云托管 CloudBase Run

参考指南

产品文档

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

参考指南

基础

概述

最近更新时间：2020-12-23 14:59:51

云托管（Tencent CloudBase Run）是 云开发（Tencent CloudBase，TCB）提供的新一代云原生应用引擎（App Engine 2.0），支持托管**任意容器化应用**。

主要特性

任意语言、任意容器镜像

您可以使用任何语言、任何框架编写应用，甚至直接使用公共的容器镜像。

原生打通云上资源

您的应用可以直接访问 CloudBase 的其它云上资源，如云函数、云数据库等；也可以访问腾讯云的其它云上资源，如 CynosDB、MySQL 等。

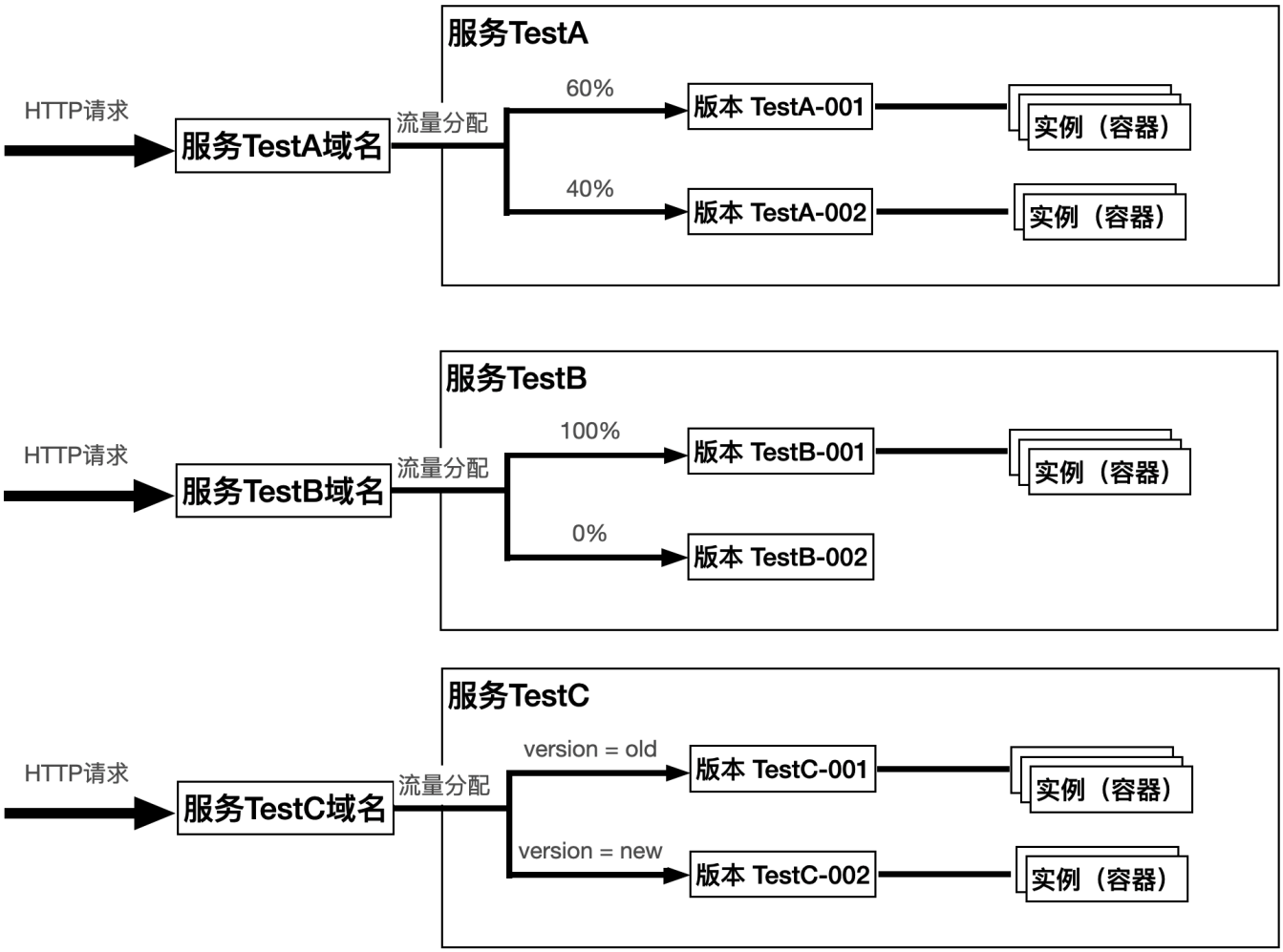
按量计费

您的应用只在被访问时计费。

零运维，弹性伸缩

CloudBase 将会根据应用的使用情况，自动伸缩计算资源，无需人工管理。

资源模型说明



上图展示了三个示例服务 TestA、TestB、TestC。每个服务都包含了两个版本，每个版本对应着一组实例（容器）资源。

对外域名

每个服务都包含一个系统自动生成的默认域名（您也可额外绑定自定义域名），可被用于访问和传入请求。请求方无需感知该服务具体有几个版本，而是将服务视为一个整体。

流量分配

当有请求传入服务时，根据您事先配置的流量分配模式（按百分比/按 URL 参数）和规则，系统会将本次请求路由到对应的版本。

弹性伸缩

不同版本承载的请求数量决定它对应的实例（容器）数量，当无任何请求时实例（容器）数量甚至可以被缩减到0，不产生任何资源消耗。这一动态扩缩过程由系统自动完成，最终您只需为实际使用的所有实例（容器）资源付费。

在您的实际使用中，每个服务最多可以有 10 个版本，每个版本的实例（容器）数量范围为 0 - 50。

新建服务

最近更新时间：2021-03-01 11:33:32

操作流程

第 1 步：新建服务

在 CloudBase 云托管控制台 的服务列表页面，选择对应的环境，单击【 新建服务 】或【 新建此私有网络下的服务 】。

第 2 步：配置服务信息

填写新建服务所需的所有信息。字段详解请参见 服务配置说明。

- 若您在上一步选择的是【 新建服务 】，且这是您在本环境创建的首个服务，则“云托管网络”字段会默认选中“系统推荐”。
- 若您在上一步选择的是【 新建服务 】，但这不是您在本环境创建的首个服务，则“云托管网络”字段会默认选中“已有私有网络“，VPC 默认选中上一个服务所在 VPC，子网默认选中上一个服务所在子网。
- 若您在上一步选择的是【 新建此私有网络下的服务 】，则“云托管网络”字段会默认选中“已有私有网络“，VPC 默认选中您所选 VPC，子网需要手动选择。

第 3 步：提交

单击【 提交 】，如果部署成功，便可以看到如下弹框：

单击【 新建版本 】可立刻开始新建版本并部署，详情请参见 部署服务。

使用限制

- **当前每个环境支持服务总数**：每个环境支持同时存在最多 5 个服务。不再使用的服务可以删除，不计入服务总数。
- **可用 VPC 总数**： 同一帐号最多可用到 5 个 VPC。若存量服务已用到 5 个不同 VPC，您再次新建服务并选择第 6 个未被云托管使用过的 VPC 时，会报错配额不足。

部署服务

最近更新时间：2021-03-01 11:36:19

概念说明

- 版本是访问流量真正承载的实体，对应着一组配置信息相同的容器集合；也是一次部署所需信息的集合，包括镜像地址、服务端口、环境变量、自动扩缩容设置等。
- 每一次升级服务，可选择通过新建版本升级，或者在原版编辑配置并重新部署。两种方式的区别和操作指南，详情请参见 升级服务。

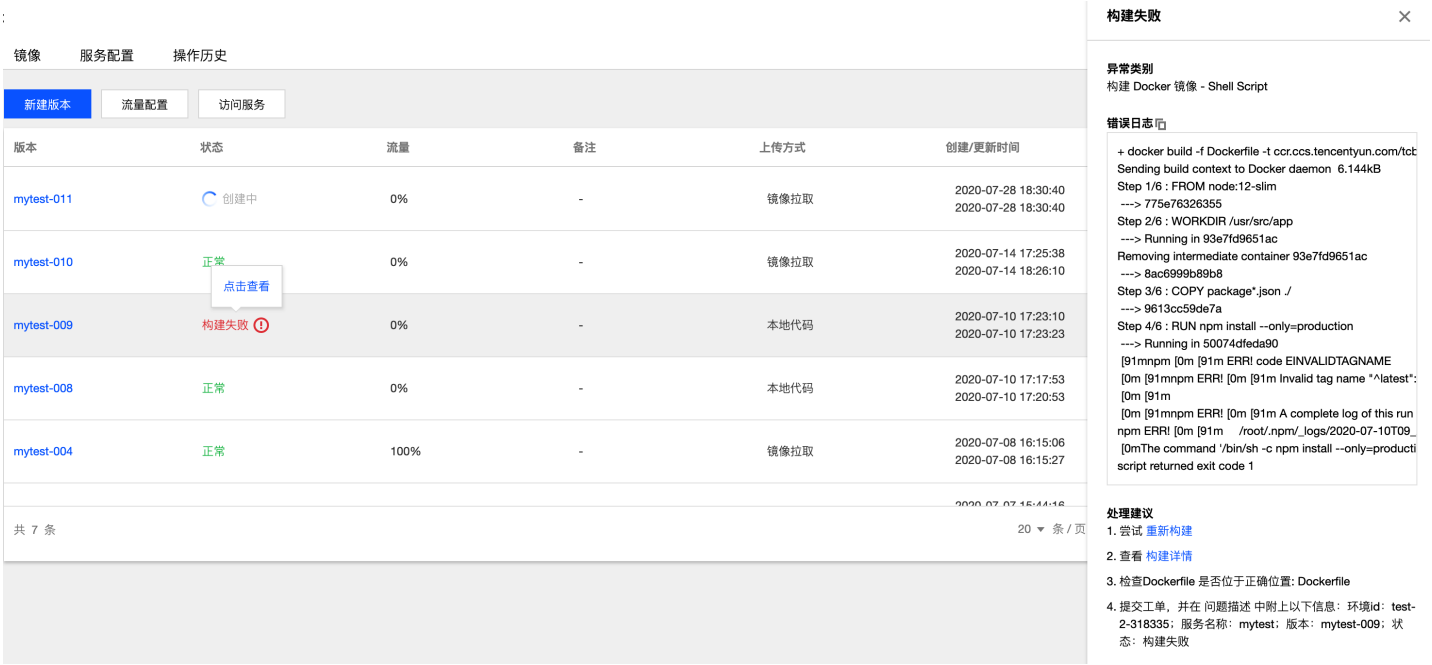
操作流程

登录 CloudBase 云托管控制台，选择您需要部署的服务，单击服务名称进入服务详情页面：

> 说明：
> 如果您还没有任何服务，请先单击【新建服务】，详情参见 新建服务。

单击【新建版本】，在新建版本窗口中，继续填写版本所需配置信息，详情请参见 版本配置说明。

填写完版本配置信息后，单击【开始部署】，部署成功则状态变为“正常”。若有报错，会变为具体的错误状态。



> **说明：**
>
> - 若在新建版本时流量策略选择“部署完成后自动开启 100%流量”，则流量会从 0%变为 100%。
> - 若选择了“部署完成后保持流量为 0 稍后再手动调整流量”，或者选择了“部署完成后自动开启 100%流量”但还想再次调整流量，可以进行手动配置，详情请参见 流量配置。

完成了部署和流量配置后，您可能希望能够快速访问自己的服务查看效果。云托管自动为您的服务分配了一个默认域名，您可以直接单击【访问服务】，通过这个默认域名访问您的服务页面。

> **说明：**
>
> - 您访问服务产生的即为真实业务流量。
> - 如您配置流量时选择的模式为“按百分比”，则点击“访问服务”将按您设定的百分比概率随机访问某个版本；
> - 如您配置流量时选择的模式为“按 URL 参数”，则点击“访问服务”产生的是一个无参数的 HTTP 请求，会访问到您设置的默认版本上。您可自行构建含参数的请求进一步验证。

使用限制

- 一个服务可同时存在最多 10 个版本。
- 版本名由系统自动生成，格式为“服务名”+“序号”，序号按照创建顺序依次递增，不支持修改（例如：testservice-001, testservice-002 等）。
- 不再使用的版本可以手动删除，已删除的版本不计入版本总数，但不会影响新建版本的序号。(例如：删除掉版本 "testservice-002"，再次新建的版本仍然会是 "testservice-003"。）

资源消耗说明

无论采用哪种副本模式，**部署过程本身都会产生一定的资源消耗**。

低成本模式

虽然副本个数最小值为 0（无流量时缩容到 0 个实例不产生资源消耗和费用），但在部署过程中会先创建出 1 个实例，待部署成功后再触发缩容到 0。
例如：副本模式为“低成本”，规格为“1 核 1G”，流量策略为“部署完成后自动开启 100%流量”，部署耗时 5 分钟，部署成功后一直无任何业务流量。因为缩容到 0 的观测期为半小时，故会产生 CPU 使用量 35（核 x 分钟）以及内存使用量 35（GiB x 分钟）。

高可用模式

部署开始即会创建出最小个数的实例，产生资源消耗。
例如: 副本模式为“高可用”，规格为“1 核 1G”，最小副本个数为 5，流量策略为“部署完成后保持流量为 0 稍后再手动调整流量”，部署耗时 5 分钟，部署成功后一直无任何业务流量。则会产生 CPU 使用量 25（核 x 分钟）以及内存使用量 25（GiB x 分钟）。

更新或回滚服务

最近更新时间：2021-03-01 11:33:42

云托管目前支持两种更新/回滚方式：

对比项	新建版本	编辑原版本配置并重新部署
更新前后 IP 不变	否，每个版本 IP 不同	是，版本不变则 IP 不变
调用方式	通过域名调用	通过域名调用或通过 IP 调用
回滚	支持，流量导回旧版本即可回滚	支持，用历史配置重新部署实现回滚
灰度发布	支持，详情请参见 流量配置	不支持
多版本并行	支持	不支持

方式一：新建版本（推荐）

每次更新服务都新建一个版本，部署成功后逐步将流量从旧版本切换到新版本。通过域名调用不强依赖固定 IP，更符合云原生理念。

操作步骤

登录 CloudBase 云托管控制台，选择您需要更新的服务，单击服务名称进入服务详情页面，单击【新建版本】：

填写更新所需的版本配置信息，详情请参见 版本配置说明。

> 说明：
> 您可以根据需要选择合适的流量配置策略，例如：
>
> - **灰度发布/蓝绿发布**：流量策略选择【部署完成后保持流量为 0，稍后手动配置流量】，待发布完成后，手动配置流量到新版本上；
> - **全量发布**：流量策略选择【部署完成后自动开启 100%流量】，待发布完成后，即自动切换全部流量到新版本上；
>
> 具体可以参见：流量配置。

随后单击【开始部署】即可。

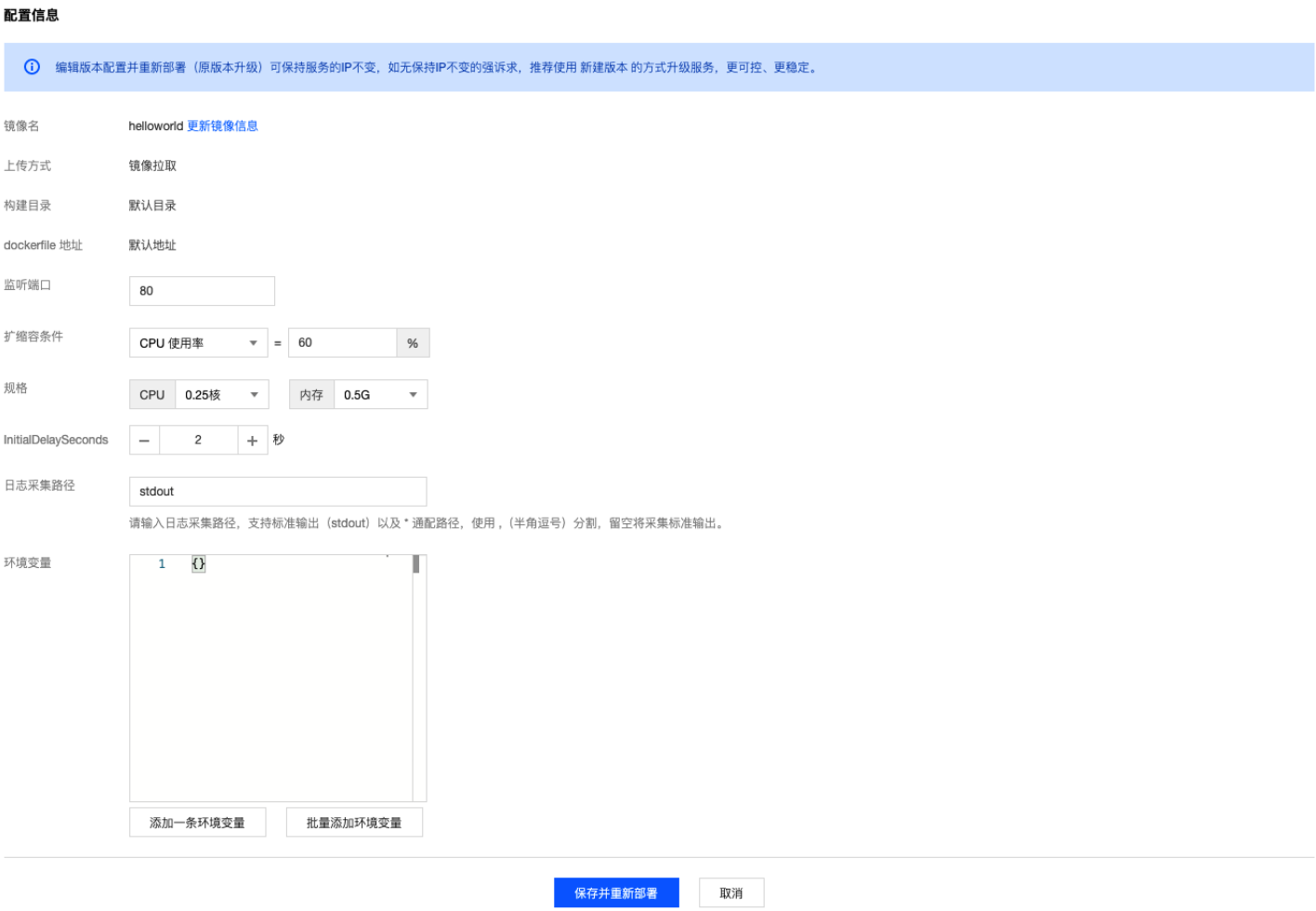
方式二：原版本编辑配置并重新部署

不新建版本，在已有版本中编辑配置信息，然后重新部署。**此方式下不支持灰度**。

操作步骤

登录 CloudBase 云托管控制台，选择您需要更新的服务，单击服务名称进入服务详情页面，选择需要更新的版本，在【操作】>【更多】下拉菜单中，选择【编辑配置并重新部署】：

填写更新所需更新的版本配置信息。



> **说明：**
>
> **– 重新部署不支持重新选择流量策略，将沿用版本当前流量配置。如有需要，请在重新部署成功后手动配置流量。**
>
> - **若需更新镜像/代码，请单击【 更新镜像信息 】，并填写表单。提交新的镜像信息并不会立刻开始部署，需要等其余配置信息也编辑完毕后触发。**

单击【 保存并重新部署 】，版本状态变为“更新中”。部署成功则状态变为“正常”，流量保持不变。若有报错，会变为具体的错误状态。

风险说明

重新部署过程中，为保证业务持续运行，将采取滚动更新实例的方式进行升级。滚动更新可能带来以下风险：

- **重新部署需要所有实例更新成功才算成功，任意实例更新失败会自动回滚整个版本，导致版本更新不成功；**
- **重新部署过程中，版本同时存在已更新的实例和尚未更新的实例，有一定概率不同请求访问同一版本时现象不一致；**
- **重新部署过程中，因正在更新的实例无法提供服务，可能导致剩余实例无法承受当前业务流量而触发扩容条件，产生额外的实例及资源消耗。**

删除服务

最近更新时间：2020-10-22 09:22:58

> **注意：**
> 删除服务为不可恢复操作，操作前，请先评估对您的业务的影响。

操作步骤

1. 登录 CloudBase 云托管控制台，再按需要切换到指定的环境。
2. 进入需要删除的服务详情。
   选择您需要删除的服务，单击服务名称进入服务详情页面。

3. 删除服务下所有版本。
   对每个版本进行单击【删除】>【完成】操作，删除所有版本。

4. 在服务列表页面，选中服务，单击【删除】即可删除服务。

版本配置说明

最近更新时间：2021-03-01 11:34:40

基础配置说明

上传方式

支持本地代码/代码库拉取/镜像拉取三种方式。

方式 1：本地代码

上传代码工程文件夹或 zip 压缩包，系统解析成功后先将代码构建为镜像、将镜像推送到服务绑定的镜像仓库储存，然后基于该镜像进行部署。

此方式需要您自行编写 Dockerfile 并包含在代码文件夹或 zip 压缩包中。

> 说明：
>
> 我们提供 Java、PHP、Node.js、Golang、Python 语言的示例代码包下载用于 Demo。除此之外的任意开发语言也都可支持。
>
> 如需更多语言的 Demo 代码包资源，请 联系我们。

方式 2：代码库拉取

获得您的授权后，从您的 GitHub / GitLab / 码云 Gitee 仓库拉取代码后，先将代码构建为镜像、将镜像推送到服务绑定的镜像仓库储存，然后基于该镜像进行部署。

此方式需要您自行编写 Dockerfile 并包含在代码库中。

> 说明：
>
> 首次使用需要您先进行授权。授权成功后，当前环境任意服务新建版本都不再需要再次授权。

方式 3：镜像拉取

无需系统提供从代码构建镜像的服务，由您自行采用任意方式构建好镜像后，手动将镜像上传至服务所绑定的镜像仓库中，然后系统基于您选定的镜像进行部署。

> 说明：
>
> 1. 如何上传镜像，请查看本文档“相关操作”中的“手动上传镜像”部分。
> 2. 使用 Demo 镜像可快速体验部署过程。选择 Demo 镜像后无需修改任何配置，直接开始部署。

监听端口

应用监听端口。默认为 80。

流量策略

部署完成后，如何开启流量。目前有如下两种选择：

- 部署完成后保持流量为 0 稍后再手动调整流量：希望部署完后先手动验证再开流量，或只希望引入部分流量进行灰度升级。
- 部署完成后自动开启 100%流量：首次部署完成后立刻开流量，或升级时进行全量升级。

备注信息

选填，会展示在版本列表页面中，对部署不产生任何影响。

自动扩缩容配置说明

副本模式

支持低成本/高可用两种模式。

模式 1：低成本

适合对成本敏感，对冷启动相对不敏感的业务。

- 副本个数最小值为 0，当没有流量打到版本上时，版本将缩容到 0，不保留任何实例，不产生任何费用。
- 连续半个小时无流量才将实际缩容到 0（避免流量偶然波动带来的误判）。再次冷启动时，可能有 30 秒服务延迟。
- 默认推荐规格为最小规格 0.25 核 0.5G，**单价最低**。您也可按需调整规格。
- 虽然没有业务流量，部署过程中仍然会先产生一个实例，部署完成后再缩容到 0，因此**部署过程本身会产生一定的资源消耗。**

> 说明：
> 将版本流量百分比设置为 0 并不是触发缩容到 0 的充要条件。低成本模式下，版本流量百分比设置为 0，一定会触发缩容到 0。但如果版本流量百分比不为 0，需要连续半小时的观测期，期间版本没有产生真实业务流量，才会触发缩容到 0。

模式 2：高可用

适合对成本相对不敏感，对服务常驻有诉求，或无法接受冷启动的业务。

- 副本个数最小值不能为 0，可以设置为 1 ~ 50 间的任意整数。即便没有流量打到版本上，仍会保持最小个数的实例数，也会**持续产生费用**。

- 默认推荐规格为 1 核 1G。您也可以按需调整规格。
- 部署过程中即开始按最小副本个数和规格产生实例，**部署过程本身会产生一定的资源消耗。**

规格

指集群中每个容器实例的配置。自动扩容时，新创建的实例将使用这个规格。同一个版本下所有容器实例规格都会保持一致。

副本个数

指当前版本在自动扩缩容时可达到的最大实例数及最小实例数。最小值下限为 0，最大值上限为 50。
如需将最小值修改为 0，请先切换副本模式至“低成本”。
如需将最小值修改为大于 0 的整数，请先切换副本模式至“高可用”。

扩缩容条件

当达到某个条件时，云托管会自动会创建/删除实例，然后检测是否再次达到条件，如果满足条件则继续扩缩容，如此反复直至实例数量达到副本个数的最小值/最大值，或不再满足扩缩容条件时停止自动扩缩容。

目前仅支持 CPU 使用率作为扩缩容条件，更多扩缩容指标敬请期待。

> 说明：
>
> - 自动扩容到副本个数最大值后若仍不足以承载业务流量，即便再次达到扩缩容条件，也不会继续创建新的实例，可能导致您的业务受影响，请您评估好业务指标后合理设定副本个数最大值。
> - 若希望提升最大值限额（大于 50），请提交工单联系我们单独处理。

高级配置说明

高级配置可不做任何修改直接使用默认值开始部署。

InitialDelaySeconds

实例创建完成后，等待一定的时间后开始进行健康检查。若健康检查失败，将重试 3 次，依然失败则判定服务版本异常。请将此值设定为大于应用启动的时间，否则版本可能持续处于异常状态。默认值为 2 秒。

日志采集路径

可设置目录或文件，支持设置多个路径。采集到的日志可以在 日志管理 中查看。支持标准输出（stdout）以及 * 通配路径（例：/logs/*），使用 ,（半角逗号）分割，留空将采集标准输出。

环境变量

用户所需的环境变量，直接传入容器中。以 key value 的形式可配置多个。

流量配置说明

最近更新时间：2020-10-21 11:12:45

CloudBase 云托管支持“**按百分比**”和“**按 URL 参数**”进行流量分配，开发者可以基于此实现**灰度发布**、**蓝绿发布**等功能。

按百分比（随机）分配流量

1. 单击【流量配置】。
2. 模式采用默认【按百分比】。
3. 选中刚部署完的最新版本和当前运行中的旧版本，并为每个版本指定流量百分比。所有百分比加起来需要等于100%或 0%，否则无法完成。未在流量配置窗口中添加的版本，将统一将流量百分比变为 0%。
4. 单击【完成】，系统立刻开始按百分比随机分配流量，将部分流量从旧版本引向新版本。

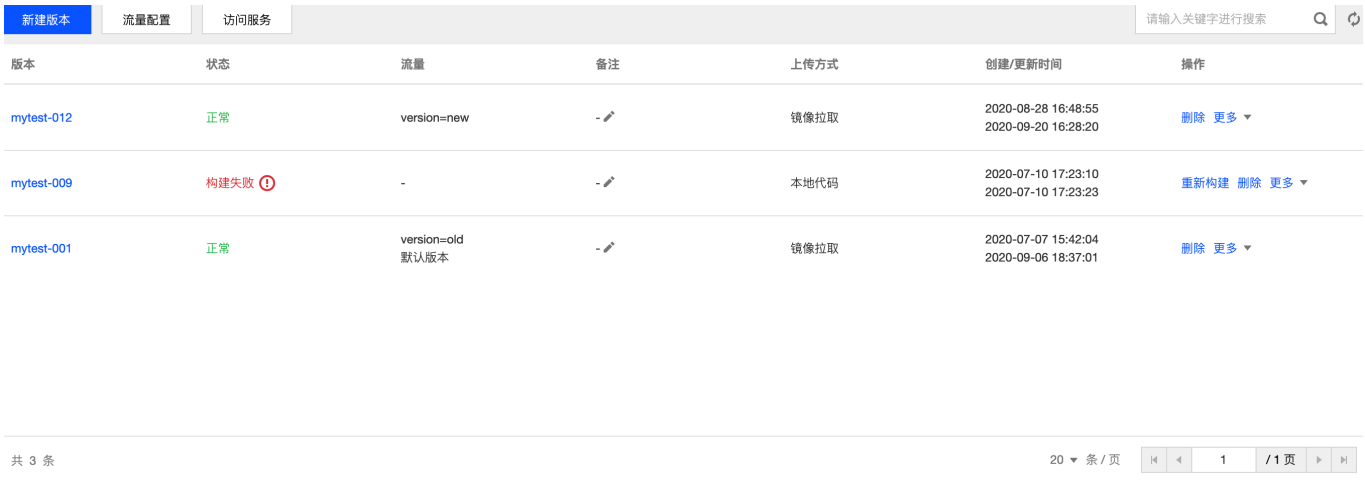
5. 成功后，可以看到对应版本的流量百分比已经变化。

按 URL 参数（定向）分配流量

1. 单击【流量配置】。
2. 模式选择【按 URL 参数】。
3. 添加部署完的最新版本，设定 URL 参数 key 和 value（支持 Glob 表达式）。符合条件的请求将路由到新版本。
4. 再添加当前运行中的旧版本和对应的 URL 参数。
5. 将旧版本设置为默认版本。
6. 在“匹配结果预测”中输入 URL 参数模拟请求，点击【预测】查看对应版本，调试配置。

7. 单击【完成】，系统立刻开始将部分流量从旧版本引向新版本。

viii. 成功后，可以看到对应版本的流量条件已经变化。



服务配置说明

最近更新时间：2021-03-01 11:34:32

服务名称

环境下服务的唯一标识。创建成功后**不支持修改服务名称**。
如果您有多个环境，不同环境之间服务可以重名。（例如：开发环境和测试环境可以同时存在服务"testservice"）。

备注信息

展示在服务列表页面，仅您自己在控制台可见，不对部署产生任何影响。

所在私有网络/云托管网络/子网

背景知识

关于什么是私有网络（VPC）及子网，请参见 私有网络 文档。
点击服务所在私有网络的名称，可以跳转到 私有网络控制台 查看私有网络的详情，以及该私有网络内您还有哪些其他云资源，可以与此私有网络内的云托管服务配合使用。

使用限制

- 处于同个私有网络中的云托管服务，可以**通过内网域名相互调用**，不产生公网流量费用，时延更短。如果您的服务需要访问腾讯云数据库或其他云服务，您可以将这些外部依赖部署于同个私有网络中。
- 不同私有网络之间的服务默认无法互通，除非您选择将多个 VPC 之间打通。这可能产生额外的费用和维护成本。
- **不支持更换服务所在私有网络**，您可以在正确的私有网络下重新创建服务后，再删除当前服务。

用法推荐

- 新建本环境内首个服务，且无特别的网络要求，请选择“系统创建”，云托管会自动为您创建并配置好一个全新的 VPC，同时**自动创建一个全新的微小型 NAT 网关**，保证您新建的服务具备访问公网的能力。
- 新建本环境内的第 2 ~ 第 N 个服务，且无特别的网络要求，推荐选择和前一个服务相同的私有网络。这样既方便服务之间互访，也可以节约私有网络数量配额。
- 如果有明确的网络要求，请选择“已有私有网络”，指定符合您要求的 VPC 及子网。

> 说明：
> 若您选择的 VPC 不是由云托管创建的，需要您自行在此 VPC 中配置了 NAT 网关，否则您的服务将无法访问公网。

NAT 网关

背景知识

关于什么是 NAT 网关及如何配置公网访问，请参见 NAT 网关 文档。

使用限制

- 云托管自动创建的 NAT 网关（含“系统创建”标签），为云托管定制的微小型网关，最大并发数 10 万，最高带宽 10Mbps。如果不满足您的使用需求，请前往 NAT 网关控制台 新增并配置符合您要求的 NAT 网关。
- 非云托管创建的 NAT 网关，在云托管控制台无法展示 IP，费用由 NAT 网关侧进行结算。请前往 NAT 网关控制台 查看并管理。

用法推荐

NAT 网关 IP，将作为您的服务访问公网的出口 IP，也可用作域名备案。但无法通过这个 IP 访问到您的服务。

域名备案

背景知识

关于什么是域名备案及如何进行域名备案，请参见 域名备案 文档。

使用限制

展示当前服务所用的 NAT 网关 IP 是否可用于域名备案。需要满足以下条件状态会展示为“可备案”，不满足则会给出具体原因。

1. 使用“系统创建”的 NAT 网关。
2. 当前帐号下有至少一个有效期大于 90 天的云托管资源包（不限环境）。

镜像仓库

背景知识

云托管使用的是容器镜像服务（TCR）个人版，没有费用。
您可以在服务详情页面中的【镜像】选项卡中，快速查看当前服务所绑定的镜像仓库中所有镜像和它们的使用情况。
更多关于腾讯云镜像仓库的介绍，请参见 容器镜像服务 文档。

使用限制

- 服务在创建时会绑定一个唯一的腾讯云镜像仓库，管理该服务下的所有版本的相关镜像。
- 不支持同一服务绑定多个镜像仓库，使得不同版本的镜像分散在多个不同的镜像仓库里。
- 所有由云托管创建的镜像仓库都会统一存放于以 “tcb” 开头的同一个命名空间之下，并以环境 id_ 服务名的格式命名。您可以在 容器镜像服务控制台 上看到由云托管创建的这个镜像仓库并对它进行管理。

- 暂不支持服务绑定公有仓库。

用法推荐

“使用系统默认仓库（推荐）”：创建服务时，云托管将自动为您创建一个与服务同名的私有仓库并与您的服务进行绑定。
“绑定已有腾讯云镜像仓库”适用以下场景：

1. 跨场景复用
   在云托管的场景之外，您已经使用了腾讯云的镜像仓库，并希望里面的镜像继续用于云托管的某个服务。此时，在创建服务时，可以选择 “绑定已有腾讯云镜像仓库” ，指定任意您账户下的私人镜像仓库。
   请注意，若如此做，则您通过云托管产生的镜像（手动上传或自动构建推送），可能也会影响您在其他场景下的镜像使用。

2. 多个服务共用同一套代码/镜像
   如果同一个服务需要同时部署 N (N>=2) 套（在同一环境部署多套请用不同服务名加以区分，分别部署在不同环境服务可以重名），则在创建第 2 – N 个服务时，可以选择“绑定已有腾讯云镜像仓库”，并指定第一个服务所绑定的镜像仓库，实现镜像在多个服务间的复用。
   两个服务都绑定到了同一个镜像仓库，仓库中所有的镜像可以同时用于两个服务，且这两个服务未来由云托管自动构建的镜像都会被推送到这同一个镜像仓库。

特殊情况

如果您曾经创建了一个服务并选择 “使用系统默认仓库” ，后来又删除了这个服务，但保留了这个同名的镜像仓库。则当您在同一环境再次用同样的名字创建服务时，选择 “使用系统默认仓库” 会自动绑定到原有的这个同名镜像仓库，而不会再次创建一个全新的镜像仓库。

此时，若您愿意继续使用之前同名服务遗留的镜像，可直接开始使用；若您不再需要遗留的镜像，请您到服务详情页面的 “镜像” 选项卡中，或到 容器镜像服务控制台 手动删除不需要的镜像。

日志管理/日志设置

背景知识

如何在云开发控制台内查询服务日志，请参见 服务日志查询。
关于腾讯云 Elasticsearch service 的介绍，请参见 Elasticsearch service 文档。

使用限制

- 服务可以选择将日志输入到云开发“日志管理”模块中，或输出到指定的 ES 中。创建服务成功后**不支持修改日志选项**。
- 选择“自定义 Elasticsearch”，则在云开发或云托管控制台上无法查看您的服务日志，请前往 Elasticsearch service 控制台 查看。

服务域名

- 默认公网域名：公网任意来源可以用来对服务进行 HTTPS 调用，会产生相应的公网流量费用。
- 内网域名：仅同一 VPC 内其他云服务（包括但不限于其他云托管服务、云服务器等）可用于访问，不产生公网流量费用。
- 允许公网通过服务域名访问服务：关闭后，通过默认公网域名/自定义公网域名访问服务将报错。当您希望服务仅内网可访问，可选择关闭。无论是否开启，不影响通过内网域名访问服务。

> 说明：
>
> - 关闭“允许公网通过服务域名访问服务”后，若内网域名也同时没有被调用，则从任何角度都没有流量会打到当前服务，会引发服务下所有版本缩容至最小副本个数。
> - 同一 VPC 内，通过内网域名访问速度会高于通过公网域名访问。

HTTP 访问服务配置

在服务域名之外，可选择性启用 HTTP 访问服务。

背景知识

关于云开发 HTTP 访问服务的介绍，请参见 HTTP 访问服务文档。

使用限制

- 是否启用、如何配置 HTTP 访问服务，均不影响对服务域名的使用。
- 关闭“允许公网通过服务域名访问服务”开关，不影响公网通过在 HTTP 访问服务中配置的地址访问服务。
- 公网通过在 HTTP 访问服务中配置的地址访问服务，会产生相应的公网流量费用。

域名备案

最近更新时间：2021-01-06 14:10:19

备案背景

为什么需要备案?

根据国务院令第292号《互联网信息服务管理办法》和工信部令第33号《非经营性互联网信息服务备案管理办法》规定，国家对经营性互联网信息服务实行许可制度，对非经营性互联网信息服务实行备案制度。未获取许可或者未履行备案手续的，不得从事互联网信息服务，否则属于违法行为。

因此，使用云托管并**绑定自定义域名**时必须先办理网站备案，备案成功并获取通信管理局下发的 ICP 备案号后才能开通域名访问。

- ICP 备案号以工信部网站公共查询为准：查询入口
- 更多相关法律法规请参见：法律法规

不备案的影响

如果网站域名未办理备案就解析到腾讯云云托管上，将被腾讯云阻断并跳转到固定页面，提醒您尽快完成备案。若需要搭建网站，请先完成网站备案再开通网站。

如果网站域名已存在备案号，但之前不是在腾讯云备案，需在腾讯云办理接入备案。如果网站未在腾讯云办理备案但解析至腾讯云云托管，将被腾讯云阻断并跳转到固定页面。详情请参见 接入备案 流程。

怎样才算备案成功

备案成功后，通信管理局会分配主体备案号给备案主体（个人或单位均可称为主体），同时也会给此次备案的网站分配网站备案号。

备案号	格式	示例（腾讯云官网-域名 cloud.tencent.com）
主体备案号	省简称 ICP 备 主体序列号	京 ICP 备 11018762号
网站备案号	省简称 ICP 备 主体序列号-网站序列号	粤 B2-20090059-1

备案准备

为了节约备案时间和顺利通过备案，建议您提前了解备案流程。
因各地管局要求不同，需准备的材料也有所不同。建议您提前了解各省、自治区、直辖市管局的 备案要求，以及相关 备案限制。

备案次数

- 每个云开发环境下，如果有符合条件的云托管服务，则可以获得1个可备案的 IP，如有多个云开发环境，则可获得多个可备案的 IP。
- 每个 IP 可以备案2次。

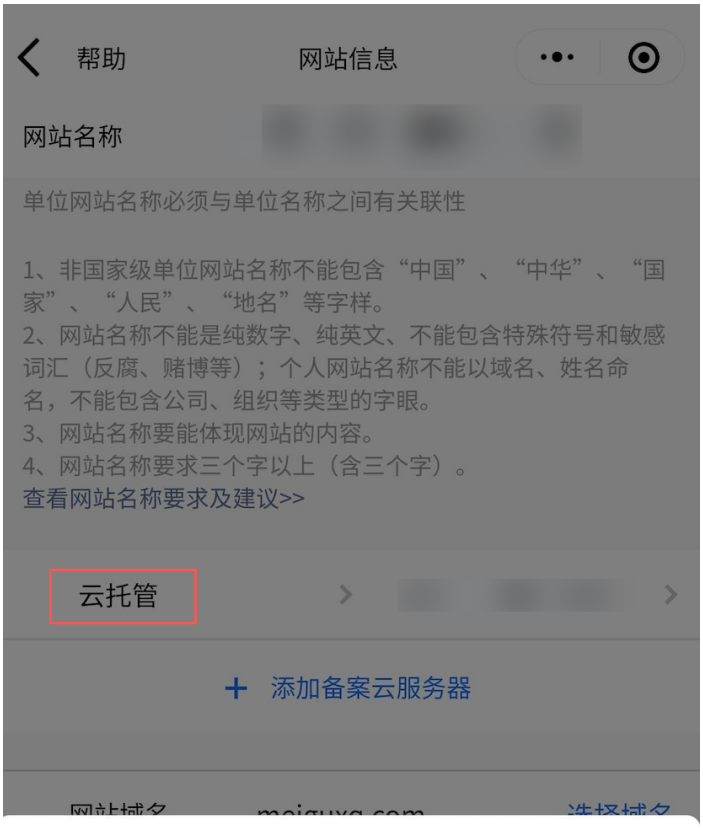
备案流程

请根据您的具体场景，查看以下对应的备案流程：

腾讯云云开发用户

1. 前往 云开发域名备案 页面，购买域名+云托管资源包组合（无域名），或单独购买云托管资源包（有域名）。购买时，选择新建环境/已有环境均可备案。
2. 等待创建云开发环境及云托管备案初始化，此过程最多需要10分钟。您无需前往云托管控制台进行任何操作。
3. 前往备案小程序，按小程序提示进行操作。

4. 执行【填写网站信息】步骤时，需在**备案类型中选择【云托管】，IP 可任意选择**。



> **注意**
>
> **如果您在上述第4步中无法选到【云托管】，或选中【云托管】后无可用 IP 地址，请参照以下步骤检查：**
>
> **i. 前往 云托管控制台。**

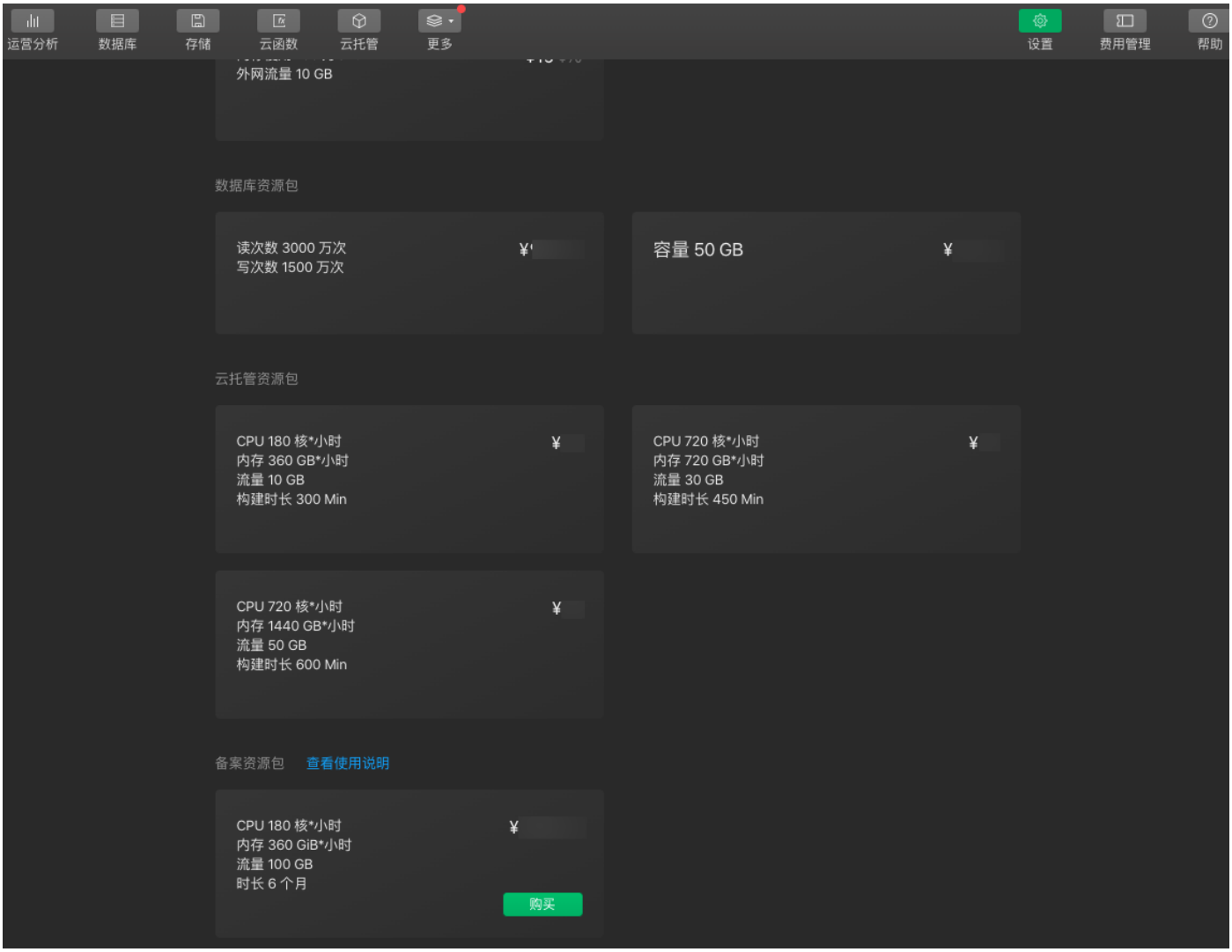
ii. 选择您在第1步中购买云托管资源包的环境，将看到名称为 “tcb-beian-auto” 的服务。如果找不到该服务，请 提交工单 联系我们进行排查。

iii. 单击该服务进入管理页面，在【服务配置】页面“域名备案”字段，查看状态是否为“可备案”。如不可备案，页面会展示不可备案的原因。若按提示原因操作仍未解决问题，请 提交工单 联系我们进行排查。

小程序云开发用户

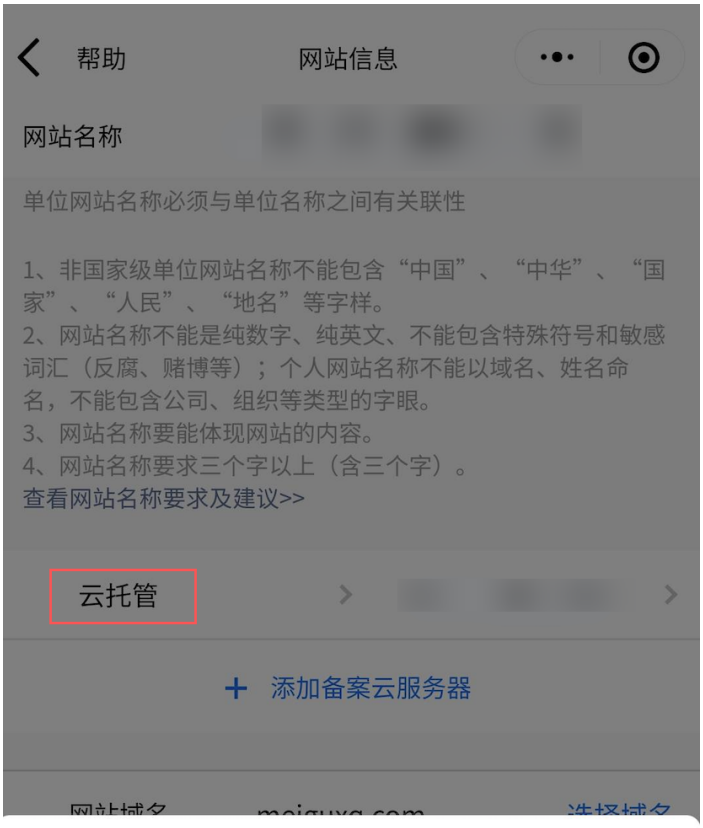
1. 打开微信开发者工具，前往云开发控制台。
2. 单击右上角的【设置】进入设置页，切换到“按量付费”。（已经是“按量付费”模式请忽略此步骤）

3. 将设置页下拉到底部，购买备案资源包。



4. 前往备案小程序，按小程序提示进行操作。

5. 执行【填写网站信息】步骤时，需在**备案类型中选择【云托管】，IP 可任意选择**。



> **注意：**
>
> **如果您在第5步中无法选到【云托管】，或选中【云托管】后无可用 IP 地址，请参照以下步骤检查：**
>
> **i. 打开微信开发者工具，前往云开发控制台。**
> **ii. 在您购买备案资源包的环境，前往云托管页面，将看到名称为 “tcb-beian-auto” 的服务。如果找不到该服务，请 提交工单 联系我们进行排查。**

更多域名备案文档

备案场景	查看文档
网站托管在腾讯云云托管，且网站的主办者和域名从未办理过备案。	首次备案
如果您已经在其他接入商处完成备案取得备案号，现需要更换服务商，例如腾讯云。	接入备案
如果您已在腾讯云进行备案，现需要新增网站。	新增网站（原备案在腾讯云）
如果您未在腾讯云进行备案，如果您想更换服务商（例如腾讯云）并且需要新增网站。	新增网站（原备案不在腾讯云）
已成功备案的网站需要变更主体或网站信息或腾讯云通知您修改备案相关信息。	变更备案
已成功备案的网站需要变更网站信息。（包括更换用于备案的云服务，例如从云服务备案变更为云托管备案）	变更网站
已成功备案，因故需要注销主体以及主体下的所有网站。	注销主体
已成功备案，因故需要注销主体下的任意网站。	注销网站
已成功备案，您的网站因故不再指向腾讯云服务器。	取消接入网站

常见问题

请参见 域名备案相关问题。

容器构建与部署
构建并部署 Node.js 应用

最近更新时间：2020-11-04 11:26:18

代码示例：Node.js

可单击下方按钮一键部署：

部署到云开发

第 1 步：编写基础应用

创建名为 helloworld 的新目录，并转到此目录中：

```
mkdir helloworld
cd helloworld
```

创建一个包含以下内容的 package.json 文件：

```
{
"name": "helloworld",
"description": "Simple hello world sample in Node",
"version": "1.0.0",
"main": "index.js",
"scripts": {
"start": "node index.js"
},
"author": "Tencent CloudBase",
"license": "Apache-2.0",
"dependencies": {
"express": "^4.17.1"
}
}
```

在同一目录中，创建一个 index.js 文件，并将以下代码行复制到其中：

```
const express = require("express");
const app = express();
app.get("/", (req, res) => {
res.send(`Hello World!`);
});
const port = 8080;
app.listen(port, () => {
console.log(`helloworld: listening on port ${port}`);
});
```

此代码会创建一个基本的 Web 服务器，侦听 8080 端口。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用官方 Node.js 12 轻量级镜像.
# https://hub.docker.com/_/node
FROM node:12-slim
# 定义工作目录
WORKDIR /usr/src/app
# 将依赖定义文件拷贝到工作目录下
COPY package*.json ./
# 以 production 形式安装依赖
RUN npm install --only=production
# 将本地代码复制到工作目录内
COPY . ./
# 启动服务
CMD [ "node", "index.js" ]
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
Dockerfile
.dockerignore
node_modules
npm-debug.log
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld .
```

构建成功后，运行 `docker images`，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld latest 1c8dfb88c823 8 seconds ago 146MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

构建并部署 PHP 应用

最近更新时间：2020-11-04 11:26:22

代码示例：PHP

可单击下方按钮一键部署：

部署到云开发

第 1 步：编写基础应用

创建名为 helloworld-php 的新目录，并转到此目录中：

```
mkdir helloworld-php
cd helloworld-php
```

创建名为 index.php 的文件，并将以下代码粘贴到其中：

```
<?php
echo sprintf("Hello World!");
```

此代码会对所有请求响应“Hello World”，HTTP 处理由容器中的 Apache Web 服务器进行。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用官方 PHP 7.3 镜像.
# https://hub.docker.com/_/php
FROM php:7.3-apache
# 将本地代码复制到容器内
COPY index.php /var/www/html/
# Apache 配置文件内使用 8080 端口
RUN sed -i 's/80/8080/g' /etc/apache2/sites-available/000-default.conf /etc/apache2/
ports.conf
# 将 PHP 配置为开发环境
# 如果您需要配置为生产环境，可以运行以下命令
```

```
# RUN mv "$PHP_INI_DIR/php.ini-production" "$PHP_INI_DIR/php.ini"
# 参考：https://hub.docker.com/_/php#configuration
RUN mv "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
Dockerfile
README.md
vendor
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld-php .
```

构建成功后，运行 docker images，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld-php latest 1c8dfb88c823 8 seconds ago 411MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

构建并部署 Java 应用

最近更新时间：2020-11-04 11:26:29

代码示例：Java

可单击下方按钮一键部署：

部署到云开发

第 1 步：编写基础应用

首先我们创建一个 Spring Boot 应用。

使用 curl 和 unzip 命令新建一个空 Web 项目：

```
curl https://start.spring.io/starter.zip \
-d dependencies=web \
-d javaVersion=1.8 \
-d bootVersion=2.3.3.RELEASE \
-d name=helloworld \
-d artifactId=helloworld \
-d baseDir=helloworld \
-o helloworld.zip
unzip helloworld.zip
cd helloworld
```

上述命令将创建一个 Spring Boot 项目。

> 说明：
>
> 如需在 Windows 系统上使用上述 curl 命令，您需要以下命令行之一：
>
> - WSL（推荐）
> - cygwin
>
> 或者选择性地使用 Spring Initializr（预加载配置）生成项目：

将 `src/main/java/com/example/helloworld/HelloworldApplication.java` 内容更新如下：

```
package com.example.helloworld;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
public class HelloworldApplication {
@RestController
class HelloworldController {
@GetMapping("/")
String hello() {
return "Hello World!";
}
}
public static void main(String[] args) {
SpringApplication.run(HelloworldApplication.class, args);
}
}
```

在 src/main/resources/application.properties 中，将服务器端口设置成 8080：

```
server.port=8080
```

以上代码会创建一个基本的 Web 服务器，并监听 8080 端口。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用官方 maven/Java 8 镜像作为构建环境
# https://hub.docker.com/_/maven
FROM maven:3.6-jdk-11 as builder
# 将代码复制到容器内
WORKDIR /app
COPY pom.xml .
```

```
COPY src ./src
# 构建项目
RUN mvn package -DskipTests
# 使用 AdoptOpenJDK 作为基础镜像
# https://hub.docker.com/r/adoptopenjdk/openjdk8
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
FROM adoptopenjdk/openjdk11:alpine-slim
# 将 jar 放入容器内
COPY --from=builder /app/target/helloworld-*.jar /helloworld.jar
# 启动服务
CMD ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/helloworld.jar"]
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
Dockerfile
.dockerignore
target/
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld-java .
```

构建成功后，运行 docker images，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld-java latest c994813b495b 8 seconds ago 271MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

构建并部署 Python 应用

最近更新时间：2020-11-04 11:32:00

代码示例：Python

可单击下方按钮一键部署：

部署到云开发

第 1 步：编写基础应用

创建名为 helloworld-python 的新目录，并转到此目录中：

```
mkdir helloworld-python
cd helloworld-python
```

创建名为 main.py 的文件，并将以下代码粘贴到其中：

```
import os
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
return 'Hello World!'
if __name__ == "__main__":
app.run(debug=True, host='0.0.0.0', port=8080)
```

以上代码会创建一个基本的 Web 服务器，并监听 8080 端口。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用官方 Python 轻量级镜像
# https://hub.docker.com/_/python
FROM python:3.8-slim
# 将本地代码拷贝到容器内
```

```
ENV APP_HOME /app
WORKDIR $APP_HOME
COPY . ./
# 安装依赖
RUN pip install Flask gunicorn
# 启动 Web 服务
# 这里我们使用了 gunicorn 作为 Server，1 个 worker 和 8 个线程
# 如果您的容器实例拥有多个 CPU 核心，我们推荐您把线程数设置为与 CPU 核心数一致
CMD exec gunicorn --bind :8080 --workers 1 --threads 8 --timeout 0 main:app
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
Dockerfile
README.md
*.pyc
*.pyo
*.pyd
__pycache__
.pytest_cache
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld-python .
```

构建成功后，运行 docker images，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld-python latest 1c8dfb88c823 8 seconds ago 123MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

构建并部署 C#（.NET）应用

最近更新时间：2020-11-04 11:26:38

第 1 步：编写基础应用

安装 .NET Core SDK 3.1。在 Console 中，使用 dotnet 命令新建一个空 Web 项目：

```
dotnet new web -o helloworld-csharp
cd helloworld-csharp
```

更新 Program.cs 中的 CreateHostBuilder 定义，侦听 8080 端口：

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;
namespace helloworld_csharp
{
public class Program
{
public static void Main(string[] args)
{
CreateHostBuilder(args).Build().Run();
}
public static IHostBuilder CreateHostBuilder(string[] args)
{
string port = "8080";
string url = String.Concat("http://0.0.0.0:", port);
return Host.CreateDefaultBuilder(args)
.ConfigureWebHostDefaults(webBuilder =>
{
webBuilder.UseStartup<Startup>().UseUrls(url);
});
}
}
}
```

将 Startup.cs 的内容更新为如下：

```
using System;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
namespace helloworld_csharp
{
public class Startup
{
// This method gets called by the runtime. Use this method to add services to the co
ntainer.
// For more information on how to configure your application, visit https://go.micro
soft.com/fwlink/?LinkID=398940
public void ConfigureServices(IServiceCollection services)
{
}
// This method gets called by the runtime. Use this method to configure the HTTP req
uest pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
app.UseRouting();
app.UseEndpoints(endpoints =>
{
endpoints.MapGet("/", async context =>
{
await context.Response.WriteAsync("Hello World!\n");
});
});
}
}
}
```

以上代码会创建一个基本的 Web 服务器，并监听 8080 端口。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用微软官方 .NET 镜像作为构建环境
# https://hub.docker.com/_/microsoft-dotnet-core-sdk/
FROM mcr.microsoft.com/dotnet/core/sdk:3.1-alpine AS build
WORKDIR /app
# 安装依赖
COPY *.csproj ./
RUN dotnet restore
# 将本地代码拷贝到容器内
COPY . ./
WORKDIR /app
# 构建项目
RUN dotnet publish -c Release -o out
# 使用微软官方 .NET 镜像作为运行时镜像
# https://hub.docker.com/_/microsoft-dotnet-core-aspnet/
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-alpine AS runtime
WORKDIR /app
COPY --from=build /app/out ./
# 启动服务
ENTRYPOINT ["dotnet", "helloworld-csharp.dll"]
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
**/obj/
**/bin/
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld-csharp .
```

构建成功后，运行 `docker images`，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld-csharp latest 1c8dfb88c823 8 seconds ago 105MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

构建并部署 Go 应用

最近更新时间：2021-01-06 17:06:08

代码示例：Go

可单击下方按钮一键部署：

部署到云开发

第 1 步：编写基础应用

创建名为 helloworld 的新目录，并转到此目录中：

```
mkdir helloworld
cd helloworld
```

创建一个包含以下内容的 go.mod 文件：

```
module helloworld
go 1.13
```

在同一目录中，创建一个 main.go 文件，并将以下代码行复制到其中：

```
package main
import (
"fmt"
"log"
"net/http"
)
func main() {
http.HandleFunc("/", handler)
port := "8080"
if err := http.ListenAndServe(":"+port, nil); err != nil {
log.Fatal(err)
}
}
func handler(w http.ResponseWriter, r *http.Request) {
```

```
fmt.Fprintf(w, "Hello World!\n")
}
```

此代码会创建一个基本的 Web 服务器，侦听 8080 端口。

第 2 步：将应用容器化

在项目根目录下，创建一个名为 Dockerfile 的文件，内容如下：

```
# 使用官方 Golang 镜像作为构建环境
FROM golang:1.15-buster as builder
WORKDIR /app
# 安装依赖
COPY go.* ./
RUN go mod download
# 将代码文件写入镜像
COPY . ./
# 构建二进制文件
RUN go build -mod=readonly -v -o server
# 使用裁剪后的官方 Debian 镜像作为基础镜像
# https://hub.docker.com/_/debian
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
FROM debian:buster-slim
RUN set -x && apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -y \
ca-certificates && \
rm -rf /var/lib/apt/lists/*
# 将构建好的二进制文件拷贝进镜像
COPY --from=builder /app/server /app/server
# 启动 Web 服务
CMD ["/app/server"]
```

添加一个 .dockerignore 文件，以从容器映像中排除文件：

```
vendor/
.dockerignore
```

```
.gcloudignore
.gitignore
```

第 3 步（可选）：本地构建镜像

如果您本地已经安装了 Docker，可以运行以下命令，在本地构建 Docker 镜像：

```
docker build -t helloworld .
```

构建成功后，运行 `docker images`，可以看到构建出的镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
helloworld latest 6948f1ebee94 8 seconds ago 82.7MB
```

随后您可以将此镜像上传至您的镜像仓库。

第 4 步：部署到 CloudBase 云托管

请参考 部署服务 与 版本配置说明。

高级
服务监控

最近更新时间：2021-03-01 11:34:08

本文介绍如何在云托管控制台查看服务监控数据。

操作步骤

1. 登录 云托管控制台，再按需要切换到指定的环境。
2. 单击【监控】切换到监控选项卡，即可查看当前环境下所有服务、服务下所有版本的监控数据。
3. 选择您需要查看的服务，单击服务名称进入服务详情页面。
4. 单击【监控】切换到监控选项卡，即可查看当前服务、服务下所有版本的监控数据。
5. 选择您需要查看的版本，单击版本名称进入版本详情页面。
6. 单击【监控】切换到监控选项卡，即可查看当前版本监控数据。

监控字段说明

- 在环境/服务/版本维度监控选项卡中，顶部时间筛选器对页面内所有监控数据生效。例如：在顶部选择时间"7天"，则统计卡片、统计曲线，均展示过去 7 天的数据。
- 监控曲线图中的“粒度”指每个监控数据对应的单位时间，会随着所选时间区间变化，时间跨度长则粒度粗。**曲线图上所有数值需配合粒度解读**。

统计卡片

- **调用次数**：收到的请求次数（包括通过服务域名、HTTP 访问服务产生的请求），以及请求“异常”的次数。
- **版本部署**：总部署次数（包括新建版本部署，以及版本重新部署），以及“部署失败”的次数。
- **实时副本数峰值**：扩容曾达到过的最大值，以及最大值的出现时间。可作为设置版本的副本最大个数的参考值。

服务监控

- **调用次数**：单位时间内服务收到的请求次数总和（包括通过服务域名、HTTP 访问服务产生的请求）。
- **平均响应时间 RT**：单位时间内服务收到的所有请求的响应时间取平均值。
- **QPS**：单位时间内服务平均每秒处理的请求数。
- **HTTP 错误**：单位时间内请求服务失败返回 HTTP 错误的次数。
- **CPU 用量**：单位时间内服务所有版本 CPU 资源消耗之和，单位为（核 x 小时）。
- **内存用量**：单位时间内服务所有版本内存资源消耗之和，单位为（GiB x 小时）。
- **实例个数**：服务所有版本的实时实例个数之和，在单位时间内取平均值。
- **实例状态不正常个数**：服务所有版本的实时实例状态不正常个数之和，在单位时间内取平均值。

版本监控

- **调用次数**：单位时间内版本收到的请求次数总和（包括通过服务域名、HTTP 访问服务产生的请求）。
- **平均响应时间 RT**：单位时间内版本收到的所有请求的响应时间取平均值。
- **QPS**：单位时间内版本平均每秒处理的请求数。
- **HTTP 错误**：单位时间内请求版本失败返回 HTTP 错误的次数。
- **CPU 用量**：单位时间内版本 CPU 资源消耗，单位为（核 x 小时）。
- **内存用量**：单位时间内版本内存资源消耗之和，单位为（GiB x 小时）。
- **CPU 使用率**：版本所有实例 CPU 使用率平均值，在单位之间内再取平均值。可作为设置扩缩容条件的参考值。
- **内存使用率**：版本所有实例内存使用率平均值，在单位之间内再取平均值。
- **实例个数**：版本的实时实例个数，在单位时间内取平均值。
- **实例状态不正常个数**：版本的实时实例状态不正常个数，在单位时间内取平均值。

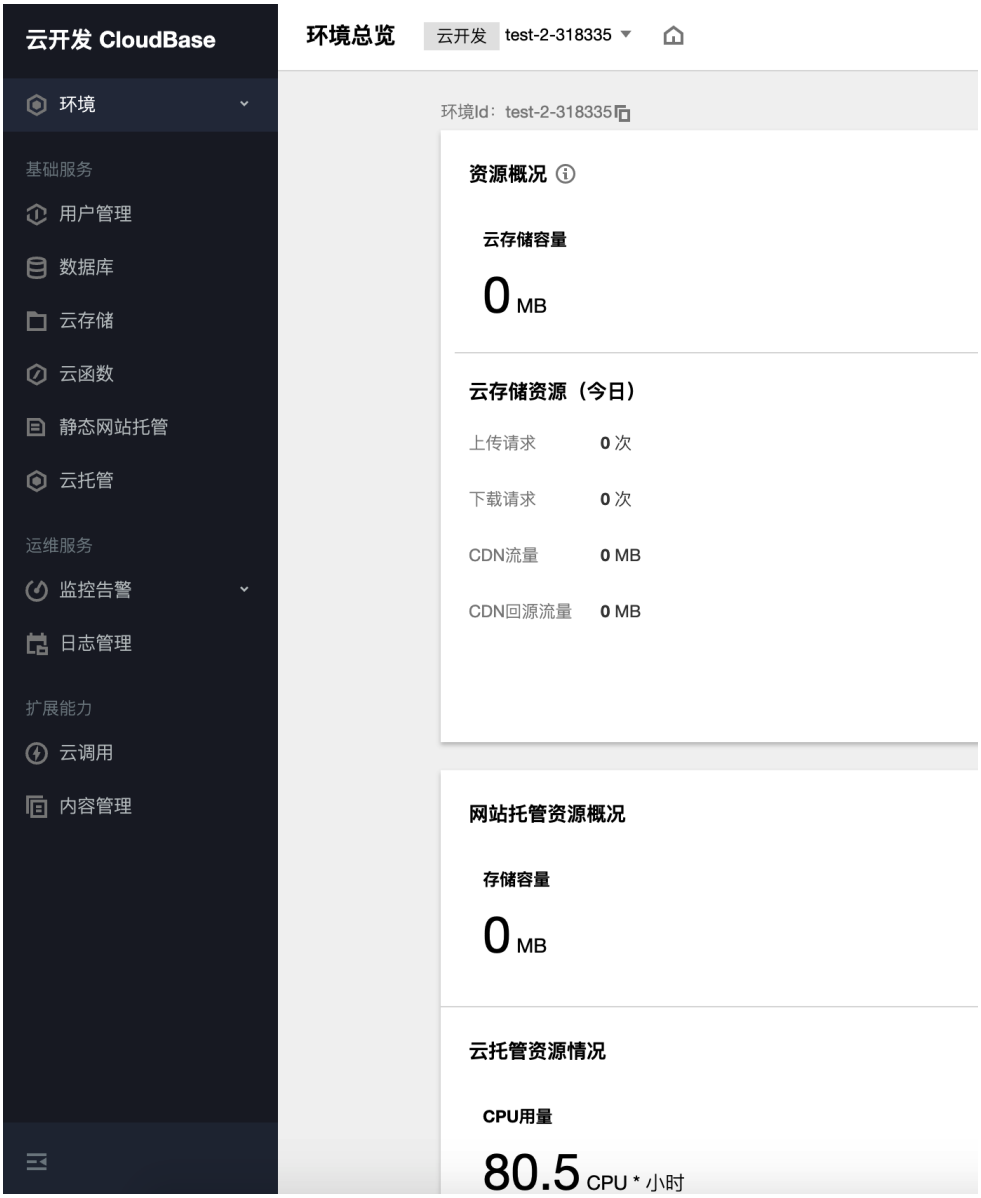
查询服务日志

最近更新时间：2020-10-21 11:13:40

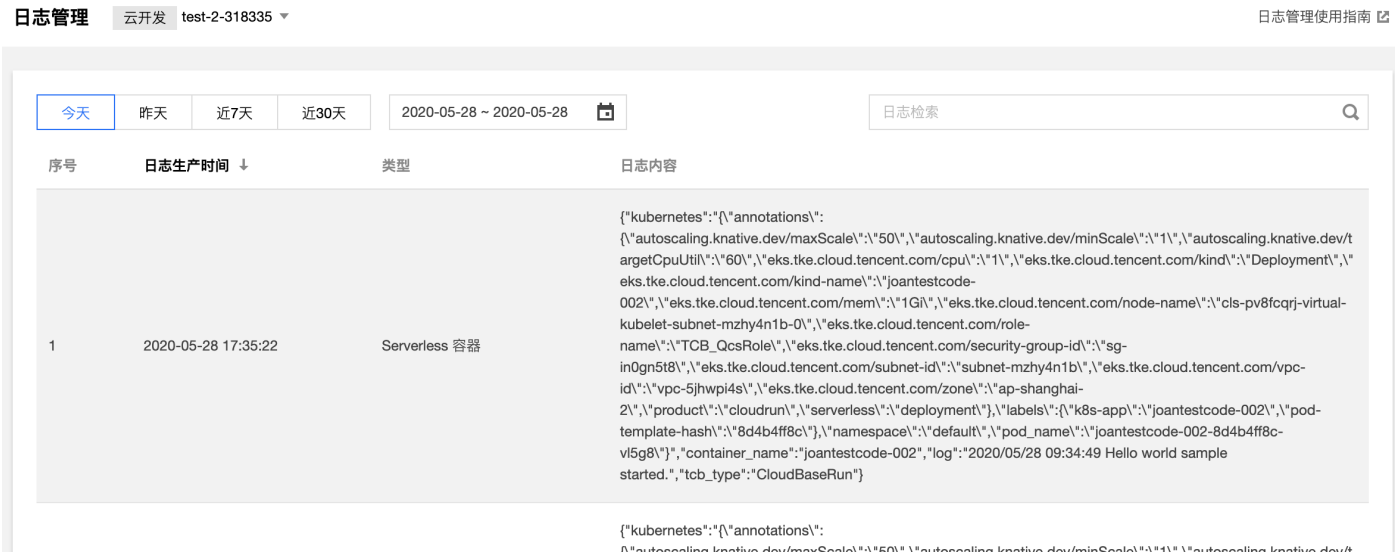
本文介绍如何在控制台查询服务日志。云托管中运行的代码中的标准输出，都会被捕捉并上报到云开发的日志系统中。

操作步骤

1. 登录 云开发控制台， 进入指定的环境后，在左侧菜单中，找到【 日志管理 】入口。



2. 查询日志。具体方法参考云开发文档 日志管理。



优化容器镜像

最近更新时间：2020-11-04 10:57:54

CloudBase 云托管支持托管任意镜像，但我们建议您**尽可能地优化您的镜像**，以获得更快的启动速度、更快的构建速度、更小的容器体积、更优的服务性能。

以下是一些推荐做法：

选择更小的基础镜像

同样一种语言的运行时容器，由于选择的基础容器不同，体积也会有很大的差异，**对于大多数业务，我们推荐您使用更小的基础镜像**。

以 Node.js 为例，官方镜像仓库 提供了以下不同体积的基础镜像：

```
REPOSITORY TAG IMAGE ID CREATED SIZE
node 15 ca36fba5ad66 2 days ago 941MB
node 15-slim 922b09b21278 2 days ago 165MB
node 15-buster 36754275e286 2 days ago 910MB
node 15-buster-slim eda2c7e487ff 2 days ago 179MB
node 15-alpine 1e8b781248bb 2 days ago 115MB
```

可以注意到 `15-slim`、`15-buster-slim`、`15-alpine` 的体积明显较小，这是由于它们使用了经过裁剪的底层操作系统镜像。使用这些裁剪、优化后的镜像通常不会影响您服务的运行。

减少镜像层数

Dockerfile 中的每一行指令，都会生成一个层，镜像的层数会直接影响镜像的体积大小。

我们推荐您**将多个指令合并串联，以减少层的数量**。例如，您可以将以下的指令合并为同一个：

```
RUN cd my-app
RUN make
RUN make install
RUN rm -rf /tmp
```

合并后：

```
RUN cd my-app && \
make && \
make install && \
rm -rf /tmp
```

充分利用层的缓存

Docker 会对镜像的每一层单独进行缓存，如果层的内容没有变化，那么会直接使用之前的缓存，以提高构建、上传镜像的速度。为了能尽量复用这一机制，我们推荐您**将镜像变动不大的层独立出来**。

例如，您的应用可能存在诸多依赖，通常来说，这些依赖在开发过程中变动较小，所以依赖的拷贝最好独立成为一层指令：

```
COPY ./deps deps
RUN make && make install
```

以上的做法将 COPY 语句独立出来，每次构建、推送镜像时，只要依赖的文件内容没有变化，那么都可以复用之前的缓存，以提高构建、推送速度。

清理不必要的文件

您可以使用 .dockerignore 在容器构建时忽略文件，以减少非必要文件的导入，同时也可以提高安全性，避免将一些本地敏感文件打包到镜像中。

多阶段构建

以 Java 服务为例，实际运行的过程中只需要最后编译生成的 jar 文件即可，而构建期使用的依赖、扩展包以及源代码文件都是不必要的。

此时我们可以使用多阶段构建的方式，**分离构建环境和运行环境**，从而精简运行时容器的体积，例如以下的 Dockerfile：

```
# 使用官方 maven/Java 8 镜像作为构建环境
# https://hub.docker.com/_/maven
FROM maven:3.6-jdk-11 as builder

# 将代码复制到容器内
WORKDIR /app
```

```
COPY pom.xml .
COPY src ./src

# 构建项目
RUN mvn package -DskipTests

# 使用 AdoptOpenJDK 作为基础镜像
# https://hub.docker.com/r/adoptopenjdk/openjdk8
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
FROM adoptopenjdk/openjdk11:alpine-slim

# 将 jar 放入容器内
COPY --from=builder /app/target/helloworld-*.jar /helloworld.jar

# 启动服务
CMD ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/helloworld.jar"]
```

使用 Webshell 调试服务

最近更新时间：2020-10-21 11:13:44

本文为您介绍云托管的 Webshell 功能，以及如何使用 Webshell 完成基本的运维需求。

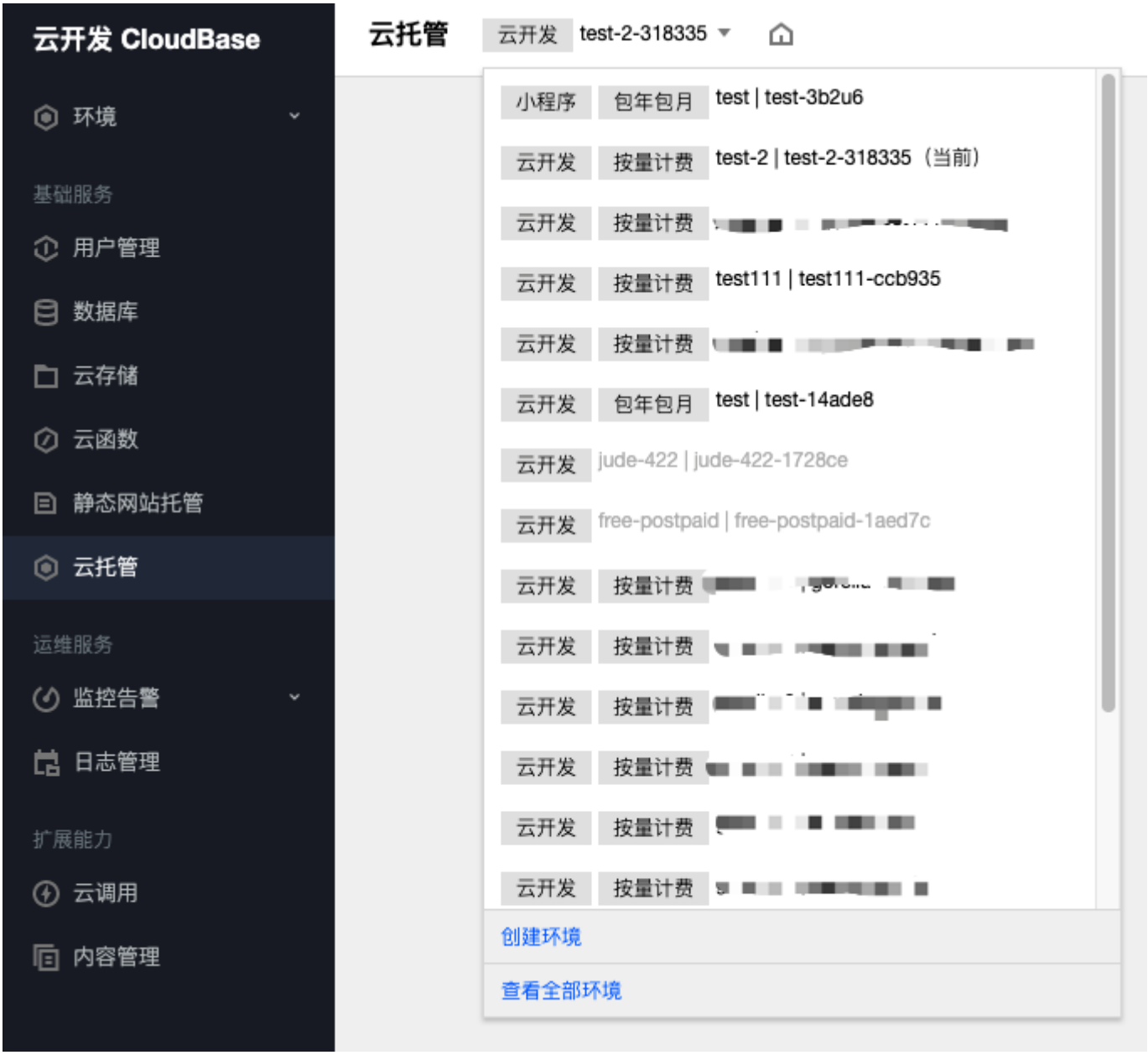
操作背景

容器是一个暂态的、供服务运行的环境，您在使用云托管时只需关注自己的服务，不需要涉及对容器的直接操作，包括创建、配置、更新、重启、销毁等等。但为了方便进行线上问题定位、排查，特别是调试关于代码本身的问题，云托管在控制台提供了简易版 Webshell，供您查看并调试自己的容器。

> ⚠ 注意：
>
> 通过 Webshell 直接操作容器可能会带来风险。云托管只是为您提供了一个直达容器的途径，您在 Webshell 中做的一切操作，云托管都无法感知、无法管控。
>
> - 若您在 Webshell 中的操作引起容器 OOM，可能会带来服务中断。
> - 若您在 Webshell 中直接修改了容器配置，可能会导致与云托管中记录的容器配置不一致，引起后续操作混乱。

步骤1：登录控制台

登录 云托管控制台，再按需要切换到指定的环境。

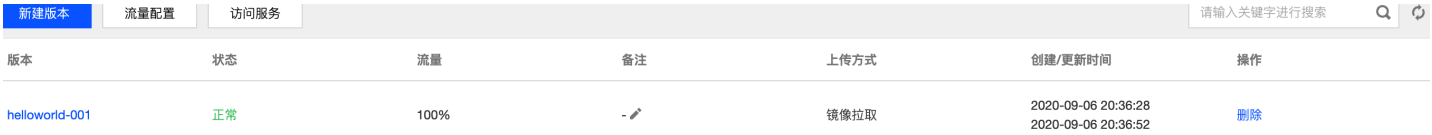


步骤2：进入服务详情页面

单击服务名称进入服务详情页面。

步骤3：进入版本详情页面

单击版本名称进入版本详情页面。



步骤4：进入实例页面

单击【实例】页签，进入实例管理页面。

步骤5：进入 Webshell 页面

单击需要调试的容器对应的【Webshell】，进入 Webshell 管理页面。

> 说明：
> 根据版本的“副本个数”、“扩缩容条件”和当前版本流量情况，您的版本下可能有多个实例（容器）。同一个版本下所有的容器都是根据“版本配置”创建出来的，配置信息完全一致，因此绝大多数情况下您任意选择一个容器进入 Webshell 都可对当前版本进行调试和问题定位。但不排除某些特殊情况下，仅有个别容器状态异常。

步骤6：调试

您可以开始对自己的服务进行调试。