目录

数据迁移实践

KS3为用户提供多种数据迁移方式，同时也提供KS3之间的数据迁移、从第三方数据源迁移至KS3的完整方案。

已支持多种公有云厂商的对象存储迁移到金山云，如金山云、阿里云、百度云、七牛云、腾讯云、亚马逊云（中国）平台的对象存储。具有灵活、高效、安全的特点。

快速入门及计费说明参见 在线迁移服务。

迁移方式

本地数据迁移

本地小规模数据迁移可选择：

- 命令行工具
- 可视化工具

若数据规模较大，建议使用KS3-import数据迁移工具。

KS3之间数据迁移

可使用命令行工具、KS3-import数据迁移工具均可实现KS3之间的数据迁移，使用KS3的在线迁移服务可以实现：

- 同一region下，不同Bucket之间的数据迁移
- 不同region下，Bucket之间的数据迁移
- 不同KS3账号之间的数据迁移

KS3跨区域数据复制

可使用跨区域复制来实现不同Region之前的数据复制。

第三方数据源迁移至KS3

- KS3-import数据迁移工具目前支持将阿里云OSS、百度云、七牛云和AWS S3的数据迁移到KS3，修改配置文件的参数项即可轻松实现。 KS3-import数据迁移工具还支持增量上传、断点续传、流量控制、并发上传、进度查询和存储类型选择。
- KS3还为开发者提供支持各种语言的API和SDK，为用户应用提供更多的场景支持。

迁移方案

无缝数据迁移方案

当用户的服务已经在自己建立的源站或者在其他云产品上运行，需要迁移到KS3上，但是又不能停止服务，此时可利用重定向回源回源功能实现。

1. 将第三方存储T0之前的数据全量迁移至KS3
2. 数据迁移完成后，在KS3上配置重定向回源回源到第三方存储
3. 用户业务切换到KS3，记此时时间为T1
4. 用户将T0至T1时间段内新增或改动的文件，使用KS3-import或其他工具上传到KS3
5. 删除第三方存储

全量迁移方式

全量迁移有三种方式可供选择：

- 邮寄服务器：将服务器寄送到第三方存储，通过内网高速网络，下载数据到服务器后，回寄到KS3机房，将数据上传至KS3
- 互联网带宽：购买第三方存储的云服务器，通过内网高速网络下载数据的同时，利用互联网带宽上传到KS3
- 专线：购买第三方存储到KS3的专线，购买第三方存储或金山云的云服务器，通过内网高速网络下载数据的同时，利用专线上传到KS3

不使用主账号

如果用户担心主账户的AKSK泄露，最安全的策略是不使用主账号的AK/SK来访问KS3，而是创建一个子用户，并赋予这个子用户足够的权限，使用这个子用户的AK/SK这样可以规避AccessKey或者密码泄露导致的问题。具体操作步骤如下：

1. 进入控制台，点击左侧导航栏下方的**访问控制**，点击左侧导航栏的**人员管理** > **子用户**界面。
2. 点击子用户页面上方**新建用户**按钮，按照需求填写**用户登录信息**以及其他必填字段，勾选 **编程访问** （启用AccessKeyID和AccessKeySecret，支持通过API或其他开发工具访问）,**控制台密码登录**勾选后会生成子账号的账户密码信息。
3. 点击确定之后，生成该账号的AccessKey，一定要在这一步通过**复制**或者下载CSV文件 **保存** 下来AK/SK,用于后续的访问。

4. 返回**子用户**界面，找到新创建的账号。创建完成之后，该子账号还是没有任何权限，点击右边的 **添加权限** ，给该账号赋予KS3FullAccess系统权限，添加完毕之后点击**确定**按钮。
5. 使用刚刚创建的IAM子用户的AK/SK,生成签名信息，构造对应的访问URL，就可以访问KS3的相关资源了。
6. 进入子用户控制台登录界面，输入主账号ID或用户名，输入IAM用户名，输入密码，点击登录。

读写权限分离

当用户要使用应用服务器对外服务的时候，KS3可以作为后端静态资源的存储。这个时候应用服务器只获取KS3的读权限，不对KS3进行写入和设置。可以设置读写权限分离，给应用服务器一个只读权限的用户即可。

1. 进入控制台，点击进入左侧导航栏下方的 **访问控制**，再点击左侧导航栏中**人员管理** > **子用户** ，进入子用户界面。
2. 点击 **新建用户**按钮 ，在访问方式中勾选 **编程访问** 。
3. 生成该账号的AccessKey，一定要在这一步保存下来用于后续的访问。
4. 返回**子用户**界面，找到刚刚创建的子用户。创建完成之后，该子账号还是没有任何权限，点击右边的"添加权限"，给该账号赋予KS3ReadOnlyAccess系统权限。
5. 使用刚刚创建的IAM子用户的AK/SK,生成签名信息，构造对应的访问URL，就可以访问KS3的相关资源了。
6. 也可以使用SDK，在SDK配置文件中写入刚刚创建的IAM子用户的AK/SK，就可以访问KS3的相关资源了。

存储空间权限分离

如果客户有两个部门，分别为研发部门和运维部门，客户在KS3已经创建了两个存储空间，rd_bucket和op_bucket,分别存放研发数据和运维数据，要求研发部门只能访问rd_bucket，运维部门只能访问op_bucket。 这时就需要创建两个子用户rd_user和op_user，分别授予rd_bucket和op_bucket的权限。具体的操作步骤如下：

1. 进入控制台，点击左侧导航栏下方的**访问控制**，再点击左侧导航栏的**人员管理**>**子用户** ，进入子用户界面，创建rd_user和op_user，并生成对应的AKSK。
2. 进入控制台，点击左侧导航栏下方的**访问控制**，再点击左侧导航栏的**权限管理**>**策略** ，进入策略管理界面。
3. 由于系统权限没有按照bucket粒度的权限，点击 **自定义策略** 标签页，点击**新建策略** 按钮，在**设置策略类型**中选择**策略语法** ，在**选择策略模板**中选择**复制系统模板**，选择**KS3FullAccess** 模板，输入策略名称“rdfullaccess”,修改策略语言如下：

```
{
"Version": "2015-11-01",
"Statement": [
{
    "Effect": "Allow",
    "Action": "ks3:ListBuckets",
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "ks3:*",
    "Resource":[
        "krn:ksc:ks3:::rd_bucket",
        "krn:ksc:ks3:::rd_bucket/*"
    ]
 }
]
}
```

4. 同样，创建新策略“opfullaccess”。

```
{
"Version": "2015-11-01",
"Statement": [
{
    "Effect": "Allow",
    "Action": "ks3:ListBuckets",
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "ks3:*",
    "Resource":[
        "krn:ksc:ks3:::op_bucket",
        "krn:ksc:ks3:::op_bucket/*"
    ]
 }
]
}
```

5. 返回用户管理界面，给rd_user和op_user分别赋予rdfullaccess和opfullaccess策略。

业务权限分离

如果用户的应用服务器，不仅仅只是读取KS3的文件，还需要写入文件时，那么只把只读权限(KS3ReadOnlyAccess)授予应用服务是不够的，授予KS3FullAccess 风险又比较大（KS3FullAccess包含删除和设置权限），那么就需要按照业务来进行权限的分配，具体的操作步骤如下：

1. 进入控制台，点击进入左侧导航栏下方的**访问控制**，再点击左侧导航栏中的**权限管理** > **策略** ，进入策略管理界面。
2. 点击 **自定义策略** 标签页，点击**新建策略** 按钮，在**设置策略类型**中选择 **产品功能/项目权限** ，选择配置服务类型为 **对象存储**，按照实际需求在操作列选择需要开启的权限，选择完成之后点击**创建策略**。
3. 进入控制台，点击进入左侧导航栏下方的**访问控制**，再点击左侧导航栏中的**人员管理** > **子用户** ，进入用户管理界面，点击**添加权限**将应用服务器所对应的子用户授予新创建好的策略。

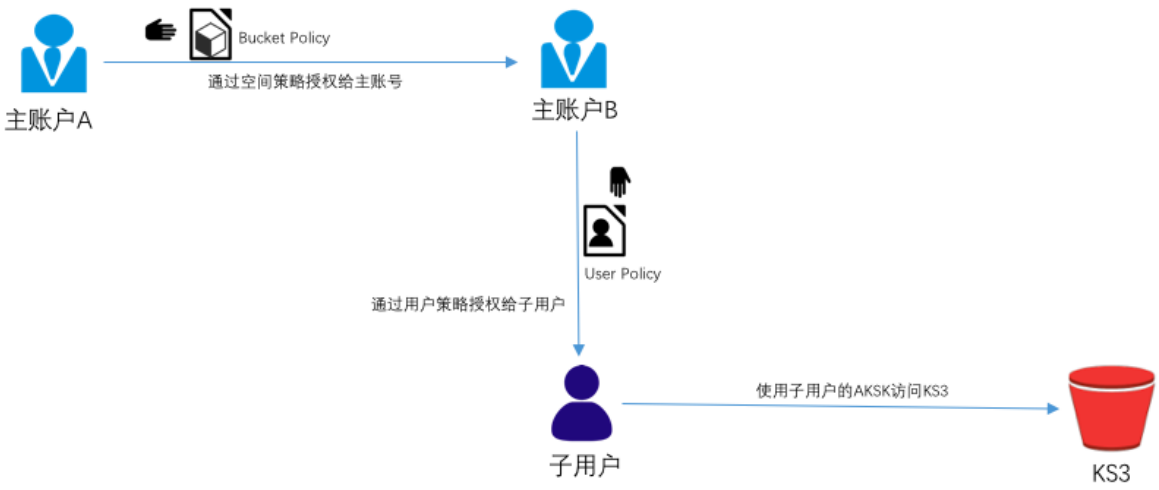
跨账户授权

假设主账户A想把自己的存储空间example_bucket的部分权限授权给主账号B下的研发人员。

那么要完成上述的功能，需要配置两条策略：

1. 需要将权限通过空间策略授予主账户B;
2. 主账户B再将这些权限通过用户策略委托给他的子用户rd_user。

原理图如下：



具体步骤如下：

1. 登录 KS3控制台，进入到A账户下需要授权的存储空间（Bucket），例如“exampleBucket”。点击**空间设置**页签 ，再点击右边**空间策略**页签，， 点击**添加策略**，用户填想要授权的账号B，操作选择ListBucket、GetObject。设置完成后点击确定。具体参考如下：

```
{
    "Version":"2015-11-01",
    "Statement": [
    {
        "Sid":"1",
        "Effect":"Allow",
        "Principal":{krn:ksc:iam::AccountB_ID:root},
        "Action":[
            "ks3:ListBucket",
            "ks3:GetObject"
         ],
        "Resource":["krn:ksc:ks3:::example_bucket",
"krn:ksc:ks3:::example_bucket/*"]
    }
  ]
}
```

2. 进入到B账户下的 **策略管理** 页面，点击**添加策略**按钮新建一条自定义策略，具体参考如下：

```
{
    "Version": "2015-11-01",
    "Statement": [
```

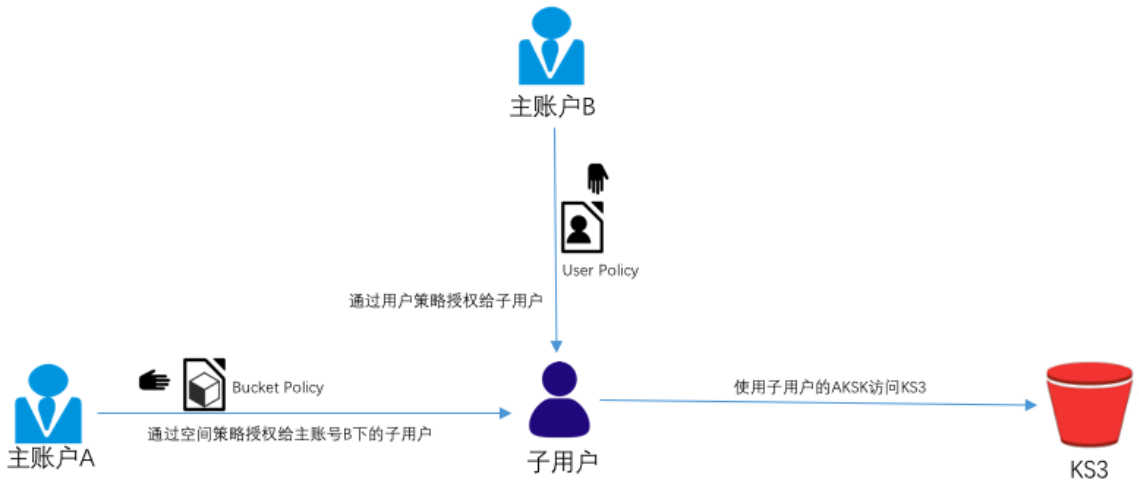
```
    {
        "Effect": "Allow",
        "Action": [
            "ks3:ListObject",
            "ks3:GetObject"
         ],
        "Resource":[
            "krn:ksc:ks3:::example_bucket",
            "krn:ksc:ks3:::example_bucket/*"
         ]
    }
  ]
}
```

3. 登录控制台，进入到B账户下的左侧导航栏下方的**访问控制** > **子用户** 页面，向rd_user授权第2步创建的策略。
4. 使用rd_user这个IAM子用户的AK/SK,生成签名信息，构造对应的访问URL，就可以访问example_bucket的相关资源了。

   注意 ：

   - 账户A还可以使用空间策略直接向账户B中的子用户授予权限。但是该子用户仍需要来自用户所属的。
   - 父账户(账户B)的权限，该子用户必须同时拥有来自资源拥有者和父账户的权限，便能够访问资源。

原理图如下：



临时授权访问

基本原理

无论是主账号还是IAM子用户都是可以长期正常使用的，对应的AK/SK发生泄露之后如果无法及时解除权限的话会很危险。如果众多的客户端需要直接与KS3交互，可以让客户端获取一个临时的、有一定的有效期的最小权限的凭证（即临时身份凭证），从而最大程度上保护用户数据安全。

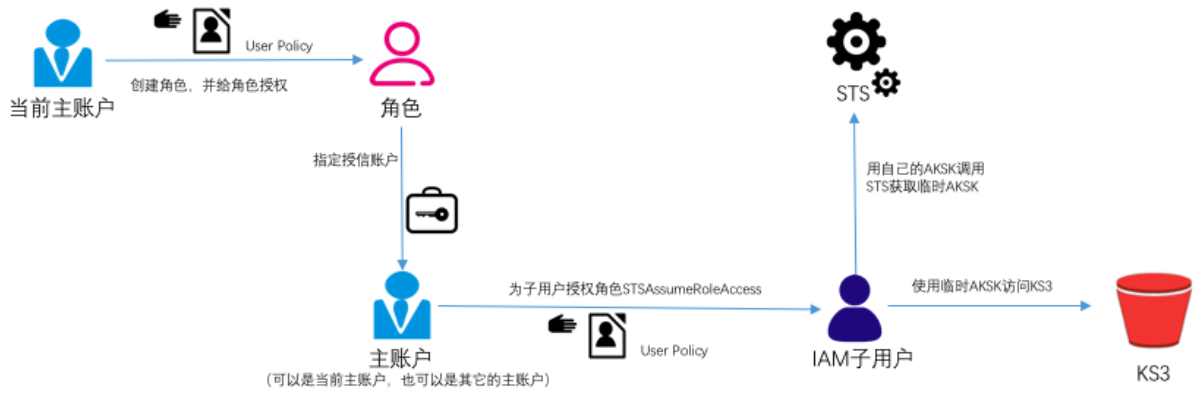
考虑到如下的案例：

客户开发的App会分发给终端用户，终端用户的数据需要直接上传到KS3，如果将主账号或者IAM子用户的AK/SK嵌入到App都是非常危险的行为，那如何才能安全的授权给众多的App用户上传数据呢，以及如何保证多个用户之间存储的隔离。

类似这种需要临时访问的场景可以使用临时身份凭证来应对，其主要操作步骤为：

- 在当前的主账号下创建一个角色，指定角色授信的主账户，并且给角色通过用户策略（user policy）授予一个最小权限。
- 然后在授信主账号下创建一个子用户，并授权给子用户角色（让这个子用户扮演这个角色），子用户就可以通过STS服务获取临时AK/SK及token，去访问KS3了，为了提升安全性该token还有一定的过期时间。

原理图如下：



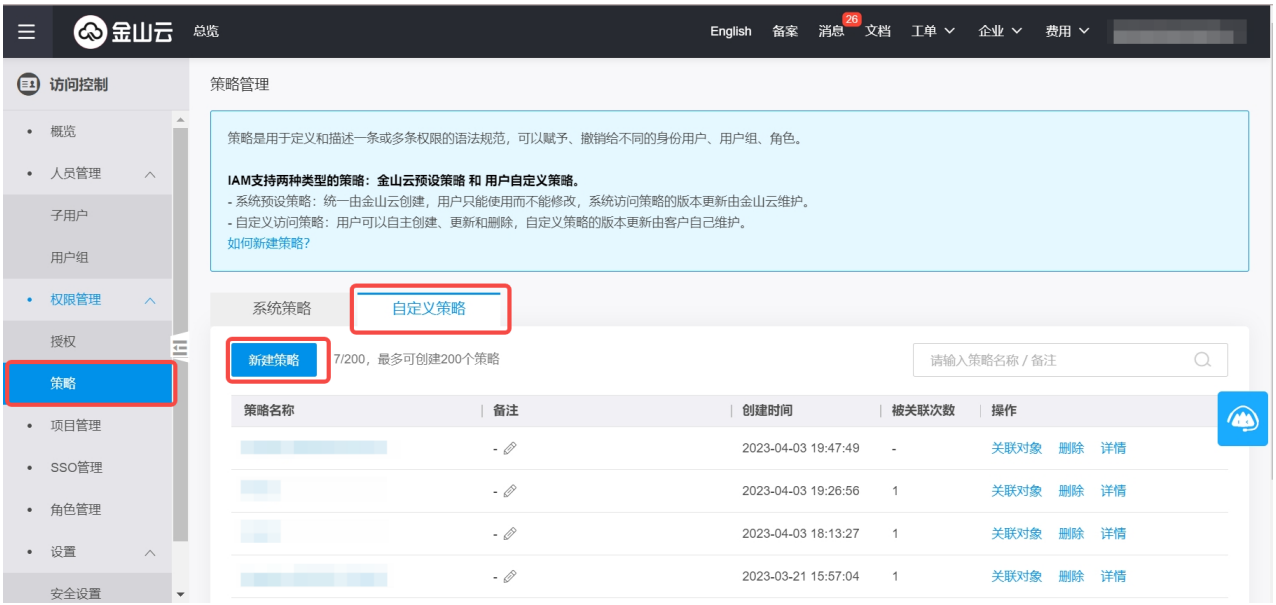
具体的步骤如下：

创建策略

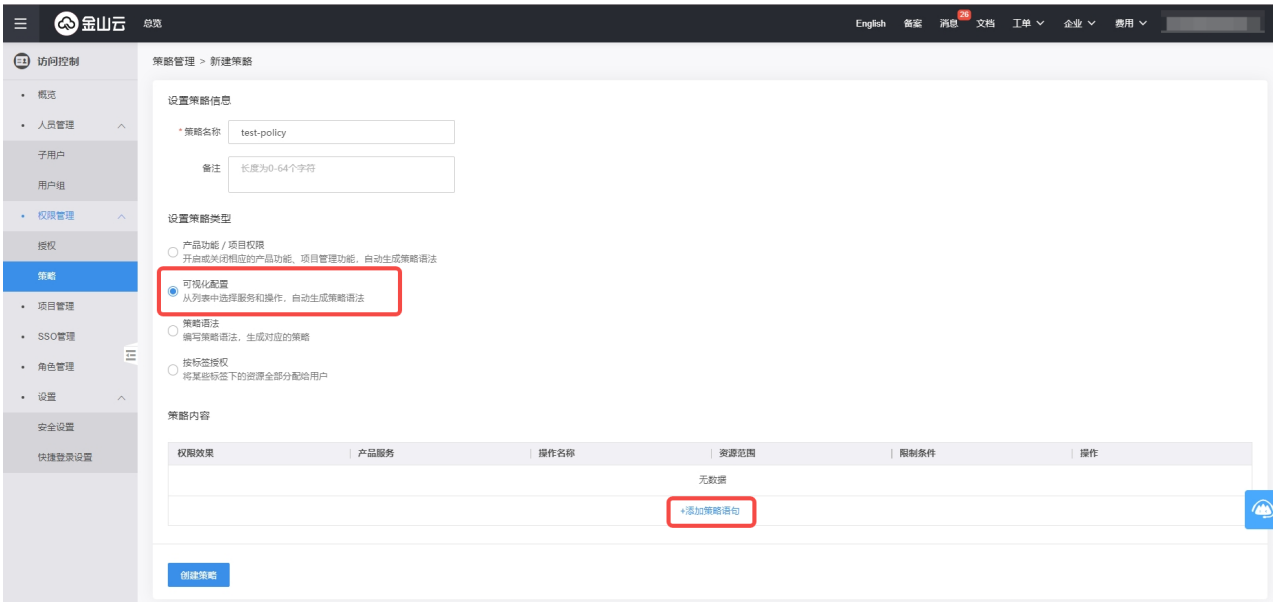
安全的STS服务，需要设置两个自定义策略：

- 子用户策略：保证子用户有扮演角色的权限
- 角色策略：保证角色对目标资源有对应权限

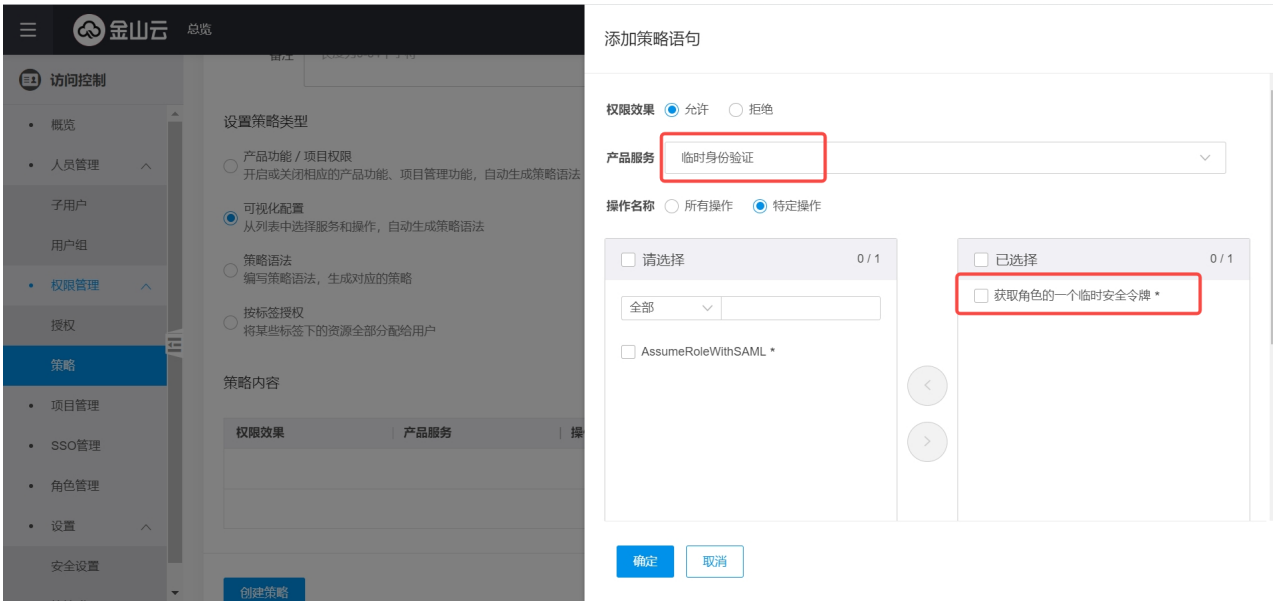
1. 进入KS3控制台，点击左侧导航栏最下方中的**访问控制**（或点击右上角账号名称下拉框里的**访问控制**），进入访问控制管理页面，点击左侧导航栏中**权限管理→策略**，进入策略管理页面，选择**自定义策略→新建策略**。



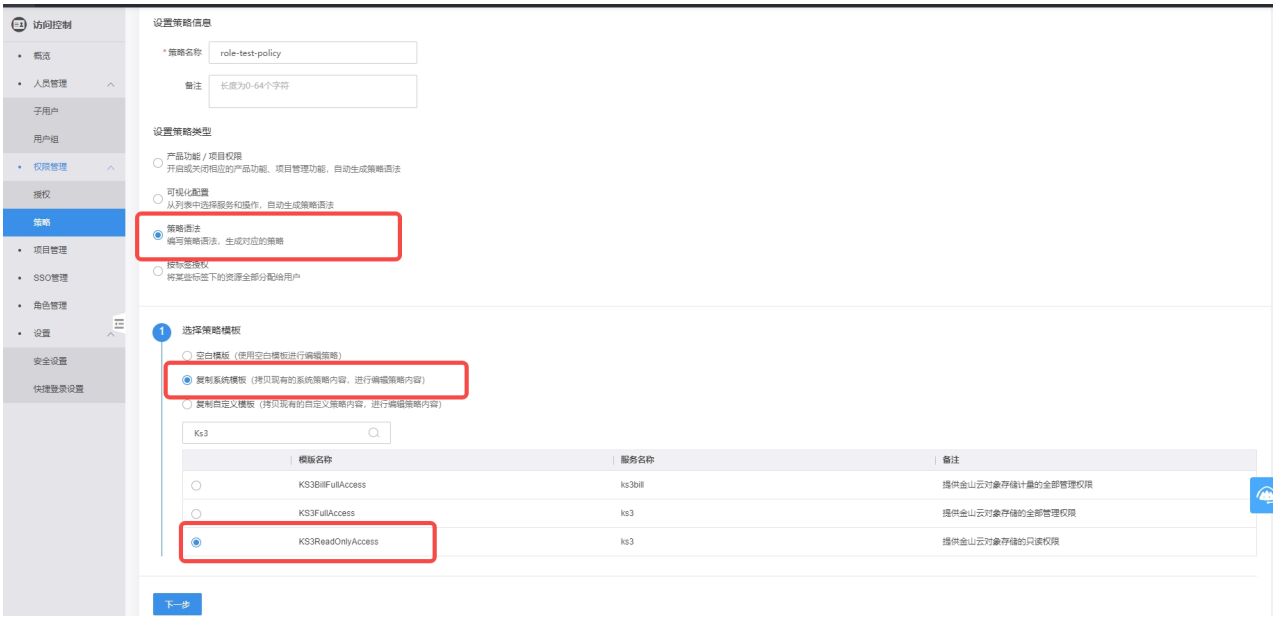
2. 在新建策略中，输入策略名，此处假设为"test-policy"，然后选择 **可视化配置→添加策略语句**。



3. 侧边弹出**添加策略语句**弹窗，产品服务下拉框中选择**临时身份验证**，在操作中选择**获取角色的一个临时安全令牌*，该权限表示可以扮演角色，点击**确定**按钮，之后点击左下方**创建策略**按钮，至此子用户策略创建完成。

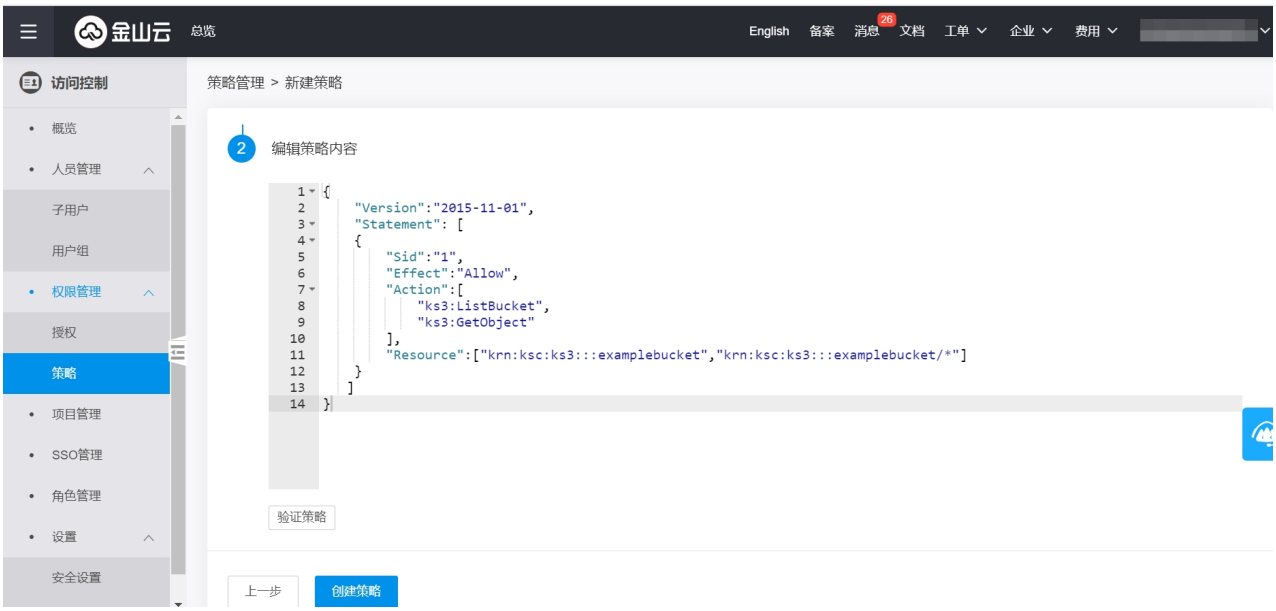


4. 再次创建自定义策略，假设名称为“role-test-policy”，设置策略类型为 **策略语法** ，可以通过模板来简化创建策略过程，选择**复制系统模板**，模板名称选择**KS3ReadOnlyAccess**，点击**下一步**。



- 也可以直接通过**可视化配置**来创建策略，产品服务选择**对象存储**，根据需求选择对应的操作权限即可。
- 如果需要**列出某个bucket下所有文件**的权限，需要同时设置**查询bucket列表**和**查询bucket ACL**的权限。
- 选择**可视化配置**时，可以对所有资源授予权限，也可以对特定资源授予权限（如某个bucket），填写特定资源信息方式参见文档：资源说明。

5. 编辑相应的权限，如下载文件、查看bucket列表等，编辑完成之后点击**创建策略**按钮，至此角色策略创建完成。



如下示例为对桶examplebucket及其内文件有get和ListBucket权限：

```
{
    "Version":"2015-11-01",
    "Statement": [
    {
        "Sid":"1",
        "Effect":"Allow",
        "Action":[
            "ks3:ListBucket",
            "ks3:GetObject"
        ],
        "Resource":["krn:ksc:ks3:::examplebucket","krn:ksc:ks3:::examplebucket/*"]
    }
   ]
}
```

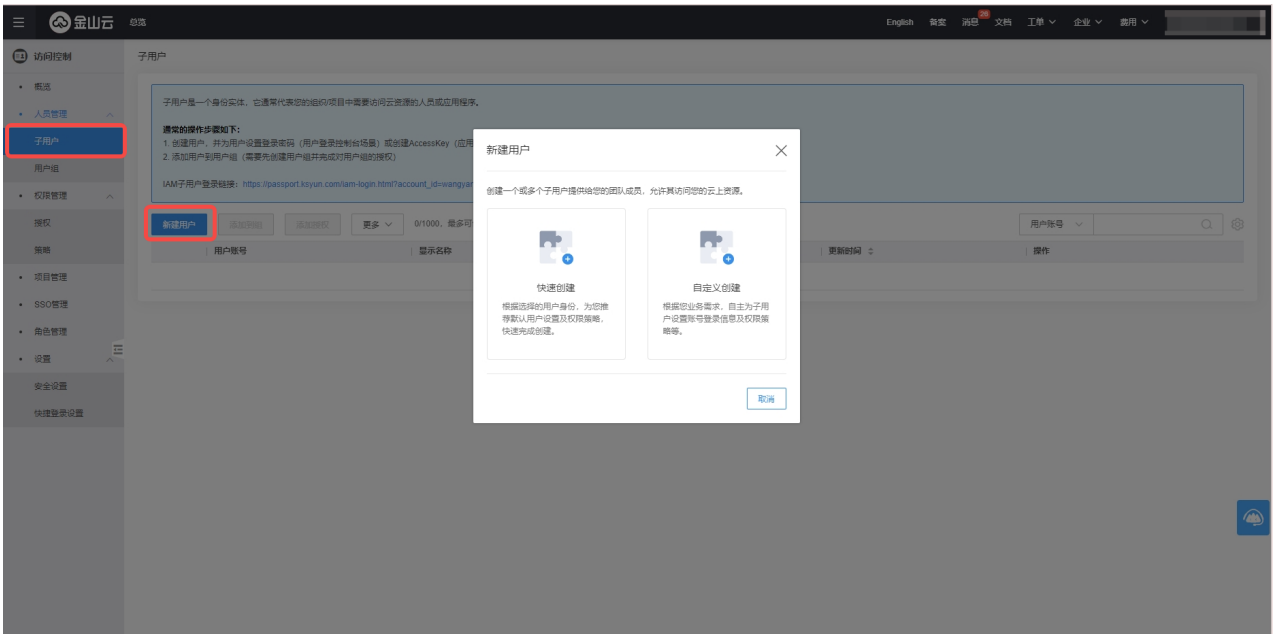
有关用户策略的更多详情请参见文档：用户策略。

创建子用户

主账户是不能扮演角色的，需要受信主账户通过用户权限（user policy）把扮演角色的权限授权给IAM子用户。

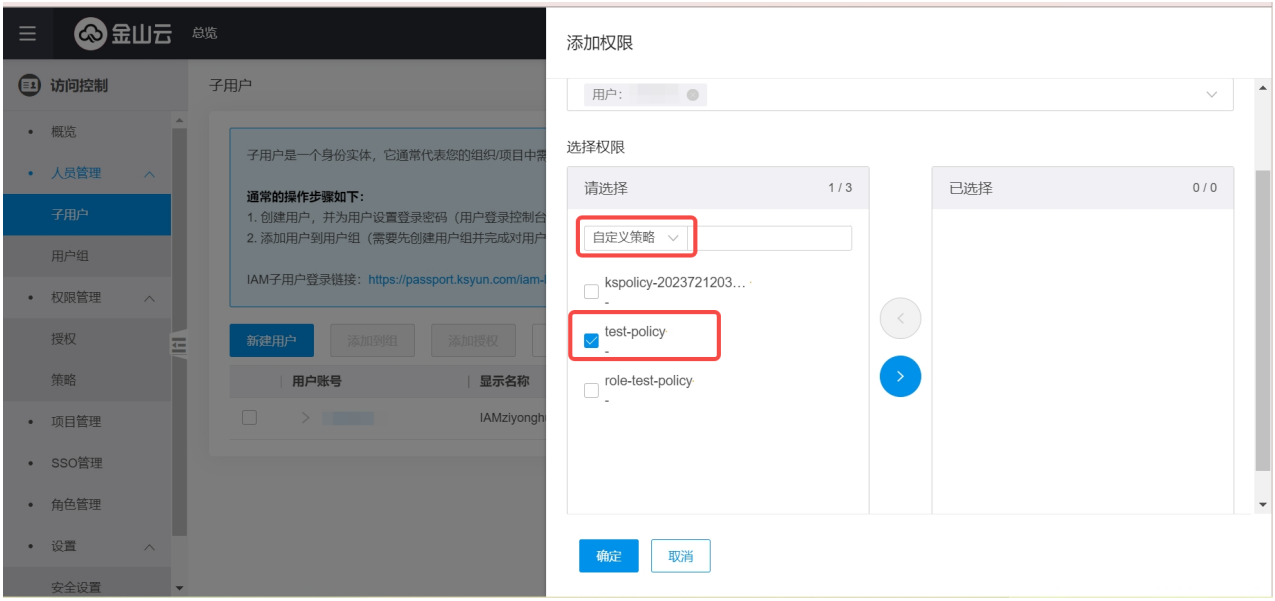
1. 进入KS3控制台，点击左侧导航栏下方的**访问控制**（或点击右上角账号名称下拉框里的**访问控制**），点击左侧导航栏中的**子用户** ，进

入子用户管理界面。点击**新建用户**按钮（根据需求选择**快速创建**或者**自定义创建**），输入子用户登录账户等信息，此处假设为test-subuser，选中**编程访问**复选框为子用户创建AK和SK。



2. 创建成功后在**子用户**页面选中刚创建的子用户，点击右侧**添加权限**来为子用户授权。

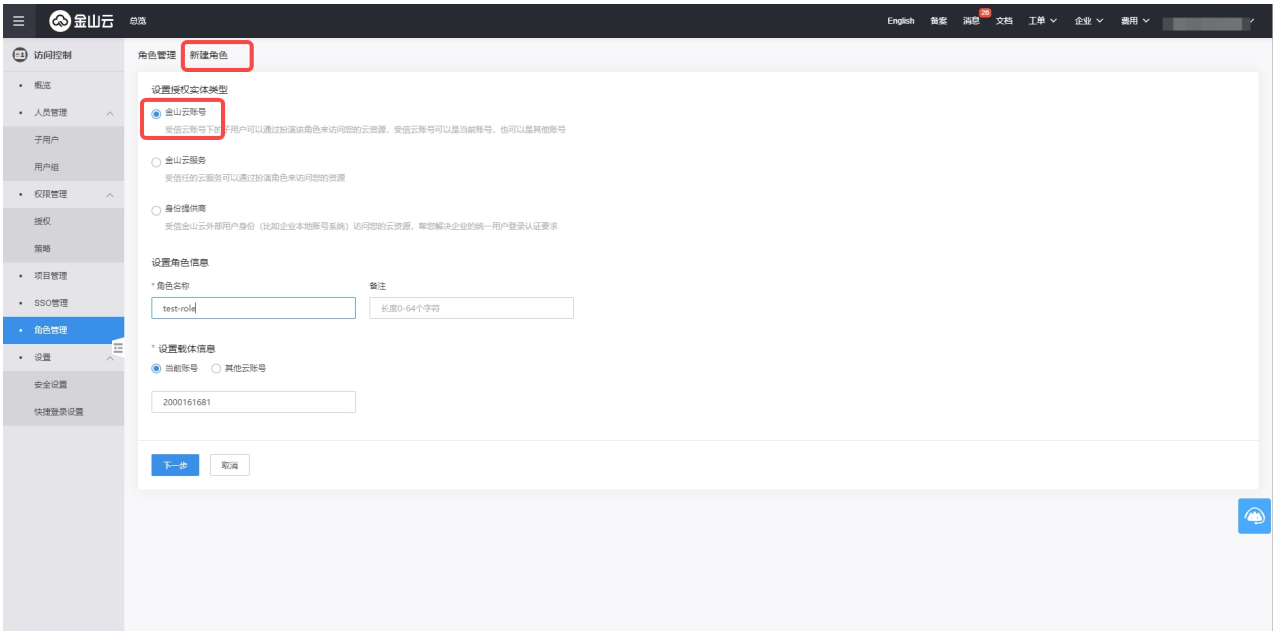
3. 在**选择权限**的下拉框中选择**自定义策略**，选中在**创建策略**步骤中创建的test-policy策略，这样子用户就具有了扮演角色的权限，但是现在还没有角色，因此接下来要创建角色。



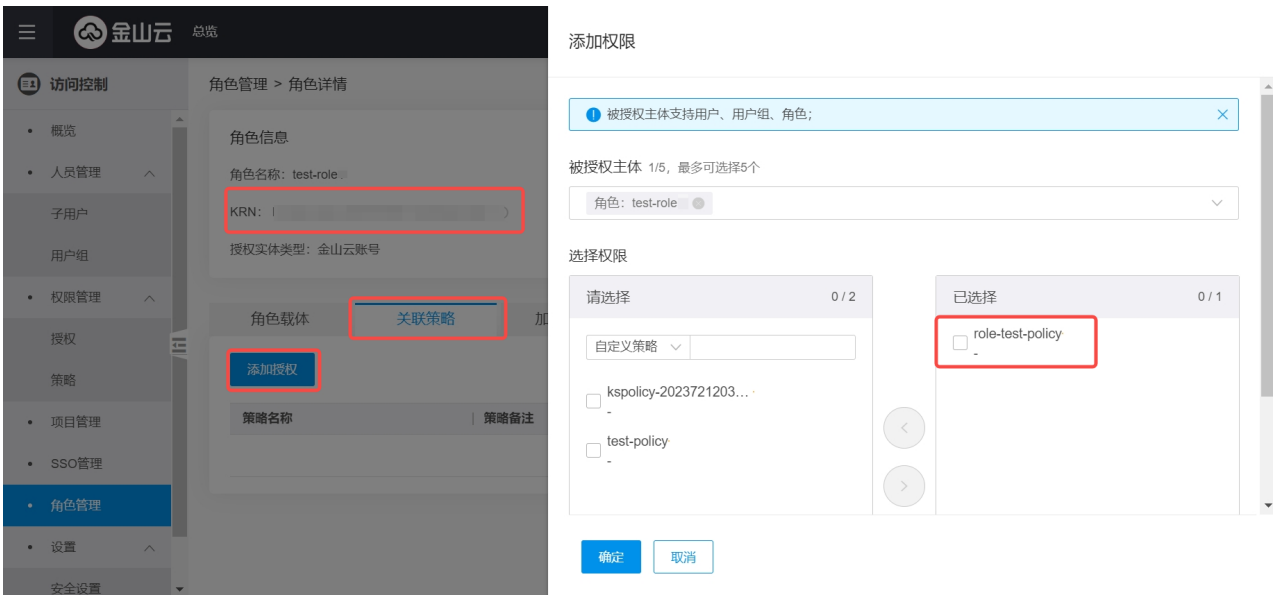
创建角色

1. 进入KS3控制台，点击左侧导航栏下方的**访问控制**（或点击右上角账号名称下拉框里的访问控制），点击左侧导航栏下方的**角色管理**，进入角色管理界面。

2. 点击**新建角色**按钮，选择**金山云账号**。假设创建一个名为test-role的角色，设置载体信息为当前账号（也可以选择其他账号，载体信息即授信账号，比如你在账号1下创建角色授信给账号2，则账号2的子用户就可以扮演账号1下的角色）。新创建的角色是没有任何权限的，点击**下一步**来设置角色权限，设置结束后点击**完成**按钮。

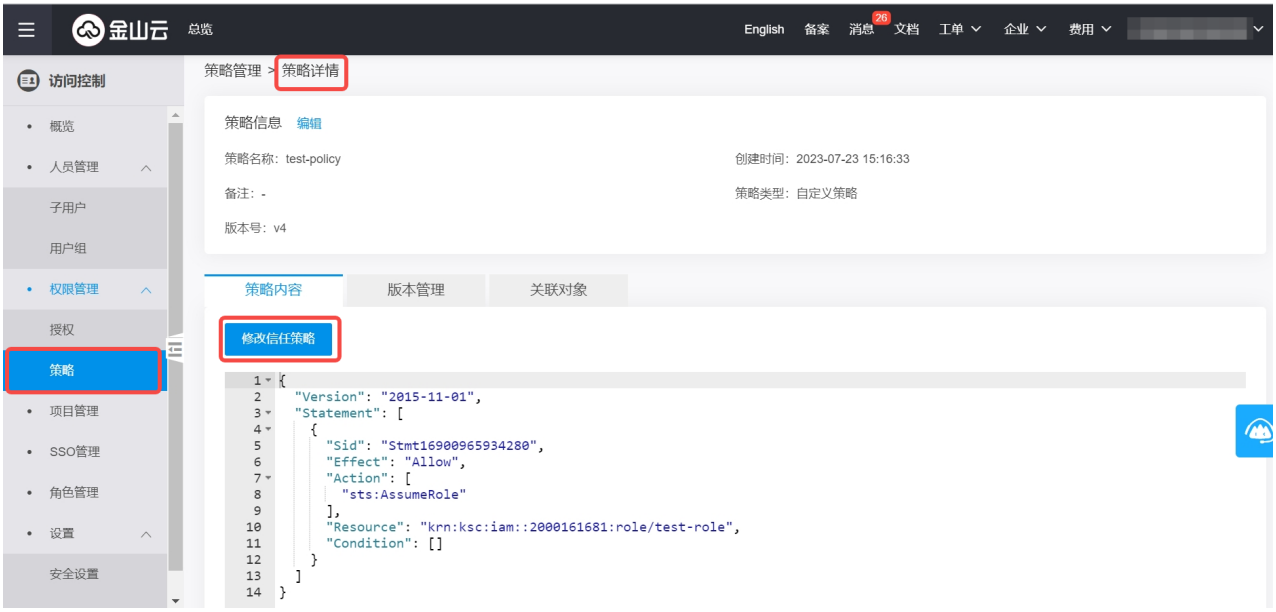


3. 创建完角色之后，回到**角色管理**页面，点击刚刚创建的角色名，进入角色详情页。需要记录下角色的**KRN字段**。角色是没有任何权限的，点击**关联策略**，点击**添加授权**按钮，弹出添加权限弹窗。在策略下拉框中选择自定义策略，选择策略role-test-policy，点击**确定**按钮。这样角色test-role就具有了相应的权限。



子用户扮演角色

至此该子用户仅具有了扮演角色的权限，但还没有关联具体的角色，因此还需要添加权限对应的资源，即角色ID。点击导航栏左侧**策略**页签，点击**自定义策略**，点击所创建的test-policy策略名称，在**策略内容**中的Resource字段值中填入角色的**KRN**。



使用STS授权访问

1. 使用子用户test-subuser的AK/SK 去调用STS服务获取临时权限。详见文档获取角色的临时身份。

```
http://sts.cn-beijing-6.api.ksyun.com/?Action=AssumeRole
&Version=2019-11-01
&RoleSessionName=Bob
&RoleKrn=krn:ksc:iam::Account_ID:role/test-role
&AUTHPARAMS
```

返回的内容如下：

```
<AssumeRoleResponse>
  <AssumeRoleResult>
      <Credentials>
        <SecretAccessKey>wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY</SecretAccessKey>
        <Expiration>2017-07-15T23:28:33.359Z</Expiration>
        <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
        <SecurityToken>V1xxxxxxxxxxxx</SecurityToken>
      </Credentials>
      <AssumedRoleUser>
         <Krn>krn:ksc:sts::123456789012:assumed-role/demo/Bob</Krn>
         <AssumedRoleId>ARO123EXAMPLE123:Bob</AssumedRoleId>
      </AssumedRoleUser>
   </AssumeRoleResult>
   <ResponseMetadata>
     <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
   </ResponseMetadata>
</AssumeRoleResponse>
```

2. 从返回的XML结果中找到临时权限：AccessKeyId、SecretAccessKey及SecurityToken，然后通过KS3 API或JAVA SDK去访问相应的资源。

GO示例：

```
package main

import (
    "bytes"
    "encoding/json"
    "fmt"
    "github.com/KscSDK/ksc-sdk-go/ksc"
    "github.com/KscSDK/ksc-sdk-go/ksc/utils"
    "github.com/KscSDK/ksc-sdk-go/service/sts"
    "github.com/ks3sdklib/aws-sdk-go/aws"
    "github.com/ks3sdklib/aws-sdk-go/aws/credentials"
    "github.com/ks3sdklib/aws-sdk-go/service/s3"
    "os"
)

func main() {
    ak := "<AccessKeyID>"
    sk := "<AccessKeySecret>"
    stsRegion := "cn-beijing-6"
    assumeRole := assumeRoleRequest(ak, sk, stsRegion) //调用ksc-go-sdk的sts服务
    var data Response
    _ = json.Unmarshal(assumeRole, &data)
    stsAk := data.AssumeRoleResult.Credentials.AccessKeyId      // sts ak
    stsSk := data.AssumeRoleResult.Credentials.SecretAccessKey  // sts sk
    stsToken := data.AssumeRoleResult.Credentials.SecurityToken // sts token
```

```
    //create ks3 client with sts ak,sk,token
    credentials := credentials.NewStaticCredentials(stsAk, stsSk, stsToken)
    client := s3.New(&aws.Config{
        Region:           "BEIJING",
        Credentials:      credentials,
        Endpoint:         "ks3-cn-beijing.ksyuncs.com", //ks3地址
        DisableSSL:       true,                         //是否禁用https
        LogLevel:         1,                            //是否开启日志,0为关闭日志, 1为开启日志 n
        S3ForcePathStyle: false,                        //是否强制使用path style方式访问
        LogHTTPBody:      true,                         //是否把HTTP请求body打入日志
        Logger:           os.Stdout,                    //打日志的位置
    })

    // example for uploading file
    params := &s3.PutObjectInput{
        Bucket:      aws.String("BucketName"),              // bucket名称
        Key:         aws.String("ObjectKey"),               // object key
        ACL:         aws.String("public-read"),             //权限, 支持private(私有), public-read(公开读)
        Body:        bytes.NewReader([]byte("PAYLOAD")),    //要上传的内容
        ContentType: aws.String("application/ocet-stream"), //设置content-type
        Metadata: map[string]*string{
            //"Key": aws.String("MetadataValue"), // 设置用户元数据
            // More values...
        },
    }
    resp, err := client.PutObject(params)
    if err != nil {
        panic(err)
    }
    fmt.Println(resp)
    //获取新的文件名
    fmt.Println(*resp.NewFileName)

}

func assumeRoleRequest(ak string, sk string, region string) []byte { //调用ksc-go-sdk的sts服务
    svc := sts.SdkNew(ksc.NewClient(ak, sk /*,true*/), &ksc.Config{Region: &region}, &utils.UrlInfo{ //debug模式的话,打开true开关
        UseSSL:      true,
        UseInternal: false,
    })
    var resp *map[string]interface{}
    var err error

    //set your assumeRole here
    assumeRoleInput := make(map[string]interface{})
    assumeRoleInput["RoleKrn"] = "<YourRoleKrn>"
    assumeRoleInput["RoleSessionName"] = "Bob"
    assumeRoleInput["DurationSeconds"] = "3600"
    //assumeRole["Policy"] = ""

    resp, err = svc.AssumeRole(&assumeRoleInput)
    if err != nil {
        fmt.Println("error:", err.Error())
        return nil
    }
    var str []byte
    if resp != nil {
        str, _ = json.Marshal(&resp)
        //fmt.Printf("%+v\n", string(str))
    }
    return str
}

type (
    Response struct {
        AssumeRoleResult struct {
            Credentials struct {
                SecretAccessKey string `json:"SecretAccessKey"`
                Expiration      string `json:"Expiration"`
                AccessKeyId     string `json:"AccessKeyId"`
                SecurityToken   string `json:"SecurityToken"`
            } `json:"Credentials"`
            AssumedRoleUser struct {
                Krn           string `json:"Krn"`
                AssumedRoleId string `json:"AssumedRoleId"`
            } `json:"AssumedRoleUser"`
        } `json:AssumeRoleResult`
        RequestId string `json:"RequestId"`
    }
)
```

跨域资源访问实践

使用场景

浏览器的同源策略限制从一个源加载的文档或脚本与来自另一个源的资源进行交互的方式，是用于隔离潜在恶意文件的关键安全机制。同协议、同域名（或IP）、以及同端口视为同一个域，一个域内的脚本仅仅具有本域内的权限，即本域脚本只能读写本域内的资源，而无法

访问其它域的资源。

跨域资源共享（Cross-Origin Resource Sharing，简称 CORS），是 HTML5 提供的标准跨域解决方案。CORS允许WEB端的应用程序访问不属于本域的资源。开发者可以利用KS3提供的接口控制跨域访问的各种权限，开发灵活的WEB应用程序。

使用规则

如果需要使用浏览器跨域访问资源，可以登录 KS3控制台，设置存储桶的相关CORS解决跨域访问问题。

控制台

登录KS3控制台,选择需要进行CORS配置的bucket,选择**空间设置** > **CORS配置** > **添加规则**，添加一条自定义CORS策略。

配置

您可以配置CORS的以下各项参数

参数设置说明：

- Allow Origin

  设定允许跨境请求的来源，多个域名以英文逗号分隔。

- Method

  设定允许的跨域请求方法，包含：GET、POST、DELETE、PUT、HEAD

- Allow Header

  设定允许的跨域请求header，多个Header以英文逗号分隔。

- Exposed Header

  设定允许从应用程序进行访问的响应头部。

- Cache Time

  设定浏览器对特定资源的预取（OPTIONS）请求返回结果的缓存时间。

  注： KS3会根据跨域请求匹配相对应的bucket下的CORS规则，根据规则设定的权限进行检查，并依次匹配每一条规则，根据规则的设定来允许请求并返回相对应的header，如想了解CORS相关操作请参考CORS配置。

防盗链实践

KS3提供的防盗链通过黑白名单的方式控制，其中黑名单用于添加禁止访问的来源域名，白名单用于添加允许访问的来源域名。防盗链功能可手动设置关闭。

录入域名的时候需要注意以下规则：

- 域名之间用逗号(,)分开，不需要写 http://
- 支持域名前使用通配符 *，可用于指代当前域名下的多级子域名

由于有些合法的请求是不会带referer来源头部的,所以有时候不能拒绝来源头部（referer）为空的请求，在KS3中，我们还提供了手动设置referer的选项。

操作步骤：

1. 登录 KS3控制台,进入想要设置的空间里，防盗链功能位于**空间设置**菜单下

2. 设置黑白名单：
   - 黑名单设置：

   - 白名单设置：

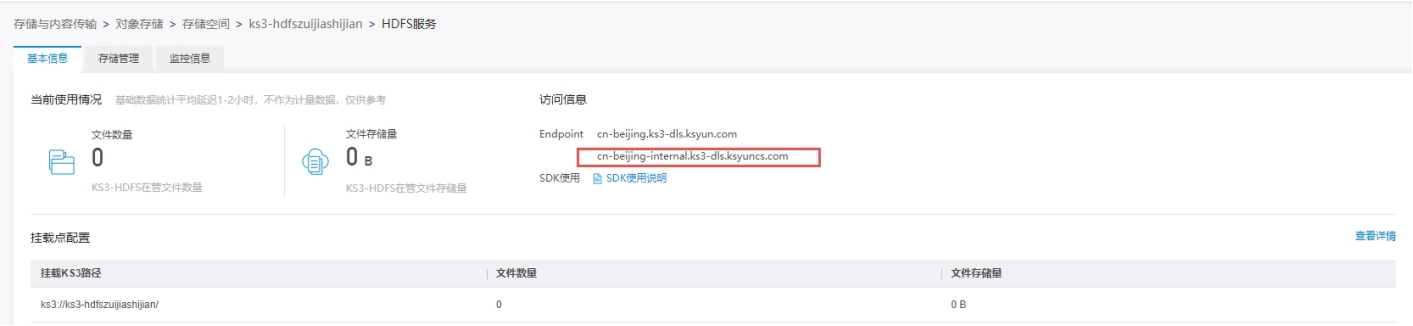
   

3. 手动设置referer

KS3-HDFS实践

环境准备

1. 登录KS3控制台，联系客服开通KS3-HDFS服务。
2. 记录KS3-HDFS的Bucket域名。



3. 大数据集群安装KS3-HDFS对应的Jar包。【Jar包地址请工单联系获取】

```
rz -bye
```

注：如果之前有使用hadoop-KS3直连客户，需要先将hadoop-ks3的Jar包删除。

4. 获取KMR ip并将Jar包拷贝到每个节点下。

```
awk '/clusterid/{print $NF}' /etc/hosts > kmrip
pscp.pssh -h kmrip alluxio-shaded-client-2.9.0.2-SNAPSHOT.jar /root
```

注：以上命令中，clusterid为变量值，填写对应的集群id。

5. 将Jar包移动至当前节点的hadoop/lib目录下。

```
mv alluxio-shaded-client-2.9.0.2-SNAPSHOT.jar /mnt/kmr/hadoop/1/hadoop-3.1.1/lib/
```

注：Jar包需要上传到集群下每个节点的hadoop/lib目录下。

6. 修改以下文件内容，增加相关KS3配置。

- core-site.xml【更改后需要重启Hdfs和Hive服务生效】

```
alluxio.user.client.base.filesystem.class=alluxio.client.file.Ks3HdfsMetaFileSystem
fs.ks3.impl=alluxio.hadoop.Ks3HdfsFileSystem
alluxio.master.rpc.addresses=ks3 hdfs endpoint地址
alluxio.ks3.hdfs.mode=true
```

注：

- 用户需填写对应的ks3-hdfs endpoint，示例：cn-beijing-internal.ks3-dls.ksyuncs.com。
- 如使用金山云KMR产品，可通过金山云控制台进行配置操作，注意所有的配置范围均须配置。

- mapred-site.xml 【更改后需要重启MapReduce】

```
mapred.output.committer.class=org.apache.hadoop.mapred.FileOutputCommitter
mapreduce.outputcommitter.factory.scheme.ks3=org.apache.hadoop.mapreduce.lib.output.PathOutputCommitterFactory
```

- spark-defaults 【更改后需要重启Spark】

```
spark.driver.extraClassPath=/mnt/kmr/hadoop/1/hadoop-3.1.1/lib
spark.executor.extraClassPath=/mnt/kmr/hadoop/1/hadoop-3.1.1/lib
spark.sql.parquet.output.committer.class=org.apache.parquet.hadoop.ParquetOutputCommitter
```

- hbase-site.xml 【更改后需要重启Hbase服务】

```
hbase.rootdir=ks3://${bucket-name}/hbase  #将bucket name替换为开通ks3-hdfs的桶名称
hbase.wal.dir=hdfs://hdfs-ha/hbase
hbase.bulkload.staging.dir=/tmp/hbase-staging
```

7. 验证。

```
hdfs dfs -ls ks3://${bucket}/
```

注：以上命令用于验证是否配置成功，如果list出Bucket下目录/文件时，代表配置成功。

8. 切换Hadoop身份。

```
su - hadoop
```

任务执行

以下将介绍KMR Hadoop、Hive或Spark如何操作KS3-HDFS。

相关信息

- 访问路径： ks3://${bucket}/path
- KS3-HDFS服务会通过您的角色授权访问KS3数据源进行所需操作

Hadoop 对接KS3

本示例为您展示如何在KS3-HDFS上进行hadoop wordcount作业。

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar wordcount ks3://${bucket}/input-path ks3://${bucket}/output-path
```

> 注：根据实际情况填写对应的输入地址（input-path）和输出地址（output-path），输出地址对应的文件名需要在KS3 Bucket中不存在。

Spark 对接KS3

本示例为您展示，Spark如何读取KS3中数据，并将处理完的数据写回至KS3。

```
import org.apache.spark.{SparkConf, SparkContext}
val conf = new SparkConf().setAppName("Test ks3")
val sc = new SparkContext(conf)
val pathIn = "ks3://${bucket}/path/*/*"
val inputData = sc.textFile(pathIn)
val cnt = inputData.count
println(s"count: $cnt")
val outputPath = "ks3://${bucket}/path"
val outpuData = inputData.map(e => s"$e has been processed.")
outpuData.saveAsTextFile(outputPath)
```

PySpark接入KS3示例

本示例为您展示，PySpark如何访问KS3中数据，并将处理完的数据写回至KS3。

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Python Spark SQL ks3 example").getOrCreate()
pathIn = "ks3://${bucket}/path/to/read"
df = spark.read.text(pathIn)
cnt = df.count()
print(cnt)
outputPath = "ks3://${bucket}/path/to/write"
df.write.format("parquet").mode('overwrite').save(outputPath)
```

Hive 对接KS3

以下将介绍如何在KMR集群中开发Hive作业流程。

在Hive中使用KS3

修改完core-site.xml配置文件后，需要重启Hive服务生效。 在Hive中读写KS3时，先创建一个external的表。

```
CREATE EXTERNAL TABLE eusers (
  userid INT)
 LOCATION 'ks3://${bucket}/path';
```

> 注：数据和表必须对应，否则无法查出数据

使用示例

Hive作业流程示例如下：

- 示例1

a. 编写如下脚本，保存为hiveSample1.sql文件。

```
CREATE EXTERNAL TABLE kmrusers (
 userid INT,
 movieid INT,
 rating INT,
 unixtime STRING )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION 'ks3://${Bucket}/users';##此处为重点，将数据放在KS3上
```

```
select * from kmrusers;
```

测试数据资源可下载如下数据集，并将其上传至KS3对应的目录。 资源下载：点击下载

b. 创建并运行作业。

```
hive -f sqlfilelocation
```

注：

- ${bucket}是KS3 Bucket，path是Bucket上的路径，需要您填写实际保存Hive脚本的位置。
- 您可以关联一个已有的集群，也可以自动按需创建一个，然后关联上创建的作业。

- 示例2：以HiBench中的scan为例。

a. 编写如下脚本，上传至KS3。

```
USE DEFAULT;
set hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat;
set mapreduce.job.maps=12;
set mapreduce.job.reduces=6;
set hive.stats.autogather=false;
DROP TABLE uservisits;
CREATE EXTERNAL TABLE uservisits (sourceIP STRING,destURL STRING,visitDate STRING,adRevenue DOUBLE,userAgent STRING,countryCode STRING,languageCode STRING,searchWord STRING,duration INT ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS SEQUENCEFILE LOCATION 'ks3://${bucket}/path';
```

注：其中${bucket}是您的KS3 Bucket，path是Bucket上的路径，需要您填写实际保存uservisits的位置。

可以下载作业需要的资源，并将其上传至您KS3对应的目录。 测试数据 ：点击下载

b. 在KMR中创建Hive作业，并运行。

```
CREATE EXTERNAL TABLE kmrusers (
 userid INT,
 movieid INT,
 rating INT,
 unixtime STRING )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION 'ks3://ks3-hdfs-test-2/users';
```

Hbase 对接KS3

利用KS3-HDFS将Hbase数据存储到KS3，以此摆脱对HDFS本地盘的依赖，降低存储成本。Hbase写模型使用LSMT，可靠写依赖对wal文件append，同时对性能要求高，因此将wal放到HDFS。

开通KS3-HDFS服务，请参见KS3-HDFS服务开通文档。

可能出现的问题

报错信息：

- master.HMaster: hbase:meta,,1.1588230740 is NOT online; state={1588230740 state=OPENING,

解决方法：

- 关闭Hbase服务（仅在刚部署时使用该操作，已经作为生产的请慎用）。
- 登录zookeeper，删除zk中Hbase node。
- 重启Hbase即可。

测试

1. hbase pe --rows=5000000 --oneCon=true --presplit=10 --nomapred randomWrite 10

```
2023-05-16 11:01:33,505 INFO  [TestClient-2] hbase.PerformanceEvaluation: Mean      = 484.29
Min       = 1.00
Max       = 9333906.00
StdDev    = 31403.17
50th      = 2.00
75th      = 2.00
95th      = 7.00
99th      = 11.00
99.9th    = 98.00
99.99th   = 1681804.71
99.999th  = 3823417.57
2023-05-16 11:01:33,505 INFO  [TestClient-2] hbase.PerformanceEvaluation: No valueSize statistics available
2023-05-16 11:01:33,505 INFO  [TestClient-2] hbase.PerformanceEvaluation: Finished class org.apache.hadoop.hbase.PerformanceEvaluation$RandomWriteTest in 1983275ms at offset 10000000 for 5000000 rows (2.48 MB/s)
2023-05-16 11:01:33,505 INFO  [TestClient-2] hbase.PerformanceEvaluation: Finished TestClient-2 in 1983275ms over 5000000 rows
2023-05-16 11:01:33,505 INFO  [main] hbase.PerformanceEvaluation: [RandomWriteTest] Summary of timings (ms): [1975841, 1962506, 1983275, 1977824, 1983030, 1966597, 1974498, 1965165, 1977491, 1974590]
2023-05-16 11:01:33,594 INFO  [main] hbase.PerformanceEvaluation: [RandomWriteTest duration ]        Min: 1962506ms        Max: 1983275ms
```

```
    Avg: 1974081ms
2023-05-16 11:01:33,594 INFO  [main] hbase.PerformanceEvaluation: [ Avg latency (us)]        407
2023-05-16 11:01:33,594 INFO  [main] hbase.PerformanceEvaluation: [ Avg TPS/QPS]        25329         row per second
```

2. hbase pe --rows=500000 --oneCon=true --presplit=10 --nomapred sequentialWrite 10

```
2023-05-16 14:04:09,300 INFO  [TestClient-9] hbase.PerformanceEvaluation: No valueSize statistics available
2023-05-16 14:04:09,300 INFO  [TestClient-9] hbase.PerformanceEvaluation: Finished class org.apache.hadoop.hbase.PerformanceEvaluation$Sequent
ialWriteTest in 191485ms at offset 4500000 for 500000 rows (2.57 MB/s)
2023-05-16 14:04:09,300 INFO  [TestClient-9] hbase.PerformanceEvaluation: Finished TestClient-9 in 191485ms over 500000 rows
2023-05-16 14:04:09,300 INFO  [main] hbase.PerformanceEvaluation: [SequentialWriteTest] Summary of timings (ms): [151740, 155557, 151781, 158
284, 191424, 152046, 191416, 191612, 158239, 191485]
2023-05-16 14:04:09,392 INFO  [main] hbase.PerformanceEvaluation: [SequentialWriteTest duration ]        Min: 151740ms        Max: 191612ms
      Avg: 169358ms
2023-05-16 14:04:09,392 INFO  [main] hbase.PerformanceEvaluation: [ Avg latency (us)]        337
2023-05-16 14:04:09,392 INFO  [main] hbase.PerformanceEvaluation: [ Avg TPS/QPS]        29855         row per second
2023-05-16 14:04:09,392 INFO  [main] client.AsyncConnectionImpl: Connection has been closed by main.
```

3. hbase pe --rows=500000 --oneCon=true --multiGet=10 --nomapred randomRead 20

```
查询QPS约670
```

关于KS3-HDFS更多详细信息请参考文档：KS3-HDFS服务概述。

Goofys

简介

Goofys是一个开源的使用Go编写的存储桶挂载工具，允许Linux和macOS挂载KS3存储桶在本地文件系统，即可像访问本地文件系统一样访问KS3存储桶，Goofys能够保持对象原来的格式。

前期准备

- 创建金山云账号，开通金山云对象存储KS3服务。
- 获取当前账号的AccessKey和SecretKey信息。
- 了解KS3的Endpoint信息。

安装

方式一：使用Go语言安装

主要面向开发人员安装，详细步骤如下：

安装FUSE

1. 执行以下命令下载安装包：

```
sudo yum install automake gcc-c++ git libcurl-devel libxml2-devel fuse-devel make openssl-devel
```

2. 执行以下命令安装FUSE：

```
sudo yum install fuse
```

下载Go语言

Goofys采用Go语言编写，所以依赖Go的运行环境。

1. 执行以下命令下载Go版本包：go1.15.linux-amd64.tar.gz

```
wget https://golang.google.cn/dl/go1.15.linux-amd64.tar.gz
```

2. 执行以下命令解压安装包：

```
tar -C /usr/local -xzf go1.15.linux-amd64.tar.gz
```

3. 执行以下命令创建Go的工作目录：

```
mkdir /opt/gowork
```

4. 设置Go的环境变量配置，首先执行以下命令进入/etc/profile：

```
vim /etc/profile //打开文件
a //进入插入模式
```

5. 执行以下命令配置环境变量参数：

```
export GOPATH=/opt/gowork
export PATH=$PATH:/usr/local/go/bin
// 按【Esc】键退出插入模式
:wq // 保存并退出
```

6. 执行以下命令使得配置生效：

```
source /etc/profile
```

7. 执行以下命令，可以查看Go的版本：

```
go version
```

输出示例如下：

```
go version go1.15 linux/amd64
```

安装Goofys

1. 执行以下命令设置Go的工作目录：

```
export GOPATH=$HOME/work
```

2. 执行以下命令去拉取Goofys：

```
go get github.com/kahing/goofys
```

注意：由于网络原因，此时可能会报错，需要VPN才可下载GitHub的仓库。

3. 执行以下命令安装Goofys：

```
go install github.com/kahing/goofys
```

方式二：直接下载安装

主要面向非开发人员安装，详细安装步骤如下：

1. 执行以下命令直接下载编译好的Goofys：

```
wget https://github.com/kahing/goofys/releases/latest/download/goofys
```

2. 执行以下命令赋予执行权限：

```
chmod +x goofys
```

3. 执行以下命令输出Goofys的版本：

```
./goofys -version
```

输出示例如下：

```
goofys version 0.24.0-45b8d78375af1b24604439d2e60c567654bcdf88
```

配置

- **挂载**

1. 执行以下命令创建.aws目录：

```
mkdir .aws
```

2. 执行以下命令进入.aws目录：

```
cd .aws
```

3. 执行以下命令创建文件credentials：

```
vi credentials
```

文件配置信息如下：

```
[default]
aws_access_key_id = AK
aws_secret_access_key = SK
```

配置项说明：

- aws_access_key_id：输入访问密钥信息AccessKey，可在金山云控制台的访问控制页面获取。
- aws_secret_access_key：输入访问密钥信息SecretKey，可在金山云控制台的访问控制页面获取。

4. 执行以下命令进入Goofys目录：

```
cd goofys
```

5. 执行以下命令为Goofys设置权限：

```
chmod +x goofys
```

6. 执行以下命令挂载存储桶：

```
goofys --subdomain --endpoint=https://[KS3 ENDPOINT] [挂载存储桶名称] [挂载目录]
```

示例：

```
goofys --subdomain --endpoint=https://ks3-cn-beijing.ksyuncs.com test-auto-bucket /mnt/  //将test-auto-bucket存储桶挂载到/mnt/目录下
```

7. 执行以下命令查看挂载结果：

```
df -h
```

注：挂载结果显示的Bucket容量为虚拟数值，KS3 Bucket实际无容量限制。

挂载成功显示结果如下所示：

```
[root@vm10-0-0-175 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  456K  1.9G   1% /run
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/vda1        50G  4.3G   43G  10% /
tmpfs           374M     0  374M   0% /run/user/0
goofystest      1.0P     0  1.0P   0% /goofys_test
```

8. 执行以下命令进入/mnt/目录，即可查看桶内文件：

```
cd /mnt/
```

- **卸载**

1. 执行以下命令卸载存储桶：

```
umount auto-test-bucket
```

2. 查看是否卸载成功：

```
df -h
```

下图所示表明已卸载成功：

```
[root@vm10-0-0-175 .aws]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  456K  1.9G   1% /run
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/vda1        50G  4.3G   43G  10% /
tmpfs           374M     0  374M   0% /run/user/0
```

常见问题

- 查看工具日志

输入以下命令查看工具日志：

```
vi /var/log/syslog
```

- 挂载调试模式

输入以下命令挂载调试：

```
goofys --subdomain --endpoint=https://[KS3 ENDPOINT] [挂载存储桶名称] [挂载目录] --debug_s3 --debug_fuse
```

S3fs

简介

S3fs是基于FUSE的文件系统，允许Linux和macOS挂载KS3存储桶在本地文件系统，即可像访问本地文件系统一样访问KS3存储桶，S3fs能够保持对象原来的格式。

前期准备

- 创建金山云账号，开通金山云对象存储KS3服务。
- 获取当前账号的AccessKey和SecretKey。
- 了解KS3的Endpoint信息。

安装

安装依赖包

- CentOS安装命令：

```
sudo yum install automake fuse fuse-devel gcc-c++ git libcurl-devel libxml2-devel make openssl-devel
```

- Ubuntu安装命令：

```
sudo apt-get install build-essential git libfuse-dev libcurl4-openssl-dev libxml2-dev mime-support automake libtool
sudo apt-get install pkg-config libssl-dev
```

安装S3fs

1. 从源码安装S3fs：

```
git clone https://github.com/s3fs-fuse/s3fs-fuse.git   //下载s3fs源码
cd s3fs-fuse
./autogen.sh    //执行autogen.sh脚本
./configure     //执行configure脚本
make   //编译
sudo make install
```

2. 检查S3fs是否安装成功：

```
s3fs --version
```

输出S3fs版本信息表明安装成功：

```
Amazon Simple Storage Service File System V1.93 (commit:528a617) with OpenSSL
```

配置

- **挂载**

1. 创建密钥文件：

```
mkdir DIR_NAME  //创建认证文件目录
echo AK:SK > PASSWD_FILE  //创建用户访问密钥文件；{AK,SK}是访问密钥对，可在金山云控制台的访问控制页面获取；PASSWD_FILE是密钥文件
chmod 600 PASSWD_FILE  //设置密钥文件只能被当前用户访问
```

示例：

```
mkdir /data/
echo AKLTWr****LaMgK:OKj7Zb/vMDaZVp****CUSYQA== > /data/passwd
chmod 600 /data/passwd
```

2. 创建mount目录：

```
mkdir MOUNT_PATH  //创建本地目录路径
```

示例：

```
mkdir /data/ks3
```

3. 挂载存储桶：

```
s3fs BUCKET_NAME MOUNT_PATH  -o passwd_file=PASS_FILE -o url=https://KS3_ENDPOINT -o sigv2
```

参数详情：	参数名称	说明
BUCKET_NAME	挂载KS3指定的桶名，若桶不为空，挂载时需要添加-o nonempty参数。	
MOUNT_PATH	挂载目录，例如已创建的/data/ks3。	
PASS_FILE	密钥文件，例如已创建的/data/passwd。	
KS3_ENDPOINT	金山云KS3各地域对应的Endpoin，例如挂载北京地域的存储桶，则输入：ks3-cn-beijing.ksyuncs.com。点击查看	
sigv2	签名版本号，默认为v2，可选v2和v4版本。	

示例：

```
s3fs s3test /data/ks3 -o passwd_file=/data/passwd -o url="https://ks3-cn-beijing.ksyuncs.com" -o sigv2 -o nonempty
```

4. 验证挂载结果：

```
df -h
```

挂载成功显示结果如下所示：

```
[root@vm10-0-0-175 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  456K  1.9G   1% /run
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/vda1        50G  3.1G   44G   7% /
tmpfs           374M     0  374M   0% /run/user/0
s3fs            4.0G     0  4.0G   0% /data/ks3
```

5. 查看桶内文件列表：

```
cd  /data/ks3
```

至此已完成KS3存储桶的挂载操作，您可以像操作本地文件一样操作KS3挂载的存储桶内文件。

- **卸载**

  1. 卸载存储桶：

  ```
  sudo umount MOUNT_PATH
  ```

  示例：

  ```
  sudo umount /data/ks3  //将桶从挂载目录/data/ks3卸载
  ```

  2. 查看是否卸载成功：

  ```
  df -h
  ```

  下图所示表明已卸载成功：

```
[root@vm10-0-0-175 ~]# sudo umount /data/ks3
[root@vm10-0-0-175 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  456K  1.9G   1% /run
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/vda1        50G  3.1G   44G   7% /
tmpfs           374M     0  374M   0% /run/user/0
```

常见问题

- 查看工具日志

输入以下命令查看工具日志：

```
vi /var/log/messages
```

- 挂载调试模式

输入以下命令挂载调试：

```
s3fs BUCKET_NAME MOUNT_PATH  -o passwd_file=PASS_FILE -o url=https://KS3_ENDPOINT -o sigV2 -d -d -f -o f2 -o curldbg
```

CloudBerry

简介

CloudBerry Explorer 是业界开发的一Windows 下直接通过 CloudBerry Explorer 来接入并管理对象存储的文件浏览器。您也可以通过CloudBerry Explorer来接入并管理金山云KS3。

CloudBerry主要功能包括：支持AK/SK登录，管理Bucket、管理Object、上传与下载、外链、同步等。

> 注意：在使用CloudBerry之前，您需要实现在金山云注册账号，并且开通金山云的对象存储服务（KS3）。详细开通步骤，请参见文档 开通KS3服务。

CloudBerry接入KS3

下载地址

- 官网下载地址：点击下载
- Window安装包：点击安装

安装及配置

安装

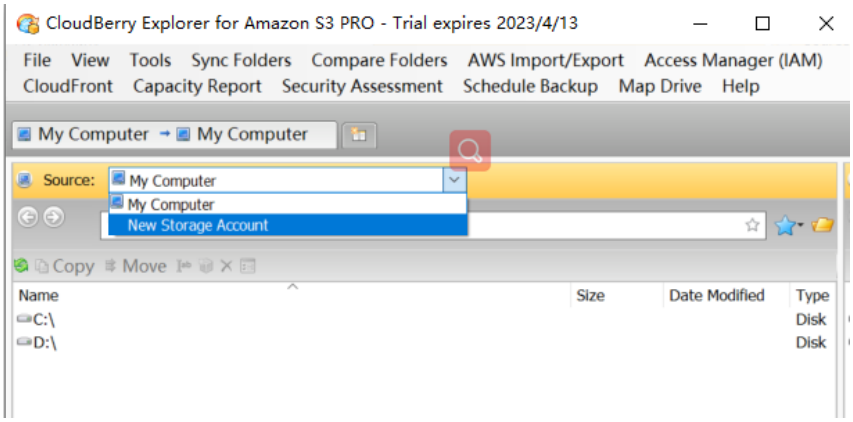
双击安装包，按照提示完成安装。

以下配置步骤以 CloudBerry Explorer Windows v5.9版本为例，其他版本的配置过程可能存在一定差异，请注意相应调整。

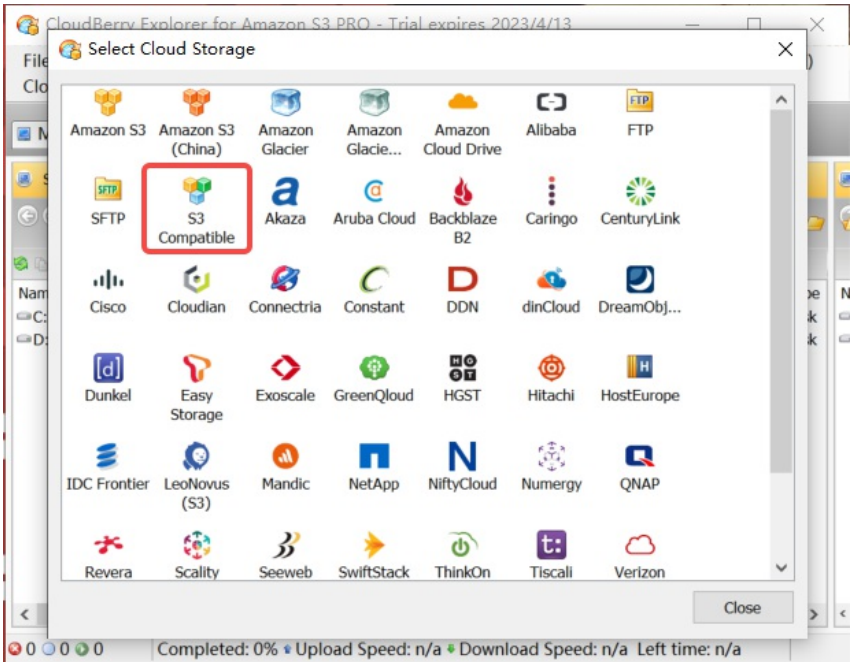
配置接入KS3

1.打开工具CloudBerry。

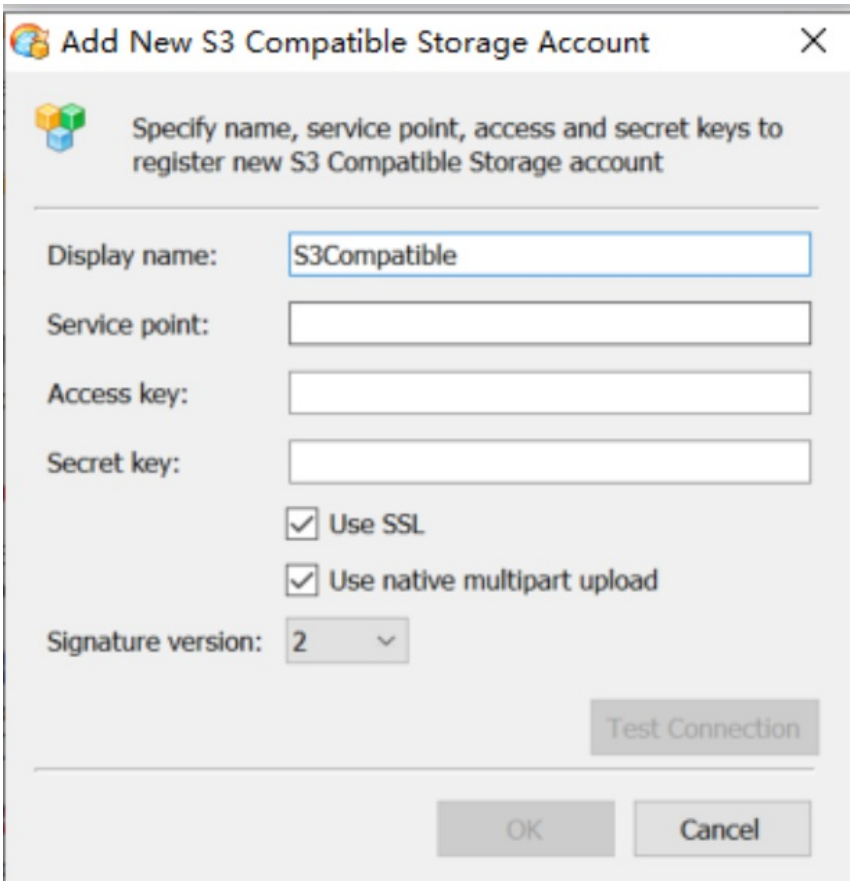
2. 在右侧的 Source 下拉菜单中点击 **New Storage Account**。



3. 在弹框中 点击选择 **S3 Compatible**。



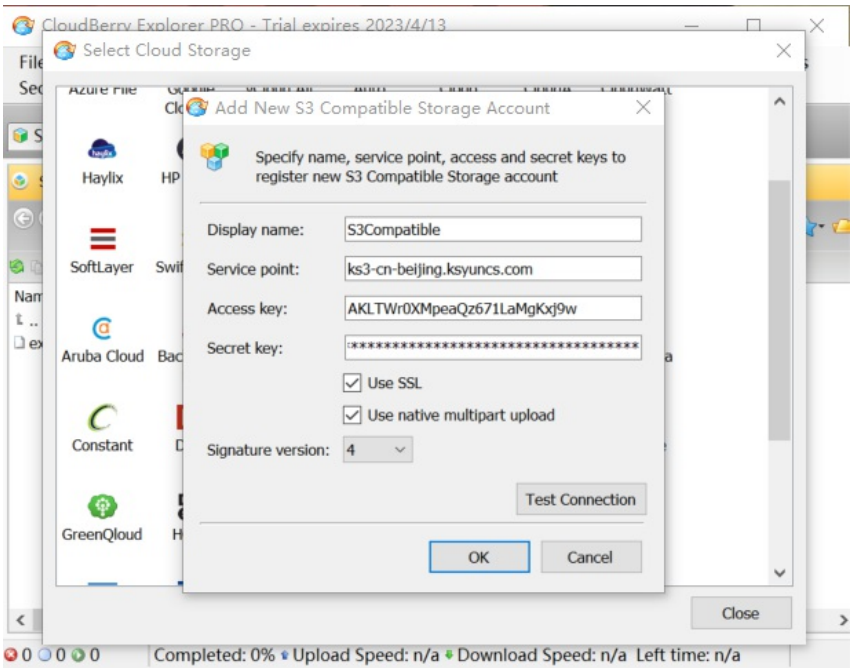
4. 在弹出的对话框中填写相应参数。



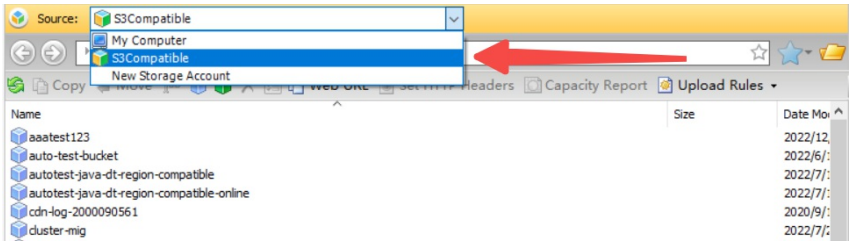
配置项说明如下：

名称	描述
Display name	自定义用户名
Service point	金山云KS3各地域对应的Endpoin，例如访问北京地域的存储桶，则输入：ks3-cn-beijing.ksyuncs.com。详情请参考文档Endpoint与Region的对应关系
Access key	访问密钥信息SecretId，可在 金山云控制台 的访问控制页面获取
Secret key	访问密钥信息Secretkey，可在 金山云控制台 的访问控制页面获取
Use SSL	SSL协议，默认勾选该选项
Signature version	签名方式，选择V2签名或V4签名，建议选择V4签名

5. 填写完毕单击 **Test Connection**，测试是否能连接成功，或者直接单击 **OK** 进行连接。



6. 连接成功之后，在Source选择刚才设置的账户名，看到账户下对应的Bucket列表，即表示已配置完成。



操作

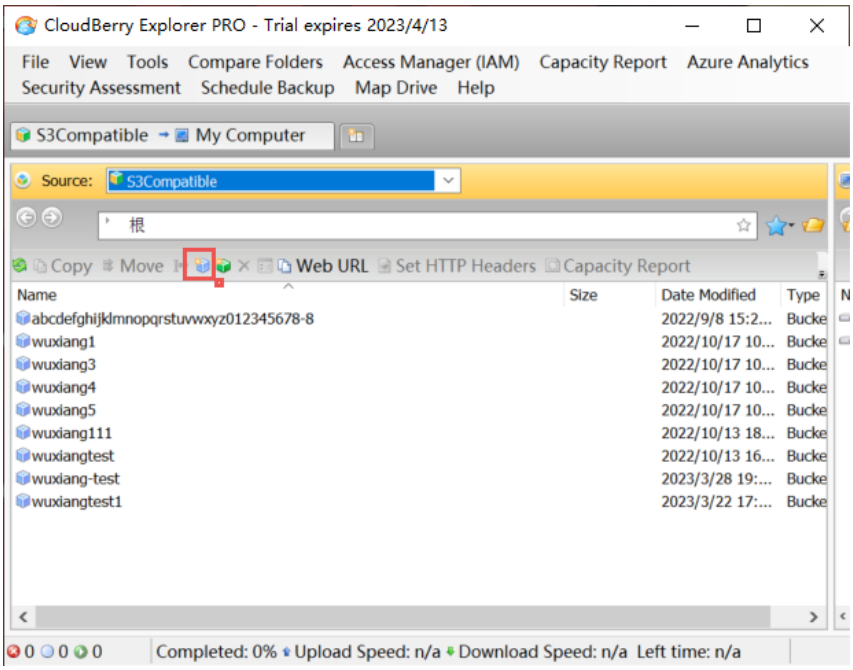
Bucket操作

在 Source 中选择之前设置的用户名，可查看该用户名下的存储桶列表。

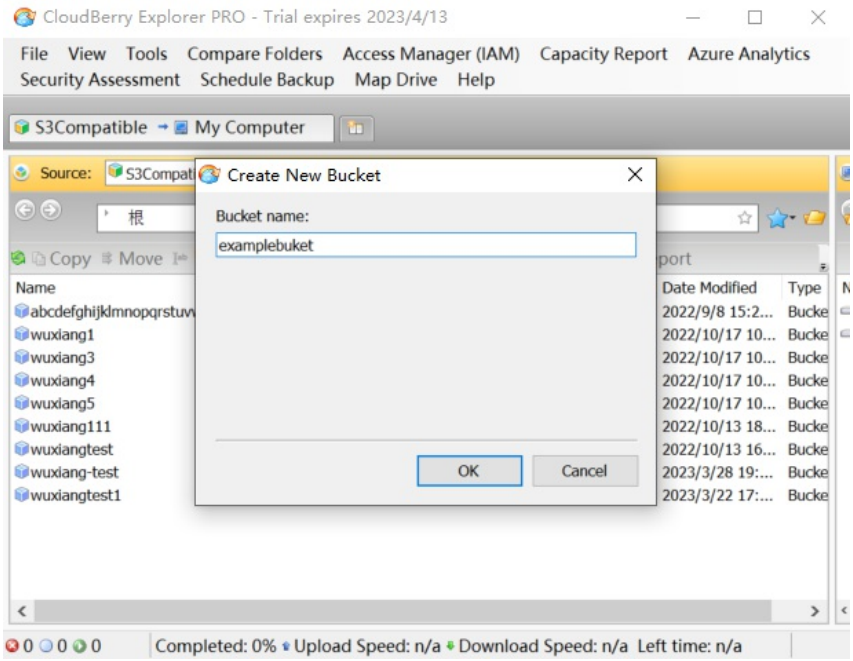
> 注意：只能查看 Service point 配置的地域所对应的存储桶。如需查看其它地域的存储桶，可单击 File > Edit Accounts，并选择用户名，修改 Service point 参数为其他地域即可。

创建Bucket

1. 在工具右侧窗口中，单击下图所示标注的图标。

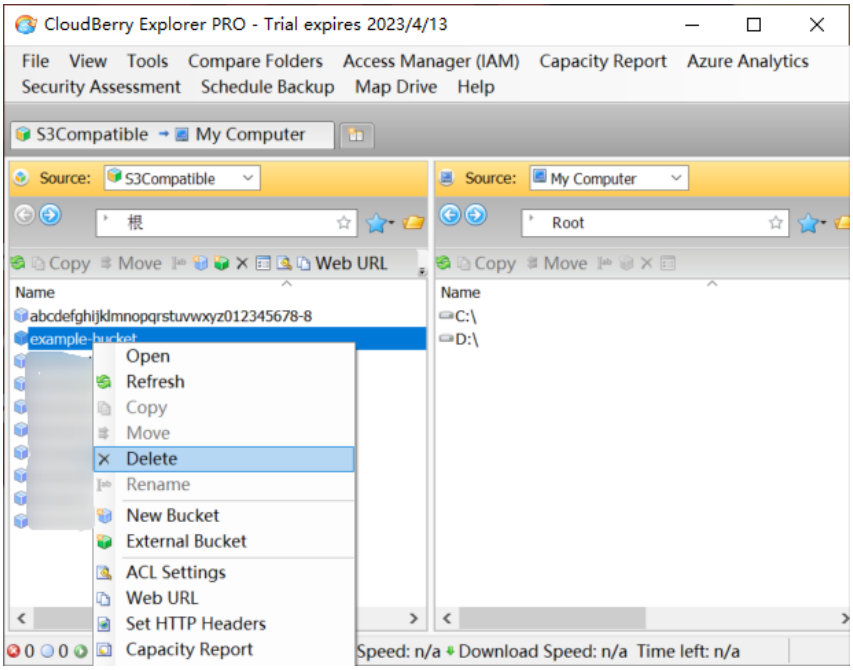


2. 在弹窗中输入完整的存储桶名称，例如 examplebucket。输入无误后，单击 **OK** 即可创建完成。

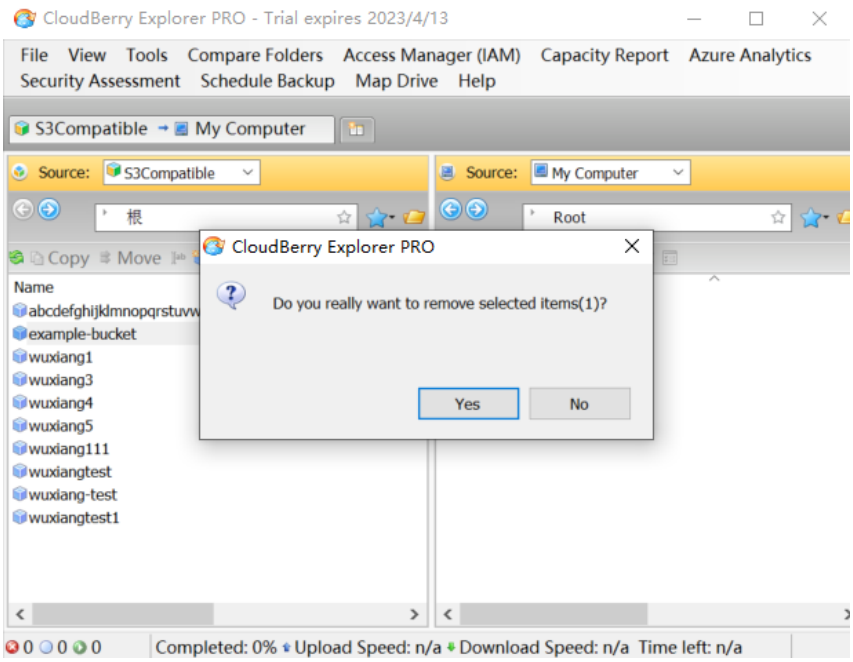


删除Bucket

1. 在工具右侧窗口的存储桶列表中选择需删除的存储桶，单击 Delete 。

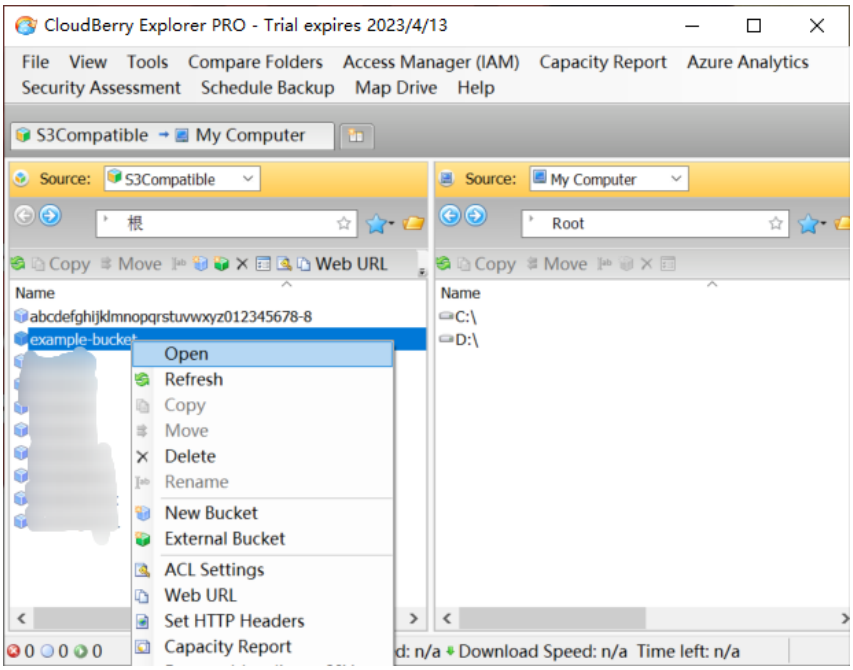


2. 确认弹窗信息，单击 Yes 即可完成删除存储桶操作。

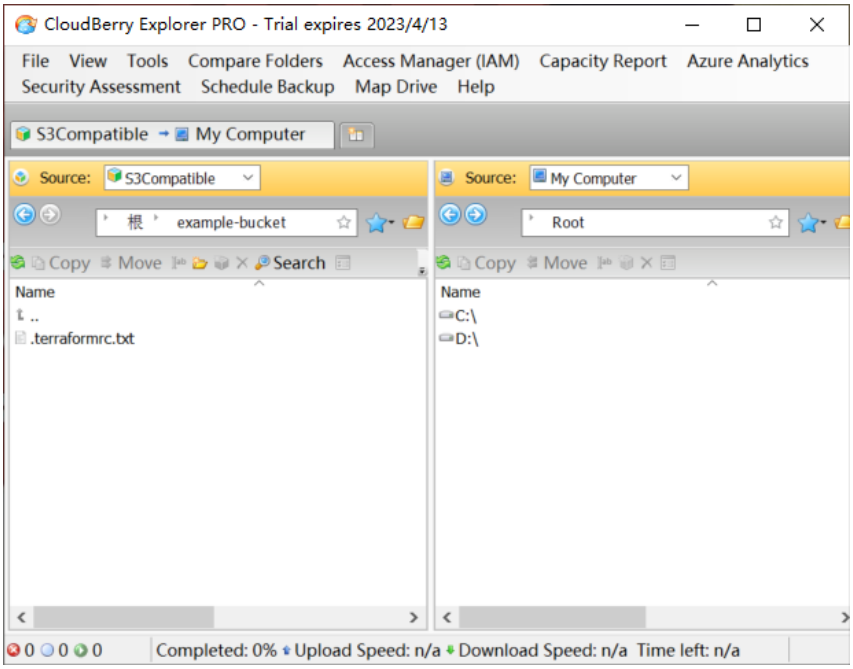


查看Bucket

1. 在工具右侧窗口的存储桶列表中选择需查看的存储桶，单击 **Open** 。



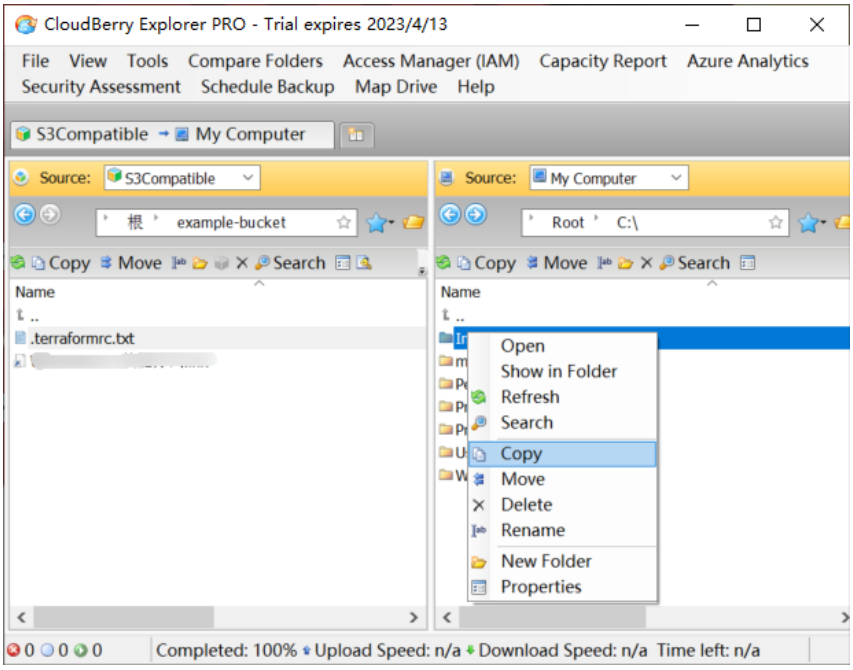
2. 在工具的右侧窗口中，可查看选择的存储桶中的内容。



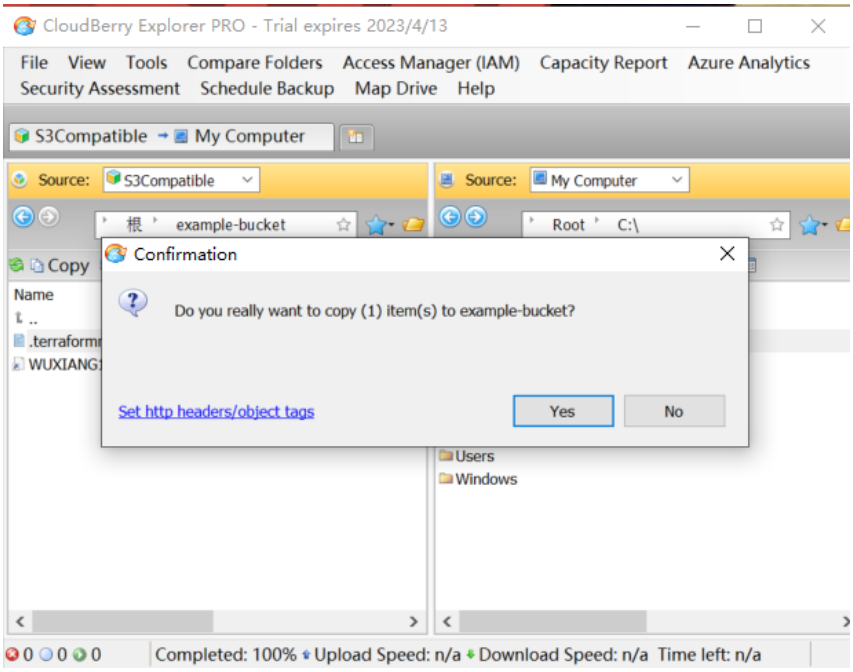
Object操作

上传 Object

1. 在工具的右侧窗口中，选择对象被复制后的目标路径，然后在左侧窗口中选择需复制的对象，右键单击选择 Copy。

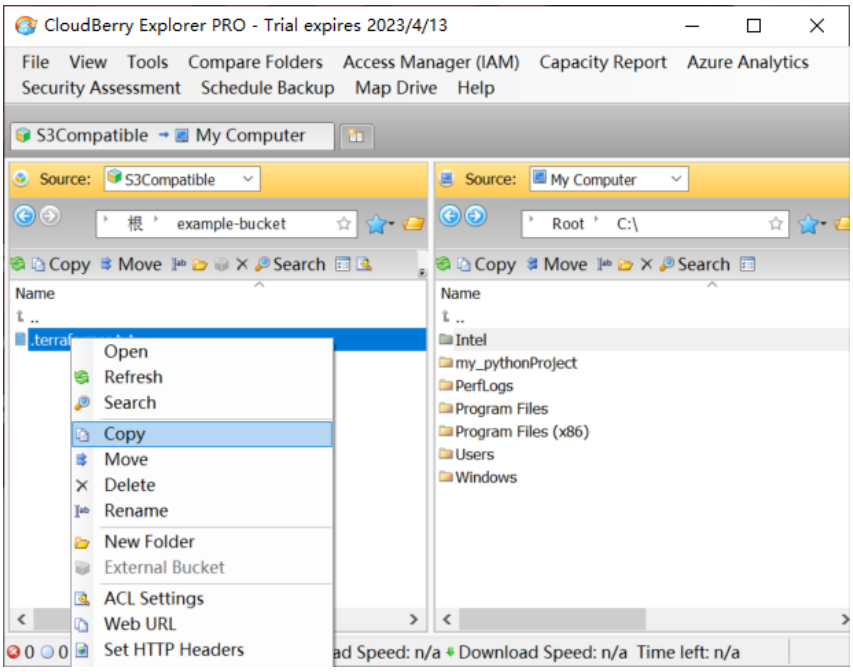


2. 确认弹窗信息，单击 Yes 即可完成上传对象操作。

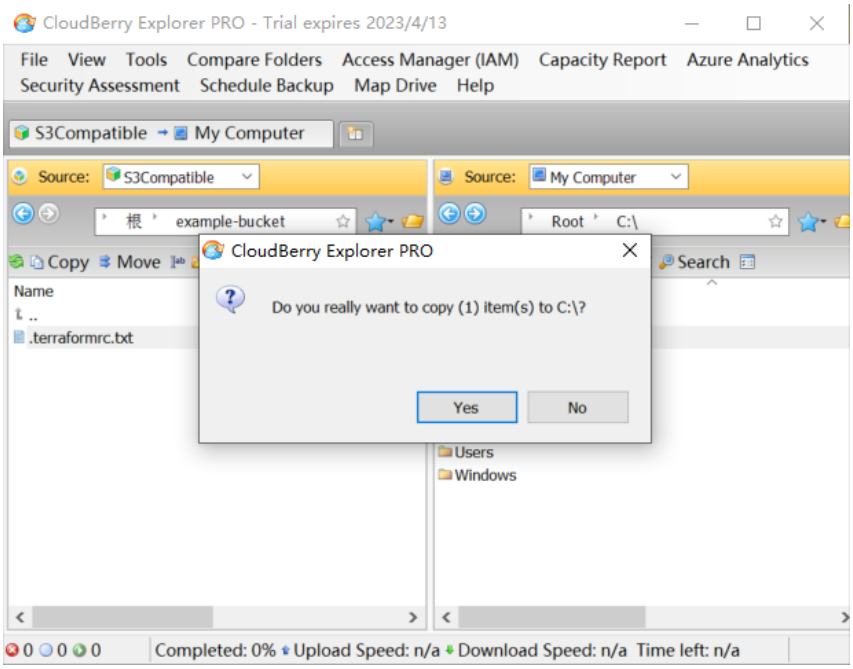


下载 Object

1. 在工具的左侧窗口中，选择被下载的对象，然后在右侧窗口中选择下载后的目标路径，右键单击选择 Copy。

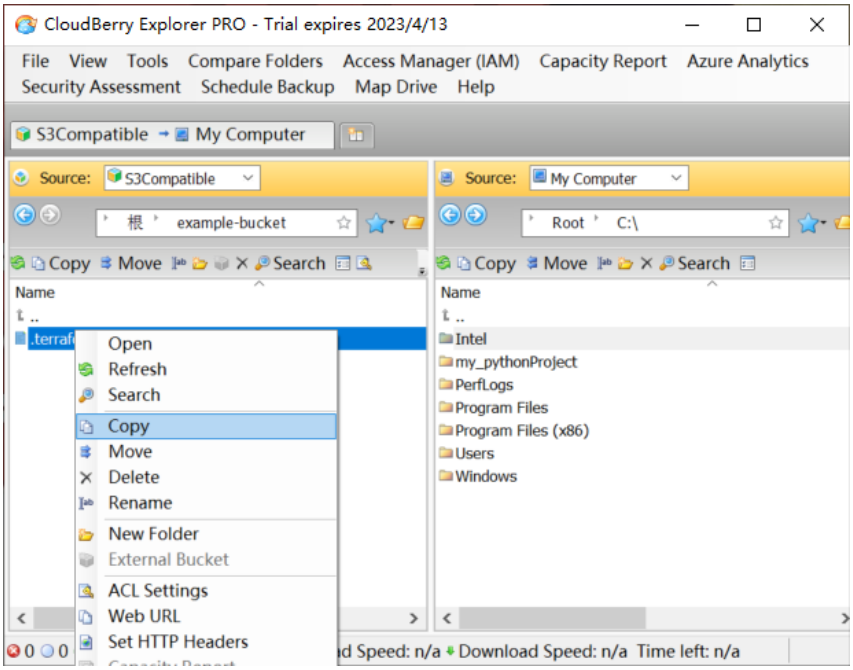


2. 确认弹窗信息，单击 Yes 即可完成下载对象操作。

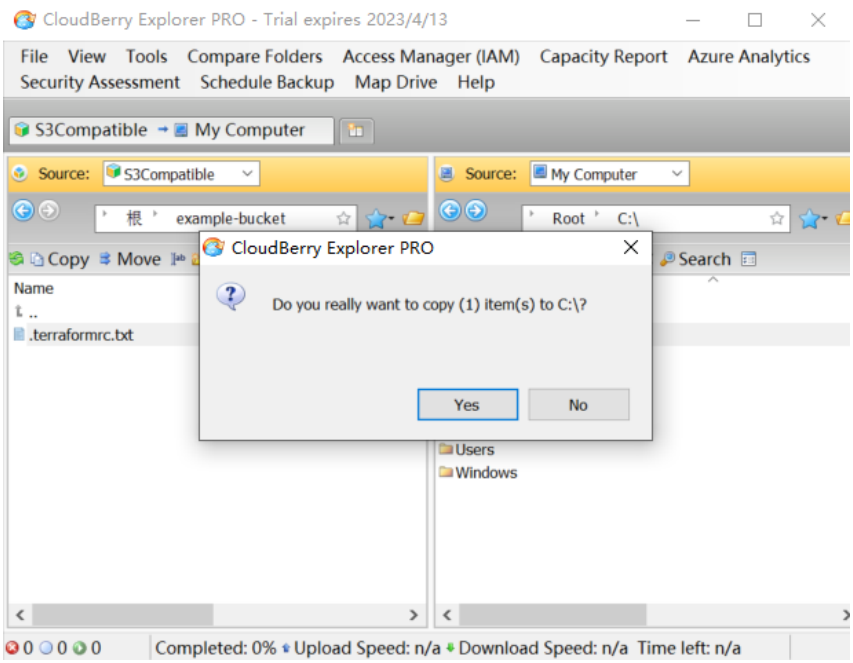


复制 Object

1. 在工具的左侧窗口中，选择需复制的对象，然后在右侧窗口中选择下载后的目标路径，右键单击选择 Copy。

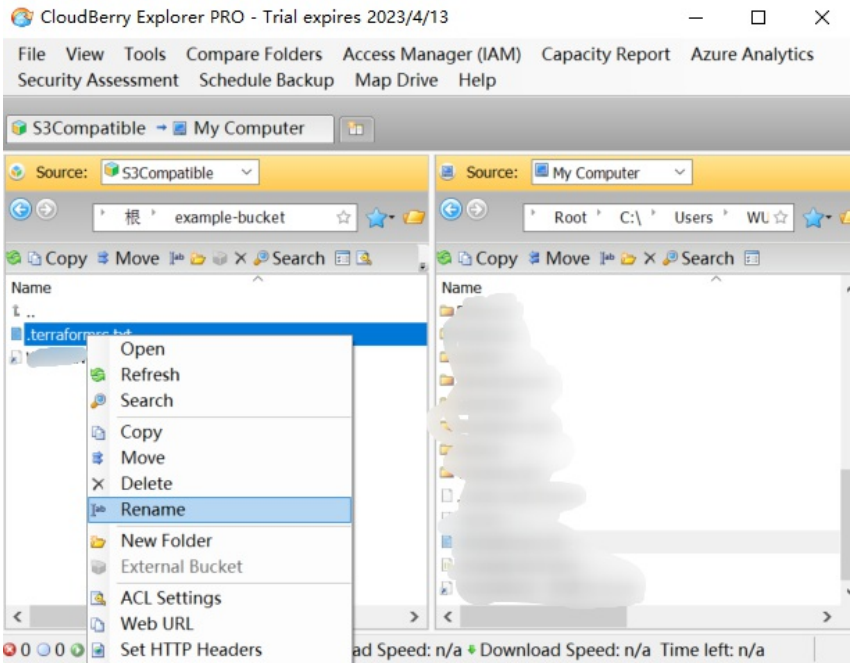


2. 确认弹窗信息，单击 Yes 即可完成复制对象操作。

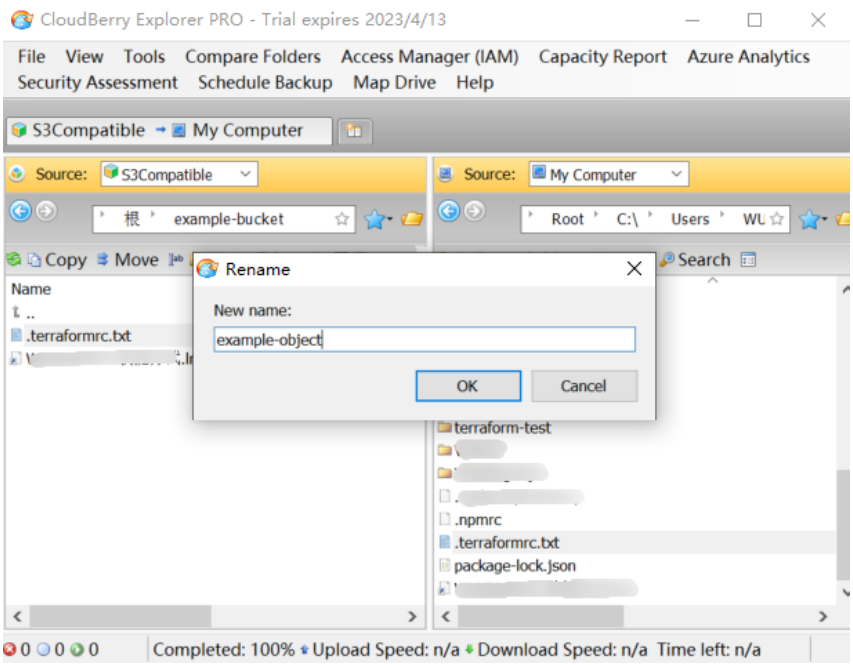


重命名 Object

1. 在工具的左侧窗口中，选择需重命名的对象，右键单击选择 Rename。

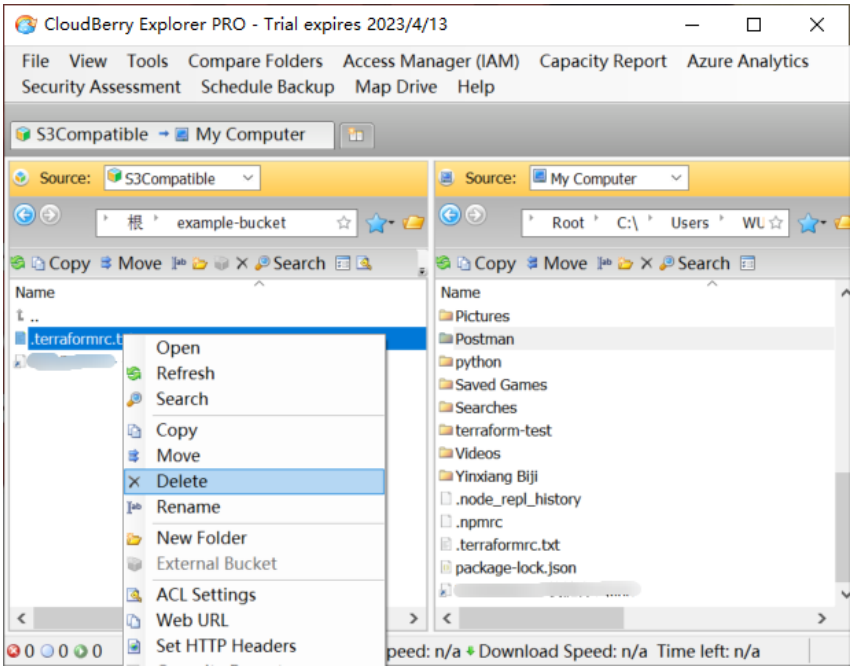


2. 在弹窗中输入完新名称，例如 example-object，输入无误后，单击 OK 即可重命名完成。

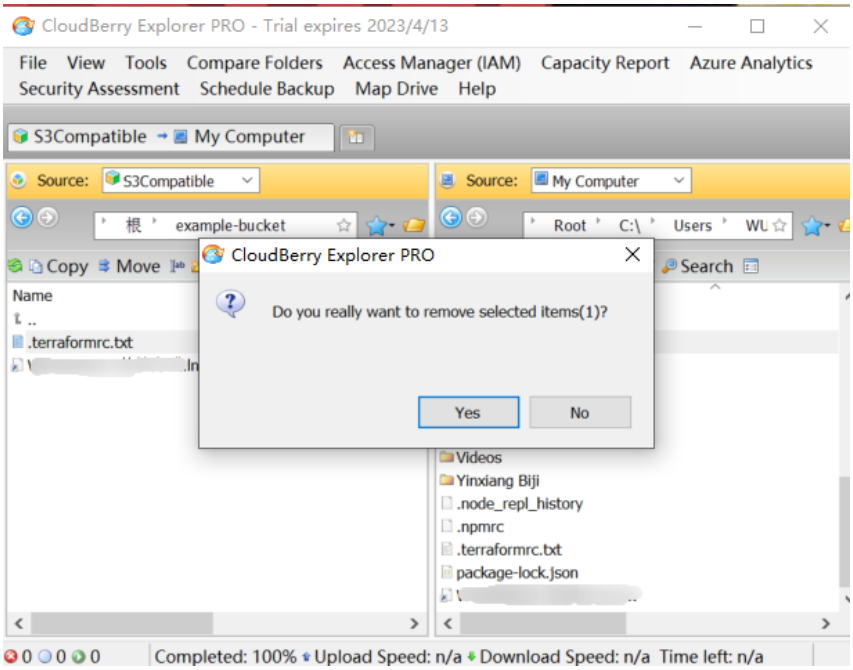


删除 Object

1. 在工具的右侧窗口中，在存储桶中找到需删除的对象，并右键单击 Delete。

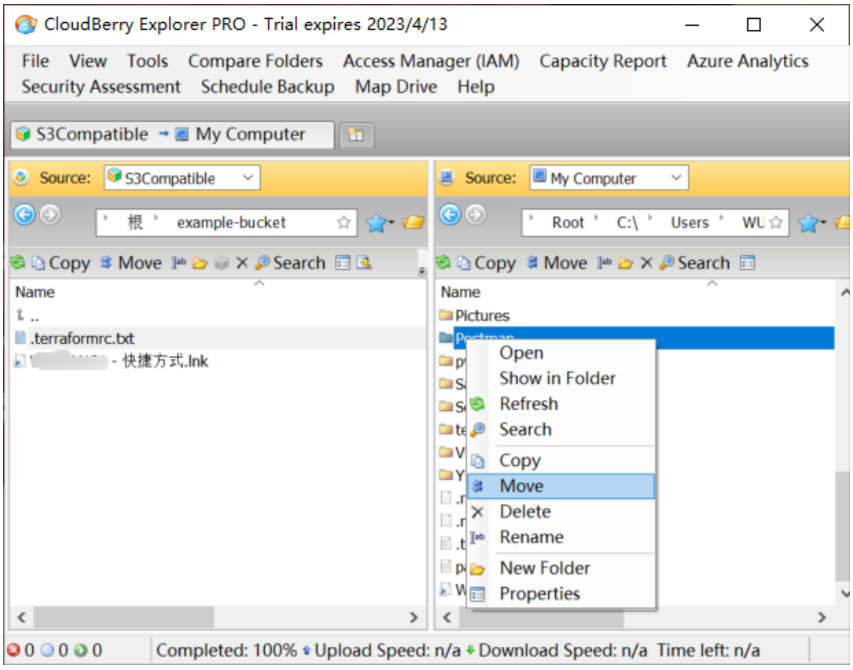


2. 确认弹窗信息，单击 Yes 即可完成对象删除操作。

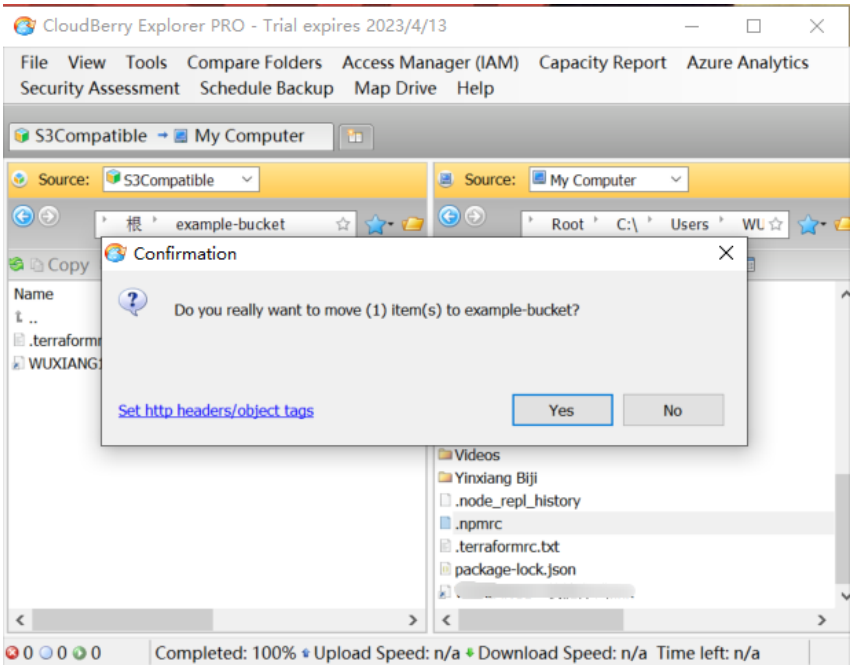


移动 Object

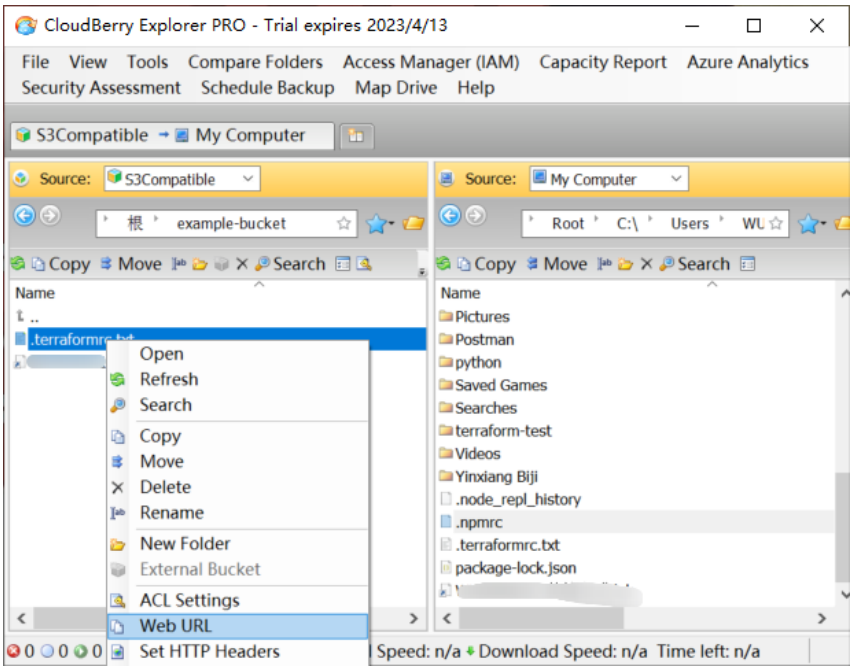
1. 在工具的右侧窗口中，选择对象被移动后的目标路径，然后在左侧窗口中选择需移动的对象，右键单击选择 Move。



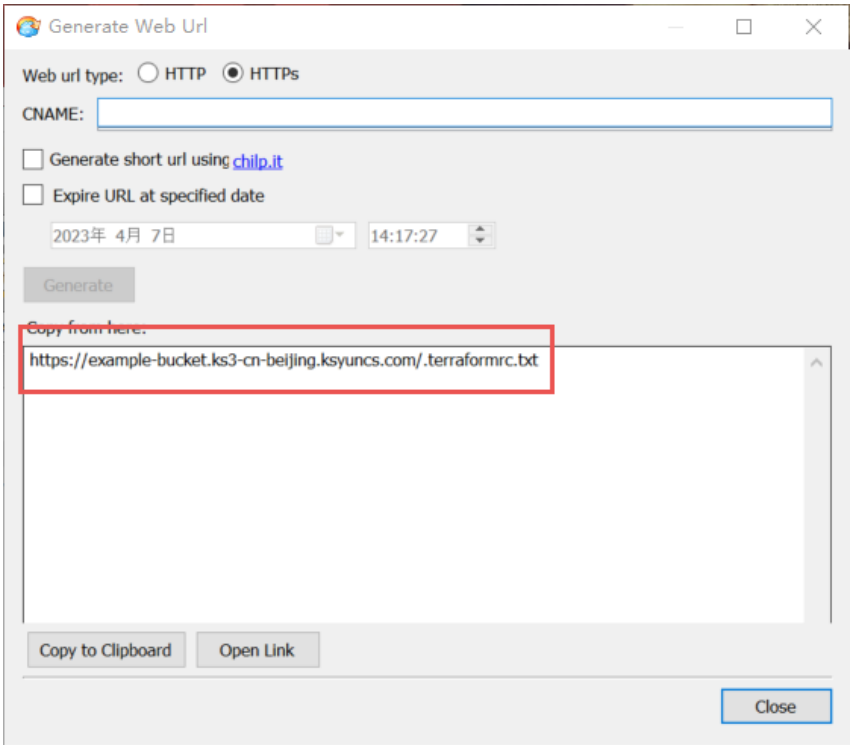
2. 确认弹窗信息，单击 Yes 即可完成对象移动操作。



1. 在工具的右侧窗口中，在存储桶中找到目标对象，并右键单击 Web URL。



2. 对象的URL会显示在如下图所示的弹出窗口中。

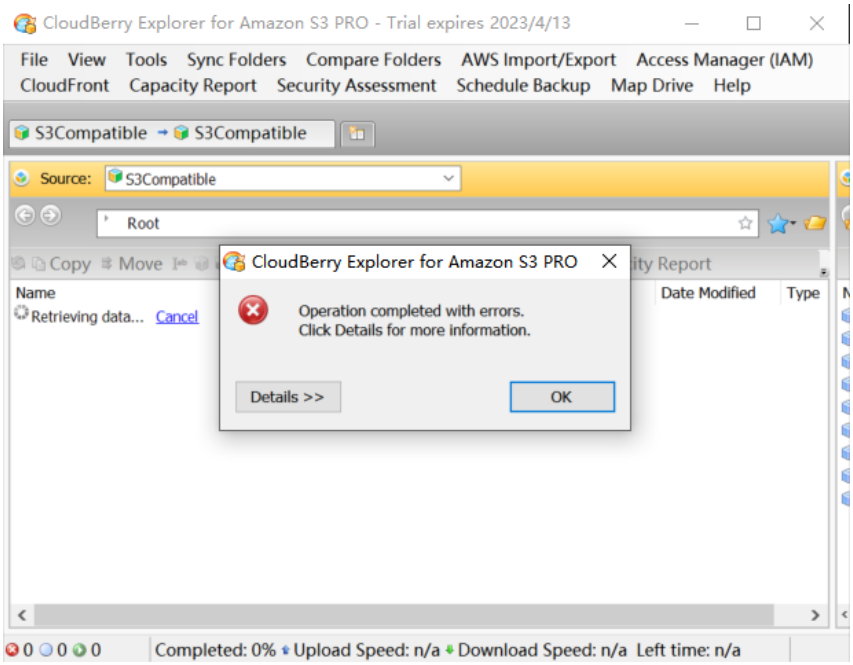


其他操作

除以上操作外，CloudBerry Explorer 还支持其他操作，例如设置对象 ACL、查看对象元数据、自定义 Headers等。用户可根据实际需求进行操作。

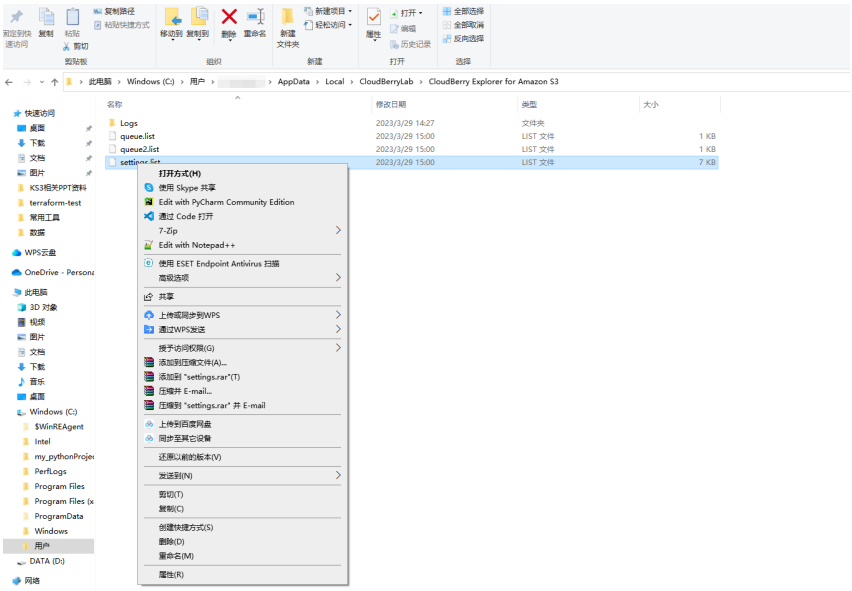
相关问题

访问Bucket时，有可能出现如下图所示的错误：



这是由于KS3仅支持S3的virtual hosted style（三级域名）访问方式，这也是S3推荐使用的访问方式。而目前版本的Cloudberry（5.9）默认使用的是Path style（二级域名）。

要将访问方式改为virtual hosted style，需要在Cloudberry Explorer的配置文件中进行设置。配置文件默认路径为：Users->AppData->Local->CloudBerryLab->Explorer for AmazonS3->settings.list



打开配置文件将对应用户的RequestStyle标签改为VHost，如下图所示，保存退出后重启CloudBerry即可生效。

注意：若VHost大小写编写错误，会产生403错误

至此已经配置完成，用户可以使用CloudBerry Explorer for AmazonS3管理和访问金山云KS3了。

S3 Browser

简介

S3 Browser是一种易于使用和强大的Amazon S3免费客户端。 金山云KS3兼容 Amazon S3 API，您可使用S3 Browser管理金山云KS3，详见 AWS S3协议兼容。

使用S3 Browser接入KS3

下载

- 官网地址：点击下载
- Window安装包：点击安装

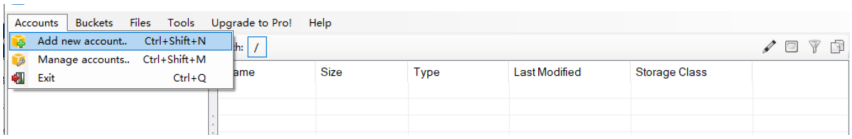
安装

双击安装包，按照提示完成安装。

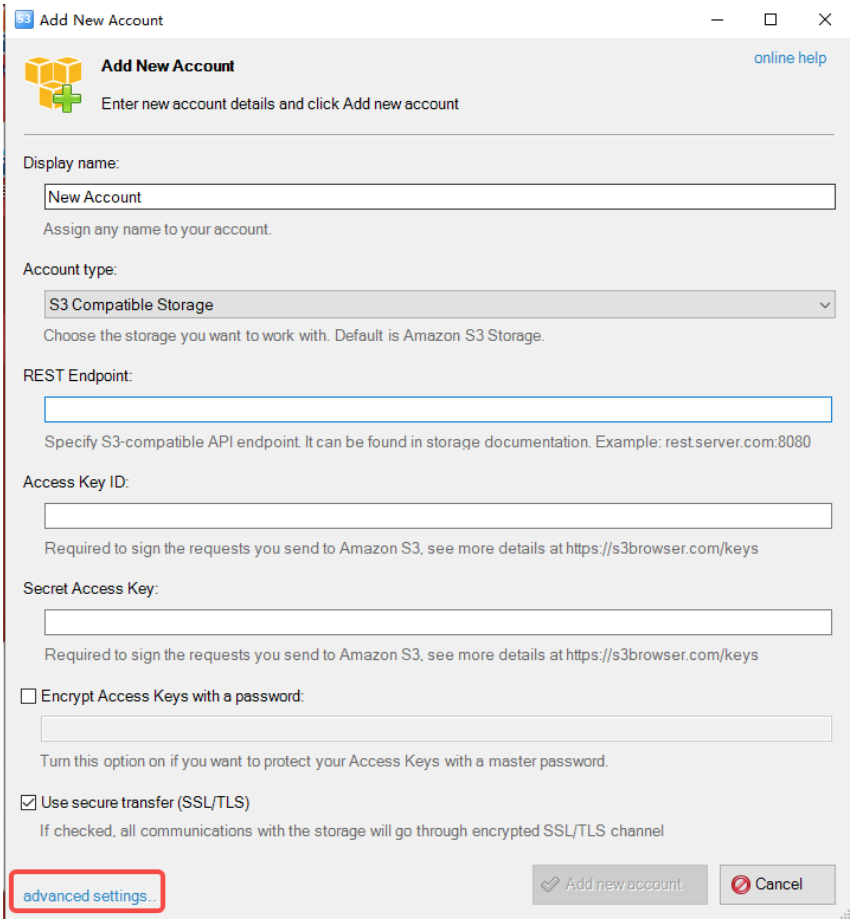
配置接入KS3

注意：以下配置步骤以 S3 Browser v5.9版本为例，其他版本的配置过程可能存在一定差异，请注意相应调整。

1. 在左侧的account 下拉菜单中点击 **add new account** 。



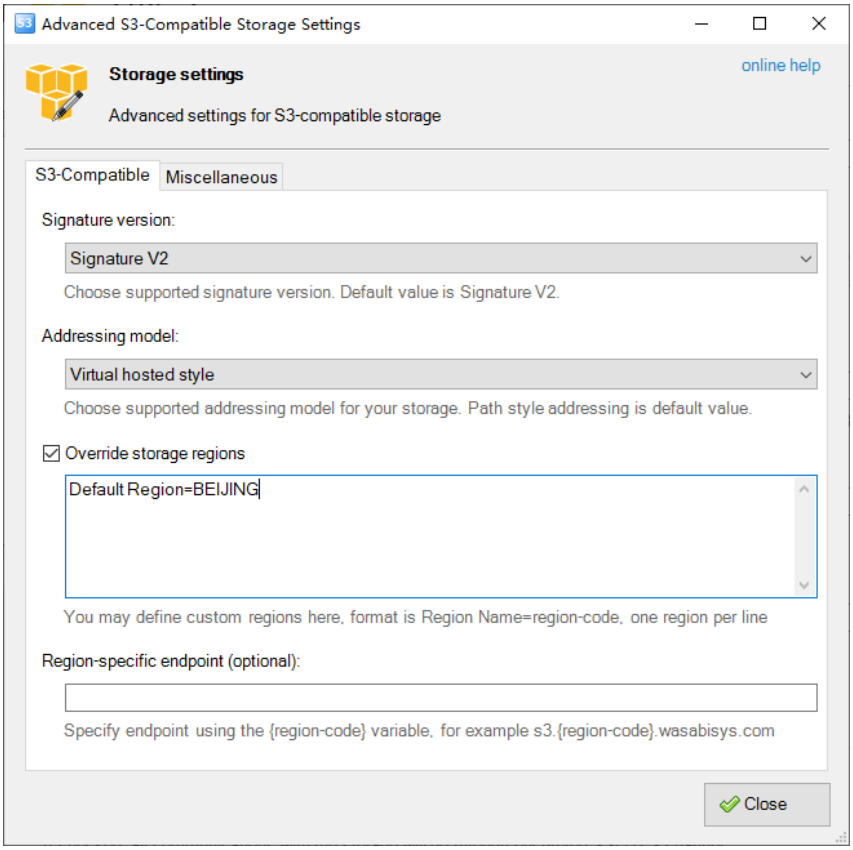
2. 在弹出的对话框中，填写以下相应参数，单击 **advance setting**。



配置项说明如下：

名称	描述
Account Name	输入自定义用户名
Account Type	账户类型，选择S3 Compatible Storage
REST Endpoint	填写金山云KS3各地域对应的Endpoin，例如访问北京地域的存储桶，则输入ks3-cn-beijing.ksyuncs.com，详情参考 Endpoint与Region的对应关系
Access Key ID	输入访问密钥信息SecretId，可在金山云控制台的访问控制页面获取
Secret Access Key	输入访问密钥信息SecretKey，可在金山云控制台的访问控制页面获取
Use secure transfer（SSL/TLS）	默认勾选该选项

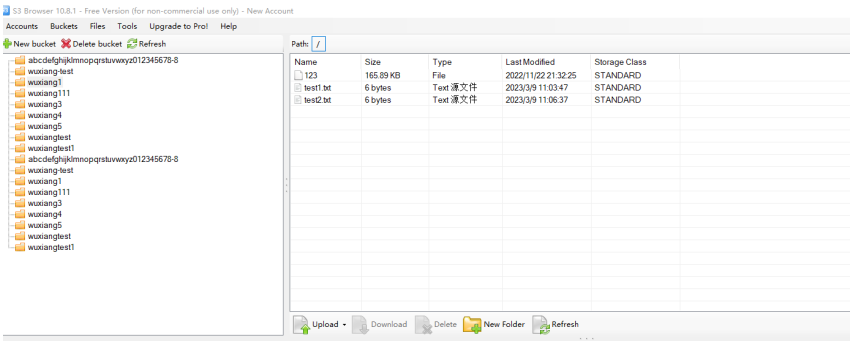
3. 在新弹出的对话框中，填写以下相应参数。



配置项说明如下：

名称	描述
Signature version	选择V2签名或V4签名
Addressing model	Path style：二级域名，Virtual hosted style：三级域名。选择Virtual hosted style
Override storage regions	填写金山云KS3各地域名称，例如访问北京地域的存储桶，则输入Default Region=BEIJING，详情参考Endpoint与Region的对应关系

4. 先单击 Close， 此时左侧显示桶列表，点击目标存储桶，右侧显示桶内文件，即表示已配置完成。



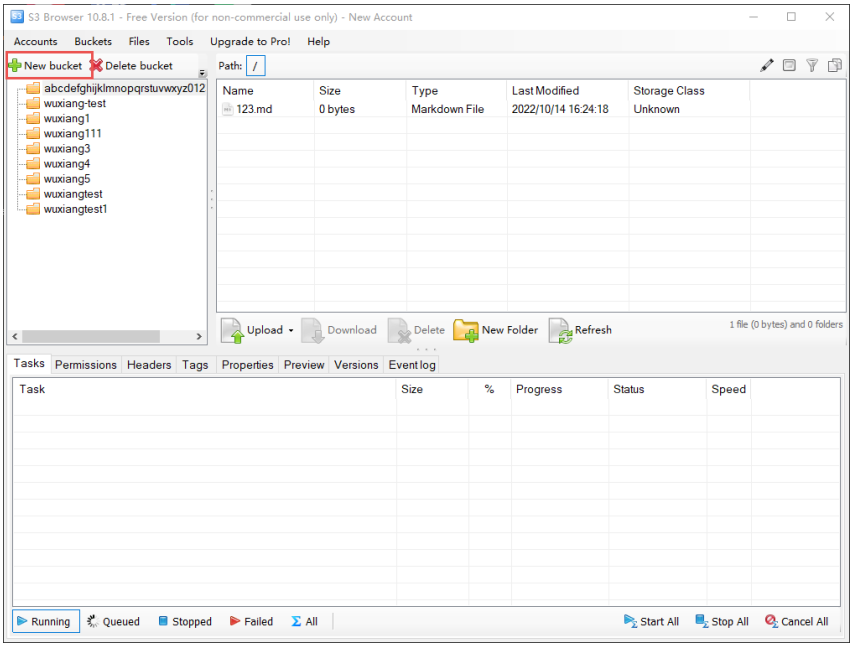
操作

Bucket操作

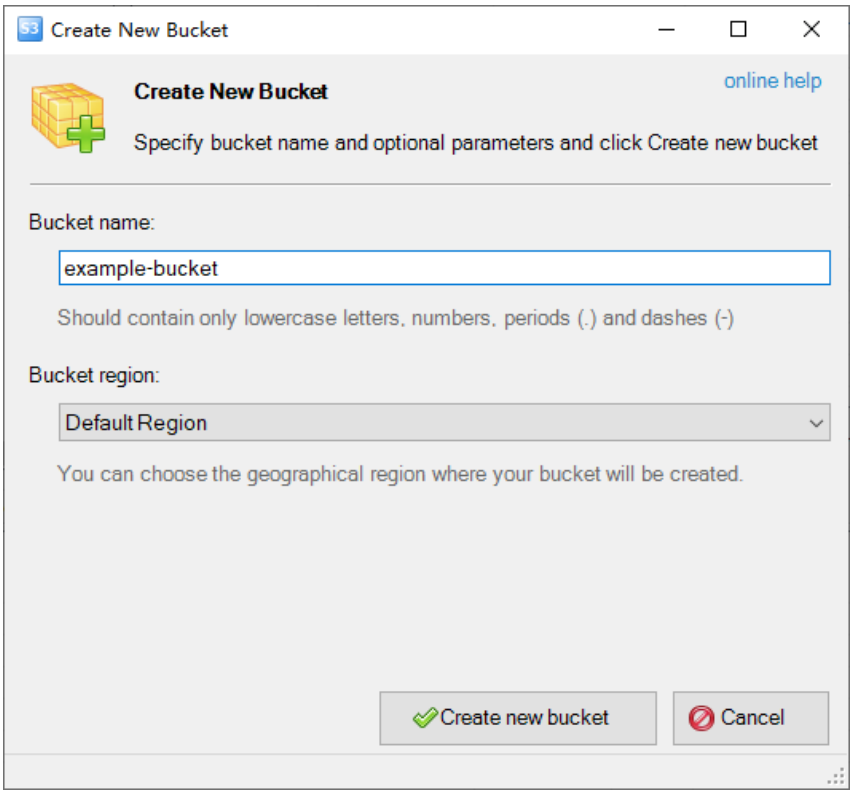
在 Source 中选择之前设置的用户名，可查看该用户名下的存储桶列表。

- 创建Bucket

1. 在工具的窗口中，单击 New bucket 或者右键单击 New bucket。

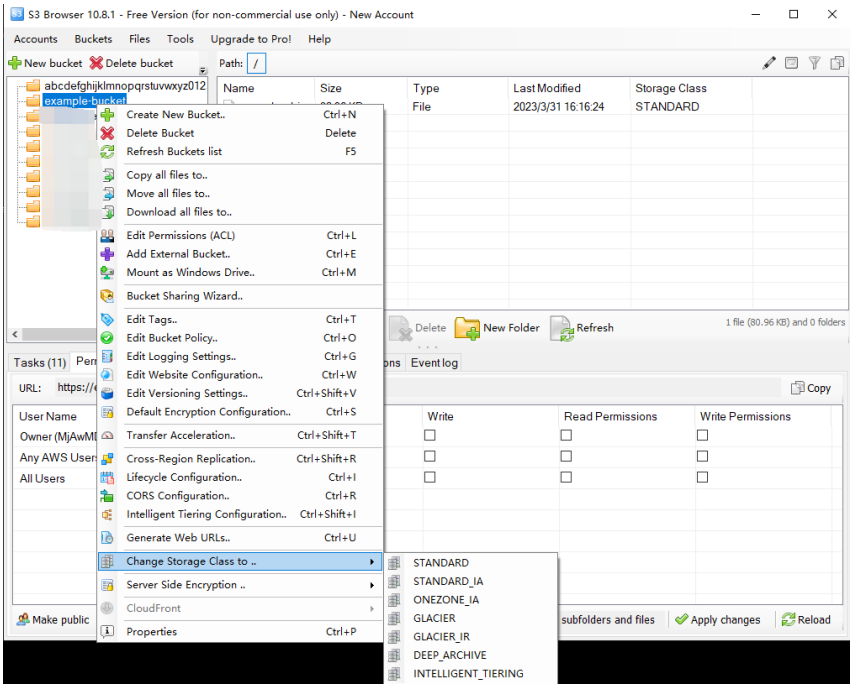


2. 在弹窗中输入完整的存储桶名称，例如 examplebucket。输入无误后，单击 √Create new bucket 即可创建完成。

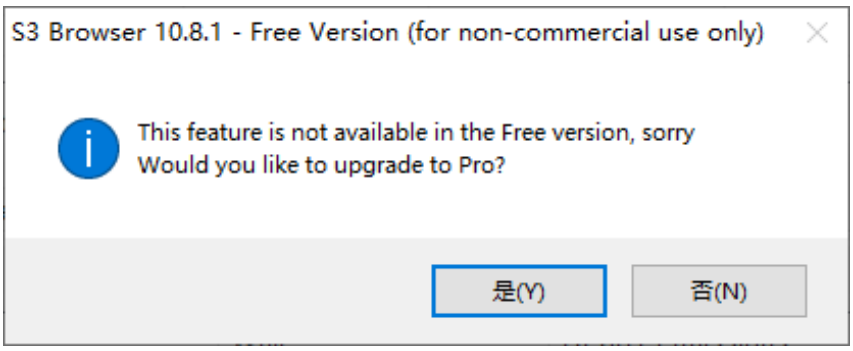


- **设置Bucket的存储类型**

创建的Bucket默认为标准存储类型，右键单击 Change Stroage Class to，选择想要转换的存储类型。

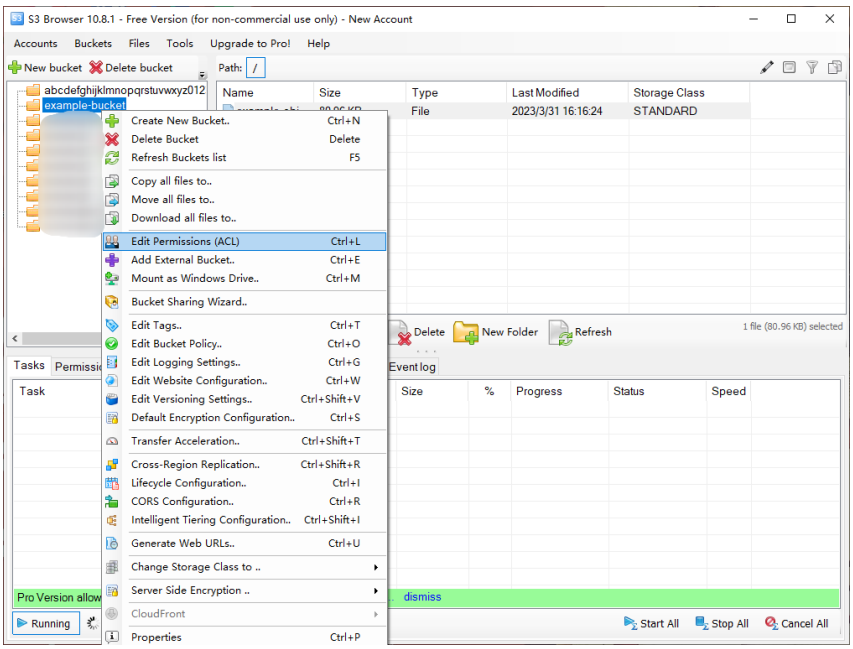


会出现如下提示框。（免费版本不支持，需要购买 S3 Browser Pro版本）



- **设置访问控制权限（ACL）**

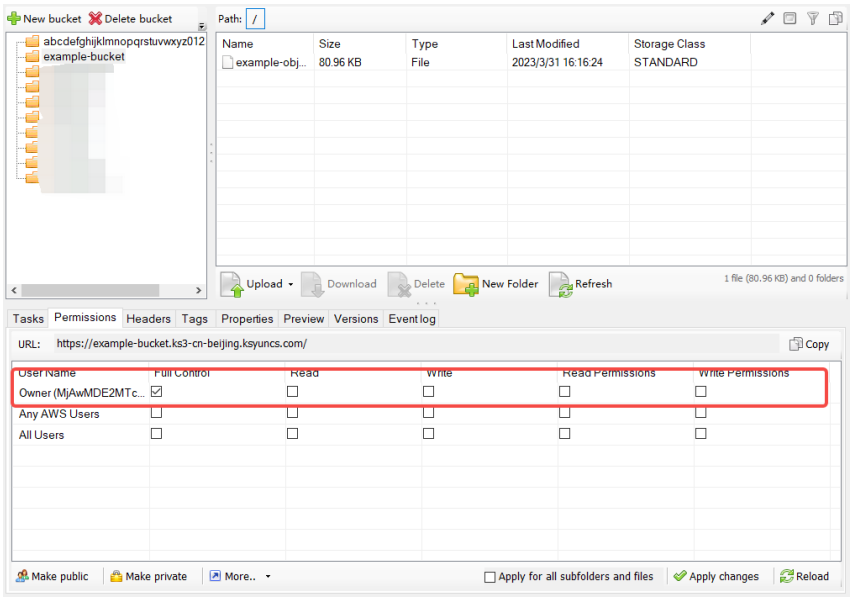
1. 在工具的窗口中，在存储空间列表选中需要设置的存储桶，单击右键 **Edit Permission（ACL）**。



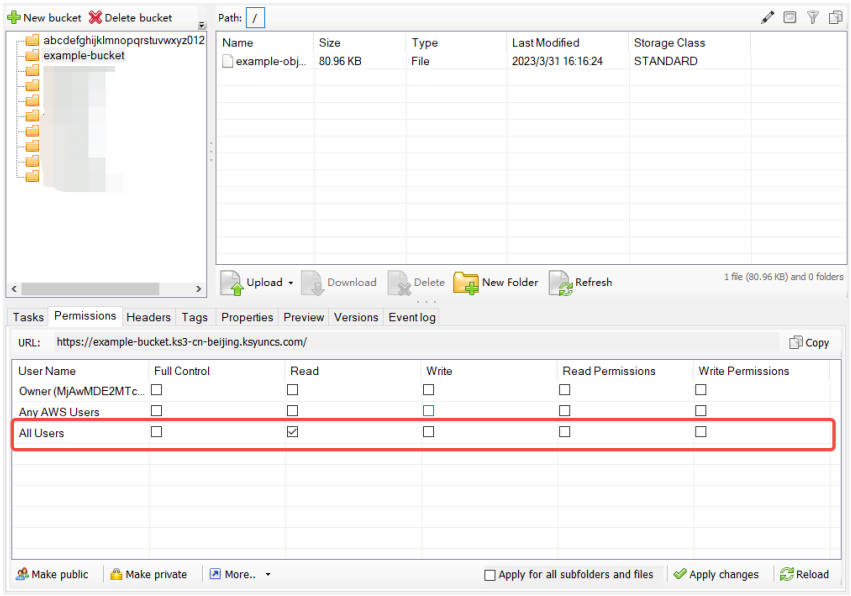
2. 按照KS3支持的规则设置，详细支持设置参见权限设置，完成设置之后，单击 **Apply changes** 即可完成Bucket ACL设置。

KS3支持以下设置：

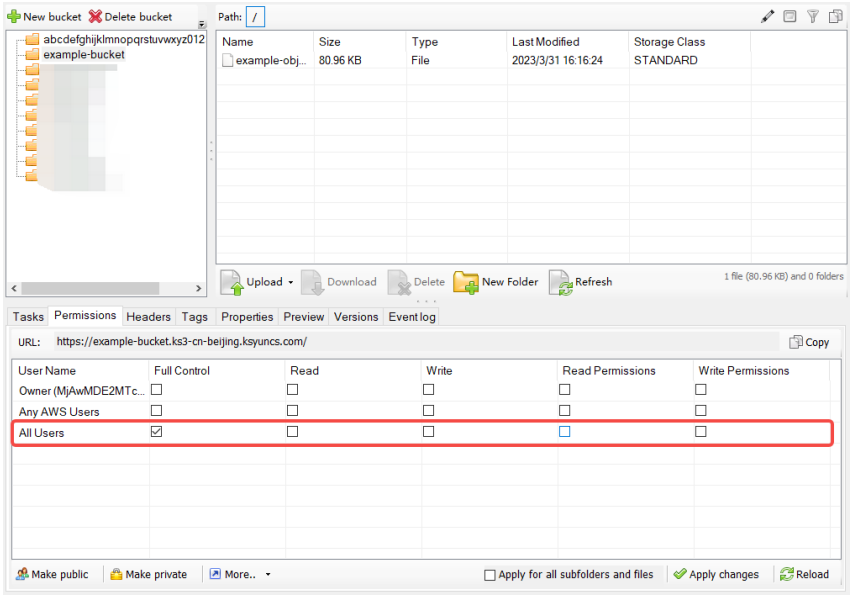
* 私有读写——Owner（Full Control）



* 公有读私有写——All Users（Read）

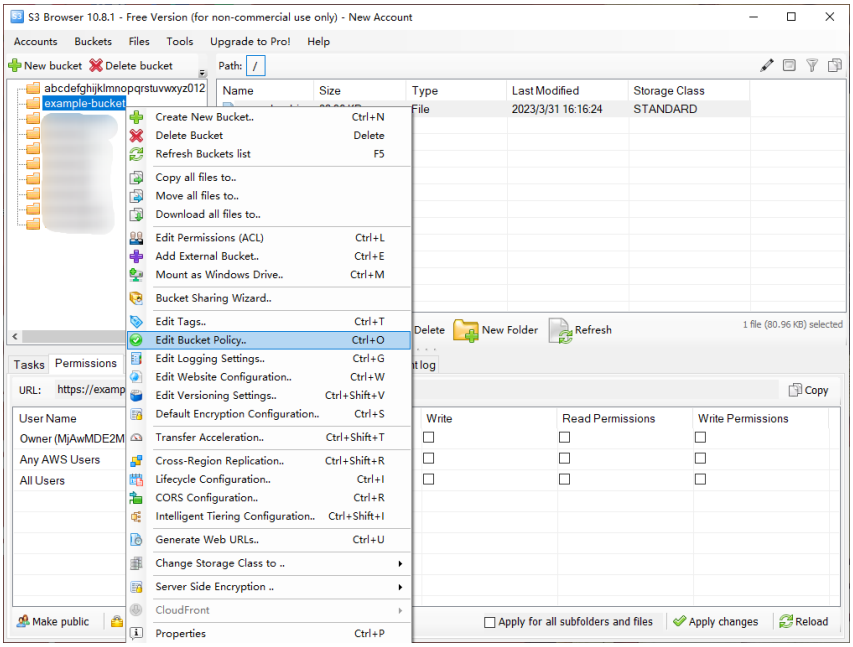


* 公开读写——All Users（Full Control）

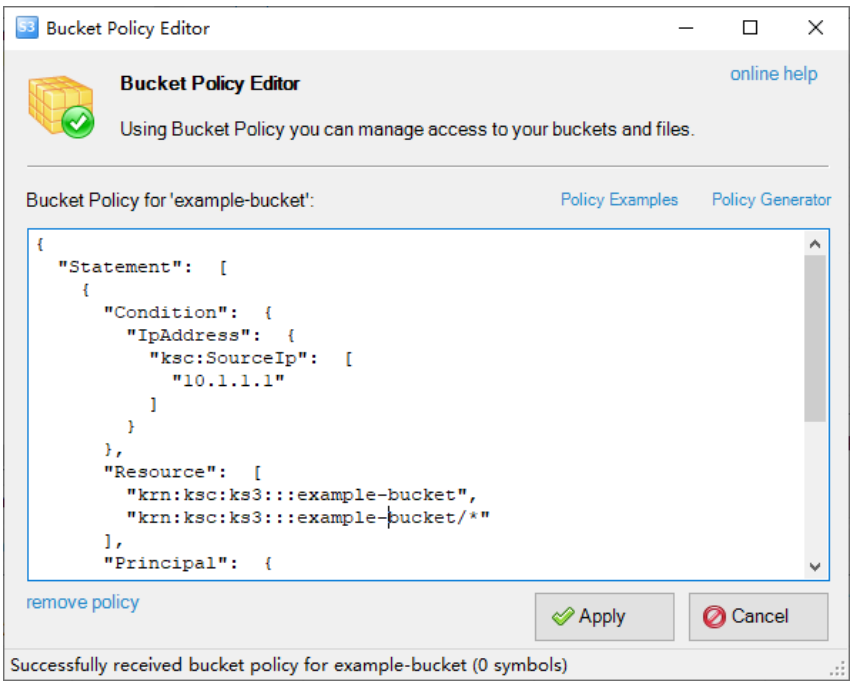


- **设置空间策略（Bucket Policy）**

1. 在工具的窗口中，在存储空间列表选中需要设置的存储桶，单击 Edit Bucket policy。

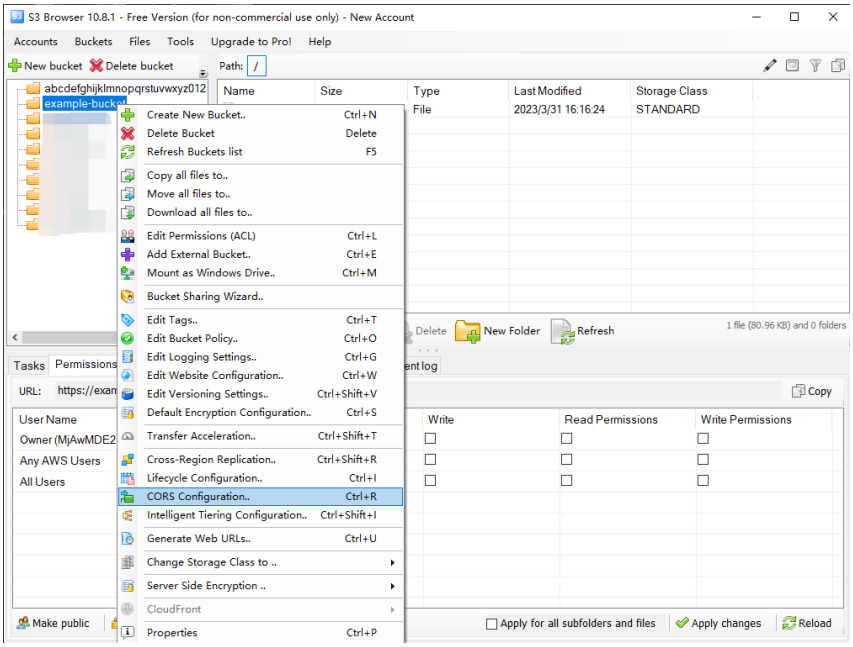


2. 按照KS3的Bucket policy规则设置，详细规则设置参见空间策略设置，完成设置之后，单击 Apply 即可完成Bucket Policy设置。

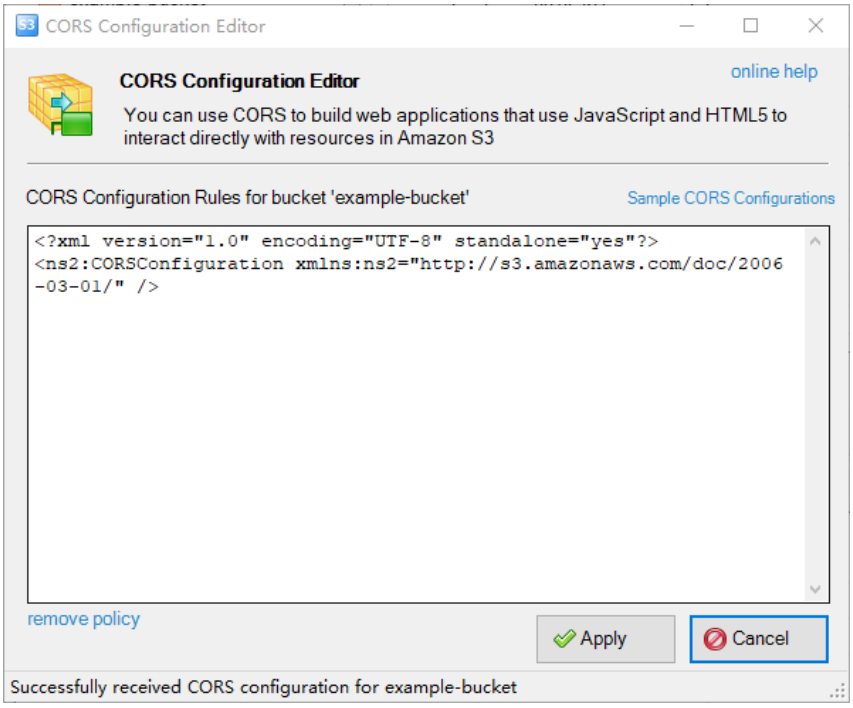


- **设置跨域资源共享（CORS）**

1. 在工具窗口的存储空间列表中，选中需要设置的存储桶，单击右键 CORS Configuration。

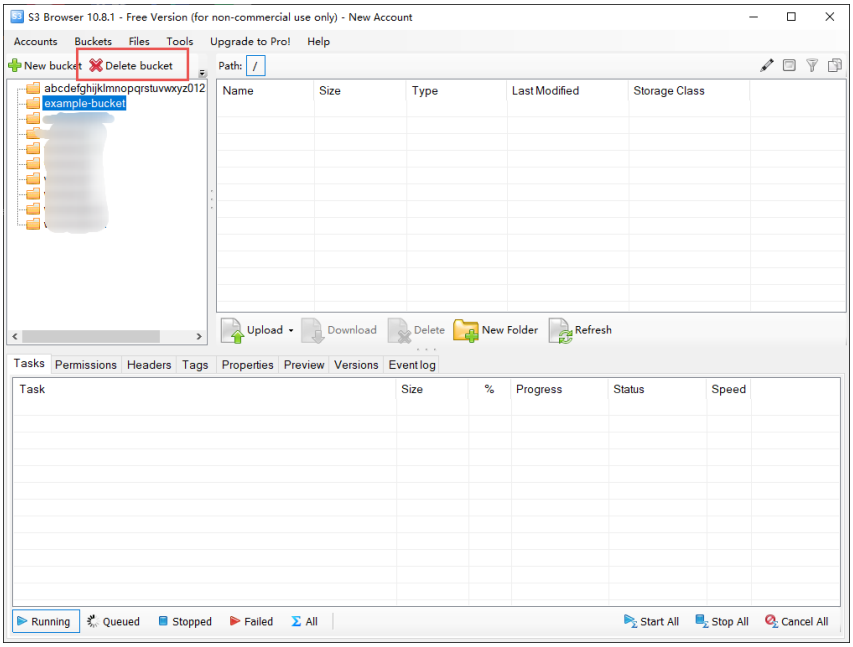


2. 弹框中输入CORS配置，可以点击下图中 Sample CORS Configuration 文字链接，在网页中挑选模板，按照自己需求修改，点击 Apply，完成设置。

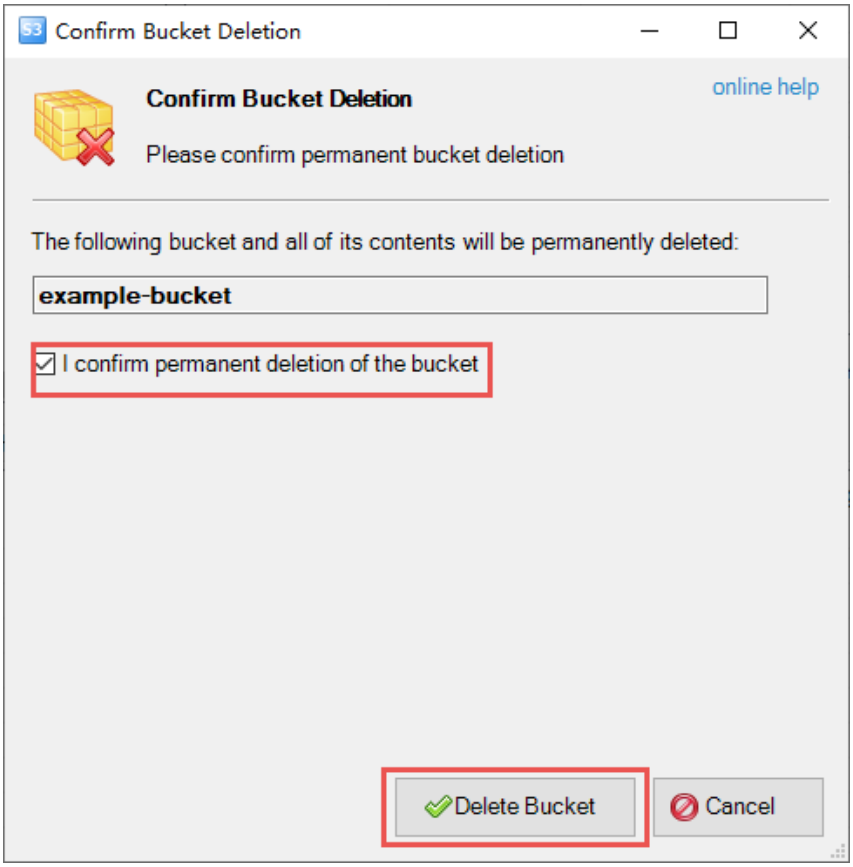


- **删除Bucekt**

1. 在工具的窗口中，在存储桶列表中选择需删除的存储桶，单击 Delete bucket 或者 右键单击 Delete bucket。



2. 确认弹窗信息，先选中I confirm permanent deletion of the bucket，之后单击 √Delete Bucket 即可完成删除存储桶操作。



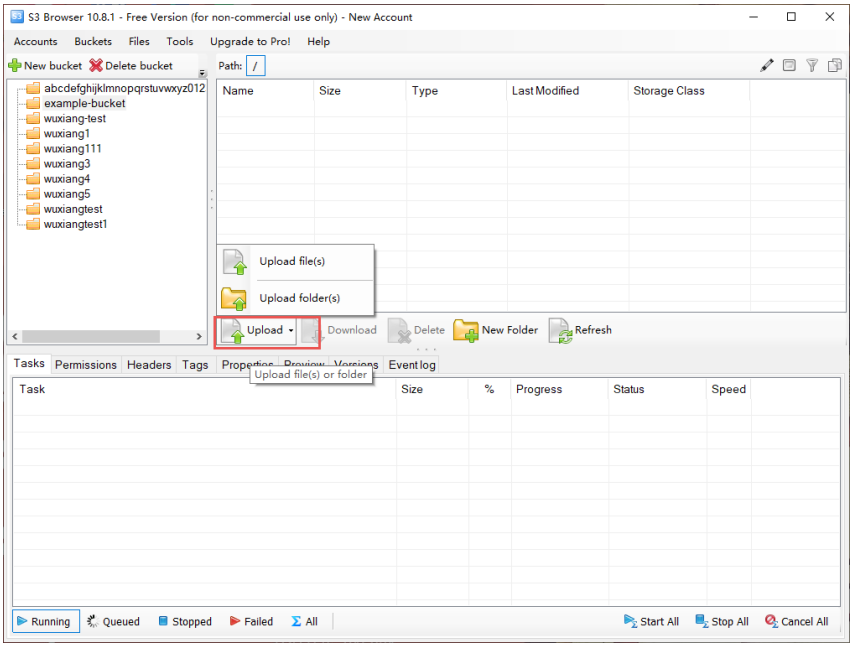
- 查看Bucket中内容

在工具窗口中的存储桶列表中选择需查看的存储桶，单击一下，左侧就会出现存储桶中的内容。

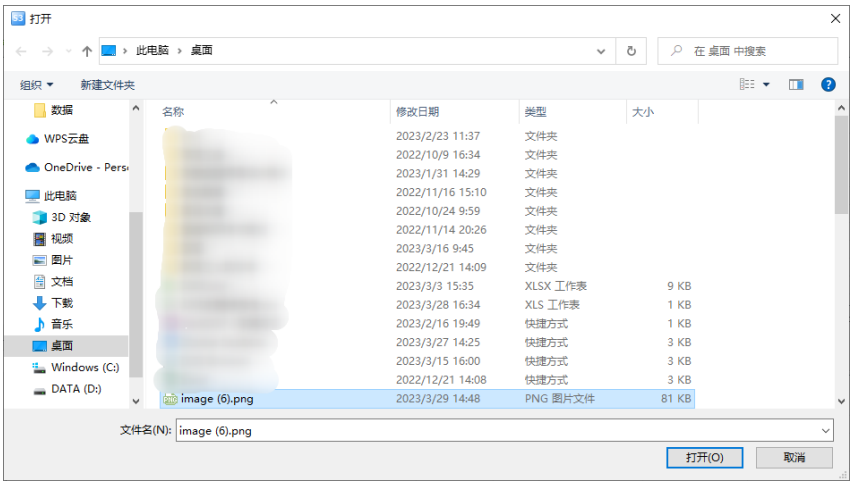
Object操作

- 上传Object

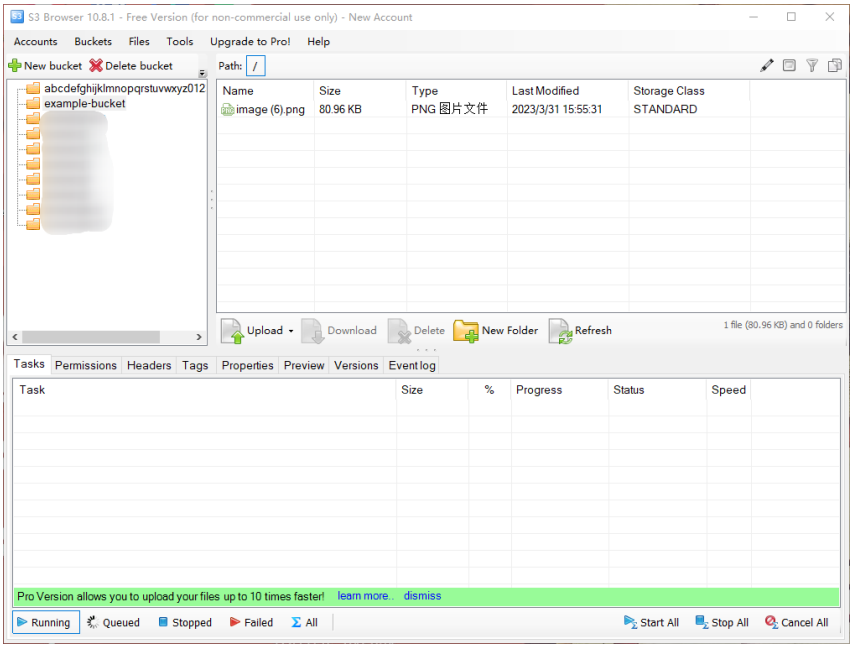
1. 在工具的窗口中，选择对象被复制后的目标存储桶，在窗口右侧单击图中标注的 Upload选项。支持文件和文件夹上传。



2. 弹窗之后，在本地选中需上传文件或者文件夹（本文以上传文件为例）后，单击**打开**。



3. 上传完成自动刷新文件列表。



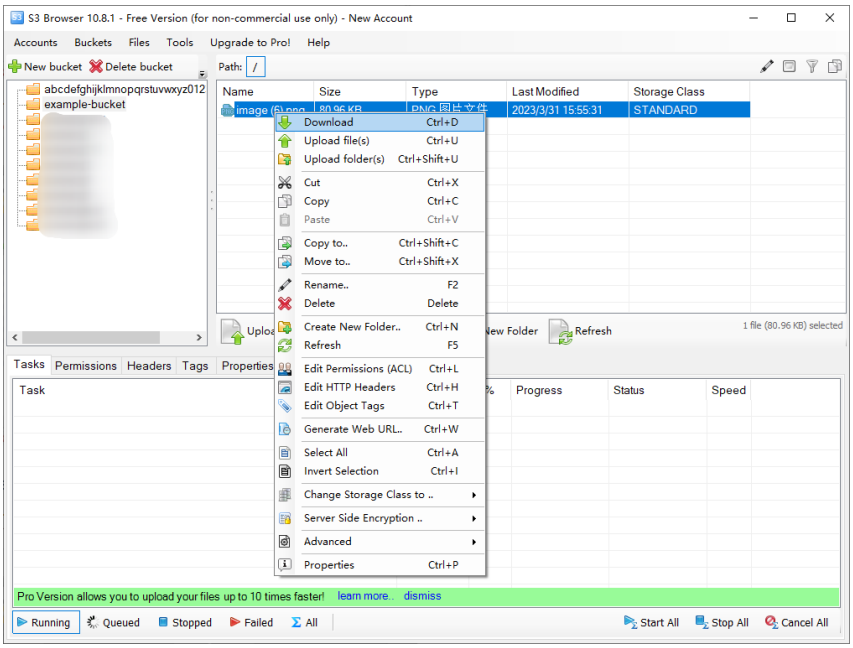
- **下载Object**

1. 在工具窗口的存储桶列表中，找到目标存储桶中需下载的对象。

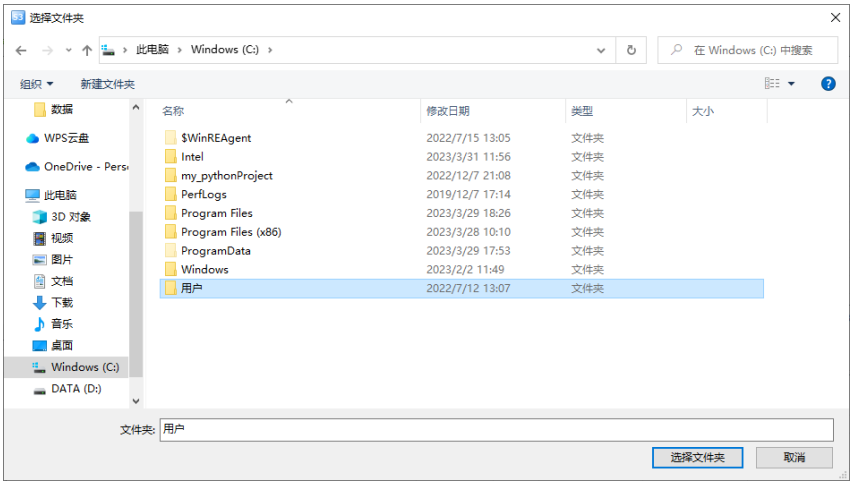
下载方式：

（1）选中需要下载的文件或者文件夹，点击图中的 Download 选项。

（2）选中需要下载的文件或者文件夹，右键选择 Download。

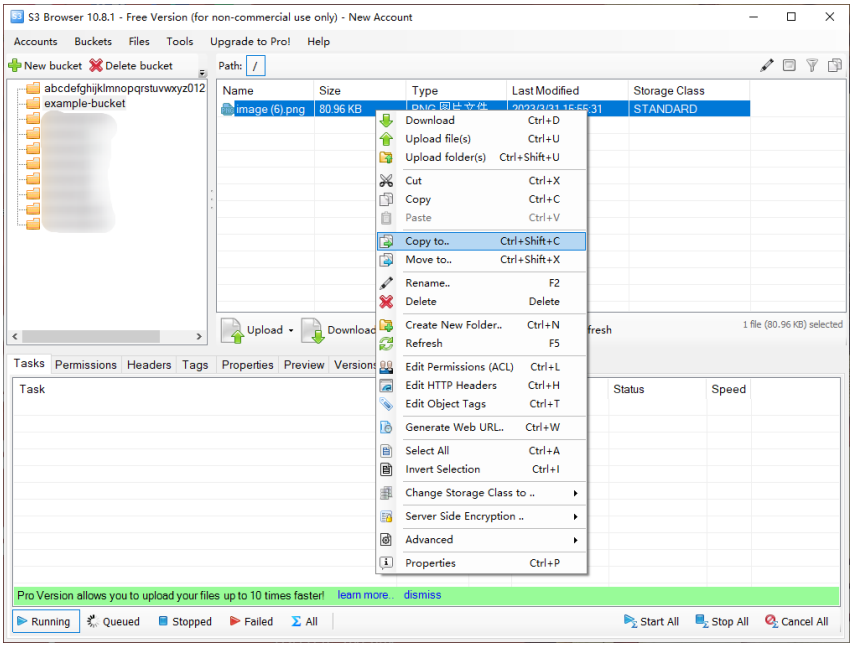


2. 弹窗之后，选择存储路径完成下载。

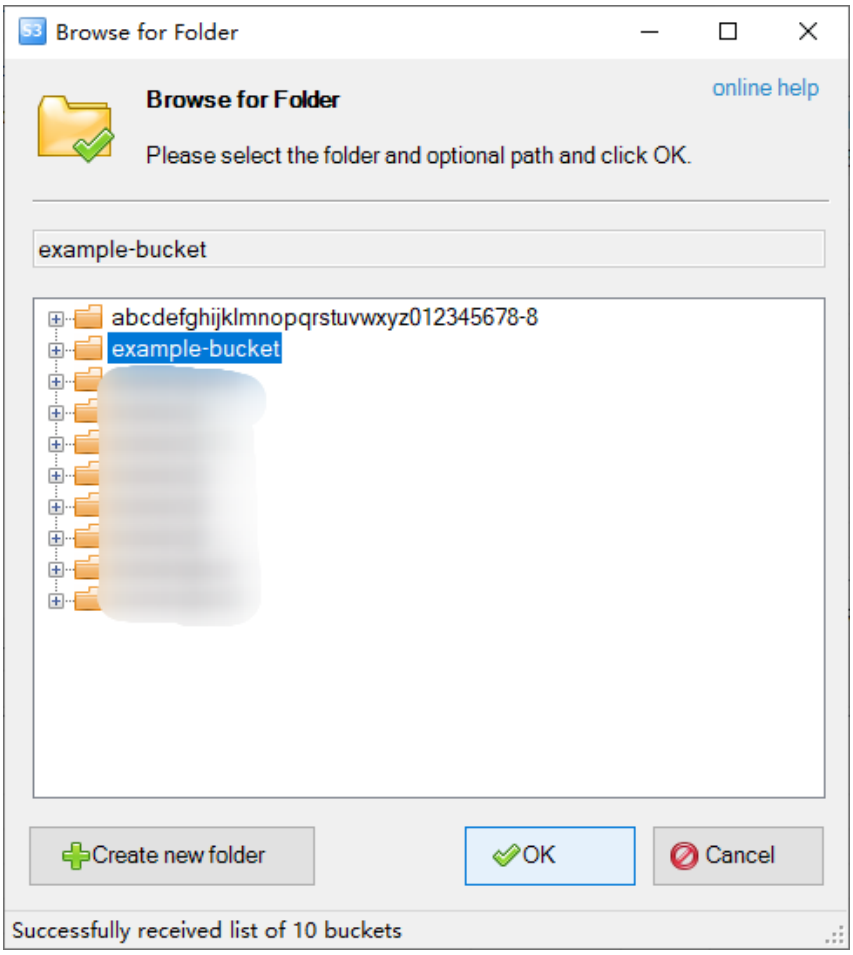


- **复制Object到存储桶**

1. 在工具窗口的存储桶列表中，找到存储桶中需复制的对象，右键选择Copy to。支持文件与文件夹复制。

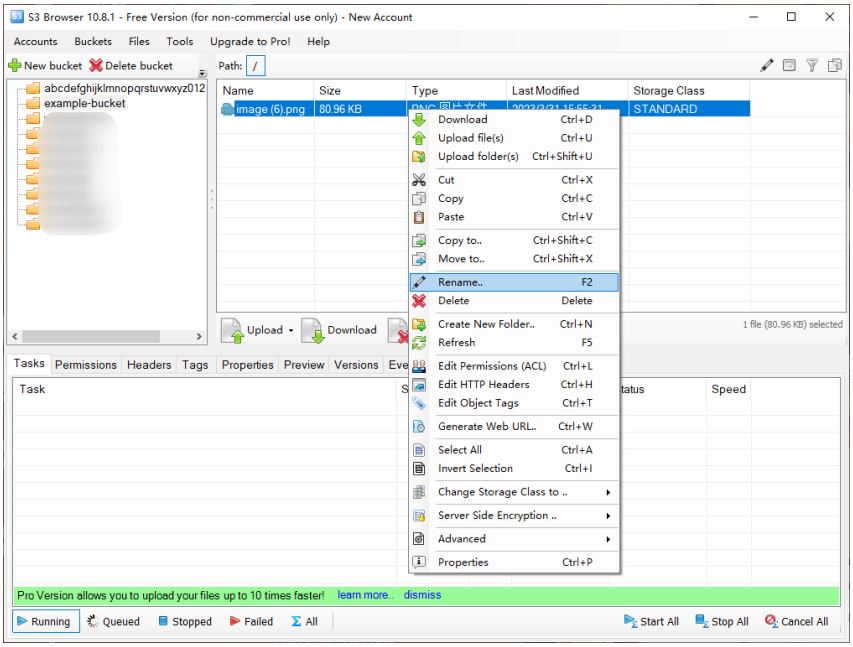


2. 选择复制对象的目标桶，单击 OK，即可完成复制对象到目标存储桶。

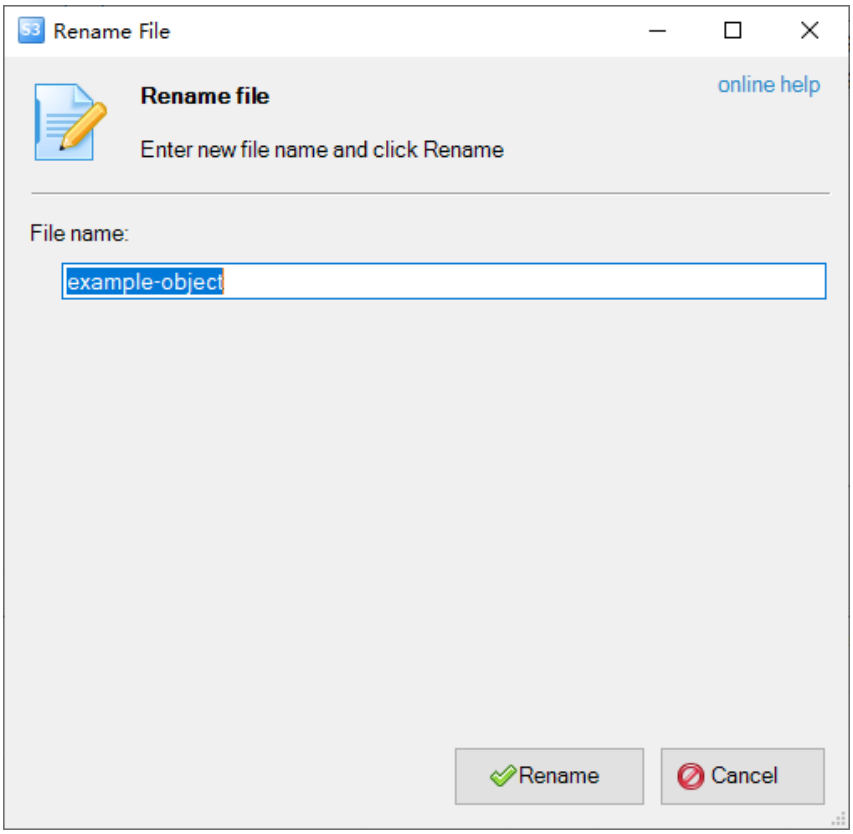


- 重命名Object

1. 在工具窗口的存储桶列表中，找到存储桶中需重命名的对象，右键选择Rename。

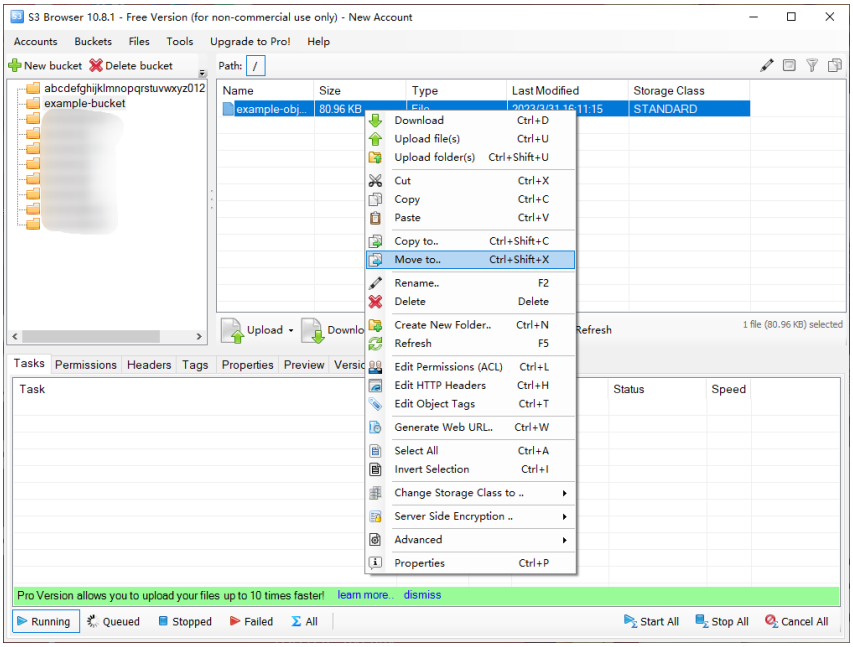


2. 在弹窗中输入新名称，例如 `example-object`，输入无误后，单击 `Rename` 即可完成重命名。



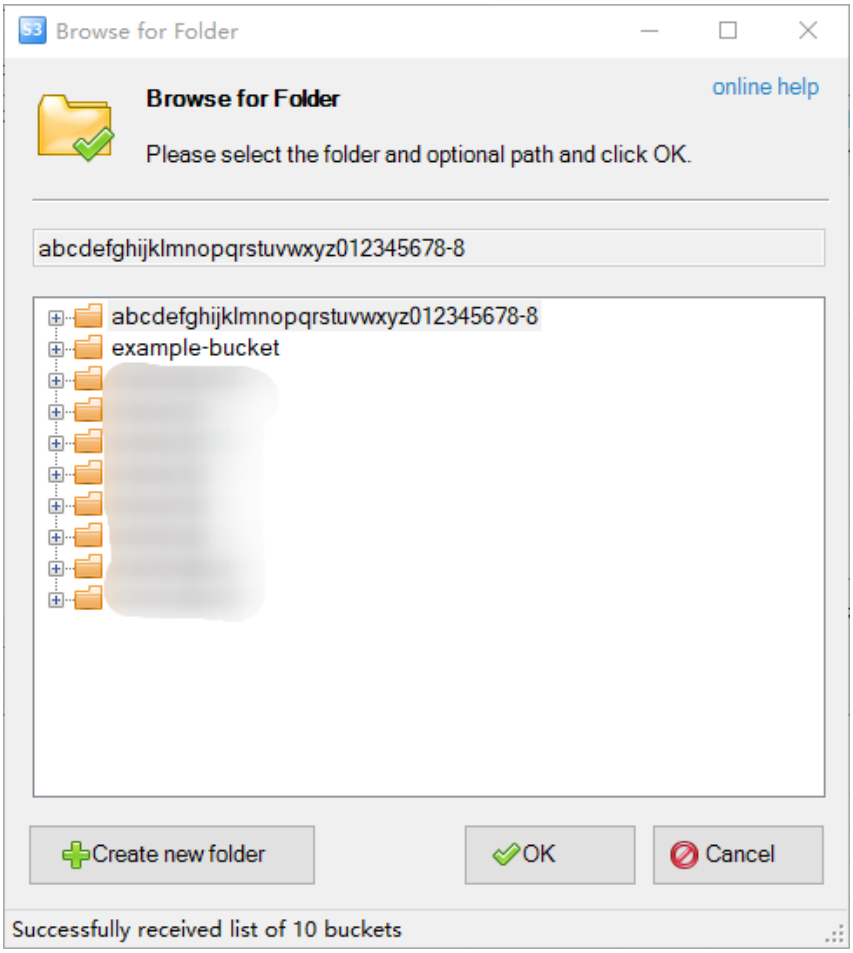
- **移动Object**

1. 在工具窗口的存储桶列表中，找到存储桶中需移动的对象，右键选择**Move to**。



2. 选择移动对象的目标桶，单击 OK，即可完成移动对象到目标存储桶。

注：此操作会将原来的对象删除。



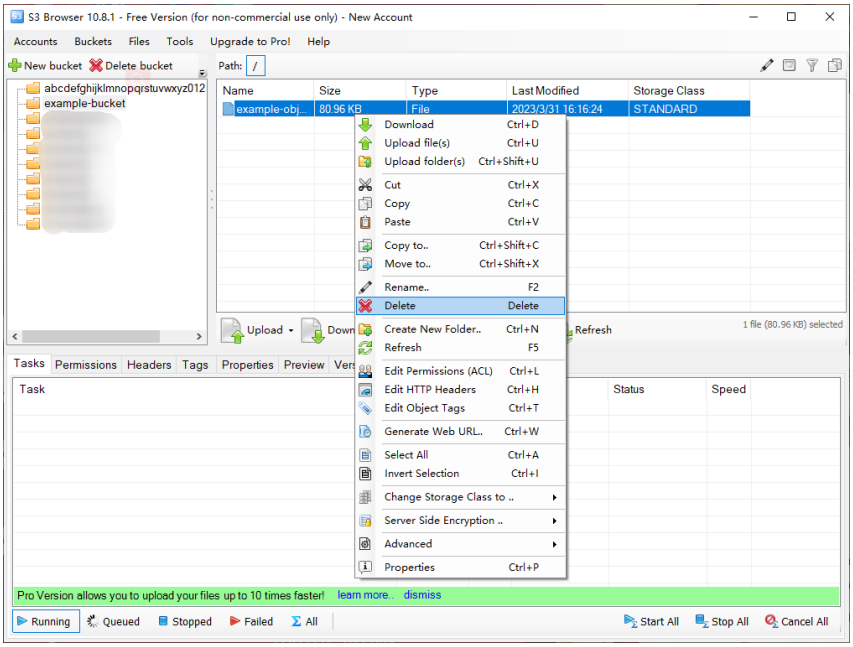
- **删除Object**

1. 在工具窗口的存储桶列表中，找到存储桶中需删除的对象。

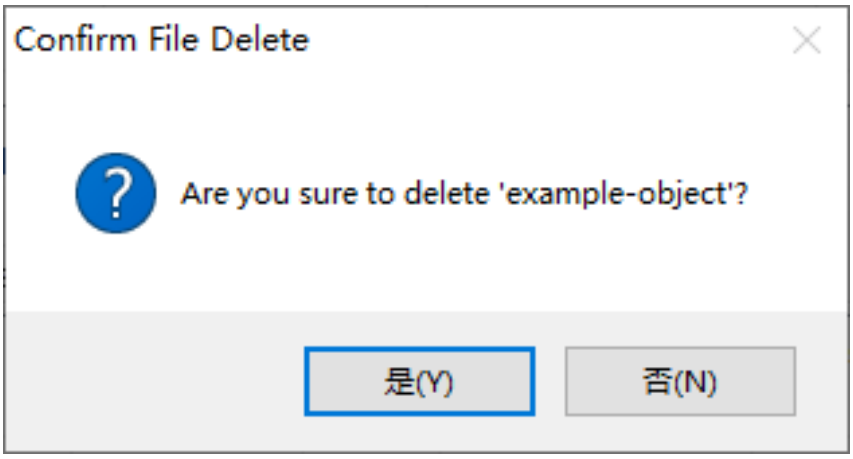
删除方式：

（1）选中需要删除的文件或者文件夹，点击图中的 Delete 选项。

（2）选中需要删除的文件或者文件夹，右键选择 Delete。

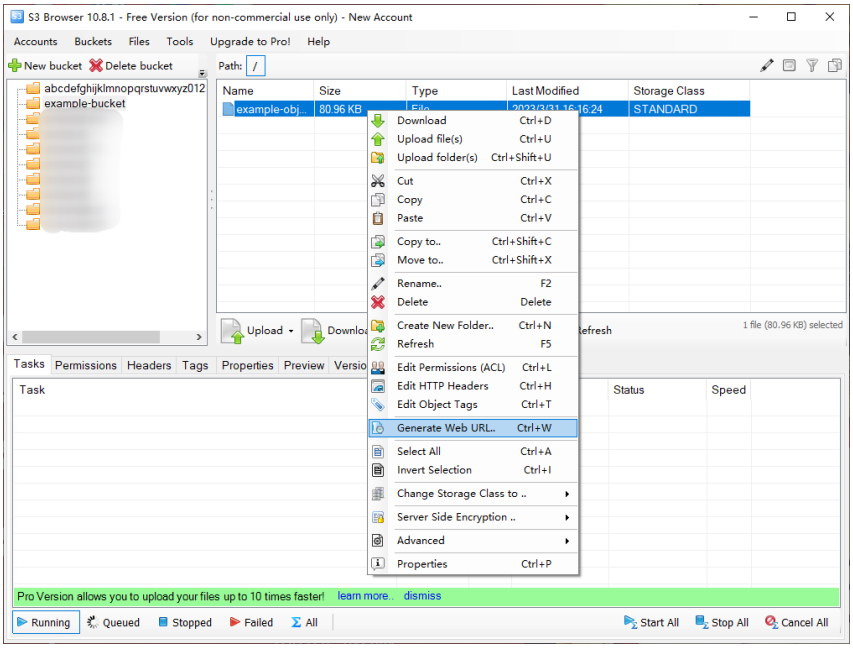


2. 确认弹窗信息，单击 **是（Y）** 即可完成对象删除操作。

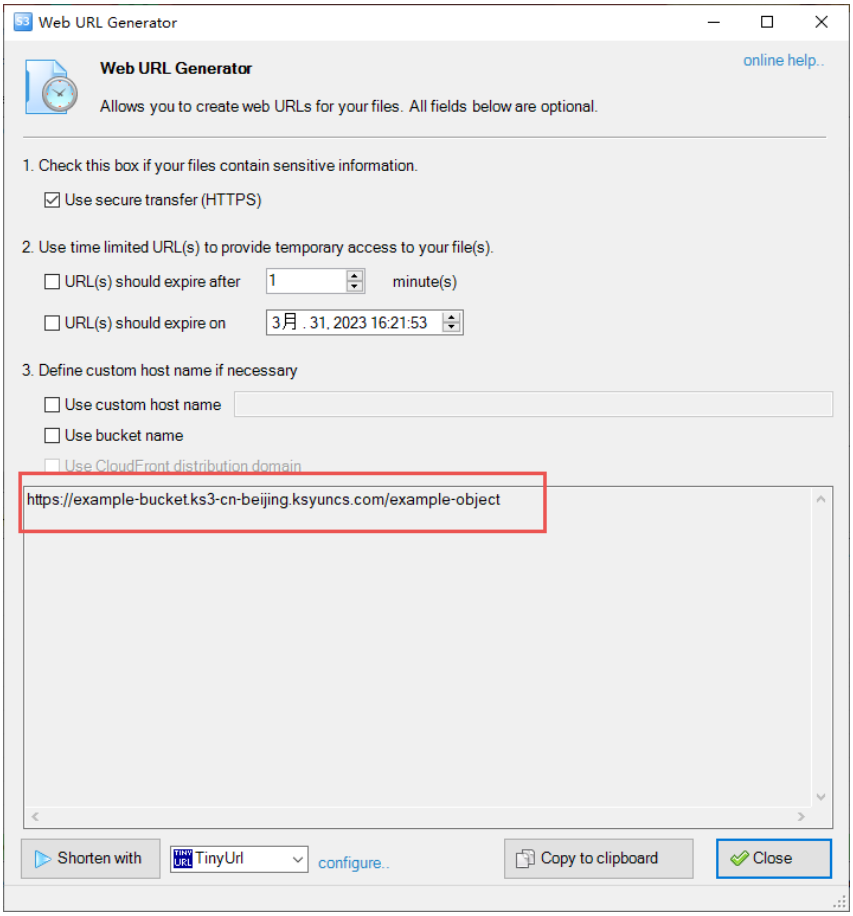


- **获取Obejct的URL**

1. 在工具窗口的存储桶列表中，找到存储桶中目标对象，右键单击Generate Web URL。



2. 弹出的窗口中，下图标注所示即为对象的URL。



MinIO Client

简介

Minio Client 简称mc，是minio服务器的客户端，对ls，cat，cp，mirror，diff，find等UNIX命令提供了一种替代方案，它支持文件系统和兼容Amazon S3的云存储服务（AWS Signature v2和v4）。

下载

- 官网地址：点击下载
- 安装包Windows：点击下载
- 安装包mac：点击下载
- 安装包linux：点击下载

安装

- **在Windows上安装**

  根据操作系统及架构，直接点击下载链接下载对应的版本。

- **在Linux上安装**

如需在 Linux系统上安装RPM格式的 MinIO Client，请运行以下命令：

```
dnf install https://dl.min.io/server/minio/release/linux-amd64/minio-20230324214123.0.0.x86_64.rpm
```

注：如需安装其他格式请在MinIO Client官网点击对应的下载链接。

- **在macOS上安装**

如需在 Linux系统上安装Homebrew格式的 MinIO Client，请运行以下命令：

```
brew install minio/stable/mc
mc alias set myminio/ http://MINIO-SERVER MYUSER MYPASSWORD
```

注：如需安装其他格式请在MinIO Client官网点击对应的下载链接。

初始化配置文件

打开cmd，执行以下命令初始化配置文件，其中和 需要填真实的值：

```
mc config host add ks3 https://ks3-cn-beijing.ksyuncs.com <AK> <SK> --api s3v4
```

注：配置文件命令格式：mc config host add [--api API-SIGNATURE]

参数说明：

名称	描述
ALIAS	服务别名
YOUR-S3-ENDPOINT	KS3 对外服务的访问域名
YOUR-ACCESS-KEY	简称AK，用于标识用户身份
YOUR-SECRET-KEY	简称SK，访问秘钥
--api API-SIGNATURE	签名方式，可选， s3v2：v2签名， s3v4(默认) ：v4签名

操作

- Bucket操作

以下命令用于查看bucket列表：

```
mc ls ks3
```

以下命令用于查看某个bucket下的文件：

```
mc ls ks3/auto-test-bucket
```

以下命令用于创建bucket：

```
mc mb ks3/test-bucket
```

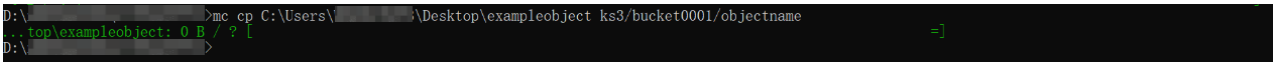
以下命令用于删除bucket：：

```
mc rm ks3/test-bucket
```

- Object操作

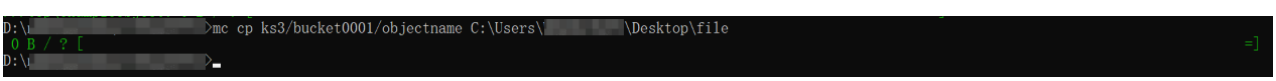
以下命令用于上传文件：

```
mc cp D:\test\demo.txt ks3/auto-test-bucket/demo.txt
```



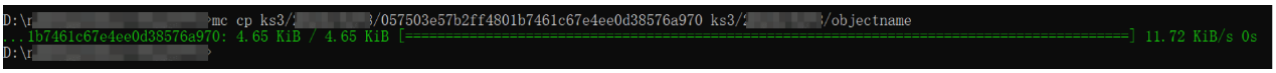
以下命令用于下载文件：

```
mc cp ks3/auto-test-bucket/demo.txt D:\test\demo.txt
```



以下命令用于复制文件：

```
mc cp ks3/auto-test-bucket/demo.txt ks3/auto-test-bucket/demo_copy.txt
```



以下命令用于删除文件：

```
mc rm ks3/auto-test-bucket/demo.txt
```

说明：

- S3支持路径（Path）请求风格和虚拟托管（Virtual Hosted）请求风格。基于安全考虑，KS3仅支持虚拟托管访问方式，虚拟托管请求风格是指将Bucket置于Host Header的访问方式。
- MinIO Client目前默认使用路径（Path）请求风格，如需使用虚拟托管（Virtual Hosted）请求风格，即三级域名，请采用以下命令：

  ```
  ./mc alias set ksyun https://ks3-cn-beijing-internal.ksyuncs.com AK SK --path off
  ```

  用户只需填写AK、SK即可。

Rclone

简介

Rclone是一个的命令行工具，用于管理云存储上的文件，支持在不同对象存储、网盘间同步上传、下载数据。

下载

- 官网地址：点击下载
- 安装包Windows：点击下载
- 安装包linux：点击下载
- 安装包mac：点击下载

安装

- **在Windows上安装**

根据操作系统及架构，直接点击下载链接下载对应的版本，下载后将安装包进行解压。

- **在Linux/macOS/BSD上安装**

要在 Linux/macOS/BSD 系统上安装 rclone，请运行以下命令：

```
sudo -v ; curl https://rclone.org/install.sh | sudo bash
```

配置文件

打开cmd，输入 rclone config 命令设置基本参数，根据需求按照命令提示进行输入，以下为配置示例：

```
D:\tools\rclone-v1.61.1-windows-amd64>rclone config
No remotes found, make a new one?
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n

Enter name for new remote.
name> ks3

Option Storage.
Storage> 5

Option provider.
provider> 25

Option env_auth.
env_auth> 1

Option access_key_id.
access_key_id> AK

```

```
Option secret_access_key.
secret_access_key> SK

Option region.
region> BEIJING

Option endpoint.
endpoint> ks3-cn-beijing.ksyuncs.com

Option location_constraint.
location_constraint>

Option acl.
acl> 1

// 访问KS3必须使用三级域名，'Edit advanced config?' 命令必须选择Yes才能配置KS3域名
Edit advanced config?
y) Yes
n) No (default)
y/n> y

Option bucket_acl.
bucket_acl> 1

// 除以下两项外，其余配置直接回车使用默认值即可

// 是否使用二级域名  true(默认) 使用二级域名    false 使用三级域名   此处必须填false
Option force_path_style.
force_path_style> false

// 是否使用v2签名   true 使用v2签名    false(默认) 使用v4签名
Option v2_auth.
v2_auth>

// 当第二次出现Edit advanced config?命令时，选择No即可
Edit advanced config?
y) Yes
n) No (default)
y/n> n

Configuration complete.

Keep this "ks3" remote?
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d> y

// 完成并退出基本参数配置
Current remotes:
e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> q

D:\tools\rclone-v1.61.1-windows-amd64>
```

以下命令行用于显示配置文件的路径：

```
rclone config file
```

以下命令行用于显示配置文件的信息：

```
rclone config show
```

操作

本文档的例子如无特别说明，均假设在Windows系统。

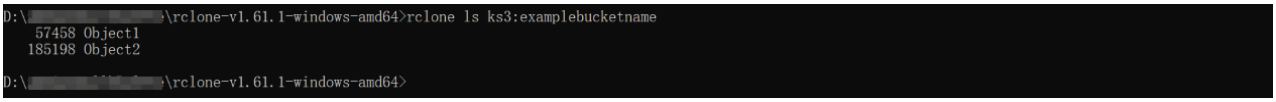
- Bucket操作

以下命令用于查看Bucket列表：

```
rclone lsd ks3:
```

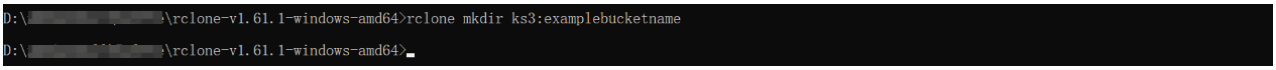
以下命令用于查看某个Bucket下的文件：

```
rclone ls ks3:test-bucket
```



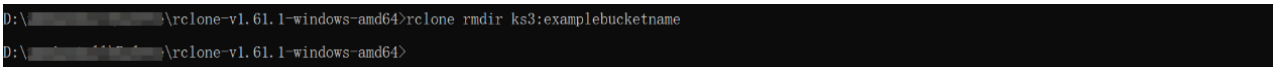
以下命令用于创建Bucket：

```
rclone mkdir ks3:test-bucket
```



以下命令用于删除Bucket：

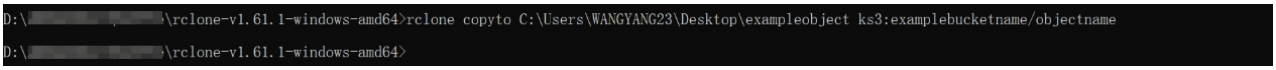
```
rclone rmdir ks3:test-bucket
```



- Object操作

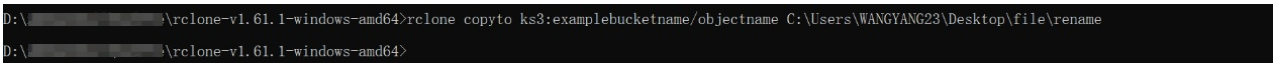
以下命令用于上传Object：

```
rclone copyto D:\test\demo.txt ks3:auto-test-bucket/demo.txt
```



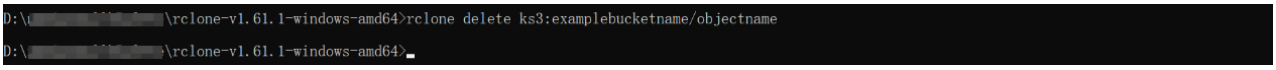
以下命令用于下载Object：

```
rclone copyto ks3:auto-test-bucket/demo.txt D:\test\demo.txt
```



以下命令用于删除Object：

```
rclone delete ks3:auto-test-bucket/demo.txt
```



S3 cmd

简介

S3cmd 是免费的命令行工具和客户端，用于在 Amazon S3 和其他兼容 S3 协议的对象存储中上传、下载和管理数据。 本文主要介绍如何使用 S3cmd 访问 KS3 上的文件。

下载

- 官网地址：点击下载

- 安装包：点击安装

  说明：

  - 使用该工具需要先安装 Python
  - 工具默认使用 V4 签名，如需使用 V2 签名，需要在命令后添加 --signature-v2
  - 全部 S3 cmd 命令行详见：点击查看

安装

使用该工具需要先安装 Python，安装命令如下，其中 setup.py 为 S3 cmd 的安装文件。

```
python setup.py install
```

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python setup.py install
Using xml.etree.ElementTree for XML processing
running install
running bdist_egg
running egg_info
writing s3cmd.egg-info\PKG-INFO
writing dependency_links to s3cmd.egg-info\dependency_links.txt
writing requirements to s3cmd.egg-info\requires.txt
writing top-level names to s3cmd.egg-info\top_level.txt
reading manifest file 's3cmd.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 's3cmd.egg-info\SOURCES.txt'
installing library code to build\bdist.win-amd64\egg
running install_lib
running build_py
creating build
creating build\lib
creating build\lib\S3
copying S3\AccessLog.py -> build\lib\S3
copying S3\ACL.py -> build\lib\S3
copying S3\BaseUtils.py -> build\lib\S3
copying S3\BidirMap.py -> build\lib\S3
copying S3\CloudFront.py -> build\lib\S3
copying S3\Config.py -> build\lib\S3
copying S3\ConnMan.py -> build\lib\S3
copying S3\Crypto.py -> build\lib\S3
copying S3\Custom_httplib27.py -> build\lib\S3
copying S3\Custom_httplib3x.py -> build\lib\S3
```

配置

1. 设置使用的AK/SK及默认Region。

```
Access key and Secret key are your identifiers for Amazon S3. Leave them empty for using the env variables.
Access Key: AKLT[redacted]
Secret Key: OPHo[redacted]
Default Region [US]: BEIJING
```

2. 设置访问的Endpoint，KS3 Endpoint信息详见Endpoint及Region说明。

```
Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.
S3 Endpoint [s3.amazonaws.com]: ks3-cn-beijing.ksyuncs.com
```

3. 设置访问方式：%(bucket)s.ks3-cn-beijing.ksyuncs.com。

```
Use "%(bucket)s.s3.amazonaws.com" to the target Amazon S3. "%(bucket)s" and "%(location)s" vars can be used
if the target S3 system supports dns based buckets.
DNS-style bucket+hostname:port template for accessing a bucket [%(bucket)s.s3.amazonaws.com]: %(bucket)s.ks3-cn-beijing.ksyuncs.com
```

4. 如不需要进行设置，可使用默认配置，不填写。

```
Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password:
Path to GPG program:
```

5. 是否以HTTPS协议进行访问。

```
When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]: y
```

6. 设置HTTP代理服务名称及对应Port，如不需要进行设置，可不填写。

```
On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:
```

7. 设置展示及确认，确认后，会进行配置及保存。

```
New settings:
  Access Key: AKLT
  Secret Key: OPHo
  Default Region: BEIJING
  S3 Endpoint: ks3-cn-beijing.ksyuncs.com
  DNS-style bucket+hostname:port template for accessing a bucket: %(bucket)s.ks3-cn-beijing.ksyuncs.com
  Encryption password:
  Path to GPG program: None
  Use HTTPS protocol: True
  HTTP Proxy server name:
  HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] y
Please wait, attempting to list all buckets...
Success. Your access key and secret key worked fine :-)

Now verifying that encryption works...
Not configured. Never mind.

Save settings? [y/N] y
Configuration saved to 'C:\Users\                \Roaming\s3cmd.ini'
```

8. 完整配置示例展示。

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd --configure
ERROR: Option --preserve is not yet supported on MS Windows platform. Assuming --no-preserve.
ERROR: Option --progress is not yet supported on MS Windows platform. Assuming --no-progress.

Enter new values or accept defaults in brackets with Enter.
Refer to user manual for detailed description of all options.

Access key and Secret key are your identifiers for Amazon S3. Leave them empty for using the env variables.
Access Key: AKLT7
Secret Key: OPHoE
Default Region [US]: BEIJING

Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.
S3 Endpoint [s3.amazonaws.com]: ks3-cn-beijing.ksyuncs.com

Use "%(bucket)s.s3.amazonaws.com" to the target Amazon S3. "%(bucket)s" and "%(location)s" vars can be used
if the target S3 system supports dns based buckets.
DNS-style bucket+hostname:port template for accessing a bucket [%(bucket)s.s3.amazonaws.com]: %(bucket)s.ks3-cn-beijing.ksyuncs.com

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password:
Path to GPG program:

When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]: y

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:

New settings:
  Access Key: AKLT
  Secret Key: OPHo
  Default Region: BEIJING
  S3 Endpoint: ks3-cn-beijing.ksyuncs.com
  DNS-style bucket+hostname:port template for accessing a bucket: %(bucket)s.ks3-cn-beijing.ksyuncs.com
  Encryption password:
  Path to GPG program: None
  Use HTTPS protocol: True
  HTTP Proxy server name:
  HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] y
Please wait, attempting to list all buckets...
Success. Your access key and secret key worked fine :-)

Now verifying that encryption works...
```

常见操作

常见Bucket操作

- **创建Bucket**

以下命令用于创建Bucket：

```
python s3cmd mb s3://{your-bucket}
```

（1）创建成功示例：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd mb s3://s3-cmd
Bucket 's3://s3-cmd/' created
```

（2）如需要创建的Bucket已存在，则会进行报错：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd mb s3://s3-cmd
ERROR: Bucket 's3-cmd' already exists
ERROR: S3 error: 409 (BucketAlreadyExists): The requested bucket name is not available. The bucket namespace is shared b
y all users of the system. Please select a different name and try again.
```

- **删除Bucket**

以下命令用于删除Bucket：

```
python s3cmd rb s3://{your-bucket}
```

（1）删除成功示例：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd rb s3://s3-cmd
Bucket 's3://s3-cmd/' removed
```

（2）如需要删除的Bucket已不存在，则会进行报错提示：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd rb s3://s3-cmd
ERROR: S3 error: 404 (NoSuchBucket): The specified bucket does not exist.
```

- **查看指定Bucket下的文件**

以下命令用于查看指定Bucket下的文件：

```
python s3cmd ls s3://{your-bucket}
```

（1）执行成功示例：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd ls s3://s3-cmd
2023-04-02 10:43         2238  s3://s3-cmd/CRR-1.txt
2023-04-02 10:43            7  s3://s3-cmd/CRR-2.txt
2023-04-02 10:43            6  s3://s3-cmd/DELETE-1.txt
2023-04-02 10:43            6  s3://s3-cmd/DELETE-2.txt
2023-04-02 10:43            6  s3://s3-cmd/MIGRATION-1.txt
2023-04-02 10:43            6  s3://s3-cmd/MIGRATION-2.txt
2023-04-02 10:43            6  s3://s3-cmd/MIGRATION-3.txt
```

（2）如需要List的Bucket内无文件，则没有文件可进行展示：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd ls s3://s3-cmd

D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>
```

（3）如Bucket不存在，则会进行报错提示：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd ls s3://s3-bucket-not-exist
ERROR: Bucket 's3-bucket-not-exist' does not exist
ERROR: S3 error: 404 (NoSuchBucket): The specified bucket does not exist.
```

- **查看Bucket列表**

以下命令用于查看查看Bucket列表：

```
python s3cmd ls
```

（1）List成功，Bucket会按照名称字母序列号进行排序：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd ls
2021-09-15 12:03  s3://2
2022-12-08 03:45  s3://2
2021-11-12 04:12  s3://b
2023-02-27 07:27  s3://b
2023-02-16 10:10  s3://b
2020-01-08 10:27  s3://b
2020-10-14 09:36  s3://c
```

常见Object操作

- 上传Object

以下命令用于上传Object：

python s3cmd python s3cmd put {D:\your-local-dir\file.txt} s3://{your-bucket/file.txt}

（1）上传成功：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd put D:\S3-Tools\file.txt s3://s3-cmd/file.txt
WARNING: Module python-magic is not available. Guessing MIME types based on file extensions.
upload: 'D:\S3-Tools\file.txt' -> 's3://s3-cmd/file.txt' (6 bytes in 0.3 seconds, 22.11 B/s) [1 of 1]
```

（2）上传后的文件在控制台也可以进行查看：

- 下载Object

以下命令用于下载Object：

python s3cmd python s3cmd get s3://{your-bucket/file.txt} [D:\your-local-dir\file.txt]

（1）下载成功。如不输入下载目标地址，则会下载至相对目录下，下图所示为下载至D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0下：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd get s3://s3-cmd/file.txt
download: 's3://s3-cmd/file.txt' -> '.\file.txt' (6 bytes in 0.1 seconds, 78.68 B/s)
```

（2）下载成功。如输入下载目标地址，则会下载至指定目录下，并可进行重命名。下图所示为下载至D:\，同时修改文件名称为：file-rename.txt：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd get s3://s3-cmd/file.txt D:\file-rename.txt
download: 's3://s3-cmd/file.txt' -> 'D:\file-rename.txt' (6 bytes in 0.1 seconds, 48.64 B/s)
```

- 复制Object

以下命令用于复制Object：

python s3cmd cp s3://{your-source-bucket/file.txt} s3://{your-destination-bucket/file.txt}

（1）复制成功，由Bucket s3-cmd复制至s3-cmd-dest，并将文件重新命名为file-copy.txt：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd cp s3://s3-cmd/file.txt s3://s3-cmd-dest/file-copy.txt
remote copy: 's3://s3-cmd/file.txt' -> 's3://s3-cmd-dest/file-copy.txt'  [1 of 1]
```

（2）复制后的文件也可以在控制台进行查看：

注意：Copy文件有一些使用约束，详细规则请查看 PUT Object Copy文档。

- 删除Object

以下命令用于删除Object：

```
python s3cmd del s3://{your-bucket/file.txt}
```

（1）删除成功：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd del s3://s3-cmd/file.txt
delete: 's3://s3-cmd/file.txt'
```

（2）如删除Bucket不存在的文件，则会进行报错提示：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd del s3://s3-cmd/file.txt
ERROR: S3 error: 404 (NoSuchKey): The specified key does not exist.
```

常见问题

S3 cmd安装完成后，可输入s3cmd --configure查看是否已安装成功，是否有其他需要的安装等。

（1）如下图所示，因缺少python-dateutil无法执行操作：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>python s3cmd --configure

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ImportError trying to import dateutil.parser.
Please install the python dateutil module:
$ sudo apt-get install python-dateutil
  or
$ sudo yum install python-dateutil
  or
$ pip install python-dateutil
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

（2）可根据提示进行相关内容的安装：

```
D:\S3-Tools\s3cmd-2.3.0\s3cmd-2.3.0>pip install python-dateutil
Collecting python-dateutil
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 3.0 kB/s
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil
ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip will chang
e the way that it resolves dependency conflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes the defau
lt.

s3cmd 2.3.0 requires python-magic, which is not installed.
Successfully installed python-dateutil-2.8.2 six-1.16.0
WARNING: You are using pip version 20.2.3; however, version 23.0.1 is available.
You should consider upgrading via the 'd:\program files\python\python39\python.exe -m pip install --upgrade pip' command
.
```