阿里云
对象存储

用户实践

文档版本：20190809

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 OSS+ROS创建Sharepoint 2016

本文介绍如何通过阿里云的服务快速创建 Sharepoint2016。

说明:

本文示例由阿里云用户 肖伊 提供，仅供参考。

背景

- 对象存储 OSS

  海量、安全、低成本、高可靠的云存储服务，提供99.9999999999%（12个9）的数据可靠性。使用RESTful API，可以在互联网任何位置存储和访问。容量和处理能力可弹性扩展，并能提供多种可选择的存储类型，全面优化存储成本。

- 资源编排 ROS

  资源编排（Resource Orchestration）是一种简单易用的云计算资源管理及自动化运维服务。用户通过模板描述多个云计算资源的依赖关系及配置等，并自动完成所有资源的创建和配置，从而达到自动化部署及运维等目的。编排模板同时也是一种标准化的资源和应用交付方式，可以随时编辑修改，使基础设施即代码（Infrastructure as Code）成为可能。

- PowerShell

  PowerShell是Windows下的一种命令行外壳程序及环境脚本，用户可以编写 .ps1 脚本或利用 .Net Framework 进行脚本的编写和运行，与 Liunx 下的.sh 脚本类似。

- Registry

  Windows注册表，本示例中用到了以下的KEY：

  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon:

  `AutoAdminLogon`： 是否开启管理员自动登录，1: True 0: False

  `DefaultUserName`： 自动登录管理员的账号

  `DefaultPassword`： 自动登录管理员的密码

  在整个最佳实践当中，我们会组合使用阿里云的下列服务以完成整体工作：

  - 阿里云OSS：我们将使用该服务提供的存储能力，在通过本地网络下载 SharePoint 的安装文件后，在 OSS 上创建“存储空间”并保存安装文件，再以内网的形式提供给需要安装 SharePoint 的机器。
  - 阿里云资源编排ROS：我们将用到下文提到的 ROS 模板，通过 ROS 自动实现从 ECS 创建到 Sharepoint 安装的整个过程。你只需在使用 ROS 进行自动化安装前输入必要的参数即可。

步骤

1. 新建 Bucket

   首先，在 OSS 中创建新的 Bucket，其中 Bucket ACL 建议设置为私有。

   Bucket 创建完成后，请从此处下载 img 文件，并存放至 OSS 中。

   这里建议将 img 文件的读写权限设置成公共读。

   该步骤完成后，我们可以获得一个阿里云提供的 URL 以访问该文件。

2. 通过ROS新建资源栈

ROS 以模板的形式申明资源（如 ECS 和 VPC）并配置资源之间的关系（如 ECS 属于哪个VPC），并支持在 ECS 部署结束后自动执行用户脚本。模板中定义的所有资源都属于一个栈，用户通过资源栈管理自己的云资源。

a. 进入ROS管理控制台，单击 资源栈管理 > 新建资源栈，开始创建。

b. 在输入脚本页面当中，用户需要选择将脚本中所创建的机器部署到哪一个Region。

参考脚本如下：

```
{
"ROSTemplateFormatVersion": "2015-09-01",
"Description": "One simple ECS instance and a security group. The
user only needs to specify the image ID.",
"Parameters": {
 "NewDomainNetbiosName": {
   "Type": "String",
   "Default": "ADXING"
 },
 "InternetMaxBandwidthOut": {
   "Type": "String",
   "Description": "Set internet output bandwidth of instance.
Unit is Mbps(Mega bit per second). Range is [0,200]. Default is
 1.While the property is not 0, public ip will be assigned for
instance.  ",
   "MinLength": "1",
   "MaxLength": "41"
 },
 "ZoneId": {
   "Type": "String",
   "Description": "The available zone Id",
   "AllowedValues": [
     "cn-shenzhen-c"
   ]
 },
 "DomainName": {
   "Type": "String",
   "Default": "adxing.com"
 },
 "SPFarmAccountPassword": {
   "NoEcho": true,
   "Type": "String",
   "Default": "Banana#12345"
 },
 "SPISOImageURI": {
   "Type": "String",
   "AllowedPattern": "^(?i)(s3|http|https):\\/\\/.+",
   "Default": "http://sharepointbucket.oss-cn-shenzhen-internal.
aliyuncs.com/officeserver.img"
 },
 "ImageId": {
   "Type": "String",
   "Description": "Image Id, represents the image resource to
startup one ECS instance,, <a href='#/product/cn-shenzhen/list/
imageList' target='_blank'>View image resources</a>",
```

```
    "Default": "win2012r2_64_dtc_17196_en-us_40G_alibase_20170915.
vhd"
  },
  "SPFarmAccount": {
    "Type": "String",
    "Default": "spFarmAcc"
  },
  "InstanceType": {
    "Type": "String",
    "Description": "The instance type",
    "AllowedValues": [
      "ecs.c5.xlarge",
      "ecs.s1.small",
      "ecs.n4.small",
      "ecs.n4.large",
      "ecs.n4.xlarge",
      "ecs.mn4.small",
      "ecs.mn4.large",
      "ecs.mn4.xlarge",
      "ecs.n1.small",
      "ecs.n1.medium",
      "ecs.n1.large"
    ],
    "Default": "ecs.c5.xlarge"
  },
  "DomainAdminPassword": {
    "NoEcho": true,
    "Type": "String",
    "Default": "Banana#12345"
  },
  "DomainAdminUser": {
    "Type": "String",
    "Default": "spAdmin"
  },
  "Password": {
    "NoEcho": true,
    "Type": "String",
    "Default": "Banana12345"
  }
},
"Resources": {
  "WebServer": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "InternetMaxBandwidthOut": {
        "Ref": "InternetMaxBandwidthOut"
      },
      "UserData": {
        "Fn::Base64": {
          "Fn::Join": [
            "",
            [
              "[powershell]\n",
              "$webclient = New-Object System.Net.WebClient\n",
              "$url = 'http://sharepointbucket.oss-cn-shenzhen.
aliyuncs.com/ros/Archive.zip'\n",
              "$file = 'C:/Archive.zip'\n",
              "$webclient.DownloadFile($url,$file)\n",
              "Expand-Archive -Path 'C:/Archive.zip' -Destinatio
nPath 'C:/sp' -Force \n",
              "$webclientSP = New-Object System.Net.WebClient\n",
              "$officeserverfile = 'C:/officeserver.img'\n",
              "$spURL = '",
              {
```

```
                "Ref": "SPISOImageURI"
              },
              "'\n",
              "$webclientSP.DownloadFile($spURL,$officeserverfile)\
n",
              "[bat]\n",
              "reg add 'HKLM\\SOFTWARE\\Microsoft\\Windows NT\\
CurrentVersion\\Winlogon' /v AutoAdminLogon  /d 1 /f /reg:64\n",
              "reg add 'HKLM\\SOFTWARE\\Microsoft\\Windows NT\\
CurrentVersion\\Winlogon' /v DefaultUserName   /d administrator /f
 /reg:64 \n",
              "reg add 'HKLM\\SOFTWARE\\Microsoft\\Windows NT\\
CurrentVersion\\Winlogon' /v DefaultPassword   /d  ",
              {
                "Ref": "Password"
              },
              " /f /reg:64 \n",
              "reg add 'HKLM\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\RunOnce' /v Install /d 'powershell.exe -Command c
:/sp/STEP1.ps1 ",
              "-DomainName  ",
              {
                "Ref": "DomainName"
              },
              " -NewDomainNetbiosName  ",
              {
                "Ref": "NewDomainNetbiosName"
              },
              " -DomainAdminPassword  ",
              {
                "Ref": "DomainAdminPassword"
              },
              " -DomainAdminUser  ",
              {
                "Ref": "DomainAdminUser"
              },
              " -SPFarmAccount  ",
              {
                "Ref": "SPFarmAccount"
              },
              " -SPFarmAccountPassword  ",
              {
                "Ref": "SPFarmAccountPassword"
              },
              " ' /f /reg:64 \n",
              "shutdown -r -t 2\n"
            ]
          ]
        }
      },
      "SecurityGroupId": {
        "Ref": "SecurityGroup"
      },
      "ImageId": {
        "Ref": "ImageId"
      },
      "InstanceType": {
        "Ref": "InstanceType"
      },
      "Password": {
        "Ref": "Password"
      }
    }
  },
```

```
  "SecurityGroup": {
    "Type": "ALIYUN::ECS::SecurityGroup",
    "Properties": {
      "SecurityGroupIngress": [
        {
          "PortRange": "-1/-1",
          "Priority": 1,
          "SourceCidrIp": "0.0.0.0/0",
          "IpProtocol": "all",
          "NicType": "internet"
        }
      ],
      "SecurityGroupEgress": [
        {
          "PortRange": "-1/-1",
          "Priority": 1,
          "IpProtocol": "all",
          "DestCidrIp": "0.0.0.0/0",
          "NicType": "internet"
        }
      ]
    }
  }
},
"Outputs": {
  "InstanceId": {
    "Value": {
      "Fn::GetAtt": [
        "WebServer",
        "InstanceId"
      ]
    }
  },
  "PublicIp": {
    "Value": {
      "Fn::GetAtt": [
        "WebServer",
        "PublicIp"
      ]
    }
  },
  "SecurityGroupId": {
    "Value": {
      "Fn::GetAtt": [
        "SecurityGroup",
        "SecurityGroupId"
      ]
    }
  }
}
```

```
}
```

c. 完成模版输入后，单击下一步，对资源栈进行参数配置。

需要配置的参数如下（关于参数的描述，请参考参数）：

ImageId：ECS 所使用的镜像，本示例中，我们使用 windows2012R2

InternetMaxBandwidthOut：ECS 的出口带宽

ZoneId：ECS 需要部署到的区域

Password：ECS 的 Administrator 密码

DomainName：示例中 Domain 的名称

NewDomainNetbiosName：示例中的 NetbiosName

DomainAdminPassword：Domain 管理员用户密码

DomainAdminUser：Domain 用户名称

SPFarmAccount：SharePoint 服务场管理员的账户名称

SPFarmAccountPassword：SharePoint 服务场管理员的账户密码

SPISOImageURI：SharePoint 的镜像地址，请使用将 img 文件上传至 OSS 后获取的URL

InstanceType：ECS的规格。

d. 模版创建完成后，在启动资源栈前输入参数内容。输入完成后，单击创建 按钮。ROS会根据脚本以及输入的参数开始创建资源。

资源创建完成后，可以在资源栈管理列表当中看到相应的资源栈。

以上是整个执行流程中需要用到的参数文件。

3. UserData 执行

ROS 模板中 UserData 的执行分为五个过程，其中的四个过程是通过四个PS1文件执行完成的。各过程的描述如下：

a. 通过 UserData 的方式对机器进行设定，使其在启动时下载 SharePoint 镜像，并启用自动登录。
b. 安装 Domain 功能，并重启。
c. 在新安装的 Domain 下创建用户，并安装 Sharepoint 所需要的模块及 MSSQL，并重启。
d. 安装 SharePoint 服务并重启。
e. 配置 SharePoint 服务。

具体执行步骤如下：

a. 机器首次启动后，系统将会下载 OSS 中的 SharePoint 文件，同时修改 Windows 的注册表，启用系统免密码自动登录。此外，系统还会设定在下一次启动时运行 Domain 的安装脚本。

b. SharePoint 文件下载完成后，系统将会重启，并且开始执行 Domain 的安装脚本。该脚本会对用户在 ROS 中设定的 Domain 进行配置。并将下一步需要执行的脚本写入注册表，等待下一次启动时运行。

c. 创建用户并安装必要先决服务及软件。

A. 系统将会根据用户在ROS当中设定的`DomainAdminUser`和`SPFarmAccount`创建2个用户，分别用于管理 Domain 和 SP 服务器场。
B. 开始安装以下内容：.NET Framework Feature，‘Application Server’ role，‘Web Server’ role, WAS Feature, 及 Windows Identity Foundation Feature

C. 开始下载并安装 MS SQL SERVER EXPRESS 2012 版本（这里演示程序默认下载EXPRESS 版本的MSSQL）

```
C:\SQLEXPR_x64_ENU.exe" /Q /IACCEPTSQLSERVERLICENSETERMS /
ACTION=install  /ROLE=AllFeatures_WithDefaults  /INSTANCENAME=
$($instanceName) /SQLSVCACCOUNT="$($serviceAccount)
```

D. 关闭 IE ESC 选项，并离线下载和安装 SharePoint 2016 Preparation softer ware。因为某些原因，直接联网下载 Preparation softer ware 可能会失败，所以这里配置了脚本，使系统先下载文件，再进行安装。

d. 当所有的先决服务和软件都安装成功后，脚本自动开始运行 SharePoint 的安装程序。安装程序会首先检查先决服务和软件是否安装完成。

   如果所有的准备工作都已完成，系统开始正式安装 SharePoint。

e. SharePoint 安装完成后，系统将进行更新，并开始进入下一个阶段，进行 SharePoint 的配置。

   系统会在 MSSQL 中创建相应的数据库，并开放端口（9527）作为默认的管理端登录端口。

f. 配置过程完成后，我们可以打开http://localhost:9527/default.aspx进入 SharePoint 的管理站点。

   在管理员站点当中，我们可以创建自己的 Web application。

   在该网站集下，我们可以继续创建自己的站点。

   完成站点的创建后，我们就可以开始体验 SharePoint 上的功能了。

2 Java SDK 的 LiveChannel 常见操作

本文介绍 Java SDK 的 LiveChannel 常见操作，如创建 LiveChannel、列举 LiveChannel 及删除 LiveChannel 等。

说明:

本文示例由阿里云用户 bin 提供，仅供参考。

创建 LiveChannel

通过 RTMP 协议上传音视频数据前，必须先调用该接口创建一个 LiveChannel。调用 PutLiveChannel 接口会返回 RTMP 推流地址，以及对应的播放地址。

说明:

您可以使用返回的地址进行推流、播放，您还可以根据该 LiveChannel 的名称来发起相关的操作，如查询推流状态、查询推流记录、禁止推流等。

以下代码用于创建 LiveChannel：

```
public static void createLiveChannel() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String liveChannelName = "<yourLiveChannelName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);
        CreateLiveChannelRequest request = new CreateLiveChannelReq
uest(bucketName,
                liveChannelName, "desc", LiveChannelStatus.Enabled,
new LiveChannelTarget());
        CreateLiveChannelResult result = oss.createLiveChannel(request
);

        //获取推流地址。
        List<String> publishUrls = result.getPublishUrls();
        for (String item : publishUrls) {
            System.out.println(item);
        }

        //获取播放地址。
        List<String> playUrls = result.getPlayUrls();
        for (String item : playUrls) {
            System.out.println(item);
        }

        oss.shutdown();
```

```
    }
```

创建 LiveChannel 的更多详情，请参考PutLiveChannel。

列举 LiveChannel

以下代码用于列举指定的 LiveChannel：

```
public static void listLiveChannels() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        ListLiveChannelsRequest request = new ListLiveChannelsRequest(
bucketName);
        LiveChannelListing liveChannelListing = oss.listLiveChannels(
request);
        System.out.println(JSON.toJSONString(liveChannelListing));
        oss.shutdown();
    }
```

列举 LiveChannel 详情，请参考ListLiveChannel。

删除 LiveChannel

说明：

- 当有客户端正在向 LiveChannel 推流时，删除请求会失败。
- DeleteLiveChannel 接口只会删除 LiveChannel 本身，不会删除推流生成的文件。

以下代码用于删除指定的 LiveChannel：

```
public static void deleteLiveChannel() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);
        LiveChannelGenericRequest request = new LiveChannelGenericRe
quest(bucketName, liveChannelName);

        try {
            oss.deleteLiveChannel(request);
        } catch (OSSException ex) {
```

```
            ex.printStackTrace();
        } catch (ClientException ex) {
            ex.printStackTrace();
        } finally {
            oss.shutdown();
        }
    }
```

删除 LiveChannel 详情，请参考DeleteLiveChannel。

设置 LiveChannel 状态

LiveChannel 有 enabled 和 disabled 两种状态供您选择。

以下代码用于设置 LiveChannel 状态：

```
public static void setLiveChannelStatus() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String liveChannelName = "<yourLiveChannelName>";
        String bucketName = "<yourBucketName>";

        // 创建OSSClient实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        try {
           oss.setLiveChannelStatus(bucketName, liveChannelName,
LiveChannelStatus.Enabled);
        } catch (OSSException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } catch (ClientException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } finally {
            oss.shutdown();
        }
    }
```

设置 LiveChannel 状态更多详情，请参考PutLiveChannelStatus。

获取 LiveChannel 状态信息

以下代码用于获取指定 LiveChannel 的推流状态信息。

```
public static void getLiveChannelStat() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String liveChannelName = "<yourLiveChannelName>";
        String bucketName = "<yourBucketName>";
```

```
        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        LiveChannelStat liveChannelStat = oss.getLiveChannelStat(
bucketName, liveChannelName);
        System.out.println(liveChannelStat.toString());
        oss.shutdown();
    }
```

获取 LiveChannel 状态信息更多详情，请参考GetLiveChannelStat。

获取 LiveChannel 配置信息

以下代码用于获取指定 LiveChannel 的配置信息：

```
public static void getLiveChannelInfo() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        String liveChannelName = "<yourLiveChannelName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        LiveChannelInfo liveChannelInfo = oss.getLiveChannelInfo(
bucketName, liveChannelName);
        System.out.println(JSON.toJSONString(liveChannelInfo));
        oss.shutdown();
    }
```

获取 LiveChannel 配置信息更多详情，请参考GetLiveChannelInfo。

生成 LiveChannel 播放列表

PostVodPlaylist 接口用于为指定的 LiveChannel 生成一个点播用的播放列表。OSS 会查询指定时间范围内由该 LiveChannel 推流生成的 ts 文件，并将其拼装为一个 m3u8 播放列表。

以下代码用于生成 LiveChannel 播放列表：

```
public static void postVodPlaylist() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String liveChannelName = "<yourLiveChannelName>";
        String bucketName = "<yourBucketName>";
        String playListName = "<yourPlayListName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);
```

```
        long startTime = getUnixTimestamp("2019-06-27 23:00:00");
        long endTIme = getUnixTimestamp("2019-06-28 22:00:00");

        try {
            oss.generateVodPlaylist(bucketName, liveChannelName,
playListName, startTime, endTIme);
        } catch (OSSException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } catch (ClientException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } finally {
            oss.shutdown();
        }
    }

    private static long getUnixTimestamp(String time) {
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss
");
        try {
            Date date = format.parse(time);
            return date.getTime() / 1000;
        } catch (ParseException e) {
            e.printStackTrace();
            return 0;
        }
    }
```

生成 LiveChannel 播放列表更多详情，请参考PostVodPlaylist。

查看 LiveChannel 播放列表

以下代码用于查看指定 LiveChannel 推流生成的、且指定时间段内的播放列表：

```
public static void getVodPlaylist() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String liveChannelName = "<yourLiveChannelName>";
        String bucketName = "<yourBucketName>";

        // 创建 OSSClient 实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        long startTime = getUnixTimestamp("2019-06-27 23:00:00");
        long endTIme = getUnixTimestamp("2019-06-28 22:00:00");

        try {
            OSSObject ossObject = oss.getVodPlaylist(bucketName,
liveChannelName, startTime, endTIme);
            System.out.println(ossObject.toString());
        } catch (OSSException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } catch (ClientException ex) {
```

```
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } finally {
            oss.shutdown();
        }
    }

    private static long getUnixTimestamp(String time) {
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss
");
        try {
            Date date = format.parse(time);
            return date.getTime() / 1000;
        } catch (ParseException e) {
            e.printStackTrace();
            return 0;
        }
    }
```

查看 LiveChannel 播放列表更多详情，请参考GetVodPlaylist。

获取 LiveChannel 推流记录

GetLiveChannelHistory 接口用于获取指定 LiveChannel 的推流记录。使用 GetLiveChannelHistory 接口最多会返回指定 LiveChannel 最近的 10 次推流记录。

以下代码用于获取 LiveChannel 推流记录：

```
public void getLiveChannelHistory() {
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
        // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https
://ram.console.aliyun.com 创建 RAM 账号。
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        String liveChannelName = "<yourLiveChannelName>";

        // 创建OSSClient实例。
        OSS oss = new OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret);

        List<LiveRecord> list = oss.getLiveChannelHistory(bucketName,
liveChannelName);
        System.out.println(JSON.toJSONString(list));
        oss.shutdown();
    }
```

获取 LiveChannel 推流记录更多详情，请参考GetLiveChannelHistory。

3 使用 Java SDK 的 SelectObject 查询 CSV 和 Json 文件

本文介绍如何使用 Java SDK 的 SelectObject 查询 CSV 和 Json 文件。

说明:

本文示例由阿里云用户 bin 提供，仅供参考。

以下代码用于查询 CSV 文件和 Json 文件：

```
 private static final String csvKey = "test.csv";

 // Endpoint 以杭州为例，其它 Region 请按实际情况填写。
 private static final String endpoint = "http://oss-cn-hangzhou.
aliyuncs.com";
 // 阿里云主账号 AccessKey 拥有所有 API 的访问权限，风险很高。
 // 强烈建议您创建并使用 RAM 账号进行 API 访问或日常运维，请登录 https://ram.
console.aliyun.com 创建 RAM 账号。
 private static final String accessKeyId = "<yourAccessKeyId>";
 private static final String accessKeySecret = "<yourAccessKeySecret
>";
 private static final String bucketName = "<yourBucketName>";

    /**
    * CreateSelectObjectMeta API 用于获取目标文件总的行数，总的列数（对于 CSV
 文件），以及 Splits 个数。
    */
    public static void createCsvSelectObjectMetadata() {
      // 创建 OSSClient 实例。
      OSS client = new OSSClientBuilder().build(endpoint, accessKeyId
, accessKeySecret);

      String content = "name,school,company,age\r\n" +
           "Lora Francis,School A,Staples Inc,27\r\n" +
           "Eleanor Little,School B,\"Conectiv, Inc\",43\r\n" +
           "Rosie Hughes,School C,Western Gas Resources Inc,44\r\n" +
           "Lawrence Ross,School D,MetLife Inc.,24";

      client.putObject(bucketName, csvKey, new ByteArrayInputStream(
content.getBytes()));

       SelectObjectMetadata selectObjectMetadata = client.createSele
ctObjectMetadata(
            new CreateSelectObjectMetadataRequest(bucketName, csvKey)
               .withInputSerialization(
                    new InputSerialization().withCsvInputFormat(
                        new CSVFormat().withHeaderInfo(CSVFormat.
Header.Use)
                            .withRecordDelimiter("\r\n"))));
       //获取 csv 的总列数。
       System.out.println(selectObjectMetadata.getCsvObjectMetadata().
getTotalLines());
       //获取 csv 的 splits 个数。
       System.out.println(selectObjectMetadata.getCsvObjectMetadata().
getSplits());
```

```
        client.shutdown();
    }

    /**
    * 查询 csv。
    */
    public static void selectCsv() {
      // 创建 OSSClient 实例。
      OSS client = new OSSClientBuilder().build(endpoint, accessKeyId
, accessKeySecret);

      try {
         //保存 Select 请求的容器。
         SelectObjectRequest selectObjectRequest =
             new SelectObjectRequest(bucketName, csvKey)
                 .withInputSerialization(
                      new InputSerialization().withCsvInputFormat(
                          new CSVFormat().withHeaderInfo(CSVFormat.
Header.Use)
                              .withRecordDelimiter("\r\n")))
                 .withOutputSerialization(new OutputSerialization().
withCsvOutputFormat(new CSVFormat()));

         // ossobject 不可修改，查询第 4 列值大于 40 的数据，会直接进行隐式转
换。
         selectObjectRequest.setExpression("select * from ossobject
where _4 > 40");
         OSSObject ossObject = client.selectObject(selectObjectRequest
);

         writeToFile(ossObject.getObjectContent(), "result.csv");
        } catch (OSSException ex) {
          System.out.println(ex.getErrorCode().concat(",").concat(ex.
getErrorMessage()));
        } catch (ClientException ex) {
          System.out.println(ex.getErrorCode().concat(",").concat(ex.
getErrorMessage()));
        } finally {
           client.shutdown();
        }
    }

    /**
     * 查询简单 json。
     */
    public static void selectSimpleJson() {
       String key = "simple.json";
       // 创建 OSSClient 实例。
       OSS client = new OSSClientBuilder().build(endpoint, accessKeyId
, accessKeySecret);

       final String content = "{\n" +
             "\t\"name\": \"Lora Francis\",\n" +
             "\t\"age\": 27,\n" +
             "\t\"company\": \"Staples Inc\"\n" +
             "}\n" +
             "{\n" +
             "\t\"name\": \"Eleanor Little\",\n" +
             "\t\"age\": 43,\n" +
             "\t\"company\": \"Conectiv, Inc\"\n" +
             "}\n" +
             "{\n" +
             "\t\"name\": \"Rosie Hughes\",\n" +
             "\t\"age\": 44,\n" +
```

```
                "\t\"company\": \"Western Gas Resources Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Lawrence Ross\",\n" +
                "\t\"age\": 24,\n" +
                "\t\"company\": \"MetLife Inc.\"\n" +
                "}";

        try {
            client.putObject(bucketName, key, new ByteArrayInputStream
(content.getBytes()));
            SelectObjectRequest selectObjectRequest =
                 new SelectObjectRequest(bucketName, key)
                     .withInputSerialization(new InputSerialization()
                          .withCompressionType(CompressionType.NONE)
                          .withJsonInputFormat(new JsonFormat().
withJsonType(JsonType.LINES)))
                          .withOutputSerialization(new OutputSeri
alization()
                          .withCrcEnabled(true)
                          .withJsonOutputFormat(new JsonFormat()))
                     .withExpression("select * from ossobject as s
where s.age > 40");

            OSSObject ossObject = client.selectObject(selectObje
ctRequest);
            writeToFile(ossObject.getObjectContent(), "result.simple.
json");
        } catch (OSSException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } catch (ClientException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } finally {
            client.shutdown();
        }
    }

    /**
     * 查询复杂 json。
     */
    public static void selectComplexJson() {
        String key = "complex.json";
        // 创建 OSSClient 实例。
        OSS client = new OSSClientBuilder().build(endpoint, accessKeyI
d, accessKeySecret);

        String content = "{\n" +
             "  \"contacts\":[\n" +
             "{\n" +
             "  \"firstName\": \"John\",\n" +
             "  \"lastName\": \"Smith\",\n" +
             "  \"isAlive\": true,\n" +
             "  \"age\": 27,\n" +
             "  \"address\": {\n" +
             "    \"streetAddress\": \"21 2nd Street\",\n" +
             "    \"city\": \"New York\",\n" +
             "    \"state\": \"NY\",\n" +
             "    \"postalCode\": \"10021-3100\"\n" +
             "  },\n" +
             "  \"phoneNumbers\": [\n" +
             "    {\n" +
             "      \"type\": \"home\",\n" +
```

```
                "      \"number\": \"212 555-1234\"\n" +
                "    },\n" +
                "    {\n" +
                "      \"type\": \"office\",\n" +
                "      \"number\": \"646 555-4567\"\n" +
                "    },\n" +
                "    {\n" +
                "      \"type\": \"mobile\",\n" +
                "      \"number\": \"123 456-7890\"\n" +
                "    }\n" +
                "  ],\n" +
                "  \"children\": [],\n" +
                "  \"spouse\": null\n" +
                "}\n" +
                "]}";

        try {
            client.putObject(bucketName, key, new ByteArrayInputStream
(content.getBytes()));
            SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                    .withInputSerialization(new InputSerialization()
                        .withCompressionType(CompressionType.NONE)
                        .withJsonInputFormat(new JsonFormat().
withJsonType(JsonType.LINES)))
                    .withOutputSerialization(new OutputSerialization()
                        .withCrcEnabled(true)
                        .withJsonOutputFormat(new JsonFormat()))
                    //返回所有 age 是 27 的记录，表达式可以根据 json 对象结构
进行嵌套调用。
                    .withExpression("select * from ossobject.contacts
[*] s where s.age = 27");

            OSSObject ossObject = client.selectObject(selectObje
ctRequest);
            writeToFile(ossObject.getObjectContent(), "result.complex.
json");
        } catch (OSSException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } catch (ClientException ex) {
            System.out.println(ex.getErrorCode().concat(",").concat(ex
.getErrorMessage()));
        } finally {
            client.shutdown();
        }
    }

    /**
     * 写入文件。
     *
     * @param in
     * @param file
     */
    private static void writeToFile(InputStream in, String file) {
        try {
            BufferedOutputStream outputStream = new BufferedOu
tputStream(new FileOutputStream(file));
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
              outputStream.write(buffer, 0, bytesRead);
            }
            outputStream.close();
```

```
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
```

SelectObject 的更多详情，请参考SelectObject。

4 使用 Python SDK 的 SelectObject 查询 CSV 和 Json 文件

本文介绍如何使用 Python SDK 的 SelectObject 查询 CSV 和 Json 文件。

说明:

本文示例由阿里云用户 fralychen 提供，仅供参考。

上传 CSV 或 Json 格式文件

您可以根据业务需求，在 OSS 管理控制台将 CSV 或 Json 格式文件上传到 OSS bucket 中。如何将文件上传至 OSS bucket，请参考上传文件。

调用测试

通过 put_object 中的 key、contant 参数创建并上传了一个名为 python_select 的文件。

说明:

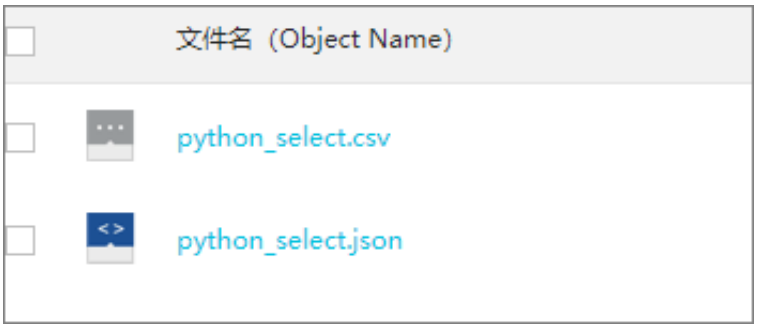
以下 Json 与 CSV 示例需分开执行。

```
import os
import oss2
from itertools import islice

# 首先初始化AccessKeyId、AccessKeySecret、Endpoint等信息。
# 通过环境变量获取，或者把诸如“<yourAccessKeyId>”替换成真实的AccessKeyId等。

# 以杭州区域为例，Endpoint可以是:
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com

access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<yourAccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<
yourAccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<yourBucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<yourEndpoint>')

# 创建存储空间实例，所有文件相关的方法都需要通过存储空间实例来调用。
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret),
endpoint, bucket_name)


### CSV示例

key =  'python_select.csv'
# 文件内容
content =  'fralychen,love,china'*10
filename =  'python_select.csv'
# 上传一个名为python_select.csv文件
bucket.put_object(key, content)
```

```
# 遍历文件上传后的文件目录。
for b in islice(oss2.ObjectIterator(bucket),10):
  print(b.key)


# 通过select_object使用sql语法查询文件。
# def select_object(self, key, sql,
#                   progress_callback=None,
#                   select_params=None,
#                   byte_range=None
#                   )

result = bucket.select_object(key, 'select * from ossobject')
print(result.read())


### JSON示例

key = 'python_select.json'
#content = "{contacts:
#         [
#             {
#             "key1":1,
#             "key2":"love china"
#             },
#             {
#             "key1":2,
#             "key2":"fralychen"
#             }
#         ]
#     }"
content = "{\"contacts\":[{\"key1\":1,\"key2\":\"love china\"},{\"key1
\":2,\"key2\":\"fralychen\"}]}"

# 上传一个名为python_select.json文件。
bucket.put_object(key,content)

result = bucket.select_object(key,'select * from ossobject')
print(result.read())
```

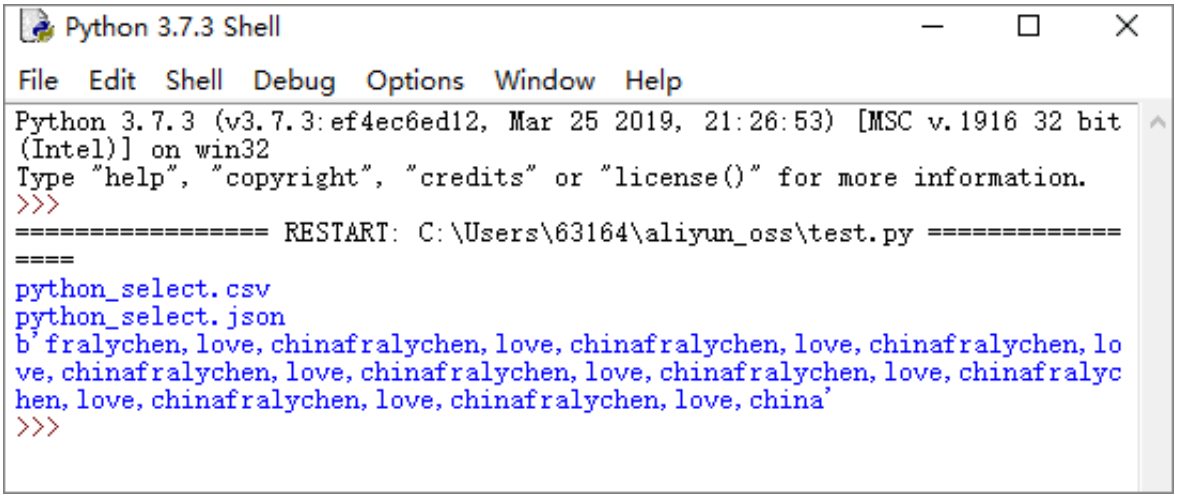
输出示例

您可以在OSS控制台上查看上传后的 python_select.csv 及 python_select.json 文件。

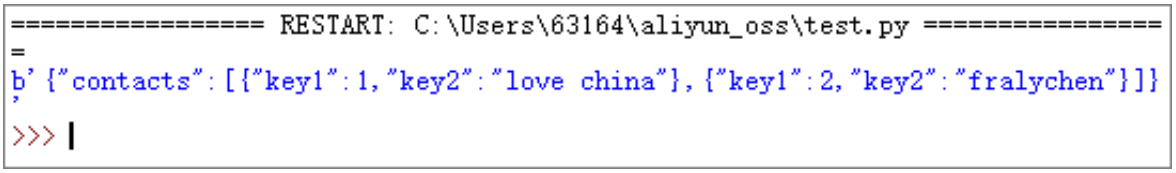


CSV 及 Json 的示例输出如下：

- CSV 示例输出



- Json 示例输出



常见 SQL 语句

常见的 SQL 应用场景及对应的 SQL 语句如下表所示：

应用场景	SQL 语句
返回前10行数据	select * from ossobject limit 10
返回第1列和第3列的整数，并且第1列大于第3列	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)
返回第1列以'陈'开头的记录的个数	select count(*) from ossobject where _1 like '陈%' 说明: 此处like之后的中文需要用UTF-8编码。
返回所有第2列时间大于2018-08-09 11:30:25且第3列大于200的记录	select * from ossobject where _2 > cast('2018-08-09 11:30:25' as timestamp) and _3 > 200
返回第2列浮点数的平均值、总和、最大值、和最小值	select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))
返回第1列和第3列连接的字符串中以'Tom'为开头以’Anderson‘结尾的所有记录	select * from ossobject where (_1 || _3) like 'Tom%Anderson'
返回第1列能被3整除的所有记录	select * from ossobject where (_1 % 3) = 0

应用场景	SQL 语句
返回第1列大小在1995到2012之间的所有记录	select * from ossobject where _1 between 1995 and 2012
返回第5列值为N、M、G、和L的所有记录	select * from ossobject where _5 in ('N', 'M', 'G', 'L')
返回第2列乘以第3列比第5列大100以上的所有记录	select * from ossobject where _2 * _3 > _5 + 100

更多参考

有关 Python SDK API 的更多信息，请参考GitHub。

有关 SelectObject 的更多信息，请参考SelectObject。

5 支付宝小程序直传实践

本文介绍如何在支付宝小程序环境下将文件（Object）上传到对象存储 OSS。

说明:

本文示例及视频素材均由阿里云用户 fralychen 提供，仅供参考。

背景

小程序是当下比较流行的移动应用，例如大家熟知的微信小程序、支付宝小程序等。它是一种全新的开发模式，无需下载和安装，因而可以被便捷地获取和传播，为终端用户提供更优的用户体验。如何在小程序环境下上传文件到 OSS 也成为开发者比较关心的一个问题。

与JavaScript客户端直传实践的原理相同，小程序上传文件到 OSS 也是利用 OSS 提供的 PostObject 接口来实现表单文件上传到 OSS。关于 PostObject 的详细介绍请参见 API 文档 PostObject。

操作视频

观看以下视频，快速了解如何在支付宝小程序中上传文件到OSS。

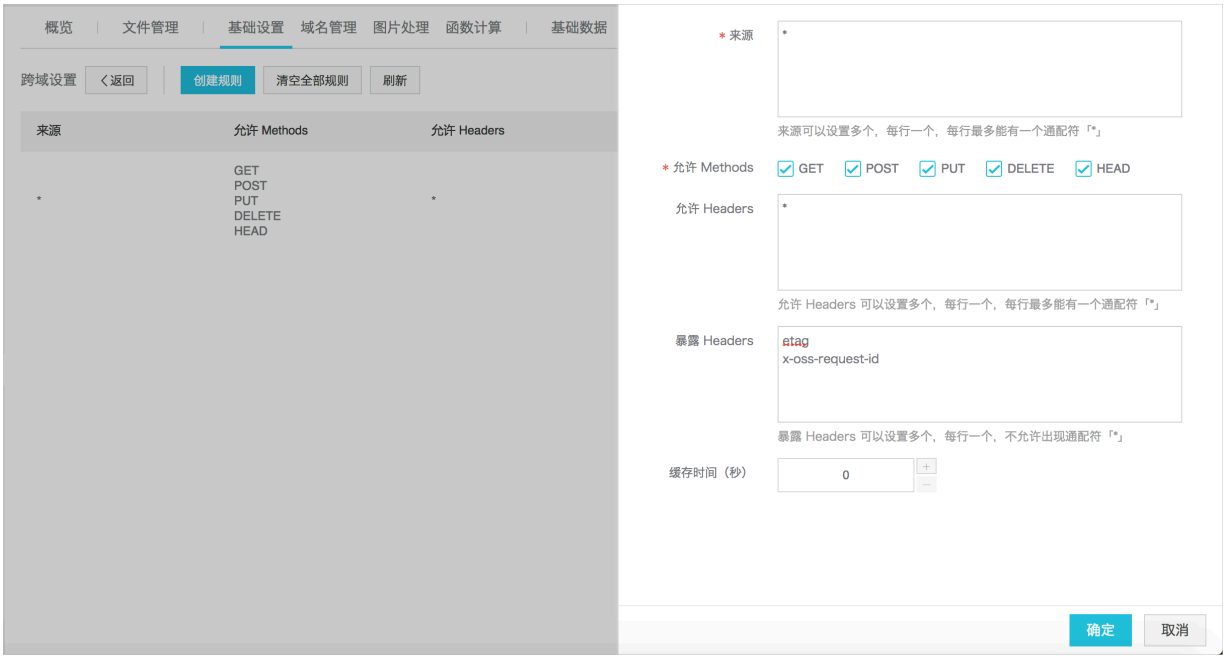
步骤 1：下载并安装支付宝小程序开发者工具

下载支付宝小程序开发者工具（以下简称 IDE）。基于您的操作系统选择相应的安装包，下载后根据安装指引，完成安装支付宝小程序开发者工具。

步骤 2：配置 Bucket 跨域

客户端进行表单直传到 OSS 时，会从浏览器向 OSS 发送带有 Origin 的请求消息。OSS 对带有 Origin 头的请求消息会进行跨域规则（CORS）的验证。因此需要为 Bucket 设置跨域规则以支持 Post 方法。

具体操作步骤请参见设置跨域访问。



步骤 3：使用 Web 端直传实践方案 Demo 进行上传测试

以下步骤展示了如何通过示例 demo 测试并获取上传需要的签名（signature）和加密策略（policy）：

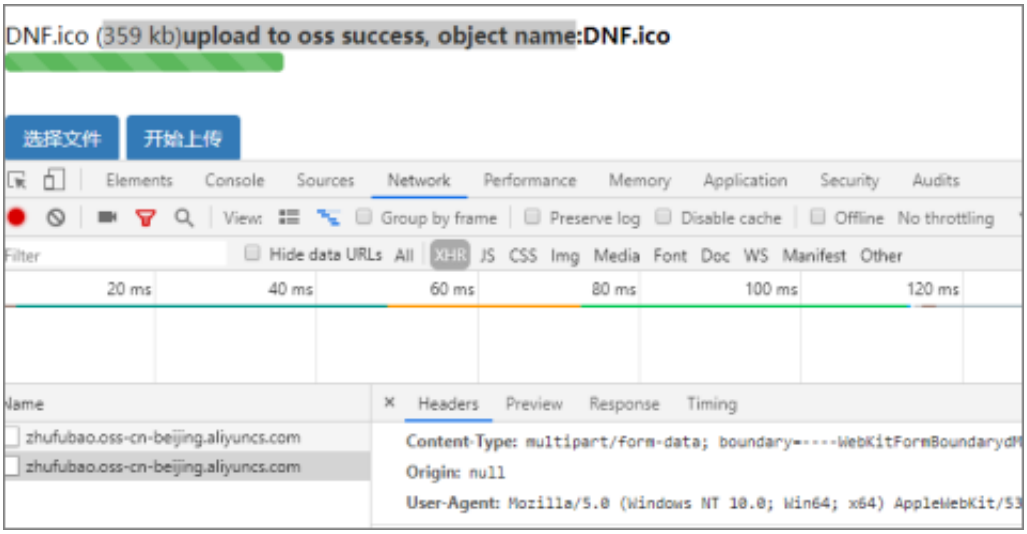
1. 下载应用服务器代码。
2. 修改 Demo 中 upload.js 的密钥和地址。

```
accessid= '6                    44';
accesskey= 'u                    lI0KLy';
host = 'http://zhufubao.oss-cn-beijing.aliyuncs.com';

g_dirname = ''
g_object_name = ''
g_object_name_type = ''
now = timestamp = Date.parse(new Date()) / 1000;

var policyText = {
    "expiration": "2020-01-01T12:00:00.000Z", //设置该Policy的失效时间，超过这个失效时间之后，就没有办法通过这个policy上传文件了
    "conditions": [
    ["content-length-range", 0, 1048576000] // 设置上传文件的大小限制
    ]
};

var policyBase64 = Base64.encode(JSON.stringify(policyText))
message = policyBase64
var bytes = Crypto.HMAC(Crypto.SHA1, message, accesskey, { asBytes: true }) ;
var signature = Crypto.util.bytesToBase64(bytes);
```

说明:

- 有关如何获取 accessid、accesskey 信息，请参考查看访问密钥基本信息。
- host 字段请填写 Bucket 域名（非 Endpoint）。

3. 打开 index.html 文件后，按 F12 开启 web 调试后进行文件上传，并获取请求中的 Form Data。



示例请求头如下：

```
## General
Request URL: http://zhufubao.oss-cn-beijing.aliyuncs.com/
Request Method: POST
Status Code: 200 OK
Remote Address: 59.110.190.29:80
Referrer Policy: no-referrer-when-downgrade

## Response Headers
Access-Control-Allow-Methods: GET, POST, PUT, HEAD, DELETE
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: etag, x-oss-request-id
Access-Control-Max-Age: 0
Connection: keep-alive
Content-Length: 0
Content-MD5: C1pYYZJo0Qqh/4YpcQtbbg==
Date: Thu, 25 Jul 2019 00:28:54 GMT
ETag: "0B5A58619268D10AA1FF8629710B****"
Server: AliyunOSS
x-oss-hash-crc64ecma: 9209453800795768232
x-oss-request-id: 5D38F7C55B40CCDBF4DB****
x-oss-server-time: 25

## Request Headers
Provisional headers are shown
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarygx
UvBqLDZHl9ZqTY
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36

## Form Data:
name: DNF.ico
key: ${filename}
policy: **==
OSSAccessKeyId: **
success_action_status: 200
signature: **
```

```
file: (binary)
```

步骤 4：创建并配置 uploadFile 项目

请按照如下步骤创建并配置 uploadFile 项目。

1. 创建项目

   在新建项目页面，填写项目名称及项目路径，后续服务勾选不启用云服务。

2. 编辑视图文件 /page/index/index.axml

```
<view> uploadFile </view>
<button onTap="uploadFile"> Click me! Upload File </button>
```

3. 编辑 /pages/index/index.js 文件

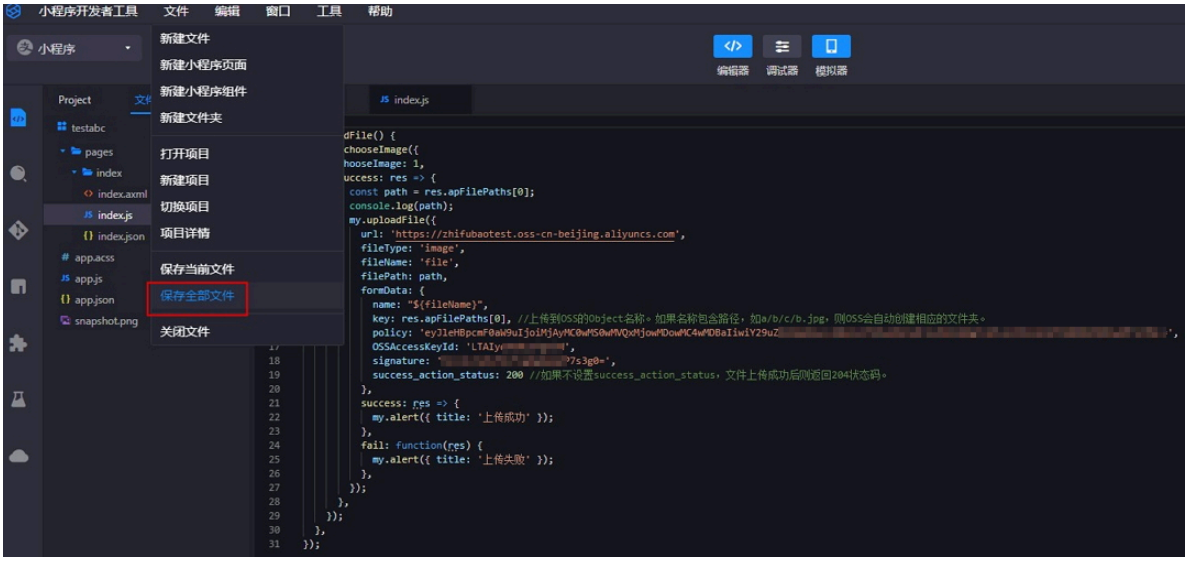
```
Page({
  uploadFile() {
    my.chooseImage({
      chooseImage: 1,
      success: res => {
        const path = res.apFilePaths[0];
        console.log(path);
        my.uploadFile({
          url: 'https://zhufubao.oss-cn-beijing.aliyuncs.com',
          fileType: 'image',
          fileName: 'file',
          filePath: path,
          formData: {
            name: "${fileName}",
            key: res.apFilePaths[0], //上传到OSS的Object名称。如果名称包
含路径，如a/b/c/b.jpg，则OSS会自动创建相应的文件夹。
            policy: '',
            OSSAccessKeyId: '',
            signature: '',
            success_action_status: 200 //如果不设置success_ac
tion_status，文件上传成功后则返回204状态码。
          },
          success: res => {
            my.alert({ title: '上传成功' });
          },
          fail: function(res) {
            my.alert({ title: '上传失败' });
          },
```

```
        });
      },
    });
  },
});
```

注意:

- 编辑 /pages/index/index.js 文件时请注意使用英文符号。
- 调试面板获取的 204 状态码表示文件已成功上传。
- 主页文件在 index.js，有关小程序的文件结构的更多信息，请参考详细了解小程序文件结构。
- 有关 uploadFile 的参数字段传递的更多信息，请参考my.uploadFile。

4. 单击文件 > 保存全部文件以保存以上编辑文件的内容。



5. 在右侧的支付宝 My App中单击Click me! Upload File，然后选择想要上传的文件。
6. 支付宝小程序中上传文件成功后，您可以到OSS管理控制台指定Bucket中查看已成功上传的文件。

常见问题

- web 直传 demo 报错

```
Error xml:<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>You have no right to access this object because of bucket
 acl.</Message>
  <RequestId>5D38EAF3B619A0AE40B6C2D9</RequestId>
  <HostId>post-test.oss-cn-hangzhou.aliyuncs.com</HostId>
```

```
</Error>
```

此问题的原因可能是 upload.js 文件的 host 字段值与实际不符，host 字段格式为 bucket.endpoint。

- 小程序中的 formData 能否不配置验证字段?

```
policy: '',
OSSAccessKeyId: '',
signature: ''
```

Bucket ACL 设置为公共读写（public-read-write）的情况下，小程序中的 formData 可以不配置验证字段，但必须进行 Bucket 跨域配置。