Magento 2 and Composer

Peter Jaap Blaakmeer

CTO elgentos

@PeterJaap

Also; co-organizer MUG050, volunteer Meet Magento NL, beer home-brewing & board games (so I like IPA’s and API’s).

elgentos
ecommerce solutions

elgentos
ecommerce solutions

What is composer?

Dependency management in PHP

Not a package manager; composer by default installs modules on a per-project basis, not globally.

Why would you use Composer?

Time save
Code reuse
Code sharing
Easy upgrades
Same code usage
Easy removal
Forces you to write clean code; no hacking

elgentos ecommerce solutions

Install composer

```
brew update && brew install homebrew/php/composer
```

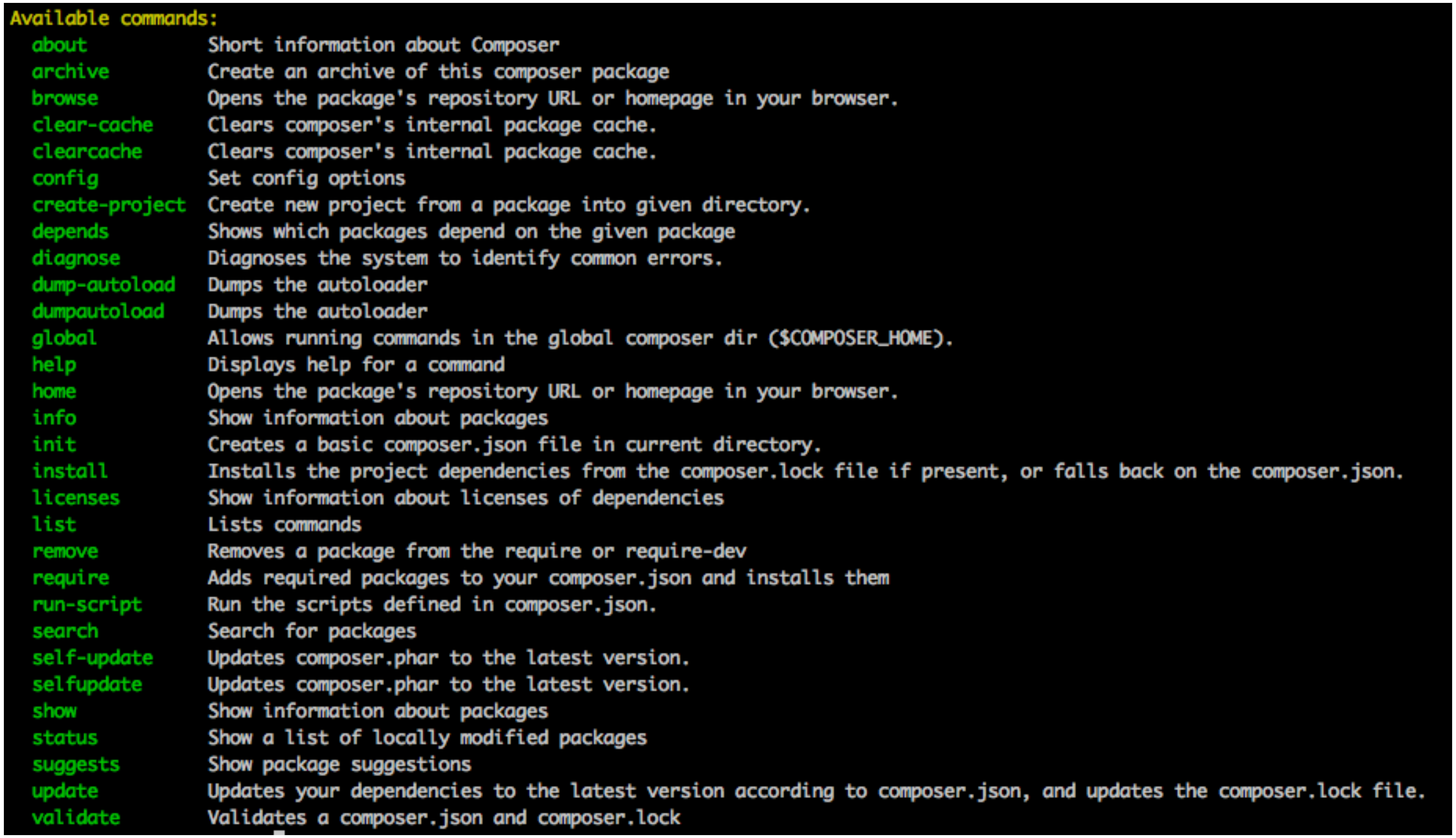
Composer **compo**nents

(see what I did there?)

composer.phar
composer.json
composer.lock

composer.phar

Binary used to work with composer



```
Available commands:
  about            Short information about Composer
  archive          Create an archive of this composer package
  browse           Opens the package's repository URL or homepage in your browser.
  clear-cache      Clears composer's internal package cache.
  clearcache       Clears composer's internal package cache.
  config           Set config options
  create-project   Create new project from a package into given directory.
  depends          Shows which packages depend on the given package
  diagnose         Diagnoses the system to identify common errors.
  dump-autoload    Dumps the autoloader
  dumpautoload     Dumps the autoloader
  global           Allows running commands in the global composer dir ($COMPOSER_HOME).
  help             Displays help for a command
  home             Opens the package's repository URL or homepage in your browser.
  info             Show information about packages
  init             Creates a basic composer.json file in current directory.
  install          Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  licenses         Show information about licenses of dependencies
  list             Lists commands
  remove           Removes a package from the require or require-dev
  require          Adds required packages to your composer.json and installs them
  run-script       Run the scripts defined in composer.json.
  search           Search for packages
  self-update      Updates composer.phar to the latest version.
  selfupdate       Updates composer.phar to the latest version.
  show             Show information about packages
  status           Show a list of locally modified packages
  suggests         Show package suggestions
  update           Updates your dependencies to the latest version according to composer.json, and updates the composer.lock file.
  validate         Validates a composer.json and composer.lock
```

composer.phar

Most used commands

```
$ composer update
$ composer install
$ composer require
$ composer create-project
```

Projects' composer.json

```
{
  "name": "magento/magento2ce",
  "description": "Magento 2 (Community Edition)",
  "type": "project",
  "version": "2.0.0",
  "license": [
    "OSL-3.0",
    "AFL-3.0"
  ],
  "require": {
    "php": "~5.5.0|~5.6.0|~7.0.0",
    "zendframework/zend-stdlib": "~2.4.6",
    "zendframework/zend-code": "~2.4.6",
    "zendframework/zend-server": "~2.4.6",
    "zendframework/zend-soap": "~2.4.6",
    "zendframework/zend-uri": "~2.4.6",
    [...]
  },
  "require-dev": {
    "phpunit/phpunit": "4.1.0",
    "squizlabs/php_codesniffer": "1.5.3",
    [...]
  },
  "replace": {
    "magento/module-marketplace": "100.0.2",
    "magento/module-admin-notification": "100.0.2",
    "magento/module-advanced-pricing-import-export": "100.0.2",
    "magento/module-authorization": "100.0.2",
    "magento/module-authorizenet": "100.0.2",
    "magento/module-backend": "100.0.2",
    "magento/module-backup": "100.0.2",
    [...]
  },
  "extra": {
```

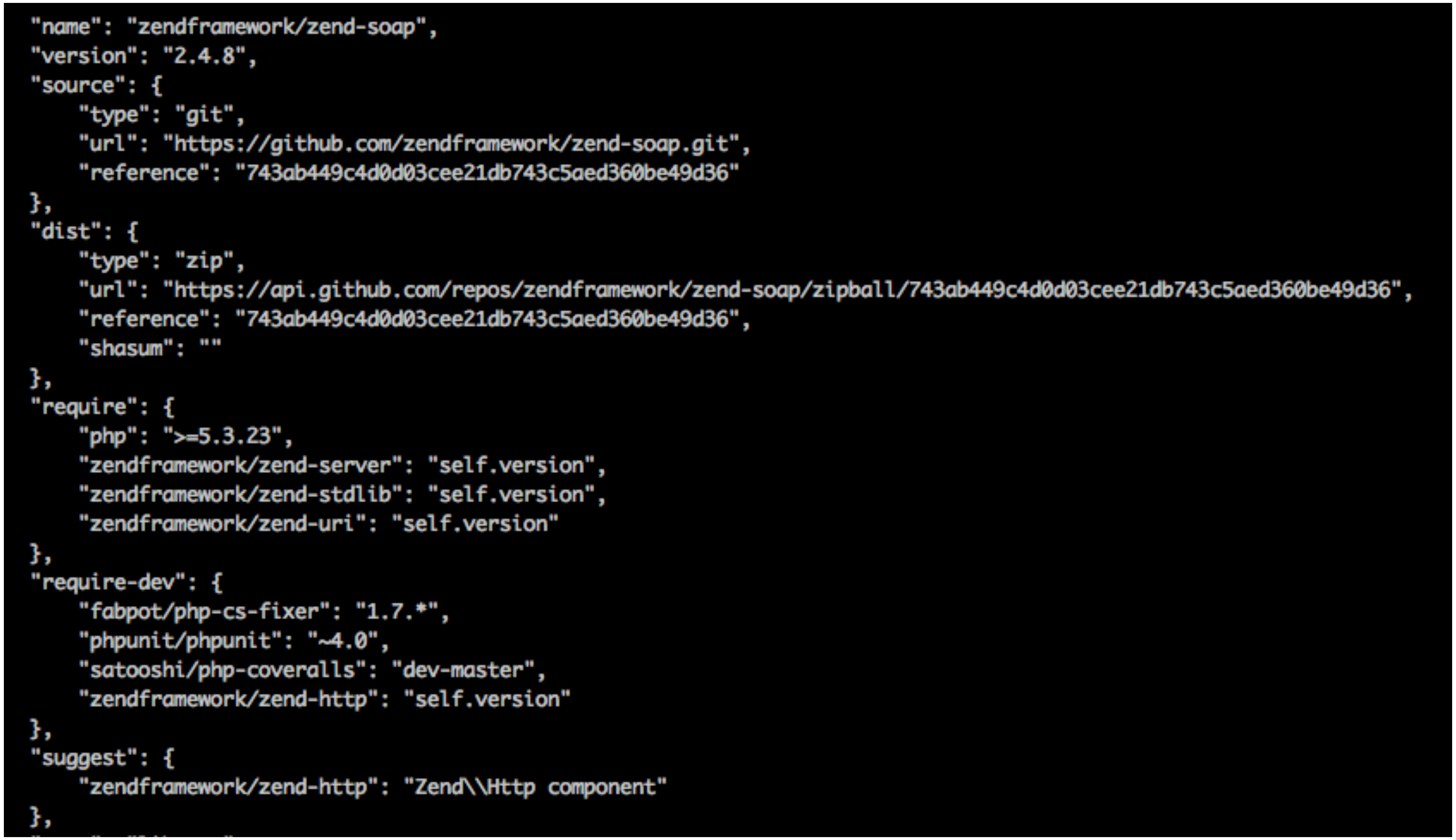
elgentos
ecommerce solutions

Extensions' composer.json

```
{
  "name": “elgentos/mage2importer",
  "description": “Fast refactored Magento 2 product importer",
  "type": “magento2-module", // or magento2-theme / magento2-language / metapackage
  "version": "1.3.37",
  "license": [
    "OSL-3.0",
    "AFL-3.0"
  ],
  "require": {
    "php": "~5.5.0|~5.6.0|~7.0.0",
    "magento/framework": "~100.0"
  },
  "extra": {
        "map": [
            [
                "*",
                "Elgentos/Mage2Importer"
            ]
        ]
    }
}
```

composer.lock

Lockfile created when running composer update



```
"name": "zendframework/zend-soap",
"version": "2.4.8",
"source": {
    "type": "git",
    "url": "https://github.com/zendframework/zend-soap.git",
    "reference": "743ab449c4d0d03cee21db743c5aed360be49d36"
},
"dist": {
    "type": "zip",
    "url": "https://api.github.com/repos/zendframework/zend-soap/zipball/743ab449c4d0d03cee21db743c5aed360be49d36",
    "reference": "743ab449c4d0d03cee21db743c5aed360be49d36",
    "shasum": ""
},
"require": {
    "php": ">=5.3.23",
    "zendframework/zend-server": "self.version",
    "zendframework/zend-stdlib": "self.version",
    "zendframework/zend-uri": "self.version"
},
"require-dev": {
    "fabpot/php-cs-fixer": "1.7.*",
    "phpunit/phpunit": "~4.0",
    "satooshi/php-coveralls": "dev-master",
    "zendframework/zend-http": "self.version"
},
"suggest": {
    "zendframework/zend-http": "Zend\\Http component"
},
```

composer.lock

What is the lock file for?

It ensures every developer uses the same version of the packages.

`composer update` - installs the latest versions referenced in composer.json & save commit hash in lock file.

`composer install` - installs a specific version identified by a commit hash in the lock file.

How to handle composer files in Git?

You should commit composer.json to keep track of which extensions are installed.

You can commit composer.lock but it is not necessary, depends on your deployment structure, but you'll probably get a lot of merge conflicts.

require vs require-dev

‘require’ is for production modules

‘require-dev’ is for dev modules

Run ‘composer install —no-dev’ on your production server to skip installing development modules

```
"require-dev": {
  "aoepeople/aoe_templatehints": "^0.4",
  "aoepeople/aoe_profiler": “^0.3",
  "pulsestorm/magento-better404": "^1.0"
},
```

Composer repositories

Packagist (default)
http://packages.magento.com/
http://packages.firegento.com/
Set up your own with Satis / Toran Proxy

Here’s a Talesh for good measure

Magento 1 + composer

Not supported by the core, but, of course, there's a module for that;

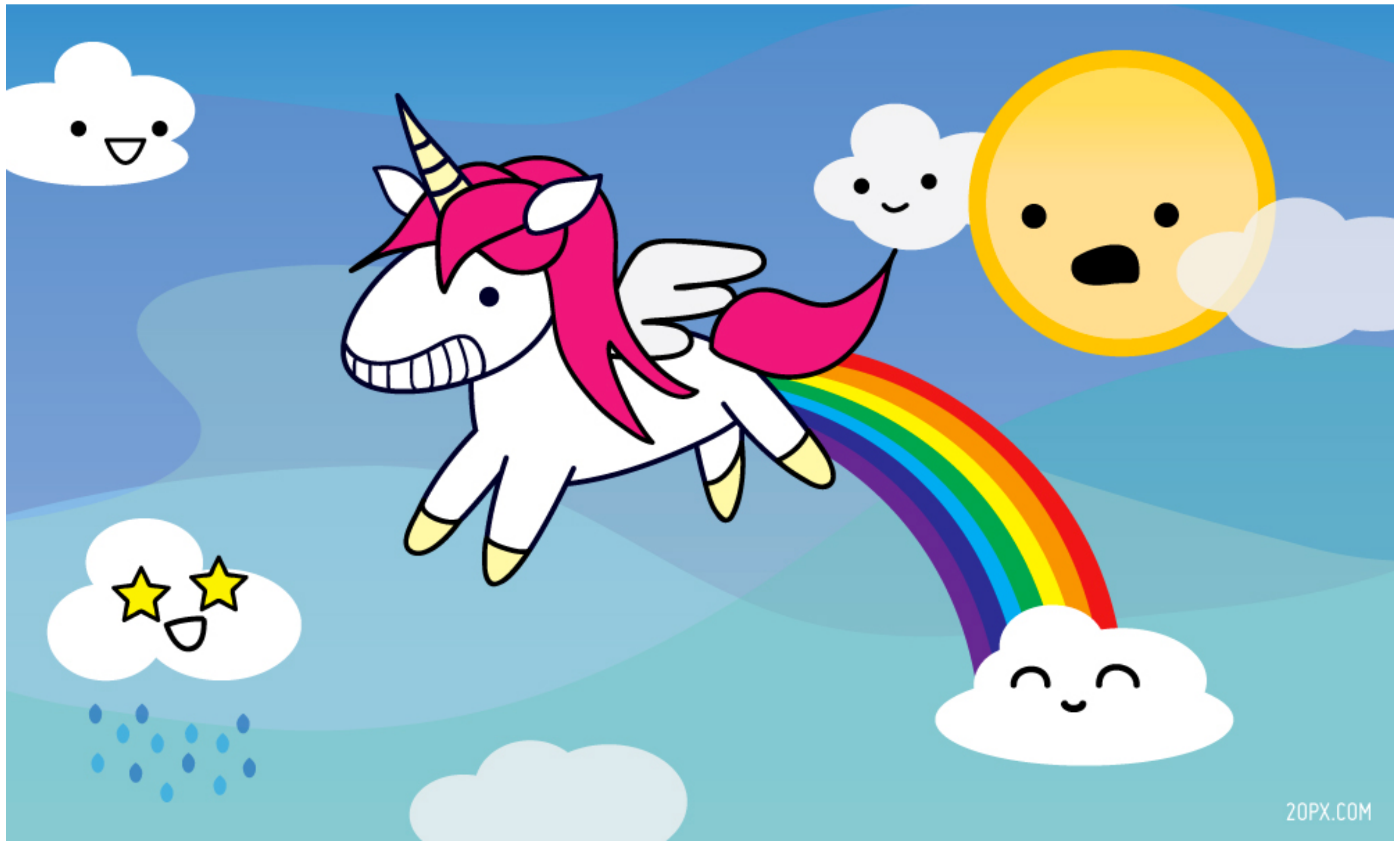
https://github.com/Cotya/magento-composer-installer

- This places the files in your composer module in the correct paths by using symlinks
- Tell the installer where to place your files through a modman file, package.xml or in the composer.json file itself through `extra > map`.
- You need to add the Magento root dir to your projects' composer.json;

```
{
  ...
  "extra":{
    "magento-root-dir":"./htdocs"
  }
}
```

Magento 2 + composer

Built-in support!



Magento itself consists of a large number of composer packages, both from 3rd party and internal components.

Magento 2 composer packages

Extensions
Libraries
Language packs
Themes
etc

Are installed in `vendor`

Repository authentication

Why?

Keeping track of installed extensions
Keeping track of extensions purchased through Marketplace
Notifications of new versions!
Installing of patches!

How?

Through your magento.com account - log in with your normal account and create a username/password combo for composer authentication.

These credentials are saved in var/composer_home/auth.json so you can Git it.

Custom extensions

Why?

Easy installable, updatable and reusable code.

How?

Place each extension in a separate Git repository
Add a composer.json that sets the name and dependencies
Add the Git repo link to the projects' composer.json file
Run composer update
Commit composer.json to Git

Metapackages

A metapackage is a package that consists of multiple other packages.

You can use this to bundle extensions that are often used together.

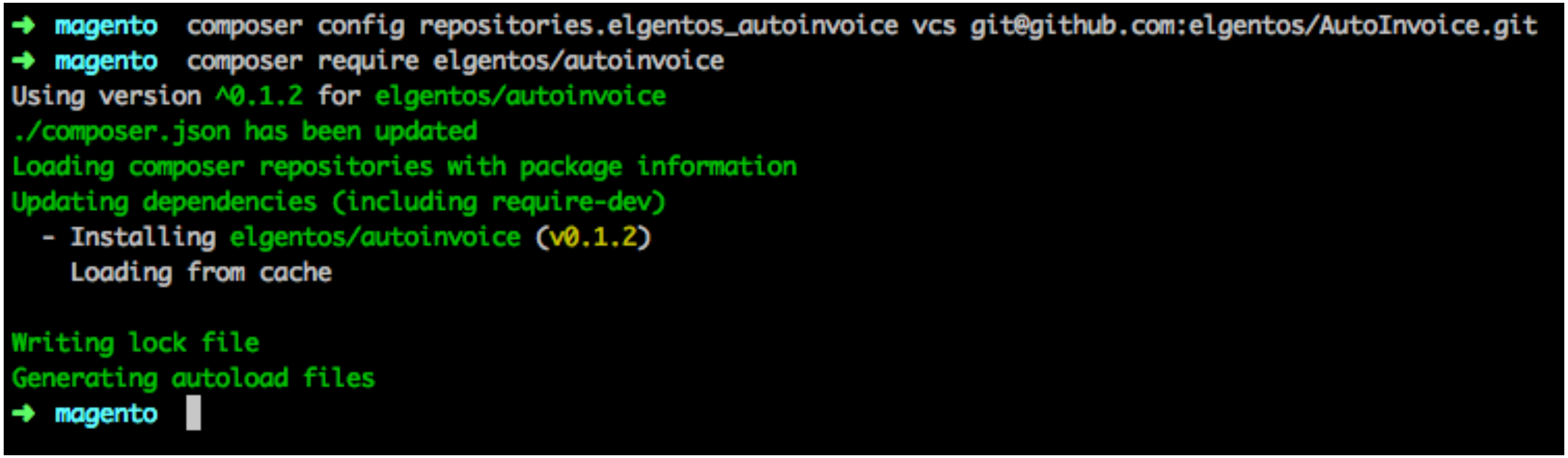
Example - editing composer.json

```
{
  "require": {
    […]
      "elgentos/autoinvoice": "^0.1.2",
    },
    "repositories": {
      "elgentos/autoinvoice": {
      "type": "vcs",
      "url": "git@github.com:elgentos/AutoInvoice.git"
    },
    […]
  }
}
```

‘repositories’ part is not necessary when package exists in Packagist/other added repos

And run composer update.

Example - using composer.phar



```
➜ magento  composer config repositories.elgentos_autoinvoice vcs git@github.com:elgentos/AutoInvoice.git
➜ magento  composer require elgentos/autoinvoice
Using version ^0.1.2 for elgentos/autoinvoice
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing elgentos/autoinvoice (v0.1.2)
    Loading from cache

Writing lock file
Generating autoload files
➜ magento  
```

Semantic versioning

- MAJOR.MINOR.PATCH
- works through Git tag
- tag every versioned release
- never use ‘dev-master’ or similar
- semver.org

Range modifiers

Tilde; ~1.2.3 will match all 1.2.x versions but will miss 1.3.0
Caret; ^1.2.3 will match any 1.x.x release including 1.3.0, but not 2.0.0

Examples

Specific version: 1.0.2
Range: >=1.0 <2.0
Range shortcut: ^1.2 (equal to >= 1.2.0 < 2.0.0, recommended)

Bottom line

- Using composer makes your code and your workflow more robust
- If you don't use it yet, start using it NOW!
- Get hands-on experience by starting to use it with Magento 1

elgentos ecommerce solutions