函数工作流

快速入门

文档版本 01

发布日期 2024-05-20

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

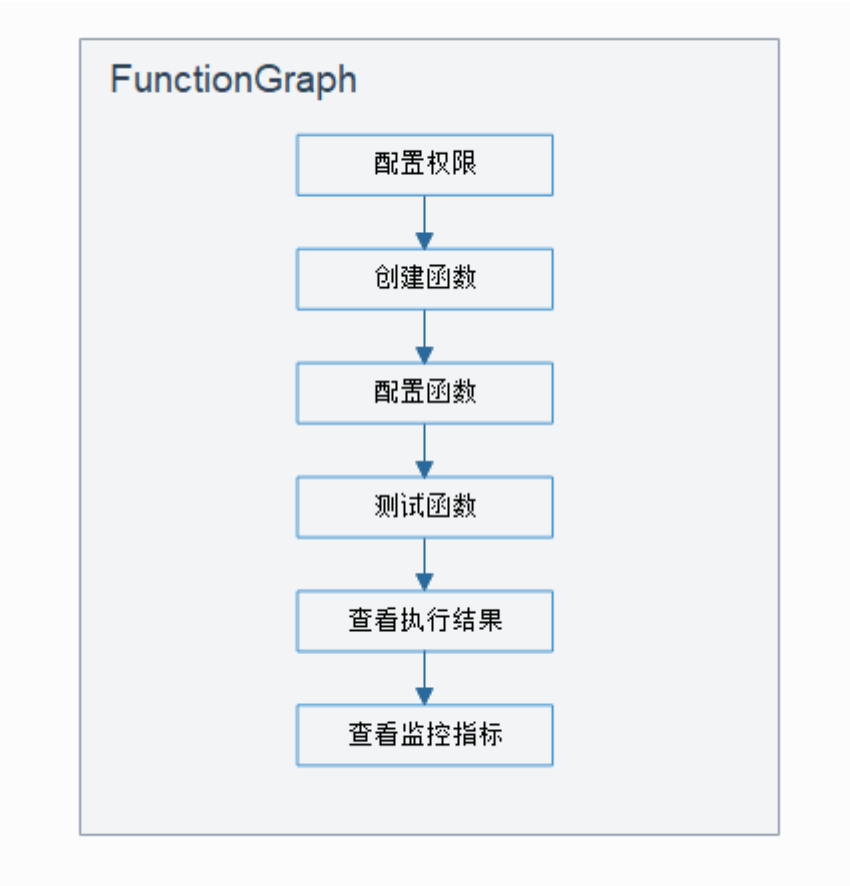
目 录

1 FunctionGraph 入门简介

使用流程

函数工作流FunctionGraph是一项基于事件驱动的函数托管计算服务。使用FunctionGraph函数，只需编写业务函数代码并设置运行的条件，无需配置和管理服务器等基础设施，函数以弹性、免运维、高可靠的方式运行。此外，按函数实际执行资源计费，不执行不产生费用。

使用FunctionGraph快速创建函数的流程如下：



1. 配置权限：确保登录的用户已有“FunctionGraph Administrator”权限。

2. 创建函数：选择使用空白模板创建函数、示例代码创建函数、容器镜像部署函数。
3. 配置函数：配置代码源或修改其他参数配置。
4. 测试函数：创建测试事件来调试函数。
5. 查看执行结果：在函数详情页面，根据配置的测试事件，查看执行结果。
6. 查看监控指标：在函数详情页面的“监控”页签，查看函数监控指标。

2 使用空白模板创建函数

概述

本章节介绍如何在函数工作流控制台使用空白模板快速开发一个简单的Hello World函数。以创建HelloWorld函数为例，介绍函数的创建及测试过程，供您快速体验FunctionGraph函数的基本功能。

步骤一：准备环境

本章节所有操作均默认具有操作权限，请确保您登录的用户已有“FunctionGraph Administrator”权限，即FunctionGraph服务所有权限，更多权限的说明请参考**权限管理**。

步骤二：创建函数

1. 登录**函数工作流控制台**，在左侧的导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面，开始创建空白函数。
3. 参考**图2-1**，函数名称输入“HelloWorld”，其他参数保持默认，完成后单击“创建函数”。

图 2-1 基本信息配置

4. 配置代码源，复制如下代码至代码窗，单击“部署”。

   样例代码实现的功能是：获取测试事件，打印测试事件信息。

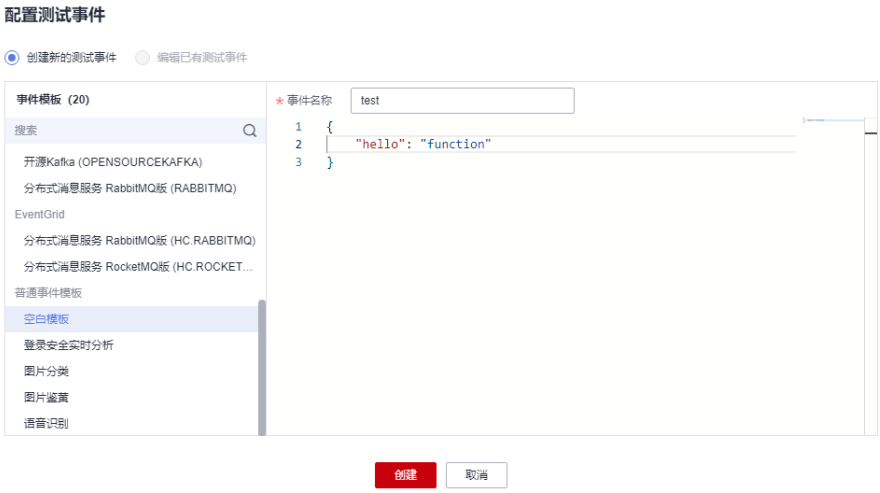
```
exports.handler = function (event, context, callback) {
    const error = null;
    const output = `Hello message: ${JSON.stringify(event)}`;
    callback(error, output);
}
```

步骤三：测试函数

1. 在函数详情页，单击“测试”，在弹窗中创建新的测试事件。
2. 选择“空白模板”，事件名称输入“test”，测试事件修改为如下所示，完成后单击“创建”。

```
{
    "hello": "function"
}
```

图 2-2 配置测试事件



步骤四：查看执行结果

单击test事件的“测试”，执行后，在右侧查看执行结果。

- “函数返回”显示函数的返回结果。
- “日志”部分显示函数执行过程中生成的日志。
- “执行摘要”部分显示“日志”中的关键信息。

图 2-3 查看执行结果

说明

此页面最多显示2K日志，了解函数更多日志信息，请参考**查询日志**。

步骤五：查看监控指标

在函数详情页面，选择“监控”页签。

- 在“监控”页签，先选择“指标”，再选择时间粒度（5分钟、15分钟、1小时），查看函数运行状态。
- 可以查看的指标有：调用次数、错误次数、运行时间（包括最大运行时间、最小运行时间、平均运行时间）、被拒绝次数、资源统计（预留实例个数）。

步骤六：删除函数

1. 在函数详情页面，单击右上角的“操作 > 删除函数”。
2. 在确认框中输入“DELETE”，然后单击“确定”，及时释放资源。

3 使用模板创建函数

概述

FunctionGraph平台提供了函数模板，本章节介绍如何在创建函数时选择模板，实现模板代码、运行环境自动填充，快速构建应用程序。

步骤一：准备环境

本章节所有操作均默认具有操作权限，请确保您登录的用户已有“FunctionGraph Administrator”权限，即FunctionGraph服务所有权限，更多权限的说明请参考**权限管理**。

步骤二：创建函数

1. 登录**函数工作流控制台**，在左侧的导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面，使用模板创建函数。
3. 参考**图3-1**，选择如下模板并单击“使用模板”。

图 3-1 选择模板

4. 函数名称输入“context”，“委托名称”选择已创建的任意委托，其他设置保持不变，单击“创建函数”。

说明

- 若不配置委托，在触发函数时，执行结果会返回
  Failed to access other services because no temporary AK, SK, or token has been obtained. Please set an agency.
- 目前函数模板新增定时开启/停止华为公有云数据库的RDS实例模板能力，可有效帮助您管理资源、降低维护成本。

图 3-2 定时开启/停止华为公有云数据库模板

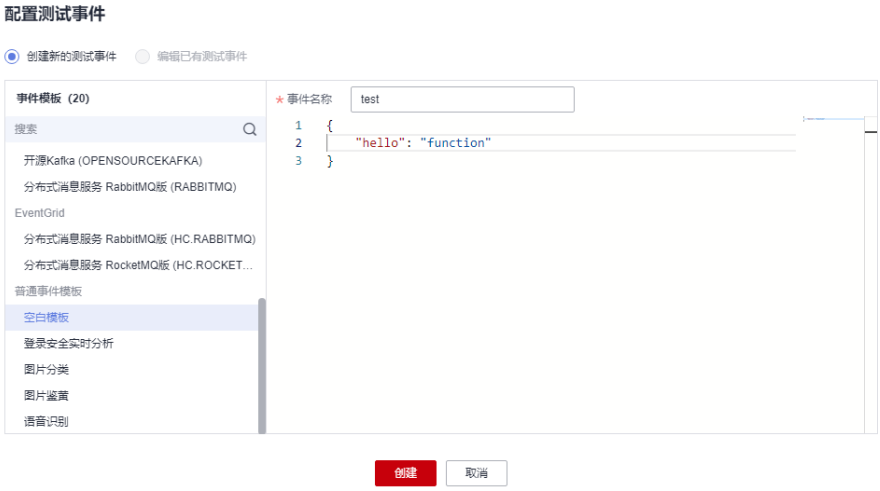
图 3-3 填写基本信息

步骤三：测试函数

1. 在函数详情页，单击“测试”，在弹窗中创建新的测试事件。

2. 选择“空白模板”，事件名称输入“test”，完成后单击“创建”。

图 3-4 配置测试事件



步骤四：查看执行结果

单击test事件的“测试”，成功执行后，在右侧查看执行结果。

- “函数返回”显示函数的返回结果。
- “日志”部分显示函数执行过程中生成的日志。
- “执行摘要”部分显示“日志”中的关键信息。

说明

此页面最多显示2K日志，了解函数更多日志信息，请参考**查询日志**。

步骤五：查看监控指标

在函数详情页面，选择“监控”页签。

- 在“监控”页签，先选择“指标”，再选择时间粒度（5分钟、15分钟、1小时），查看函数运行状态。
- 可以查看的指标有：调用次数、错误次数、运行时间（包括最大运行时间、最小运行时间、平均运行时间）、被拒绝次数、资源统计（预留实例个数）。

步骤六：删除函数

1. 在函数详情页面，单击右上角的“操作 > 删除函数”。
2. 在确认框中输入“DELETE”，然后单击“确定”，及时释放资源。

4 使用容器镜像部署函数

4.1 开发 HTTP 函数示例

概述

使用自定义镜像开发HTTP函数时，用户需要在镜像中实现一个http server，并监听**8000（下文示例中提及的8000端口请不要变动）端口**接收请求。备注：HTTP函数只支持APIG触发器。

步骤一：准备环境

本章节所有操作均默认具有操作权限，请确保您登录的用户已有“FunctionGraph Administrator”权限，即FunctionGraph服务所有权限，更多权限的说明请参考**权限管理**。

步骤二：制作镜像

以在linux x86 64位系统上制作镜像为例。（系统配置无要求）

1. 创建一个空文件夹
   ```
   mkdir custom_container_http_example && cd custom_container_http_example
   ```
2. 以Nodejs语言为例，实现一个Http Server，其他语言请参考**创建HTTP函数**。
   创建一个main.js文件，引入express框架，接收POST请求，打印请求Body到标准输出并返回Hello FunctionGraph, method POST给客户端。
   ```
   const express = require('express');

   const PORT = 8000;

   const app = express();
   app.use(express.json());

   app.post('/*', (req, res) => {
       console.log('receive', req.body);
       res.send('Hello FunctionGraph, method POST');
   });

   app.listen(PORT, () => {
     console.log(`Listening on http://localhost:${PORT}`);
   });
   ```


3. 创建一个package.json文件，此文件用于向npm提供信息，使其能够识别项目以及处理项目的依赖关系。

```
{
  "name": "custom-container-http-example",
  "version": "1.0.0",
  "description": "An example of a custom container http function",
  "main": "main.js",
  "scripts": {},
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
     "express": "^4.17.1"
  }
}
```

   - name：值为项目名。
   - version：值为项目版本。
   - main：列举文件为程序的入口文件。
   - dependencies：列出npm上可用的项目的所有依赖项。

4. 创建Dockerfile文件

```
FROM node:12.10.0

ENV HOME=/home/custom_container
ENV GROUP_ID=1003
ENV GROUP_NAME=custom_container
ENV USER_ID=1003
ENV USER_NAME=custom_container

RUN mkdir -m 550 ${HOME} && groupadd -g ${GROUP_ID} ${GROUP_NAME} && useradd -u $
{USER_ID} -g ${GROUP_ID} ${USER_NAME}

COPY --chown=${USER_ID}:${GROUP_ID} main.js ${HOME}
COPY --chown=${USER_ID}:${GROUP_ID} package.json ${HOME}

RUN cd ${HOME} && npm install

RUN chown -R ${USER_ID}:${GROUP_ID} ${HOME}

RUN find ${HOME} -type d | xargs chmod 500
RUN find ${HOME} -type f | xargs chmod 500

USER ${USER_NAME}
WORKDIR /

EXPOSE 8000
ENTRYPOINT ["node", "main.js"]
```

   - FROM：指定基础镜像为node:12.10.0，基础镜像必须设置，值可修改。
   - ENV：设置环境变量，设置HOME环境变量为/home/custom_container，设置GROUP_NAME和USER_NAME为custom_container，USER_ID和GROUP_ID为1003，这些环境变量必须设置，值可修改。
   - RUN：格式为RUN <命令>，例如RUN mkdir -m 550 ${HOME}表示构建容器时创建${USER_NAME}用户的home目录。
   - USER：切换${USER_NAME}用户。
   - WORKDIR：切换工作目录到${USER_NAME}用户的“/”目录下。
   - COPY：将main.js和package.json拷贝到容器的${USER_NAME}用户的home目录下。
   - **EXPOSE：暴露容器的8000端口，请勿修改。**
   - ENTRYPOINT：使用node main.js命令启动容器，请勿修改。

说明

1. 可以使用任意基础镜像。
2. 在云上环境会默认使用uid 1003，gid 1003 启动容器。uid、gid可以在函数页面的“设置 > 常规设置 >容器镜像覆盖”板块中修改，但不可以是root或其他保留id。
3. **HTTP函数示例中涉及的8000端口请勿修改。**

5. 构建镜像

   指定镜像的名称为custom_container_http_example，版本为latest，“.”指定Dockerfile所在目录，镜像构建命令将该路径下所有的内容打包给容器引擎帮助构建镜像。

   ```
   docker build -t custom_container_http_example:latest .
   ```

步骤三：本地验证

1. 启动docker容器

   ```
   docker run -u 1003:1003 -p 8000:8000 custom_container_http_example:latest
   ```

2. 打开一个新的命令行窗口，向开放的8000端口发送消息，支持访问模板代码中根目录“/”下所有路径，以下以“helloworld”为例。

   ```
   curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}' localhost:8000/helloworld
   ```

   按照模块代码中返回

   ```
   Hello FunctionGraph, method POST
   ```

3. 在容器启动端口可以看到

   ```
   receive {"message":"HelloWorld"}
   ```

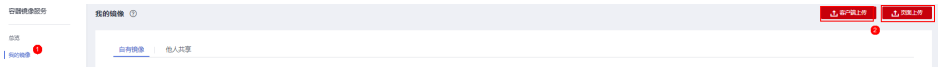
   ```
   [root@ecs-74d7 ~]# docker run -u 1003:1003 -p 8000:8000 custom_container_http_example:latest
   Listening on http://localhost:8000
   receive { message: 'HelloWorld' }
   ```

   或者使用docker logs命令获取容器的日志

   ```
   [root@ecs-74d7 custom_container_http_example]# docker logs 1354c3580638
   Listening on http://localhost:8000
   receive { message: 'HelloWorld' }
   [root@ecs-74d7 custom_container_http_example]#
   ```

步骤四：上传镜像

1. 登录容器镜像服务控制台，在左侧导航栏选择“我的镜像”。
2. 单击右上角的“客户端上传”或“页面上传”。
3. 根据指示上传镜像。



4. 上传成功后，在“我的镜像”界面可查看。

步骤五：创建函数

1. 在服务控制台左侧导航栏，选择“计算 > 函数工作流”。进入函数工作流控制台后在左侧导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面，使用容器镜像部署函数。
3. 填写基本信息。

- 函数类型：选择“HTTP函数”
- 函数名称：输入“custom_container_http”
- 容器镜像：输入上一步上传到SWR的镜像。（示例填写：swr.{局点id}.myhuaweicloud.com/{组织名称}/{镜像名称}:{版本名称}
- 现有委托：使用包含SWR Admin权限的委托，如果没有委托，请参考**创建委托**。

4. 完成后单击“创建函数”。

步骤六：测试函数

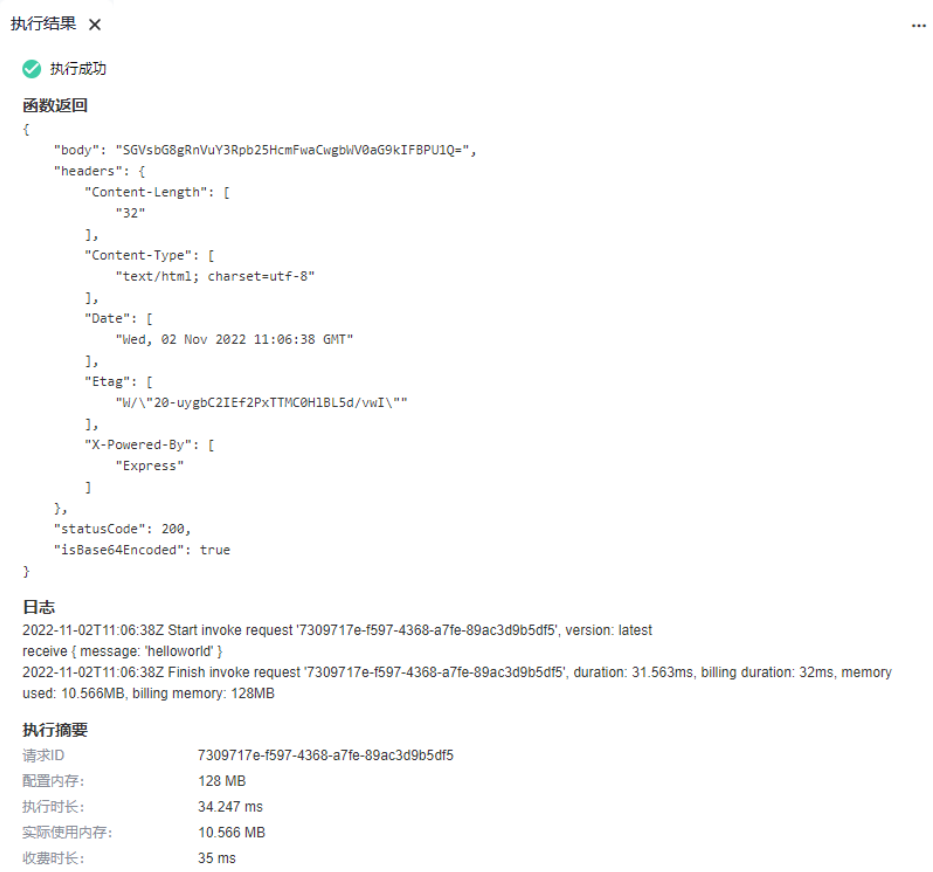
1. 在函数详情页，单击“测试”，在弹窗中创建新的测试事件。
2. 选择“apig-event-template”，事件名称输入“helloworld”，测试事件修改为如下所示，完成后单击“创建”。

```
{
    "body": "{\"message\": \"helloworld\"}",
    "requestContext": {
        "requestId": "11cdcdcf33949dc6d722640a13091c77",
        "stage": "RELEASE"
    },
    "queryStringParameters": {
        "responseType": "html"
    },
    "httpMethod": "POST",
    "pathParameters": {},
    "headers": {
        "Content-Type": "application/json"
    },
    "path": "/helloworld",
    "isBase64Encoded": false
}
```

步骤七：查看执行结果

单击helloworld事件的“测试”，执行后，在右侧查看执行结果，执行结果如下图。

图 4-1 执行结果



- “函数返回”显示函数的返回结果。
- “日志”部分显示函数执行过程中生成的日志。
- “执行摘要”部分显示“日志”中的关键信息。

说明

此页面最多显示2K日志，了解函数更多日志信息，请参考**查询日志**。

步骤八：查看监控指标

在函数详情页面，选择“监控”页签。

- 在“监控”页签，先选择“指标”，再选择时间粒度（5分钟、15分钟、1小时），查看函数运行状态。
- 可以查看的指标有：调用次数、错误次数、运行时间（包括最大运行时间、最小运行时间、平均运行时间）、被拒绝次数、资源统计（预留实例个数）。

步骤九：删除函数

1. 在函数详情页面，单击右上角的“操作 > 删除函数”。
2. 在确认框中输入“DELETE”，然后单击“确定”，及时释放资源。

4.2 开发事件函数示例

概述

使用自定义镜像开发事件函数时，用户需要在镜像中实现一个http server，并监听8000端口接收请求。其中，请求路径/init 默认为函数初始化入口，请根据需要实现该接口。请求路径/invoke为函数执行入口，触发器事件转到该接口处理，请求参数请参见**函数支持的事件源**。

步骤一：准备环境

本章节所有操作均默认具有操作权限，请确保您登录的用户已有“FunctionGraph Administrator”权限，即FunctionGraph服务所有权限，更多权限的说明请参考**权限管理**。

步骤二：制作镜像

以在linux x86 64位系统上制作镜像为例。（ 系统配置无要求 ）

1. 创建一个空文件夹
   ```
   mkdir custom_container_event_example && cd custom_container_event_example
   ```
2. 以Nodejs语言为例，实现一个Http Server，处理函数初始化init请求和函数调用invoke请求并响应。

   创建一个main.js文件，引入express框架，实现Method为POST和Path为/invoke的函数执行入口，实现Method为POST和Path为/init的函数初始化入口。
   ```
   const express = require('express');

   const PORT = 8000;

   const app = express();
   app.use(express.json());

   app.post('/init', (req, res) => {
     console.log('receive', req.body);
     res.send('Hello init\n');
   });

   app.post('/invoke', (req, res) => {
     console.log('receive', req.body);
     res.send('Hello invoke\n');
   });

   app.listen(PORT, () => {
     console.log(`Listening on http://localhost:${PORT}`);
   });
   ```
3. 创建一个package.json文件，此文件用于向npm提供信息，使其能够识别项目以及处理项目的依赖关系。
   ```
   {
     "name": "custom-container-event-example",
     "version": "1.0.0",
     "description": "An example of a custom container event function",
     "main": "main.js",
     "scripts": {},
     "keywords": [],
     "author": "",
     "license": "ISC",
   ```



```
  "dependencies": {
     "express": "^4.17.1"
  }
}
```

- name：值为项目名。
- version：值为项目版本。
- main：列举文件为程序的入口文件。
- dependencies：列出npm上可用的项目的所有依赖项。

4. 创建Dockerfile文件

```
FROM node:12.10.0

ENV HOME=/home/custom_container
ENV GROUP_ID=1003
ENV GROUP_NAME=custom_container
ENV USER_ID=1003
ENV USER_NAME=custom_container

RUN mkdir -m 550 ${HOME} && groupadd -g ${GROUP_ID} ${GROUP_NAME} && useradd -u $
{USER_ID} -g ${GROUP_ID} ${USER_NAME}

COPY --chown=${USER_ID}:${GROUP_ID} main.js ${HOME}
COPY --chown=${USER_ID}:${GROUP_ID} package.json ${HOME}

RUN cd ${HOME} && npm install

RUN chown -R ${USER_ID}:${GROUP_ID} ${HOME}

RUN find ${HOME} -type d | xargs chmod 500
RUN find ${HOME} -type f | xargs chmod 500

USER ${USER_NAME}
WORKDIR /

EXPOSE 8000
ENTRYPOINT ["node", "main.js"]
```

- FROM：指定基础镜像为node:12.10.0，基础镜像必须设置，值可修改。
- ENV：设置环境变量，设置HOME环境变量为/home/custom_container，设置GROUP_NAME和USER_NAME为custom_container，USER_ID和GROUP_ID为1003，这些环境变量必须设置，值可修改。
- RUN：格式为RUN <命令>，例如RUN mkdir -m 550 ${HOME}表示构建容器时创建${USER_NAME}用户的home目录。
- USER：切换${USER_NAME}用户。
- WORKDIR：切换工作目录到${USER_NAME}用户的“/”目录下。
- COPY：将main.js和package.json拷贝到容器的${USER_NAME}用户的home目录下。
- EXPOSE：暴露容器的8000端口，请勿修改。
- ENTRYPOINT：使用node /home/tester/main.js命令启动容器。

说明

1. 可以使用任意基础镜像。
2. 在云上环境会默认使用uid 1003，gid 1003 启动容器。uid、gid可以在函数页面的“设置 > 常规设置 >容器镜像覆盖”板块中修改，但不可以是root或其他保留id。

5. 构建镜像

   指定镜像的名称为custom_container_event_example，版本为latest，“.”指定Dockerfile所在目录，镜像构建命令将该路径下所有的内容打包给容器引擎帮助构建镜像。


```
docker build -t custom_container_event_example:latest .
```

步骤三：本地验证

1. 启动docker容器
   ```
   docker run -u 1003:1003 -p 8000:8000 custom_container_event_example:latest
   ```
2. 打开一个新的命令行窗口，向开放的8000端口发送消息，访问模板代码中指定的/init路径
   ```
   curl -XPOST -H 'Content-Type: application/json' localhost:8000/init
   ```
   按照模块代码中返回
   ```
   Hello init
   ```
3. 打开一个新的命令行窗口，向开放的8000端口发送消息，访问模板代码中指定的/invoke路径
   ```
   curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}' localhost:8000/invoke
   ```
   按照模块代码中返回
   ```
   Hello invoke
   ```
4. 在容器启动端口可以看到
   ```
   Listening on http://localhost:8000
   receive {}
   receive { message: 'HelloWorld' }
   ```

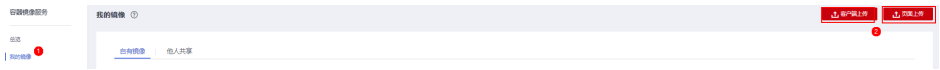
```
[root@ecs-74d7 ~]# docker run -u 1003:1003 -p 8000:8000 custom_container_event_example:latest
Listening on http://localhost:8000
receive {}
receive { message: 'HelloWorld' }
```

或者使用docker logs命令获取容器的日志

```
[root@ecs-74d7 custom_container_event_example]# docker logs 5560e1ec09d3
Listening on http://localhost:8000
receive {}
receive { message: 'HelloWorld' }
[root@ecs-74d7 custom_container_event_example]#
```

步骤四：上传镜像

1. 登录容器镜像服务控制台，在左侧导航栏选择“我的镜像”。
2. 单击右上角的“客户端上传”或“页面上传”。
3. 根据指示上传镜像。



4. 上传成功后，在“我的镜像”界面可查看。

步骤五：创建函数

1. 在服务控制台左侧导航栏，选择“计算 > 函数工作流”。进入函数工作流控制台后在左侧导航栏选择“函数 > 函数列表”。
2. 单击右上方的“创建函数”，进入“创建函数”页面，使用容器镜像部署函数。
3. 填写基本信息。
   - 函数类型：选择“事件函数”
   - 函数名称：输入“custom_container_event”

- 容器镜像：输入上一步上传到SWR的镜像。（示例填写：swr.{局点id}.myhuaweicloud.com/{组织名称}/{镜像名称}:{版本名称}
- 现有委托：使用包含SWR Admin权限的委托，如果没有委托，请参考**创建委托**。

4. 完成后单击“创建函数”。
5. 在函数详情页“设置 > 高级设置”，开启“配置初始化函数”，即调用init接口进行初始化。

步骤六：测试函数

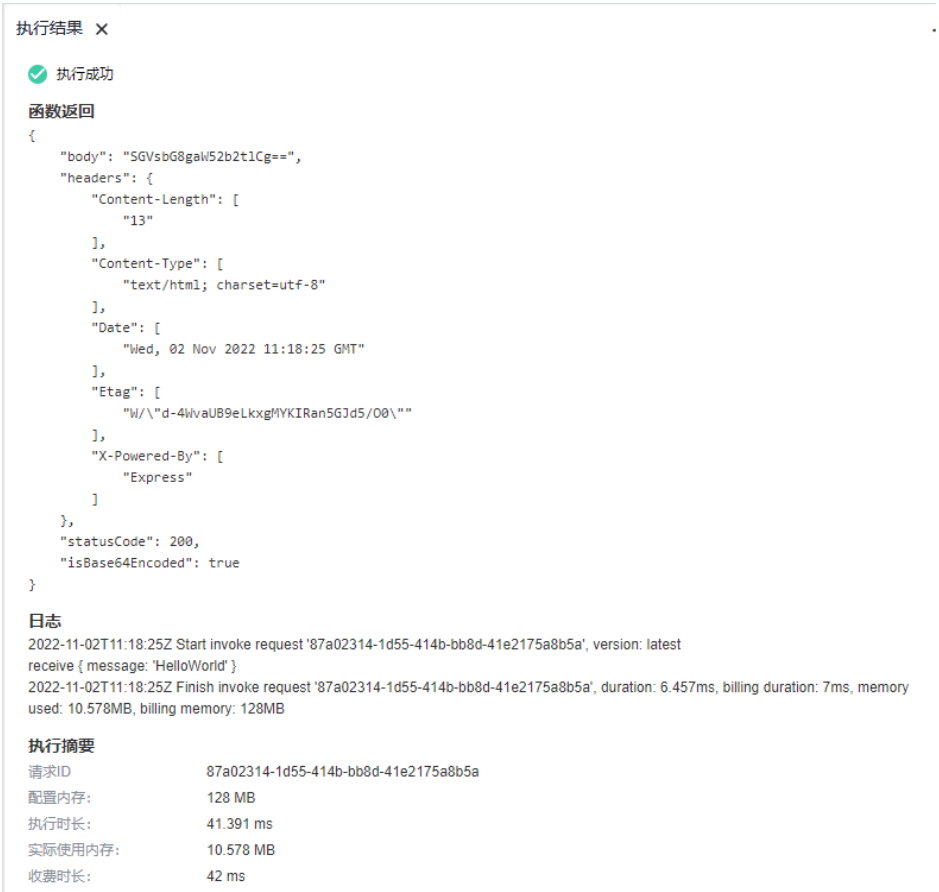
1. 在函数详情页，单击“测试”，在弹窗中创建新的测试事件。
2. 选择“空白模板”，事件名称输入“helloworld”，测试事件修改为如下所示，完成后单击“创建”。

```
{
  "message": "HelloWorld"
}
```

步骤七：查看执行结果

单击helloworld事件的“测试”，执行后，在右侧查看执行结果，执行结果如下图。

图 4-2 执行结果



- “函数返回”显示函数的返回结果。

- “日志”部分显示函数执行过程中生成的日志。
- “执行摘要”部分显示“日志”中的关键信息。

说明

此页面最多显示2K日志，了解函数更多日志信息，请参考**查询日志**。

步骤八：查看监控指标

在函数详情页面，选择“监控”页签。

- 在“监控”页签，先选择“指标”，再选择时间粒度（5分钟、15分钟、1小时），查看函数运行状态。
- 可以查看的指标有：调用次数、错误次数、运行时间（包括最大运行时间、最小运行时间、平均运行时间）、被拒绝次数、资源统计（预留实例个数）。

步骤九：删除函数

1. 在函数详情页面，单击右上角的“操作 > 删除函数”。
2. 在确认框中输入“DELETE”，然后单击“确定”，及时释放资源。

5 入门实践

当您了解如何创建函数等基本操作后，可以根据自身的业务需求使用函数工作流 FunctionGraph 提供的一系列常用实践。

本文介绍函数工作流 FunctionGraph 常用实践，帮助您更好的使用函数工作流。

表 5-1 常用最佳实践

实践	描述
使用函数压缩图片	本实践基于函数工作流服务实践所编写，用于指导您使用函数工作流服务实现图片压缩的功能
使用函数为图片打水印	本实践基于函数工作流服务实践所编写，用于指导您使用函数工作流服务实现为图片打水印的功能。
函数+LTS：日志实时分析实战	• 通过LTS云日志服务，快速完成ECS等服务器的任务运行日志采集、加工和转换。 • 通过函数工作流服务中的函数创建LTS触发器获取日志数据，经由自定义函数对日志中的关键信息进行分析和处理，过滤出告警日志。 • SMN消息通知服务通过短信和邮件推送告警信息，通知业务人员进行处理。 • 将函数处理后的日志数据投递至OBS桶中集中存储，便于后续处理。

实践	描述
函数+CTS：登录/登出安全分析实战	• 通过CTS云审计服务，完成对公有云账户对各个云服务资源操作动作和结果的实时记录。 • 通过在函数工作流服务中创建CTS触发器获取订阅的资源操作信息，经由自定义函数对资源操作的信息进行分析和处理，产生告警日志。 • SMN消息通知服务通过短信和邮件推送告警信息，通知业务人员进行处理。
定时开关华为公有云虚拟机	当您需要在特定时间打开或者关闭华为公有云虚拟机时，可以考虑通过函数服务调用华为云ECS接口，定时开关虚拟机。 • 开机节点：需要定时打开的虚拟机。 • 关机节点：需要定时关闭的虚拟机。
使用函数处理IOT数据	• 该案例演示您如何使用FunctionGraph 与IotDA 服务组合，处理物联网设备上报以及设备状态变动的相关数据。物联网设备在IotDA平台进行管理，设备产生的数据可以从IotDA直接流转触发FunctionGraph的函数运行。用户可以根据需要编写函数处理这些数据。 • 通常该组合，可以适用于以下场景，如将设备上报的数据在处理后进行存储到如OBS；对上报的数据进行结构化，清洗然后存储到数据库；根据设备状态变化进行事件通知等。
工作流+函数：自动化处理OBS中数据	本实践基于函数流服务实践所编写，用于指导您使用函数流服务实现OBS数据处理的功能。