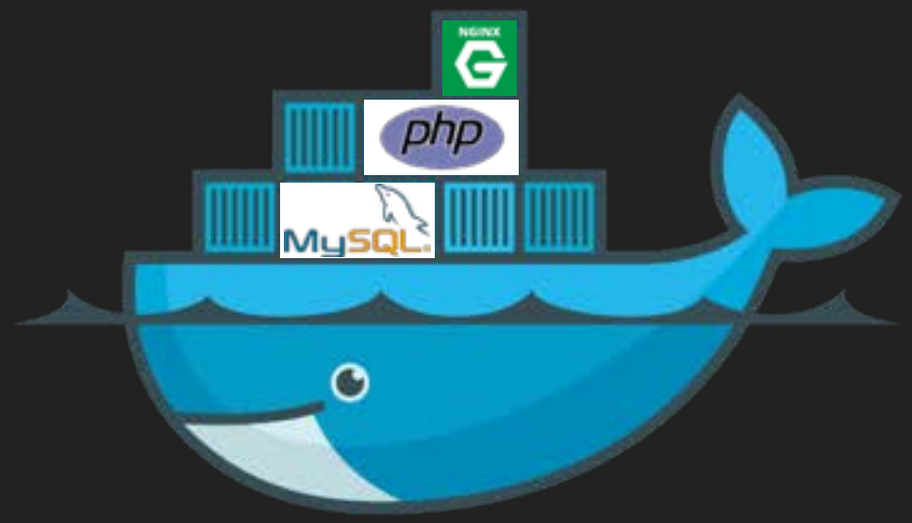


利用Docker优化PHP开发流程

周悦秋@好雨云

ABOUT ME

周悦秋　好雨云 联合创始人

10余年工作经验
混过看雪论坛
写过C++程序
曾任澳客网系统架构师、基础技术部总监

目前专注云计算方向的技术研究

对k8s、docker、自动化构建等技术有丰富的应用经验

PHP开发流程中的问题

如何借助Docker解决问题

PHP开发流程中的问题

环境问题

测试问题

发布问题

如何借助Docker解决问题

环境问题

开发环境问题

部署

Apache(Nginx)

rewrite规则
第三方模块
虚拟主机配置
gzip压缩
缓存时间
……

PHP

PHP版本
PHP参数配置
安装扩展（配置）
缓存管理
……

MySQL

MySQL版本
参数配置
存储引擎
缓存配置
索引优化
……

开发环境问题

环境不一致

运维小鸟

小明，你上的新代码有问题，线上出事故了！用户注册页面白屏了

😨 我勒个去，我代码测试环境好好的啊。而且代码一样，QC都测过的！

PHP开发－小明

运维小鸟

扯~，线上日志都报出来了，却少XX方法。

线上是不是没安装XX扩展?

PHP开发－小明

运维小鸟

你怎么不早说，我现在装一下，本次事故也有你的份

&^%%^#CAO

PHP开发－小明

Docker 如何解决**开发环境**问题

Docker 如何解决开发环境问题

代码即环境

代码即环境：用代码来定义环境

代码即环境：用代码来定义环境

简单安装，灵活配置

彻底解决环境不一致问题

代码即环境：Demo

Demo: PHP命令行

文件结构

```
.
├── Dockerfile
└── your-script.php
```

Dockerfile 内容

```
FROM php:5.6-cli
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
CMD [ "php", "./your-script.php" ]
```

打包镜像并运行程序

```
$ docker build -t my-php-app .
$ docker run -it --rm my-php-app
```

直接运行php程序

```
$ docker run -it --rm -v "$PWD":/usr/src/myapp \
-w /usr/src/myapp php:5.6-cli php your-script.php
```

Demo: PHP+Apache

文件结构

```
.
├── Dockerfile
└── src
    └── index.php
```

Dockerfile 内容

```
FROM php:5.6-apache
COPY src/ /var/www/html/
```

打包镜像并运行

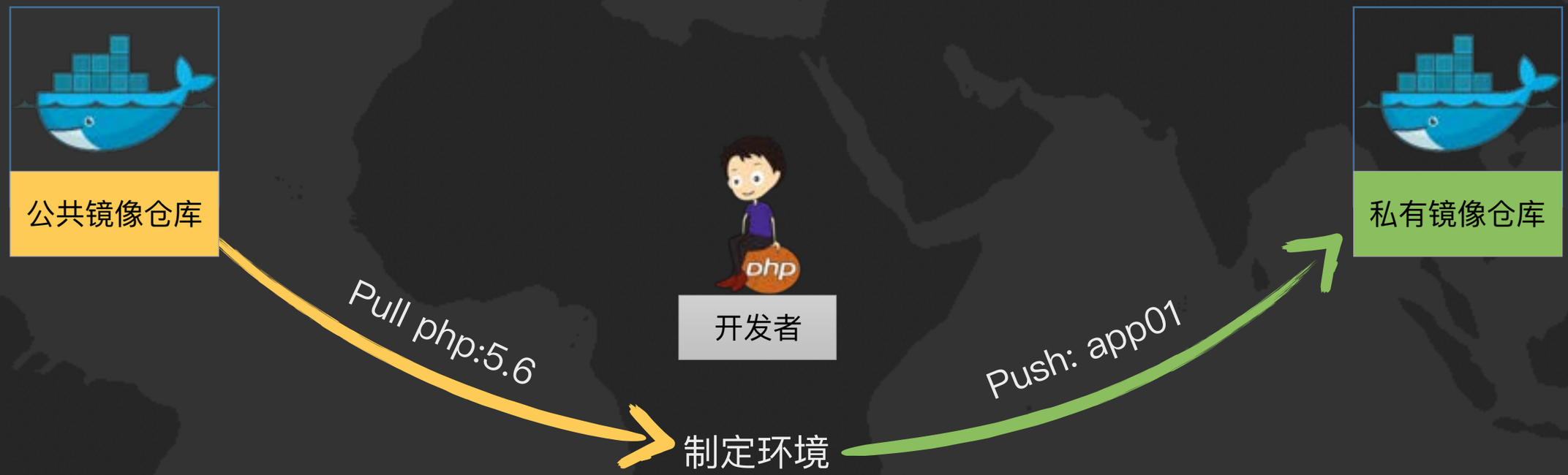
```
$ docker build -t my-php-app .
$ docker run -it --rm my-php-app
```

自定义 php.ini 文件

```
FROM php:5.6-apache
COPY config/php.ini /usr/local/etc/php/
COPY src/ /var/www/html/
```

代码即环境：镜像仓库

将打包的环境与代码推到镜像仓库




```
docker pull php:5.6-apache
docker build -t app01
docker push app01
```

PHP开发流程中的问题

环境问题

测试问题

发布问题

如何借助Docker解决问题

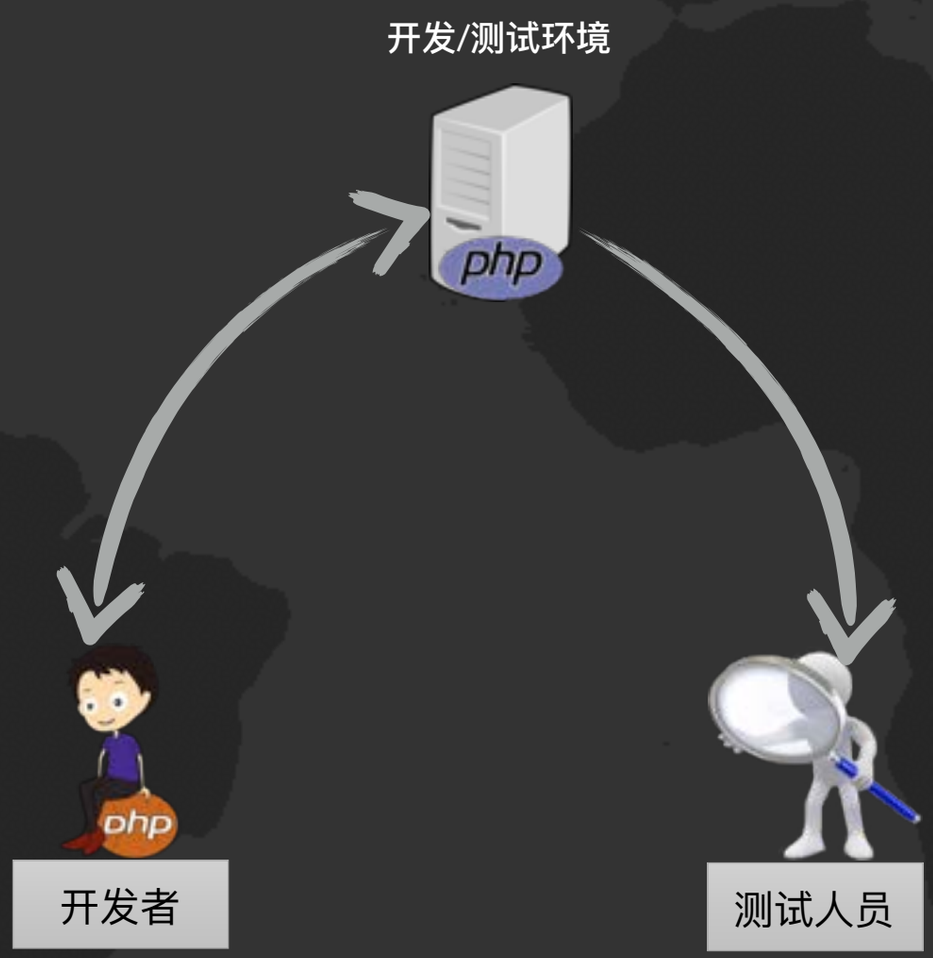
测试问题

测试环境

压力测试

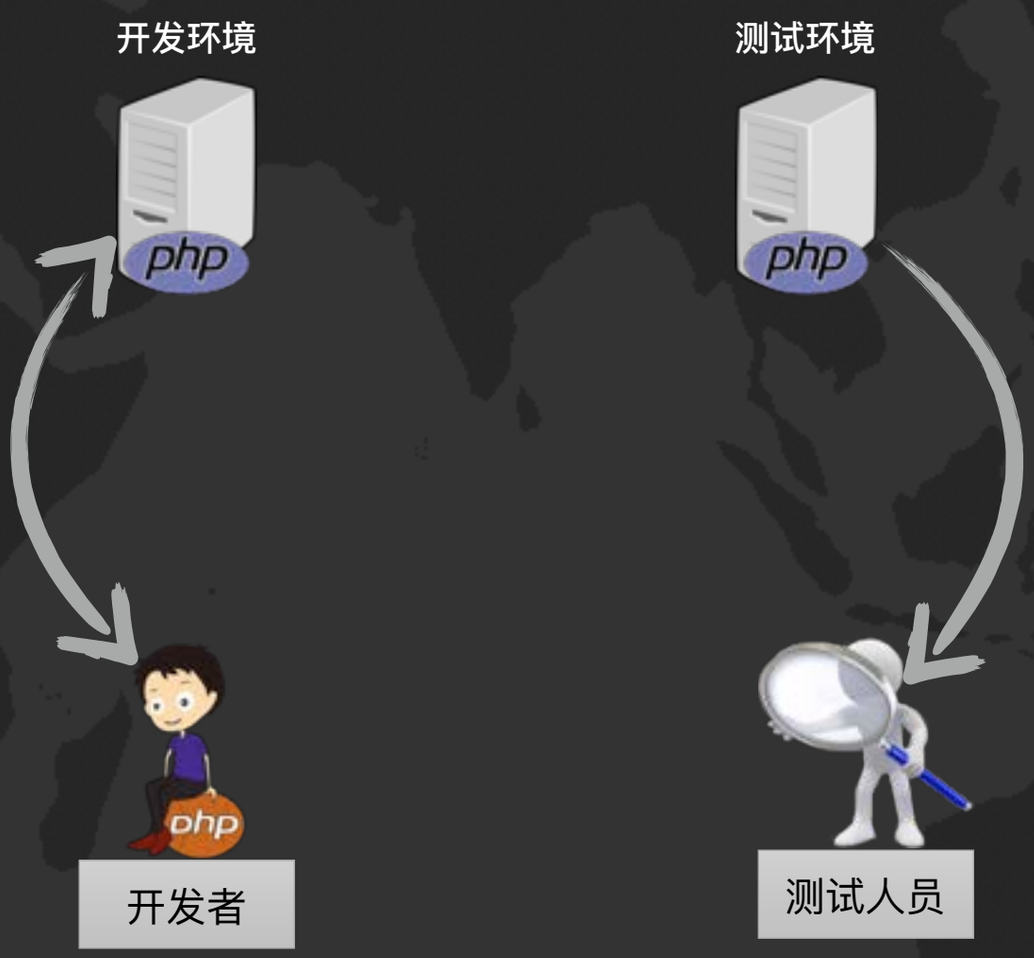
测试环境

开发与测试环境混用



开发与测试相互影响

开发与测试环境分离



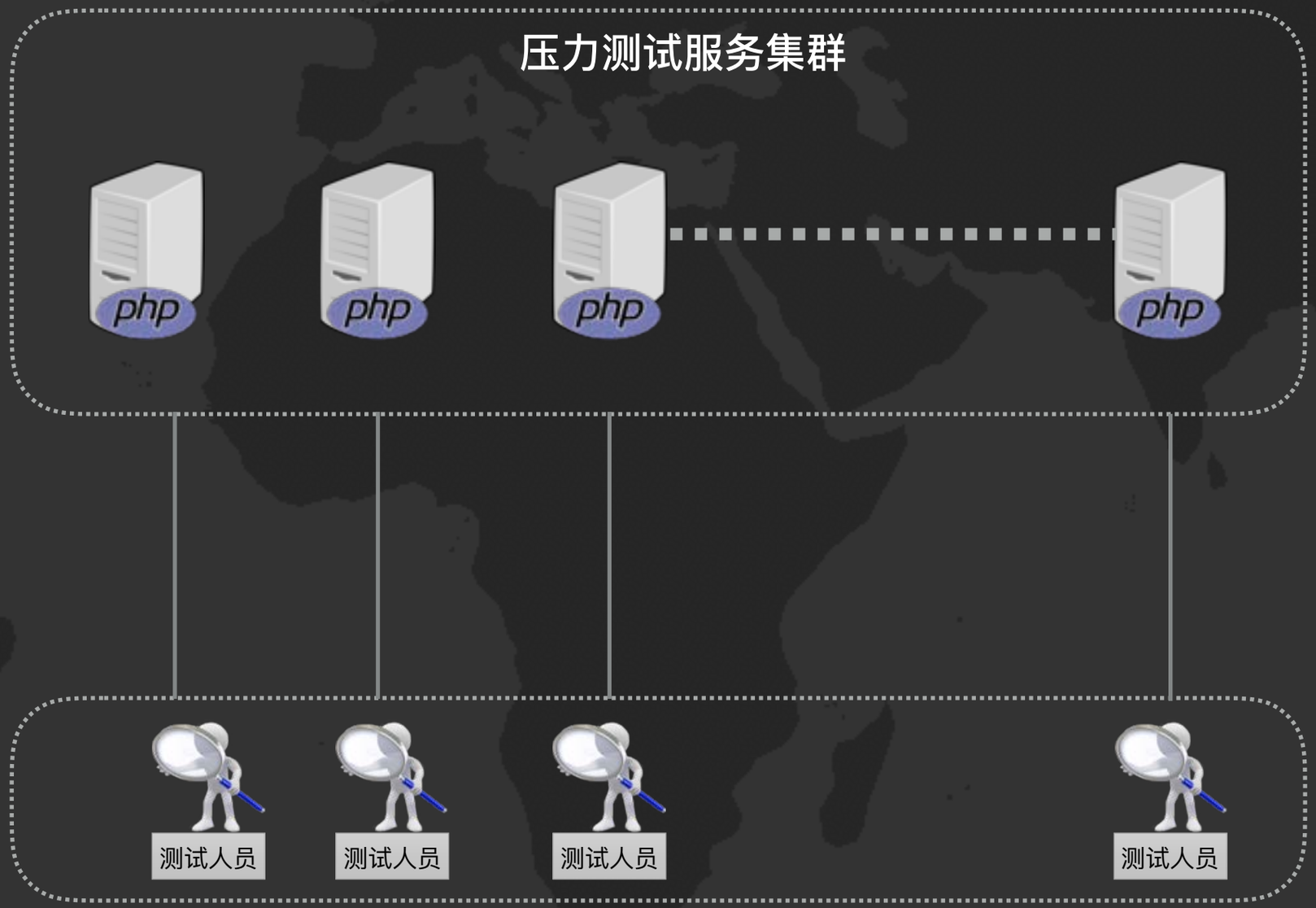
开发与测试环境一致性问题

测试问题

测试环境

压力测试

压力测试



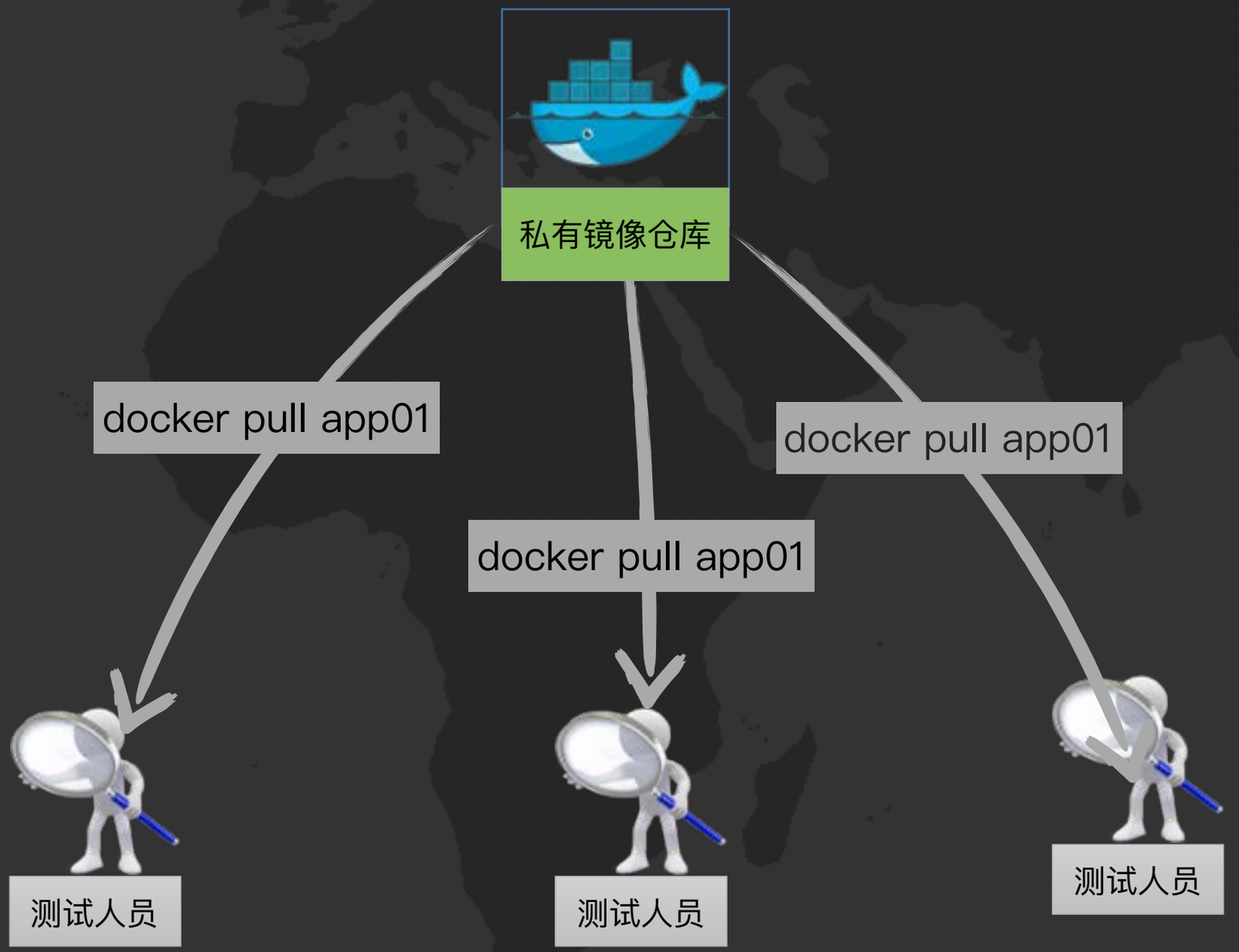
如何快速扩展测试环境

Docker 如何解决测试问题

Docker 如何解决测试问题

环境复用，秒级批量部署

环境复用



测试人员可以方便的拉取开发者推送的环境

秒级批量部署

私有镜像仓库

docker pull app01

压力测试服务集群

测试人员

测试人员

测试人员

可以批量拉取app01并立即启动

PHP开发流程中的问题

环境问题

测试问题

发布问题

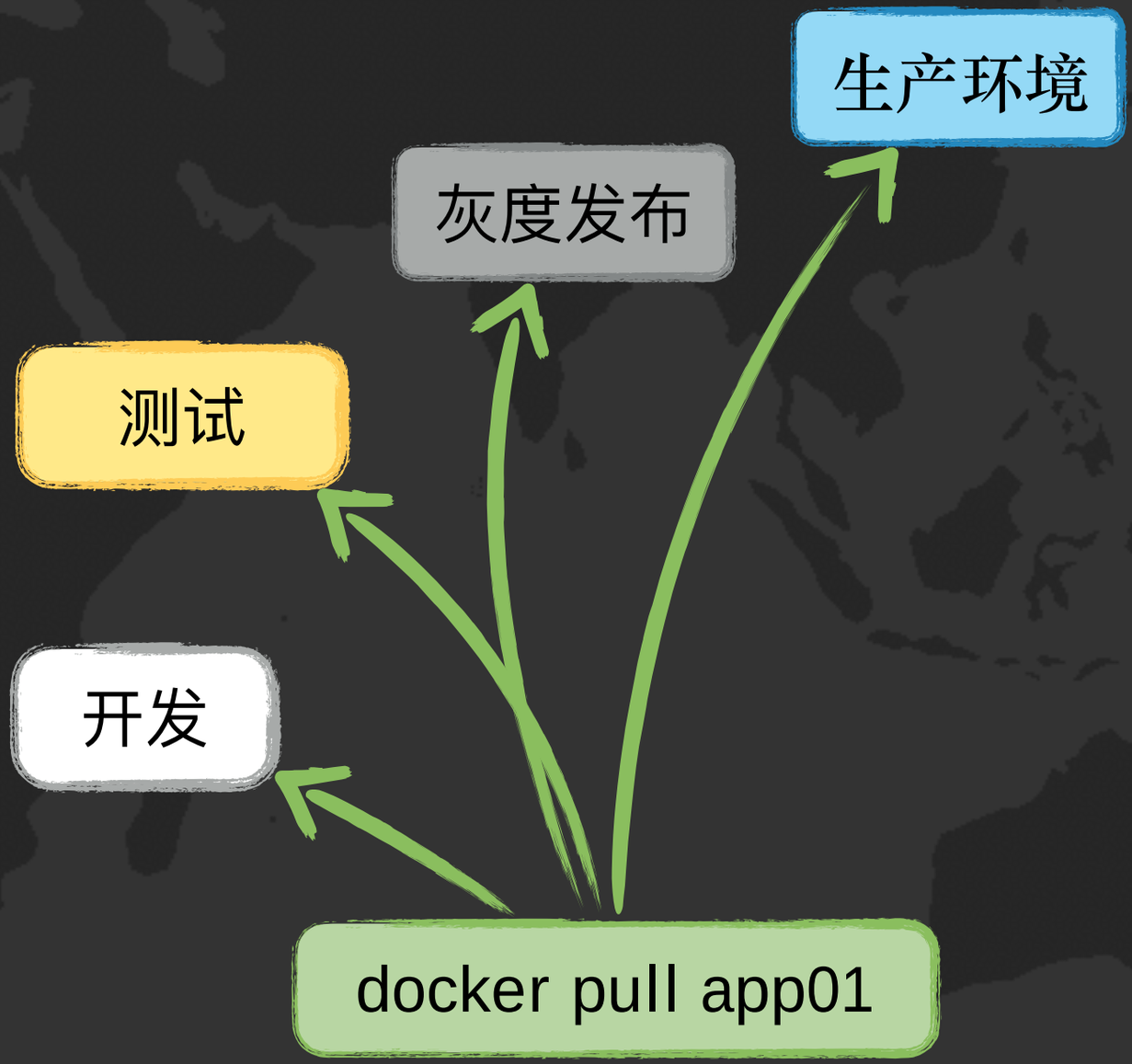
如何借助Docker解决问题

发布问题

传统方式

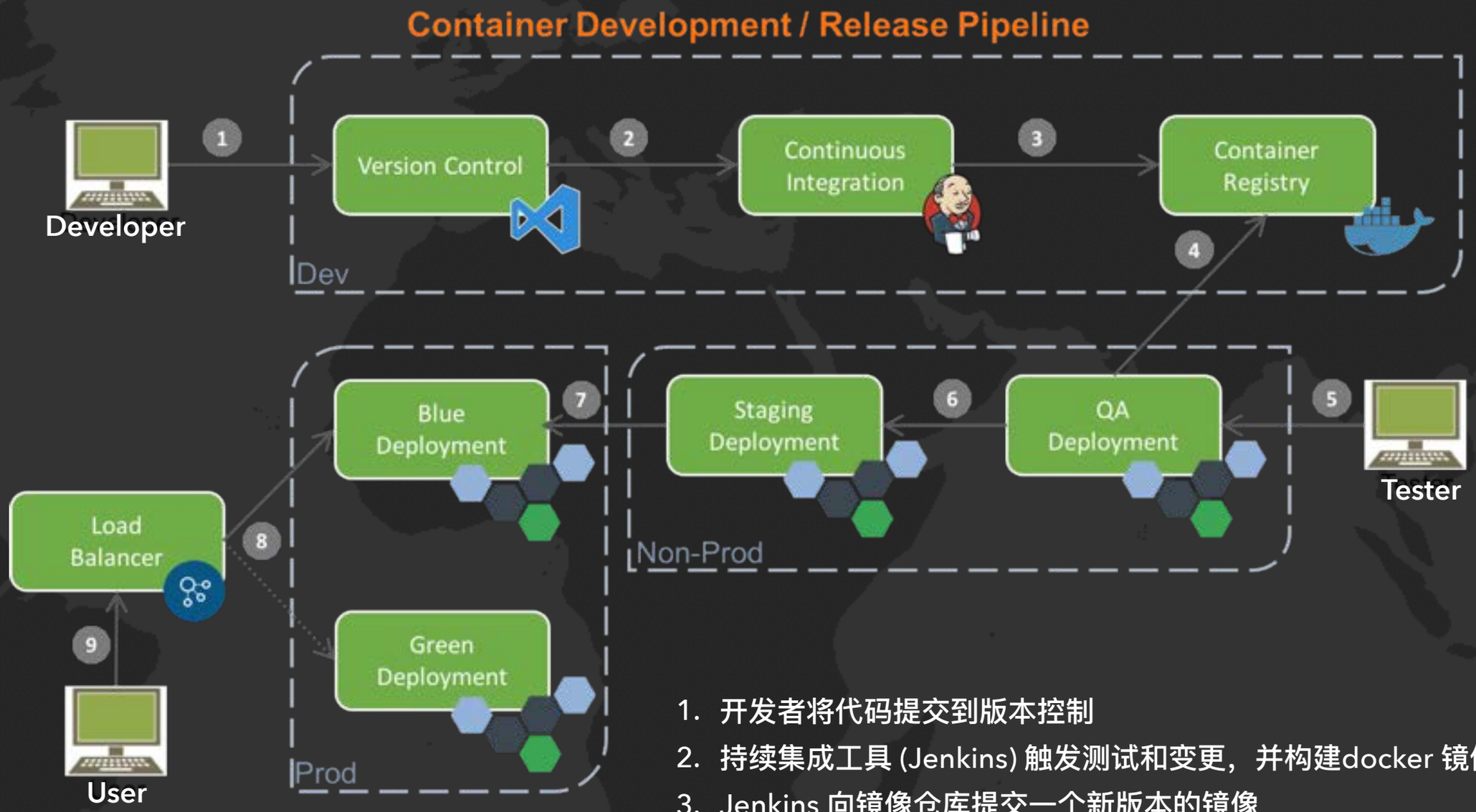
人工（半自动）发布，效率很差

Docker方式



秒级发布到所有场景

Docker化的开发流程



1. 开发者将代码提交到版本控制
2. 持续集成工具 (Jenkins) 触发测试和变更，并构建docker 镜像
3. Jenkins 向镜像仓库提交一个新版本的镜像
4. QA 服务器端从镜像仓库拉取镜像，运行并测试
5. 开始在非生产环境 / 生产环境进行测试
6. 如果测试通过，预发布环境（Staging Deployment）会拉取这个镜像
7. 如果测试通过， “Blue” 生产环境 (非在线) 会拉取这个镜像
8. 镜像最终在 “Green” 生产环境发布，所有用户都可以访问到新的变更

总结

Docker 不是技术的创新，而是一种新的思维方式

Docker 表面改变了运维方式，实质改变了交付方式

- 封装：代码与环境合体 —— 应用
- 发布：像发布代码一样发布应用（代码＋环境）
- 分发：一次构建，到处运行

“干货”分享

12因素法则

为软件交付提供了一整套的方法论

- 使用**标准化**流程自动配置，从而使新的开发者花费最少的学习成本加入这个项目。
- 和操作系统之间尽可能的**划清界限**，在各个系统中提供**最大的可移植性**。
- 适合**部署**在现代的**云计算平台**，从而在服务器和系统管理方面节省资源。
- 将开发环境和生产环境的**差异降至最低**，并使用**持续交付**实施敏捷开发。
- 可以在工具、架构和开发流程不发生明显变化的前提下实现**扩展**。

配置文件一致性问题、进程模型解决大并发问题、将日志作为事件流……

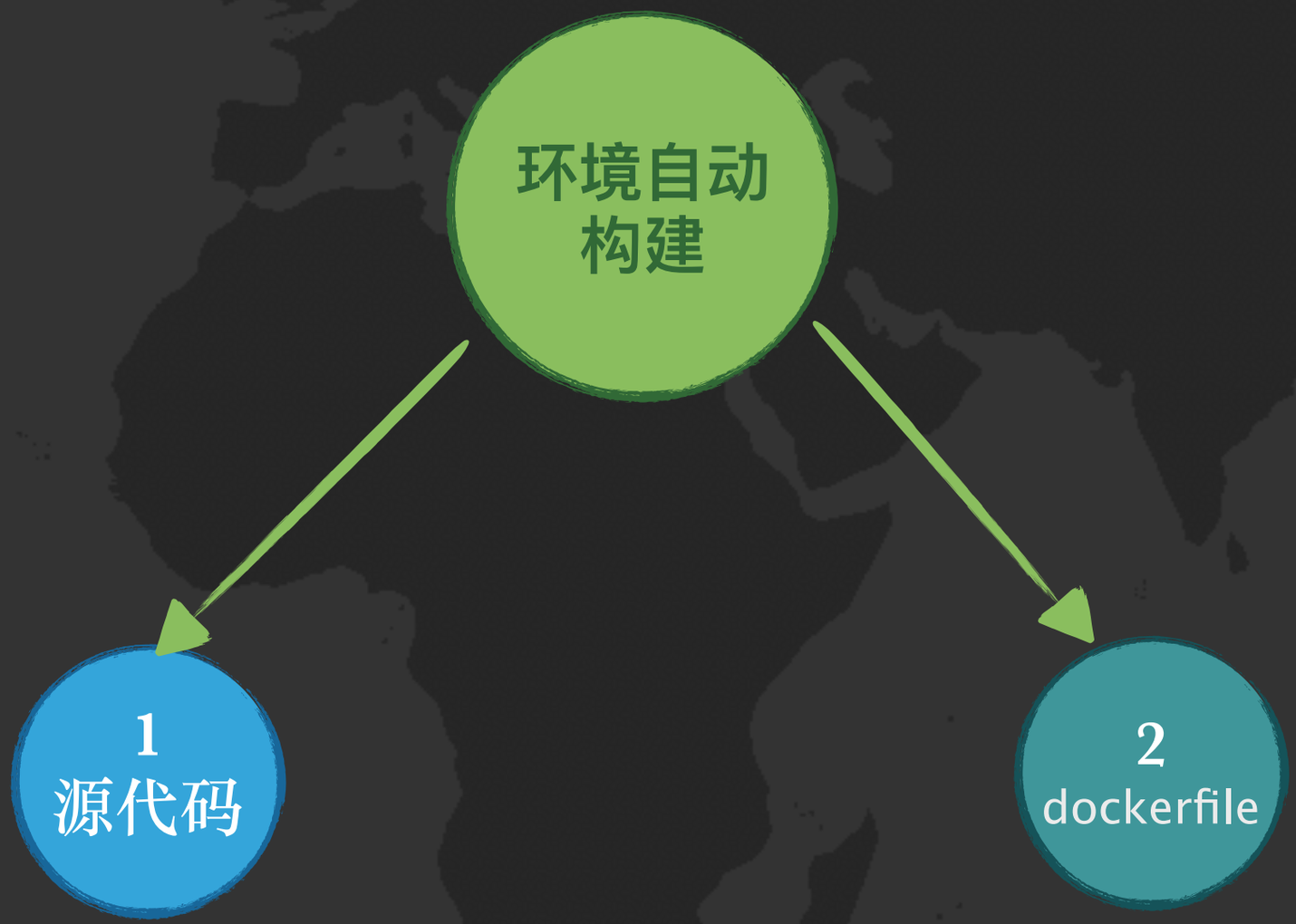
http://12factor.net/zh_cn/

好雨产品遵循12因素法则

实现 产品快速交付、高效运维

产品快速交付

Build



源码环境自动构建

Builder 可执行镜像

PHP 环境构建脚本

+

Composer

App镜像（环境＋代码）

参考文章：可执行镜像--开发环境的Docker化之路（InfoQ）

Builder 可执行镜像

composer.json

```
{
    "name": "laravel/laravel",
    "description": "The Laravel Framework.",
    "require": {
        "php": ">=5.5.9",
        "ext-bcmath": "*",
        "ext-memcached": "*",
        "ext-mongo": "*",
        "ext-xsl": "*",
        "laravel/framework": "5.1.*"
    },
    "require-dev": {
        "fzaninotto/faker": "~1.4",
        "mockery/mockery": "0.9.*",
        "phpunit/phpunit": "~4.0",
        "phpspec/phpspec": "~2.1"
    },
    "autoload": {
        "classmap": [
            "database"
        ],
        "psr-4": {
            "App\\": "app/"
        }
    },
    "autoload-dev": {
        "classmap": [
            "tests/TestCase.php"
        ]
    },
    "config": {
        "preferred-install": "dist"
    }
}
```

1

2

PHP 环境构建脚本

PHP 基础环境/扩展

Composer

保存镜像

示例

文件结构

```
.
├── composer.json
├── index.php
└── Procfile
```

PHP版本
扩展描述

composer.json

```
{
    "require": {
        "php":"5.5.26",
        "ext-bcmath": "*",
        "ext-memcached": "*",
        "ext-mongo": "*",
        "ext-xsl": "*"
    }
}
```

Web Server

Procfile

```
web: vendor/bin/php-apache
```

谢谢

好雨公众号

我的微信