**https://getcomposer.org/**

composer

[**https://packagist.org/**](https://packagist.org/)

composer packages main repository

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

install locally

**php composer.phar**

local composer version

**sudo mv composer.phar /usr/bin/composer**

[https://getcomposer.org/doc/00-intro.md#globally](https://getcomposer.org/doc/00-intro.md" \l "globally)

install globally after local installation on ubuntu 22

**composer self-update**

**composer selfupdate**

updates composer to the latest version

**sudo apt install composer**

global installation: + не нужно обновлять отдельно; - более старая версия

**[https://getcomposer.org/doc/03-cli.md#bash-completions](https://getcomposer.org/doc/03-cli.md" \l "bash-completions)**

завершение (дописывание) команд по нажатию на Tab

**composer** command **--help**

command description

**composer init**

Creates a basic composer.json file in current directory.

будет заданно несколько вопросов: имя, тип, описание, подключаемые пакеты …

**!**внимание: для project обязательна версия (например v1.0.0) иначе при скачивании команда

create-project будет выдавать ошибку "Could not find package vendor/name with stability stable".

Версия указывается в git tag (смотри шпаргалку по git) и push отдельно от commit-а.

**composer search** packageName

search package in CLI

**composer require**

**composer r**

composer will prompt you to search for a package

**composer require** vendor/package or package

**composer require --dev** vendor/package or package

adds required packages to your composer.json and installs them

**composer remove**  vendor/package vendor/package2

**composer rm**  vendor/package

**composer uninstall**  vendor/package

removes a package from the require or require-dev

**composer.json**

<https://getcomposer.org/doc/01-basic-usage.md>

<https://getcomposer.org/doc/04-schema.md>

перечисляются установленные пакеты в приложении и настройки (см далее)

! должен быть включен в git

**require**

пакеты необходимые в production

**require-dev**

пакеты необходимые только при разработке

**composer.lock**

фиксирует точные версии всех зависимостей, перечисленных в composer.json

! должен быть включен в git

**vendor** directory

Папка с установленными пакетами.

**! НЕ** должна быть включена в git

**composer check-platform-reqs**

check that platform requirements are satisfied

**composer install**

**composer i**

installs the project dependencies from the composer.lock file if present,

or falls back on the composer.json

imho: команда для production, т.к. скачает версии пакетов указанные в composer.lock,

а не последние как при composer update, которые не тестировались в app.

**composer install --no-dev**

отключает установку пакетов require-dev (при установки app in production)

**composer reinstall** vendor/package vendor/package2

uninstalls and reinstalls the given package names

**composer outdate**

shows a list of installed packages that have updates available

**composer update**

**composer u**

**composer upgrade**

updates your dependencies to the latest version, and updates the composer.lock file

imho: при разработке приложения эту команду лучше использовать вместо composer install,

т.к. установятся последние версии пакетов, если версия не ограниченна в composer.json.

**composer update** vendor/package vendor/package2

update a few packages to the latest version

**composer update** vendor/package**:2.0.1** vendor/package2**:3.0.\***

update a few packages to a specific version

**composer update "**vendor/\***"**

update all vendor packages to the latest version

**composer update --lock**

update composer.lock after changing composer.json without updating package versions

**composer show**

**composer info**

show app packages (e.g. faker)

**composer show -p**

show platform (system) packages (e.g. php-curl)

**composer.json**

так-же содержит настройки приложения

**"autoload": {**

**"psr-4": {**

**"App\\": "src/"**

**}**

**},**

автозагрузка классов приложения, записана по стандарту psr-4, так-же может использоваться

устаревший стандарт psr-0 (не очень понял в чем разница, см. оф. док.)

**composer dump-autoload**

**composer dumpautoload**

dumps the autoloader

при изменении путей в секции "autoload" файла composer.json необходимо переопределить

этой командой массив в файле vendor\composer\autoload\_psr4.php (иначе нов. настр. не раб.)

**composer dump-autoload --optimize**

optimizes PSR0 and PSR4 packages to be loaded with classmaps too, good for production.

для production при изменении путей в секции "autoload" файла composer.json необходимо

переопределить этой командой массив в файле vendor\composer\autoloadet\_classmap.php

**composer clear-cache**

**composer clearcache**

**composer cc**

clears composer's internal package cache

так-же очищается карта классов для автозагрузки PSR-4

**composer validate**

validates a composer.json and composer.lock

!команду необходимо выполнять перед commit-ом для проверки файлов на ошибки

**composer create-project vendor/package dir**

creates new project from a package into given directory, this is equivalent of git clone

**--prefer-dist**

принудительная установка dist(дистрибутив) пакетов - скомпилированных и не

содержащих исходный код для увеличения скорости работы проекта (поведение по умолчанию)

**--prefer-source**

принудительная установка source(исходного) пакетов - содержащих исходный код для

возможности его редактирования

**composer create-project vendor/package dir --no-dev**

отключает установку пакетов require-dev

**composer create-project laravel/laravel laravel8 "8.\*"**

третьем параметром можно передать версию пакета/проекта

**composer config --list**

list configuration settings

https://getcomposer.org/doc/03-cli.md#config

изменить конфигурационные настройки