



# 当R语言 遇到容器云

赋能R语言开发者新的开发和交付形式

报告人：汤怡玮

01

容器与容器云

02

容器化的R

03

PaaS下的Shiny容器集群

04

业务与R框架分离

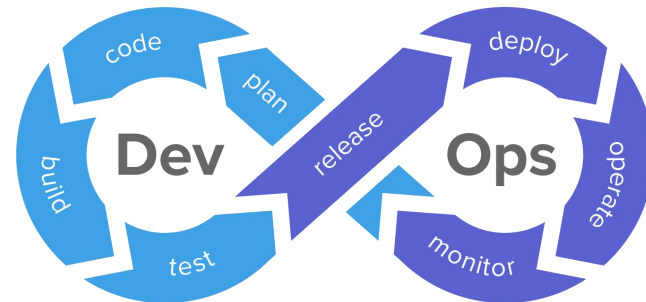


“

容器云让中心IT的角色从“工厂式”交付朝着“服务提供商”的方向发展。

## DevOps

DevOps的提出，打通了“软件开发人员（Dev）”和“IT运维技术人员（Ops）”之间的壁垒，使得构建、测试、发布软件能够更加地快捷、频繁和可靠。



## 什么是 容器

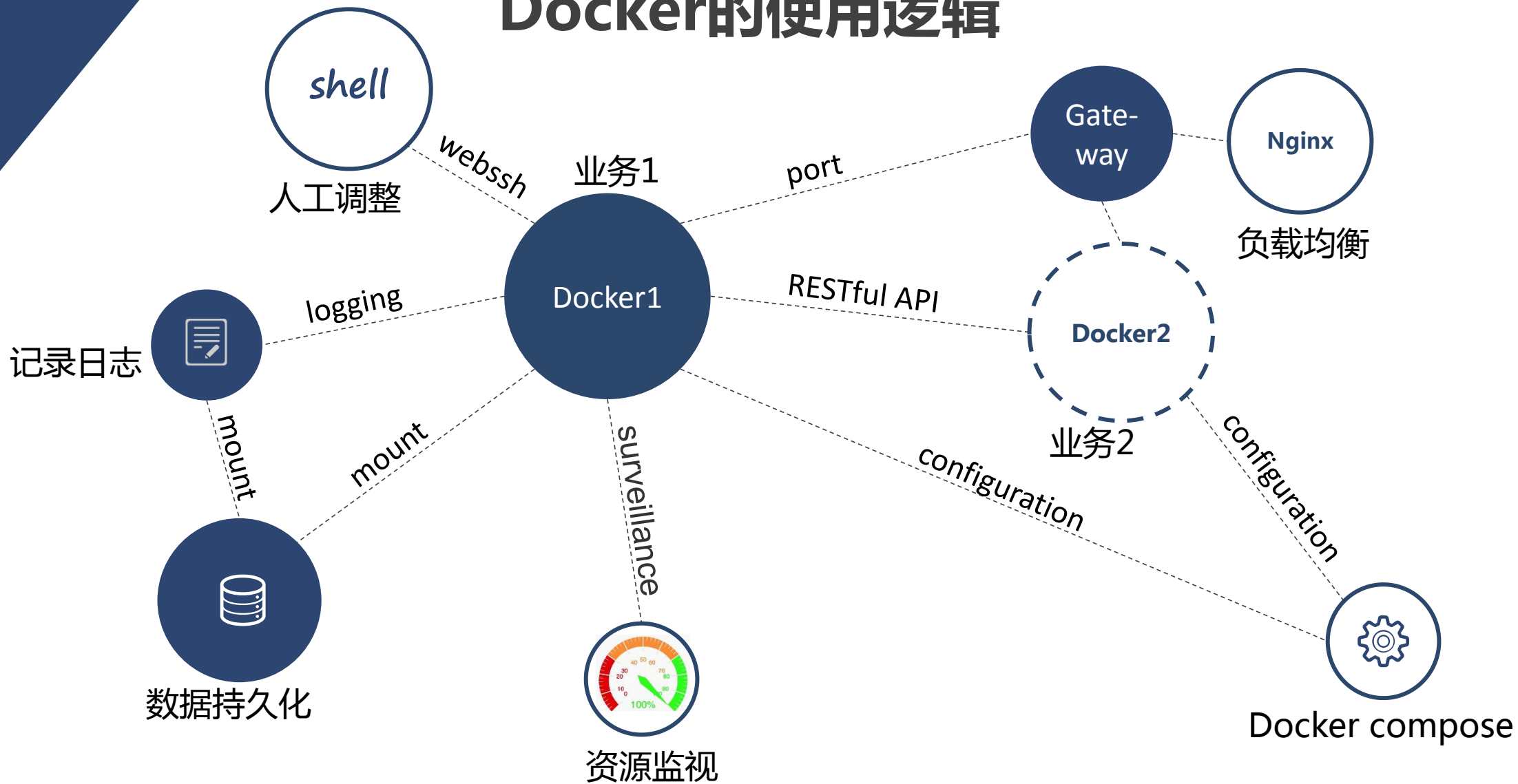
应用程序的“集装箱”

## 容器的 标准化

Docker  
Anywhere

- 1) 使用了容器技术  
——服务之间互相隔离、弹性调整资源配置
- 2) 只打包了必要的依赖环境  
——轻量级、易于移植、跨平台通用
- 3) 标准化的镜像构建流程  
——易于开发、快速部署

# Docker的使用逻辑





# 容器云PaaS的优势

让开发者更加专注代码本身，  
剩下的交给PaaS即可，大大降低了docker的使用门槛。



## 无需硬件运维

硬件层与PaaS层分离



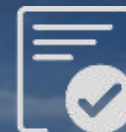
## 更少的代码

只有选项式配置



## 双向弹性部署

资源弹性&容器数量弹性


















## 易于合作开发维护

能够设置合作者权限&直接拉取repo

好的容器云Paas能提供怎样的服务？

<https://app.paas.cloudfraft.cn>



 MyApp	Running	 211	 4/15
 MariaDB		 93	 1/1
 Nginx		 53	 1/4
 Tomcat		 65	 1/6
 Docker		 72	 1/8



合作  
更易重复

为什么  
R要容器化？

更新  
自主隔离

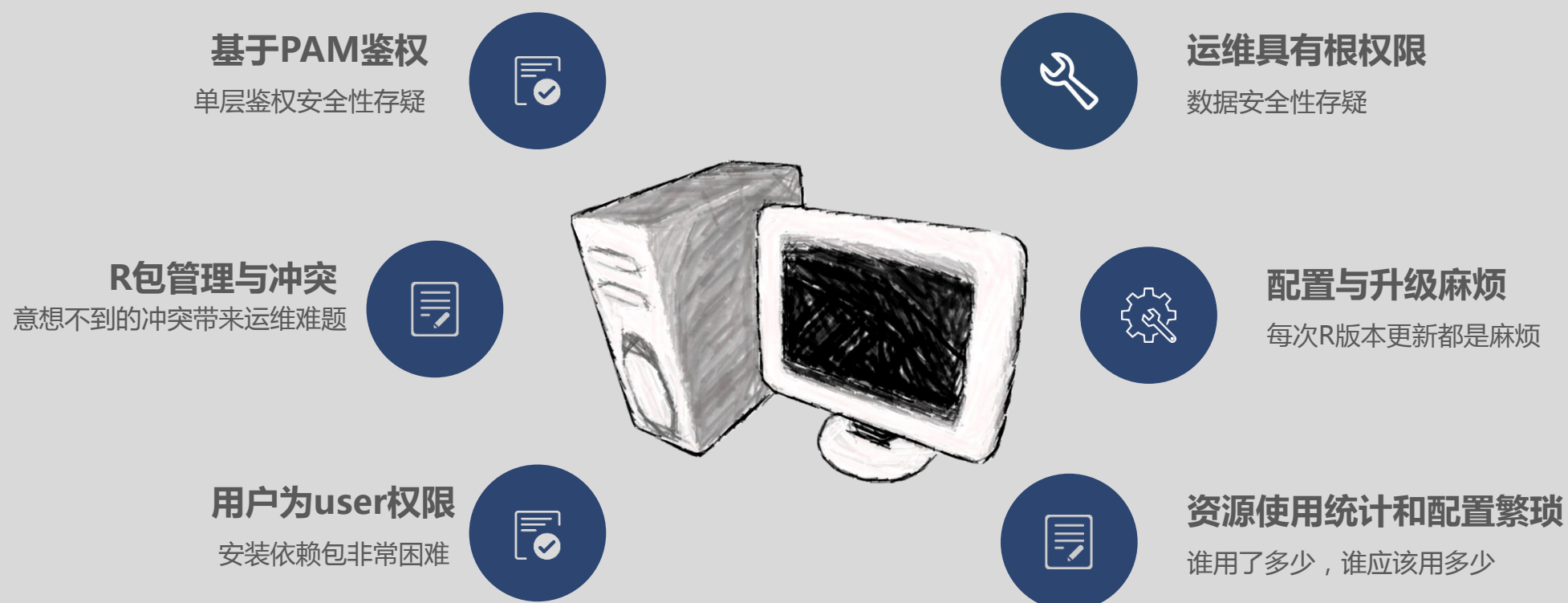


大数据概念火热的背景下，数据分析任务繁重，  
诸多挑战也随之而来

- R的版本更新迭代较为频繁，依赖环境多而杂
- R包之间存在冲突，有些包的安装需要root权限
- 项目不再局限于单人或本地团队，线上的协同开发也受到重视
- 有限资源的统筹利用

Rstudio server

# 过去的R共享开发环境



# 容器化的R共享开发环境





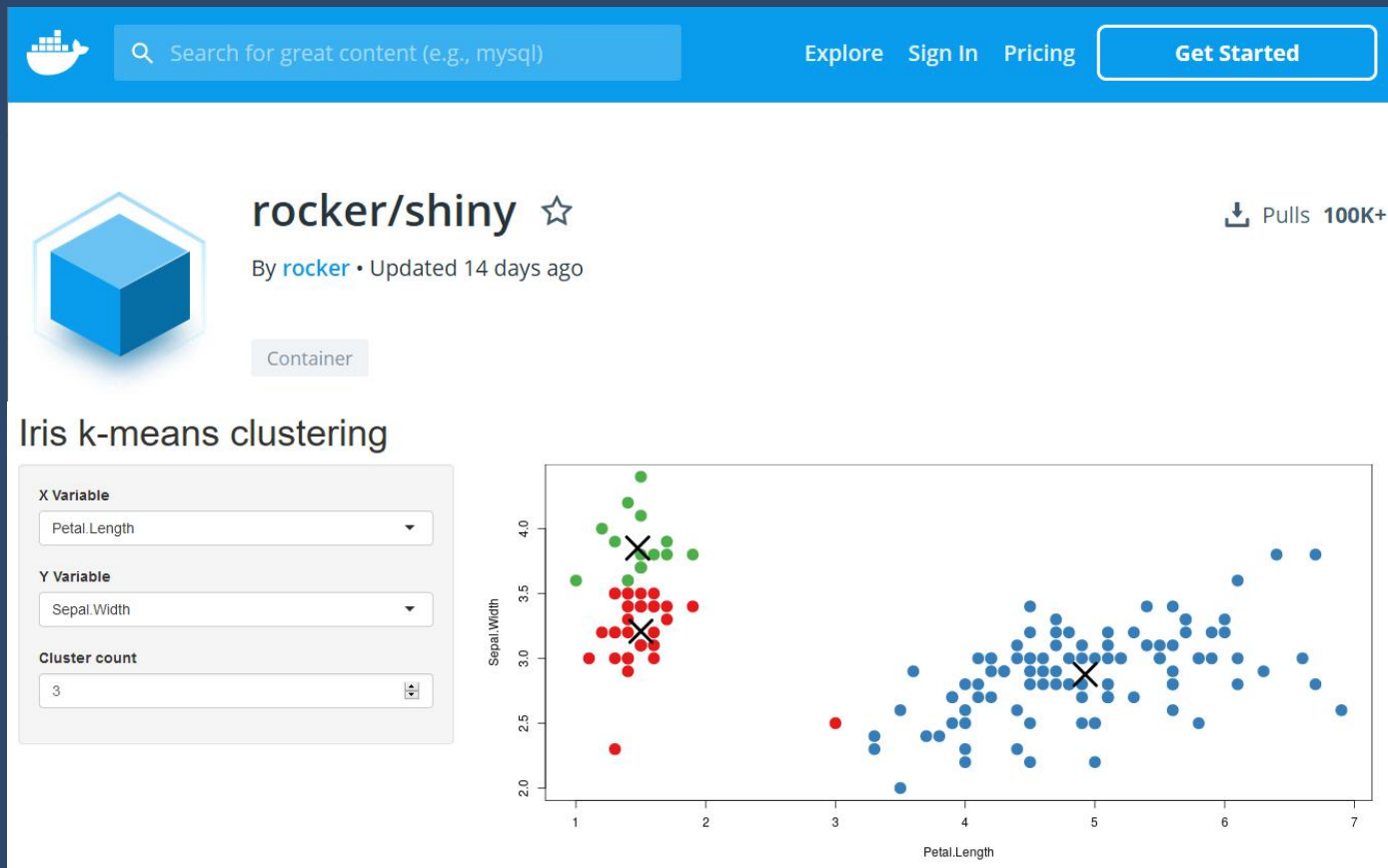
## Shiny作为产品的优势

- 1) 基于R代码，降低web设计的门槛
- 2) 数据可视化
- 3) 交互式输入与交付

## Shiny的容器化与PaaS——从数据分析工厂到提供数据分析服务

## 自建Shiny的缺点

- 1) 代码托管
- 2) 运维要求高，版本更替麻烦
- 3) 手动持续集成
- 4) 非高可用



# PaaS下的Shiny容器集群

Git托管代码实现协作开发

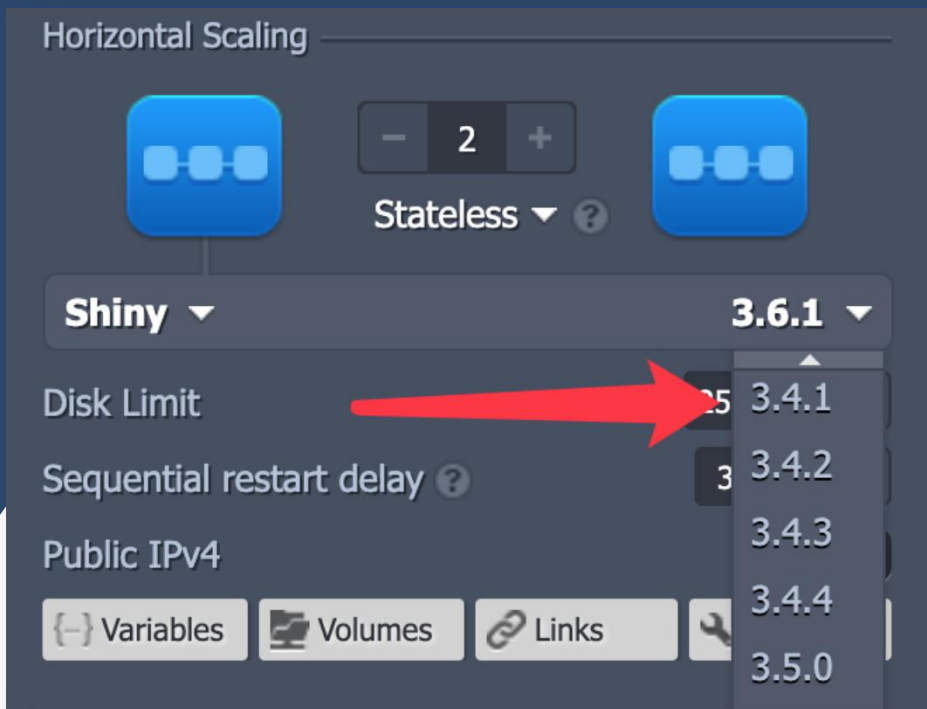
-> 开发/迭代过程持续交付



# PaaS下的Shiny容器集群

PaaS平台基于docker镜像  
部署Shiny集群

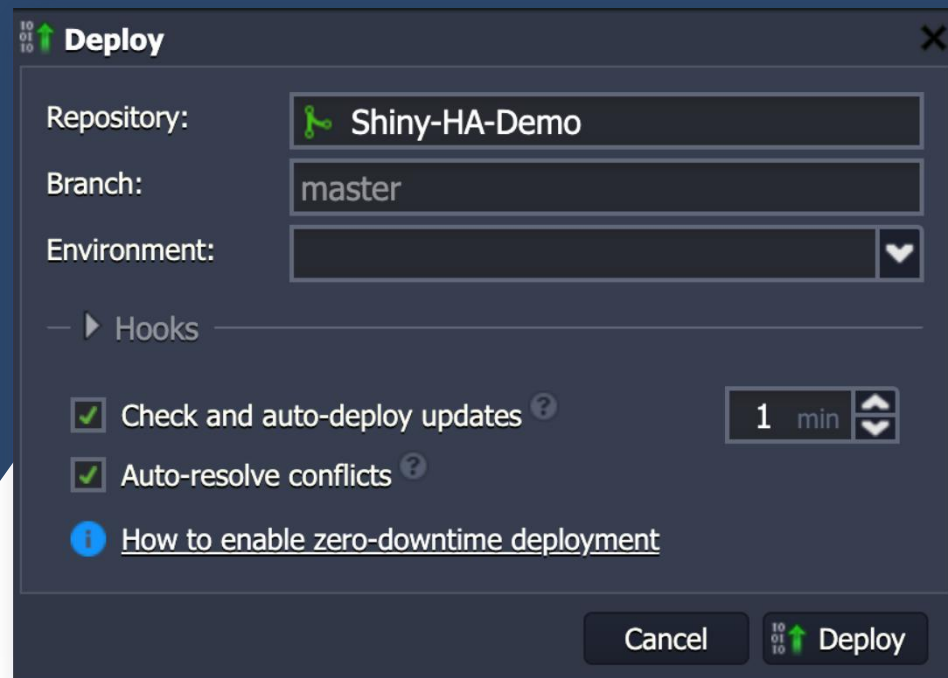
-> 无需运维，自动部署依赖环境





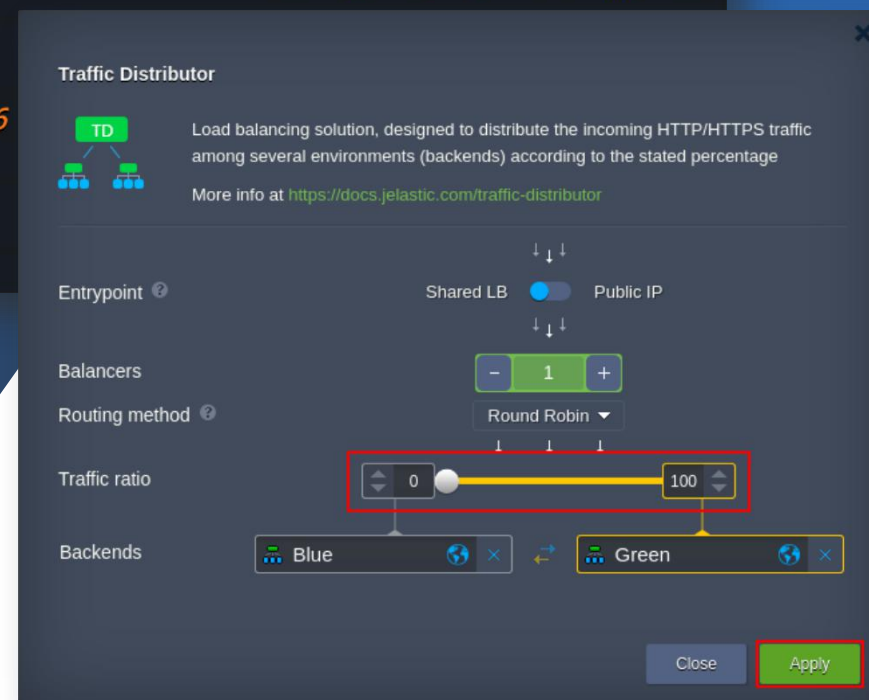
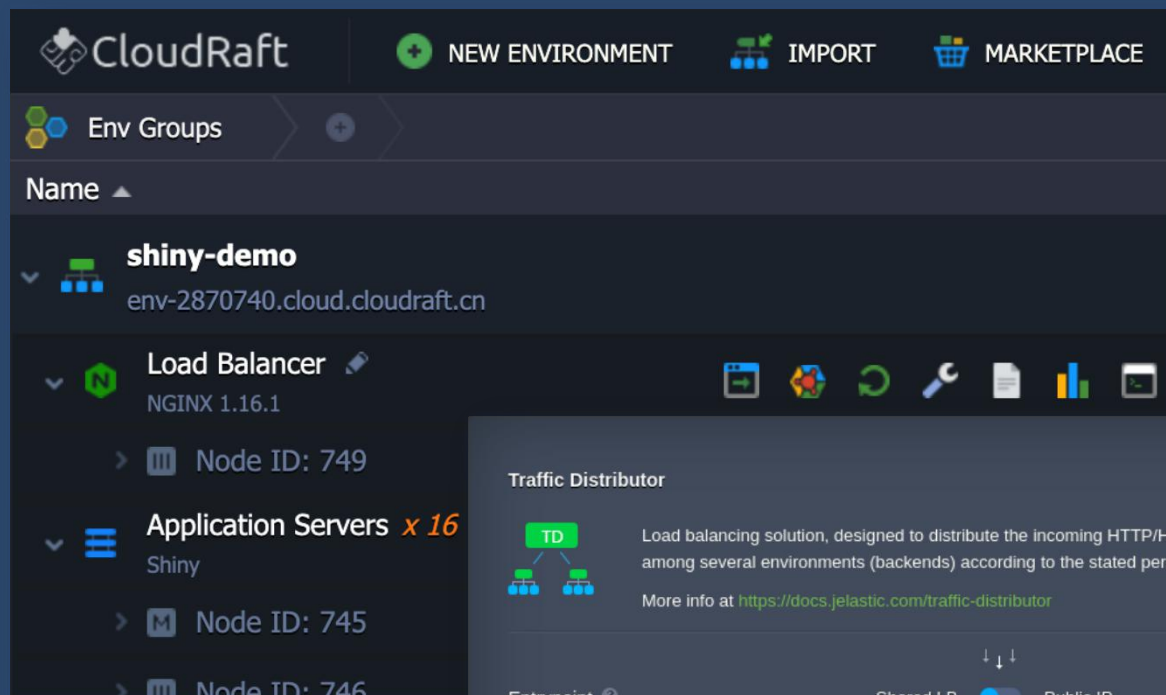
# PaaS下的Shiny容器集群

自动拉取Git中的Shinyapp代码在  
所有Shiny容器中进行部署  
->无需人工干预

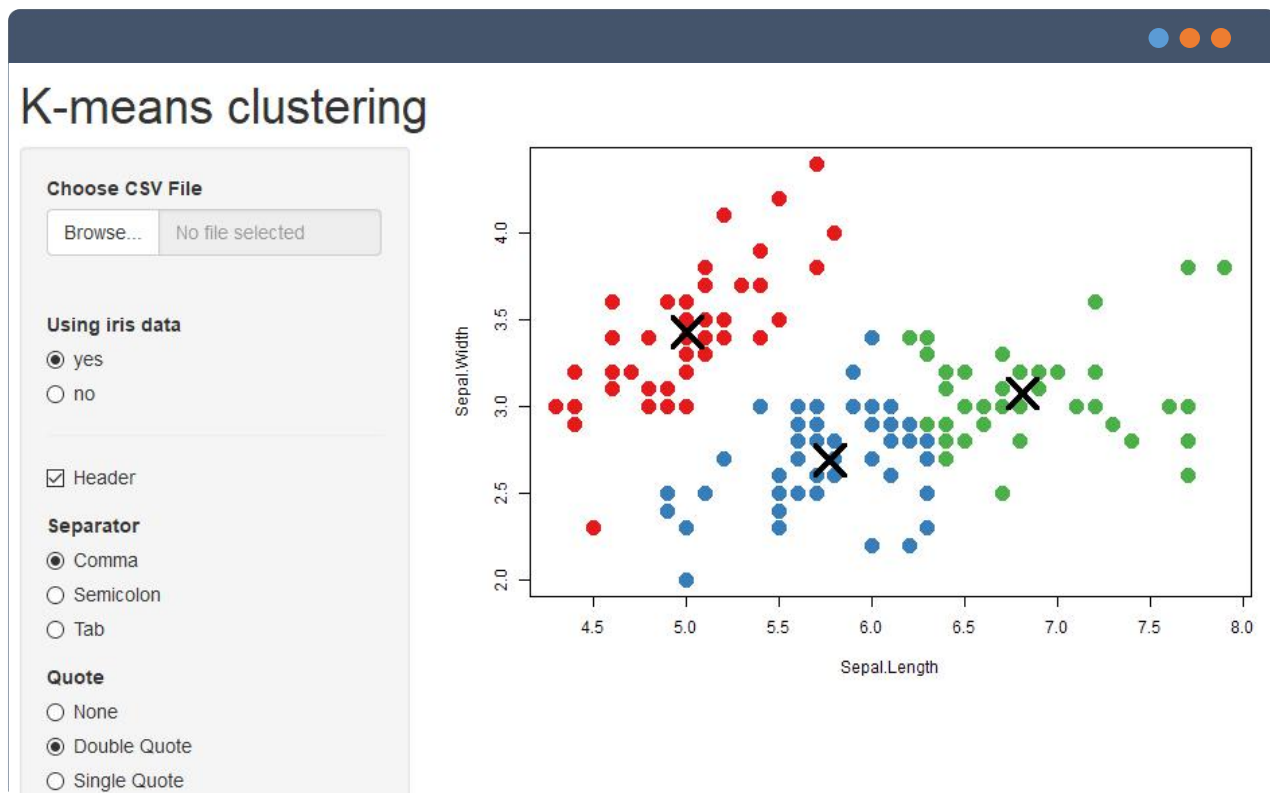
A dark-themed dialog box titled "Deploy" with a close button (X) in the top right corner. It contains fields for "Repository:" (Shiny-HA-Demo), "Branch:" (master), and "Environment:" (a dropdown menu). Below these is a "Hooks" section with two checked checkboxes: "Check and auto-deploy updates" and "Auto-resolve conflicts". To the right of the first checkbox is a timer set to "1 min". At the bottom of the hooks section is a link "How to enable zero-downtime deployment". At the bottom right of the dialog are two buttons: "Cancel" and "Deploy" (which has a green upward arrow icon).

# PaaS下的Shiny容器集群

使用Nginx将用户计算请求分配到多个相同的Shiny容器中，实现计算的负载均衡  
->实现多线程、高可用



# 基于PaaS平台的高可用Shiny



➤ 部署一个高可用的Shiny应用平台

代码:

<https://www.rcoding.net/CloudRaft/Shiny-HA-Demo>

效果:

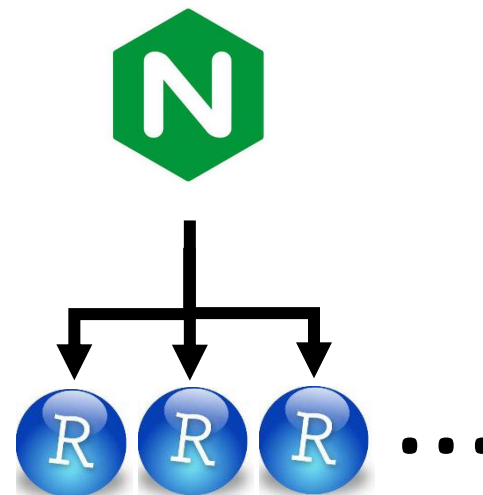




# 基于PaaS平台的高可用shiny

## 部署方式

- ✓ 使用**Git仓库**托管ShinyApp的代码（例如：[www.rcoding.net](http://www.rcoding.net)）
- ✓ 使用容器云部署Shiny集群（rocker/shiny）：
- ✓ 前端Nginx做负载均衡
- ✓ 后端横向扩展Shiny-Server的Docker容器（stateful）
- ✓ 给Docker设置启动脚本，从Git仓库拉取最新的代码
- ✓ 部署完成



过度复杂的代码  
不适用Shiny的风格

1

项目  
缺陷

R语言  
的缺陷

2 缺乏内存管理

3 数据管理



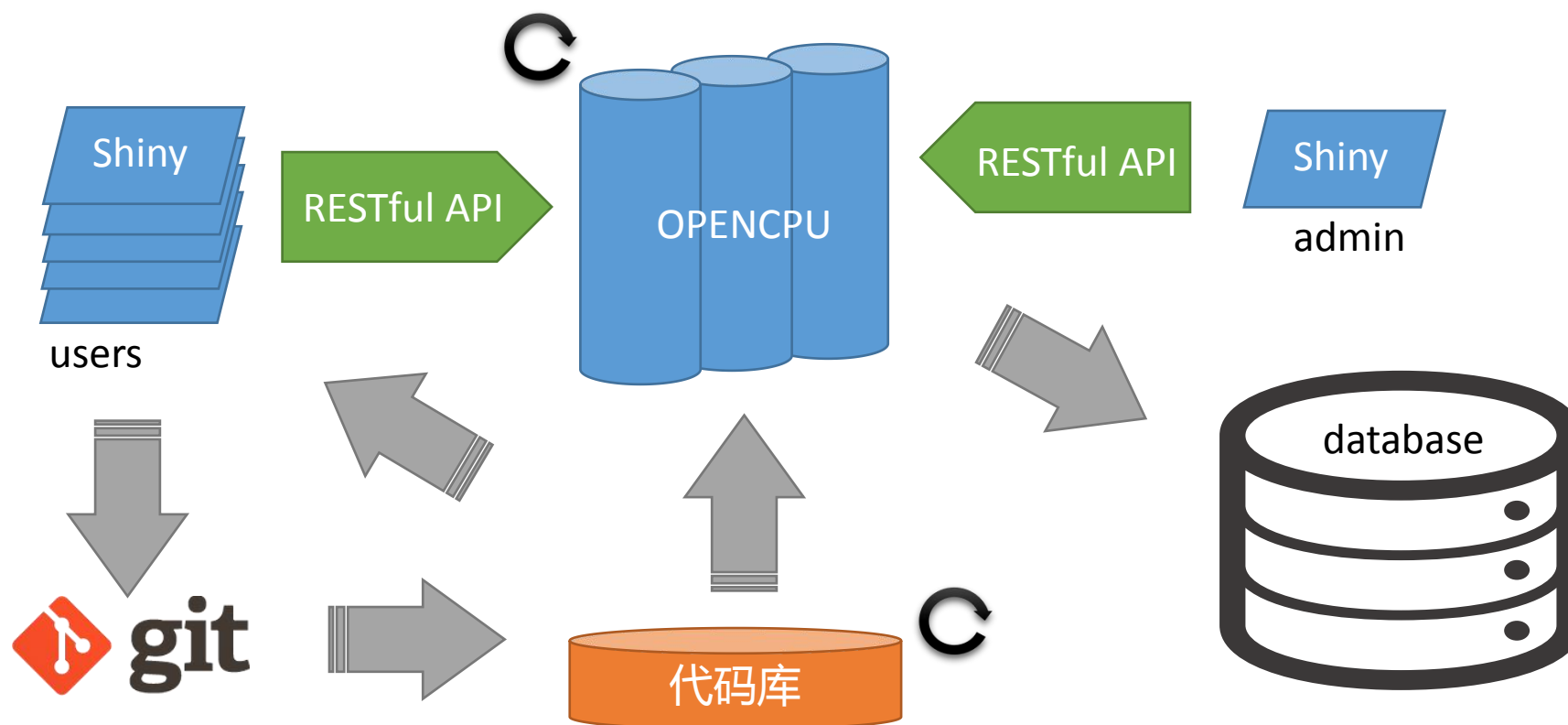
## 解决方案：业务与R框架分离

- 使用opencpu Docker来控制资源用量
- 代码以R包形式挂载到opencpu
- 将需要更新保存的数据以数据库形式保留



# 业务与R框架分离（实验中）

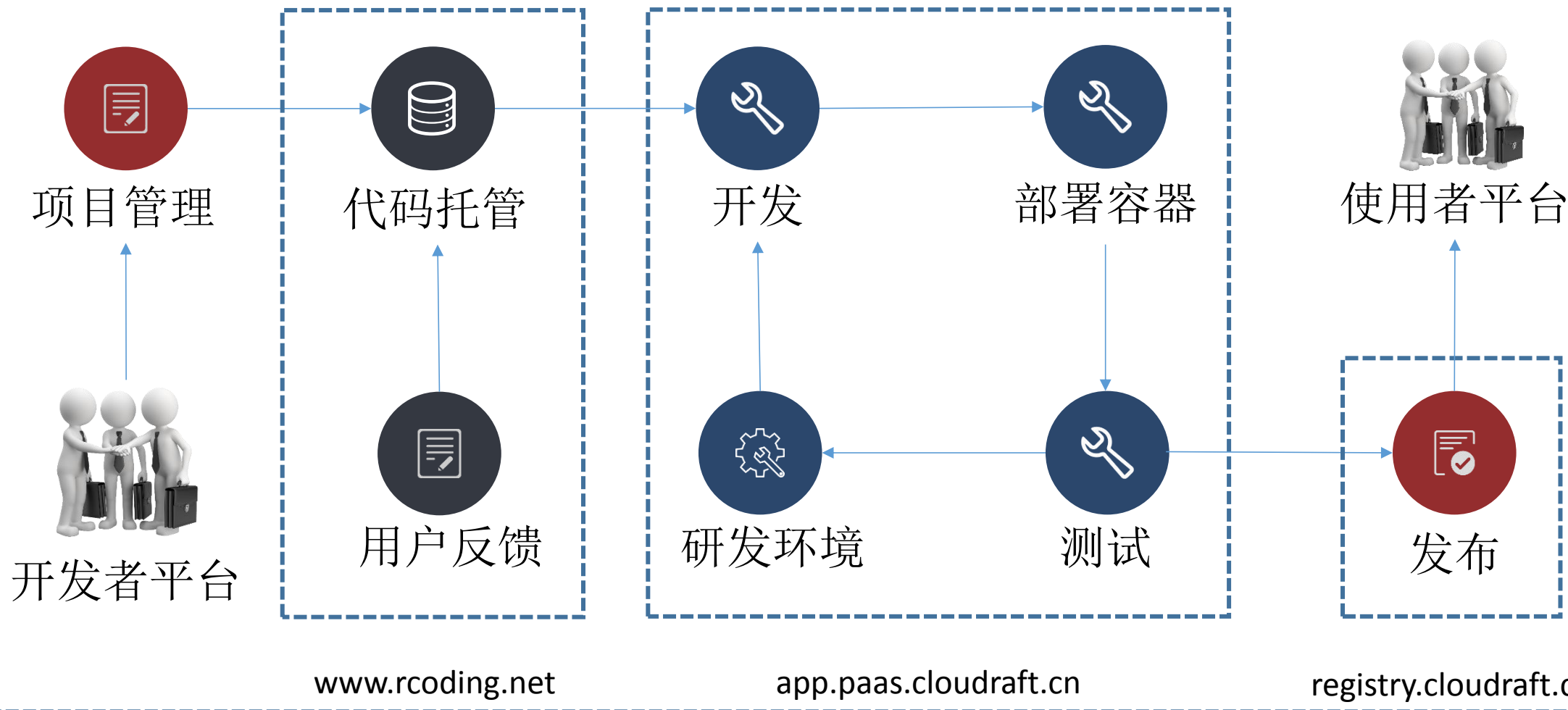
前后端分离



业务分离

Devops collaboration

# Shiny协作开发devops





**THANK YOU FOR WATCHING**

.....