

# 云开发 CLI 工具

## 开发指南

### 产品文档



腾讯云

## 【 版权声明 】

©2013–2023 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

## 【 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

### 开发指南

概述

多地域

配置文件

环境

环境基础

安全域名

登录方式

### 云函数

函数管理

函数配置项

触发器

函数部署

本地运行

代码更新

日志

函数版本

文件层

流量路由

预置并发

### 云存储

静态网站托管

HTTP 访问服务

从 v0 升级到 v1

# 开发指南

## 概述

最近更新时间：2022-02-18 16:48:21

CloudBase CLI 是一个开源的命令行界面交互工具，用于帮助用户快速、方便的部署项目，管理云开发资源。

### 安装 CloudBase CLI

#### 步骤1：安装 Node.js

如果本机没有安装 Node.js，请从 [Node.js](#) 官网下载二进制文件直接安装，建议选择版本为 LTS，版本必须为 8.6.0+。

#### 步骤2：安装 CLI

安装 `@cloudbase/cli`。

- 使用 NPM

```
npm i -g @cloudbase/cli
```

- 使用 Yarn

```
yarn global add @cloudbase/cli
```

#### ⚠ 注意：

如果 `npm install -g @cloudbase/cli` 失败，您可能需要 [修改 npm 权限](#)，或者以系统管理员身份运行如下代码：

```
sudo npm install -g @cloudbase/cli
```

#### 步骤3：测试安装是否成功

如果安装过程没有错误提示，一般就是安装成功了。下面，我们可以继续输入命令：

```
tcb -v
```

如果看到输出版本号，说明已经安装成功。

## 步骤4：所有命令

使用 `cloudbase -h` 查看所有可用命令

```
tcb -h
```



就可以看到目前 CLI 支持的所有能力和命令了。建议可以经常使用 `-h` 来查看命令。

## 代理设置

使用 CloudBase CLI 时，需要您的终端能够访问网络。**如果无法直接访问公网**，可以设置 HTTP 代理使 CLI 能够正常使用，CLI 会读取 `http_proxy` 或 `HTTP_PROXY` 环境变量，自动设置网络代理服务。

例如，您可以在终端中运行以下命令，设置 CLI 通过 `http://127.0.0.1:8000` 的代理服务访问网络：

```
export HTTP_PROXY=http://127.0.0.1:8000
```



### ⚠ 注意

`http://127.0.0.1:8000` 只是一个示例，请勿直接使用，具体的代理配置根据您的网络配置决定。

上面的命令只是临时设置，当您关闭终端后，代理会自动失效，下次开启终端后需要重新设置。如果需要一直通过代理访问公网，可以把命令加入到终端的配置文件中。

## 开通云开发服务

如果您已经拥有按量计费的云开发环境，您可以跳过此步骤。

在开始使用云开发服务之前，您需要登录腾讯云 [云开发控制台](#)，确保已经开通了云开发服务，并且已经创建了可以使用的环境。如果您不了解如何创建环境，可以参照云开发快速入门 – [开通环境](#) 文档进行操作。

## 登录

首先登录您的腾讯云账号，在获取到您的授权之后，CloudBase CLI 才能操作您的资源。CloudBase CLI 提供了两种获取授权的方式：腾讯云–云开发控制台授权以及腾讯云–云 API 密钥授权。

### 腾讯云–云开发控制台授权

在您的终端中输入下面的命令：

```
tcb login
```



CloudBase CLI 会自动打开云开发控制台获取授权，您需要单击同意授权按钮允许 CloudBase CLI 获取授权。如您没有登录，您需要登录后才能进行此操作。

## 腾讯云-云 API 密钥授权

### ⚠ 注意

腾讯云 API 密钥可以操作您名下的所有腾讯云资源，请妥善保管和定期更换密钥，当您更换密钥后，请及时删除旧密钥。

首先您需要到腾讯云官网获取 [云 API 密钥](#)，然后在终端中输入下面的命令：

```
tcb login --key
```



回车后，请按提示输入云 API 密钥的 SecretId 和 SecretKey 即可完成登录。

## CI 中的登录

在 CI（持续集成）构建中，您可以使用下面的方式通过 API 密钥直接登录，避免交互式输入：

```
tcb login --apiKeyId xxx --apiKey xxx
```



## 临时密钥登录

通过腾讯云临时密钥登录，可以在某些比较敏感的场景下使用，使用短期有效的密钥。

```
tcb login --apiKeyId xxx --apiKey xxx --token xxx
```



## 子账号

子账号需要主账号授权资源访问后，才可使用云开发资源。子账号默认没有访问云开发资源的权限，因此使用子账号登录云开发控制台，无法访问云开发资源。

此时，需要为子账号添加预设策略的方式来访问云开发资源。具体操作步骤如下：

1. 登录 [CAM 控制台](#)，选择左侧菜单 **用户** > **用户列表**。
2. 进入用户列表页面，单击 **新建用户**。
3. 进入新建用户页面，根据提示填写用户相关信息。
4. 信息填写完毕后，前往策略列表中选择 TCB 预设授权策略。
5. 单击 **完成**，即可完成创建子账号。

除了上述在创建子账号时添加策略的方式外，也可通过策略关联用户的方式授权，具体请参见 [授权管理](#) 指引。

## TCB 预设策略

您可以使用以下预设策略为您的子账号授予相关权限：

| 策略                                                      | 描述                                                                             |
|---------------------------------------------------------|--------------------------------------------------------------------------------|
| <code>QcloudAccessForTCBRole</code>                     | 该策略授予云开发（TCB）对云资源的访问权限。                                                        |
| <code>QcloudAccessForTCBRoleInAccessCloudBaseRun</code> | 该策略供云开发（TCB）服务角色（TCB_QcsRole）进行关联，用于 TCB 访问其他云服务资源。包含私有网络 VPC、云服务器 CVM 相关操作权限。 |

#### ⚠ 注意

该权限包含底层对象存储、云函数、日志、监控、VPC 等资源的全读写权限，一旦为子账号添加此策略，将意味着子账号拥有上述资源，并自动拥有后续新增资源的全读写权限，因此请慎重选择，详情可点击权限名称跳转查看权限规则。

以下预设策略是在您使用 TCB 服务时，授予子账号 CAM 只读访问权限。对子账号进行

`QcloudCamReadOnlyAccess` 授权操作后，子账号可使用 CLI 工具网页授权登录；若未进行授权，仅可使用子账号 **API 密钥** 进行登录。您可根据需要对子账号进行开通：

| 策略                                   | 描述                     |
|--------------------------------------|------------------------|
| <code>QcloudCamReadOnlyAccess</code> | 该策略授予用户与权限（CAM）只读访问权限。 |

## 创建项目与部署

### 步骤1：初始化

您可以使用下面的命令创建一个项目，创建项目时 CloudBase CLI 根据您的输入的项目名创建一个文件夹，并写入相关的配置和模板文件。

#### ⚠ 注意

使用 `tcb new` 命令需要 CLI 1.0+ 版本。

```
tcb new app node-starter
```



云开发项目是和云开发环境资源关联的实体，云开发项目聚合了云函数、数据库、文件存储等服务，您可以在云开发项目中编写函数，存储文件，并通过 CloudBase 快速的操作您的云函数、文件存储、数据库等资源。

云开发项目文件结构：

```
.
├── .gitignore
├── functions // 云函数目录
│   ├── node-app
│   │   └── index.js
└── cloudbaserc.json // 项目配置文件
```

## 步骤2：编写函数

默认情况下，所有 Node 和 PHP 函数都统一存放在 `functions` 目录下，并以函数名作为文件夹名称，如 `functions/cloudbase/index.js`。对于 Java 函数时，则需要将 `jar` 文件名修改为函数名称，放在 `functions` 目录下即可，例如 `functions/cloudbase.jar`。

如果您想将函数存放在其他目录，您可以通过配置文件中的 `functionRoot` 选项指定您想存放函数的目录，`functionRoot` 选项代表了云函数文件夹相对于项目根目录的路径。

例如，创建一个 Node.js 函数 App，下面是 `functions/app/index.js` 的内容。

```
"use strict";

exports.main = async (event, context) => {
  console.log("Hello World");
  console.log(event);
  console.log(context);
};
```

## 步骤3：修改配置

默认情况下，项目配置存储在 `cloudbaserc.json` 文件中，默认生成的函数配置为 Node 语言相关的配置，其他语言如 PHP，Java 等需要修改对应的 `handler`（运行入口）和 `runtime`（运行时），参见 [配置文件](#) `cloudbaserc.json` 文件说明部分。

如果您想指定其他文件作为配置文件，可以在使用 CLI 命令时添加 `--config-file config-path` 参数指定配置文件，目前支持 JS 和 JSON 格式的配置文件。

```
{
  {
    "envId": "xxx",
    "functionRoot": "./functions",
    "functions": [
      {
```



```
"name": "node-app",
"timeout": 5,
"envVariables": {},
"runtime": "Nodejs10.15",
"memorySize": 128,
"handler": "index.main"
}
]
```

## 步骤4：部署函数

最后，在项目根目录下（cloudbaserc.json 所在目录）运行 `tcb fn deploy node-app` 命令，即可部署 `node-app` 函数。

```
tcb fn deploy node-app
```



部署完成后可以使用 `tcb fn list` 命令查看已经部署完成的函数列表。

```
tcb fn list
```



## 使用说明

在介绍文档中，默认省略了环境 ID，默认在 cloudbaserc.json 文件所在目录使用 CloudBase CLI 命令。

## 所有命令

使用 `tcb -h` 查看所有可用命令：

```
tcb -h
```



```
Usage: cloudbase [options] [command]
```



Options:

|                                          |                |
|------------------------------------------|----------------|
| <code>--verbose</code>                   | 打印出内部运行信息      |
| <code>-r, --region &lt;region&gt;</code> | 指定环境地域         |
| <code>--mode &lt;mode&gt;</code>         | 指定加载 env 文件的环境 |
| <code>--config-file &lt;path&gt;</code>  | 指定配置文件路径       |
| <code>-v, --version</code>               | 输出当前版本         |

|                                                    |                              |
|----------------------------------------------------|------------------------------|
| -h, --help                                         | 查看命令帮助信息                     |
| Commands:                                          |                              |
| login [options]                                    | 登录腾讯云账号                      |
| logout                                             | 登出腾讯云账号                      |
| env list                                           | 展示云开发环境信息                    |
| env create <alias>                                 | 创建云开发免费环境                    |
| env rename [options] <name>                        | 修改云开发环境别名                    |
| env login list [options]                           | 列出环境登录配置                     |
| env login create [options]                         | 添加环境登录方式配置                   |
| env update [options]                               | 更新环境登录方式配置                   |
| env domain list [options]                          | 列出环境的安全域名列表                  |
| env domain create [options] <domain>               | 添加环境安全域名，多个以斜杠 / 分隔          |
| env domain delete [options]                        | 删除环境的安全域名                    |
| fn list [options]                                  | 展示云函数列表                      |
| fn deploy [options] [name]                         | 部署云函数                        |
| fn delete [options] [name]                         | 删除云函数                        |
| fn config                                          | 函数配置管理                       |
| fn invoke [options] [name]                         | 触发云端部署的云函数                   |
| fn log [options] <name>                            | 打印云函数日志                      |
| fn detail [options] <name>                         | 获取云函数信息                      |
| fn copy [options] <name> [newFunctionName]         | 拷贝云函数                        |
| fn run [options]                                   | 本地运行云函数（当前仅支持 Node）          |
| fn trigger                                         | 函数触发器操作                      |
| fn code                                            | 函数代码管理                       |
| fn code update [options] <name>                    | 更新云函数代码                      |
| fn code download [options] <name> [dest]           | 下载云函数代码                      |
| framework deploy [options] [module]                | 云开发 Serverless 应用框架：部署全栈应用   |
| framework compile [options] [module]               | 云开发 Serverless 应用框架：编译应用描述文件 |
| framework run [options] [command]                  | 云开发 Serverless 应用框架：执行自定义命令  |
| hosting detail [options]                           | 查看静态网站服务信息                   |
| hosting deploy [options] [filePath] [cloudPath]    | 部署静态网站文件                     |
| hosting delete [options] [cloudPath]               | 删除静态网站文件/文件夹，文件夹需指定 --dir 选项 |
| hosting list [options]                             | 展示文件列表                       |
| hosting download [options] <cloudPath> [localPath] | 下载文件/文件夹，文件夹需指定 --dir 选项     |
| storage upload [options] <localPath> [cloudPath]   | 上传文件/文件夹                     |
| storage download [options] <cloudPath> <localPath> | 下载文件/文件夹，文件夹需指定 --dir 选项     |
| storage delete [options] [cloudPath]               | 删除文件/文件夹，文件夹需指定 --dir 选项     |

|                                          |                          |
|------------------------------------------|--------------------------|
| storage list [options] [cloudPath]<br>列表 | 获取文件存储的文件列表，不指定路径时获取全部文件 |
| storage url [options] <cloudPath>        | 获取文件临时访问地址               |
| storage detail [options] <cloudPath>     | 获取文件信息                   |
| storage get-acl [options]                | 获取文件存储权限信息               |
| storage set-acl [options]                | 设置文件存储权限信息               |
| service create [options]                 | 创建 HTTP 访问服务             |
| service delete [options]                 | 删除 HTTP 访问服务             |
| service domain list [options]            | 查询 HTTP 访问服务自定义域名        |
| service domain unbind [options] <domain> | 解绑自定义 HTTP 访问服务域名        |
| service list [options]                   | 获取 HTTP 访问服务列表           |
| service switch [options]                 | 开启/关闭 HTTP 访问服务          |
| service auth                             | HTTP 访问服务访问鉴权管理          |
| open [link]                              | 在浏览器中打开云开发相关连接           |
| new <appName> [template]                 | 创建新的云开发应用                |

## 示例

- 登录

```
tcb login
```



- 创建新的云开发应用

```
tcb new appName
```



- 部署云函数

```
tcb fn deploy
```



- 查看命令使用介绍

```
tcb fn -h
```



# 多地域

最近更新时间：2022-02-18 16:43:35

CLI 1.2.0 版本起，增加了对多地域的支持，您可以通过 CLI 操作云开发不同地域的资源。

## 说明

默认情况下，CLI 可以直接操作上海地域的环境，当您需要操作其他地域的环境时，请通过 `-r` 全局参数指定地域，例如：

```
# 展示出广州地域的环境列表
tcb env list -r gz

# 将函数部署到广州地域的环境
tcb fn deploy app -r gz -e envID

...
```

当不指定 `-r` 参数时，即视为环境为上海地域的环境。

## 说明

`-r` 参数适用于全部的命令。

## 地域列表

`gz` 为广州地域的缩写，全称为 `ap-guangzhou`，在使用 CLI 时，可以使用缩写，下面是全部地域的简称信息：

```
# 已支持地域
gz: ap-guangzhou
bj: ap-beijing
sh: ap-shanghai

# 可能会支持的地域
# nj: ap-nanjing
# cd: ap-chengdu
# cq: ap-chongqing
```

## 配置文件

您也可以在配置文件中指定环境的地域信息，无需添加 `-r` 参数，如下所示：

```
{
  "version": "2.0",
  "envId": "envId",
  "region": "ap-guangzhou",
  "framework": {
    "name": "koa-starter",
    "plugins": {
      "node": {
        "use": "@cloudbase/framework-plugin-node",
        "inputs": {
          "name": "koa-starter",
          "path": "/koa-starter"
        }
      }
    }
  }
}
```



# 配置文件

最近更新时间：2022-02-18 16:49:16

为了应对复杂的使用场景，我们提供了项目级的 `cloudbaserc.json` 配置文件，可以在 CLI 和 VS Code 插件中共享使用。

配置文件包含了使用云开发 CLI 或 VS Code 插件的过程中的相关配置，可以简化 CLI 和 VS Code 的使用。

默认情况下，使用 `cloudbase init` 初始化项目时，会生成 `cloudbaserc.json` 文件作为配置文件，您也可以使用 `--config-file` 指定其他文件作为配置文件，文件必须满足格式要求。

## 动态变量

CLI 0.9.1+ 版本引入了 2.0 新版本配置文件，支持了动态变量的特性。在 `cloudbaserc.json` 中声明 `"version": "2.0"` 即可启用新的特性，新版配置文件只支持 JSON 格式。

动态变量特性允许在 `cloudbaserc.json` 配置文件中使用动态变量，从环境变量或其他数据源获取动态的数据。使用 `{{}}` 包围的值定义为动态变量，可以引用数据源中的值。如下所示：

```
{
  "version": "2.0",
  "envId": "envId",
  "functionRoot": "./functions",
  "functions": [
    {
      "name": "{{variable}}"
    }
  ]
}
```

## 数据源

CloudBase 定义多个数据源的命名空间，用于区分不同的数据源。您可以通过 `命名空间.变量名` 引用数据，如 `{{tcb.envId}}`。

| 命名空间 | 变量名                | 含义                               |
|------|--------------------|----------------------------------|
| tcb  | <code>envId</code> | 配置文件或通过命令行参数指定的环境 Id             |
| util | <code>uid</code>   | 24 位的随机字符串                       |
| env  | <code>*</code>     | 从 <code>.env</code> 类型文件中加载的环境变量 |

## 环境变量

CloudBase 对环境变量做了特别支持，以解决不同开发阶段、不同配置，使用 CLI & Framework 部署资源的问题。CloudBase 支持使用 `.env` 类型文件作为主要数据源，使用不同的后缀区分不同的阶段、场景，如 `.env.development` 可以表示开发阶段的配置，`.env.production` 可以表示生产环境的配置等。

CloudBase 定义了一些约束：默认情况下，CLI & Framework 会自动加载 `.env` 和 `.env.local` 文件中的数据，开发者可以通过使用 `--mode <mode>` 选项添加特定环境的配置。

```
.env          # 在所有的环境中被载入
.env.local    # 在所有的环境中被载入，可以加入 .gitignore 忽略
.env.[mode]   # 只在指定的模式中被载入
```

在加载数据源时，`.env` 和 `.env.local` 会被直接加载（如果存在的话），当指定 `--mode [mode]` 时，会再加载 `.env.[mode]` 文件，并按照如下的顺序合并覆盖同名变量：`.env.[mode] > .env.local > .env`，即 `.env.[mode]` 中的同名变量会覆盖 `.env.local` 和 `.env` 文件中的同名变量，以此类推。

当使用 `tcb framework deploy --mode test` 命令时，会自动加载 `.env`，`.env.local` 以及 `.env.test` 等三个文件中的环境变量合并使用。

### 说明

我们建议您将密钥等私密配置放在 `.env.local` 文件中，并将 `.env.local` 加入 `.gitignore` 配置中。

如 `.env.local` 文件中存在以下变量

```
DB_HOST = localhost
DB_USER = root
DB_PASSWORD = s1mpl3
```

则可以在配置文件中使用

```
{
  "version": "2.0",
  "envId": "xxx",
  "functionRoot": "./functions",
  "functions": [
    {
      "name": "test",
      "envVariables": {
```

```
"PASSWORD": "{{env.DB_PASSWORD}}"  
}  
}  
]  
}
```

## 扩展 env 语法

一般情况下，您可以直接在 `env` 文件中使用键值对。

```
FOO=bar  
VUE_APP_SECRET=secret
```

但是，当我们需要使用复杂的环境变量时，简单的键值对就有些力不从心了。所以，CloudBase 扩展了 `.env` 支持的语法，支持了复合键值对，您可以通过 `.` 为同名键添加属性，例如：

```
Book.Name=Test  
Book.Publish=2020  
Book.Authors.0=Jack  
Book.Authors.1=Mike
```

会被编译为包含以下属性的 Book 对象。

```
{  
  "Name": "Test",  
  "Publish": "2020",  
  "Authors": ["Jack", "Mike"]  
}
```

您可以在 `cloudbaserc.json` 直接使用 `{{env.Book.Name}}` 引用相关属性。

### ⚠ 注意：

当使用的 `.env` 值为对象时，在编译时会被转换成 JSON 字符串，例如 `{{env.Book}}` 会被编译为 `{"Name":"Test","Publish":"2020","Authors":["Jack","Mike"]}`。

## 字段

字段说明如下表所示：

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|



| 字段           | 类型                   | 说明                                                                                                      |
|--------------|----------------------|---------------------------------------------------------------------------------------------------------|
| version      | String               | 当前配置文件的版本，目前支持的版本号有： <code>"2.0"</code> 。当 <code>version</code> 字段不存在时，默认当前配置文件为 <code>"1.0"</code> 版本。 |
| envId        | String               | 环境 ID，是环境的唯一标识。                                                                                         |
| region       | String               | 当前环境的地域信息，上海地域的环境可以不填，其他地域的环境则必须填写。                                                                     |
| functionRoot | String               | 函数代码存放的文件夹路径，相对于项目根目录的路径。                                                                               |
| functions    | Array<CloudFunction> | 函数配置项组成的数组，有关函数配置项的详细信息，请参见 <a href="#">函数配置项</a> 。                                                     |

### 简单参考配置

下面是一个简单的 CLI 配置文件包含的基本信息：

```
{
  "version": "2.0",
  "envId": "dev-xxxx",
  "functionRoot": "functions",
  "functions": [
    {
      "name": "app",
      "timeout": 5,
      "envVariables": {
        "key": "value"
      }
    }
  ]
}
```

# 环境

## 环境基础

最近更新时间：2022-02-18 16:48:40

### 查看所有环境

如需查看所拥有的所有环境的基本信息时，可以使用下面的命令获取环境列表：

```
tcb env list
```



您会得到类似下面的输出：

| 名称   | 环境 Id       | 套餐版本  | 来源  | 创建时间                | 环境状态 |
|------|-------------|-------|-----|---------------------|------|
| base | base-224c6b | 基础版 1 | 小程序 | 2019-04-09 13:06:09 | 正常   |
| cli  | cli         | 按量计费  | 云开发 | 2019-12-17 16:25:25 | 正常   |

环境 Id 是环境的唯一标志，在全局范围内唯一。环境名称用户自行定义，可以帮助用户方便的区分不同的环境。

Status 表明了环境的状态，只有当环境状态正常时，您才可以操作函数、数据库、存储等资源。新建的环境也可能处于不可用的状态，请耐心等待初始化完成。如果您的环境状态一直处于不可用状态时，可以提交 [工单](#) 将问题反馈给我们。

### 创建环境

您可以使用下面的命令创建一个新的环境：

```
tcb env create env-name
```



每个用户可以创建的环境数量有限，如果您的环境数量已经使用完，您会得到一个创建失败的提醒。

# 安全域名

最近更新时间：2022-02-18 16:52:19

当您需要在网页应用中使用云开发的身份验证服务时，需要将您的网站的域名（发起请求的页面的域名）加入安全域名名单中。安全域名是云开发服务认可的用户请求来源域名，所有来自非安全域名名单中的请求都不会被响应。

## ⚠ 注意

所有 CloudBase CLI 命令均在配置文件所在目录执行。

## 查看安全域名

使用下面的命令查看所有配置的安全域名：

```
tcb env domain list
```



您会得到类似下面的输出：

```
→ tcb env:domain:list
```

| 域名 Id | 域名   | 创建时间                | 状态  |
|-------|------|---------------------|-----|
|       |      | 2020-04-02 15:16:09 | 启用中 |
|       | .com | 2020-04-02 15:16:20 | 启用中 |

## 新增安全域名

您可以使用下面的命令添加新的安全域名：

```
tcb env domain create <domain>
```



您需要指定被添加的域名 domain，当需要添加多个域名时，需要以 / 分隔，如 `abc.com/def.com`。

### # 添加一个域名

```
tcb env domain create www.xxx.com
```

### # 添加多个域名

```
tcb env domain create www.domain1.com/www.domain2.com/www.domain3.com
```



## 删除安全域名

您可以使用下面的命令删除安全域名：

```
tcb env domain delete
```



回车后，CloudBase CLI 会拉取您配置的所有安全域名，您可以选择删除指定的域名。

# 登录方式

最近更新时间：2022-02-18 16:49:02

当您需要使用云开发的身份验证服务时，您可以配置需要使用的登录方式。目前云开发支持自定义登录、微信公众平台、微信开放平台登录等多种登录方式。

## 查看登录方式

您可以使用下面的命令列出环境配置的登录方式列表，查看环境配置的登录方式，以及相关的状态。

```
tcb env login list
```



您会得到类似于下图的输出：

| 平台   | 平台 Id     | 创建时间                | 状态 |
|------|-----------|---------------------|----|
| 匿名登录 | anonymous | 2020-02-03 14:58:21 | 启用 |

## 新建登录方式

您可以使用下面的命令新建登录方式：

```
tcb env login create
```



您需要选择配置的平台，登录方式状态，以及对应的 AppId 和 AppSecret，登录方式请选择启用。在添加方式时不会校验 AppId 和 AppSecret 的有效性，只有在请求时，AppId 和 AppSecret 才会被校验，所以请确保您添加的 AppId 和 AppSecret 是有效的。

## 修改登录方式

您也可以使用 `tcb env login update` 修改已经配置的登录方式，如切换启用状态，修改 AppId 和 AppSecret。

# 云函数

## 函数管理

最近更新时间：2022-02-18 16:45:03

云函数是一段运行在云端的代码，无需管理服务器，在开发工具内编写、一键上传部署即可运行后端代码。云开发中的云函数可让用户将自身的业务逻辑代码上传，并通过云开发的调用触发函数，从而实现后端的业务运作。

### ⚠ 注意

所有 CloudBase CLI 命令均在配置文件所在目录执行。

## 查看函数

您可以使用下面的命令列出所有云函数，查看函数的基本信息：

```
tcb fn list
```



您会得到类似下面的输出：

| 函数 ID            | 函数名称         | 运行时           | 创建时间                   | 修改时间                   | 状态       |
|------------------|--------------|---------------|------------------------|------------------------|----------|
| lam-<br>bsk77mry | node-<br>sdk | Nodejs<br>8.9 | 2020-03-24<br>11:42:21 | 2020-04-02<br>15:00:26 | 部署<br>完成 |
| lam-<br>4t96an8m | app          | Nodejs<br>8.9 | 2020-04-02<br>09:59:24 | 2020-04-02<br>11:00:50 | 部署<br>完成 |
| lam-<br>jxc3y8ys | liqi         | Nodejs<br>8.9 | 2020-03-28<br>21:17:13 | 2020-03-28<br>21:18:43 | 部署<br>完成 |
| lam-<br>3gbgxf56 | sum          | Nodejs<br>8.9 | 2020-03-24<br>11:41:16 | 2020-03-24<br>11:42:46 | 部署<br>完成 |

## 指定返回条数和偏移量

默认情况下，fn list 命令只会列出前 20 个函数，如果您的函数较多，需要列出其他的函数，您可以通过下面的选项指定命令返回的数据长度以及数据的偏移量：

```
-l, --limit <limit>  返回数据长度，默认值为 20  
-o, --offset <offset> 数据偏移量，默认值为 0
```

例如：

```
# 返回前 10 个函数的信息  
tcb fn list -l 10  
# 返回第 3 - 22 个函数的信息（包含 3 和 22）  
tcb fn list -l 20 -o 2
```

## 下载云函数代码

您可以通过下面的命令下载云函数代码：

```
tcb fn code download <functionName> [destPath]
```

默认情况下，函数代码会下载到 **functionRoot** 下，以函数名称作为存储文件夹，您可以指定函数存放的文件夹地址，函数的所有代码文件会直接下载到指定的文件夹中。

## 触发函数

您可以在本地通过 CloudBase CLI 直接触发您的云函数：

```
# 触发 app 函数  
tcb fn invoke app  
  
# 触发配置文件中的全部函数  
tcb fn invoke
```

## 查看函数详情

前面提到的 `fn list` 命令只能查看函数的简单信息，如果您想查看函数的详细信息，您可以使用下面的命令：

```
# 查看 app 函数的详情  
tcb fn detail app
```

输出：

```
状态：部署完成
```

```
代码大小 ( B ) : 1695
环境变量(key=value): key=value
函数名称: test
执行方法: index.main
内存配置(MB): 256
修改时间: 2019-08-19 21:15:39
环境 Id: dev-xxx
运行环境: Nodejs8.9
超时时间(S): 20
网络配置: 无
触发器:
myTrigger: {"cron": "0 0 2 1 * * *"}
函数代码 ( Java 函数以及入口大于 1 M 的函数不会显示 ) :
'use strict';
exports.main = async (event, context, callback) => {
  console.log(event);
  return 'hello world'
};
```

## 删除函数

您可以通过下面的命令删除函数：

```
# 删除 App 函数
tcb fn delete app
```

```
# 删除配置文件中的所有的函数
tcb fn delete
```

## 复制函数

您可以通过下面的命令快速复制一个已经存在的函数：

```
# 复制 App 函数为 App2 函数
tcb fn copy app app2
```

使用 `fn copy` 命令时需要指定原函数名称以及复制后新的函数名称。当前环境 Id 和目标环境环境 Id 是可选的，如果没有指定目标环境 Id，函数会被复制到当前环境中。



 说明

如果新函数的名称已经存在，则复制操作会被终止。如果您想要覆盖已经存在的函数，可以使用下面的命令。

## 覆盖同名函数

```
# 复制 App 函数为 App2 函数，如果 App2 函数存在，覆盖已经存在的 App2 函数
tcb fn copy app app2 --force
```

 注意：

复制函数操作不会复制函数的触发器。

# 函数配置项

最近更新时间：2022-02-18 16:46:47

在 CLI 配置文件中，functions 数组可以包含多个函数配置项，函数配置项包含了函数名称（name），函数运行配置（config），函数调用传入参数（params）等多项与函数相关的信息，影响着函数操作的行为表现。

## 函数配置项

一个简单的例子：

```
{
  // 关联环境 ID
  envId: "dev-xxxx",
  // 函数配置
  functions: [
    {
      // functions 文件夹下函数文件夹的名称，即函数名
      name: "app",
      // 超时时间，单位：秒 S
      timeout: 5,
      runtime: "Nodejs8.9",
      installDependency: true,
      handler: "index.main"
    }
  ]
}
```

### ⚠ 注意：

0.6.0 版本起，为了简化使用，我们对 `cloudbaserc.js` 配置文件 `functions` 选项中的 `config` 选项进行了扁平化处理，原有 `config` 选项中的所有配置项都可以直接写在 `functions` 选项中。

下面为目前所有支持的配置项：

| 配置项  | 是否必填 | 类型     | 描述                   |
|------|------|--------|----------------------|
| name | 是    | String | 云函数名称，即为函数部署到云开发后的名称 |
|      |      |        |                      |

|                   |   |                             |                                                |
|-------------------|---|-----------------------------|------------------------------------------------|
| params            | 否 | Object/JSONObject           | CLI 调用云函数时的函数入参                                |
| triggers          | 否 | Array<CloudFunctionTrigger> | 触发器配置                                          |
| handler           | 否 | String                      | 函数处理方法名称，名称格式支持“文件名称.函数名称”形式                   |
| ignore            | 否 | String/Array<String>        | 部署/更新云函数代码时的忽略文件，支持 glob 匹配规则                  |
| timeout           | 否 | Number                      | 函数超时时间（1 – 60S）                                |
| envVariables      | 否 | Object                      | 包含环境变量的键值对对象                                   |
| vpc               | 否 | VPC                         | 私有网络配置                                         |
| runtime           | 否 | String                      | 运行时环境配置，可选值：Nodejs8.9, Nodejs10.15 Php7, Java8 |
| memorySize        | 否 | Number                      | 函数内存，默认值为 256，可选 128、256、512、1024、2048         |
| installDependency | 否 | Boolean                     | 是否云端安装依赖，目前仅支持 Node.js                         |
| codeSecret        | 否 | String                      | 代码加密密钥，格式为 36 位大小写字母、数字                        |

#### ⚠ 注意：

- runtime 默认为 Nodejs10.15，使用 Node.js 运行时可不填，使用 Php 和 Java 则必填。
- 启用代码加密后，将无法在小程序 IDE、腾讯云控制台中查看云函数的代码和信息。

## CloudFunctionTrigger

| 名称 | 是否必 | 类型 | 描述 |
|----|-----|----|----|
|----|-----|----|----|

|        | 填 |        |                                   |
|--------|---|--------|-----------------------------------|
| name   | 是 | String | 触发器名称                             |
| type   | 是 | String | 触发器类型，可选值：timer                   |
| config | 是 | String | 触发器配置，在定时触发器下，config 格式为 cron 表达式 |

## VPC

| 名称       | 是否必填 | 类型     | 描述        |
|----------|------|--------|-----------|
| vpclId   | 是    | String | VPC ID    |
| subnetId | 是    | String | VPC 子网 ID |

## 更新函数运行时配置

创建函数时，Cloudbase CLI 会为函数提供一些默认的配置，所以不需要添加配置信息也可以直接部署函数。也可以通过下面的命令修改函数的运行时配置。

```
# 更新 app 函数的配置
tcb fn config update app
```

```
# 更新配置文件中所有函数的配置信息
tcb fn config update
```

### ⚠ 注意：

- 目前支持修改的函数配置包含超时时间 timeout、环境变量 envVariables、运行时 runtime，VPC 网络以及 installDependency 等选项。
- CloudBase CLI 会从配置文件中读取函数的配置信息并更新，CloudBase CLI 会更新配置文件中存在的函数的所有配置，暂不支持指定更新单个配置选项。

## 配置项参考

```
{
  // 关联环境 ID
  "envId": "dev-xxxx",
  // 函数配置
  "functions": [
    {
      // functions 文件夹下函数文件夹的名称，即函数名
      "name": "app",
      // 超时时间，单位：秒 S
      "timeout": 5,
      // 环境变量
      "envVariables": {
        "key": "value"
      },
      // 私有网络配置，如果不使用私有网络，可不配置
      "vpc": {
        // vpc id
        "vpcId": "vpc-xxx",
        // 子网 id
        "subnetId": "subnet-xxx"
      },
      // 运行时，目前可选运行包含：Nodejs8.9, Nodejs10.15, Php7, Java8
      // 此参数可以省略，默认为 Nodejs10.15
      "runtime": "Nodejs10.15",
      // 是否云端安装依赖，仅支持 Node.js 项目
      "installDependency": true,
      // 函数触发器，说明见文档：https://cloud.tencent.com/document/product/876/32314
      "triggers": [
        {
          // name: 触发器的名字
          "name": "myTrigger",
          // type: 触发器类型，目前仅支持 timer（即定时触发器）
          "type": "timer",
          // config: 触发器配置，在定时触发器下，config 格式为 cron 表达式
          "config": "0 0 2 1 * * *"
        }
      ],
    }
  ],
}
```

```
// 函数处理入口，Node.js 和 PHP 项目可以省略，默认值为 index.main
// 因 Java 的 handler 配置较为特殊，所以当运行时为 Java 时，handler 不能省略
// 如：package.Class::mainHandler
"handler": "index.main",
// fn invoke 本地触发云函数时的调用参数
"params": {},
// 部署/更新云函数时忽略的文件
"ignore": [
  // 忽略 markdown 文件
  "*.md",
  // 忽略 node_modules 文件夹
  "node_modules",
  "node_modules/**/*"
]
}
```

# 触发器

最近更新时间：2022-02-18 16:47:29

触发器是按照一定规则触发函数的模块的抽象，Cloudbase 云函数目前仅支持定时触发器。

如果云函数需要定时/定期执行，即定时触发，您可以使用云函数定时触发器。已配置定时触发器的云函数，会在相应时间点被自动触发，函数的返回结果不会返回给调用方。

示例代码：

```
{
  envId: "xxx",
  functions: [
    {
      // triggers 字段是触发器数组，目前仅支持一个触发器，即数组只能填写一个，不可添加多个
      triggers: [
        {
          // name: 触发器的名字，规则见下方说明
          name: "myTrigger",
          // type: 触发器类型，目前仅支持 timer（即定时触发器）
          type: "timer",
          // config: 触发器配置，在定时触发器下，config 格式为 cron 表达式，规则见下方说明
          config: "0 0 2 1 * * *"
        }
      ]
    }
  ]
}
```

## 创建函数触发器

您可以使用下面的命令创建一个触发器：

```
# 创建 app 函数配置的触发器
tcb fn trigger create app
```

Cloudbase CLI 会自动读取 `cloudbaserc.json` 文件中指定函数配置的定时触发器，并创建云函数触发器。如果配置文件中没有包含触发器配置，则会创建失败。

一个函数可以包含多个触发器，一个触发器包含了以下 3 个主要信息：name, type, config

```
{
  // name: 触发器的名字，规则见下方说明
  name: 'myTrigger',
  // type: 触发器类型，目前仅支持 timer（即定时触发器）
  type: 'timer',
  // config: 触发器配置，在定时触发器下，config 格式为 cron 表达式
  config: '0 0 2 1 * * *'
}
```

当没有指定函数名时，Cloudbase CLI 会创建 `cloudbaserc.json` 文件包含的所有函数的所有触发器。

## 删除函数触发器

您可以使用下面的命令删除函数的触发器：

```
# 删除 app 函数的名为 trigger 的触发器
tcb fn trigger delete app trigger
```

同样，当没有指定函数名时，Cloudbase CLI 会删除 `cloudbaserc.json` 文件包含的所有函数的所有触发器。当只指定了函数名时，Cloudbase CLI 会删除指定函数的所有触发器，当同时指定了函数名称和触发器名称时，Cloudbase CLI 只会删除指定的触发器。

```
# 删除 cloudbaserc.json 文件中所有函数的所有触发器
tcb fn trigger delete

# 删除函数 app 的所有触发器
tcb fn trigger delete app

# 删除函数 app 的触发器 trigger
tcb fn trigger delete app trigger
```

## 触发器规则

- 定时触发器名称（name）：最大支持 60 个字符，支持 `a-z`，`A-Z`，`0-9`，`-` 和 `_`。必须以字母开头，且一个函数下不支持同名的多个定时触发器。
- 定时触发器触发周期（config）：指定的函数触发时间。填写自定义标准的 Cron 表达式来决定何时触发函数。有关 Cron 表达式的更多信息，请参考以下内容。

Cron 表达式有七个必需字段，按空格分隔。其中，每个字段都有相应的取值范围：

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|



| 排序  | 字段 | 值                                                              | 通配符                  |
|-----|----|----------------------------------------------------------------|----------------------|
| 第一位 | 秒  | 0 – 59 的整数                                                     | <code>' - * /</code> |
| 第二位 | 分钟 | 0 – 59 的整数                                                     | <code>' - * /</code> |
| 第三位 | 小时 | 0 – 23 的整数                                                     | <code>' - * /</code> |
| 第四位 | 日  | 1 – 31 的整数（需要考虑月的天数）                                           | <code>' - * /</code> |
| 第五位 | 月  | 1 – 12 的整数或 JAN、FEB、MAR、APR、MAY、JUN、JUL、AUG、SEP、OCT、NOV 和 DEC  | <code>' - * /</code> |
| 第六位 | 星期 | 0 – 6 的整数或 MON、TUE、WED、THU、FRI、SAT 和 SUN，其中 0 指星期一，1 指星期二，以此类推 | <code>' - * /</code> |
| 第七位 | 年  | 1970 – 2099 的整数                                                | <code>' - * /</code> |

## 通配符

| 通配符                  | 含义                                                                       |
|----------------------|--------------------------------------------------------------------------|
| <code>,</code> （逗号）  | 代表取用逗号隔开的字符的并集。例如：在“小时”字段中 1, 2, 3 表示 1 点、2 点和 3 点                       |
| <code>-</code> （短横线） | 包含指定范围的所有值。例如：在“日”字段中，1 – 15 包含指定月份的 1 号到 15 号                           |
| <code>*</code> （星号）  | 表示所有值。在“小时”字段中， <code>*</code> 表示每个小时                                    |
| <code>/</code> （正斜杠） | 指定增量。在“分钟”字段中，输入 1/10 以指定从第一分钟开始的每隔十分钟重复。例如，第 11 分钟、第 21 分钟和第 31 分钟，以此类推 |

 注意

在 Cron 表达式中的“日”和“星期”字段同时指定值时，两者为“或”关系，即两者的条件均生效。

## 示例

下面列举一些 Cron 表达式和相关含义：

- `* / 5 * * * * *` 表示每 5 秒触发一次。
- `0 0 2 1 * * *` 表示在每月的 1 日的凌晨 2 点触发。
- `0 15 10 * * MON-FRI *` 表示在周一到周五每天上午 10:15 触发。
- `0 0 10,14,16 * * * *` 表示在每天上午 10 点，下午 2 点，下午 4 点触发。
- `0 * / 30 9-17 * * * *` 表示在每天上午 9 点到下午 5 点内每半小时触发。
- `0 0 12 * * WED *` 表示在每个星期三中午 12 点触发。

# 函数部署

最近更新时间：2022-02-18 16:46:55

## ⚠ 注意

`fn deploy` 命令部署函数的文件大小总计不能超过 50 M，否则可能会部署失败。

在一个包含 `cloudbaserc.js` 配置文件的项目下，您可以直接使用下面的命令部署云函数：

```
tcb fn deploy <functionName>
```

使用 `fn deploy` 时，`functionName` 选项是可以省略的，当 `functionName` 省略时，Cloudbase CLI 会部署配置文件中的全部函数：

```
# 部署配置文件中的全部函数
tcb fn deploy
```

## 全部参数

Usage: tcb fn deploy [options] [name]

### 部署云函数

#### Options:

- `-e, --envId <envId>` 环境 Id
- `--code-secret <codeSecret>` 传入此参数将保护代码，格式为 36 位大小写字母和数字
- `--force` 如果存在同名函数，上传后覆盖同名函数
- `--path <path>` 自动创建 HTTP 访问服务访问路径
- `--all` 部署配置文件中的包含的全部云函数
- `--dir <dir>` 指定云函数的文件夹路径
- `-h, --help` 查看命令帮助信息

## 代码加密

在部署云函数时，您可以通过指定 `--code-secret codeSecret` 对上传的代码进行加密，`codeSecret` 支持 36 位的大小写字母、数字组成：

```
tcb fn deploy app --code-secret 7sGLwMnhgEfKmkqg2dMjB6xWk2hCxsAgGR6w
```

**⚠ 注意:**

启用代码加密后，将无法在小程序 IDE、腾讯云控制台中查看云函数的代码和信息。

## 覆盖同名函数

部署函数时，很可能会遇到已经存在同名云函数的情况，此时 Cloudbase CLI 会终止部署，询问是否覆盖同名函数，您可以选择覆盖已有的云函数或者终止部署。

如果您确定要覆盖可能存在的同名云函数，您可以在命令后附加 `--force` 选项指定 Cloudbase CLI 覆盖已存在的云函数。

```
tcb fn deploy dev --force
```

**⚠ 注意**

覆盖函数时，也会覆盖函数的配置和触发器。

## 默认选项

Cloudbase CLI 为 Node.js 云函数提供了默认选项，您在部署 Node.js 云函数时可以不用指定云函数的配置，使用默认配置即可部署云函数。

云函数默认配置：

```
{
  // 超时时间 5S
  timeout: 5,
  // 运行时
  runtime: 'Nodejs8.9',
  // 自动安装依赖
  installDependency: true,
  // 处理入口
  handler: 'index.main',
  // 忽略 node_modules 目录
  ignore: ['node_modules', 'node_modules/**/*', '.git']
}
```



## deploy 命令做了什么？

`fn deploy` 会读取 `cloudbaserc.json` 文件中指定函数的配置，并完成以下几项工作：

1. 将函数打包成压缩文件，并上传函数代码。
2. 部署函数配置，包括超时时间、网络配置等。
3. 部署函数触发器。

# 本地运行

最近更新时间：2022-02-18 16:47:17

Cloudbase CLI v0.4.0+ 版本支持了本地运行云函数（当前仅支持 Node 云函数），您可以直接使用 `tcb fn run` 在本地运行云函数。

## 说明

`tcb fn run` 默认以 `index.main` 作为函数执行入口。

## 通过函数路径运行

您可以使用 `--path` 选项指定函数入口文件的路径，直接运行云函数。

```
tcb fn run --path ./index.js
```

# 省略 JS 文件名称，以 `index.js` 为入口执行

```
tcb fn run --path ./functions/sum
```

## 通过函数名运行

您可以使用 `--name` 选项指定需要运行的云函数（需要使用 `cloudbaserc` 配置文件），使用 `--name` 选项时，可以通过 `cloudbaserc` 配置文件执行函数执行入口，例如 `index.boot`。

```
tcb fn run --name test
```

# 代码更新

最近更新时间：2022-02-18 16:46:38

当您的函数代码发生改变时，您可以使用下面的命令更新您的云函数的代码：

```
# 更新 app 函数的代码
tcb fn code update app
```



`functions:code:update` 命令和 `functions:deploy` 命令的主要区别是：`functions:code:update` 命令只会更新函数的代码以及执行入口，不会修改函数的其他配置，而 `functions:deploy` 命令则会修改函数的代码、配置以及触发器等。

# 日志

最近更新时间：2022-02-18 16:47:05

您可以通过下面的命令打印云函数的运行日志，使用此命令时必须指定函数的名称：

```
# 查看 App 函数的调用日志
tcb fn log app
```

输出：

```
请求时间：2019-08-15 12:12:00
函数名称：app
计费时间(ms)：100
运行时间(ms)：0.38
调用次数：1
占用内存：35.730 MB
请求 Id：d451aac7-bf12-11e9-xxxx-525400697544
调用状态：成功
返回结果：{"Message":"","Time":"2019-08-15T04:12:00Z","TriggerName":"myTrigger","Type":"Timer"}
日志：
...

END RequestId: d451aac7-bf12-11e9-xxxx-525400697544
Report RequestId: d451aac7-bf12-11e9-xxxx-525400697544 Duration:0ms Memory:256MB
MaxMemoryUsed:35.730469MB
```

默认情况下，Cloudbase CLI 会打印最近的 20 条日志，您可以通过在命令后附加下面的可用选项指定返回日志的筛选条件：

```
-i, --reqId <reqId> 函数请求 Id
-o, --offset <offset> 数据的偏移量，Offset + Limit不能大于10000
-l, --limit <limit> 返回数据的长度，Offset + Limit不能大于10000
--order <order> 以升序还是降序的方式对日志进行排序，可选值 desc 和 asc
--orderBy <orderBy> 根据某个字段排序日志,支持以下字段：
function_name,duration, mem_usage, start_time
--startTime <startTime> 查询的具体日期，例如：2019-05-16 20:00:00，只能与
endtime 相差一天之内
```



--endTime <endTime> 查询的具体日期，例如：2019-05-16 20:59:59，只能与  
startTime 相差一天之内  
-e, --error 只返回错误类型的日志  
-s, --success 只返回正确类型的日志

例如： `tcb fn log app -l 2` 打印 `app` 函数的最新 2 条日志信息。

```
请求时间: 2019-08-15 17:04:43
函数名称: test-scf
计费时间(ms): 100
运行时间(ms): 0.44
调用次数: 1
占用内存: NaN MB
请求 Id: 68649b0f-af84-11e9-a803-525400e8849e
调用状态: 成功
返回结果: {"key":"test","userInfo":
{"appId":"wx9c4c30a432a38ebc","openId":"on01a6UeSuBLGTQpc_PAjS_RK_4o"}}
日志: ....
```



# 函数版本

最近更新时间：2022-02-18 16:44:39

云函数的版本包含了函数的代码及配置。在实际的开发过程中，可通过发布版本固定函数代码及配置内容，减少影响业务系统的问题因素。

## 说明

函数在创建后缺省具有一个最近版本/最新版本（\$LATEST），仅 \$LATEST 版本的配置和代码支持修改。发布时以 \$LATEST 版本的配置和代码作为基础进行发布，生成新版本。

## 发布新版本

您可以使用下面的命令发布新的函数版本：

```
tcb fn publish-version <name> [description] // name 函数名, description 版本描述
```



## 查看函数版本列表

您可以通过下面的命令查看函数版本：

```
tcb fn list-function-versions <name>
```



您会得到类似下面的输出：

| 版本       | 描述   | 创建时间                | 修改时间                | 状态   |
|----------|------|---------------------|---------------------|------|
| \$LATEST | –    | 2021-10-25 19:02:44 | 2020-12-11 11:14:37 | 部署完成 |
| 1        | test | 2021-10-20 14:50:18 | 2021-10-20 14:50:18 | 部署完成 |
| 2        | 111  | 2021-10-21 16:34:07 | 2021-10-21 16:34:07 | 部署完成 |
| 3        | 1222 | 2021-10-21 16:34:38 | 2021-10-21 16:34:38 | 部署完成 |

# 文件层

最近更新时间：2022-02-18 16:45:39

如果您的云函数拥有较多的依赖库或公共代码文件，您可以使用云函数中的层进行管理。使用层管理，您可以将依赖放在层中而不是部署包中，可确保部署包保持较小的体积。

## 说明

- 层中的文件将会添加到 `/opt` 目录中，此目录在函数执行期间可访问。
- 如果您的函数已绑定了多个层，这些层将按顺序合并到 `/opt` 目录中。如果同一个文件出现在多个层中，平台将会保留最大序号层里的文件。
- 如果您正在使用层版本被删除，与该层版本绑定的函数将会继续运行。

## 创建层

您可以使用下面的命令创建层，创建层时，需要指定层别名、环境 Id、以及层内容的路径。层的名称由层别名和环境 Id 组成，层名称为层的唯一标识。每个层支持多种云函数运行时，既可以与多种不同运行时的云函数进行绑定。CLI 创建的层默认支持 `Nodejs8.9`、`Nodejs10.15`、`Php7`、`Java8` 等 3 个运行时。

上传层内容为 ZIP 文件格式，使用 CLI 上传时，您可以指定 `file` 选项为文件夹或者 ZIP 文件路径，当路径为文件夹时，CLI 将会打包文件夹中的内容为 ZIP 文件。

每次创建成功后，会生成新的层版本，层版本从 1 开始，新的层版本为已有版本的最大值 + 1。文件层不能更新，仅能发布新的层或删除已有的层。文件层的任意一个版本均可以删除，删除后的版本无法再绑定到新的云函数。删除文件层及版本不影响已经绑定的云函数，在已经绑定的函数中仍然可以使用。

单个用户最多创建 20 个文件层，单个文件层版本号从 1 开始，最大可到 200。

### ❗ 说明

不建议层文件的大小超过 100M。

#### # 上传文件目录

```
tcb fn layer create <alias> -e envId --file ./content
```

#### # 上传 ZIP 文件

```
tcb fn layer create <alias> -e envId --file ./file.zip
```



## 云函数绑定层

云函数绑定层后，就可以在云函数中访问层内容。您可以使用下面的命令绑定层到云函数，`name` 参数为云函数的名称。

```
tcb fn layer bind <name> -e envId
```



#### ❗ 说明

每个函数最大可配置 5 个文件层（层名 + 版本唯一确定）。

## 云函数解绑层

您可以使用下面的命令解绑云函数绑定的层，`name` 参数为云函数的名称。

```
tcb fn layer unbind <name> -e envId
```



## 调整层的顺序

您可以使用下面的命令对云函数绑定的层进行重新排序，`name` 参数为云函数的名称。

```
tcb fn layer sort <name> -e envId
```



## 下载层的文件

您可以使用下面的命令下载层的文件，只能下载指定环境的层文件。

```
tcb fn layer download -e envId
```



## 删除层

您可以使用下面的命令删除层，只能删除指定环境中创建的层，当层的所有版本都被删除时，即意味着层也被删除，您无法再看到此层。

```
tcb fn layer delete -e envId
```



在完全删除文件层后，如果创建新的同名文件层，使用原有文件层最大版本 + 1 的版本号。

## 层展示

您可以使用下面命令获取层的相关信息。

```
# 展示所有环境中的层
```

```
tcb fn layer list
```



# 展示指定环境中的层

```
tcb fn layer list -e envId
```

# 展示函数绑定的层

```
tcb fn layer list -e envId --name <name>
```

# 展示层的版本信息

```
tcb fn layer list -e envId --layer <layer>
```

# 流量路由

最近更新时间：2022-02-18 16:45:49

云函数支持流量路由设置。通过该设置，您可便捷控制函数版本在实际使用场合或环境中的灰度上线或回滚流程，避免一次性上线可能带来的风险。

## 配置函数版本路由

您可以使用下面的命令配置函数版本路由：

```
tcb fn config-route <name> <version1> <traffic1> [version2] [traffic2]
```



### 说明

- 如果只指定 version1 及 traffic1，其余流量均分配至 \$LATEST 版本。
- 如果同时指定 version1 traffic1 及 version2 traffic2，全部流量分配至两版本上。

## 查看函数版本路由

您可以通过下面的命令查看函数版本路由配置：

```
tcb fn get-route <name>
```



您会得到类似下面的输出：

| 版本 | 流量比例 |
|----|------|
| 4  | 10   |
| 2  | 90   |

# 预置并发

最近更新时间：2022-02-18 16:45:56

预置并发支持并发实例按配置预先启动，同时云函数平台不会主动回收这些实例，会尽可能地保障有相应数量的可以处理请求的并发实例。

## 说明

您可通过此功能，为函数的指定版本设定预置并发额度。通过配置预置并发，可预先进行计算资源的准备，降低冷启动、运行环境初始化及业务代码初始化引起的耗时。

## 设置预置并发

您可以使用下面的命令设置函数指定版本的预置并发：

```
tcb fn set-provisioned-concurrency <name> <version> <concurrency> // name 函数名，  
version 函数版本，concurrency 预置并发数
```



## 查看预置并发

您可以通过下面的命令查看函数预置并发：

```
tcb fn get-provisioned-concurrency <name> [version]
```



您会得到类似下面的输出：

| 设置并发数 | 已完成并发数 | 预置任务状态 | 状态说明 | 版本号 |
|-------|--------|--------|------|-----|
| 10    | 7      | 进行中    | —    | 1   |

## 删除函数预置并发

您可以通过下面的命令删除函数预置并发：

```
tcb fn delete-provisioned-concurrency <name> <version>
```



# 云存储

最近更新时间：2022-02-18 16:47:40

云存储是云开发为用户提供的文件存储能力，用户可以通过云开发提供的 CLI 工具、SDK 对存储进行操作，如上传、下载文件。存储在云存储中的文件默认提供 CDN 加速访问，用户可以快速访问云存储中的文件。

## 路径说明

- localPath** 为本地文件或文件夹的路径，为 **目录/文件名** 的形式，如 `./index.js`、`static/css/index.css` 等。
- cloudPath** 为云存储文件或文件夹的相对根目录的路径，为 **目录/文件名** 的形式，如 `index.js`、`static/css/index.js` 等。

### ⚠ 注意

Windows 系统中 localPath 为本地路径形式，是系统可以识别的路径，通常使用 \ 分隔符。  
**cloudPath** 是云端文件路径，均需要使用 / 分隔符。

## 上传文件

使用下面的命令上传文件/文件夹，当 CLI 检测到 localPath 为文件夹时，会自动上传文件内的所有文件。

```
tcb storage upload localPath cloudPath
```



## 下载文件

使用下面的命令下载文件/文件夹，需要下载文件夹时，需要指定 `--dir` 参数。

### # 下载文件

```
tcb storage download cloudPath localPath
```

### # 下载文件夹

```
tcb storage download cloudPath localPath --dir
```



## 下载全部文件

当指定 cloudPath 为 `/` 时，即代表下载云存储中的全部文件。

### # 下载文件夹

```
tcb storage download / localPath --dir
```





## 删除文件

使用下面的命令删除云端文件/文件夹，需要删除文件夹时，需要指定 `--dir` 参数。

```
# 删除文件
tcb storage delete cloudPath

# 删除文件夹
tcb storage delete cloudPath --dir
```



## 删除全部文件

云端路径为空时，即代表删除云存储中的全部文件。

```
tcb storage delete
```



## 列出文件列表

使用下面的命令列出文件夹下的文件：

```
tcb storage list cloudPath
```



## 获取文件访问链接

使用下面的命令获取文件的临时访问链接：

```
tcb storage url cloudPath
```



## 获取文件简单信息

使用下面的命令获取文件的简单信息：

```
tcb storage detail cloudPath
```



## 获取文件访问权限

使用下面的命令获取文件的访问权限设置信息：

```
tcb storage get-acl
```



## 设置文件访问权限

使用下面的命令设置文件的访问权限：

```
tcb storage set-acl
```



# 静态网站托管

最近更新时间：2022-02-18 16:49:31

云开发为开发者提供静态网页托管的能力，静态资源（HTML、CSS、JavaScript、字体等）的分发由对象存储 COS 和拥有多个边缘网点的 CDN 提供支持。您可在腾讯云控制台进行静态网站的部署，提供给您的用户访问。目前云开发静态网页托管能力仅在腾讯云云开发控制台支持，小程序 IDE 侧控制台暂不支持。

仅有付费方式为按量付费的环境可开通静态网页托管能力，预付费方式环境不可开通。

云开发的静态网页托管能力，目前在开通环境时，静态网页托管服务不会直接开通，而是需要您在环境中单独开通。

## ❗ 说明

使用 CLI 操作静态网站服务前请先到 [云开发控制台](#) 开通静态网站服务。

## 全量部署

云开发 CLI 提供了直接部署网站文件的命令，在您需要部署的文件夹目录下，直接运行 `tcb hosting deploy` 命令即可将当前目录下所有的文件部署静态网站中。

示例：

```
# docs 构建目录
cd docs
# 部署全部文件
tcb hosting deploy -e envId
```



## 部署文件

使用下面的命令将文件上传到静态网站的存储空间中的指定路径，当不指定 `cloudPath` 时，文件将上传到根目录。

```
# 部署文件到指定目录
tcb hosting deploy localPath cloudPath -e envId
```



```
# 将 hosting 目录下的所有文件部署到根目录
tcb hosting deploy hosting -e envId

# 将 static 目录下的 index.js 文件部署到 static/index.js
tcb hosting deploy ./static/index.js static/index.js -e envId
```



 注意

- 当您使用 vue.history 模式时，需要在 [静态网站](#) 设置栏中配置错误页面为 app 依赖的页面。
- 部署文件单个文件理论最大值为 50TB，文件数量无限制，具体以实际情况为准。

## 服务信息

使用下面的命令展示静态网站的状态，访问域名等信息：

```
tcb hosting detail -e envId
```



## 删除文件

使用下面的命令删除静态网站的存储空间中的文件或文件夹：

```
tcb hosting delete cloudPath -e envId
```



## 删除全部文件

云端路径为空时，表示删除全部文件。

```
tcb hosting delete -e envId
```



## 查看文件列表

使用下面的命令部署展示静态网站存储空间中文件：

```
tcb hosting list -e envId
```



## 路径说明

- **localPath** 为本地文件或文件夹的路径，为 **目录/文件名** 的形式，如 `./index.js` 、`static/css/index.css` 等。
- **cloudPath** 为云存储文件或文件夹的相对根目录的路径，为 **目录/文件名** 的形式，如 `index.js` 、`static/css/index.js` 等。

 注意

Windows 系统中 localPath 为本地路径形式，是系统可以识别的路径，通常使用 \ 分隔符。

`cloudPath` 是云端文件路径，均需要使用 / 分隔符。

# HTTP 访问服务

最近更新时间：2022-02-18 16:44:14

HTTP 访问服务支持通过 HTTP 链接访问云开发资源。可以通过 CLI 工具、控制台管理访问域名和云函数访问路径。

## 创建

您可以使用下面的命令，通过 CLI 命令行交互的方式创建 HTTP 访问服务地址。

```
tcb service create -e envId
```



您也可以指定指定云函数名称 `functionName` 和触发路径 `servicePath` 创建 HTTP 访问服务地址。

```
tcb service create -p servicePath -f functionName
```



## 删除 HTTP 访问服务链接

您可以使用下面的命令，通过 CLI 命令行交互的方式删除云函数 HTTP 访问服务链接。

```
tcb service delete -e envId
```



您也可以通过参数指定需要删除的 HTTP 访问服务信息，如 HTTP 访问服务地址的路径，或者 HTTP 访问服务 Id

```
# 指定 HTTP 访问服务绑定的路径  
tcb service delete -p servicePath
```



```
# 指定 HTTP 访问服务 Id  
tcb service delete -i serviceId
```

## 查询 HTTP 访问服务信息

您可以通过下面的命令列出所有的 HTTP 访问服务链接，查看它们的基本信息：

```
tcb service list
```



您会得到类似下面的输出：

| ID                                   | 路径    | 函数名      | 创建时间                |
|--------------------------------------|-------|----------|---------------------|
| 39fe3034-86d8-4333-a5ee-541a10xxxxxx | /api  | node-sdk | 2020-03-17 13:08:17 |
| Cc8c47d1-a6e6-4654-b184-69d3bcxxxxxx | /list | list     | 2020-03-13 11:11:24 |

### 设置更多的查询约束条件

```
-d domain    指定域名
-p servicePath 指定 HTTP 访问服务路径
-i serviceId  指定 HTTP 访问服务 Id
```



### 绑定 HTTP 访问服务自定义域名

您可以通过下面的命令绑定 HTTP 访问服务域名：

```
tcb service domain bind domain
```



#### ⚠ 注意：

绑定自定义域名之前，请先设置您的域名的 CNAME 记录值为 **默认域名**，CNAME 记录不存在时会导致域名绑定失败。

### 解绑 HTTP 访问服务自定义域名

您可以通过下面的命令解绑 HTTP 访问服务域名：

```
tcb service domain unbind domain
```



### 查询 HTTP 访问服务自定义域名

您可以通过下面的命令列出所有的 HTTP 访问服务域名，查看它们的基本信息：

```
tcb service domain list
```



您会得到类似下面的输出：

| 域名            | 状态   | 创建时间                |
|---------------|------|---------------------|
| www.xxxxx.com | 部署成功 | 2019-12-17 10:41:45 |

### 设置更多的查询约束条件

-d domain    指定域名





# 从 v0 升级到 v1

最近更新时间：2022-02-18 16:46:10

CloudBase CLI v1 版本已经规划了许久，在 v1 版本中，我们完全改变了命令的使用方法，简化了部分命令的长度，并向更为标准的规范对齐，这也是后期我们努力的发展发向：为用户提供更简单、更强大的 CLI 工具。

同时，CloudBase CLI v1 版本完全兼容了 CloudBase v.0 中的命令用法，您不用担心升级所可能带来的问题，这不是一次破坏性的、不兼容的升级。尽管如此，我们任建议您使用新的命令，这是未来的发展方向。

## 升级说明

### tcb 命令

在 v1 版本中，我们引入了 `tcb` 命令，可以一键创建、部署云开发项目。

### 命令格式变更

在 v1 版本中，所有的命令由 `env:list` 格式变成了 `env list` 格式，即命令中所有的 `:` 符号修改为空格使用。这有助于我们合并同类型的命令，提供更简单易懂的命令 help 提示。同时也符合大部分的命令使用习惯，降低学习成本。

### 命令名变更

- 原 `functions` 命令改为了简短的 `fn` 命令，如 `tcb fn list`。
- 原云接入 `service` 命令改为了 `access` 命令。

### 移除编程式 API

在 v1 版本中，我们彻底移除了 CLI 的编程式使用，如果您需要以管理员身份操作云开发的资源，请使用 `@cloudbase/manager-node` SDK。