

云开发 CloudBase

CLI 使用指南

产品文档



腾讯云

【 版权声明 】

©2013–2020 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

CLI 使用指南

概述

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

文件存储

静态网站

常见问题

CLI 使用指南

概述

最近更新时间：2020-10-18 14:10:45

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

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+。

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
```

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：

```
cloudbase -v
```

如果看到输出版本号，说明已经安装成功。

4. 再多了解1个命令

为了简化输入，CloudBase 命令可以简写成 tcb。tcb 是云开发产品英文的简称：Tencent Cloud Base。尝试到命令行终端输入：

```
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
```

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

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，请妥善保存和定期更换密钥，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
```

创建项目与部署

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。

```
tcb init
```

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 cli 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
```

```
| └─ app
| └─ index.js
└─ cloudbaserc.json // 项目配置文件
```

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);
};
```

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），参考 [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": "app",
        "timeout": 5,
        "envVariables": {},
```

```
"runtime": "Nodejs8.9",  
"handler": "index.main"  
}  
]  
}
```

4. 部署函数

最后，在项目根目录下（`cloudbaserc.json` 所在目录）运行 `cloudbase functions:deploy` 命令，即可部署 App 函数：

```
tcb functions:deploy app
```

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

```
tcb functions: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 `cloudbase -h` 查看所有可用命令：

```
tcb -h
```

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

Options:

```
--config-file <path> 设置配置文件，默认为 ./cloudbaserc.js 或 .cloudbaserc.json  
-V, --version 输出当前 CloudBase CLI 版本  
-h, --help 输出帮助信息
```

Commands:

```
init [options] 创建并初始化一个新的项目
```

```
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> 创建新的云开发环境
env:rename <name> [envId] 重命名云开发环境
env:login:list [envId] 列出环境登录配置
env:login:create [envId] 创建环境登录配置
env:login:update [envId] 更新环境登录方式配置
env:domain:list [envId] 列出环境的安全域名列表
env:domain:create <domain> [envId] 添加环境安全域名, 多个以斜杠 / 分隔
env:domain:delete [envId] 删除环境的安全域名
functions:list [options] [envId] 展示云函数列表
functions:download [options] <functionName> [dest] [envId] 下载云函数代码
functions:deploy [options] [functionName] [envId] 部署云函数
functions:delete [functionName] [envId] 删除云函数
functions:detail [options] [functionName] [envId] 获取云函数信息
functions:code:update [options] <functionName> [envId] 更新云函数代码
functions:config:update [functionName] [envId] 更新云函数配置
functions:copy [options] <functionName> <newFunctionName> [envId] [targetEnvId] 拷贝云函数
functions:log [options] <functionName> [envId] 打印云函数日志
functions:trigger:create [functionName] [envId] 创建云函数触发器
functions:trigger:delete [functionName] [triggerName] [envId] 删除云函数触发器
functions:invoke [functionName] [params] [envId] 触发云端部署的云函数
functions:run [options] 本地运行云函数 (当前仅支持 Node)
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] 获取文件存储的文件列表
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
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
```

```
server:deploy [name] 部署 node 服务
server:log [options] <name> 查看日志
server:reload <name> 重启 node 服务
server:stop <name> 停止应用
server:show 查看状态
open [link] 在浏览器中打开云开发相关连接
```

示例

- 登录

```
cloudbase login
```

- 初始化云开发项目

```
cloudbase init
```

- 部署云函数

```
cloudbase functions:deploy
```

- 查看命令使用介绍

```
cloudbase functions:log -h
```

配置文件

最近更新时间：2020-11-02 18:30:57

配置文件可以简化 CloudBase CLI 使用，方便项目开发，当使用命令参数缺省时，CloudBase CLI 会从配置文件中解析相关参数并使用，方便开发者以更简单的方式使用 CloudBase CLI。

默认情况下，使用 `cloudbase init` 初始化项目时，会生成 `cloudbaserc.js` 或 `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。

字段

字段说明如下表所示：

字段	类型	说明
envId	String	环境 ID，是环境的唯一标识。
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，请参考 函数配置项 文档。

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：

```
{
  // 关联环境 ID
  envId: "dev-xxxx",
  // 云函数文件夹名称，相对路径，可以省略 './'
  functionRoot: "functions",
  // 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，在 Node 云函数中可以通过 process.env.key 访问
      envVariables: {
        key: "value"
      }
    }
  ]
}
```

```
}  
]  
}
```

环境

环境基础

最近更新时间：2021-01-08 09:30:21

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：

```
cloudbase 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-554:ali	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：

```
cloudbase 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：2020-11-02 18:08:12

⚠ 注意：

所有 CloudBase CLI 命令均在配置文件所在目录执行。

当您需要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。

查看安全域名

使用下面的命令查看所有配置的安全域名：

```
cloudbase 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
5c4dd2c2-95b2-427e-a15f-d4ccff121832	www.ink.com	2020-04-02 15:16:09	启用中
a697f6e6-02a1-4567-87e1-6c2960715a92	coda.ink.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：

```
cloudbase env:domain:create <domian>
```

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
```

```
cloudbase env:domain:create www.xxx.com
```

添加多个域名

```
cloudbase 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：

```
cloudbase 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：2020-10-09 16:12:49

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。

```
cloudbase 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：

```
cloudbase 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 `cloudbase 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：2020-12-20 22:18:08

⚠ 注意：

所有 CloudBase CLI 命令均在配置文件所在目录执行。

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。

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：

```
cloudbase functions: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-bsk77mry	node-sdk	Nodejs8.9	2020-03-24 11:42:21	2020-04-02 15:00:26	部署完成
lam-4t96an8m	app	Nodejs8.9	2020-04-02 09:59:24	2020-04-02 11:00:50	部署完成
lam-jxc3y8ys	liqi	Nodejs8.9	2020-03-28 21:17:13	2020-03-28 21:18:43	部署完成
lam-3gbgxf56	sum	Nodejs8.9	2020-03-24 11:41:46	2020-03-24 11:42:16	部署完成

指定返回条数和偏移量

默认情况下，functions: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
```

如：

```
# 返回前 10 个函数的信息
cloudbase functions:list -l 10
# 返回第 3 - 22 个函数的信息（包含 3 和 22）
cloudbase functions: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：

```
cloudbase functions: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
cloudbase functions:invoke app
# 触发配置文件中的全部函数
cloudbase functions:invoke
```

查看函数详情

前面提到的 `functions: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
cloudbase functions:detail app
# 查看配置文件中的所有函数的详情
cloudbase functions:detail
```

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
cloudbase functions:delete app
# 删除配置文件中的所有的函数
cloudbase functions: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
functions:copy app app2
```

使用 `functions: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。

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
functions: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：2020-12-19 20:00:12

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
-----	------	----	----

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, Php7, Java8
installDependency	否	Boolean	是否云端安装依赖，目前仅支持 Node.js

CloudFunctionConfig

名称	是否必填	类型	描述
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, Php7, Java8
installDependency	否	Boolean	是否云端安装依赖，目前仅支持 Node.js

⚠ 注意：

runtime 默认为 Nodejs8.9，使用 Node 运行时可不填，使用 Php 和 Java 则必填。

CloudFunctionTrigger

名称	是否必填	类型	描述
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
vpcId	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
cloudbase functions:config:update app
# 更新配置文件中所有函数的配置信息
cloudbase functions:config:update
```

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。

触发器

最近更新时间：2020-11-02 18:07:43

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
cloudbase functions: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
cloudbase functions: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
cloudbase functions:trigger:delete

# 删除函数 app 的所有触发器
cloudbase functions:trigger:delete app

# 删除函数 app 的触发器 trigger
cloudbase functions: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	, - * /
第二位	分钟	0 – 59 的整数	, - * /
第三位	小时	0 – 23 的整数	, - * /
第四位	日	1 – 31 的整数（需要考虑月的天数）	, - * /
第五位	月	1 – 12 的整数或 JAN、FEB、MAR、APR、MAY、JUN、JUL、AUG、SEP、OCT、NOV 和 DEC	, - * /
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	, - * /
第七位	年	1970 – 2099 的整数	, - * /

通配符

通配符	含义
,（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
-（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
（星号）	表示所有值。在“小时”字段中， 表示每个小时

通配符	含义
/ (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：2020-12-19 19:41:16

⚠ 注意：

`functions: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：

```
cloudbase functions:deploy <functionName>
```

使用 `functions: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
cloudbase functions:deploy
```

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。

```
cloudbase functions: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？

functions: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：2020-11-02 17:54:27

本地运行

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 `cloudbase functions:run` 在本地运行云函数。

`cloudbase functions: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

```
cloudbase functions:run --path ./index.js

# 省略 JS 文件名称，以 index.js 为入口执行
cloudbase functions: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，如 `index.boot`。

```
cloudbase functions:run --name test
```

代码更新

最近更新时间：2020-11-02 18:07:13

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
cloudbase functions: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：2020-12-19 19:59:14

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
cloudbase functions: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 Max
MemoryUsed: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

如：cloudbase functions: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：....

文件存储

最近更新时间：2020-11-02 18:13:08

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。

```
cloudbase 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
cloudbase storage:download cloudPath localPath

# 下载文件夹
cloudbase storage:download cloudPath 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
cloudbase storage:delete cloudPath

# 删除文件夹
cloudbase storage:delete cloudPath --dir
```

列出文件列表

使用下面的命令列出文件夹下的文件：

```
cloudbase 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：

```
cloudbase 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：

```
cloudbase 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：

```
cloudbase 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：

```
cloudbase storage:set-acl
```

静态网站

最近更新时间：2020-12-19 21:51:50

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，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。

⚠ 注意：

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 `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
$ cloudbase 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
cloudbase hosting:deploy localPath cloudPath -e envId
```

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

```
# 将 static 目录下的 index.js 文件部署到 static/index.js
cloudbase 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：

```
cloudbase 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：

```
cloudbase hosting:delete cloudPath -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：

```
cloudbase 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。

常见问题

最近更新时间：2020-12-21 19:00:21

无法登录，CLI 工具没有响应怎么处理？

请检查您的终端能否访问网络。

函数部署失败的常见原因？

package.json 中依赖不存在。

云开发 CLI 工具是否收费？

CLI 工具本身是免费的，如果涉及到使用云资源，可以了解云开发产品相关 [购买指南](#)。

云开发 CLI 工具必须先安装 Node.js ？

是的，安装 Node.js 的时候，NPM 就会自动安装好，因此可以通过 NPM 安装 CLI。

云开发 CLI 工具可以在 Windows 上使用吗？

目前支持 MacOS 和 Windows 平台，可以放心使用。