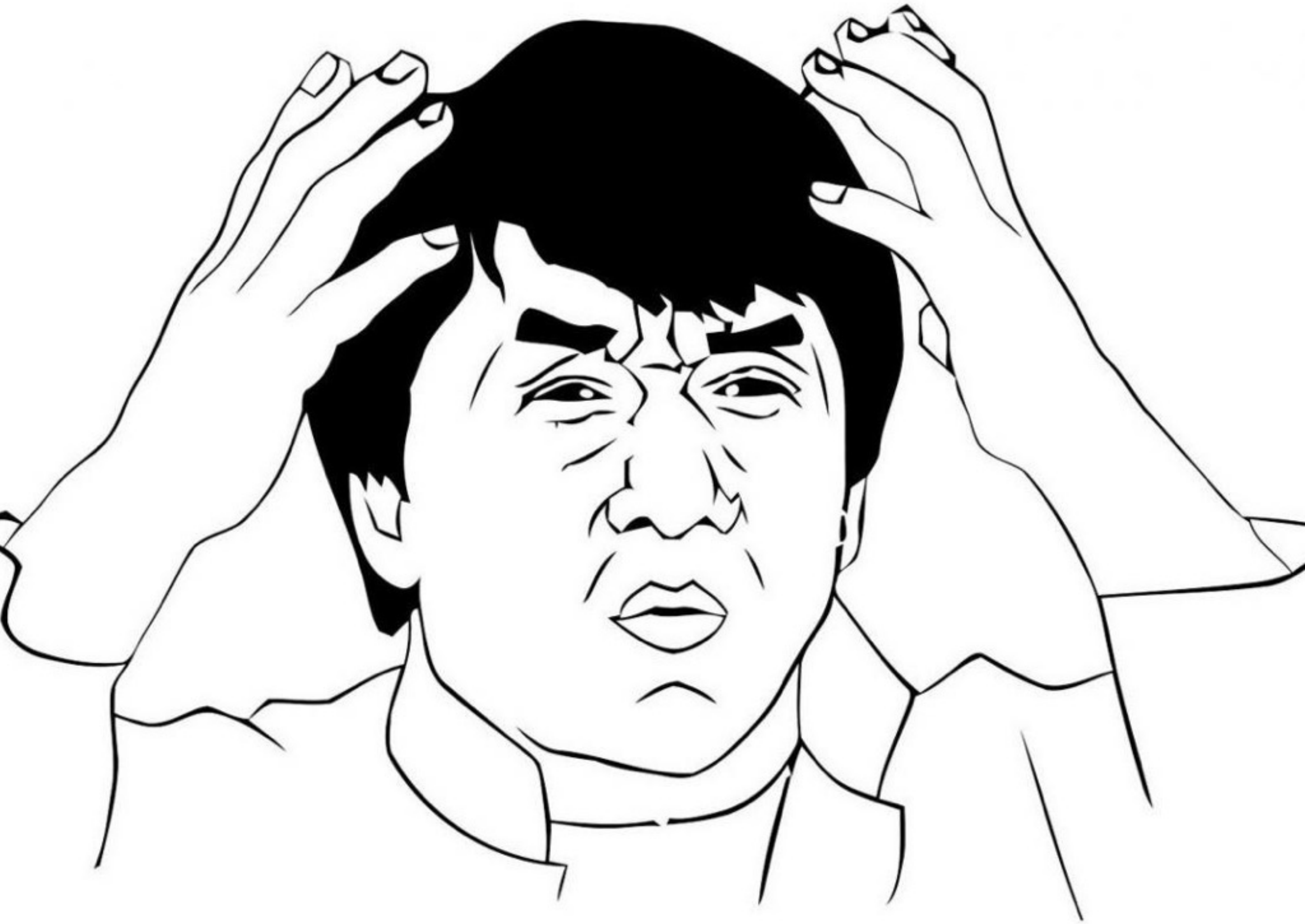
Composer 二三事

@lifesign

Let’s Make Music

What is Composer

“Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.”



What does that mean

- scope is per project, not global
- resolves dependencies
- run installation task

Features

- Autoload
- PSR-0/PSR-4
- Install dependencies & binaries
- Scripts
- Create project from a package
- Installers
- Auth
- Semantic Versioning
- Integrate with git/svn/mercurial/etc
- and more…

Miscellaneous

- 1st release: **1/03/2012**
- Inspired by npm and bundler
- 100% PHP
- use Symfony components
- Autoload

没有 Composer 之前

- PEAR 包管理
- 代码复用率低，Ctrl+C && Ctrl+V
- 自动加载方式混乱

Why Composer

为什么需要 composer

- 统一自动加载方式
- 代码模块化，降低复用的成本
- 更科学的版本更新
- Scrum 敏捷开发

Install Composer

- Require PHP >= 5.3.2
- homebrew
  - $ brew install composer
- 手动下载
  - $ wget https://getcomposer.org/composer.phar
  - $ mv composer.phar /usr/local/bin/composer
  - $ chmod +x /usr/local/bin/composer

Experience

Step 1

Define `composer.json`

```
{
    "require": {
        "monolog/monolog": "1.2.*"
    }
}
```

Step 2

run `composer install`

```
☁  testcomposer  tree -L 2
.
├── composer.json
├── composer.lock
└── vendor
    ├── autoload.php
    ├── composer
    ├── monolog
    └── psr
```

Step 3

Include autoloader

```
## include autoloader

require __DIR__ . ‘/vendor/autoload.php';


## use packages

$log = new Monolog\Logger('name');

$handler = new Monolog\handler\Streamhandler('app.log', ...)

$log->pushHandler($handler);

$log->addInfo(‘Hello Ads’);
```

Basics

Basic Commands

```
$ composer list
```

config:	Set config options
create-project:	Create new project from a package into given directory.
global:	Allows running commands in the global composer dir
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.js
update:	Updates your dependencies to the latest version according to composer.json, and updates the composer.lock

…and more

安装依赖包

- 方式一：
  - 编辑 composer.json && composer install
- 方式二：
  - $ composer require ads/devtools
  - $ composer require ads/devtools:~1.1

更新依赖

- 更新全部：
  - $ composer update -vvv
- 更新指定包：
  - $ composer update ads/devtools -vvv

Metadata

composer.json

- 基础字段：name,description,authors,keywords, license
- 依赖声明字段：require, require-dev
- 自动加载字段: autoload, autoload-dev
  - PSR-4
  - PSR-0
  - classmap
  - files
- 其它字段：scripts, minimum-stability, bin, repositories, support, config

Name

- 格式: “vendor/package-name”
- vendor: 开发者(公司、组织)名称
- package-name: 包名

Description

- 简短的包描述
- 尽量在一行内结束
- 对于需要发布的包（库），这是必须填写的

Authors

- 格式: “name <email>”
- 可以有多个 author，(需要手动加入)
- 默认抓取 ~/.gitconfig 配置

minimum-stability

- 只能用在 root-package
- 过滤依赖稳定性
- 可设定的值(不稳定到稳定)：dev、alpha、beta、RC、stable(默认)

License

- 指定授权
  - Apache-2.0
  - BSD
  - MIT
  - …

Require

- 格式: “vendor/package-name”: “version”
- 指定相关依赖或者平台环境(version, extension…)

Require-Dev

- 格式同 require
- 主要用于指定开发时的依赖
- 只能用在 root-package

Autoload

Autoload

PSR-0

PSR-4

Classmap

Files

PSR-0(废弃)

```
{
    "autoload": {
        "psr-0": {
            "MyNamespace\\": ["src/"]
        }
    }
}
```

Filesystem

```
.
 └──── src
     └──── MyNamespace
         └──── Model
             └──── User.php
```

How to use

```
require __DIR__ .”vendor/autoload.php”;

$user = new MyNamespace\Model\User;
```

PSR-4

```
{
    "autoload": {
        "psr-4": {
            "MyNamespace\\": ["src/"]
        }
    }
}
```

Filesystem

```
└──── src
    └──── Model
        └──── User.php
```

How to use

```
require __DIR__ .”vendor/autoload.php”;

$user = new MyNamespace\Model\User;
```

Classmap

```
{
    "autoload": {
        "classmap": [
            "src/", "lib/", "DB.php"
        ]
    }
}
```

Filesystem

```
.
├── DB.php
├── lib
└── src
```

How to use

```
require __DIR__ .”vendor/autoload.php”;

$db = new DB;
```

Files

```
{
    "autoload": {
        "files": ["classes/DB.php"]
    }
}
```

Filesystem

```
.
 └──── classes
    └──── DB.php
```

How to use

```
require __DIR__ .”vendor/autoload.php”;

$db = new DB;
```

Don’t forget dump-autoload

```
$ composer dump-autoload
```

Dependencies

语义化版本
(Semantic Versioning)

http://semver.org/

版本号组成

- MAJOR.MINOR.PATCH
  - MAJOR：通常会发生 api 变更，不向后兼容
  - MINOR：新增功能，向后兼容
  - PATCH：补丁，向后兼容，修复bug

Release Operators

- “~”: 指定向后兼容的最小版本(版本号倒数第二个数+1)
  - ~1.2 等于 >=1.2 && <2.0.0
  - ~1.2.3 等于 >=1.2.3 && <1.3.0
- “^”: 允许大版本前的所有版本
  - ^1.2 等于 >=1.2 && <2.0.0
  - ^1.2.3 等于 >= 1.2.3 && < 2.0.0 (区别在这里)

版本号使用总结

- 精准匹配：1.2.3
- 范围操作：>= 1.0，>=1.0 <2.0，>=1.0 <1.5 || >=2.0
- 连字符：1.0 - 2.0
- 通配符：2.0.*
- 波浪运算符：~1.5
- ^ 运算符：^1.5

版本锁定

- 版本锁定文件: composer.lock
- `composer install` 会生成锁定文件
- install 时若存在 lock文件，优先读取 lock 文件中的依赖版本
- composer update 会更新锁定文件
- 是否应该加入版本控制
  - 团队内部 (application): yes
  - 工具类库 (package): no

Composer Workflow

新项目

- 创建 composer.json，并添加依赖的扩展包定义
- 运行 `composer install`, 安装扩展包生成 composer.lock
- 提交 composer.lock 到代码版本控制中

项目协作者

- 克隆项目后，根目录下执行 `composer install`，从 composer.lock 中安装**指定版本**的扩展包及其依赖

为项目添加新扩展包

- 使用 `composer require vendor/package` 添加扩展包

- 提交更新后的 composer.json 和 composer.lock 到代码版本控制器中

Package Development

如何创建一个包?

```
$ mkdir my-package

$ cd my-package

$ composer init
```

测试已经创建的包

- $ mkdir my-package-test
- $ composer init
- 添加 repositories 项目
  - type 为 path
  - url 使用相对路径指向 my-package
- 在 require 中 添加 youname/my-package:*
- $ composer install

Problems

- packagist 镜像在国外，间歇性被墙
- composer.phar 文件下载本身也是被墙的
- 企业内部代码，不想上传到公有库
- …

Composer 国内镜像

- 全局(推荐)
  - composer config -g repo.packagist composer https://packagist.phpcomposer.com
- 修改项目的 composer.json
  - composer config repo.packagist composer https://packagist.phpcomposer.com

内网镜像

- http://toranproxy.com/
- https://github.com/composer/satis

Tips

require-dev

```
{
    "require-dev": {
        "phpunit/phpunit": "4.1.*",
        "satooshi/php-coveralls": "dev-master"
    }
}
```

Install with dev dependencies

```
$ composer install --dev
```

Install without dev dependencies

```
$ composer install --no-dev
```

Load Local Package

```
{
    "require": {
        "ads/baymax": "dev-master"
    },
    "repositories": [
        {
            "type": "vcs",
            "url": "git@git.culiu.org:ad-system/baymax.git"
        }
    ]
}
```

Update Only Lockfile

```
$ composer update --lock
```

Composer In Production

```
$ composer dump-autoload -o
```

Composer Parallel Install

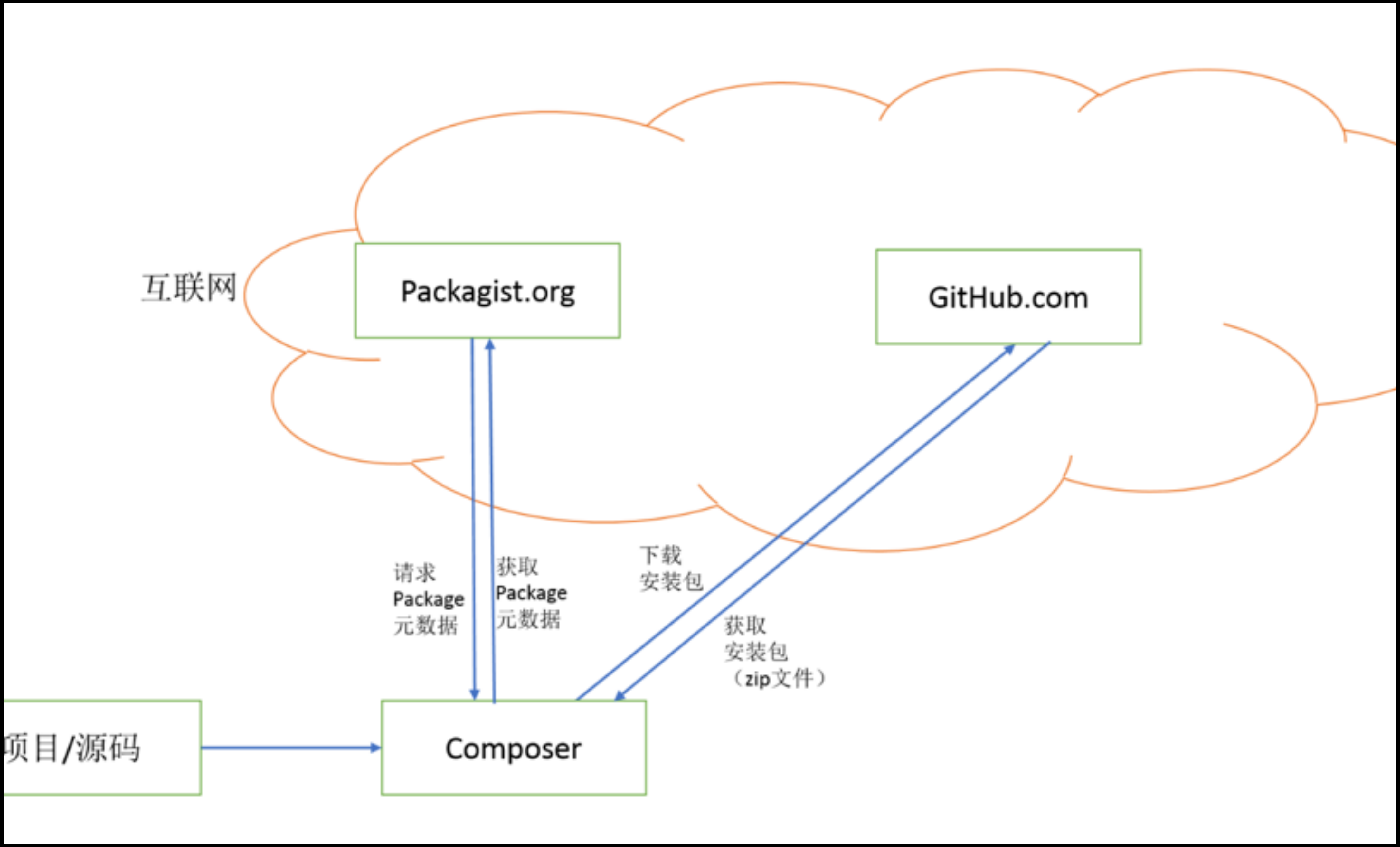
```
$ composer global require "hirak/prestissimo:^0.3"
```

benchmark: laravel install 288s -> 26s

Shorten Command

- composer g
- composer u
- …

Advance Composer


互联网
Packagist.org
GitHub.com
请求 Package 元数据
获取 Package 元数据
下载 安装包
获取 安装包 （zip文件）
项目/源码
Composer

Composer 架构

- composer scripts
- composer plugins
- composer global install
- …

参考资料

- http://getcomposer.org/
- http://packagist.org/
- http://semver.org/
- http://www.php-fig.org/
- http://www.phpcomposer.com/
- http://toranproxy.com/
- https://github.com/composer/satis

Q&A