblioteki PHP w webroot]; A --> F[Ma słabo udokumentowane działanie];
```

Nie uaktualnia Drupal Core

Wymusza bałagan w VCS


Nie aktualizuje scaffold files

Umieszcza biblioteki PHP w webroot

Ma słabo udokumentowane działanie

Plik composer.json (8.4.x)

```
{
  "require": {
    ...,
    "wikimedia/composer-merge-plugin": "~1.4"
  },
  "replace": {
    "drupal/core": "~8.4"
  },
  "extra": {
    "merge-plugin": {
      "include": ["core/composer.json"],
    }
  }
}
```



Poprawiony plik composer.json (8.4.x)

```
{
    "require": {
        ...,
        "drupal/core": "~8.4"
    }
}
```

- > `composer require drupal/google_analytics`
- > `composer require drupal/metatag`
- > `composer require drupal/colorbox`
- > `composer require symfony/var-dumper`
- > `composer require drupal/admin_toolbar`
- > ...
- > `composer update (DEV) -> VCS PUSH -> composer install (PROD)`

Composer i Drupal - nieoficjalny szablon

drupal-composer/drupal-project

composer update

Uaktualnia Drupal Core

Zachowuje maksymalny łań w VCS

Aktualizuje scaffold files

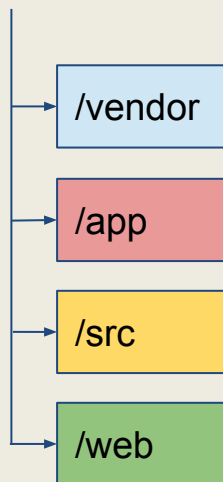
Stara się umieszczać kod PHP poza webroot

```
composer create-project drupal-composer/drupal-project:8.x-dev katalog  
--stability dev --no-interaction
```

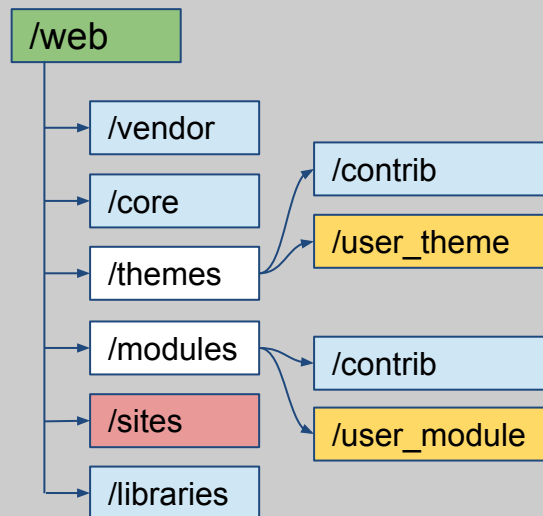

Z jakimi problemami musi zmierzyć się
Composer przy pracy z Drupal 8?

Problem: struktura katalogów

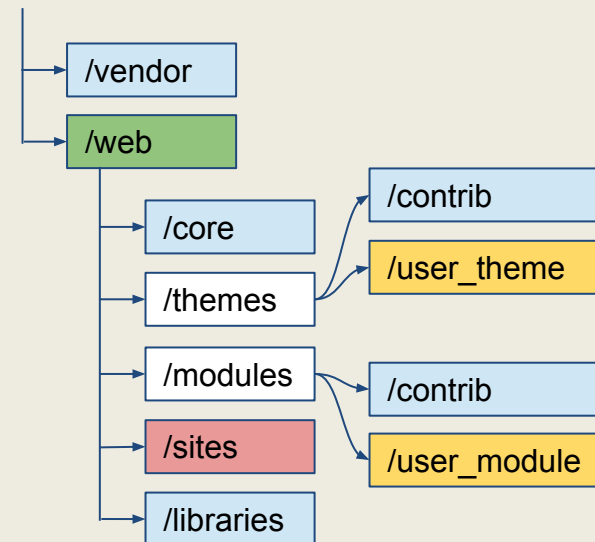
Symfony 2/3



drupal/drupal



drupal-composer/drupal-project



Pliki w webroot
(index/css/js/img)

Kod użytkownika
(np. moduły, skórki)

Biblioteki
(np. moduły contrib)

Inne pliki
(konfiguracja, log, upload)

Problem: struktura katalogów

```
{
  "require": {
    "composer/installers": "^1.0.24",
    ...
  },
  "extra": {
    "installer-paths": {
      "core": ["type:drupal-core"],
      "modules/contrib/{$name}": ["type:drupal-module"],
      "profiles/contrib/{$name}": ["type:drupal-profile"],
      "themes/contrib/{$name}": ["type:drupal-theme"],
      "drush/contrib/{$name}": ["type:drupal-drush"],
      "modules/custom/{$name}": ["type:drupal-custom-module"],
      "themes/custom/{$name}": ["type:drupal-custom-theme"]
    }
  }
}
```

Problem: “breaking API changes”

Jak uniknąć aktualizacji do wersji wstecznie niekompatybilnej?

W Composerze można określić wersję pakietu:

```
"drupal/core": "8.4"
```

```
"drupal/core": ">=8.0 <8.4"
```

```
"drupal/core": "8.*"
```

```
"drupal/core": "~8.0.5"
```

>=8.0.5 <8.1

```
"drupal/core": "^8.0.5"
```

>=8.0.5 <9.0

```
"drupal/core": "^8.0.5-dev"
```

Wymuś wersję DEV

```
"drupal/core": "^8.0.5@dev"
```

Zezwalaj na wersję DEV (min. stability)

Uwaga!

- 8.x-3.0 → **3.0.0**
- 8.x-3.0-alpha26 → **3.0.0-alpha26**
- 8.x-1.x → **1.x-dev**

Problem: update pakiet po pakiecie

> `composer outdated`

```
alchemy/zippy                0.4.3          0.4.8
dflydev/dot-access-configuration v1.0.1        v1.0.2
doctrine/annotations        v1.2.7        v1.4.0
drupal/devel                 1.x-dev 3f24745 1.x-dev 4819ea5
egulias/email-validator     1.2.14        2.1.2
jcalderonzumba/gastonjs    v1.1.0        v1.2.0
phpunit/php-code-coverage   2.2.4         5.2.1
phpunit/phpunit              4.8.35       6.1.4
phpunit/phpunit-mock-objects 2.3.8         4.0.1
```

> `composer update alchemy/zippy --with-dependencies`

> `composer update drupal/devel --with-dependencies`

> `composer update --dry-run`

Problem: pakiety JS/CSS/frontend

- Część pakietów dostępna dla Composera (Bootstrap, jQuery...).
- Część pakietów Bower i NPM w <https://asset-packagist.org/>
- Możliwość użycia pakietów **Bower** przez plugin.
- Możliwość zaimportowania każdego repozytorium GIT/SVN...

ALE

Kosztom rezygnacji z automatycznego wykrywania najnowszych wersji. Trzeba ją określić ręcznie.



Problem: JS/CSS/image assets

- Drupal ma skomplikowaną strukturę folderów.
- Drupal nie zawiera czegoś w rodzaju pakietu Assetic Bundle.
- Composer też nie oferuje rozwiązania Out-Of-The-Box.

Problem: autoload zewnętrznych bibliotek PHP

- Obsługiwane standardy: **PSR-0**, **PSR-4**, classmap, file autoload.
- Autoloader Composer'a jest używany domyślnie przez Drupal'a.

```
{
    "require": {
        "foo/bar": "dev-master",
    },
    "autoload": {
        "classmap": ["foo/bar/classes/"]
    }
}
```


Problem: forkowanie i patchowanie

- Plugin [cweagans/composer-patches](#).
- Bardzo prosta konfiguracja:

```
"extra": {  
  "patches": {  
    "drupal/drupal": {  
      "Example local patch": "patch/to/example.patch"  
    }  
  }  
}
```

Problem: automatyzacja i skrypty

- Można “przyjąć” je w wielu punktach.
- Tworzenie nowych skryptów nie jest skomplikowane.
- Skrypty są dostępne w formie pluginów:
 - ◆ Do generowania scaffold files (robots.txt, index.php...).
 - ◆ Do sprawdzania wersji Composer'a.
 - ◆ Do zarządzania plikami .htaccess w Drupalu.
 - ◆ Do tworzenia plików konfiguracyjnych PHP i YAML w nowych projektach

Problem: Drupal 7?

- Dobra wiadomość: **da się** 
- Wymaga paru sztuczek, wszystkie one zebrane są w szablonie: **drupal-composer/drupal-project**
- Dawniej poprzez **composer_manager**
- Obecnie poprzez oficjalne repozytorium.
- Autoload za pomocą modułu **composer_autoload**

Podsumowanie zalet

Porządek w VCS dzięki .lock

Szybka aktualizacja

Wersjonowanie
semantyczne

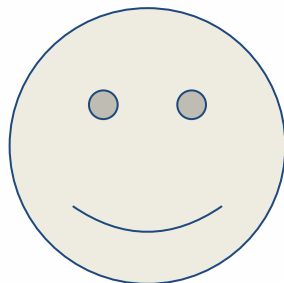
Skrypty instalacyjne

Pojedynczy plik PHAR

Ściąganie zależności

Obsługa np. PEAR

Autoload PSR-0/4



Łatwe patchowanie

Podsumowanie wad

Co z assets?

Problematyczny deploy?

Bezpieczeństwo?

Nie stworzony do frontendu

Merge conflict w .lock

Brak komentarzy w .json

Pamięciożerność

Tylko CLI

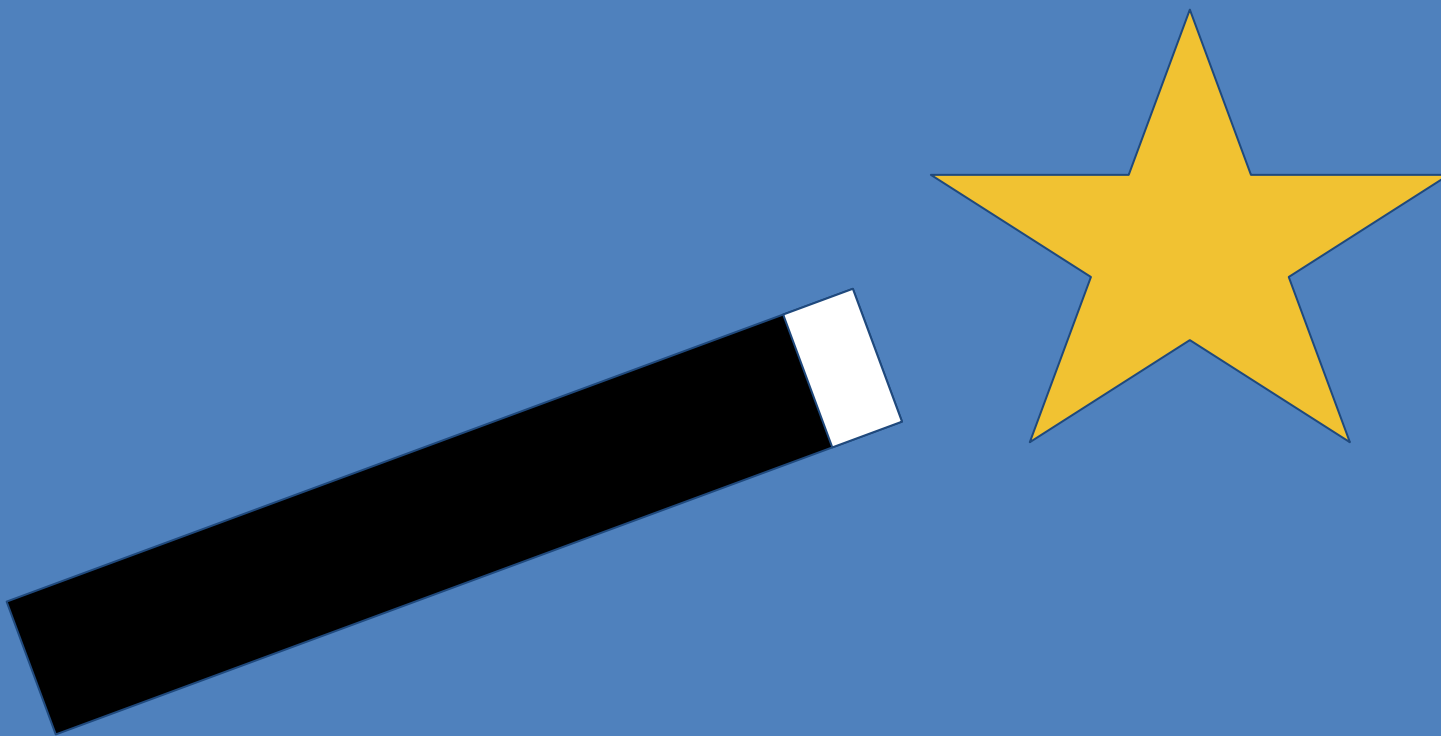
Słaba współpraca
z Acquia Dev Desktop



Kiepska obsługa pakietów
bez composer.json



Pytania?





Pytania?