目录

数据迁移

数据迁移

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。

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Up-tool数据迁移工具

KS3之间数据迁移

直接在控制台操作或使用命令行工具、Up-tool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

第三方数据源迁移至KS3

- Up-tool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。Up-tool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，此时可利用镜像回源功能实现。

- 将第三方存储T0之前的数据全量迁移至KS3
- 数据迁移完成后，在KS3上配置镜像回源到第三方存储
- 用户业务切换到KS3，记此时时间为T1
- 用户将T0至T1时间段内新增或改动的文件，使用KS3 Up-tool或其他工具上传到KS3
- 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，通过内网告诉网络，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，通过内网告诉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

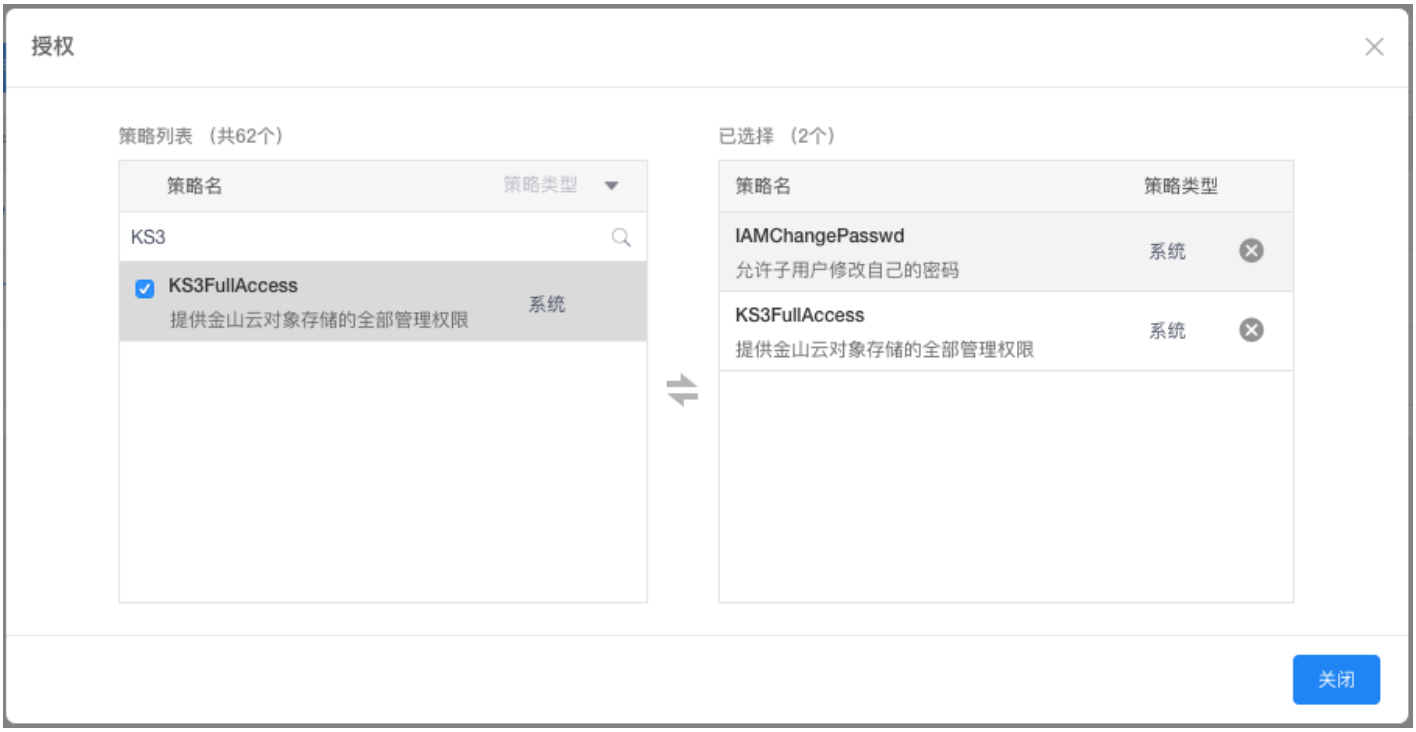
如果用户担心主账户的AKSK泄露，最安全的策略是不使用主账号的AK/SK来访问KS3，而是创建一个子用户，并赋予这个子用户足够的权限，使用这个子用户的AK/SK这样可以规避AccessKey或者密码泄露导致的问题。具体操作步骤如下：

1， 在控制台上操作请点击进入【身份与管理】->【用户管理】，进入用户管理界面

2， 点击“+新建用户”按钮，勾选“为该用户生成AccessKey”。

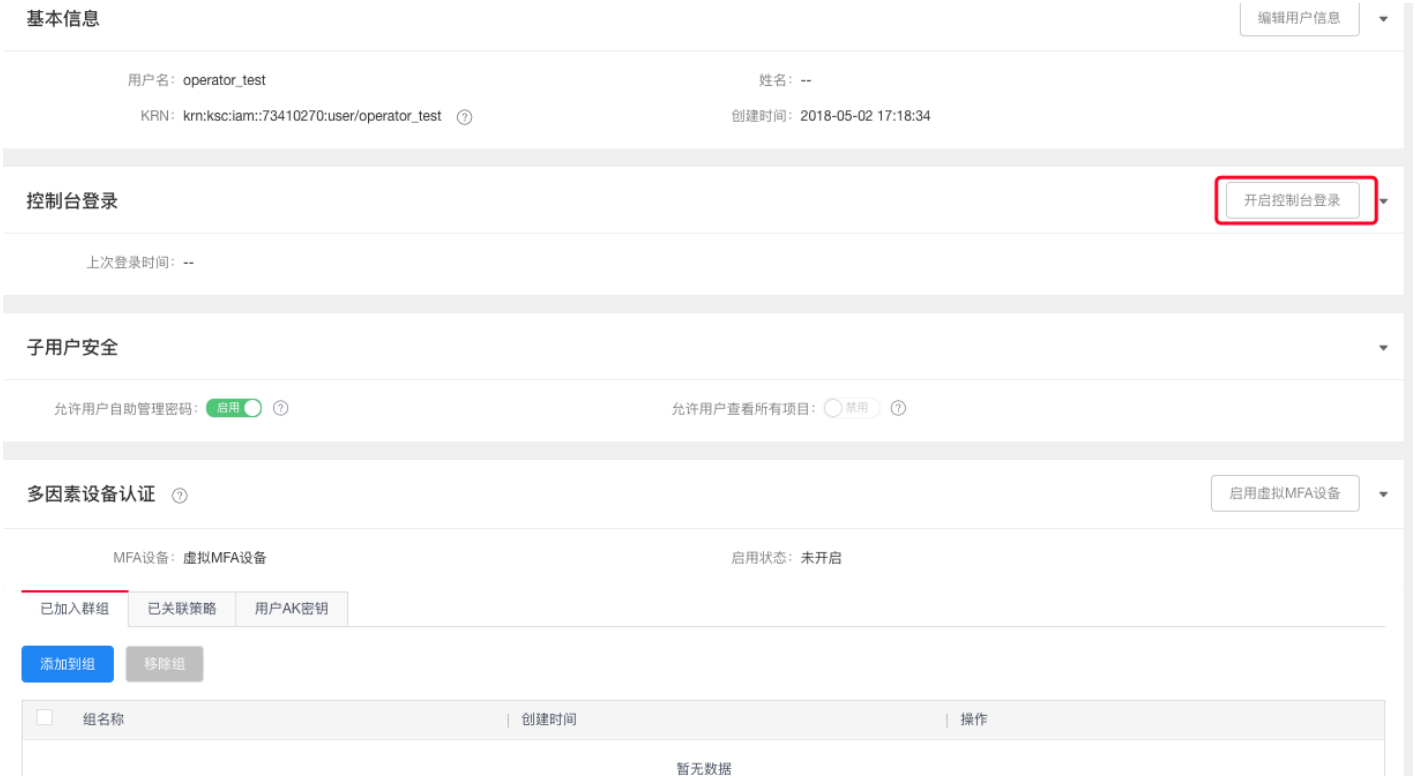
3， 生成该账号的AccessKey，一定要在这一步保存下来用于后续的访问。

4， 返回用户管理界面，这里显示一个名为operator_test的账号已创建。创建完成之后，该子账号还是没有任何权限，点击右边的"授权"链接，给该账号赋予KS3FullAccess系统权限。



5， 使用operator_test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KS3的相关资源了。

6，同时也可以通过子用户登录控制台。点击子用户的链接，进入子用户设置页面，开启控制台登录功能。



7，进入控制台登录界面，点击IAM账号链接，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，在控制台上操作请点击进入【身份与管理】->【用户管理】，进入用户管理界面

2，点击“+新建用户”按钮，勾选“为该用户生成AccessKey”。

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，返回用户管理界面，这里显示一个名为ram_test的账号已创建。创建完成之后，该子账号还是没有任何权限，点击右边的"授权"链接，给该账号赋予KS3ReadOnlyAccess系统权限。

5，使用ram_test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KS3的相关资源了。

6，也可以使用SDK，在SDK配置文件中写入ram_test这个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，在控制台上操作请点击进入【身份与管理】->【用户管理】，进入用户管理界面，创建rd_user和op_user，并生成对应的AKSK。

2，在控制台上操作请点击进入【身份与管理】->【策略管理】，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，“新建”按钮策略按钮，选择“按策略语法创建”，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、在控制台上操作请点击进入【身份与管理】->【策略管理】，进入策略管理界面。

2、点击“自定义策略”标签页，“新建”按钮策略按钮，选择“按业务权限创建”，选择服务类型为“KS3”，点击“下一步”，选择“上传文件”和“下载文件”两个业务权限，点击“下一步”，点击“创建策略”。

3、在控制台上操作请点击进入【身份与管理】->【用户管理】，进入用户管理界面，将应用服务器所对应的子用户授予新创建好的策略。

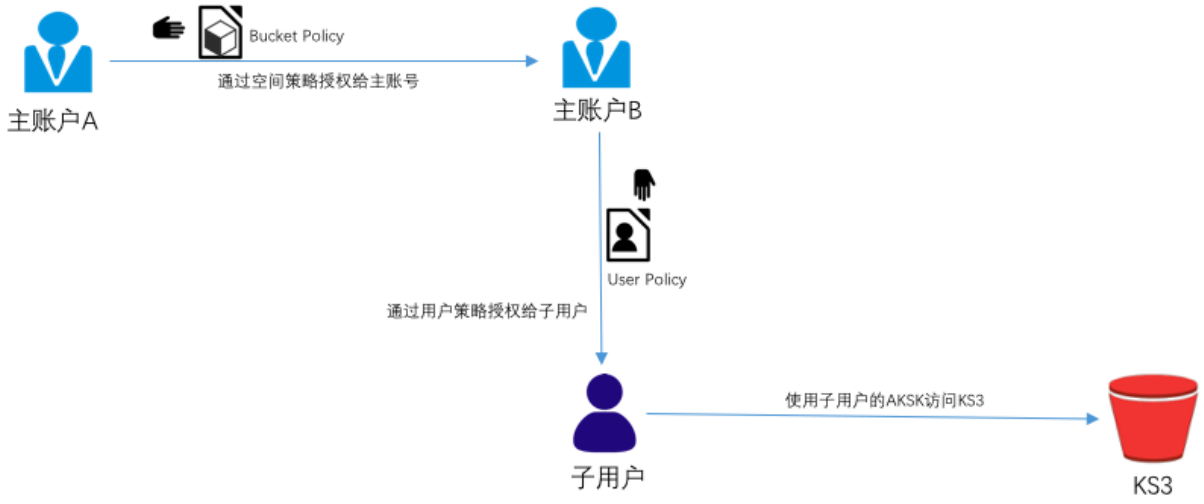
跨账户授权

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：

- 首先，需要将权限通过空间策略授予主账户B;
- 第2步，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，进入到A账户下需要授权的存储空间（Bucket），例如“exampleBucket”。进入【空间设置】->【空间策略】下，配置一条空间策略，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，新建一条自定义策略，授权策略语言如下：

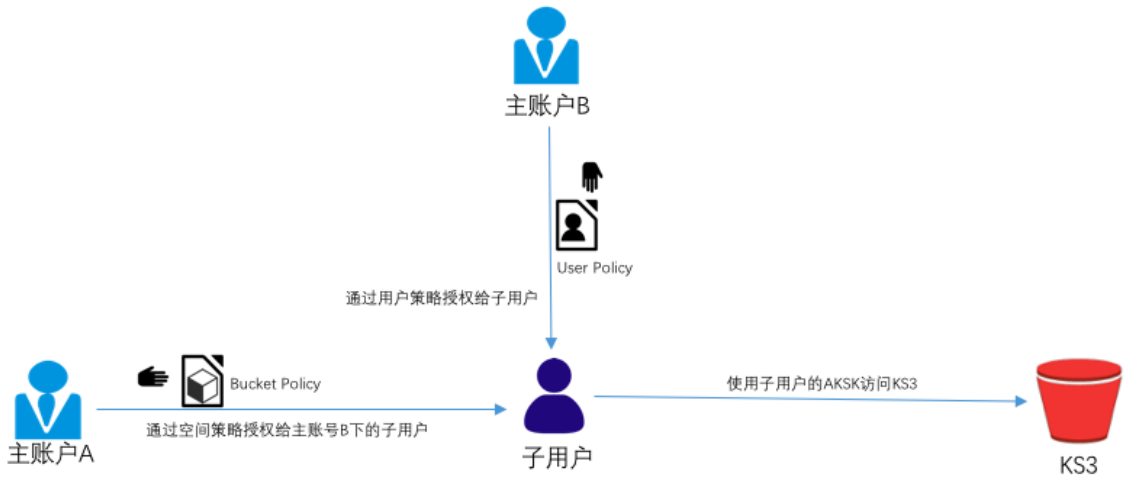
```
{
    "Version": "2015-11-01",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ks3:ListObject",
            "ks3:GetObject"
         ],
        "Resource":[
            "krn:ksc:ks3:::example_bucket",
            "krn:ksc:ks3:::example_bucket/*"
         ]
  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

3，进入到B账户下的【用户管理】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

无论是主账号还是IAM子用户都是可以长期正常使用的，如果对应的AKSK发生泄露之后如果无法及时解除权限的话会很危险。

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AKSK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角色的临时身份来完成：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 poicy）授权。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AKSK，去访问KS3了。 原理图如下：

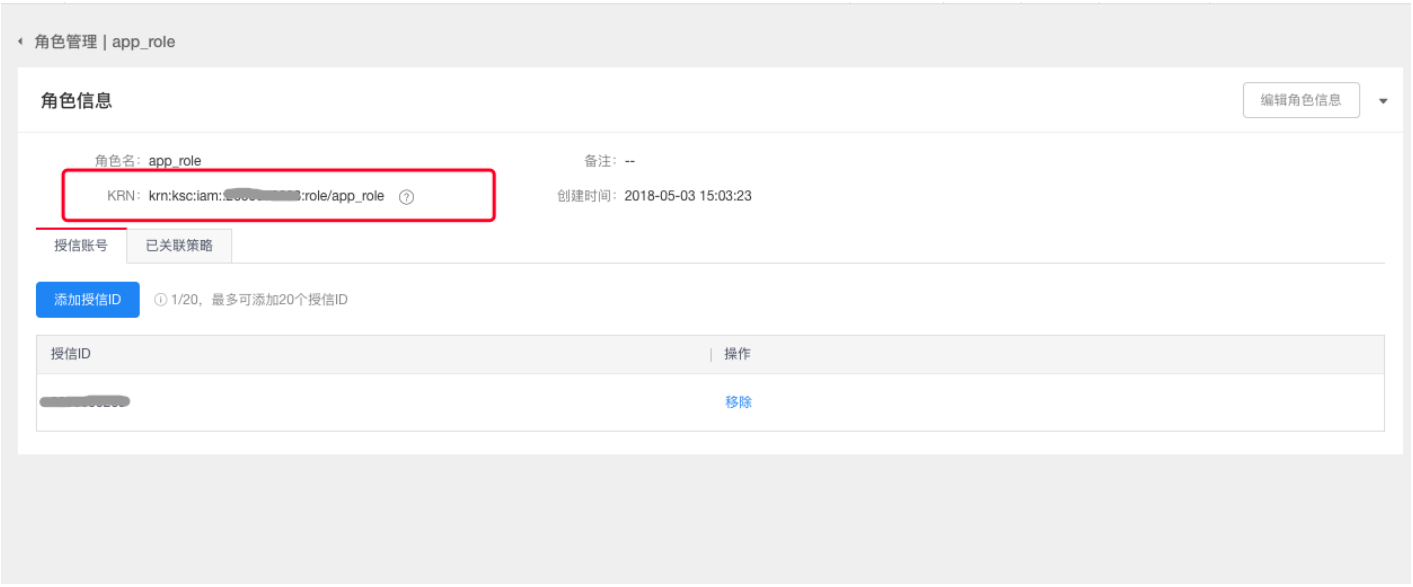
具体的步骤如下：

创建角色

1，在控制台上操作请点击进入【身份与管理】->【角色管理】，进入角色管理界面。

2，点击“新建角色”按钮，选择授信云账号，可以选择当前的账号，也可以选择其它云账号。我们创建一个名为“app_role”的角色，并选择当前的账号为授信账号。

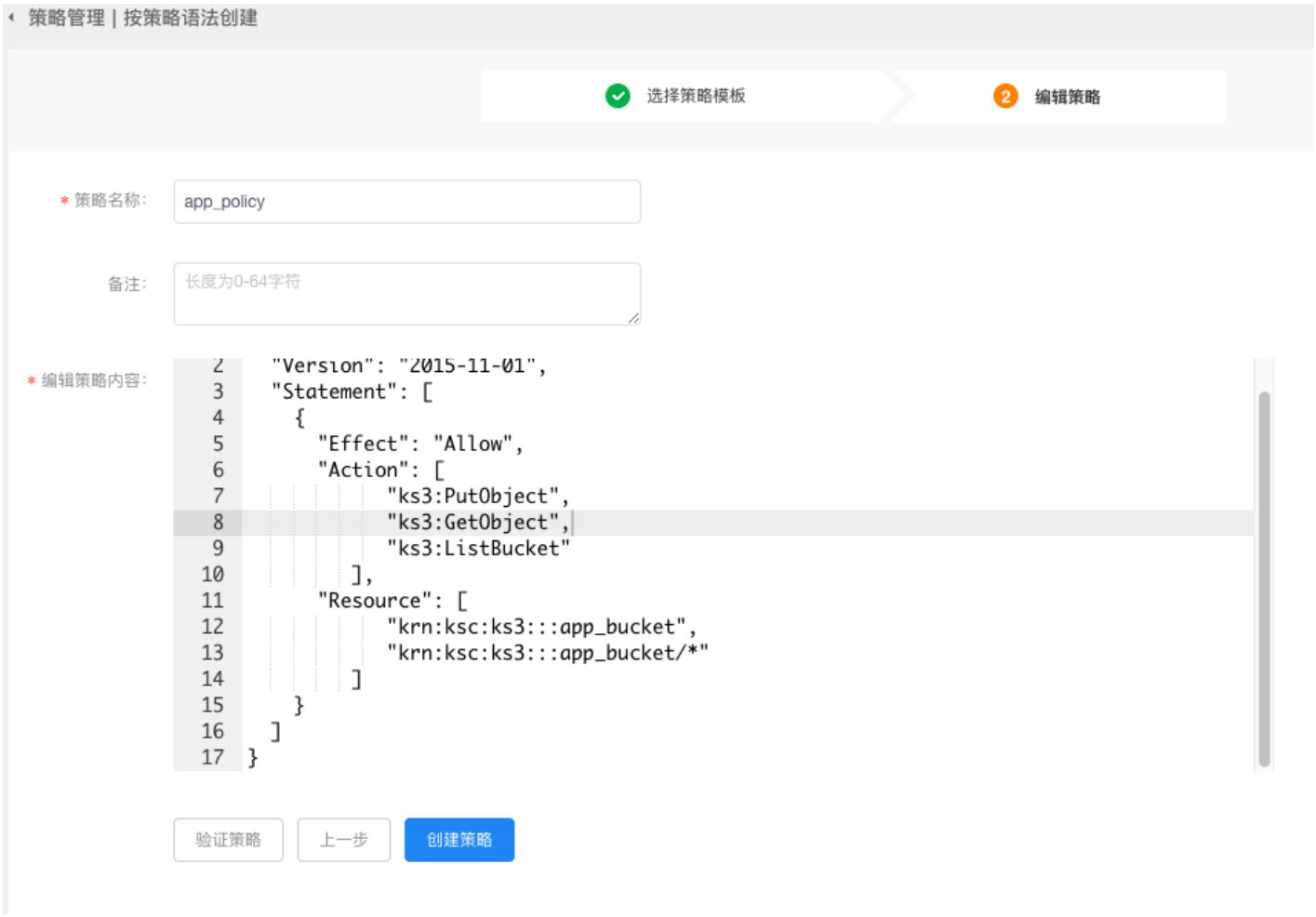
创建完毕后，进入角色页面，记录下角色app_role的KRN



3，创建完角色之后，角色是没有任何权限的，因此需要新建一个自定义的授权策略。授权策略如下：

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

如图所示：



这里表示的就是对app_bucket拥有写入文件、列出空间下文件、下载文件的权限。策略的名字定为app_policy。

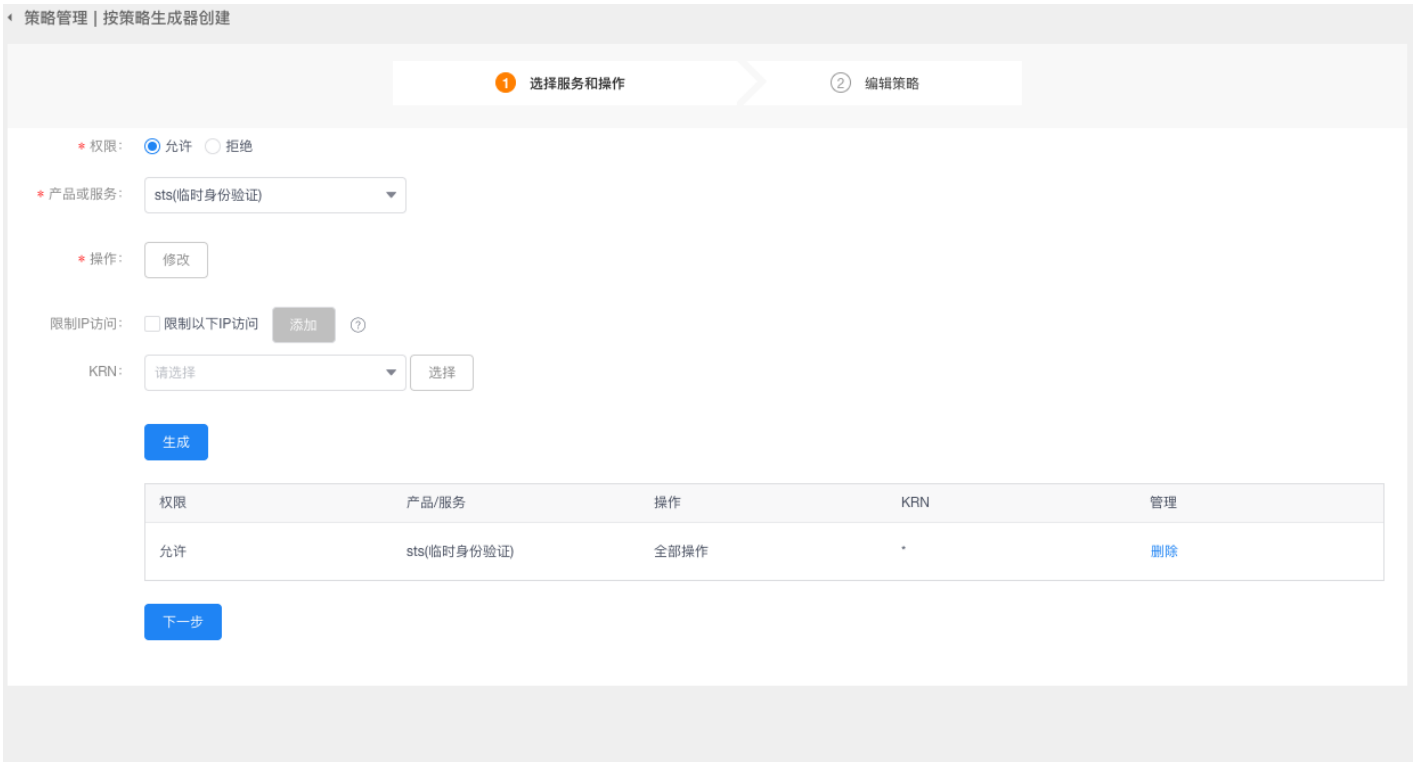
4，建立完成之后即可在角色管理里面给app_role添加上app_policy的自定义策略。

子用户扮演角色

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，我们进入【策略管理】页面，新建一个自定义的授权策略，选择“按策略生成器创建”：

2，服务选择“STS（临时身份验证）”，操作选择“AssumeRole”，点击“生成”，点击“确认”，进入编辑页面。



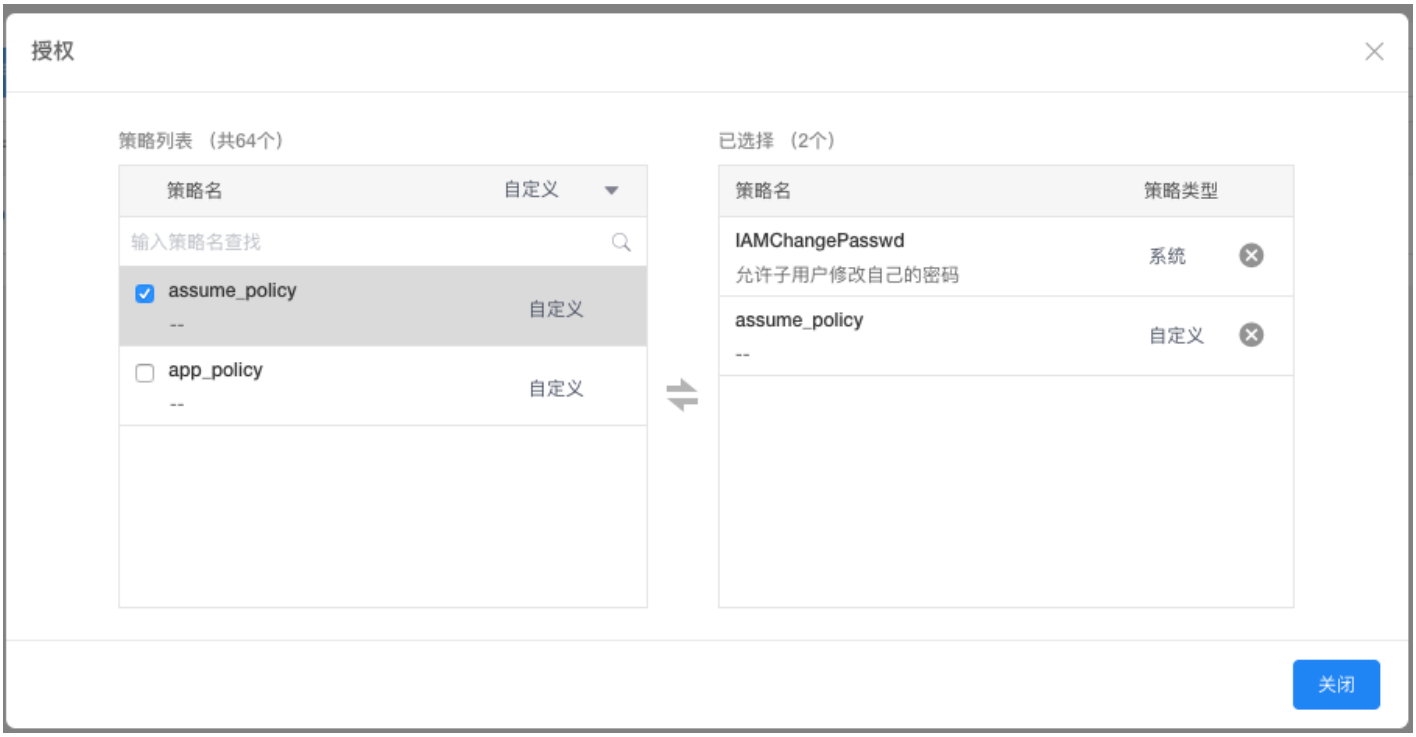
3，在编辑页面中的Resource填入app_role的KRN。

```
{
    "Version": "2015-11-01",
    "Statement": [
    {
    "Sid": "Stmt15253327178180",
    "Effect": "Allow",
    "Action": ["sts:AssumeRole"],
    "Resource": ["krn:ksc:iam::Account_ID:role/app_role"]
    }
    ]
}
```

策略名称记录为“assume_policy”。

4，进入【用户管理】页面，创建一个IAM子用户“app_user”，勾选“为该用户生成AccessK ey”

5，完成之后即可在用户管理里面给app_user添加上assume_policy的自定义策略。



使用STS授权访问

1，使用app_user的AKSK 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=2015-11-01
&RoleSessionName=Bob
&RoleKrn=krn:ksc:iam::Account_ID:role/app_role
&AUTHPARAMS
```

返回的内容如下：

```
<AssumeRoleResponse>
  <AssumeRoleResult>
      <Credentials>
        <SecretAccessKey>wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY</SecretAccessKey>
        <Expiration>2018-05-15T23:28:33.359Z</Expiration>
        <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
      </Credentials>
      <AssumedRoleUser>
         <Krn>krn:ksc:iam::Account_ID:role/app_role</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和SecretAccessKey，然后通过KS3 API或KS3 SDK去访问相应的资源。

跨域资源访问

跨域资源访问

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

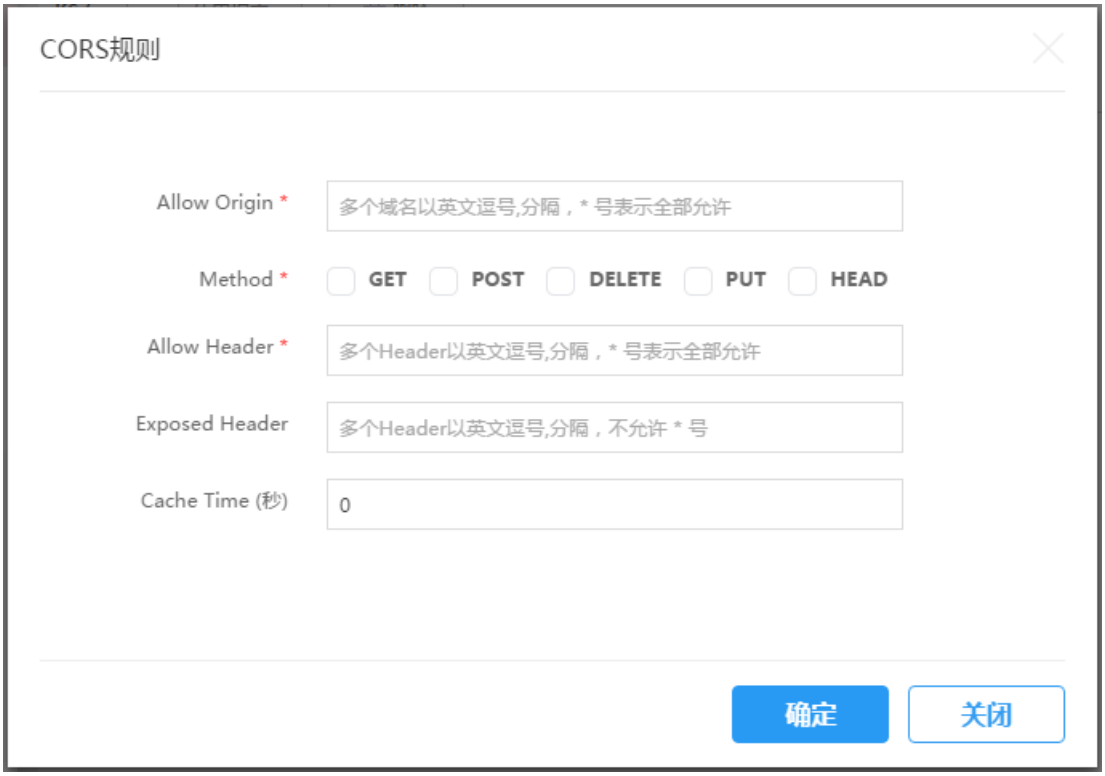
如果需要使用浏览器跨域访问资源，可以登录控制台，设置存储桶的相关CORS解决跨域访问问题。

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的各项参数，如下图：



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CROS相关操作请点击此处。

防盗链

存储空间维护

防盗链

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。

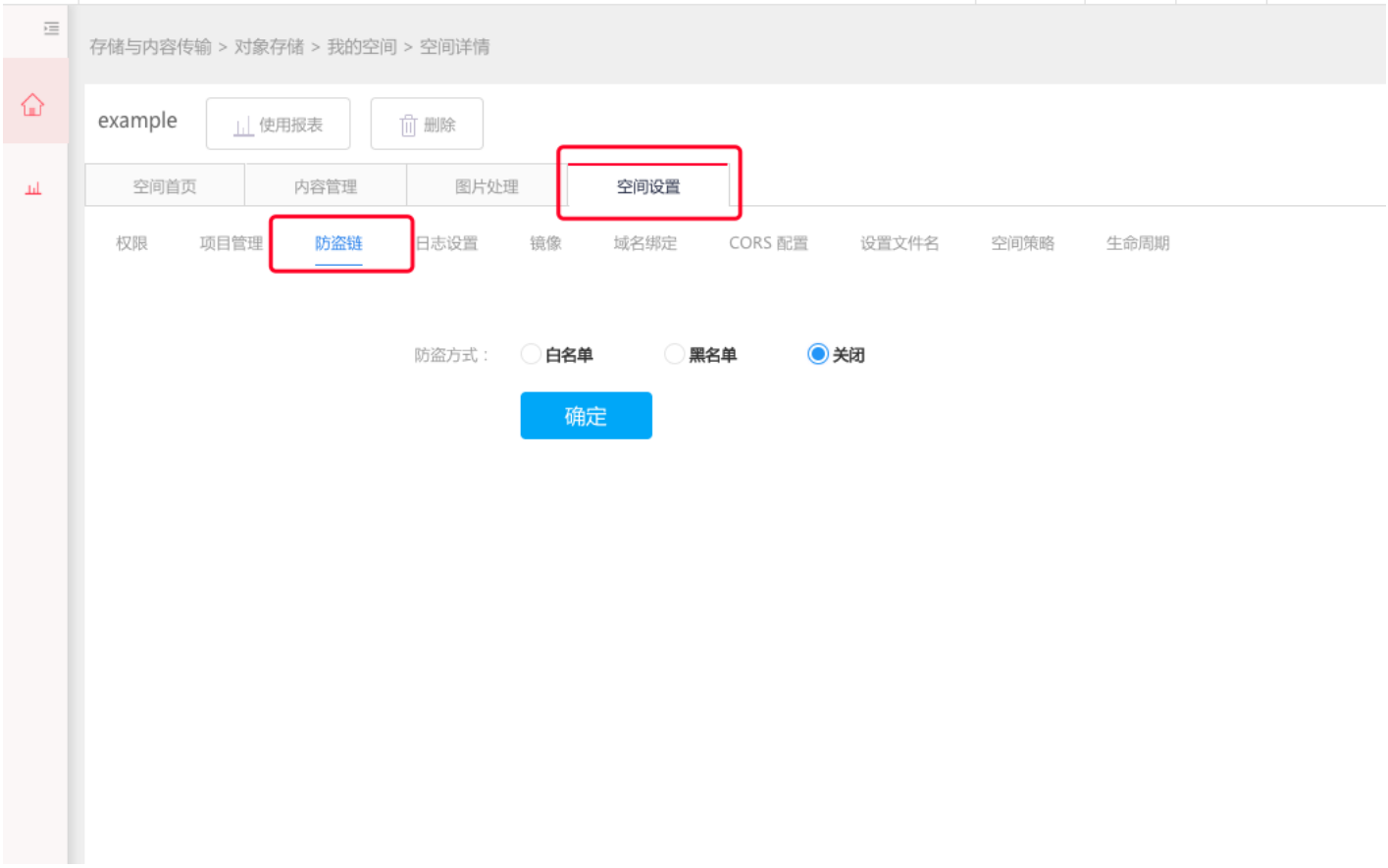
录入域名的时候需要注意以下规则：

- 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

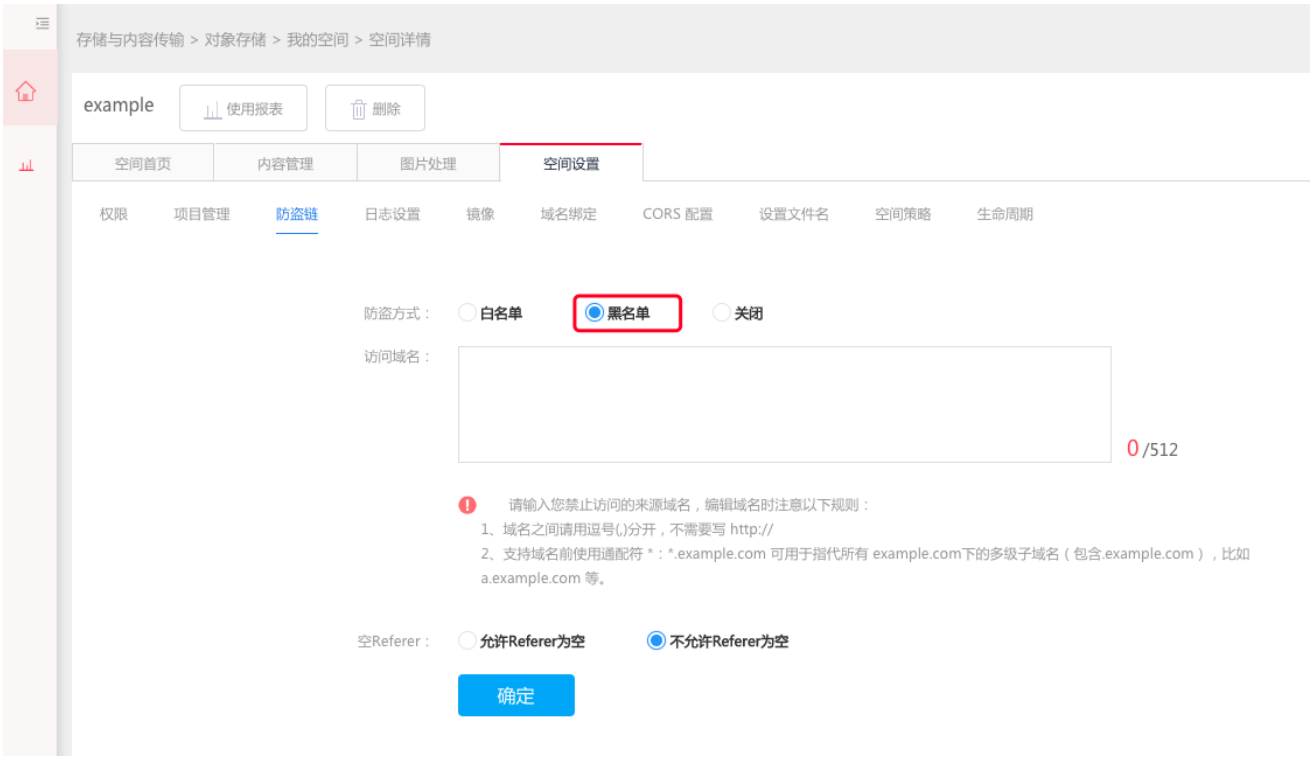
由于有些合法的请求是不会带referer来源头部的,所以有时候不能拒绝来源头部（referer）为空的请求，在KS3中，我们还提供了手动设置referer的选项。

操作步骤：

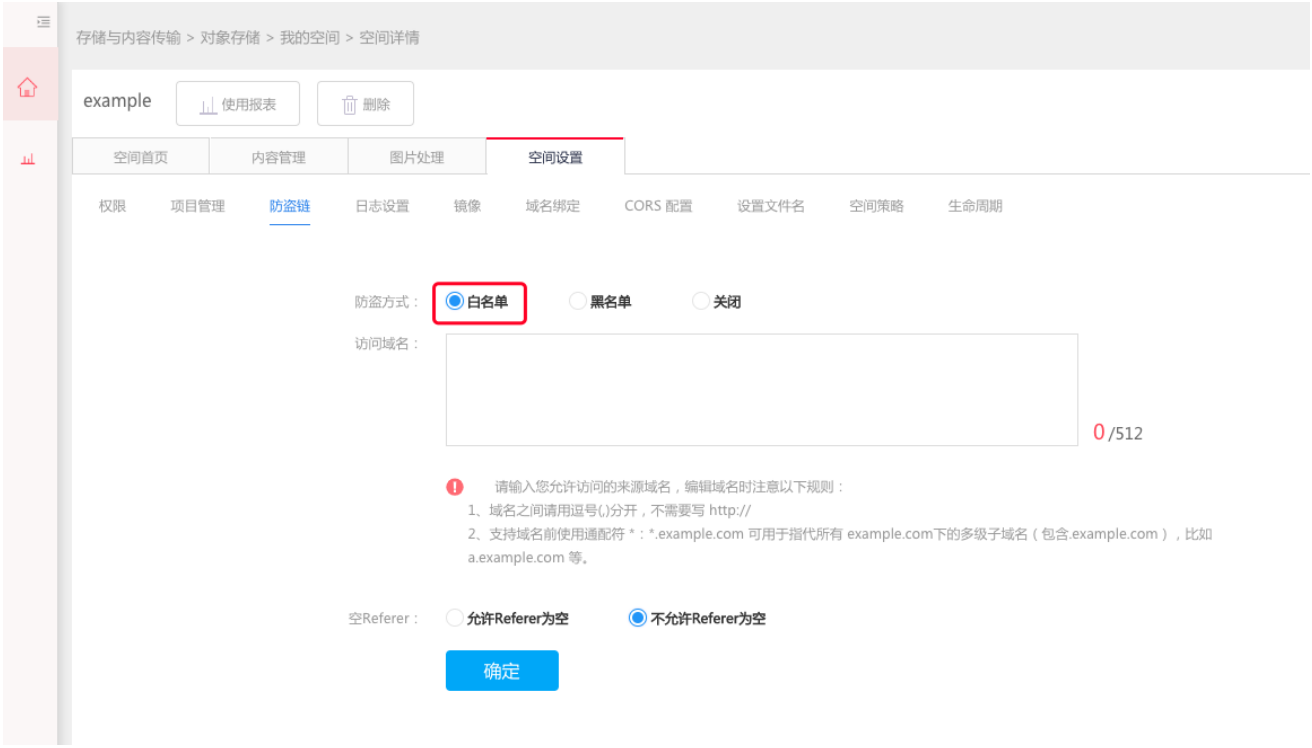
- 防盗链功能位于空间设置菜单下



- 设置黑白名单：
  - 黑名单设置：



  - 白名单设置：



- 手动设置referer