需求文档

目录

[2 引言 2](#_Toc13670798)

[2.1 目的 2](#_Toc13670799)

[2.2 文档约定 2](#_Toc13670800)

[2.3 预期的读者和阅读意见 2](#_Toc13670801)

[2.4 产品的范围 2](#_Toc13670802)

[2.5 参考文档 2](#_Toc13670803)

[3 综合描述 3](#_Toc13670804)

[3.1 产品的前景 3](#_Toc13670805)

[3.2 产品的功能 3](#_Toc13670806)

[3.3 用户类和特征 4](#_Toc13670807)

[3.4 运行环境 4](#_Toc13670808)

[3.5 设计和实现上的限制 4](#_Toc13670809)

[3.6 假设和依赖 4](#_Toc13670810)

[4 外部接口需求 4](#_Toc13670811)

[4.1 用户界面 4](#_Toc13670812)

[4.2 硬件接口 7](#_Toc13670813)

[4.3 软件接口 7](#_Toc13670814)

[4.4 通信接口 12](#_Toc13670815)

[5 系统特性 12](#_Toc13670816)

[5.1 功能需求 12](#_Toc13670817)

[6 其它非功能需求 12](#_Toc13670818)

[6.1 性能需求 12](#_Toc13670819)

[6.2 安全设施需求 13](#_Toc13670820)

[6.3 安全性需求 13](#_Toc13670821)

# 引言

## 目的

目前很多的容器云平台通过Docker等技术提供应用运行平台，从而实现运维自动化，快速部署应用、弹性伸缩和动态调整应用环境资源，提高研发运营效率。

## 文档约定

（1）页面的左边距为2.5cm,右边距为2.0cm，装订线靠左，行距为最小值20磅。

（2）标题最多分三级，分别为标题宋体四号、加粗。

（3）正文字体为宋体五号，无特殊情况下，字体颜色均采用黑色。

（4）出现序号的段落不采用自动编写功能而采用人工编号 ，各级别的序号依次为（1）、1）、a)等，特殊情况另作规定。

## 预期的读者和阅读意见

本文适用于想要了解使用Paas平台的人员，供其快速了解掌握本产品。

## 产品的范围

本产品适合个人或企业，为其提供软件研发平台。

## 参考文档

官网资料，简书等各种博客论文等。

# 综合描述

## 产品的前景

PaaS平台即服务是将基础设施平台作为一种服务呈现给用户的商业模式，是一种比较低成本的方案，对那些资金有限，并且IT资源有限，急需扩展IT基础支撑的企业有着巨大的吸引力。随着云计算领域的发展，目前PaaS服务是云计算技术和应用创新最活跃的领域，与IaaS服务相比，PaaS服务对应用开发者来说将形成更强的业务粘性，因此PaaS服务的重点并不在于直接的经济效益，而更着重于构建和形成紧密的产业生态。

云计算相对来说仍然还不成熟，新服务和功能会一直出现。云开发平台和PaaS也会在新的功能登场时不断演变。然而，由于用户社区对开源软件的影响很深，因此无法保证新功能会以足够快的速度出现以满足你的开发需求。

因此，我们设计了基于docker的paas平台的软件，供用户使用。

## 产品的功能

1).开发平台

2).运行时环境

3).支持运行开发的中间件

4).支持多种服务接入功能

5).支持统一监控管理

6).统一用户、权限分配等

## 用户类和特征

学生类，教师类等。

## 运行环境

有编译环境的系统。

## 设计和实现上的限制

1）巨型应用可能导致问题

2）数据限制可能带来的潜在矛盾

3）通常没有文件系统支持

4）无法处理超高性能的应用场景

5）定制运行时可能会产生问题

## 假设和依赖

1).上传下载的文件只能是压缩文件

2).定制运行时会有一定的问题

# 外部接口需求

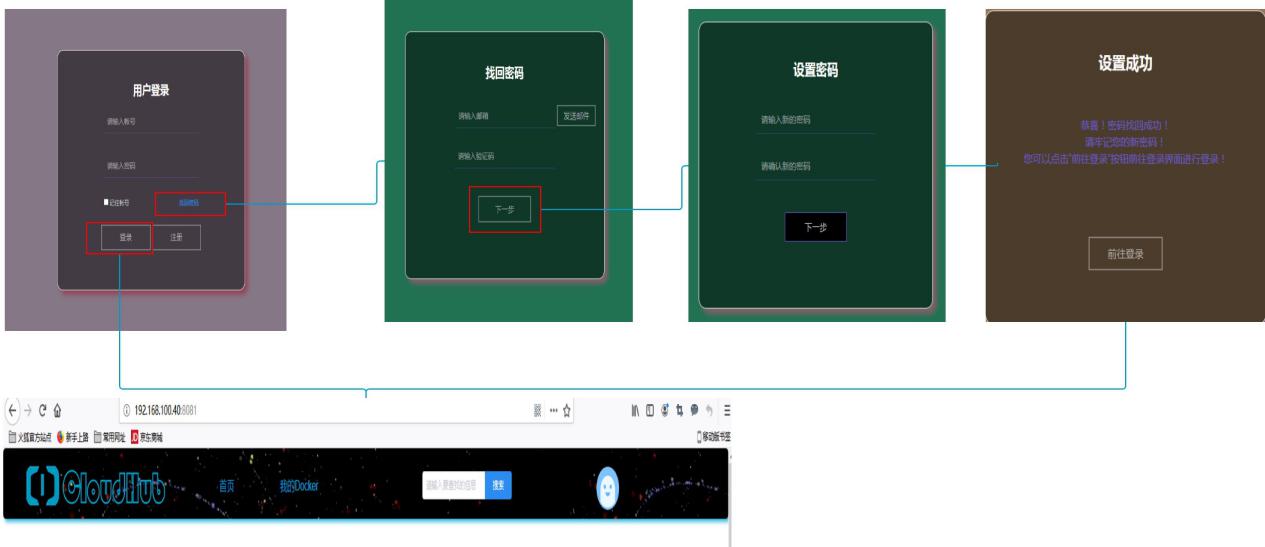
## 用户界面

1).首页

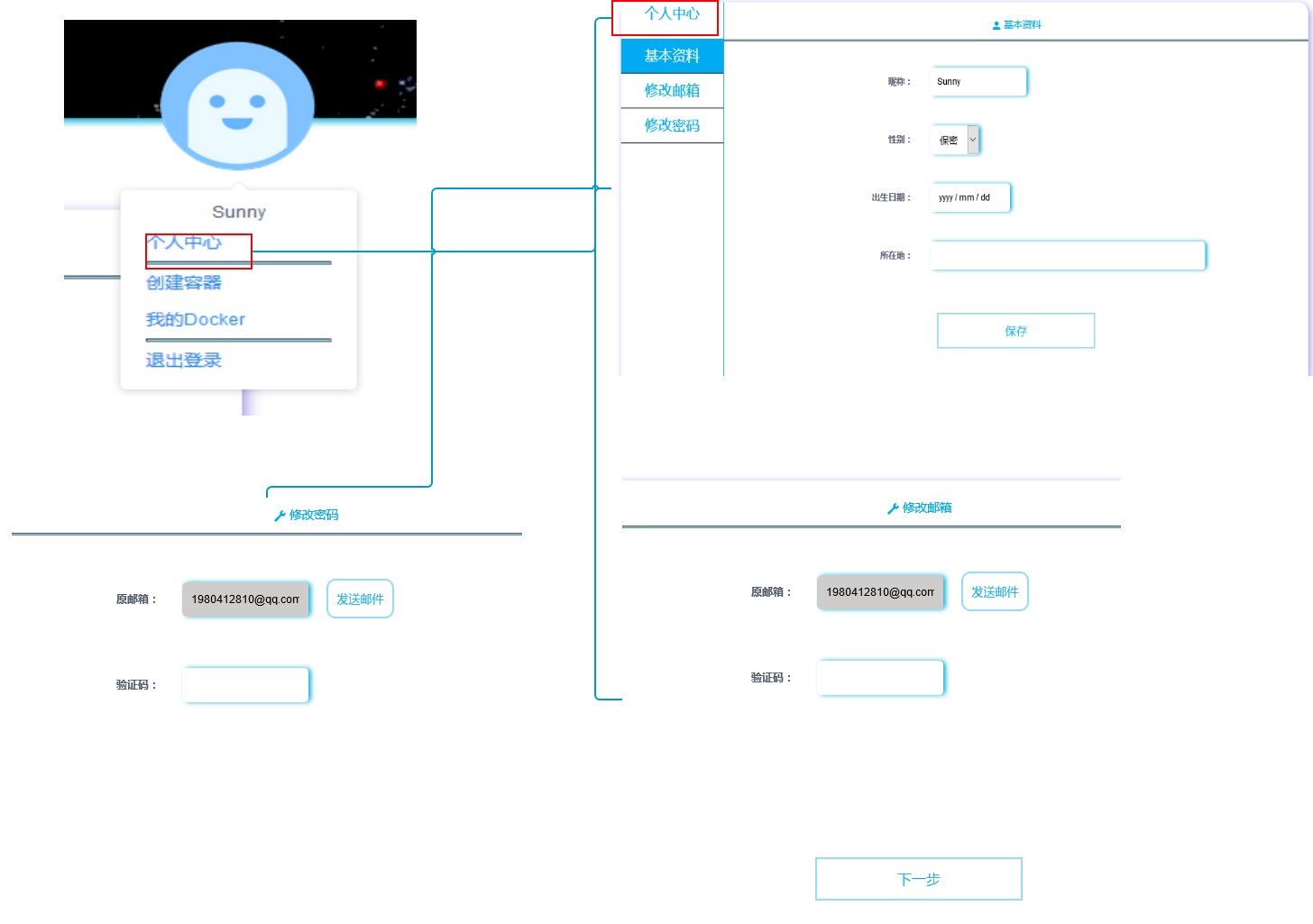
2).注册：用户点击注册按钮即可进行注册，获取id用户身份。在注册界面中，每次刷新，可变换不同的界面颜色，并且输入邮箱，获取验证码方能注册成功。注册成功之后进入主页面。



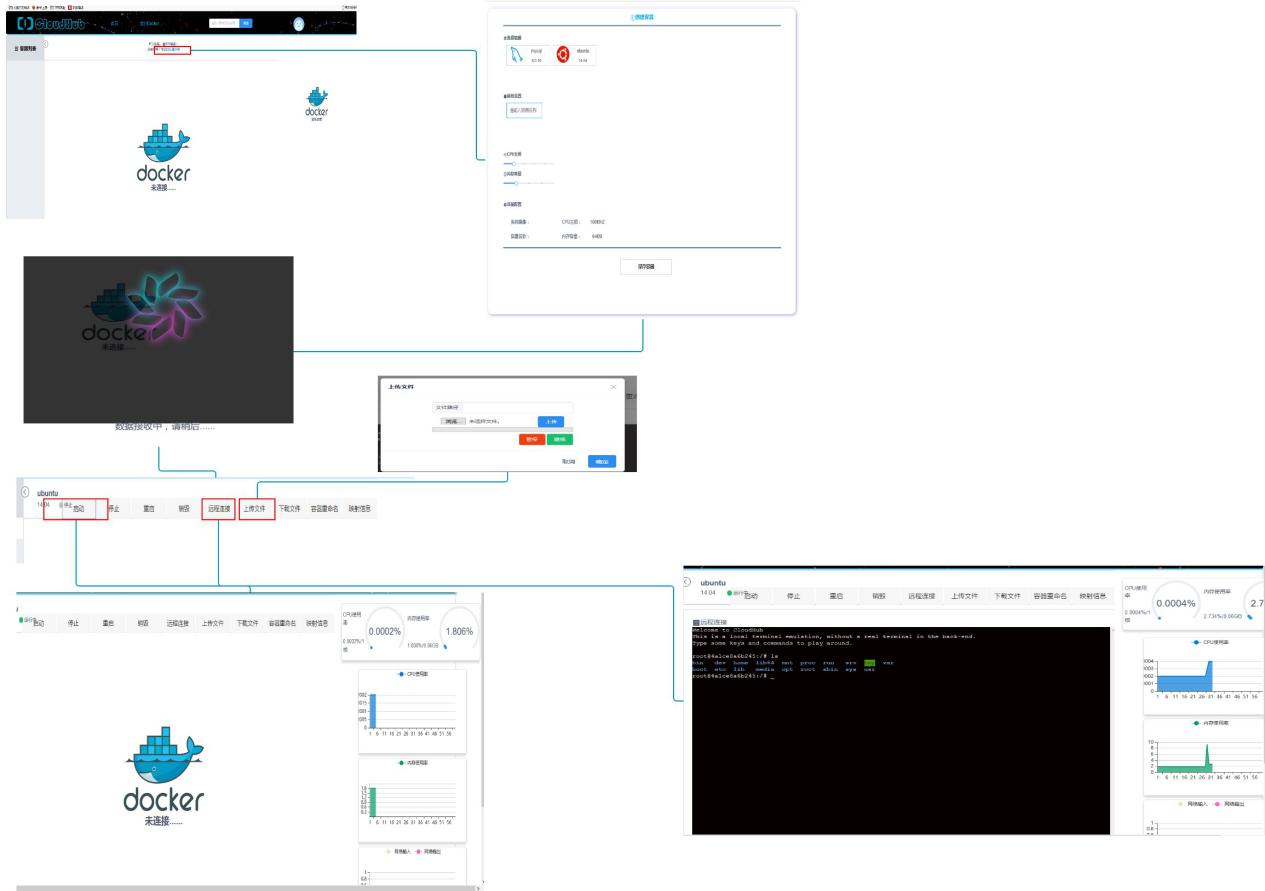
3).登录:在此界面，输入账号密码即可登录。同时，若忘记密码，可点击忘记密码按钮进行重新设置密码。输入正确的账号密码，即可进入登录界面。



4).个人信息：点击个人中心，可查看基本资料，同时修改邮箱，密码等。



5).我的docker：点击我的docker即进入docker操作界面。在此界面，可以对容器进行操作，包括创建，启动，远程控制，上传下载等操作。下图显示了启动，远程控制，上传界面信息。



## 硬件接口

显示器使用VGA或HDMI或DVI接口

键盘/鼠标使用ps2或usb接口

## 软件接口

中间件：

http处理器接口：

//登陆

func (this \*Handler) Login\_handler(w http.ResponseWriter, r \*http.Request)

//注册

func (this \*Handler) Register\_handler(w http.ResponseWriter, r \*http.Request)

//密码找回

func (this \*Handler) Retrieve\_handler(w http.ResponseWriter, r \*http.Request)

//主页

func (this \*Handler) Index\_handler(w http.ResponseWriter, r \*http.Request)

//个人信息

func (this \*Handler) Account\_handler(w http.ResponseWriter, r \*http.Request)

//创建容器

func (this \*Handler) Createdocker\_handler(w http.ResponseWriter, r \*http.Request)

//我的docker

func (this \*Handler) Mydocker\_handler(w http.ResponseWriter, r \*http.Request)

//docker容器资源（websocket）

func (this \*Handler) Resource\_handler(w http.ResponseWriter, r \*http.Request)

//docker容器终端（websocket）

func (this \*Handler) Shell\_handler(w http.ResponseWriter, r \*http.Request)

//docker容器文件上传（websocket）

func (this \*Handler) Upload\_handler(w http.ResponseWriter, r \*http.Request)

//docker容器文件下载

func (this \*Handler) Download\_handler(w http.ResponseWriter, r \*http.Request)

//刷新已加载的模板

func (this \*Handler) Refresh\_handler(w http.ResponseWriter, r \*http.Request)

rpc管理器接口：

//连接rpc服务器

func (this \*Rpc\_manger) conn\_rpc()

//统一rpc调用

func (this \*Rpc\_manger) Rpc\_call(method string, args interface{}, reply interface{}) error

//初始化rpc管理器

func (this \*Rpc\_manger) Init(host string)

//关闭rpc管理器

func (this \*Rpc\_manger) Close()

验证码管理器接口：

//发送验证码 参数:电子邮箱

func (this \*Captcha\_manger) Send\_captcha(email string) error

//获取以及产生的验证码 参数：电子邮箱

func (this \*Captcha\_manger) Get\_captcha(email string) string

//删除验证码 参数：电子邮箱

func (this \*Captcha\_manger) Delete\_captcha(email string)

//计时器 删除超时的验证码

func (this \*Captcha\_manger) timer\_captcha()

//验证码管理器初始化

func (this \*Captcha\_manger) Init(user string, pwd string, domain string, timeout int)

//验证码管理器关闭

func (this \*Captcha\_manger) Close()

docker管理器接口：

//连接docker服务器

func (this \*Docker\_manger) conn\_docker()

// 列出镜像

func (this \*Docker\_manger) Listimage(image \*[]string) error

//下载镜像

func (this \*Docker\_manger) Pullimage(image string) error

//创建docker容器

func (this \*Docker\_manger) Createcontainer(image string, ports string, cmd string) (string, error)

//限制 docker资源

func (this \*Docker\_manger) Containerupdate(containerID string, frequency int64, memory int64) error

//docker容器启动

func (this \*Docker\_manger) Startcontainer(containerID string) error

//docker容器重启

func (this \*Docker\_manger) Restartcontainer(containerID string) error

//docker容器关闭

func (this \*Docker\_manger) Stopcontainer(containerID string) error

// docker容器删除

func (this \*Docker\_manger) Removecontainer(containerID string) error

//返回docker容器host

func (this \*Docker\_manger) Returncontainerinfo(containerID string) ([]string, error)

//docker容器资源使用率

func (this \*Docker\_manger) Containerres(containerID string) ([]byte, error)

//连接docker容器实现远程终端功能

func (this \*Docker\_manger) Containershell(conn\_web \*websocket.Conn, settings []string)

//docker容器文件上传

func (this \*Docker\_manger) Containerfile\_upload(conn\_web \*websocket.Conn, settings []string)

//docker容器文件下载

func (this \*Docker\_manger) Containerfile\_download(w http.ResponseWriter, r \*http.Request, settings []string)

//初始化docker管理器

func (this \*Docker\_manger) Init(host string, cpu\_f int64)

//关闭docker管理器

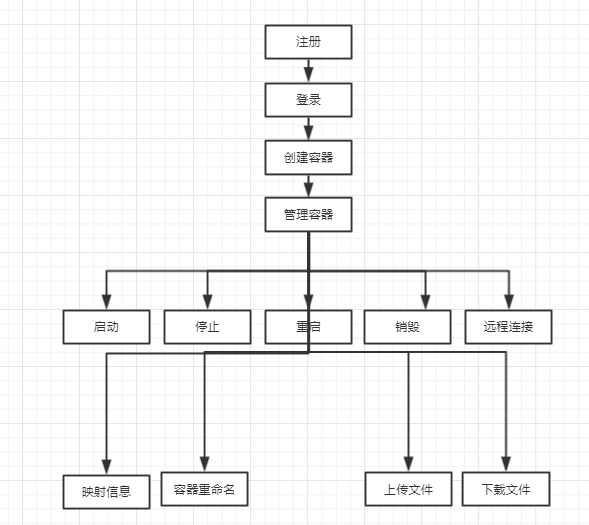
func (this \*Docker\_manger) Close()

## 通信接口

web浏览器

# 系统特性

## 功能需求



# 其它非功能需求

## 性能需求

1. 响应时间：在95％的情况下，一般时段响应时间不超过1.5秒，高峰时段不超过4秒；

2. 系统容量：支持3000用户同时在线

3. 资源使用率：CPU占用率<=50%。内存占用率<=50%。

## 安全设施需求

1. 对输入有提示，数据有检查，防止数据异常；

2. 系统健壮性强，应该能处理系统运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、硬件设备失败等，系统应该能正确的处理，恰当的回避；

3. 支持自动重连机制

## 安全性需求

1. 保证管理者和注册用户的密码安全，数据库访问控制。

2. 提供运行日志管理及安全审计功能，可追踪系统的历史使用情况。

3. 严格权限访问控制，用户在经过身份认证后，只能访问其权限范围内的数据，只能进行其权限范围内的操作

5.4 软件质量标准属性

1. 在95%的故障中，系统最多需要20秒重启；

2. 提供数据备份和恢复功能，使得在由于系统的错误或其他原因引起系统的数据丢失或系统的数据被破坏时，能够及时恢复和还原数据。