阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY | 阿里云云开发平台

Serverless云开发从入门到实战

手把手带你掌握Serverless和云开发技术

1 大平台无门槛上手
3 大基础能力系统学习
4 大实战案例轻松掌握

藏经阁

阿里云开发者电子书系列

钉钉扫一扫加入
Serverless 云开发沟通交流群

阿里云开发者“藏经阁”
海量免费电子书下载

目录

第一章 云开发入门篇 3

认识 Serverless 云开发平台 5

1 分钟开发一个 API 13

1 分钟开发一个网站 18

第二章 云开发框架篇 21

Koa 框架实现服务端渲染部署上线 22

SSR 框架实现服务端渲染部署上线 27

NodeJS 及 Python 主流框架部署上线 38

Springboot 应用部署上线 41

第三章 云开发后端篇 52

如何访问云 OTS 数据库和实践 53

如何访问云 MySQL 数据库和实践 62

第四章 云开发提高篇 70

轻量易用的运维监控能力 71

不改变本地流程的 CI/CD 能力 82

第五章 云开发实战篇 90

开发钉钉机器人 91

智能生产代码与 AI 应用的开发实践 98

快速开发天猫精灵智能应用问答百科 105

前后端一体化应用开发实战 116

第一章 云开发入门篇

认识 Serverless 云开发平台

作者 | 风驰 阿里巴巴高级技术专家

在阿里云的云开发平台，可以一站式支持整个研发团队在线上进行应用研发工作。使用阿里云账号登录平台，进入到平台之后我们可以选择不同的项目团队，就能看到云开发平台的应用列表操作界面，在这个界面大家可以看到一些简单的入口，那么下文将按顺序分别介绍一下。

在应用列表操作界面的顶部，是阿里云云开发平台的介绍，旁边是非常重要的帮助入口。进入这个入口可以找到平台提供的重要文档，例如《 完整的快速入门 》，文档将指引首次登录的开发者如何创建团队和管理团队，如何创建应用产品开发部署等，基本上涵盖了一个完整研发生命周期的全部内容指导。

一、帮助文档

团队协同里重点介绍了在整个研发平台，团队怎么创建，团队的成员角色是怎样子的，以及怎么将一个开发任务分配给一个具体的开发成员。

场景与解决方案里包含基础 Helloworld 等云原生的 Serverless 的应用模版，讲解如何在云开发平台创建和部署。2020 年 9 月云栖大会期间我们组织了 Hello World 上手活动，旨在帮助大家通过简单的 Hello World，一个标准的程序员或者开发者的通用语言，来快速感知云原生 Serverless 应用如何开发。

在这个入口里有很多场景和解决方案。比如天猫精灵的方案怎么开发，比如基于前端的 NodeJS 的一系列的应用框架怎么快速迁移上来？比如说 EggJS、Express、KOA NextJS、Nuxt 以及更多的应用，都可以通过这些方式快速迁移上来。还有阿里巴巴的 Serverless 框架，Midway Serverless，以及基于 Midway Serverless 系列怎么操作数据库的 RDS、OTS 等等。除此之外， Python 语言的开发迁移、PHP 的开发迁移，以及 Java 的 SpringCloud，SpringBoot，SpringMVC 等等的迁移。

基于阿里云的智能视觉平台提供的开放能力，我们可以做很多有趣的应用。比如，阿里巴巴自研的从图像智能生成代码的 ImgCook 的应用，通过 ImgCook 生成一个 H5，这个 H5 的背后是一个上传图片和识别图片的功能，所以通过阿里云智能服务可以实现很多有趣的创意。

OSS 文件浏览器直传方案，它可以实现与往常不一样的文件上传。它向开发者展示了如何在应用、存储、分离的情况下实现文件上传。

高阶的开发辅助里有 Codeup 代码托管的详细使用方法介绍，以及 Java 的在线流水线 Flow 的使用方法介绍。如果你有深度学习的需求，或想更多地挖掘阿里云云开发平台的服务能力，可以多点开看看这个入口。

服务条款里介绍了阿里云云开发平台是通过什么方式为大家提供服务的。其中费用与账单中，开发者可以查看在这个平台上已经应用开发部署了哪些资源和使用了哪些服务，费用一栏是与之对应的。如何计费、怎么支付，这里都有详细的介绍。

作为一名合格的开发者，阅读文档是一件很重要的事情。云开发平台的帮助文档里，可以为大家解答了很多重要的问题。

二、团队协同界面

在阿里云标志的左侧有 9 个点的入口，点开是一个可以展开的团队协同的界面。界面里有工作台、知识库、测试管理、制品仓库、流水线、代码管理、企业成员等等一系列团队协同的工具，点开任何一个工具都可以应用到团队中。

作为开发者可以基于工作台创建一些任务，并对一些任务和项目进行管理。比如代码管理，每一个应用都会给使用者分配一个代码仓库，通过代码管理可以详细地了解应用的代码仓库是什么样子，以及针对它的所有更细节的操作你都可以实现。

如果想做更多的团队协同，可以通过云开发平台左上角 9 个点的入口展开菜单，去做更多的事情。

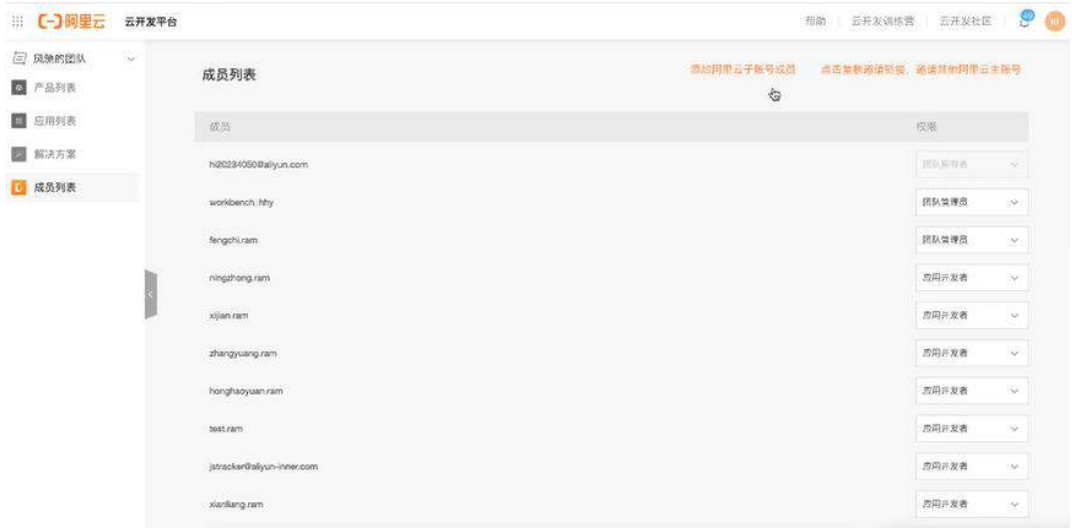
三、团队权限分配

云开发平台登录后，可以通过左侧的当前团队查看自己所在的团队及团队所给予的权限。这就保障了大家在做任务或想创建应用的时候，及时了解自己的权限，避免不必要的时间浪费。

产品列表、应用列表、解决方案以及成员列表，是当你有一定的权限后才能看到完整的入口。举个例子，如果在当前团队只是开发者，这里的视图就会发生变化，因为开发者没有权限管理和创建应用。

以上所介绍的就是团队协同的概念及应用。当大家把团队整个搬上来时，可以给每一位团队成员分配权限。那么如何把团队搬上平台，怎么给团队成员分配权限?

四、邀请团队成员



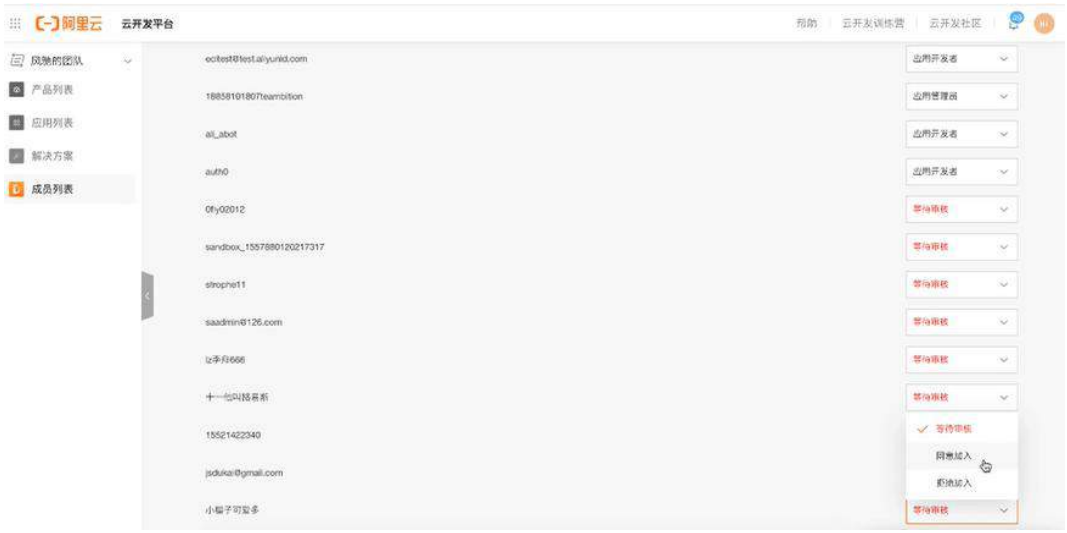
在当下团队下打开成员列表，可以看一张视图，主体是当前团队所有成员，右上角有两个入口，一个是添加阿里云子账号成员，一个是点击复制邀请链接，邀请其他阿里云主账号，这两个有什么区别呢?

点击复制邀请链接，是指邀请其他阿里云主账号。意味着你邀请的人首先拥有阿里云的账号，他接受邀请加入你的团队之后，也是用他的阿里云账号登录。由于他是独立的阿里云账号,作为团队管理者你可能对他没法做管控,这种情况适用像开源的共建项目的相互协同。这种方式就会比较灵活，每个人都用自己的账号登录。

当受邀人打开链接后，会看到显示邀请人姓名的提示界面，点击同意加入团队后，等待审批。邀请人这边界面会提示有新用户申请加入，审批同意后受邀人即可通过阿里账户登录团队。

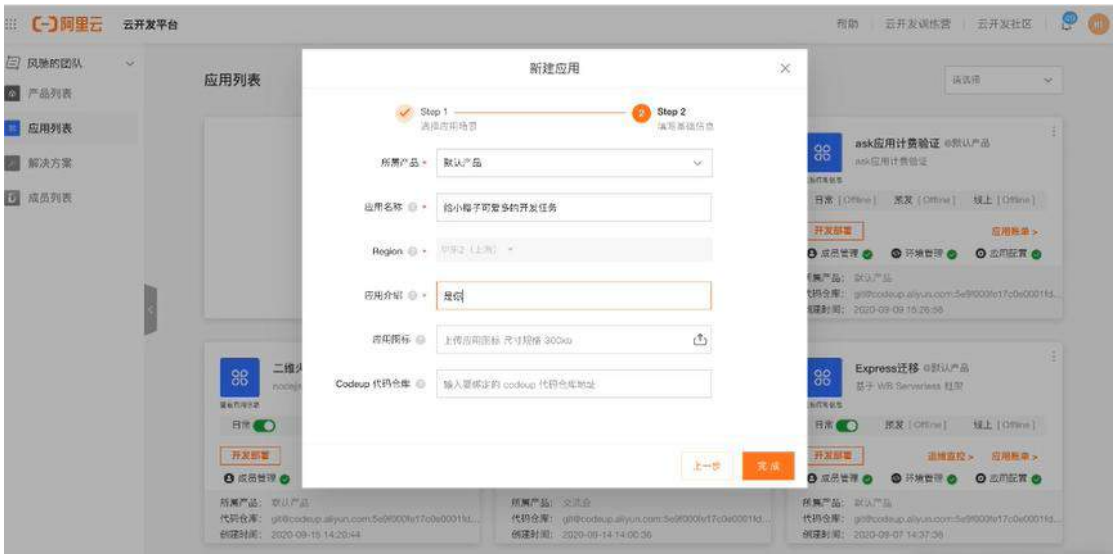
邀请阿里云子账号成员，是指邀请你主账号之下创建的子账号。首先我们可以通过前往创建 ram 子账号入口，选择主账号人并给他创建子账户，并在创建过程中勾选控制选项，并设定登陆密码。子账号创建生效后，只需勾选此子账号，就完成了子账号邀请工作。

大家在加入团队后可以按分配的任务进行执行，实现协同开发。所有团队成员角色和他们对应的权限，可以是应用开发者、应用管理者以及团队管理园等权限，通过第一点所提到的帮助文档里团队协同中查看。



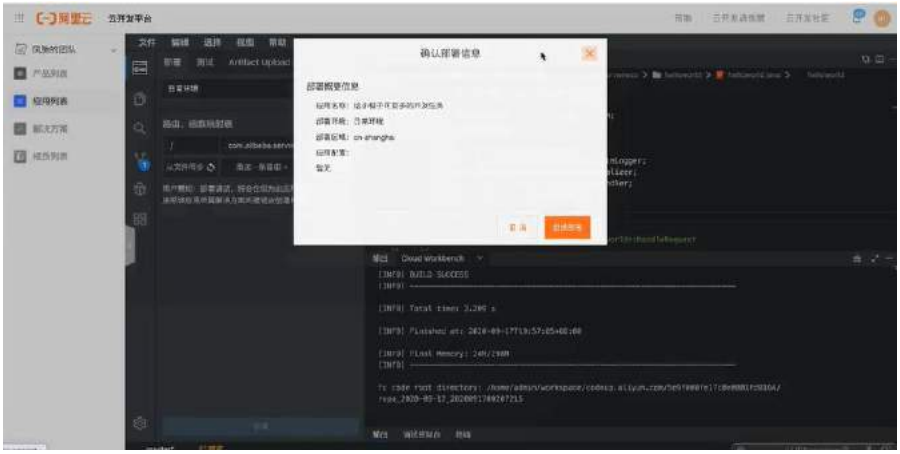
五、部署与测试

管理员进入应用列表，点击创建应用并选择默认产品完成创建应用。创建应用后进入代码仓库，将我们选择的解决方案初始化脚手架文件内置到代码仓库中，等待分配。



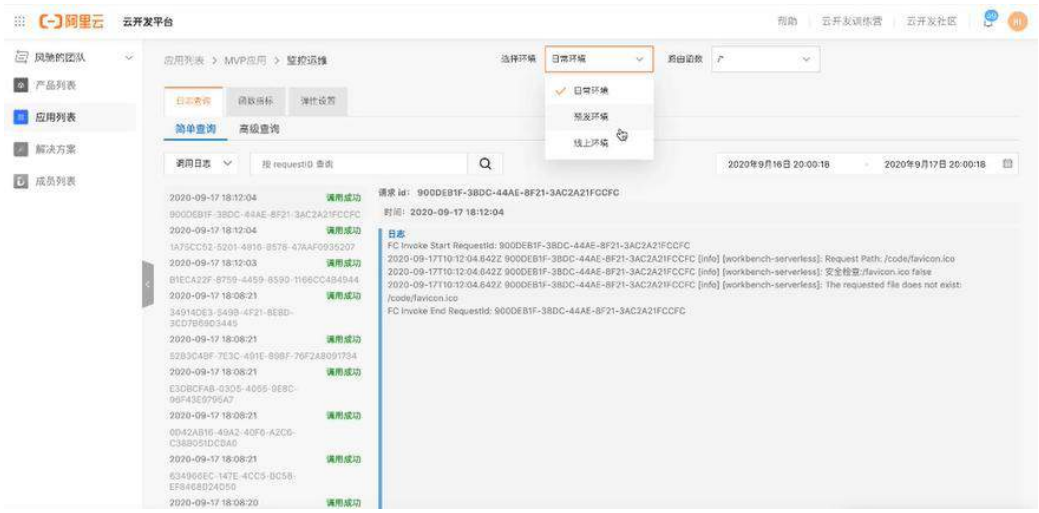
当任务准备就绪，点击成员管理，选择并勾选目标开发员，任务分配成功。新加入的成员系统会默认为应用开发者，作为团队拥有者可以因需调整成员的权限。

当成员在进行开发任务的时候，团队拥有者可以打开界面与之同步进行开发，这就是有趣的结对编程。完成代码开发后，点击 CloudIDE 左侧第一个部署的 tab 进行一键快速部署。



如果部署成功，此应用标题下会有绿色显示，表示已经是 Online 状态；如果没有部署成功，右边会显示 Offline 状态。这就让哪些应用在哪个环境，是否成功部署，一目了然。

部署的应用如果想下线并停止服务也很简单，只需点击当下应用，并确认下线停止服务即可。如果是真实的线上服务应用，这个操作要非常谨慎。



当部署的应用出错了，我们可以通过运维监控入口进入，查看运维监控的面板。可选择需要查看的运维日志及监控环境。同样在上文提及的第一点，通过帮助文档的快速入门中学习怎么查看日志和测试。

1 分钟开发一个 API

作者 | 洪浩原 阿里云高级技术工程师

如果你是第一次使用阿里云云开发平台，请先阅读第一篇文章开通云开发平台。

接下来，我们以 NodeJS 语言的视角来完成这次开发。如果您擅长其它语言，不必担心，在文章结尾我们提供了其它语言的使用方式，而且它们几乎是完全相同的。

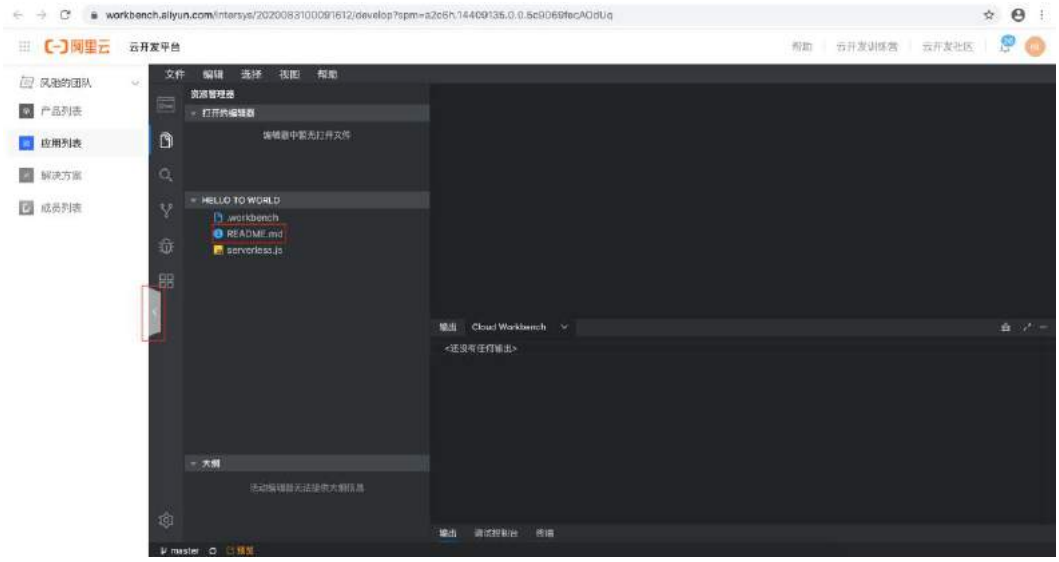
进入云开发平台后，点击【创建新应用】，选择【空应用】选项卡，并选中函数计算 FC 作为我们代码的运行服务器。

从下拉的语言列表中，选择你擅长的语言，点击下一步，填写信息来创建出一个云开发平台应用。

开通云服务：点击应用卡片上的“环境管理”，查看服务的开通情况，未开通的服务点击立即开通，根据提示开通。开通后应用卡片环境管理后面的小图片会变成绿色的对勾形状。

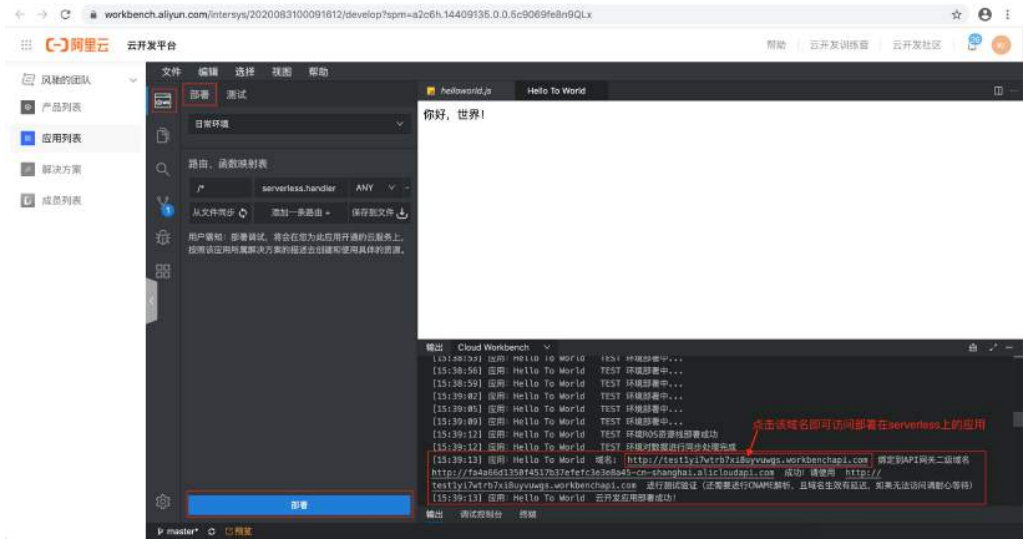
开通云资源访问授权:点击窗口右下方的授权图标，同意云开发平台对云资源的访问授权。

稍等几秒页面刷新，就可以点击【开发部署】来进入我们真正的开发界面了。

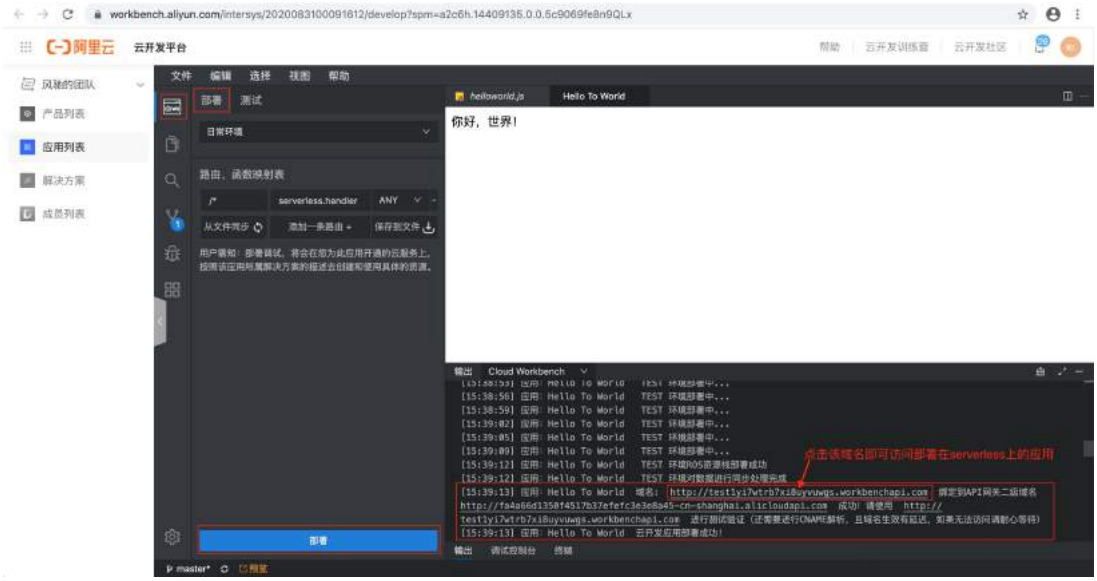


进入 CloudIDE 之后的第一步，是寻找左侧文件列表的 README.md 文件，里面有关于这种解决方案的丰富的说明信息。

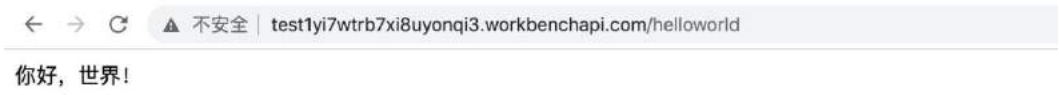
按照其中给出的提示，我们复制一段符合 FC 函数计算格式的 NodeJS 代码到 js 文件中，然后在 do sth 下面编写我们自己的 helloworld 相关的代码；将返回值状态更改为 200，将返回值 body 更改为我们编写的 helloworld 变量，保存文件即可。注意一定要 Ctrl+S 或者鼠标点击来保存刚编写的文件。



打开 CloudIDE 最左侧「WB」插件的「部署」标签页，点击「部署」，会弹出部署信息确认，点击「继续部署」开始构建部署，请耐心等待，直到部署完成。部署完成后会看到如下信息，云开发平台会免费分配一个临时的二级域名对部署的应用进行访问。



将上图中的测试域名复制到浏览器，或者直接 Ctrl+鼠标点击，来访问部署好的 API，并在后面追加上图左侧路由表中填入的部署路径，即可看到结果：



现在，你就可以在你的 html 里，或者其它程序片段中嵌入这个临时的域名，来让其他人或者程序代码访问这个 API 了。如果您了解如何编写 NodeJS 代码，那么整个过程就只需点击一次点击【部署】按钮，除此之外，您并不需要购买或维护哪怕一台服务器，不需要有任何的 Serverless 知识储备，也不需要您繁琐地备案一个域名（我们的域名仅供临时访问），就可以跑起来一个有无限扩展可能的 API 了。

当然，作为 FC 函数计算，我们会按根据实际的流量消耗对应的资源为您处理用户请求，整个过程开始按量计费。不过它有一定的免费额度，所以用来测试的时候大可不必担心费用问题。

为避免不必要的费用消耗，可以主动将部署的应用停止服务，当然，如果是真实的线上业务，就确保它们在线上正常运行就好。

、

如果您使用其它语言版本，那么过程和 NodeJS 是几乎一样的，唯一的区别在于 README.md 文件中，您会创建不同语言对应格式的代码文件，来实现您的 API 逻辑。

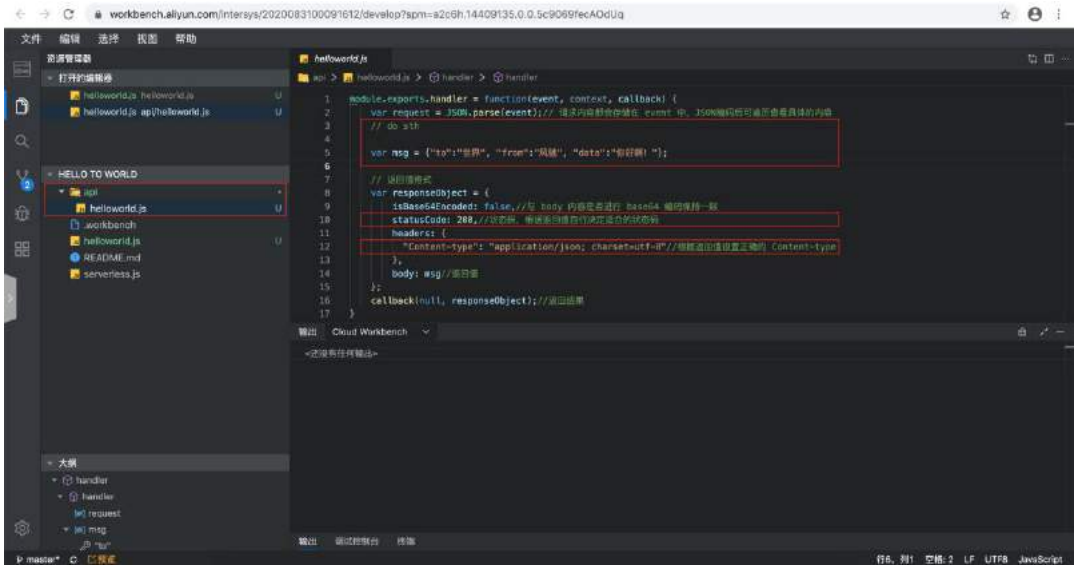
1 分钟开发一个网站

作者 | 洪浩原 阿里云高级技术工程师

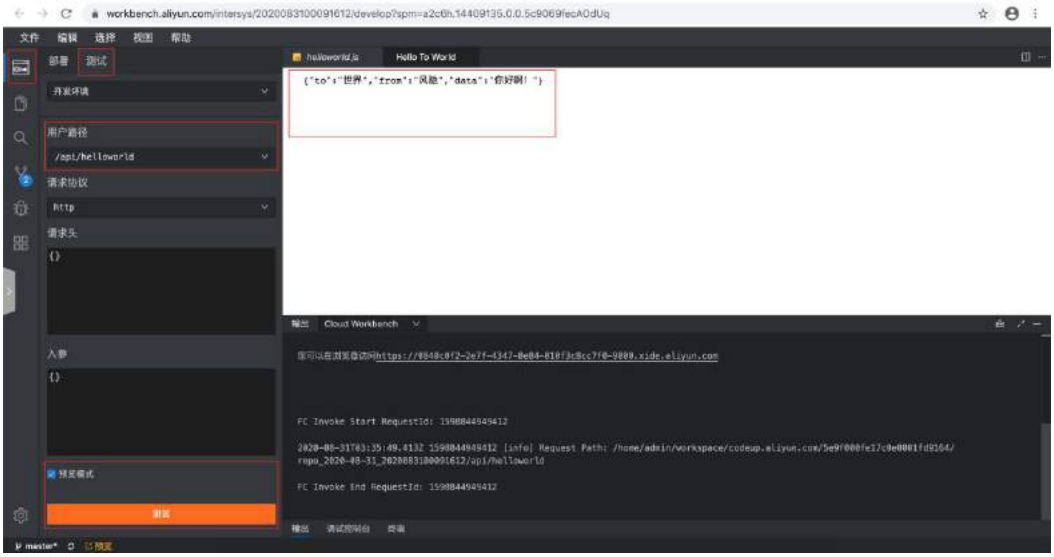
如果你是第一次使用阿里云云开发平台，请先阅读第一篇文章开通云开发平台。

上一节课我们学习了 API 的开发，但是 API 只能为前端提供服务，我们还需要一个前端网站，来将 API 转换成可视化的界面。

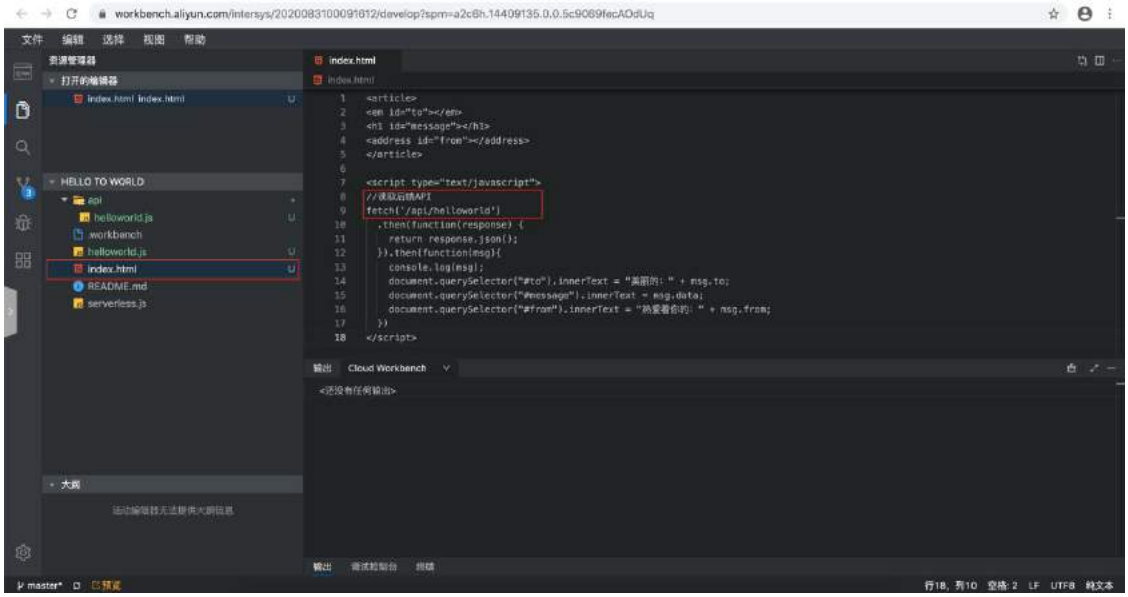
接着上一节课的过程，我们依旧以 NodeJS 作为例子进入应用的开发页面，在 Cloud IDE 中 创建 api 目录，把根目录下的 helloworld.js 复制一份到 api 目录，打开 api 目录下的 helloworld.js 文件，把返回值改为一个 JSON 对象，把返回值类型改为 application/json 类型。其它语言也有对应设置返回头的方法，需要参考各自语言的 httpSDK 和 README.md：



保存文件，打开「WB」插件，选择【测试】选项卡，无需【部署】，可以直接测试一下刚才写的 API 结果。在用户路径中写入 /api/helloworld：

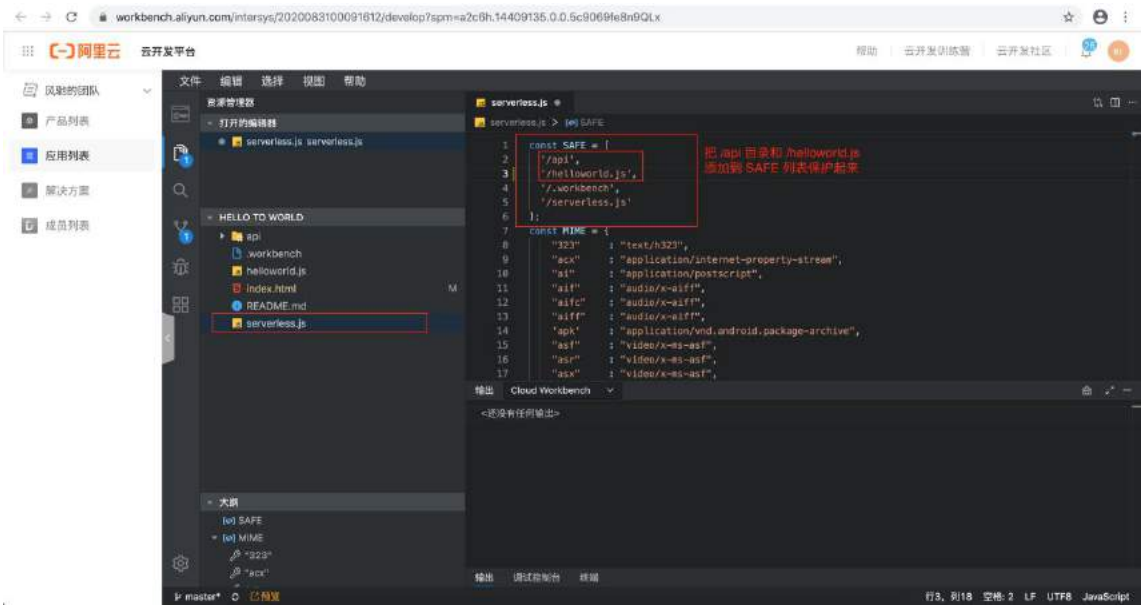


非常好，顺利输出了我们想要的结果，接下来我们在根目录下创建一个首页 index.html，写上一些基本的 html 标签和文字（或者从网上 copy 一段 html 示例代码）。并使用一段 js 的 fetch 指令来调用后端服务 /api/helloworld 进行输出：



保存，点击【部署】，将写好的 html 和 js 一同部署到 FC 函数计算上。部署之后，得到了临时的测试域名，可以在测试域名后携带/index.html 的路径来查看结果（因为我们的文件命名为了 index.html，所以根路径也可以看到结果）：

在本示例中，我们基于 NodeJS 提供后端服务，但是 .js 文件扩展名会被浏览器默认成下载行为，这就对我们的应用安全会造成安全风险。比如直接访问域名+“/index.js”来获取我们的 js 代码。我们要将所有后端服务的 .js 文件保护起来。打开「serverless.js」，在「SAFE」列表中，将要保护的后端服务文件/目录 添加进去即可：



重新部署后，在浏览器中访问 /helloworld.js， /api/helloworld.js 就不会被下载了

第二章 云开发框架篇

Koa 框架实现服务端渲染部署上线

作者 | 冰森 淘系前端技术专家

一、什么是 Koa？Koa 是一个 Node.js Web Framework

很多人都了解 Java，Java 里的 Maven 的是一个依赖的中心仓库，里面会有非常多的依赖；那么如果大家了解 Node.js 的话，会发现与 Maven 对标的叫 npm，npm 也是 Node.js 开发者常用的仓库。那么 Koa 就是 npm 里面一个非常常用的框架，一个依赖库。

如果大家浏览过 Koa 的官网，就会发现 Koa 主要是基于可扩展模型 http 协议的一个 Web 框架。Koa 的扩展方式是基于最经典的洋葱模型，它的洋葱模型就是它最核心的部分，即 Koa 中间件。

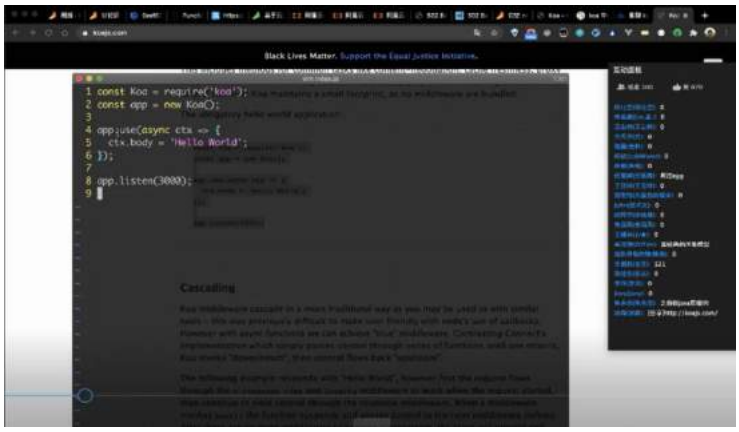
二、Koa 的基础使用

当我们用 http 做一些请求的时候，总会带着各种各样的 path，在程序员的话术里叫 router。我们通过不同的 router 进去，就能帮我路由到不同的方法上面去。我们现在推荐直接使用 Koa，因为它已经给我们提供好了 router。

在 Koa 官网上的 github 里可以找到很多 Koa 中间件和依赖库。

（演示）通过 npm install，在本地建一个仓库，这个仓库直接通过 node.js 的包管理工具 npm 的一个 install 命令，就可以把 Koa 装到本地。

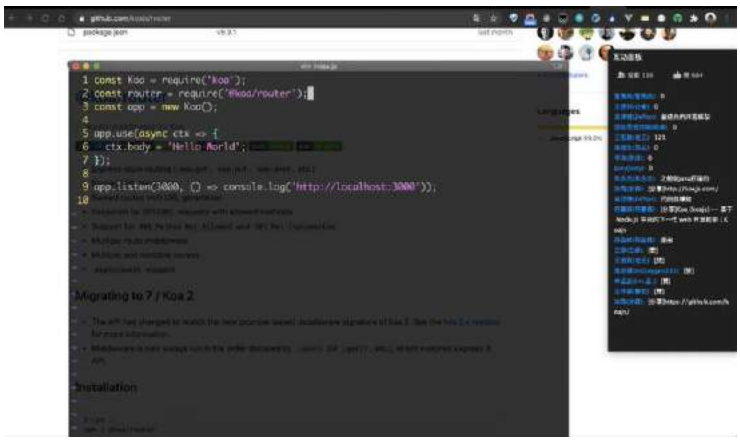
首先看一下在原生机器上使用 Koa 的例子。将 Koa 官网上的 helloworld 拷贝，它是通过 Node.js 的最基础的依赖的加载机制，require 把 Koa 这个模块加载进来之后，它是一个 class new 一个 Koa 的实例，然后通过 Koa 的扩展方式，也就是 Koa 中间件方式，通过 use 加载 Koa 中间件，可以实现在一个非常小的内核上扩展 web 应用，它的 Koa 内核是一个非常简单的代码。



Koa 中间件其实是一个很简单的 Function。这个 Function 会有一个 ctx 参数传进来，这个 ctx 相比原生的 Koa http 协议更具吸引力的是，Koa 它把原生的 http 的 request 对象和 response 对象全都代理到了 ctx 对象上可。也就是说它把请求和响应的各种操作的方法全都代理到了 ctx 上了。

有学过 Midway 的同学会发现它们的 ctx 本质上都是 Koa。关于 Koa 的 ctx 上面具体代理很多东西，大家可以到 Koa 的官网上查看它的文档。这样一个简单的代码，它应用了一个最小的中间件，这个中间件就是直接拿到 ctx，然后 ctx.body 我们可以到官方文档上面找到，在官网上 ctx.body 就是 response.body。

Koa router 这样一个中间件的使用方式。在 Koa group 里寻找中间件，装好中间件。我写个 hi serverless，这里加了一个路由，我们再加第二个，第二个取名叫 api，然后它这种中间件的使用方式也是通过这些中间件，最后要返回一个 Function 给 Koa 的 APP。（演示成果）我们先访问的是/path，出来的就是 hi serverless；再访问/api，出来的就是 hi api。



三、如何在云上写 Koa 应用?

上面所介绍和演示的都是在本地的 Koa 基础使用。那么如何在云开发平台上写 Koa 应用呢?

通常我们在 Node.js 里面，把不同的 path 叫做不同的路由，特别是在 Koa 体系中，Koa 和 express 这两套体系主要的相关的这些衍生产品，他们都喜欢管不同的 http 请求的 path 叫路由。本质上通过一个 http 请求的时候传了不同的 path，然后通过不同的 path 映射到不同的处理方法上。

（演示）我们这边已经有一个已经 ready 的 Koa 的应用迁移方案，大家可以像前几天学的去打开这个界面并新建一个应用，在选择解决方案的时候去勾选 Koa 的应用迁移方案，然后完成新建。

完成新建后，点开发部署并进入云上 IDE 的界面。云上 IDE 可以让我们直接在云上写代码，还可以在云上测试。这个整理好的 Koa 应用里面，使用到了 Koa router，这个 router 已经整理了好几个经典的用法，并且还加了一个 body parser 中间件。

关于 body parser 中间件，要给大家提一下 http 这样一个通信协议它本身是有不同的方法的，这个方法英文单词叫 Method。不同的方法有不同的语意，虽然之前有几个很火的帖子和文章一直在讨论说 http 请求方法其实本质上没有区别，但是它们本身有不同的语意，这些语意有一些仅仅是获取资源而不做任何修改，有一些动作是会产生新的资源，这些新资源出来的一些 http 请求，通常是被称之为那种有修改的请求。

body parser 是专门用来去解析除了 get 请求，就是单纯的获取资源外的处理一些 put 请求，或 post 修改或是新增内容请求的数据 body 的解析中间件，这样的中间件其实用法跟我们刚刚的 router 其实是类似的，它本质上也是会返回一个函数，然后这个函数会传给 Koa，然后 Koa 会把这些 body parser 的中间件，还有 router 这些中间件 compose 到一起，然后让你顺势去使用它。

云上 Koa 模板跟本地跑的 Koa，最大的区别是云上的 APP。我们这个地方因为跟本地稍微有一点区别，所以在云上的 Koa 的应用是需要把你的 APP 给通过 module.exports 给导出来，这样我们的阿里云才能够感知到你的应用，然后才能使用，但是在本地

或者是普通的服务器上跑，大家是不需要做这个操作的。了解了区别之后，我们就会知道如把本地的 Koa 应用迁移到云上的话，最主要的地方是把这个 APP 通过这样一个方式给导出来就行了。

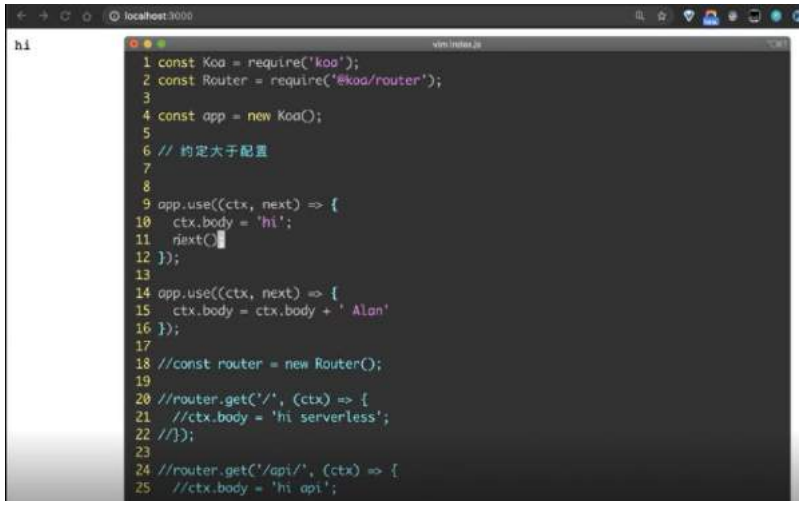
导出来之后，云端的同学就知道有这样一个 APP，它的运行方式跟我们在本地其实是一样的，我们这边的话代码叫 APP.js。在云上运行直接是 Node APP.js。

四、什么是 Koa 中间件?

导出的 APP 会在后台运行应用时被 require。Koa 的中间件把自己定义为普通的 function，然后每一个部分都有它自己的 function，把这些 function 组合到一起，接连运行下去，这是 Koa 对中间件的定义。Koa 中间件是它自己定义的一个扩展方法。Koa 遵循的设计模式是跟 Ruby on rails 比较像的，它遵循的原则叫约定大于配置。Koa 中间件做的约定是一个函数，它有两个参数，一个 ctx 一个 next，然后这样子的一个函数可以作为一个中间件来使用。

由于 Koa 比较流行，所以在 Node 里一般讨论中间件就是指 Koa 中间件。你也可以自己写一个你觉得更好的中间件模型，如果你做的框架流行起来，那么中间件的标准就会变成你的。然后我们在云上跑跟在本地跑区别是差不多的。

（演示）比如说我写一个 body=hi，然后再写一个，我在上面写一个 body=hi，这个约定大于配置是说函数本身的，现在我们开始讨论中间件到底是什么东西，然后我把 ctx.body 加等于相当于一个 Alan，这里就有两个函数，这两个函数都被 Koa 给使用了。



这里的加号写成了等号，这两个函数就都被使用了。这两个函数被组合使用，它一开始是 body 上是一个 hi，然后在第二个函数被使用的时候，这个 body 就变成了一个 hi 加上 Alan。这是一个很简单的把这两个函数组合起来的一个例子。

我们刚刚说了它有一个约定，就是有两个参数一个 ctx 一个 next，要调 next，它才会帮你去调下一个函数。

我们尝试调一下，hi Alan 就出来了。然后 ctx 和 next 这两个参数是 Koa 给你约定的，然后这个 next 是指向下一个函数，那么一个函数套下一个函数，然后再继续套下一个函数，就会发现我们可以很自由的在一个函数从一个 Path。换句话说，比如说我们要实现一个 user register 用户注册一个 API 的时候，在注册之前，我们可以在这个函数数组上面的中间任意一个位置加上一个新的逻辑去过滤进来的请求。

（演示）比如说加一个安全的函数，就会发现所写的函数不管在这中间的任何地方，它其实都处于逻辑中间，所以 Koa 的作者把这种函数叫做中间件。这个顺序的话，谁先被它 use 就谁先出现在这个数组里面，但是你也可以把 APP 上面的数组强行拉出来，然后塞到中间任意一个位置，去做逻辑的处理。

当你去写函数的时候，你会发现这个函数可以很轻易的在你的整个逻辑链路的任何位置去做一个扩展，所以它扩展的时候，它就好像一个中间的东西，所以就把这个东西命名成了一个 middleware，是吧？它并不是一个什么 software，也不是一个 hardware，它是一个 middleware。它可以根据业务模块着层处理，而且你可以把一些逻辑单独用一个函数抽出来，然后去复用，在某些需要使用这个逻辑的路由上面，你再去挂这样一个函数。

总结一下，当我们从本地的 Koa 应用迁移到云上的一个很大的区别，是云上的一个应用需要把 APP 导出来，然后在云开发平台去把 Node 的进程提起来，完成测试预览，再输入你的端口，然后去访问这个端口，才能去预览 Koa 应用。

SSR 框架实现服务端渲染部署上线

作者 | 张宇昂 优酷大文娱技术专家

本篇内容主要介绍 FaaS 场景下的 SSR 框架，也就是在 FaaS 场景下怎么开发服务端渲染页面。服务端渲染是指前后端同构的服务端渲染，即 SSR。

在介绍 SSR 框架之前，首先简单介绍一下 Serverless。

首先是 NodeJS 的定位。在很多公司 NodeJS 都在充当 BFF 层的角色，甚至 NodeJS 只出现在本地开发，仅仅做一些开发工具和构建工具的角色。最主要的原因，可能是因为前端工程师在运维方面的知识比较稀缺，就导致了可能很多公司不敢 NodeJS，不敢让前端工程师做服务端。

但是随着 Serverless 概念的出现，上面提到的问题就很好解决了。Serverless 不需要关注运维，只需要关注自身业务逻辑的开发，因为运维的工作在云开发平台都已经解决了，这对前端工程师来说是收益最大的。

从 BFF 到 SFF 层本质上开发模式并没有改变，但是却大大提升了开发效率。上图列举了在 SFF 场景下，经常做的 5 个方面工作：

第一个， API 接口；

第二个，API Proxy；

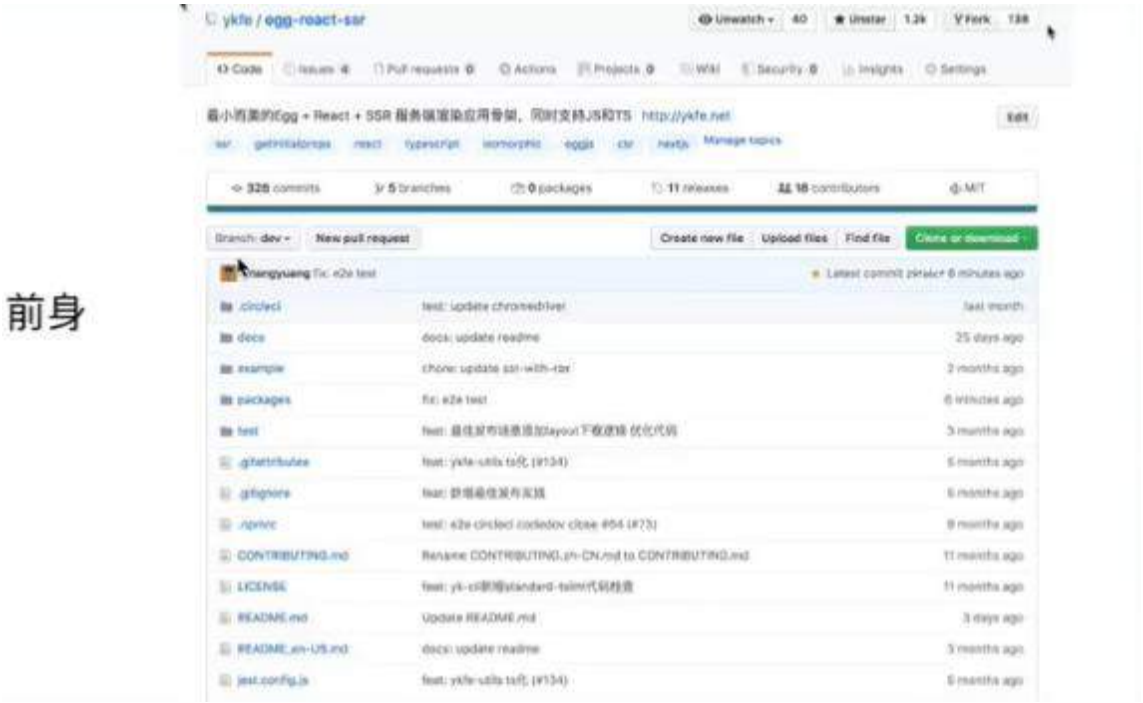
第三个，Render 渲染层。渲染层可以分为客户端渲染和服务端渲染；

第四个，SSR+API，就是如何开发服务端渲染应用，下面会着重介绍在 FaaS 应用里，如何同时开发服务端生产应用和 API 接口；

第五个，Gateway 网关层。

一、SSR 框架前身

SSR 框架是由 Egg+React+SSR 演变来的。这个项目已经开源一年多了，目前大概有 1200 个 star 了。阿里内部已经有七八个部门用于线上应用的开发了，阿里外部也有很多公司正在使用这个项目来开发线上应用。大家对应用开发的整体反馈是，比一些业界类似的框架好很多。



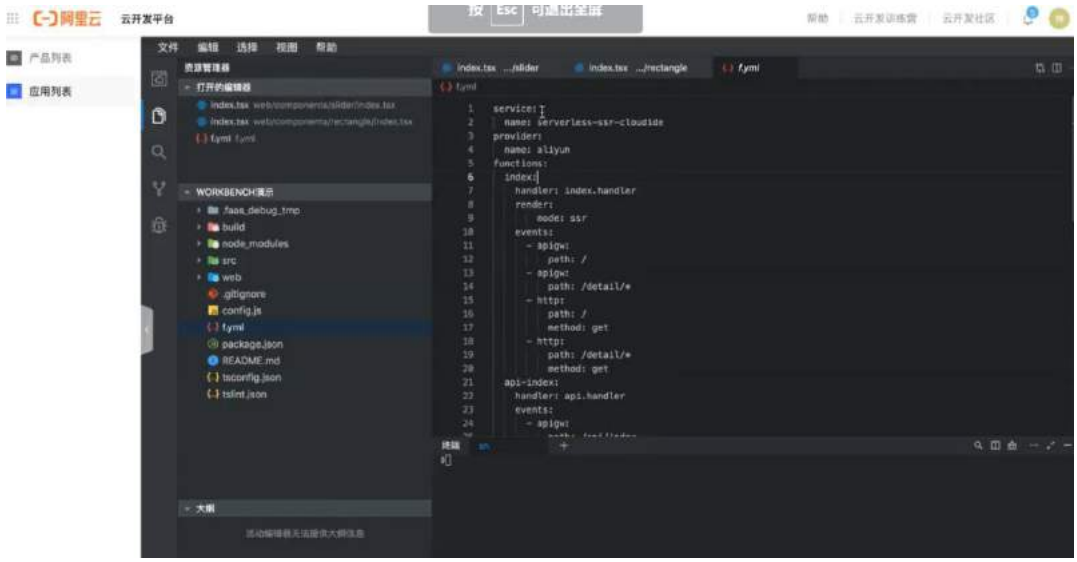
二、FaaS 场景下的 SSR 框架

这个框架针对 FaaS 场景做了很多优化和制定了一些规范。尽量保证在云开发和本地的开发体验保持一致。

三、实操演示 SSR 框架的使用

首先登陆云开发平台 http://workbench.aliyun.com，然后创建应用。在创建的时候的时候，选择 WEB 和 FaaS 场景下的 SSR 框架。创建完成安装依赖，tips 是可以用 cnpm 而不是用 npm 来安装依赖，cnpm 安装依赖要快很多。

如下图可以看到 yml 文件，可以看到和 API 接口看起来差不多。



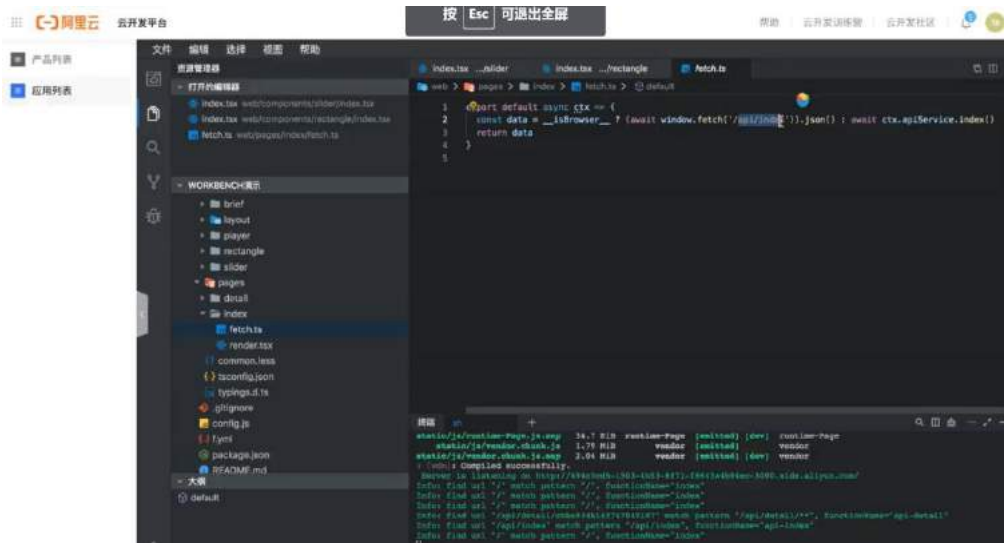
上图所示新增的 render 字段的意思是，当检测到函数里有这个字段，这个函数要返回 htmlStr 界面的，而不是返回 API 服务。index handler 的作用很简单，就是从 ssr-core 中 emloy 的 render 方法，然后再把返回的结果到 htmlStr。

本地开发也是通过 cnpm start 来启动服务，可以直接使用 SSR 框架。启动后进行构建，同时启动 FaaS 服务，最后会返回一个地址，这个地址可以直接在浏览器打开来预览当前的应用。

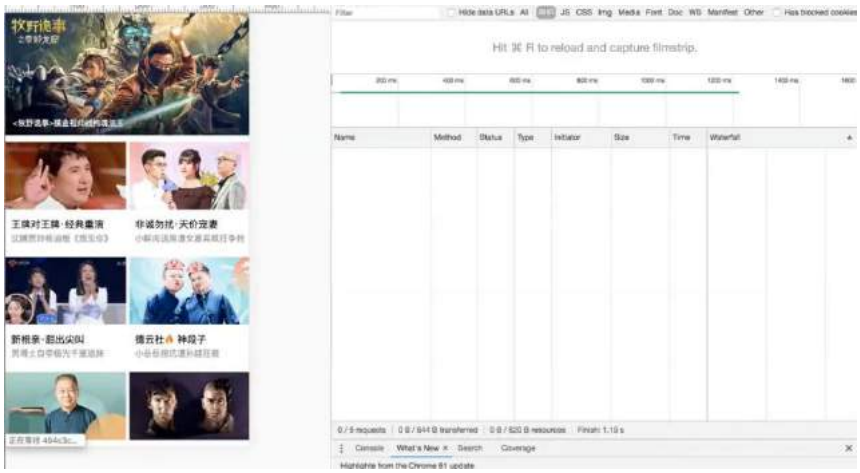
（展示预览）从页面的源码可以看出它包含了一个完整的 HTML 结构，也就是说这个页面是由服务端而不是客户端渲染产生的，点击刷新也不会产生因客户端渲染而产生的白屏。

由根路由切换到详情页，这个切换过程仅限于前端的切换，不会向服务端发出请求。所以详情页的数据其实是由前端发起 HTTP 请求获取的，这个请求也会写在 FaaS 服务里的。包括退回到首页的操作也是前端路由的切换逻辑，只有刷新才是真正访问到服务器。

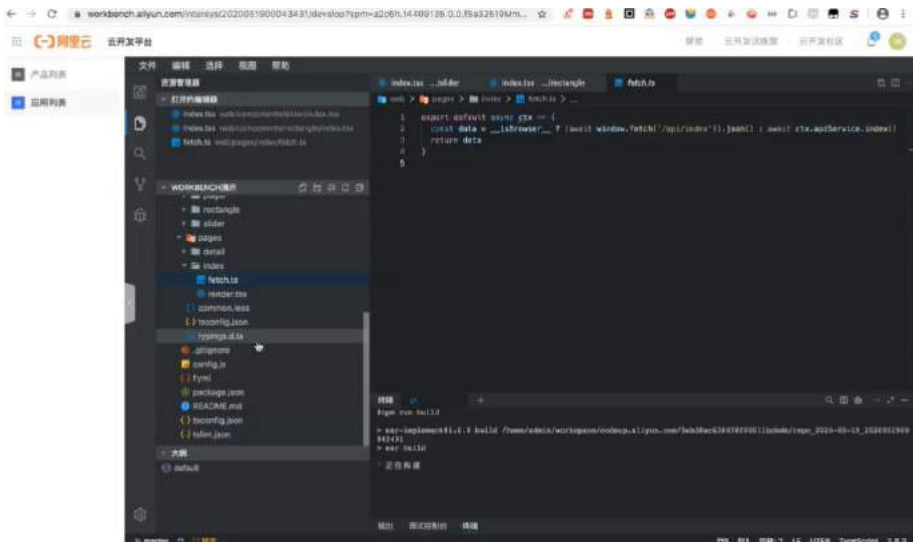
如下图在 f.yml 文件中，api-index 和 api-detail 底下是定义的两个 API 的接口服务。打开 web 目录，就是前端组件存放的目录。打开 Pages 文件夹，这个就是首页组件对应的路由，这里有一个 fetch 文件，它定义了数据是怎么获取的。这里区分了服务端和客户端两个场景的数据获取。再看 API index 函数，它是返回一个 Json 数据结构。如果是在服务端的话，非常简单，可以直接通过 ctx.apiService 拿到自己写的一个 Service 服务直接调用，然后就可以返回正确结果。



SSR 框架还支持客户端和服务端的一键切换的能力。以下图为例，它是服务端渲染，在 url 参数后面加 csr=1，就可以迅速的切换为客户端渲染，切换之后页面的源码就变成空的了。

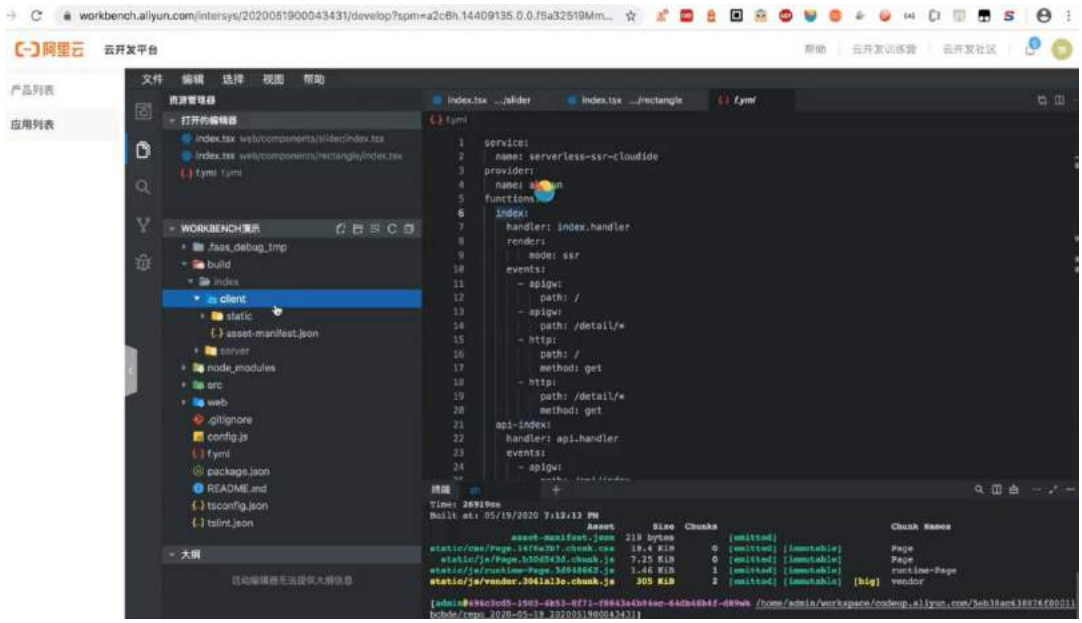


回到控制台页面继续发布，发布之前要先 build，把前端文件进行打包，打包成生产环境需要用到的文件大小。



如果使用 Cloud ID 发布，可以直接通过左边第一个 tag；如果是本地发布，可以直接通过一个命令来发布。

发布时选择日常环境发布。构建之后会放在 build 目录下，然后根据函数作为文件夹的区分。比如，当前渲染层函数叫 index，构建出来的结果都放在 build index 函数文件夹下，分为 client 和 server 两个文件夹。



Client 文件夹之前的课程已经详细介绍过了，这里就不多赘述。看下 server 文件夹，因为是服务端渲染所以多出来一个 server 文件夹，比如 Page.server.js 就是在服务端使用到的文件。

Serverless 场景和传统的 NodeJS 应用的发布场景有什么区别呢？传统 NodeJS 应用发布场景，很多成熟的公司发布构建大部分是在 CI 平台上进行的，比如说本地把代码谱写到仓库之后自动出发这个构建逻辑；但是在云平台发布场景，代码打包是在本地进行的，意思是点了部署后，首先它是会把本地的代码都达成一个 code.zip 压缩包，然后再把压缩包上传，同时对包的大小也是限制的。值得注意的一点，它也会把 node-modules 的文件夹一起打包进去。

所以如果开发应用运行在 Serverless 场景，就需要很清楚哪些依赖是在生产方面用的，哪些依赖是在开发方面用的。如果不注意，打的包就会很容易特别大，超出平台限制。

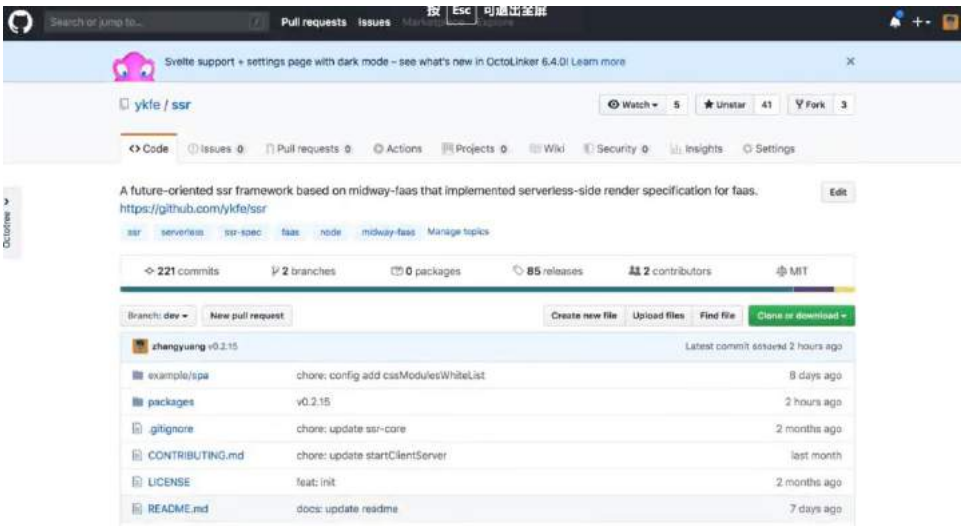
部署成功后会返回一个地址，可以通过这个地址打开预览或申请新域名指到这个地址使用。

总结而言，本地开发只需要关注 start 命令，执行 ssr start；build 就是执行 ssr build 命令。如果是本地开发，也可以直接通过 ssr deploy 一键发布到云端。

四、ssr-spec 规范

这部分主要介绍在怎样的规范下，才能在 Serverless 场景下开发出一个渲染层的页面。

（实操演示讲解）如下图，框架在 github 的地址。



在这个地址可以看到 SSR 的规范。

这个规范主要是基于单页应用和多页应用两个不同类型的应用制定不同的规范，分为 SPA 和 MPA。当下非常常见应用是 SPA 类型的。

首先就 src 目录放一些 FaaS 服务端相关的代码。这里要做的就是在文件夹下新增一个 web 目录， web 目录里放一些前端相关的东西，比如前端组件。

这虽然是为 FaaS 场景打造的，但是与服务端无关。在 web 目录下，上图所示的文件夹结构也可以参考。文件夹里的路由用的是约定式，不需要去手动编写路由配置表。

举个例子，pages 文件夹会放页面的登录，有的也会将根组件放在这里，如果 pages 有 index 文件夹，就会把 index 文件夹映射默认为根路由。

fetch.ts 文件，在这个文件主要做一些数据获取方面的逻辑，比如页面需要在服务端调用哪个接口或者调哪个服务获取数据，然后在文件导出的函数里把它返回出来就可以了。

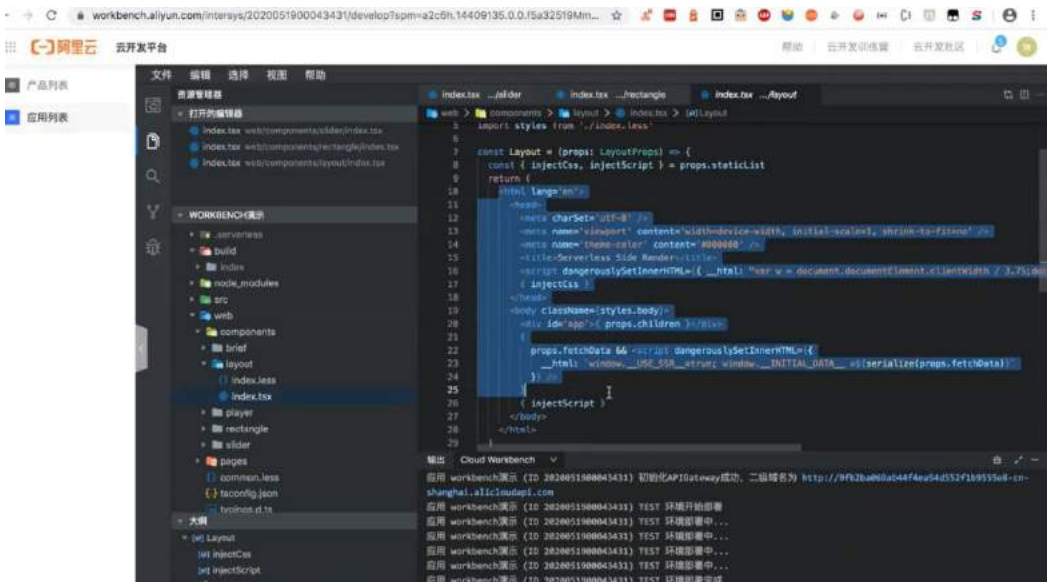
index.less 是样式文件。

render.tsx，定义了它是怎么渲染的，其实就是 render 方法。

components 文件夹存放了一些公共组件

pages 文件夹，如果页面登录有对应的组件就会放在这里；详情页可以放在 detail 文件夹中。

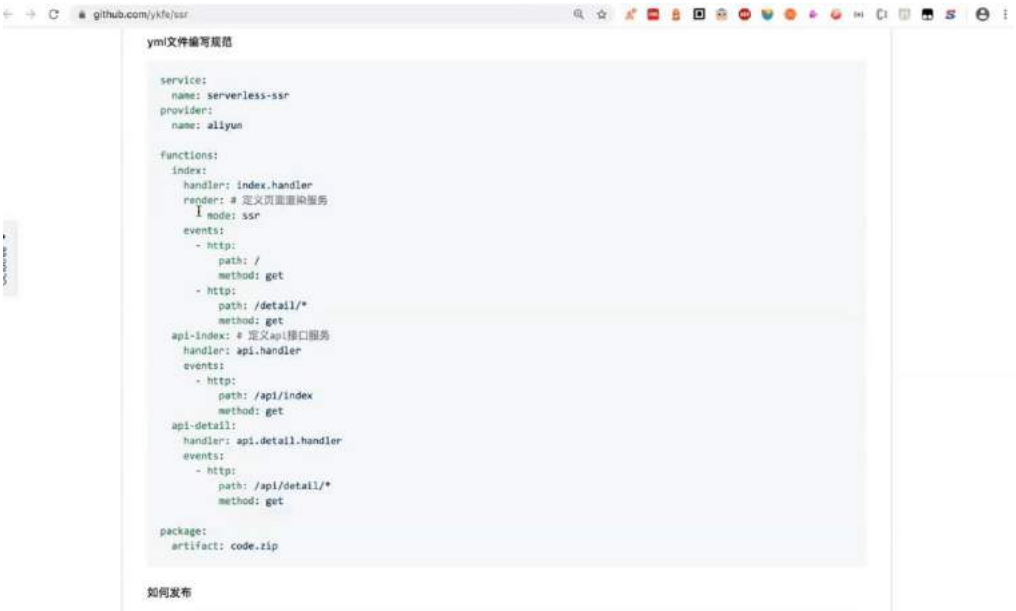
Layout 文件夹，采用的是 html 结构。如果当前开发的是单页面应用，只需要一个 layout 就够了；如果是为不同的路由做不同逻辑区分，可以在下图代码处拿到 context，然后根据请求不同做不同逻辑处理。



五、yml 文件的编写规范

yml 文件编写规范，函数里增加了 render 字段，也就是说，如果检测到这个函数有 rander 字段，就说明这个函数会有渲染服务。

比如 render 有个 mode，mode 默认是以 ssr 模式来渲染的，如果把 ssr 写成 csr 就会把它降级为客户端。举个例子，页面可能会有些极端情况，当发生时页面在服务上会报错，这时候可以通过改 yml 文件来为客户端渲染。开发接口服务，只需要 handler 和 events 是这两个字段就可以了。

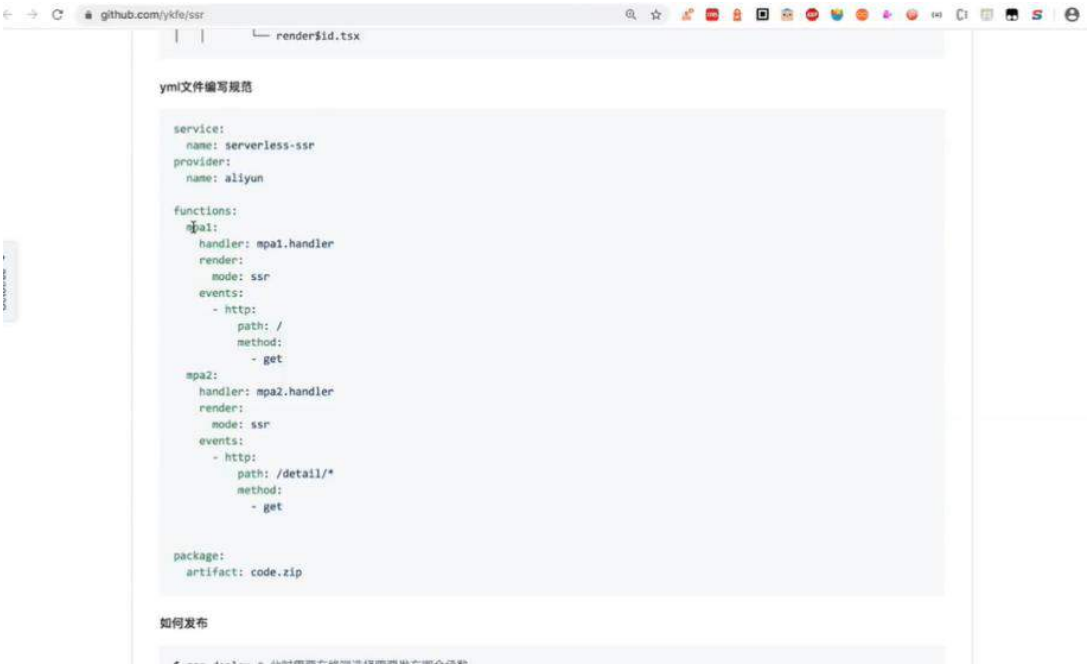


发布这里对接的是 Midway FaaS 的发布功能，如果在云平台的话，直接点 tab 按键就可以发布。

六、MPA 的页面规范

开发多页面应用，跟单页面应用总体上比较类似，区别是多页面应用有不同的 layout，每个 pages 文件夹下面都是一个独立的页面和应用。如果不给每个文件夹添加独立 layout，那么就是用默认的 layout。

构建出来的 build 目录也会根据 yml 文件，基于多页和多函数构建出对应数量的文件夹。如下图所示，yml 文件如果有多个函数，每个函数都定义了一个 render 字段的话，这就需要开发一个 MPA 多页面应用。



使用 MPA 多页面应用很简单，只需要按照上文介绍的规范，添加一个 web 目录，然后从 ssr-core 模块里倒入 render 方法，再把请求的.ctx 传进去，就会自动渲染组件，再把渲染好的结果返回。

展示形式

http://ssr-fc.com/ -> mpa1 函数 -> 渲染mpa1文件夹下的render组件
http://ssr-fc.com/detail/* -> mpa2 函数 -> 渲染mpa2文件夹下的render组件

渲染函数

1）在 FaaS 函数里

在 FaaS 函数里，只需要调用ssr-core提供的render方法传入ctx即可

```
import { render } from 'ssr-core'

async handler () {
  try {
    const htmlStr = await render(this.ctx)
    return htmlStr
  } catch (error) {
    return error
  }
}
```

根据f.yml或者query来判断当前渲染模式

2）在 Node.js Web 框架里

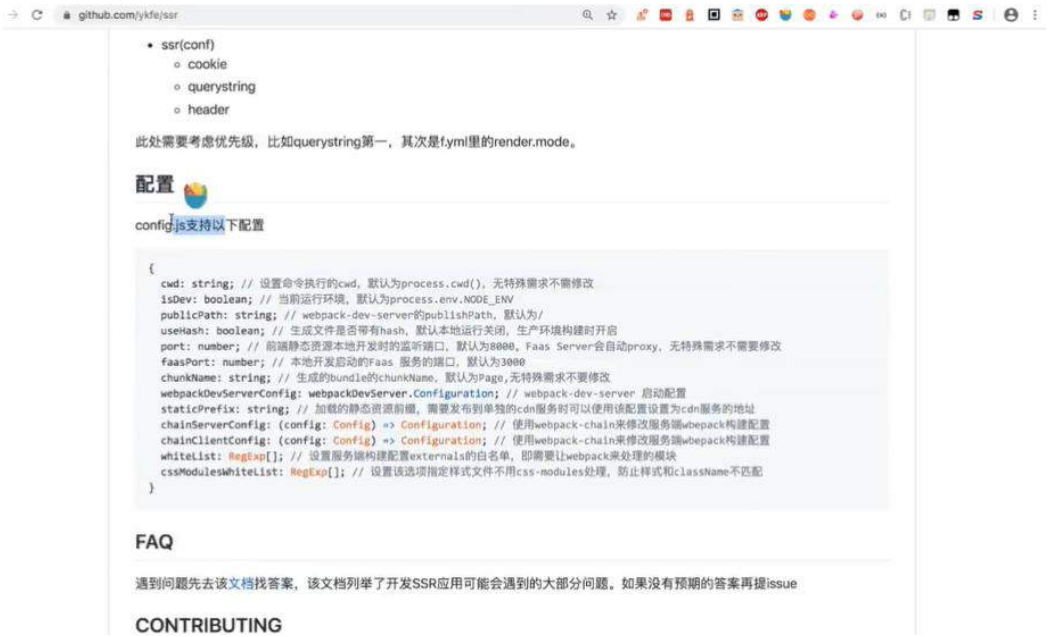
该渲染方式实现是服务端无关的，理论上可以支持任何Node.js框架只需引入render方法以及有一个web目录，用法与Faas函数保持一致。

```
const Koa = require('koa');
const { render } = require('ssr-core')
const app = new Koa();
```

七、模式切换

降级也有多种方式。

为了更加灵活，如下图，支持 config.js。如果想改默认的配置或端口等，都有对应的配置可以更改。这个框架集成了很多流行的前端 UI 框架，不需要再为集成框架做额外配置了。



- ssr(conf)
  - cookie
  - querystring
  - header

此处需要考虑优先级，比如querystring第一，其次是f.yml里的render.mode。

配置

config.js支持以下配置

```
{
  cwd: string; // 设置命令执行的cwd，默认为process.cwd()，无特殊需求不需修改
  isDev: boolean; // 当前运行环境，默认为process.env.NODE_ENV
  publicPath: string; // webpack-dev-server的publishPath，默认为/
  useHash: boolean; // 生成文件是否带有hash，默认本地运行关闭，生产环境构建时开启
  port: number; // 前端静态资源本地开发时的监听端口，默认为8000，Faas Server会自动proxy，无特殊需求不需要修改
  faasPort: number; // 本地开发启动的Faas 服务的端口，默认为3000
  chunkName: string; // 生成的bundle的chunkName，默认为Page,无特殊需求不要修改
  webpackDevServerConfig: webpackDevServer.Configuration; // webpack-dev-server 启动配置
  staticPrefix: string; // 加载的静态资源前缀，需要发布到单独的cdn服务时可以使用该配置设置为cdn服务的地址
  chainServerConfig: (config: Config) => Configuration; // 使用webpack-chain来修改服务端webpack构建配置
  chainClientConfig: (config: Config) => Configuration; // 使用webpack-chain来修改服务端webpack构建配置
  whiteList: RegExp[]; // 设置服务端构建配置externals的白名单，即需要让webpack来处理的模块
  cssModulesWhiteList: RegExp[]; // 设置该选项指定样式文件不用css-modules处理，防止样式和className不匹配
}
```

FAQ

遇到问题先去该文档找答案，该文档列举了开发SSR应用可能会遇到的大部分问题。如果没有预期的答案再提issue

CONTRIBUTING

NodeJS 及 Python 主流框架部署上线

作者 | 风驰 阿里巴巴高级技术专家

本篇内容主要分享如何通过 NodeJS 和 Python 两个开发语言中常用的框架完成开发应用，如何将已经开发的应用部署到云。

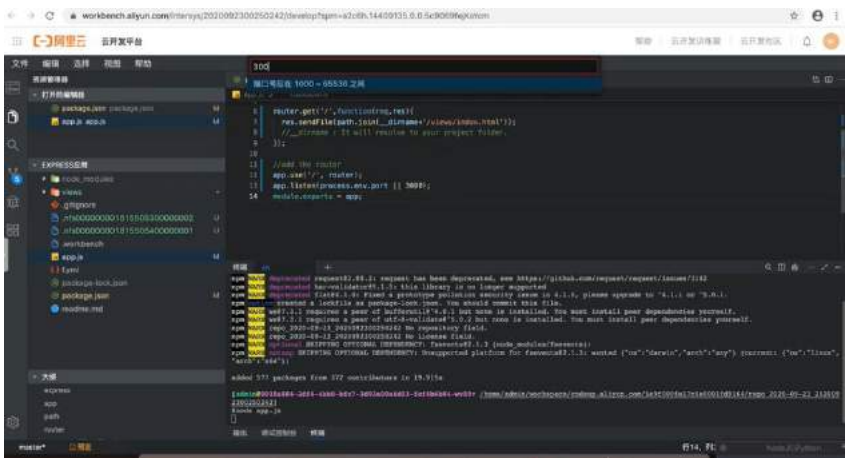
一、如何通过 NodeJS 主流开发框架将应用部署到云

1. 基于 express 应用迁移方案的演示

首先创建应用，选择 express 应用迁移方案，然后等待代码仓库初始化，初始化完成，点击开发部署进入到在线的开发环境中，左侧就是初始化的代码包。然后将已经开发好的 express 应用直接拖拽到 Cloud IDE 的文件列表的根目录。接下来是添加依赖，在 package.json，找到 Midway 的开发依赖，复制出来再打开 Cloud IDE 目录中的 package.json，添加保存。

如果 app.js 是主应用的入口文件，那么需要复制帮助文档里 app.js 的内容到开发平台，将入口暴露出来。然后安装依赖，依赖安装后，进行调试，直接在终端里输入 app.js 的指令。

如果想要预览，可以点击左下角的预览，然后在弹出的输入框里输入端口号，把 3000 端口映射出去，成功后，点击右下角的访问。



调试结束之后如果没问题，就可以去部署。点开 IDE 左侧第一个 tab 的 workbench 部署插件，然后选择部署环境，点击确认就完成了部署。

需要注意的是，基于 MidwayServerless 做 express 应用迁移，默认会把 app/和 config/这两个目录打包进去，如果你的应用需要将其他的目录也一起打包，需要在 f.yml 中添加如下的配置，比如 util 等。

点开 workbench 部署插件，选择日常环境，然后点部署。部署完成平台会给分配一个二级域名，用于访问部署成功的线上结果。所以通过 MidwayServerless，我们是可以很方便的把存量的 express 应用迁移到云上的。

2. 基于 NodeJS 存量应用的迁移方案演示

首先创建应用，选择 NodeJS 存量应用迁移方案，然后补充名称等信息，点击完成。

代码仓库初始化完成后，把应用拖进去，然后打开 package.json 添加开发依赖，添加 mime 和 request 两个依赖，然后安装依赖。下一步是，配置应用的入口将 Serverless_config_common_framework 重命名为 serverless_config.js。重命名之后配置框架和应用入口文件，然后完成开始测试。

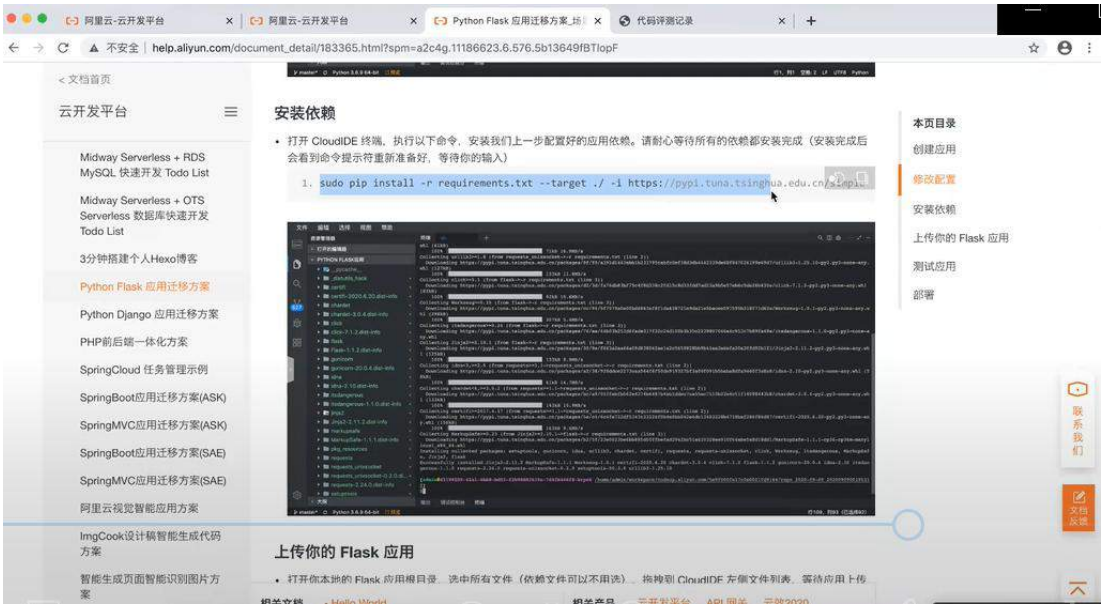
测试打开 Web 插件，点击测试并勾选预览模式，启动测试。测试没有问题，就可以直接部署了。部署成功也会分配一个二级域名，我们通过域名打开线上地址，可以查看结果。

二、如何通过 Python 将应用部署到云

1. Python Flask 应用迁移方案为例来介绍。

参考帮助文档的步骤，首先创建应用，并选择 Python 前后端一体化应用方案，然后等待代码仓库初始化。初始化之后进入到在线开发环境，打开 README.md，先了解操作提示。然后添加 Flask 修改配置，保存，然后将 serverless_config_flask.py 重命名为 serverless_config.py，接着配置入口。

配置入口之后，安装依赖。复制帮助文档安装依赖下的代码，然后打开云开发平台，点击终端进行安装。安装完依赖，再上传 Flask 应用，最后测试，测试没问题就可以点击部署，完成部署到线上的操作。



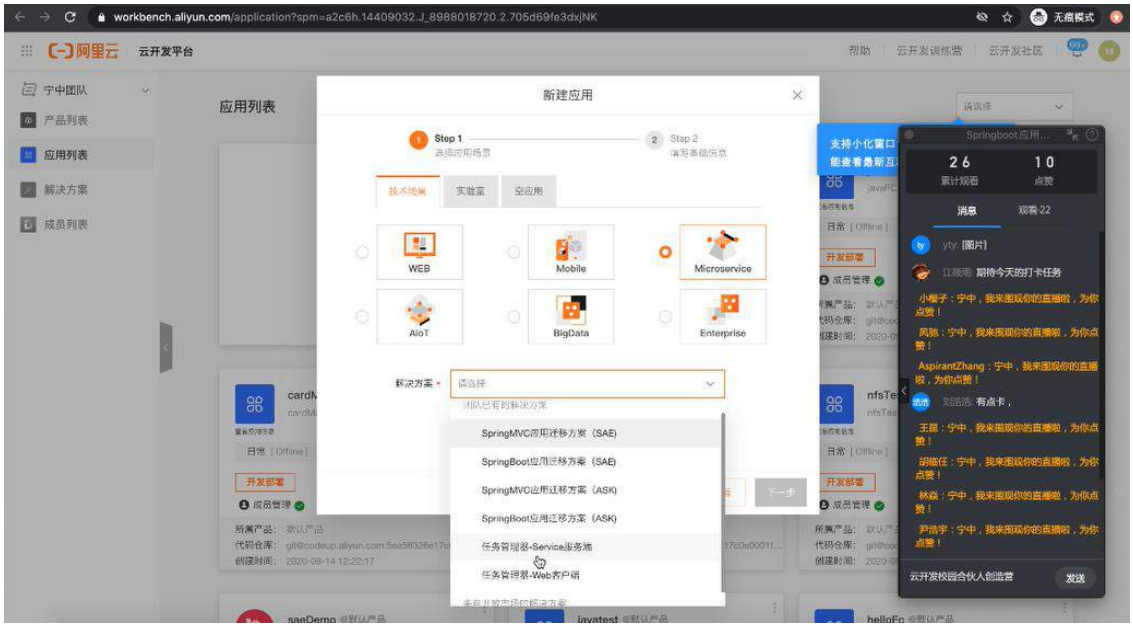
以上就是全部关于如何基于 NodeJS 和 Python 主流框架开发和部署的介绍。结合上一篇的运维相监测内容，如何分配几个环境的域名，如何绑定 CDN 加速，如何绑定 SSL 证书等等，大家就能完全掌握如何基于云开发 Serverless 管理高流量的应用了。

Springboot 应用部署上线

作者 | 宁中 阿里云高级技术专家

本篇将分享如何将 SpringBoot 应用通过云开发平台部署到服务型 Serverless 上；如何通过一套代码，协同支持线下环境、线上环境和云端开发测试环境的开发。讲解过程将通过一个任务管理系统如何搬迁并部署到 Serverless 上来演示。

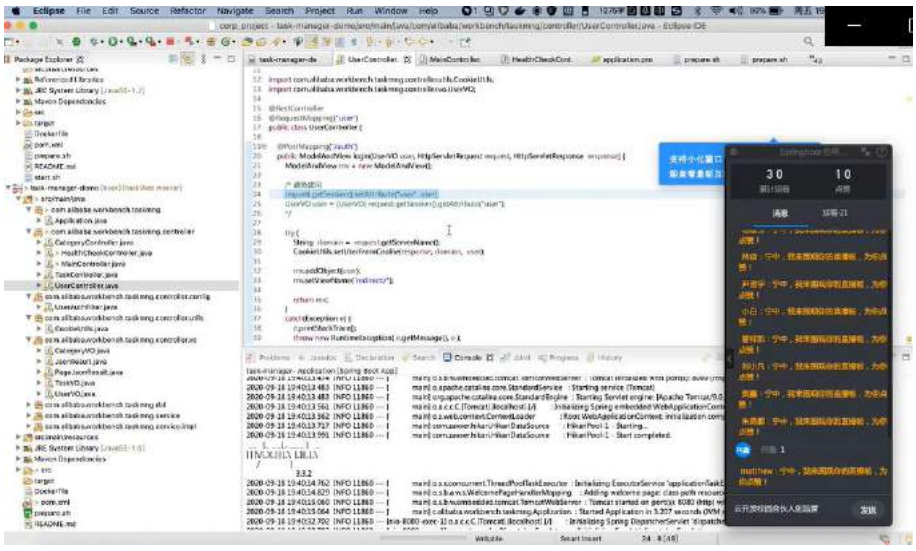
首先打开任务管理系统的登录页面并登录，这是一个分布式的任务管理系统。我们看看如何将任务管理器-Service 端和 Web 客户端整合成一个应用，并应用部署到云开发平台上。



在任务系统页面右侧添加一个任务，左侧任务列表的底部就会显示新添加的任务。

一、确认此应用是不是云原生的应用，是不是无状态的应用。

怎么来保证它的无状态呢？那就要确认它的 Session 不会保存在单台主机上。通过代码我们可以发现，比如常用的登陆态，登录态不能通过传统的单机模式把 Session 设置到 Web 容器的 Session 的属性里去，如果这样的话它其实是存在这台主机的服务端，这台主机服务端就相当于有一个状态了。



在 Serverless 上，主机随时会漂移或随时会扩容。这个时候如果用户登录到其它主机上，它的登陆态就会丢失。所以迁移的前提是把应用有状态的特征变成无状态，怎么把它变成无状态呢？有两种方式。

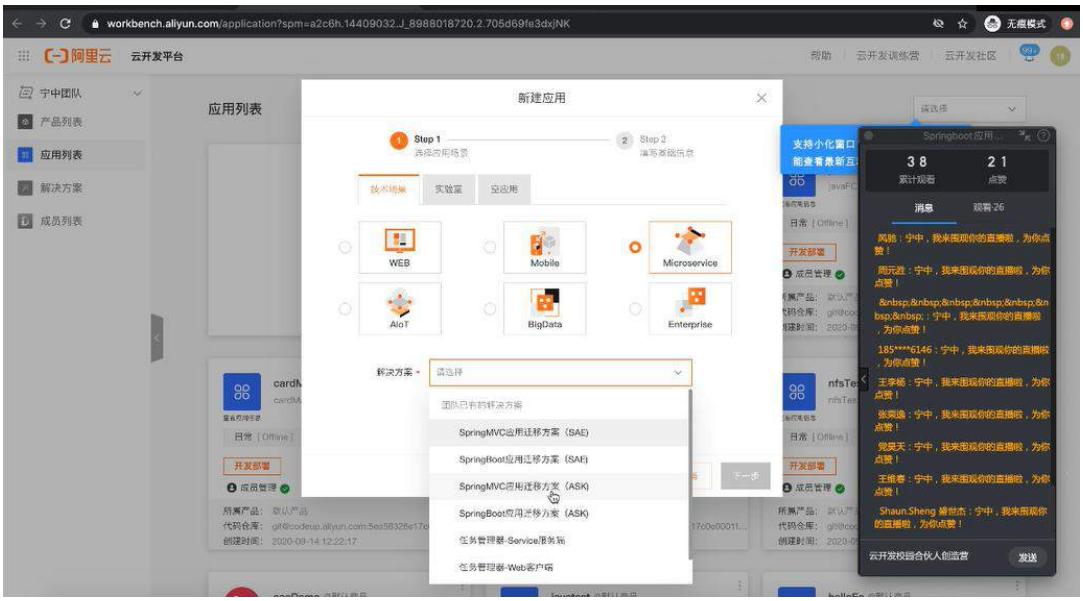
第一种，把登录态信息存储到 cookie 上，让浏览器保存。

第二种，在大型应用场景，登录态把它存储到共享服务上。共享服务是指数据库或 Redis 缓存，一般是存在缓存里而不是存数据库里的。或者是专门启动一个保存用户登录态的服务，设置一个专门的微服务管理，这是大型应用场景的做法。登录态里面可能就存储在用户的一些权限信息、组织信息、一些状态、或是具体应用的信息，都可以存在这里。

一般轻量级的应用，都是通过存储到 cookie 上，cookie 在浏览器和服务端来回传，它的登陆态就都保存起来了。这样用户每次访问时是无感的。

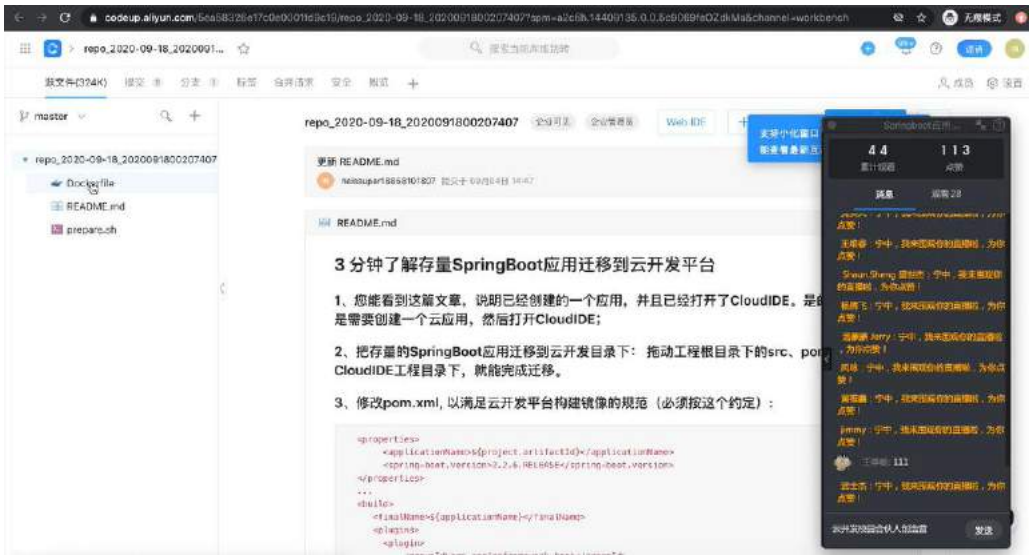
二、怎么通过云开发平台把应用迁移并部署到 Serverless 服务上。

在云开发平台创建迁移项目，点击创建后选择技术场景下的微（Microservice）服务，然后选择 Springboot 应用迁移方案 SAE。

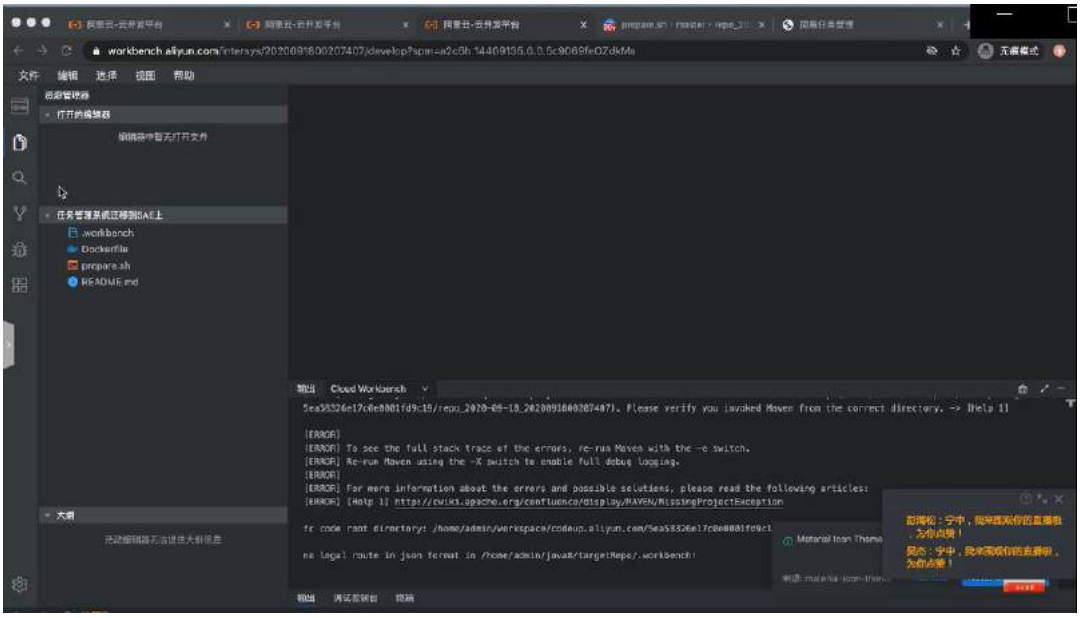


关于解决方案，我们展开说一下。云开发平台上有三种 Serverless 计算服务，一种是函数计算，之前的篇幅已经介绍过了；另外两种是服务型的 Serverless，它包含两种计算服务方式：a 是轻量型的微服务引擎 SAE 即 Service Application Engine； b 另外一种更适合大型应用场景的 ASK 即 Application Serverless Kubernetes，这个也是按量计费的服务方式。这两个典型服务型 Serverless 都是免运维的，意思是它能自动伸缩自动弹性，宕机自动扩容，不用评估流量，不用担心不够用。

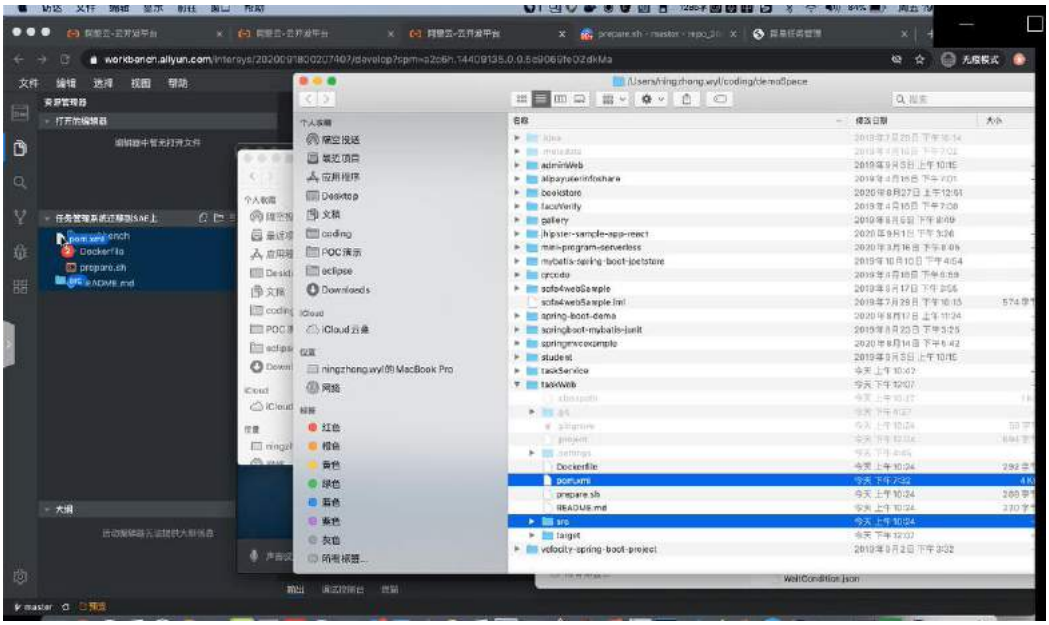
选好解决方案后，补充应用名称和介绍，确定完成创建。项目启动后，它会自动生成与项目对应的代码仓库。点进去会发现代码仓库里面包含三个文件：Dockerfile、README.md 和启动脚本生成文件 prepare.sh.



回到云开发平台创建的应用页面，点击开发部署打开 Cloud IDE。

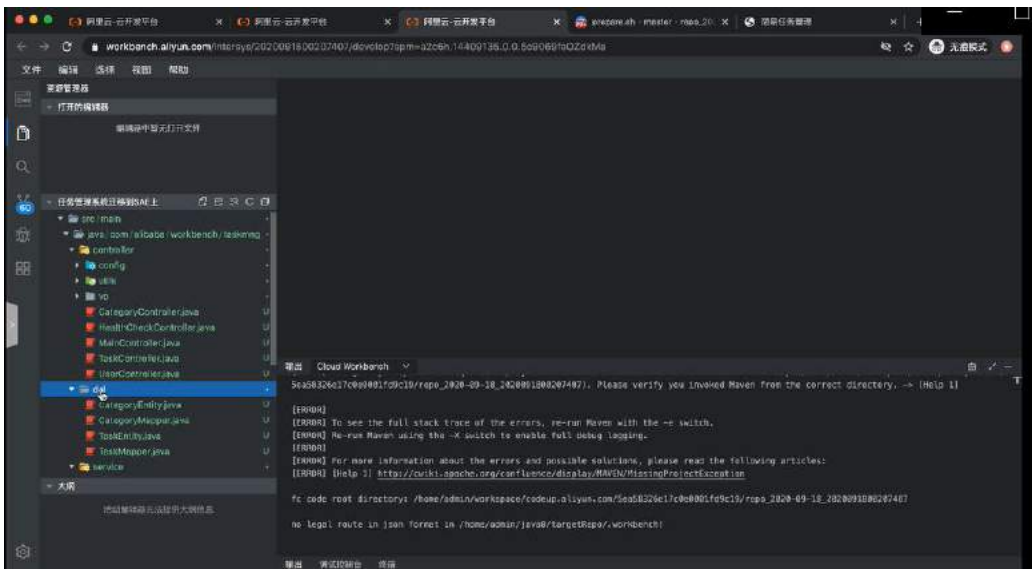


把本地的代码，迁移到打开的 IDE 文件目录中去，也就是部署到项目中去。这个操作有两种方式，方式一是把项目文件直接拖放到 IDE 代码目录中。



另外一种方式更简单。大家把 codeup 代码仓库 checkout 出来到本地，然后再把这个代码拷贝到相关目录下提交到 codeup 代码仓库，然后再在 Cloud IDE 中把代码拉取到 IDE 工程目录中。

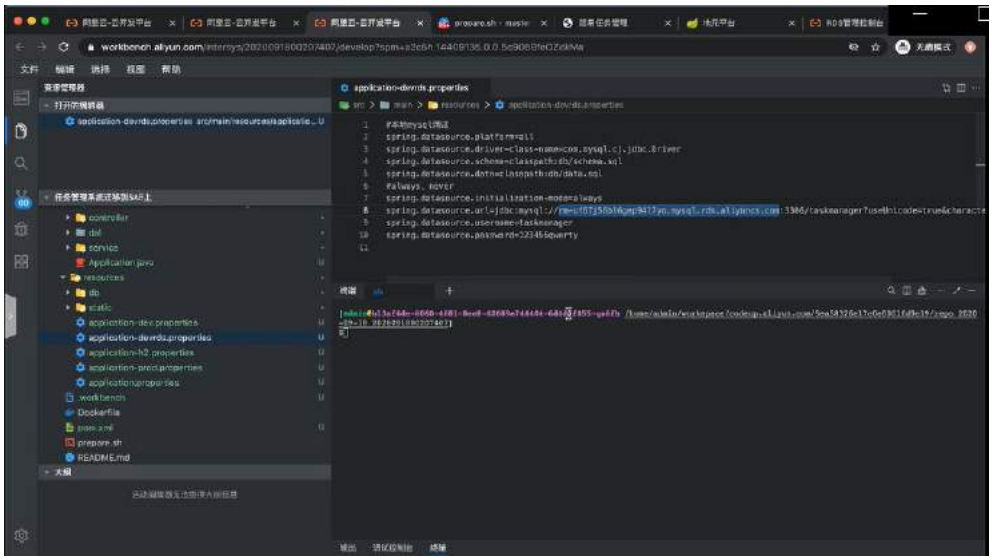
拖完代码后，从 IDE 页面代码目录结构可以看到这是一个很典型的应用。目录中包括控制层、服务层、数据访问层等等。



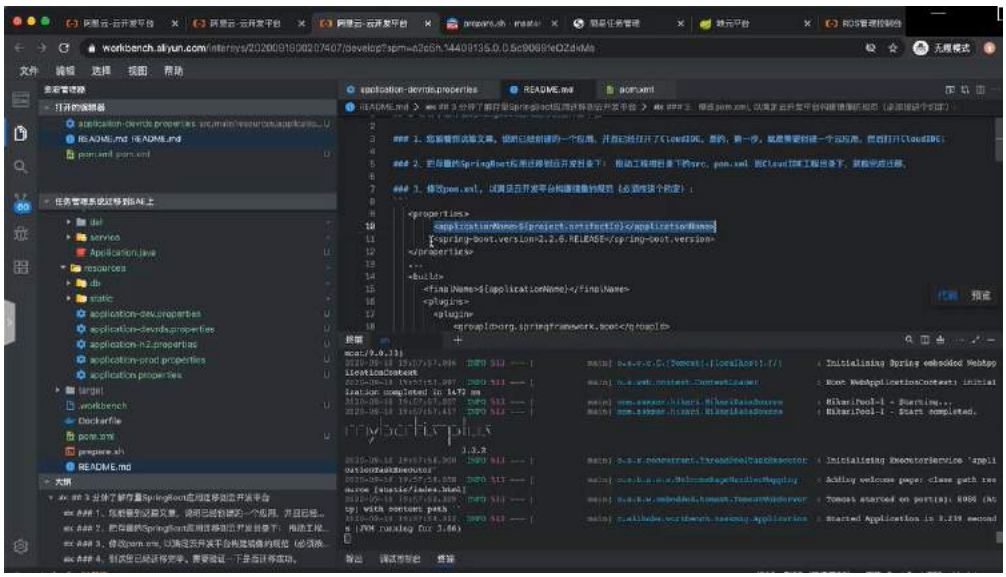
完成代码设置，可以实现应用的本地启动。尝试 CloudIDE 云端启动的时候，要确保应用连接到云端的数据库。方法就是，提前在云端申请一个数据库，并做好配置。

三、怎么部署应用?

首先在云端准备一个测试数据库，这个数据库是外网能访问的 rds 的数据库。如果我们在阿里云上买了数据库，那么查看实例的时候，会看到每个数据库都有一个内网地址和外网地址，外网地址表示在外网通过地址可以访问到这个数据库的。我们在 Cloud IDE 里面配置就是的这个地址。

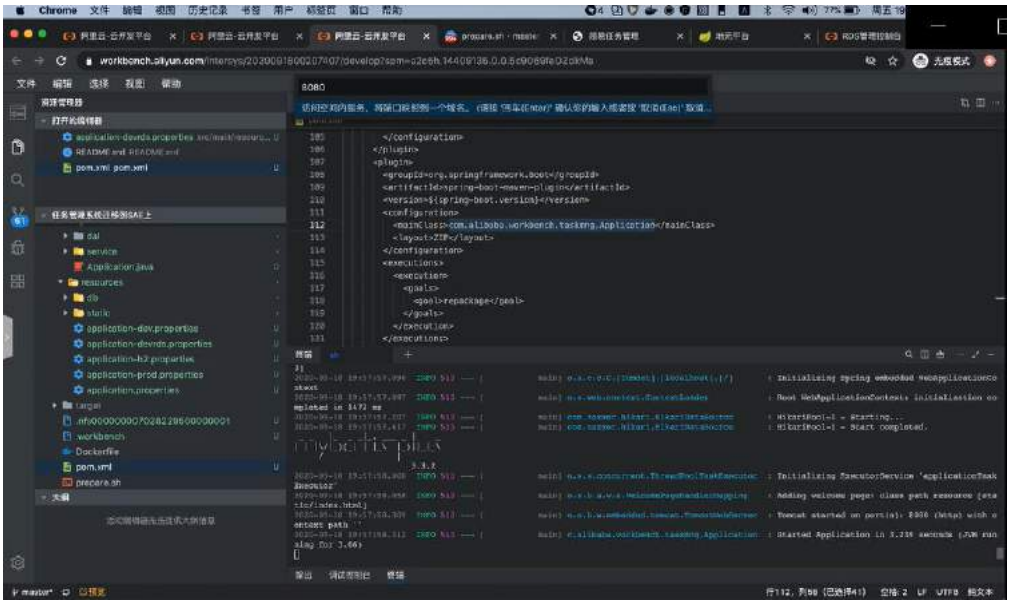


启动的时候 Spring-boot 会下载很多依赖，说明 Cloud IDE 这个容器它是能访问外网的。然后打开 README.md，按照提示的第三步修改 pom.xml 文件，让 pom.xml 文件符合迁移的约束，这个是必须的。然后将 Springboot 打包，将启动类打成启动包。

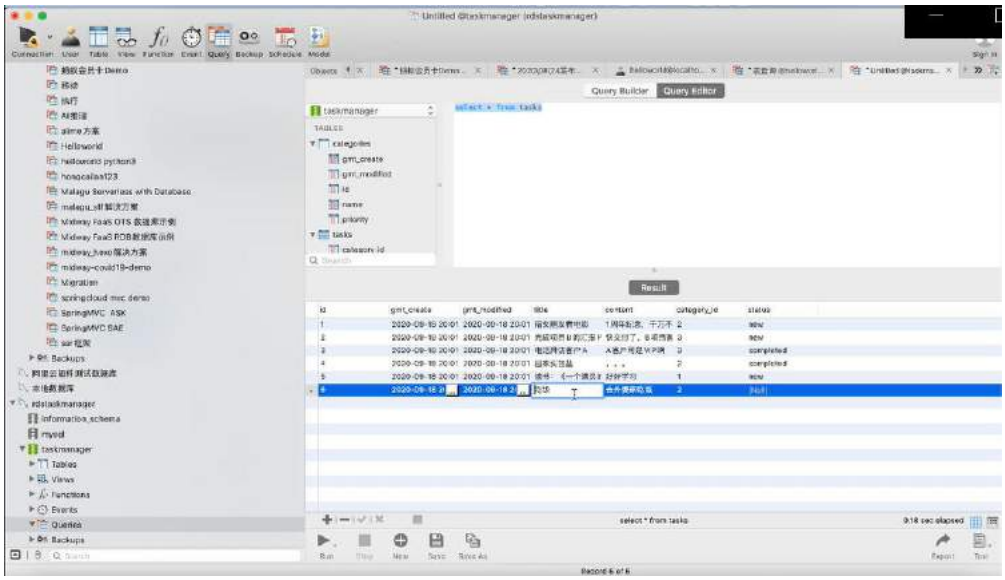


四、当本地启动成功，如何访问和调试?

在 Cloud IDE 页面左下角点击预览，并把 8080 端口映射出来并点击访问，就可以访问服务了。

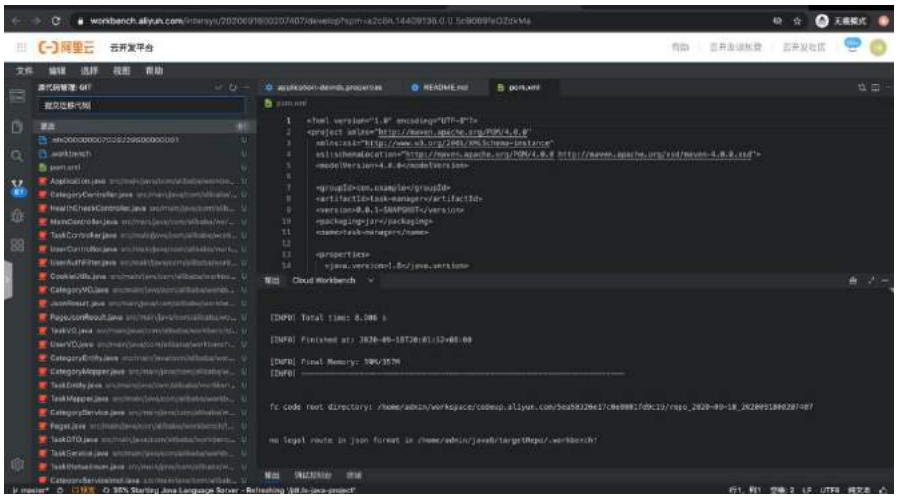


这个登录页面是访问容器的，登录进去后，进入到任务管理界面。任务管理系统里面建了两个表，如果在主页面添加数据，表格中也会对应的添加进来。在没有电脑的情况下，使用一台平板电脑打开这个浏览器登录 Cloud IDE 界面就可以写代码，这是浏览器在线编程的方式。

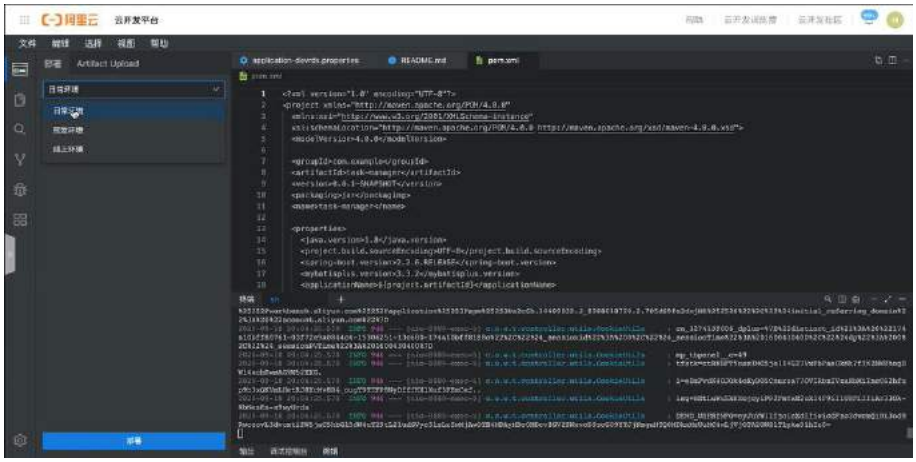


五、部署操作详解

部署到云端的时候一定要提交代码，因为云端去集成构建的时候，Flow 集成环境是通过代码仓库去拉代码。

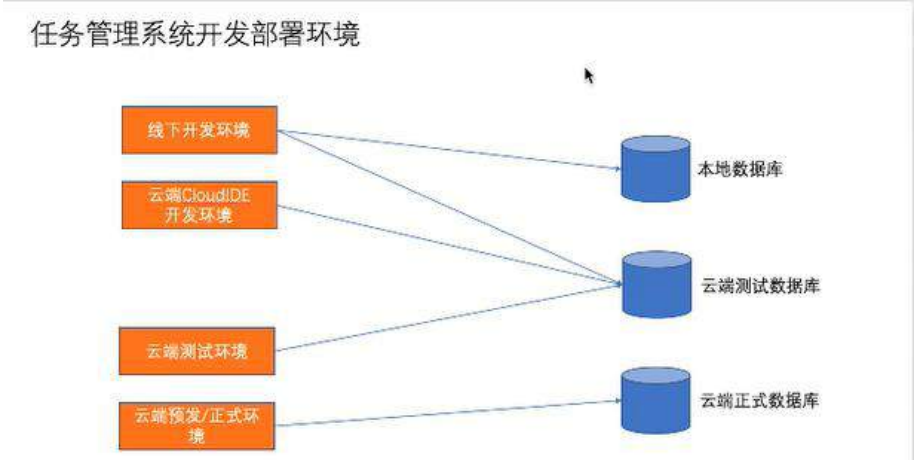


应用部署，系统默认有三套环境，一个日常环境，一个是预发，一个是线上。我们来讲解部署到日常环境的情况。



六、如何使用这三套环境

线下开发环境，测试环境、预发环境和正式环境的需要在代码配置层面保证一致。比如我们有三个数据库，一个是本地数据库，一个是云端测试数据库，一个云端正式数据库。我们的工作环境会经常在这几个环境中切换，对应的数据库连接如下图。



在本地 IDE 中写代码的效率是比较高的，在云端 Cloud IDE 开发环境更方便联调代码。当遇到没有自己电脑的情况，或者修改简单 bug 的时候，会用到云端的 Cloud IDE，在上面可以实现写代码。未来，云端开发体验基本上可以做到和本地 IDE 开发体验一致。如果能达到这样的效果，未来所有的开发都可以在线上实现。现在主要还是通过本地开发环境，写好代码后提交到代码仓库里完成项目的实现。

开发完毕，需要部署到测试环境进行验证，这就需要部署到云端的测试环境上，这个测试环境可能会连接到测试数据库，或者单独有个专门的测试数据库，尽量跟开发隔离。

测试环境完成后，基本上就可以把这个应用发布到线上了。发布之前会有一个预发的环境，这个预发环境只是没有对外提供服务而已。这个预发，内网可以访问而外部客户是访问不到的。预发环境都是连的正式数据库。预发也没问题了，就可以发布到正式环境，直接向外提供服务了。

部署完成，我们通过日志可以看到整个过程。

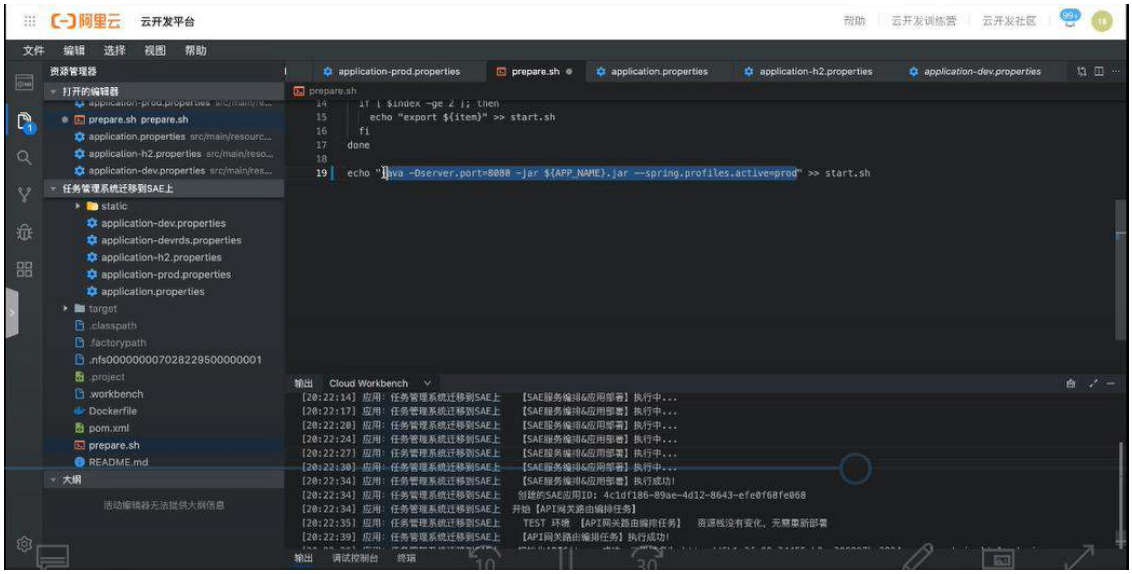
第一步，Java 构建申请运行环境，然后在 Flow 容器里去构建编译代码，编译代码完成是去打镜像，然后把镜像上传到镜像仓库里面去。

第二步，编排 VPC 的网络，编排网络就是要构建一个基础设施的环境，就是一个虚拟的网络。如果访问一个存量的应用，存量的数据库或者是存量的 Redis 或者中间件服务，这些都需要编排在同一个 VPC 里面，才能访问、才能互联互通。

第三步，SAE 的三套环境初始化和 SAE 服务和编排部署。部署完成，就去绑定域名。域名对我们普通开发者个人来说，是个比较高的门槛。我们考虑到这一点，阿里云云开平台会给开发者默认创建了一个域名，但是这个域名只有半个小时生效期。

第四步，测试环境完成后就可以发布到线上环境，不过在发布前要前往预发环境验证，验证发布是否正确。

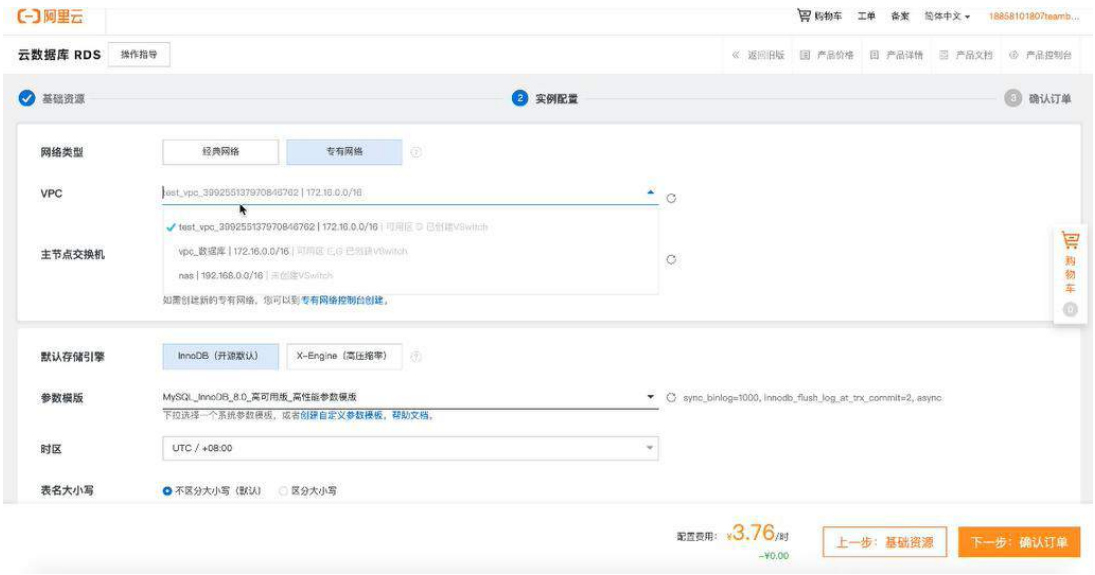
第五步，推送线上发布。在部署线上的时候，prepare.sh 生成的脚本要调整一下，原来指定的正式环境要把它改成 prod，相当于将容器镜像脚本启动。端口一定是 8080，即使本地开发时使用其他端口，部署上线时候也一定要改成 8080 端口。



展开讲一下 Spring 配置文件。这个文件下的 application-dev.properties 可以链接本地数据库；application-devrds.properties 连接的是远程测试数据库，这个是在线开发模式；application-h2 自己没有数据库，可以使用内存数据库；application-prod.properties 是正式数据库，一般是在 VPC 里购买的数据库。

七、如何购买数据库?

购买数据库的时候要选择 VPC 和交换机，并选择网络，这个网络跟应用要保证在同一个 VPC 里。



八、课程汇总

把一个 Springboot 迁移到云开发平台。

第一步，要把当前应用，无论是本地开发还是云端开发，变成无状态的；

第二步，把代码搬迁到云开发平台上，云开发平台会给你创建一个代码仓库；

第三步，根据迁移工程里面的 README.md 或者云开发平台上帮助文档的说明，修改 pom.xml 文件，加上代码和配置，再加上 Springboot 打包的插件，提交代码；

第四步，提交代码然后点部署，部署的时候它是只有一个环境的，只有一个 prod 。线上的配置相当有两个层级，第一个层级是区分开发模式；第二个层级是区分三套环境，测试环境、预发环境和正式环境；

第五步，测试环境测试验证，最后部署到正式环境。

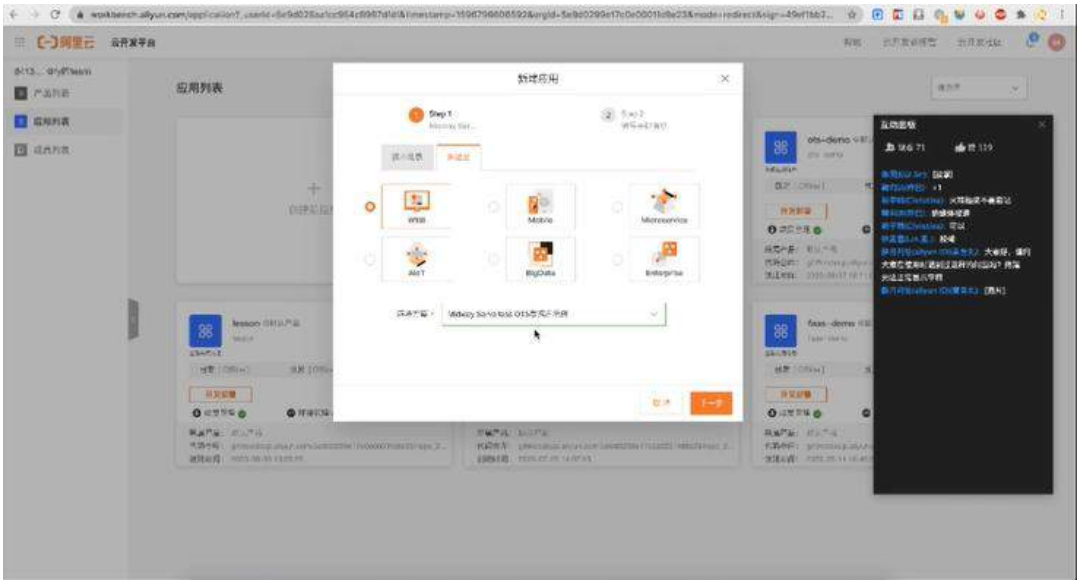
第三章 云开发后端篇

如何访问云 OTS 数据库和实践

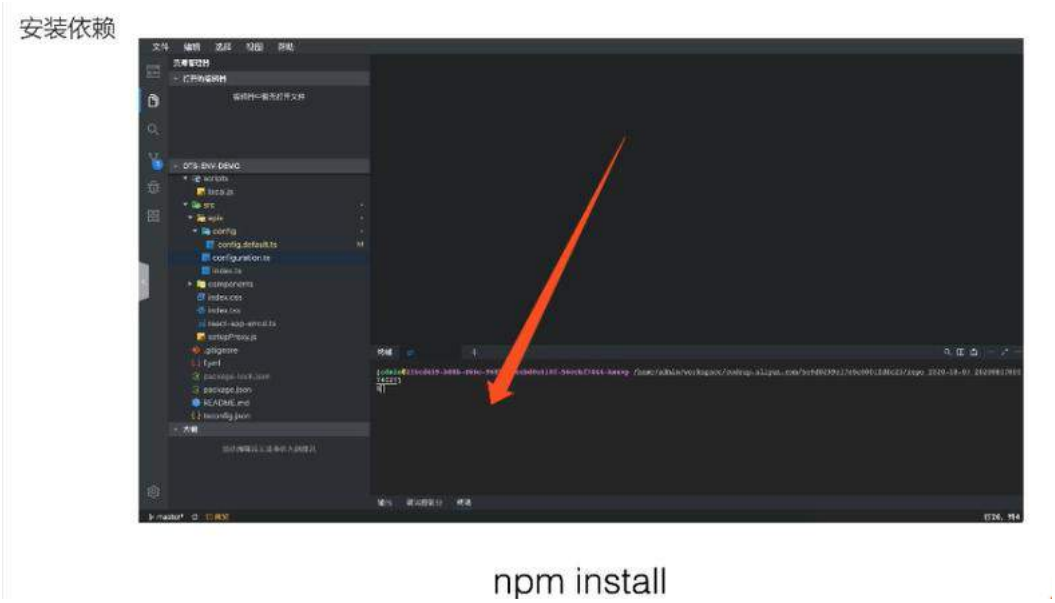
作者 | 繁易 淘系 Node 架构技术专家

基于云开发系列课程的深入，本篇内容将为大家介绍在实际的应用开发中如何把数据存储到数据库，还有我们如何从数据库里快速调取需要的数据。希望通过此课程，大家可以快速学习到 Midway+数据库的一个操作。

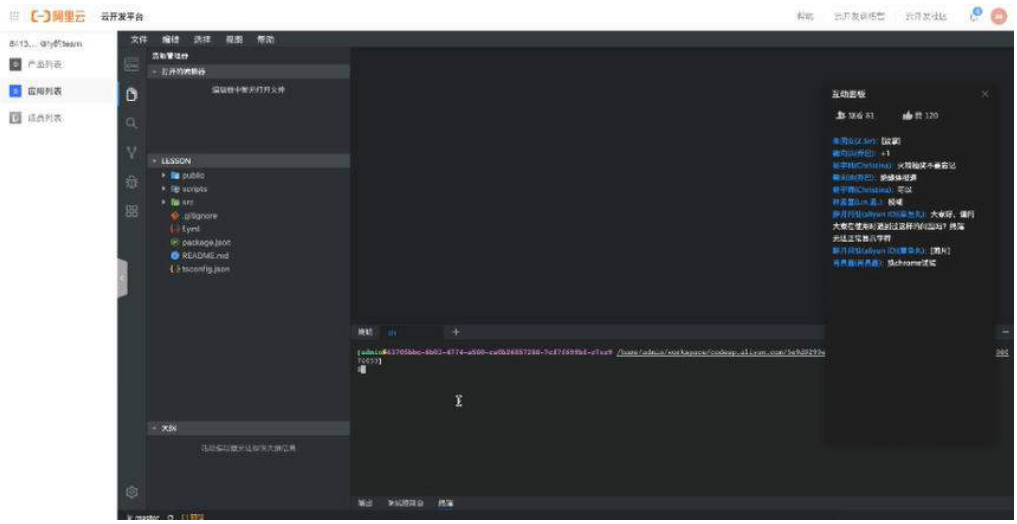
一、如何创建 OTS 数据库?



首先打开云开发平台，创建新的应用，选择“实验室”场景，然后选择 WEB 场景，下拉条选择开发市场的解决方案，就叫 Midway Serverless OTS 数据库示例。选择之后直接进入下一步，填写好名称等相关信息，完成应用创建。等应用初始完成后，点击开发部署进入应用。在初始化空间之后，我们要做的第一件事情就是安装依赖。



进入到云开发平台界面，点击终端按钮，输入 npm install 命令，然后安装一些相关的依赖，这些依赖都是我们启动服务这边会需要用到的。



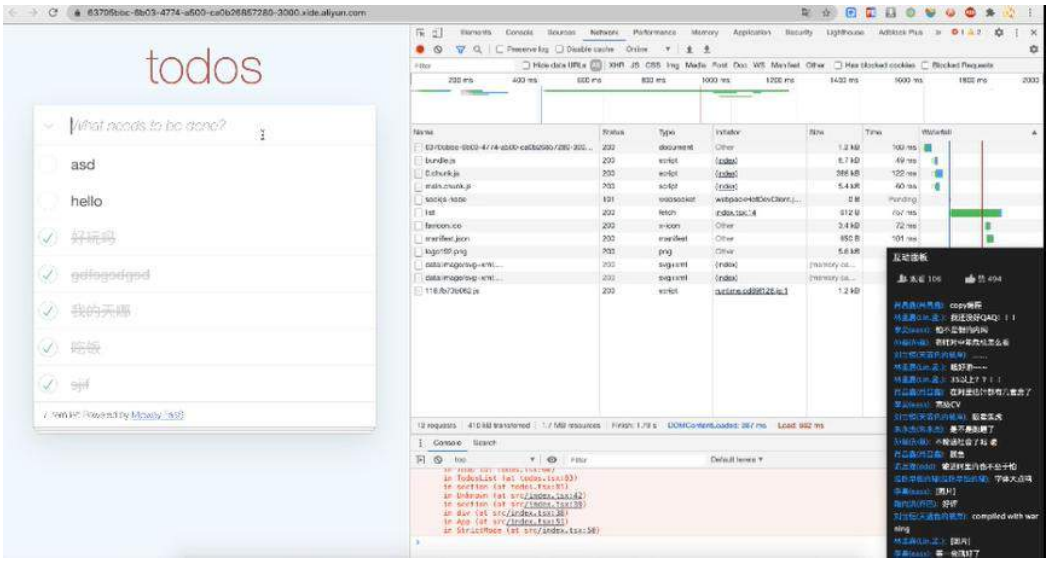
在使用 Midway serverless 的时候，是可以做非常多事情的。比如淘宝源 cnpm，因为在中国 npm 进项比较慢，所以淘宝工程师就做了一个淘宝进项来加速 npm 的安装，cnpm 的安装速度非常快。npm i 也是一种装依赖的方式，i 就是 install 的一个缩写。

安装依赖之后，可以通过 npm run dev 的方式启动开发服务。npm run dev，意思是运行在 package.json 里面写的 dev meaning，只要通过运行 meaning 就能启动开发服务。

运行应用

npm run dev

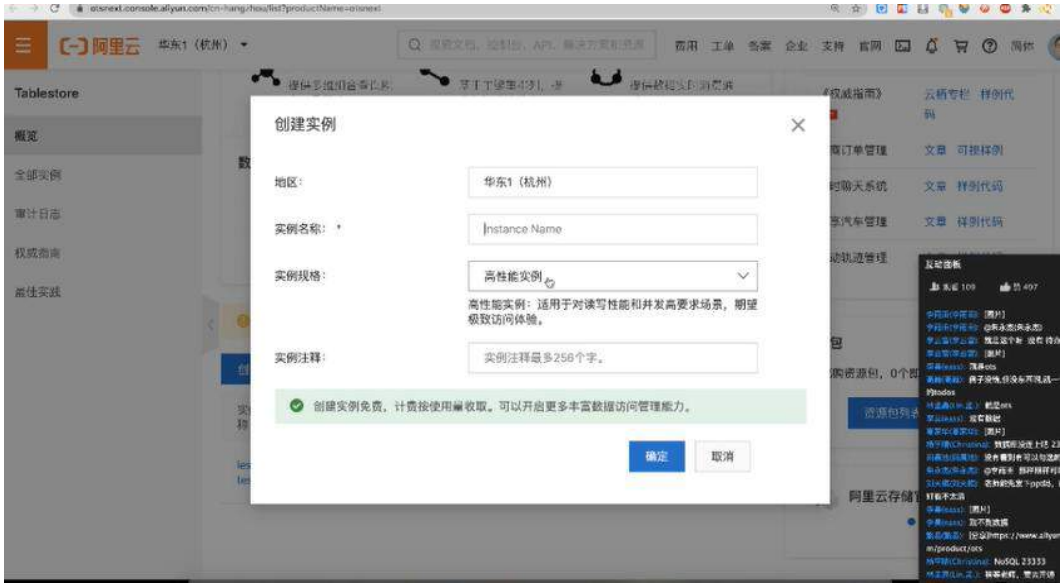
启动成功后，点开就能看到 todo list 页面，在 todo list 的页面里，默认连上我们提供的数据库。当在 todo list 里看到内容界面就可以往里添加或删除某个 todos，然后点击完成。



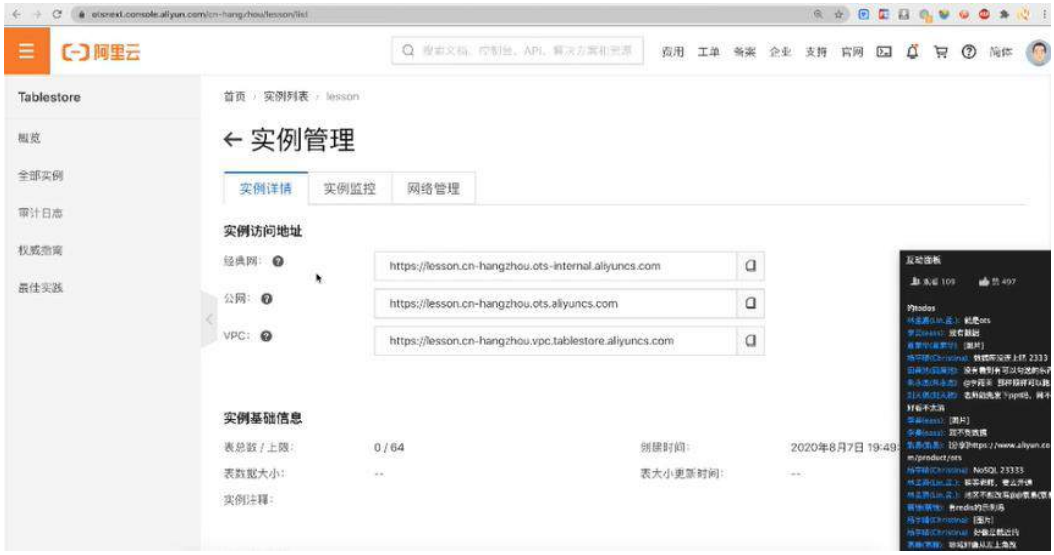
提示：在创建应用的时候要选实验室里面的 Midway OTS 数据库，而不是 RDS。

二、如何去使用 OTS 数据库?

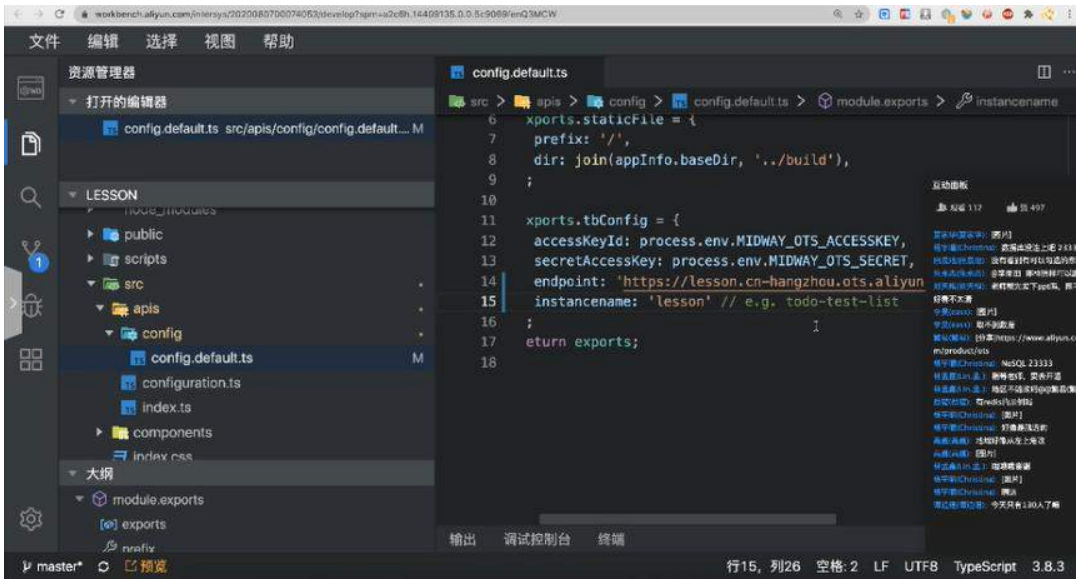
OTS 是阿里云推出的一款数据库。点进创建的 OTS 数据库后，会看到管理控制台的界面（要提前开通权限）。在这个界面里首先要填实例名称，再选确定，然后就创建了一个新的 OTS 数据库。



创建完后点进去到实例管理界面，会看到有几个地址，我们利用右侧的复制按钮，复制出公网地址。

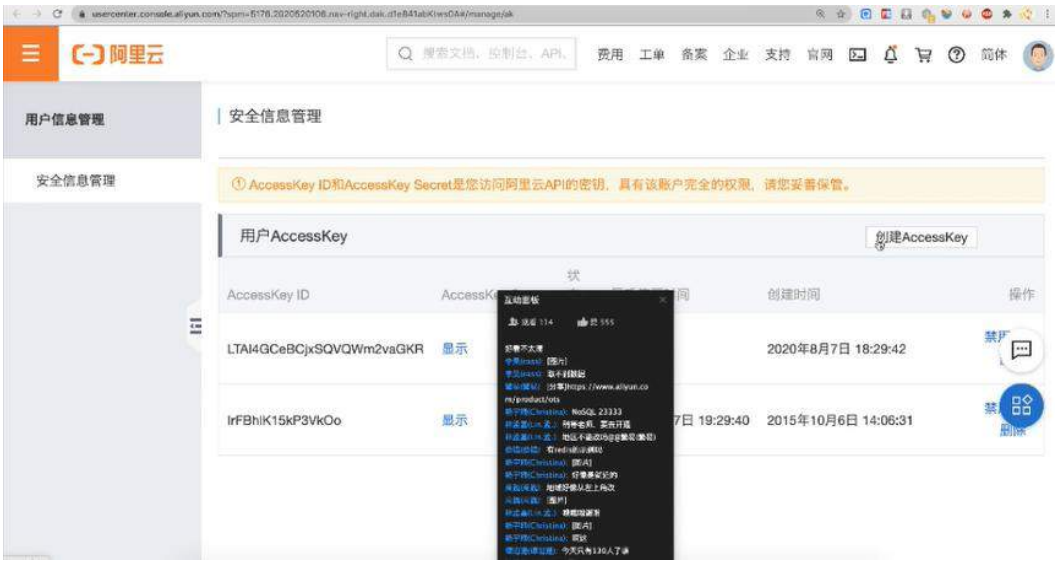


复制完成后，继续回到云开发的 IDE，找到 src 文件夹下面的 config.default.ts 文件，找到里面叫 tbConfig 的字段，在 tbConfig 这里能看到有一个 endpoint，这就是要填复制公网地址的地方。最下面是 incidencename，填进创建的实例名称即可。

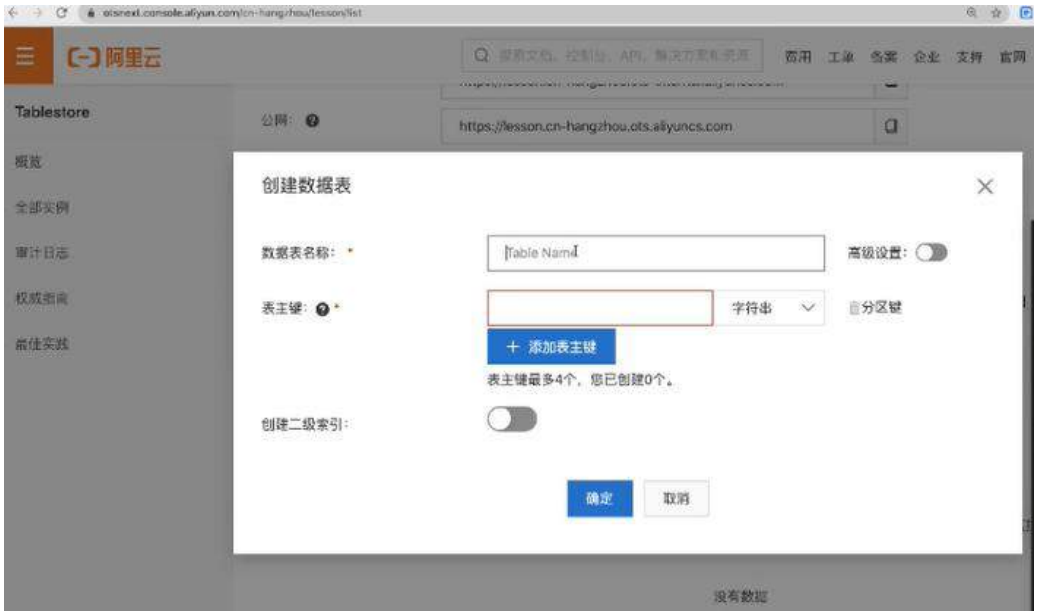


最后是阿里云的 accessKeyId 和 secretAccessKey。这两个地方需要填的内容可以到个人信息页去获取。

获取途径：点开表格存储管理页面，点击右上角，在浮出的窗口里会有 accessKey 管理，点击它。如果没有 accessKey，可以点一下右上角的创建 accessKey，完成创建自己的 AccessKey 和 secretAccessKey。在创建完之后一定要复制好 key 的 ID 和密码。

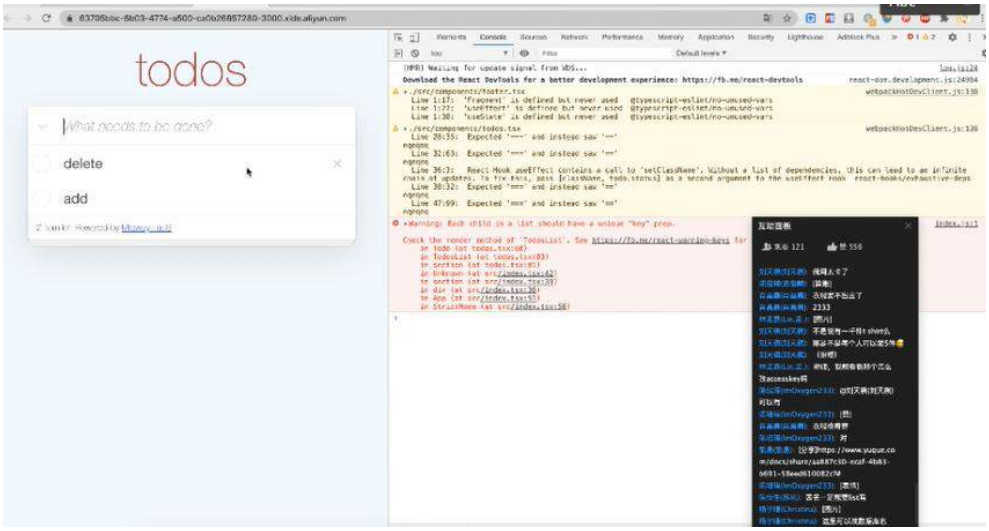


填好自己的 accessKeyID 和 SecretAccessKey 后，在表格存储的管理页面，创建新的一张数据表用来存储列表数据。补充表格的名称并设置好主键后，确定完成创建数据表。



创建数据表后，再次打开终端，重新运行 npm run dev。这次系统就会使用刚刚创建的数据库，而不是系统默认的数据库，去完成整个 todo list 的开发。

当我们创建完成属于自己的 todo list，就可以对这个属于自己的便签小程序进行增加、删除等勾选操作。



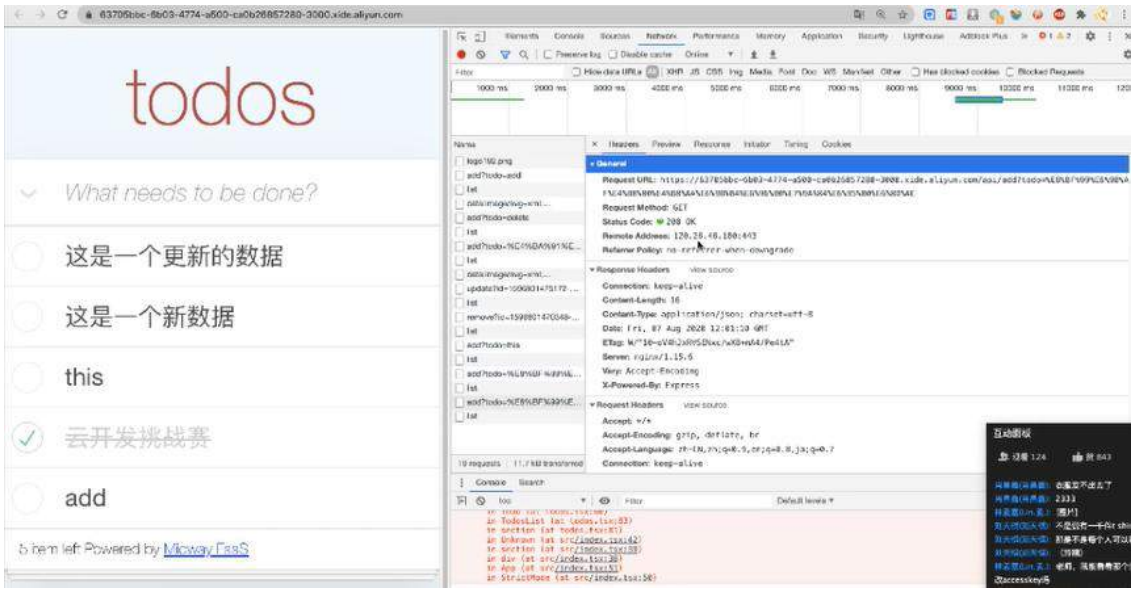
在阿里云开发平台左侧目录里，可以看到 configuration 的文件，这就是初始化和使用阿里云 table store 的文件。阿里云提供的官方的包叫 table store，这就比以往开发数据库需要手动去机器上部署然后连接的过程，方便快捷多了，现在只需引入这个包就可以直接使用了。

有了这个包，就可以直接写入对应 tbConfig 里的 KeyId、endpoint instancename 等内容。完成这步操作就能在代码里直接使用这份数据库，然后通过阿里云包提供的方法来直接获取到一些数据，并返回给前端。通过这样的操作就能把数据库的数据做好存取和增删改查。那么即使用户离开，他下次回来后还能看到上次存取的数据。

通过 OTS 示例可以快速地开发一个带数据库增删改查的程序。如果想应用数据库更多的功能，比如说查找、聚合、排序等都可以参考阿里云 OTS 的产品文档。

三、怎么添加数据?

添加数据可以直接在 todo list 里面增加。在输入栏添加数据，然后新数据会调用 add 的接口，在实际 sdd 的代码里面，它就会接收到请求参数。



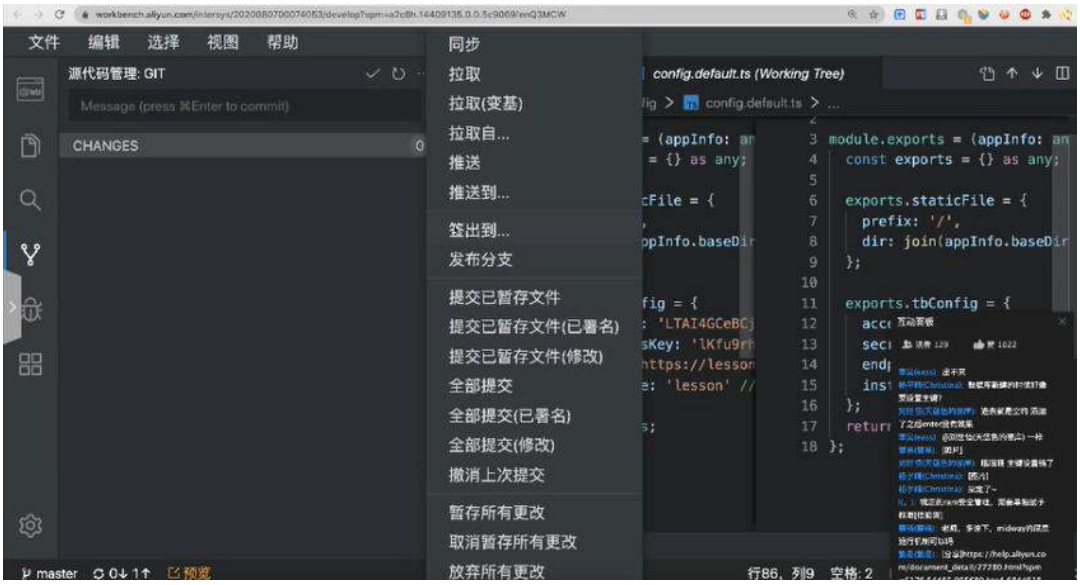
四、如何做到表格存储数据库?

表格存储其实是在做一些海量化结构数据存储的 Serverless 的 NOSQL 数据库，对很多产品都很适合。阿里云还有安全管理，不过需要单独赋予权限，如何赋予权限可以参考阿里云平台表格存储文档的讲解。

五、后附：课程 FAQ

1. 关于部署。

在部署前要把代码推到远端的一个分支，输入提交信息，点击推送。



推送后在左侧点击部署，然后页面就能部署到线上了。

2. 魔法值如何处理?

魔法值相当于一些可能很难懂它的意思，但是又不得不用的量。这个时候要么加一个注释，要么用一个常量去注明意思。简单来说就是要留下一段解释，或者是一段好读懂的代码，让未来同学知道是怎么做的。

3. 如何应对高并发?

这跟 Serverless 特性相关。Serverless 有一个特性叫动态修库容，动态修库容的意思是当流量洪峰过来的时候，它会自动拓展一些实例，应对这个流量洪峰。为什么要用表格存储？因为表格存储是一个 NoSQL 的数据库，表格存储的设计之初也是为了应对流量洪峰去做的一些相关的事情，它跟 Serverless 一样，会根据你流量的大小来去自动抗压。

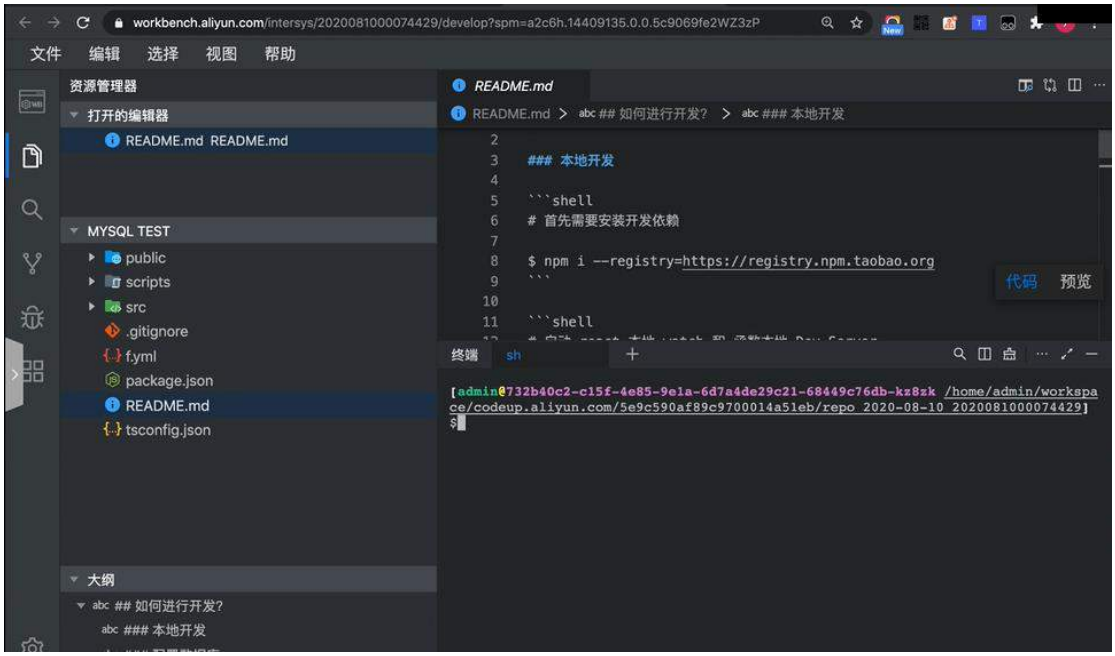
如何访问云 MySQL 数据库和实践

作者 | 洗剑 淘系 Node 架构技术专家

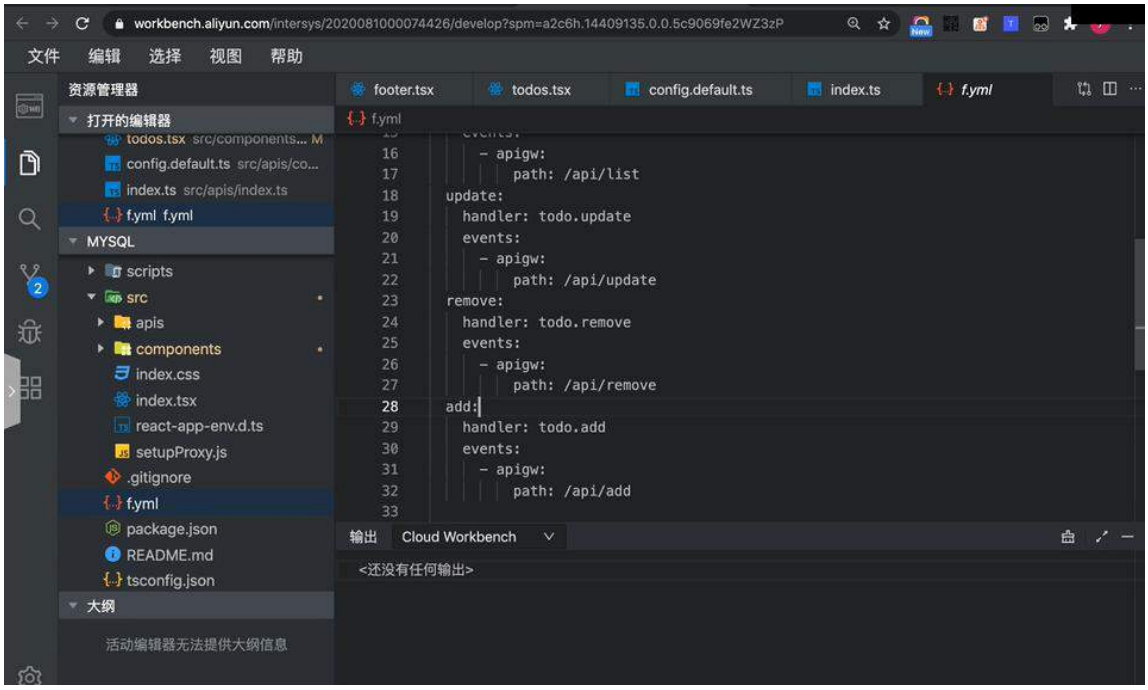
上一篇内容分享了如何创建、使用 OTS 数据库，如何增加数据。本篇主要介绍 Midway Serverless 及 Mysql 数据库进行应用开发。

一、创建 MySQL 数据库应用实例

首先像上一课介绍的，先到阿里云云开发平台创建一个数据库。在选择实验室后，使用 Midway Serverless MySQL 数据库的示例来创建。点击完成之后，云开发平台会自动创建这个项目仓库，同时把一些项目模板创建好，然后就点击开发部署。这个时候云开发平台会打开一个纯云端的 WEB IDE，在 WEB IDE 里可以感受到和本地 IDE 几乎一样的开发体验。它提供一个内置终端，可以在里面执行需要的命令。比如先要安装一个开发的依赖，然后去执行本地的 Dev 环境，这样整个项目就可以在本地给运行起来了。这时在本地就可以进行代码修改了。



WEB IDE 的左边是项目的目录数据结构，第一节课已经详细的介绍过目录结构，在此不再做赘述。这个项目里有几个函数，有渲染前端 html 界面的函数，有数据的增删改查函数，list，update，remove 和 add，这几个分别对应我们接下来要分享的使用 MySQL 数据库的增删改查四个功能。



左侧目录架构里有一个叫 config 的文件夹，在 config 文件夹里面有一个 config.default.ts，熟悉 egg.js 的同学会知道对于配置会有多套环境，这些配置会与 config.default.ts 合并，生成一份线上在运行当前环境的一个配置的对象。在配置里会配置一些，比如今天主要介绍的连接 MySQL 数据库，在这个配置里面我们就会通过环境变量，把一些 MySQL 的配置给加进来。

在阿里云云开发平台，当你创建好一个实例的时候，云开发平台会给你默认的提供一些配置，比如 RDS 数据库名称、数据库的连接地址、密码、端口等信息，这些信息会在云开发平台通过环境变量的形式，注入到系统的一个环境变量里面去。

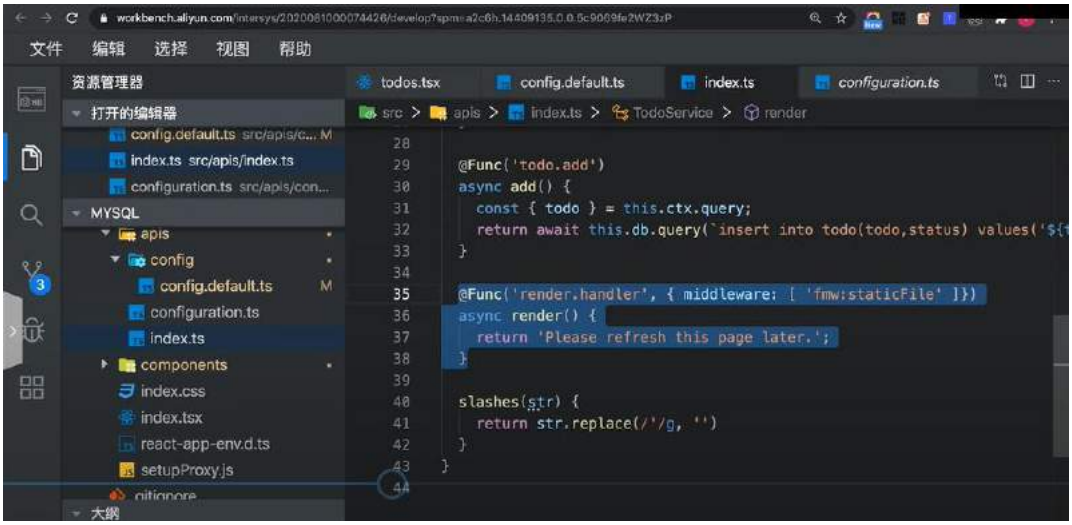
这个环境变量在外面提供的各种配置里面其实都有，我们可以把这些配置拷贝到内部的环境变量里，比如要连接到某些服务器上自创的数据库，或者是其他平台提供的数据库的时候，可以把数据库的连接信息配置到这。比如说数据库的 host 和 port。一般来说 Mysql 默认的连接端口是 3306 端口，当配置好了之后，我们通过 configuration.ts 文件，在上面导入下配置，importConfig 就把配置导入到我们整个 Midway 的项目里面去了。

index 是函数代码，可以通过@Inject 把刚刚创建的数据库的实例注进来，然后通过@Config 装饰器，把 dbConfig 的配置注进来。这样注入进来就能拿到 dbConfig，就是刚刚一些配置的数据，然后在 onReady 里进行数据库连接。

当数据库连接完成，我们把数据库名、用户名、密码、端口和连接地址都配好后，再进行数据库的认证连接，这样连接就完成了。如果连接不完成，连接出错了，这个页面会捕获错误直接输出提示；如果连接成功了，系统就会把连接成功的 db 实例，注入到整个 Midway IoC 容器里面。这样就可以在其他的任何一个地方去通过@Inject 注入代码里，最后就可以通过 this.db 达到已经连接好的数据库的实例。

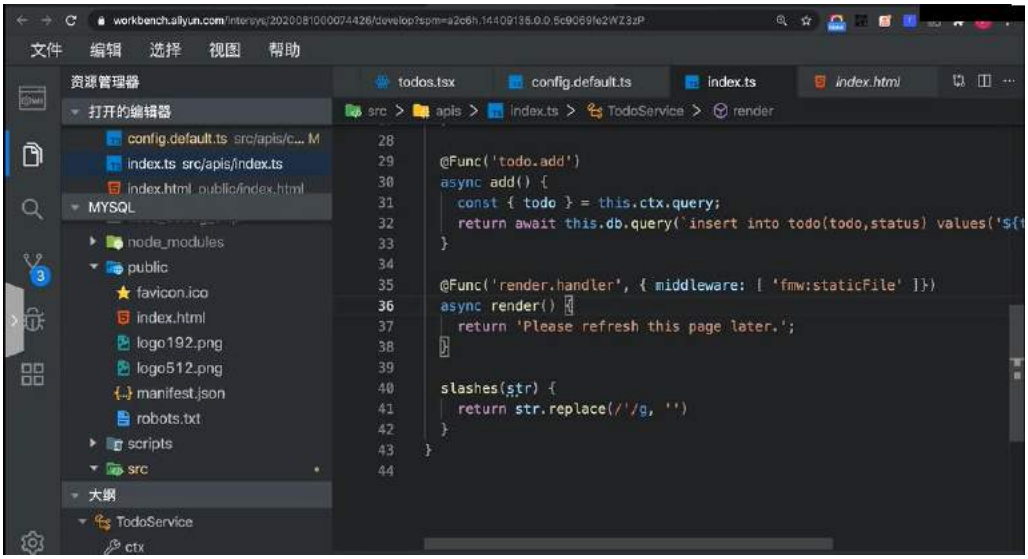
熟悉 Mysql 的同学都知道普通的增删改查和执行各种 SQL 语句，比如查询是 select，更新是 update，删除是 delete，插入新增是 insert，这些都是一些比较常见的语句，这些都可以通过 this.db.query 去执行，然后就可以拿到对应的存在数据库里的结果。

我们再介绍一个与上文提到的写法不太一样的函数。上面提到的函数只是声明了这个函数的函数标识，下面这个函数在里面提供了一个中间件的配置。在 Midway Serverles 框架里面，支持提供就配置一些中间件，相当一个请求进来通过中间件进行拦截，或者说通过中间件进行一些处理，然后再把截下来的数据进行一些返回的处理。



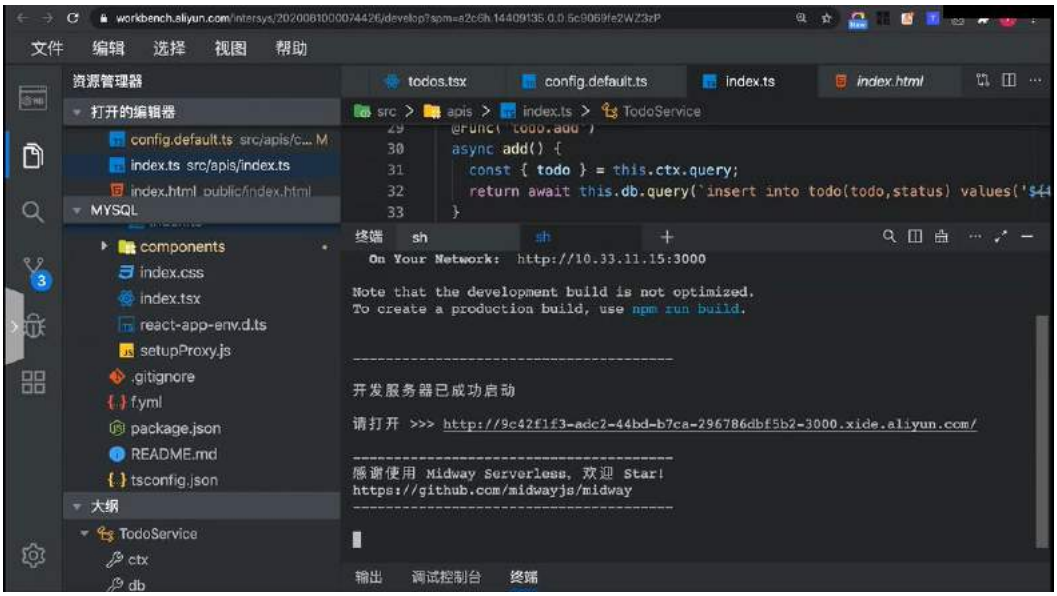
render 函数是用来做前端页面渲染的，前端页面会包含 html 文件和 css 样式的代码，还会有一些 script 前端的解释脚本。StaticFile 这个文件可以把一些前端静态的文件通过这个函数进行输出，在 default 这里可以把一些静态文件的目录指定了一下是 build 的目录，build 的目录其实是通过 public 目录来编译生成的。

public 的目录里面，有 html 前端渲染出来的 html 模板，通过一些编译，会把它生成一个显示出来的 html 代码。当一个请求首先通过浏览器访问某个域名的地址，再进入到这个函数，最后进入到阿里云的网关，阿里云的网关就会根据这个函数的请求路径去找到对应的函数去执行，然后把这个请求转化到函数里面，再在函数里拿到 request 对象，这个时候中间件就开始执行了，中间件执行的时候会判断。比如请求路径的文件正好在静态文件 public 目录里边有，那么就直接把文件给返回了。



在所有的 Midway 项目里运行，都是执行 npm run dev 的，这个时候会去启动前端的一个 dev server，然后再启动一个后端的 dev server，比如请求一些函数的接口代理。

当在控制台里面输出开发服务器成功启动后，并且输出一个阿里云的网址，打开网址，可以看到输出 info，这个关键的 info 会匹配到 api/list 接口，然后会请求 list 函数，然后去执行一个 query 的查询，最后从 todo 数据库表里面返回前 5 条数据。



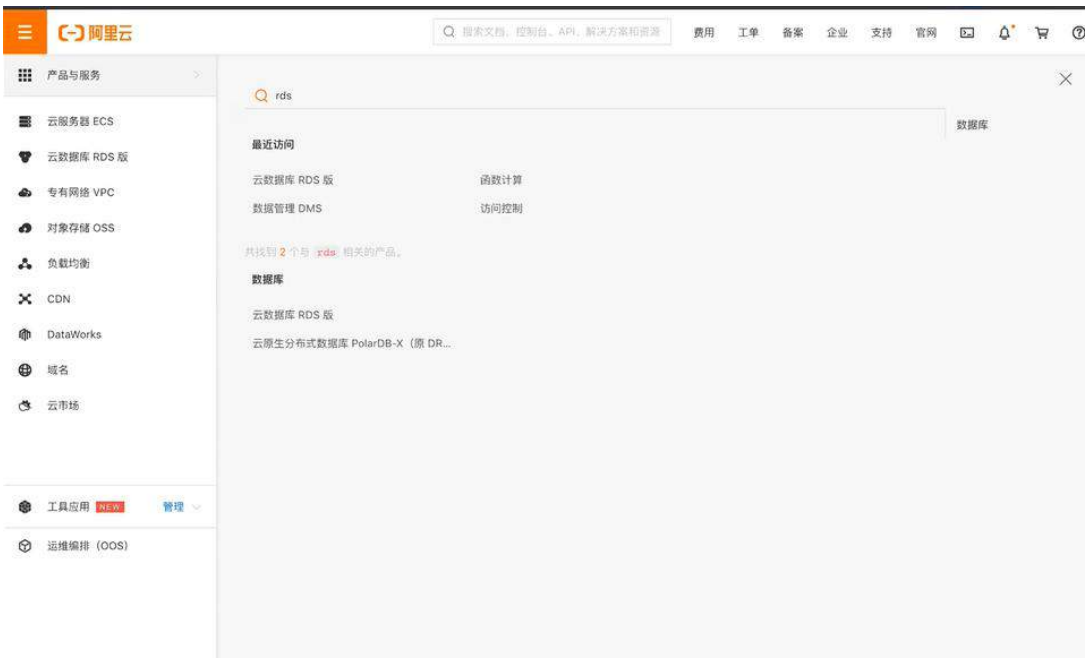
（演示）可以看到这就是我们用本地 server 运行起来的一个基于 Mysql 数据库的一个实例，它其实也是和 OTS 一样的数据库。

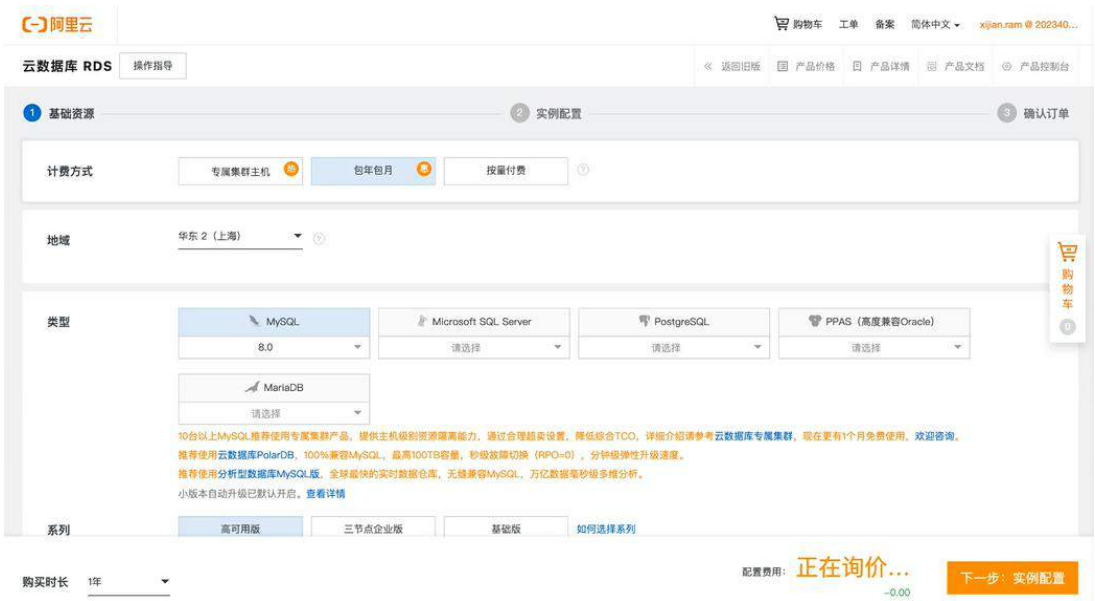
上篇关于 OTS 介绍的介绍大家可以得知， OTS 它是阿里云的表格存储，它是一个非关系型的数据库，它的数据是一条一条在里面有记录的，而且这个数据库和 Mysql 数据库的区别就在于 Mysql 数据库需要购买一个双核 2G 的实例的资源，而 OTS 表格存储是按使用流量计费的。

比如说我个人的博客，目前我使用的是 OTS 数据库，因为每天的流量比较低，存储的数据也相对少一些，这样的话每个月或每年整个数据库的使用费用都很低，几乎没有。但如果使用 Mysql 数据库的话，成本就相对要高一些。阿里云云开发平台为了大家尝试使用这个，会给大家在创新应用的时候免费创建一个 RDS 就是 Mysql 数据库的实例，大家可以去免费的去体验，去使用。

二、怎么去创建自己的 Mysql 数据库呢?

首先登录阿里云，登陆后会发现在左侧会有一个产品和服务，在这里可以搜索 RDS，搜索结果会出现数据库 RDS 版，然后点击一下就可以进到 RDS 的管理控制台。点击 RDS 管理控制台左侧的实例列表，就可以在右上角来创建实例，创建实例后就会跳出一个新的 RDS 的购买页面，大家可以选择自己要使用什么样的 RDS 数据库，最后勾选确定。





刚刚演示的 Mysql 数据库，只展示了 5 条数据，如果需要修改可以在 list 的函数里通过 select from，然后设置 limit 限制多少条数据。以上是 Mysql 数据库的介绍，下面介绍下如何创建 OTS 数据库，其实前一篇内容都已经介绍过了。

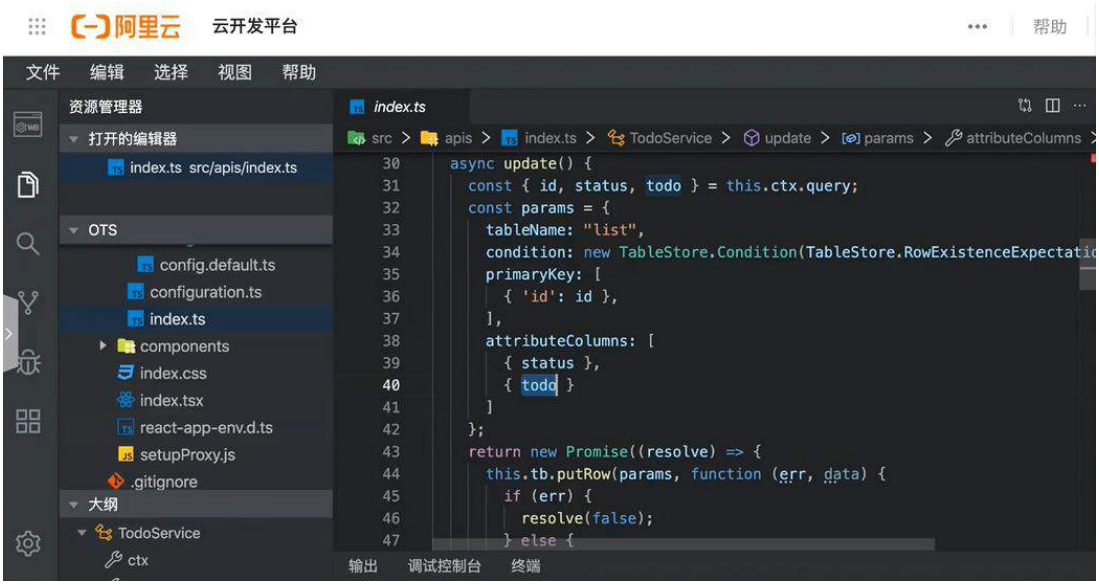
三、如何创建 OTS 数据库及两种数据库的区别和应用

我们创建一个 OTS 数据库，创建好后点击开发部署。OTS 数据库整个代码结构和 Mysql 基本上差不多，在 apis 里面也会有一些 configuration，在 config 里也同样会有 config.default，也是通过环节变量来注入的一些 OTS 的配置的数据。

在 OTS 的表格里面也可以通过一些增删改查，获取里面的数据并进行展示。因为 OTS 数据库是一个非关系型的数据库，是没法通过 Mysql 进行数据执行的，也就是说没法通过 SQL 语句进行数据查询。如果有这样的需求，需要通过 table Store 的一个实例，然后通过一些参数来查询里面的数据，比如说 getRange，getRange 这个方法就是通过 table Store，批量进行数据获取。如果你要获取单条数据，可以用 getRow。

update 其实是 putRow，put 就是把一些数据进行更新。执行 putRow 的时候整个参数对象是需要在一个 OTS 的数据库里面。它是通过主键进行划区，然后进行一些数据的分区存储的，所以主键是必须要填的，这个主键与 Mysql 数据库创建时数据后面标注的 primaryKey 是一样的。无论是查询或是修改，都是通过主键去进行修改的。

Mysql 数据库需要修改一个数据的话，一般通过 update 语句然后去 set，比如，去 set 一条 a 等于 10，b 等于 20 的数据；但是在 OTS 数据库里，要把某个数据的某些项更改则是通过一个数组传进去的。如果要把它的状态进行一些更改，把它要做事情的内容进行一些更改，它的执行是一个回调形式的，把它通过 promise 进行一些包装，然后变成一个 promise，之后就可以通过 async/await 去执行。



remove 是一样的，在 OTS 数据库里这些操作也都是通过 primaryKey 去做的。在 list 的时候可以看到我们是这么去写的，有 inclusiveStartPrimaryKey，是指从哪条开始到哪条结束，是批量性的获取，它定义 primaryKey 排序的规则，比如说逆序排序或者是一个正序的排序，通过排序的规则以及它 ID 的最大值和 ID 的最小值，可以按批量的把这个数据库获取出来。因为 OTS 数据库返回的数据格式，是一个大的 Object，不像 Mysql 获取后的数据是一条一条的这种数据结果。通过 OTS 数据库获取的数据，需要对 Object 进行一些处理之后才能变成的一条一条的这种形式。我们提供 OTS 数据转化的封装很简单，通过 rows 这个方法就可以把数据变成一条一条的这种结果形式。

第四章 云开发提高篇

轻量易用的运维监控能力

作者 | 欲休 阿里云技术专家

本篇已经是云开发平台的第九篇内容分享，从第一篇到第九篇，大家可以发现，这是从云原生应用诞生、应用创建、开发代码、迁移部署和部署后的运维监控，一整套操作。本篇内容主要分享轻量化运维监控这部分。

一、如何绑定域名、设置泛域名并设置多环境绑定?

首先先创建一个香港节点的应用开始，因为后面会涉及到域名的解析。

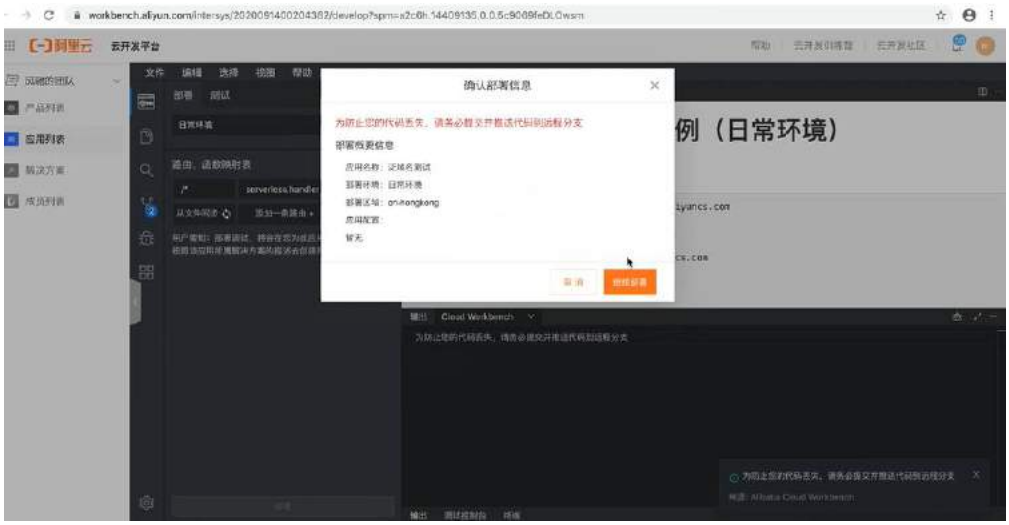
应用创建后，首先要确保进入到在线的开发环境，然后去部署。部署后复制 CNAME 域名，然后打开域名管理平台，比如说阿里云的 DNS 域名管理，然后添加记录选择记录类型 CNAME，主机记录里填写 ***+自取名称***，后缀是主域名 workbench.fun。这就意味着，只需要做一次解析，未来*可以用任何域名替代。

创建之后，把上述复制的 CNAME 域名，粘贴到记录值框中，然后点击确认。记录值就把它填写我们这边复制过来的 CNAME 地址，第一次部署之后得到的 CNAME 地址复制下来，然后把它填写到记录值里边，填写之后点确认。

确认后再回到云开发平台，点击应用配置并去绑定域名。上文中已经绑定的域名是*company. Workbech.fun，*作为通配符可以用任何内容替换，比如在日常环境下，将*换成 test，那么域名就变成了 test.company.workbench.fun；预发环境一般用 pre 替代*；线上环境不是泛域名，所以这里还需要再解析，这里我们用 production 的缩写 prod 来替换*，就成了 prod.company.workbench.fun。配置完成后点击确定保存。

这样操作下来，我们就可以通过一次域名解析，对一个应用的三个环境同时配置线上真实域名，这些配置的域名不会变化，每次都可以通过它们去访问。

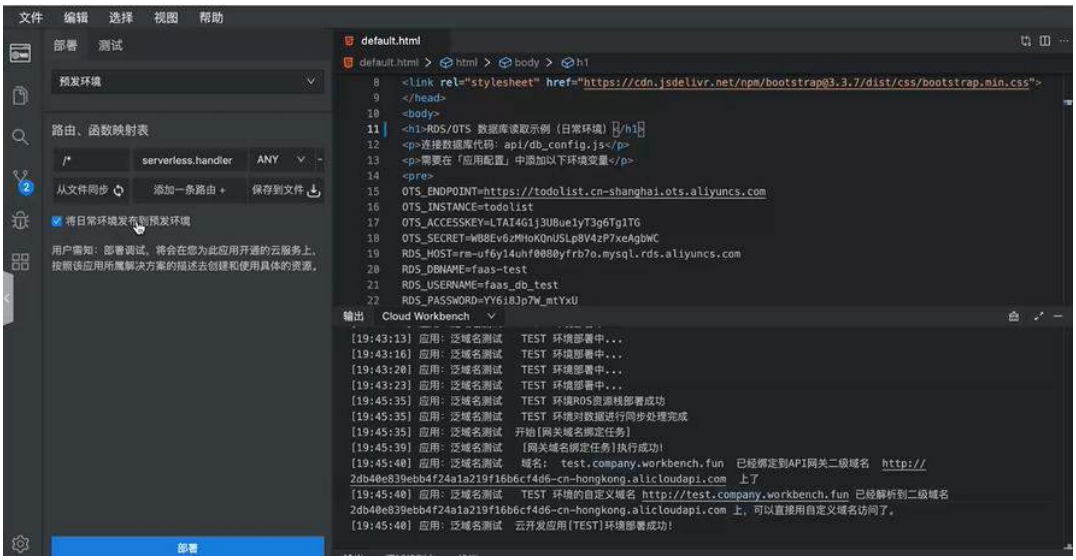
保存后再去云开发平台部署，比如部署日常环境。前提是安装了依赖，部署保存后可以先测试一下。当测试后页面显示日常环境，就可以部署到日常环境，点击继续部署，就可以等它构建、打包、部署直到完成。



部署成功后，在部署成功的消息里可看到“TEST 环境的自定义域名 test.company.workbench.fun 已经绑定到 API 网关二级域名上，可以用自定义域名访问了”。

打开 test 的自定义域名，而不是平台分配的临时二级域名，因为临时的二级域名只有 30 分钟时效，所以在真实的开发环境当中，一般大家都需要绑定一个固定的域名。

接下来部署预发环境，理论上讲当日常环境没有问题就可以选择部署预发环境。还是回到云开发平台，选择预发环境，然后勾选“将日常环境发布到预发环境”，然后部署。

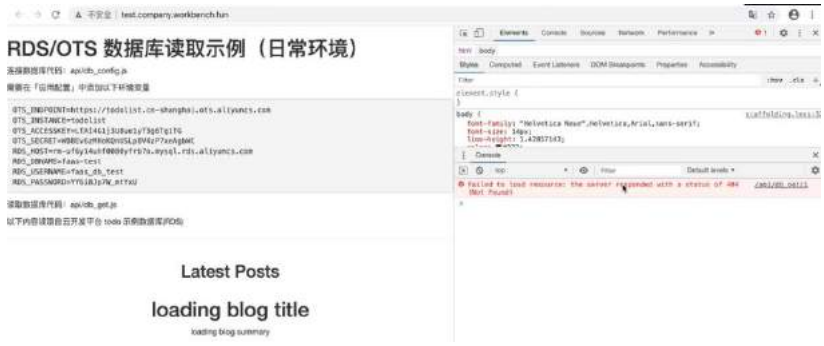


预发环境部署好了，点开 pre.company.workbench.fun 的域名，打开预发环境。这样日常和预发两套环境就固定下来了，都没有问题的话，接下来就可以把它部署到线上环境。

部署到线上环境的方法也是一样的。线上环境部署成功后，打开自定义域名，与前两个环境是一样的。所以，当我们日常需要一个环境做联合测试，大家可以用日常环境；用预发环境做回归测试；前两者都没问题，再发布到线上环境。

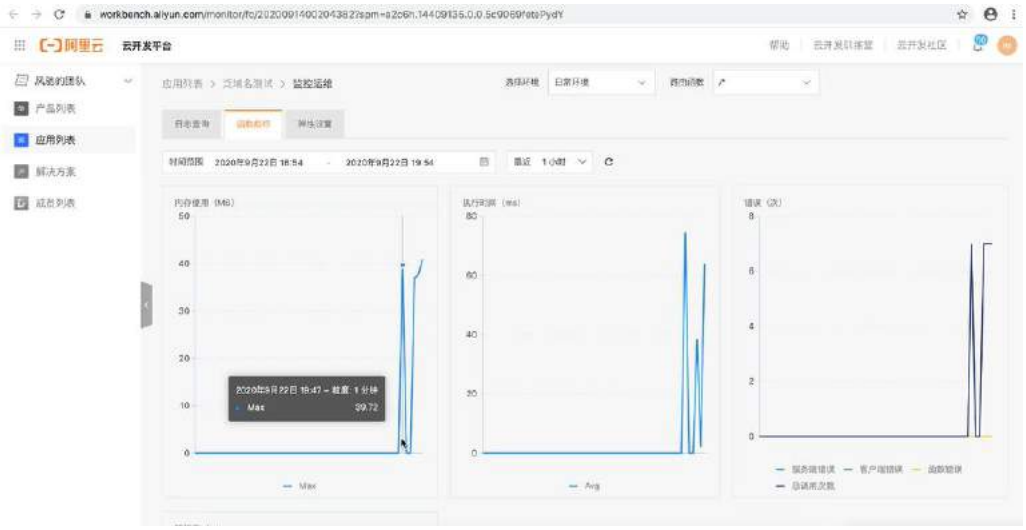
二、如何检查故障并解决故障?

如何检查线上的故障。在刚刚部署的日常应用页面，可以看到标题是 RDS/OTS 数据库读取示例，下侧小字提示“以下内容读取自云开发平台 todo 示例数据库（RDS）”，但是下面的页面并没有 loading 出具体内容。那么我们可以通过点击右键并点击检查，看看具体问题。

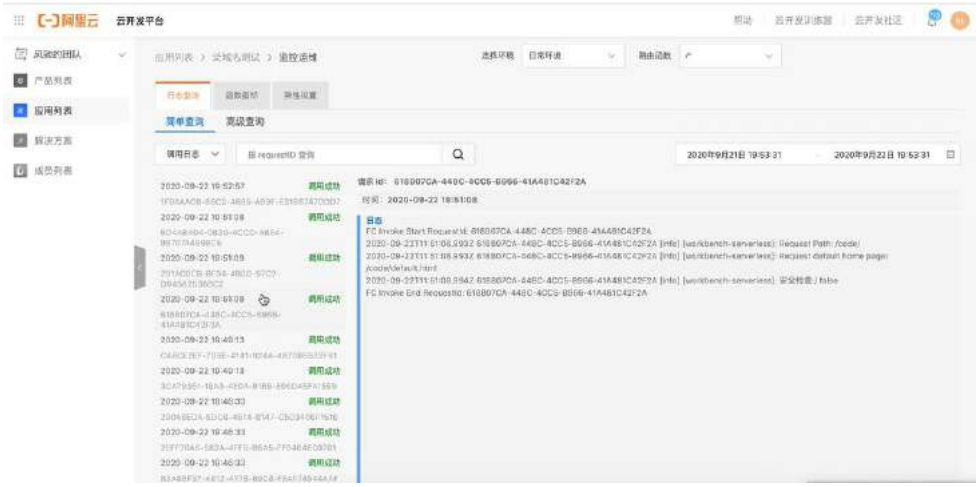


发现有错误报警后，我们再次回到云开发平台。在云开发平台可以看到当应用上线后，就会出现“运维监控”和“应用账单”两个入口。运维监控是可以帮助快速查看应用线上运行的日志。那么为了解决上述的问题，我们点开运维监控查看运行日志。

打开运维监控后，顶部会让我们选择要查看的运维监控环境。我们点开日常环境，这里面有日志查询和函数指标。点击函数指标会切换到函数指标的一个界面，可以看到日常环境函数的调用情况，比如说内存使用情况，执行的时间，出错的情况，然后包括错误率等等这些信息。

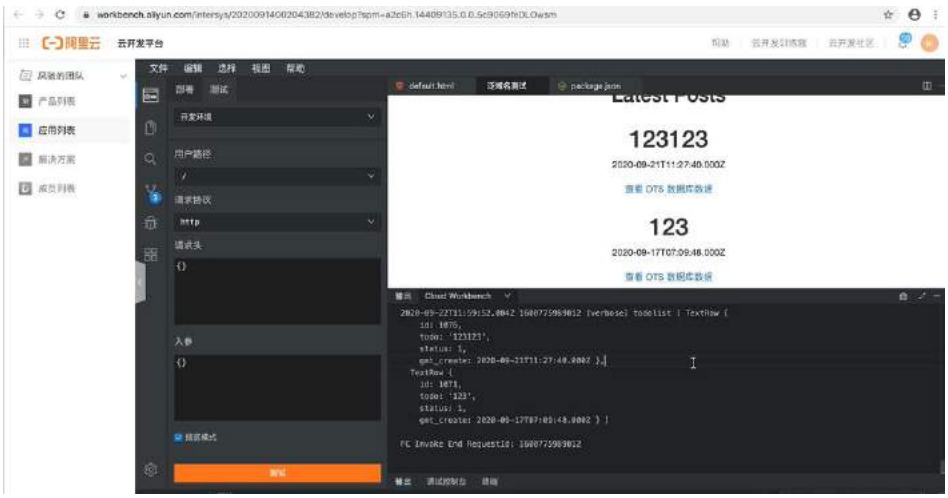


点开日志查询里有个简单查询，有个高级查询，一般情况下简单查询就足够我们去使用了。在日志查询里，点开每一个请求都可以看到请求背后的信息。这里我们可以点开每一个请求，看到请求背后的日志。



那么我们怎么能通过海量的请求日志信息，定位到具体的问题呢？举个例子，比如 default 要发起 api/db_get 的请求，它会引入 db_conflict，也就是配置数据库的信息，然后再根据用户使用的数据库来切换不同的代码去执行不同的逻辑。

我们看一下 db_config.js，它会去读取不同的数据库配置。我们可以把这些数据库配置录入到我们的系统环境里面，比如说应用环境。我们先在日常环境把这个数据库连接信息这些环境变量添加进去。添加进去之后，再来日常环境测试一下，测试后再重新请求获取信息。这样就成功了。



所以大家如果要通过日志去定位问题，可以在你的代码里去输入日志的消息。具体方法，可以打开帮助文档中的快速入门，在运维监控部分里，有教大家怎么在应用里去写日志。大家可以通过这个文档来快速的去了解如何在应用里写日志，如何通过日志定位应用的线上的问题。

以上分享的是在开发调试阶段查看日志并定位问题。如果已经部署到线上了，怎么操作呢？部署成功后，可以登录原日常环境自定义域名，打开页面标题提示会由“日常环境“变成”线上环境“，说明部署成功。

接下来来看这个部署成功的应用，这个应用既有前端代码也有后端代码，那么这些代码用户在访问的时候就会产生一些流量，有些流量是可以缓存的。那么怎么去缓存获得更高的用户访问性能并降低下行流量呢，推荐的做法是通过给应用做 CDN 的全站加速来实现。CDN 全站加速在帮助页面-快速入门中也有介绍的文档。

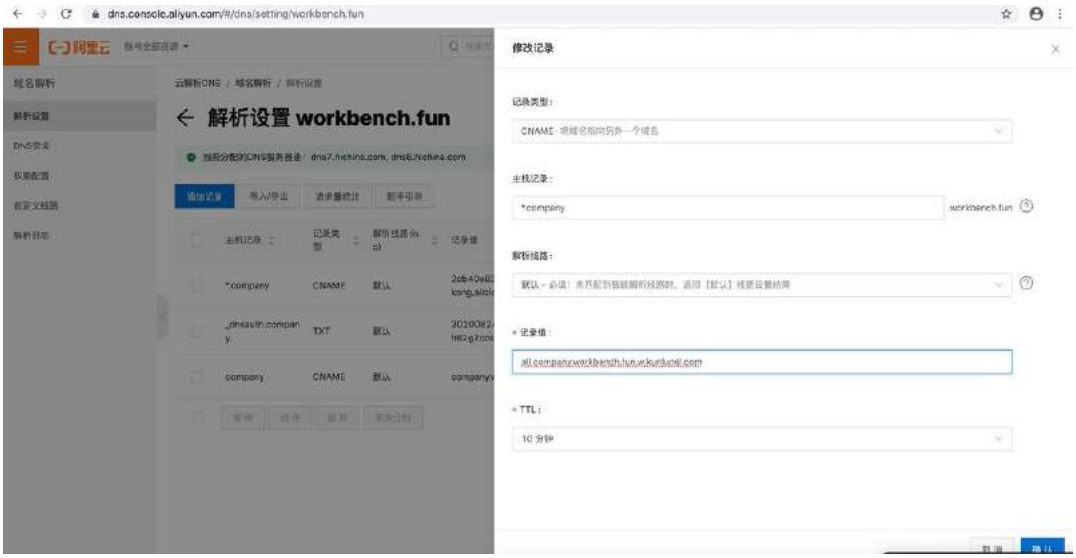
三、如何做到 CDN 的全站加速?

首先需要购买资源包。打开 CDN 控制台，点购买资源包，可以选择 18 块钱全国通用包 100GB，基本上正常情况下，个人应用都是够用的。购买之后，再到 CDN 控制台添加一个域名，比如泛域名*.company.workench.fun。

资源组不用选，业务类型是动态加速，源站信息选源站域名；然后将应用的 CNAME 二级域名地址复制到域名栏中；端口选择取决于用什么访问，如果用 http 访问选择 80 端口；再选择加速区域，然后点击下一步完成。

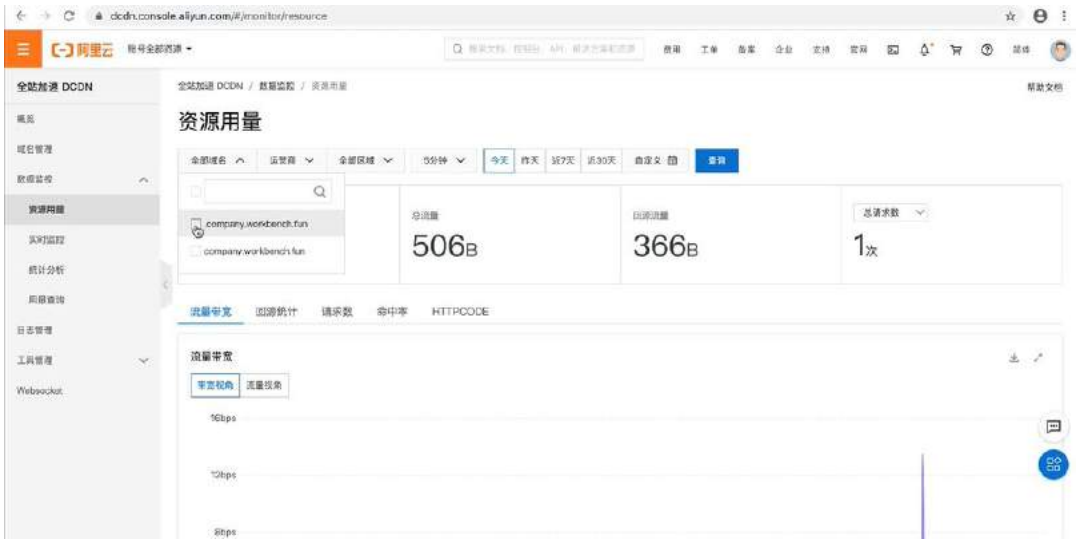
在实际使用中需要先将域名备案，这样可以在加速区域选择“仅中国内地”或“全球”，否则会有提示去备案。那么我们的课程分享的域名是测试域名，没有备案，所以选择“全球（不包含中国内地）”。

提交成功后返回域名列表，这里会出现一个 CDN 新分配的 CNAME 地址，这个新地址需要重新做一次解析。复制新的 CNAME 地址后再次返回 DNS 域名管理平台，点击添加记录。记录类型选择 CNAME，主机记录填写之前的域名地址，在记录值框中，将复制的新的 CNAME 地址粘贴，然后点击确认完成。

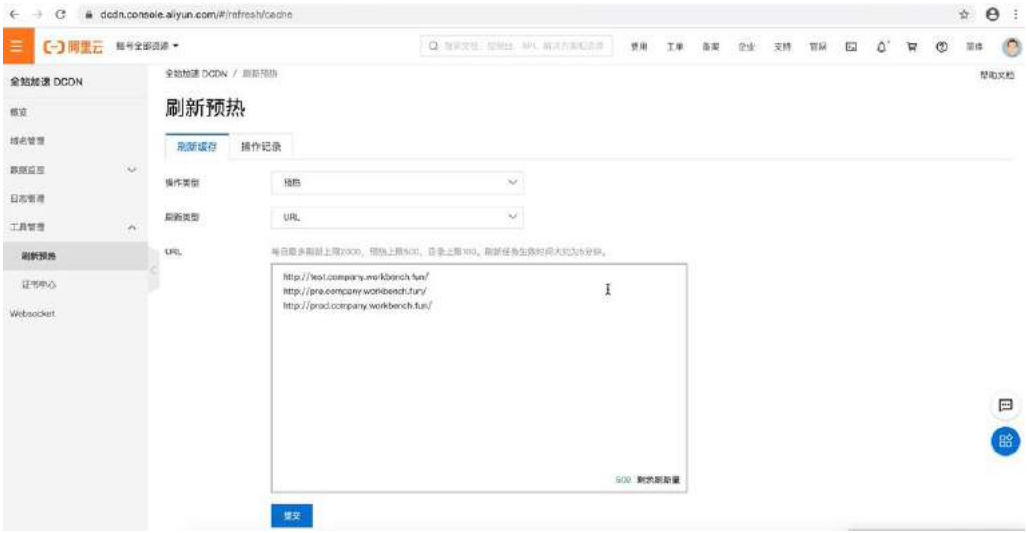


用泛域名的好处是，每个环境都可以直接去使用 CDN 加速服务。

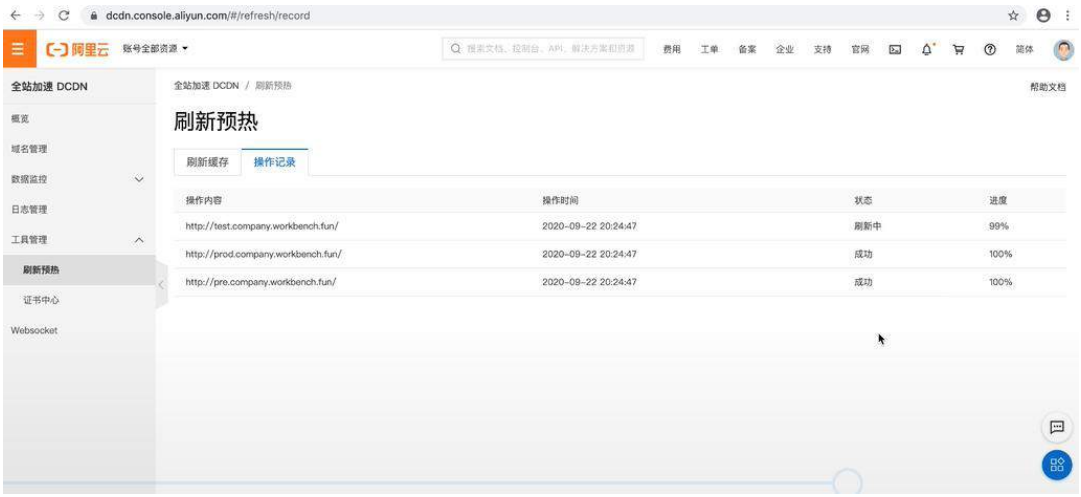
当“配置中”变为“正常运行”，表明配置成功。配置成功后，可以通过左侧目录中的资源用量、实时监控、统计分析等，查看配置成功域名具体日志信息。



在真正使用 CDN 的时候，可以通过页面左侧目录工具管理中的“刷新预热”，通过它可以提前把要缓存的地址，URL 或目录，添加进来。这部分的操作在帮助文档中也有详细的介绍。



点击提交后，可以点击操作记录，可以看到它会帮助针对添加的地址进行刷新，达到提前去除缓存的效果。相当于我们主动的把缓存同步到各个 CDN 节点，然后用户来访问的时候就可以直接享受到 CDN 的加速效果。

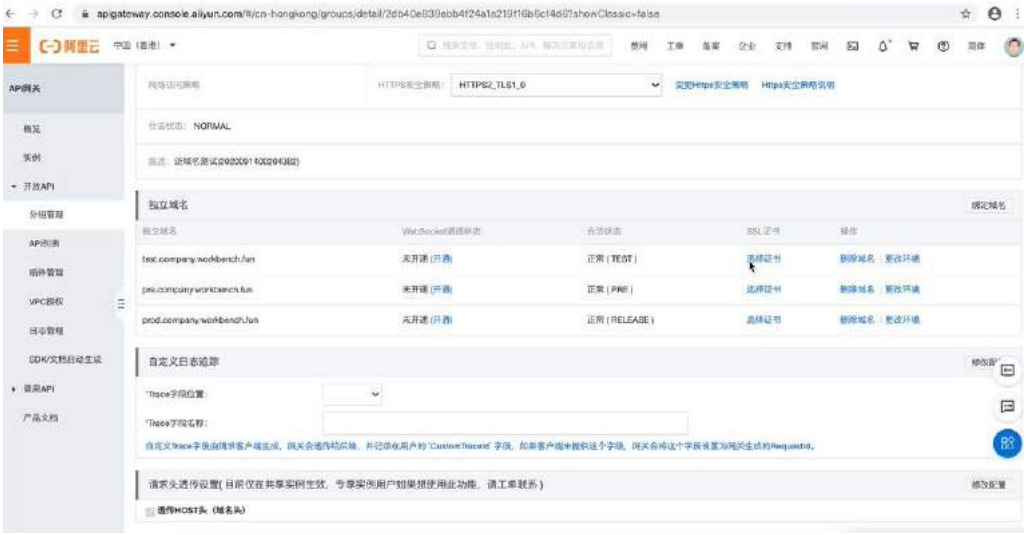


这样做可以有两个好处，第一，用户访问应用性能更快；第二，应用的下行流量的费用会更低。

四、如何用 https 访问应用?

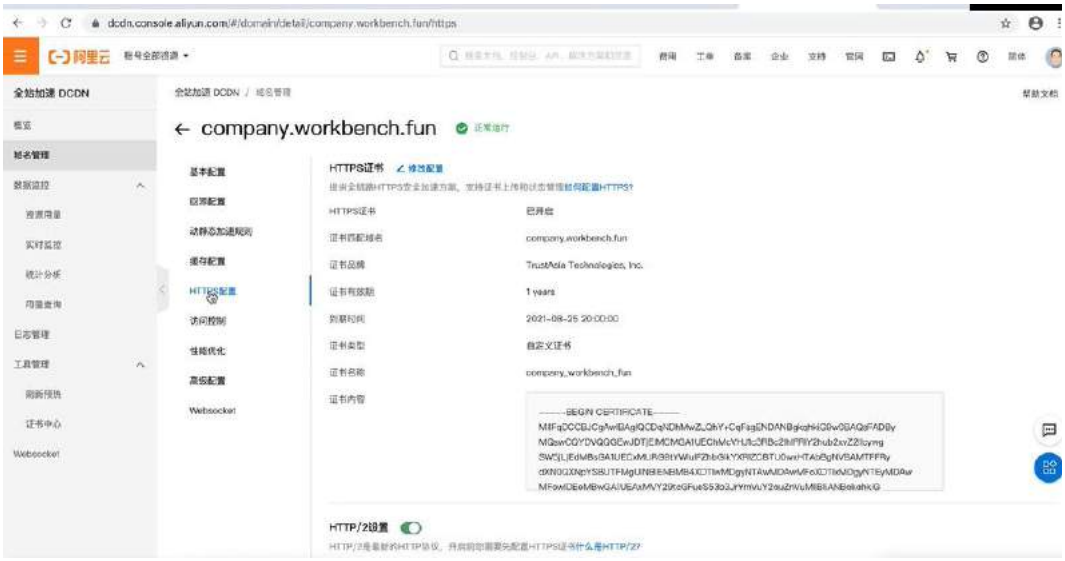
这个在帮助文档中也有详细介绍。用 https 访问有两种路口，第一种情况是没有配置 CDN 加速的情况下，是需要到 API 网关操作的，但前提是需要有 https 的证书。帮助文档中也有介绍如何申请免费证书的方法。

接下来介绍下如何在 API 网关配置 https 证书。首先打开 API 网关的控制台，然后选择应用所在的区域，比如香港节点。然后点开刚刚测试的应用，下拉看到独立域名部分，右侧后面部分可以看到“选择证书”。



如果你是第一次操作，要点击手动添加证书，然后补充证书名称和内容。怎么补充这些内容在帮助文档中都有介绍。全部填写完整后，点击确定证书添加完成。完成后就可以用 https 访问域名了。

第二种情况，如果已经配置了 CDN 加速，如何绑定 https？如果配置了 CDN 加速，CNAME 地址是需要改的。打开 CDN 域名控制台，点击左侧“域名管理”，点击对应域名右侧的“配置”，进入域名详情后点击左侧“https 配置”。



然后点击“修改配置”，在弹出的配置界面开启“https 安全加速”，开启后会弹出一个确认提醒，按照提示勾选“我已了解”，然后点击确认即可。

不改变本地流程的 CI/CD 能力

作者 | 欲休 阿里云技术专家

云开发平台系列课程之前一直是围绕网页端来进行分享的，也就是基于云开发平台的控制台。今天的内容是介绍如何通过云开发平台新开放的 Open API 来实现一些更定制化、更丰富的 CI/CD 的功能。

之所以会开放 Open API 来实现这些功能，是因为有些团队在本地已经做好了工具，希望能够使用云开发平台的部署测试或是创建应用等模块化的能力，增强他们已经有的 CI/CD 能力。

大家在云开发平台帮助页面可以找到《与本地 CI/CD hook 集成》，会有详细的关于 CI/CD 的介绍。

阿里云 Serverless 云开发平台不仅支持在线 CloudIDE 开发、测试和 CI/CD，同时也支持本地开发。在本地环境下，阿里云 Serverless 云开发平台在不影响原有开发流程的前提下，提供 hook 文件支撑项目本地 CI/CD，可以让原有项目轻松升级到云原生 Serverless 架构。

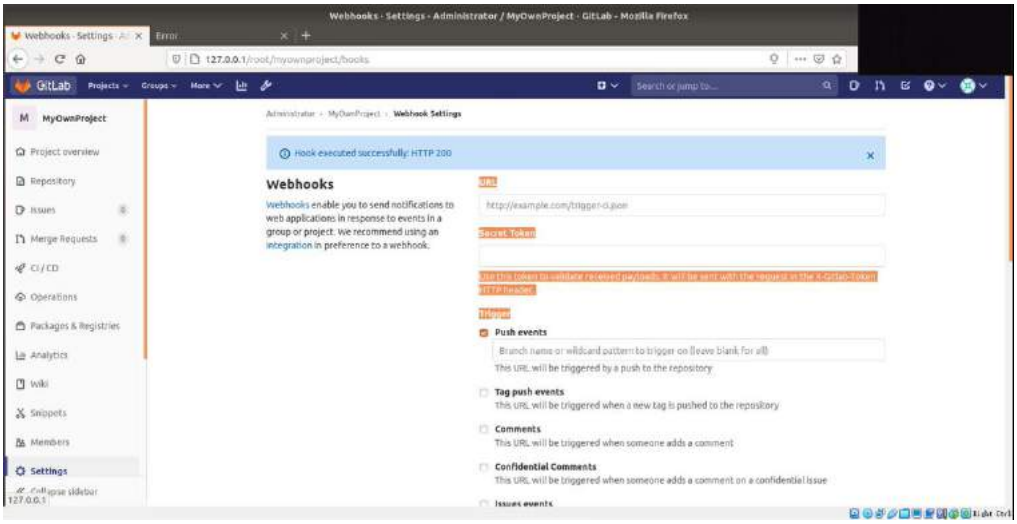
Hook 文件需要集成在代码托管平台的触发器逻辑中。以 Gitlab 举例，Gitlab 提供两种钩子：WEB Hook 和 Custom Hook，Web Hook 触发时调用提供 HTTP/HTTPS 接口，而 Custom Hook 则调用相关脚本，最终这两种 Hooks 都会调用或实现相关的 CI/CD 逻辑，实现项目自动化部署。

Hook 是在对远端仓库做铺代码等基础操作的时候，Gitlab 会对这些操作做一些捕获或拦截，然后通过这些捕获或拦截的动作，实现操作上的强化。

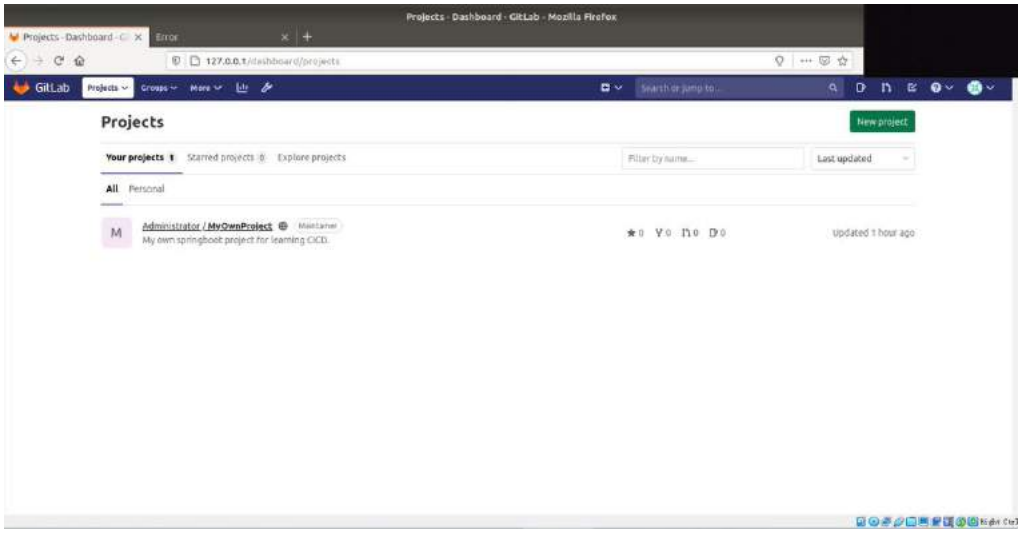
CI/CD 流程是围绕 WEB Hook 的机制来实现的。CI/CD 依赖 4 个环境变量：CI_WORKBENCH_ID（云开发平台创建的工程 ID），CI_ACCESS_KEY（阿里云账号的 Access key）、CI_ACCESS_SECRETE（阿里云账号对应的 Accesskey Secret）、CI_GITFLOW_PATH（可选，指定 gitflow 文件路径）。

如何构建环境大家可以参考帮助文档的介绍，这里不做赘述。

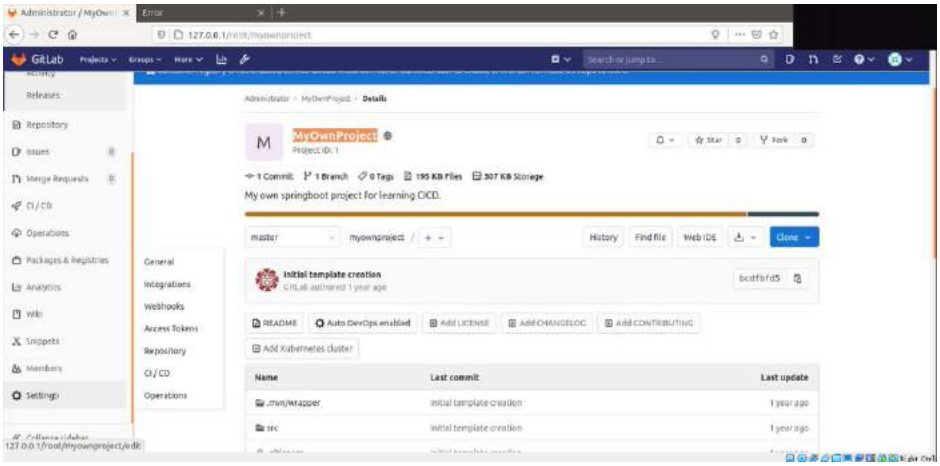
（演示）我们登录一个虚拟机并用 Docker 的方式建设一个 Gitlab 的私人服务器。这就相当于一个团队几个队员想一起在某个代码仓库或多个代码仓库，实现协作工作，但是又不想把代码放到公有云上。这个 Gitlab 私人服务器就相当于在模拟使用云开发平台前已有的本地开发环境。



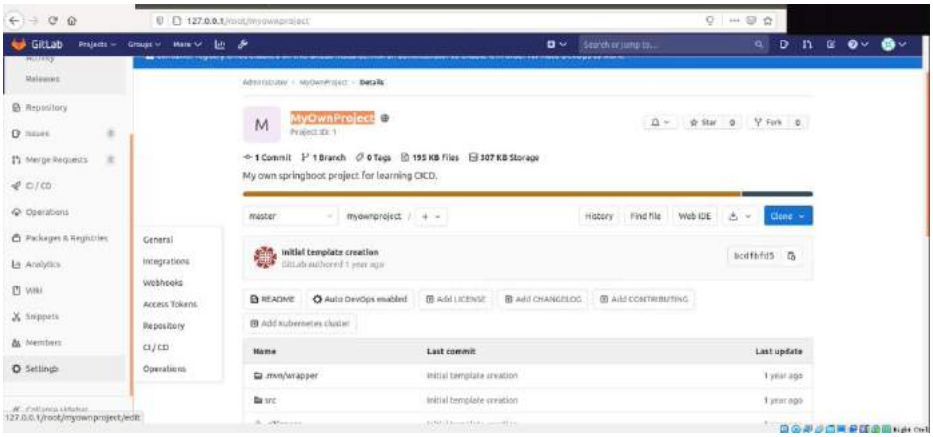
在 Gitlab 里创建的服务器跑起来之后，可以创建自己的工程，在这里我们创建一个 CI/CD 的工程来做演示。



如果想配置 CI/CD，第一步就是要在 Gitlab 里开启这个工程对于 CI/CD 的支持，也就是让 Gitlab 在 push 代码仓库的时候，Gitlab 会自动触发 CI/CD 的流程。

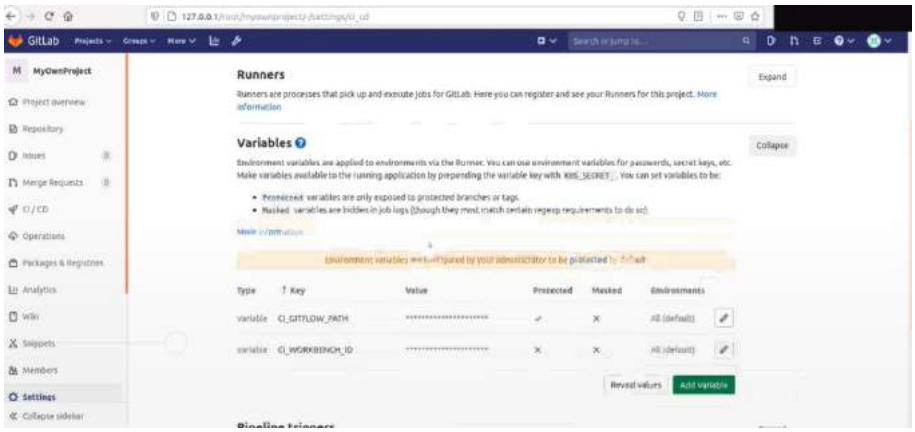


点开页面左侧最下面的设置键，会看到 Variables，这是 CI/CD 过程中会用到的变量，也就是跟这个仓库本身有关的变量。



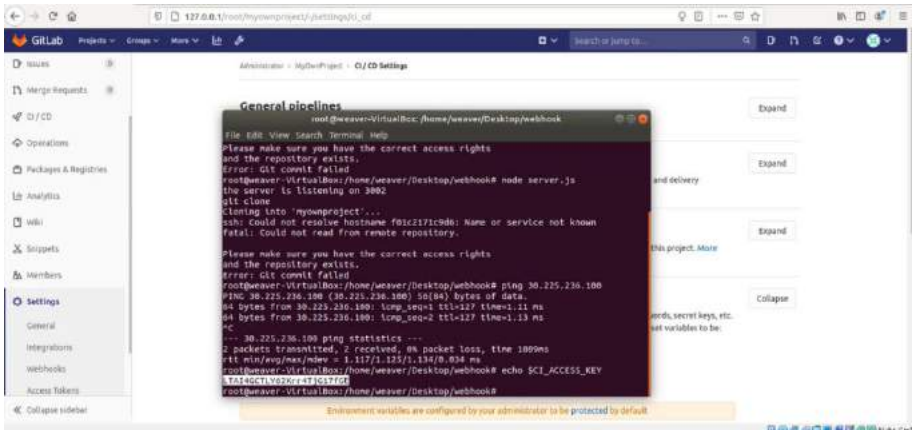
比如本地已经有个建好的仓库，若想把它部署到云开发平台上，那么需要在两者之间建立连接，而连接的方式就是通过云开发平台上的应用 ID，让仓库对应到云开发平台上已经创建的应用上。那么当围绕所有的本地已有的工程做一些部署，例如设置环境变量或设置团队成员等操作，云开发平台就都会知道。

第一步，设置 ID。在帮助文档中可以看到介绍，CI_WORKBENCH_ID 是云开发平台创建的工程 ID，与每个 gitlab 仓库相对应，因此需要针对 CI_WORKBENCH_ID 做单独的配置。通过设置键到 Variables 中进行设置。



第二步，配置自己的 AccessKey ID 和 AccessKey Secret。阿里云提供了一套 AccessKey ID 和 AccessKey Secret 的认证机制，每个同学都可以在自己的阿里云首页右上角找到 AccessKey 管理，点开管理就可以配置自己的 AccessKey ID 和 AccessKey Secret，AccessKey ID 和 AccessKey Secret 是你访问阿里云所有公共开放 API 的密钥，只要有 AccessKey ID 和 AccessKey Secret，就等价于有了用户名和密码，可以任意使用账号的云资源。因为本地仓库与阿里云的账户没有联系，不方便使用用户名和密码来让 webhook 使用阿里云的云资源进行部署，所以可以让 webhook 通过 ID 和 Secret 的这种方式调用阿里云的开放接口。

如果已经获取了 AccessKey ID 和 AccessKey Secret，那么 Gitlab 就能知道它该用何种方式套用云平台的开放接口，实现加强 CI/CD 的能力的需求。如何配置，可以参考帮助文档中的说明。

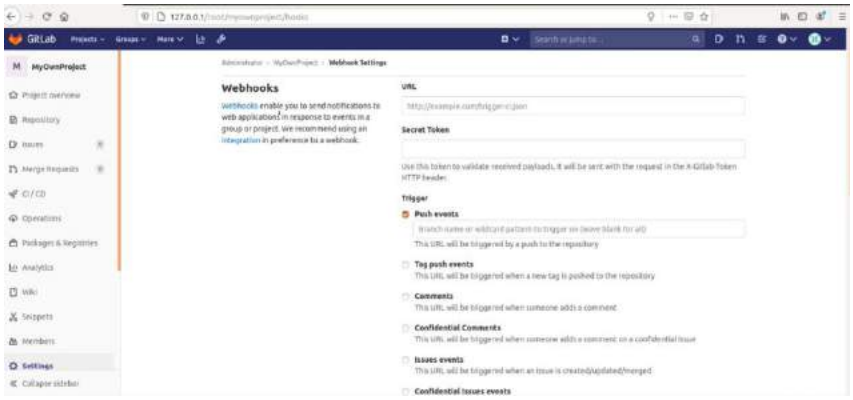


配置好之后，脚本就会自动识别到 AccessKey ID 和 AccessKey Secret，从而可以帮我们去用一些云开发平台提供的 OpenAPI 然后来加强本地 Hook 的能力。

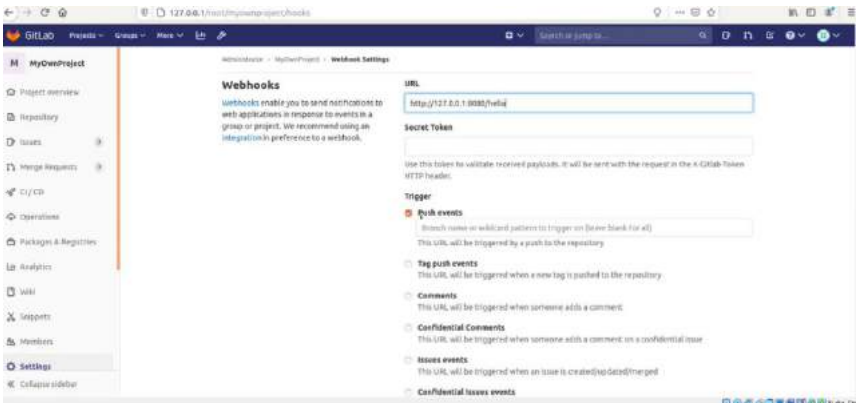
一、操作演示

前提是需要有一个 Gitlab 私服，或者有自己本地的其它 CI/CD 工具。有了这个前提就可以接着往下操作了。

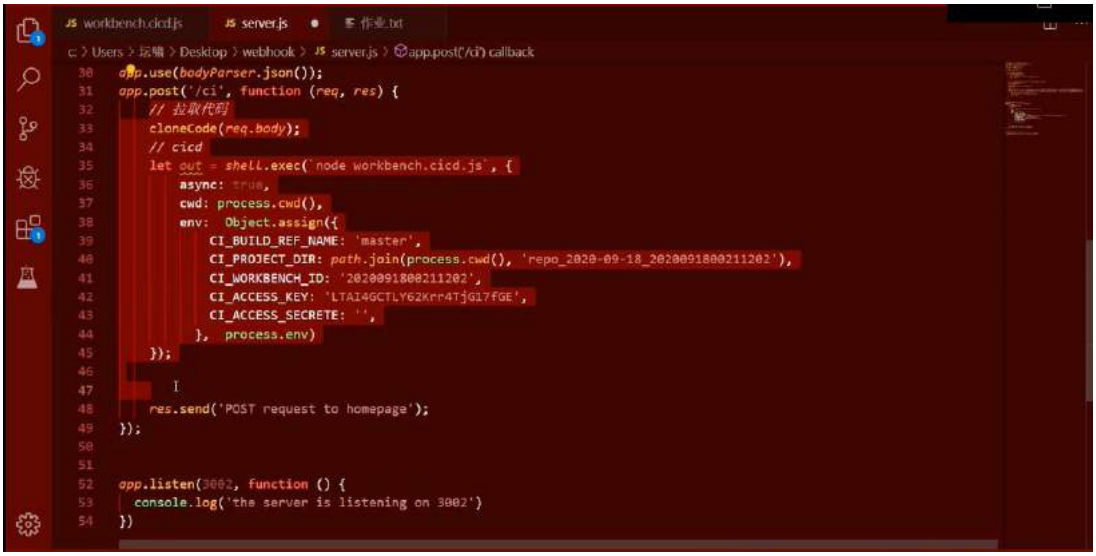
第一步，解压缩一个 Hook。Hook 可以用 Java 来写，也可以用 js 来写，它只是一个暴露的服务，通过一个预定义的端口来提供服务。然后在服务里想让它做什么都可以，比如想在脚本里执行一些记录日志、安全审核、发送警告等操作，都是可以的。我们就可以把执行开发部署指令写到 Hook 中。在帮助文档中有提供一个已经写好的 JavaScript Hook。



（演示）我们将 Gitlab 里面的 Hook 端口设置为 8080。写好之后往下拉，有个叫 Trigger 的触发器，勾选这个触发器下面的选项，那么就可以用选项对应的方式执行上面写的脚本的操作。比如选择 push event。让仓库在发生 push 时自动调用 Hook 来引起 CI/CD 流程。



（演示）给大家看我已经写好的一个脚本，这个是用 Node.js 写的。它的核心逻辑是对 3002 端口提供了监听，而监听里面只需要填一个路径，就是 Hook 脚本暴露出来的路径。每次调 CI 的时候，它都会执行如下图表述的逻辑，一共两步：第一步去 Gitlab 仓库拉取代码（这步可有可无，你也可以自己添加一些其它业务逻辑），第二步调 CI/CD 流程。

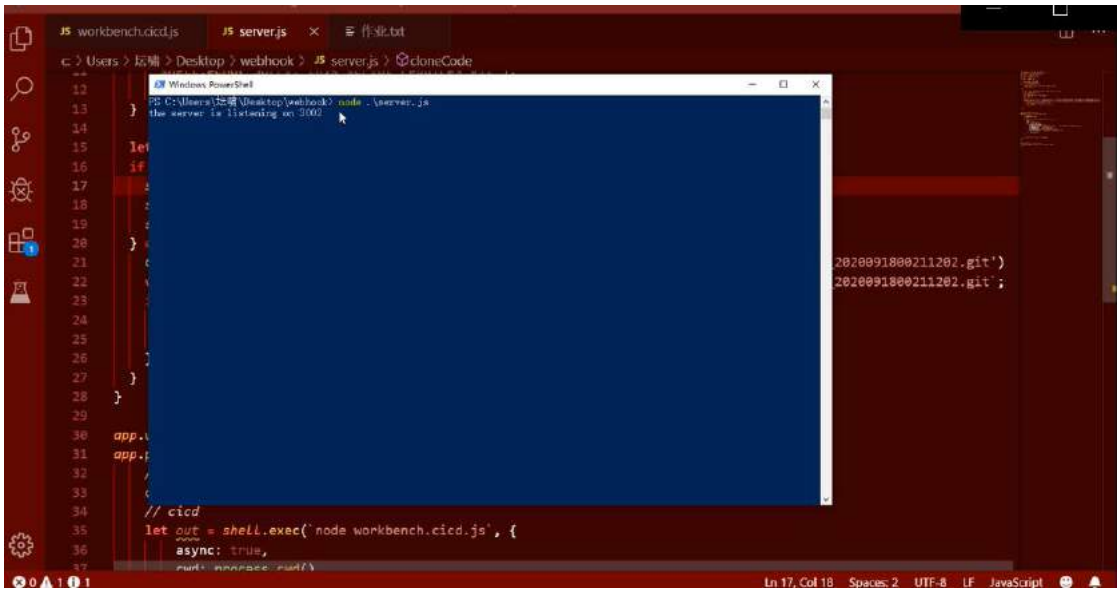


从这个逻辑里可以看出，第二步调 CI/CD 流程时执行了 workbench.cicd.js 这个脚本。这个脚本会完成从云开发平台控制台里面点击部署，然后从部署到打包代码，再把代码部署到云开发平台的一系列流程。

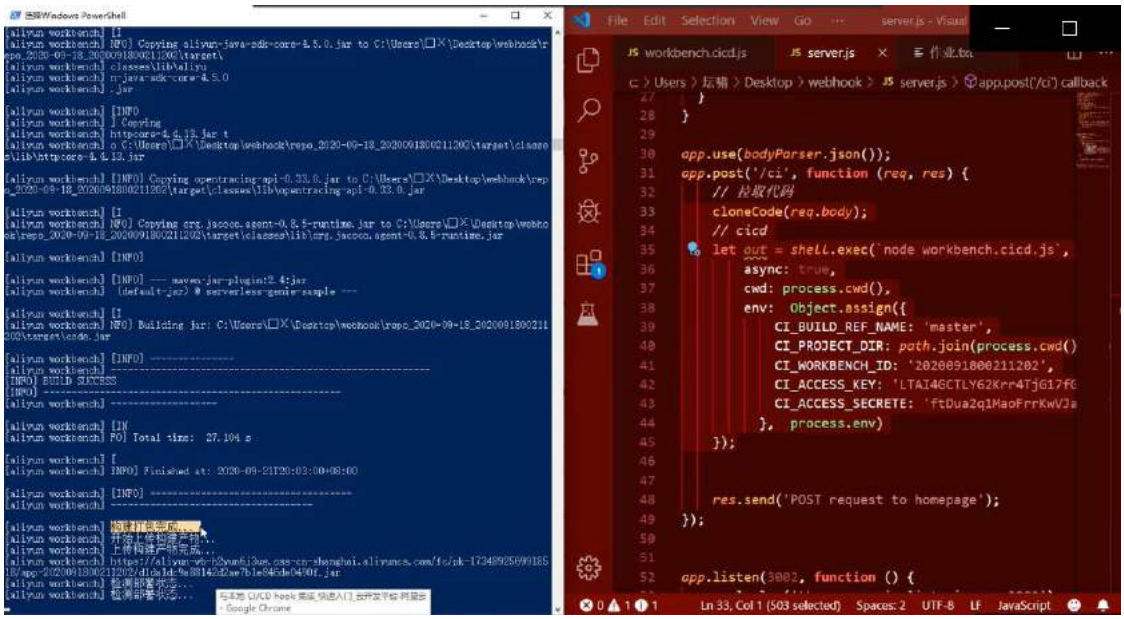
在帮助文档中，我们可以看到上面演示的关于 CI/CD 集成的步骤，第一步下载 CI/CD hook，它有两个文件：第一个是 workbench.js 脚本，第二个是 package.json 一个依赖。

二、测试演示

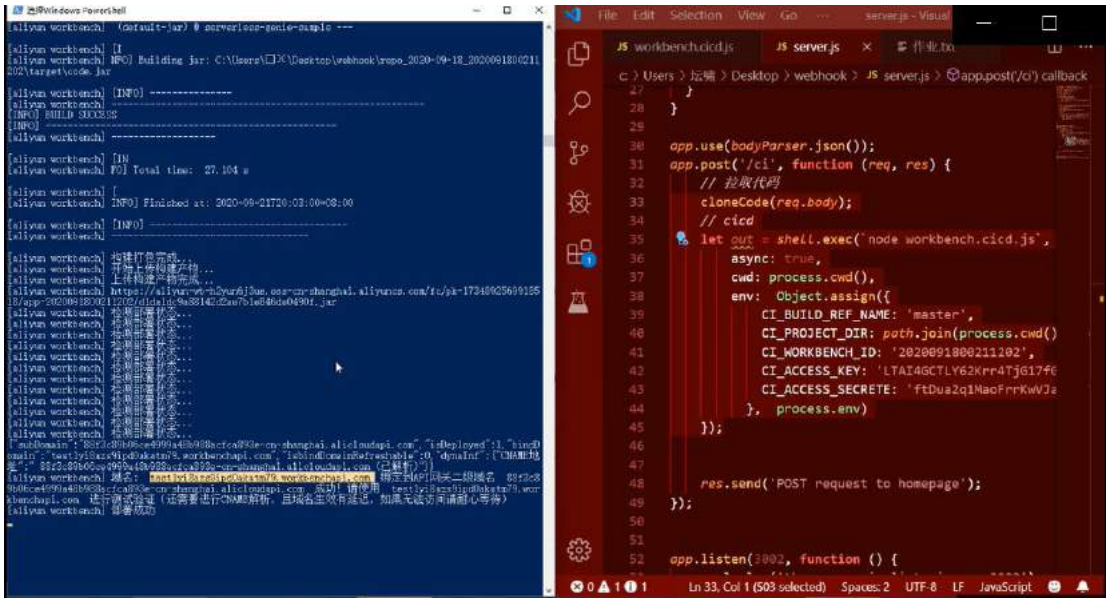
首先在本地把 server.js 脚本跑起来。这就是一个小型的服务器，它为本地开启这样一个端口，使得 Web Hook 执行 push 操作的时候可以通过这个端口调用服务。



然后点击 push event，发起一次发布。那么可以看到脚本已经开始执行 CI/CD 的工作了，可以看到他自动执行，跟从控制台里去点部署后的操作是一样的。



当这个域名它绑到 API 网关的二级域名，就说明部署成功了。



最后简单介绍下 CI/CD 的一些除了帮助完成“部署”操作之外的其它能力，通过帮助文档中的介绍可以获得更详细的信息。CI/CD 提供的大多是一些比较基础的功能，比如 GetAppInfo，它可以让你在 Hook 脚本中随时获取到当前在云端正在部署应用的一些信息，比如应用名称、域名、来自哪个解决方案或产品等等信息；DeployApp，这是一个部署应用的操作，当你把它加到 WebHook 流程里后，再触发 CI/CD 时，就也可以触发云开发平台的部署，也就是说 push 后，如果想把代码再部署上去，就可以在 DeployApp 接口中完成这个应用的部署操作。更多的功能大家可以去帮助文档看详细的介绍。

第五章 云开发实战篇

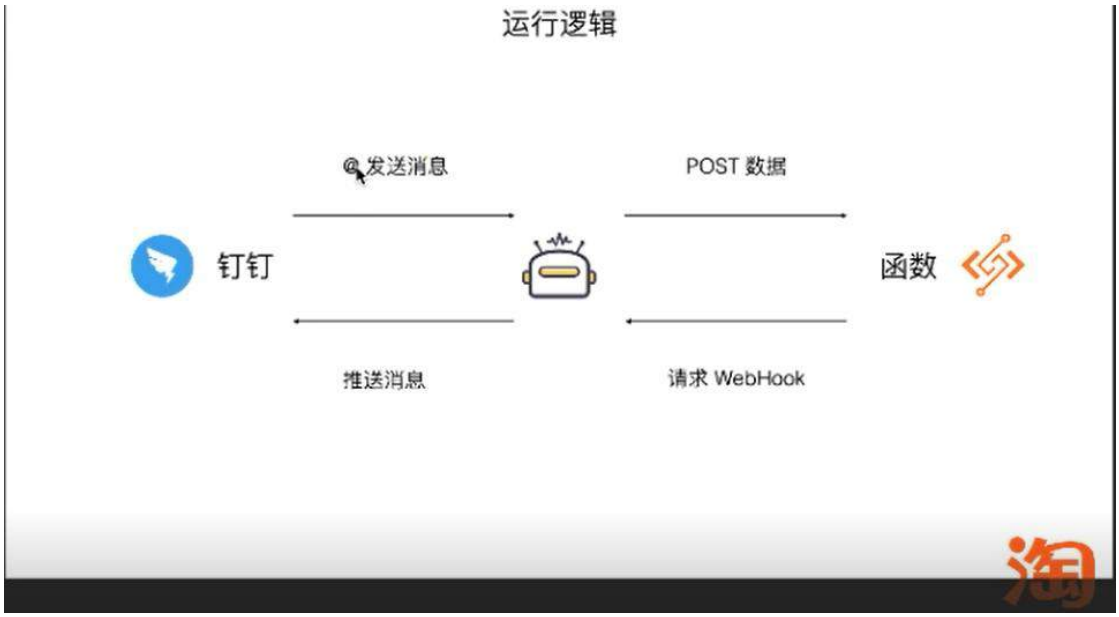
开发钉钉机器人

作者 | 洗剑 淘系 Node 架构技术专家

本篇主要讲解如何使用 Midway Serverless 进行函数的开发，并使用 Midway Serverless 开发一个钉钉的机器人。

想必大家都体验过钉钉机器人，举个例子，当我们在钉钉上@钉钉的机器人，提出天气的询问，机器人就会响应当下天气的情况。那么如何通过几行代码开发出你自己的钉钉机器人呢?

一、钉钉机器人的运行逻辑。

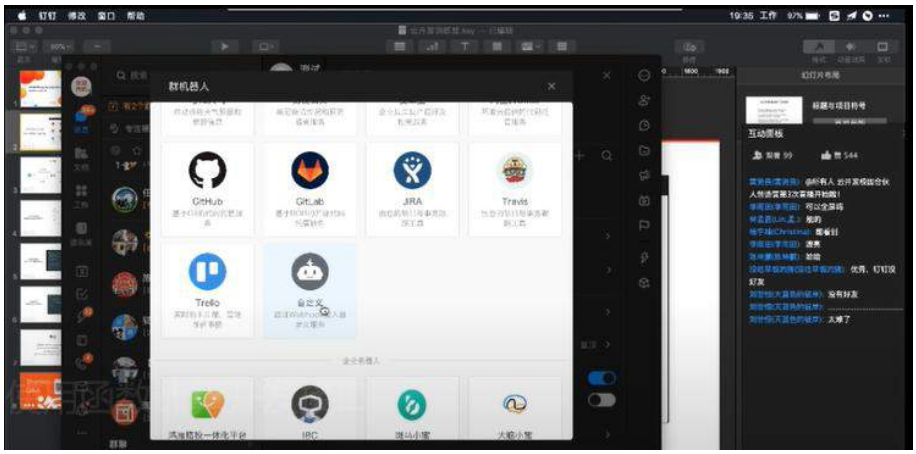


当我们在钉钉里通过@机器人发送消息，这个机器人就会通过 http 的 post 请求去把@他的文字等数据发送到阿里云上创建的函数里，函数接收到钉钉的数据请求之后，就会去解析这些数据，然后做处理。然后通过钉钉机器人的 WebHook，再把服务端的消息推送到用户的钉钉群里面。这就是一整套从钉钉群@机器人，然后收到机器人反馈的实现逻辑。

二、创建钉钉机器人的操作步骤

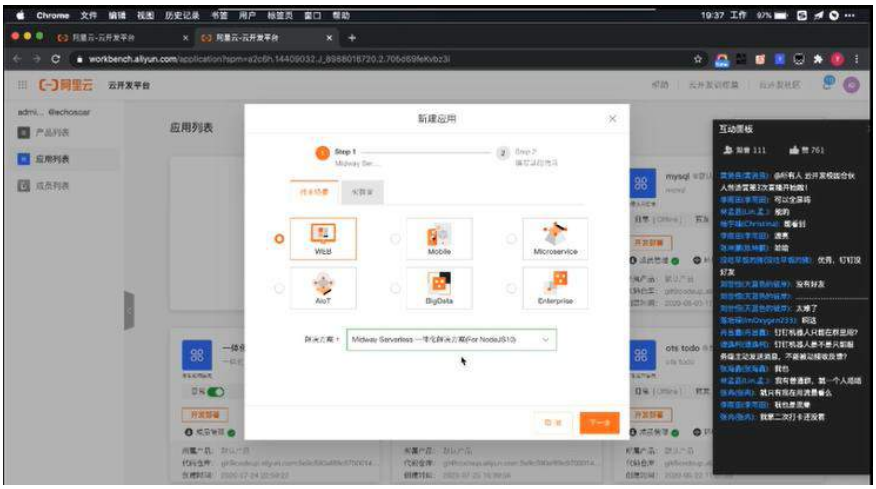
1. 创建钉钉群

在钉钉里创建一个群，因为钉钉群需要超过三人以上才可以，所以创建群的时候可以添加几个人进来，然后再把他们踢出去就可以进行下面的操作了。群创建完成后，添加一个群助手，即添加一个机器人。然后这里我们选择使用 WebHook 接入一个自定义机器人。



2. 创建应用

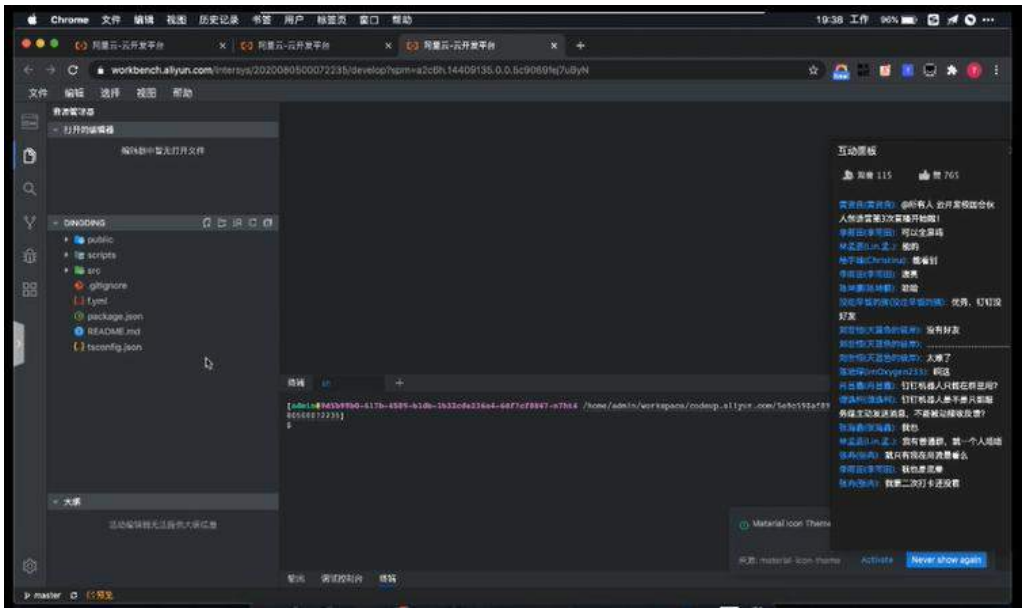
接下来在云开发平台使用 Midway 的模板来创建一个应用。打开云开发平台地址，http://workbench.aliyun.com，然后点击免费云开发创建应用，接下来在技术场景里选择 Web 场景，在下拉菜单里选择 Midway 一体化解决方案，然后就是使用 NodeJS 来进行开发，填写应用名称和应用介绍，点击完成。



点击完成之后，云开发平台就会自动创建此个应用的仓库，同时系统会把 Midway 代码通过模板发进去并自动生成。稍后出现开发部署的按钮，点击就可以进入 Web IDE 里进行纯云端的 Serverless 函数开发。

3. 部署开发

进入到 Web IDE 里，会看到整个代码项目目录已经创建好了。



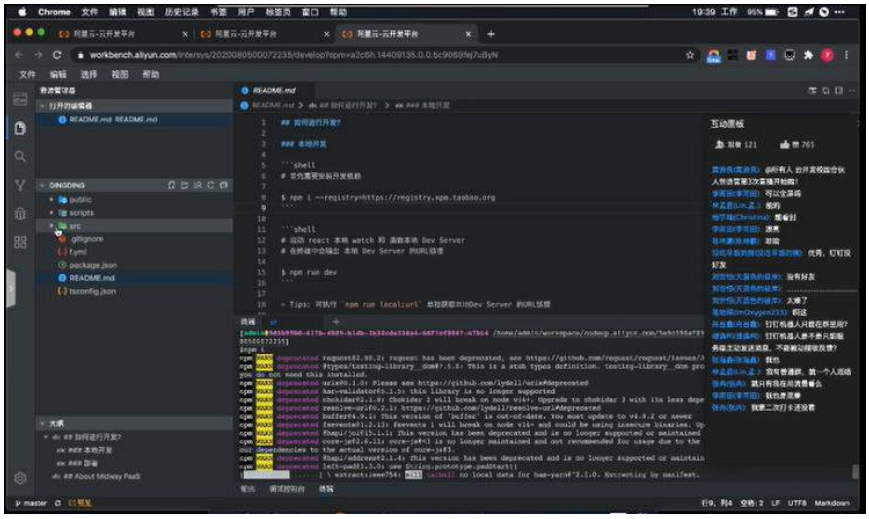
图解：Midway 的一体化应用的项目结构（可查看上篇内容获取详情）

src 目录中包含 tsx 代码，前端 react 的 JSX 的代码；apis 目录里面是后端的函数代码，即 Midway JS 框架的后端函数。在同一应用里可同时进行前后端的一体化开发。开发前可编写前端 UI 界面，然后编写后端的函数，前端的界面可以直接通过 http 请求阿里云在本地的开发函数。

Midway 框架是基于阿里云 NodeJS10 环境的，Midway 框架提供了 4 个函数，他们分别是：render、index、detail 和 list。render 绑定的路径是/*，意思是匹配所有请求的路径都会导到函数上面来；index、detail 和 list 是分别是三个 API 接口。然后当路径请求进来的时候，首先会去进行这种非通配符的路径匹配，如果匹配到了 index 接口那就执行 index 的函数，然后把 index 的返回值结果进行响应。如果路径没有匹配到，它就会通过通配符/*，到 render 函数里面。

点开 README，根据 README 里面的提示如何进行本地开发。

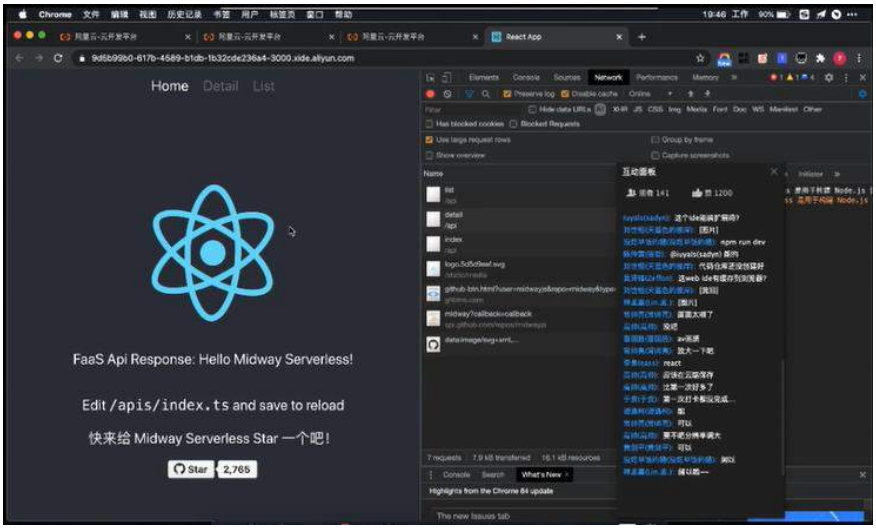
首先在终端使用 npm install 命令安装前端依赖。阿里云云开发平台为了提高大家的安装的速度，终端里是默认集成了淘宝的 npm 镜像的，所以大家可以直接使用 npm install。



4. 启动本地开发 server

安装完成后执行 npm run dev ，就可以启动一个本地开发的 server。 这个时候会启动前端 react 和后端 Midway 的本地 Dev Server。因为是一体化开发，所以前后端拥有同一个端口，可以同时接收前后端的各种请求。

在本地 Dev Server 启动起来后，可以通过本地的开发服务器打开正在开发的项目。



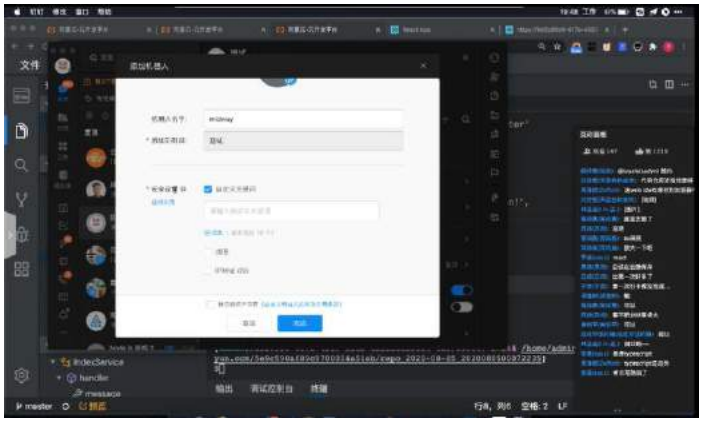
图解：前端后端项目展示

在 src 目录下面有 APIS 目录，APIS 目录里面其实就是我们整个后端的接口代码。比如前端首页的 Hello Midway Serverless，是通过 index 接口来响应返回的。当我们在后端进行一些修改并保存，刷新后会同步到前端页面。

5. 创建钉钉机器人

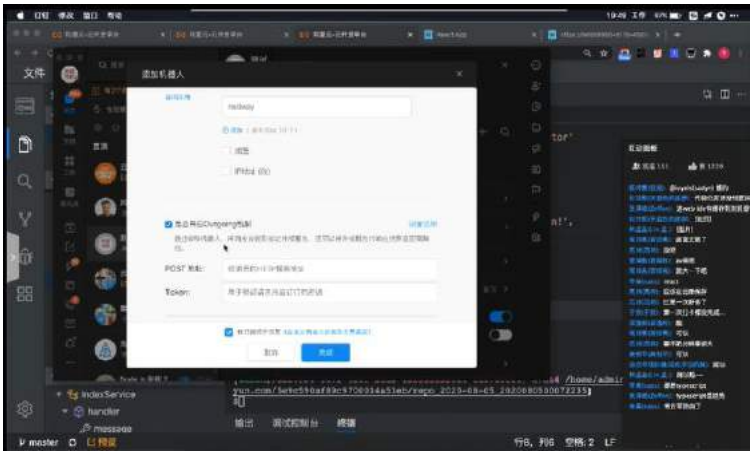
打开创建的钉钉群，通过智能群助手添加机器人。

点击添加机器人后，选择自定义机器人。添加完成前可根据需求填写机器人名字并进行安全设置。安全设置可以保护创建机器人不被盗用，所以作为创建者在安全设置这里添加一个自定义的关键词，那么只有设定的关键词出现时，钉钉才会去把消息推送到群里。



然后开启 outgoing，outgoing 机制是为了在群里@机器人时，它能够根据适配 POS 地址，接收请求并反馈群消息。

最后点击同意完成创建。



点击完成后，页面会出现一个 Webhook。为了实现能够把消息推送到群里，需要通过向 Webhook 发一个 post 请求。

三、后附：课程 FAQ

1. 如何在函数内获取 http post 的请求的 body 呢?

在 Midway 的框架里有 inject 的 LC 注入的装饰器，它可以实现把上下文注入到函数的时间点里。通过上下文，可以获取 inject 从上面取到 http 请求的 body。

2. 如何在函数内向钉钉机器人发送消息?

首先发送一个 post 请求，然后用 markdown 定义消息的类型和内容

3. 为什么要安装 axios?

通过 axios 的包发送 http 请求，http 是在 Node 里，我们也可以通过原生的 http 一起发送请求，但需要通过写代码做一些比如什么时候发送数据，反馈什么信息之类的设置。

4. axios 的安装命令是什么?

安装命令：npm i axios--save（i 是 instyle 的缩写，s 是 save 的缩写），这样就可以安装到项目里。

5. 如何理解 ctx 的上下文?

当你请求使用 Midway 框架时，Midway 框架会把你的 http 的请求通过转化成为我们熟知的 qua 请求。然后我们把请求的 request 创建上下文对象，可以理解为创建一个 object 并把请求对象挂载到 ctx 上面。同时我们也会把一些 url 参数请求的比如说 this.ctx.query，挂到 ctx 上面。这样，大家就可以通过 ctx 去快速的取到 http 请求里的内容。

当写完一个函数要返回数据时，也可以通过 ctx.body 把响应的数据挂载到 http 的 response 上去。直接用 return 一个字符串也可以得到同样的效果。

6. @是如何变成请求的?

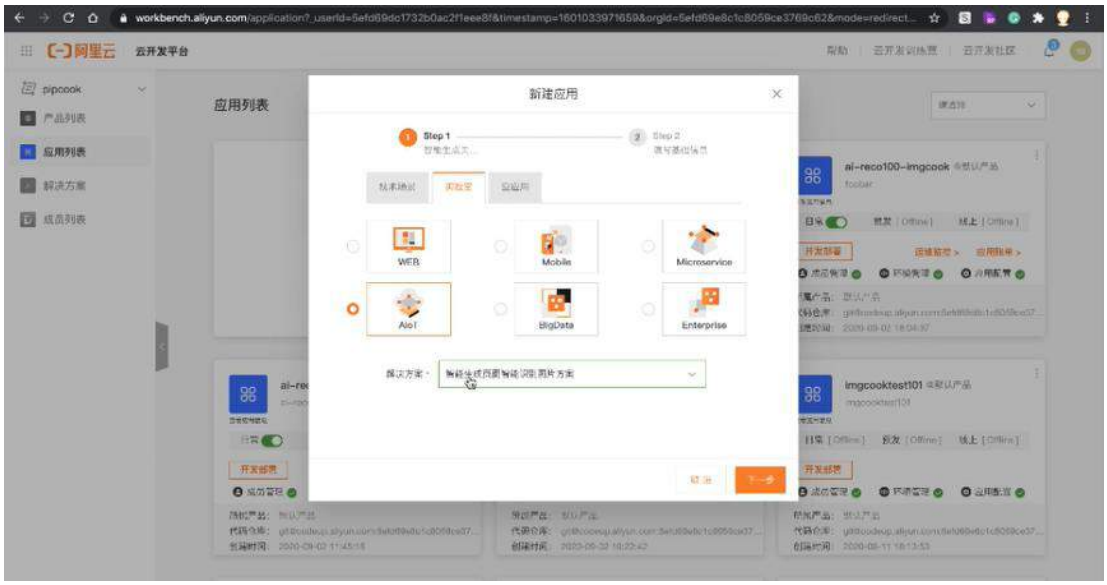
如上图显示，在钉钉群里@机器人的时候，@机器人的那段文字，会通过 post http 请求把数据发送到函数里。然后通过函数里的处理，再通过钉钉机器人的 Webhook，把这个消息再反回钉钉，然后再返回到钉钉机器人。钉钉机器人就会把这个消息再推送到钉钉的群里。

智能生产代码与 AI 应用的开发实践

作者 | 雷姆 淘系前端技术专家

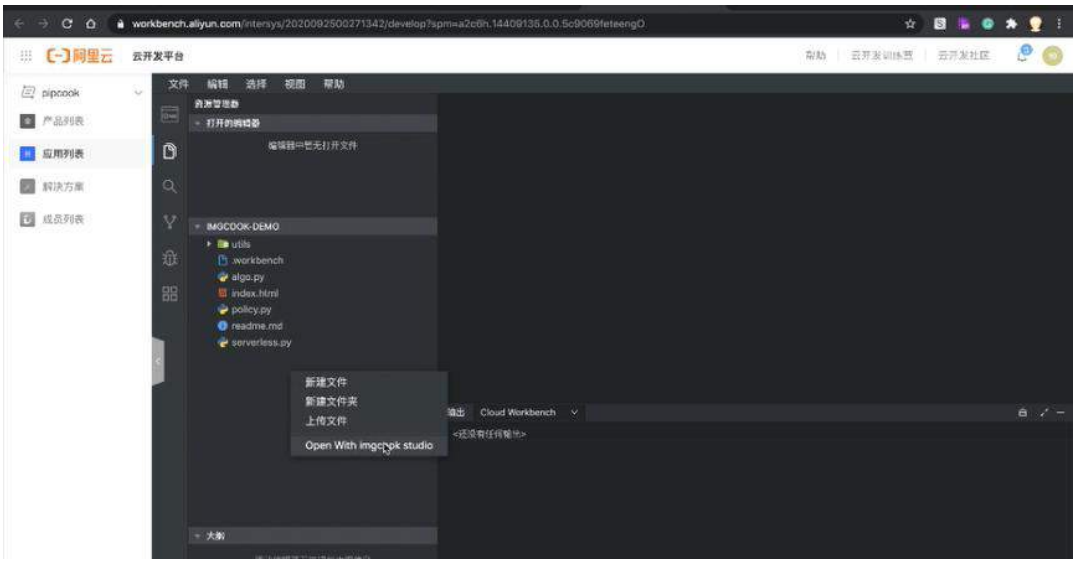
本篇内容将向大家展示如何用 ImgCook 加 workbench 平台完成智能生成垃圾分类的页面。大家也可以在云开发平台帮助文档中找到《智能生成页面智能识别图案方案》了解具体信息。

首先在 Serverless 云开发平台上创建应用。创建应用的时候选择“实验室”，然后选 AIoT，然后下拉选择智能生成页面智能识别图片方案，补充好应用名称等点击完成。

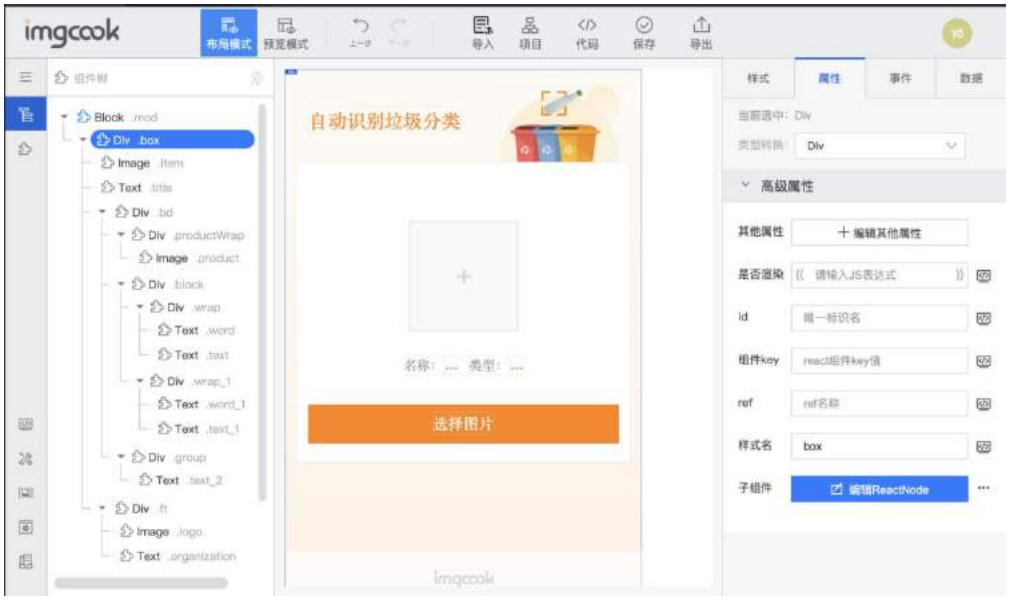


进入开发前先介绍下帮助文档中提及的“下载测试用的 Sketch 设计稿”。这个设计稿需要打开 Cloud IDE 再下载“垃圾分类设计稿 Sketch 文档”到本地。如下图所示，这个设计稿是一个静态页面，进入这个页面后点击选择图片的按钮，它会弹出一个选择文件的一个对话框，然后选择对应的图片，它就会把这个图片对应的垃圾的分类类型显示出来。

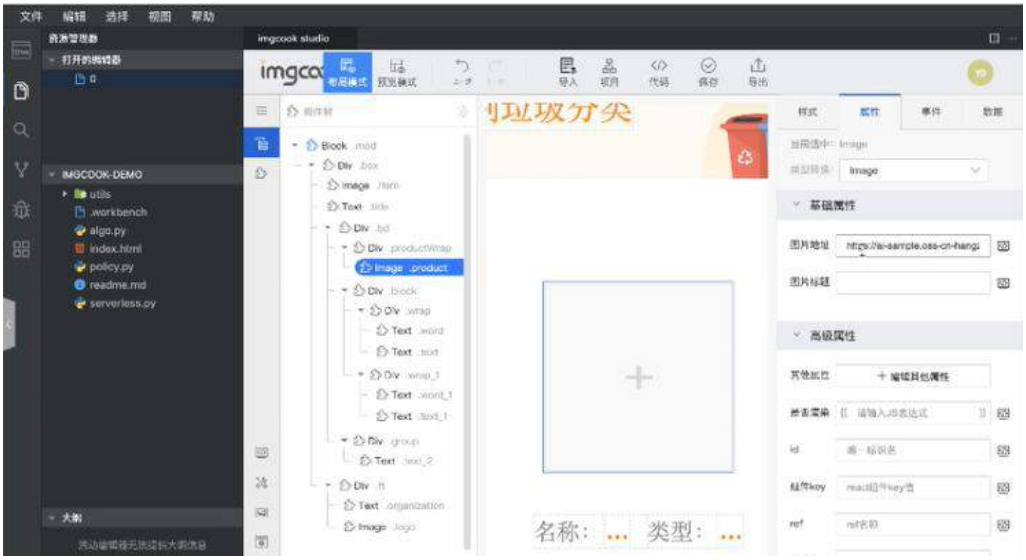
创建好应用后就可以进入开发，在左侧目录文件区域点击右键，出现弹窗选择最下面的“Open With imgcook studio”，就打开了 imgcook 的页面了。



打开 ImgCook 页面后，点击上侧的“导入“键，选择上文介绍的设计稿，上传后选择导出。导出后，设计稿代码就被自动生成出来了，在 ImgCook 的工作界面会展示出来由代码生成的可视化 UI 效果。



代码生成后通过左侧展开对应设计稿的段数，点击某个节点，右侧就会展示出与节点对应的详细信息。比如点击图片节点，右侧会显示图片的属性和一些相关的配置。我们也可以通过这个部分对设计稿某些节点进行修改。



添加页面交互逻辑，绑定后端 API。

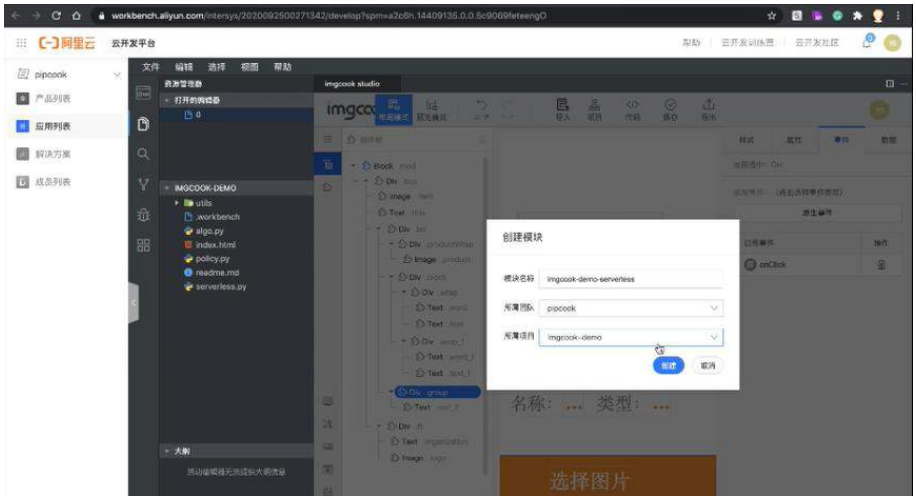
接下来我们要做的是完成用户交互操作。首先，点击生成 UI 界面上的+，在右侧高级属性面板，将「id」属性值设置为 image，然后点击+下面名称和类型旁边的…补充名称、类型等内容。

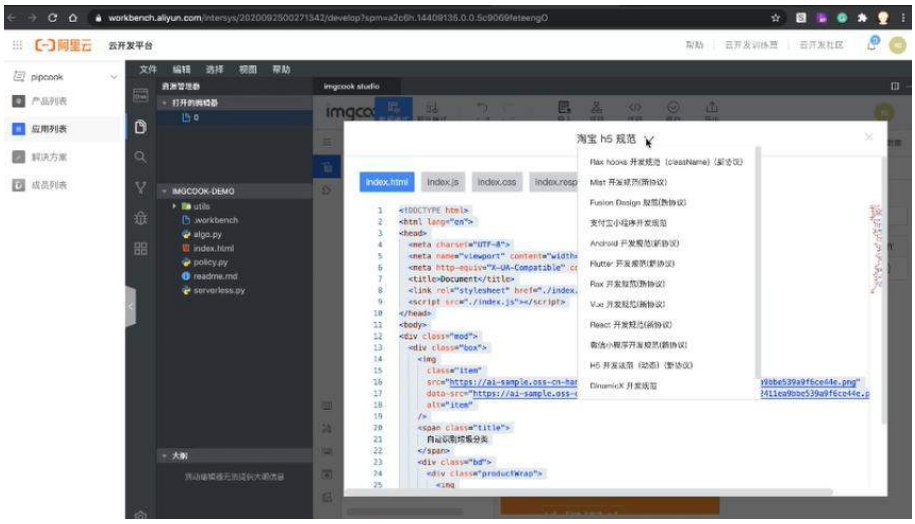
所有设置完成后，点击“选择图片”并在右侧事件中添加 onClick 事件。

点击 onClick 后，在弹出的输入框，输入代码作为处理函数。所需的代码可以通过帮助文档《智能生成页面智能识别图案方案》查看并复制使用。

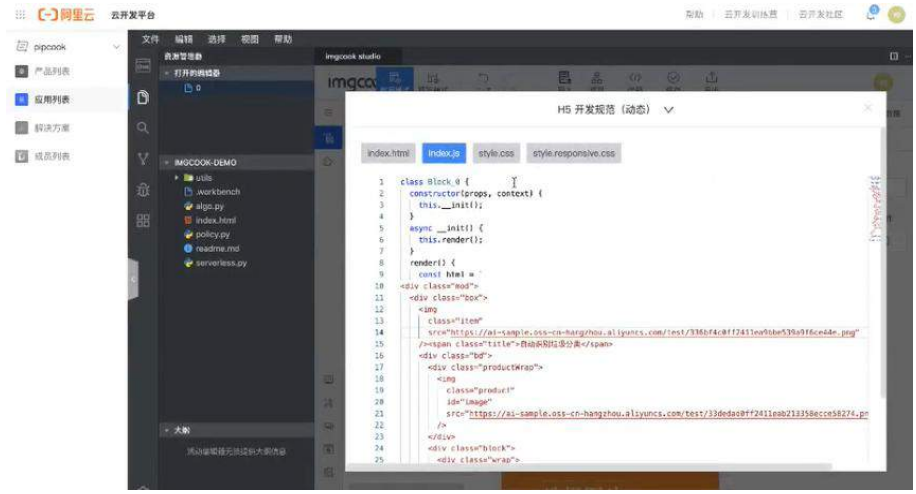
粘贴进去函数之后，保存。那么代码生成的准备工作就基本完成了。

接下来保存项目，并选择对应的开发规范。ImgCook 是可以根据不同的需求生成不同的规范的代码。示例展示的是选择了纯 h5 代码生成。

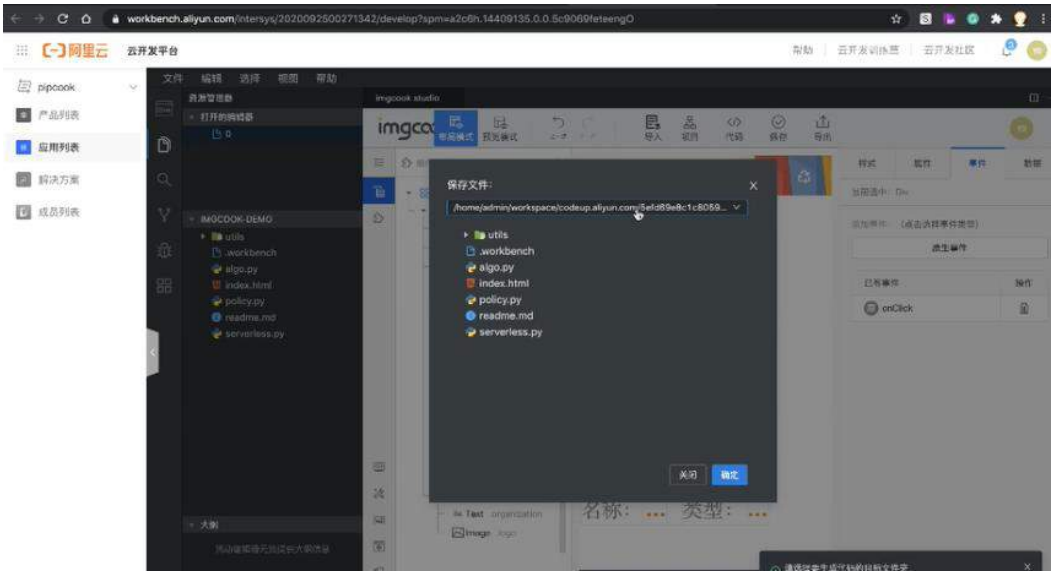




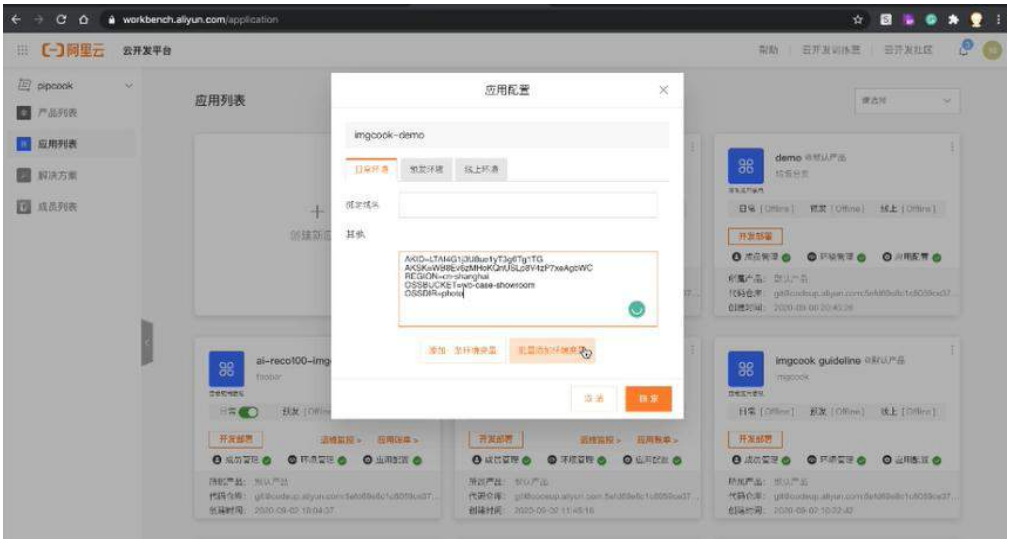
选择代码之后可以看到对应的几个文件，index.html，index.js，style.css 和 style.responsive.css。



最后是选择导出到指定的目录。选择导出，会跳出一个可供选择的目录框，如果不做选择会默认保存到根目录下。

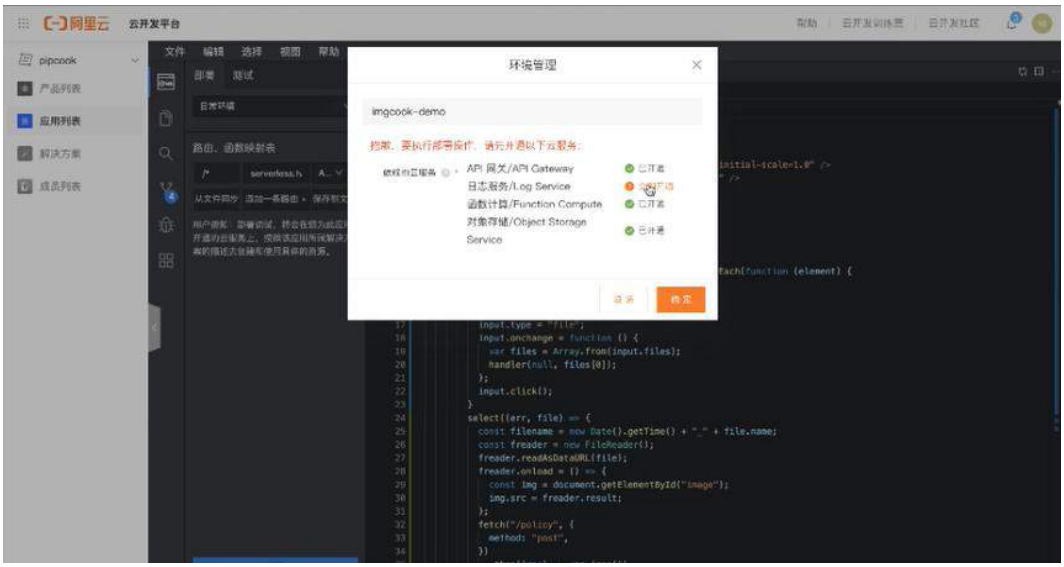


保存后，需要为此应用输入测试所需的图片上传和图片智能识别的环境变量。输入的代码可以通过帮助文档获取。配置的方法是点开应用配置，然后将在帮助文档中复制的代码粘贴进去，选择确定完成。

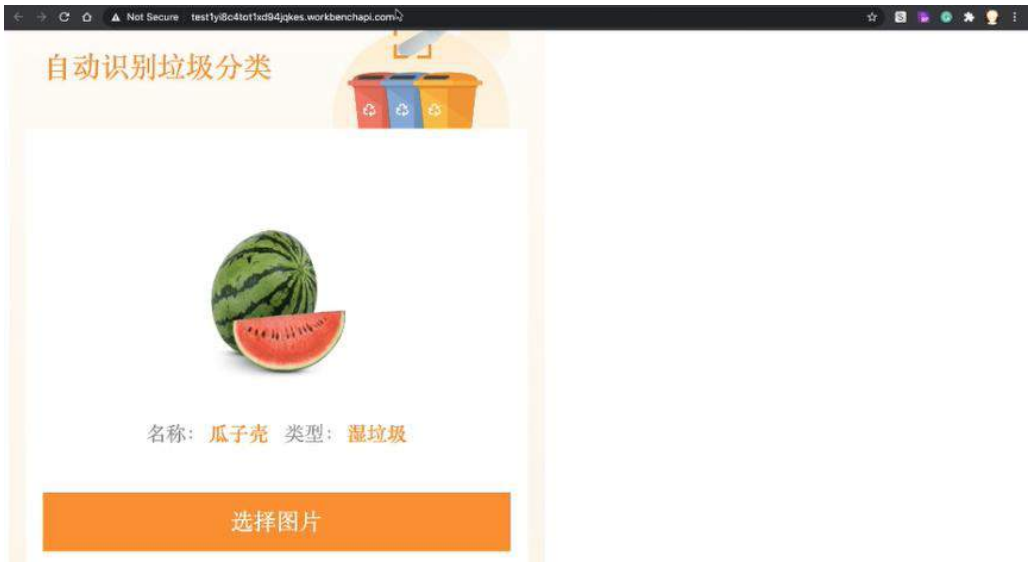


最后就是可以部署并检验效果了。

回到 Cloud IDE 页面，点击左侧的 WB 插件，然后点击部署。



完成部署后，可以通过分配的域名点击查看并测试。



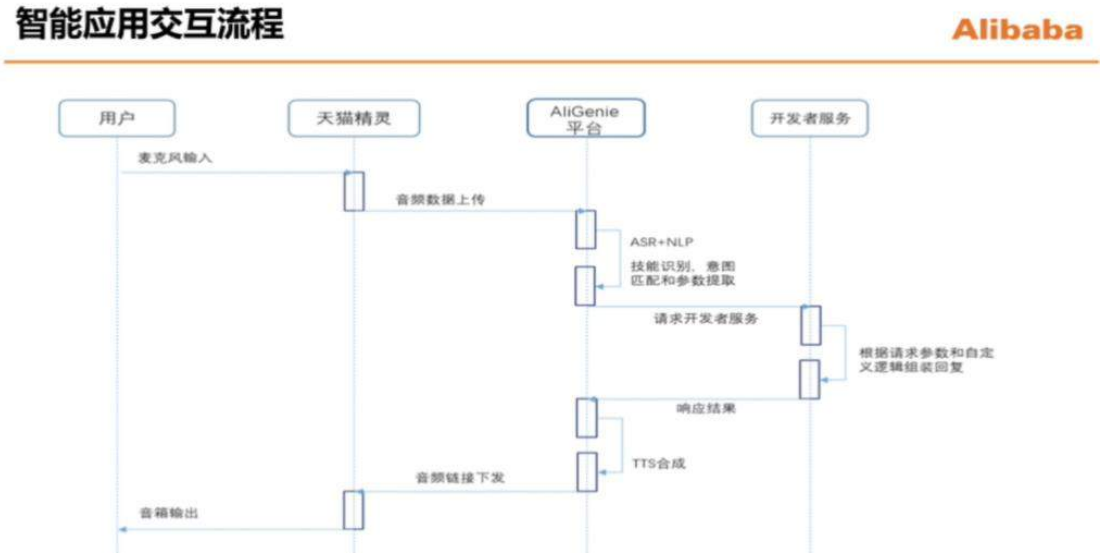
最后提醒大家，因为上文使用的环境变量是测试用的，那么在真正使用的时候需要把环境变量配置的数值切换为自己的阿里云账号参数。

快速开发天猫精灵智能应用问答百科

作者 | 尘空 天猫精灵高级开发工程师

本文主要介绍天猫精灵智能应用云原生开发模式及如何使用云原生开发模式开发一个智能应用。

我们都知道天猫精灵具有语音互动和问答的能力，那么语音交互的流程是怎样的呢？



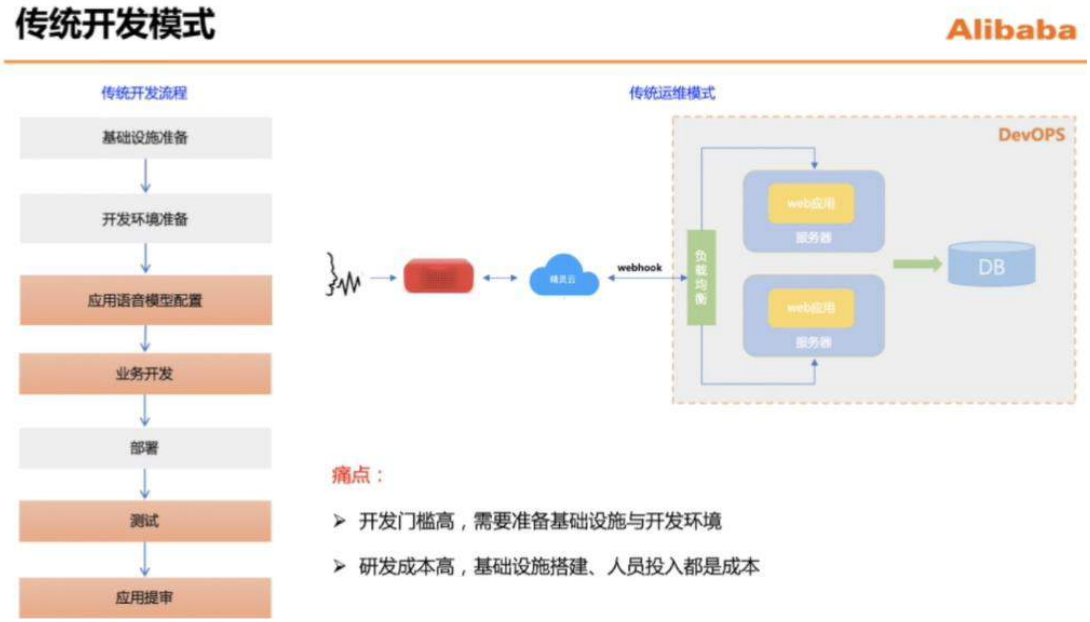
如上图所示，精灵平台会把用户说的话转成文字，再加上自然语言处理，识别出对应的意图，提取出对应的参数，最后传给开发者。开发者根据业务逻辑进行处理，把对应的结果返回给精灵平台，精灵平台对返回的信息做 TTS 合成，把对应的音频链接下发给天猫精灵，天猫精灵会把对应的结果汇报给用户。这就是整个智能应用的交互流程。

一、传统开发模式

在介绍云开发模式之前，先来看看传统开发模式。

1. 传统的开发流程

首先需要做好基础设施的准备，这里包括服务器、域名等等；然后是开发环境的准备，例如 ID 的安装、运行环境配置、代码框架搭建等等；接着需要配置语音模型，再然后是业务开发、部署，最后是应用提审。传统的开发流程对开发者来说门槛比较高，因为需要去准备好基础设施与开发环境。



2. 传统运维模式

传统运维模式提倡 DevOPS，希望通过一系列工具和自动化技术来降低运维难度，促进研发运维一体化，提升软件的开发效率，缩短软件的交付周期。

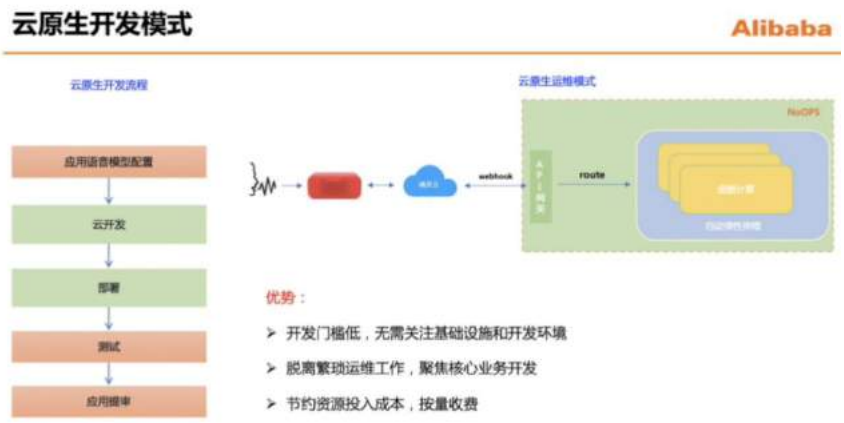
总的来说，传统开发模式有两大痛点：开发门槛高和研发成本高。

二、云原生开发模式

1. 云原生开发流程

云原生开发流程是直接进入智能应用开发平台进行语音模型的配置，然后进入云开发去开发业务代码，然后是部署和测试，最后是应用提审。

在云开发的流程中不需要关注基础设施和开发环境，所以对开发者来说开发门槛很低。



2. 云原生运维模式

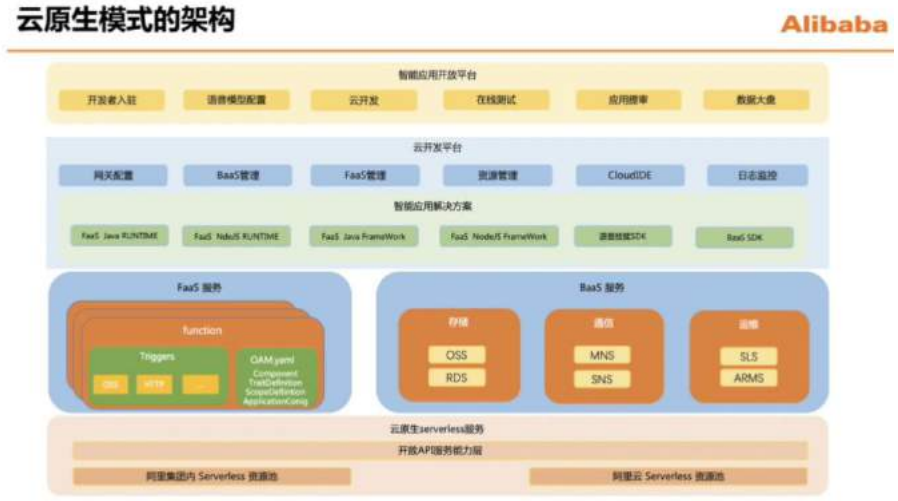
云原生运维模式提倡的是 noOps 理念，它会根据流量的大小自动地弹性伸缩。

云原生开发模式的优势是：

- 开发门槛低，无需关注基础设施和开发环境；
- 以帮助开发者脱离繁琐的运维工作，聚焦核心的业务开发；
- 云原生的运维模式是按量收费的，帮助企业节约资源投入成本。尤其是初创企业的业务试错期，它的流量非常低甚至可以达到免费。

云原生模式的架构

最上层是智能应用开放平台，它提供了智能应用的全生命周期管理能力。其中的云开发主要依托于天猫精灵与阿里云合作共建的云开发平台。



向下一层的云开发平台，主要提供了 CloudIDE、资源管理、日志监控等能力。当前也为天猫精灵提供了 NodeJS 和 Java 两种语言的解决方案，并且集成了语音技能 SDK，帮助开发者快速开发智能应用。

再向下一层是 FaaS 服务和 BaaS 服务。FaaS 服务是通过触发器触发的，在智能应用的场景中，主要使用的是 HTTP 触发器；BaaS 服务主要提供了一些存储能力、通信能力和运维能力，这些能力主要通过 BaaS SDK 进行调用。

最底层阿里集团和阿里云的 Serverless 资源池，它主要通过开放 API 服务能力层进行调度。

以上就是整个云原生的架构。

三、何使用云原生开发模式进行开发

以问答百科为例。

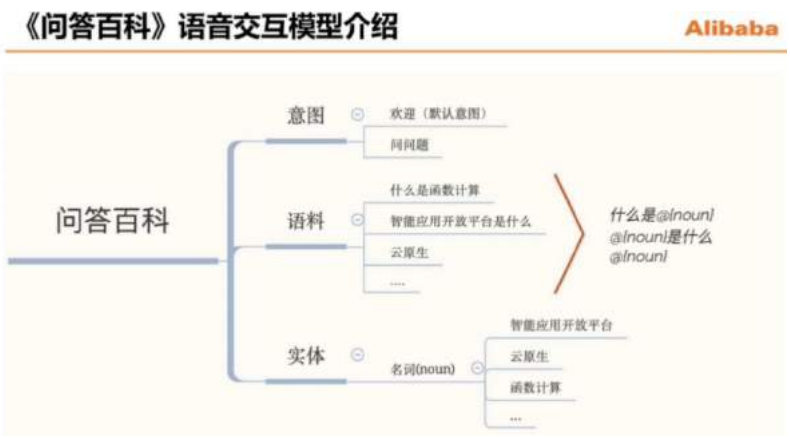
1. 问答百科的交互模型

- 意图介绍

问答百科有两个意图，当唤醒问答百科时，进入的是欢迎意图；当向问答百科问问题时，进入的是问题的意图。

- 语料介绍

语料是指用户说的话，比如用户可能问什么是函数计算，意图就是不同语料所代表的同一种意思。



- 实体介绍

语料当中像“函数计算”、“智能应用平台”都是可以被替换的，是作为参数可以被提取的。那么这些词的集合就是一个实体，实体是指某一类名词的集合，相当于提供了有限定范围的词典。

在问答百科当中，这些词都是可以根据知识库的范围去动态补充的。

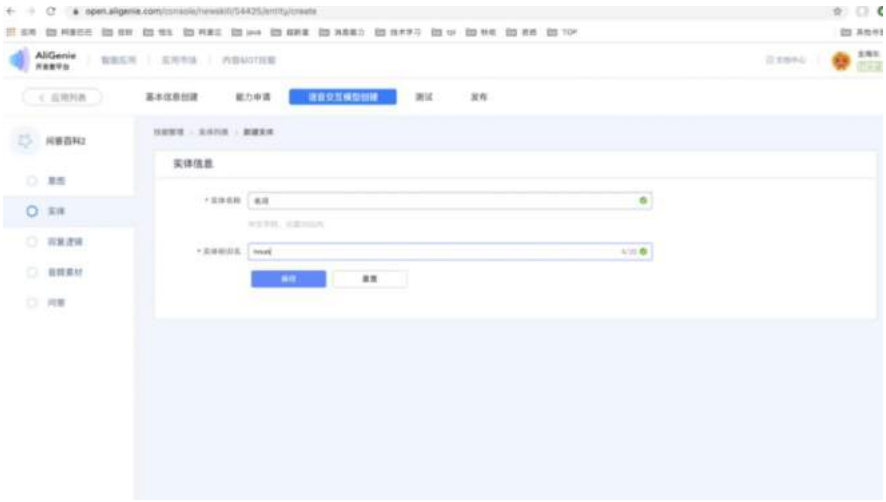
通过语料和实体就可以拼装出一个语料模板，如上图右侧所示“什么是@[noun]”“@[noun]是什么”

2. 如何使用云原生模式进行开发

- 创建语音交互模型

第一步，创建语音技能。打开智能应用开放平台，创建一个语音技能问答百科，设置一些标签，点击确认创建。

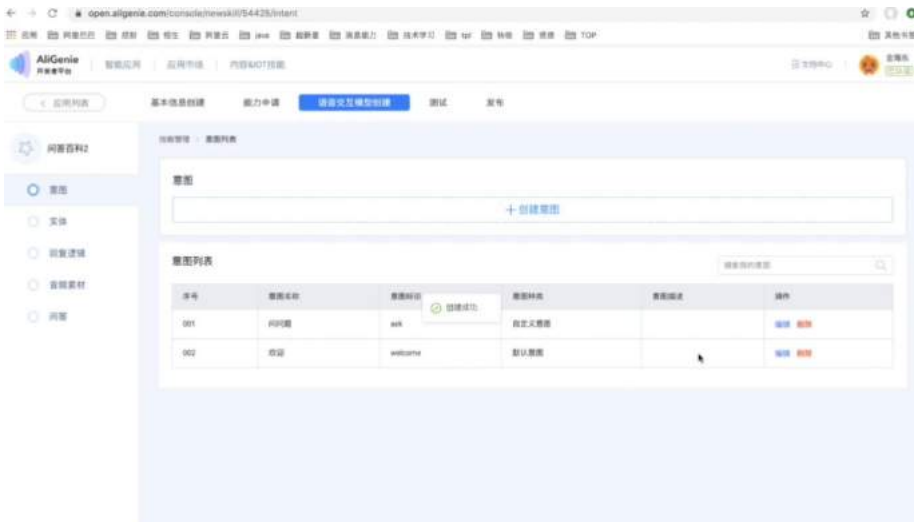
第二步，创建语音交互模型。首先创建实体，补充实体名称并做好标识（noun），完成创建。



第三步，创建意图。

首先是欢迎意图，它的标识是 welcome，再将欢迎意图设置为默认意图。那么当唤醒问答百科的时候就会默认进入欢迎意图。

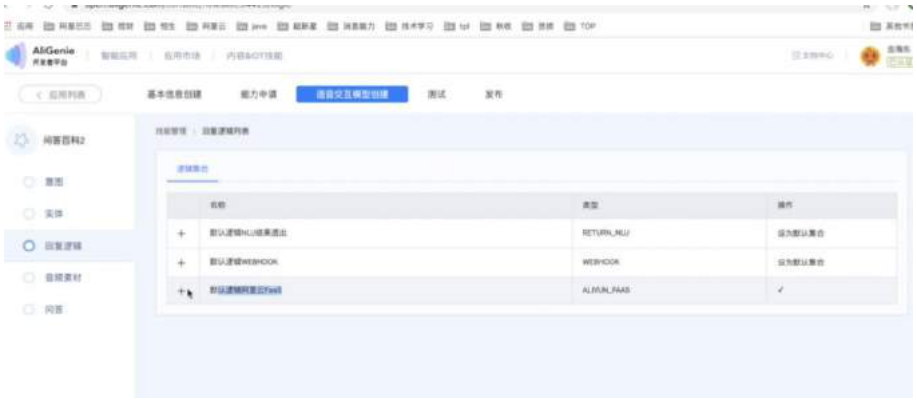
其次是问题的意图。当向天猫精灵问问题的时候，默认进入的是问题意图。创建完成，再配置问题意图对应的语料，比如说“什么是函数计算”、“智能应用开放平台是什么”等等。那么“函数计算”和“智能应用开放平台”都是可以被提取的参数，是可以被替换的，所以把它标注为实体。



这样，整个语音交互模型就创建完成了。

- 设置回复逻辑

由于使用的是云原生开发模式，所以需要设置的回复逻辑为默认阿里云 FaaS，将它设置为默认集合，然后去开通。



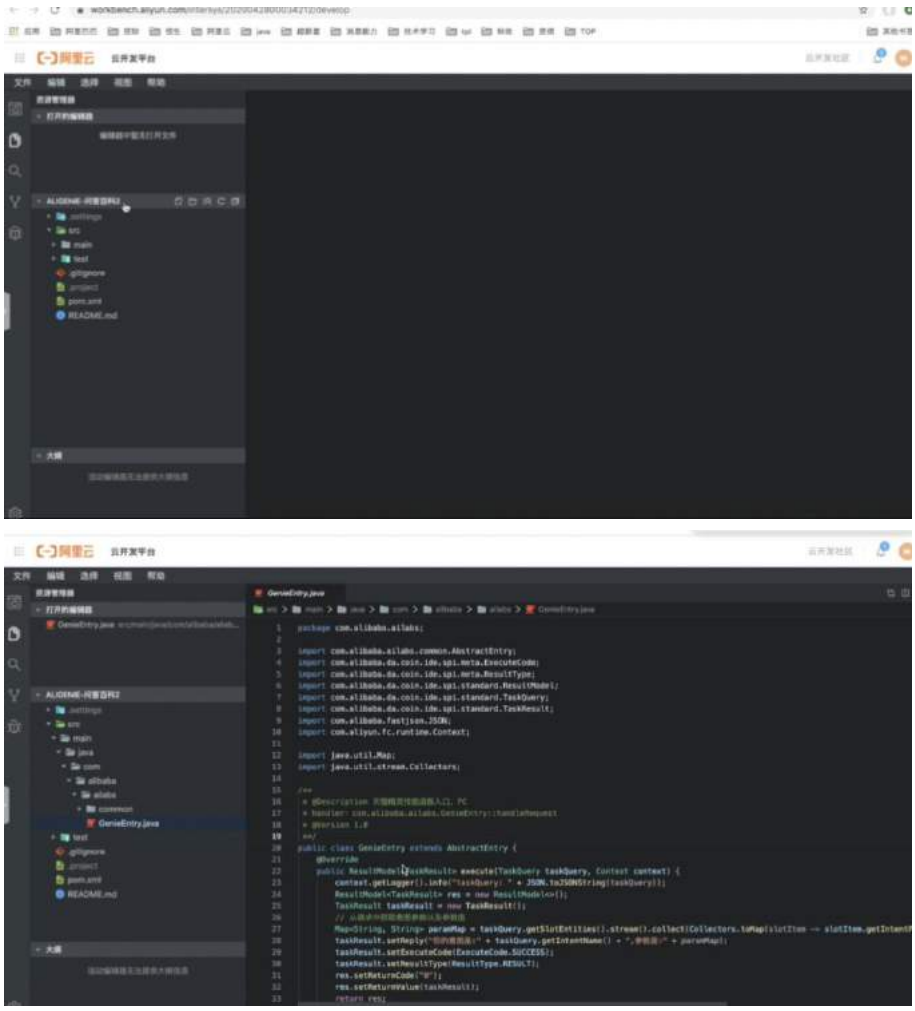
提醒：如果是首次使用阿里云 FaaS 进行开发，建议先阅读一下新手引导文档，文档中有关于如何使用阿里云 FaaS 进行开发的详细介绍。

点击开通后，登录到云开发平台。

点击刚刚创建的智能应用下面的代码仓库地址（也支持进行本地开发），在弹出窗口，生成 SSH 密钥，然后进行绑定。绑定完成后，就可以通过本地下载代码进行开发。

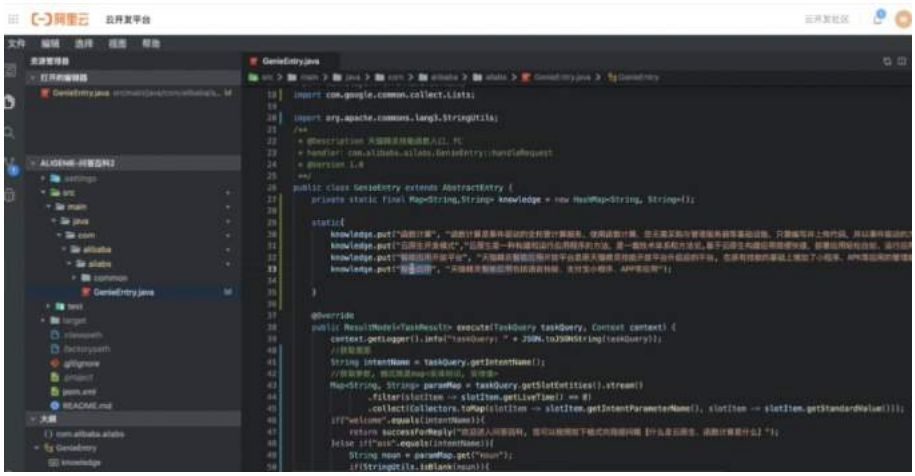
由于此次介绍的是云开发的模式，所以可以直接使用 Cloud IDE 进行开发。

那么直接点击“进入开发”，打开一个 Cloud IDE 后会加载一些插件，然后下载初始化代码。如下图所示可以看到左侧的项目代码结构。



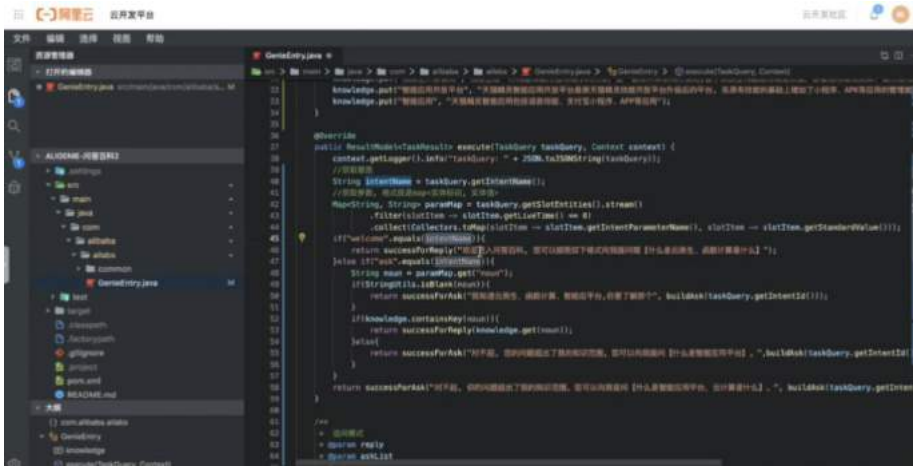
通过上图右侧的初始代码就可以直接进行语音技能 SDK 开发。

问答百科的逻辑是首先要有一个数据库去存储知识，如下示例是通过 map 存储知识，mapkey 就是实体值，比如“函数计算”、“云原生模式”等等。它的 value 就是对应的答案。



下面是整个函数的执行入口。首先输出一段请求参数的日志，然后获取意图名称，再获取参数，然后再判断这个意图是不是欢迎意图。如果是欢迎意图就播放一段话；如果是问题意图就获取对应的参数，再接下来判断参数存不存在。

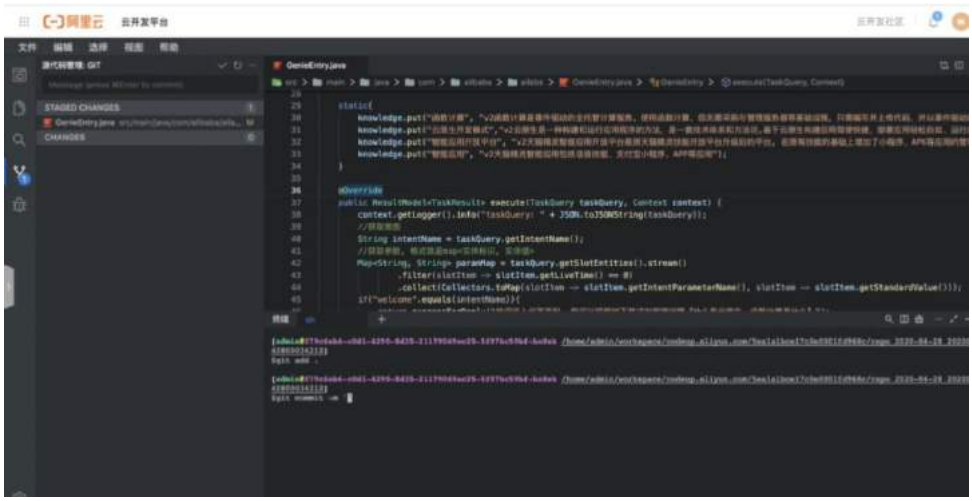
如果参数不存在就追问参数是什么；如果参数存在就判断当前问题是不是知识库里包含的问题。如果知识库里面存在该问题对应的答案，就直接返回；如果不存在，则直接回答“对不起您的问题超出了我的范围”。



如果既不是欢迎意图也不是问题意图，也可以直接返回成“对不起您的问题超出了我的知识范围”。

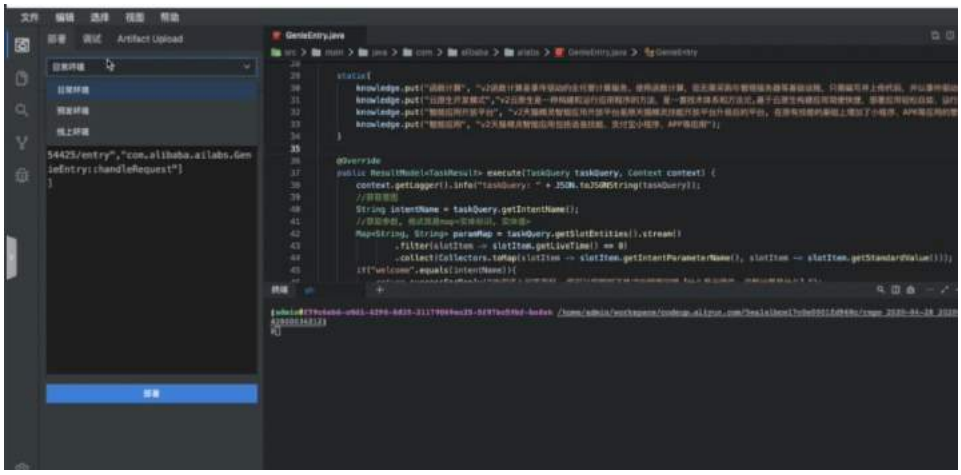
以上就是一个非常简单的问答百科的后台逻辑。当开发完这套逻辑之后，需要把代码上传到代码仓库。

上传代码仓库有两种方法，第一种是可以使用 CloudIDE 自带的插件去完成添加推送；第二种是可以通过 git 的命令进行操作。



- 应用部署

部署分三个环境：日常环境、预发环境、线上环境。



首先选择日常环境点击部署，点击后系统会开始构建打包。当部署完成且相关信息完全提交到天猫精灵平台后，接下来可以试着选择线上环境进行部署。完成后可以到智能应用开放平台进行在线测试，去验证逻辑是否正确。

- 在线测试和提审

问答百科页面有在线测试功能。

首先可以使用唤醒词对技能进行唤醒。当唤醒技能时，它会回答“欢迎进入问答百科，你可以按照如下格式向我提问”。然后再通过上面的设置，提出问题进行测试。

测试完成并无误，就可以对应用进行提审了。

前后端一体化应用开发实战

作者 | 繁易 淘系 Node 架构技术专家

本篇内容将分享 Midway Serverless 一体化应用开发实战。为什么需要一体化？大家如果有过开发经验会发现，在过往开发过程中前端归前端，后端归后端，这两部分是分别开发的，这样就会造成效率问题。所以阿里云推出了云端一体化应用开发实践，主要特点就是前后端一同协作开发。其实这个在 Midway 时代就有，但是没有当下 Midway Serverless 这么融合，可以做到前端后端一同开发一同发布。

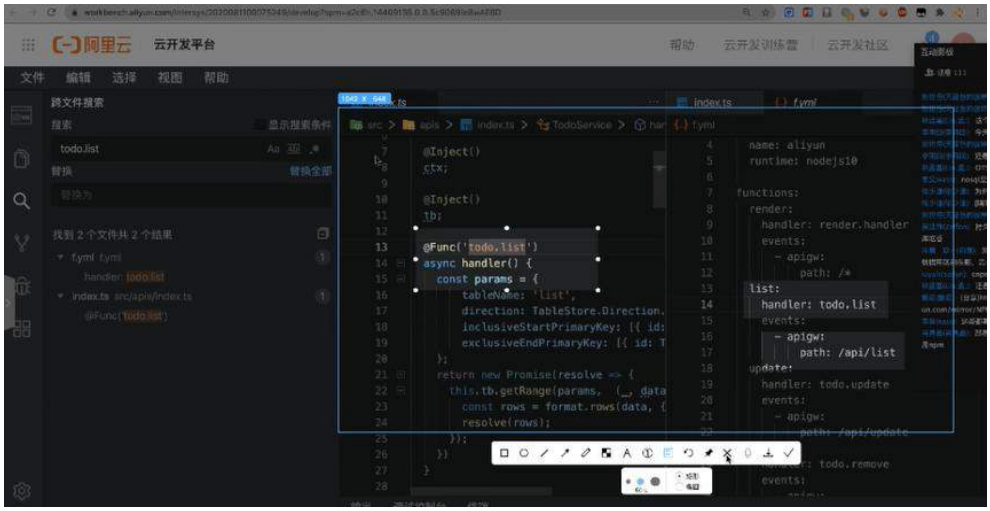
一、示例演示

首先是创建应用，创建方法前几篇课程文章都有介绍，在这里就不再赘述。创建应用的时候应用场景选择 WEB，解决方案选择 Midway Serverless OTS 数据库。创建完应用，安装依赖，最后启动。

启动完成后，准备工作就结束了。下面正式进入演示，如何开发一个一体化应用，如何去调用接口。

首先介绍下代码结构，在这个代码结构里有 apis 目录和一个传统前端的目录。我们以获取 todolist 接口为例，在传统前端目录里可以看到获取 todo 接口是一个叫 getlist 的方法，并用 fetch 调用 api/list 的接口。传统后端的接口是写在云端所以需要单独调用它；但是在一体化工程里，后端是可以直接写在 apis 里面的。举个例子，在前端写的 api/list 的代码，在 f.yml 里是可以找到对应的，它是在 handler 这里，handler 的作用就是查询所有的 todos 并返回。

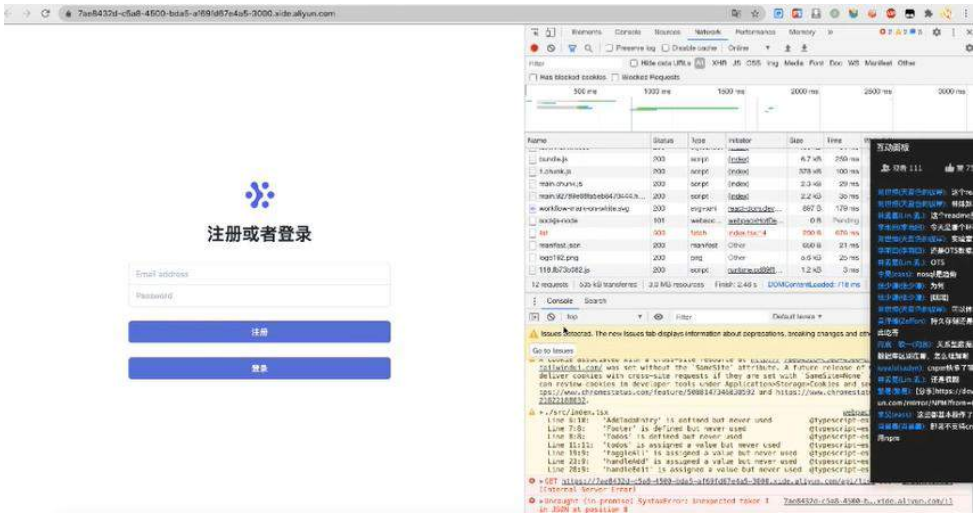
可以看到 handler 和 func 的 todo list 是相对应的，通过这样的方法，就可以把接口信息、函数信息和具体代码相关联，这样前端的调用就直接调用本地开发函数。



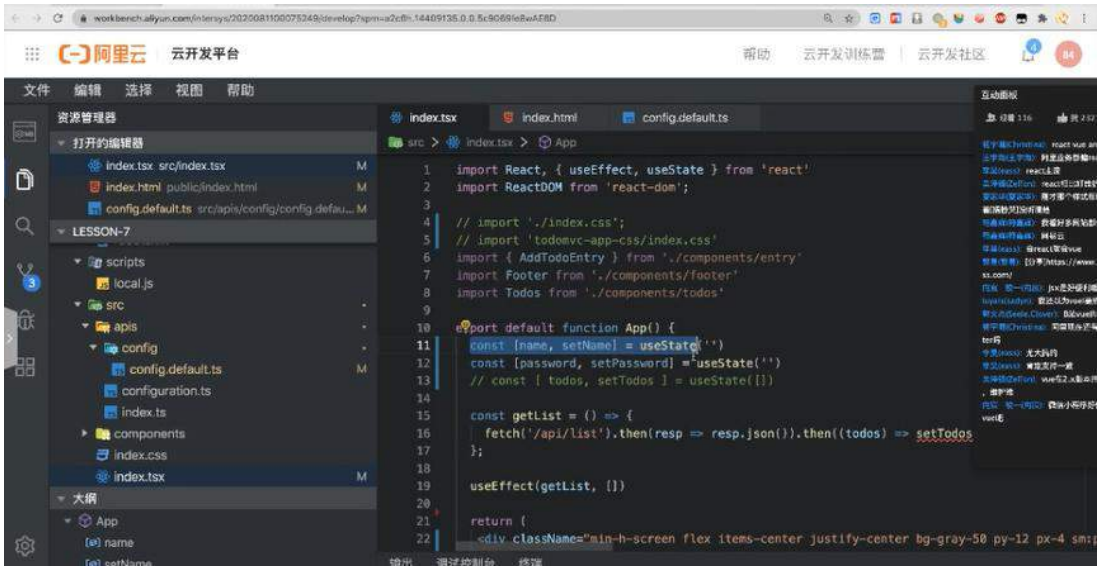
在前端社区中，主流的前端框架就三个：React， Vue 和 Angular；就分布来看，Angular 在国外比较主流，Vue 和 React 在国内比较受欢迎，这三个在开发模式上都有自己的一些特点。在阿里云云开发平台上，使用 React 更多一些。所以今天演示的 demo 也是和 React 相关的。国内很多网站都是使用 React 做开发的。

Vue 在国内也比较受欢迎，因为 Vue 可以支持中文，并且有中文文档和中文社区，且上手门槛低，更容易受新人喜欢。这些方面都比 react 要更受欢迎。

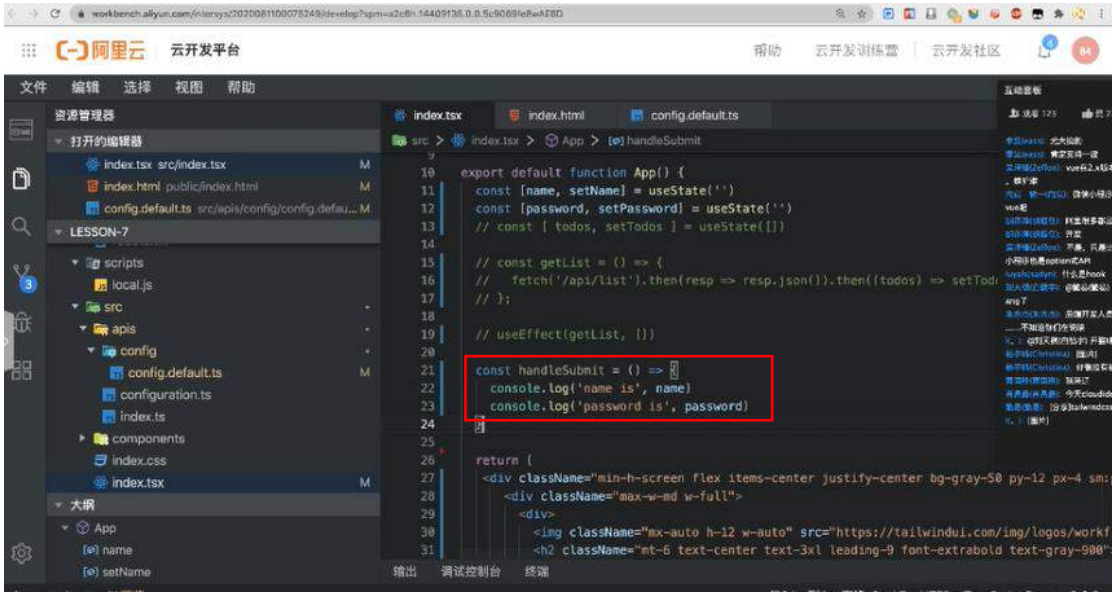
我们可以用比较流行的样式库 Tailwind css 创建一个样式，然后把样式的 css 加载，这样一个简单的网页就建成了。



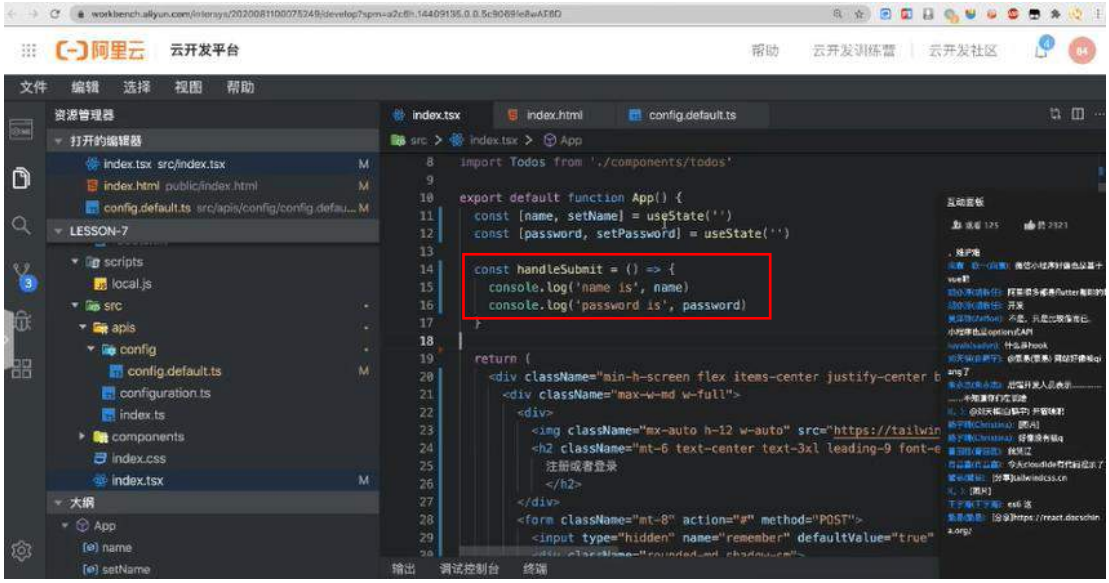
下面的开发就基于这个简单的网页演示。我们通过 input 里的 aria-label 的 Email address 和 Password 来调整 name 和 password，以对应今天演示的的账号和密码。



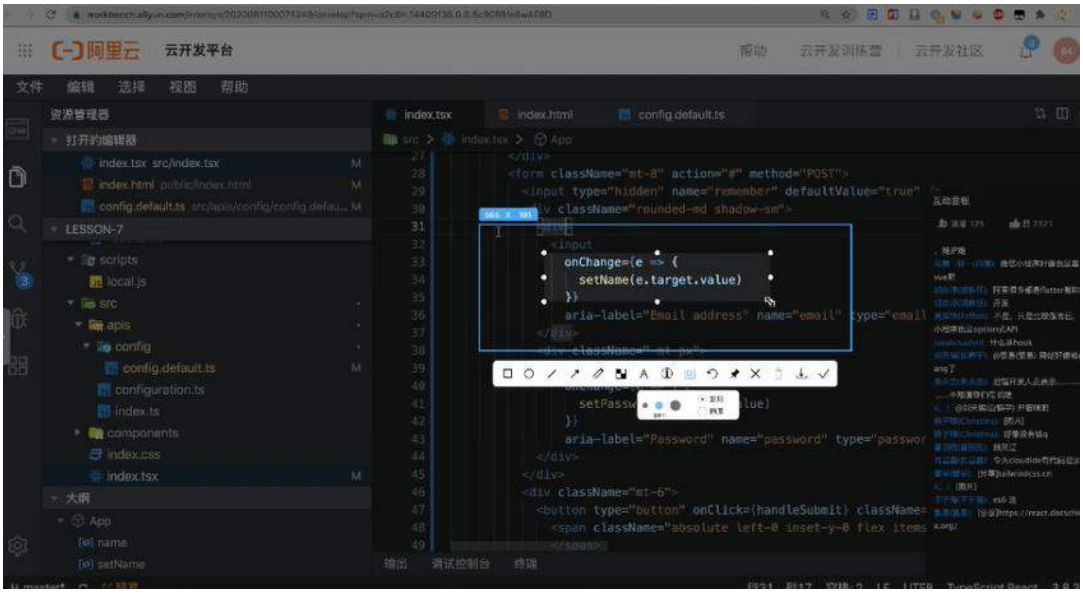
如下图这里加一个函数，用来管理提交事件。



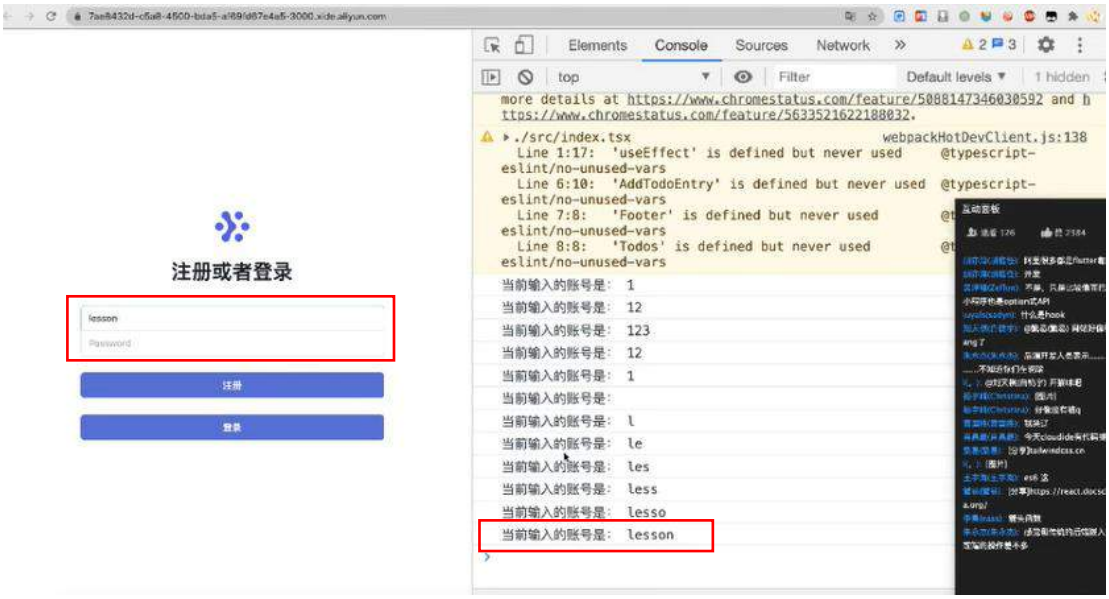
大家可以看到我这边写了一个叫我在这边写了一个叫做 handleSubmit 的事件，它的作用就是显示输入的账号和密码，然后会返回两个值，一个是 Name 代表已经输入的值，一个是 Set Name 代表设置当前正在输入的值。



如下图，当输入框的文字发生改变的时候，就会触发 onChange 事件，然后传递出一个 Event。

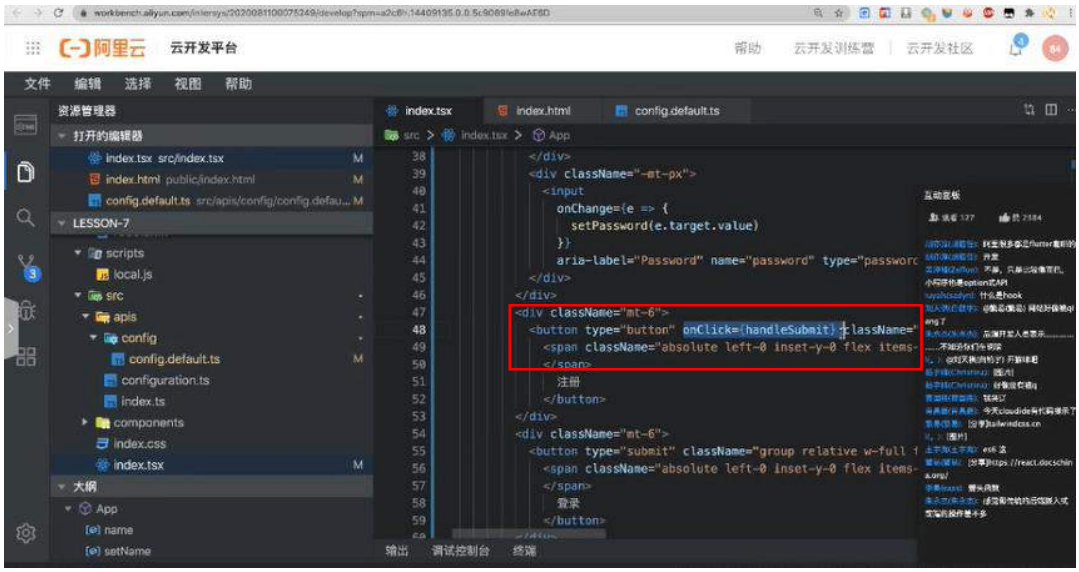


点 value 并输入一个 console.log 函数，然后在前端输入账号名，就能够通过监听前端事件的方式，看到输入值。



密码也同理。有了账号和密码之后，怎么把这个账号密码去提交给后端呢?

第一步就是获取账号和密码，在按钮上加一个 onClick 回调的事件。通过这个事件，当按钮被点击，就会触发 handleSubmit 的一个表达方式，在 handleSubmit 过程中，会显示正在注册的账户名和密码。

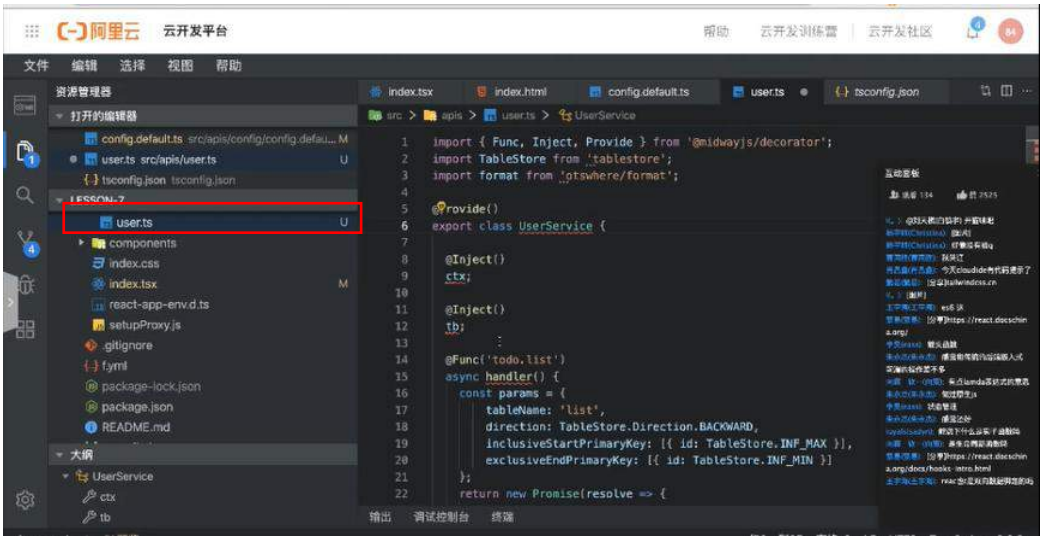


所以前端已经拿到了注册的账号和密码，那么怎么发给后端呢?

熟悉 react 的同学应该了解钩子函数，钩子函数其实是 React 最近推出的一种编程范式，叫 Hooks。在 React 官方网站有专门介绍 Hooks。

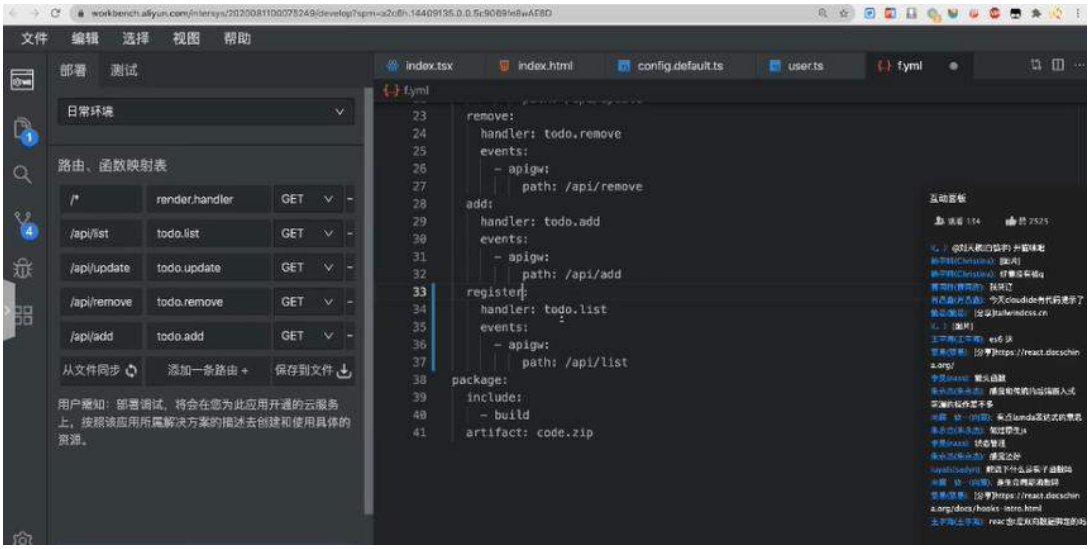
简单介绍下 React 和 Hooks 的工作。React 在淡化生命周期的作用，再用函数值组件做很多没有生命周期的概念。Hooks 是淡化生命周期的一个作用，而 React 是单向的数据流不是双向数据绑定，这点和 Vue 还是不太一样的。

回到 Cloud IDE 开发平台，在左侧目录新建一个 users.ts 的文件夹，去重新开发注册和登录的访问功能。先把他之前的代码直接给拷过来，然后命名为 User Service，然后把一些无关紧要的代码给删掉。

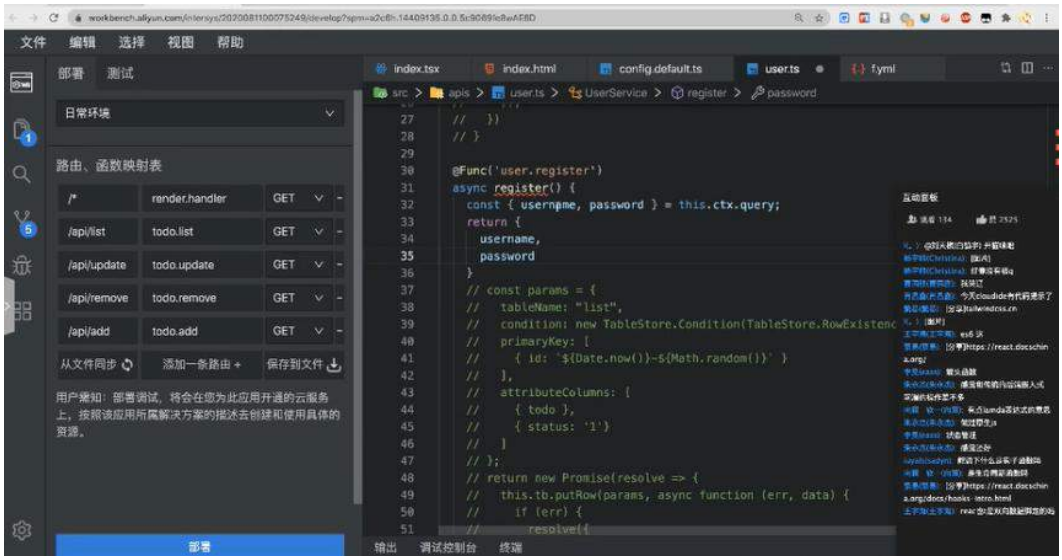


然后把他一些函数相关的东西给改成自己的。

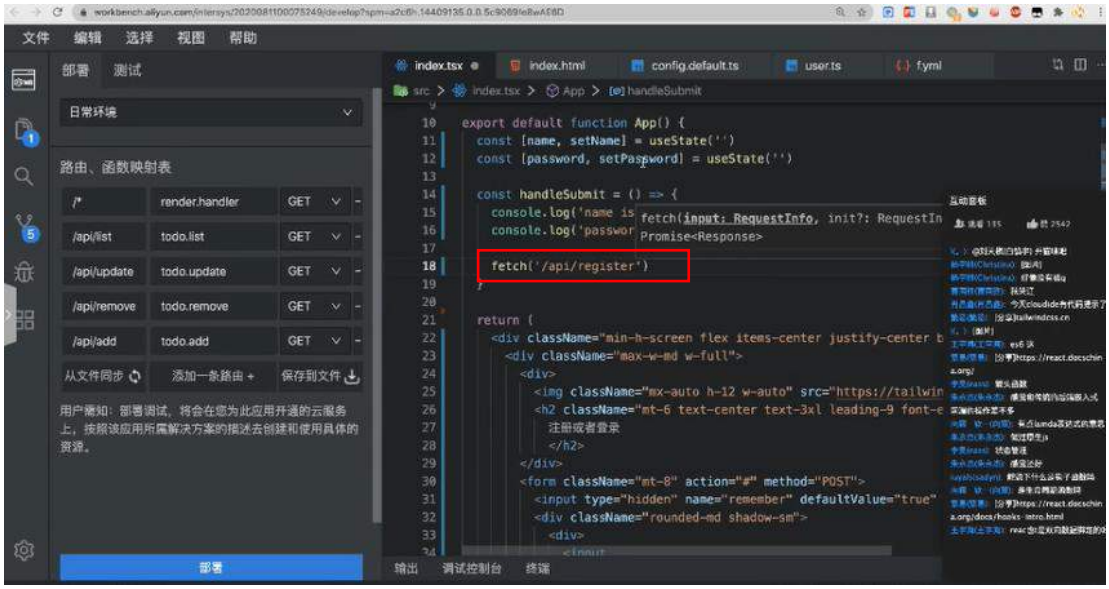
如前面所提示的，所有函数都是需要到 f.yml 注册的，所以调整完函数，就去注册，在这里把这个注册的函数取名为 register。



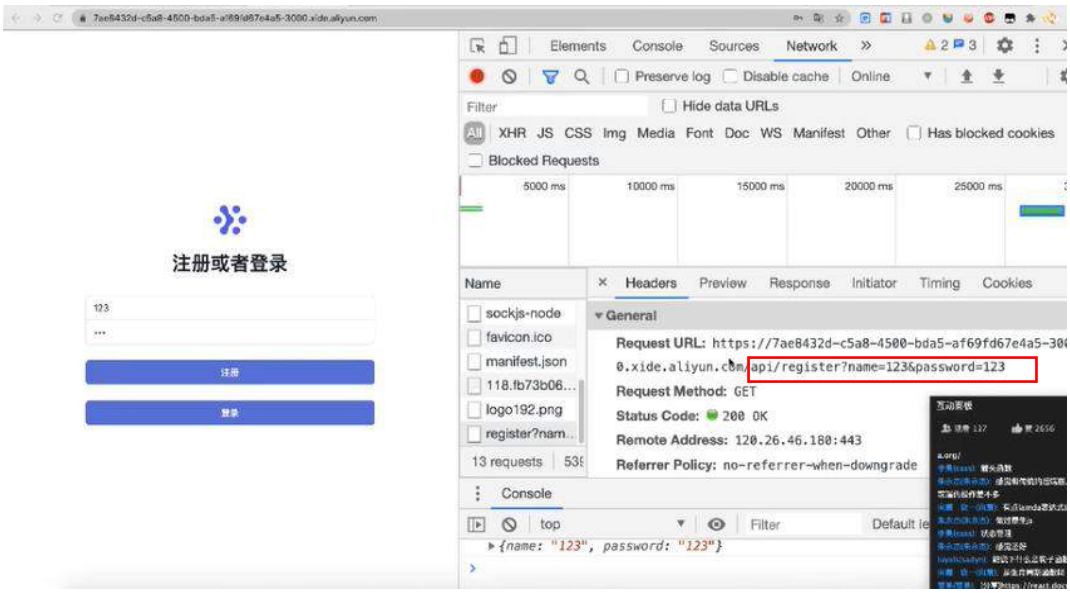
这个简单函数的作用是在从前端传递过来的查询参数里面取出 name 和 password，并且返回给前端。



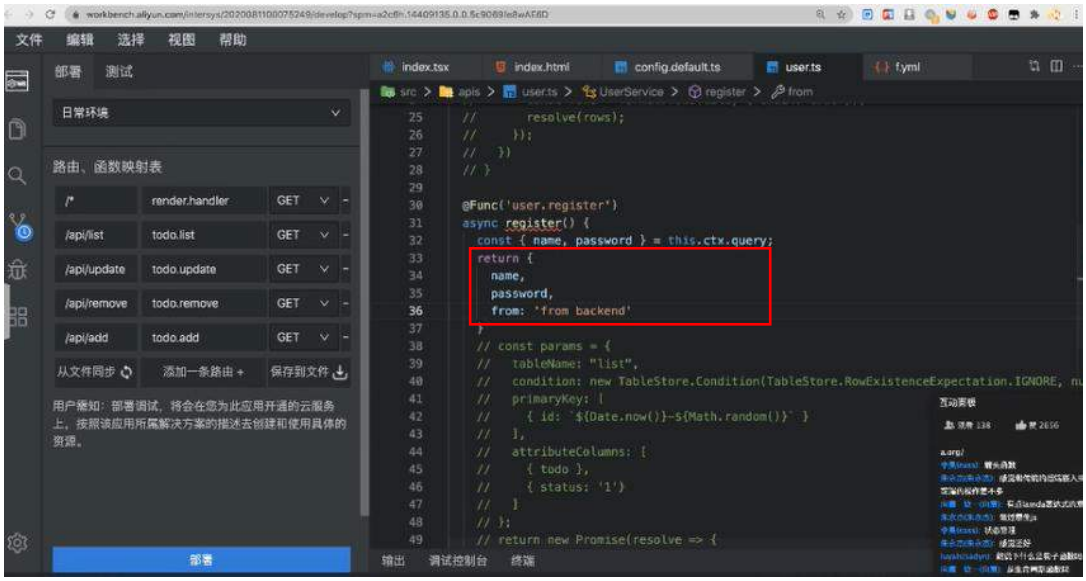
在这一步我们可以调用 Fetch/API/register。Fetch 是很常用的，是不管用哪个框架，发 API 请求几乎都会去用到的。



都准备好后，在前端发起一个简单的请求，并带上账号和密码。如下图所示，可以看到这个请求已经发送到后端了，在请求地址的最后挂了一串查询参数，拉到下面 Query string Parameters 里面有 name 和 password。

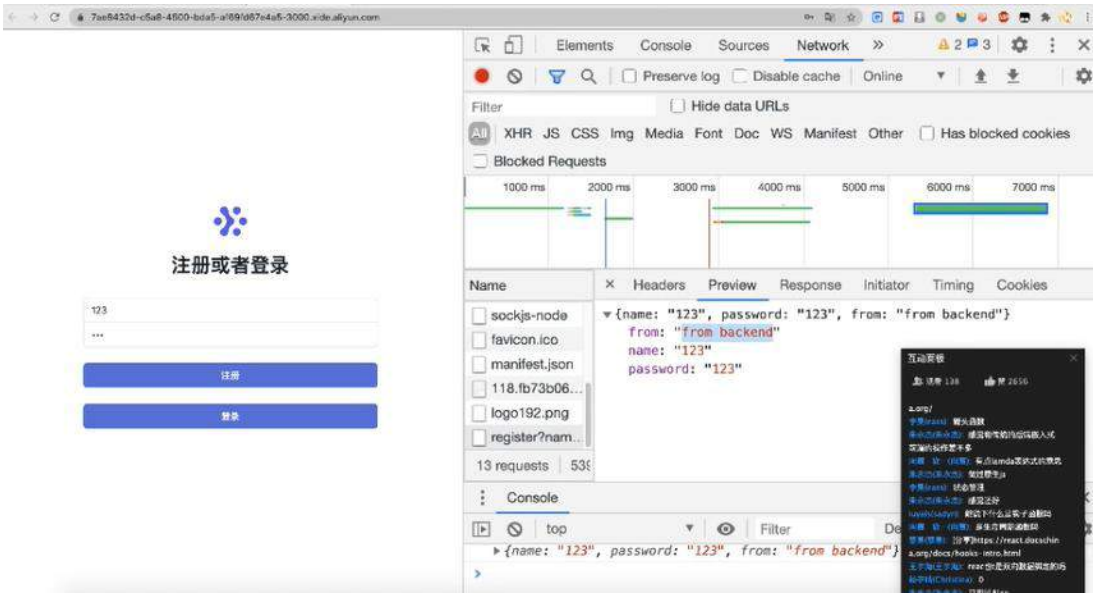


后端也会把 name 和 password 返回。



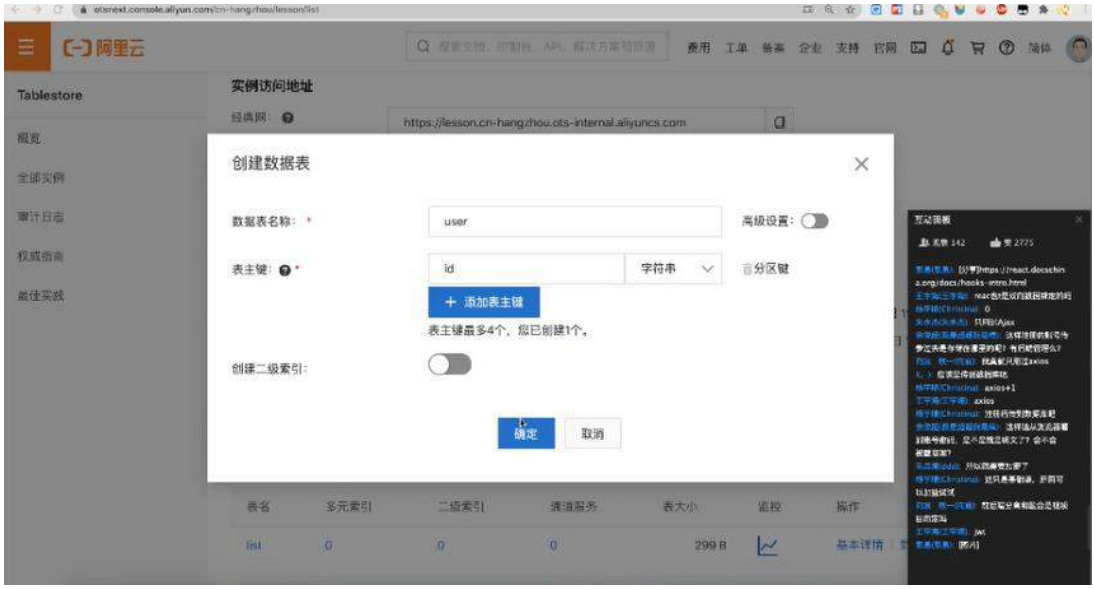
通过这种简单的方式，就可以实现把前端的信息传递给后端，后端接收到之后就可以存取数据库。我们前后端开发交互的核心点，就是前端怎么把数据给后端，后端怎么把数据落到数据库，然后后端怎么再把数据从数据库里面取出来再返回给前端。

如下图可以看到 from backend，证明这是从后端返回的。

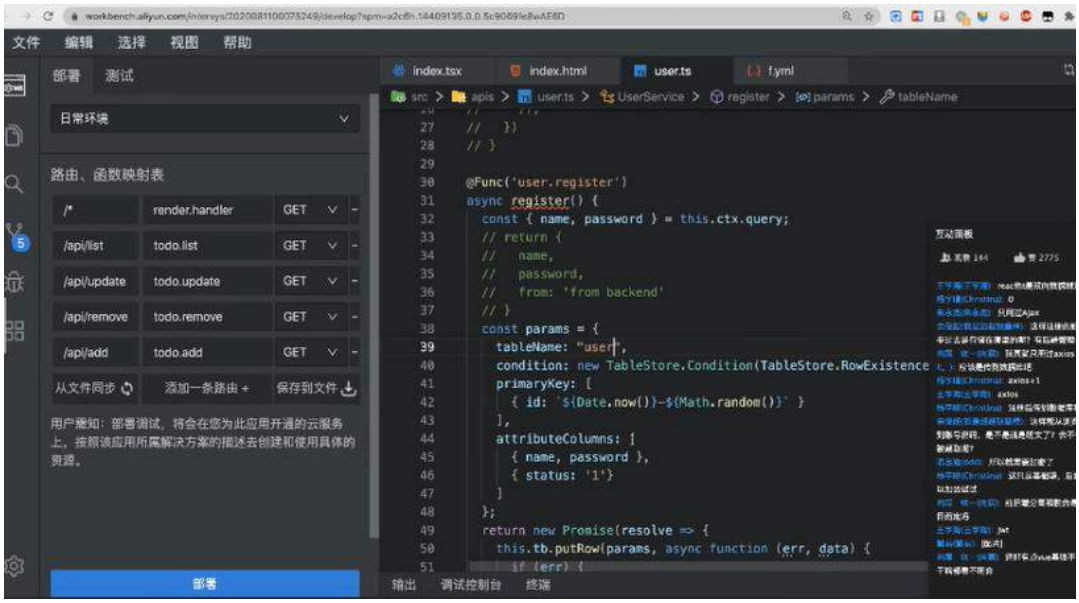


二、如何把数据落到数据库里

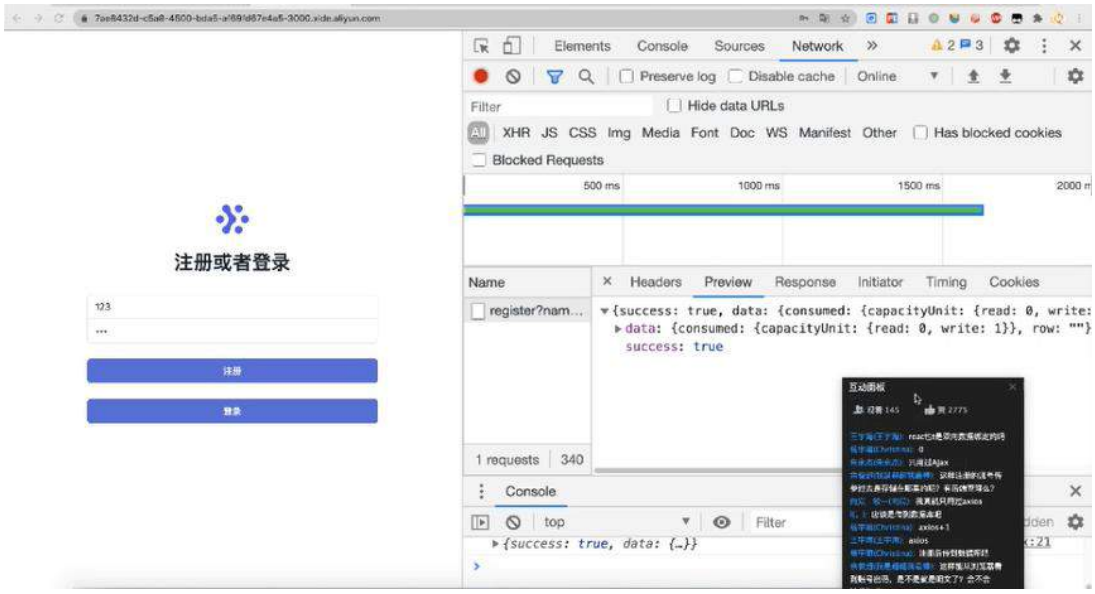
首先需要在阿里云表格存储里面新建一张表。



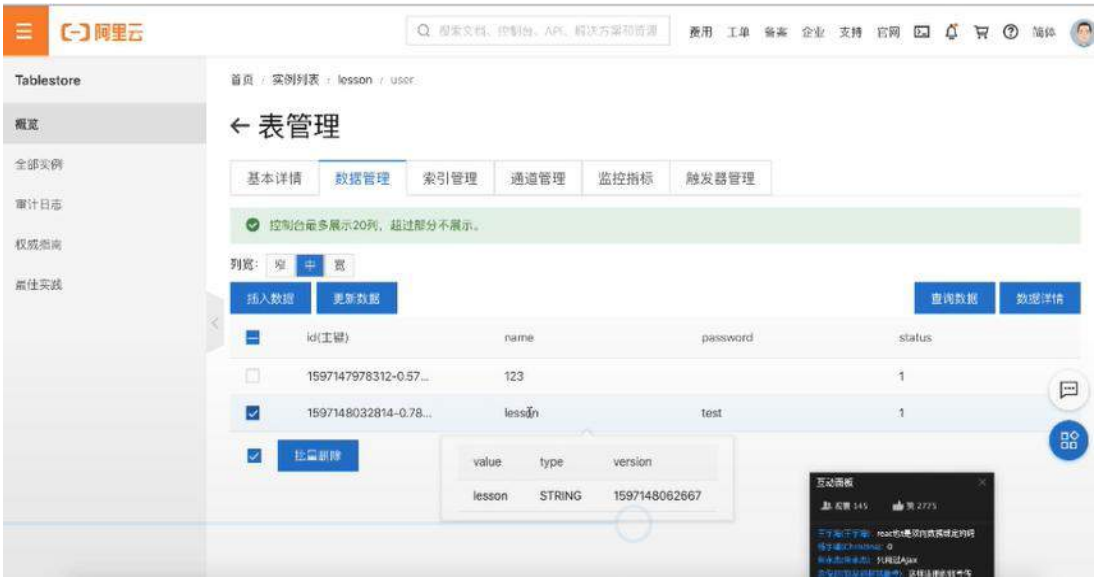
创建之后返回 Cloud IDE 开发平台，利用上次 todo 的代码，做一个将数据存储到数据库的操作。



然后返回前端测试一下接口。可以看到 success 的提示，表示数据已经成功过存储到数据库了。

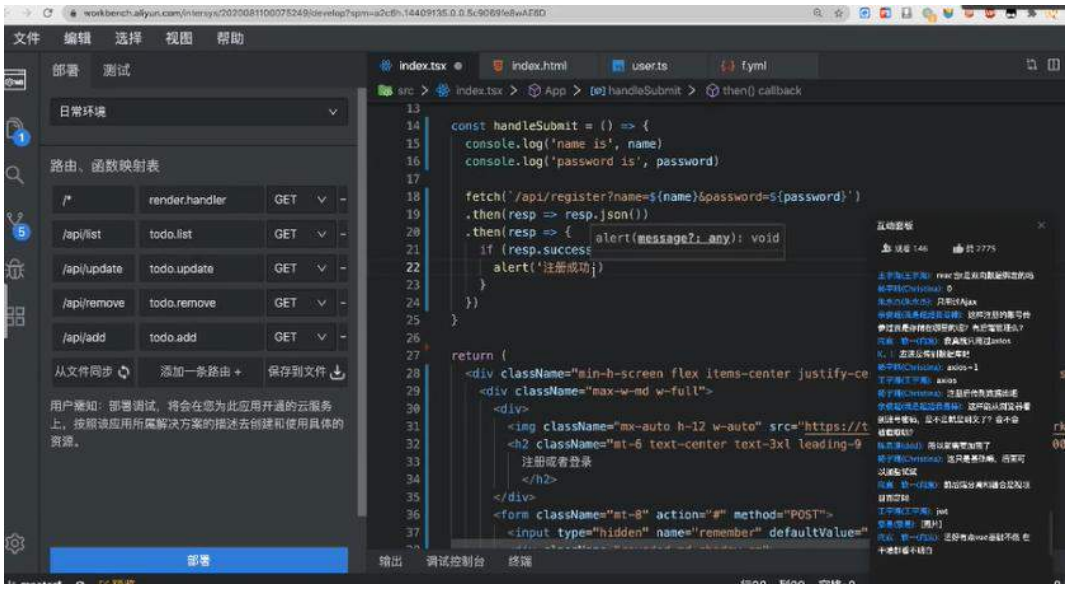


然后再点新建存储表格的数据管理，就可以看到刚刚注册的账号和密码。



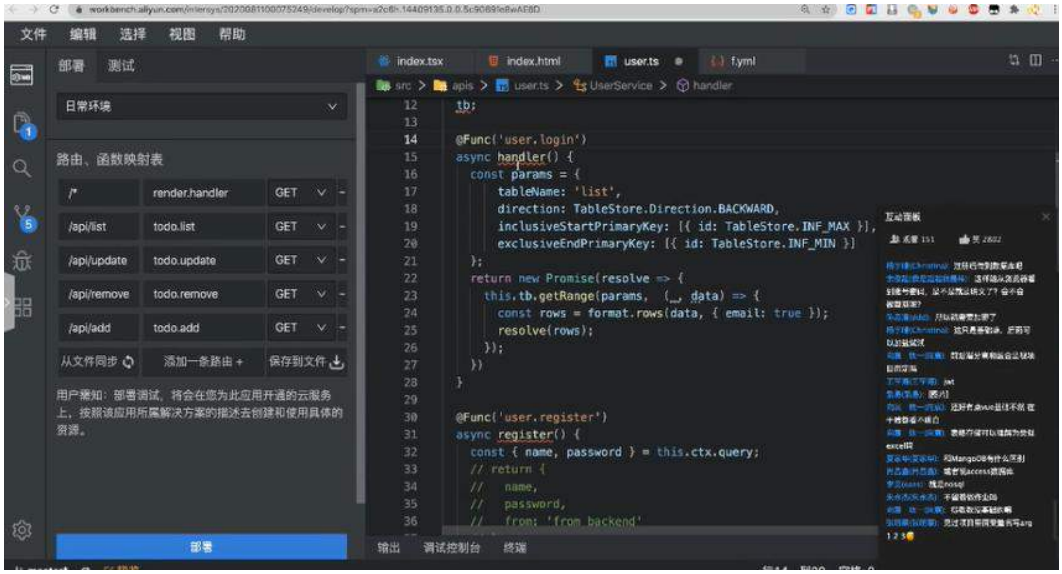
这就是一个非常简单的注册的过程。从表单里面输入数据，提交到后端，后端提交到数据库，数据库入库成功，这就完成了整个注册过程。

现在再尝试给前端补上最基础的交互叫“注册成功”，这样在注册成功后，就能弹出一个提示。

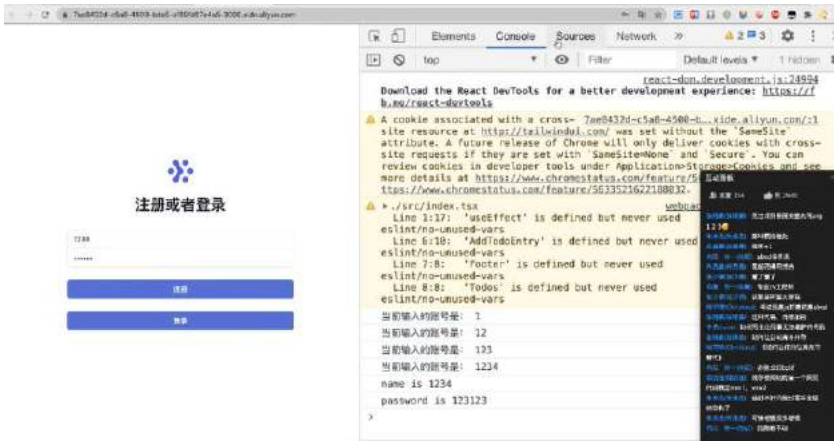


三、关于登录

登录部分有个叫 handle login 的方法，与上面叫 handle register 的方法对应。在 handle login 方法里，写 user.login…一个简单的函数，代码还是复制使用 todo 的。



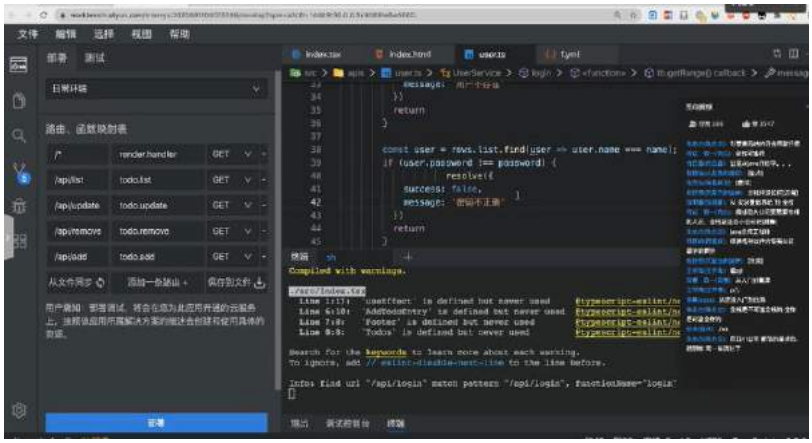
然后测一下刚才写的 log in 函数。还是先输出 name 和 password。



其实不难理解，注册在后端去写数据，然后往数据库里面存数据。那么登陆怎么理解呢？登录就是读和写。包括说数据库的读，还有数据库的写，然后通过读写加数据库的形式，查验密码，就完成登录了。

读到用户数据之后会做匹配，就是从数据库里面查到用户，然后去做一个验证和比对。查用户有两种方式，一种是通过数据库的参数来查这个用户是否在用户表内；如果用户在表内，再查密码是否正确；如果密码也正确，就会做登陆的措施。

还可以在登录这里做一些交互，比如“该用户不存在”或“密码不正确”。设置方法是，首先从列表里面取出 user，加上“密码不正确”的提示。这样登录的时候如果密码不正确就会有对应提示。



通过以上简单的交互我们已经实现了正常的登录功能，这也是今天课程的基础主题，也算是从 0 到 1，通过今天的展示，想必同学们已经了解了如何实现注册登录、从前端开发到后端入库、校验等一系列功能的实现。

钉钉扫一扫加入
Serverless 云开发沟通交流群

阿里云开发者“藏经阁”
海量免费电子书下载