Composer PHP package manager

START HERE

`composer` **`init`**
Create composer.json

`composer` **`require`** *`<vendor/pkg>`*
Add a dependency

`composer` **`install`**
Install from the lock file

`composer` **`update`**
Upgrade within constraints

`composer` **`remove`** *`<vendor/pkg>`*
Drop a dependency

`composer` **`show`**
List installed packages

`composer` **`outdated`**
See newer versions

`composer` **`dump-autoload`**
Rebuild the autoloader

composer.json your constraints → `update` / `require` → **composer.lock** exact versions → `install` → **vendor/** packages + autoload.php

CONCEPTS

Commit composer.lock for applications: install then gives every machine the same versions.

install reads the lock file; update re-solves composer.json and rewrites the lock.

Libraries usually do not commit composer.lock; only applications do.

PROJECT

`composer` **`init`**
Create composer.json interactively

`composer` **`create-project`** *`<vendor/pkg> <dir>`*
Start a new project from a package

`composer` **`validate`**
Check composer.json is valid

`composer` **`validate --strict`**
Treat warnings as errors

`composer` **`config`** *`<key> <value>`*
Set a config value

`composer` **`config --list`**
Show the effective config

`composer` **`config --global`** *`<key> <value>`*
Set a user-wide config value

PROJECT (cont.)

`composer` **`diagnose`**
Check system and network problems

INSTALL & REMOVE

`composer` **`install`**
Install exactly what the lock says

`composer` **`install --no-dev`**
Skip require-dev packages

`composer` **`install -o`**
Optimize the autoloader

`composer` **`install -a`**
Authoritative classmap (production)
Skips filesystem lookups for missing classes, so classes created later are not found.

`composer` **`install --dry-run`**
Show what would happen

`composer` **`remove`** *`<vendor/pkg>`*
Remove a dependency

`composer` **`clear-cache`**
Empty the download cache

REQUIRE & UPDATE

`composer` **`require`** *`<vendor/pkg>`*
Add a package and install it

REQUIRE & UPDATE (cont.)

`composer` **`require`** *`<vendor/pkg>:<constraint>`*
Add with a version constraint
e.g. composer require "monolog/monolog:^3.0" (quote the constraint).

`composer` **`require --dev`** *`<vendor/pkg>`*
Add a dev-only dependency

`composer` **`update`**
Update everything within constraints

`composer` **`update`** *`<vendor/pkg>`*
Update only one package

`composer` **`update`** *`<vendor/pkg>`* **`-W`**
Also update its dependencies

`composer` **`update --dry-run`**
Preview an update

INSPECT

`composer` **`show`**
List installed packages

`composer` **`show`** *`<vendor/pkg>`*
Details of one package

`composer` **`show -t`**
Dependency tree

`composer` **`outdated`**
List packages with newer versions

`composer` **`outdated -D`**
Only direct dependencies

INSPECT (cont.)

`composer` **`why`** *`<vendor/pkg>`*
Why is this installed?

`composer` **`why-not`** *`<vendor/pkg> <version>`*
Why can't this version be installed?

`composer` **`search`** *`<text>`*
Search Packagist

`composer` **`check-platform-reqs`**
Check PHP and extension needs

AUTOLOAD & SCRIPTS

`composer` **`dump-autoload`**
Regenerate the autoloader

`composer` **`dump-autoload -o`**
Generate an optimized classmap

`composer` **`run-script`** *`<name>`*
Run a script from composer.json

`composer` *`<name>`*
Shortcut for run-script
Works unless the name clashes with a built-in command.

`composer` **`run-script -l`**
List available scripts

`composer` **`run-script`** *`<name>`* **`--`** *`<args>`*
Pass arguments to a script

AUTOLOAD & SCRIPTS (cont.)

`composer` **`exec`** *`<bin>`*
Run a binary from vendor/bin

SECURITY & AUDIT

`composer` **`audit`**
Check for known vulnerabilities

`composer` **`audit --locked`**
Audit the lock file only

GLOBAL & OPTIONS

`composer` **`global require`** *`<vendor/pkg>`*
Install a tool for your user

`composer` **`self-update`**
Update Composer itself

`composer` **`-d`** *`<dir> <command>`*
Run in another directory

`composer` **`-n`** *`<command>`*
Never ask questions (CI)

`composer` **`-v / -vv / -vvv`**
More output

`composer` **`--ignore-platform-reqs install`**
Ignore PHP/extension checks
Only a workaround; the code may still fail at runtime.

GLOBAL & OPTIONS (cont.)

`composer` **`--version`**
Show the Composer version

`composer` **`help`** *`<command>`*
Help for a command

`composer` **`list`**
List all commands

RECIPES

New project with a dev tool

```
composer init
composer require monolog/monolog
composer require --dev phpunit/phpunit
vendor/bin/phpunit
```

Production deploy

```
composer install --no-dev -o --no-interaction
# or: -a for an authoritative classmap
```

Careful dependency upgrade

```
composer outdated -D
composer update <vendor/pkg> --dry-run
composer update <vendor/pkg>
composer audit
```

Find out why a version is blocked

```
composer why <vendor/pkg>
composer why-not <vendor/pkg> <version>
composer update <vendor/pkg> -W
```