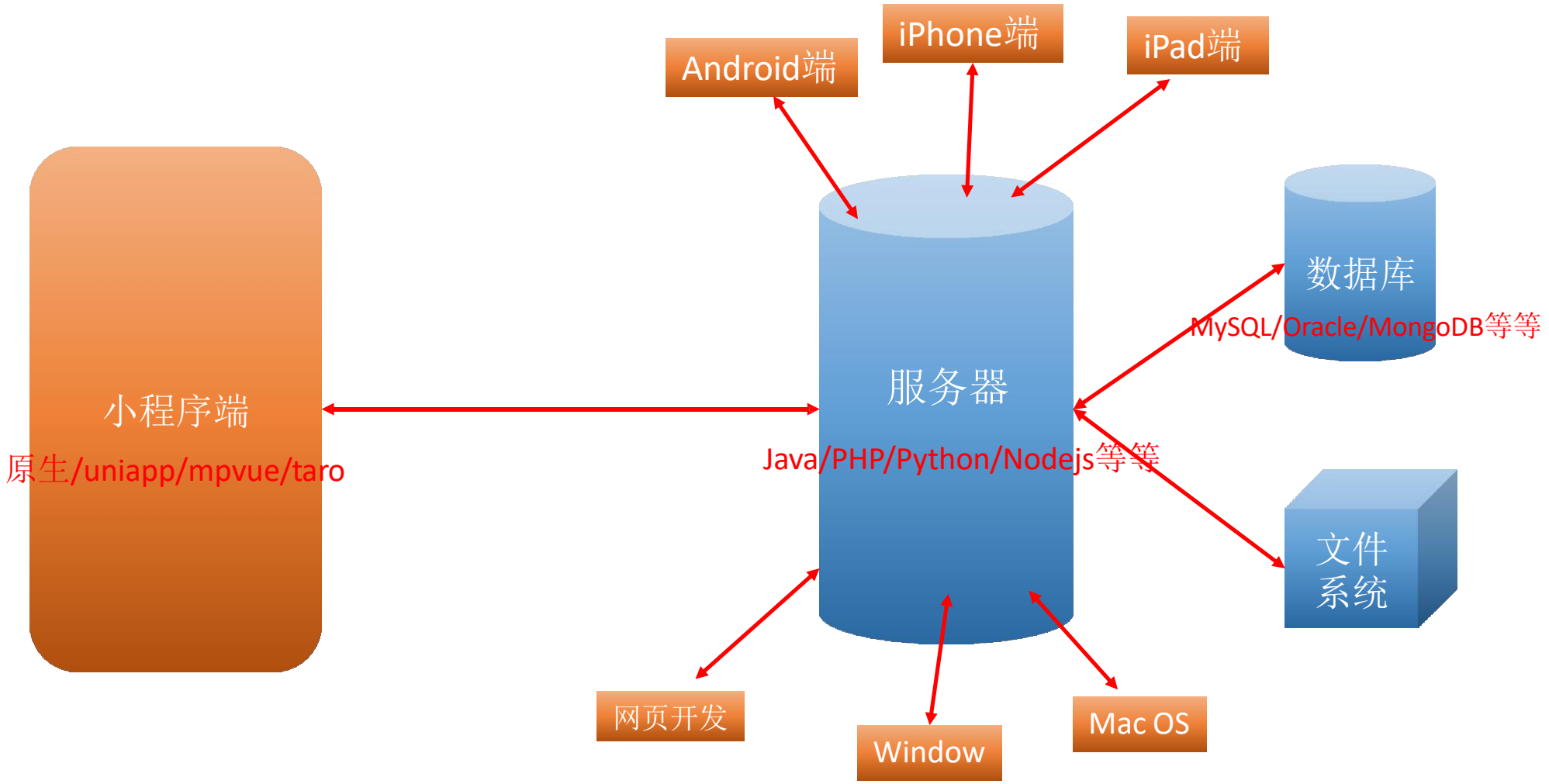
小程序 - 云开发详解

王红元 coderwhy

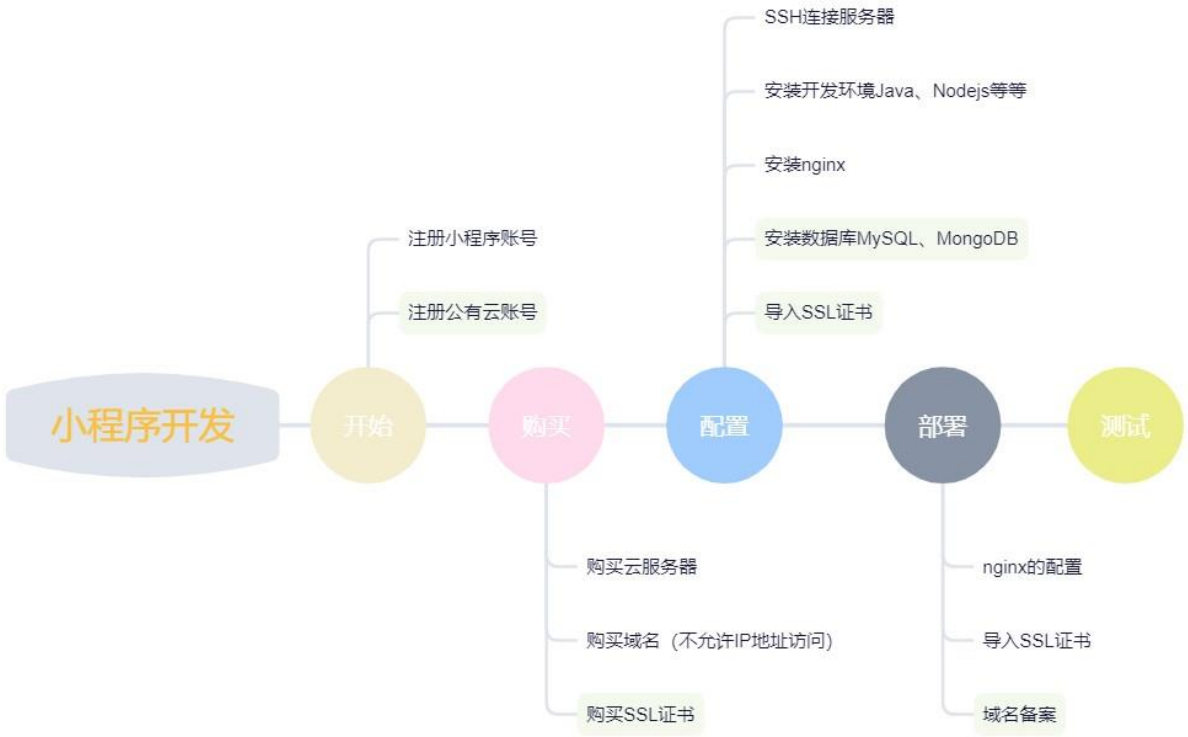
目录
content

完整的小程序项目


Android端
iPhone端
iPad端
小程序端
原生/uniapp/mpvue/taro
服务器
Java/PHP/Python/Nodejs等等
数据库
MySQL/Oracle/MongoDB等等
文件系统
网页开发
Window
Mac OS

开发成本的考虑

- **但是，如果一个 小公司或个人 只是想开发一个小程序推广自己的产品或者实现某个想法了?**
- **按照传统的开发模式，我们需要考虑哪些东西呢?**
  - 成本角度：维护服务器成本，并且需要考虑并发量大后服务器的扩展。
  - 技术研发：对于单纯会前端的人来说，学习后端相关的技术，成本较高。

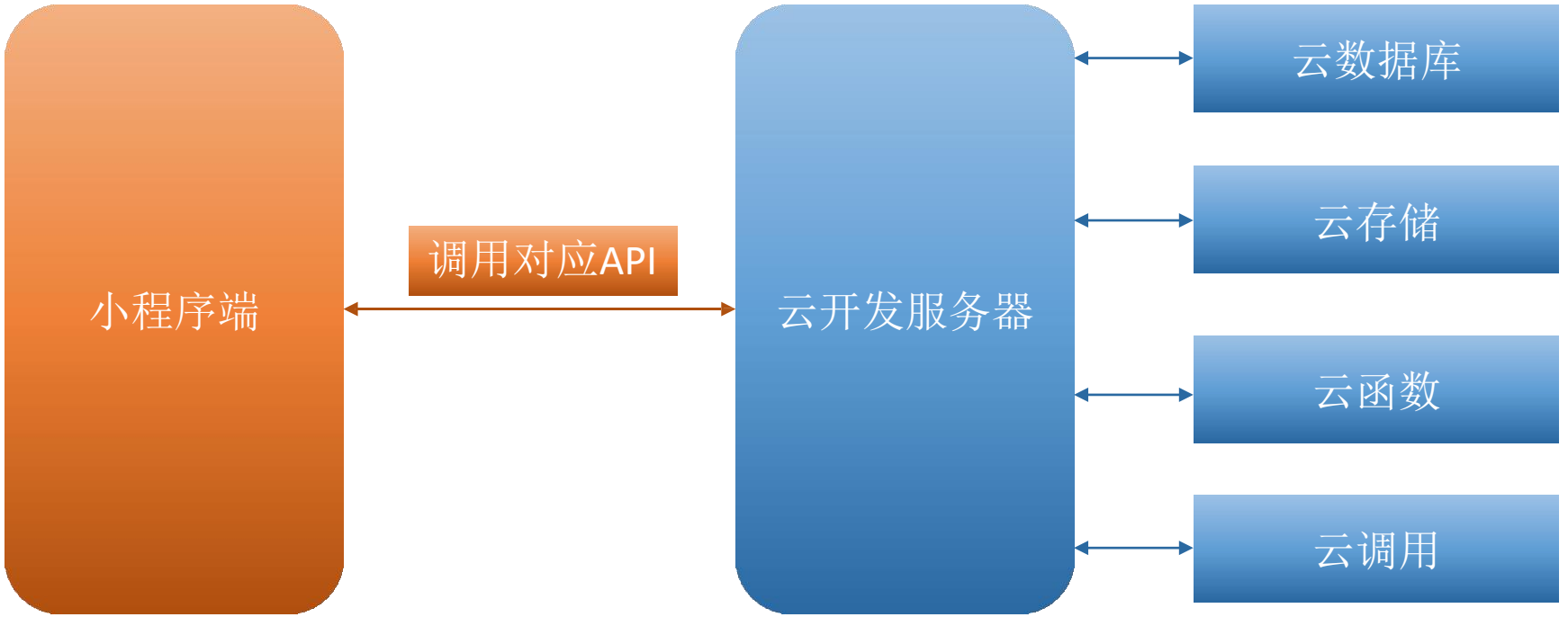


新的开发模式

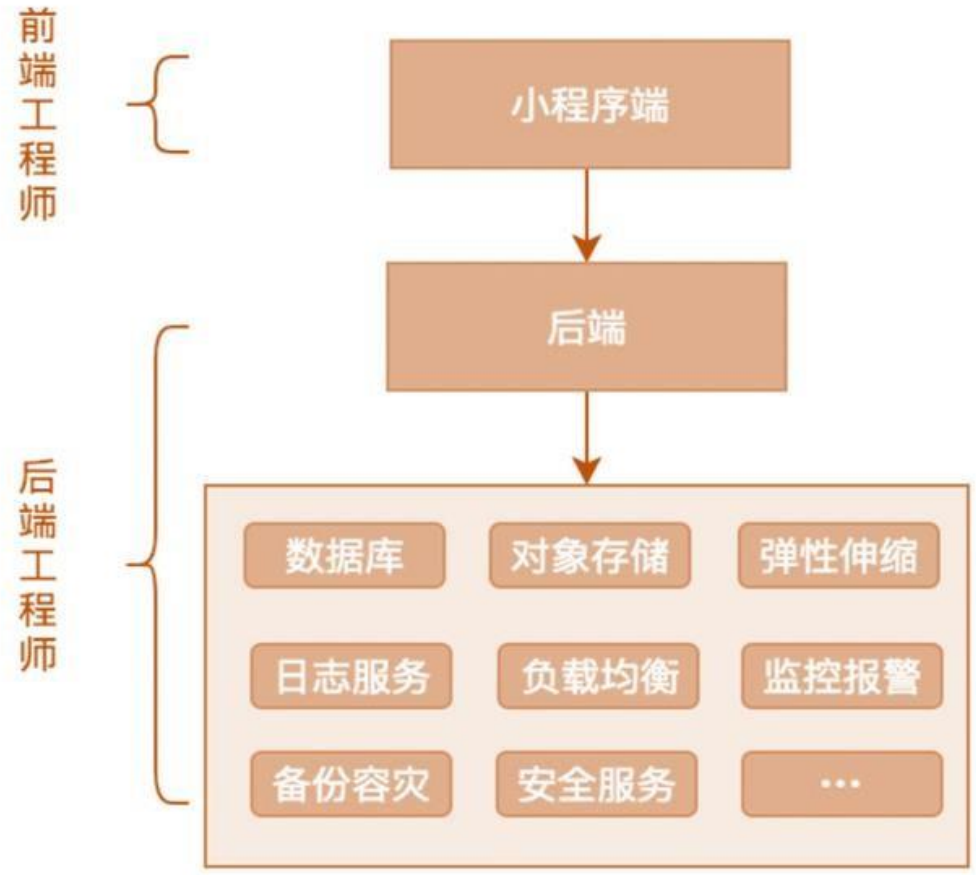
是否有一种新的开发模式

让开发者可以更多地专注业务逻辑

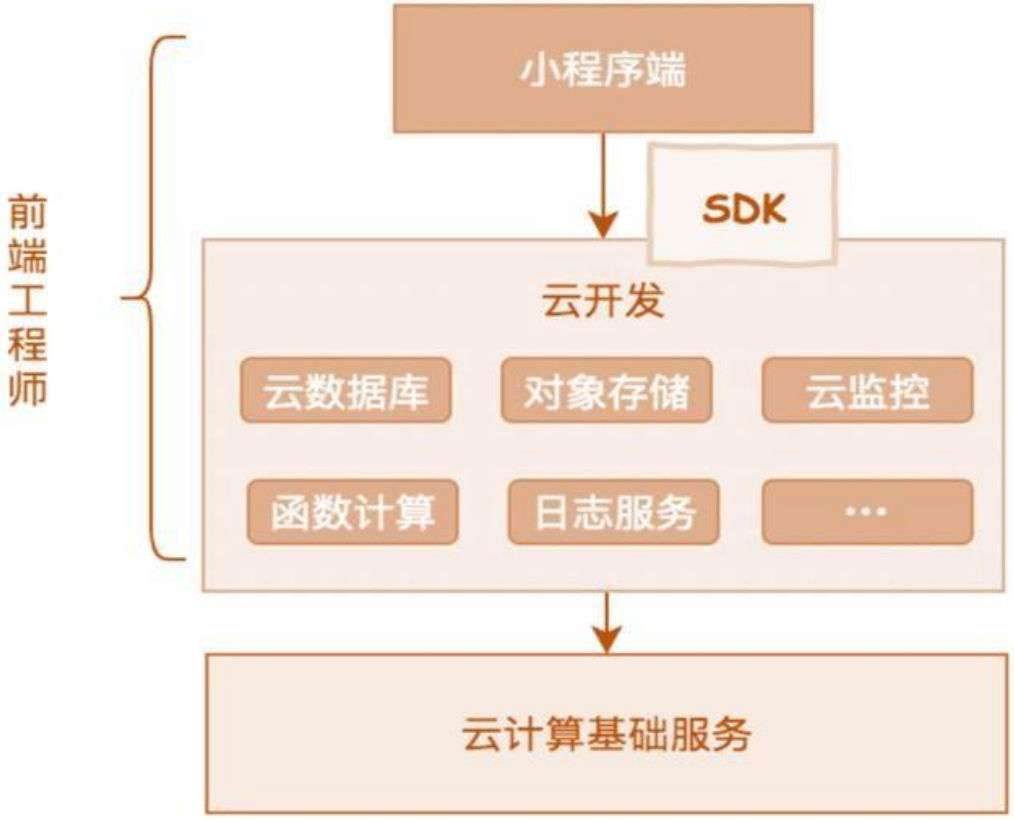
云开发的模式



云开发模式和传统模式对比

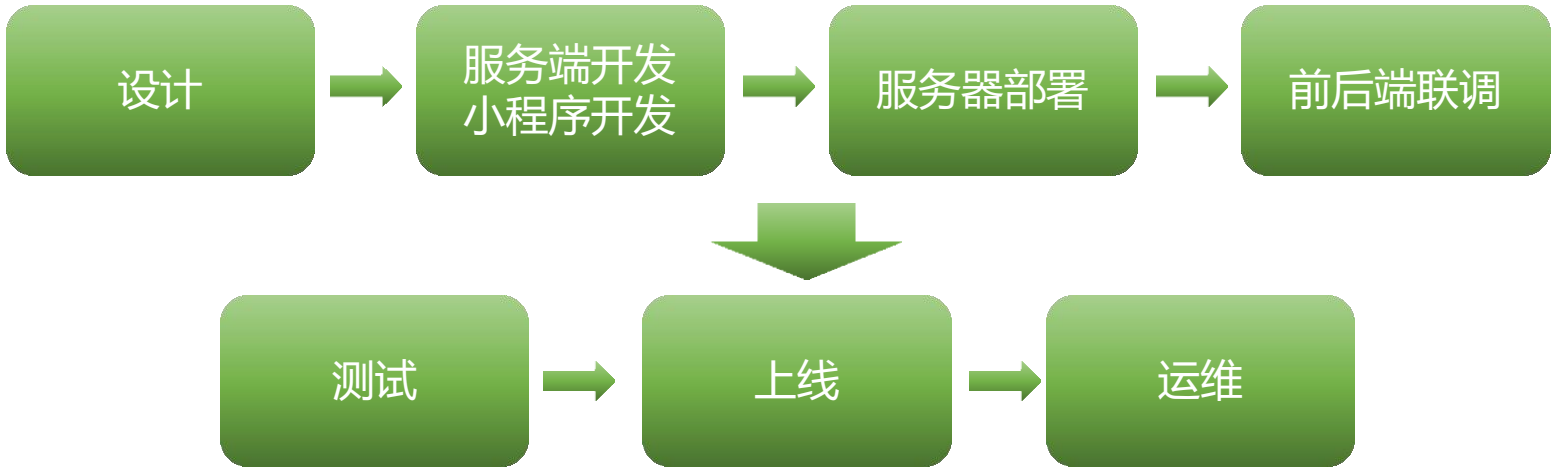


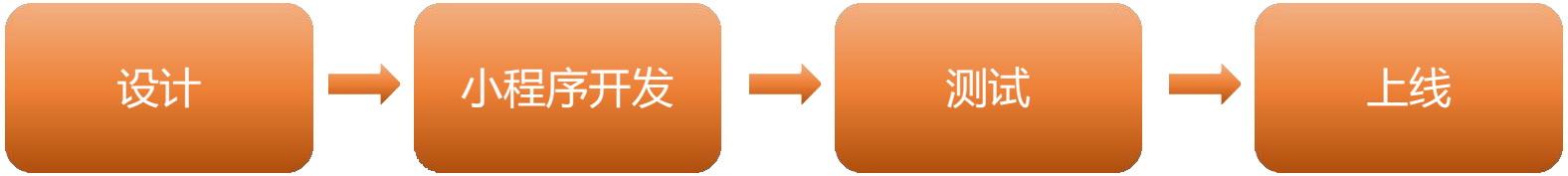
传统小程序开发



基于 Serverless 的小程序开发

项目流程对比

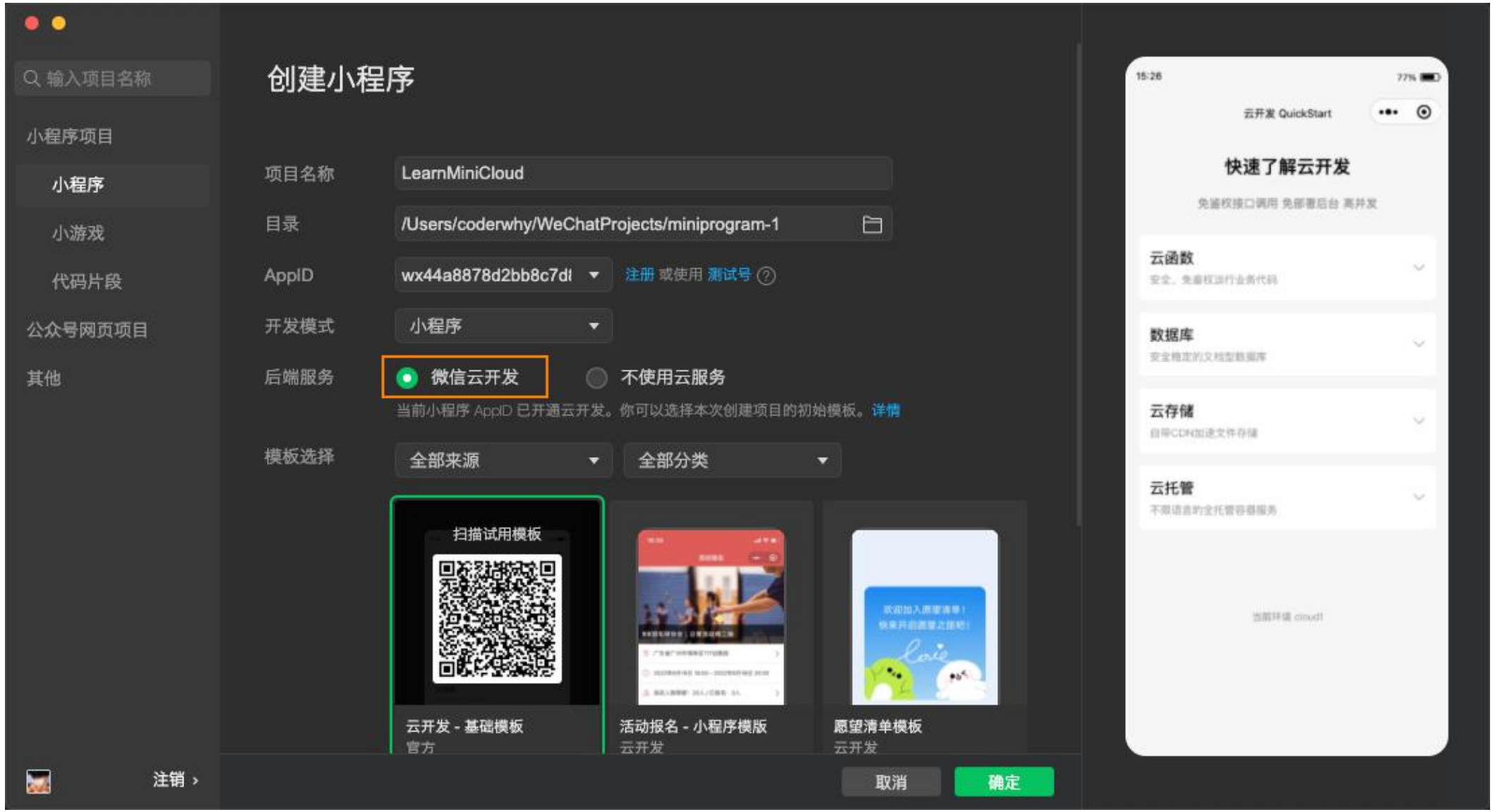




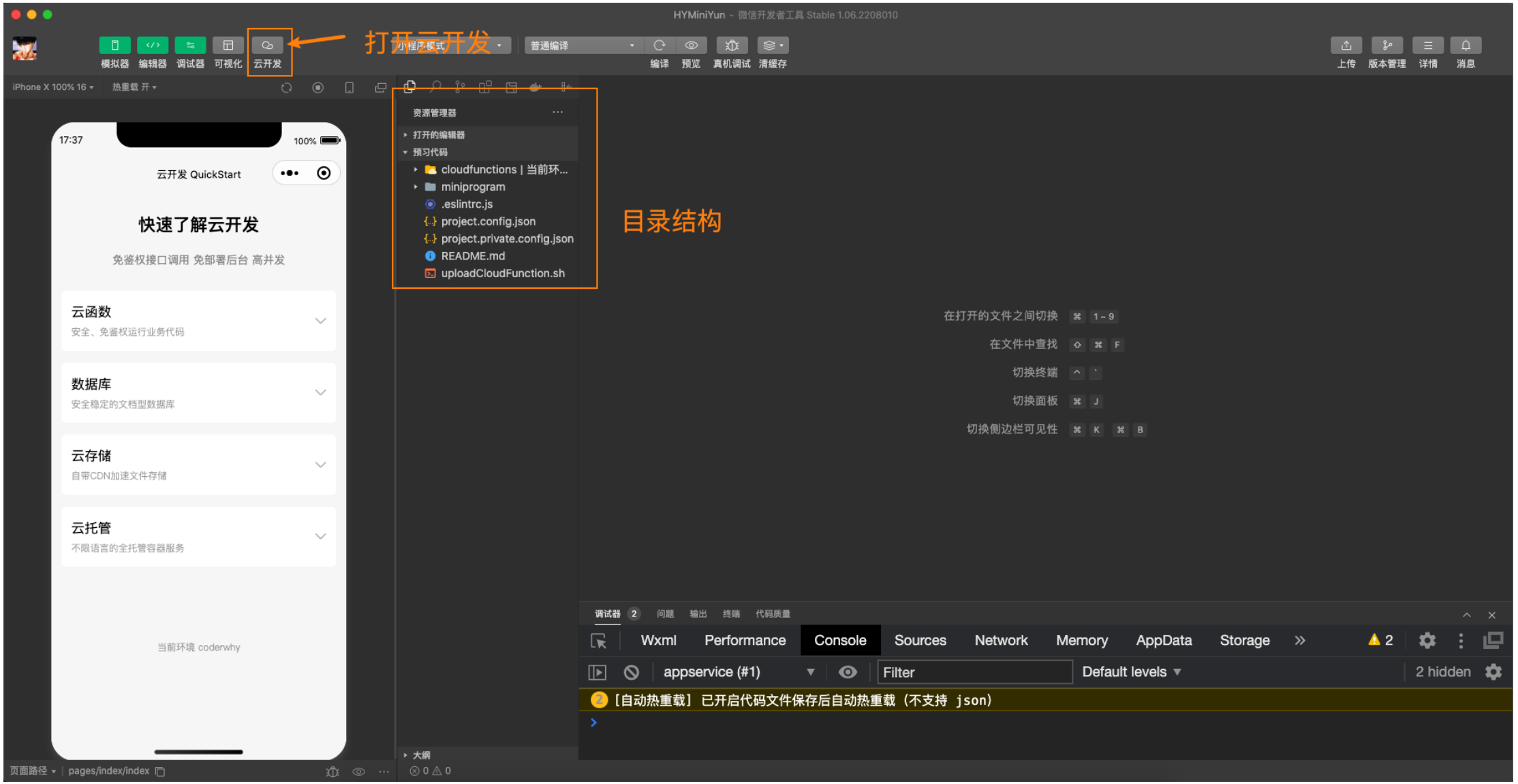
云开发核心技术

- **云开发主要包含三大核心技术：**
- **云数据库：**
  - 提供在小程序端直接对数据库进行增删改查的能力；
  - 数据库是类似于MongoDB的文档存储的数据库，操作非常方便；
- **云存储：**
  - 可以在小程序端直接上传、下载、删除文件；
  - 自带CDN，提高文件访问速度；
  - 可以获取临时链接，支持在小程序外访问；
- **云函数：**
  - 提供了在服务器代码的执行能力；
  - 包含微信天然的私有鉴权；
  - 更大权限的操作数据库等；
  - 进行云调用、HTTP请求等操作；

创建云开发项目

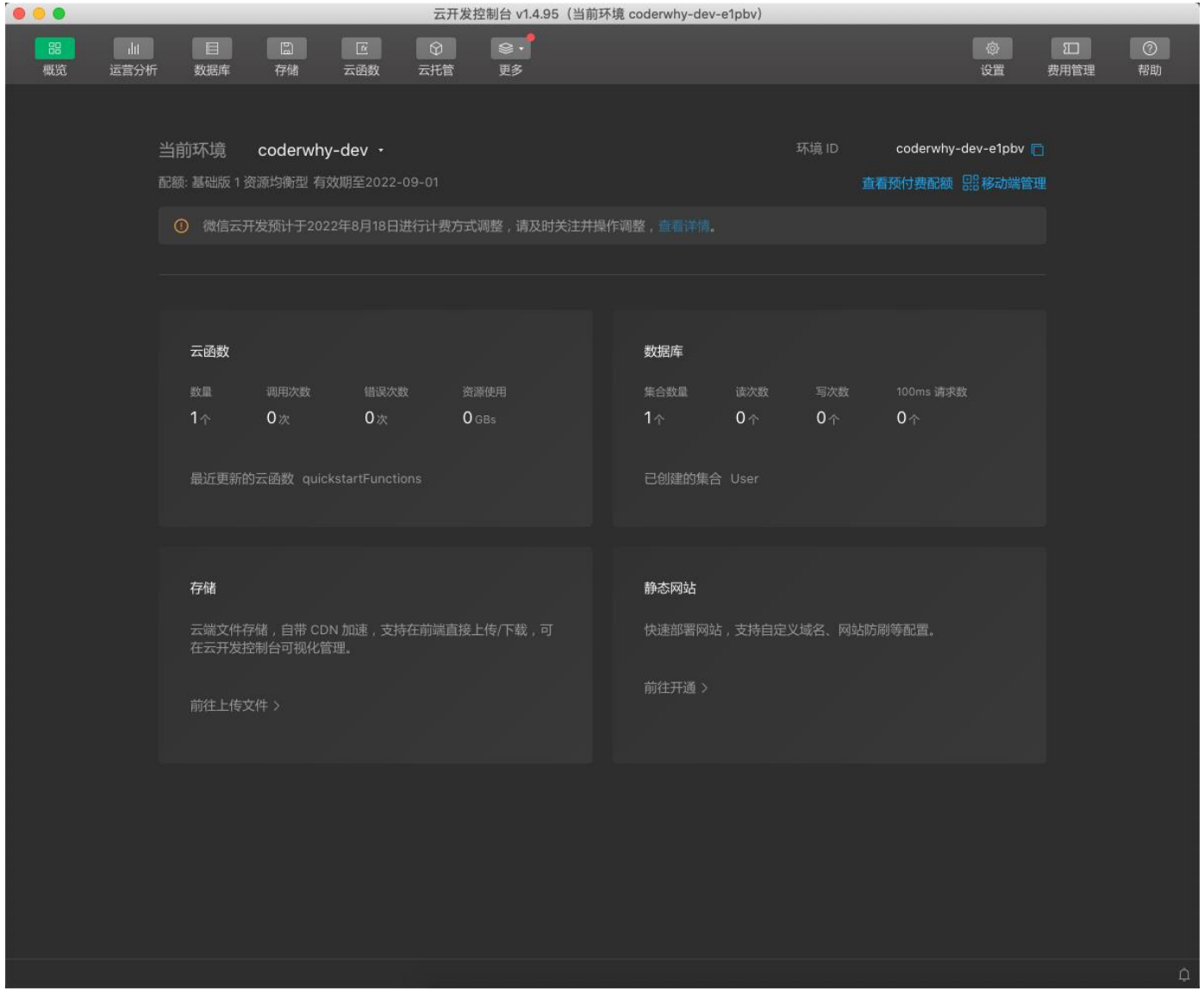

输入项目名称
小程序项目
小程序
小游戏
代码片段
公众号网页项目
其他
注销 ›
创建小程序
项目名称
LearnMiniCloud
目录
/Users/coderwhy/WeChatProjects/miniprogram-1
AppID
注册 或使用 测试号
开发模式
小程序
后端服务
微信云开发
不使用云服务
当前小程序 AppID 已开通云开发。你可以选择本次创建项目的初始模板。详情
模板选择
全部来源
全部分类
扫描试用模板
云开发 - 基础模板
官方
活动报名 - 小程序模版
云开发
愿望清单模板
云开发
取消
确定
15:26
77%
云开发 QuickStart
快速了解云开发
免鉴权接口调用 免部署后台 高并发
云函数
数据库
云存储
自带CDN加速文件存储
云托管

项目目录结构


HYMiniYun - 微信开发者工具 Stable 1.06.2208010
模拟器
编辑器
调试器
可视化
云开发
打开云开发
普通编译
编译
预览
真机调试
清缓存
上传
版本管理
详情
消息
iPhone X 100% 16
热重载 开
17:37
100%
云开发 QuickStart
快速了解云开发
免鉴权接口调用 免部署后台 高并发
云函数
安全、免鉴权运行业务代码
数据库
安全稳定的文档型数据库
云存储
自带CDN加速文件存储
云托管
不限语言的全托管容器服务
当前环境 coderwhy
资源管理器
打开的编辑器
预习代码
cloudfunctions | 当前环...
miniprogram
.eslintrc.js
project.config.json
project.private.config.json
README.md
uploadCloudFunction.sh
目录结构
在打开的文件之间切换
在文件中查找
切换终端
切换面板
切换侧边栏可见性
调试器 2
问题
输出
终端
代码质量
Wxml
Performance
Console
Sources
Network
Memory
AppData
Storage
appservice (#1)
Filter
Default levels
2 hidden
[自动热重载] 已开启代码文件保存后自动热重载（不支持 json）
大纲
页面路径
pages/index/index

云开发控制台

- **云开发控制台**
  - 运营分析
  - 数据库（云数据库）
  - 存储（云存储）
  - 云函数
- **开通云开发**
  - 设置 – 环境名称 – 创建环境



环境和配额

- **什么是环境：**
  - 一个环境对应一整套独立的云开发资源，包括数据库、存储空间、云函数等资源。
  - 各个环境是相互独立的，用户开通云开发后即创建了一个环境，默认可拥有最多两个环境。
  - 在实际开发中，建议每一个正式环境都搭配一个测试环境，所有功能先在测试环境测试完毕后再上到正式环境。

- **什么是配额：**
  - 默认有一定的免费配额（已经改成了1个月免费）；
  - 后期可以根据自己的业务量选择对应的更高配额；
  - https://developers.weixin.qq.com/miniprogram/dev/wxcloud/billing/quota.html

云开发项目初始化

- 在小程序端开始使用云能力前，需先调用 wx.cloud.init 方法完成云能力初始化

字段	数据类型	必填	默认值	说明
env	string | object	是（否）	默认选中环境	后续API 调用的默认环境配置，传入字符串形式的环 境 ID 可以指定所有服务的默认环境
traceUser	boolean	否	false	是否在将用户访问记录到用户管理中，在控制台中 可见

```
// 2.初始化云开发能力
wx.cloud.init({
  env: "coderwhy-dev-e1pbv",
  traceUser: true
})
```

project.config.json

```
{
  "miniprogramRoot": "miniprogram/",
  "cloudfunctionRoot": "cloudfunctions/",
```

云数据库介绍

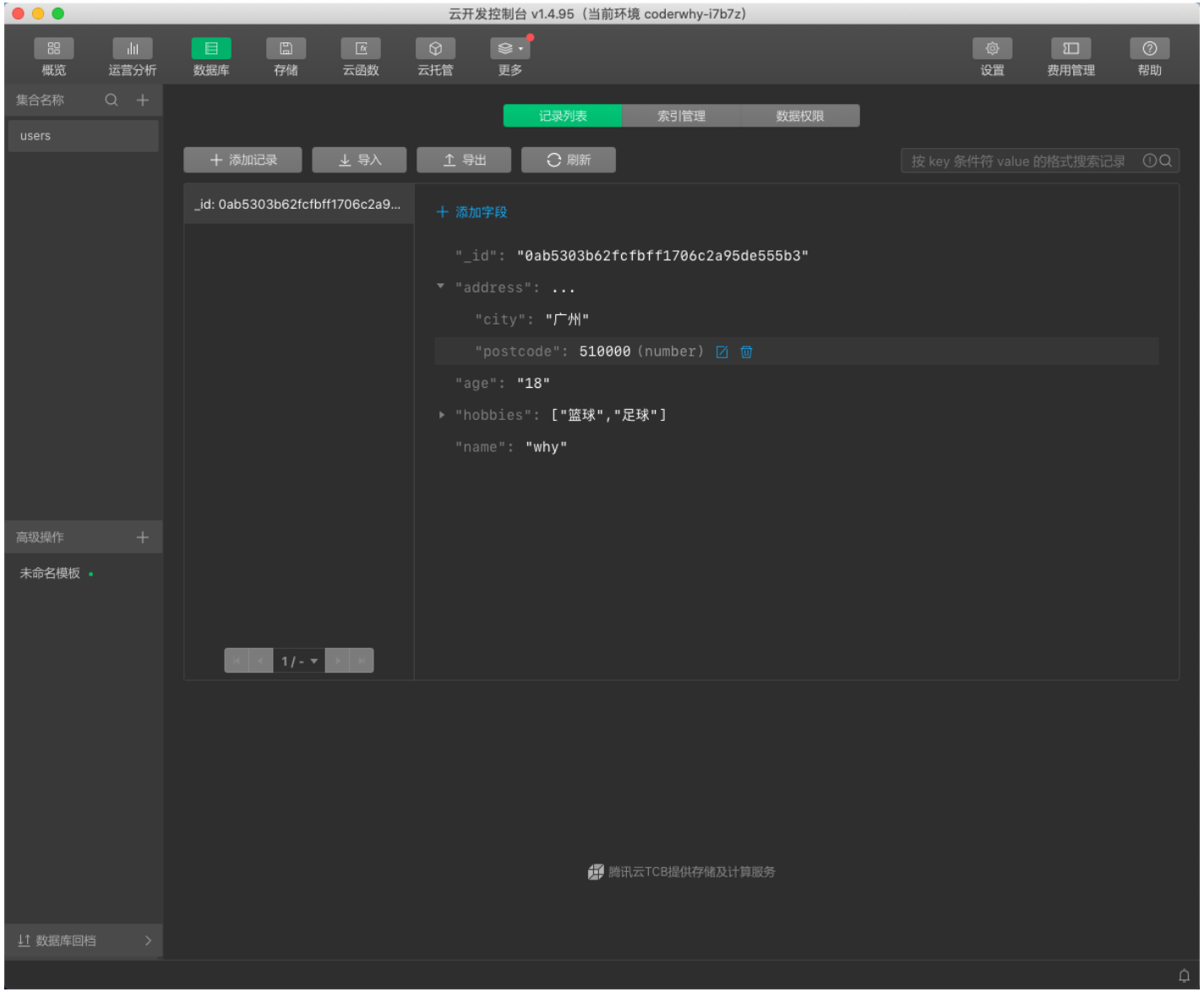
- **JSON数据库：**
  - 云开发提供了一个文档型数据库，类似于MongoDB，里面存放的是一条条JSON格式的对象；
  - 一个数据库可以包含多个集合，一个集合中包含多个JSON对象；

关系型	文档型
数据库 database	数据库 database
表 table	集合 collection
行 row	记录record / doc
列 column	字段 field

- 提供方便的API调用：学习这些API即可；
- 提供了小程序端和服务器端（云函数）中调用的区分；

操作数据库 – 控制台操作

- **演练：**
  - 创建集合
  - 创建一条数据
  - 添加字段
  - 导入一组数据



添加数据 和 调用结果

- **添加数据的调用过程：**
  - 1. 获取数据库对象
  - 2.获取操作的集合
  - 3.添加数据
- **获取操作后的回调结果：**
  - 基于回调：传入success、fail、complete
  - 基于Promise：使用then、catch、finally

```
onInsertDataTap() {
  // 1.获取数据库和集合
  const db = wx.cloud.database()
  const collection = db.collection("users")

  // 2.在数据库中添加数据
  collection.add({
    data: {
      name: "kobe",
      age: 30,
      hobbies: ["篮球", "足球"],
      address: {
        "city": "洛杉矶",
        "postcode": 11111
      }
    },
    success: (res) => {
      console.log(res);
    }
  })
},
```

修改数据

- **修改数据有两种方式：**
  - update：更新（增加）某一个字段
  - set: 使用新对象替换原来对象

```
onUpdateDataTap() {
  const db = wx.cloud.database()
  const collection = db.collection("users")

  // 更新某一个字段
  collection.doc("0ab5303b62fcfbff1706c2a95de555b3").update({
    data: {
      age: 20,
      hobbies: ["篮球", "乒乓球"]
    }
  })
  // 替换正条数据
  collection.doc("058dfefe62fcff411588020820f07714").set({
    data: {
      name: "james",
      age: 25
    }
  })
},
```

删除数据

- 对记录使用 remove 方法可以删除该条记录

```
onDeleteDataTap() {
  collection.doc("058dfefe62fcff411588020820f07714").remove({
    success: (res) => {
      console.log("删除成功:", res);
    }
  })
},
```

- 如果需要更新多个数据，需在 Server 端进行操作（云函数）；

查询数据一

- **查询数据的方式：**
- **方式一：通过ID查询精确的某一条数据；**
  - 使用doc查询ID
- **方式二：根据条件查询满足条件的数据；**
  - 使用where作为条件
- **方式三：通过指令过滤数据；**
  - 使用db.command的指令
- **方式四：通过正则表达式匹配符合的数据；**
  - 使用db.RegExp创建正则规则
- **方式五：获取整个集合的数据（小程序端一次性最多20条，云函数中可以获取100条）；**
  - 直接调用get
- **方式六：过滤、分页、排序查询数据**
  - 使用field、skip、limit、orderBy

查询指令

- 假设我们需要查询进度大于 30% 的待办事项，那么传入对象表示全等匹配的方式就无法满足了，这时就需要用到查询指令。
- 数据库 API 提供了大于、小于等多种查询指令，这些指令都暴露在 db.command 对象上。

查询指令	说明
eq	等于
neq	不等于
lt	小于
lte	小于或等于
gt	大于
gte	大于或等于
in	字段值在给定数组中
nin	字段值不在给定数组中

```
// 3.根据查询指令
const _ = db.command
collection.where({
  rid: _.gt(5094760)
}).get().then(res => {
  console.log(res);
})
```

数据查询 – 代码

```
// 1.精准查询
collection.doc("6a6a269a62fd0d00029351665d32187a")
  .get().then(res => {
    console.log(res.data);
  });

// 2.根据条件查询
collection.where({
  nickname: "MrGemini"
}).get().then(res => {
  console.log(res.data);
})

// 3.根据查询指令
const _ = db.command
collection.where({
  rid: _.gt(5094760)
}).get().then(res => {
  console.log(res);
})
```

```
// 4.正则表达式查询
collection.where({
  nickname: db.RegExp({
    regexp: "x",
    options: "i"
  })
}).get().then(res => {
  console.log(res);
})

// 5.获取整个集合
collection.get().then(res => {
  console.log(res);
})

// 6.综合查询
collection.field({
  nickname: true,
  roomName: true,
  rid: true
}).skip(2).limit(3).orderBy("rid", "asc")
.get().then(res => {
  console.log(res);
})
```

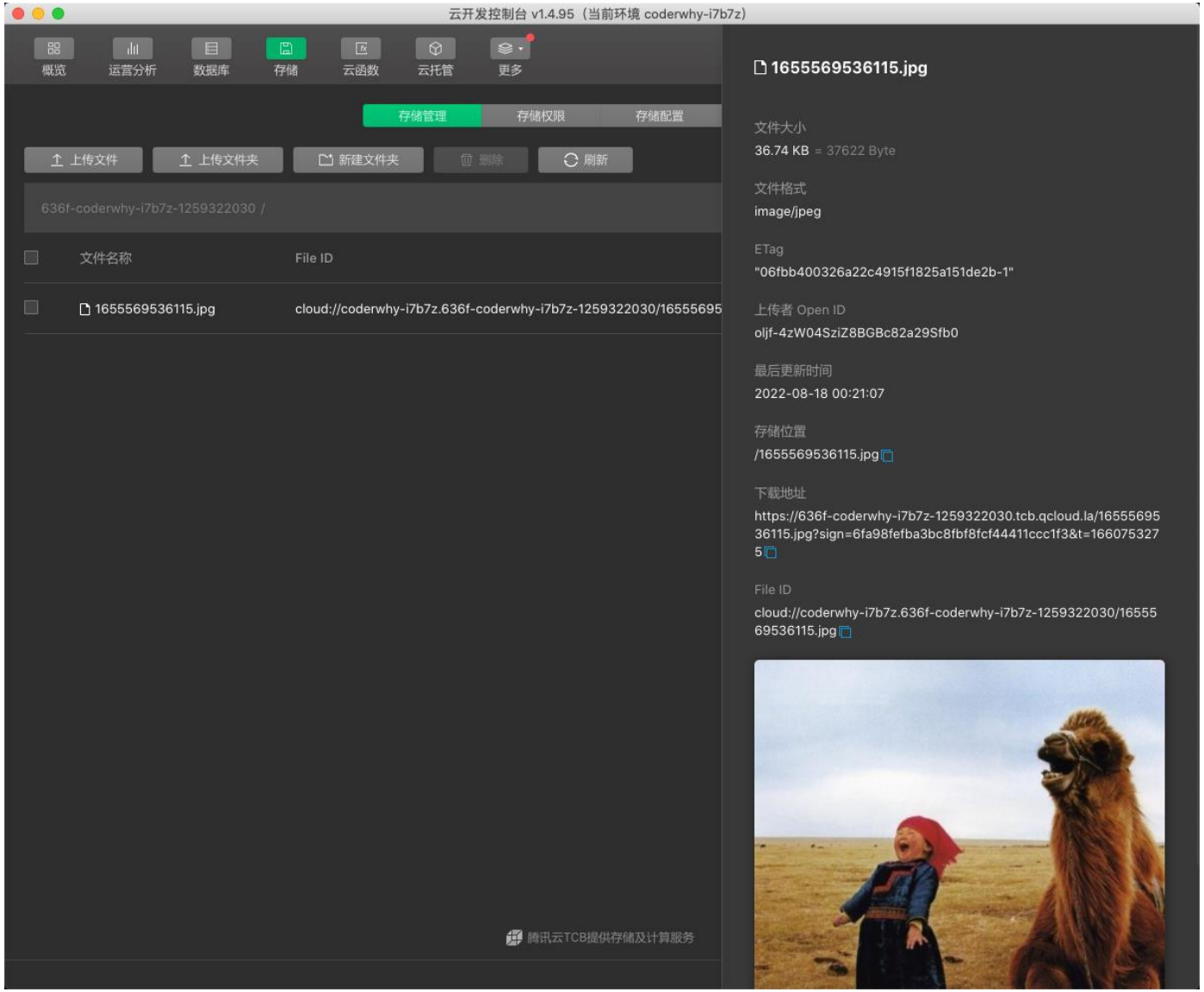
云存储介绍

- **云存储用于将文件存储到云端：**
  - 云存储提供高可用、高稳定、强安全的云端存储服务；
  - 持任意数量和形式的非结构化数据存储，如视频和图片；
  - 并在控制台进行可视化管理；

- **云存储常见的操作：**
  - 上传文件到云存储中（图片、视频、音频等等都可以）
  - 获取文件的临时链接（在外网可以访问）
  - 下载文件到本地（本地文件缓存）
  - 将云存储中的文件删除

云存储操作 – 在控制台操作

- **控制台演练：**
  - 上传文件到云存储中
  - 获取fileID在项目中显示
  - 获取URL在浏览器显示



云存储 – 上传文件

```
async onUploadFileTap() {
  // 1.从手机上选择一个图片
  const res = await wx.chooseMedia({ mediaType: "image" })
  const filePath = res.tempFiles[0].tempFilePath

  // 2.拼接图片的名称
  const timestamp = new Date().getTime()
  const openid = "coderwhy"
  const extension = filePath.split(".").pop()
  const filename = `${timestamp}_${openid}.${extension}`

  // 3.上传图片
  const result = await wx.cloud.uploadFile({
    filePath,
    cloudPath: "images/" + filename
  })
  console.log(result);
},
```

云存储 – 获取临时链接

- **为什么要获取临时链接?**
  - 我们将文件上传到云存储后，可以通过fileID在小程序中直接访问;
  - 但是，如果我们希望在小程序以外的地方访问（比如浏览器、手机端），那么fileID是不可以的;
  - 这个时候，我们可以通过获取临时链接，该链接可以在小程序以外访问;

```
async onGetTempURLTap() {
  const res = await wx.cloud.getTempFileURL({
    fileList: ["cloud://coderwhy-i7b7z.636f-coderwhy-i7b7z-1259322030/1655569536115.jpg"]
  })
  console.log(res);
}
```

- **注意：文件链接有效期为两个小时;**

云存储 – 下载文件

- 如果文件是放在云存储中，那么必然需要有网络的情况下才能访问。
- 某些情况下，我们可能希望把某些重要的文件下载到本地，就可以使用云存储的文件下载了。

```
// 1.下载图片
const res = await wx.cloud.downloadFile({
  fileID: "cloud://coderwhy-i7b7z.636f-coderwhy-i7b7z-1259322030/images/1660754565930_coderwhy.png"
})

// 2.显示图片
this.setData({ localImageURL: res.tempFilePath })
```

云存储 – 删除文件

- 某些文件不再使用时，可以将其从云存储中删除掉，这样可以省略空间

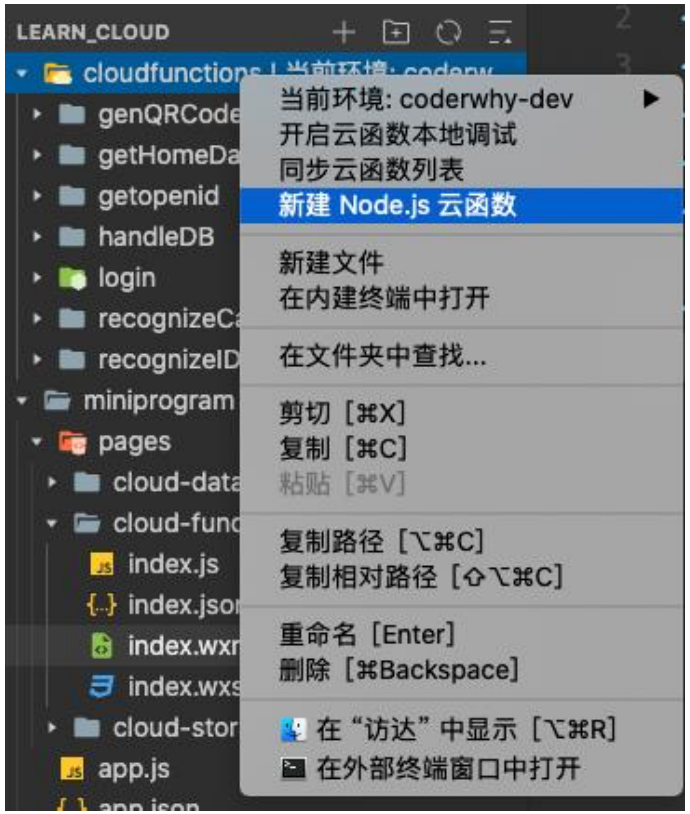
```
// 删除文件
const res = await wx.cloud.deleteFile({
  fileList: ["cloud://coderwhy-i7b7z.636f-coderwhy-i7b7z-1259322030/images/1660754565930_coderwhy.
  png"]
})
console.log(res);
```

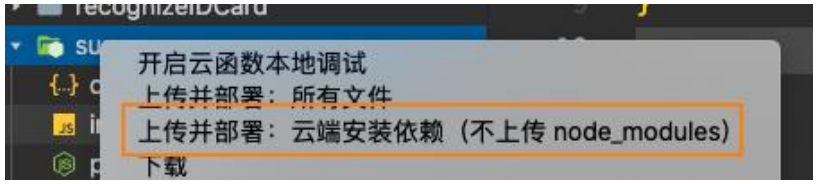
云函数和云调用

- **云函数即在云端（服务器端）运行的函数：**
  - 在物理设计上，一个云函数可由多个文件组成，占用一定量的CPU 内存等计算资源；
  - 各云函数完全独立，可分别部署在不同的地区；
  - 开发者无需购买、搭建服务器，只需编写函数代码并部署到云端即可在小程序端调用；
  - 同时云函数之间也可互相调用；
- **云函数的编写方式：**
  - 一个云函数的写法与一个在本地定义的 JavaScript 方法无异，代码运行在云端 Node.js 中；（需要专门学习Nodejs吗？）
  - 当云函数被小程序端调用时，定义的代码会被放在Node.js 运行环境中执行；
  - 我们可以如在 Node.js 环境中使用 JavaScript 一样在云函数中进行网络请求等操作，而且我们还可以通过云函数后端 SDK 搭配使用多种服务，比如使用云函数 SDK 中提供的数据库和存储 API 进行数据库和存储的操作
- **云开发的云函数的独特优势在于与微信登录鉴权的无缝整合。**
  - 当小程序端调用云函数时，云函数的传入参数中会被注入小程序端用户的 openid，开发者无需校验 openid 的正确性因为微信已经完成了这部分鉴权，开发者可以直接使用该 openid。

云函数使用过程

- **云函数的使用过程：**
  - 1.创建一个云函数
  - 2.编写云函数的代码逻辑
  - 3.将云函数上传到云端
  - 4.小程序中对云函数调用





云函数的基本使用

- 案例：让云函数帮我们计算两个数字的和


```
async onNumSumTap() {
  const res = await wx.cloud.callFunction({
    name: "sum",
    data: {
      num1: 20,
      num2: 30
    }
  })

  console.log(res.result);
},
```

云函数 – 获取openID

- openid可以用于作为用户身份的标识符，所以在云开发中我们可以获取用户openid来验证用户是否已经登录。
- 在云函数中获取微信调用上下文
  - Cloud.getWXContext(): Object
  - https://developers.weixin.qq.com/miniprogram/dev/wxcloud/reference-sdk-api/utils/Cloud.getWXContext.html

```
// 云函数入口函数
exports.main = async (event, context) => {
  const wxContext = cloud.getWXContext()

  return {
    event,
    openid: wxContext.OPENID,
    appid: wxContext.APPID,
    unionid: wxContext.UNIONID,
  }
}
```

云函数 – 操作数据库

- **云函数中对数据库的操作限制更少，所以我们常常会在云函数中进行数据库操作：**
  - 比如可以根据条件一次性删除多条数据；
  - 比如对数据请求的个数没有严格的限制；

```
// 云函数入口文件
const cloud = require('wx-server-sdk')

cloud.init()

// 云函数入口函数
exports.main = async (event, context) => {

  const db = cloud.database()
  const collection = db.collection("users")

  const _ = db.command
  const res = await collection.where({
    age: _.gt(10)
  }).remove()

  return res
}
```

云函数 – 发送http请求

- 云函数中支持对其他服务器进行http请求，也支持使用axios库发生网络请求：

```
// 云函数入口文件
const axios = require("axios")
const cloud = require('wx-server-sdk')

cloud.init()

// 云函数入口函数
exports.main = async (event, context) => {
  const wxContext = cloud.getWXContext()
  const res = await axios.get("http://123.207.32.32:8000/home/multidata")
  return res.data
}
```

- 所以对于小程序某些域名的限制无法配置时，我们可以通过云函数作为代理来请求数据，再返回给小程序端；

云调用 – 生成小程序码

- **生成小程序码的逻辑：**
  - 对于使用云开发生成小程序码，相对于自己搭建服务器简单很少，因为它拥有天然的鉴权功能。
  - https://developers.weixin.qq.com/miniprogram/dev/api-backend/open-api/qr-code/wxacode.createQRCode.html

```
// 云函数入口文件
const cloud = require('wx-server-sdk')

cloud.init({
  env: cloud.DYNAMIC_CURRENT_ENV,
})

// 云函数入口函数
exports.main = async (event, context) => {
  const result1 = await cloud.openapi.wxacode.createQRCode({
    "path": 'page/cloud-database/index',
    "width": 430
  })

  const extensionName = result1.contentType.split("/").pop()
  const result = await cloud.uploadFile({
    cloudPath: "images/minicode." + extensionName,
    fileContent: result1.buffer
  })

  return result
}
```