

2016-11-28

上海全端网络科技有限公司

www.newtouch.com

新致云NEWTOUCHoNE技术白皮书

v1.0

目录

[产品简介 4](#_Toc467073152)

[功能架构 4](#_Toc467073153)

[功能架构图 4](#_Toc467073154)

[主要模块 5](#_Toc467073155)

[管理平台 5](#_Toc467073156)

[用户登录与注册 5](#_Toc467073157)

[团队管理 5](#_Toc467073158)

[创建应用 5](#_Toc467073159)

[参数配置 6](#_Toc467073160)

[代码模板 6](#_Toc467073161)

[日志搜索 6](#_Toc467073162)

[应用管理 7](#_Toc467073163)

[代码构建 7](#_Toc467073164)

[应用部署 7](#_Toc467073165)

[动态扩缩 7](#_Toc467073166)

[日志查看 8](#_Toc467073167)

[应用监控 8](#_Toc467073168)

[环境迁移 8](#_Toc467073169)

[负载均衡 9](#_Toc467073170)

[灰度发布 9](#_Toc467073171)

[自动启停 9](#_Toc467073172)

[日志搜索 10](#_Toc467073173)

[应用集成 10](#_Toc467073174)

[SDK集成 10](#_Toc467073175)

[PaaS服务关联 10](#_Toc467073176)

[服务管理 11](#_Toc467073177)

[微服务架构支持 11](#_Toc467073178)

[服务注册 11](#_Toc467073179)

[服务发现 11](#_Toc467073180)

[服务代理 12](#_Toc467073181)

[服务监控 12](#_Toc467073182)

[安全与监控 12](#_Toc467073183)

[租户隔离 12](#_Toc467073184)

[权限控制 12](#_Toc467073185)

[资源管理 13](#_Toc467073186)

[运行环境 13](#_Toc467073187)

[资源调度 13](#_Toc467073188)

[PaaS服务管理 13](#_Toc467073189)

[RDS数据库 13](#_Toc467073190)

[Redis缓存 15](#_Toc467073191)

[对象存储 16](#_Toc467073192)

[企业视图 17](#_Toc467073193)

[统一监控 18](#_Toc467073194)

[应用管理 18](#_Toc467073195)

[Docker管理 18](#_Toc467073196)

[节点监控 18](#_Toc467073197)

[日志监控 18](#_Toc467073198)

[异常警告 18](#_Toc467073199)

[日志查询 19](#_Toc467073200)

[产品特点 19](#_Toc467073201)

[价值分析 19](#_Toc467073202)

# 产品简介

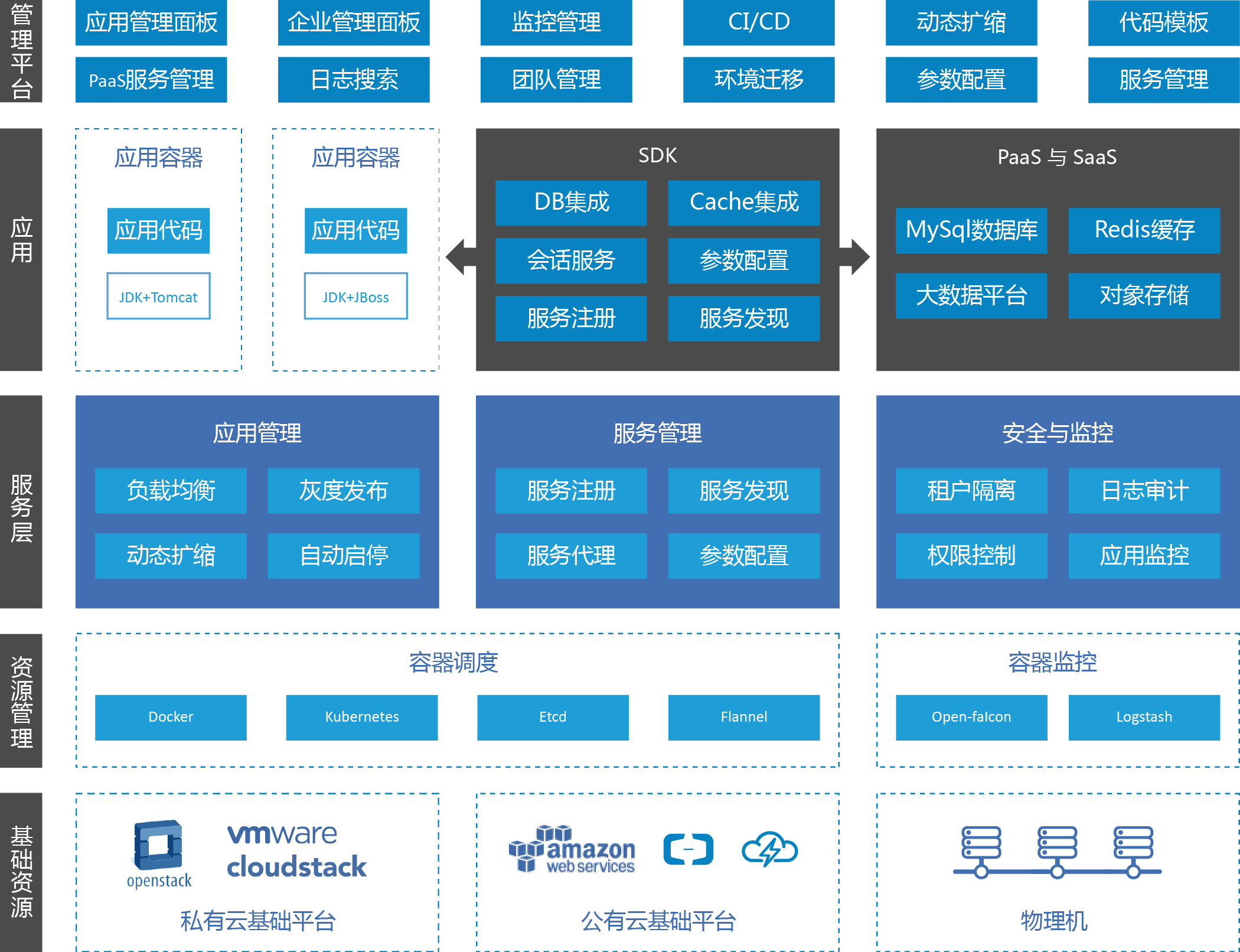
“NewtouchOne”是基于Docker、Kubernetes等容器技术，提供应用全生命周期服务的私有云服务平台。

我们致力于为企业级用户打造持续交付和自动运维平台，解决用户从技术选型、环境准备、服务依赖、自动编译打包，到线上部署、运行、监控、维护的全套需求。

“NewtouchOne”支持私有云、混合云、公有云部署模式，提供用户私有集群的容器化管理和资源智能化分配、项目组间隔离，提供全流程标准化的团队管理、应用管理、应用持续集成、镜像构建、部署管理、微服务架构支持、容器运维和多层级监控服务。

# 功能架构

## 功能架构图



* 平台可以部署在任意的公有云、私有云、物理机上，只需基于Linux 系统即可。
* 资源统一调度，采用Google开源的Kubernetes容器调度技术实现。
* 采用小米Open-falcon监控平台，监控每个容器的资源使用情况
* 采用ELK日志收集与查询系统，统一收集容器日志、应用日志、系统日志。
* 平台提供可视化界面对应用进行管理：应用创建、构建、部署、查看日志、监控等操作。
* 基于Java Spring 的SDK用于连接运行在平台之上的应用与平台所提供的PaaS服务，让应用无缝连接PaaS服务。
* 平台直接提供可配置的参数管理、会话管理等功能，能让应用快速进行环境迁移。
* 为开发人员提供友好的控制台界面，无需复杂的操作以及学习Docker等技术即可使用容器技术来部署应用。
* 提供独立的企业视图，在一个统一的管理界面上可以监控和管理平台上所有的应用，并开放Docker相关底层功能，可以让熟悉相关技术的人进行深度的容器技术使用和运维。

# 主要模块

## 管理平台

### 用户登录与注册

提供用户的自主登录、注册功能。并可以在后台设置每个用户可以创建的应用数量。主要为企业内部开发人员自主申请账号，并同时进行身份鉴别和管理。可以对接企业内部的账户体系。每个用户创建的应用属于该用户独有，对资源进行了多租户隔离。

### 团队管理

团队管理主要应对企业内部的项目管理模式（一个项目有多个开发人员同时维护，以及多个应用），团队可以管理多个用户，属于这个团队的用户可以查看到这个团队下的应用。

### 创建应用

应用是程序代码在平台上运行的逻辑单元，通过应用可以对程序代码进行相应的管理如：构建、部署等。创建应用主要分为两种方式：

1. 从已有的代码仓库关联

对于已经在平台之外的Git仓库上创建好的应用代码，可以通过这种方式将代码部署到平台上。只需要填写Git地址，用户名和密码即可。同时可以选择是否需要创建或关联依赖的PaaS服务（MySQL数据库、Redis缓存等）。

1. 从代码模板创建一个新的应用

对于新创建的应用，可以选择从代码模板创建一个新的应用。代码模板是企业内部预习定制好的一些基础代码框架，包含符合企业内部技术选型、安全规范的代码框架。

应用创建好后，才可以进行后续的所有管理操作，比如：代码构建、部署等。

在微服务架构模式下，一个应用可以包含多个服务，每个服务可以是独立的部署单元。应用与部署的环境无关，仅仅是逻辑上的管理主体。

### 参数配置

所有应用程序运行都离不开参数配置，比如数据库连接地址、外部接口调用地址等。而且这些参数都是与环境有关的。为了实现应用代码在环境之间无缝迁移，平台提供了统一的参数管理功能，将应用所需要的参数放在平台的参数配置系统当中。在应用程序启动的时候通过SDK自动获取与当前应用所运行的环境上的应用参数，注入到spring 参数体系当中。让应用无缝的使用这些参数。应用代码无需任何改动即可获取到这些参数。同时统一的参数配置系统可以实现动态修改参数、版本管理等功能。

### 代码模板

代码模板是一套经过企业内部认可的代码框架，这个框架包含了初始化的代码、技术选型、所依赖的服务。通过标准化代码模板，可以保证通过平台上创建的应用使用的是同一套技术体系，这对于企业内部推广技术标准有着独特的优势，同时平台未来会提供应用所使用的技术框架列表查询，可以让企业管理人员一目了然的掌握每个应用所使用的开发框架，便于后续的安全基线检查、版本更新通知等。

代码模板可以是一套应用程序的基础框架，也可以是一套完整的应用代码。可以实现相同功能的代码在不同应用之间共享使用，降低开发成本。

### 日志搜索

平台为每个应用提供一套独立的日志搜索界面，可以通过关键字、时间条件等查询应用产生的日志。对于微服务架构的应用可以进行聚合搜索，跨服务搜索。

## 应用管理

### 代码构建

平台直接支持从代码到镜像的构建，使用人员无需编写Dockerfile文件和理解Docker容器相关概念，降低使用难度。只需要代码本身是Java语言且是标准的Maven工程结构即可（未来会增加对其他语言的支持）。支持的类型如下：

1. SpringBoot类型的Java应用，打包结构为jar包格式。
2. 普通的JavaEE Web工程，打包结构为war包格式。
3. 支持Maven的多模块结构，但需要指定具体的模块名称。

代码构建与环境无关，且与运行的应用中间件无关（SpingBoot类型除外，需要在代码里面自己指定应用中间件类型和版本，如：tomcat或jetty）。

### 应用部署

应用部署是将代码构建出来的镜像部署到平台上，真正的将应用运行起来。部署时可以选择运行的环境（JDK版本、Tomcat版本、JBoos版本），部署成功后会自动分配一个三级域名，可以通过这个域名直接访问到应用首页。

当应用因为某些原因宕机后,平台会自动重新启动实例，并保证实例个数为设定的最小副本数。

部署成功后，可以在平台上查看应用的逻辑部署图（外网访问地址、内部代理转发层、实例应用之间的逻辑架构图）。

### 动态扩缩

应用的动态扩缩实现的前提是应用本身是无状态的，可以增加和缩减应用实例而不影响程序运行逻辑。

平台原生提供了对应用的动态扩缩能力，而且操作非常简单，只需设定阈值即可自动完成，无需手工介入。

通过动态扩缩设置，可以指定应用最小运行的实例个数，并设置相应的阈值来实现应用实例的动态扩缩，平台会自动监控每个实例是否达到阈值，并自动增加或缩减应用实例。整个过程无需人工参与。

### 日志查看

查看应用运行的日志记录，是排除应用运行异常的重要手段。运行在平台之上的应用只需将日志内容输入到console即可(使用log4j或者logback设置A为STOUT)，平台会自动收集日志和提供查询工具。

当前平台提供两种模式查看日志:

1. 通过控制台提供的日志查看功能，可以实时查看最近500条日志，并且滚动更新，这种操作模式和传统的操作模式一致。
2. 通过日志查询平台（平台包含了一套统一的ELK日志查询系统），根据关键字或者时间维度综合查询。

未来会增加对微服务架构模式下的聚合日志查询。可以通过日志索引到微服务之间的调用关系。

### 应用监控

每个应用在运行的时候，平台会自动在容器底层添加监听Agent，作为容器本身资源使用情况的搜集器，并实时抓取(延迟约30S)数据，推送到统一监控平台。使用人员可以在控制台上查看到各自应用的资源使用情况(CPU、内存、网络、磁盘使用率)。

在企业视图可以监控所有应用的资源使用情况，也可以查看宿主机（虚拟机或物理机）的资源使用情况，从而判断是否需要增加宿主机。

未来会增加应用接口级别的监控，包括接口调用耗时、次数、异常统计、接口健康度等指标。

### 环境迁移

在企业内部通常存在多套环境，比如测试环境、UAT环境、准生产环境、生产环境等。通常一个应用在发布一个版本后都需要在各种环境下都测试一遍，因此需要在每个环境下都部署同一个版本的应用，这个部署过程称之为环境迁移。同理，这里的环境迁移可以通指同一个应用在不同的环境之间部署，不仅仅是企业内部的不同环境，也可以适用于公有云、私有云、混合云之间的迁移。

在容器模式下，做环境迁移会非常简单，只需要将打包好的镜像在不同的环境上部署一遍即可。无需针对这个应用进行额外的配置，真正做到开箱即用。

在平台上预先将应用所依赖的参数配置和服务创建好后，只需要将镜像在新的环境下重新部署一遍即可。

当迁移到公有云环境时，如果需要采用公有云的PaaS服务（如使用阿里云提供的RDS,Redis服务），平台会自动进行路由切换，应用程序无需修改任何代码即可完成公有云的环境部署。便于在开发环境使用企业内部PaaS服务，生产环境使用公有云服务之间转换。

### 负载均衡

每个部署在平台上的应用，默认就可以获得负载均衡的能力，而且无需做任何配置即可享用，所有访问应用接口都是通过一个虚拟的Service服务代理来做负载均衡。当应用的副本数大于1个时，平台会自动将访问该应用的请求采用轮询策略分发到不同的实例上，保证请求压力均衡分配。当某个应用实例销毁或新增时，负载均衡器会自动感知到该实例的状态，会自动在轮询策略上添加或删除该实例。

### 灰度发布

对于线上已经在运行的应用，为了避免全部停机影响正在使用的用户，同时也为了验证新版本的系统是否符合上线条件，通常都会采用AB测试方式来实现。为了实现这个功能，在发布策略上通常采取灰度发布的方式。先让应用实例当中的某几个更新为最新版本，在测试正常后，再逐渐将其他应用实例更新为最新的版本。

在平台上可以通过发布一个新的应用实例来实现该功能，先构建一个新版本的镜像，然后基于这个镜像发布一个新的应用实例，并将这个应用实例的标签添加与已经运行的应用实例一样的标签内容，默认平台上的请求会自动将后续新的请求转发到这个新的应用实例上，从而可以让新旧版本共存，而无需整体停机再启动。 后续只需要根据调整新旧版本的副本数（每减少一个旧版本副本数，就新增一个新版本副本数）即可滚动替换旧版本应用。

### 自动启停

在整个平台里运行的应用都是使用同一个资源池。为了更加高效的使用资源池，可以将一些使用率不高的应用做自动启停策略，从而及时释放资源。可以在平台上设定应用在多长时间内无任何请求访问时，自动关闭应用。当有新的请求访问这个应用时，会自动将应用启动，从而继续满足业务使用。

该功能会造成应用停机再启动时，在启动的过程当中服务不可用。但对于那些只是临时使用一段时间比如（演示、测试某个特性的应用）来说可以使用该功能做到资源的及时释放。

### 日志搜索

所有应用在运行后产生的日志，平台会自动收集到一个统一个日志管理中心。

统一的日志管理中心提供管理、搜索、收集、分析功能。所有运行在平台之上的应用都可以直接使用这个平台提供的服务能力。后续会对接大数据平台提供更精确的日志统计、分析、实时数据抓取等API给应用使用，便于做更多的业务级别的日志统计分析操作。

除了在控制台界面可以查看某个应用本身运行日志外，还可以通过统一个日志搜索平台获取历史的日志信息，包括应用日志、代理请求日志、系统运行日志。以及可以使用日志搜索平台所提供的日志查询、分析、统计、报警等功能。

## 应用集成

### SDK集成

运行在平台之上的应用为了获取平台所提供的服务能力，需要嵌入一个SDK，这个SDK的作用仅仅是作为应用与平台所提供的服务之间的连接纽带。

SDK所提供的功能与平台所提供的服务一一对应。

1. 参数配置，应用在嵌入SDK后，SDK会自动获取平台上配置的参数，并注入spring property体系当中。应用代码使用参数只需要根据参数名称自动注入即可，无需将参数写入在应用代码里面的properties文件当中。所有参数自动获取并且在平台上修改后，会自动更新spring bean里面注入该参数的值（需要配置策略是更新属性还是重新构建spring bean，或者重新加载spring context）。
2. PaaS服务对接。平台所提供的PaaS服务，会以参数的方式或API的方式提供到应用，比如在平台上创建的MySQL数据库，会直接注入数据库连接地址、用户名、密码等参数给应用。对于应用使用方来说，只需要引用参数名称即可。如果使用spring boot框架，完全不需要做任何配置即可直接使用数据库功能。

具体的SDK使用方法，请参见SDK说明文档。

### PaaS服务关联

平台本身以及提供了很多常见的PaaS服务，如：MySQL数据库、Redis缓存、MangoDB、hadoop、文件对象存储等服务。这些服务都是通过SDK的方式与应用进行关联，同时控制台上提供了每种服务的操作页面，可以进行部分服务的参数配置、监控等操作，具体参见PaaS服务管理章节。

## 服务管理

### 微服务架构支持

在微服务架构体系当中，同一个应用（这里指逻辑上的一个应用，不是具体某个运行的程序）会存在多个服务，每个服务都会独立部署、有各自的副本数、服务与服务之间存在调用依赖。同时需要对所有服务除了单独的监控之外，还需要从整体上进行监控，甚至包括全局的异常排查、性能指标统计等操作。

对比传统的单体应用架构，在微服务架构体系下，必须针对微服务架构做很多管理上的功能，而这些功能几乎所有微服务架构都会面临使用。因此，我们的平台将这些支撑微服务架构的管理功能直接集成到平台之上。让运行在这个平台之上的微服务应用轻松使用这些管理功能。

包含的主要功能如下：

* 服务注册
* 服务发现
* 服务路由
* 服务断路
* 服务监控

### 服务注册

平台自带服务注册中心，所有应用只需在需要暴露的服务接口上添加注解即可自动实现服务的注册。

同时添加相关的文档注解，平台会自动生成一份文档以及对应的测试界面供其他调用通过界面操作的方式测试使用。

### 服务发现

对于服务调用方，需要在代码层面获取到服务提供方的接口地址、端口等信息。如果将这些信息写在参数里面的话，当服务提供方地址发生变化后，需要通知所有调用方进行修改，这在大规模使用微服务情况下几乎无法操作。为此，通常的做法是采用服务自动发现功能，调用方无需直接知道服务放的实际地址和端口，只需要知道一个别名即可。在具体的接口调用时由平台来做其中的转换，保证服务方地址变化后调用方依然可以连通。

应用在需要调用的服务接口上使用平台提供的SDK客户端即可实现该功能，在调用的地方注入服务客户端，同时添加注解声明服务名称（该名称为平台上服务提供方的名称）、版本号、协议即可。

### 服务代理

通常在企业内部都会存在很多历史遗留应用或者某些无法使用容器部署的应用，对于这些应用有可能还需要以服务的方式提供给平台上的应用使用，为了能够继续利用这些应用，可以在平台上通过设置服务代理的方式将平台之外的应用集成到平台之中，便于其他调用方继续使用平台的服务发现功能调用这些服务。

### 服务监控

当微服务应用部署运行后，不再是简单的对某个微服务做监控，而是需要从整体上来监控，比如一次请求会调用哪些微服务，每个服务的健康指标，都应该聚合在一块监控。

平台通过统一的日志收集、集中的服务注册中心、服务发现，可以将微服务之间的接口调用健康指标都获取到，最终以图表和数据的方式展示给用户。

## 安全与监控

### 租户隔离

平台提供一套租户隔离机制，便于应对企业内部不同部门、不同项目组之间的权限隔离要求。

通过租户隔离，可以实现团队与团队、用户与用户之间的应用不可以相互查看和操作(可以通过API进行程序之间的调用)。且在底层资源使用层面也是使用kubernetes的租户隔离机制。

应用依赖的服务（如：MySQL数据库），可以在同一个用户或团队之间的应用共享。

### 权限控制

权限控制分为应用角色控制和功能使用控制。

角色分为个人与团队，个人可以创建个人的应用。其他用户不可以操作个人创建的应用。团队可以将个人加入到团队当中，当创建应用的时候选择属于团队时，这个应用可以被这个团队下的所有人操作。

功能控制上分为应用开发人员和企业管理人员。

应用开发人员使用的管理界面是平台提供的控制台，通过控制台可以进行应用的创建、查看、代码构建、部署和简单的运维操作（监控和日志查询等）。基本的原则是不需要应用开发人员了解Docker、Kubernetes概念和相关的知识就可以直接使用这些系统带来的特性。尽量做到上手简单、符合平时的使用习惯。

企业管理人员使用的管理界面是平台提供的企业视图，通过企业视图可以进行深度的运维与监控操作。可以在企业视图里面进行Docker、Kubernetes原生的API和命令操作，便于更加深度的运维操作。

## 资源管理

### 运行环境

平台底层的运行环境基于Linux系统，对操作系统版本要求为：centos7+、RHL7.0+其他发行版本参见Docker和Kubernetes最新版本安装要求即可。平台本身不依赖具体的运行环境，可以是物理机也可以是虚拟机，或者公有云提供的虚拟机都可以进行部署。

整个平台之上的所有应用，底层运行环境使用的是同一套共享资源，不对某一个应用进行独立的资源绑定。当底层的物理机或虚拟机发生故障时，系统会自动进行漂移操作。保证应用与底层运行资源状态无关。

### 资源调度

平台之上的应用使用底层资源时，采用的策略是基于资源池的概念，统一调度的方式，保证资源的有效利用。当前平台的资源调度采用Kubernetes方案实现。

## PaaS服务管理

### RDS数据库

RDS是一种稳定可靠、可弹性伸缩的在线数据库服务。基于Docker和全SSD盘高性能存储，支持MySQL，默认部署主备架构且提供了容灾、备份、恢复、监控、迁移等方面的全套解决方案，彻底解决数据库运维的烦恼.

**多层安全防护体系，通过十项安全合规认证，为您抵御90%以上的网络攻击**

* 安全事前防护：RDS可设置允许连接实例的IP白名单，严格控制访问源；同时支持通过VPC来获取更高程度的网络访问控制。
* 安全事中防护：RDS访问链路支持SSL加密，杜绝中间人攻击行为；支持开启透明数据加密（TDE），数据落盘自动加密，远离拖库风险。
* 安全事后审计：RDS支持SQL审计功能，记录所有访问源和访问行为信息。对所有安全及故障事件做到有据可查。

**重高可用（容灾）架构，提供99.95%的业务可用性保障**

* **主备架构：**RDS实例采用主备架构，两个实例位于不同服务器，自动同步数据。主实例不可用时，系统会自动将数据库连接切换至备实例。
* **同城容灾：**在不同可用区部署主备实例，独立的电力、网络环境可提升数据可靠性。
* **异地容灾：**RDS for MySQL支持创建异地灾备实例，通过数据传输实现异地数据实时同步，在突发状况下，用户可将异地灾备实例切换为主实例，保障业务可用性。

**弹性扩展，实现100%资源利用率**

* 可按需升级实例的内存、磁盘空间大小，提升业务处理能力；单实例最大支持10000IOPS，1TB存储空间;
* RDS for MySQL 支持挂载只读实例，扩展读取能力；每个只读实例拥有独立的链接地址，可由应用端控制压力分配。

**内网外网同时连接，方便本地化管理**

* 在高安全访问模式下，RDS实例支持内网和外网同时连接。外网地址直接连接，方便本地管理实例；内网地址可与同一地域内的实例等产品连接，缩短响应时间。

**自动备份，两年内数据恢复，解决90%以上的系统故障**

* 每天自动备份数据并上传至[对象存储OSS](https://www.aliyun.com/product/oss/)，提高数据容灾能力的同时有效降低磁盘空间占用。
* 通过备份文件及日志文件，可恢复最长两年内的数据。有效防范因误操作等原因对业务数据造成不可逆的影响。

**自动监控预警，定期性能巡检，为您分担60%以上的运维工作**

* 提供CPU利用率、IOPS、连接数、磁盘空间等实例信息实时监控及报警，随时随地了解实例动态

是一种稳定可靠、性能卓越、可弹性伸缩的数据库服务。基于飞天分布式系统和全SSD盘高性能存储，支持主备版和集群版两套高可用架构。提供了全套的容灾切换、故障迁移、在线扩容、性能优化的数据库解决方案。

### Redis缓存

Redis是一种稳定可靠、性能卓越、可弹性伸缩的数据库服务。基于Docker和全SSD盘高性能存储，支持主备版和集群版两套高可用架构。提供了全套的容灾切换、故障迁移、在线扩容、性能优化的数据库解决方案。

**双机热备高可用架构，提供极高的业务可用性保障**

云数据库Redis服务采用主备双机的高可用架构，容灾切换及故障迁移完全透明化。两个数据节点位于不同不同服务器上，自动同步数据。主节点对外提供服务，当主节点出现故障，系统自动切换备用节点接管服务，整个切换过程对用户全部透明。

**提供数据持久化功能**

云数据库Redis基于Docker系统和全SSD盘高性能存储，默认开启数据持久化功能，提供高速数据读写能力的同时提供了极高的数据可靠性。

集群版分布式架构大幅提升Redis性能

Redis提供集群版实例，轻松突破Redis自身单线程瓶颈。 集群版提供实例的分片及读取算法，整体过程对用户透明，免去用户开发及运维Redis集群的烦恼。云数据库Redis集群版本支持绝大多数Redis命令，可极大满足对于Redis大容量或高性能的业务需求。

**高性能存储设备保障**

云数据库Redis直接构建在物理机上，底层基于Docker和全SSD盘高性能存储，提供了极高的数据读写能力。

在数据量可控及对Redis协议兼容性要求较高的应用，可选择云数据库Redis主备版实例。

**多层安全防护体系，为您抵御90%以上的网络攻击**

在网络入口实时监测，当发现超大流量攻击时，对源IP进行清洗，清洗无效情况下可以直接拉进黑洞。

VPC私有网络：私有安全网络保证数据安全，用户可以基于VPC构建自己的私有网络，从访问源进行直接的风险控制。

强密码验证： Redis默认开启了密码验证功能，只有通过密码认证过的连接方可进行数据操作，极大的保证了数据安全性。

**完善的性能监控，为您分担基础运维工作**

提供CPU利用率、IOPS、连接数等实例信息实时监控及报警，随时随地了解实例动态。

**专业的数据库管理平台，实现一键式管理**

提供专业的管理控制台，实例运维操作可便捷的进行一键式操作。

提供专业的DMS数据管理平台，提供可视化的Redis数据管理，图形化轻松管理数据对象。全面提升研发、运维效率。

### 对象存储

**对象存储，是提供海量、安全和高可靠的云存储服务。RESTful API的平台无关性，容量和处理能力的弹性扩展。**

**对象存储功能：**

**方便、灵活的使用方式 ，让您像操作本地文件一样**

* 提供标准的RESTful API接口，兼容S3接口，丰富的SDK包，客户端工具、控制台，您可以像使用文件一样，方便上传/下载、检索、管理用于Web网站或者移动应用海量的数据；
* 支持流式写入和读出；
* 支付数据生命周期管理。

**强大、灵活的安全机制**

* 灵活的鉴权，授权机制。提交URL鉴权和授权机制，及白名单、防盗链，主子账号功能；
* 提供跨数据中心复制功能，实现数据异地容灾；

**强大的数据处理能力，协助你对存储的文件进行加工处理。**

* 图片处理：支持jpg, png, bmp, gif, tiff等多种图片格式的文件格式转换、缩略图、剪裁、水印、缩放等多种操作；
* 音视频转码：基于OSS存储，为您提供高质量，高速并行音视频转码能力，让您的音视频文件轻松应对各种终端设备；
* 内容加速分发：OSS作为源站，搭配CDN进行加速分发，稳定、无回源带宽限制、性价比高，一键配置。

## 企业视图

企业视图是给企业内部运维团队深度管理平台上的应用所提供的一套管理界面，可以通过这个管理界面实现对平台上所有应用的监控和管理，相当于提供一套权限最大的管理功能。在这之上可以进行统一监控、应用的管理、以及深度的运维操作。对于熟悉Docker和Kubernetes的运维人员来说，提供了一套操作界面可以直接对Docker和Kubernetes进行管理，便于深度运维。

### 统一监控

提供全视角的统一监控界面，包括底层系统资源的使用情况、应用的资源使用情况、容器的资源使用情况。

未来会集成应用监控，可以从应用层面监控应用的API调用情况、数据访问等应用层面的性能指标数据。

### 应用管理

同样在企业视图提供一套对应用的管理操作界面，进行应用的代码构建、部署等操作。

### Docker管理

提供Docker的常用命令操作，可以直接查看宿主机上运行的Docker容器。包括对Kubernetes的操作、docker镜像上传等操作。

### 节点监控

提供对运行应用程序所在的宿主机进行资源使用情况的监控。

### 日志监控

提供一套全局的日志监控功能，可以监控应用上的日志错误率、日志产生数量等指标。

### 异常警告

基于日志监控的结果已经资源监控的结果，提供警报功能，可以将警报内容以邮件或短信的形式发放到负责人。

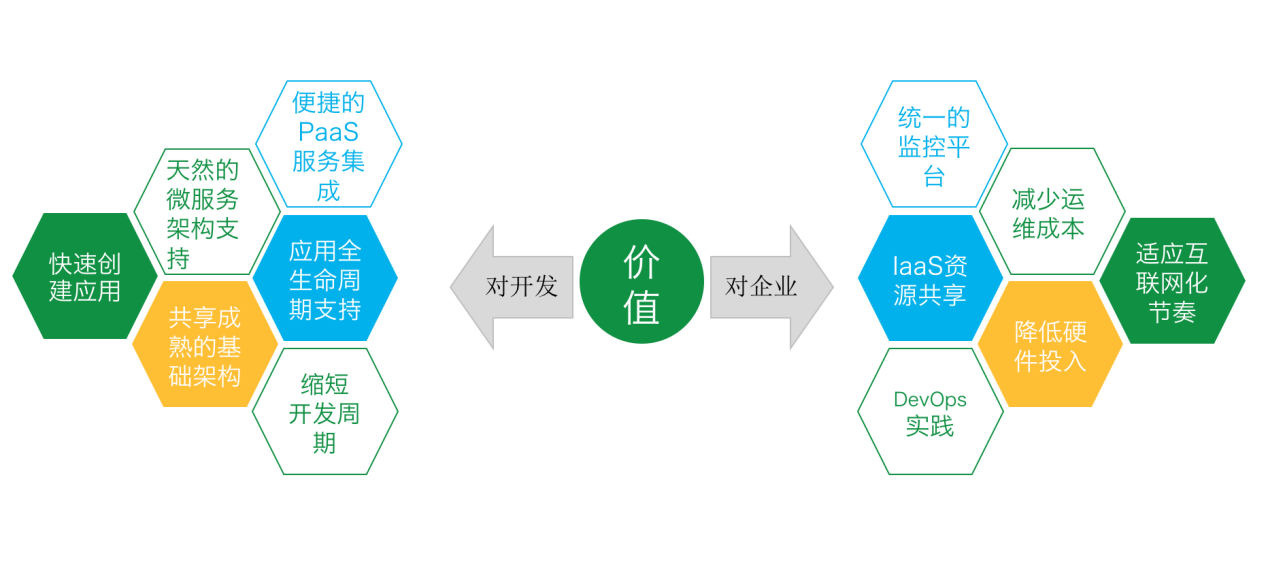
### 日志查询

提供一套全局的日志查询功能，可以查询所有应用的日志。根据时间、关键字、应用名称等维度进行查询。

# 产品特点



# 价值分析



对开发人员：

1. 通过平台可以快速搭建一个可以使用的且满足企业内部规范的应用框架代码。仅需要几个步骤即可完成，对人员能力无限制。
2. 天然的微服务架构支持，无需复杂基础架构设计，直接使用平台提供的微服务架构支持。
3. 平台直接集成常用的PaaS服务，无论是开发环境还是测试环境都可以快速的使用，减少繁杂的安装配置以及程序对接工作。
4. 应用的全生命周期支持，从代码创建到构建、部署都可以在平台上完成，不在需要手工搭建各种CI/CD工具。

对企业管理人员：

1. 提供统一的监控平台，所有应用的运行状态一目了然。
2. 底层资源共享，不在担心资源无法有效利用。
3. DevOps实践变得更加可能。
4. 适应互联网化下的快速IT更新节奏，让IT不再是业务发展的瓶颈。