拉勾教育
— 互联网人实战大学 —

《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

第30讲：前端热点技术之 Serverless

引言

Serverless 并不属于前端技术

- 它是一个非常高效的工具
- 对于只专注于前端领域的工程师而言
- 了解 Serverless 背后的思想
- 对提升开发思维会有一定的帮助

什么是 Serverless

Serverless

由“server”和“less”两个单词组合而成

中文的意思就是“无服务器”

不是语言或框架，而是一种软件的部署方式

Serverless 从何而来

Serverless

一种构建和管理基于微服务架构的完整流程

完全由云厂商管理

Serverless 的组成

Faas
（Function-as-a-Service）

函数即服务

一个函数就是一个服务

BaaS
（Backend-as-a-Service）

后端即服务

集成了中间件技术

Serverless 的特点

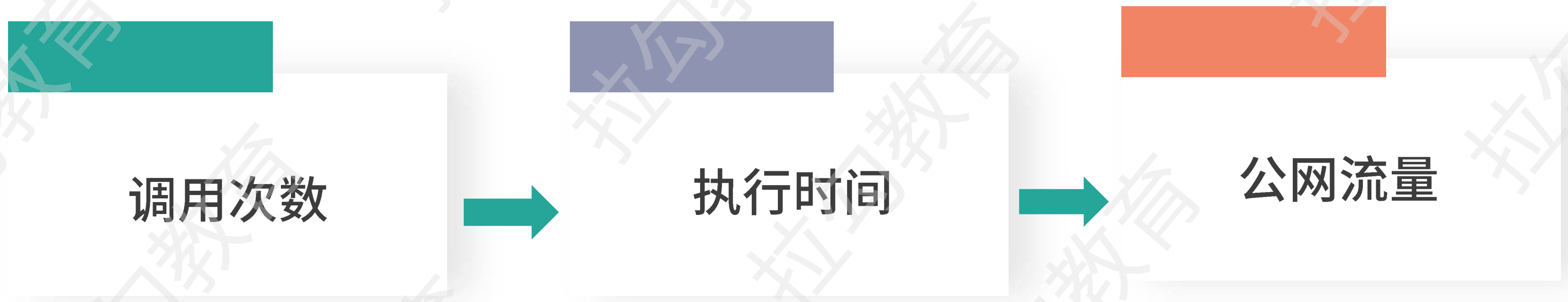
免维护

Serverless 提供了运行代码的环境

能自动实现负载均衡、弹性伸缩这类高级功能

Serverless 的特点

费 用



Serverless 的特点

拉勾教育
— 互联网人实战大学 —

#	计费项	价 格	免费额度
1	调用次数（次）	0.00000133（元）	1,000，000（次）
2	执行时间（CU-秒）	0.00011108（元）	400，000（CU-秒）
3	公网流量（GB）	0.8（元）	0

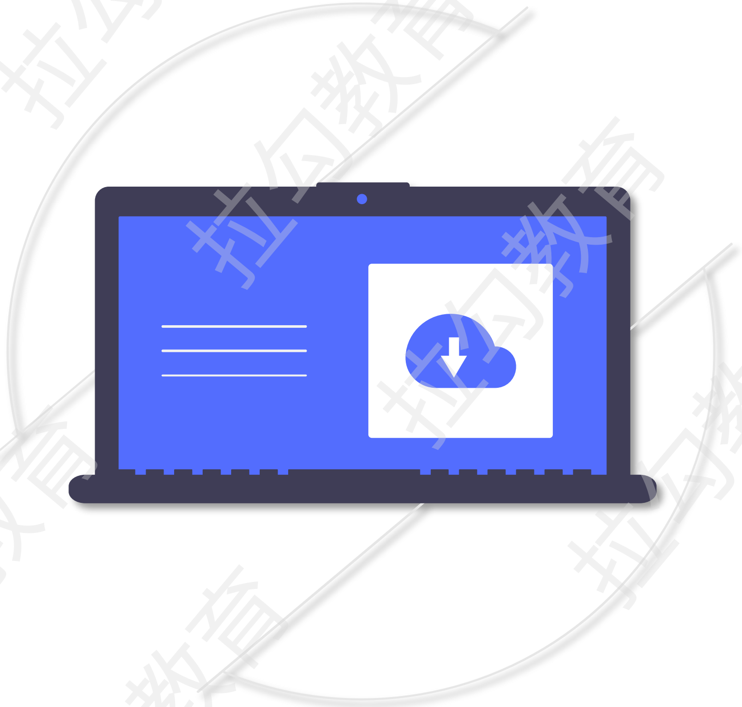
注：如果您的应用有稳定的执行时间，购买预付费（包年包月）计算力可以有效节约成本

Serverless 的特点

深度绑定

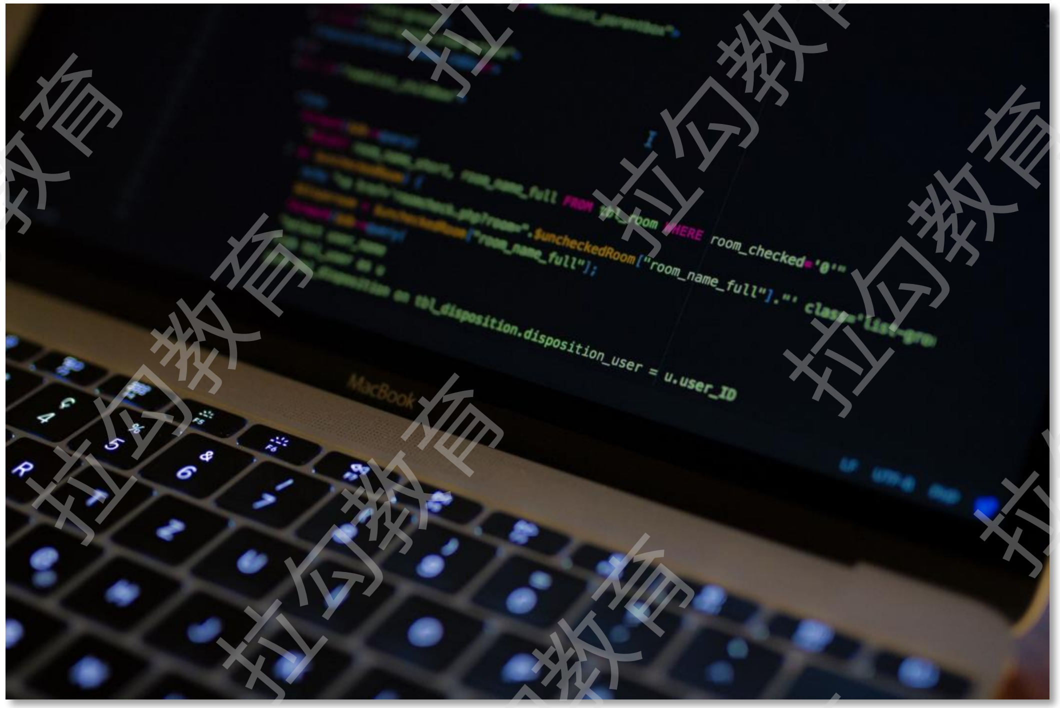
通常使用某个云厂商的 Serverless 产品时

函数计算、对象存储、数据库等



Serverless 的特点

拉勾教育
— 互 联 网 人 实 战 大 学 —



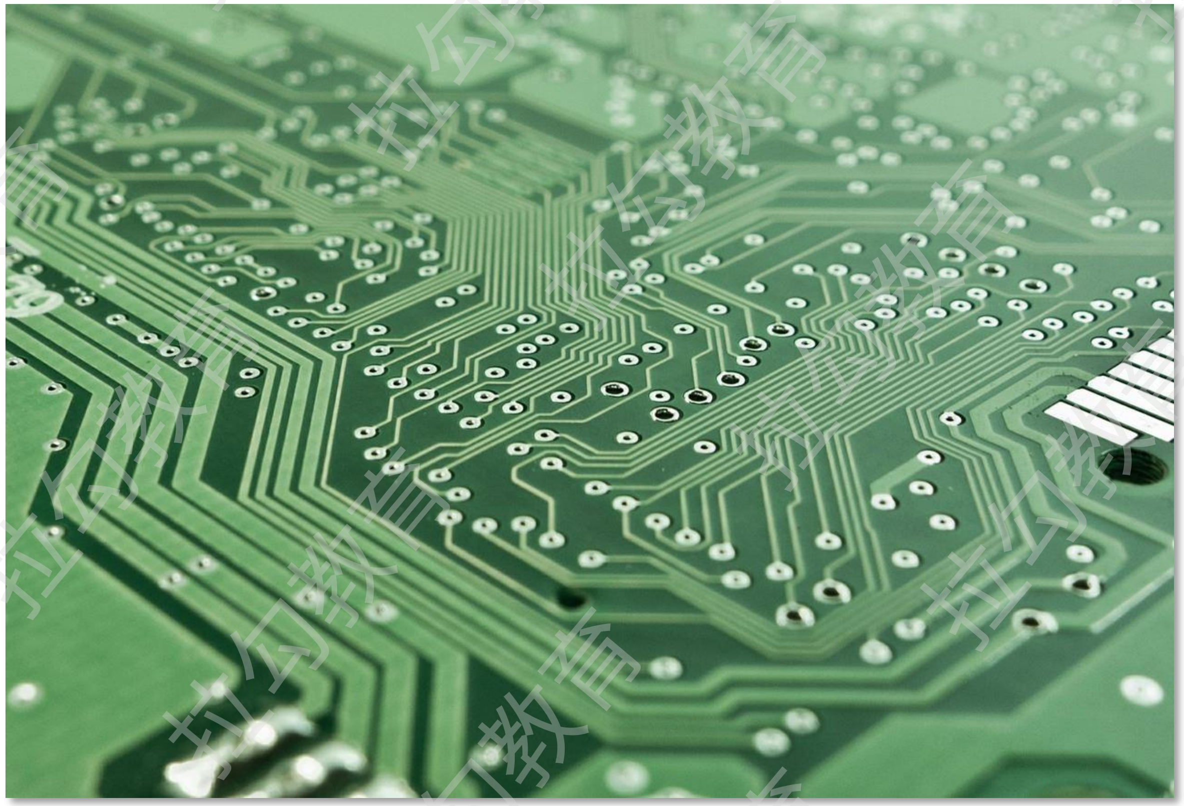
运行时长限制

函数执行时间是有限制的

Serverless 的特点

冷启动

首次执行时会创建运行容器



Serverless 的应用场景

阿里云的函数计算

事件函数 01

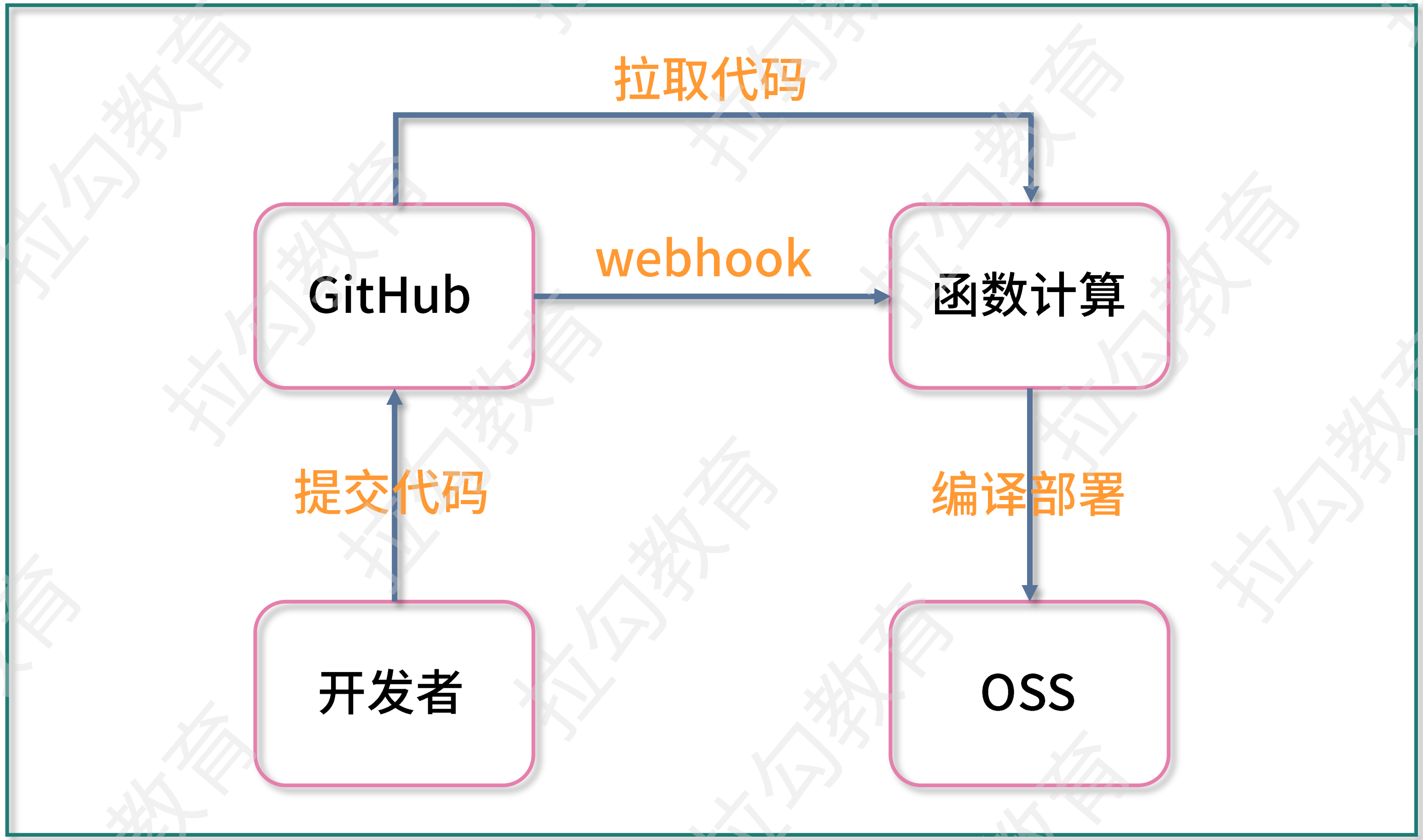
- 通过 SDK 提供的 API 函数调用进行一些轻量计算操作
- 通过配置时间和间隔，自动执行

HTTP函数 02

- 每一个 HTTP 函数都有特定的域名
- 为前后端分离架构的 Web 应用提供后端数据支撑

Serverless 实例

自动部署流程图

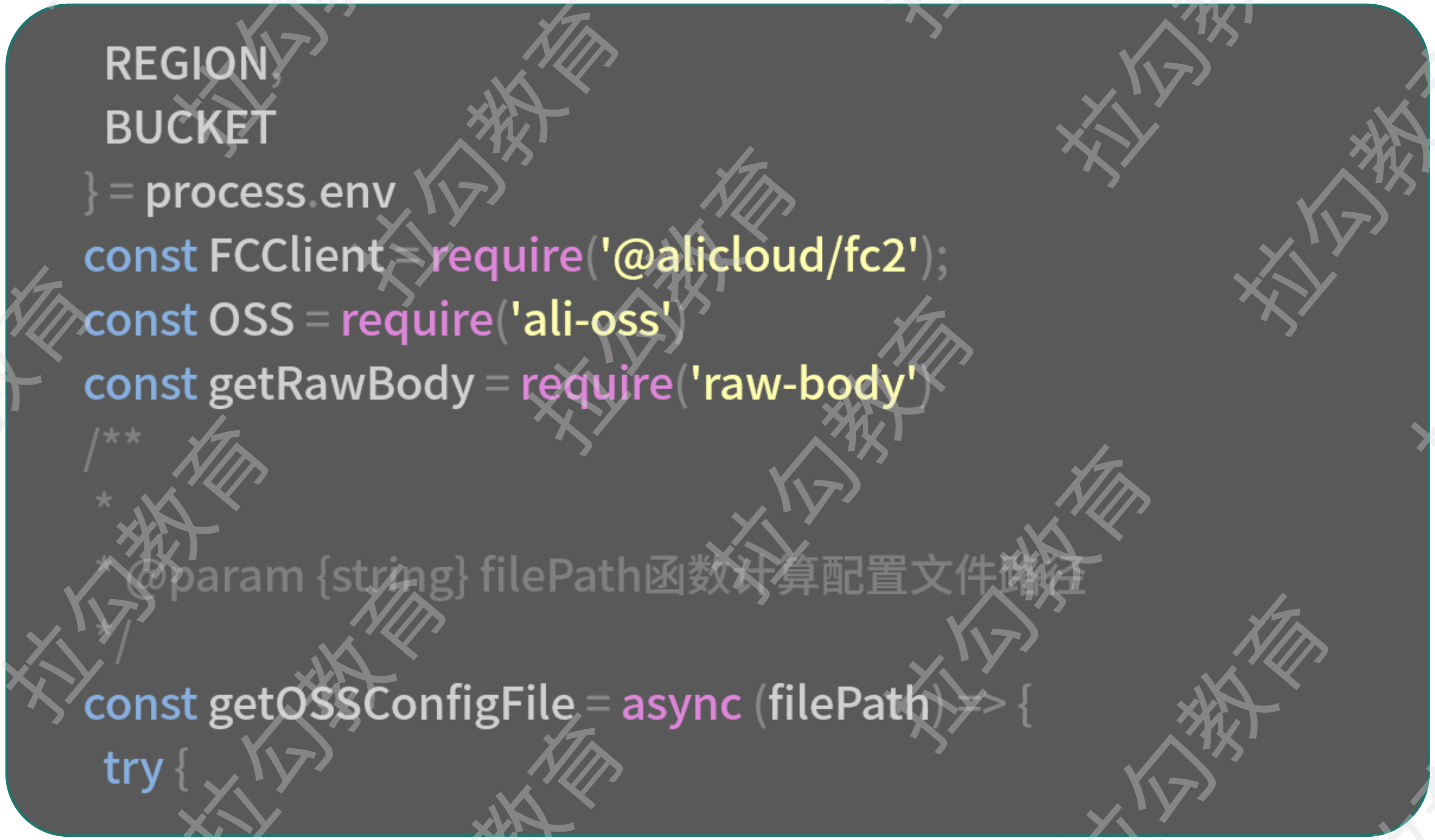


Serverless 实例

拉勾教育
— 互联网人实战大学 —

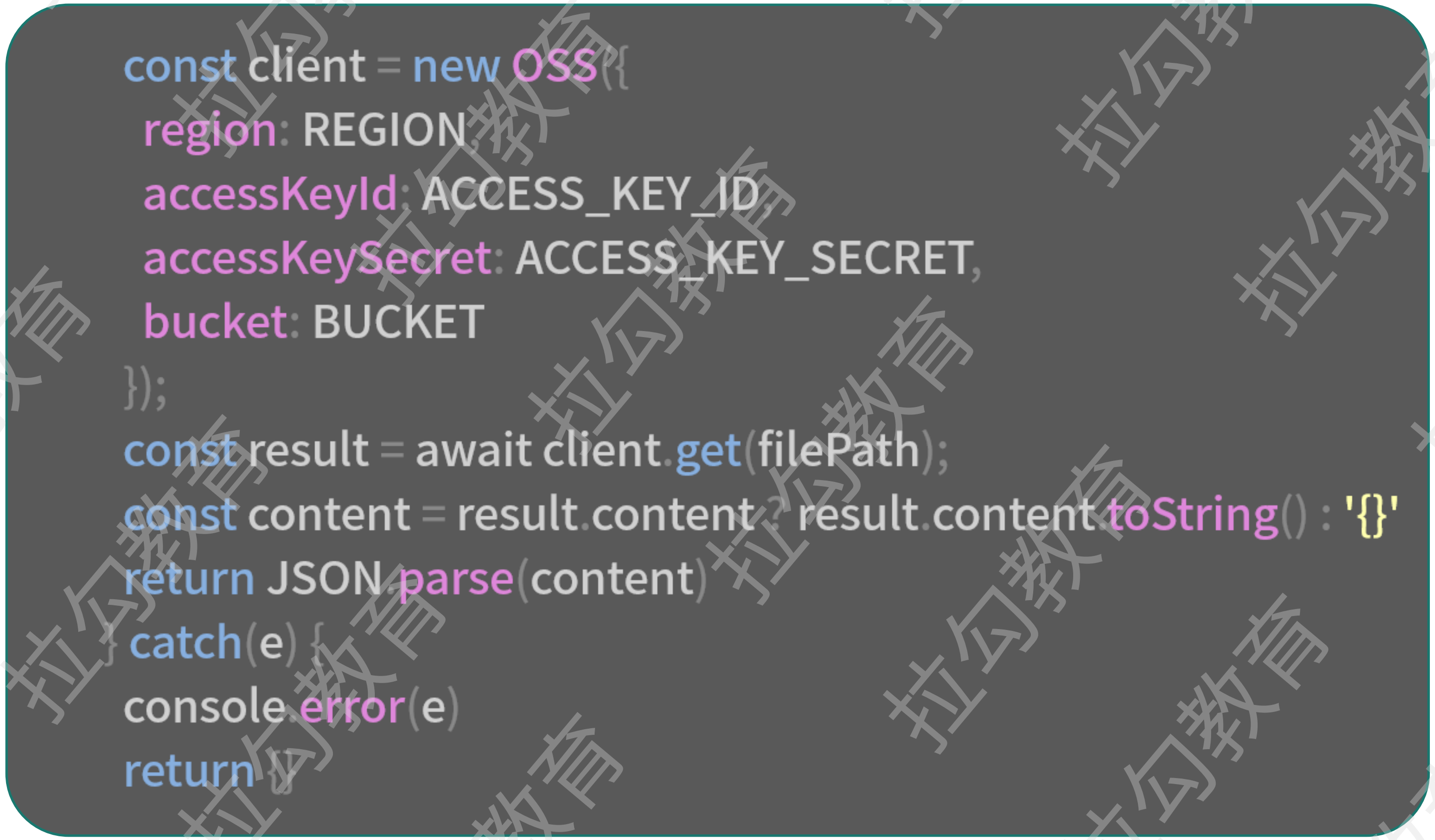
```
/**
 * ACCOUNT_ID z主账号ID
 * ACCESS_KEY_ID 访问 bucket 所需要的 key
 * ACCESS_KEY_SECRET访问bucket  所需要的secret
 * REGION bucket 所在的 region
 * BUCKET 用于储存配置文件的 bucket
 */
const {
 ACCOUNT_ID,
 ACCESS_KEY_ID,
 ACCESS_KEY_SECRET,
```

Serverless 实例



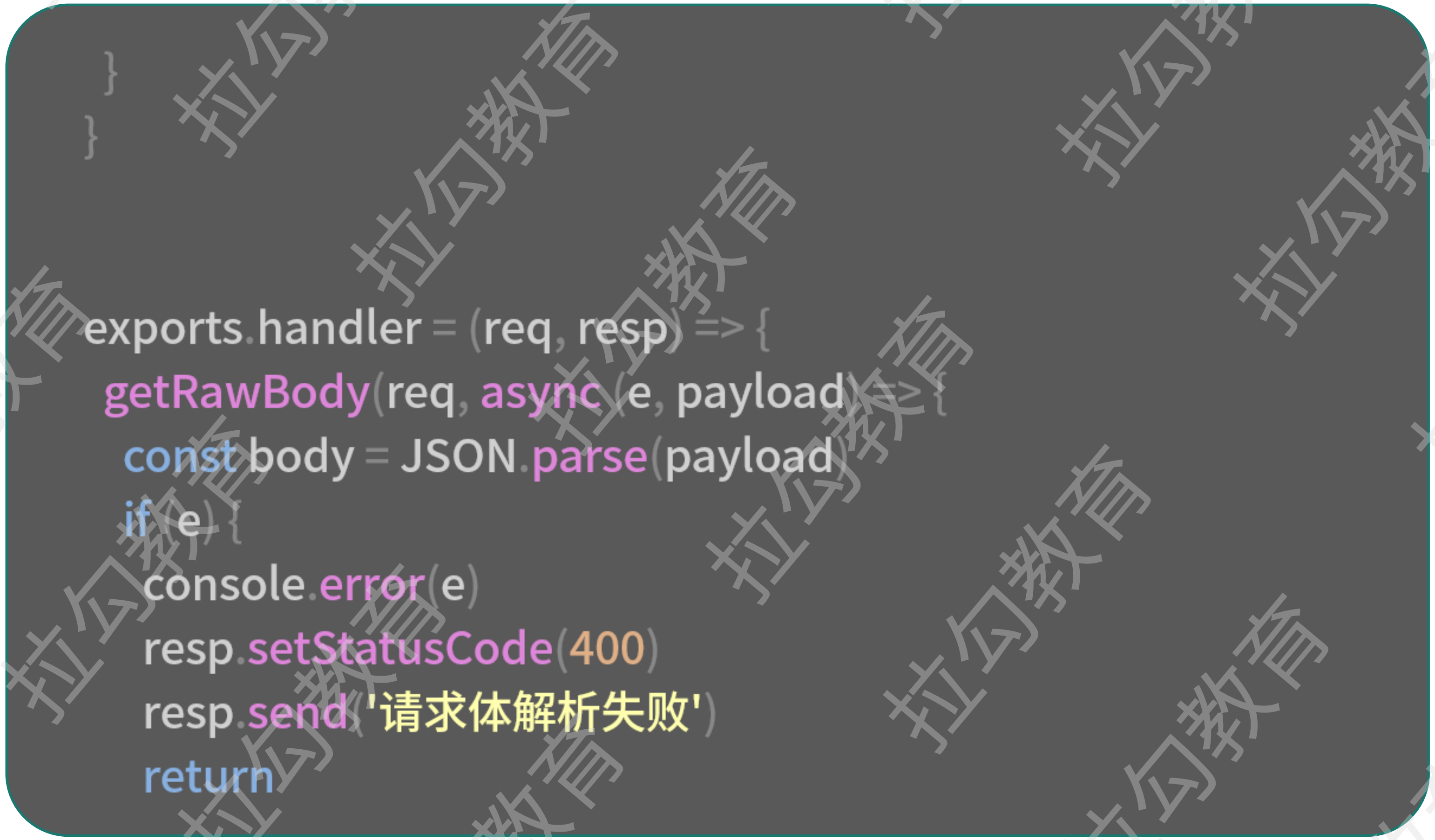
```
  REGION,
  BUCKET
} = process.env
const FCClient = require('@alicloud/fc2');
const OSS = require('ali-oss')
const getRawBody = require('raw-body')
/**
 *
 * @param {string} filePath函数计算配置文件路径
 */
const getOSSConfigFile = async (filePath) => {
  try {
```

Serverless 实例



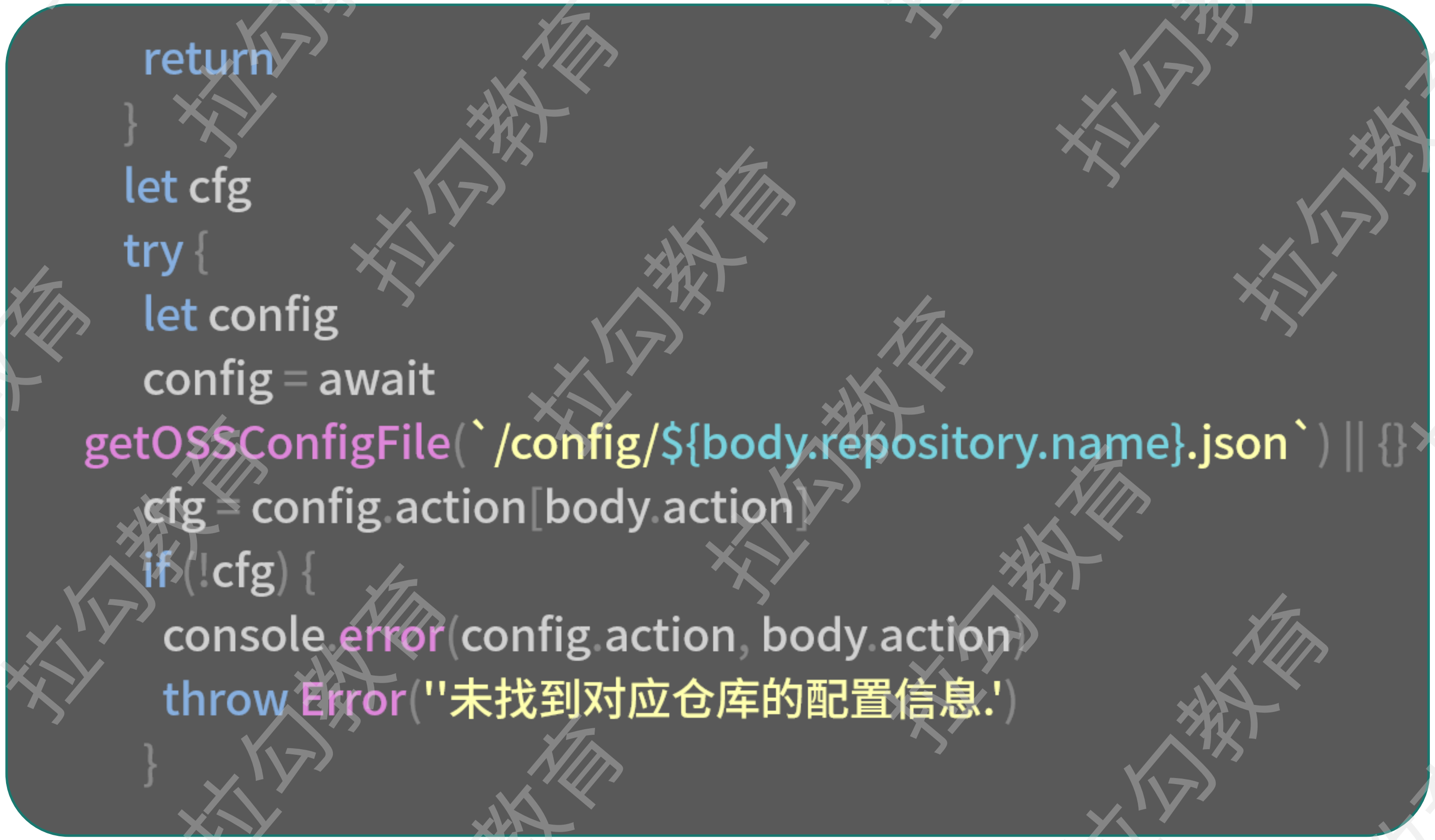
```
const client = new OSS({
  region: REGION,
  accessKeyId: ACCESS_KEY_ID,
  accessKeySecret: ACCESS_KEY_SECRET,
  bucket: BUCKET
});
const result = await client.get(filePath);
const content = result.content ? result.content.toString() : '{}'
return JSON.parse(content)
} catch(e) {
console.error(e)
return {}
```

Serverless 实例



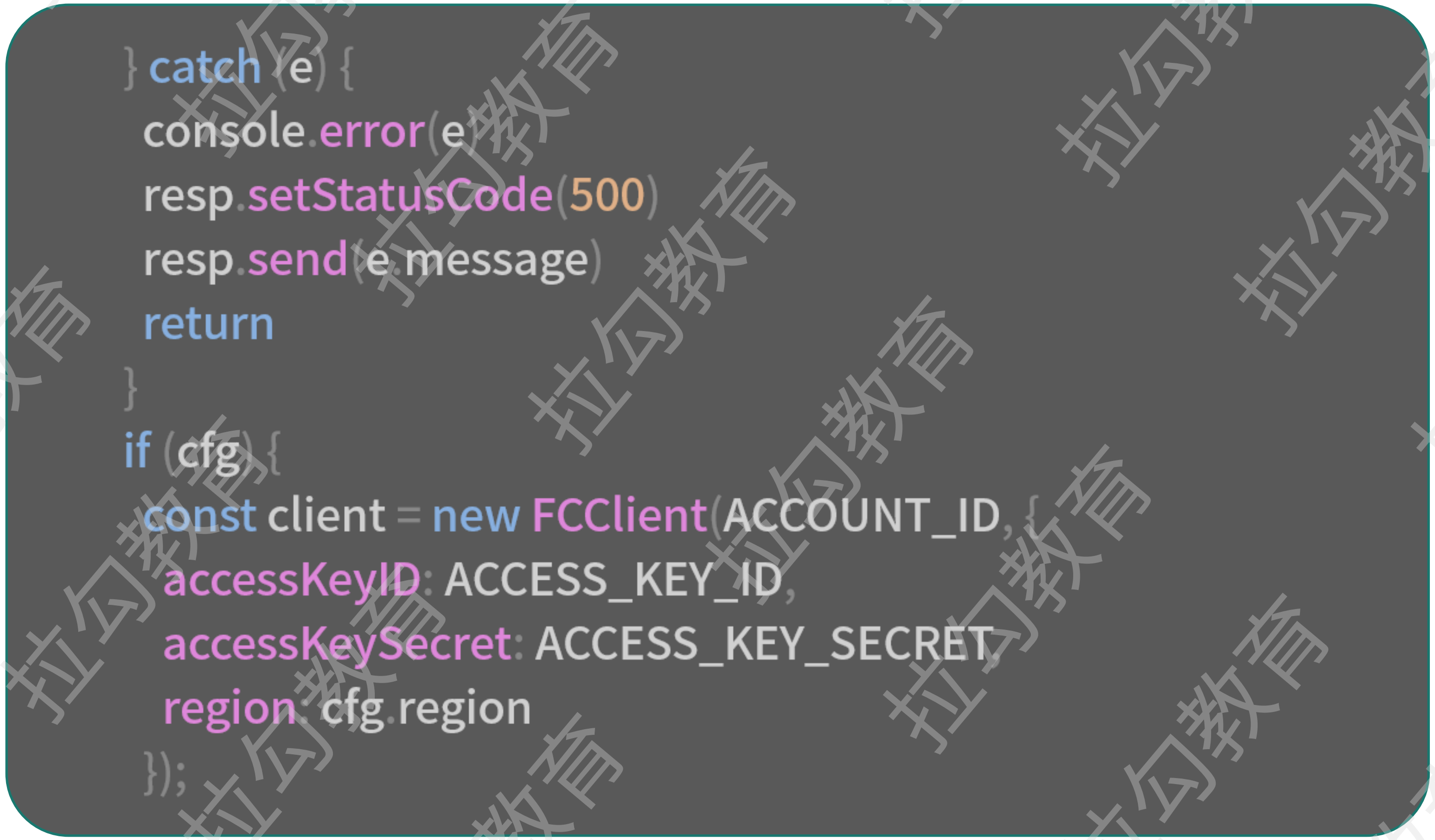
```
    }
  }

exports.handler = (req, resp) => {
  getRawBody(req, async (e, payload) => {
    const body = JSON.parse(payload)
    if (e) {
      console.error(e)
      resp.setStatusCode(400)
      resp.send('请求体解析失败')
      return
```

Serverless 实例



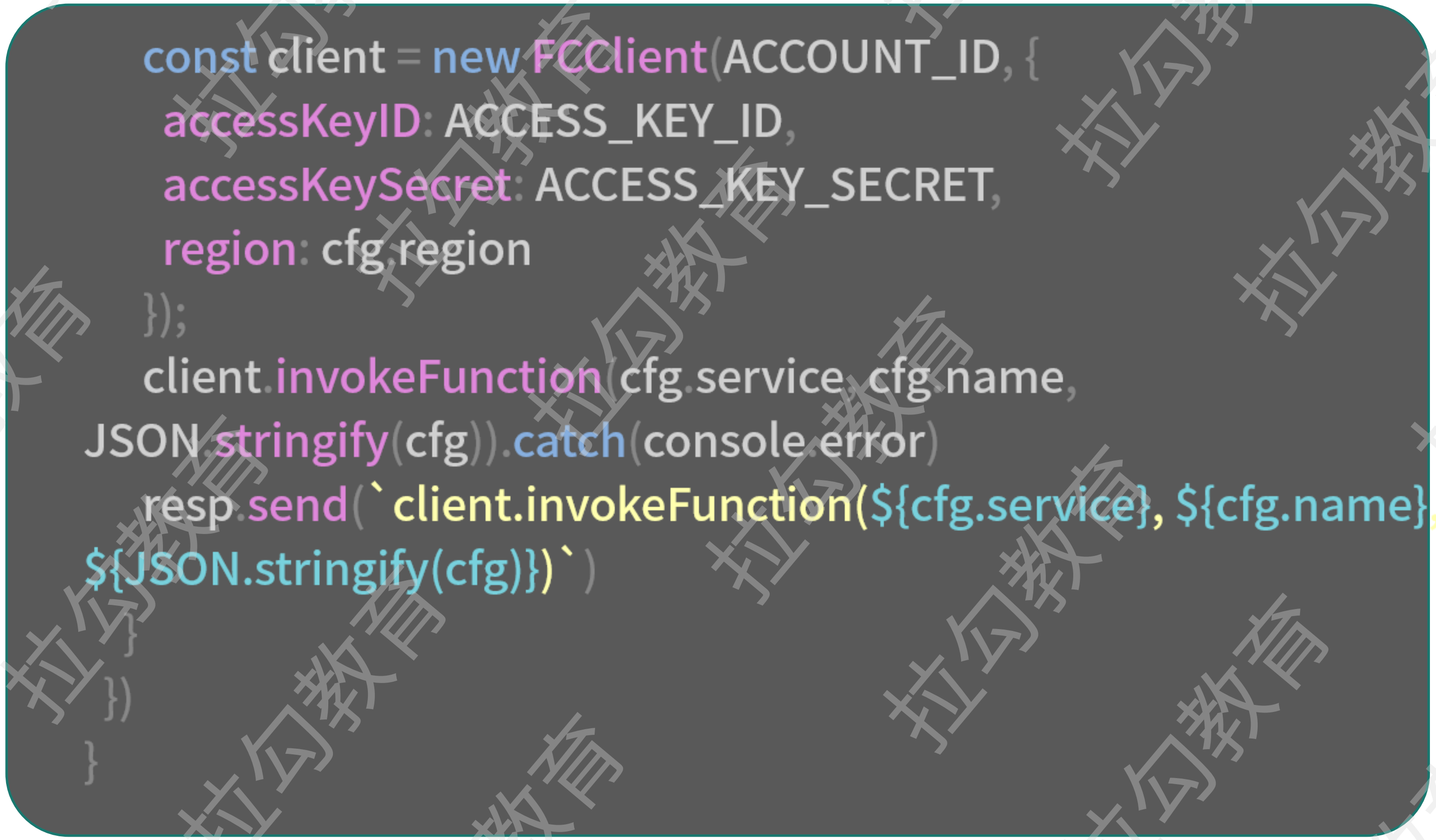
```
    return
   }
  let cfg
  try {
    let config
    config = await
getOSSConfigFile(`/config/${body.repository.name}.json`) || {}
    cfg = config.action[body.action]
    if (!cfg) {
     console.error(config.action, body.action)
     throw Error('"未找到对应仓库的配置信息.')
    }
```

Serverless 实例



```
} catch (e) {
  console.error(e)
  resp.setStatusCode(500)
  resp.send(e.message)
  return
}
if (cfg) {
  const client = new FCClient(ACCOUNT_ID, {
    accessKeyID: ACCESS_KEY_ID,
    accessKeySecret: ACCESS_KEY_SECRET,
    region: cfg.region
  });
```

Serverless 实例



```
    const client = new FCClient(ACCOUNT_ID, {
     accessKeyID: ACCESS_KEY_ID,
     accessKeySecret: ACCESS_KEY_SECRET,
     region: cfg.region
    });
    client.invokeFunction(cfg.service, cfg.name,
JSON.stringify(cfg)).catch(console.error)
    resp.send(`client.invokeFunction(${cfg.service}, ${cfg.name},
${JSON.stringify(cfg)})`)
  }
 })
}
```

Serverless 实例

- 函数执行完成后会存活一段时间

  再次调用会执行之前创建的函数

- 创建了随机目录并修改工作目录到随机目录下

  以获取写权限

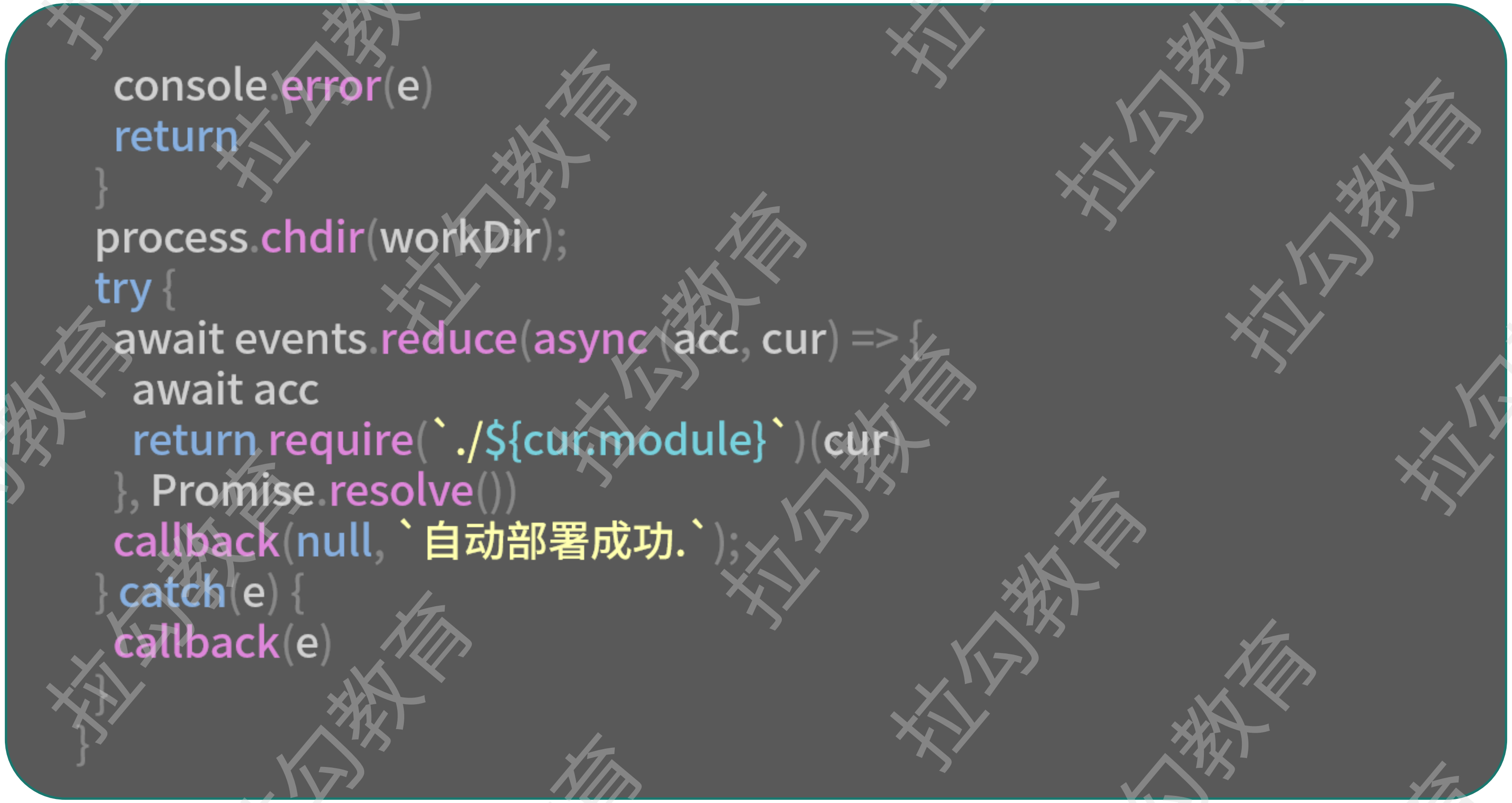
Serverless 实例

```
const fs = require('fs')
/**
 *
 * @param {*} event
 * {
 *   repo仓库地址
 *   region   bucket所在区域
 *   bucket编译后部署的bucket
 *   command编译命令
 * }
 * @param {*} context
 * @param {*} callback
 */
exports.handler = async (event, context, callback) => {
```

Serverless 实例

```
exports.handler = async (event, context, callback) => {
 const {events} = Buffer.isBuffer(event) ?
JSON.parse(event.toString()) : event
 let dir = Math.random().toString(36).substr(6)
 //设置随机临时工作目录，避免容器未销毁的情况下，重复拉取仓库失败
 const workDir = `/tmp/${dir}`
 //为了保证后续流程能找到临时工作目录，设置为全局变量
 global.workDir = workDir
 try {
  fs.mkdirSync(workDir)
 } catch(e) {
  console.error(e)
  return
 }

 process.chdir(workDir);
```

Serverless 实例



```
  console.error(e)
  return
 }
 process.chdir(workDir);
 try {
  await events.reduce(async (acc, cur) => {
   await acc
   return require(`./${cur.module}`)(cur)
  }, Promise.resolve())
  callback(null, `自动部署成功.`);
 } catch (e) {
  callback(e)
 }
}
```

Serverless 实例

01
拉取仓库代码

02
安装依赖并构建

03
将生成的代码上传部署

Serverless 实例

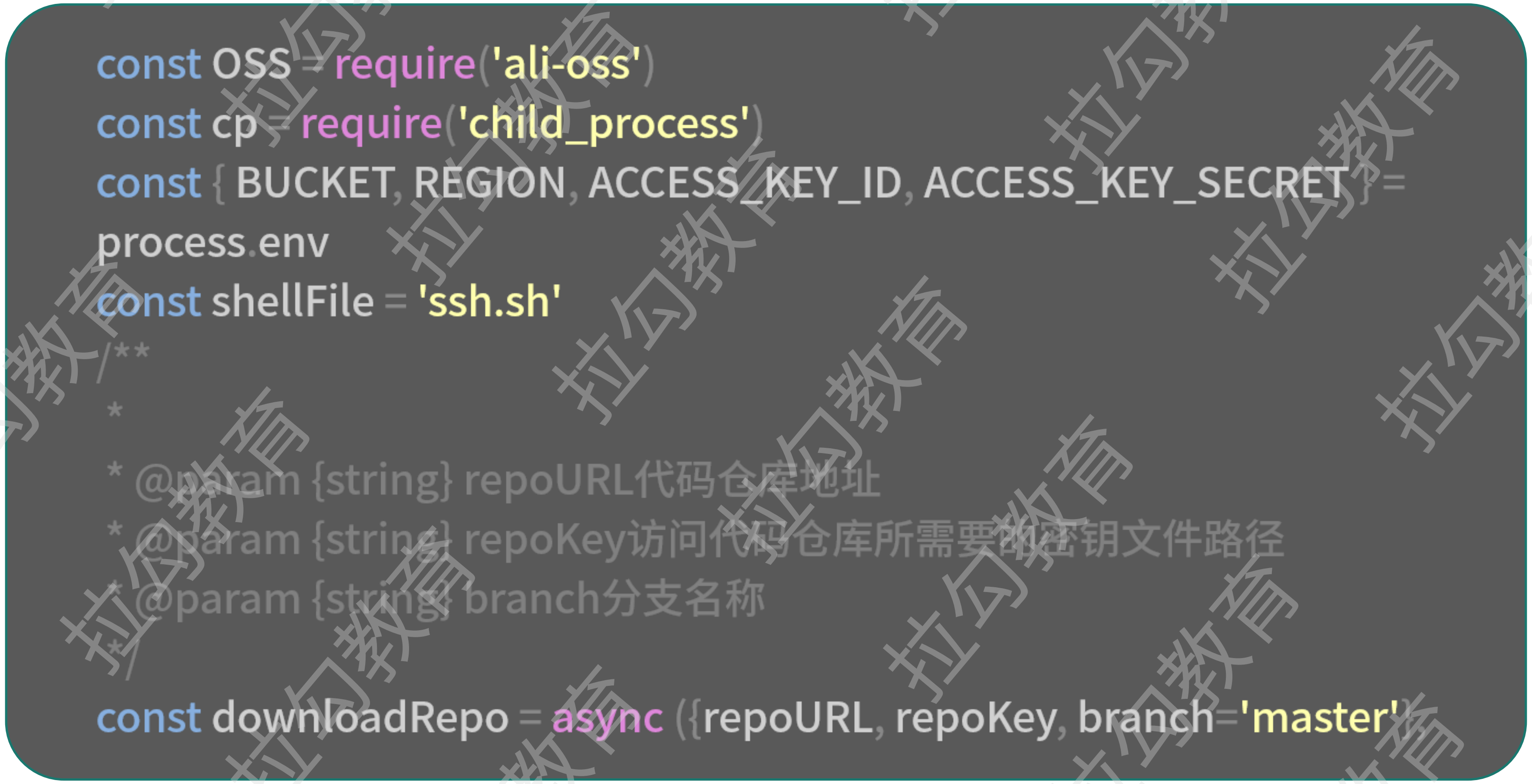
拉勾教育
— 互联网人实战大学 —

- 身份认证
- 首次进行 git clone 操作

Serverless 实例

```
#!/bin/sh
ID_RSA=/tmp/id_rsa
exec /usr/bin/ssh -o StrictHostKeyChecking=no -o
GSSAPIAuthentication=no -i $ID_RSA "$@"
```

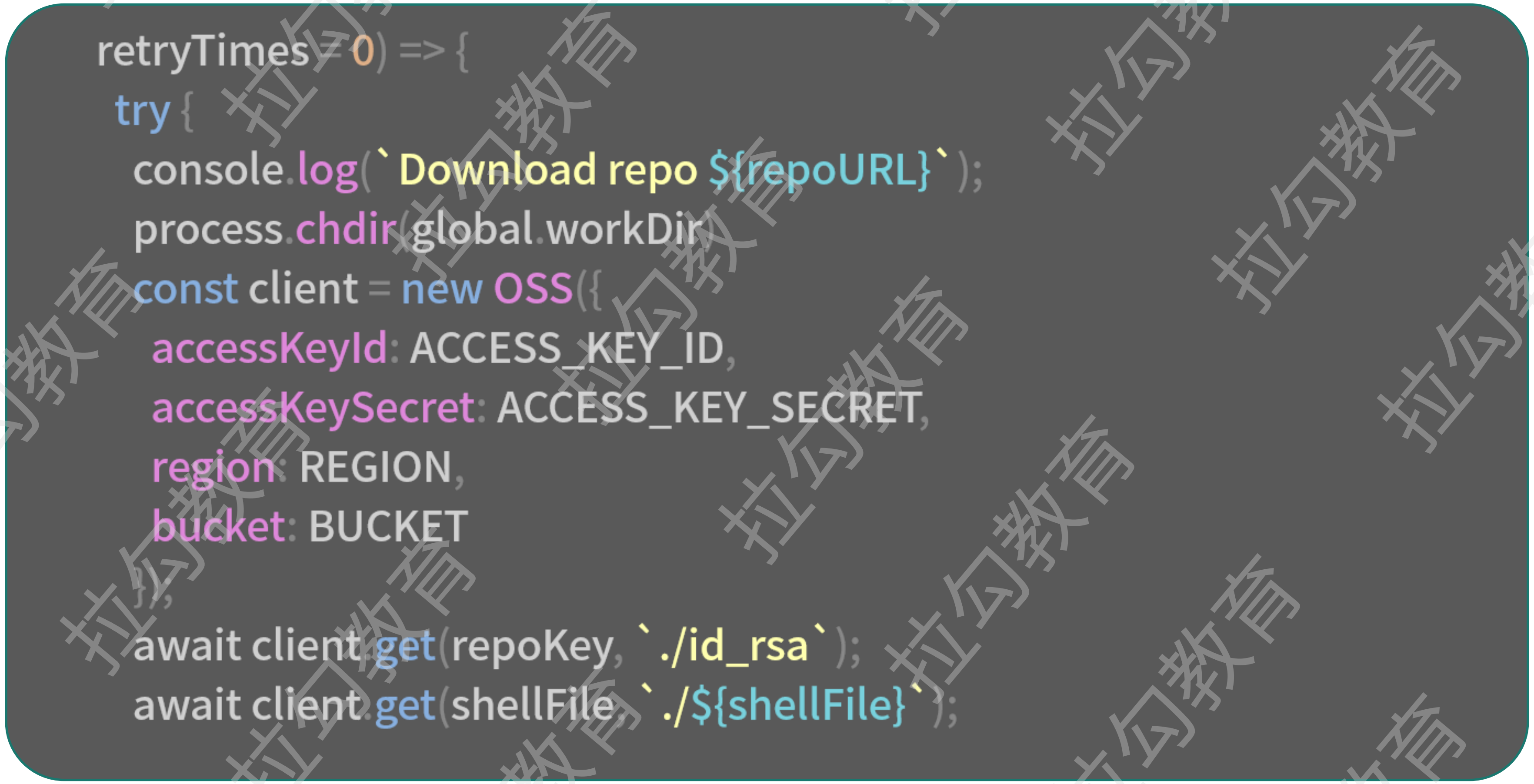
Serverless 实例



```
const OSS = require('ali-oss')
const cp = require('child_process')
const { BUCKET, REGION, ACCESS_KEY_ID, ACCESS_KEY_SECRET } =
process.env
const shellFile = 'ssh.sh'
/**
 *
 * @param {string} repoURL代码仓库地址
 * @param {string} repoKey访问代码仓库所需要的密钥文件路径
 * @param {string} branch分支名称
 */

const downloadRepo = async ({repoURL, repoKey, branch='master'
```

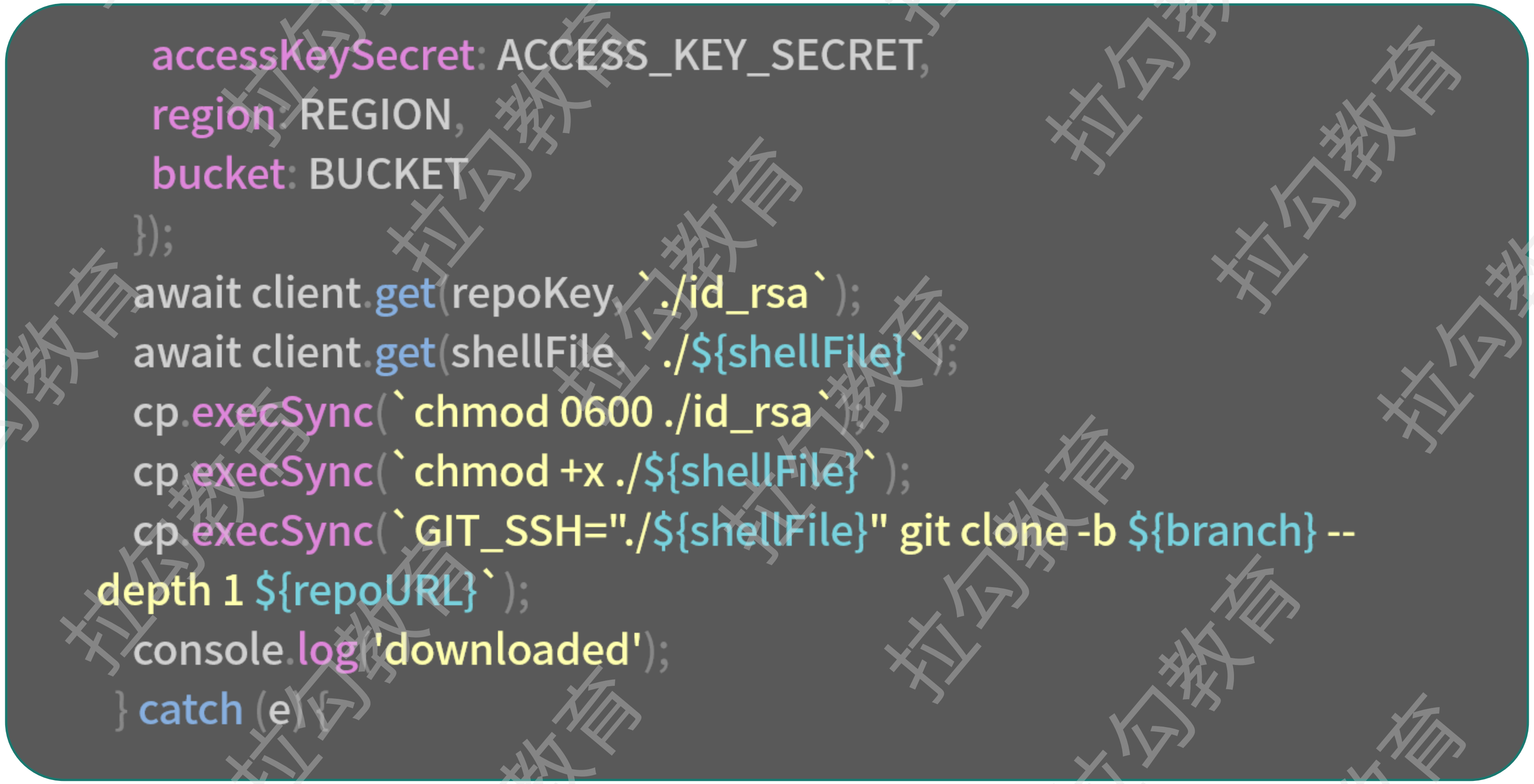
Serverless 实例

拉勾教育
— 互联网人实战大学 —



```
retryTimes = 0) => {
 try {
  console.log(`Download repo ${repoURL}`);
  process.chdir(global.workDir);
  const client = new OSS({
   accessKeyId: ACCESS_KEY_ID,
   accessKeySecret: ACCESS_KEY_SECRET,
   region: REGION,
   bucket: BUCKET
  });
  await client.get(repoKey, `./id_rsa`);
  await client.get(shellFile, `./${shellFile}`);
```

Serverless 实例

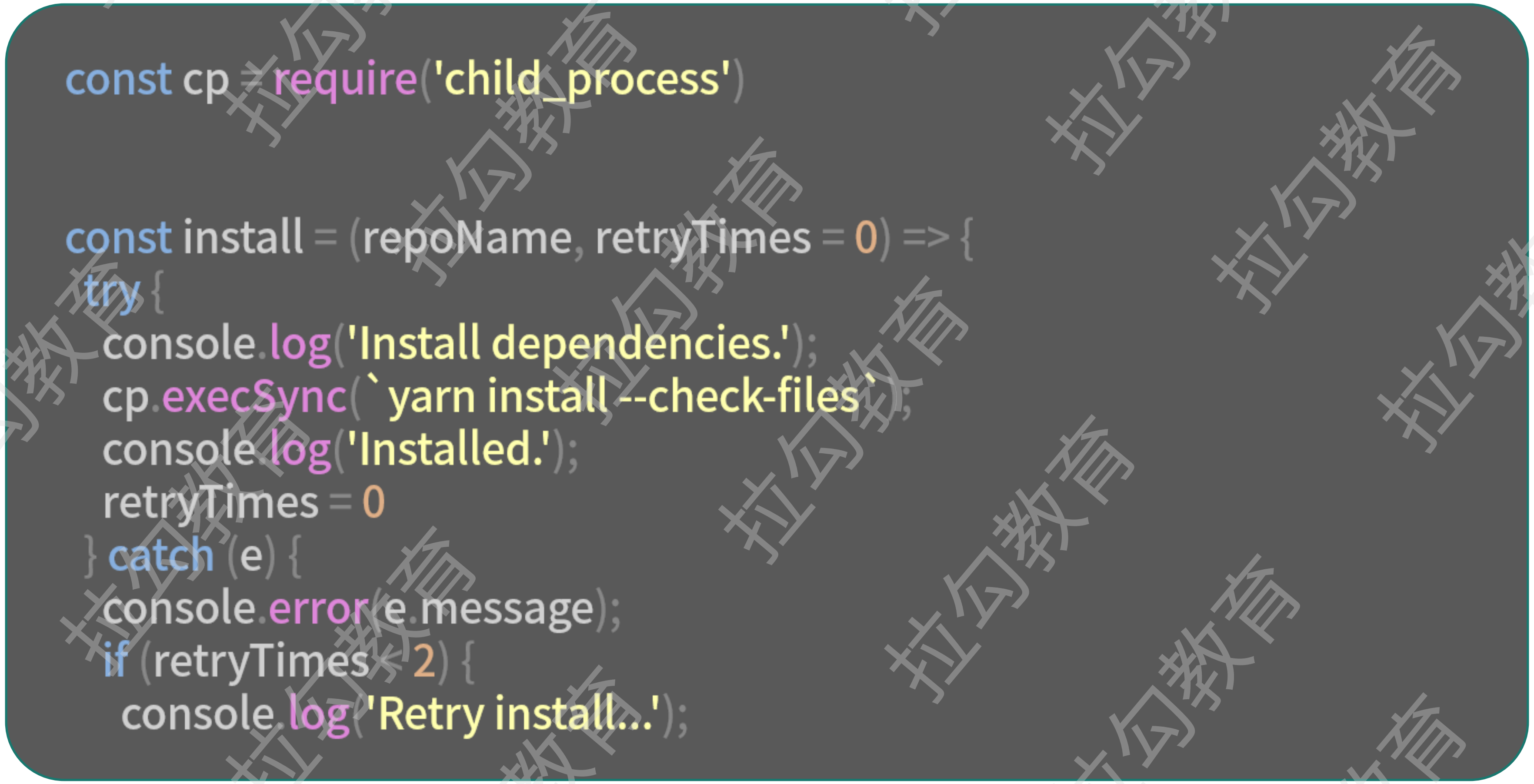


```
    accessKeySecret: ACCESS_KEY_SECRET,
    region: REGION,
    bucket: BUCKET
  });
  await client.get(repoKey, `./id_rsa`);
  await client.get(shellFile, `./${shellFile}`);
  cp.execSync(`chmod 0600 ./id_rsa`);
  cp.execSync(`chmod +x ./${shellFile}`);
  cp.execSync(`GIT_SSH="./${shellFile}" git clone -b ${branch} --
depth 1 ${repoURL}`);
  console.log('downloaded');
 } catch (e) {
```

Serverless 实例

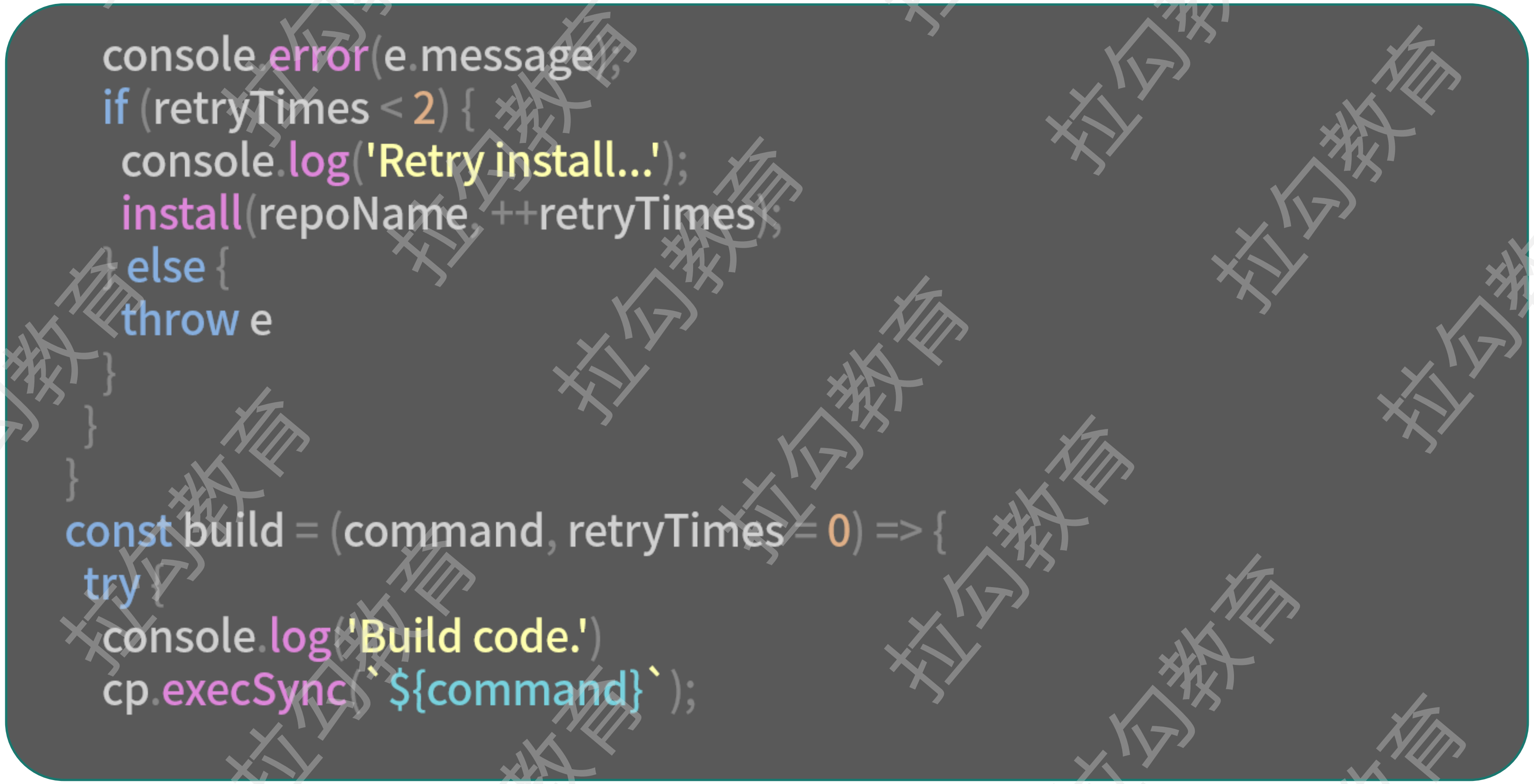
```
depth 1 ${repoURL}`);
   console.log('downloaded');
  } catch (e) {
   console.error(e);
   if (retryTimes < 2) {
    downloadRepo({repoURL, repoKey, branch}, retryTimes++);
   } else {
    throw e
   }
  }
};
module.exports = downloadRepo
```

Serverless 实例



```
const cp = require('child_process')


const install = (repoName, retryTimes = 0) => {
 try {
  console.log('Install dependencies.');
  cp.execSync(`yarn install --check-files`);
  console.log('Installed.');
  retryTimes = 0
 } catch (e) {
  console.error(e.message);
  if (retryTimes < 2) {
   console.log('Retry install...');
```

Serverless 实例



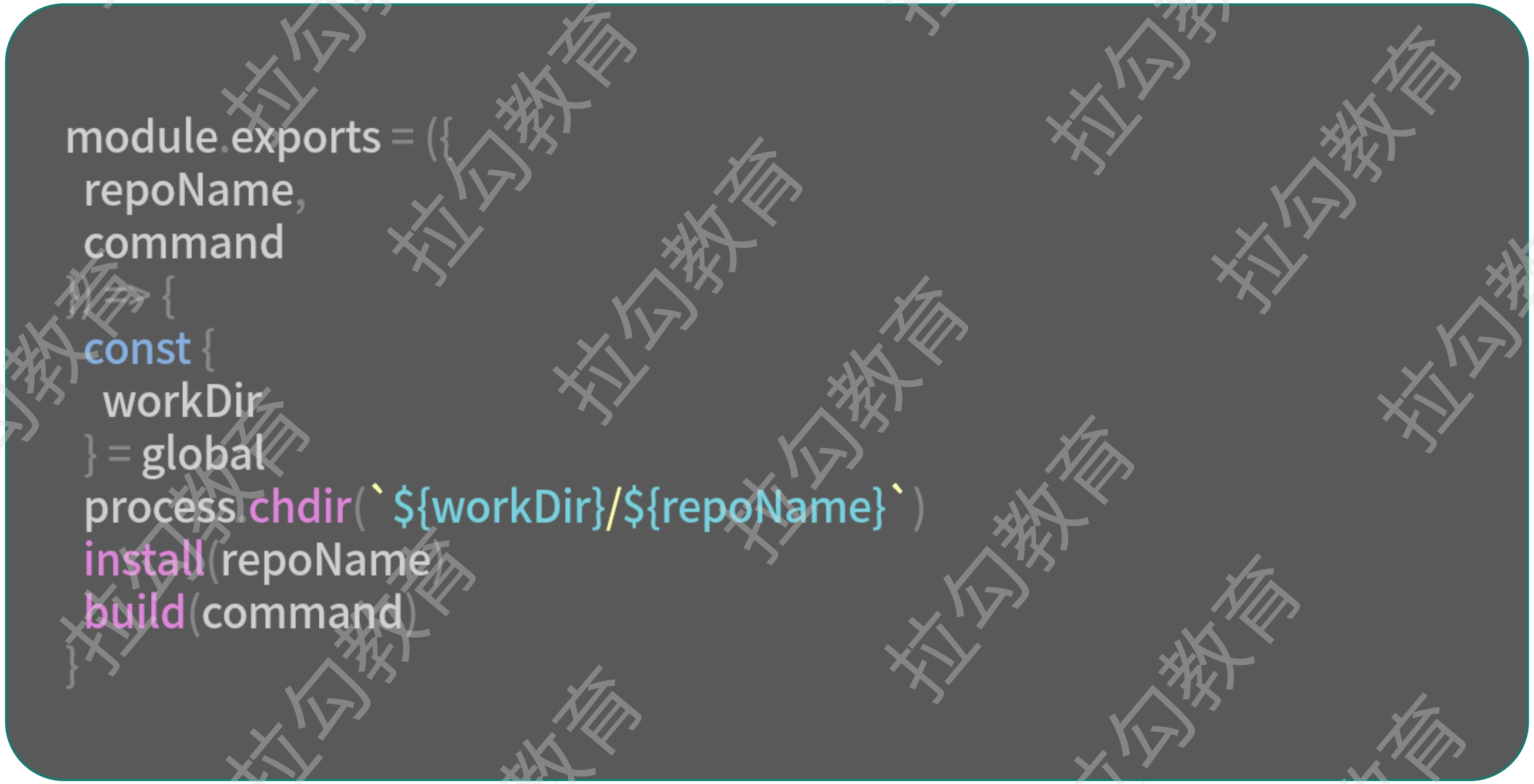
```
    console.error(e.message);
    if (retryTimes < 2) {
      console.log('Retry install...');
      install(repoName, ++retryTimes);
    } else {
      throw e
    }
  }
}

const build = (command, retryTimes = 0) => {
  try {
    console.log('Build code.')
    cp.execSync(`${command}`);
```

Serverless 实例

```
      console.log('Built.');
    } catch (e) {
      console.error(e.message);
      if (retryTimes < 2) {
        console.log('Retry build...');
        build(command, ++retryTimes);
      } else {
        throw e
      }
    }
  };

  module.exports = ({
```

Serverless 实例

拉勾教育
— 互 联 网 人 实 战 大 学 —

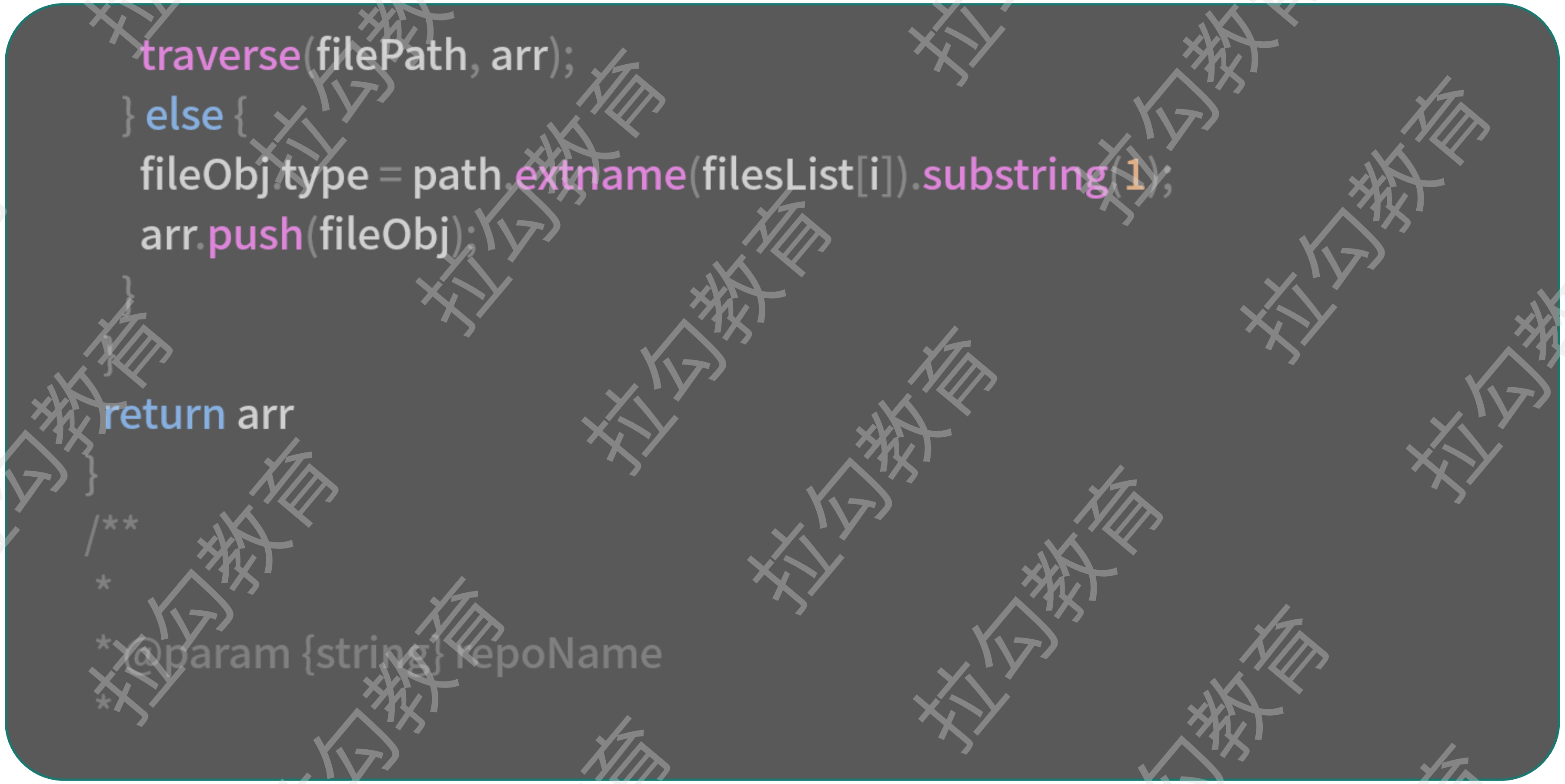


```
module.exports = ({
  repoName,
  command
}) => {
  const {
    workDir
  } = global
  process.chdir(`${workDir}/${repoName}`)
  install(repoName)
  build(command)
}
```

Serverless 实例

```
const path = require('path');
const OSS = require('ali-oss');
//遍历函数
const traverse = (dirPath, arr = []) => {
  var filesList = fs.readdirSync(dirPath);
  for (var i = 0; i < filesList.length; i++) {
    var fileObj = {};
    fileObj.name = path.join(dirPath, filesList[i]);
    var filePath = path.join(dirPath, filesList[i]);
    var stats = fs.statSync(filePath);
    if (stats.isDirectory()) {
      traverse(filePath, arr);
```

Serverless 实例

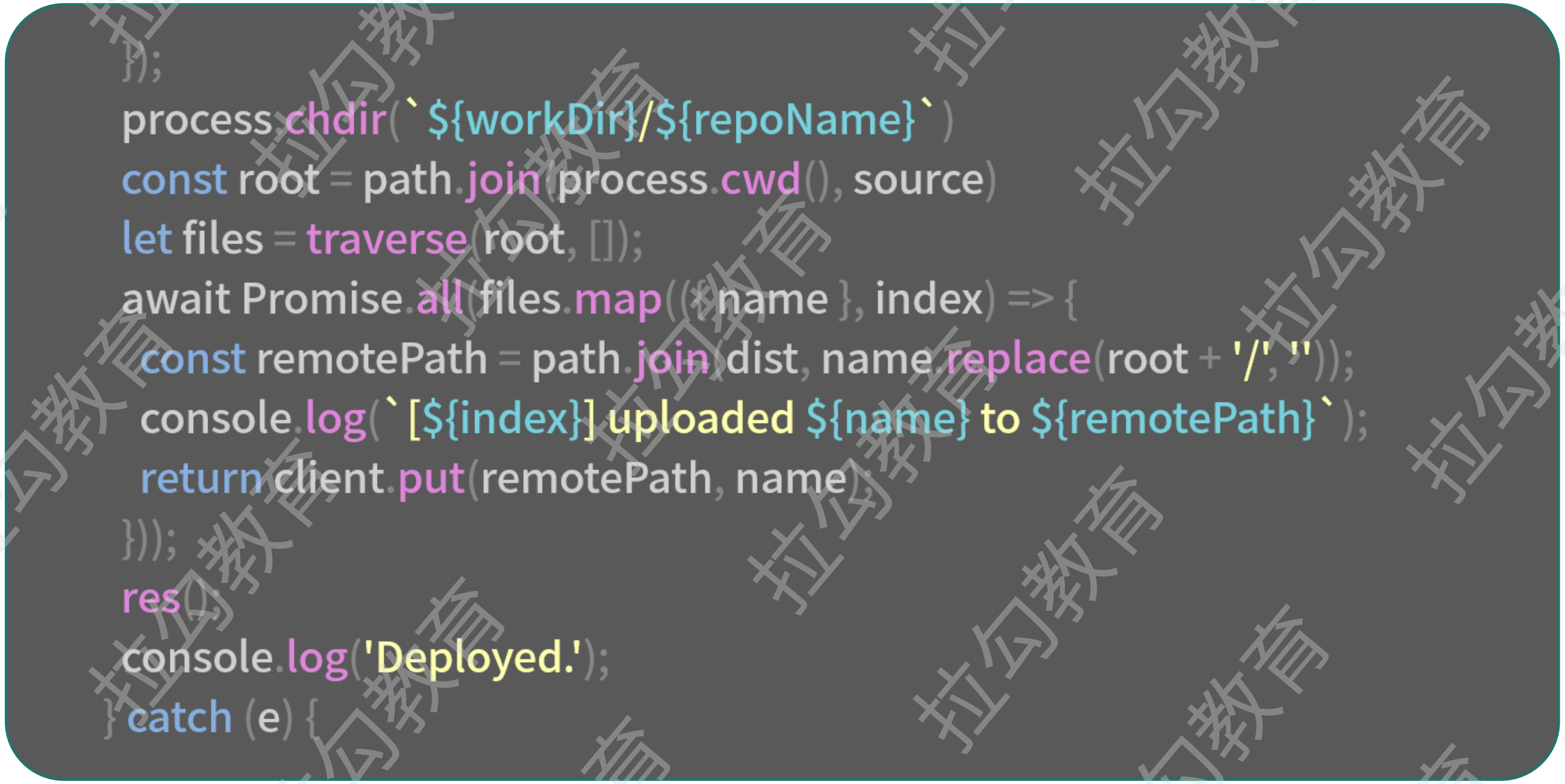


```
      traverse(filePath, arr);
    } else {
      fileObj.type = path.extname(filesList[i]).substring(1);
      arr.push(fileObj);
    }
  }
  return arr
}
/**
 *
 * @param {string} repoName
 *
```

Serverless 实例

```
*/
const deploy = ({ dist = '', source, region, accessKeyId, accessKeySecret,
bucket, repoName }, retryTimes = 0) => new Promise(async (res) => {
 const { workDir } = global
 console.log('Deploy.');
 try {
  const client = new OSS({
   region,
   accessKeyId,
   accessKeySecret,
   bucket
```

Serverless 实例



```
  });
  process.chdir(`${workDir}/${repoName}`)
  const root = path.join(process.cwd(), source)
  let files = traverse(root, []);
  await Promise.all(files.map(({ name }, index) => {
    const remotePath = path.join(dist, name.replace(root + '/', ''));
    console.log(`[${index}] uploaded ${name} to ${remotePath}`);
    return client.put(remotePath, name);
  }));

  res();
  console.log('Deployed.');
} catch (e) {
```

Serverless 实例

```
  } catch (e) {
   console.error(e);
   if (retryTimes < 2) {
    console.log('Retry deploy.');
    deploy({ dist, source, region, accessKeyId, accessKeySecret, bucket },
++retryTimes);
   } else {
    throw e
   }
  }
})

module.exports = deploy
```

总结

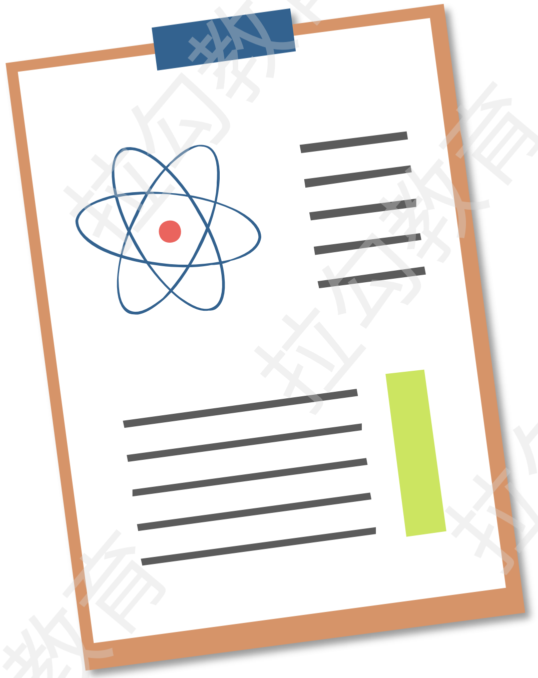
Serverless 是一个具有通用性、开箱即用的产品

概念介绍以及函数计算的具体实例

作业题

拉勾教育
— 互联网人实战大学 —

尝试部署一个 Serverless 服务



Next：31 | 《微前端和功能的可重用性》