



# Cornell DrupalCamp 2017

October 19-20, 2017

Cornell University Ithaca, NY

[camp.drupal.cornell.edu](http://camp.drupal.cornell.edu)



Drupal &

# Composer for Absolute Beginners

Alison Jo McCauley

Drupal Developer, Cornell University

# What is **Composer**?

**Composer** is a (command-line) tool for dependency management in PHP.

With **composer**, you declare the libraries / packages / tools your project depends on -- and **composer** will manage (install/update) them for you.

- Dependency management??

Basically... package management, but per-project, not global.

 [getcomposer.org](https://getcomposer.org)

← **pro tip!** *This is the **composer** website.*





# composer! + Drupal!

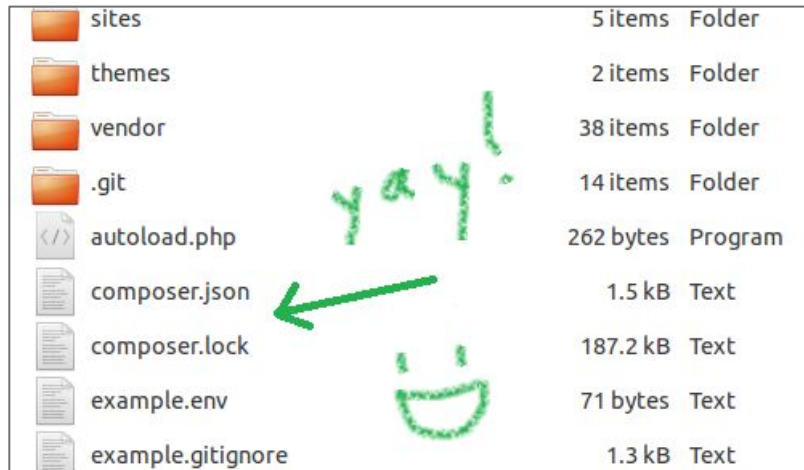
So, using **composer** with Drupal... you declare the libraries / packages / tools your Drupal project depends on -- and **composer** will manage (install/update) them for you.



# composer! + Drupal!

So, using **composer** with Drupal... you declare the libraries / packages / tools your Drupal project depends on -- and **composer** will manage (install/update) them for you.

- Ah! There are **composer.json** and **composer.lock** files in Drupal 8!



sites	5 items	Folder
themes	2 items	Folder
vendor	38 items	Folder
.git	14 items	Folder
autoload.php	262 bytes	Program
composer.json	1.5 kB	Text
composer.lock	187.2 kB	Text
example.env	71 bytes	Text
example.gitignore	1.3 kB	Text



# composer! + Drupal!

So, using **composer** with Drupal... you declare the libraries / packages / tools your Drupal project depends on -- and **composer** will manage (install/update) them for you.

- ~~Ah! There are composer.json and composer.lock files in Drupal 8!~~
- Nope, those aren't for us.



sites	5 items	Folder
themes	2 items	Folder
vendor	38 items	Folder
.git	14 items	Folder
autoload.php	262 bytes	Program
composer.json	1.5 kB	Text
composer.lock	187.2 kB	Text
example.env	71 bytes	Text
example.gitignore	1.3 kB	Text







# Composer and YOUR Drupal site

You'd use **composer** to...

- Get the (non-core) modules and themes on which your site depends.
- Get the libraries and any other packages on which your site depends.
- Update + remove packages.
- Track / maintain precise details about the Stuffs your project is made (*composed!*) of -- **composer.json does what drush.make does**, but more fancy-pants.



# Composer and YOUR Drupal site

You'd use **composer** to...

- Get the (non-core) modules and themes on which your site depends.
- Get the libraries and any other packages on which your site depends.
- Update + remove packages.

...including whatever packages ^^**those** packages need!

...using the right package versions for your site needs!

...keeping track of exactly which version of what is installed!



# Composer and YOUR Drupal site

You won't use **composer** to...

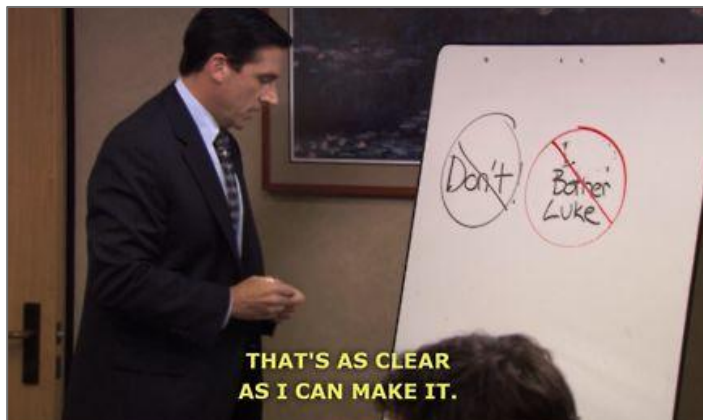
- Do any database things!
  - Enable / disable modules (etc.) on your Drupal site.  
(although the terminology within Drupal is still “install” / “uninstall” !)
  - Manage site configuration.
  - Content anythings.



# Composer and YOUR Drupal site

You won't use **composer** to...

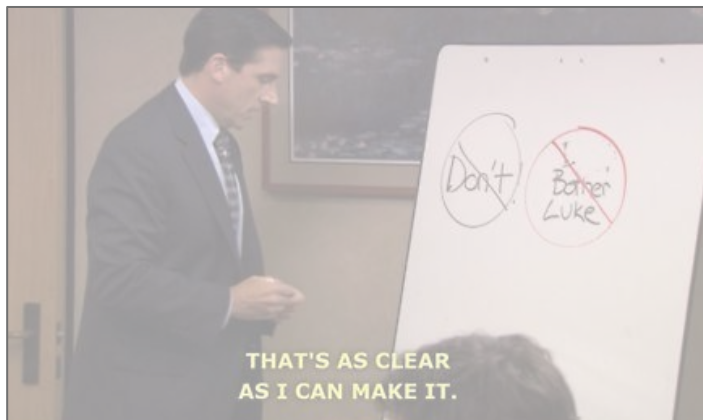
- Do any database things!
  - Enable / disable modules (etc.) on your Drupal site.  
(although the terminology within Drupal is still “install” / “uninstall” !)
  - Manage site configuration.
  - Content anythings.
- Do anything NOT from the command-line (CLI).



# Composer and YOUR Drupal site

You won't use **composer** to...

- Do any database things!
  - Enable / disable modules (etc.) on your Drupal site.  
(although the terminology within Drupal is still “install” / “uninstall” !)
  - Manage site configuration.
  - Content anythings.
- Do anything NOT from the command-line (CLI).



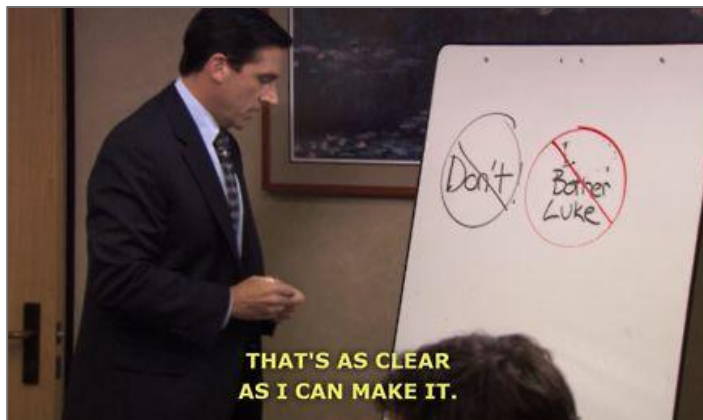
- Enable / disable modules (etc.) on your Drupal site.  
(although the terminology within Drupal is still “install” / “uninstall” !)

*^^ **pro tip!** That’s right -- when you “install” a contrib module via **composer**, the module files are added to your project (and **composer** makes your project’s dependency details reflect the added module), but the module is NOT “installed” on your Drupal site.*

# Composer and YOUR Drupal site

You won't use **composer** to...

- Do any database things!
  - Enable / disable modules (etc.) on your Drupal site.  
(although the terminology within Drupal is still “install” / “uninstall” !)
  - Manage site configuration.
  - Content anythings.
- Do anything NOT from the command-line (CLI).



# Composer and YOUR Drupal site

*Prerequisites: Install composer! (aforementioned fancy website link ^^ )*

- Create a *new* Drupal 8 project -- a new **composer** project!
  - (Converting an existing Drupal site ~~is not impossible, but it's not for Absolute Beginners.~~) ← ***for all intents and purposes... it's impossible***
- Add modules and whatever other packages as dependencies.
- Use **composer** to install (download) the packages to your project.





# composer

- composer.json -- lives in project root; its presence means “This is a **composer**-managed project.”
  - Contains list of project dependencies, including version constraints -- what versions or version ranges are needed
  - Also contains some structural specifications, like installation paths, extra composer scripts, and misc project config.

# composer! + Drupal!

Your composer-managed Drupal project will have a `composer.json` file in its root directory.

**^^ pro tip!** *You'll run all `composer` commands from the directory that contains `composer.json`!*

- This file is where composer keeps track of all the packages (modules, libraries, etc.) your project needs -- i.e. the ✨dependencies✨



*Let's go!*



show composer.json >>

# composer! + Drupal!

Your composer-managed Drupal project will also have a `composer.lock` file in its root directory.

- This file is a record of exactly what packages and package versions are currently installed on your project.

^^ **pro tip!** *Installed == downloaded!*



# composer! + Drupal!

Your composer-managed Drupal project will have a `composer.lock` file in its root directory.

- This file is a record of exactly what packages and package versions are currently installed on your project.

^^ **pro tip!** *Installed == downloaded!*

```
$ composer install
```



# composer

```
$ composer install
```

`composer install` grabs all your project's package artifacts based on the specs in `composer.lock`.



# composer

- If you run `composer install`, it will use what is in the `composer.lock` file -- even if there are details in `composer.json` that are not reflected in `composer.lock`
- For example, even if you have not changed dependencies, there could be newer versions of packages that fit version constraints in `composer.json`, but are not yet reflected in `composer.lock`.

***Pro tipssss!*** To update `composer.lock` to reflect latest everything, run `composer update`. ***AND***, if you're having any issues with `composer.lock`, it's always ok to just delete `composer.lock` and run `composer install`.



# composer! + Drupal!

Package files come from Packagist.org by default, but you can add repositories (sources) to your composer.json, to allow packages from other sources.

- The official Drupal composer package service is now:

<https://packages.drupal.org/8>

# composer! + Drupal!

Package files come from Packagist.org by default, but you can add repositories (sources) to your composer.json, to allow packages from other sources.

- The official Drupal composer package service is now:

<https://packages.drupal.org/8>

<https://www.drupal.org/node/2718229#drupal-packagist>

Using Composer to manage contributed dependencies for a Drupal site is unique. It requires some Drupal-specific configuration in your composer.json file, namely:

1. Define Drupal.org as the source of drupal packages
2. Define the directories to which Drupal projects should be downloaded

If you are using `drupal-composer/drupal-project` as a template for your Drupal site, this configuration is already handled for you. Otherwise, **you will need to**

# composer! + Drupal!

Package files come from Packagist.org by default, but you can add repositories (sources) to your composer.json, to allow packages from other sources.

- The official Drupal composer package service is now:

<https://packages.drupal.org/8>

<https://www.drupal.org/node/2718229#drupal-packagist>

Using Composer to manage contributed dependencies for a Drupal site is unique. It requires some Drupal-specific configuration in your composer.json file, namely:

1. Define Drupal.org as the source of drupal packages
2. Define the directories to which Drupal projects should be downloaded

If you are using `drupal-composer/drupal-project` as a template for your Drupal site, this configuration is already handled for you. Otherwise, **you will need to**

yayyy!!



# composer

Package files get downloaded to your project's vendor directory by default, and/or wherever you specified if you have "install-path" in composer.json.

# composer! + Drupal!

Package files get downloaded to your project's vendor directory by default, and/or wherever you specified if you have "install-path" in composer.json.

- The Super Special **drupal-composer/drupal-project** has configurations that send module files into the web/modules directory, theme files into the web/themes directory, etc.

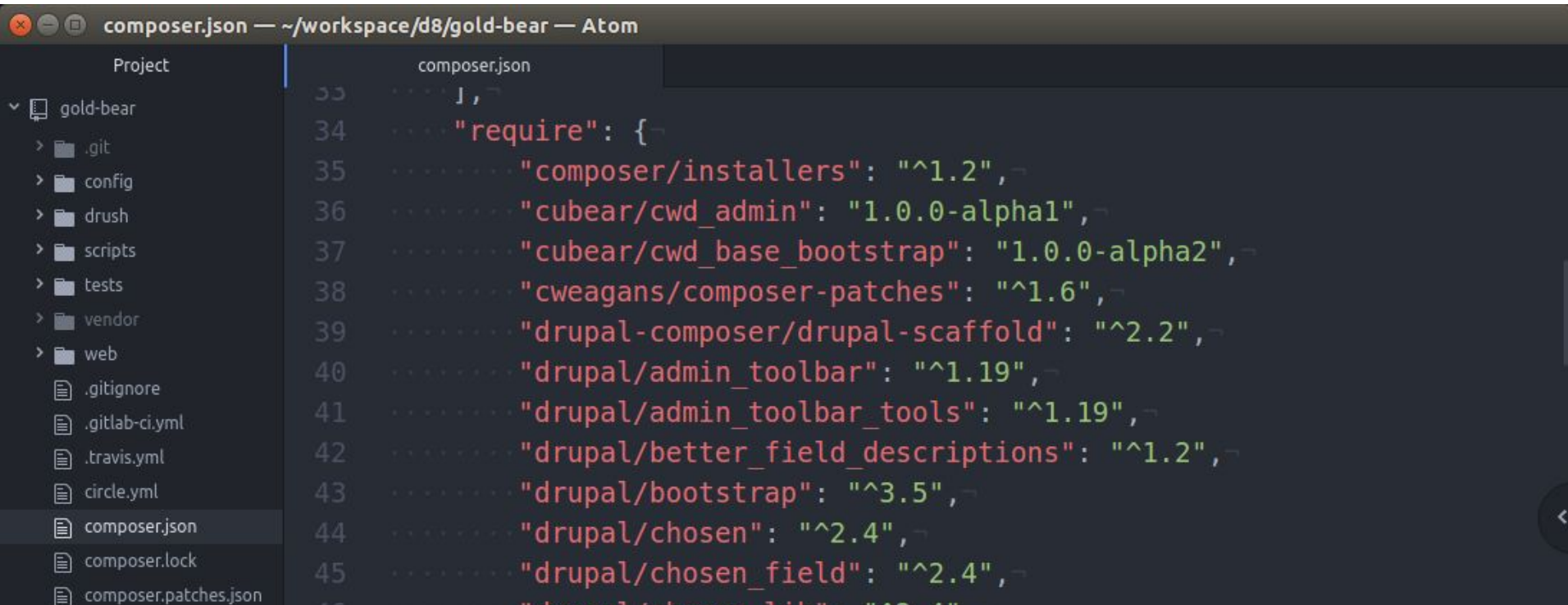
```
$ composer require the_vendor_name/the_package_name:^X.x
```

- Adds a line to your composer.json and downloads the package artifacts.
- Downloads the package (version X.x) to your project -- to vendor/ (based on the\_vendor\_name) or based on “installer-paths” if specified.

↑ **specificity** == ↑ **safety!**

- Updates composer.lock with all kinds of details about what it just downloaded -- including the exact version.

```
$ composer require drupal/better_field_descriptions:^1.2
```



The screenshot shows the Atom editor interface. The title bar indicates the file is `composer.json` located in `~/workspace/d8/gold-bear`. The left sidebar shows a project tree for `gold-bear` with folders like `.git`, `config`, `drush`, `scripts`, `tests`, `vendor`, and `web`, along with files like `.gitignore`, `.gitlab-ci.yml`, `.travis.yml`, `circle.yml`, `composer.json`, `composer.lock`, and `composer.patches.json`. The main editor area displays the contents of `composer.json`, showing a `require` section with the following dependencies:

```
34     "require": {  
35         "composer/installers": "^1.2",  
36         "cubear/cwd_admin": "1.0.0-alpha1",  
37         "cubear/cwd_base_bootstrap": "1.0.0-alpha2",  
38         "cweagans/composer-patches": "^1.6",  
39         "drupal-composer/drupal-scaffold": "^2.2",  
40         "drupal/admin_toolbar": "^1.19",  
41         "drupal/admin_toolbar_tools": "^1.19",  
42         "drupal/better_field_descriptions": "^1.2",  
43         "drupal/bootstrap": "^3.5",  
44         "drupal/chosen": "^2.4",  
45         "drupal/chosen_field": "^2.4",  
46         "drupal/ctools": "1.0.0-alpha1"
```



- gold-bear
  - .git
  - config
  - drush
  - scripts
  - tests
  - vendor
    - alchemy
    - asm89
    - behat
    - bin
    - caxy
    - composer
    - container-interop
    - cweagans
    - dflydev
    - dnoegel
    - doctrine
    - drupal
    - drupal-composer
    - easyrdf
    - egulias
    - ezyang
    - fabpot
    - gabordemoij
    - guzzlehttp

```
33     ],
34     "require": {
35         "composer/installe...
36         "cubear/cwd_adm...
37         "cubear/cwd_bas...
38         "cweagans/compo...
39         "drupal-compose...
40         "drupal/admin_t...
41         "drupal/admin_t...
42         "drupal/better_...
43         "drupal/bootstr...
44         "drupal/chosen'...
45         "drupal/chosen_...
46         "drupal/chosen_...
47         "drupal/coffee'...
48         "drupal/config_...
49         "drupal/config_...
50         "drupal/config_...
51         "drupal/config_...
52         "drupal/console...
53         "drupal/core": ...
54         "drupal/ctools'...
55         "drupal/develop'...
```

# composer: Version constraints

- The two version constraints you'll see most often are the ^ and ~ .
  - ~ ----- ~8.3 won't go to 8.4.0
  - ^ ----- ^8.3 will go to 8.4.x, 8.5.x, etc. to 8.9xxx

Both operators tell Composer 'use at least this version or higher', but ~ says stay within the same minor release (8.3.x), while the ^ says stay within the same major release (all releases up to, but not including, 9.0.0).



# composer: Version constraints

ANYWAY...

- Whatever is put after the ^ version constraint will use the newest available release **in that major version**.
  - It will not go above the specified MAJOR version.
- You can also specify the exact version number you would like to use, and it will not change until you manually required a different version.
- Drupal projects, ^ or exact numbers are most common; ~ less so.

↑ **specificity** == ↑ **safety!**



## composer! + Drupal! Base package?

- As you may have noticed, **composer** + drupal projects have a different directory structure than vanilla drupal projects.
- It's widely accepted that if you're going to do a composer-managed Drupal 8 project, you should use the **drupal-composer/drupal-project** package as your base.

*^^ Do you use Pantheon? Good news -- the pantheon D8 composer repo (pantheon-systems/example-drops-8-composer) totally uses **drupal-composer/drupal-project** -- phew!*



## composer! + Drupal! Base package?

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

- When you create a project with this base, the directory structure is all set up, and there are some helpful configs built-in.
- ...Then, add whatever modules / other packages, initialize git in the project root, commit, and deploy to your target environment.



## composer remove

- `composer remove` simply removes the package from your project -- removing it from `composer.lock`, `composer.json`, and the actual package files.



## composer remove! + Drupal!

- `composer remove` simply removes the package from your project -- removing it from `composer.lock`, `composer.json`, and the actual package files.

***Pro tip!*** Make sure you *uninstall modules from Drupal **before** removing them from your project.*



# composer! + Drupal! Patches?

- If you DO need stinking patches...

<https://github.com/cweagans/composer-patches>

*^^ Good news! This package comes with **drupal-composer/drupal-project** -- so just keep this link in mind for when you want to add patches (instructions in the README).*





composer! + Drupal! Pain points?



lolololololololololololol

#laughsob

#tearsofEverything



# composer! + Drupal! Pain points?

- Learning curve! For example, confusing dependency errors, composer.lock conflicts...
- **composer** is RESOURCE HEAVY and takes for.eh.vur....
  - ^^ makes troubleshooting those errors extra super duper fun!
  - Prestissimo plugin for parallel downloading -- but **composer** still takes ages to do the voodoo that it does so well.
  - **Pro tip!** -vvv to feel like something is happening!
- CLI only
- **Existing Drupal 8 sites?** lololololnope *(no but seriously, you can't)*

See also: [Composer and Drupal are still strange bedfellows \(Jeff Geerling\)](#)



# The end!

@alisonjo2786 everywhere; [alison@cornell.edu](mailto:alison@cornell.edu)

- [Drupal + composer docs \(d.o\)](#)
- [Jeff Geerling: Composer and Drupal are still strange bedfellows](#)
- [drupal-composer/drupal-project](#)
- [D.o thread about improving Drupal + composer integration](#)
- And, a few d.o threads related to the future of Drupal + composer + site building, etc (from my soapbox at the end) -- many many more threads are linked to in the “related links” on these two issues ;-)
- [\[META\] Replace update\\_manager with a more powerful solution](#)
- [Use Composer to build sites without forcing users to learn Composer](#)
- [Drush PR re: compatibility with Drupal 8.4+ and related things](#)
  - Also: [Drush + Drupal core compatibility is fragile \(d.o\)](#)

